

Product Documentation

User Manual

Deyel 9.2

CONTENTS

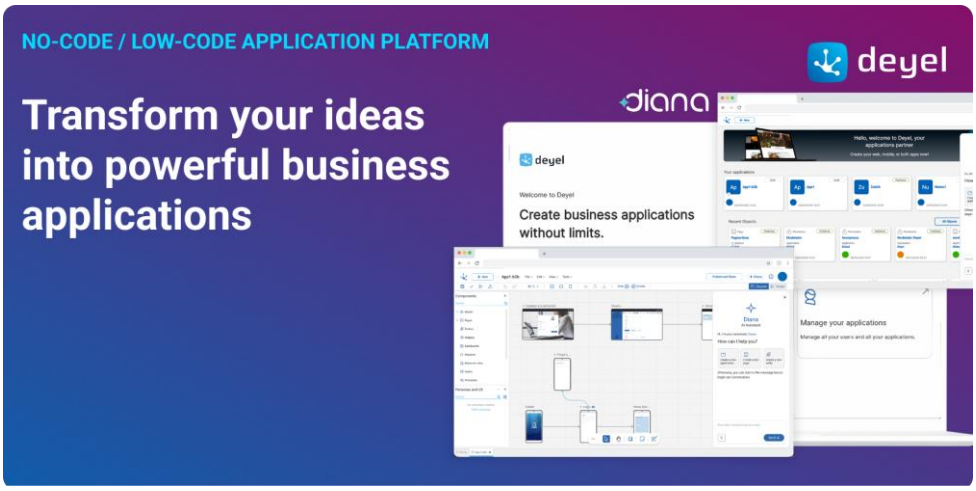
1. Welcome to Deyel 5
2. Getting to know Deyel 5
   - 2.1. Low-Code and No-Code Platform 6
   - 2.2. Implement in the Cloud or on a Dedicated Server 7
   - 2.3. What types of Applications can be Created? 7
   - 2.4. How to Create Deyel Applications? 8
   - 2.5. How are Deyel Applications Composed? 9
   - 2.6. Modelers and Tools 11
   - 2.7. Diana, Deyel's AI Assistant 15
3. Platform Manual 16
   - 3.1. Architecture 17
   - 3.2. General Concepts 20
   - 3.3. Platform access 27
   - 3.4. Modeler 28
     - 3.4.1. Diana 30
     - 3.4.2. Initial View 52
     - 3.4.3. Advanced View 55
     - 3.4.4. User Preferences 56
     - 3.4.5. Objects Modeling 57
     - 3.4.6. Objects Grid 60
     - 3.4.7. Object States 70
     - 3.4.8. Deyel Bot 71
     - 3.4.9. Applications Modeling 73
     - 3.4.10. Processes Modeling 152
     - 3.4.11. Entities Modeling 290
     - 3.4.12. Rules Modeling 735
     - 3.4.13. Pages Modeling 968

3.4.14. Dashboards Modeling 1412
3.4.15. Widgets Modeling 1422
3.4.16. Value Lists Modeling 1463
3.4.17. Reports Modeling 1475
3.4.18. Publication History 1490
3.5. Admin 1492
3.5.1. Applications 1493
3.5.2. Environment 1539
3.5.3. Security 1622
3.6. Integration 1665
3.6.1. Deyel Rest API 1667
3.7. Business Social Network - Tedis 1686
3.7.1. Chats 1687
3.7.2. Conversations Associated with Objects 1690
3.8. BAM and Process Analysis 1695
3.8.1. Process Analysis - Process Behavior 1696
3.8.2. Process Analysis - Monthly Duration 1697
3.8.3. Process Analysis - Process Activity 1697
3.8.4. Process Analysis - Activation Ranking 1698
3.8.5. Process Analysis - Case Details 1699
3.8.6. Activity Analysis - Activity Behavior 1700
3.8.7. Process BAM - Load Analysis 1700
3.8.8. Process BAM - Work in Progress 1701
3.8.9. Activity BAM - Load Analysis 1702
3.8.10. Activity BAM - Work in progress 1703
3.9. Configuration 1703
3.9.1. Security 1704
3.9.2. Calendars Definition 1731
3.10. Requirements 1739
3.11. Cloud Service Operation 1740
3.11.1. Backups 1743

3.11.2. Disaster Recovery Plan (DRP) 1743
3.11.3. Types of Deployments 1747
3.12. On-Premise Installation 1755
3.12.1. Directories Structure 1760
3.12.2. Installation in Apache Tomcat 1761
3.12.3. Installation in Docker 1767
3.12.4. Installation on AWS 1768
3.12.5. High Availability 1775
3.12.6. Amazon S3 Adapter 1776
3.13. Download PDF 1780

1. Welcome to Deyel

Introduction to Deyel



2. Getting to know Deyel

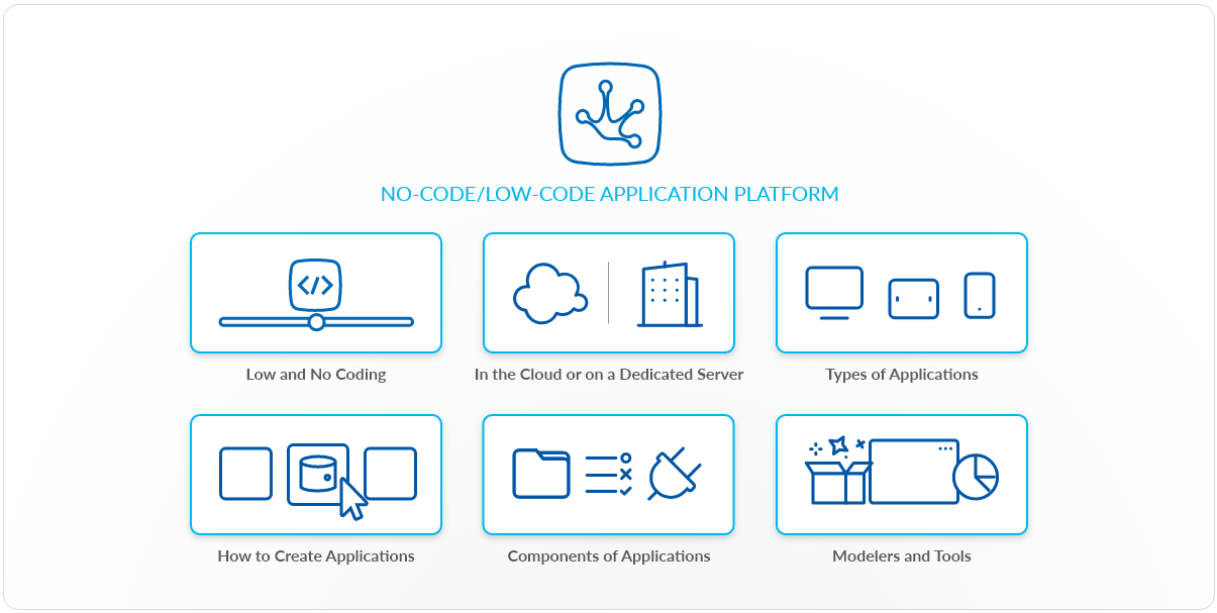
Introduction to Deyel

Deyel is a low-code and no-code platform for creating applications easily, at low cost, with agility, speed, and scalability. The adoption rate of this type of technology as a different way of creating applications is high. They are geared towards business users, not only for no-code users but also to those users who have coding skills.

Citizen Developers are non-technical application creators who can model many types of applications. From web and mobile applications to internal or external applications for medium-sized or mission-critical companies.

Furthermore, the platform incorporates **Diana**, its AI multi-agent that can create low-code enterprise applications. **Diana** enables conversational development, meaning that users can prototype ideas in minutes by describing the application they need in natural language, and the artificial intelligence assistant helps model the solution.

This integration leverages the agility of artificial intelligence for the rapid start of projects, while the maintenance, the governance, and the evolution of the application can be performed using the platform's low-code tools, ensuring the sustainability, traceability, and security of the solutions.



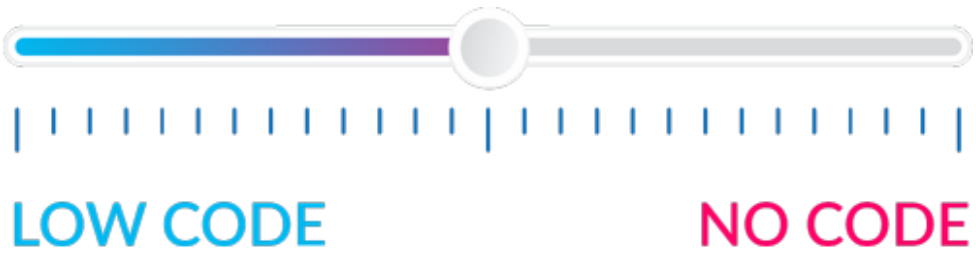
Below are summarized the main features and functionalities of the platform of **Deyel**.

- Low-Code and No-Code Platform
- Implement in the Cloud or on a Dedicated Server.
- What types of Applications can be Created?
- How to Create Deyel Applications?
- How are Deyel Applications Composed?
- Modelers and Tools
- Diana, Deyel's AI Assistant

2.1. Low-Code and No-Code Platform

A low-code application platform is a platform that allows applications to be developed and executed by abstracting and minimizing the use of programming languages.

They are aimed at technically-minded business users and developers who must go beyond standard solutions to create more complex and sophisticated business processes.



These platforms offer a variety of key features:

- They provide the ability to develop a complete application with web and mobile user interfaces, business logic, and data storage.

- They have a development approach oriented towards model-driven or UX/UI methodologies.
- They provide a dedicated execution environment for the developed applications, ensuring their optimal functioning and availability for end users.

On the other hand, a platform defined as no-code (No-Code Application Platform) corresponds to one that allows software to be created without coding.

They are aimed at people without technical knowledge or experience. This is because the tools use an intuitive, feature-rich interface, such as the drag-and-drop functionality, ready-made templates, and pre-built components that allow software creation visually.

2.2. Implement in the Cloud or on a Dedicated Server

Deyel Cloud allows the development, testing, deployment, and execution of web and mobile applications without worrying about the operational aspects of the platform's technology, ensuring high availability.

Modeled and deployed applications are hosted directly on the cloud infrastructure provided by Deyel Cloud, based on Amazon Web Services (AWS) regions, availability zones, and data centers distributed around the world. The infrastructure of Deyel Cloud and its applications can be deployed in any AWS region globally, across different availability zones.



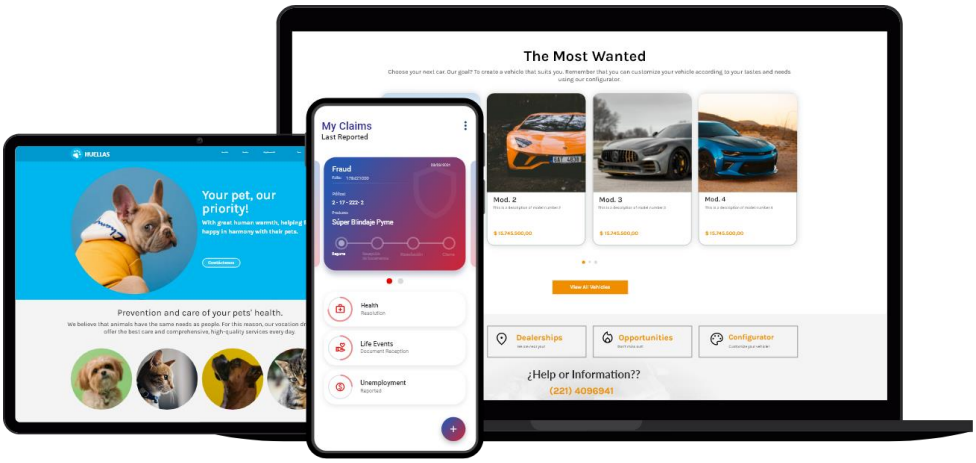
If the business prefers, an On-Premise Installation can be chosen, with the same capabilities and advantages, allowing it to integrate quickly into the existing technological platform and operate in its own environment.

2.3. What types of Applications can be Created?

With **Deyel** , it is possible to develop a wide variety of applications. Web applications can be developed to be accessed from desktop computers or mobile devices, as well as mobile applications that

can be installed on cell phones or tablets. Responsive applications that operate efficiently in both web and mobile environments can also be created.

Mobile applications are compatible with both Android and iOS operating systems, allowing the creation of specific applications for each platform or developing them as Progressive Web Applications (PWA).



The applications developed can be:

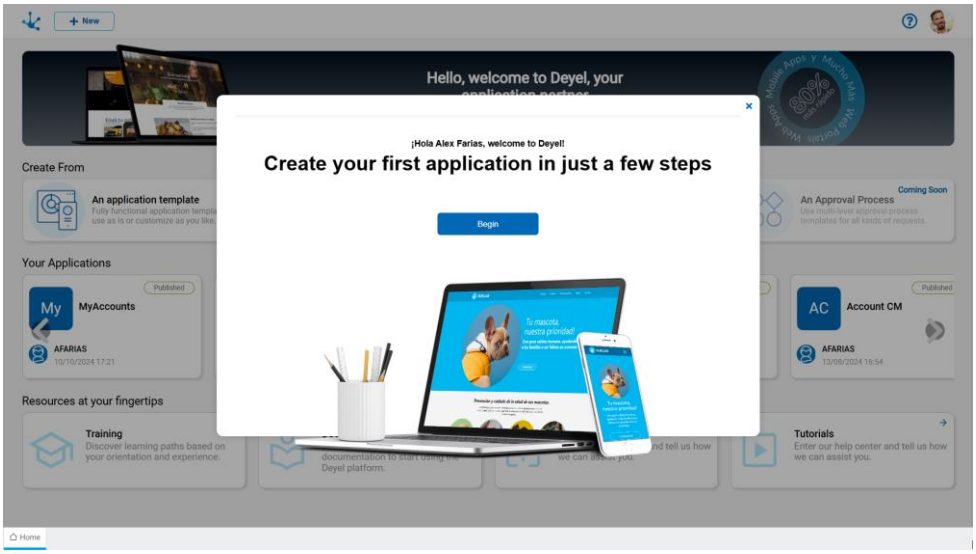
- B2B (Business to Business) Applications that establish commercial relationships between companies, as well as tools that can be used internally within an organization.

- B2C (Business to Customers) Applications designed for the end consumer, which must have an attractive and functional interface, which motivates the user to choose to use them.

2.4. How to Create Deyel Applications?

For creating applications with **Deyel** in an agile way and without the need for prior knowledge, it is possible to start the modeling from different bases.

Once an adequate level of experience in modeling is reached, it is possible to create applications from scratch. Alternatively, by understanding the existing methods, it is possible to continue using pathways based on foundations, which are generally much more efficient and allow for the direct creation of an application, focusing on its specific features, rather than on aspects that have been resolved through patterns and common components applicable to all applications. In this context, one can choose to use application development methodologies based on the UX/UI design process, or the model-driven methodology.



From Application Templates

By choosing to use application templates, access to fully functional solutions that cover different topics and sectors. These templates can be used as they are or customized to fit specific needs.

Based on Available Data

If starting with data, it is possible to design entities that represent the required information, using entity templates, Excel spreadsheets, or asking artificial intelligence to create models adapted to the requirements.

Based on the Desired User Interface

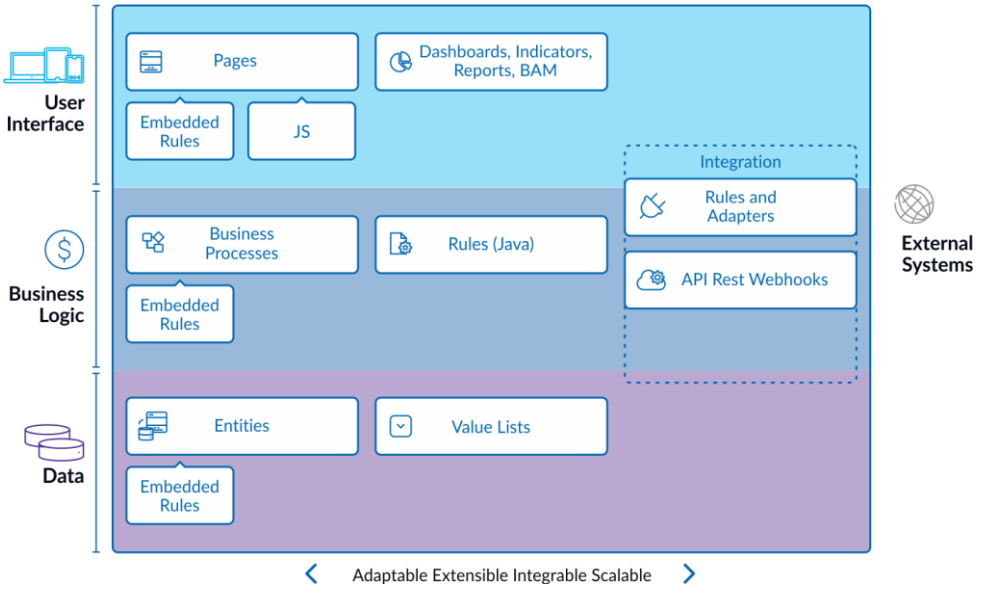
If starting with the user interface, page templates are available to facilitate a quick start to modeling the application, taking advantage of the experience that the platform offers in application development. Additionally, it is possible to follow the UX/UI design process through the applications modeler, which allows structuring workflows and creating high-definition prototypes.

Based on Approval Process Templates

Finally, for the development of applications that resolve approval processes of any kind, specific templates can be used to facilitate the management of the required applications.

2.5. How are Deyel Applications Composed?

In **Deyel** , applications are made up of different types of objects and can be classified into the groups shown in the following image.



User Interface

For the user interface, page-type objects are mainly used. These allow the modeling of the required interface, whether for mobile phones, tablets, or desktop computers, always with pixel-perfect precision. Other objects complete the user interface of applications such as management widgets, widget dashboards, application reports, and all reports related to monitoring activities and business processes.

Business Logic

At the business logic level, the main objects are business processes and rules. Processes allow for the modeling of human or automatic activities and their relationship with those responsible for their execution. Any type of behavior across the business processes can be modeled using BPMN 2.0 notation. Additionally, through advanced rules, specific behavior can be defined

Data Persistence

At the data persistence or data model level, entities are used that represent the business objects and their persistence in the database. Value lists are also used, which are much lighter objects that allow representing the values used by applications both in the interface and in the entities.

Other Objects

At all three levels, behavior can be added using rules embedded in objects. These rules have a very simple syntax that does not require programming knowledge, and its complexity is similar to an Excel formula. Through these rules, conditions of visibility, editability, requirement, validations and calculations can be defined in all objects, whether in the interface, in the business logic or the data model.

These embedded rules allow adding behavior without the need to code since the wizard makes it even easier in addition to having a simple syntax.
Beyond this, if it is necessary to do something specific at the interface layer, JavaScript code can be added to the pages, using the Deyel SDK. If required at the business logic level, Java-based rules can

be developed using the Deyel SDK, which allows viewing the different objects in the application as if they were Java objects.

Through adapters and integration rules, applications developed with **Deyel** can be integrated with other applications. Adapters define the coordinates and credentials used to integrate with the other application. Then the rules based on these adapters will be able to interact with the APIs or other integration mechanisms provided by the external applications, whether through a database, web services, or some other type of event or integration mechanism. On the other hand, applications that are developed with **Deyel** automatically generate a Rest API with which any developed objects can be invoked from other applications. When working with REST APIs, there is also the possibility of using webhooks to achieve a more robust integration with external applications.

It is important to highlight that applications developed with **Deyel** are adaptable and extensible, using the modelers of each object, they are integrable through the integration facilities provided by the platform. They are scalable, since through the horizontal scaling provided by the platform, an application can start with a few users and scale almost unlimitedly.

2.6. Modelers and Tools

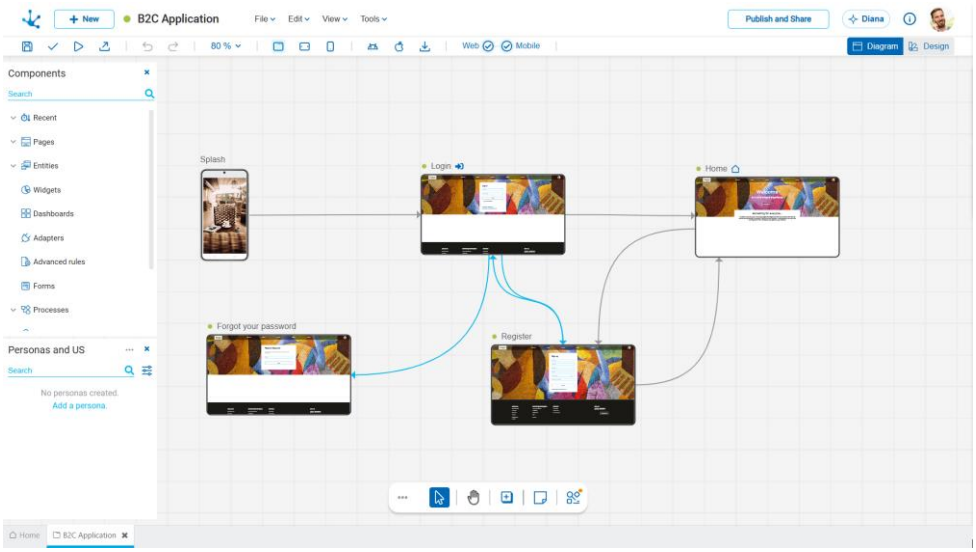
For the development of applications, **Deyel** allows using a modeler, which allows the definition and maintenance of the different objects that comprise them.

Modelers

Each type of object uses a specific modeler, which has its own features, such as the graphic modeling area, expanded menus, top and side toolbars, properties panels and design wizards. When creating a new object in a modeler or opening a previously defined one, a design space can be accessed where, once the modeling is finished, the object can be published and becomes available for use in applications.

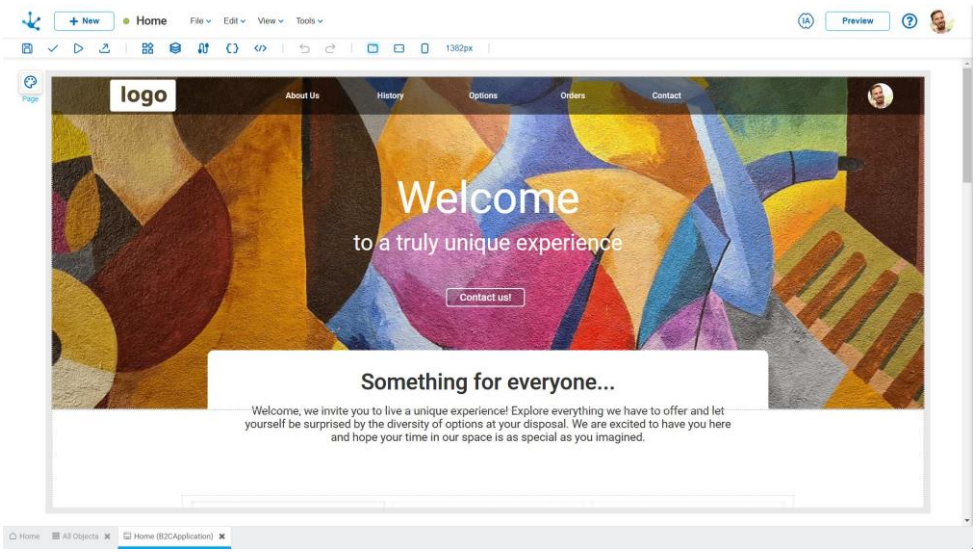
Applications Modeler

The applications modeler allows to easily and intuitively model applications, to design their navigation flows and the pages that implement the application interface. In addition, notes and comments can be added. It is modeled with the appearance of the user interface, at each of the implemented breakpoints. Both mobile and web applications can be generated from the same modeler.



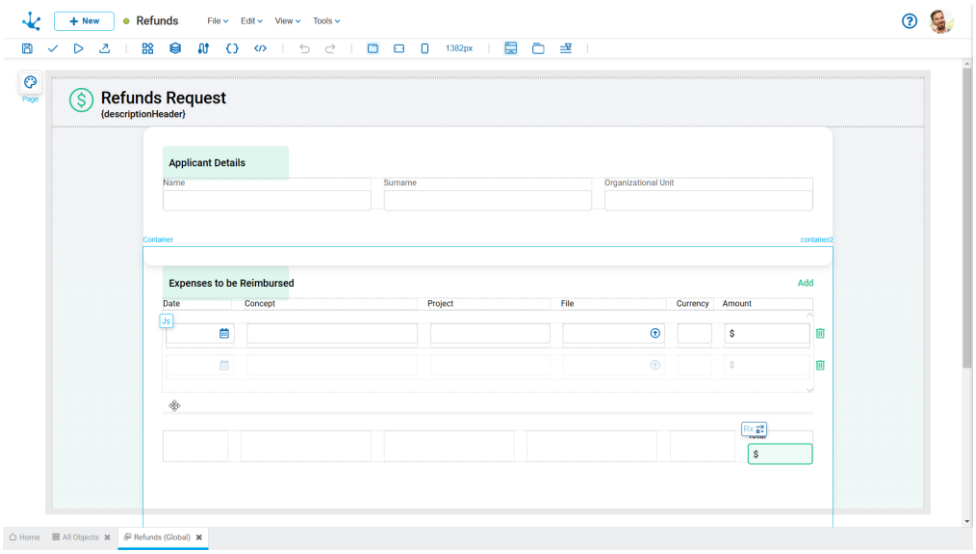
Pages Modeler

The pages modeler facilitates the creation of fully responsive, pixel-perfect pages that automatically adjust to the screen or device they are being viewed on, ensuring the best user experience at all times.



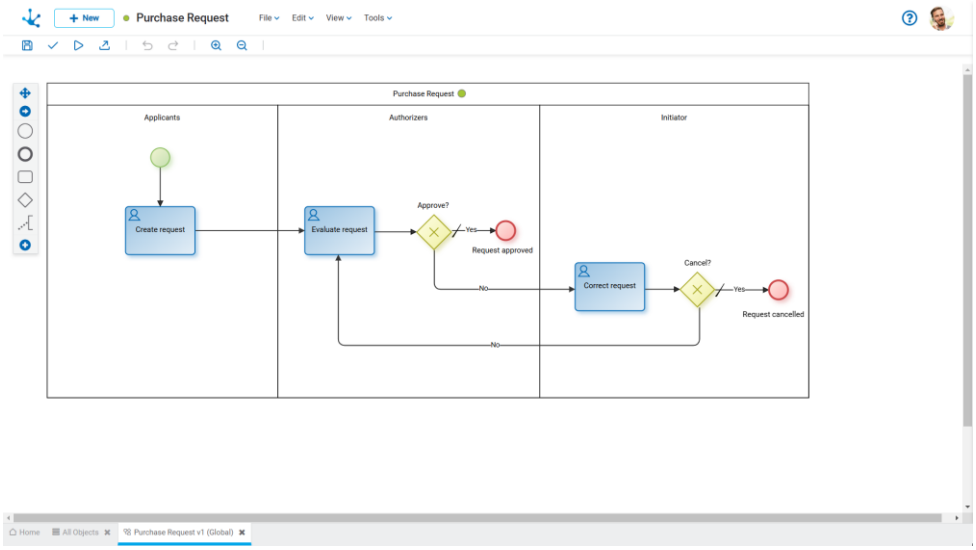
Entities Modeler

Through the use of the entities modeler, it is possible to represent entities as they are perceived in the real world by dragging and dropping elements that define their appearance. Each of these elements that represent the attributes are defined by a field within the entity and each of them can have their behavior defined.



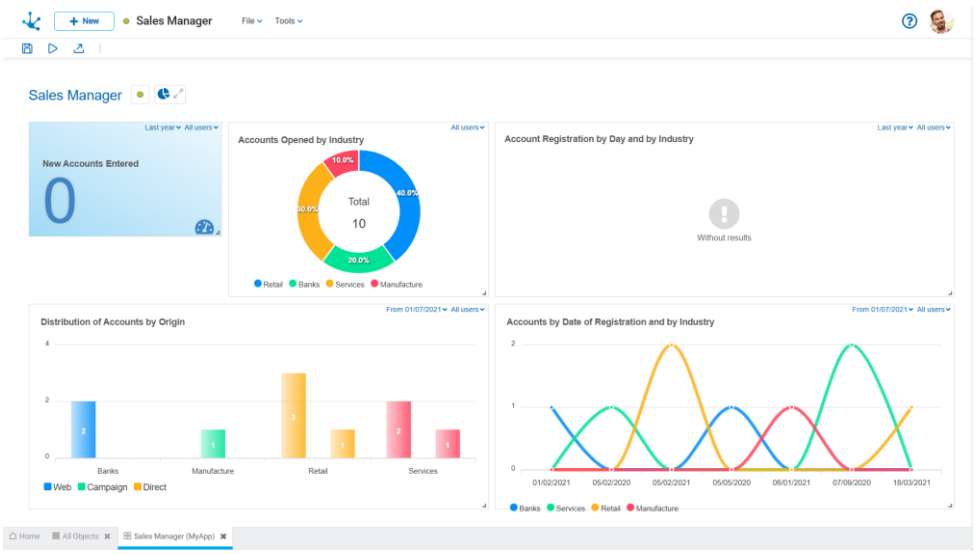
Processes Modeler

The processes modeler allows designing, displaying and improving business processes, creating visual representations of them, using standardized symbols of BPMN 2.0 (Business Process Model and Notation).



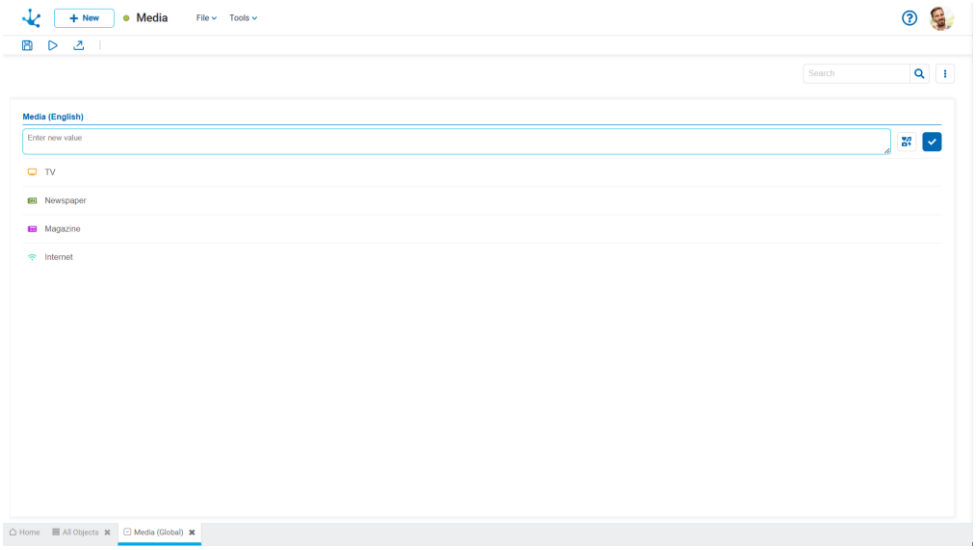
Dashboards, Widgets and Reports Modelers

The widgets modeler allows to intuitively and easily design and implement graphics that display business information, while the dashboards modeler simplifies the design of the interface to display such graphics. The presentation of business information can also be designed very easily in the format of reports.



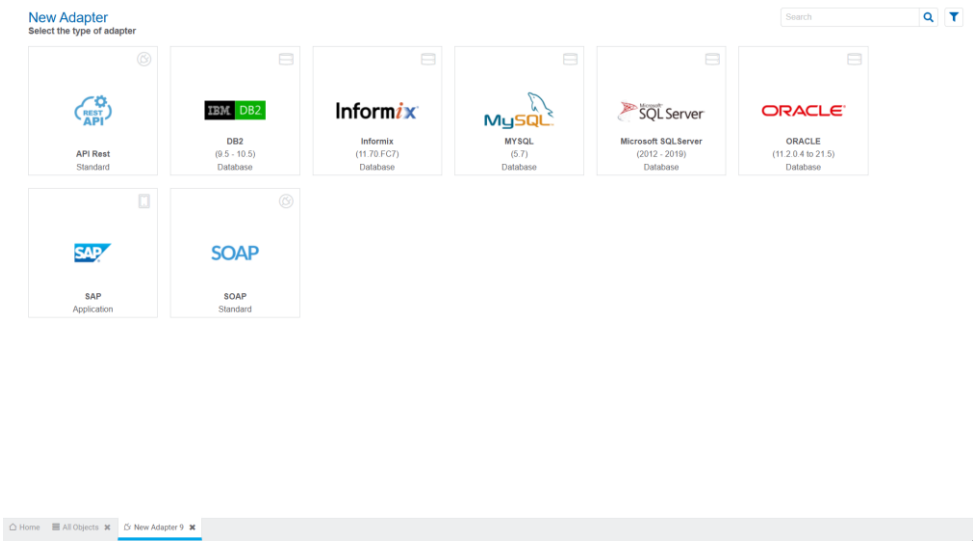
Value Lists Modeler

The predefined option sets that users can select when interacting with entity fields in the applications can be defined and managed by using the value lists modeler.



Adapters and Advanced Rules Modelers

The advanced rules modeler is aimed at IT developers, where they can generate the code necessary to extend the functionality of the modeled applications, while the adapters modeler allows configuring specific properties for communication of advanced rules, both within the same platform and with external applications.



Tools

Deyel also provides tools that allow managing applications throughout their life cycle.

- Security tools
  They allow the administration of users, organizational units, job positions, roles and application access permissions.

- Application Building and Installation Tools
  Facilitates the implementation of new applications and the installation of changes and modifications to existing applications.

- Execution Console
  By using it, citizen developers can review the execution of their applications, by analyzing error indicators, warnings, and using a grid that displays the actions performed when executing each of the objects.

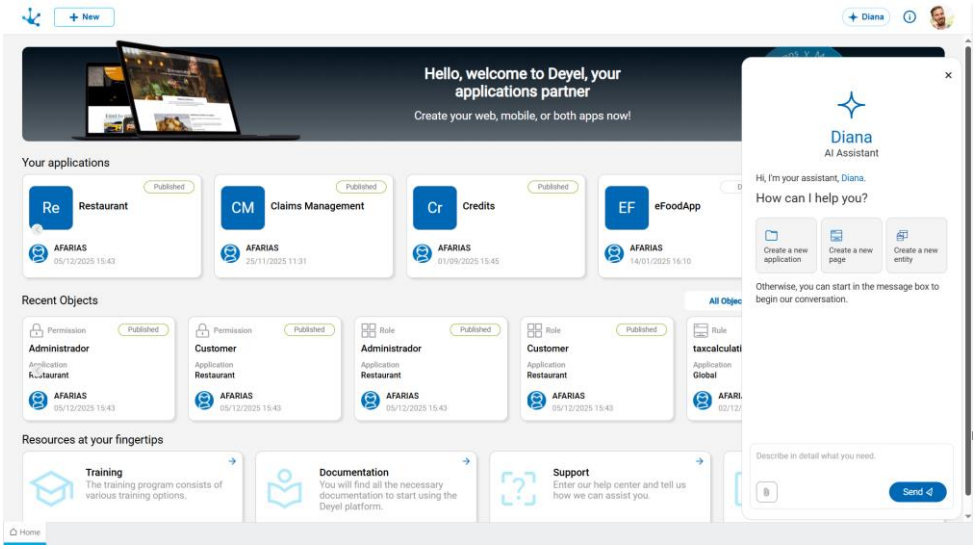
2.7. Diana, Deyel's AI Assistant

Diana is the team of artificial intelligence agents of **Deyel**, designed to accelerate low-code and no-code applications development through conversational prompts.

It brings together specialized agents that assist in modeling an application, defining its personas, user stories and user flows, generating pages, defining entities and creating advanced and integration rules. It also includes an expert agent that teaches how to use the platform using the product documentation.

The primary role of **Diana** is to teach, simplify, guide, and automate the development process using natural language. Through prompts, it allows for the rapid generation of a basic application structure, which can then be adjusted and customized using the Deyel modeler.

Diana acts as an orchestrator for the specialized agents, activating the one that best fits the modeler user's needs at any given moment. Although the interaction takes place through a single chat, the conversation is directed by the agent best suited to handle the requested task.



3. Platform Manual



Architecture



General Concepts



Modeler



Admin



Diana



Integration











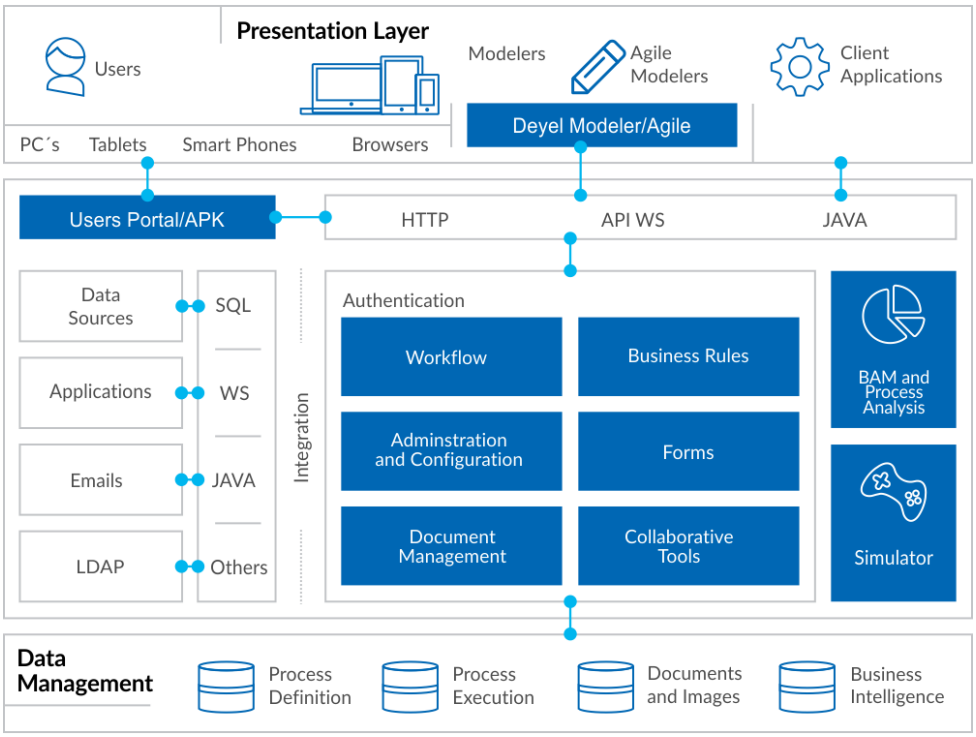
Tedis Business Social Network

BAM and Process Analysis

3.1. Architecture

Deyel is a product based on Java technology, developed with modern object-oriented patterns. It supports several industry standards such as HTML5, CSS3, RWD, web services, multiple database and application servers.

The following chart describes the main components of its architecture.



Graphic Modeler

Using the graphic modeler, business users can graphically design their processes using **BPMN 2.0** (Business Process Model and Notation) symbols in an intuitive and easy-to-use environment. Each process with its activities, interconnections, events, times, alerts, etc. can be defined and also documented. In this same workspace, aspects of processes implementation can be included, such as user interfaces of the activities, integration with other applications, variables and business rules, among others.

User Portal

Through the portal of **Deyel**, users can interact with their list of activities and execute their tasks, start cases of their processes, consult them, access collaborative tools, monitoring reports, etc. The administration and configuration functionality can also be used through the portal. The portal is based on standards such as RWD, HTML5 and CSS3. They can be used on mobile devices or desktop PCs through web browsers. Its appearance is configurable according to the characteristics of each organization.

Workflow Engine

The workflow engine of **Deyel** interprets and executes the definition of business processes that are modeled in the graphic modeler. The modeler stores the graphically modeled processes within the process repository, so that the workflow engine can interpret these definitions and execute them. The execution state remains in the case execution repository.

Forms Administrator

Through this **Deyel** tool, forms that represent the user interface of human activities are generated in the modeled processes. Forms contain the process variables that are used by business rules and other related objects.

Forms have an interface rich in components (jQuery and Bootstrap), support RWD (Responsive Web Design) concept and are based on standards such as HTML5 and CSS3.

Business Rules

The behavior of processes is defined by business rules. Rules can be used in validations, specific business logic, process flow control, integration with other applications, etc.

Deyel offers wizards the business user that allow them to intuitively define the behavior of the rule. These wizards present the user with control structures, access to available parameters and variables, previously defined business rules, executing components, logical operators, etc. Rules persist in the object catalog of **Deyel.**

Collaborative Tools

With the messenger service of **Deyel**, users can communicate easily. Messages can be used among users in private, or they can be public when they are associated with cases or processes definitions. Associating them with cases provides a collaborative tool among the participants of the case, making these messages part of the case. On the other hand, messages related to the definition of processes enable a direct communication channel to improve processes among the participants of the case and process owners. Also, subscriptions enable an option to choose to receive messages about cases or processes of interest.

Integration

Deyel provides tools and services that allow any application to interact in a bi-directional way with the defined processes. Making use of adapters based on Web Services, Java, JDBC and others, allows to integrate existing applications or data sources.

Deyel also has Java APIs and web services that allow any application to interact with processes through them.

BAM and Process Analysis

Deyel offers powerful capabilities for analysis and browsing of multidimensional structures that allow to analyze the operation of processes and their tasks, both historically (Process Analysis) and in real time (BAM - Business Activity Monitoring).

Through predefined reports, it is possible to observe and understand the operation of processes and detect possible improvements. They analyze in brief the behavior of processes and their tasks, being able to identify what was done on time, with delay, deviations against maximum and expected durations, trends, participant performance, bottlenecks, etc.

The behavioral information is displayed as a grid and through charts, always having the reference of the expected trend curves to contrast against reality. In this way, behavior can be graphically monitored. In all cases, it is possible to drill down into the information, reaching the process level, task, executor, date range, case, etc. Multiple views of the information can be generated by navigating it through multiple dimensions, such as time, priorities, participants, initiators, states of cases and tasks, among others.

Administration and Configuration

The administration and configuration of **Deyel** allows to define the security and authentication of process participants with their different profiles and roles, their relation with LDAP repositories, workgroups, their districts, calendars, time zones, user portal appearance and also technical aspects of the solution.

3.2. General Concepts

To develop an application with **Deyel** , a set of concepts is used to allow easy and intuitive modeling of each of its objects.

Application

A **Deyel** application is a set of processes, business entities, pages, forms, value lists, advanced rules, adapters, roles, permissions, dashboards, widgets and reports, related to each other with the objective of solving a specific request. These objects can be defined within the same application or shared from other applications.

Applications developed with **Deyel** can be web or mobile (Android and IOs). In both cases, the applications can be developed responsively to adapt to the different devices that use them.

Through the applications modeler, independent applications can be easily and intuitively modeled. The navigation flows of these applications can be designed through the modeling of their user flows, wireflows and the pages that implement the application interface.

Applications can be created from scratch or based on predefined templates. Using templates helps to create applications more quickly using patterns and common functionalities found in business applications.

Entity

An entity or business entity is a representation of a real-world concept in an application. For example, an entity can represent a product, a customer, or an order. Entities have attributes that describe them, such as their name, price, or creation date. Entities store application data within themselves.

Business entities can be modeled by citizen developers either in a classic tabular form or graphically, without the need for technical knowledge. Through the entity modeler, entities can be modeled as they are perceived in the real world, by dragging and dropping elements that define their appearance. These elements have properties configured with default values, so that citizen developers do not need to configure technical properties, but it allows IT developers to customize their operation in detail.

Relations can be established between different entities, whether they are business entities specific to the application or parameterization entities such as value lists. Relations with other objects can also be established through integration rules.

Entities, through their modeling, automatically generate:

- The persistence structure in the database.
- The user interface for maintaining their data and also to be used in business process activities.
- A Rest API so that the entity can be used by other applications.

Scope

Entities can be used in the application in which they were defined or in other applications as well.

Attribute

An attribute is the information unit within an entity.

Page

Pages define the UI of **Deyel** applications. Through page modeling, **Deyel** applications achieve a perfect UX. Pages show, create, and edit application entity information and also allow users to invoke automations through business processes and rules.

Pages are made up of one or more graphic elements.

Scope

Pages can be used in the application in which they were defined or in other applications as well.

Types

- Modeled Pages
  Through the page modeler, pixel-perfect user interfaces can be obtained by defining the behavior of each element at any of the proposed breakpoints.

- Built-in Pages

Built-in pages in the platform are pages that are commonly used in all applications and that are already available to be used by any application without coding.

Examples

- Login pages, registration pages
- Forgot my password
- CRUD and forms grid
- Dashboards
- Chats
- To do list
- My cases
- Calendars
- User profile

Graphic Element

Graphic elements are the graphic interface components of **Deyel** pages. There are structural elements (section, container, layout, repeater, slider, page), simple ones (text, button, image, field, iframe, video, icon, widget) and advanced ones (process buttons, relation navigation, operation buttons, header).

Process

A process is a sequence of human or automatic tasks (services), the performance of which allows meeting a business objective. The sequence has a start, an end, and may contain connectors. It is represented as a process diagram based on the standard of **BPMN 2.0** (Business Process Model and Notation).

Scope

Processes can be used in the application in which they were defined or in other applications as well.

Form

A form is the user interface of business entities. It can be used in user activities of processes or in the data administration of the entity itself.

Business entities of applications can be modeled graphically by citizen developers without the need for technical knowledge. Through the form modeler, business entities of applications can be modeled as they are perceived in the real world, by dragging and dropping controls that define their appearance. Controls have properties set to default values so that citizen developers do not need to set technical properties but allows IT developers to adjust their behavior in detail. Relations among different entities can be established, whether they are business entities of the application, parameterization entities such as value lists or with other objects through integration rules.

The forms modeler together with the engine of **Deyel**, are in charge of solving the data model persistence generated by business entities of applications.

Scope

Forms can be used in the application in which they were defined or in other applications as well.

Examples

- The "Vacation Request" form is used in the "Vacation" process.
- The "Employee" form is used to update data stored in its fields during its life cycle. In addition, such form fields can be used from different processes related to it that contain user activities.

Control

A control allows to define the appearance of a form field in the user interface. In addition, controlling may add default behavior and persistence characteristics to the form field, which can be modeled later.

Through a set of controls, a pattern-based interface design guide can be defined in order to solve common situations and achieve a homogeneous interface.

Forms are created from a gallery of controls, each of them allows defining:

- The appearance of a form field in the user interface.
- The layout of fields and graphic elements within the form.
- Actions to do from the user interface.

Examples

- Text Box
- Number
- Container

Rule

Business rules allow to define the behavior of application objects. Rules can be used in validations, specific business logic, process flow control, integration with other applications, field display control in forms, scheduled tasks, calendars, widgets and agents, among other functionalities.

There are rules that can be modeled without programming and others are based on Java code.

Types

- Embedded Rules
  Embedded rules are used to define the behavior, validations, and calculations of entity fields, as well as page validations. Embedded rules allow to define logical conditions and arithmetic expressions, which are evaluated using entities, pages and their related processes. A wizard integrated in the corresponding modelers is used for definition. This wizard is found in the property definition panels of the fields and containers of the entities.

Scope

Embedded rules belong only to the object that contains them.

Examples

- Define conditions in flows within a process.
- Execute automatic actions in activities of a process.
- Define requirement, visibility, edition, calculations and validations in the fields of an entity or page.

- Advanced Rules
  Advanced rules are based on Java and are used to include complex logic or integrations with other applications. They are developed using the SDK for Java of **Deyel**, both from the advanced rules modeler included in **Deyel** as in any Java IDE to be used.

  Scope

  Advanced rules are global for all applications.

  Examples

  - Execute automatic actions in activities of a process.
  - Execute scheduled tasks.
  - Execute business logic within the source of an widget.

Function

Software unit that performs a specific task that is executed when invoked from a programming code or from the modeling wizards. It can receive input parameters and return a result.

Value List

A value list is a set of predetermined data used to report and unify criteria when rusing a field. There are options that the user can drop down and select when completing an entity, a page or a form.

Scope

Value lists can be used by the forms, entities, or pages of the application in which they were defined, or also in others.

Examples

- Client Type
- Industry
- State

Widget

Widgets allow to know the business evolution through the use of a graphic representation, based on the execution analysis of applications. The information represented in the form of charts is obtained from application entities or by using advanced rules.

Widgets can be represented, for example, as metrics, as pie charts, column charts, horizontal bar charts, area charts or line charts, among others.

Scope

Widgets can be used in the application in which they were defined or in other applications as well.

Dashboard

A dashboard is a collection of widgets that present relevant information for the business user, with high visual impact.

Scope

Dashboards can be used in the application in which they were defined or in other applications as well.

Types

- Application
  They are modeled through the dashboard modeler by IT modeler users and are defined in the applications.

- User
  They are modeled by business users and can only be used by those who defined them since they are not part of the applications.

Adapter

An adapter allows integration with different applications and platforms by using advanced business rules. Its purpose is to exchange data and share behaviors.

An adapter encapsulates the complexity of communicating with an external component. It allows different advanced rules to use it to invoke the operations that such component exposes.

Scope

Adapters are global for all applications.

Types

- Built-in
  They are adapters defined in **Deyel**, available in all environments from the modeler's grid. There can only be one of each type in each environment.

  Examples

  - Twitter
  - Mercado Libre
  - Deyel SDK
  - Docusign
  - GoogleDrive

- Standard
  These adapters allow access to web services from external providers, in their different communication architectures, such as Rest API and SOAP.

- Database
  Adapters that allow to define access to relational databases through the Java JDBC protocol. **Deyel** provides custom adapters for the most popular database engines on the market, facilitating configuration and optimizing interaction with such engines.

Report

A report contains entity information that is presented with a certain layout. The report can be modeled very simple way by selecting its information and quickly designing its presentation. When modeling it, choose the fields of the entities that are part of the report and of those related entities. The order of these fields can be modeled and by using filters, the information contained in the report can be selected.

Scope

Reports can be used in the application in which they were defined or in other applications as well.

Types

- Application
  They are modeled through the report modeler by IT modeler users and are defined in the applications.

- User
  They are modeled by business users and can only be used by those who defined them since they are not part of the applications.

Use of Concepts

Some considerations regarding the use of concepts described in this topic are described below.

Property

They describe in general terms applications, entities, processes, rules, and functions, among other objects.

Control

The term control is used only in the modeling of the graphic interface of forms.

Field

Entities, pages and forms store their data in a set of fields. In entity, page, and form modelers, a field can be defined based on an attribute from another entity; in this case, it is called a related attribute and can have persistence.

Attribute

It is the smallest unit of information of an entity, it corresponds to the data model.

Variable

A variable corresponds to the attribute of an entity, a page or a form, and is used by business processes to define its state and behavior. It can contain data of the temporary conditions of the instances of the processes (cases), of the business objects or references to the latter.

Scope

- Local: It can only be accessed and modified during the execution of the process case that uses it. Variables of completed cases can be accessed but not modified.
- Global: It can be accessed and modified by the operations of the corresponding form or entity, by the processes that use it or by business rules.

3.3. Platform access

Through this platform, it is possible to create business applications without limits. **Deyel** is designed to develop AI-powered software solutions, specifically tailored to the needs of each organization.

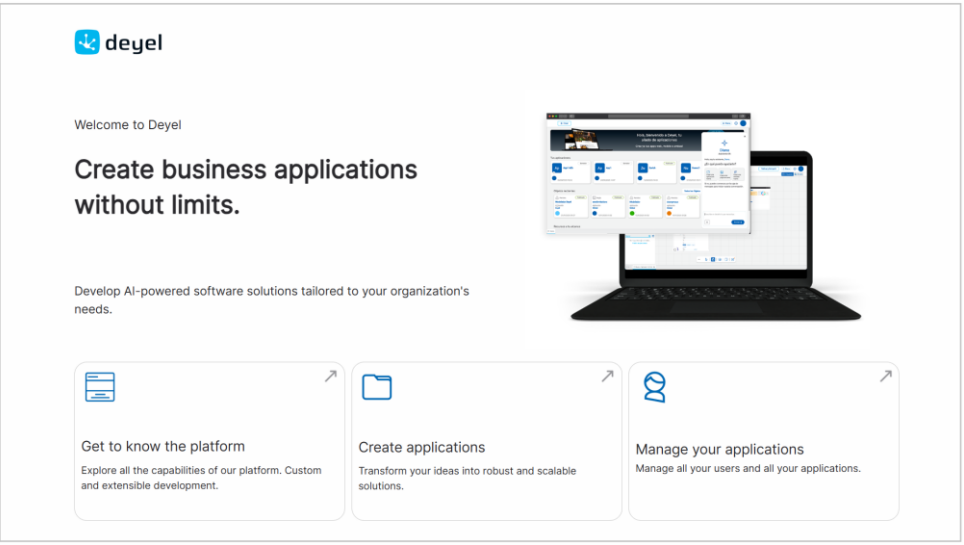
Home

To access the platform, use the URL corresponding to the configured environment, for example "https://myenvironment.deyel.com". This allows immediate access to the main functionalities of the tool.

- **Get to know the platform:** Allows exploring all the capabilities available through the resources of documentation and training. From these resources, it is possible to obtain the information needed to create, manage, and implement applications.

- **Create applications:** through access to the applications modeler, it is possible to transform ideas into robust and scalable solutions, fostering solid business growth. From this module, it is possible to model and manage all the objects that make up an application, such as entities, processes, pages, business rules, dashboards, reports, scheduled tasks, and other necessary elements.

- **Manage your applications:** through the administration console , a comprehensive control of the environment is maintained, allowing users to be managed and the state of all applications to be displayed in one place. From this console, it is possible to monitor the health of the applications, review executed processes, manage scheduled tasks, configure environment properties, and define the necessary security policies.



> *The access URL may display the welcome application or, if an organization application has been published, it will be loaded instead.*

3.4. Modeler

Deyel Modeler

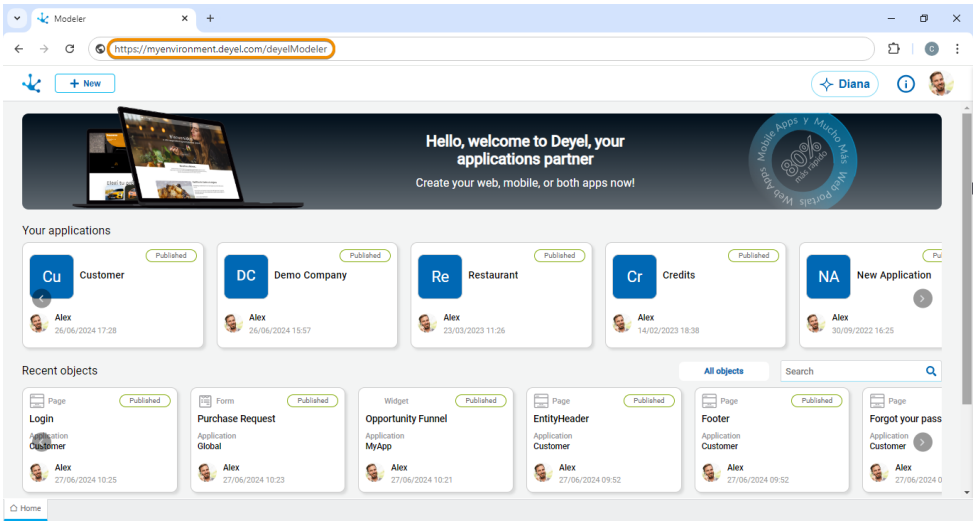
The Deyel modeler is a key component for application development on the platform. It represents the main means of accessing applications, their objects, and the resources necessary for their modeling. An application is the central element of all modeling, as it groups the other **Deyel** objects and its execution enables the resolution of business requirements.

The modeler includes the assistance of **Diana**, the artificial intelligence assistant of **Deyel,** that acts as a co-pilot during modeling. **Diana** guides the modeler user, interpreting requests in natural language and assisting in the generation and extension of objects, whether it is a complete application or individual elements such as entities, pages or rules.

The Deyel modeler allows creating different types of objects that make up an application. Each of them uses specific tools, with its own modeler, which offers visual design areas, toolbars, properties panels, and design wizards. **Diana** can be used in a complementary way, allowing the modeler user to freely alternate between visual development and AI-assisted development, according to their needs and preferences.

Access to the Modeler

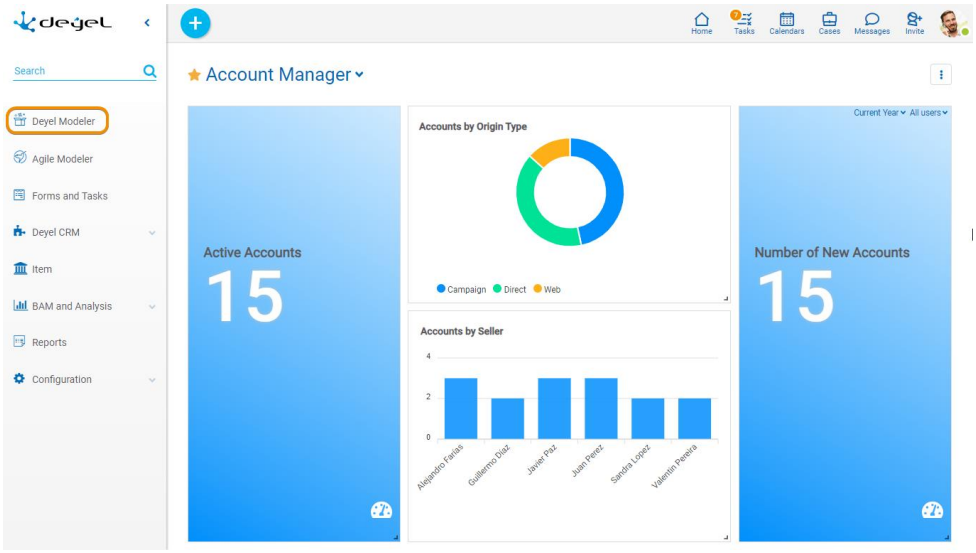
Access to the modeler is done through a URL. The access path is the same as that of the **Deyel** environment but the suffix "/deyelModeler" is added.



Example:
If the access path to the environment is "https://myenvironment.deyel.com", then the path to the modeler is "https://miambiente.deyel.com/deyelModeler".

If the user is not authenticated on the platform, they will be prompted to enter their user ID and password.

Another way to enter it is from an application menu, using the "Deyel Modeler" option, which opens the modeler in a new browser tab.



Views

The **Deyel** modeler can be used from two perspectives, depending on the experience level of the user responsible for the modeling.

- Initial view
- Advanced view

The view to use is selected from the user preferences.

3.4.1. Diana

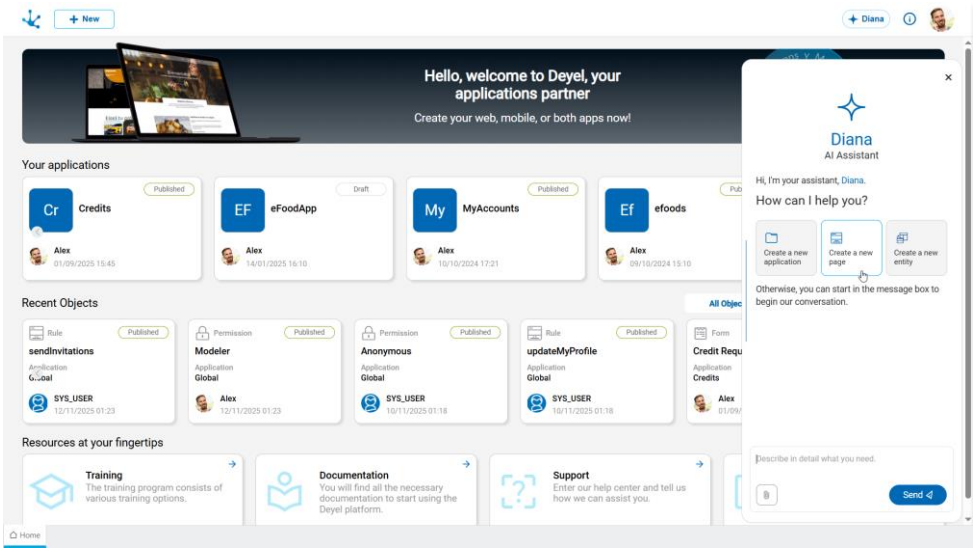
Diana is the team of artificial intelligence agents of **Deyel**, designed to accelerate low-code and no-code application development through conversational prompts.

It brings together specialized agents who assist in modeling an application, defining its, user stories and personas and user flows, generating pages, defining entities and creating advanced and integration rules. It also includes an expert agent that teaches how to use the platform by leveraging the product documentation.

The primary role of **Diana** is to teach, simplify, guide, and automate the development process using natural language. Through prompts, it allows for the rapid generation of a basic application structure, which can then be adjusted and customized using the Deyel modeler.

Diana acts as an orchestrator for the specialized agents, activating the one that best matches the modeler's needs at each moment. Although the interaction takes place through a single chat, the conversation is directed by the agent best suited to handle the requested task.

To start **Diana**, select its icon, located in the top right corner of the Deyel modeler, or access it from each specific modeler.



Main Features

Conversational development with prompts

The modeler user interacts with **Diana** in natural language with prompts, to build their application and objects. Based on the conversation, the main objects are generated automatically:

- Applications with the definition of user stories and personas.
- Canvas with the navigation flows.
- Relation between user flows and user stories.
- Pages, advanced rules, integration and entities modeling.

Best practices and methodological approach

Diana guides the user through a structured approach that promotes orderly and coherent application modeling. This includes:

- Clear description of personas.
- User stories aligned with business objectives.
- Well-defined user flows that are related to the stories.
- A proposal of next steps to complete the modeling of all application objects.

Assistance in using the platform

In addition to generating objects, **Diana** teaches how to use **Deyel** through natural and contextualized conversations. Developers can consult on how to perform tasks, request recommendations or clarifications on specific functionalities, obtaining immediate and accurate guidance.

In this way, new users can quickly learn the platform's basic concepts and begin taking advantage of all its capabilities.

Proposal and confirmation workflow

For each request, **Diana** proposes an action, for example, creating a page, an entity, or a rule, which the modeler user validates before its execution on the platform.

This mechanism ensures control, quality, and transparency throughout the entire modeling process.

Smart templates to accelerate application development

Diana can generate objects conversationally way, based on predefined templates that incorporate standard functions and best practices. These templates are adapted to different business scenarios, such as:

- B2B templates for internal applications in companies.
- B2C templates for end-customer-oriented applications.
- Vertical templates for industry-specific solutions, such as loan origination for banks, claims management for insurance companies, and insurance product sales.

Smart context and consistency in generation

Diana understands the current state of the application, the role of each user, and the reference documentation, always responding in a fully contextualized manner.

When generating new objects, it maintains the coherence of the model:

- It reuses standard field naming conventions.
- It respects the defined data types.
- It proposes relations with value lists, advanced rules, and other existing entities.
- It suggests advanced rules consistent with those already modeled.

This ensures that applications grow in an organized, consistent, and well-governed manner.

Simple maintenance with zero or low-coding tools

While the initial modeling process be done conversationally, the maintenance and the evolution of applications are carried out using the visual tools of **Deyel**, facilitating the continuous improvement of the application.

Advantages:

- Governance and traceability of changes.
- Ease of use for citizen developers.
- Sustainable and easy-to-maintain applications in the long term.

Benefits

- **Agile start**: applications ready faster thanks to prompts and templates.

- **Quality and consistency**: objects generated following best practices and application modeling standards.
- **Flexibility**: develop from scratch or use templates depending on the case.
- **Affordable maintenance**: visual low-code and no-code tools.
- **Productivity and scalability**: citizen developers and IT teams working in synergy.
- **Competitive differentiation**: **Diana** turns **Deyel** into an AI-first platform, combining the best of conversational development with the robustness of low-code and no-code

Agents that Make up Diana

Diana guides the modeler user in using the tool, accessing the documentation of **Deyel** and coordinating the assistance of the agents according to the modeling needs. This makes it the ideal copilot to support the user throughout the creation of an application.

The specialized agents of **Diana** collaborate in the different stages of modeling, contributing specific knowledge and skills to assist the modeler user in their tasks.

The functionalities of each agent are detailed, along with how their assistance contributes to accelerating development, maintaining model coherence, and ensuring the quality of the applications generated with **Deyel**:

- Applications Agent
- Entities Agent
- Pages Agent
- Rules Agent

3.4.1.1. Applications Agent

This agent acts as an intelligent assistant that accelerates the application development, ensuring functional consistency and alignment with business objectives.

It is designed to assist in the creation and extension of applications within **Deyel's** development environment.

Main Functionalities

- **Conversational interaction** where the agent is based on an interactive chat with the modeler user, who describes in natural language the application he wants to create.

  During the conversation, the agent asks follow-up questions to gather the necessary information and understand the business objectives of the application.

The user can respond freely or attach a file with the complete request or any additional information that helps define the context.
The agent continues the interaction as many times as necessary until all the data required for the creation of the application is gathered.

- **Business-aware assistance** , as the agent includes artificial intelligence trained on business models, allowing it to provide contextual information, best practices, and suggestions based on common use cases.

  This capacity for interpretation and contextual assistance improves the accuracy and quality of the functional requirements gathering.

- **Generation of applications** based on the definition of personas, user stories and user flows, including the actions associated with each of them. The generation follows a development methodology, which ensures consistency between the created objects, the alignment with business objectives, and the application of best modeling practices.

  These definitions allow the application's foundation to be built automatically, including its functional structure and the connection logic between the different components.
  Applications are generated from base B2B (internal) or B2C (end-customer oriented) templates that incorporate standard functionalities and best modeling practices, and can be adapted according to the user's requests.
  There are also versions geared towards specific use cases, such as loan origination, claims management, or the sale of insurance products.
  Once the generation is complete, the application is automatically displayed on the canvas, with all the elements previously defined during the conversation.

- **Extension of existing applications** incorporating, editing, or deleting personas, user stories, flows, and actions in existing applications.

  This functionality allows the application to be easily adapted to new requests or functional scenarios, without losing consistency with the existing model.

- **The applications generated by the agent consist of** Deyel objects, this allows for its subsequent maintenance, evolution, and administration using the platform's visual tools.

Step-by-Step Guide for Using the Application Agent

1. Start interaction with Diana

The modeler user can start from the “Diana” button, located in the upper right corner of the modeler, or from the “Create” button, by selecting the “AI Diana” option.
By doing so, the conversation panel will open, where the modeler can describe the application they want to generate.

2. Describe the purpose of the application

The modeler starts the conversation by explaining in general terms what type of application they need and their purpose within the business.

Diana interprets the request and directs the modeler user to the application agent, which guides the user with supplementary questions to obtain the necessary context.

3. **Define personas and user stories**

   The agent suggests the definition of the personas and their user stories. If the modeler user already knows them in advance, they can describe them themselves.
   This information allows understanding the main objectives, needs, and actions that should be considered in the application.

4. **Select the application type and its name**

   The agent inquires whether it is a B2B (internal) or B2C (end-customer) application and requests the application name, if it has not been provided.
   This information is used to generate the base structure and establish the modeling context.

5. **Identify user flows and their associated actions**

   Based on user stories, **Diana** suggests the corresponding user flows, including the actions related to each flow.
   The user can review, adjust or expand the suggestions according to the particularities of the business process they wish to model.

6. **Attach additional information**

   If detailed documentation or requests are available, the modeler user can attach files with the complete request or supplementary information at any time, allowing the agent to interpret the request more accurately.

7. **Application generation**

   With all the information gathered, the agent makes a proposal for the application to be generated and, if accepted, automatically generates it in the modeler's canvas, incorporating:

   - The user personas and stories, which are displayed in the left side panel.
   - The user flows and associated actions.
   - The relations between user flows and user stories.
   - The components that can be solved based on the provided specifications, which are displayed in the components view.

8. **Review and confirmation**

   Once created, the user can review the application on the canvas, validate its structure, and request additional adjustments or extensions as needed.

9. Extension or modification

Diana allows extending or modifying the application at any time. Through new conversations, the user can request the addition, editing, or deletion of personas, user stories, user flows, and actions, always maintaining functional consistency and alignment with business objectives.

10. Interaction with specialized agents

Once the initial structure has been generated, **Diana** activates and coordinates specialized agents as appropriate. In this way, the entity agent, the page agent, and the rules agent can intervene to define, generate, or extend each of the objects required for the application, maintaining consistency and adherence to the model's standards.

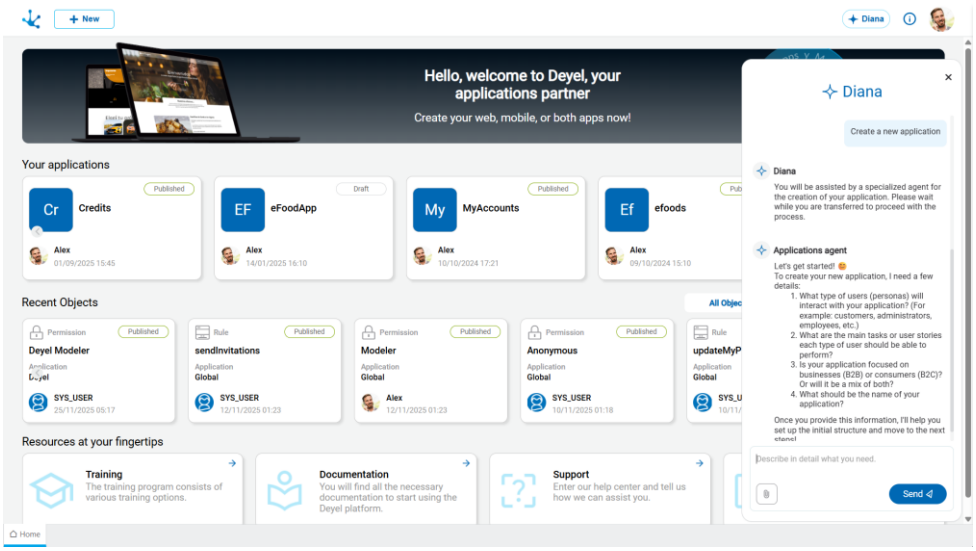
Example

To create an application with **Diana**, the user can start the process from:

- The button "Diana", in the top right corner of the platform's modelers.
- The button "Create" in the top left corner, then select the "Diana AI" option.

In both cases, the **Diana** assistant panel opens and interaction can begin in the dialog box, entering a description of the application to be created.

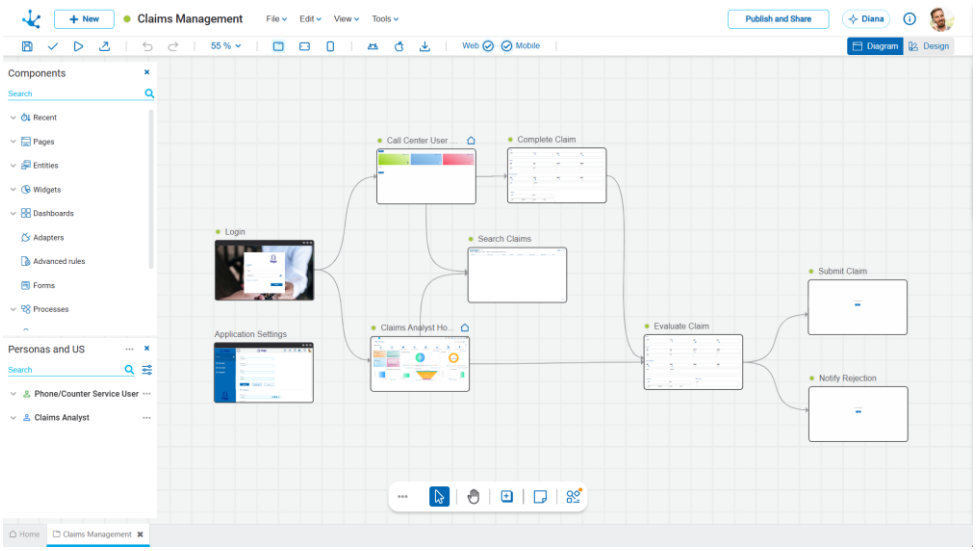
Optionally, a file can be attached so that **Diana** understands the request.



By clicking on "Send", **Diana** responds by proposing the personas, user stories and the type of application to create, for example, B2B or B2C, and finally asks for the name of the application.

Once all the necessary information is available, **Diana** automatically generates the application, incorporating the following elements into the modeler's design area:

- Personas and user stories.
- Canvas with user flows.
- Relation of user flows with user stories.
- Entities, pages, and advanced rules that were generated based on the entered specifications.



3.4.1.2. Entities Agent

This agent allows the generation and extension of entities based on a conversational request, ensuring that they are correctly defined and adapted to business needs while complying with the modeling standards of **Deyel** .

The result includes the entity with its fields, grid, and useful filters. In addition, the agent uses the application context to determine relations with other entities, value lists, and rules, applying standards and best practices that facilitate its maintenance and ensure the integrity of the application.

Main Functionalities

- **Interaction through a conversation in natural language** with the agent, who guides the modeler user step by step to gather the necessary information.

  During the dialogue, the agent can request details about the purpose, name, or application where the entity should be included, adapting their questions according to the context. If the modeler user is already working within an application, the new entity is automatically added to it.

  Besides, the modeler user can attach files with requests or additional information to help define the entity's structure more accurately.

- **Intelligent assistance in modeling** as the agent uses contextual knowledge and best data modeling practices to suggest field names, data types, relations, and structures consistent with

the rest of the application. This ensures consistency and reuse of common elements, optimizing the quality of the model.

- **Entity generation** based on the descriptions of the modeler user, considering the context of the application and ensuring compliance with modeling standards.

  The resulting entity includes all defined fields and their associated grid, with filters useful for its administration and display.

- **Extending entities by adding fields** in entities, according to the needs of the modeler user, either from a request or suggestions from the agent themselves.

  This functionality is performed within the entities modeler, maintaining the coherence and traceability of the design.

- **Generation of entity instances**, to populate them with sample or test data, ensuring that they respect the relations established with other entities, value lists or rules. This allows validating the model and displaying the behavior with business information.

- **Entities available in the Deyel modeler**, they can be modified, extended or related to other objects through the platform's visual tools.

Step-by-Step Guide for Using the Entity Agent

1. **Start interaction with Diana**

   If the modeler user is working in the modeler, they can start the conversation directly with **Diana** to define a new entity. **Diana** recognizes the context and automatically activates the entity agent.

2. **Define the entity through a guided interaction**

   The modeler user begins the definition by specifying which entity they wish to create.
   Based on this initial description, the entity agent leads the conversation to gather necessary information, such as purpose, name, and the application in which it will be included. If the user is already working within an application, the new entity is automatically added to it.

   The agent builds the entity's data model, suggesting the fields that make it up, the relations with other entities, value lists, and any advanced rules. Through progressive interaction, it allows the entity to be developed collaboratively and accurately, adapting to the business needs. The modeler user can review, modify, or extend the structure until the desired outcome is achieved.

   Additionally, files with requests or any supplementary information can be attached to enrich the definition.

3. **Generation of the entity**

   The agent automatically generates the entity in the entities modeler, including its grid, filters, and relations configured as indicated during the conversation.

4. **Generation of instances (optional)**

   The modeler user can ask **Diana** to create data instances to populate the entity with business examples, making it easier to validate and test the model.

5. **Final review and adjustments**

   The generated entity can be displayed, modified, or extended within the modeler, maintaining its coherence with the rest of the application. The modeler user can continue the conversation with **Diana** to extend the entity and relate it to other objects, or to do so from the graphic modeling area available on the platform.

Features of the Generated Fields

The agent allows defining and configuring the properties of each field that makes up the entity, ensuring its correct description and operation within the data model.

The available properties are as follows:

- Name
- Description
- Data type
- Size
- Visibility in the entity's grid
- Availability in the search filters
- Group of fields to which it belongs: It allows grouping them into containers, for example, “Personal data” can contain “First name”, “Last name” and “Marital status”.
- Relation with other objects, being able to link to the data of another entity, an advanced rule or a value list.

Breakpoints

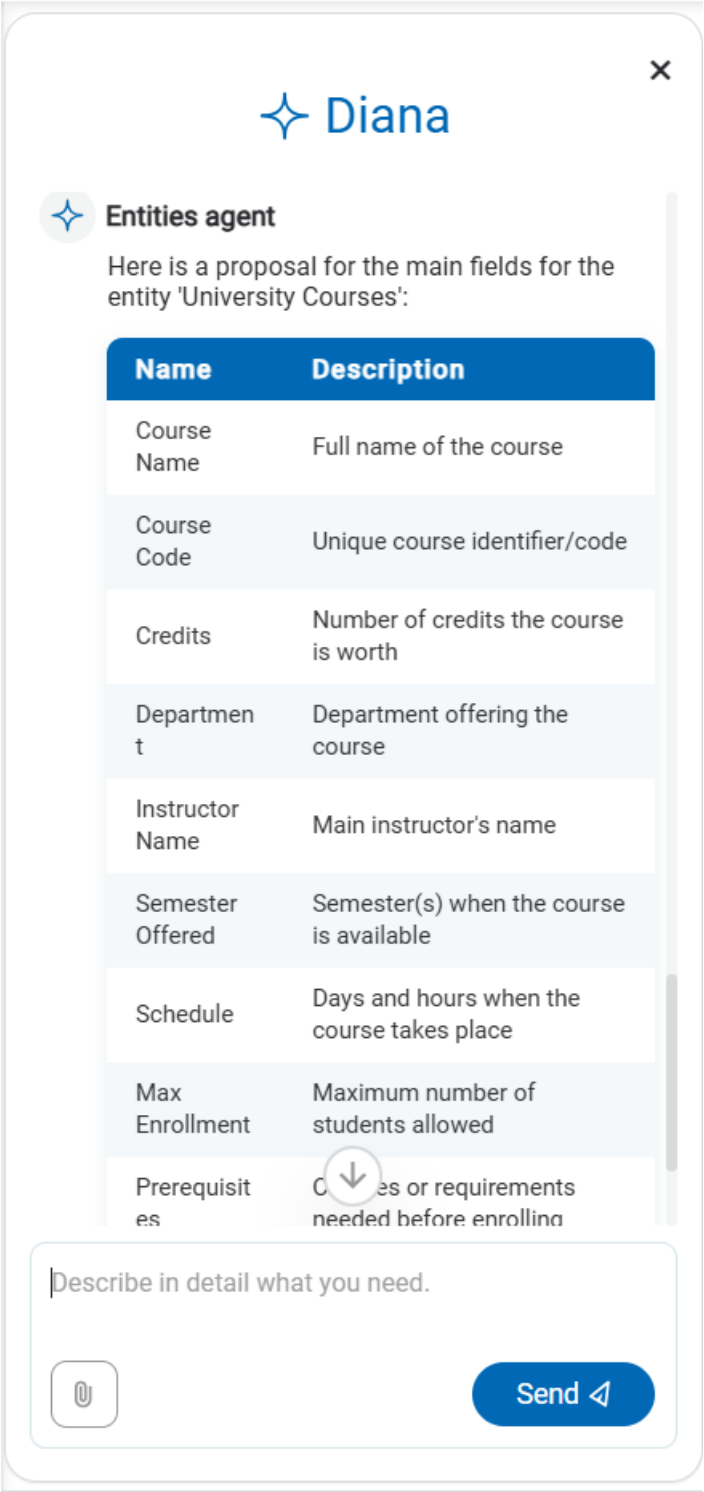
The generated entities are fully adaptable and display correctly at all resolutions and breakpoints, ensuring a consistent experience in both web and mobile environments.

Example

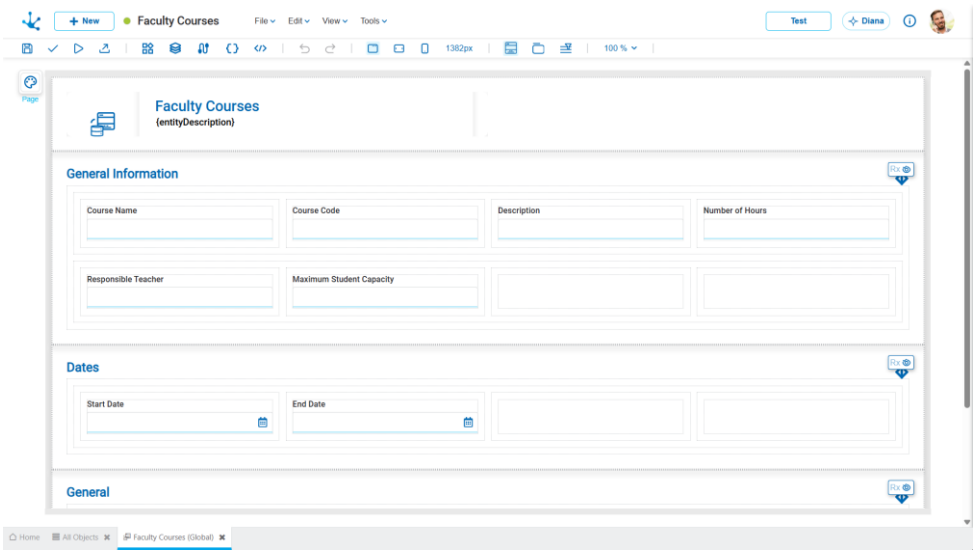
The modeler user initiates the conversation with the entity agent and makes the following request:

```
I would like to create an entity that represents the courses of a faculty.
```

The agent suggests the corresponding fields within the conversation:



Once the field structure is confirmed, the agent then generates the entity in a tab of the entities modeler.



The modeler user can request the extension of the entity and the agent suggests the new fields in the side panel. After confirmation, the additional fields are automatically added to the end of the original entity.

Generation of Instances

The generation of instances with the assistance of **Diana** allows for the quick population of an entity with contextualized data tailored to user requirements.

This functionality streamlines development by facilitating testing, demonstrations, and configuration of working environments with coherent and functional data sets.

The modeler user can request the agent to generate a specific number of instances, indicating the desired data features in the conversation.

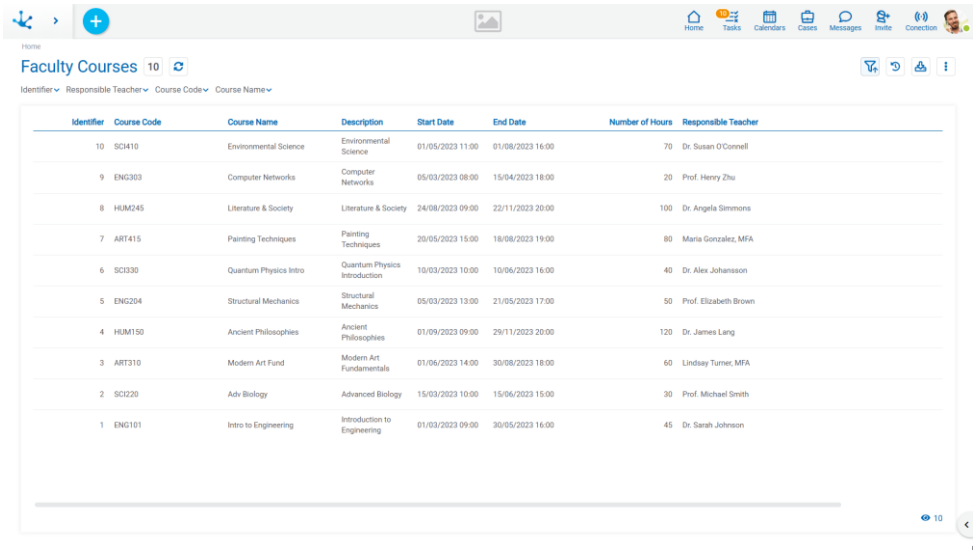
Example

The modeler user requests the following from the entity agent:

```
Generate 10 instances for the entity “Faculty Courses” with the following data:
```

- University courses in different areas (engineering, science, art, humanities).
- Number of hours: between 20 and 120.
- Maximum capacity: between 15 and 40 students.
- Dates: start between March 1 and September 1 of the current year; end within the following 3 months.
- Responsible professor: plausible academic names.

The agent generates instances of the "Faculty Courses" entity with data consistent with the established conditions, ready to be displayed and used in the entity grid.



3.4.1.3. Page Agent

This agent allows application pages to be generated in a simple, conversational way, interpreting the modeler user's requests and proposing visual structures that are consistent with the application's overall design.

Through the dialogue with **Diana**, the user can define the functionality, elements, and style of each page, without needing any design or programming knowledge.

Main Functionalities

- **Conversational interaction** to create pages, through a natural language dialogue with the agent, who guides the modeler user to gather the necessary information.

  The user can also attach reference files, such as mockups, or images (Figma), for the agent to use as a basis during generation.

- **Intelligent assistance with design and user-experience knowledge** enables the agent to apply UX/UI best design practices and leverage the application's design system to generate pages that are visually consistent, functional, and accessible.

  It proposes balanced structures, optimized layouts, and appropriate elements based on the nature of the page and its content, following best practices for pages modeling.

- **Generation of 100% responsive pages** that automatically adapt to different resolutions and devices, ensuring correct display in all environments.

- **Addition of images to improve display** in order to enhance the presentation of the generated page. If the required page contains image elements, the wizard can include sample images taken from the internet.

- **Generated pages available within the platform**, this enables editing, customization, and maintenance through the pages modeler.

Step-by-Step Guide for Using the Page Agent

1. **Start interaction with Diana**

   From the applications modeler, the user can start the conversation with **Diana** to create a page, that automatically activates the page agent.

2. **Describe the page to be generated and determine the application to be used.**

   The modeler user begins by indicating what type of page they want to create within the application and its purpose.

   The agent may request additional information to understand its functionality.

3. **Attach references (optional)**

   The modeler user can attach design files, such as images, html files, or even URLs. Reference documentation can also be added so that **Diana** uses it as a guide in generating the page.

4. **Confirm the proposed design**

   The agent describes the sections of the page to be generated, with details of the elements within each one. In this way, the agent requires confirmation from the modeler user to begin generation.

5. **Page generation**

   Once the necessary information has been collected, the agent automatically generates the page and presents it in the pages modeler in a new tab.
   The page includes all the defined elements, organized according to design and usability best practices.

6. **Preview and test**

   The user can preview and test the created page directly in the pages modeler, validating its behavior and visual consistency before its final publication.

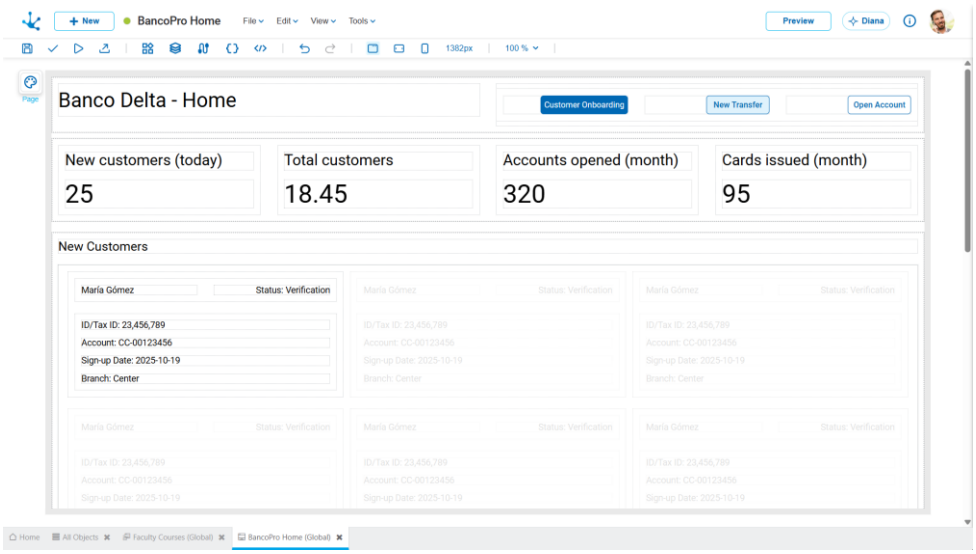
Example

1. **Home page for a banking application**

The modeler user requests the following from the page agent:

```
Generate a complete home page for a banking application, where it is possible to view new customer data, create new customer records using buttons, and see the latest money transactions, among other operations.
```

The agent creates a page with the requested structure, including the elements necessary to display information, perform operations, and navigate between sections.

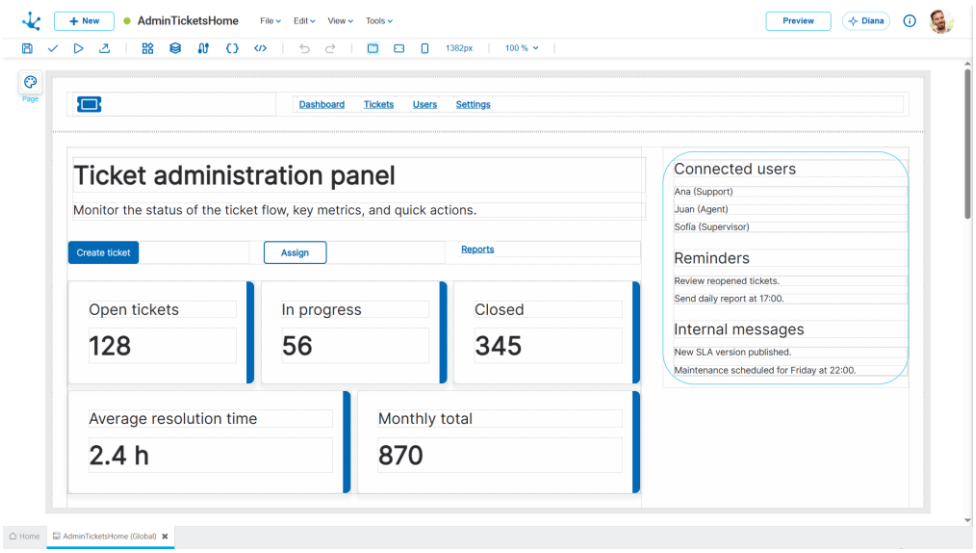


2. Home page for a ticketing system administrator

The modeler user requests the generation of the page by entering the following message:

```
I want to generate a home page for a B2B ticketing application, so that the
administrator can see all the relevant information for that type of appli-
cation.
```

The result obtained is the page included below.



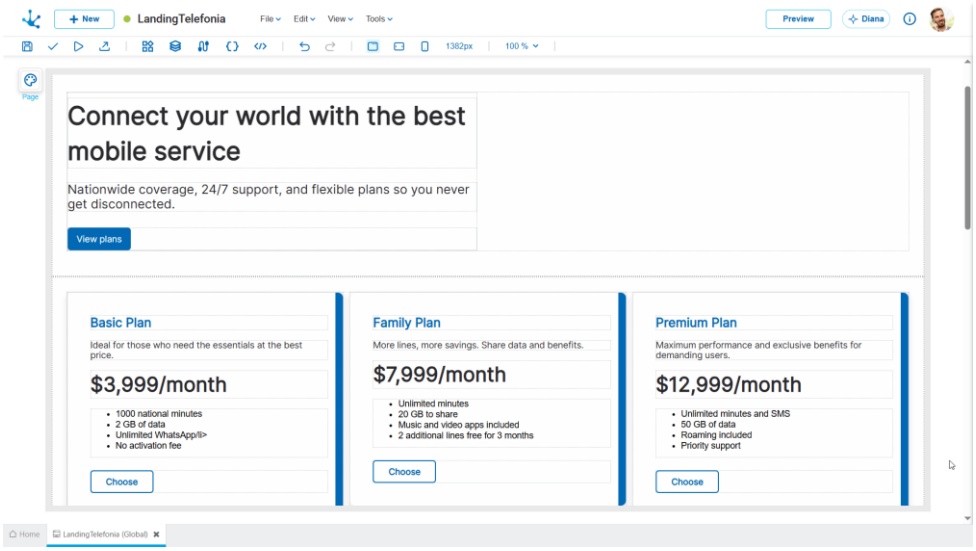
3. Welcome page for a telephone company

The modeler user sends the following message to the page wizard with a generation request:

```
I need to create a welcome page for a telecommunications company, where the
different plans offered are displayed, customer reviews are shown, a con-
```

tact form is included, along with other elements that may be considered important to include.

The page generated by the specialist agent is like the one included below.

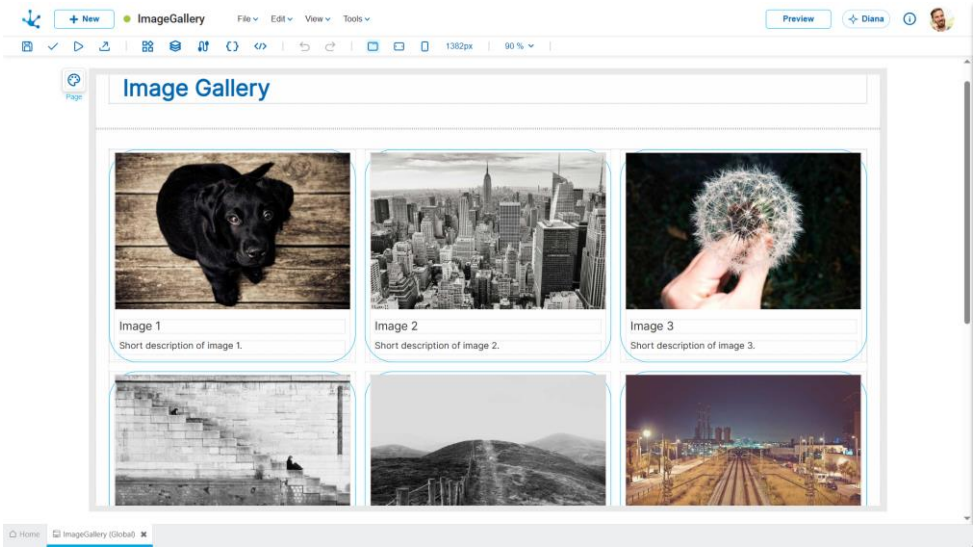


4. General pages

The modeler user sends the following message to the page wizard to create a page with general features within an application:

I need an image gallery to be generated, in a 3x3 grid format. Use random images.

The agent can create a page with the requested features.



3.4.1.4. Rules Agent

This agent assists both the modeler and the IT developer in generating business rules within an application, based on natural language descriptions. These rules can be advanced or integration-related.

Through a guided and interactive approach, **Diana** allows the modeler user to generate rules in a structured and secure manner, accelerating development times.

The agent guides the modeler user through the process of defining, configuring, and testing the rules, enabling them to implement these without any programming knowledge. In cases that require more advanced technical configurations or complex integrations, the modeler user can request the assistance of an IT developer, who provides specialized support to complete the implementation.

In addition, the agent provides the necessary tools to perform tests on the created rules, ensuring their correct functioning before publication.

If the rule is advanced, the modeler user explains in the conversation what they want to achieve, and can rely on any additional documentation they attach. The agent will propose a description of the rule, including a summary of the logic, input and output parameters, and which application objects they will relate to. If the modeler user agrees, the rule code will be generated, based on an SDK adapter.

In the case of an integration rule, the modeler user starts the conversation and indicates that they need to integrate with an application for a specific purpose, and attaches the documentation of the API with which the integration should be performed. The agent proposes the rule description and the mapping of the rule parameters with the corresponding endpoint information. If the modeler user confirms the proposal, the rule code is generated, including preprocessing of the input parameters to invoke the endpoint, processing and postprocessing that maps the endpoint response to the output parameters. The generated rule is based on the adapter corresponding to the external application or service.

Main Functionalities

- **Conversational interaction** that is agent-guided allows the creation of business rules based on the collected data.

  The modeler user can express the request in natural language and attach files with additional documentation or technical specifications, such as API documentation. The agent can request information until they get everything necessary to generate the rule completely.

- **Smart assistance** with technical and business knowledge provided by the rules agent, which leverages AI trained on business models and code development to suggest appropriate validations, calculations, and integration strategies. This allows for optimizing the logic definition and ensures that the rules meet the functional requirements of the application.

- **Generation of advanced rules** to automate validations, calculations, and action flows within the application.

  These rules are generated from the descriptions provided by the modeler user or from attached files, and may include input and output parameters, conditions, actions, logs, and maximum execution times. Once accepted, these rules are automatically generated, ready for publication and testing.

- **Generation of integration rules** that facilitate the creation of rules that connect the application with external services or applications, defining the points of information exchange.

  The modeler user can enter information into the conversation or provide documents that describe the service interface, such as a Swagger file or an OpenAPI file, which details the endpoints, methods, and data structures. The agent uses this information to propose the integration configuration, the input and output parameters, and the necessary technical adjustments.

- **Automatic generation of test cases for the rules**, ensuring that they function correctly and meet the established requests.

  These cases are based on defined parameters and ensure that the rules meet the expected results before publication.

- **Rules available in the Deyel modeler**, which allows for its review, editing or expansion from the rules modeler.

  This ensures centralized and sustainable management of the application's business logic.

Step-by-step instructions for using the Rules Agent

1. **Start interaction with Diana**

   From the applications or rules modeler, the modeler user can start the conversation with Diana to define a new rule.
   If the request is related to the business logic or an external integration, Diana automatically routes the conversation to the rules agent.

2. **Describe the rule to be created**

   The developer begins by explaining in natural language what the rule needs to do and they can also attach files to complete the conversation. If applicable, they can describe which application may include and provide its name. The agent interprets the request and guides the user to gather the necessary technical and functional information.

3. **Gather parameters and configurations**

   Through guided questions, the agent helps define input and output parameters, conditions, actions, logs, maximum execution times, and whether the rule should be exposed as a REST API.

4. **Define the rule type**

   If it is an advanced rule, the agent focuses on automating validations, calculations, or internal flows.

   If it is an integration rule, they request the data of the external service it needs to communicate with, using a Swagger or OpenAPI document as a reference if available.

5. **Rule generation**

Once all the information is gathered, the agent automatically generates the rule code and opens it in a new tab of the rules modeler, ready to be reviewed and tested. The agent analyzes the objects that make up the application to propose their use within the code to be generated.

6. **Test case generation (optional)**

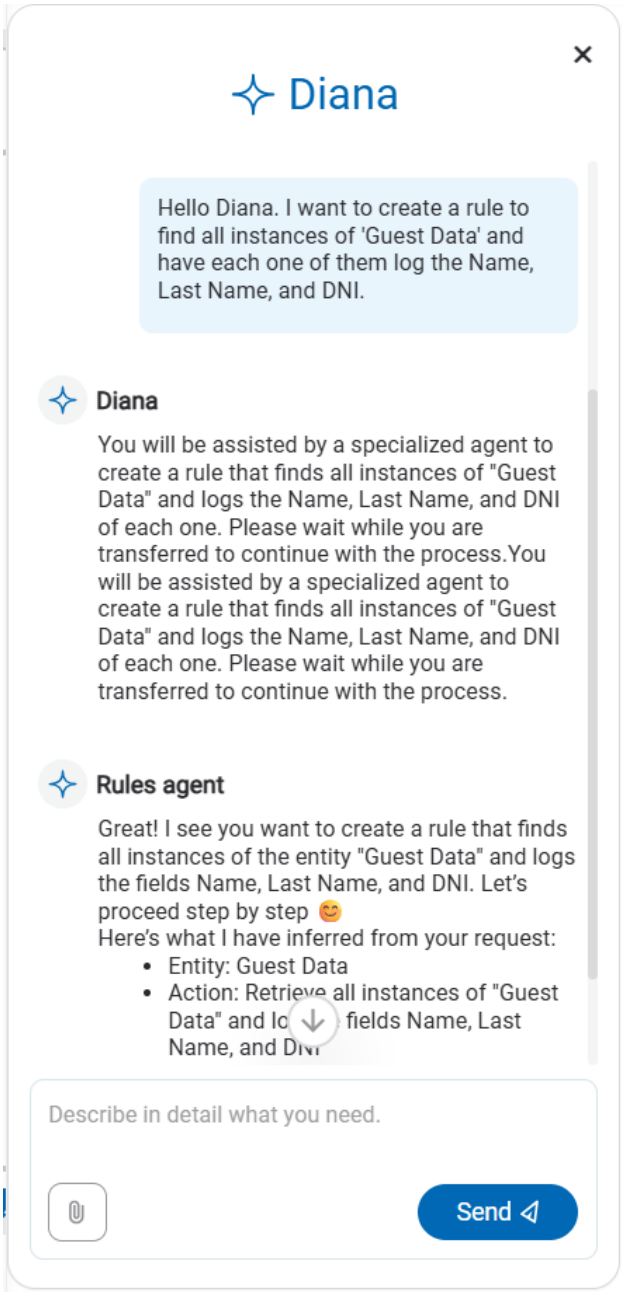
   If the modeler user requests the automatic creation of test cases, the agent proposes a set of cases based on the current rule configuration, which the developer can accept or adjust as needed. Confirmed cases are created automatically and become available for execution.

7. **Review and confirmation**

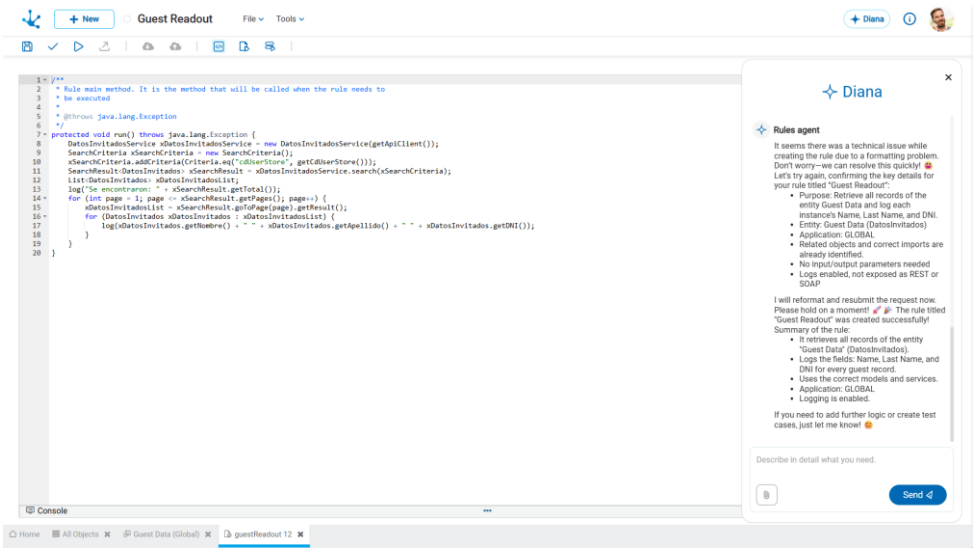
   After generation, the developer can review the code, make adjustments, or publish the rule, ensuring its correct functioning within the application.

Example of Advanced Rule Generation

To get started, the developer must inform the agent of the request for the rule:

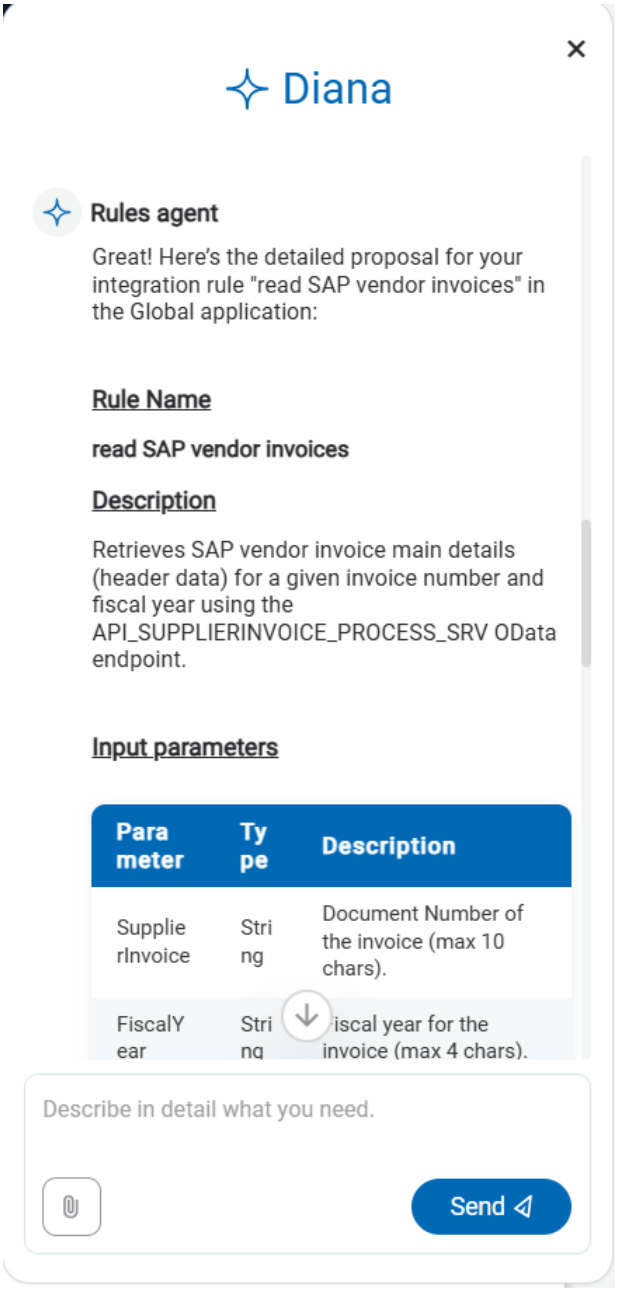


Once this data is confirmed, the agent generates the rule and opens it in a new tab of the rules modeler, ready to be published and tested.

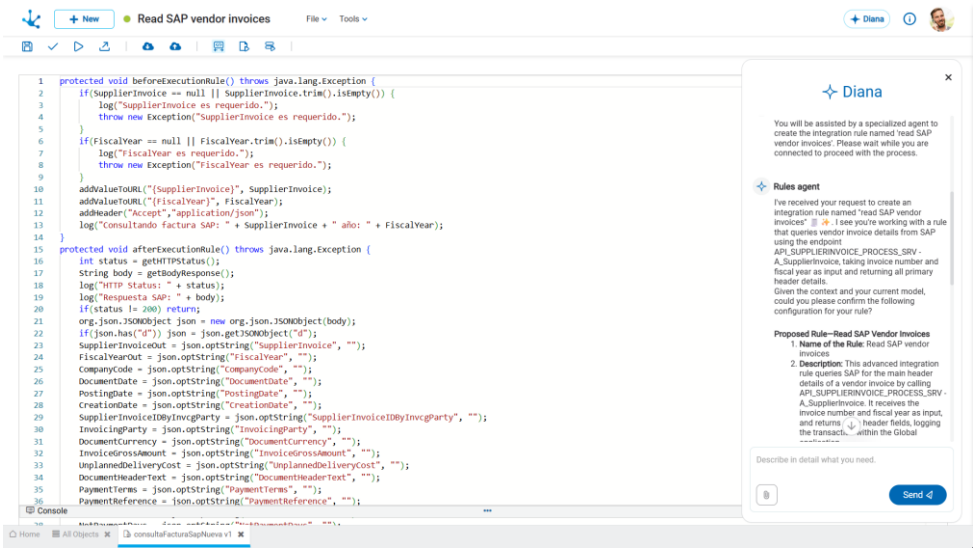


Example of Integration Rule Generation

The developer specifies via an attachment which service external to **Deyel** it needs to be integrated and the request for the rule,. In this example, the request is to retrieve invoice data for a supplier from SAP.
With all the information gathered, the agent proposes an integration configuration, including suggested input and output parameters for the rule.
The agent requests additional configurations, such as enabling logs, exposing the rule as a REST API, or maximum execution times, and displays the complete proposal.

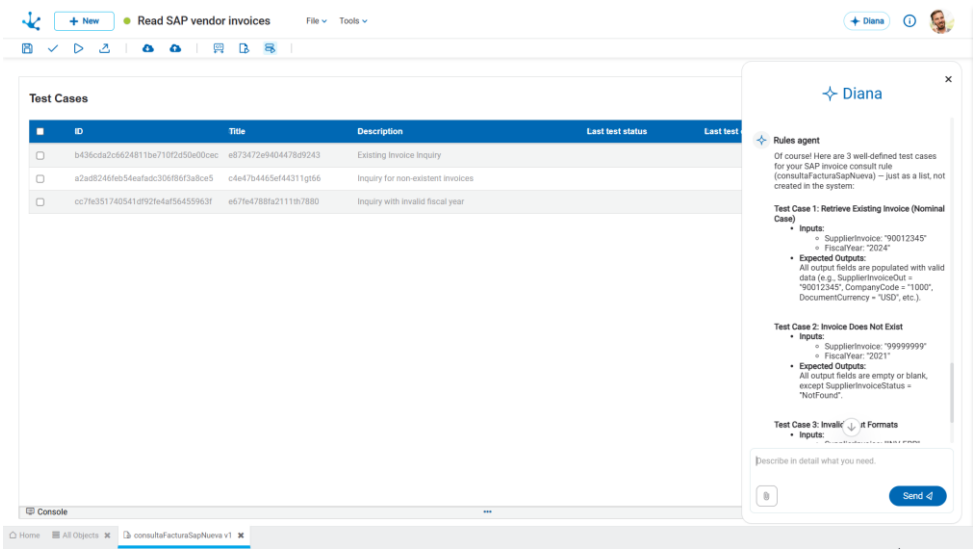


Once the developer confirms the proposal, the agent generates the corresponding code and opens it in a new tab of the rules modeler, ready to be published and tested.



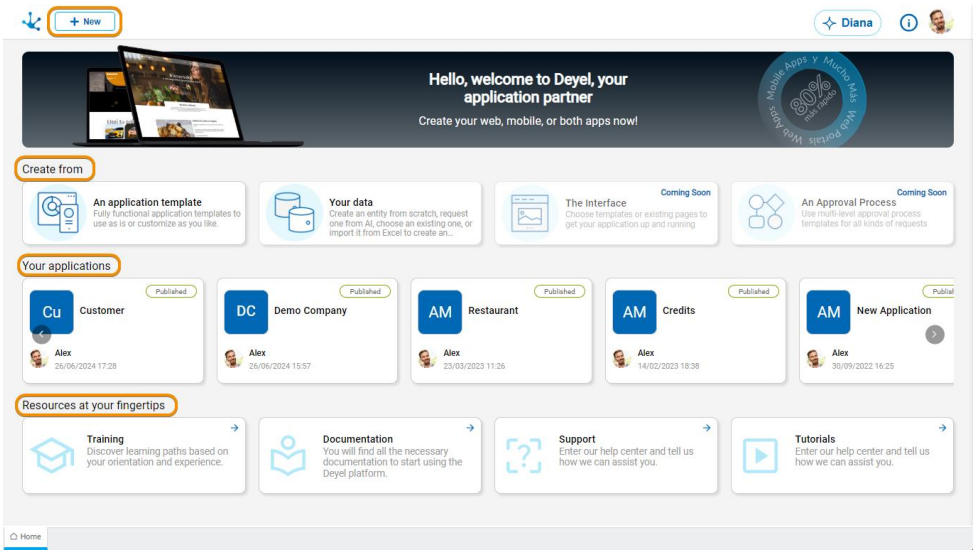
Test Case Generation

If the developer requests the automatic creation of test cases for a rule in the modeler, the agent can propose a certain number of cases, using data based on the current configuration of the rule.
This proposal may be accepted or modified, indicating specific cases. Once confirmed, the test cases are automatically generated and displayed in the rule's test case list, ready to be executed.



3.4.2. Initial View

The initial view of the **Deyel** modeler is recommended for modeler users who are starting their experience with the platform. When accessing, a panel that guides step by step in creating an application is displayed.



Sections

This view is structured into different sections, which allow the modeler user to use different facilities for the development of their applications.

Top Bar

From the top bar, the following facilities can be selected:



It allows the creation of applications from a template, from data, from the interface, or from an approval process.



It opens a panel to start a conversation with Diana.


Provides the ability to access the **Deyel** help center.


It is used to quickly display a summary of the logged-in user data, such as their name, email address, and messaging service state. It can also be accessed to manage profile data of your own, to your preferences for using the modeler, to the facility of messenger service and it also can log out.

Create From

It allows starting the modeling of an application from various origins. It is possible to create an application from a template, from data, from the interface, or an approval process.

Your Applications

It presents the applications modeled within the platform, ordered by update date in descending order, which speeds up access to the recently used ones.

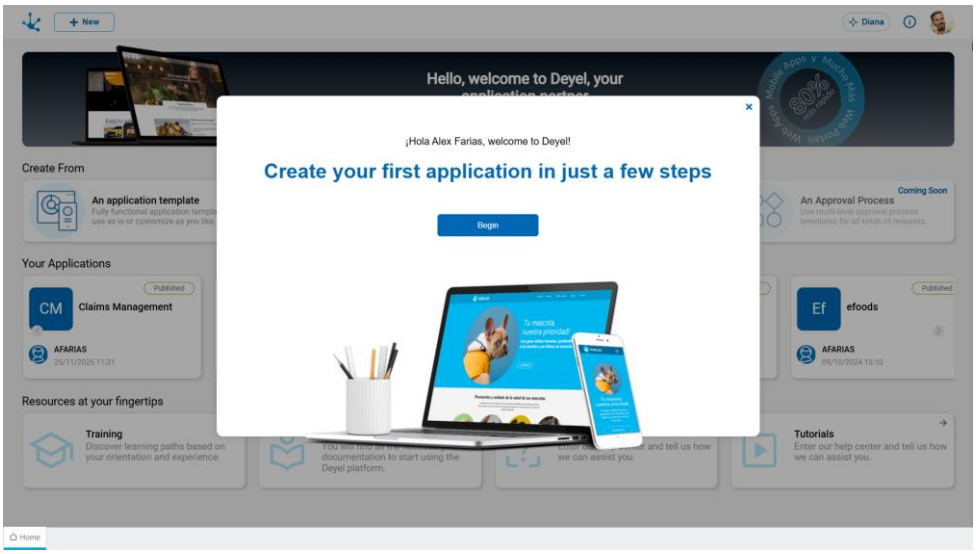
Resources at your Fingertips

In this section it is possible to access the support resources for modeling, it includes links to the documentation, the training page, the incident reporting application, and the tutorials on the tool.

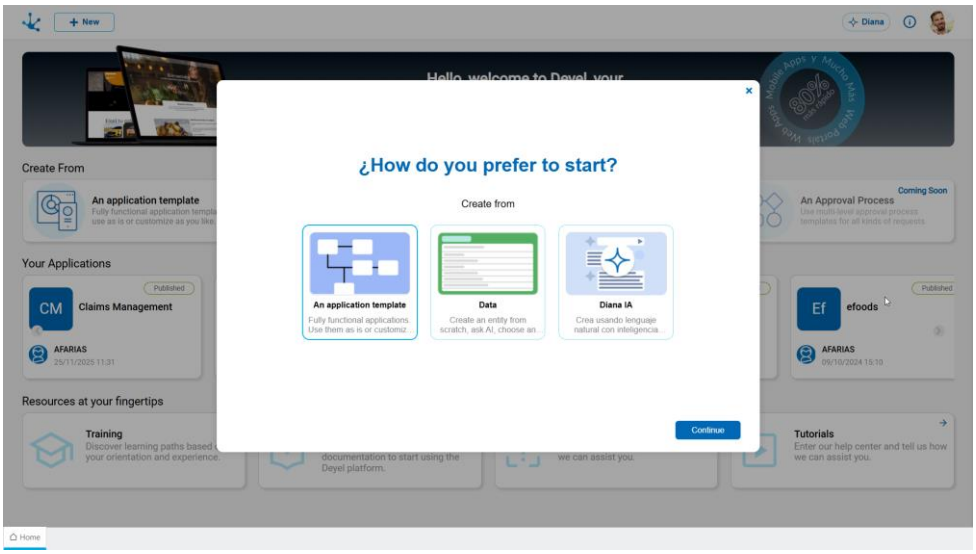
3.4.2.1. Create Your First Application

When entering the modeler, a panel designed to guide users in creating their first application easily is displayed.

This panel includes a welcome interface with an introductory message and the option to start creating.



A new panel is then displayed where the user can select the creation method that best suits their needs.



Available Options for Creating an Application

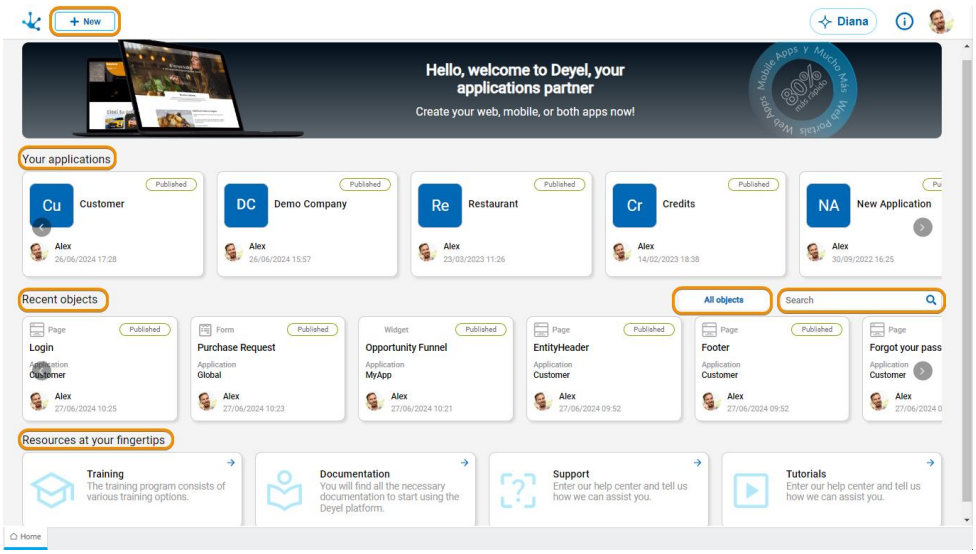
From an Application Template: It allows building an application from pre-configured applications, which speeds up development. These templates can be used as they are or customized according to the user's needs.

From your Data: Enables the creation of an entity-based application. It can be generated with the help of AI, selected from existing ones, or imported from an Excel file.

From Diana: It opens the application modeler and a panel that allows you to start a conversation with Diana to generate the application in an assisted way.

3.4.3. Advanced View

An advanced view of the **Deyel** modeler is recommended for modeler users with the highest level of experience using the platform.



Sections

This view is structured into different sections, which allow the modeler user to use different facilities for the development of their applications.

Top Bar

From the top bar, the following facilities can be selected:



It allows the creation of applications from a template, from data, from the interface, or with Diana's assistance. Unlike the button "New" of the initial view of the modeler, it also gives the possibility of creating all types of **Deyel** objects that can be modeled,



It opens a panel to start a conversation with Diana.


Provides the ability to access the **Deyel** help center.


It is used to quickly display a summary of the logged-in user data, such as their name, email address, and messaging service state. It can also be accessed to manage profile data of your own, to your preferences for using the modeler, to the facility of messenger service and it also can log out.

Your Applications

It presents the applications modeled within the platform, ordered by update date in descending order, which speeds up access to the recently used ones.

Recent Objects

It allows displaying the modeled objects, sorted by modification date in descending order, providing quick access to recently worked-on objects.

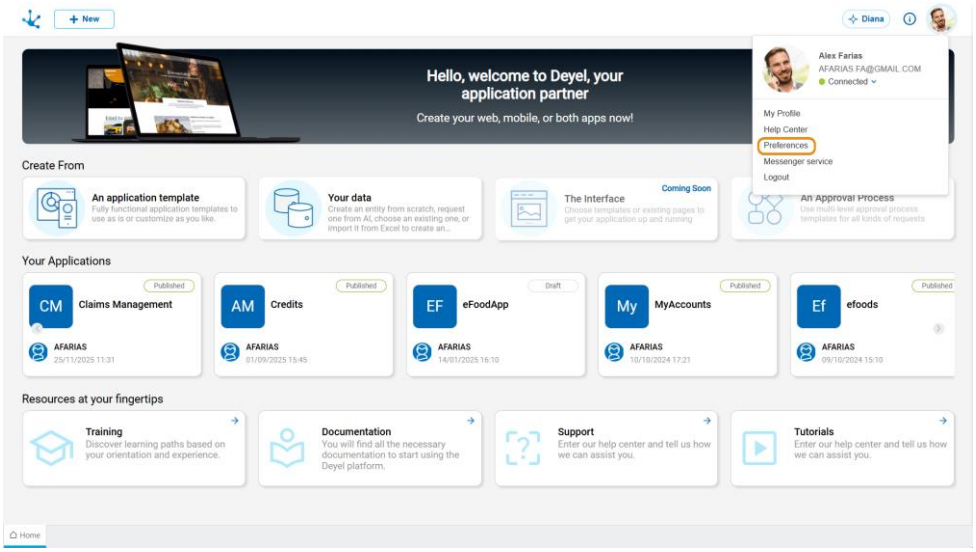
Using the "All Objects" button, the grid with the total number of modeled objects is displayed. Additionally, this section includes a search function that allows for quick searches.

Resources at your Fingertips

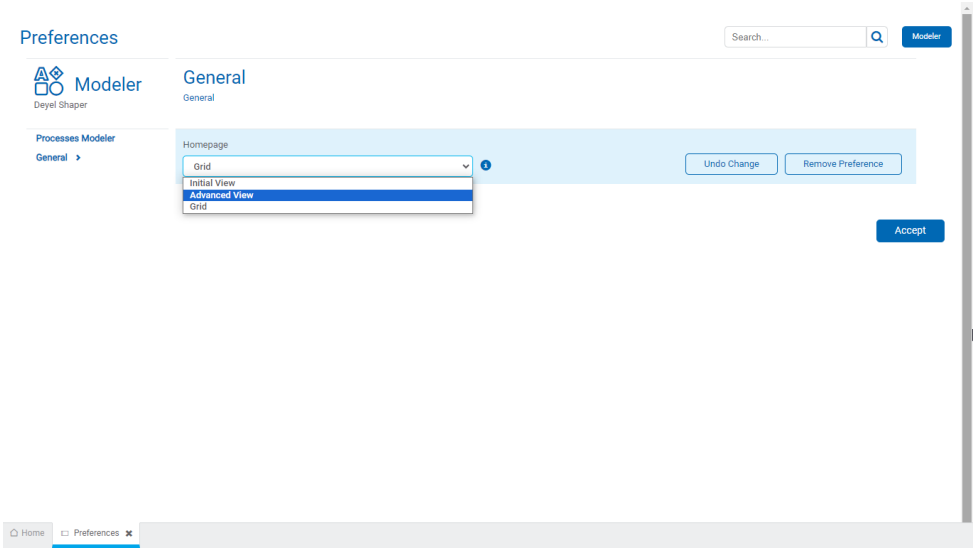
In this section it is possible to access the support resources for modeling, it includes links to the documentation, the training page, the incident reporting application, and the tutorials on the tool.

3.4.4. User Preferences

The modeler view to be used can be chosen from the user profile icon in the top bar, by selecting the "Preferences" option.



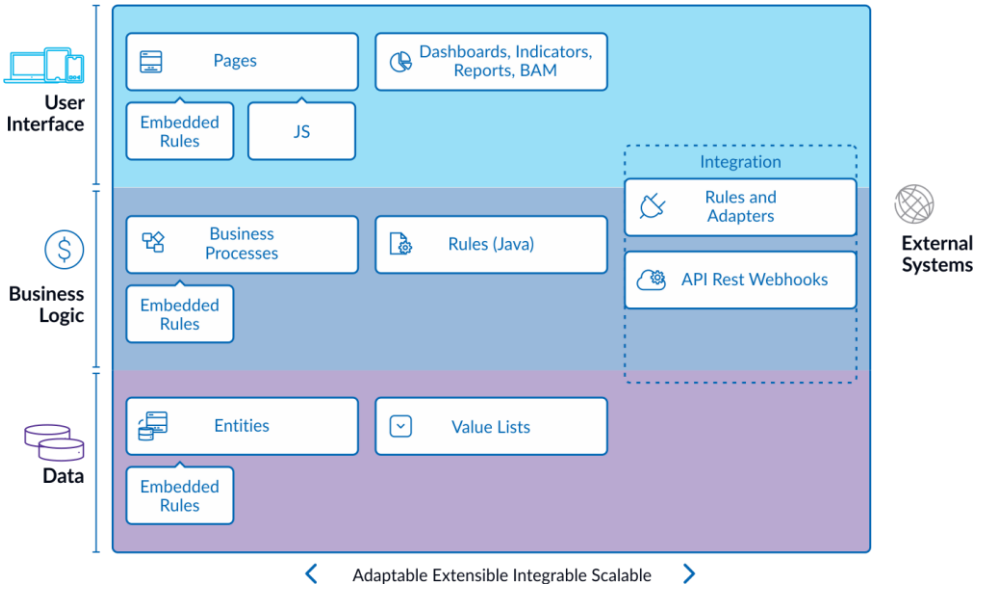
The different views for the modeler can be selected from the “General” option of the “Preferences” menu. **Deyel**.



- Initial View
- Advanced View
- Objects Grid

3.4.5. Objects Modeling

In **Deyel** , applications are made up of different types of objects and can be classified into the groups shown in the following image.



User Interface

For the user interface, page-type objects are mainly used. These allow the modeling of the required interface, whether for mobile phones, tablets, or desktop computers, always with pixel-perfect precision. Other objects complete the user interface of applications such as management widgets, widget dashboards, application reports, and all reports related to monitoring activities and business processes.

Business Logic

At the business logic level, the main objects are business processes and rules. Processes allow for the modeling of human or automatic activities and their relationship with those responsible for their execution. Any type of behavior across the business processes can be modeled using BPMN 2.0 notation. Additionally, through advanced rules, specific behavior can be defined

Data Persistence

At the data persistence or data model level, entities are used that represent the business objects and their persistence in the database. Value lists are also used, which are much lighter objects that allow representing the values used by applications both in the interface and in the entities.

Other Objects

At all three levels, behavior can be added using rules embedded in objects. These rules have a very simple syntax that does not require programming knowledge, and its complexity is similar to an Excel formula. Through these rules, conditions of visibility, editability, requirement, validations and calculations can be defined in all objects, whether in the interface, in the business logic or the data model.

These embedded rules allow adding behavior without the need to code since the wizard makes it even easier in addition to having a simple syntax.
Beyond this, if it is necessary to do something specific at the interface layer, JavaScript code can be added to the pages, using the Deyel SDK. If required at the business logic level, Java-based rules can

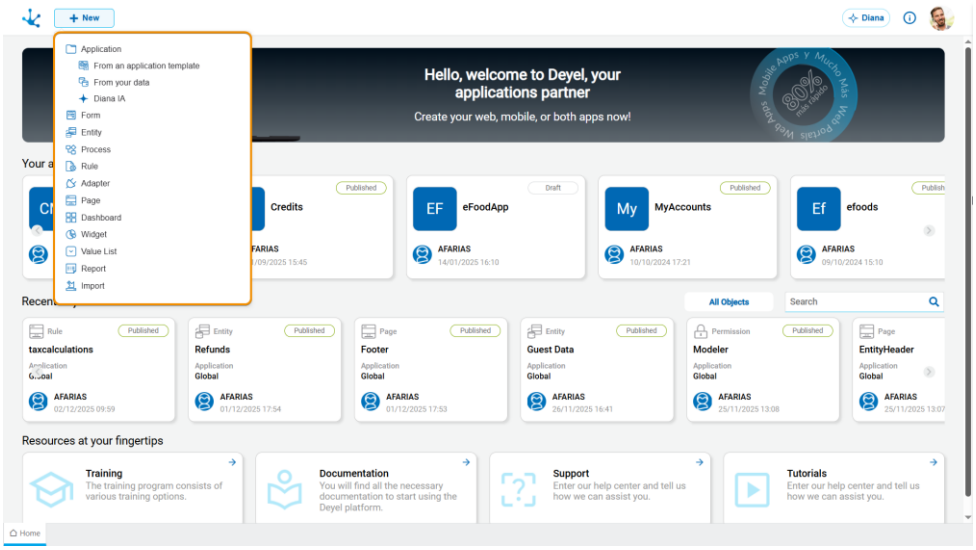
be developed using the Deyel SDK, which allows viewing the different objects in the application as if they were Java objects.

Through adapters and integration rules, applications developed with **Deyel** can be integrated with other applications. Adapters define the coordinates and credentials used to integrate with the other application. Then the rules based on these adapters will be able to interact with the APIs or other integration mechanisms provided by the external applications, whether through a database, web services, or some other type of event or integration mechanism. On the other hand, applications that are developed with **Deyel** automatically generate a Rest API with which any developed objects can be invoked from other applications. When working with REST APIs, there is also the possibility of using webhooks to achieve a more robust integration with external applications.

It is important to highlight that applications developed with **Deyel** are adaptable and extensible, using the modelers of each object, they are integrable through the integration facilities provided by the platform. They are scalable, since through the horizontal scaling provided by the platform, an application can start with a few users and scale almost unlimitedly.

Object Creation

The **Deyel** modeler allows the creation of different types of objects that make up an application, and for each of them, its respective modeler can be accessed from the button "Create". By pressing it, a menu is displayed to create the different objects or perform the import into the environment.



By selecting each of the options, a new tab opens with the modeler corresponding to the selected object.

Application

Form

Entity

Process

Rule

Adapter

Page

Dashboard

Widget

Value List

Report

Import Objects

The import option opens the panel corresponding to the functionality of importing objects to the environment where they are being modeled.

Import

Home

The tab with the icon corresponds to the view of the modeler that the user selected in their preferences. When opening other objects in different tabs, the home tab always allows returning to the selected modeler view.

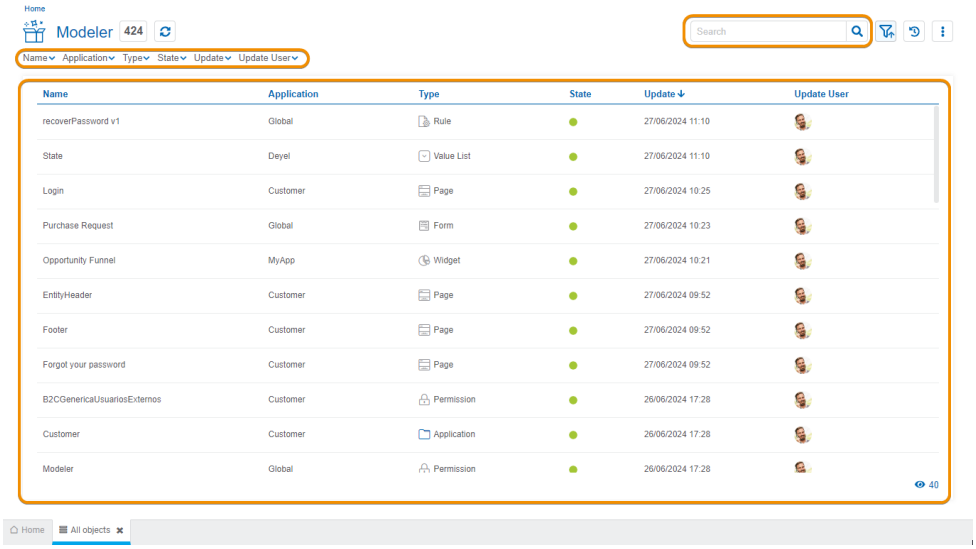
3.4.6. Objects Grid

The modeler's object grid allows the display of all objects modeled in **Deyel**, or those that are the result of having applied search filters.

Sections

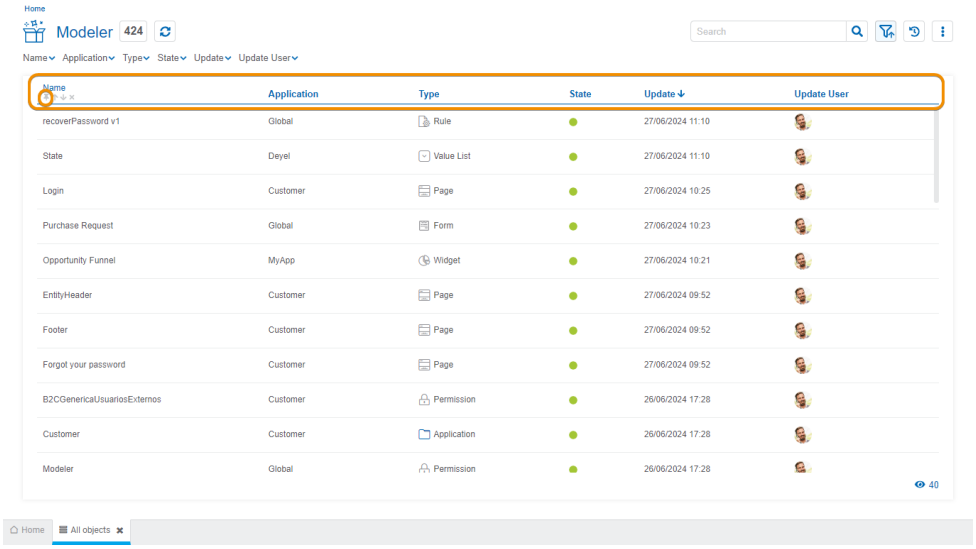
The grid is made up of a main section with columns and rows, in addition to sections that facilitate modeling the applications and their objects.

- Top bar
- Search filters



Columns

The objects grid allows the display of different columns and by hovering over the name of each one it is possible to indicate its behavior.



The grid columns correspond to the characteristics of the modeled objects in **Deyel**.

Name

Name used in the interface to reference the object, corresponds to the Descriptive Name property of each object.

Application
Application the object belongs to, corresponds to the Application property of each object.

Type
Identifies the type of object defined in **Deyel**.

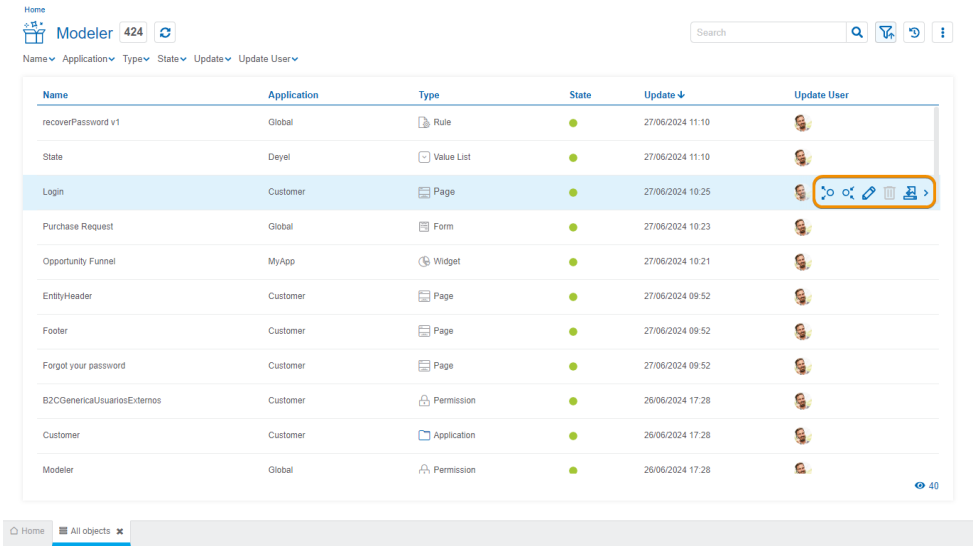
State
Indicates the state of the object: draft, modified, published.

Update
Date of last update made to the object.

Update User
Last user that updated the object.

Rows

A grid row represents a **Deyel** object modeled in the environment. Each object in the grid can be opened in its corresponding modeler by double-clicking on the selected row; while hovering the cursor over it enables icons that allow operations to be performed on the object.



Operations

Operation icons are enabled when hovering over each row of the grid.

Modify
Opens the modeler for the selected object in a new tab.

Delete

Deletes the selected object, after confirmation. This operation can be performed if the object is in "Draft" state and does not have other associated objects of **Deyel**.

 Export
Opens a panel for the user to select and confirm the export of the selected object.

 Precedents
Opens the grid of precedents of the object selected in a new tab.

 Dependents
Opens the grid of dependents of the object selected in a new tab.

Display of Icons on Rows

The user can decide if they want to see the icons with the operations of the objects on the row.

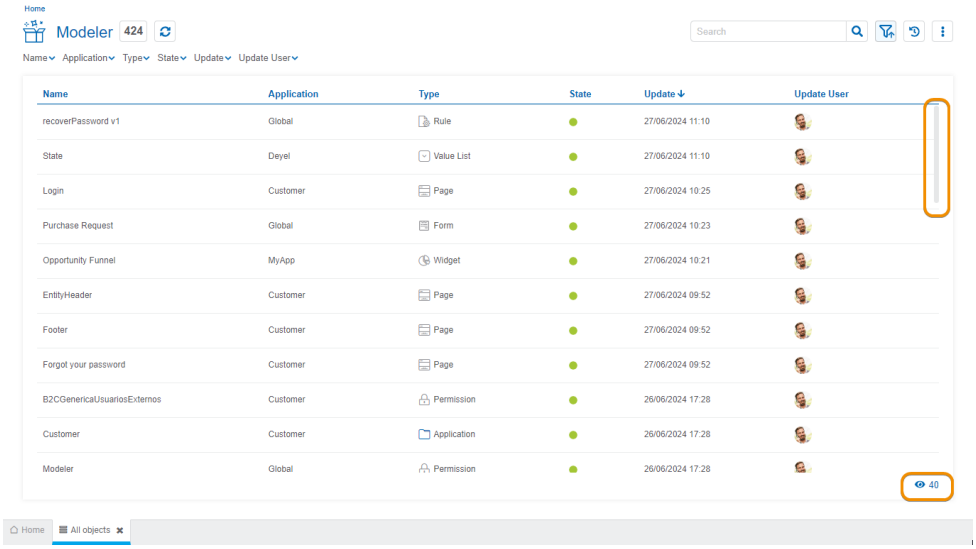
 Displays the available icons for each object.

 Hides the available icons for each object.

Scroll Bar

By scrolling vertically, rows are incorporated in the grid, allowing scrolling up or down using a scroll bar.

The number of objects in the grid can be seen in the bottom right margin. This number increases as the scroll bar is scrolled vertically.



3.4.6.1. Top Bar

From the top bar, different options related to the content and presentation of the objects grid can be selected.

Number of Objects

Indicates the number of objects, either total or that resulting from the application of a search filter.

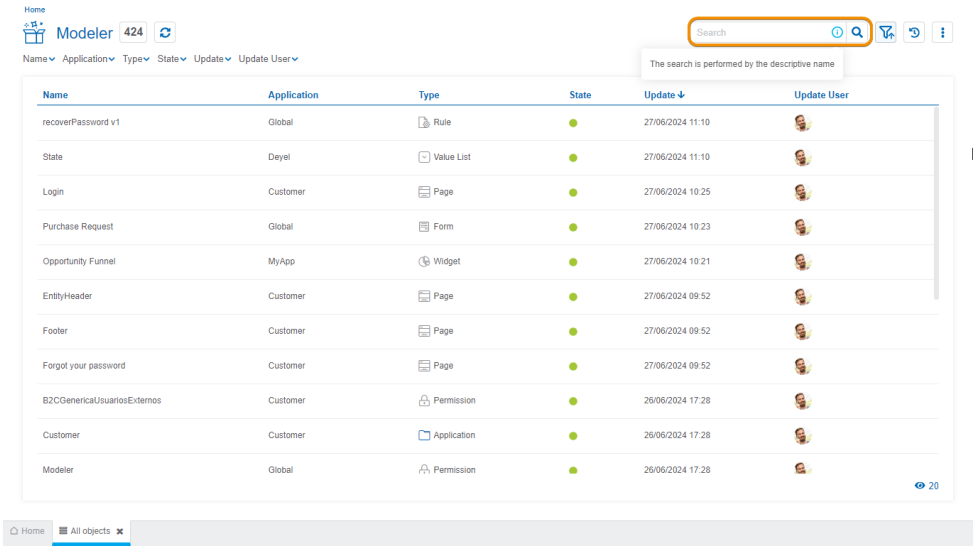
Update Grid

Allows reloading the updated grid, keeping the selected filters.

Quick Search

Allows to filter objects by column Name based on the characters entered in the search field, it corresponds to the Descriptive Name property of the object.

Click on the icon to display the object property on which this type of search can be performed.



Search Filters

The icon enables, on the results grid, the set of fields for which search filters can be selected.

Favorite Searches

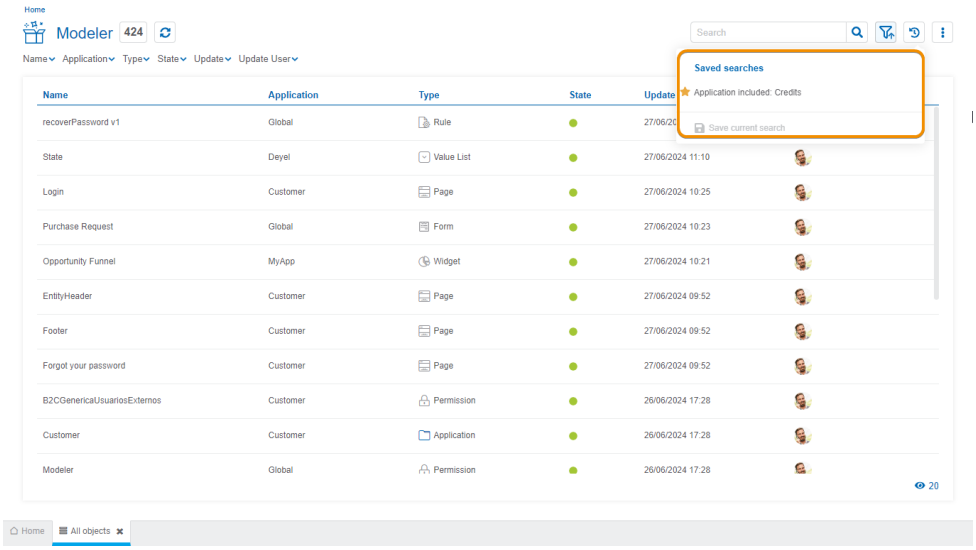
It allows to save the filters selected by the user and to administer them.

Clicking on the icon expands a panel with the list of saved filters and new filters can be added. Every time a search is selected, it is executed and the grid is updated.

It allows to check one of the searches as favorite, which is indicated by the icon ★ and can be deleted by pressing this icon.

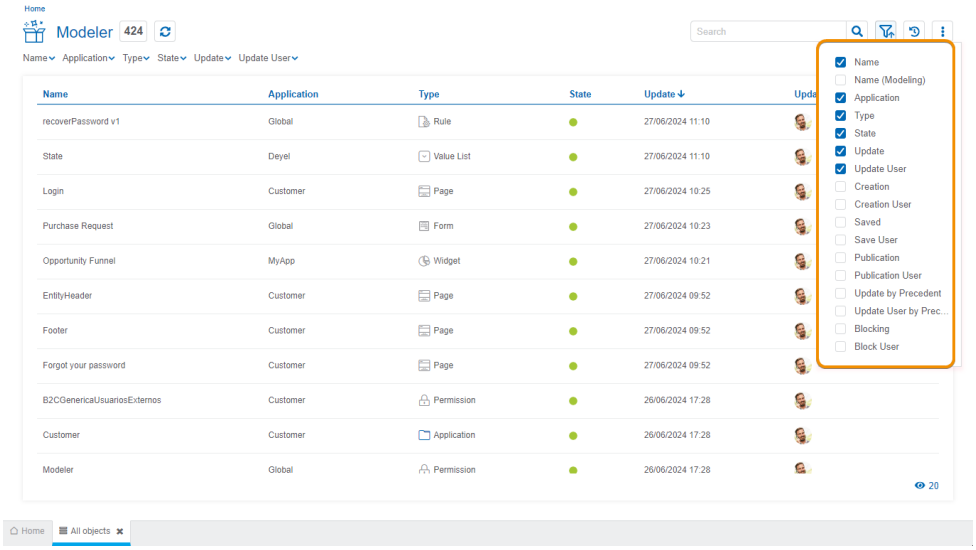
★ Indicates that the search was selected as favorite by the user.

Pressing the icon ☆ to the left of a search, allows the user to indicate that as their favorite.



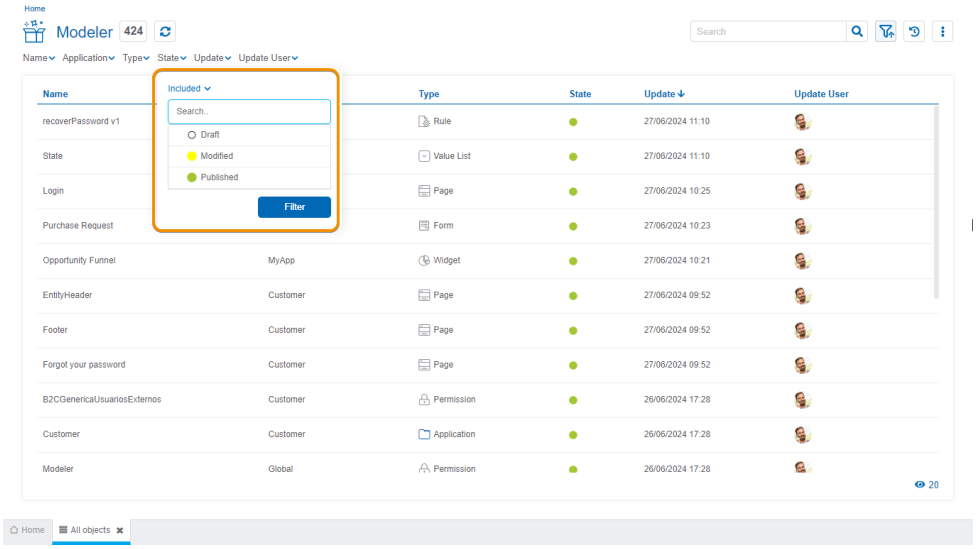
Columns Display

It displays a panel with the names of the grid columns. By means of a check mark, the user can activate or deactivate the display of each column. The set of selected columns is valid until the same user modifies it again.



3.4.6.2. Search Filters

The icon corresponds to the search filters and enables a row to select the different criteria to filter objects.



For each search criteria a value can be entered or selected from a list, to use it as a filter.

All the filters that are necessary can be added, each time one is added a new search is automatically executed that updates the task grid.

The filters applied in the search are highlighted where they were entered, each followed by an × icon. By pressing such icon the corresponding filter is deleted and a new search that updated the task grid is automatically executed.

To remove all search filters, the icon must be pressed, which is located on the right of the last filter. This way the complete list of tasks is loaded again in the grid.

Filters

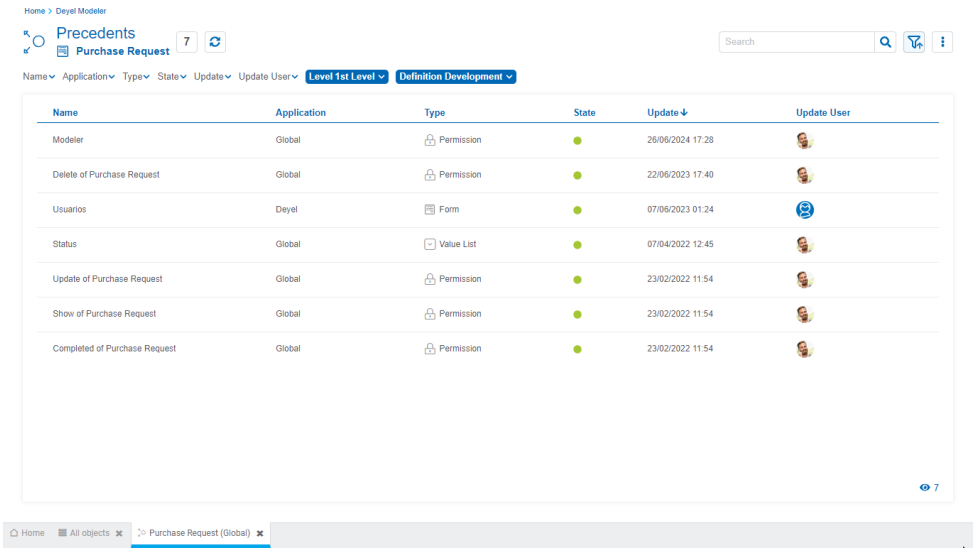
The filters available in the objects grid correspond to all its columns.

- Name
- Application
- Type
- State
- Update
- Update user

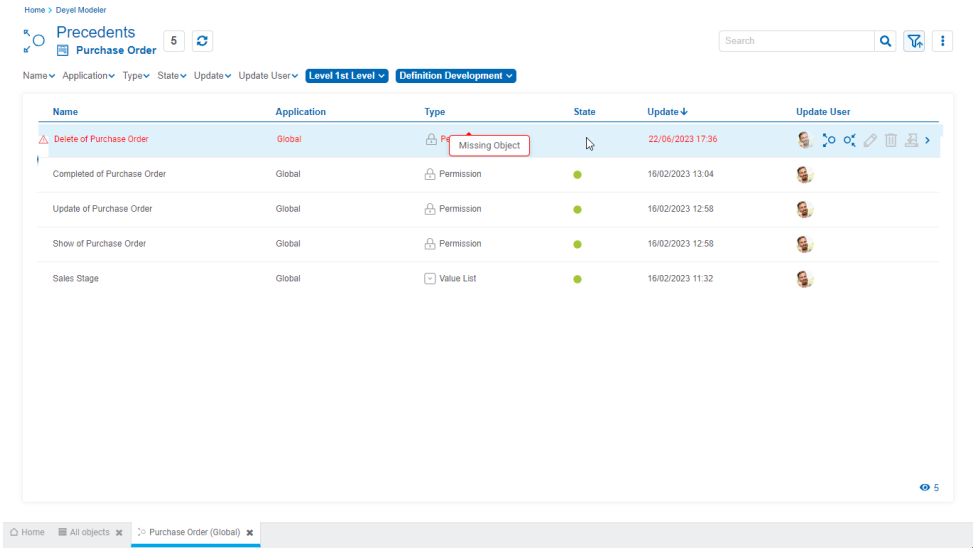
3.4.6.3. Precedents

The precedents grid allows to determine which are the objects that a specific **Deyel** object needs to execute.

The analysis of precedents is very useful to determine which objects are necessary for the **Deyel** selected object to execute correctly. Also, it allows to determine which objects are required to perform a complete export.



It may happen that one or more precedent objects are not found in the environment. In this case, the row is displayed in red with a warning icon.



Precedents may belong to the same object application or to other applications. Precedents show returns all precedent objects without applying security. It is possible that a user does not have permissions to see an object, but that does not prevent them from seeing the object as a precedent of another to which they have permission. The precedent objects on which the user does not have permission are displayed in the precedents grid without the possibility of executing any function on them (open, modify, export).

The precedents grid is very similar to the object's grid, but it has the following special features:

- The possibility of saving favorite searches is not available, as it is not necessary.
- In the event that a precedent is not found in the environment, it is displayed in red, with a warning icon and with the label “Nonexistent Object”. In this way, it is possible to determine which are the missing precedents of an object.
- It has 2 additional filters.

Additional Filters

The additional filters are "Level" and "Definition", they must always have a value and cannot be left blank. For this reason, the individual option to remove the filter is not available. However, you can use the option to remove all filters, which in the case of additional filters returns them to the initial value.

Level

It allows to filter the precedents of:

- 1st Level: this value is taken as the initial value.
- All Levels: it is a recursive precedence.

Definition

It allows to filter the precedents of:

- Development: this value is taken as the initial value.
- Execution

The concept of definition is different from the concept of state. The concept of state of a **Deyel** object indicates the stage of development it is in and if it is operational. On the other hand, the concept of definition allows objects to be modified without affecting the execution version of the object that is operational.

Internally, **Deyel** objects have 2 definitions: development and execution, which is completely transparent to users. The development definition is the one used when modeling an object, while the execution definition is the one that is operational.

If an object is in the "Modified" state, it means that the development definition may be different from the execution definition. This means that a **Deyel** object can be modified without affecting the execution definition. As both definitions are different, precedents can also be different.

If an object is in the "Published" state, it means that the development definition and the execution definition match. This means that the object was modeled and is in use. So the precedents of both definitions are the same.

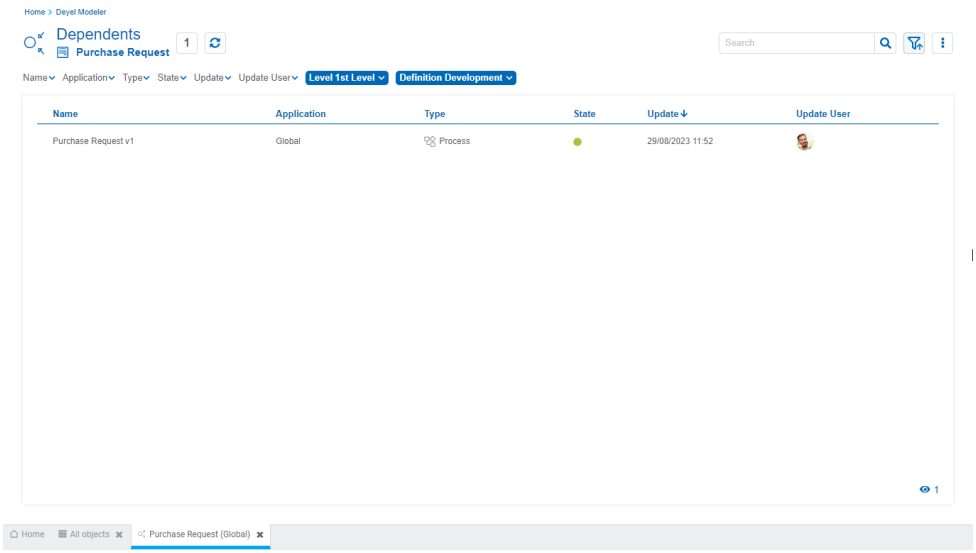
In this way, both definitions coexist, to allow new developments without affecting the published object that is operational. When an object is published and a change is made operational, the execution definition is updated with the changes coming from the development definition, making both definitions identical.

3.4.6.4. Dependents

The dependents grid allows to determine which are the objects from where a specific **Deyel** object is being used.

The analysis of dependents allows to determine in which objects the modification of a certain object impacts.

In the object's grid, the dependent icon is not available in the applications, because the applications do not have them. No object has the application as precedent, because objects are contained in an application, but do not use it directly to execute.



The dependents grid works in a similar way to the precedents grid.

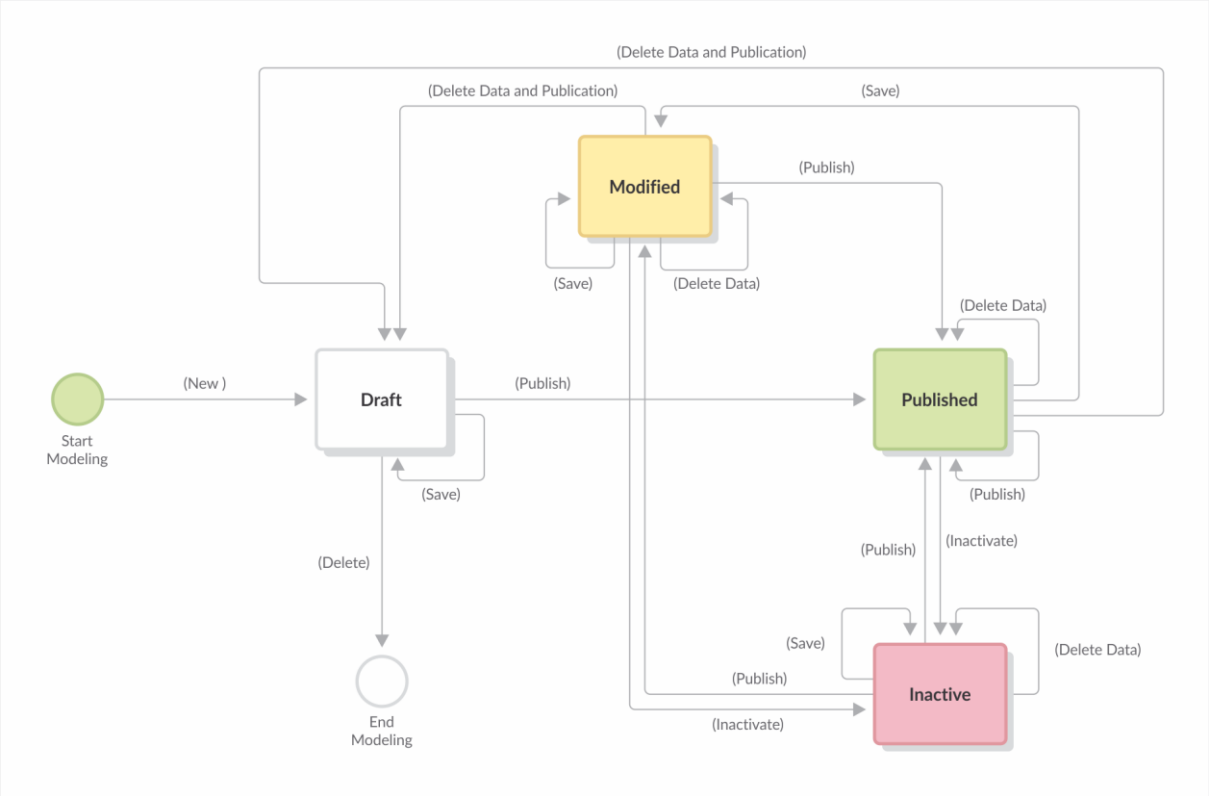
3.4.7. Object States

States indicate the stage of development in which **Deyel** objects are and if they are operational.

The following **Deyel** objects defined in applications have states:

- Form
- Agile Form
- Process
- Rule
- Dashboard
- Widget
- Value List
- Adapter
- Report

The possible states and the events that produce transitions among them are described below, ensuring the modelers of predictable behavior for each type of object.



Possible States

Draft
State corresponding to the beginning of the modeling or when it was saved but not published.

Published
State of the object once published, that is, it is available for use.

Modified
If an object is modified when it is in the "Published" state, it becomes "Modified".

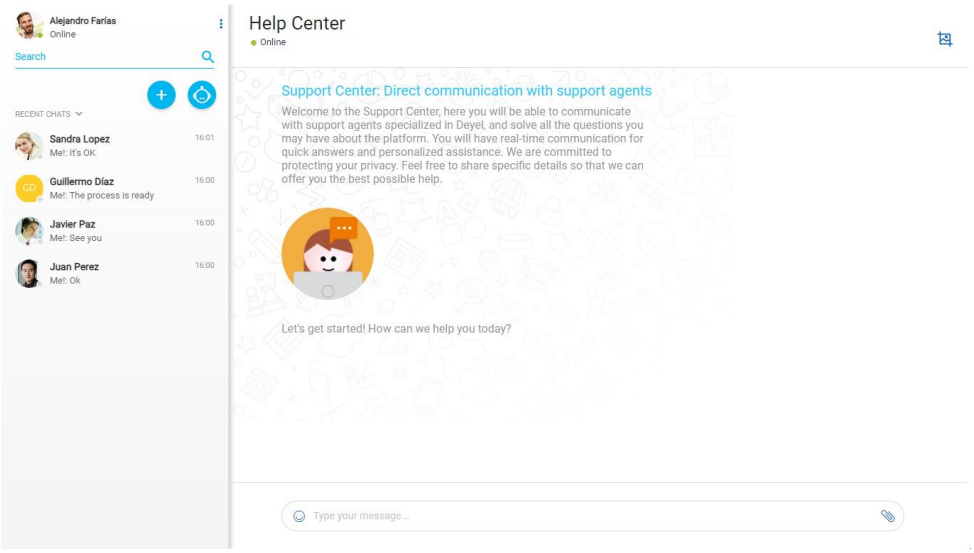
Inactive
State corresponding to a published object for which only instances or cases can be shown.
As of processes, forms and agile forms, instances or cases of objects that are in this state cannot be created, delete or modified. On the other hand, rules can move to inactive state, if they are not being used in another object, that is, since they are inactive, they cannot be used in the modeling of another object.

> *Available in future versions.*

3.4.8. Deyel Bot

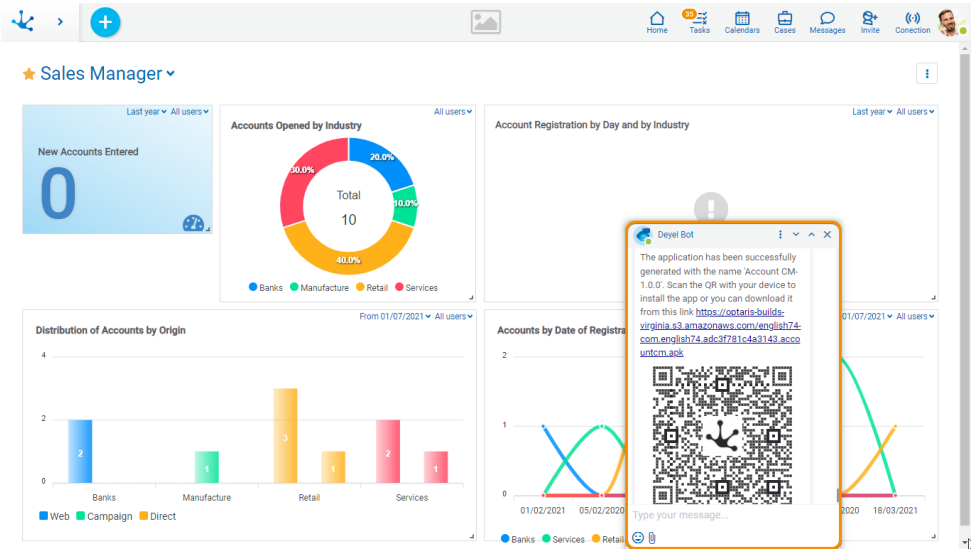
Deyel Bot is a user assistance tool in using **Deyel** that is characterized by simultaneously fulfilling two particular roles or functions. On the one hand, Deyel Bot is a chatbot whose goal is to keep the user

informed about the operations they carry out in **Deyel**, and on the other hand, it is a virtual wizard that helps resolve all inquiries about the **Deyel** platform.



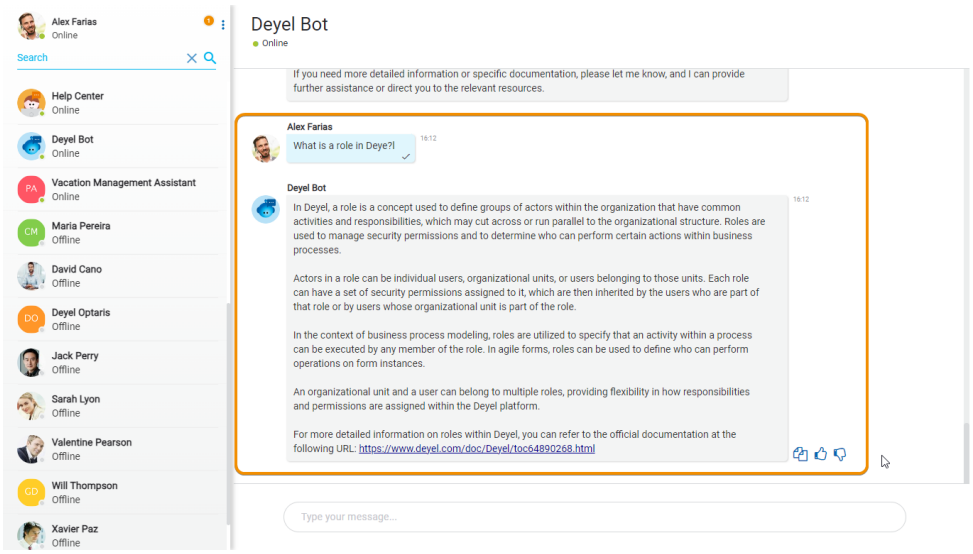
As a chatbot, Deyel Bot receives notifications about the operations carried out, such as:

- The generation of a mobile application for Android or IOS.
- A report execution.
- The upload of execution logs to the application execution console.
- When inviting a user, Deyel Bot notifies the result of the invitation to the user who generated it, that is, whether the invited user accepts or cancels the invitation, indicating the date and time.



In turn, Deyel Bot is a virtual wizard called Deyel Expert. It is available in full-screen messaging. It is based on ChatGPT and designed to provide fast and accurate responses about **Deyel**. It avoids wasting time searching through documentation or requesting specialized consulting. Deyel Bot is always active and available to all users, it has been trained with all the documentation of **Deyel**, offering responses based on deep and always up-to-date knowledge.

Deyel Bot saves the history of queries and conversations with the user, maintaining the context of the conversation so that the user can follow up on a response and keep the thread of the conversation.



When hovering over the response provided by Deyel Bot, three icons are displayed: , and .

For each response from Deyel Bot, the user has the opportunity to rate it and provide a comment with the aim of giving feedback to Deyel Bot, helping to improve its functionality. The rating can be positive or negative. If it is negative, a comment to expand or explain the reason for the rating can be entered.

For modeler users, when the response rating is negative, the option to access a new chat with a Support Center agent becomes available. Here, the user can have a conversation and interact with a person to address their query. Once the user's query has been resolved, the chat with the Support Center will be closed. To start a new conversation with the Support Center, the user must first attempt to resolve their issue again through Deyel Bot. If unsuccessful, they can then initiate a new conversation with a Support Center Agent.

It should be noted that, like all AI-based technology, Deyel Bot's responses should be considered suggestions that may have some degree of deviation. However, most of the time, the responses provided are very accurate, precise, and truthful.

3.4.9. Applications Modeling

Applications Modeler

An application is the main object of all modeling, as it is the final product that must address a business requirement.

The applications modeler allows for easily and intuitively modeling independent applications. It allows designing the navigation of applications through the creation of a diagram, where the navigation flows are defined.

From the applications modeler, each of the objects that compose it can be accessed, opening in new tabs of the modeler.

To create **Deyel** applications quickly and without prior knowledge, you can start from application templates, from data, or from Diana AI.

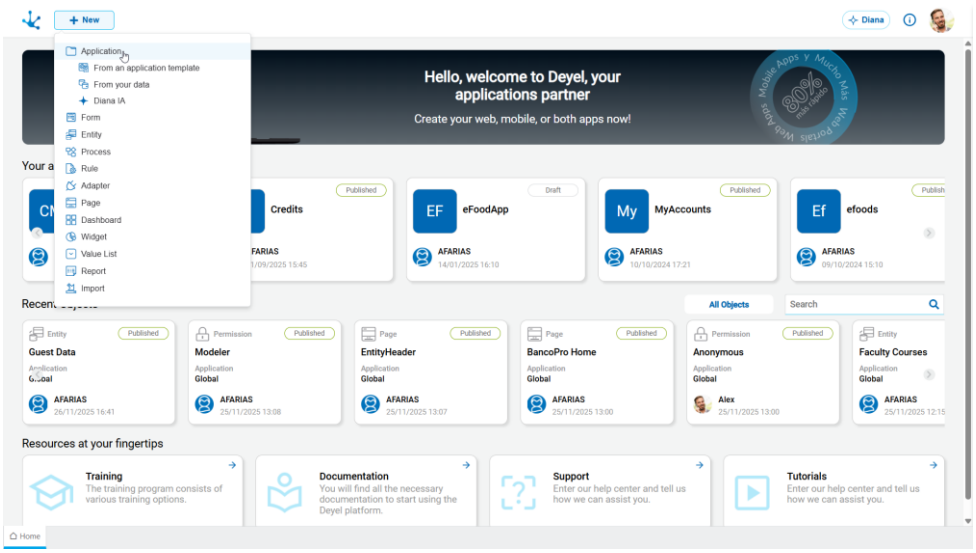
Templates help build B2B (Business-to-Business), B2C (Business-to-Customers) applications, or a wide range of other applications based on specific business concepts.

B2B applications are applications that establish a commercial relation with another company or can be used internally within the business. For example, a purchase order authorization process or any type of authorization within the business.

B2C applications are applications aimed at an end consumer and must have an interface that encourages them to choose to use the application. These applications can have a large number of users and sometimes their use may depend on events, such as, for example, a “black Friday”.

With **Deyel** , all types of applications can be developed. Web applications can be developed to be consulted from desktop computers or mobile devices, mobile applications that can be installed on cell phones or tablets or responsive applications that can be used as web and mobile. The mobile applications can be executed on both Android and iOS, generating applications for each operating system, or they can be PWA (Progressive Web Application).

In order to ensure that all applications have a consistent, usable interface aligned with the visual identity defined in the organization, the **Deyel** modeler incorporates a design system. This system allows centralized management of visual elements, such as text, buttons, colors, fonts, among others, so that any adjustments made are automatically reflected in the application.





This button is used to create an application from some of the following options:

 An application template.
 Your data.
 Diana IA.

The general features of the applications modeler and the main elements that compose it are described in the following topics:

- Modeling Facilities
- General Properties

3.4.9.1. Modeling Facilities

The general characteristics of this modeler are specified below.

New Application

The modeler user defines a new application, that after being published can be accessed from the access path configured in the application properties.

When creating an application, the method of modeling it must be selected, thereby accessing the applications modeler, which comprises various sections.

Ways to Create the Application

- From a Template
  Allows you to create an application from a predefined template based on business needs.

- From Data
  Creates applications based on structured information from a business entity, such as accounts or surveys.

- From Diana
  Uses the application modeler and the **Diana** assistant, which generates the application in an AI-assisted manner.

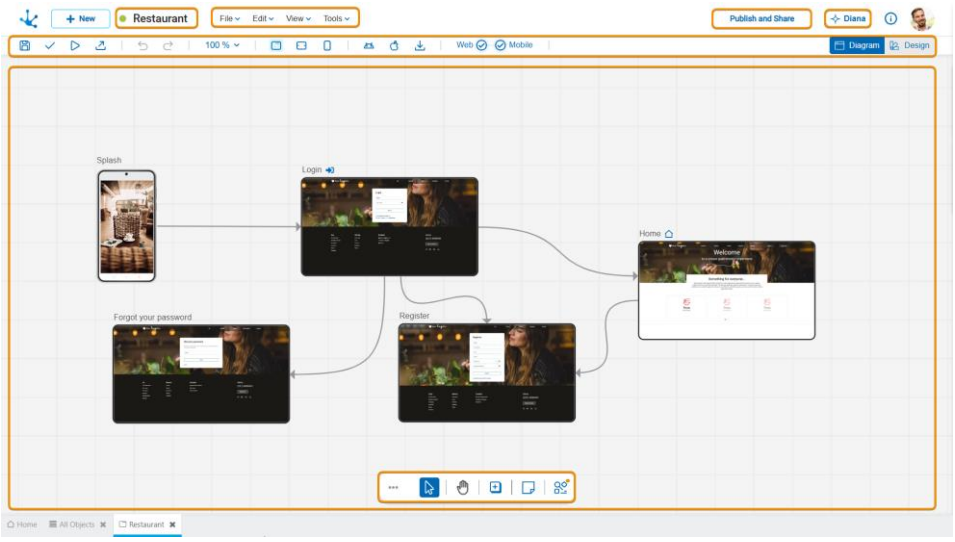
Workspace Sections

- Application Information
  - State
  - Name

- Publish and Share

When clicking on the the “Publish and Share” button, a panel opens with different options. If the application is not published, it allows modifying the property Access Path and click the “Publish” button".
If the application is already published, the following is displayed:
- A QR code to scan and display the application on a mobile phone or tablet.
- The application URL, accompanied by two icons: to execute the application and to copy the URL

- Expanded Menu
- Diana
- Top Toolbar
- Components View
- Characters and User Stories View
- Graphic Modeling Area
- Floating Palette

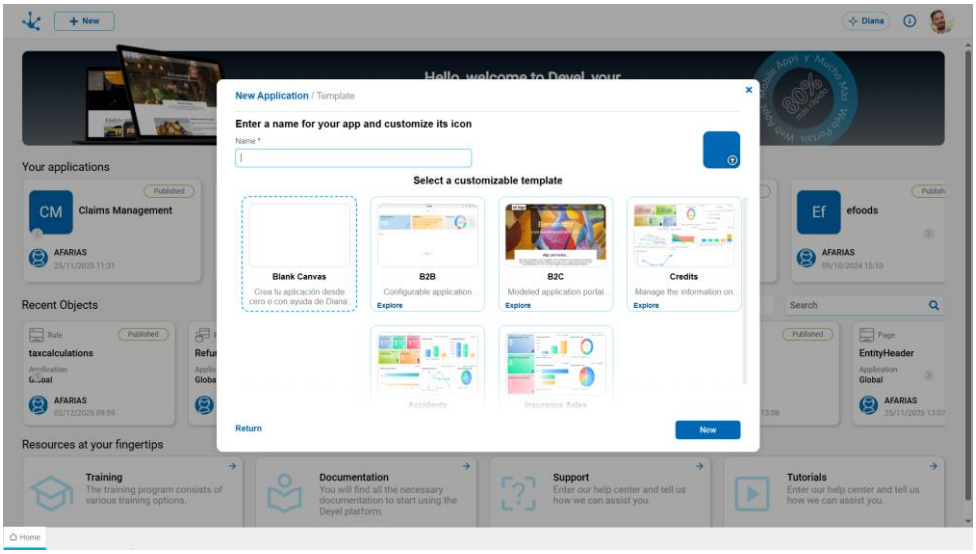


3.4.9.1.1. Create from a Template

When choosing to create an application from templates, fully functional applications of various themes and businesses can be modeled, allowing them to be used directly or customized to fit specific requests.

Templates offer a solid and flexible foundation to start the development of applications. They are functional solutions designed to be extended and adapted according to the requirements of each business.

When building an application, it opens a panel that allows to define some of its general properties must be defined, such us Name and its Icon, and select a built-in template. Once the template is selected, the applications modeler opens.



Built-in Templates

A template type can be selected according to the business requirement.

Hovering over each template displays its features. If 'Explore' is selected, a panel opens with its description and additional details.

B2B (Business to Business)

These templates are aimed at improving back-office operations. They provide generic functionality that allows developers to immediately start working on creating applications, tailored to business-to-business needs. They are ideal for those seeking to optimize internal processes. B2B templates are an ideal starting point for developing robust and scalable solutions.

B2C (Business to Customers)

These templates are designed for businesses that interact directly with consumers. They offer generic functionalities that facilitate a quick start to development. They are specially adapted to improve the customer experience, facilitating the creation of portals and applications that improve the interaction and satisfaction of the final consumer.

Credits

Allows the management of information on the credits requested through a configurable and customizable portal, designed to be used immediately or customized according to specific requirements. This solution is ideal for financial institutions that wish to implement tools tailored to their needs for credit management.
Using this template provides an initial framework that significantly facilitates and speeds up the development and implementation processes.

Claims

Enables agile and efficient management of claims reports through a configurable portal that can be adapted to the needs of the business user. It contains home pages designed for call center users and claims analysts, with dynamic menus that adjust to the permissions of each profile. Thus, each user has simple access to the information and tools necessary to optimize the comprehensive management of complaints.

Insurance Sales

Enables agile and efficient management of insurance sales through a configurable portal that can be adapted to the needs of the business user. It contains home pages designed for insurance producers and subscribers, with dynamic menus that adjust to the permissions of each profile. Thus, each user has simple access to the information and tools necessary to optimize the comprehensive management of insurance sales.

3.4.9.1.1.1. B2B

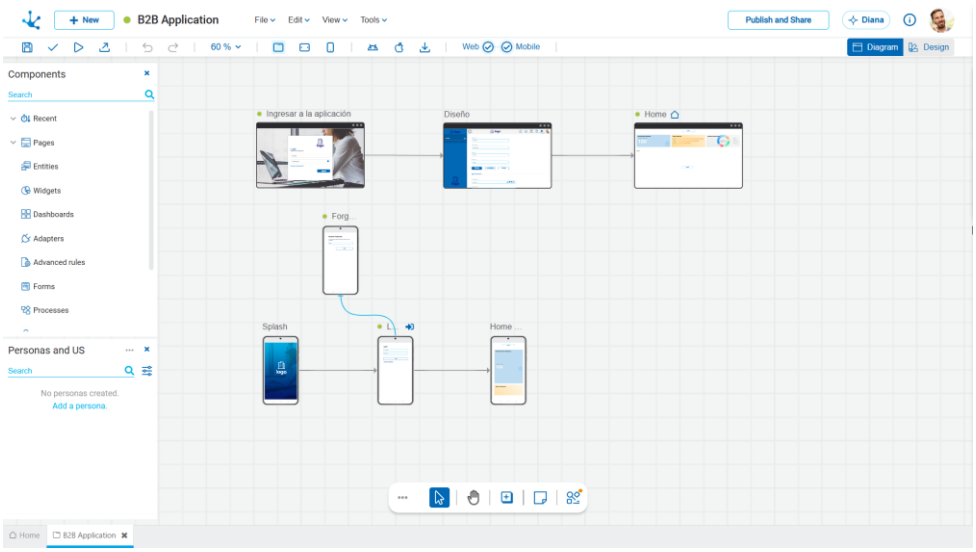
B2B applications establish a commercial relation with another company or can be used internally within a business.

Selecting a B2B template creates an application with a set of built-in elements. In this type of application, the login page, the home page and, through the “Design” element, the top toolbar and the side menu can be customized.

This type of application consists of three pages with a specific configuration:

- Applications Login
- Design
- Mobil Home with Tabs

Because their functionality is generic, they cannot be freely modeled and only a limited set of properties can be configured.



The “Home” page offers an immediate and accessible overview of the most relevant information. A section is highlighted here to include progress widgets that allow monitoring of the current state and development of application entities. It is important to note that the instances comprising the total displayed in each graph can be shown by simply clicking on the graphic section of the widget of interest.

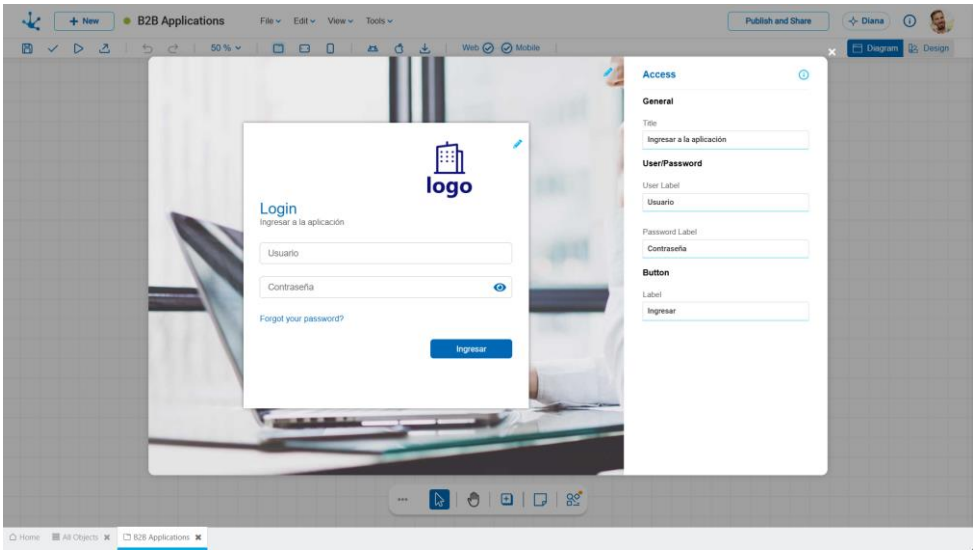
On this same page, elements can be modeled with references to the most recent actions, making it easier to resume pending tasks or review recent operations without wasting time.

To ensure a complete user experience, the application includes pages such as "Forgot password" and "Change password" as well as a "User profile" area with personal information and account settings.

The user interface has been designed to ensure a fluid and adaptable experience on any device. Whether accessed from a desktop computer, tablet, or mobile device, the application adjusts to offer the best possible display and functionality.

Application Login

A set of specific properties can be configured on this page to model the login.



Title
It allows defining the text displayed at the top of the page, below the "Login" text.

User Label
It allows redefining the indicative text of the content to be entered in the field corresponding to the user for each application.

Password Label
It allows redefining the indicative text of the content to be entered in the field corresponding to the password for each application.

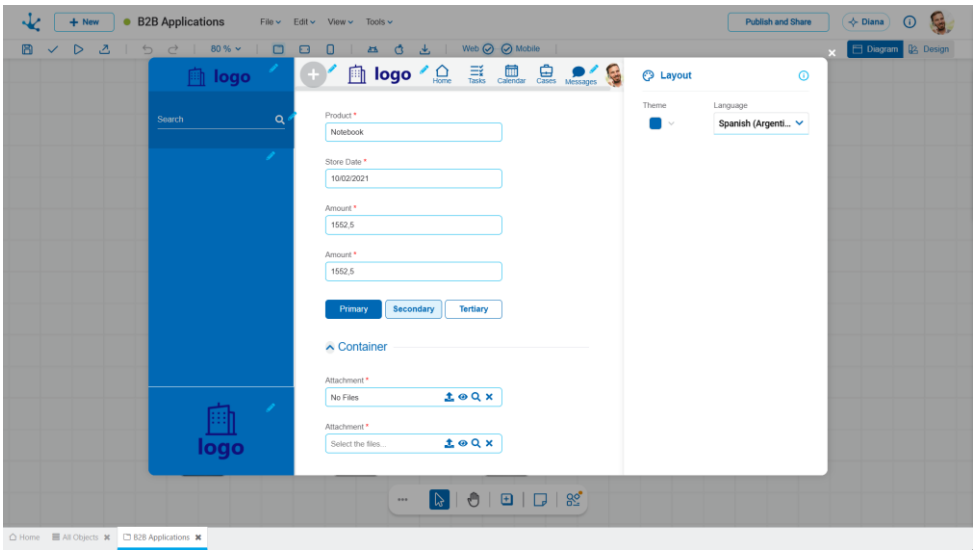
Label
It allows redefining the label text of the button used for logging in.

Logo
It allows the inclusion of a logo on the application login page, which is important for reinforcing brand identity.

Background Image
It allows customization of the visual appearance of the page.

Design

The "Design" page allows configuring various visual and functional aspects of the application through a set of properties.



Theme
It is used to select the overall visual appearance of the application, from a set of pre-built themes. Themes define the visual aspects that impact the aesthetics of the application, such as primary and secondary colors, text fonts, and button styles.
In the central section, a preview of the user interface can be seen, displaying the selected theme applied to the interface elements. This helps in making decisions regarding the customization of the application.

Language
It allows selecting the default language to use in the user interface. Although a user can choose in which language the application is presented, if they do not, the interface language is determined by the one configured in this property.

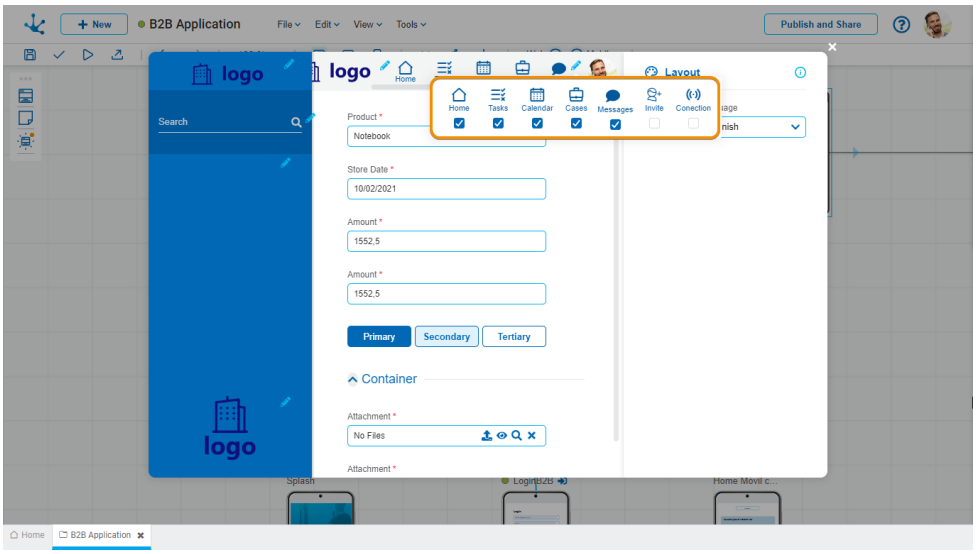
Logos
A set of properties related to the logos used in the web portal can be configured. Including:
- Top left logo
- Bottom left logo
- Top center logo

They allow adding corporate or brand logos within the user interface, by selecting the icon .

Context Menu
It allows modeling the business operations that the user has enabled for execution, by selecting the icon . For example, starting a process or entering data into a business entity. It is optional for applications to have this icon enabled.

Top Toolbar

In the top toolbar, the icons used as shortcuts to default platform functionalities can be activated or deactivated.



Bar Elements

Home It allows direct access to the "Home" page of the application.

Tasks Redirects to the management page of tasks of **Deyel**.

Calendar It allows access to the page where the tasks and relevant information are displayed on a calendar interface.

Cases Redirects to the management page of cases of **Deyel.**

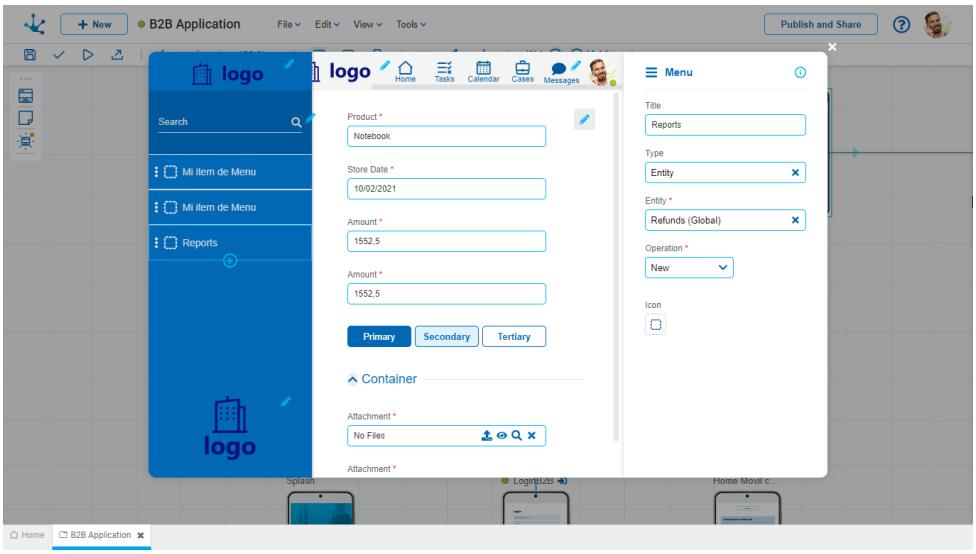
Messages It allows activating or deactivating the chat functionality, facilitating instant messaging within the platform.

Invite It allows expanding the user network of an application, simplifying the process of adding new members who use the application.

Conection It allows monitoring the state of the user's local network, which is useful to evaluate the performance of the application.

Side Menu

It provides a set of tools to design a simple and well-structured navigation of the application. Selecting the icon enables access to the editing, creation, and sorting of menu items that allow for configuring a set of properties.



Title

Defines the text that is displayed on each menu item.

Type

Specifies the type of object related to the menu item.

Possible Values

- Page
- Deyel Page
- Entity
- Process
- Link

Object

Each of the above options allows linking the menu item with different objects within the application. For example, if the chosen type is 'Page', a specific page of the application can be selected. If, however, the type is 'Entity', a business entity can be chosen.

Operation

Defines the operation that is performed when the object is selected.

Possible Values

- Create
- Grid

Icon

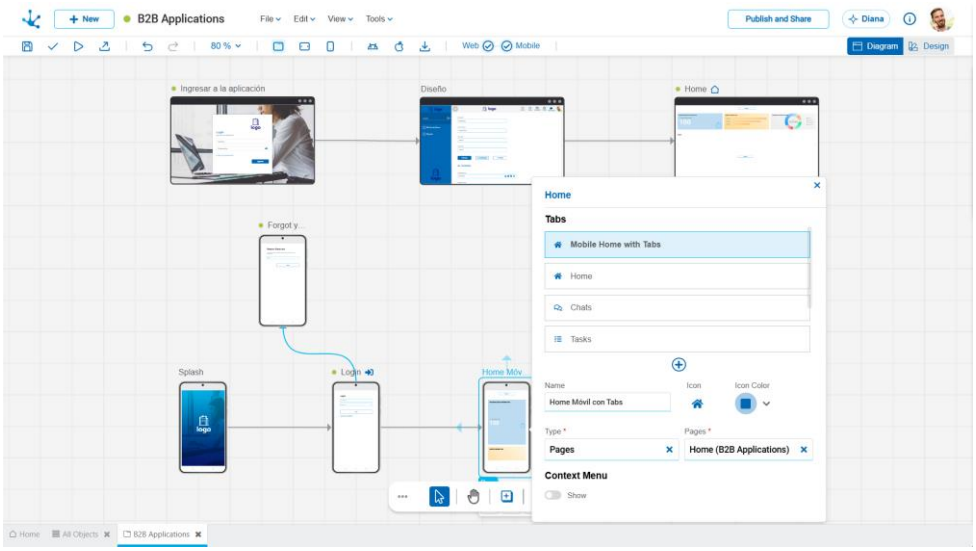
It graphically describes the functionality to which it is redirected, improving usability.

Mobile Home with Tabs

The 'Mobile Home page with Tabs' offers a set of configurable properties to design and customize the navigation tabs of the mobile application.

There are predefined tabs whose redirection targets cannot be modified, as they are related to pages with generic platform functionality.

Selecting the icon of each tab, they can be hidden or made visible, while selecting the icon also allows the creation of a new tab.



Name

It allows defining the title of the menu item associated with each tab. These titles should provide a clear and concise description of the content or functionality associated with the tab.

Icon

It allows selecting a visual icon that represents that content or functionality, enabling quick identification of the purpose of each tab.

Icon color

It allows customizing the color of the selected icon.

Type

It is used to define the type of object related to each tab.

Object
Each of the above options allows linking the tab with different objects within the application.

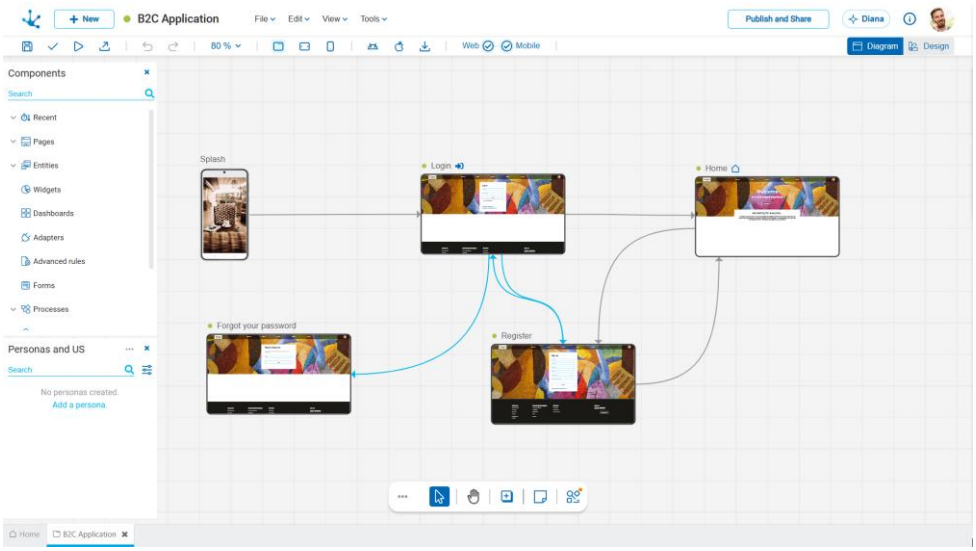
Show
It allows defining whether the context menu is enabled or not.

3.4.9.1.1.2. B2C

B2C applications are applications aimed at an end consumer and may have a great number of users.

By selecting a B2C template, an application is created composed of elements that together create a complete and functional user experience from the first moment.

This template aims to present the company and showcase a main product to its customers. The product features are generic, and can be adapted according to the real business requirement.



The "Home" page is designed to capture the user's attention and provide easy access to the main areas of the site. In the first contact with the user, the image and brand of the company are established.

The application features a section to list products at the bottom of the home page.

In terms of user actions, the template includes login pages, registration and password recovery pages, as well as a user profile where customers can manage their data and preferences.

Headers and footers are designed to provide consistent information across all pages, such as navigation links, contact details, and copyright information. These headers and footers can be maintained centrally in the model, as reusable pages, with changes automatically reflected on all pages that use them.

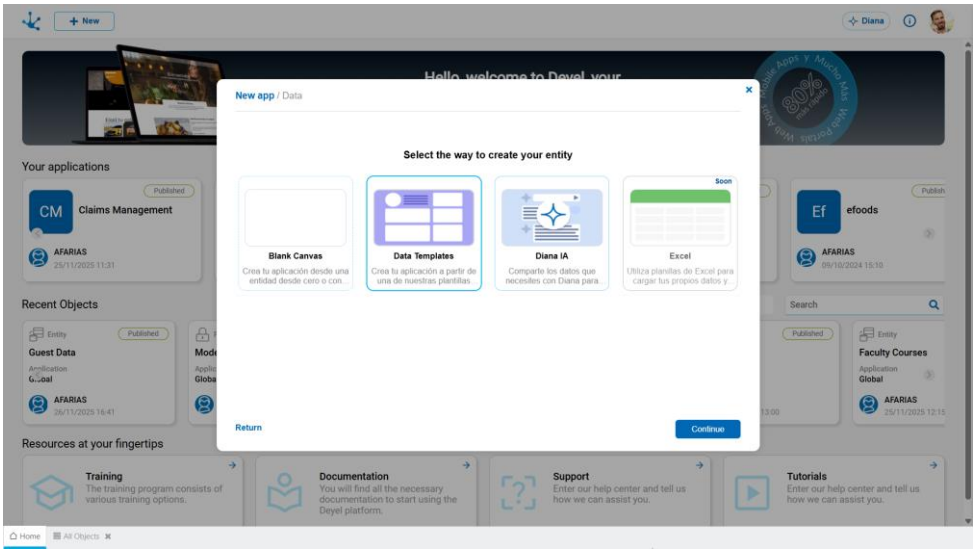
The user interface is designed to be responsive, ensuring that the application functions and works perfectly on desktop, tablets and mobile devices.

3.4.9.1.2. Create from Data

There is another approach to the way of creating applications and it is called “Data-Driven Applications”.

These applications are those capable of managing detailed and structured information about a specific business entity, such as a CRM account, or a survey in an internal evaluation application. They have a central business entity and a set of supporting objects related to this entity.

When choosing to create a data-driven application, entities that represent the data can be designed, starting from a blank canvas, using entity templates, from Excel spreadsheets, or by asking Diana to model what is needed.



Ways to Create the Application

The different ways to create a data-driven application can be selected.

Template

Contains all the objects necessary to administrate a business entity. Templates are made up of pages to navigate the application, a main entity, and other auxiliary objects such as dashboards, widgets, and value lists. All these objects can be modeled and the user can modify and extend them.

Once this option is selected, a panel with the different available templates opens:

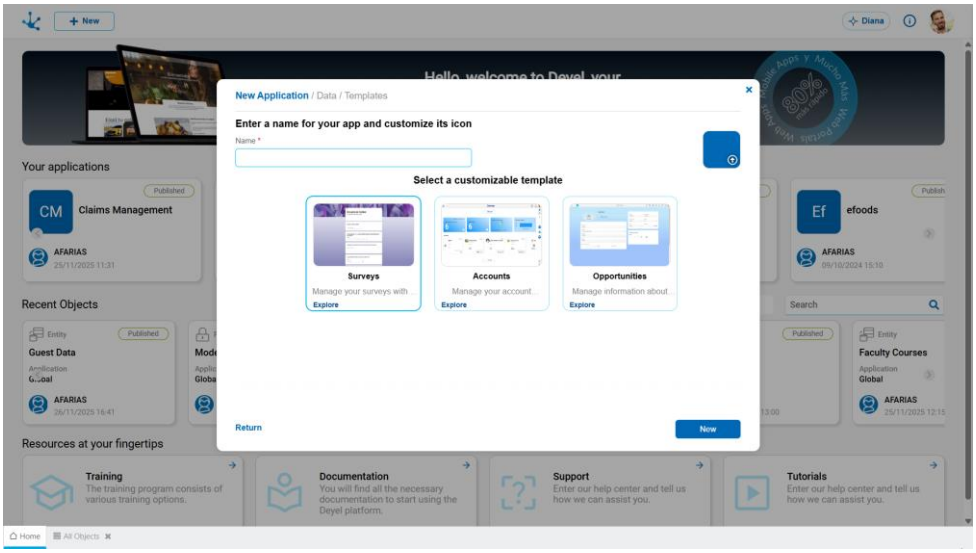
Surveys
Allows the management of surveys through a configurable and customizable portal, ready to use or customize to specific requirements.

Accounts
Facilitates account information management through a configurable and adaptable portal, designed for immediate use or customization as needed.

Opportunities
Offers tools to manage customer opportunity information through a configurable and flexible portal, ready to be used or adjusted according to specific needs.

Hovering over each template displays its features. If 'Explore' is selected, a panel opens with its description and additional details.

Once a template has been selected, some of the following general properties must be defined, such us Name and its Icon, to then access the applications modeler.



Diana AI

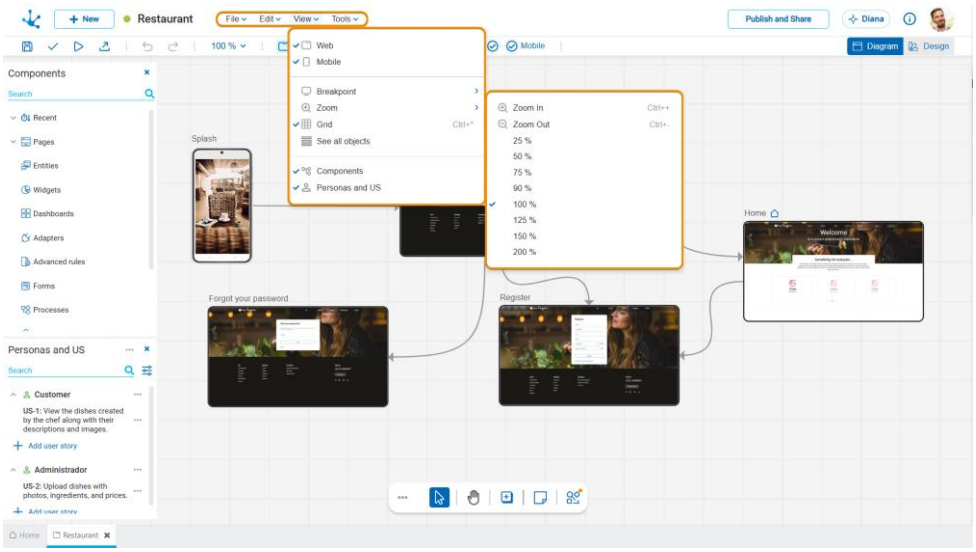
Opens the applications modeler and the panel that allows starting a conversation with Diana to generate the application in an assisted manner.

Excel

This option is ideal for those who already have the attributes of an entity defined in an Excel spreadsheet. The Deyel modeler can take this file and automatically build an application, which closely reflects the entire data structure provided. It is a quick and efficient way to convert a simple Excel definition into a functional and robust application.

3.4.9.1.3. Expanded Menu

It is a horizontal list of options containing vertical submenus with different operations on the application or on its modeling. In turn, each option in a submenu may expand a dependent submenu.

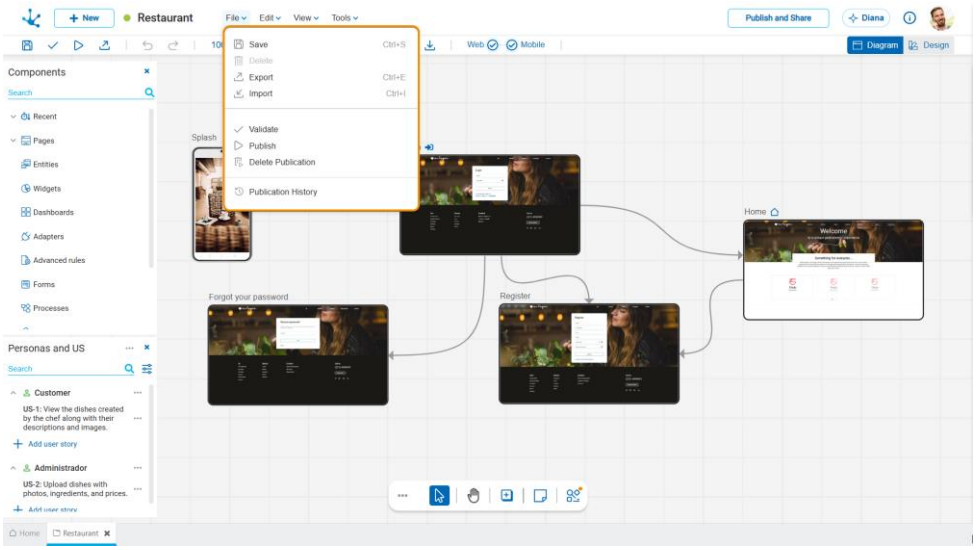


The expanded menu is made up of:

- File Submenu
- Edit Submenu
- View Submenu
- Tools Submenu

3.4.9.1.3.1. File Submenu

This submenu opens by clicking on the “File” option and allows to perform operations on the application.



Save

This icon allows to save the object in the repository of **Deyel**, leaving its state as "Draft" or "Modified". If certain conditions are met, the modeler user receives a message indicating that the operation was performed correctly, otherwise they receive an explanatory message.

Condition

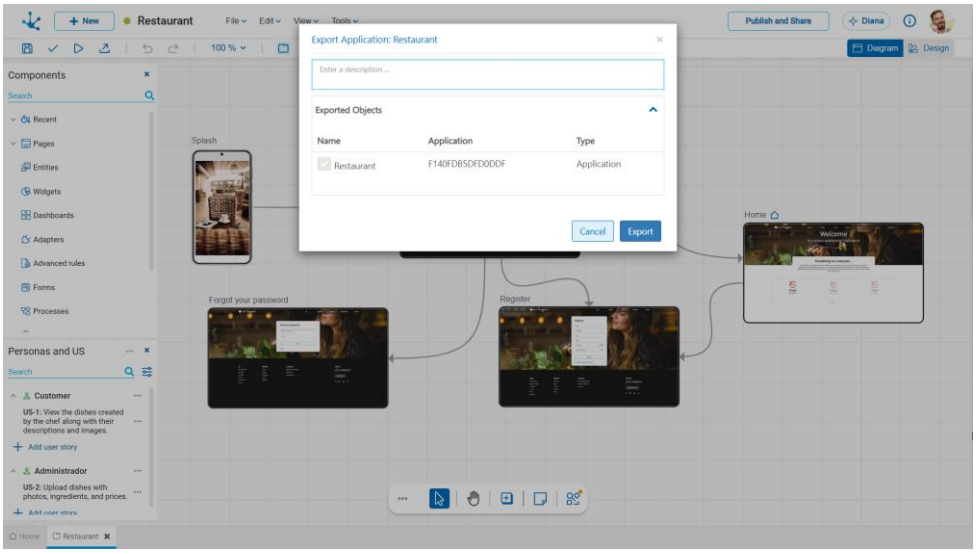
- Name must be unique.

Delete

It allows to delete the object only if it is in "Draft" state and does not have other associated objects of **Deyel** that were previously saved or published. When deleted, the tab it is on closes and the object is removed from the modeler's grid.

Export

This icon opens a window for the user to select and confirm the export of the object.



Properties

Description

In this property a text explaining the reason for the operation can be entered.
This text can be modified upon import and is displayed in the description column of the export record.

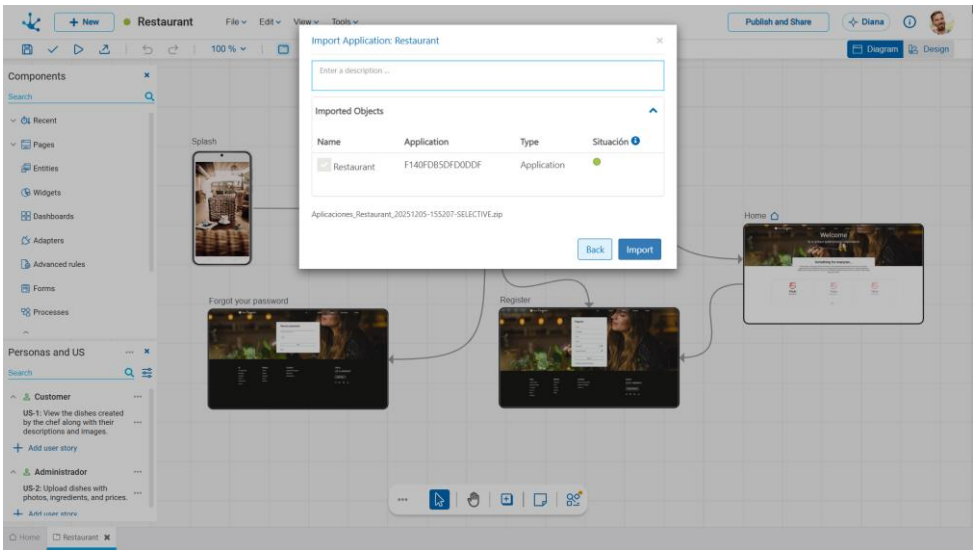
Exported Objects

By expanding the container, the object being exported is shown.

Press the "Cancel" button to undo export or press the "Export" button to finish.

Import

It allows to open a window for the user to select and confirm the import of the object.



Performing this operation is equivalent to using the import from the modeler's context menu.

Validate

This icon allows to validate if the object is ready to be published, that is, the same validations are carried out as when publishing and the result of them is reported.

Publish

Through this icon the object changes to "Published" state and the modeler user receives the corresponding message, indicating the result of the operation. The condition for publishing is the same as for the "Save" operation.

Delete Publication

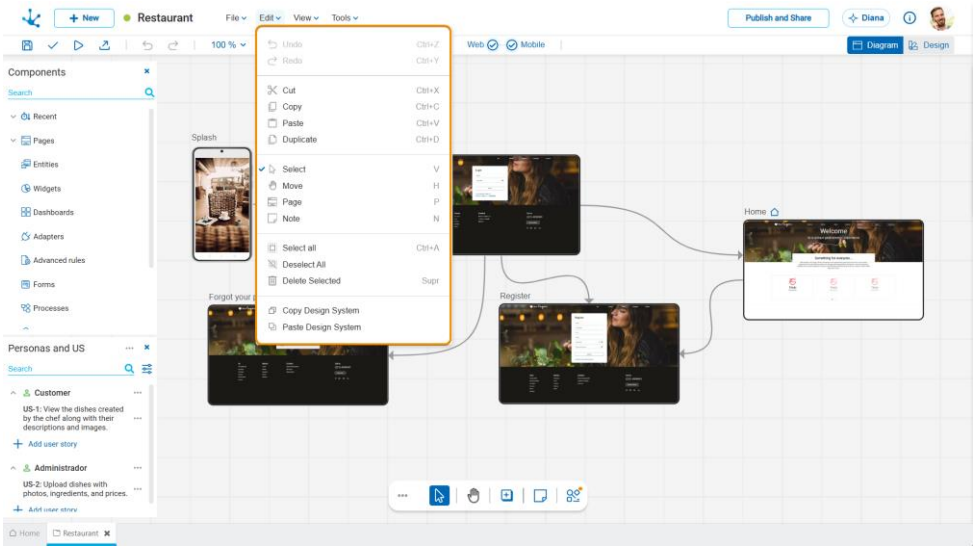
Allows to leave the object unavailable for use by returning it to the "Draft" state.

Publication History

This icon allows managing versions of published objects, displaying their history, to have better control of the changes that each object has.

3.4.9.1.3.2. Edit Submenu

This submenu is opened by clicking on the “Edit” option and allows the performance of operations within the modeler.



Undo (Ctrl+Z)

Allows to easily reverse the changes made in the modeler.

Redo (Ctrl+Y)

Allows to easily reapply the changes made in the modeler.

Cut (Ctrl+X)

Allows to send the selected element to the clipboard while deleting it from the graphic modeling area.

Copy (Ctrl+C)

It allows to copy any application element and temporarily place it on the clipboard.

Paste (Ctrl+V)

It allows to take the item from the clipboard and place it elsewhere in the graphic modeling area.

Duplicate (Ctrl+D)

It allows to make a copy of any application element within the graphic modeling area.

Select (Ctrl+V)

The selection mode allows elements in the modeling area to be selected and edited, and it is the default mode. While active, it can be temporarily switched to navigation mode by holding down the space bar or by right-clicking and dragging the cursor.

Move (Ctrl+H)

The navigation mode allows navigating the modeling area without using scroll bars, making it easier to explore the application. To move, simply click and drag the cursor.

Action (Ctrl+A)

Creates an action in the graphic modeling area to represent an operation that is executed within the functional flow of the application.

Note (Ctrl+N)

Creates a note in the graphic modeling area to include brief comments about the modeling.

Select All (Ctrl+A)

It allows selecting all the elements of the application.

Deselect All

Allows the selection of all application elements to be cleared.

Delete Selected Elements (Del)

It allows to delete all selected elements from the graphic modeling area.

Copy Design System

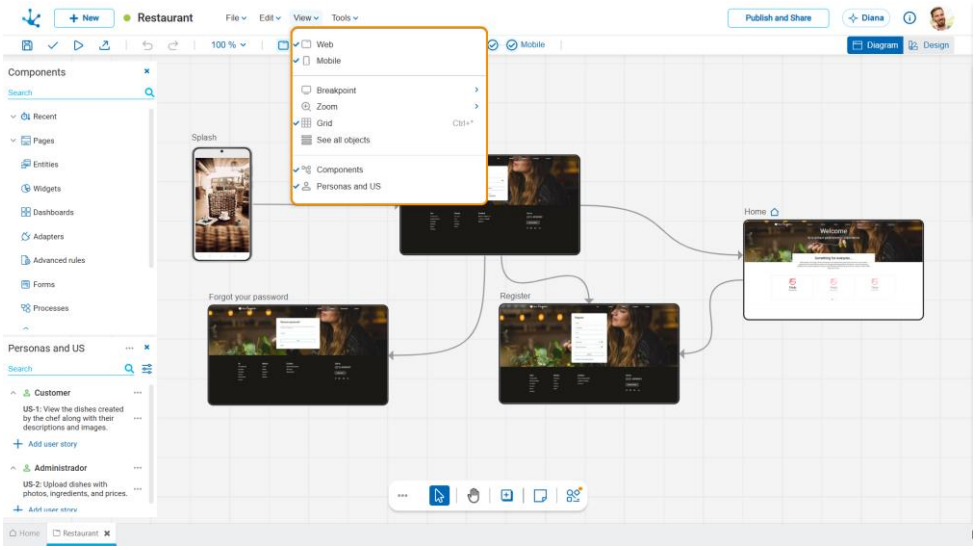
It allows copying all styles, components, and configurations from the current design system to incorporate it into another application.

Paste Design System

Allows the design system copied from another application to be applied, preserving the same styles and properties.

3.4.9.1.3.3. View Submenu

This submenu opens by clicking on the "View" option and allows to select different ways of displaying the application.



Web

When selecting this option, only web-type pages are displayed.

Mobile

When selecting this option, only mobile-type pages are displayed.

> *Both options can be selected at the same time.*

Breakpoint

By clicking on “Breakpoint”, a secondary submenu is displayed, where the following options can be selected:

- Desktop Computer
- Tablet
- Mobile

Zoom

Selecting "Zoom" allows adjusting the viewing size of the modeling area, using built-in options such as 25%, 50%, 75%, 90%, 100%, 125%, 150% and 200%. The shortcut Ctrl+ to zoom in and Ctrl- to zoom out can also be used. Another option to adjust the size is to hold down the Ctrl key and use the mouse wheel.

Grid

This option provides grid lines that allow to perfectly align elements within the graphic modeling area.

View all items

When this option is selected, it automatically switches from the applications modeler tab to the tab displaying the grid of all objects.

Components

By activating this option, the components are displayed in the left side panel of the modeling area.

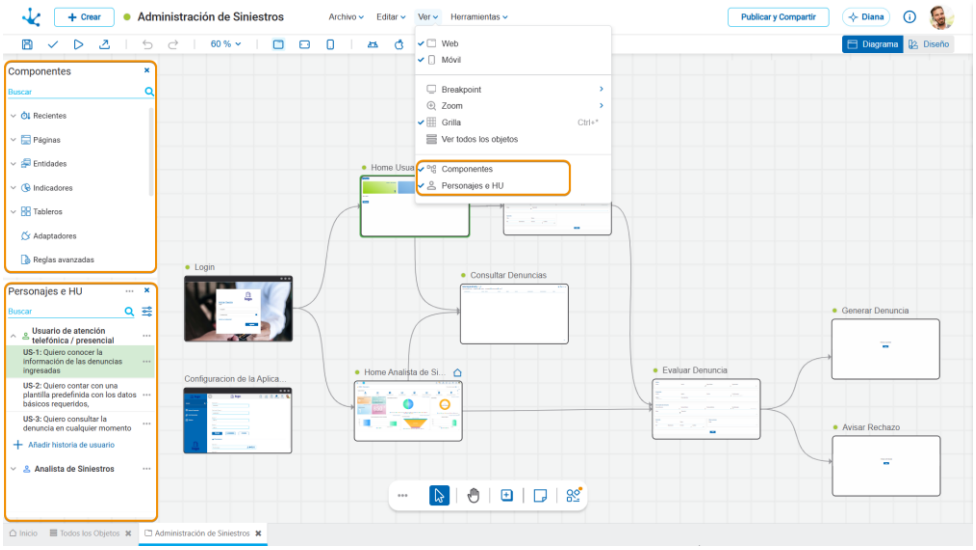
Personas and US

By activating this option, both personas and user stories are displayed in the left side panel of the modeling area.

Components, Characters, and User Stories

The components, personas, and user stories views allow you to organize, visualize, and manage the application's elements in a tree structure within the left sidebar of the applications modeler.

They are only displayed if the "View" submenu options of the applications modeler are activated.



Component View

It includes all the application's objects organized by category, making them easy to browse and providing direct access.

- Recent
  Displays the latest saved or published objects.

- Types of objects
  Groups objects according to their type, such as entities, pages, and processes.

Functionalities

- Search bar
  It allows filtering objects by name within any category, speeding up their location without having to navigate through the entire structure.

- Direct access
  By double-clicking an object's name, it opens directly in the corresponding modeler.

Personas and User Stories View

This view allows the display and management of the personas and their respective user stories. It facilitates the traceability of the functionalities associated with each profile within the application.

What are personas and user stories?

- **Personas**: They represent the different types of users who interact with the application. Each persona has a specific role with associated permissions, needs, and functionalities.

  Example

Insurance producer, Client, Administrator.

- **User stories (US)**: These are brief descriptions that express what a user wants to achieve with the application. They are fundamental to defining functional requirements from the user's perspective. They usually follow this structure:

  "As a **[persona]**, I want to **[perform an action**] to **[achieve a goal]**"

  Example:
  "As an insurance producer, I want to register the client's data and the risk to be insured date in order to start the request process."

Functionalities

- Creation of new personas
  Next to the title "Personas and US", the icon ••• displays a menu with the option "Create new persona". By selecting it, a panel opens that allows the input of the name, color, and description of the new persona.

- Display and creation of stories
  When hovering over a persona, their user stories are displayed. After the last story, the icon + with the "Add user story" option is displayed. When clicked, a panel opens to enter the properties and define the structure of the new story.

- Search bar
  It allows searching for personas by name, as well as user stories by name or description, making it easier to find them.

.

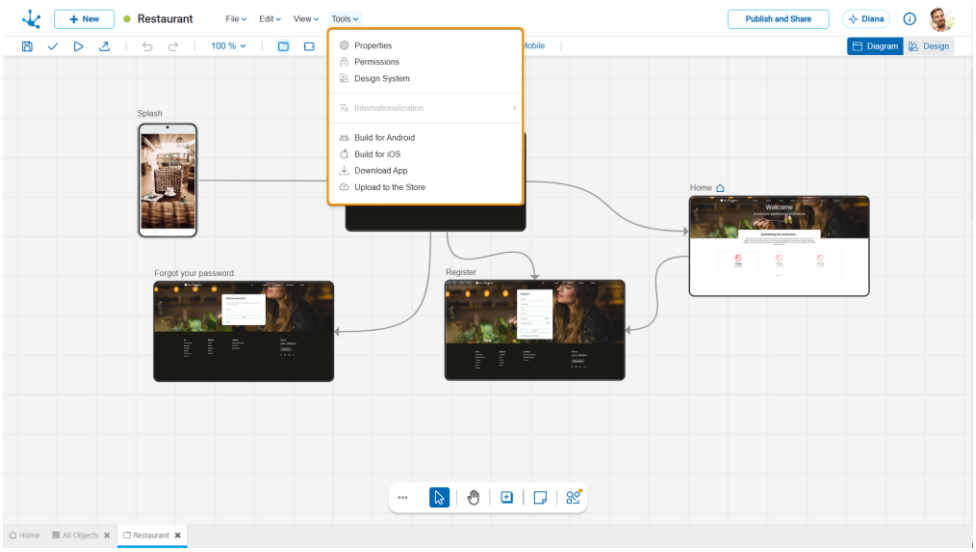
- Display filters
  Clicking on the icon ⚙, it is possible to apply filters to show only implemented stories, unimplemented stories, or all stories.

- Editing and deleting elements
  The icon •••, visible next to each user story or persona, provides access to a menu with options to edit or delete.

- Highlighting of linked stories:
  When clicking on a user story, it is highlighted with a color. If the story is already implemented in the application canvas, the action where it was defined is also highlighted.
  This provides a quick display of the relation between the story and its implementation within the application.

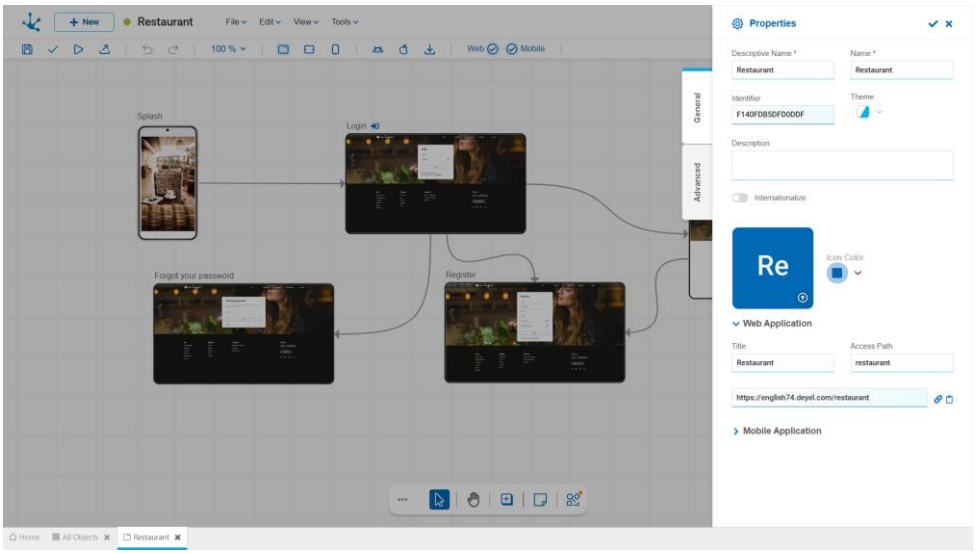
3.4.9.1.3.4. Tools Submenu

This submenu is opened by clicking on the "Tools" option and allows for modeling the application's properties and permissions, as well as performing operations to make it available to users on different devices.



Properties

Opens the panel of properties of the application.



Permissions

Allows to assign the security functions for use and modeling of the object to the existing permissions or to new permissions that the user creates, without having to go to the permission settings option from the menu.

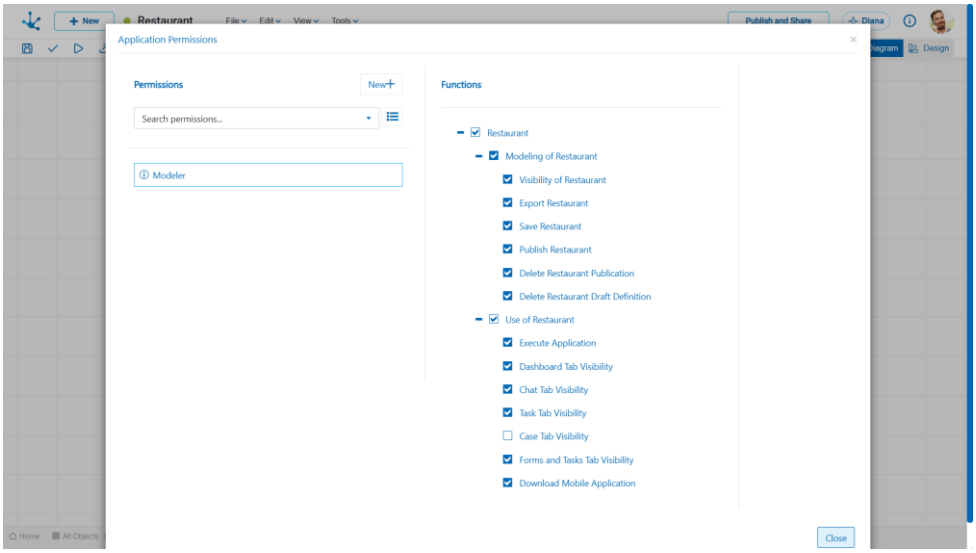
Sections

- Permissions: Permissions to which object functions are assigned.

- Security Functions: Represents the total set of security functions, modeling and use of the object. Those that are marked are the ones included in the selected permission.

By default, all security functions for a new object are assigned to the permission Modeler.

Users who are assigned the permissions have access to the functions included in it.



New Opens a panel to create a new permission and once created, the security functions included in it must be selected .

Search permissions... Allows to select a permission from a list and enables the input of characters to filter the values in the list.

Opens the wizard to select a permission and once chosen, the necessary object security functions must be checked.

To unrelate a permission from the object, hover the cursor over the permission and press the icon ✖. If there are functions selected for that permission, they must be unchecked in order to delete it.

Modeling Security Functions

- Visibility: Allows to display the object in the Deyel modeler grid.
- Export: Enables the operation to export the object from the tools submenu of the expanded menu.
- Save: Enables the operation of saving modifications made to the object.
- Publish: Enables the operation of publishing the object leaving its state as "Published".
- Delete publication: Enables the operation of deleting the object publication leaving its state as "Draft".

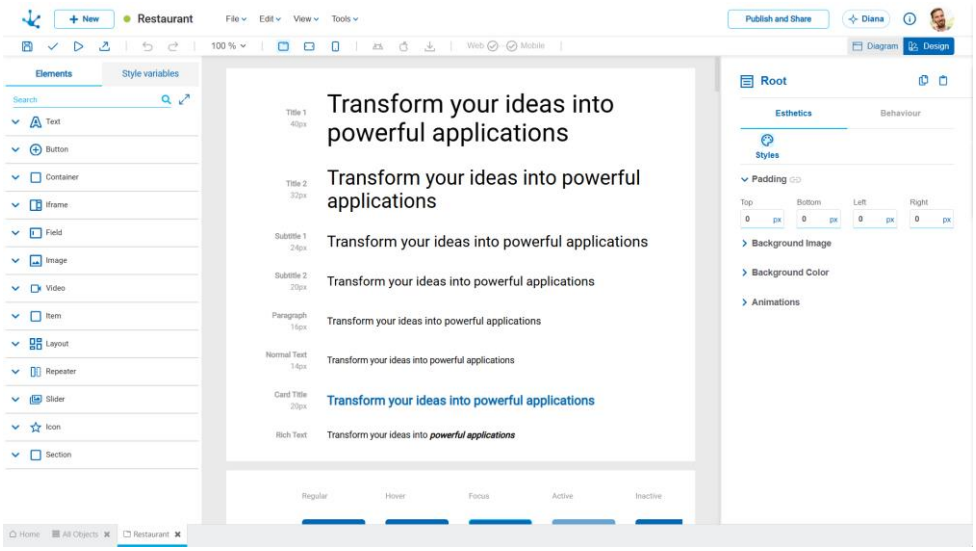
- Delete draft definition: Enables the operation of deleting the object.

Use Security Functions

- Execute Application.
- Dashboard Tab Visibility: Allows to display the dashboard tab on the mobile device.
- Chat Tab Visibility: Allows to display the chat tab on the mobile device.
- Task Tab Visibility: Allows to display the task tab on the mobile device.
- Forms and Tasks Tab Visibility: Allows to display the forms and tasks tab on the mobile device.
- Download mobile application: Enables to download the mobile application from the user profile of the portal.

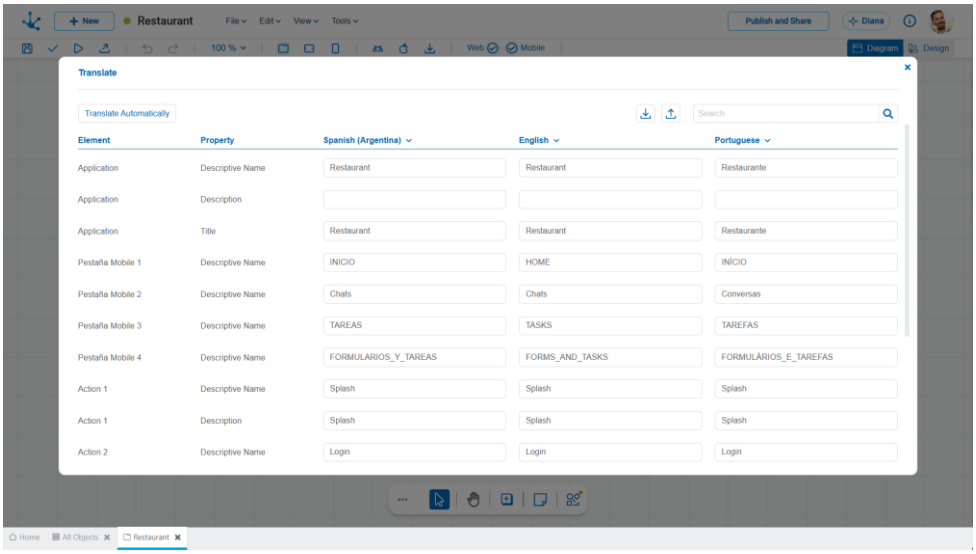
Design System

Opens the application's design system panels.



Internationalization

The "Internationalization" option is enabled when the property Internationalize is activated. To access the translation panel, it is necessary to select "Internationalization" and then choose "Translate".



In this panel, all the translatable properties of the object are displayed, along with their content in the original language, and translations in the other available languages can be entered manually.

If a value is modified in the original language column, the change is applied directly to the object's property.

In the header of each language column, a menu that allows selecting another language can be displayed, updating the column with its contents.

Options Available in the Top Margin

Translate Automatically Generates translations from the original language to other languages using artificial intelligence.

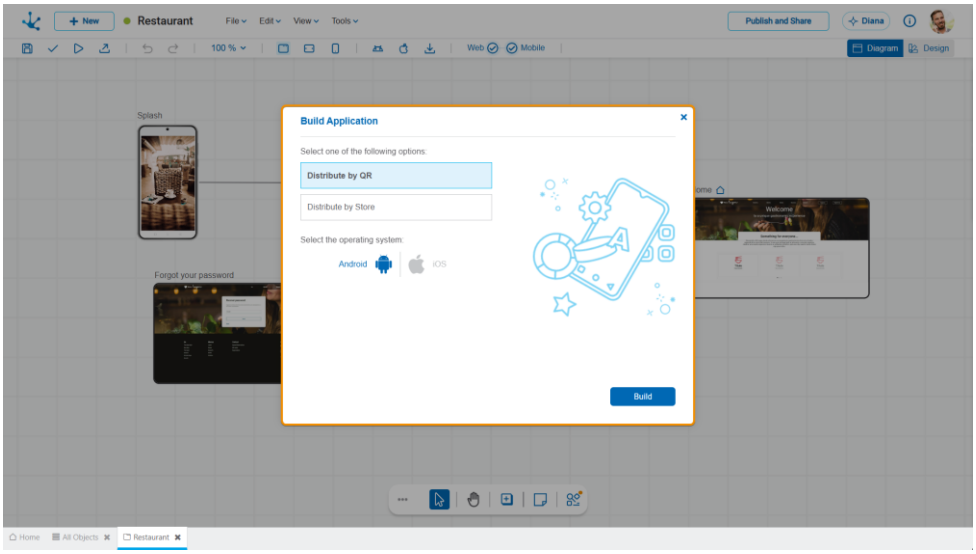
Downloads an Excel file with the contents of the translation panel.

Allows selecting an Excel file to import translations, making it easier to manage large volumes of content.

Filters the contents of columns based on the characters entered in the search field.

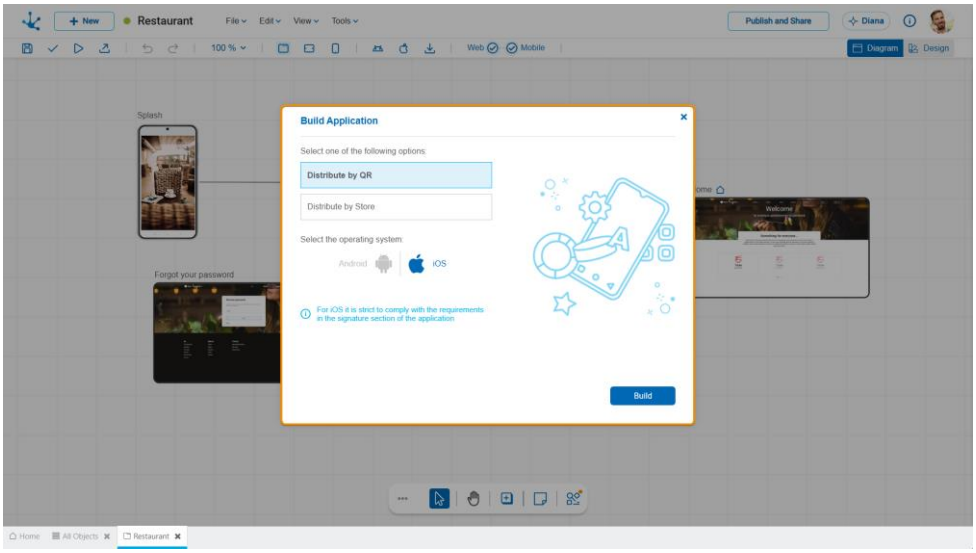
Build for Android

Allows to generate the application to be installed on a mobile device, for which a specific sequence of tasks must be performed for its distribution on the Android operating system. When generating the application, a panel opens that allows the entry of the distribution type and the operating system. The mobile application must be published to be built.



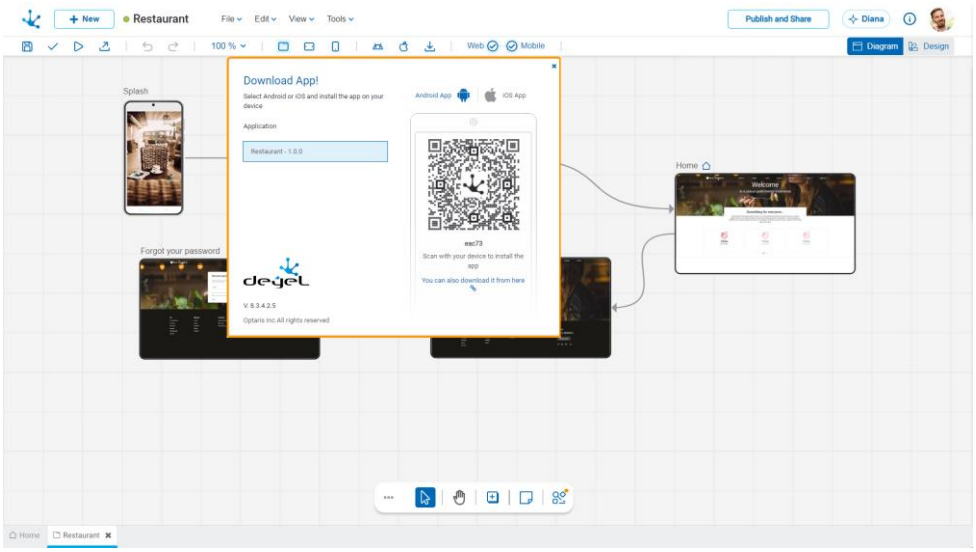
Build for iOS

Allows to generate the application to be installed on a mobile device, for which a specific sequence of tasks must be performed for its distribution on the iOS operating system. When generating the application, a panel opens that allows the entry of the distribution type and the operating system. The mobile application must be published to be built.



Download App

Once the mobile application is built, it can be downloaded to the mobile device by scanning the corresponding code, using a camera or a QR code reader. This menu option can only be used via QR code for distribution on devices with Android or iOS operating systems.



Application

It allows selecting the application to download from a list of available applications.

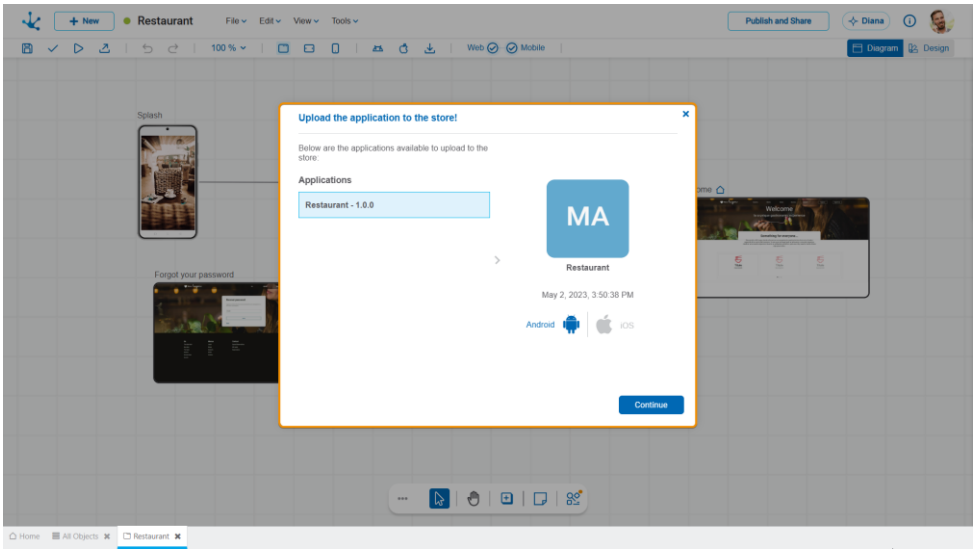
Android App It allows generating the QR code to install the application on a device with the Android operating system.

iOS App It allows generating the QR code to install the application on a device with the iOS operating system.

You can also download it from here The application can also be downloaded by clicking on this text.

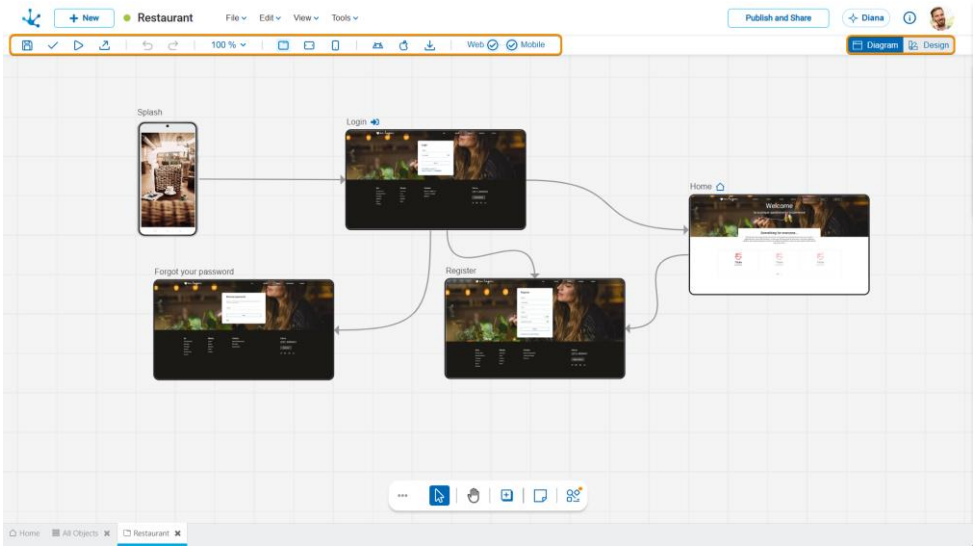
Upload to the Store

It allows publishing the application in the official store of the selected operating system, after performing the sequence of specific tasks for its distribution via store.



3.4.9.1.4. Top Toolbar

Contains icons for quick access to the most used operations of the expanded menu, as well as the buttons that allow access to the graphic modeling area or to the design system.



File

- Save
- Validate
- Publish
- Export

Edit

- Undo
- Redo

View

100 % Zoom Percentage

Breakpoints

- Desktop Computer
- Tablet
- Mobile

Web Web

Móvil Mobile

Tools

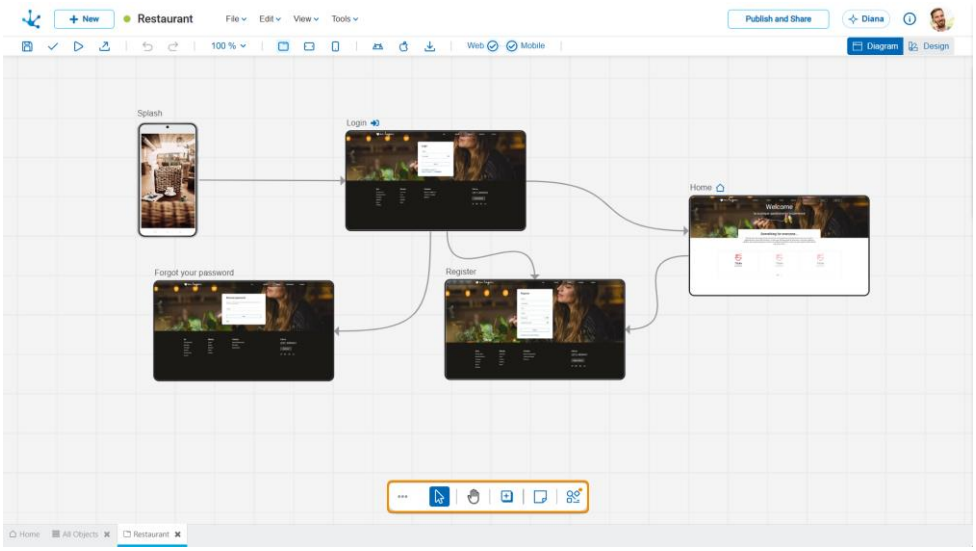
- Build for Android
- Build for iOS
- Download App

Diagram Allows visual modeling of the application in the graphic modeling area, where the navigation flows and the relations between the different pages are defined.

Design Allows centralized access to the design system of the application, such as elements, fonts, colors, buttons, texts and element subtypes, among others. From here, the modeling user can define, adjust, and maintain visual consistency throughout the application, ensuring a uniform and cohesive user experience.

3.4.9.1.5. Floating Palette

This bar that contains the different elements that can be modeled in the application, including the navigation and selection modes for the modeling area.



It allows selecting different modeling elements:

- Action
- Note
- Pages modeled outside the application modeler

Navigation and Selection Modes

Navigation and selection modes can be activated from the side toolbar or via keyboard shortcuts. When one of them is activated, the other one is automatically deactivated.

 The navigation mode allows navigating the modeling area without using scroll bars, making it easier to explore the application. To move, simply click and drag the cursor.

 Selection mode is the default and allows selecting and editing elements in the modeling area. While active, it can be temporarily switched to navigation mode by holding down the space bar or by right-clicking and dragging the cursor.

 It allows the palette to be moved within the modeling area by clicking on the icon and dragging it.

Keyboard shortcuts:

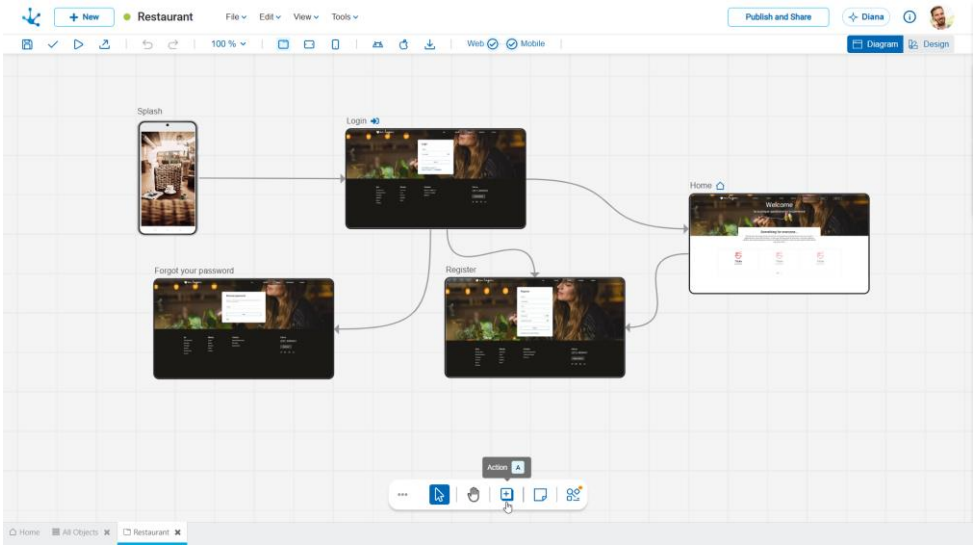
- H Activate navigation mode.
- V Return to selection mode.

3.4.9.1.5.1. Action

It allows to create an action in the graphic modeling area, by dragging the icon to the desired position. If the icon is clicked, the action is created in a certain position and should be dragged to the chosen location.

An action represents an operation within the functional flow of the application. It is generally associated with a user story and defines what should happen when the business user interacts with the interface, for example by clicking a button, selecting an option, or moving forward in a process.

Each action describes a specific task performed in the application, such as displaying a page, updating an entity's data, or navigating to another point in the flow. The actions are visually connected within the modeler's canvas, allowing the complete flow of the application to be built graphically.



Operations

Add Actions

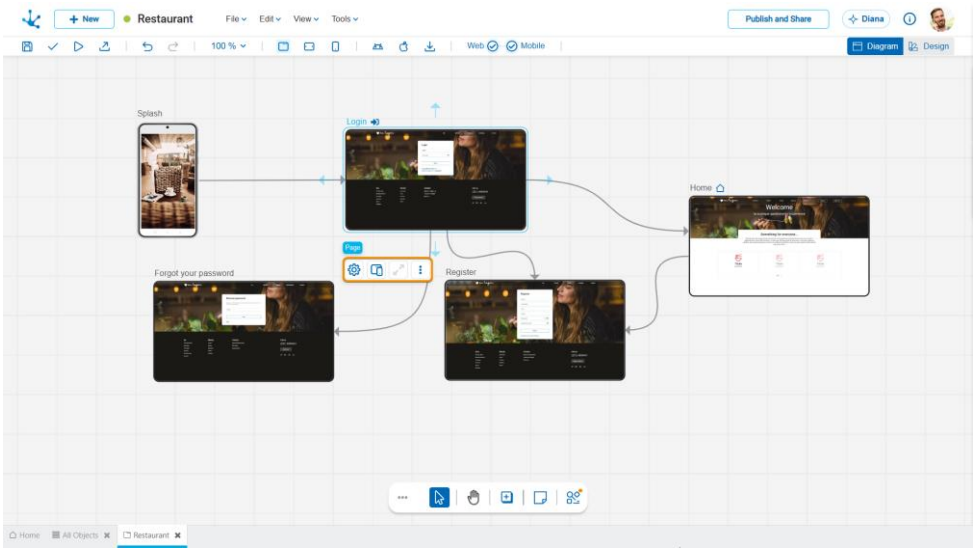
Actions can be added to the graphic modeling area in any location via the “drag and drop” facility. Once the element is configured, double-clicking on it opens a new tab corresponding to the pages modeler.

Modify Position of an Action

Once the action is added, its position can be changed by dragging it within the graphic modeling area. If the position of an element is changed, the change is applied to all breakpoints.

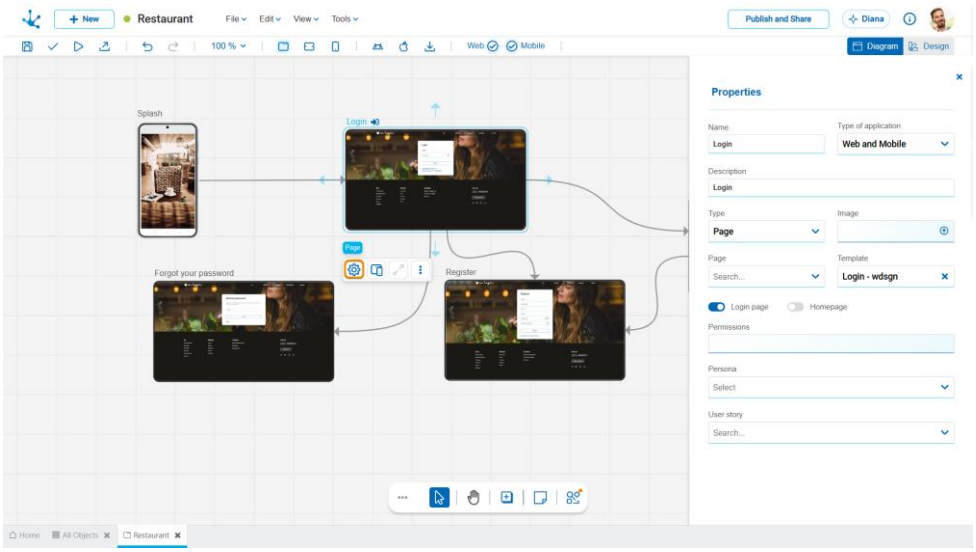
Context Menu

Selecting an element opens a palette of icons which represent the configuration of its properties, the type of web or mobile page, the option to modify the size of an element and a menu, whose options correspond to operations that can be performed on the selected action.



Configuration Properties

The configuration properties panel opens when selecting the icon  .



Name
It is used at the modeling level to reference the action.

Type of Application
It allows to define if the application is web-type, mobile-type or both.

Description
Text that defines the action describing its functionality.

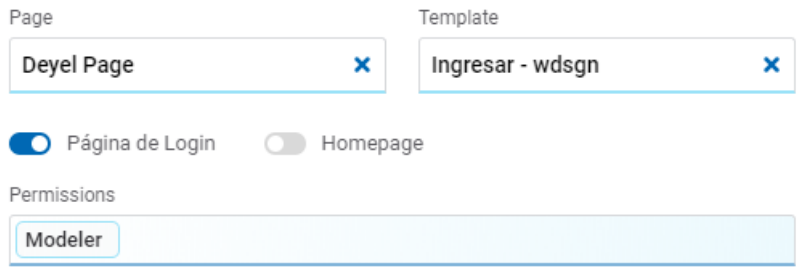
Type
It allows to select the content displayed on the action and depending on the option selected, different properties are enabled.

Image
It allows to define an image that represents the action in the applications modeler, it is valid if the properties for the selected item type are not completed .

Properties according to the Selected Type

Different properties are enabled depending on the selected type of object.

Page



Page
The pages modeled in the environment are displayed.

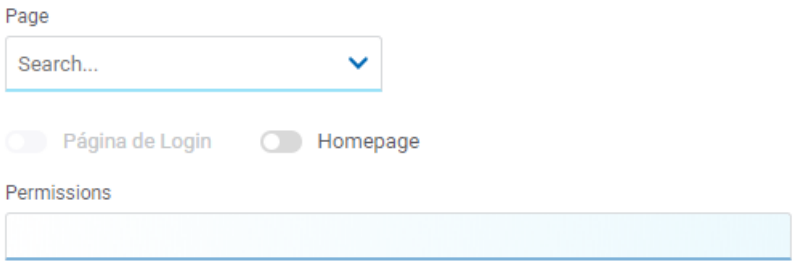
Template
It allows selecting a template to model the page if the previous property is not completed.

Login Page
Indicates that the unlinked page corresponds to a login page.

Home Page
Indicates that the unlinked page corresponds to a home page.

Permissions
Indicates all the permissions of the selected page. They are not modifiable.

Deyel Page

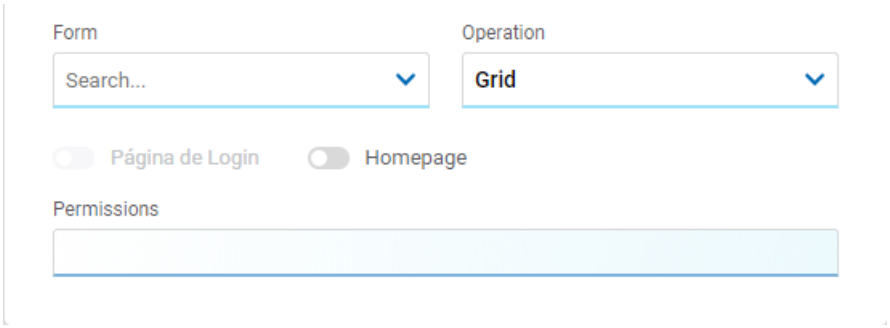


Page
The pages modeled in the environment are displayed.

Home Page
Indicates that the unlinked page corresponds to a home page.

Permissions
Indicates all the permissions of the selected page. They are not modifiable.

Form



Form
The forms modeled in the environment are displayed.

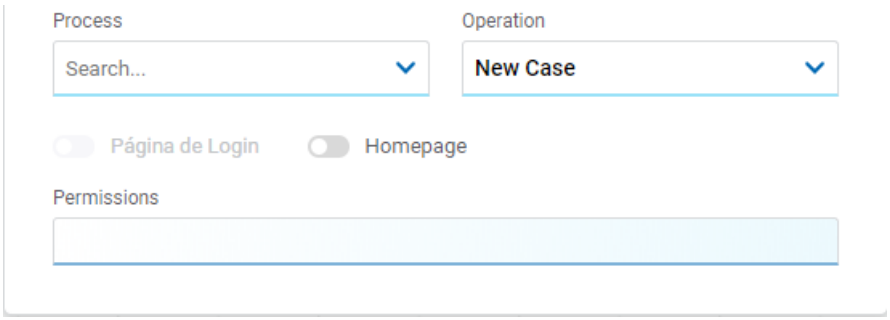
Operation
The operation that is performed when selecting the form is defined.
Possible Values

- New: Indicates that the panel of the object selected in the previous property is opened for the creation of an instance.
- Grid: Indicates that the grid of the object selected in the previous property is opened.

Home Page
Indicates that the unlinked page corresponds to a home page.

Permissions
Indicates all the permissions of the selected page. They are not modifiable.

Process



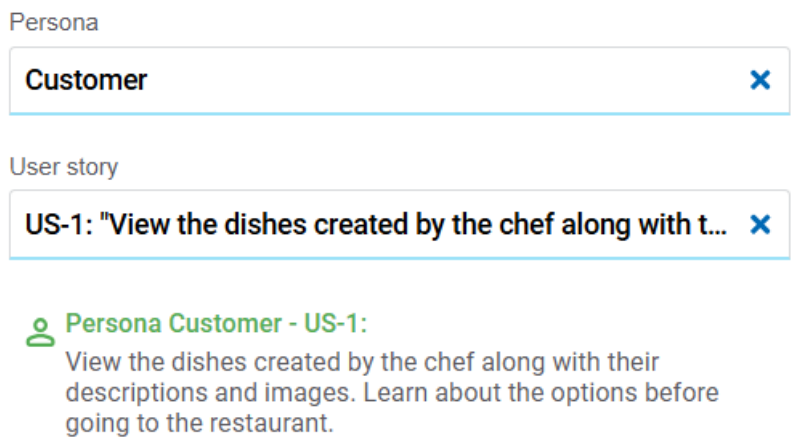
Process
The processes modeled in the environment are displayed.

Operation
The operation that is performed when selecting the process is defined.
Possible Value

- New Case

Home Page
Indicates that the unlinked page corresponds to a home page.

Permissions
Indicates all the permissions of the selected page. They are not modifiable.

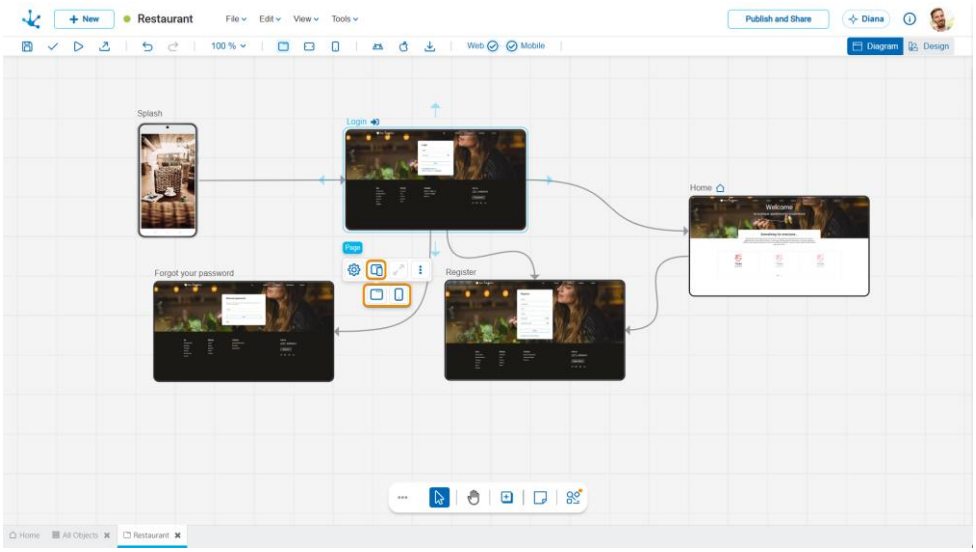


Persona
Indicates the persona that interacts with the action.

User Story
The user stories of the selected persona are displayed.

Type of Page

The type of page is selected by means of the icon .



Possible Values

- Web: This type of application can be accessed from the web.

- Mobile: This type of application can be accessed from mobile devices.

Expand or Collapse the Page

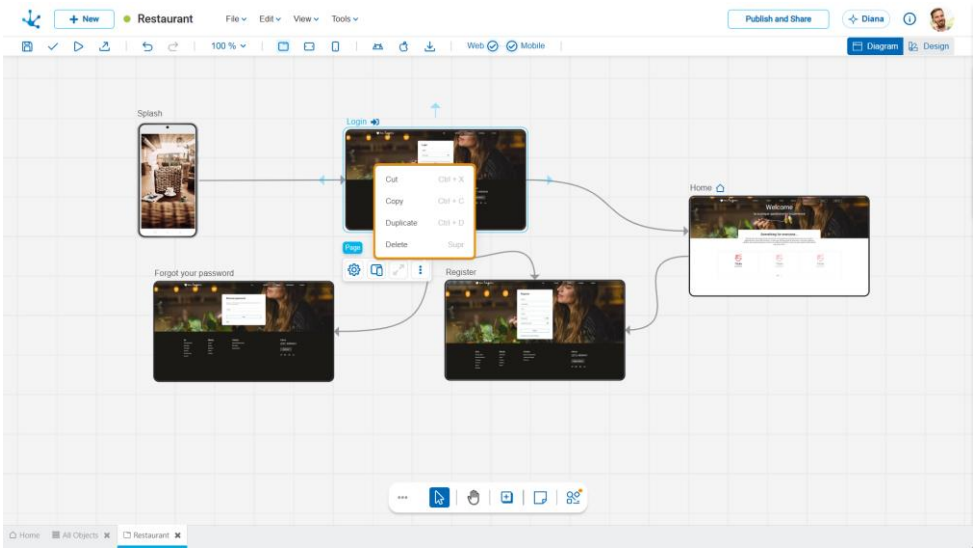
Expands the size of the page, displaying all its content without height restrictions. Can be used for long pages or pages with multiple sections.

Reduces the size of the page, displaying the content in a compact version. It helps keep the modeling area organized and makes navigation between pages easier.

More options

The menu to select additional options is expanded with the icon .

> *The options menu can also be opened by right-clicking mouse on the selected item.*



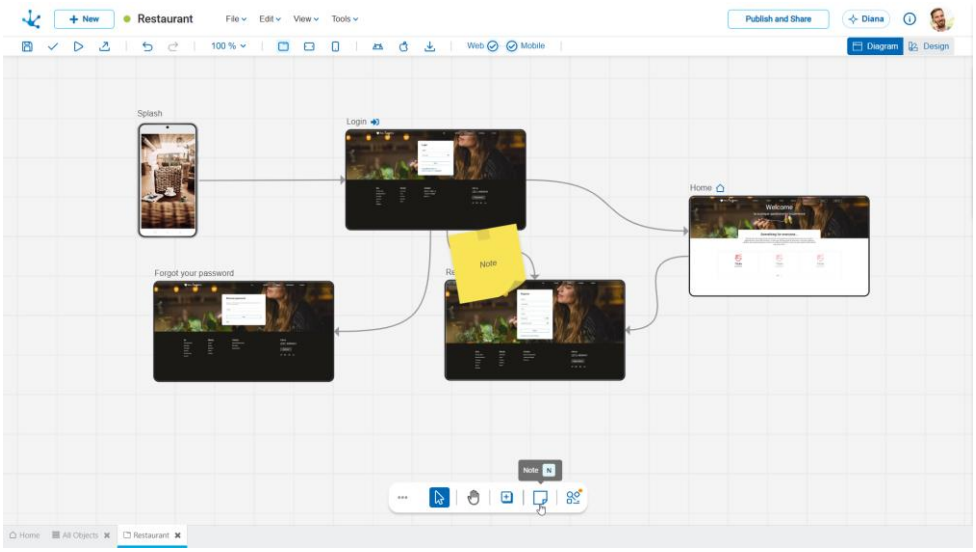
The options correspond to operations that can be performed on the selected element.

- Cut (Ctrl+X)
- Copy (Ctrl+C)
- Duplicate (Ctrl+D)
- Delete (Del)

3.4.9.1.5.2. Note

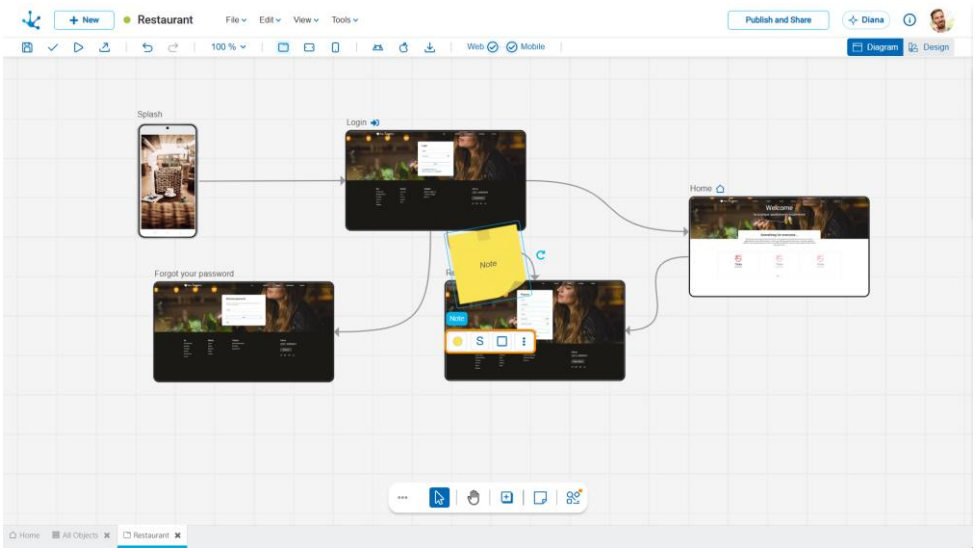
It allows to create an element in the graphic modeling area to include brief comments about the modeling. This can be done by dragging the icon to the desired position or by clicking on it.

Notes do not affect the execution of the application.



Context Menu

Selecting a note opens a palette of icons in order to choose its color, size, and type of note. In addition, a menu can be opened with options corresponding to operations on the selected element.



Color

It allows to model a background color for the note, selecting it from a palette.



Size

It allows to define the dimension of the graphic element.

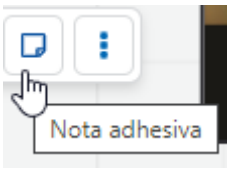


Possible Values

- S (small)
- M (medium)
- L (large)

Type of Note

It allows to define the appearance of the graphic element.



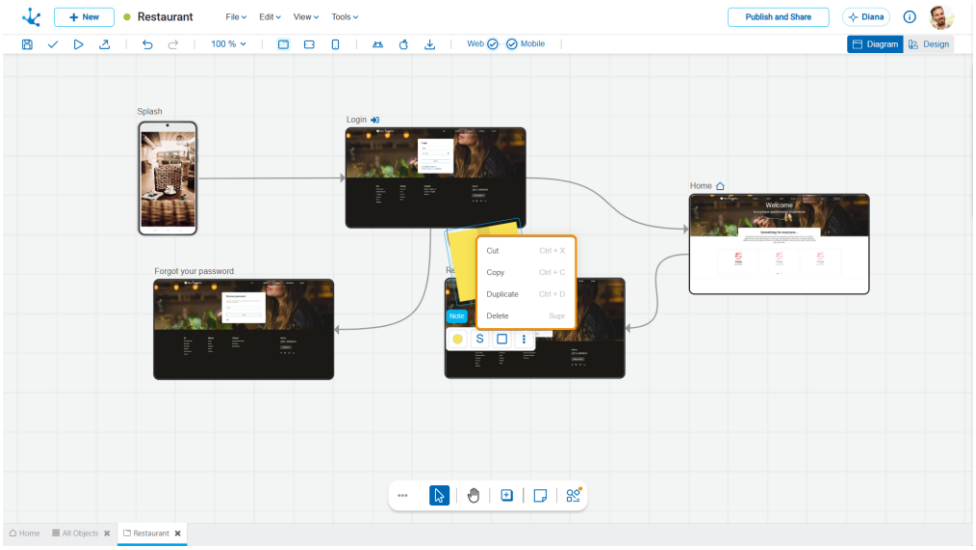
Possible Values

- Note
- Post-it Note

More options

The menu to select additional options is expanded with the icon  .

> *The options menu can also be opened by right-clicking mouse on the selected item.*

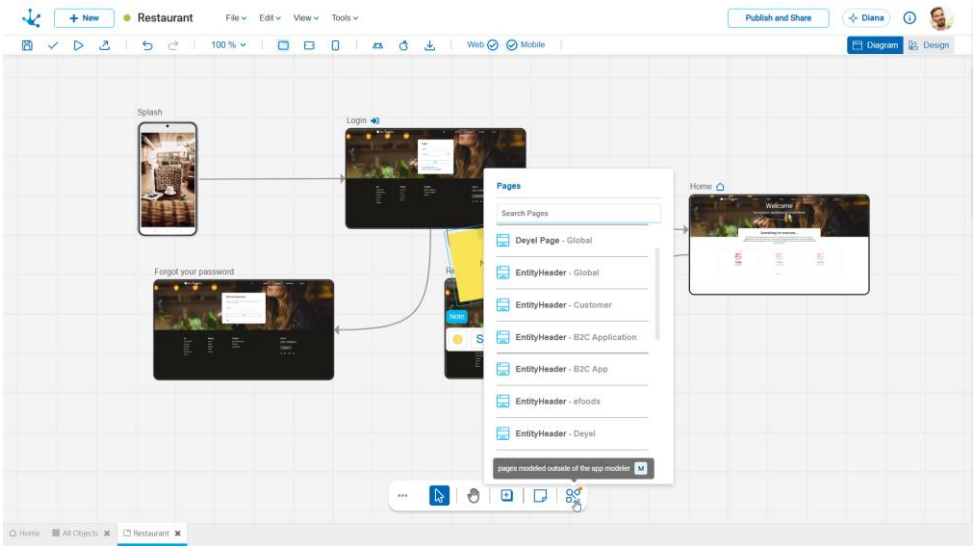


The options correspond to operations that can be performed on the selected element.

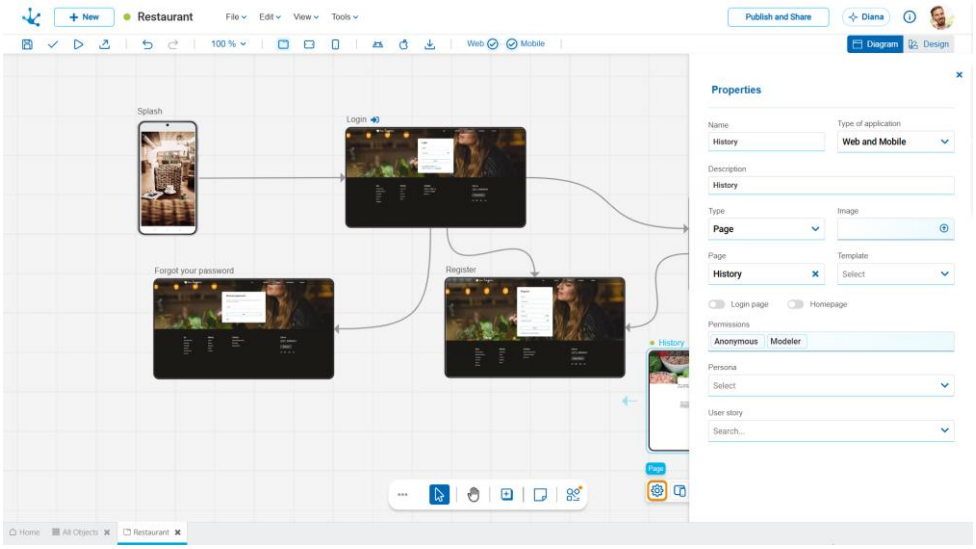
- Cut (Ctrl+X)
- Copy (Ctrl+C)
- Duplicate (Ctrl+D)
- Delete (Del)

3.4.9.1.5.3. Modeled Page

It allows incorporating a previously modeled page that does not belong to the application into the graphic modeling area. Clicking the icon displays the list of available pages. If a page is selected from the list, it is automatically inserted into a default location and can then be moved manually to the desired position. Alternatively, the page can be dragged directly from the list to the preferred position.

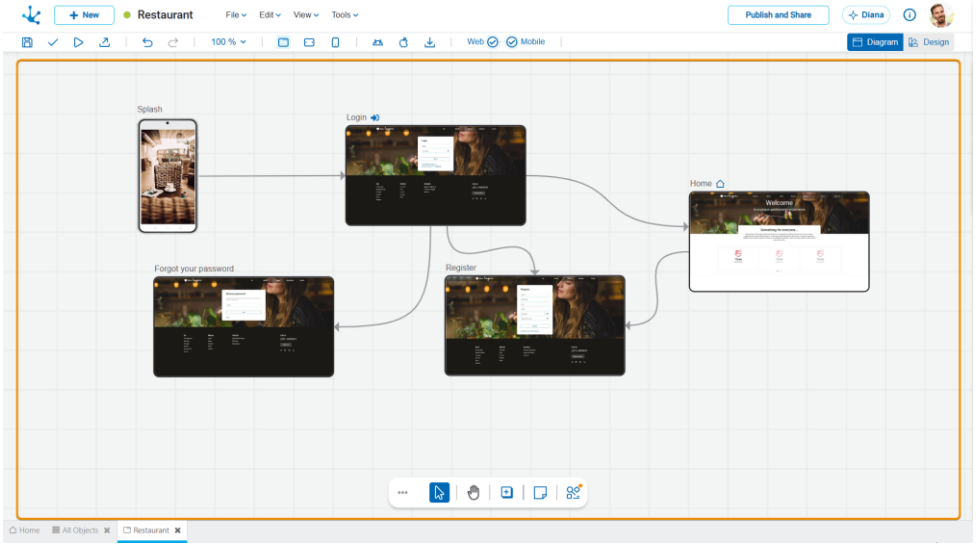


Once the page is located in the graphic modeling area, it uses the same context menu as the pages that were created and linked to the application. Its properties are automatically filled with the values from the selected page.



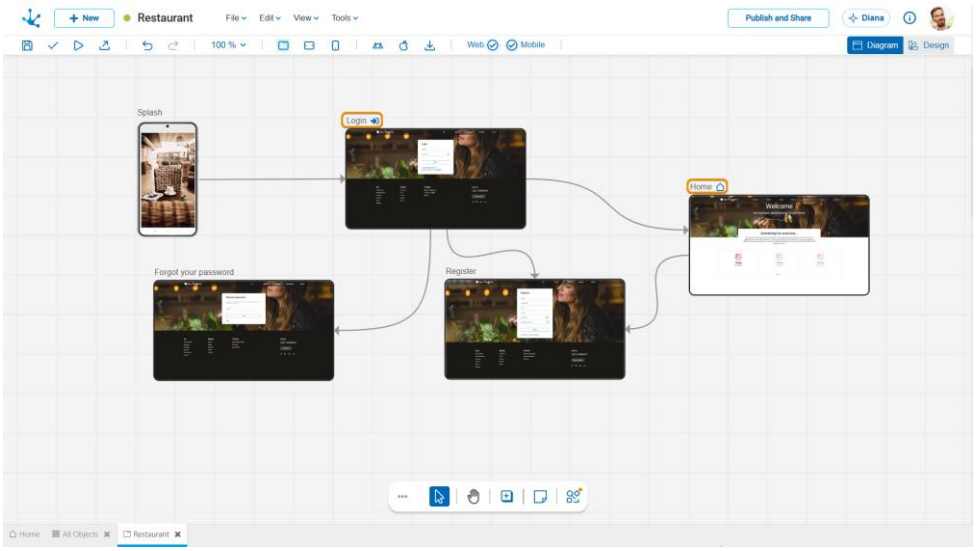
3.4.9.1.6. Graphic Modeling Area

The graphic modeling area is the location of the different elements of the side toolbar, allowing navigation to be modeled through navigation flows between such elements. It may initially appear blank if none of the built-in templates were selected.

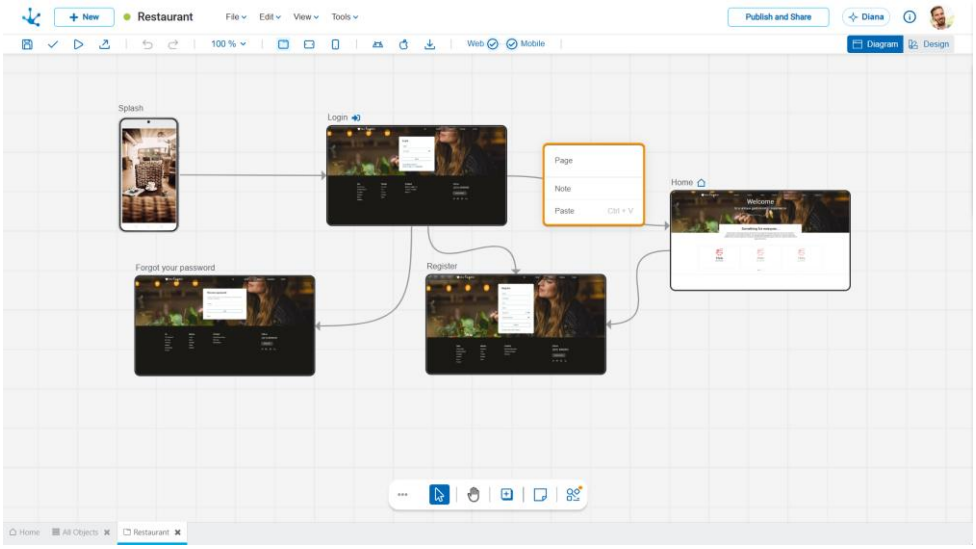


Its state, its name and the corresponding icon are displayed at the top of each action, if the Login Page or Home Page properties were modeled.

Home
Login

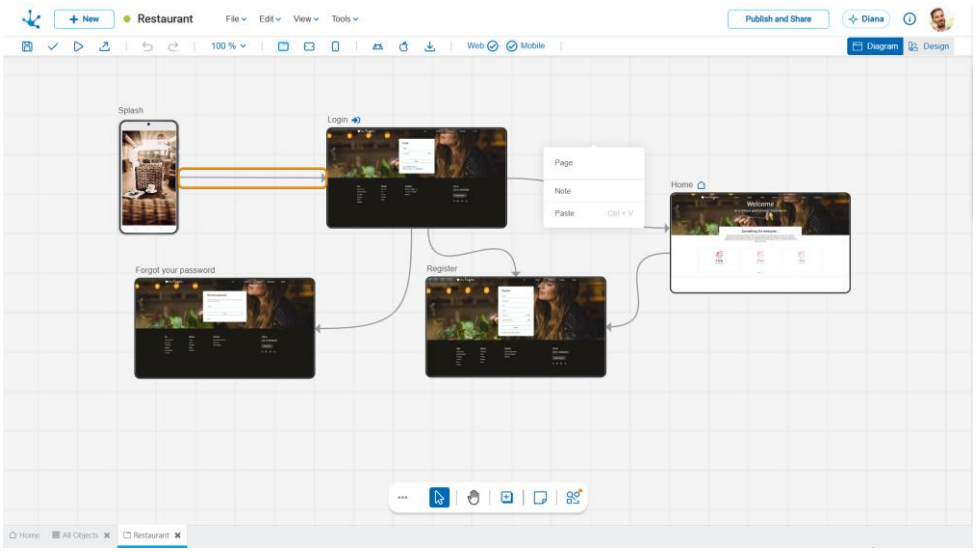


By right-clicking on the modeling area, a menu with the operations corresponding to the elements is displayed.



Navigation Flow

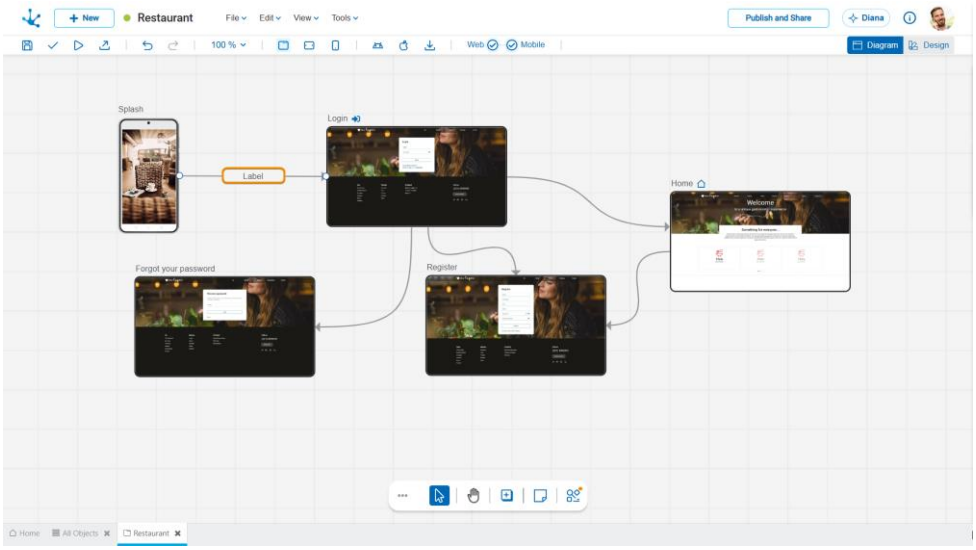
Represents navigation between actions.



Modeling Flow
It is the default flow type, which is used in the modeling stage of the application. Indicates that it has been modeled but not yet implemented.

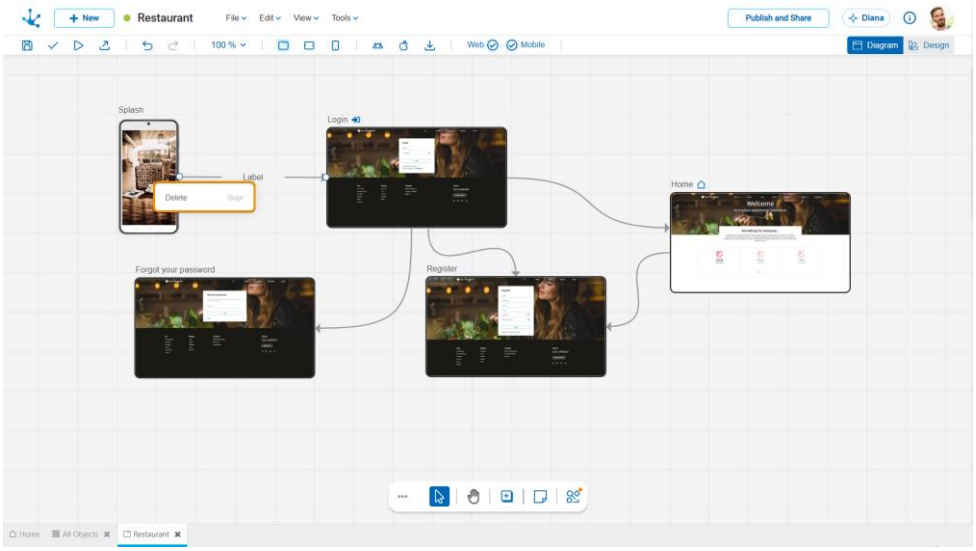
Implemented Flow
Indicates that there is an implemented navigation between two actions. That is, when a page that has a hyperlink to another one is published, if the flow is not modeled it is automatically generated in a light blue color. On the other hand, if it already exists, it changes its color from gray to light blue.

By double clicking on a flow, a text can be entered as a label. By clicking on a flow, a point of curvature is selected and created, which when dragged allows the flow to be given a new shape.



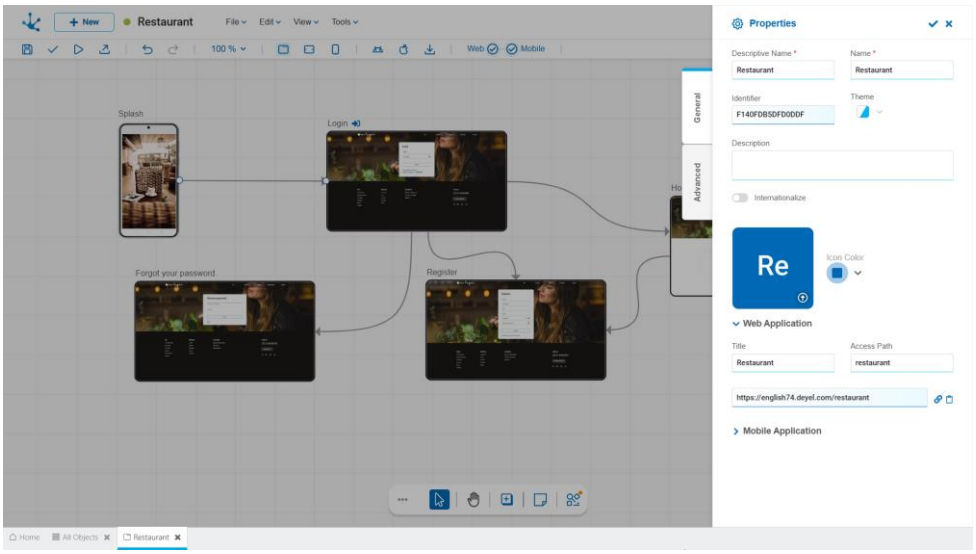
Delete Flow

By right-clicking on the flow, a menu with the operation "Delete" is displayed.



3.4.9.2. General Properties

The properties panel is displayed on the right side of the applications modeler, where the first tab corresponds to general information.



Descriptive Name

Name used by the user to reference the application.

Name

It is used at the modeling level to reference the application.

Identifier

Uniquely identifies the application.

Theme
It allows to select the color scheme in the application through the selected theme.

Description
Text that defines the application describing its functionality.

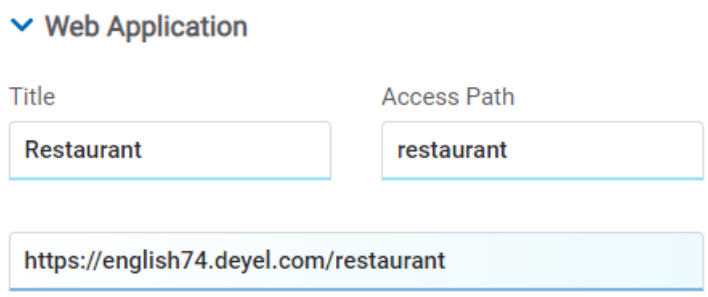
Internationalize
Enables the translation of object properties and its elements into different languages. When activated, it enables the option "Internationalization" from the "Tools" drop-down menu and the property Language of Existing Values, which is displayed on the right.

Language of Existing Values
Displayed when the property Internationalize is activated for the first time and indicates the original language. Its selection is required and the available options are: "Spanish (Argentina)", "Spanish (Mexico)", "English" and "Portuguese".

Image
An image is automatically associated to the application with the initials of its name and optionally a file can be uploaded. This image is displayed in the browser tab when executing the application.

Icon Color
It allows to select a color for the image within the color range of the theme. If a file was uploaded as an image, this property is invalid.

Web Application



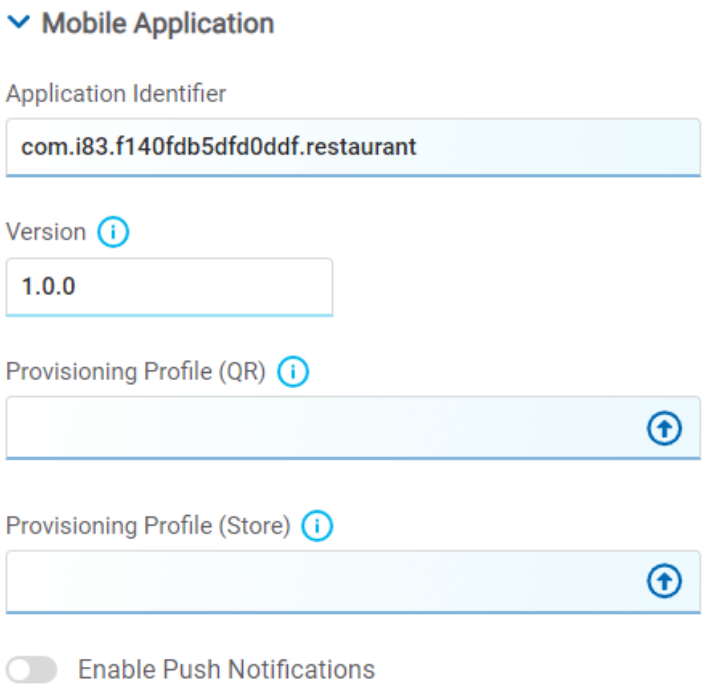
Title
Text displayed in the browser tab when executing the application. It allows to identify which application is being used.

Access Path
It allows to enter the name to access the application in order to execute it from a browser. The full login path is displayed below this property, including the name of the environment, concatenated by a ¨/¨. This property must be defined so that the application can be executed in web mode.
If the application has to run in the root path of the environment, the property must be completed only with a slash "/".

> *Only one application per environment can be executed in its root path.*

For example, if this property is filled in with a “purchases” value, it can be accessed with the URL “myenvironment.deyel.com/purchases”. Meanwhile, if it is filled in with “/”, it can be accessed with the URL “myenvironment.deyel.com/” or “myenvironment.deyel.com”.

Mobile Application



Application Identifier
Identifier of the mobile application used to distribute it to different devices.

Version
Version number of the mobile application that keeps the semantic versioning format <main version>.<increased version>.<correction>.

Provisioning Profile (QR)
Configuration file that guarantees the security and authenticity of the mobile application and can distribute it via QR to devices with iOS operating system.

Provisioning Profile (Store)
Configuration file that guarantees the security and authenticity of the mobile application and can distribute it via application store to devices with iOS operating system.

Enable Push Notifications
If an application has been modeled to use the "Push" notifications functionality, this check must be enabled.
Selecting this option enables additional properties.

Google Services (Android)

File that allows interacting with Google services that are used in the mobile application with Android operating system. This file is generated by following the steps for the Android application distribution.

Google Services (iOS)
File that allows interacting with Google services that are used in the mobile application with iOS operating system. This file is generated by following the steps for the iOS application distribution.

3.4.9.3. Design System

The design system (DS) of **Deyel** is a set of reusable standards and predefined components that guide the build of user interfaces. It establishes a solid foundation for application development, ensuring visual and functional consistency, which is essential for usability and user experience.

The implementation of the design system of **Deyel** pursues the following main objectives:

- Ensure visual and functional consistency, ensuring a consistent and predictable user experience across all applications.
- Centralize the modeling of page element and entity properties, enabling their centralized adaptation and modification, thus facilitating both the development and the maintenance of applications.
- Optimize the modeling of pages and entities, ensuring that each application has the necessary elements and subtypes from the start, minimizing specific configuration during development.

The use of the design system (DS) of **Deyel** guarantees key benefits:

- Reduces the learning curve for users.
- Improves the perceived quality of the applications.
- Accelerates development by providing ready-to-use components and clear guidelines.

Components

The design system (DS) of **Deyel** includes components which, when configured and used centrally, guarantee the uniformity and visual quality of the applications.

- **Elements**
  Defines the properties and behaviors of the various user interface elements within the design system, such as text, buttons, and fields, among others.
- **Font family**
  Sets fonts, sizes, and styles to ensure readability and the visual structure of applications.
- **Color palette**
  Defines the selected colors and their correct use within applications.

Implementation

The implementation of the design system of **Deyel** is organized through a structured workflow that guides the process from building to maintenance of the applications.

When it comes to existing applications, the design system can be incorporated in a step-by-step process, allowing the validation of proper functionality at each stage and adaptation of fonts, colors, and subtypes without compromising stability.

Finally, to ensure the effectiveness of the design system and its long-term evolution, it is essential to apply a set of best practices and recommendations, which promote order, style centralization, responsible management, and respect for the underlying design logic.

Workspace Sections

The design system workspace is enabled by selecting the "Design" button on the top toolbar of the applications modeler, and is divided into panels.

- **Left panel**

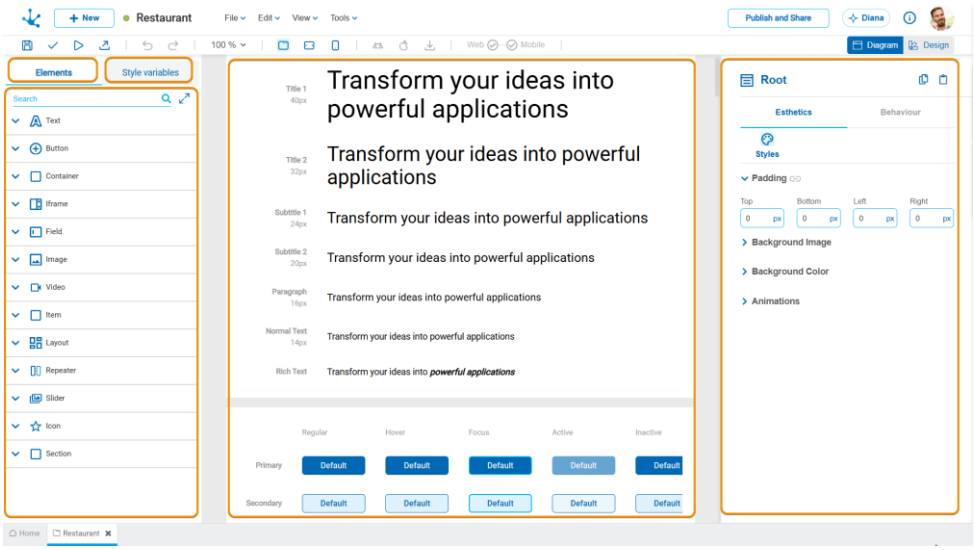
  Displays the different available elements and the style variables associated, depending on the tab selected.

- **Center panel**

  Allows displaying the different types and subtypes of elements, depending on where the mouse pointer is positioned in the left panel.

- **Right panel**

  Allows configuring the properties that define each type of selected element.

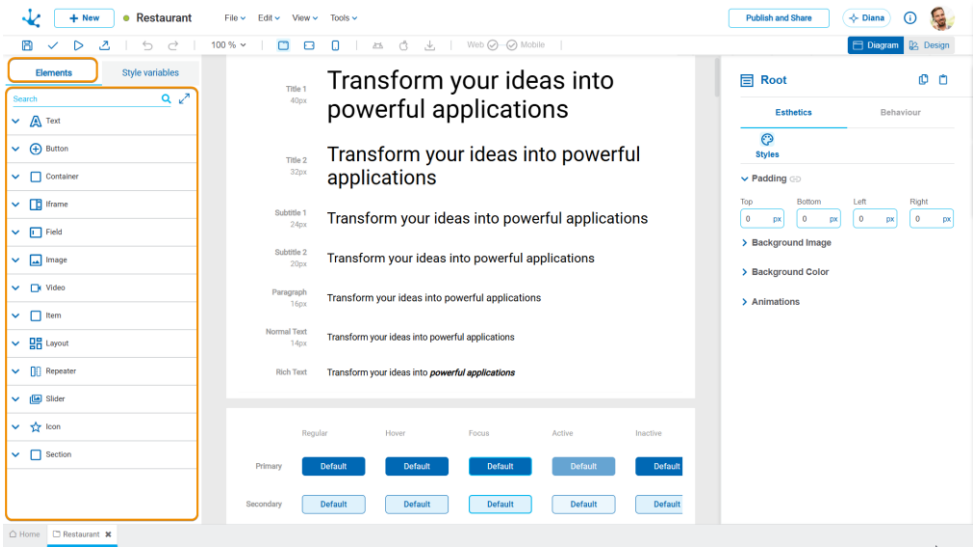


3.4.9.3.1. Components

The design system of **Deyel** includes components that, when configured and used correctly, ensure the uniformity and visual quality of applications.

Elements

Within the "Elements" tab of the left panel, subtypes are configured to standardize the behavior and appearance of user interface elements.



For the different types of elements in the design system of **Deyel**, their subtypes must be reviewed or configured, by defining their properties, such as color, size, spacing and hierarchy, in accordance with the application's visual identity.

Text

- Check the hierarchy of texts on the pages and configure the design system to match.
- Verify that the fonts match the defined typographic variables.
- If the default text subtypes are not sufficient, new ones can be built by assigning them a name and a clear description.

Button

- Define the width and the style of the buttons.
- Check the font used.

The procedure for configuring the subtypes of buttons is similar to that of texts.

Field

- Define working width. For example, 12% or 100%.
- Detect the most representative field to take it to the SD.

- It is essential to verify that the subtypes are not in a "modified" state on the pages; if so, the modified property should be reviewed and the subtype reassigned or the modification standardized in the SD if it is valid.

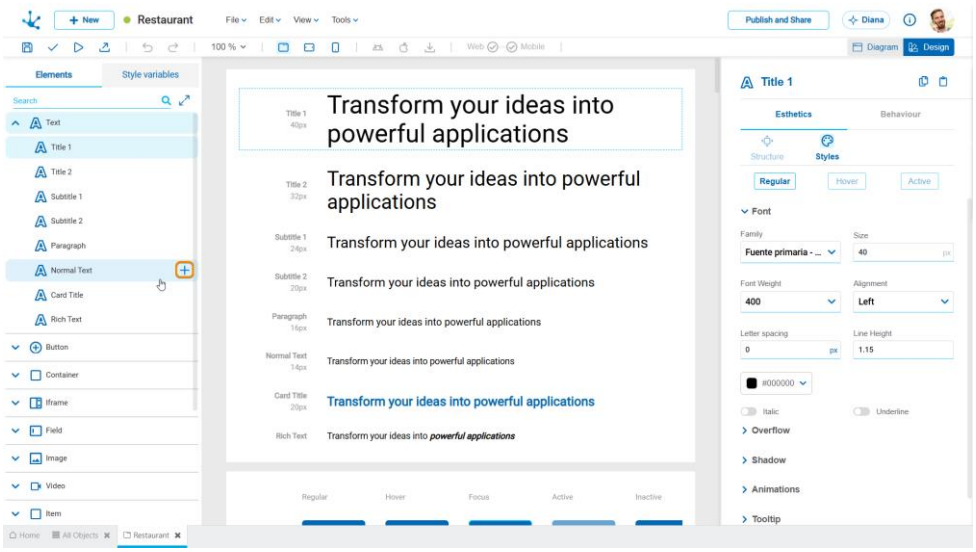
Other Types of Elements

For the remaining element types, the same principles of review, identification, and subtype configuration described for texts, buttons, and fields must be applied.

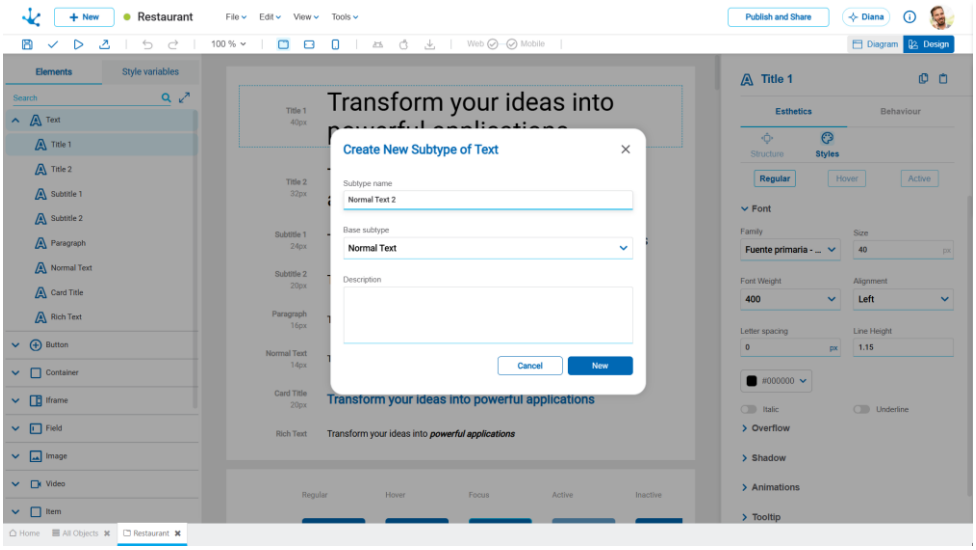
Once an element has been identified, its properties can be copied and applied to the specific subtype in the design system.

New Subtype

A new subtype of an item can be created by clicking on the icon + of the selected base subtype.



When doing so, a panel opens that allows defining the name and description of the new subtype.



Once created, the subtype must be configured in the properties panel and the application published.

Repeat these steps for the subtypes that are not available by default in the design system panel, paying special attention to the styles configured according to the element type.

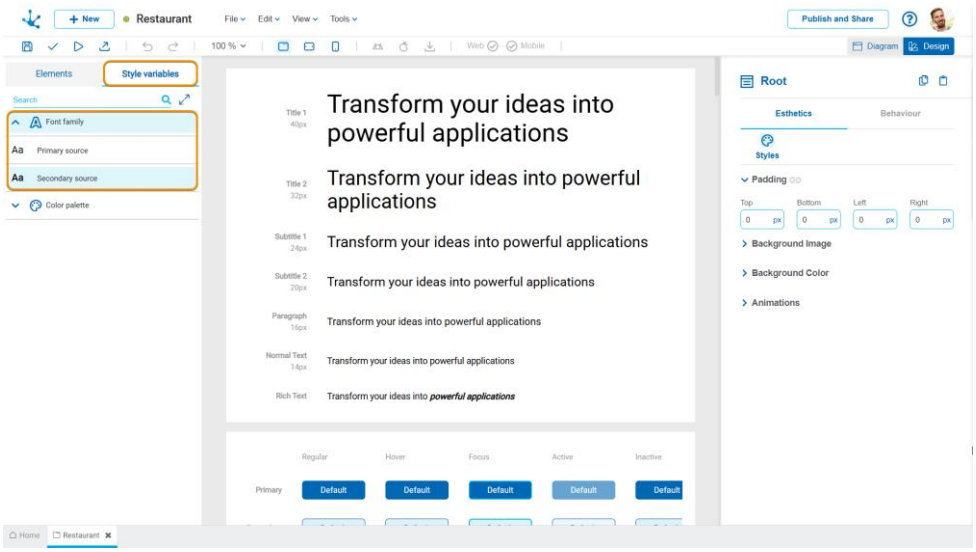
Font Family

A font family is a set of typefaces that share the same basic design but vary in aspects such as weight (light, regular, bold), style (italic, condensed, extended), or width. Furthermore, it structures the interface, since it not only fulfills an aesthetic role, but also hierarchically organizes the information and guides the user's navigation within the application.

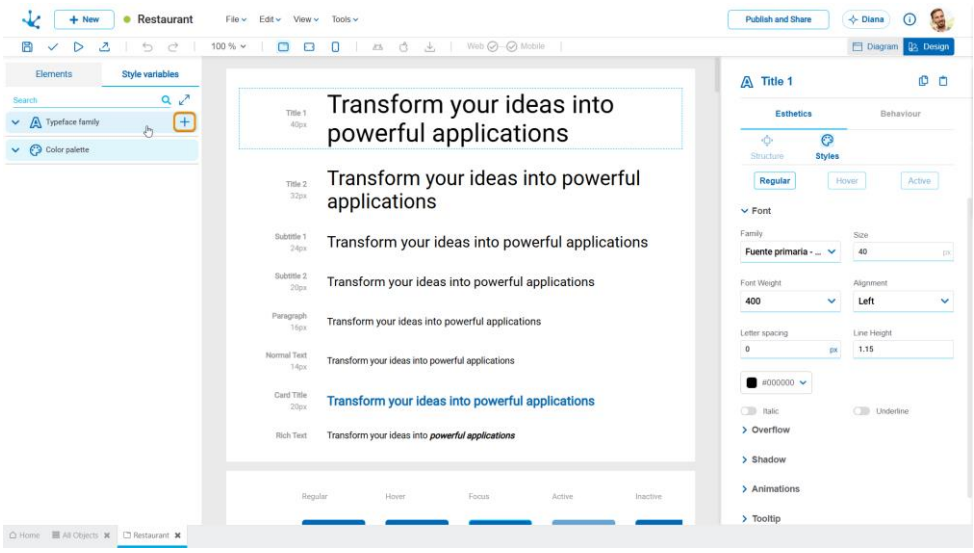
In the design system of **Deyel**, the following fonts are distinguished:

- **Primary font**: It is the main font that structures the interface, facilitates reading, and contributes to a clear and orderly appearance. It should be sufficient for most of the text in the application.
- **Secondary font**: Its use is exceptional and is reserved for situations where the hierarchy of the primary typographic variables is not sufficient to generate the desired contrast. Its definition is optional.

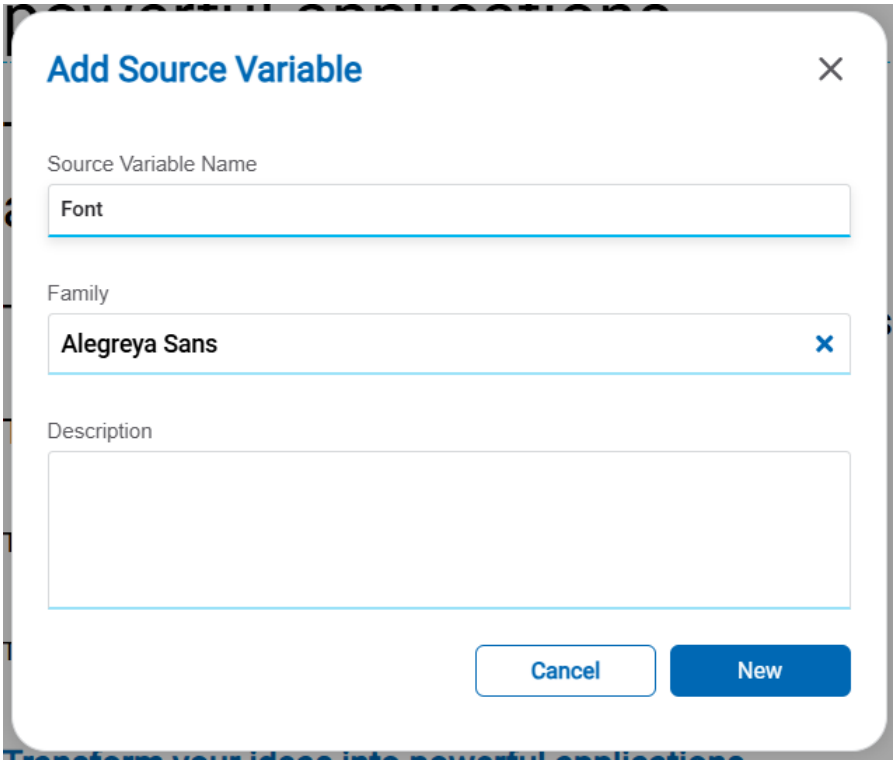
It is defined in the tab "Style variables" of the left panel.



New fonts can be created by clicking on the icon +.



When doing so, a panel opens that allows defining the name, family and description of the font.



Color Palette

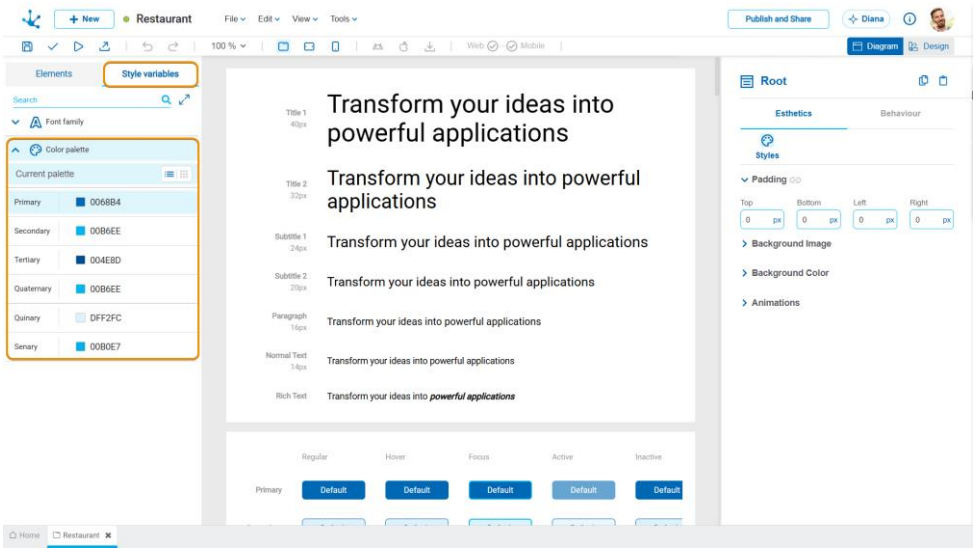
The color palette is essential for visual identity and information hierarchy. In the design system of **Deyel**, colors are organized around the [theme modeled in the application] and according to specific criteria.

The choice of palette should be based on brand identity, accessibility (contrast and legibility) and visual consistency throughout the entire application.

The palette configuration is structured around functional roles of color, rather than application modeling. These roles determine their use and visual weight within the application.

- **Primary criterion:** It represents the brand or main tone of the application. It is applied to key components that guide the user's attention. It occupies approximately 30% of the visual weight and is used for main interface elements.
- **Secondary criterion:** It complements the primary color and reinforces the visual hierarchy. It occupies 10% of the visual weight and is used to highlight elements such as links, buttons, borders or specific parts of components (highlights).
- **Tertiary criterion:** It defines the background and content areas, ensuring contrast and readability. It provides visual stability and forms the structural basis of the interface. It constitutes 60% of the visual weight and is the basis of the structure, that is, the background color of the interface.

It is defined in the tab "Style variables" of the left panel, next to the font family.



3.4.9.3.2. Implementation

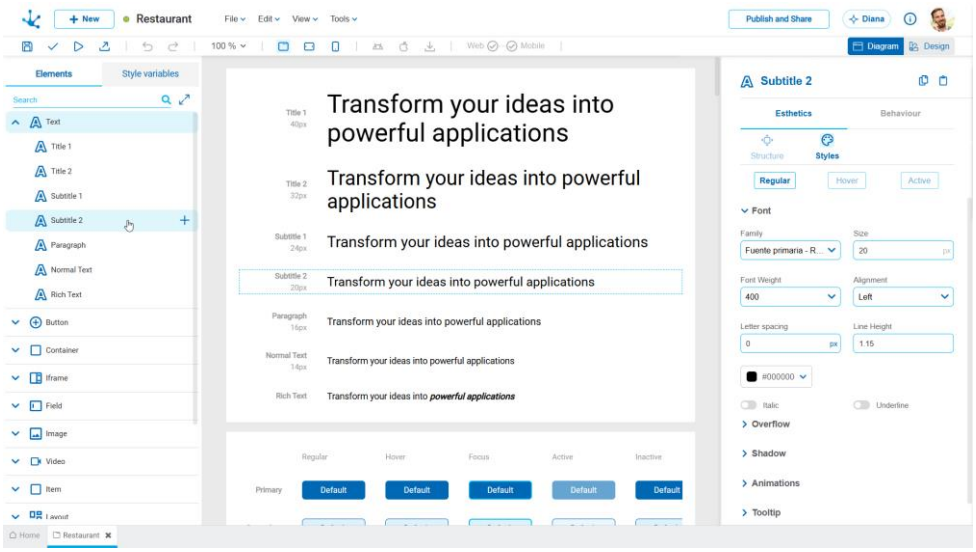
The implementation and maintenance of the design system in **Deyel** are managed through a structured workflow, which organizes tasks and responsibilities to ensure efficient use and maintain consistency across all applications.

It is important to designate a design system manager, as not all modeler users need to be assigned permission to modify it.

Application Build and Maintenance

First, all subtypes to be used in the application must be determined. This involves a thorough review of the design system to identify the necessary elements and their importance.

During application modeling, defined subtypes should be used, trying to minimize the modification of individual properties. This ensures that elements inherit the features from the design system, maintaining visual consistency.



Definition and Adaptation of Subtypes

If changes to existing subtypes or the creation of new ones are required, it is essential to coordinate with the team responsible for the user interface. The definition and adaptation of these elements should be a collaborative process to ensure they align with design and usability principles.

3.4.9.3.2.1. In Existing Applications

Implementing the design system into an existing application requires a careful approach to ensure a smooth transition and avoid disruptions in functionality.

Behavior in a Pre-Generated Application

An already generated application must continue to operate when implementing a design system. When opening pages and entities, the applied design system is displayed, although some properties may appear as modified. Pages and entities that are not published retain their original state.

Variable assignment, such as color assignment, is performed automatically whenever the color exists in the design system.

Steps for Implementation

The following steps should be followed when implementing the design system in a development environment to verify the correct functioning of the already modeled and operational applications.

1. Verify that the application functions correctly before implementing the design system.

2. Publish the most relevant pages and entities of the application, without publishing the entire application. At this stage, pages and entities do not yet have the associated design system.

3. Verify that the application continues to function correctly after this publication.

4. Publish the application so that pages and entities adopt the design system.

5. Access the modeler and verify that the page elements and entities are displayed correctly, without alterations.

6. Confirm that the application functions correctly after applying the published application design system.

If no errors are detected, a set of adjustments must be made to integrate the design system into the already implemented application.

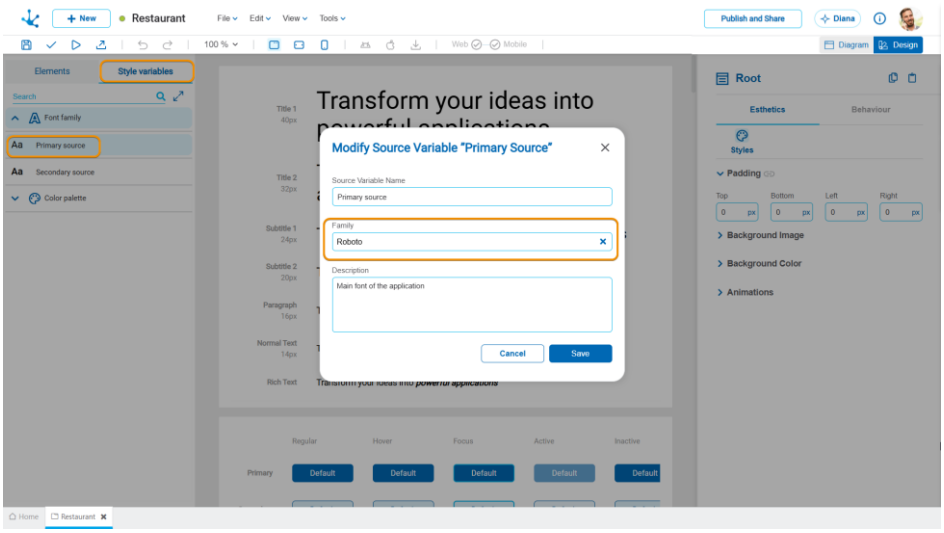
Adjustments to Integrate the Design System

The most representative pages and entities of each application should be considered, and the styles and color codes of the main elements should be identified. These codes and styles can be copied and assigned to the corresponding color variables and elements, within the design system.

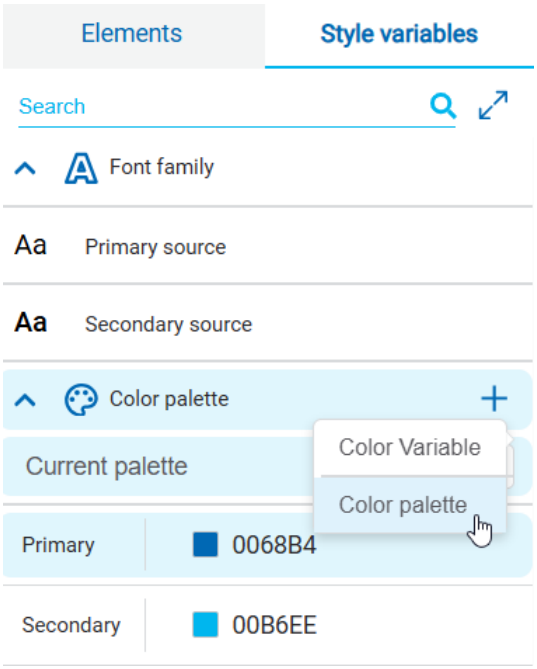
Modifications to Style Variables

It is essential to make all typography and color modifications in the tab "Style Variables", to maintain a centralized configuration and avoid individual changes in each interface.

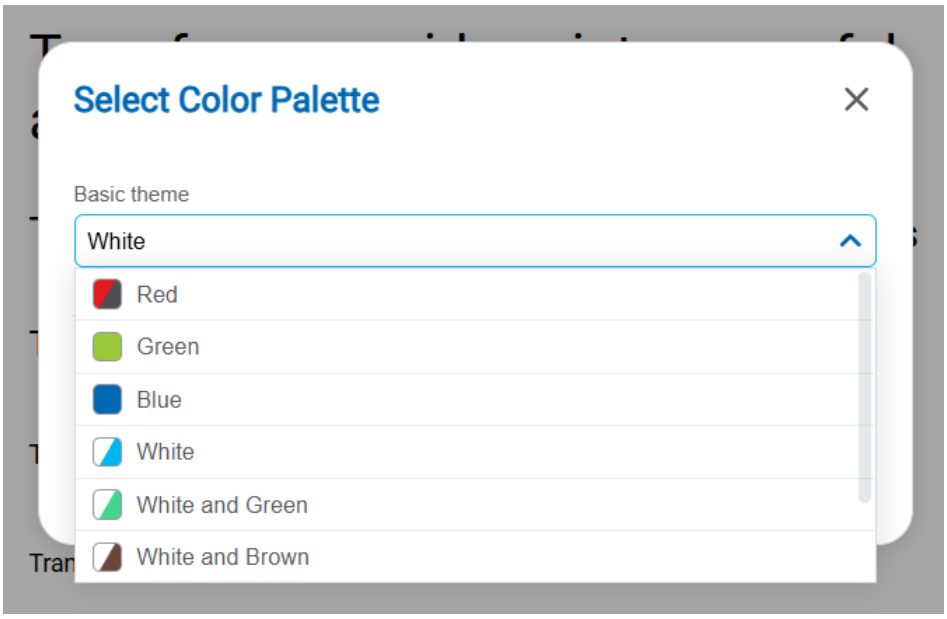
1. Check the font family defined in the variable "Font family" and edit it according to the fonts used in the application.



2. First, determine the base theme defined in the application properties, and then review the configuration in the variable "Color Palette", where the same base theme must be selected to maintain visual consistency among all modeled elements.



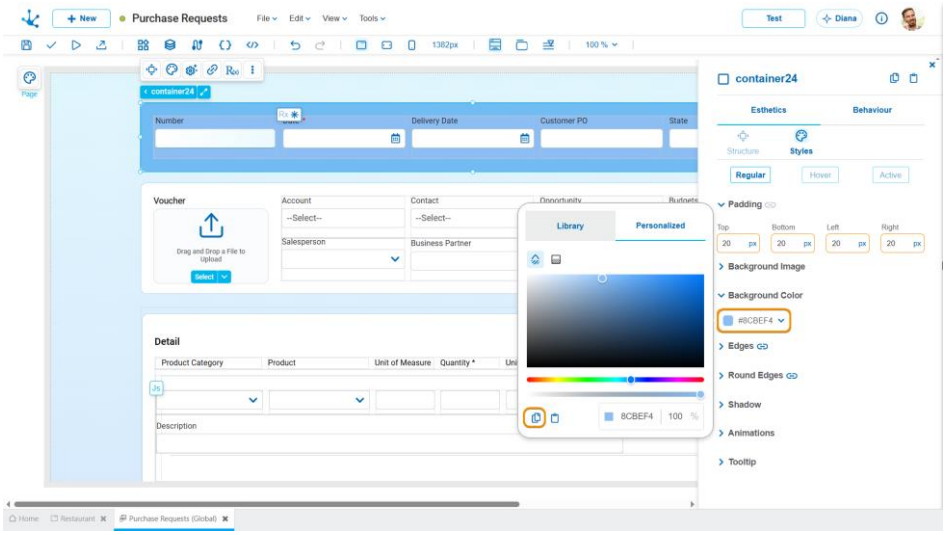
3. In the color palette, choose the appropriate base theme to automatically import the theme colors into the design system.



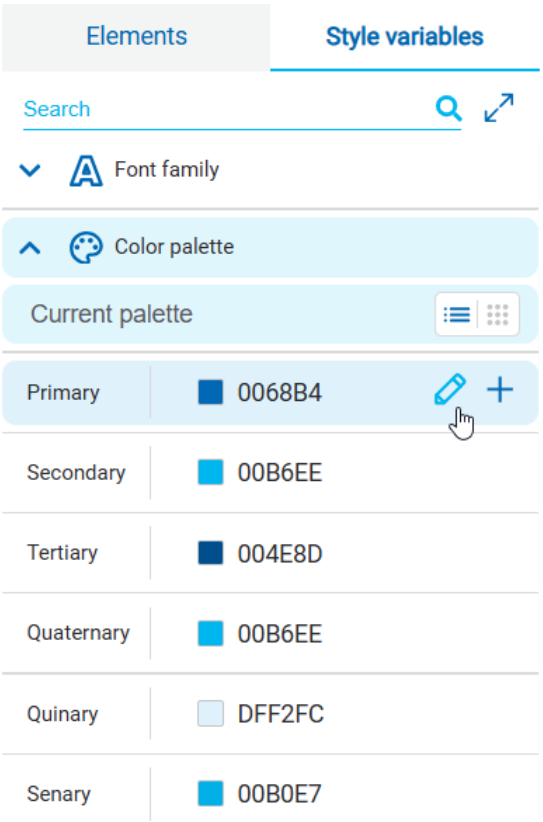
4. Once it has been verified that the theme colors are applied, configure the primary, secondary, and tertiary colors.

Primary Color

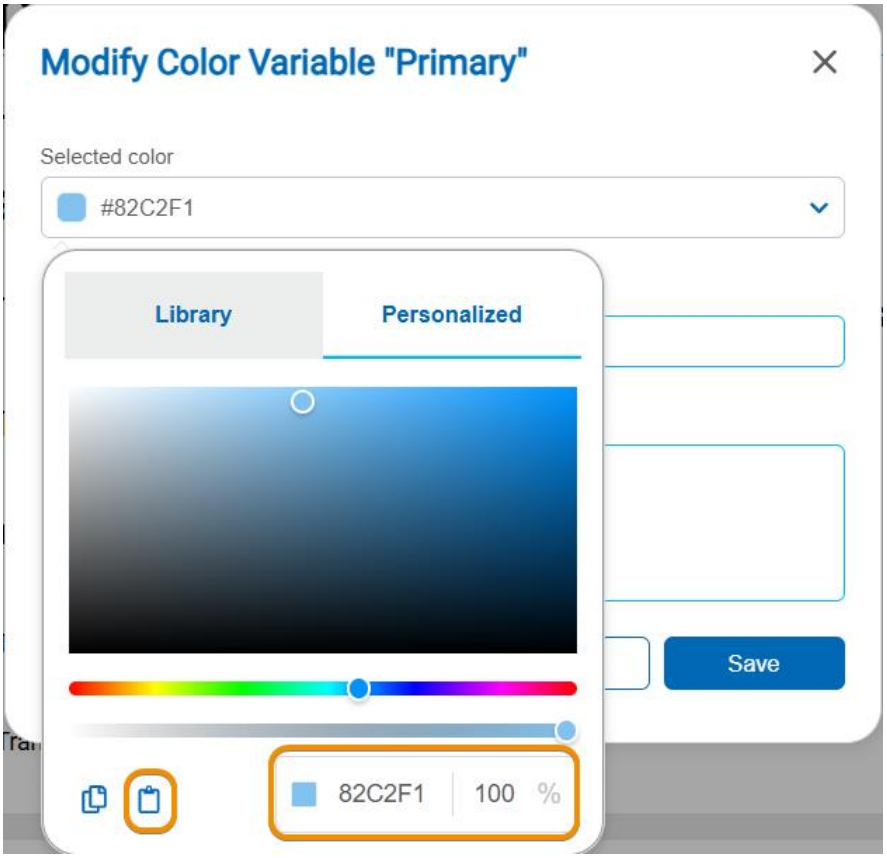
- In the most representative pages and entities of the application, obtain the color code of the main element.



- Select the icon to modify the primary color variable in the design system.



- Paste the element's color code into the primary color variable.



- Repeat this procedure for the secondary and tertiary color variables.

Modifications to the Elements

Within the "Elements" tab, subtype configuration is carried out, allowing the standardization of the behavior and appearance of user interface elements.

Subtype configuration includes the following actions:

- Review the most representative pages and entities.
- Identify the element that best defines each subtype in these objects.
- Apply their properties to the design system.

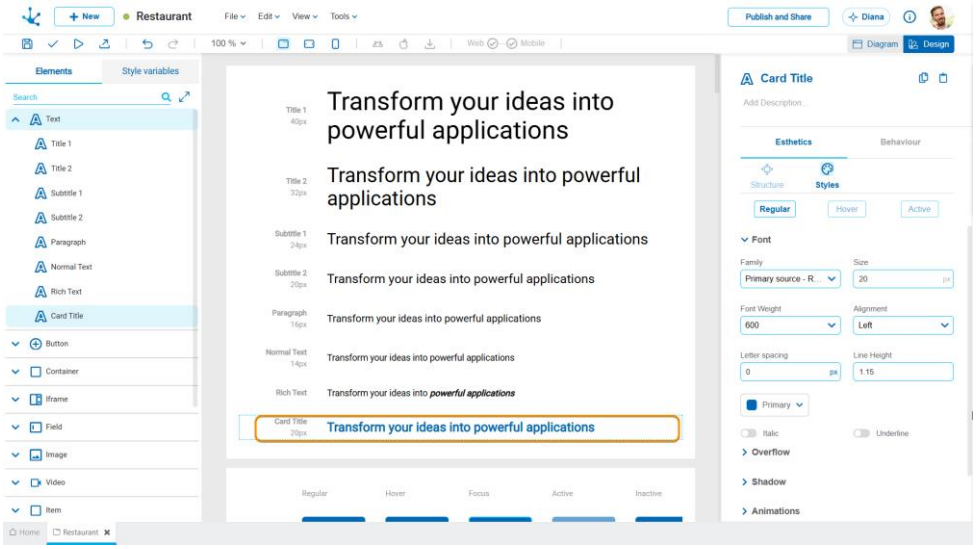
For each element type, it is necessary to review and configure their specific subtypes.

Text

For each text subtype:

- Review the text hierarchy across pages and entities, and configure the design system to maintain visual consistency.
- Verify that the fonts are linked to the defined typographic variables.

- If the default text subtypes are not sufficient, new ones can be created, by assigning them a name and a clear description, for example "Card Title" corresponds to an added subtype.

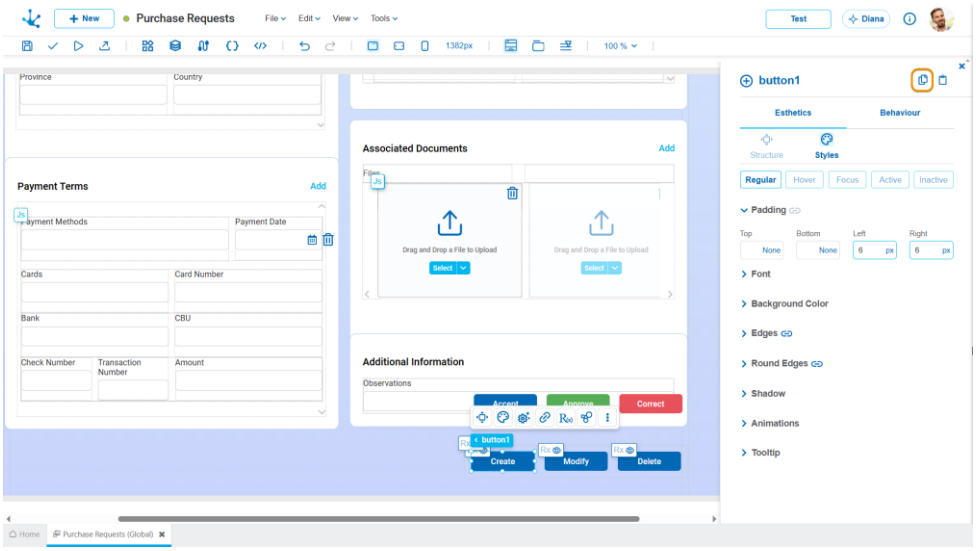


Button

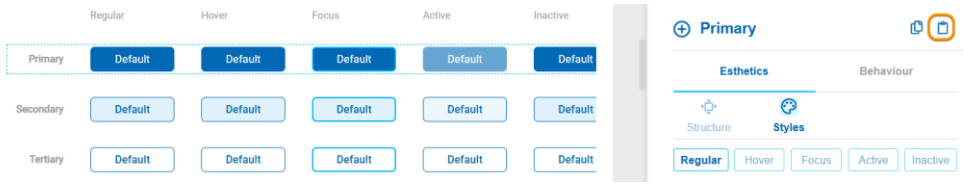
For each button subtype:

- In the design system, define the width and style.
- In the pages and entities of the application, review the font used.

The buttons implemented in the application should be reviewed, and their styles copied for each of their states.



The copied style must be pasted over the button element subtype in the design system, for each button state.

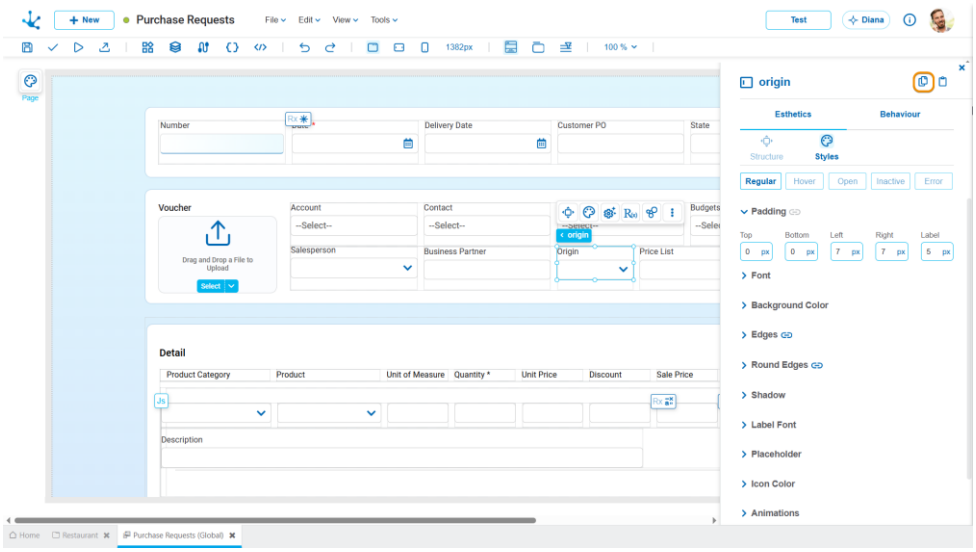


Fields

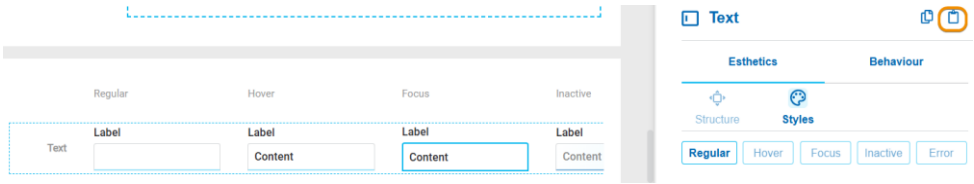
For each field subtype, it is necessary:

- To define the working width in the design system by setting the percentage occupied by the element, for example, 12% or 100%.
- To identify the most representative field of the element in the application pages and entities, in order to incorporate it into the design system.

The fields implemented in the application should be reviewed, and their styles copied for each of their states.



The copied style must be pasted onto the corresponding element subtype in the design system.



Other Types of Elements

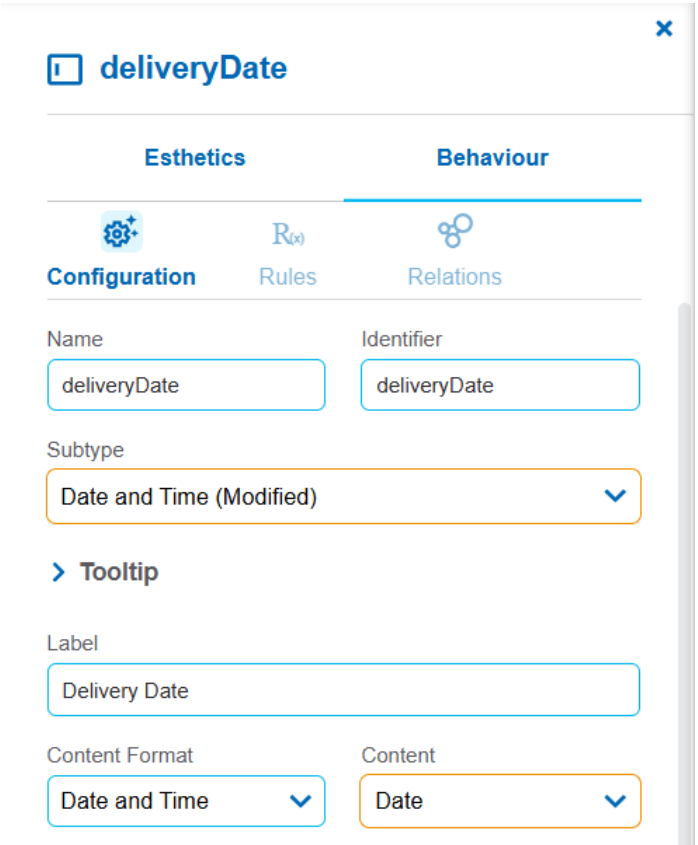
The same principles of review, identification, and subtype configuration described for texts, buttons, and fields must be applied. That is, by identifying the element in the application's pages and entities, copying its properties using the "Copy styles" option and pasting onto the corresponding subtype in the design system. In this way, each behavior of the element maintains the configured styles.

> *Some inconsistency may occur if, when copying and pasting, the source element uses a font or color defined outside of the design system. The design system must be checked to ensure that the configured subtype uses the style variables corresponding to the system. For example, for a text element, the background color must be one of those defined in the color palette.*

Review the Pages and Entities with the Design System

Each element must be associated with the correct subtype from the design system.

On pages and entities, verify that the subtypes are not in the "modified" state that is, marked with an orange border. If any subtype has been modified, review the changed property and either select an existing subtype or standardize the modification in the design system.



Additional Considerations

Reusable Page

Reusable pages respect the design system of their own application. If a reusable page from another application is used, it maintains the design system of its source application.

Automatic Property Assignment

If the value of any property in the design system, such as color or font, matches the value of a modeled element on an existing application page or entity, the subtype name is automatically updated. This simplifies the adaptation of existing applications to the new design system.

3.4.9.3.3. Best Practices

For successful implementation and efficient maintenance of the design system of **Deyel**, it is recommended to follow the following best practices:

- **Scheduling and order:** Plan time to create the design system and work in an organized manner. This involves, among other tasks, defining style variables, identifying common elements in the most representative pages, and correctly assigning subtypes.
- **Assigning a responsible user:** Designate a design system responsible user to ensure its proper management and development.
- **Using descriptions:** Use clear descriptions for elements to facilitate their understanding and use.
- **Avoid making changes in the page or entity panel**: Properties defined in the design system always take priority. It is not recommended to modify them directly in the panel of each page or entity, as this can cause inconsistencies and make maintenance more difficult.
- **Impact of changes when publishing the application**: Changes made to the design system are immediately visible on the diagram. When the application is published, these changes affect all pages and entities that use the subtypes.

Recommendations on Design System Behavior

There are aspects that must be considered after implementing the design system in **Deyel**:

- **Parent width and responsiveness:** It is necessary to review the width of the parent element containing the text subtype elements and ensure consistency in their display. The width of the container can affect the responsiveness of the element at different breakpoints.
- **Rich text element:** It is customized in each object and not standardized. It is recommended to keep the existing rich text element and review its impact on the application with the team.
- **Respect the core logic of the design system:** The design system should be used in accordance with its core logic, meaning that unnecessarily extensions of the color palette or the creation of

non-standard variants should be avoided. This ensures the integrity and consistency of the design system.

The success of the design system of **Deyel** depends on careful planning, disciplined use of components, and ongoing collaboration among team members.
Following these guidelines helps to achieve a more efficient development environment and high-quality applications with a consistent user experience.

3.4.9.4. Mobile Application Distribution

To distribute mobile applications, it is necessary to digitally sign them to validate their integrity and authenticity. These signatures are used by both app stores and devices to verify that the application is authentic.

The following topics detail the steps that must be taken to sign and distribute applications on devices with Android and iOS operating systems.

Android

In the Android operating system, the following tasks must be performed:

- Keystore file generation.
- Collection of Google Services file.
- Configuration of **Deyel** environment properties.
- Distribution of the mobile application via QR code or store.

iOS

In the iOS operating system, the following tasks must be performed:

- Data collection for signature.
- Collection of Google Services file.
- Creation of the certificate and provisioning profile.
- Configuration of **Deyel** environment properties.
- Distribution of the mobile application via QR code or store.

3.4.9.4.1. Android Applications

This topic details the tasks that must be performed to complete the distribution of mobile applications on devices with Android operating system.

For these tasks, it is important to be familiar with the following concepts:

- Keystore
  Binary file that contains cryptographic keys, such as private key password and digital certificates, that are used to sign and authenticate Android applications before they are distributed.

- Keystore-alias
  Alias assigned within the Keystore file to a private key password and its corresponding certificate.

- Keystore-password
  It is the password used to access and unlock the key passwords stored within the Keystore file.

- Store-password
  It is the password used to protect the Keystore file.

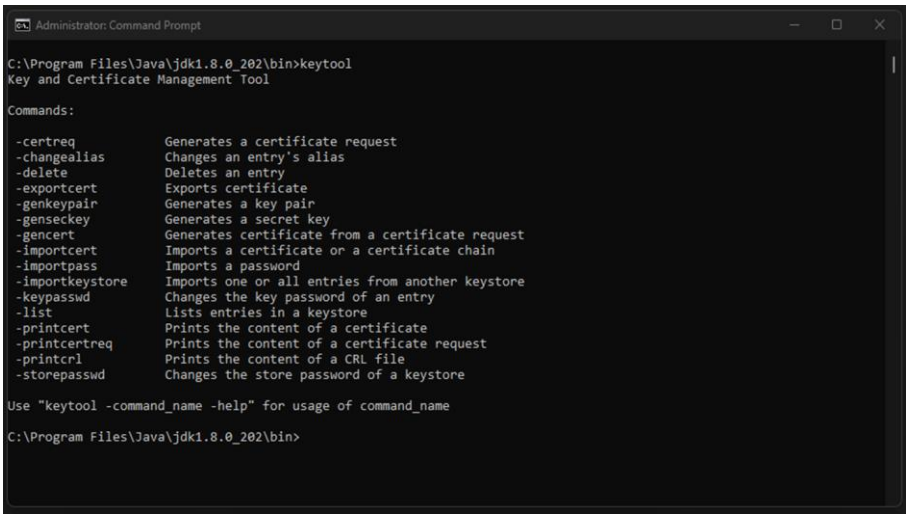
- Google-services
  File that allows interacting with Google services that are used in the mobile application.

Keystore File Generation

The Keystore file is used to sign APK applications. To generate it, the following steps must be followed:

1. Install Java JDK for Mac or Windows.

2. Open the command line interface and execute the `keytool` command located in the Java JDK bin folder, for example `C:\Program Files\Java\jdk1.8.0_202\bin`.



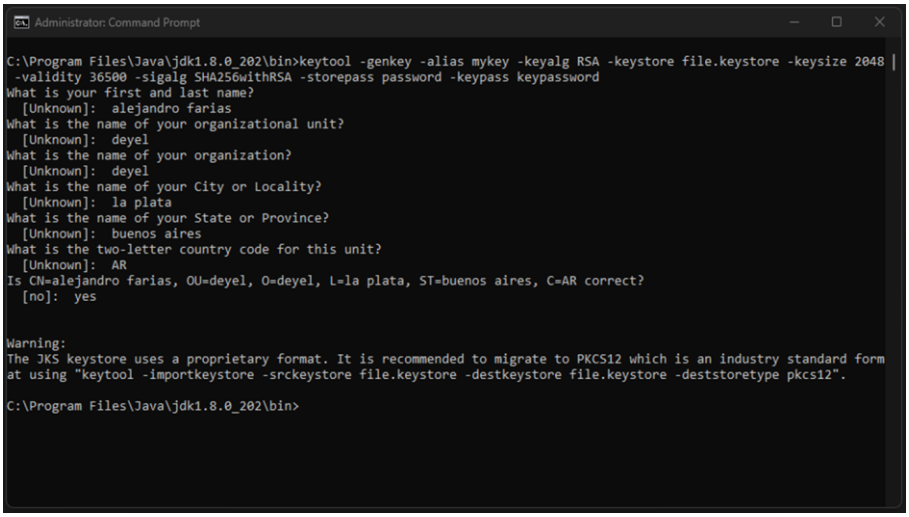
3. Write the following instruction on the command line:

   ```
   keytool -genkey -alias KEYSTORE-ALIAS -keyalg RSA -keystore
   file.keystore -keysize 2048 -validity 36500 -sigalg SHA256withRSA -
   storepass STORE-PASSWORD -keypass KEYSTORE-PASSWORD
   ```

   Values for KEYSTORE-ALIAS, STORE-PASSWORD and KEYSTORE-PASSWORD must be defined and replaced in the previous statement.

   > *If access is denied, open the command line interface in administrator mode.*

4. Answer the following questions, pressing “Enter” after each one and type “yes” when asked to confirm the information.



5. Upon finishing answering these questions, the Keystore file is generated and saved in the current working directory.

Collection of Google Services File

The mobile applications generated by Deyel use Google services internally, which are necessary to, among other things, receive pop-up notifications on the device.

To enable these Google services, it is necessary to follow the steps mentioned in the guide for collection of Google Services file .

Configuration of the Properties of Deyel Environment

To distribute signed applications, it is necessary to complete the properties of the Deyel environment with the data values for the signature previously obtained in the generation of the Keystore file.

- KeyStore: The Keystore file generated must be converted to base64 format. Conversion can be done using online converters, for example https://base64.guru/.
- KeyStore Password: The Keystore-password value used for signing in the command-line interface.
- KeyStore Alias: The Keystore-alias value used for signing in the command-line interface.
- Store Password: The Store-password value used for signing in the command-line interface.

> *Given that in the Android operating system it is not strictly necessary to sign an application to install it on mobile devices, it is optional to complete the properties of the **Deyel** environment in the distribution via QR code, but it is not optional if distribution is through Store.*

If an application has been modeled to allow using the notifications functionality "Push", it is necessary to ensure that the previously downloaded file in the property "Enable Push Applications" is configured according to the *Android Application distribution guide*, at the "Obtaining the Google Services File" section.

Distribution of the Mobile Application

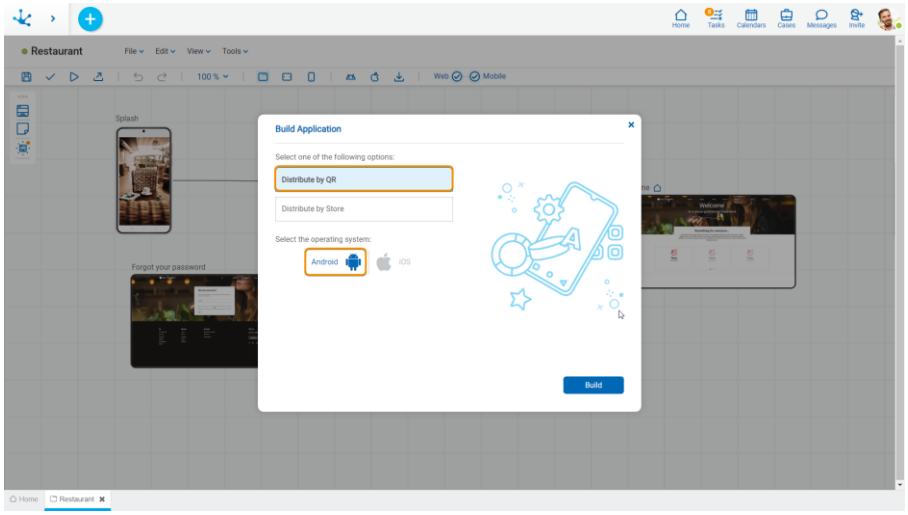
The distribution of the mobile application can be done via QR code and through the application store, each with different characteristics.

Distribution via QR Code

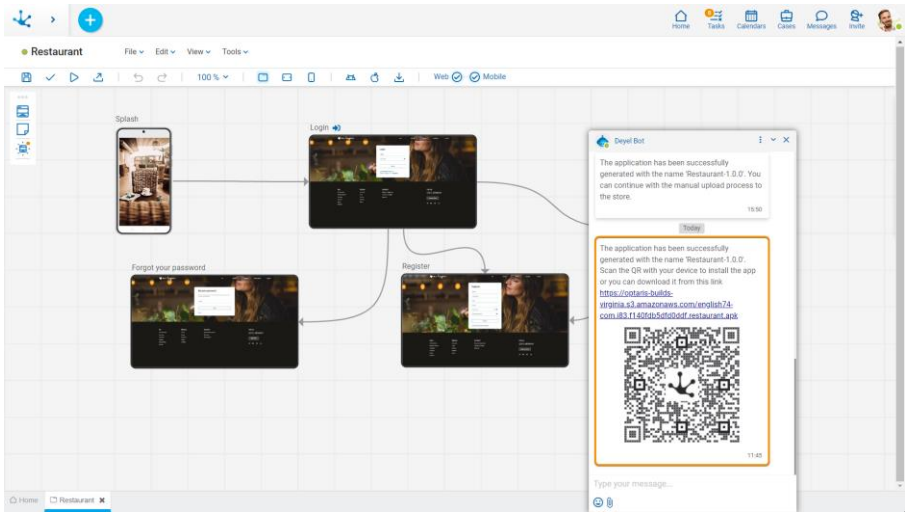
This distribution results in a file with an ".apk" extension (Android Application Package) that can be installed on any device with the Android operating system.

To get this file, it is necessary to perform the following steps:

1. Open the option "Build for Android" from the expanded menu of the applications modeler and in the open panel select distribution via QR code in the Android operating system.



2. After a few minutes, a message from Deyel Bot is received with the result of the operation, a link to the file with an ".apk" extension and a QR code to enable installation directly from the device, using the camera or a QR code reader. Additionally, it is also possible to download from the option "Download App" from the expanded menu of the applications modeler.



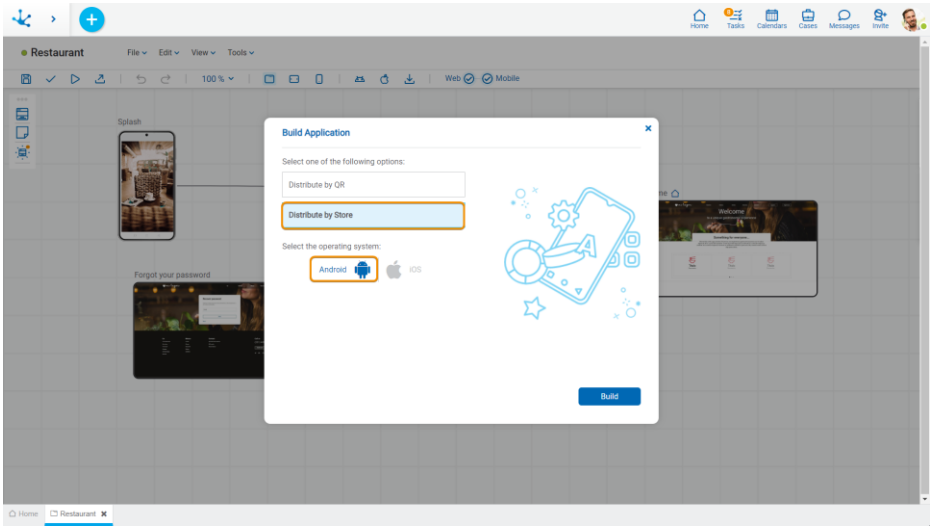
Distribution via Store

This distribution results in a file with an ".aab" extension (Android App Bundle) that is required by Google so that the application can be uploaded to the store.

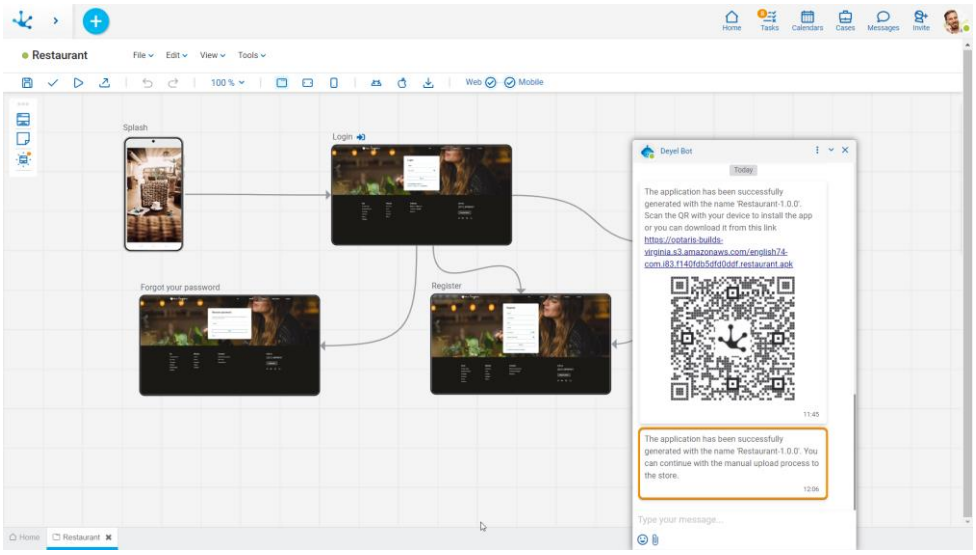
> *The properties of the **Deyel** environment must be previously configured according to the step "Configuration of the Properties of Deyel Environment" .*

To obtain the file with ".aab" extension (Android App Bundle), it is necessary to perform the following steps:

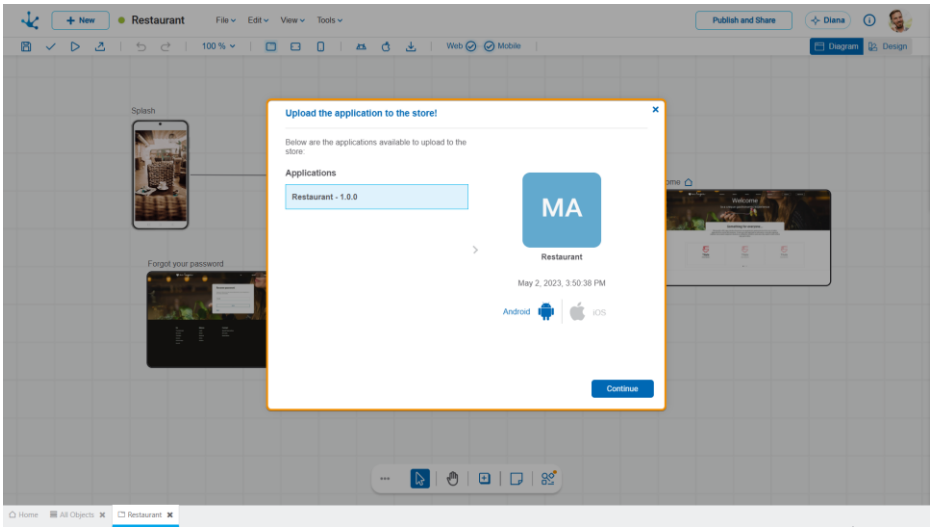
1. Open the option "Build for Android" from the expanded menu of the applications modeler and in the open panel select distribution via QR code in the Android operating system.



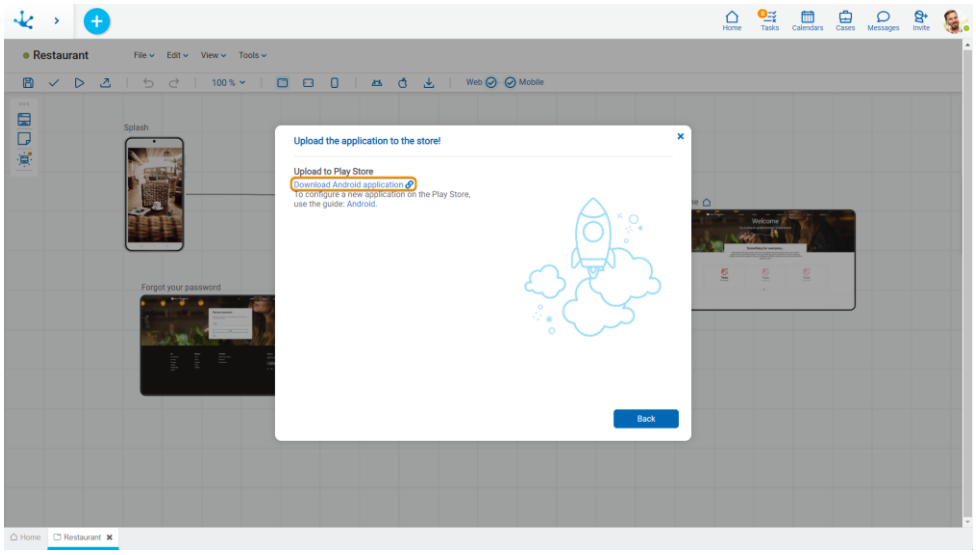
2. After a few minutes, a message from Deyel Bot with the result of the operation is received.



3. Next, the option "Upload to the Store" should be opened from the expanded menu of the applications modeler, where the application is displayed as available for selection. Select the application and click on the "Continue" button.



4. In the next panel, the "Download application for Android" option must be selected to advance and the result is a file with an ".aab" extension that must be used to upload to the store.



5. The application is ready to be uploaded to Google Play. The procedure to configure a new application and upload the file with the ".aab" extension (Android App Bundle) just downloaded is described in the Google guide for uploading an application.

3.4.9.4.2. iOS Applications

This topic details the tasks that must be performed to complete the distribution of mobile applications on devices with the iOS operating system.

For these tasks it is important to know the following concepts:

- AppleID
  Unique identifier for the Apple developer account. This account is obtained by making a subscription to Apple as detailed in the following steps.

- Apple team identifier
  Identifier that refers to certificates and profiles defined in the Apple developer account.

- Password for Apple services
  Password used to upload the application to the store.

- Google-services
  File that allows interacting with Google services that are used in the mobile application.

Data Collection for Signature

Signatures are used to validate the integrity and authenticity of a mobile application. The app store and each mobile device use the signature to perform this validation.

Certificates identify who has signed an application or wants to use a service. They have a code signature to assure the user that an application comes from a reliable source and has not been modified since the certificate was requested.

Provisioning profiles authorize the application to use certain services and ensure that the developer is known, that is, develops, uploads, or distributes the application.

To obtain the certificate and provisioning profile, the guide to get the signature for iOS applications must be followed.

Configuration of the Properties of Deyel Environment

To distribute signed applications, it is necessary to complete the properties of the Deyel environment with the values previously obtained in the previous step.

- Apple ID: Email registered to the Apple membership, which is used as the Apple ID identifier to access the developer account control panel.
- Apple team identifier: Apple team identifier obtained in the membership detail section.
- Password for Apple services: Password received for Apple services.
- Apple Certificate: When downloading the certificate from the Apple developer panel, it has a ".cer" extension and it should be converted to ".p12" format before configuring this environment property.

  - In the command line interface, convert it to a file with extension ".p12" as follows:

    - Change the iOS certificate format to a file with extension ".pem".

      ```
      openssl x509 -inform DER -outform PEM -in distribution.cer -out distribution.cer.pem
      ```

      where "`distribution.cer`" is the name of the certificate and "`distribution.cer.pem`" is the name of the resulting file.

    - Convert file extension ".pem" to ".p12".

      ```
      openssl pkcs12 -export -inkey keyname.key -in distribution.cer.pem -out distribution.p12
      ```

      where "`distribution.cer.pem`" is the file name ".pem" and "`distribution.p12`" is the name of the resulting file.

    - The file with extension ".p12" generated must be converted to base64 format. The conversion can be done using online converters, for example https://base64.guru/.

  > *Both commands must be executed in the same directory where the private key was generated according to the iOS application distribution guide in the "Certificate Creation" section, step 2.a.*

  > *The password associated with the file with extension ".p12" corresponds to the one entered when executing the conversion command ".pwm" to ".p12".*

> *Another alternative to obtain the file with extension ".p12" is to perform the conversion in https://www.leaderssl.es/tools/ssl_converter.*

- Apple certificate password: Corresponds to the password of the file with extension ".p12" previously generated.

Additionally, it is necessary to complete the provisioning profile property with the file with extension ".mobileprovision", downloaded from the Apple developer panel, according to the instructions given in the previous section "Obtaining Data for the Signature". This file must be different depending on whether it is to be distributed via QR code or via the application store.

If an application has been modeled to allow using the notifications functionality "Push", it is necessary to ensure that the previously downloaded file in the property "Enable Push Applications" is configured according to the iOS application distribution guide, at the "Obtaining the Google Services File" section.

Distribution of the Mobile Application

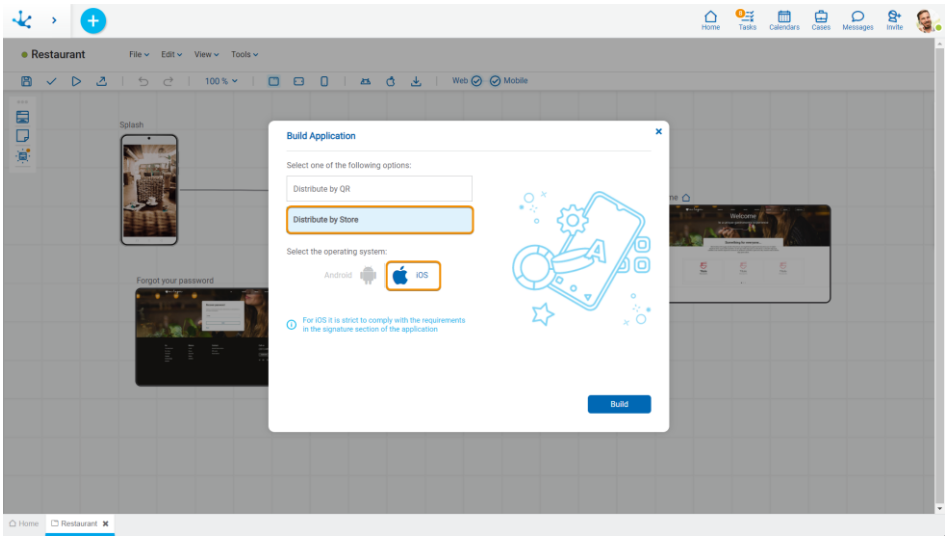
The distribution of the mobile application can be done via QR code and through the application store, each with different characteristics.

Distribution via QR Code

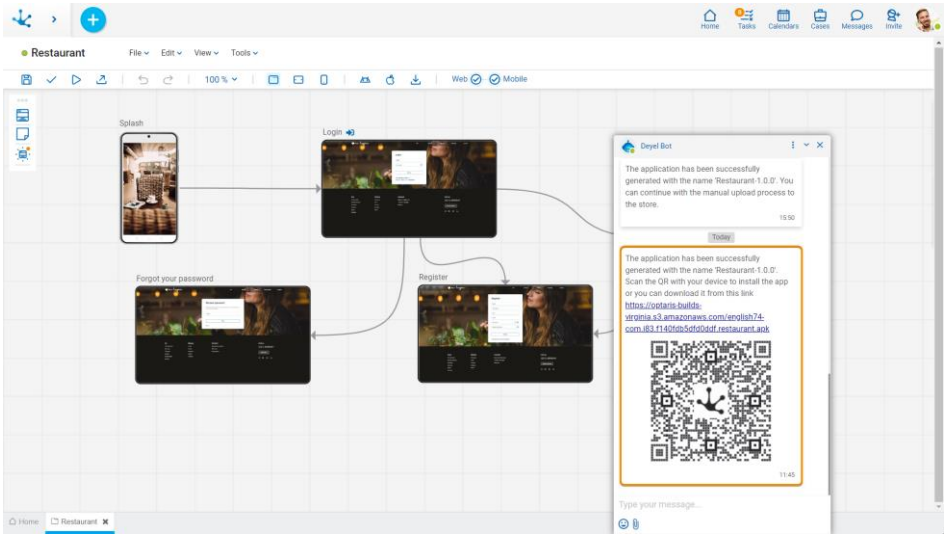
This distribution results in a file with an ".ipa" extension (iOS AppStore Package) that can only be installed on a device with the iOS operating system that has been registered in the Apple developer panel, according to the iOS application distribution guide, in step 6 of section "Provisioning Profile Creation".

Once the data for the signature has been obtained and the properties of the **Deyel** environment have been configured, it is necessary to perform the following steps to create the file with extension ".ipa":

1. Select the option "Build for iOS" from the expanded menu of the applications modeler and in the open panel select distribution via QR code in the iOS operating system.



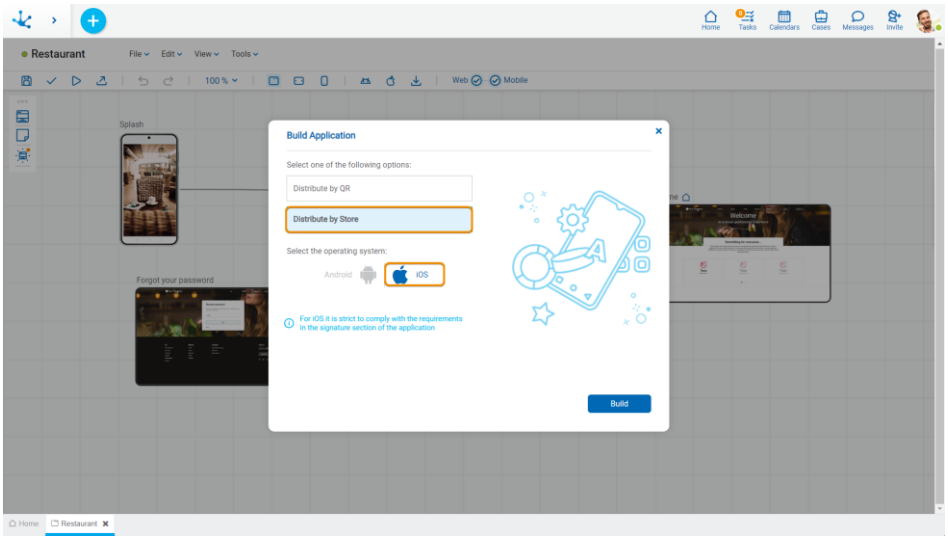
2. After a few minutes, a message from Deyel Bot is received with the result of the operation, a link to a Safari website to download the file with the ".ipa" extension and a QR code to enable installation directly from the device, using the camera or a QR code reader. Additionally, it is also possible to download from the option "Download App" from the expanded menu of the applications modeler.



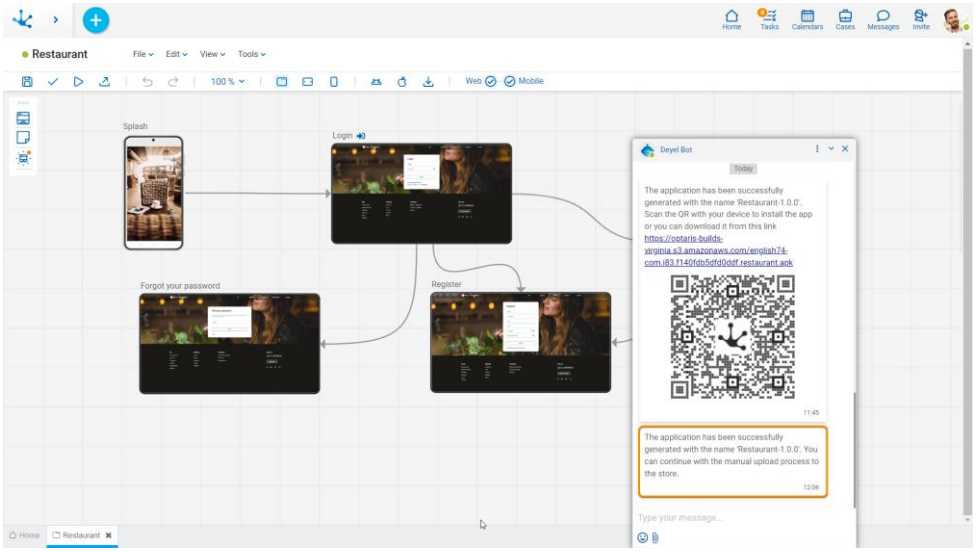
Distribute via Store

Once the data for the signature has been obtained and the properties of the **Deyel** environment have been configured, the following steps must be carried out to upload to "App Store Connect", which is the platform that administrates the applications that reach the Apple Store:

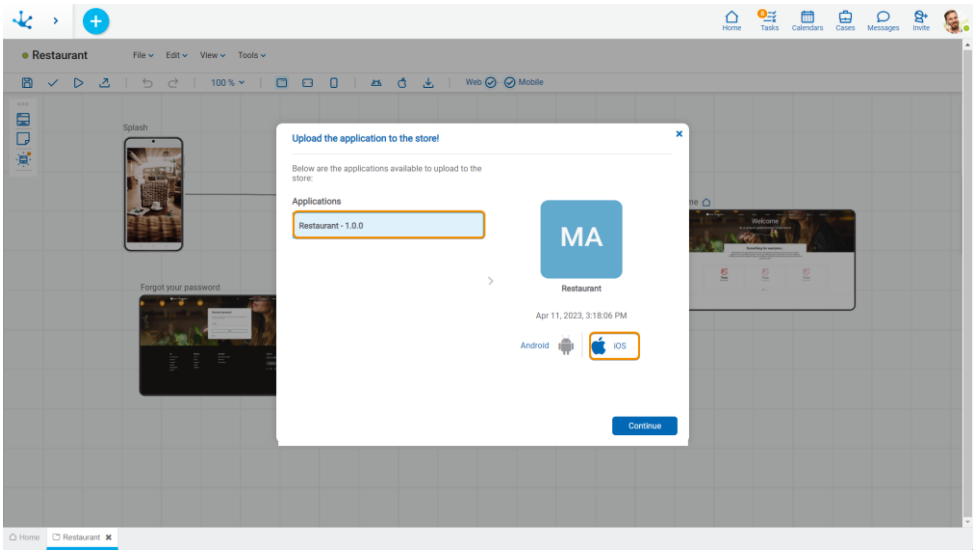
1. Select the option "Build for IOS" from the expanded menu of the applications modeler and in the open panel select distribution via store in the iOS operating system.



2. After a few minutes, a message from Deyel Bot with the result of the operation is received.



3. Next, the option "Upload to the Store" should be opened from the expanded menu of the applications modeler, where the application is displayed as available for selection. Select the application and click on the "Continue" button.



4. As a result, the application was uploaded to “App Store Connect”. To finish uploading to the App Store, the manual steps described in the iOS application distribution guide, in section "Distribution through Store" must be performed .

3.4.9.5. Progresive Web Application

All **Deyel** applications are Progressive Web Apps (PWA). These are web applications that use web capabilities to offer an experience similar to that of a native application on the device that the user usually uses, for Android or iOS operating systems, with an optimized interface, quick response, and access to the device's functionalities.

They combine the best of web and native applications, providing an improved and robust user experience on the web. They work on all browsers thanks to the concept of progressive enhancement, are adaptable to different screen sizes, from mobile devices to desktops, and are independent of connectivity, meaning they can operate offline or on low-quality networks.

Progressive enhancement is a strategy that prioritizes basic, functional content for all users while providing enhanced functionality and a richer experience for those with more advanced browsers and devices. This is achieved by first building a basic version of the application and then adding layers of enhancements that take advantage of the more advanced capabilities of modern browsers.

When running **Deyel** applications, from its “Home” page, they can be installed from the browser on both a desktop and a mobile device. They can be easily added to the device's home screen, without having to go through an app store. They allow the use of "push notifications and other functionalities to maintain user engagement.

Features

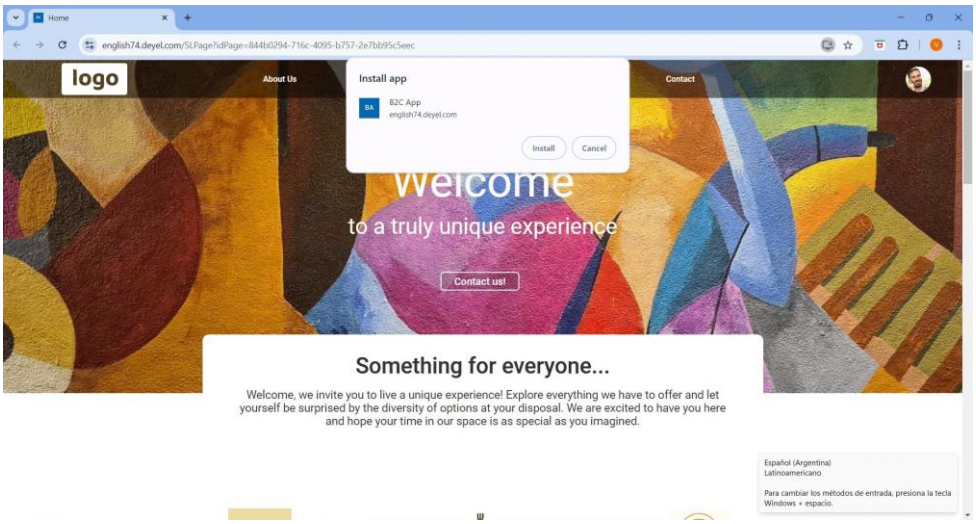
PWA applications are characterized by having the following functionalities:

- Progressive: They work for all users, regardless of the browser they use, as they are built from basic content that can then be enhanced with advanced capabilities.

- Responsive: They adapt to different screen sizes, from mobile devices to desktop computers.
- Connectivity independent: They use the concept "service workers" to work offline or on low-quality networks.
- Similar to native apps: They look and work like a native application on the chosen device, providing a similar user experience.
- Upgradable: They are always up-to-date thanks to the update process of the "service workers" concept.
- Safe: They rely on HTTPS to protect the integrity of the application and user data.
- Installable: Users can install them on their device's home screen without needing to go through an app store.
- User engagement: They can send notifications "push" and have additional functionalities to maintain user engagement.

Installation

When running a **Deyel** application, it can be installed on the selected device following different steps, depending on the browser used.



Desktop computer

Steps for Installation in Google Chrome

1. Open the Chrome browser and go to the URL of the PWA application modeled in **Deyel**.
2. Since it is a PWA application, an installation icon is displayed in the browser address bar, usually on the right. Click on the icon, which represents a small computer with an arrow, or, in the three-dot menu at the top right of the browser, select the option "Install <nombre de la aplicación>".

3. Follow the instructions to install the application.
4. Once installed, the PWA app opens in a separate window and can be found in the device's app menu.

Steps for Installation in Microsoft Edge

1. Open the Edge browser and go to the URL of the PWA application modeled in **Deyel**.
2. Since it is a PWA application, an installation icon is displayed in the browser address bar, usually on the right. Click on this icon, or, in the three-dot menu at the top right of the browser, select the option "Applications" and then "Install this application".
3. Follow the instructions to install the application.
4. Once installed, the PWA application opens in a separate window and can be found in the device's application menu or home.

Mobile device

Steps for Installation in Google Chrome (Android/iOS)

1. Open the Chrome browser and go to the URL of the PWA application modeled in **Deyel**.
2. Since it is a PWA application, an installation icon or an "Add to home screen" message is displayed on Android. Click on the icon or message, or alternatively, use the three-dot menu at the top right of the browser, select the option "Add to home screen". On iOS, the share icon should be pressed and the option "Add to home screen" should be selected.
3. Confirm the operation in the window where the application name and the "Add" button are displayed. By pressing this button, the PWA application is installed and can be found on the home screen of the mobile device.

Steps for Installation on Safari (iOS)

1. Open the Safari browser and go to the URL of the PWA application modeled in **Deyel**.
2. Click on the share icon at the bottom of the browser and select the option "Add to home".
3. Confirm the operation in the window where the application name and the "Add" button are displayed. By pressing this button, the PWA application is installed and can be found on the home screen of the mobile device.

Steps for Installation in Microsoft Edge (Android/iOS)

1. Open the Edge browser and go to the URL of the PWA application modeled in **Deyel**.
2. Click on the three dots at the top on Android or at the bottom on iOS and select the option "Add to home screen" or "Install this app" depending on the case.
3. Confirm the operation in the window where the application name and the "Add" button are displayed. By pressing this button, the PWA application is installed and can be found on the home screen of the mobile device.

Steps for Installation in Mozilla Firefox (Android)

1. Open the Firefox browser and go to the URL of the PWA application modeled in **Deyel**.
2. Since the application is a PWA, in the three-dot menu in the top right corner, select "Add to home screen". In this way, Firefox creates a shortcut on the home screen that behaves like a PWA.
3. After confirming the operation, the PWA application can be found on the home screen of the mobile device.

3.4.9.6. Best Practices

The best practices for the applications modeling in **Deyel** consist of a set of recommendations, to be met whenever possible.

They allow to build clearer and more understandable applications models.

Below are some best practices that should be considered in the construction of pages, oriented to the information contained in them, to the way of exposing it in relation to the users for whom it is intended, to the type of application or breakpoints that are selected, to the design patterns that define the user interface and the visual consistency of the pages.

Information contained in the applications

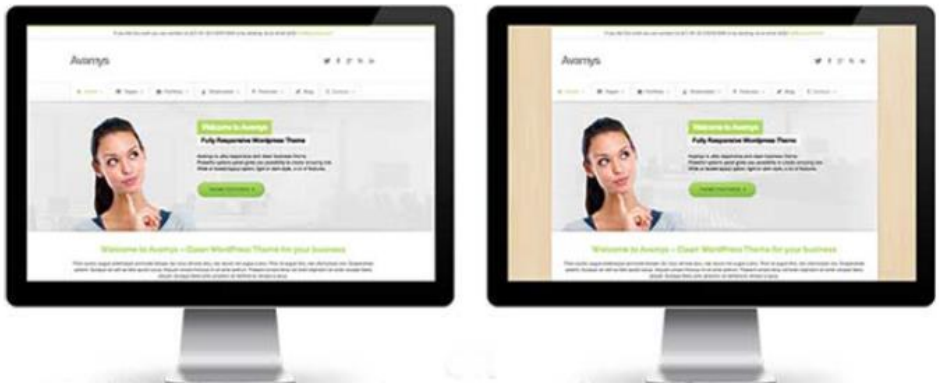
- Define the structure, hierarchies, elements that compose the user interface.
- Determine what message you want to convey.

Users for whom the applications are intended

- Define what type of users we are targeting.
- Consider this characteristic to establish the aesthetics to model.

Selection of application type and breakpoints

- Before starting to model applications, you must define what type of application you are going to created, if it is web or mobile.
- After defining the type of application, it must be determined which breakpoints are to be considered. If it is mobile, the tablet and mobile breakpoints must be considered, while if it is web, all three breakpoints must be considered.
- In the case of the desktop breakpoint, you must decide whether to work as full width or maximum width. This means that you must define whether you want the page to grow to the full width or limit the width to a certain measurement that is established.



Total Width **Maximum Width**

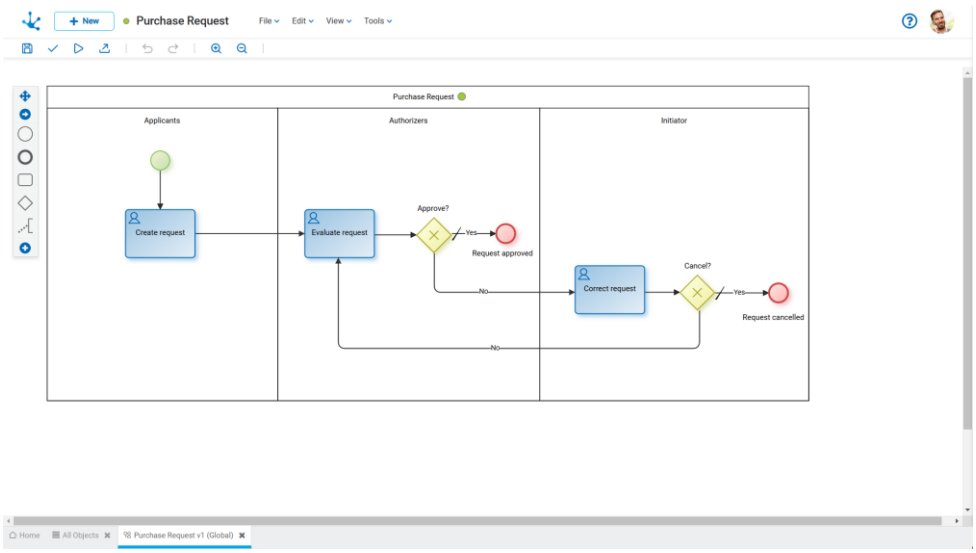
3.4.10. Processes Modeling

Processes Modeling

The process modeler from **Deyel** is a tool which allows modelers to design business processes in a collaborative work environment, using symbols from **BPMN 2.0** (Business Process Model and Notation).

By using this intuitive tool, the different business units can visualize the processes that are carried out within an organization and easily understand them.

When creating a new process or opening one already defined, it is allowed the access to a design space where the business process can be defined. Once it is finished, the process can be published and is available for use within the organization.



The general characteristics of the process modeling environment and the main elements that compose it, are described in the topics:

- Modeling Facilities
- Graphic Elements

3.4.10.1. Modeling Facilities

Process Modeling> Introduction to process modeler

Process Modeling> Modeling a process from a template

General characteristics of this modeler are specified below.

New Process

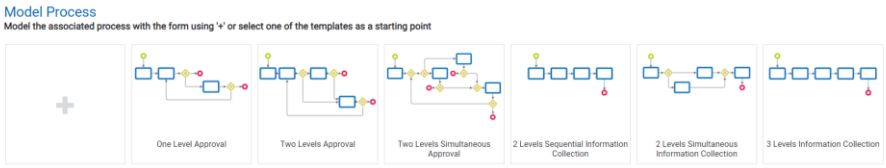
The modeler user can design new procesess which, after being defined and published, can be used both by the applications modeled in **Deyel** or by any other application.

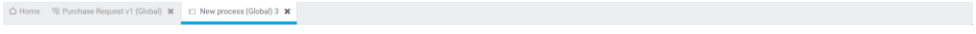
In the same workspace and by using concepts from **BPMN 2.0** (Business Process Model and Notation), it can be simply modeled in an intuitive environment:

- The process diagram with its activities, interconnections, events, time, alerts and actors involved, in a way that allows interaction with the business user and that they easily understand the process scope.

- Business rules which define the behavior of each of the diagram elements, as well as the validations, automatic actions for sending messages and integration with other applications.

Steps for Creating a New Process





Step 1: Open the Assistant with Templates



This button is used to create a process from the option  Process.

Allows to define the new process selecting one of the predefined templates to base on, or starting from an empty design area.

Step 2: Enter the Templates Properties

The properties are grouped in containers.

Basic Information

Presents information common to all processes.

Name of the Process

The modeler should enter the name of the process they are creating.

Owner

User, organizational unit or role that is modeling the process, it is the responsible user of the process maintenance. The predetermined owner is the logged in user.

 Opens the profile show panel.

 Deletes the selected participant.

 Selects an existing participant. This icon is enabled when deleting a participant.

Initiator

The participant who can initiate the process, it can be a user, an organizational unit, a role or an agent. For a participant to start a process, it must be published. The predetermined initiator is the agent All users.

 Opens a “New Participant” window to define a new role.

 Opens the profile show panel.

 Deletes the selected participant.

 Selects an existing participant. This icon is enabled when deleting a participant.

Participants

This container is only visualized if the new process is defined from a predefined template. The participants in the container depend on the type of template that has been selected. If an authorization template is selected, authorizing users will be displayed, while if a data collection template is selected, participating areas will be displayed. The predetermined value for the participants is the logged in user.

As many participants as there are actors in the selected template are defined. For each of the participants, a specific user can be selected, as well as an organizational unit, a role or an agent.

 Opens a window to define a new role.

 Opens the profile show panel.

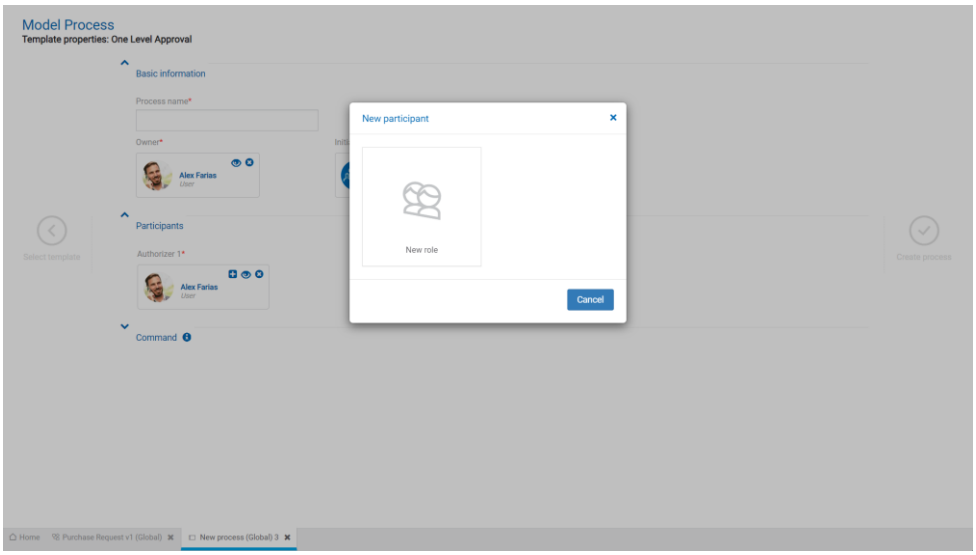
Deletes the selected participant.

Selects an existing participant. This icon is enabled when deleting a participant.

New Participant

New Role

Allows to enter the name of the new role and the users that make it up.



Commands

A process can be used as a command, in such case the following properties must be completed.

Start from Chat

Indicates if a user can initiate the process from a conversation. If such property is checked the following must be completed:

Command Name

Allows to enter the command name with which the process can be initiated from a conversation.

Define New Assistant

Indicates if a new chatbot is defined or if an existing one is selected. By default the user can visualize the property Select Assistant Bot.

Assistant Bot Name

If the property Define New Assistant is checked, the name of it must be indicated.

Select Assistant Bot

If the property Define New Assistant is not checked, there must be a selection for which chatbot has the command.

Step 3: Close Properties Panel

Select Template
The user can return to the templates assistant to select a new one.

Create Process
The user moves on to the process workspace and depending on the option selected in the assistant, the graphic modeling area is displayed with a different content:

- With the process diagram corresponding to the selected template.
- With the minimal graphic elements so that a process can be saved, if none of the predefined templates have been selected.

Workspace Sections

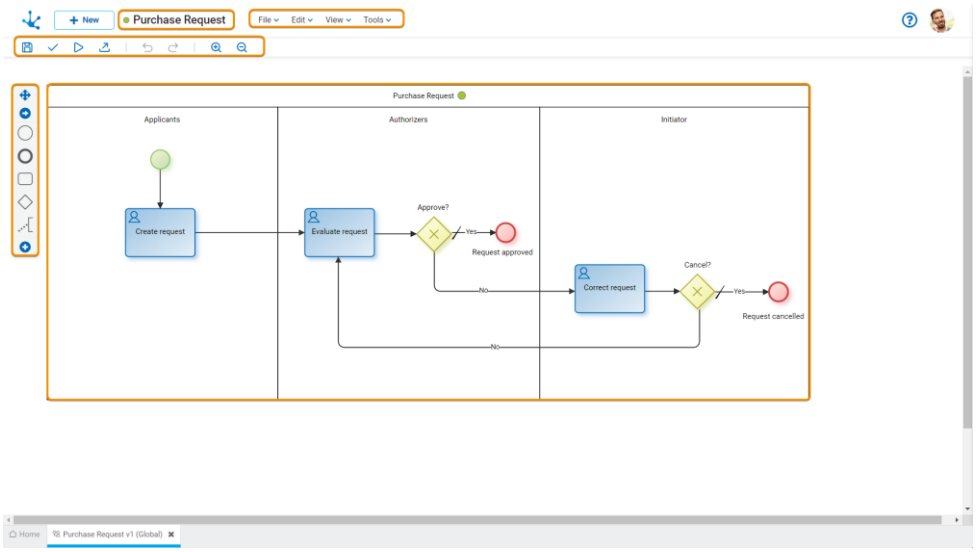
• Process Information
  - State
  - Name

• Expanded Menu

• Top Toolbar

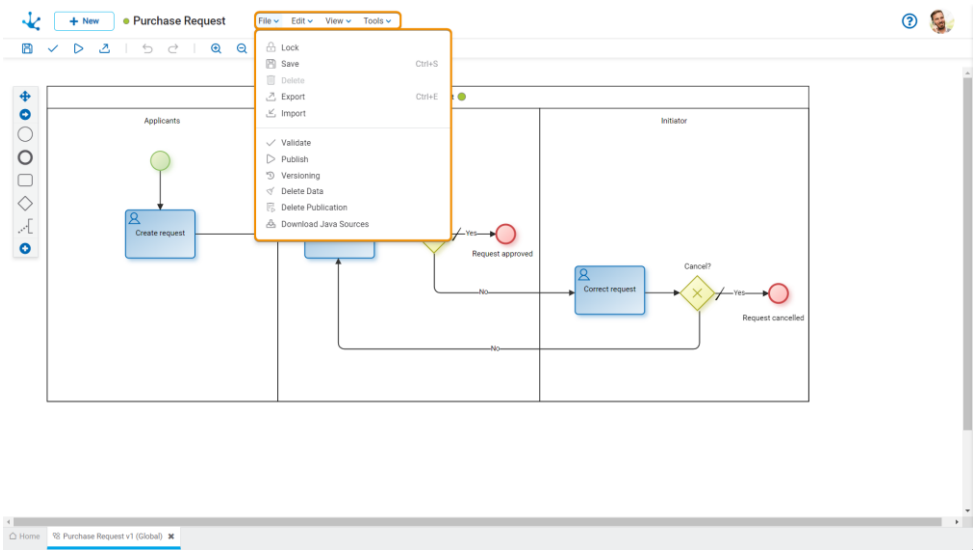
• Graphic Modeling Area

• Side Toolbar



3.4.10.1.1. Expanded Menu

It is a horizontal list of options containing vertical submenus with different operations on the process or on its modeling. In turn, each option in a submenu may expand a dependent submenu.

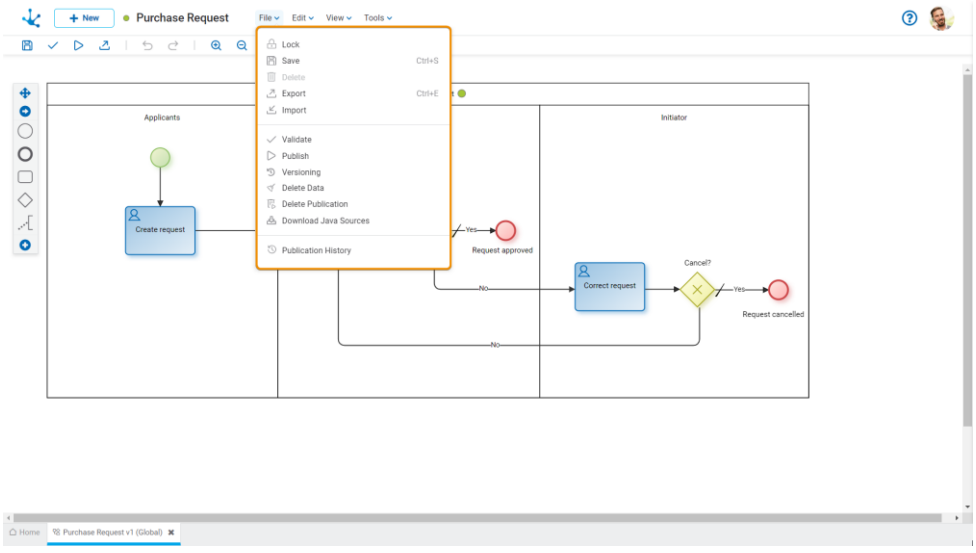


The expanded menu is made up of:

- File Submenu
- Edit Submenu
- View Submenu
- Tools Submenu

3.4.10.1.1.1. File Submenu

This submenu opens by clicking on the “File” option and allows the performance of operations on the process.



/ Lock/Unlock

It allows locking a process diagram to ensure that no one can modify it until the user currently using it unlocks or releases it.

It allows unlocking a process so that another user, organizational unit or role, defined as owner, can modify it.

Save

This icon allows to save the object in the repository of **Deyel**, leaving its state as "Draft" or "Modified". If certain conditions are met, the modeler user receives a message indicating that the operation was performed correctly, otherwise they receive an explanatory message.

Conditions

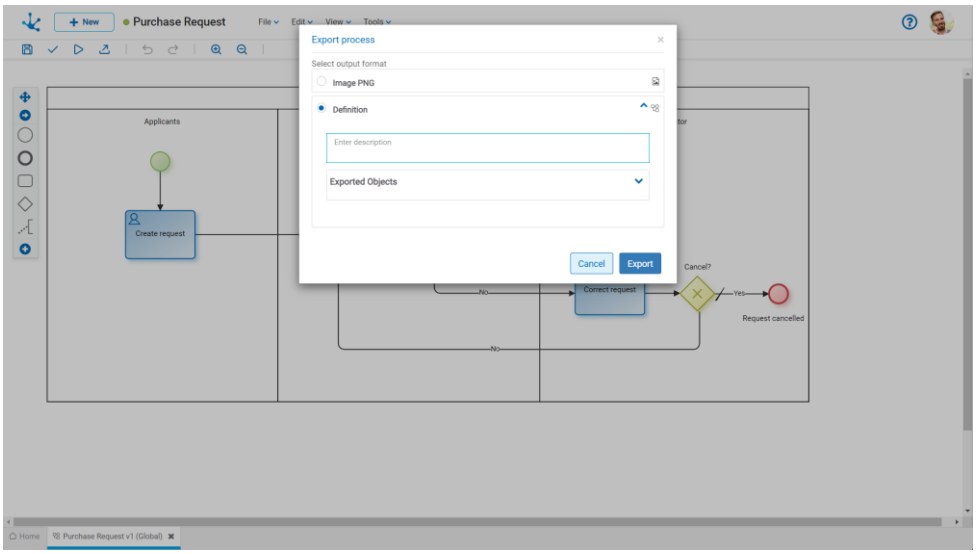
- The object application is required.
- The name in the application must be unique.
- The object should not be locked by another user.

Delete

It allows to delete the object only if it is in "Draft" state and does not have other associated objects of **Deyel** that were previously saved or published. When deleted, the tab it is on closes and the object is removed from the modeler's grid.

Export

This icon opens a window for the user to select and confirm the export of the object.



Output Format

PNG Image
A file with a .png extension is generated, containing the image of the selected area of the process.

Definition
It is the default export format for all objects.

Description
In this property a text explaining the reason for the operation can be entered.
This text can be modified upon import and is displayed in the description column of the export record.

Exported Objects
By expanding the container, the name, the application and the type of objects related to the exported process are displayed. Objects not meant to be exported can be unchecked.

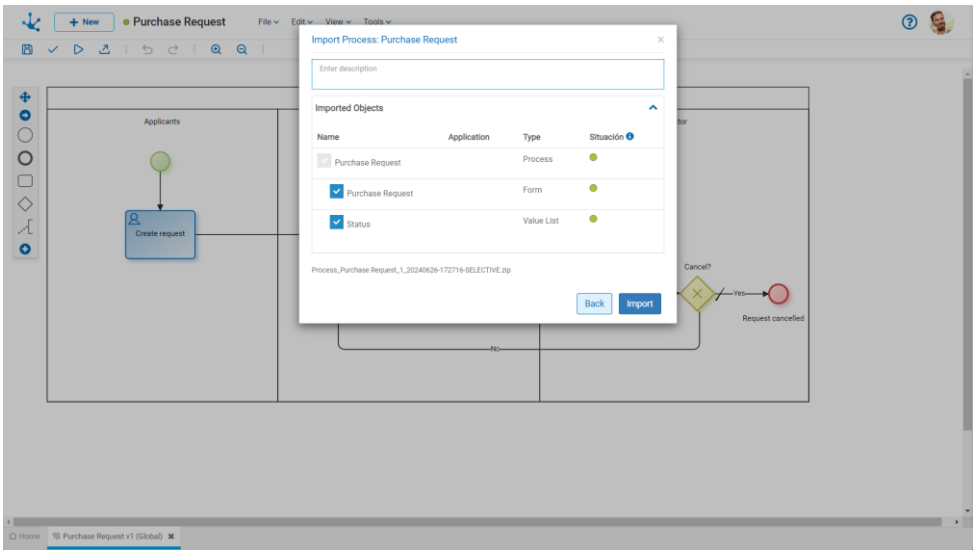
- Forms related to the process. If there are value lists associated with fields, their definitions are also included.
- Advanced rules used in automatic actions, in fields, from the "Relation" tab of the field properties or included in embedded rules.

> *Forms representing related entities are not included.*

Click on the "Cancel" button to undo export or click on the "Export" button to finish.

Import

It allows to open a window for the user to select and confirm the import of the object.



Validate

This icon allows to validate if the object is ready to be published, that is, the same validations are carried out as when publishing and the result of them is reported.

Publish

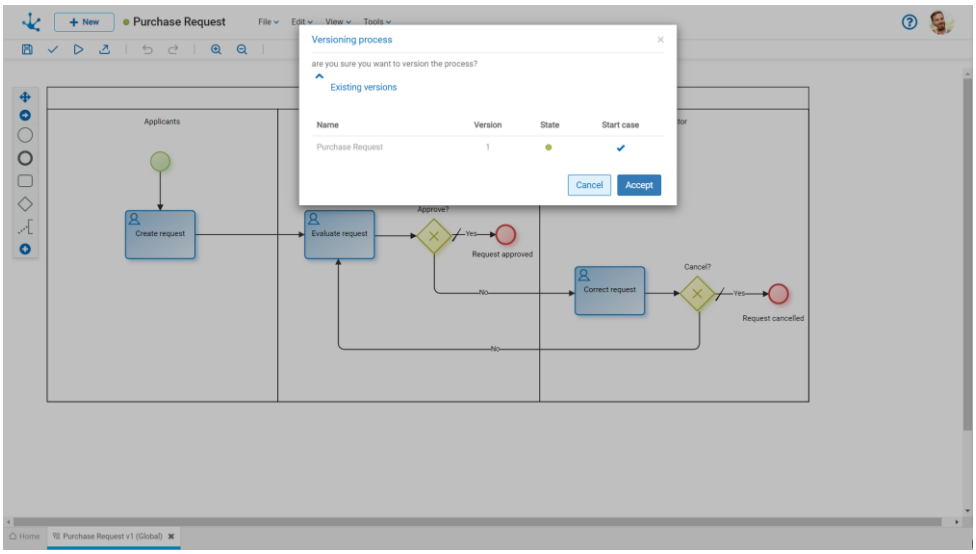
Through this icon the object changes to “Published” state and the modeler user receives the corresponding message, indicating the result of the operation. The condition for publishing is the same as for the "Save" operation.

Conditions

- The process must contain at least one lane with an activity and a participant.
- "Shipping" type tasks must contain valid values for their type.
- The maximum duration must be modeled to define alerts.
- If there are cases initiated, activities should not be deleted.
- Activities cannot contain more than one output flow.
- Output flows from inclusive and exclusive gateways must have a condition or button.
- In tasks, threads and processes the name is required.
- In “User” or “Undefined” type tasks, the "Execution" tab must be defined.
- In subprocesses, a subprocess must be selected.
- In the process, the modeler participant is required.
- In the lane, if the executing agent for a given activity is chosen, an activity must be defined for the participant.
- In the lane, if the agent based on form field is chosen, a form field must be defined for the participant.

Versioning

Allows the user to version the process, display the previous versions and show them. By hovering over each previous version, an "Open" button is made available to show that version.



Delete Data

All cases of the process and their attachments, if any, are deleted. If each case had an associated form instance, it would also be deleted. Additionally, BAM information, the cases and the activities related to the process-version are deleted.

Delete Publication

It allows removing the use process, which returns to the "Draft" state, in addition to deleting the process cases and the associated form instances, if any. In addition, BAM information, the cases and the activities related to the process-version are deleted.

Download Java Sources

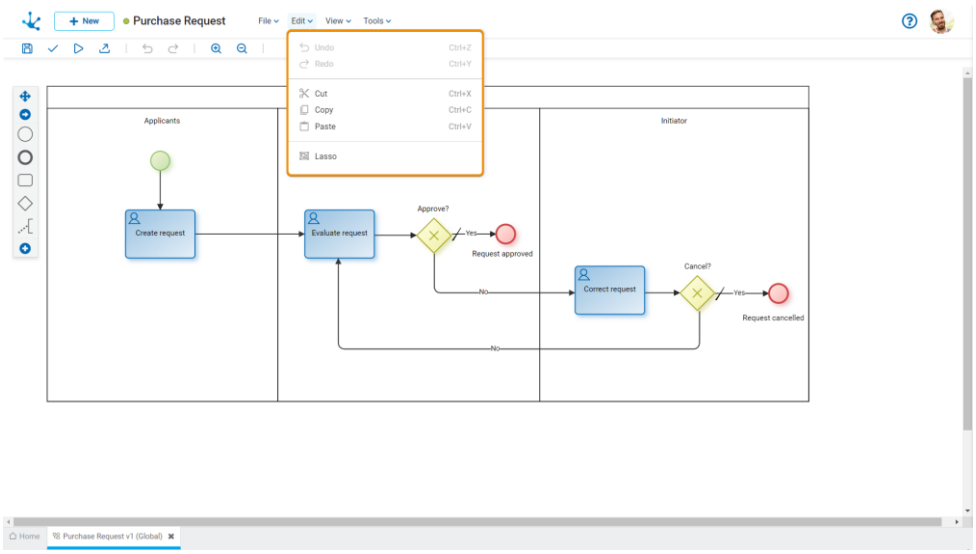
This icon allows to download the Java files that represent the object's model and service, so that it can be used in advanced rules.
Pressing the icon displays a message to confirm file download.

Publication History

This icon allows managing versions of published objects, displaying their history, to have better control of the changes that each object has.

3.4.10.1.1.2. Edit Submenu

This submenu is opened by clicking on the “Edit” option and allows the performance of operations within the modeler.



Undo (Ctrl+Z)

Allows to easily reverse the changes made in the modeler.

Redo (Ctrl+Y)

Allows to easily reapply the changes made in the modeler.

Cut (Ctrl+X)

Allows to send the selected element to the clipboard while deleting it from the graphic modeling area.

Copy (Ctrl+C)

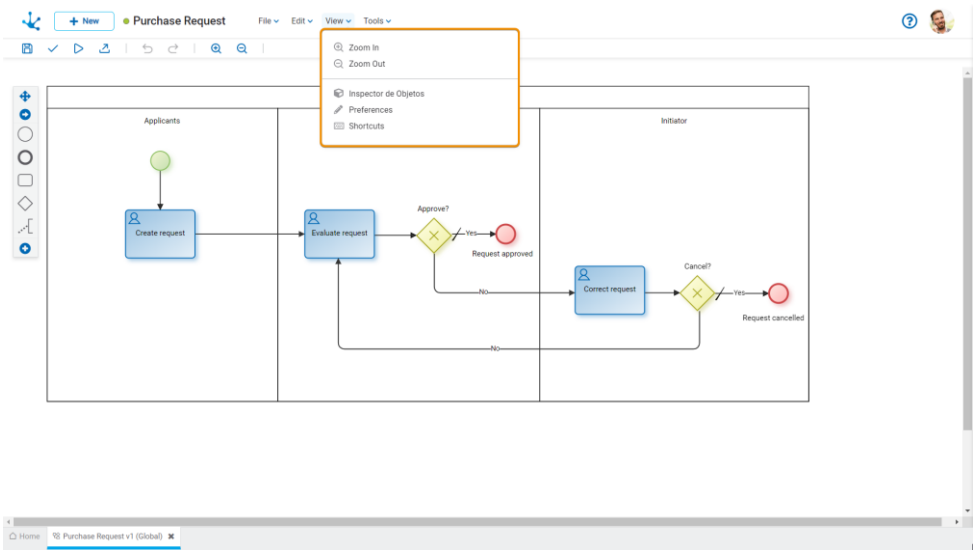
It allows to copy any application element and temporarily place it on the clipboard.

Paste (Ctrl+V)

It allows to take the item from the clipboard and place it elsewhere in the graphic modeling area.

3.4.10.1.1.3. View Submenu

This submenu opens by clicking on the "View" option and allows the selection of different ways to display the process.



Zoom In

Increases the display size of the diagram.

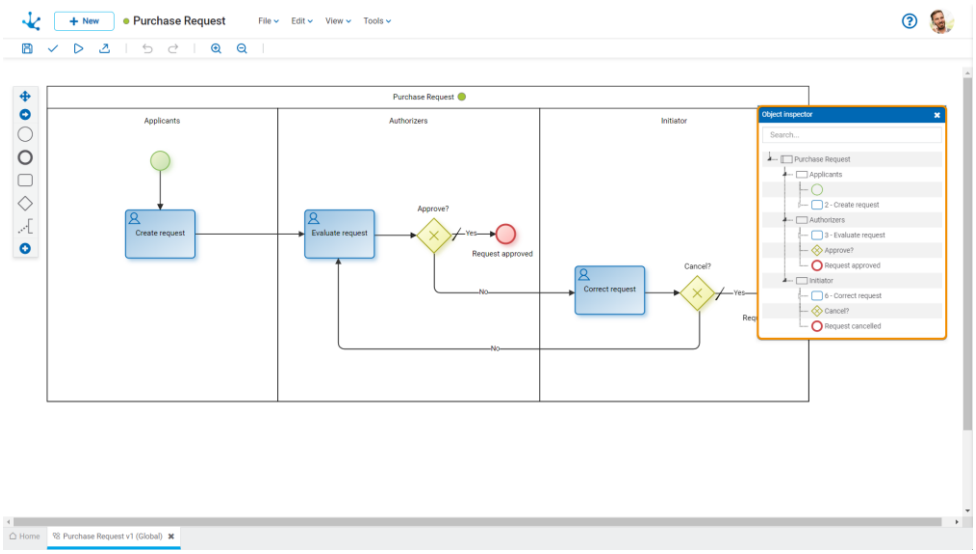
Zoom Out

Decreases the display size of the diagram.

Object Inspector

By clicking on this icon, a panel is displayed/hidden where all the elements of the process are shown in a tree structure.

This view facilitates the task of analysis and development, especially in complex diagrams.



At the top of the inspector there is a search field that allows searching for elements based on their name, identification code, behavior related to the execution of an activity, or defined automatic actions. Once a text is entered in the search field, the inspector highlights the process elements that match the entered text.

The properties panel for each element can be opened from the object inspector by double-clicking on its name.

Preferences

Defines the display preferences for processes modeling.

- Show symbol identifiers
  Indicates whether the internal identifiers of symbols are shown in the diagram. The default value is "No".

- Grid size
  Define the number of pixels used when moving symbols in the diagram while dragging them with the mouse. The default value is “1”, although “10” or “20” can be selected for larger movements of the symbols with the mouse.

- View minimap
  Indicates whether the thumbnail view is displayed in the lower right corner. This view consists of a small panel where the complete process is displayed.

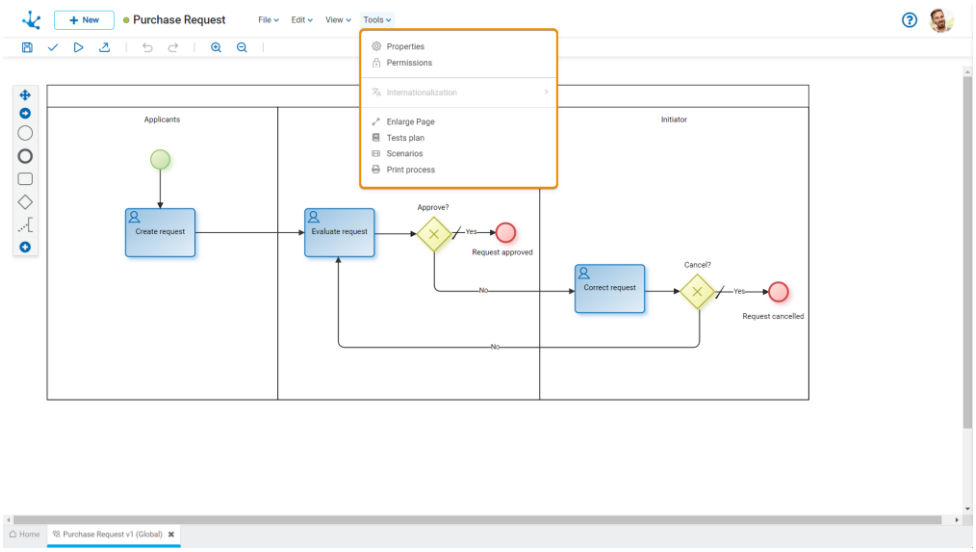
- Display validations on shapes
  Indicates whether the exclamation mark is displayed on shapes with validation errors.

Shortcuts

Opens a panel with all available keyboard shortcuts to use in the processes modeling.

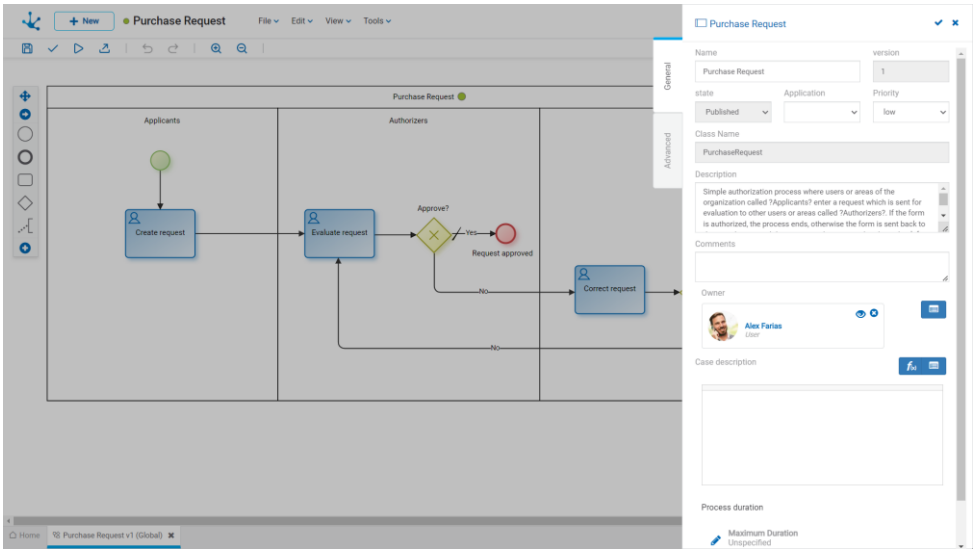
3.4.10.1.1.4. Tools Submenu

This submenu opens by clicking on the “Tools” option and allows modeling properties and permissions of the process, as well as internationalization and various modeling functionalities.



Properties

Opens the panel of properties of the process.



Permissions

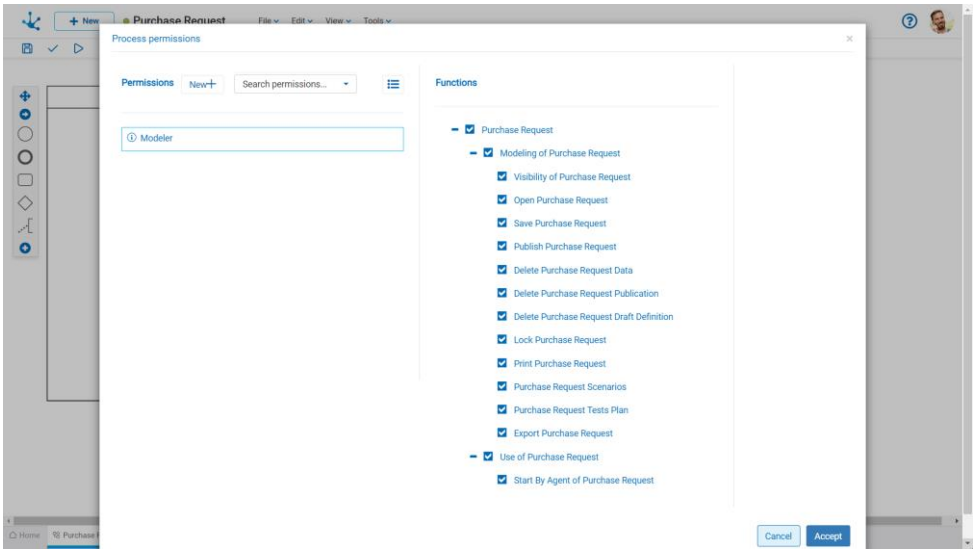
Allows to assign the security functions for use and modeling of the object to the existing permissions or to new permissions that the user creates, without having to go to the permission settings option from the menu.

Sections

- Permissions: Permissions to which object functions are assigned.
- Security Functions: Represents the total set of security functions, modeling and use of the object. Those that are marked are the ones included in the selected permission.

By default, all security functions for a new object are assigned to the permission Modeler.

Users who are assigned the permissions have access to the functions included in it.



Modeling Security Functions

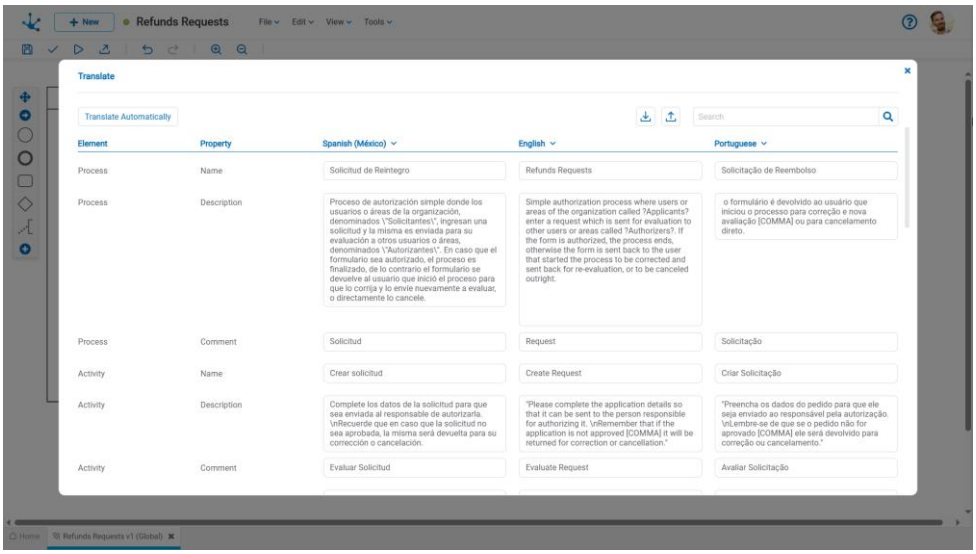
- Visibility: Allows to display the object in the Deyel modeler grid.
- Open: Allows to open the object from the Deyel modeler.
- Save: Enables the operation of saving modifications made to the object.
- Publish: Enables the operation of publishing the object leaving its state as "Published".
- Delete data: Enables the delete data operation.
- Delete publication: Enables the operation of deleting the object publication leaving its state as "Draft".
- Delete draft definition: Enables the operation of deleting the object.
- Lock/unlock: Enables the lock/unlock operation, only the user who locks it can modify it.
- Print: Enables the process print operation.
- Scenarios: Enables the operation of creating different testing scenarios.
- Test plan: Enables the operation of creating a test plan and adding scenarios to it.
- Export: Enables the operation to export the object from the tools submenu of the expanded menu.

Use Security Functions

- Start by agent: It allows the start of a process if "Initiator Restricted by Function" is defined in the first lane as a participant.

Internationalization

The "Internationalization" option is enabled when the property Internationalize is activated. To access the translation panel, it is necessary to select "Internationalization" and then choose "Translate".



In this panel, all the translatable properties of the object are displayed, along with their content in the original language, and translations in the other available languages can be entered manually.

If a value is modified in the original language column, the change is applied directly to the object's property.

In the header of each language column, a menu that allows selecting another language can be displayed, updating the column with its contents.

Options Available in the Top Margin

Translate Automatically Generates translations from the original language to other languages using artificial intelligence.

Downloads an Excel file with the contents of the translation panel.

Allows selecting an Excel file to import translations, making it easier to manage large volumes of content.

Filters the contents of columns based on the characters entered in the search field.

Enlarge Sheet

Increases the modeling area, which makes scroll bars to be displayed and the area of the diagram that is visible in the window is shaded in the thumbnail view. As scroll bars move, the shaded area scrolls in the thumbnail view.

Tests Plan

In a new tab, it opens a panel with the grid corresponding to the test plans defined for the process. Operation buttons are available for each grid row. This option is enabled only after saving the process.

Print Processes

Allows to print the process layout. It opens a new tab with the print functionality for the process. This option is enabled only after saving the process.

3.4.10.1.2. Predefined Templates

Deyel provides a collection of standard process templates applicable to any organization. The modeler user of **Deyel** can select the template that best suits their needs to start modeling a new process immediately.

Besides, modifications deemed necessary to the process generated from the selected template can be added, as well as the definition of participants of each activity and the conditions that determine the data flow and its visibility.

According to the characteristics of the process they model, templates can be classified into groups.

Approval Process Templates

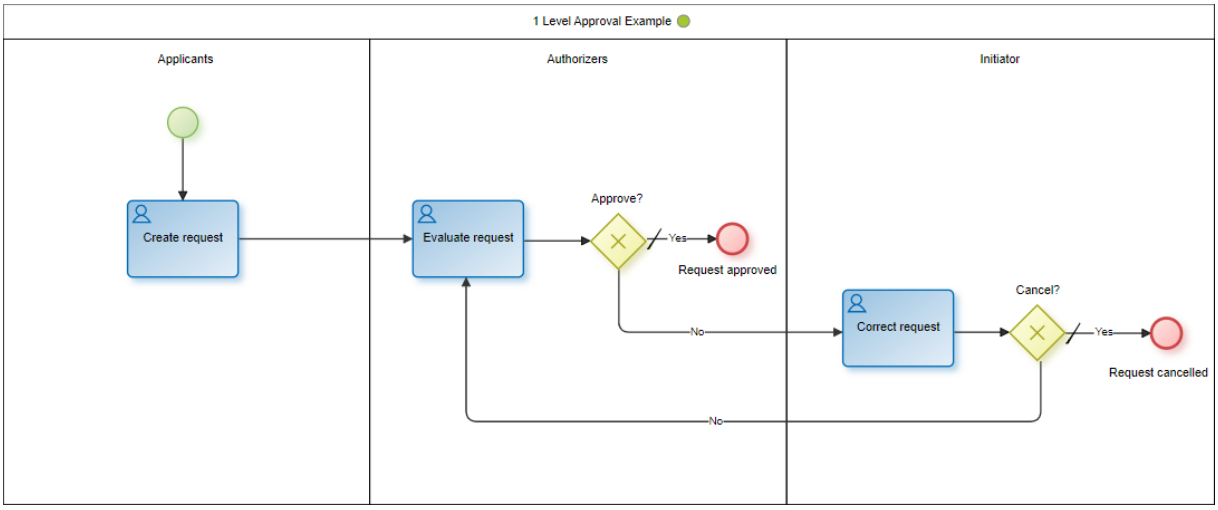
They model conditional branching situations, such as requests that are approved or sent for correction by the requestor.
These templates are suitable for defining vacation request processes, expense reimbursement requests or absence from work notice, among others.

When executing the process, business users display the buttons corresponding to the available actions in each activity, such as “Approve”, “Correct”, “Deny” or “Cancel”.

1 Level Approval

This template is used to implement processes that consider a single approval activity.

It allows users or organization areas, called “Applicants”, to enter a request and send it for evaluation to other users or areas called “Authorizers”. If the form is approved, the process ends, otherwise the form is sent back to the person who started the process to correct it and send it again for re-evaluation, or to cancel it outright.

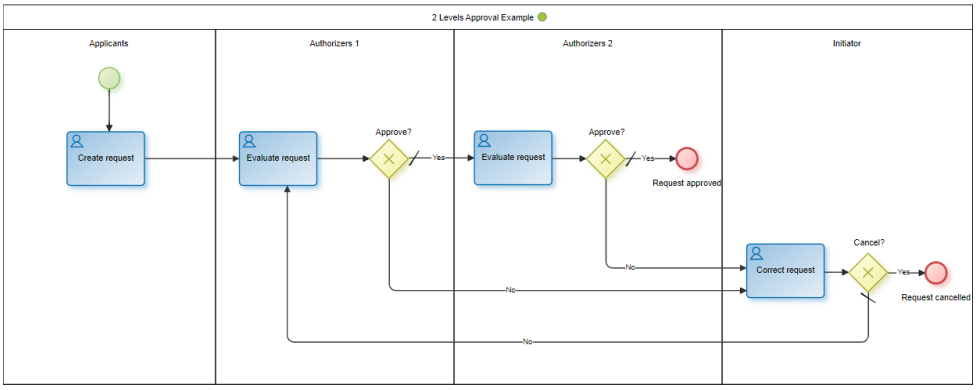


2 Levels Approval

This template is used to define processes with 2 sequential approval activities.

Adds an approval level to the template explained in "1 Level Approval ", that is, after the application is approved by the "Authorizers 1" level, it goes to the "Authorizers 2" level for evaluation. If the form is not approved in any of the authorization levels, it is sent back to the person who started the process to correct it and send it again for re-evaluation, or to cancel it outright.

These templates can be used to define authorization processes in organizations with different responsible persons for each approval level.



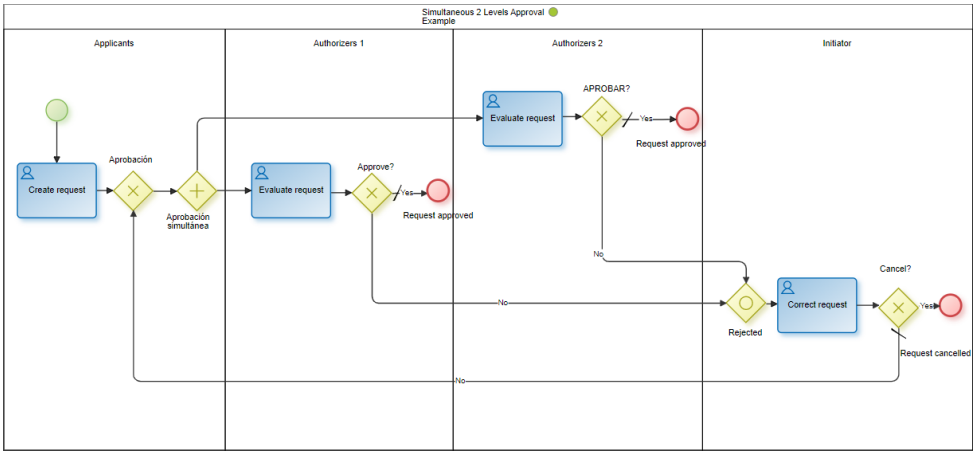
Simultaneous 2 Levels Approval

This template is used to define processes with 2 parallel approval activities.

It has two levels of approval as well as the template explained in “2 Levels Approval”. Once the application is created it must be evaluated by the "Authorizers 1" level and the "Authorizers 2" level for approval, regardless of the order in which it is authorized. The case is finished only after having both

approvals. If the form is not approved in any of the authorization levels, it is sent back to the person who started the process to correct it and send it again for re-evaluation, or to cancel it outright.

These templates can be used to define authorization processes in organizations with different responsible persons for each approval level and where the approval sequence is not relevant .



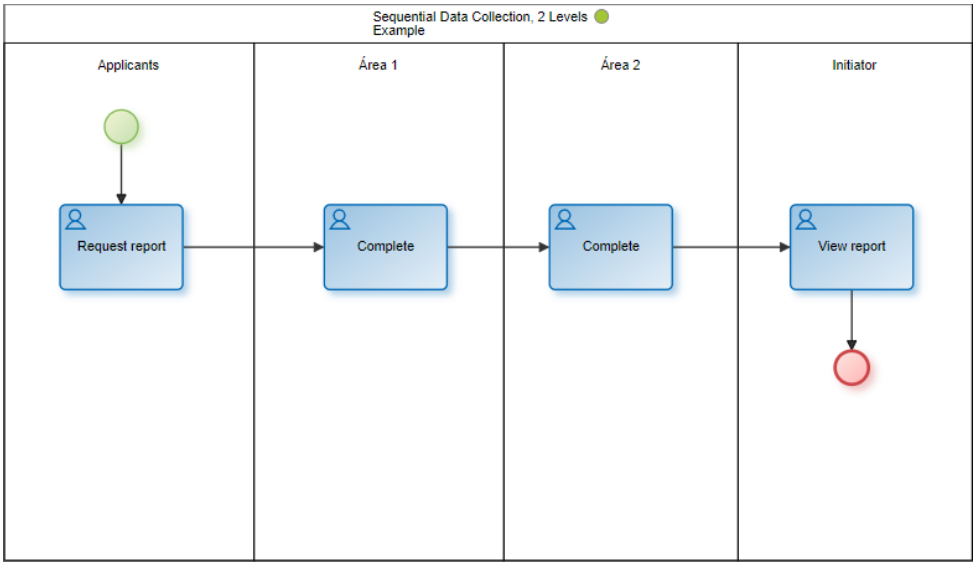
Data Collection Process Templates

They model processes where different business or office users, or the same but at different times, update the information of a case.

These templates can be used to define processes for hiring requirements, password generation, quote requirements, content publication, among others.

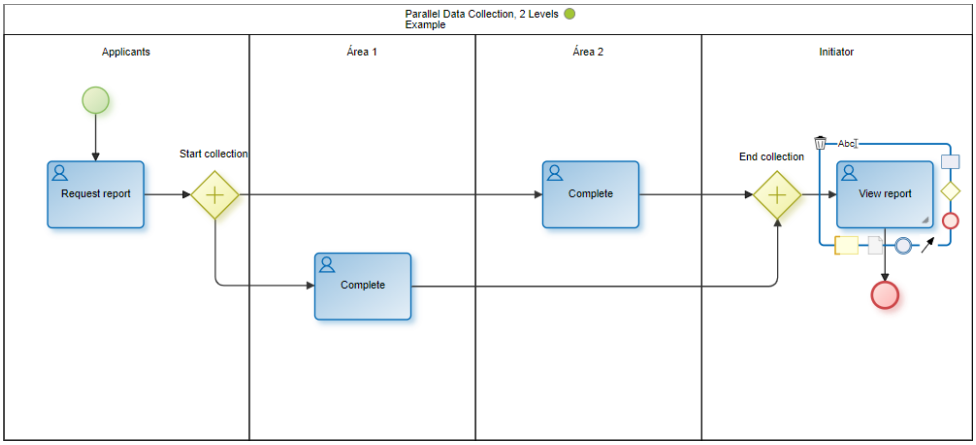
Sequential Data Collection, 2 Levels

This template is used to define processes that collect data in 2 different stages. It can involve more than one business or office user and the request is ultimately forwarded to whoever started the process.



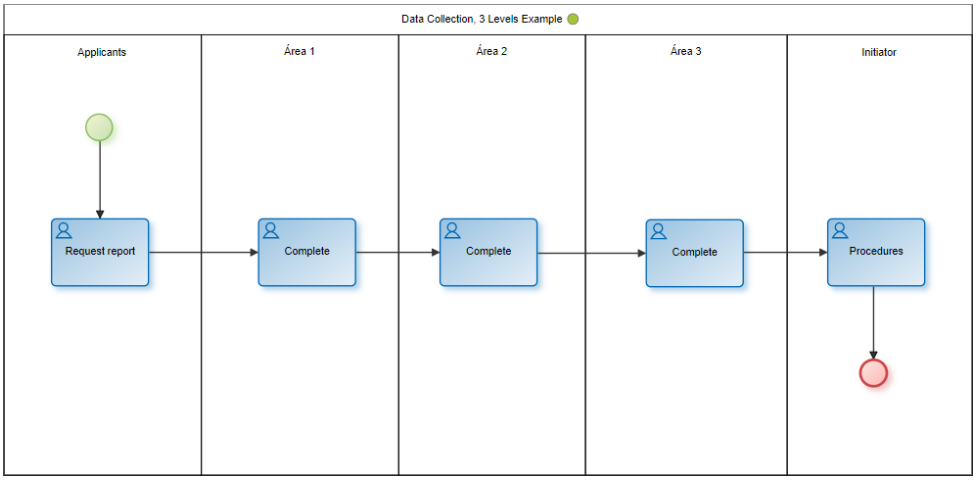
Parallel Data Collection, 2 Levels

This template is used to define processes that collect data in 2 different areas, regardless of the order in which they are performed. It can involve more than one business or office user and the request is ultimately forwarded to whoever started the process.



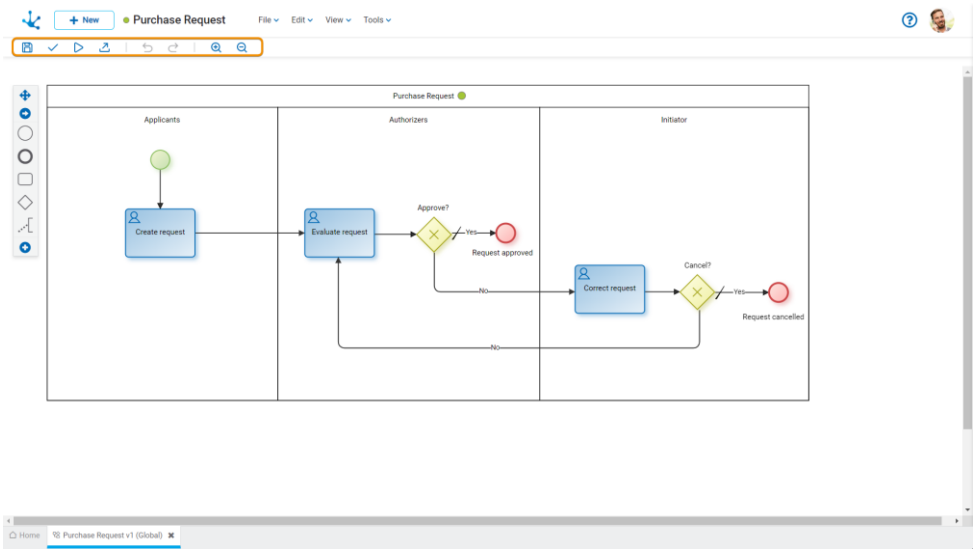
Data Collection, 3 Levels

This template is used to define processes that collect data in 3 different stages. It can involve more than one business or office user and the request is ultimately forwarded to whoever started the process.



3.4.10.1.3. Top Toolbar

Contains icons for the quick access to the most used operations from the expanded menu.



File

- Save
- Validate
- Publish
- Export

Edit

- Undo
- Redo

View

Zoom In
Zoom Out

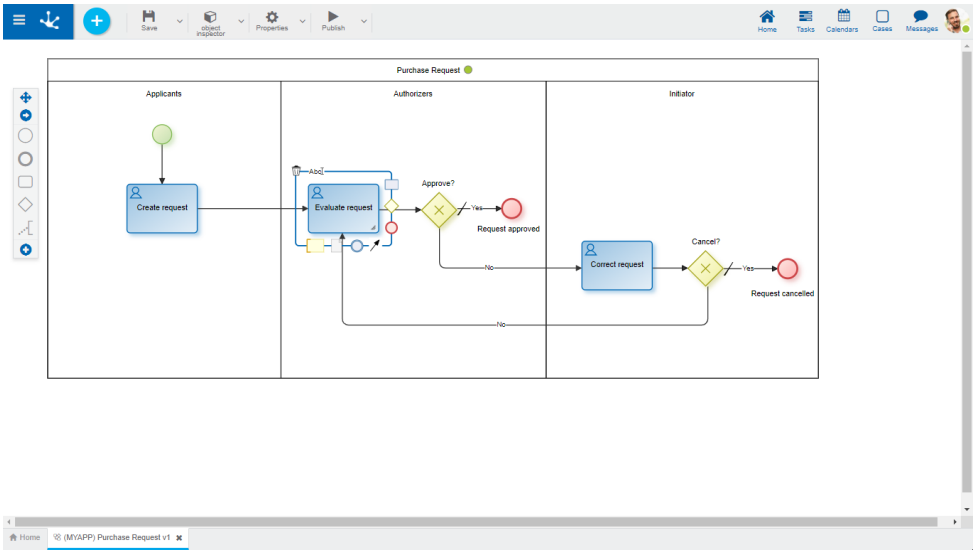
3.4.10.1.4. Graphic Modeling Area

Processes Modeling > Graphic Modeling Area

Graphic modeling area is where processes are designed by using graphic elements from **BPMN 2.0** (Business Process Model and Notation)

Pie Menu

Circular menu which is displayed by making click on each modeled graphic element in the design area, showing the different symbols with which the selected object can connect to.



To incorporate a new connected symbol, pie menu of the graphic element selected must be opened, drag and drop the symbol to be connected on the modeling area.

Common Options

Allows to delete the symbol

Allows to edit the name of the symbol for its modification

Context Menu

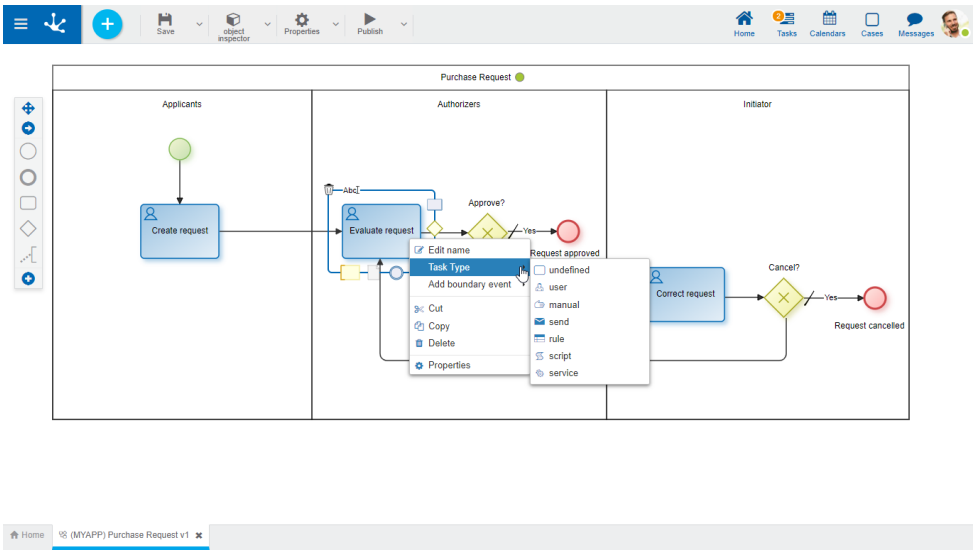
On clicking on the selected graphic element with the right mouse button, it opens a vertical menu with its options grouped as own of the object and common to all objects, besides from an option to open the properties.

Options

- Own of the Object
  - Edit name
  - Available types according to the object
  - Other own actions of each object

- Common to All Objects
  - Cut
  - Copy
  - Delete

- Properties
  Opens the properties panel of the graphic element, it is the same panel that is displayed by double clicking on the symbol.



Thumbnail View

Complete image of the process which is visualized on the lower right corner of the graphic modeling area, the active work area is reflected in such image. It is visualized whenever you activate the option Thumbnail View of the option Preferences in the top toolbar.

Scrolling in the Modeling Area

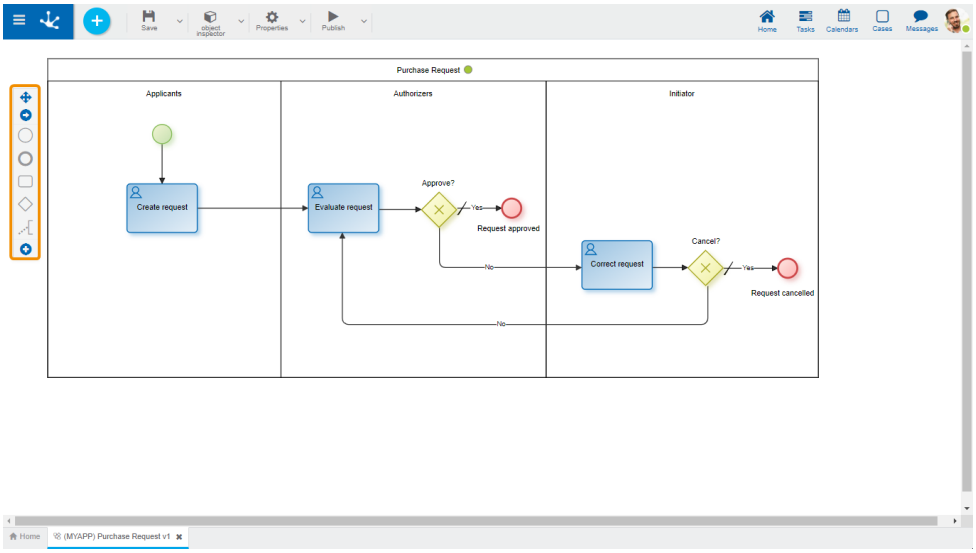
The vertical and horizontal scroll bars are used to slide the graphic modeling area, in order to allow the complete edition of the process. They are automatically activated when the diagram exceeds the size of the work window.

The modeling area can be enlarged by using the option Enlarge Page on the top toolbar.

3.4.10.1.5. Side Toolbar

Process Modeling > Graphic modeling area

It is the toolbar that contains the graphic elements corresponding to the different types of symbols that can be defined in the process diagram, based on **BPMN2.0** (Business Process Model and Notation).



Bar Elements

Scroll Bar

Allows to move the icon bar to any position in the graphic modeling area.

/ Show/Hide Names

Either displays a panel to the right of the bar with the names of icons, or closes it.

Start Event

A start event Indicates the beginning of a process, so they have no input flow but may have a trigger event.

Types

- Standard
- File
- Command
- Rule
- Email
- Signal
- Timer

 End Event

An end event Indicates the end of a process, so it has no output flows.

Types

- Standard
- Signal
- Terminal

 Task

A task represents a work unit that is carried out as part of a process execution.

 Gateway

An gateway represents a branch point. Gateways can be of different types.

- Exclusive without marker
- Exclusive with marker
- Inclusive
- Parallel

 Annotation

An annotation allows the process modeler to add additional data, intended for the diagram reader.

 /  Show/Hide more Shapes

Allows to display the sidebar in full or reduced form.

 Subprocess

A subprocess is an activity that refers to another independently defined process.

 Intermediate Event

An intermediate event indicates where events may occur between the beginning and the end of a process. They affect the process flow, but they will not start nor end a process directly.

Types

- Capture link
- Catch signal
- Throw link
- Throw signal
- Timer

Data Object

A data object represents information that flows through the process such as documents or emails.

Group

A group allows to group symbols visually.

3.4.10.2. Graphic Elements

The graphical elements of **Deyel** process modeler are based on **BPMN 2.0** (Business Process Model and Notation).

BPMN 2.0 is a standardized graphical notation that allows to model business processes in a workflow format. It provides an easy and understandable notation for all those involved in the business, whether they are business analysts (who define the processes), technical developers (responsible for implementing them) or business managers (who monitor and manage them).

BPMN 2.0 is intended to be the common language to reduce the communication gap that frequently occurs between the business process design and its implementation.

To model processes, elements must be dragged from the left palette to the selected position in the graphic modeling area and the properties of each element must be entered.

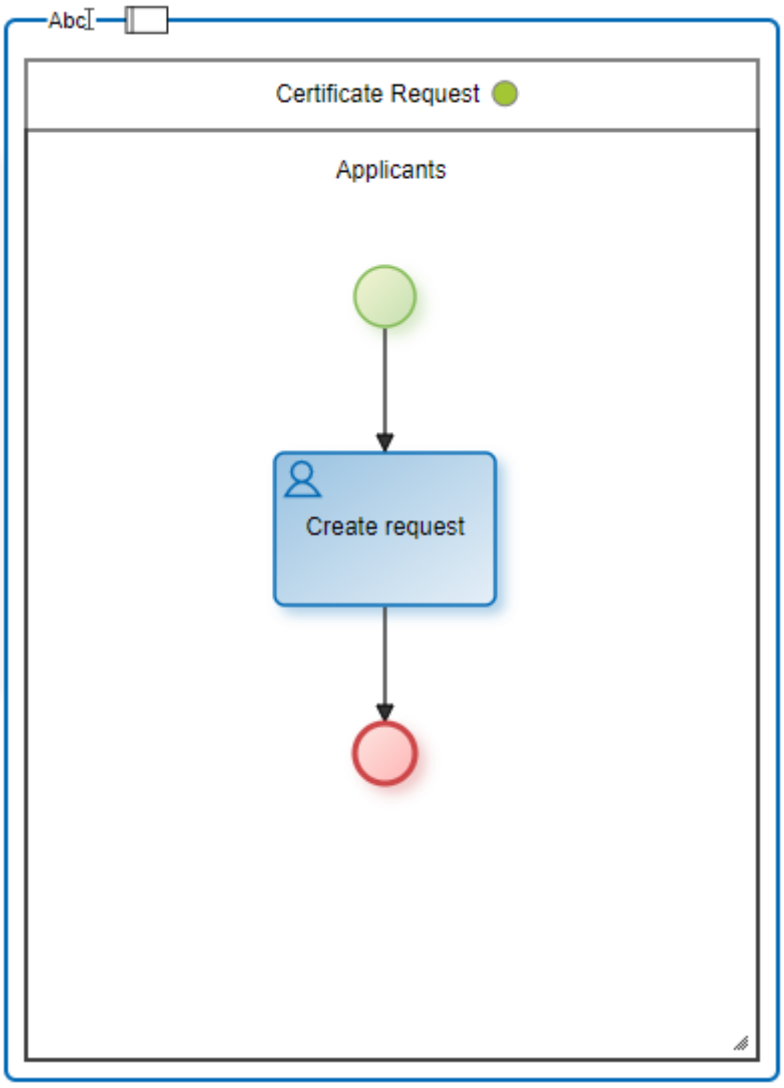
- Pool
- Lane
- Activity
- Flow
- Event
- Gateway
- Artifact

3.4.10.2.1. Pool

The pool is a graphical representation of a business process (Business Process).

"It is a set of one or more activities whose performance allows meeting a business objective, normally within an organizational structure context, which defines functional roles and relationships among them." [Workflow Management Coalition]

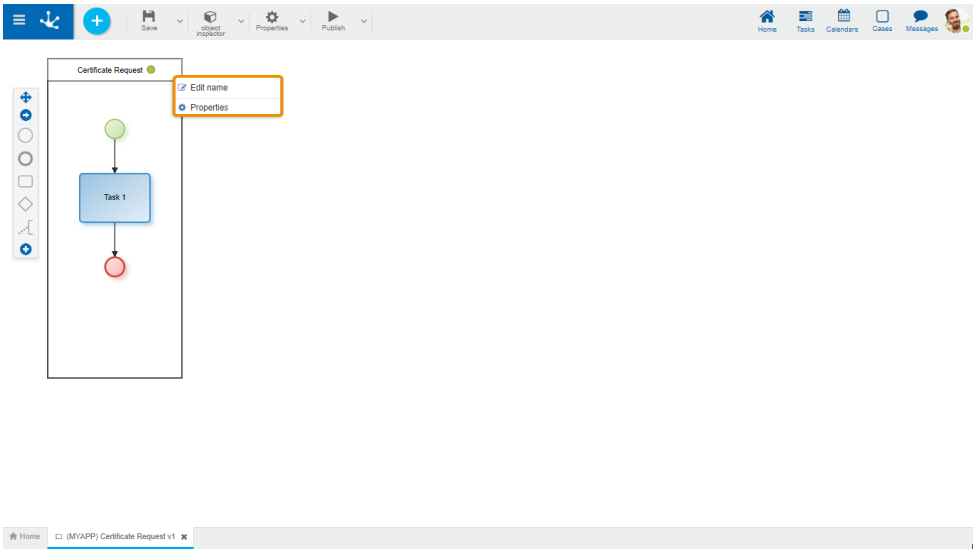
Pie Chart Menu



Abc	Opens an area for editing the name.
	Adds a lane.

Clicking on this icon adds a lane to the right inside the pool.
Every pool must have at least one lane. When you create a process without a template, a lane is automatically generated.

Context Menu



- Edit name: Allows to edit the name. Same functionality as the icon Abc
- Properties: Opens the properties panel to show and/or modify.

3.4.10.2.2. Lane

Lanes allow organizing the activities of a process. Each lane represents one participant.

"The lane is a partition that is used to organize and categorize activities within a pool. It is generally used to represent internal roles (for example managers), systems (e.g. business application) or internal departments (e.g. finance)." [Workflow Management Coalition]

Types of Participants

User
The activities included in the lane are executed by a specific user or by their delegated users, if any.

Organizational Unit
The activities included in the lane are executed by every user that belongs to the organizational unit informed.

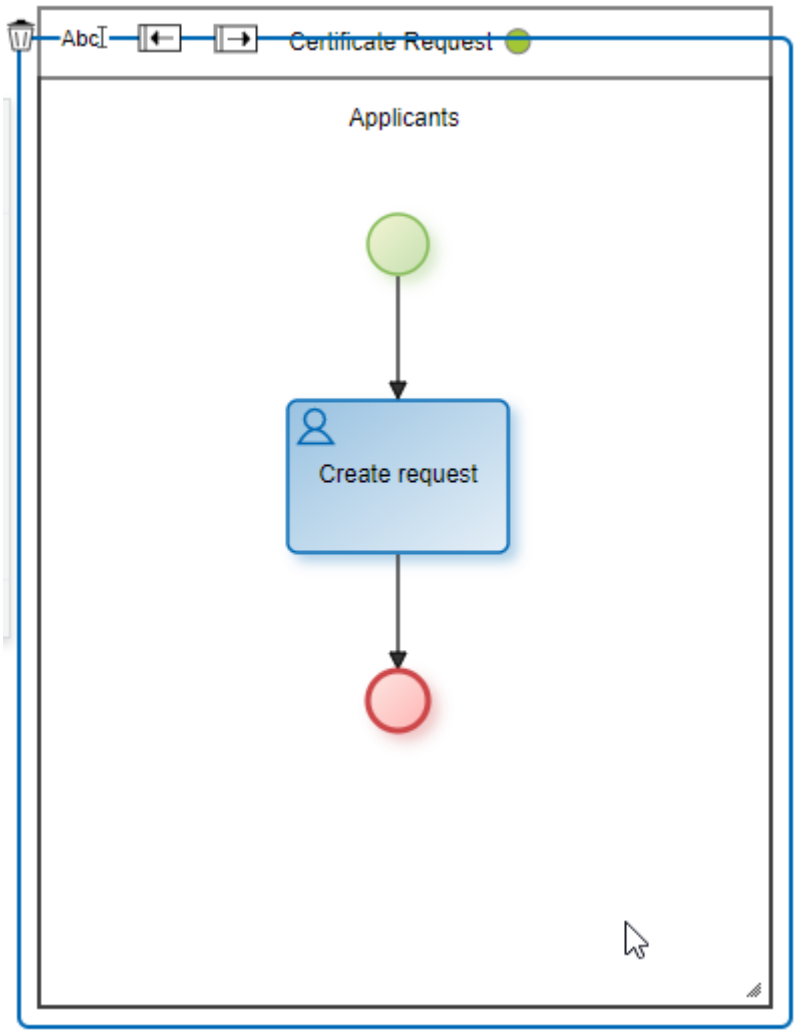
Role
The activities included in the lane are executed by the actors defined within the role.

Agent
The activities included in the lane are executed by a participant who is dynamically identified when the activity is executed.

A predefined agent can be assigned as lane manager or else, define a new agent.

Lanes without Participants
If at the time of modeling a process, the persons responsible for each activity are not known, the lane may be left without an assigned participant as long as the process state is "Draft" or "Modified".

Pie Chart Menu



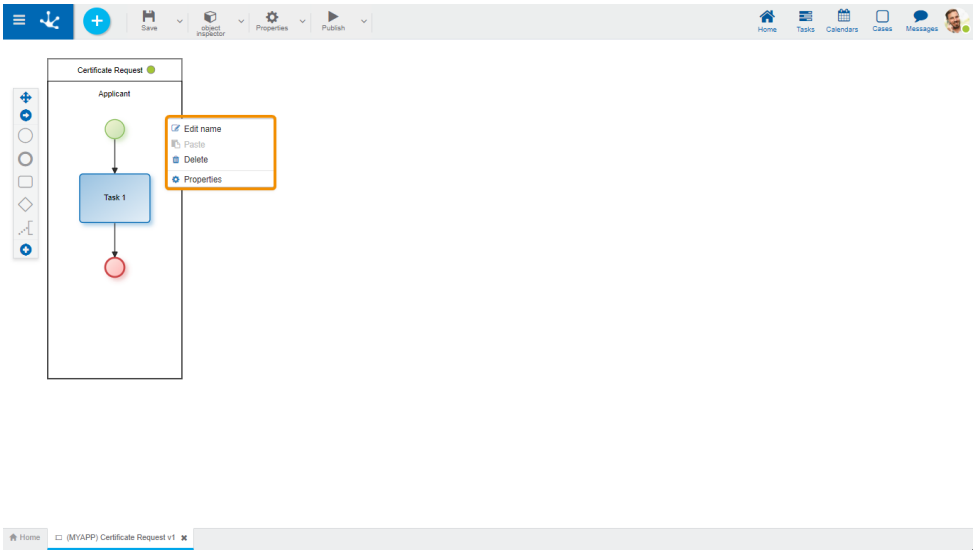
Abc Opens an area for editing the name.

Deletes the lane.

Add a lane to the left.

Add a lane to the right.

Context Menu



- Edit name: Allows to edit the name. Same functionality as the icon AbcI.
- Paste: Allows to paste a previously copied graphic element into the lane.
- Delete: Allows to delete the lane. Same functionality as the icon .
- Properties: Opens the properties panel to show and/or modify.

Lane extension
When more space is required to incorporate elements into the lane, either horizontally or vertically, click on the icon and drag the cursor until the required lane size is achieved.

3.4.10.2.3. Activity

An activity can be understood as the representation of a work unit that is carried out as part of a process execution.

"An activity is the description of a task that is a logical step within a process." [Workflow Management Coalition]



The graphic representation of this element is a rectangular image with rounded corners that contains a label with the name of the task inside it, identifying the types of activities with different icons inside.

Types

- Tasks
- Subprocess

3.4.10.2.3.1. Task

A standard activity or task is an indivisible work unit performed as part of the execution of a process.

Types

The **BPMN 2.0** notation (Business Process Model and Notation) allows to typify standard activities or tasks in such a way that the graphic element used indicates the task type it represents.
The task type definition is made in different ways:

- From the task context menu, selecting the option "Task type".
- From the task property panel.



Undefined Task
Task without type.



User Task
Task that needs human intervention to be performed. When an activity modeled as a user task must be executed, it is informed to the participants of the process in My Tasks, in the top toolbar It is recommended to use this notation to model all the activities in which the user has to create, modify or show a form.



Manual Task
Represents a task to be performed by the user without any application support. Only moves on to the next activity.
It is recommended to use this notation to model activities such as making a telephone call to the customer, paying a visit or installing audio equipment, among others.



Send Task
It is used to send a message to an external participant regarding the process. When the message has been sent, the task ends.
This task type is used to model automatic mailing tasks. They do not define user participation.



Business Rule Task
Represents a call to an advanced rule, being able to exchange parameters between the task and the rule. They do not define user participation.

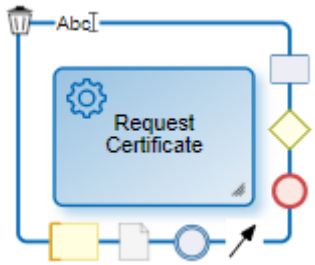


Script Task
This task type only performs automatic actions. They do not define user participation.



Service Task
Represents the execution of a service, automatically and without user intervention, which is carried out through the adapters. This notation is recommended to model the use of a web service, an automated software component or an integration rule. They do not define user participation.

Pie Chart Menu



Opens an area to edit the name.

Deletes the task.

Adds an activity.

Adds a gateway.

Adds an end event.

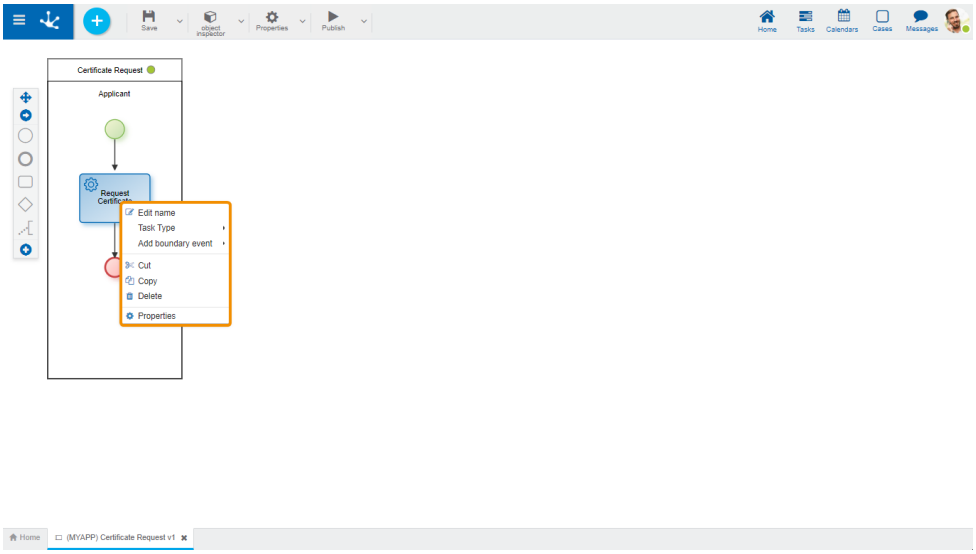
Adds a flow.

Adds an intermediate event.

Adds a data object.

Adds a comment.

Context Menu



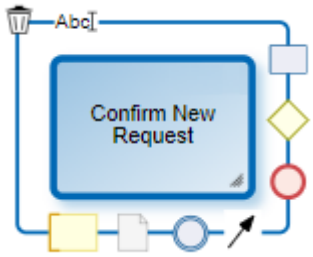
- Edit name: Edits the name inside the rectangle that represents the activity. Same functionality as the icon Abc.
- Task type: Defines the type of task.
- Add border event: Incorporates an border event to the task.
- Cut: Deletes the selected task. It can be pasted in another process location.
- Copy: Copies the selected task. It can be pasted in another process location.
- Delete: Deletes the selected task.
- Properties: Opens the property panel to show and/or modify.

3.4.10.2.3.2. Subprocess

It is an abstract activity that represents an independently defined process. It is used to hide levels of detail in processes and allows to reuse modeled processes. For example, if different processes have a common part, it can be defined only once and be reused in each process.

Once the execution of the subprocess is complete, control returns to the process that uses it and continues with the next modeled activity.

Pie Chart Menu



Abc Opens an area for editing the name.

Deletes the subprocess.

Adds an activity.

Adds a gateway.

Adds an end event.

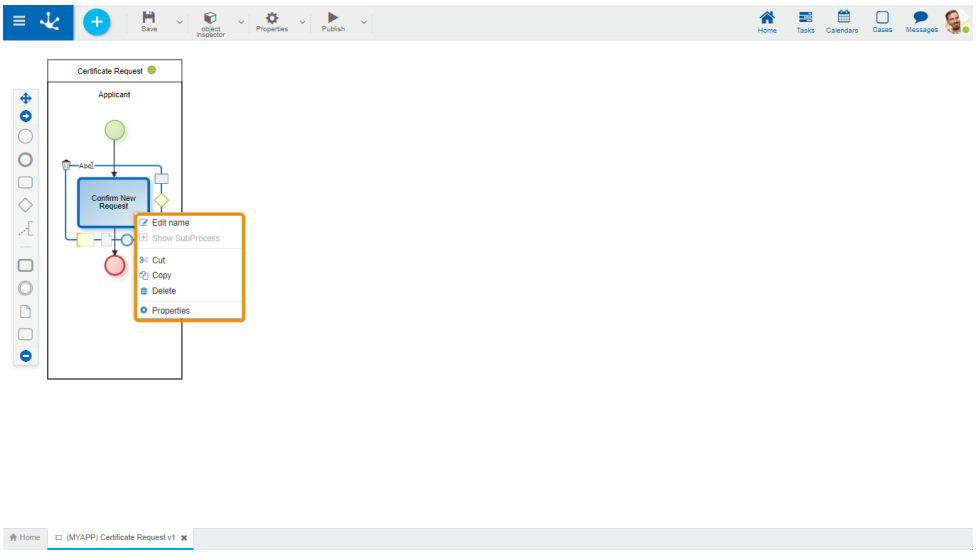
Adds a flow.

Adds an intermediate event.

Adds a data object.

Adds a comment.

Context Menu



- Edit Name: Edits the name inside the rectangle that represents the activity. Same functionality as the icon AbcI.
- Show Subprocess: Opens the subprocess in a new modeler tab.
- Cut: Deletes the selected subprocess. It can be pasted in another location.
- Copy: Copies the selected subprocess. It can be pasted in another location.
- Delete: Deletes the selected subprocess.
- Properties: Opens the properties panel to show and/or to update.

3.4.10.2.4. Flow

A flow represents the connection between an activity and its subsequent activities. Flows are used to establish the sequence in which defined activities are executed.

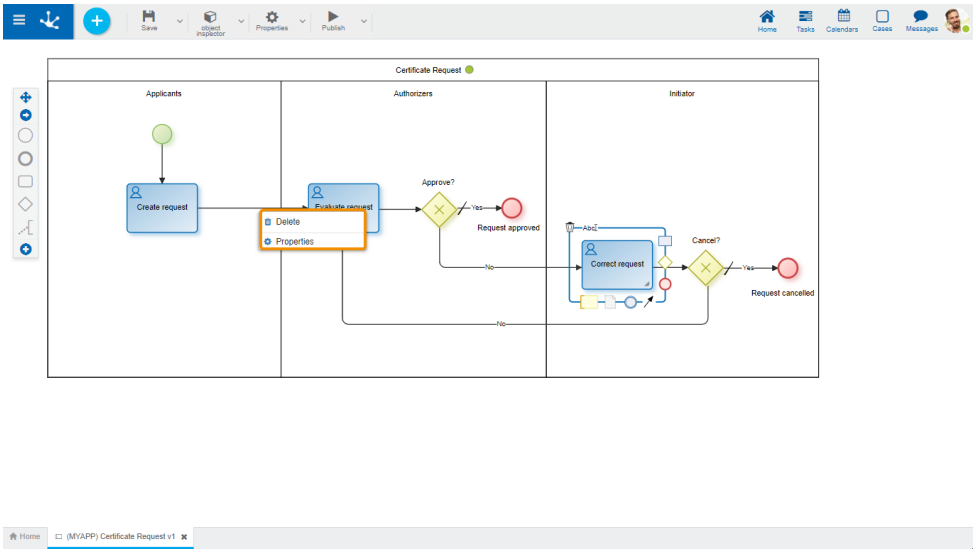
"Sequence flows are used to show the order in which activities are carried out within a process." [Workflow Management Coalition]

Normal flow

A flow can determine a required path, that is, the process has no option not to execute it nor an optional path, in which case, the subsequent activity may or may not be executed.

Default Flow
The exclusive and inclusive gateways must have a defined predetermined flow so that the process can execute its path in case none of the conditions for the output flows of the aforementioned gateways are met.

Context Menu



- Delete: Allows to eliminate the flow.
- Properties: Opens the properties panel to show and/or modify.

3.4.10.2.5. Event

An event is an fact or incident that occurs during the execution of a business process. In general, they require an action and allow a reaction.
Represents situations such as start of activities, end of activities, document change of state, message arrivals, among others.

"An event is something that happens during the course of a process. They generally have a cause and an impact." [Workflow Management Coalition]

Start events
They indicate the way in which a process is started. They have no input flow.



Standard
It does not have a trigger event, therefore it indicates that the process does not start automatically.



Command
Indicates that the process starts by executing a command from a chat window of the instant messaging of **Deyel**.
These events are used to model processes that are executed from a chat window, by means

of actionable messages if data entry or decision making is required.



Email
Indicates that the process starts when an email arrives to a certain email account, which may come from another participant, another process or other application. They are detected by scheduled tasks called EMAIL-type Event Generators.



Timer
Indicates that the process starts after a certain time cycle or on a specific date.



File
Indicates that the process starts when a new file is detected in a certain folder.



Rule
Indicates that the process starts from the execution of a certain business rule.



Signal
Indicates that the process starts each time it catches the indicated signal. The current process may not be the only receiver of the transmitted signal.

End events
Indicate the completion of the process. They have no outflow.



Standard
Indicates the completion of the process without adding any behavior.



Signal
Indicates the end of a process and sends a signal to other processes that are waiting to catch it.



Terminal
Indicates that all process activities are immediately finished when the process comes to an end. When any of the paths reaches this event, the entire process is finished.

Intermediate events
Indicate possible event occurrences that take place between the beginning and the end of a process.



Throw Link and Catch Link
These events facilitate the use of diagrams. They represent the connection of two sections of the same process distantly located in the process diagram.



[Throw Signal]
Used to send a signal that is received by signal catching events, whether they are initial, intermediate or final indistinctly, of the current process or of another process.



[Catch Signal]
Indicates that the process is waiting for the reception of a signal sent by the same process or another, in order to advance.

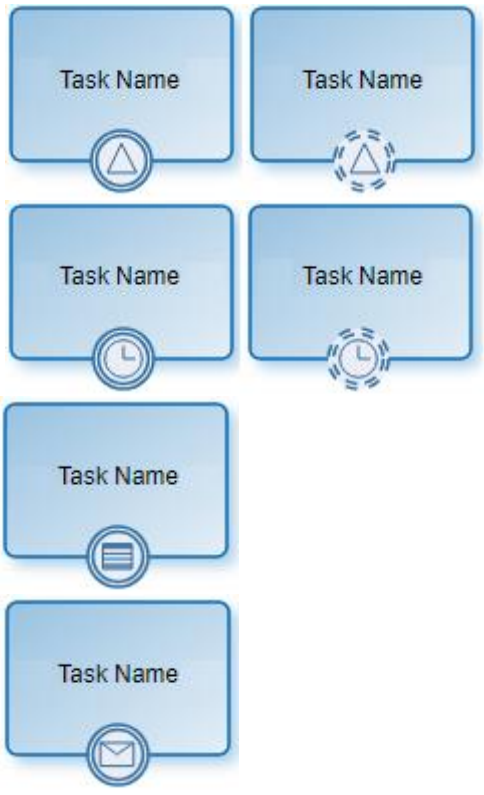


[Timer]
Acts as a mechanism that waits until a certain date or during a specified cycle.

Border event

They are intermediate events that are directly associated with an activity.
These events can be triggered only when the activity to which they are associated is being executed.
The output flows of border events are called exception flows, as they are executed only if the border event is activated.
They are used in particular for managing exceptions and waiting periods with certain deadlines.

- Interrupting events
  When the border event is activated, the execution of the current activity is interrupted and the process continues through the output flow of the activated border event.
  They are represented by a solid line.
- Non-interrupting events
  When the border event is activated, the process starts a path to treat the exception, in parallel with the normal process.
  They are represented by a dashed line.

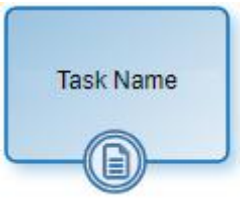


[Signal]
This event is waiting for the reception of a signal that can be emitted by this or another process.

[Timer]
This event is activated at a specified moment or at certain time intervals.

[Rule]
These type of events represent the execution of a business rule. They are detected by scheduled tasks called "Component-type Event Generators".

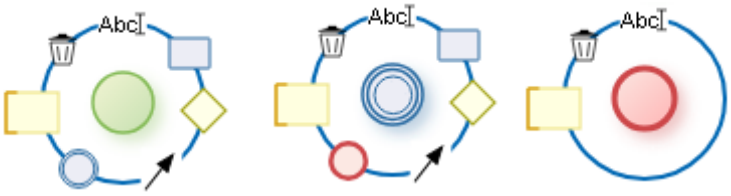
[Email]
These type of events represent the arrival of an email to a certain email account. They are detected by scheduled tasks called [EMAIL-type Event Generators].



File
These type of events represent the appearance of a new file in a certain folder. They are detected by scheduled tasks called "FILE-type Event Generators".

Pie Chart Menu

The options presented in the pie chart menu vary depending on whether it is a start, intermediate or end event.



Abc Opens an area to edit the name.

Deletes the event.

Adds an activity.

Adds a gateway.

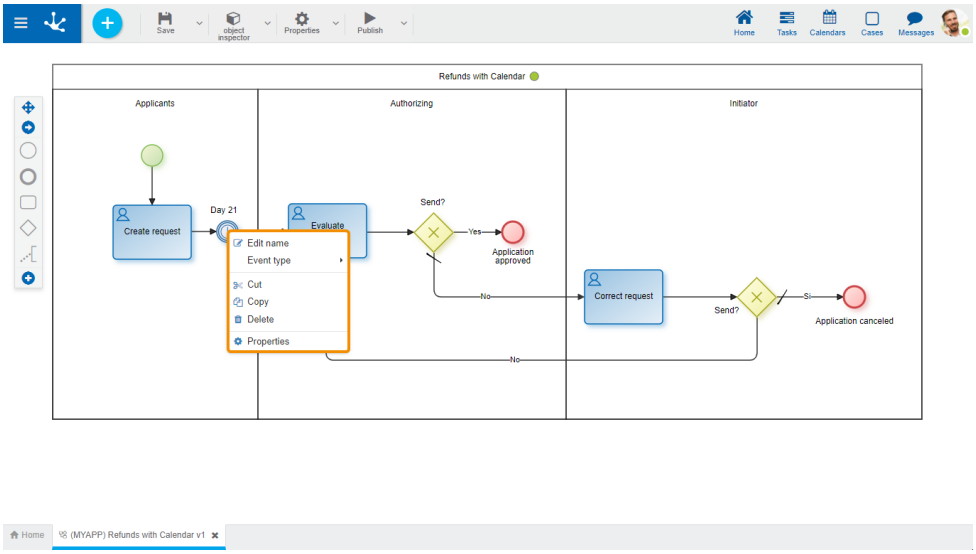
Adds an end event.

Adds a flow.

Adds an intermediate event.

Adds a comment.

Context Menu



- Edit name: Allows to edit the event name. Same functionality as the icon Abc.
- Event type: Allows to define the type of event. Its options vary depending on whether it is a start, intermediate or end event.
- Cut: Deletes the selected event. It can be pasted in another process location.
- Copy: Copies the selected event. It can be pasted in another process location.
- Delete: Deletes the selected event.
- Properties: Opens the properties panel to show and/or modify.

3.4.10.2.6. Gateway

A gateway is incorporated into the model to determine whether the data flow routes within a process branch or merge depending on the definition of conditions.

"Gateways are used to control how sequence flows interact as they converge and diverge within a process." [Workflow Management Coalition]

Types





Exclusive Gateway

An exclusive gateway is where only one of the exit paths continues with the task sequence.
Gateways have two representations, although both have the same behavior. By convention, the same representation must be used for the entire definition of a process.
- Exclusive gateway without marker.
- Exclusive gateway with marker.

The possible exit paths of the gateway are set by selecting one of the following options:

- **Definition of conditions**
  It is necessary to define conditions for gateway output flows, which are evaluated sequentially. When a condition is met, the process continues along this path, without evaluating the remaining conditions.

- **Button definition**
  It is necessary to define buttons for gateway output flows. When users press a button, the process continues along that path.

The exclusive gateway must have an output flow defined as default so that the process can continue when no conditions are met.



Parallel Gateway

It is used to create parallel paths and synchronize activities. All exit paths are activated without evaluating the conditions.
Parallel gateways can be defined in pairs, one as a divergence element to activate several parallel paths simultaneously and the other as a convergence element to synchronize previously activated paths. In the latter case, when the gateway is used for synchronization, the output is activated when the input paths are completed, that is, all activities preceding such point have ended.
The output flows of a parallel gateway do not require definition of conditions, since all exit paths must be completed.



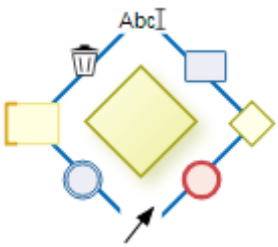
Inclusive Gateway

It is used to activate one or more paths and synchronize activities. All output flow conditions are evaluated.
Inclusive gateways can be defined in pairs, one as a divergence element to generate parallel activities and the other as a convergence element to synchronize previously activated paths. For divergence, at least one exit path must be activated, while for convergence, all activated paths must be completed.

- **Definition of conditions**
  It is necessary to define conditions for inclusive gateway output flows. All conditions are evaluated and the process takes all the paths where conditions are met.
- **Button definition**
  It is necessary to define buttons for gateway output flows. When users press a button, the process continues along that path.The same button can be defined for more than one flow.

An inclusive gateway should have an output flow defined as default, in order to prevent execution errors, when no flow condition is met.

Pie Chart Menu



Abc Opens an area for editing the name.

Deletes the gateway.

Adds an activity.

Adds a gateway.

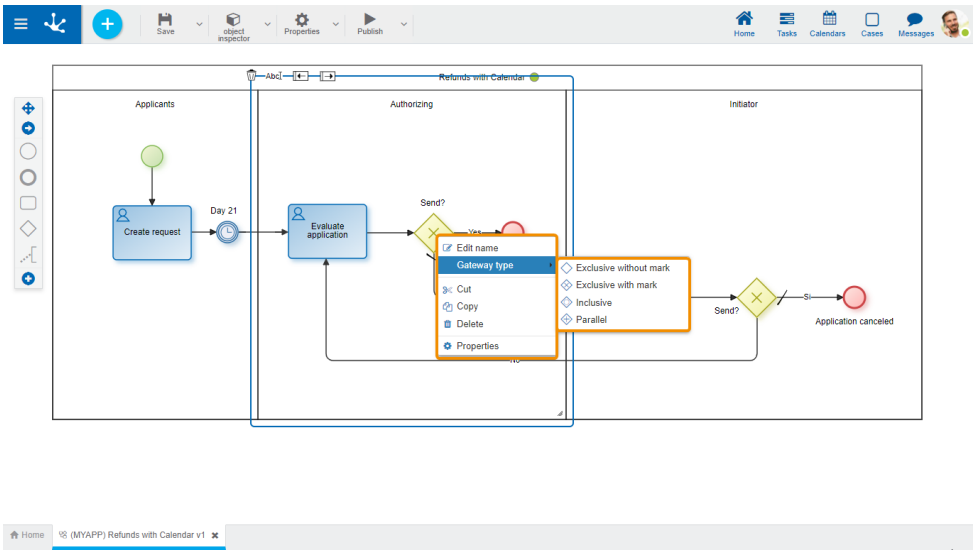
Adds an end event.

Adds a flow.

Adds an intermediate event.

Adds a comment.

Context Menu



- Edit name: Opens an area for editing the name. Same functionality as the icon AbcI.
- Gateway Type: Defines the type of gateway.
- Cut: Deletes the selected gateway. It can be pasted in another process location.
- Copy: Copies the selected gateway. It can be pasted in another process location.
- Delete: Deletes the selected gateway.
- Properties: Opens the property panel to show and/or modify.

3.4.10.2.7. Artifact

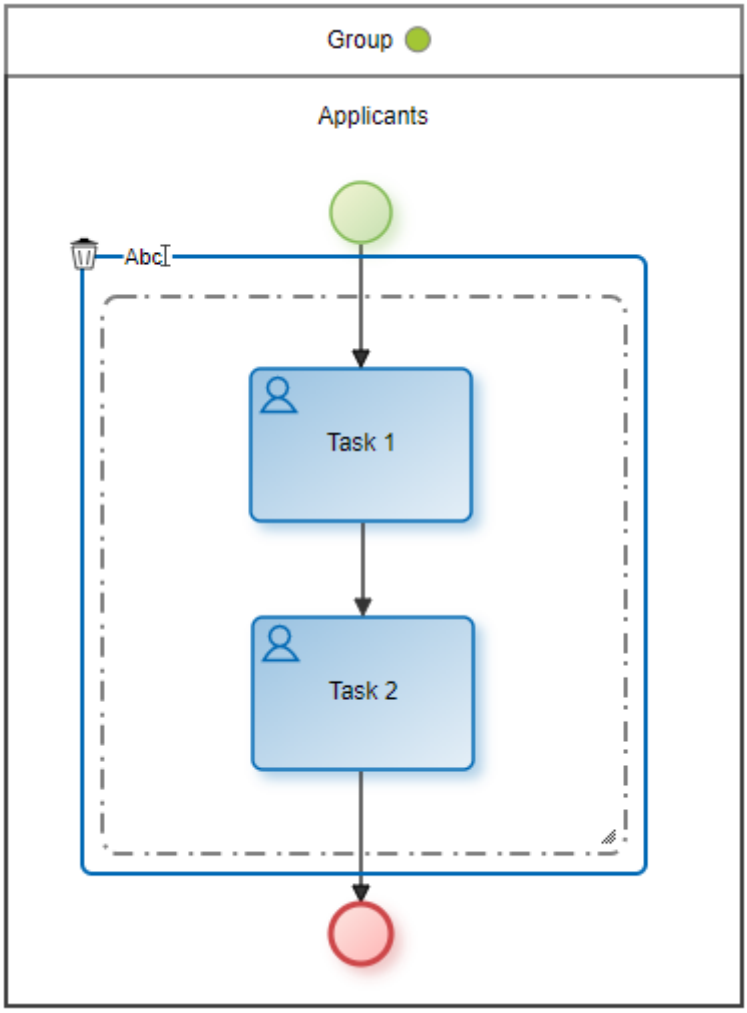
They are shapes that improve the expressiveness of diagrams, representing data related to the process but with no influence on performance.

"Artifacts are used to provide additional information about the process."[Workflow Management Coalition]

Group

Visually groups shapes in the diagram.

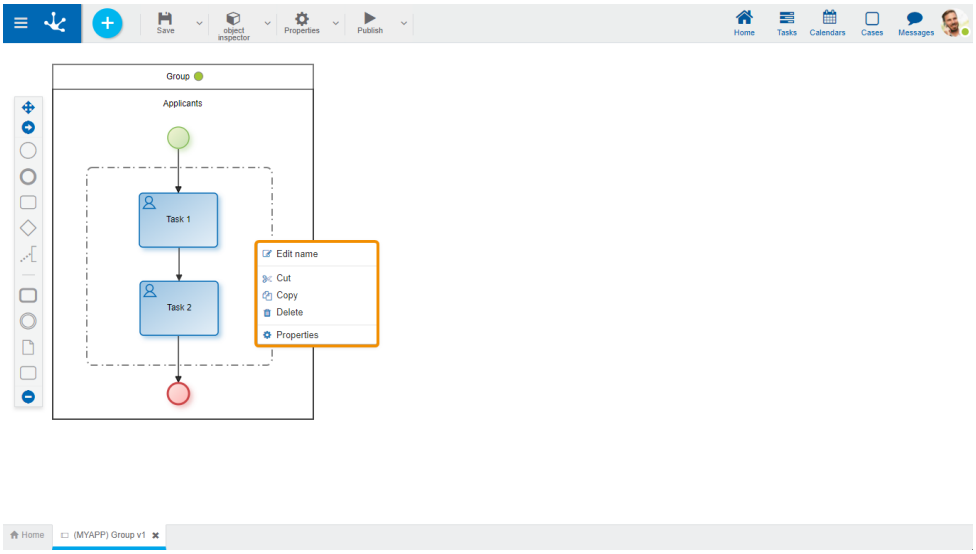
Pie Chart Menu



Abc Opens an area for editing the name.

Deletes the group.

Context Menu

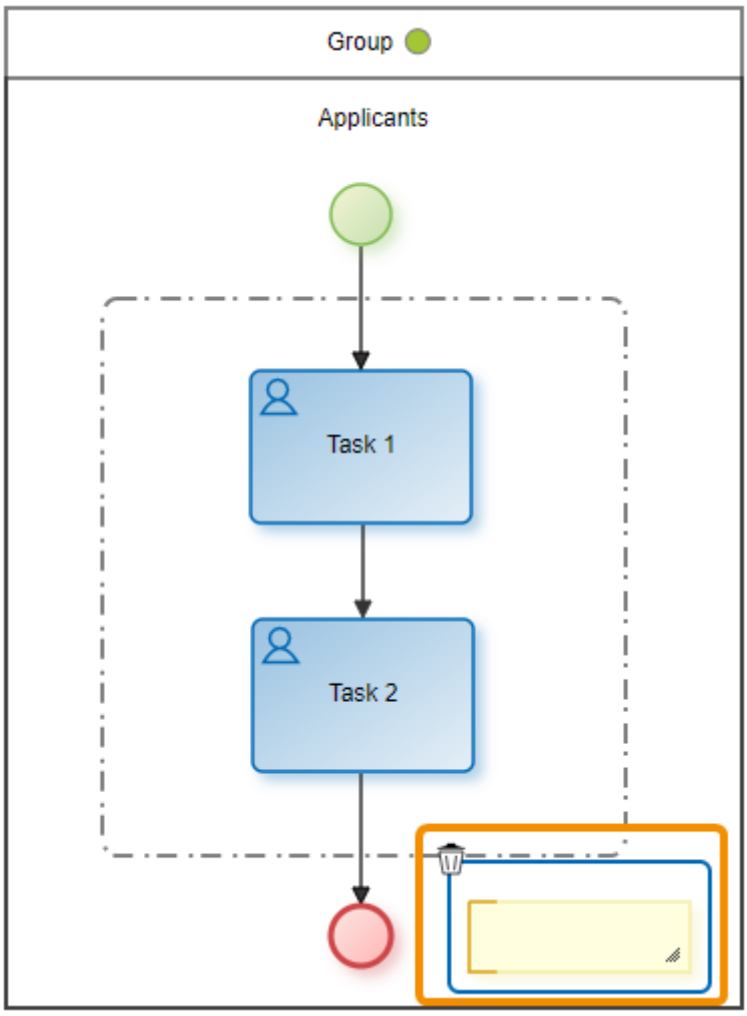


- Edit name: Edits the group name. Same functionality as the icon AbcI.
- Cut: Deletes the selected group keeping the shapes it groups. It can be pasted in another process location.
- Copy: Copies the selected group. It can be pasted in another process location. It does not copy the elements it groups.
- Delete: Deletes the selected group keeping the shapes it groups. Same functionality as the icon .
- Properties: Opens the properties panel to show and/or modify.

Annotation

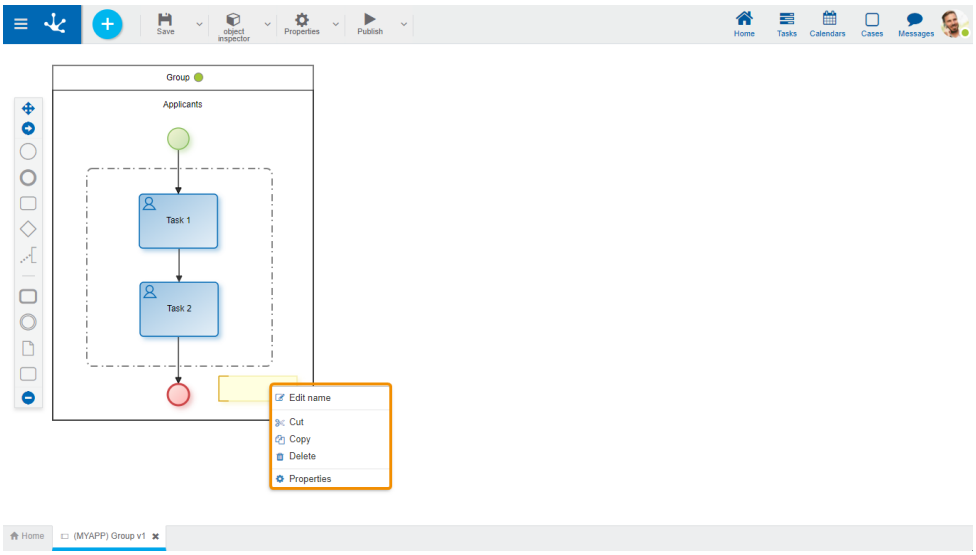
Area where the modeler can write explanatory texts.

Pie Chart Menu



Deletes the selected annotation.

Context Menu

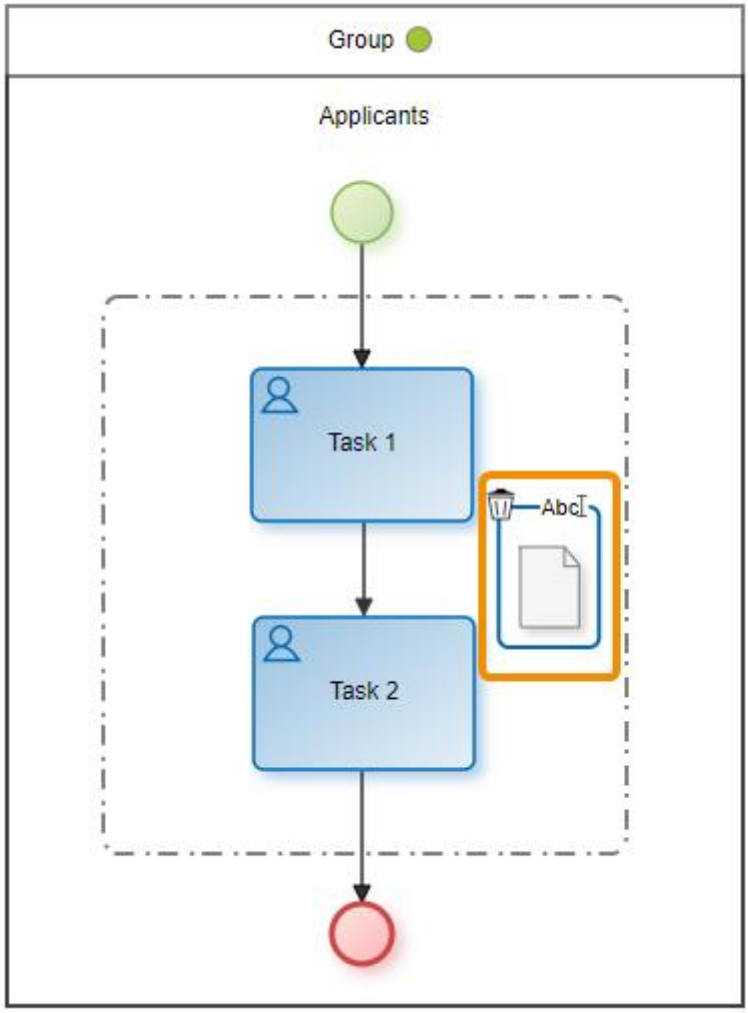


- Edit name: Allows to edit the annotation text.
- Cut: Deletes the selected annotation. It can be pasted in another process location.
- Copy: Copies the selected annotation. It can be pasted in another process location.
- Delete: Deletes the selected annotation. Same functionality as the icon .
- Properties: Opens the properties panel to show and/or modify.

Data Object

Represents the information that accompanies the process circuit or is generated during it, such as documents or notifications.

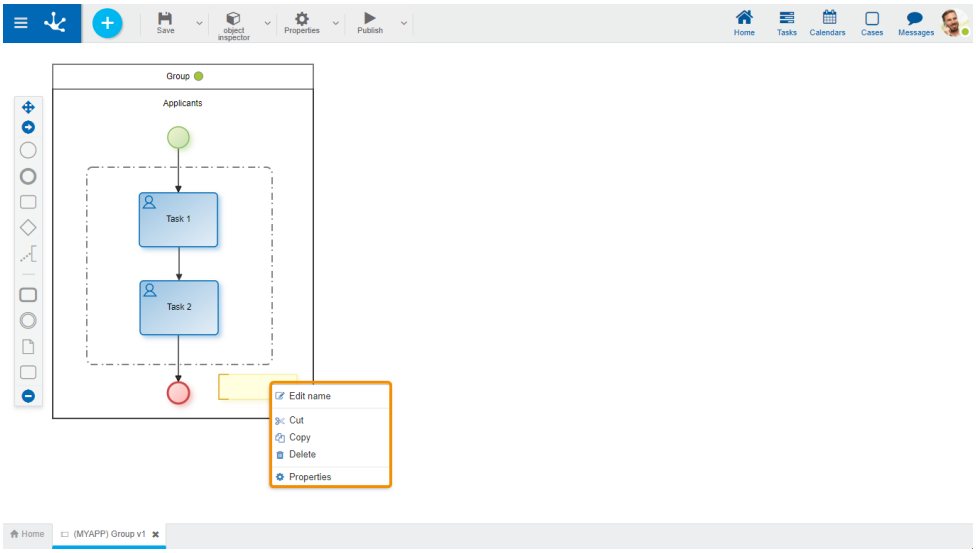
Pie Chart Menu



Abc Opens an area for editing the name.

Deletes the selected data object.

Context Menu

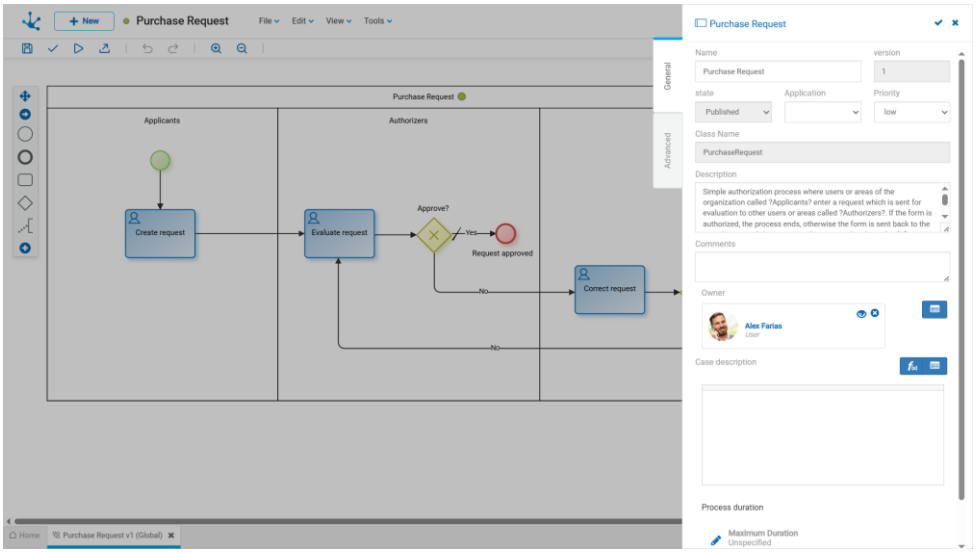


- Edit name: Edits the name of the data object. Same functionality as the icon AbcI.
- Cut: Deletes the selected data object. It can be pasted in another process location.
- Copy: Copies the selected data object. It can be pasted in another process location.
-Delete: Deletes the selected data object. Same functionality as the icon .
- Properties: Opens the properties panel to show and/or modify.

3.4.10.3. Process Properties

There are different ways to enter a process properties panel:

- Press the icon from the top toolbar.
- Double click on the graphic element corresponding to pool.
- Access the context menu of the pool graphic element.

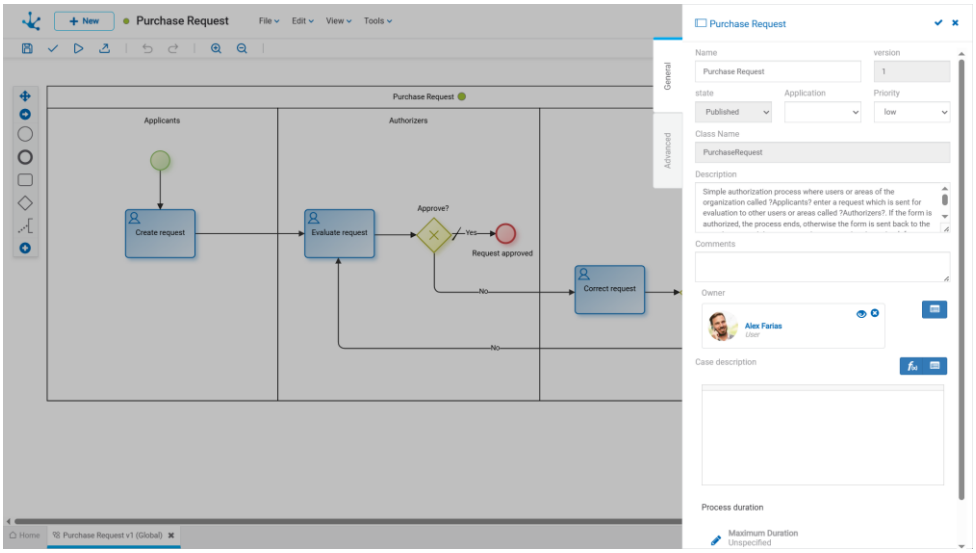


Tabs

- General
- Advanced

3.4.10.3.1. General

The properties panel is displayed on the right side of the process modeler, where the first tab corresponds to general information.



Properties

Name

It is the name used at the modeling level to refer to the process, for example, in the modeler's grid and in the object inspector.

Version

Process version number. Along with the Identifier property, they form the unique identification of the process.
This attribute allows to incorporate modifications in the process design without impacting executions that are still active (unfinished cases).
It is a non-modifiable attribute. The modeler user can version the process from the top toolbar.

State

Indicates the current state of the process. It is automatically updated when start, modify, save, publish and inactivate operations are executed.
The state of the process determines the level of validations that are carried out on it and the possibility of being used.

Application

Allows to define the application to which the object belongs. If no application is reported, the object is assigned to the application "Global".

Priority
Indicates the priority assigned to the process activities, when they do not have an expressly defined priority. Possible values are URGENT, HIGH, MEDIUM, LOW.

Description
Text that defines the process describing its functionality.

Comments
Observations on the process.

Owner
Indicates the person who has control and vision of the process from start to finish. The owner can be a user, an organizational unit, a role, or an agent.
It is, by default, the logged in user.

 Opens the profile show property.

 Deletes the informed owner

 Allows to select an existing owner

You can also use the autocomplete function. As characters are entered into this field, the first five matches dynamically appear to select one of them.

Case Description
It is a description that represents each one of the process executions.

 Allows to select functions from a list to incorporate dynamic information into the case description.

 Allows incorporating attributes of the entities that are used in the execution of the process to the case description. For example, in the "Refund Request" process, the request date can be incorporated, which is a form variable used in such process.

The description entered is displayed in:

- The last tasks, by pressing the Tasks icon on the top toolbar.
- The task execution panel, below the activity name.
- The Description column in the tasks grid.
- The last tasks, on the element that represents the form in the grid of "Forms and Tasks".

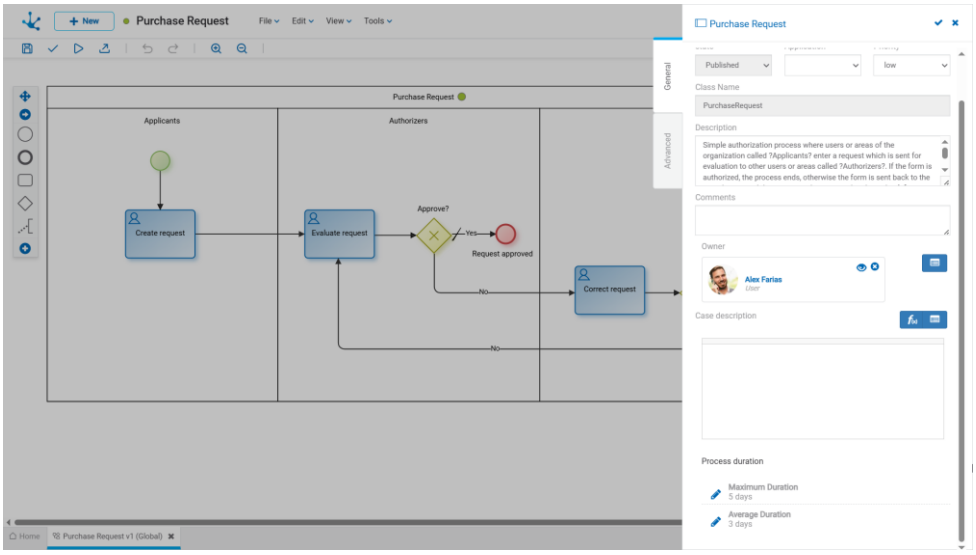
Process Duration
Allows to set the Maximum duration and Average duration to carry out the process.

 Allows to edit the maximum and average duration.

- By duration
  A deadline in days, hours, minutes and seconds is reported.
- By due date
  A date-type variable is selected, from a form associated with the process, using the icon  In the process execution, the value of such variable is used as due date. If the value of the variable used to define the due date is modified, process duration is automatically updated.

 Saves changes made to duration.

 Cancels changes made to duration.



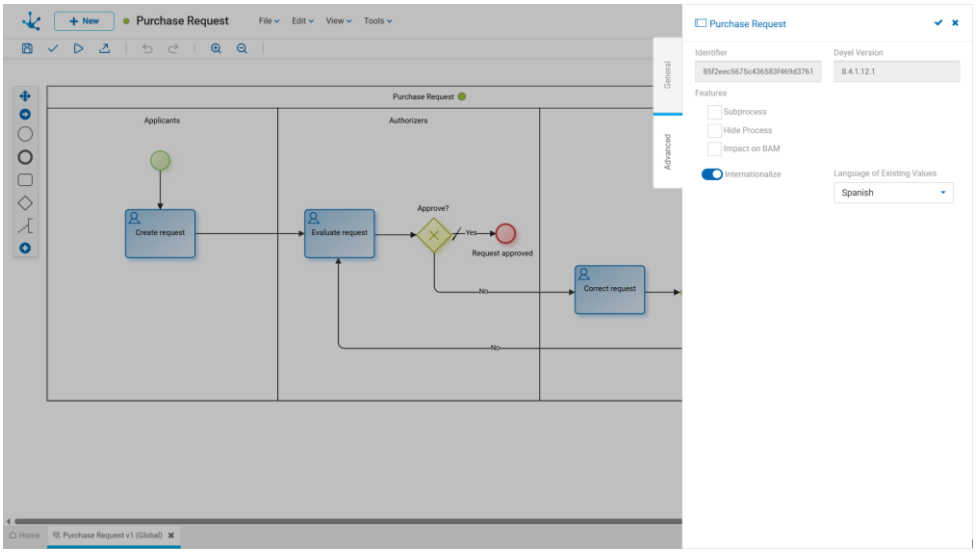
Actions

The icon ✔ is used to confirm the modifications made in the properties panel.

The icon ✖ is used to close the properties panel, if it was not previously saved, changes are discarded.

3.4.10.3.2. Advanced

The second tab of the side panel corresponds to advanced properties of the process.



Properties

Identifier

Alphanumeric code that is automatically assigned to the process. Users cannot modify it. Along with the Version property, they form the unique identification of the process.

Deyel Version

Indicates a version by **Deyel** with which the process was last saved or published.

Features

One of the following options can be indicated:

Subprocess

Indicates that a process is defined to be used exclusively as a subprocess of others. It can only be started when it is invoked from another process in an activity of type subprocess. It cannot be started by a user manually from the functionality "Start Cases".

Hide Process

Indicates if a process remains visible in the Context Menu and from Forms and Tasks so that users can start it.

Impact on BAM

Indicates if process data is taken into account in the BAM reports. By default, the property is not checked and the process is not included in BAM. The property can be checked so that from a certain moment, the **Deyel** information goes to BAM.

The following situations should be taken into account:

- If the process is published, modified, or has data, the property can be checked, but not unchecked.
- If the property needs to be unmarked, all instances of the process must first be deleted with the "Delete Data" function so that the property is enabled for modification.

Internationalize

Enables the translation of object properties and its elements into different languages. When activated, it enables the option "Internationalization" from the "Tools" drop-down menu and the property Language of Existing Values, which is displayed on the right.

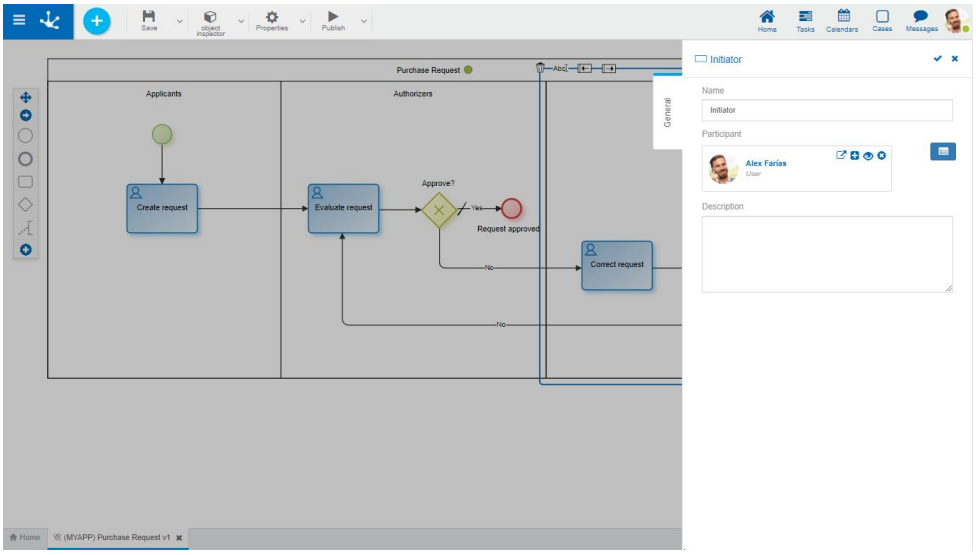
Language of Existing Values

Displayed when the property Internationalize is activated for the first time and indicates the original language. Its selection is required and the available options are: "Spanish", "English" and "Portuguese".

3.4.10.4. Lane Properties

There are different ways to enter the properties panel of a lane:

- Double click on the graphic element corresponding to the lane.
- Access the context menu of the lane graphic element.



General Tab

Properties

Name

Lane name. It must clearly represent those who perform the tasks within the lane.

Participant

It is the participant in charge of executing the activities included in the lane.

Allows to select an existing participant, which can be a user, an organizational unit, a role, or an agent. The autocomplete formula can also be used. As characters are entered into this field, the first five matches dynamically appear to select one of them.

Allows to display a window with complete information on the participant, if he is defined as a user, organizational unit or role.

Opens a window to define a new role.

Opens the profile show panel only if it is a user.

Deletes the informed participant.

Description

Conceptual description of who executes the tasks within the lane.

Actions

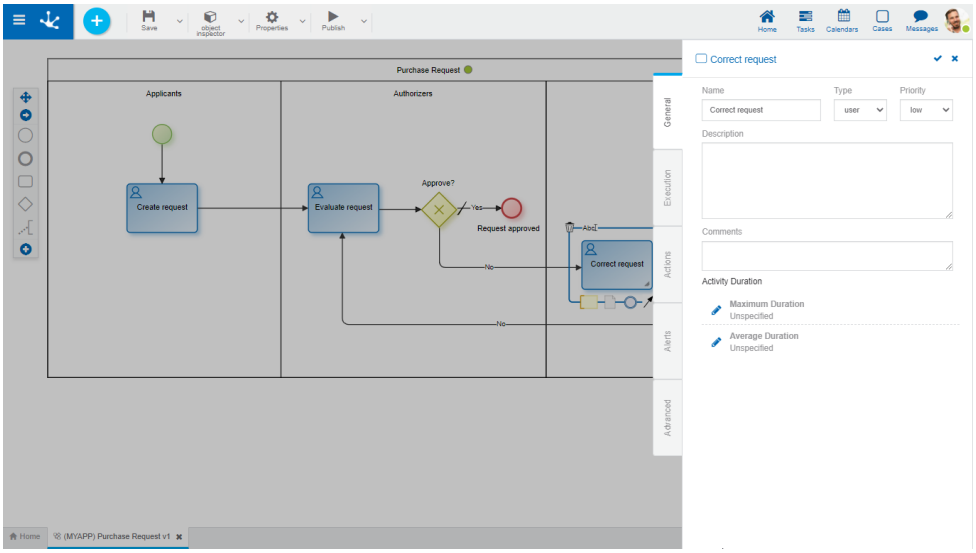
The icon is used to confirm the modifications made in the properties panel.

The icon is used to close the properties panel, if it was not previously saved, changes are discarded.

3.4.10.5. Activity Properties

There are different ways to enter the properties panel of an activity:

- Double click on the graphic element corresponding to the standard activity or task.
- Access the context menu of the activity.



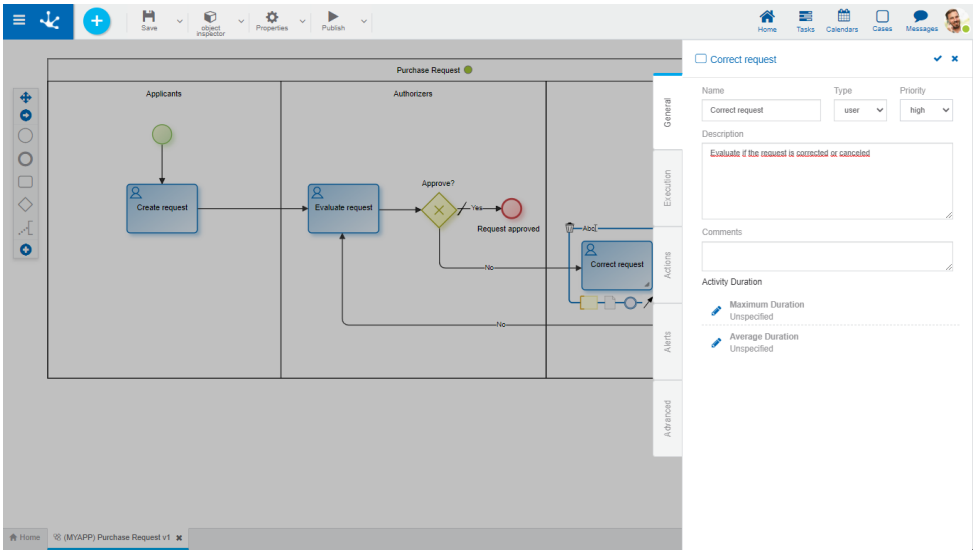
Tabs

- General
- Execution
- Actions
- Alerts
- Advanced

The business modeler user can have access to all tabs except for "Advanced". The IT modeling user can access all of them.
For each of the "General", "Execution", "Actions" and "Alerts" tabs, access permissions are defined from the security module. Users can see only the tabs they are allowed to access.

3.4.10.5.1. General

The properties panel is displayed on the right side of the activities modeler, where the first tab corresponds to general information.



Properties

Name
Activity name. It refers to the action performed by the activity.

Type
Defines the type of activity.

Priority
Indicates the activity priority. Possible values are URGENT, HIGH, MEDIUM, LOW.
The user can display this property in the to-do list and use it as a sort criteria.

Description
Description of the activity that can be used as an online manual.
The user who executes the activity can see it by clicking on the icon .

Comment
In this field, the designer can document in detail what the task consists of.

Activity Duration
Allows to set the Maximum Duration and the Average Duration of the activity.

When the execution of the activities begins, these durations are transformed into a maximum due date and an expected expiration date, respectively. The activity due date calculation is made in different ways, depending on whether the definition of maximum duration is made according to duration or due date.

In the case show or in the tasks show, the due date and time of the task are displayed and in addition, on these durations, alerts can be set.

Edits the maximum and average duration.

- By duration
  A deadline in days, hours, minutes and seconds is reported.

- By due date
  A date type variable is selected from a form associated with the process, using the wizard icon  .

  In the activity execution, such variable value is used as due date. If this variable value is modified, the due date of the activity is automatically modified.

Actions

The icon ✔ is used to confirm the modifications made in the properties panel.

The icon ✖ is used to close the properties panel, if it was not previously saved, changes are discarded.

3.4.10.5.1.1. Activity Due Date Calculation

The activity due date is calculated in different ways, depending on whether the maximum duration was modeled from a duration or by selecting a form field related to the process.

> *The calculation detail is made for the due date but it is also valid for the average date, depending on the modeling of the average duration of the activity.*

Due Date Based on a Duration

Elements involved in the calculation:

- Activity start date and time.
- Activity maximum duration.
- Calendar of the participant who executes the activity.
- Configuration of holidays and special dates.

Calendar of the Participant Who Executes the Activity

When the participant is a user:

- The calendar established in the user's profile is considered.
- If not indicated, the calendar established in the user's organizational unit is considered.
- If the unit does not indicate a calendar, it moves up in the hierarchy, searching for a higher unit to establish a calendar.
- If no higher unit indicates a calendar, the standard calendar is considered.

When the activity is executed in an organizational unit:

- The calendar established in the organizational unit is considered.
- If the unit does not indicate a calendar, it moves up in the organizational structure, searching for a higher unit to establish a calendar.
- If no higher unit indicates a calendar, the standard calendar is considered.

When the activity is assigned to a generic role:

- The standard calendar is considered.
- When role players assign themselves an activity, the due date is recalculated considering the calendar of the executing participants.

Steps for Activity Due Date Calculation

Step 1: The maximum duration of the activity is calculated, expressed in seconds.

The number of days, hours, minutes and seconds, indicated in the activity definition, is considered.

The number of hours and minutes of a working day is retrieved from the executing participant's calendar.

The maximum duration of the activity is calculated as:

```
Maximum Duration in Seconds =
Activity.Days          * (Calendar.Hours * 3600 + Calendars.Minutes * 60) +
Activity.Hours  * 3600 +
Activity.Minutes       * 60 + Activity.Seconds
```

Step 2: The user's working times are defined, through the executing participant's calendar, together with the definition of general holidays and specific dates.

Each of the work bands is perfectly defined by the calendar, and has a start date and time and an end date and time, therefore, this specification can be converted to a number of work seconds.

The definition of "Holidays" means that certain periods are considered non-working days.

The definition of "Specific Date" allows to establish specific working times for a specific date.

The definition of specific dates that is performed at the calendar level takes precedence over the general specific dates.

Step 3: Express the Start Date and Time of the activity, in the time zone of the executing participant's calendar.

Based on that date, working times are assigned until all the seconds that correspond to the activity duration are assigned.

In this way, the activity due date is calculated, according to the user's working times.

Due Date Based on a Form Field

The due date takes the value of a form field.

If the field data type is Date or Local Date, if the value entered is MM/DD/YYYY, the due date is MM/DD/YYYY at 23:59:59.999

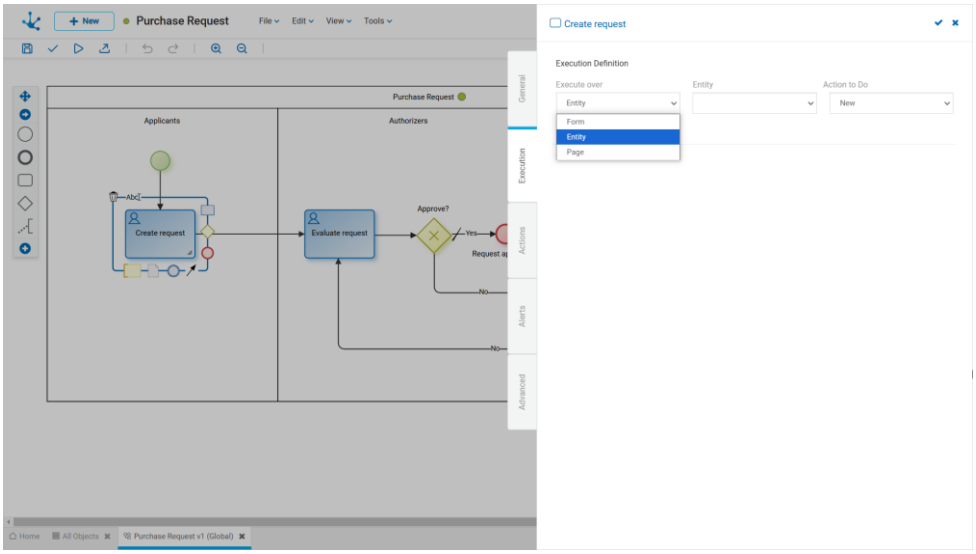
If the field data type is Date and Time, the DD/MM/YYYY HH:MM:SS value is considered as the due date.

If the field type is Local Date and Time, the DD/MM/YYYY HH:MM:SS value is transformed to the standard calendar time zone and is considered as the due date.

3.4.10.5.2. Execution

Process Modeling > Modeling the execution of a process

The second side panel tab of a task corresponds to execution information and it is displayed only for manual, user, or undefined tasks.
It allows defining the behavior of the object related to the task, which will establish the interface for its execution. This way allows choosing to execute the task using an entity, a form, or a page.



The Execute on property allows selecting the type of object to be displayed during task execution.

Possible Values

- Entity
- Form
- Page

3.4.10.5.2.1. Execute on a Form

When executing the task, the selected form is displayed, where the behavior of each of the fields and containers may be redefined to meet the specific requests of the task.

The forms that can be selected are those that are related to the process.

Execution Definition

In this section, the general properties of the task execution are defined.

Form
Displays the name of the form related to the process used in the task.

Action to Do
Action performed on the object in this task.

- **Open**
  Displays the instance in show mode. Fields cannot be edited.
- **New**
  Creates a new instance that is related to the case. When an instance already exists, it works as the "Modify" action.
- **Modify**
  Displays the existing instance and allows modifying the fields defined as editable.

Options

Require Confirmation
This option displays a confirmation message to execute a task.
A "Confirmation message" container is displayed by selecting this check. When displayed, an editing area opens to enter the text of the message or an embedded rule that returns the confirmation message if applicable. If the rule does not return a text, the task is executed without requiring confirmation.

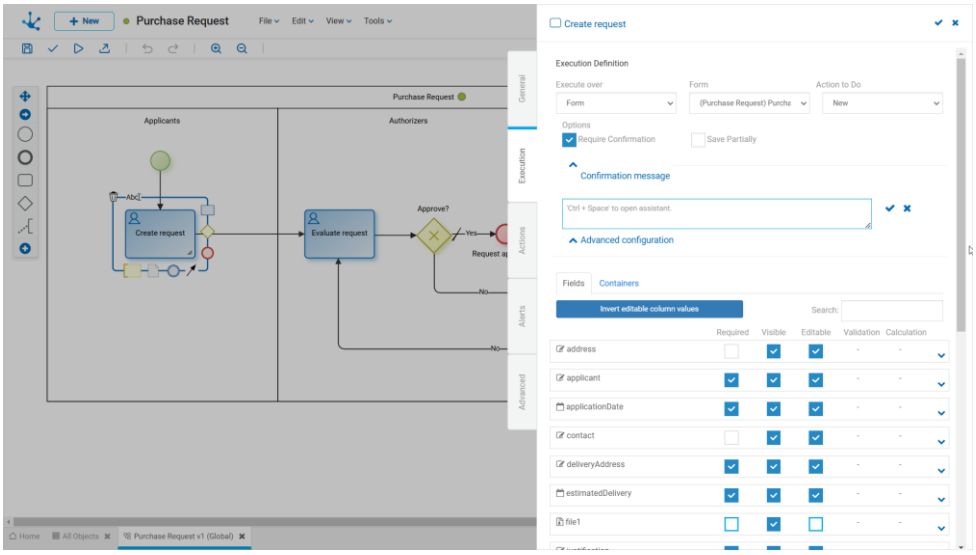
Save Partially
This option indicates that when executing the task, the "Save" button is added, which allows to keep the values entered by the user, with no need to execute the task.

Advanced Settings

To show this section, an action to do must have been previously defined.

Fields

In this tab, a grid with the object fields is displayed. The properties of these fields can be redefined for the execution of each task.



Invert editable column values

Inverts the values of the Editable property, for all the fields displayed in the grid. If the search filter narrows it, the modification only affects filtered fields.

Search:

As characters are entered to the search field, it dynamically filters the fields whose Label or Name properties, indistinctly, match the text being entered.

Fields Grid

It is made up of object fields according to the filter applied. Each field is identified by its Name property.

The grid columns correspond to the modeled behavior properties, whether defined by the object modeler or by the task itself.

Properties can be modeled simply by checking or unchecking them. If they have embedded rules defined, they are displayed by icons. Checks and icons are displayed in blue if properties or rules were modeled in the object, while they are in light blue if they were modeled in the task.

If properties or rules are modeled by the object and any of them are modified by the task, these modifications overwrite the definitions of the object only for the task execution.

> *If any property or rule of the object is overwritten by a task, subsequent modifications made by the object have no impact on the behavior already defined in the task.*

> *If a field is added to the object, it is added to the tasks that use the object, as non-required and non-editable, regardless of how it has been modeled on the object.*

Required

Indicates if the field is required during the task execution and if any required rule is defined, it displays the icon .

Visible

Indicates if the field remains visible in the task and if any visibility rule is defined, it displays the icon .

Editable

Indicates if the field is editable for the task and if any editability rule is defined, it displays the icon .

Validation

Indicates if there is a validation rule for the field with the icon .

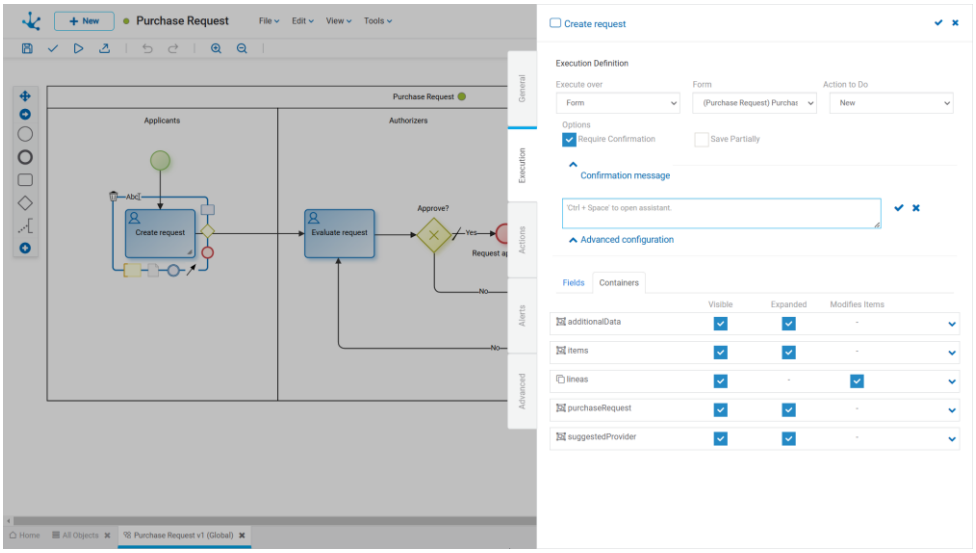
Calculation

Indicates if there is a calculation rule for the field with the icon .

> *Clicking the icon expands the area of values and rules modeling of each field.*

Containers

In this tab, a grid with the form containers is displayed. The properties of these containers can be modified to suit the execution of each task.



The grid columns correspond to the properties of containers.

Visible

This check indicates that the container remains visible when executing the task. If it has associated visibility rules, it is represented by the icon .

Expanded
This check indicates that the container is displayed expanded when executing the task.

Modifies Item
Only for iterative containers. This check indicates if iterative occurrences can be added or deleted. This property is displayed by default as checked; however, if an iterative container is added to the object, it is added in this tab with this property unselected.

> *Clicking the icon*  *expands the embedded rules modeling area for each container.*

3.4.10.5.2.2. Execute on an Entity

When executing the task, the selected entity is displayed, where the behavior of each of the fields and containers may be redefined to meet the specific requests of the task.

The entities that can be selected are those related to the process.

Execution Definition

In this section, the general properties of the task execution are defined.

Entity
Displays the name of the entity related to the process used in the task.

Action to Do
Action performed on the object in this task.

- **Open**
  Displays the instance in show mode. Fields cannot be edited.
- **New**
  Creates a new instance that is related to the case. When an instance already exists, it works as the "Modify" action.
- **Modify**
  Displays the existing instance and allows modifying the fields defined as editable.

Options

Require Confirmation
This option displays a confirmation message to execute a task.
A "Confirmation message" container is displayed by selecting this check. When displayed, an editing area opens to enter the text of the message or an embedded rule that returns the confirmation message if applicable. If the rule does not return a text, the task is executed without requiring confirmation.

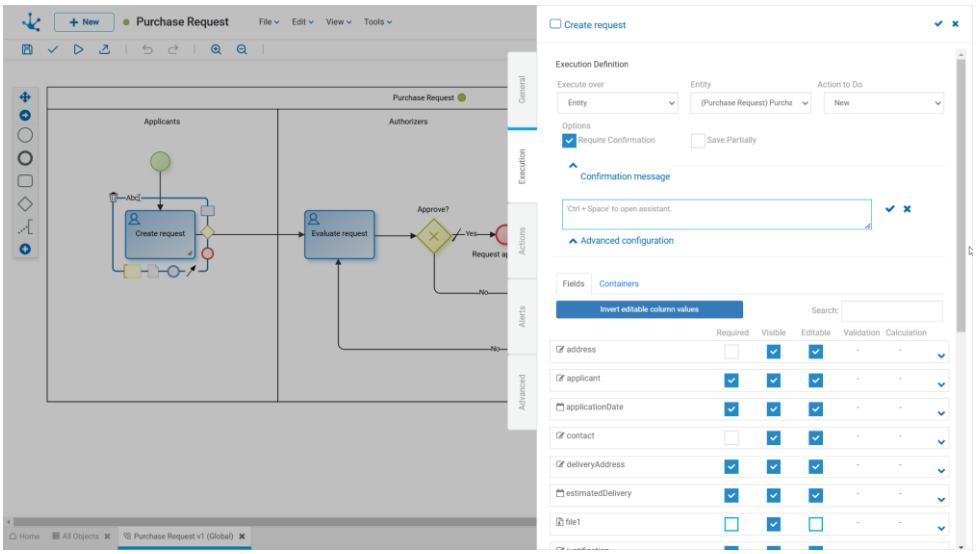
Save Partially
This option indicates that when executing the task, the "Save" button is added, which allows to keep the values entered by the user, with no need to execute the task.

Advanced Settings

To show this section, an action to do must have been previously defined.

Fields

In this tab, a grid with the object fields is displayed. The properties of these fields can be redefined for the execution of each task.





Inverts the values of the Editable property, for all the fields displayed in the grid. If the search filter narrows it, the modification only affects filtered fields.



As characters are entered to the search field, it dynamically filters the fields whose Label or Name properties, indistinctly, match the text being entered.

Fields Grid

It is made up of object fields according to the filter applied. Each field is identified by its Name property.

The grid columns correspond to the modeled behavior properties, whether defined by the object modeler or by the task itself.

Properties can be modeled simply by checking or unchecking them. If they have embedded rules defined, they are displayed by icons. Checks and icons are displayed in blue if properties or rules were modeled in the object, while they are in light blue if they were modeled in the task.

If properties or rules are modeled by the object and any of them are modified by the task, these modifications overwrite the definitions of the object only for the task execution.

> *If any property or rule of the object is overwritten by a task, subsequent modifications made by the object have no impact on the behavior already defined in the task.*

> *If a field is added to the object, it is added to the tasks that use the object, as non-required and non-editable, regardless of how it has been modeled on the object.*

Required
Indicates if the field is required during the task execution and if any required rule is defined, it displays the icon  .

Visible
Indicates if the field remains visible in the task and if any visibility rule is defined, it displays the icon  .

Editable
Indicates if the field is editable for the task and if any editability rule is defined, it displays the icon  .

Validation
Indicates if there is a validation rule for the field with the icon  .

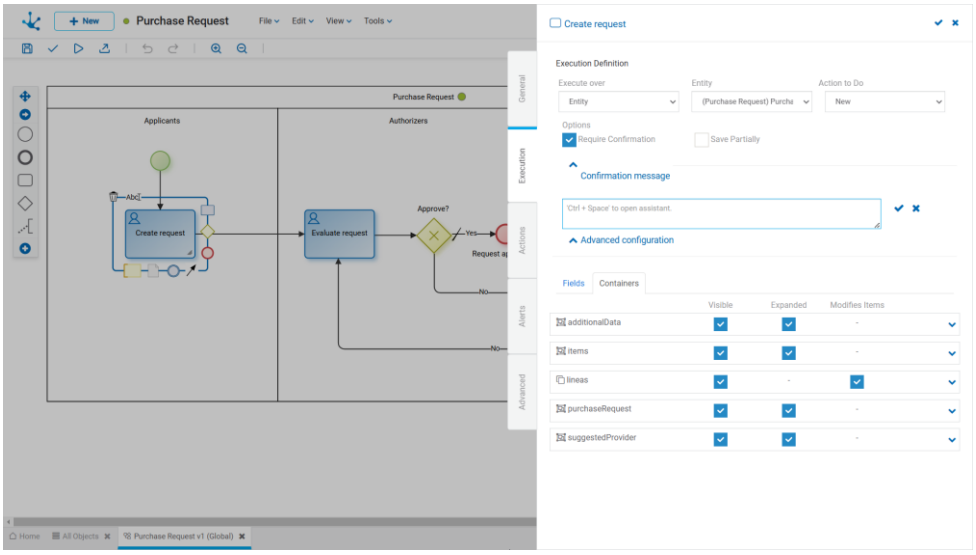
Calculation
Indicates if there is a calculation rule for the field with the icon  .

> *Clicking the icon*  *expands the area of values and rules modeling of each field.*

Containers

In this tab, a grid is displayed with the containers and sections that have the property Configurable within Activity active, along with repeaters, which are automatically included if they contain at least one field.

For containers, sections and repeaters, it is possible to modify the visibility. Additionally, in the case of repeaters, it is possible to adjust the property Modifies Items. These configurations allow adapting the behavior of elements based on the execution of each task.



The grid columns correspond to the properties of containers.

Visible

This check indicates that the container remains visible when executing the task. If it has associated visibility rules, it is represented by the icon .

Expanded

This check is only available when executing on a form.

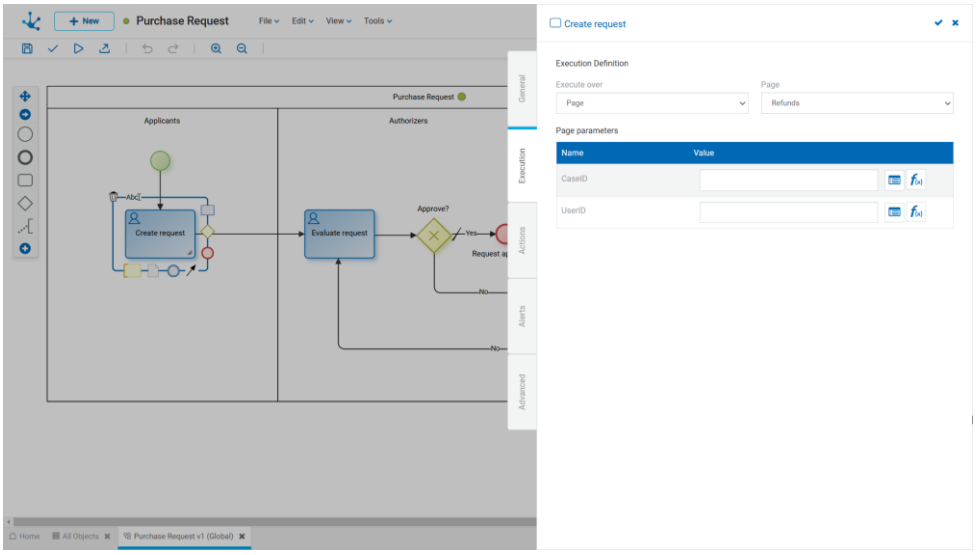
Modifies Item

Only for iterative containers. This check indicates if iterative occurrences can be added or deleted. This property is displayed by default as checked; however, if an iterative container is added to the object, it is added in this tab with this property unselected.

> *Clicking the icon expands the embedded rules modeling area for each container.*

3.4.10.5.2.3. Execute on a Page

When executing the task, the selected page is displayed. In the modeling, values for the input parameters can be specified.



Execution Definition

In this section, the general properties of the task execution are defined.

Page

Displays the name of the page displayed in the task.

Page parameters

This section lists the page parameters. Its value can be set using the variable and function wizard.

Page features that enable task execution

The modeling user should consider that the page selected to be displayed when executing the task must include some button or graphic element that allows the user to indicate that they want to execute it.

For the execution to work correctly, it is necessary that said graphic element has a particular behavior, defined by Javascript code.

JavaScript example to execute a button in a process

```
function submitBtn_onClick(event) {
    $d.executeActivity({
        buttonToExecute: "Submit",
        formId: "ENTITY_ID",
        fieldsActivity:{
            name: $d.getInput("name").getValue(),
            email: $d.getInput("email").getValue(),
```

```
                phone: $d.getInput("phone").getValue(),
                idParam: $d.getParameter("id").getValue(),
            }
        });
```

Deyel SDK for Javascript, provides the "executeActivity" method, which receives the following parameters:

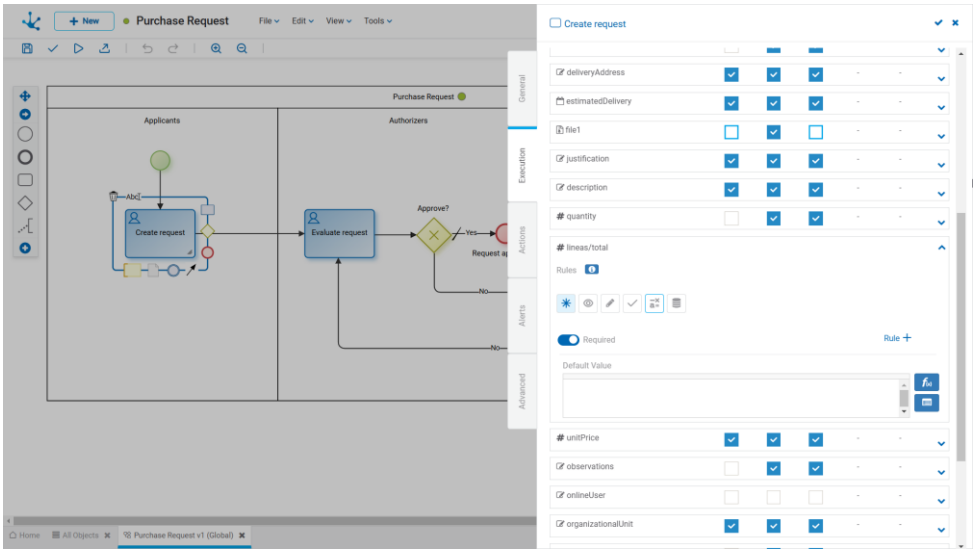
- buttonToExecute: Button clicked by the user, which allows the correct path to be selected to evaluate the conditions of the gates.
- formId: Entity or form to be modified.
- fieldsActivity: List of fields with their respective values.

When executing this method, the task is executed, redirecting the user to the next activity.

3.4.10.5.2.4. Values and Rules Modeling Area

From this area, behavior, validation and calculation rules are defined and default values of the fields are established.
The properties and rules that are defined in the task overwrite those that had been defined in the fields of the related object.

Field Properties Panel



Properties

Rules

It is possible to define rules of behavior, validation, and calculation, associated with a field for the task. The behavior rules of the field defined in the task overwrite those modeled in the field of the related object, while validation rules are added.

> *If properties or rules of the related object are overwritten from a task, subsequent modifications from the object for the* **overwritten properties**, *have no impact on the behavior already defined in the task.*



Required

Indicates whether the field is required during the execution of the task.
 Required Not required (default)
Rule + Opens an editing area where a rule to determine the required condition can be defined by using the wizard (ctrl + space). If a rule is defined, the icon is displayed with a light blue border.



Visible

Indicates whether the field is visible. If this property is not selected, the field will not be displayed during the execution of the task.
 Visible (default) Not visible
Rule + Opens an editing area where a rule to determine the visibility condition can be defined by using the wizard (ctrl + space). If a rule is defined, the icon is displayed with a light blue border.



Editable

Indicates if the field is editable. If this property is not selected, the user cannot enter or modify values in the field during the execution of the task.
 Editable (default) Not editable
Rule + Opens an editing area where a rule to determine the editability condition can be defined by using the wizard (ctrl + space). If a rule is defined, the icon is displayed with a light blue border.



Validation

Rule + Opens an editing area where the condition that determines if the field value is correct can be defined. It is possible to define more than one rule. If rules are defined, the icon is displayed with light blue borders.



Calculation

Rule + Opens an editing area where the expression to be executed to calculate the field value can be defined. If a rule is defined, the icon is displayed with a light blue border.

Persists
Indicates whether the value displayed in the field in a task is stored or not in the database. Persistence is defined by default, with the exception of indicating that it does not persist.

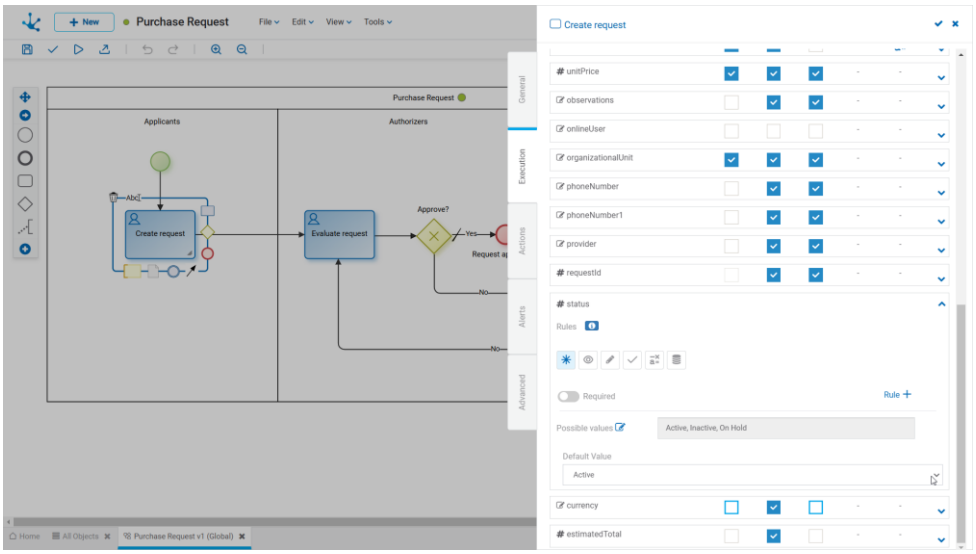
Persists (default) Does not persist

Undo changes
It allows deleting rules or behavior properties modeled from the task. In this way, behavior properties and rules in effect during the task execution revert to those originally defined.

Displays syntax examples to write rules.

Default Value
It is an area to define the field default value.
Allows to select functions from a list to incorporate dynamic information into the field.
It allows incorporating information of the objects that participate in the process execution to the field content. For example, in the Refund Request process, the request date can be incorporated, this data is available in the date field of the Refund Request object.

Field Properties Panel with Relation



If the field has a relation, the possible values and the default value of the field in the task are displayed.

Possible Values

Clicking on the icon enables a panel to filter data from the list in the task.

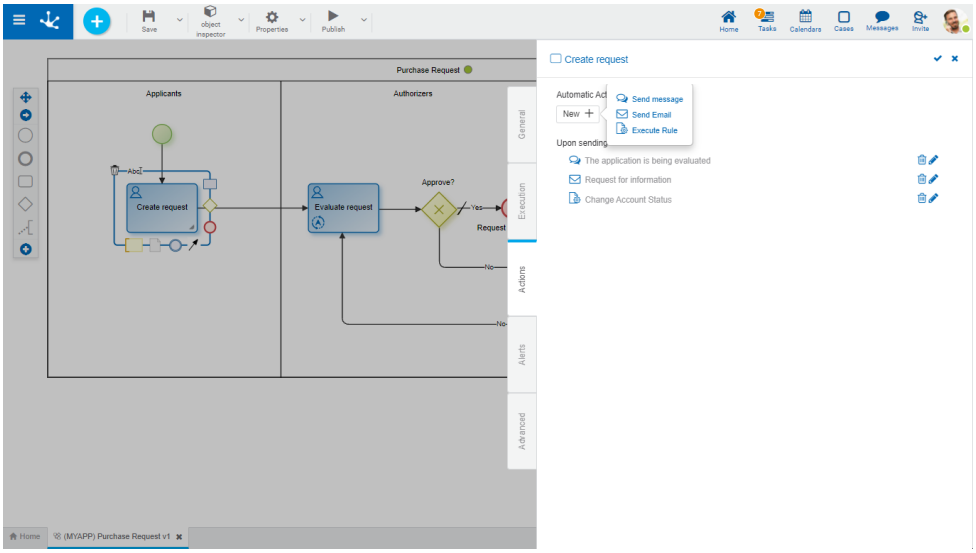
Invert Selection Inverts the selected values.

Select all Selects all values.

Exclude Values: It allows excluding checked values from the selection.

3.4.10.5.3. Automatic Actions

In the "Actions" tab, automatic actions executed in the activity are defined.



The icon on the symbol of an activity indicates that it has an automatic action. Hovering the cursor over the icon enables a panel that shows in detail all the automatic actions modeled for the activity.

New + Pressing this button you can create new automatic actions of different types.

Types

Send message

Send Email

Execute rule

For existing actions, the icon allows editing a previously defined automatic action and the icon deletes it.

Properties

The different types of automatic actions share properties and have other specific ones.

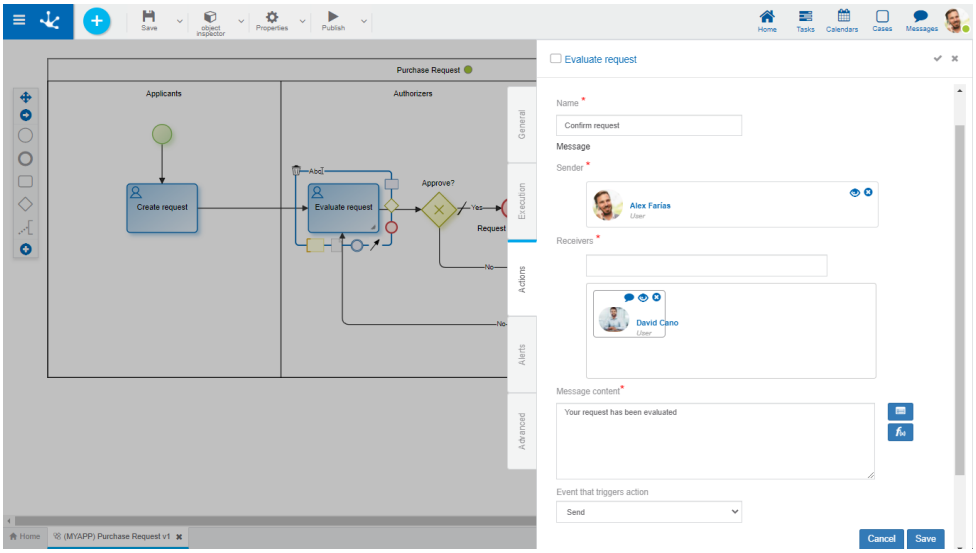
Name
Name that identifies the automatic action in the activity and is used in the description of the activity execution in the graphic show of a case.

Event that triggers the action
Moment when the automatic action is executed, defined by one of the following events:

- Send: Indicates that the action is executed when the task execution finishes.
- Receive: Indicates that the action is executed when the task execution starts.
- Disabled: Execution of the action is disabled.

3.4.10.5.3.1. Send Message

Allows to define the sending of a message via the chat of **Deyel**. The message may include process-related form variables and functions that allow to incorporate links to case shows, to do lists, forms, application access links, among others.



An asterisk "" on the label indicates that the property is required.*

Properties

In addition to the properties shared with email sending and execution of rules, message sending actions have specific properties.

Sender
Indicates a **Deyel** user that sends the message. Allows to use the autocomplete function to select it.

Receivers
Indicates **Deyel** users that will receive the message. It is possible to select users or agents that return users at the time of execution. In both cases the autocomplete facility can be used to select them.

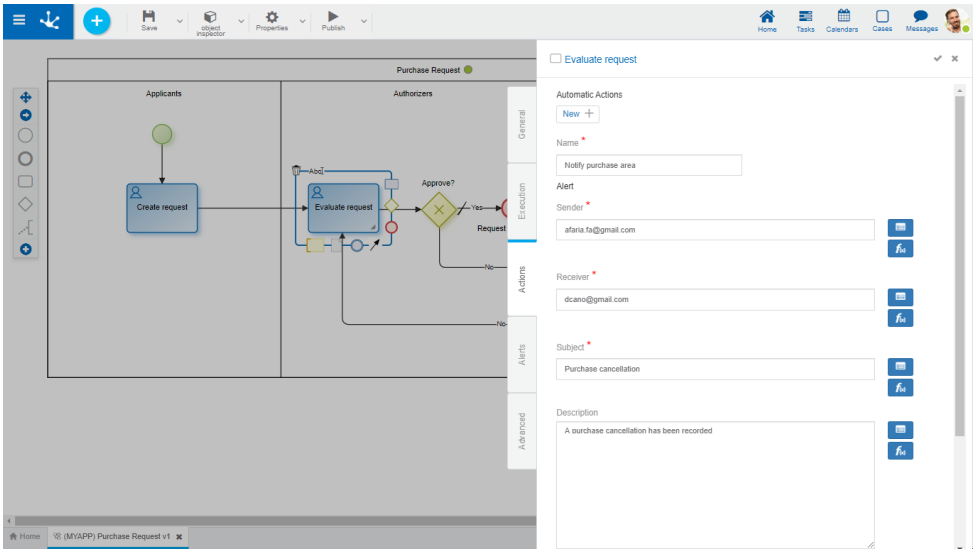
Message content

The text that makes up the body of the message is reported, this text can contain variables and functions. To select them, use their wizards of variables and functions.

> *For both senders and receivers, once the user has been selected, the chat facility can be used if the user is different from the one that is connected and the facility to display the user's profile .*

3.4.10.5.3.2. Send Email

Allows to define email sending. They can include process variables and functions that allow to incorporate links to case shows, to do lists, forms, application access links, among others.



> *An asterisk "*" on the label indicates that the property is required.*

Properties

In addition to the properties shared with email sending and execution of rules, message sending actions have specific properties.

In all properties, text can be combined with values obtained through the variables wizard or the function wizard of **Deyel**.

Sender

Indicates the email address from which the message is sent. If the email server is Google or Microsoft, it is not necessary to complete this property and the account entered in the environment property User or account to establish connection will become the sender.

Receiver

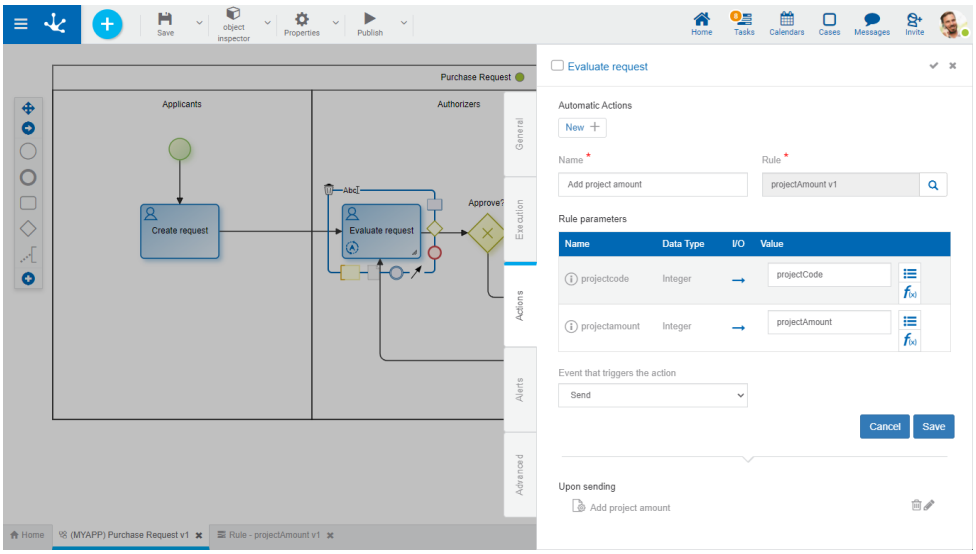
Indicates the email address to whom the message is sent. More than one address can be informed separated by ";".

Subject
Contains the subject of the email that is being sent.

Description
The text that makes up the body of the email is reported. Html tags can be included to format the message.

3.4.10.5.3.3. Execute Rule

Allows to define the execution of an advanced rule of **Deyel**.



An asterisk "" on the label indicates that the property is required.*

Properties

In addition to the properties shared with message sending and email sending, rule execution actions have specific properties.

Rule
Name of the advanced rule assigned to the automatic action. When creating the action, the rule is selected from a grid that contains the previously published rules.

Rule parameters
Depending on the selected advanced rule, a grid with its modeled parameters is enabled. For each parameter, report the properties:

Name
Parameter name as defined when modeling the advanced rule.

Data Type
Indicates the data type with which the parameter was defined. It can be any type of Java object allowed in **Deyel**. The Java object type can be displayed in full form, as a help, by placing the mouse over the data type in each grid element.

I/O
Specifies whether the parameter is input → or output ←.

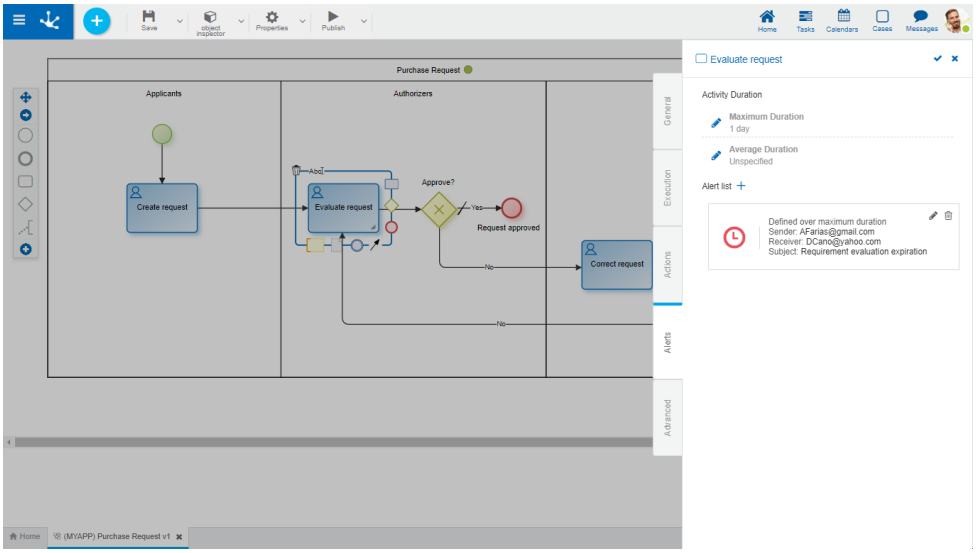
Value

If the parameter is input, it corresponds to the initial value taken by the parameter, by entering a fixed value or selecting it from the wizard of variables or functions. This value must match the one defined for the parameter.
If the parameter is output, it corresponds to the form field that will contain the value returned by the rule and can be selected by using the wizard of variables or functions.

> *The parameter show is available from the "i" icon.*

3.4.10.5.4. Alerts

For each standard activity or task, the IT modeler can define the issuance of electronic notifications by sending emails based on the activity execution times. They are defined in the "Alerts" tab of the activity properties panel.



Properties

Activity Duration

This section displays the Maximum Duration and the Average Duration that have been informed in the "General" tab. Alerts definition is performed considering any of these durations, as indicated in the property Define alert on.

Click on the icon to edit the duration and modify it.

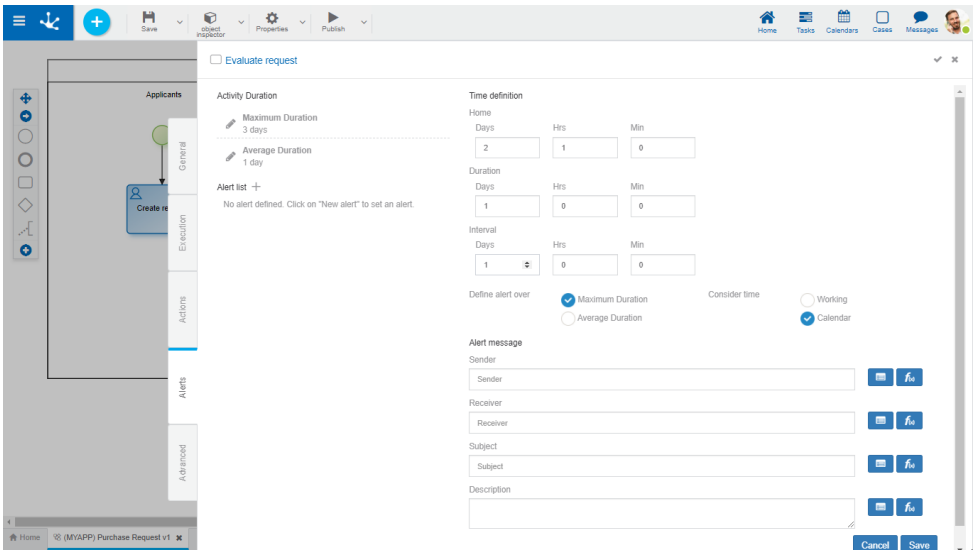
Alerts List
Displays the list of alerts set for the activity.

Indicates that the alert is set based on average duration.

Indicates that the alert is set based on maximum duration.

In order to define alerts, it is necessary to have entered the maximum or average duration of the activity.
To incorporate an alert, press the icon +. To modify it, press the icon and to delete it, press the icon .

Definition of Times



Alert start
Period of time that determines the start of the alert issuance, depending on the Maximum Duration or Average Duration established for the activity.
The time period can be measured either on a calendar time or working time basis, as specified in the property Consider time.

Days, Hours and Minutes values may:
• Not be informed: The alert is sent when the activity duration is fulfilled.
• Be informed with positive values: The alert is sent after the activity duration fulfillment.
• Be informed with negative values: The alert is sent before the activity duration fulfillment.

Alert duration
Period of time during which the alert is sent. The number of days, hours and minutes informed here allows to calculate the moment in which the alert sending must be finished as from the Alert start. The alert duration must be informed.
The time period can be measured either on a calendar time or working time basis, as specified in the property Consider time.

Alert interval
Time interval or frequency with which the alert is sent. Values are informed in Days, Hours and Minutes. If not informed, the alert will be sent every time the scheduled task is executed Execution of Activity Alerts.

The time period can be measured either on a calendar time or working time basis, as specified in the property Consider time.

Define alert on

The alert may be defined in relation to the Maximum Duration or Average Duration set in the "General" tab or from the same "Alerts" tab.

Consider time

Indicates that the periods set in the Start, Duration and Interval properties should be measured as calendar time or working time, taking into account the definition of environment calendars.
If working time is informed, the calendar defined for the participant responsible for executing the activity, if there is one, is considered first, or as a second option, the calendar defined for their organizational unit. When there are no special calendars, the standard environment calendar is considered.

Alert message

Sender

Email address from which the email is sent.
A value can be entered as long as the email server is not Google nor Microsoft.
Another possibility is to select the emailWorkflow() function, it is suggested when the email server is Google or Microsoft, and in this case it is not necessary to assign it a value in the property Value of the emailWorkflow() function because it will always take the address entered in the property User or account to establish connection.

Receiver

Receiver's email address.

Subject

Contains the email subject.
The field can be completed manually by combining it with variables available in **Deyel** using the icon  .

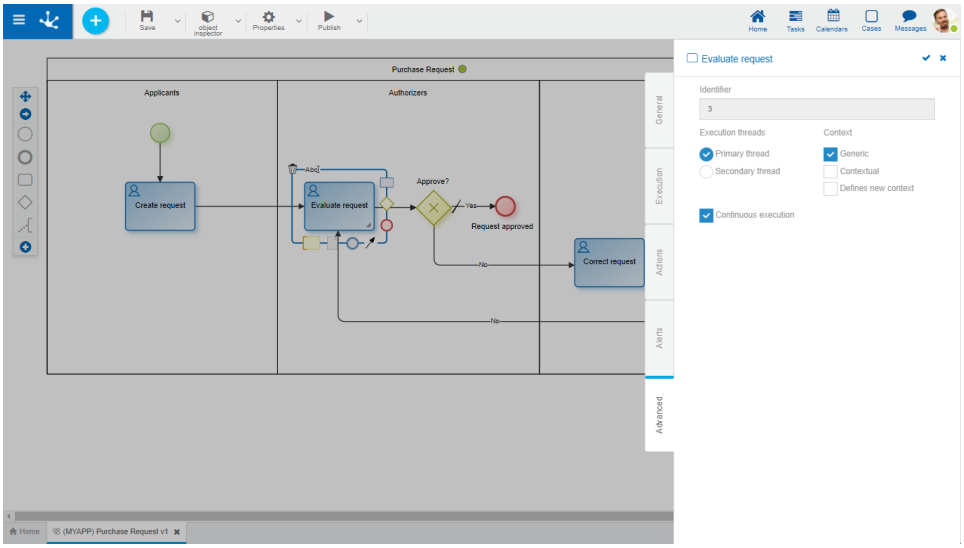
Description

It is the body of the email. The field can be completed manually and also combine it with variables available in **Deyel**, clicking on the icon  . Html tags can be used to format the message.

Click on the "Save" button to store the new alert that is incorporated into Alerts List, from where it can be modified or deleted.
Click on the "Cancel" button to close the new alert panel, discarding the changes.

3.4.10.5.5. Advanced

IT modeler users can set advanced level modeling properties in this tab.



Properties

Identifier

Activity code that identifies the activity univocally in the process. This attribute is automatically generated by the modeler and cannot be modified.

Execution Threads

Indicates if the activity is executed on the Main Thread of the case or on the Secondary Thread.

Context

A process execution context determines the universe of users the case can receive to execute its current task.
When those responsible for the activities are expressly indicated users, it is not necessary to select any of the context attributes.

Generic

This attribute can be selected when the lane responsible defined for the activity is a role or an agent.



When the case reaches the activity defined with Generic context, all the role/agent users receive the case in a generic way, without the possibility of execution, so that one of them can assign the case or the coordinator of the role can assign it to a particular user.
The moment the case is assigned to a user, it automatically disappears from the other users' task list. To assign the case, press the "Assign" button.



When the next activity has no Generic attribute defined, the list of role/agent users is submitted from the current activity, to select who the case is given to.
Note:If the current executor is included in the role/agent of the next activity and also the continuous execution, property is selected in the current activity, the next activity execution interface is automatically presented to the current user.

Contextual

When the first case activity is executed, the organizational unit of the user that executes it is established as the "contextual unit" of the case.

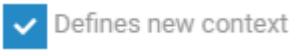


It can be selected in cases where the lane responsible in which the activity is located is defined with role per unit.
When the case reaches the activity with the Contextual attribute selected, it can only be executed by those users with role per unit and who also belong to the context unit or to some unit that depends on it. The Generic attribute should not be selected.



When the activity has no Contextual attribute selected, it can be executed by the expressly defined responsible users or by the role/agent, without applying any other restriction.

Defines New Context



When the activity execution begins, a new case execution context is established. This means that the context unit becomes the organizational unit to which the user who is executing the activity belongs. This option is used in very specific situations with a high degree of modeling experience.



The case execution context is not modified. It is the most used option.

Continuous Execution

This indicator is used to specify that after executing the activity it should automatically continue with the next one, whenever possible. In this way, the user can execute the case activities continuously, without accessing the to-do list.

Continuous execution is possible when:

- The responsible for the next activity is a user and it is the same user that executes the current activity.
- The responsible for the next activity is an organizational unit and it is the same organizational unit of the user that executes the current activity.
- The person responsible for the next activity is a role / agent and the user executing the current activity belongs to the role/agent.
  If in the next activity the Contextual attribute is checked, the organizational unit verifies that the current user is the context unit or a unit that depends on it.



According to the definition of subsequent activity, if the current user can execute it, then its execution starts automatically.

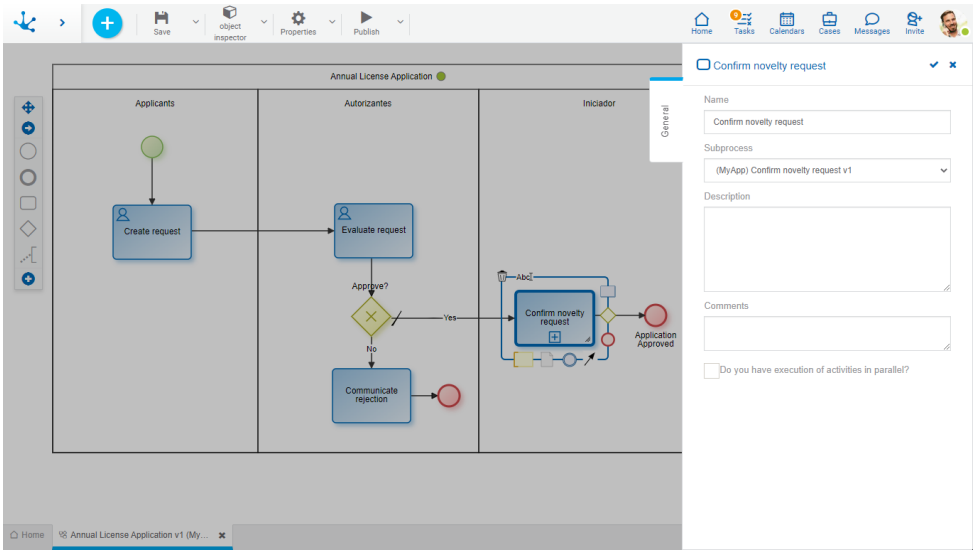


In this case, even though the current user can execute the next task, they must take the case from "My Tasks".

3.4.10.6. Subprocess Properties

There are different ways to enter a subprocess properties panel:

- Double click on the graphic element corresponding to the subprocess or abstract activity.
- Access the context menu of the subprocess graphic element.



General Tab

Properties

Name
Activity name. Describes in a generic way the objective of the process that is being represented.

Subprocess
Process represented by this abstract activity. Presents a list of all processes defined to select one of them.

Description
Conceptual description of the activities performed.

Comments
In this field the designer can document in detail what the subprocess execution consists of.

Do you have execution of activities in parallel?
This property should be checked if the selected subprocess has parallel or inclusive gateways modeled.

Actions

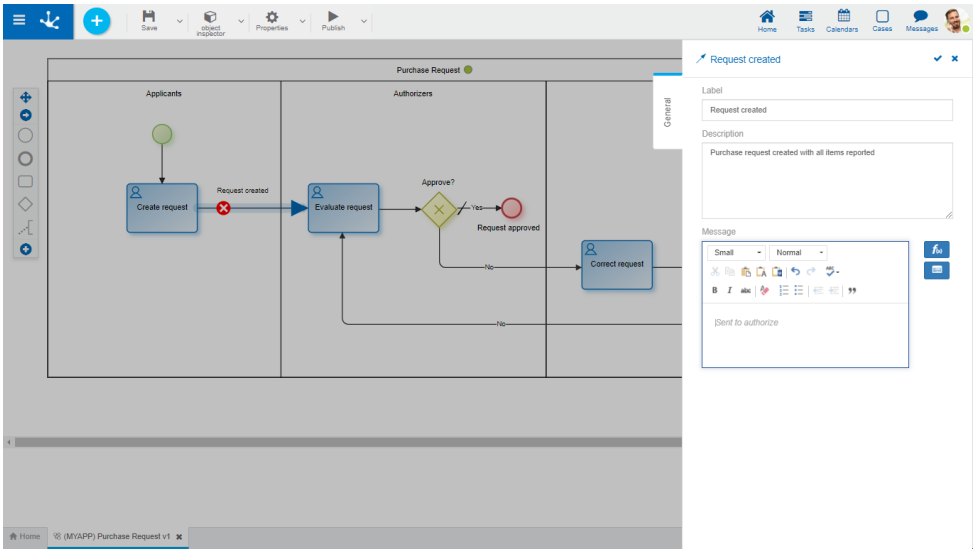
The icon ✔ is used to confirm the modifications made in the properties panel.

The icon ✖ is used to close the properties panel, if it was not previously saved, changes are discarded.

3.4.10.7. Flow Properties

There are different ways to enter a flow properties panel:

- Double click on the graphic element corresponding to the flow.
- Access the context menu of the flow graphic element.



General Tab

Properties

Label
Flow label.

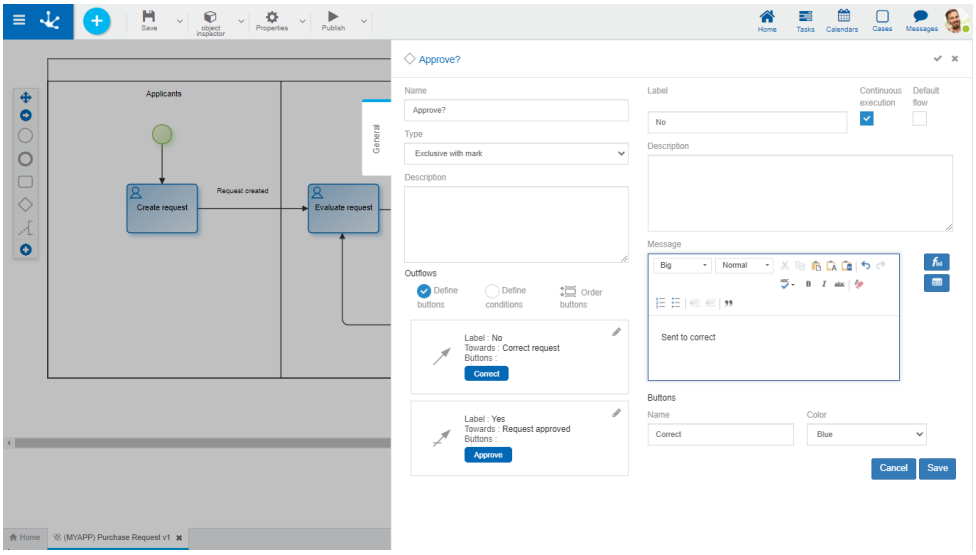
Description
Text field used to document the meaning of the flow.

Message
Text displayed when moving to the destination activity of the flow.

Allows to select functions of **Deyel** from a list to incorporate dynamic information into the message.

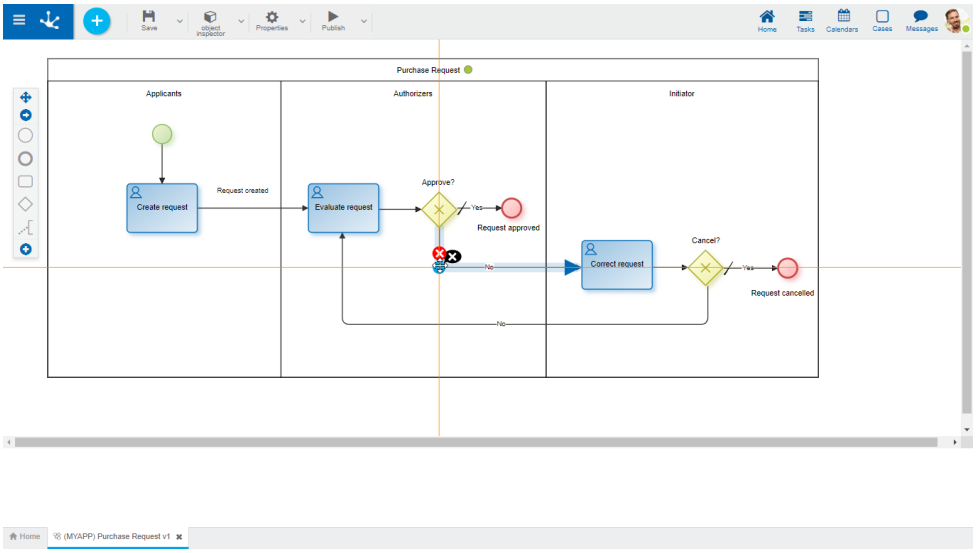
Allows to incorporate information on entities used in the process execution to the message.

When flow cases are gate outputs, in addition to flow properties, the panel displays the gate properties.



Visual Flow Options

These options are displayed when hovering over the graphic element.



Deletes the flow.

Indicates there is a vertex. Positioning the cursor over it enables a black cross that, when selected, will delete such vertex.

Deletes the vertex. Position the cursor over the blue circle to display the icon and then click on it.

To add a new vertex to the flow, keep the mouse button pressed until a blue point (vertex) is displayed. Once generated a new vertex, it can be moved with the Drag&Drop function.

Actions

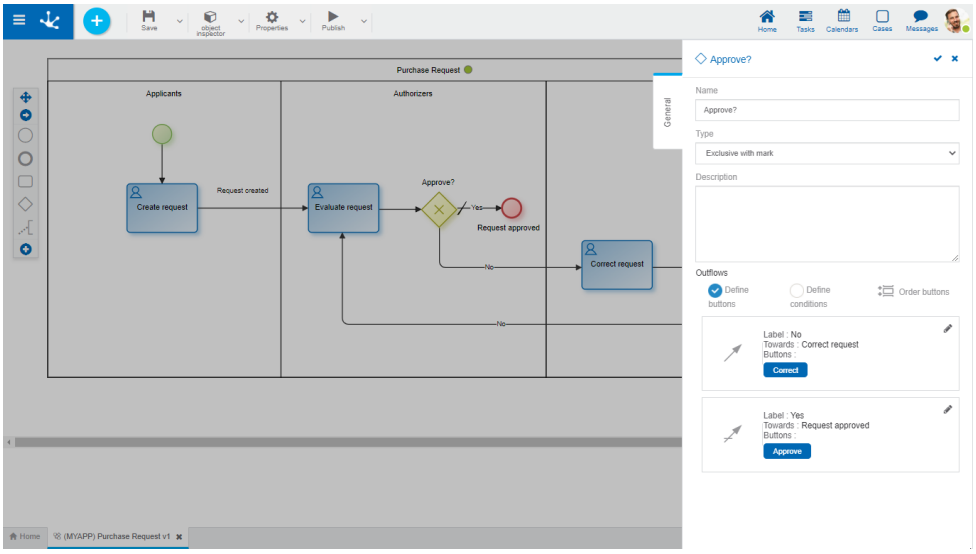
The icon ✔ is used to confirm the modifications made in the properties panel.

The icon ✖ is used to close the properties panel, if it was not previously saved, changes are discarded.

3.4.10.8. Gateway Properties

There are different ways to enter the gateway properties panel:

- Double click on the graphic element corresponding to the gateway.
- Access the context menu of the gateway graphic element.



General Tab

Properties

Name
Gateway name that should reflect what is being evaluated.

Type
Indicates the gateway type.

Description
Text that describes the gateway purpose.

Output flows

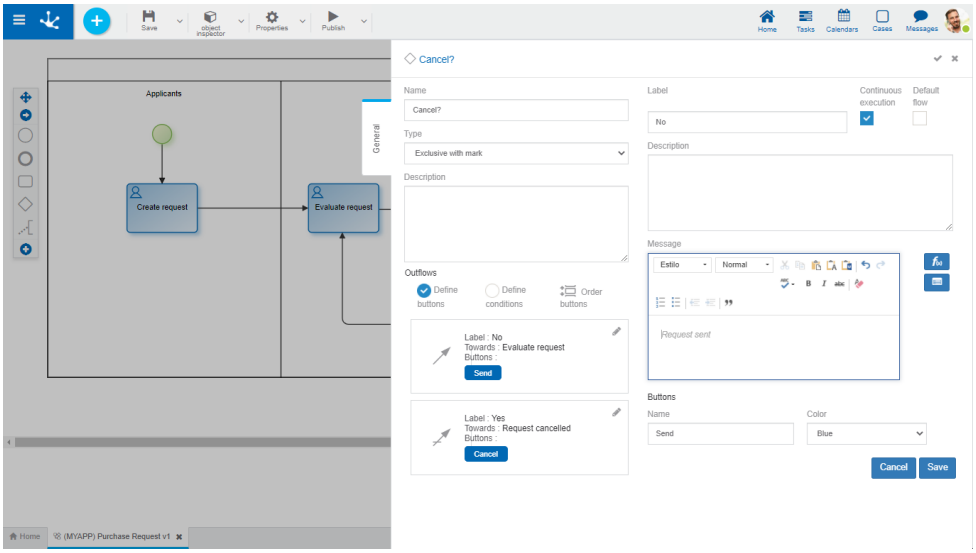
Default Flow
Allows to define a default flow indicating which flow to follow when other flow conditions are not met. This property is required.

Continuous Execution

Allows to define the continuous execution of each output flow of the exclusive gateways.

Define buttons

Allows to assign a button for the user to push at each gateway output flow. If the gateway is inclusive, more than one button can be assigned to each flow, and each button may have more than one flow assigned.
Buttons become visible in the execution activity previous to the gateway.
A name and a color are defined for each button .



Displays a properties panel that allows to establish the button name and color assigned to the flow, and also modifies other flow properties.
Within this panel, if it is an inclusive gateway, the following possibilities are enabled.

New button

Allows to add a new button to the flow.

Select assistant bot

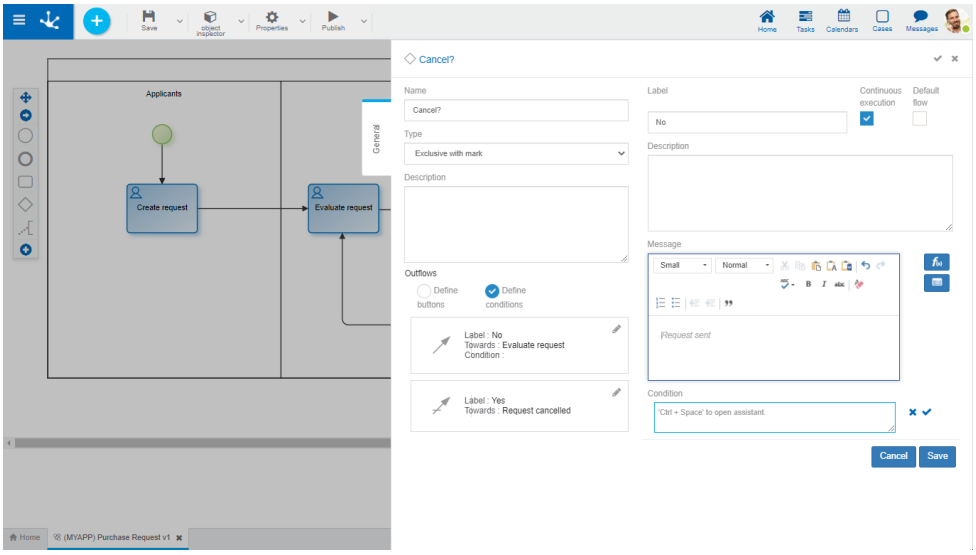
Allows to assign an existing button to the flow.

Order buttons

This functionality is enabled when you select Define buttons. Clicking on it, displays a panel to order buttons with the Drag&Drop function.

Define conditions

Allows to assign conditions to be met so that the process continues its execution following the flow.



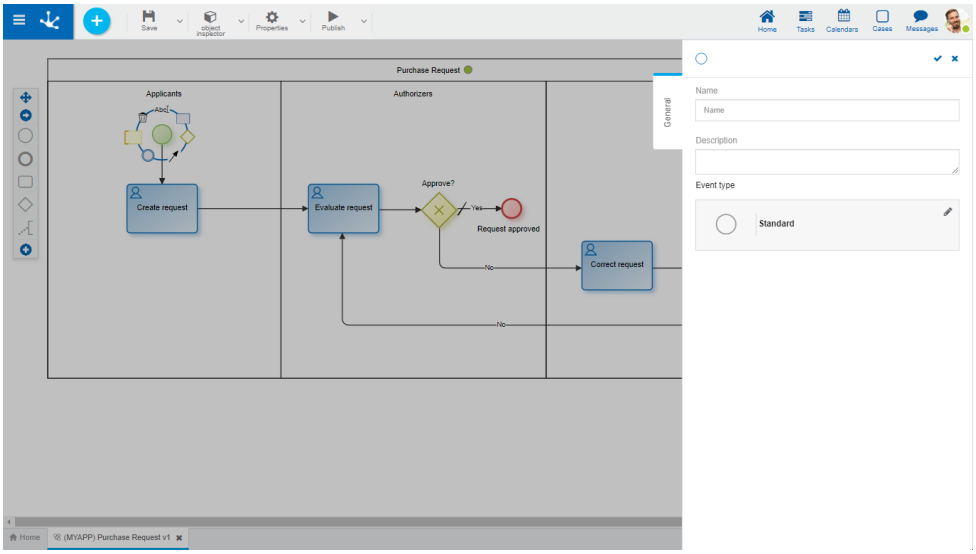
Displays a properties panel where the flow conditions are modeled by means of the embedded rules wizard that allows to select fields of entities that participate in the process and operations to be applied. Additionally, it is also possible to modify other flow properties.

> *If the process is related to several forms, the wizard allows to select the form fields indicated as main entity in form properties.*
> *If an entity is not defined as the main entity, the first form fields displayed when expanding the list in the Execution Definition property can be selected in the process activities.*

3.4.10.9. Event Properties

There are different ways to enter the properties panel an event:

- Double click on the graphic element corresponding to the event.
- Access the context menu of the event graphic element.



Actions

The icon ✔ is used to confirm the modifications made in the properties panel.

The icon ✖ is used to close the properties panel, if it was not previously saved, changes are discarded.

3.4.10.9.1. Starts Events

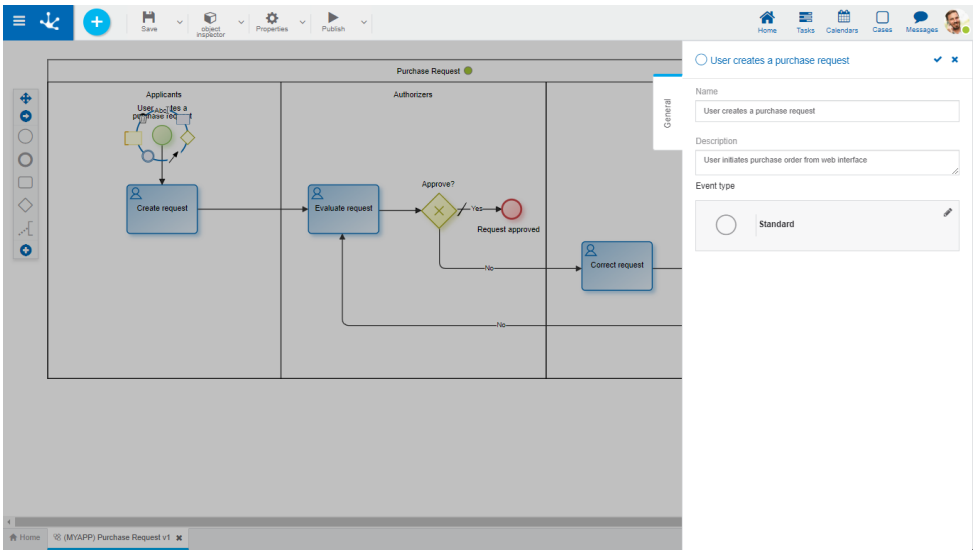
General Tab

In the "General" tab, common properties for all start events and the special ones for each case are defined, which are enabled when assigning the type of start event.

- Standard
- Command
- Email
- Signal
- Timer
- File
- Rule

3.4.10.9.1.1. Standard

The properties panel of the standard start event is displayed on the right side of the process modeler.



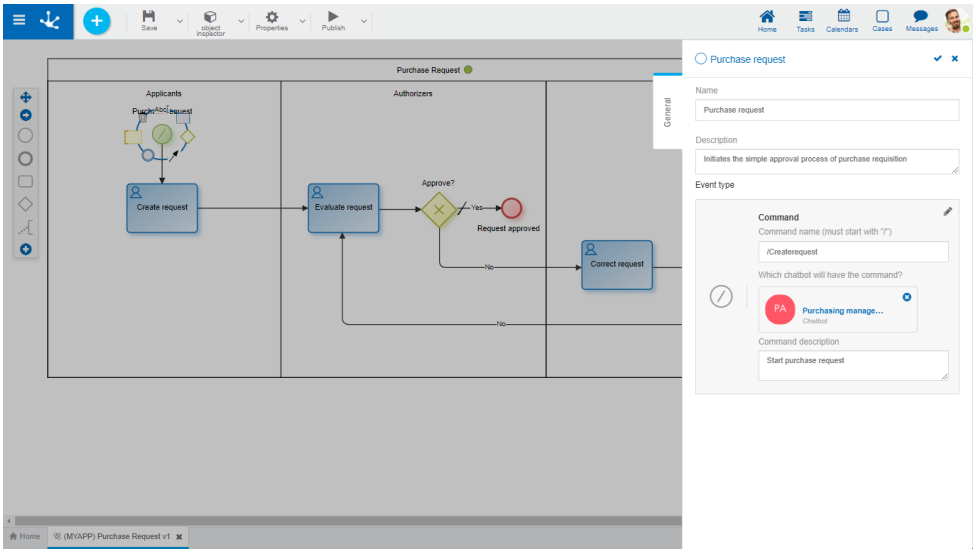
Properties

Name
Text that is displayed in the diagram next to the graphic element of the event.

Description
Field that allows to document the event.

3.4.10.9.1.2. Command

The properties panel of the start event by command is displayed on the right side of the process modeler.



Properties

Name
Text that is displayed in the diagram next to the graphic element of the event.
It is recommended to provide a text that identifies the command.

Description
Field that allows to document the event.

Command
Text assigned by the user to identify the command that starts the process. MIt must start with a slash "/" and followed by the name the user informed, with no blank spaces.
By default it presents the command established in the process creation. It can be modified.

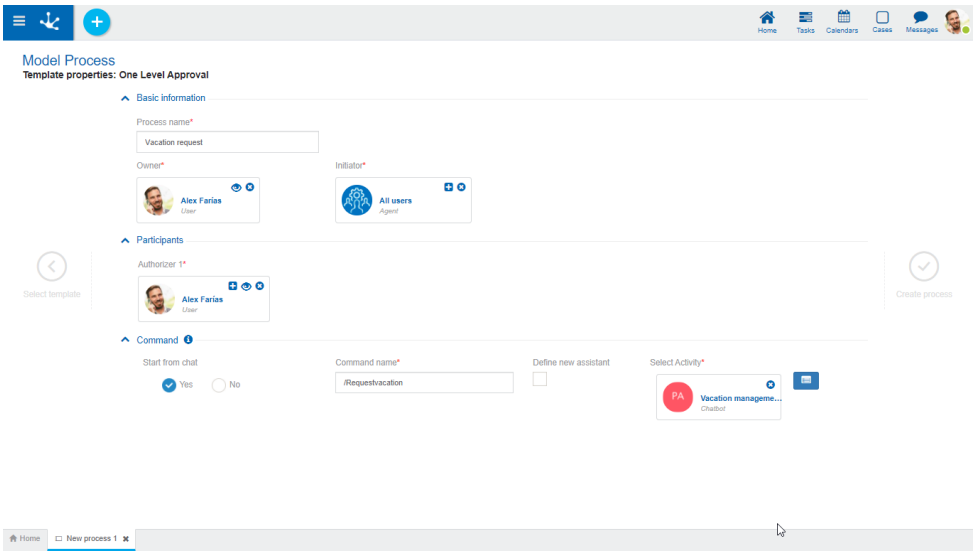
Chatbot
Indicates the chatbot used to execute the command.
By default, the chatbot informed in the process creation is displayed, and besides, if the Define new assistant process property is checked, a new chatbot is automatically created, configuring itself as wizard of the user that is modeling the process.
To modify the assigned chatbot, click on the delete icon. This property has autocomplete function, therefore as characters are entered, the first five matches are dynamically displayed to select one of them.

Command Description

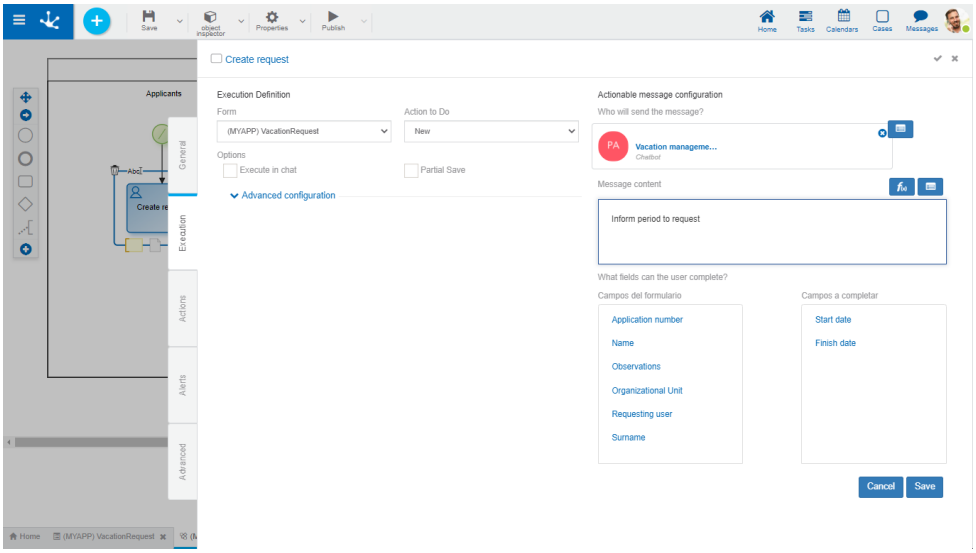
Explanatory text that is displayed when selecting the command from the chat window.

Example

The command start event is automatically defined when the process property is checked Start from Chat, when modeling a new process, whether from a template or not.



To modify the type in an already defined start event, access the context menu of the graphic element and select the type of start event from the options that are presented.
The first activity of a process that starts with a command event, must have the Execute on Chat property checked.



Properties

Who will send the message?

The user or chatbot that sends the message from the chat is informed. It generally matches the chatbot assigned in the command start event.
Note that if a user is informed, that user does not see their own message.

Message content

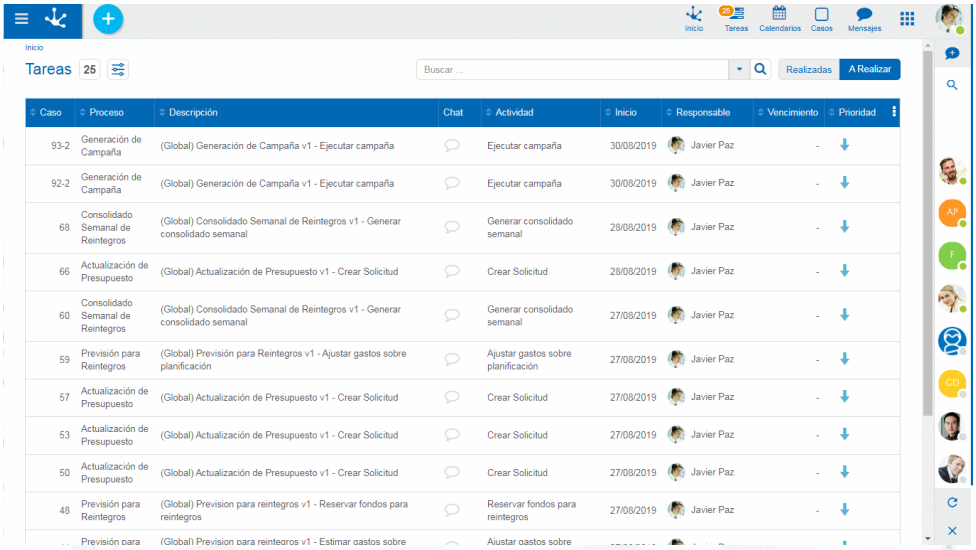
Inform the text displayed in the chat at the beginning of the process.

What fields can the user complete?

Move the fields that the user must complete from the chat to the fields box to complete, in the first process activity.

Use of the Process

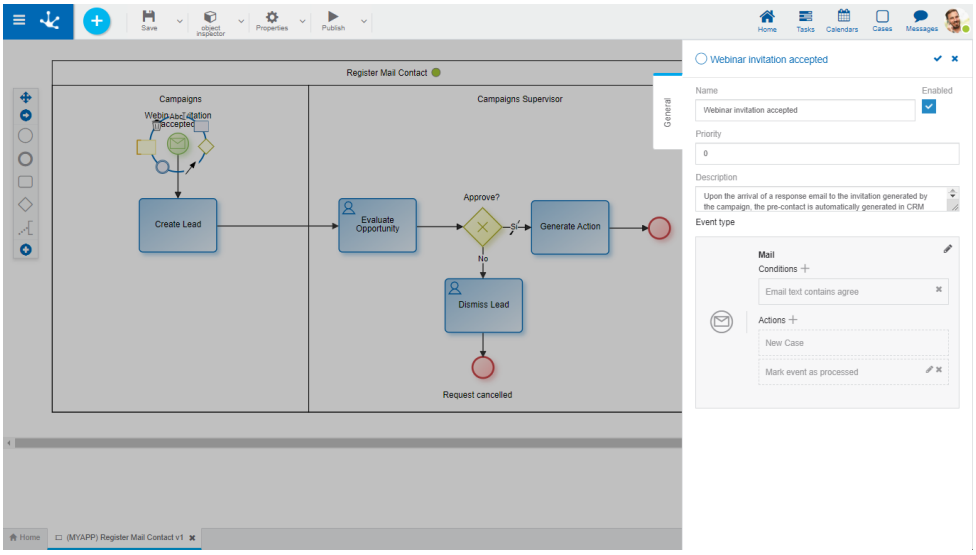
- To start the process by chat, the user must have permission to start the process.
- If the command is assigned to a chatbot configured as "visible", users can initiate the command directly from the chatbot in a chat.
- If the chatbot with the command is configured as "not visible", then necessarily some user must be using it as wizard. A chat with that user must be opened to access the command.



In the chat window, the data required in the form must be informed and one of the proposed buttons of the actionable message must be pressed.

3.4.10.9.1.3. Email

The properties panel of the start event by email is displayed on the right side of the process modeler.



Properties

Name

Text that is displayed in the diagram next to the graphic element of the event.
It is recommended to inform a text in reference to the email expected to be received.

Enabled

Indicates that the event is enabled to receive emails and execute if applicable.

Priority

Execution priority assigned to the event. It is reported as an integer, with 0 being the highest priority. If the event performs the action mark event as processed, events with lower priority are not executed.

Description

Text that allows documenting detailed information about the event.

Conditions

They are the conditions that must be met for the event to be executed. If there are defined conditions, a list of them is displayed.

To add a condition, press the icon + and from the panel that opens, select Property and Operator in the corresponding drop-down lists. Report Value and where applicable, indicate in Parameter the field to be evaluated of the form associated with the process.

Property	Example
• Email subject • Email text	Email subject **contains** "Information request"
• Email sender • Email recipient	Email sender **Equals** AtencionConsultas@Deyel.com *More than one address can be report-*

Property	Example
	ed separated by ; (semicolon) without leaving spaces
• Name without the attachment extension in n position (starts at 1) • Attachment extension in n position (starts at 1)	Name without the attachment extension in n position **starts with** "Sheet", indicated in parameter 1 (first position)
• Number of email days elapsed	Number of days elapsed **Equals** 3

Actions

They are the actions that are carried out automatically when the event occurs.

- Create Case

For all start events, a case is compulsorily created for the process and a version in which the event is executed.

- Attach file

Allows to attach one or more files to the case.

Parameter	Description
Email attachment	Position number of the email attachment that is attached to the case. (*)
Depends on the previous execution	Indicates if the action is executed only when the previously defined actions were executed successfully.(*)

> *Note: The parameters selected with (*) are mandatory.*

- Attach form

Allows to relate a form to the case.

Parameter	Description
User Code	If it is not reported, the current user is assigned.
Values map	Report <field1><value1><field2><value2>…, where fieldN corresponds to form field names and valueN to the values that are assigned to each field. It has higher priority than

Parameter	Description
	DS_XML_CONTENT_PATH
XML	File path that contains the XML. It is mandatory if DS_VALUE_MAP is not reported
Annex Type	Annex type attached (*).
Create version if it exists and if it is unique	If when attaching the form, it already exists and is unique, a historical version of it is previously created.
Reset previous form and it is unique	Indicates that if there is a form associated with the case and it is unique, data is deleted from it.
Depends on the previous execution	The action is executed only if the previously defined actions were executed successfully.(*).

> *Note: The parameters marked with (*) are mandatory.*

- Check event as processed

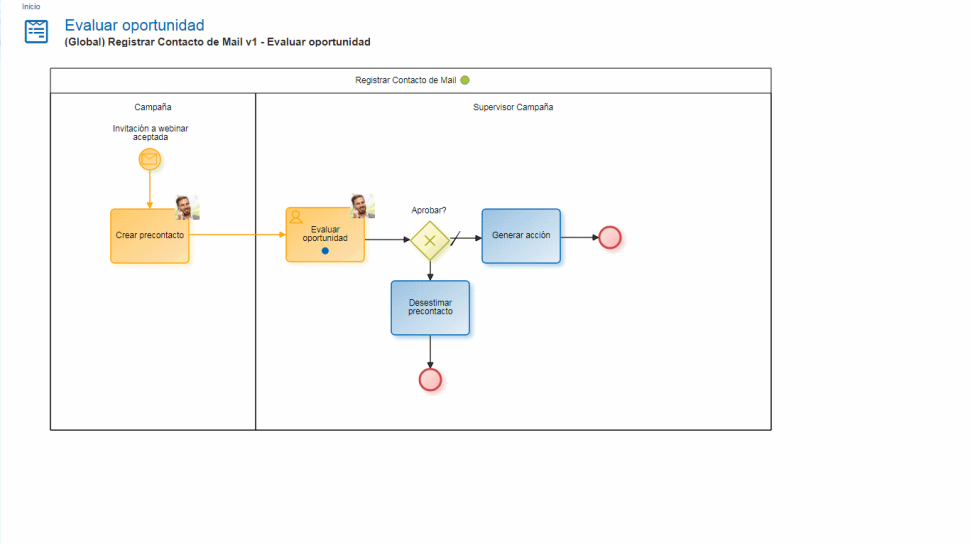
An automatic response is sent to the email received indicating that it has been processed.

Parameter	Description
Depends on the previous execution	Indicates if the action is executed only when the previously defined actions were executed successfully (*).

> *Note: The parameters marked with (*) are mandatory.*

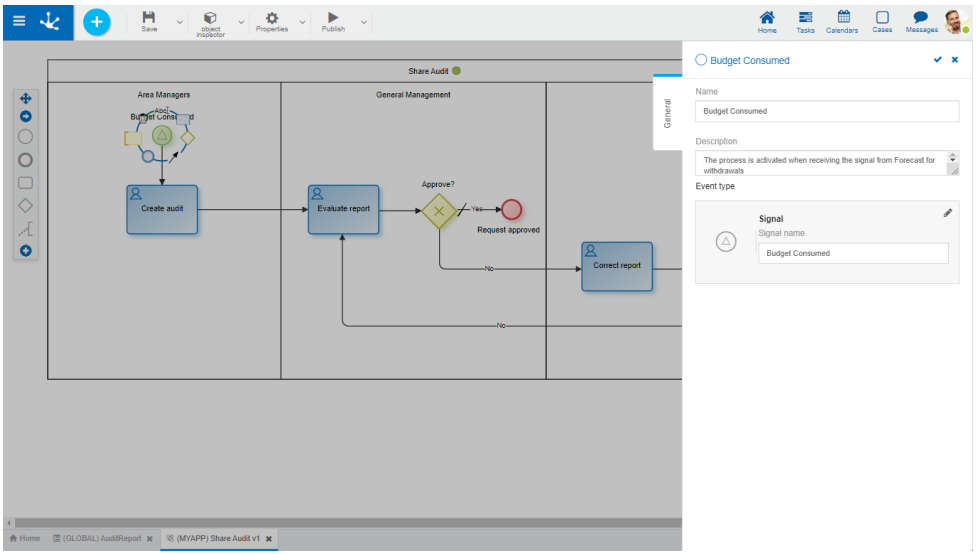
Example of Use

The "Register Email Contact" process starts its execution automatically when an email arrives to the email account established in the environment configuration, under the title "Invitation to **Deyel** Webinar" and the word "Agree" is in the content.



3.4.10.9.1.4. Signal

The properties panel of the start event by signal is displayed on the right side of the process modeler.



Properties

Name

Text that is displayed in the diagram next to the graphic element of the event.
It is recommended to inform a text in reference to the signal expected to be received.

Description

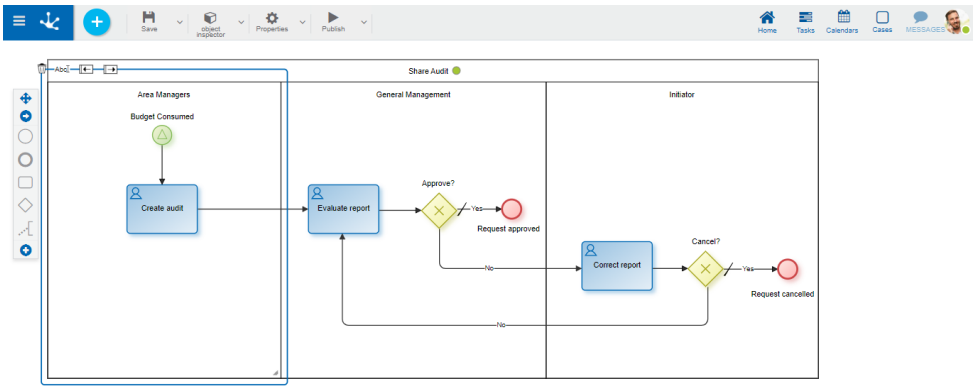
Text that allows documenting detailed information about the event.

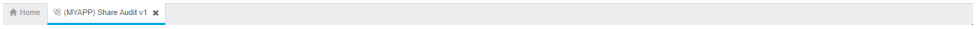
Signal Name

Signal that the event expects to receive to start execution automatically. The signal is sent by a signal end event or a throw signal intermediate event.
There may be more than one process waiting for the same signal.

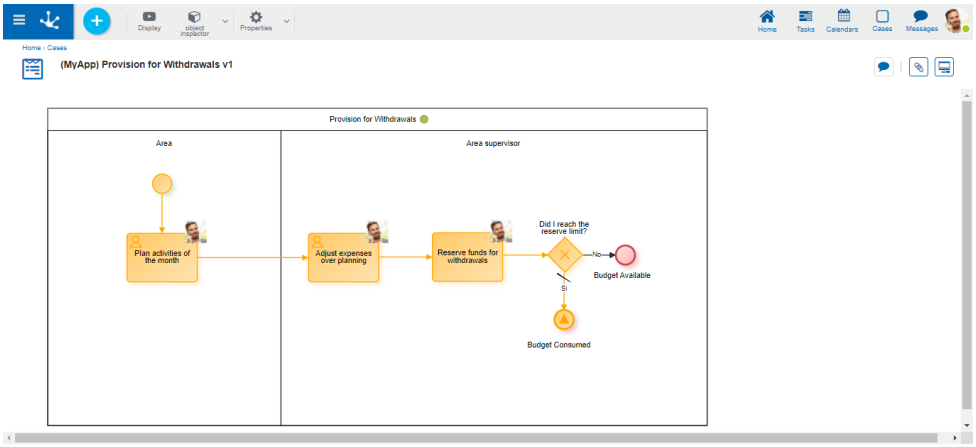
Example of Use

The audit process starts automatically upon receiving the "Budget Consumed" signal sent by the "Provision for Refunds",process in one of its possible endings.



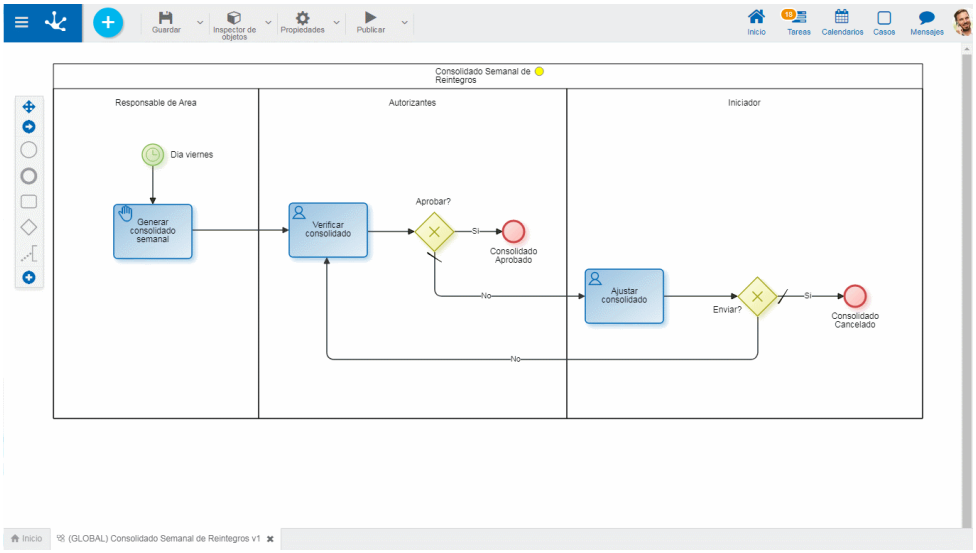


"Provision for Refunds" process in which the "Consumed Budget" signal is generated.



3.4.10.9.1.5. Timer

The properties panel of the [timer start event] is displayed on the right side of the process modeler.



Properties

Name

Text that is displayed in the diagram next to the graphic element of the event.
It is recommended to inform a text in reference to the event scheduling.

Enabled

This property is checked to indicate that an event is active and can be automatically executed according to its scheduling.

Description

Text that allows documenting detailed information about the event.

Conditions

They are the conditions that must be met for the event to be executed. If there are defined conditions, a list of them is displayed.

To add a condition, press the icon + and from the panel that opens, select Property and Operator in the corresponding drop-down lists. Report Value and where applicable, indicate in Parameter the field to be evaluated of the form associated with the process.

For this type of event, apply the following conditions:

- Current time
- Current day
- Current month
- Current year
- Current hour
- Current minute
- Current seconds
- Current milliseconds
- Weekday

How often should it execute?

Type

Allows to define the execution schedule for the event

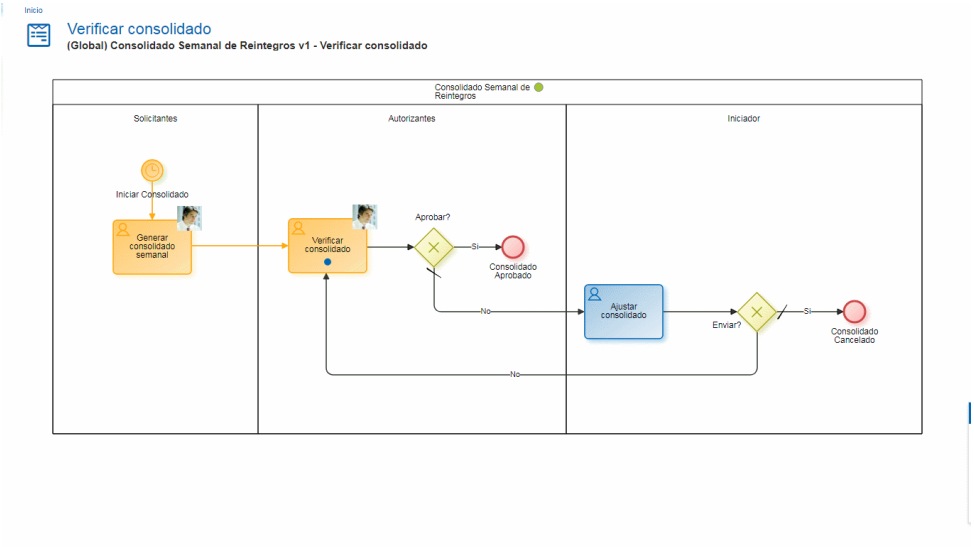
- Regular intervals
- Daily at a specific time

Value

Allows to define the interval value in seconds or at a specific time, according to the previously defined type.
The "Save schedule" button must be pressed to keep the established schedule. It can be seen on the Scheduled Tasks Monitor.

Example of Use

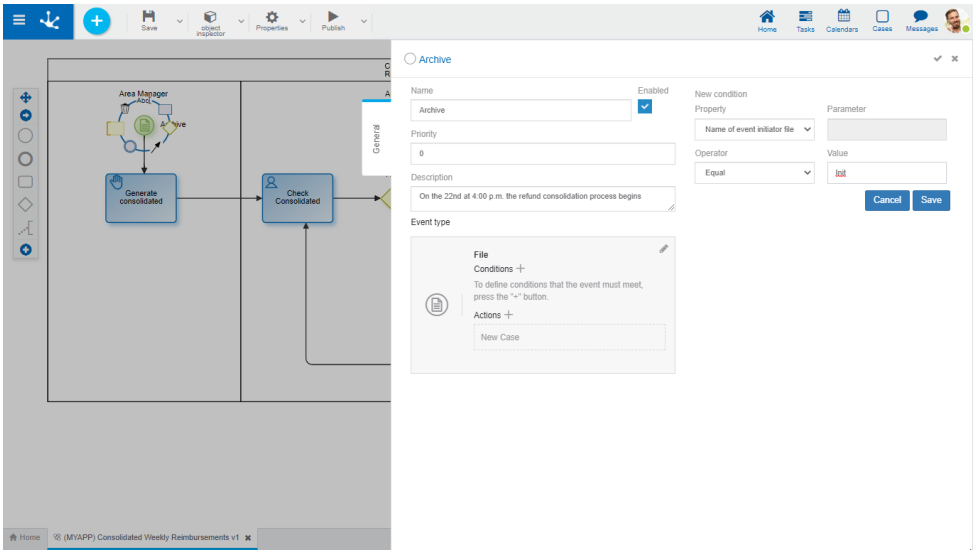
The "Consolidated Weekly Refund" process begins its execution automatically on Fridays at 10:00.



3.4.10.9.1.6. File

This type of event is available only for on-premises installations, as they require access to resources on the client's computer.

The properties panel of the start event per file is displayed on the right side of the process modeler.



Properties

Name
Text that is displayed in the diagram next to the graphic element of the event.
It is recommended to inform a text in reference to the email expected to be received.

Enabled
This property is checked when the event activates, waiting for the file.

Priority
Execution priority assigned to the event. It is reported as an integer, with 0 being the highest priority. If the event performs the action mark event as processed, events with lower priority are not executed.

Description
Text that allows documenting detailed information about the event.

Conditions
They are the conditions that must be met for the event to be executed. If there are defined conditions, a list of them is displayed.
To add a condition, press the icon + and from the panel that opens, select Property and Operator in the corresponding drop-down lists. Report Value and where applicable, indicate in Parameter the field to be evaluated of the form associated with the process.

Property	Example
Name of the event initiator file, without the extension	Name of the event initiator file **Contains** "Ticket"
Name of the folder that contains the event initiator file	Name of the folder that contains the event initiator file **Equals** "Receipt of Tickets"
File extension (without dot)	File extension **Equals** XML

Property	Example
Event initiator file path	Event initiator file path **Equals** "C:\Claims\Tickets Attention"
Folder path that contains the event initiator file	Folder path that contains the event initiator file **Equals** "D:\Claims Attention"
Number of file days elapsed	Number of file days elapsed **Greater than or Equal to** 9
File type (File or Folder)	

Actions

They are the actions that are carried out automatically when the event occurs.

- Create Case

For all start events, a case is compulsorily created for the process and a version in which the event is executed.

- Attach file

Allows to attach a file to the case.

Parameter	Description
Depends on the previous execution	Indicates if the action is executed only when the previously defined actions were executed successfully.(*)

> *Note: The parameters selected with (*) are mandatory.*

- Attach form

Allows to relate a form to the case.

Parameter	Description
User Code	If it is not reported, the current user is assigned.
Values map	Report <field1><value1><field2><value2>…, where fieldN corresponds to form field names and valueN to the values that are assigned to each field. It has higher priority than DS_XML_CONTENT_PATH

Parameter	Description
XML	File path that contains the XML. It is mandatory if DS_VALUE_MAP is not reported
Annex Type	Annex type attached (*).
Create version if it exists and if it is unique	If when attaching the form, it already exists and is unique, a historical version of it is previously created.
Reset previous form and it is unique	Indicates that if there is a form associated with the case and it is unique, data is deleted from it.
Depends on the previous execution	The action is executed only if the previously defined actions were executed successfully.(*).

> *Note: The parameters marked with (*) are mandatory.*

- Check event as processed

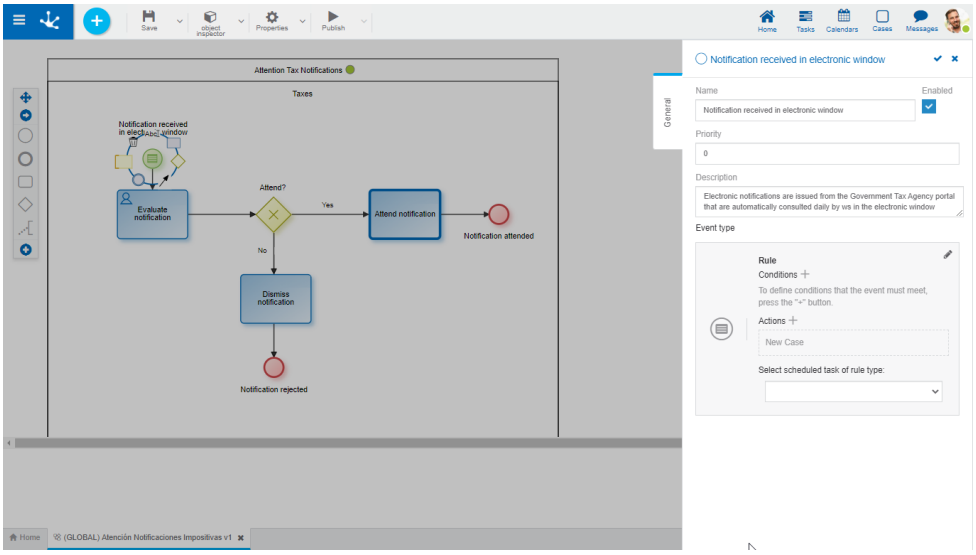
An automatic response is sent to the email received indicating that it has been processed.

Parameter	Description
Depends on the previous execution	Indicates if the action is executed only when the previously defined actions were executed successfully (*).

> *Note: The parameters marked with (*) are mandatory.*

3.4.10.9.1.7. Rule

The properties panel of the start event by rule is displayed on the right side of the process modeler.



Properties

Name
Text that is displayed in the diagram next to the graphic element of the event.
It is recommended to inform a text in reference to the rule whose execution is expected.

Enabled
This property is checked to indicate that the event activates, waiting for the rule.

Priority
Execution priority assigned to the event. It is reported as an integer, being 0 the top priority.

Description
Text that allows documenting detailed information about the event.

Conditions
They are the conditions that must be met for the event to be executed. If there are defined conditions, a list of them is displayed.

To add a condition, press the icon + and from the panel that opens, select Property and Operator in the corresponding drop-down lists. Report Value and where applicable, indicate in Parameter the field to be evaluated of the form associated with the process.

Actions
They are the actions that are carried out automatically when the event occurs.

- Create Case

For all start events, a case is compulsorily created for the process and a version in which the event is executed.

- Attach file

Allows to attach a file to the case.

Parameter	Description

Parameter	Description
File to attach	File attached to the case.
Depends on the previous execution	Indicates if the action is executed only when the previously defined actions were executed successfully.(*)

> *Note: The parameters selected with (*) are mandatory.*

- Attach form

Allows to relate a form to the case.

Parameter	Description
User Code	If it is not reported, the current user is assigned.
Values map	Report <field1><value1><field2><value2>…, where fieldN corresponds to form field names and valueN to the values that are assigned to each field. It has higher priority than DS_XML_CONTENT_PATH
XML	File path that contains the XML. It is mandatory if DS_VALUE_MAP is not reported
Annex Type	Annex type attached (*).
Create version if it exists and if it is unique	If when attaching the form, it already exists and is unique, a historical version of it is previously created.
Reset previous form and it is unique	Indicates that if there is a form associated with the case and it is unique, data is deleted from it.
Depends on the previous execution	The action is executed only if the previously defined actions were executed successfully.(*).

> *Note: The parameters marked with (*) are mandatory.*

Select scheduled rule task

The executed business rule is selected. It must be a previously published advanced rule.

Example of Use

The "Attention to Tax Notifications" process starts automatically when the business rule associated with the rule start event detects a notification in the government portal, following the WS specifications published in it.

3.4.10.9.2. Intermediate Events

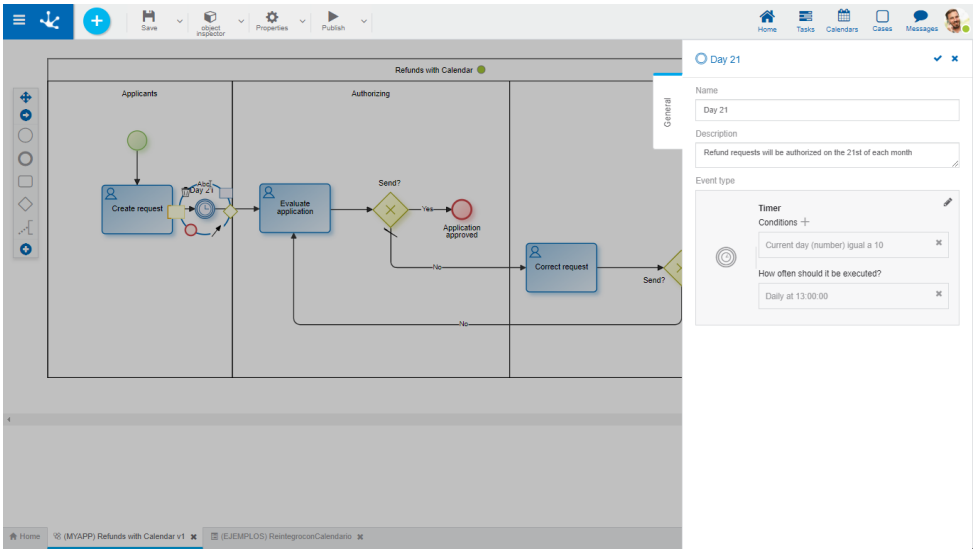
General Tab

In the "General" tab, common properties for all intermediate events and the specific ones for each case are defined, which are enabled when assigning the type of intermediate event.

- Timer
- Throw and Catch a Signal
- Throw and Catch Link

3.4.10.9.2.1. Timer

The properties panel of the timer intermediate event is displayed on the right side of the process modeler.



Properties

Name
Text that is displayed in the diagram next to the graphic element of the event.
It is recommended to inform a text in reference to the event scheduling.

Description
Text that allows documenting detailed information about the event.

Conditions
They are the conditions that must be met for the event to be executed. If there are defined conditions, a list of them is displayed.

To add a condition, press the icon + and from the panel that opens, select Property and Operator in the corresponding drop-down lists. Report Value and where applicable, indicate in Parameter the field to be evaluated of the form associated with the process.

Property	Example
• Current time • Current day (numeric) • Current month (numeric) • Current year (numeric)	Current time **Equals** 13:30 Current month **Less than or equals** 6 *Time format is HH:MM. Supports one-digit values for the time.* *When showing the current month, a number between 1 and 12 must be informed in value.*
• Current hour (numeric) • Current minute (numeric) • Current seconds (numeric) • Current milliseconds	Current time (numeric) **Equals** 1.50 Current milliseconds **Greater** 48600000
• Day of the week (Sunday= 1)	Day of the week equals 6 (6 = Friday) *An integer between 1 and 7 must be informed in value, where 1 corresponds to Sunday.*
• DateTime parameter (*) • Date parameter (*) • Time parameter (*)	Inform in Parameter a form field that is converted to date/time/date time for its evaluation *The date format is DD/MM/YYYY.* *The datetime format is DD/MM/YYY HH:MM, with a blank space.* *Time format is HH:MM.* *Both formats support one-digit values for day and month.*
• Working days elapsed since the date (*) • Years elapsed since the date (*) • Months elapsed since the date (*) • Days elapsed since the date (*)	Inform in Parameter a specific date in DD/MM/YYYY format or a form field that contains a date *For these properties an integer value must be informed.*

Property	Example
• Hours elapsed since the “time” hour (*) • Minutes elapsed since the "time" hour (*) • Seconds elapsed since the "time" hour (*)	Inform in Parameter a specific time in DD/MM/YYYY format or a form field that contains a time *For these properties an integer value must be informed.*

(*) Properties available only for intermediate and border events

How often should it execute?

Type

Allows to define the execution schedule for the event

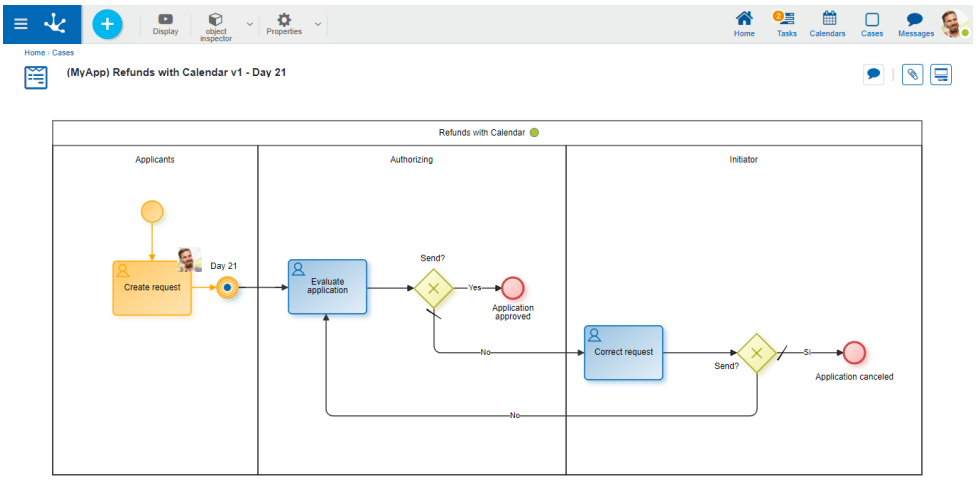
- Regular intervals
- Daily at a specific time

Value

Allows to define the interval value in seconds or at a specific time, according to the previously defined type.
The "Save schedule" button must be pressed to keep the established schedule. It can be seen on the Scheduled Tasks Monitor.

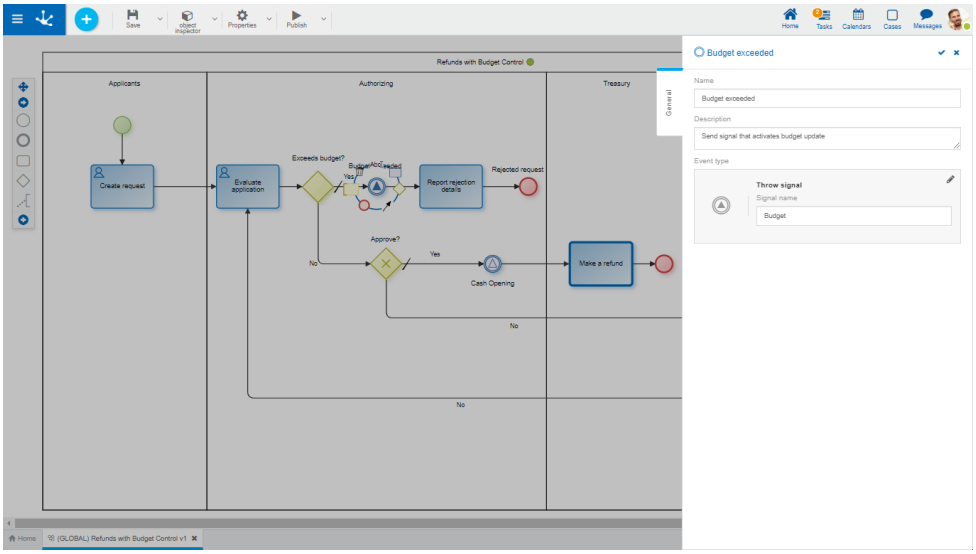
Example of Use

The "Refund Request with Calendar" process begins at the request of users. The authorizer receives requests not until day 21.



3.4.10.9.2.2. Throw and Catch a Signal

The properties panel of the throw and catch signal intermediate events, is displayed on the right side of the process modeler.



Properties

Name

Text that is displayed in the diagram next to the graphic element of the event.
It is recommended to inform a text in reference to the signal expected to be received or sent.

Description

Text that allows documenting detailed information about the event.

Signal Name

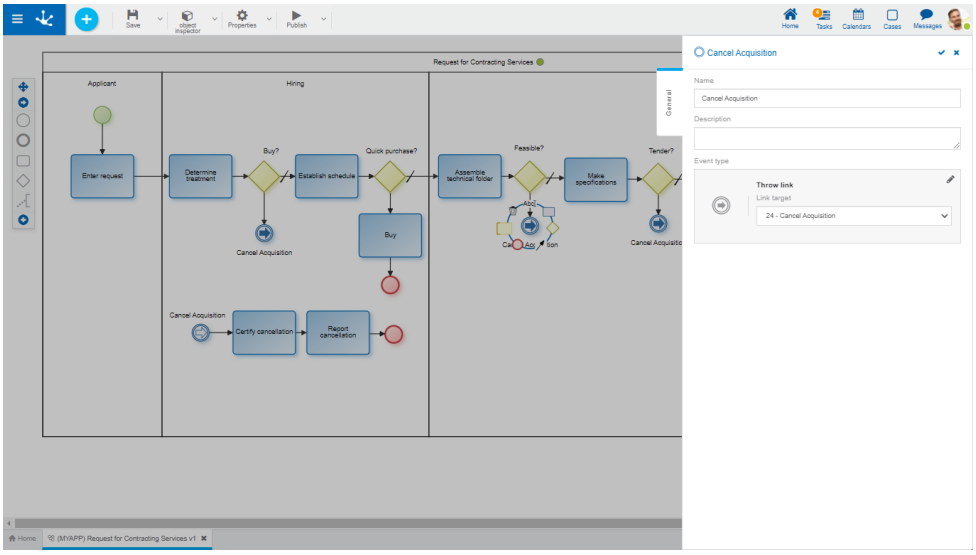
Signal identifier that the event expects to receive or sends in order to continue executing the case automatically, depending on whether it is a catch or a launch event. A character string is specified. There may be more than one process waiting for the same signal.

Example of Use

The "Refund with Budget Control" process sends a signal to automatically start the "Review Budgeting" process, when the established budget has been exceeded.
Approved refund requests await a treasury signal that enables the cash flow.

3.4.10.9.2.3. Throw and Catch Link

The properties panel of the throw link and capture link intermediate events, is displayed on the right side of the process modeler.



Properties

Name

Text that is displayed in the diagram next to the graphic element of the event.
If it is a catch link event, it is also the identifier used to reach it, from the throw link event.

Description

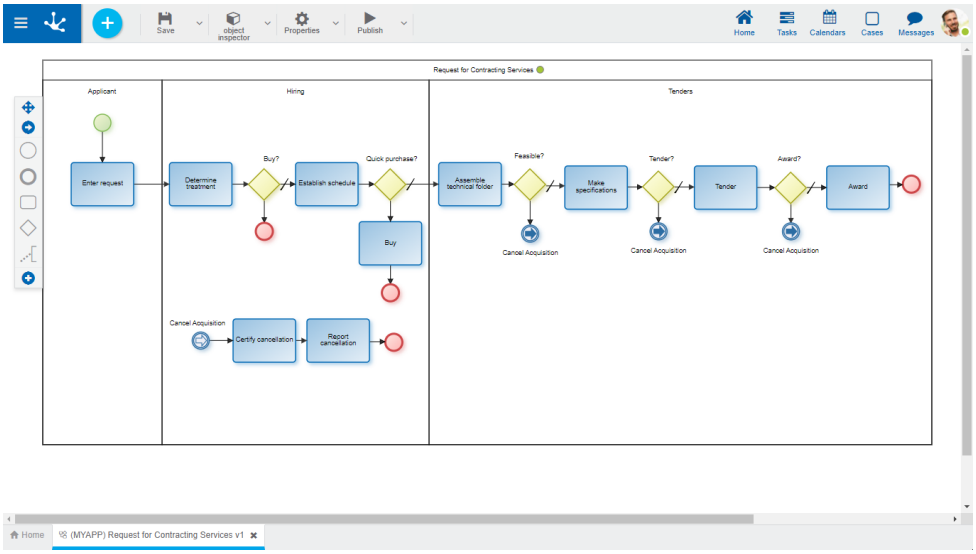
Text that allows documenting detailed information about the event.

Link Destination

This attribute corresponds to the throw link event. Allows to select the event type catch link to go to.

Example of Use

The "Request for Contracting Services" process, to facilitate its interpretation, presents several links to the "Cancel Acquisition" event. These events are particularly useful in complex diagrams.



3.4.10.9.3. Border Event

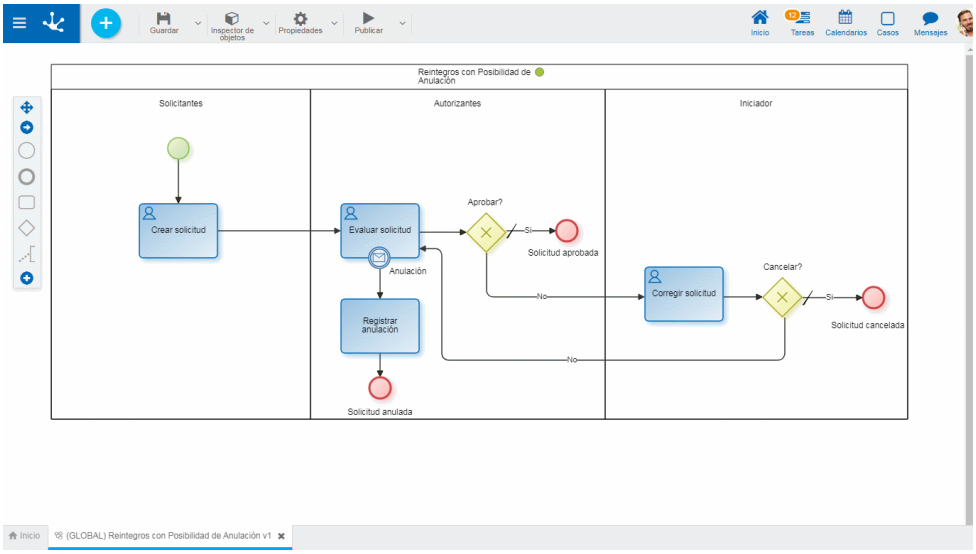
General Tab

In the "General" tab, common properties for all border events and the specific ones for each case are defined, which are enabled when assigning the type of border event.

- Email
- Throw and Catch a Signal
- Timer
- File
- Rule

3.4.10.9.3.1. Email

The properties panel of the email border event is displayed on the right side of the process modeler.



Properties

Name
Event name. It is displayed in the process next to the graphic element that represents the event.

Description
Field that allows documenting the event.

Priority
Execution priority assigned to the event. It is reported as an integer, with 0 being the highest priority. If the event performs the action mark event as processed, events with lower priority are not executed.

Interrupting
The email border event is always interrupting. This means that if the established conditions are met, the event actions are executed and the case follows the event output flow.

Enabled
Indicates that the event is enabled to receive emails and execute if applicable.

Conditions
Allows to define conditions to be met for the email event to be executed.

Actions
They are the actions that are carried out automatically when the event occurs.

- Move Case

The case affected by the event moves on to the next activity.

- Attach file

Allows to attach a file to the case.

Parameter	Description

Parameter	Description
Email attachment	Position number of the email attachment that is attached to the case. (*)
Depends on the previous execution	Indicates if the action is executed only when the previously defined actions were executed successfully.(*)

> *Note: The parameters selected with (*) are mandatory.*

- Attach form

Allows to relate a form to the case.

Parameter	Description
User Code	If it is not reported, the current user is assigned.
Values map	Report <field1><value1><field2><value2>…, where fieldN corresponds to form field names and valueN to the values that are assigned to each field. It has higher priority than DS_XML_CONTENT_PATH
XML	File path that contains the XML. It is mandatory if DS_VALUE_MAP is not reported
Annex Type	Annex type attached (*).
Create version if it exists and if it is unique	If when attaching the form, it already exists and is unique, a historical version of it is previously created.
Reset previous form and it is unique	Indicates that if there is a form associated with the case and it is unique, data is deleted from it.
Depends on the previous execution	The action is executed only if the previously defined actions were executed successfully.(*).

> *Note: The parameters marked with (*) are mandatory.*

- Check event as processed

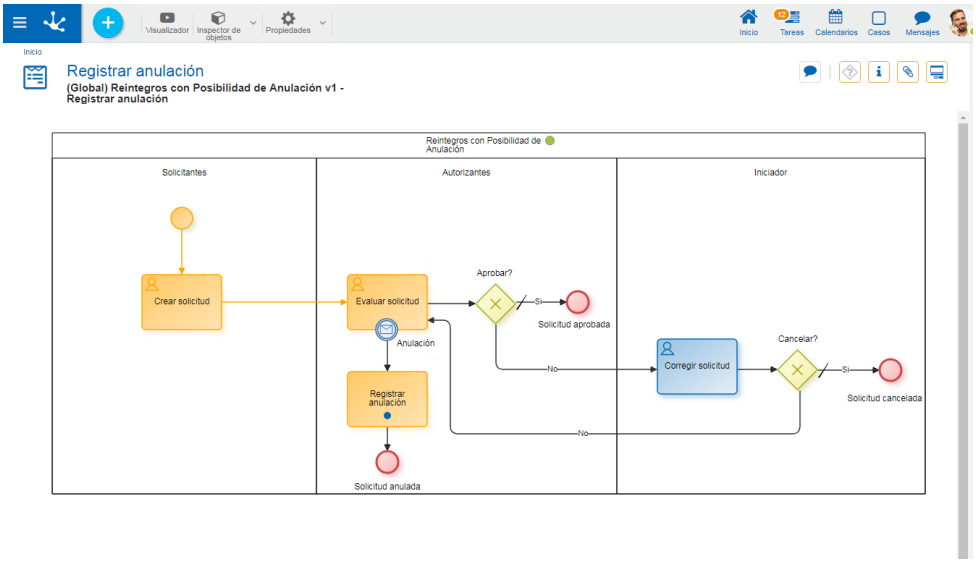
An automatic response is sent to the email received indicating that it has been processed.

Parameter	Description
Depends on the previous execution	Indicates if the action is executed only when the previously defined actions were executed successfully (*).

> *Note: The parameters marked with (*) are mandatory.*

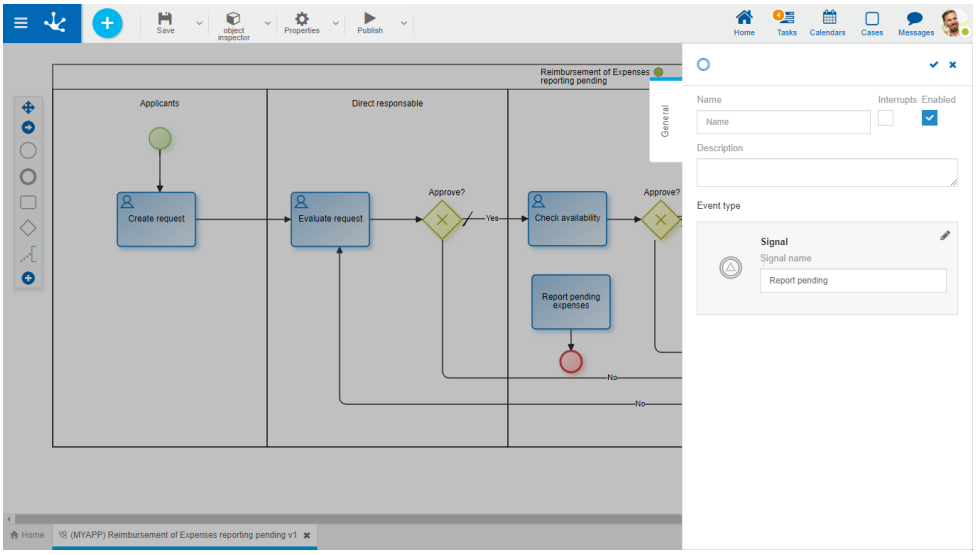
Examples of Use

In the "Refunds with Possibility of Cancellation" process, if the applicant notices a mistake in the request sent, they can send an email to cancel it, as long as it has not been approved yet.



3.4.10.9.3.2. Signal

The properties panel of the signal border event is displayed on the right side of the process modeler.



Properties

Name
Event name. It is displayed in the process next to the graphic element that represents the event.

Description
Field that allows documenting the event.

Interrupting
Defines if the event is interrupting or non-interrupting. In both cases the signal must be received when the current activity of the case is the one that has the border event.

- Interrupting: When the signal is received, the execution of the current activity is interrupted and the case follows the event output flow.
- Non-interrupting: When the signal is received, the execution of other tasks begins, defined as event output flow. The current activity is not interrupted and the case continues after it finishes.

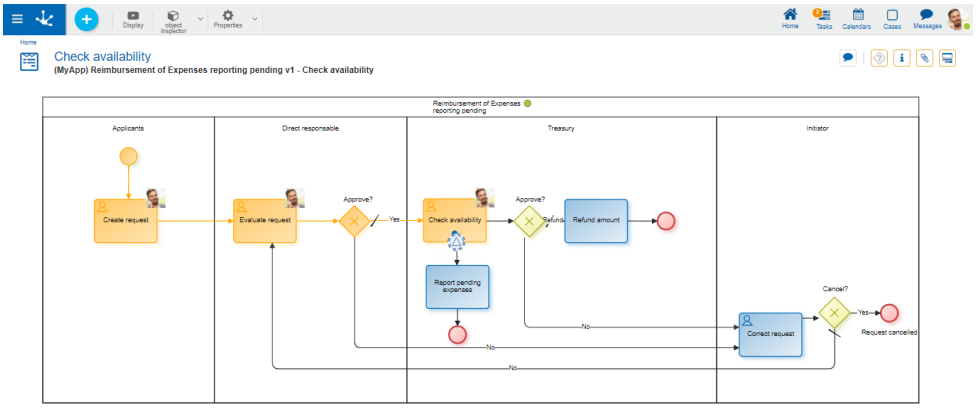
Enabled
Indicates if the event is activated to wait for the signal and execute.

Signal Code
It is the signal identifier. A character string is specified.

Examples of Use

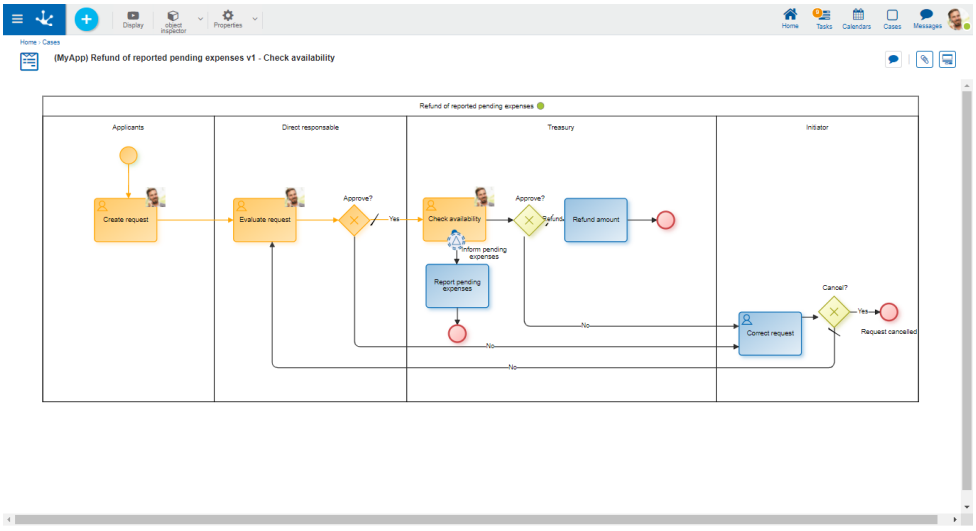
Interrupting Signal Border Event

In the "Refund with possible payment suspension" process, if the "Suspension of refunds" signal is received, the cases in the "Check documentation" activity are suspended and the process ends.



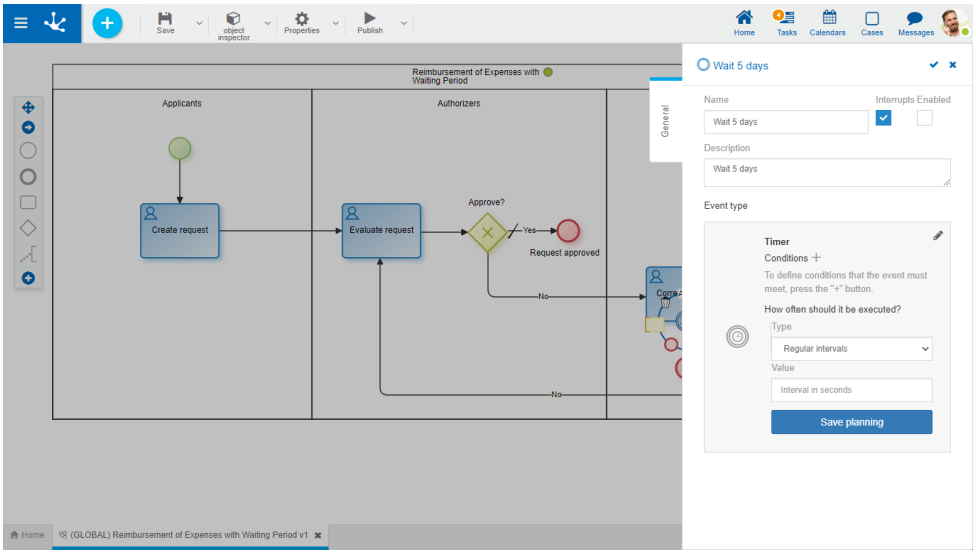
Interrupting Signal Border Event

When the "Inform pending expenses" signal is received, in the "Refund of reported pending expenses" process, a report of the cases in the "Check availability" activity is generated. The cases continue through the circuit defined in the process.



3.4.10.9.3.3. Timer

The properties panel of the timer border event is displayed on the right side of the process modeler.



Properties

Name

Event name. It is displayed in the process next to the graphic element that represents the event.

Description

Field that allows documenting the event.

Interrupting

Defines if the event is interrupting or non-interrupting.

- Interrupting: When the conditions of the event are met, the execution of the current activity is interrupted and the case follows the event output flow.
- Non-interrupting: When the event conditions are met, the execution of other tasks begins, defined as event output flow. The current activity is not interrupted and the case advances after it finishes.

The event conditions are evaluated when the current activity of the case is the one that has the border event.

Enabled

Indicates if the case is enabled to be executed as scheduled, when the established conditions are met.

Conditions

They are the conditions that must be met for the event to be executed. If there are defined conditions, a list of them is displayed.

To add a condition, press the icon + and from the panel that opens, select Property and Operator in the corresponding drop-down lists. Report Value and where applicable, indicate in Parameter the field to be evaluated of the form associated with the process.
For this type of event, apply all the conditions.

How often should it execute?

Type

Allows to define the execution schedule for the event

- Regular intervals
- Daily at a specific time

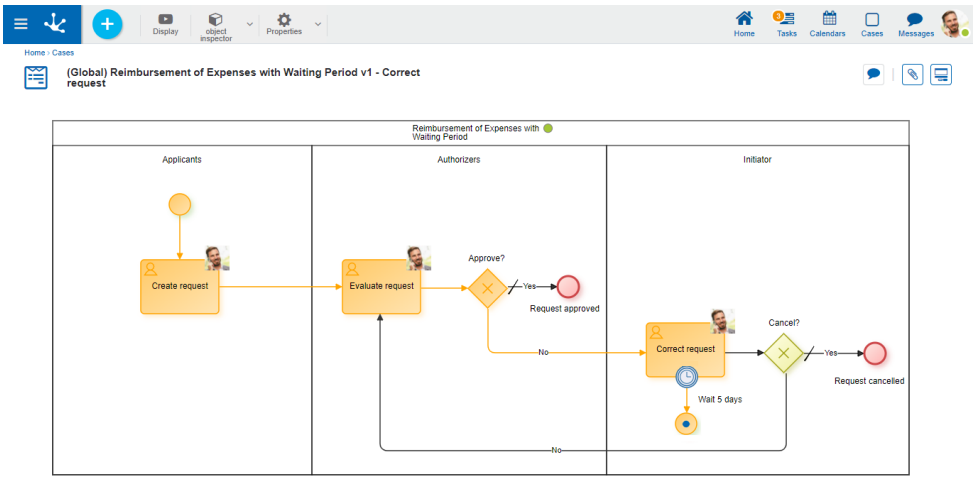
Value

Allows to define the interval value in seconds or at a specific time, according to the previously defined type.

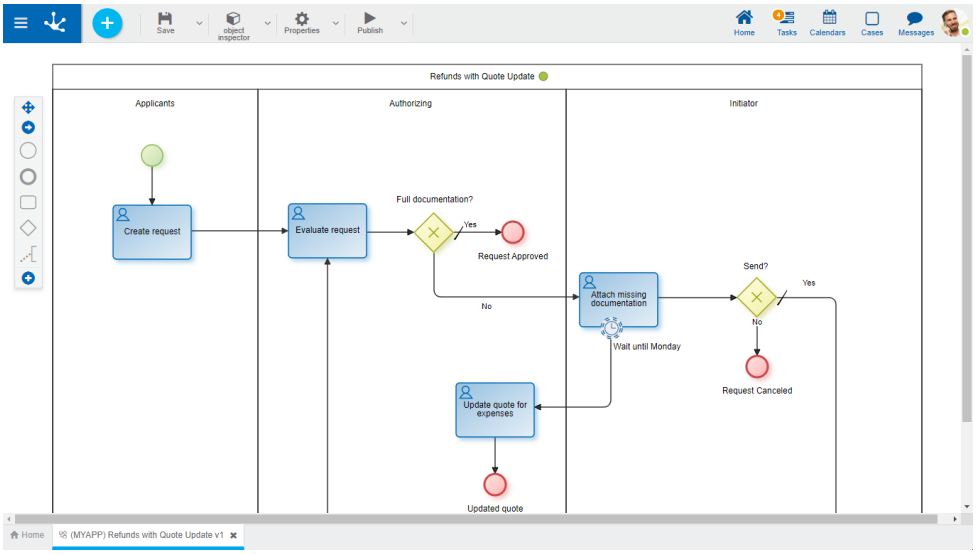
The "Save schedule" button must be pressed to keep the established schedule. It can be seen on the Scheduled Tasks Monitor.

Examples of Use

In the "Expenses Refund with Waiting Period" process, if the applicant receives the request for its modification, they have 5 days to correct it. If the deadline expires and the request was not resubmitted, the process ends automatically and the request expires.

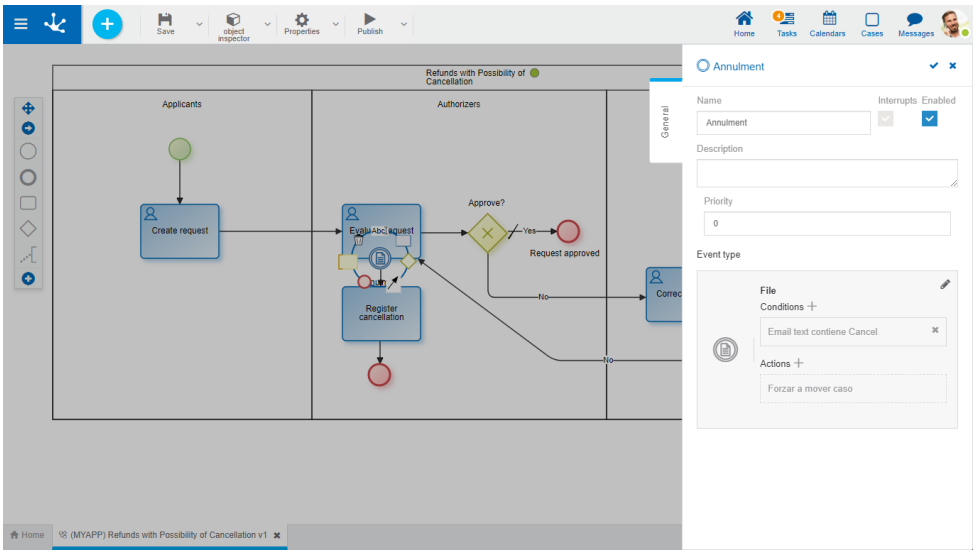


If in the "Refunds with Quote Update" process, while the case is in the "Attach missing documentation" activity, the conditions established in the timer border event are met, the quote is updated. When the applicant attaches the missing documentation, the case returns to the "Evaluate request" activity.



3.4.10.9.3.4. File

The properties panel of the file border event is displayed on the right side of the process modeler.



Properties

Name

Event name. It is displayed in the process next to the graphic element that represents the event.

Description

Field that allows documenting the event.

Priority

Execution priority assigned to the event. It is reported as an integer, with 0 being the highest priority. If the event performs the action mark event as processed, events with lower priority are not executed.

Interrupting

The file border event is always interrupting. This means that if the established conditions are met, the event actions are executed and the case follows the event output flow.

Conditions

Allows to define conditions to be met for the event to be executed.

Actions

They are the actions that are carried out automatically when the event occurs.

- Move Case

The case affected by the event moves on to the next activity.

- Attach file

Allows to attach one or more files to the case.

Parameter	Description
Depends on the previous execution	Indicates if the action is executed only when the previously defined actions were executed successfully (*).

> *Note: The parameters marked with (*) are mandatory.*

- Attach form

Allows to relate a form to the case.

Parameter	Description
User Code	If it is not reported, the current user is assigned.
Values map	Report <field1><value1><field2><value2>…, where fieldN corresponds to form field names and valueN to the values that are assigned to each field. It has higher priority than DS_XML_CONTENT_PATH
XML	File path that contains the XML. It is mandatory if DS_VALUE_MAP is not reported
Annex Type	Annex type attached (*).

Parameter	Description
Create version if it exists and if it is unique	If when attaching the form, it already exists and is unique, a historical version of it is previously created.
Reset previous form and it is unique	Indicates that if there is a form associated with the case and it is unique, data is deleted from it.
Depends on the previous execution	The action is executed only if the previously defined actions were executed successfully.(*).

> *Note: The parameters marked with (*) are mandatory.*

- Check event as processed

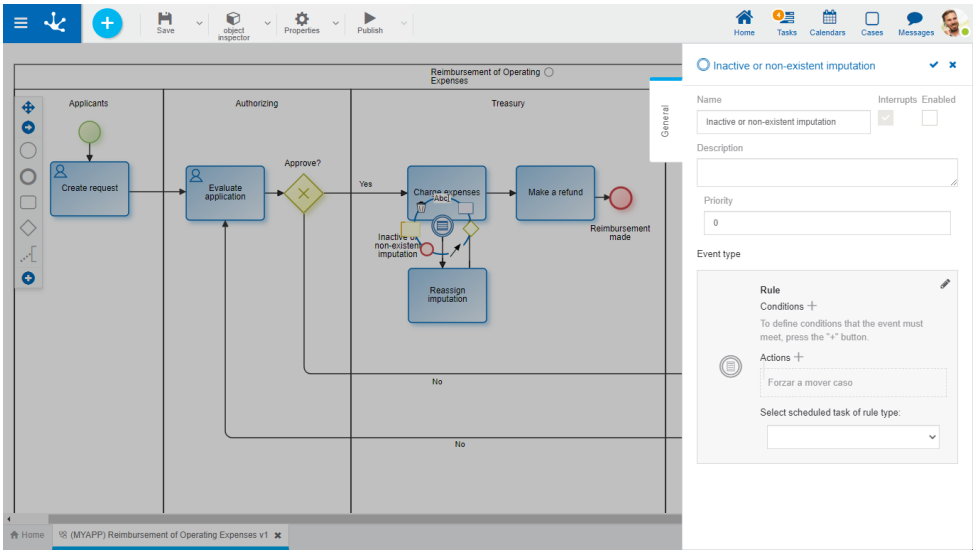
An automatic response is sent to the email received indicating that it has been processed.

Parameter	Description
Depends on the previous execution	Indicates if the action is executed only when the previously defined actions were executed successfully (*).

> *Note: The parameters marked with (*) are mandatory.*

3.4.10.9.3.5. Rule

The properties panel of the rule border event is displayed on the right side of the process modeler.



Properties

Name
Event name. It is displayed in the process next to the graphic element that represents the event.

Description
Field that allows documenting the event.

Priority
Determines the priority of the event.

Interrupting
The rule border event is always interrupting. This means that if the established conditions are met, the event actions are executed and the case follows the event output flow.

Enabled
Indicates that the event is enabled to be executed according to the rule.

Conditions
They are the conditions that must be met for the event to be executed. If there are defined conditions, a list of them is displayed.

To add a condition, press the icon + and from the panel that opens, select Property and Operator in the corresponding drop-down lists. Report Value and where applicable, indicate in Parameter the field to be evaluated of the form associated with the process.

Actions
They are the actions that are carried out automatically when the event occurs.

- Move Case

The case affected by the event moves on to the next activity.

- Attach file

Allows to attach one or more files to the case.

Parameter	Description
File to attach	File attached to the case. (*)
Depends on the previous execution	Indicates if the action is executed only when the previously defined actions were executed successfully.(*)

> *Note: The parameters selected with (*) are mandatory.*

- Attach form

Allows to relate a form to the case.

Parameter	Description
User Code	If it is not reported, the current user is assigned.
Values map	Report <field1><value1><field2><value2>…, where fieldN corresponds to form field names and valueN to the values that are assigned to each field. It has higher priority than DS_XML_CONTENT_PATH
XML	File path that contains the XML. It is mandatory if DS_VALUE_MAP is not reported
Annex Type	Annex type attached (*).
Create version if it exists and if it is unique	If when attaching the form, it already exists and is unique, a historical version of it is previously created.
Reset previous form and it is unique	Indicates that if there is a form associated with the case and it is unique, data is deleted from it.
Depends on the previous execution	The action is executed only if the previously defined actions were executed successfully.(*).

> *Note: The parameters marked with (*) are mandatory.*

Select scheduled rule task

Allows to select the rule to use, which must be an advanced rule and must be published.

Example of Use

In the "Refund of Operating Expenses" process, if the "Charge expenses" activity is executed and a rule that detects inactive or non-existent expense charges is executed, charges are reassigned so that the case can move to the "Make a refund" activity.

3.4.10.9.4. End Events

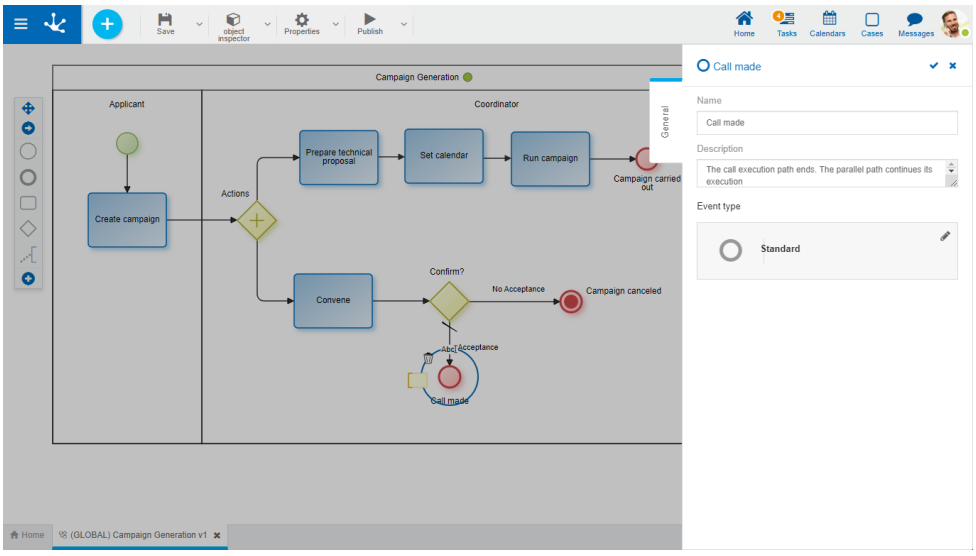
General Tab

In the "General" tab, common properties for all end events and the specific ones for each case are defined, which are enabled when assigning the type of end event.

- Standard
- Signal
- Terminal

3.4.10.9.4.1. Standard

The properties panel of the standard end event is displayed on the right side of the process modeler.



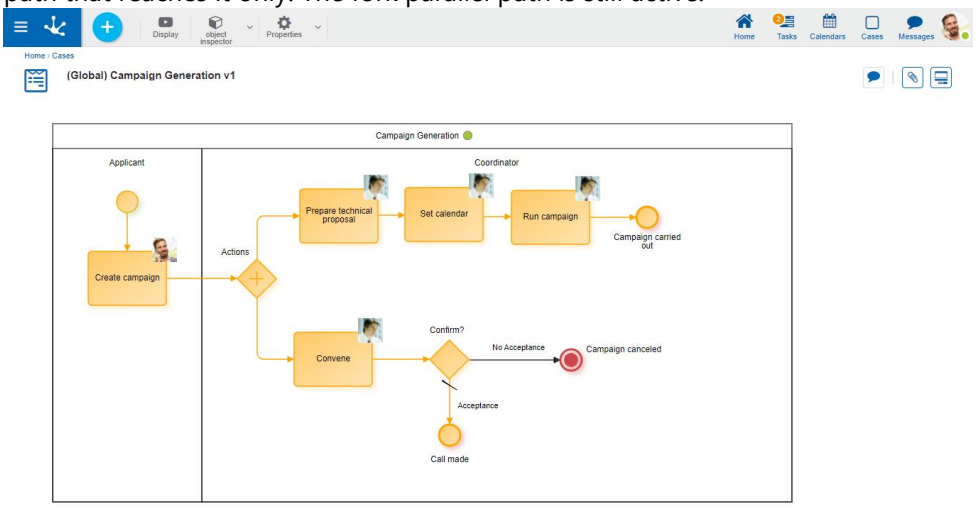
Properties

Name
Text that is displayed in the diagram next to the graphic element of the event.

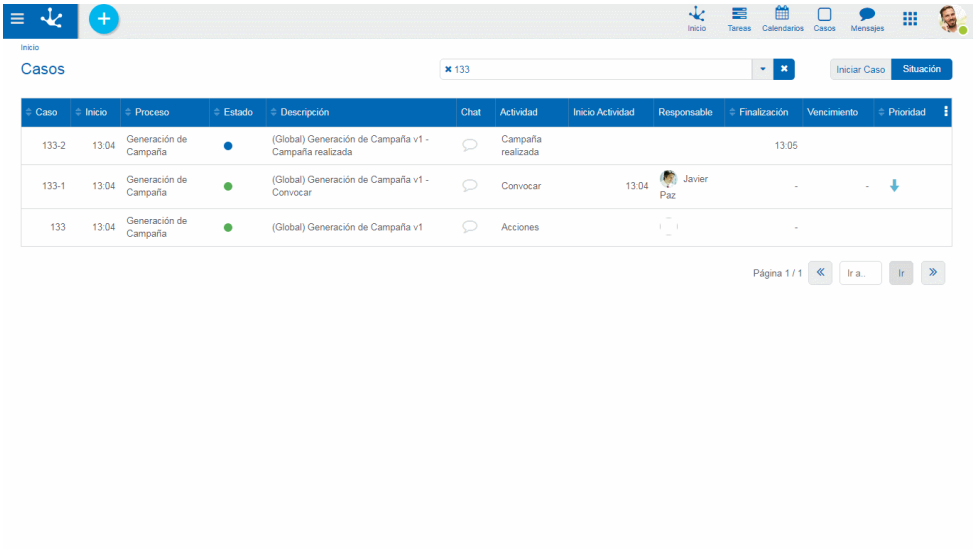
Description
Field that allows documenting the event.

Example of Use

The "Generate Campaign" process has two paths. The standard end event finishes the execution of the path that reaches it only. The fork parallel path is still active.

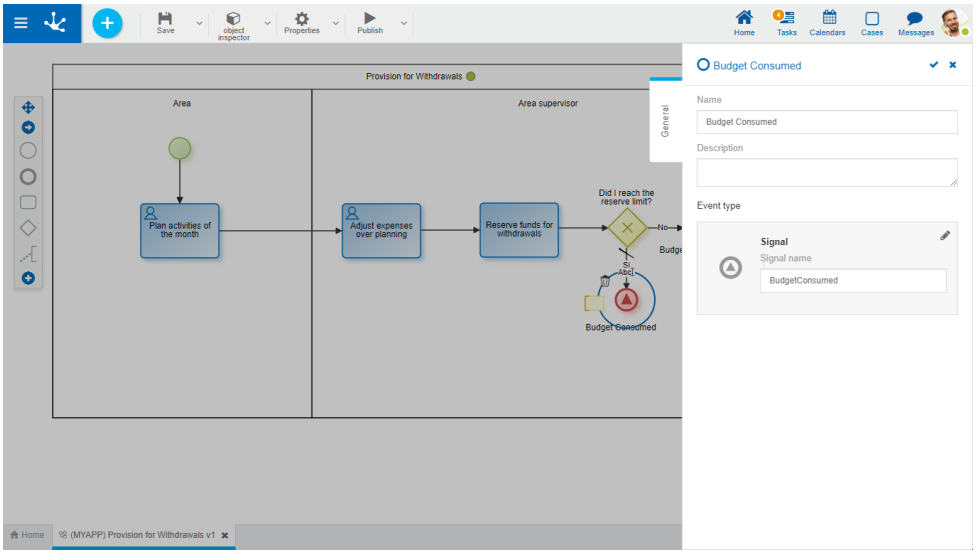


In the case show, the path end in which the standard end event is located can be seen. The parallel path is still active.



3.4.10.9.4.2. Signal

The properties panel of the signal end event is displayed on the right side of the process modeler.



Properties

Name

Text that is displayed in the diagram next to the graphic element of the event.
It is recommended to refer to the waiting signal.

Description

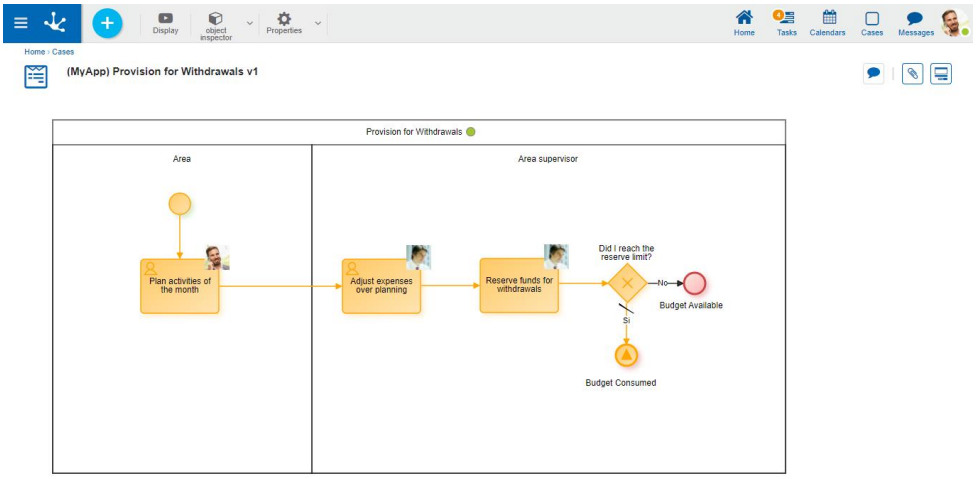
Text that allows documenting detailed information about the event.
The processes that send the signal can be informed.

Signal Name

Enter a string of characters representing the signal code to send.
The name entered must match the one configured when the signal is received.

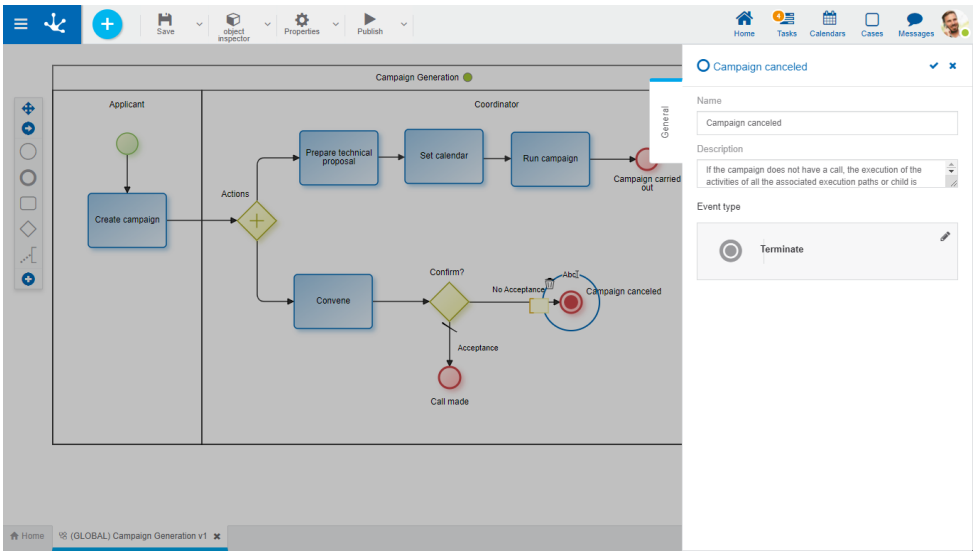
Example of Use

If the "Provision for Refunds" process reaches the availability limit, it sends the "Consumed Budget" signal. The "Actions Auditing" and "Budget Update" processes start when the signal start events receive the aforementioned signal.



3.4.10.9.4.3. Terminal

The properties panel of the terminate end event is displayed on the right side of the process modeler.



Properties

Name

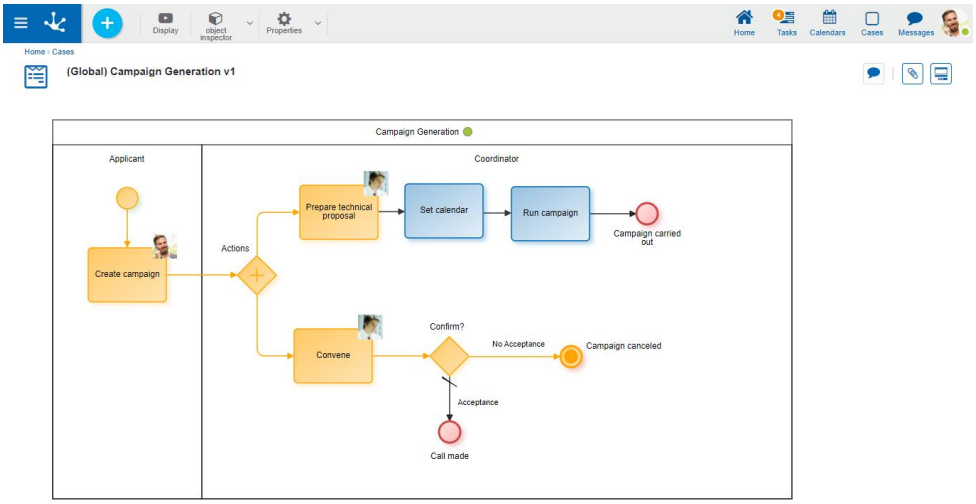
Text that is displayed in the diagram next to the graphic element of the event.

Description

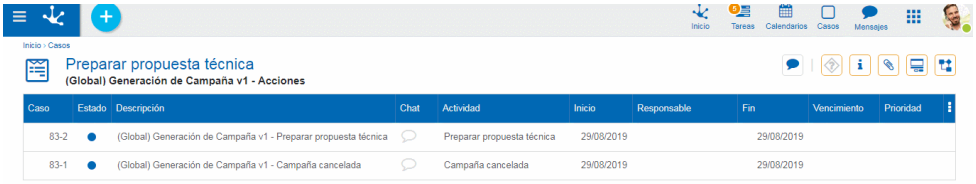
Text that allows documenting detailed information about the event.

Example of Use

The "Campaign Generation" process, all paths are canceled when the evaluation of the call is not satisfactory.



In the case show, the termination of all fork paths can be verified.



3.4.10.10. Good Practices

Good practices for processes modeling in **Deyel** consist of a set of recommendations, to be met whenever possible.

They allow to build clearer and more understandable processes models, standardizing the use of the graphic elements that compose them.

- Processes Modeling
- Lanes Modeling
- Activities Modeling
- Events Modeling

- Flows Modeling
- Gateways Modeling

3.4.10.10.1. Processes Modeling

Good Practices

Name

- The name of the processes and sub processes must clearly describe their main purpose.
- Short names or abbreviations must not be used.

Diagrams Size

- Long diagrams do not allow to give a global perspective to the readers, they make reading difficult and don't communicate clearly the purpose of the processes.

Number of Activities

- Although various factors affect the complexity of a process, the number of activities is one of the most relevant factors.
- As a general criterion, diagrams with a maximum of 15 to 20 activities are recommended.

Unification of Activities

- If a set of consecutive tasks can be executed sequentially by the same person, at the same time, then these activities must be integrated into one.

Subprocesses

- The use of subprocesses is recommended to group activities with the same purpose.

Activities Alignment

- It is recommended to keep the activities aligned with each other, horizontally and vertically. This practice allows ordering the diagrams, keeping them neat visually.

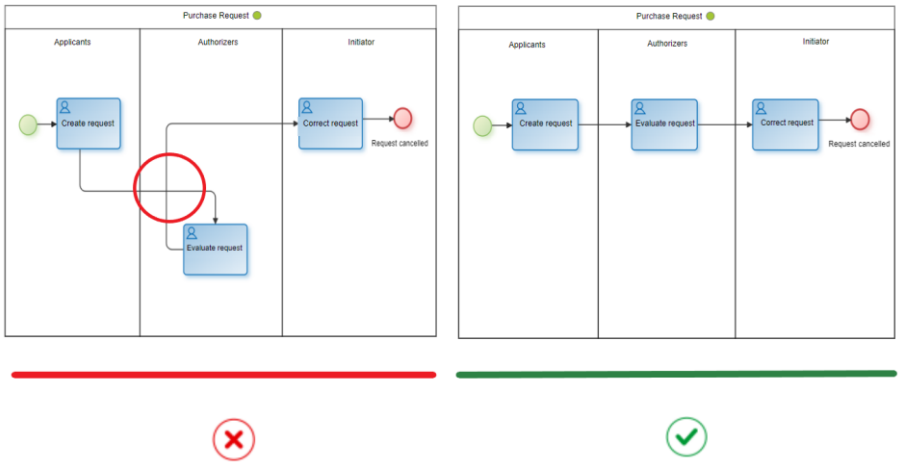
Standard Size of Elements.

- It is recommended not to change the size of diagram elements.
- Keeping the standard sizes avoids discrepancies and keeps the visual elements homogeneous.

Flow Orientation

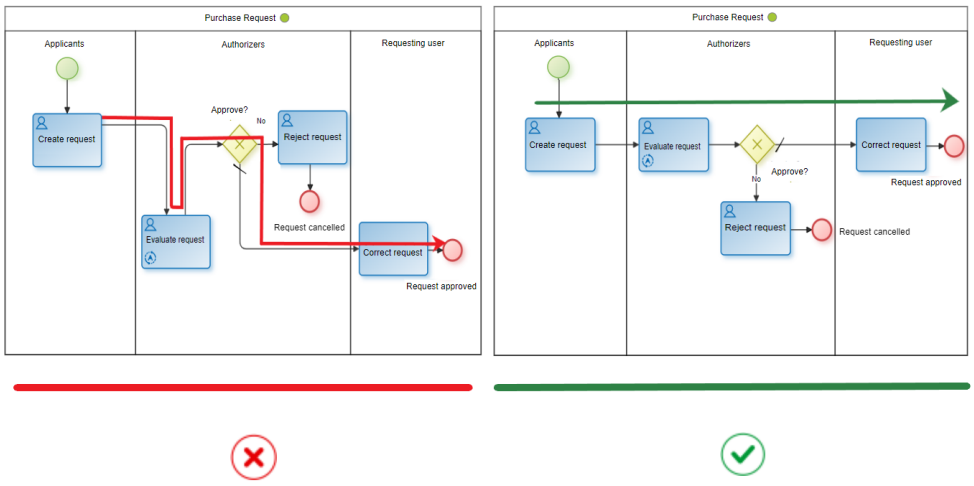
- For people who read from left to right, it is intuitive to follow a logical flow from left to right or from top to bottom. It is recommended to diagram the process flows in this direction.

- The crossing of connectors must be avoided, maintaining a chronological sequence and a consistent flow direction.



Primary Scenario Clarity

- The main road should be easily identified when reading the diagram, so it is recommended to diagram the main road first and then add the alternative roads.



Links

- It is recommended to use this type of event to avoid complexity in the flows.


Throw Link


Catch Link

Artifacts

- It is recommended not to overuse the objects that serve to document the process. Notations, associations, groups and data objects are elements that can clarify the diagram, but the excessive use of these types of elements can increase the visual complexity of the model.
- In order not to use a large number of artifacts, it is recommended to document the details within each activity or each element of the model.

3.4.10.10.2. Lanes Modeling

Good Practices

Name

- The lane name must be chosen carefully, so that it accurately represents the participant of the process.
- It is recommended to use the description field to document the responsibility.

Participant Type

- Avoid using responsibles of type user as much as possible.
- It is recommended to use roles or agents, that give greater possibilities to changes.

Need

- A lane must be created only if at least one intermediate task or event is executed on it.
- It is recommended not to create lanes to represent participants who only run automatic activities.

3.4.10.10.3. Activities Modeling

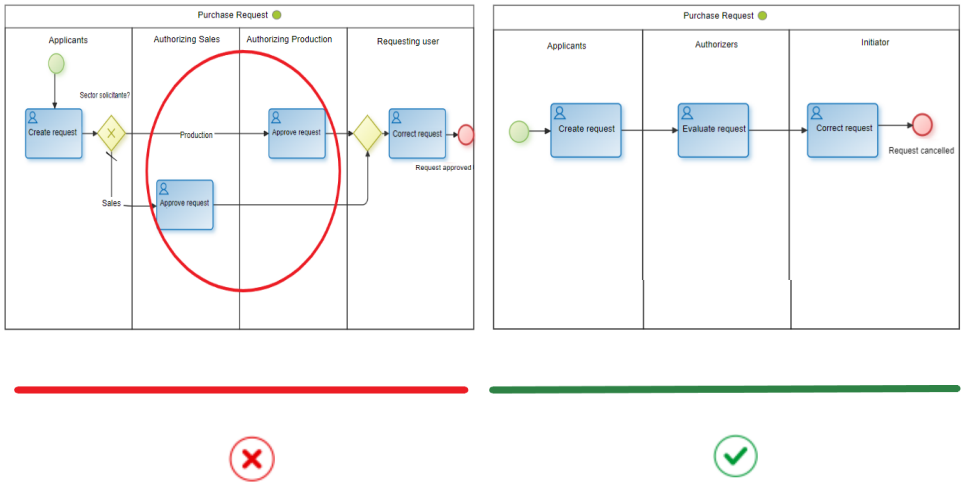
Good Practices

Name

- The activities name is made up of an infinitive verb and an object.
- The name must clearly identify the objective of the activity. For example, "Create request", "Authorize request", "Notify rejection", and so on.
- The first letter must be used in uppercase and the others in lowercase. Short names or abbreviations must not be used.
- It is recommended to avoid long names, they usually denote that certain details of the activity affect their name. The name must represent the objective while details are documented within the activity.
- The use of "and" and "or" in the names of activities must be avoided. These types of compound names can mean that they are actually two different activities.
  For example, if the activity is called “Review and approve request”, you must consider which is the correct option. If the same participant performs both actions at the same time, then the most relevant action must be chosen and used to name the activity, then it can be assumed that the primary objective is approval and the review is a subtask. The correct name is “Approve request”. On the other hand, if it is considered that both actions have the same relevance and are made at different times, then it is convenient to separate them as “Review request” and “Approve request”.

Repetition

- Multiple instances of the same task should not be diagrammed to represent multiple participants.
- It is recommended to use roles or agents so that the participant assignment is made accordingly, when executing the process.



Typing

- It must be indicated each activity type.
- It is recommended to be careful selecting the appropriate type, which allows a correct interpretation of the diagram.

Automatic Tasks vs Automatic Actions

- There are certain actions, such as sending an email, executing a rule, etc. that can be modeled in different ways in **Deyel**:
  a) Representing them explicitly in the model, through shipping type activities, business rule, services, etc.
  b) Specifying automatic actions that are executed when starting or ending a user activity or manual activity.

- The recommendation is that the tasks to be viewed individually have to be modeled as different activities and the type of each one is selected.

- Those others that do not need to be visually explicit, should be modeled as automatic actions of user activity. For example, a rule that retrieves the initial value of a field before executing the user activity,

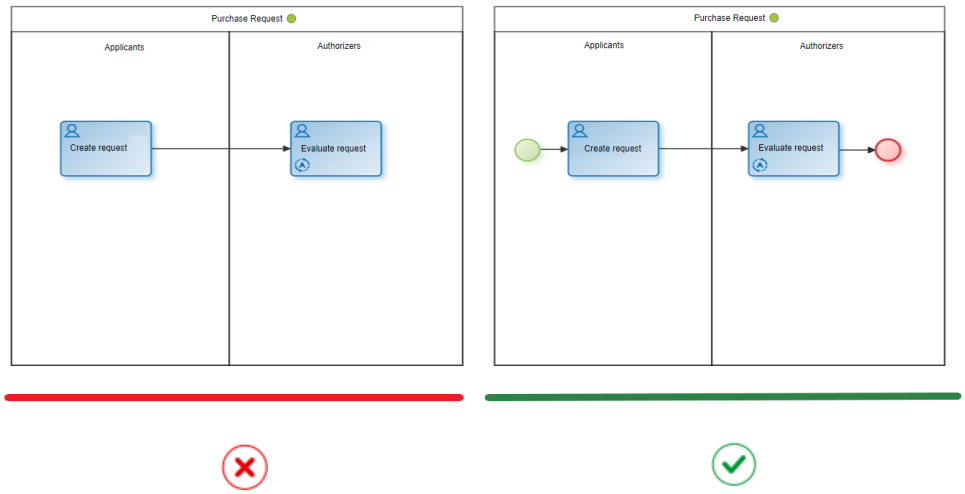
- It is important to maintain a uniform criterion in this regard and also consider that long diagrams are not convenient.

3.4.10.10.4. Events Modeling

Good Practices

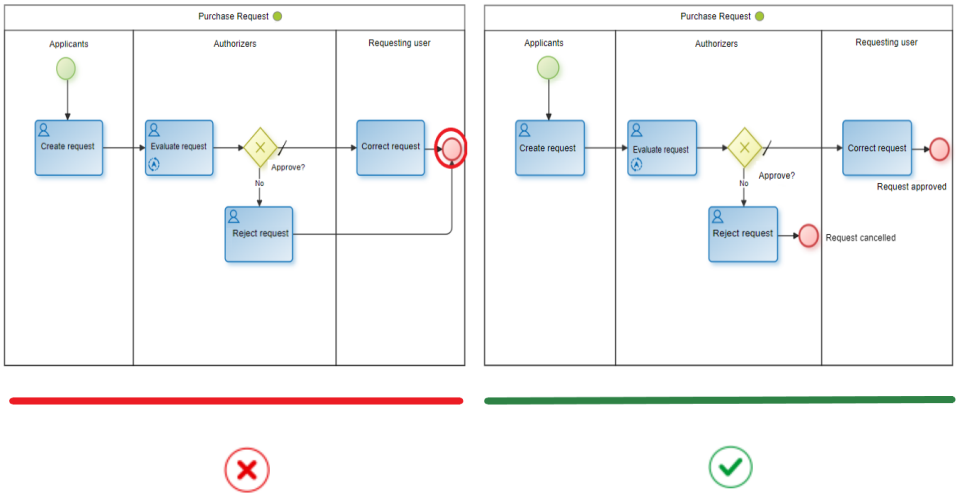
Start and End Events

- It is recommended to always use the start and end events in each process and subprocess to represent its beginning and end.

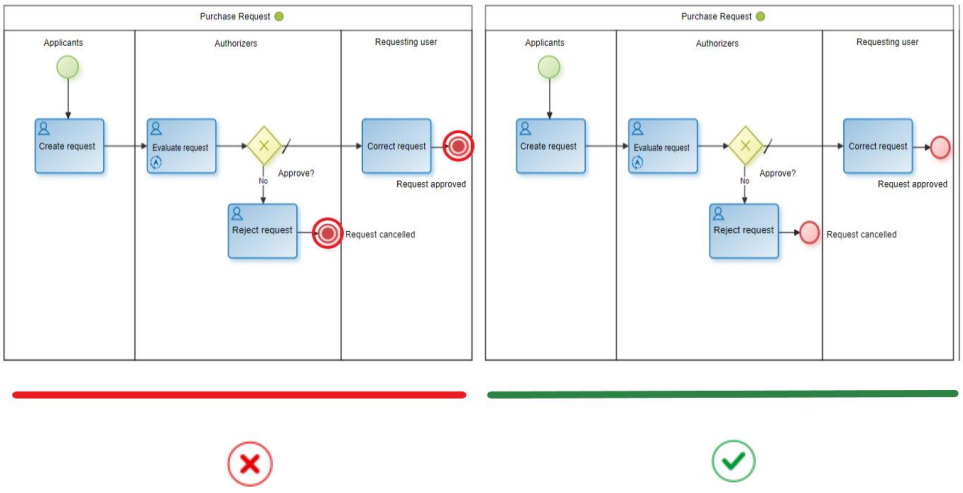


Name

- The start and end events must not be tagged when they are unique. It is very common to name them as "Start of the process" and "End of the process" but this is redundant and unnecessary.
- Events must be identified when multiple start or end events are used.
- The start events must have labels indicating the condition or fact that triggers the start of the process.
- The end events must be named indicating the final state of the process. In general the name must use a noun, which is the business object, along with a verb in the past tense. For example: "Request approved", "Request canceled", etc.
- It is recommended to identify the final successful and unsuccessful states.



- Terminal end events must only be used when strictly necessary. They are used to model situations where multiple alternative paths are enabled and only the completion of one of them is required to complete the entire process.



3.4.10.10.5. Flows Modeling

Good Practices

Use of Legends

- It is recommended to use the legends to represent the condition through which a flow is passing.
- Legends have to be visually associated with the flow and are not confused with other legends.

Avoid Crossings

- As far as possible the crossings of flows should be avoided.
- If necessary, let it be in places where the legends of the flows are not confused.

Avoid Diagonal Flows

- It is always preferable to use "L" shaped flows.
- There can be multiple vertices in the flow and it is recommended to minimize their number.

Anchor Points or Ports

- Central anchor points have to be used in activities.
  - Flows enter from the top side and the left side.
  - Flows come out from the bottom side and the right side.
- In the gateways it is recommended to use the anchor points of the vertices.
  - Flows enter through the top or left vertices, using the same entry point.
  - It is recommended to distribute the outgoing flows neatly.
  - The gateway name must not be below the flows.

3.4.10.10.6. Gateways Modeling

Good Practices

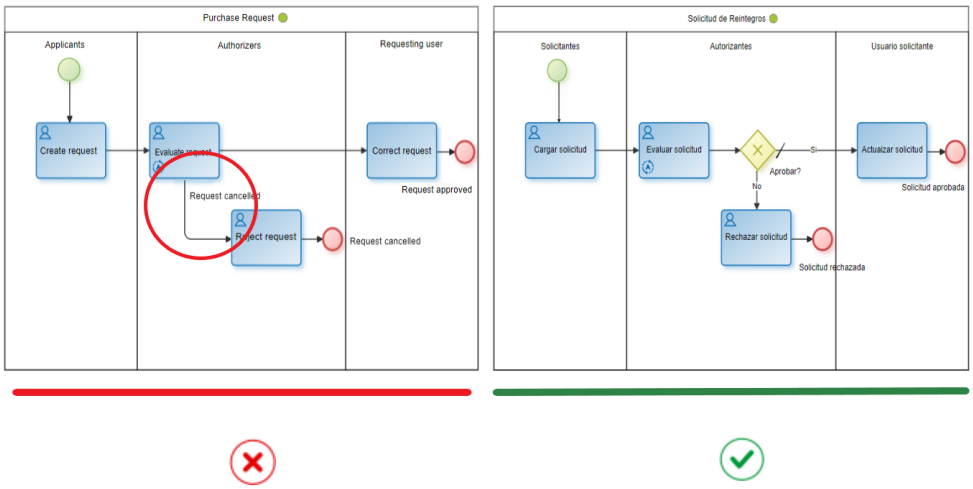
Name

- All divergence gateways must be named.
- A question is generally used that represents the decision or logical condition evaluated. For example: "Authorized Request?".

- All outgoing flows must be named.
- Their name must represent the responses or possible values of the condition evaluated.
  For example, if a gateway is called "Request Authorized?", Its outgoing flows should be called "Yes", "No".

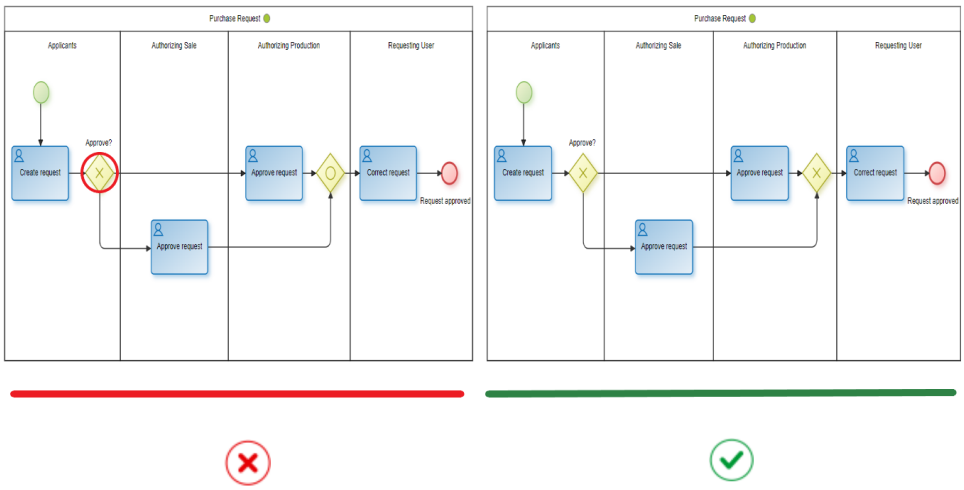
Gateways for Forks

- Flows must not be branched using tasks.
- Gateways should always be used for this purpose.



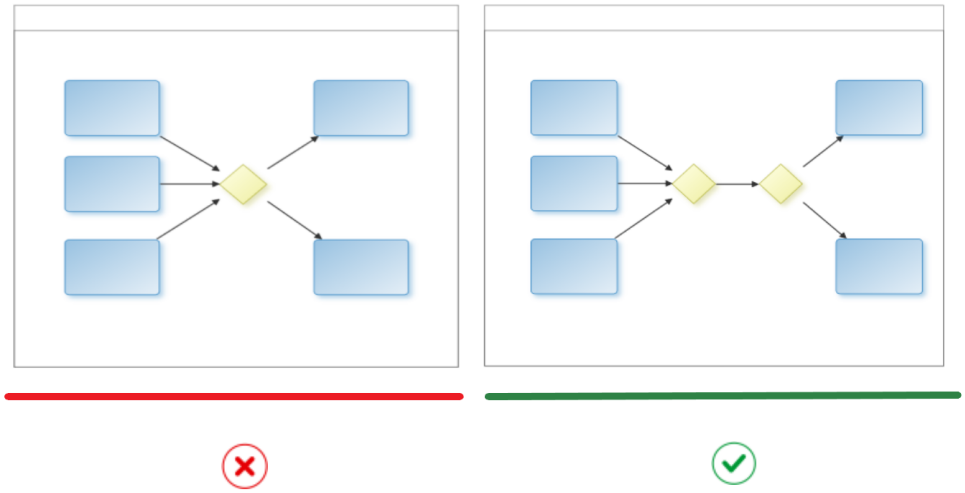
Gateways for Synchronization

- To synchronize flows, it is recommended to always use the same type of gateway that is used to fork them.



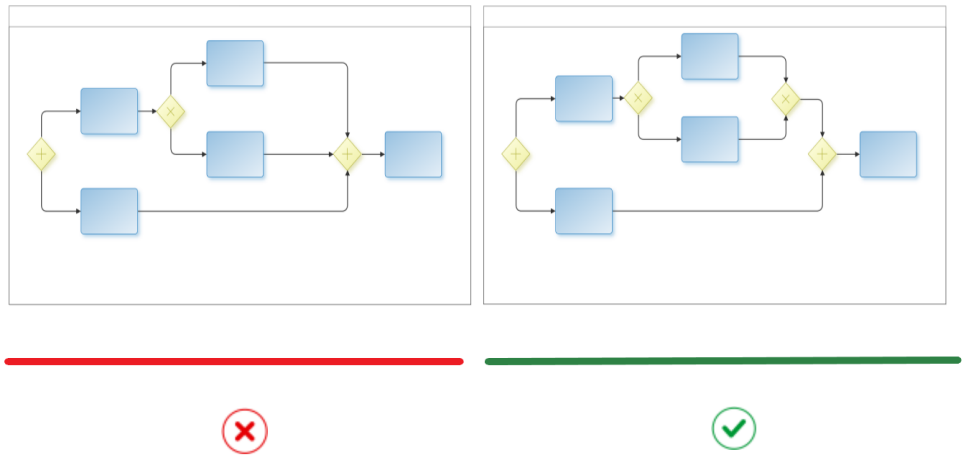
Separate the forks and Synchronizations

- It is recommended not to use gateways to join and separate flows at the same time.



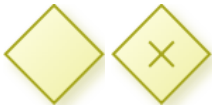
Balancing

- Flow branches must be equivalently synchronized.
- The number of flows leaving a fork must be equal to the number of flows that are received in a synchronization.



Exclusive Gateway

Has two representations. The same representation must be used for the entire definition of a process.



- It can be used as a fork or synchronization element.
- It is used to represent a fork in the flow, where there are several possible alternatives but only one should be considered.
  They are mutually exclusive alternatives.
- These gateways require the definition of a default flow.
  This flow represents the path taken when none of the other conditions are met.
  We recommend that the most common path, the most probable, is the one that is modeled as the default flow.
- The exclusive gateway can be defined in pairs, one as a divergent element to activate several mutually exclusive paths and if these paths converge, a gateway of this type must be used as a convergent element, to graphically represent the concurrence of the flows.

Parallel Gateway

Its graphic representation is as follows:



- It can be used as a fork or synchronization element.
- When used as a fork, the output flows do not require conditions as all paths must be followed. Likewise, it is recommended to put a legend in each flow that represents the condition that was met in order to be able to transit such flow.
- When the gateway is used as a convergent element (synchronization), all the predecessor activities are expected to finish their execution.
- The parallel gateway must be defined in pairs, one as a divergent element to activate several parallel paths, and the other as a convergent element, to synchronize the previously activated paths.

Inclusive Gateway

Its graphic representation is as follows:



- It can be used as a fork or synchronization element.

- The inclusive gateway as a fork element is used at a point in the process where one or more alternatives may be viable based on a condition.

  For example, after creating a document, it must be approved by the corresponding offices according to the type of document created.
  To implement the multi-decision pattern it is necessary to use the inclusive gateway.
  This gateway allows one or more paths to be enabled according to the evaluation of the conditions of each flow.

- If none of the conditions are met, a default flow can optionally be defined, that takes into account this situation.
  If this default flow is not defined, then an error is obtained stating that there are no viable alternatives.

- The inclusive gateway as a convergence element is activated if at least one case reaches the inclusive gateway and if some other case of the processes execution that could reach the gateway (the canceled threads are not considered) arrived previously.

- The inclusive gateways can be defined in pairs. That is, one as a divergent element that generates parallel activities, and then that converge in an inclusive gateway of convergence, which synchronizes the previous paths.

3.4.10.10.6.1. Patterns

Patterns for processes design are examples that show how to connect activities to solve a common problem.

Parallel Paths

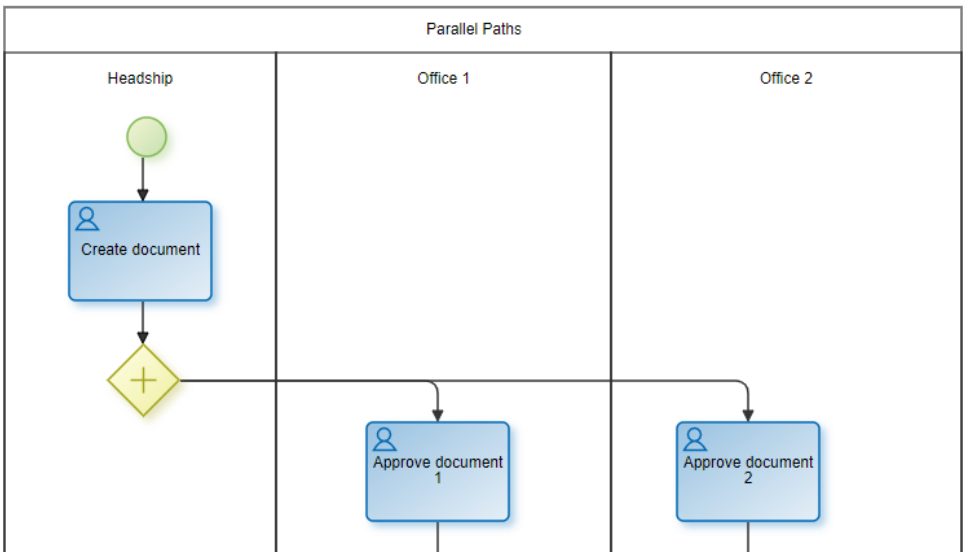
A parallel path is a point in the process execution where the execution flow is divided into two or more flows that are executed in parallel.

Example

A document is created in one office, which requires the approval of two other offices.

Implementation

To implement parallel paths, it is necessary to use a parallel gateway. This gateway creates all alternative paths without evaluating conditions.



Synchronization

Synchronization is a point in the process where two or more different flows of the process are joined into a single flow. It is called synchronization because it waits for all the flows to be joined to complete before proceeding to the next activity.

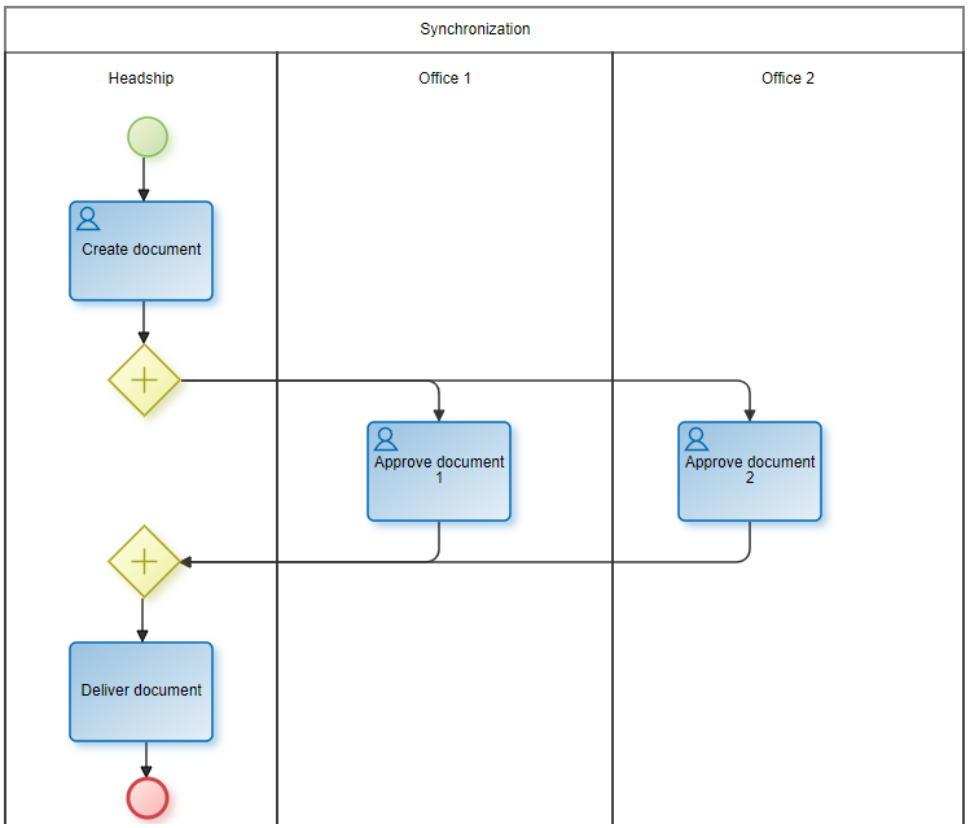
Example

The document created by one office must be delivered after it has been approved by two other offices.

Implementation

In the example we use the parallel gateway as convergence which means that it synchronizes the previously activated paths.

The synchronization pattern can also be modeled with the exclusive and inclusive gateway, according to the needs of the business.



Exclusive Decision

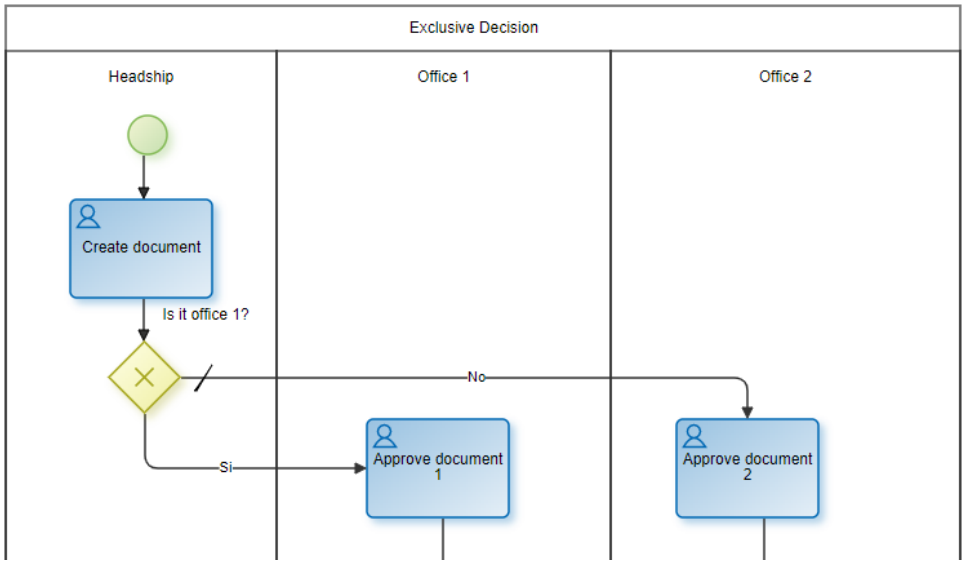
Exclusive decision is a point in the process where a path is chosen from several paths based on a condition or information from the process.

Example

When creating a document, it must be approved by an office that corresponds to the type of document created.

Implementation

The exclusive decision pattern can be modeled with the exclusive gateway. The exclusive decision has multiple output flows, but only one of them can be made based on the conditions of each stream.



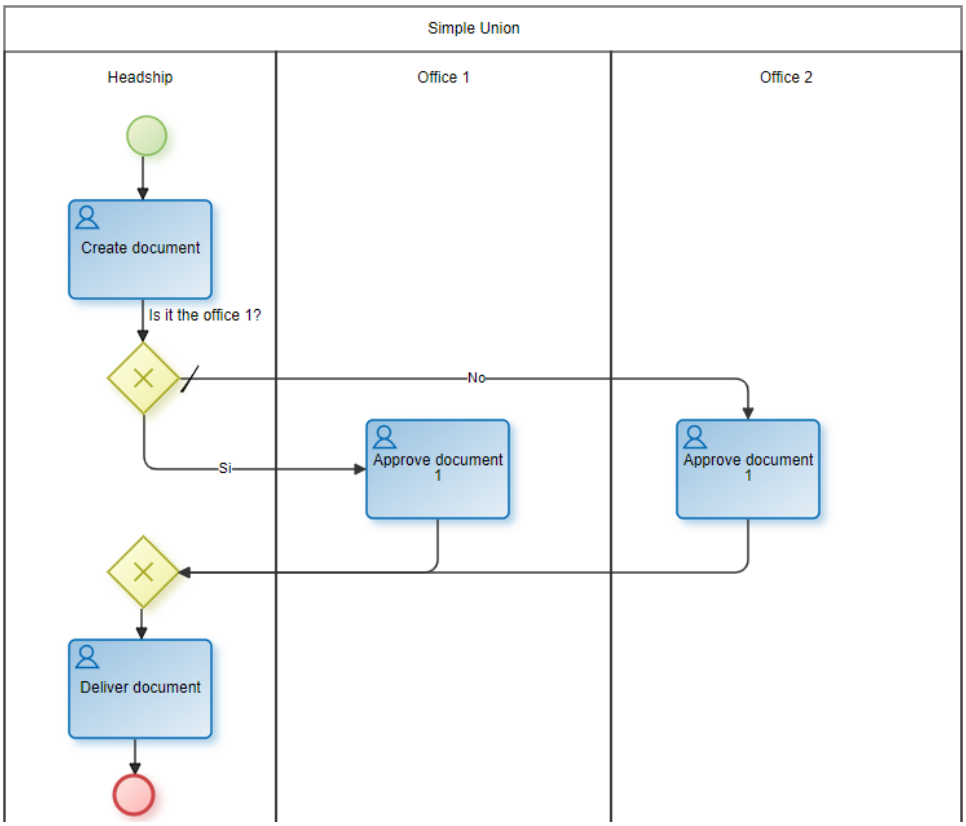
Simple Union

Simple union is a point in the process where two or more alternate paths meet without synchronization. This pattern assumes that of the different paths only one is executed.

Example

After creating a document, it must be approved by the corresponding offices according to the type of document created. Once approved by the corresponding office, it must be delivered by the office that created it.

Implementation



Multi-Decision

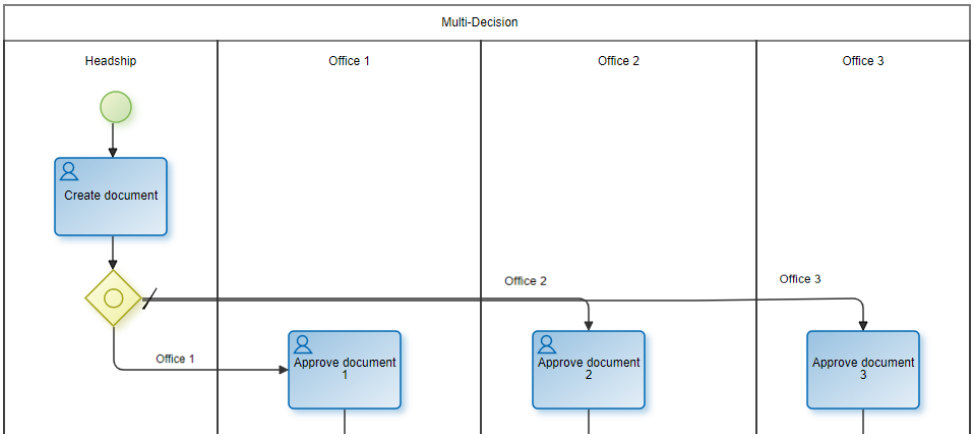
The multi-decision pattern is used to model a point in the process where a set of paths are chosen based on a condition.

Example

After creating a document, it must be approved by the corresponding office or offices according to the type of document created.

Implementation

To implement the multi-decision pattern it is necessary to use the inclusive gateway. This gateway allows one or more paths to be enabled according to the evaluation of the conditions of each flow.



Synchronized Structured Union

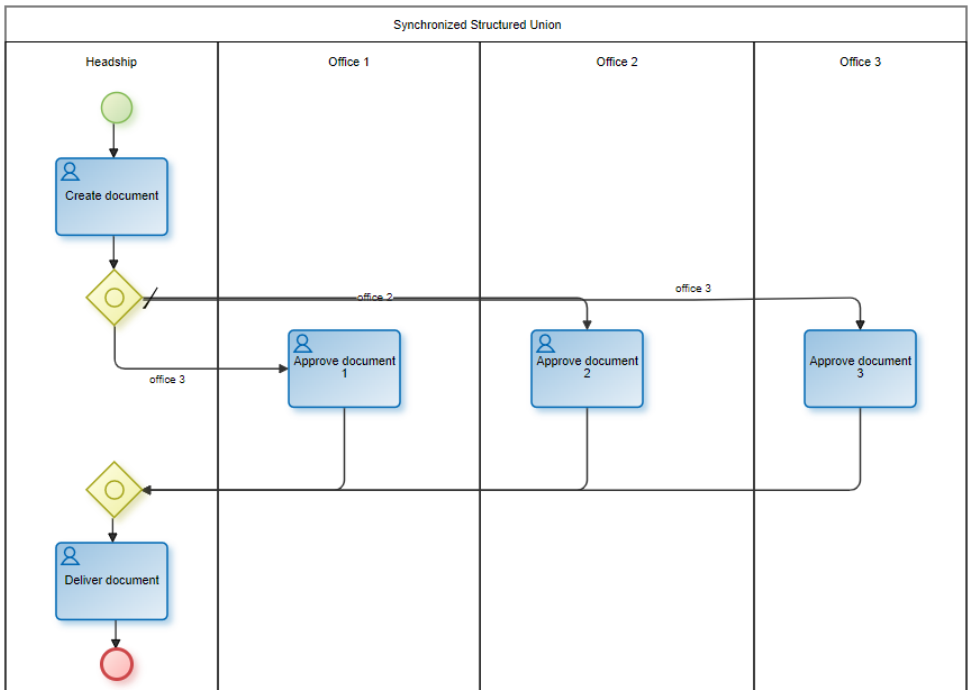
It is a point in the process where multiple paths that were previously activated in the process converge in a single thread of execution.

Example

Once the document has been approved by the appropriate office or offices, it must be delivered by the office that created the document.

Implementation

To implement this pattern it is necessary to use two inclusive gateways, one for divergence (activate some paths) and the other for synchronization (synchronize the activated paths).



Multi-Merge

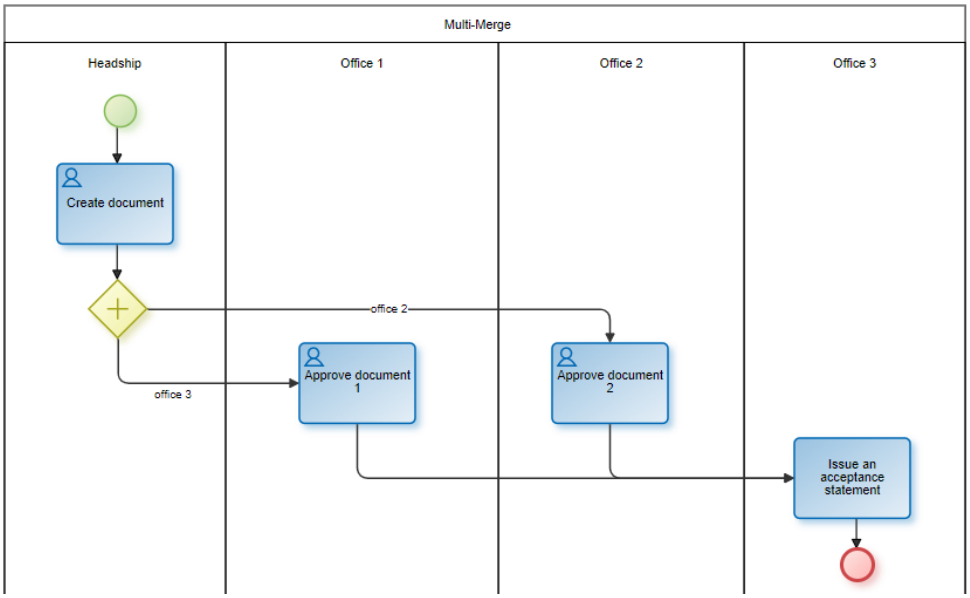
The multi-merge pattern is used to model the convergence of two or more paths in a single path. Each time an input path is activated it activates the next activity in the execution flow.

Example

After creating a document, it must be approved by two offices. Each time an office approves it, a third office must send an approval release.

Implementation

The pattern uses a parallel gateway to enable two paths. Each time the activities on one path are executed, the next activity will be as well. In this case, the "Send approval statement" activity is executed twice.



3.4.11. Entities Modeling

Entities Modeling

The entity modeler is a tool that allows defining an object representing business data and designing its user interface graphically. It also enables defining the features of its fields and behavior in associated processes.

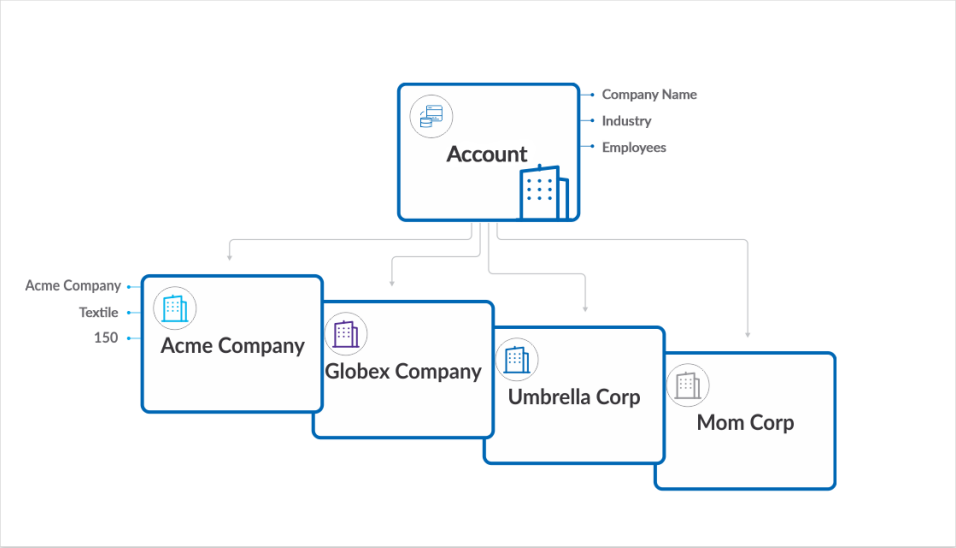
It allows modeling entities as they are perceived in the real world by dragging and dropping elements that define their appearance. These elements have properties set with default values. Furthermore, the persistence structure is implicitly modeled in the database of **Deyel**. This enables data loaded into instances of the entities to be stored in that database.

The entities modeling is complemented by the assistance of **Diana**, the conversational system with artificial intelligence of **Deyel**. It can guide the modeler user in the definition of an entity through dialogue in natural language, proposing fields, relations and configurations based on the needs of the request. This allows creating or extending entities without having to start from visual modeling, or to freely alternate between both approaches according to the user's preference.

When defining an entity, its Rest API is automatically generated, so that the entity can be integrated with other applications.

An entity or business entity is a representation of a real-world concept in an application. For example, in a CRM application, entities can be Accounts, Opportunities, Budgets, etc. An entity has attributes that describe it, for example in an Account its attributes can be the Company Name, the Industry to which it belongs, and the Number of employees. Besides, an entity stores the application's data within itself.

An instance of an entity is a specific occurrence of that entity within the application. It represents a single version of the entity with its own associated values and attributes, each account corresponds to a unique instance of the entity within the application.

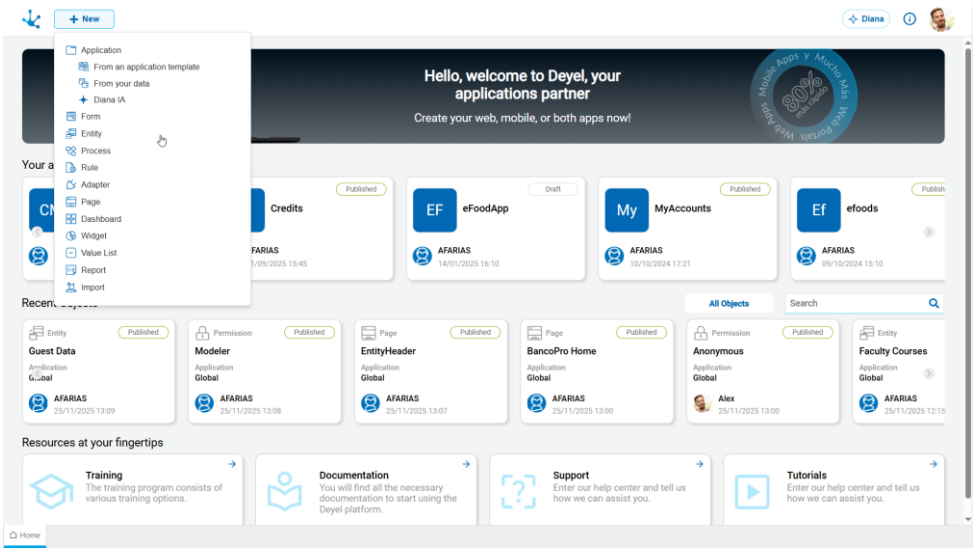


Examples:

- In a CRM application, entities can be Accounts, Opportunities and Budgets.
- The attributes of an "Account" entity can be "Company Name", "Industry" and "Number of Employees".
- An instance of the "Account" entity can be a specific account with its company name, industry type, and number of employees.

The entity modeler inherits the graphical functionality of the pages modeler. While an entity has its data in its graphical interface, the page can contain different entities within the same graphic interface. Unlike entity fields, the fields on pages do not persist.

The entity modeler allows the use of a set of different types of elements and another set of pre-assembled elements. The latter facilitate the modeling of entities and are known as advanced elements. All these topics are detailed under this module.





This button is used to create an entity from the option  Entity.

The general features of the entity modeler and the elements that compose it are described in the following topics:

- Modeling Facilities
- Entity Properties

3.4.11.1. Modeling Facilities

General characteristics of this modeler are specified below.

New Entity

The modeler user defines a new entity, which is available for use after being published.

It can be modeled simultaneously:

- Data model layer
  Involves modeling all the fields with associated data types, identifier field, fields used as filters in searches, field data lengths, behavior, and other validations related to business rules.

- Entity display layer
  As its name implies, it is entirely related to the visual aspect of the entity. This implies having control and decision-making regarding the arrangement, order, and width of different fields and containers, interactive buttons, titles, labels, and informative messages, among other topics. There is also the option to display or hide fields and containers and the latter can be seen expanded or not.

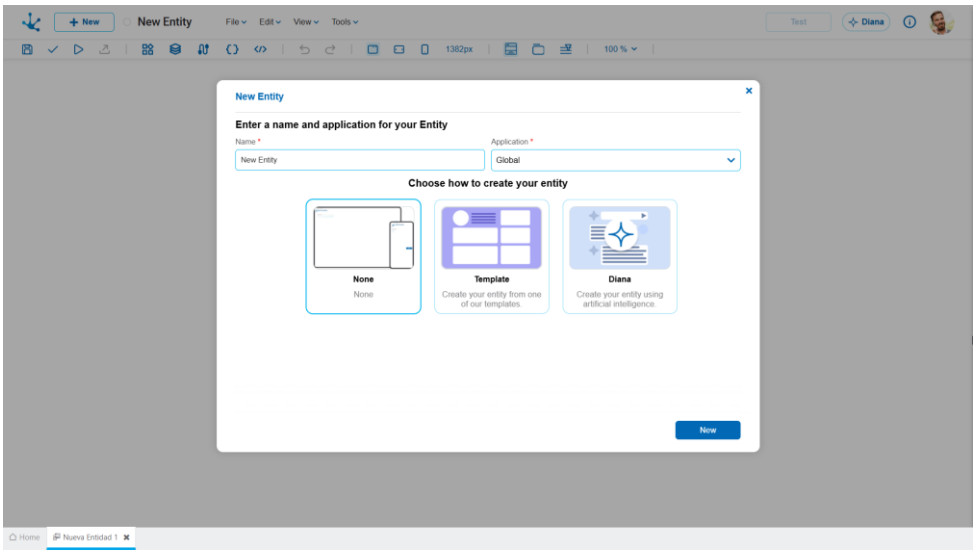
Ways to Create the Entity

When creating the entity, it opens a panel that allows defining some of its general properties and choose from the following options:

- Blank canvas
  The modeler starts without selecting a template.

- Built-in-template
  It allows creating an entity from a built-in template, which can be selected from a set of options. Once chosen, the template is displayed in the graphic modeling area of the new entity.

- Diana
  Use the entity modeler and **Diana**, the assistant that generates the entity in an AI-assisted manner.



Workspace

- Entity Information
  - State
  - Name

- Expanded Menu

- Diana

- Top Toolbar

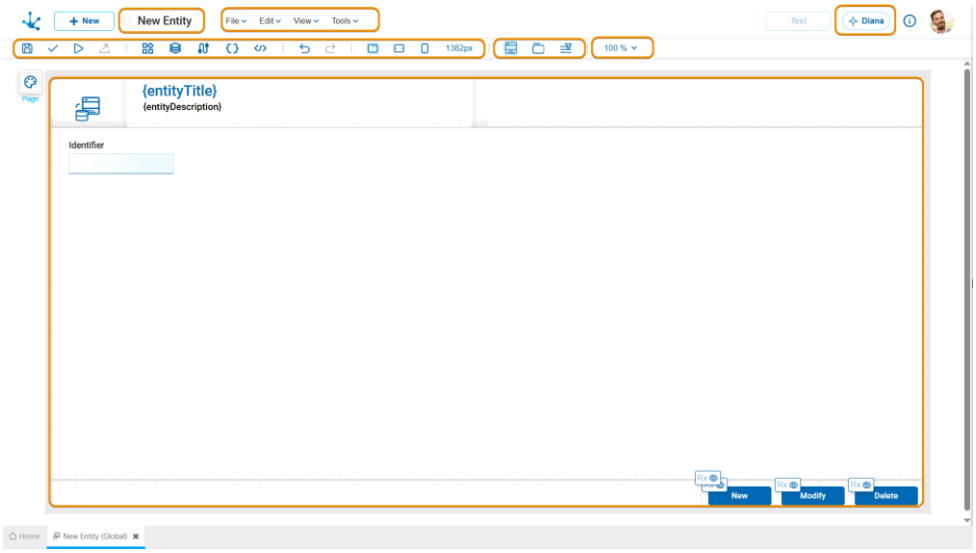
- Design Options

- Modeling Area
  - Section
  - Identifier

Every new entity contains an identifier field defined by default within the graphic modeling area

It has the following features:

- Its data type is an integer with a length of 10.
- It is autonumeric, increasing value, starting from 1.
- It cannot be deleted.
- The only behavior property that can be modeled is Visible.
- It cannot be included within a repeater element.
- The Initial Value and Contents Format properties cannot be modeled.
- Relations cannot be modeled.



3.4.11.1.1. Create with AI

Generation of Entities with Artificial Intelligence

Deyel can use artificial intelligence to generate a complete entity based on a request, simply by providing a definition of the entity to be modeled. The result complies with modeling standards, generates the entity grid including utility filters, and uses the application context for the fields modeling related to other entities and value lists.

AI Entities Generation Wizard

Generating entities with AI saves time by creating from scratch, using simple descriptions in natural language, customizing through prompts and using all the advantages that AI provides to model an entity, respecting conventions and good modeling practices. This wizard can be used from the entity creation panel.

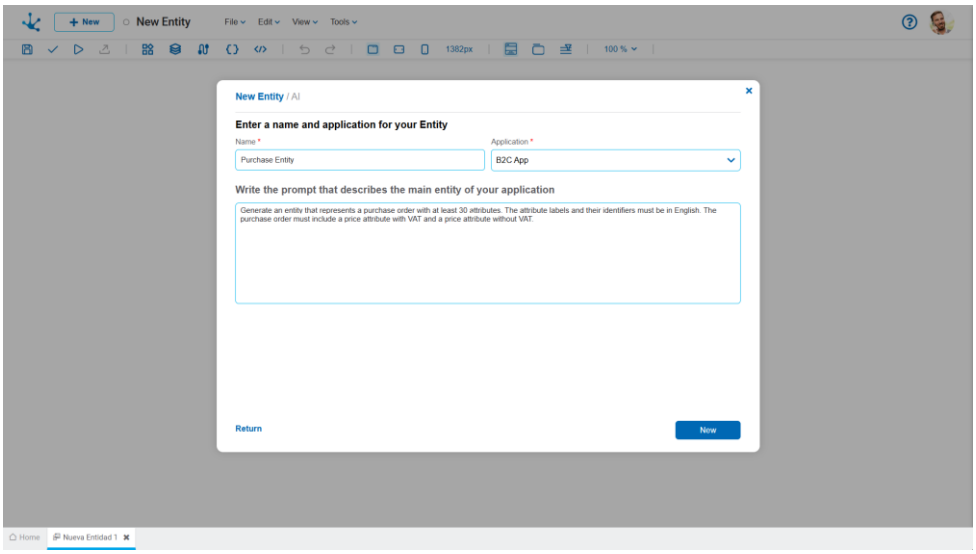
New Entity

When creating an entity, once the "AI – Create your entity using artificial intelligence" option is selected, a panel opens with the AI entity generation wizard, displaying a set of properties to fill in.

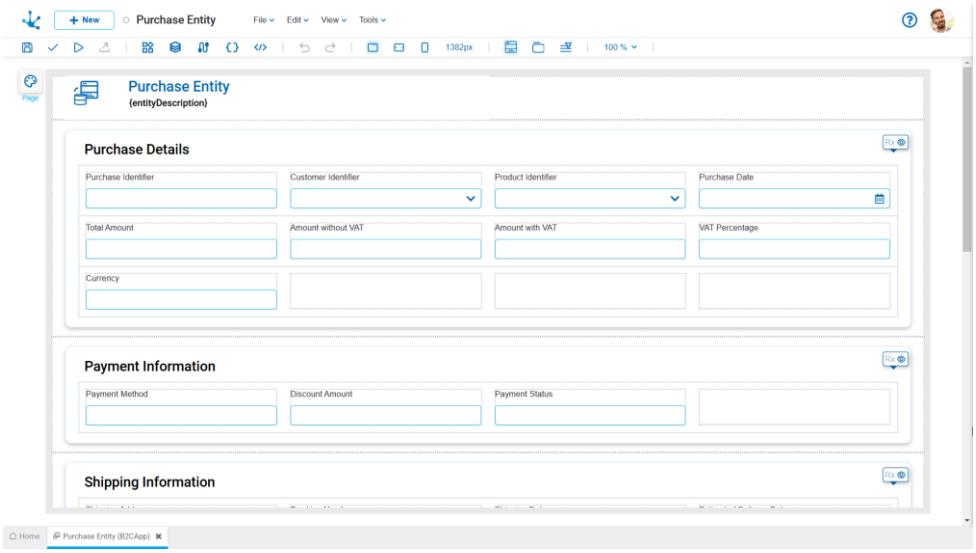
Name
It is used at the modeling level to reference the entity.

Application
Allows to define the application to which the object belongs. If no application is reported, the object is assigned to the application "Global".

Write the prompt that describes the main entity of the application
It allows the input of a detailed description written in natural language, specifying the features, functionalities, or desired design for the entity. The prompt is used by **Deyel** to generate the entity according to the instructions provided.



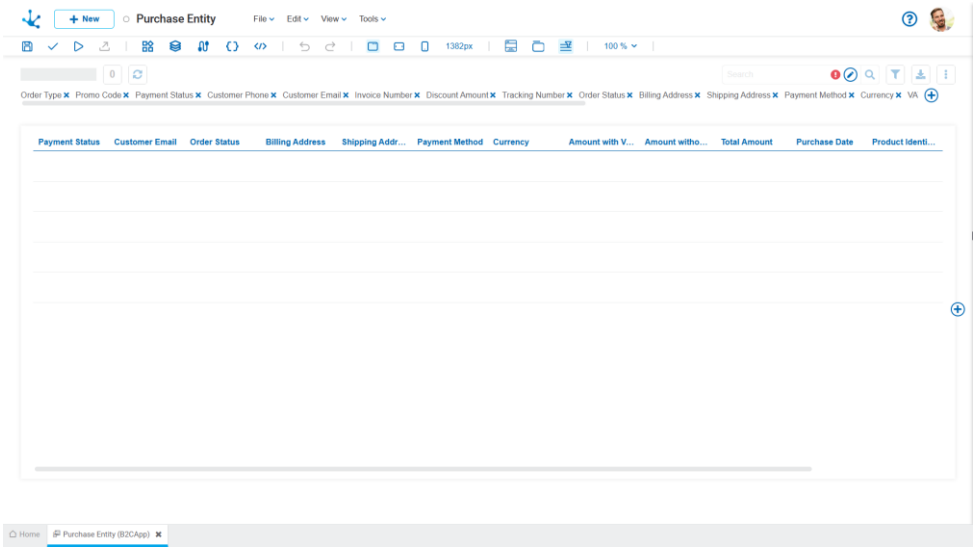
By clicking on the “New” button within the AI entity generation wizard, the entity that meets the prompt requirement is automatically obtained.



Entity generation not only allows the creation of freely chosen attributes, but also includes specific fields as indicated in the prompt.

Grid and Filters

Additionally, the grid of the entity is generated, where the columns are generated based on the most relevant fields, and filters are generated, based on fields that are relevant for performing searches.



Breakpoints

It is important to note that the generated entity is optimally displayed in all resolutions and breakpoints.

Generation in the Context of an Application

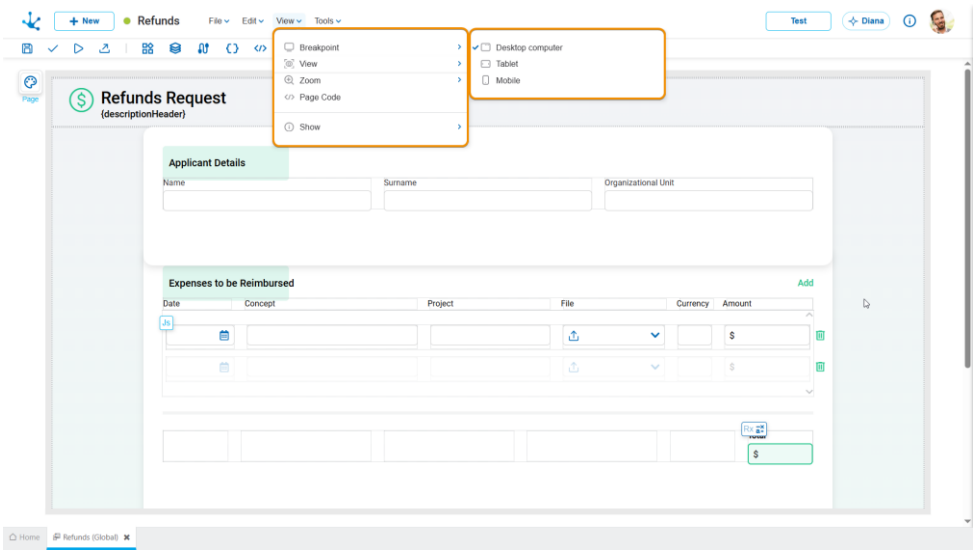
When generating an entity, the application's contextual entities and value lists are used. When an entity is generated, it implicitly inherits the context of the application in which it is defined. This enables automatic modeling of relations with other previously defined entities and value lists.

The entity is generated with the relations defined to the existing objects:

- Entities
- Value List

3.4.11.1.2. Expanded Menu

It is a horizontal list of options containing vertical submenus with different operations on the entity or on its modeling. In turn, each option in a submenu may expand a dependent submenu.

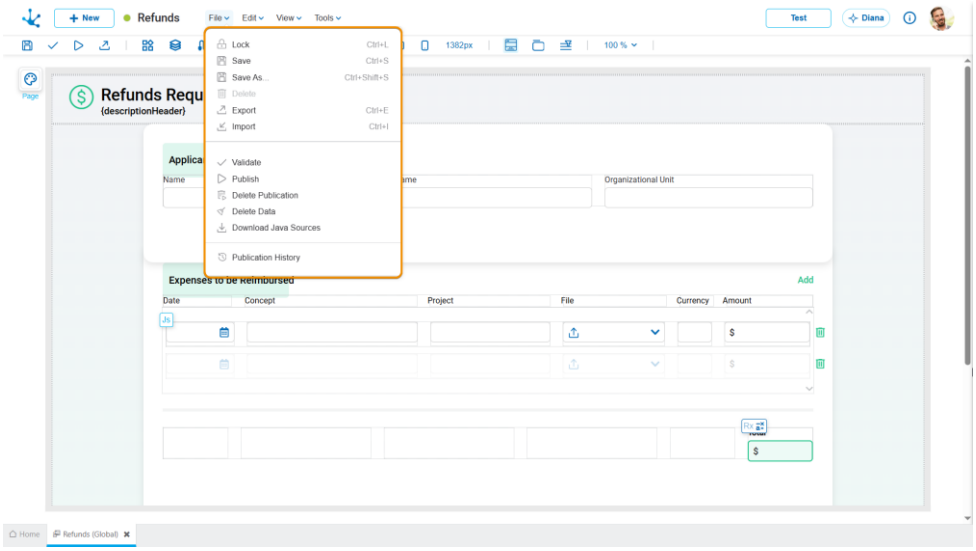


The expanded menu consists of:

- File Submenu
- Edit Submenu
- View Submenu
- Tools Submenu

3.4.11.1.2.1. File Submenu

This submenu opens by clicking on the “File” option and allows performing operations on the entity.



Save

This icon allows to save the object in the repository of **Deyel**, leaving its state as "Draft" or "Modified". If certain conditions are met, the modeler user receives a message indicating that the operation was performed correctly, otherwise they receive an explanatory message.

Conditions

- The application of the object must exist.
- The name must be unique in the application.
- The object permissions are required.
- The fields modeled in process activities as required should not be deleted, if they are used as parameters in automatic activities, in flow messages or conditions, in embedded rules of other fields in the form or in process activities, or as related attributes in entities

Save As

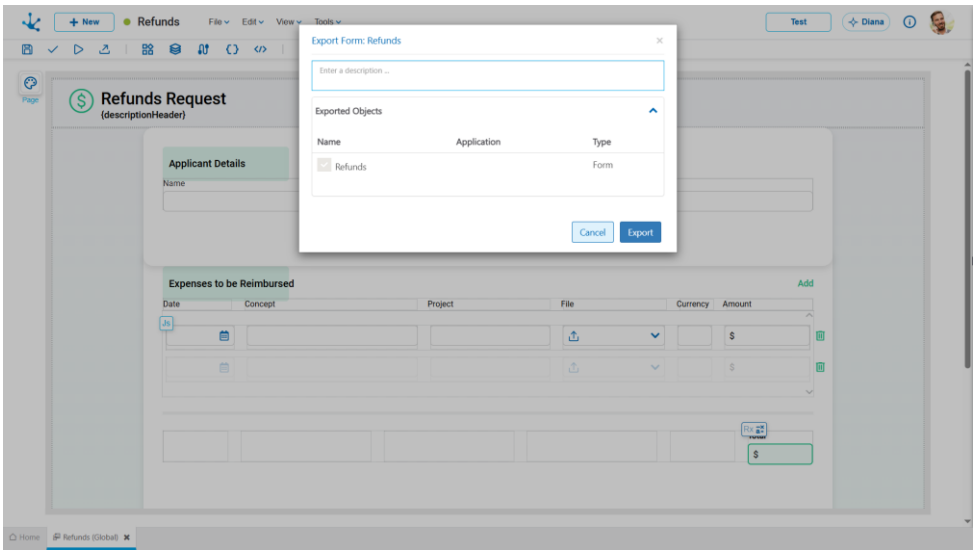
It allows creating a copy of the modeled entity by entering a new name and selecting the application to which it belongs. The copy opens in a new tab in state "Draft".

Delete

It allows to delete the object only if it is in "Draft" state and does not have other associated objects of **Deyel** that were previously saved or published. When deleted, the tab it is on closes and the object is removed from the modeler's grid.

Export

This icon opens a window for the user to select and confirm the export of the object.



Properties

Description

In this property a text explaining the reason for the operation can be entered.
This text can be modified upon import and is displayed in the description column of the export record.

Exported Objects

By expanding the container, the objects related to the form being exported are shown. Objects not meant to be exported are unchecked.
The related objects that can be exported along with the form are:

- Value lists associated with fields.
- Advanced rules used in fields, from the "Relation" of the field properties

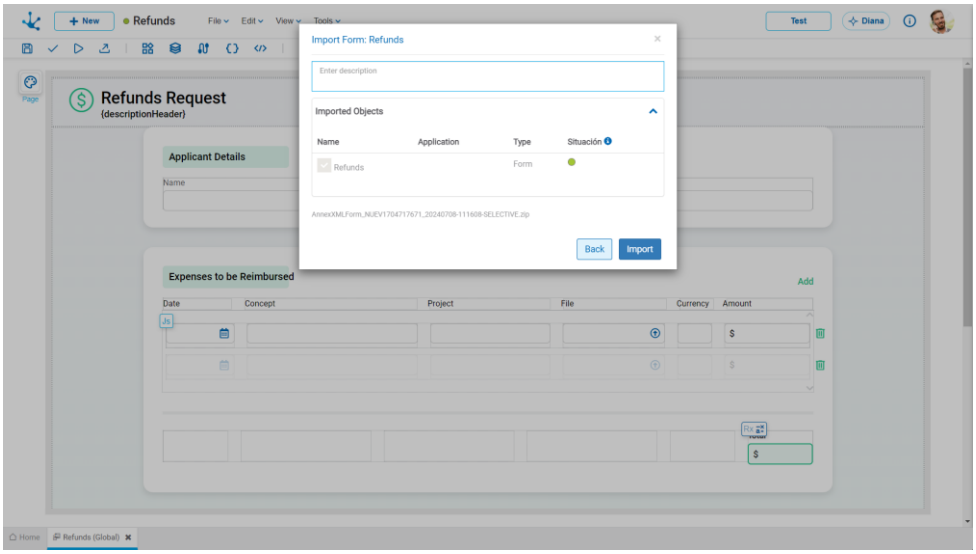
Click the "Cancel" button to abort the export without effect or the "Export" button to complete the operation.

> *Entities corresponding to related entities are not included in the export.*

Import

It allows to open a window for the user to select and confirm the import of the object.

Performing this operation is equivalent to using the import functionality from the button "Create" in the different views of the modeler.



Validate

This icon allows to validate if the object is ready to be published, that is, the same validations are carried out as when publishing and the result of them is reported.

Publish

Through this icon the object changes to “Published” state and the modeler user receives the corresponding message, indicating the result of the operation. The condition for publishing is the same as for the "Save" operation.

Conditions

- The entities and related rules must be published.
- Relations to published processes should not be removed.
- If new instances are created while modeling an entity already published without data, the entity cannot be republished.
- If instances are created while modeling an entity, it cannot be saved or published without reopening it in the modeler. Another option to save or publish the modeling in progress is to delete instances.

Delete Publication

Allows to leave the object unavailable for use by returning it to the "Draft" state.

Delete Data

Instances generated when using the entity and its attached files, if any, are deleted.

> *If instances of an entity are related to process cases related to the entity, those cases must be deleted first using the option Delete Data from the file submenu of the process modeler expanded menu.*

Download Java Sources

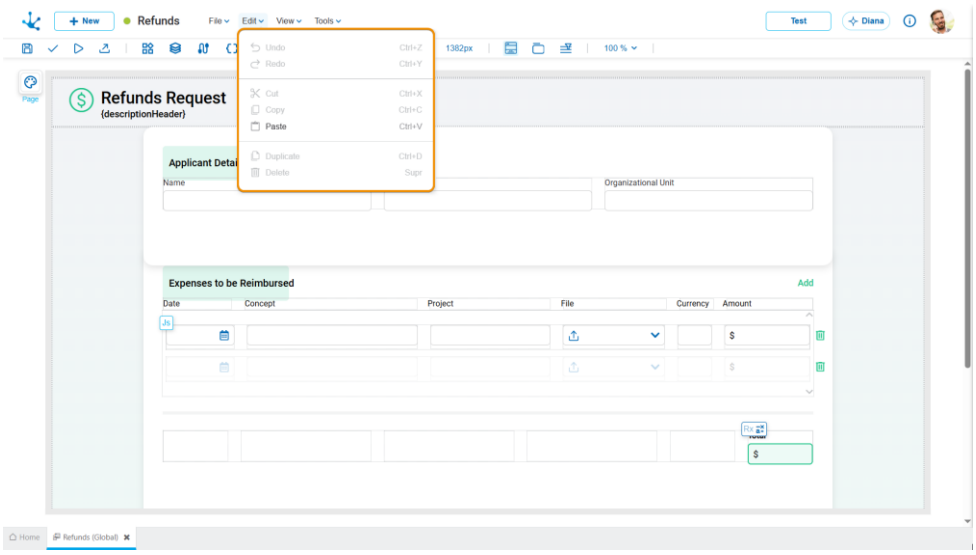
This icon allows to download the Java files that represent the object's model and service, so that it can be used in advanced rules.
Pressing the icon displays a message to confirm file download.

Publication History

This icon allows managing versions of published objects, displaying their history, to have better control of the changes that each object has.

3.4.11.1.2.2. Edit Submenu

This submenu is opened by clicking on the “Edit” option and allows the performance of operations within the modeler.



Undo (Ctrl+Z)

Allows to easily reverse the changes made in the modeler.

Redo (Ctrl+Y)

Allows to easily reapply the changes made in the modeler.

Cut (Ctrl+X)

Allows to send the selected element to the clipboard while deleting it from the graphic modeling area.

Copy (Ctrl+C)

It allows to copy any application element and temporarily place it on the clipboard.

Paste (Ctrl+V)

It allows to take the item from the clipboard and place it elsewhere in the graphic modeling area.

Duplicate (Ctrl+D)

It allows to make a copy of any application element within the graphic modeling area.

Copy styles (Ctrl+Alt+C)

It allows copying all current configurations from the styles panel of the selected element.

Paste styles (Ctrl+Alt+C)

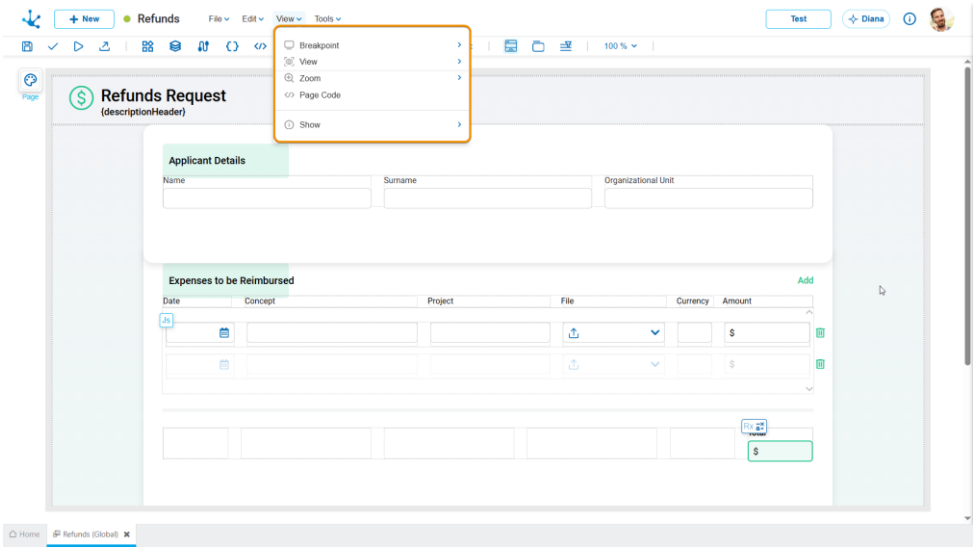
It allowsto apply the previously copied configurations to another element of the same type.

Delete (Del)

It allows to delete the selected element from the graphic modeling area.

3.4.11.1.2.3. View Submenu

This submenu opens by clicking on the "View" option and allows selecting different ways of displaying the entity.



Breakpoint

They ensure that users always view the best version of their entity, regardless of the device they are using.

The modeler includes the most common breakpoints (desktop, tablet, and mobile), allowing the entity to adapt to individual screen sizes by defining what takes priority in the layout. Styles can be rearranged, elements can be shown or hidden, and the layout can be customized for each breakpoint.

By clicking on “Breakpoint”, a secondary submenu is displayed, where the following options can be selected:

- Desktop Computer
- Tablet
- Mobile

Zoom

Selecting "Zoom" allows adjusting the viewing size of the modeling area, using built-in options such as 25%, 50%, 75%, 90%, 100%, 125%, 150% and 200%. The shortcut Ctrl+ to zoom in and Ctrl- to zoom out can also be used. Another option to adjust the size is to hold down the Ctrl key and use the mouse wheel.

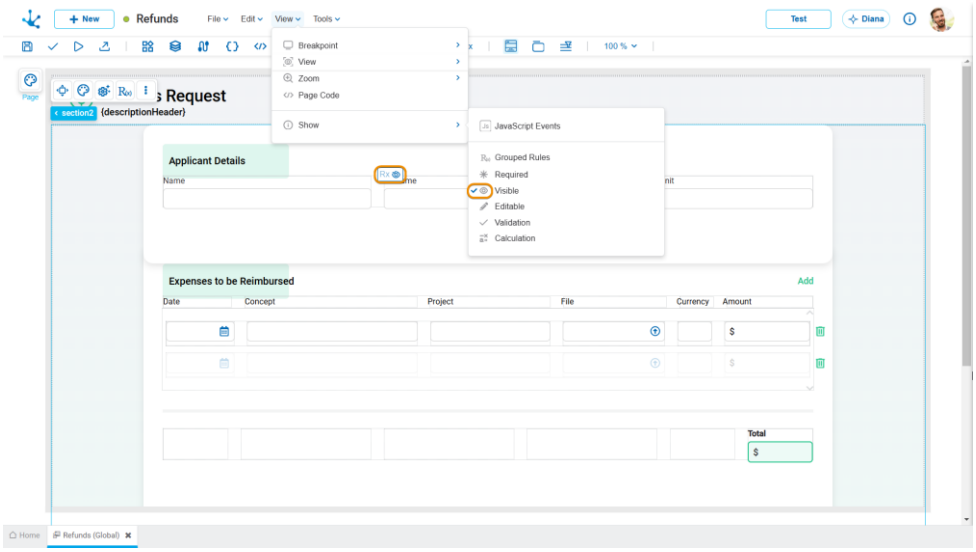
Page Code

Selecting this option opens or closes an area for code editing at the bottom margin of the modeling area.

Show

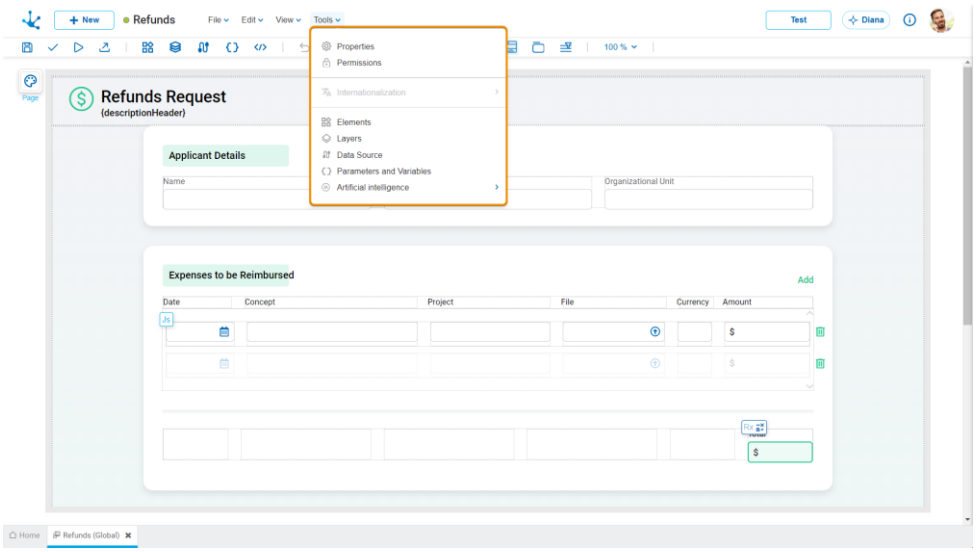
This functionality allows showing or hiding the set of icons that indicate the behavior that affects the elements. Clicking on the "Show" option displays a secondary submenu, where the icons corresponding to the modeled rules that should be displayed for each element can be selected.

- JavaScript Events
- Grouped Rules
- Required
- Visible
- Editable
- Validation
- Calculation



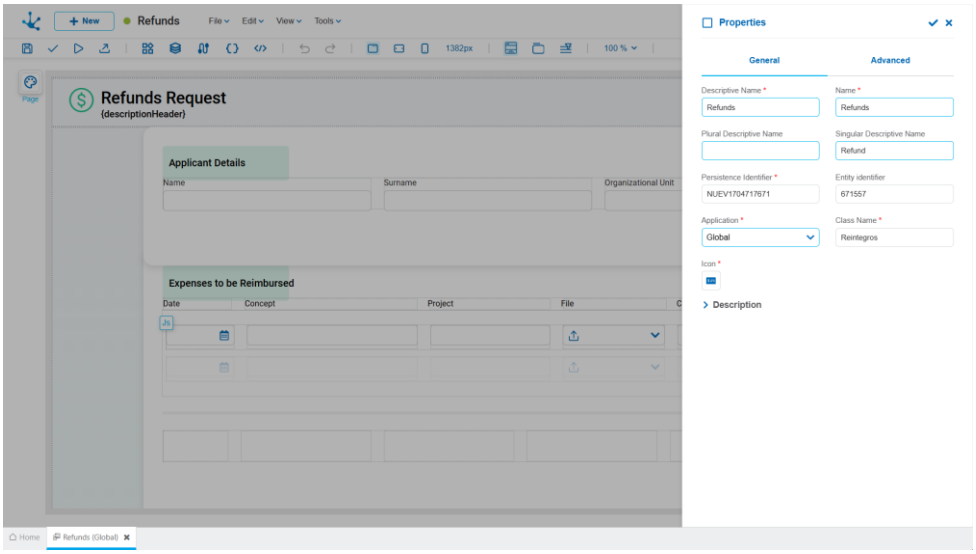
3.4.11.1.2.4. Tools Submenu

This submenu is opened by clicking on the "Tools" option and it allows modeling entity properties and permissions, as well as different modeling functionalities.



Properties

This option allows opening the properties panel of the entity to model its features.



Permissions

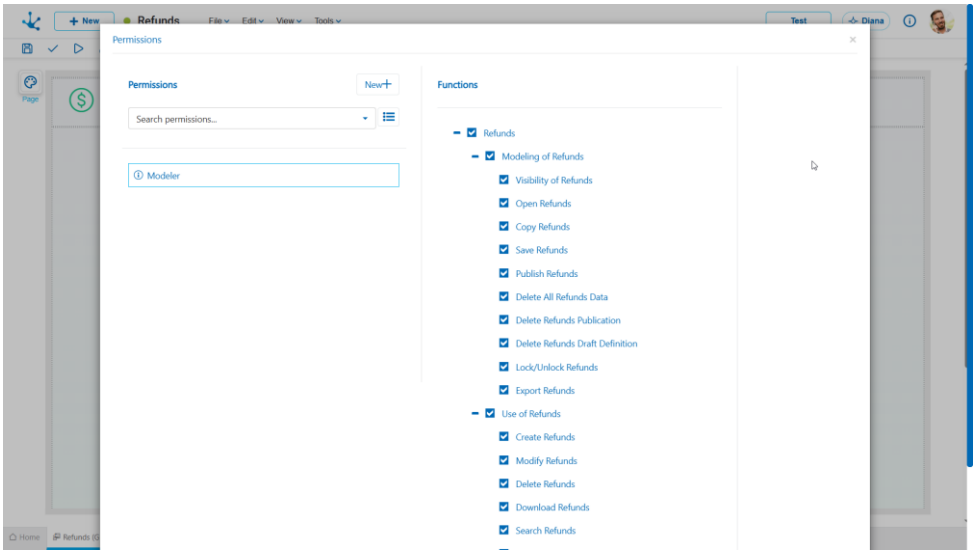
Allows to assign the security functions for use and modeling of the object to the existing permissions or to new permissions that the user creates, without having to go to the permission settings option from the menu.

Sections

- Permissions: Permissions to which object functions are assigned.
- Security Functions: Represents the total set of security functions, modeling and use of the object. Those that are marked are the ones included in the selected permission.

By default, all security functions for a new object are assigned to the permission Modeler.

Users who are assigned the permissions have access to the functions included in it.



New Opens a panel to create a new permission and once created, the security functions included in it must be selected .

Search permissions... Allows to select a permission from a list and enables the input of characters to filter the values in the list.

Opens the wizard to select a permission and once chosen, the necessary object security functions must be checked.

To unrelate a permission from the object, hover the cursor over the permission and press the icon . If there are functions selected for that permission, they must be unchecked in order to delete it.

Modeling Security Functions

- Visibility: Allows to display the object in the Deyel modeler grid.
- Open: Allows to open the object from the Deyel modeler.
- Export: Enables the operation to export the object from the tools submenu of the expanded menu.
- Save: Enables the operation of saving modifications made to the object.
- Publish: Enables the operation of publishing the object leaving its state as "Published".
- Delete publication: Enables the operation of deleting the object publication leaving its state as "Draft".
- Delete draft definition: Enables the operation of deleting the object.

Use Security Function

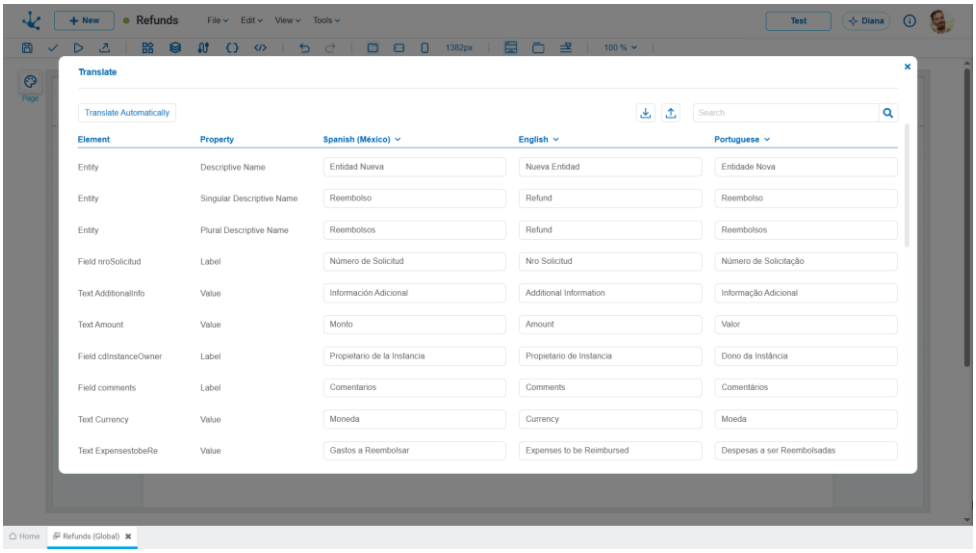
- Create: Enables the operation of creating an instance of the object.
- Modify: Enables the operation of modifying an instance of the object.
- Delete: Enables the operation of deleting an instance of the object.
- Download: Enables the operation of downloading the instances of the object in Excel.
- Search: Allows using the grid to show the entity instances.

- Show: Enables the operation of showing an instance of the object.
- Show/Modify/Delete Private Instance: They enable user operations on a private instance, regardless of the organizational unit they belong to and the rest of the criteria that define the privacy.

> *The security functions of private instances are displayed if the following properties are checked in the entity* Hierarchical Privacy *and* Privacy by Permissions.
> *When an entity is configured as anonymous, the use function is automatically assigned to the "Anonymous" permission.*

Internationalization

The "Internationalization" option is enabled when the property Internationalize is activated. To access the translation panel, it is necessary to select "Internationalization" and then choose "Translate".



In this panel, all the translatable properties of the object are displayed, along with their content in the original language, and translations in the other available languages can be entered manually.

If a value is modified in the original language column, the change is applied directly to the object's property.

In the header of each language column, a menu that allows selecting another language can be displayed, updating the column with its contents.

Options Available in the Top Margin

Translate Automatically Generates translations from the original language to other languages using artificial intelligence.

Downloads an Excel file with the contents of the translation panel.

 Allows selecting an Excel file to import translations, making it easier to manage large volumes of content.
 Filters the contents of columns based on the characters entered in the search field.

Elements

This option allows opening the elements panel that enables selecting those elements that will be incorporated into the entity.

Layers

This option allows opening the layers panel of the entity, and displaying its elements organized hierarchically.

Data Source

This option allows opening the data source panel from where ,the data sources that will be used on the entity can be defined.

Parameters and Variables

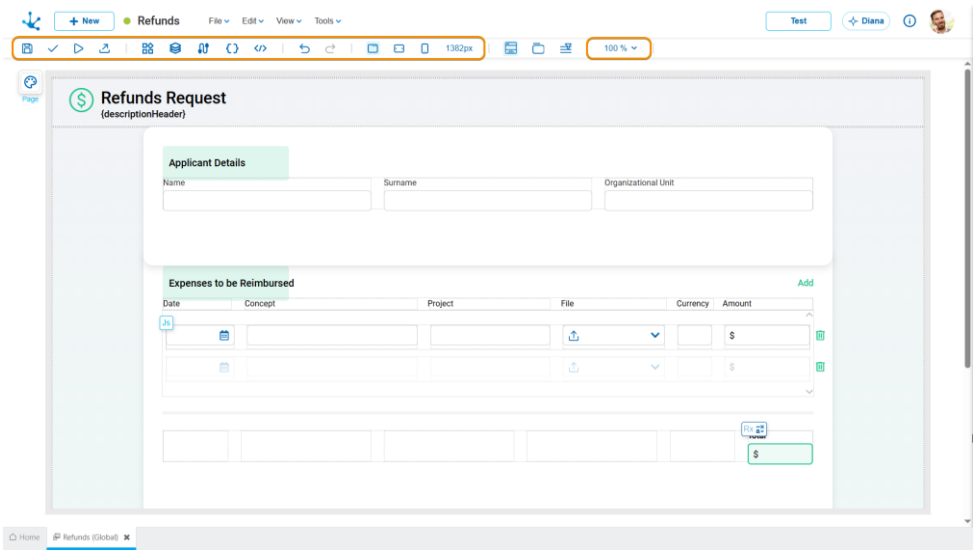
This option allows opening the parameters and variables panel, from where the ones that will be used on the entity can be defined.

Artificial Intelligence: Generation of Instances with AI

This option allows opening the Generation of instances with AI allowing automatic creation of instances of the modeled entity using artificial intelligence.
The artificial intelligence engine integrated with **Deyel** is capable of generating this data, which can be used in the design and testing stages, during the application development cycle.

3.4.11.1.3. Top Toolbar

Contains icons for quick access to the most used operations of the expanded menu.



File

- Save
- Validate
- Publish
- Export

Tools

- Elements
- Layers
- Data Source
- Parameters and Variables
- Page Code

Edit

- Undo
- Redo

View

100 % Zoom Percentage

Breakpoints

- Desktop Computer
- Tablet

Mobile

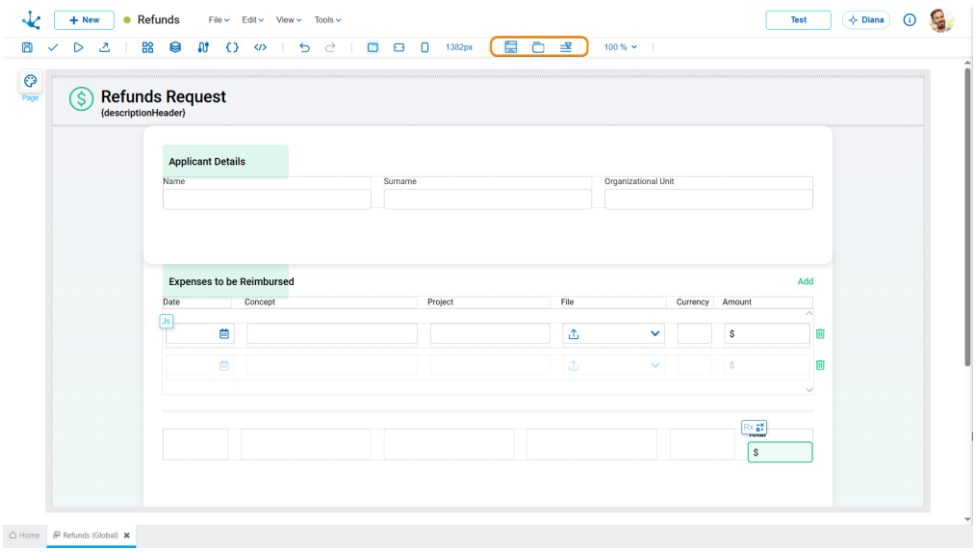
px Pixels

3.4.11.1.4. Design Options

When modeling an entity, it is necessary to define the set of fields it comprises, the type of information stored in each one, their arrangement within the entity, and the features with which they are presented in the interface. This allows working on both the data model and its display simultaneously.

The following design options to display the different modeling aspects of an entity can be selected in the top toolbar:

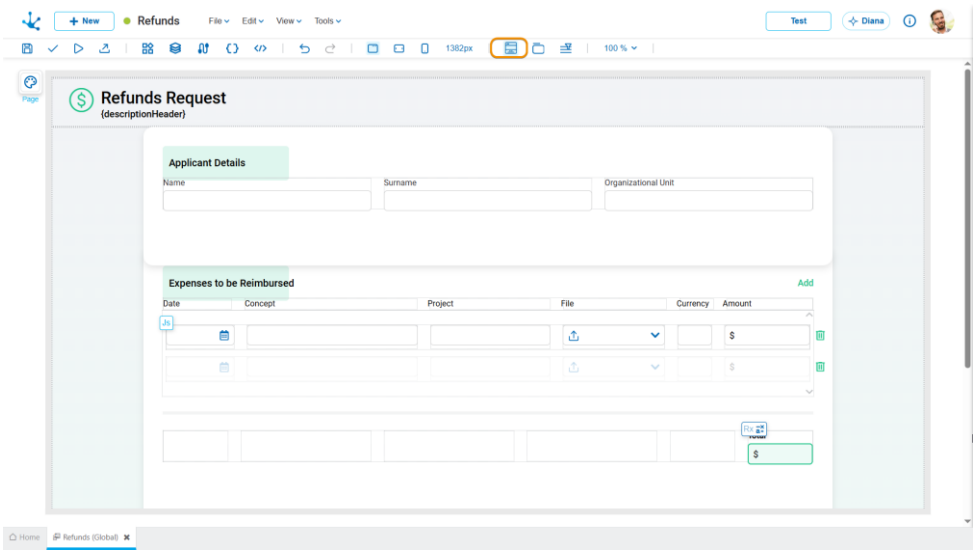
Entity View
Fields
Grid and Filters



3.4.11.1.4.1. Entity View

This design option corresponds to the edition of the entity in the modeling area. In this view the graphical interface of the entity is modeled and the properties of the fields that make up the entity can be accessed.

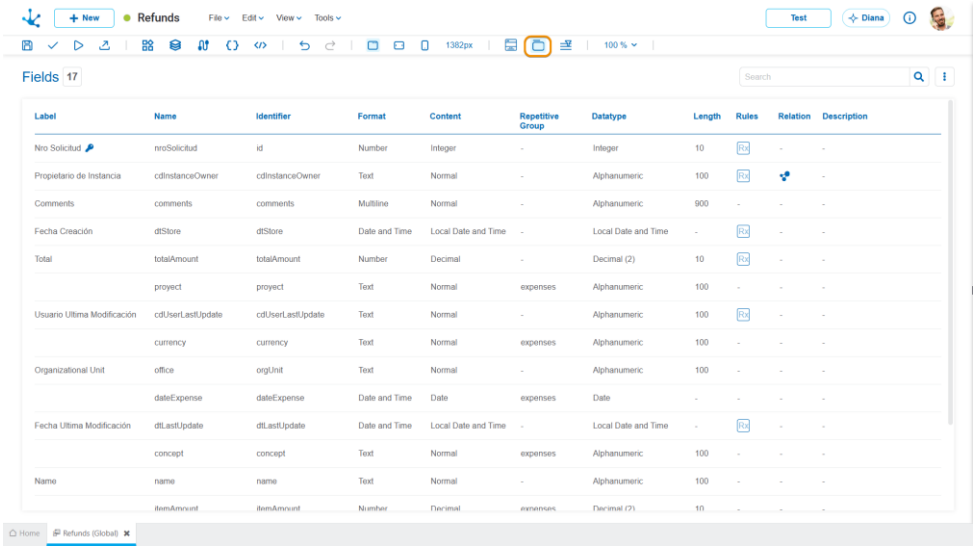
It allows modeling the appearance, by dragging and dropping elements, and the set of fields that define the business attributes of the entity. This allows working on both the data model and its display simultaneously.



3.4.11.1.4.2. Fields

This design option corresponds to a grid where all the fields of the entity are displayed. At the top left, a totalizer is displayed with the number of defined fields. The grid features a quick search, filters and the functionality to sort, detailing the properties of each field and allowing operations to be performed on them.

It allows having a quick view of the configuration properties of each field, such as label, name, identifier, description, format, whether it belongs to a repetitive group, data type, maximum content length, if It has relationships to other objects and if it has embedded rules.



Columns

Name

Name assigned to reference a field when modeling, allowing the field to be uniquely identified within the entity. Used in rule wizards to refer to the field within the conditions. It can be modified by the user, and does not allow spaces or special characters.

Identifier

It is the name assigned to reference a field in the programming code. It is used to refer to the field within the Java code in the tools menu option of advanced rules and in the JavaScript code in the page code area of the entity modeler. It allows to uniquely identify the field within the modeled entity. It generates automatically from the value entered in the property Name . It can be modified by the user, as long as no data has been uploaded into the entity and it does not allow spaces or special characters.

Label

It allows entering the text that is displayed on the field. Supports white space.

Format

Defines the general way in which field data is stored and managed.

Possible values

- Text
- Number
- Multiline
- Date and Time
- File

Content

Defines the different types of data or information that can be entered and stored under a specific format, specifying the variations and details allowed within the general format.

Possible values depending on the specified format

- Text
  - Normal
  - Password
  - Mail
- Number
  - Decimal
  - Integer
- Date and Time
  - Date and Time
  - Date and Local Time
  - Time
  - Local time
  - Date
  - Local Date

Repetitive Group

If it has a value, it indicates the identifier of the repetitive group to which the field belongs.

Data Type

The data type is displayed only in this grid and is directly related to how the information is stored in the database of **Deyel**. This is technical data, which is useful for advanced modelers.

Length

Determines the maximum length of the value based on the data type. This property is only visible if the field is of text or numeric type.

Rules

If the field has any associated rule of requirement, visibility, editability, validation or calculation, an icon is displayed indicating the type of rule.

Relation

The icons in this column allow for quick identification of the relation type each field has, if any. The possible relations are:

 Entity

 Value List

 Advanced rule

Description

Text that defines the field and optionally its content.

Operations

By hovering the cursor over a line, icons are enabled to perform operations on the field.

 It allows deleting the field only in entities without data, and requires confirmation by the user.

 Opens the field properties panel, where they can be modified.

Data Type

Correspondence between the format, content and data type of each field of the entity being modeled can be seen in this table.

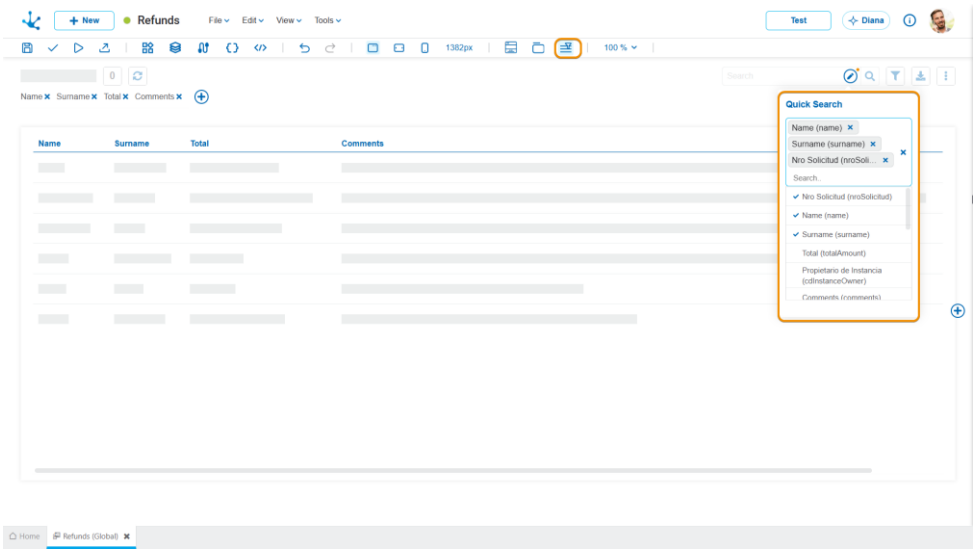
Format	Content	Data Type
Text	Normal, Password, Email	Alphanumeric
Multilínea	Normal, Password, Email	Alphanumeric, Extensive Alphanumeric
Number	Decimal	Decimal
Number	Integer	Integer, Large Integer
Date and Time	Date, Local Date	Date
Date and Time	Date, Local Date and Time	Date and Time
Date and Time	Hour, Local Hour	Hour
Checkbox / Toogle	Checkbox, Switch	Boolean
File	File	**Repository:** Database **Repository:** AWS S3 **Repository: AWS Public**

3.4.11.1.4.3. Grid and Filters

This design option corresponds to the way the entity instances are displayed and the functionality of searching on them.

Grid

It allows modeling the columns that are displayed in the results and search grids, when executing the entity.



New fields can be added by pressing the icon ⊕ to the right of the last column of the grid and its order can be modeled by dragging the shaded rectangles of the columns to swap their positions.

When hovering over the title of each column, the following icons are displayed:

It allows modeling the width of each column by moving the limit bar to the right to enlarge the column, or to the left to shrink it.

Deletes the column from the grid.

Establish the column by which the default sorting will be performed, it is ascending or descending depending on the selected icon.

It is displayed when a field has a relation modeled to a value list . Pressing the icon allows to select the content type to show in the column. The default value is "Text", which can be changed to "Icon" or "Icon and Text".

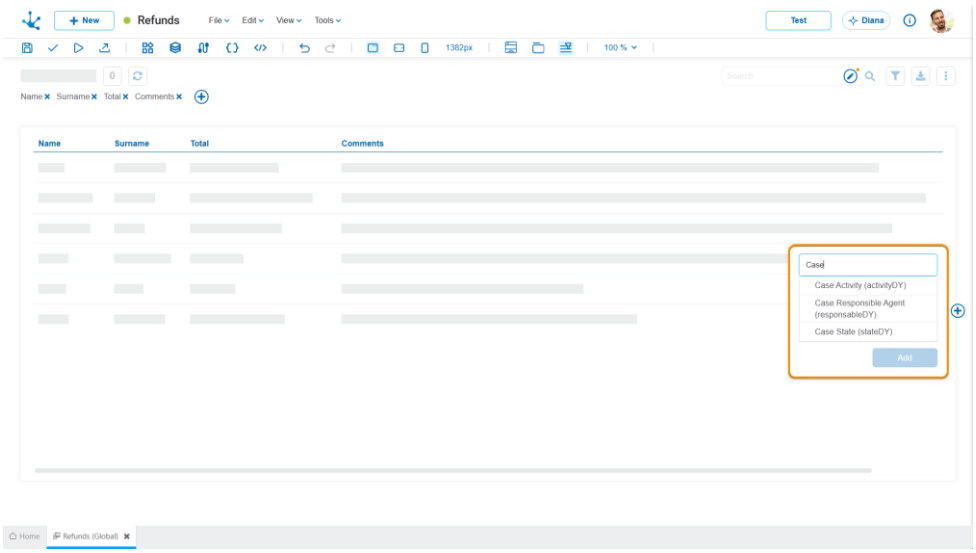
In entities with associated processes, columns related to the case execution can be added when modeling the result grid.

Case Activity
It is the activity of the process in which the case is.

Case Responsible Agent
Refers to the user, role or office that the case has pending on its task list.

Case State
It is represented by a circle of different colors.
- Active: Green
- Finalized: Blue

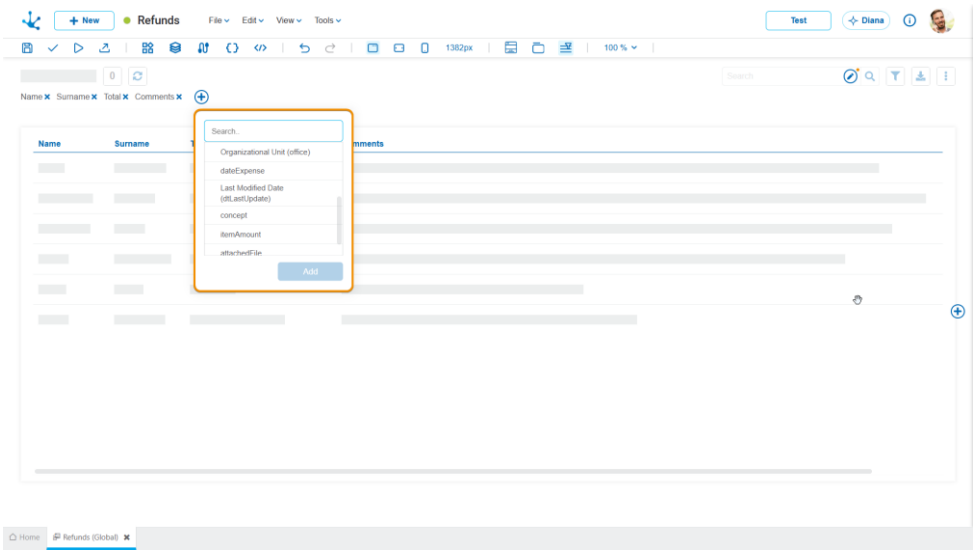
- Cancelled: Red
- Discontinued: Gray



Filters

The definition of filters allows modeling search criteria on the entity instances. Each filter corresponds to an entity field or to a field related to its execution. The latter correspond to the creation and last modification users, the creation and last modification dates, and the instance owner.

Pressing the icon ⊕ located in the filter line opens a panel to select the fields to be filtered by.



Quick Search

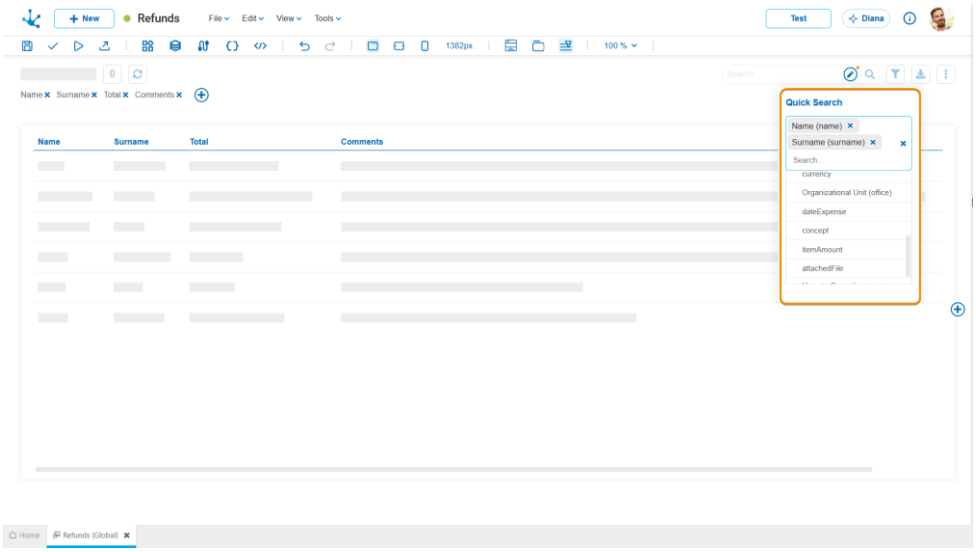
Quick search defines the filters that can be used in the search field when executing the entity grid.

The entity fields that are included in the quick search can be chosen when modeling the grid. The available fields are:

- Alphanumeric.
- Numeric.
- Objects-related such as value lists and forms.
- Instance owner.
- Audit fields (creation user and last modification user).

Fields can be added by pressing the icon . If this search is not modeled, it is not displayed in the grid when executing the entity. Hovering over the icon indicates that at least one field should be selected for the search to be visible.

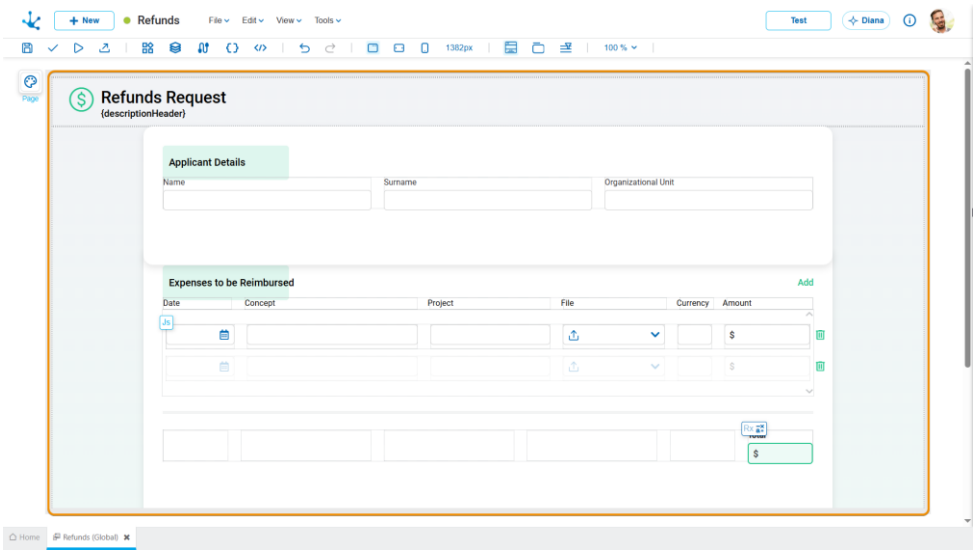
When selecting at least one field, the icon disappears and the icon displays, notifying the modeler that the quick search has been modeled.



3.4.11.1.5. Modeling Area

It is the space where entities can be modeled using the "drag and drop" functionality to incorporate different elements.

The modeling area of an entity is divided into sections within which elements can be added.



Properties

The general properties and the advanced properties of the entity can be opened in the right side panel of the modeling area.

3.4.11.1.5.1. Elements

An entity contains elements of different types, specific properties can be modeled for each of these types.

Some elements may contain other elements, in these cases they are called superior elements while the elements they contain are called inferior elements.

Superior Elements

- Containers
- Sections
- Design
- Repeater
- Slider

Operations

Add Elements to the Entity

The elements can be added to an entity via “drag-and-drop”, at any location within a superior element.

When adding an element, it automatically docks to the top or bottom border of a higher-hierarchy element, always to the nearest one.

When elements are added, their width is defined by default as a percentage, with the exception of the button element, which has an automatic width.

Modify Position of an Element

Once the element is added, its position can be changed by dragging it within the entity or by changing its structure and position properties with respect to the superior element.

If an element changes its position, this applies to all breakpoints.

Modify Element Size

The dimensions of an element can be modified by dragging the handles on its outline or by modifying its structure properties.

Guides

There are different types of guides that are displayed when selecting an element and pressing the “Alt” key.

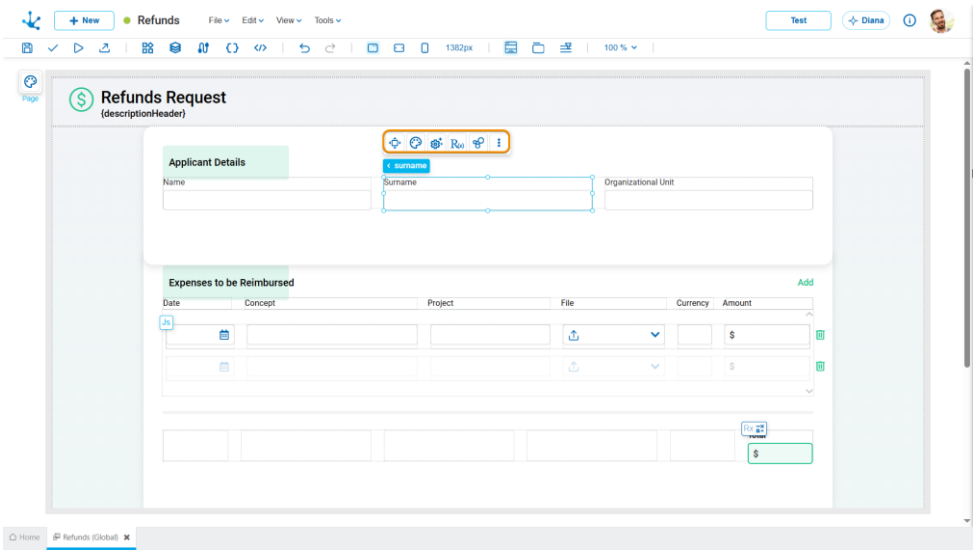
- Docking guides: they are displayed when moving an element and the position where it is moved depends on how close it is to its highest hierarchy element. In addition, the distance to the margins of the superior elements is indicated.
- Alignment Guides: they are displayed to help elements stay aligned.
- Guides to know the space between elements.

Properties

Each element has specific properties that are displayed using a context menu, which is enabled when selecting the element. Properties are grouped according to their functionality and are detailed for each element type.

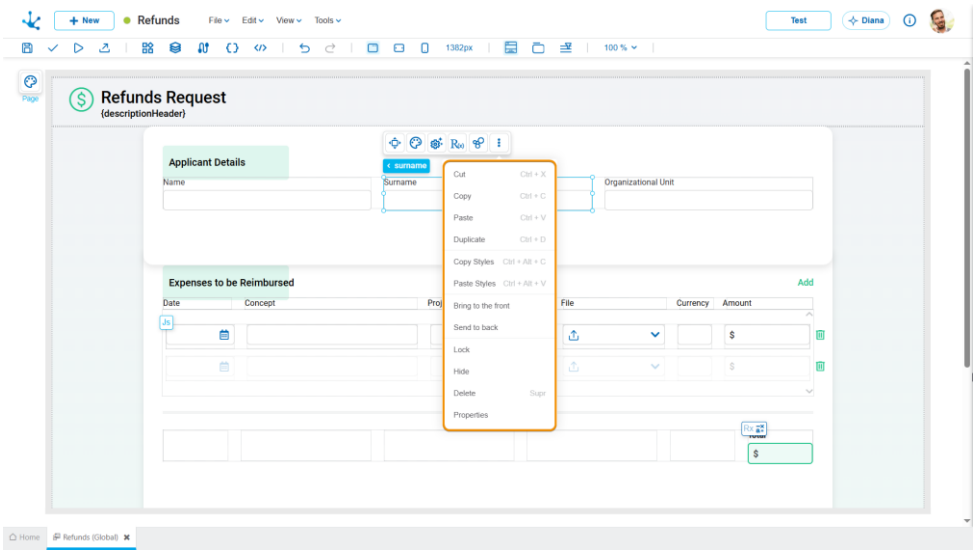
Context Menu

When selecting an element, a palette of icons is opened, where each one represents a group of specific properties of the selected element. The name is also displayed and when the mouse is slid over it, the path from the entity to the element can be seen, indicating the superior elements where the element is found.



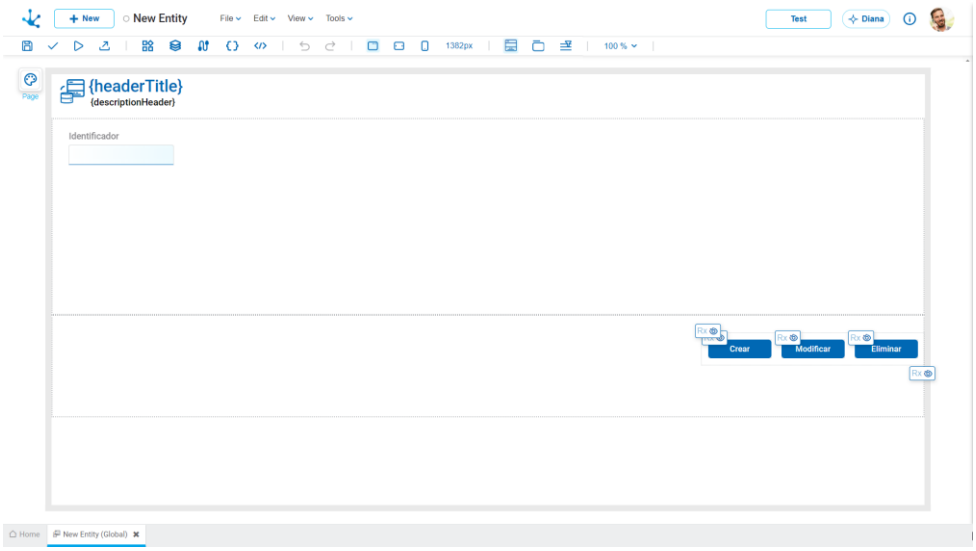
Clicking the right button of the mouse on an element expands a second context menu whose options correspond to operations that can be performed on the selected element. The same menu is displayed by pressing the icon ⋮ of the palette.

- Cut (Ctrl+X)
- Copy (Ctrl+C)
- Paste (Ctrl+V)
- Duplicate (Ctrl+D)
- Copy styles (Ctrl+Alt+C)
- Paste styles (Ctrl+Alt+V)
- Bring to front
- Send to back
- Lock/Unlock When an item is locked, it cannot be moved until it is unlocked.
- Hide: Hides the element in the active breakpoint and in smaller breakpoints. Once hidden, it can be shown again from Layers.
- Delete (Del)
- Properties



Section

Entities are divided into sections where elements can be included. By default, the entity has three defined sections.



Operations

Add

It allows adding a new section by clicking on the icon ⊕ of an existing section.

Modify Size

The section height can be adjusted from its bottom border, by clicking on it and dragging it. Another way to modify its size is from the structure panel of the section.

Move

A section can be moved using the functionality of layers.

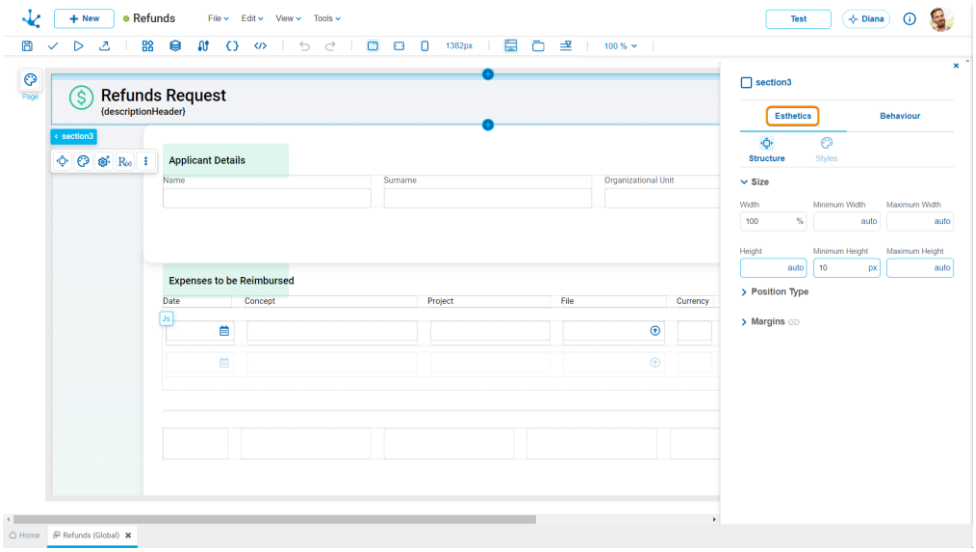
Classification of Properties

Properties are divided into two groups: esthetic properties and behavior properties.

Esthetic Properties

In the esthetic properties panel, the following are grouped:

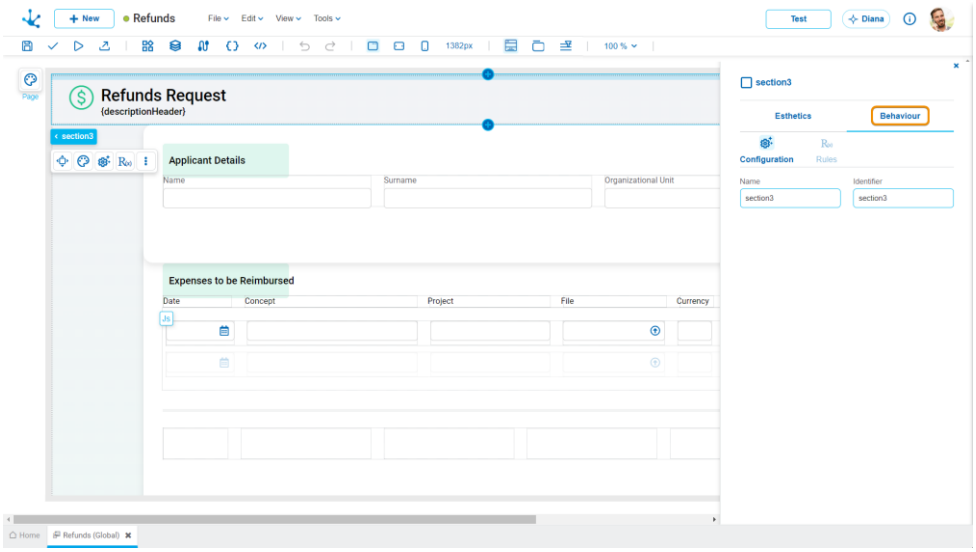
Structure Properties
Style Properties



Behavior Properties

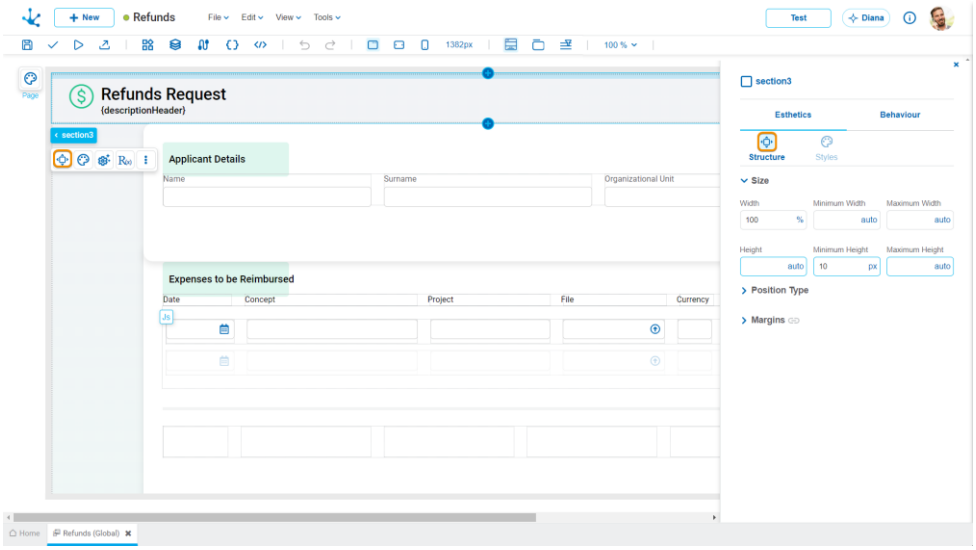
In the behavior properties panel, the following are grouped:

- Configuration Properties
- Rule Properties

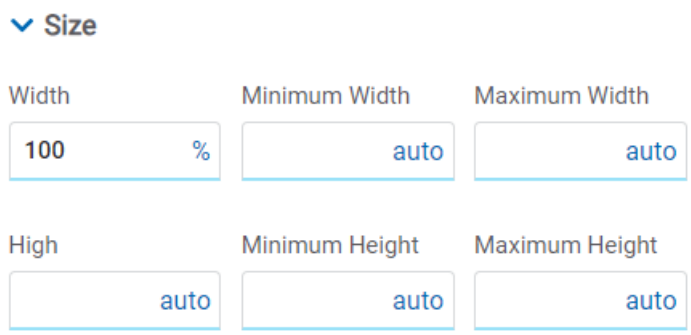


Structure Properties

The structure properties panel of a section opens when selecting the icon  of its context menu.



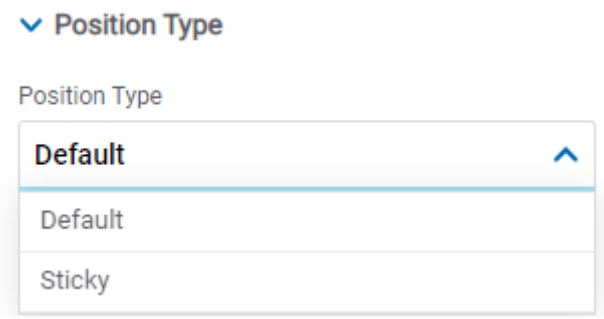
Size



All size properties can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh). Additionally, for Width and Height properties, the “auto” option is added.

> *Sections do not allow modifying the Width property.*

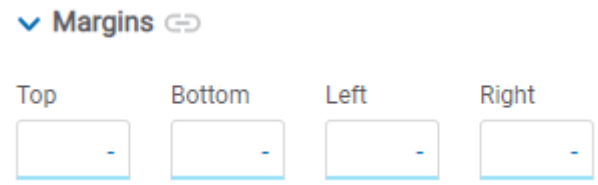
Position Type



Possible Values

- Default: The element has a relative position for the superior element where it was placed (container or section).
- Sticky: It is available only in sections, it allows the section to remain visible while users scroll through it. At first, the section is seen where it has been positioned in the entity, but it "sticks" to the screen as it is scrolled down.

Margins



It allows to define the distance of elements from the borders of their top element. The behavior of margins depends on the docking of the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Top
Distance to the top border of the highest ranking element.

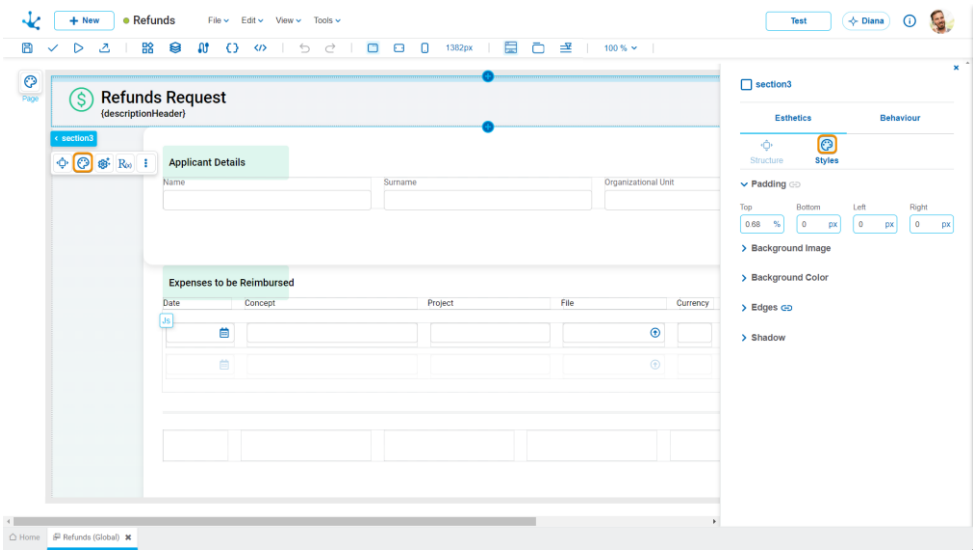
Bottom
Distance to the bottom border of the highest ranking element.

Left
Distance to the left border of the highest ranking element.

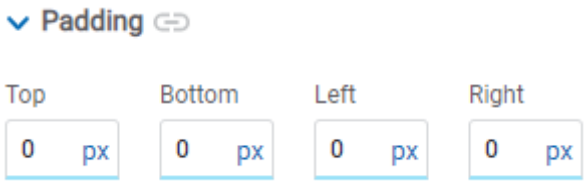
Right
Distance to the right border of the highest ranking element.

Style Properties

The style properties panel of a section opens when selecting the icon of its context menu.



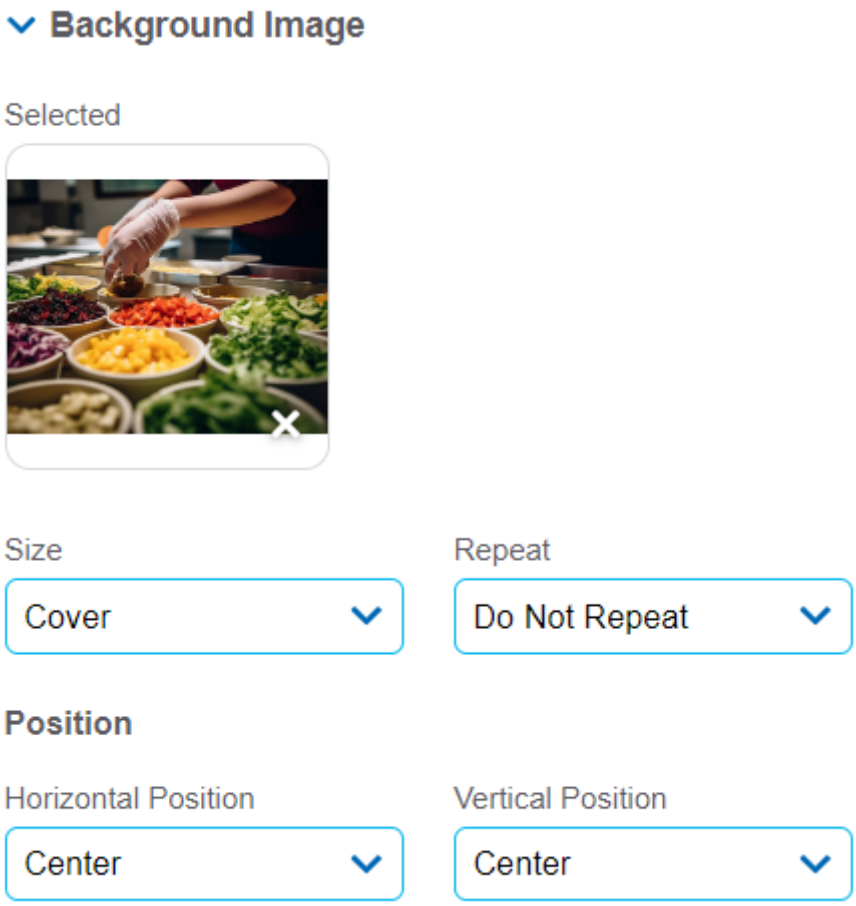
Padding



All padding space properties allow to create a space around the element borders (top, bottom, sides) and its content. The padding space configured is inward, while the margin is configured relative to a top element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Background Image

The element properties are represented by icons on its context menu, where its operations are also available.
PropsEstiloImagenFondo3



It allows to add a background image to the section.

Size

Possible Values

- Cover
- Content
- Auto

Repeat

Possible Values

- Repeat
- Repeat Hor.
- Repeat Vert.
- Spacing
- Rounded
- Do Not Repeat

Horizontal and Vertical Position

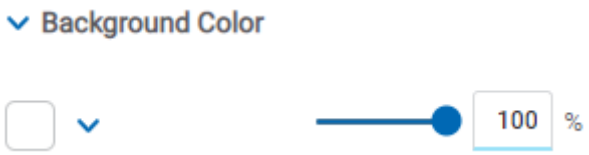
Possible Values

- Center
- Left
- Right

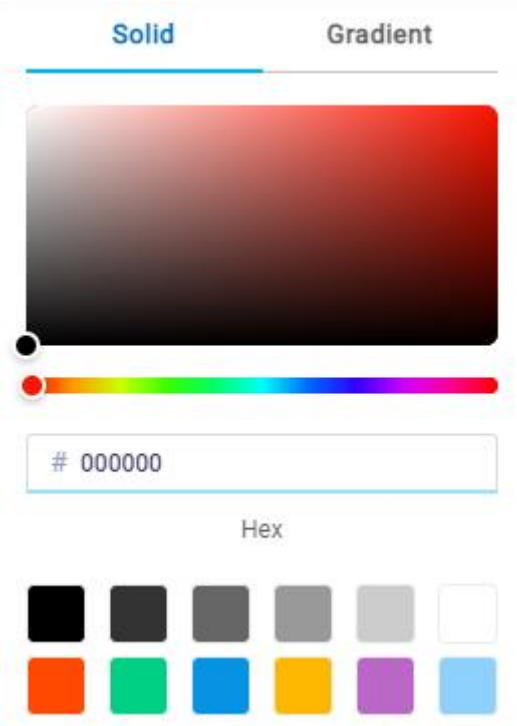
Background Color

It allows to model a background color for the element, it can be solid or gradient and different properties are defined for each one.

Solid



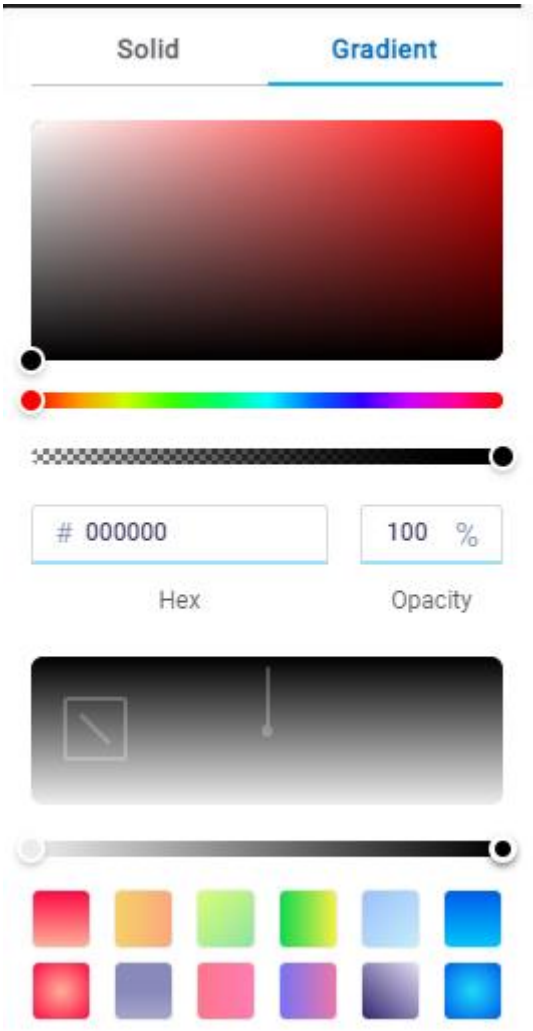
This option enables not only to select the color from a palette but also to define the degree of transparency.



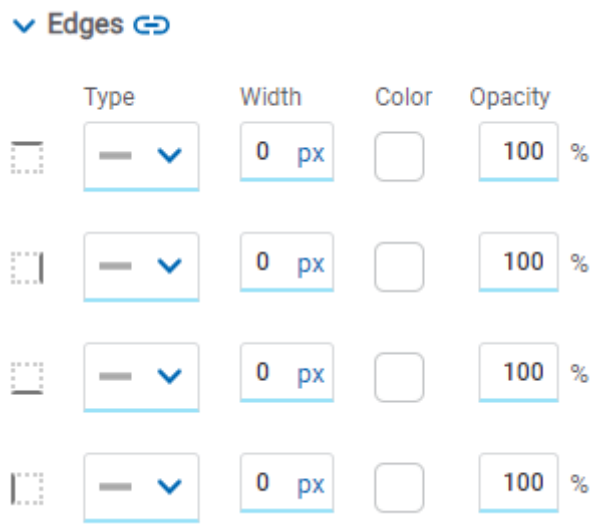
Gradient



This option enables to select not only the color from a palette but also to define its opacity and intensity according to the selected angle.

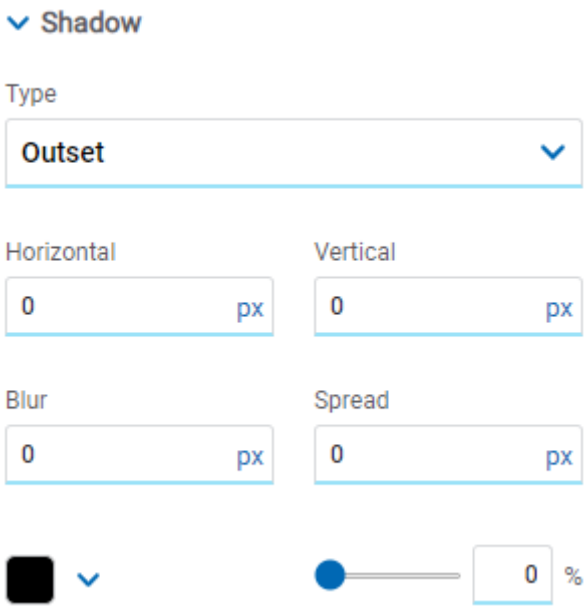


Edges



It allows to define the style of borders. Each one has its type, width, color and opacity defined, the latter as a percentage.

Shadow



Allows to define a shadow effect around the element.

Horizontal
Horizontal size of the shadow to the right of the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Vertical
Vertical size of the shadow below the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

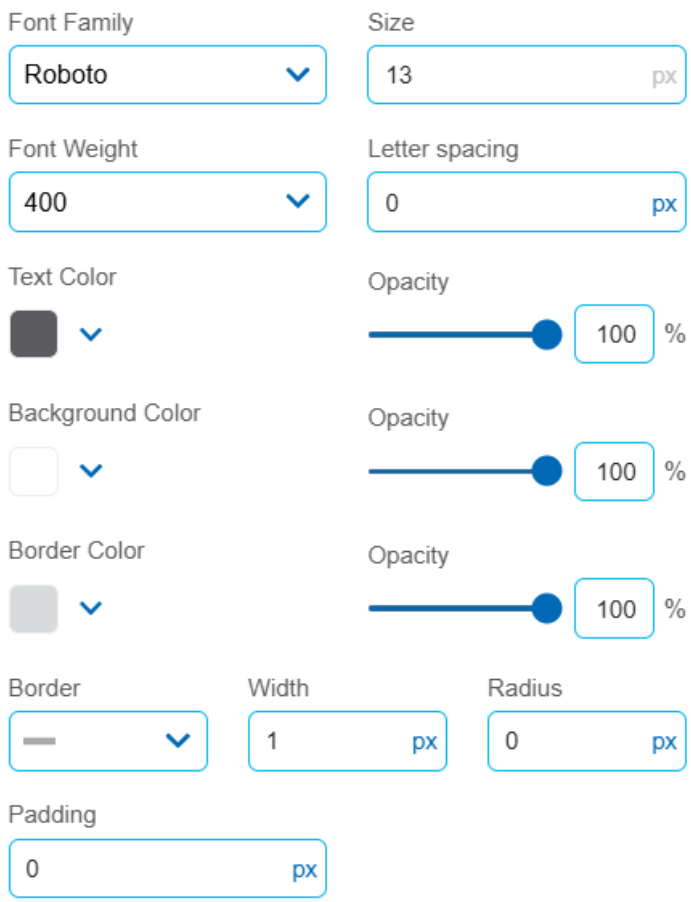
Blur
The greater this value is, the greater and lighter the shadow becomes. If not specified, its value is 0 and the shadow border is darker. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Spread
Positive values will cause the shadow to expand and grow bigger, while negative values will cause the shadow to shrink. If not specified, its value is 0 and the shadow will have the same size as the element.

Color
Defines the color of the shadow.

Transparency
Defines the transparency of the shadow.

Tooltip



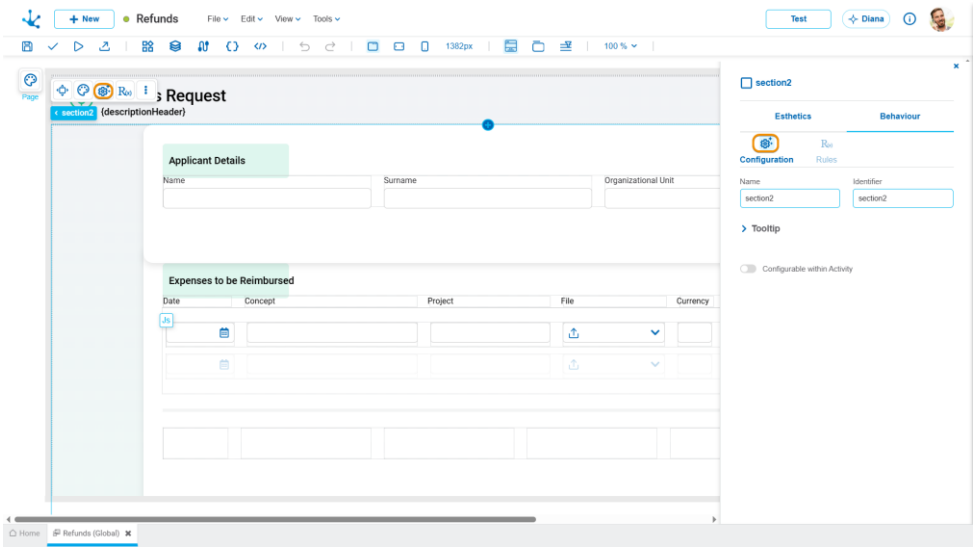
Source
It allows customizing the font, size, thickness and letter spacing of the pop-up message.

Colors
The color and opacity of the text, background and border can be defined. Borders are set uniformly for all sides of the tooltip.

Border
The border type, width, and border radius can be configured.

Configuration Properties

The configuration properties panel of a section opens when selecting the icon  of its context menu.



Name
Name used to reference the section during modeling.

Identifier
Uniquely identifies the section. It is used in the Javascript SDK.

Tooltip
Pop-up message that appears when hovering over an element, providing additional information.

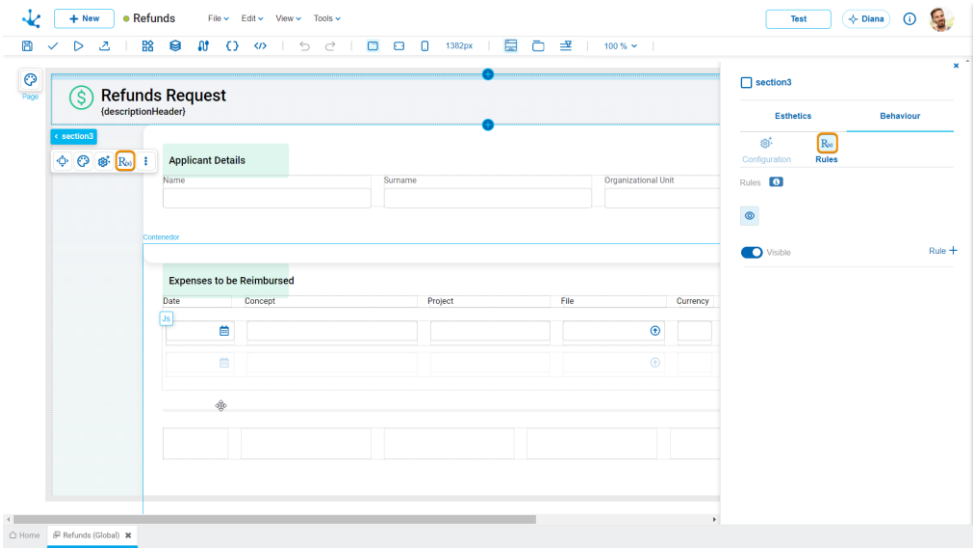


Text
Content of the message being displayed.

Direction
Position of the message in relation to the element (top, bottom, left, or right).

Configurable within Activity
This property is displayed only if the entity is related to a process. When activated, the element is enabled in the execution panel of the associated process, allowing to configure its visibility within their various activities.

The embedded rules properties panel of a section opens when selecting the icon Rx of its context menu.



Properties

Rules

It is possible to define embedded rules for behavior, validation and calculation associated with a section using the wizard (ctrl + space).

Displays syntax examples to write rules.

Visible

Indicates whether the section is visible. If this property is not checked, the section is not displayed on the page.

Visible (default) Not visible

Rule + Opens an editing area where a rule to determine the visibility condition can be defined. If a rule is defined, the icon is displayed with a light blue border.

Operations when defining the rule:

Saves the new or modified rule

Cancels the operation

Operations once the rule is defined:

Edits the existing rule

Deletes the rule

Events

Sections allow the use of different events.

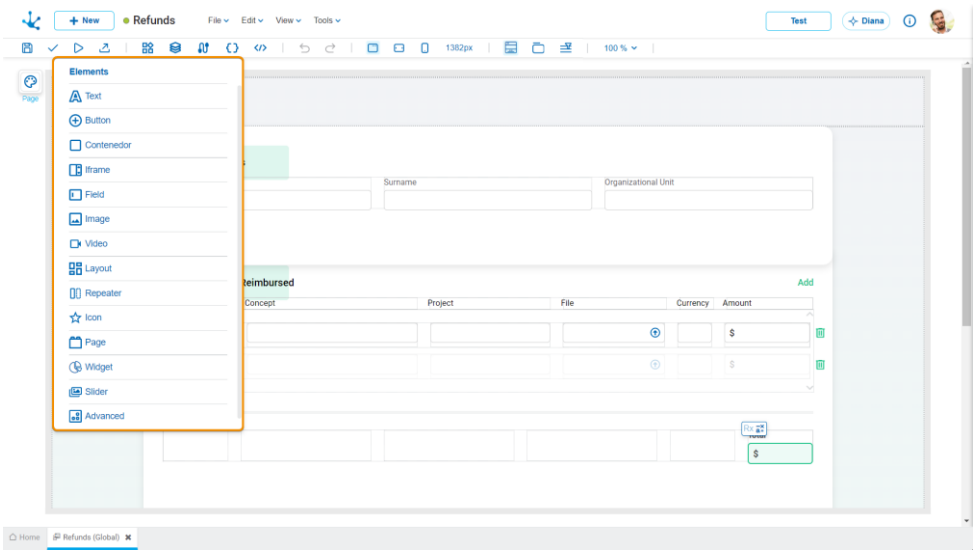
Event	Description
onMouseIn()	It is executed when the cursor is positioned on the element.
onMouseOut()	It is executed when the cursor moves out of the element.
onClick()	It is executed when clicking on the element.
onDoubleClick()	It is executed when clicking twice on the element.
onInit()	It is executed before the element is loaded.
afterViewInit()	It is executed after viewing the element.
onViewportEnter()	It is executed when the element is visible.
onViewportLeave()	It is executed when the element is no longer visible.

Element Types

The elements define the content of the entity and can be of different types.

- Text
- Button
- Container
- Iframe
- Field
- Image
- Video
- Layout
- Repeater
- Icon
- Page
- Widget
- Slider
- Advanced

For each of these types of elements, their properties can be modeled, grouped into panels for structure, style, configuration, hyperlink, and embedded rules.



Text

Texts can be included in an entity, defining the features by modeling its properties.

The element properties are represented by icons on its context menu, where its operations are also available.

Subtypes

When selecting the option "Text" from the icon in the top toolbar, a list opens with the different subtypes of this element, which can be dragged into the modeling area. Each subtype has the element's properties modeled and predefined in a specific way.

- Title 1
- Title 2
- Subtitle 1
- Subtitle 2
- Paragraph
- Normal Text
- Rich Text

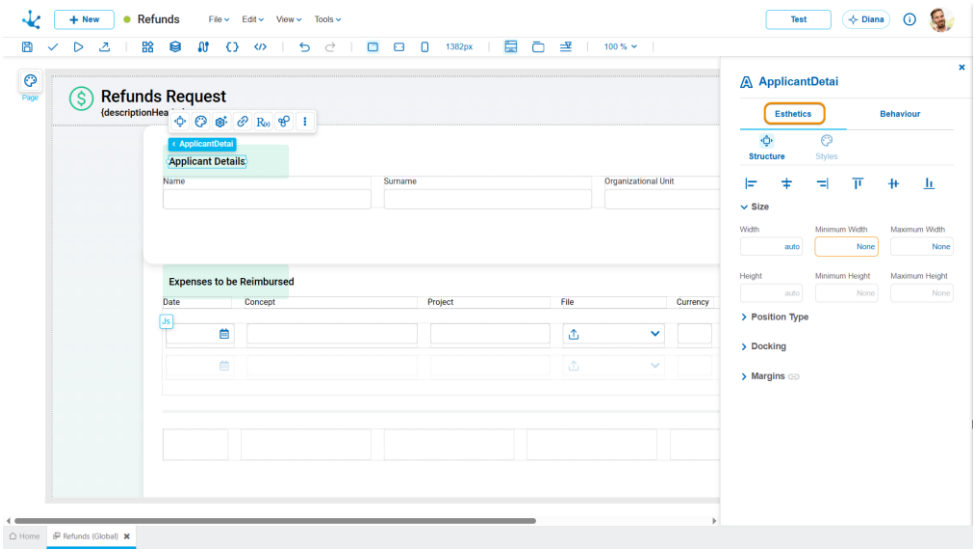
Classification of Properties

Properties are divided into two groups: esthetic properties and behavior properties.

Esthetic Properties

In the esthetic properties panel, the following are grouped:

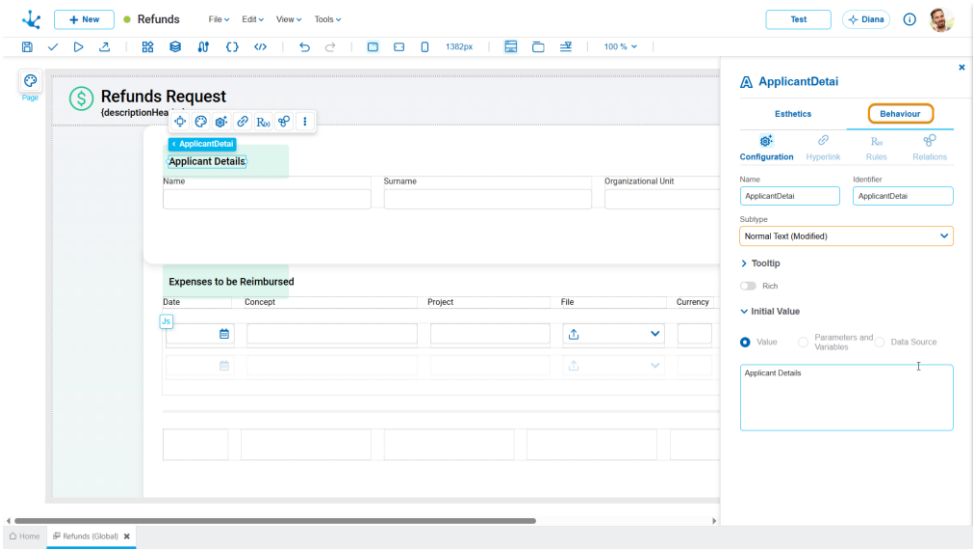
- Structure Properties
- Style Properties



Behavior Properties

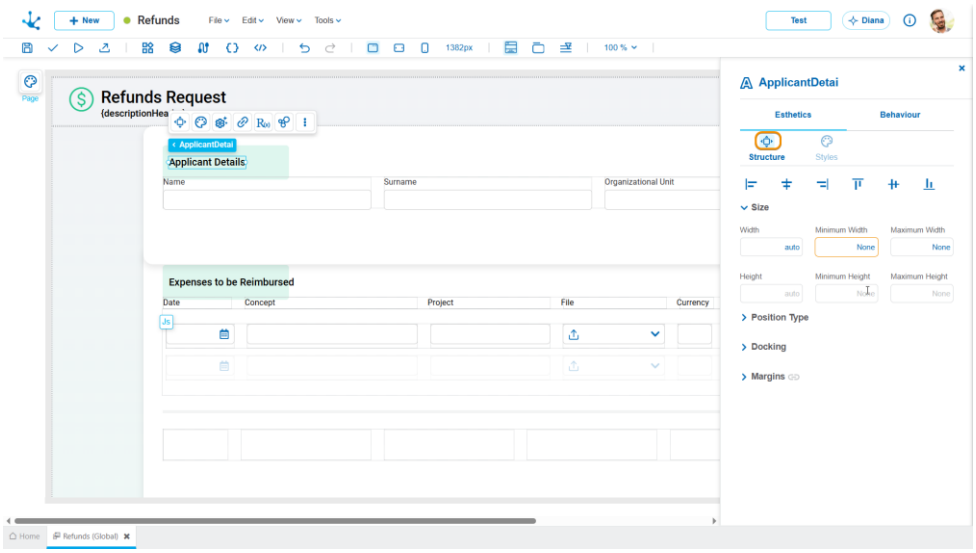
In the behavior properties panel, the following are grouped:

- Configuration Properties
- Hyperlink Properties
- Rule Properties
- Relation Properties



Structure Properties

The structure properties panel of an element opens when selecting the icon of its context menu.

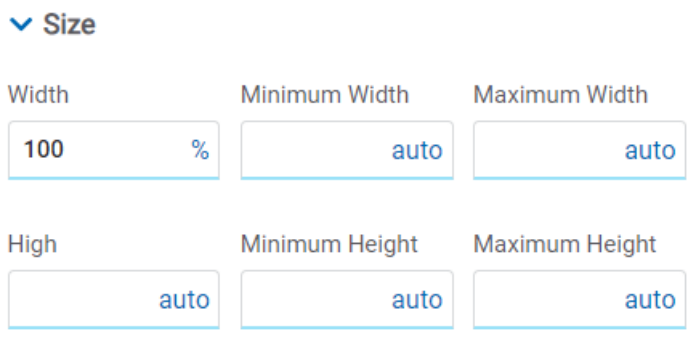


Alignment

Allows to align the element by selecting one of the available icons.

- Align to the left.
- Align to horizontal center.
- Align to the right.
- Align up.
- Align vertical center.
- Align bottom.

Size

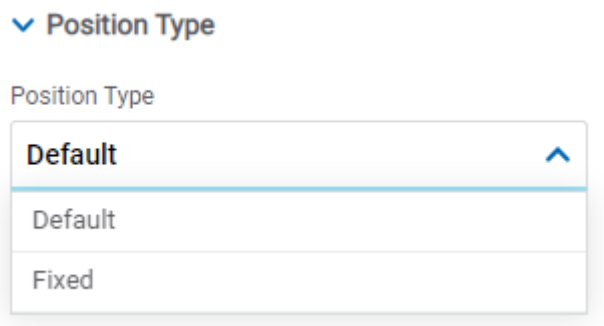


All size properties can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh). Additionally, for Width and Height properties, the “auto” option is added.

> *If selecting percentage (%) for the width and height properties, the size is calculated relative to the top element.*

Position Type

It determines if at the time of execution the element remains fixed on the page or if it moves as the scroll bar moves forward or backward.



Possible Values

- Default: The element has a relative position with respect to the superior element where it was placed (container or section).
- Fixed: Fixes the element to the page so that it is always visible in the same position. It allows to increase the visibility of important content.

Docking



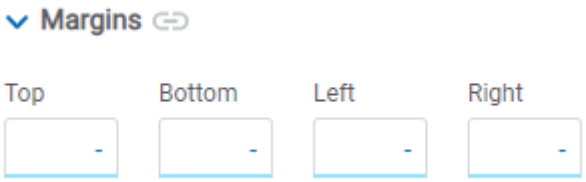
Determines the vertical and horizontal position of elements within the page section or container, when the screen is resized.

When an element is added or moved within the section or container, the modeler automatically docks it to the nearest corners or edges. It can also be configured by clicking on the arrows of the side to be docked to or on the center.

Once the element is docked, its exact position is indicated in the Margins property. The direction of the docking arrows determine the values that are automatically displayed for margins.

The docking position is indicated in the modeling area by dotted lines on the element.

Margins



It allows to define the distance of elements from the borders of their top element. The behavior of margins depends on the docking of the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Top
Distance to the top border of the highest ranking element.

Bottom
Distance to the bottom border of the highest ranking element.

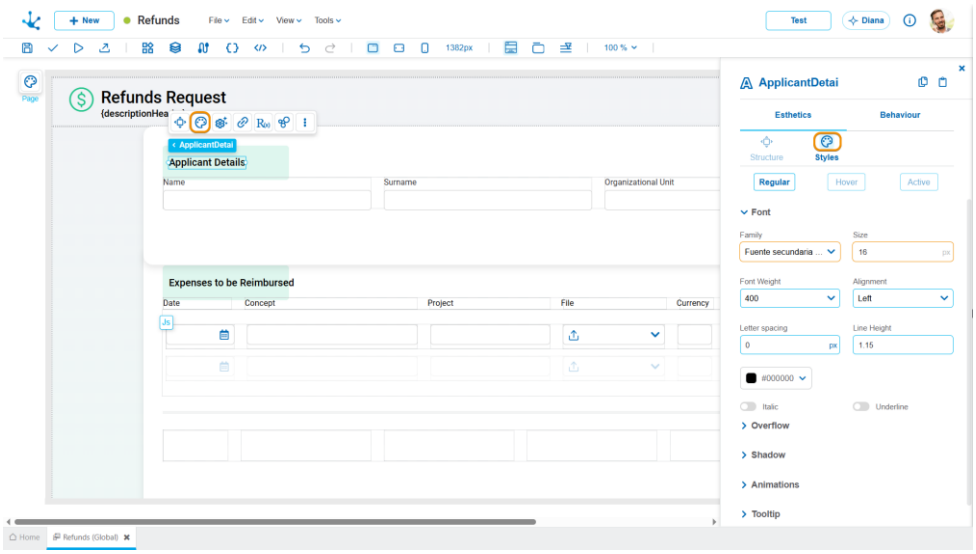
Left
Distance to the left border of the highest ranking element.

Right
Distance to the right border of the highest ranking element.

Allows the value entered in one of the margins to be copied to the other ones automatically.
Allows to indicate different values for each margin.

Style Properties

The style properties panel of an element opens when selecting the icon of its context menu.

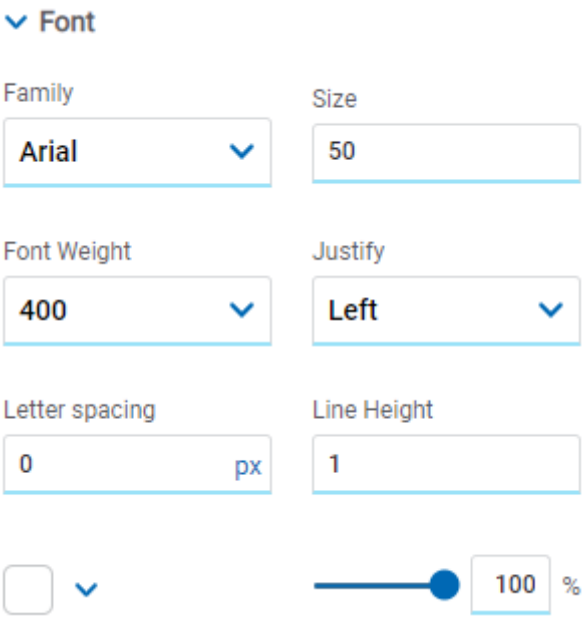


Styles

It allows copying the entire current configuration of the style panel for the selected element (Ctrl+Alt+C).

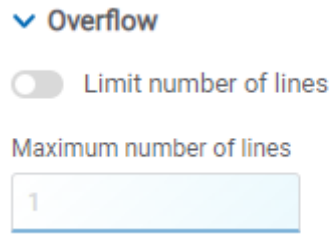
It allows applying the previously copied configuration to another element of the same type (Ctrl+Alt+V).

Source



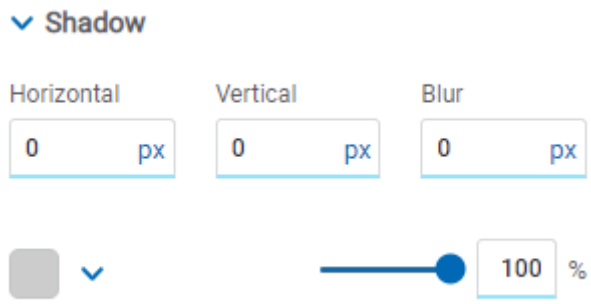
It allows for defining the text style.

Overflow



Allows limiting the number of lines that contain the text, indicating the maximum number of lines. If the text exceeds the number of lines indicated in this property, then it is truncated and displays the icon ⋯.

Shadow



Allows defining a shadow effect around the element.

Horizontal
Horizontal size of the shadow to the right of the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

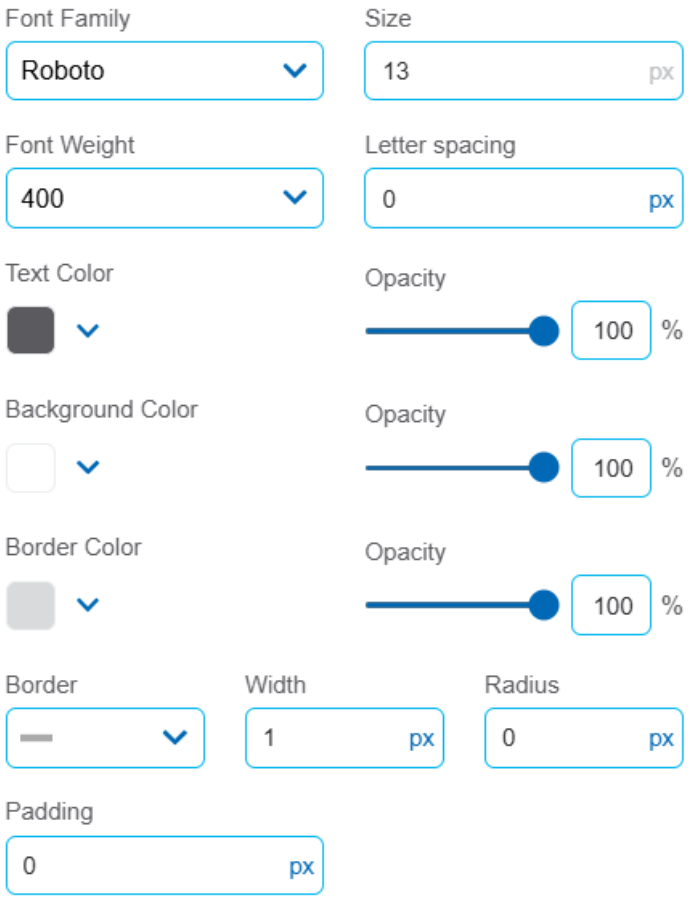
Vertical
Vertical size of the shadow below the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Blur
The greater this value is, the greater and lighter the shadow becomes. If not specified, its value is 0 and the shadow border is darker. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Color
Defines the color of the shadow.

Transparency
Defines the transparency of the shadow.

Tooltip



Source

It allows customizing the font, size, thickness and letter spacing of the pop-up message.

Colors

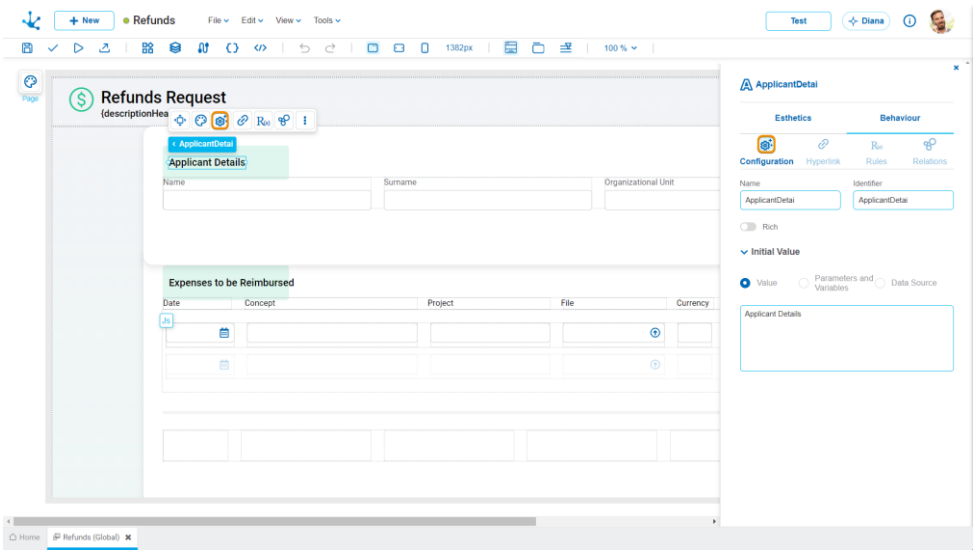
The color and opacity of the text, background and border can be defined. Borders are set uniformly for all sides of the tooltip.

Border

The border type, width, and border radius can be configured.

Configuration Properties

The configuration properties panel of an element opens when clicking the icon  of its context menu.



Name
Name used to reference the element during modeling.

Identifier
Uniquely identifies the element. It is used in the Javascript SDK.

Tooltip
Pop-up message that appears when hovering over an element, providing additional information.

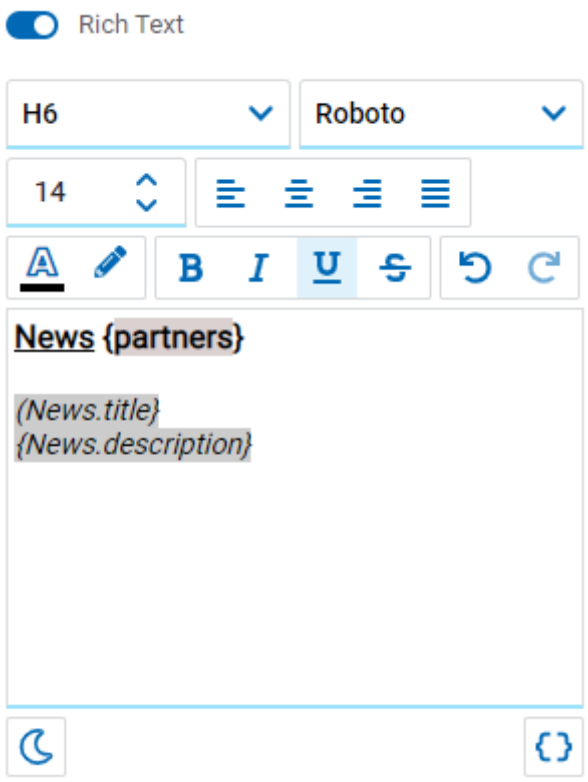


Text
Content of the message being displayed.

Direction
Position of the message in relation to the element (top, bottom, left, or right).

Rich Text

It allows defining formats that modify the appearance of the text element's content. Different styles can be combined in the same text, some parts can be highlighted, and more than one data source, parameter, or variable can be added.



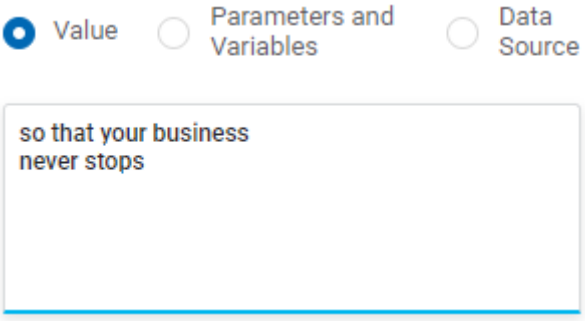
Changes the style of the code editor to dark mode.

Changes the style of the code editor to the previous mode.

It allows the incorporation of the content of variables, parameters, and information from different data sources into the text.

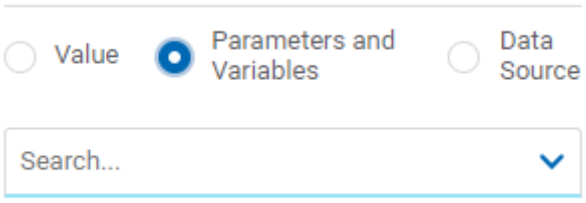
Initial Value

It allows selecting the source of the element's content and optionally defining the text style.



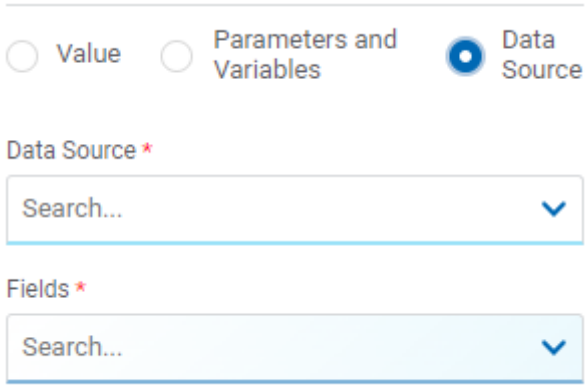
Value
Allows to enter a text that is displayed in the element.

Parameters and Variables



Parameters and Variables
Allows to select a parameter or a variable from, whose value is displayed in the element.

Data Source

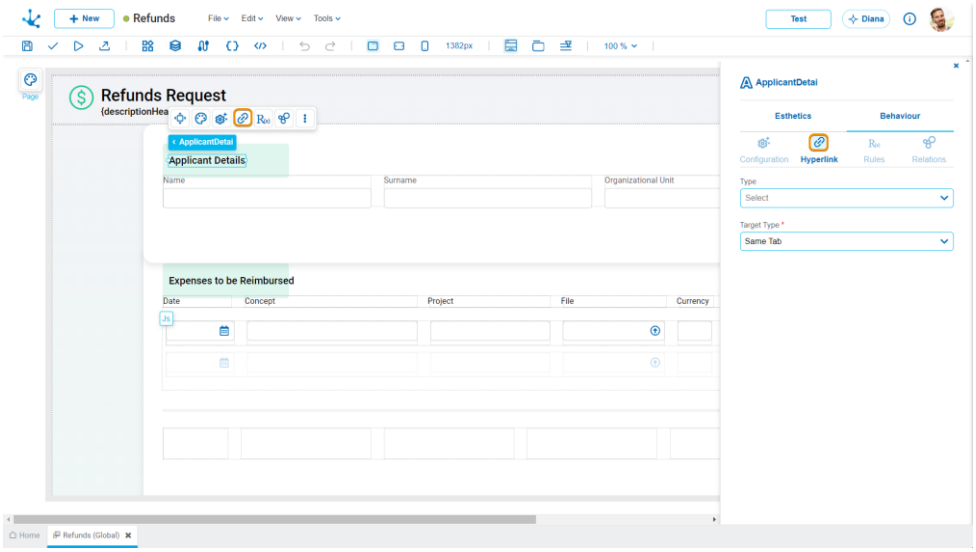


Data Source
It allows selecting a previously defined data source within the object.

Fields
It allows selecting a field from the data retrieved in the chosen data source, whose value is displayed in the element.

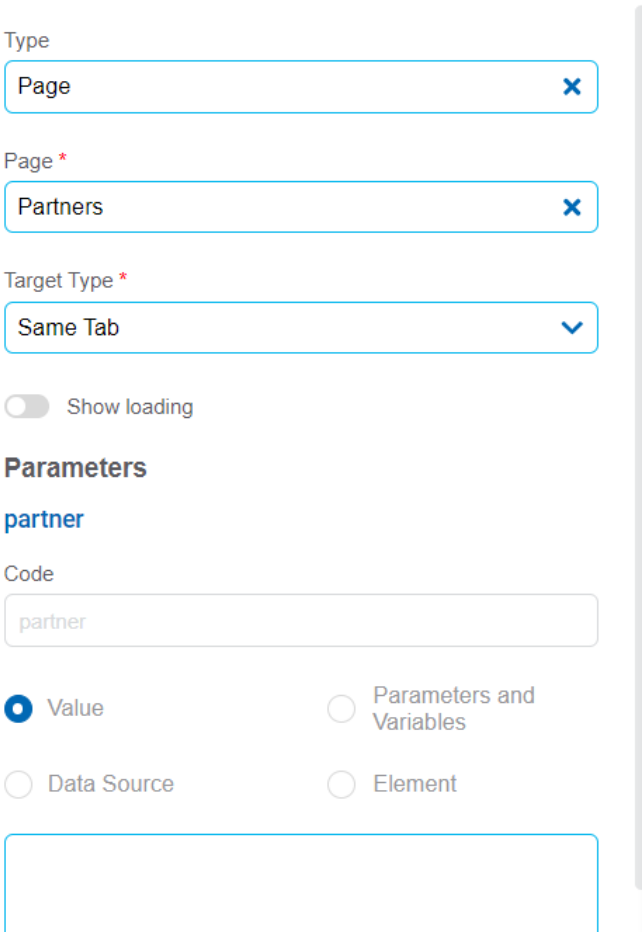
Hyperlink Properties

The hyperlink properties panel of an element opens when clicking the icon of its context menu.



Define the behavior of the element when selecting it. Different properties are enabled depending on the type of object selected.

Page



Page
The pages modeled in the environment are displayed.

Destination Type
The available options for opening the object are displayed.

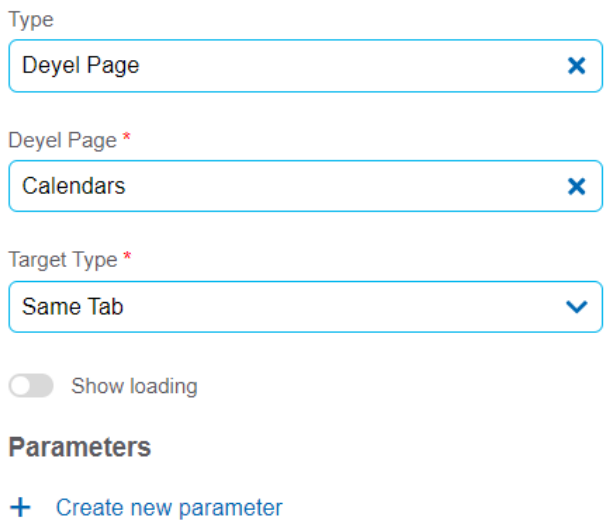
Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  - Modal Horizontal Size
    Defines its width.
  - Modal Vertical Size
    Defines its height.
- Iframe: selecting this option enables an additional property.
  - Destination Iframe
    Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Parameters
If there are parameters, their value can be specified with text, parameters and variables or data source.

Deyel Page



Deyel Page
The pages belonging to **Deyel** are displayed.

Destination Type
The available options for opening the object are displayed.

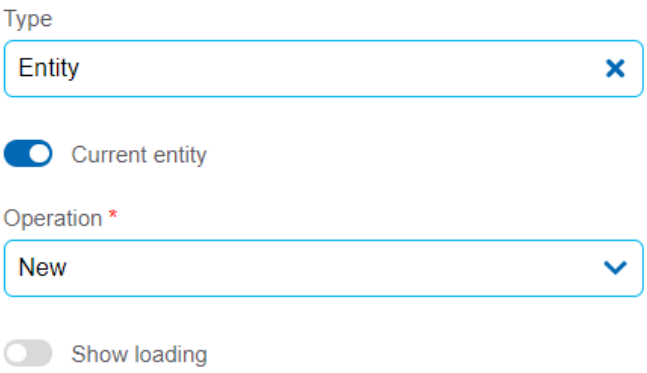
Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  - Modal Horizontal Size
    Defines its width.
  - Modal Vertical Size
    Defines its height.
- Iframe: selecting this option enables an additional property.
  - Destination Iframe
    Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Parameters
Allows sending parameters to the selected object type.

Entity



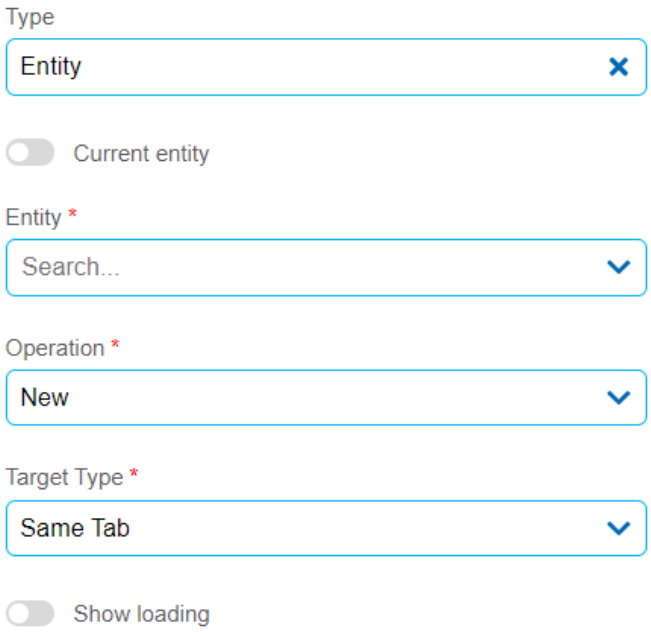
Current Entity
If this property is checked, the operation applies directly to the entity being modeled.

Operation
Defines the operation made when selecting the object.

Possible Values

- Create: Indicates that the panel of the current entity is opened to create an instance.
- Update: Indicates that the panel of the current entity is opened to update an instance.
- Delete: Indicates that the panel of the current entity is opened to delete an instance.

- Send Email: It is visible if the property Send Emails is enabled. Clicking on the modeled element opens a panel during instance execution to send an email. The panel size is adjusted with the properties Modal Width and Modal Height.
- Emails: Visible if the Send Emails property is enabled. Clicking on the modeled element opens a panel during instance execution to manage sent emails. The panel is displayed in the iframe defined in the Target Iframe property.



Entity
If the property Current Entity, is not checked, the entities modeled in the environment are displayed.

Operation
Defines the operation made when selecting the object.

Possible Values

- New: The panel of the entity selected in the previous property is opened to create an instance.
- Update The panel of the entity selected in the previous property is opened to update an instance.
- Show: The panel of the entity selected in the previous property is opened to show an instance.
- Grid: Indicates that the grid of the entity selected in the previous property is opened.

Destination Type
The available options for opening the object are displayed.

Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  Modal Horizontal Size
  Defines its width.

Modal Vertical Size
Defines its height.

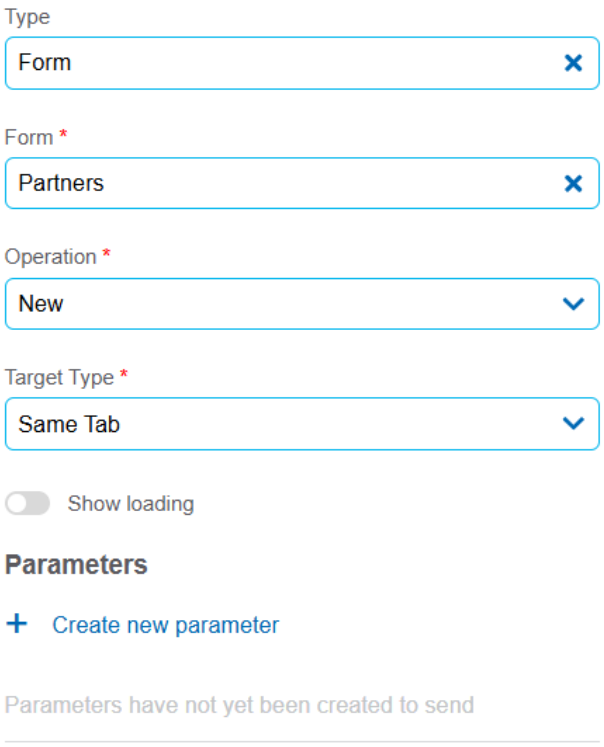
- Iframe: selecting this option enables an additional property.

Destination Iframe
Expands the iframes previously defined on the object.

- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Forms



Forms
The forms modeled in the environment are displayed.

Operation
Defines the operation made when selecting the object.

Possible Values

- Create: Indicates that the panel of the object selected in the previous property is opened for the creation of an instance.
- Update: Indicates that the panel of the object selected in the previous property is opened for the update of an instance.
- Show: Indicates that the panel of the object selected in the previous property is opened for the query of an instance.

- Grid: Indicates that the grid of the object selected in the previous property is opened.

Destination Type
The available options for opening the object are displayed.

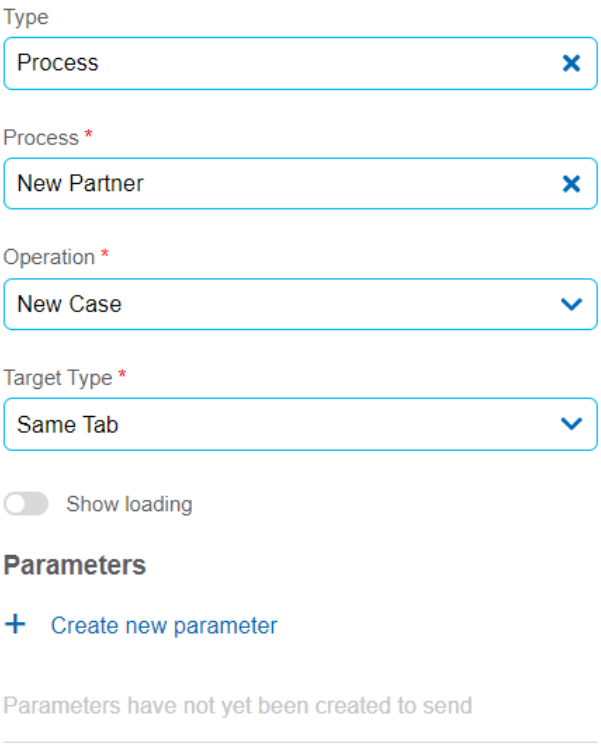
Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  Modal Horizontal Size
  Defines its width.
  Modal Vertical Size
  Defines its height.
- Iframe: selecting this option enables an additional property.
  Destination Iframe
  Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Parameters
Allows sending parameters to the selected object type.

Process



Process
The processes modeled in the environment are displayed.

Operation
Defines the operation made when selecting the object.

Destination Type
The available options for opening the object are displayed.

Possible Values

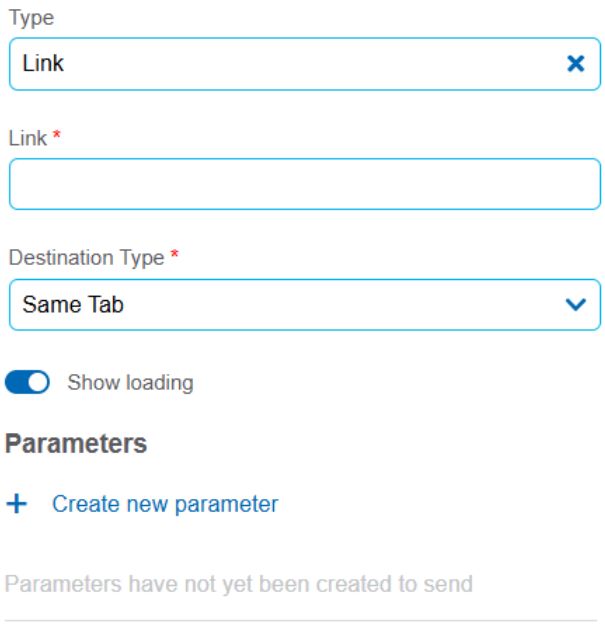
- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  - Modal Horizontal Size
    Defines its width.
  - Modal Vertical Size
    Defines its height.
- Iframe: selecting this option enables an additional property.
  - Destination Iframe
    Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Parameters

Allows sending parameters to the selected object type.

Link



Link
Allows to enter any link.

Destination Type
The available options for opening the object are displayed.

Possible Values

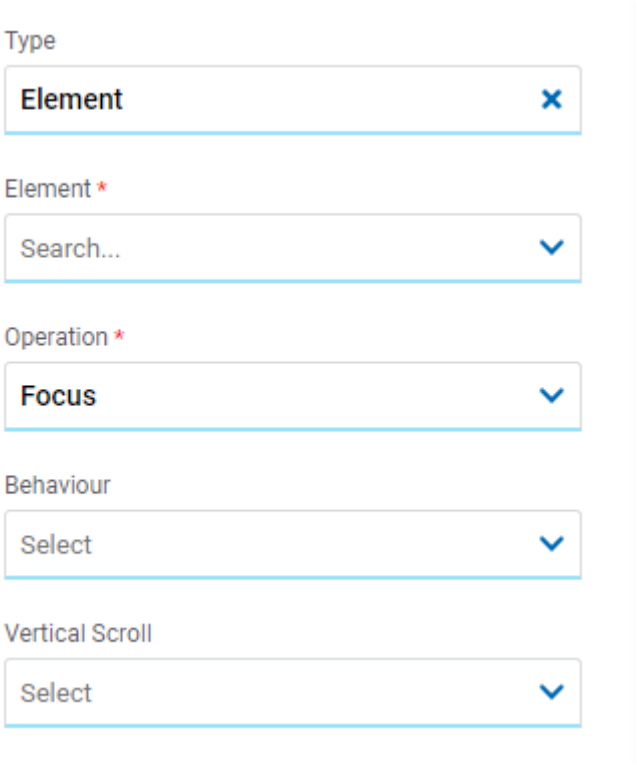
- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  Modal Horizontal Size
  Defines its width.
  Modal Vertical Size
  Defines its height.
- Iframe: selecting this option enables an additional property.
  Destination Iframe
  Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Parameters

Allows sending parameters to the selected object type.

Element



Element
The modeled elements in the object are displayed.

Operation
Defines the operation made when selecting the element.

Possible Values

- Focus
- Show
- Hide
- Show/Hide

Behaviour
Establishes the transition of the animation. It is only modelable if the "Focus" operation was selected.

Possible Values

- Auto
- Smooth

Vertical Scroll
It is only modelable if the "Focus" operation was selected.
Possible Values

- Start: Moves to the start of the selected element.
- Closest: Moves to the closest position to the selected element from the element the event fires.
- Center: Moves to the center of the selected element.
- End: Moves to the end of the selected element.

Repeater



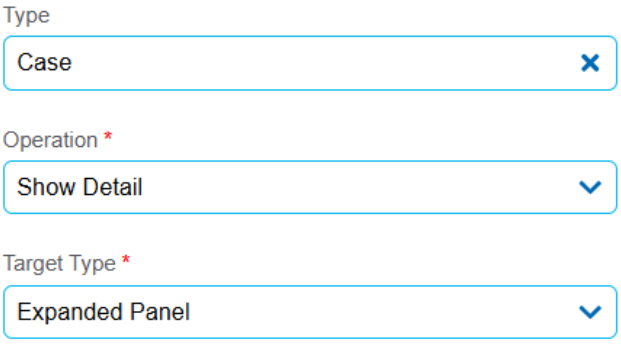
Repeater
The modeled repeaters from the object are displayed.

Operation
Defines the operation made when selecting the element.

Possible Values

- New: Allows for the creation of an item within the repeater. If this option is selected the additional properties are enabled.
  Creation Order
  It indicates in which position the new item of the container is added.
  Possible Values
  - Home
  - End
- Delete: Allows for the deletion of an item within the repeater.

Case



Case
Show case information.
Operation
Defines the operation made when selecting the element.

Possible Values

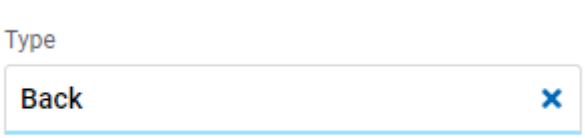
- Show Detail
- Show Graphic Execution

Destination Type
The available options for opening the object are displayed.

Possible Values

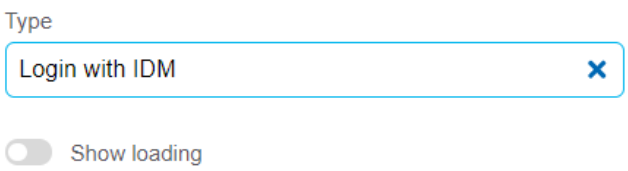
- Expanded Panel
- Modal: if this option is selected the additional properties are enabled.
  Modal Horizontal Size
  Define its width.
  Modal Vertical Size
  Define its height.

Back



It allows associating the event to go back in the browser to the element.

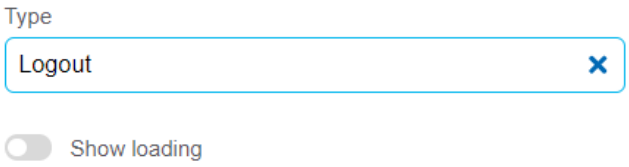
LogIn with IDM



Allows login with IDM.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

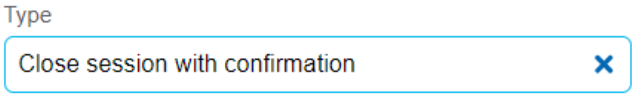
Logout



Allows logging the user out.

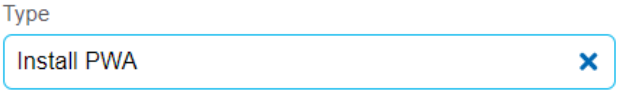
Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Close Session with Confirmation



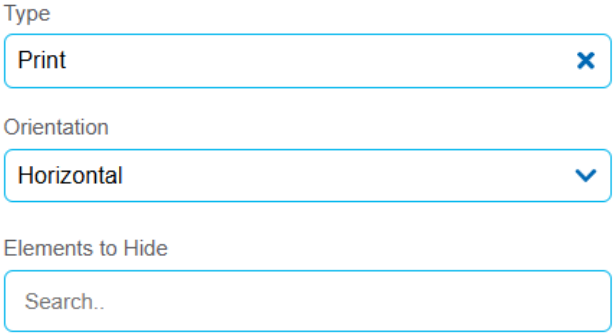
Displays a confirmation modal to logout the user.

Install PWA



Allows to install the application in the browser.

Print



This type of hyperlink associated with an element allows printing the content of the entity or page that is being executed.

Orientation

Determines the layout of the content to be printed.

Possible Values

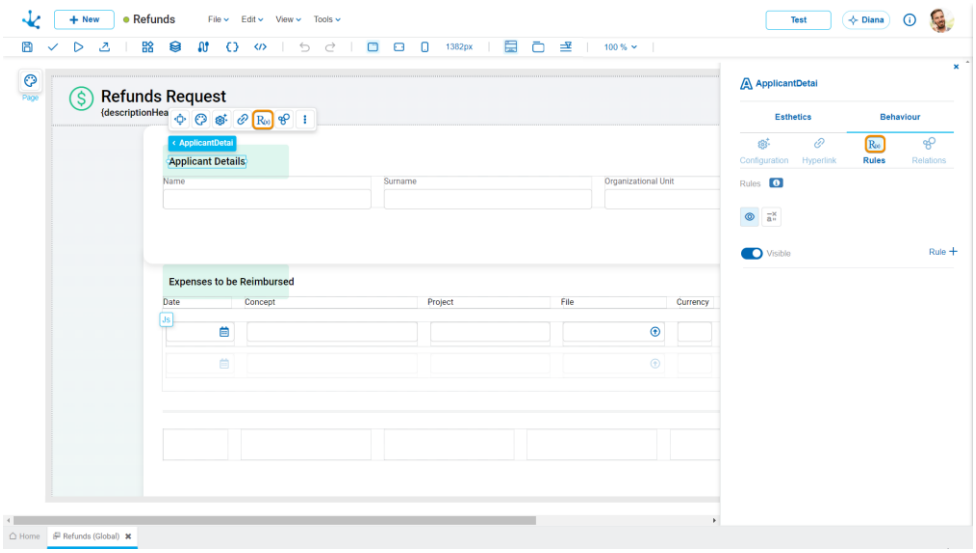
- Horizontal
- Vertical

Elements to hide

It allows selecting which elements of the entity or page should not be included in the printout.

Rule Properties

The rule properties panel of an element opens when clicking the icon  of its context menu.



Properties

Rules

Embedded rules on behavior, validation and calculation can be defined, associated with an element, by using the wizard (ctrl + space).

Shows syntax examples for writing the rules.

Visible

Indicates whether the element is visible. If this property is not checked, the element is not displayed in the page.

Visible (default) Not visible

Rule + Opens an edit area where a rule to determine the visibility condition can be defined. If a rule is defined, the icon is displayed with light blue borders.

Calculation

Rule + Opens an edit area where you can define the expression to be executed to calculate the element value. If a rule is defined, the icon is displayed with light blue borders.

Operations when defining the rule:

- Saves the new or modified rule
- Cancels the operation

Operations once the rule is defined:

- Edits the existing rule
- Deletes the rule

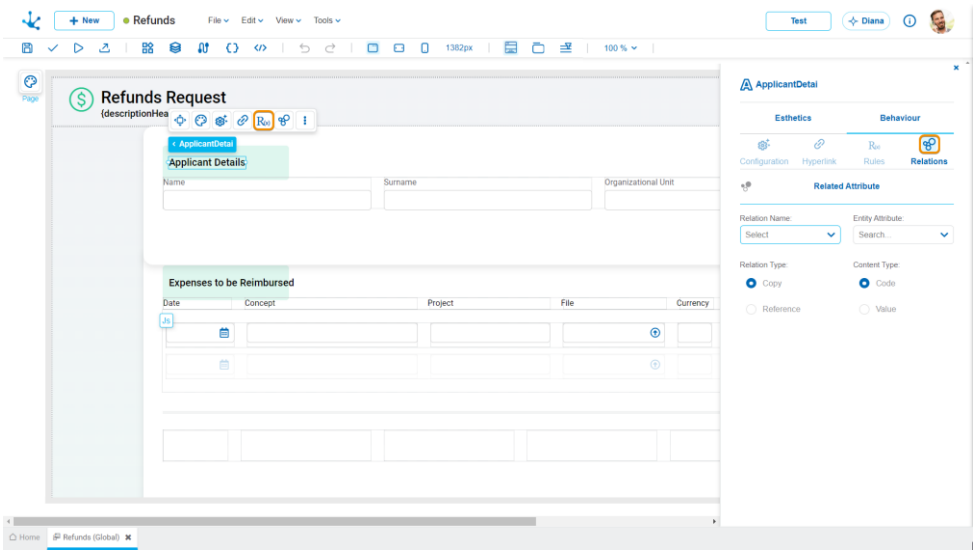
Events

Texts allow the use of different events.

Event	Description
onMouseIn()	It is executed when the cursor is positioned on the element.
onMouseOut()	It is executed when the cursor moves out of the element.
onClick()	It is executed when clicking on the element.
onDoubleClick()	It is executed when clicking twice on the element.
onInit()	It is executed before the element is loaded.
afterViewInit()	It is executed after viewing the element.
onViewportEnter()	It is executed when the element is visible.
onViewportLeave()	It is executed when the element is no longer visible.

Relation Properties

The relation properties panel of an element opens when clicking the icon of its context menu.



Related Attribute

A related attribute indicates that the field value is retrieved from the attribute value of another entity. One of the relations defined on the entity must be selected, and then it should be specified which attribute of the related entity it links to.

A field can be modeled as a related attribute and it can also have a relation defined to an entity, a value list, or a rule. This allows the value of the related attribute to be retrieved on this field and the functionality provided by the relation to be used.

Properties

Relation Name
The list of relations defined in the entity is displayed, allowing the selection of one of them.

Entity Attribute
The list of attributes of the selected entity compatible with the field type is displayed. Attributes can have a relation modeled with an entity, a value list, or a rule in the selected entity.

Relation Type
Copy
When this property is checked, once the retrieved value is uploaded, it does not automatically update even if the value of the related attribute changes.
Reference
When this property is checked, the field value is subject to the current value of the related attribute, that is, the updated value is automatically displayed.

Content Type
The element content varies depending on whether the Code or Value property is checked when:
- The entity attribute has a modeled relation.
- The field has the option "Entity" selected in the Values Obtained from property.

Code
When this property is checked, if the relation is modeled with a:
- Related entity, the identifier of the entity is displayed.

- Value list, the value list code is displayed.
- Rule, the code returned by the rule's execution is displayed.

Value

When this property is checked, if the relation is modeled with a:

- Related entity, the short description of the entity is displayed.
- Value list, the value list description is displayed.
- Rule, the description returned by the rule's execution is displayed.

Button

A button can be included in an entity, defining the features by modeling its properties.

The element properties are represented by icons on its context menu, where its operations are also available.

Subtypes

When selecting the option "Button" from the icon in the top toolbar, a list opens with the different subtypes of this element, which can be dragged into the modeling area. Each subtype has the element's properties modeled and predefined in a specific way.

- Primary
- Secondary
- Tertiary

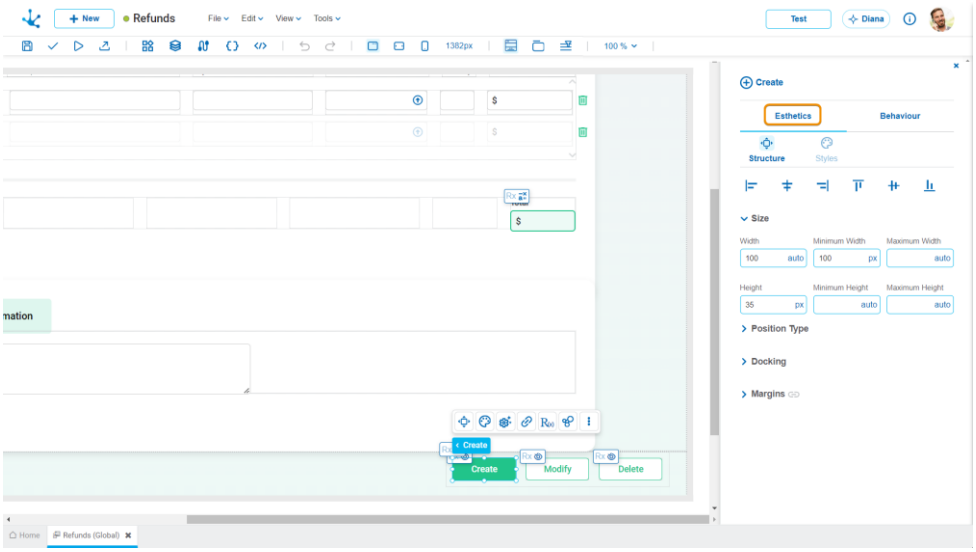
Classification of Properties

Properties are divided into two groups: esthetic properties and behavior properties.

Esthetic Properties

In the esthetic properties panel, the following are grouped:

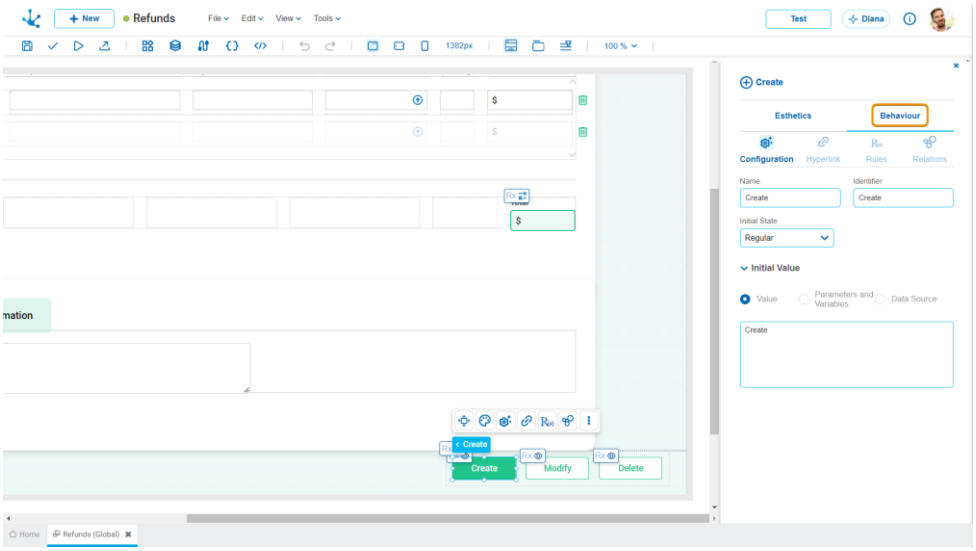
- Structure Properties
- Style Properties



Behavior Properties

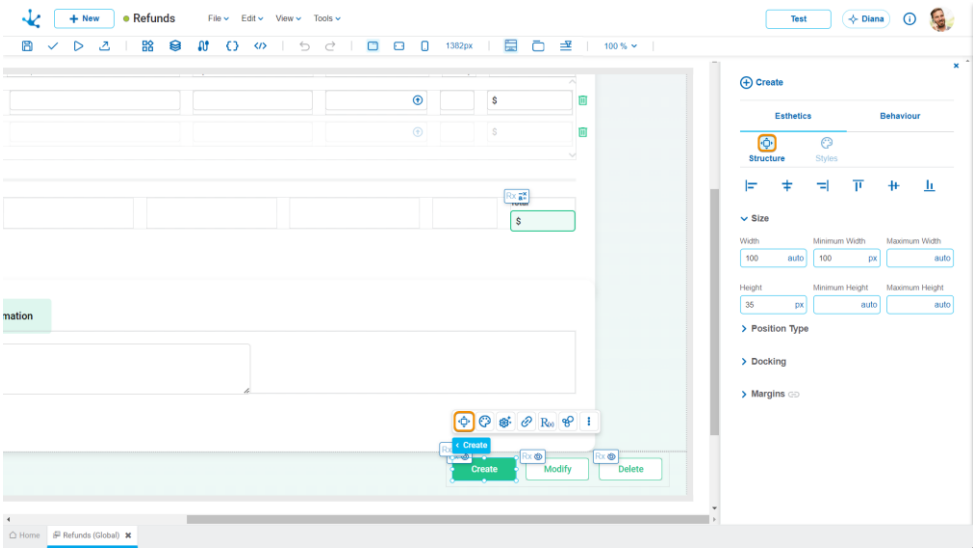
In the behavior properties panel, the following are grouped:

- Configuration Properties
- Hyperlink Properties
- Rule Properties
- Relation Properties



Structure Properties

The structure properties panel of an element opens when selecting the icon  of its context menu.

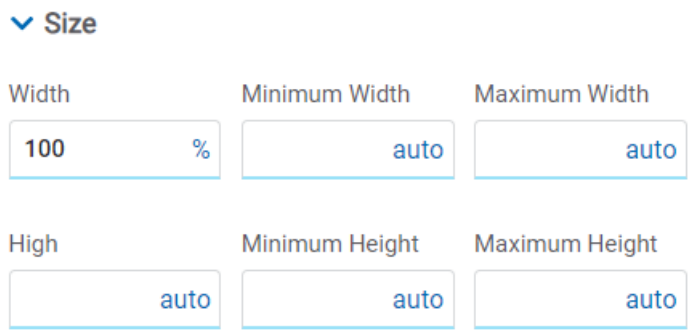


Alignment

Allows to align the element by selecting one of the available icons.

- Align to the left.
- Align to horizontal center.
- Align to the right.
- Align up.
- Align vertical center.
- Align bottom.

Size

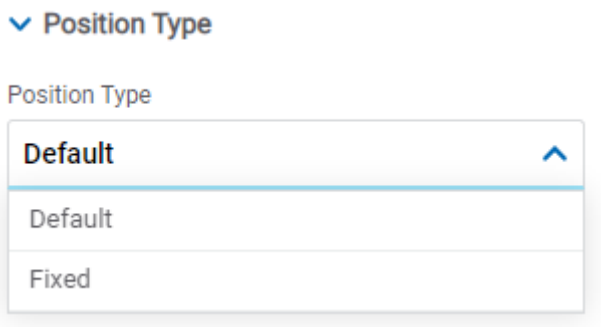


All size properties can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh). Additionally, for Width and Height properties, the “auto” option is added.

> *If selecting percentage (%) for the width and height properties, the size is calculated relative to the top element.*

Position Type

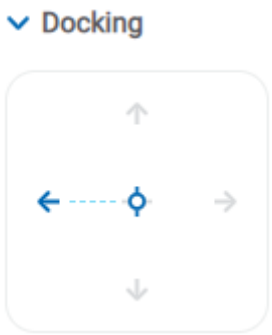
It determines if at the time of execution the element remains fixed on the page or if it moves as the scroll bar moves forward or backward.



Possible Values

- Default: The element has a relative position with respect to the superior element where it was placed (container or section).
- Fixed: Fixes the element to the page so that it is always visible in the same position. It allows to increase the visibility of important content.

Docking



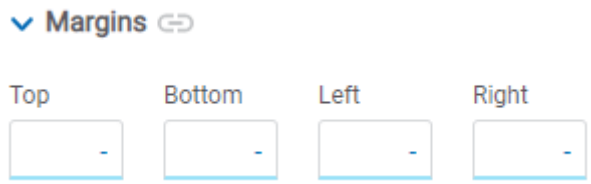
Determines the vertical and horizontal position of elements within the page section or container, when the screen is resized.

When an element is added or moved within the section or container, the modeler automatically docks it to the nearest corners or edges. It can also be configured by clicking on the arrows of the side to be docked to or on the center.

Once the element is docked, its exact position is indicated in the Margins property. The direction of the docking arrows determine the values that are automatically displayed for margins.

The docking position is indicated in the modeling area by dotted lines on the element.

Margins



It allows to define the distance of elements from the borders of their top element. The behavior of margins depends on the docking of the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Top
Distance to the top border of the highest ranking element.

Bottom
Distance to the bottom border of the highest ranking element.

Left
Distance to the left border of the highest ranking element.

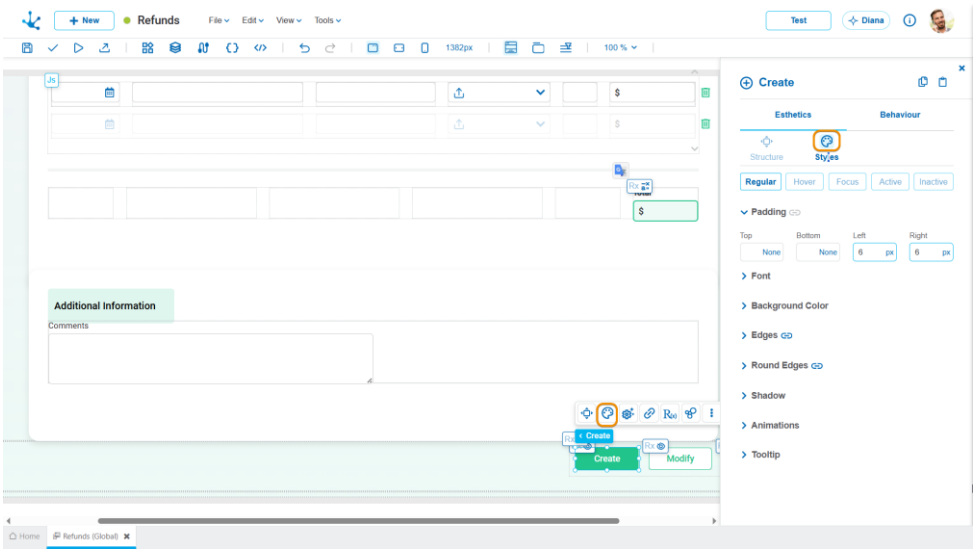
Right
Distance to the right border of the highest ranking element.

Allows the value entered in one of the margins to be copied to the other ones automatically.

Allows to indicate different values for each margin.

Style Properties

The style properties panel of an element opens when selecting the icon of its context menu.



This type of element may take different states and for each of them different values for its properties may be modeled.



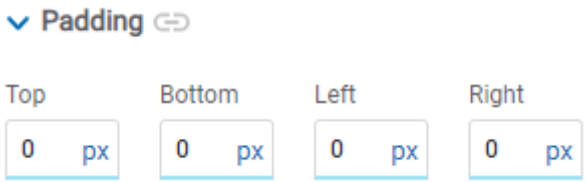
- Regular: the default state of the element, when it is enabled and the mouse pointer is not over it.
- Hover: the mouse pointer is over the element, indicating that it is interactive.
- Focus: the element has been selected using the keyboard (for example, with the Tab key) and is ready to receive an action.
- Active: the element is being pressed, either with the mouse or through a keyboard action.
- Inactive: the element is disabled and cannot receive any user interaction.

Styles

It allows copying the entire current configuration of the style panel for the selected element (Ctrl+Alt+C).

It allows applying the previously copied configuration to another element of the same type (Ctrl+Alt+V).

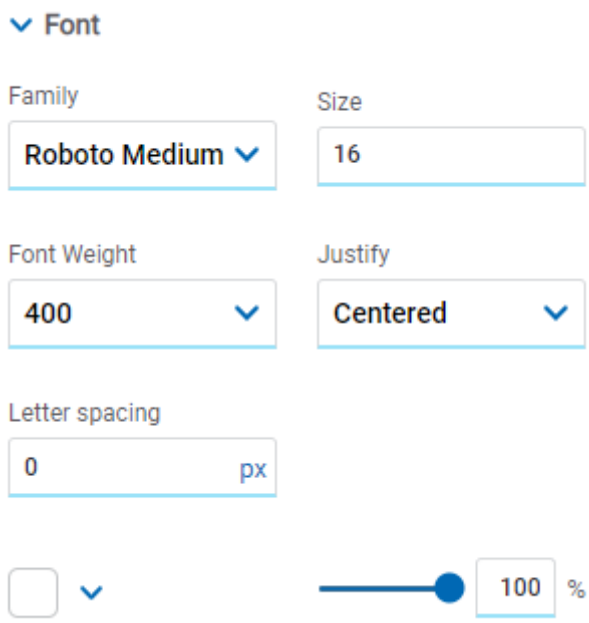
Padding



All padding space properties allow to create a space around the element borders (top, bottom, sides) and its content. The padding space configured is inward, while the margin is configured relative to a top element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Allows values entered in one of the paddings to be copied to the other ones automatically.
Allows to indicate different values for each padding.

Font

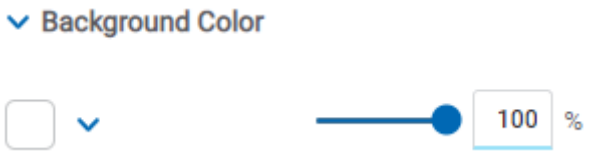


It allows to define the text style.

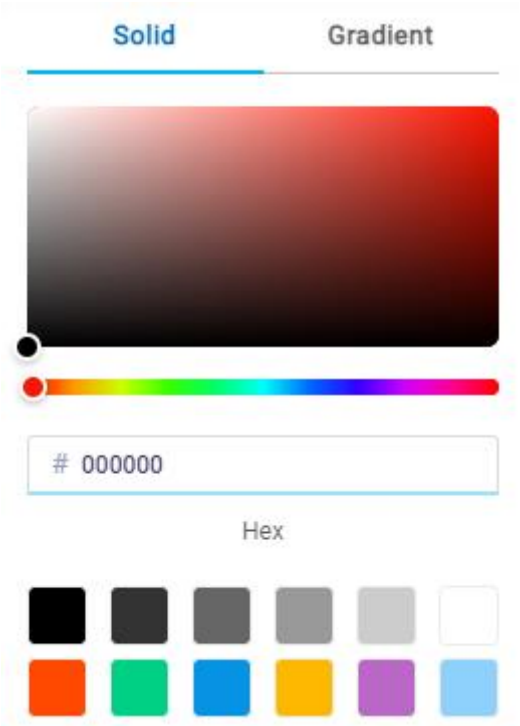
Background Color

It allows to model a background color for the element, it can be solid or gradient and different properties are defined for each one.

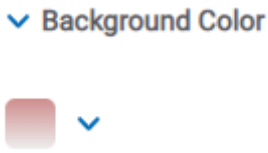
Solid



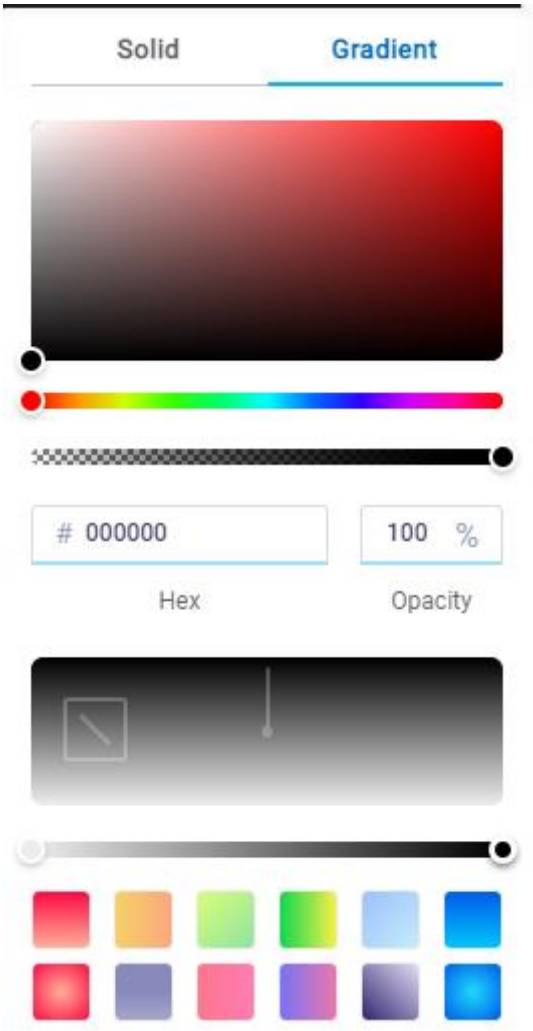
This option enables not only to select the color from a palette but also to define the degree of transparency.



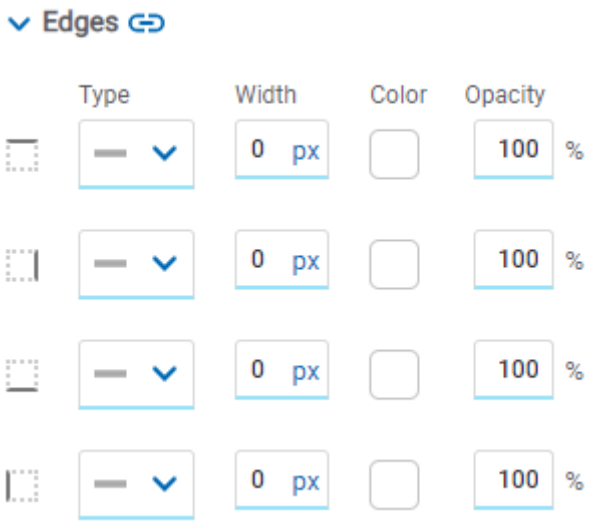
Gradient



This option enables to select not only the color from a palette but also to define its opacity and intensity according to the selected angle.



Edges



It allows to define the style of borders. Each one has its type, width, color and opacity defined, the latter as a percentage.

Allows the value entered in one of the borders to be copied to the other ones automatically.
Allows to indicate different values for each border.

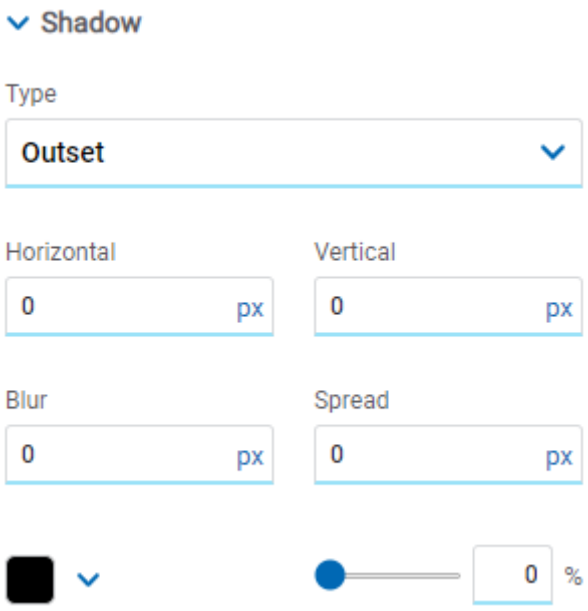
Round Edges



It allows to define the round edges at the corners of elements.
It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Allows the value entered in one of the borders to be copied to the other ones automatically.
Allows to indicate different values for each border.

Shadow



Allows to define a shadow effect around the element.

Type

Possible Values

- Outset
- Inset

Horizontal
Horizontal size of the shadow to the right of the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Vertical
Vertical size of the shadow below the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

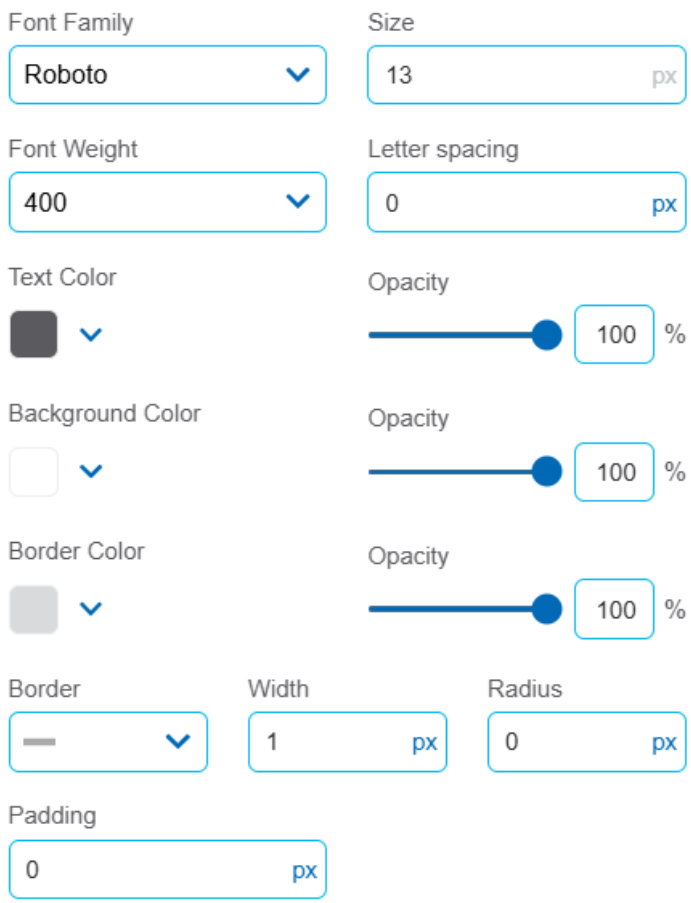
Blur
The greater this value is, the greater and lighter the shadow becomes. If not specified, its value is 0 and the shadow border is darker. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Spread
Positive values will cause the shadow to expand and grow bigger, while negative values will cause the shadow to shrink. If not specified, its value is 0 and the shadow will have the same size as the element.

Color
Defines the color of the shadow.

Transparency
Defines the transparency of the shadow.

Tooltip



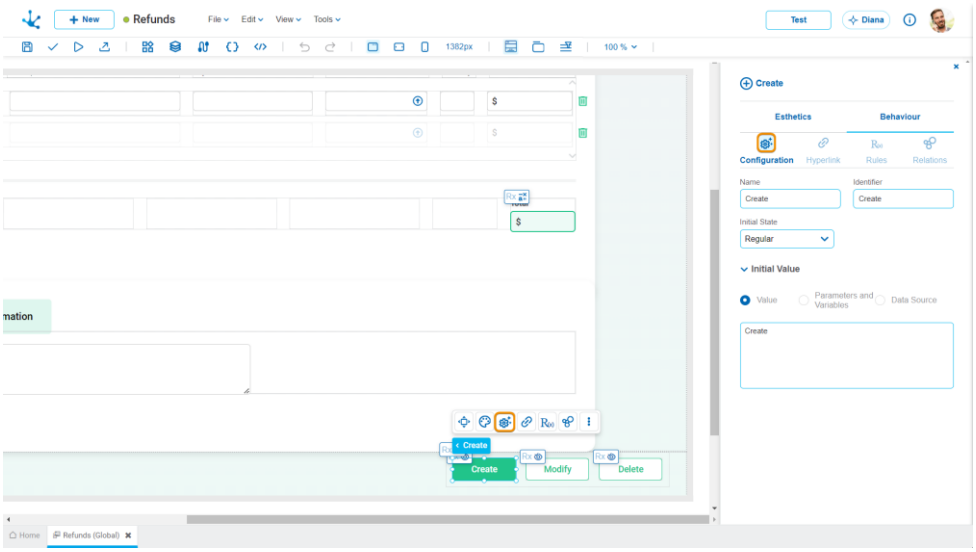
Source
It allows customizing the font, size, thickness and letter spacing of the pop-up message.

Colors
The color and opacity of the text, background and border can be defined. Borders are set uniformly for all sides of the tooltip.

Border
The border type, width, and border radius can be configured.

Configuration Properties

The configuration properties panel of an element opens when clicking the icon  of its context menu.

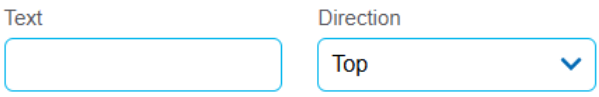




Name
Name used to reference the element during modeling.

Identifier
Uniquely identifies the element. It is used in the Javascript SDK.

Tooltip
Pop-up message that appears when hovering over an element, providing additional information.



Text
Content of the message being displayed.

Direction
Position of the message in relation to the element (top, bottom, left, or right).

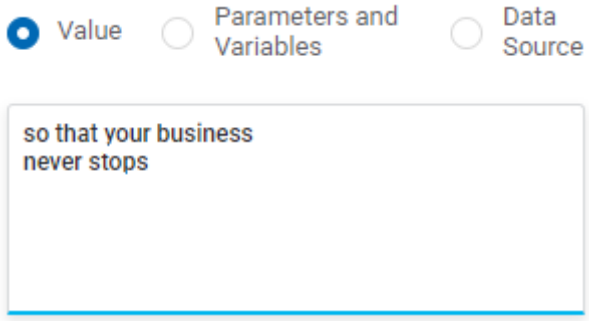
Initial State
It allows defining the state of the button when the entity is loaded for the first time, ensuring that its initial behavior is as expected.

Possible Values

- Regular: The button is enabled, allowing immediate user interaction. It is used for actions that are available from the start.
- Inactive The button is disabled until certain conditions are met or a specific action is performed. Used for flows that require prior validations before enabling the button action.

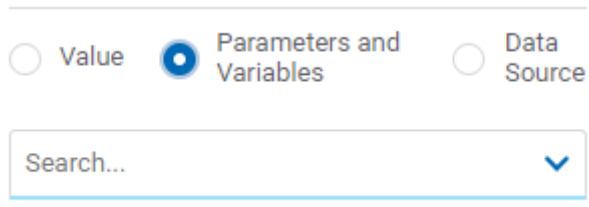
Initial Value

It allows selecting the source of the element's content and optionally defining the text style.



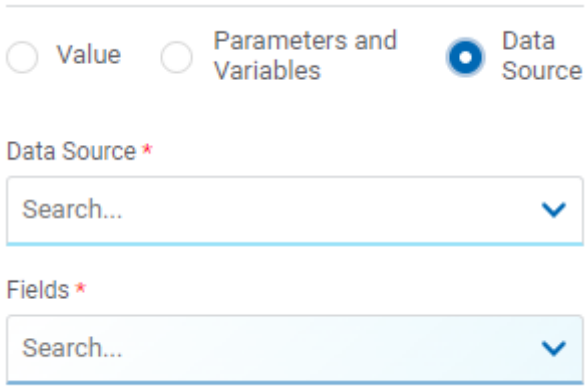
Value
Allows to enter a text that is displayed in the element.

Parameters and Variables



Parameters and Variables
Allows to select a parameter or a variable from, whose value is displayed in the element.

Data Source



Data Source

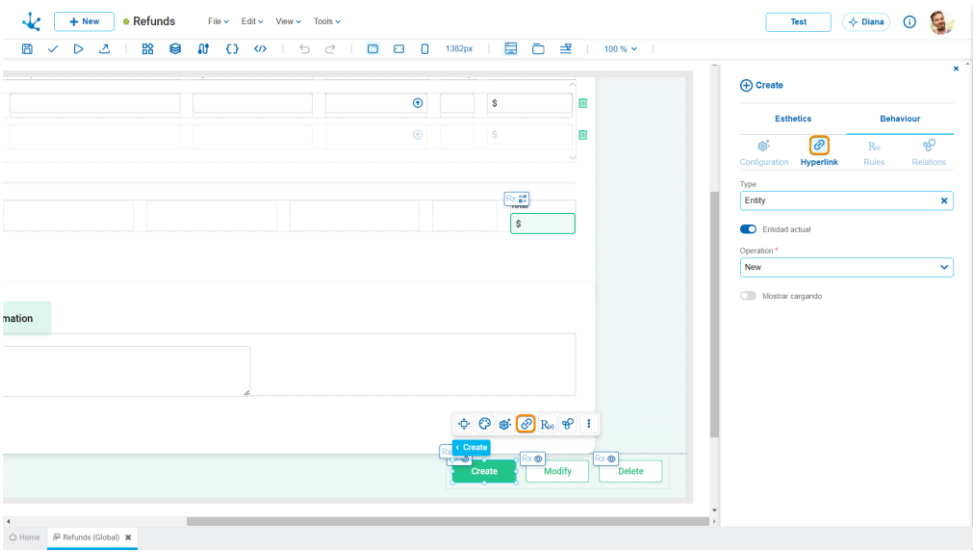
It allows selecting a previously defined data source within the object.

Fields

It allows selecting a field from the data retrieved in the chosen data source, whose value is displayed in the element.

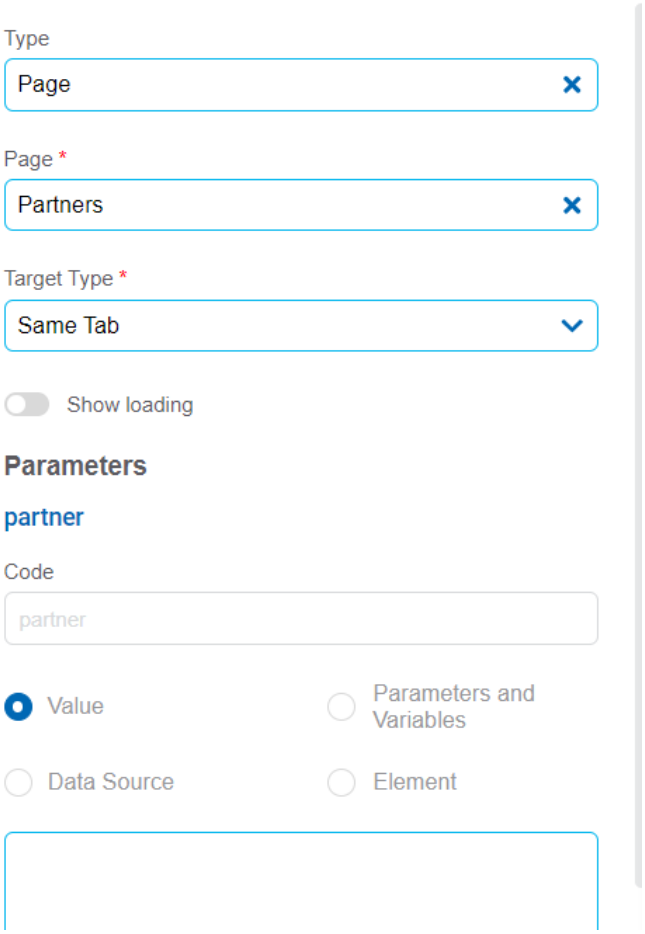
Hyperlink Properties

The hyperlink properties panel of an element opens when clicking the icon of its context menu.



Define the behavior of the element when selecting it. Different properties are enabled depending on the type of object selected.

Page



Page
The pages modeled in the environment are displayed.

Destination Type
The available options for opening the object are displayed.

Possible Values

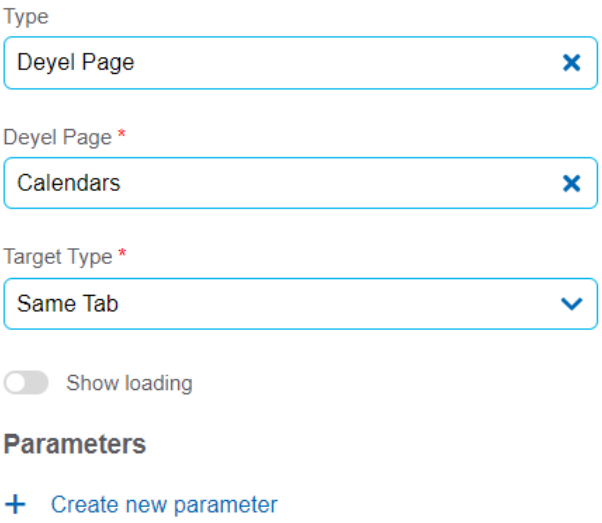
- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  Modal Horizontal Size
  Defines its width.
  Modal Vertical Size
  Defines its height.
- Iframe: selecting this option enables an additional property.
  Destination Iframe
  Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading

Shows the loading icon and disables the user interaction until the redirection is over.

Parameters
If there are parameters, their value can be specified with text, parameters and variables or data source.

Deyel Page



Deyel Page
The pages belonging to **Deyel** are displayed.

Destination Type
The available options for opening the object are displayed.

Possible Values

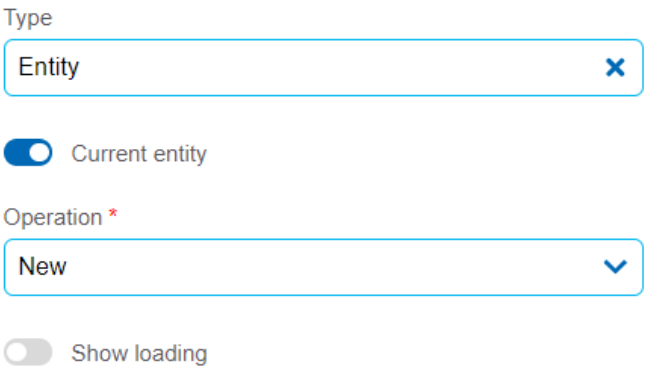
- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  - Modal Horizontal Size
    Defines its width.
  - Modal Vertical Size
    Defines its height.
- Iframe: selecting this option enables an additional property.
  - Destination Iframe
    Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Parameters

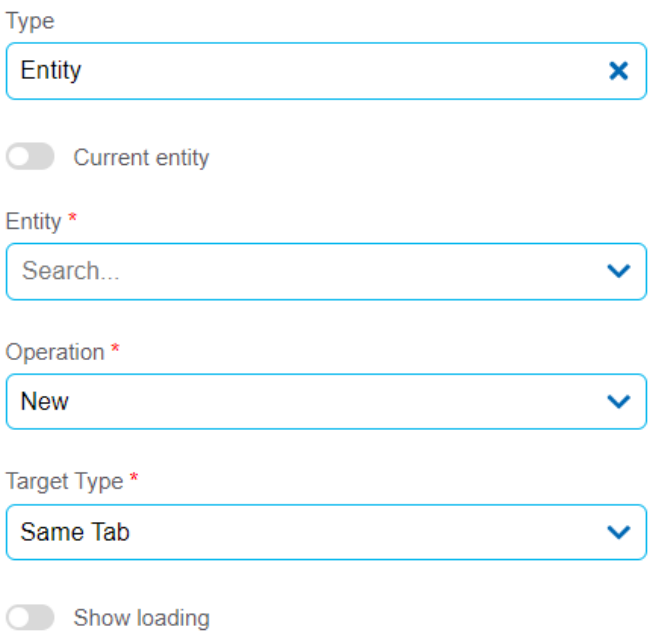
Allows sending parameters to the selected object type.

Entity



Current Entity
If this property is checked, the operation applies directly to the entity being modeled.

Operation
Defines the operation made when selecting the object.
Possible Values
- Create: Indicates that the panel of the current entity is opened to create an instance.
- Update: Indicates that the panel of the current entity is opened to update an instance.
- Delete: Indicates that the panel of the current entity is opened to delete an instance.
- Send Email: It is visible if the property Send Emails is enabled. Clicking on the modeled element opens a panel during instance execution to send an email. The panel size is adjusted with the properties Modal Width and Modal Height.
- Emails: Visible if the Send Emails property is enabled. Clicking on the modeled element opens a panel during instance execution to manage sent emails. The panel is displayed in the iframe defined in the Target Iframe property.



Entity
If the property Current Entity, is not checked, the entities modeled in the environment are displayed.

Operation
Defines the operation made when selecting the object.

Possible Values

- New: The panel of the entity selected in the previous property is opened to create an instance.
- Update The panel of the entity selected in the previous property is opened to update an instance.
- Show: The panel of the entity selected in the previous property is opened to show an instance.
- Grid: Indicates that the grid of the entity selected in the previous property is opened.

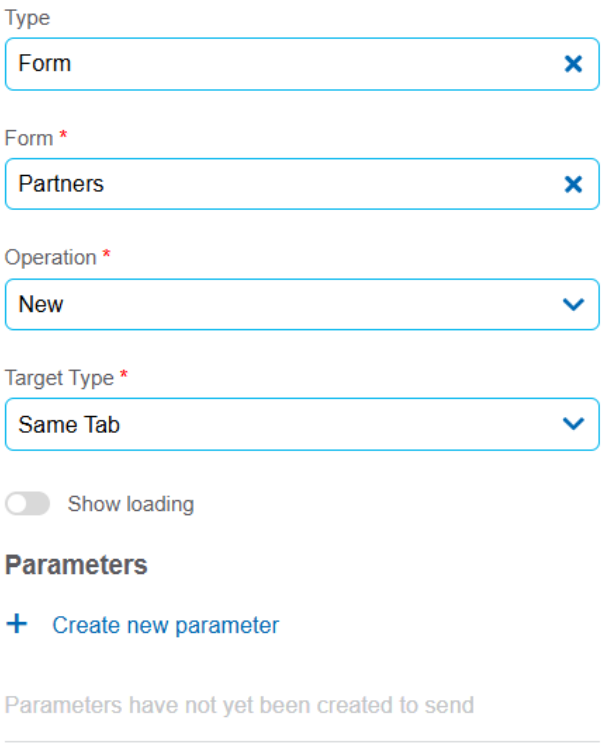
Destination Type
The available options for opening the object are displayed.

Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  Modal Horizontal Size
  Defines its width.
  Modal Vertical Size
  Defines its height.
- Iframe: selecting this option enables an additional property.
  Destination Iframe
  Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Forms



Forms
The forms modeled in the environment are displayed.

Operation
Defines the operation made when selecting the object.

Possible Values

- Create: Indicates that the panel of the object selected in the previous property is opened for the creation of an instance.
- Update: Indicates that the panel of the object selected in the previous property is opened for the update of an instance.
- Show: Indicates that the panel of the object selected in the previous property is opened for the query of an instance.
- Grid: Indicates that the grid of the object selected in the previous property is opened.

Destination Type
The available options for opening the object are displayed.

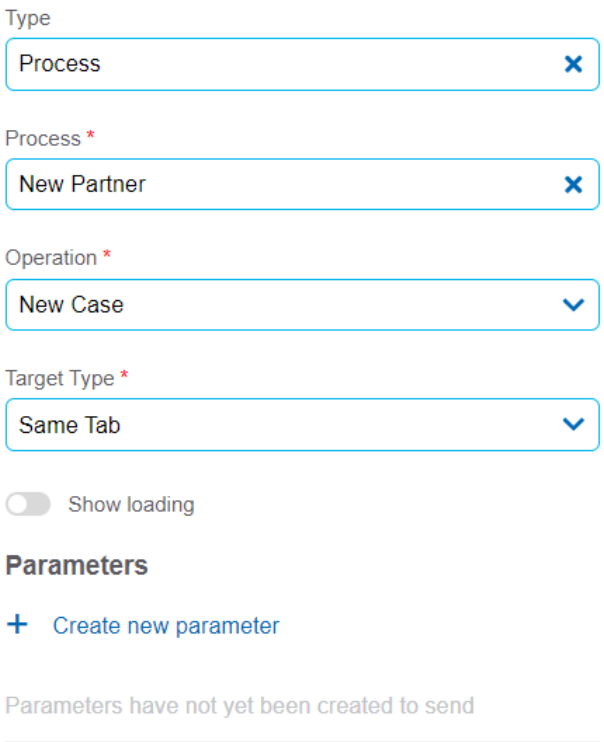
Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  Modal Horizontal Size
  Defines its width.
  Modal Vertical Size
  Defines its height.
- Iframe: selecting this option enables an additional property.
  Destination Iframe
  Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Parameters
Allows sending parameters to the selected object type.

Process



Process
The processes modeled in the environment are displayed.

Operation
Defines the operation made when selecting the object.

Destination Type
The available options for opening the object are displayed.

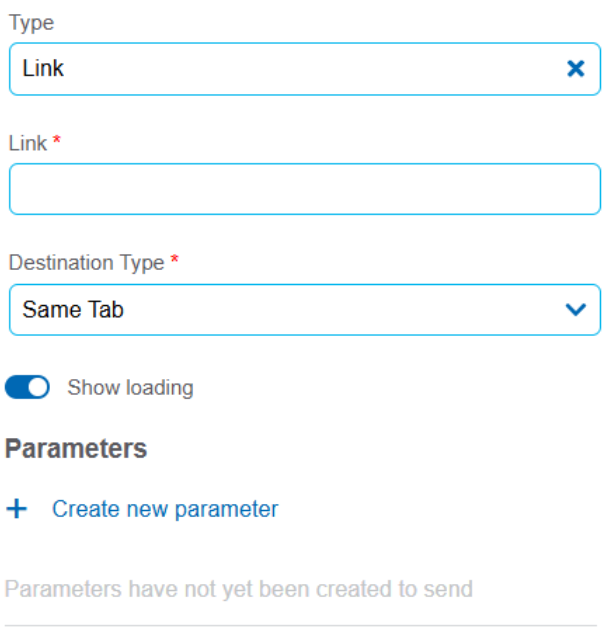
Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  Modal Horizontal Size
  Defines its width.
  Modal Vertical Size
  Defines its height.
- Iframe: selecting this option enables an additional property.
  Destination Iframe
  Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Parameters
Allows sending parameters to the selected object type.

Link



Link
Allows to enter any link.

Destination Type
The available options for opening the object are displayed.

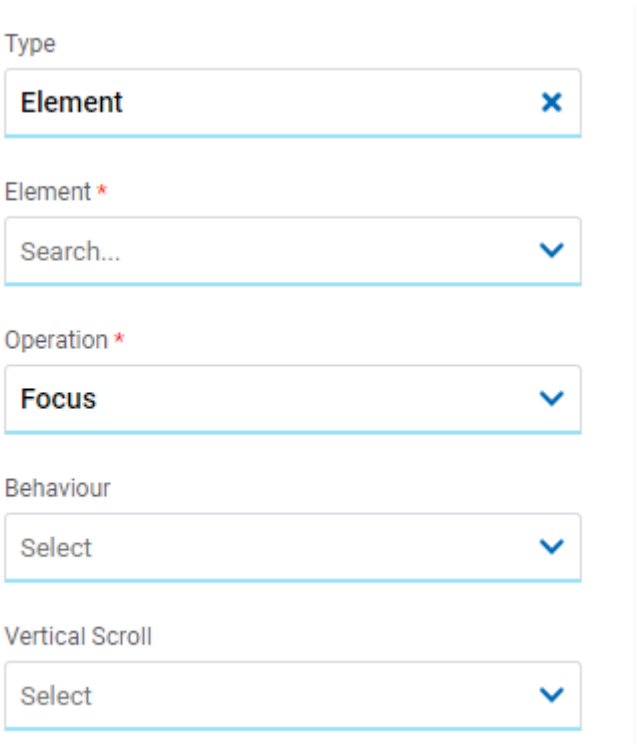
Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  Modal Horizontal Size
  Defines its width.
  Modal Vertical Size
  Defines its height.
- Iframe: selecting this option enables an additional property.
  Destination Iframe
  Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Parameters
Allows sending parameters to the selected object type.

Element



Element
The modeled elements in the object are displayed.

Operation
Defines the operation made when selecting the element.

Possible Values

- Focus
- Show
- Hide
- Show/Hide

Behaviour
Establishes the transition of the animation. It is only modelable if the "Focus" operation was selected.

Possible Values

- Auto
- Smooth

Vertical Scroll
It is only modelable if the "Focus" operation was selected.
Possible Values

- Start: Moves to the start of the selected element.
- Closest: Moves to the closest position to the selected element from the element the event fires.
- Center: Moves to the center of the selected element.
- End: Moves to the end of the selected element.

Repeater



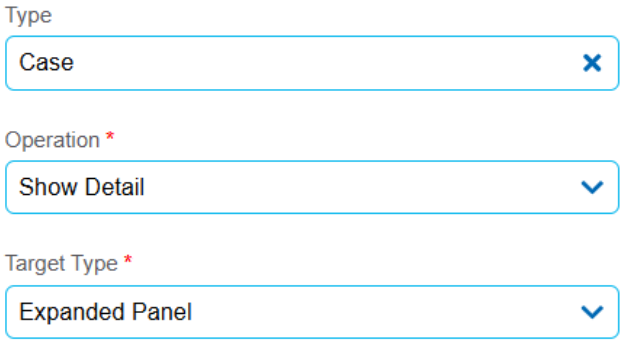
Repeater
The modeled repeaters from the object are displayed.

Operation
Defines the operation made when selecting the element.

Possible Values

- New: Allows for the creation of an item within the repeater. If this option is selected the additional properties are enabled.
  Creation Order
  It indicates in which position the new item of the container is added.
  Possible Values
  - Home
  - End
- Delete: Allows for the deletion of an item within the repeater.

Case



Case
Show case information.

Operation
Defines the operation made when selecting the element.

Possible Values

- Show Detail
- Show Graphic Execution

Destination Type
The available options for opening the object are displayed.

Possible Values

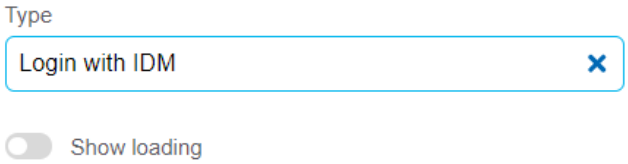
- Expanded Panel
- Modal: if this option is selected the additional properties are enabled.
  Modal Horizontal Size
  Define its width.
  Modal Vertical Size
  Define its height.

Back



It allows associating the event to go back in the browser to the element.

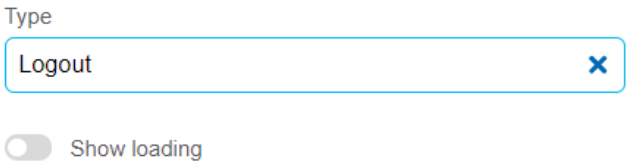
LogIn with IDM



Allows login with IDM.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

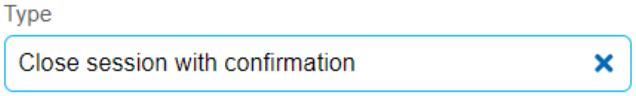
Logout



Allows logging the user out.

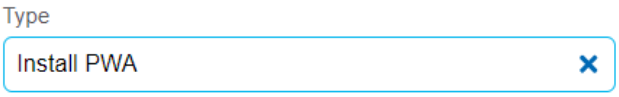
Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Close Session with Confirmation



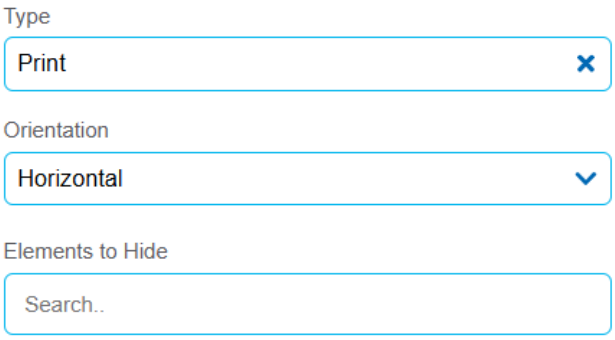
Displays a confirmation modal to logout the user.

Install PWA



Allows to install the application in the browser.

Print



This type of hyperlink associated with an element allows printing the content of the entity or page that is being executed.

Orientation

Determines the layout of the content to be printed.

Possible Values

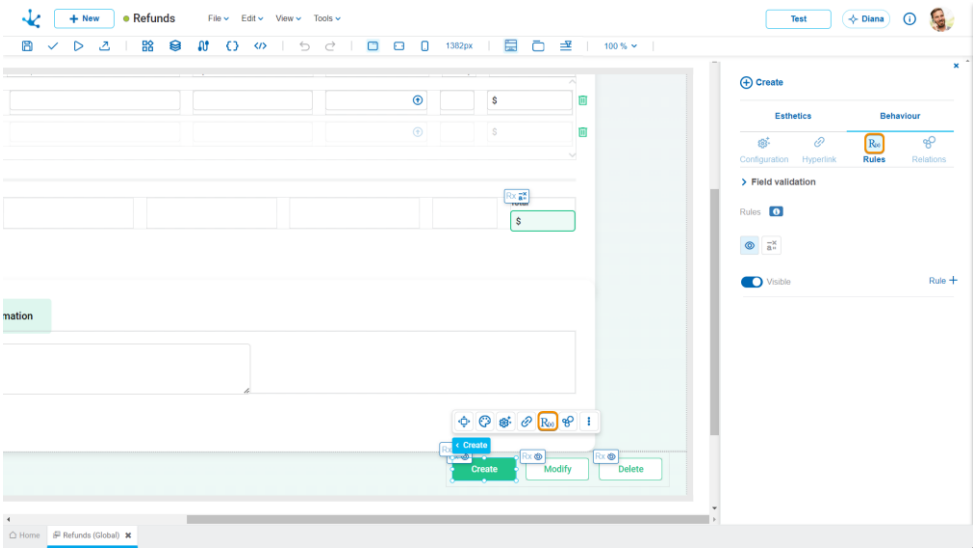
- Horizontal
- Vertical

Elements to hide

It allows selecting which elements of the entity or page should not be included in the printout.

Rule Properties

The rule properties panel of an element opens when clicking the icon  of its context menu.



Properties

Rules

Embedded rules on behavior, validation and calculation can be defined, associated with an element, by using the wizard (ctrl + space).

Shows syntax examples for writing the rules.

Visible

Indicates whether the element is visible. If this property is not checked, the element is not displayed in the page.

Visible (default) Not visible

Rule + Opens an edit area where a rule to determine the visibility condition can be defined. If a rule is defined, the icon is displayed with light blue borders.

Calculation

Rule + Opens an edit area where you can define the expression to be executed to calculate the element value. If a rule is defined, the icon is displayed with light blue borders.

Operations when defining the rule:

Saves the new or modified rule

Cancels the operation

Operations once the rule is defined:

Edits the existing rule

Deletes the rule

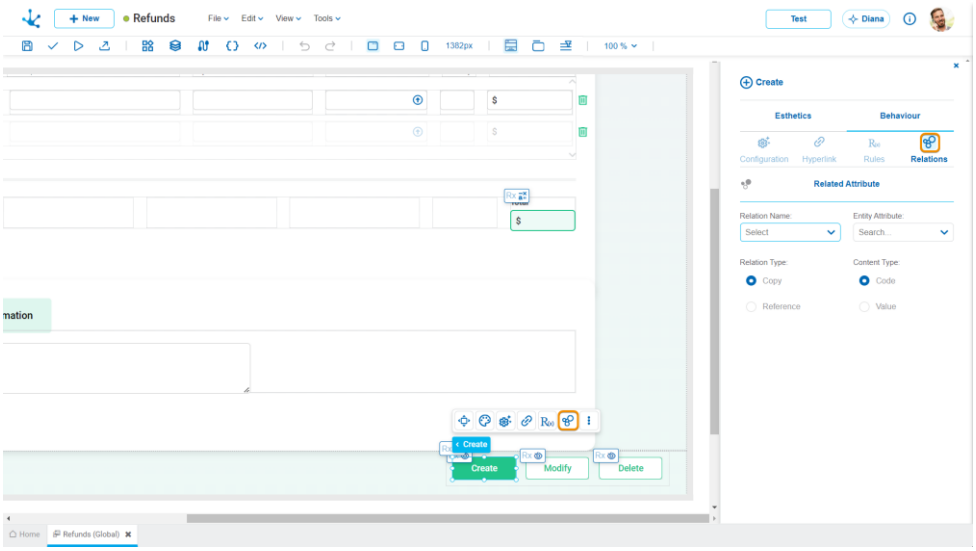
Events

Buttons allow the use of different events.

Event	Description
onMouseIn()	It is executed when the cursor is positioned on the element.
onMouseOut()	It is executed when the cursor moves out of the element.
onClick()	It is executed when clicking on the element.
onDoubleClick()	It is executed when clicking twice on the element.
onInit()	It is executed before the element is loaded.
afterViewInit()	It is executed after viewing the element.
onViewportEnter()	It is executed when the element is visible.
onViewportLeave()	It is executed when the element is no longer visible.

Relation Properties

The relation properties panel of an element opens when clicking the icon  of its context menu.



Related Attribute

A related attribute indicates that the field value is retrieved from the attribute value of another entity. One of the relations defined on the entity must be selected, and then it should be specified which attribute of the related entity it links to.

A field can be modeled as a related attribute and it can also have a relation defined to an entity, a value list, or a rule. This allows the value of the related attribute to be retrieved on this field and the functionality provided by the relation to be used.

Properties

Relation Name
The list of relations defined in the entity is displayed, allowing the selection of one of them.

Entity Attribute
The list of attributes of the selected entity compatible with the field type is displayed. Attributes can have a relation modeled with an entity, a value list, or a rule in the selected entity.

Relation Type
Copy
When this property is checked, once the retrieved value is uploaded, it does not automatically update even if the value of the related attribute changes.
Reference
When this property is checked, the field value is subject to the current value of the related attribute, that is, the updated value is automatically displayed.

Content Type
The element content varies depending on whether the Code or Value property is checked when:
- The entity attribute has a modeled relation.
- The field has the option "Entity" selected in the Values Obtained from property.

Code
When this property is checked, if the relation is modeled with a:
- Related entity, the identifier of the entity is displayed.
- Value list, the value list code is displayed.
- Rule, the code returned by the rule's execution is displayed.

Value
When this property is checked, if the relation is modeled with a:
- Related entity, the short description of the entity is displayed.
- Value list, the value list description is displayed.
- Rule, the description returned by the rule's execution is displayed.

Container

Container boxes are used to create an adaptive and structured layout for the entity. The elements are dragged into the container and then styled for different breakpoints. The elements automatically attach to the container to resize in direct relation to it.

The entity modeler also includes other structural elements that facilitate field modeling, improving the time it takes to model an entity without worrying about the aesthetic aspects of the fields. All aesthetic aspects of these elements are defined by default to adapt to the different breakpoints.

The element properties are represented by icons in its context menu, where its operations are also available.

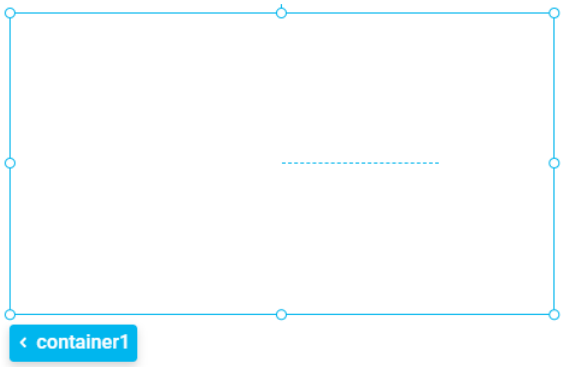
Subtypes

When selecting the option "Container" from the icon in the top toolbar, a list opens with the different subtypes of this element, which can be dragged and dropped into the modeling area. Each subtype has the element's properties modeled and built in a specific way.

- Simple
- Background Color
- Border
- Shadow
- Collapsible
- Repetitive Field Group
- Collapsible Repetitive Card Group
- Collapsible Repetitive Tape Group

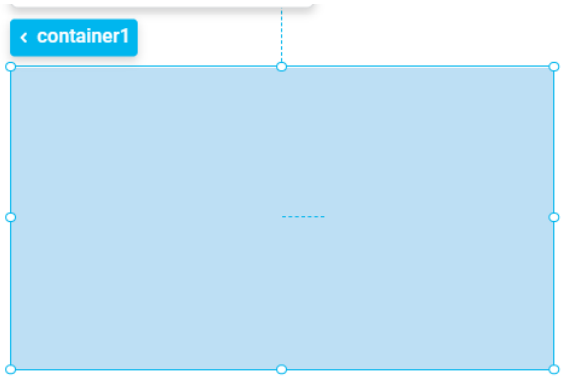
Simple

It allows modeling a simple-style container. The content can be any element.



Background Color

It allows modeling a container with a preset background color, which can be modified. The content can be any element.



Border

It allows modeling a container with a preset background color and a border, which can be modified. The content can be any element.

Shadow

It allows modeling a container with a preset background color, a border, and a shadow, which can be modified. The content can be any element.

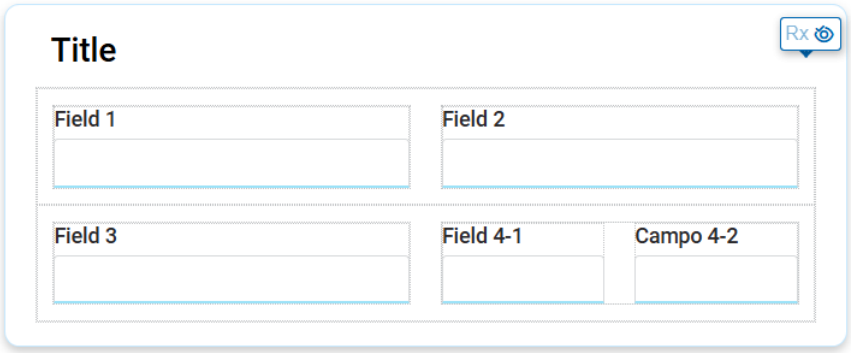


Collapsible

It allows grouping fields and offer drop-down behavior. This means that its content can be easily expanded or closed, by clicking on the corresponding icon, ↓ or ↑, depending on the state of the container.

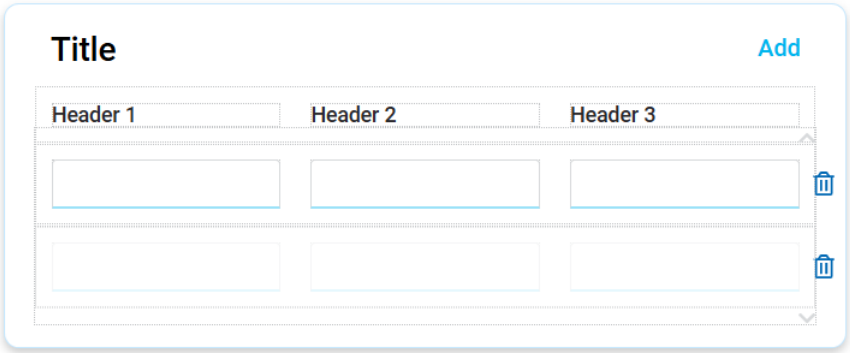
This container not only groups fields, but can also contain any other element, providing great flexibility in design.

The collapsible container is primarily used within design elements, to visually organize entities with a large number of fields. By expanding or collapsing the container, information can be viewed in a more organized manner, making it easier to navigate through data groups.



Repetitive Group of Fields

It allows organizing repetitive field structures in a grid format, with descriptions for each field and headers for columns. Additionally, being a repetitive structure, it includes the facility to add and delete rows within the grid.

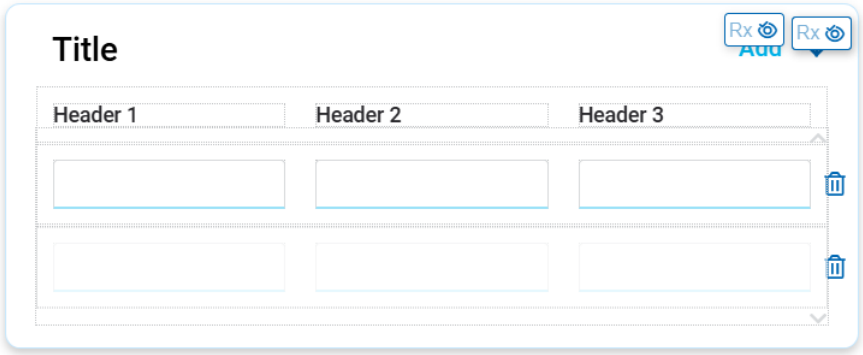


Collapsible Repetitive Card Group

It allows organizing repetitive field structures in a grid format, with descriptions for each field and headers for the columns. Additionally, being a repetitive structure, it includes the facility to add and delete rows within the grid.

This container has a fixed size, and items are scrolled page by page within the repeater, using the icons and . In the repeater configuration properties panel, it is modeled as subtype "card" and the number of rows to paginate.

The collapsible repetitive tape group is ideal for saving multiple instances of the same type of information while making it easier to display entities with many fields.

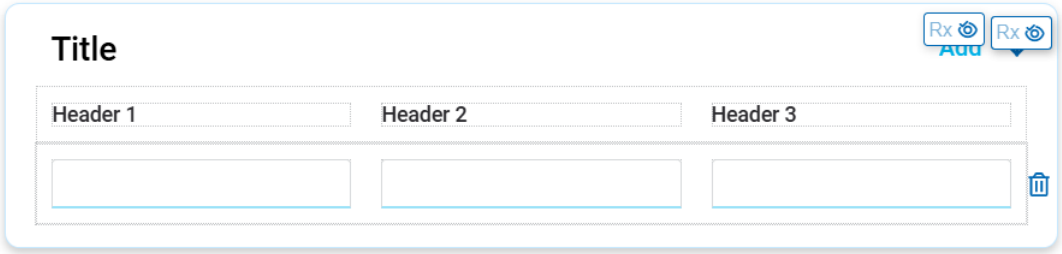


Collapsible Repetitive Tape Group

It allows organizing repetitive field structures in a grid format, with descriptions for each field and headers for the columns. Additionally, being a repetitive structure, it includes the facility to add and delete rows within the grid.

This container expands indefinitely as new items are added, offering greater flexibility. In the repeater configuration properties panel, it is modeled as "tape" subtype.

The collapsible repetitive tape group is ideal for saving multiple instances of the same type of information while making it easier to display entities with many fields.



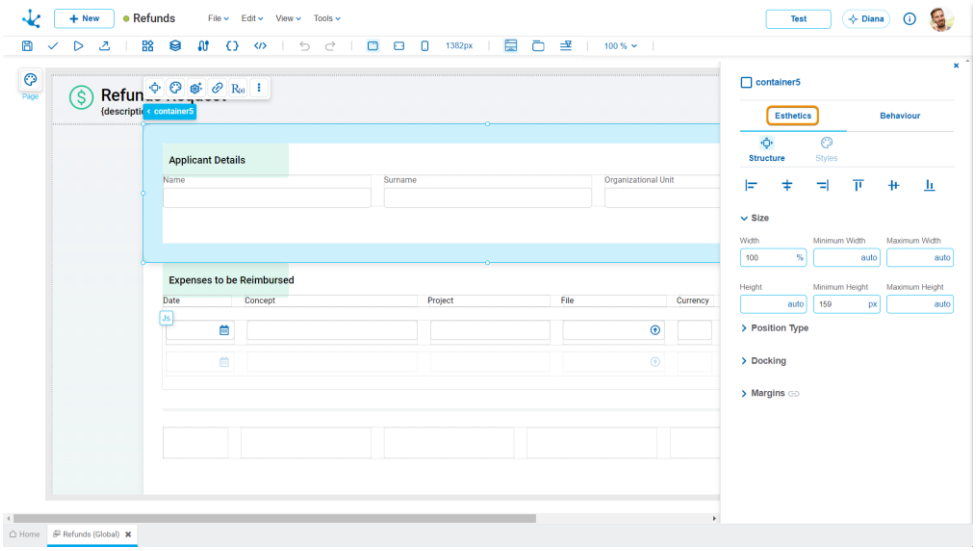
Classification of Properties

Properties are divided into two groups: esthetic properties and behavior properties.

Esthetic Properties

In the esthetic properties panel, the following are grouped:

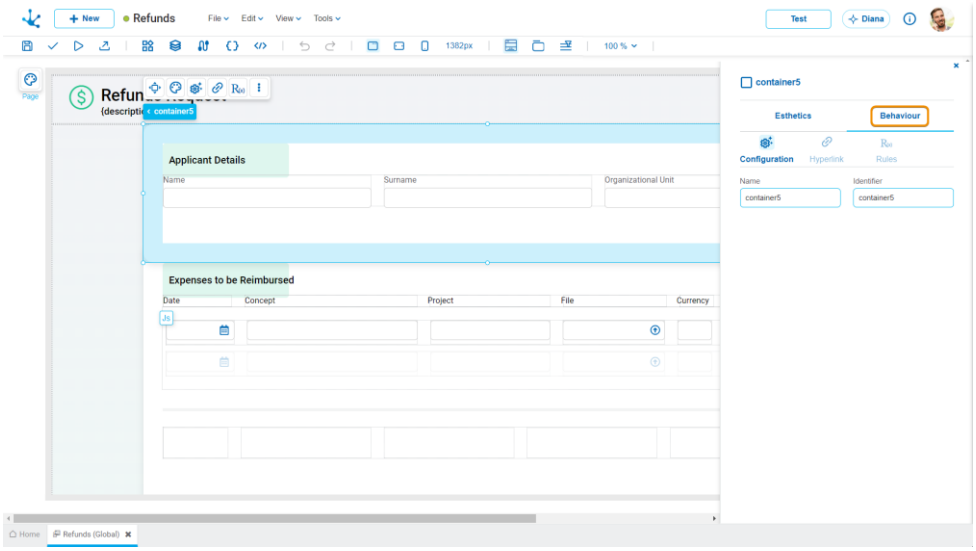
- Structure Properties
- Style Properties



Behavior Properties

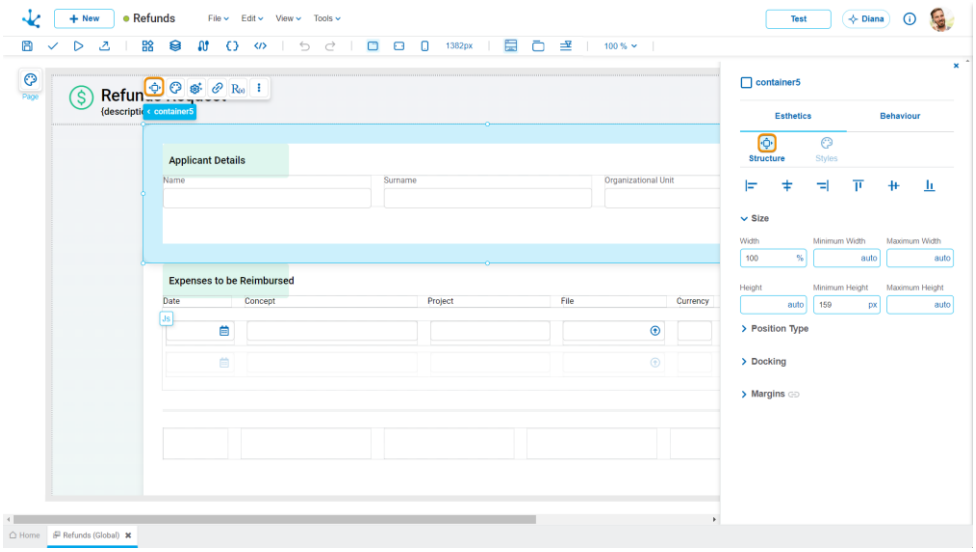
In the behavior properties panel, the following are grouped:

- Configuration Properties
- Hyperlink Properties
- Rule Properties



Structure Properties

The structure properties panel of an element opens when selecting the icon  of its context menu.

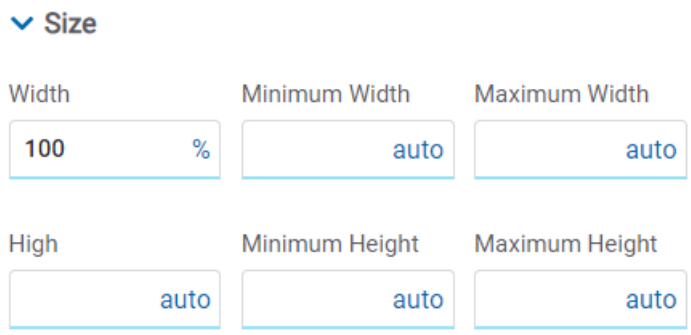


Alignment

Allows to align the element by selecting one of the available icons.

- Align to the left.
- Align to horizontal center.
- Align to the right.
- Align up.
- Align vertical center.
- Align bottom.

Size

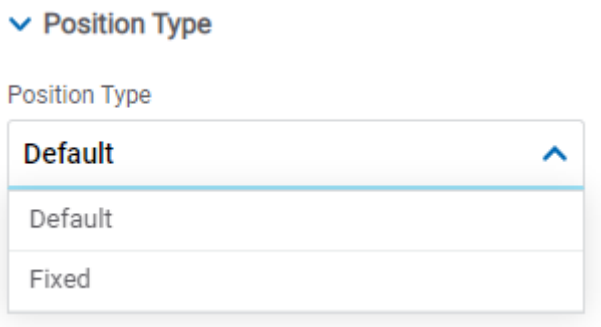


All size properties can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh). Additionally, for Width and Height properties, the “auto” option is added.

> *If selecting percentage (%) for the width and height properties, the size is calculated relative to the top element.*

Position Type

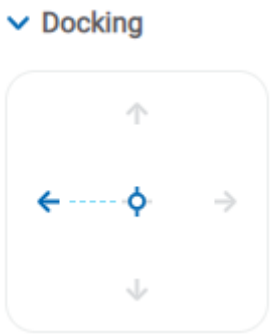
It determines if at the time of execution the element remains fixed on the page or if it moves as the scroll bar moves forward or backward.



Possible Values

- Default: The element has a relative position with respect to the superior element where it was placed (container or section).
- Fixed: Fixes the element to the page so that it is always visible in the same position. It allows to increase the visibility of important content.

Docking



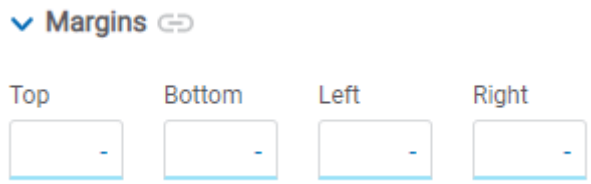
Determines the vertical and horizontal position of elements within the page section or container, when the screen is resized.

When an element is added or moved within the section or container, the modeler automatically docks it to the nearest corners or edges. It can also be configured by clicking on the arrows of the side to be docked to or on the center.

Once the element is docked, its exact position is indicated in the Margins property. The direction of the docking arrows determine the values that are automatically displayed for margins.

The docking position is indicated in the modeling area by dotted lines on the element.

Margins



It allows to define the distance of elements from the borders of their top element. The behavior of margins depends on the docking of the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Top
Distance to the top border of the highest ranking element.

Bottom
Distance to the bottom border of the highest ranking element.

Left
Distance to the left border of the highest ranking element.

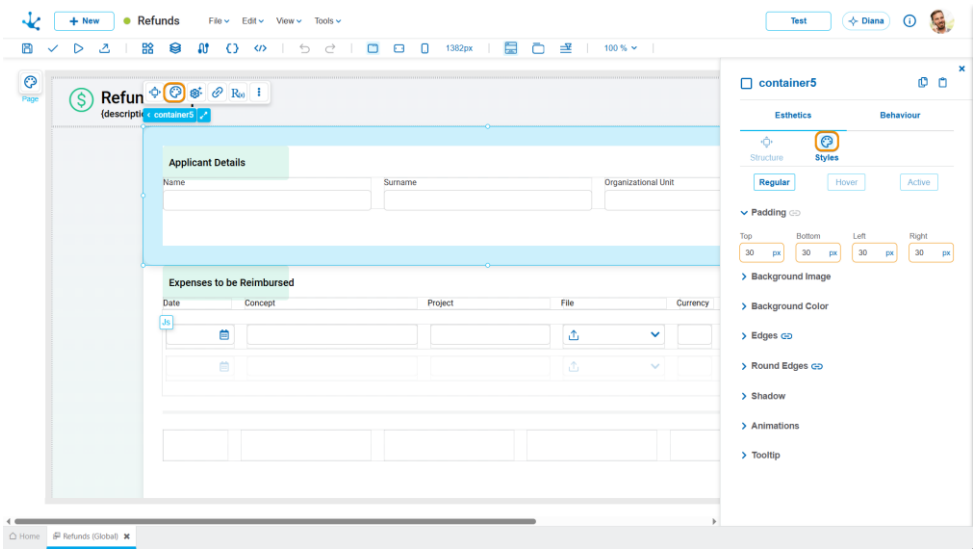
Right
Distance to the right border of the highest ranking element.

Allows the value entered in one of the margins to be copied to the other ones automatically.

Allows to indicate different values for each margin.

Style Properties

The style properties panel of an element opens when selecting the icon of its context menu.



This type of element may take different states and for each of them different values for its properties may be modeled.



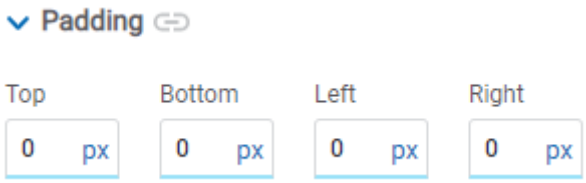
- Regular: The mouse pointer is not over the element.
- Hover: The mouse pointer is over the element.

Styles

It allows copying the entire current configuration of the style panel for the selected element (Ctrl+Alt+C).

It allows applying the previously copied configuration to another element of the same type (Ctrl+Alt+V).

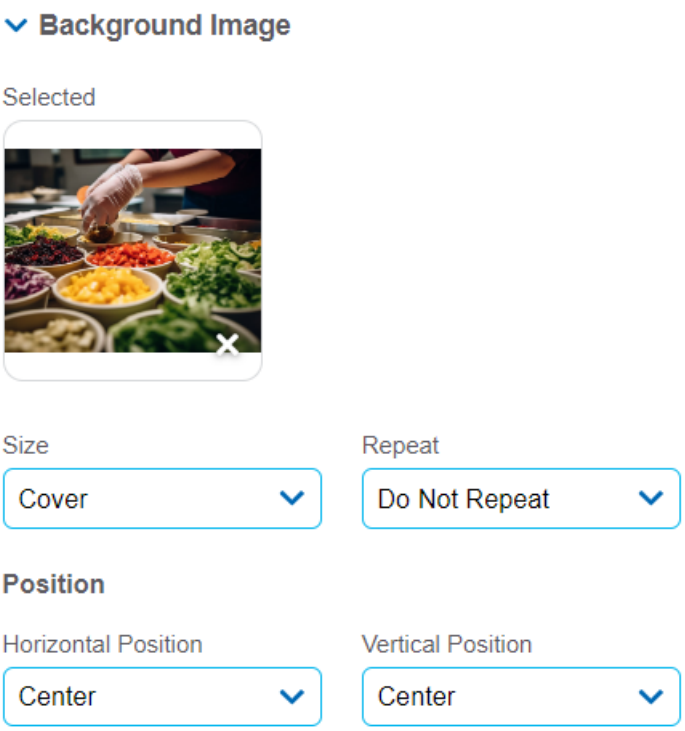
Padding



All padding space properties allow to create a space around the element borders (top, bottom, sides) and its content. The padding space configured is inward, while the margin is configured relative to a top element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Allows values entered in one of the paddings to be copied to the other ones automatically.
Allows to indicate different values for each padding.

Background Image



Allows to add a background image to the element.

Selected

An image can be uploaded from the computer where it is being modeled.

Size

Possible Values

- Cover
- Content
- Auto

Repeat

Possible Values

- Repeat
- Repeat Hor.
- Repeat Vert.
- Spacing
- Rounded
- Do Not Repeat

Horizontal Position

Possible Values

- Center

- Left
- Right

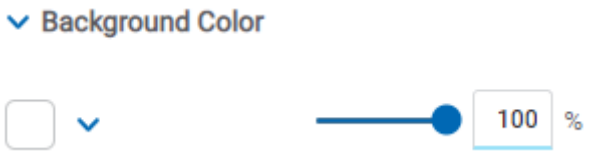
Vertical Position

- Center
- Top
- Botton

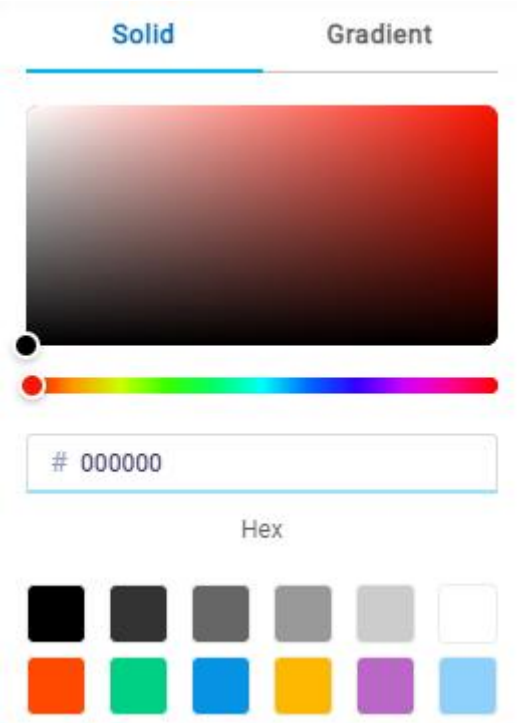
Background Color

It allows to model a background color for the element, it can be solid or gradient and different properties are defined for each one.

Solid



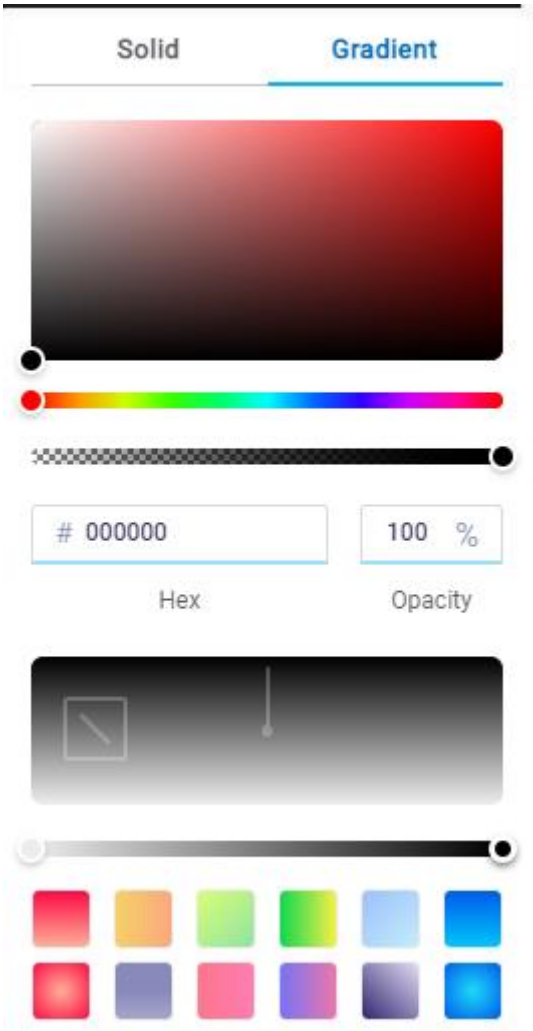
This option enables not only to select the color from a palette but also to define the degree of transparency.



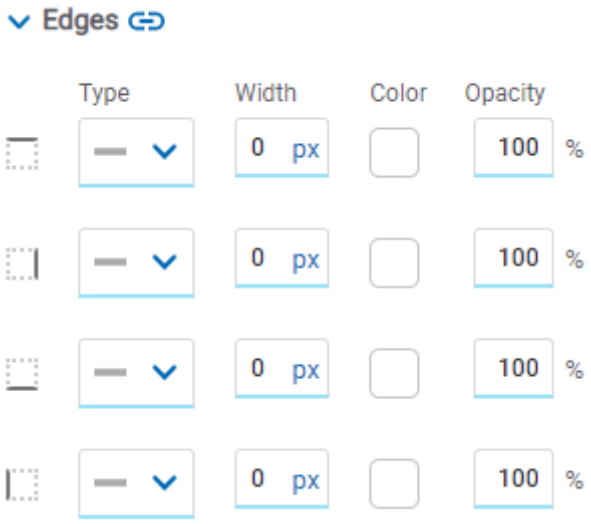
Gradient



This option enables to select not only the color from a palette but also to define its opacity and intensity according to the selected angle.



Edges



It allows to define the style of borders. Each one has its type, width, color and opacity defined, the latter as a percentage.

Allows the value entered in one of the borders to be copied to the other ones automatically.
Allows to indicate different values for each border.

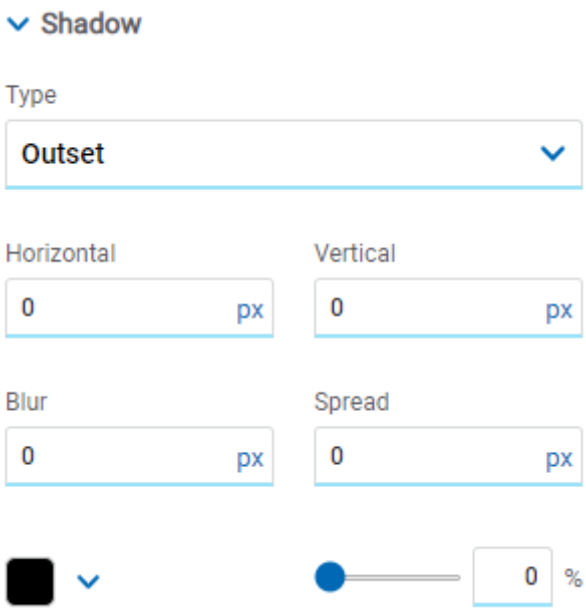
Round Edges



It allows to define the round edges at the corners of elements.
It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Allows the value entered in one of the borders to be copied to the other ones automatically.
Allows to indicate different values for each border.

Shadow



Allows to define a shadow effect around the element.

Horizontal
Horizontal size of the shadow to the right of the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Vertical
Vertical size of the shadow below the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

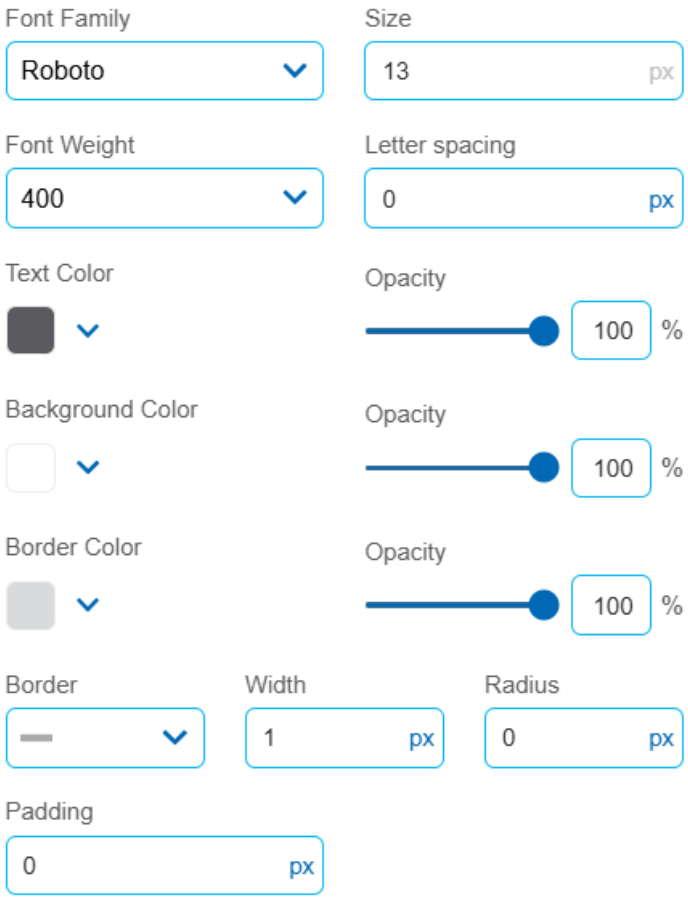
Blur
The greater this value is, the greater and lighter the shadow becomes. If not specified, its value is 0 and the shadow border is darker. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Spread
Positive values will cause the shadow to expand and grow bigger, while negative values will cause the shadow to shrink. If not specified, its value is 0 and the shadow will have the same size as the element.

Color
Defines the color of the shadow.

Transparency
Defines the transparency of the shadow.

Tooltip



Source

It allows customizing the font, size, thickness and letter spacing of the pop-up message.

Colors

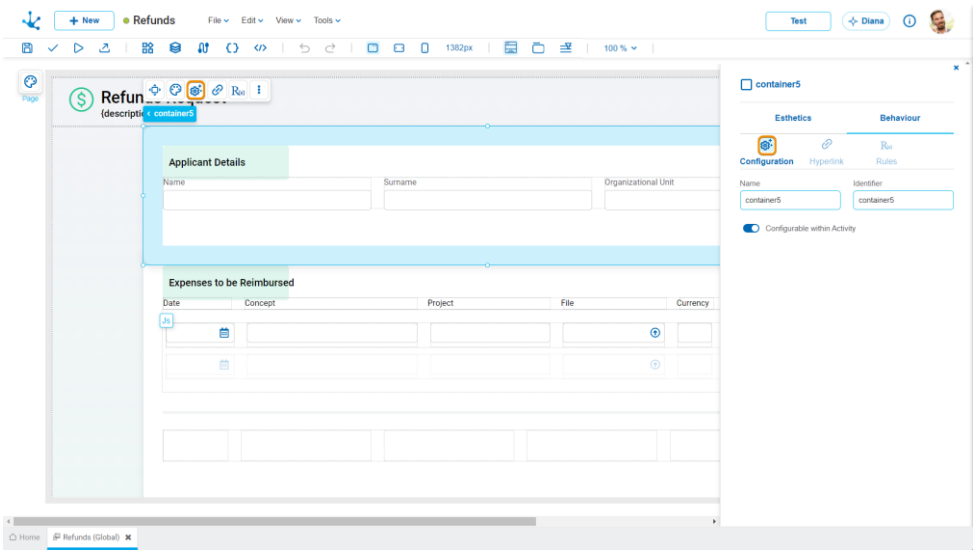
The color and opacity of the text, background and border can be defined. Borders are set uniformly for all sides of the tooltip.

Border

The border type, width, and border radius can be configured.

Configuration Properties

The configuration properties panel of an element opens when clicking the icon  of its context menu.



Name
Name used to reference the element during modeling.

Identifier
Uniquely identifies the element. It is used in the Javascript SDK.

Tooltip
Pop-up message that appears when hovering over an element, providing additional information.



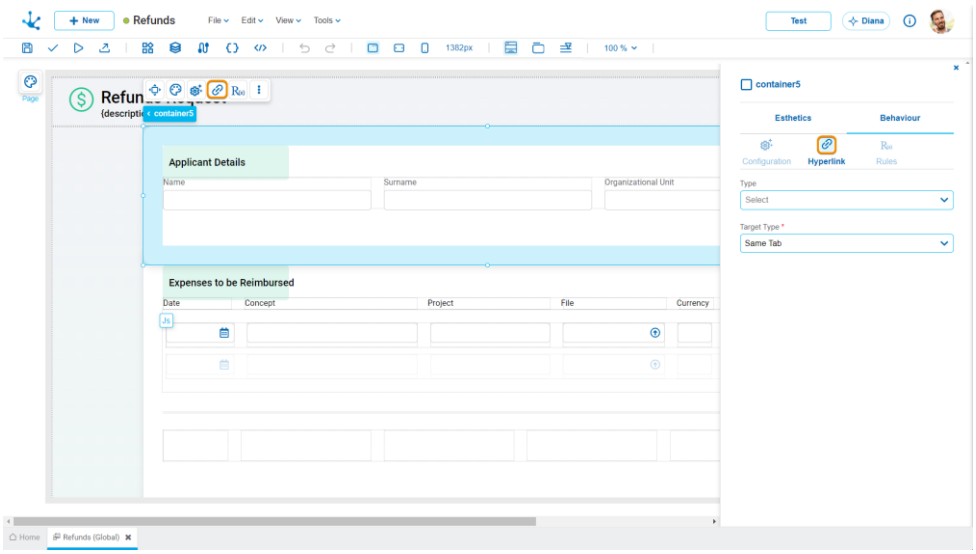
Text
Content of the message being displayed.

Direction
Position of the message in relation to the element (top, bottom, left, or right).

Configurable within Activity
This property is displayed only if the entity is related to a process. When activated, the element is enabled in the execution panel of the associated process, allowing to configure its visibility within their various activities.

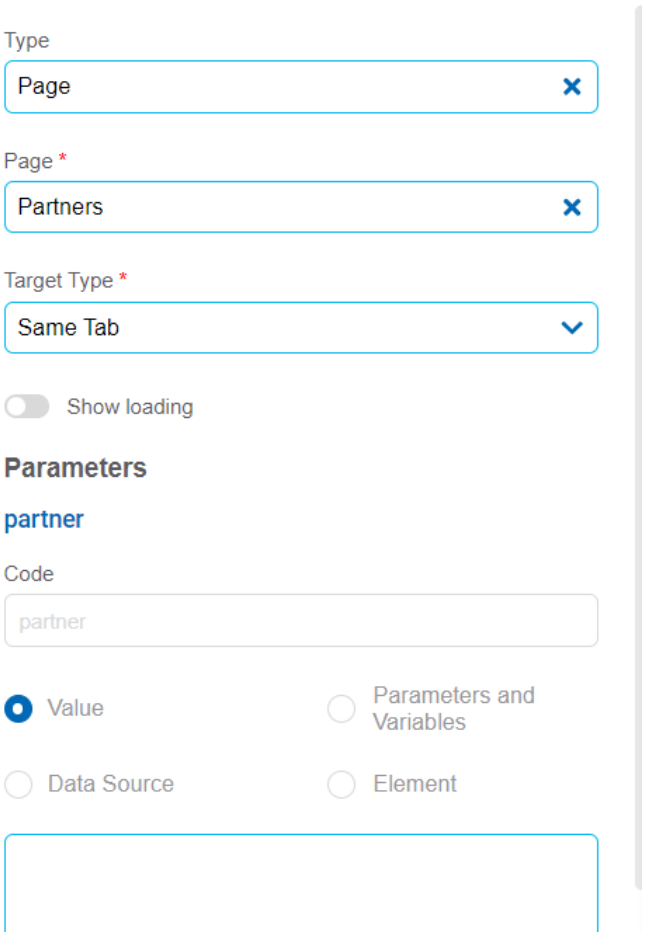
Hyperlink Properties

The hyperlink properties panel of an element opens when clicking the icon  of its context menu.



Define the behavior of the element when selecting it. Different properties are enabled depending on the type of object selected.

Page



Page
The pages modeled in the environment are displayed.

Destination Type
The available options for opening the object are displayed.

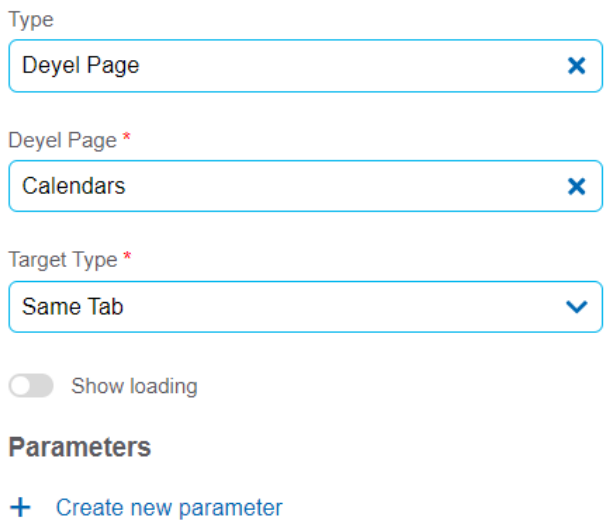
Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  - Modal Horizontal Size
    Defines its width.
  - Modal Vertical Size
    Defines its height.
- Iframe: selecting this option enables an additional property.
  - Destination Iframe
    Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Parameters
If there are parameters, their value can be specified with text, parameters and variables or data source.

Deyel Page



Deyel Page
The pages belonging to **Deyel** are displayed.

Destination Type
The available options for opening the object are displayed.

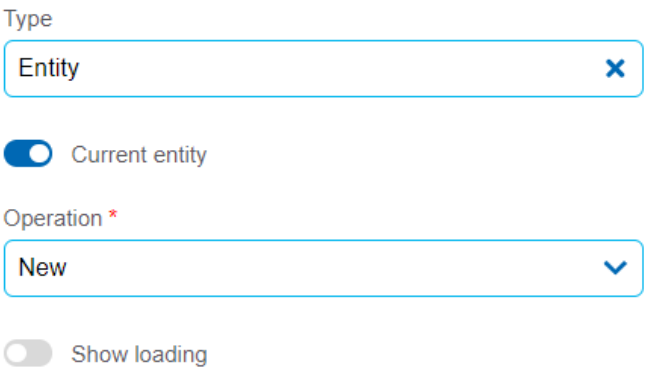
Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  - Modal Horizontal Size
    Defines its width.
  - Modal Vertical Size
    Defines its height.
- Iframe: selecting this option enables an additional property.
  - Destination Iframe
    Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Parameters
Allows sending parameters to the selected object type.

Entity



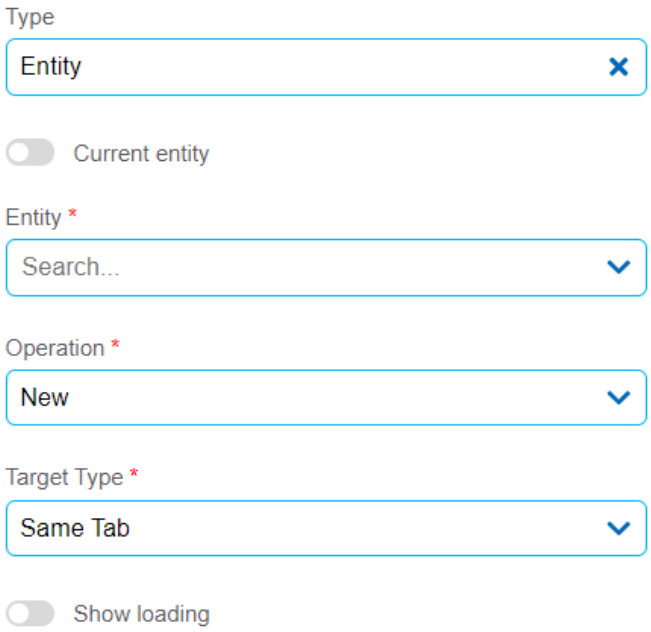
Current Entity
If this property is checked, the operation applies directly to the entity being modeled.

Operation
Defines the operation made when selecting the object.

Possible Values

- Create: Indicates that the panel of the current entity is opened to create an instance.
- Update: Indicates that the panel of the current entity is opened to update an instance.
- Delete: Indicates that the panel of the current entity is opened to delete an instance.

- Send Email: It is visible if the property Send Emails is enabled. Clicking on the modeled element opens a panel during instance execution to send an email. The panel size is adjusted with the properties Modal Width and Modal Height.
- Emails: Visible if the Send Emails property is enabled. Clicking on the modeled element opens a panel during instance execution to manage sent emails. The panel is displayed in the iframe defined in the Target Iframe property.



Entity
If the property Current Entity, is not checked, the entities modeled in the environment are displayed.

Operation
Defines the operation made when selecting the object.

Possible Values

- New: The panel of the entity selected in the previous property is opened to create an instance.
- Update The panel of the entity selected in the previous property is opened to update an instance.
- Show: The panel of the entity selected in the previous property is opened to show an instance.
- Grid: Indicates that the grid of the entity selected in the previous property is opened.

Destination Type
The available options for opening the object are displayed.

Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  Modal Horizontal Size
  Defines its width.

Modal Vertical Size
Defines its height.

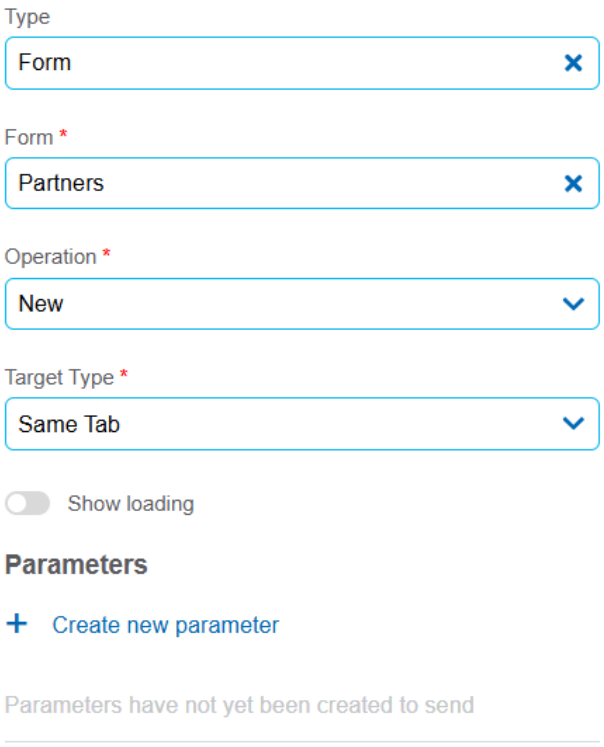
- Iframe: selecting this option enables an additional property.

Destination Iframe
Expands the iframes previously defined on the object.

- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Forms



Forms
The forms modeled in the environment are displayed.

Operation
Defines the operation made when selecting the object.

Possible Values

- Create: Indicates that the panel of the object selected in the previous property is opened for the creation of an instance.
- Update: Indicates that the panel of the object selected in the previous property is opened for the update of an instance.
- Show: Indicates that the panel of the object selected in the previous property is opened for the query of an instance.

- Grid: Indicates that the grid of the object selected in the previous property is opened.

Destination Type
The available options for opening the object are displayed.

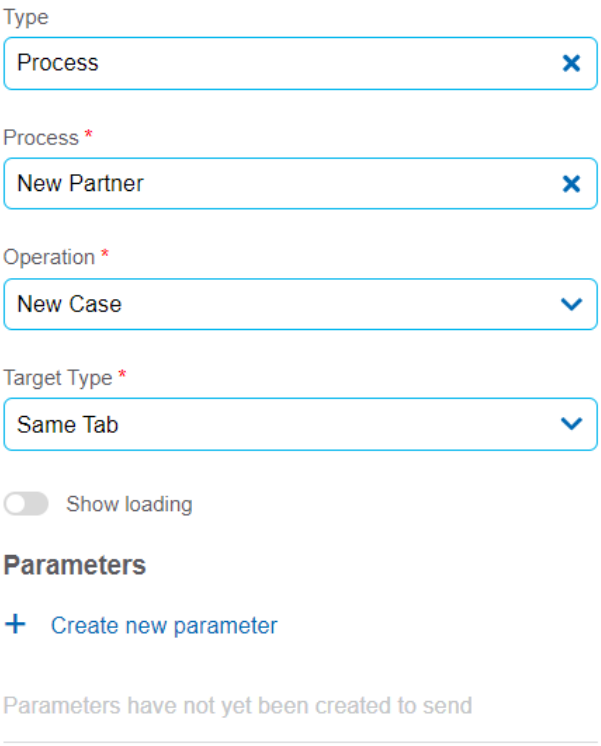
Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  Modal Horizontal Size
  Defines its width.
  Modal Vertical Size
  Defines its height.
- Iframe: selecting this option enables an additional property.
  Destination Iframe
  Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Parameters
Allows sending parameters to the selected object type.

Process



Process
The processes modeled in the environment are displayed.

Operation
Defines the operation made when selecting the object.

Destination Type
The available options for opening the object are displayed.

Possible Values

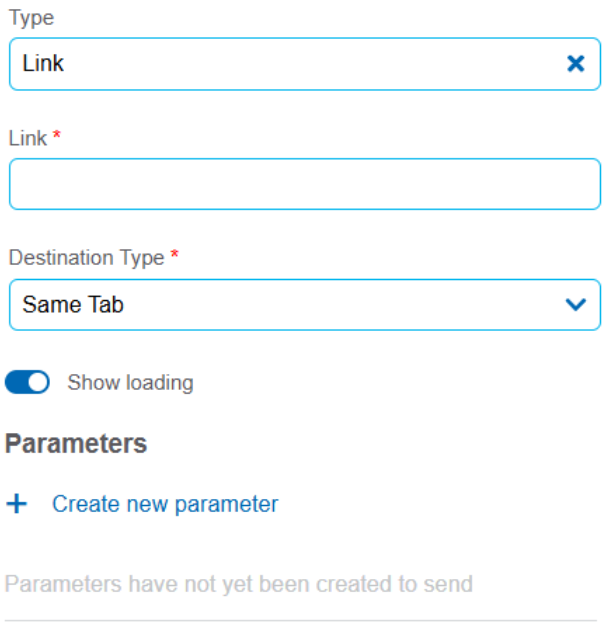
- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  - Modal Horizontal Size
    Defines its width.
  - Modal Vertical Size
    Defines its height.
- Iframe: selecting this option enables an additional property.
  - Destination Iframe
    Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Parameters

Allows sending parameters to the selected object type.

Link



Link
Allows to enter any link.

Destination Type
The available options for opening the object are displayed.

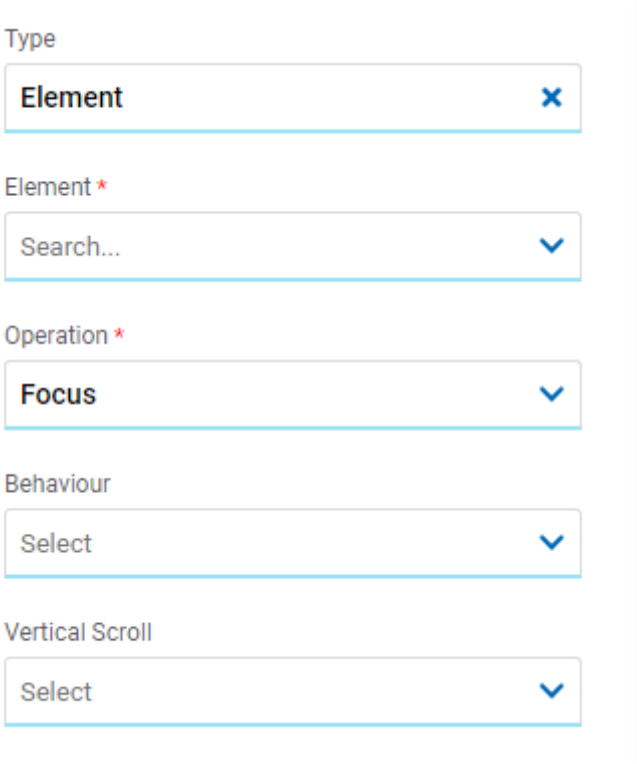
Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  Modal Horizontal Size
  Defines its width.
  Modal Vertical Size
  Defines its height.
- Iframe: selecting this option enables an additional property.
  Destination Iframe
  Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Parameters
Allows sending parameters to the selected object type.

Element



Element
The modeled elements in the object are displayed.

Operation
Defines the operation made when selecting the element.

Possible Values

- Focus
- Show
- Hide
- Show/Hide

Behaviour
Establishes the transition of the animation. It is only modelable if the "Focus" operation was selected.

Possible Values

- Auto
- Smooth

Vertical Scroll
It is only modelable if the "Focus" operation was selected.
Possible Values

- Start: Moves to the start of the selected element.
- Closest: Moves to the closest position to the selected element from the element the event fires.

- Center: Moves to the center of the selected element.
- End: Moves to the end of the selected element.

Repeater



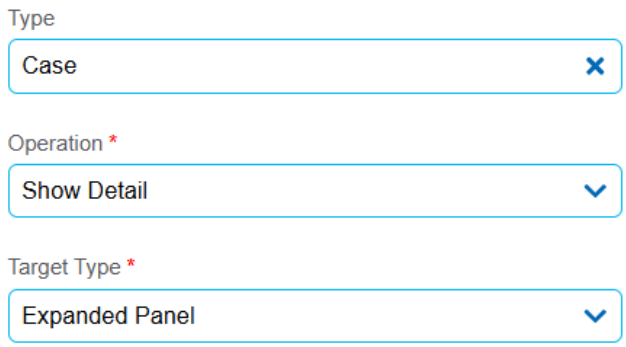
Repeater
The modeled repeaters from the object are displayed.

Operation
Defines the operation made when selecting the element.

Possible Values

- New: Allows for the creation of an item within the repeater. If this option is selected the additional properties are enabled.
  ### Creation Order
  It indicates in which position the new item of the container is added.
  Possible Values
  - Home
  - End
- Delete: Allows for the deletion of an item within the repeater.

Case



Case

Show case information.

Operation
Defines the operation made when selecting the element.

Possible Values

- Show Detail
- Show Graphic Execution

Destination Type
The available options for opening the object are displayed.

Possible Values

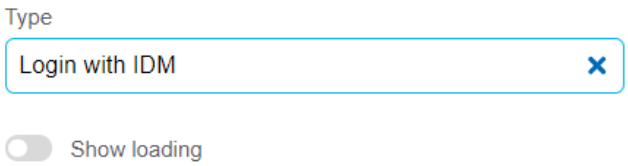
- Expanded Panel
- Modal: if this option is selected the additional properties are enabled.
  Modal Horizontal Size
  Define its width.
  Modal Vertical Size
  Define its height.

Back



It allows associating the event to go back in the browser to the element.

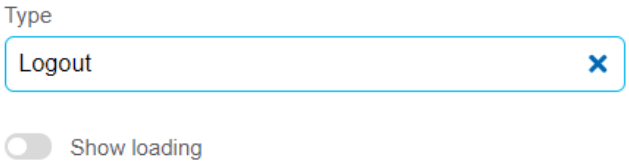
LogIn with IDM



Allows login with IDM.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

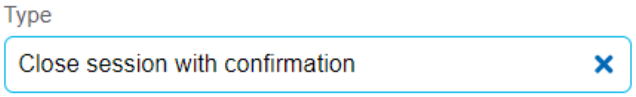
Logout



Allows logging the user out.

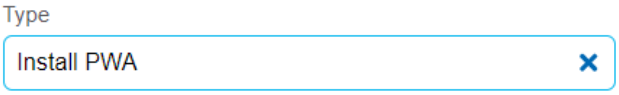
Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Close Session with Confirmation



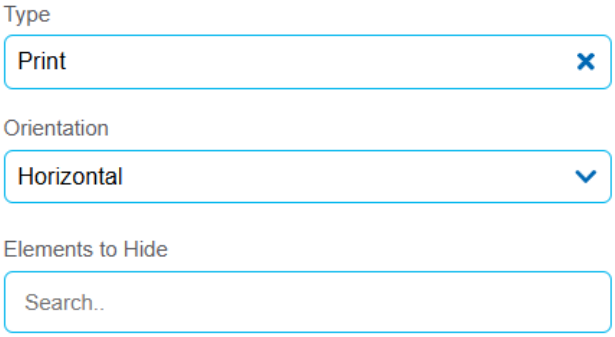
Displays a confirmation modal to logout the user.

Install PWA



Allows to install the application in the browser.

Print



This type of hyperlink associated with an element allows printing the content of the entity or page that is being executed.

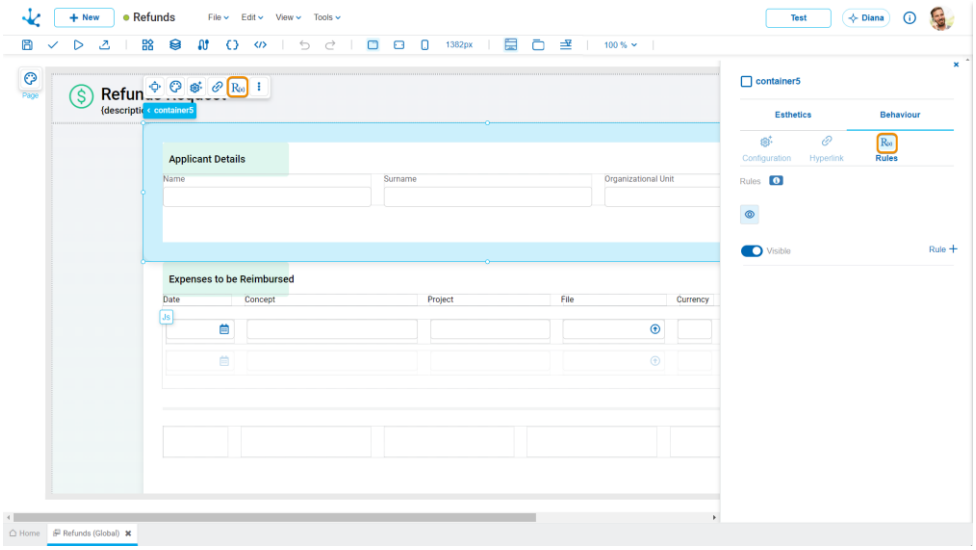
Orientation

Determines the layout of the content to be printed.
Possible Values
- Horizontal
- Vertical

Elements to hide
It allows selecting which elements of the entity or page should not be included in the printout.

Rule Properties

The rule properties panel of an element opens when clicking the icon  of its context menu.



Properties

Rules
Embedded rules on behavior, validation and calculation can be defined, associated with an element, by using the wizard (ctrl + space).
Shows syntax examples for writing the rules.

Visible

Indicates whether the element is visible. If this property is not checked, the element is not displayed in the page.
Visible (default) Not visible
Rule + Opens an edit area where a rule to determine the visibility condition can be defined. If a rule is defined, the icon is displayed with light blue borders.

Operations when defining the rule:

Saves the new or modified rule

Cancels the operation

Operations once the rule is defined:

- Edits the existing rule
- Deletes the rule

Events

Containers allow the use of different events.

Event	Description
onMouseIn()	It is executed when the cursor is positioned on the element.
onMouseOut()	It is executed when the cursor moves out of the element.
onClick()	It is executed when clicking on the element.
onDoubleClick()	It is executed when clicking twice on the element.
onInit()	It is executed before the element is loaded.
afterViewInit()	It is executed after viewing the element.
onViewportEnter()	It is executed when the element is visible.
onViewportLeave()	It is executed when the element is no longer visible.

Iframe

This element is used to show **Deyel** objects or external web pages. For example, it can contain forms, pages, cases, etc. Its position and size must be set to respond to different breakpoints.

The element properties are represented by icons in its context menu, where its operations are also available.

Subtypes

When selecting the option "Iframe" from the icon in the top toolbar, a list opens with the different subtypes of this element, which can be dragged into the modeling area. Each subtype has the element's properties modeled and predefined in a specific way.

- Iframe with Scroll

- Adaptable Iframe

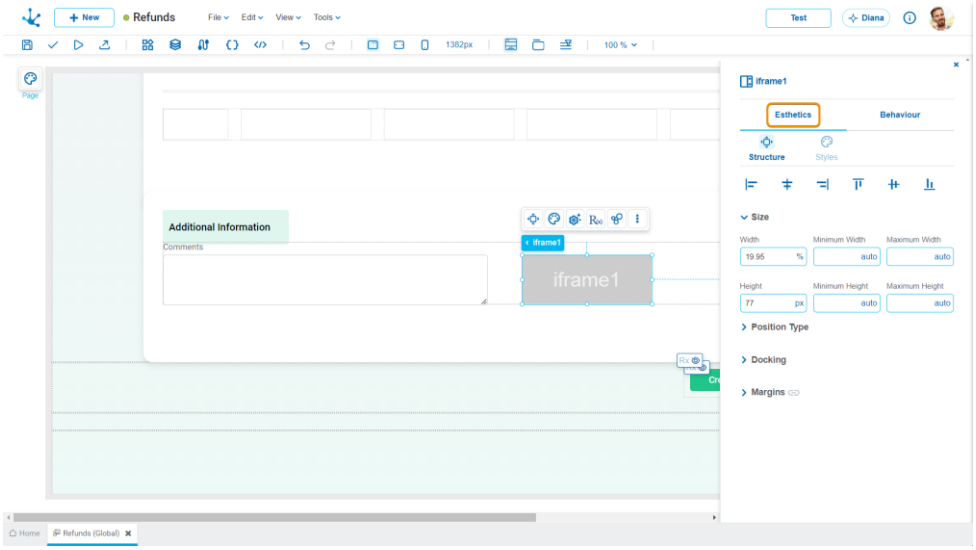
Classification of Properties

Properties are divided into two groups: esthetic properties and behavior properties.

Esthetic Properties

In the esthetic properties panel, the following are grouped:

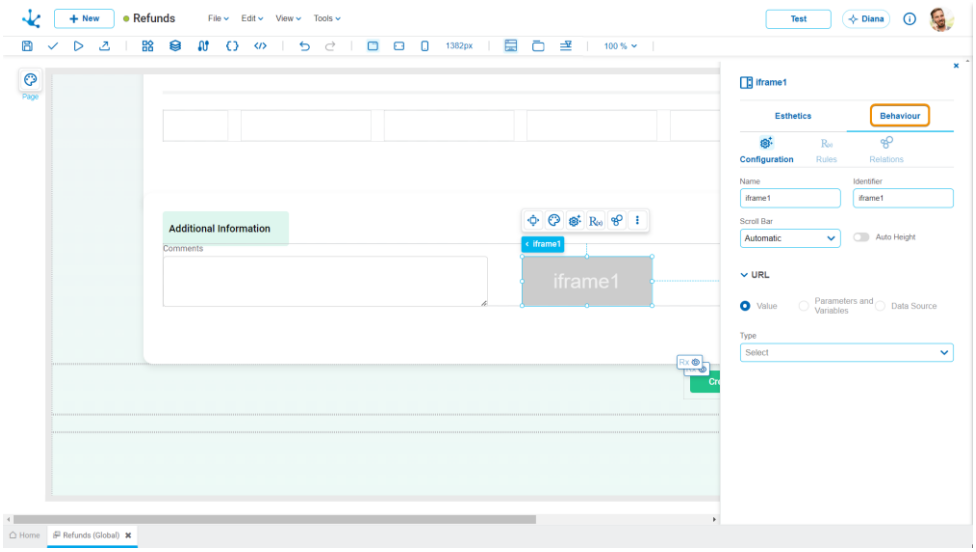
- Structure Properties
- Style Properties



Behavior Properties

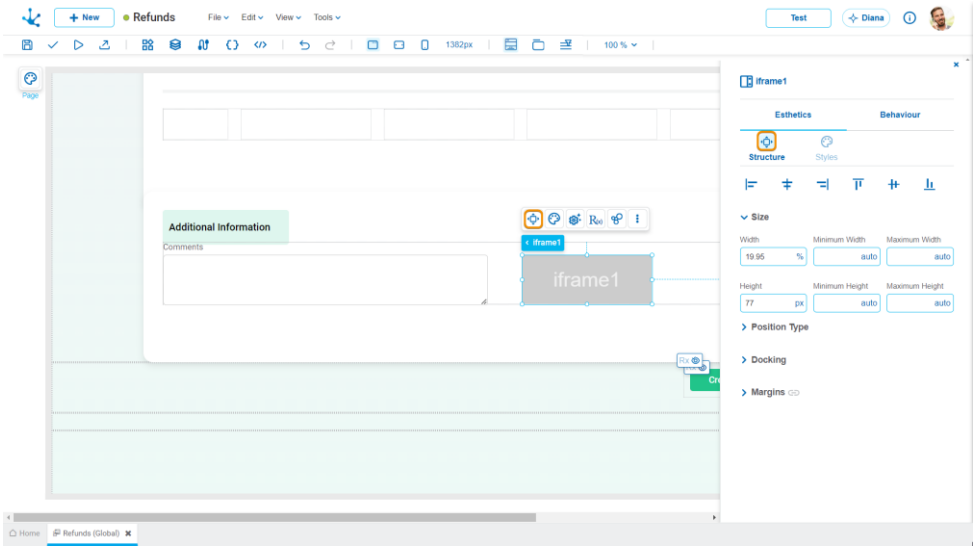
In the behavior properties panel, the following are grouped:

- Configuration Properties
- Rule Properties
- Relation Properties



Structure Properties

The structure properties panel of an element opens when selecting the icon  of its context menu.



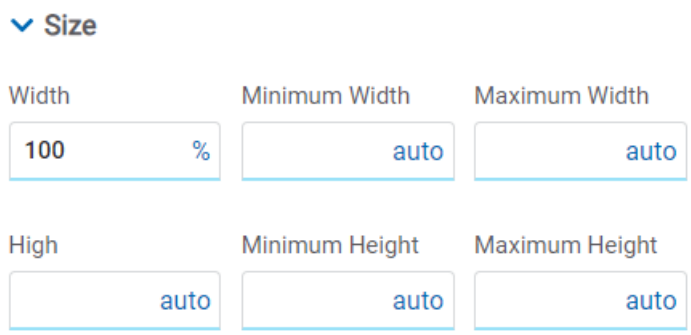
Alignment

Allows to align the element by selecting one of the available icons.

-  Align to the left.
-  Align to horizontal center.
-  Align to the right.
-  Align up.

- Align vertical center.
- Align bottom.

Size

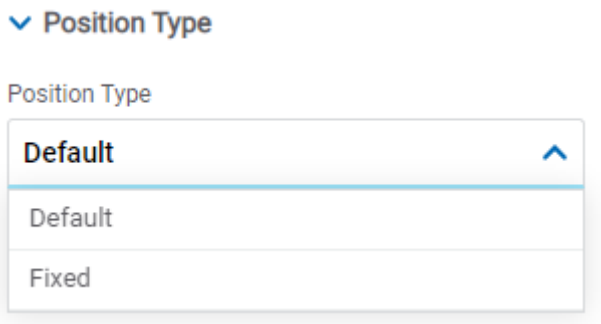


All size properties can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh). Additionally, for Width and Height properties, the "auto" option is added.

> *If selecting percentage (%) for the width and height properties, the size is calculated relative to the top element.*

Position Type

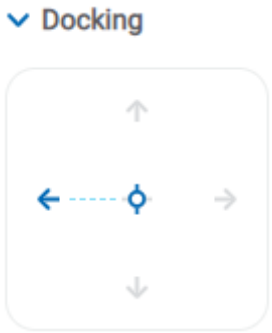
It determines if at the time of execution the element remains fixed on the page or if it moves as the scroll bar moves forward or backward.



Possible Values

- Default: The element has a relative position with respect to the superior element where it was placed (container or section).
- Fixed: Fixes the element to the page so that it is always visible in the same position. It allows to increase the visibility of important content.

Docking



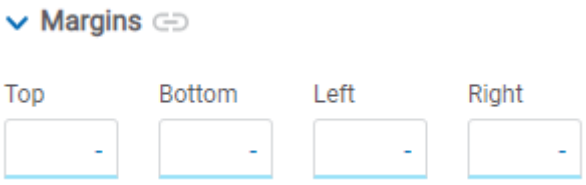
Determines the vertical and horizontal position of elements within the page section or container, when the screen is resized.

When an element is added or moved within the section or container, the modeler automatically docks it to the nearest corners or edges. It can also be configured by clicking on the arrows of the side to be docked to or on the center.

Once the element is docked, its exact position is indicated in the Margins property. The direction of the docking arrows determine the values that are automatically displayed for margins.

The docking position is indicated in the modeling area by dotted lines on the element.

Margins



It allows to define the distance of elements from the borders of their top element. The behavior of margins depends on the docking of the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Top
Distance to the top border of the highest ranking element.

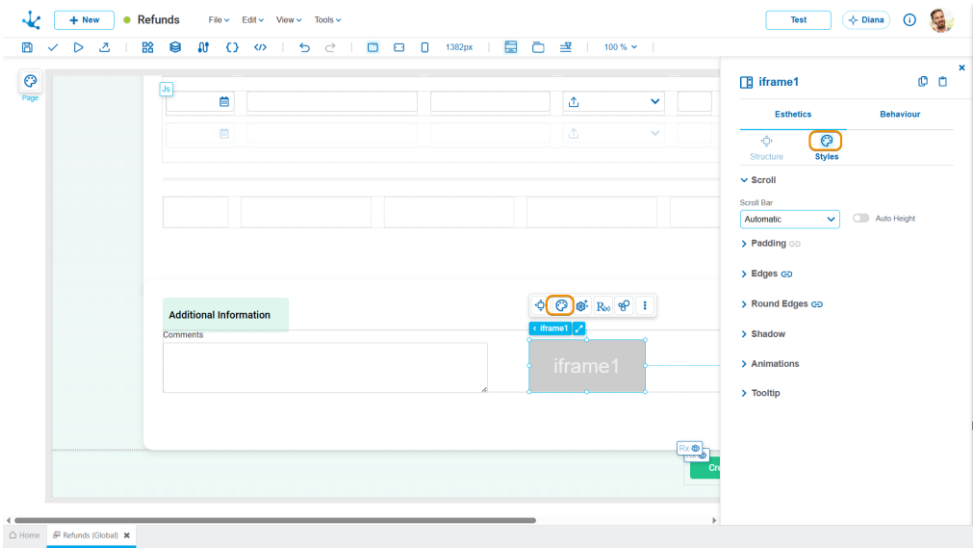
Bottom
Distance to the bottom border of the highest ranking element.

Left
Distance to the left border of the highest ranking element.

Right
Distance to the right border of the highest ranking element.

Allows the value entered in one of the margins to be copied to the other ones automatically.
Allows to indicate different values for each margin.

The style properties panel of an element opens when selecting the icon of its context menu.

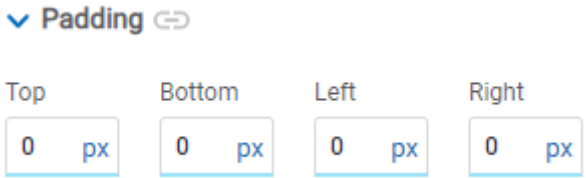


Styles

It allows copying the entire current configuration of the style panel for the selected element (Ctrl+Alt+C).

It allows applying the previously copied configuration to another element of the same type (Ctrl+Alt+V).

Padding

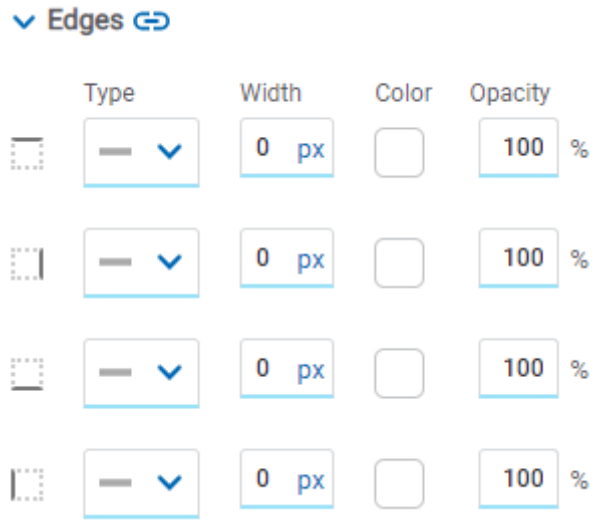


All padding space properties allow to create a space around the element borders (top, bottom, sides) and its content. The padding space configured is inward, while the margin is configured relative to a top element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Allows values entered in one of the paddings to be copied to the other ones automatically.

Allows to indicate different values for each padding.

Edges



It allows to define the style of borders. Each one has its type, width, color and opacity defined, the latter as a percentage.

Allows the value entered in one of the borders to be copied to the other ones automatically.
Allows to indicate different values for each border.

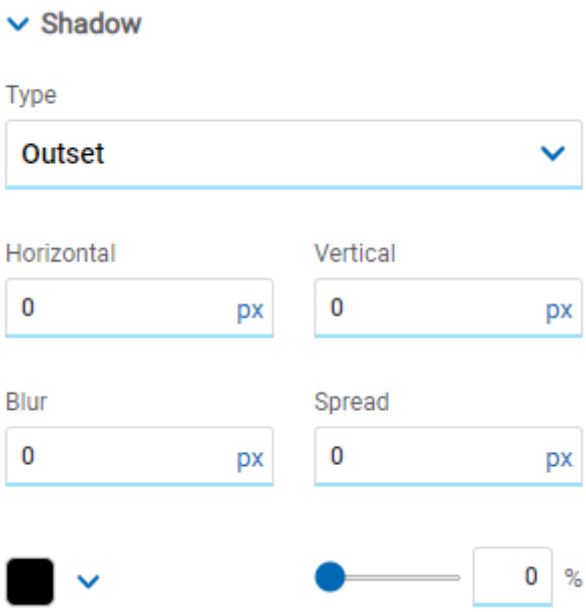
Round Edges



It allows to define the round edges at the corners of elements.
It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Allows the value entered in one of the borders to be copied to the other ones automatically.
Allows to indicate different values for each border.

Shadow



Allows to define a shadow effect around the element.

Type

Possible Values

- Outset
- Inset

Horizontal
Horizontal size of the shadow to the right of the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Vertical
Vertical size of the shadow below the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

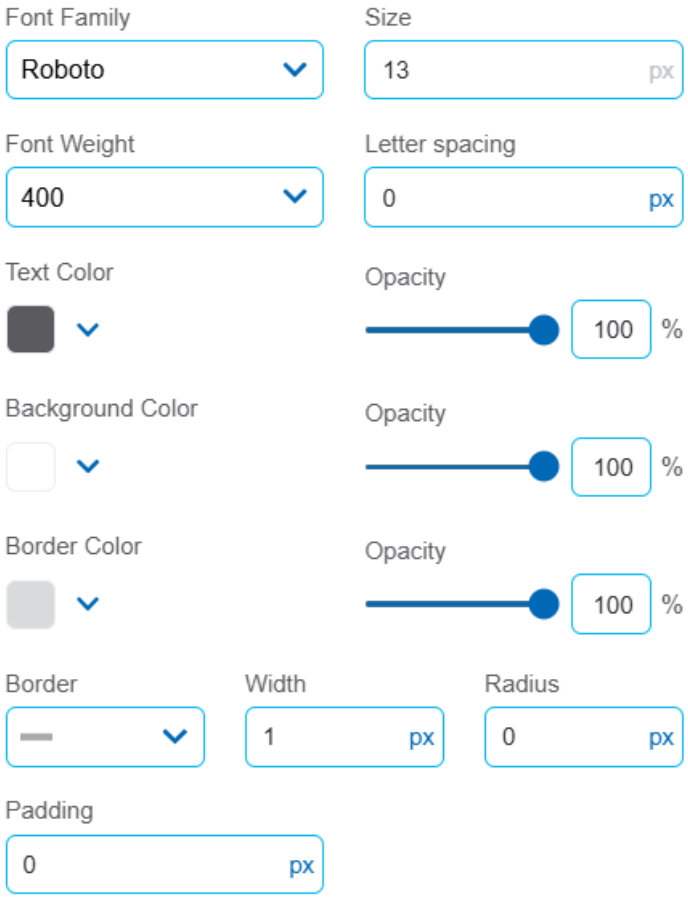
Blur
The greater this value is, the greater and lighter the shadow becomes. If not specified, its value is 0 and the shadow border is darker. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Spread
Positive values will cause the shadow to expand and grow bigger, while negative values will cause the shadow to shrink. If not specified, its value is 0 and the shadow will have the same size as the element.

Color
Defines the color of the shadow.

Transparency
Defines the transparency of the shadow.

Tooltip



Source

It allows customizing the font, size, thickness and letter spacing of the pop-up message.

Colors

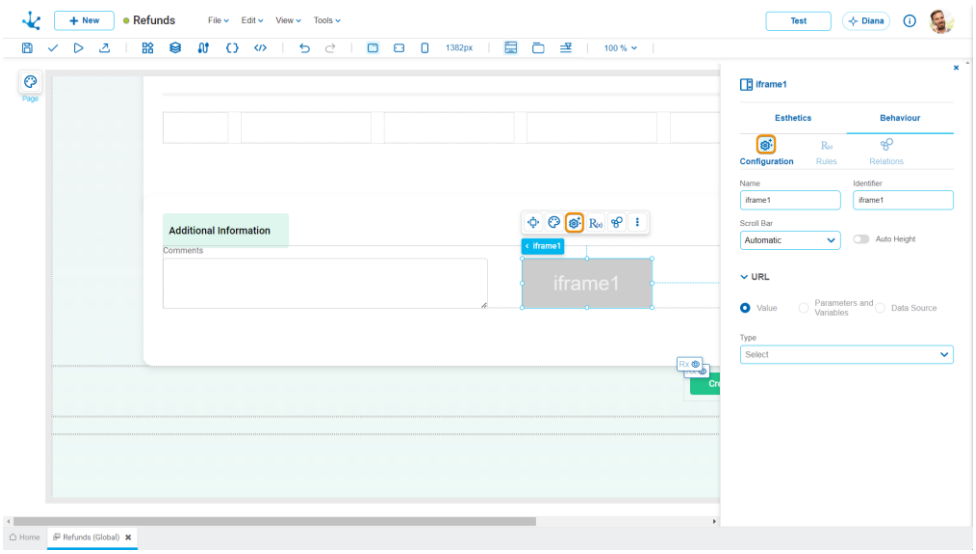
The color and opacity of the text, background and border can be defined. Borders are set uniformly for all sides of the tooltip.

Border

The border type, width, and border radius can be configured.

Configuration Properties

The configuration properties panel of an element opens when clicking the icon  of its context menu.



Properties



Name
Name used to reference the element during modeling.

Identifier
Uniquely identifies the element. It is used in the Javascript SDK.

Tooltip
Pop-up message that appears when hovering over an element, providing additional information.



Text
Content of the message being displayed.

Direction
Position of the message in relation to the element (top, bottom, left, or right).

Scroll



Auto Height

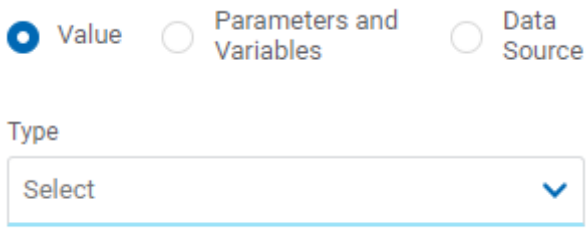
It allows the height of the iframe to adjust to the size of its content if the content exceeds the size of the iframe. If this property is not checked, the different values of the Scroll Bar property can be selected.

Scroll Bar
Indicates whether the scroll on the element is displayed: always, never, or according to the size of the iframe content.

URL

Allows to select the source of the element's content.

Value



- Page
  Page
  It allows selecting the external page that is displayed in the iframe. Once it is selected, its parameters are displayed if it has any.

- Deyel Page
  Deyel Page
  It allows selecting the **Deyel** page displayed in the iframe. Once it is selected, its parameters are displayed if available, or else a parameter can be added.

- Entity
  Entity
  It allows selecting a name for the entity displayed on the iframe. Once selected, its parameters are displayed if available, or else a parameter can be added.

- Form
  Form
  It allows selecting a name for the form displayed on the iframe. Once selected, its parameters are displayed if available, or else a parameter can be added.

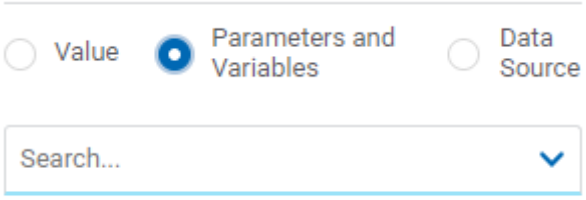
  Operation
  It allows selecting whether to view the creation of an instance or the form grid.

- Process
  Process
  It allows selecting the name of the process that is executed in the iframe. Once selected, its parameters are displayed if available, or else a parameter can be added.

Operation
Indicates that the creation of a case is displayed.

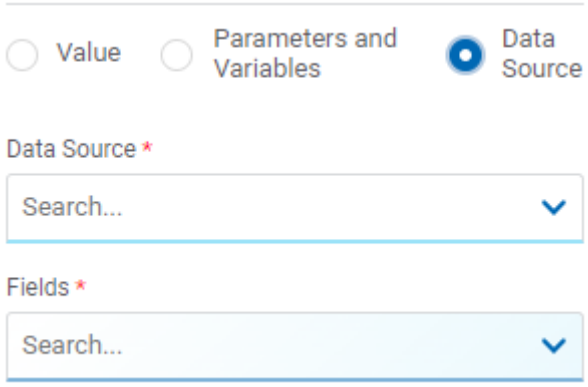
- Link
  Link
  It allows the entry of the link to the page that executes in the iframe.

Parameters and Variables



Parameters and Variables
Allows to select a parameter or a variable from, whose value is displayed in the element.

Data Source

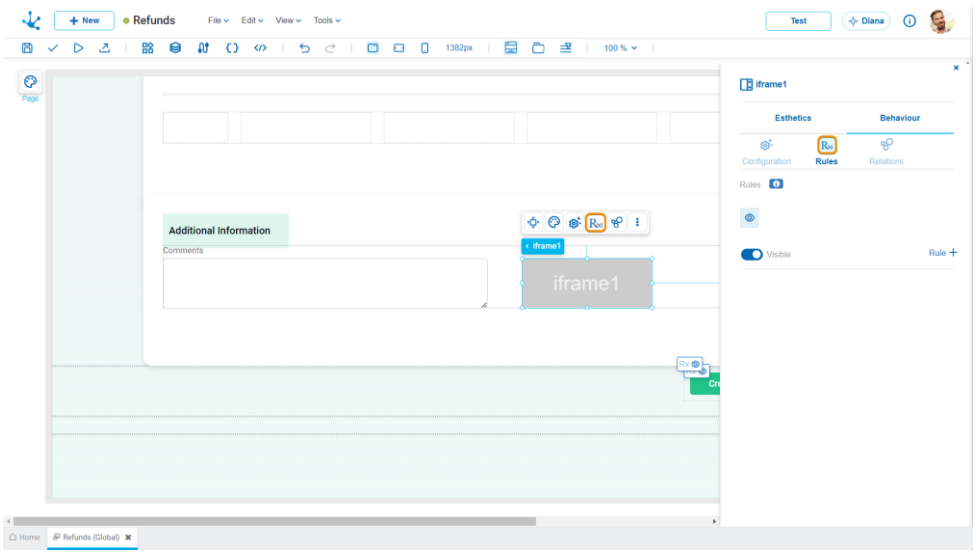


Data Source
It allows selecting a previously defined data source within the object.

Fields
It allows selecting a field from the data retrieved in the chosen data source, whose value is displayed in the element.

Rule Properties

The rule properties panel of an element opens when clicking the icon  of its context menu.



Properties

Rules
Embedded rules on behavior, validation and calculation can be defined, associated with an element, by using the wizard (ctrl + space).
Shows syntax examples for writing the rules.

Visible
Indicates whether the element is visible. If this property is not checked, the element is not displayed in the page.
Visible (default) Not visible
Rule + Opens an edit area where a rule to determine the visibility condition can be defined. If a rule is defined, the icon is displayed with light blue borders.

Operations when defining the rule:

Saves the new or modified rule

Cancels the operation

Operations once the rule is defined:

Edits the existing rule

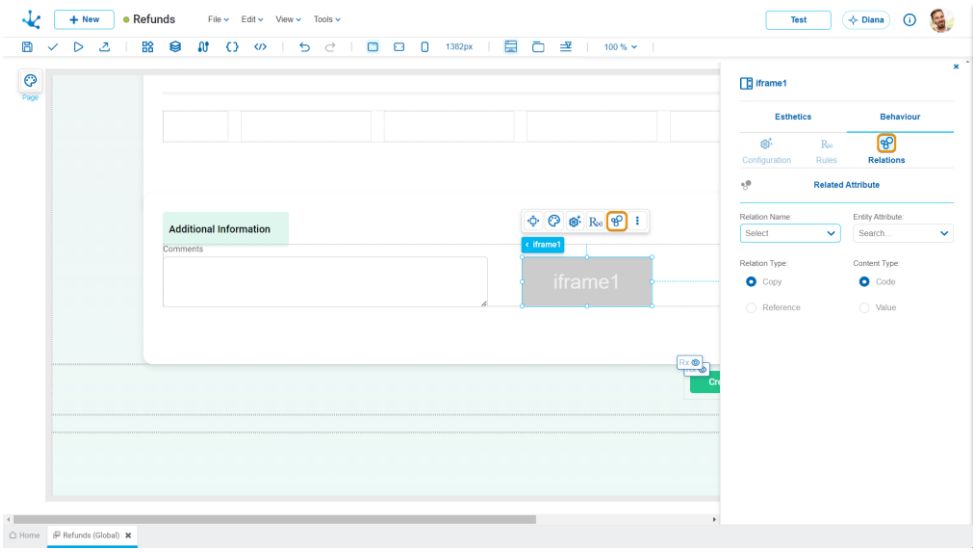
Deletes the rule

Events

Iframes allow the use of an event.

Event	Description
onLoad()	It is executed once all elements of the iframe are loaded.

The relation properties panel of an element opens when clicking the icon  of its context menu.



Related Attribute

A related attribute indicates that the field value is retrieved from the attribute value of another entity. One of the relations defined on the entity must be selected, and then it should be specified which attribute of the related entity it links to.

A field can be modeled as a related attribute and it can also have a relation defined to an entity, a value list, or a rule. This allows the value of the related attribute to be retrieved on this field and the functionality provided by the relation to be used.

Properties

Relation Name
The list of relations defined in the entity is displayed, allowing the selection of one of them.

Entity Attribute
The list of attributes of the selected entity compatible with the field type is displayed. Attributes can have a relation modeled with an entity, a value list, or a rule in the selected entity.

Relation Type
Copy

When this property is checked, once the retrieved value is uploaded, it does not automatically update even if the value of the related attribute changes.

Reference

When this property is checked, the field value is subject to the current value of the related attribute, that is, the updated value is automatically displayed.

Content Type

The element content varies depending on whether the Code or Value property is checked when:

- The entity attribute has a modeled relation.
- The field has the option "Entity" selected in the Values Obtained from property.

Code

When this property is checked, if the relation is modeled with a:

- Related entity, the identifier of the entity is displayed.
- Value list, the value list code is displayed.
- Rule, the code returned by the rule's execution is displayed.

Value

When this property is checked, if the relation is modeled with a:

- Related entity, the short description of the entity is displayed.
- Value list, the value list description is displayed.
- Rule, the description returned by the rule's execution is displayed.

This element is used for data entry.

The element properties are represented by icons in its context menu, where its operations are also available.

Subtypes

When selecting the option "Field" from the icon in the top toolbar, a list opens with the different subtypes of this element, which can be dragged and dropped into the modeling area.

- Text
- Number
- Multiline
- Date and Time
- Options Group
- Toggle
- File

The subtype of a field can be selected in the property Content Format.

File Subtype

When the selected subtype is File, aspects related to its storage must be considered. The location where the attached file is saved is determined by the persistence configuration.

It is possible to attach files to fields both from the computer and from Google Drive. This functionality is automatically available when configuring and publishing the Google Drive adapter, without requiring changes to the modeling.

When executing the entity, a file can be added by clicking on the modeled field.

The following options are available:

- From the computer: allows selecting a local file.
- From Google Drive: a window will open to log in to Google (if necessary) and then select the file from a Google account.

Once selected and confirmed, the file is displayed in the field.

Whenever a file is uploaded, the icon is displayed to the right of the field, sometimes automatically and other times when hovering the cursor over it.

When clicked, the following actions are displayed:

- Preview
- Download
- Delete

When hovering the cursor over the file, the preview option is displayed directly, regardless of the field configuration.

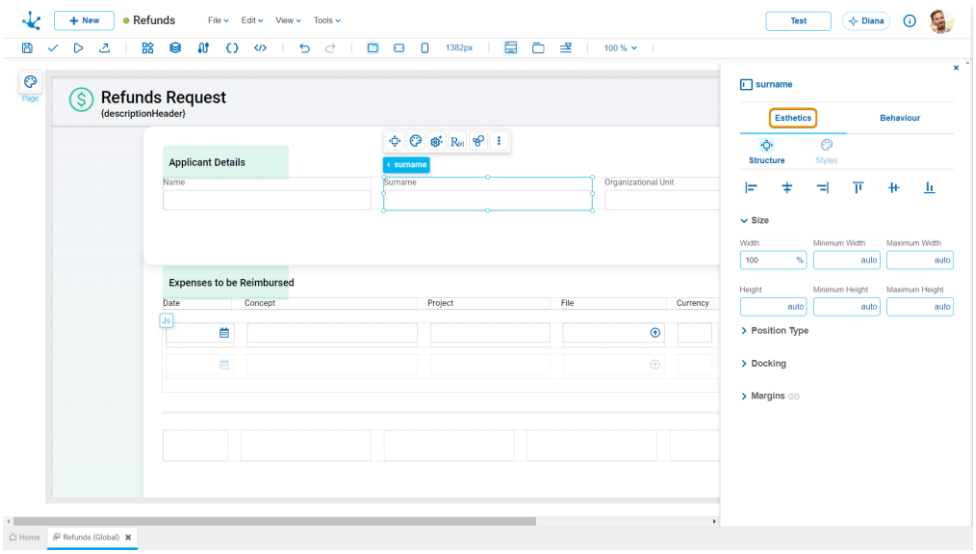
Classification of Properties

Properties are divided into two groups: esthetic properties and behavior properties.

Esthetic Properties

In the esthetic properties panel, the following are grouped:

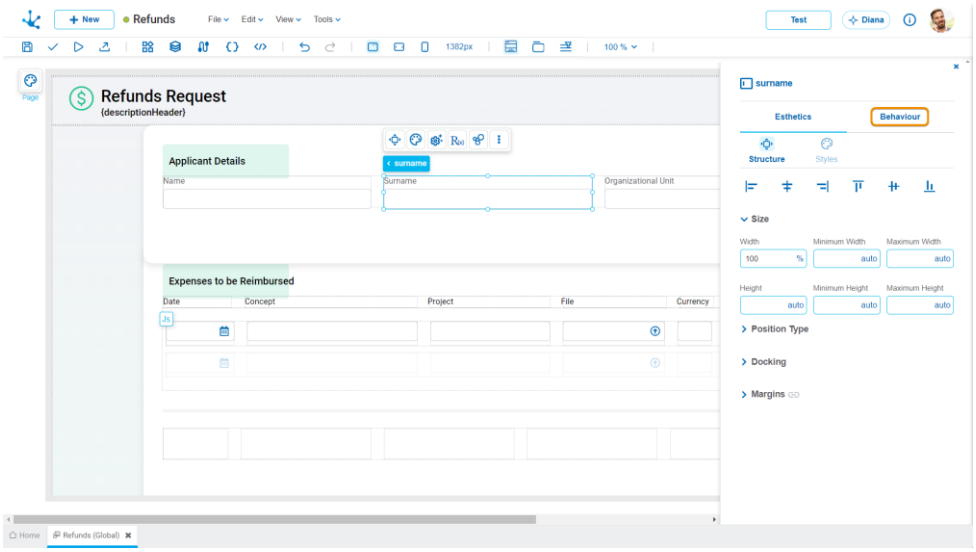
- Structure Properties
- Style Properties



Behavior Properties

In the behavior properties panel, the following are grouped:

- Configuration Properties
- Rule Properties
- Relation Properties



Considerations

When an entity has one or more instances created, certain modifications to its modeling are restricted, as detailed below.

- Delete fields.

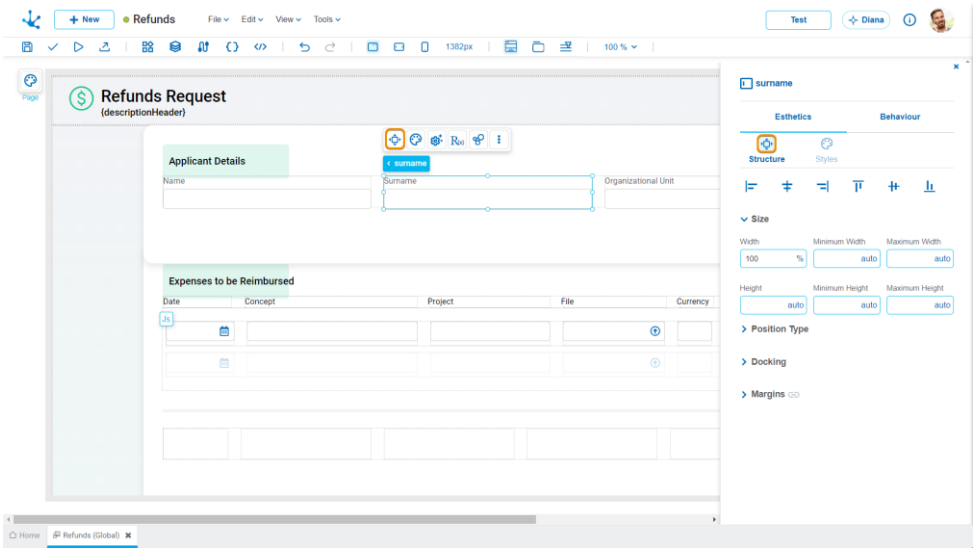
- Delete sections or containers that contain fields.
- Modify the identifier of a field.
- Change the content format of the field between incompatible types.
  - Example: A "Text" field cannot be changed into a "Number" field.
  - Exception: A "Text" field can be changed into a "Multiline" field.
- Modify the Content property with incompatible changes.
  - Cannot be modified in fields with Content Format "Number".
  - Cannot be modified in fields with Content Format "Date and Time".

When the entity has instances created, only the following modifications are allowed:

- Modify the property Content of fields with Content Format "File".
- Modify the property Content between "Integer" and "Decimal".

Structure Properties

The structure properties panel of an element opens when selecting the icon of its context menu.



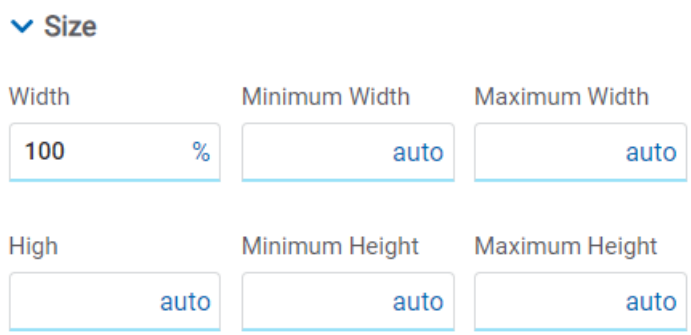
Alignment

Allows to align the element by selecting one of the available icons.

- Align to the left.
- Align to horizontal center.

- Align to the right.
- Align up.
- Align vertical center.
- Align bottom.

Size

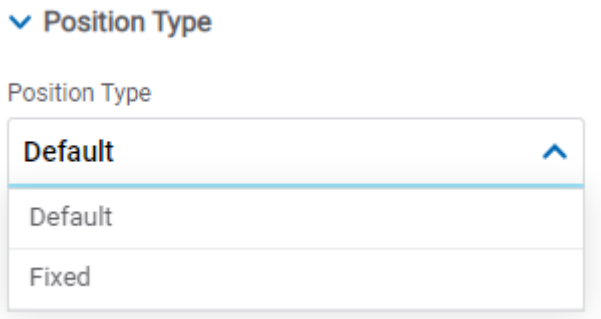


All size properties can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh). Additionally, for Width and Height properties, the "auto" option is added.

> *If selecting percentage (%) for the width and height properties, the size is calculated relative to the top element.*

Position Type

It determines if at the time of execution the element remains fixed on the page or if it moves as the scroll bar moves forward or backward.



Possible Values

- Default: The element has a relative position with respect to the superior element where it was placed (container or section).
- Fixed: Fixes the element to the page so that it is always visible in the same position. It allows to increase the visibility of important content.

Docking



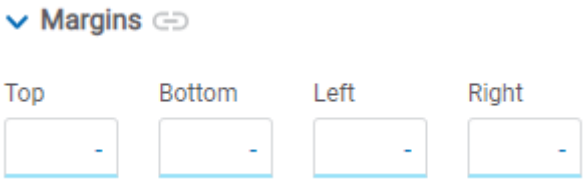
Determines the vertical and horizontal position of elements within the page section or container, when the screen is resized.

When an element is added or moved within the section or container, the modeler automatically docks it to the nearest corners or edges. It can also be configured by clicking on the arrows of the side to be docked to or on the center.

Once the element is docked, its exact position is indicated in the Margins property. The direction of the docking arrows determine the values that are automatically displayed for margins.

The docking position is indicated in the modeling area by dotted lines on the element.

Margins



It allows to define the distance of elements from the borders of their top element. The behavior of margins depends on the docking of the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Top
Distance to the top border of the highest ranking element.

Bottom
Distance to the bottom border of the highest ranking element.

Left
Distance to the left border of the highest ranking element.

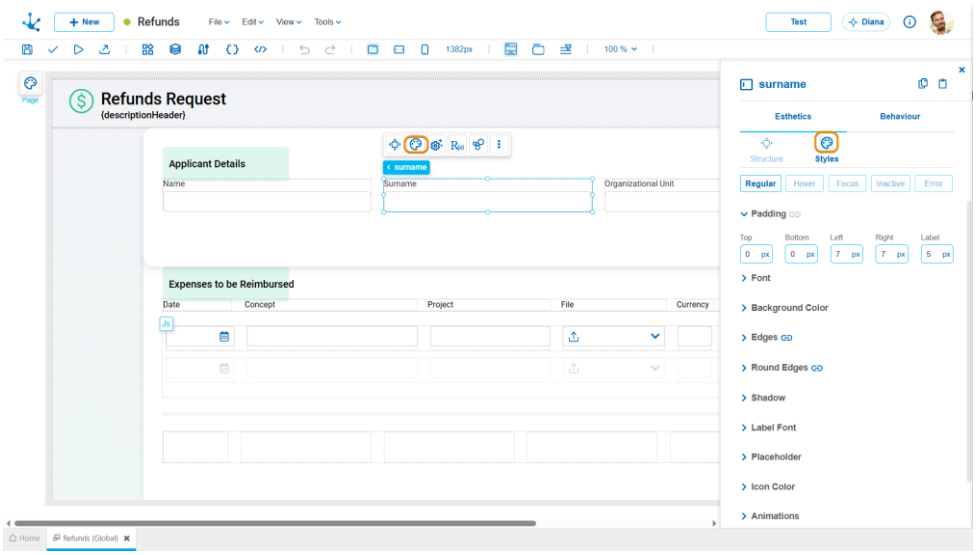
Right
Distance to the right border of the highest ranking element.

Allows the value entered in one of the margins to be copied to the other ones automatically.

Allows to indicate different values for each margin.

Style Properties

The style properties panel of an element opens when selecting the icon of its context menu.



This type of element may take different states and for each of them different values for its properties may be modeled.



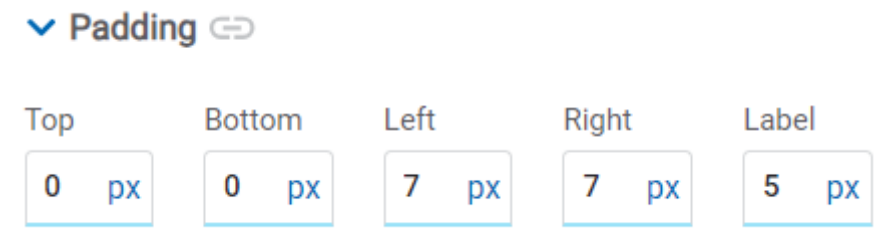
- Regular: the mouse pointer is not over the element.
- Hover: the mouse pointer is over the element.
- Focus: the element is pressed.
- Inactive: The element is not active.
- Error: The element is in error.

Styles

It allows copying the entire current configuration of the style panel for the selected element (Ctrl+Alt+C).

It allows applying the previously copied configuration to another element of the same type (Ctrl+Alt+V).

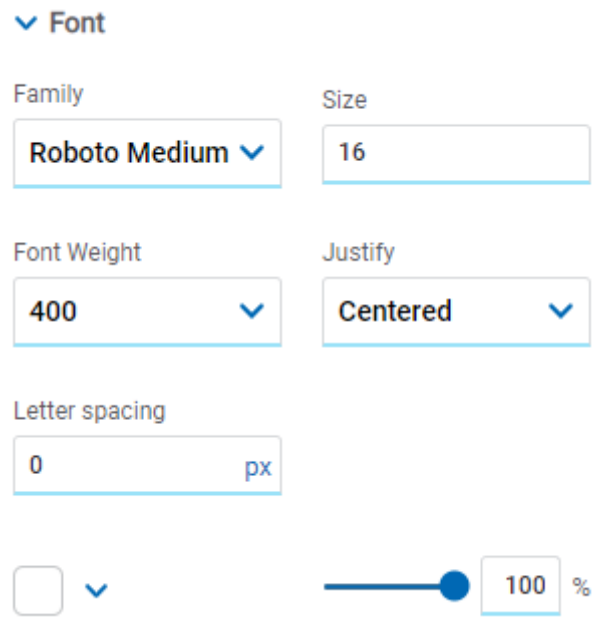
Padding



All padding space properties allow to create a space around the element borders (top, bottom, sides) and its content. The padding space configured is inward, while the margin is configured relative to a top element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Allows values entered in one of the paddings to be copied to the other ones automatically.
Allows to indicate different values for each padding.

Font

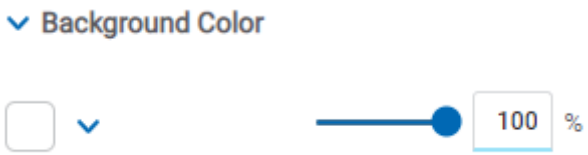


It allows to define the text style.

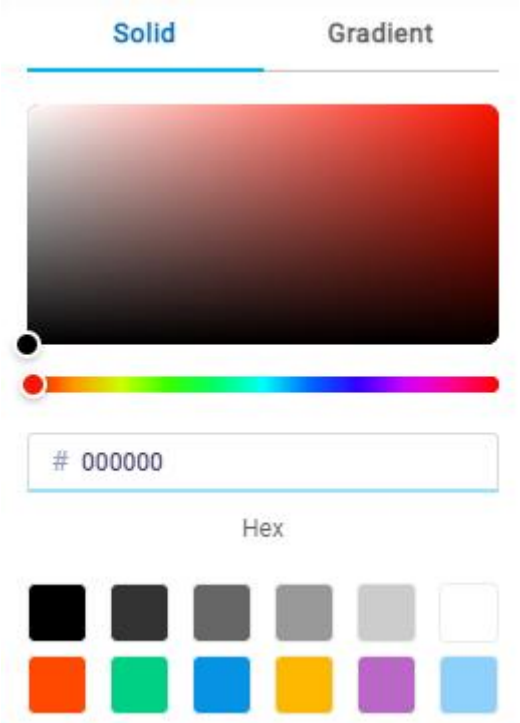
Background Color

It allows to model a background color for the element, it can be solid or gradient and different properties are defined for each one.

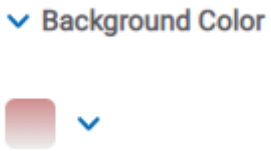
Solid



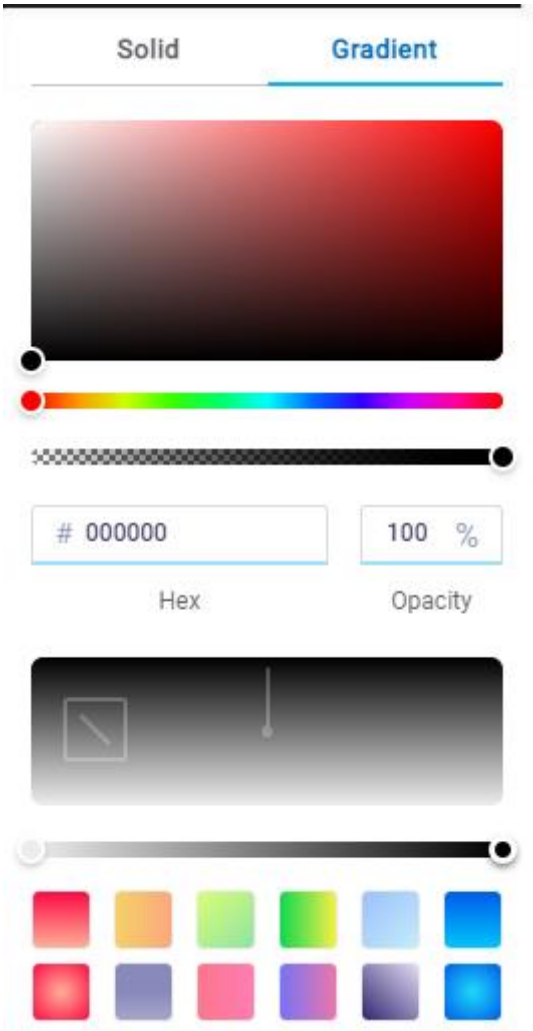
This option enables not only to select the color from a palette but also to define the degree of transparency.



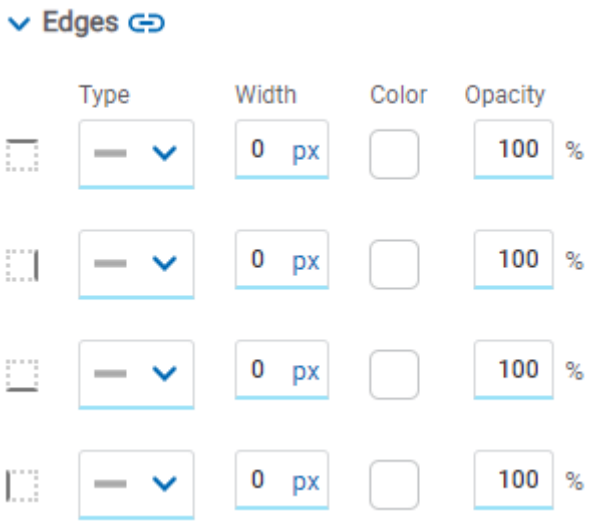
Gradient



This option enables to select not only the color from a palette but also to define its opacity and intensity according to the selected angle.



Edges



It allows to define the style of borders. Each one has its type, width, color and opacity defined, the latter as a percentage.

Allows the value entered in one of the borders to be copied to the other ones automatically.
Allows to indicate different values for each border.

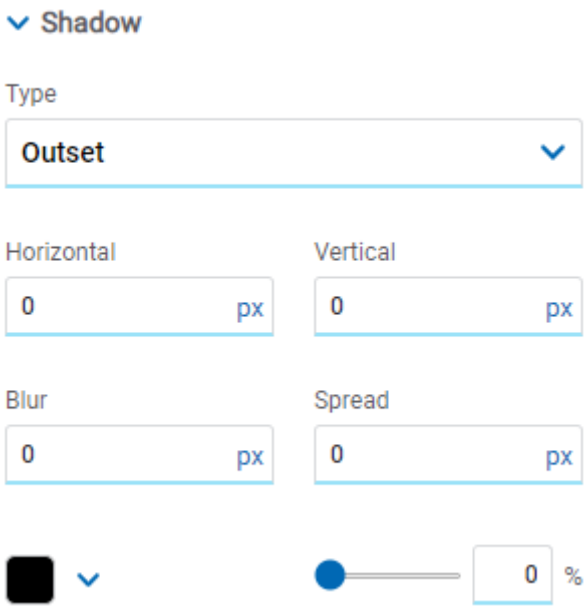
Round Edges



It allows to define the round edges at the corners of elements.
It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Allows the value entered in one of the borders to be copied to the other ones automatically.
Allows to indicate different values for each border.

Shadow



Allows to define a shadow effect around the element.

Type

Possible Values

- Outset
- Inset

Horizontal
Horizontal size of the shadow to the right of the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Vertical
Vertical size of the shadow below the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

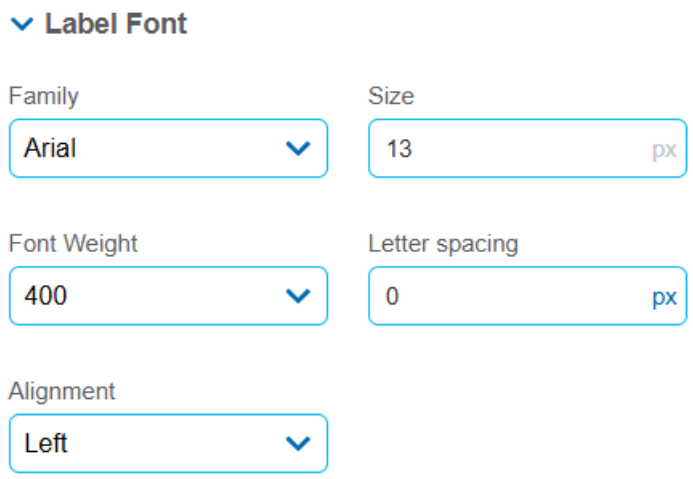
Blur
The greater this value is, the greater and lighter the shadow becomes. If not specified, its value is 0 and the shadow border is darker. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Spread
Positive values will cause the shadow to expand and grow bigger, while negative values will cause the shadow to shrink. If not specified, its value is 0 and the shadow will have the same size as the element.

Color
Defines the color of the shadow.

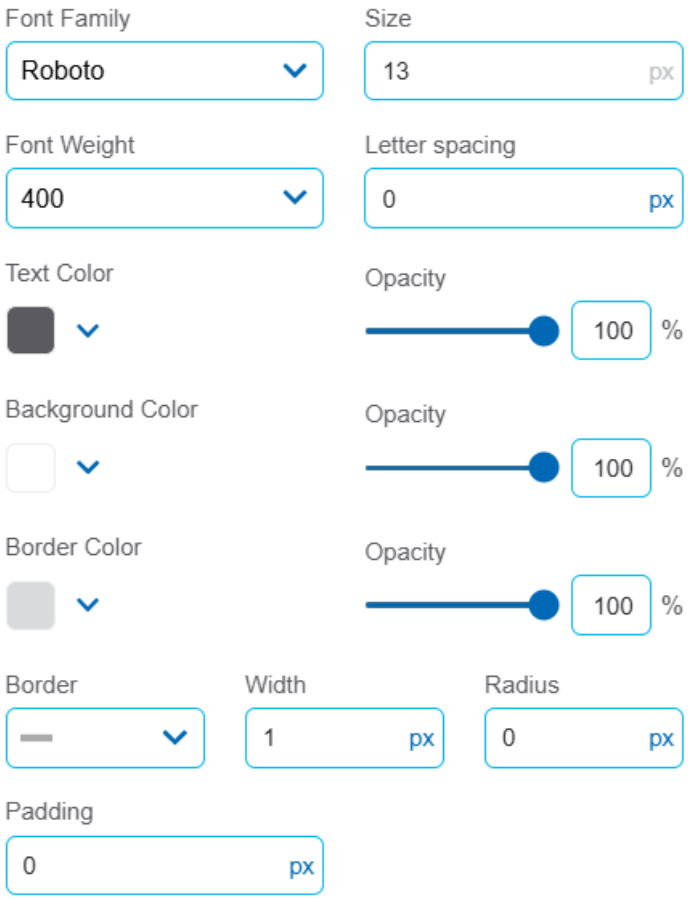
Transparency
Defines the transparency of the shadow.

Label Font



It allows to define the label style.

Tooltip



Source

It allows customizing the font, size, thickness and letter spacing of the pop-up message.

Colors

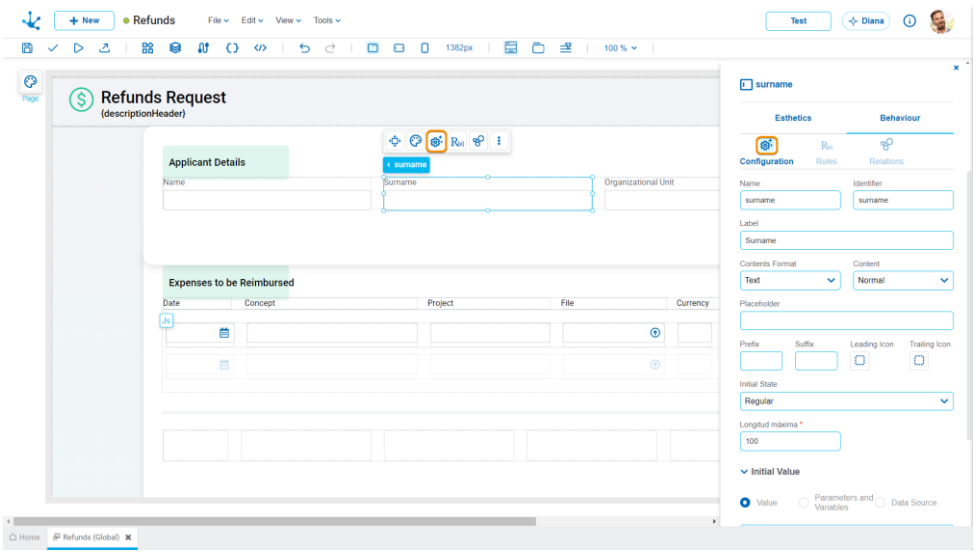
The color and opacity of the text, background and border can be defined. Borders are set uniformly for all sides of the tooltip.

Border

The border type, width, and border radius can be configured.

Configuration Properties

The configuration properties panel of an element opens when clicking the icon  of its context menu.



Name

Name assigned to reference a field when modeling, allowing the field to be uniquely identified within the entity. Used in rule wizards to refer to the field within the conditions. It generates automatically from the Label property, can be modified by the user and does not allow spaces or special characters.

Identifier

It is the name assigned to reference a field in the programming code, used to refer to the field within Java code in the "Execution Code" tab of advanced rules and the JavaScript code in the "Page Code" tab of the entity modeler. It allows to uniquely identify the field within the modeled entity. It can be modified by the user, as long as no data has been loaded into the entity, and it does not allow spaces or special characters.

Tooltip

Pop-up message that appears when hovering over an element, providing additional information.



Text

Content of the message being displayed.

Direction

Position of the message in relation to the element (top, bottom, left, or right).

Label

It allows entering the text that is displayed on the field. Supports white space.

Content Format

It allows modifying the selected field type.

Possible Values

- Text
- Number
- Multiline

- Date and Time
- File

Content

Options available for this property depend on the value of the property Content Format. For each possible value, the options are different:

- Text: the options are "Normal, "Password", "Phone Number" and "Mail".
- Number: the options are "Decimal" and "Integer".
- Date and Time: the options are "Date and Time", "Local Date and Time", "Time", "Local Time", "Date" and "Local Date".
- File: the options for images are "*", "jpeg", "png", "gif", "svg+xml", "bmp", "webp", "tiff" and "x-icon". The options for audio are "*", "mpeg", "wav", "ogg", "midi", "aac" and "x-ms-wma". Video options are "*", "mp4", "quicktime", "webm", "x-msvideo", "x-flv", "3gpp". Application options are "*", "pdf", "msword", "vnd.ms-excel", "vnd.ms-powerpoint", "zip", "x-rar-compressed", "x-tar", "x-gzip", "x-bzip2", "jason"and "xml". The options for text are "*", "plain", "csv" and "xml".

If this option is selected, an additional property is enabled.

Display Mode

It allows modifying the way the file uploaded by the user is presented.

Possible Values

- File Name: Shows only the file name when the user uploads it, without an additional preview.
- Previewer: Presents a preview of the uploaded file, providing the user with a more user-friendly and interactive view.

Placeholder

It allows guiding the user about the content when entering the element. The placeholder text is displayed within the element.

Picker

It allows defining an interval picker that helps the user to increase or decrease the numeric field value according to a defined interval. Only available if the Field Type property is of type "Number".

Possible Values

- Align to the left: The picker is displayed on the left side of the numeric field.
- Align to the right: The picker is displayed on the right side of the numeric field.
- Hide: The picker is not displayed in the field; the arrow keys must be used to increase or decrease the value.

Interval

It allows defining the value that is increased or decreased each time the user interacts with the picker. For example, if the interval value is set to 5, clicking the increment icon increases the numeric field value by 5 units. Similarly, clicking the decrement icon decreases the field value by 5 units.

Prefix

It allows incorporating a text or symbol that is displayed on the left side of the field, before the entered value. It is used to provide additional context immediately, making it easier to understand the purpose or format of the field. For example, the $ sign can be set for monetary fields, indicating that a value in a specific currency is expected.

Suffix

It allows incorporating a text or symbol that is displayed on the right side of the field, immediately after the entered value. It is used to complement the information provided, offering details that indicate the context or units of the value entered. The meaning of the value can be specified by including

units of measurement, percentages, or frequencies—for example, "kg" in a field intended to record weight.

Leading Icon

It allows incorporating an icon on the left side of the field, before the value that is entered or displayed. This leading icon can graphically represent the expected data type, making it easier to quickly identify the purpose of the field. For example, a phone icon can be used to indicate that a contact number should be entered.

Trailing Icon

It allows incorporating an icon on the right side of the field, immediately after the value that is entered or displayed. This trailing icon is intended to complement the meaning of the field through a visual element, making it easier to understand the context in which it is located. For example, a "%" icon can be used to indicate that a percentage value should be entered.

Initial State

Defines the visual and functional state of the field when the entity is loaded for the first time. This setting allows determining whether a field is enabled and ready to receive interaction or, on the contrary, is disabled, waiting for an action or precondition to be activated.

Possible Values

- Regular: The field is enabled by default. Allows the user to interact with the field immediately upon loading the entity.
- Inactive: the field is shown as disabled and does not allow interaction.

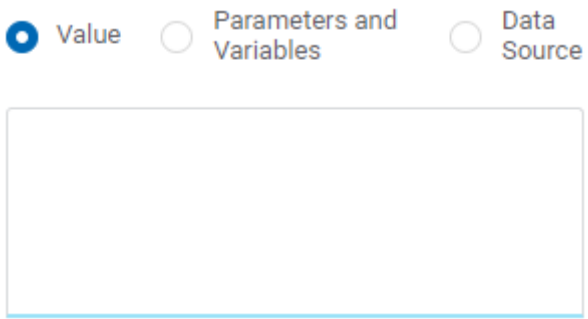
Maximum Length

It allows defining the maximum number of characters that can be entered in a text field.

Initial Value

Allows to select the source of the element's content.

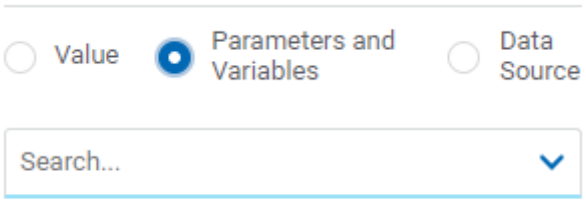
Value



Value

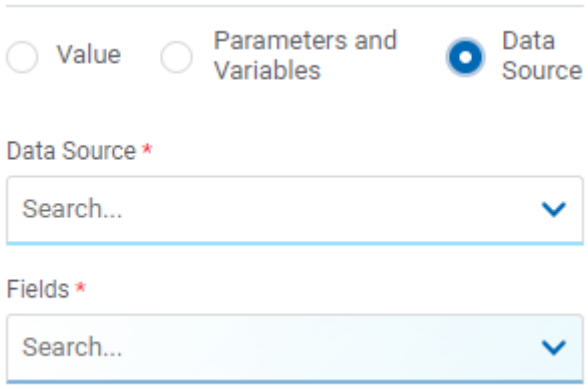
Allows to enter a text that is displayed in the element.

Parameters and Variables



Parameters and Variables
Allows to select a parameter or a variable from, whose value is displayed in the element.

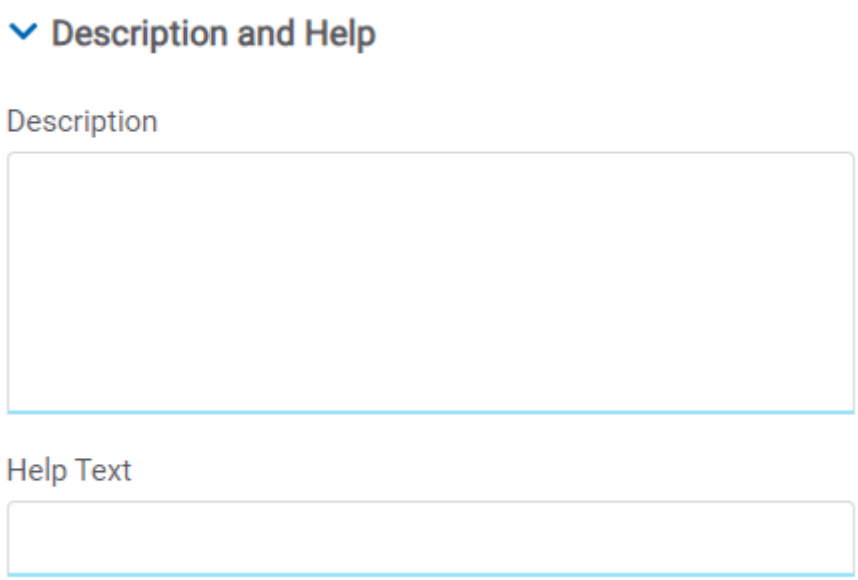
Data Source



Data Source
It allows selecting a previously defined data source within the object.

Fields
It allows selecting a field from the data retrieved in the chosen data source, whose value is displayed in the element.

Description and Help

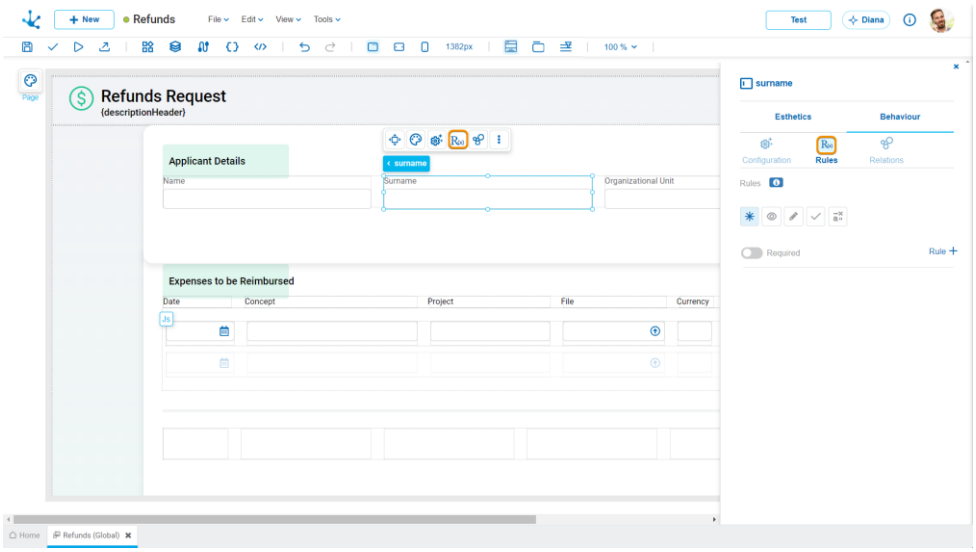


Description
Text that defines the field and optionally its content.

Help Text
This function is to guide the user about the content to be entered in the field. The text entered as help is displayed when the user hovers the cursor over the field.

Rule Properties

The rule properties panel of an element is opened when clicking the icon Rx of the context menu.



Properties

Rules

Embedded rules on behavior, validation and calculation can be defined, associated with an element, by using the wizard (ctrl + space).

Shows syntax examples for writing the rules.

Required

Indicates whether the element is required on the page.

Required Not required (default)

Rule + Opens an edit area where a rule to determine the required condition can be defined. If a rule is defined, the icon is displayed with light blue borders.

Visible

Indicates whether the element is visible. If this property is not checked, the element is not displayed in the page.

Visible (default) Not visible

Rule + Opens an edit area where a rule to determine the visibility condition can be defined. If a rule is defined, the icon is displayed with light blue borders.

Editable

Indicates if the element is editable. If this property is not checked, the user cannot enter or modify values in the element.

Editable (default) Not editable

Rule + Opens an edit area where a rule to determine the editability condition can be defined. If a rule is defined, the icon is displayed with light blue borders.

Validation

Rule + Opens an edit area where you can define the condition that determines if the element value is correct or not. It is possible to define more than one rule. If rules are defined, the icon is displayed with light blue borders.

Calculation

Rule + Opens an edit area where you can define the expression to be executed to calculate the element value. If a rule is defined, the icon is displayed with light blue borders.

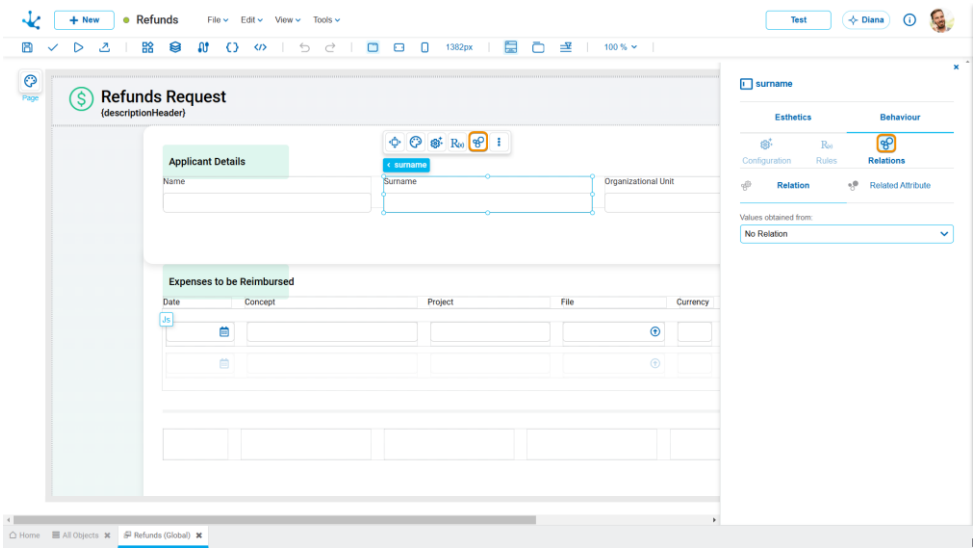
Operations when defining the rule:

- Saves the new or modified rule
- Cancels the operation

Operations once the rule is defined:

- Edits the existing rule
- Deletes the rule

The relation properties panel of an element opens when clicking the icon of its context menu.



Properties are divided into two groups, relation properties and related attribute properties.

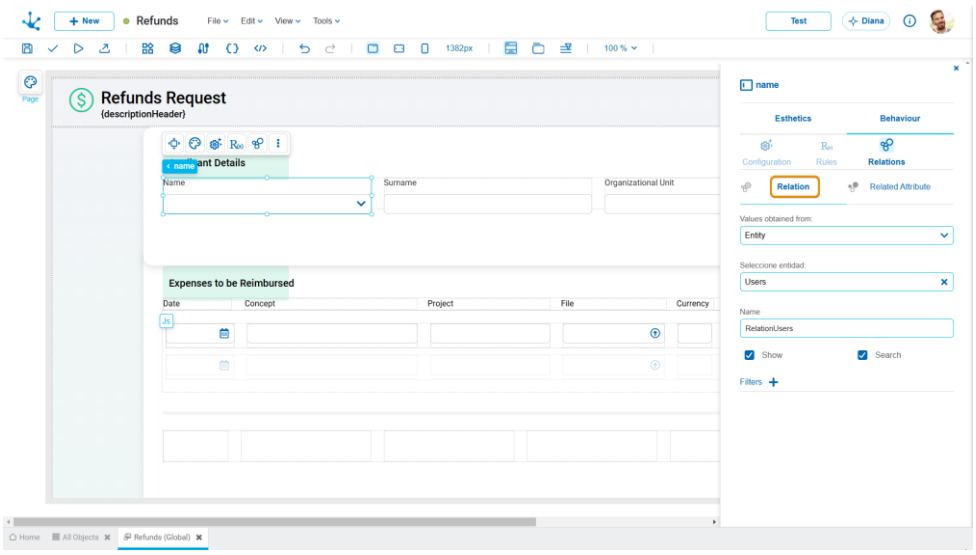
Relation

A relation allows obtaining values from different sources for the element being modeled. Within the "Relation" tab, the property Values obtained from is defined, along with the specific settings for each type of source. If the element has no relation, the default option “No Relation” should be modeled.

Possible Values

- No Relation
- Entity
- Value List
- Rule

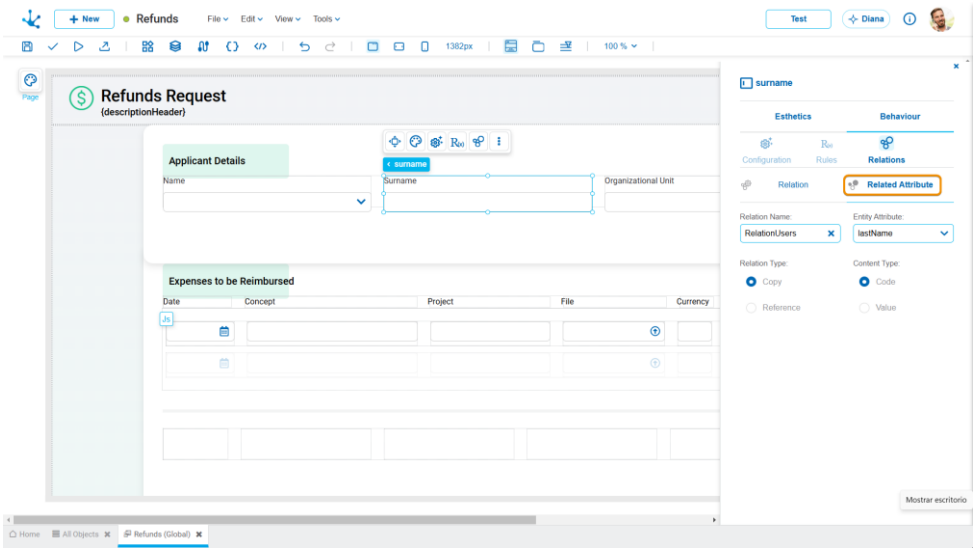
The relation is only valid for text-type fields.



Related Attribute

A related attribute indicates that the field value is retrieved from the attribute value of another entity. One of the relations defined on the entity must be selected, and then it should be specified which attribute of the related entity it links to.

A field can be modeled as a related attribute and it can also have a relation defined to an entity, a value list, or a rule. This allows the value of the related attribute to be retrieved on this field and the functionality provided by the relation to be used.



Properties

Relation Name

The list of relations defined in the entity is displayed, allowing the selection of one of them.

Entity Attribute

The list of attributes of the selected entity compatible with the field type is displayed. Fields can have a relation modeled with an entity, a value list, or a rule in the selected entity.

Relation Type

Copy

When this property is checked, once the retrieved value is uploaded, it does not automatically update even if the value of the related attribute changes.

Reference

When this property is checked, the field value is subject to the current value of the related attribute, that is, the updated value is automatically displayed.

Content Type

The element content varies depending on whether the Code or Value property is checked when:

- The entity attribute has a modeled relation.
- The field has the option "Entity" selected in the Values Obtained from property.

Code

When this property is checked, if the relation is modeled with a:

- Related entity, the identifier of the entity is displayed.
- Value list, the value list code is displayed.
- Rule, the code returned by the rule's execution is displayed.

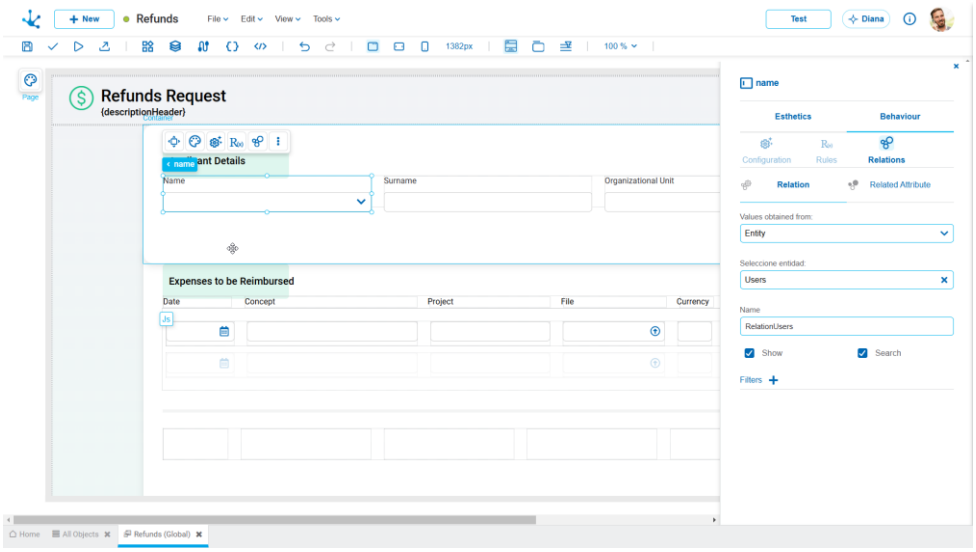
Value

When this property is checked, if the relation is modeled with a:

- Related entity, the short description of the entity is displayed.
- Value list, the value list description is displayed.
- Rule, the description returned by the rule's execution is displayed.

Entities

When relating a field to an entity, the field can take a value that is obtained from any instance of the entity, specifically from the content modeled in its short description.



To define a relation with an entity, first select the entity to establish the relation with, and then complete a set of properties.

Properties

Name
Name of the relation between both entities; it is a modeling-oriented property. It does not allow blanks and must be unique for each modeled entity.

Permissions
The relation permissions define which functionalities are enabled for the field.

Show
Check this property to allow the user of the entity to display the instance of the entity with which the relation is established.

Search
Check this property to allow the entity user to access the results grid and search for the instance of the entity with which the relation is established.

Filter
It allows for narrowing the search results on the related entity.
This filter is applied both in the field's autocomplete and in the show through the magnifying glass when hovering the mouse over the field.
To create a filter, click on the icon + and a panel opens to complete the following properties:

Filtered Attribute
It allows the selection of an attribute of the related entity.

Condition
It allows the selection of a condition as part of the filter.

Type
The possible values to select are "Value" and "Field".

Value
It allows the input of fixed values.

Field
It allows the selection of a page field. It must be considered that its content depends on the Content property of the related attribute

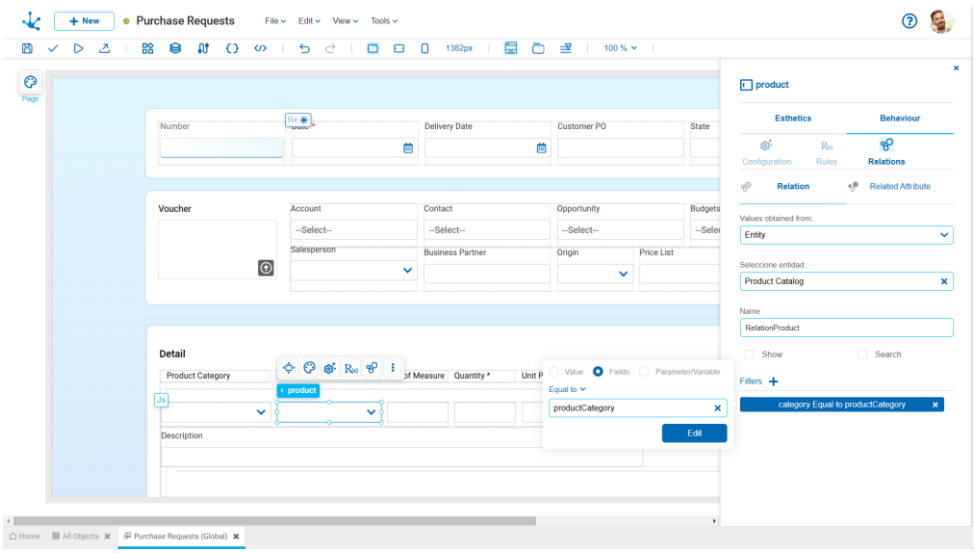
Example of Filtered Entities

In this example, it is described how to model an entity that includes a field related to another entity, and how to apply a filter on that field based on the value of another field in the related entity, meeting a specific condition.

1. Modeling the entity to be related
A "Product Catalog" entity is modeled with the following fields.
- Code: Unique product identifier.
- Description: Product detail or name.
- Category: Product classification (e.g. vehicles, real estate, household appliances).

2. Entity modeling
On the "Purchase Request" entity, the following fields are included.
- Product Category: related to a value list.
- Product: related to the "Product Catalog" entity.

3. Filter configuration on the "Product" field
   - Modeling the relation with the "Product Catalog" entity.
   - Add a filter in order to define that the "Category" field of the "Product Catalog" entity matches the value selected in the "Product Category" field of the "Purchase Order" entity.

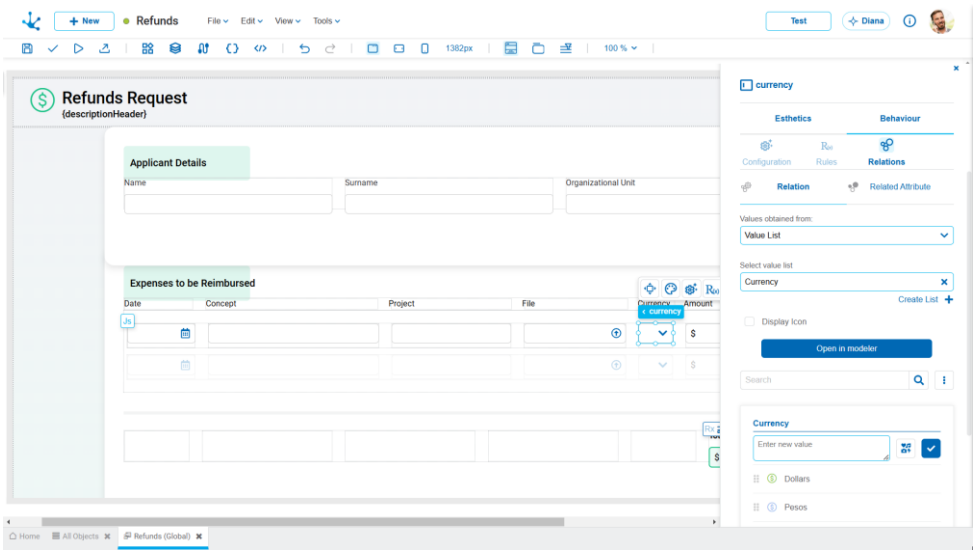


When executing the entity, if a category is selected, in the "Product" field will only display catalog values corresponding to that category.

Value List

When a value list relates to a field, a set of options organized under a specific criterion is defined. This allows the application user to display the possible values that the field can take.

From the modeler, new lists can be created, edited, and linked to entity fields.



To create a new list of values, you must click on the icon +.
A panel opens where the list name is entered, the application it belongs to is selected, and if it is specified to be sorted alphabetically, the descending order can be selected. Clicking the "Accept" button creates the value list, subsequently allowing the entry of new values.

The value list can be used in any other field of the same or another entity.

Property

Select value list
It allows selecting a specific list from the set of existing lists with the option to filter the set by entering text into the search field located above the list.

Other Elements

Open in modeler
It allows using the value lists modeler for the definition, instead of the field properties panel.

 Allows to filter values from the list based on the characters entered. If a list is very long it helps users to easily visualize the desired values.

 Enables an option that allows to add the internal code to the list values.

Operations on Values

-  Allows adding each entered value to the list of values.
-  It is displayed if the list has the Icons property modeled. It allows to associate icons to the list values.

- Double click: Allows to modify a value in the list.
- Move: Allows to change the position of a value within the list by dragging the value with the mouse.

Hovering the cursor over each of the values entered, a set of icons is displayed and this allows to perform different operations.

-  Allows to delete a value from the list of values. Once deleted, it is displayed in gray and crossed out.
-  Allows to restore a previously deleted value.

Display the Selected Line

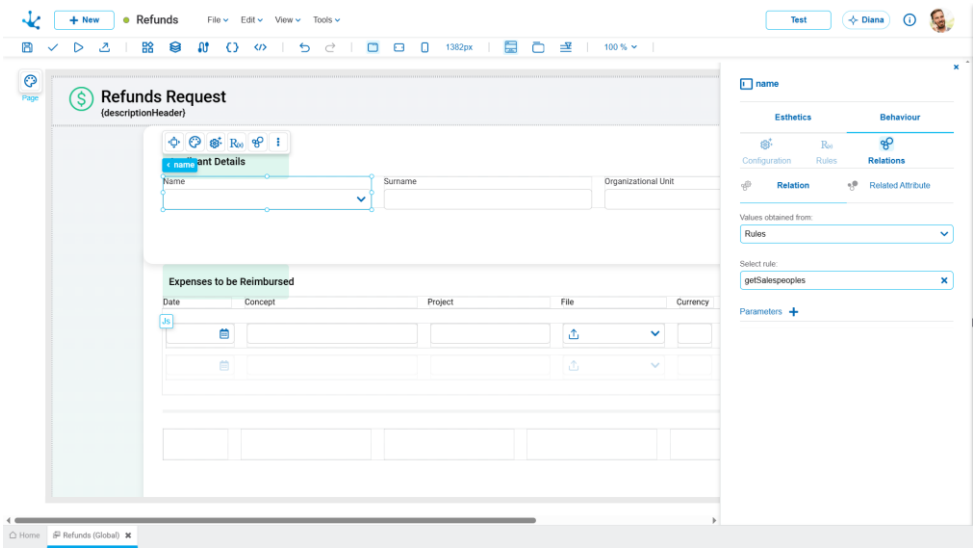
-  Hides the icons that are displayed.
-  Shows hidden icons.

The rules that relate to fields aim to implement business logic to dynamically define the values that the field can offer as selection options in a dropdown.

Relating a field to a business rule provides greater flexibility than binding to a value list, allowing to apply custom logic, add search filters, and dynamically adjust results based on different criteria.

Once the rule is implemented, it can be associated with a field. To do this, it is necessary to activate the Enable Relation to a Field property.

> *Rules can only relate to text fields.*



Properties

Select rule

It allows selecting a rule from the set of rules modeled in the environment, with the option to filter the set by entering text into the search field.

Parameters

If it is necessary to pass input parameters to the rule, they must be selected and assign values to them, which can be fixed values, variables or fields of the same entity.

Example

A previously modeled rule "getUserByName" can be related and enter a fixed value "perez" as a parameter. In this way, all the users that contain the text 'perez' in their last name.

In **Deyel**, there are different options for storing files attached to entities and pages, depending on the specific needs and requirements of the application.

To define the type of repository to use, it is necessary to consider the required level of security, the necessary data volume, and whether the files are used on web pages.

Types of File Repositories

The types of repositories used for file persistence are the following:

File in Data Base

The files are stored in the database of **Deyel** and have a maximum supported size of 5Mb. This option allows files to be stored centrally along with the associated information.

Recommended for:

- Clients who want to keep all data, including files, within the database of **Deyel**.
- Environments where the goal is to centralize information and the files to be stored are few and small in size (less than 1 MB).

Advantages:

- The files are fully integrated with the other fields of the entity.
- No additional infrastructure is required if the database already manages other application data.
- Easy to manage at the database level through backups and replication.

Disadvantages:

- It is not suitable for storing large volumes of files, and each file must be smaller than 1 GB in size.
- Limited scalability compared to other storage options.

Typical use cases:

- Storage of small to medium capacity files, such as internal documents.

File in Amazon S3 Repository

Amazon S3 (Simple Storage Service) is a highly scalable and secure cloud storage platform. Files are stored in S3 buckets, making them easy to access, manage, and control. In **Deyel**, files are hosted in private buckets, accessible only from the platform environment. This solution allows for storing large files, making it ideal for applications that require high data volumes or large files.

Recommended for:

- Data that requires access only from **Deyel** considering the permissions modeled on the different objects. These files may include invoices, sworn statements, tax reports, etc., which require data security and protection.

- Organizations that use Amazon Web Services (AWS) and want to take advantage of its cloud storage infrastructure.

Advantages:

- High availability and scalability of storage.
- Advanced features: access control, data encryption, and automatic backup.
- Direct integration with other AWS services.

Disadvantage:

- In on-premise installations, an internet connection and an AWS account are required for the creation and management of dedicated buckets.

Typical use cases:

- Confidential files that require encryption and access control.
- Companies that need a robust, scalable storage solution capable of handling large volumes of data.

File in Amazon S3 Public Repository

Amazon S3 (Simple Storage Service) is a highly scalable and secure cloud storage platform. Files are stored in S3 buckets, making them easy to access and manage. In this case, the files are hosted in public buckets, freely accessible from the web without restrictions. This option allows for storage of large files and facilitates open access to them.

Recommended for:

- Public files used on websites, such as logo images, documents without sensitive data, or marketing materials.

Advantages:

- Easy access for users from the application without the need for authentication.

Disadvantages:

- Risk of unwanted exposure if permission settings are not managed properly.
- It is not suitable for storing sensitive or confidential data.

Typical use case:

- Images, marketing files, logos, or public documents that must be accessible from anywhere without restrictions.

File in File System (FileSystem)

In this mode, the file is stored directly in the installation's local file system. Instead of using a database or cloud service, files are stored in a directory on the local server or a shared location within the internal network.

This type of repository is only available in On-Premise installations.

Recommended for:

- Environments where storing files in the database or in the cloud is not desired.
- Situations where it is preferable to keep files on the local file system for performance reasons or specific infrastructure requests.

Advantages:

- Fast and direct access to files, especially in local environments.
- No additional infrastructure is needed if a file server is already in place.

Disadvantages:

- Not available in the Cloud version.
- It requires careful management to avoid file loss due to potential infrastructure failures.

Typical use cases:

- Applications that require direct and fast access to files in a controlled environment.
- Situations where data does not need to be accessible outside the local environment or does not contain sensitive information.

Image

It allows the incorporation of any visual multimedia content.

The element properties are represented by icons in its context menu, where its operations are also available.

Subtypes

When selecting the option "Image" from the icon in the top toolbar, a list opens with the different subtypes of this element, which can be dragged into the modeling area. Each subtype has the element's properties modeled and predefined in a specific way.

- Rounded Image
- Square Image
- Small Image
- Medium Image
- Large Image

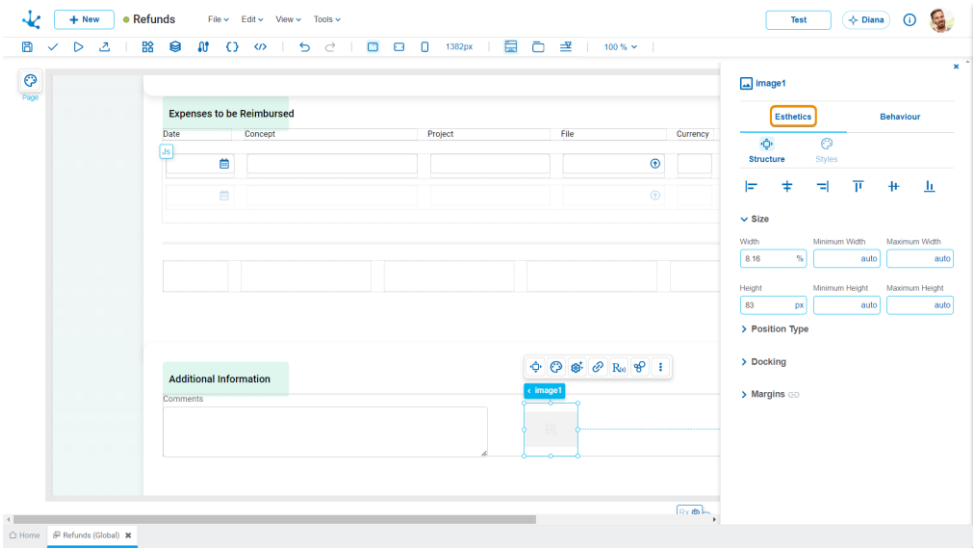
Classification of Properties

Properties are divided into two groups: esthetic properties and behavior properties.

Esthetic Properties

In the esthetic properties panel, the following are grouped:

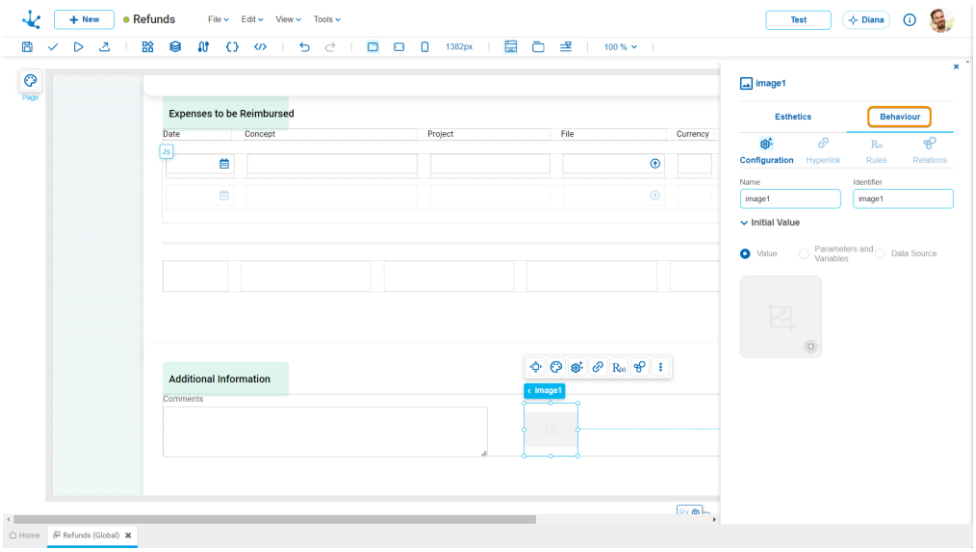
- Structure Properties
- Style Properties



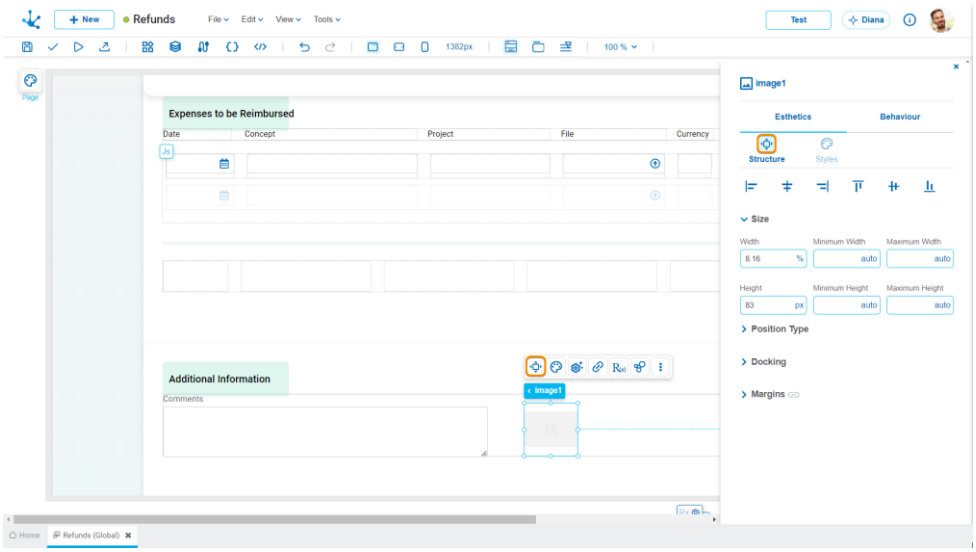
Behavior Properties

In the behavior properties panel, the following are grouped:

- Configuration Properties
- Hyperlink Properties
- Rule Properties
- Relation Properties



The structure properties panel of an element opens when selecting the icon  of its context menu.

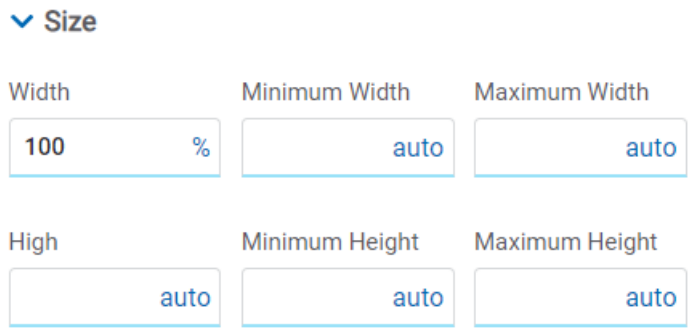


Alignment

Allows to align the element by selecting one of the available icons.

-  Align to the left.
-  Align to horizontal center.
-  Align to the right.
-  Align up.
-  Align vertical center.
-  Align bottom.

Size

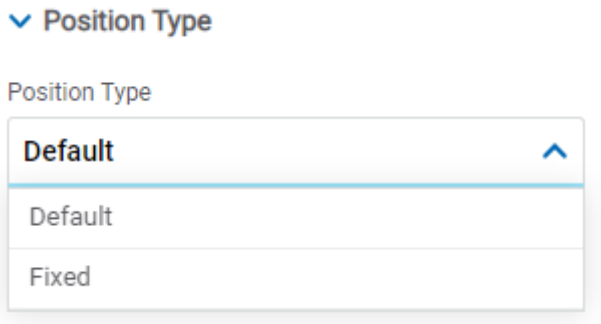


All size properties can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh). Additionally, for Width and Height properties, the "auto" option is added.

> *If selecting percentage (%) for the width and height properties, the size is calculated relative to the top element.*

Position Type

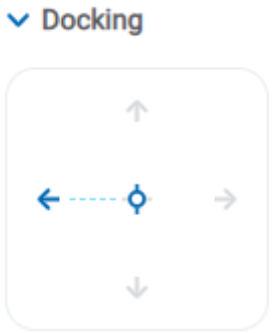
It determines if at the time of execution the element remains fixed on the page or if it moves as the scroll bar moves forward or backward.



Possible Values

- Default: The element has a relative position with respect to the superior element where it was placed (container or section).
- Fixed: Fixes the element to the page so that it is always visible in the same position. It allows to increase the visibility of important content.

Docking



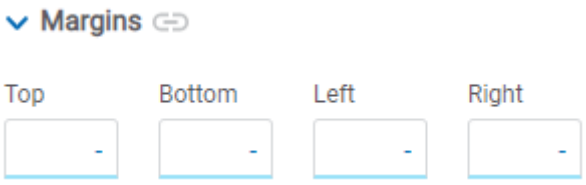
Determines the vertical and horizontal position of elements within the page section or container, when the screen is resized.

When an element is added or moved within the section or container, the modeler automatically docks it to the nearest corners or edges. It can also be configured by clicking on the arrows of the side to be docked to or on the center.

Once the element is docked, its exact position is indicated in the Margins property. The direction of the docking arrows determine the values that are automatically displayed for margins.

The docking position is indicated in the modeling area by dotted lines on the element.

Margins



It allows to define the distance of elements from the borders of their top element. The behavior of margins depends on the docking of the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Top
Distance to the top border of the highest ranking element.

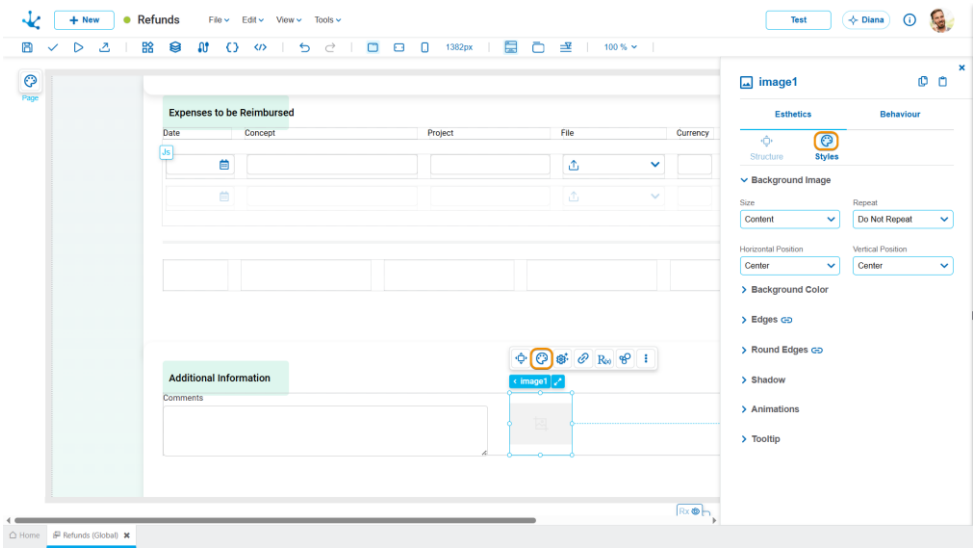
Bottom
Distance to the bottom border of the highest ranking element.

Left
Distance to the left border of the highest ranking element.

Right
Distance to the right border of the highest ranking element.

Allows the value entered in one of the margins to be copied to the other ones automatically.
Allows to indicate different values for each margin.

The style properties panel of an element opens when selecting the icon of its context menu.



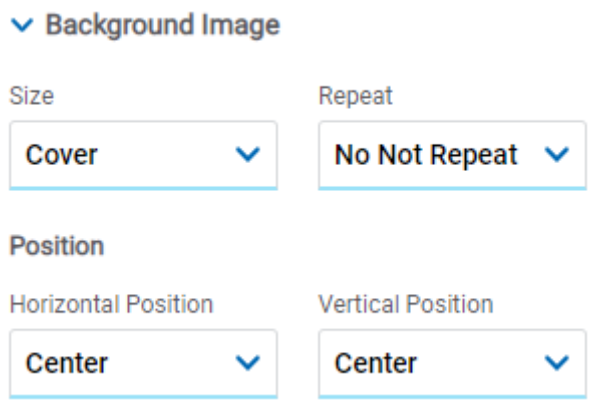
Styles

It allows copying the entire current configuration of the style panel for the selected element (Ctrl+Alt+C).

It allows applying the previously copied configuration to another element of the same type (Ctrl+Alt+V).

Background Image

Defines the properties of the background image.



Size

Possible Values

- Cover
- Content
- Auto

Repeat

Possible Values

- Repeat
- Repeat Hor.
- Repeat Vert.
- Spacing
- Rounded
- Do Not Repeat

Horizontal and Vertical Position

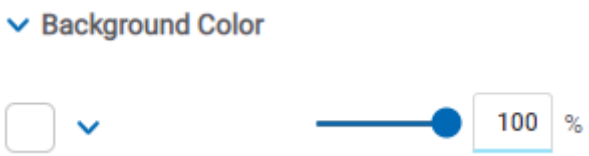
Possible Values

- Center
- Left
- Right

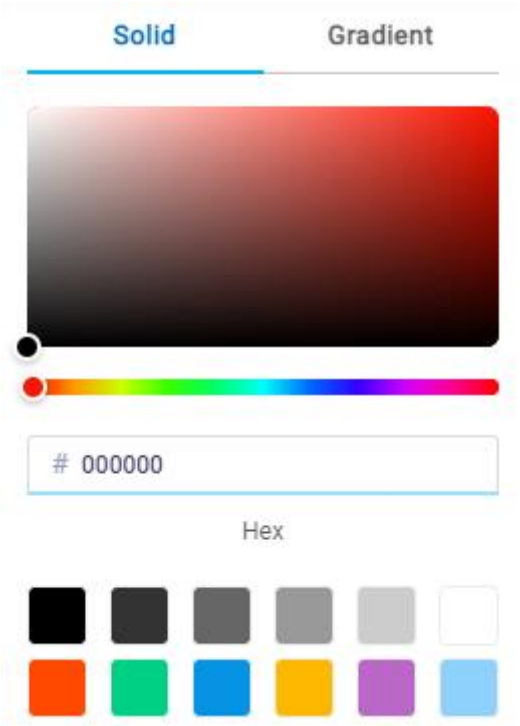
Background Color

It allows to model a background color for the element, it can be solid or gradient and different properties are defined for each one.

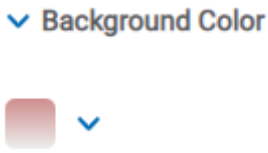
Solid



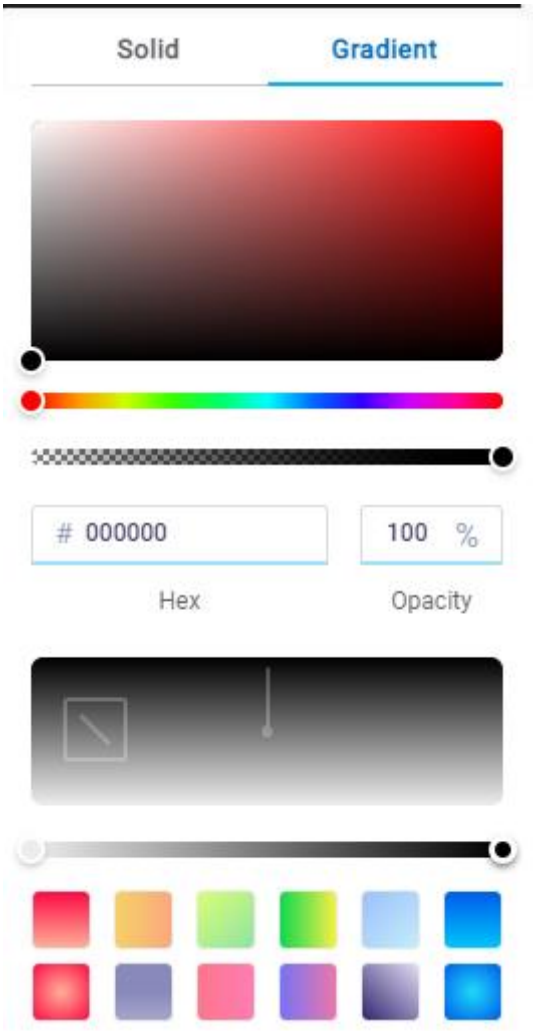
This option enables not only to select the color from a palette but also to define the degree of transparency.



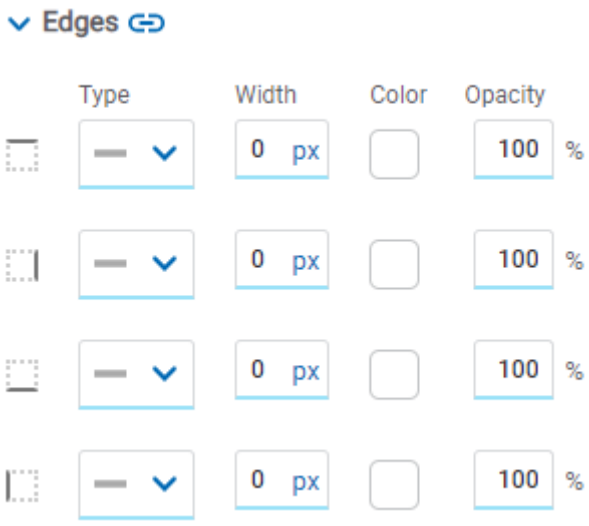
Gradient



This option enables to select not only the color from a palette but also to define its opacity and intensity according to the selected angle.



Edges



It allows to define the style of borders. Each one has its type, width, color and opacity defined, the latter as a percentage.

Allows the value entered in one of the borders to be copied to the other ones automatically.
Allows to indicate different values for each border.

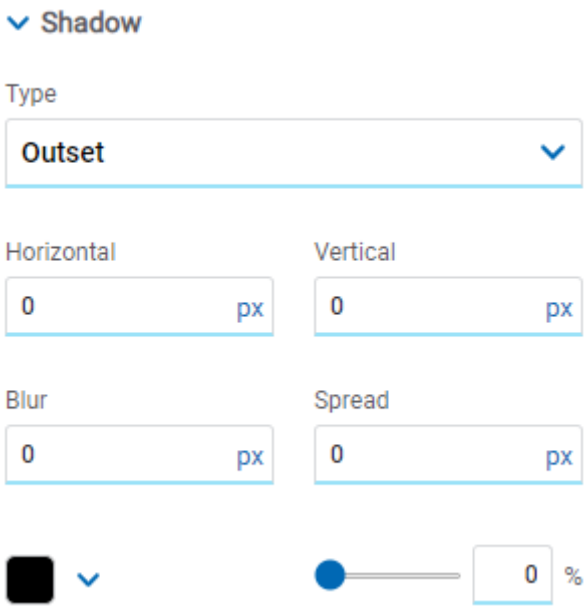
Round Edges



It allows to define the round edges at the corners of elements.
It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Allows the value entered in one of the borders to be copied to the other ones automatically.
Allows to indicate different values for each border.

Shadow



Allows to define a shadow effect around the element.

Type

Possible Values

- Outset
- Inset

Horizontal
Horizontal size of the shadow to the right of the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Vertical
Vertical size of the shadow below the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

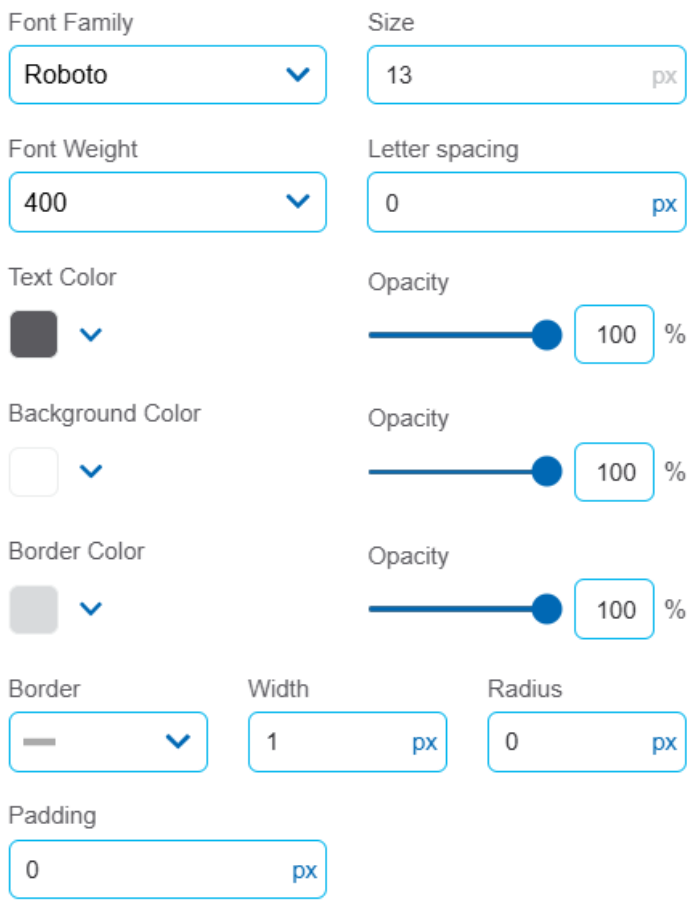
Blur
The greater this value is, the greater and lighter the shadow becomes. If not specified, its value is 0 and the shadow border is darker. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Spread
Positive values will cause the shadow to expand and grow bigger, while negative values will cause the shadow to shrink. If not specified, its value is 0 and the shadow will have the same size as the element.

Color
Defines the color of the shadow.

Transparency
Defines the transparency of the shadow.

Tooltip



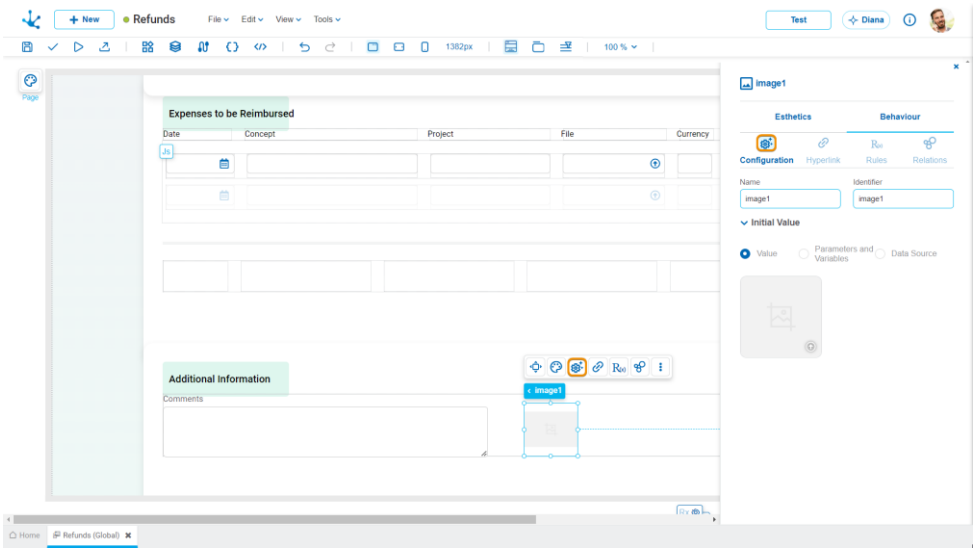
Source
It allows customizing the font, size, thickness and letter spacing of the pop-up message.

Colors
The color and opacity of the text, background and border can be defined. Borders are set uniformly for all sides of the tooltip.

Border
The border type, width, and border radius can be configured.

Configuration Properties

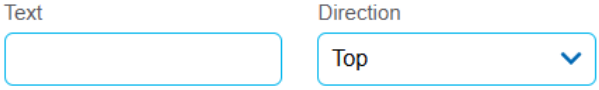
The configuration properties panel of an element opens when clicking the icon  of its context menu.



Name
Name used to reference the element during modeling.

Identifier
Uniquely identifies the element. It is used in the Javascript SDK.

Tooltip
Pop-up message that appears when hovering over an element, providing additional information.

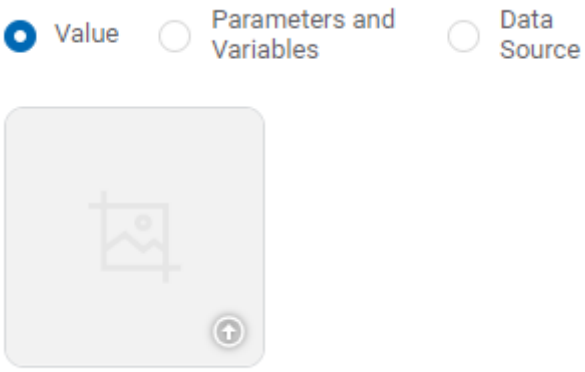


Text
Content of the message being displayed.

Direction
Position of the message in relation to the element (top, bottom, left, or right).

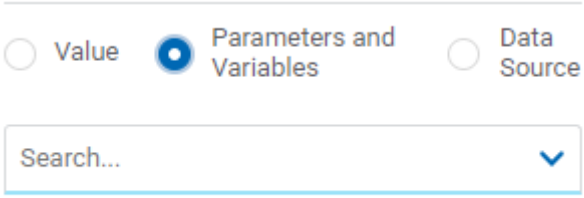
Allows to select the source of the element's content.

Initial Value



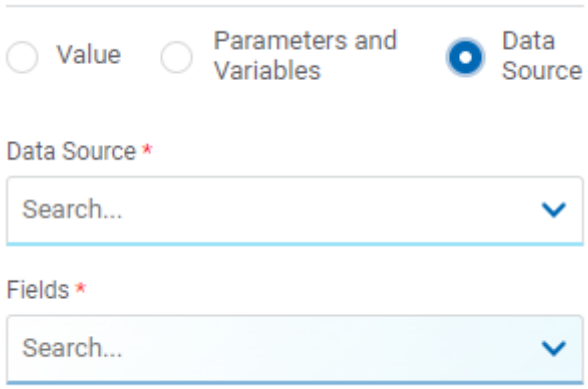
It allows adding an image from a file on the user's computer.

Parameters and Variables



Parameters and Variables
Allows to select a parameter or a variable from, whose value is displayed in the element.
This value can be a valid url.

Data Source



Data Source

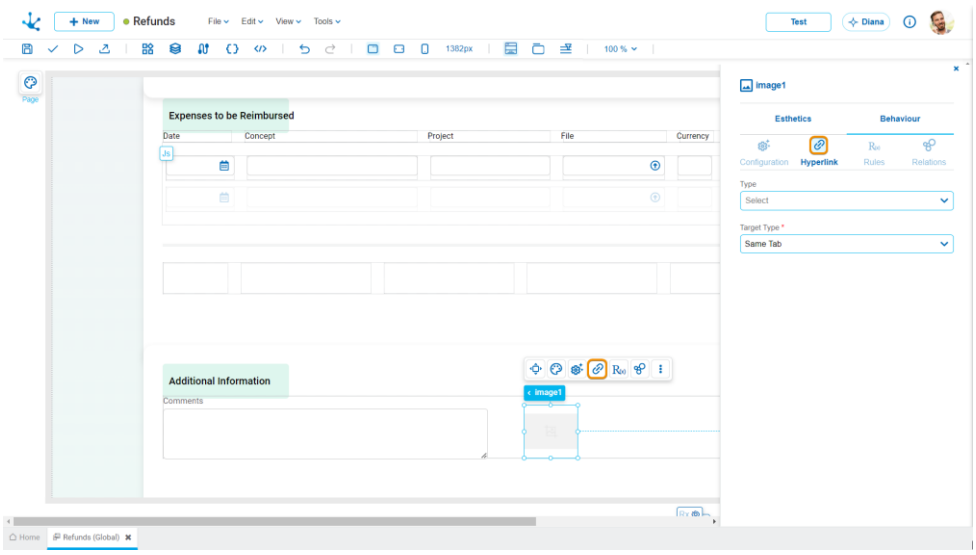
It allows selecting a previously defined data source within the object.

Fields

It allows selecting a field from the data retrieved in the chosen data source, whose value is displayed in the element.

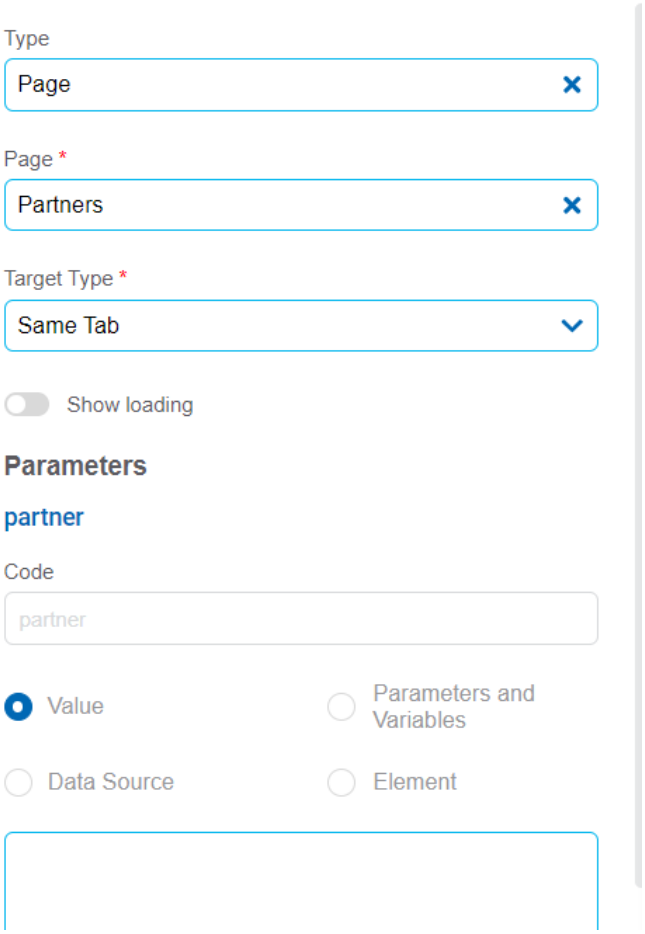
Hyperlink Properties

The hyperlink properties panel of an element opens when clicking the icon of its context menu.



Define the behavior of the element when selecting it. Different properties are enabled depending on the type of object selected.

Page



Page

The pages modeled in the environment are displayed.

Destination Type

The available options for opening the object are displayed.

Possible Values

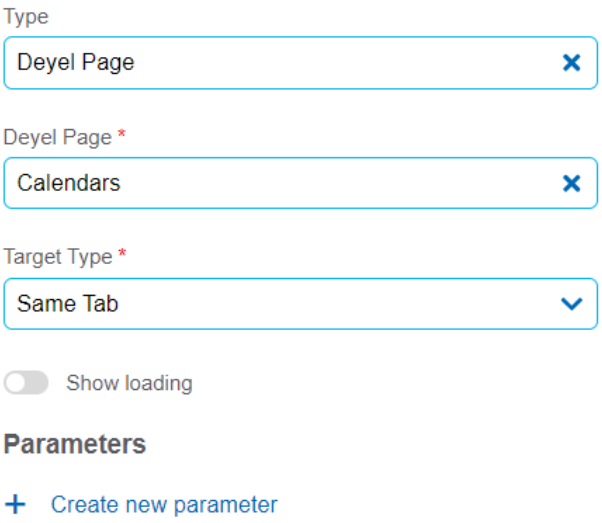
- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  - Modal Horizontal Size
    Defines its width.
  - Modal Vertical Size
    Defines its height.
- Iframe: selecting this option enables an additional property.
  - Destination Iframe
    Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading

Shows the loading icon and disables the user interaction until the redirection is over.

Parameters
If there are parameters, their value can be specified with text, parameters and variables or data source.

Deyel Page



Deyel Page
The pages belonging to **Deyel** are displayed.

Destination Type
The available options for opening the object are displayed.

Possible Values

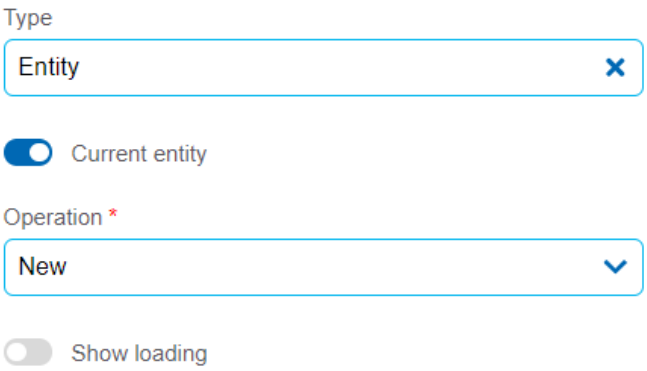
- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  - Modal Horizontal Size
    Defines its width.
  - Modal Vertical Size
    Defines its height.
- Iframe: selecting this option enables an additional property.
  - Destination Iframe
    Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Parameters

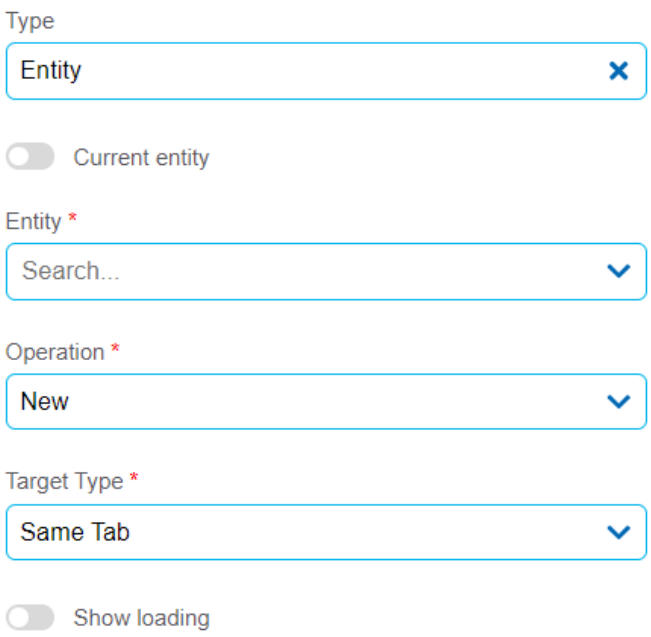
Allows sending parameters to the selected object type.

Entity



Current Entity
If this property is checked, the operation applies directly to the entity being modeled.

Operation
Defines the operation made when selecting the object.
Possible Values
- Create: Indicates that the panel of the current entity is opened to create an instance.
- Update: Indicates that the panel of the current entity is opened to update an instance.
- Delete: Indicates that the panel of the current entity is opened to delete an instance.
- Send Email: It is visible if the property Send Emails is enabled. Clicking on the modeled element opens a panel during instance execution to send an email. The panel size is adjusted with the properties Modal Width and Modal Height.
- Emails: Visible if the Send Emails property is enabled. Clicking on the modeled element opens a panel during instance execution to manage sent emails. The panel is displayed in the iframe defined in the Target Iframe property.



Entity
If the property Current Entity, is not checked, the entities modeled in the environment are displayed.

Operation
Defines the operation made when selecting the object.

Possible Values

- New: The panel of the entity selected in the previous property is opened to create an instance.
- Update The panel of the entity selected in the previous property is opened to update an instance.
- Show: The panel of the entity selected in the previous property is opened to show an instance.
- Grid: Indicates that the grid of the entity selected in the previous property is opened.

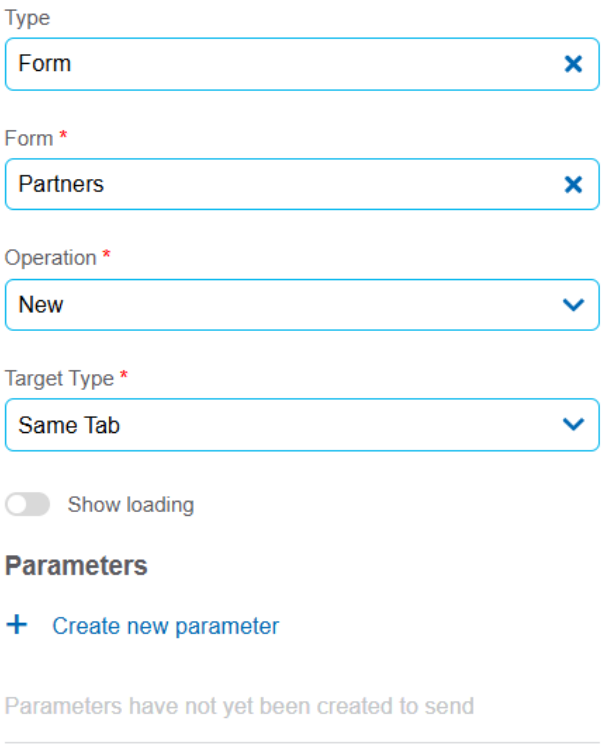
Destination Type
The available options for opening the object are displayed.

Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  Modal Horizontal Size
  Defines its width.
  Modal Vertical Size
  Defines its height.
- Iframe: selecting this option enables an additional property.
  Destination Iframe
  Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Forms



Forms
The forms modeled in the environment are displayed.

Operation
Defines the operation made when selecting the object.

Possible Values

- Create: Indicates that the panel of the object selected in the previous property is opened for the creation of an instance.
- Update: Indicates that the panel of the object selected in the previous property is opened for the update of an instance.
- Show: Indicates that the panel of the object selected in the previous property is opened for the query of an instance.
- Grid: Indicates that the grid of the object selected in the previous property is opened.

Destination Type
The available options for opening the object are displayed.

Possible Values

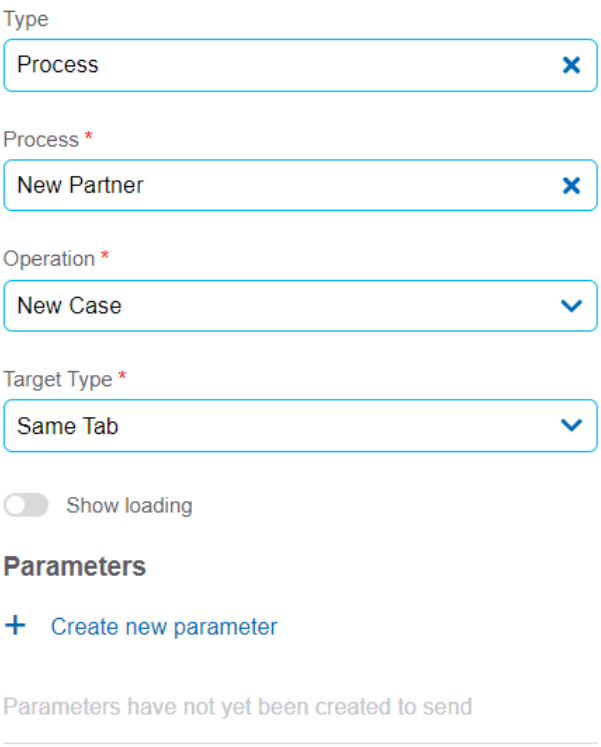
- Same Tab

- New Window
- Modal: selecting this option enables additional properties.
  Modal Horizontal Size
  Defines its width.
  Modal Vertical Size
  Defines its height.
- Iframe: selecting this option enables an additional property.
  Destination Iframe
  Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Parameters
Allows sending parameters to the selected object type.

Process



Process
The processes modeled in the environment are displayed.

Operation
Defines the operation made when selecting the object.

Destination Type
The available options for opening the object are displayed.

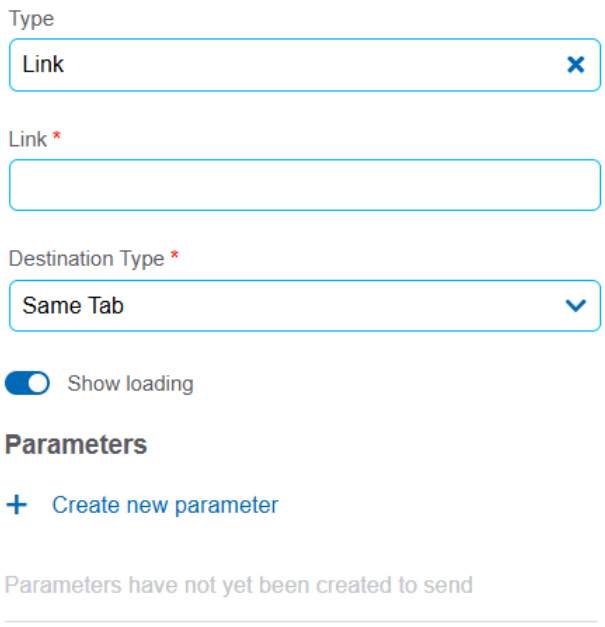
Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  Modal Horizontal Size
  Defines its width.
  Modal Vertical Size
  Defines its height.
- Iframe: selecting this option enables an additional property.
  Destination Iframe
  Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Parameters
Allows sending parameters to the selected object type.

Link



Link
Allows to enter any link.

Destination Type

The available options for opening the object are displayed.

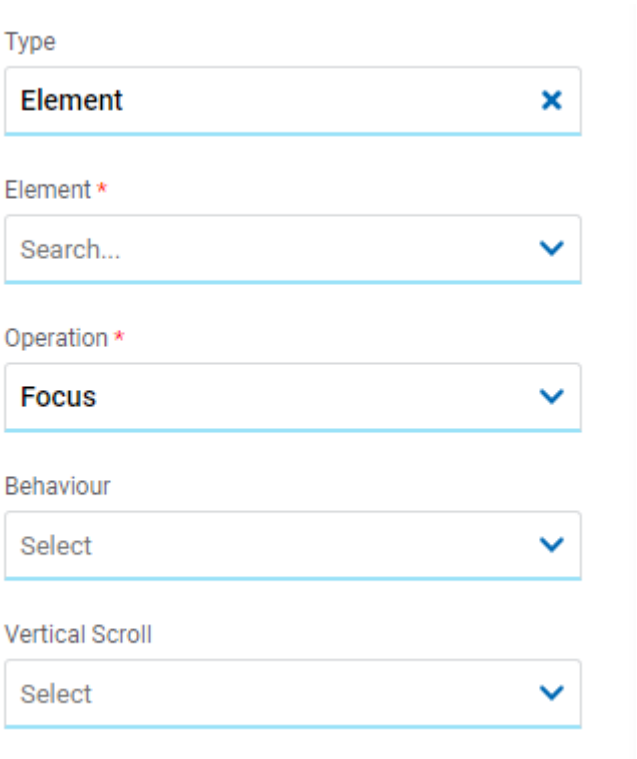
Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  Modal Horizontal Size
  Defines its width.
  Modal Vertical Size
  Defines its height.
- Iframe: selecting this option enables an additional property.
  Destination Iframe
  Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Parameters
Allows sending parameters to the selected object type.

Element



Element
The modeled elements in the object are displayed.

Operation
Defines the operation made when selecting the element.

Possible Values

- Focus
- Show
- Hide
- Show/Hide

Behaviour
Establishes the transition of the animation. It is only modelable if the "Focus" operation was selected.

Possible Values

- Auto
- Smooth

Vertical Scroll
It is only modelable if the "Focus" operation was selected.
Possible Values

- Start: Moves to the start of the selected element.
- Closest: Moves to the closest position to the selected element from the element the event fires.
- Center: Moves to the center of the selected element.
- End: Moves to the end of the selected element.

Repeater



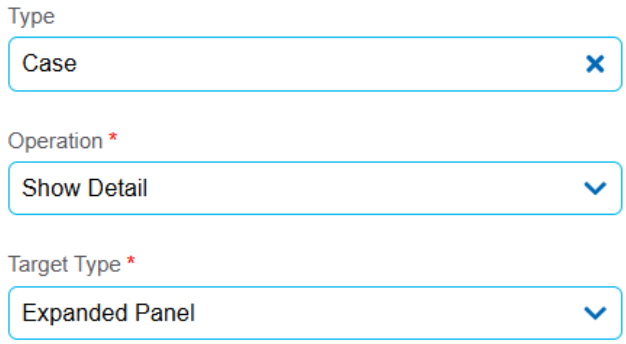
Repeater
The modeled repeaters from the object are displayed.

Operation
Defines the operation made when selecting the element.

Possible Values

- New: Allows for the creation of an item within the repeater. If this option is selected the additional properties are enabled.
  Creation Order
  It indicates in which position the new item of the container is added.
  Possible Values
    - Home
    - End
- Delete: Allows for the deletion of an item within the repeater.

Case



Case
Show case information.

Operation
Defines the operation made when selecting the element.

Possible Values

- Show Detail
- Show Graphic Execution

Destination Type
The available options for opening the object are displayed.

Possible Values

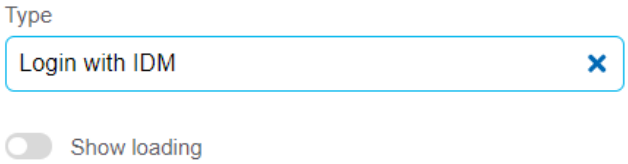
- Expanded Panel
- Modal: if this option is selected the additional properties are enabled.
  Modal Horizontal Size
  Define its width.
  Modal Vertical Size
  Define its height.

Back



It allows associating the event to go back in the browser to the element.

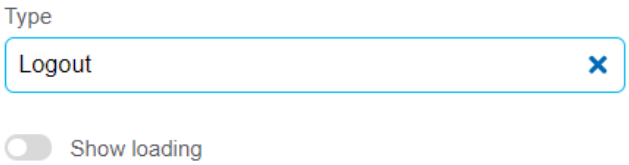
LogIn with IDM



Allows login with IDM.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

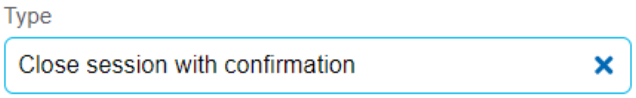
Logout



Allows logging the user out.

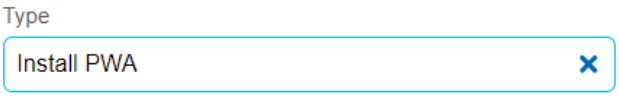
Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Close Session with Confirmation



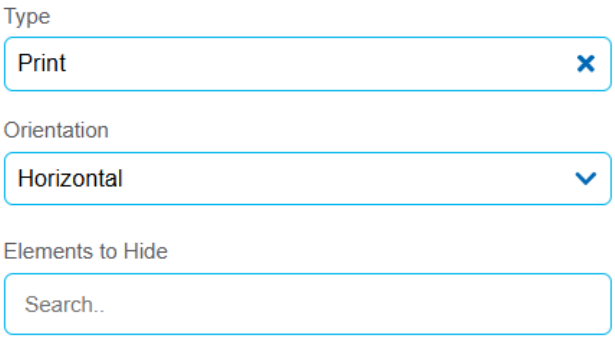
Displays a confirmation modal to logout the user.

Install PWA



Allows to install the application in the browser.

Print



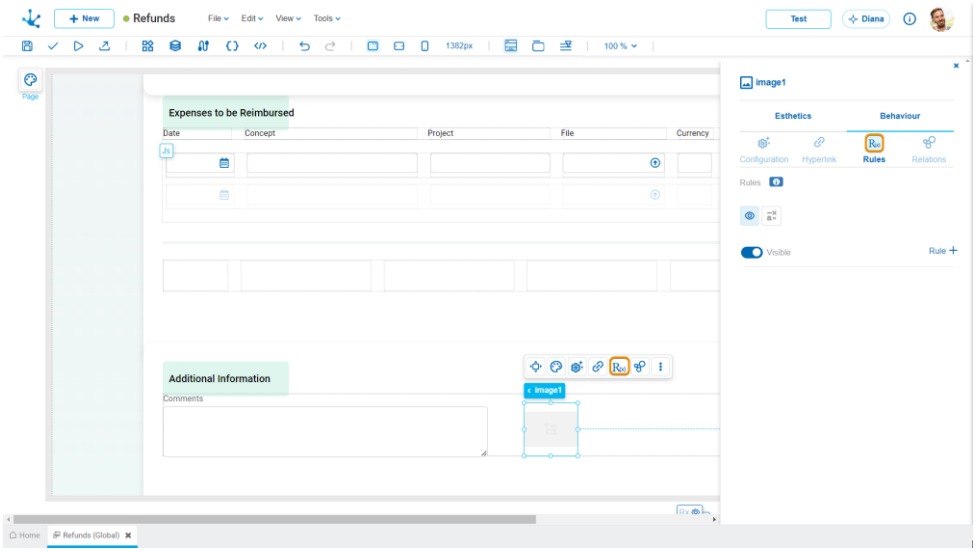
This type of hyperlink associated with an element allows printing the content of the entity or page that is being executed.

Orientation
Determines the layout of the content to be printed.
Possible Values
- Horizontal
- Vertical

Elements to hide
It allows selecting which elements of the entity or page should not be included in the printout.

Rule Properties

The rule properties panel of an element opens when clicking the icon  of its context menu.



Properties

Rules

Embedded rules on behavior, validation and calculation can be defined, associated with an element, by using the wizard (ctrl + space).

Shows syntax examples for writing the rules.

Visible

Indicates whether the element is visible. If this property is not checked, the element is not displayed in the page.

Visible (default) Not visible

Rule + Opens an edit area where a rule to determine the visibility condition can be defined. If a rule is defined, the icon is displayed with light blue borders.

Calculation

Rule + Opens an edit area where you can define the expression to be executed to calculate the element value. If a rule is defined, the icon is displayed with light blue borders.

Operations when defining the rule:

Saves the new or modified rule

Cancels the operation

Operations once the rule is defined:

Edits the existing rule

Deletes the rule

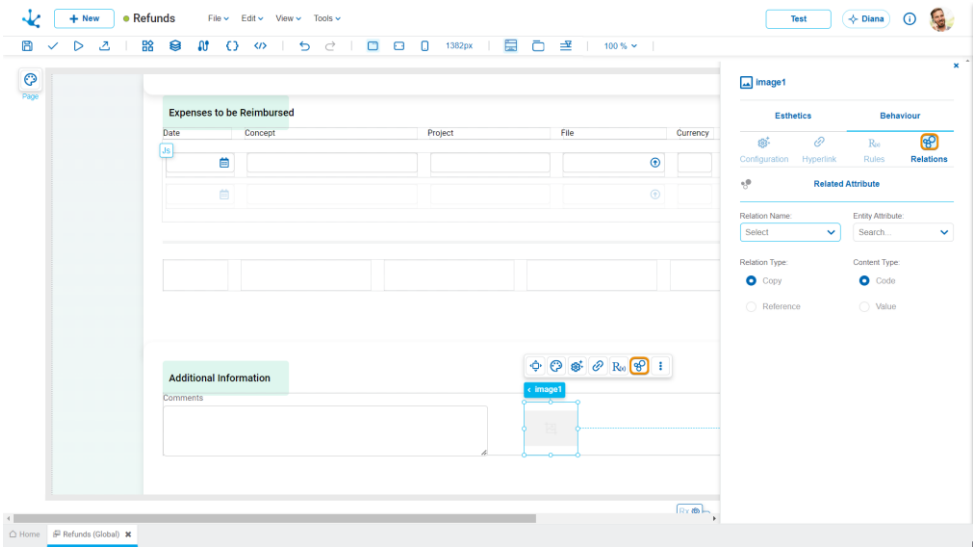
Events

Images allow the use of different events.

Event	Description
onMouseIn()	It is executed when the cursor is positioned on the element.
onMouseOut()	It is executed when the cursor moves out of the element.
onClick()	It is executed when clicking on the element.
onDoubleClick()	It is executed when clicking twice on the element.
onInit()	It is executed before the element is loaded.
afterViewInit()	It is executed after viewing the element.
onViewportEnter()	It is executed when the element is visible.
onViewportLeave()	It is executed when the element is no longer visible.

Relation Properties

The relation properties panel of an element opens when clicking the icon  of its context menu.



Related Attribute

A related attribute indicates that the field value is retrieved from the attribute value of another entity. One of the relations defined on the entity must be selected, and then it should be specified which attribute of the related entity it links to.

A field can be modeled as a related attribute and it can also have a relation defined to an entity, a value list, or a rule. This allows the value of the related attribute to be retrieved on this field and the functionality provided by the relation to be used.

Properties

Relation Name
The list of relations defined in the entity is displayed, allowing the selection of one of them.

Entity Attribute
The list of attributes of the selected entity compatible with the field type is displayed. Attributes can have a relation modeled with an entity, a value list, or a rule in the selected entity.

Relation Type
Copy
When this property is checked, once the retrieved value is uploaded, it does not automatically update even if the value of the related attribute changes.
Reference
When this property is checked, the field value is subject to the current value of the related attribute, that is, the updated value is automatically displayed.

Content Type
The element content varies depending on whether the Code or Value property is checked when:
- The entity attribute has a modeled relation.
- The field has the option "Entity" selected in the Values Obtained from property.

Code
When this property is checked, if the relation is modeled with a:
- Related entity, the identifier of the entity is displayed.
- Value list, the value list code is displayed.
- Rule, the code returned by the rule's execution is displayed.

Value
When this property is checked, if the relation is modeled with a:
- Related entity, the short description of the entity is displayed.
- Value list, the value list description is displayed.
- Rule, the description returned by the rule's execution is displayed.

Video

It allows the incorporation of any visual multimedia content.

The element properties are represented by icons in its context menu, where its operations are also available.

Subtypes

When selecting the option "Video" from the icon in the top toolbar, a list opens with the different subtypes of this element, which can be dragged into the modeling area. Each subtype has the element's properties modeled and predefined in a specific way.

- Small Video
- Medium Video
- Large Video

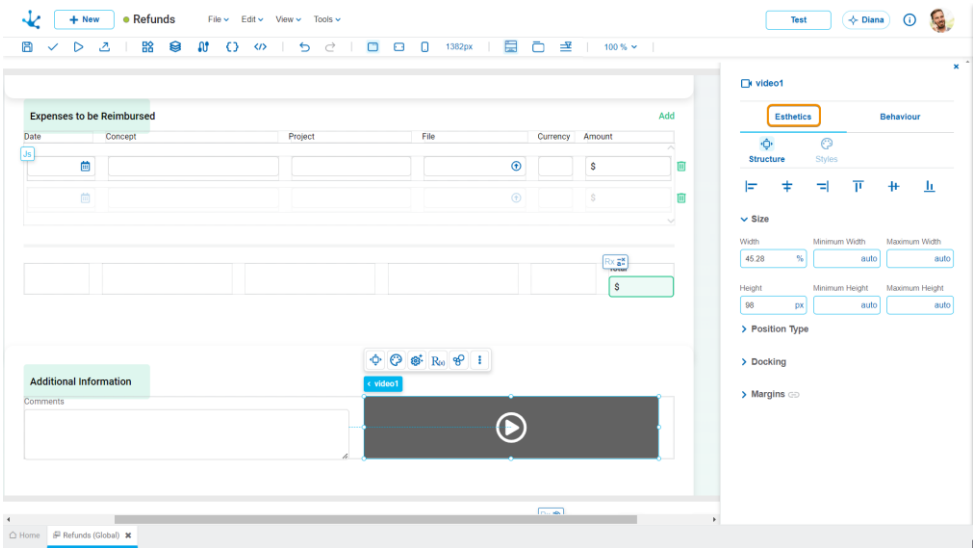
Classification of Properties

Properties are divided into two groups: esthetic properties and behavior properties.

Esthetic Properties

In the esthetic properties panel, the following are grouped:

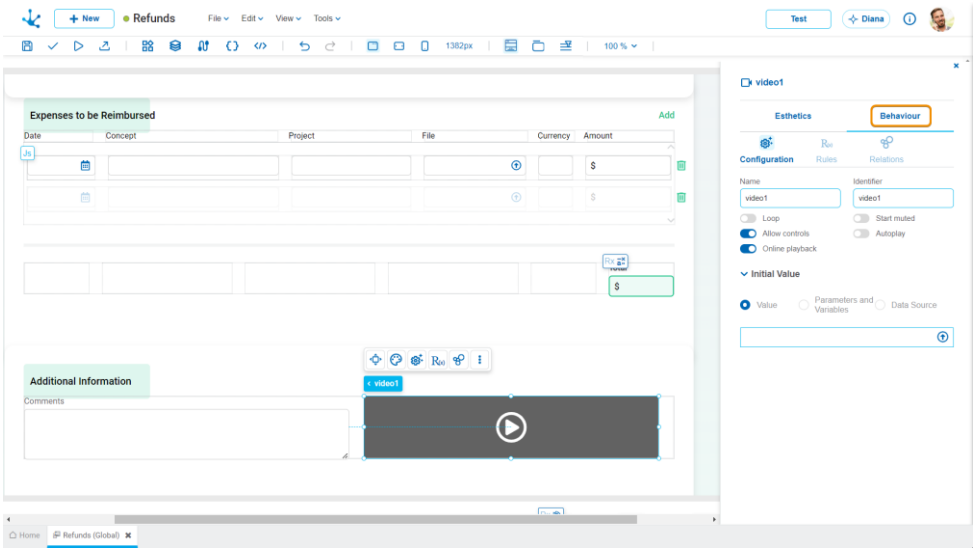
- Structure Properties
- Style Properties



Behavior Properties

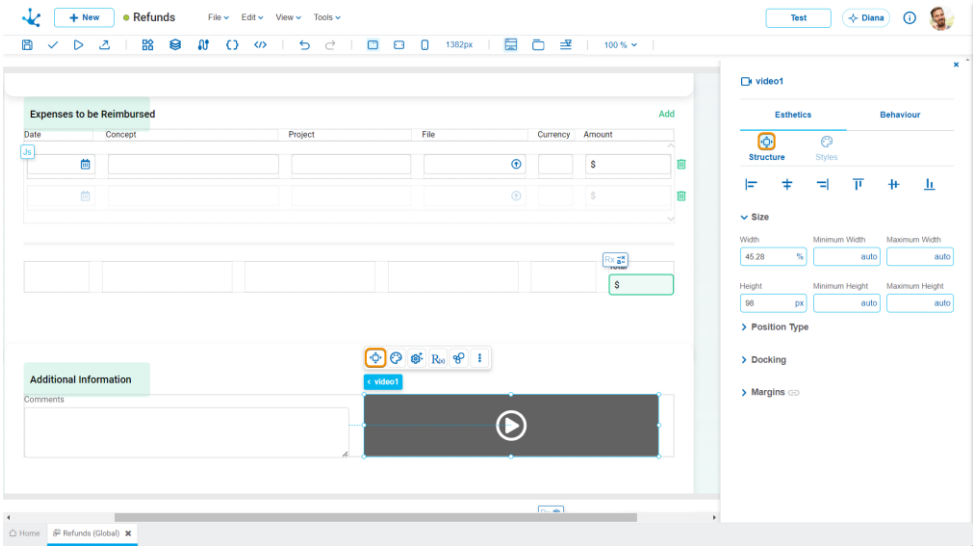
In the behavior properties panel, the following are grouped:

- Configuration Properties
- Rule Properties
- Relation Properties



Structure Properties

The structure properties panel of an element opens when selecting the icon  of its context menu.



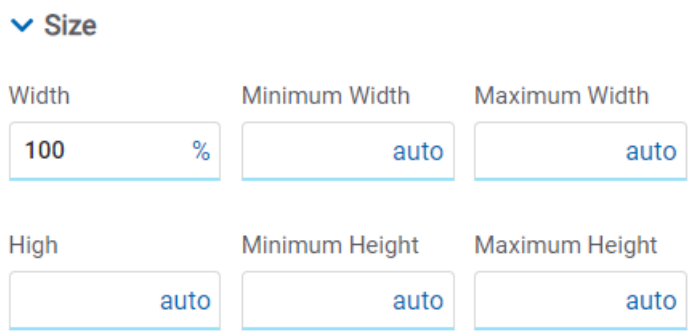
Alignment

Allows to align the element by selecting one of the available icons.

-  Align to the left.
-  Align to horizontal center.
-  Align to the right.
-  Align up.

- Align vertical center.
- Align bottom.

Size

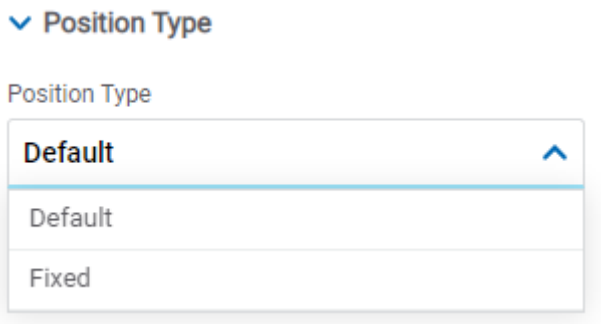


All size properties can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh). Additionally, for Width and Height properties, the "auto" option is added.

> *If selecting percentage (%) for the width and height properties, the size is calculated relative to the top element.*

Position Type

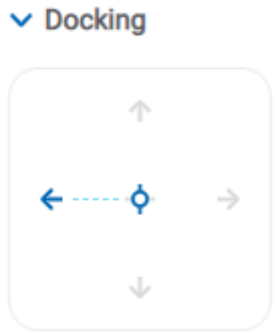
It determines if at the time of execution the element remains fixed on the page or if it moves as the scroll bar moves forward or backward.



Possible Values

- Default: The element has a relative position with respect to the superior element where it was placed (container or section).
- Fixed: Fixes the element to the page so that it is always visible in the same position. It allows to increase the visibility of important content.

Docking



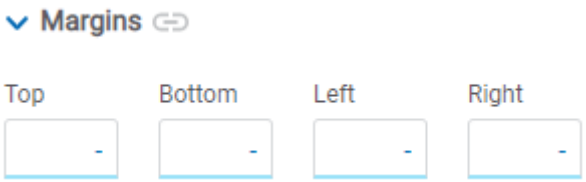
Determines the vertical and horizontal position of elements within the page section or container, when the screen is resized.

When an element is added or moved within the section or container, the modeler automatically docks it to the nearest corners or edges. It can also be configured by clicking on the arrows of the side to be docked to or on the center.

Once the element is docked, its exact position is indicated in the Margins property. The direction of the docking arrows determine the values that are automatically displayed for margins.

The docking position is indicated in the modeling area by dotted lines on the element.

Margins



It allows to define the distance of elements from the borders of their top element. The behavior of margins depends on the docking of the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Top
Distance to the top border of the highest ranking element.

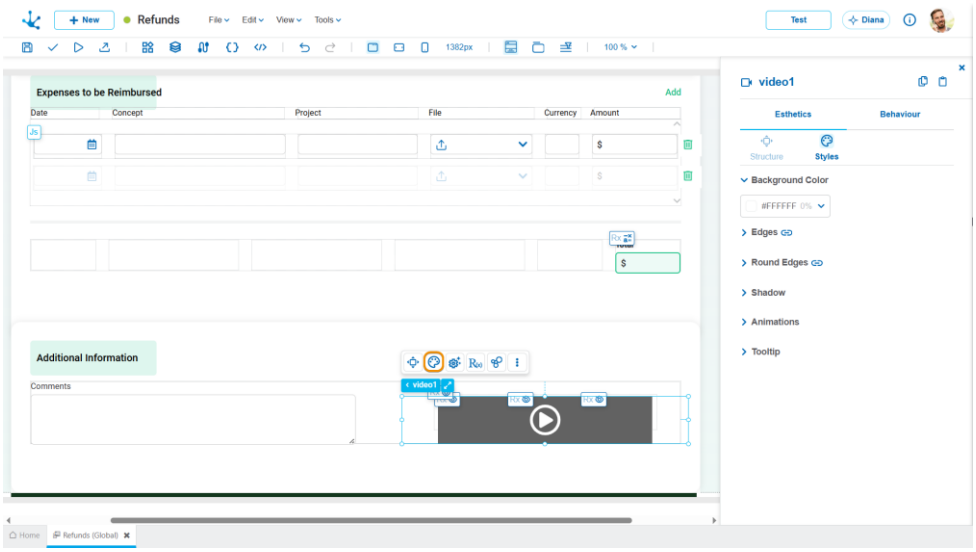
Bottom
Distance to the bottom border of the highest ranking element.

Left
Distance to the left border of the highest ranking element.

Right
Distance to the right border of the highest ranking element.

Allows the value entered in one of the margins to be copied to the other ones automatically.
Allows to indicate different values for each margin.

The style properties panel of an element opens when selecting the icon of its context menu.



Styles

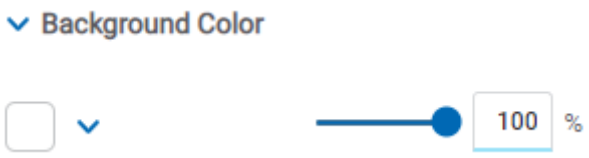
It allows copying the entire current configuration of the style panel for the selected element (Ctrl+Alt+C).

It allows applying the previously copied configuration to another element of the same type (Ctrl+Alt+V).

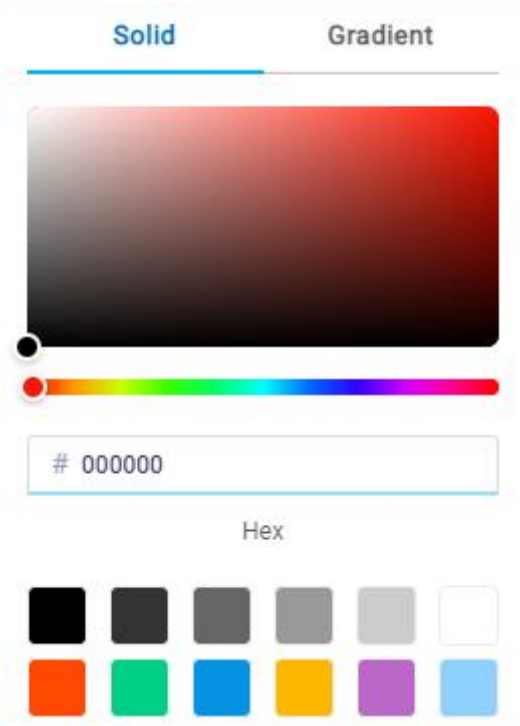
Background Color

It allows to model a background color for the element, it can be solid or gradient and different properties are defined for each one.

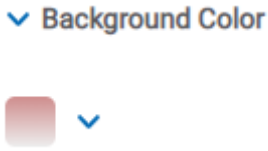
Solid



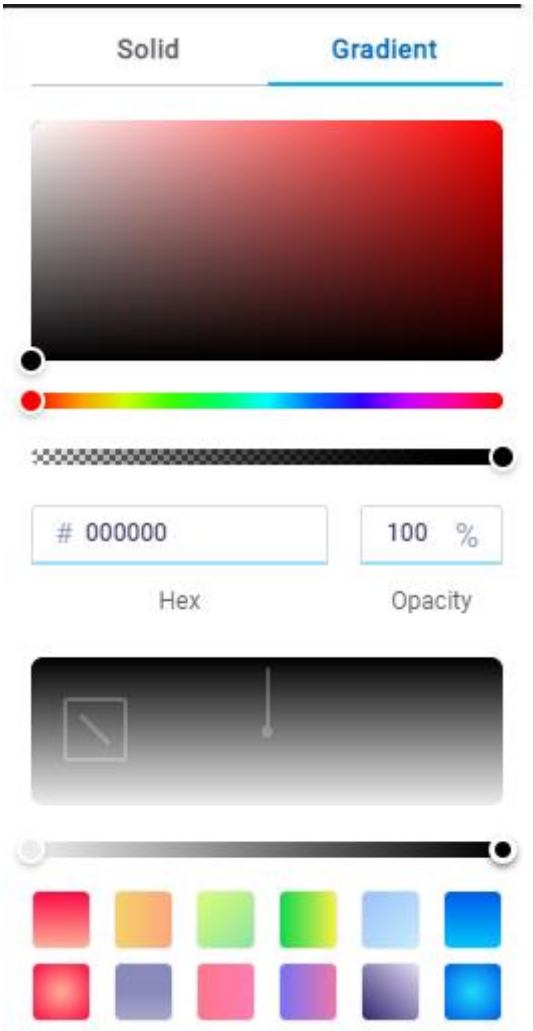
This option enables not only to select the color from a palette but also to define the degree of transparency.



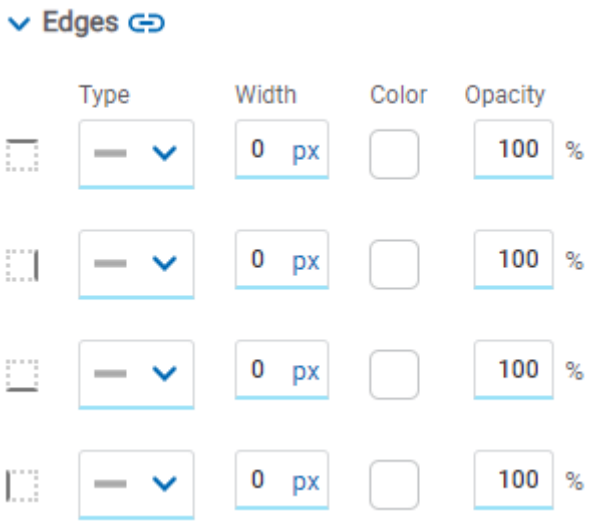
Gradient



This option enables to select not only the color from a palette but also to define its opacity and intensity according to the selected angle.



Edges



It allows to define the style of borders. Each one has its type, width, color and opacity defined, the latter as a percentage.

Allows the value entered in one of the borders to be copied to the other ones automatically.
Allows to indicate different values for each border.

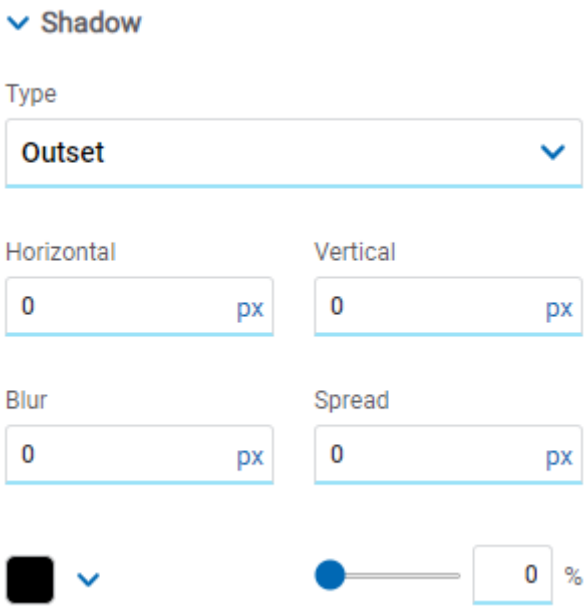
Round Edges



It allows to define the round edges at the corners of elements.
It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Allows the value entered in one of the borders to be copied to the other ones automatically.
Allows to indicate different values for each border.

Shadow



Allows to define a shadow effect around the element.

Type

Possible Values

- Outset
- Inset

Horizontal
Horizontal size of the shadow to the right of the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Vertical
Vertical size of the shadow below the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

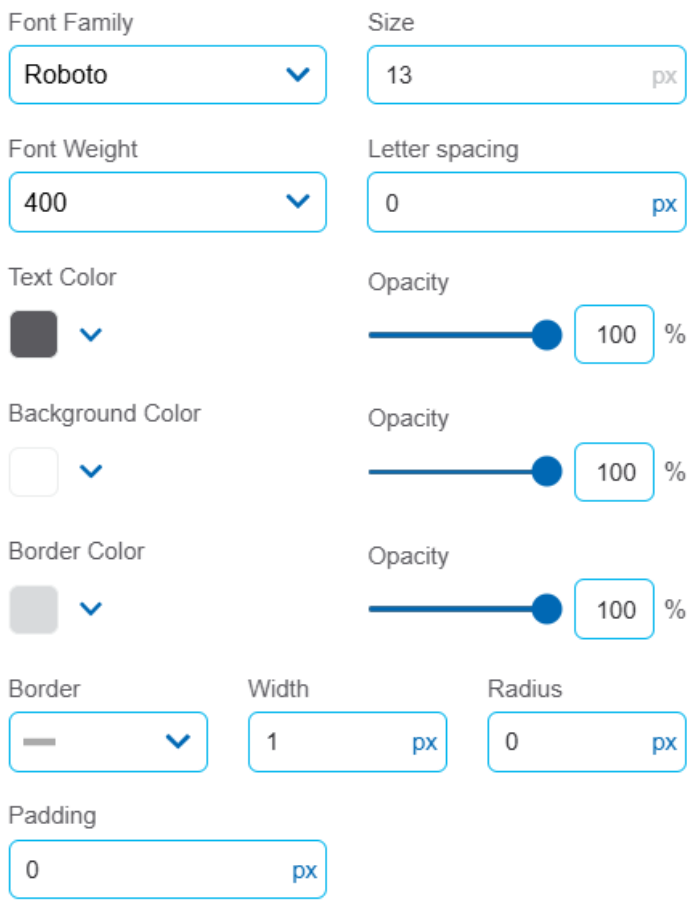
Blur
The greater this value is, the greater and lighter the shadow becomes. If not specified, its value is 0 and the shadow border is darker. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Spread
Positive values will cause the shadow to expand and grow bigger, while negative values will cause the shadow to shrink. If not specified, its value is 0 and the shadow will have the same size as the element.

Color
Defines the color of the shadow.

Transparency
Defines the transparency of the shadow.

Tooltip



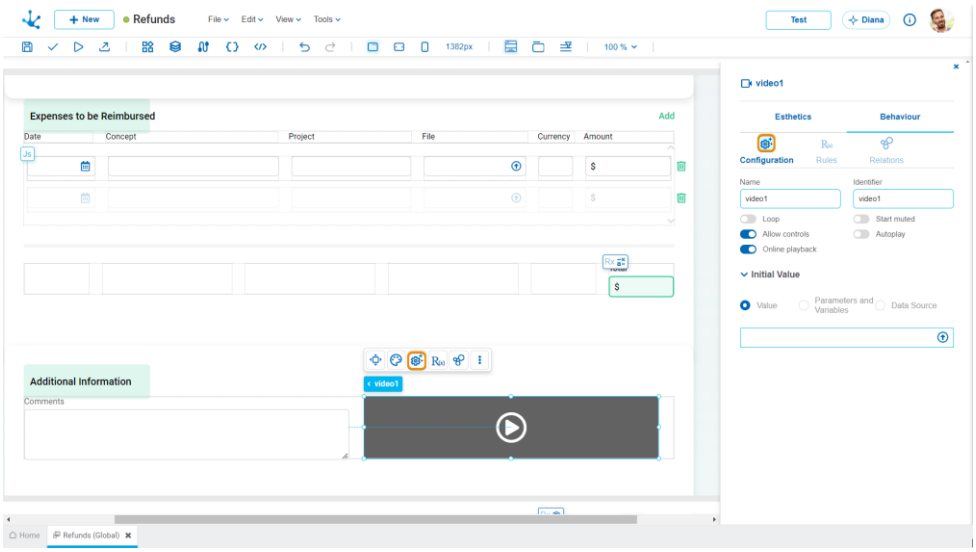
Source
It allows customizing the font, size, thickness and letter spacing of the pop-up message.

Colors
The color and opacity of the text, background and border can be defined. Borders are set uniformly for all sides of the tooltip.

Border
The border type, width, and border radius can be configured.

Configuration Properties

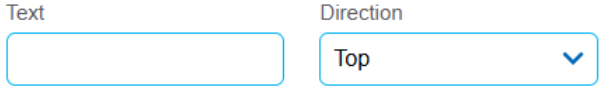
The configuration properties panel of an element opens when clicking the icon  of its context menu.



Name
Name used to reference the element during modeling.

Identifier
Uniquely identifies the element. It is used in the Javascript SDK.

Tooltip
Pop-up message that appears when hovering over an element, providing additional information.



Text
Content of the message being displayed.

Direction
Position of the message in relation to the element (top, bottom, left, or right).

Loop
It allows the video to play on a continuous loop. When the video ends, it automatically restarts from the beginning and continues playing indefinitely.

Start muted
It allows the video to start without sound. The user can enable audio manually if desired.

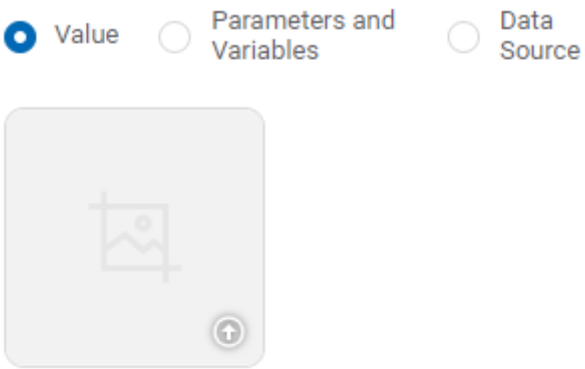
Allow controls
Enables video playback controls such as play and pause buttons, progress bar, volume, and fullscreen option.

Autoplay
It allows the video to start playing automatically as soon as the entity is loaded. It can be combined with the Start muted property to prevent unexpected audio playback.

Online playback
It allows the video to play online within the entity, rather than opening in a fullscreen player. It is especially useful on mobile devices, where videos often play full screen by default.

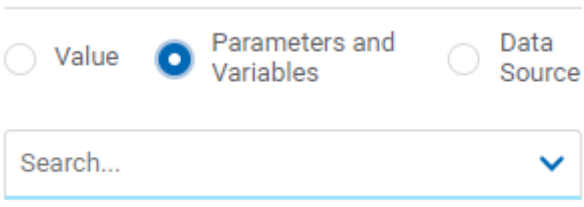
Initial Value

Allows to select the source of the element's content.



It allows adding a video from a file on the user's computer.

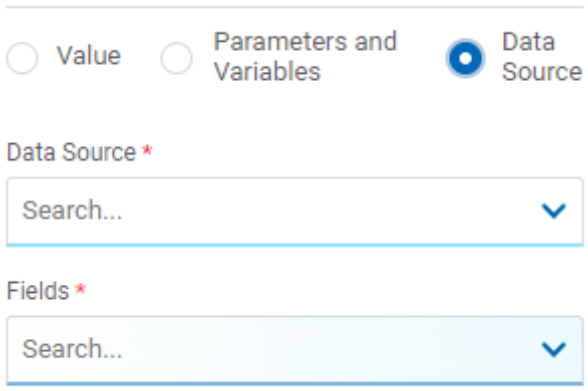
Parameters and Variables



Parameters and Variables

Allows to select a parameter or a variable from, whose value is displayed in the element. This value can be a valid url.

Data Source



Data Source

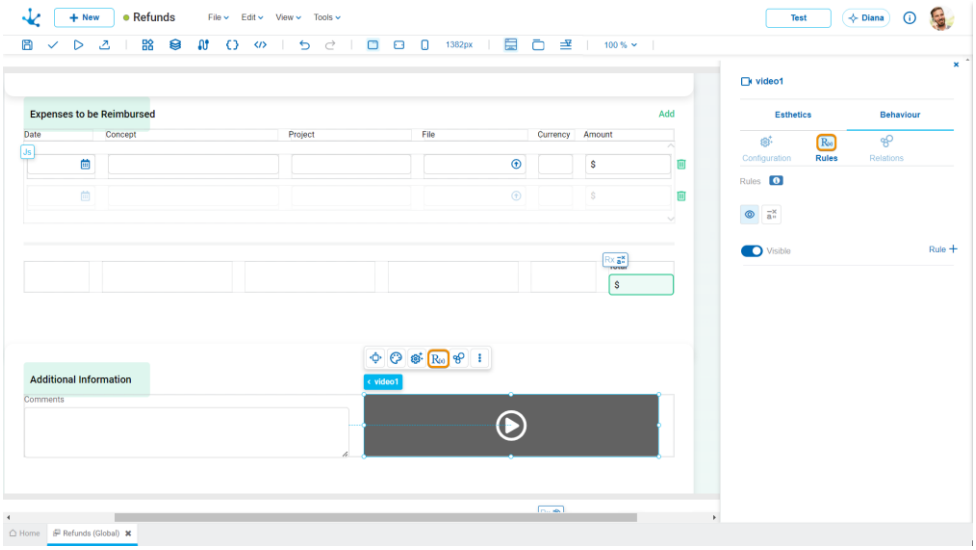
It allows selecting a previously defined data source within the object.

Fields

It allows selecting a field from the data retrieved in the chosen data source, whose value is displayed in the element.

Rule Properties

The rule properties panel of an element opens when clicking the icon  of its context menu.



Properties

Rules

Embedded rules on behavior, validation and calculation can be defined, associated with an element, by using the wizard (ctrl + space).

Shows syntax examples for writing the rules.

Visible

Indicates whether the element is visible. If this property is not checked, the element is not displayed in the page.

Visible (default) Not visible

Rule + Opens an edit area where a rule to determine the visibility condition can be defined. If a rule is defined, the icon is displayed with light blue borders.

Calculation

Rule + Opens an edit area where you can define the expression to be executed to calculate the element value. If a rule is defined, the icon is displayed with light blue borders.

Operations when defining the rule:

Saves the new or modified rule

Cancels the operation

Operations once the rule is defined:

Edits the existing rule

Deletes the rule

Events

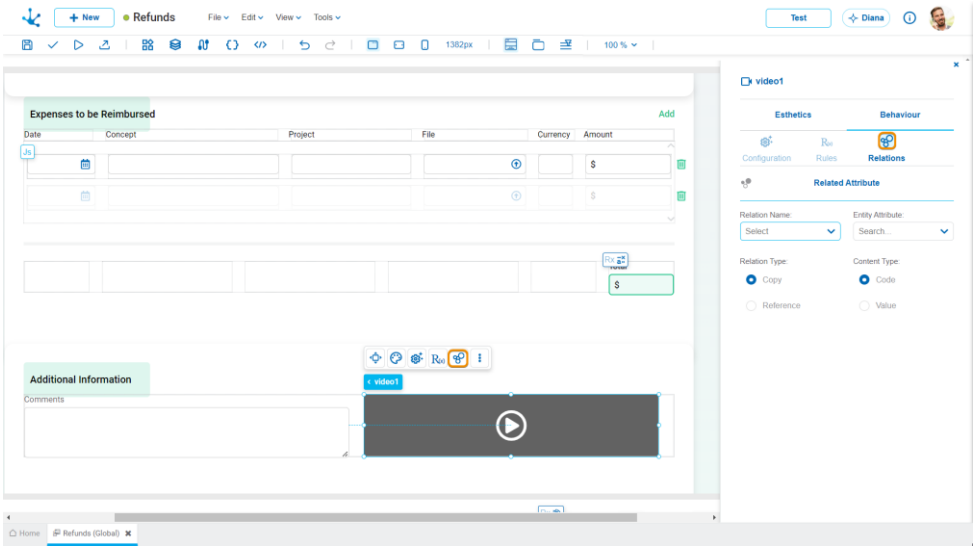
Videos allow the use of different events.

Event	Description
onMouseIn()	It is executed when the cursor is positioned on the element.
onMouseOut()	It is executed when the cursor moves out of the element.
onClick()	It is executed when clicking on the element.
onDoubleClick()	It is executed when clicking twice on the element.

Event	Description
onInit()	It is executed before the element is loaded.
afterViewInit()	It is executed after viewing the element.
onViewportEnter()	It is executed when the element is visible.
onViewportLeave()	It is executed when the element is no longer visible.

Relation Properties

The relation properties panel of an element opens when clicking the icon of its context menu.



Related Attribute

A related attribute indicates that the field value is retrieved from the attribute value of another entity. One of the relations defined on the entity must be selected, and then it should be specified which attribute of the related entity it links to.

A field can be modeled as a related attribute and it can also have a relation defined to an entity, a value list, or a rule. This allows the value of the related attribute to be retrieved on this field and the functionality provided by the relation to be used.

Properties

Relation Name
The list of relations defined in the entity is displayed, allowing the selection of one of them.

Entity Attribute

The list of attributes of the selected entity compatible with the field type is displayed. Attributes can have a relation modeled with an entity, a value list, or a rule in the selected entity.

Relation Type

Copy

When this property is checked, once the retrieved value is uploaded, it does not automatically update even if the value of the related attribute changes.

Reference

When this property is checked, the field value is subject to the current value of the related attribute, that is, the updated value is automatically displayed.

Content Type

The element content varies depending on whether the Code or Value property is checked when:

- The entity attribute has a modeled relation.
- The field has the option "Entity" selected in the Values Obtained from property.

Code

When this property is checked, if the relation is modeled with a:

- Related entity, the identifier of the entity is displayed.
- Value list, the value list code is displayed.
- Rule, the code returned by the rule's execution is displayed.

Value

When this property is checked, if the relation is modeled with a:

- Related entity, the short description of the entity is displayed.
- Value list, the value list description is displayed.
- Rule, the description returned by the rule's execution is displayed.

Layout

This element is a structure composed of a set of Items and allows them to be structured based on different styles: column, row, stripe, and mosaic. These items are containers, allowing other elements to be incorporated. These elements can move along with each item when the layout structure is resized.

The entity modeler also includes other structural elements that facilitate field modeling, improving the time it takes to model an entity without worrying about the aesthetic aspects of the fields. All aesthetic aspects of these elements are defined by default to adapt to the different breakpoints.

Both groups of structural elements are used by subtype selection.

The element properties are represented by icons in its context menu, where its operations are also available.

Subtypes

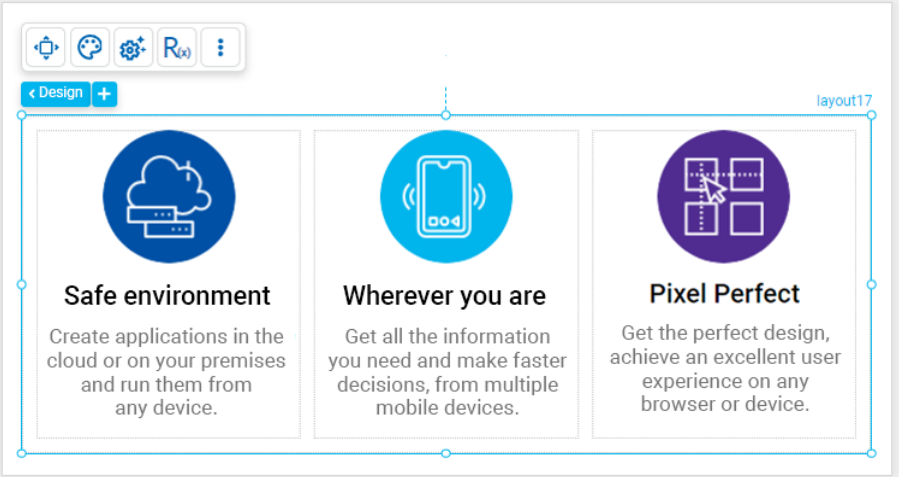
When selecting the option "Layout" from the icon in the top toolbar, a list opens with the different subtypes of this element, which can be dragged and dropped into the modeling area. Each subtype has the element's properties modeled and built in a specific way.

- Row

- Column
- Stripe
- Mosaic
- Columns and Rows
- Rows and Columns
- Field Group
- Fields in Rows and Columns

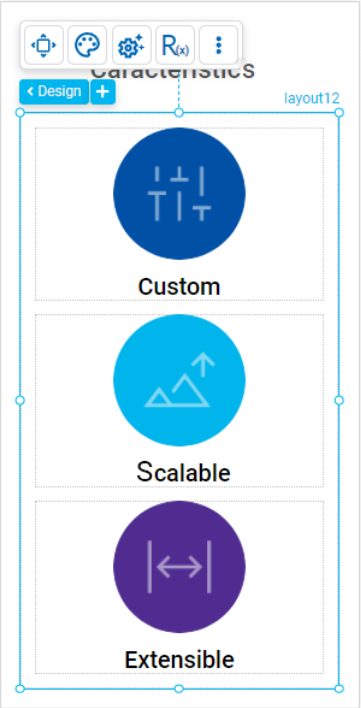
Row

It allows to define items arranged in rows.Within each row item, any type of element can be dragged.



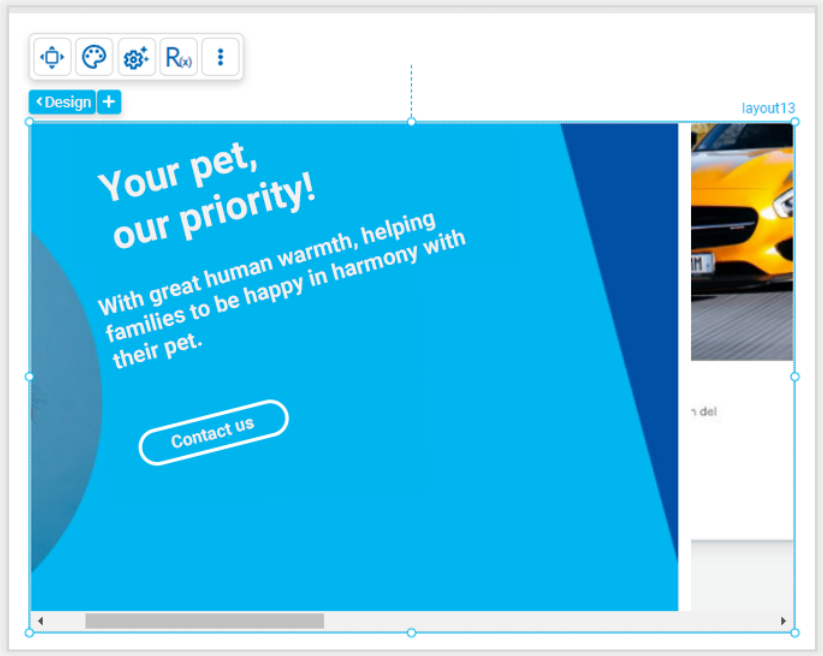
Column

It allows to define items arranged in columns. Within each column item, any type of element can be dragged.



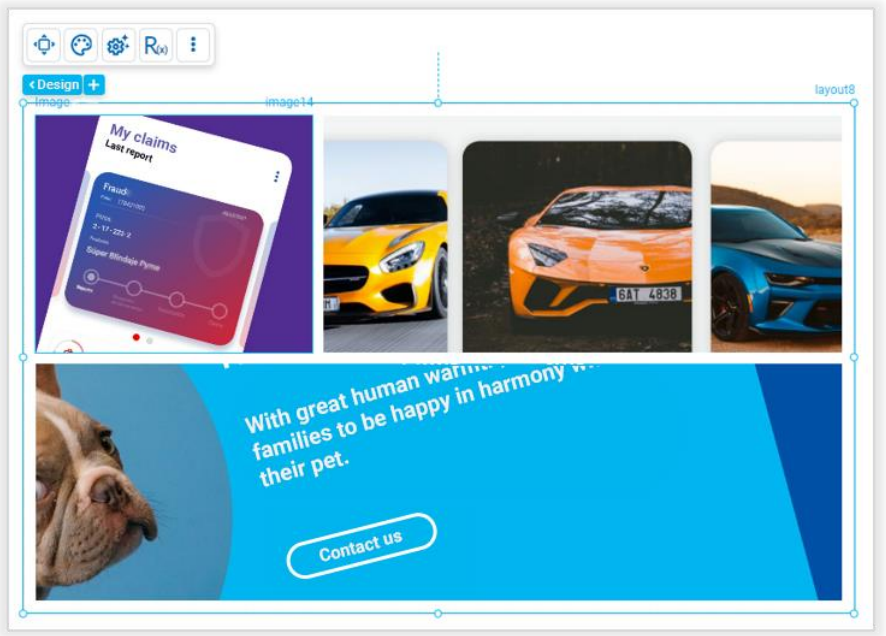
Stripe

It allows to define items arranged in a horizontal strip that covers the entire available width, incorporating a scroll bar to move. Within each stripe item, any type of element can be dragged.



Mosaic

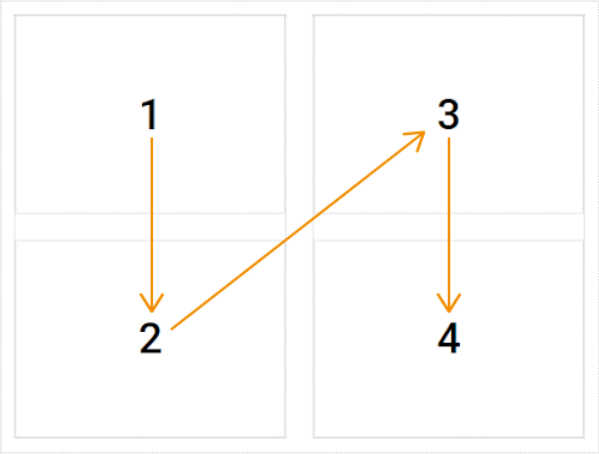
It allows to define items arranged in a grid, within each item any type of element can be dragged. Mosaics can be uniform or varied in size, depending on the desired style.



Columns and Rows

It allows organizing elements within cells, structured in columns and rows. Each cell is called a row column item and can contain any other element.

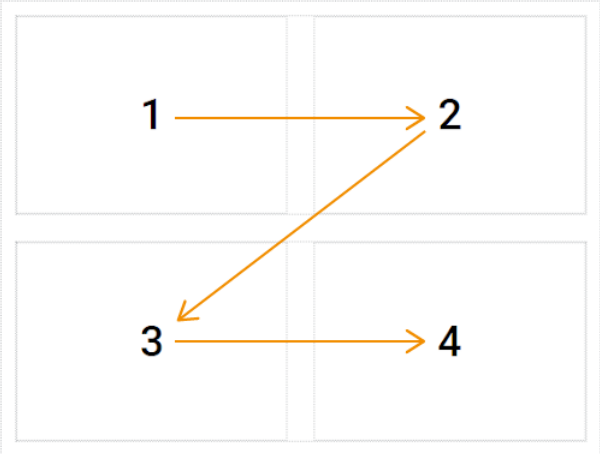
This layout subtype is especially useful when cells need to automatically adapt to the mobile breakpoint, stacking entire columns from left to right.



Rows and Columns

It allows organizing elements within cells, structured in rows and columns. Each cell is called a row column item and can contain any other element.

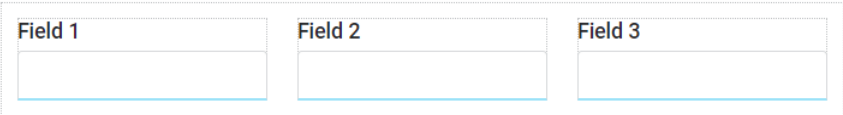
This layout subtype is especially useful when cells need to automatically adapt to the mobile breakpoint, stacking from left to right within each row.



Field Group

It allows grouping one or more fields, organizing them in a row or a column, as needed.

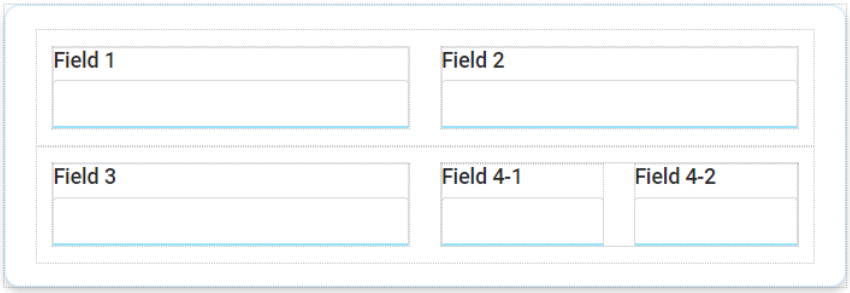
This layout subtype is especially useful when fields have to be grouped together, and easily relocated within other layouts.



Fields in Rows and Columns

It allows defining fields arranged in rows and columns. They can be modeled with one or multiple rows and with one or multiple columns. One or more fields can be defined in each cell.

This layout subtype is ideal for use directly on a section, especially when fields need to be arranged from left to right in each row, and to be arranged in vertical columns, top to bottom, across the entire screen. Additionally, in the mobile breakpoint, the layout adapts by stacking the content of each row from left to right, to ensure a clear display.



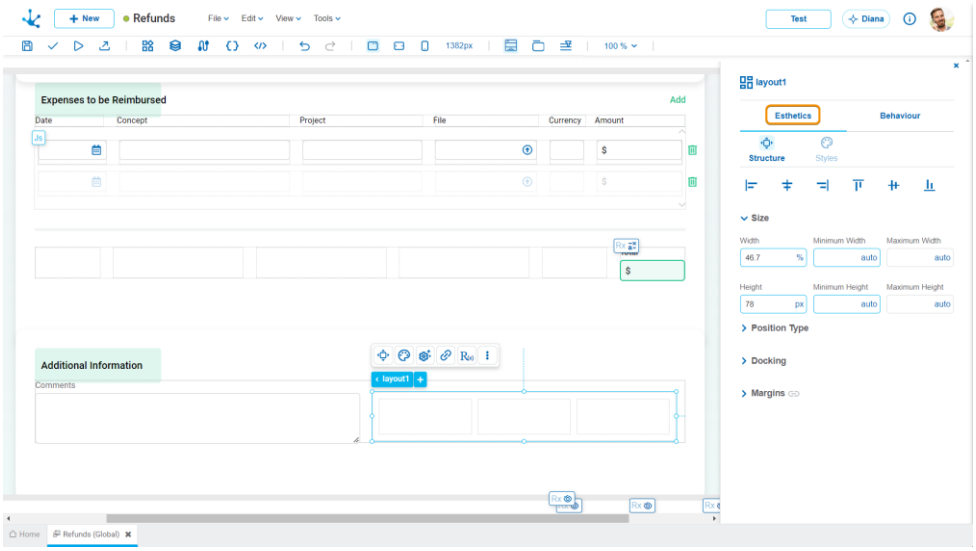
Classification of Properties

Properties are divided into two groups: esthetic properties and behavior properties.

Esthetic Properties

In the esthetic properties panel, the following are grouped:

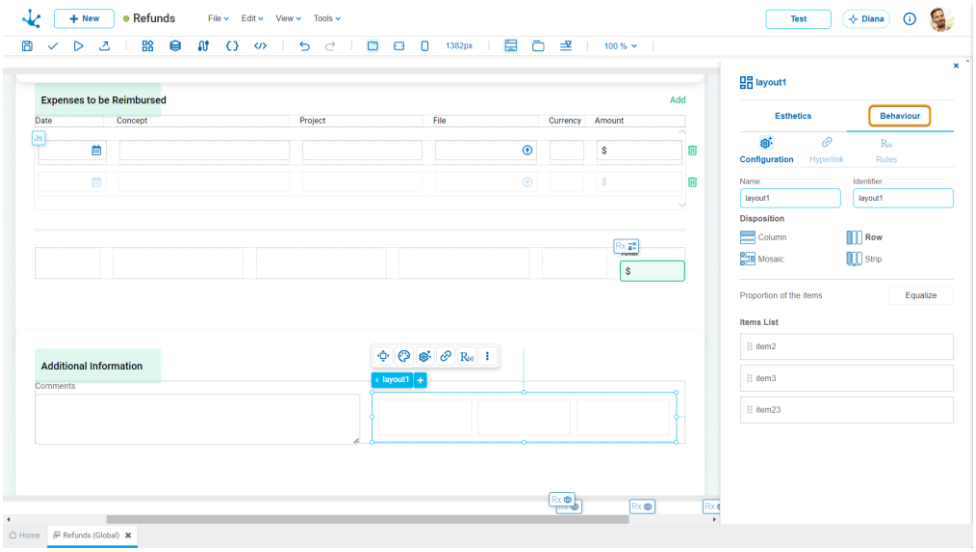
- Structure Properties
- Style Properties



Behavior Properties

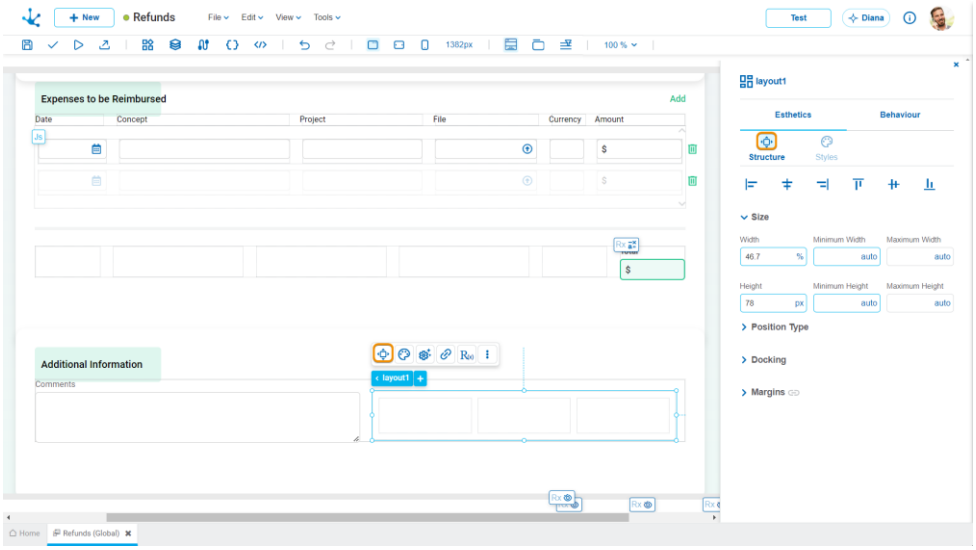
In the behavior properties panel, the following are grouped:

- Configuration Properties
- Rule Properties



Structure Properties

The structure properties panel of an element opens when selecting the icon  of its context menu.



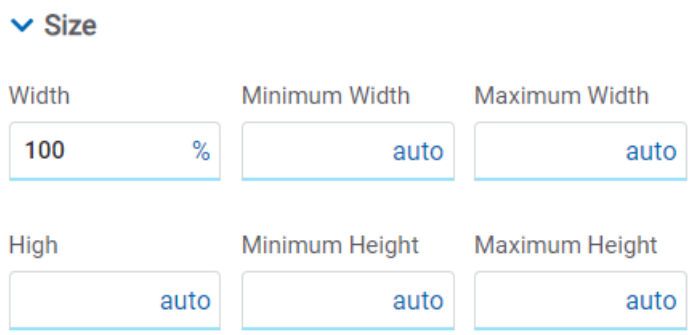
Alignment

Allows to align the element by selecting one of the available icons.

- Align to the left.
- Align to horizontal center.
- Align to the right.
- Align up.

- Align vertical center.
- Align bottom.

Size

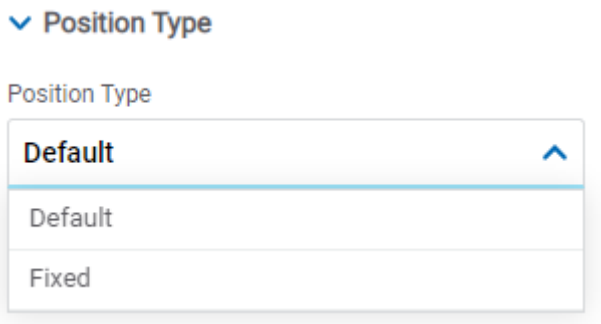


All size properties can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh). Additionally, for Width and Height properties, the "auto" option is added.

> *If selecting percentage (%) for the width and height properties, the size is calculated relative to the top element.*

Position Type

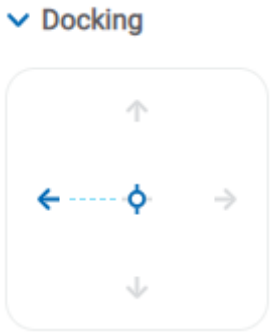
It determines if at the time of execution the element remains fixed on the page or if it moves as the scroll bar moves forward or backward.



Possible Values

- Default: The element has a relative position with respect to the superior element where it was placed (container or section).
- Fixed: Fixes the element to the page so that it is always visible in the same position. It allows to increase the visibility of important content.

Docking



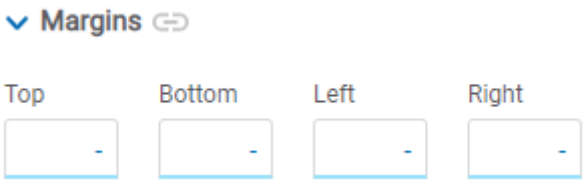
Determines the vertical and horizontal position of elements within the page section or container, when the screen is resized.

When an element is added or moved within the section or container, the modeler automatically docks it to the nearest corners or edges. It can also be configured by clicking on the arrows of the side to be docked to or on the center.

Once the element is docked, its exact position is indicated in the Margins property. The direction of the docking arrows determine the values that are automatically displayed for margins.

The docking position is indicated in the modeling area by dotted lines on the element.

Margins



It allows to define the distance of elements from the borders of their top element. The behavior of margins depends on the docking of the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Top
Distance to the top border of the highest ranking element.

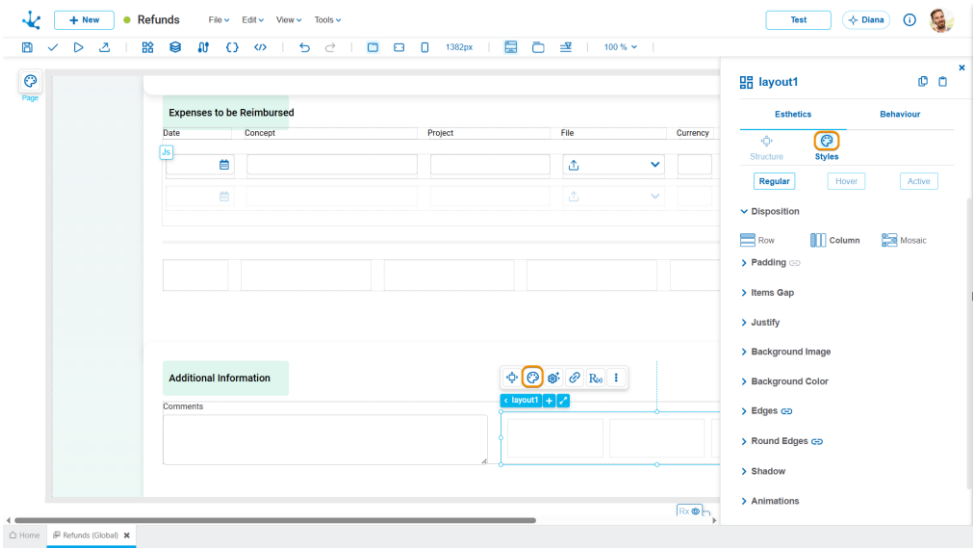
Bottom
Distance to the bottom border of the highest ranking element.

Left
Distance to the left border of the highest ranking element.

Right
Distance to the right border of the highest ranking element.

Allows the value entered in one of the margins to be copied to the other ones automatically.
Allows to indicate different values for each margin.

The style properties panel of an element opens when selecting the icon of its context menu.



This type of element may take different states and for each of them different values for its properties may be modeled.



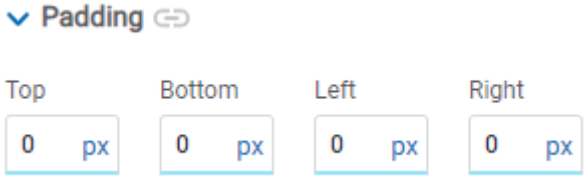
- Regular: The mouse pointer is not over the element.
- Hover: The mouse pointer is over the element.

Styles

It allows copying the entire current configuration of the style panel for the selected element (Ctrl+Alt+C).

It allows applying the previously copied configuration to another element of the same type (Ctrl+Alt+V).

Padding

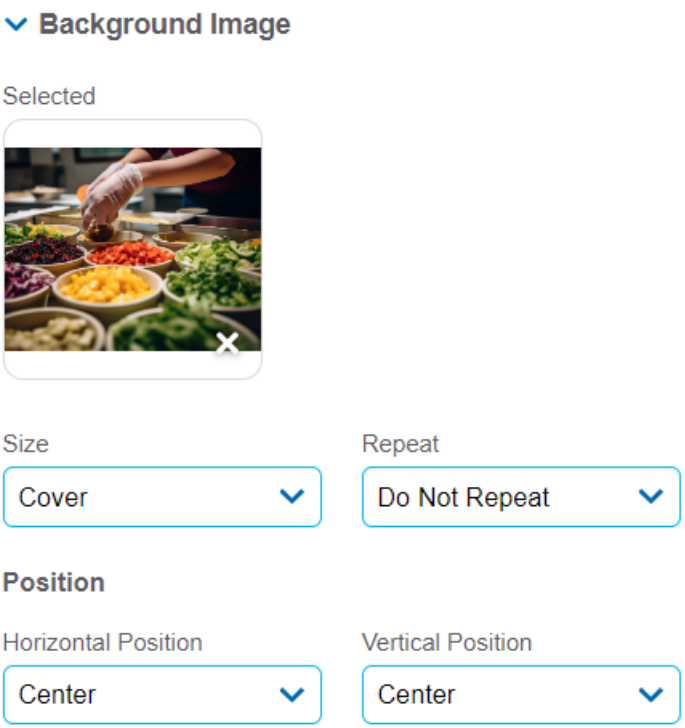


All padding space properties allow to create a space around the element borders (top, bottom, sides) and its content. The padding space configured is inward, while the margin is configured relative to a top element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Allows values entered in one of the paddings to be copied to the other ones automatically.
Allows to indicate different values for each padding.

Items Gap
It defines the spacing between the items in the layout.

Background Image



Allows to add a background image to the element.

Selected
An image can be uploaded from the computer where it is being modeled.

Size

Possible Values

- Cover
- Content
- Auto

Repeat

Possible Values

- Repeat
- Repeat Hor.
- Repeat Vert.
- Spacing
- Rounded
- Do Not Repeat

Horizontal Position

Possible Values

- Center
- Left
- Right

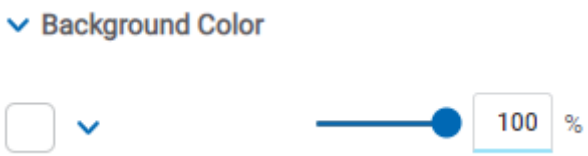
Vertical Position

- Center
- Top
- Botton

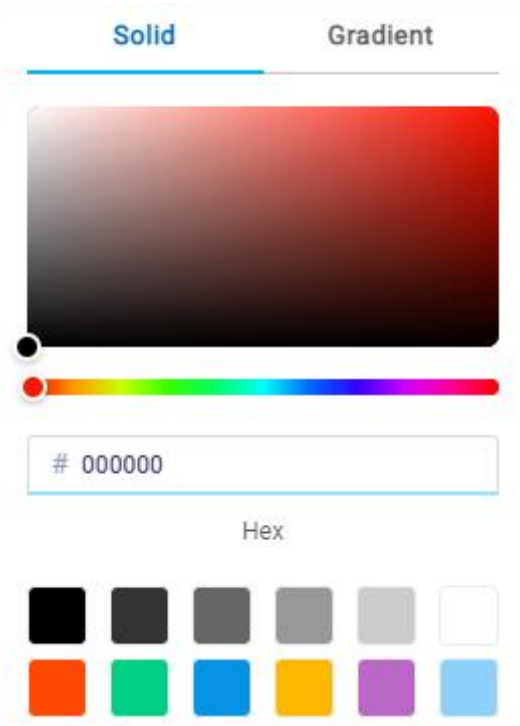
Background Color

It allows to model a background color for the element, it can be solid or gradient and different properties are defined for each one.

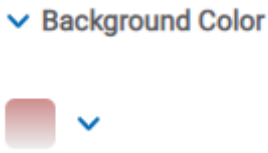
Solid



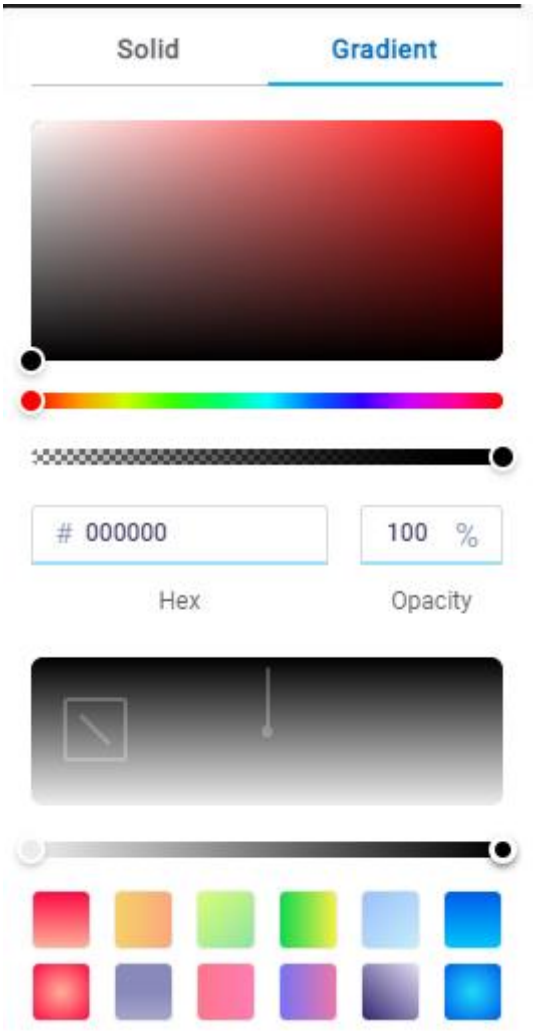
This option enables not only to select the color from a palette but also to define the degree of transparency.



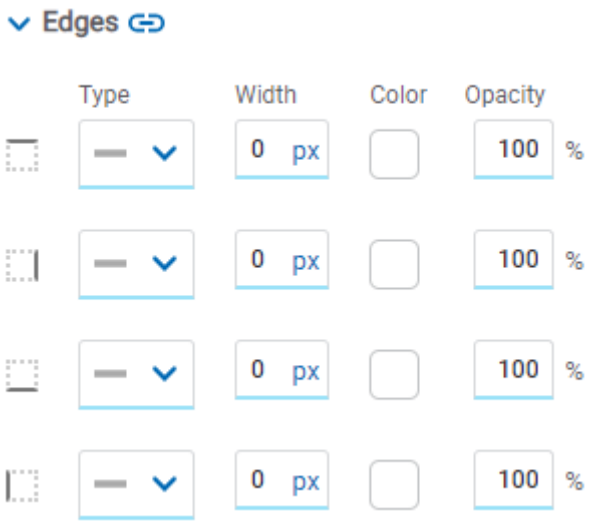
Gradient



This option enables to select not only the color from a palette but also to define its opacity and intensity according to the selected angle.



Edges



It allows to define the style of borders. Each one has its type, width, color and opacity defined, the latter as a percentage.

Allows the value entered in one of the borders to be copied to the other ones automatically.

Allows to indicate different values for each border.

Round Edges

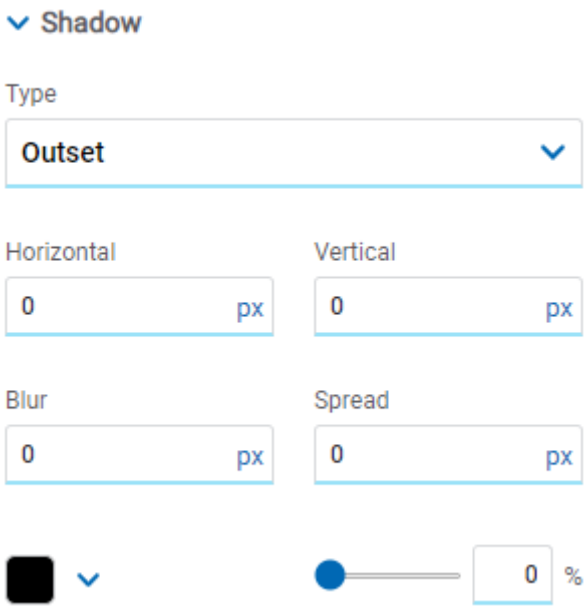


It allows to define the round edges at the corners of elements.
It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Allows the value entered in one of the borders to be copied to the other ones automatically.

Allows to indicate different values for each border.

Shadow



Allows to define a shadow effect around the element.

Horizontal
Horizontal size of the shadow to the right of the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Vertical
Vertical size of the shadow below the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

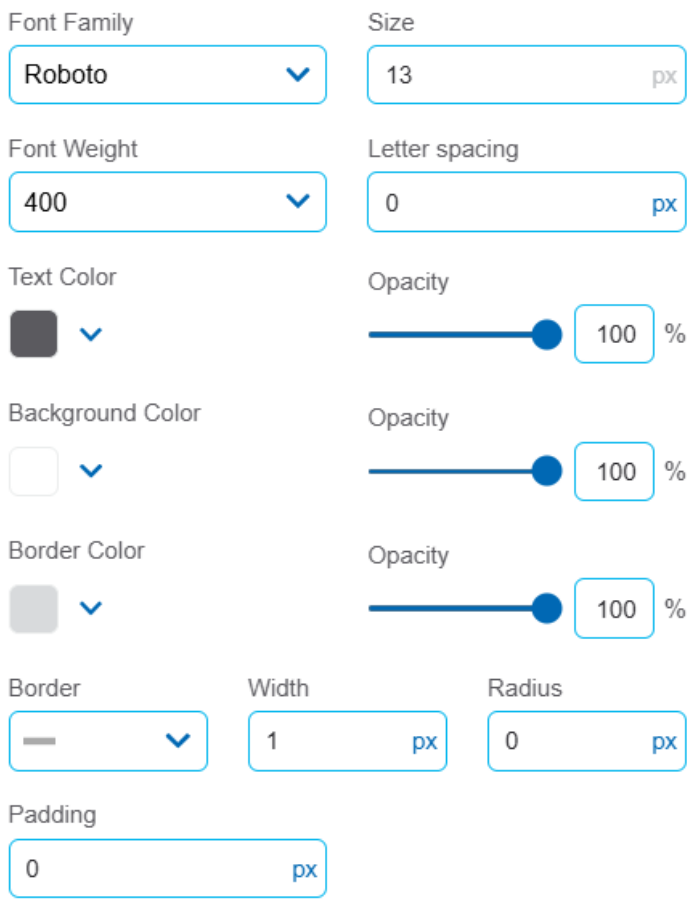
Blur
The greater this value is, the greater and lighter the shadow becomes. If not specified, its value is 0 and the shadow border is darker. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Spread
Positive values will cause the shadow to expand and grow bigger, while negative values will cause the shadow to shrink. If not specified, its value is 0 and the shadow will have the same size as the element.

Color
Defines the color of the shadow.

Transparency
Defines the transparency of the shadow.

Tooltip



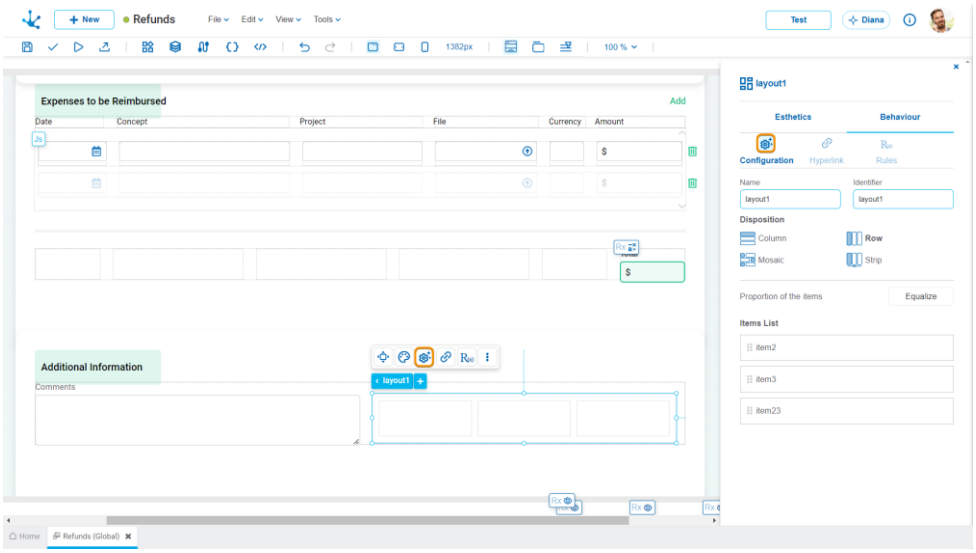
Source
It allows customizing the font, size, thickness and letter spacing of the pop-up message.

Colors
The color and opacity of the text, background and border can be defined. Borders are set uniformly for all sides of the tooltip.

Border
The border type, width, and border radius can be configured.

Configuration Properties

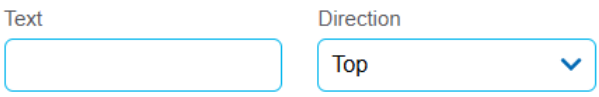
The configuration properties panel of an element opens when clicking the icon  of its context menu.



Name
Name used to reference the element during modeling.

Identifier
Uniquely identifies the element. It is used in the Javascript SDK.

Tooltip
Pop-up message that appears when hovering over an element, providing additional information.

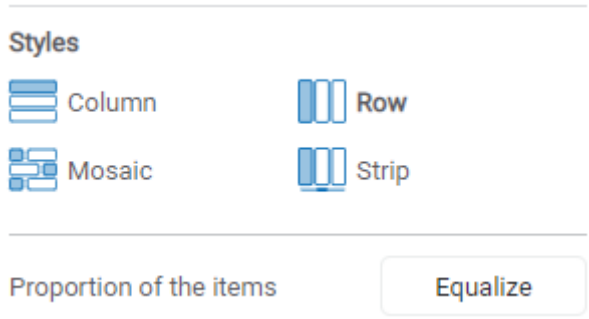


Text
Content of the message being displayed.

Direction
Position of the message in relation to the element (top, bottom, left, or right).

Styles

Indicates the way in which items are grouped within the layout.



Column
Items are placed vertically within the layout.

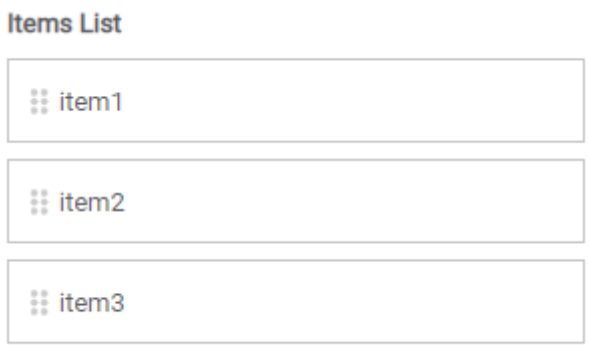
Row
Items are placed horizontally within the layout.

Mosaic
Combines the vertical and horizontal placement of items based on their size within the layout.

Strip
Items are placed horizontally within the layout. Allows scrolling.

Items Proportion
Pressing the “Equalize” button adjusts the size of the items proportionally within the layout.

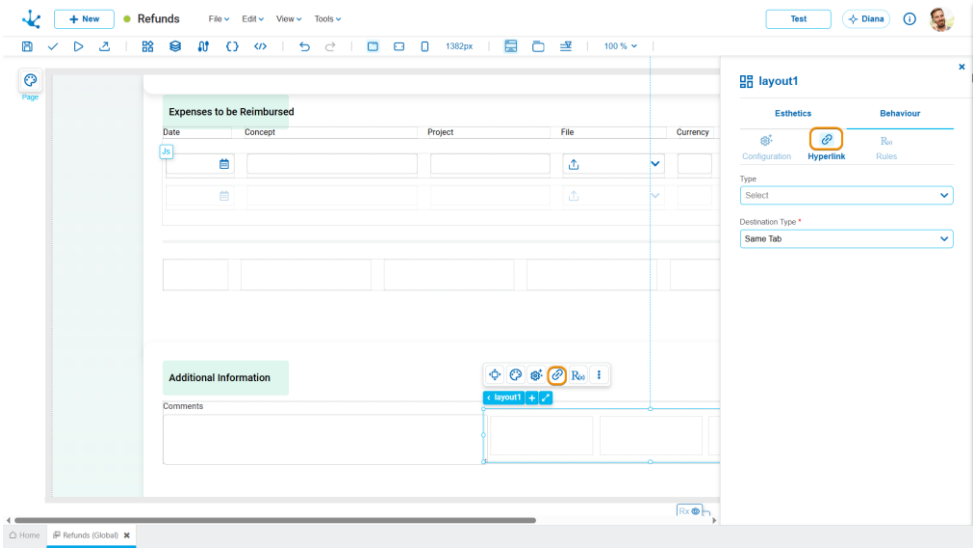
Items List



The items contained in the layout are displayed, and their position can be changed by dragging them within the list.

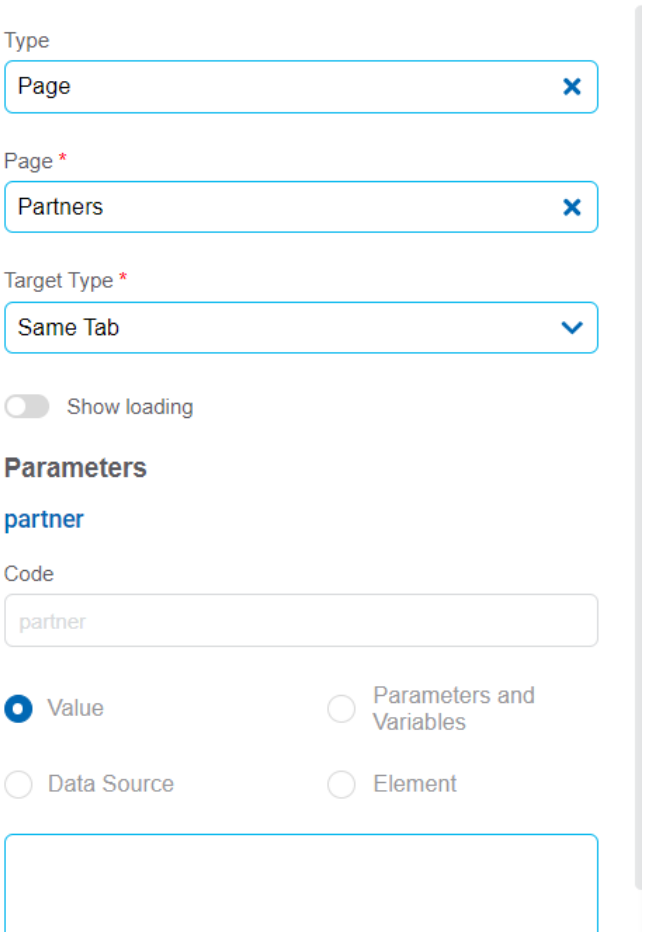
Hyperlink Properties

The hyperlink properties panel of an element opens when clicking the icon of its context menu.



Define the behavior of the element when selecting it. Different properties are enabled depending on the type of object selected.

Page



Page
The pages modeled in the environment are displayed.

Destination Type
The available options for opening the object are displayed.

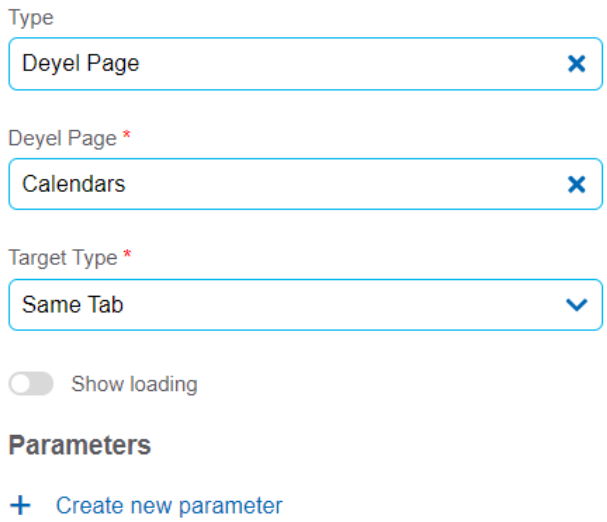
Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  - Modal Horizontal Size
    Defines its width.
  - Modal Vertical Size
    Defines its height.
- Iframe: selecting this option enables an additional property.
  - Destination Iframe
    Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Parameters
If there are parameters, their value can be specified with text, parameters and variables or data source.

Deyel Page



Deyel Page
The pages belonging to **Deyel** are displayed.

Destination Type
The available options for opening the object are displayed.

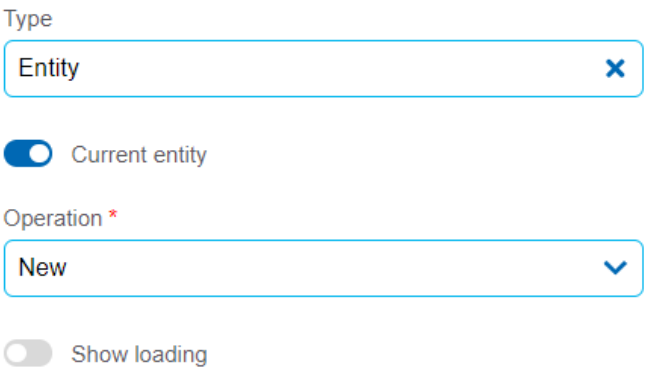
Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  - Modal Horizontal Size
    Defines its width.
  - Modal Vertical Size
    Defines its height.
- Iframe: selecting this option enables an additional property.
  - Destination Iframe
    Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Parameters
Allows sending parameters to the selected object type.

Entity



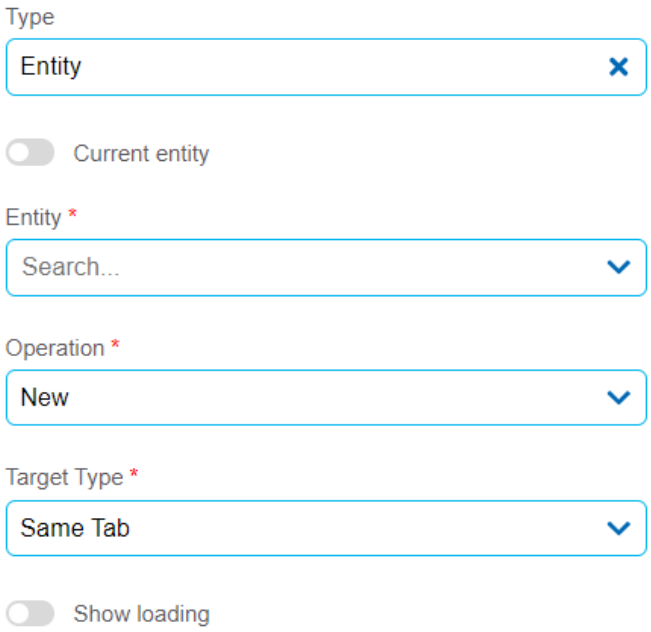
Current Entity
If this property is checked, the operation applies directly to the entity being modeled.

Operation
Defines the operation made when selecting the object.

Possible Values

- Create: Indicates that the panel of the current entity is opened to create an instance.
- Update: Indicates that the panel of the current entity is opened to update an instance.
- Delete: Indicates that the panel of the current entity is opened to delete an instance.

- Send Email: It is visible if the property Send Emails is enabled. Clicking on the modeled element opens a panel during instance execution to send an email. The panel size is adjusted with the properties Modal Width and Modal Height.
- Emails: Visible if the Send Emails property is enabled. Clicking on the modeled element opens a panel during instance execution to manage sent emails. The panel is displayed in the iframe defined in the Target Iframe property.



Entity
If the property Current Entity, is not checked, the entities modeled in the environment are displayed.

Operation
Defines the operation made when selecting the object.

Possible Values

- New: The panel of the entity selected in the previous property is opened to create an instance.
- Update The panel of the entity selected in the previous property is opened to update an instance.
- Show: The panel of the entity selected in the previous property is opened to show an instance.
- Grid: Indicates that the grid of the entity selected in the previous property is opened.

Destination Type
The available options for opening the object are displayed.

Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  Modal Horizontal Size
  Defines its width.

Modal Vertical Size
Defines its height.

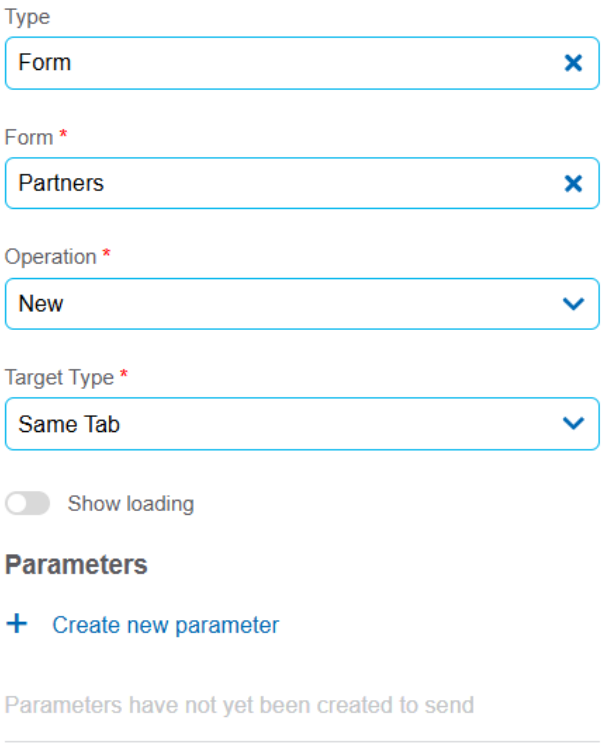
- Iframe: selecting this option enables an additional property.

Destination Iframe
Expands the iframes previously defined on the object.

- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Forms



Forms
The forms modeled in the environment are displayed.

Operation
Defines the operation made when selecting the object.

Possible Values

- Create: Indicates that the panel of the object selected in the previous property is opened for the creation of an instance.
- Update: Indicates that the panel of the object selected in the previous property is opened for the update of an instance.
- Show: Indicates that the panel of the object selected in the previous property is opened for the query of an instance.

- Grid: Indicates that the grid of the object selected in the previous property is opened.

Destination Type
The available options for opening the object are displayed.

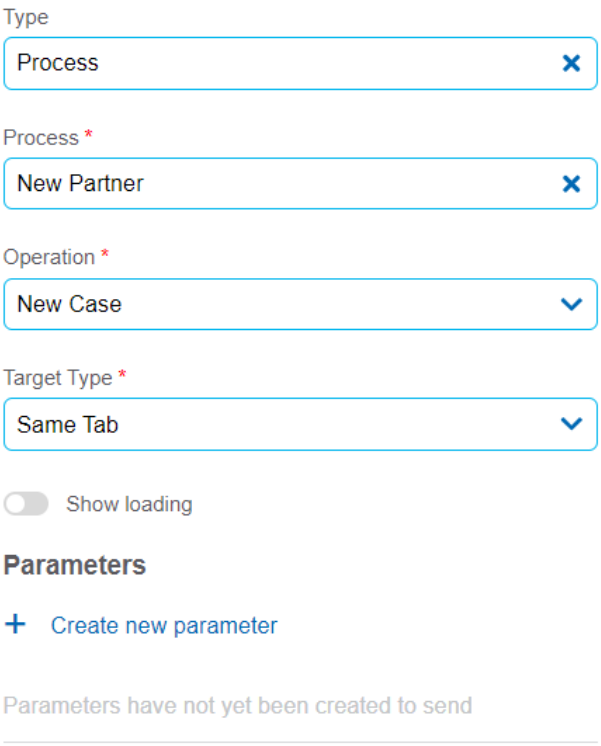
Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  Modal Horizontal Size
  Defines its width.
  Modal Vertical Size
  Defines its height.
- Iframe: selecting this option enables an additional property.
  Destination Iframe
  Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Parameters
Allows sending parameters to the selected object type.

Process



Process
The processes modeled in the environment are displayed.

Operation
Defines the operation made when selecting the object.

Destination Type
The available options for opening the object are displayed.

Possible Values

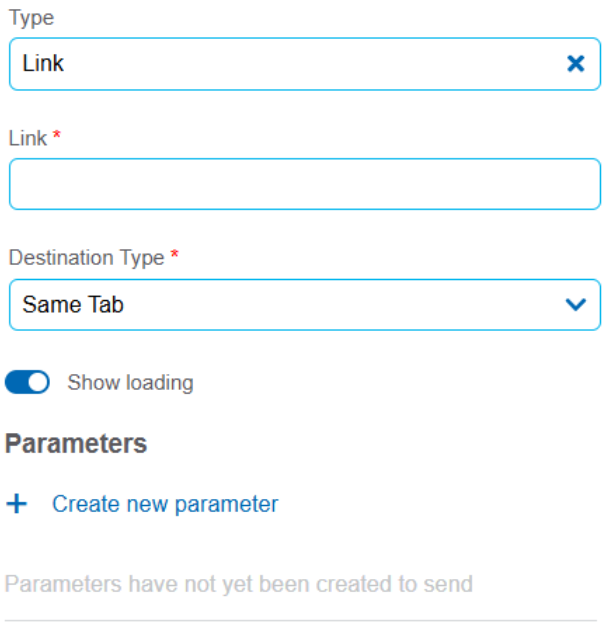
- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  - Modal Horizontal Size
    Defines its width.
  - Modal Vertical Size
    Defines its height.
- Iframe: selecting this option enables an additional property.
  - Destination Iframe
    Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Parameters

Allows sending parameters to the selected object type.

Link



Link
Allows to enter any link.

Destination Type
The available options for opening the object are displayed.

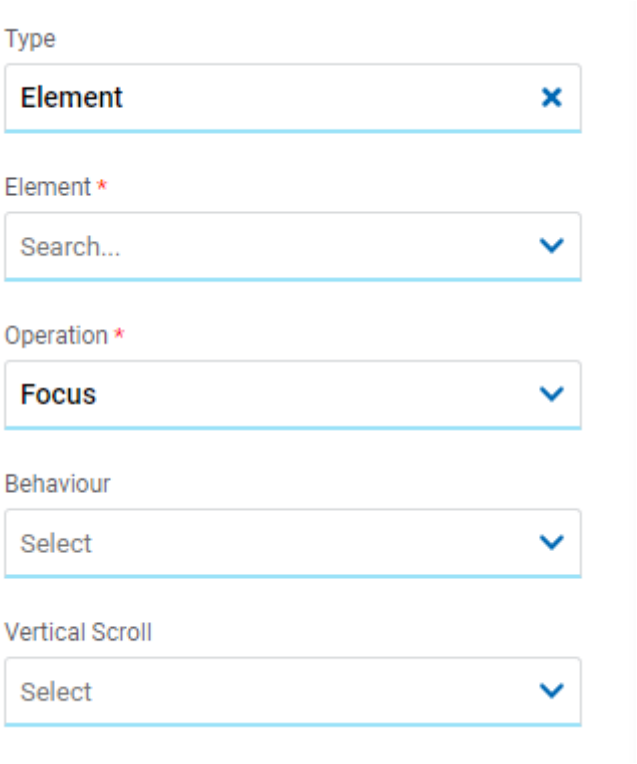
Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  Modal Horizontal Size
  Defines its width.
  Modal Vertical Size
  Defines its height.
- Iframe: selecting this option enables an additional property.
  Destination Iframe
  Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Parameters
Allows sending parameters to the selected object type.

Element



Element
The modeled elements in the object are displayed.

Operation
Defines the operation made when selecting the element.

Possible Values

- Focus
- Show
- Hide
- Show/Hide

Behaviour
Establishes the transition of the animation. It is only modelable if the "Focus" operation was selected.

Possible Values

- Auto
- Smooth

Vertical Scroll
It is only modelable if the "Focus" operation was selected.
Possible Values

- Start: Moves to the start of the selected element.
- Closest: Moves to the closest position to the selected element from the element the event fires.

- Center: Moves to the center of the selected element.
- End: Moves to the end of the selected element.

Repeater



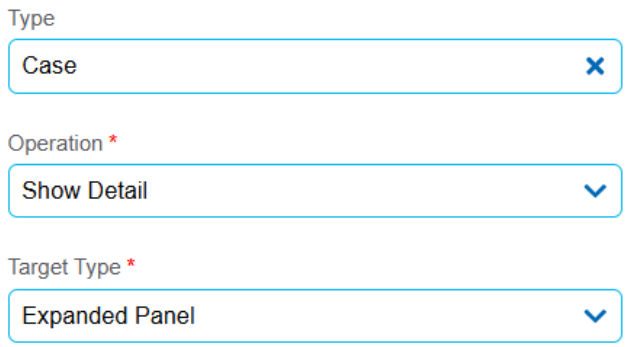
Repeater
The modeled repeaters from the object are displayed.

Operation
Defines the operation made when selecting the element.

Possible Values

- New: Allows for the creation of an item within the repeater. If this option is selected the additional properties are enabled.
  ### Creation Order
  It indicates in which position the new item of the container is added.
  Possible Values
  - Home
  - End
- Delete: Allows for the deletion of an item within the repeater.

Case



Case

Show case information.

Operation
Defines the operation made when selecting the element.

Possible Values

- Show Detail
- Show Graphic Execution

Destination Type
The available options for opening the object are displayed.

Possible Values

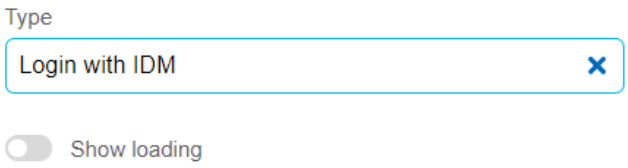
- Expanded Panel
- Modal: if this option is selected the additional properties are enabled.
  Modal Horizontal Size
  Define its width.
  Modal Vertical Size
  Define its height.

Back



It allows associating the event to go back in the browser to the element.

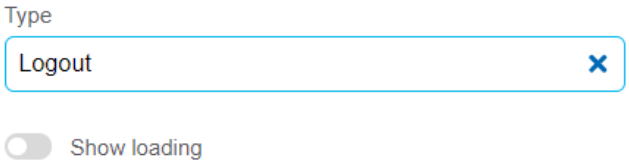
LogIn with IDM



Allows login with IDM.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

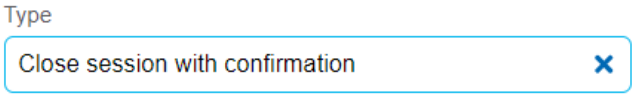
Logout



Allows logging the user out.

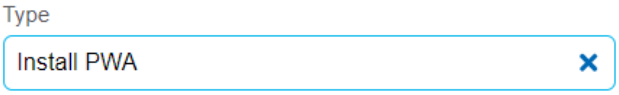
Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Close Session with Confirmation



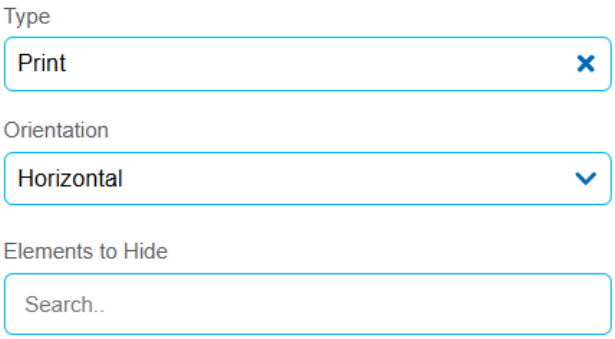
Displays a confirmation modal to logout the user.

Install PWA



Allows to install the application in the browser.

Print



This type of hyperlink associated with an element allows printing the content of the entity or page that is being executed.

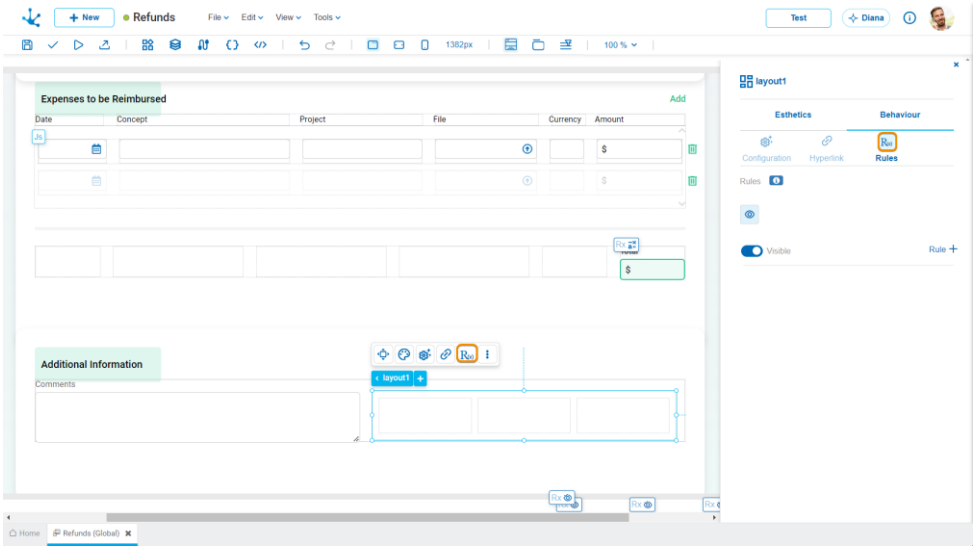
Orientation

Determines the layout of the content to be printed.
Possible Values
- Horizontal
- Vertical

Elements to hide
It allows selecting which elements of the entity or page should not be included in the printout.

Rules Properties

The rule properties panel of an element opens when clicking the icon Rx of its context menu.



Properties

Rules
Embedded rules on behavior, validation and calculation can be defined, associated with an element, by using the wizard (ctrl + space).
Shows syntax examples for writing the rules.

Visible

Indicates whether the element is visible. If this property is not checked, the element is not displayed in the page.
Visible (default) Not visible
Rule + Opens an edit area where a rule to determine the visibility condition can be defined. If a rule is defined, the icon is displayed with light blue borders.

Operations when defining the rule:

Saves the new or modified rule

Cancels the operation

Operations once the rule is defined:

Edits the existing rule

Deletes the rule

Item

Items can be included to the layout and their features are defined by modeling their properties.

Operations

Add

It allows adding an item to the layout by clicking on the icon from the element's context menu.

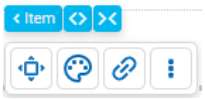


Move

The position of an item is changed in the layout configuration properties panel.

Change Size

The size of an item can be modified not only from its context menu, using the icons to increase its size and to reduce it, but also from its structure properties panel.



Context Menu

Clicking the right button of the mouse on an item expands a second context menu whose options correspond to operations that can be performed on the selected item. The same menu is displayed by pressing the icon from the palette.

- Cut (Ctrl+X)
- Copy (Ctrl+C)
- Paste (Ctrl+V)
- Duplicate (Ctrl+D)
- Lock/Unlock When an item is locked, it cannot be moved until it is unlocked.
- Hide: Hides the item in the breakpoint that is being used and in minor breakpoints. Once hidden, it can be shown again from Layers.

- Delete (Del)
- Properties

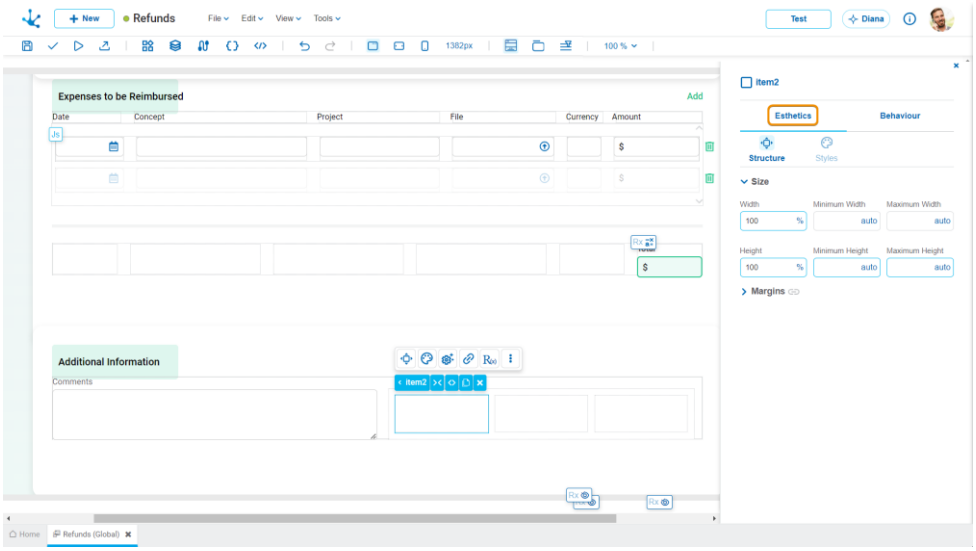
Classification of Properties

Properties are divided into two groups: esthetic properties and behavior properties.

Esthetic Properties

In the esthetic properties panel, the following are grouped:

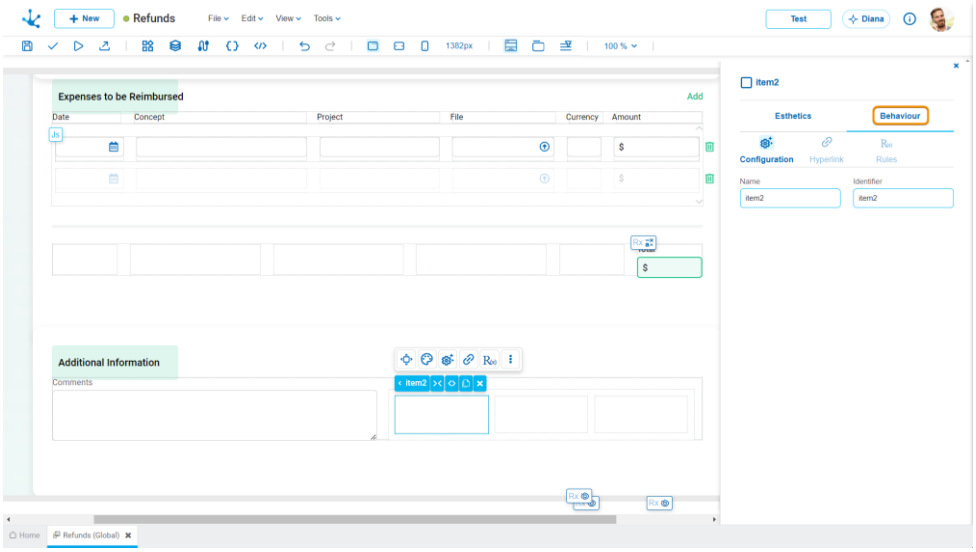
- Structure Properties
- Style Properties



Behavior Properties

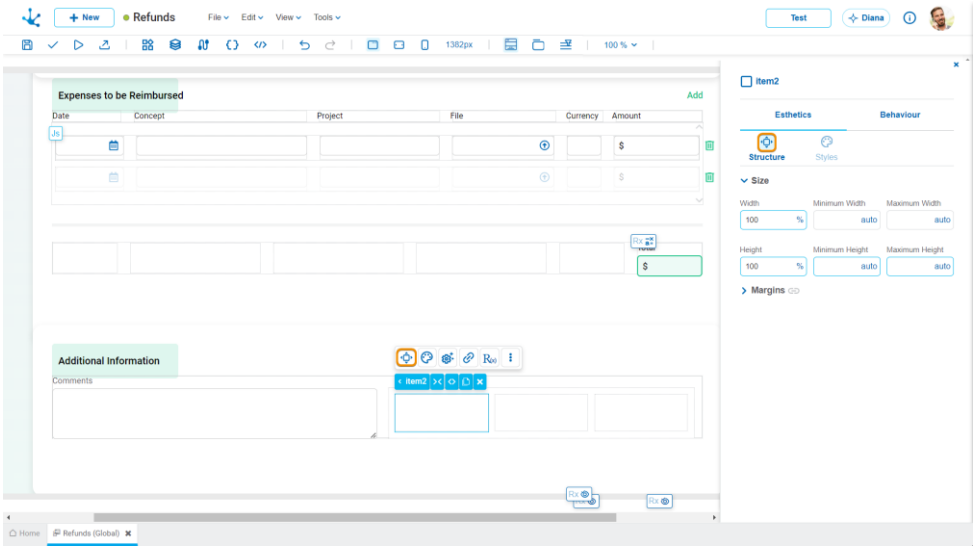
In the behavior properties panel, the following are grouped:

- Configuration Properties
- Hyperlink Properties
- Rule Properties

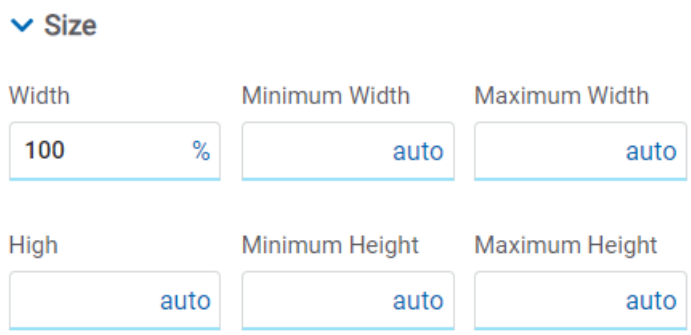


Structure Properties

The structure properties panel of an element opens when selecting the icon  of its context menu.



Size



All size properties can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh). Additionally, for Width and Height properties, the "auto" option is added.

> *If selecting percentage (%) for the width and height properties, the size is calculated relative to the top element.*

Margins



Allows to define the distance among items and also from items to the borders of the containing element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Top
Distance to the top border of the containing element.

Bottom
Distance to the bottom border of the containing element.

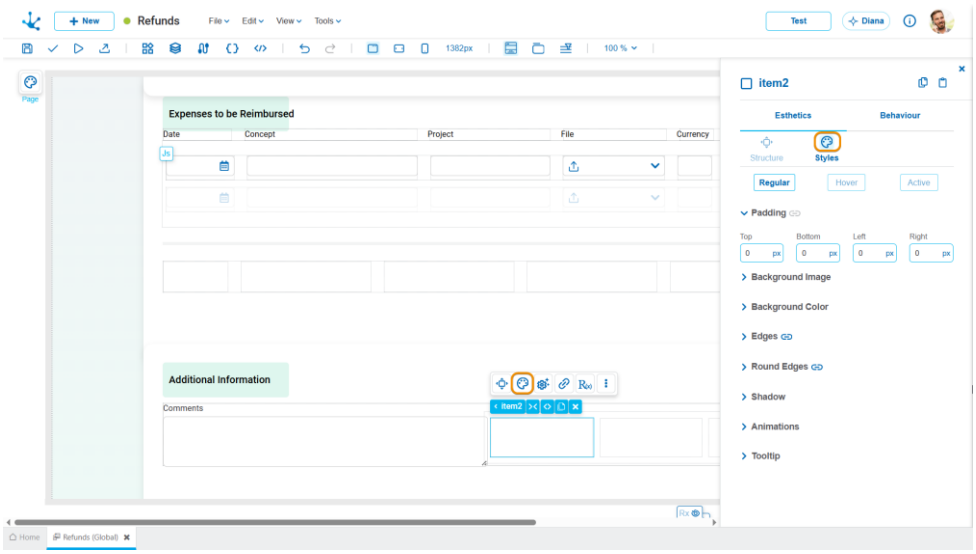
Left
Distance to the left border of the containing element or to the previous item.

Right
Distance to the right border of the containing element or to the following item.

Allows the value entered in one of the margins to be copied to the other ones automatically.
Allows to indicate different values for each margin.

Style Properties

This type of element may take different states and for each of them different values for its properties may be modeled.





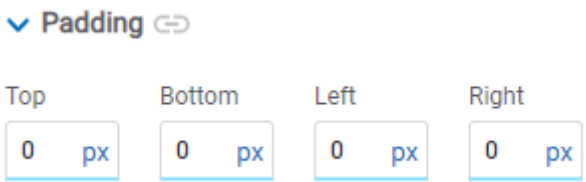
- Regular: The mouse pointer is not over the element.
- Hover: The mouse pointer is over the element.

Styles

It allows copying the entire current configuration of the style panel for the selected element (Ctrl+Alt+C).

It allows applying the previously copied configuration to another element of the same type (Ctrl+Alt+V).

Padding



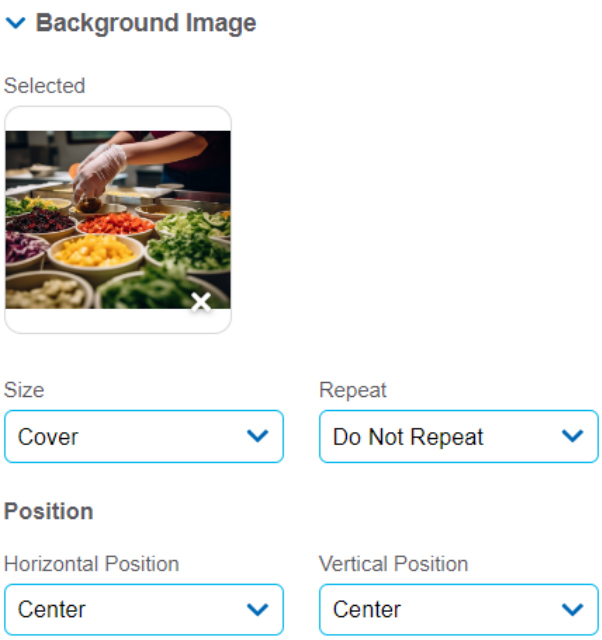
All padding space properties allow to create a space around the item borders (top, bottom, sides) and its content. The padding space configured is inward, while the margin is configured relative to a top item. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Allows values entered in one of the paddings to be copied to the other ones automatically.

Allows to indicate different values for each padding.

Background Image

It allows the addition of a background image to the item.

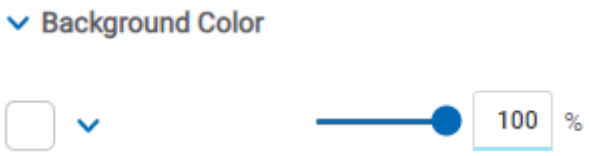


Selected
An image can be uploaded from the computer where it is being modeled.

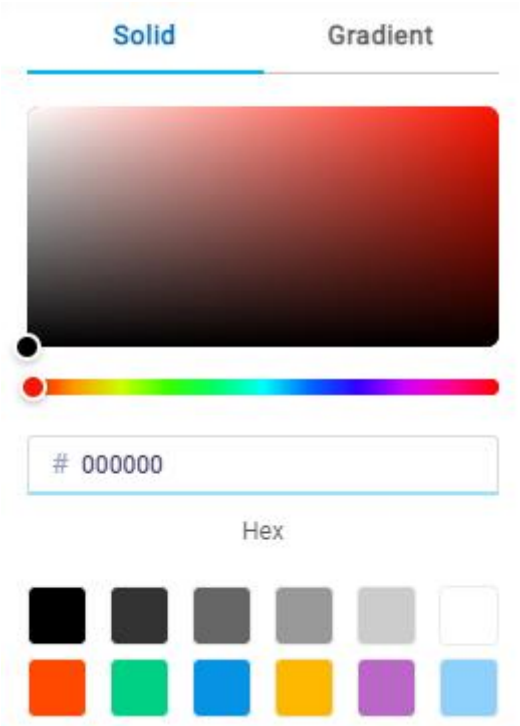
Background Color

It allows to model a background color for the element, it can be solid or gradient and different properties are defined for each one.

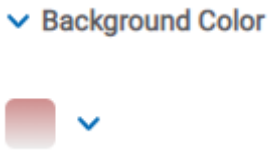
Solid



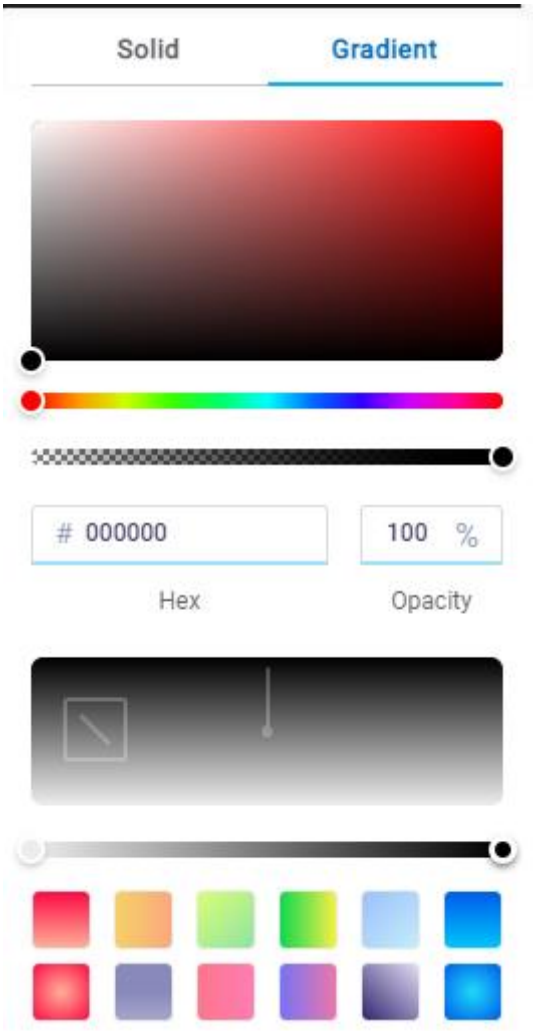
This option enables not only to select the color from a palette but also to define the degree of transparency.



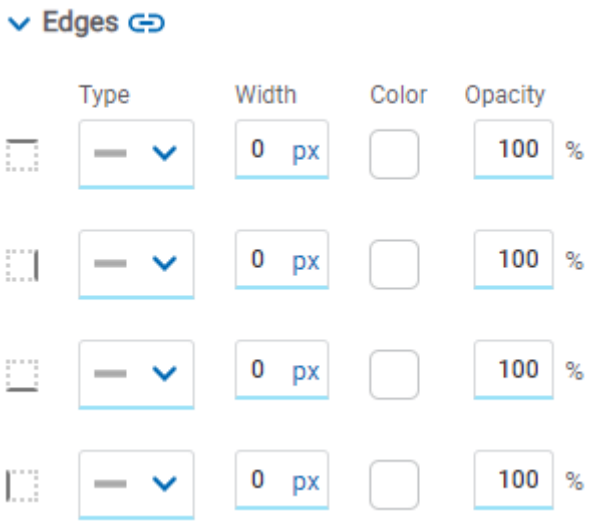
Gradient



This option enables to select not only the color from a palette but also to define its opacity and intensity according to the selected angle.



Edges



It allows to define the style of borders. Each one has its type, width, color and opacity defined, the latter as a percentage.

Allows the value entered in one of the borders to be copied to the other ones automatically.
Allows to indicate different values for each border.

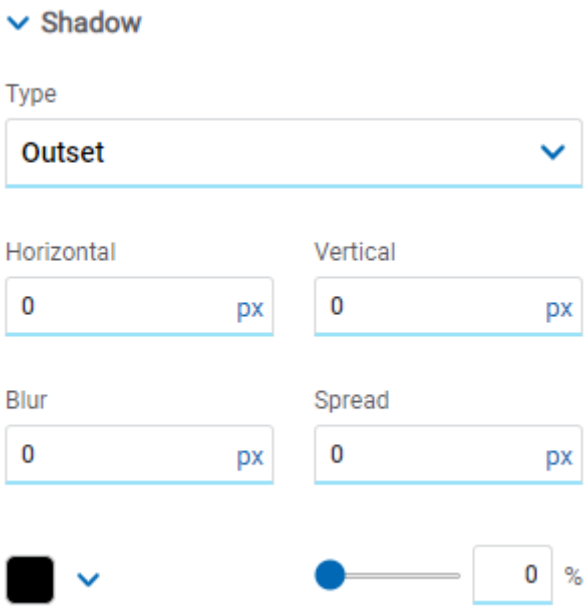
Round Edges



It allows to define the round edges at the corners of items.
It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Allows the value entered in one of the borders to be copied to the other ones automatically.
Allows to indicate different values for each border.

Shadow



Allows to define a shadow effect around the item.

Type

Possible Values

- Outset
- Inset

Horizontal
Horizontal size of the shadow to the right of the item. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Vertical
Vertical size of the shadow below the item. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

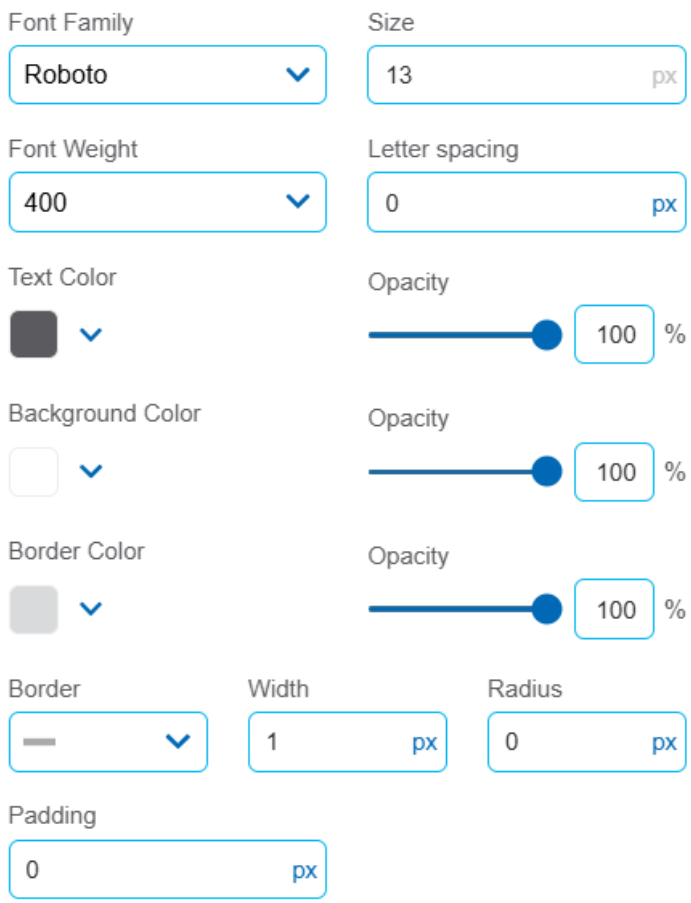
Blur
The greater this value is, the greater and lighter the shadow becomes. If not specified, its value is 0 and the shadow border is darker. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Spread
Positive values will cause the shadow to expand and grow bigger, while negative values will cause the shadow to shrink. If not specified, its value is 0 and the shadow will have the same size as the item.

Color
Defines the color of the shadow.

Transparency
Defines the transparency of the shadow.

Tooltip



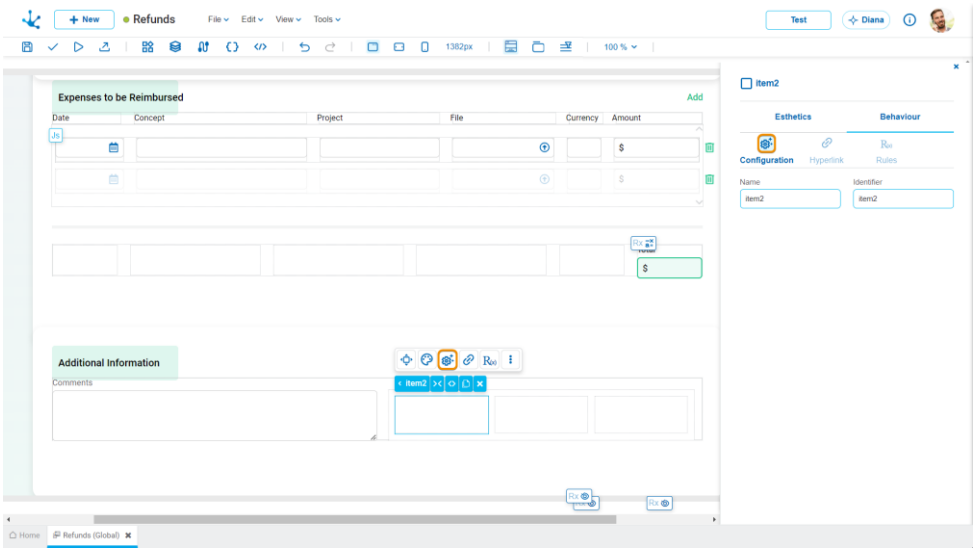
Source
It allows customizing the font, size, thickness and letter spacing of the pop-up message.

Colors
The color and opacity of the text, background and border can be defined. Borders are set uniformly for all sides of the tooltip.

Border
The border type, width, and border radius can be configured.

Configuration Properties

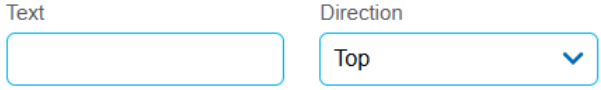
The configuration properties panel of an element opens when clicking the icon  of its context menu.



Name
Name used to reference the element during modeling.

Identifier
Uniquely identifies the element. It is used in the Javascript SDK.

Tooltip
Pop-up message that appears when hovering over an element, providing additional information.



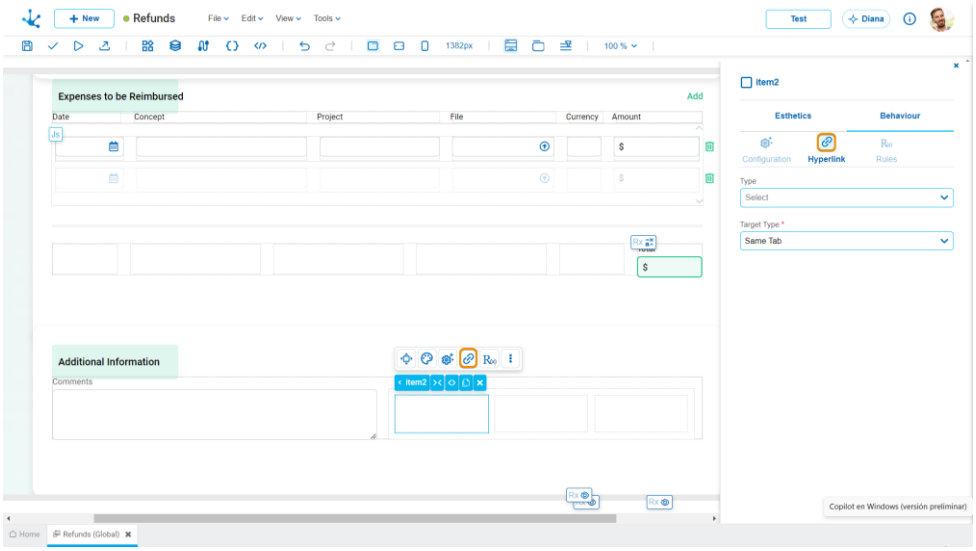
Text
Content of the message being displayed.

Direction

Position of the message in relation to the element (top, bottom, left, or right).

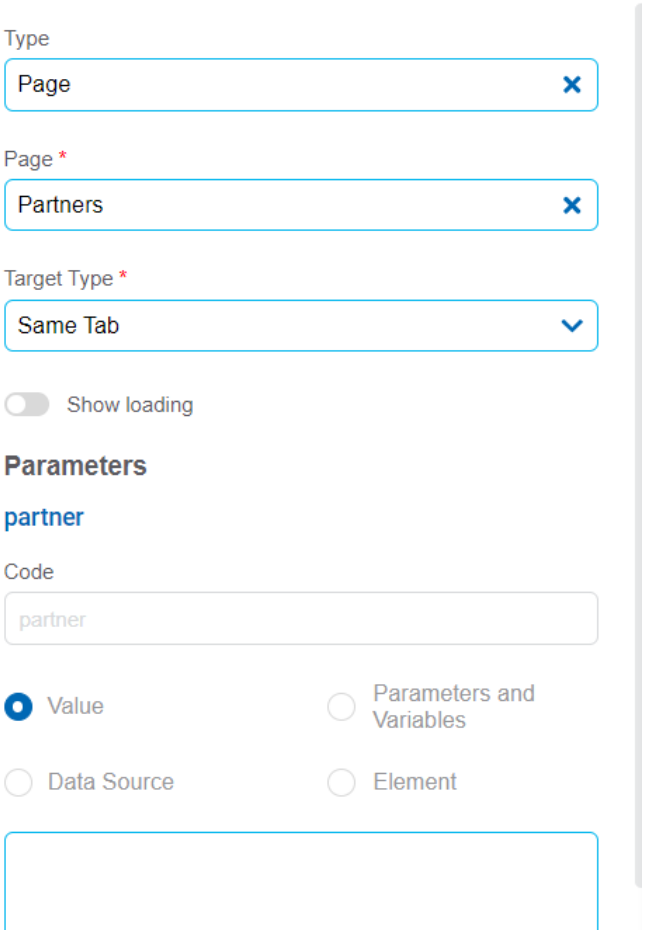
Hyperlink Properties

The hyperlink properties panel of an element opens when clicking the icon  of its context menu.



Define the behavior of the item when selecting it. Different properties are enabled depending on the type of object selected.

Page



Page
The pages modeled in the environment are displayed.

Destination Type
The available options for opening the object are displayed.

Possible Values

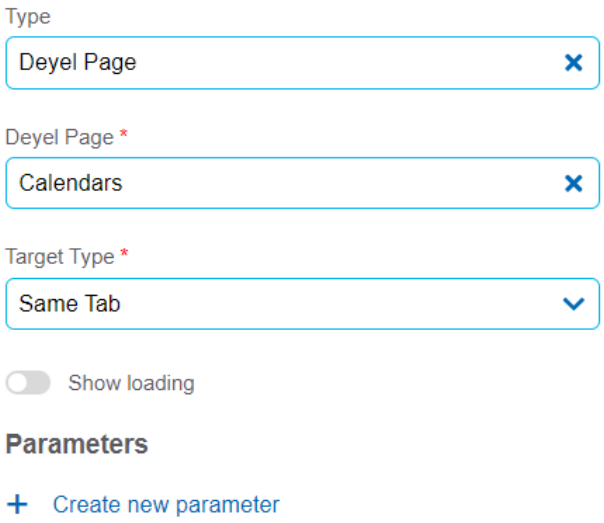
- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  Modal Horizontal Size
  Defines its width.
  Modal Vertical Size
  Defines its height.
- Iframe: selecting this option enables an additional property.
  Destination Iframe
  Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading

Shows the loading icon and disables the user interaction until the redirection is over.

Parameters
If there are parameters, their value can be specified with text, parameters and variables or data source.

Deyel Page



Deyel Page
The pages belonging to **Deyel** are displayed.

Destination Type
The available options for opening the object are displayed.

Possible Values

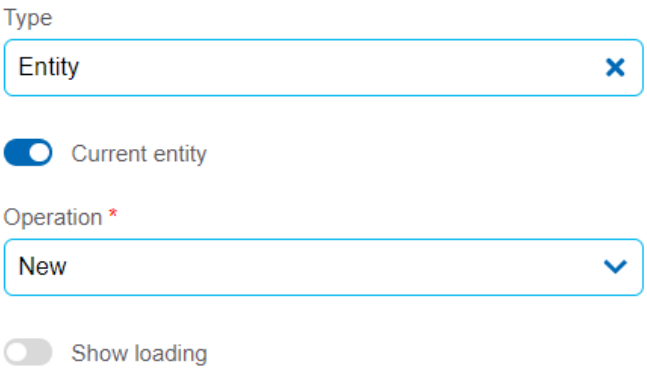
- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  - Modal Horizontal Size
    Defines its width.
  - Modal Vertical Size
    Defines its height.
- Iframe: selecting this option enables an additional property.
  - Destination Iframe
    Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Parameters

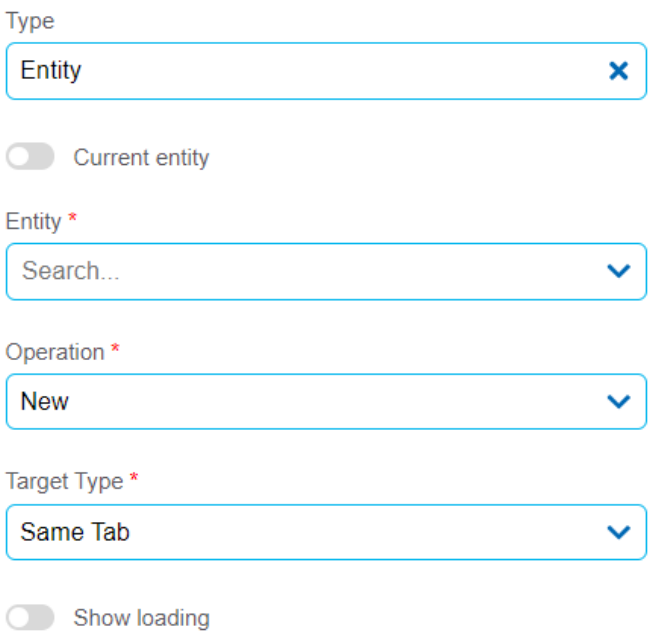
Allows sending parameters to the selected object type.

Entity



Current Entity
If this property is checked, the operation applies directly to the entity being modeled.

Operation
Defines the operation made when selecting the object.
Possible Values
- Create: Indicates that the panel of the current entity is opened to create an instance.
- Update: Indicates that the panel of the current entity is opened to update an instance.
- Delete: Indicates that the panel of the current entity is opened to delete an instance.
- Send Email: It is visible if the property Send Emails is enabled. Clicking on the modeled element opens a panel during instance execution to send an email. The panel size is adjusted with the properties Modal Width and Modal Height.
- Emails: Visible if the Send Emails property is enabled. Clicking on the modeled element opens a panel during instance execution to manage sent emails. The panel is displayed in the iframe defined in the Target Iframe property.



Entity
If the property Current Entity, is not checked, the entities modeled in the environment are displayed.

Operation
Defines the operation made when selecting the object.

Possible Values

- New: The panel of the entity selected in the previous property is opened to create an instance.
- Update The panel of the entity selected in the previous property is opened to update an instance.
- Show: The panel of the entity selected in the previous property is opened to show an instance.
- Grid: Indicates that the grid of the entity selected in the previous property is opened.

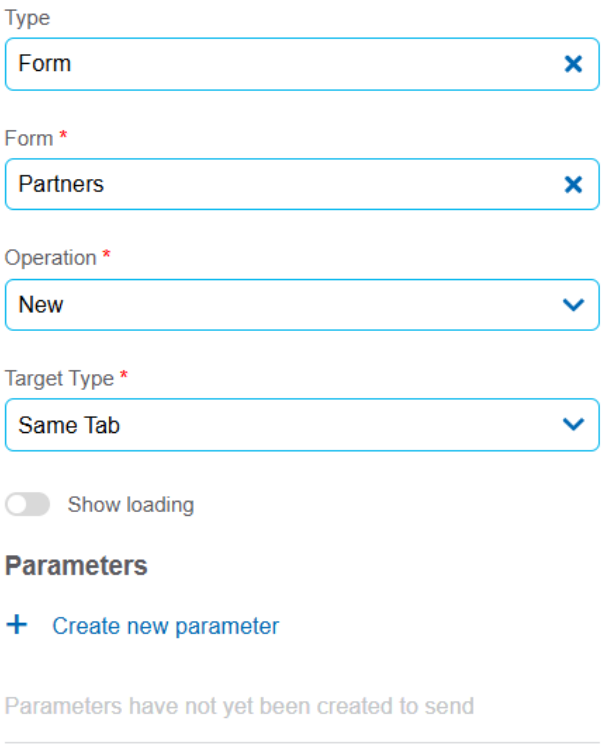
Destination Type
The available options for opening the object are displayed.

Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  Modal Horizontal Size
  Defines its width.
  Modal Vertical Size
  Defines its height.
- Iframe: selecting this option enables an additional property.
  Destination Iframe
  Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Forms



Forms
The forms modeled in the environment are displayed.

Operation
Defines the operation made when selecting the object.

Possible Values

- Create: Indicates that the panel of the object selected in the previous property is opened for the creation of an instance.
- Update: Indicates that the panel of the object selected in the previous property is opened for the update of an instance.
- Show: Indicates that the panel of the object selected in the previous property is opened for the query of an instance.
- Grid: Indicates that the grid of the object selected in the previous property is opened.

Destination Type
The available options for opening the object are displayed.

Possible Values

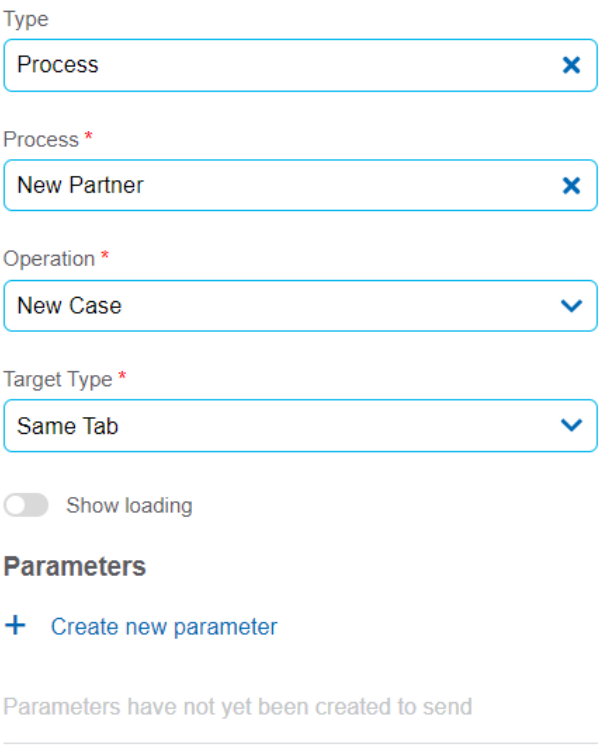
- Same Tab

- New Window
- Modal: selecting this option enables additional properties.
  Modal Horizontal Size
  Defines its width.
  Modal Vertical Size
  Defines its height.
- Iframe: selecting this option enables an additional property.
  Destination Iframe
  Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Parameters
Allows sending parameters to the selected object type.

Process



Process
The processes modeled in the environment are displayed.

Operation
Defines the operation made when selecting the object.

Destination Type
The available options for opening the object are displayed.

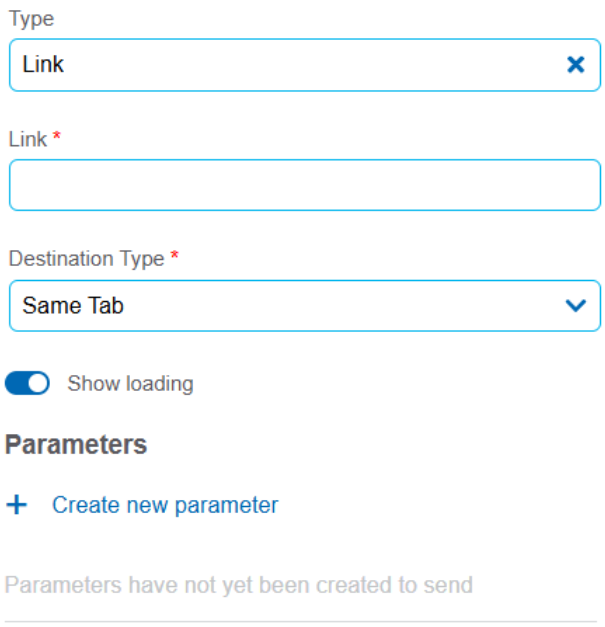
Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  Modal Horizontal Size
  Defines its width.
  Modal Vertical Size
  Defines its height.
- Iframe: selecting this option enables an additional property.
  Destination Iframe
  Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Parameters
Allows sending parameters to the selected object type.

Link



Link
Allows to enter any link.

Destination Type

The available options for opening the object are displayed.

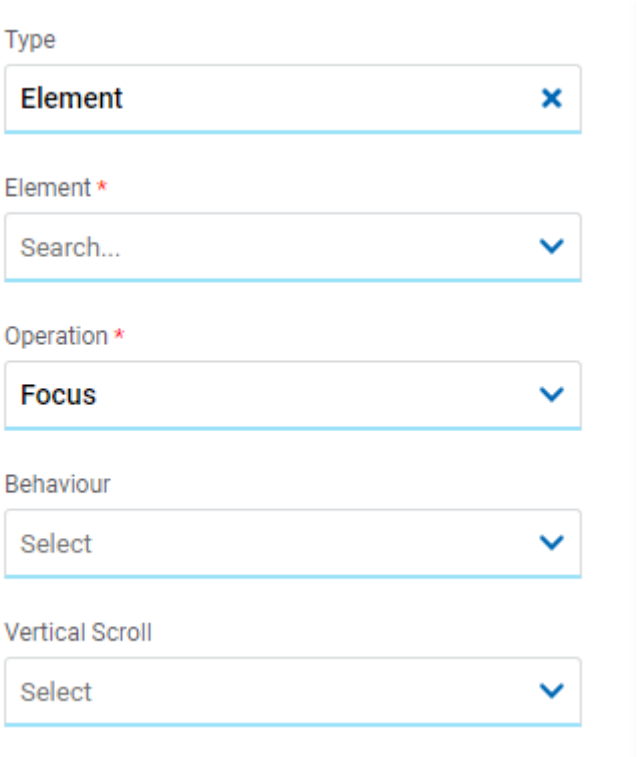
Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  Modal Horizontal Size
  Defines its width.
  Modal Vertical Size
  Defines its height.
- Iframe: selecting this option enables an additional property.
  Destination Iframe
  Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Parameters
Allows sending parameters to the selected object type.

Element



Element
The modeled elements in the object are displayed.

Operation
Defines the operation made when selecting the element.

Possible Values

- Focus
- Show
- Hide
- Show/Hide

Behaviour
Establishes the transition of the animation. It is only modelable if the "Focus" operation was selected.

Possible Values

- Auto
- Smooth

Vertical Scroll
It is only modelable if the "Focus" operation was selected.
Possible Values

- Start: Moves to the start of the selected element.
- Closest: Moves to the closest position to the selected element from the element the event fires.
- Center: Moves to the center of the selected element.
- End: Moves to the end of the selected element.

Repeater



Repeater
The modeled repeaters from the object are displayed.

Operation
Defines the operation made when selecting the element.

Possible Values

- New: Allows for the creation of an item within the repeater. If this option is selected the additional properties are enabled.
  Creation Order
  It indicates in which position the new item of the container is added.
  Possible Values
    - Home
    - End
- Delete: Allows for the deletion of an item within the repeater.

Case



Case
Show case information.

Operation
Defines the operation made when selecting the element.

Possible Values

- Show Detail
- Show Graphic Execution

Destination Type
The available options for opening the object are displayed.

Possible Values

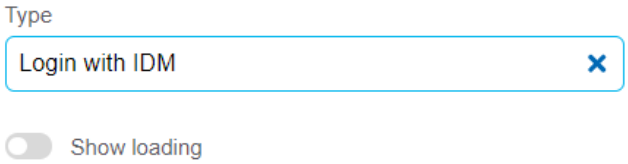
- Expanded Panel
- Modal: if this option is selected the additional properties are enabled.
  Modal Horizontal Size
  Define its width.
  Modal Vertical Size
  Define its height.

Back



It allows associating the event to go back in the browser to the element.

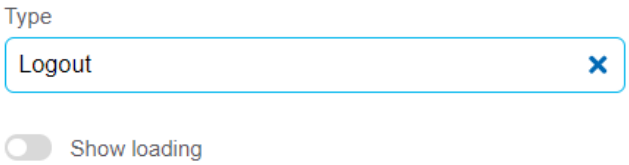
LogIn with IDM



Allows login with IDM.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

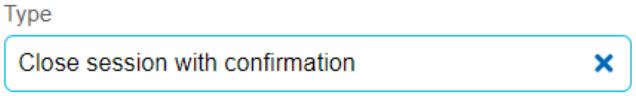
Logout



Allows logging the user out.

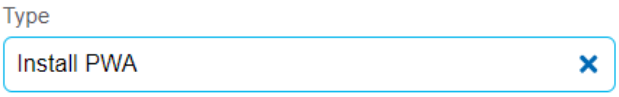
Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Close Session with Confirmation



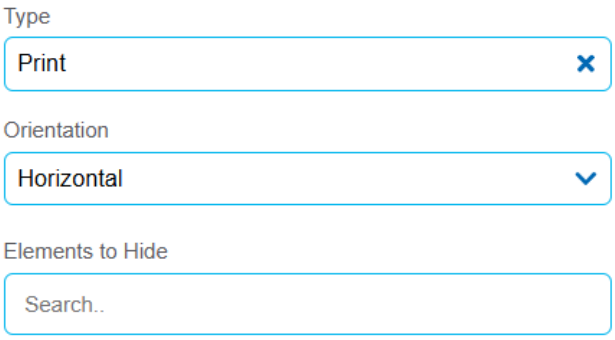
Displays a confirmation modal to logout the user.

Install PWA



Allows to install the application in the browser.

Print



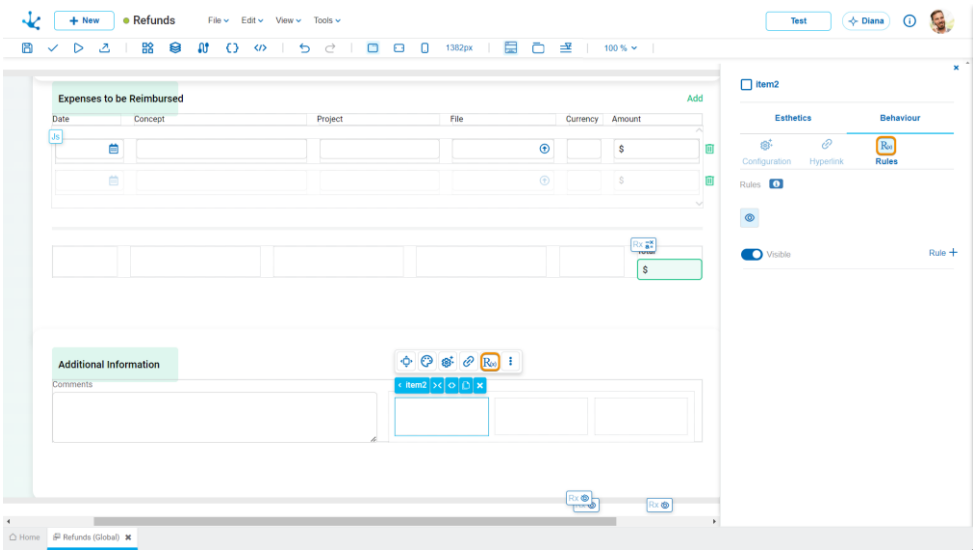
This type of hyperlink associated with an element allows printing the content of the entity or page that is being executed.

Orientation
Determines the layout of the content to be printed.
Possible Values
- Horizontal
- Vertical

Elements to hide
It allows selecting which elements of the entity or page should not be included in the printout.

Rule Properties

The rule properties panel of an element opens when clicking the icon  of its context menu.



Properties

Rules

Embedded rules on behavior, validation and calculation can be defined, associated with an element, by using the wizard (ctrl + space).

Shows syntax examples for writing the rules.

Visible

Indicates whether the element is visible. If this property is not checked, the element is not displayed in the page.

Visible (default) Not visible

Rule + Opens an edit area where a rule to determine the visibility condition can be defined. If a rule is defined, the icon is displayed with light blue borders.

Events

Items allow the use of different events.

Event	Description
onMouseIn()	It is executed when the cursor is positioned on the element.
onMouseOut()	It is executed when the cursor moves out of the element.
onClick()	It is executed when clicking on the element.
onDoubleClick()	It is executed when clicking twice on the element.

Event	Description
onInit()	It is executed before the element is loaded.
afterViewInit()	It is executed after viewing the element.
onViewportEnter()	It is executed when the element is visible.
onViewportLeave()	It is executed when the element is no longer visible.

Repeater

The repeater element allows displaying a set of instances retrieved by a data source, unlike other elements that only allow displaying a single instance.

Visually similar to the design element, but only the first item can be modified and it must have a data source defined. This item repeats as many times as instances are obtained from the data source.

If the items included in the repeater show information about the instances of the data source defined therein; such source must be previously configured.

The element properties are represented by icons in its context menu, where its operations are also available.

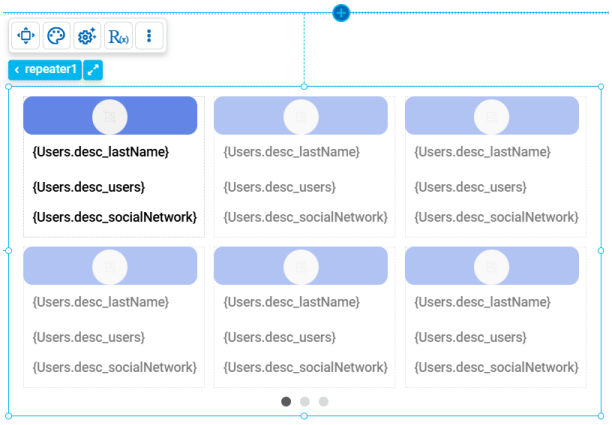
Subtypes

When selecting the option "Repeater" from the icon  in the top toolbar, a list opens with the different subtypes of this element, which can be dragged and dropped into the modeling area. Each subtype has the element's properties modeled and built in a specific way.

- Cards
- Tape
- Slider

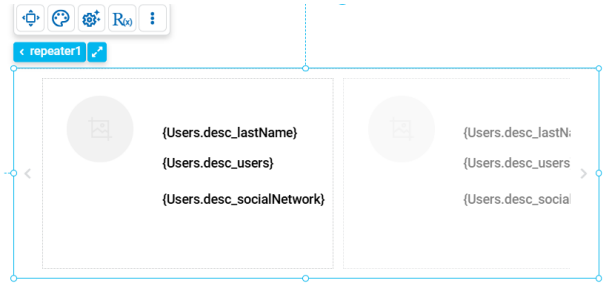
Cards

This type of repeater organizes its content in a visual card format, where each card acts as an independent unit that may contain different elements with values obtained from a data source. This type of repeater is ideal for representing structured content in an attractive and functional way.



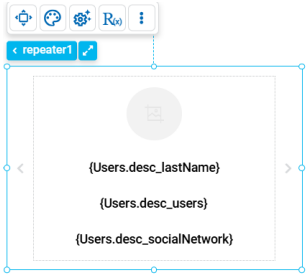
Tape

This type of repeater organizes elements in a horizontally movable structure, as if they were part of a continuous tape. It is ideal for displaying content that may come from a data source, in a limited space, while allowing users to explore more elements using browsing controls.



Slider

This type of repeater organizes elements in an interactive format that can be scrolled horizontally or vertically. It is ideal for displaying featured content, such as images, products or news, in an interactive and compact way, with values obtained from a data source.



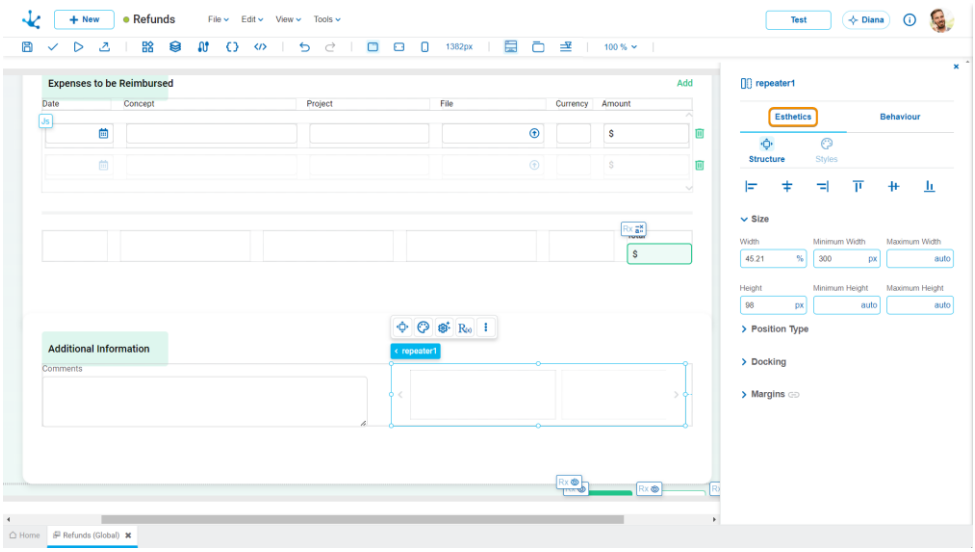
Classification of Properties

Properties are divided into two groups: esthetic properties and behavior properties.

Esthetic Properties

In the esthetic properties panel, the following are grouped:

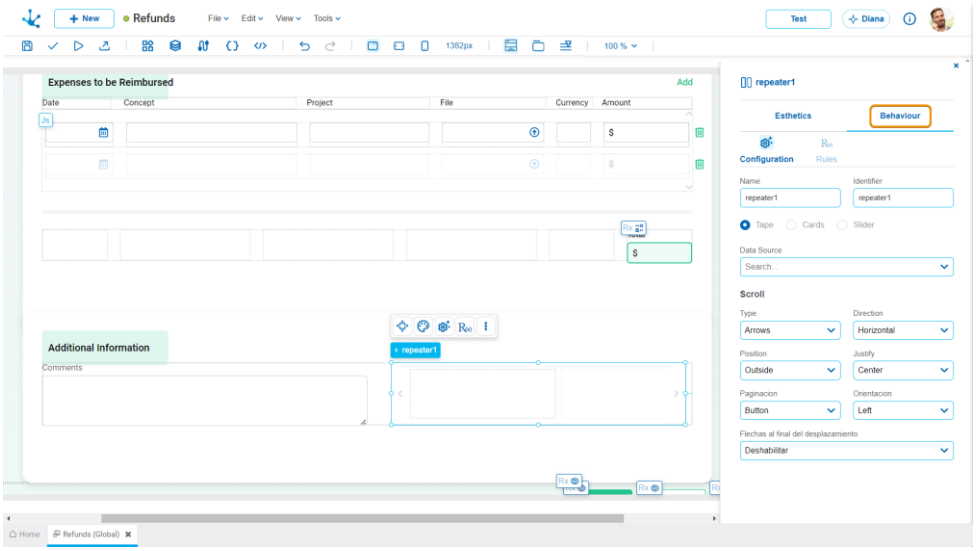
- Structure Properties
- Style Properties



Behavior Properties

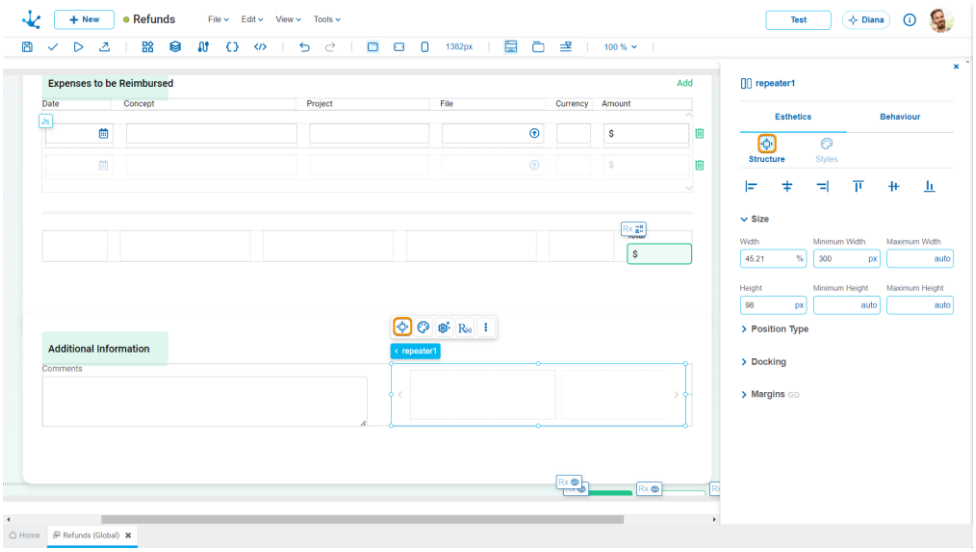
In the behavior properties panel, the following are grouped:

- Configuration Properties
- Rule Properties

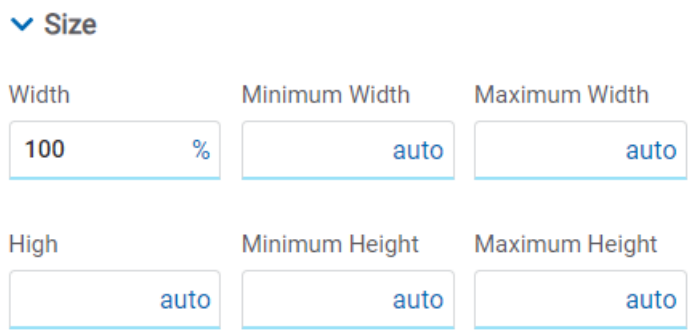


Structure Properties

The structure properties panel of an element opens when selecting the icon  of its context menu.



Size

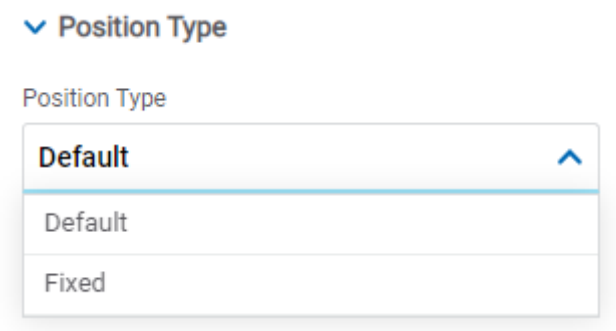


All size properties can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh). Additionally, for Width and Height properties, the "auto" option is added.

> *If selecting percentage (%) for the width and height properties, the size is calculated relative to the top element.*

Position Type

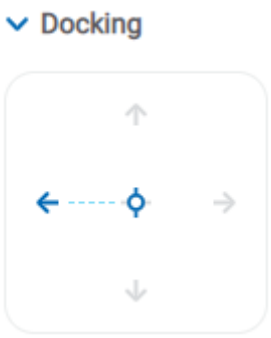
It determines if at the time of execution the element remains fixed on the page or if it moves as the scroll bar moves forward or backward.



Possible Values

- Default: The element has a relative position with respect to the superior element where it was placed (container or section).
- Fixed: Fixes the element to the page so that it is always visible in the same position. It allows to increase the visibility of important content.

Docking



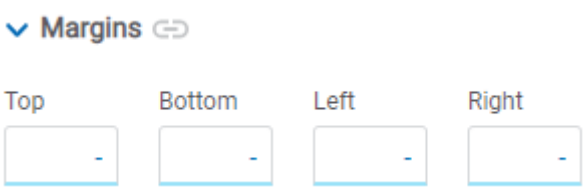
Determines the vertical and horizontal position of elements within the page section or container, when the screen is resized.

When an element is added or moved within the section or container, the modeler automatically docks it to the nearest corners or edges. It can also be configured by clicking on the arrows of the side to be docked to or on the center.

Once the element is docked, its exact position is indicated in the Margins property. The direction of the docking arrows determine the values that are automatically displayed for margins.

The docking position is indicated in the modeling area by dotted lines on the element.

Margins



It allows to define the distance of elements from the borders of their top element. The behavior of margins depends on the docking of the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Top
Distance to the top border of the highest ranking element.

Bottom
Distance to the bottom border of the highest ranking element.

Left
Distance to the left border of the highest ranking element.

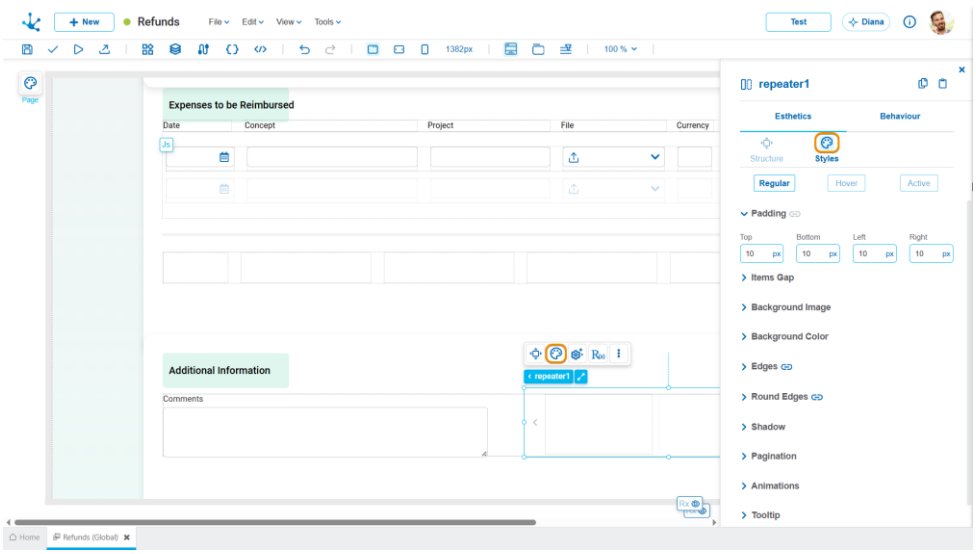
Right
Distance to the right border of the highest ranking element.

Allows the value entered in one of the margins to be copied to the other ones automatically.

Allows to indicate different values for each margin.

Style Properties

This type of element may take different states and for each of them different values for its properties may be modeled.



This type of element may take different states and for each of them different values for its properties may be modeled.



- Regular: The mouse pointer is not over the element.

- Hover: The mouse pointer is over the element.

Styles

It allows copying the entire current configuration of the style panel for the selected element (Ctrl+Alt+C).

It allows applying the previously copied configuration to another element of the same type (Ctrl+Alt+V).

Padding

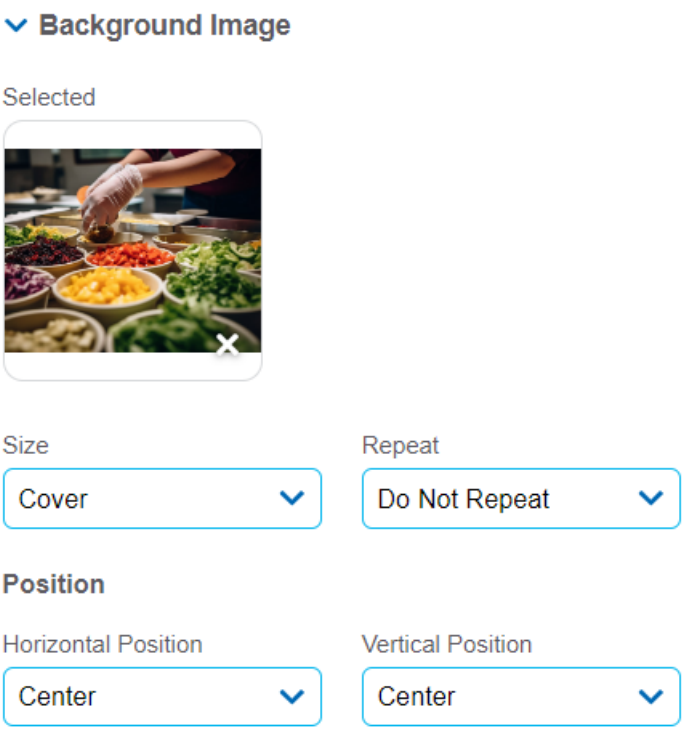


All padding space properties allow you to create a space around the borders (top, bottom, sides) and the bottom elements. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Allows values entered in one of the paddings to be copied to the other ones automatically.
Allows to indicate different values for each padding.

Items Gap
It defines the spacing between the items in the repeater.

Background Image



Allows to add a background image to the element.

Selected

An image can be uploaded from the computer where it is being modeled.

Size

Possible Values

- Cover
- Content
- Auto

Repeat

Possible Values

- Repeat
- Repeat Hor.
- Repeat Vert.
- Spacing
- Rounded
- Do Not Repeat

Horizontal Position

Possible Values

- Center

- Left
- Right

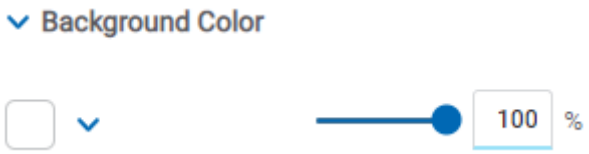
Vertical Position

- Center
- Top
- Botton

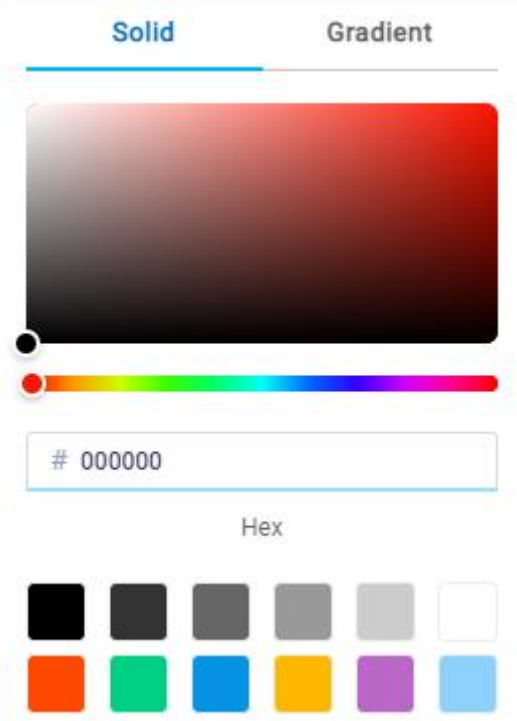
Background Color

It allows to model a background color for the element, it can be solid or gradient and different properties are defined for each one.

Solid



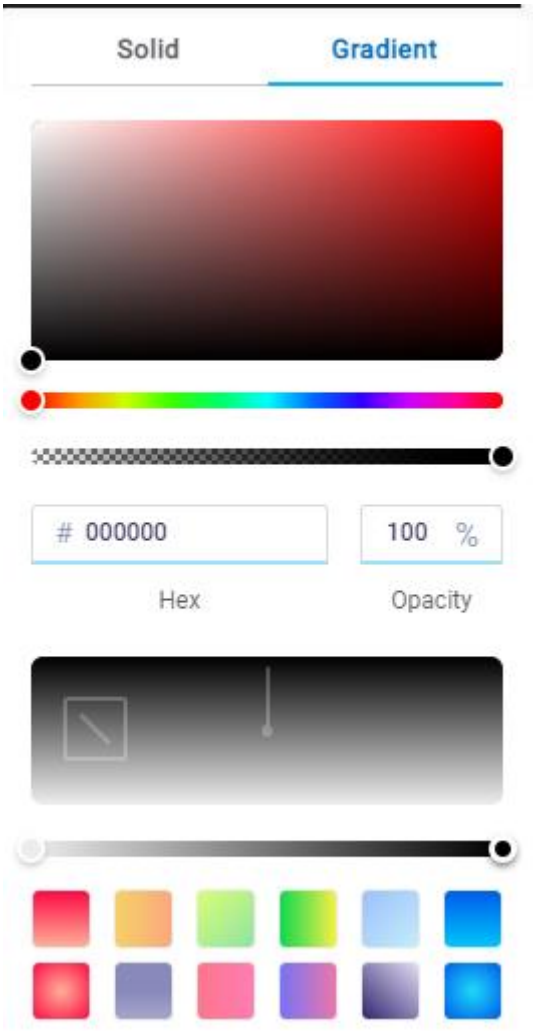
This option enables not only to select the color from a palette but also to define the degree of transparency.



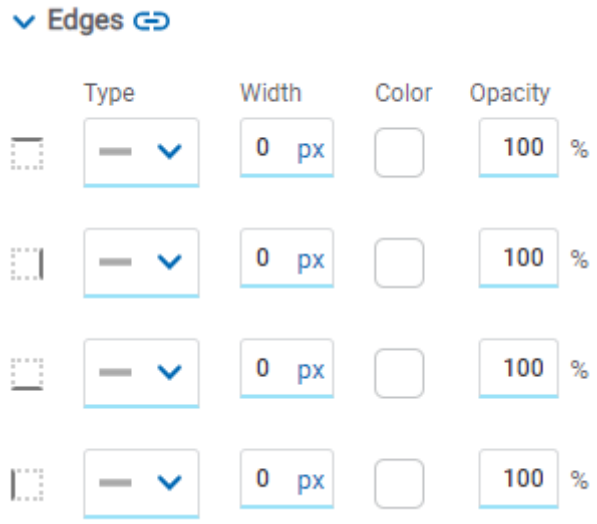
Gradient



This option enables to select not only the color from a palette but also to define its opacity and intensity according to the selected angle.



Edges



It allows to define the style of borders. Each one has its type, width, color and opacity defined, the latter as a percentage.

Allows the value entered in one of the borders to be copied to the other ones automatically.
Allows to indicate different values for each border.

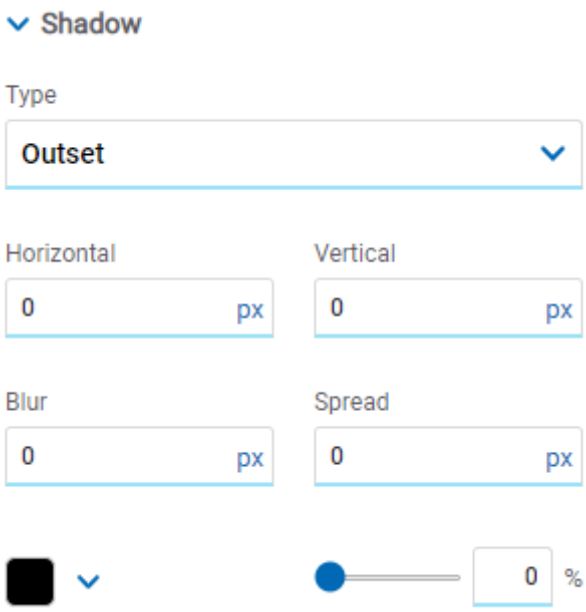
Round Edges



It allows to define the round edges at the corners of elements.
It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Allows the value entered in one of the borders to be copied to the other ones automatically.
Allows to indicate different values for each border.

Shadow



Allows to define a shadow effect around the element.

Horizontal
Horizontal size of the shadow to the right of the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Vertical
Vertical size of the shadow below the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Blur
The greater this value is, the greater and lighter the shadow becomes. If not specified, its value is 0 and the shadow border is darker. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Spread
Positive values will cause the shadow to expand and grow bigger, while negative values will cause the shadow to shrink. If not specified, its value is 0 and the shadow will have the same size as the element.

Color
Defines the color of the shadow.

Transparency
Defines the transparency of the shadow.

Pagination

The pagination properties depend on the type of scrolling modeled in the repeater's configuration properties and can be adjusted to achieve the desired behavior.

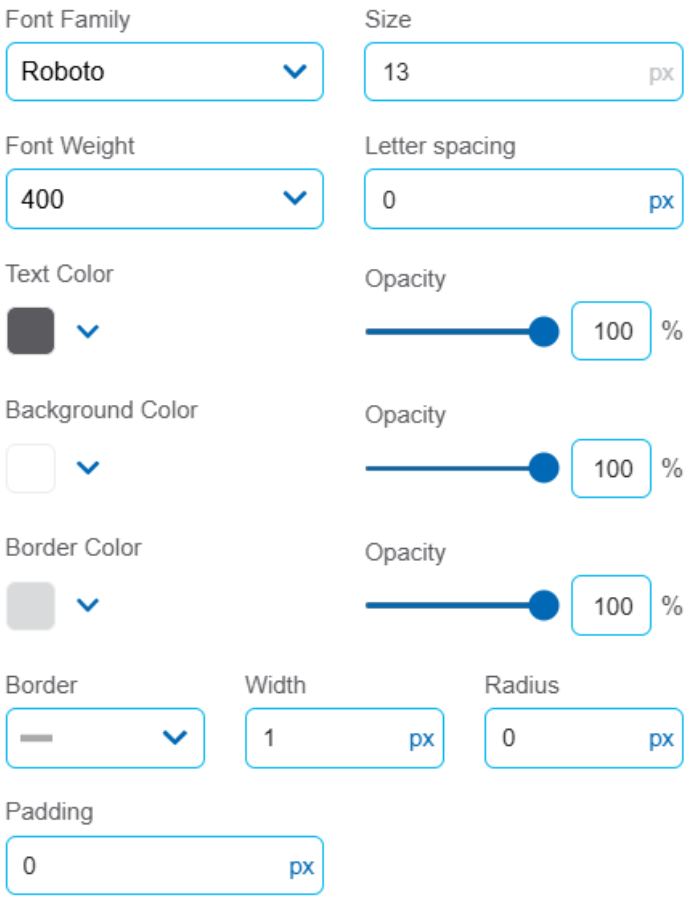
Arrow Scrolling
Arrows: size, color, background, disabled arrow color, and disabled background color.

> *The background color and the disabled background color only have an effect when the Position property is modeled as 'Inside'.*

Dot Scrolling
Dots: size, spacing, color and color of the selected dot.

Page Scrolling
Font: family, size and color
Buttons: size, spacing
Colors: background and font for normal, selected, and disabled states. It also allows setting the background color on hover.

Tooltip

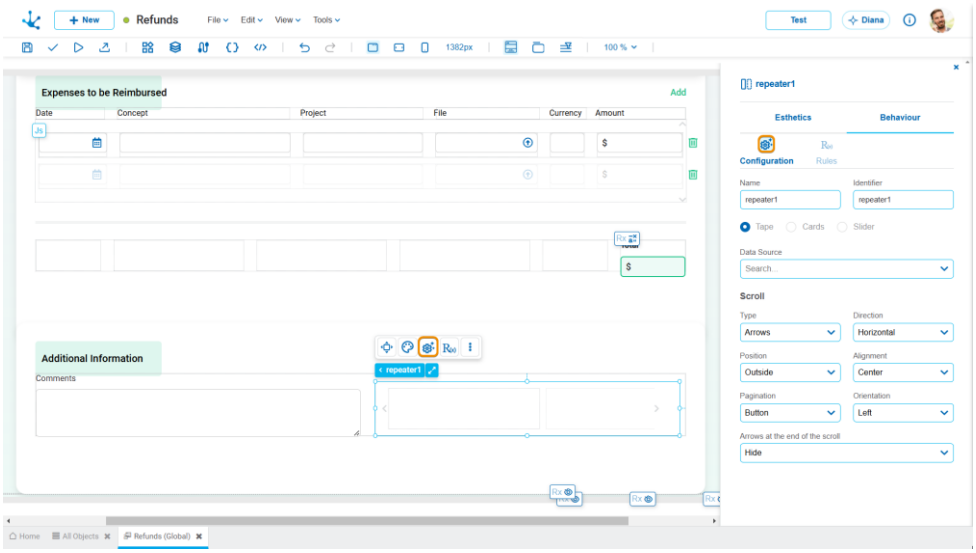


Source
It allows customizing the font, size, thickness and letter spacing of the pop-up message.

Colors
The color and opacity of the text, background and border can be defined. Borders are set uniformly for all sides of the tooltip.

Border
The border type, width, and border radius can be configured.

The configuration properties panel of an element opens when clicking the icon  of its context menu.



Name
Name used to reference the element during modeling.

Identifier
Uniquely identifies the element. It is used in the Javascript SDK.

Tooltip
Pop-up message that appears when hovering over an element, providing additional information.



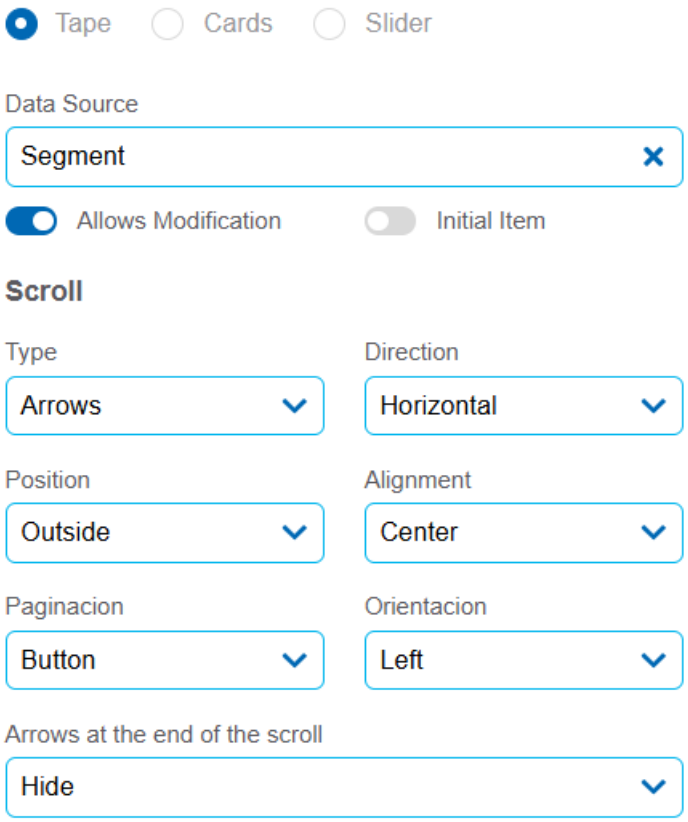
Text
Content of the message being displayed.

Direction
Position of the message in relation to the element (top, bottom, left, or right).

Items can be repeated in tape, card, or slider format.

Tape

When the entity is executed, the items scroll within the repeater as the forward and backward icons are pressed. The number of items initially displayed on the tape depends on the size of the repeater.



Data Source

It allows selecting the source from which the data displayed on the repeater is retrieved. Once the source is selected, the Allows Modification property is enabled.

Allows Modification

It allows adding or removing items from the repeater using a modeled hyperlink. If the property is not activated, the number of items is fixed and cannot be modified. When activated, the Initial Item property is displayed.

Initial Item

It allows adding an empty item to the repeater using a modeled hyperlink.

> *If the repeater is empty at runtime, the initial item is automatically added, without the user needing to use the hyperlink.*

Type

If the entity is used from a desktop, the "Arrows" option should be selected so that the icons < *and* > indicating the forward or backward movement of the items on the tape are enabled. If used from a touch breakpoint, the "None" option can be selected to perform the scrolling.
If the selected value is "Arrows" the enabled properties are: Position and Alignment.

Possible Values

- Arrows
- None

Direction

Allows to select the direction in which the items within the element are displayed.

Possible Values

- Horizontal
- Vertical

Position and Alignment

Select the position of the scroll icons within the repeater.

Possible Values for Position

- Top
- Bottom
- Inside
- Outside

Possible Values for Alignment

- If the “Inside” or “Outside” values are selected for the Position property, the possible values of the Alignment property are: “Top”, “Center”, “Bottom”.
- If the “Top” or “Bottom” values are selected for the Position property, the possible values of the Alignment property are: “Left”, “Center”, “Right”.

Pagination

Defines how the elements of a repeater are loaded and displayed in the entity.

Possible Values

- Button: Loading more items is triggered manually, the user must click on a button labeled "Upload more" to show additional elements.
- Infinite: elements load automatically as the user scrolls down the entity. This behavior is known as infinite scroll.

Orientation

Defines the position at which the pagination element appears within the repeater container.

Possible Values

- Left: the pagination element is aligned to the left side of the container.
- Right: the pagination element is aligned to the right side of the container.

Arrows at the end of the scroll

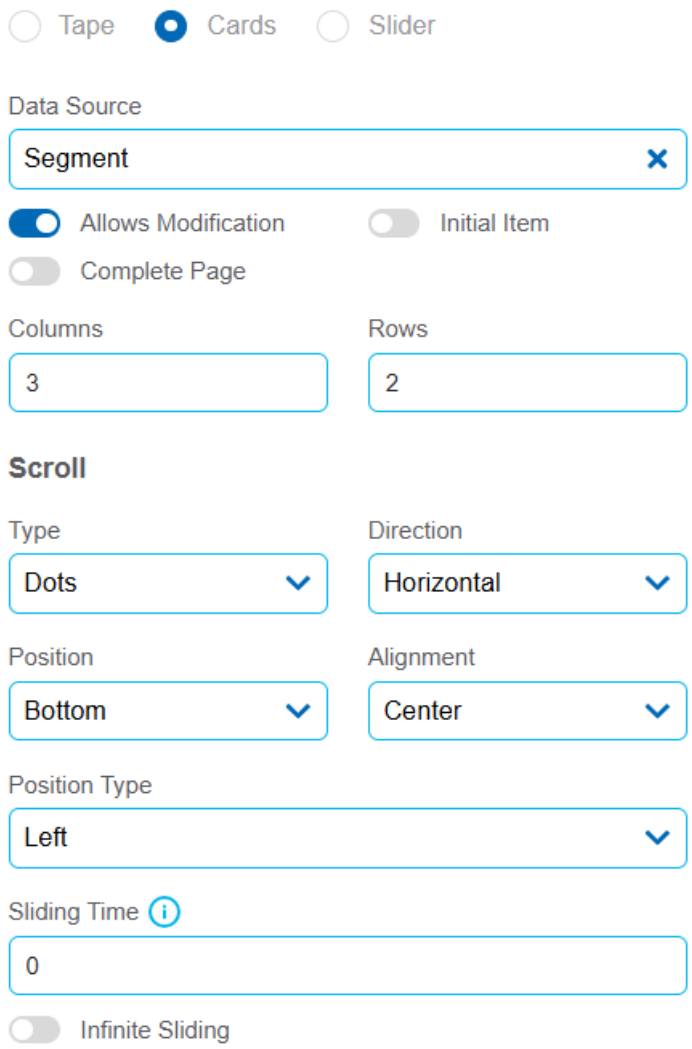
Defines the behavior of the navigation arrows when the last element is reached.

Possible Values

- Disable: The scroll arrows are disabled, indicating that there is no more content.
- Hide: Scroll arrows disappear completely, avoiding unnecessary navigation elements to be presented.

Cards

Unlike the tape, the items paginate within the repeater as the forward and backward icons are pressed. The number of items that are initially displayed in the repeater depends on the values modeled in the Columns and Rows properties.



Data Source
It allows selecting the source from which the data displayed on the repeater is retrieved. Once the source is selected, the Allows Modification property is enabled.

Allows Modification
It allows adding or removing items from the repeater using a modeled hyperlink. If the property is not activated, the number of items is fixed and cannot be modified. When activated, the Initial Item and Complete Page properties are displayed.

Initial Item
It allows adding an empty item to the repeater using a modeled hyperlink.

Complete Page
Automatically adds the necessary empty items to the repeater to reach the total quantity defined in its modeling, via a hyperlink

> *If the repeater is empty at execution, the initial item is automatically added or the defined number of items is completed and the user does not need to use the hyperlink.*

Columns
It allows defining the number of columns in the repeater.

Rows
It allows defining the number of rows in the repeater.

Type
Defines the type of control to move forward or backward between the groups of items visible in the repeater.

Possible Values

- Dots: each dot represents a group of visible items and they can be selected to change the view.
- Arrows: icons on the sides that allow moving forward or backward through groups of items..
- Pages: combines arrows with page numbers. The user can navigate between pages using arrows or access one directly. Up to two pages before and after the current one are displayed, with ellipses if there are more. The active page is highlighted to indicate the current position.

Direction
Allows to select the direction in which the items within the element are displayed.

Possible Values

- Horizontal
- Vertical

Position and Alignment
Select the position of the scroll icons within the repeater.

Possible Values for Position

- Top
- Bottom
- Inside Top
- Inside Bottom

Possible Values for Alignment

- Left
- Center
- Right

Position Type
It allows defining how cards are aligned within their container.

Possible Values

- Left
- Centered
- Right

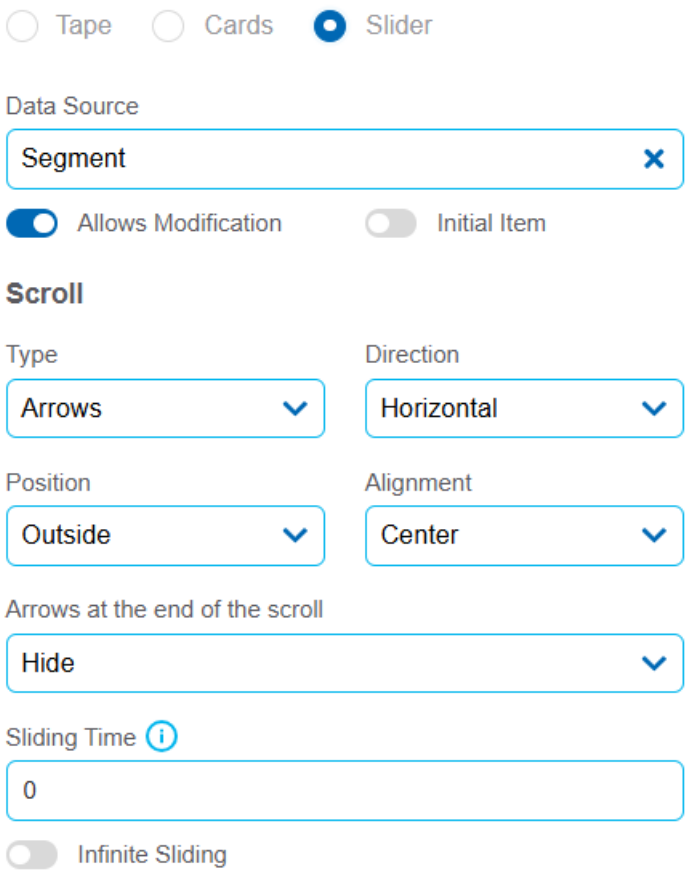
Sliding Time
It allows defining the interval in seconds for the automatic transition between cards.

Infinite Sliding

It allows the card type repeater to slide continuously, returning to the beginning once the end is reached.

Slider

It behaves similarly to the Tape-type repeater, offering additional features to customize its behavior. It is designed to show and focus on a single element at a time.



Data Source

It allows selecting the source from which the data displayed on the repeater is retrieved. Once the source is selected, the Allows Modification property is enabled.

Allows Modification

It allows adding or removing items from the repeater using a modeled hyperlink. If the property is not activated, the number of items is fixed and cannot be modified. When activated, the Initial Item property is displayed.

Initial Item

It allows adding an empty item to the repeater using a modeled hyperlink.

> *If the repeater is empty at runtime, the initial item is automatically added, without the user needing to use the hyperlink.*

Type

Defines the type of control to move forward or backward between items in the repeater.

Possible Values

- Dots: each dot represents an item and they can be selected to change the view.
- Arrows: icons on the sides that allow moving forward or backward one item at a time.
- Pages: combines arrows with page numbers. The user can navigate between items using arrows or go directly to a specific one. Up to two items before and after the current one are displayed, with ellipses if there are more. The active item is highlighted to indicate the current position.

Direction

Allows to select the direction in which the items within the element are displayed.

Possible Values

- Horizontal
- Vertical

Position and Alignment

Select the position of the scroll icons within the repeater.

Possible Values for Position

- Top
- Bottom
- Inside
- Outside

Possible Values for Alignment

- If the "Inside" or "Outside" values are selected for the Position property, the possible values of the Alignment property are: "Top", "Center", "Bottom".
- If the "Top" or "Bottom" values are selected for the Position property, the possible values of the Alignment property are: "Left", "Center", "Right".

Arrows at the end of the scroll

Defines the behavior of the navigation arrows when the last element is reached.

Possible Values

- Disable: The scroll arrows are disabled, indicating that there is no more content.
- Hide: Scroll arrows disappear completely, avoiding unnecessary navigation elements to be presented.

Sliding Time

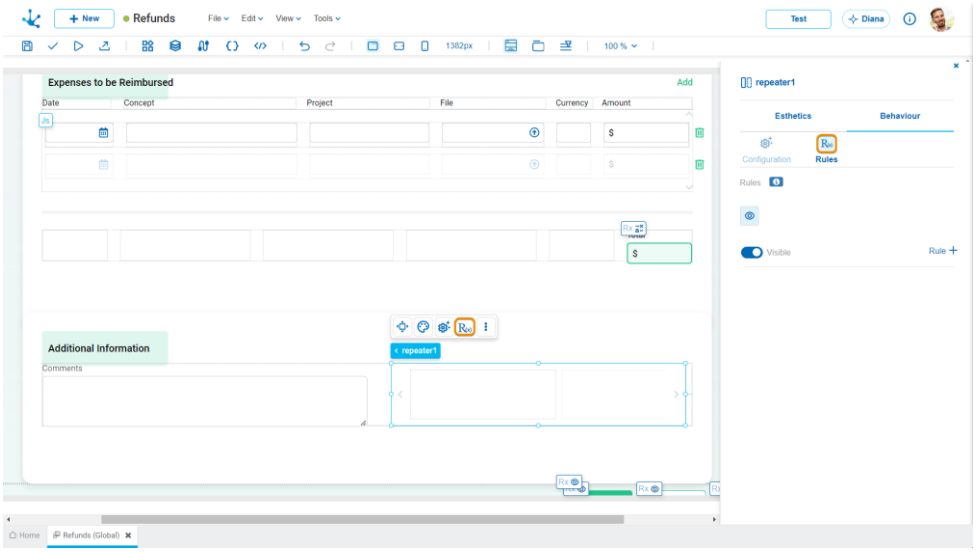
It allows defining the interval in seconds for the automatic transition between cards.

Infinite Sliding

It allows the card type repeater to slide continuously, returning to the beginning once the end is reached.

Rule Properties

The rule properties panel of an element opens when clicking the icon  of its context menu.



Properties

Rules

Embedded rules on behavior, validation and calculation can be defined, associated with an element, by using the wizard (ctrl + space).

Shows syntax examples for writing the rules.

Visible

Indicates whether the element is visible. If this property is not checked, the element is not displayed in the page.

Visible (default) Not visible

Rule + Opens an edit area where a rule to determine the visibility condition can be defined. If a rule is defined, the icon is displayed with light blue borders.

Operations when defining the rule:

Saves the new or modified rule

Cancels the operation

Operations once the rule is defined:

Edits the existing rule

Deletes the rule

Item

Items are defined within the repeater and only the first one can be modified so changes are applied to the rest.

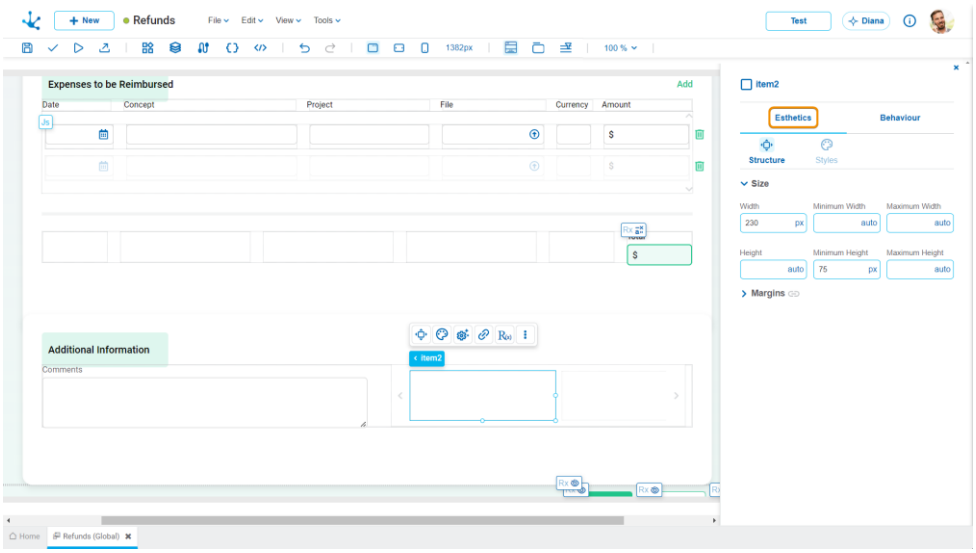
Classification of Properties

Properties are divided into two groups: esthetic properties and behavior properties.

Esthetic Properties

In the esthetic properties panel, the following are grouped:

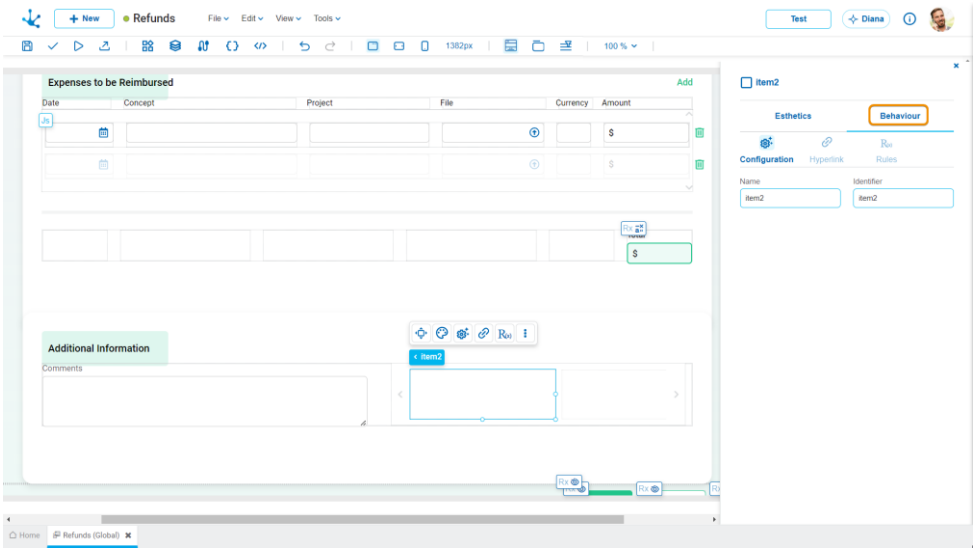
- Structure Properties
- Style Properties



Behavior Properties

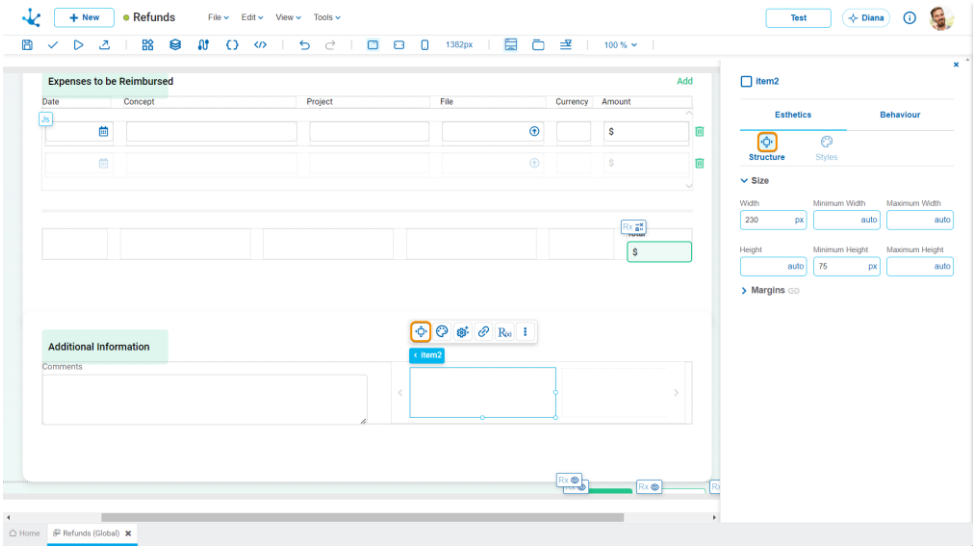
In the behavior properties panel, the following are grouped:

- Configuration Properties
- Hyperlink Properties
- Rule Properties

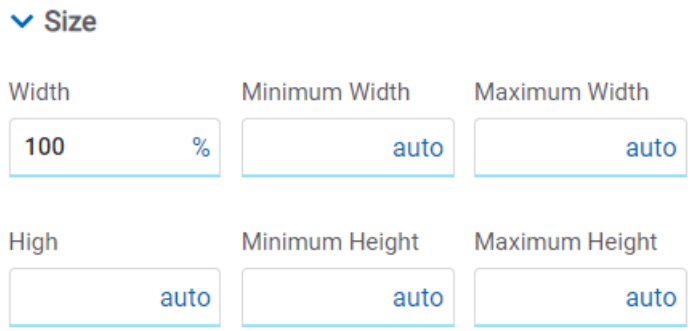


Structure Properties

The structure properties panel of an element opens when selecting the icon  of its context menu.



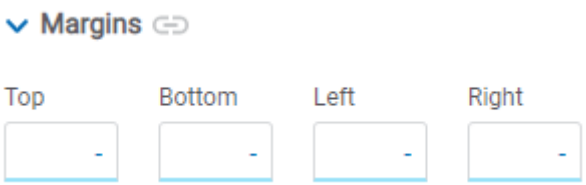
Size



All size properties can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh). Additionally, for Width and Height properties, the "auto" option is added.

> *If selecting percentage (%) for the width and height properties, the size is calculated relative to the top element.*

Margins



Allows to define the distance among items and also from items to the borders of the containing element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Top
Distance to the top border of the containing element.

Bottom
Distance to the bottom border of the containing element.

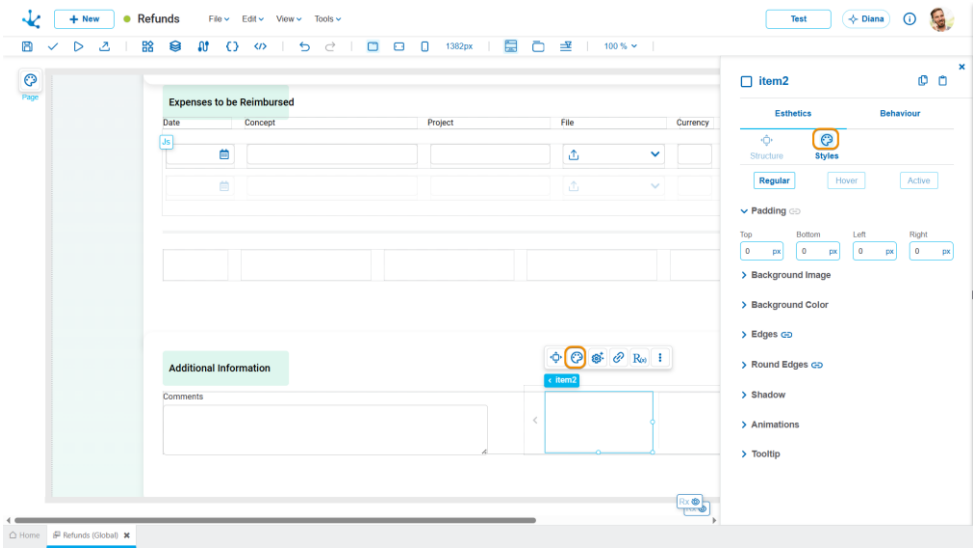
Left
Distance to the left border of the containing element or to the previous item.

Right
Distance to the right border of the containing element or to the following item.

Allows the value entered in one of the margins to be copied to the other ones automatically.
Allows to indicate different values for each margin.

Style Properties

This type of element may take different states and for each of them different values for its properties may be modeled.



This type of element may take different states and for each of them different values for its properties may be modeled.



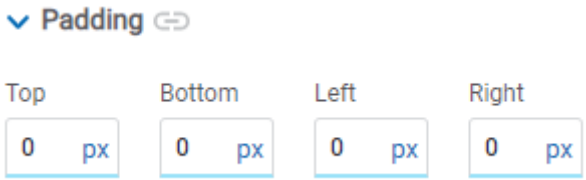
- Regular: The mouse pointer is not over the element.
- Hover: The mouse pointer is over the element.

Styles

It allows copying the entire current configuration of the style panel for the selected element (Ctrl+Alt+C).

It allows applying the previously copied configuration to another element of the same type (Ctrl+Alt+V).

Padding

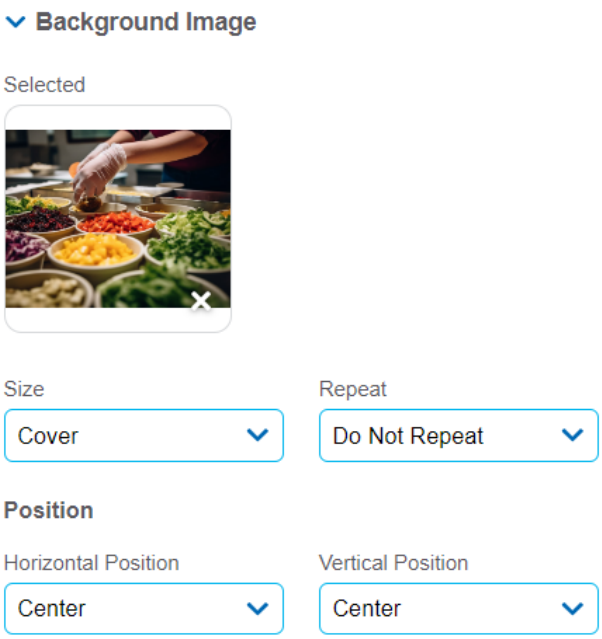


All padding space properties allow to create a space around the item borders (top, bottom, sides) and its content. The padding space configured is inward, while the margin is configured relative to a top item. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Allows values entered in one of the paddings to be copied to the other ones automatically.
Allows to indicate different values for each padding.

Background Image

It allows the addition of a background image to the item.

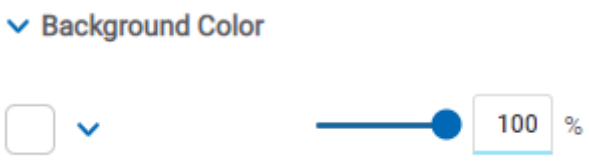


Selected
An image can be uploaded from the computer where it is being modeled.

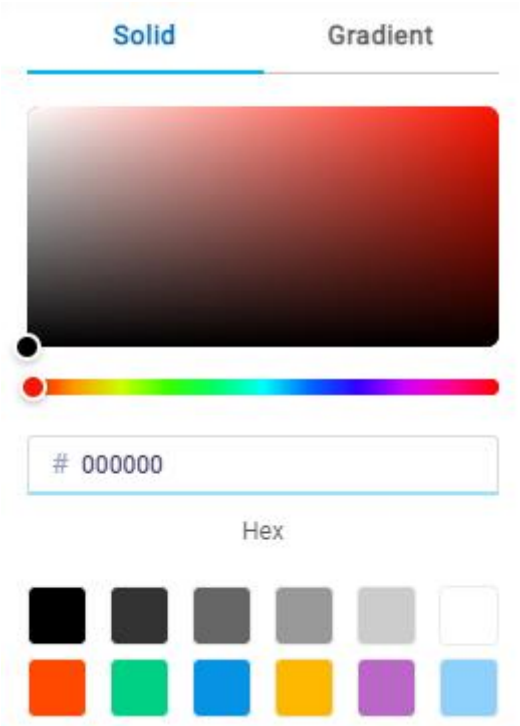
Background Color

It allows to model a background color for the element, it can be solid or gradient and different properties are defined for each one.

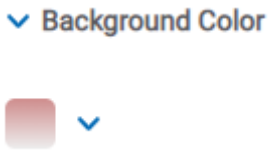
Solid



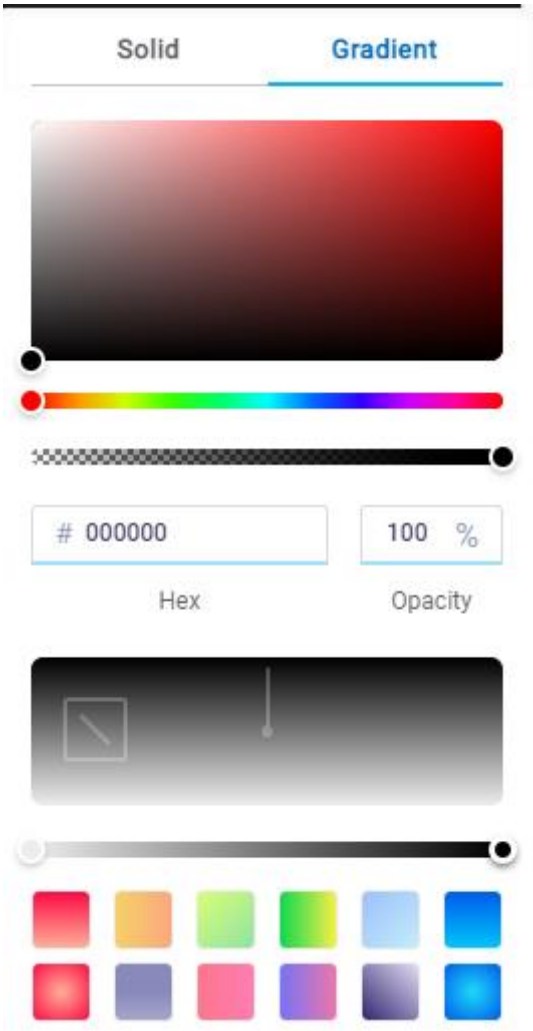
This option enables not only to select the color from a palette but also to define the degree of transparency.



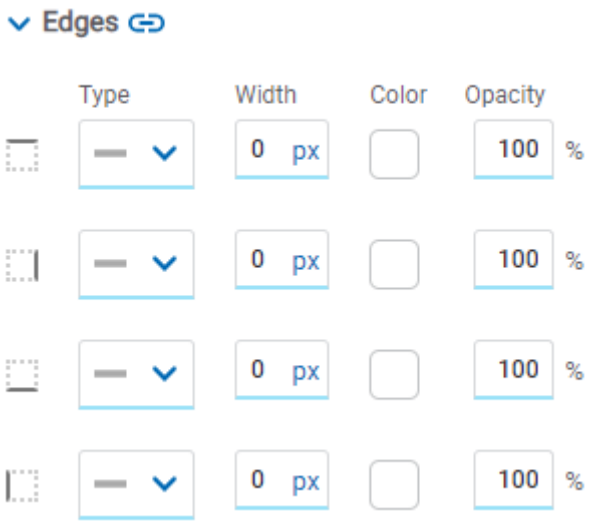
Gradient



This option enables to select not only the color from a palette but also to define its opacity and intensity according to the selected angle.



Edges



It allows to define the style of borders. Each one has its type, width, color and opacity defined, the latter as a percentage.

Allows the value entered in one of the borders to be copied to the other ones automatically.
Allows to indicate different values for each border.

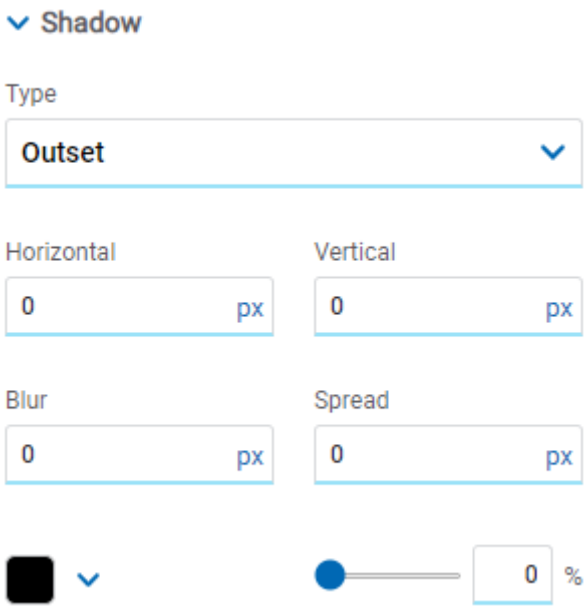
Round Edges



It allows to define the round edges at the corners of items.
It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Allows the value entered in one of the borders to be copied to the other ones automatically.
Allows to indicate different values for each border.

Shadow



Allows to define a shadow effect around the item.

Type

Possible Values

- Outset
- Inset

Horizontal
Horizontal size of the shadow to the right of the item. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Vertical
Vertical size of the shadow below the item. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

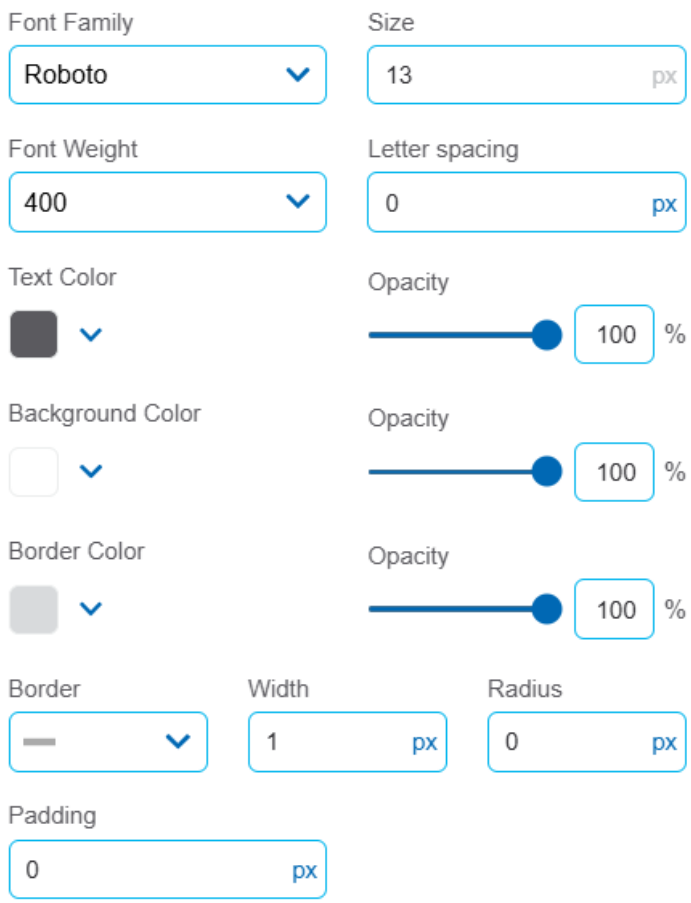
Blur
The greater this value is, the greater and lighter the shadow becomes. If not specified, its value is 0 and the shadow border is darker. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Spread
Positive values will cause the shadow to expand and grow bigger, while negative values will cause the shadow to shrink. If not specified, its value is 0 and the shadow will have the same size as the item.

Color
Defines the color of the shadow.

Transparency
Defines the transparency of the shadow.

Tooltip



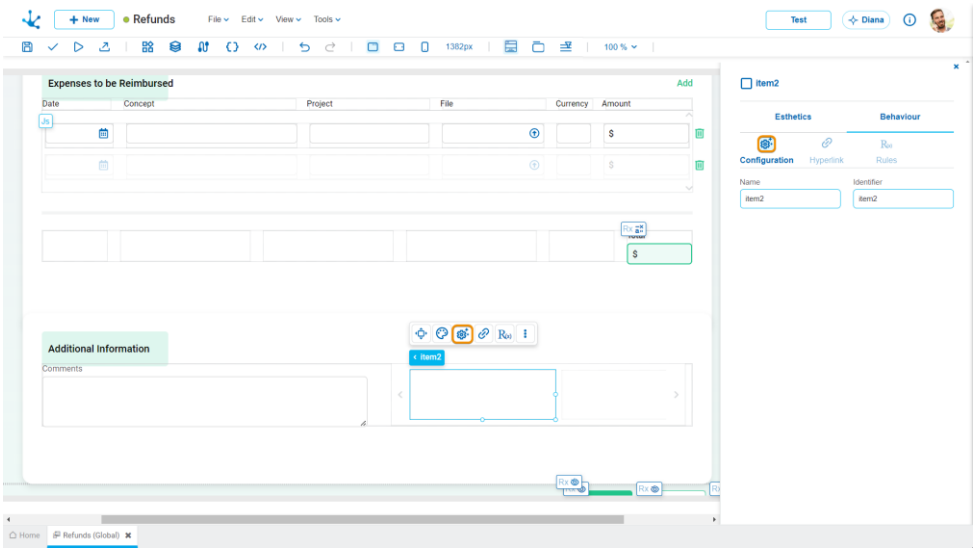
Source
It allows customizing the font, size, thickness and letter spacing of the pop-up message.

Colors
The color and opacity of the text, background and border can be defined. Borders are set uniformly for all sides of the tooltip.

Border
The border type, width, and border radius can be configured.

Configuration Properties

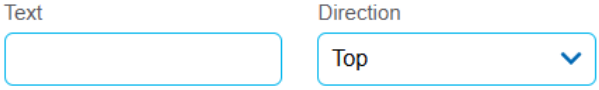
The configuration properties panel of an element opens when clicking the icon  of its context menu.



Name
Name used to reference the element during modeling.

Identifier
Uniquely identifies the element. It is used in the Javascript SDK.

Tooltip
Pop-up message that appears when hovering over an element, providing additional information.



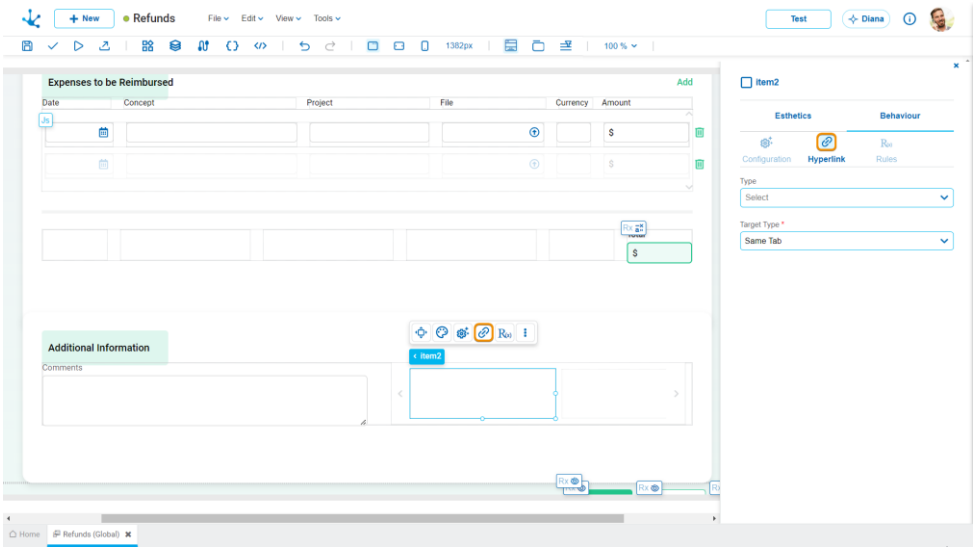
Text
Content of the message being displayed.

Direction

Position of the message in relation to the element (top, bottom, left, or right).

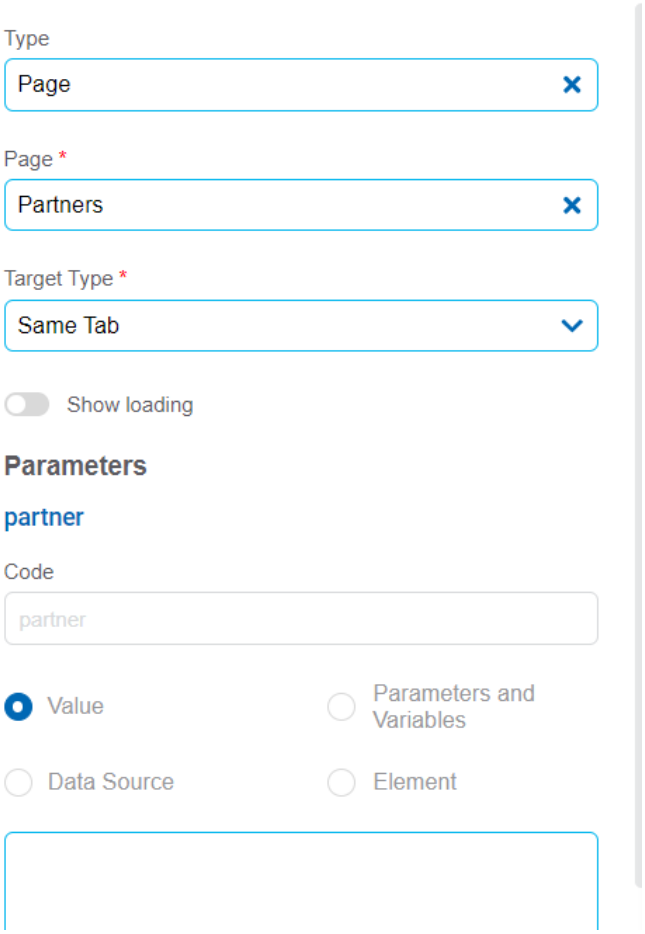
Hyperlink Properties

The hyperlink properties panel of an element opens when clicking the icon  of its context menu.



Define the behavior of the item when selecting it. Different properties are enabled depending on the type of object selected.

Page



Page
The pages modeled in the environment are displayed.

Destination Type
The available options for opening the object are displayed.

Possible Values

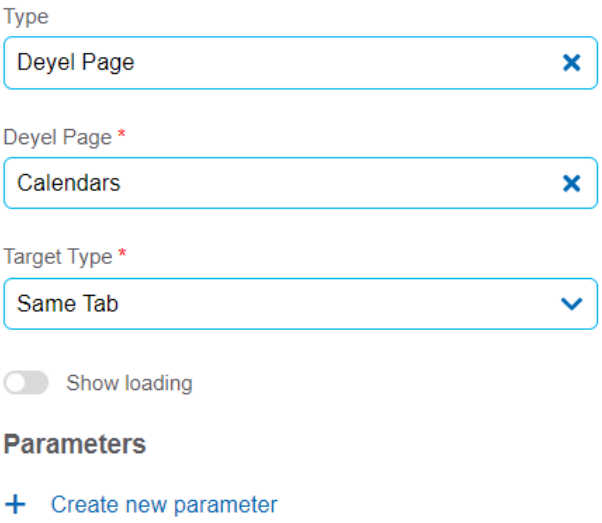
- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  Modal Horizontal Size
  Defines its width.
  Modal Vertical Size
  Defines its height.
- Iframe: selecting this option enables an additional property.
  Destination Iframe
  Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading

Shows the loading icon and disables the user interaction until the redirection is over.

Parameters
If there are parameters, their value can be specified with text, parameters and variables or data source.

Deyel Page



Deyel Page
The pages belonging to **Deyel** are displayed.

Destination Type
The available options for opening the object are displayed.

Possible Values

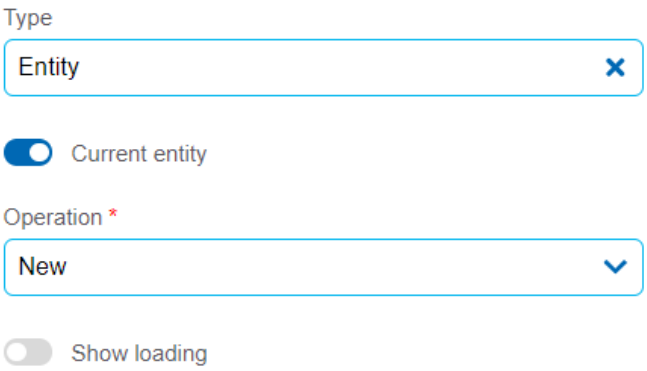
- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  - Modal Horizontal Size
    Defines its width.
  - Modal Vertical Size
    Defines its height.
- Iframe: selecting this option enables an additional property.
  - Destination Iframe
    Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Parameters

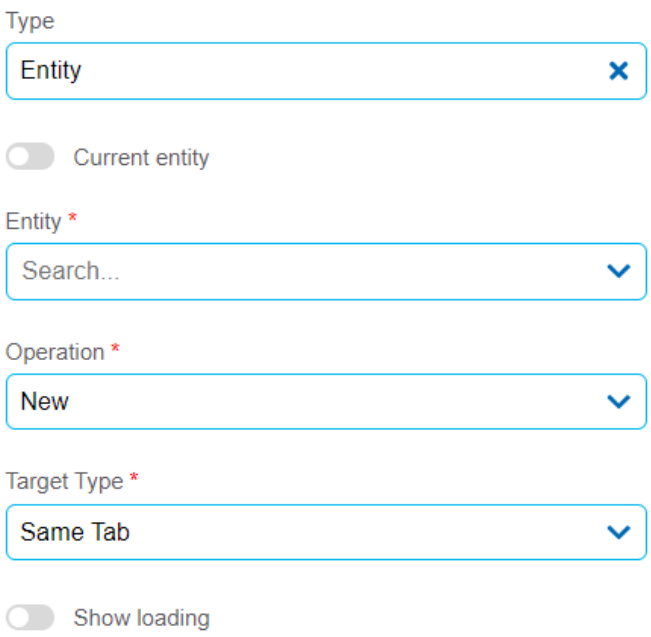
Allows sending parameters to the selected object type.

Entity



Current Entity
If this property is checked, the operation applies directly to the entity being modeled.

Operation
Defines the operation made when selecting the object.
Possible Values
- Create: Indicates that the panel of the current entity is opened to create an instance.
- Update: Indicates that the panel of the current entity is opened to update an instance.
- Delete: Indicates that the panel of the current entity is opened to delete an instance.
- Send Email: It is visible if the property Send Emails is enabled. Clicking on the modeled element opens a panel during instance execution to send an email. The panel size is adjusted with the properties Modal Width and Modal Height.
- Emails: Visible if the Send Emails property is enabled. Clicking on the modeled element opens a panel during instance execution to manage sent emails. The panel is displayed in the iframe defined in the Target Iframe property.



Entity
If the property Current Entity, is not checked, the entities modeled in the environment are displayed.

Operation
Defines the operation made when selecting the object.

Possible Values

- New: The panel of the entity selected in the previous property is opened to create an instance.
- Update The panel of the entity selected in the previous property is opened to update an instance.
- Show: The panel of the entity selected in the previous property is opened to show an instance.
- Grid: Indicates that the grid of the entity selected in the previous property is opened.

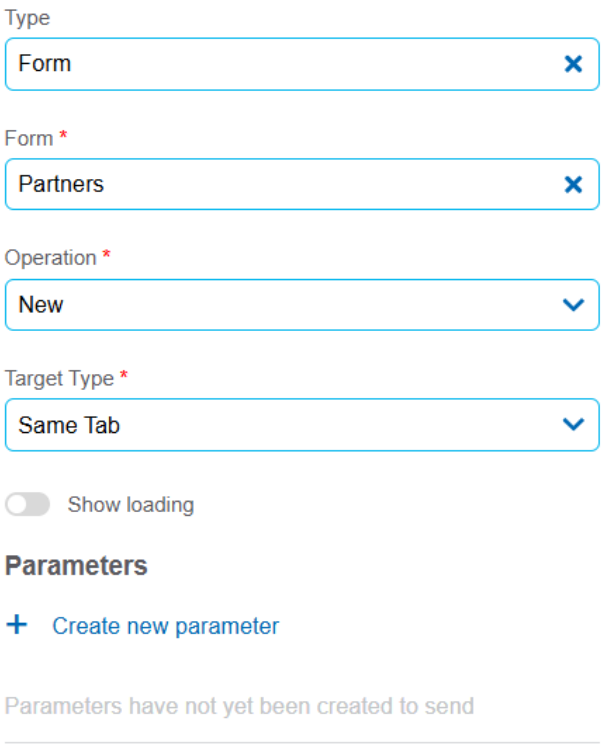
Destination Type
The available options for opening the object are displayed.

Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  Modal Horizontal Size
  Defines its width.
  Modal Vertical Size
  Defines its height.
- Iframe: selecting this option enables an additional property.
  Destination Iframe
  Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Forms



Forms
The forms modeled in the environment are displayed.

Operation
Defines the operation made when selecting the object.

Possible Values

- Create: Indicates that the panel of the object selected in the previous property is opened for the creation of an instance.
- Update: Indicates that the panel of the object selected in the previous property is opened for the update of an instance.
- Show: Indicates that the panel of the object selected in the previous property is opened for the query of an instance.
- Grid: Indicates that the grid of the object selected in the previous property is opened.

Destination Type
The available options for opening the object are displayed.

Possible Values

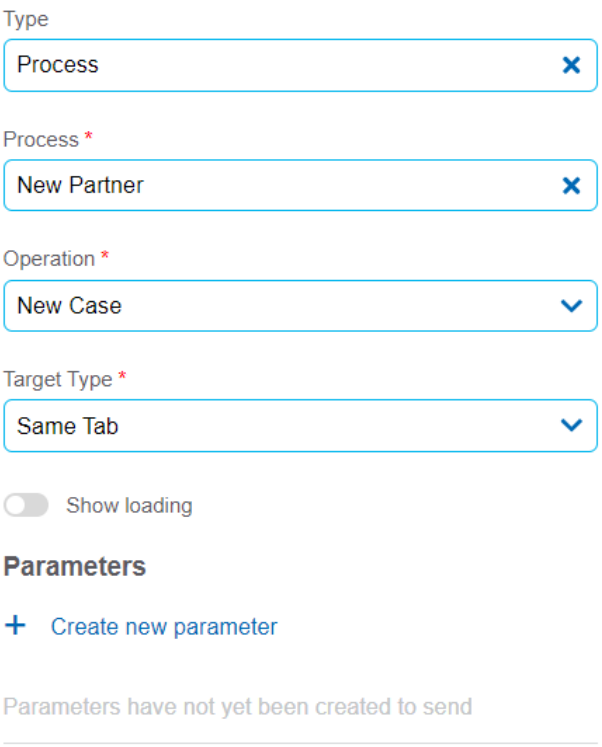
- Same Tab

- New Window
- Modal: selecting this option enables additional properties.
  Modal Horizontal Size
  Defines its width.
  Modal Vertical Size
  Defines its height.
- Iframe: selecting this option enables an additional property.
  Destination Iframe
  Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Parameters
Allows sending parameters to the selected object type.

Process



Process
The processes modeled in the environment are displayed.

Operation
Defines the operation made when selecting the object.

Destination Type
The available options for opening the object are displayed.

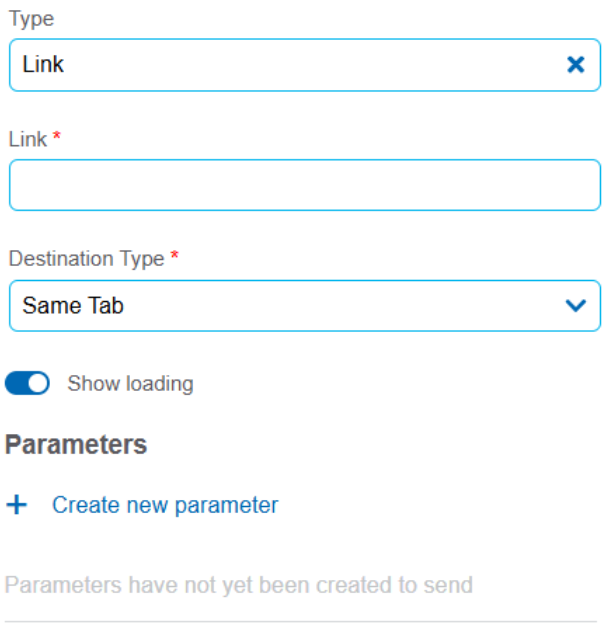
Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  Modal Horizontal Size
  Defines its width.
  Modal Vertical Size
  Defines its height.
- Iframe: selecting this option enables an additional property.
  Destination Iframe
  Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Parameters
Allows sending parameters to the selected object type.

Link



Link
Allows to enter any link.

Destination Type

The available options for opening the object are displayed.

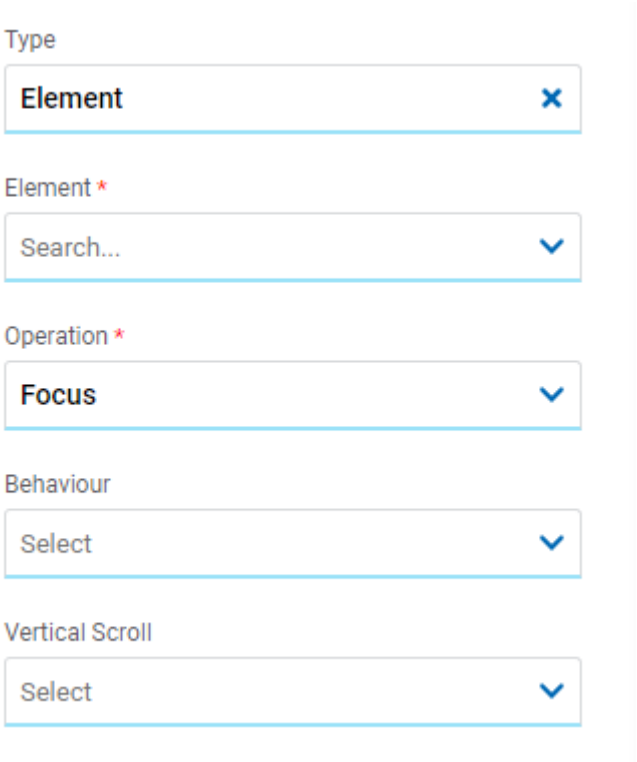
Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  Modal Horizontal Size
  Defines its width.
  Modal Vertical Size
  Defines its height.
- Iframe: selecting this option enables an additional property.
  Destination Iframe
  Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Parameters
Allows sending parameters to the selected object type.

Element



Element
The modeled elements in the object are displayed.

Operation
Defines the operation made when selecting the element.

Possible Values

- Focus
- Show
- Hide
- Show/Hide

Behaviour
Establishes the transition of the animation. It is only modelable if the "Focus" operation was selected.

Possible Values

- Auto
- Smooth

Vertical Scroll
It is only modelable if the "Focus" operation was selected.
Possible Values

- Start: Moves to the start of the selected element.
- Closest: Moves to the closest position to the selected element from the element the event fires.
- Center: Moves to the center of the selected element.
- End: Moves to the end of the selected element.

Repeater



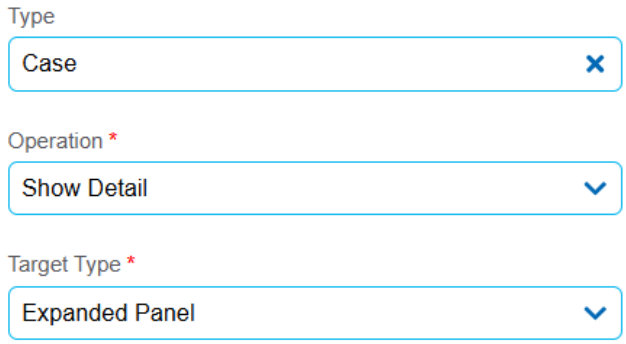
Repeater
The modeled repeaters from the object are displayed.

Operation
Defines the operation made when selecting the element.

Possible Values

- New: Allows for the creation of an item within the repeater. If this option is selected the additional properties are enabled.
  Creation Order
  It indicates in which position the new item of the container is added.
  Possible Values
    - Home
    - End
- Delete: Allows for the deletion of an item within the repeater.

Case



Case
Show case information.

Operation
Defines the operation made when selecting the element.

Possible Values

- Show Detail
- Show Graphic Execution

Destination Type
The available options for opening the object are displayed.

Possible Values

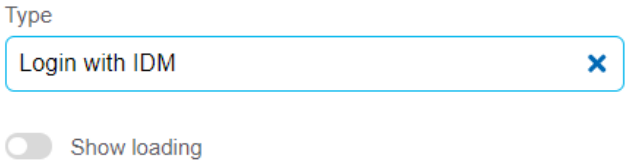
- Expanded Panel
- Modal: if this option is selected the additional properties are enabled.
  Modal Horizontal Size
  Define its width.
  Modal Vertical Size
  Define its height.

Back



It allows associating the event to go back in the browser to the element.

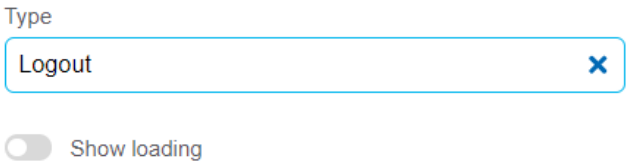
LogIn with IDM



Allows login with IDM.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

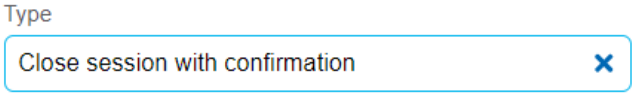
Logout



Allows logging the user out.

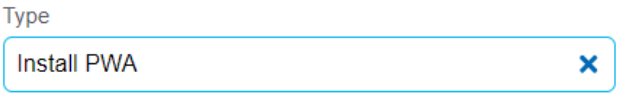
Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Close Session with Confirmation



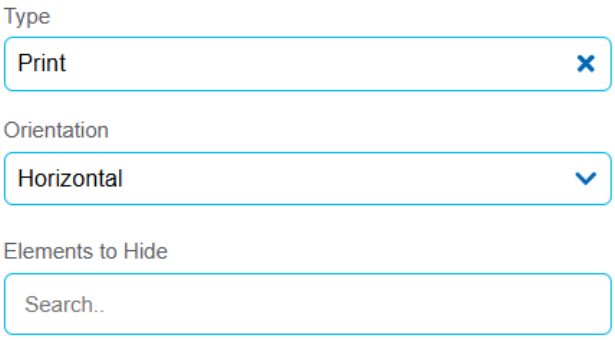
Displays a confirmation modal to logout the user.

Install PWA



Allows to install the application in the browser.

Print



This type of hyperlink associated with an element allows printing the content of the entity or page that is being executed.

Orientation

Determines the layout of the content to be printed.

Possible Values

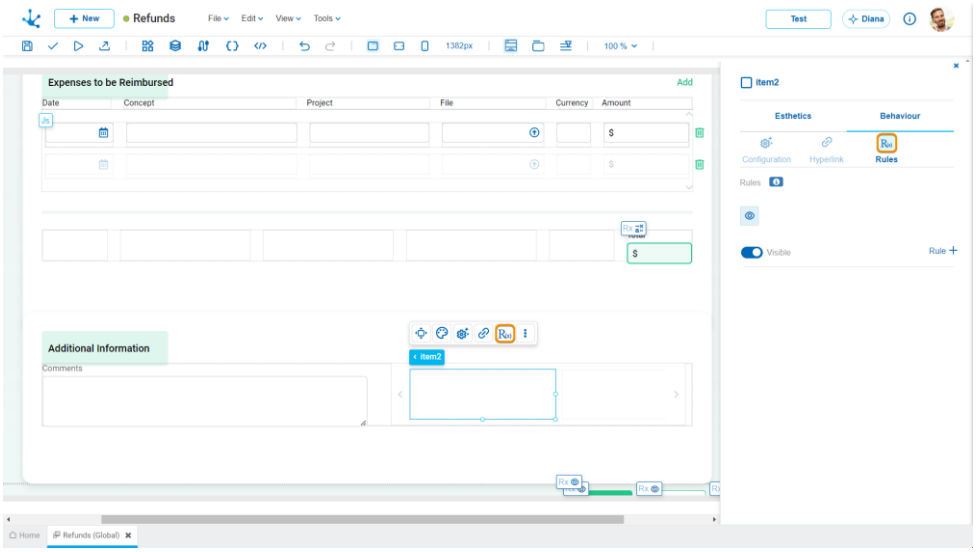
- Horizontal
- Vertical

Elements to hide

It allows selecting which elements of the entity or page should not be included in the printout.

Rule Properties

The rule properties panel of an element opens when clicking the icon  of its context menu.



Properties

Rules

Embedded rules on behavior, validation and calculation can be defined, associated with an element, by using the wizard (ctrl + space).

Shows syntax examples for writing the rules.

Visible

Indicates whether the element is visible. If this property is not checked, the element is not displayed in the page.

Visible (default) Not visible

Rule + Opens an edit area where a rule to determine the visibility condition can be defined. If a rule is defined, the icon is displayed with light blue borders.

Operations when defining the rule:

Saves the new or modified rule

Cancels the operation

Operations once the rule is defined:

Edits the existing rule

Deletes the rule

Events

Items allow the use of different events.

Event	Description
onMouseIn()	It is executed when the cursor is positioned on the element.
onMouseOut()	It is executed when the cursor moves out of the element.
onClick()	It is executed when clicking on the element.
onDoubleClick()	It is executed when clicking twice on the element.
onInit()	It is executed before the element is loaded.
afterViewInit()	It is executed after viewing the element.
onViewportEnter()	It is executed when the element is visible.
onViewportLeave()	It is executed when the element is no longer visible.

Icon

It allows the incorporate of any type of icon.

The element properties are represented by icons in its context menu, where its operations are also available.

Subtypes

When selecting the option "Icon" from the icon  in the top toolbar, a list opens with the different subtypes of this element, which can be dragged into the modeling area. Each subtype has the element's properties modeled and predefined in a specific way.

- Small Icon
- Medium Icon
- Large Icon

Classification of Properties

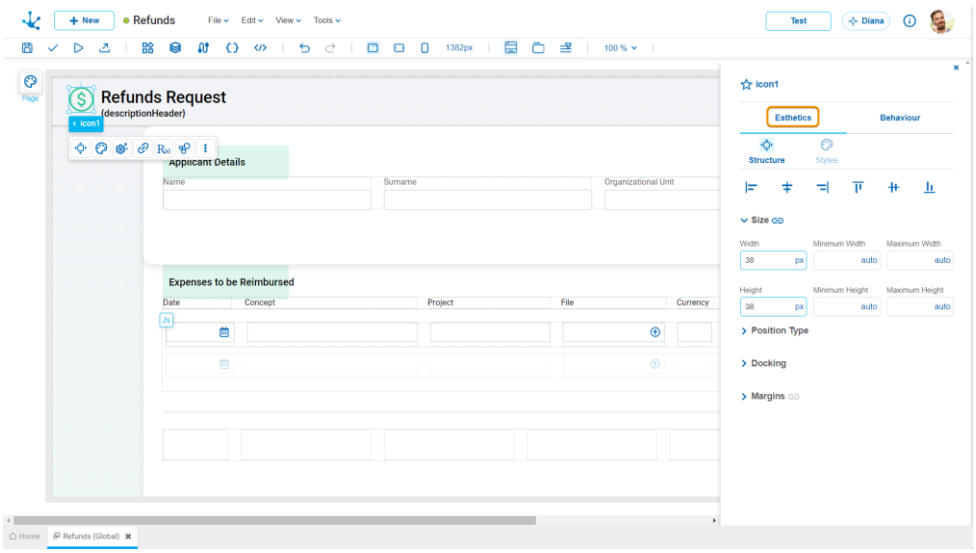
Properties are divided into two groups: esthetic properties and behavior properties.

Esthetic Properties

In the esthetic properties panel, the following are grouped:

- Structure Properties

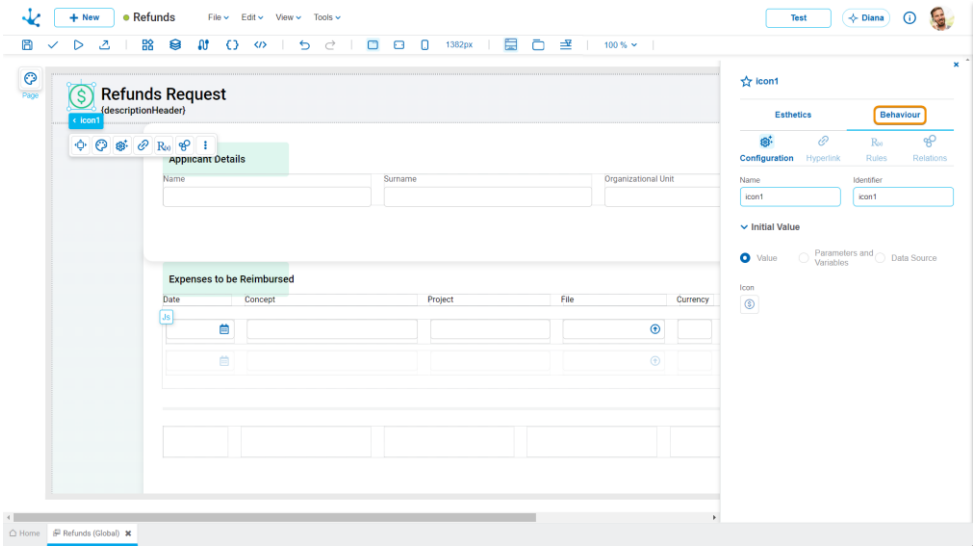
- Style Properties



Behavior Properties

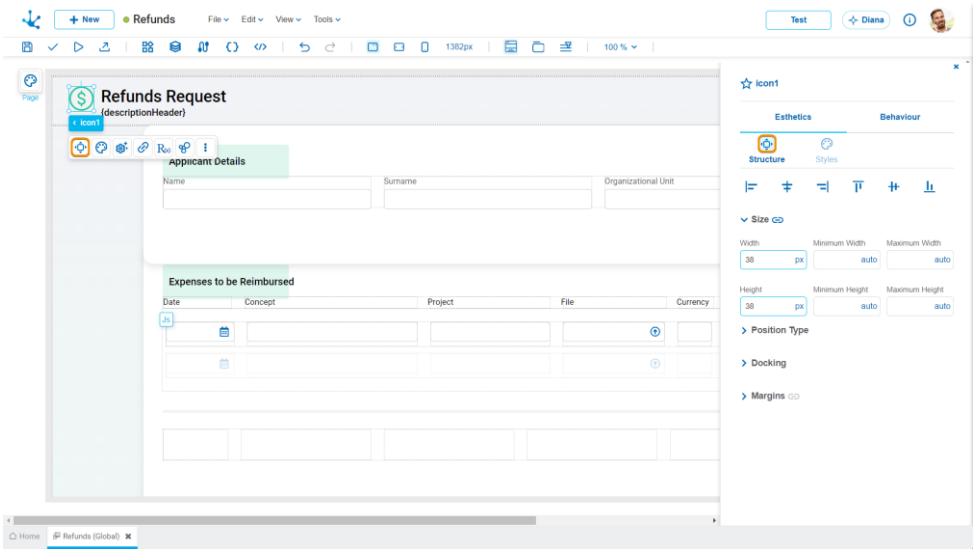
In the behavior properties panel, the following are grouped:

- Configuration Properties
- Hyperlink Properties
- Rule Properties
- Relation Properties



Structure Properties

The structure properties panel of an element opens when selecting the icon  of its context menu.

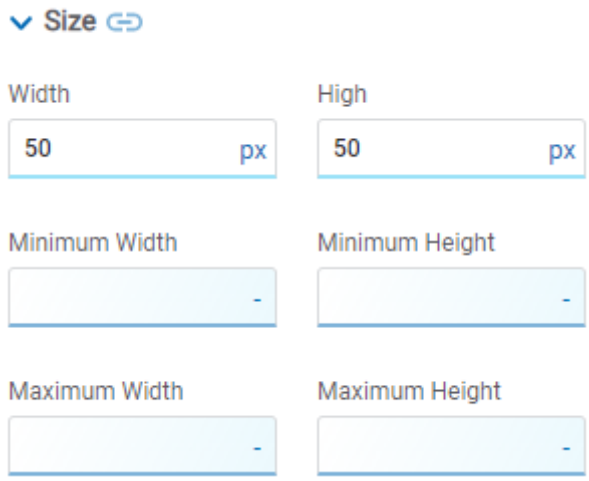


Alignment

Allows to align the element by selecting one of the available icons.

- Align to the left.
- Align to horizontal center.
- Align to the right.
- Align up.
- Align vertical center.
- Align bottom.

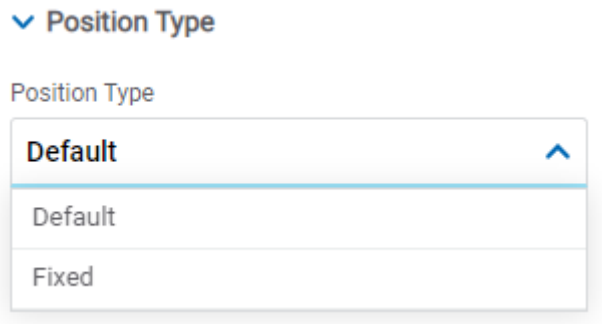
Size



It allows the input of Width and Height. The value entered in one of the properties is automatically copied to the other. These properties are expressed in pixels.

Position Type

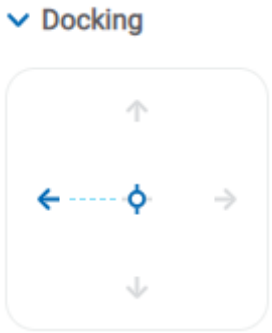
It determines if at the time of execution the element remains fixed on the page or if it moves as the scroll bar moves forward or backward.



Possible Values

- Default: The element has a relative position with respect to the superior element where it was placed (container or section).
- Fixed: Fixes the element to the page so that it is always visible in the same position. It allows to increase the visibility of important content.

Docking



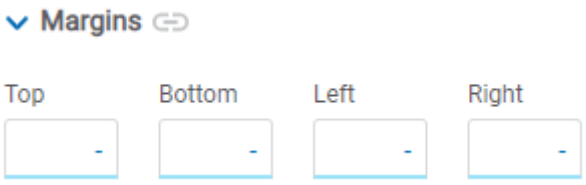
Determines the vertical and horizontal position of elements within the page section or container, when the screen is resized.

When an element is added or moved within the section or container, the modeler automatically docks it to the nearest corners or edges. It can also be configured by clicking on the arrows of the side to be docked to or on the center.

Once the element is docked, its exact position is indicated in the Margins property. The direction of the docking arrows determine the values that are automatically displayed for margins.

The docking position is indicated in the modeling area by dotted lines on the element.

Margins



It allows to define the distance of elements from the borders of their top element. The behavior of margins depends on the docking of the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Top
Distance to the top border of the highest ranking element.

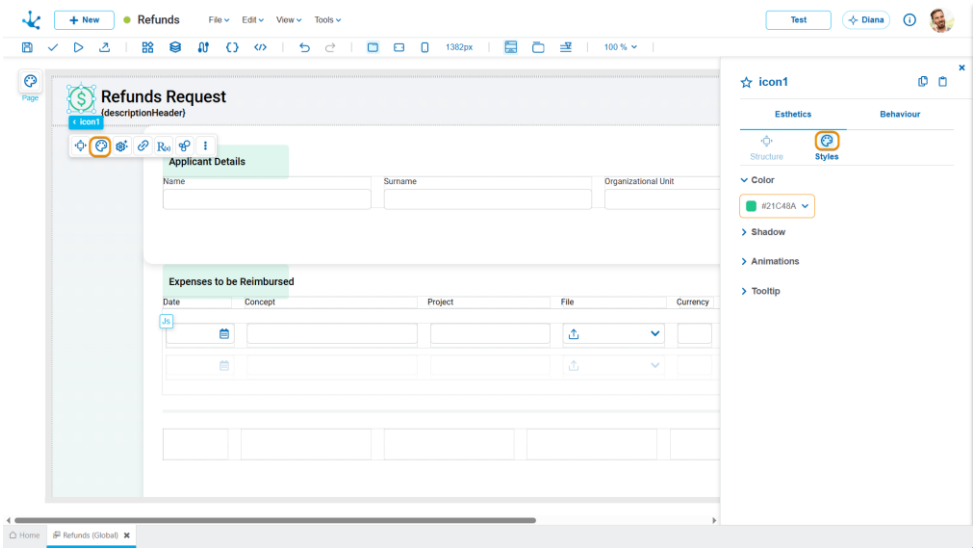
Bottom
Distance to the bottom border of the highest ranking element.

Left
Distance to the left border of the highest ranking element.

Right
Distance to the right border of the highest ranking element.

Allows the value entered in one of the margins to be copied to the other ones automatically.
Allows to indicate different values for each margin.

The style properties panel of an element opens when selecting the icon of its context menu.



Styles

It allows copying the entire current configuration of the style panel for the selected element (Ctrl+Alt+C).

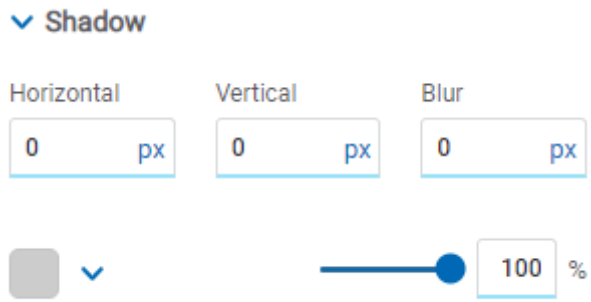
It allows applying the previously copied configuration to another element of the same type (Ctrl+Alt+V).

Color



It allows to define the color of the element.

Shadow



Allows defining a shadow effect around the element.

Horizontal
Horizontal size of the shadow to the right of the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

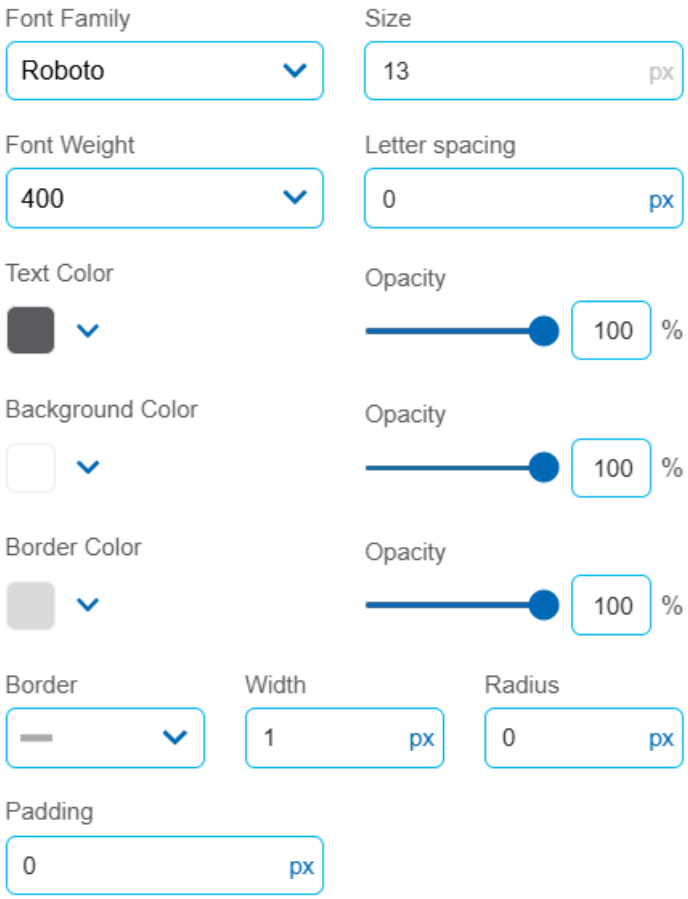
Vertical
Vertical size of the shadow below the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Blur
The greater this value is, the greater and lighter the shadow becomes. If not specified, its value is 0 and the shadow border is darker. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Color
Defines the color of the shadow.

Transparency
Defines the transparency of the shadow.

Tooltip



Source

It allows customizing the font, size, thickness and letter spacing of the pop-up message.

Colors

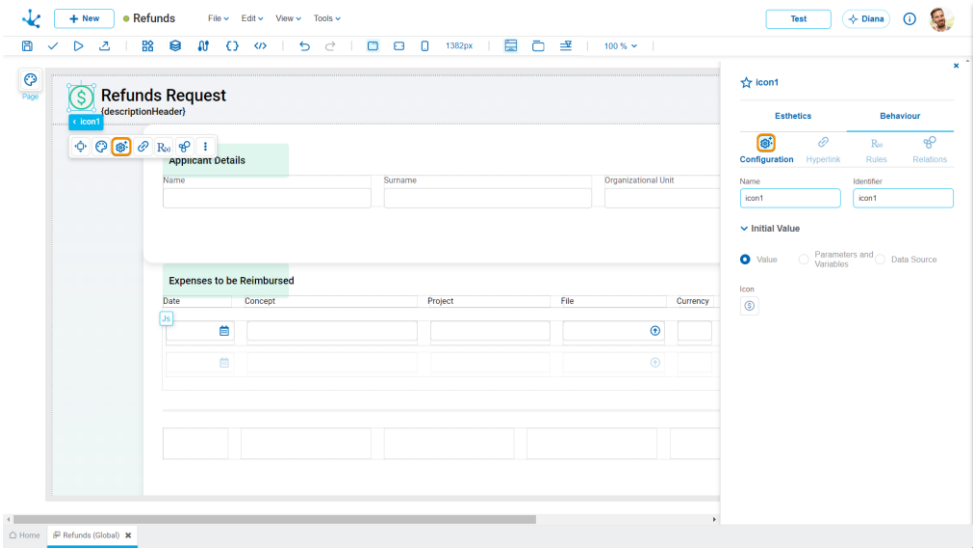
The color and opacity of the text, background and border can be defined. Borders are set uniformly for all sides of the tooltip.

Border

The border type, width, and border radius can be configured.

Configuration Properties

The configuration properties panel of an element opens when clicking the icon  of its context menu.



Name
Name used to reference the element during modeling.

Identifier
Uniquely identifies the element. It is used in the Javascript SDK.

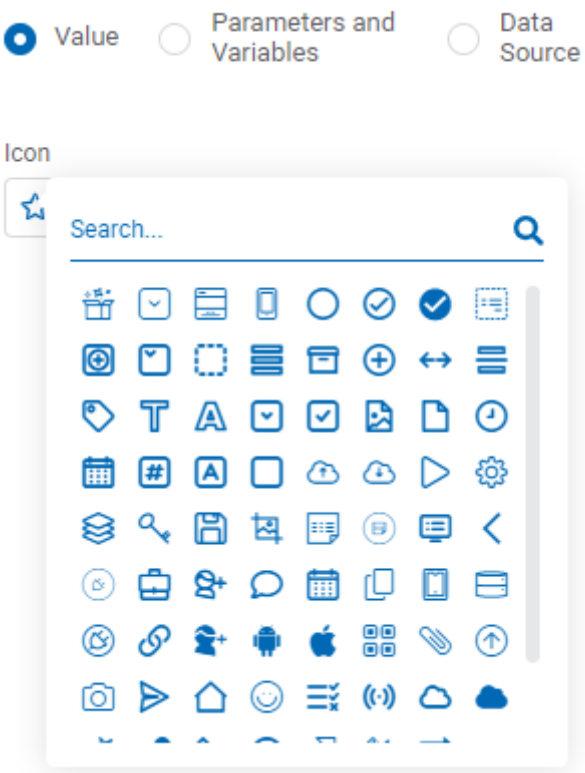
Tooltip
Pop-up message that appears when hovering over an element, providing additional information.



Text
Content of the message being displayed.

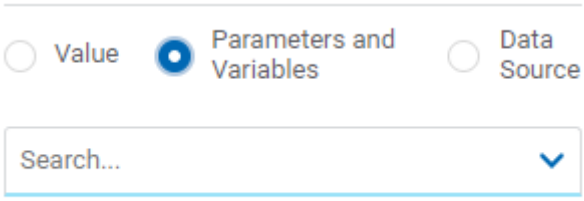
Direction
Position of the message in relation to the element (top, bottom, left, or right).

Initial Value



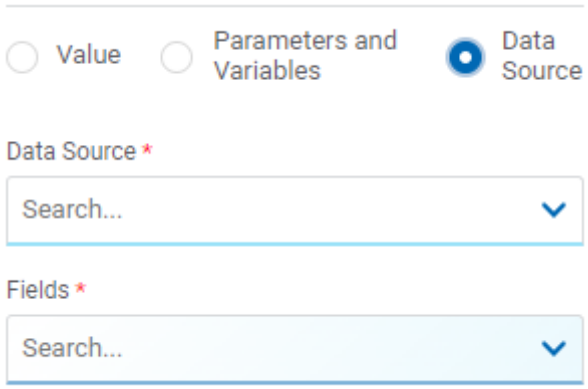
It allows selecting an icon from a palette.

Parameters and Variables



Parameters and Variables
Allows to select a parameter or a variable from, whose value is displayed in the element.
This value can be the code that represents the icon or a valid url.

Data Source



Data Source

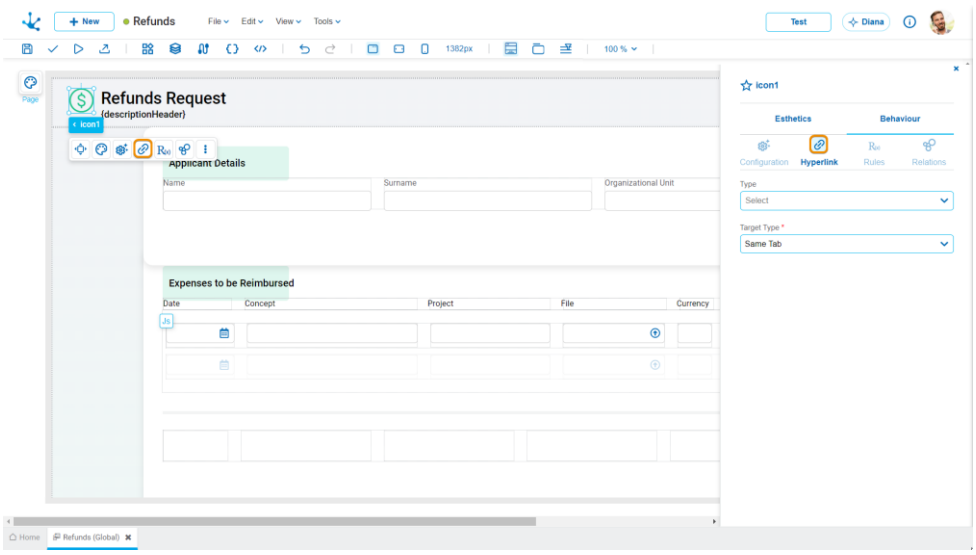
It allows selecting a previously defined data source within the object.

Fields

It allows selecting a field from the data retrieved in the chosen data source, whose value is displayed in the element.

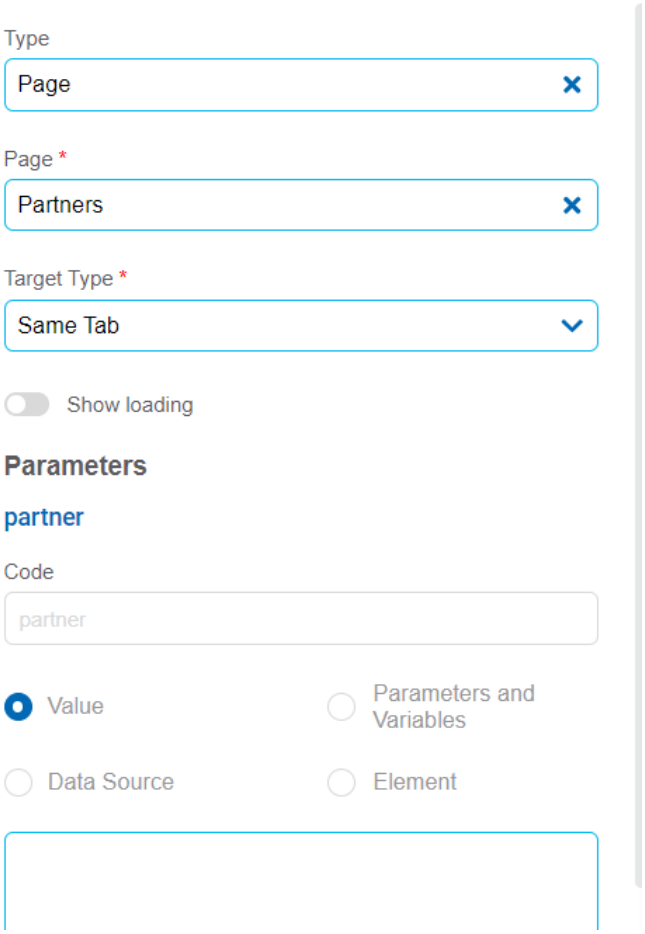
Hyperlink Properties

The hyperlink properties panel of an element opens when clicking the icon  of its context menu.



Define the behavior of the element when selecting it. Different properties are enabled depending on the type of object selected.

Page



Page

The pages modeled in the environment are displayed.

Destination Type

The available options for opening the object are displayed.

Possible Values

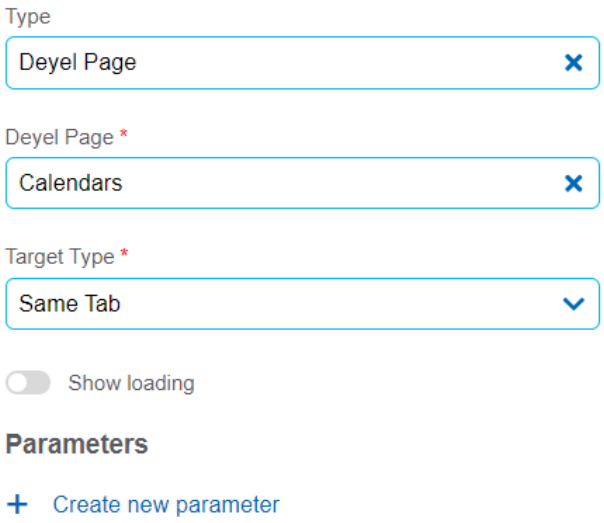
- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  - Modal Horizontal Size
    Defines its width.
  - Modal Vertical Size
    Defines its height.
- Iframe: selecting this option enables an additional property.
  - Destination Iframe
    Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading

Shows the loading icon and disables the user interaction until the redirection is over.

Parameters
If there are parameters, their value can be specified with text, parameters and variables or data source.

Deyel Page



Deyel Page
The pages belonging to **Deyel** are displayed.

Destination Type
The available options for opening the object are displayed.

Possible Values

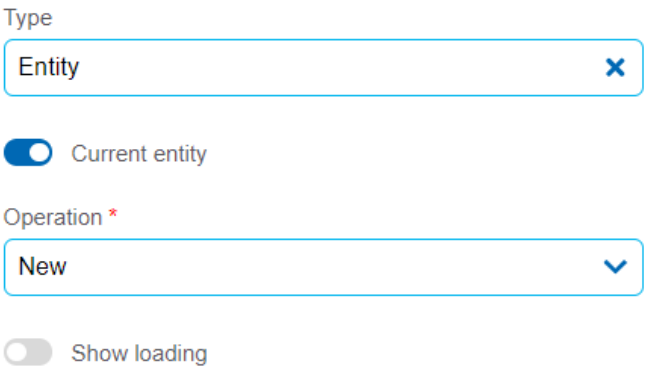
- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  - Modal Horizontal Size
    Defines its width.
  - Modal Vertical Size
    Defines its height.
- Iframe: selecting this option enables an additional property.
  - Destination Iframe
    Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Parameters

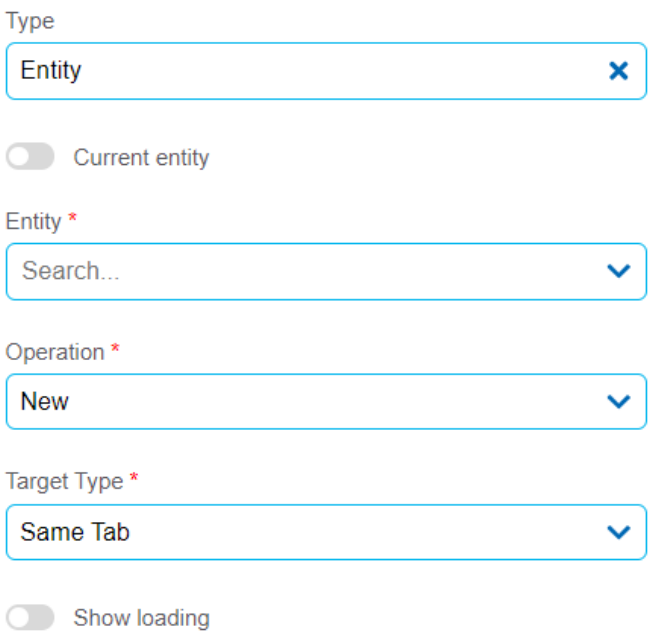
Allows sending parameters to the selected object type.

Entity



Current Entity
If this property is checked, the operation applies directly to the entity being modeled.

Operation
Defines the operation made when selecting the object.
Possible Values
- Create: Indicates that the panel of the current entity is opened to create an instance.
- Update: Indicates that the panel of the current entity is opened to update an instance.
- Delete: Indicates that the panel of the current entity is opened to delete an instance.
- Send Email: It is visible if the property Send Emails is enabled. Clicking on the modeled element opens a panel during instance execution to send an email. The panel size is adjusted with the properties Modal Width and Modal Height.
- Emails: Visible if the Send Emails property is enabled. Clicking on the modeled element opens a panel during instance execution to manage sent emails. The panel is displayed in the iframe defined in the Target Iframe property.



Entity
If the property Current Entity, is not checked, the entities modeled in the environment are displayed.

Operation
Defines the operation made when selecting the object.

Possible Values

- New: The panel of the entity selected in the previous property is opened to create an instance.
- Update The panel of the entity selected in the previous property is opened to update an instance.
- Show: The panel of the entity selected in the previous property is opened to show an instance.
- Grid: Indicates that the grid of the entity selected in the previous property is opened.

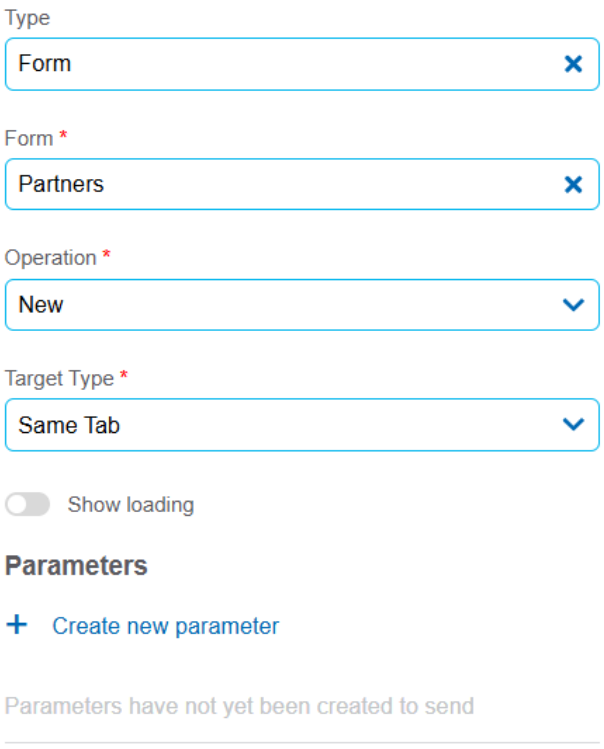
Destination Type
The available options for opening the object are displayed.

Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  Modal Horizontal Size
  Defines its width.
  Modal Vertical Size
  Defines its height.
- Iframe: selecting this option enables an additional property.
  Destination Iframe
  Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Forms



Forms
The forms modeled in the environment are displayed.

Operation
Defines the operation made when selecting the object.

Possible Values

- Create: Indicates that the panel of the object selected in the previous property is opened for the creation of an instance.
- Update: Indicates that the panel of the object selected in the previous property is opened for the update of an instance.
- Show: Indicates that the panel of the object selected in the previous property is opened for the query of an instance.
- Grid: Indicates that the grid of the object selected in the previous property is opened.

Destination Type
The available options for opening the object are displayed.

Possible Values

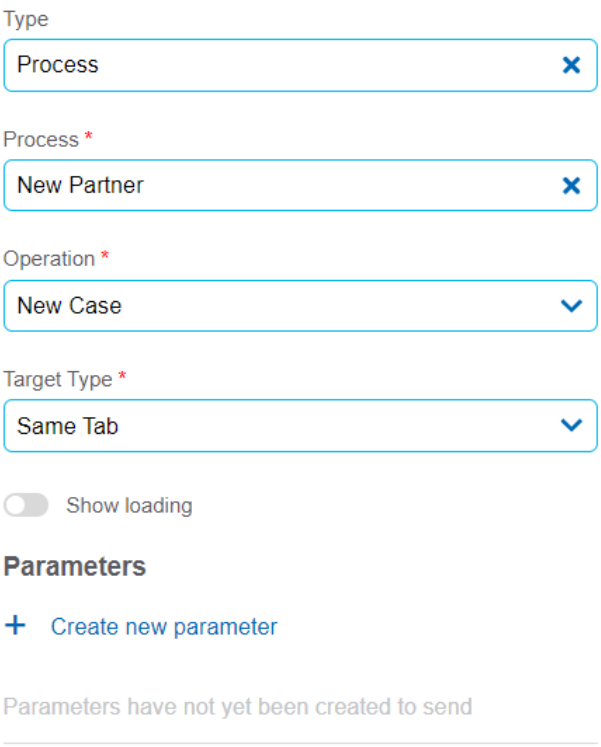
- Same Tab

- New Window
- Modal: selecting this option enables additional properties.
  Modal Horizontal Size
  Defines its width.
  Modal Vertical Size
  Defines its height.
- Iframe: selecting this option enables an additional property.
  Destination Iframe
  Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Parameters
Allows sending parameters to the selected object type.

Process



Process
The processes modeled in the environment are displayed.

Operation
Defines the operation made when selecting the object.

Destination Type
The available options for opening the object are displayed.

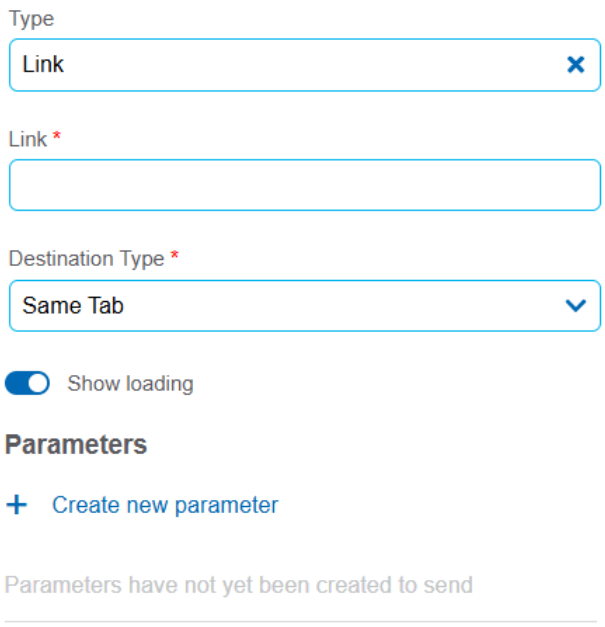
Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  Modal Horizontal Size
  Defines its width.
  Modal Vertical Size
  Defines its height.
- Iframe: selecting this option enables an additional property.
  Destination Iframe
  Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Parameters
Allows sending parameters to the selected object type.

Link



Link
Allows to enter any link.

Destination Type

The available options for opening the object are displayed.

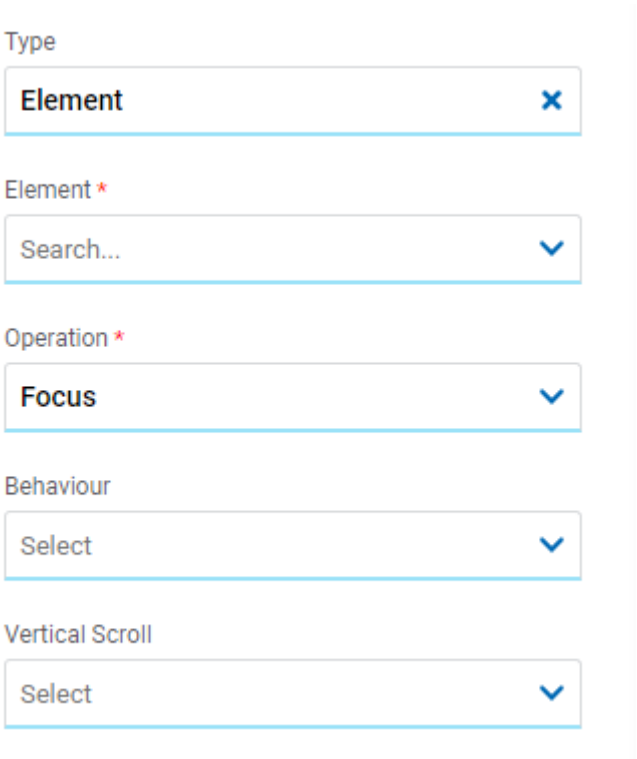
Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  Modal Horizontal Size
  Defines its width.
  Modal Vertical Size
  Defines its height.
- Iframe: selecting this option enables an additional property.
  Destination Iframe
  Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Parameters
Allows sending parameters to the selected object type.

Element



Element
The modeled elements in the object are displayed.

Operation
Defines the operation made when selecting the element.

Possible Values

- Focus
- Show
- Hide
- Show/Hide

Behaviour
Establishes the transition of the animation. It is only modelable if the "Focus" operation was selected.

Possible Values

- Auto
- Smooth

Vertical Scroll
It is only modelable if the "Focus" operation was selected.
Possible Values

- Start: Moves to the start of the selected element.
- Closest: Moves to the closest position to the selected element from the element the event fires.
- Center: Moves to the center of the selected element.
- End: Moves to the end of the selected element.

Repeater



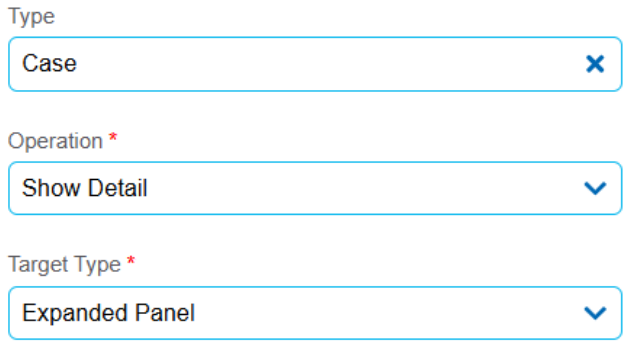
Repeater
The modeled repeaters from the object are displayed.

Operation
Defines the operation made when selecting the element.

Possible Values

- New: Allows for the creation of an item within the repeater. If this option is selected the additional properties are enabled.
  Creation Order
  It indicates in which position the new item of the container is added.
  Possible Values
    - Home
    - End
- Delete: Allows for the deletion of an item within the repeater.

Case



Case
Show case information.

Operation
Defines the operation made when selecting the element.

Possible Values

- Show Detail
- Show Graphic Execution

Destination Type
The available options for opening the object are displayed.

Possible Values

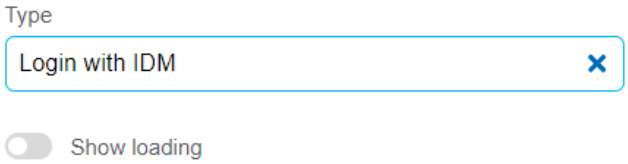
- Expanded Panel
- Modal: if this option is selected the additional properties are enabled.
  Modal Horizontal Size
  Define its width.
  Modal Vertical Size
  Define its height.

Back



It allows associating the event to go back in the browser to the element.

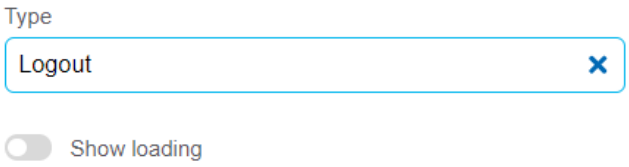
LogIn with IDM



Allows login with IDM.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

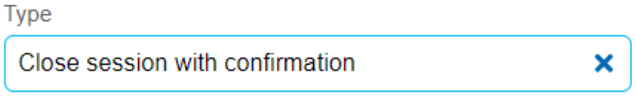
Logout



Allows logging the user out.

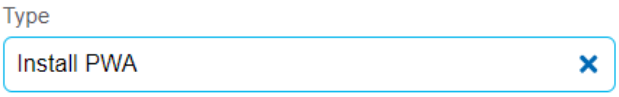
Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Close Session with Confirmation



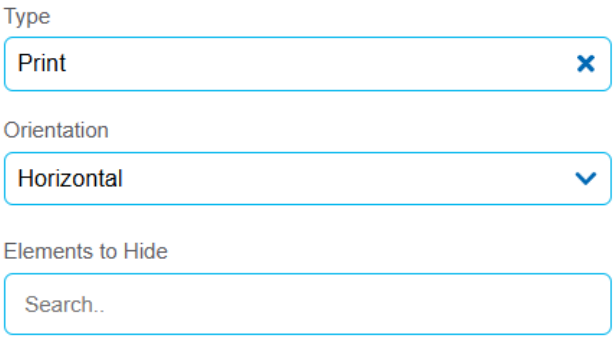
Displays a confirmation modal to logout the user.

Install PWA



Allows to install the application in the browser.

Print



This type of hyperlink associated with an element allows printing the content of the entity or page that is being executed.

Orientation

Determines the layout of the content to be printed.

Possible Values

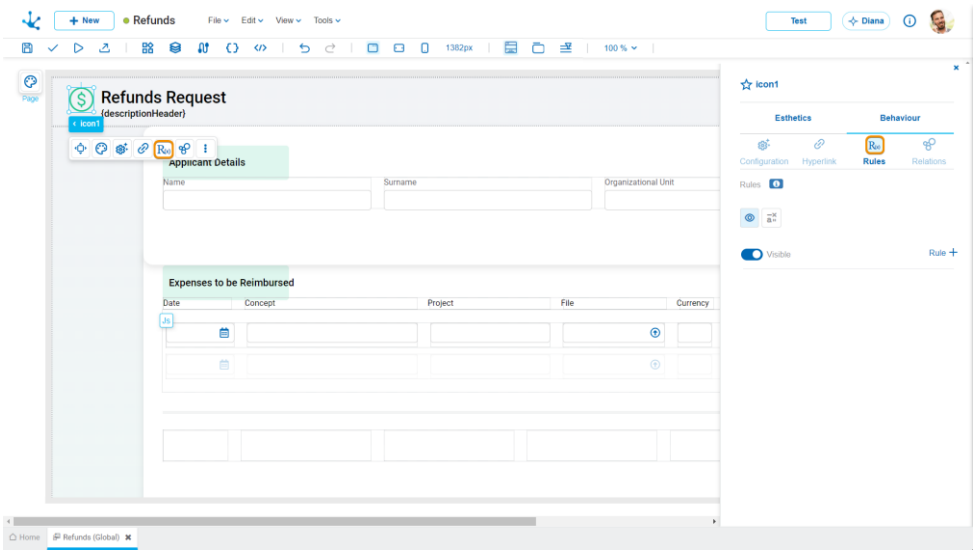
- Horizontal
- Vertical

Elements to hide

It allows selecting which elements of the entity or page should not be included in the printout.

Rule Properties

The rule properties panel of an element opens when clicking the icon  of its context menu.



Properties

Rules

Embedded rules on behavior, validation and calculation can be defined, associated with an element, by using the wizard (ctrl + space).

Shows syntax examples for writing the rules.

Visible

Indicates whether the element is visible. If this property is not checked, the element is not displayed in the page.

Visible (default) Not visible

Rule + Opens an edit area where a rule to determine the visibility condition can be defined. If a rule is defined, the icon is displayed with light blue borders.

Calculation

Rule + Opens an edit area where you can define the expression to be executed to calculate the element value. If a rule is defined, the icon is displayed with light blue borders.

Operations when defining the rule:

Saves the new or modified rule

Cancels the operation

Operations once the rule is defined:

Edits the existing rule

Deletes the rule

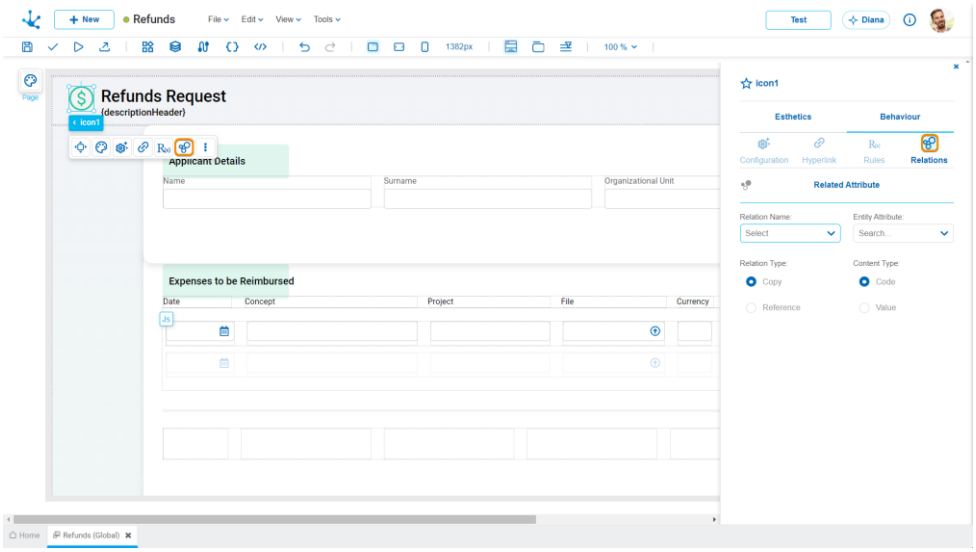
Events

Icons allow the use of an event.

Event	Description
onClick()	It executes when clicking on the element.

Relation Properties

The relation properties panel of an element opens when clicking the icon  of its context menu.



Related Attribute

A related attribute indicates that the field value is retrieved from the attribute value of another entity. One of the relations defined on the entity must be selected, and then it should be specified which attribute of the related entity it links to.

A field can be modeled as a related attribute and it can also have a relation defined to an entity, a value list, or a rule. This allows the value of the related attribute to be retrieved on this field and the functionality provided by the relation to be used.

Properties

Relation Name
The list of relations defined in the entity is displayed, allowing the selection of one of them.

Entity Attribute
The list of attributes of the selected entity compatible with the field type is displayed. Attributes can have a relation modeled with an entity, a value list, or a rule in the selected entity.

Relation Type

Copy

When this property is checked, once the retrieved value is uploaded, it does not automatically update even if the value of the related attribute changes.

Reference

When this property is checked, the field value is subject to the current value of the related attribute, that is, the updated value is automatically displayed.

Content Type

The element content varies depending on whether the Code or Value property is checked when:

- The entity attribute has a modeled relation.
- The field has the option "Entity" selected in the Values Obtained from property.

Code

When this property is checked, if the relation is modeled with a:

- Related entity, the identifier of the entity is displayed.
- Value list, the value list code is displayed.
- Rule, the code returned by the rule's execution is displayed.

Value

When this property is checked, if the relation is modeled with a:

- Related entity, the short description of the entity is displayed.
- Value list, the value list description is displayed.
- Rule, the description returned by the rule's execution is displayed.

Page

Any reusable page can be included in an entity, defining the features by modeling its properties.

The element properties are represented by icons on its context menu, where its operations are also available.

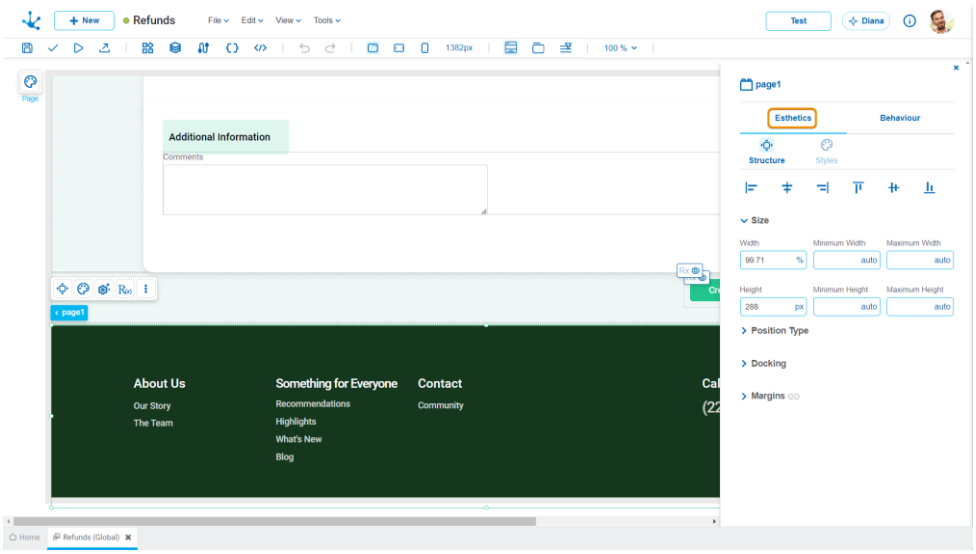
Classification of Properties

Properties are divided into two groups: esthetic properties and behavior properties.

Esthetic Properties

In the esthetic properties panel, the following are grouped:

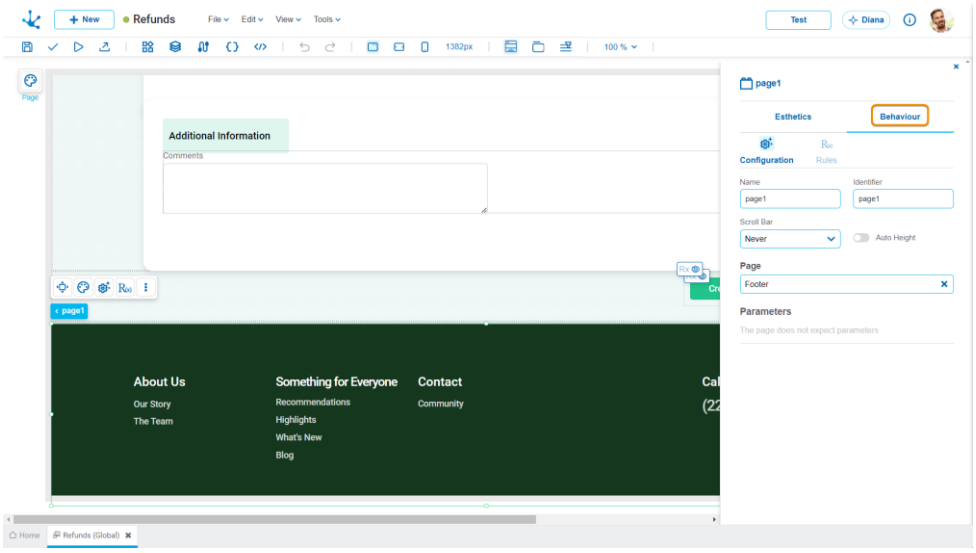
- Structure Properties
- Style Properties



Behavior Properties

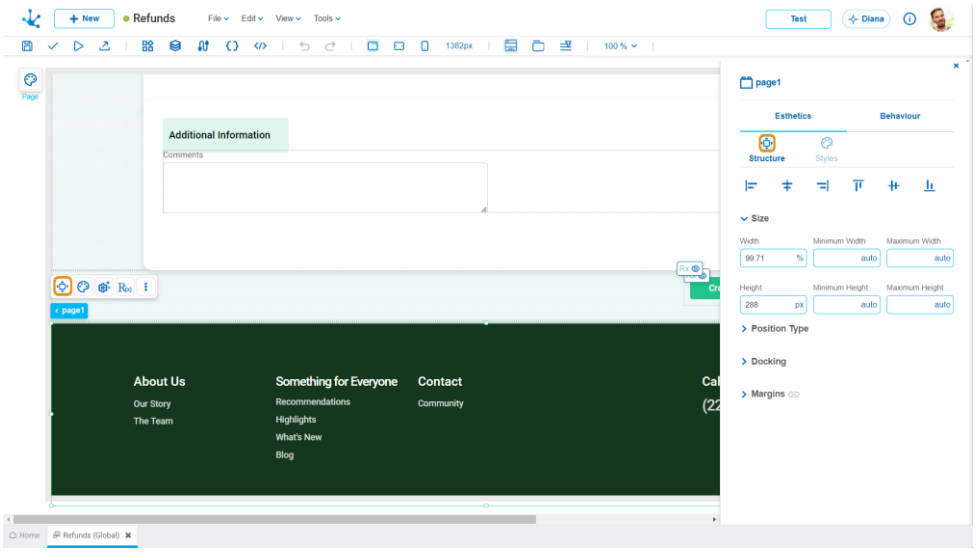
In the behavior properties panel, the following are grouped:

- Configuration Properties
- Rule Properties



Structure Properties

The structure properties panel of an element opens when selecting the icon of its context menu.

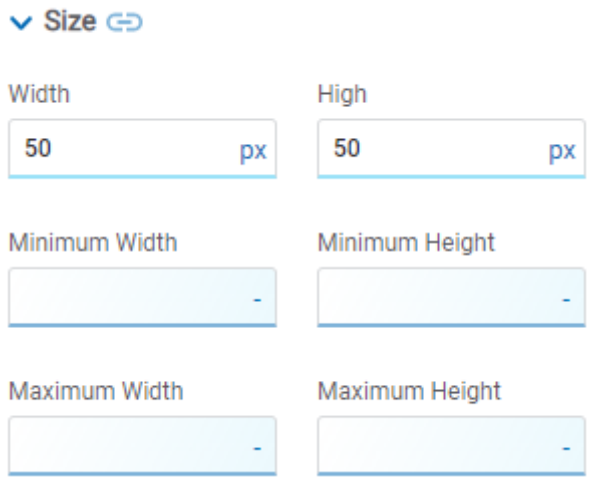


Alignment

Allows to align the element by selecting one of the available icons.

- Align to the left.
- Align to horizontal center.
- Align to the right.
- Align up.
- Align vertical center.
- Align bottom.

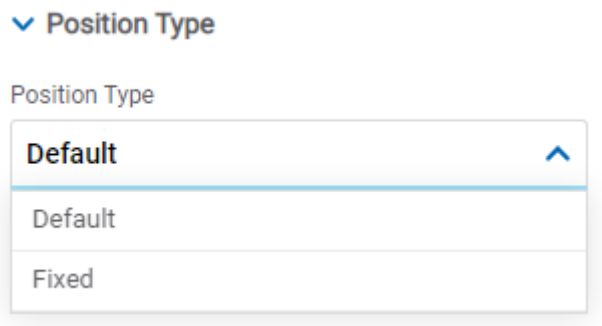
Size



It allows the input of Width and Height. The value entered in one of the properties is automatically copied to the other. These properties are expressed in pixels.

Position Type

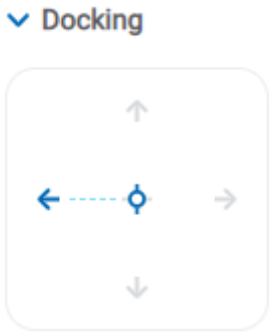
It determines if at the time of execution the element remains fixed on the page or if it moves as the scroll bar moves forward or backward.



Possible Values

- Default: The element has a relative position with respect to the superior element where it was placed (container or section).
- Fixed: Fixes the element to the page so that it is always visible in the same position. It allows to increase the visibility of important content.

Docking



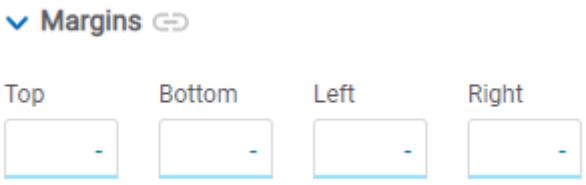
Determines the vertical and horizontal position of elements within the page section or container, when the screen is resized.

When an element is added or moved within the section or container, the modeler automatically docks it to the nearest corners or edges. It can also be configured by clicking on the arrows of the side to be docked to or on the center.

Once the element is docked, its exact position is indicated in the Margins property. The direction of the docking arrows determine the values that are automatically displayed for margins.

The docking position is indicated in the modeling area by dotted lines on the element.

Margins



It allows to define the distance of elements from the borders of their top element. The behavior of margins depends on the docking of the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Top
Distance to the top border of the highest ranking element.

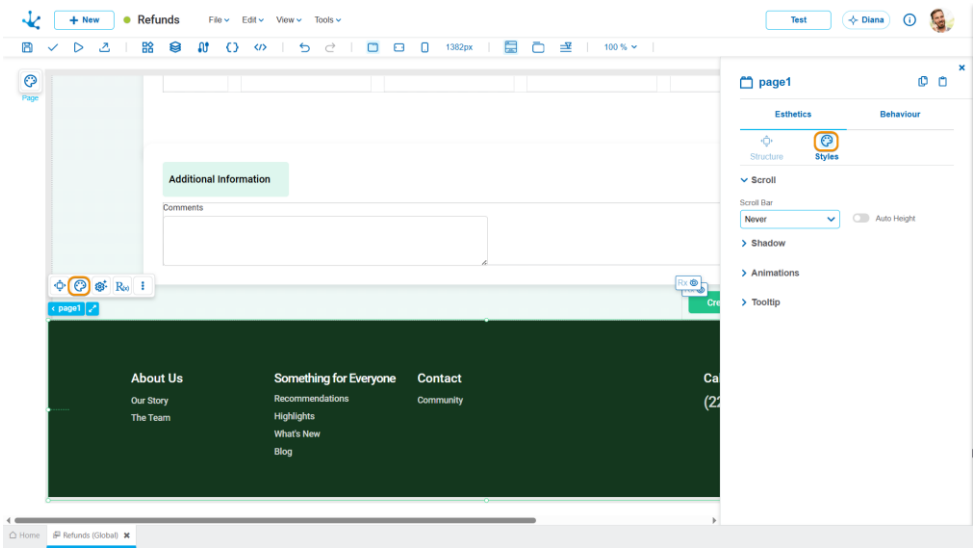
Bottom
Distance to the bottom border of the highest ranking element.

Left
Distance to the left border of the highest ranking element.

Right
Distance to the right border of the highest ranking element.

Allows the value entered in one of the margins to be copied to the other ones automatically.
Allows to indicate different values for each margin.

The style properties panel of an element opens when selecting the icon of its context menu.

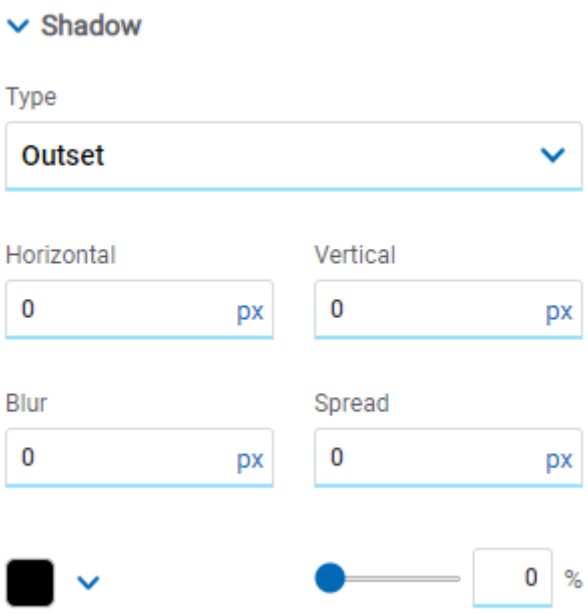


Styles

It allows copying the entire current configuration of the style panel for the selected element (Ctrl+Alt+C).

It allows applying the previously copied configuration to another element of the same type (Ctrl+Alt+V).

Shadow



Allows to define a shadow effect around the element.

Type

Possible Values

- Outset
- Inset

Horizontal
Horizontal size of the shadow to the right of the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Vertical
Vertical size of the shadow below the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

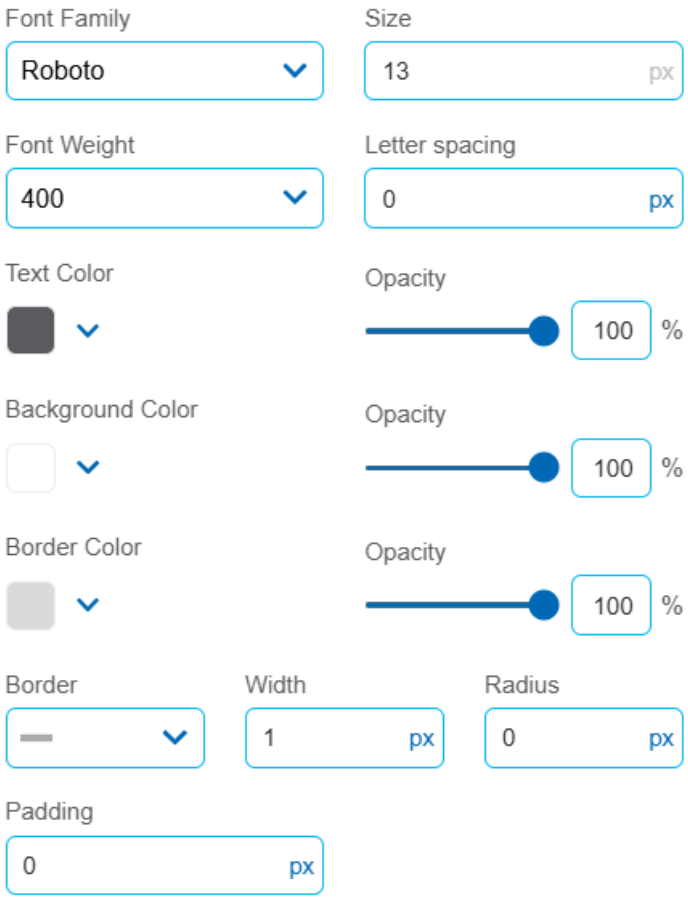
Blur
The greater this value is, the greater and lighter the shadow becomes. If not specified, its value is 0 and the shadow border is darker. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Spread
Positive values will cause the shadow to expand and grow bigger, while negative values will cause the shadow to shrink. If not specified, its value is 0 and the shadow will have the same size as the element.

Color
Defines the color of the shadow.

Transparency
Defines the transparency of the shadow.

Tooltip



Source

It allows customizing the font, size, thickness and letter spacing of the pop-up message.

Colors

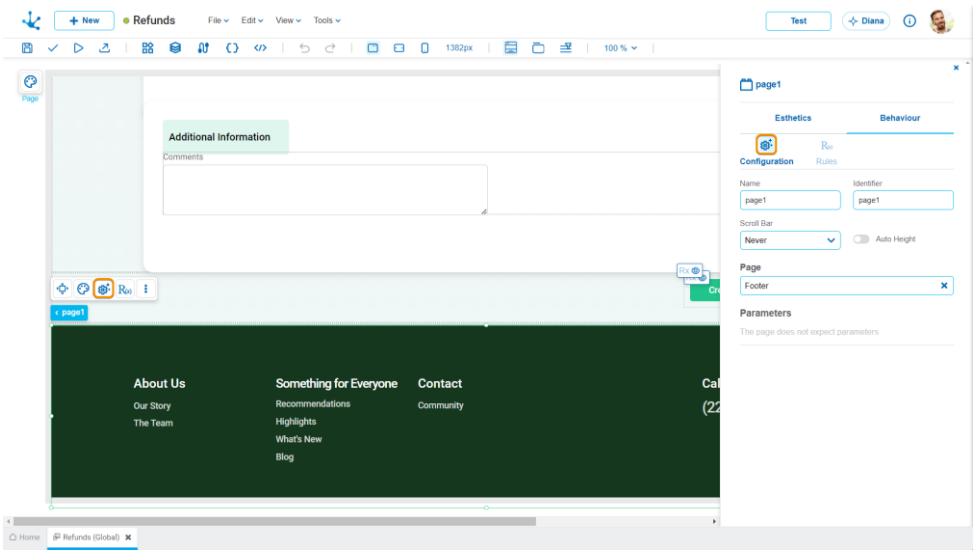
The color and opacity of the text, background and border can be defined. Borders are set uniformly for all sides of the tooltip.

Border

The border type, width, and border radius can be configured.

Configuration Properties

The configuration properties panel of an element opens when clicking the icon  of its context menu.



Name
Name used to reference the element during modeling.

Identifier
Uniquely identifies the element. It is used in the Javascript SDK.

Tooltip
Pop-up message that appears when hovering over an element, providing additional information.



Text
Content of the message being displayed.

Direction
Position of the message in relation to the element (top, bottom, left, or right).

Scroll Bar
It allows defining how the scroll bar behaves inside the reusable page container.

Possible Values

- Automatic: The scroll bar appears only when the contents of the container exceed its visible size.
- Never: The scroll bar does not appear, regardless of the content size.

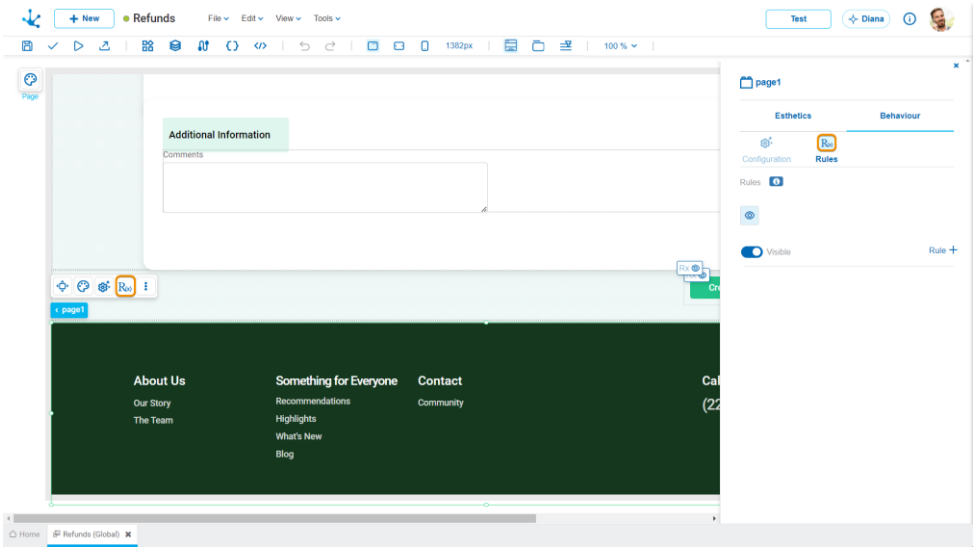
Auto Height
Determines whether the reusable page container automatically adjusts its height to fit the size of the content it contains.
If this property is checked, the container dynamically grows or shrinks in height based on the loaded content, eliminating the need for scrolling. In this case, the value of the Scroll Bar property automatically adjusts to "Never", since the container does not require sliding.

Page
The pages modeled in the environment are displayed.

Parameters
It allows sending parameters to the selected page.

Rule Properties

The rule properties panel of an element opens when clicking the icon  of its context menu.



Properties

Rules
Embedded rules on behavior, validation and calculation can be defined, associated with an element, by using the wizard (ctrl + space).
Shows syntax examples for writing the rules.

Visible
Indicates whether the element is visible. If this property is not checked, the element is not displayed in the page.
Visible (default) Not visible
Rule + Opens an edit area where a rule to determine the visibility condition can be defined. If a rule is defined, the icon is displayed with light blue borders.

Operations when defining the rule:

Saves the new or modified rule

Cancels the operation

Operations once the rule is defined:

Edits the existing rule

Deletes the rule

Widget

Any widget can be included in an entity by defining its features through the modeling of its properties.

The element properties are represented by icons on its context menu, where its operations are also available.

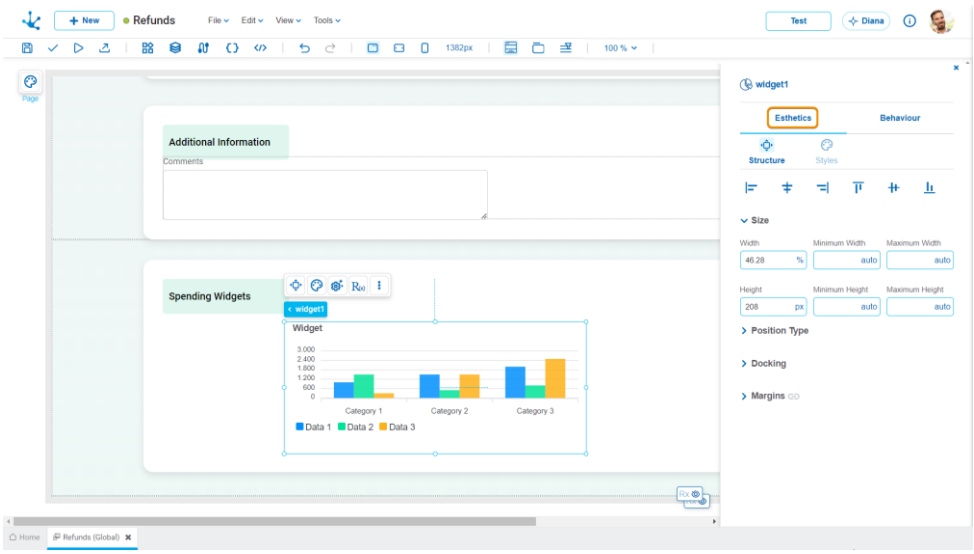
Classification of Properties

Properties are divided into two groups: esthetic properties and behavior properties.

Esthetic Properties

In the esthetic properties panel, the following are grouped:

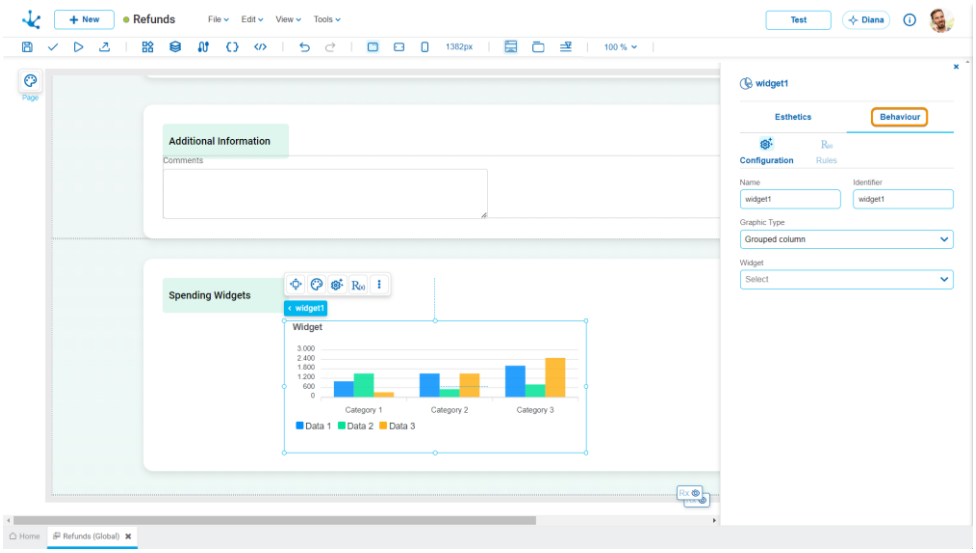
- Structure Properties
- Style Properties



Behavior Properties

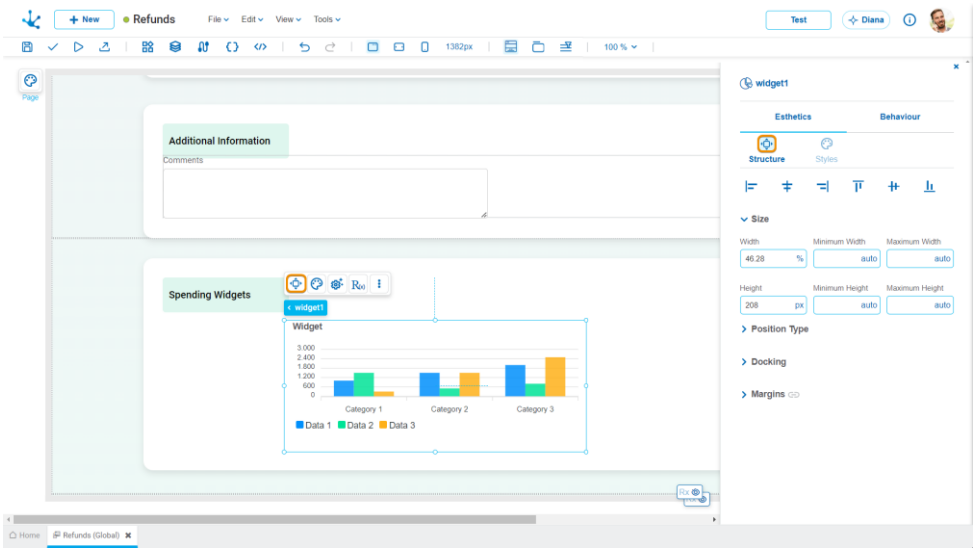
In the behavior properties panel, the following are grouped:

- Configuration Properties
- Rule Properties



Structure Properties

The structure properties panel of an element opens when selecting the icon of its context menu.



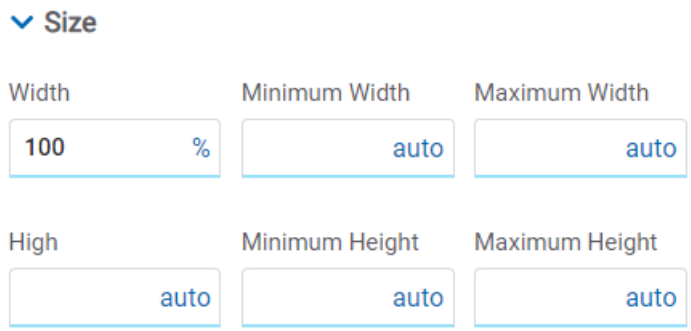
Alignment

Allows to align the element by selecting one of the available icons.

- Align to the left.
- Align to horizontal center.
- Align to the right.

- Align up.
- Align vertical center.
- Align bottom.

Size

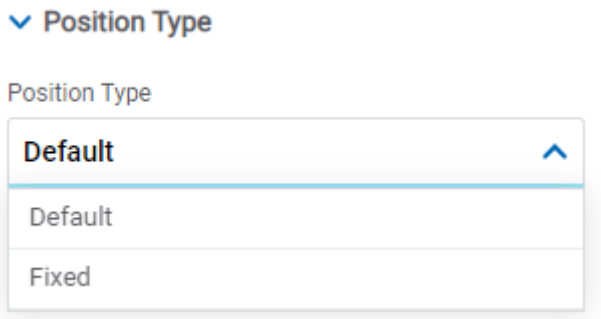


All size properties can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh). Additionally, for Width and Height properties, the "auto" option is added.

> *If selecting percentage (%) for the width and height properties, the size is calculated relative to the top element.*

Position Type

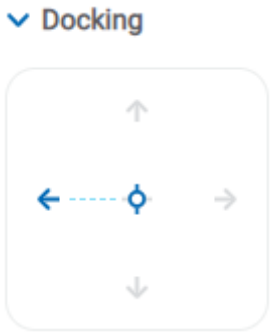
It determines if at the time of execution the element remains fixed on the page or if it moves as the scroll bar moves forward or backward.



Possible Values

- Default: The element has a relative position with respect to the superior element where it was placed (container or section).
- Fixed: Fixes the element to the page so that it is always visible in the same position. It allows to increase the visibility of important content.

Docking



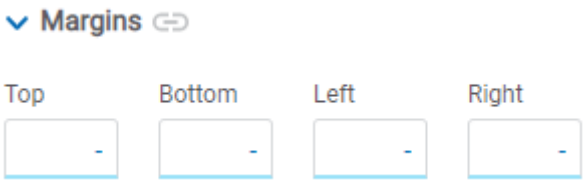
Determines the vertical and horizontal position of elements within the page section or container, when the screen is resized.

When an element is added or moved within the section or container, the modeler automatically docks it to the nearest corners or edges. It can also be configured by clicking on the arrows of the side to be docked to or on the center.

Once the element is docked, its exact position is indicated in the Margins property. The direction of the docking arrows determine the values that are automatically displayed for margins.

The docking position is indicated in the modeling area by dotted lines on the element.

Margins



It allows to define the distance of elements from the borders of their top element. The behavior of margins depends on the docking of the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Top
Distance to the top border of the highest ranking element.

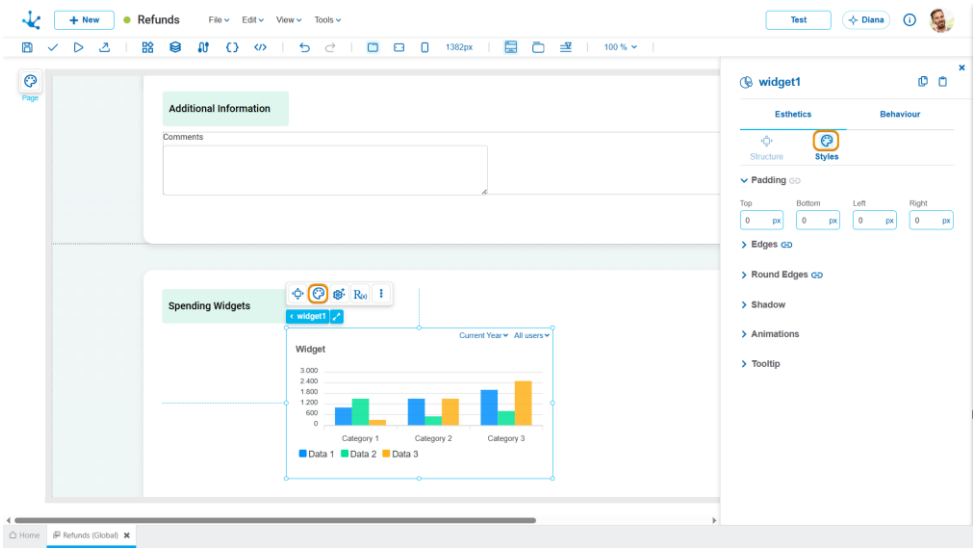
Bottom
Distance to the bottom border of the highest ranking element.

Left
Distance to the left border of the highest ranking element.

Right
Distance to the right border of the highest ranking element.

Allows the value entered in one of the margins to be copied to the other ones automatically.
Allows to indicate different values for each margin.

The style properties panel of an element opens when selecting the icon of its context menu.

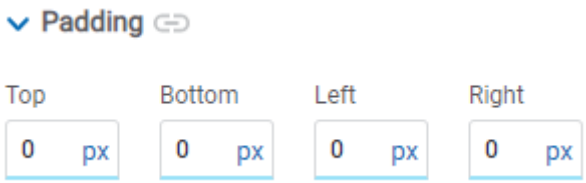


Styles

It allows copying the entire current configuration of the style panel for the selected element (Ctrl+Alt+C).

It allows applying the previously copied configuration to another element of the same type (Ctrl+Alt+V).

Padding

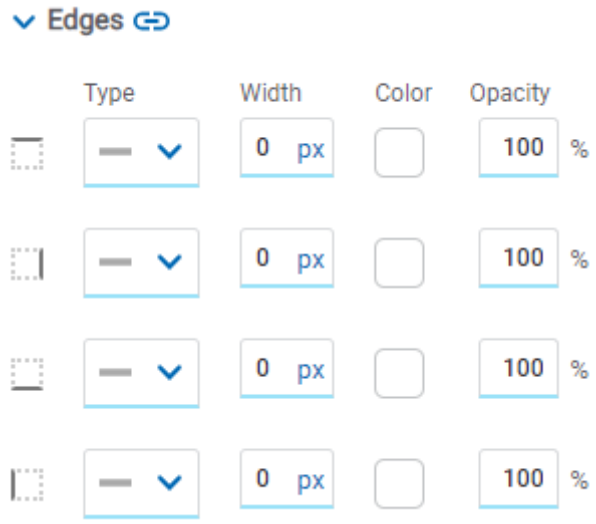


All padding space properties allow to create a space around the element borders (top, bottom, sides) and its content. The padding space configured is inward, while the margin is configured relative to a top element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Allows values entered in one of the paddings to be copied to the other ones automatically.

Allows to indicate different values for each padding.

Edges



It allows to define the style of borders. Each one has its type, width, color and opacity defined, the latter as a percentage.

Allows the value entered in one of the borders to be copied to the other ones automatically.
Allows to indicate different values for each border.

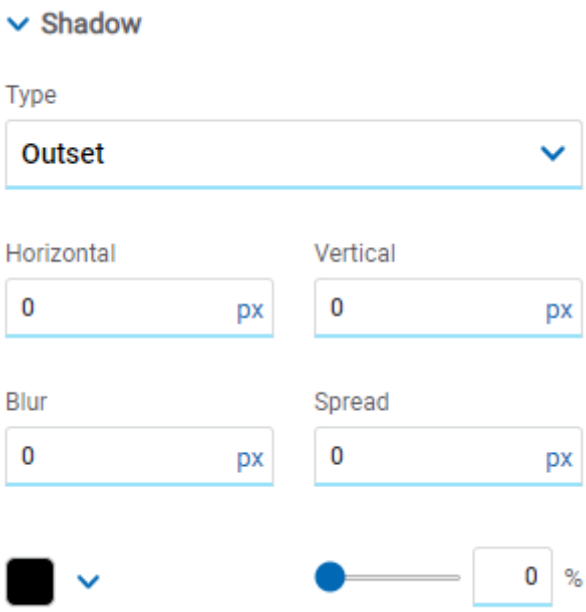
Round Edges



It allows to define the round edges at the corners of elements.
It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Allows the value entered in one of the borders to be copied to the other ones automatically.
Allows to indicate different values for each border.

Shadow



Allows to define a shadow effect around the element.

Type

Possible Values

- Outset
- Inset

Horizontal
Horizontal size of the shadow to the right of the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Vertical
Vertical size of the shadow below the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

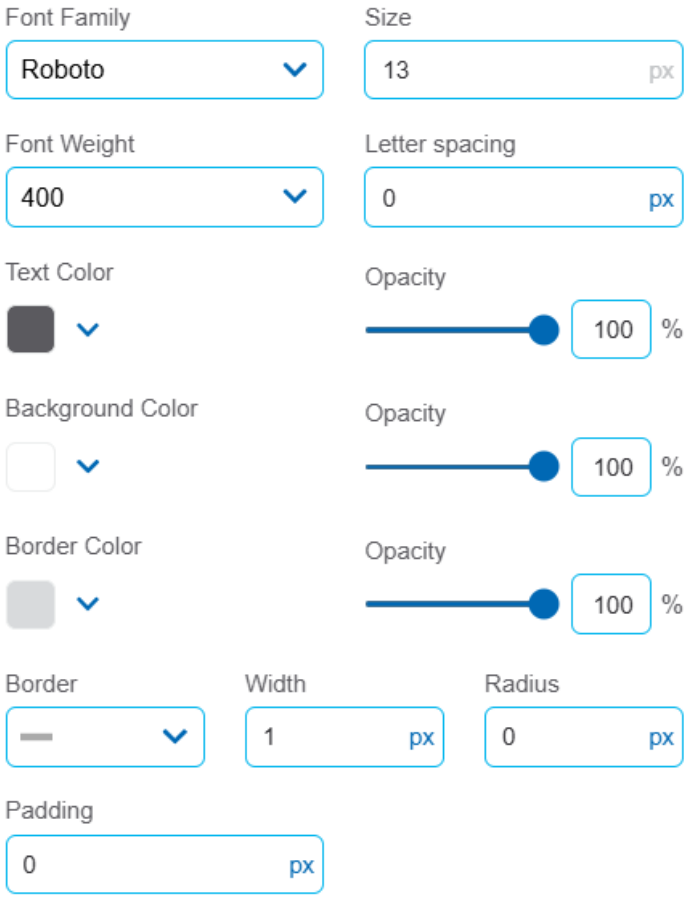
Blur
The greater this value is, the greater and lighter the shadow becomes. If not specified, its value is 0 and the shadow border is darker. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Spread
Positive values will cause the shadow to expand and grow bigger, while negative values will cause the shadow to shrink. If not specified, its value is 0 and the shadow will have the same size as the element.

Color
Defines the color of the shadow.

Transparency
Defines the transparency of the shadow.

Tooltip



Source

It allows customizing the font, size, thickness and letter spacing of the pop-up message.

Colors

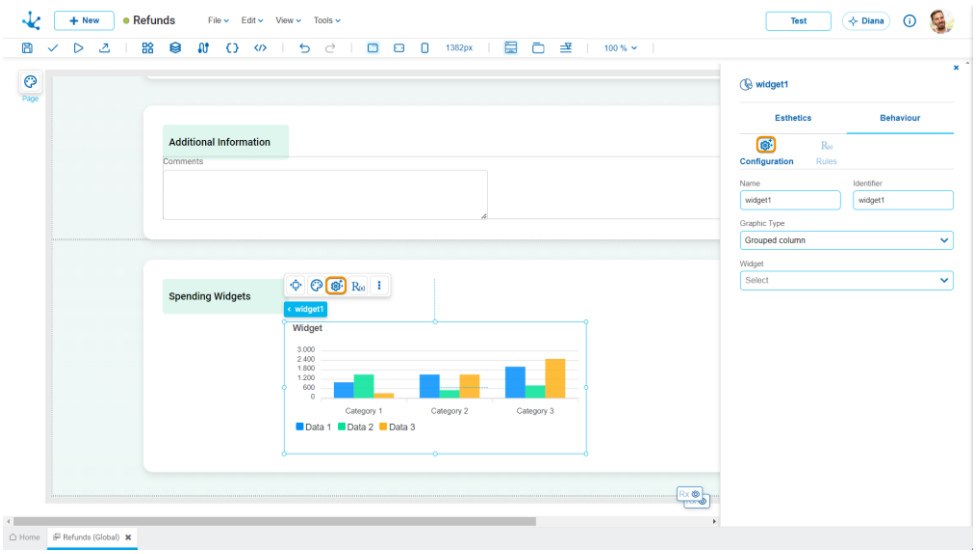
The color and opacity of the text, background and border can be defined. Borders are set uniformly for all sides of the tooltip.

Border

The border type, width, and border radius can be configured.

Configuration Properties

The configuration properties panel of an element opens when clicking the icon  of its context menu.



Name
Name used to reference the element during modeling.

Identifier
Uniquely identifies the element. It is used in the Javascript SDK.

Tooltip
Pop-up message that appears when hovering over an element, providing additional information.



Text
Content of the message being displayed.

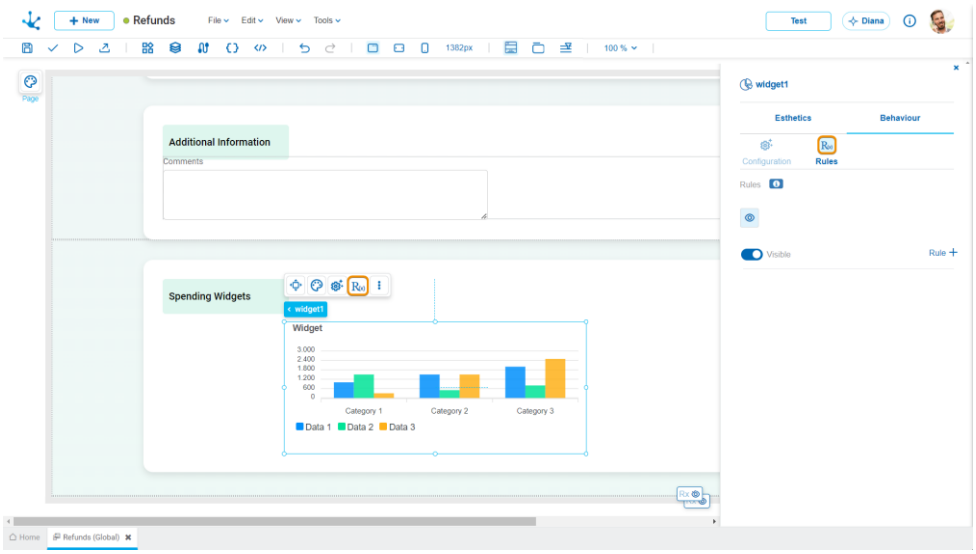
Direction
Position of the message in relation to the element (top, bottom, left, or right).

Chart Type
The types of charts for the modeled widgets are displayed in the environment.

Widget
The widgets modeled in the environment are displayed, according to the type of chart defined in the previous property.

Rule Properties

The rule properties panel of an element opens when clicking the icon $R_{(x)}$ of its context menu.



Properties

Rules

Embedded rules on behavior, validation and calculation can be defined, associated with an element, by using the wizard (ctrl + space).

Shows syntax examples for writing the rules.

Visible

Indicates whether the element is visible. If this property is not checked, the element is not displayed in the page.

Visible (default) Not visible

Rule + Opens an edit area where a rule to determine the visibility condition can be defined. If a rule is defined, the icon is displayed with light blue borders.

Operations when defining the rule:

Saves the new or modified rule

Cancels the operation

Operations once the rule is defined:

Edits the existing rule

Deletes the rule

Slider

It allows the sequential display of various elements, such as images, videos, or any other content in a horizontal sequence. Users can slide horizontally through the set of items, either manually using navigation controls or automatically. This enhances the user experience, offering a more dynamic and personalized interaction with the content.

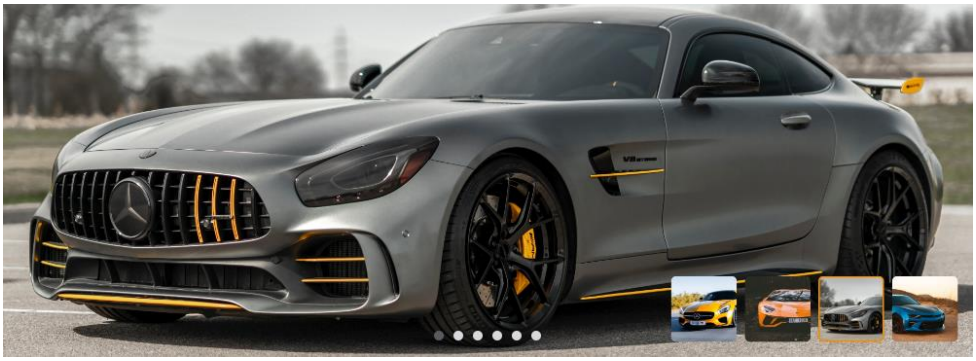
Subtypes

When selecting the option "Slider" from the icon in the top toolbar, a list opens with the different subtypes of this element, which can be dragged and dropped into the modeling area. Each subtype has the element's properties modeled and built in a specific way.

- Dots
- Arrows

Dots

This type of slider uses browsing dots, usually located at the bottom. Each dot represents an element of the set, allowing the user to display and directly select by pressing the corresponding dot.



Arrows

This type of slider incorporates browsing arrows located on both sides of it. The arrows allow the user to move forward or backward in the sequence, moving between items one at a time.



The element properties are represented by icons on its context menu, where its operations are also available.

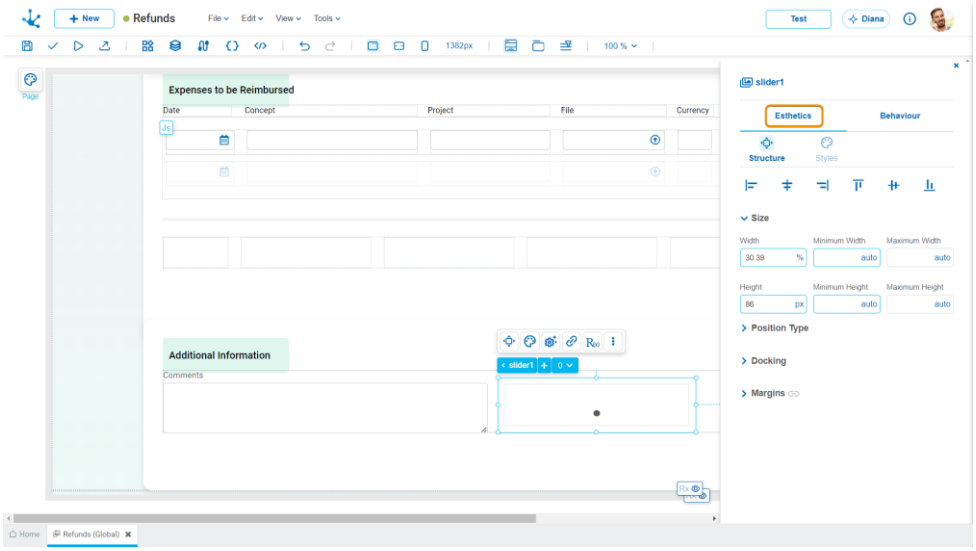
Classification of Properties

Properties are divided into two groups: esthetic properties and behavior properties.

Esthetic Properties

In the esthetic properties panel, the following are grouped:

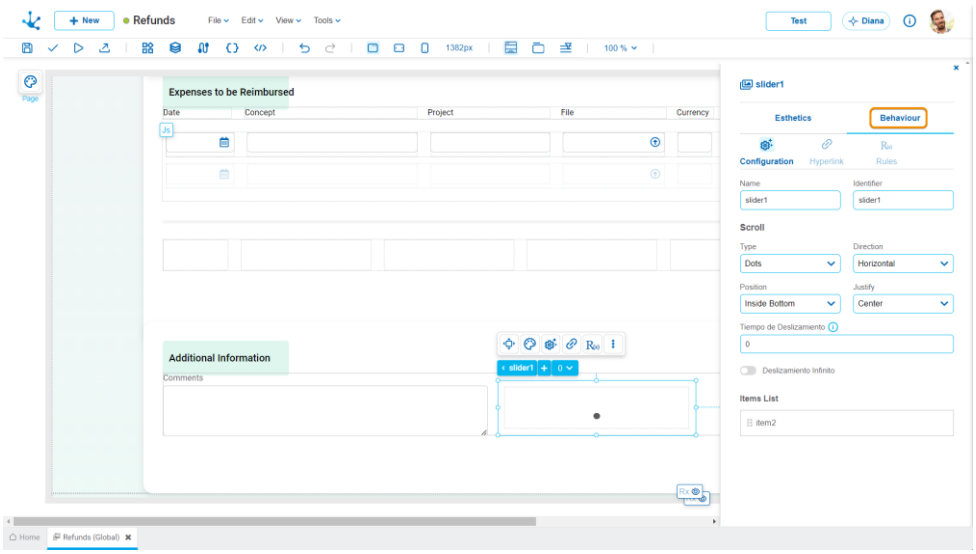
- Structure Properties
- Style Properties



Behavior Properties

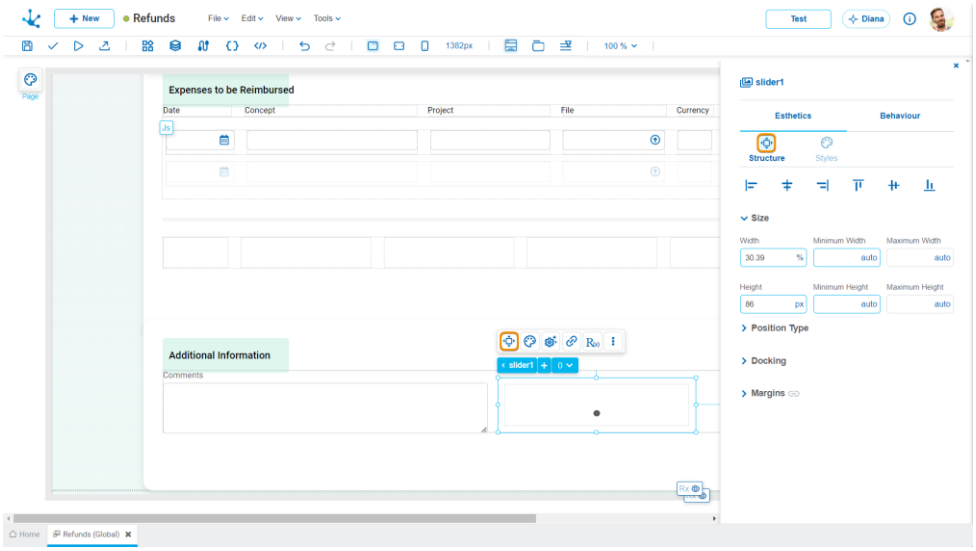
In the behavior properties panel, the following are grouped:

- Configuration Properties
- Hyperlink Properties
- Rule Properties



Structure Properties

The structure properties panel of an element opens when selecting the icon  of its context menu.



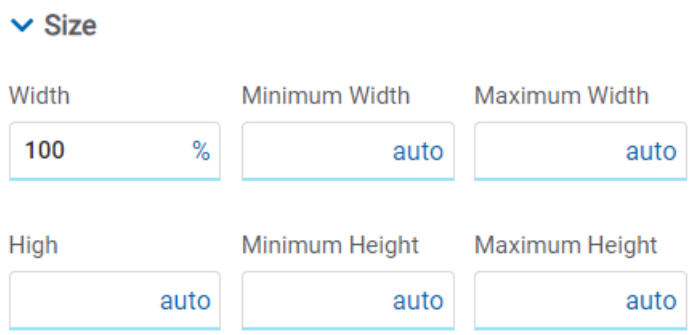
Alignment

Allows to align the element by selecting one of the available icons.

-  Align to the left.
-  Align to horizontal center.
-  Align to the right.
-  Align up.

- Align vertical center.
- Align bottom.

Size

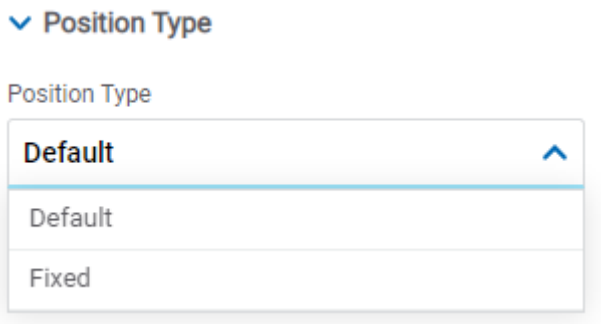


All size properties can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh). Additionally, for Width and Height properties, the "auto" option is added.

> *If selecting percentage (%) for the width and height properties, the size is calculated relative to the top element.*

Position Type

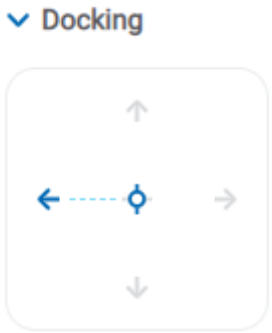
It determines if at the time of execution the element remains fixed on the page or if it moves as the scroll bar moves forward or backward.



Possible Values

- Default: The element has a relative position with respect to the superior element where it was placed (container or section).
- Fixed: Fixes the element to the page so that it is always visible in the same position. It allows to increase the visibility of important content.

Docking



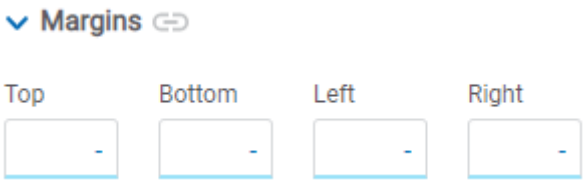
Determines the vertical and horizontal position of elements within the page section or container, when the screen is resized.

When an element is added or moved within the section or container, the modeler automatically docks it to the nearest corners or edges. It can also be configured by clicking on the arrows of the side to be docked to or on the center.

Once the element is docked, its exact position is indicated in the Margins property. The direction of the docking arrows determine the values that are automatically displayed for margins.

The docking position is indicated in the modeling area by dotted lines on the element.

Margins



It allows to define the distance of elements from the borders of their top element. The behavior of margins depends on the docking of the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Top
Distance to the top border of the highest ranking element.

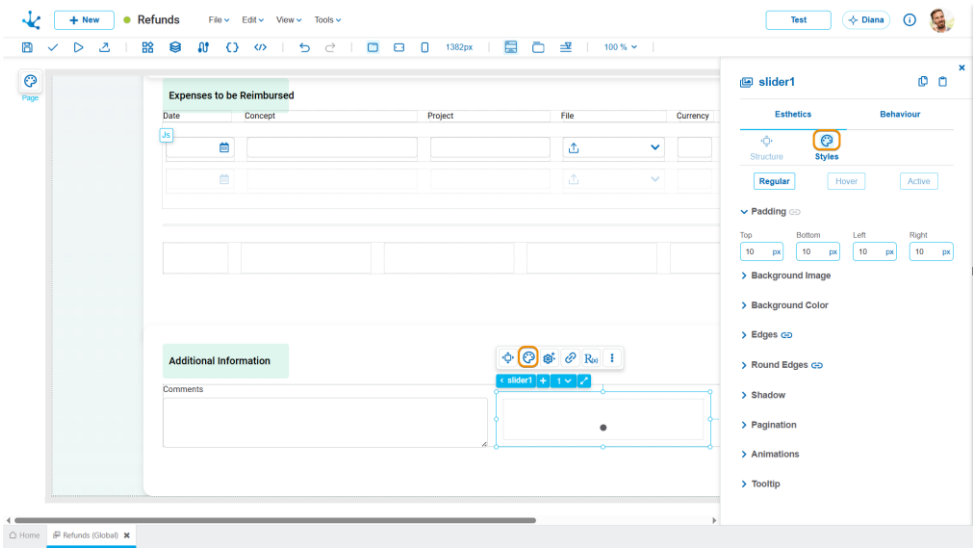
Bottom
Distance to the bottom border of the highest ranking element.

Left
Distance to the left border of the highest ranking element.

Right
Distance to the right border of the highest ranking element.

Allows the value entered in one of the margins to be copied to the other ones automatically.
Allows to indicate different values for each margin.

The style properties panel of an element opens when selecting the icon of its context menu.

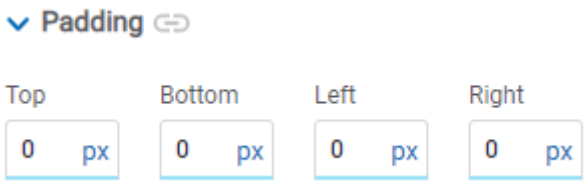


Styles

It allows copying the entire current configuration of the style panel for the selected element (Ctrl+Alt+C).

It allows applying the previously copied configuration to another element of the same type (Ctrl+Alt+V).

Padding

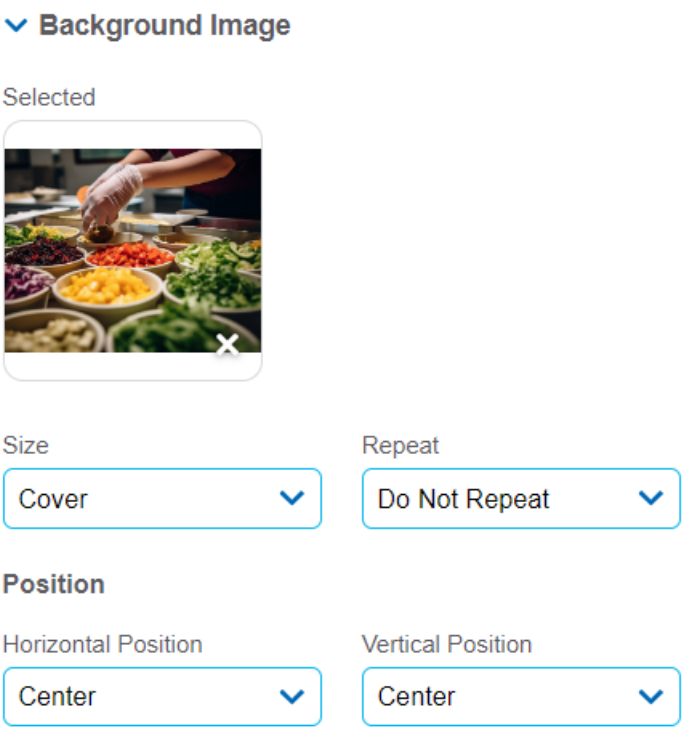


All padding space properties allow to create a space around the element borders (top, bottom, sides) and its content. The padding space configured is inward, while the margin is configured relative to a top element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Allows values entered in one of the paddings to be copied to the other ones automatically.

Allows to indicate different values for each padding.

Background Image



Allows to add a background image to the element.

Selected

An image can be uploaded from the computer where it is being modeled.

Size

Possible Values

- Cover
- Content
- Auto

Repeat

Possible Values

- Repeat
- Repeat Hor.
- Repeat Vert.
- Spacing
- Rounded
- Do Not Repeat

Horizontal Position

Possible Values

- Center

- Left
- Right

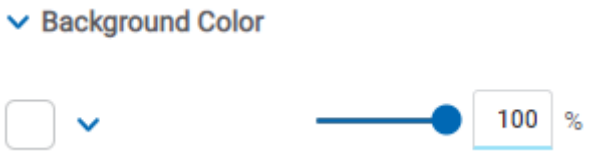
Vertical Position

- Center
- Top
- Botton

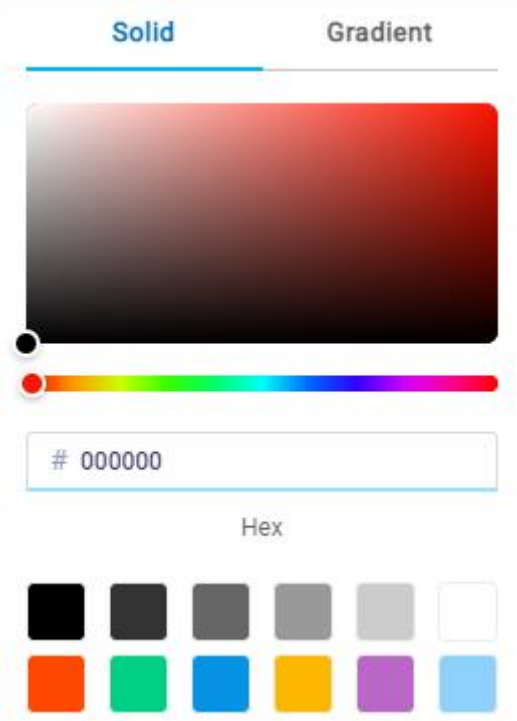
Background Color

It allows to model a background color for the element, it can be solid or gradient and different properties are defined for each one.

Solid



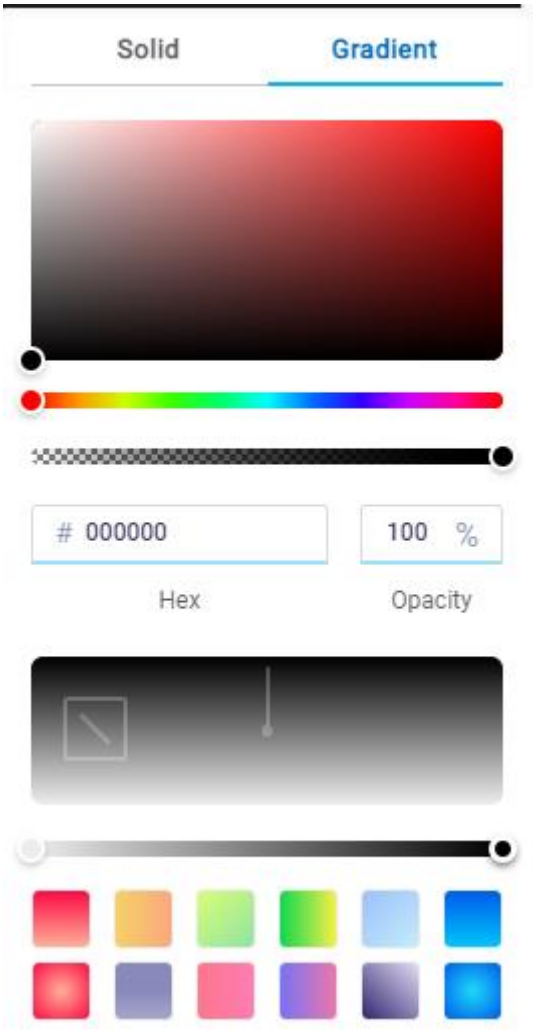
This option enables not only to select the color from a palette but also to define the degree of transparency.



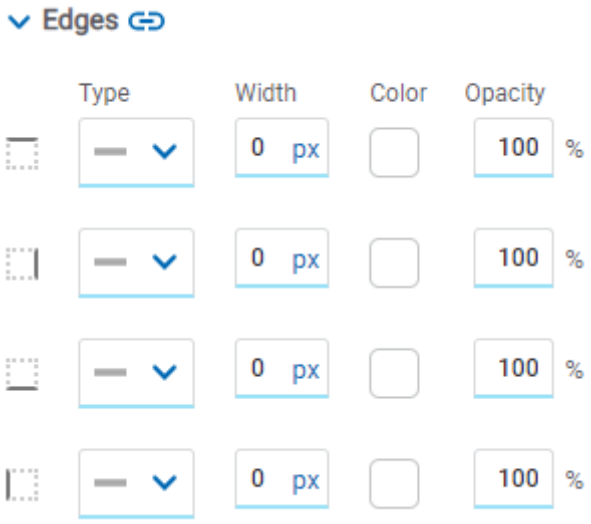
Gradient



This option enables to select not only the color from a palette but also to define its opacity and intensity according to the selected angle.



Edges



It allows to define the style of borders. Each one has its type, width, color and opacity defined, the latter as a percentage.

Allows the value entered in one of the borders to be copied to the other ones automatically.
Allows to indicate different values for each border.

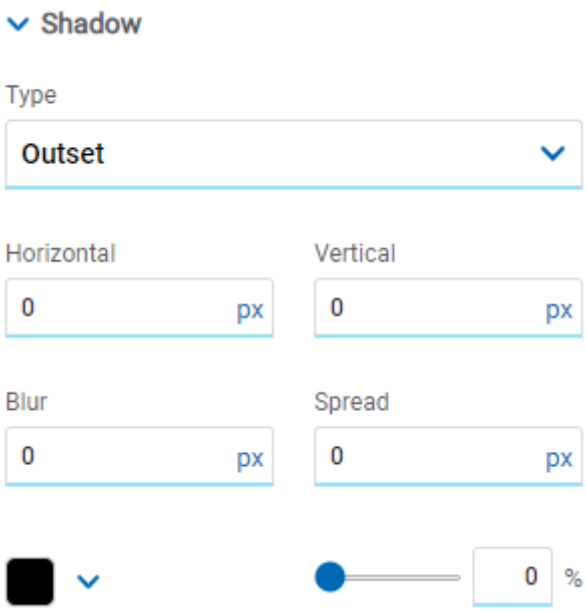
Round Edges



It allows to define the round edges at the corners of elements.
It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Allows the value entered in one of the borders to be copied to the other ones automatically.
Allows to indicate different values for each border.

Shadow



Allows to define a shadow effect around the element.

Horizontal
Horizontal size of the shadow to the right of the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Vertical
Vertical size of the shadow below the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

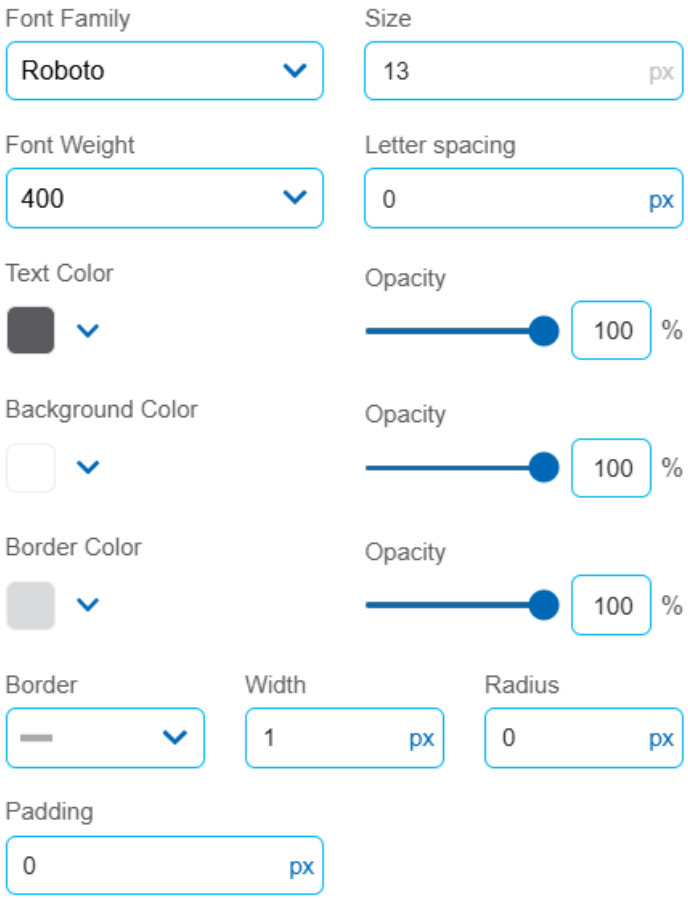
Blur
The greater this value is, the greater and lighter the shadow becomes. If not specified, its value is 0 and the shadow border is darker. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Spread
Positive values will cause the shadow to expand and grow bigger, while negative values will cause the shadow to shrink. If not specified, its value is 0 and the shadow will have the same size as the element.

Color
Defines the color of the shadow.

Transparency
Defines the transparency of the shadow.

Tooltip



Source

It allows customizing the font, size, thickness and letter spacing of the pop-up message.

Colors

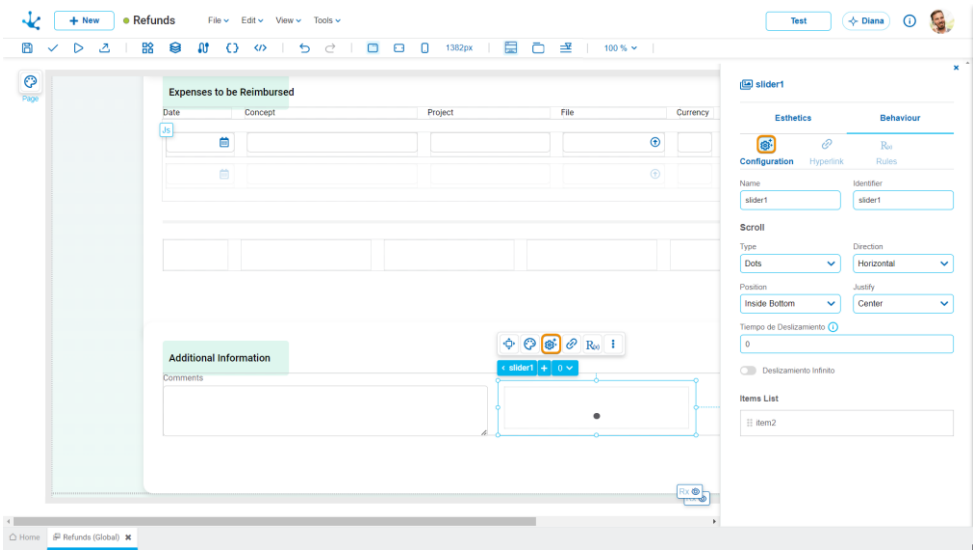
The color and opacity of the text, background and border can be defined. Borders are set uniformly for all sides of the tooltip.

Border

The border type, width, and border radius can be configured.

Configuration Properties

The configuration properties panel of an element opens when clicking the icon  of its context menu.



Name
Name used to reference the element during modeling.

Identifier
Uniquely identifies the element. It is used in the Javascript SDK.

Tooltip
Pop-up message that appears when hovering over an element, providing additional information.



Text
Content of the message being displayed.

Direction
Position of the message in relation to the element (top, bottom, left, or right).

Scroll

Type
It allows selection between types of sliding, either by dots or by arrows.

Direction
It allows for the sliding direction, horizontal or vertical, to be defined, depending on the needs of the content.

Position
It allows defining the position of the slider controls, either inside or outside the slider, placing them at the top or bottom.

Justify

The slider controls can be aligned to the left, center, or right of it, according to the desired layout.

Arrows at the End of the Scroll

It allows modeling the sliding controls when reaching the end of the sequence. This option is only applicable to arrow-type sliders.

Possible Values

- Disable
- Hide

Sliding Time

It allows adjusting the speed of the automatic sliding, expressed in seconds, to enhance the user experience and adapt to the type of content.

Infinite Sliding

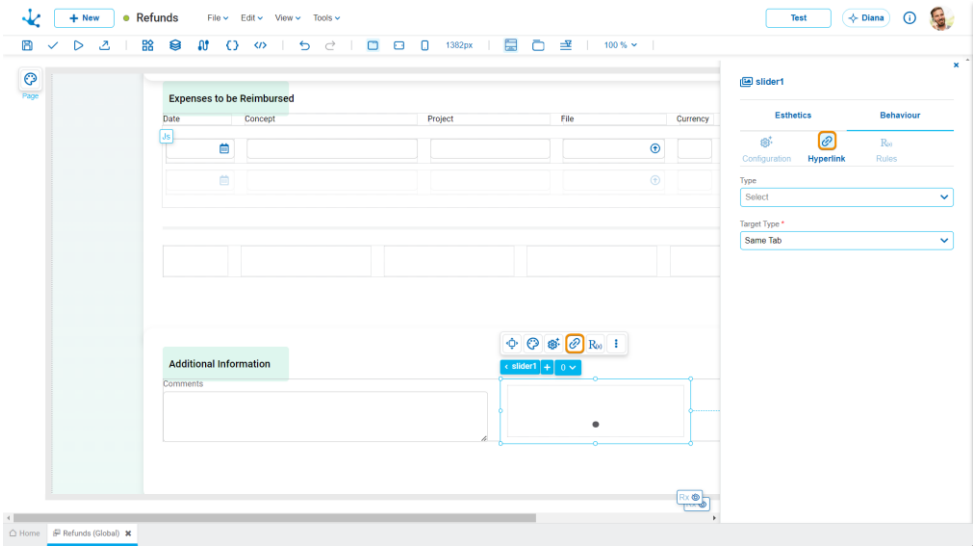
It allows enabling or disabling the option to restart the sequence once the end is reached.

Items List

It allows changing the order of items in the sequence.

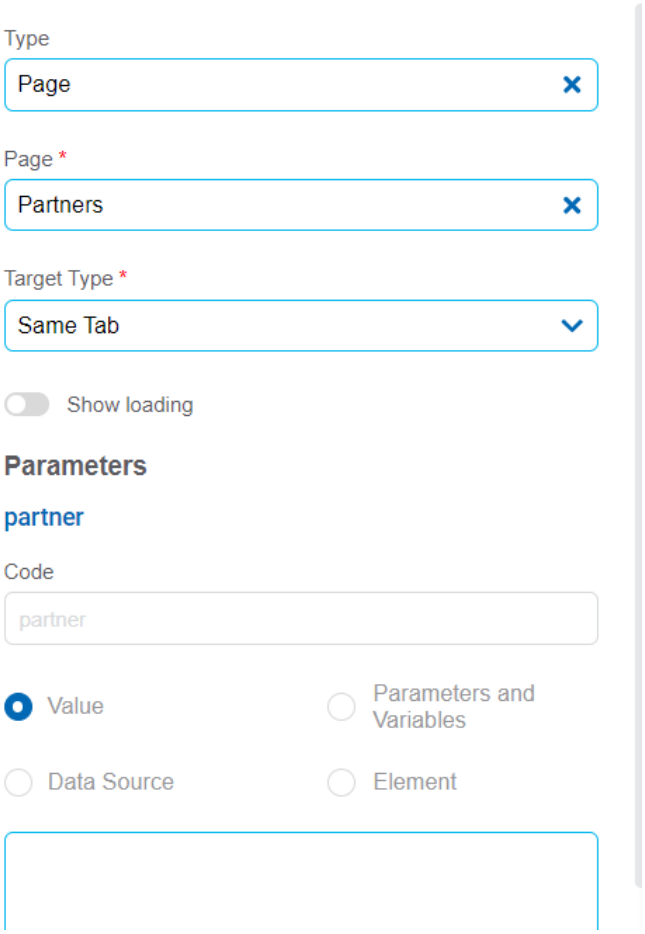
Hyperlink Properties

The hyperlink properties panel of an element opens when clicking the icon  of its context menu.



Define the behavior of the element when selecting it. Different properties are enabled depending on the type of object selected.

Page



Page
The pages modeled in the environment are displayed.

Destination Type
The available options for opening the object are displayed.

Possible Values

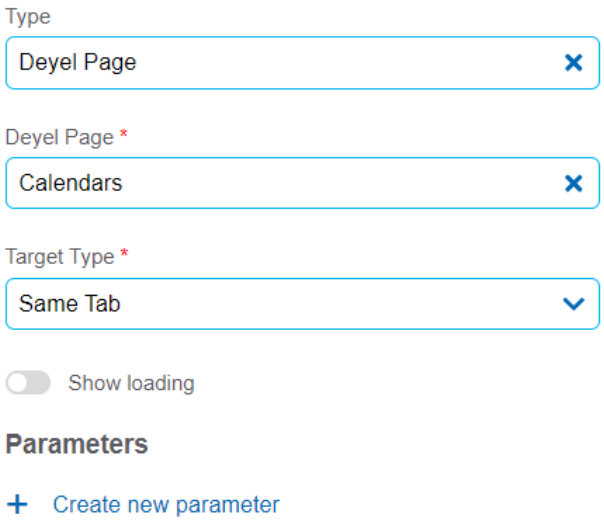
- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  Modal Horizontal Size
  Defines its width.
  Modal Vertical Size
  Defines its height.
- Iframe: selecting this option enables an additional property.
  Destination Iframe
  Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading

Shows the loading icon and disables the user interaction until the redirection is over.

Parameters
If there are parameters, their value can be specified with text, parameters and variables or data source.

Deyel Page



Deyel Page
The pages belonging to **Deyel** are displayed.
Destination Type
The available options for opening the object are displayed.

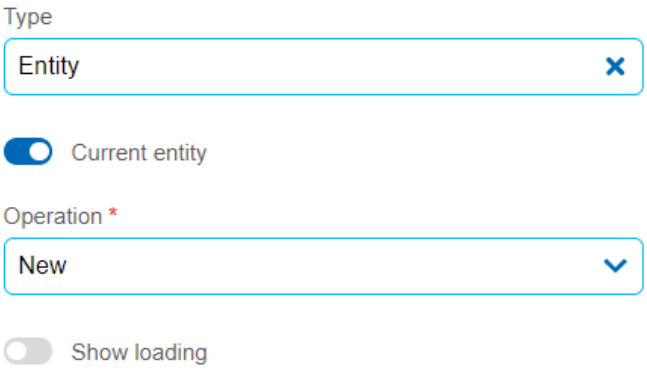
Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  Modal Horizontal Size
  Defines its width.
  Modal Vertical Size
  Defines its height.
- Iframe: selecting this option enables an additional property.
  Destination Iframe
  Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Parameters
Allows sending parameters to the selected object type.

Entity



Current Entity

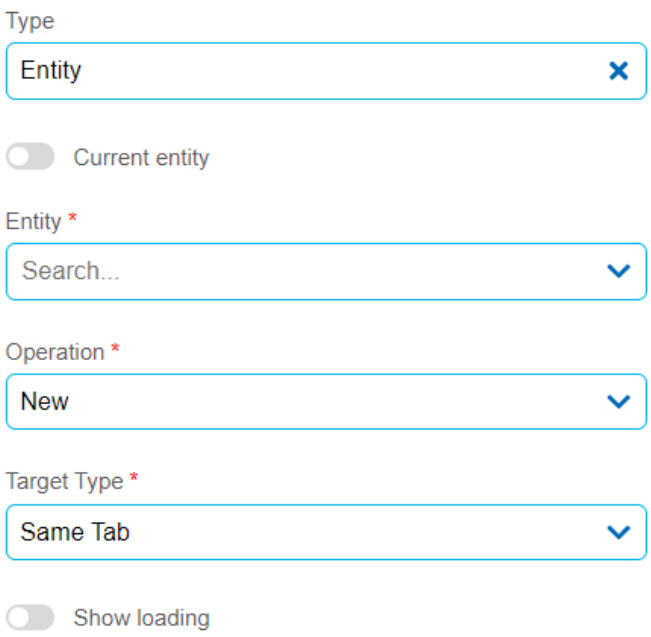
If this property is checked, the operation applies directly to the entity being modeled.

Operation

Defines the operation made when selecting the object.

Possible Values

- Create: Indicates that the panel of the current entity is opened to create an instance.
- Update: Indicates that the panel of the current entity is opened to update an instance.
- Delete: Indicates that the panel of the current entity is opened to delete an instance.
- Send Email: It is visible if the property Send Emails is enabled. Clicking on the modeled element opens a panel during instance execution to send an email. The panel size is adjusted with the properties Modal Width and Modal Height.
- Emails: Visible if the Send Emails property is enabled. Clicking on the modeled element opens a panel during instance execution to manage sent emails. The panel is displayed in the iframe defined in the Target Iframe property.



Entity
If the property Current Entity, is not checked, the entities modeled in the environment are displayed.

Operation
Defines the operation made when selecting the object.

Possible Values

- New: The panel of the entity selected in the previous property is opened to create an instance.
- Update The panel of the entity selected in the previous property is opened to update an instance.
- Show: The panel of the entity selected in the previous property is opened to show an instance.
- Grid: Indicates that the grid of the entity selected in the previous property is opened.

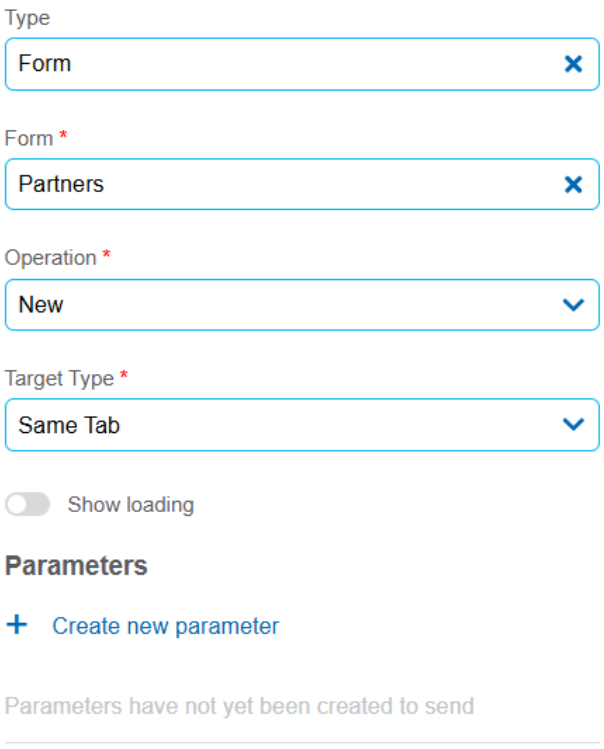
Destination Type
The available options for opening the object are displayed.

Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  Modal Horizontal Size
  Defines its width.
  Modal Vertical Size
  Defines its height.
- Iframe: selecting this option enables an additional property.
  Destination Iframe
  Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Forms



Forms
The forms modeled in the environment are displayed.
Operation
Defines the operation made when selecting the object.

Possible Values

- Create: Indicates that the panel of the object selected in the previous property is opened for the creation of an instance.
- Update: Indicates that the panel of the object selected in the previous property is opened for the update of an instance.
- Show: Indicates that the panel of the object selected in the previous property is opened for the query of an instance.
- Grid: Indicates that the grid of the object selected in the previous property is opened.

Destination Type
The available options for opening the object are displayed.

Possible Values

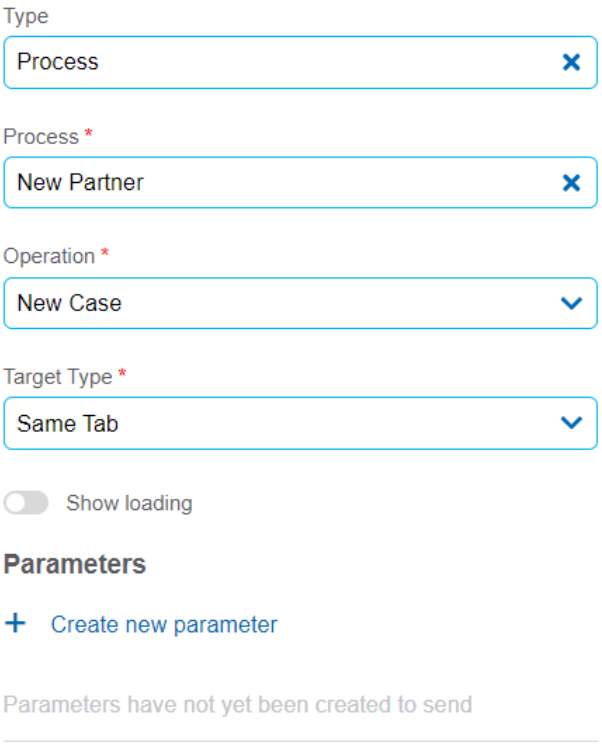
- Same Tab
- New Window

- Modal: selecting this option enables additional properties.
  Modal Horizontal Size
  Defines its width.
  Modal Vertical Size
  Defines its height.
- Iframe: selecting this option enables an additional property.
  Destination Iframe
  Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Parameters
Allows sending parameters to the selected object type.

Process



Process
The processes modeled in the environment are displayed.

Operation
Defines the operation made when selecting the object.

Destination Type

The available options for opening the object are displayed.

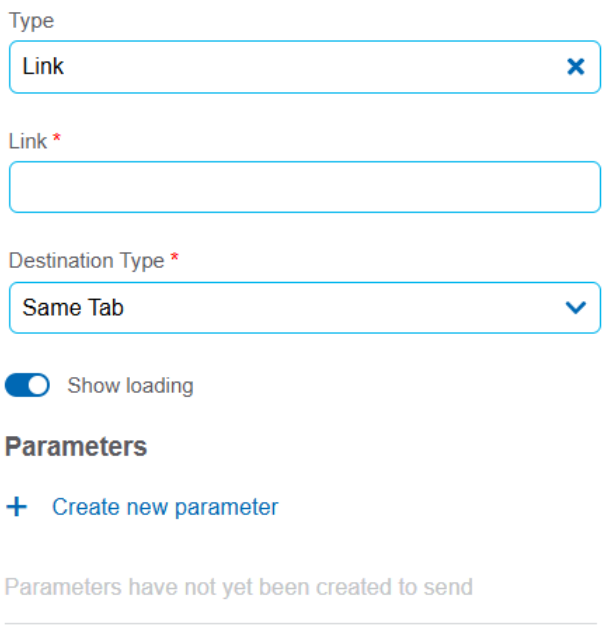
Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  Modal Horizontal Size
  Defines its width.
  Modal Vertical Size
  Defines its height.
- Iframe: selecting this option enables an additional property.
  Destination Iframe
  Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Parameters
Allows sending parameters to the selected object type.

Link



Link
Allows to enter any link.

Destination Type
The available options for opening the object are displayed.

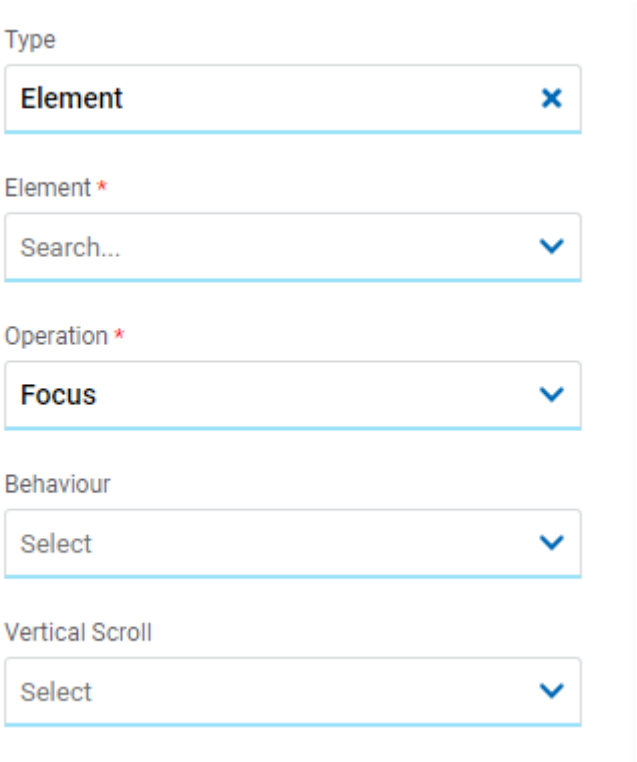
Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  Modal Horizontal Size
  Defines its width.
  Modal Vertical Size
  Defines its height.
- Iframe: selecting this option enables an additional property.
  Destination Iframe
  Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Parameters
Allows sending parameters to the selected object type.

Element



Element
The modeled elements in the object are displayed.

Operation

Defines the operation made when selecting the element.

Possible Values

- Focus
- Show
- Hide
- Show/Hide

Behaviour
Establishes the transition of the animation. It is only modelable if the "Focus" operation was selected.

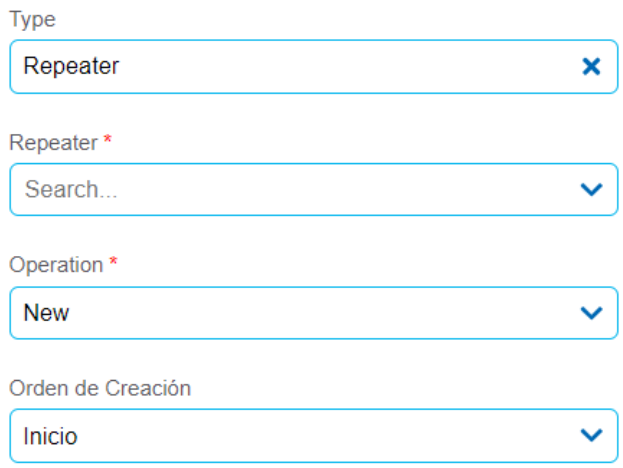
Possible Values

- Auto
- Smooth

Vertical Scroll
It is only modelable if the "Focus" operation was selected.
Possible Values

- Start: Moves to the start of the selected element.
- Closest: Moves to the closest position to the selected element from the element the event fires.
- Center: Moves to the center of the selected element.
- End: Moves to the end of the selected element.

Repeater



Repeater
The modeled repeaters from the object are displayed.

Operation
Defines the operation made when selecting the element.

Possible Values

- New: Allows for the creation of an item within the repeater. If this option is selected the additional properties are enabled.
  Creation Order
  It indicates in which position the new item of the container is added.
  Possible Values
  - Home
  - End
- Delete: Allows for the deletion of an item within the repeater.

Case



Case
Show case information.

Operation
Defines the operation made when selecting the element.

Possible Values

- Show Detail
- Show Graphic Execution

Destination Type
The available options for opening the object are displayed.

Possible Values

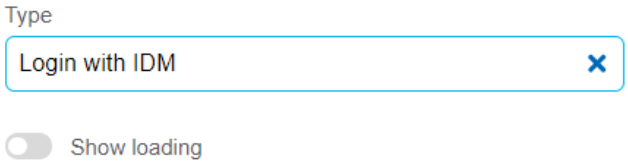
- Expanded Panel
- Modal: if this option is selected the additional properties are enabled.
  Modal Horizontal Size
  Define its width.
  Modal Vertical Size
  Define its height.

Back



It allows associating the event to go back in the browser to the element.

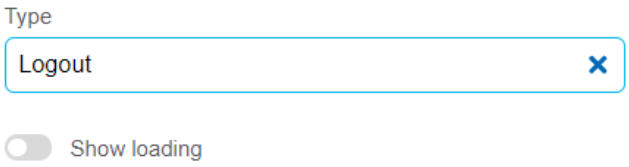
LogIn with IDM



Allows login with IDM.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

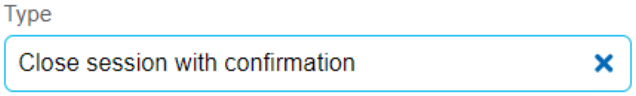
Logout



Allows logging the user out.

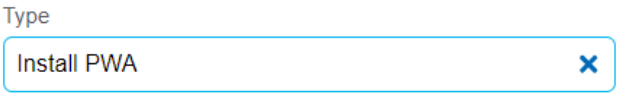
Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Close Session with Confirmation



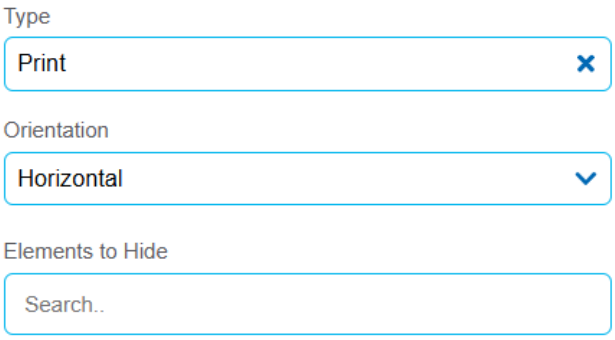
Displays a confirmation modal to logout the user.

Install PWA



Allows to install the application in the browser.

Print



This type of hyperlink associated with an element allows printing the content of the entity or page that is being executed.

Orientation

Determines the layout of the content to be printed.

Possible Values

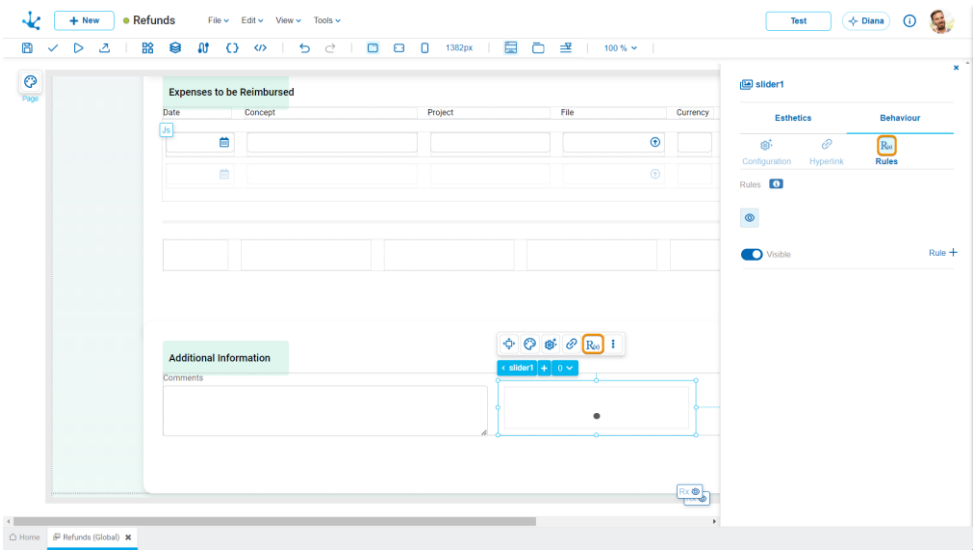
- Horizontal
- Vertical

Elements to hide

It allows selecting which elements of the entity or page should not be included in the printout.

Rule Properties

The rule properties panel of an element opens when clicking the icon  of its context menu.



Properties

Rules

Embedded rules on behavior, validation and calculation can be defined, associated with an element, by using the wizard (ctrl + space).

Shows syntax examples for writing the rules.

Visible

Indicates whether the element is visible. If this property is not checked, the element is not displayed in the page.

Visible (default) Not visible

Rule + Opens an edit area where a rule to determine the visibility condition can be defined. If a rule is defined, the icon is displayed with light blue borders.

Operations when defining the rule:

Saves the new or modified rule

Cancels the operation

Operations once the rule is defined:

Edits the existing rule

Deletes the rule

Item

Items are defined within the slider and only the first one can be updated since the changes are applied to the rest.

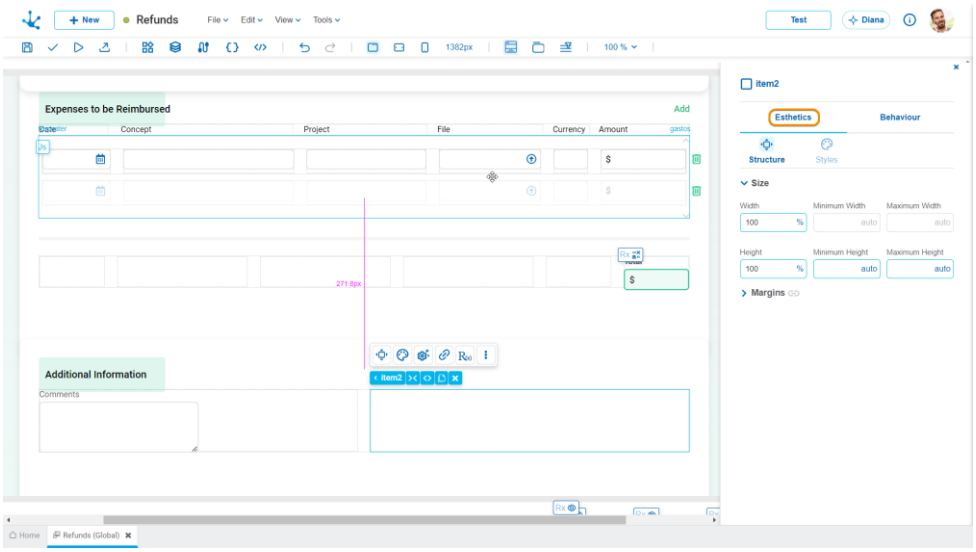
Classification of Properties

Properties are divided into two groups: esthetic properties and behavior properties.

Esthetic Properties

In the esthetic properties panel, the following are grouped:

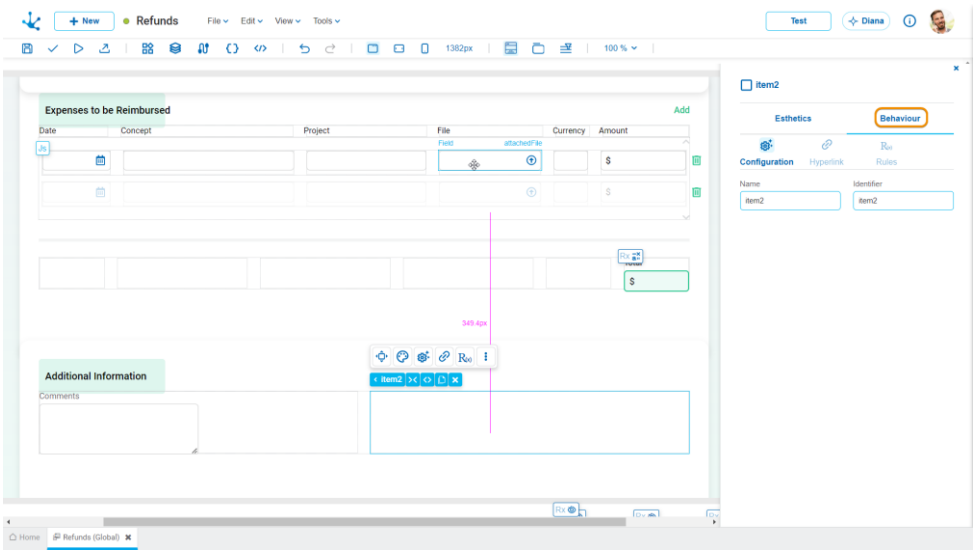
- Style Properties



Behavior Properties

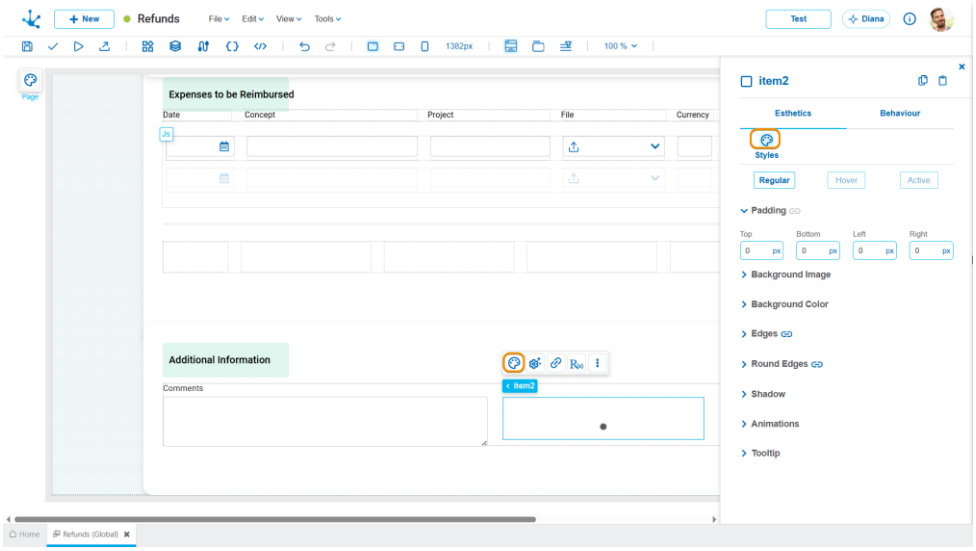
In the behavior properties panel, the following are grouped:

- Configuration Properties
- Hyperlink Properties
- Rule Properties



Style Properties

This type of element may take different states and for each of them different values for its properties may be modeled.



This type of element may take different states and for each of them different values for its properties may be modeled.



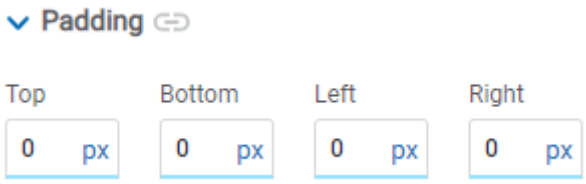
- Regular: The mouse pointer is not over the element.
- Hover: The mouse pointer is over the element.

Styles

It allows copying the entire current configuration of the style panel for the selected element (Ctrl+Alt+C).

It allows applying the previously copied configuration to another element of the same type (Ctrl+Alt+V).

Padding

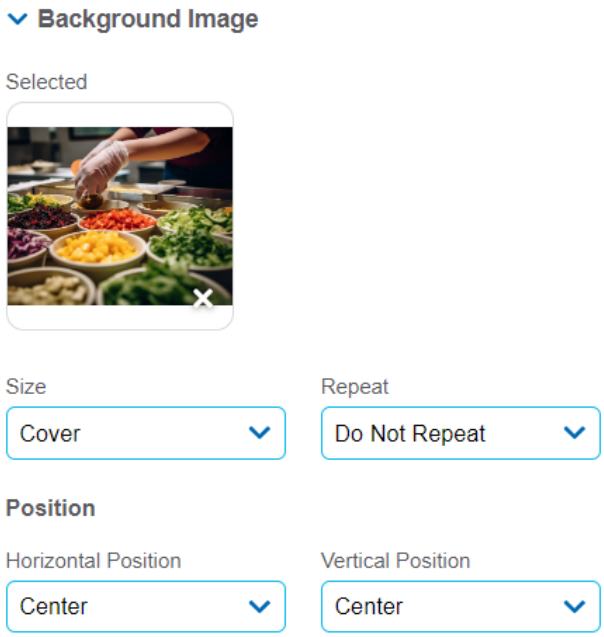


All padding space properties allow to create a space around the item borders (top, bottom, sides) and its content. The padding space configured is inward, while the margin is configured relative to a top item. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Allows values entered in one of the paddings to be copied to the other ones automatically.
Allows to indicate different values for each padding.

Background Image

It allows the addition of a background image to the item.

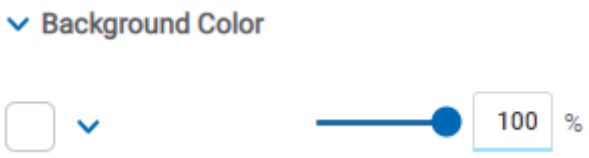


Selected
An image can be uploaded from the computer where it is being modeled.

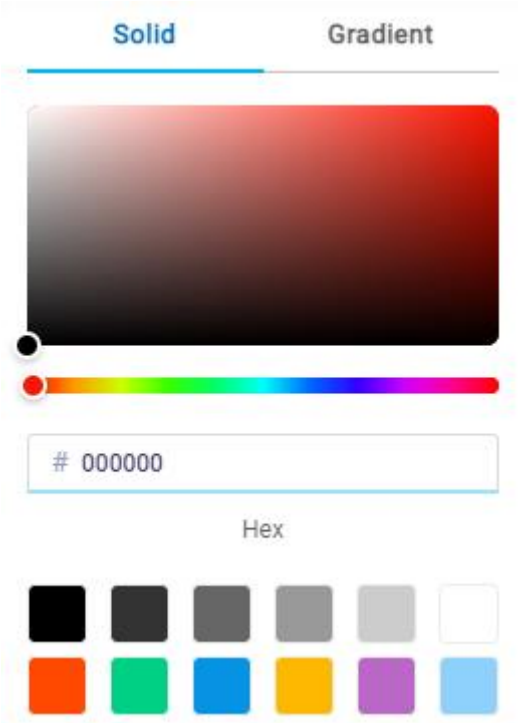
Background Color

It allows to model a background color for the element, it can be solid or gradient and different properties are defined for each one.

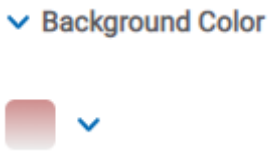
Solid



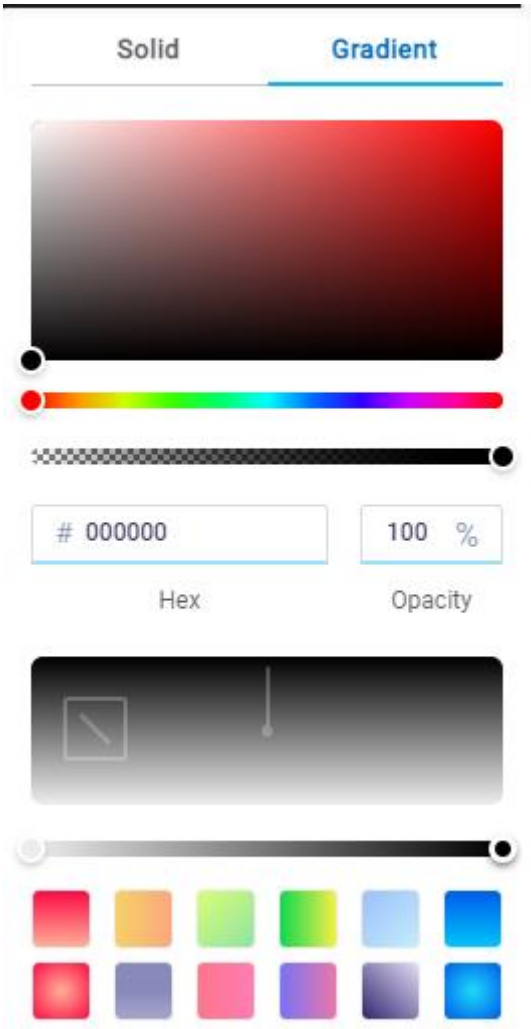
This option enables not only to select the color from a palette but also to define the degree of transparency.



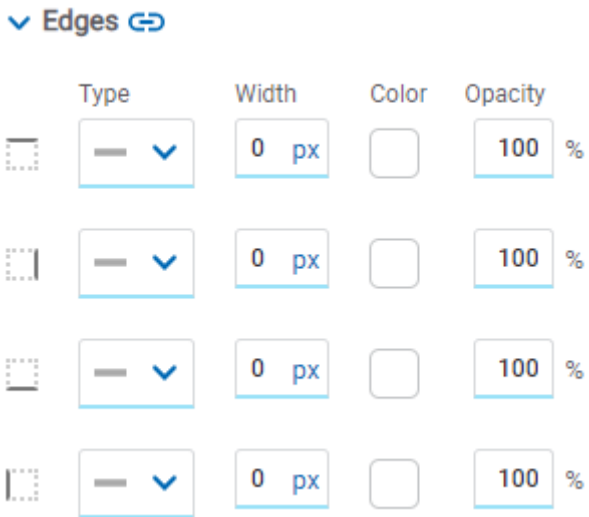
Gradient



This option enables to select not only the color from a palette but also to define its opacity and intensity according to the selected angle.


Solid
Gradient
000000
100 %
Hex
Opacity

Edges


Edges
Type
Width
Color
Opacity
0 px
100 %
0 px
100 %
0 px
100 %
0 px
100 %

It allows to define the style of borders. Each one has its type, width, color and opacity defined, the latter as a percentage.

Allows the value entered in one of the borders to be copied to the other ones automatically.
Allows to indicate different values for each border.

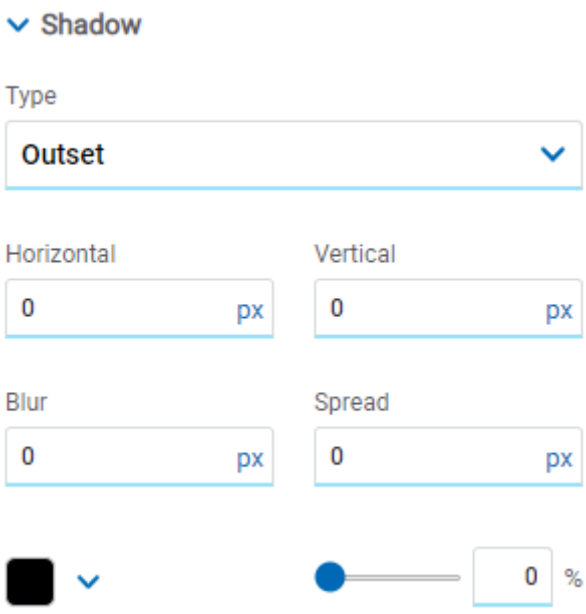
Round Edges



It allows to define the round edges at the corners of items.
It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Allows the value entered in one of the borders to be copied to the other ones automatically.
Allows to indicate different values for each border.

Shadow



Allows to define a shadow effect around the item.

Type

Possible Values

- Outset

- Inset

Horizontal
Horizontal size of the shadow to the right of the item. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Vertical
Vertical size of the shadow below the item. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

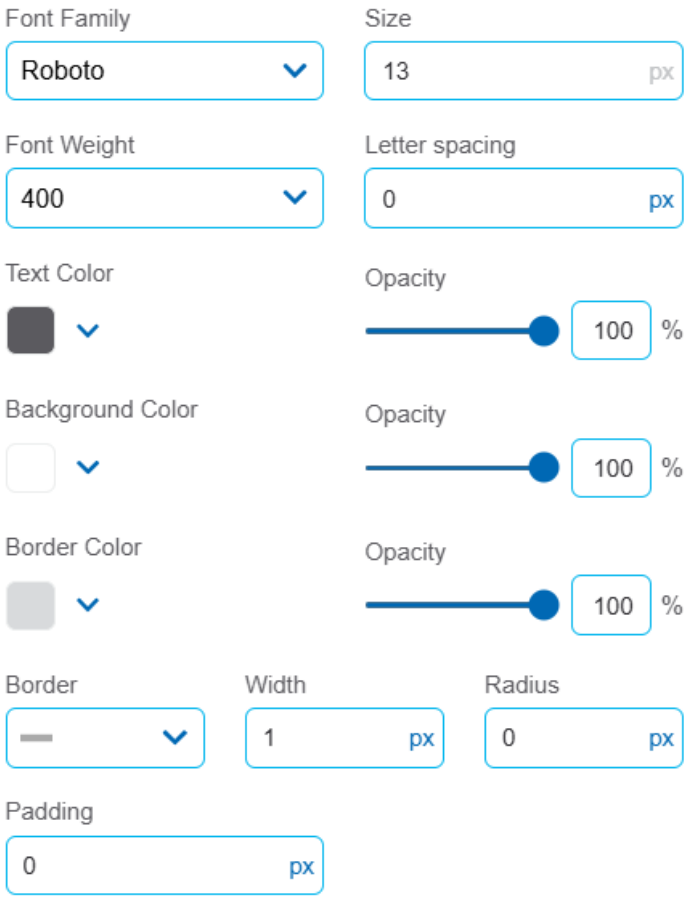
Blur
The greater this value is, the greater and lighter the shadow becomes. If not specified, its value is 0 and the shadow border is darker. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Spread
Positive values will cause the shadow to expand and grow bigger, while negative values will cause the shadow to shrink. If not specified, its value is 0 and the shadow will have the same size as the item.

Color
Defines the color of the shadow.

Transparency
Defines the transparency of the shadow.

Tooltip



Source

It allows customizing the font, size, thickness and letter spacing of the pop-up message.

Colors

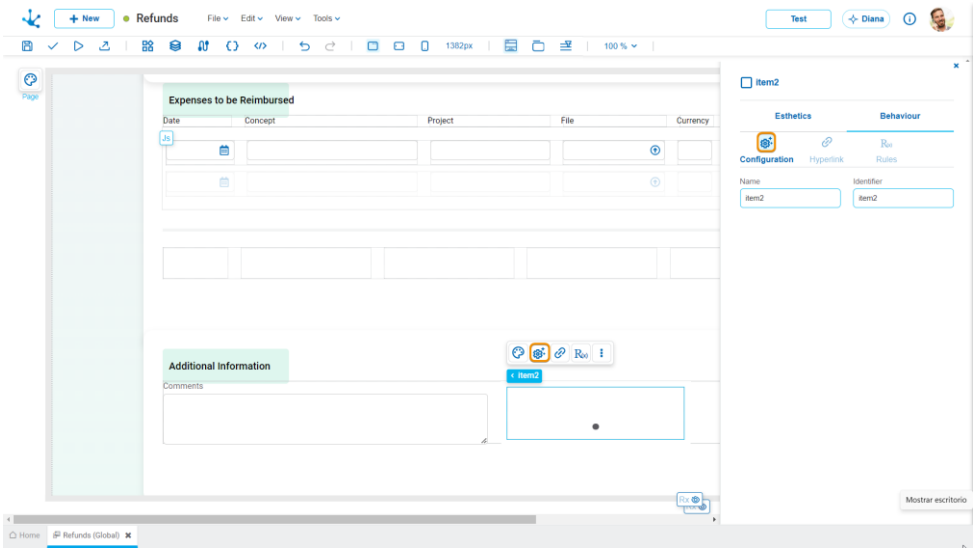
The color and opacity of the text, background and border can be defined. Borders are set uniformly for all sides of the tooltip.

Border

The border type, width, and border radius can be configured.

Configuration Properties

The configuration properties panel of an element opens when clicking the icon  of its context menu.



Name
Name used to reference the element during modeling.

Identifier
Uniquely identifies the element. It is used in the Javascript SDK.

Tooltip
Pop-up message that appears when hovering over an element, providing additional information.

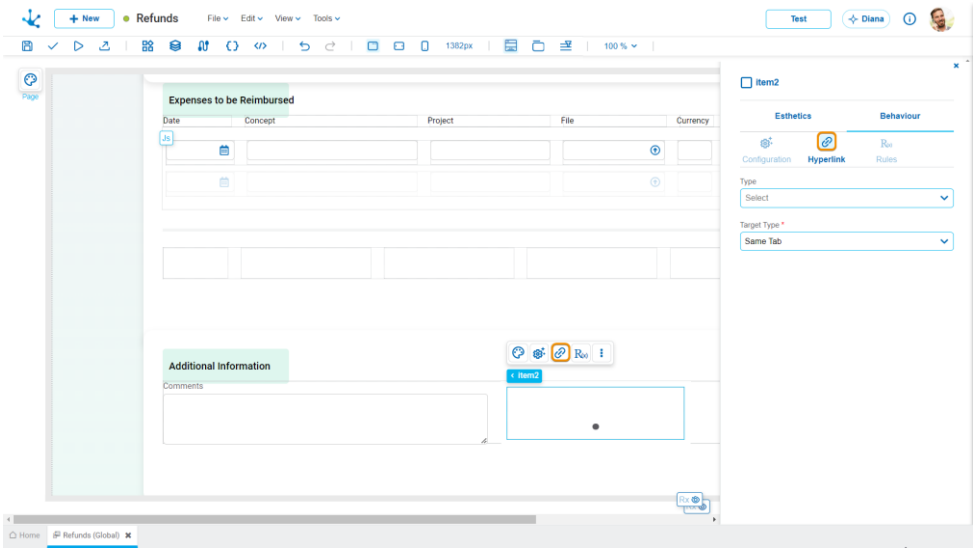


Text
Content of the message being displayed.

Direction
Position of the message in relation to the element (top, bottom, left, or right).

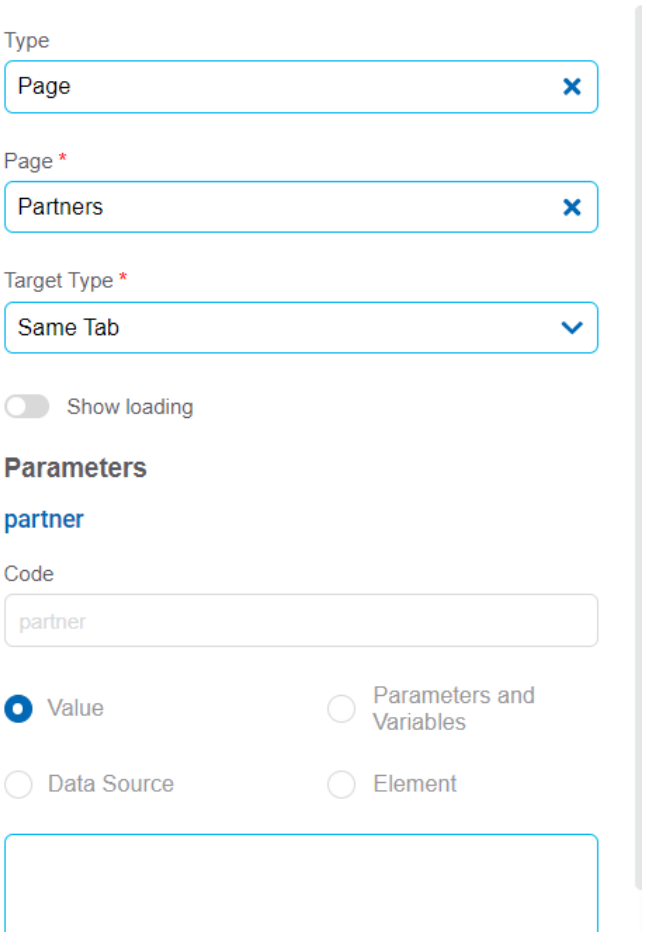
Hyperlink Properties

The hyperlink properties panel of an element opens when clicking the icon  of its context menu.



Define the behavior of the element when selecting it. Different properties are enabled depending on the type of object selected.

Page



Page
The pages modeled in the environment are displayed.

Destination Type
The available options for opening the object are displayed.

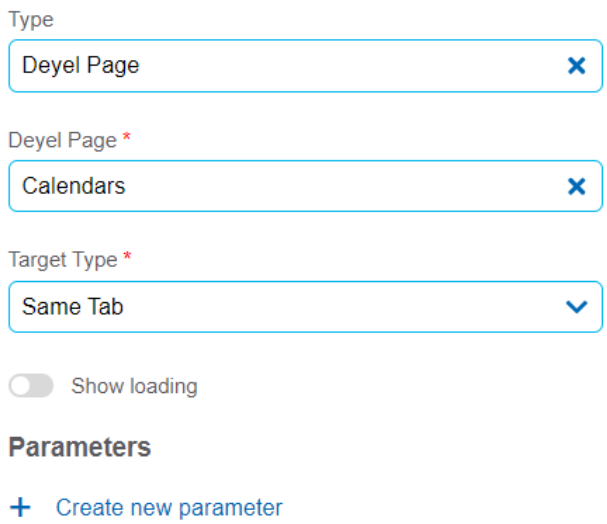
Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  - Modal Horizontal Size
    Defines its width.
  - Modal Vertical Size
    Defines its height.
- Iframe: selecting this option enables an additional property.
  - Destination Iframe
    Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Parameters
If there are parameters, their value can be specified with text, parameters and variables or data source.

Deyel Page



Deyel Page
The pages belonging to **Deyel** are displayed.

Destination Type
The available options for opening the object are displayed.

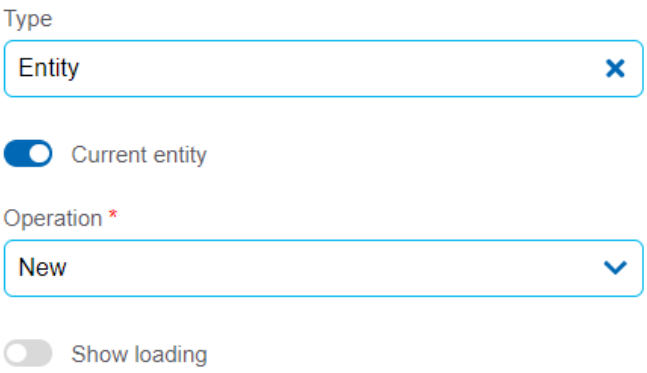
Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  - Modal Horizontal Size
    Defines its width.
  - Modal Vertical Size
    Defines its height.
- Iframe: selecting this option enables an additional property.
  - Destination Iframe
    Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Parameters
Allows sending parameters to the selected object type.

Entity



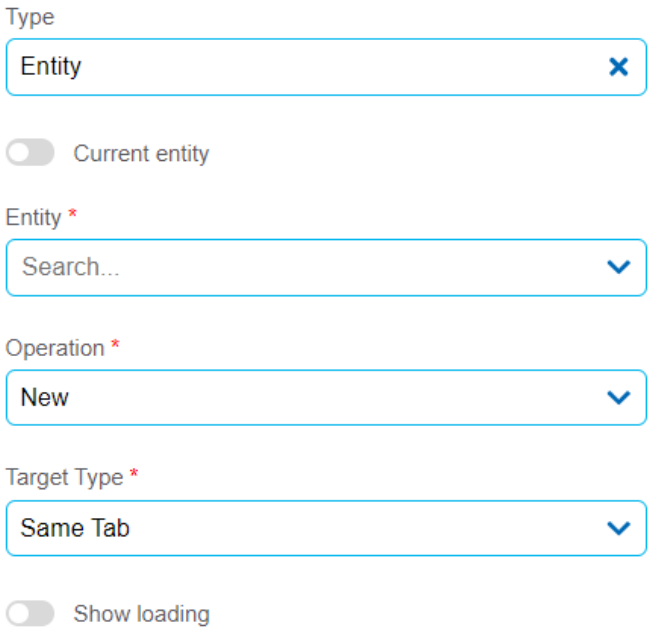
Current Entity
If this property is checked, the operation applies directly to the entity being modeled.

Operation
Defines the operation made when selecting the object.

Possible Values

- Create: Indicates that the panel of the current entity is opened to create an instance.
- Update: Indicates that the panel of the current entity is opened to update an instance.
- Delete: Indicates that the panel of the current entity is opened to delete an instance.

- Send Email: It is visible if the property Send Emails is enabled. Clicking on the modeled element opens a panel during instance execution to send an email. The panel size is adjusted with the properties Modal Width and Modal Height.
- Emails: Visible if the Send Emails property is enabled. Clicking on the modeled element opens a panel during instance execution to manage sent emails. The panel is displayed in the iframe defined in the Target Iframe property.



Entity
If the property Current Entity, is not checked, the entities modeled in the environment are displayed.

Operation
Defines the operation made when selecting the object.

Possible Values

- New: The panel of the entity selected in the previous property is opened to create an instance.
- Update The panel of the entity selected in the previous property is opened to update an instance.
- Show: The panel of the entity selected in the previous property is opened to show an instance.
- Grid: Indicates that the grid of the entity selected in the previous property is opened.

Destination Type
The available options for opening the object are displayed.

Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  Modal Horizontal Size
  Defines its width.

Modal Vertical Size
Defines its height.

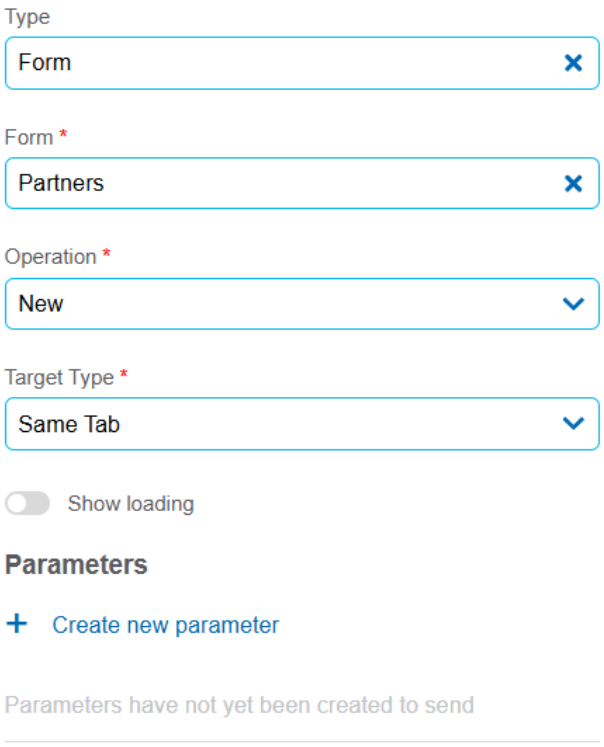
- Iframe: selecting this option enables an additional property.

Destination Iframe
Expands the iframes previously defined on the object.

- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Forms



Forms
The forms modeled in the environment are displayed.

Operation
Defines the operation made when selecting the object.

Possible Values

- Create: Indicates that the panel of the object selected in the previous property is opened for the creation of an instance.
- Update: Indicates that the panel of the object selected in the previous property is opened for the update of an instance.
- Show: Indicates that the panel of the object selected in the previous property is opened for the query of an instance.

- Grid: Indicates that the grid of the object selected in the previous property is opened.

Destination Type
The available options for opening the object are displayed.

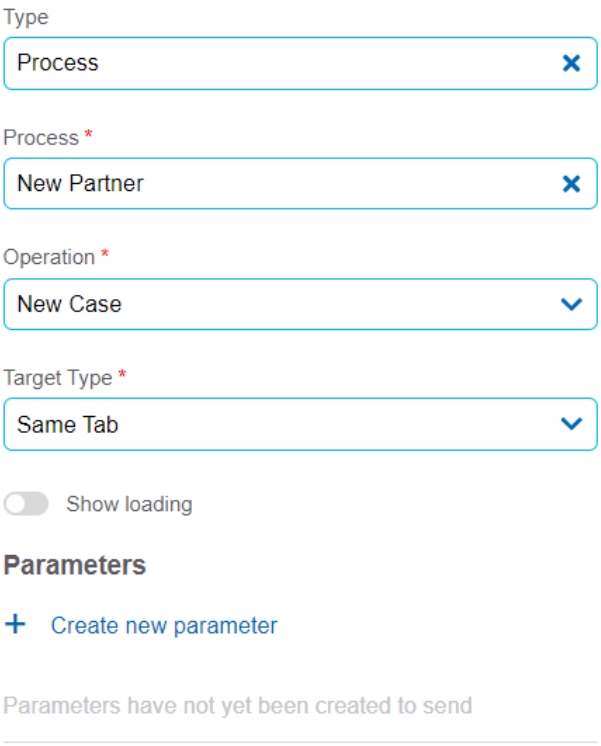
Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  Modal Horizontal Size
  Defines its width.
  Modal Vertical Size
  Defines its height.
- Iframe: selecting this option enables an additional property.
  Destination Iframe
  Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Parameters
Allows sending parameters to the selected object type.

Process



Process
The processes modeled in the environment are displayed.

Operation
Defines the operation made when selecting the object.

Destination Type
The available options for opening the object are displayed.

Possible Values

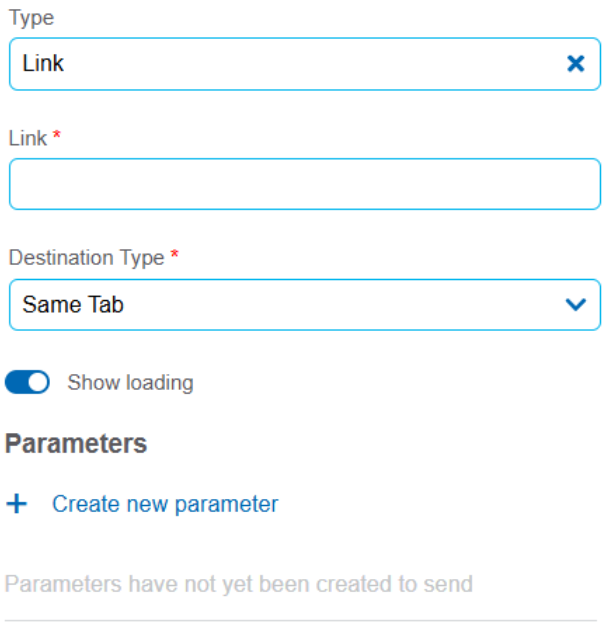
- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  - Modal Horizontal Size
    Defines its width.
  - Modal Vertical Size
    Defines its height.
- Iframe: selecting this option enables an additional property.
  - Destination Iframe
    Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Parameters

Allows sending parameters to the selected object type.

Link



Link
Allows to enter any link.

Destination Type
The available options for opening the object are displayed.

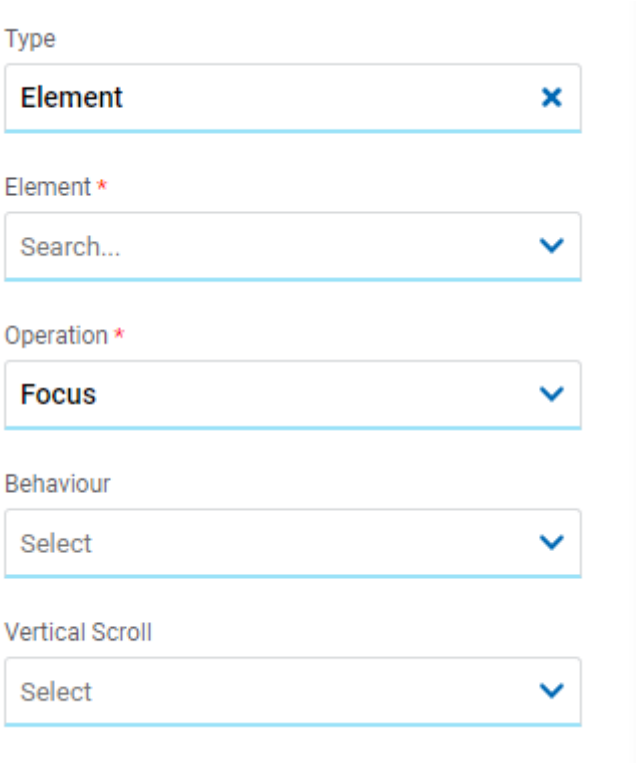
Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  Modal Horizontal Size
  Defines its width.
  Modal Vertical Size
  Defines its height.
- Iframe: selecting this option enables an additional property.
  Destination Iframe
  Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Parameters
Allows sending parameters to the selected object type.

Element



Element
The modeled elements in the object are displayed.

Operation
Defines the operation made when selecting the element.

Possible Values

- Focus
- Show
- Hide
- Show/Hide

Behaviour
Establishes the transition of the animation. It is only modelable if the "Focus" operation was selected.

Possible Values

- Auto
- Smooth

Vertical Scroll
It is only modelable if the "Focus" operation was selected.
Possible Values

- Start: Moves to the start of the selected element.
- Closest: Moves to the closest position to the selected element from the element the event fires.

- Center: Moves to the center of the selected element.
- End: Moves to the end of the selected element.

Repeater



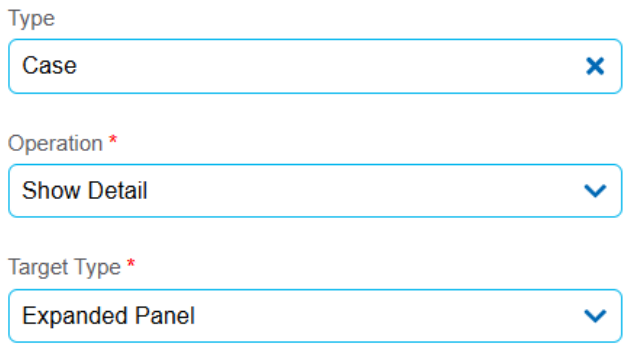
Repeater
The modeled repeaters from the object are displayed.

Operation
Defines the operation made when selecting the element.

Possible Values

- New: Allows for the creation of an item within the repeater. If this option is selected the additional properties are enabled.
  ### Creation Order
  It indicates in which position the new item of the container is added.
  Possible Values
  - Home
  - End
- Delete: Allows for the deletion of an item within the repeater.

Case



Case

Show case information.

Operation
Defines the operation made when selecting the element.

Possible Values

- Show Detail
- Show Graphic Execution

Destination Type
The available options for opening the object are displayed.

Possible Values

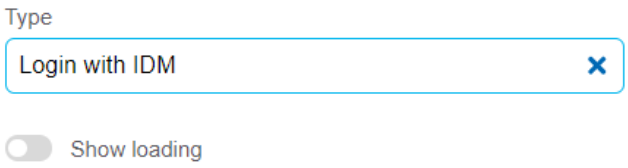
- Expanded Panel
- Modal: if this option is selected the additional properties are enabled.
  Modal Horizontal Size
  Define its width.
  Modal Vertical Size
  Define its height.

Back



It allows associating the event to go back in the browser to the element.

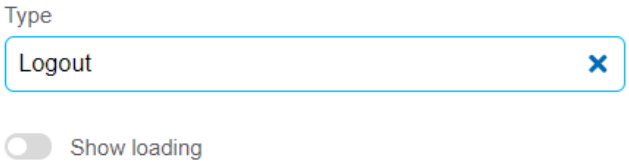
LogIn with IDM



Allows login with IDM.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

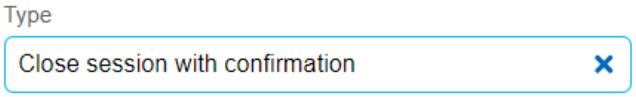
Logout



Allows logging the user out.

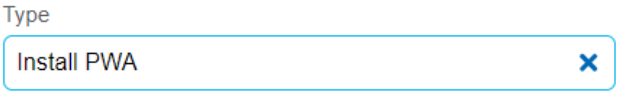
Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Close Session with Confirmation



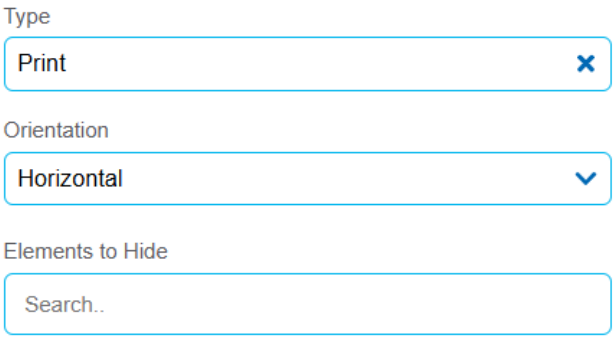
Displays a confirmation modal to logout the user.

Install PWA



Allows to install the application in the browser.

Print



This type of hyperlink associated with an element allows printing the content of the entity or page that is being executed.

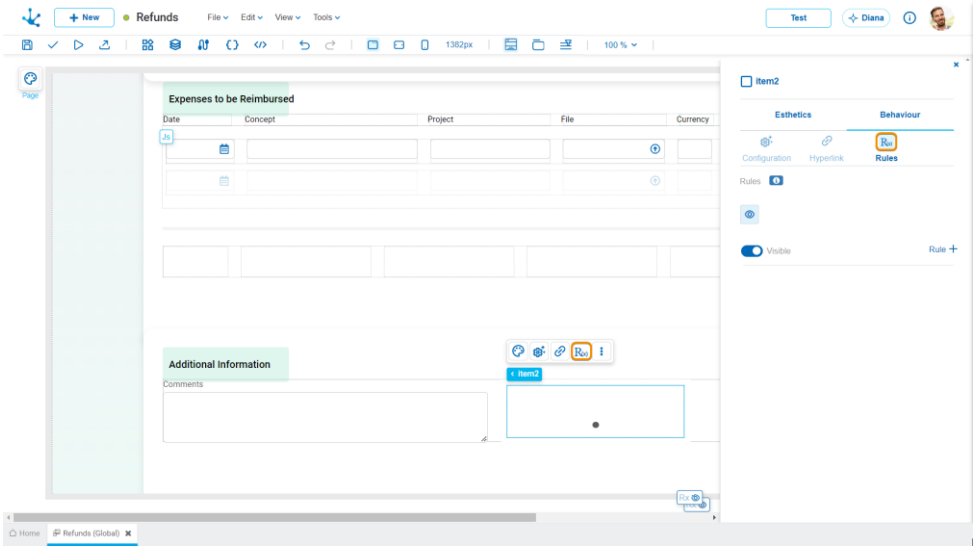
Orientation

Determines the layout of the content to be printed.
Possible Values
- Horizontal
- Vertical

Elements to hide
It allows selecting which elements of the entity or page should not be included in the printout.

Rule Properties

The rule properties panel of an element opens when clicking the icon  of its context menu.



Properties

Rules
Embedded rules on behavior, validation and calculation can be defined, associated with an element, by using the wizard (ctrl + space).
Shows syntax examples for writing the rules.

Visible

Indicates whether the element is visible. If this property is not checked, the element is not displayed in the page.
Visible (default) Not visible
Rule + Opens an edit area where a rule to determine the visibility condition can be defined. If a rule is defined, the icon is displayed with light blue borders.

Operations when defining the rule:

Saves the new or modified rule

Cancels the operation

Operations once the rule is defined:

- Edits the existing rule
- Deletes the rule

Events

Items allow the use of different events.

Event	Description
onMouseIn()	It is executed when the cursor is positioned on the element.
onMouseOut()	It is executed when the cursor moves out of the element.
onClick()	It is executed when clicking on the element.
onDoubleClick()	It is executed when clicking twice on the element.
onInit()	It is executed before the element is loaded.
afterViewInit()	It is executed after viewing the element.
onViewportEnter()	It is executed when the element is visible.
onViewportLeave()	It is executed when the element is no longer visible.

Advanced

It allows modeling and executing the functionality of an entity.

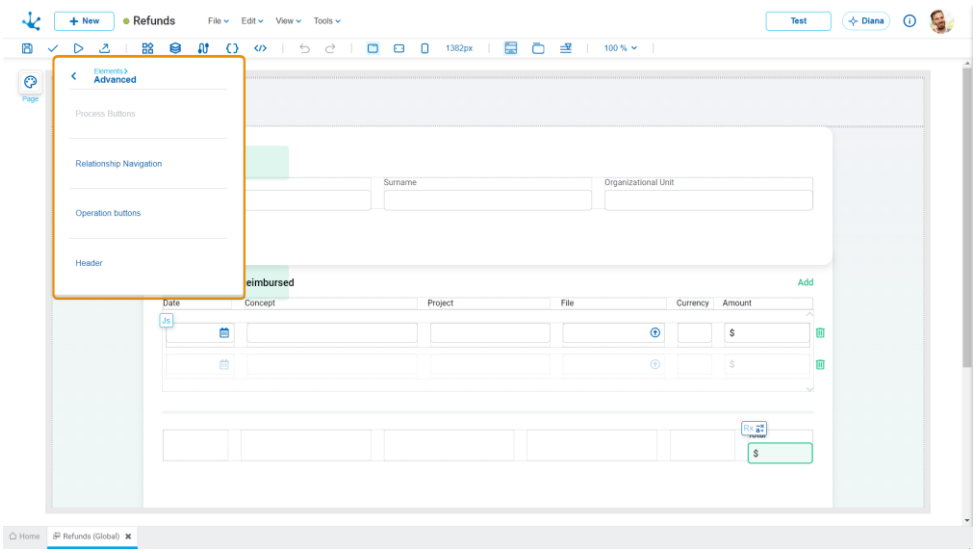
The element properties are represented by icons on its context menu, where its operations are also available.

Subtypes

When selecting the option "Advanced" from the icon in the top toolbar, a list opens with the different subtypes of this element, which can be dragged into the modeling area.

- Process Buttons
- Relation Navigation
- Operation Buttons

- Header



Process Buttons

This advanced element is enabled only if the entity has a related process modeled. When placing the element within the entity modeling area, three buttons are previewed as an example, and it is possible to see, in this element, at execution time, the buttons modeled on the process flows.

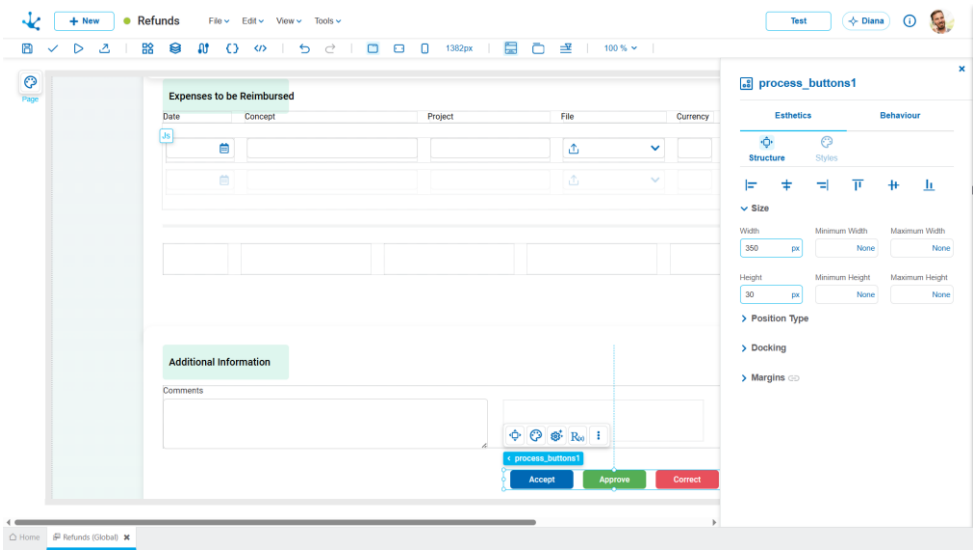
Classification of Properties

Properties are divided into two groups: esthetic properties and behavior properties.

Esthetic Properties

In the esthetic properties panel, the following are grouped:

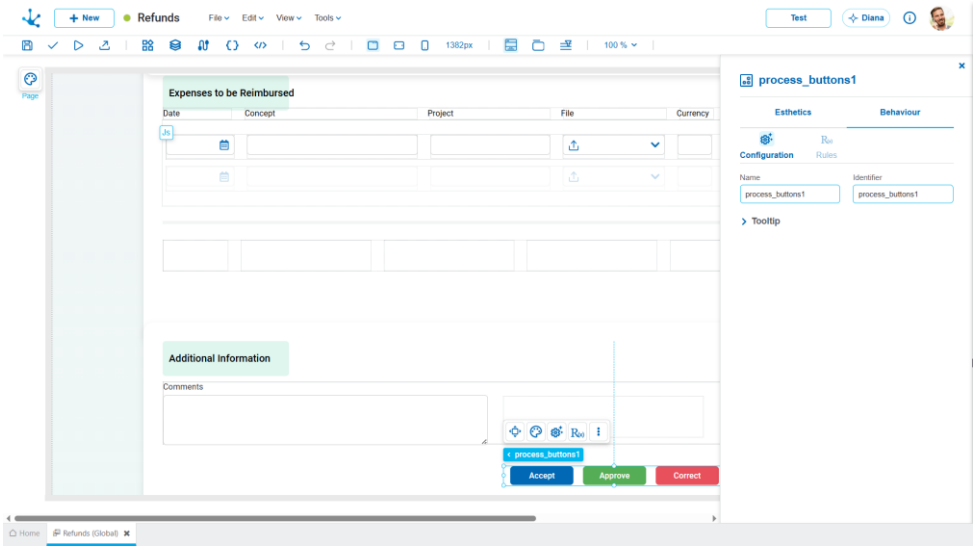
- Structure Properties
- Style Properties



Behavior Properties

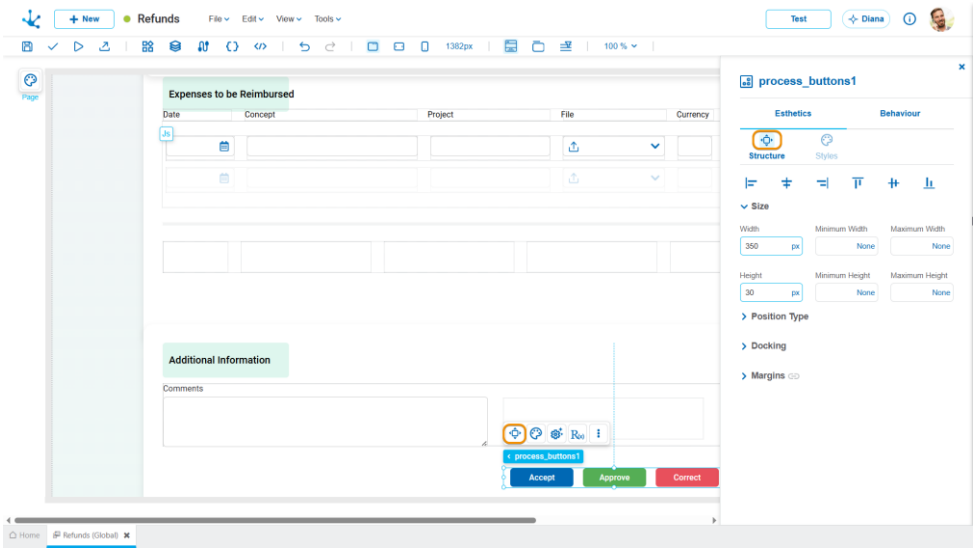
In the behavior properties panel, the following are grouped:

- Configuration Properties
- Rule Properties



Structure Properties

The structure properties panel of an element opens when selecting the icon  of its context menu.

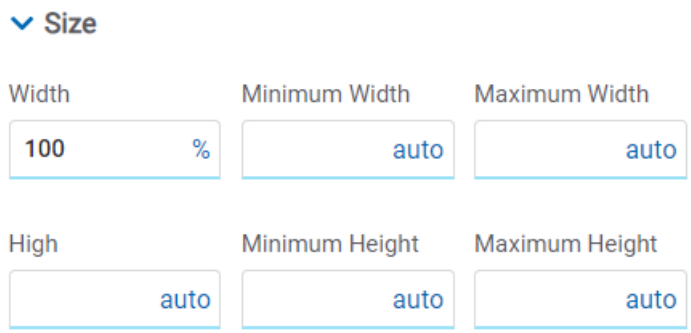


Alignment

Allows to align the element by selecting one of the available icons.

- Align to the left.
- Align to horizontal center.
- Align to the right.
- Align up.
- Align vertical center.
- Align bottom.

Size

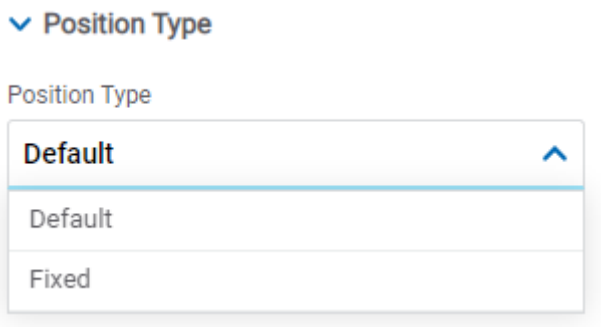


All size properties can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh). Additionally, for Width and Height properties, the "auto" option is added.

> *If selecting percentage (%) for the width and height properties, the size is calculated relative to the top element.*

Position Type

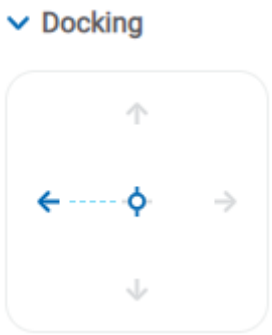
It determines if at the time of execution the element remains fixed on the page or if it moves as the scroll bar moves forward or backward.



Possible Values

- Default: The element has a relative position with respect to the superior element where it was placed (container or section).
- Fixed: Fixes the element to the page so that it is always visible in the same position. It allows to increase the visibility of important content.

Docking



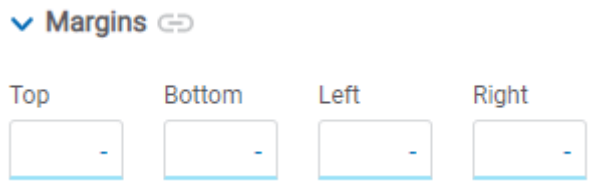
Determines the vertical and horizontal position of elements within the page section or container, when the screen is resized.

When an element is added or moved within the section or container, the modeler automatically docks it to the nearest corners or edges. It can also be configured by clicking on the arrows of the side to be docked to or on the center.

Once the element is docked, its exact position is indicated in the Margins property. The direction of the docking arrows determine the values that are automatically displayed for margins.

The docking position is indicated in the modeling area by dotted lines on the element.

Margins



It allows to define the distance of elements from the borders of their top element. The behavior of margins depends on the docking of the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Top
Distance to the top border of the highest ranking element.

Bottom
Distance to the bottom border of the highest ranking element.

Left
Distance to the left border of the highest ranking element.

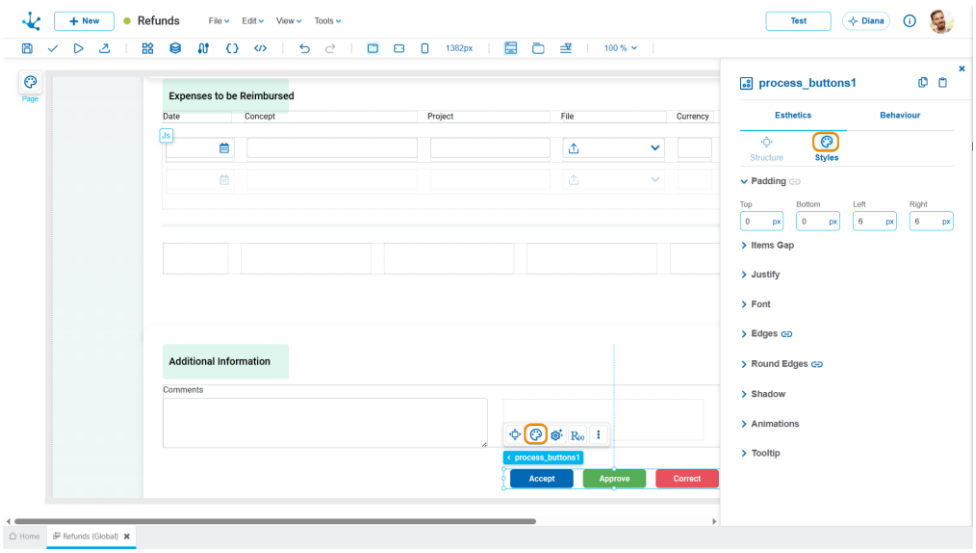
Right
Distance to the right border of the highest ranking element.

Allows the value entered in one of the margins to be copied to the other ones automatically.

Allows to indicate different values for each margin.

Style Properties

The style properties panel of an element opens when selecting the icon of its context menu.



Styles

It allows copying the entire current configuration of the style panel for the selected element (Ctrl+Alt+C).

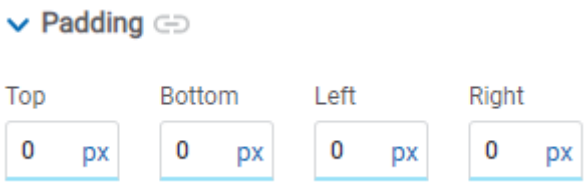
It allows applying the previously copied configuration to another element of the same type (Ctrl+Alt+V).

Align Buttons
Process buttons can be aligned according to the desired layout.
Possible Values

- Center
- Left
- Right
- Line spacing
- Padding
- Even spacing
- Stretching

Item Spacing
Allows defining the space between item-type elements contained within a layout.

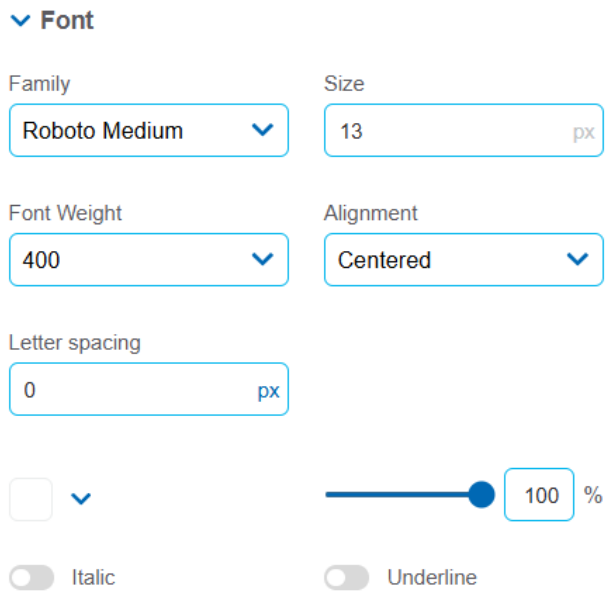
Padding



All padding space properties allow to create a space around the element borders (top, bottom, sides) and its content. The padding space configured is inward, while the margin is configured relative to a top element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

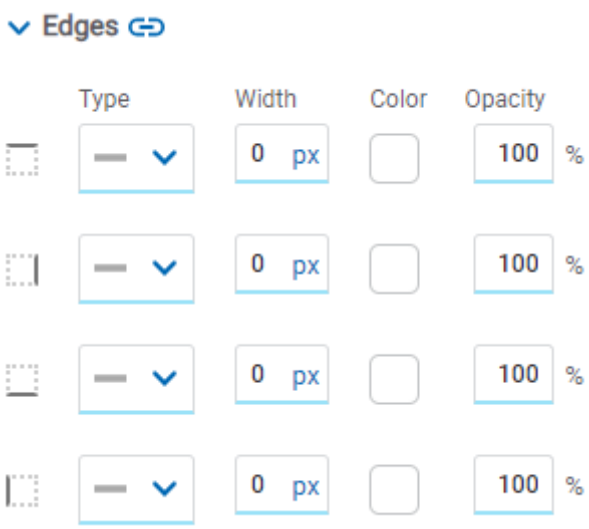
Allows values entered in one of the paddings to be copied to the other ones automatically.
Allows to indicate different values for each padding.

Source



Allows defining the text style.

Edges



It allows to define the style of borders. Each one has its type, width, color and opacity defined, the latter as a percentage.

Allows the value entered in one of the borders to be copied to the other ones automatically.
Allows to indicate different values for each border.

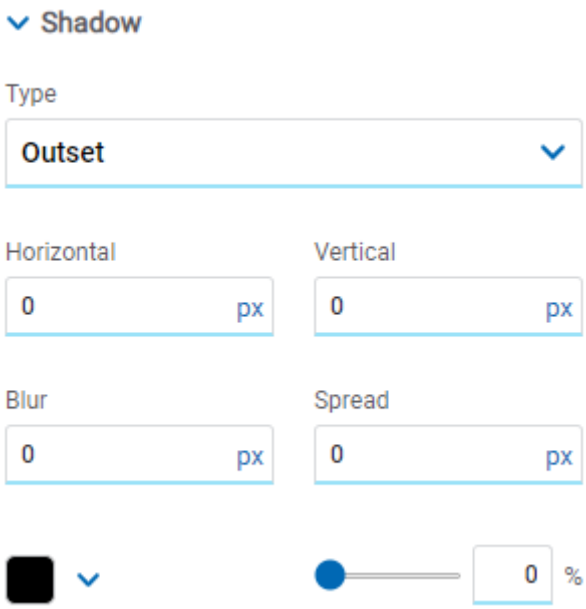
Round Edges



It allows to define the round edges at the corners of elements.
It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Allows the value entered in one of the borders to be copied to the other ones automatically.
Allows to indicate different values for each border.

Shadow



Allows to define a shadow effect around the element.

Type

Possible Values

- Outset
- Inset

Horizontal

Horizontal size of the shadow to the right of the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Vertical

Vertical size of the shadow below the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Blur

The greater this value is, the greater and lighter the shadow becomes. If not specified, its value is 0 and the shadow border is darker. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Spread

Positive values will cause the shadow to expand and grow bigger, while negative values will cause the shadow to shrink. If not specified, its value is 0 and the shadow will have the same size as the element.

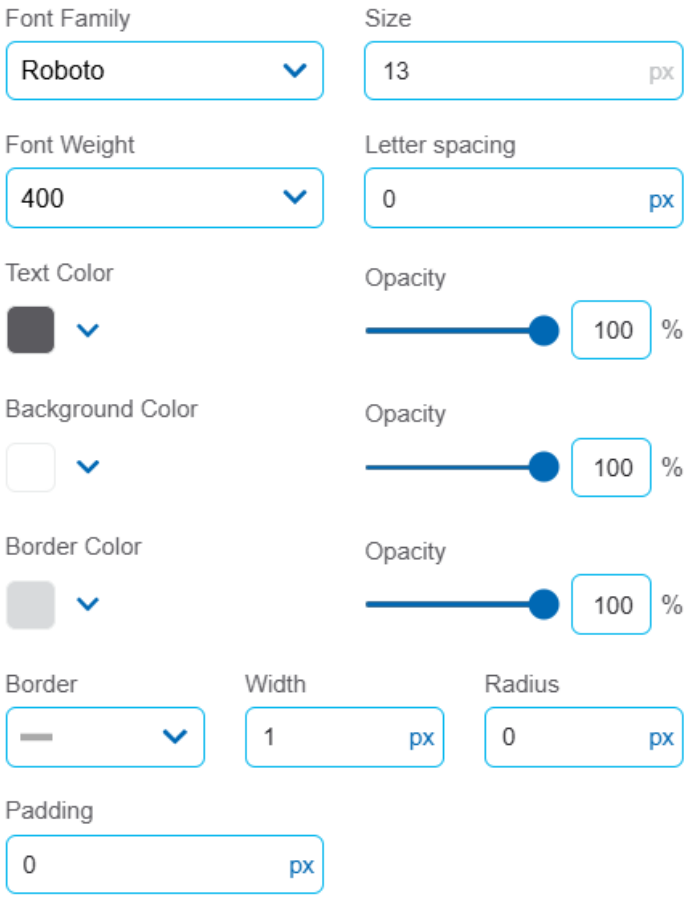
Color

Defines the color of the shadow.

Transparency

Defines the transparency of the shadow.

Tooltip



Source

It allows customizing the font, size, thickness and letter spacing of the pop-up message.

Colors

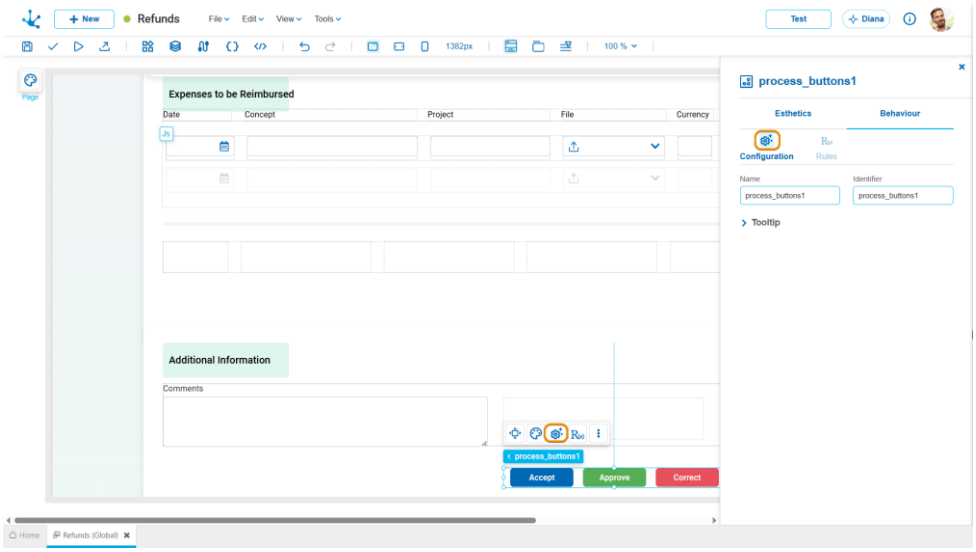
The color and opacity of the text, background and border can be defined. Borders are set uniformly for all sides of the tooltip.

Border

The border type, width, and border radius can be configured.

Configuration Properties

The configuration properties panel of an element opens when clicking the icon of its context menu.



Name
Name used to reference the element during modeling.

Identifier
Uniquely identifies the element. It is used in the Javascript SDK.

Tooltip
Pop-up message that appears when hovering over an element, providing additional information.



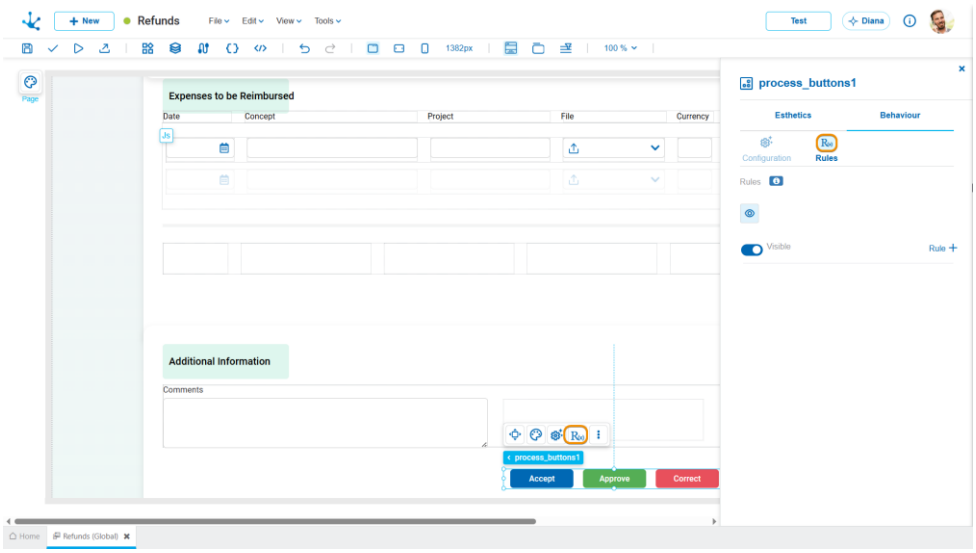
Text
Content of the message being displayed.

Direction
Position of the message in relation to the element (top, bottom, left, or right).

Modelable in Activity
When enabled, it is possible to model its behavior through an activity in the container section.

Rule Properties

The rule properties panel of an element opens when clicking the icon of its context menu.



Properties

Rules

Embedded rules on behavior, validation and calculation can be defined, associated with an element, by using the wizard (ctrl + space).

Shows syntax examples for writing the rules.

Visible

Indicates whether the element is visible. If this property is not checked, the element is not displayed in the page.

Visible (default) Not visible

Rule + Opens an edit area where a rule to determine the visibility condition can be defined. If a rule is defined, the icon is displayed with light blue borders.

Operations when defining the rule:

Saves the new or modified rule

Cancels the operation

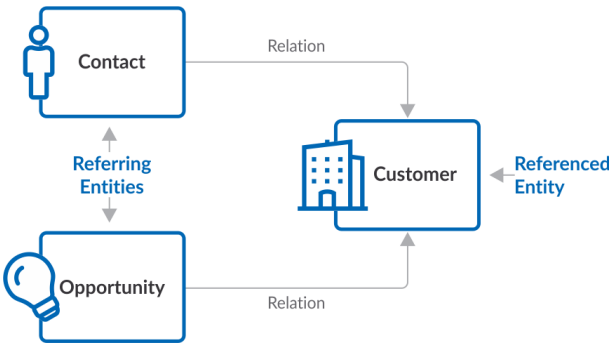
Operations once the rule is defined:

Edits the existing rule

Deletes the rule

Relation Navigation

Allows modeling of how navigation references between related instances are displayed and behave.



When a relationship navigation element is created in the modeling area, a navigation menu is incorporated, where links to navigate the relationships can be configured.

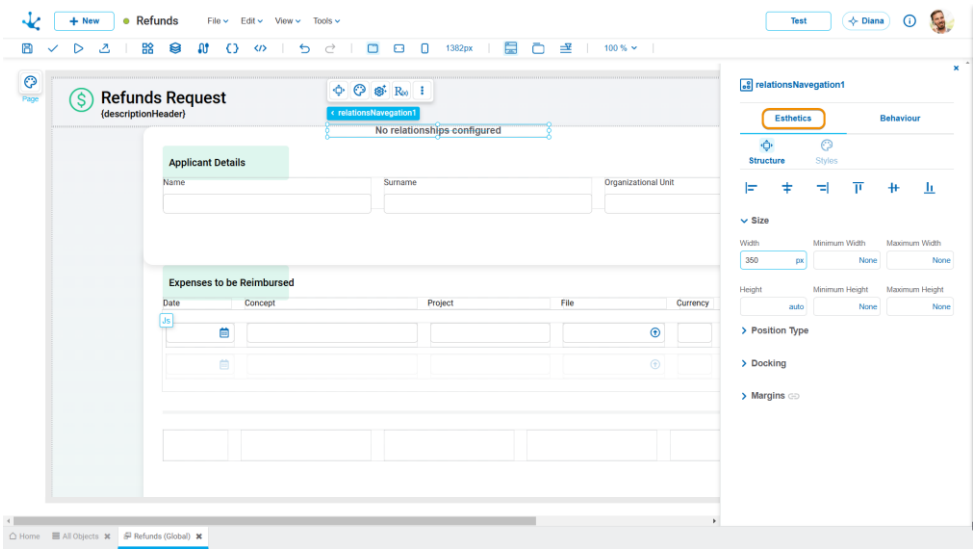
Classification of Properties

Properties are divided into two groups: esthetic properties and behavior properties.

Esthetic Properties

In the esthetic properties panel, the following are grouped:

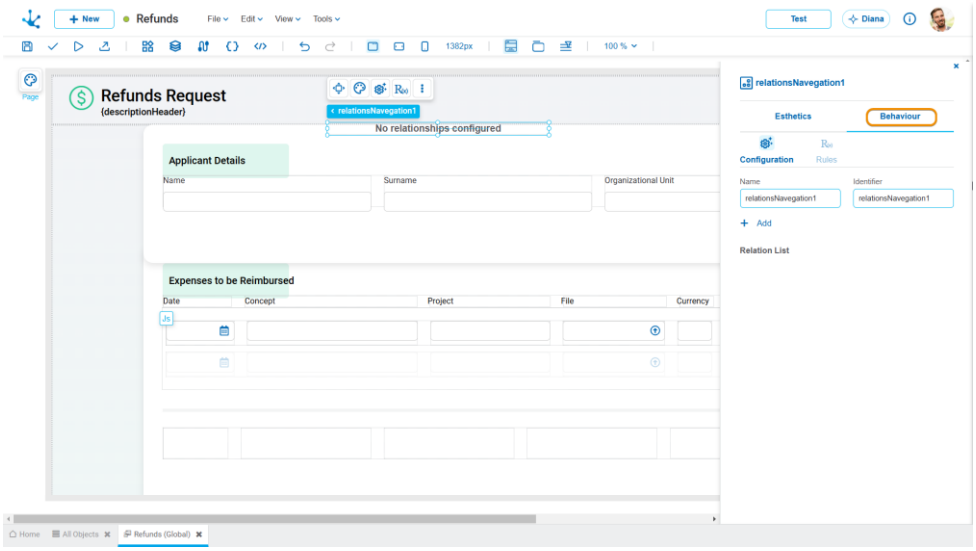
- Structure Properties
- Style Properties



Behavior Properties

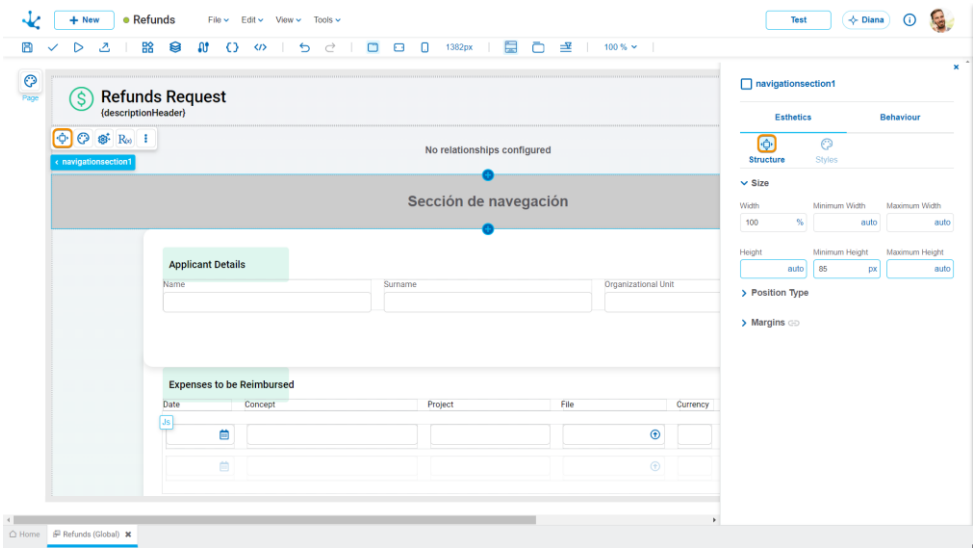
In the behavior properties panel, the following are grouped:

- Configuration Properties
- Rule Properties

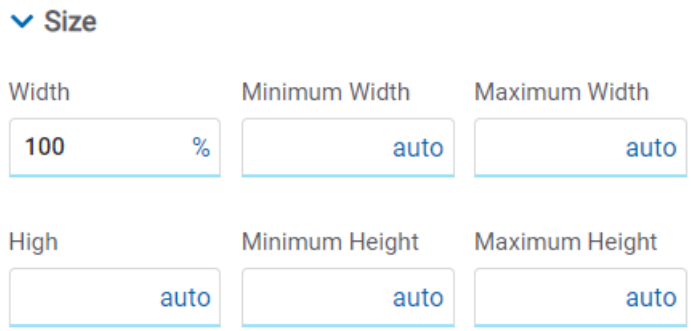


Structure Properties

The structure properties panel of an element opens when selecting the icon  of its context menu.



Size

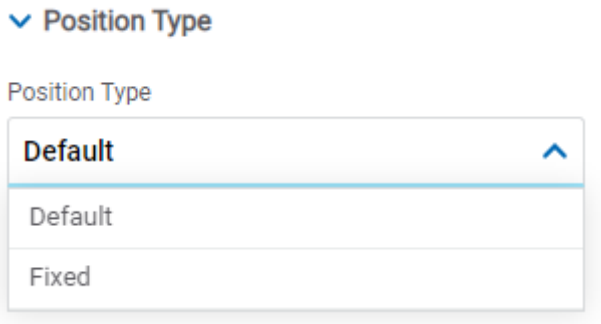


All size properties can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh). Additionally, for Width and Height properties, the "auto" option is added.

> *If selecting percentage (%) for the width and height properties, the size is calculated relative to the top element.*

Position Type

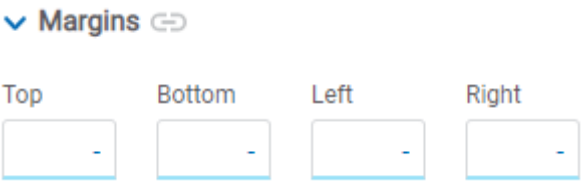
It determines if at the time of execution the element remains fixed on the page or if it moves as the scroll bar moves forward or backward.



Possible Values

- Default: The element has a relative position with respect to the superior element where it was placed (container or section).
- Fixed: Fixes the element to the page so that it is always visible in the same position. It allows to increase the visibility of important content.

Margins



It allows to define the distance of elements from the borders of their top element. The behavior of margins depends on the docking of the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Top
Distance to the top border of the highest ranking element.

Bottom
Distance to the bottom border of the highest ranking element.

Left
Distance to the left border of the highest ranking element.

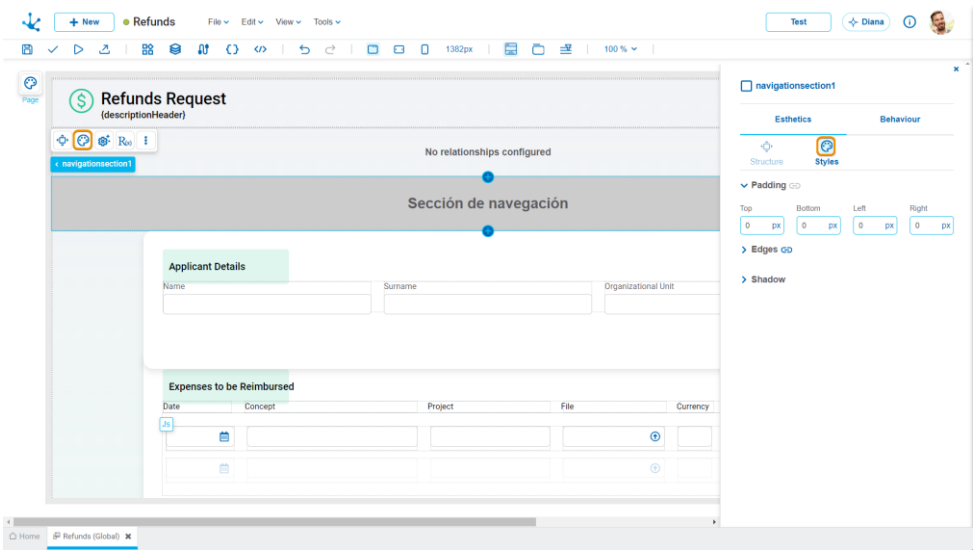
Right
Distance to the right border of the highest ranking element.

Allows the value entered in one of the margins to be copied to the other ones automatically.

Allows to indicate different values for each margin.

Style Properties

The style properties panel of an element opens when selecting the icon of its context menu.

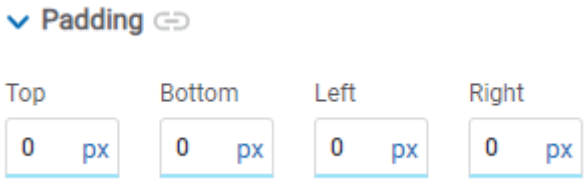


Styles

It allows copying the entire current configuration of the style panel for the selected element (Ctrl+Alt+C).

It allows applying the previously copied configuration to another element of the same type (Ctrl+Alt+V).

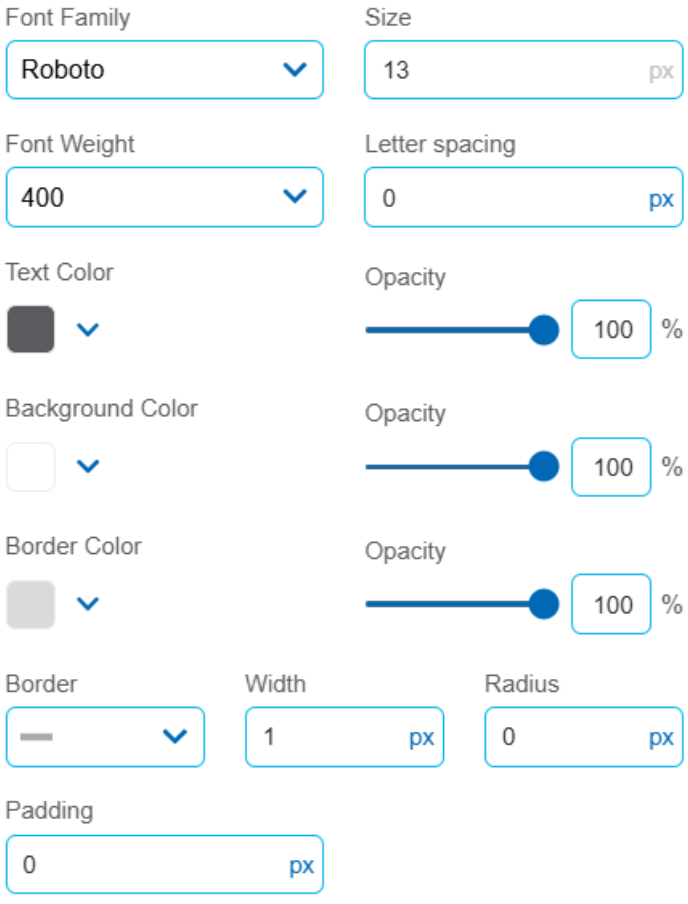
Padding



All padding space properties allow to create a space around the element borders (top, bottom, sides) and its content. The padding space configured is inward, while the margin is configured relative to a top element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Allows values entered in one of the paddings to be copied to the other ones automatically.
Allows to indicate different values for each padding.

Tooltip



Source
It allows customizing the font, size, thickness and letter spacing of the pop-up message.

Colors

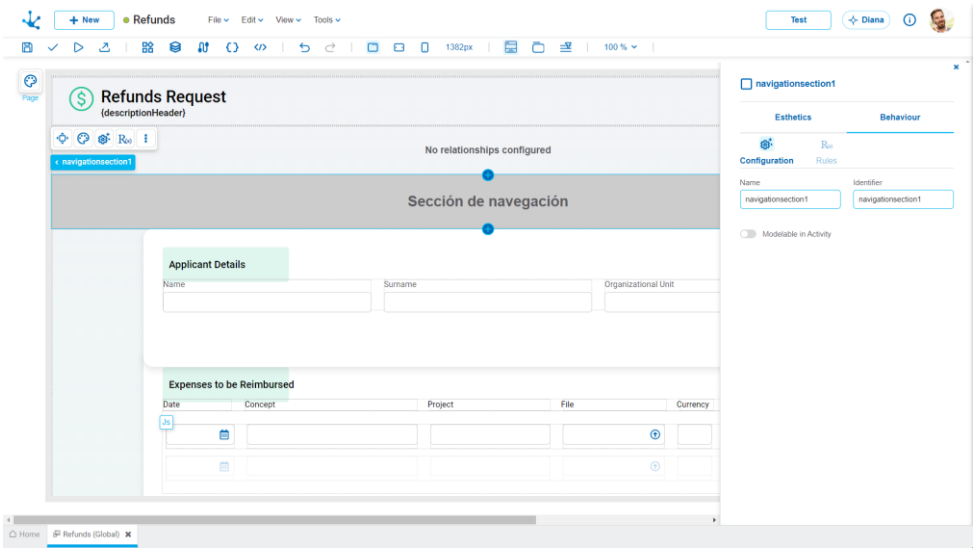
The color and opacity of the text, background and border can be defined. Borders are set uniformly for all sides of the tooltip.

Border

The border type, width, and border radius can be configured.

Configuration Properties

The configuration properties panel of an element opens when clicking the icon  of its context menu.



Name

Name used to reference the element during modeling.

Identifier

Uniquely identifies the element. It is used in the Javascript SDK.

Tooltip

Pop-up message that appears when hovering over an element, providing additional information.



Text

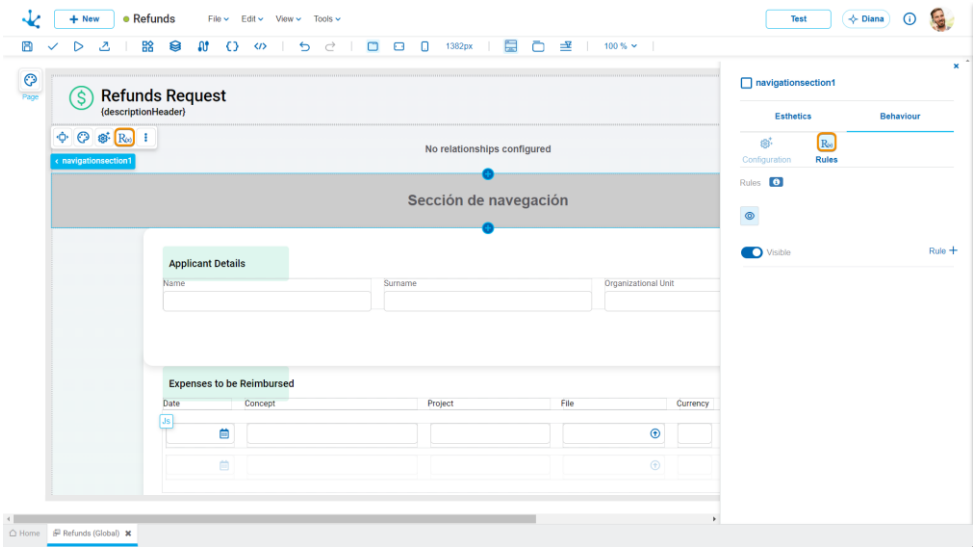
Content of the message being displayed.

Direction

Position of the message in relation to the element (top, bottom, left, or right).

Rule Properties

The rule properties panel of an element opens when clicking the icon R(x) of its context menu.



Properties

Rules

Embedded rules on behavior, validation and calculation can be defined, associated with an element, by using the wizard (ctrl + space).

Shows syntax examples for writing the rules.

Visible

Indicates whether the element is visible. If this property is not checked, the element is not displayed in the page.

Visible (default) Not visible

Rule + Opens an edit area where a rule to determine the visibility condition can be defined. If a rule is defined, the icon is displayed with light blue borders.

Operations when defining the rule:

Saves the new or modified rule

Cancels the operation

Operations once the rule is defined:

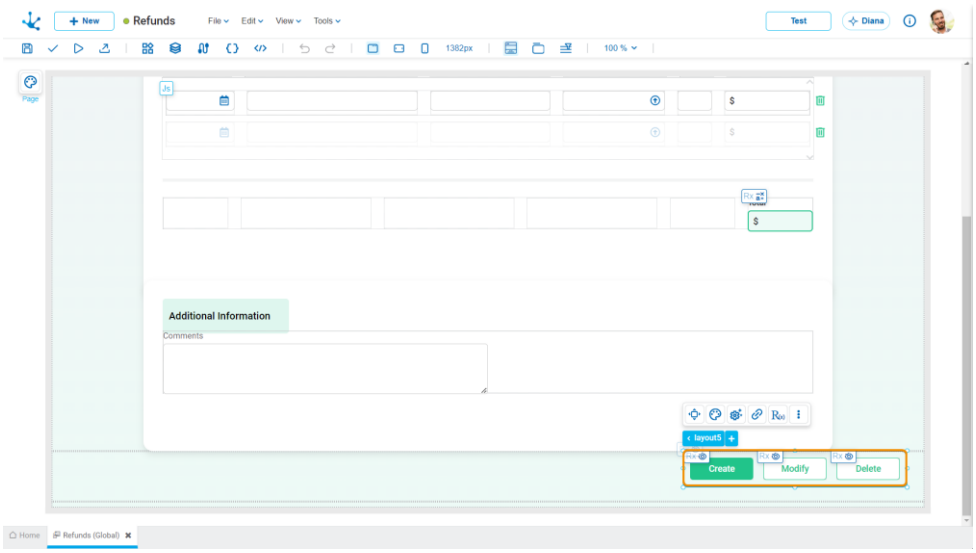
Edits the existing rule

Deletes the rule

Operation Buttons

Three operations can be carried out on an instance of an entity: creation, update, and deletion. This advanced element allows adding a button to the interface for each available operation on the entity when used in an application.

This advanced element is based on a set of simple elements, each of which has a default configuration to facilitate modeling. By analyzing how it is composed, a layout can be identified, with a visibility rule that allows it to be hidden, if the entity is being executed in a process. This is because it is not feasible to access these buttons while a process is running, as the process activity determines the operations to be performed. Within this layout there are Items, each item contains a button and these items are hidden depending on the operation being executed. For example, if a creation operation is being executed, it doesn't make sense to show the delete button. This logic is controlled by the modeling of conditions. Each button has a configuration that allows performing an operation to administrate the data of an instance, being able to create, modify, or delete that instance.

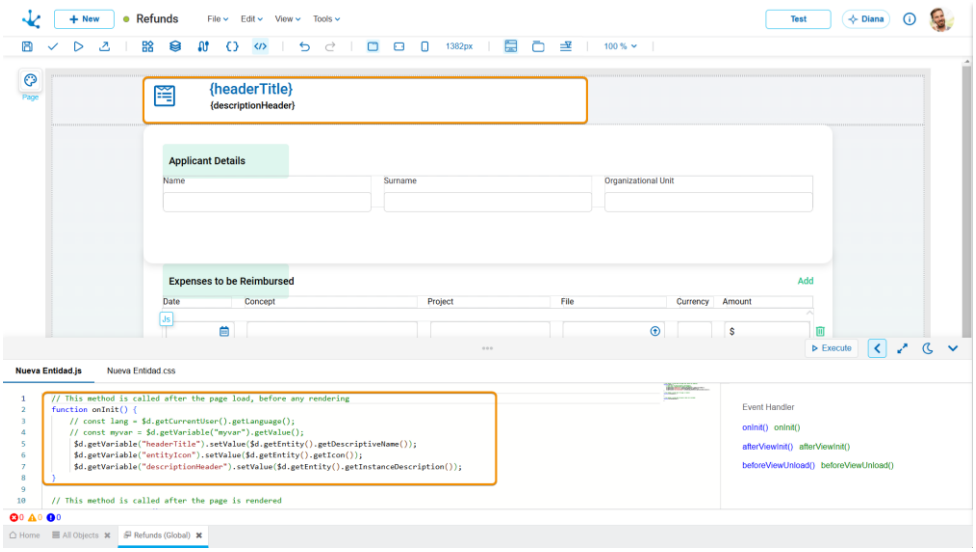


Header

The entities modeler allows including a header as part of the entity.

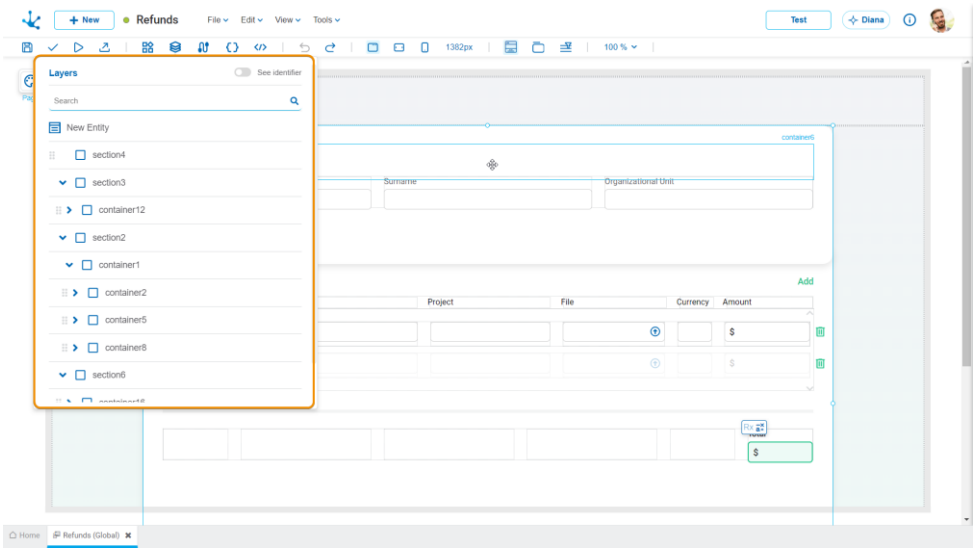
This element is composed of a page that includes the descriptive name of the entity, its icon and a description. Through JavaScript code, the elements of this page are built in with the values of the entity properties.

- headerTitle: Takes the value of the property Descriptive Name.
- descriptionHeader: Takes the value of the property Description.
- entityIcon: Takes the value of the Icon property of the entity.



3.4.11.1.5.2. Layers

It allows to display the entity elements organized hierarchically. First, sections are displayed, ordered as they are located on the entity and within each one, all its elements can be seen.



Operations

Locate Elements

Hovering over an element in the layers panel highlights such element in the modeling area, making it easier to identify without the need to select it. If selected, in addition to being highlighted, its context menu is displayed.

When an element is selected in the modeling area and the layers panel is opened, the chosen element is displayed in the hierarchy.

Filter Elements

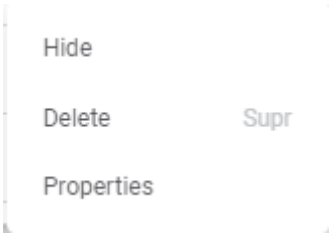
When performing a search, matching elements are highlighted in the layers menu, while those that do not meet the search criteria collapse their hierarchical display, if they had one.

Locate and Show Hidden Elements

Hidden elements on the entity are displayed in gray in the layers panel next to the icon . When pressing this icon, the element becomes visible. When clicking on the name instead, the element is temporarily displayed in the modeling area.

Elements Panel

Selecting the icon displays a panel with different options.



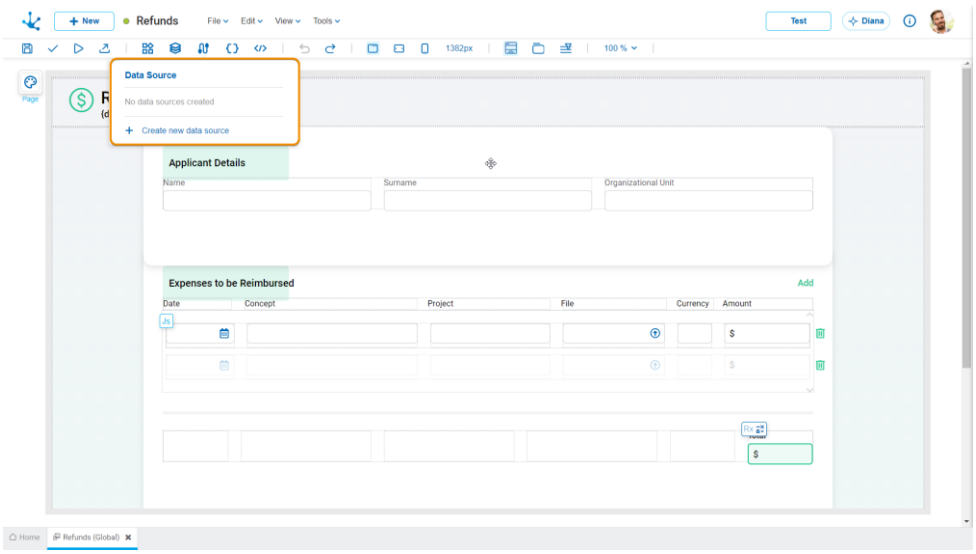
Reorder Elements

It allows to easily order the elements within the same superior element, dragging them in the list to the desired position.

3.4.11.1.5.3. Data Source

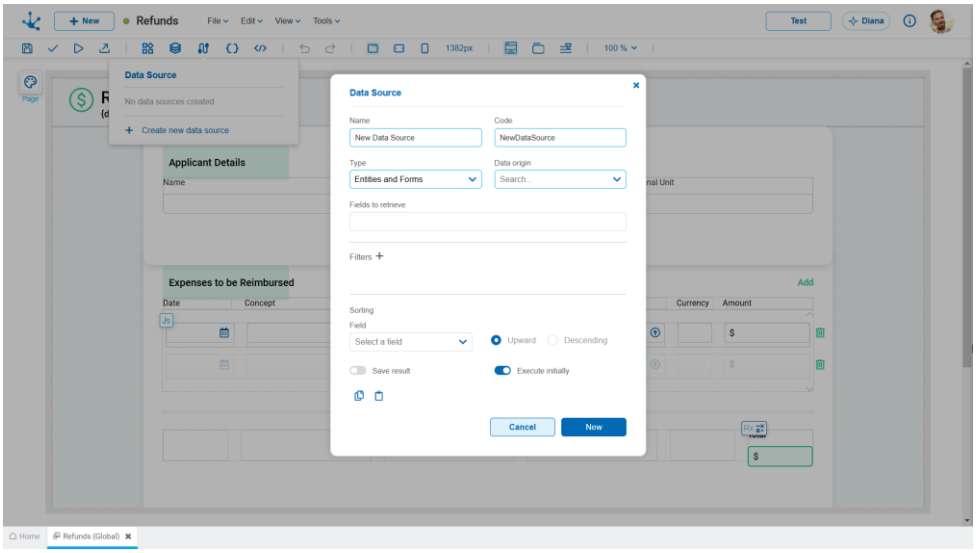
A data source retrieves data from entities and forms or executes advanced rules depending on the properties modeled on it.

Through the icon ,a set of data sources to be used on the entity can be defined.



Operations can be performed on each line of the existing sources. Click on the icons and that delete or update it respectively.

A new data source can be created from the icon . When defining a new data source or selecting an existing one, a panel with its information opens.



Properties

Data sources can originate in entities, forms or in rules, in all cases, they share some properties and have specific ones.

Name

Name used in the modeler to reference the data source.

Code

It is used internally to reference the data source.

Type
Determines the type of object from which the data is to be taken. The possible values for this property are “Entities and Form” and “Rule”.

Data origin
It allows to select the object from which the data will be taken.

Sorting

Field
It allows to select the field the information will be sorted by in ascending or descending order.

Save result
It allows defining whether the data obtained by a data source should be stored temporarily. This functionality is useful for optimizing entity performance and reducing unnecessary requests to the server.

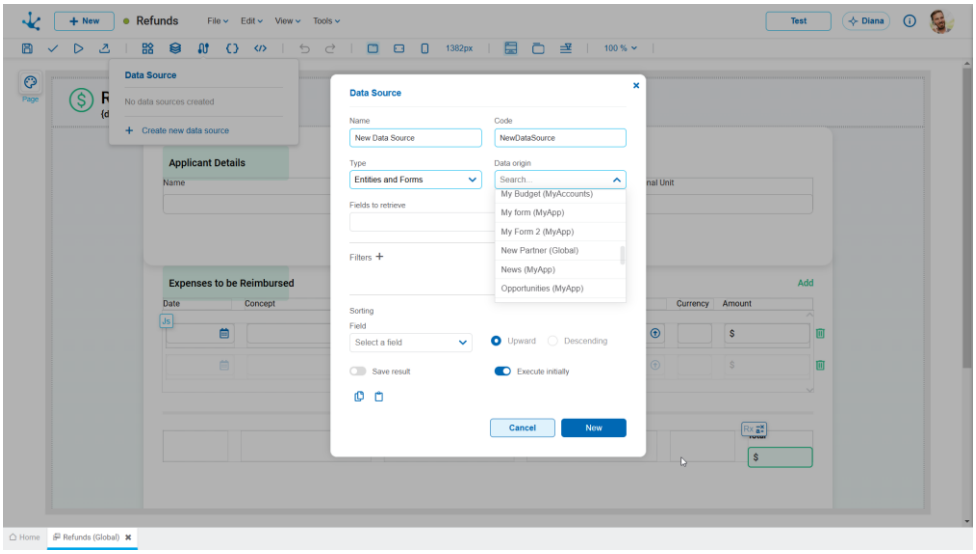
Execute initially
It allows configuring whether the data source is automatically executed when loading the entity.

Copy data source
Allows copying the entire current definition of the selected data source to another entity.

Paste data source
Allows applying the entire current definition of the copied data source to another entity.

Entities and Forms as Data Source

Specific properties must be modeled when the data source is of entity type or form type.



Properties

Fields to retrieve

It allows the fields of the selected object to be selected.

Filter

The icon + allows to expand a panel with the available fields to define the selection criteria of the data.

Depending on the type of field, filter conditions are different, fixed values, variables or parameters can be entered and combined by means of operators. Once the filter criteria have been defined, the "Add" button should be pressed so as to define the filter.

Filter Conditions

Numeric Fields

The value entered should be numeric.
Search criteria:

- Greater than
- Greater equal to
- Less than
- Less equal to
- Between
- With Data
- No Data

Alphanumeric Fields

Enter a text to search for.
Search criteria:

- Contains
- Equal to
- Starts with
- Does not start with
- No Data
- With Data

Date Fields

A calendar opens to select the date, and it can be filtered using different search criteria.
Options:

- Today
- Last 7 days
- Current Month
- Current Year
- Last Month
- Last Year
- From (Requires selection of a start date)
- To (Requires selection of an end date)
- Range (Requires the selection of a start date and an end date)
- Equal (Requires selection of a date)

DateTime Fields

A calendar opens to select the date and time, it can be filtered using different search criteria.

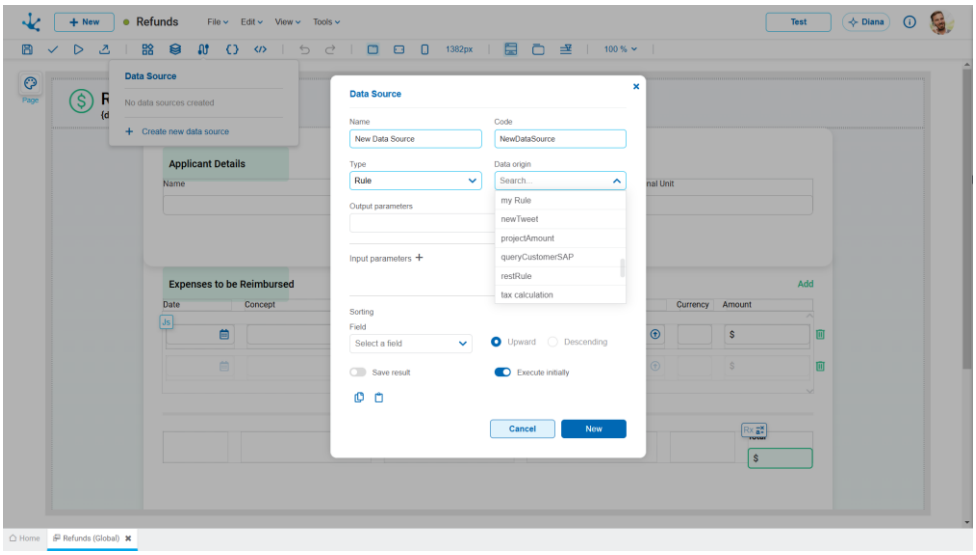
- From
- Until

- Equal
- Range

Value Lists Fields
The list values are displayed.
Search criteria:
- Included
- Not Included
- With Data
- No Data

Rule as Data Source

When the data source is of the rule type, specific properties must be modeled.



Properties

Output parameters
The output parameters of the selected rule are displayed.

Input parameters
The icon + allows expanding a panel with all the input parameters available for selection.
Entity parameters, fixed values, or variables can be entered for each chosen parameter.

3.4.11.1.5.4. Parameters and Variables

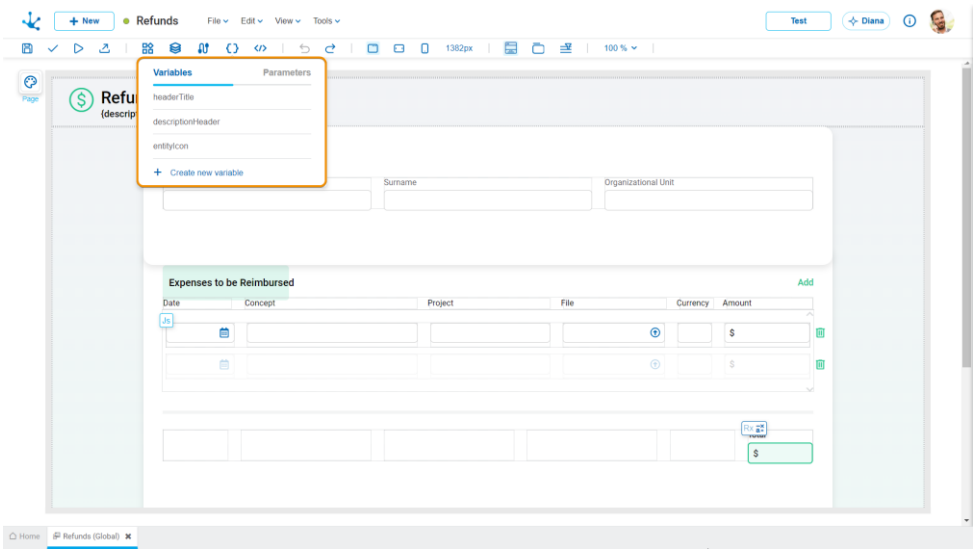
The modeled entity allows to define variables and input parameters by means of the icon {}. Once the panel is opened, variables are displayed by default, parameters can also be selected.

Variables

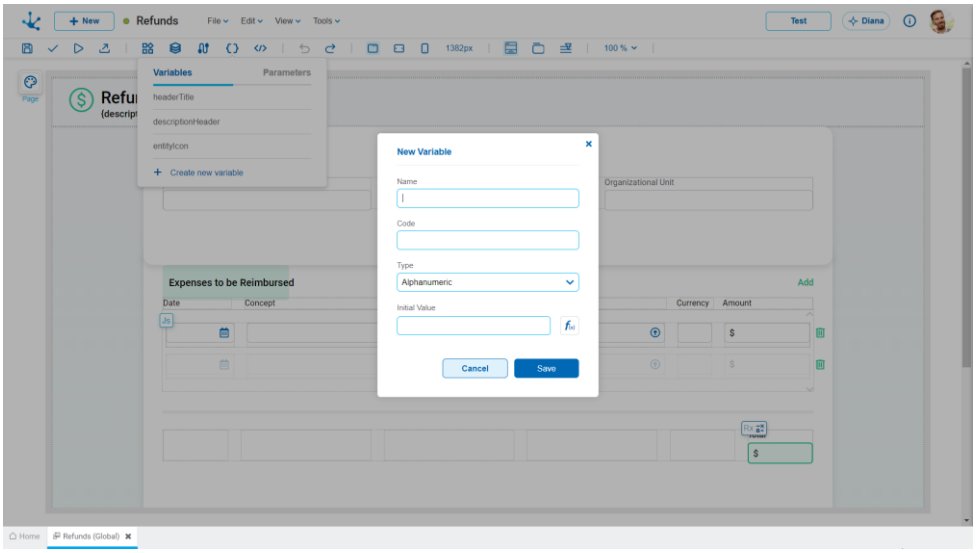
Variables used in entities are defined.

They allow to define data that can be used as the value of elements and in the filters of data sources. They can only be modified by code.

Operations can be performed on variables. Click on the icons and on each row to delete or update it respectively.



A new variable can be created from the icon . When a new variable is defined or an existing one is selected, a panel with its information opens.

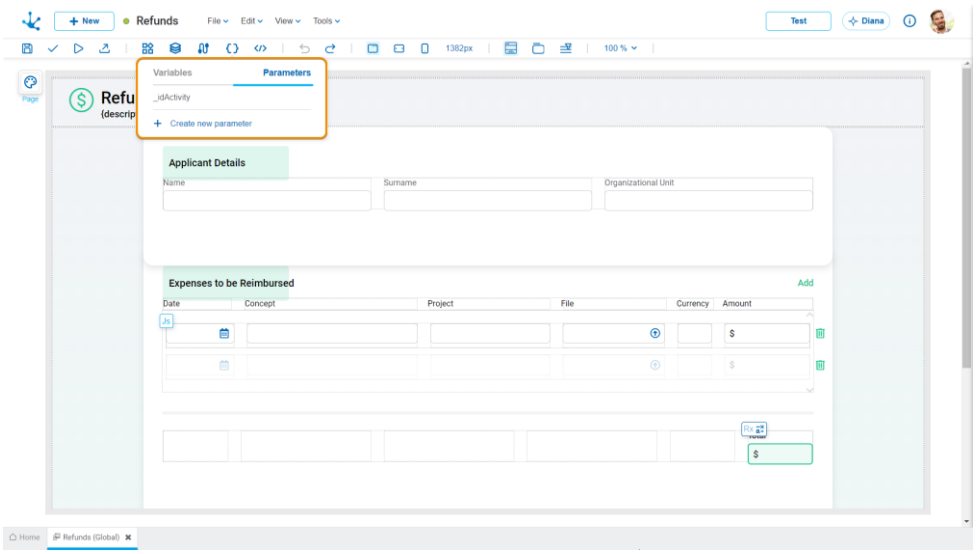


Parameters

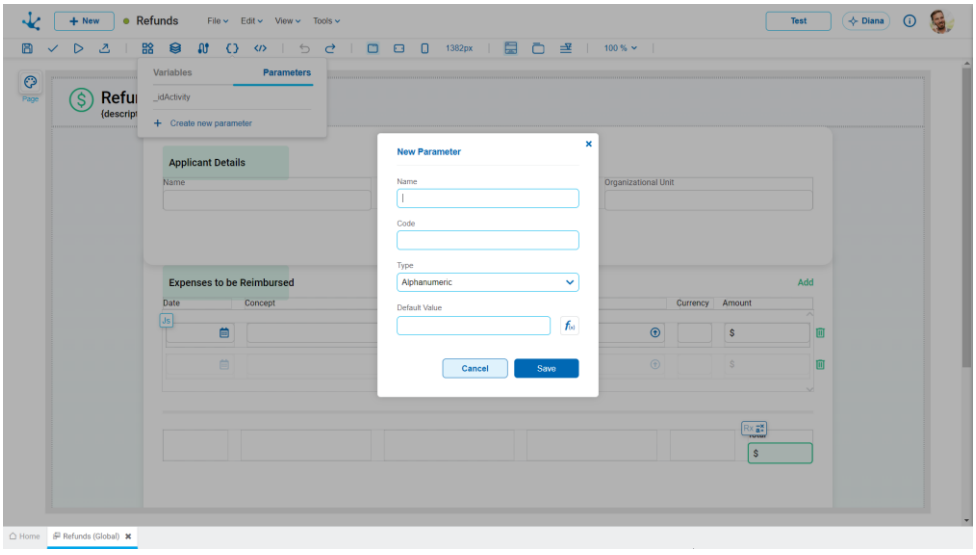
The parameters used in entities are defined.

Data received by the entitiy that cannot be modified. As with variables, they can be used in elements and data sources.

Operations can be performed parameters. Click on the icons and on each row to delete or update it respectively.



A new parameter can be created from the icon . When defining a new parameter or selecting an existing one, a panel with its information opens.

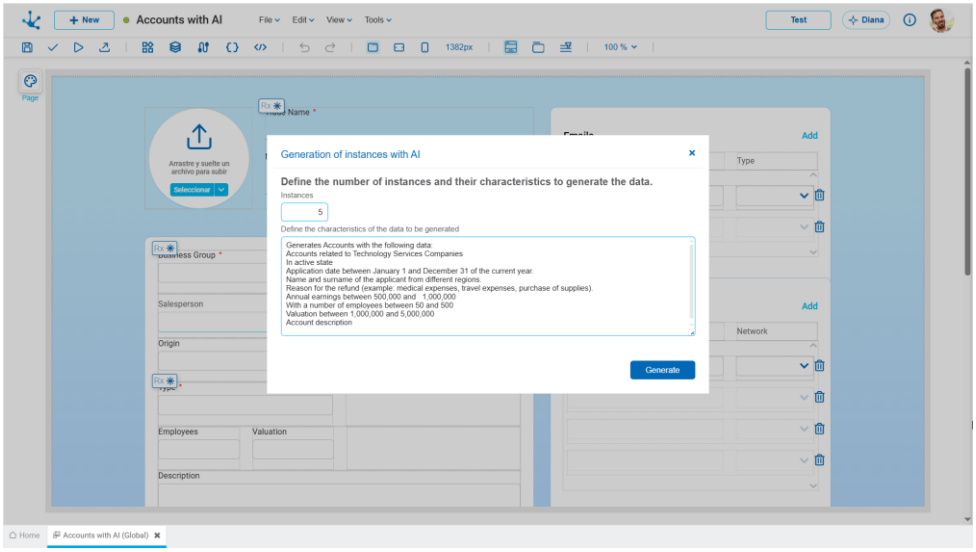


3.4.11.1.5.5. Generación de Instancias con IA

AI-powered entity instantiation allows for the quick population of an entity with contextualized data tailored to user requirements.

This functionality streamlines development by facilitating testing, demonstrations, and configuration of working environments with a consistent, functional data set.

Using this facility opens a panel where the number of instances (up to 20) and a prompt with the data specifications can be entered.
By clicking on "Generate", data is created in the edited entity.



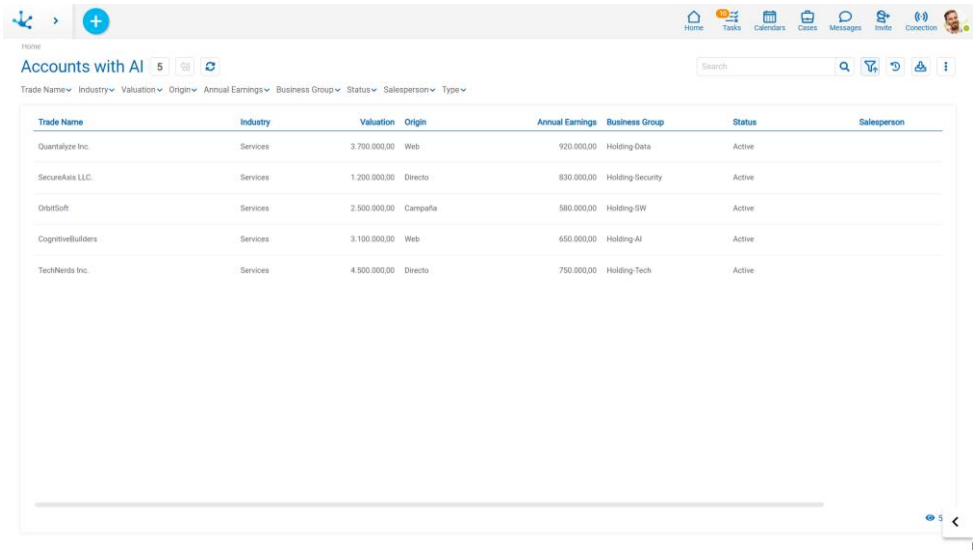
Artificial intelligence generates data based on the context defined in the prompt, the clarity and detail of which influence the accuracy of the result.

> *It is suggested not to indicate the use of a value list in the prompt.*

Example of data generated from a prompt

If the following prompt is used to generate instances with artificial intelligence for the entity “Accounts with AI”, the result in the image is obtained.

```
Generates Accounts with the following data:
Accounts related to Technology Services Companies
In active state
Application date between January 1 and December 31 of the current year.
Name and surname of the applicant from different regions.
Reason for the refund (example: medical expenses, travel expenses, purchase
of supplies).
Annual earnings between 500,000 and   1,000,000
With a number of employees between 50 and 500
Valuation between 1,000,000 and 5,000,000
Account description
```



3.4.11.1.5.6. Code Editing

It is the area within the modeler where the functionality of the entity and its elements can be extended through code.

This area contains two tabs:

- JavaScript Tab: It allows writing custom JavaScript functions to add interactivity to the entity, dynamically handle data, manage events, and integrate with other services or scripts.
- CSS Tab (Cascading Style Sheets): It is used to define custom style rules that modify the appearance of the entity and its elements. Here, specific styles such as colors, sizes, borders, and animations can be applied, ensuring that the design meets the project's requirements.

The code editing area is displayed at the bottom of the entity, following a bar with icons. The icon ••• in the center of the bar allows you to modify the size of the area, while the icon ⌄ on the far right allows to close it.

Editor Options

▷ Execute It allows you to have a view of the modeled entity, showing how it is displayed on different devices.

Increases the panel size and covers the entity modeling area.

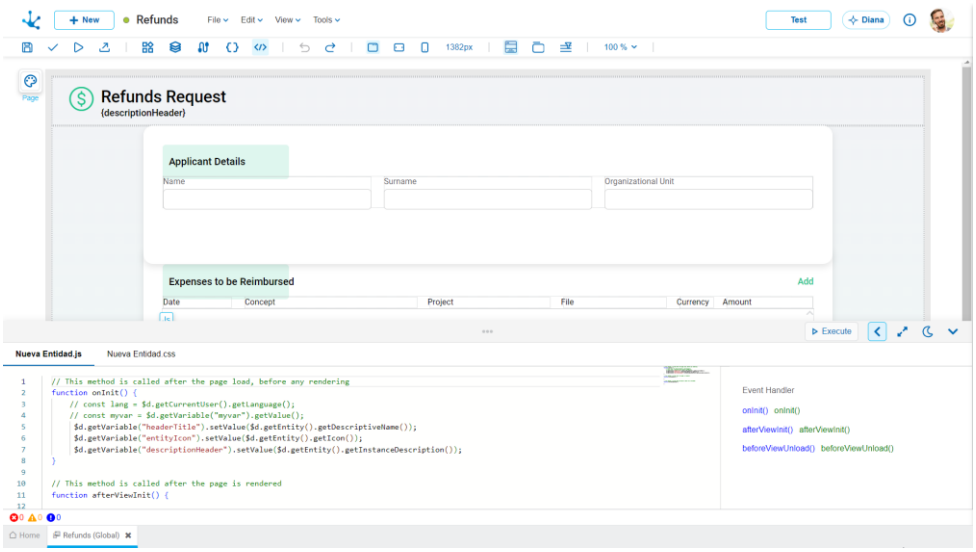
Returns the panel to its previous size.

Changes the style of the code editor to dark mode.

Changes the style of the code editor to the previous mode.

JavaScript Tab

Each element has a list of built-in events depending on the element type.



Sections

- Javascript code editor of the entity: It allows to add and update functions associated with the events of elements or entities. Includes an auto-complete functionality that speeds up writing by automatically suggesting SDK functions.

- Event handler: The event handler is displayed to the right of the code editor, it contains the identifier and the list of events of the selected element. If no element is selected, the list of entity events expands.

- Message Monitoring Console: Located below the code editor, this advanced tool facilitates the monitoring and management of messages during development.

Operations of JavaScript Functions

Associate

The entity has JavaScript functions associated with its events defined by default, unlike its elements, which do not have them.

To create and associate a JavaScript function with an event, select the event in the event handler, enabling the new function name to be entered. By default, a name is generated consisting of the element identifier concatenated with the event name. Clicking on the icon + creates the JavaScript function associated with the event in the code area, and the name of the function is added to the event list.

If the name of the JavaScript function already exists within the code, a number is automatically suffixed to the name.

Select

Selecting a JavaScript function in the event handler positions the cursor over it in the code editor.

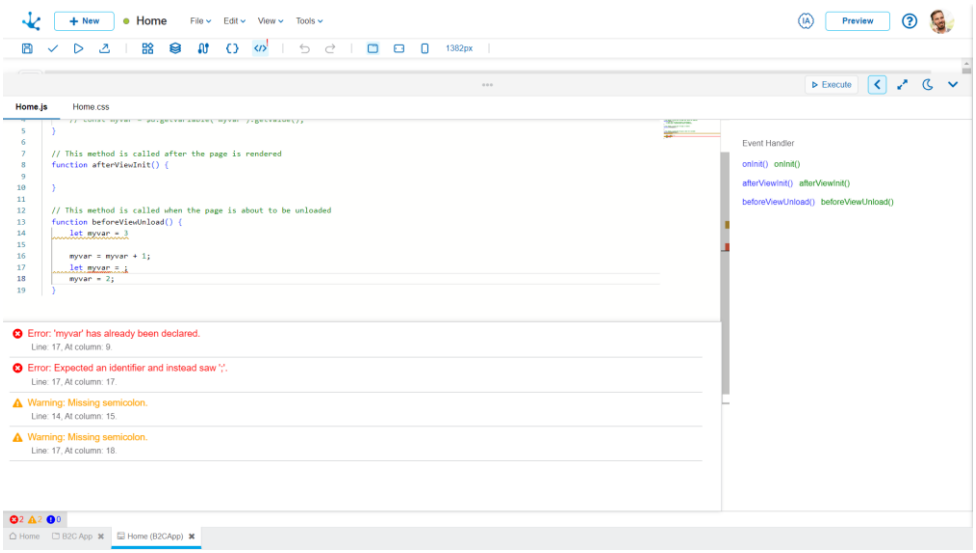
Disassociate

To disassociate a JavaScript function from an event, click on the icon that is displayed when hovering the cursor over it in the event list. In this way, the association of the function with the event is deleted but it is not deleted from the code.

> *If the JavaScript function is deleted from the code area, it still appears associated with the event handler. However, the event does not execute any function and does not throw an error.*

Message Monitoring Console

You can access this console by clicking on the icons corresponding to the messages, which are located in the bottom bar of the code editor.



Upon accessing the console, the messages are displayed in an organized and categorized manner. Details about each message are provided, including its location in the code and possible solutions. The messages are updated in real time, with continuous monitoring that allows you to visualize errors and warnings as they are detected.

Message Types

Syntax Errors: Detects and notifies syntax errors in the code.

Warnings: Highlights potential issues that could affect the performance or quality of the code, providing alerts on aspects that may need attention or review.

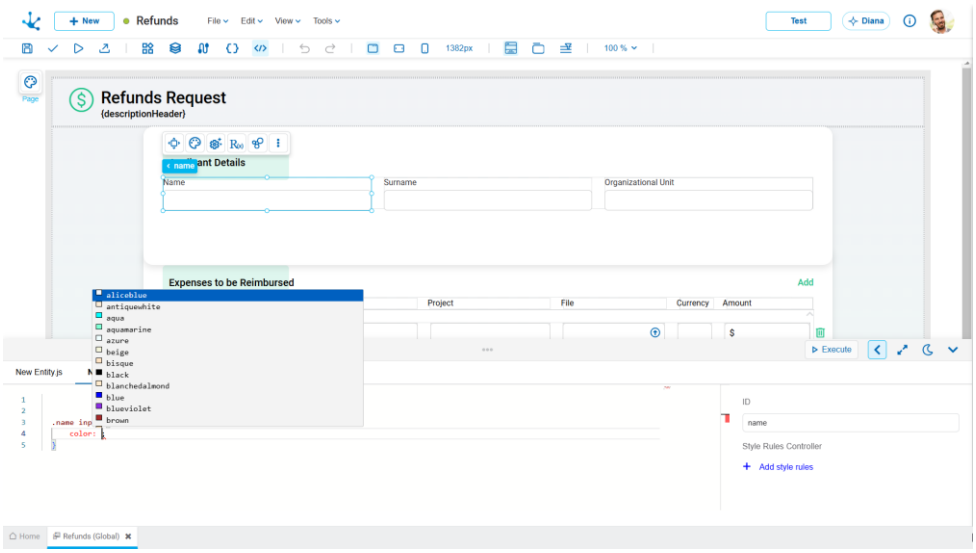
Usage Information: Identifies situations in the implementation of methods and functions, offering recommendations to improve the code.

CSS Tab

The Styles tab allows applying CSS rules to change the appearance of elements within the modeling tool.

Sections

- CSS code editor: Rules can be written manually or by selecting attributes using a wizard, making it easier to configure colors, fonts, sizes, margins, and other visual settings.
- Rule controller: The rule controller is displayed to the right of the code editor, which allows managing and applying specific styles to selected elements. If no element is selected, no associated rules are displayed, but if an element is selected, custom styles can be added, modified, or deleted for that element.



Steps to change the style of an element

- Navigate to the CSS tab: Initially, the panel does not contain applied style rules.
- Selecting an element: When clicking on an element within the modeling area, its ID is displayed in the interface. In the Style Rule Controller section, the option "+ Add style rule" is displayed.
- Add style rule: When this option is clicked, a CSS rule is automatically generated in the code editor, named according to the identifier of the selected element. Multiple rules can be created for each element.
- Edit the rule: Within the rule body, press "CTRL + spacebar" to invoke the wizard. Select the attribute and the value to be modified so as to customize the design as needed.
- Apply changes: Click on the button "Execute" from the editor to display the updates.

Eventos

Entities, their sections, and their elements have different events defined. These events are displayed in the area of page code, where code can be entered, allowing it to execute when a specific event occurs.

Entity

Event	Description
onInit()	It is executed before loading the entity.
afterViewInit()	It is executed after rendering the entity.
beforeViewUnload()	It is executed before closing the entity.

Sections

Event	Description
onMouseIn()	It is executed when the cursor is positioned on the element.
onMouseOut()	It is executed when the cursor moves out of the element.
onClick()	It is executed when clicking on the element.
onDoubleClick()	It is executed when clicking twice on the element.
onInit()	It is executed before the element is loaded.
afterViewInit()	It is executed after viewing the element.
onViewportEnter()	It is executed when the element is visible.
onViewportLeave()	It is executed when the element is no longer visible.

Elements

The events shared by all elements are detailed.

Event	Description
onMouseIn()	It is executed when the cursor is positioned on the element.
onMouseOut()	It is executed when the cursor moves out of the element.
onClick()	It is executed when clicking on the element.
onDoubleClick()	It is executed when clicking twice on the element.
onInit()	It is executed before the element is loaded.
afterViewInit()	It is executed after viewing the element.
onViewportEnter()	It is executed when the element is visible.
onViewportLeave()	It is executed when the element is no longer visible.

Button

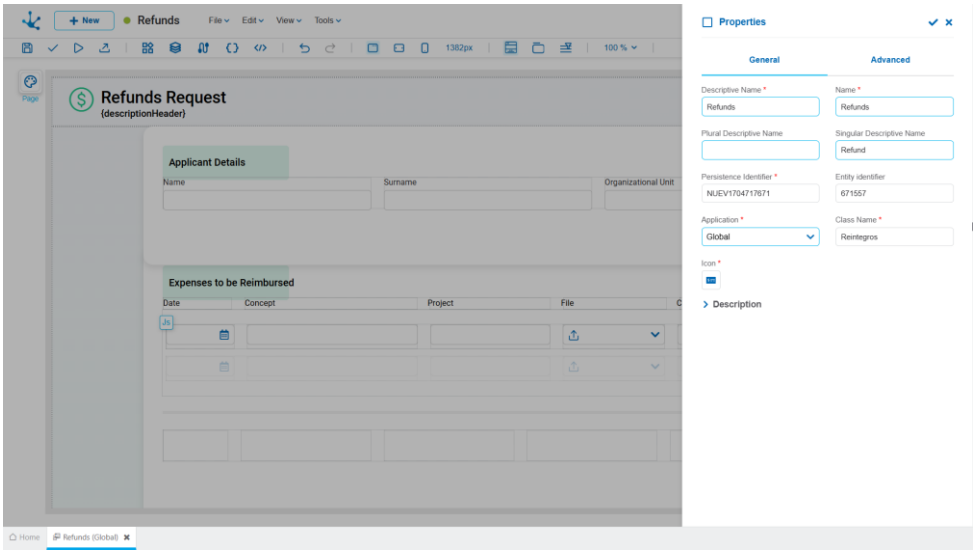
The events shared by all fields are detailed.

Event	Description
onInit()	It is executed before loading the entity.
afterViewInit()	It is executed after rendering the entity.
beforeViewUnload()	It is executed before closing the entity.

3.4.11.2. Entity Properties

The properties of entities can be entered both at the time of their creation and when modifying an existing one.

To enter the entity properties panel, use the icon  which is in the tools submenu.

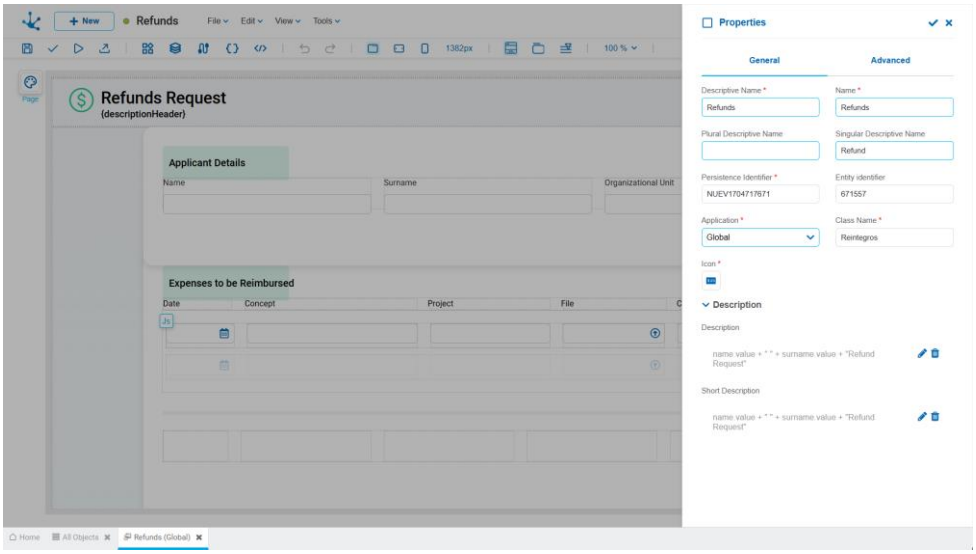


Tabs

- General
- Advanced

3.4.11.2.1. General

The properties panel is displayed on the right side of the entity modeler, where the first tab corresponds to general information.



An asterisk "" on the label indicates that the property is required.*

Properties

Descriptive Name
Name used by the user to reference the entity.

Name
It is used at the modeling level to reference the entity, for example in rules or as a parameter.

Plural Descriptive Name
The text entered in this property is used as a title in the entity's results grid, while if it is not filled, the Descriptive Name property is used.

Singular Descriptive Name
The text entered in this property is used in the list of related entities, while if it is not filled, the Descriptive Name property is used.

Persistence Identifier
Uniquely identifies the entity. It can be edited by the user as long as the entity is in draft state.

Entity Identifier
Uniquely identifies the entity. It is a number assigned by **Deyel**, not editable by the user.

Application
Allows to define the application to which the object belongs. If no application is reported, the object is assigned to the application "Global".

Class Name
Name that represents the object in the SDK service and model classes. It can only be modified if the object is in draft state.

Icon
It allows the entity to be assigned a representative image.
The code to be entered corresponds to the Font Awesome standard. The identifier of the icon selected from the https://fontawesome.com/v4/icons/ standard must be preceded by the characters "fa".

Description
Defines the entity describing its functionality and optionally its content.
It can include text and values of variables from the entity, which are modeled using the variables wizard. The use of variables included in repeaters and variables of the file and image types is excluded.
In the execution of the entity, the description entered is displayed in:

- The last updates, on the element that represents the entity in the Forms and Tasks grid.
- In the entity header, if the entitydescription variable is used.

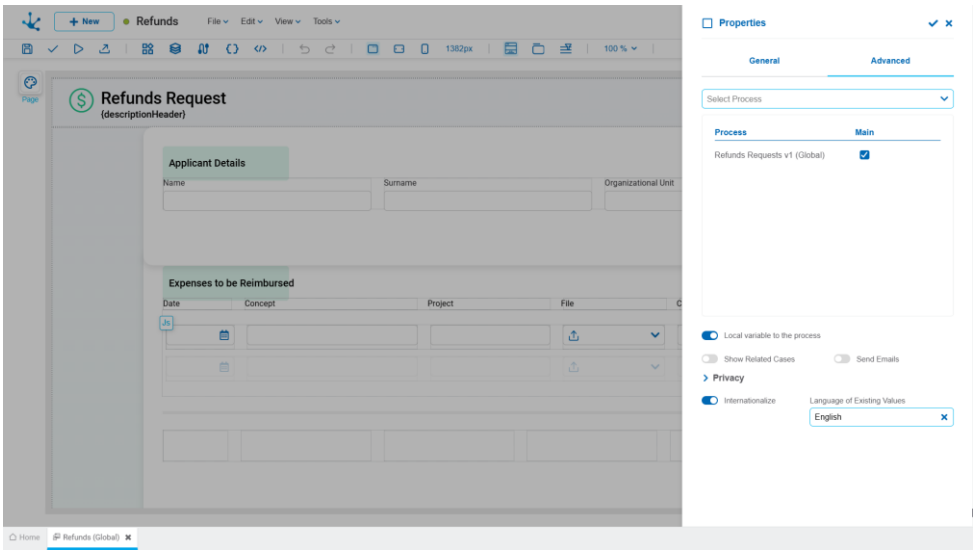
Short Description
Text oriented to be a shortened description of an entity instance. It can be used to identify the entity in related entities, in a more user-friendly way than using its identifier.
It can include text and values of variables from each entity instance, modeled using the variables wizard. Using variables corresponding to repetitive elements, files, images, and toggles is excluded.

In the execution of the entity, the identification entered in this property is displayed when using the referring entity field in:

- The entity instance.
- The entity's grid.
- The filter panel of the entity’s grid.

3.4.11.2.2. Advanced

The second tab of the side panel corresponds to advanced information of the entity.



Properties

Select Process

Allows to select a process from the drop-down list of available processes, each of the chosen processes is added to the grid of Related Processes.
Once a process is selected, the property Configurable within Activity is enabled in the sections and containers of the entity.

Related Processes

Every entity can be associated with one or more processes. The user can select them from the set of available processes and define entity relation properties. Once selected, the processes are displayed in a grid with their corresponding properties.

- Process
- Main

  In complex processes with a large number of variables, these variables can be defined in different entities, all related to the process. If this property is checked, it means that the modeled entity is the most representative for the related process. This means that in the gallery of forms and tasks, the main entity is the one where the tasks are displayed, even if other entities are used in the different activities.

  A process can be related to several entities, of which only one can be indicated as the main entity. If in the Related Processes grid, the process with this property selected is already defined as main in another entity, then the user receives an error message when the entity is published.

Local Variables to the Process

Checking this property indicates that the entity fields have a local scope to the related process, that is, these variables can only be accessed and modified during the execution of the corresponding case. The variables of completed cases can only be accessed by them.

Leaving this property unchecked indicates that the variables have a global scope, that is, they can be accessed and modified by the operations of the corresponding entity, by each case of the processes that use the entity or by business rules.

Show Related Cases

If this property is checked, each time an entity instance is created or shown in the user portal, an additional section of data is displayed, to associate the entity with a case.

Send Emails

This property enables emails to be sent from entity instances, with the sending records associated with those instances.
When activated, the following functionalities are enabled:

- **Mass email**
  A selection column is added to the results grid of the entity. Selecting one or more instances displays an icon that allows sending emails in bulk to the recipients listed in the selected instances.
- **Operation "Send Email"**
  Configurable within the hyperlink properties of an element. Clicking on the element during the execution of the instance opens a panel to send the email. This operation is available in the hyperlink for the entity type object.
- **Operation "Emails"**
  Configurable within the hyperlink properties of an element. Clicking on the element during the execution of the instance allows for the display of the emails sent. This operation is available in the hyperlink for the entity type object.

Privacy

Hierarchical Privacy

Check that indicates if the entity is private or not. If this check is indicated, the Privacy by Permissions property is displayed.

Privacy by Permissions

Mark indicating the creation of security functions: "Show Private Instance", "Modify Private Instance" and "Delete Private Instance" to be assigned in the design option "Permissions" of the entity. These security functions are available if the entity is in "Published" state, since they are use functionalities.

Internationalize

Enables the translation of object properties and its elements into different languages. When activated, it enables the option "Internationalization" from the "Tools" drop-down menu and the property Language of Existing Values, which is displayed on the right.

Language of Existing Values

Displayed when the property Internationalize is activated for the first time and indicates the original language. Its selection is required and the available options are: "Spanish (Argentina)", "Spanish (Mexico)", "English" and "Portuguese".

3.4.12. Rules Modeling

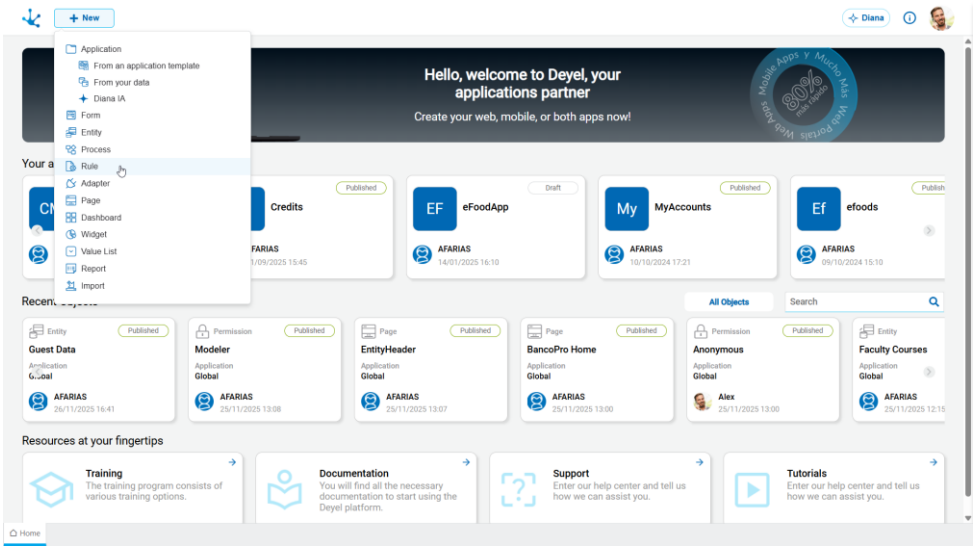
The rules modeler is the tool that allows defining the business logic within an application, either by creating advanced rules or integration rules. The former may involve validations, calculations, data

transformations, or other actions that are part of the functional behavior of the application, while the latter allow it to be connected with other external applications.

The rules are developed using different adapters which facilitate their configuration and execution, without the need to write complex code. This allows developers to define the required logic without worrying about the inherent complexity of communication.

Rules modeling incorporates **Diana**, the AI-powered conversational assistant of **Deyel**, that can guide the modeler user in creating rules using natural language, interpreting the request and generating the necessary code.

In this way, the modeler user can alternate between traditional modeling and AI-assisted modeling, using the path that is most convenient for each stage of development.



Its main characteristic lies in its simplicity to model rules using different modeling facilities.

The general characteristics of the rules modeling environment and the main elements that compose it are described in the following topics:

- Modeling Facilities
- Rules Properties

3.4.12.1. Business Rules

A business rule allows to perform a specific task, being able to receive information, perform certain processing and inform the results obtained.

In **Deyel** there are business rules used to define the behavior of entities, processes and forms. Rules can be used in validations, specific business logic, process flow control, integration with other applications, and field display control in entities, among other functionalities.

Types

There are rules that can be modeled without programming and others are based on Java code.

- Embedded Rules
- Advanced Rules

3.4.12.1.1. Embedded Rules

Embedded rules are used to define the behavior of application objects, such as entities, pages, processes and forms, without the need for programming.

These rules are defined using a simple syntax, similar to that used in spreadsheets, with the help of a wizard , if necessary. This wizard is integrated into the corresponding object modelers, within the property definition panels of fields, containers, forms and conditional flows, as well as page and entity elements.

Through the use of embedded rules it is possible to define calculations, validations and conditions that modify the behavior of application objects. In entities, pages, forms and process activities, it is possible to define requirement, visibility and editability rules, as well as calculation and validation rules at field or element level. Furthermore, in the case of processes, gateway conditions can also be modeled using embedded rules.

Unlike the advanced rules, embedded rules are not displayed into the modeler's grid, as they can only be used in the **Deyel** object where they are defined.

Types

Behaviour

A behavior rule is used to verify the fulfillment of a condition. A logical expression is evaluated and a "True" or "False" value is returned depending on whether the condition is met or not.

Embedded rules are modeled in the fields and containers of entities and forms, either from their respective modelers or from the process modeler when modeling their activities. They can also be defined in entity and page elements, as well as in process gateways when conditions are required.

They are classified according to the purpose of their use.

- Visibility
  The modeled condition defines whether a field, a page or entity element or a form section is visible to the user at the time of use.

- Editability
  The modeled condition defines whether an element of a page or entity or a form field are editable by the user at the moment of using them.

- Required

The modeled condition defines if a field or an element of a page or entity are required when using them.

Validation

A validation rule is used to ensure the proper information entry or to verify incorrect status. It can be set at the field level for entities, pages and forms.

This rule evaluates a condition and returns a message when it is met, indicating the wrong or invalid situation to the user. If no message returns, it means that the validation was successful.

Conditions involving one or multiple fields and the relations among them can be defined.

Calculation

To define arithmetic expressions that allow to perform calculations to fill in field values when using the entity, the page or the form. The rule is defined with a calculation algorithm for the target field that is populated with the resulting value. This value must be of the same type as the field that contains it.

Conditions can also be defined to determine when to execute the calculation rule.

Use

Embedded rules are modeled within the definition panels of **Deyel** objects that contain them.

Forms Modeler

- Rules in Field Properties
- Rules in Container Properties
- Rules in Form Properties

Processes Modeler

- Rules in Activity Properties
- Rules in Gateway Properties

Pages Modeler

- Rules in Element Properties

Entities Modeler

- Rules in Element Properties

3.4.12.1.1.1. Rule Elements

An embedded rule is an expression that is made up of operands combined by logical or arithmetic operators. where each operand can be a constant, a variable, or a function. Precedence can be determined by using ().

Examples

- Name == “John”
- Amount <= 25.5
- (2+5-1)/2
- addDays(day(),3)

Constants

Constants correspond to the different types of data used in **Deyel**.

- Alphanumeric
  Eg: “This is a Text constant”

- Numeric Integer or Decimal
  Eg: 10.3 where the decimal symbol corresponds to the value configured in the environment

- Date, Time, Date and Time
  Some of the exclusive rules functions can be used to parse from alphanumeric format to the selected date type constant.
  Eg: parseDate("07/25/2023")
  parseTime("15:20")
  parseDateTime("07/25/2023 15:20")

- Logical
  Eg; True,False

Variables

Variables can be fields, group containers, iterative containers, related form fields, entity and page element properties, variables, and page parameters. The variables available for expression depend on where the rule is embedded.

Both form fields and page elements are referenced by name.

The type of data they represent is displayed to the right of the name.

- Fields

  In the wizard, they are identified with ● on the left.

Example: Nameandsurname

- Iterative containers

  Whenever an iterative container is used, indicate the field name separated by a ".".

  In the wizard, they are identified with ● on the left.

  Example: Items.Quantity

- Form field relation

  The wizard allows selecting a field that has a relation to a value list, a rule, or an entity.

  In the wizard, they are identified with ● on the left.

  Example: stateRelation

- Possible values of related fields

  When modeling a related field, either to a value list, rule, or entity, the wizard allows to select the name of the field concatenated with ¨Relation", and the possible values of its relation.

  In the wizard, they are identified with ● on the left.

  Example: stateRelation.Active

- Properties of page and entity elements

  The wizard allows using the “value”, “visible” and “editable” properties of page and entity elements. These properties are available depending on the type of element being modeled.

  When the “value” property of an “input” element is used, the format of the returned value depends on the type of data entered. For the other elements, the returned value is of “string” type.

  When the "visible" and "editable" properties of all the elements of a page or entity are used, the returned value is of "boolean" type.

  Examples
  - Firstnameandlastname.value
  - Firstnameandlastname.visible
  - Firstnameandlastname.editable

In the wizard, they are identified with ● on the left.

- Variables and page parameters

The wizard allows using the variables and parameters of a page, and the value returned in the embedded rule corresponds to the type of each one of them.

In the wizard, they are identified with ● on the left.

- Entities and forms related to the process

  The wizard allows the use of all entities or forms that are related to the process. In this way, it is possile to define an expression that includes fields of different entities and forms.

  When using a field from an entity or form, it is necessary to first select the corresponding entity or form, and then choose the desired field.

  Example
  Within an activity, the fields "total" and "regime" from the "Invoice" entity are used. If the process is also related to the "PurchaseOrder" entity, the following embedded rules can be modeled, which use the fields: "amount" and "regime":
  - IF(total.value > PurchaseOrder.amount, "The total amount of the invoice cannot exceed that defined in the purchase order associated with the case")
  - IF(regimen.value != PurchaseOrder.regimen, "The tax regime on the invoice must be the same as that defined in the purchase order associated with the case")

  In the wizard, the fields related to entities are displayed ● on the left.

Functions

To the right of the name of the functions or rules, the data type of the value they return is displayed.

- Functions

  By using the editing wizard, the functions of the different categories defined in **Deyel** can be used.

- Advanced rules

  The advanced rules that can be used in embedded rules are those that return a single parameter. The wizard does not show rules that do not return a value or that return more than one parameter.

In both cases, in the wizard, they are identified with ● on the left.

Available Operators

Logical Operators

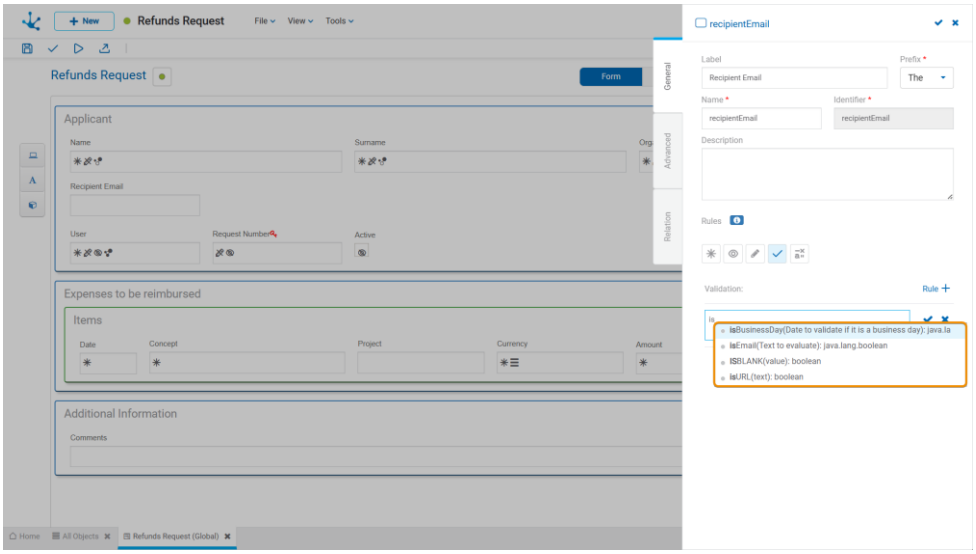
Operator	Description	Example
==	Compares if two operands are equal	5 == getDay()
!=	Compares if two operands are different	last name != "Jones"
<	Compares if the first operand is less than the second one	1 < 3
>	Compares if the first operand is greater than the second one	4 > 1
<=	Compares if the first operand is less or equal than the second one	1 <= 3
>=	Compares if the first operand is greater than or equal to the second one	3 >= 3

Arithmetic Operators

Operator	Description	Example
+	Adds two operands	5 + field3
-	Substracts two operands	price - discount
*	Multiplies two operands	price * 1.21
/	Divides the first operand by the second one	10 / 2
%	Calculates the division remainder of the first operand by the second one (mod)	5 % 3, the result is 2

3.4.12.1.1.2. Editing Wizard

A wizard to model the rules can be used in the rule editing area. This wizard is activated by pressing the "Ctrl + Space" keys. As text is entered in the wizard, it uses the auto-complete functionality and shows only the elements whose names begin with the text typed by the user.



3.4.12.1.1.3. Functions

A function is a software unit. It performs a specific task, receives input parameters, and returns a result.

They can be used:

From the function selection wizard in:

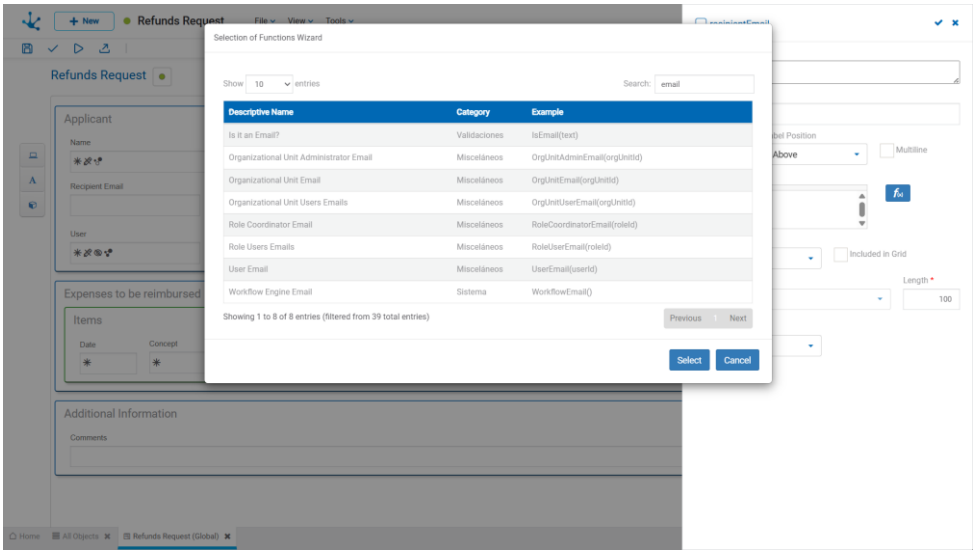
- Form fields, when defining the Default Value.
- The processes, by defining the Case description property and in alerts and automatic activity parameters.
- The process flows, when defining the Message.

When defining conditions in:

- Rules in form properties.
- Field Properties to verify conditions of edition, mandatory nature, visibility, validation and calculation rules.
- Page elements, to verify conditions of edition, mandatory nature, visibility, validation and calculation rules.
- Gateway properties when output flow conditions are evaluated.

Function Selection Wizard

The wizard is used from the icon f(x). When pressed, a window with the list of functions opens and once the function has been selected, it is included in the description area of the form or case.



Elements

- Selection of number of records to display in the window.
- Search field to be able to filter functions, filtering can be performed by the properties Descriptive Name, Category or Example.
- List of functions where properties are displayed in a grid mode.
- Buttons for paging.

Properties

Descriptive Name
It is used in the interface when it is referenced, in the functions list.

Category
Grouping by function type.

Example
Corresponds to how the function can be used in the calculation or condition areas.

Categories

Functions are grouped into categories.

- Execution
- System
- Date and time
- Math
- Validations
- Miscellaneous
- Exclusives for Rules

Execution

These functions retrieve information related to the form instance that is being used, to the process if it had one associated and from the connected user.

Descriptive Name	Description	Example
Current User Last Name	Returns the last name of the user that started the session.	currentUserLast-Name()
Current Activity Code	Returns the code of the activity in progress. It does not work in the first process activity because the case has not been created.	currentActivityCo-de()
Current Process Code	Returns the code of the process in progress. It does not work in the first process activity because the case has not been created.	processCode()
Current Activity Description	Returns the description of the activity in progress. It does not work in the first process activity because the case has not been created.	currentActivityDes-cription()
Current Process Description	Returns the description of the process in progress. It does not work in the first process activity because the case has not been created.	processDescription()
Email of the Executing Participant's Organizational Unit	Returns the organizational unit email of the participant responsible for executing the activity in progress, if it is a role, it returns empty. It does not work in the first process activity.	userOrgUnitEmail()
Email of the User Administrator of the Executing Participant's Organizational Unit	Returns the administration user email of the organizational unit of the participant responsible for executing the current activity. It does not work in the first process activity.	orgUnitAdmi-nEmail()
Email of the Role Coordinator Responsible for the Current Activity	Returns the email of the role coordinator user modeled as responsible for the activity in progress, only if the person responsible is a role, otherwise it returns empty. It does not work in the first process activity.	currentRoleCoordi-natorEmail()
Executing Participant Email	Returns the email of the participant responsible for executing the current activity It does not work in the first process activity. If the responsible participant is a role or an organizational unit, it returns the emails of all users belonging to the role or organizational unit.	currentUserEmail()

Descriptive Name	Description	Example
Current Activity Name	Returns the name of the activity in progress. It does not work in the first process activity because the case has not been created.	currentActivity-Name()
Name of the Current User's Organizational Unit	Returns the organizational unit name of the user who started the session.	currentUserOrgUni-tName()
Current Process Name	Returns the name of the process in progress. It does not work in the first process activity.	processName()
Current User Name	Returns the name of the user that started the session.	currentUserName()
Case ID	Returns the case id without applying the case display mask. It does not work in the first process activity becau-se the case has not been created.	caseNumber()
Executing Parti-cipant's Organi-zational Unit	Returns the organizational unit code of the participant responsible for executing the current activity. If the parti-cipant responsible is a role, it returns empty. It does not work in the first process activity.	currentOrgUnitCo-de()
Code of the Cu-rrent User's Or-ganizational Unit	Returns the organizational unit code of the user that started the session.	currentUserOr-gUnit()
Executing Parti-cipant	Returns the participant responsible for executing the current activity It does not work in the first process activi-ty. It can return a user, a role or a unit: - Returns a user when the participant responsible for the activity is a user, or when it is a role and the user is assig-ned the activity. - Returns a role only when the participant responsible is a role and the activity is generic, until one of the users is assigned the activity. - Returns an organizational unit when the participant responsible for the activity is a unit.	executingUser()
Executing User of the Current Pro-cess Activity	Returns the user that executed the activity informed by parameter. It requires the code of an activity already exe-cuted. It does not work in the first process activity.	activityExecutingU-ser(activityNo)

Descriptive Name	Description	Example
Current User	Returns the user that started the session.	currentUser()
Current Process Version	Returns the version of the process in progress. It does not work in the first process activity.	processVersion()
Last Button Pressed	Returns the last button pressed.	lastButtonPressed()

System

These functions retrieve information related to the environment where the form and the modeled application are being used, if there is one associated.

Descriptive Name	Description	Example
Web Server Address	Returns the web server address.	webServerAddress()
Workflow Engine Email	Returns the workflow engine email address.	workflowEmail()
Workflow Engine Name	Returns the workflow engine name.	workflowEngineName()

Date and Time

Retrieve information related to dates and times and carry out operations with them.

Descriptive Name	Description	Example
Current Year	Returns the current year.	year()
Difference Between Work Days	Returns the difference in business days between 2 past dates as parameters.	difBetweenWorkDays(dateFrom,dateTo)
Difference in Time	Returns the time difference between 2 times entered as parameters. The format must be HH:MM:SS.	difInTime(timeFrom,timeTo

Descriptive Name	Description	Example
		)
Difference in Minutes between Times	Returns the difference in minutes between 2 times entered as parameters. The format is HH:MM:SS.	difInMinutes(timeFrom,timeTo)
Difference between Days	Returns the difference in days between 2 dates entered as parameters. The format is DD/MM/YYYY.	diffBetweenDays(dateFrom, dateTo)
Current Day	Returns the current day.	day()
In between Dates	Validates that a date falls within a valid date range. The format is DD/MM/YYYY.	inBetweenDates(Datep,StartDatep,pEndDate)
Current Date	Returns the server current date.	Date()
Local Date	Returns the local date according to the user's time and time zone.	localDate()
Current Time	Returns the server current time.	time()
Local time	Returns the local time according to the user's time zone.	localTime()
Current Month	Returns the current month.	month()
Month Name	Returns the name of the month according to the number entered.	monthName(monthnumber)
Add days	Returns the date obtained as a result of adding days to a date.	addDays(StartDatep,Daysp)
Add work days	Returns the date obtained as a result of adding days to a date. Adds only work days.	addWorkDays(pDate,pDays)
Last business day of the month	Returns the last business day of the month of the date entered as a parameter.	lastWorkDay(pDate)
Subtract days	Returns the date obtained as a result of subtracting days to a date.	subtractDays(pDate, pDays)

Descriptive Name	Description	Example
Current time and date	Returns the current time and date of the server.	dateAndTime()

Math

These functions perform mathematical operations based on the fields of entities, pages, and forms.

Descriptive Name	Description	Example
Sum	Returns the result of the sum of all the values of a field within a container of multiple occurrences or a repeating group, taken as a parameter. The field data type must be integer or decimal.	sum(field)

Validations

The functions in this category allow to confirm a question about the information used in the entity, page or form.

Descriptive Name	Description	Example
Is it an Email?	Verifies that the text entered as a parameter has an email format.	isEmail(text)
Only Letters?	Verifies that the value entered as a parameter consists only of letters.	onlyLetters(Text)
Only Numbers?	Verifies that the value entered as a parameter is a valid number.	onlyNumbers(Text)
Is it a Business Day?	Verifies that the value entered as a parameter is a business day.	isBusinessDay(pDate)

Miscellaneous

This category groups functions that return information on different topics.

Descriptive Name	Description	Example
Organizational Unit Email	Returns the email address of an organizational unit whose code is entered as a parameter.	orgUnitEmail(orgUnitId)
Organizational Unit Users Emails	Returns the email address of an organizational unit users whose code is entered as a parameter.	orgUnitUserEmail(orgUnitId)
Organizational Unit Administrator Email	Returns the email address of an organizational unit administrator whose code is entered as a parameter.	orgUnitAdminEmail(orgUnitId)
Role Coordinator Email	Returns the email address of a role coordinator whose code is entered as a parameter.	roleCoordinatorEmail(roleid)
Email of the Activity Responsible Participant	Returns the email address of the participant responsible for executing an activity whose code is entered as a parameter. If the participant is a role or an organizational unit, it returns the emails of all its members.	activityResponsibleEmail(activityNo)
User Email	Returns the email address of a user whose code is entered as a parameter.	userEmail(userid)
Role Users Emails	Returns the email addresses of an organizational unit users whose code is entered as a parameter.	roleUserEmail(roleid)
Case Show Link	Returns the link to the case show in progress. Requires the user to be logged in.	caseShowLink()
Activity Execution Link	Returns the link to the execution of the activity in progress, requires the user to be connected.	activityExecLink()
My Tasks Link	Returns the link to the user's task list.	myTasksLink()
Login Link	Returns the link to the user portal home panel.	loginLink()

Exclusives for Rules

Functions in this category are used in embedded rules modeling by means of their wizard.

Descriptive Name	Description	Example
IF(logical_test,value_if_true,value_if_false)	Checks if the condition indicated in "logic_test" is met. If it is true, it returns the value informed in "value_if_true", if not, it returns the value informed	IF(amount > 1000000, "Amount exceeds limit")

Descriptive Name	Description	Example
	in "value_if_false". The "value_if_false" parameter is optional, if the condition is not met and such parameter is not defined, the function does not return a value.	If the amount is greater than 1,000,000, the user receives the "Amount exceeds limit" message.
AND(logical_value1, logical_value2,...)	Evaluates that all parameters return "True", in which case it returns "True" value.	AND(day() > 5, month() == "April") Returns "True" only when the day of the month is greater than 5 and the month is April.
OR(logical_value1, logical_value2,...)	Evaluates all parameters and returns "False" value if all parameters return "False". If at least one of them returns "True", it returns "True" value.	OR(contact == "AFA-RIAS", currentUser() == "AFARIAS") Returns "False" only when the current user and contact are diffe-rent from AFARIAS.
NO(logical_value)	IF"logical_value" is "True", it returns "False" value. IF"logical_value" is "False", it returns "True" value.	NO(5 > 1) Returns "False", since it is the logical value con-trary to evaluation 5>1.
ADD(number1, number2,...)	Sum the individual values of each of the specified parameters.	ADD(AmountInvoiced, TaxAmount) Returns the value resul-ting from adding both amounts.
CON-CAT(text1,text2,...)	Joins multiple text elements into one.	CONCAT("Hello,", cu-rrentUser()) Returns "Hello AFA-RIAS", where AFARIAS is the current user.
MAX(number1, number2,...)	Returns the maximum value from a list of parame-ters. Ignores logical values and text.	MAX(2, 5, customer-Number, 22) Returns 22.
ISBLANK(value)	Checks the value of the reported parameter and returns "True" if it has no content. If the parameter is an iterative field, all its occurrences must be em-pty for it to return true.	ISBLANK(description) Returns "False", when description has content.
isURL(text)	Indicates whether the text entered corresponds to a	IF(NO(isUrl(field)),

Descriptive Name	Description	Example
	valid URL format. Valid Examples: • www.sitename.com ; sitename.com; https://sitename.com • 192.1.1.111; 192.1.1.111:8090 ; 192.1.1.111:8090/myfunction	"Must inform URL") If field is equal to "De-yel.com", the user does not receive any mes-sages.
REPLACE(text to search; text to re-place; original text)	Replaces in "original text", the "text to search" mat-ches with "text to replace".	REPLACE("Supplier XX", "New Supplier", contractText) As a result, the original content of the contrac-tText field remains with all occurrences of Supplier XX replaced with New Supplier.
REGEXMATCH(text, regular_expression)	Returns "True" value if the first parameter meets the regular expression indicated in the second parame-ter. Regular expressions, also known as regex, are widely used, standardized writing patterns used mainly to process texts. They allow, for example, to validate formats, extract a part of text or replace occurrences of a character string. There are many free access sites that provide information on use and syntax and offer the possibility to perform si-mulations as tests. If the '\' character is used in the expression, it should be duplicated. If it is not duplicated, the expression is not evaluated as expected.	REGEX-MATCH(myField,"^[0-9]*$") Returns "True" if myField is made up of numeric characters between 0 and 9. REGEX-MATCH(myField,"^[AZ]{3}\\d{6}") The bar in the expres-sion is duplicated.
equalInAllVa-lues(iterative_field, value)	Returns "True" value if all occurrences of "iterati-ve_field" match with "value".	equalInAllVa-lues(customerNo, 136) Returns "True" if all occurrences of the Cus-tomerNo iterative field has a 136 value.
equalInSomeVa-lue(iterative_field, value)	Returns "True" value if any of the occurrences of "iterative_field" matches with "value".	equalInSomeVa-lue(customerNo, 136) Returns "True" if one or more occurrences of CustomerNo iterative field has a 136 value.
FIRSTVALUE(list)	Returns the first value in a list.	FIRSTVA-

Descriptive Name	Description	Example
		LUE(Items.quantity) Returns 5, the first value of the iterative quantity [5, 3, 2, ...].
QUANTITY(list)	Returns the length of a list.	QUANTITY(Items.price) Returns 3, the iterative price being [100,50,30].
EXTRACT(text, position, amount_characters)	Returns a subset of text characters, given an initial position and length.	EXTRACT("The user was successfully deleted", 13, 12) Returns the characters "successfully".
NUMBER(alphanumeric_value)	Transforms an alphanumeric into a number.	NUMBER("1988") Returns number 1988.
ROUND(numeric_value, decimals)	Rounds a number to a specified number of decimal places. If the digit after the specified position is greater than or equal to 5, the digit is rounded up. Otherwise, it is rounded down. This happens regardless of the sign. It is recommended to use this function when using 4- and 5-place decimals.	ROUND(49.9999, 0) Returns number 50.
extension(file)	Returns the extension of a file type field.	extension(contract) If the contract field is equal to "contract_2022.pdf", it returns "pdf".
length(text)	Returns the number of characters entered in an alphanumeric field or the number of digits in a numeric field.	length(password) If the password field is equal to "deyel123", it returns number 8.
REMOVESPACES(text)	Removes spaces at the beginning and end of a field.	REMOVESPACES(email) If the email field is equal to " afarias.fa@gmail.com ", returns "afarias.fa@gmail.com".
ESCAPEHTML(text)	Removes formatting from the content of a rich type text field.	escapeHTML(notes) Returns content of "notes" field without for-

Descriptive Name	Description	Example
		matting.
mathPow(base, exponent)	Returns the base raised to the indicated exponent.	mathPow(2, 3) Returns number 8.
parseDate(text)	Returns the value in date format equivalent to the one entered as a literal.	parseDate("07/25/2023") Returns the value corresponding to 07/25/2023
parseDateTime(text)	Returns the value in date and time format equivalent to the one entered as a literal.	parseDateTime("07/25/2023 15:07") Returns the value corresponding to 07/25/2023 15:07
parseTime(text)	Returns the value in time format equivalent to the one entered as a literal.	parseTime("15:20") Returns the value corresponding to 15:20

> *Functions are allowed to be nested in parameters.*

3.4.12.1.2. Advanced Rules and Integration

Advanced rules can be invoked from entities, pages, processes, forms and scheduled tasks, to incorporate business logic.

In entities, pages, forms and processes they can be included using the rules wizard, whereas in scheduled tasks the reference to the rule is done manually.

Types

Standard Rules

These are rules developed in the Java programming language that use the Deyel SDK adapter to develop logic without accessing components external to **Deyel**.

> *Standard rules developed in earlier versions of **Deyel** can continue to use the "Standard Rule" adapter'', which is no longer available.*

Integration Rules

These are rules that make use of adapters that provide integration with components external to **Deyel**.

3.4.12.1.2.1. Modeling Facilities

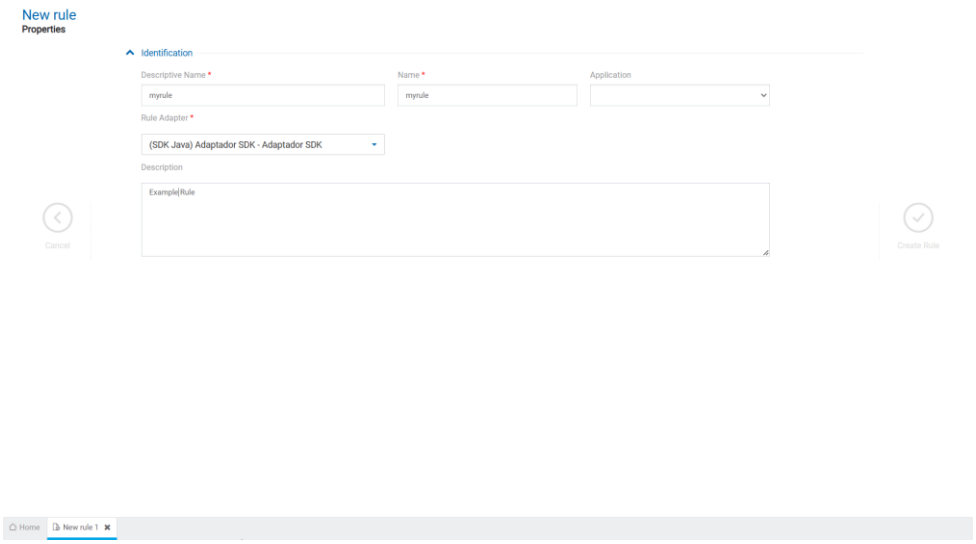
Phase 6: Advanced Rules Modeling

The advanced rules modeler is a tool that allows to easily design business rules, allowing the addition of the necessary logic to make them work when they are executed.

Furthermore, it is possible to use Diana, the AI-powered conversational assistant of **Deyel** that guides the modeler user in creating rules using natural language, interpreting the request and generating the necessary code.

New Rule

The modeler user can define new rules, which after being published are available to be used in the portal.



Steps to Create a Rule

Step 1: Open the Rules Wizard



This button is used to create an advanced rule from the option  Rule.

Step 2: Enter the Rule Properties

Properties are organized under the container Identification, which contains information common to all the rules.

Descriptive name
Name used by the modeler to refer to the rule, display it in the rules gallery and in the object tree.

Name

Used internally to refer to the rule within entities, forms, embedded rules, processes or as parameters of other advanced rules. Unlike other objects, the rule name also acts as a unique identifier.

Application
Allows to define the application to which the object belongs. If no application is reported, the object is assigned to the application "Global".

Adapter
Used to determine the adapter to be used by the rule. A list of adapters available for selection is displayed.

Operation
Used to choose the operation to be executed and is only displayed when the chosen adapter type is web services.

Description
Text that defines the rule describing its functionality and optionally its content.

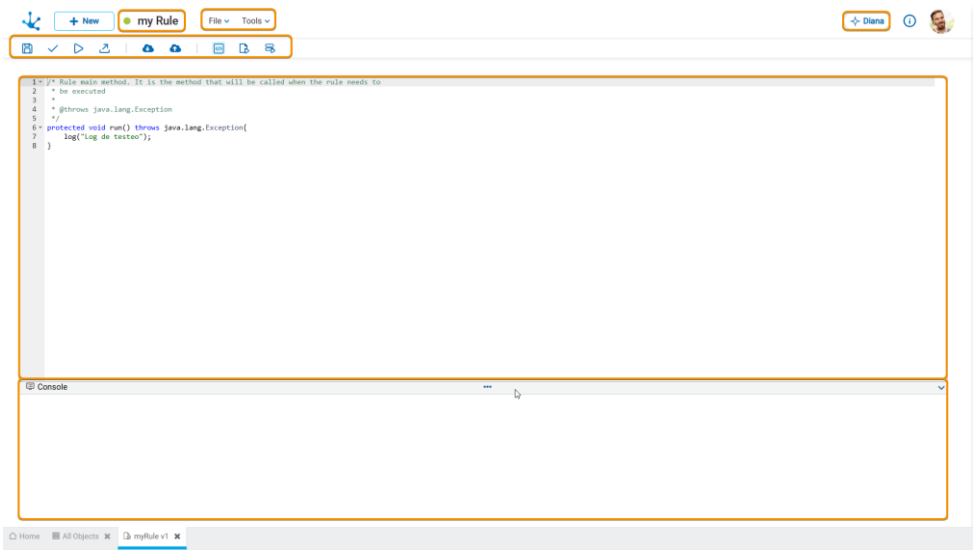
Step 3: Close the Rules Wizard

Cancel
Allows going back to the Deyel modeler and display the objects in the gallery.

Create Rule
Moves to the rule workspace.

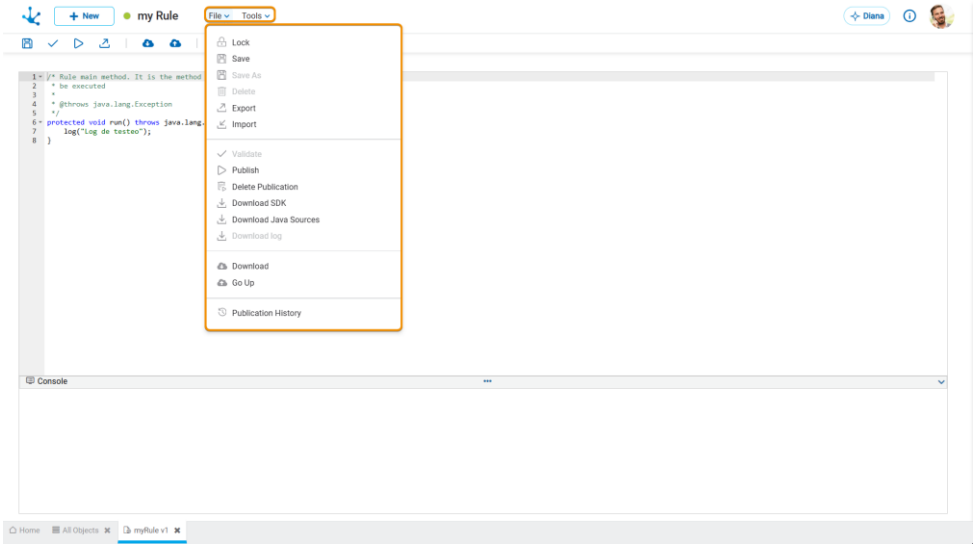
Workspace Sections

- Rule Information
  - State
  - Name
  - Blocking
- Expanded Menu
- Diana
- Top Toolbar
- Modeling Area
- Console



Expanded Menu

It is a horizontal list of options containing vertical submenus with different operations on the rule or on its modeling. In turn, each option in a submenu may expand a dependent submenu.

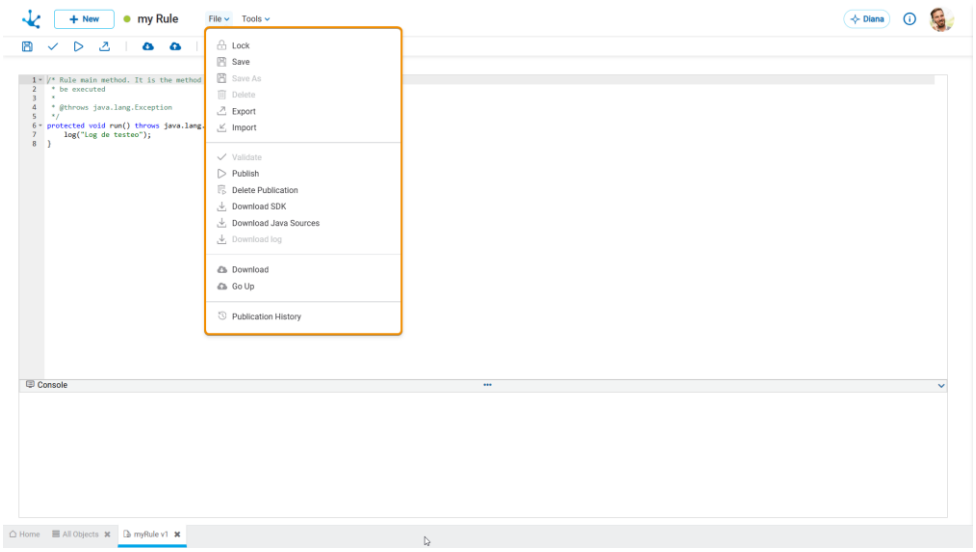


The expanded menu consists of:

- File Submenu
- Tools Submenu

File Submenu

This submenu opens by clicking on the “File” option and allows the performance of operations on the rule.



/ Lock/Unlock

- It allows blocking a rule to ensure that no one can modify it until the person who is using it unlocks it, that is, releases it.
- It allows unlocking a rule so that another user can modify it.

Save

This icon allows to save the object in the repository of **Deyel**, leaving its state as "Draft" or "Modified". If certain conditions are met, the modeler user receives a message indicating that the operation was performed correctly, otherwise they receive an explanatory message.

Conditions

- The object should not be locked by another user,
- There must be an adapter that uses the rule.

Save As

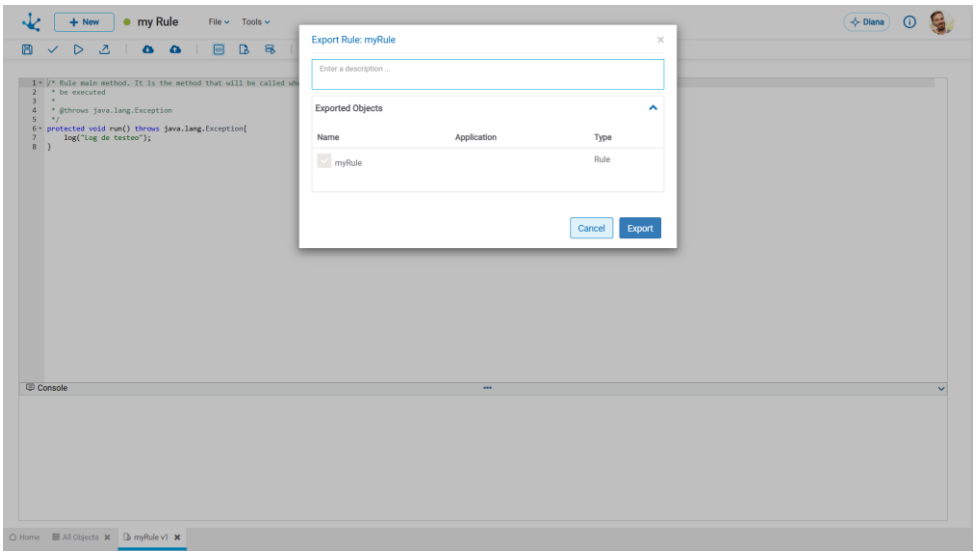
It allows the creation of a copy of the modeled rule, defining a new descriptive name, name, application, and description. That copy opens in a new tab in [state](#) “Draft”.

Delete

It allows to delete the object only if it is in "Draft" state and does not have other associated objects of **Deyel** that were previously saved or published. When deleted, the tab it is on closes and the object is removed from the modeler's grid.

Export

This icon opens a window for the user to select and confirm the export of the object.



Description

In this property a text explaining the reason for the operation can be entered.
This text can be modified upon import and is displayed in the description column of the export record.

Exported Objects

By expanding the container, the object being exported is shown.

Press the "Cancel" button to undo export or press the "Export" button to finish.

Validate

This icon allows to validate if the object is ready to be published, that is, the same validations are carried out as when publishing and the result of them is reported.

Publish

Through this icon the object changes to "Published" state and the modeler user receives the corresponding message, indicating the result of the operation. The condition for publishing is the same as for the "Save" operation.

Conditions

- The packages corresponding to Java objects included in the rule code must be known to **Deyel**.
- The objects related in the rule must be previously published.

Delete Publication

It allows the advanced rule to be disabled from use, returning it to the state “Draft”.

Download SDK

This icon allows the download of the JAR file that represents Deyel’s SDK.

Download Java sources

This icon allows to download the Java files that represent the object's model and service, so that it can be used in advanced rules.
Pressing the icon displays a message to confirm file download.

Download logs

It allows downloading all the logs generated for the rule.

Download

It allows downloading the Java rule and the model and service classes of the objects related to the object wizard. A compressed file (.zip) containing the classes to edit later in an IDE for Java is generated. This icon is displayed for rules with all types of adapters with the exception of SOAP rules

Upload

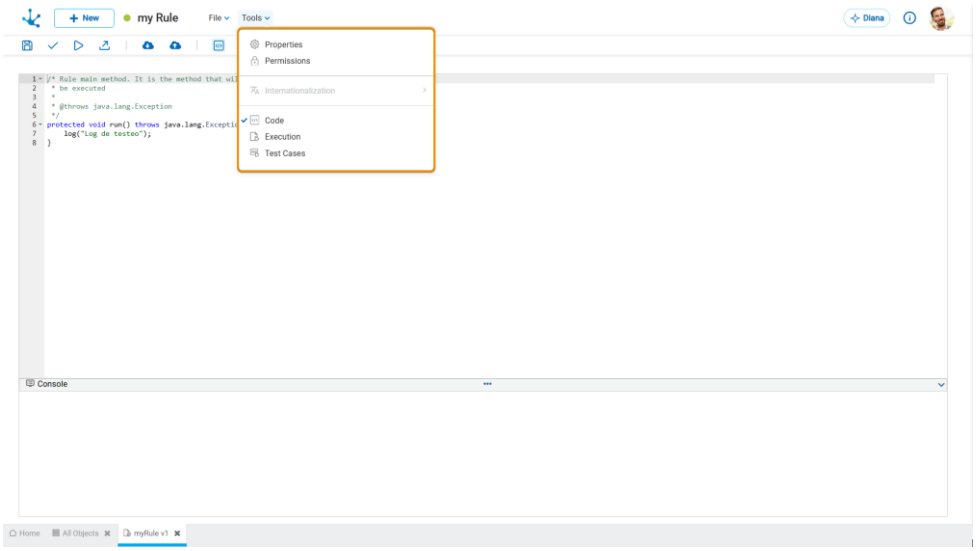
It allows uploading a source file to update the Java rule. Such an operation updates the window displaying the imported source code in the Java code editing area, but it is not saved until the user saves or publishes the rule. The code must be compilable; otherwise, the load operation will result in a compilation error. This icon is displayed for rules with all types of adapters except for SOAP rules

Publication History

This icon allows managing versions of published objects, displaying their history, to have better control of the changes that each object has.

Tools Submenu

This submenu opens by clicking on the "Tools" option and it allows modeling advanced rule properties and their permissions, as well as internationalization and execution options selection.



Properties

Opens the panel of rule properties.

Permissions

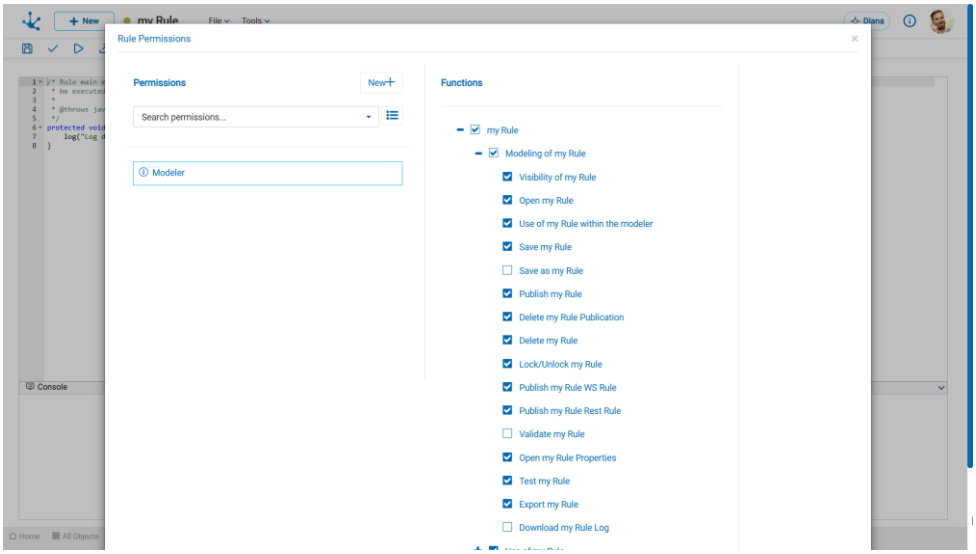
Allows to assign the security functions for use and modeling of the object to the existing permissions or to new permissions that the user creates, without having to go to the permission settings option from the menu.

Sections

- Permissions: Permissions to which object functions are assigned.
- Security Functions: Represents the total set of security functions, modeling and use of the object. Those that are marked are the ones included in the selected permission.

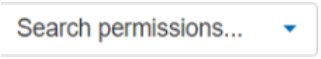
By default, all security functions for a new object are assigned to the permission Modeler.

Users who are assigned the permissions have access to the functions included in it.





Opens a panel to create a new permission and once created, the security functions included in it must be selected .



Allows to select a permission from a list and enables the input of characters to filter the values in the list.


Opens the wizard to select a permission and once chosen, the necessary object security functions must be checked.

To unrelate a permission from the object, hover the cursor over the permission and press the icon . If there are functions selected for that permission, they must be unchecked in order to delete it.

Modeling Security Functions

- Visibility: Allows to display the object in the Deyel modeler grid.
- Open: Allows to open the object from the Deyel modeler.
- Use within the modeler: It allows the use of the object from another modelable object.
- Save: Enables the save, download rule and upload rule operations.
- Publish: Enables the operation of publishing the object leaving its state as "Published".
- Delete publication: Enables the operation of deleting the object publication leaving its state as "Draft".
- Delete: Enables the delete rule operation.
- Lock/unlock: Enables the lock/unlock operation, only the user who locks it can modify it.
- Publish WS rule: Enables the publish rule operation as SOAP Web Services.
- Publish Rest rule: Enables the operation to publish the Rest API type rule.
- Open properties: It allows to show the rule properties panel.
- Test: Enables the operation to test the rule execution from the "Execution" option.
- Export: Enables the operation to export the rule definition.

- Download log: Enables the operation to download the rule log.

Use Security Functions

- Execute rule via Rest: It allows the smart thing user ("Rest API type Client") to execute the endpoints of the rules Rest API.

Menu Options

When creating or modifying an advanced rule, it allows writing Java code or SQL code as applicable, testing the rule, and reviewing its compilation errors, displaying them in the console.

Depending on the type of advanced rule, different menu options are displayed.

All rule types, except for the form extension rules, have the "Test Cases" option.

- Standard Rule
  The available menu options are: "Code", "Execution" and "Test Cases".
  .
- JDBC Rule
  The available menu options are: "Code", "Pre/Post", "Execution" and "Test Cases".

- SOAP Rule
  The available menu options are: "Execution" and "Test Cases".

- Forms Extension Rule
  The available menu option is: "Code".

- Rest Rule
  The available menu options are: "Pre/Post", "Execution" and "Test Cases".

Code

The code area allows displaying and editing the rule code, which can be Java code or SQL code.

Code generation can also be assisted by the rules agent of **Diana**, who interprets the instructions of the modeler user expressed in natural language and automatically generates the corresponding logic in this panel.

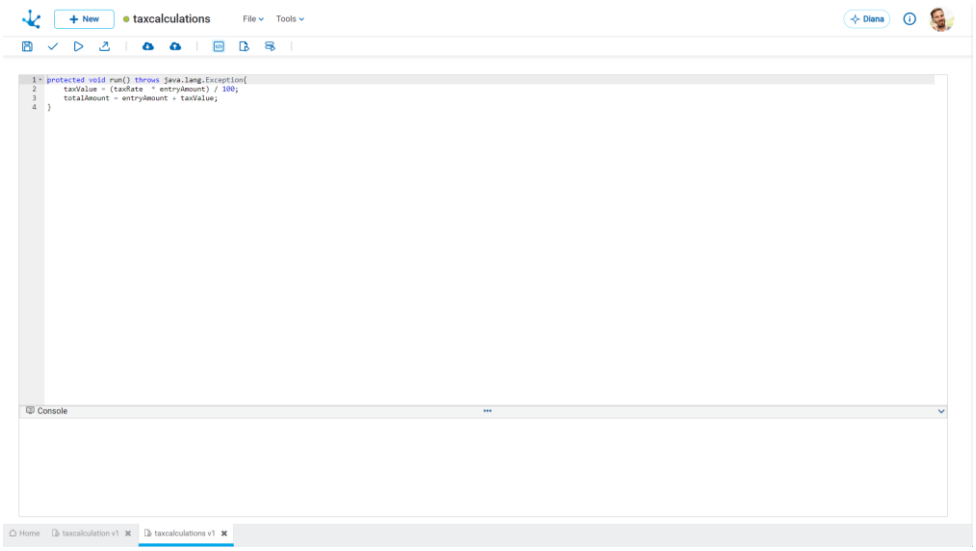
In this way, the modeler user can alternate between manual code editing and AI-assisted creation, depending on their experience level and development needs.

Java Code

The Java code area is displayed when the modeler is started for a standard type rule that uses an "SDK Rule" type adapter.

> *Standard rules developed in earlier versions of **Deyel** can continue to use the "Standard Rule" adapter'', which is no longer available.*

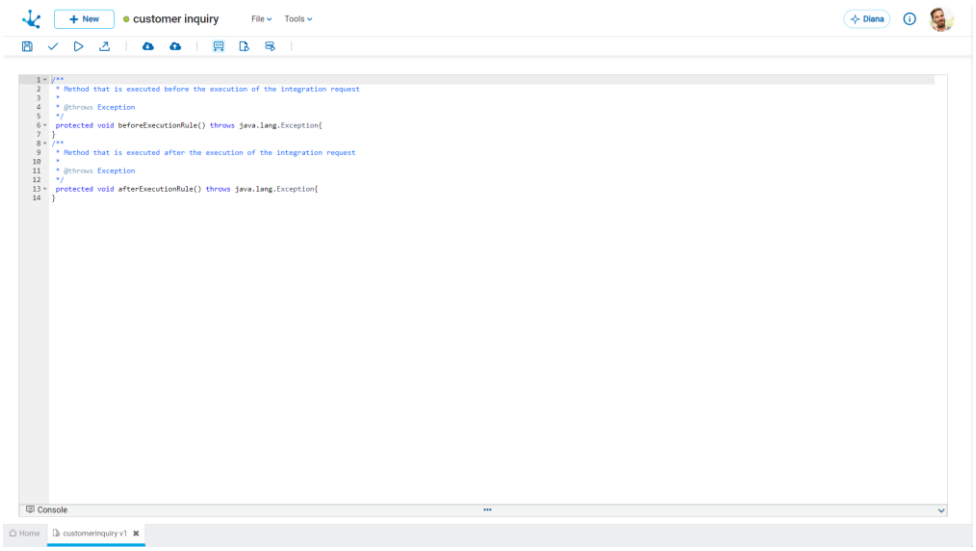
The Java code that executes the rule is written in this area and Deyel SDK for Java must be used for its development.



SQL code

The SQL code area is displayed when the modeler is started for a rule that uses a "JDBC" adapter.

In this area, the SQL show that executes the rule is written.



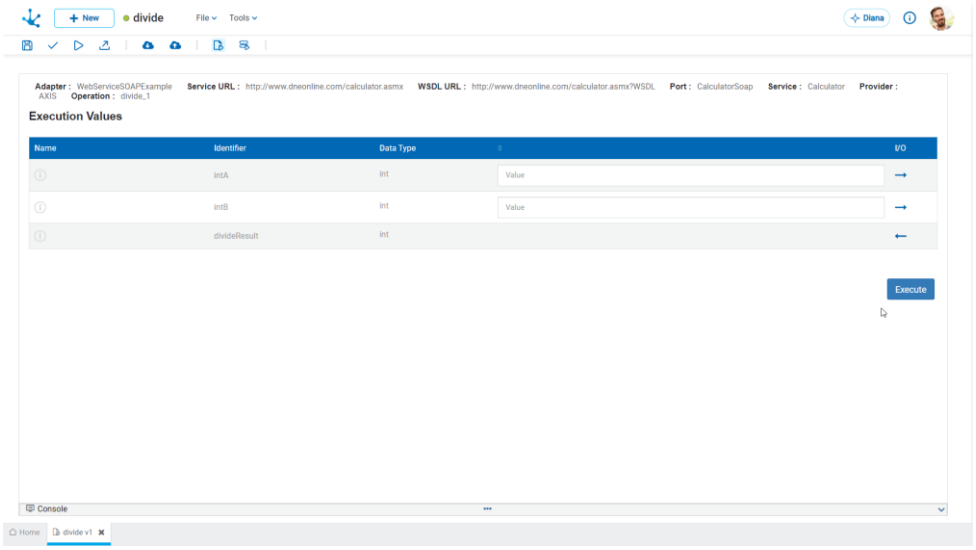
Execution

The execution of a rule allows verifying its operation within the context of the rules modeler and evaluating whether the defined logic meets the expected behavior. The execution may involve internal operations within the application or interactions with external applications when it comes to integration rules.

This area is displayed for all types of rules except those for extending form functionality.

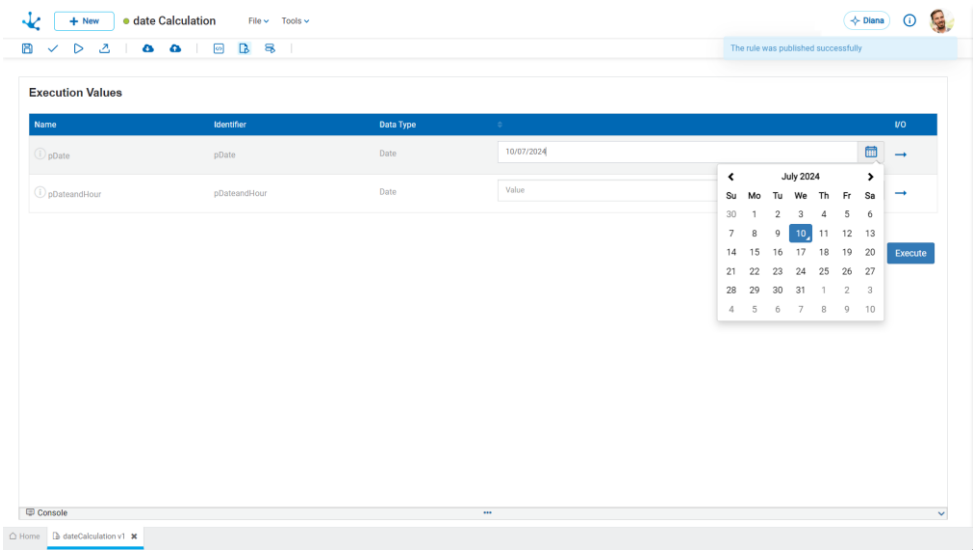
For rules that use a SOAP adapter, this area is accessed by default when entering the modeler. For JDBC and SOAP web services rules, the adapter used is specified.

Test values can be entered into the input parameters defined for the rule.



For each input parameter, and depending on its type, hovering over each field displays a help text that indicates the format in which the value should be entered.

Additionally, for "Date" and "Timestamp" parameters, a date or date and time selection panel is displayed, as appropriate. The selected date is displayed respecting the mask that has been configured in the environment.



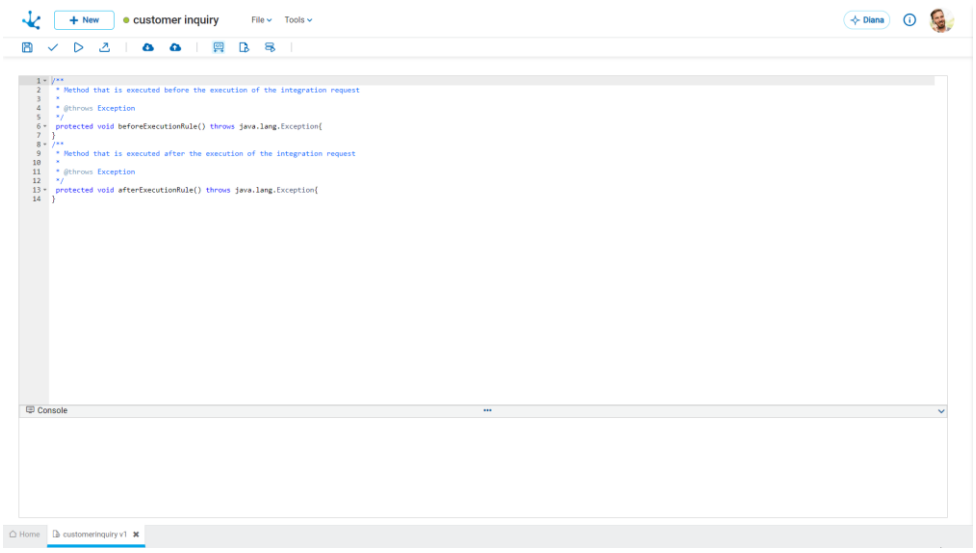
Pre/Post

When starting the modeler for a rule that uses the "Rest" type adapter, the area displayed is Pre/Post WS, and for a "JDBC" type rule is Pre/Post SQL.

Two required initial methods must be coded:

- **protected void beforeExecutionRule() throws java.lang.Exception{}**
  It allows adding logic before the ApiRest service call or the JDBC invocation.
  Example: An object can be read using the Deyel SDK and the result used as an input parameter in the JDBC invocation.

- **protected void afterExecutionRule() throws java.lang.Exception{}**
  It allows adding logic after the ApiRest service call or the JDBC invocation.
  Example: The result of the JDBC invocation can be received, processed, and saved in a form.



Additionally, in the “Rest” type rule, the following methods can be invoked:

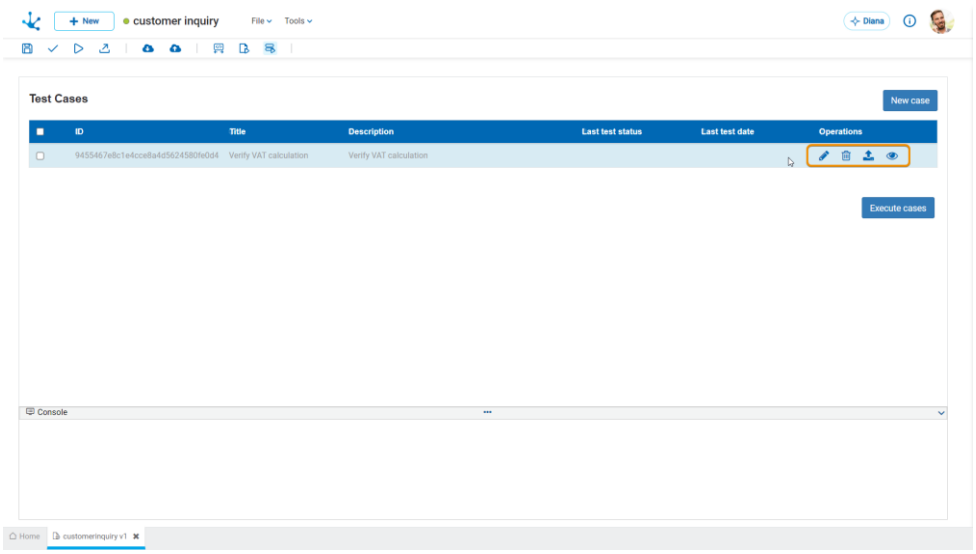
Before Execution (beforeExecutionRule)	After Execution (afterExecutionRule)
setBody(String body): Allows defining the body parameter of the request.	String getBodyResponse(): Allows to get the response body.
addHeader(String key, String value): Allows adding a heading to the request.	Integer getHTTPStatus():Allows to get the HTTP state of the response.
addQueryParam(String key, String value): Allows adding a parameter show to the request.	List getHeadersResponse (); It allows obtaining a header list of the HTTP request executed. This list consists of objects of org.apache.http. header type.
addValueToURL(String key, String value): Replaces the text specified in the first parameter with the value of the second parameter in the url of the service to be called.	

Test Cases

Test cases allow defining controlled scenarios to validate the behavior of a rule before its final implementation. A test case can include input data, the context in which it is executed, and the expected result, ensuring that the rule responds correctly in different situations.
With the assistance of the rules agent of **Diana**, the user can create test cases in natural language, describing the situation they wish to evaluate and allowing the wizard to automatically generate the corresponding configuration. This accelerates the definition of validation scenarios and facilitates continuous rule testing for both technical and functional users.

This area is displayed for all types of rules except for the form functionality extension rules.

In a test case, it is possible to define a title with a description and store values for all the parameters defined in the rule, to later test the rule with those values.



By selecting the mark field located in the header line, all existing test cases are selected to subsequently execute them. However, if only some of them are to be executed, each of the chosen lines should be selected with a checkmark.

New case Opens a panel to create a new test case where the title and description are required fields. Additionally, values can be entered for all the parameters defined in the rule.

> *Unlike the Execution option, output parameters can be assigned values, which are considered expected values upon completion of the test case execution. For instance, if the expected value 'true' is defined for the boolean-type parameter "pOutput", and at the end of the rule execution, the actual value assigned to this parameter is "false", the state of the test case execution will be marked as "Failed"".*

Execute cases Performs the execution of previously selected cases.

Available Operations

By hovering the mouse pointer over each line, at the level of the "Operations "column, the following options are enabled:

Opens a panel to edit the data of the selected test case.

Deletes a test case.

In the Execution option, the input parameters are initialized with the corresponding values stored in the test case.

Opens a panel to show data of the selected test case.

Execution States

A test case has two possible execution states:

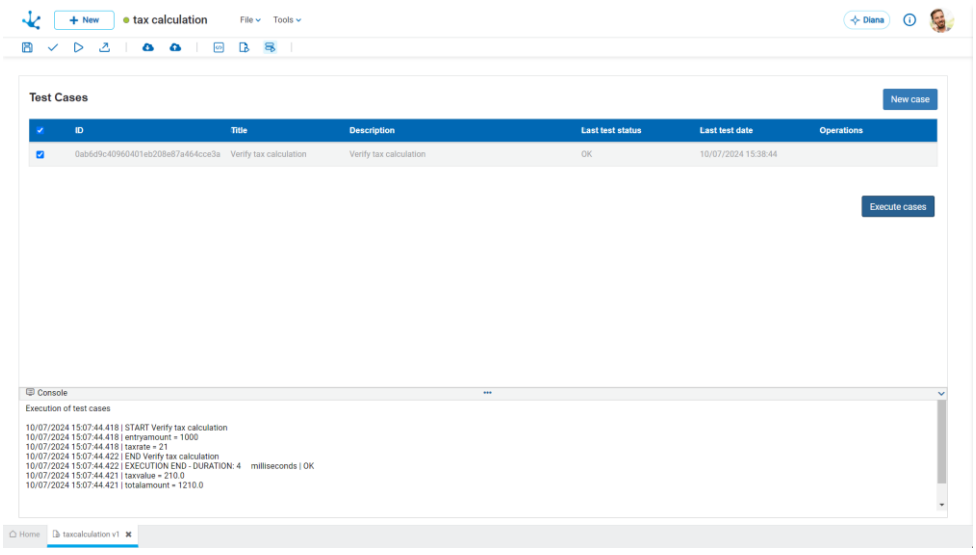
- Ok
  A case is considered successful if, at the end of its execution, there are no errors in the rule. If output parameters with an expected value were defined, they should match the value obtained.

- Failed
  A case is considered failed if, at the end of its execution, there was an error in the rule code or if any output parameter with an expected value differs from the value obtained.

Console

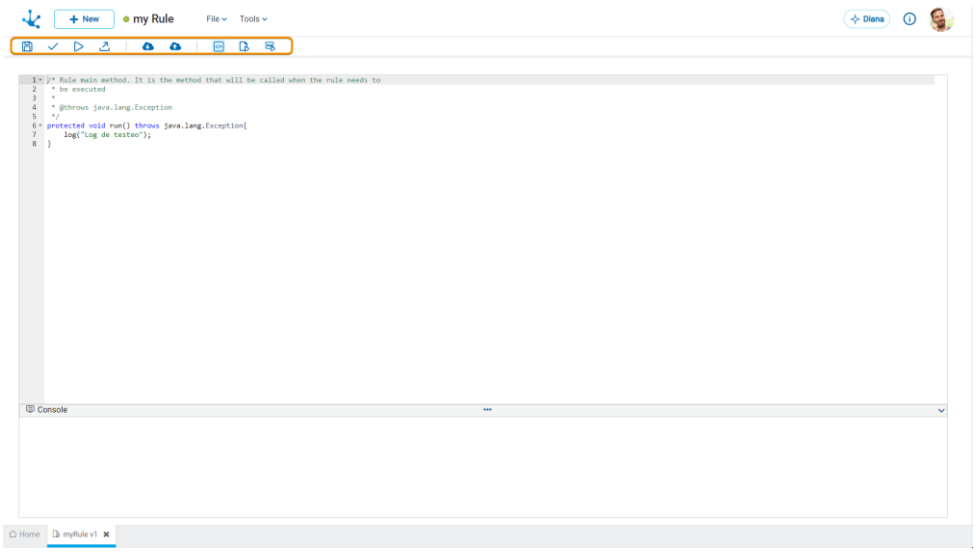
At the end of the execution of a test case, the console is displayed as a bar at the bottom border of the test case area. In the center of the bar, there is an icon ••• that allows to modify the size of the console area, and, on the far right there is an icon ⌄ that allows to close it.
The following information is displayed:

- Date and time of the case execution start.
- Values entered for input parameters.
- Values obtained for output parameters.
- Date and time of the case execution completion and duration.
- Execution state.



Top Toolbar

Contains icons for quick access to the most used operations of the expanded menu.



File

- Save
- Validate
- Publish
- Export

- Download
- Upload

Tools

- Code
- Execution
- Test Cases

Modeling Area

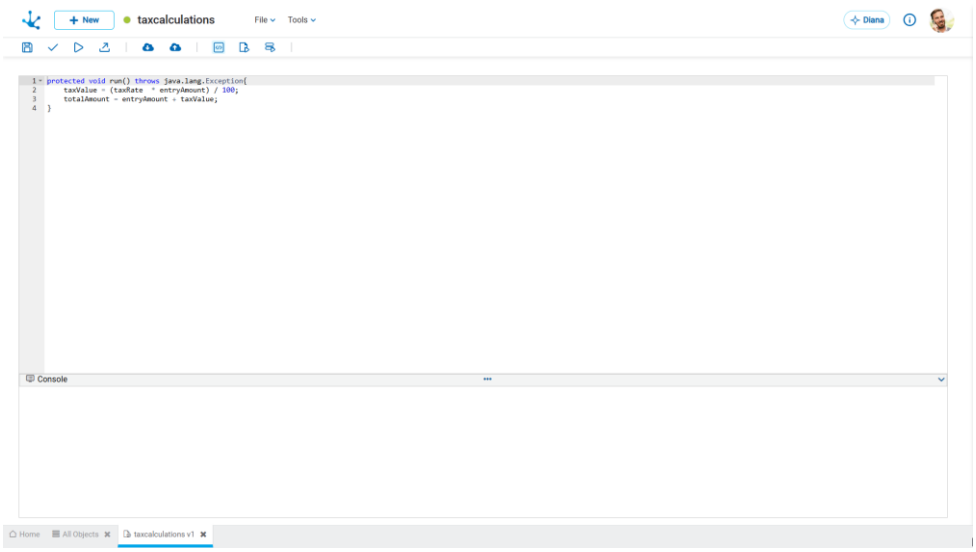
The modeling area is the space where the rules are built, including the steps and operations that define their behavior. The available options depend on the type of rule and adapter selected.

The modeling area can also be complemented with the assistance of the rules agent of **Diana**, the AI-powered conversational assistant of **Deyel**, which can propose the rule code from instructions expressed in natural language, allowing alternation between traditional visual modeling and AI-assisted interaction, depending on the needs of the modeler user.

Standard Rule

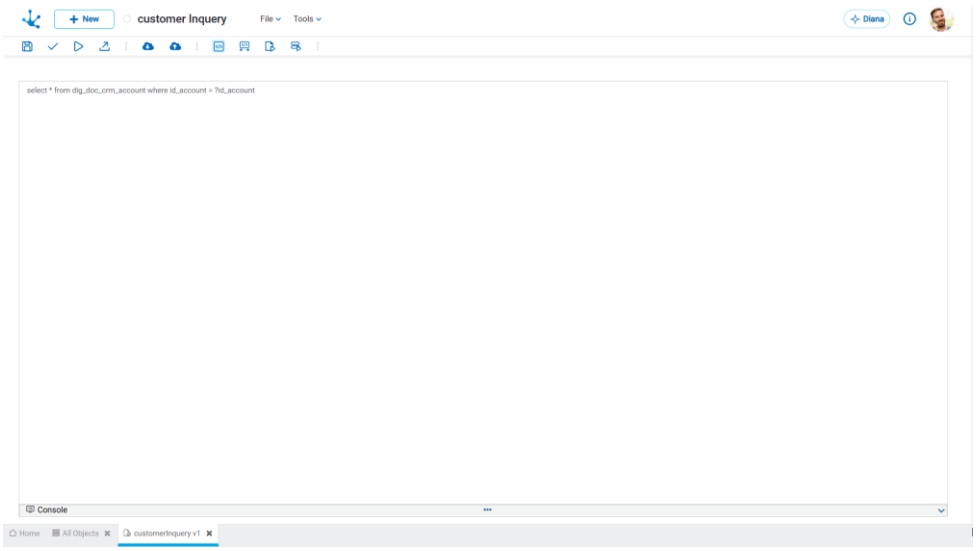
Standard type rules use Java code to implement business logic. They use “SDK Rule” type adapters.

> *Standard rules developed in earlier versions of **Deyel** can continue to use the “Standard Rule” adapter'', which is no longer available.*



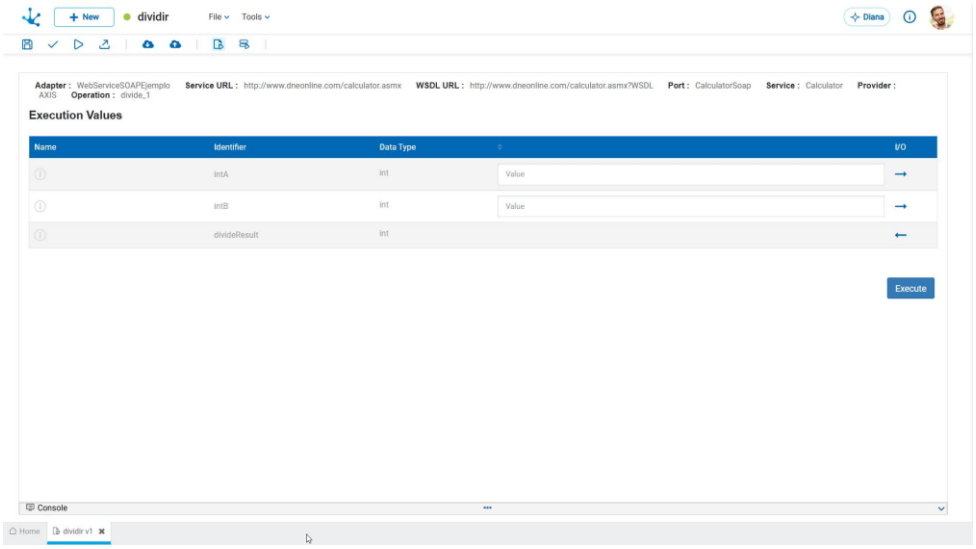
JDBC Rule

JDBC type rules are those used to perform operations related to databases, using a previously defined "JDBC" adapter. From database shows to the execution of store procedures can be performed. These rules use the adapters available for database connection.



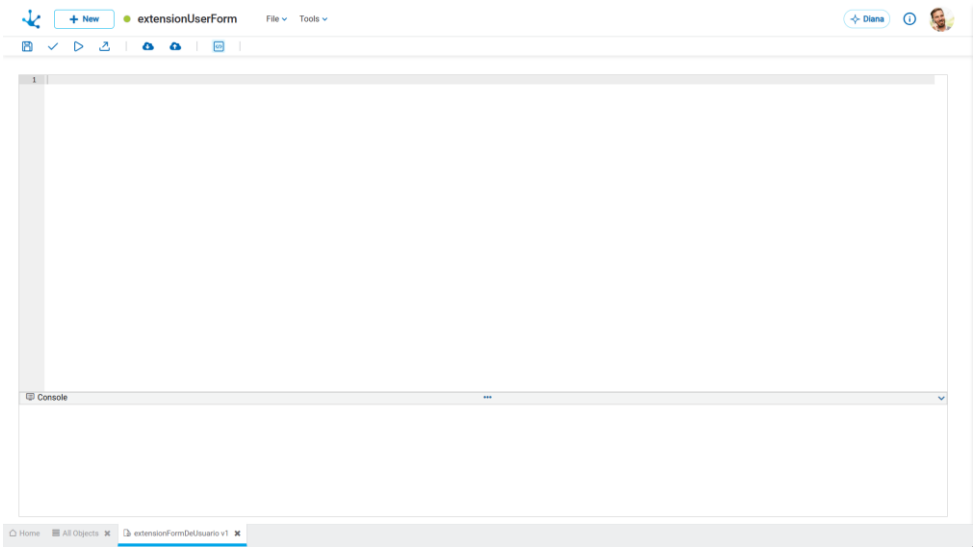
SOAP Rule

SOAP web service type rules are used to use external SOAP services in order to achieve integration with them. They use SOAP Rule" type adapters.



Forms Extension Rule

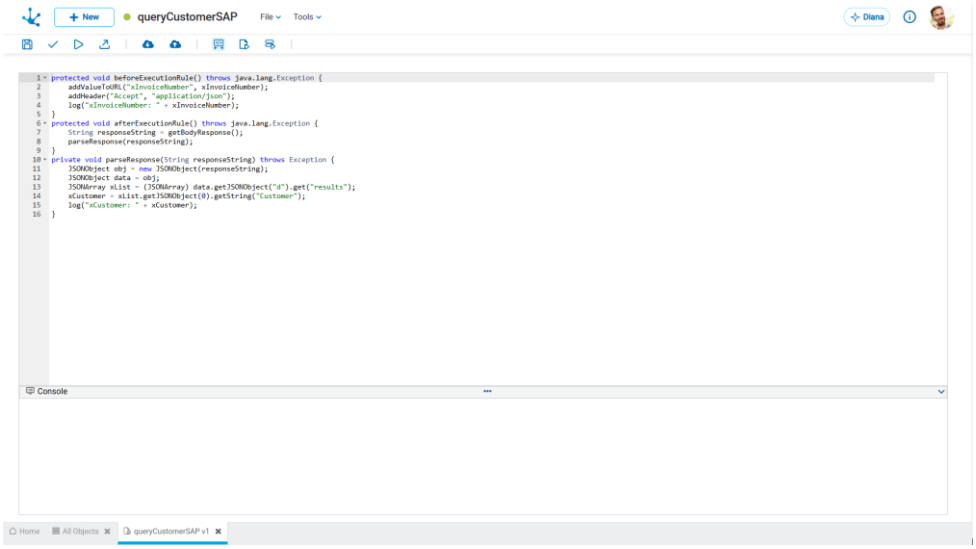
Forms extension type rules allow to use logic that cannot be done in the form modeler. They use the "Forms Extension" adapter.



Rest rule

REST type rules are used to invoke ApiRest services external to **Deyel**.

They use Java code in their Pre/Post section to execute business logic before and after the ApiRest service call. They use adapters of "API Rest" type or application adapters as "SAP", "Mercado Libre", "Google Drive", etc.

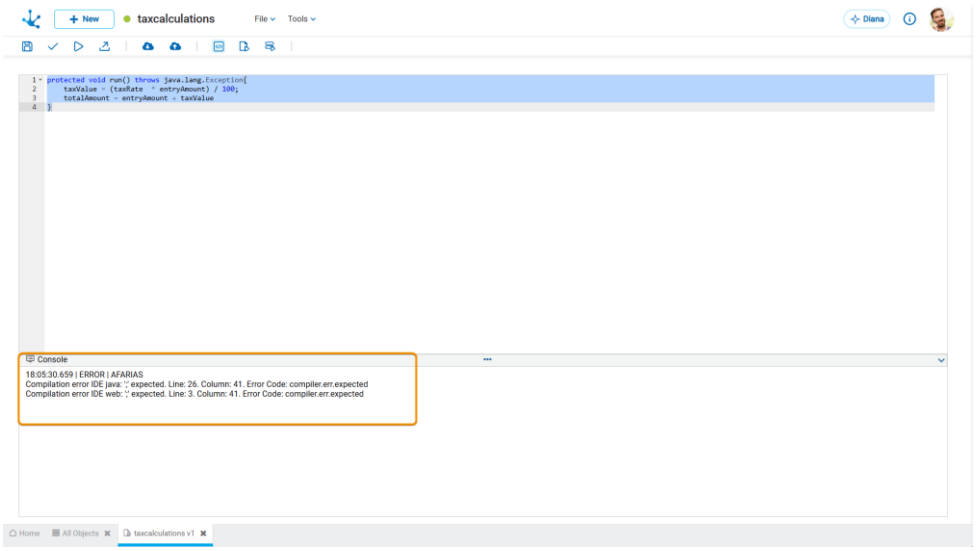


Console

The log console is displayed as a bar at the bottom border of the rule. In the center of the bar, there is an icon ••• that allows to modify the size of the console area, and, on the far right there is an icon that allows to close it. It is available for all types of advanced rules.

In this area, apart from displaying the errors that arise when publishing or executing a rule, the logs written in its code are displayed.

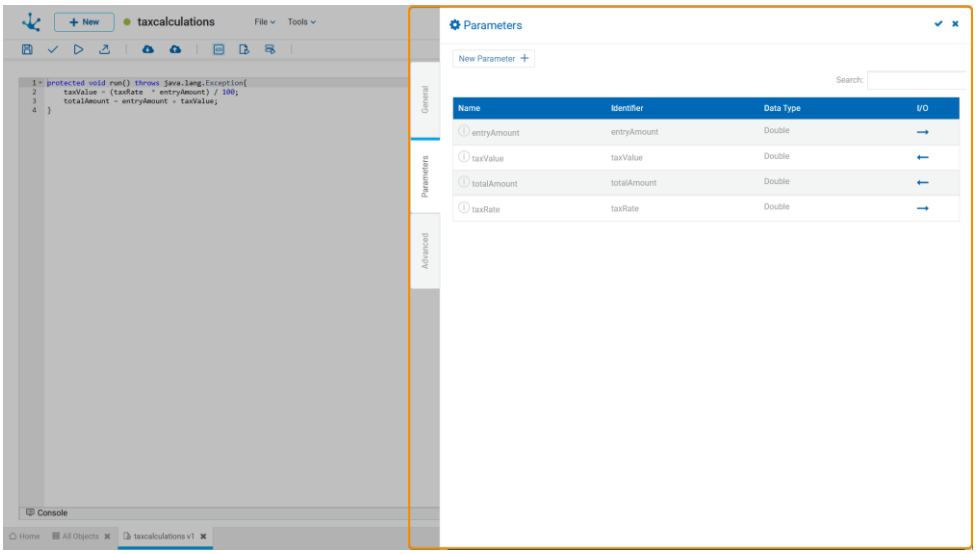
> *For the logs to be displayed, it is necessary to model the property Enable to Write Logs.*



3.4.12.1.2.2. Rule Properties

The advanced rule properties can be entered both when they are created while using the wizard, as well as when modifying an existing one.

To enter the rule properties panel, use the icon in the top toolbar.



Tabs

Entering the properties panel, one, two or three tabs are displayed, depending on the type of rule.

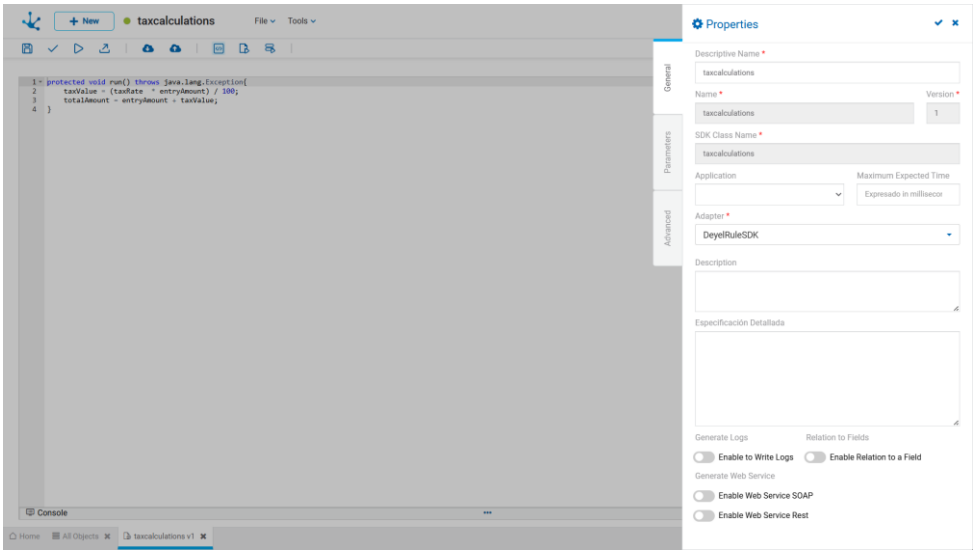
- General
  It is available for all types of rules.

- Parameters
  It is available for all rules except those of the form extension type.

- Advanced
  It is available for all rules except for standard rules, SOAP rules and form extension type rules.

General

The properties panel is displayed on the right side of the advanced rules modeler, where the first tab corresponds to general information.



> *An asterisk "*" on the label indicates that the property is required.*

Properties

Descriptive Name
Name used by the modeler to reference the rule, display it in the rules gallery and in the object tree.

Name
Used internally to reference the rule within forms, entities, pages, embedded rules, processes or as parameters of other advanced rules. Unlike other objects, the rule name also acts as a unique identifier to identify a rule. Once saved, the rule cannot be changed.

Version
Used internally with the rule name to reference the rule within forms, entities, pages, embedded rules, processes or as parameters of other advanced rules. Once saved, the rule cannot be changed.

SDK Class Name
Name that represents the object in the SDK service and model classes.

Application
Allows to define the application to which the object belongs. If no application is reported, the object is assigned to the application "Global".

Adapter
Used to determine the adapter to be used by the rule. Only it is displayed on this screen.

Operation
Used to choose the operation to be executed. This property is only displayed when the chosen adapter type is "Web Services". It is displayed in this section, it is necessary to go to the "parameters" tab in order to change it.

Description
Text that defines the rule describing its functionality and optionally its content.

Generate Logs

Enable to Write Logs

It allows displaying the logs added to the rule code in the console.

> When a rule is imported in test or production environments, log writing is automatically disabled.

Generate Web Service

Enable Web Service Soap

Used to publish the rule as a SOAP web service to be consumed from outside **Deyel**. This property is only displayed when the chosen adapter type is either "Standard Rule" or "JDBC".

Enable Web Service Rest

Used to publish the rule as a Rest API to be consumed from outside **Deyel**. This property is only displayed when the chosen adapter type is "Standard Rule".

> *When this option is enabled, the getUserAgent() method is available to retrieve the user agent from which the rule is executed.*
> *For example, in the rule code, the value obtained by this method can be assigned to an output parameter as follows:*
> *pOutput = getUserAgent();*
> *This method is available only for SDK rules.*

Relation to Fields

Enable Relation to a Field

It allows modeling a relation to this rule, in the field properties of the entity or page.
When activating this property, the rule becomes suitable for linking to fields and the corresponding parameters necessary for its correct implementation are automatically generated.

Specific Properties

Rules with a "Rest" adapter type add specific properties for this type of execution.

HTTP Method

Configures the HTTP method to use.

Context Type

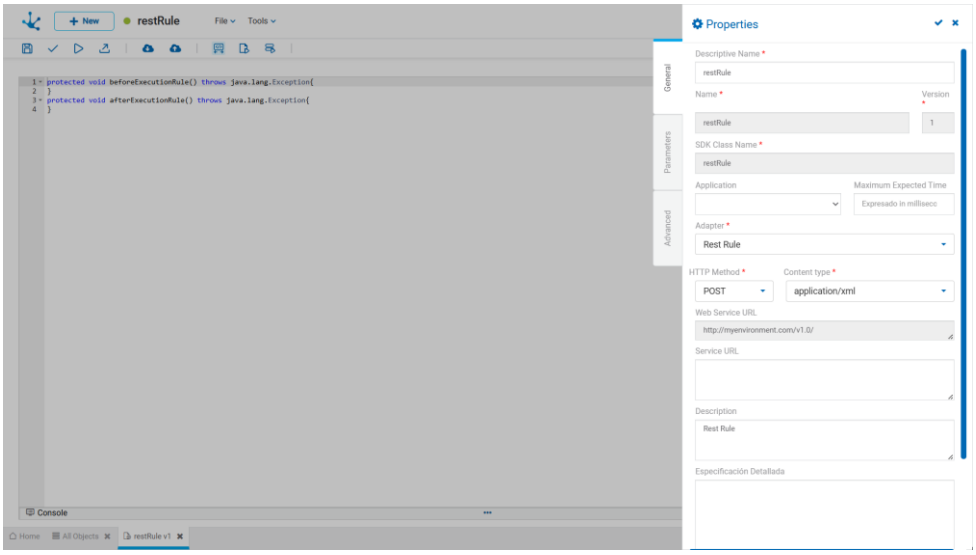
Configures the content type of the request.

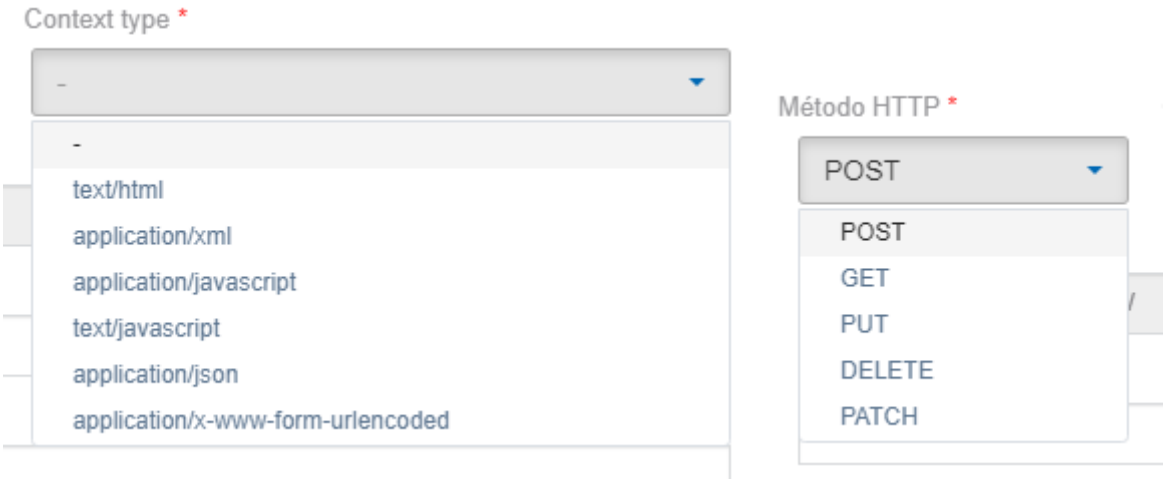
Web Services URL

Url entered in the adapter, displayed in a reading mode to facilitate the correct writing of the final url.

URL Service

Url of the service to consume, the final url is the concatenation of the Web Services URL and the URL Service. The addValueToURL(String key, String value) method can be used within the Pre/Post WS section to replace a part of the url (determined by the key) with a new value (determined by the value).





Actions

The icon ✔ is used to confirm the modifications made in the properties panel.

The icon ✖ is used to close the properties panel, if it was not previously saved, changes are discarded.

Parameters

The second tab of the side panel corresponds to the rules parameter. In this tab, the input and output parameters are defined, which are the communication channel between rules and the processes that use them.

If the rule has enabled the property Enable Relation to a fFeld, some parameters are automatically generated to ensure correct implementation.

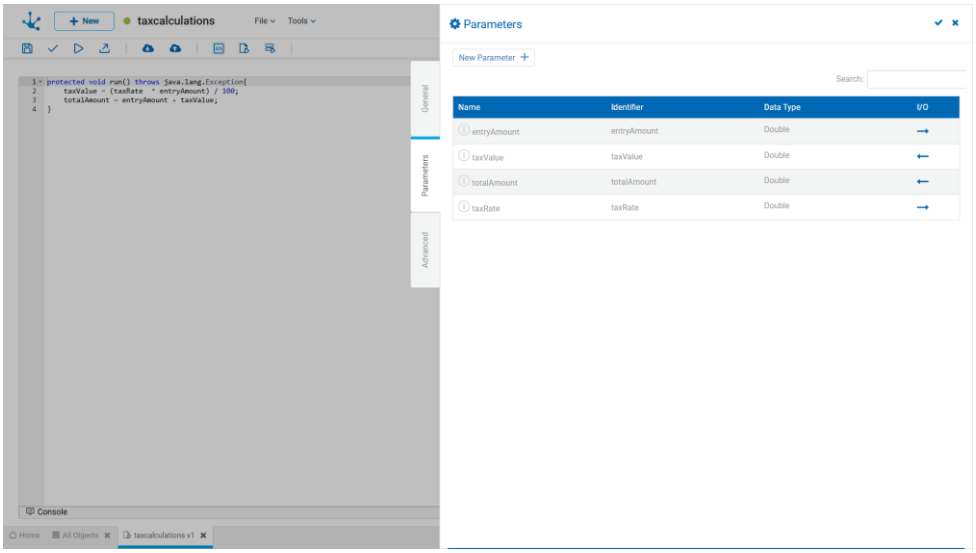
Parameters can be configured in any advanced rule except those of forms extension type.

Parameter Grid

The grid columns identify the properties of each parameter.

By placing the mouse over the grid, the following operations are enabled:

Allows modifying an existing parameter.
Allows deleting a parameter.



Predefined Parameters of Rules Related to Fields

When a rule is related to a field, the following parameters are displayed on the grid:

- **Input**
  This input parameter defines the search and pagination criteria for obtaining the data. The parameter is of type RuleRelationInput and that object contains the following attributes:
  - likeValue: Value entered into an autocomplete field, used as a criterion to filter the possible output values.
  - page: Number of the requested results page.
  - pageSize: Number of results to display per page.
  - currentValues: List of codes previously selected in a field related to the rule, avoiding the repetition of values already chosen.

- **Output**
  This parameter returns the results organized into pages. The parameter is of type RuleRelationOutput and that object contains the following attributes:
  - results: List of results retrieved for the relation. Each item in the list is an object of type RuleValue.
  - pagin: Set of pagination attributes that allow navigation between results.

  RuleValue Object
  Each option in the results list is a RuleValue object, and that object contains the following attributes:

- code: A unique identifier for each value.
- value: Description of the value.

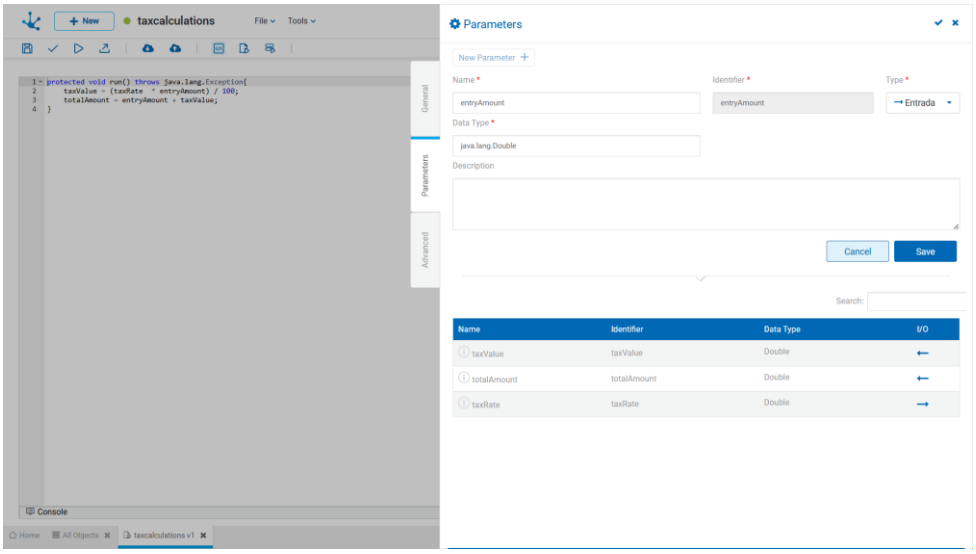
Pagin Object

The Pagin object handles the organization of results across multiple pages, and that object contains the following attributes:

- page: Current page.
- pageSize: Number of results to display per page.
- pages: Total number of pages available.
- total: Total number of results.

Creating a Parameter

New Parameter + It allows adding a new parameter to the grid by filling in all its values.



.

Properties

Name

Name the modeler uses to refer to the parameter.

Identifier

Uniquely identifies the parameter. It is used within Java programming code.

Type

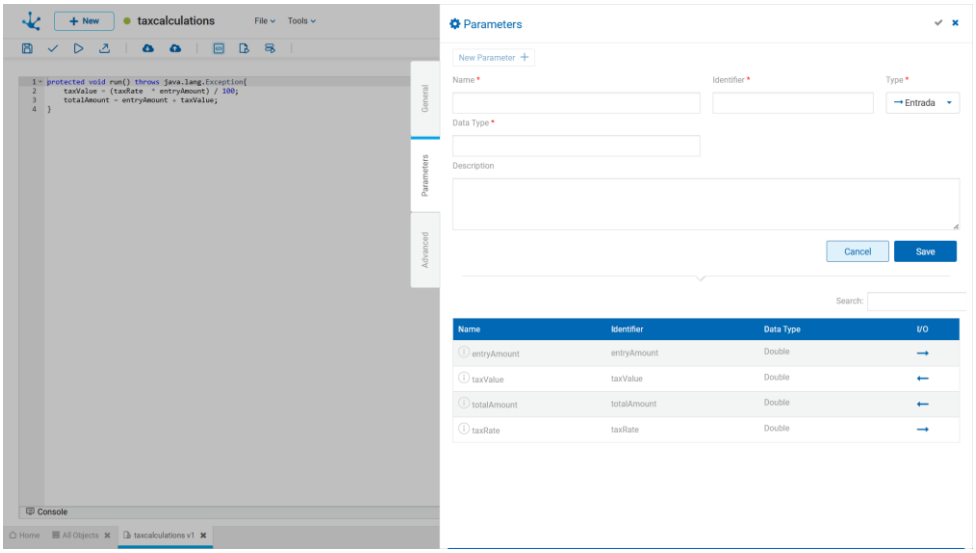
Specifies whether the parameter is input → or output ←.

Data Type

It can be any type of Java object allowed in **Deyel**, either a primitive (integer) or an array.
The type wizard, while typing, gives the most common data type values to use.
By placing the mouse over the data type in the grid, the complete data type can be displayed, to help.

Description

Describes the parameter use. Its writing when modeling the parameters is optional.
If it was entered, the colored icon is displayed in the grid, in the Name column , otherwise the icon is colorless .
Placing the cursor over the icon shows the help.



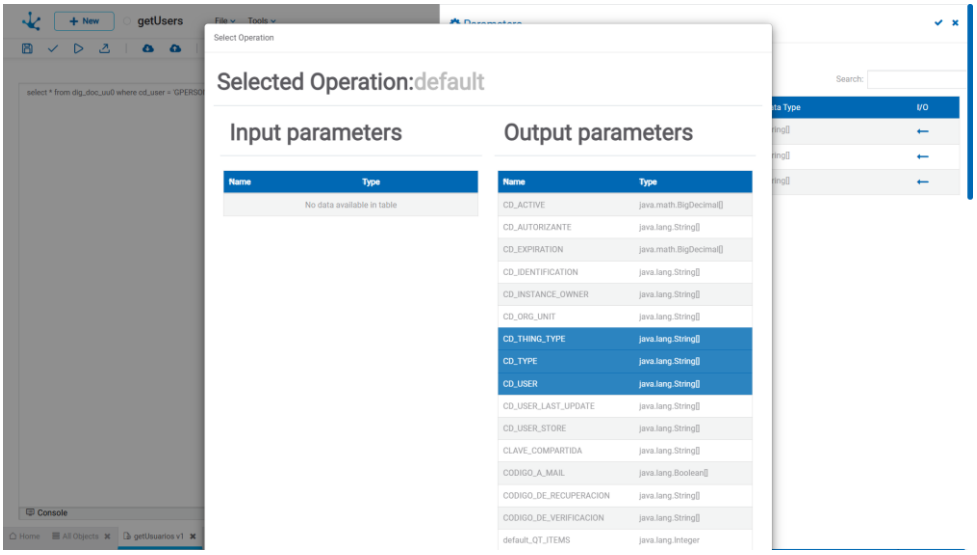
> *An asterisk "*" on the label indicates that the property is required.*

Wizard to Create Parameters in JDBC Rules

New Parameter + Allows to invoke the wizard available for JDBC type rules.

This wizard allows to select the parameters defined in the SQL statement as input parameters and the columns resulting from the query execution as output parameters.
In this way, the probability of errors is minimized by not having to manually transcribe the names of columns and parameters.

For the wizard to return results, it is necessary to define the SQL statement to execute before invoking it. If it were a Select statement, each input parameter should be preceded by a “?” mark. If executing a store procedure, the output parameters can also be specified, each of them should be preceded by an "@" character.

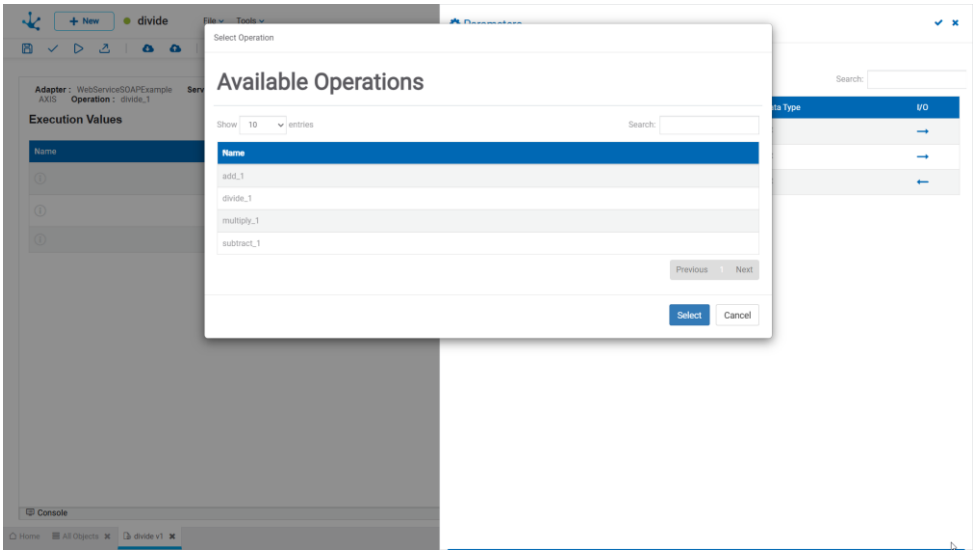


Wizard to Add Operations in SOAP Rules

Add Operation Allows to invoke the wizard available for SOAP type rules.

The wizard analyzes the adapter used and shows the operations available for such object, along with the input and output parameters for each one.

It allows to select the operation that is executed by default and its parameters, to transfer them to the rule interface. In this way, the probability of errors is minimized by not having to manually transcribe the names of operations and parameters.



Advanced

The third tab of the side panel corresponds to the objects wizard. When entering the modeler, it is displayed for all rules except for SOAP rules and form extension rules.

This tab lists the objects to be used in the rule.

Creating a relation using the object wizard allows to:

- Work with the model and service classes of the related object from the rule modeler without having to depend on a Java IDE.
- Always keep the Java imports statements of related objects up to date in the rule, regardless of whether the object has been modified after having created the relation.

If significant changes are made to the related objects used in the rules, when they are published:

- Java imports statements of the rules that use them are updated. Rules remain in the same state they were in, with updated modification date.
- Verify that the rule using them is not in a modified state. If modified, it will be requested to be published or unpublished.

Related Objects Grid

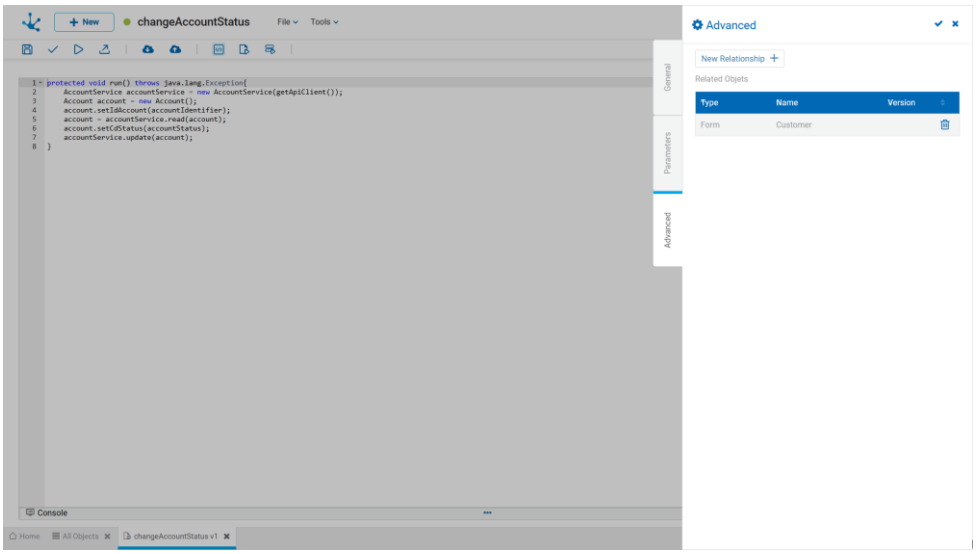
The grid columns identify the related object.

Type
Indicates the object type. It can be a form, process or rule.

Name
It is used at the modeling level to refer to the object.

Version
Indicates the object version for processes and rules.

By hovering the mouse over each related object in the grid, the delete operation is enabled.



New Relationship + Opens a panel with the forms, processes and rules that the user can select according to their security permissions. Once selected, it is displayed in the grid of related objects.

3.4.12.1.2.3. Deyel SDK for Java

By using the Deyel SDK (Software Development Kit) **Deyel** incorporates the ease of developing advanced and integration rules that interact with the platform objects and developed applications, in an easy and intuitive way.

These rules can be developed in the advanced rules modeler in the case of those that require low code or when they are more complex in a Java IDE, inheriting all its advantages.

Tools to Develop Advanced Rules with SDK

Deyel SDK allows to use **Deyel** objects in advanced rules, by using one class with the model and another one with the object's service.

Model Class

The model class corresponds to the properties that each object has.

For example, the "Account (CRM_ACCOUNT)" form model class of the CRM application contains the Identifier, Company, Opening, Industry, Source and Status fields, among others. It also contains the getters and setters for each of these fields.

The model includes:

- Object constructor.
- Set of getters and setters for each object attribute.
- Getters for each object attribute for use in SQL shows, from the Deyel SDK.
- Internal classes if necessary. For example, the representation of iteratives for forms.

Service Class

A service class contains the operations to manage the model. For example, the service class of a form allows to perform the create, update, delete, query, and find operations.

The service class includes:

- CRUD (Create, Read, Update, Delete) of the object.
- Searches
- Specific methods to work on the model.
- A specific method for executing an SQL show on the model.

Each object service class has a common method available to initialize all objects

Operation	Description	Parameters
getAPIClient()	Initializes the service to be able to use it.	

The service can also be initialized using a specific method that enables the management of its own connection to the database and the use of methods to execute the commit and rollback commands.

Operation	Description	Parameters
getOwnApiClient()	Initializes the service and returns the API client of such service to be used in services that require managing their own connection to the database.	
ownConnectionCommit()	Executes a commit command to the connected database so that the operations performed on the services of **Deyel** objects are applied.	
ownConnectionRollback()	Executes a rollback command to the connected database. If an error occurs, it is used to cancel all operations performed on the services of **Deyel** environment.	

Interaction with Application Objects

In order to interact with **Deyel** objects from a Java IDE, the model and service classes corresponding to each object must be downloaded.

These classes are included when downloading the rule that interacts with such **Deyel**objects, from the "Download" option available on the top toolbar of the rule modeler.

By using the model and service classes, Deyel SDK allows to use the following objects:

- Forms
- Cases
- Rules
- Value Lists
- Users

- Organizational Units
- Calendars
- Email Sending
- Java Files

Entities

The model class contains the properties with its getters and setters, while the service class contains the operations performed with the model.

For example, if the "Account" entity is used, when downloading the entity Java sources, an account.java file is obtained, representing the model and another accountService.java file with the available operations of the service.

Model Class Content

An entity model contains:

- Model constructor.
- Getters and setters of its fields.
- Getters for each field for use in SQL shows from the Deyel SDK.
- Inner classes representing the “Multiple occurrences” containers model, if any.

Specific method of the model class

In addition to the getters and setters, there are specific methods related to fields.

Operation	Description	Parameters
getReferen-ce_”idFileTypeField”()	It allows to retrieve the FileRe-ference object associated with the file field of the instance.	

Data Type

Model properties for an entity can have the following data types.

Equivalence between entity field data types and Java model is shown.

Control	Data Type	Java
Text	Alphanumeric (length)	String
	Alphanumeric Uppercase (len-	String

Control	Data Type	Java
	gth) Large Alphanumeric Rich Text	 String String
Number	Integer Large Integer Decimal	Integer Long Double
Time	Time Local time	java.sql.Time java.sql.Time
Date	Date Date and Time Local Date Date and Local Time	java.sql.Date java.sql.Timestamp java.sql.Timestamp java.sql.Timestamp
Image	Image in Folder Image in Database	String String
File	File in Database	String
Check	Boolean	Boolean

Service Class Content

The service allows to perform the following CRUD (Create, Read, Update, Delete) and search (search) operations, as well as containing specific methods, for example to retrieve files from the entity.
In addition, the service class includes a specific method to execute SQL shows on the corresponding model.

Operation	Description	Parameters
create(entity)	Create an entity instance.	**entity**: Entity model class

Operation	Description	Parameters
read(entity)	Reads an entity instance.	**entity**: Entity model class
update(entity)	Updates an entity instance.	**entity**: Entity model class
delete(entity)	Deletes an entity instance.	**entity**: Entity model class
readCases(entity)	Allows to retrieve the case instances associated with an entity instance. Returns a list of cases since an entity instance can be related to more than one case.	**entity**: Entity model class
bind(entity,case)	Allows to associate a case to an entity instance.	**entity**: Entity model class **case** : Case model class
shortDescription()	It allows retrieving the short description of the instance sent by parameter.	**formInstance**
get"idField"_description()	It allows retrieving the descriptive value of the indicated field.	**formInstance**
search(searchCriteria)	Allows retrieving the entity instances that meet the search criteria.	**searchCriteria**: Class that allows defining a set of criteria and search configuration parameters
executeQuery(entity, queryCriteria)	Executes the generated show on the model using the different available methods.	**entity**: Entity model class **queryCriteria**: Class that allows defining an SQL show through a set of criteria.

Examples of Use

The examples use the "Account" entity and each example contains the use of "Account" model class and "AccountService" service class.

1. Service creation

This service is created only once in the rule and is reused in different operations.

```
AccountService accountService = new AccountServi-
ce(getApiClient());
```

2. Create an entity instance

An instance of "Account" model class is created, values are assigned to its properties. Using the "AccountService" service class, the new account is saved with the assigned values through the create (account) method.

```
Account account = new Account() ;
account.setNuIdentifNumber("010000123");
account.setDsIndustry("1");                    // 1 = Agriculture
account.setDsSource("7");                      // 7 = Social Networ-
king
account.setDsCompany("Trees Company");
account.setDtOpening((Date) new java.util.Date());
account.setFlLogo(new File("/photo.png"));
account.setCdStatus("1");                      // 1 = Active
Integer accountNumber = accountService.create(account);
log("Account created with account number:"+ accountNumber);
```

3. Read an entity instance

To obtain data from an existing account, an "Account" model class instance is created, an identifier chosen through the corresponding setter is indicated and such instance is read using the read(account) method of the "AccountService" service class. The instance contains all its properties which can be queried using the corresponding getters. The example reads the dsCompany, dtOpening, dtStore, and dtLastUpdate properties.

```
Account account = new Account();
Integer readAccountNumber = 50000 ;
account.setIdAccount(readAccountNumber);
account = accountService.read(account) ;
```

```
// Once read, continue working
String dscompany = account.getDsCompany();
Date dtopening = account.getDtOpening();

// Audit data
Timestamp dtStore = account.getDtStore();
Timestamp dtLastUpdate = account.getDtLastUpdate();
```

4. Modify an entity instance

To modify data from an existing account, an "Account" model class instance is created, an identifier chosen through the corresponding setter is indicated and such instance is read using the read(account) method of the “AccountService” service class. The dsCompany property is then modified using the corresponding setter and the update(account) method of the service is invoked to update the account.

```
Account account = new Account();
account.setIdAccount(50000);
account = accountService.read(account);

// Once read, update
account.setDsCompany("Trees Company");
accountService.update(account);
```

5. Delete an entity instance

To delete an existing account, an instance of the "Account" model class is created, indicating its identifier using the corresponding setter. The delete(account) method of the service is then invoked to delete the instance.

```
Account account = new Account();
account.setIdAccount(50000);
account = accountService.read(account);
accountService.delete(account);
```

6. Use of file and image type fields

To retrieve a file related to an entity instance, use the corresponding get+<fieldname>+(entity) method of the "AccountService" service class.

In this example, the getFlLogo(account) method of the "AccountService" service class is invoked, which allows retrieval of the file corresponding to a logo associated with a File object.

```
Account account = new Account();
account.setIdAccount(50000);

// Account reading
account = accountService.read(account);

// Logo image reading
File logo = accoountService.getFlLogo(account);

// Assign a new logo to the account
account.setFlLogo(new File("Path"));
accountService.update(account);
```

7. Use of internal objects

Internal objects represent the set of fields modeled in containers of multiple occurrences.

In this example, two new “Account.PhoneLine” internal objects are instantiated and values are assigned to their nrPhone and tpPhone properties.
A new list of “Account.PhoneLine” objects can be saved with the setPhoneLine(lsPhones) method or by adding two instances of the “Account.PhoneLine” object to the current list.

```
Account.PhoneLine phones= new Account.PhoneLine();
phones.setNrPhone("01149834589");
phones.setTpPhone("Line phone");

Account.PhoneLine phones1= new Account.PhoneLine();
phones1.setNrPhone("0115123987");
phones1.setTpPhone("Cell phone");

List<Account.PhoneLine> lsPhones = new ArrayList();
lsPhones.add(phones);
lsPhones.add(phones1);
account.setPhoneLine(lsPhones);

// Another option is
account.getPhoneLine().add(phones);
account.getPhoneLine().add(phones1);
```

8. Reading of cases associated with entity instances

The example retrieves the case associated with an entity instance and executes that case.

To use this example in addition to the "Account" (Account) entity, the "Account Registration" (Account Registration) process that uses that entity must be modeled. The case can be started manually or by Deyel SDK methods.

To read cases associated with an entity instance, an instance of the "Account" model class is created, indicating the identifier chosen using the corresponding setter. The "AccountService" service is instantiated and using the readCases(account) method, the "Cases" list is retrieved with the cases associated with the instance with identifier 10.
Having the "Cases" list, it is possible to iterate through it to retrieve some or all of its cases. The "AccountRegistration" service is instantiated and using the execute(accountRegistrationCase) method, the first associated case is executed.

> *Both the entity instance and the case instance must be created.*

```
Account account = new Account();
account.setId(10);

AccountService accountService = new AccountServi-
ce(getApiClient());
List<Cases> listOfCases = accountService.readCases(account);

AccountRegistration  accountRegistrationCase = new AccountRe-
gistration(listOfCases.get(0));

AccountRegistrationService accountRegistrationService = Accoun-
tRegistrationService(getApiClient());
accountRegistrationService.execute(accountRegistrationCase);
```

9. Associate a case with an entity instance

In the example, a case associated with an "Account" entity instance is read to then associate such case to a "Contact" entity instance, leaving both instances associated with the same case.

The case associated with the "Account" entity instance is read with the readCases(account) method, described in example 8.
The case is associated with the "Contact" entity instance, with the
bind(contact,accountRegistrationCase) method.

> *A case can only be linked to an entity instance if there is a relation between the entity and the process to which the case to be associated belongs.*

```
Account account = new Account();
account.setId(10);

AccountService accountService = new AccountServi-
ce(getApiClient());
List<Cases> listOfCases = accountService.readCases(account);

AccountRegistration accountRegistrationCase = (AccountRegistra-
tion) listOfCases.get(0);
AccountRegistrationService accountRegistrationService = Accoun-
tRegistrationService(getApiClient());

Contact contact = new Contact();
contact.setId(20);

ContactService contactService = new ContactServi-
ce(getApiClient());

contactService.bind(contact,accountRegistrationCase);
```

10. Retrieve the code and description of the field value associated with a value list

In order to use this example, the "States" value list should be modeled in **Deyel** and values should have been entered to this list.

Code	Descriptive Value
1	Active
2	Inactive

It is required to relate an "Account" form field with the "States" value list.

> *When a field has a relation defined, the content can use its code or its descriptive value.*

This example retrieves the field code associated with the value list.

```
String stateCode = account.getAccountState();
//Returns: 1
```

While in the following example, the description of the field associated with the value list is retrieved.

```
String stateDescription = accountServi-
ce.getAccountSate_description(account);
//Returns: Active
```

> *The code is retrieved through the model object and the description is retrieved through the service object.*

11. Updating entity instances with commit or rollback

This example declares a variable that represents the service of the object to be updated. By using it, the list of existing instances is retrieved and traversed with the FOR statement. For each of them, the service is redefined, obtaining its own API client, the VAT value is calculated and updated, and an attempt is made to perform a commit command or a rollback command to the connected database, as appropriate.

```
protected void run() throws java.lang.Exception{
    TestFormService xTestFormService = new TestFormServi-
ce(getApiClient());
    SearchCriteria xSearchCriteria = new SearchCriteria();
    List<TestForm> xListResult = xTestFormServi-
ce.search(xSearchCriteria).getResult();
    for(int j = 0; j < xListResult.size(); j++) {
        try {
            xTestFormService = new TestFormServi-
ce(getOwnApiClient());
            TestForm xElement = xListResult.get(j);
            xElement.setValueOfVAT((vatrate * entryamount) /
100);
            xTestFormService.update(xElement);
            ownConnectionCommit();
        } catch (Exception e) {
            ownConnectionRollback();
        }
```

```
            }
        }
```

Search

Searches on entity instances can be performed using the following objects.

Criteria- represents search criteria on entity data. It is made up of elements that can be fields and values, connected by operators.

Operator	Description
eq	Equal
neq	Different
gt	Greater
gte	Greater equal
lt	Less
lte	Less equal
between	Between
betweene	Between and admits equals
nbetween	Out of range
like	Contain
nlike	It does not contain
startsWith	Starts with
nstartsWith	It does not start with
in	Included
nin	Not included
endsWith	Ends with

Operator	Description
nendsWith	Does not end with
isNull	It is null
notNull	It is not null

SearchCriteria: an object that groups search criteria (Criteria object), it also allows parameterizing the order of the result, the size of the reading page, and the number of pages to be retrieved. Reading by pages is used since the data volume on a form can be very large.

Entity Service: service class of the entity on which the search is being carried out. Contains the search(searchCriteria) operation that must receive the SearchCriteria object as a parameter and returns the SearchResult object.
Example:
SearchResult searchResult = accountService.search(searchCriteria);

SearchResult: object that contains the search results. Contains a list of entity instances, which are instances of the corresponding model, for example "Account". This object allows to know the total number of reading pages resulting from the search, the size of each page, and the number of pages to retrieve.

Search Examples

1. Search

This example retrieves a list of accounts with active state, where such state corresponds to code "1". A quantity of 15 lines per reading page and the number of pages to retrieve are defined. The results are ordered by company name in ascending order.

```
SearchCriteria searchCriteria = new SearchCriteria();
Criteria criteria1 =
        Criteria.eq("CdStatus","1");
searchCriteria.addCriteria(criteria1);
searchCriteria.setPageSize(15);
searchCriteria.setPage(1);
searchCriteria.addOrderAsc("DsCompany");


SearchResult searchResult =
            accountService.search(searchCriteria);
List<Account> instancesResult = searchResult.getResult();
```

2. Constructor types for search criteria

This example creates a Criteria object and defines different search conditions using the "equal", "different", "greater", "greater equal", "less", "less equal", "between", "between and allows equals", "out of range", "contains", "does not contain", "starts with", "does not start with", "included", "not included", "ends with", "does not end with", "is null", "is not null".

```
// Equal cdStatus= 1 corresponds to an active account
 Criteria criteria = Criteria.eq("CdStatus", "1");

// Different cdStatus= 1 corresponds to an active account
// Criteria criteria =  Criteria.neq ("CdStatus","1");

// Greater
// Criteria criteria = Criteria.gt ("qtScore", 20) ;

// Greater equal
// Criteria criteria = Criteria.gte ("qtScore", 20) ;

// Less
// Criteria criteria = Criteria.It ("qtStore", 50) ;

// Less equal
// Criteria criteria = Criteria.Ite ("qtStore", 50) ;

// Between
// Criteria criteria = Criteria.between ("dtOpening",
// Date.valueOf("01/01/1990"), Date.valueOf("01/01/2020"));

// Between and admits equals
// Criteria criteria = Criteria.between("dtOpening",
// Date.valueOf("01/01/1990"), Date.valueOf("01/01/2020"));

// Out of range
// Format YYYY-MM-DD is used
// SimpleDateFormat format =
//      new SimpleDateFormat("yyyy-MM-dd");
// Date dateFrom =
// new Date (format.parse( "1990-01-01" ).getTime());
// Date dateTo =
```

```
//      new Date(format.parse( "2020-01-01" ).getTime());
//
// Criteria criteria =
// Criteria.nbetween ("dtOpening", dateFrom, dateTo );

// Contains In an internal object
// Criteria criteria = Criteria.like("eMailLine/dsEmail",
// yahoo.com.ar");
```

> ***Deyel*** *uses the data type java.sql.Date for "date" type fields, so the transformation from java.util.Date to java.sql.Date must be done, when it is required to create an object with this type of data.*

```
// Contains
// Criteria criteria = Criteria.like("dsCompany","Inc.");

// It does not contain
// Criteria criteria = Criteria.nlike("dsCompany","Inc.");

// Starts with
// Criteria criteria = Criteria.startsWith("dsCompany","The");

// It does not start with
// Criteria criteria = Criteria.nstartsWith("dsCompany","The");

// Included dsIndustry = 6 is Communications, 8 is Education
// Criteria criteria =
// Criteria.in("dsIndustry", Arrays.asList("6", "8" ));

// Not Included dsIndustry 6 is Communications, 8 is Education
// Criteria criteria =
// Criteria.nin("dsIndustry", Arrays.asList("6", "8" ));

// Ends with
// Criteria criteria =
// Criteria.endsWith("dsCompany","Corporation");

// Does not end with
// Criteria criteria =
```

```
// Criteria.nendsWith("dsCompany","Corporation");

// It is null
// Criteria criteria =
// Criteria.isNull("CdStatus");

// It is not null
// Criteria criteria =
// Criteria.notNull("CdStatus");
```

3. Operations on the results object

The SearchResult object is iterative and as such, it can be scrolled through in different ways.
The example uses a FOR statement to iterate over each of the instances that meet the search conditions of the dsIndustry and qtScore properties, retrieving the idAccount and dsCompany properties.

```
SearchCriteria searchCriteria = new SearchCriteria();
 searchCriteria.setPageSize(15);
 searchCriteria.addOrderDesc("idAccount");

 // Accounts with industry equal to communications
 searchCriteria.addCriteria
    (Criteria.eq("dsIndustry", "6"));

// Accounts with a score greater than 20
 searchCriteria.addCriteria
         (Criteria.gt("qtScore", 20)  );

 // Scrolls through all the accounts in the search result
 String stringData1 = "";
 for (Account account: searchResult) {
     stringData1 += " - " + account.getIdAccount()
                  + " - " + account.getDsCompany();
 }
 // If I only need the first results sheet
 SearchResult<Account> resultFirstPage =
         accountService.search(searchCriteria);
 List<Account> firstList = resultFirstPage .getResult();

 String stringData = "";
 stringData += " Amount of data " + firstList.size();
```

```
stringData += " Pages " + resultFirstPage .getPages();
stringData += " Actual Page " +
                    resultFirstPage .getCurrentPage();
```

Browsing between Related Entities

If the “Contact” entity has a relation with the “Account” entity,the related account can be retrieved from a business rule, from a contact.

An instance of the “Contact” model class is created and the account identifier is retrieved using the corresponding getter.
Next, an instance of the "Account" model class is created indicating the identifier retrieved from the contact in the setter, and the account instance is read using the read(myAccount) method of the "AccountService" service class.

```
Contact contact = new Contact() ;
Integer idAccount = contact.getIdCompany();
// Retrieve the id of a contact account
// and I read the account with that id

Account myAccount = new Account();
myAccount.setIdAccount(idAccount);
myAccount = accountService.read(myAccount);
```

Ejecución de Consultas SQL

SQL queries are useful when working with large volumes of data. A major advantage of using these queries is that they allow retrieving only the information needed, instead of loading the entire model object.

If the SearchCriteria object is used to search for data, the complete objects for that entity are retrieved, even if only, for example, the identifiers of the instance are needed. This means that more information than necessary is being loaded.
Instead, with the SQL queries available in the Deyel SDK, only the specific data required can be retrieved, making processing faster and more efficient.

Another major advantage of using SQL queries with the Deyel SDK is the ability to perform JOIN operations between entities, a functionality that is not available in SearchCriteria.

Searches for SQL Queries

Searches on SQL queries can be performed using the following objects.

QueryCriteria: object on which the different parts of the SQL show are added. The fields to retrieve from an entity instance can be specified with this object, whether to work or not with iteratives, as well as the search order and grouping, the limit conditions, and other operations detailed in the examples.

Entity Service - executeQuery() method: service class of the entity on which the search is carried out. Contains the executeQuery(formInstance, queryCriteria) operation, which takes as parameters the model object of the entity being worked with and the QueryCriteria object. Returns a QueryResult object as a result.

Entity Model - getter ... QueryName(): model class of the entity on which you want to execute the SQL query. It contains getters for each of the modeled fields, and these methods have the particularity of ending with QueryName. They should be used when adding the different parts of the query to be generated.

QueryResult: object that contains the results of the SQL query execution. They are grouped by column (entity field), which means that to interact with the QueryResult object, the field to retrieve should be indicated to get a list that represents each of the rows retrieved after the SQL query.

Methods

Methods for working with the various objects are described below.

Methods to execute with the QueryCriteria object

Method	Description	Parameters
createSelect(pList)	Creates a "SELECT" statement and adds the values in the pList parameter.	Java.util.List type **pList**: : contains each of the fields to be retrieved from the entity, excluding the iterative container fields
createSelectWithIterative(pList, pMap)	Creates a "SELECT" statement and adds the values in the pList and pMap parameter.	Java.util.List type **pList**: contains each of the fields to be retrieved from the entity, excluding the iterative container fields Java.util.Map type **pMap**: contains, as a key, the iterative identifier. The iterative fields are added as a key value of the map within a java.util.List object
addConditionLimits(pPageNumber,pPerPage)	Adds limit conditions to the SQL statement along with the	**pPageNumber**: page number to retrieve

Method	Description	Parameters
	page to retrieve.	**pPerPage**: number of records to retrieve on the page. By default, it is 100, and at most, 10000
addGroupBy(pColumn)	Adds the GROUP BY clause to the SQL statement.	String **pColumn**: represents the identifier of the entity field by which grouping is to be performed
addGroupBy(pIterativeName, pColumn)	Adds the GROUP BY clause to the SQL statement.	String **pIterativeName**: indicates the identifier of the iterative to which the field indicated in pColumn belongs String **pColumn**: represents the identifier of the entity field by which grouping is to be performed
addOrderCriteria(pColumn,pOrderDesc)	Adds the ORDER BY clause to the SQL statement.	String **pColumn**: represents the identifier of the entity field by which sorting is to be performed Boolean **pOrderDesc**: with value "True" indicates descending order, "False", ascending
addOrderCriteria(pIterativeName,pColumn,pOrderDesc)	Adds the ORDER BY clause to the SQL statement considering iterative fields.	String **pIterativeName**: indicates the identifier of the iterative to which the field indicated in pColumn belongs String **pColumn**: represents the identifier of the entity field by which sorting is to be performed Boolean **pOrderDesc**: with value "True" indicates descending order, "False", ascending
addConditionBetween(pColumn,pValueFrom, pValueTo)	Adds the BETWEEN condition to the statement. The value searched will be between the	String **pColumn**: represents the identifier of the entity field

Method	Description	Parameters
	values indicated in the pValue-From and pValueTo parameters.	Object Type **pValueFrom**: represents the initial value from which the match with the pColumn value is sought Object Type **pValueTo**: represents the final value from which the match with the pColumn value is sought
addConditionBetween(pIterativeName,pColumn, pValueFrom, pValueTo)	Adds the BETWEEN condition to the statement. The value searched will be between the values indicated in the pValue-From and pValueTo parameters.	String **pIterativeName**: indicates the identifier of the iterative to which the field indicated in pColumn belongs String **pColumn**: represents the identifier of the entity field Object Type **pValueFrom**: represents the initial value from which the match with the pColumn value is sought Object Type **pValueTo**: represents the final value from which the match with the pColumn value is sought
addConditionGreater(pColumn,pValue, useGreaterOrEqual)	Adds the > or >= condition to the SQL statement.	String **pColumn**: represents the identifier of the entity field Object Type **pValue**: contains the value being compared for the field contained in pColumn Boolean **useGreaterOrEqual**: indicates whether or not the greater than or equal operator is used. With value "True", >=, "False", > is used
addConditionGreater(pIterativeName, pColumn, pValue, useGreaterOrEqual)	Adds the > or >= condition to the SQL statement.	String **pIterativeName**: indicates the identifier of the iterative to which the field indicated in pColumn belongs String **pColumn**: represents the identifier of the entity field

Method	Description	Parameters
		Object Type **pValue**: contains the value being compared for the field contained in pColumn Boolean **useGreaterOrEqual**: indicates whether or not the greater than or equal operator is used. With value "True", >=, "False", > is used
addConditionLike(pColumn, pValue)	Adds the LIKE condition to the SQL statement.	String **pColumn**: represents the identifier of the entity field String **pValue**: contains the value for the LIKE condition
addConditionLike(pIterativeName, pColumn, pValue)	Adds the LIKE condition to the SQL statement.	String **pIterativeName**: indicates the identifier of the iterative to which the field indicated in pColumn belongs String **pColumn**: represents the identifier of the entity field String **pValue**: contains the value for the LIKE condition
addConditionLower(pColumn, pValue, useLowerOrEqual)	Adds the < or <= condition to the SQL statement.	String **pColumn**: represents the identifier of the entity field Object Type **pValue**: contains the value being compared for the field contained in pColumn Boolean **useLowerOrEqual**: indicates whether or not the greater than or equal operator is used. With value "True", <=, "False", < is used
addConditionLower(pIterativeName, pColumn, pValue, useLowerOrEqual)	Adds the < or <= condition to the SQL statement.	String **pIterativeName**: indicates the identifier of the iterative to which the field indicated in pColumn belongs String **pColumn**: represents the

Method	Description	Parameters
		identifier of the entity field Object Type **pValue**: contains the value being compared for the field contained in pColumn Boolean **useLowerOrEqual**: indicates whether or not the greater than or equal operator is used. With value "True", <=, "False", < is used
addConditionNotEquals(pColumn, pValue)	Adds the != condition to the SQL statement.	String **pColumn**: represents the identifier of the entity field Object Type **pValue**: contains the value being compared for the field contained in pColumn
addConditionNotEquals(pIterativeName, pColumn, pValue)	Adds the != condition to the SQL statement.	String **pIterativeName**: indicates the identifier of the iterative to which the field indicated in pColumn belongs String **pColumn**: represents the identifier of the entity field Object Type **pValue**: contains the value being compared for the field contained in pColumn
addConditionEquals(pColumn, pValue)	Adds the = condition to the SQL statement.	String **pColumn**: represents the identifier of the entity field Object Type **pValue**: contains the value being compared for the field contained in pColumn
addConditionEquals(pIterativeName, pColumn, pValue)	Adds the = condition to the SQL statement.	String **pIterativeName**: indicates the identifier of the iterative to which the field indicated in pColumn belongs String **pColumn**: represents the identifier of the entity field Object Type **pValue**: contains

Method	Description	Parameters
		the value being compared for the field contained in pColumn
addConditionNot-Null(pColumn)	Adds the IS NOT NULL condition to the SQL statement.	String **pColumn**: represents the identifier of the entity field that is required to be non-null.
addConditionNot-Null(pIterativeName,pColumn)	Adds the IS NOT NULL condition to the SQL statement.	String **pIterativeName**: indicates the identifier of the iterative to which the field indicated in pColumn belongs String **pColumn**: represents the identifier of the entity field that is required to be non-null.
addConditionNulla-ble(pColumn)	Adds the IS NULL condition to the SQL statement.	String **pColumn**: represents the identifier of the entity field that is required to be null.
addConditionNulla-ble(pIterativeName,pColumn)	Adds the IS NULL condition to the SQL statement.	String **pIterativeName**: indicates the identifier of the iterative to which the field indicated in pColumn belongs String **pColumn**: represents the identifier of the entity field that is required to be null.
addConditionNOTIN(pColumn, pValues)	Adds the NOT IN condition to the SQL statement.	String **pColumn**: represents the identifier of the entity field List **pValues**: This parameter is a value list (List) that will be used in the NOT IN clause of the SQL statement. Each value in this list represents a value that should be excluded from the results.
addConditionNO-TIN(pIterativeName, pColumn, pValues)	Adds the NOT IN condition to the SQL statement.	String **pIterativeName**: indicates the identifier of the iterative to which the field indicated in pColumn belongs String **pColumn**: represents the identifier of the entity field

Method	Description	Parameters
		List **pValues**: This parameter is a value list (List) that will be used in the NOT IN clause of the SQL statement. Each value in this list represents a value that should be excluded from the results.
addConditionIN(pColumn, pValues)	Adds the IN condition to the SQL statement.	String **pColumn**: represents the identifier of the entity field List **pValues**: This parameter is a value list (List) that will be used in the IN clause of the SQL statement. Each value in this list represents a value that should be included in the results.
addConditionIN(pIterativeName, pColumn, pValues)	Adds the IN condition to the SQL statement.	String **pIterativeName**: indicates the identifier of the iterative to which the field indicated in pColumn belongs String **pColumn**: represents the identifier of the entity field List **pValues**: This parameter is a value list (List) that will be used in the IN clause of the SQL statement. Each value in this list represents a value that should be included in the results.
addJOIN(pColumn, pObjectRelation, pKeyObjectRelation)	Adds the INNER JOIN clause to the SQL statement. A JOIN is performed where the pColumn field is equal to the pKeyObjectRelation field of the pObjectRelation object.	String **pColumn**: represents the identifier of the entity field with which the JOIN associated with the field represented by the pKeyObjectRelation field of the pObjectRelation object is sought Object type **pObjectRelation**: represents the model object on which an INNER JOIN is executed String **pKeyObjectRelation**:

Method	Description	Parameters
		represents the entity field identifier of the pObjectRelation model object
addSum(pColumn)	Adds the SUM function to the SQL statement.	String **pColumn**: represents the identifier of the entity field on which the SUM function is performed
addSum(pIterativeName, pColumn)	Adds the SUM function to the SQL statement.	String **pIterativeName**: indicates the identifier of the iterative to which the field indicated in pColumn belongs String **pColumn**: represents the identifier of the entity field on which the SUM function is performed
addCount(pColumn)	Adds the COUNT function to the SQL statement.	String **pColumn**: represents the identifier of the entity field on which the COUNT function is performed
addCount(pIterativeName, pColumn)	Adds the COUNT function to the SQL statement.	String **pIterativeName**: indicates the identifier of the iterative to which the field indicated in pColumn belongs String **pColumn**: represents the identifier of the entity field on which the COUNT function is performed
addAVG(pColumn)	Adds the AVG function to the SQL statement.	String **pColumn**: represents the identifier of the entity field on which the AVG function is performed
addAVG(pIterativeName, pColumn)	Adds the AVG function to the SQL statement.	String **pIterativeName**: indicates the identifier of the iterative to which the field indicated in pColumn belongs String **pColumn**: represents the identifier of the fentity field on which the AVG function is per-

Method	Description	Parameters
		formed

Methods to execute with the QueryCriteria object

Method	Description	Parameters
getQueryResult(pColumn): List	Returns a list containing the values corresponding to the field specified by pColumn.	String **pColumn**: represents the identifier of the entity field from which the retrieved values are to be obtained
getCountResult(pColumn):List	Returns a list containing the results of the COUNT functions for the field specified by pColumn.	String **pColumn**: represents the identifier of the entity field for which the count was performed.
getSumResult(pColumn): List	Returns a list containing the results of the SUM functions for the field specified by pColumn.	String **pColumn**: represents the identifier of the entity field for which the sum was performed.
getAVGResult(pColumn): List	Returns a list with the average (AVG) corresponding to the field indicated in pColumn.	String **pColumn**: represents the identifier of the entity field from which the retrieved rows are to be obtained
getExistsMoreResults(): boolean	Returns a boolean value indicating whether there are more results that meet the conditions of the executed SQL statement and that can be processed.	

Method to execute with the Entity Service object

Method	Description	Parameters
executeQuery(formInstance, queryCriteria): QueryResult	Executes an SQL statement based on the specified criteria and returns a QueryResult object with the corresponding	**formInstance**: object of the entity model being worked with and on which the SQL statement is executed.

Method	Description	Parameters
	results.	**queryCriteria**: QueryCriteria object that contains the SQL statement settings, configured through the methods available in that class.

Execution Examples

1. Search

This example retrieves a list of account identifiers with active state, where the active state corresponds to code "1". It is defined that there are 15 records per reading page, and the number of pages to retrieve is specified. The results are ordered by company name in ascending order.

```
//Creating QueryCriteria object
QueryCriteria xQueryCriteria = new QueryCriteria();

//Creating model object
Account xAccount = new Account();

//Creating the service for the entity
AccountService xAccountService = new AccountServi-
ce(getApiClient());

//Selecting the identifier as the column to retrieve
List<String> xColumns = new ArrayList<>();
xColumns.add(Account.getIdAccountQueryName());

//Configuring the query to select the specified columns.
xQueryCriteria.createSelect(xColumns);

//Add conditions to the query
xQueryCrite-
ria.addConditionEquals(Account.getCdStatusQueryName(), "1");
xQueryCriteria.addConditionLimits(15, 1);
xQueryCrite-
ria.addOrderCriteria(Account.getDsCompanyQueryName(), false);
```

```
//Executing the query and obtaining the results
QueryResult xQueryResult = xAccountServi-
ce.executeQuery(xAccount, xQueryCriteria);
List instancesResult = xQueryRe-
sult.getQueryResult(Account.getIdAccountQueryName());
```

2. Creating the "SELECT" statement

To create a query, two methods can be used to incorporate the corresponding SELECT statement into the QueryCriteria object, along with the fields to be retrieved from the entity.

First Method

```
createSelect(List<String> xColumns)
```

This method incorporates the “SELECT” statement to the QueryCriteria object and adds each of the entity's fields that have been previously added to the pColumns list. Each field added to the pColumns list must be a non-iterative field.

```
//Selecting the identifier as the column to retrieve
List<String> xColumns = new ArrayList<>();
xColumns.add(Account.getIdAccountQueryName());

//Creating the SELECT statement with the specified columns
xQueryCriteria.createSelect(xColumns);
```

Second Method

```
createSelectWithIterative(List<String> pFilters, Map<String,
List<String>> pMap)
```

This method allows adding fields that belong to an iterative container of the entity to the QueryCriteria object. Fields that do not belong to iteratives should be included in the pFilters list, while fields that belong to iteratives should be added to a separate list. This list must then be added to a variable of type “HashMap”, using the identifier of the iterative to which the fields belong as the key.
To work with the iterative fields and add them to the corresponding variable, it is necessary to use the iterative class, which is included within the form model object.

```
```

```
//Selecting the identifier as the column to retrieve
List<String> xColumns = new ArrayList<>();
xColumns.add(Account.getIdAccountQueryName());

//Selecting the dsEmail field from the eMailLine iterative con-
tainer
Map<String,List<String>> xMap = new HashMap<>();
List<String> xIterativeColumns = new ArrayList<>();
xIterativeColumns.add(Account.EMailLine.getDsEmailQueryName());
xMap.put(Account.getEMailLineQueryName(), xIterativeColumns);

//Creating the select
xQueryCriteria.createSelectWithIterative(xColumns, xMap);
```

3. Limits of results

When working with SQL queries in the Deyel SDK, it is important to note that there is a limit to the number of results returned. This limit can be adjusted by the following method:

```
addConditionLimits(int pPageNumber, int pPerPage)
```

In this method, the first parameter, pPageNumber, indicates the number of the page to be retrieved, while the second parameter, pPerPage, specifies the maximum number of records per page.

> *By default, Deyel retrieves up to 100 records per page, and the maximum limit of records to retrieve per page is 10,000. If a higher value is set, the SQL query execution will fail.*

4. Search conditions

Various search conditions can be added to the QueryCriteria object to perform SQL queries.

This example creates a QueryCriteria object and defines different search conditions using the "equal", "different", "greater", "greater or equal", "less", "less equal", "between", "like", "is null" and "is not null", " in" and "not in" with the corresponding addCondition method.

For each type of condition, two methods are available:

- Method for fields belonging to iteratives: Used to add conditions to fields that are part of an iterative container.
- Method for fields not belonging to iteratives: Used to add conditions to fields that are not part of an iterative container.

Operator "Between"

Available Methods:

```
addConditionBetween(String pColumnName, Object pValueFrom, Ob-
ject pValueTo)
```

```
addConditionBetween(String pIterativeName, String pColumnName,
Object pValueFrom, Object pValueTo)
```

Example for a non-iterative field:

```
// Non-iterative field
// Adds a condition so that the qtAnualRev field of the Account
form is between the values 500,000 // and 800,000
xQueryCrite-
ria.addConditionBetween(Account.getQtAnualRevQueryName(),
500000, 800000);
```

Example for an iterative field:

```
// Iterative field
// Adds a condition so that the qtAmount field of the Product
iteration in the Opportunity form is between the values 5,000
and 10,000
xQueryCrite-
ria.addConditionBetween(Opportunity.getProductQueryName(), Op-
portunity.Product.getQtAmountQueryName(), 5000, 10000);
```

Operator "Greater Than", "Greater Than or Equal To"

The distinction between the use of the operator "greater than or equal to" and the operator "greater than" is made through the last parameter of the corresponding method.
Possible values for this parameter are:

- True: The operator used is "greater than or equal to" (>=).
- False The operator used is "greater than" (>=).

Available Methods:

```
addConditionGreater(String pColumnName, Object pValue, boolean
useGreaterOrEqual)
```

```
addConditionGreater(String pIterativeName, String pColumnName,
Object pValue, boolean useGreaterOrEqual)
```

Examples for a non-iterative field:

```
// Non-iterative field
// Adds a condition so that the qtAnualRev field of the Account
form is greater than 500,000
xQueryCrite-
ria.addConditionGreater(Account.getQtAnualRevQueryName(),
500000, false);
```

```
// Non-iterative field
// Adds a condition so that the qtAnualRev field of the Account
form is greater than or equal to 500,000
xQueryCrite-
ria.addConditionGreater(Account.getQtAnualRevQueryName(),
500000, true);
```

Examples for an iterative field:

```
// Iterative field
// Adds a condition so that the qtAmount field of the Product
iteration in the Opportunity form is
// greater than 5,000
addConditionGreater(Opportunity.getProductQueryName(), Opportu-
nity.Product.getQtAmountQueryName(), 5000, false);
```

```
// Iterative field
// Adds a condition so that the qtAmount field of the Product
iteration in the Opportunity form is
greater than or equal to 5,000
xQueryCrite-
ria.addConditionGreater(Opportunity.getProductQueryName(), Op-
portunity.Product.getQtAmountQueryName(), 5000, True);
```

Operator “Less Than" or “Less Than or Equal To”

The distinction between the use of the operator "less" and the operator "less than or equal to" is made through the last parameter of the corresponding method. Possible values for this parameter are:

- True: The operator used is "less than or equal to" (<=).
- False The operator used is "less than" (<).

Available Methods:

```
addConditionLower(String pColumnName, Object pValue, boolean
useLowerOrEqual)
```

```
addConditionLower(String pIterativeName, String pColumnName,
Object pValue, boolean useLowerOrEqual)
```

Examples for a non-iterative field:

```
// Non-iterative field
// Adds a condition so that the qtAnualRev field of the Account
form is less than 500,000
xQueryCrite-
ria.addConditionLower(Account.getQtAnualRevQueryName(), 500000,
false);
```

```
// Non-iterative field
// Adds a condition so that the qtAnualRev field of the Account
```

```
form is less than or equal to 500,000
xQueryCrite-
ria.addConditionLower(Account.getQtAnualRevQueryName(), 500000,
true);
```

Examples for an iterative field:

```
// Iterative field
// Adds a condition so that the qtAmount field of the Product
iteration in the Opportunity form is
// less than 5,000
xQueryCrite-
ria.addConditionLower(Opportunity.getProductQueryName(), Oppor-
tunity.Product.getQtAmountQueryName(), 5000, false);
```

```
// Iterative field
// Adds a condition so that the qtAmount field of the Product
iteration in the Opportunity form is
less than or equal to 5,000
xQueryCrite-
ria.addConditionLower(Opportunity.getProductQueryName(), Oppor-
tunity.Product.getQtAmountQueryName(), 5000, true);
```

Operator "Like"

Available Methods:

```
addConditionLike(String pColumnName, String pValue)
```

```
addConditionLike(String pIterativeName, String pColumnName,
String pValue)
```

To use the "Like" condition in a SQL query, it is necessary to send the entire search pattern as a String type. This includes the "%" characters that are used as wildcards in the pattern.

Example for a non-iterative field:

```
// Non-iterative field
// Adds a condition so that the Account form's dsCompany field
matches the pattern '%ACME%'
xQueryCrite-
ria.addConditionLike(Account.getDsCompanyQueryName(),"%ACME%");
```

Example for an iterative field:

```
// Iterative field
// Adds a condition for the dsEmail field of the iterative
eMailLine of the Account form to meet
// with the pattern '%@gmail.com%'
xQueryCrite-
ria.addConditionLike(Account.getEMailLineQueryName(), Ac-
count.EMailLine.getDsEmailQueryName(), "%@gmail.com%");
```

Operator “Different”

Available Methods:

```
addConditionNotEquals(String pColumnName, Object pValue)
```

```
addConditionNotEquals(String pIterativeName, String pCo-
lumnName, Object pValue)
```

Example for a non-iterative field:

```
// Non-iterative field
// Adds a condition so that the Account form's dsCompany field
is not equal to "ACME"
xQueryCrite-
ria.addConditionNotEquals(Account.getDsCompanyQueryName(),"ACME
");
```

Example for an iterative field:

```
// Iterative field
// Adds a condition so that the dsEmail field of the iterative
eMailLine of the Account form is not
// equal to "example@gmail.com"
xQueryCrite-
ria.addConditionNotEquals(Account.getEMailLineQueryName(), Ac-
count.EMailLine.getDsEmailQueryName(), "example@gmail.com");
```

Operator "Equal"

Available Methods:

```
addConditionEquals(String pColumnName, Object pValue)
```

```
addConditionEquals(String pIterativeName, String pColumnName,
Object pValue)
```

Example for a non-iterative field:

```
// Non-iterative field
// Adds a condition so that the Account form's dsCompany field
is equal to "ACME"
xQueryCrite-
ria.addConditionNotEquals(Account.getDsCompanyQueryName(),"ACME
");
```

Example for an iterative field:

```
// Iterative field
// Adds a condition so that the dsEmail field of the iterative
eMailLine of the Account form is equal to // "exam-
ple@gmail.com"
xQueryCrite-
ria.addConditionEquals(Account.getEMailLineQueryName(), Ac-
```

```
    count.EMailLine.getDsEmailQueryName(), "example@gmail.com");
```

Operator "Not Null"

Available Methods:

```
    addConditionNotNull(String pColumnName)
```

```
    addConditionNotNull(String pIterativeName, String pColumnName)
```

Example for a non-iterative field:

```
    // Non-iterative field
    // Adds a condition so that the Account form's cdStatus field
    is not null
    xQueryCrite-
    ria.addConditionNotNull(Account.getCdStatusQueryName());
```

Example for an iterative field:

```
    // Iterative field
    // Adds a condition so that the dsEmail field of the iterative
    eMailLine of the Account form is not
    // null
    xQueryCrite-
    ria.addConditionNotNull(Account.getEMailLineQueryName(), Ac-
    count.EMailLine.getDsEmailQueryName(),;
```

Operator "Is Null"

Available Methods:

```
    addConditionNullable(String pColumnName)
```

```
```

```
addConditionNullable(String pIterativeName, String pColumnName)
```

Example for a non-iterative field:

```
// Non-iterative field
// Adds a condition to check that the Account form's cdStatus
field is null
xQueryCrite-
ria.addConditionNullable(Account.getCdStatusQueryName());
```

Example for an iterative field:

```
// Iterative field
// Adds a condition to check that the dsEmail field of the ite-
rative eMailLine of the Account form // is null
xQueryCrite-
ria.addConditionNullable(Account.getEMailLineQueryName(), Ac-
count.EMailLine.getDsEmailQueryName(),;
```

Operator "In"

Available Methods:

```
addConditionIN(String pColumnName, List pValues)
```

```
addConditionIN(String pIterativeName, String pColumnName,
String pValues)
```

Example for a non-iterative field:

```
// Non-iterative field
// Adds a condition that ensures that the dsIndustry field of
the Account form is within the set of included values
// in xElements (4 and 6)
List xElements = new ArrayList();
```

```
xElements.add(4);
xElements.add(6);
xQueryCriteria.addConditionIN(Account.getDsIndustryQueryName(),
xElements);
```

Example for an iterative field:

```
// Iterative field
// Creates a list containing the countries to filter (in this
case, only "Argentina")
List xCountries = new ArrayList();
xCountries.add("Argentina");
// Creates a new instance of QueryCriteria to add the condition
xQueryCriteria = new QueryCriteria();
// Adds the condition that filters the dsCountry field of the
LsAddress iterative in the Account form
// is in the set of values included in xCountries. In this ca-
se, Argentina.
xQueryCriteria.addConditionIN(Account.getLsAddressQueryName(),
Account.LsAddress.getDsCountryQueryName(), xCountries);
```

Operator "Not In"

Available Methods:

```
addConditionNOTIN(String pColumnName, List pValues)
```

```
addConditionNOTIN(String pIterativeName, String pColumnName,
List pValues)
```

Example for a non-iterative field:

```
// Non-iterative field
// Adds a condition that ensures that the dsIndustry field of
the Account form is not within the set of included values
// in xElements (4 and 6)
```

```
List xElements = new ArrayList();
xElements.add(4);
xElements.add(6);
xQueryCrite-
ria.addConditionNOTIN(Account.getDsIndustryQueryName(), xEle-
ments);
```

Example for an iterative field:

```
// Iterative field
// Creates a list containing the countries to filter (in this
case, only "Argentina")
List xCountries = new ArrayList();
xCountries.add("Argentina");
// Creates a new instance of QueryCriteria to add the condition
xQueryCriteria = new QueryCriteria();
// Adds a condition to verify that the dsCountry field of the
LsAddress iterative in the Account form
// is not within the set of values included in xCountries. In
this case, Argentina.
xQueryCrite-
ria.addConditionNOTIN(Account.getLsAddressQueryName(), Ac-
count.LsAddress.getDsCountryQueryName(), xCountries);
```

5. Grouping

The use of grouping is useful to combine results with functions such as COUNT or SUM. It allows grouping the retrieved results according to the indicated field or column.

Available Methods:

```
addGroupBy(String pColumnName)
```

```
addGroupBy(String pIterativeName, String pColumnName)
```

Example for a non-iterative field:

```
// Non-iterative field
// Groups by the cdStatus field of the Account form
xQueryCriteria.addGroupBy(Account.getCdStatusQueryName());
```

Example for an iterative field:

```
// Iterative field
// Groups by the dsCountry field of the lsAddress iterative of
the Account form
 QueryCriteria.addGroupBy(Account.getLsAddressQueryName(), Ac-
count.LsAddress.getDsCountryQueryName());
```

6. Sorting

A sort order can be added to query results, either in ascending or descending order. This criterion is indicated by a boolean value in the last parameter of the available methods. Possible values are:

- True: Uses descending order.
- False Uses ascending order.

Available Methods:

```
addOrderCriteria(String pColumnName, boolean pOrderDesc)
```

```
addOrderCriteria(String pIterativeName, String pColumnName,
boolean pOrderDesc)
```

Example for a non-iterative field:

```
// Non-iterative field
// Sorts by the qtAnualRev field of the Account form in descen-
ding order
xQueryCrite-
ria.addOrderCriteria(Account.getQtAnualRevQueryName());
```

Example for an iterative field:

```
// Iterative field
// Sorts by the dsCountry field of the lsAddress iterative of
the Account form in ascending order
QueryCriteria.addOrderCriteria(Account.getLsAddressQueryName(),
Account.LsAddress.getDsCountryQueryName(), true);
```

7. Add Functions

Aggregation functions such as COUNT, SUM, or AVG can be added to the QueryCriteria object. These functions must be applied before using the createSelect or createSelectWithIterative methods.

"Count" function

Available Methods:

```
addCount(String pColumnName)
```

```
addCount(String pIterativeName, String pColumnName)
```

Example for a non-iterative field:

```
// Non-iterative field
// Adds the COUNT function to the cdStatus field of the Account
form. COUNT is used
// to determine the number of accounts grouped by state
QueryCriteria xQueryCriteria = new QueryCriteria();
List<String> xList = new ArrayList<>();

xList.add(Account.getCdStatusQueryName());
xQueryCriteria.createSelect(xList);
xQueryCriteria.addCount(Cuenta.getCdStatusQueryName());
xQueryCriteria.addGroupBy(Account.getCdStatusQueryName());
```

“SUM” function

Available Methods:

```
addSum(String pColumnName)
```

```
addSum(String pIterativeName, String pColumnName)
```

Example for a non-iterative field:

```
// Non-iterative field
// Adds the SUM function to the qtAnualRev field of the Account
form. The SUM function is used to
// determine the sum of the annual earnings of all accounts
QueryCriteria xQueryCriteria = new QueryCriteria();
xQueryCriteria.createSelect(new ArrayList());
xQueryCriteria.addSum(Account.getQtAnualRevQueryName());
```

“AVG” function

Available Methods:

```
addAVG(String pColumnName)
```

```
addAVG(String pIterativeName, String pColumnName)
```

Example for a non-iterative field:

```
// Non-iterative field
Adds the AVG function to the qtAnualRev field of the Account
form. The AVG function is used to determine
// the average of the annual earnings of all accounts.
QueryCriteria xQueryCriteria = new QueryCriteria();
String xQTANUALREV = Account.getQtAnualRevQueryName();
xQueryCriteria.addAVG(xQTANUALREV);
```

```
xQueryCriteria.createSelect(new ArrayList<>());
xQueryResult = xAccountService.executeQuery(new Acount(),
xQueryCriteria);
```

Example for an iterative field:

```
// Iterative field
// Adds the AVG function for the qtSalesPrice field of the
Items iterative of the Sales Note form. The AVG function is
used to determine the average selling prices of the items pre-
sent in the Sales Note with dsNumber = 1.
xQueryCriteria = new QueryCriteria();
SalesNoteService xSalesNoteService = new SalesNoteServi-
ce(getApiClient());
String xSalesPrice = Sa-
lesNote.Items.getQtSalesPriceQueryName();
xQueryCriteria.addAVG(SalesNotes.getItemsQueryName(), xSales-
Price);
xQueryCrite-
ria.addConditionEquals(SalesNotate.getDsNumberQueryName(),1);
HashMap<String, List<String>> xMap = new HashMap<>();
xMap.put(SalesNote.getItemsQueryName(), new ArrayList<>());
xQueryCriteria.createSelectWithIterative(new ArrayList<>(),
xMap);
xQueryResult = xSaleNoteService.executeQuery(new SalesNote(),
xQueryCriteria);
xAVGList = xQueryResult.getAVGResult(xSalePrice);
```

8. Join between entities

Available Method:

```
addJOIN(String pColumnName, FormInstance pObjectRelation,
String pKeyObjectRelation)
```

Below is an example that gets the name of all accounts and the tickets associated with each of them. For each ticket, the ID and title are retrieved.

```
// Instantiates the QueryCriteria object
QueryCriteria xQueryCriteria =  new QueryCriteria();

// Creates a list of fields to select
List xList = new ArrayList<>();

// Creates instances of the entities
Account xAccount = new Account();
Ticket xTicket = new Ticket();

// Creates an instance of the Account service using the API
client
AccountService xAccountService = new AccountServi-
ce(getApiClient());

// Adds  field names to the selection list
xList.add(Ticket.getIdTicketQueryName());
xList.add(Ticket.getDsTitleQueryName());
xList.add(Account.getDsCompanyQueryName());

// Creates the selection with the specified fields
xQueryCriteria.createSelect(xList);

// Adds the JOIN between Account and Ticket
xQueryCriteria.addJOIN(Account.getIdAccountQueryName(),// Join
field in Account
 xTicket, // Ticket Object (relation with the Ticket)
Ticket.getIdTicketAccountQueryName() // Join field in Tickett
);

// Execute the query using the Account service
QueryResult xQueryResult = xAccountServi-
ce.executeQuery(xAccount, xQueryCriteria);
// Get the query results
List <String> xTicketIds = xQueryRe-
sult.getQueryResult(Ticket.getIdTicketQueryName());
List <String> xTicketTitles = xQueryRe-
sult.getQueryResult(Ticket.getDsTitleQueryName());
List <String>  xComapanyNames = xQueryRe-
sult.getQueryResult(Account.getDsCompanyQueryName());
```

```
// Iterates over the results and records the information
for (int i = 0; i < xTicketIds.size(); i++) {
   log("Account : " + xCompanyNames.get(i) + ". Ticket number:
" + xTicketIds.get(i) + ". Title: " + xTicketTitles.get(i));
}
```

9. Operating on the results

Once an SQL query is executed, the QueryResult object contains the results grouped by the retrieved column (entity field). To interact with these results, the following methods can be used:

First Method: Retrieves the results for a specific column from the SQL query.

```
getQueryResult(String pColumn): List
```

This method allows retrieving the results for a specific column from the SQL query. Receives the name of the field from which to obtain results and returns a list containing the values corresponding to that column for each row retrieved in the query.

```
// Retrieves the corporate name of accounts whose state is "1"

// Creates an instance of the QueryCriteria object to define
the query criteria
QueryCriteria xQueryCriteria = new QueryCriteria();

// Creates an instance of the service to execute the query
AccountService xAccountService = new AccountServi-
ce(getApiClient());

// Creates an instance of the Account entity
Account xAccount = new Account();

// Defines the columns to be retrieved
List xList = new ArrayList <> ();
xList.add(Account.getDsCompanyQueryName()); // Adds the company
name field to the selection list

// Sets the query selection criteria
xQueryCriteria.createSelect(xList);
```

```
// Adds a condition to filter accounts whose state is "1"
xQueryCrite-
ria.addConditionEquals(Account.getCdStatusQueryName(), "1");

// Executes the query and gets the results
QueryResult xQueryResult = xAccountServi-
ce.executeQuery(xAccount, xQueryCriteria);

// Retrieves the list of company names from the query results
List xComapanyNames = xQueryRe-
sult.getQueryResult(Account.getDsCompanyQueryName());
```

As seen in the last line of the example, the results for the required field are obtained. The size of the list corresponds to the number of rows that have met the SQL query criteria.

If more than one field was selected to be retrieved when generating the SQL query, the size of the corresponding lists will always be the same for each field. If, for example, in the "SELECT" statement, in addition to retrieving the dsCompany field, there is also a request to retrieve data from the qtAnualRev field, it is necessary to ask the QueryResult object for the results for each of these fields.

A structure that allows iteration over the elements can be used to operate on each of the elements of the list. For example, using a for loop:

```
 for(int i=0; i < xQtAnualRevResults.size(); i++){
   log("Account name: " + xAccountNameResults.get(i) + ". An-
nual earnings: " + xQtAnualRevResults.get(i));
}
```

As observed, the sizes of the lists match. Therefore, if multiple fields are selected, it is sufficient to generate a single iterator and retrieve each necessary piece of data. These correspond to each row retrieved from the database.

Second Method: Retrieves the results for a specific column from the function "COUNT".

```
getCountResult(String pColumn): List
```

This method must be given the name of the field on which the count was performed. For example:

```
List xComapanyNameCount = xQueryRe-
```

```
sult.getCountResult(Account.getDsCompanyQueryName());
```

A list is returned because an SQL statement can include more than one COUNT function. It is iterated in the same way as in the example of the first method.

Third Method: Retrieves the results for a specific column from the function "SUM".

```
getSumResult(String pColumn): List
```

Similar to the COUNT function, to get the results of the SUM function, the method must be given the name of the field on which the sum was performed. For example:

```
List xComapanyNameCount = xQueryRe-
sult.getSumResult(Account.getDsCompanyQueryName());
```

A list is returned because an SQL statement can include more than one SUM function. It is iterated in the same way as in the example of the first method.

Fourth Method: Retrieves the result of the function "AVG".

```
getAVGResult(String pColumn): List
```

Just like with the COUNT and SUM functions, to obtain the results of the AVG function, the method must receive the name of the field on which the AVG function was applied.
For example:

```
List xAVGList = xQueryRe-
sult.getAVGResult(Account.getQtAnnualRevQueryName());
```

A list is returned, and if it is not empty, the result of the applied AVG function is obtained from position 0 of the list.

Fifth Method: Checks if more results have not been returned due to the limits established.

```
getExistsMoreResults(): boolean
```

This method is essential to check if there are more results for the generated SQL query that have not been returned due to limits set, either by default or user-configured values.

If a query with more than 10,000 records retrieved is performed, the way to iterate over them is shown in the following example:

```
// There are more than 10,000 accounts that meet the state con-
dition equal to "1"
// Retrieve the corporate name of the accounts whose state is
"1"

QueryCriteria xQueryCriteria = new QueryCriteria();
AccountService xAccountService = new AccountServi-
ce(getApiClient());
Account xAccount = new Account();

List xList = new ArrayList();
xList.add(Account.getDsCompanyQueryName());
xQueryCriteria.createSelect(xList);
xQueryCrite-
ria.addConditionEquals(Account.getCdStatusQueryName(),"1");

boolean thereareMore = true;
int page = 1; //page| 1
int perPage = 10000; //10,000 records per page

while(thereareMore){
   xQueryCriteria.addConditionLimits(page, perPage);
   QueryResult xQueryResult = xAccountService.executeQuery(
xAccount, xQueryCriteria);
   List xComapanyNameResults = xQueryRe-
sult.getQueryResult(Account.getDsCompanyQueryName());

   for(int i=0; i < xCompanyNameResults.size(); i++){
     log("Account company name: " + xCompanyNameRe-
sults.get(i));
    }
   thereareMore = xQueryResult.getExistsMoreResults();
   if(therearemore) {
     page++;
    }
}
```

10,000 records are set to be retrieved per page, page number 1 is iterated over, the corresponding data is obtained, and the QueryResult object is shown to see if there are any pending results. If so, the page number is incremented to get the results from the next page.

Forms

The model class contains the properties with its getters and setters, while the service class contains the operations performed with the model.

For example, if the "Account (CRM_ACCOUNT)" form from the CRM application is used, when downloading the form Java sources, an account.java file is obtained, representing the model and another accountService.java file with the available operations of the service.

Model Class Content

A form model contains:

- Model constructor.
- Getters and setters of its fields.
- Getters for each field for use in SQL shows from the Deyel SDK.
- Inner classes representing the “Multiple occurrences” containers model, if any.

Specific method of the model class

In addition to the getters and setters, there are specific methods related to fields.

Operation	Description	Parameters
getReferen-ce_”idFileTypeField”()	It allows to retrieve the FileRe-ference object associated with the file field of the instance.	

Data Type

Model properties for a form can have the following data types.

Equivalence between form field data types and Java model is shown.

Control	Data Type	Java
Text	Alphanumeric (length) Alphanumeric Uppercase (len-gth)	String String

Control	Data Type	Java
	Large Alphanumeric	String
	Rich Text	String
Number	Integer	Integer
	Large Integer	Long
	Decimal	Double
Time	Time	java.sql.Time
	Local time	java.sql.Time
Date	Date	java.sql.Date
	Date and Time	java.sql.Timestamp
	Local Date	java.sql.Timestamp
	Date and Local Time	java.sql.Timestamp
Image	Image in Folder	String
	Image in Database	String
File	File in Database	String
Check	Boolean	Boolean

Service Class Content

The service allows to perform the following CRUD (Create, Read, Update, Delete) and search (search) operations, as well as containing specific methods, for example to retrieve files from the form.
In addition, the service class includes a specific method to execute SQL shows on the corresponding model.

Operation	Description	Parameters
create(entity)	Create a form instance.	**entity**: Form model class

Operation	Description	Parameters
read(entity)	Reads a form instance.	**entity**: Form model class
update(entity)	Updates a form instance.	**entity**: Form model class
delete(entity)	Deletes a form instance.	**entity**: Form model class
readCases(entity)	Allows to retrieve the case instances associated with a form instance. Returns a list of cases since a form instance can be related to more than one case.	**entity**: Form model class
bind(entity,case)	Allows to associate a case to a form instance.	**entity**: Form model class **case** : Case model class
shortDescription()	It allows retrieving the short description of the instance sent by parameter.	**formInstance**
get“idField”_description()	It allows retrieving the descriptive value of the indicated field.	**formInstance**
search(searchCriteria)	Allows retrieving the form instances that meet the search criteria.	**searchCriteria**: Class that allows defining a set of criteria and search configuration parameters
executeQuery(entity, queryCriteria)	Executes the generated show on the model using the different available methods.	**entity**: Form model class **queryCriteria**: Class that allows defining an SQL show through a set of criteria.

Examples of Use

The examples use the "Account (CRM_ACCOUNT)" form of the CRM application and each example contains the use of "Account" model class and "AccountService" service class.

1. Service creation

This service is created only once in the rule and is reused in different operations.

```
AccountService accountService = new AccountServi-
ce(getApiClient());
```

2. Create a form instance

An instance of “Account” model class is created, values are assigned to its properties. Using the “AccountService” service class, the new account is saved with the assigned values through the create (account) method.

```
Account account = new Account() ;
account.setNuIdentifNumber("010000123");
account.setDsIndustry("1");                    // 1 = Agriculture
account.setDsSource("7");                      // 7 = Social Networ-
king
account.setDsCompany("Trees Company");
account.setDtOpening((Date) new java.util.Date());
account.setFlLogo(new File("/photo.png"));
account.setCdStatus("1");                      // 1 = Active
Integer accountNumber = accountService.create(account);
log("Account created with account number:"+ accountNumber);
```

3. Read a form instance

To obtain data from an existing account, an "Account" model class instance is created, an identifier chosen through the corresponding setter is indicated and such instance is read using the read(account) method of the “AccountService” service class. The instance contains all its properties which can be queried using the corresponding getters. The example reads the dsCompany, dtOpening, dtStore, and dtLastUpdate properties.

```
Account account = new Account();
Integer readAccountNumber = 50000 ;
account.setIdAccount(readAccountNumber);
account = accountService.read(account) ;

// Once read, continue working
String dscompany = account.getDsCompany();
```

```
Date dtopening = account.getDtOpening();

// Audit data
Timestamp dtStore = account.getDtStore();
Timestamp dtLastUpdate = account.getDtLastUpdate();
```

4. Modify a form instance

To modify data from an existing account, an "Account" model class instance is created, an identifier chosen through the corresponding setter is indicated and such instance is read using the read(account) method of the “AccountService” service class. The dsCompany property is then modified using the corresponding setter and the update(account) method of the service is invoked to update the account.

```
Account account = new Account();
account.setIdAccount(50000);
account = accountService.read(account);

// Once read, update
account.setDsCompany("Trees Company");
accountService.update(account);
```

5. Delete a form instance

To delete an existing account, an instance of the "Account" model class is created, indicating its identifier using the corresponding setter. The delete(account) method of the service is then invoked to delete the instance.

```
Account account = new Account();
account.setIdAccount(50000);
account = accountService.read(account);
accountService.delete(account);
```

6. Use of file and image type fields

To retrieve a file related to a form instance, use the corresponding get+<fieldname>+(entity) method of the "AccountService" service class.
In this example, the getFlLogo(account) method of the "AccountService" service class is invoked, which allows retrieval of the file corresponding to a logo associated with a File object.

```
Account account = new Account();
account.setIdAccount(50000);

// Account reading
account = accountService.read(account);

// Logo image reading
File logo = accoountService.getFlLogo(account);

// Assign a new logo to the account
account.setFlLogo(new File("Path"));
accountService.update(account);
```

7. Use of internal objects

Internal objects represent the set of fields modeled in containers of multiple occurrences.

In this example, two new "Account.PhoneLine" internal objects are instantiated and values are assigned to their nrPhone and tpPhone properties.
A new list of "Account.PhoneLine" objects can be saved with the setPhoneLine(lsPhones) method or by adding two instances of the "Account.PhoneLine" object to the current list.

```
Account.PhoneLine phones= new Account.PhoneLine();
phones.setNrPhone("01149834589");
phones.setTpPhone("Line phone");

Account.PhoneLine phones1= new Account.PhoneLine();
phones1.setNrPhone("0115123987");
phones1.setTpPhone("Cell phone");

List<Account.PhoneLine> lsPhones = new ArrayList();
lsPhones.add(phones);
lsPhones.add(phones1);
account.setPhoneLine(lsPhones);

// Another option is
account.getPhoneLine().add(phones);
account.getPhoneLine().add(phones1);
```

8. Reading of cases associated with form instances

The example retrieves the case associated with a form instance and executes that case.

To use this example in addition to the "Account" (Account) form, the "Account Registration" (Account Registration) process that uses that form must be modeled. The case can be started manually or by Deyel SDK methods.

To read cases associated with a form instance, an instance of the "Account" model class is created, indicating the identifier chosen using the corresponding setter. The "AccountService" service is instantiated and using the readCases(account) method, the "Cases" list is retrieved with the cases associated with the instance with identifier 10.
Having the "Cases" list, it is possible to iterate through it to retrieve some or all of its cases. The "AccountRegistration" service is instantiated and using the execute(accountRegistrationCase) method, the first associated case is executed.

> *Both the form instance and the case instance must be created.*

```
Account account = new Account();
account.setId(10);

AccountService accountService = new AccountServi-
ce(getApiClient());
List<Cases> listOfCases = accountService.readCases(account);

AccountRegistration  accountRegistrationCase = new AccountRe-
gistration(listOfCases.get(0));

AccountRegistrationService accountRegistrationService = Accoun-
tRegistrationService(getApiClient());
accountRegistrationService.execute(accountRegistrationCase);
```

9. Associate a case with a form instance

In the example, a case associated with an "Account" form instance is read to then associate such case to a "Contact" form instance, leaving both instances associated with the same case.

The case associated with the "Account" form instance is read with the readCases(account) method, described in example 8.
The case is associated with the "Contact" form instance, with the bind(contact,accountRegistrationCase) method.

> *A case can only be linked to a form instance if there is a relation between the form and the process to which the case to be associated belongs.*

```
Account account = new Account();
account.setId(10);

AccountService accountService = new AccountServi-
ce(getApiClient());
List<Cases> listOfCases = accountService.readCases(account);

AccountRegistration accountRegistrationCase = (AccountRegistra-
tion) listOfCases.get(0);
AccountRegistrationService accountRegistrationService = Accoun-
tRegistrationService(getApiClient());

Contact contact = new Contact();
contact.setId(20);

ContactService contactService = new ContactServi-
ce(getApiClient());

contactService.bind(contact,accountRegistrationCase);
```

10. Retrieve the code and description of the field value associated with a value list

In order to use this example, the "States" value list should be modeled in **Deyel** and values should have been entered to this list.

Code	Descriptive Value
1	Active
2	Inactive

It is required to relate an "Account" form field with the "States" value list.

> *When a field has a relation defined, the content can use its code or its descriptive value.*

This example retrieves the field code associated with the value list.

```
String stateCode = account.getAccountState();
```

```
//Returns: 1
```

While in the following example, the description of the field associated with the value list is retrieved.

```
String stateDescription = accountServi-
ce.getAccountSate_description(account);
//Returns: Active
```

> *The code is retrieved through the model object and the description is retrieved through the service object.*

11. Updating form instances with commit or rollback

This example declares a variable that represents the service of the object to be updated. By using it, the list of existing instances is retrieved and traversed with the FOR statement. For each of them, the service is redefined, obtaining its own API client, the VAT value is calculated and updated, and an attempt is made to perform a commit command or a rollback command to the connected database, as appropriate.

```
protected void run() throws java.lang.Exception{
    TestFormService xTestFormService = new TestFormServi-
ce(getApiClient());
    SearchCriteria xSearchCriteria = new SearchCriteria();
    List<TestForm> xListResult = xTestFormServi-
ce.search(xSearchCriteria).getResult();
    for(int j = 0; j < xListResult.size(); j++) {
        try {
            xTestFormService = new TestFormServi-
ce(getOwnApiClient());
            TestForm xElement = xListResult.get(j);
            xElement.setValueOfVAT((vatrate * entryamount) /
100);
            xTestFormService.update(xElement);
            ownConnectionCommit();
        } catch (Exception e) {
            ownConnectionRollback();
        }
    }
}
```

Search

Searches on form instances can be performed using the following objects.

Criteria- represents search criteria on form data. It is made up of elements that can be fields and values, connected by operators.

Operator	Description
eq	Equal
neq	Different
gt	Greater
gte	Greater equal
lt	Less
lte	Less equal
between	Between
betweene	Between and admits equals
nbetween	Out of range
like	Contain
nlike	It does not contain
startsWith	Starts with
nstartsWith	It does not start with
in	Included
nin	Not included
endsWith	Ends with
nendsWith	Does not end with
isNull	It is null

Operator	Description
notNull	It is not null

SearchCriteria: an object that groups search criteria (Criteria object), it also allows parameterizing the order of the result, the size of the reading page, and the number of pages to be retrieved. Reading by pages is used since the data volume on a form can be very large.

Form Service: service class of the form on which the search is being carried out. Contains the search(searchCriteria) operation that must receive the SearchCriteria object as a parameter and returns the SearchResult object.
Example:
SearchResult searchResult = accountService.search(searchCriteria);

SearchResult: object that contains the search results. Contains a list of form instances, which are instances of the corresponding model, for example "Account". This object allows to know the total number of reading pages resulting from the search, the size of each page, and the number of pages to retrieve.

Search Examples

1. Search

This example retrieves a list of accounts with active state, where such state corresponds to code "1". A quantity of 15 lines per reading page and the number of pages to retrieve are defined. The results are ordered by company name in ascending order.

```
SearchCriteria searchCriteria = new SearchCriteria();
Criteria criteria1 =
         Criteria.eq("CdStatus","1");
searchCriteria.addCriteria(criteria1);
searchCriteria.setPageSize(15);
searchCriteria.setPage(1);
searchCriteria.addOrderAsc("DsCompany");


SearchResult searchResult =
            accountService.search(searchCriteria);
List<Account> instancesResult = searchResult.getResult();
```

2. Constructor types for search criteria

This example creates a Criteria object and defines different search conditions using the "equal", "different", "greater", "greater equal", "less", "less equal", "between", "between and allows equals", "out of range", "contains", "does not contain", "starts with", "does not start with", "included", "not included", "ends with", "does not end with", "is null", "is not null".

```
// Equal cdStatus= 1 corresponds to an active account
 Criteria criteria = Criteria.eq("CdStatus", "1");

// Different cdStatus= 1 corresponds to an active account
// Criteria criteria =  Criteria.neq ("CdStatus","1");

// Greater
// Criteria criteria = Criteria.gt ("qtScore", 20) ;

// Greater equal
// Criteria criteria = Criteria.gte ("qtScore", 20) ;

// Less
// Criteria criteria = Criteria.It ("qtStore", 50) ;

// Less equal
// Criteria criteria = Criteria.Ite ("qtStore", 50) ;

// Between
// Criteria criteria = Criteria.between ("dtOpening",
// Date.valueOf("01/01/1990"), Date.valueOf("01/01/2020"));

// Between and admits equals
// Criteria criteria = Criteria.between("dtOpening",
// Date.valueOf("01/01/1990"), Date.valueOf("01/01/2020"));

// Out of range
// Format YYYY-MM-DD is used
// SimpleDateFormat format =
//      new SimpleDateFormat("yyyy-MM-dd");
// Date dateFrom =
// new Date (format.parse( "1990-01-01" ).getTime());
// Date dateTo =
//      new Date(format.parse( "2020-01-01" ).getTime());
//
// Criteria criteria =
```

```
// Criteria.nbetween ("dtOpening", dateFrom, dateTo );

// Contains In an internal object
// Criteria criteria = Criteria.like("eMailLine/dsEmail",
// yahoo.com.ar");
```

> ***Deyel*** *uses the data type java.sql.Date for "date" type fields, so the transformation from java.util.Date to java.sql.Date must be done, when it is required to create an object with this type of data.*

```
// Contains
// Criteria criteria = Criteria.like("dsCompany","Inc.");

// It does not contain
// Criteria criteria = Criteria.nlike("dsCompany","Inc.");

// Starts with
// Criteria criteria = Criteria.startsWith("dsCompany","The");

// It does not start with
// Criteria criteria = Criteria.nstartsWith("dsCompany","The");

// Included dsIndustry = 6 is Communications, 8 is Education
// Criteria criteria =
// Criteria.in("dsIndustry", Arrays.asList("6", "8" ));

// Not Included dsIndustry 6 is Communications, 8 is Education
// Criteria criteria =
// Criteria.nin("dsIndustry", Arrays.asList("6", "8" ));

// Ends with
// Criteria criteria =
// Criteria.endsWith("dsCompany","Corporation");

// Does not end with
// Criteria criteria =
// Criteria.nendsWith("dsCompany","Corporation");

// It is null
```

```
// Criteria criteria =
// Criteria.isNull("CdStatus");

// It is not null
// Criteria criteria =
// Criteria.notNull("CdStatus");
```

3. Operations on the results object

The SearchResult object is iterative and as such, it can be scrolled through in different ways.
The example uses a FOR statement to iterate over each of the instances that meet the search conditions of the dsIndustry and qtScore properties, retrieving the idAccount and dsCompany properties.

```
SearchCriteria searchCriteria = new SearchCriteria();
 searchCriteria.setPageSize(15);
 searchCriteria.addOrderDesc("idAccount");

 // Accounts with industry equal to communications
 searchCriteria.addCriteria
    (Criteria.eq("dsIndustry", "6"));

// Accounts with a score greater than 20
 searchCriteria.addCriteria
        (Criteria.gt("qtScore", 20)  );

 // Scrolls through all the accounts in the search result
 String stringData1 = "";
 for (Account account: searchResult) {
     stringData1 += " - " + account.getIdAccount()
                 + " - " + account.getDsCompany();
 }
 // If I only need the first results sheet
 SearchResult<Account> resultFirstPage =
        accountService.search(searchCriteria);
 List<Account> firstList = resultFirstPage .getResult();

 String stringData = "";
 stringData += " Amount of data " + firstList.size();
 stringData += " Pages " + resultFirstPage .getPages();
 stringData += " Actual Page " +
                    resultFirstPage .getCurrentPage();
```

Browsing between Related Entities

In the CRM application, the “Contact (CRM_CONTACT)” form has a relation with the “Account (CRM_ACCOUNT)” form and the related account can be retrieved from a business rule, from a contact.

An instance of the “Contact” model class is created and the account identifier is retrieved using the corresponding getter.
Next, an instance of the "Account" model class is created indicating the identifier retrieved from the contact in the setter, and the account instance is read using the read(myAccount) method of the "AccountService" service class.

```
Contact contact = new Contact() ;
Integer idAccount = contact.getIdCompany();
// Retrieve the id of a contact account
// and I read the account with that id

Account myAccount = new Account();
myAccount.setIdAccount(idAccount);
myAccount = accountService.read(myAccount);
```

Executing SQL Queries

SQL queries are useful when working with large volumes of data. A major advantage of using these queries is that they allow retrieving only the information needed, instead of loading the entire model object.

If the SearchCriteria object is used to search for data, the complete objects for that entity are retrieved, even if only, for example, the identifiers of the instance are needed. This means that more information than necessary is being loaded.
Instead, with the SQL queries available in the Deyel SDK, only the specific data required can be retrieved, making processing faster and more efficient.

Another major advantage of using SQL queries with the Deyel SDK is the ability to perform JOIN operations between entities, a functionality that is not available in SearchCriteria.

Searches for SQL Queries

Searches on SQL queries can be performed using the following objects.

QueryCriteria: object on which the different parts of the SQL show are added. The fields to retrieve from an entity instance can be specified with this object, whether to work or not with iteratives, as well as the search order and grouping, the limit conditions, and other operations detailed in the examples.

Form Service - executeQuery() method: service class of the form on which the search is carried out. Contains the executeQuery(formInstance, queryCriteria) operation, which takes as parameters the model object of the entity being worked with and the QueryCriteria object. Returns a QueryResult object as a result.

Form Model - getter ... QueryName(): model class of the form on which you want to execute the SQL query. It contains getters for each of the modeled fields, and these methods have the particularity of ending with QueryName. They should be used when adding the different parts of the query to be generated.

QueryResult: object that contains the results of the SQL query execution. They are grouped by column (entity field), which means that to interact with the QueryResult object, the field to retrieve should be indicated to get a list that represents each of the rows retrieved after the SQL query.

Methods

Methods for working with the various objects are described below.

Methods to execute with the QueryCriteria object

Method	Description	Parameters
createSelect(pList)	Creates a "SELECT" statement and adds the values in the pList parameter.	Java.util.List type **pList**: : contains each of the fields to be retrieved from the form, excluding the iterative container fields
createSelectWithIterative(pList, pMap)	Creates a "SELECT" statement and adds the values in the pList and pMap parameter.	Java.util.List type **pList**: contains each of the fields to be retrieved from the form, excluding the iterative container fields Java.util.Map type **pMap**: contains, as a key, the iterative identifier. The iterative fields are added as a key value of the map within a java.util.List object
addConditionLimits(pPageNumber,pPerPage)	Adds limit conditions to the SQL statement along with the page to retrieve.	**pPageNumber**: page number to retrieve **pPerPage**: number of records to retrieve on the page. By default, it is 100, and at most, 10000

Method	Description	Parameters
addGroupBy(pColumn)	Adds the GROUP BY clause to the SQL statement.	String **pColumn**: represents the identifier of the form field by which grouping is to be performed
addGroupBy(pIterativeName, pColumn)	Adds the GROUP BY clause to the SQL statement.	String **pIterativeName**: indicates the identifier of the iterative to which the field indicated in pColumn belongs String **pColumn**: represents the identifier of the form field by which grouping is to be performed
addOrderCriteria(pColumn,pOrderDesc)	Adds the ORDER BY clause to the SQL statement.	String **pColumn**: represents the identifier of the form field by which sorting is to be performed Boolean **pOrderDesc**: with value "True" indicates descending order, "False", ascending
addOrderCriteria(pIterativeName,pColumn,pOrderDesc)	Adds the ORDER BY clause to the SQL statement considering iterative fields.	String **pIterativeName**: indicates the identifier of the iterative to which the field indicated in pColumn belongs String **pColumn**: represents the identifier of the form field by which sorting is to be performed Boolean **pOrderDesc**: with value "True" indicates descending order, "False", ascending
addConditionBetween(pColumn,pValueFrom, pValueTo)	Adds the BETWEEN condition to the statement. The value searched will be between the values indicated in the pValueFrom and pValueTo parameters.	String **pColumn**: represents the identifier of the form field Object Type **pValueFrom**: represents the initial value from which the match with the pColumn value is sought Object Type **pValueTo**: repre-

Method	Description	Parameters
		sents the final value from which the match with the pColumn value is sought
addConditionBetween(pIterativeName,pColumn, pValueFrom, pValueTo)	Adds the BETWEEN condition to the statement. The value searched will be between the values indicated in the pValueFrom and pValueTo parameters.	String **pIterativeName**: indicates the identifier of the iterative to which the field indicated in pColumn belongs String **pColumn**: represents the identifier of the form field Object Type **pValueFrom**: represents the initial value from which the match with the pColumn value is sought Object Type **pValueTo**: represents the final value from which the match with the pColumn value is sought
addConditionGreater(pColumn,pValue, useGreaterOrEqual)	Adds the > or >= condition to the SQL statement.	String **pColumn**: represents the identifier of the form field Object Type **pValue**: contains the value being compared for the field contained in pColumn Boolean **useGreaterOrEqual**: indicates whether or not the greater than or equal operator is used. With value "True", >=, "False", > is used
addConditionGreater(pIterativeName, pColumn, pValue, useGreaterOrEqual)	Adds the > or >= condition to the SQL statement.	String **pIterativeName**: indicates the identifier of the iterative to which the field indicated in pColumn belongs String **pColumn**: represents the identifier of the form field Object Type **pValue**: contains the value being compared for the field contained in pColumn Boolean **useGreaterOrEqual**:

Method	Description	Parameters
		indicates whether or not the greater than or equal operator is used. With value "True", >=, "False", > is used
addConditionLike(pColumn, pValue)	Adds the LIKE condition to the SQL statement.	String **pColumn**: represents the identifier of the form field String **pValue**: contains the value for the LIKE condition
addConditionLi-ke(pIterativeName, pColumn, pValue)	Adds the LIKE condition to the SQL statement.	String **pIterativeName**: indica-tes the identifier of the iterative to which the field indicated in pColumn belongs String **pColumn**: represents the identifier of the form field String **pValue**: contains the value for the LIKE condition
addConditionLower(pColumn, pValue, useLowerOrEqual)	Adds the < or <= condition to the SQL statement.	String **pColumn**: represents the identifier of the form field Object Type **pValue**: contains the value being compared for the field contained in pColumn Boolean **useLowerOrEqual**: indicates whether or not the greater than or equal operator is used. With value "True", <=, "False", < is used
addConditionLo-wer(pIterativeName, pColumn, pValue, useLowerOrEqual)	Adds the < or <= condition to the SQL statement.	String **pIterativeName**: indica-tes the identifier of the iterative to which the field indicated in pColumn belongs String **pColumn**: represents the identifier of the form field Object Type **pValue**: contains the value being compared for the field contained in pColumn

Method	Description	Parameters
		Boolean **useLowerOrEqual**: indicates whether or not the greater than or equal operator is used. With value "True", <=, "False", < is used
addConditionNotEquals(pColumn, pValue)	Adds the != condition to the SQL statement.	String **pColumn**: represents the identifier of the form field Object Type **pValue**: contains the value being compared for the field contained in pColumn
addConditionNotEquals(pIterativeName, pColumn, pValue)	Adds the != condition to the SQL statement.	String **pIterativeName**: indicates the identifier of the iterative to which the field indicated in pColumn belongs String **pColumn**: represents the identifier of the form field Object Type **pValue**: contains the value being compared for the field contained in pColumn
addConditionEquals(pColumn, pValue)	Adds the = condition to the SQL statement.	String **pColumn**: represents the identifier of the form field Object Type **pValue**: contains the value being compared for the field contained in pColumn
addConditionEquals(pIterativeName, pColumn, pValue)	Adds the = condition to the SQL statement.	String **pIterativeName**: indicates the identifier of the iterative to which the field indicated in pColumn belongs String **pColumn**: represents the identifier of the form field Object Type **pValue**: contains the value being compared for the field contained in pColumn
addConditionNotNull(pColumn)	Adds the IS NOT NULL condition to the SQL statement.	String **pColumn**: represents the identifier of the form field that is required to be non-null.

Method	Description	Parameters
addConditionNotNull(pIterativeName,pColumn)	Adds the IS NOT NULL condition to the SQL statement.	String **pIterativeName**: indicates the identifier of the iterative to which the field indicated in pColumn belongs String **pColumn**: represents the identifier of the form field that is required to be non-null.
addConditionNullable(pColumn)	Adds the IS NULL condition to the SQL statement.	String **pColumn**: represents the identifier of the form field that is required to be null.
addConditionNullable(pIterativeName,pColumn)	Adds the IS NULL condition to the SQL statement.	String **pIterativeName**: indicates the identifier of the iterative to which the field indicated in pColumn belongs String **pColumn**: represents the identifier of the form field that is required to be null.
addConditionNOTIN(pColumn, pValues)	Adds the NOT IN condition to the SQL statement.	String **pColumn**: represents the identifier of the form field List **pValues**: This parameter is a value list (List) that will be used in the NOT IN clause of the SQL statement. Each value in this list represents a value that should be excluded from the results.
addConditionNOTIN(pIterativeName, pColumn, pValues)	Adds the NOT IN condition to the SQL statement.	String **pIterativeName**: indicates the identifier of the iterative to which the field indicated in pColumn belongs String **pColumn**: represents the identifier of the form field List **pValues**: This parameter is a value list (List) that will be used in the NOT IN clause of the SQL statement. Each value in this list represents a value that should be excluded from

Method	Description	Parameters
		the results.
addConditionIN(pColumn, pValues)	Adds the IN condition to the SQL statement.	String **pColumn**: represents the identifier of the form field List **pValues**: This parameter is a value list (List) that will be used in the IN clause of the SQL statement. Each value in this list represents a value that should be included in the results.
addConditionIN(pIterativeName, pColumn, pValues)	Adds the IN condition to the SQL statement.	String **pIterativeName**: indicates the identifier of the iterative to which the field indicated in pColumn belongs String **pColumn**: represents the identifier of the form field List **pValues**: This parameter is a value list (List) that will be used in the IN clause of the SQL statement. Each value in this list represents a value that should be included in the results.
addJOIN(pColumn, pObjectRelation, pKeyObjectRelation)	Adds the INNER JOIN clause to the SQL statement. A JOIN is performed where the pColumn field is equal to the pKeyObjectRelation field of the pObjectRelation object.	String **pColumn**: represents the identifier of the form field with which the JOIN associated with the field represented by the pKeyObjectRelation field of the pObjectRelation object is sought Object type **pObjectRelation**: represents the model object on which an INNER JOIN is executed String **pKeyObjectRelation**: represents the form field identifier of the pObjectRelation model object
addSum(pColumn)	Adds the SUM function to the SQL statement.	String **pColumn**: represents the identifier of the form field on

Method	Description	Parameters
		which the SUM function is performed
addSum(pIterativeName, pColumn)	Adds the SUM function to the SQL statement.	String **pIterativeName**: indicates the identifier of the iterative to which the field indicated in pColumn belongs String **pColumn**: represents the identifier of the form field on which the SUM function is performed
addCount(pColumn)	Adds the COUNT function to the SQL statement.	String **pColumn**: represents the identifier of the form field on which the COUNT function is performed
addCount(pIterativeName, pColumn)	Adds the COUNT function to the SQL statement.	String **pIterativeName**: indicates the identifier of the iterative to which the field indicated in pColumn belongs String **pColumn**: represents the identifier of the form field on which the COUNT function is performed
addAVG(pColumn)	Adds the AVG function to the SQL statement.	String **pColumn**: represents the identifier of the form field on which the AVG function is performed
addAVG(pIterativeName, pColumn)	Adds the AVG function to the SQL statement.	String **pIterativeName**: indicates the identifier of the iterative to which the field indicated in pColumn belongs String **pColumn**: represents the identifier of the form field on which the AVG function is performed

Methods to execute with the QueryCriteria object

Method	Description	Parameters
getQueryResult(pColumn): List	Returns a list containing the values corresponding to the field specified by pColumn.	String **pColumn**: represents the identifier of the form field from which the retrieved values are to be obtained
getCountResult(pColumn):List	Returns a list containing the results of the COUNT functions for the field specified by pColumn.	String **pColumn**: represents the identifier of the form field for which the count was performed.
getSumResult(pColumn): List	Returns a list containing the results of the SUM functions for the field specified by pColumn.	String **pColumn**: represents the identifier of the form field for which the sum was performed.
getAVGResult(pColumn): List	Returns a list with the average (AVG) corresponding to the field indicated in pColumn.	String **pColumn**: represents the identifier of the form field from which the retrieved rows are to be obtained
getExistsMoreResults(): boolean	Returns a boolean value indicating whether there are more results that meet the conditions of the executed SQL statement and that can be processed.	

Method to execute with the Form Service object

Method	Description	Parameters
executeQuery(formInstance, queryCriteria): QueryResult	Executes an SQL statement based on the specified criteria and returns a QueryResult object with the corresponding results.	**formInstance**: object of the entity model being worked with and on which the SQL statement is executed. **queryCriteria**: QueryCriteria object that contains the SQL statement settings, configured through the methods available in that class.

Execution Examples

1. Search

This example retrieves a list of account identifiers with active state, where the active state corresponds to code "1". It is defined that there are 15 records per reading page, and the number of pages to retrieve is specified. The results are ordered by company name in ascending order.

```
//Creating QueryCriteria object
QueryCriteria xQueryCriteria = new QueryCriteria();

//Creating model object
Account xAccount = new Account();

//Creating the service for the entity
AccountService xAccountService = new AccountServi-
ce(getApiClient());

//Selecting the identifier as the column to retrieve
List<String> xColumns = new ArrayList<>();
xColumns.add(Account.getIdAccountQueryName());

//Configuring the query to select the specified columns.
xQueryCriteria.createSelect(xColumns);

//Add conditions to the query
xQueryCrite-
ria.addConditionEquals(Account.getCdStatusQueryName(), "1");
xQueryCriteria.addConditionLimits(15, 1);
xQueryCrite-
ria.addOrderCriteria(Account.getDsCompanyQueryName(), false);

//Executing the query and obtaining the results
QueryResult xQueryResult = xAccountServi-
ce.executeQuery(xAccount, xQueryCriteria);
List instancesResult = xQueryRe-
sult.getQueryResult(Account.getIdAccountQueryName());
```

2. Creating the "SELECT" statement

To create a query, two methods can be used to incorporate the corresponding SELECT statement into the QueryCriteria object, along with the fields to be retrieved from the entity.

First Method

```
createSelect(List<String> xColumns)
```

This method incorporates the "SELECT" statement to the QueryCriteria object and adds each of the entity's fields that have been previously added to the pColumns list. Each field added to the pColumns list must be a non-iterative field.

```
//Selecting the identifier as the column to retrieve
List<String> xColumns = new ArrayList<>();
xColumns.add(Account.getIdAccountQueryName());

//Creating the SELECT statement with the specified columns
xQueryCriteria.createSelect(xColumns);
```

Second Method

```
createSelectWithIterative(List<String> pFilters, Map<String,
List<String>> pMap)
```

This method allows adding fields that belong to an iterative container of the entity to the QueryCriteria object. Fields that do not belong to iteratives should be included in the pFilters list, while fields that belong to iteratives should be added to a separate list. This list must then be added to a variable of type "HashMap", using the identifier of the iterative to which the fields belong as the key.
To work with the iterative fields and add them to the corresponding variable, it is necessary to use the iterative class, which is included within the form model object.

```
//Selecting the identifier as the column to retrieve
List<String> xColumns = new ArrayList<>();
xColumns.add(Account.getIdAccountQueryName());

//Selecting the dsEmail field from the eMailLine iterative con-
tainer
Map<String,List<String>> xMap = new HashMap<>();
List<String> xIterativeColumns = new ArrayList<>();
xIterativeColumns.add(Account.EMailLine.getDsEmailQueryName());
xMap.put(Account.getEMailLineQueryName(), xIterativeColumns);
```

```
//Creating the select
xQueryCriteria.createSelectWithIterative(xColumns, xMap);
```

3. Limits of results

When working with SQL queries in the Deyel SDK, it is important to note that there is a limit to the number of results returned. This limit can be adjusted by the following method:

```
addConditionLimits(int pPageNumber, int pPerPage)
```

In this method, the first parameter, pPageNumber, indicates the number of the page to be retrieved, while the second parameter, pPerPage, specifies the maximum number of records per page.

> *By default, Deyel retrieves up to 100 records per page, and the maximum limit of records to retrieve per page is 10,000. If a higher value is set, the SQL query execution will fail.*

4. Search conditions

Various search conditions can be added to the QueryCriteria object to perform SQL queries.

This example creates a QueryCriteria object and defines different search conditions using the "equal", "different", "greater", "greater or equal", "less", "less equal", "between", "like", "is null" and "is not null", " in" and "not in" with the corresponding addCondition method.

For each type of condition, two methods are available:

- Method for fields belonging to iteratives: Used to add conditions to fields that are part of an iterative container.
- Method for fields not belonging to iteratives: Used to add conditions to fields that are not part of an iterative container.

Operator "Between"

Available Methods:

```
addConditionBetween(String pColumnName, Object pValueFrom, Object pValueTo)
```

```
addConditionBetween(String pIterativeName, String pColumnName,
Object pValueFrom, Object pValueTo)
```

Example for a non-iterative field:

```
// Non-iterative field
// Adds a condition so that the qtAnualRev field of the Account
form is between the values 500,000 // and 800,000
xQueryCrite-
ria.addConditionBetween(Account.getQtAnualRevQueryName(),
500000, 800000);
```

Example for an iterative field:

```
// Iterative field
// Adds a condition so that the qtAmount field of the Product
iteration in the Opportunity form is between the values 5,000
and 10,000
xQueryCrite-
ria.addConditionBetween(Opportunity.getProductQueryName(), Op-
portunity.Product.getQtAmountQueryName(), 5000, 10000);
```

Operator "Greater Than", "Greater Than or Equal To"

The distinction between the use of the operator "greater than or equal to" and the operator "greater than" is made through the last parameter of the corresponding method.
Possible values for this parameter are:

- True: The operator used is "greater than or equal to" (>=).
- False The operator used is "greater than" (>=).

Available Methods:

```
addConditionGreater(String pColumnName, Object pValue, boolean
useGreaterOrEqual)
```

```
addConditionGreater(String pIterativeName, String pColumnName,
```

```
    Object pValue, boolean useGreaterOrEqual)
```

Examples for a non-iterative field:

```
    // Non-iterative field
    // Adds a condition so that the qtAnualRev field of the Account
    form is greater than 500,000
    xQueryCrite-
    ria.addConditionGreater(Account.getQtAnualRevQueryName(),
    500000, false);
```

```
    // Non-iterative field
    // Adds a condition so that the qtAnualRev field of the Account
    form is greater than or equal to 500,000
    xQueryCrite-
    ria.addConditionGreater(Account.getQtAnualRevQueryName(),
    500000, true);
```

Examples for an iterative field:

```
    // Iterative field
    // Adds a condition so that the qtAmount field of the Product
    iteration in the Opportunity form is
    // greater than 5,000
    addConditionGreater(Opportunity.getProductQueryName(), Opportu-
    nity.Product.getQtAmountQueryName(), 5000, false);
```

```
    // Iterative field
    // Adds a condition so that the qtAmount field of the Product
    iteration in the Opportunity form is
    greater than or equal to 5,000
    xQueryCrite-
    ria.addConditionGreater(Opportunity.getProductQueryName(), Op-
    portunity.Product.getQtAmountQueryName(), 5000, True);
```

Operator “Less Than" or “Less Than or Equal To”

The distinction between the use of the operator "less" and the operator "less than or equal to" is made through the last parameter of the corresponding method. Possible values for this parameter are:

- True: The operator used is "less than or equal to" (<=).
- False The operator used is "less than" (<).

Available Methods:

```
addConditionLower(String pColumnName, Object pValue, boolean
useLowerOrEqual)
```

```
addConditionLower(String pIterativeName, String pColumnName,
Object pValue, boolean useLowerOrEqual)
```

Examples for a non-iterative field:

```
// Non-iterative field
// Adds a condition so that the qtAnualRev field of the Account
form is less than 500,000
xQueryCrite-
ria.addConditionLower(Account.getQtAnualRevQueryName(), 500000,
false);
```

```
// Non-iterative field
// Adds a condition so that the qtAnualRev field of the Account
form is less than or equal to 500,000
xQueryCrite-
ria.addConditionLower(Account.getQtAnualRevQueryName(), 500000,
true);
```

Examples for an iterative field:

```
// Iterative field
// Adds a condition so that the qtAmount field of the Product
```

```
    iteration in the Opportunity form is
    // less than 5,000
    xQueryCrite-
    ria.addConditionLower(Opportunity.getProductQueryName(), Oppor-
    tunity.Product.getQtAmountQueryName(), 5000, false);
```

```
    // Iterative field
    // Adds a condition so that the qtAmount field of the Product
    iteration in the Opportunity form is
    less than or equal to 5,000
    xQueryCrite-
    ria.addConditionLower(Opportunity.getProductQueryName(), Oppor-
    tunity.Product.getQtAmountQueryName(), 5000, true);
```

Operator "Like"

Available Methods:

```
    addConditionLike(String pColumnName, String pValue)
```

```
    addConditionLike(String pIterativeName, String pColumnName,
    String pValue)
```

To use the "Like" condition in a SQL query, it is necessary to send the entire search pattern as a String type. This includes the "%" characters that are used as wildcards in the pattern.

Example for a non-iterative field:

```
    // Non-iterative field
    // Adds a condition so that the Account form's dsCompany field
    matches the pattern '%ACME%'
    xQueryCrite-
    ria.addConditionLike(Account.getDsCompanyQueryName(),"%ACME%");
```

Example for an iterative field:

```
// Iterative field
// Adds a condition for the dsEmail field of the iterative
eMailLine of the Account form to meet
// with the pattern '%@gmail.com%'
xQueryCrite-
ria.addConditionLike(Account.getEMailLineQueryName(), Ac-
count.EMailLine.getDsEmailQueryName(), "%@gmail.com%");
```

Operator “Different”

Available Methods:

```
addConditionNotEquals(String pColumnName, Object pValue)
```

```
addConditionNotEquals(String pIterativeName, String pCo-
lumnName, Object pValue)
```

Example for a non-iterative field:

```
// Non-iterative field
// Adds a condition so that the Account form's dsCompany field
is not equal to "ACME"
xQueryCrite-
ria.addConditionNotEquals(Account.getDsCompanyQueryName(),"ACME
");
```

Example for an iterative field:

```
// Iterative field
// Adds a condition so that the dsEmail field of the iterative
eMailLine of the Account form is not
// equal to "example@gmail.com"
xQueryCrite-
ria.addConditionNotEquals(Account.getEMailLineQueryName(), Ac-
count.EMailLine.getDsEmailQueryName(), "example@gmail.com");
```

Operator "Equal"

Available Methods:

```
addConditionEquals(String pColumnName, Object pValue)
```

```
addConditionEquals(String pIterativeName, String pColumnName,
Object pValue)
```

Example for a non-iterative field:

```
// Non-iterative field
// Adds a condition so that the Account form's dsCompany field
is equal to "ACME"
xQueryCrite-
ria.addConditionNotEquals(Account.getDsCompanyQueryName(),"ACME
");
```

Example for an iterative field:

```
// Iterative field
// Adds a condition so that the dsEmail field of the iterative
eMailLine of the Account form is equal to // "exam-
ple@gmail.com"
xQueryCrite-
ria.addConditionEquals(Account.getEMailLineQueryName(), Ac-
count.EMailLine.getDsEmailQueryName(), "example@gmail.com");
```

Operator "Not Null"

Available Methods:

```
addConditionNotNull(String pColumnName)
```

```
addConditionNotNull(String pIterativeName, String pColumnName)
```

Example for a non-iterative field:

```
// Non-iterative field
// Adds a condition so that the Account form's cdStatus field
is not null
xQueryCrite-
ria.addConditionNotNull(Account.getCdStatusQueryName());
```

Example for an iterative field:

```
// Iterative field
// Adds a condition so that the dsEmail field of the iterative
eMailLine of the Account form is not
// null
xQueryCrite-
ria.addConditionNotNull(Account.getEMailLineQueryName(), Ac-
count.EMailLine.getDsEmailQueryName(),;
```

Operator "Is Null"

Available Methods:

```
addConditionNullable(String pColumnName)
```

```
addConditionNullable(String pIterativeName, String pColumnName)
```

Example for a non-iterative field:

```
// Non-iterative field
// Adds a condition to check that the Account form's cdStatus
field is null
xQueryCrite-
ria.addConditionNullable(Account.getCdStatusQueryName());
```

```

```

Example for an iterative field:

```
    // Iterative field
    // Adds a condition to check that the dsEmail field of the ite-
    rative eMailLine of the Account form // is null
    xQueryCrite-
    ria.addConditionNullable(Account.getEMailLineQueryName(), Ac-
    count.EMailLine.getDsEmailQueryName(),;
```

Operator "In"

Available Methods:

```
    addConditionIN(String pColumnName, List pValues)
```

```
    addConditionIN(String pIterativeName, String pColumnName,
    String pValues)
```

Example for a non-iterative field:

```
    // Non-iterative field
    // Adds a condition that ensures that the dsIndustry field of
    the Account form is within the set of included values
    // in xElements (4 and 6)
    List xElements = new ArrayList();
    xElements.add(4);
    xElements.add(6);
    xQueryCriteria.addConditionIN(Account.getDsIndustryQueryName(),
    xElements);
```

Example for an iterative field:

```
    // Iterative field
    // Creates a list containing the countries to filter (in this
```

```
case, only "Argentina")
List xCountries = new ArrayList();
xCountries.add("Argentina");
// Creates a new instance of QueryCriteria to add the condition
xQueryCriteria = new QueryCriteria();
// Adds the condition that filters the dsCountry field of the
LsAddress iterative in the Account form
// is in the set of values included in xCountries. In this ca-
se, Argentina.
xQueryCriteria.addConditionIN(Account.getLsAddressQueryName(),
Account.LsAddress.getDsCountryQueryName(), xCountries);
```

Operator "Not In"

Available Methods:

```
addConditionNOTIN(String pColumnName, List pValues)
```

```
addConditionNOTIN(String pIterativeName, String pColumnName,
List pValues)
```

Example for a non-iterative field:

```
// Non-iterative field
// Adds a condition that ensures that the dsIndustry field of
the Account form is not within the set of included values
// in xElements (4 and 6)
List xElements = new ArrayList();
xElements.add(4);
xElements.add(6);
xQueryCrite-
ria.addConditionNOTIN(Account.getDsIndustryQueryName(), xEle-
ments);
```

Example for an iterative field:

```
```

```
// Iterative field
// Creates a list containing the countries to filter (in this
case, only "Argentina")
List xCountries = new ArrayList();
xCountries.add("Argentina");
// Creates a new instance of QueryCriteria to add the condition
xQueryCriteria = new QueryCriteria();
// Adds a condition to verify that the dsCountry field of the
LsAddress iterative in the Account form
// is not within the set of values included in xCountries. In
this case, Argentina.
xQueryCrite-
ria.addConditionNOTIN(Account.getLsAddressQueryName(), Ac-
count.LsAddress.getDsCountryQueryName(), xCountries);
```

5. Grouping

The use of grouping is useful to combine results with functions such as COUNT or SUM. It allows grouping the retrieved results according to the indicated field or column.

Available Methods:

```
addGroupBy(String pColumnName)
```

```
addGroupBy(String pIterativeName, String pColumnName)
```

Example for a non-iterative field:

```
// Non-iterative field
// Groups by the cdStatus field of the Account form
xQueryCriteria.addGroupBy(Account.getCdStatusQueryName());
```

Example for an iterative field:

```
// Iterative field
// Groups by the dsCountry field of the lsAddress iterative of
```

```
the Account form
 QueryCriteria.addGroupBy(Account.getLsAddressQueryName(), Ac-
count.LsAddress.getDsCountryQueryName());
```

6. Sorting

A sort order can be added to query results, either in ascending or descending order. This criterion is indicated by a boolean value in the last parameter of the available methods. Possible values are:

- True: Uses descending order.
- False Uses ascending order.

Available Methods:

```
addOrderCriteria(String pColumnName, boolean pOrderDesc)
```

```
addOrderCriteria(String pIterativeName, String pColumnName,
boolean pOrderDesc)
```

Example for a non-iterative field:

```
// Non-iterative field
// Sorts by the qtAnualRev field of the Account form in descen-
ding order
xQueryCrite-
ria.addOrderCriteria(Account.getQtAnualRevQueryName());
```

Example for an iterative field:

```
// Iterative field
// Sorts by the dsCountry field of the lsAddress iterative of
the Account form in ascending order
QueryCriteria.addOrderCriteria(Account.getLsAddressQueryName(),
Account.LsAddress.getDsCountryQueryName(), true);
```

7. Add Functions

Aggregation functions such as COUNT, SUM, or AVG can be added to the QueryCriteria object. These functions must be applied before using the createSelect or createSelectWithIterative methods.

“Count” function

Available Methods:

```
addCount(String pColumnName)
```

```
addCount(String pIterativeName, String pColumnName)
```

Example for a non-iterative field:

```
// Non-iterative field
// Adds the COUNT function to the cdStatus field of the Account
form. COUNT is used
// to determine the number of accounts grouped by state
QueryCriteria xQueryCriteria = new QueryCriteria();
List<String> xList = new ArrayList<>();

xList.add(Account.getCdStatusQueryName());
xQueryCriteria.createSelect(xList);
xQueryCriteria.addCount(Cuenta.getCdStatusQueryName());
xQueryCriteria.addGroupBy(Account.getCdStatusQueryName());
```

“SUM” function

Available Methods:

```
addSum(String pColumnName)
```

```
addSum(String pIterativeName, String pColumnName)
```

Example for a non-iterative field:

```
// Non-iterative field
// Adds the SUM function to the qtAnualRev field of the Account
form. The SUM function is used to
// determine the sum of the annual earnings of all accounts
QueryCriteria xQueryCriteria = new QueryCriteria();
xQueryCriteria.createSelect(new ArrayList());
xQueryCriteria.addSum(Account.getQtAnualRevQueryName());
```

“AVG” function

Available Methods:

```
addAVG(String pColumnName)
```

```
addAVG(String pIterativeName, String pColumnName)
```

Example for a non-iterative field:

```
// Non-iterative field
Adds the AVG function to the qtAnualRev field of the Account
form. The AVG function is used to determine
// the average of the annual earnings of all accounts.
QueryCriteria xQueryCriteria = new QueryCriteria();
String xQTANUALREV = Account.getQtAnualRevQueryName();
xQueryCriteria.addAVG(xQTANUALREV);
xQueryCriteria.createSelect(new ArrayList<>());
xQueryResult = xAccountService.executeQuery(new Acount(),
xQueryCriteria);
```

Example for an iterative field:

```
// Iterative field
// Adds the AVG function for the qtSalesPrice field of the
Items iterative of the Sales Note form. The AVG function is
```

```
used to determine the average selling prices of the items pre-
sent in the Sales Note with dsNumber = 1.
xQueryCriteria = new QueryCriteria();
SalesNoteService xSalesNoteService = new SalesNoteServi-
ce(getApiClient());
String xSalesPrice = Sa-
lesNote.Items.getQtSalesPriceQueryName();
xQueryCriteria.addAVG(SalesNotes.getItemsQueryName(), xSales-
Price);
xQueryCrite-
ria.addConditionEquals(SalesNotate.getDsNumberQueryName(),1);
HashMap<String, List<String>> xMap = new HashMap<>();
xMap.put(SalesNote.getItemsQueryName(), new ArrayList<>());
xQueryCriteria.createSelectWithIterative(new ArrayList<>(),
xMap);
xQueryResult = xSaleNoteService.executeQuery(new SalesNote(),
xQueryCriteria);
xAVGList = xQueryResult.getAVGResult(xSalePrice);
```

8. Join between entities

Available Method:

```
addJOIN(String pColumnName, FormInstance pObjectRelation,
String pKeyObjectRelation)
```

Below is an example that gets the name of all accounts and the tickets associated with each of them. For each ticket, the ID and title are retrieved.

```
// Instantiates the QueryCriteria object
QueryCriteria xQueryCriteria =  new QueryCriteria();

// Creates a list of fields to select
List xList = new ArrayList<>();

// Creates instances of the entities
Account xAccount = new Account();
Ticket xTicket = new Ticket();
```

```
// Creates an instance of the Account service using the API
client
AccountService xAccountService = new AccountServi-
ce(getApiClient());

// Adds  field names to the selection list
xList.add(Ticket.getIdTicketQueryName());
xList.add(Ticket.getDsTitleQueryName());
xList.add(Account.getDsCompanyQueryName());

// Creates the selection with the specified fields
xQueryCriteria.createSelect(xList);

// Adds the JOIN between Account and Ticket
xQueryCriteria.addJOIN(Account.getIdAccountQueryName(),// Join
field in Account
 xTicket, // Ticket Object (relation with the Ticket)
Ticket.getIdTicketAccountQueryName() // Join field in Tickett
);

// Execute the query using the Account service
QueryResult xQueryResult = xAccountServi-
ce.executeQuery(xAccount, xQueryCriteria);
// Get the query results
List <String> xTicketIds = xQueryRe-
sult.getQueryResult(Ticket.getIdTicketQueryName());
List <String> xTicketTitles = xQueryRe-
sult.getQueryResult(Ticket.getDsTitleQueryName());
List <String>  xComapanyNames = xQueryRe-
sult.getQueryResult(Account.getDsCompanyQueryName());

// Iterates over the results and records the information
for (int i = 0; i < xTicketIds.size(); i++) {
   log("Account : " + xCompanyNames.get(i) + ". Ticket number:
" + xTicketIds.get(i) + ". Title: " + xTicketTitles.get(i));
}
```

9. Operating on the results

Once an SQL query is executed, the QueryResult object contains the results grouped by the retrieved column (entity field). To interact with these results, the following methods can be used:

First Method: Retrieves the results for a specific column from the SQL query.

```
getQueryResult(String pColumn): List
```

This method allows retrieving the results for a specific column from the SQL query. Receives the name of the field from which to obtain results and returns a list containing the values corresponding to that column for each row retrieved in the query.

```
// Retrieves the corporate name of accounts whose state is "1"

// Creates an instance of the QueryCriteria object to define
the query criteria
QueryCriteria xQueryCriteria = new QueryCriteria();

// Creates an instance of the service to execute the query
AccountService xAccountService = new AccountServi-
ce(getApiClient());

// Creates an instance of the Account entity
Account xAccount = new Account();

// Defines the columns to be retrieved
List xList = new ArrayList <> ();
xList.add(Account.getDsCompanyQueryName()); // Adds the company
name field to the selection list

// Sets the query selection criteria
xQueryCriteria.createSelect(xList);

// Adds a condition to filter accounts whose state is "1"
xQueryCrite-
ria.addConditionEquals(Account.getCdStatusQueryName(), "1");

// Executes the query and gets the results
QueryResult xQueryResult = xAccountServi-
ce.executeQuery(xAccount, xQueryCriteria);
```

```
// Retrieves the list of company names from the query results
List xComapanyNames = xQueryRe-
sult.getQueryResult(Account.getDsCompanyQueryName());
```

As seen in the last line of the example, the results for the required field are obtained. The size of the list corresponds to the number of rows that have met the SQL query criteria.

If more than one field was selected to be retrieved when generating the SQL query, the size of the corresponding lists will always be the same for each field. If, for example, in the "SELECT" statement, in addition to retrieving the dsCompany field, there is also a request to retrieve data from the qtAnualRev field, it is necessary to ask the QueryResult object for the results for each of these fields.

A structure that allows iteration over the elements can be used to operate on each of the elements of the list. For example, using a for loop:

```
 for(int i=0; i < xQtAnualRevResults.size(); i++){
   log("Account name: " + xAccountNameResults.get(i) + ". An-
nual earnings: " + xQtAnualRevResults.get(i));
}
```

As observed, the sizes of the lists match. Therefore, if multiple fields are selected, it is sufficient to generate a single iterator and retrieve each necessary piece of data. These correspond to each row retrieved from the database.

Second Method: Retrieves the results for a specific column from the function "COUNT".

```
getCountResult(String pColumn): List
```

This method must be given the name of the field on which the count was performed. For example:

```
List xComapanyNameCount = xQueryRe-
sult.getCountResult(Account.getDsCompanyQueryName());
```

A list is returned because an SQL statement can include more than one COUNT function. It is iterated in the same way as in the example of the first method.

Third Method: Retrieves the results for a specific column from the function "SUM".

```
getSumResult(String pColumn): List
```

Similar to the COUNT function, to get the results of the SUM function, the method must be given the name of the field on which the sum was performed. For example:

```
List xComapanyNameCount = xQueryRe-
sult.getSumResult(Account.getDsCompanyQueryName());
```

A list is returned because an SQL statement can include more than one SUM function. It is iterated in the same way as in the example of the first method.

Fourth Method: Retrieves the result of the function "AVG".

```
getAVGResult(String pColumn): List
```

Just like with the COUNT and SUM functions, to obtain the results of the AVG function, the method must receive the name of the field on which the AVG function was applied.
For example:

```
List xAVGList = xQueryRe-
sult.getAVGResult(Account.getQtAnnualRevQueryName());
```

A list is returned, and if it is not empty, the result of the applied AVG function is obtained from position 0 of the list.

Fifth Method: Checks if more results have not been returned due to the limits established.

```
getExistsMoreResults(): boolean
```

This method is essential to check if there are more results for the generated SQL query that have not been returned due to limits set, either by default or user-configured values.

If a query with more than 10,000 records retrieved is performed, the way to iterate over them is shown in the following example:

```
// There are more than 10,000 accounts that meet the state con-
dition equal to "1"
// Retrieve the corporate name of the accounts whose state is
"1"
```

```
QueryCriteria xQueryCriteria = new QueryCriteria();
AccountService xAccountService = new AccountServi-
ce(getApiClient());
Account xAccount = new Account();

List xList = new ArrayList();
xList.add(Account.getDsCompanyQueryName());
xQueryCriteria.createSelect(xList);
xQueryCrite-
ria.addConditionEquals(Account.getCdStatusQueryName(),"1");

boolean thereareMore = true;
int page = 1; //page| 1
int perPage = 10000; //10,000 records per page

while(thereareMore){
   xQueryCriteria.addConditionLimits(page, perPage);
   QueryResult xQueryResult = xAccountService.executeQuery(
xAccount, xQueryCriteria);
   List xComapanyNameResults = xQueryRe-
sult.getQueryResult(Account.getDsCompanyQueryName());

   for(int i=0; i < xCompanyNameResults.size(); i++){
     log("Account company name: " + xCompanyNameRe-
sults.get(i));
    }
   thereareMore = xQueryResult.getExistsMoreResults();
   if(therearemore) {
     page++;
    }
}
```

10,000 records are set to be retrieved per page, page number 1 is iterated over, the corresponding data is obtained, and the QueryResult object is shown to see if there are any pending results. If so, the page number is incremented to get the results from the next page.

Cases

The model class contains properties with their getters and setters, while the service class contains operations to be performed on the model.

For example, when downloading the Java sources of the process, if using the "Request Payment" process, a requestPayment.java file is generated with the model methods and also another requestPaymentService.java file with the available service operations.

Model Class Content

A case model contains:

- Model constructor.
- Getters and setters of the case properties.
- Getters to get the related forms.
- Getters to get the activities in progress.
- Getters to get the process corresponding to the case.

Getters and setters of the case properties

The case model class contains a set of getters and setters for the case properties.

Operation	Description	Parameters
getCdCase()	Gets the identifier of the case.	
setCdCase(casenumber)	Allows to set the case identifier value.	String **casenumber**: Case number
getCdState()	Gets the state of the case. Possible values: **ACTIVE** - Active **CANCELLED** Cancelled **ENDEDCASE** - Ended	
getDsCase()	Gets the description of the case.	
getDtEnded()	Gets the end date of the case.	
getDtExpiration()	Gets the expiration date of the case.	
getDtInitiated()	Gets the start date of the case.	
getPriority()	Gets the priority of the case. Possible values: **1** - Urgent	

Operation	Description	Parameters
	2 - High **3** - Medium **4** - Low	
setLastPressedButton(button)	Allows to set the last button pressed.	String **button**: Name of the last button pressed.
getLastPressedButton()	Gets the last button pressed.	
get + <Nombre de la clase del formulario> + Entity()	Gets an instance of the form class related to the process.	
getLsVinculatedForms()	Gets a list of form instances related to the case.	
getVinculatedFormByIdEnti-ty(idEntity)	Gets a form instance related to the case.	String **idEntity**: Identifier of the form instance
getProcess()	Gets the process corresponding to the case.	
getLsExecutedActivities()	Gets the executed activities of the case.	
getLsCurrentActivities()	Gets the activities in progress of the case.	

Getter to get the related forms

Given a case, the values of the form fields related to it can be used, in the same way as if the activities were executed manually.

The model class has a getter for each form related to the process, with a "get + <Nombre de la clase del formulario> + Entity()" structure. Once an instance has been retrieved, it is used in the same way as for forms.

Example
If the "Request Payment" process related to the "Payment" and "Report" forms is modeled, the RequestPayment class has the following methods:

- getPaymentEntity(): Gets an instance of the Payment class that corresponds to the "Payment" form related to the case.
- getReportEntity(): Gets an instance of the Report class that corresponds to the "Report" form related to the case.

> *The case cannot have forms associated with the same class name, even if they belong to different applications.*

Getters to get the activities in progress

Given a case, the list of its activities in progress can be accessed, using the getLsCurrentActivities() case properties method of the model class, which gets a list of "ExecutedActivity" objects with their methods.

Operation	Description	Parameters
getCdActivity()	Gets the code of the activity.	
getDsNameActivity()	Gets the activity name.	
getCdState()	Gets the state of the activity. Possible values: **EXEC** - In execution **CANCELLEDACT** - Cancelled **ENDEDACT** - Ended	
getPriority()	Gets the activity priority. Possible values: **1** - Urgent **2** - High **3** - Medium **4** - Low	
getDtEnded()	Gets the end date of the activity.	
getDtExpiration()	Gets the expiration date of the activity.	
getDtInitiated()	Gets the start date of the activity.	
getCdUserExec()	Gets the code of the user that executed the activity.	
getCdUserInit()	Gets the code of the user that started the activity.	

Operation	Description	Parameters
getNuDurationSeconds()	Gets the duration in seconds of the activity execution.	
getTpActivity()	Gets the activity type. Possible values: **TP_GATEWAY** - Gateway **TP_STANDARD** - Standard **TP_ABSTRACT** - Abstract	
getTpParticipant()	Gets the type of participant for the activity. Possible values: **USER** - User **ORG_UNIT** - Organizational Unit **ROLE** - Role **THING** - Userthing	
getLsExecutedActions()	Gets a list of actions executed by the activity.	

Getter to get the process corresponding to the case

A case process can be obtained through the getProccess() method of the model class, which returns an instance of the "Process" class with its methods.

Operation	Description	Parameters
getCdProcess()	Gets the code of the process.	
getCdVersion()	Gets the version of the process.	
getIdApplication()	Gets the application identifier of the process.	
getCdFirstActivity()	Gets an Activity object with the first process activity,	
getDsComment()	Gets the comment of the process.	

Operation	Description	Parameters
getDsDescription()	Gets the description of the process.	
getDsName()	Gets the name of the process.	

Service Class Content

The service allows the following operations to be performed on the cases:

Operation	Description	Parameters
startCase(case) startCase(case, pressedButton)	Starts a case, executing the initial activity of the related process. Returns the code of the created case.	Case **case**: Model of the case to start. Related form fields must have values Case **case**: Model of the case to start. Related form fields must have values String **pressedButton**: Name of the button that is pressed to advance in the activity.
execute(case)	Executes the current activity of a case. Returns the state of the updated case.	Case **case**: Model of the case with assigned case number
execute(case,activity)	Executes the activity of a case indicated in the parameter. Returns the state of the updated case.	Case **case**: Model of the case with assigned case number ExecutedActivity **activity**: Activity retrieved from the list of activities in progress of the case
read(case)	Reads a case.	Case **case**: Model of the case with assigned case number
cancelCase(case, observa-	Cancels a case.	Case **case**: Model of the case with

Operation	Description	Parameters
tion)		case number to be cancelled. Cancellation is done with the online user executing the sdk rule String **observation**: Reason for cancellation
reassignCase(case, reassignUser)	Assignment of the task in execution of the case.	Case **case**: Case model with case number assigned to the task String **reassignUser**: Code of the user to whom the case task is assigned
reactivate(case)	Reactivate a canceled case.	Case **case**: Model of the case with case number to be reactivated.

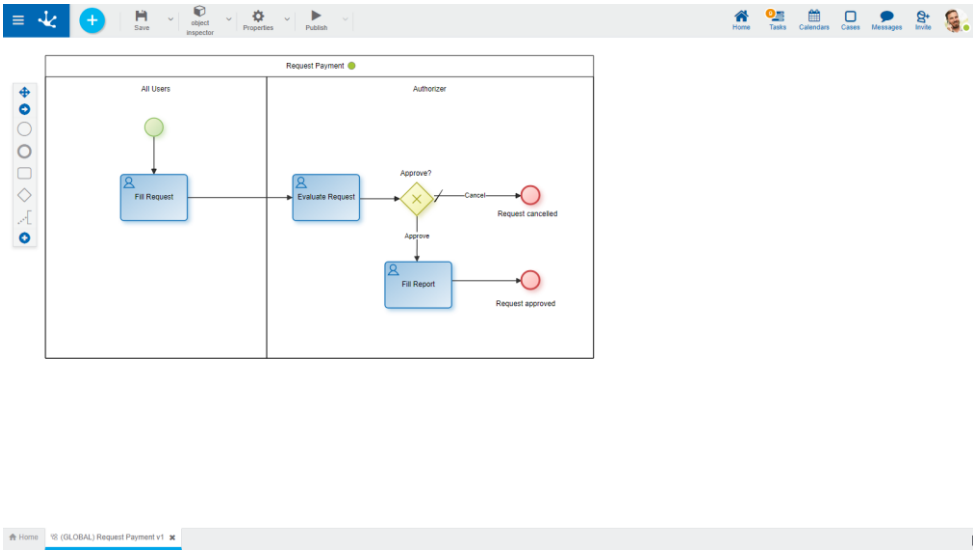
Examples of Use

The examples use the “Request Payment” process and each example contains the use of the “RequestPayment” model class and the “RequestPaymentService” service class.

In addition, “Request” and “Report” classes are used, which represent models of forms related to the “Request Payment” process.

The examples detail how to start a case, read it, execute and assign its activities, finish it and cancel it.

Process Diagram for Examples 1 to 7



1. Starting a case

To start the case, the first activity "Fill Request" is executed. The case moves to the “Evaluate Request” activity under the liability of “Afarias” user, configured in the “Authorizer” lane.

Executing the “Fill Request” activity performs the “Create” operation on the “Request” form with the values assigned to its fields.

An instance of the "RequestPayment" model class is created, in the "Request" form obtained through the getRequestEntity() method, values are assigned to its fields with the corresponding setters. The case is started using the "RequestPaymentService" service class with the startCase(myFirstPaymentCase) method.

```
RequestPayment myFirstPaymentCase = new RequestPayment();
myFirstPaymentCase.getRequestEntity().setAmount(new Dou-
ble(200));

SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd");
myFirstPaymentCase.getRequestEntity().setDueDate(new Da-
te(format.parse("2020-06-28").getTime()));
myFirstPaymentCase.getRequestEntity().setMessage("This is my
first payment request .. i need to buy more coffee !!");

RequestPaymentService myService = new RequestPaymentServi-
ce(getApiClient());
String cdCase = myService.startCase(myFirstPaymentCase);
```

2. Reading the case

An instance of the “RequestPayment” model class is created, it is assigned a case number generated by the startCase(myFirstPaymentCase) method of the “RequestPaymentService” service class, in the previous example.

The case is read with the read(myFirstPaymentCase) method of the "RequestPaymentSercice" service class and the case number.

```
RequestPayment myPaymentCase = new RequestPayment();
myPaymentCase.setCdCase(cdCase);
myPaymentCase = myService.read(myFirstPaymentCase);
```

3. Activity execution that defines modeled buttons

The case is in the “Evaluate Request” activity, which has two modeled buttons. The next activity to be executed depends on the button selected, in this example “Approve” is selected. If no button is specified, the modeled flow is followed as default.

The “Approve” button is assigned to the “Request Payment” model class using the corresponding setter. The “Evaluate Request” activity is executed with execute(myPaymentCase) method of the “RequestPaymentService” service class.

As a result, the case is in the “Fill Report” activity and its state is “ACTIVE”,

```
myPaymentCase.setLastPressedButton("Approve");
myService.execute(myPaymentCase);
```

4. Activity execution not defining modeled buttons

The case is in the “Fill Report” activity. This activity execution creates the “Report” form. This form has a field called "request" that is related to the "Request" entity.

The “request” field is completed by means of the corresponding setter, with the identifier of the “Request” entity related to the case, which is obtained by means of the getRequestEntity() method.

The rest of the fields are completed and the “Fill Report” activity is executed with the execute(myPaymentCase) method of the “RequestPaymentService” service class.

The case is sent to the “Request approved” end event and its state is ended.

```
Integer requestId = myPaymentCa-
se.getRequestEntity().GetRequestId();
```

```
myPaymentCa-
se.getReportEntity().setRequest(requestId.toString());
myPaymentCase.getReportEntity().setPaymentMethod("Credit");
myPaymentCase.getReportEntity().setSummary("Filling the report
using the SDK.");
myService.execute(myPaymentCase);
```

5. Cancellation of a case

An instance of the “RequestPayment” model class is created, it is assigned the case number to be canceled.
The cancellation reason is entered in the “observation” parameter and the case is canceled with the cancelCase(myPaymentCase, observation) method of the “RequestPaymentService” service class.

```
RequestPaymentService myService = new RequestPaymentServi-
ce(getApiClient());
RequestPayment myPaymentCase = new RequestPayment();
myPaymentCase.setCdCase("00000000911120000");
myService.cancelCase(myPaymentCase,"Duplicate case");
```

6. Assignment of the case task in execution

An instance of the “RequestPayment” model class is created, indicating the case number to which the task is assigned.
The user code is entered in the "reassignUser" parameter and the task is assigned with the reassignCase(myPaymentCase, reassignUser) method of the "RequestPaymentService" service class.

> *When the task of a case is assigned, the online user that executes the sdk rule is used. This user can assign their own tasks, those of their team and those of the participants of a role, if they were their coordinator.*

```
RequestPaymentService myService = new RequestPaymentServi-
ce(getApiClient());
RequestPayment myPaymentCase = new RequestPayment();
myPaymentCase.setCdCase("00000000911120000");
myService.reasignCase(myPaymentCase,"JPEREZ");
```

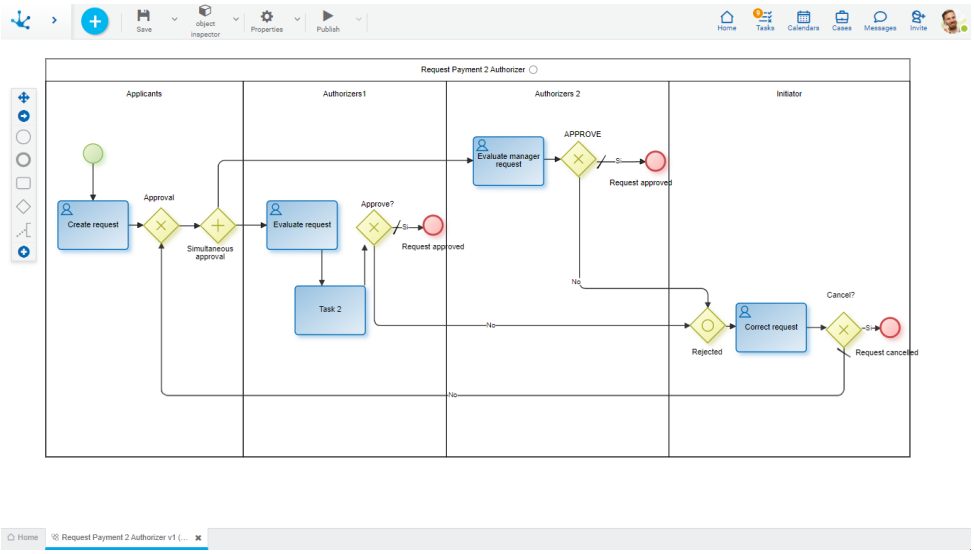
7. Reactivation of a canceled case

An instance of the “RequestPayment” model class is created, and the canceled case number is assigned to it.

The case is reactivated with the reactivate(myPaymentCase) method of the “RequestPaymentService” service class.

```
RequestPaymentService myService = new RequestPaymentServi-
ce(getApiClient());
RequestPayment myPaymentCase = new RequestPayment();
myPaymentCase.setCdCase("00000000911120000");
myService.reactivate(myPaymentCase);
```

Process Diagram for Example 9



8. Execution of a specific activity

Activities "Evaluate request" (9) and "Evaluate manager request" (11) are sent to be evaluated by two authorizers.
Activity "Evaluate manager request" (11) is executed. JPAZ user is the authorizer and rejects the activity in this case.

An instance of the “RequestPayment2Authorizer” model class is created, a case number to be read is assigned.
The case is read with the read(xMyCase) method of the "RequestPayment2AuthorizerSercice" service class and the case number.

Every activity in progress is retreived with the getLsCurrentActivities() method and an ExecutedActivity object is instantiated.
If the activity is "Evaluate manager request" (11), the “Reject” button is assigned through the corresponding setter and the activity is executed with the execute(xMyCase. xActivity, user) method of the “RequestPayment2AuthorizerService” service class.

```
RequestPayment2AuthorizerService xRequestPaymentService = new
RequestPayment2AuthorizerService(getApiClient());
RequestPayment2Authorizer xMyCase = new RequestPay-
ment2Authorizer();

xMyCase.setCdCase(idCase);
xMyCase = xRequestPaymentService.read(xMyCase);

for (int i = 0 ; i < xMyCase.getLsCurrentActivities().size();
i++){
    ExecutedActivity xActivity = xMyCa-
se.getLsCurrentActivities().get(i);

    if (xActivity.getCdActivity()== 11) {
        xMyCase.setLastPressedButtom("Reject");
       xRequestPaymentService.execute(xMyCase, xActivity);
    }
}
```

Search

Case searches of a process can be performed through the interaction of the following objects.

Criteria: represents search criteria on case data. It is made up of elements that can be properties and values, connected by operators.

Property Identifier	Description
Case.searchCdState	Case state
Case.searchPriority	Case priority
Case.searchDsCase	Case description
Case.searchDtInitiated	Case start date
Case.searchDtEnded	Case end date
Case.searchCdActivity	Activity code
Case.searchCdCase	Case number

Property Identifier	Description
Case.searchMyteam	My Team's Cases

Operator	Description
eq	Equal
gt	Greater
lt	Less
betweene	Between and admits equals
like	Contain

Possible values of the case state:

Identifier	Description
Case.stateActive	Active
Case.stateCancelled	Cancelled
Case.stateEnded	Finalized

Possible values of the case priority:

Identifier	Description
Case.priorityUrgent	Urgent
Case.priorityHigh	High
Case.priorityMedium	Medium
Case.priorityLow	Low

SearchCriteria: an object that groups search criteria (Criteria object), it also allows parameterizing the order of the result, the size of the reading page, and the number of pages to be retrieved. Reading by pages is used since the data volume on a process can be very large.

Process Service: service class of the process on which the case search is being carried out. Contains the search(searchCriteria) operation that must receive the SearchCriteria object as a parameter and returns the SearchResult object.
Example:
SearchResult searchResult = actionsService.search(searchCriteria);

SearchResult: object that contains the search results. Contains a list of the process cases, for example "Actions". This object allows to know the total number of reading pages resulting from the search, the size of each page, and the number of pages to retrieve.

Search Examples

1. Search

This example retrieves a list of cases from the "Actions" process with an active state, where the state property corresponds to the "Case.searchCdState" identifier and the active value corresponds to the "Case.stateActive" identifier. The number of pages to retrieve is also defined.

```
AccionesService xaccionservice = new AccionesServi-
ce(getApiClient());
SearchCriteria searchCriteria = new SearchCriteria();
Criteria criteria1 =
          Criteria.eq(Case.searchCdState,Case.stateActive);
searchCriteria.addCriteria(criteria1);
searchCriteria.setPage(1);

SearchResult searchResult =
             xaccionservice.search(searchCriteria);
List<Acciones> instancesResult = searchResult.getResult();
```

2. Constructor types for search criteria

This example creates a Criteria object and defines different search conditions using the "between and admits equals", "contains" and "greater" operators.
Additionally, a criterion is included to filter the cases belonging to the team of the user passed as a parameter.

```
// Between and admits equals. Dates must be in year-month-day
```

```
format.
// SearchCriteria searchCriteria = new SearchCriteria();
// SimpleDateFormat format =
//      new SimpleDateFormat("yyyy-MM-dd");
// Date dateFrom =
//      new Date(format.parse( "2020-01-01" ).getTime());
// Date dateTo =
//      new Date(format.parse( "2021-01-01" ).getTime());
//
// criteria  criteriaDate =  Crite-
ria.betweene(Case.searchDtInitiated,
// DateFrom, DateTo);
// searchCriteria.addCriteria(criteria Date);

// Contains. The case description contains the test word
// Criteria criteriaLike = Crite-
ria.like(Case.searchDsCase,"test");

// Greater. Case number greater than 80
// Criteria criteria = Criteria.gt (Case.searchCdCase, "80") ;

// My team's cases, pass the online user code
// Criteria.eq(Case.searchMyteam,"JPAZ");
```

> ***Deyel*** *uses the data type java.sql.Date for "date" type fields, so the transformation from java.util.Date to java.sql.Date must be done, when it is required to create an object with this type of data.*

3. Operations on the results object

The SearchResult object is iterative and as such, it can be scrolled through in different ways.
The example uses a FOR statement to iterate over the cases that meet the search conditions of example 1 (active state), retrieving the case start date and priority.

```
AccionesService xaccionservice = new AccionesServi-
ce(getApiClient());
SearchCriteria searchCriteria = new SearchCriteria();
// Cases with active state
Criteria criteria1 = Criteria.eq(Case.searchCdState,"ACTIVE");
    searchCriteria.addCriteria(criteria1);
```

```
        searchCriteria.setPageSize(10);
        searchCriteria.setPage(1);

        SearchResult searchResult = xaccionservi-
    ce.search(searchCriteria);

        List <Case>instancesResult = searchResult.getResult();
        // Loop through the list of cases
        String log = "";
        for (Case cases :instancesResult){
            log = cases.getDtInitiated() +" - "+ ca-
    ses.getPriority();
        }
```

Rules

The model class contains getters and setters for each of the parameters defined in an advanced rule, while the service class contains the operation to be performed with the model.

The model name as well as the service name are determined by the property SDK Class Name, specified when modeling the advanced rule.

For example, if the "taxCalculation" advanced rule is used when downloading the Java sources of the rule, you get a taxCalculation.java file that represents the model and another taxCalculationService.java file with the available operation of the service.

Model Class Content

An advanced rule model contains:

- Model constructor.
- General getters.
- Set of getters and setters of its parameters. The output parameters only have getters.

General getters

Operation	Description	Parameter
getCdRule()	Gets the identifier of the rule.	
getCdVersion()	Gets the version of the rule.	

Data Type

The parameters of an advanced rule have their equivalence with the data types of form fields.

Service Class Content

The service allows the following operations to be performed:

Operation	Description	Parameter
execute(rule)	Executes an advanced rule.	Rule **rule**: Model of the rule to be executed

Example of Use

The example invokes the "taxCalculation" advanced rule, using the model and service classes.

Values for "amount" and "taxRate" input parameters are defined in the model using its setters. Next, the execute(myRule) method of the service is executed and a VAT value is obtained from the getter of the "taxValue" parameter.

```
// Calculation of 21% VAT from the Entered Amount
taxCalculationService myService =
        new taxCalculationService (getApiClient());
taxCalculation myRule = new taxCalculation() ;
myRule.setAmount(new Double(100.12));
myRule.setTaxRate(new Double(21.00));

myRule = myService.execute(myRule);
// Gets the Amount of VAT Applied
Double doubleResult = myRule.getTaxValue();
log("The result of the tax value is:"
                                + doubleResult);
```

Rules Related to Fields

When a rule is enabled to be used in the relation to a field of an entity or of a page., two predefined parameters (input and output) are automatically generated. These parameters require certain considerations for the correct implementation of the rule.

- Pagination is a fundamental requirement to ensure fluid navigation between results and guarantee their correct presentation and access.
- Autocomplete implementation is necessary so that the related field can display suggestions in real time. If not implemented, users will not receive results while typing.
- Input parameters can be of any data type compatible with rule modeling, except for arrays or lists, which are not allowed.

Example of Implementation of the Rule “getUsersByName”

The purpose of this rule is to return a list of users whose names partially match the text entered in the related field. Additionally, as an extra filter, the condition is applied that the last name contains the text received in the input parameter “last name”.".

For example, if the user enters "Ju" in the related field and the input parameter “last name” is "Perez", the valid results could include users like: “Juana Perez”, “Juan Perez”, “Julian Costa Perez”.

Use of the following input and output parameters:

- Input parameters

  input: Object of type RuleRelationInput, containing the search and pagination criteria.

  last name: Text string (String) used as a filter to search for a match with users' last names.

- Output parameters

  output: Object of type RuleRelationOutput, which returns the list of results along with the pagination information.

Java Implementation Code

Below is an example of how to implement the rule "getUsersByName" in Java, which filters users by first and last name using the search and pagination criteria defined in the input parameters.

```
// Initializing the output parameter
output = new RuleRelationOutput();

UserService userService = new UserService(getApiClient());
SearchCriteria searchCriteria = new SearchCriteria();

// Apply selection criteria based on currentValues
if (!input.getCurrentValues().isEmpty()) {
```

```
                searchCriteria.addCriteria(Criteria.in("cdUser", in-
    put.getCurrentValues()));
    }
    // Apply auto-complete criteria
    if (input.getLikeValue() != null) {
       searchCriteria.addCriteria(
       Criteria.like("dsFirstName", input.getLikeValue()));
    }
    // Apply additional user-defined criteria
    if (lastname != null) {
       searchCriteria.addCriteria(Criteria.like("dsLastName", last-
    name));
    }

    // Apply pagination parameters
    searchCriteria.setPage(input.getPage());
    searchCriteria.setPageSize(input.getPageSize());

    SearchResult<User> SearchResult = userServi-
    ce.search(searchCriteria);
    for (User user : searchResult.getResult()) {
        // Generating output values
        RuleValue ruleValue = new RuleValue();
        ruleValue.setCode(user.getUserCode());
        ruleValue.setValue(userService._shortDescription(user));
        output.getResults().add(ruleValue);
    }

    // Configure pagination data in the response
    int page = searchResult.getCurrentPage();
    int pageSize = searchResult.getSearchCriteria().getPageSize();
    int total = searchResult.getTotal();
    output.setPaging(new Paging(page, pageSize, total));
```

Value Lists

All value lists have the same structure and are administrated by a single model class and a single service class, which are included in Deyel SDK.

The model class contains properties with getters and setters, while the service class contains the operations to be performed with the model.

The "ValueList" model class has the value list structure, while the "ValueListData" model class has the data.

ValueList Model Class Content

The "ValueList" model contains:

- Model constructor.
- Getters and setters to access its properties.
- ValueListData Collection.

ValueListData Model Class Content

This class is in a "Value list" model class collection. It represents each of the values in the value list.

The "ValueListData" model contains:

- Model constructor.
- Getters and setters to access its properties.

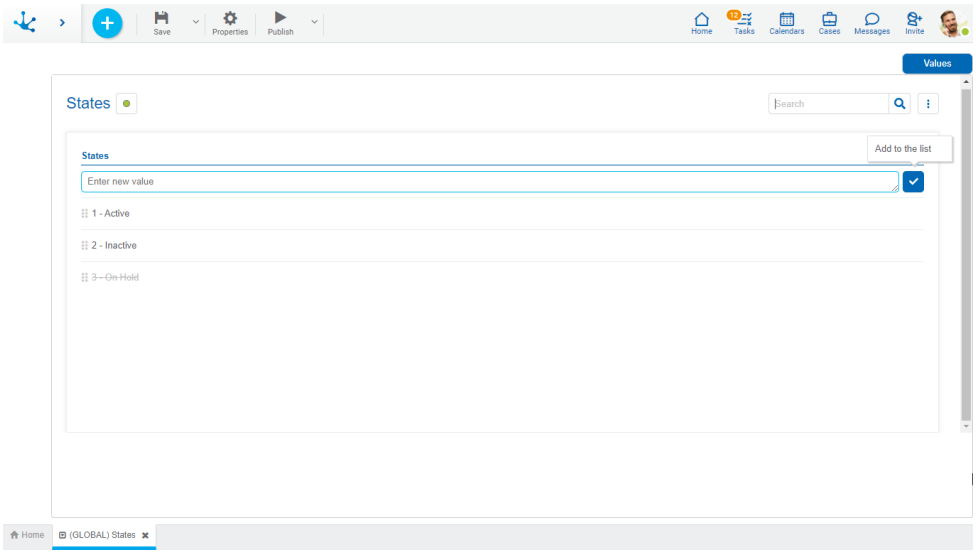
ValueListService Service Class Content

The service allows to read the value list.

Operation	Description	Parameters
read(valuelist)	Reads the value list data received as a parameter.	**valuelist**: Value list model class

Example of Use

In order to use this example, model the “States” value list in **Deyel** and previously enter the values to this list.



Read these values using Deyel SDK classes, which retrieve the following:

Code	Descriptive Value	Low Logic
1	Active	False
2	Inactive	False
3	On Hold	True

To read the value list, instantiate the “ValueListService” service. The "ValueList" model class is instantiated and the value of the identifier property of the list to be read is indicated, using the corresponding setters.
This instance is read using the read(valueList) method of the “ValueListService” service class. The instance contains all its properties which can be shown using the corresponding getters.

To read the data, access the “ValueListData” list with the corresponding getters of the “ValueList” model class.

Having the "ValueListData" list, it is possible to iterate through it to retrieve some or all of its values. This example retrieves the first value.

```
ValueListService valueListService = new ValueListServi-
ce(getApiClient());

ValueList valueList =  new ValueList();
valueList.setId("id_value_list");

valueList = valueListService.read(valueList);
```

```
valueList.getDsDisplayName();
valueList.getDsName();

List <ValueListData> valueListData = value-
List.getValueListData();

ValueListData firstValue = valueListData.get(0);

Integer  code = firstValue.getCode();  // code  = 1
String value= firstValue.getValue();   // value = "Active"
Boolean isDeleted = firstValue.getDeleted(); // isDeleted =
false
```

Users

The model class contains the properties with its getters and setters, while the service class contains the operations performed with the model.

Model Class Content

A user's model contains:

- Model constructor.
- Getters and setters of its attributes.

Getters to get user attributes

The model class for a case contains a set of getters for the attributes of the case.

Operation	Description	Parameters
getFirstName()	Gets the user first name.	
getLastName()	Gets the user last name.	
getEmail()	Gets the user email.	
getUserCode()	Gets the user code.	
getAlias()	Gets the user alias.	

Operation	Description	Parameters
getOrganizationalUnit()	Gets the user organizational unit.	
getAuthorizedUserCode()	Gets the authorizer user code.	
getCalendar()	Gets the user calendar.	
getJobs()	Gets the job positions.	
getDelegates()	Gets the user delegated users.	
getDelegatesOf()	Gets the users where the user is defined as a delegate user.	
getActive()	Gets the user state. Possible values: **1** - Active **0**- Inactive	
getExpiration()	Gets the user expiration date.	
getUserDurationDays()	Gets the number of days the password expires since the user was created. Possible values: **0** - Never **30** - 30 days **60** - 60 days **90** - 90 days	
getLicenses()	Gets a list of licensed products for the user.	
getPermissions()	Gets a list with the user permissions.	
getRoles()	Gets a list with the user roles.	
getIdentificationType()	Gets the identification type of the user personal data.	
getIdentificationNumber()	Gets the identification number of the user personal data.	

Operation	Description	Parameters
getPhone()	Gets the user phone number.	
getExtensionNumber()	Gets the user phone extension number.	
getNationality()	Gets the user nationality.	
getBirthday()	Gets the user birthdate.	
getCountry()	Gets the user country.	
getState()	Gets the user state	
getCity()	Gets the user city.	
getZipCode()	Gets the user zip code.	
getStreetName()	Gets the user street name.	
getStreetNumber()	Gets the user street number.	
getDepartament()	Gets the user department code.	
getLinkedinAccount()	Gets the linkedin user code.	
getTwitterAccount()	Gets the twitter user code.	
getFacebookAccount()	Gets the facebook user code.	
getYouTubeAccount()	Gets the youtube user code.	
getSkypeAccount()	Gets the skype user code.	
getObservation()	Gets the user observations.	

Service Class Content

The service allows the following operations to be performed:

Operation	Description	Parameters

Operation	Description	Parameters
read(user)	Reads an instance.	User **user**: Model object to read
create(user)	If the executing user has permission to create users, a user is created. If successful, it returns the user code, otherwise it gives an exception with an error message.	User **user**: Model object to create
update(user)	If the executing user has permission to modify users, the modification is performed. If successful, it returns the user code, otherwise it gives an exception with an error message.	User **user**: Model object to modify
delete(user)	If the executing user has permission to delete users, it deletes them. If it is not successful, it throws an exception with an error message.	User **user**: Model object to delete
getUserImage(user)	Gets the user profile image.	User **user**: Model object to read
getThumbnail(user)	Gets the thumbnail image of the user profile.	User **user**: Model object to read
verifySecurity(user,function)	Returns true if the user has the security function specified as a parameter in his permissions, and false if otherwise.	User **user**: Model object to read String **function**: Code of the specified safety function
search(searchCriteria)	Allows to retrieve the user instances that meet the search criteria.	SearchCriteria **searchCriteria**: Search criteria
invite(Mail, Subject, Text, IdApplication, OrganizationalUnit, Licenses, Permissions, Roles)	Allows inviting users to a specific application of **Deyel**. If it is not reported, Global application is assumed. If the invited user has a user account, the permissions from the invitation they did not previously have are added. If the session is open, the user enters	String **Mail**: email of the user to invite String **Subject**: email subject to be sent. String **Text**: content of the email to be sent. String **IdApplication**: identifier

Operation	Description	Parameters
	the specified application, otherwise, the user is redirected to the login page of that application. If the invited user does not have a user account, one will be created when they accept the invitation, and they will be logged into **Deyel** from the specified invitation. If any of the data is not valid, an exception is returned indicating the reason.	of the application to which the invited user will enter String **OrganizationalUnit**: organizational unit code of the invited user List<User.Licenses> **Licenses**: License types assigned to the invited user List<User.Permissions> **Permissions**: permissions assigned to the invited user List<User.Roles> **Roles**: roles assigned to the invited user

Example of Use

The examples use the **Deyel** user object and each example contains the "User" model class and the "UserService" service class.

1. Service creation

This service is created only once in the rule and is reused in different operations.

```
UserService userService = new UserService(getApiClient());
```

2. User reading

In this example, a user with the userCode "AFARIAS" is read, using the read(user) method of the "UserService" service class. The value of the lastName property is obtained with the getter of the "User" model class.

```
User user = new User();
user.setUserCode("AFARIAS");
user = userService.read(user);
```

3. Create a user

This example creates a user with the userCode property with value "NEW". It is activated with a license for the **Deyel** product and the End user permission.

```
UserService userService = new UserService(getApiClient());
User user = new User();
user.setUserCode("NEW");
user.setFirstName("New");
user.setLastName("User");
user.setEmail("newuser@optaris.com");
user.setAlias("newuser@optaris.com");
user.setOrganizationalUnit("0000000003");
user.setActive(1);
List<User.Licenses> licensesList = new ArrayList<>();
User.Licenses licenses = new User.Licenses();
licenses.setProduct("DEYE");
licenses.setLicenseType("EVE");
licensesList.add(licenses);
list<User.Permissions> permissionsList = new ArrayList<>();
User.Permissions permissions = new User.Permissions();
permissions.setApplication("DEYEL");
permissions.setPermissionCode("ENDUSER");
permissionsList.add(permissions);
user.setLicenses(licensesList);
user.setPermissions(permissionsList);
userService.create(user);
```

4. User update

In this example, the user identified with userCode "AFARIAS" is updated. It is inactivated and the organizational unit is changed.

```
User user = new User();
user.setUserCode("AFARIAS");
user = userService.read(user);
user.setOrganizationalUnit("0000000008");
user.setActive(0);
userService.update(user);
```

5. User deletion

In this example, the user identified with userCode "AFARIAS" is deleted.

```
User user = new User();
user.setUserCode(“AFARIAS”);
user = userService.read(user);
userService.delete(user);
```

6. Get the user profile image

In this example, a user identified with the userCode "AFARIAS" is read and its profile image is obtained through the getUserImage(user) method of the "UserService" service class.

```
User user = new User();
user.setUserCode("AFARIAS");
userService.getUserImage(user);
```

7. Get the thumbnail image of the user profile.

In this example, a user identified with the userCode "AFARIAS" is read and its thumbnail profile image is obtained through the getThumbnail(user) method of the "UserService" service class.

```
User user = new User();
user.setUserCode("AFARIAS");
userService.getThumbnail(user);
```

8. Security check

In this example it is verified if a user identified with userCode “AFARIAS' has the “DUU0C001” security function defined. The verifySecurity((user, “DuU0C001”) method of the “UserService” service class is used with a user that has been read using the read(user) method and the specified security function, as parameters.

```
User user = new User();
user.setUserCode("AFARIAS");
user = userService.read(user);
```

```
boolean userHasAccess = userService.verifySecurity(user,
"DUU0C001");
```

9. Invitation to users

In this example, an invitation is sent to the email "proof@optaris.com". This invitation contains: a license for two products, two permissions and two roles. The specific application that the user must access is established.

```
UserService userService = new UserService(getApiClient());
List<User.Licenses> licensesList = new ArrayList<>();
// Define 2 licenses
User.Licenses licenses1 = new User.Licenses();
licenses1.setProduct("DEYEL");
licenses1.setLicenseType("NOM");
User.Licenses licenses2 = new User.Licenses();
licenses2.setProduct("CONT");
licenses2.setLicenseType("MIT");
licensesList.add(licenses1);
licensesList.add(licenses2);
// Define 2 permissions
List<User.Permissions> permissionsList = new ArrayList<>();
User.Permissions permissions1 = new User.Permissions();
permissions1.setApplication("DEYEL");
permissions1.setPermissionCode("ENDUSER");
permissions2.setApplication("GLOBAL");
permissions2.setPermissionCode("MODELER");
permissionsList.add(permissions1);
permissionsList.add(permissions2);
// Define 2 roles
List<User.Roles> rolesList = new ArrayList<>();
User.Roles role1 = new User.Roles();
role1.setRole("DEYEL_TESTFUNCTIONS");
role1.setApplication("DEYEL");
User.Roles role2 = new User.Roles();
role2.setRole("ARGLOBAL_TIMCOMMUNICATION");
role2.setApplication("GLOBAL");
rolesList.add(role1);
rolesList.add(role2);
//Define the application
String idApp = "AED9B9C79826C48";
```

```
    try {
          userService.invite("test@optaris.com", "Invitation test
with app", "test text", idApp, "0000000008", licensesList, per-
missionsList, rolesList);
} catch (Exception e) {
          log(e.getMessage());
}
```

Search

There are two types of search: Eager and Lazy.

Search on users can be performed by using the SearchCriteria object. Such an object implements the setReadTypeLazy() and setReadTypeEager() methods to indicate the type of search to perform, and the getReadType() method to retrieve the set search type.

> *It is not necessary to specify the use of Eager search with the setReadTypyEager() method because it is the one used by default.*

Lazy search reading by default retrieves the values of the following user properties: UserCode, FirstName, LastName, Active, Type, Alias, Email, OrganizationalUnit, Calendar, the user image reference that is retrieved by the getUserImageLink() method, and the user thumbnail image reference that is retrieved by the method getThumbnailLink() of the User object.

The searchCriteria object implements the setLazyFields() method where the properties to be obtained are defined. This method is used when the value of a property that is not retrieved by default in the Lazy search is required.

To retrieve the user properties: dsFirstName, dsLastName, cdActive, codeMail, ImgUser, the setLazyFields() method is used, which implements the searchCriteria object.

```
List xProjection = new java.util.ArrayList();
xProjection.add("dsFirstName");
xProjection.add("dsLastName");
xProjection.add("cdActive");
xProjection.add("codigoAMail");
xProjection.add("imgUser");
xSearchCriteria.setLazyFields(xProjection);
```

Search Examples

1. Eager Search

In this example a list is retrieved with the users that in the dsEmail property contain "gmail" and that in the cdOrgUnit property have the identifier of the "Sales" organizational unit (0000000007). The results are sorted by last name in ascending order. The list is scrolled displaying the first and last names of the users. The same criteria are used as for the form search.

```
SearchCriteria searchCriteria = new SearchCriteria();
Criteria criteria1 = Criteria.like("dsEmail", "gmail");
Criteria criteria2 = Criteria.eq("cdOrgUnit", "0000000007");
searchCriteria.addCriteria(criteria1);
searchCriteria.addCriteria(criteria2);
searchCriteria.addOrderAsc("dsLastName");
SearchResult SearchResult = userService.search(searchCriteria);
Iterator it = SearchResult.getResult().iterator();
while (it.hasNext()) {
    User user = (User) it.next();
    log(user.getFirstName() + " " + user.getLastName());
}
```

2. Lazy Search

This is an example to get the values of the properties “dsFirstName” and “dsLastName”.
The users retrieved are those that in the cdOrgUnit property have values of the organizational unit identifier other than "Sales" (0000000007) and "Systems" (0000000008). The results are sorted by last name in descending order and the page size is set to 5.
The list is scrolled displaying the first and last names of the users. The same criteria are used as for the form search.

```
SearchCriteria xSearchCriteria = new SearchCriteria();
xSearchCriteria.setReadTypeLazy();
xSearchCriteria.addCriteria(Criteria.nin("cdOrgUnit", ja-
va.util.Arrays.asList("0000000007", "0000000008" )));
xSearchCriteria.setPageSize(5);
List xProjection = new java.util.ArrayList();
xProjection.add("dsFirstName");
xProjection.add("dsLastName");
xSearchCriteria.addOrderDesc("dsLastName");
xSearchCriteria.setLazyFields(xProjection);
SearchResult xSearchResult= xUserServi-
ce.search(xSearchCriteria);
```

```
for (int page = 1; page <= xSearchResult.getPages(); page++) {
List<User> lsUser = xSearchResult.goToPage(page).getResult();
for (User xUser : lsUser) {
 log(xUser.getFirstName() + " " + xUser.getLastName()+ " page =
"+page);
 }
}
```

Organizational Units

The model class contains the properties with its getters and setters, while the service class contains the operation to be performed with the model.

Model Class Content

A user's model contains:

- Model constructor.
- Getters and setters of its attributes.

Getters to get the organizational unit attributes

The model class of a case contains a set of getters for the organizational unit attributes.

Operation	Description	Parameters
getDisplayName()	Gets the descriptive name of the organizational unit.	
getName()	Gets the name of the organizational unit.	
getManager()	Gets the manager user of the organizational unit.	
getSuperiorOrganizationalUnit()	Gets the superior organizational unit of the organizational unit.	
getDescription()	Gets the description of the organizational unit.	
getEmail()	Gets the email of the organiza-	

Operation	Description	Parameters
	tional unit.	
getCalendar()	Gets the calendar of the organizational unit.	
getManagerCanReadPrivateData()	Returns true if the organizational unit manager has permission to read private data, false otherwise.	
getManagerCanUpdatePrivateData()	Returns true if the organizational unit manager has permission to update private data, false otherwise.	
getManagerCanDeletePrivateData()	Returns true if the organizational unit manager has permission to delete private data, false otherwise.	
getUserCanReadPrivateData()	Returns true if the organizational unit users have permission to read private data, false otherwise.	
getUserCanUpdatePrivateData()	Returns true if the organizational unit users have permission to update private data, false otherwise.	
getUserCanDeletePrivateData()	Returns true if the organizational unit users have permission to delete private data, false otherwise.	

Service Class Content

The service allows the following operations to be performed:

Operation	Description	Parameter
read(orgUnit)	Reads an instance.	User **OrgUnit**: Code of the organizational unit to read

Example of Use

The **Deyel** organizational unit object is used. It contains the use of the "OrganizationalUnit" model class and the "OrganizationalUnitService" service class.

1. Service creation

This service is created only once in the rule and is reused in different operations.

```
OrganizationalUnitService organizationalUnitService = new Orga-
nizationalUnitService(getApiClient());
```

2. Reading of the organizational unit

This example retrieves the manager property value of an organizational unit identified with the "000000008' code, using the read method (organizationalUnit) of the "OrganizationalUnitService" service class. The manager property value is obtained through the corresponding getter of the "OrganizationalUnit" model class.

```
OrganizationalUnit organizationalUnit = new OrganizationalU-
nit();
organizationalUnit.setOrganizationalUnit("0000000008");
OrganizationalUnit xOrganizationalUnitReaded = organizationalU-
nitService.read(organizationalUnit); log("Manager: " + organi-
zationalUnitReaded.getManager());
```

Calendars

For the calendar object, only service class operations are defined to work with dates, the model class is not necessary because the **Deyel** data types are used. The service class is predefined in the Deyel SDK and does not need to be downloaded.

Service Class Content

The service allows the following operations to be performed:

Operation	Description	Parameters
worksDays(start, end)	Obtains work days between dates, using the current user's calendar or another one specified as a parameter.	Timestamp **start**: Period start date Timestamp **end**: Period end date
worksDays(start, end, calendar)	Obtains work days between dates, using the calendar specified as a parameter.	Timestamp **start**: Period start date Timestamp **end**: Period end date Integer **calendar**: Code of the specified calendar
convertToTimeZone(date, calendar)	Converts a date into a specified time zone.	Timestamp **date**: Date to convert Integer **calendar**: Calendar code with specified time zone

Example of Use

Examples contain the use of "CalendarService" service class.

1. Service creation

This service is created only once in the rule and is reused in different operations.

```
CalendarService calendarService = new
    CalendarService(getApiClient());
```

2. Calculation of work days between dates

In this example, a budget number is read and the number of work days between the budget entry date and the budget due date is calculated.

A budget with the dsNumber "100234" property value of the "Budget" model class is read, using the read(budget) method of the "BudgetService" service class.
With the workDays(start, end) method of the "CalendarService" service class, the number of work days between the dtQuote and dtDue properties values is obtained, from the corresponding getter methods of the "Budget" model class.

```
BudgetService budgetService = new BudgetServi-
ce(getApiClient());
 String dsNumber = "100234";
 Budget budget = new Budget();
 budget.setDsNumber(dsNumber);
 budget = budgetService.read(budget);
 Timestamp start = new Times-
tamp(budget.getDtQuote().getTime());
 Timestamp end = new Timestamp(budget.getDtDue().getTime());
 calendarService.worksDays(start, end);
```

3. Time calculation with another time zone

This example converts local time to Mexico time, set by calendar 2 time zone, using the convertToTimeZone(localTime, 2) method of the "CalendarService" service.

```
Timestamp localTime = Timestamp.valueOf(LocalDateTime.now());
    Timestamp timeMexico =
calendarService.convertToTimeZone(localTime, 2);
   log("Local Time: " + localTime + " Time Mexico: " + timeMe-
xico);
```

Email Sending

Email delivery is administrated by a single model class and a single service class, included in the Deyel SDK.
The model class contains the properties with its getters and setters, while the service class contains the operations to be performed with the model.

Email Model Class Content

The "Email" model contains:

- Model constructor.
- Getters and setters to access the email properties.

Getters and setters of the Email model properties

Operation	Description	Parameters
getTo()	Retrieves the list of recipients' email addresses.	
getCc()	Retrieves the list of email addresses of those who will receive a carbon copy of the email.	
getBcc()	Retrieves the list of email addresses of those who will receive a blind carbon copy of the email.	
getReplyTo()	Retrieves the list of email addresses configured with automatic response.	
getSubject()	Allows to retrieve the email subject.	
setSubject()	Allows to define the email subject.	String **subject**: Email subject
getText()	Allows to retrieve the email text.	
setText()	Allows to define the email text.	String **text**: Email text
getAttachments()	Allows to retrieve the list of email attachments.	
setAttachments()	Allows to define a list of attachments to the email.	List of Files: Files that are sent as attachments in the email
getAttachmentReferences()	Allows to retrieve the set of file references associated with the email.	
setAttachments()	Allows to define the set of references to files and associate them to the email.	FileReference List
setAlias()	Allows to define an alias for the email.	String **alias**: Alias text

> *The Reply To attribute in the email standard allows to assign recipient addresses of the reply email. When the email recipient selects "Reply" from their email client, the boxes entered with Reply To are proposed as recipients and can be updated if the user so wishes.*

> *The maximum defined for the total sum of the attached files size is 25mb, both for File and FileReference objects.*

> *When using the setAlias() method, the issuing account is not being masked, which is considered spam, but rather other informative data is being added.*

EmailService Service Class Content

Service allows the following operation to be performed:

Operation	Description	Parameters
send(email)	Sends an email.	Email **email**: Email model class

Example of Use

The example contains the use of "EmailService" service class.

1- Sending an email with an attachment

In this example, an email is sent where its different sections are specified: "Recipient", "With carbon copy", "With a blind copy",
"Reply to", "Subject", "Email alias", "Email body". An invitation card generated by an external method is attached to the email (generating the file is not covered in this example).

The "EmailService" service and the "Email" model class are instantiated, and the value of their properties is indicated by means of the corresponding getters and setters of the model class.
An invitation card is created, which is attached to the "Email" model class with the corresponding method.

To send the email, the send(email) method of the "EmailService" service is executed.

```
EmailService emailService = new EmailService(getApiClient());

Email email = new Email();
```

```
email.getTo().add("partner@gmail.com");

email.getCc().add("afarias@deyel.com");

email.getBcc().add("slopez@deyel.com");

email.getReplyTo().add("contact@deyel.com");

email.setSubject("Invitation to Event");

email.setAlias("Internal Communication");

email.setText("Hi Partner!!, our new webinar will be next Fri-
day..");

File invitationCardFile = createInvitationCard();

email.getAttachments().add(invitationCardFile);

emailService.send(email);
```

2- Sending an email with an attached file read from a form field

This example shows two alternatives for retrieving a file from a form. The retrieved file is sent by email.

> *The Java object model of a form provides methods to retrieve the reference to the file stored in its fields. This reference is a more efficient Java object for this type of operation and allows for faster rule processing time.*

> *When the files retrieved from a form are sent via email, the use of the FileReference alternative is recommended.*

An "Account" form instance is read using the read(account) method of the "AccountService" service class.

```
AccountService accountService = new AccountServi-
ce(getApiClient());
Account account = new Account();
account.setIdAccount(50);
account = accountService.read(account);
```

File Alternative

The "fllogo" field of the "Account" form is retrieved, creating a new File object and the file is sent by email.

```
File logo = accountService.getFlLogo(account);
email.getAttachments().add(logo);
emailService.send(email);
```

FileReference Alternative

The "fllogo" field of the "Account" form is retrieved, using FileReference and the file is sent by email.

```
FileReference logoReference = account.getReference_FlLogo();
email.getAttachmentReferences().add(logoReference);
emailService.send(email);
```

Java Files

Deyel allows the use of temporary files during the execution of advanced rules, facilitating the temporary storage and management of data. To do this, it uses the FileService service, through which it is possible to create and manage files of type "java.io.File" in a controlled manner and without requiring any additional storage configurations.

Objects of type "java.io.File" can only be managed by the FileService service class, unlike the other objects in the Deyel SDK, which use the model and service classes.

Content of the FileService Service Class

The service class name in Deyel SDK is fixed and must be used as FileService.

The service allows the following operations to be performed:

Operation	Description	Parameters
createTempFile(String pFileName)	Creates a temporary file within **Deyel**.	String **pFileName**: Name and extension of the file to create

Example of Use

In the following example, FileService is used to create a temporary file with a .txt extension. A message is written and, in case of an error, the exception is logged.

```
FileService xFileService = new FileService(getApiClient());
java.io.File xTempFile = xFileService.createTempFile( pFileName
"tempFile.txt");


try (java.io.FileWriter xWriter = new ja-
va.io.FileWriter(xTempFile.getPath())) {
    xWriter.write( str "Temporary text file created using the
file service.");
} catch (Exception e) {
    log( plog "An error occurred while writing to the temporary
file.");
}
```

Programming Advanced Rules with a Java IDE

To develop an advanced rule using an IDE, a series of steps must be followed.

Once the rule is created in the advanced rules modeler, it is downloaded to be included in the IDE project, it is developed and, once finished, it is imported from the context menu into the Deyel modeler.

Example of Use

This example describes the steps to develop the "getCostsWithTax" rule in the IDE.

A rule named "getCostsWithTax" is created. It receives a list of amounts as input parameter. Such list is added and the VAT cost is obtained. Returns the total VAT and the total amount of the sum as output parameter.

The “getCostsWithTax'' rule invokes the "taxCalculation" rule, which receives the sum of amounts as an input parameter, in the "inputAmount" field. Returns the VAT value as an output parameter, in the "taxValue" field.

> *For the example, the "taxCalculation" rule is assumed to be available within the environment.*

Step 1: Create the "getCostsWithTax" rule

In the rule modeler, a rule is created with the "getCostsWithTax" descriptive name.
The previously created adapter is selected, enter a description and press "Create Rule".

An input parameter is defined:

Name	Data Type
"amountList"	Double[]

Defined as output parameters:

Name	Data Type
"totalAmount"	Double
"AmountOfTaxApplied"	Double

This rule should be related to the "taxCalculation" rule using the wizard to include related objects, from the advanced properties tab.

Once the parameters have been entered and the new relation has been added, select the "Save" option from the top toolbar.

Step 2: Download the "getCostsWithTax" rule

From the top toolbar of the rule modeler, select the "Download" option.
The rule download includes the Java sources for developing and testing the rule.

Step 3: Setup the local environment

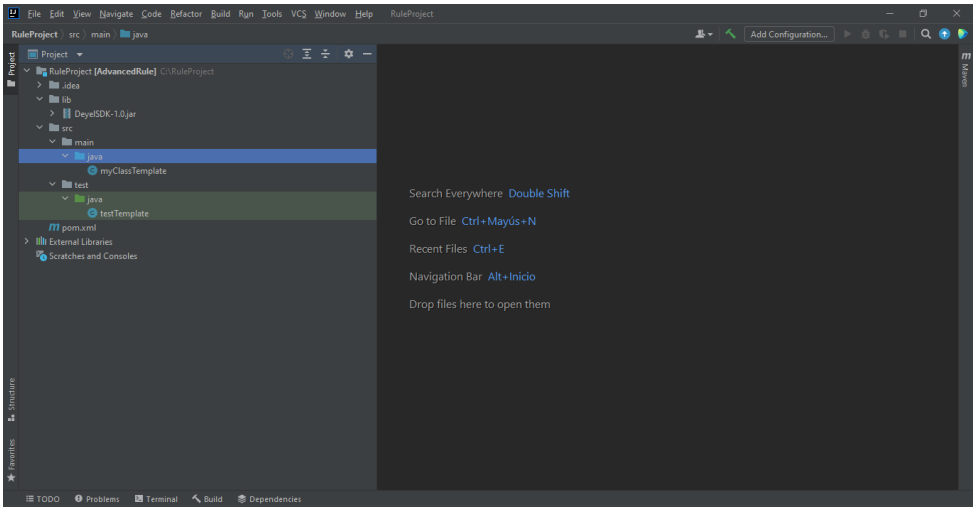
To develop the rule you need to have a local environment to work with Java.

Step 4: Include the "getCostsWithTax" rule in the Java IDE project

The rule download generates a file with a zip extension containing a "src" folder that respects the structure of Maven projects.

The file with zip extension should be copied to the project folder and decompressed in the same folder.

The structure of the project already configured is shown:



Step 5: Rule development

When the "getCostsWithTax" rule is included in the Java IDE project, two classes are obtained:

Name	Description
getCostsWithTax_1	Includes the rule logic
getCostsWithTax_1_Test	Includes the values to test from the IDE

The following code is defined in the getCostsWithTax_1 class:

```
   public class getCostsWithTax_1 extends
getCostsWithTax_1_abstract{
  protected void run() throws java.lang.Exception{
  // The sum of the amounts is calculated
  totalAmount = 0.0 ;
  for (Double amount: AmountList
      totalAmount += amount;
  }
  // 21% VAT on the amount entered is calculated
taxCalculationService myService =
          new taxCalculationService(getApiClient());
taxCalculation myRule = new taxCalculation();
myRule.setInputAmount(totalAmount);
myRule.setTaxRate(new Double(21.0));
```

```
        myRule = myService.execute(myRule);
          // Gets the amount of VAT applied
          amountOfTaxApplied = myRule.getTaxValue() ;

          }
      }
```

Setup of the Local Environment

The steps necessary to setup the local environment to work with Java are described below. After completing these steps, the rule can be included in the Java IDE.

> *For projects created in a version earlier than 7.7, it is recommended to build a new project with the Maven project structure.*

Step 1: Install the Java JDK

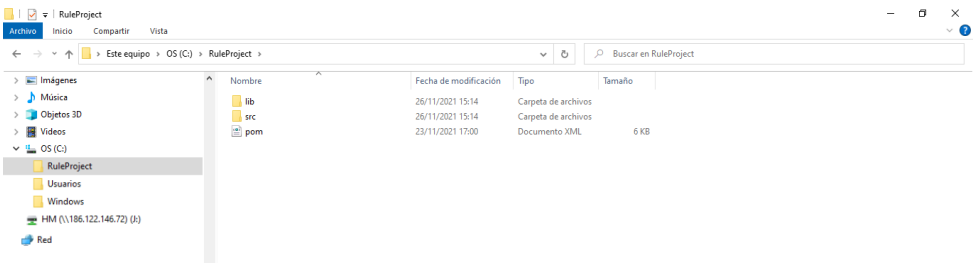
Download and install the Java JDK (Java Development Kit).

Step 2: Install an IDE of preference to work with Java

Download and install a Java IDE.

Step 3: Download the Maven Project

From the top toolbar in the save submenu of the rule modeler select the option "Download SDK". Downloading the Maven project, a file with a zip extension is obtained, once it is unzipped into a folder, it is subsequently imported into the project defined in the IDE.



Step 4: Create the project in a Java IDE

In order to create a project in a Java IDE it is necessary to:

- Import from the IDE the Maven project previously downloaded in the folder in Step 3. Folders will automatically be associated with the project in the IDE used.

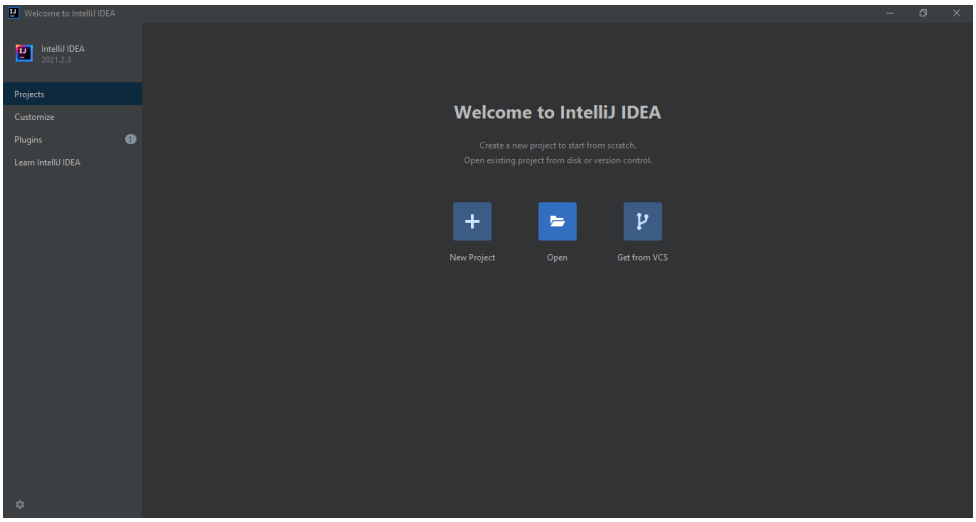
[Guide to create the project in the IntelliJ IDE]

[Guide to create a project in Eclipse IDE]

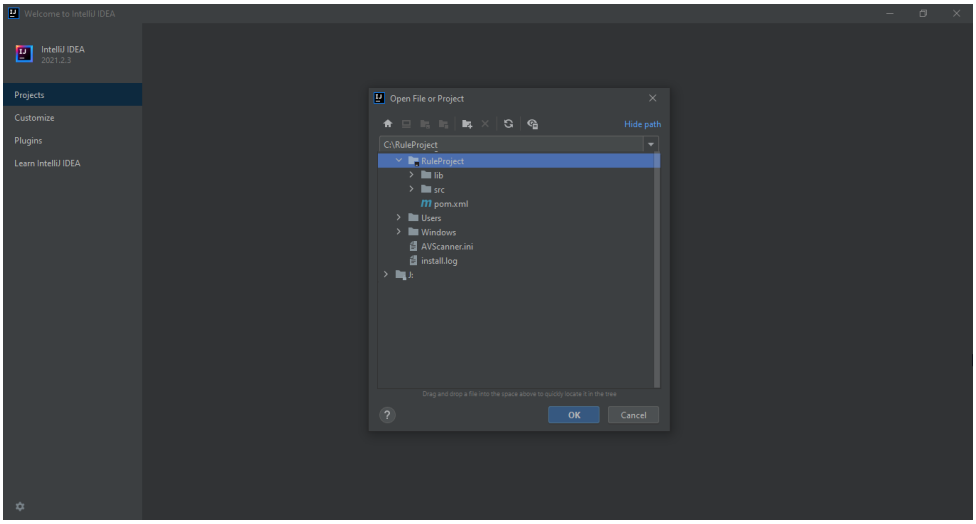
Guide to create the project in the IntelliJ IDE

The steps required to create the project in the Intellij IDE are described below.

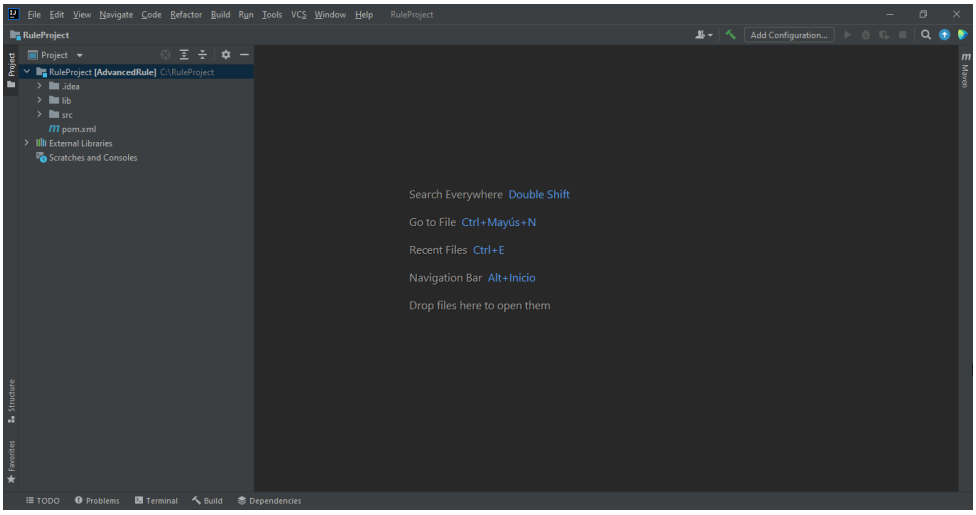
1. Press the button corresponding to open.



2. Select the folder in which the Maven project downloaded from **Deyel** was unzipped. Press the button corresponding to accept.



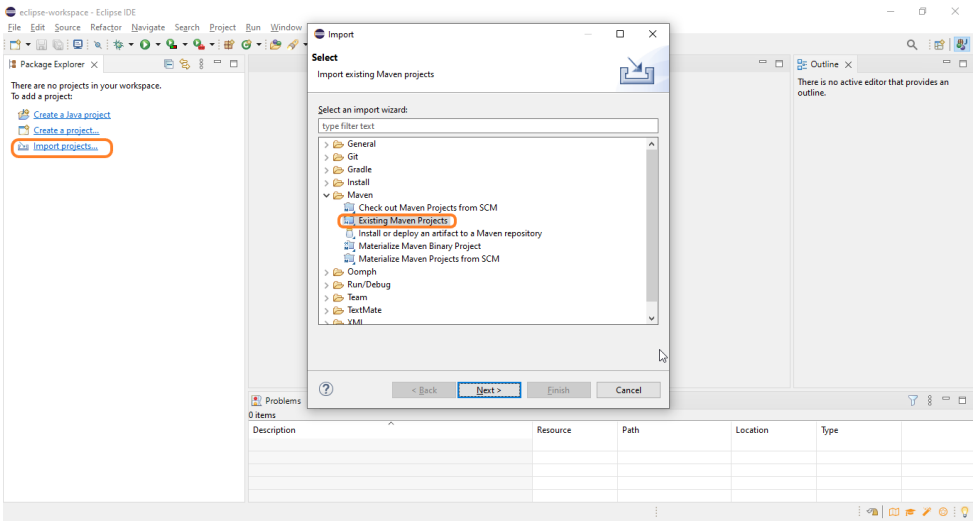
3. End of installation, the project has already been imported.



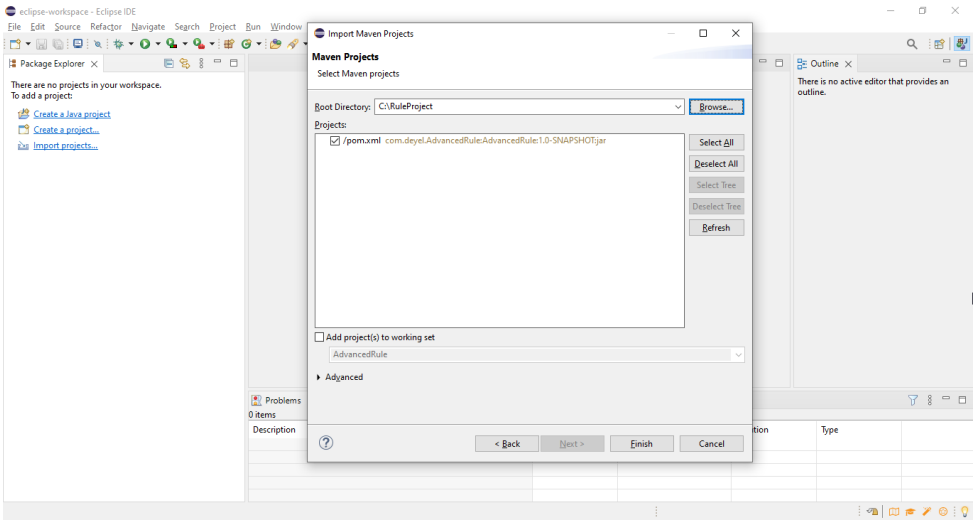
Guide to create a project in Eclipse IDE

The steps required to create a project in Eclipse IDE are described below.

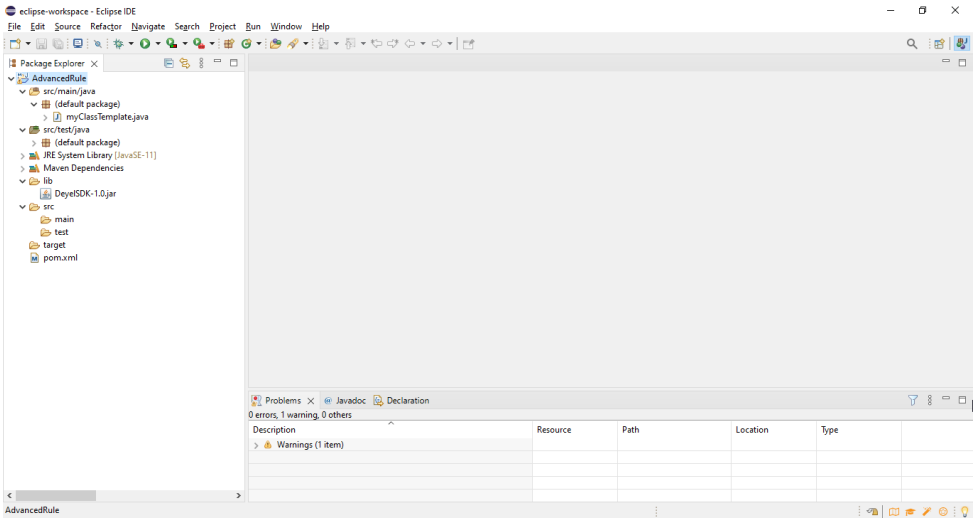
1. Select "Import Projects". Choose the option "Existing Maven Projects'' and press the button corresponding to next.



2. Select the folder in which the Maven project downloaded from **Deyel** was unzipped. Press the button corresponding to end.



3. End of installation, the project has already been imported.



3.4.12.1.2.4. Built-in Rules

Deyel includes examples of built-in rules to perform certain functionalities. These rules can be used as a reference for the development of new rules that meet the specific logic required by the business.

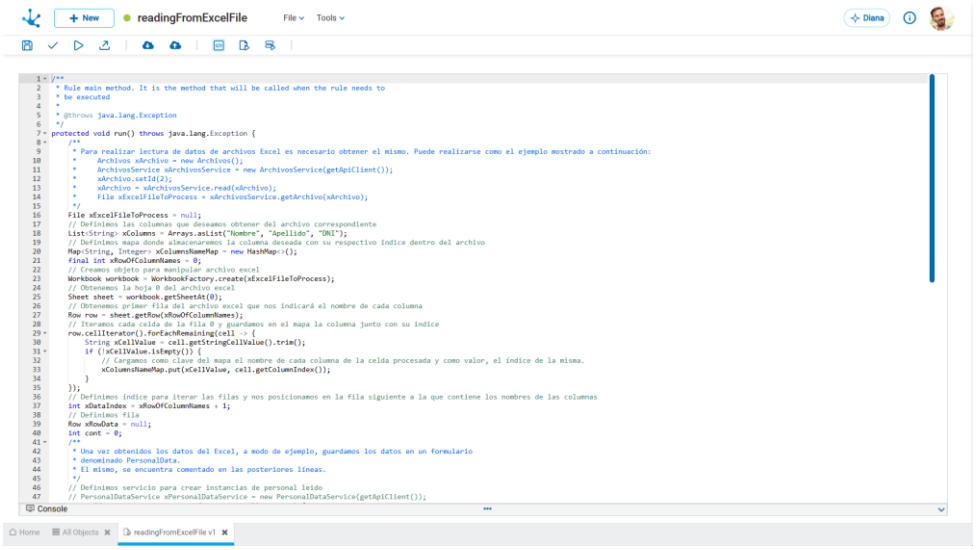
Examples

Below are some examples of built-in rules.

Reading data from an Excel file

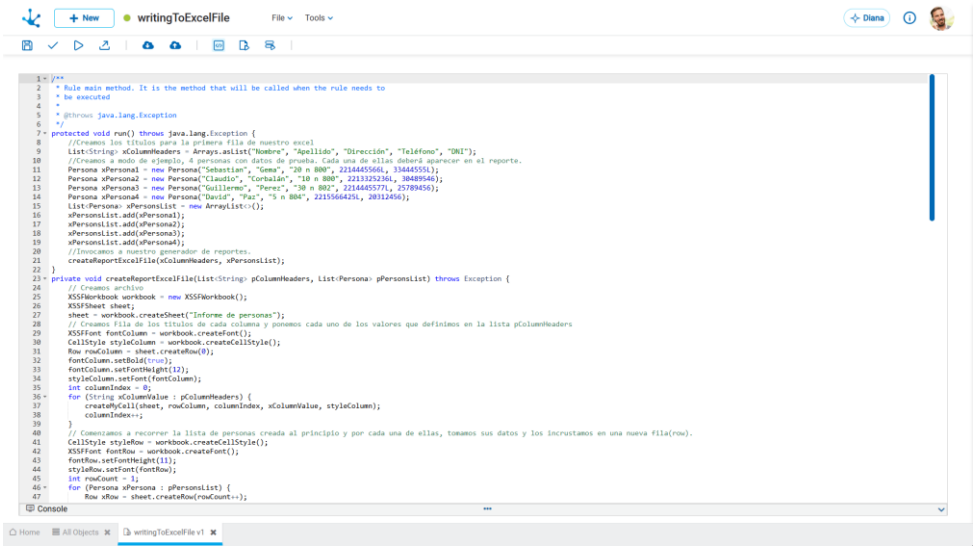
This example is used to read an Excel file and for each row a destination form instance is created.
An initial comment exemplifies how to recover the Excel file, which must be loaded as an attachment in source form instance.

From each row of this file, the values of its cells are retrieved, which are used to populate the fields of a new destination form instance.



Writing data to an Excel file

This example is used to save the data obtained from an array to an Excel file which is sent by email. From each occurrence of the array, the values entered to complete the cells of the Excel file rows are recovered. The first row corresponds to column names.
Finally, a name is given to the file, along with the extension “xlsx”. Once the file is complete, it is closed and sent as an email attachment.



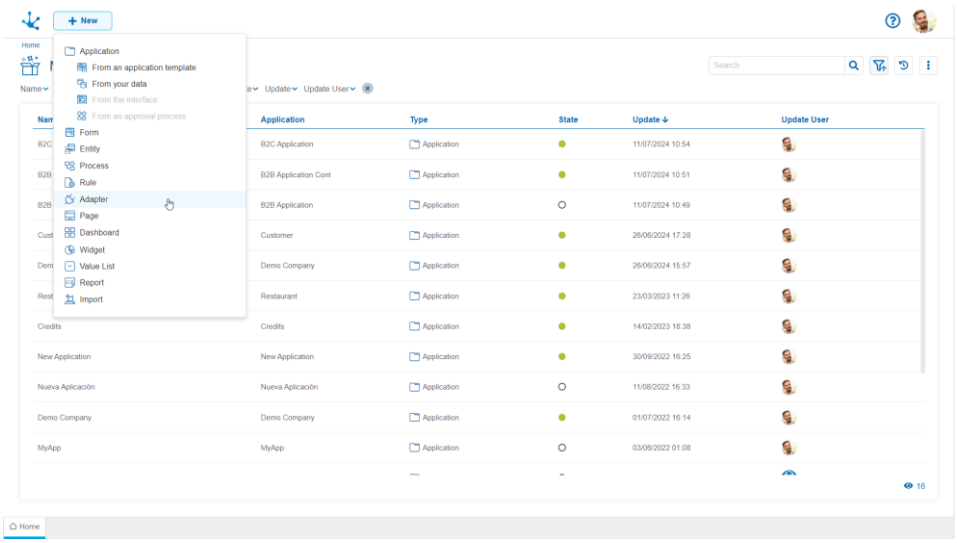
3.4.12.2. Adapters

In **Deyel** , adapters that allow integration with different applications and platforms through the use of advanced business rules are defined. Its purpose is to exchange data and share behaviors.

An adapter encapsulates the complexity of communicating with an external component. It allows different advanced rules to use it to invoke the operations that such component exposes.

When an adapter is defined, the following is configured:

- The attributes that identify the platform with which you want to interact.
- The way to establish communication.
- The operations the software component exposes.



New This button is used to define an adapter from the option Adapter.

The definition of adapters is done using the modeling facilities.

Types

Standard

Advanced rules that do not communicate with external suppliers use the adapter:

- Deyel SDK

The Deyel SDK adapter, for development and execution of business rules, allows the rules that use it to make use of the Deyel SDK facilities.

Within the standard adapters, those that allow access to web services from external suppliers are also used, in their different communication architectures:

- Rest API

- SOAP

The rules that use these adapters make invocations to specific endpoints, made up of the service URL and information about the operation to be performed. These calls can use different types of authentication, which depend on the security settings implemented by the provider.

Database

Adapters that allow to define access to relational databases through the Java JDBC protocol.

Deyel provides custom adapters for the most popular database engines on the market, facilitating configuration and optimizing interaction with such engines.

The available adapters are:

- IBM/DB2
- Informix
- MySQL
- Microsoft SQL Server
- ORACLE

The rules used by these adapters can execute SQL statements to get and write data to and from any database.

Application

These adapters allow to define access to market applications that expose services to interact with them, integrating operation between **Deyel** and such applications. Thus, the rules that use these adapters can do it safely and encapsulated.

The available adapters are:

- AWSRekognition
- AwsTextract
- Azure AD
- Docusign
- GoogleCalendar
- GoogleDrive
- IDMAuthorizationCode
- Mail Reader
- Mercado Libre
- Mercado Pago
- Niubiz
- OneDrive
- Open AI
- PayPal
- SAP
- Stripe
- Tiendanube
- Twitter

- WhatsApp Business

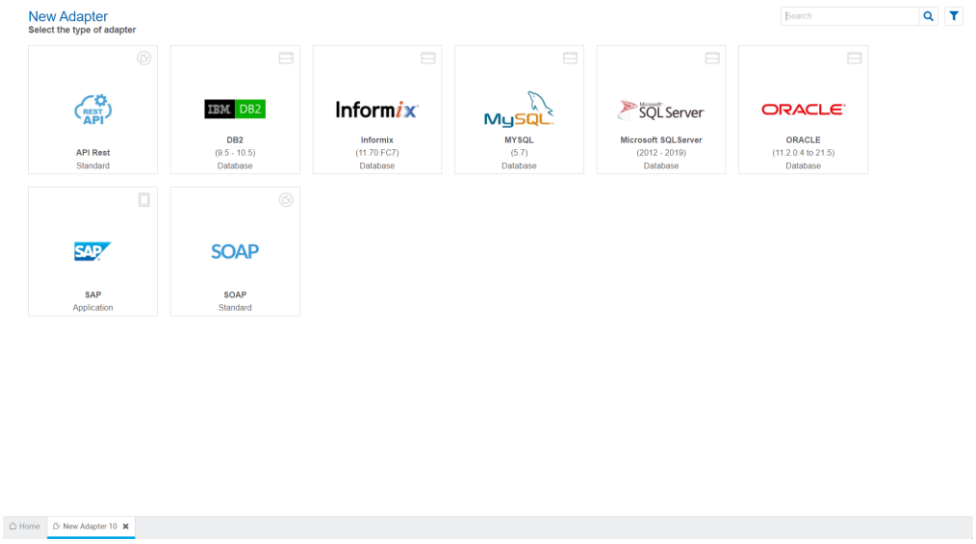
For each of the adapters, with the exception of AwsTextract and AWSRekognition, an application on each of the platforms should be created in order to establish the relation between the platform and the environment of **Deyel**. When creating this application, each platform provides the necessary values to configure the connection credentials of the corresponding adapter.

In particular, to configure the Niubiz adapter, the merchant should be registered and attached to this platform, and it should be granted the corresponding merchant code.

3.4.12.2.1. Modeling Facilities

New Adapter

To define a new adapter, the modeling user must select its type from a gallery.



In the gallery, for each adapter, the type it belongs to is displayed, and its version is displayed only if applicable. Once the type is selected, go to the properties panel of each one to define its own characteristics.

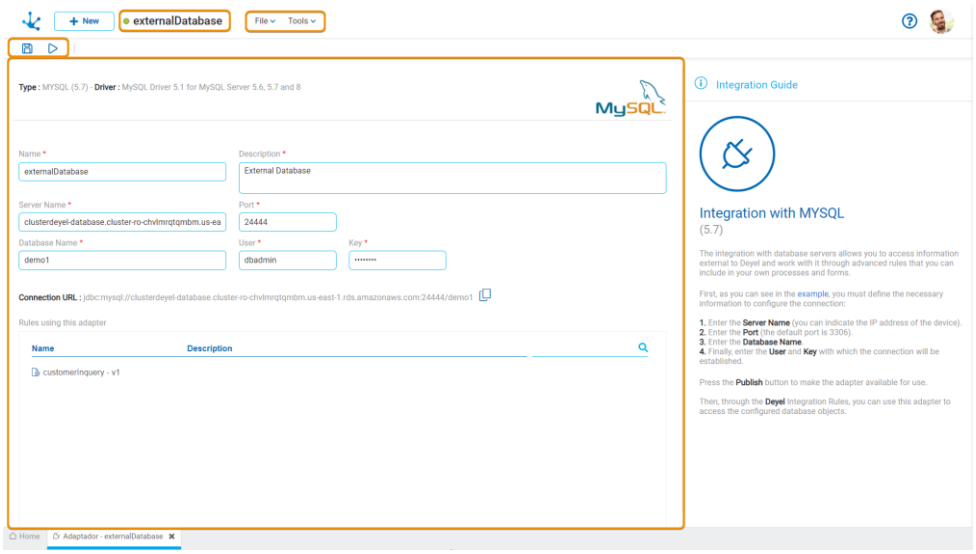
Workspace Sections:

- Adapter Information
  - State
- Name

- Expanded Menu

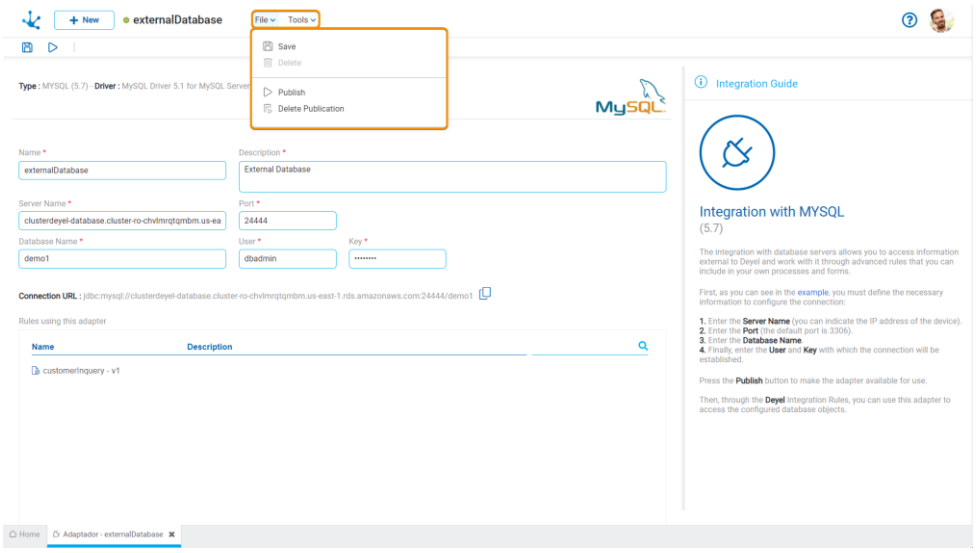
- Top Toolbar

- Adapter Definition Area



3.4.12.2.1.1. Expanded Menu

It is a horizontal list of options containing vertical submenus with different operations on the adapter or on its modeling. In turn, each option in a submenu may expand a dependent submenu.

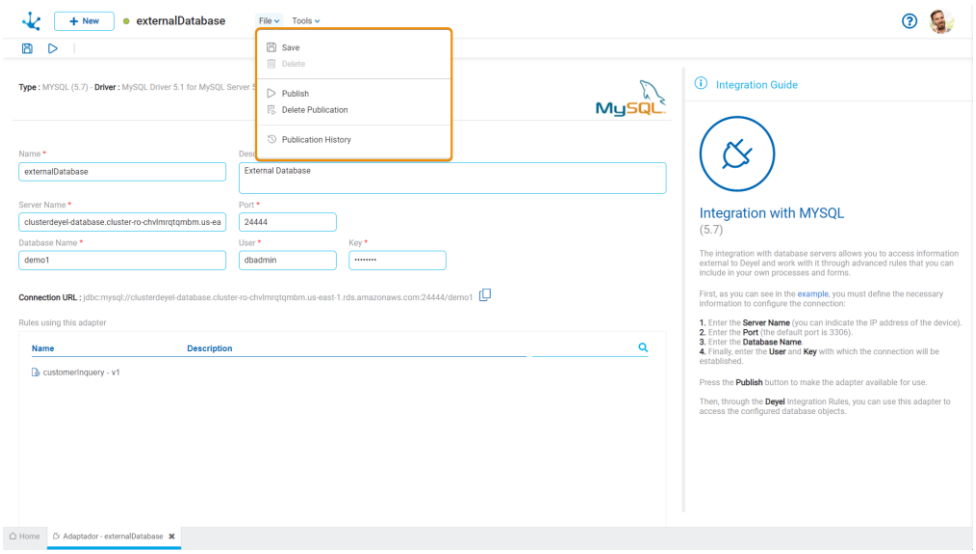


The expanded menu is made up of:

- File Submenu
- Tools Submenu

File Submenu

This submenu opens by clicking on the “File” option and allows the performance of operations on the adapter.



Save

This icon allows to save the object in the repository of **Deyel**, leaving its state as "Draft" or "Modified". If certain conditions are met, the modeler user receives a message indicating that the operation was performed correctly, otherwise they receive an explanatory message.

Conditions

- Name must be unique.
- All the fields indicated as required must be completed.

Delete

It allows to delete the object only if it is in "Draft" state and does not have other associated objects of **Deyel** that were previously saved or published. When deleted, the tab it is on closes and the object is removed from the modeler's grid.

Publish

Through this icon the object changes to "Published" state and the modeler user receives the corresponding message, indicating the result of the operation. The condition for publishing is the same as for the "Save" operation.
As for integration adapters with services from other platforms, authentication is performed. However, when it comes to database integration adapters, the connection is tested.

Conditions

- Integration adapters with services from other platforms require valid URLs, authentication methods, and other connection attributes such as users, tokens, or keys.

- Database integration adapters require valid specified values for the server name, database, schemas, users, and user password to establish the connection.

Delete Publication

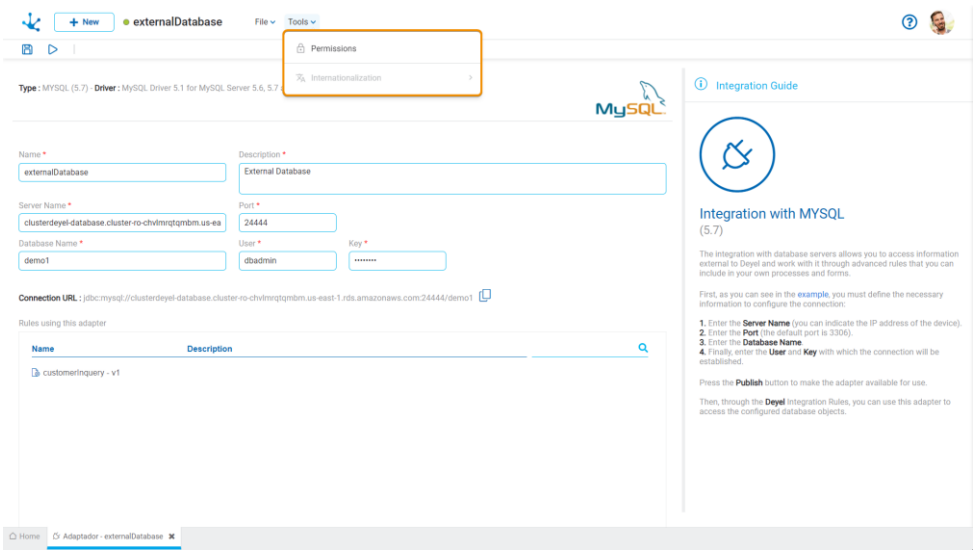
It allows disabling the adapter for use by returning it to the [state] "Draft".

Publication History

This icon allows managing versions of published objects, displaying their [history], to have better control of the changes that each object has.

Tools Submenu

This submenu opens by clicking on the "Tools" option and allows modeling adapter properties.



Permissions

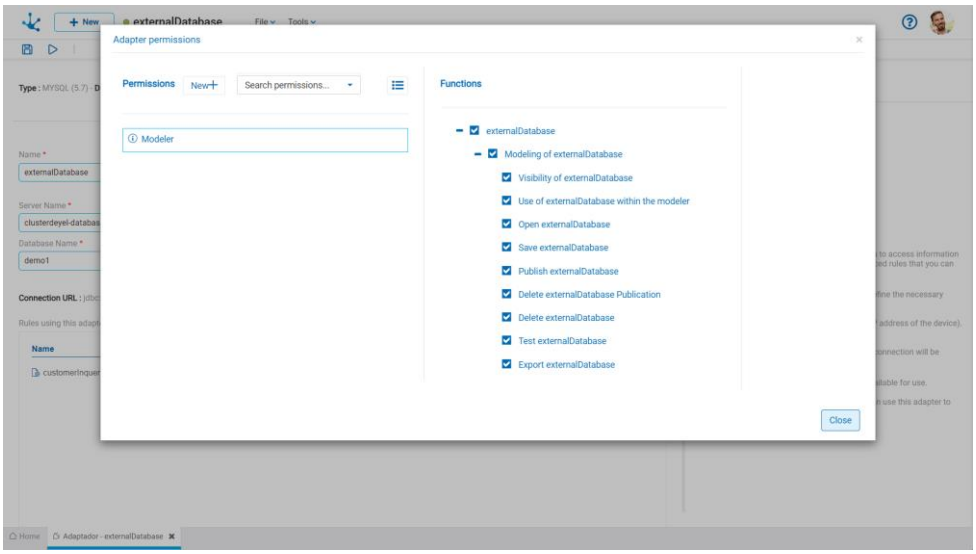
Allows to assign the [security functions] for use and modeling of the object to the existing permissions or to new permissions that the user creates, without having to go to the [permission] settings option from the menu.

Sections

- Permissions: Permissions to which object functions are assigned.
- Security Functions: Represents the total set of security functions, modeling and use of the object. Those that are marked are the ones included in the selected permission.

By default, all security functions for a new object are assigned to the permission Modeler.

Users who are assigned the permissions have access to the functions included in it.



New Opens a panel to create a new permission and once created, the security functions included in it must be selected .

Search permissions... Allows to select a permission from a list and enables the input of characters to filter the values in the list.

Opens the wizard to select a permission and once chosen, the necessary object security functions must be checked.

To unrelate a permission from the object, hover the cursor over the permission and press the icon ✖. If there are functions selected for that permission, they must be unchecked in order to delete it.

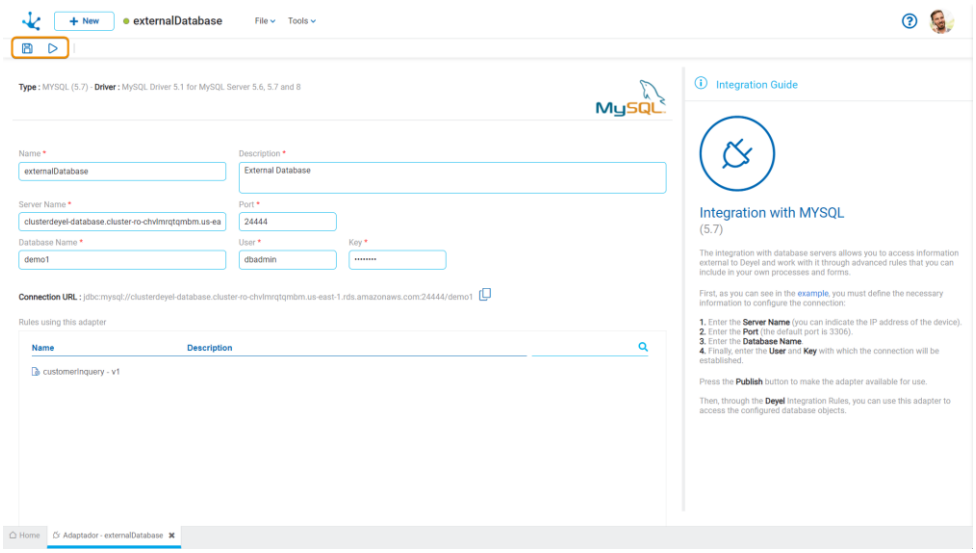
Modeling Security Functions

- Visibility: Allows to display the object in the Deyel modeler grid.
- Use within the modeler: It allows the use of the object from another modelable object.
- Open: Allows to open the object from the Deyel modeler.
- Save: Enables the operation of saving modifications made to the object.
- Publish: Enables the operation of publishing the object leaving its state as "Published".
- Delete publication: Enables the operation of deleting the object publication leaving its state as "Draft".
- Delete: Enables the operation of deleting the adapter.

- Test: Enables the operation to test the adapter.
- Export: Enables the operation to export the object from the tools submenu of the expanded menu.

3.4.12.2.1.2. Top Toolbar

Contains icons for quick access to the most used operations of the expanded menu.



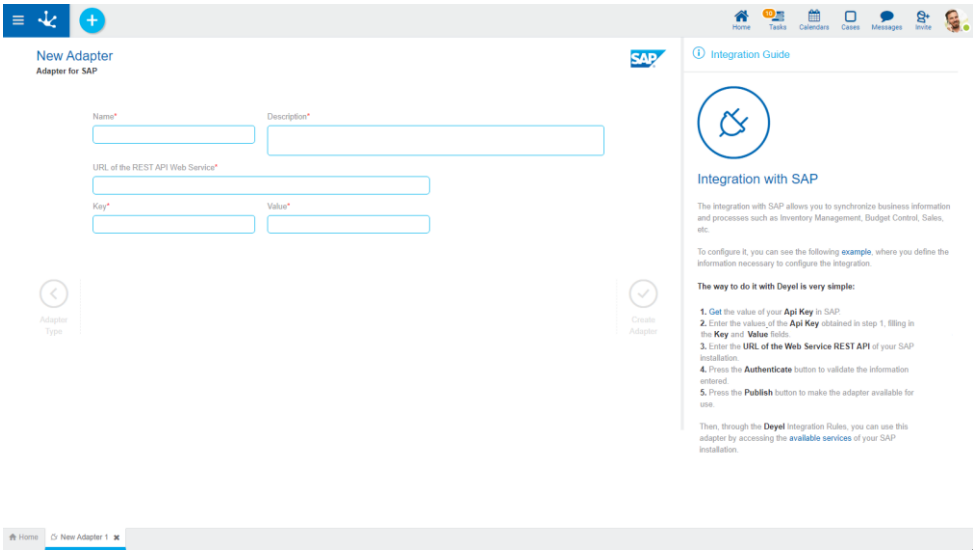
File

Save
Publish

3.4.12.2.2. Definition Area

The definition area consists of a panel with the adapter properties and an integration guide that assists and guides the modeler. Depending on each adapter, it may contain examples of use or references to external sites that facilitate configuration.

The adapter properties can be entered both at the time of their creation and when modifying an existing one. Each adapter allows configuration of specific properties, although they share some, for example the name that identifies them.



> *An asterisk "*" on the label indicates that the property is required.*

Properties

In addition to these shared properties, each type of adapter has specific properties.

Name

Name of the adapter that uniquely identifies it. It is mandatory. Identifier that is displayed in the modeler's grid and selecting the adapter when creating an advanced rule.

Description

Text that defines the adapter describing its functionality. It is mandatory. This description is displayed along with its name by selecting the adapter when creating an advanced rule.

Standard Adapters

- Rest API
- Deyel SDK
- Soap

Database Adapters

- IBM/DB2
- Informix
- Microsoft SQL Server
- MySQL
- Oracle

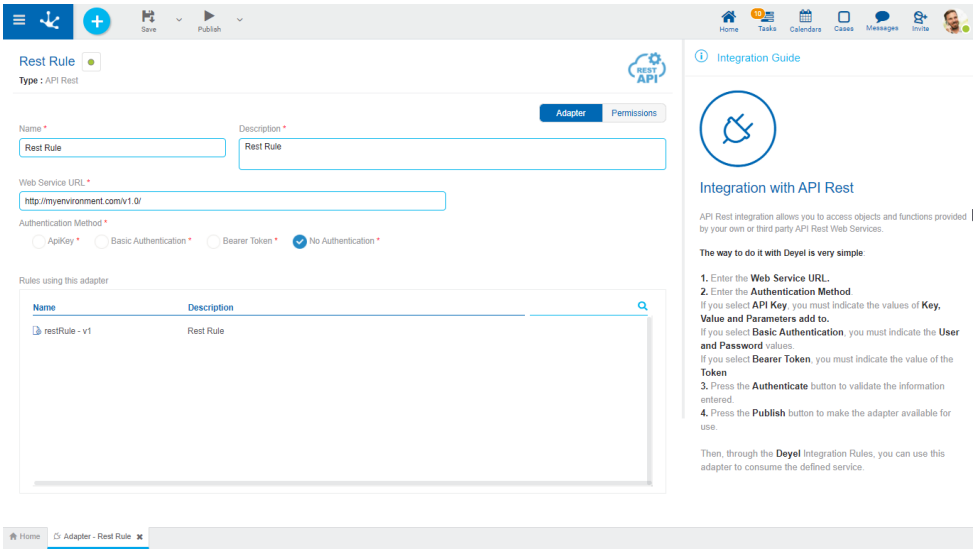
Application Adapters

- AWSRekognition
- AwsTextract

- Azure AD
- Docusign
- GoogleCalendar
- GoogleDrive
- IDMAuthorizationCode
- Mail Reader
- Mercado Libre
- Mercado Pago
- Niubiz
- OneDrive
- OpenAI
- PayPal
- SAP
- Stripe
- Tiendanube
- Twitter
- WhatsApp Business

3.4.12.2.2.1. API Rest

Apart from properties shared by adapters those specific to API Rest are added.



> *An asterisk "*" on the label indicates that the property is required.*

Web Service URL

Address of the Rest service with which it is being integrated.

Authentication Method

Identifies the way **Deyel** must authenticate with the service depending on the security scheme it implements.
The possible schemes are:

- ApiKey: in this case the service platform generates a combination of unique values called Key and Value. In addition to these values, the value for the property Parameters add to must be indicated,

where it is specified in which way the parameters should be sent to the service, it can be done in the Header or as Query Params.

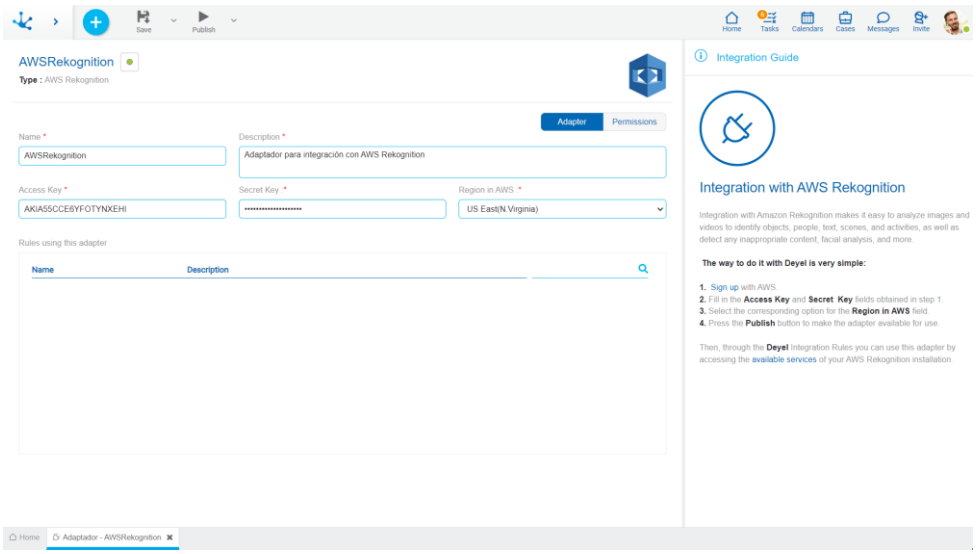
- Basic Authentication: it is the simplest authentication scheme. It must specify the User and Password, provided by the participant responsible of the service.
- Bearer Token: it is a scheme similar to ApiKey but in this case it must specify the Token, which is generated by the service.
- No Authentication: it does not require any access validation.

Rules using this adapter

In this grid, the advanced rules used by the adapter are displayed and their definition can be shown from each line.

3.4.12.2.2.2. AWSRekognition

Apart from the properties shared by adapters, those specific to AWSRekognition are added.



> *An asterisk "*" on the label indicates that the property is required.*

Access Key

Identification key of the service that AWSRekognition issues when creating an application on its platform.

Secret Key

Secret key associated to the property Access Key, are used together when requesting access to the resources provided by the platform.

Region on AWS

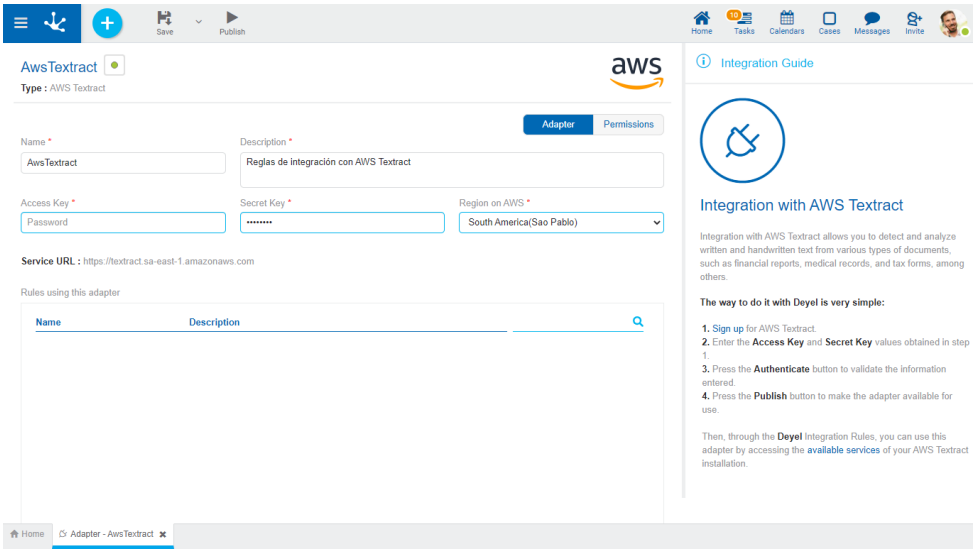
Indicates the location of the services to be used. The region determines the transaction processing quotas, based on the AWS documentation.

Rules using this adapter

In this grid, the advanced rules used by the adapter are displayed and their definition can be shown from each line.

3.4.12.2.2.3. AwsTextract

Apart from properties shared by adapters those specific to AwsTextract are added.



> *An asterisk "*" on the label indicates that the property is required.*

Access Key
Identification key of the service that AwsTextract issues when creating an application on its platform.

Secret Key
Secret key associated to the property Access Key, are used together when requesting access to the resources provided by the platform.

Region on AWS
Indicates the location of the services to be used. The region determines the transaction processing quotas, based on the documentation.

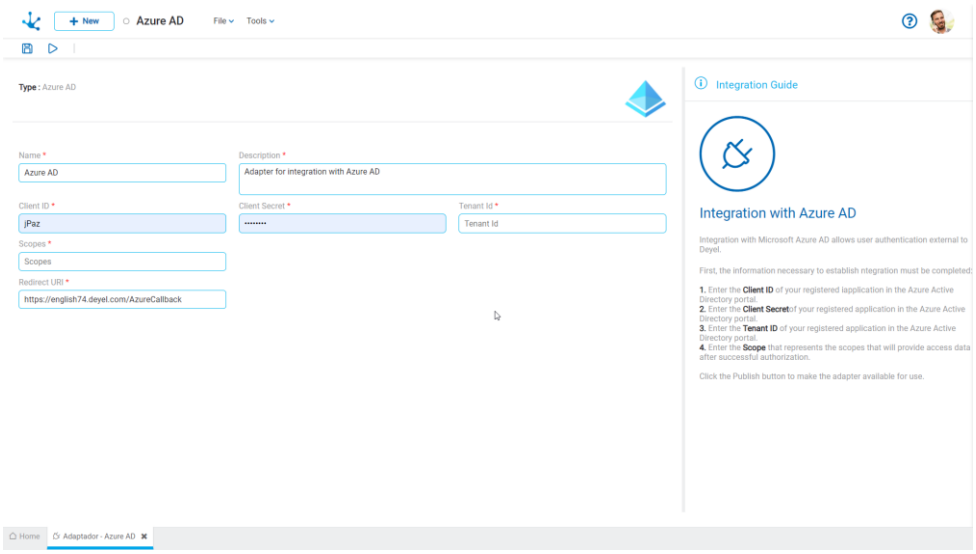
Service URL
This property is filled in automatically when selecting the Region on AWS property.

Rules using this adapter
In this grid, the advanced rules used by the adapter are displayed and their definition can be shown from each line.

3.4.12.2.2.4. Azure AD

This adapter must be configured when the user authentication is delegated to Microsoft Azure AD. For this, register an application in the Azure AD portal and get the required credentials.

Apart from the properties shared by adapters , those specific to Azure AD are added.



> *An asterisk "*" on the label indicates that the property is required.*

Client ID
Identification of the service that Azure AD issues when creating an application on its platform.

Client Secret
Secret key associated to the Client ID property, they are used together when requesting access to the resources provided by the platform.

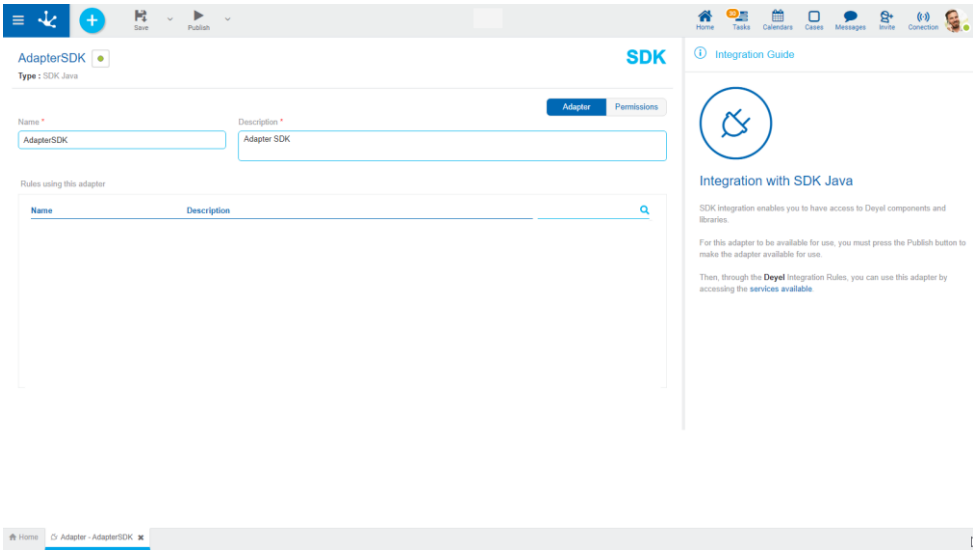
Tenant Id
Identifier of the directory where the application is registered in the Azure AD portal.

Scopes
Represents the scopes provided by the access data after successful authorization.

Redirect URL
Indicates the redirect URL that Microsoft responds to when the user is successfully authenticated. It is not editable.

3.4.12.2.2.5. Deyel SDK

Apart from properties shared by adapters the grid of rules used by the adapter is added.



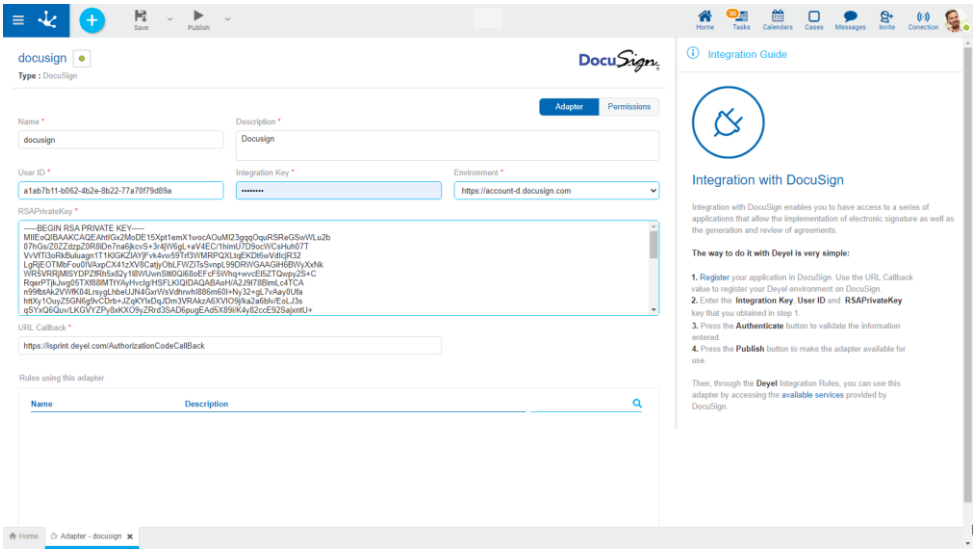
An asterisk "" on the label indicates that the property is required.*

Rules using this adapter

In this grid, the advanced rules used by the adapter are displayed and their definition can be shown from each line.

3.4.12.2.2.6. DocuSign

Apart from properties shared by adapters those specific to DocuSign are added.



An asterisk "" on the label indicates that the property is required.*

User ID

Identification of the user account registered in DocuSign.

Integration Key

Identification key of the service that DocuSign issues when creating an application on its platform.

Environment
Corresponds to the DocuSign URL against which APIRest operations are performed, it can be the production environment or the development environment. This URL can be obtained in the properties of the application registered in DocuSign under the name "Account Base URI".

RSAPrivateKey
Secret key associated to the Integration Key property, they are used together when requesting access to the resources provided by the platform.

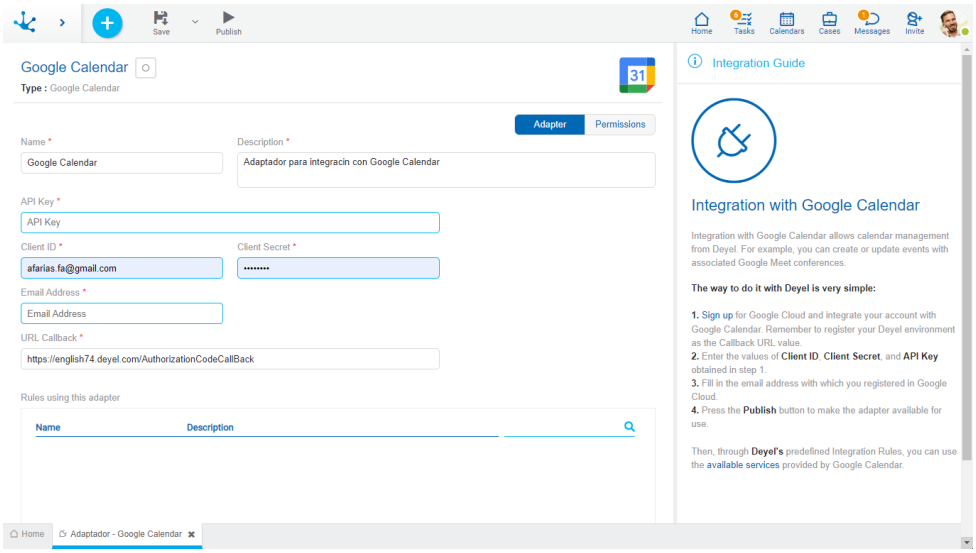
URL Callback
This property must be defined when creating the application on DocuSign and corresponds to the URL that handles the return of requests for services that are made from the **Deyel** environment to the DocuSign platform.
The value for this property must have the name of the environment Deyel https://.deyel.com/AuthorizationCodeCallBack.

Rules using this adapter
In this grid, the advanced rules used by the adapter are displayed and their definition can be shown from each line.

3.4.12.2.2.7. GoogleCalendar

Integration with Google Calendar allows to manage calendars from **Deyel**. For example, events can be created or updated with associated conferences via Google Meet.

Apart from the properties shared by adapters , those specific to GoogleCalendar are added.



An asterisk "" on the label indicates that the property is required.*

API Key
Access key to the service issued by GoogleCloud when creating an application on its platform.

Client ID
Identification of the service that GoogleCloud issues when creating an application on its platform.

Client Secret
Secret key associated to the Client ID property, they are used together when requesting access to the resources provided by the platform.

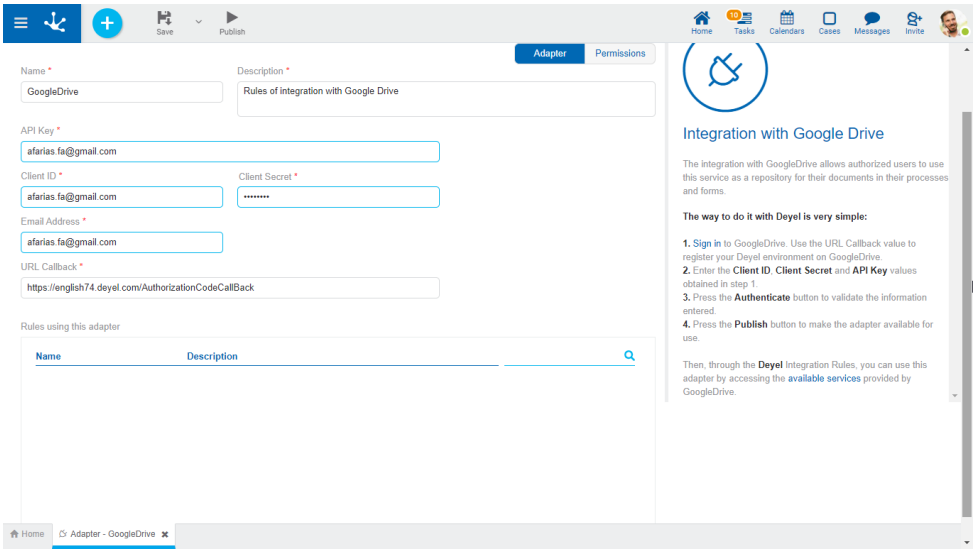
Email Address
Account associated and owner of the GoogleCloud space.

URL Callback
This property should be defined when creating the application on Google Cloud and corresponds to the URL that handles the return of requests for services that are made from the **Deyel** environment to the Google Calendar platform.
The value for this property must have the name of the **Deyel** environment:
https://<miambiente>.deyel.com/AuthorizationCodeCallBack.

It is a predefined, non-editable value.

3.4.12.2.2.8. GoogleDrive

Apart from properties shared by adapters those specific to GoogleDrive are added.



> *An asterisk "*" on the label indicates that the property is required.*

API Key
Access key to the service that Google Drive issues when creating an application on its platform.

Client ID
Identification of the service that Google Drive issues when creating an application on its platform.

Client Secret
Secret key associated to the Client ID property, they are used together when requesting access to the resources provided by the platform.

Email Address
Associated account and owner of the Google Drive space.

URL Callback
This property should be defined when creating the application on Google Drive and corresponds to the URL that handles the return of requests for services that are made from the **Deyel** environment to the Google Drive platform.
The value for this property must have the name of the **Deyel** environment:
https://<miambiente>.deyel.com/AuthorizationCodeCallBack.

It is a predefined, non-editable value.

Rules using this adapter
In this grid, the advanced rules used by the adapter are displayed and their definition can be shown from each line.

Predefined Rules

There are predefined rules in **Deyel** to use the adapter, they perform operations using the Google Drive API.

Name	Description	Parameters
insertFileInGoogleDrive	Adds a file within the Google Drive account configured on the adapter.	Input • **pFile**: File to be added in Google Drive. • **parentId**: Identifier of the directory in which the file is added. It can be obtained from the URL of the current Google Drive directory. Output • **fileId**: Identifier of the file added in Google Drive.
insertNewFileInGoogleDrive	Adds a file within the Google Drive account. Determines whether its access is public or not, by assigning a permission type and a role to a specific user.	Input • **pFile**: File to be added in Google Drive. • **parentId**: Identifier of the directory in which the file is added. It can be obtained from the URL of the current Google Drive directory.

Name	Description	Parameters
		• **type**: Indicates the type of permission being added. Possible values are: "user" or "anyone". • **role**: Indicates the role associated to the type of permission being added. Possible values are: "writer", "reader". • **cdUser**: User code of **Deyel** to whom permission is granted. • **title**: File name with which it is added in Google Drive. If the parameter is not filled in, the file is added with the original file name. Output • **fileId**: Identifier of the file added in Google Drive. • **alternateLink**: URL obtained from the added file to be used according to the permissions obtained. • **permissionId**: Identifier of the permission associated to the file added in Google Drive.
deleteFileInGoogleDrive	Deletes a certain file from the Google Drive account.	Input • **fileId**: Identifier of the file stored in Google Drive to be deleted.
getFileFromGoogleDrive	Gets the link of a certain file from the Google Drive account.	Input • **fileId**: Identifier of the file stored in Google Drive to be obtained. Output • **alternateLink**: URL of the file obtained to be used according to the permissions of the current user.
createCopyFileFromGoogleDrive	Makes a copy of a certain file from the Google Drive	Input

Name	Description	Parameters
	account. Its name should be provided. By default, it is generated with the name of the original file, concatenating the "-copy" value.	• **fileId**: Identifier of the file stored in Google Drive to be copied. • **nameOfCopy**: Indicates the name under which the file copy is generated. This parameter is optional. If it has no value, the name of the copied file will be that of the original file concatenated with "-copy". Output • **alternativeLinkOfCopiedFile**: URL of the file copied to be used according to the permissions of the current user. • **fileIdCopied**: Identifier of the file copied in Google Drive.
createPermissionInGoogleDriveFile	Adds a permission, either to read or to read and write, to a specific file in the Google Drive account for a specific user.	Input • **fileId**: Identifier of the file stored in Google Drive to be added a permission. • **type**: Indicates the type of permission being added. Possible values are: "user" or "anyone". • **role**: Indicates the role associated to the type of permission generated. Possible values are: "writer", "reader". • **cdUser**: Code of **Deyel** user to whom permission is granted. Output • **permissionId**: Identifier of the permission created and associated to the Google Drive file.
deletePermissionInGoogleDriveFile	Deletes permissions for a specific user on a certain file in the Google Drive account.	Input • **fileId**: Identifier of the file stored in Google Drive which permissions are to be deleted.

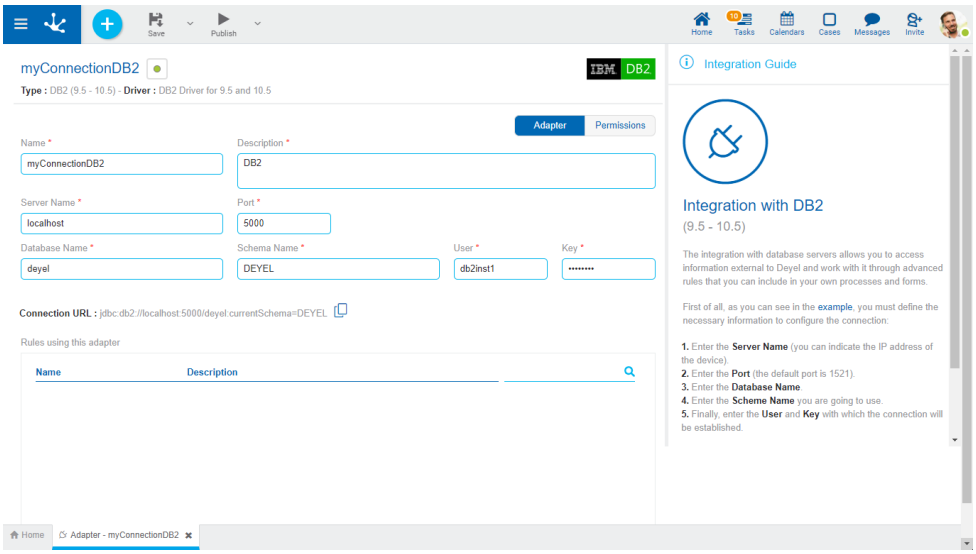
Name	Description	Parameters
		• **cdUser**: Code of **Deyel** user whose permissions are removed.
getPermissionFromGoogleDriveFile	Gets a permission associated with a certain file in the Google Drive account.	Input • **fileId**: Identifier of the file stored in Google Drive whose permission is to be obtained. • **permissionId**: Identifier of the permission associated to the Google Drive file. Output • **permissionType**: Indicates the type of permission. Possible values are: "user", "group", "domain", "anyone". • **permissionRole**: Indicates the permission role. Possible values are: "owner", "organizer", "fileOrganizer", "writer", "commenter", "reader". • **permissionUser**: Indicates the user to which the permission is associated.
getPermissionsFromGoogleDriveFile	Gets all permissions associated with a certain file in the Google Drive account.	Input • **fileId**: Identifier of the file stored in Google Drive whose permissions are to be obtained. Output • **listOfPermissions**: List containing the permissions associated with the Google Drive file.
exportToPdfGoogleDriveFile	Generates a temporary file in PDF format from a certain file stored in Google Drive. The name of the PDF file can be indicated through the fileName parameter, by default it is the identifier concatenated with the extension ".pdf".	Input • **fileId**: Identifier of the file stored in Google Drive to be generated in PDF format. • **fileName**: Name used for the file generated as PDF.

Name	Description	Parameters
		Output • **pdfFile**: File obtained from Google Drive in PDF format.

> *These rules are in "Draft" state and should be published by a user with administrator permissions, The Google Drive adapter should be previously published.*

3.4.12.2.2.9. IBM/DB2

Apart from properties shared by adapters those specific to IBM/DB2 are added.



> *An asterisk "*" on the label indicates that the property is required.*

Server Name
Name or IP address of the server where the database engine is installed.

Port
Port number used by the database engine.

Database Name
Corresponds to the identification of the database with which you want to integrate.

Schema Name
Name of the schema within the database with which you want to integrate.

User
Identification of the database user that has permissions on the schema.

Key
Password of the database user making the connection.

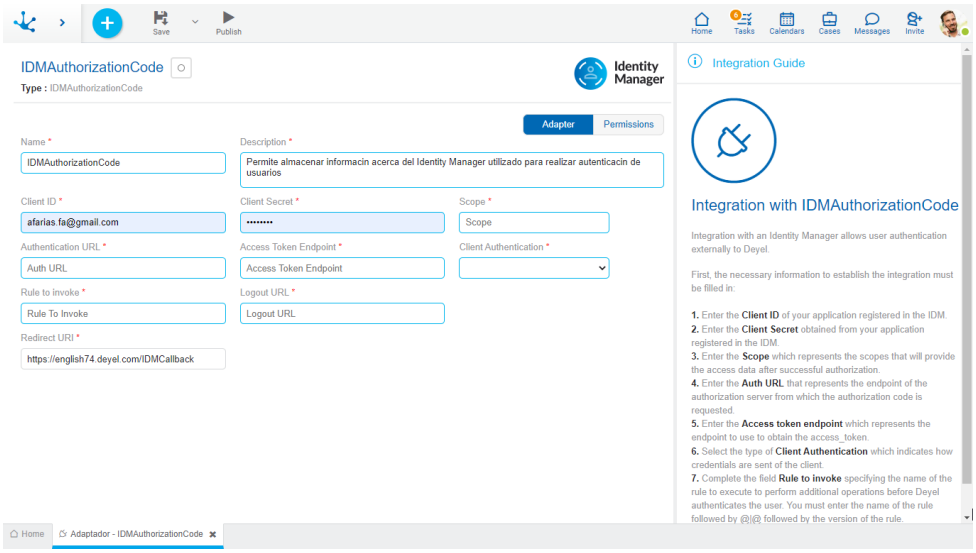
Connection URL
Shows the connection string for the adapter.

Allows to copy the connection URL.

Rules using this adapter
In this grid, the advanced rules used by the adapter are displayed and its definition can be shown from each line.

3.4.12.2.2.10. IDMAuthorizationCode

It is used to store properties when user authentication is performed on an external Identity Manager using the Authorization Code flow.



> *An asterisk "*" on the label indicates that the property is required.*

Authentication URL
Represents the endpoint of the authorization server to which the authorization code is requested.

Access Token Endpoint
Indicates the endpoint to which the access token request is made.

Client Id
Indicates the Client id of the application registered in the IDM.

Client Secret
Indicates the Client Secret of the application registered in the IDM.

Redirect URI

It is not editable. Indicates the redirection URL to which the IDM responds with the code authorization and the corresponding access token.

Scope
Represents the scopes provided by the access data after successful authorization.

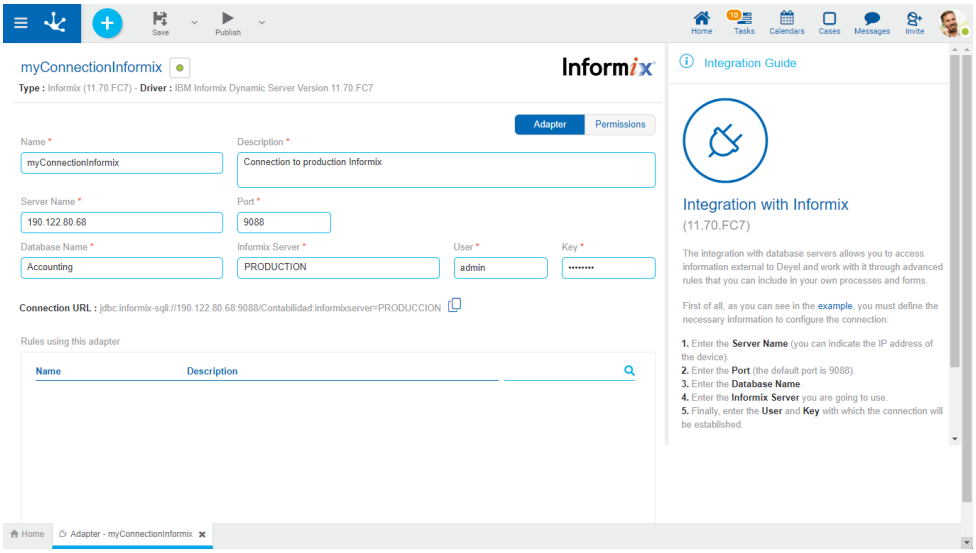
Rule to Invoke
It is used to indicate the rule that performs an additional operation with the access token before **Deyel** authenticates the user internally. This rule must be published before the adapter. It must respect the String data type parameter named "access_token" as input parameter. The user that is registering is obtained.

Client Authentication
Indicates how client credentials are sent. The options are:
- Basic Auth Header
- Client Credentials in Body

Logout URL
Indicates the URL where the user is automatically redirected to close the IDM session, once the **Deyel** session is closed.

3.4.12.2.2.11. Informix

Apart from properties shared by adapters those specific to Informix are added.



> *An asterisk "*" on the label indicates that the property is required.*

Server Name
Name or IP address of the server where the database engine is installed.

Port
Port number used by the database engine.

Database Name

Corresponds to the identification of the database with which you want to integrate.

Informix Server
Name of the server that manages the database with which you want to integrate.

User
Identification of the database user that has permissions on the schema.

Key
Password of the database user making the connection.

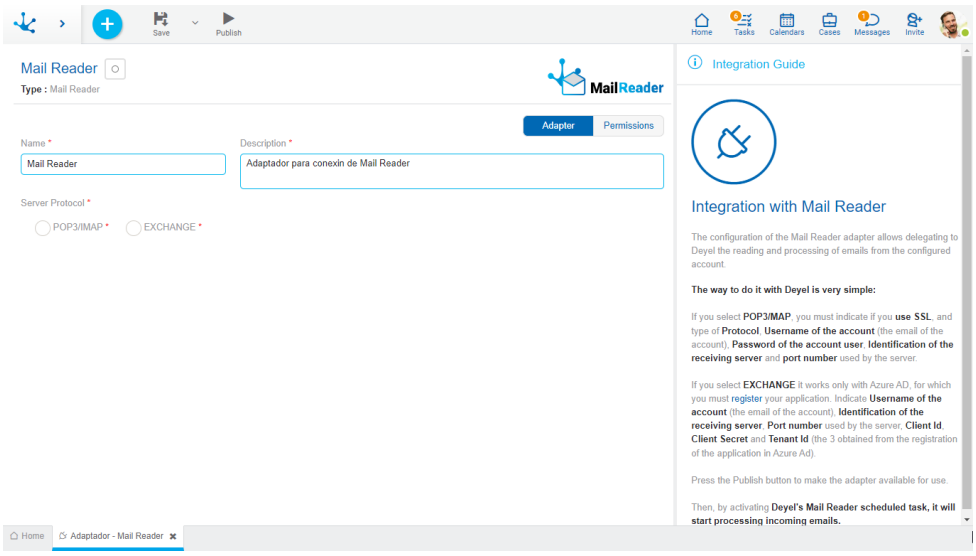
Connection URL
Shows the connection string for the adapter.

Allows to copy the connection URL.

Rules using this adapter
In this grid, the advanced rules used by the adapter are displayed and their definition can be shown from each line.

3.4.12.2.2.12. Mail Reader

Apart from the properties shared by adapters, those specific to Mail Reader are added.



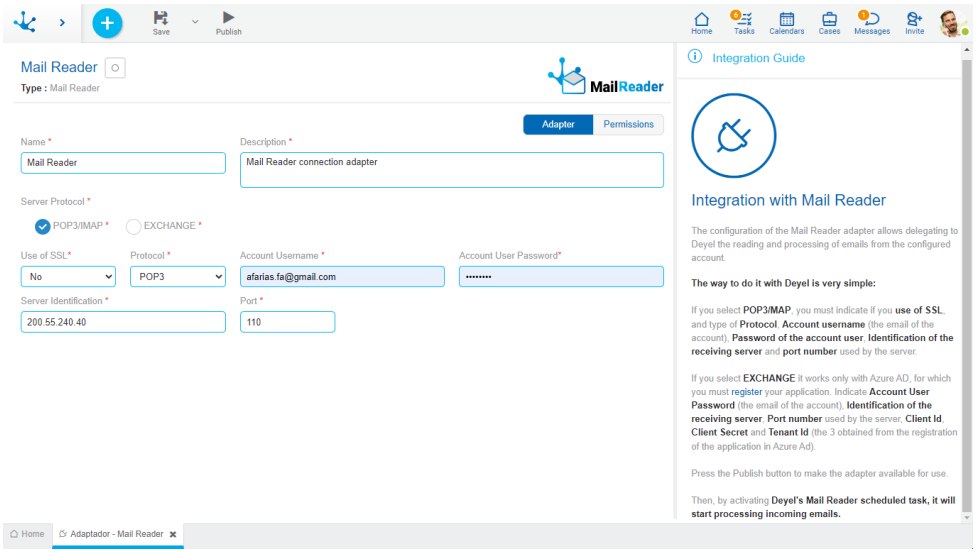
> *An asterisk "*" on the label indicates that the property is required.*

The adapters modeling allows to define which protocol is used for reading emails.

- POP3/IMAP
- EXCHANGE

Different properties are enabled depending on the selected option.

POP3/IMAP protocol



> *An asterisk "*" on the label indicates that the property is required.*

Use of SSL
Determines if the SSL protocol is used for sending emails.

Protocol
Allows to select whether the connection is under the IMAP or POP3 protocol.

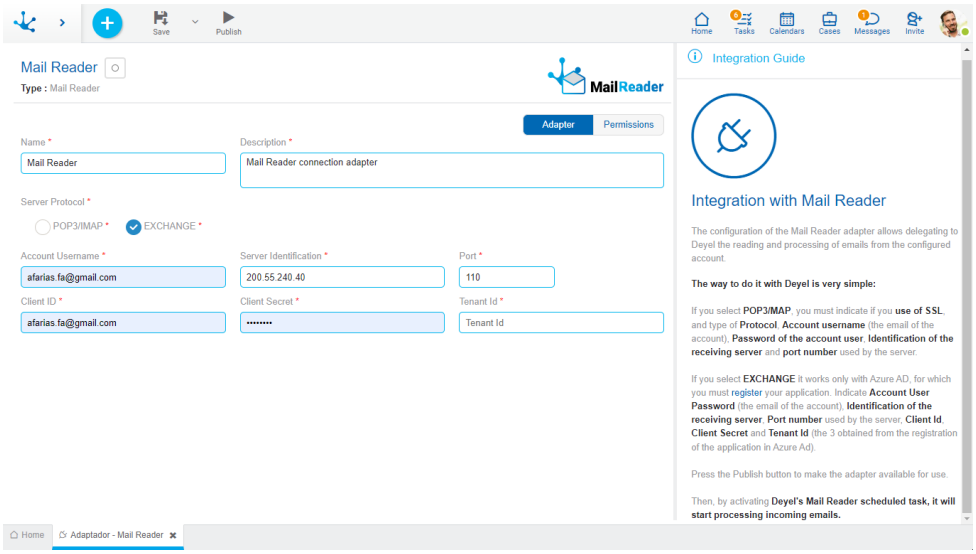
Account Username
Defines the account where the reading of emails is carried out.

Account User Password
Secret key associated to the Account Username property, they are used together to establish the connection with the selected protocol.

Server Identification
Identifies whether the server with which the connection is established is IMAP or POP3.

Port
Port number for accessing the IMAP or POP3 server.

EXCHANGE protocol



An asterisk "" on the label indicates that the property is required.*

Server Identification
It must be completed with the value: "outlook.office365.com".

Account Username
Defines the account where the reading of emails is carried out.

Port
Port number for accessing the EXCHANGE server with which the connection is established.

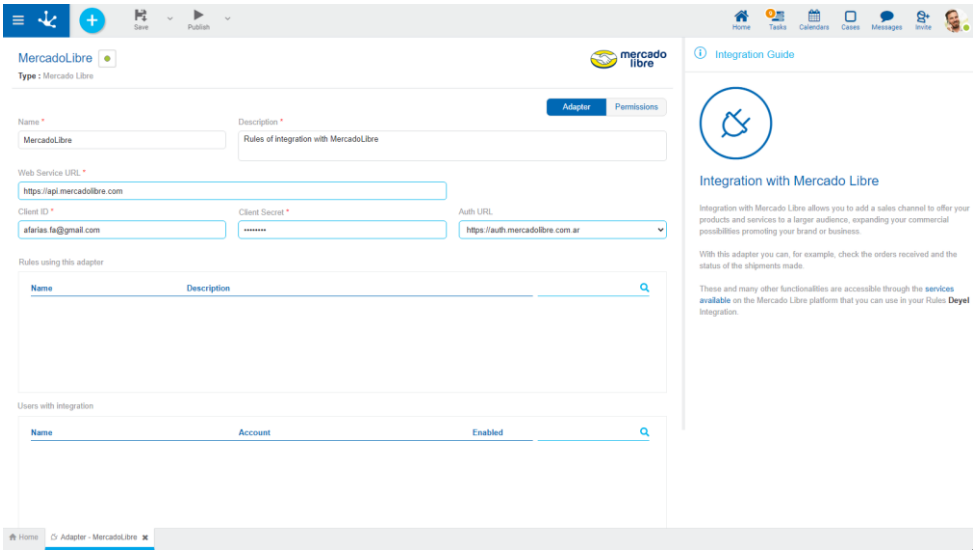
Client Id
Identification of the application registered in the Google Cloud developer console.

Client Secret
Secret key associated to the Client ID, they are used together to perform user authentication and subsequent use of the application.

Tenant Id
Indicates the id of the directory where the application is registered in the Google Cloud developer console.

3.4.12.2.2.13. Mercado Libre

Apart from properties shared by adapters those specific to Mercado Libre are added.



An asterisk "" on the label indicates that the property is required.*

Web Service URL

Corresponds to the URL that attends requests for services that are made from the **Deyel** environment to the Mercado Libre platform.

Client ID

Identification of the service that Mercado Libre issues when creating an application on its platform.

Client Secret

Secret key associated to the Client ID property, they are used together when requesting access to the resources provided by the platform.

Auth URL

Corresponds to the Mercado Libre URL against which the authentication must be performed.

Rules using this adapter

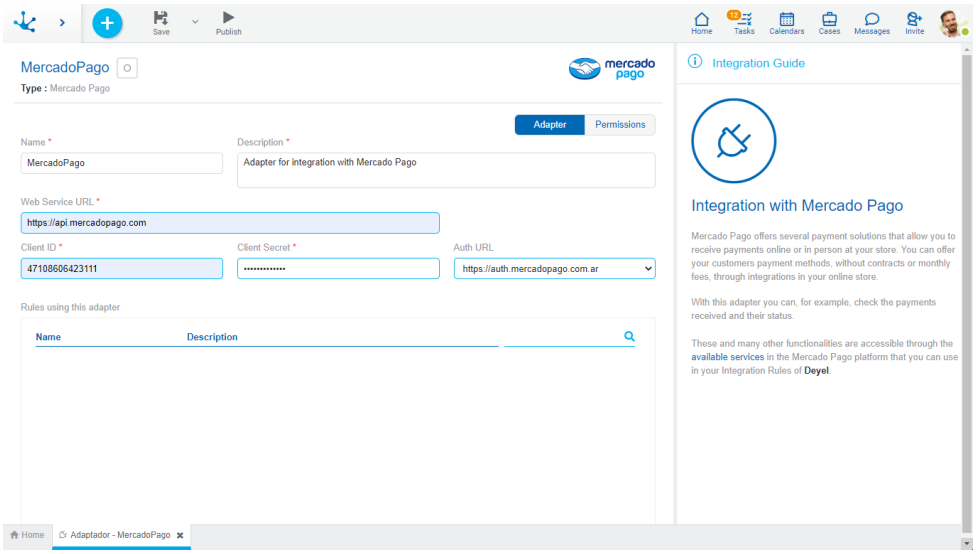
In this grid, the advanced rules used by the adapter are displayed and their definition can be shown from each line.

Users with integration

In this grid you can see the users who have the Mercado Libre integration configured and the state of each one, which can be enabled or disabled. The state can be modified from each line.

3.4.12.2.2.14. Mercado Pago

Apart from properties shared by adapters those specific to Mercado Pago are added.



> *An asterisk "*" on the label indicates that the property is required.*

Web Service URL

Corresponds to the URL that handles requests for services that are made from the **Deyel** environment to the Mercado Pago platform.

Client ID

Identification of the service that Mercado Pago issues when creating an application on its platform.

Client Secret

Secret key associated to the Client ID property, they are used together when requesting access to the resources provided by the platform.

Auth URL

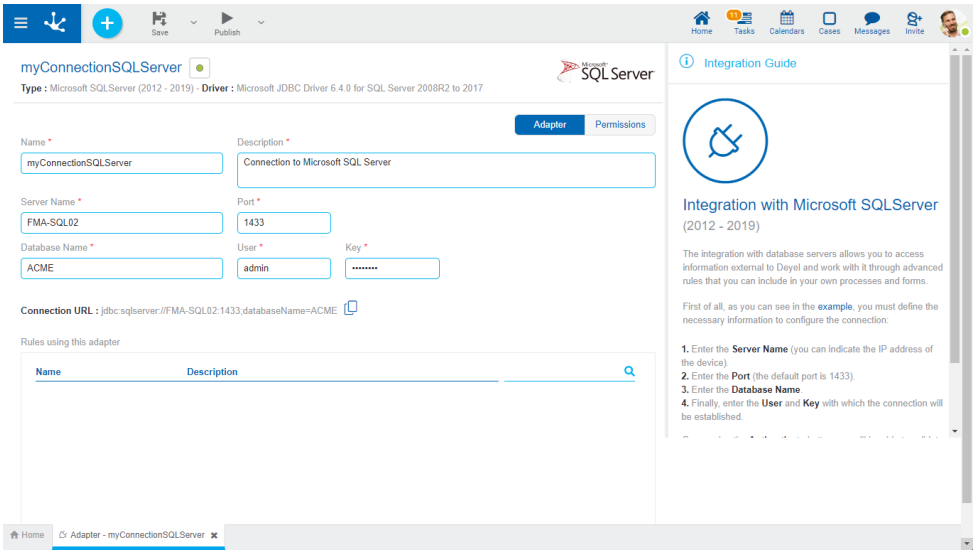
Corresponds to the Mercado Pago URL against which the authentication must be performed.

Rules using this adapter

In this grid, the advanced rules used by the adapter are displayed and their definition can be shown from each line.

3.4.12.2.2.15. Microsoft SQL Server

Apart from properties shared by adapters those specific to Microsoft SQL Server are added.



An asterisk "" on the label indicates that the property is required.*

Server Name
Name or IP address of the server where the database engine is installed.

Port
Port number used by the database engine.

Database Name
Corresponds to the identification of the database with which you want to integrate.

User
Identification of the database user that has permissions on the schema.

Key
Password of the database user making the connection.

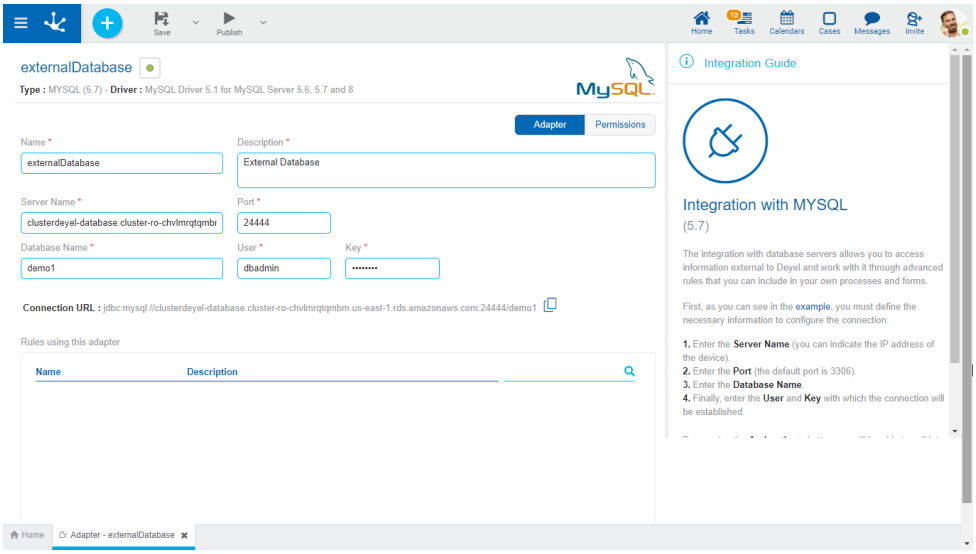
Connection URL
Shows the connection string for the adapter.

Allows to copy the connection URL.

Rules using this adapter
In this grid, the advanced rules used by the adapter are displayed and their definition can be shown from each line.

3.4.12.2.2.16. MySQL

Apart from properties shared by adapters those specific to MySQL are added.



An asterisk "" on the label indicates that the property is required.*

Server Name
Name or IP address of the server where the database engine is installed.

Port
Port number used by the database engine.

Database Name
Corresponds to the identification of the database with which you want to integrate.

User
Identification of the database user that has permissions on the schema.

Key
Password of the database user making the connection.

Connection URL
Shows the connection string for the adapter.

Allows to copy the connection URL.

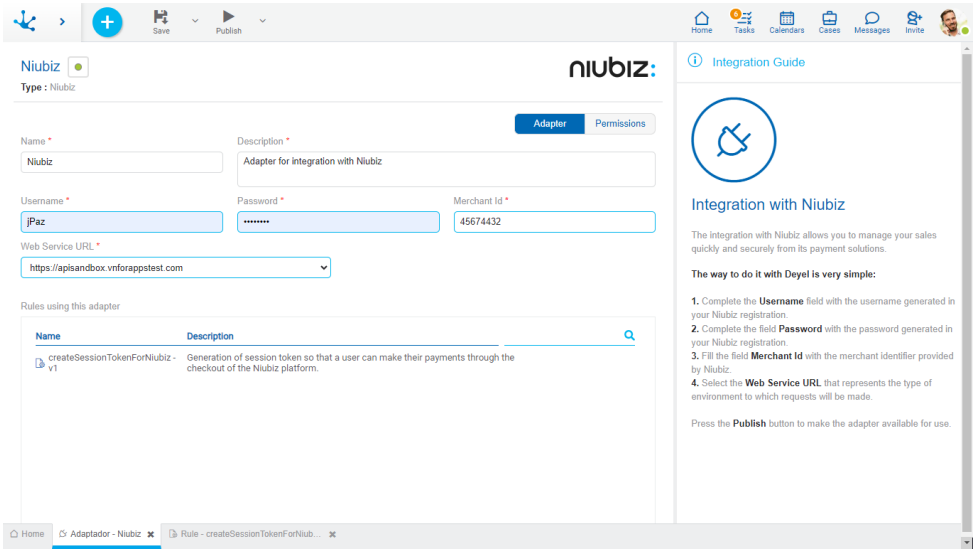
Rules using this adapter
In this grid, the advanced rules used by the adapter are displayed and their definition can be shown from each line.

3.4.12.2.2.17. Niubiz

Integration with Niubiz allows managing sales quickly and securely through its payment solutions.

For its configuration, the merchant that performs the integration should be registered and attached to the Niubiz platform and thus have a merchant code assigned.

Apart from the properties shared by adapters those specific to Niubiz are added.



An asterisk "" on the label indicates that the property is required.*

Username

Username generated when registering in Niubiz. It is used together with the Password attribute to generate the corresponding access token.

Password

Password generated when registering in Niubiz. It is used together with the Username attribute to generate the corresponding access token.

Merchant Id

Merchant identifier provided by Niubiz upon registration.

Web Service URL

It represents the type of environment to which the requests are made and can be selected between a productive or sandbox environment.

Built-in Rules

In **Deyel**, there is a built-in rule that performs operations using the Niubiz API and provides the corresponding session token to use the payment checkout of the platform.

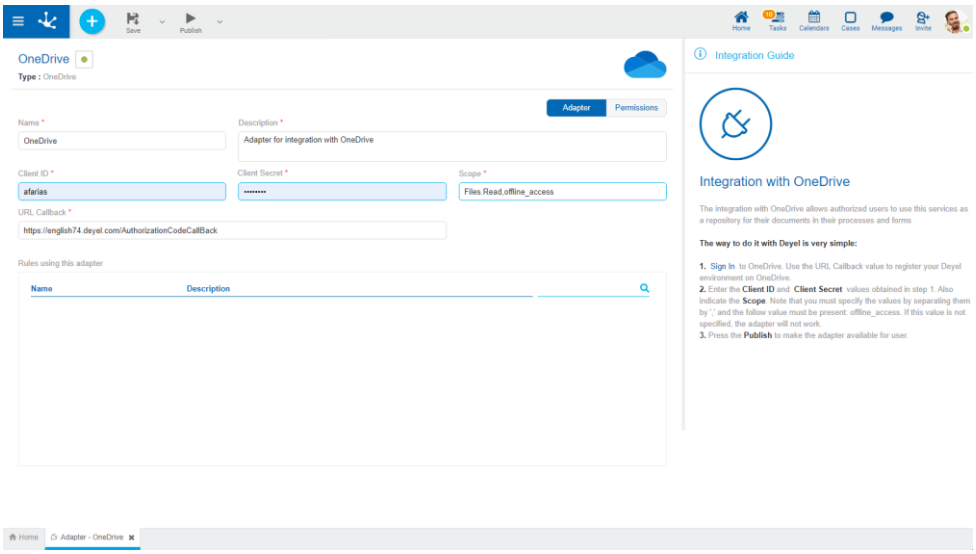
Name	Description	Parameters
createSessionTokenFor-Niubiz	Generates the session token, for a user to perform their payments through the Niubiz platform checkout.	Input • **pAmount**: amount to show and pay through the Niubiz checkout.

Name	Description	Parameters
		• **pMerchantDefineData**: values assigned by Niubiz for each merchant to form the anti-fraud object. Output • **pSessionToken**: session token generated to instantiate the Niubiz checkout. • **pMerchantId**: merchant code. It is used to instantiate the payment checkout.

> *This rule is in "Draft" state and should be published by a user with administrator permissions, The Niubiz adapter should be previously published.*

3.4.12.2.2.18. OneDrive

Apart from properties shared by adapters those specific to OneDrive are added.



> *An asterisk "*" on the label indicates that the property is required.*

Client ID

Identification of the service that OneDrive issues when creating an application on its platform.

Client Secret

Secret key associated to the Client ID property, they are used together when requesting access to the resources provided by the platform.

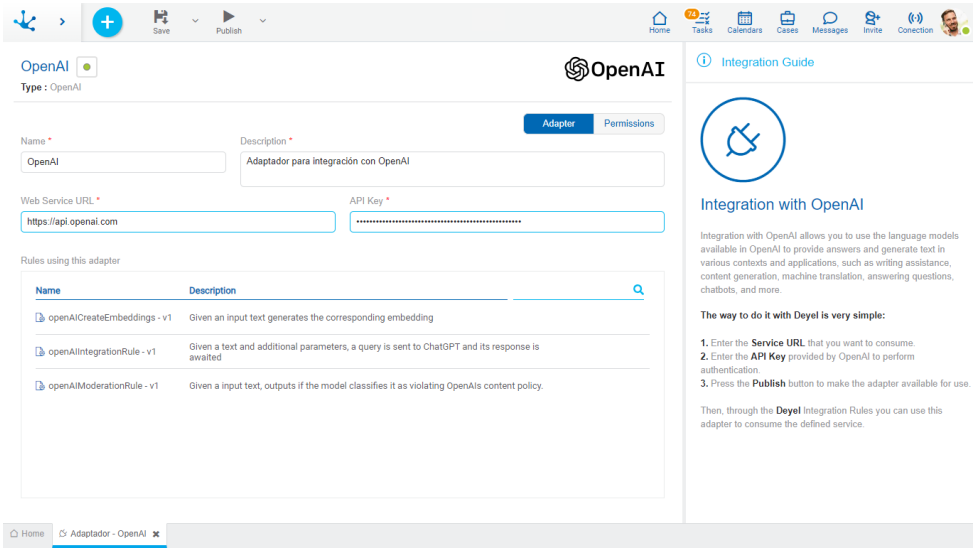
Scope
Determines what type of access is granted to the application when the user logs in, based on documentation.

URL Callback
This property must be defined when creating the application on OneDrive and corresponds to the URL that handles the return of requests for services that are made from the **Deyel** environment to the GoogleDrive platform.
The value for this property must be composed with the name of the environment Deyel https://.deyel.com/AuthorizationCodeCallBack.

Rules using this adapter
In this grid, the advanced rules used by the adapter are displayed and their definition can be shown from each line.

3.4.12.2.2.19. OpenAI

Apart from the properties shared by adapters those specific to OpenAI are added.



> *An asterisk "*" on the label indicates that the property is required.*

Web Service URL
Corresponds to the URL that handles requests for services that are made from the **Deyel** environment to the OpenAI platform.

API Key
It is the access key granted by OpenAI to perform user authentication.

Rules using this adapter
In this grid, the advanced rules used by the adapter are displayed and their definition can be shown from each line.

Built-in Rules

There are built-in rules to use the adapter in **Deyel** that perform operations using the OpenAI API.

Name	Description	Parameters
openAICreateEmbeddings	Given an input text generates the corresponding embedding.	Input • **input**: Text from which the embedding is generated. • **model**: Model identifier to use. It can use text-embedding-ada-002 or text-search-ada-doc-001. By default, text-embedding-ada-002 is used. Output • **response**: Embedding generated.
openAIIntegrationRule	Given a text and additional parameters, a query is sent to ChatGPT and its answer is awaited.	Input • **model**: Model identifier to use. It can use any provided by OpenAI. • **messages**: List of messages indicating the context of the conversation. Each message should keep to the format required by OpenAI where the role and content must be specified. • **name**: Name of the author of the message. • **temperature**: Value indicating randomness in the response by OpenAI. Values correspond to those indicated in its API Reference. • **topp**: Alternative to the temperature parameter. Either one should be used. • **n**: Indicates the number of possible answers for the chat indicated in the message list. • **maxtokens**: Maximum number of tokens to generate in the chat completion. It is limited by the model

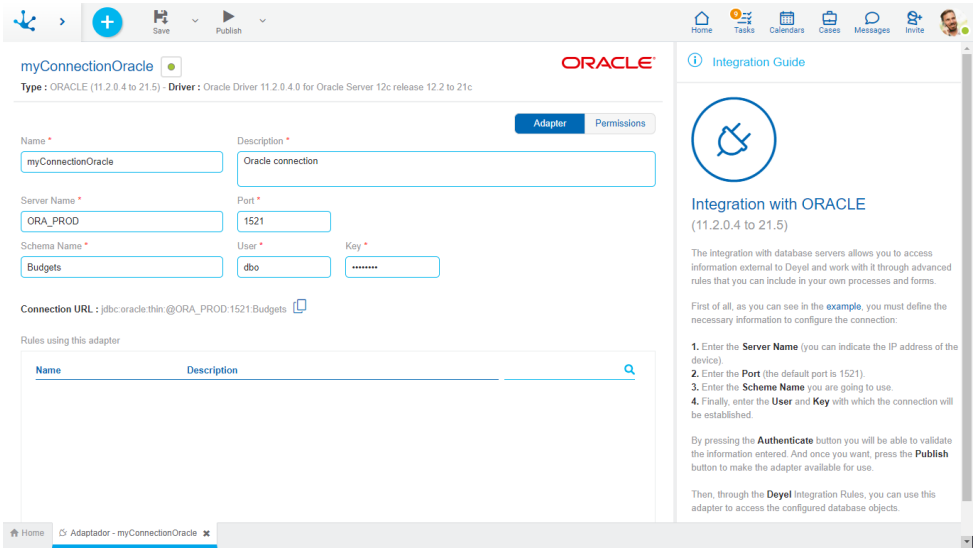
Name	Description	Parameters
		used. • **presencepenalty**: Number between -2.0 and 2.0. Positive values penalize new tokens based on whether they appear in the text so far, increasing the likelihood that the model will talk about new topics. • **frequencypenalty**: Number between -2.0 and 2.0. Positive values penalize new tokens based on their existing frequency in the text so far, reducing the likelihood that the model will repeat the same line word for word. • **user**: Unique identifier representing the end user, which can help OpenAI monitor and detect abuse. Output • **completeResponse**: Full answer by OpenAI. • **listOfMessages**: List of messages generated by OpenAI. • **listOfMessagesWithRoles**: List of messages generated by OpenAI with roles included.
openAIModerationRule	Given an input text, it is checked for inappropriate content according to the OpenAI content policies.	Input • **input**: Input text to classify. Output • **hate**: Represents whether or not the given text has hate content. • **hatethreatening**: Represents whether or not the given text has threatening or hate content. • **selfharm**: Represents whether or not the given text has self-harm content.

Name	Description	Parameters
		• **sexual**: Represents whether or not the given text has sexual content. • **sexualminors**: Represents whether or not the given text has sexual content on minors. • **violence**: Represents whether or not the given text has violence content. • **violencegraphic**: Represents whether or not the given text has violence graphic content.

> *These rules are in "Draft" state and they should be published by a user with administrator permissions, The OpenAI adapter should be previously published.*

3.4.12.2.2.20. Oracle

Apart from properties shared by adapters those specific to Oracle are added.



> *An asterisk "*" on the label indicates that the property is required.*

Server Name

Name or IP address of the server where the database engine is installed.

Port

Port number used by the database engine.

Schema Name
Name of the schema within the database with which you want to integrate.

User
Identification of the database user that has permissions on the schema.

Key
Password of the database user making the connection.

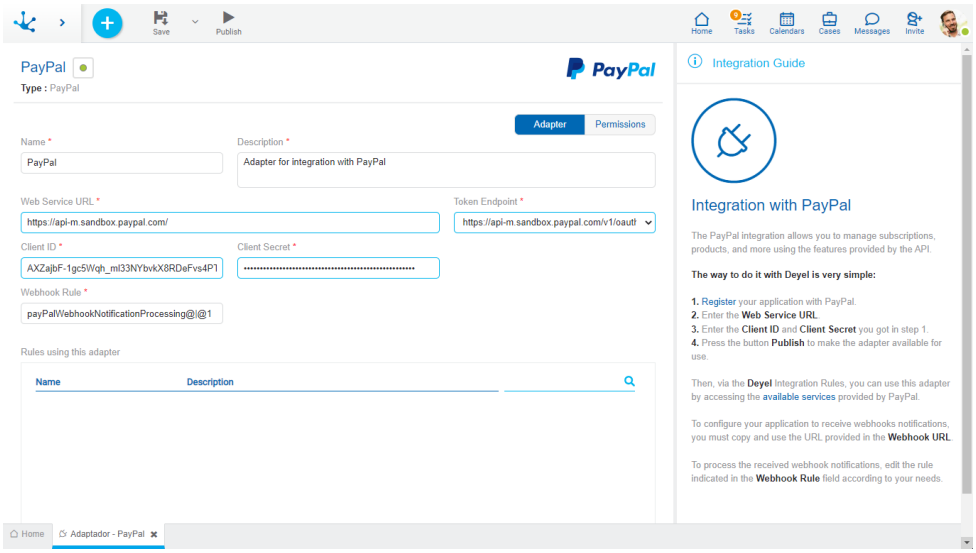
Connection URL
Shows the connection string for the adapter.

Allows to copy the connection URL.

Rules using this adapter
In this grid, the advanced rules used by the adapter are displayed and their definition can be shown from each line.

3.4.12.2.2.21. PayPal

Apart from the properties shared by adapters, those specific to PayPal are added.



> *An asterisk "*" on the label indicates that the property is required.*

Web Service URL
Corresponds to the URL that handles requests for services that are made from the **Deyel** environment to the PayPal platform.

Token Endpoint
Allows to select whether the token is obtained from a production environment or a test environment.

Client ID

Identification of the service that PayPal issues when creating an application on its platform.

Client Secret
Secret key associated to the Client ID property, they are used together when requesting access to the resources provided by the platform.

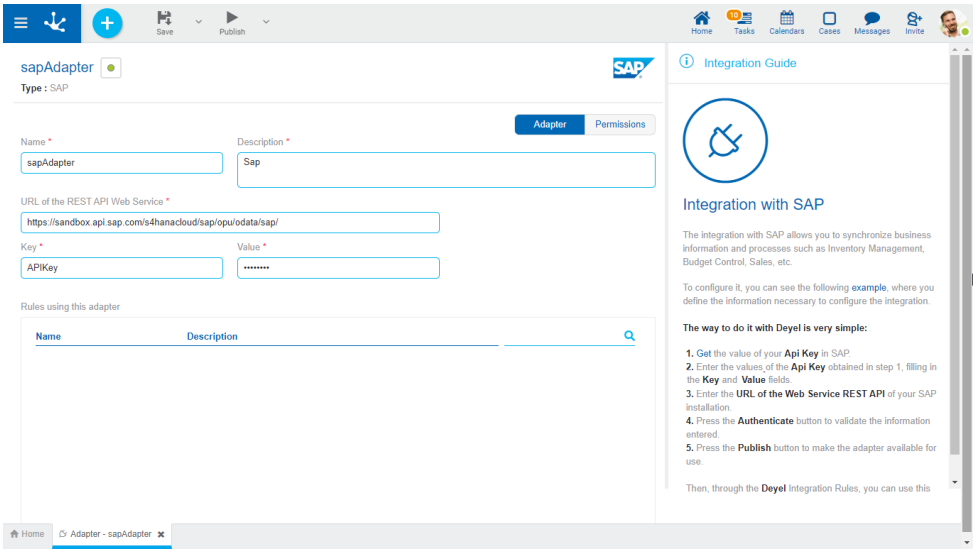
Webhook Rule
Indicates the rule that must be modified or created in case it does not exist, to administrate the necessary webhook notifications. It is a non-editable property.

Webhook URL
Indicates the URL that must be copied and configured in the application registered with PayPal, in order to receive and process webhook notifications. It is a non-editable property.

Rules using this adapter
In this grid, the advanced rules used by the adapter are displayed and their definition can be shown from each line.

3.4.12.2.2.22. SAP

Apart from properties shared by adapters those specific to SAP are added.



> *An asterisk "*" on the label indicates that the property is required.*

URL of the REST API Web Service
Service URL of the SAP environment with which it is integrating.

Key
Identifier that serves as a means of authentication to use of the service, provided by the SAP environment.

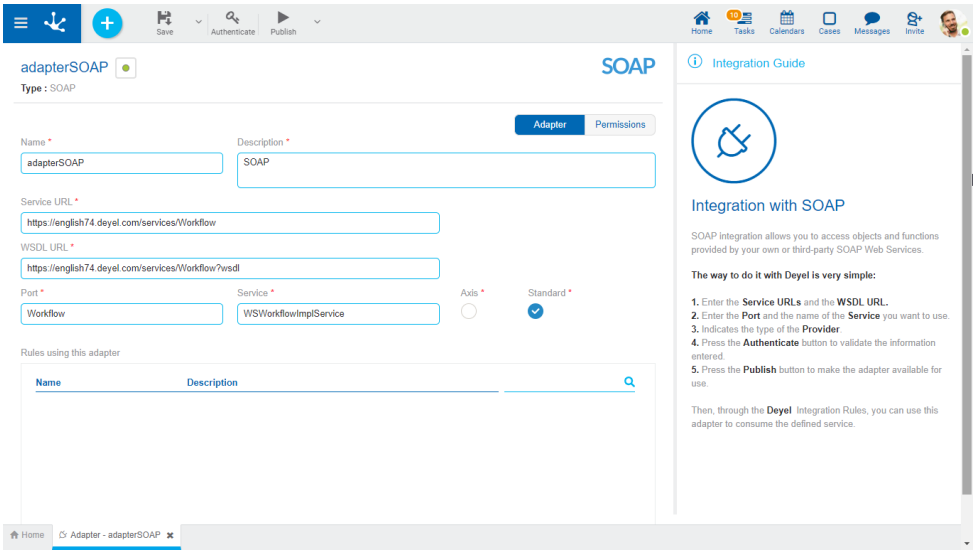
Value
Key associated to the property Key, also provided by the SAP environment.

Rules using this adapter

In this grid, the advanced rules used by the adapter are displayed and their definition can be shown from each line.

3.4.12.2.2.23. Soap

Apart from properties shared by adapters those specific to Soap are added.



> *An asterisk "*" on the label indicates that the property is required.*

Service URL

Address of the SOAP service with which you want to integrate.

WSDL URL

Address where the definition of the SOAP service classes and methods is obtained.

Port

Port number used by the SOAP service.

Service

It is the name of the service class.

Authentication Method

Identifies how **Deyel** will authenticate with the service. Possible values are:

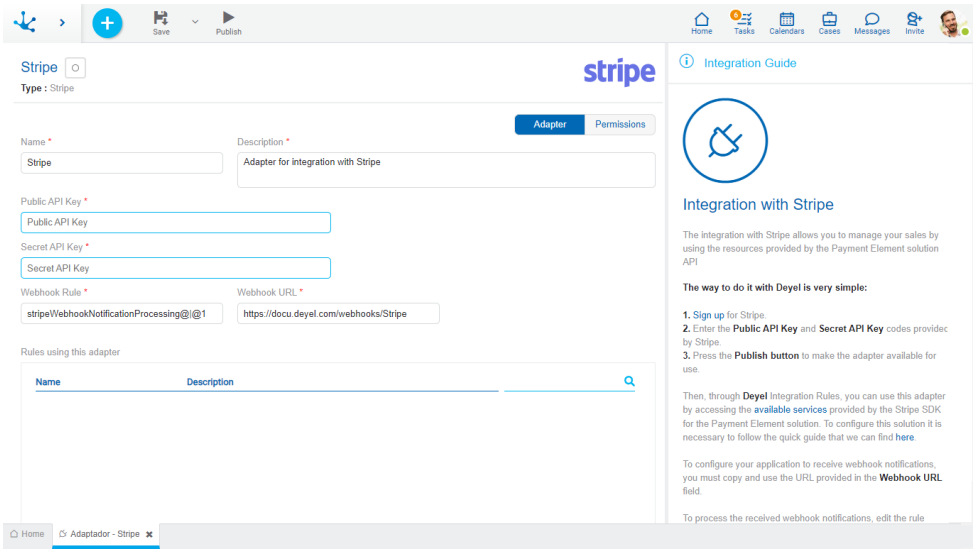
- Axis
- Standard

Rules using this adapter

In this grid, the advanced rules used by the adapter are displayed and their definition can be shown from each line.

3.4.12.2.2.24. Stripe

Apart from the properties shared by adapters, those specific to Stripe are added.



An asterisk "" on the label indicates that the property is required.*

Public API Key
It is the public key given by Stripe to build payment flows using the Stripe.js library.

Secret Api Key
It is the secret key given by Stripe, used to request access to the resources provided by the platform.

Webhook Rule
Indicates the name of the rule that should be used, to administrate the necessary webhook notifications. If the rule does not exist, it must be created. It is a non-editable property.

Webhook URL
Indicates the URL to be copied and configured in the application registered with Stripe, in order to receive and process webhook notifications. It is a non-editable property.

Rules using this adapter
In this grid, the advanced rules used by the adapter are displayed and their definition can be shown from each line.

Available Methods

Advanced rules created on the Stripe adapter have built-in methods.

Name	Description	Result
getPublicAPIKey()	Gets the public key configured on the adapter.	String

Name	Description	Result
createPaymentIntent(pAmount, pCurrency, pRelatedObject)	Used to create a payment attempt and instantiate the Stripe payment form. Parameters • Double **pAmount**: Payment amount. • String **pCurrency**: Payment currency code. According to currency codes available on Stripe. • String **pRelatedObject**: Identifier of the **Deyel** object to associate with the payment. It's optional.	String

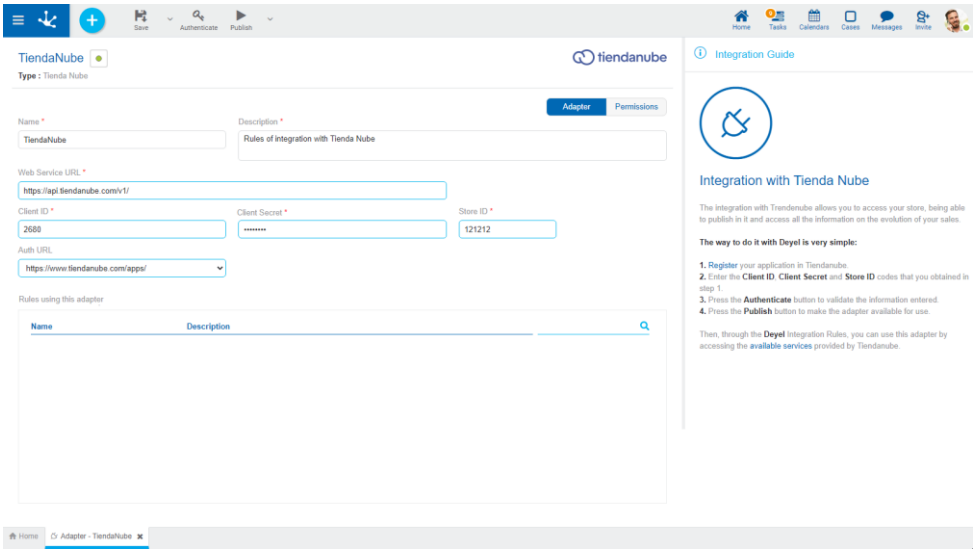
For example, to use Stripe's Payment Element solution, an advanced rule that uses the getPublicAPIKey() method and stores the obtained value in an output parameter of the rule should be created.

Subsequently, the previous advanced rule must be executed and the value of the output parameter must be used as the public key.

For detailed information, refer to the Stripe documentation.

3.4.12.2.2.25. Tiendanube

Apart from properties shared by adapters those specific to Tiendanube are added.



An asterisk "" on the label indicates that the property is required.*

Web Service URL

Corresponds to the URL that handles the return of requests for services that are made from the **Deyel** environment to the Tiendanube platform.

Client ID
Identification of the service that Tiendanube issues when creating an application on its platform.

Client Secret
Secret key associated to the Client ID property, they are used together when requesting access to the resources provided by the platform.

Store ID
It is the unique identification of the store in Tiendanube

Auth URL
It corresponds to the URL of Tiendanube against which the authentication must be performed.

Rules using this adapter
In this grid, the advanced rules used by the adapter are displayed and their definition can be shown from each line.

3.4.12.2.2.26. Twitter

Deyel allows integration with the Twitter platform through this adapter. In it, the connection credentials are registered and the interaction of the network users of **Deyel** with Twitter can be centrally managed.

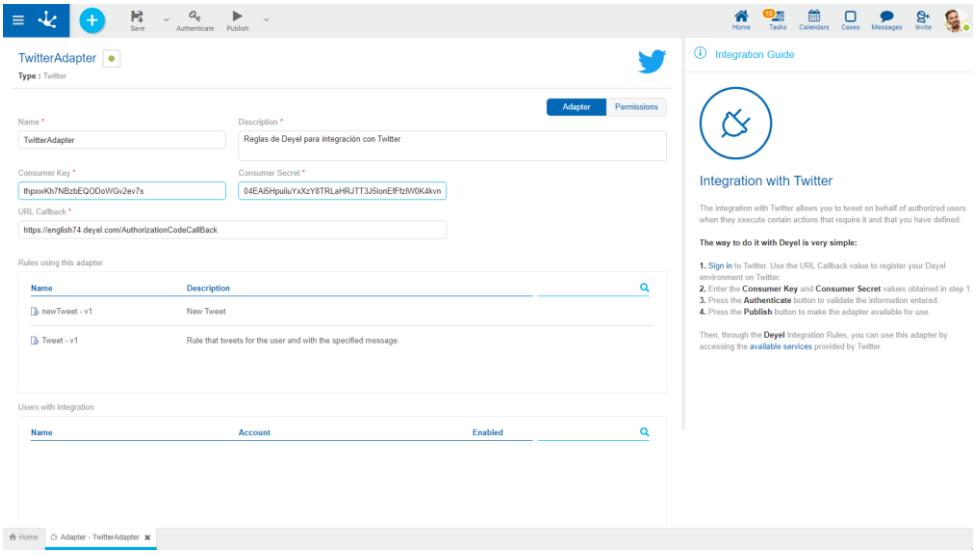
From their profile, each user can enter their credentials to access their Twitter account so that **Deyel** uses them when connecting on its behalf, for example, to publish a post.

Steps for Integration with Twitter

- Register an application on Twitter that identifies the instance of **Deyel**.
- Model the properties of the Twitter adapter detailed below.
- Users must authorize **Deyel** to use the Twitter account from their profile.

Properties

Apart from properties shared by adapters those specific to Twitter are added.



> *An asterisk "*" on the label indicates that the property is required.*

Consumer Key
Identification key of the service that Twitter issues when creating an application on its platform.

Consumer Secret
Secret key associated to the property Consumer Key, are used together when requesting access to the resources provided by the platform.

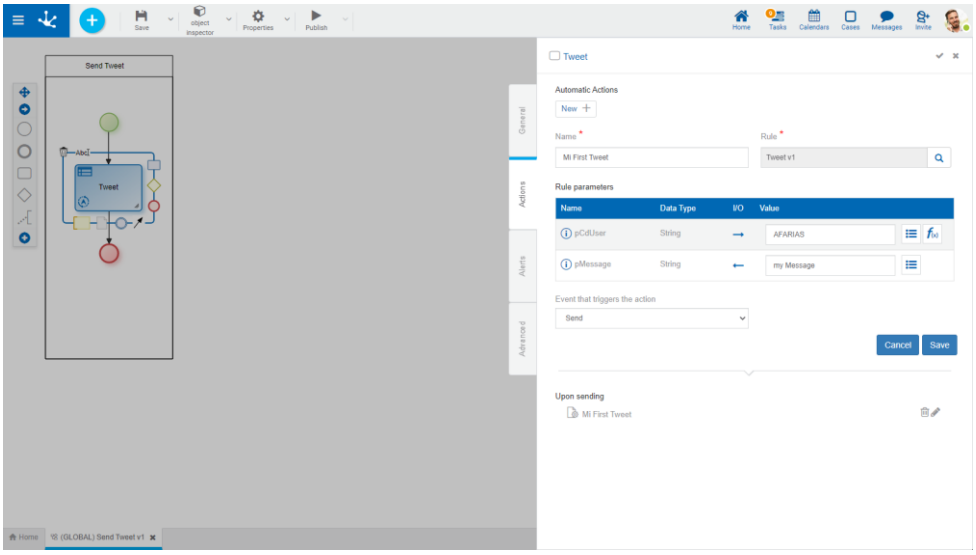
URL Callback
This property must be defined when creating the application on Twitter and corresponds to the URL that handles the return of requests for services that are made from the **Deyel** environment to the Twitter platform.
The value for this property must be composed with the name of the environment Deyel https://.deyel.com/AuthorizationCodeCallBack.

Rules using this adapter
In this grid, the advanced rules used by the adapter are displayed and their definition can be shown from each line.

Users with integration
This grid shows users who have configured the integration with Twitter in their profile and the state of each one, which can be enabled or disabled. The state can be modified from each line.

Example of Use

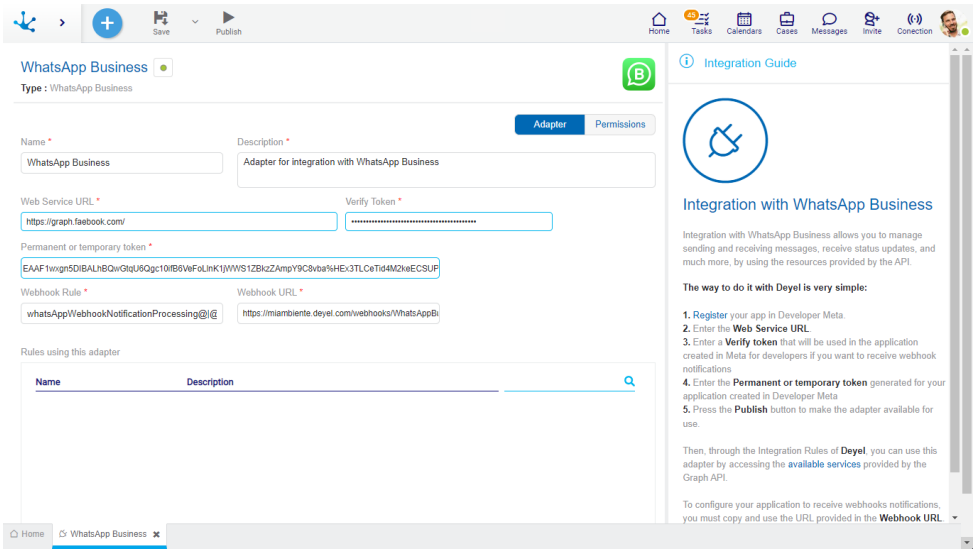
In a content publishing management process, an activity can publish a tweet automatically.



It must be a rule type activity and as an automatic action it invokes a "Tweet v1 - Rule" rule that publishes a tweet to the user with the specified message. The user that publishes the tweet and the message are reported as parameters.

3.4.12.2.2.27. WhatsApp Business

Apart from the properties shared by adapters, those specific to WhatsApp Business are added.



> *An asterisk "*" on the label indicates that the property is required.*

Web Service URL

Corresponds to the URL that handles requests for services that are made from the **Deyel** environment to the WhatsApp Business platform.

Verify Token

Verification token configured in the application developed in WhatsApp Business. It is used to verify that the webhook that is indicated to receive notifications works correctly.

Permanent or Temporary Token

Temporary or permanent token obtained in the application developed in WhatsApp Business. It is used to access the different endpoints offered by the WhatsApp API.

Webhook Rule

Indicates the rule that must be modified or created in case it does not exist, to administrate the necessary webhook notifications. It is a non-editable property.

Webhook URL

Indicates the URL that must be copied and configured in the application registered with WhatsApp Business, in order to receive and process webhook notifications. It is a non-editable property.

Rules using this adapter

In this grid, the advanced rules used by the adapter are displayed and their definition can be shown from each line.

3.4.13. Pages Modeling

 Pages Modeling

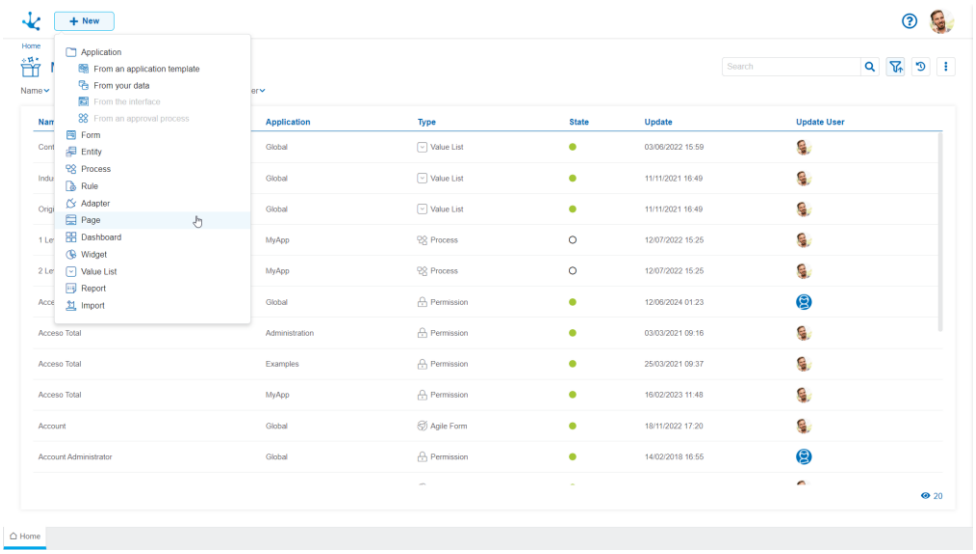
The page modeler allows to create custom interfaces that make up the application being modeled. Pages can show business information and define data sources if necessary.

Pages are fully responsive, automatically adjusting to the screen or device they are being viewed on to deliver the best user experience, every time. The elements can be modeled as fixed (size in px), which means that their size does not change, regardless of the breakpoint, or fluid (size in %) which means that the element automatically changes its size according to the breakpoint and the modeler preferences. These units of measure help create a layout that fits perfectly in any breakpoint.

To have a page that is as adaptable as possible, it is convenient to use relative measurements, allowing inferior elements to change size in relation to the superior elements, ensuring that elements do not overlap and look good on all types of screens, this is achieved by setting the size of elements from highest to lowest hierarchy.

Elements can be dragged anywhere on the page, within sections or containers. They can also be dragged into layout elements.

Although the page modeler is aimed at modeling without writing a single line of code, there is the possibility of extending the operation of the page and its elements through events and Deyel SDK for JS.



New This button is used to create a page from the option Page.

The general characteristics of the page modeler and the main elements that compose it are described in the following topics:

- Modeling Facilities
- Modeling Area
- General Properties
- Style Properties

3.4.13.1. Modeling Facilities

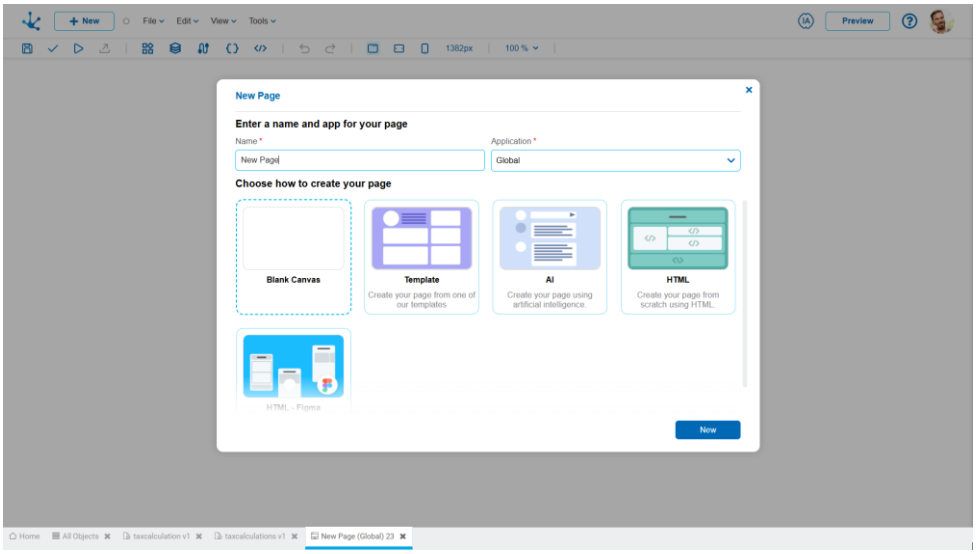
General characteristics of this modeler are specified below.

New Page

The modeler user defines a new page, which after being published is available to be used on any device.

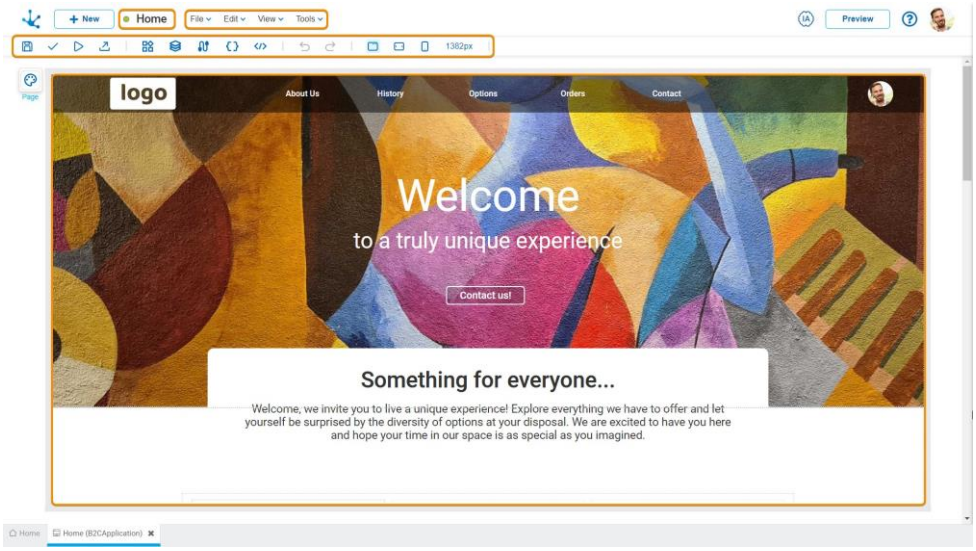
When creating the page, it opens a panel that allows defining some of its general properties and choose from the following options:

- Blank canvas: The modeler starts without selecting a template.
- Template: It allows creating a page from a built-in template, which can be selected from a set of options. Once chosen, the template is displayed in the graphic modeling area of the new page.
- Artificial Intelligence:It allows the creation of a page using artificial intelligence.
- HTML: It allows the creation of a page from an html file.
- HTML - Figma: It allows the creation of a page from an html file generated from a Figma design.



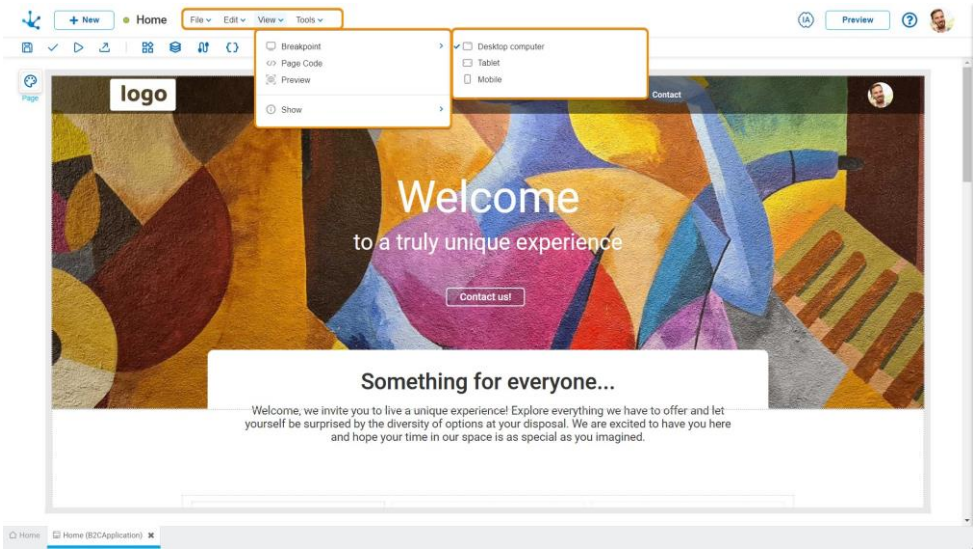
Workspace

- Page Information
  - State
  - Name
- Expanded Menu
- Top Toolbar
- Modeling Area



3.4.13.1.1. Expanded Menu

It is a horizontal list of options containing vertical submenus with different operations on the page or on its modeling. In turn, each option in a submenu may expand a dependent submenu.

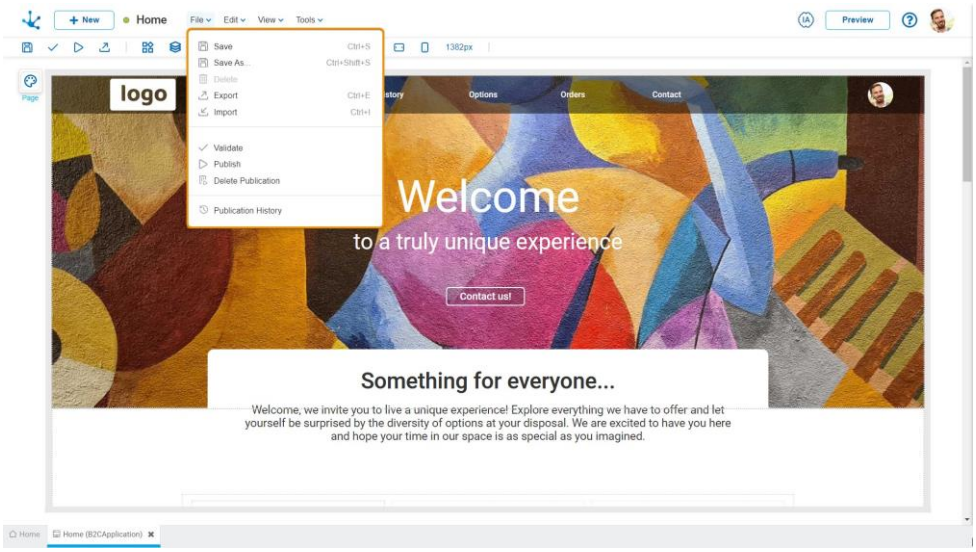


The expanded menu is made up of:

- File Submenu
- Edit Submenu
- View Submenu
- Tools Submenu

3.4.13.1.1.1. File Submenu

This submenu opens by clicking on the “File” option and allows the performance of operations on the page.



Save

This icon allows to save the object in the repository of **Deyel**, leaving its state as "Draft" or "Modified". If certain conditions are met, the modeler user receives a message indicating that the operation was performed correctly, otherwise they receive an explanatory message.

Conditions

- The application of the object must exist.
- The name must be unique in the application.
- The object permissions are required.

Save As

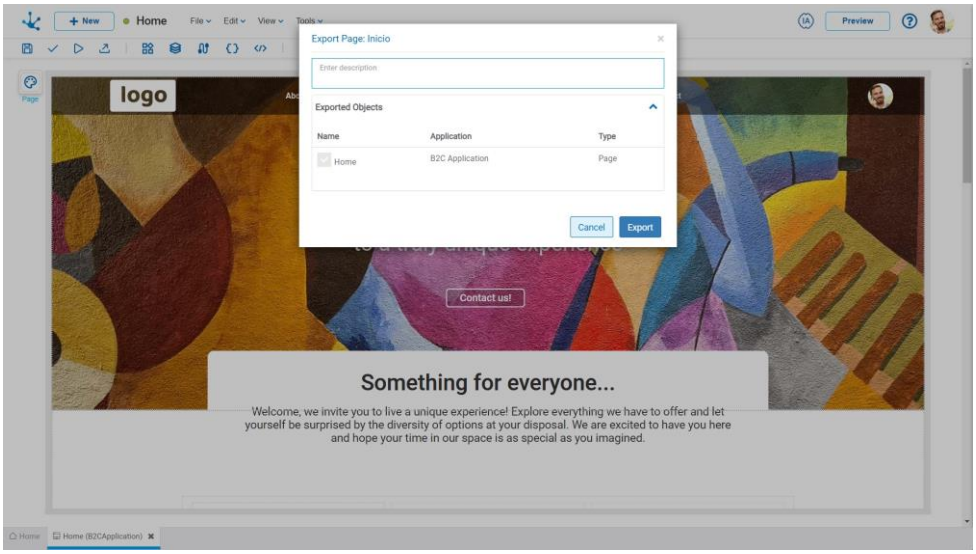
It allows creating a copy of the modeled page by entering a new name and selecting the application to which it belongs. It is also possible to indicate whether the page is reusable or not. The copy opens in a new tab in [state](#) "Draft".

Delete

It allows to delete the object only if it is in "Draft" [state](#) and does not have other associated objects of **Deyel** that were previously saved or published. When deleted, the tab it is on closes and the object is removed from the modeler's grid.

Export

This icon opens a window for the user to select and confirm the export of the object.



Properties

Description

In this property a text explaining the reason for the operation can be entered.
This text can be modified upon import and is displayed in the description column of the export record.

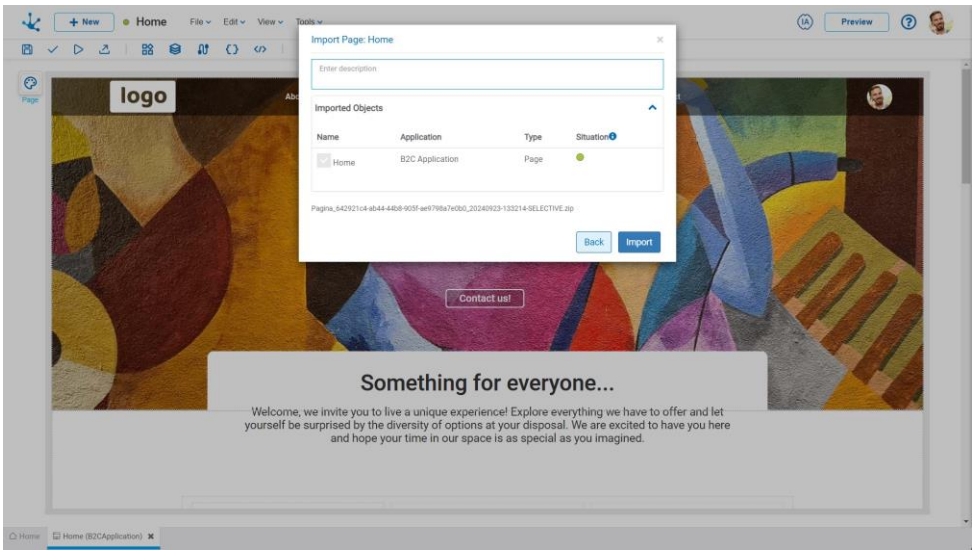
Exported Objects

By expanding the container, the object being exported is shown.

Press the "Cancel" button to undo export or press the "Export" button to finish.

Import

It allows to open a window for the user to select and confirm the import of the object.



Validate

This icon allows to validate if the object is ready to be published, that is, the same validations are carried out as when publishing and the result of them is reported.

Publish

Through this icon the object changes to “Published” state and the modeler user receives the corresponding message, indicating the result of the operation. The condition for publishing is the same as for the "Save" operation.

Condition

- The entities and related rules used in a data source must be published.

Delete Publication

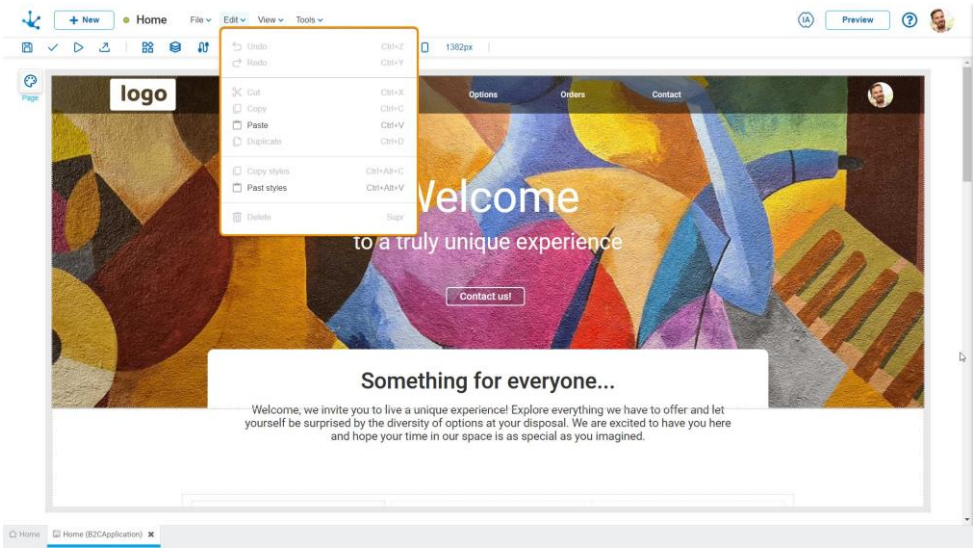
Allows to leave the object unavailable for use by returning it to the "Draft" state.

Publication History

This icon allows managing versions of published objects, displaying their history, to have better control of the changes that each object has.

3.4.13.1.1.2. Edit Submenu

This submenu is opened by clicking on the “Edit” option and allows to perform operations within the modeler.



Undo (Ctrl+Z)

Allows to easily reverse the changes made in the modeler.

Redo (Ctrl+Y)

Allows to easily reapply the changes made in the modeler.

Cut (Ctrl+X)

Allows to send the selected element to the clipboard while deleting it from the graphic modeling area.

Copy (Ctrl+C)

It allows to copy any application element and temporarily place it on the clipboard.

Paste (Ctrl+V)

It allows to take the item from the clipboard and place it elsewhere in the graphic modeling area.

Duplicate (Ctrl+D)

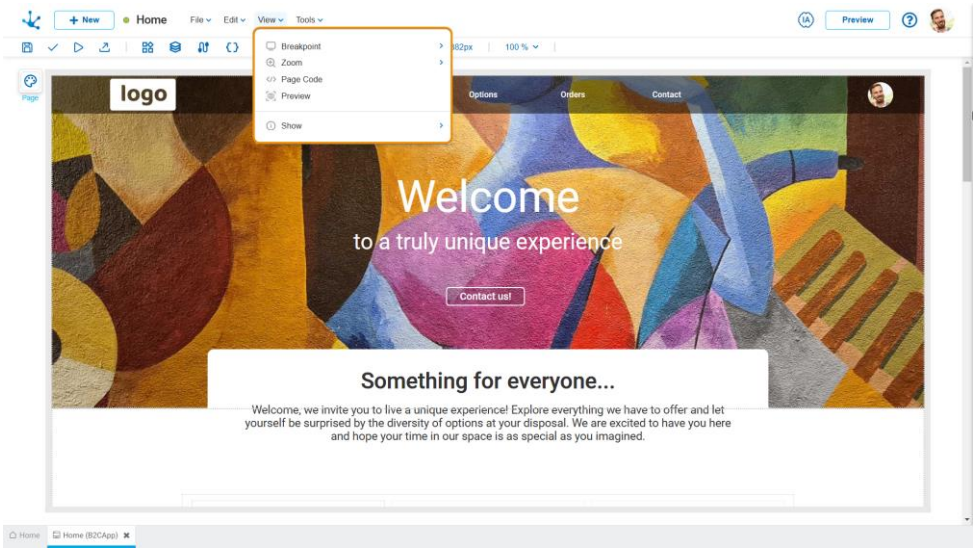
It allows to make a copy of any application element within the graphic modeling area.

Delete (Del)

It allows to delete the selected element from the graphic modeling area.

3.4.13.1.1.3. View Submenu

This submenu opens by clicking on the "View" option and allows to select different ways of displaying the page.



Breakpoint

They ensure that users always view the best version of their page, regardless of the device they are using.

The modeler includes the most common breakpoints (desktop, tablet, and mobile), allowing the page to adapt to individual screen sizes by defining what takes priority in the layout. Styles can be rearranged, elements can be shown or hidden, and the layout can be customized for each breakpoint.

By clicking on “Breakpoint”, a secondary submenu is displayed, where the following options can be selected:

- Desktop Computer
- Tablet
- Mobile

Zoom

Selecting "Zoom" allows adjusting the viewing size of the modeling area, using built-in options such as 25%, 50%, 75%, 90%, 100%, 125%, 150% and 200%. The shortcut Ctrl+ to zoom in and Ctrl- to zoom out can also be used. Another option to adjust the size is to hold down the Ctrl key and use the mouse wheel.

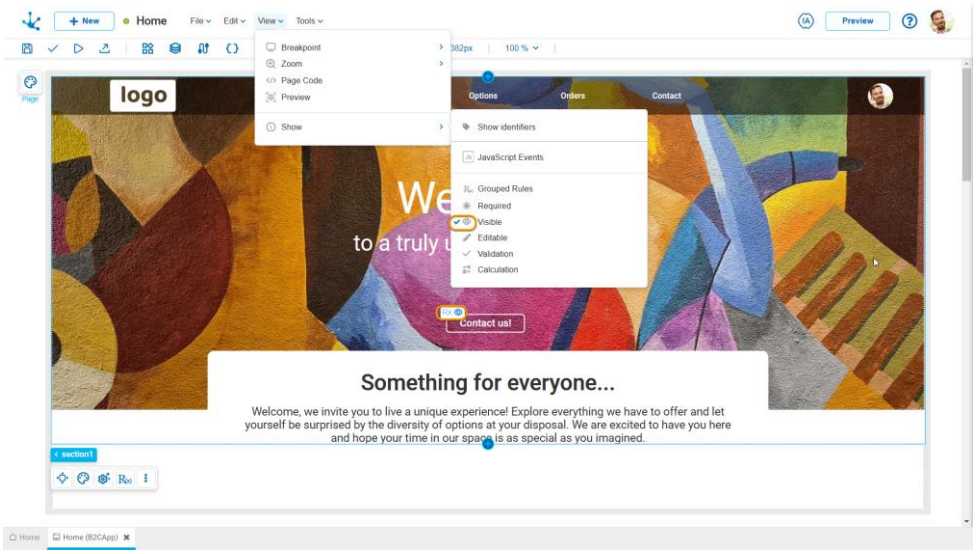
Page Code

Selecting this option opens or closes an area for code editing at the bottom margin of the modeling area.

Show

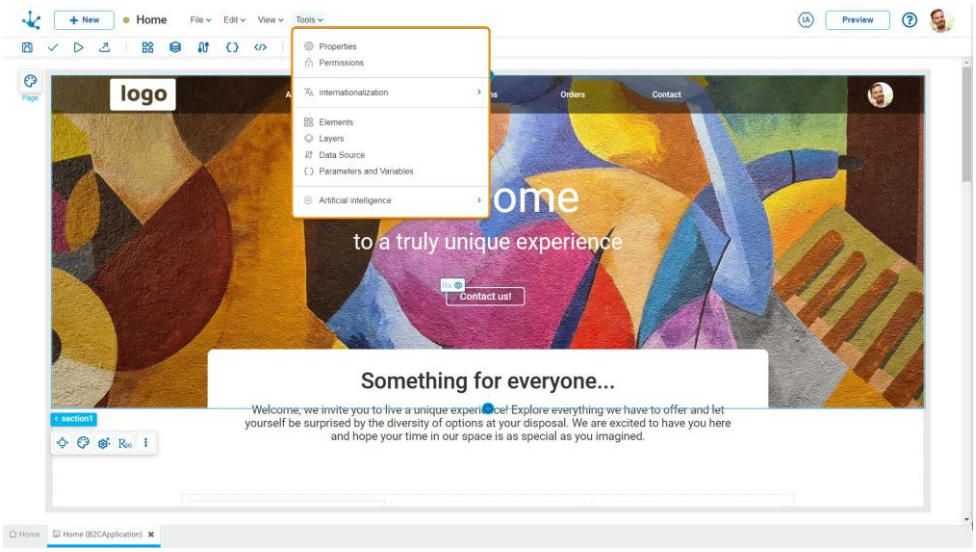
This functionality allows showing or hiding the set of visual icons that indicate the behavior that affects the elements.
Clicking on the "Show" option displays a secondary submenu, where the icons corresponding to the modeled rules that should be displayed for each element can be selected.

- JavaScript Events
- Grouped Rules
- Required
- Visible
- Editable
- Validation
- Calculation



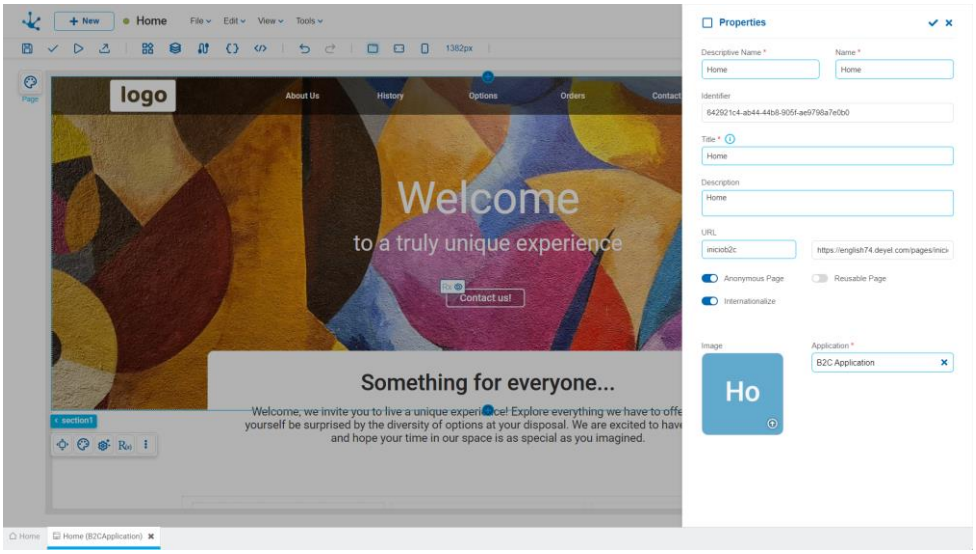
3.4.13.1.1.4. Tools Submenu

This submenu is opened by clicking on the “Tools” option and it allows modeling page properties and permissions, as well as different modeling functionalities.



Properties

This option allows opening the general properties panel of the page in order to model its features.



Permissions

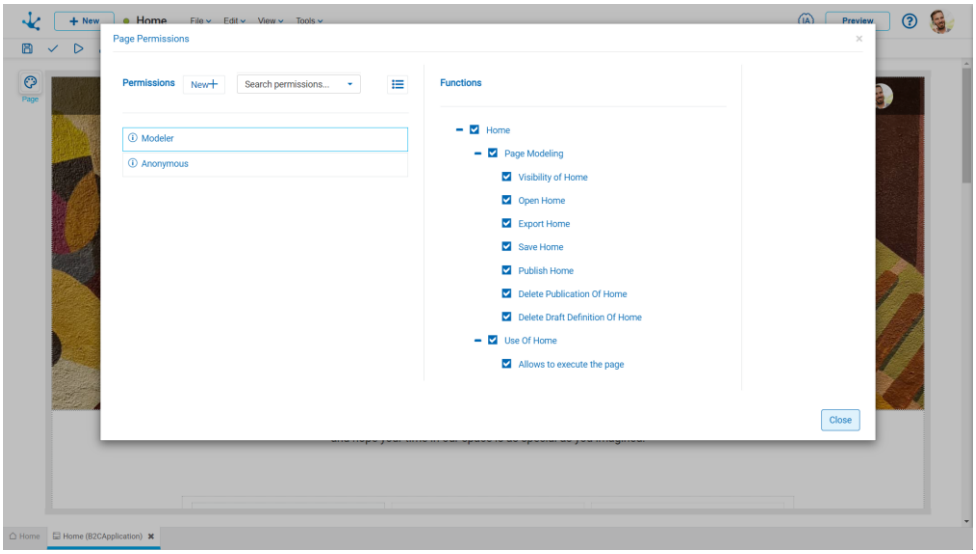
Allows to assign the security functions for use and modeling of the object to the existing permissions or to new permissions that the user creates, without having to go to the permission settings option from the menu.

Sections

- Permissions: Permissions to which object functions are assigned.
- Security Functions: Represents the total set of security functions, modeling and use of the object. Those that are marked are the ones included in the selected permission.

By default, all security functions for a new object are assigned to the permission Modeler.

Users who are assigned the permissions have access to the functions included in it.



New Opens a panel to create a new permission and once created, the security functions included in it must be selected .

Search permissions... Allows to select a permission from a list and enables the input of characters to filter the values in the list.

Opens the wizard to select a permission and once chosen, the necessary object security functions must be checked.

To unrelate a permission from the object, hover the cursor over the permission and press the icon . If there are functions selected for that permission, they must be unchecked in order to delete it.

Modeling Security Functions

- Visibility: Allows to display the object in the Deyel modeler grid.
- Open: Allows to open the object from the Deyel modeler.
- Export: Enables the operation to export the object from the tools submenu of the expanded menu.
- Save: Enables the operation of saving modifications made to the object.
- Publish: Enables the operation of publishing the object leaving its state as "Published".
- Delete publication: Enables the operation of deleting the object publication leaving its state as "Draft".
- Delete draft definition: Enables the operation of deleting the object.

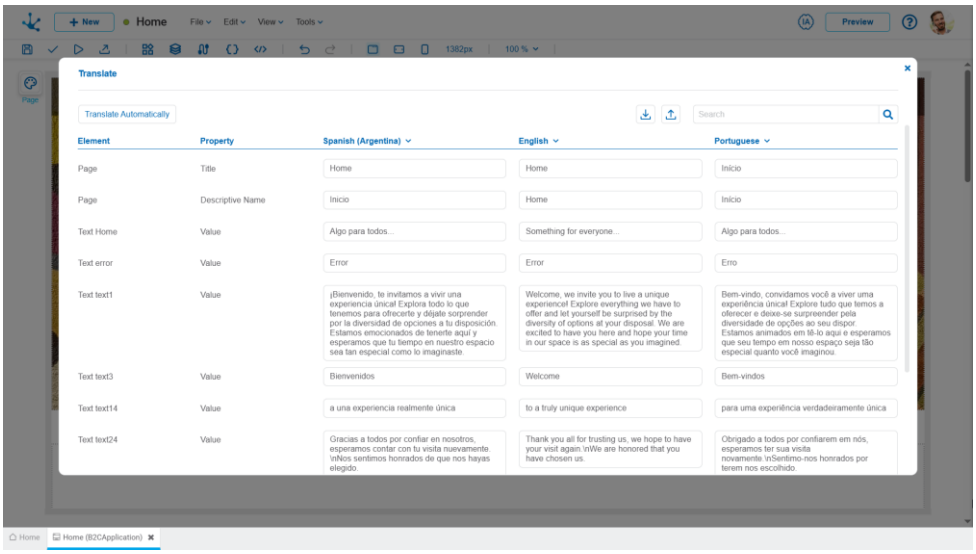
Use Security Function

- It allows to execute the page.

> *When a page is configured as anonymous, the use function is automatically assigned to the "Anonymous" permission.*

Internationalization

The "Internationalization" option is enabled when the property Internationalize is activated. To access the translation panel, it is necessary to select "Internationalization" and then choose "Translate".



In this panel, all the translatable properties of the object are displayed, along with their content in the original language, and translations in the other available languages can be entered manually.

If a value is modified in the original language column, the change is applied directly to the object's property.

In the header of each language column, a menu that allows selecting another language can be displayed, updating the column with its contents.

Options Available in the Top Margin

Translate Automatically Generates translations from the original language to other languages using artificial intelligence.

Downloads an Excel file with the contents of the translation panel.

Allows selecting an Excel file to import translations, making it easier to manage large volumes of content.

Filters the contents of columns based on the characters entered in the search field.

Elements

This option allows opening the elements panel from where it is possible to select those that will be incorporated into the page.

Layers

This option allows opening the layers panel of the page where the page elements organized hierarchically can be displayed.

Data Source

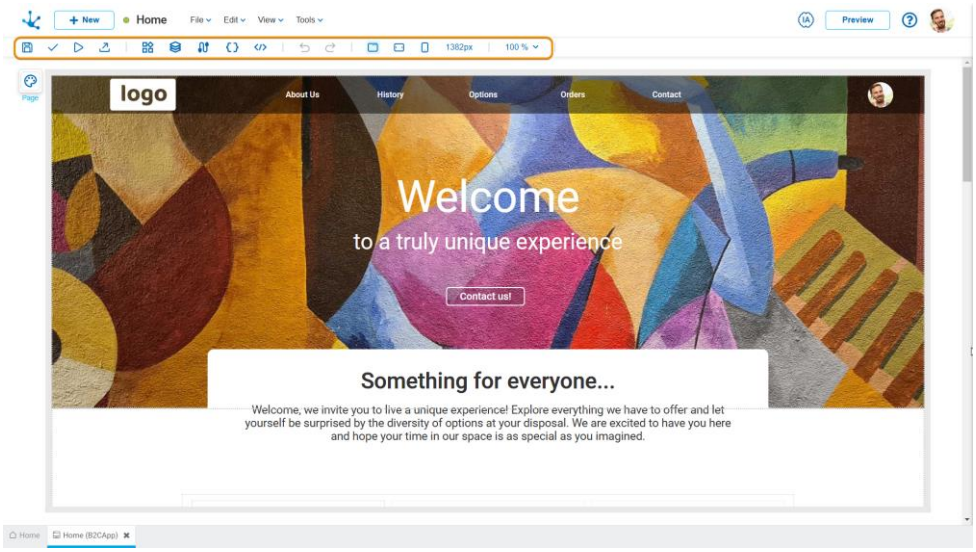
This option allows opening the data source panel from where ,those that will be used on the page can be defined.

Parameters and Variables

This option allows opening the parameters and variables panel from where those that will be used on the page can be defined.

3.4.13.1.2. Top Toolbar

Contains icons for quick access to the most used operations of the expanded menu.



File

- Save
- Validate
- Publish
- Export

Tools

Elements
Layers
Data Source
Parameters and Variables
Page Code

Edit

Undo
Redo

View

100 % Zoom Percentage

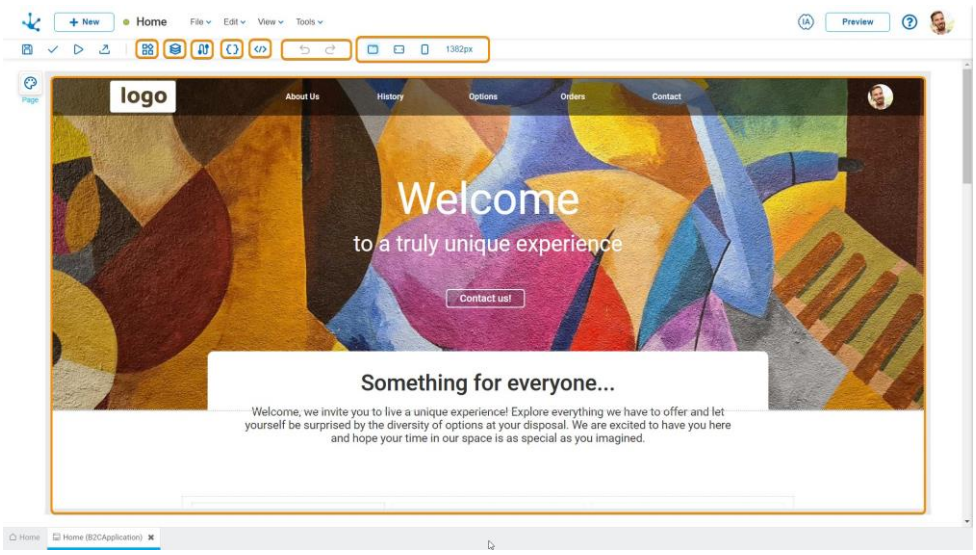
Breakpoints

Desktop Computer
Tablet
Mobile

px Pixels

3.4.13.1.3. Modeling Area

It is the space where pages can be modeled using the "drag and drop" of different elements.



The modeling area of a page is divided into sections within which elements can be added.

Preview

To the right of the expanded menu is the Preview button.

Properties

The general properties and the style properties of the page can be opened in the right side panel of the modeling area.

Events

Pages allow the use of different events.

Event	Description
onInit()	It is executed before loading the page.
afterViewInit()	It is executed after rendering the page.
beforeViewUnload()	It is executed before closing the page.

3.4.13.1.3.1. Elements

A page contains elements of different types, each of these types can be modeled with specific properties.

Each element can contain other elements, in these cases they are called superior elements while the elements they contain are called inferior elements.

Superior Elements

- Containers
- Sections
- Layout
- Repeater

Operations

Add Elements on the Page

Elements can be added to a page via “drag-and-drop” to any location on the page, in isolation, or embedded in superior elements.

When adding an element, it automatically docks to the top or bottom border of the closest highest ranking element.

When added, the elements width is defined by default as a percentage, with the exception of the button element, which has an automatic width.

Modify Position of an Element

Once the element is added, its position can be changed by moving it within the page or by changing its structure and position properties with respect to the superior element.

If an element changes its position, this applies to all breakpoints.

Modify Element Size

The dimensions of an element can be modified by dragging the handles on its outline or by modifying its structure properties.

Guides

There are different types of guides that are displayed when selecting an element and pressing the “Alt” key.

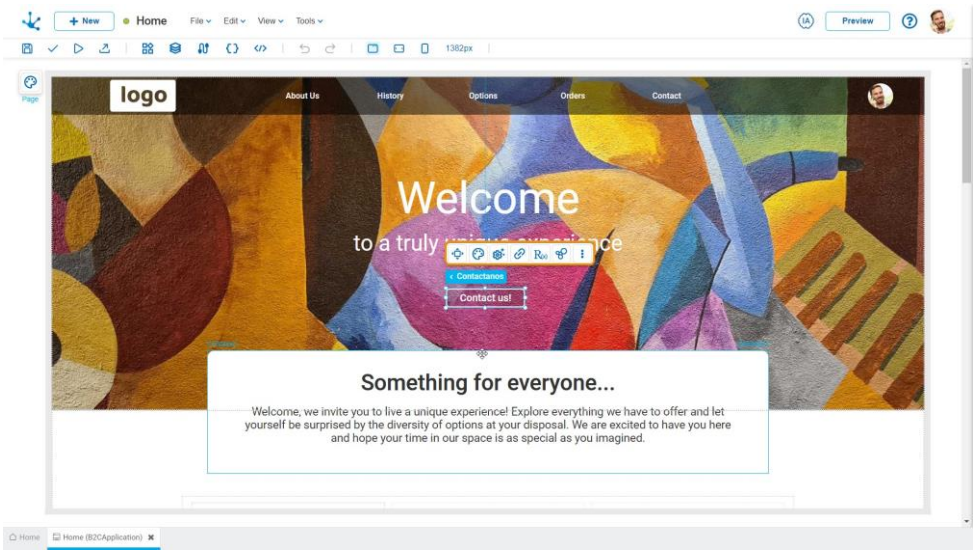
- Docking guides: they are displayed when moving an element and the position where it is moved depends on how close it is to its highest hierarchy element. In addition, the distance to the margins of the upper element is indicated.
- Alignment guides: displayed to help items align.
- Guides of space between elements to know the space between them.

Properties

Each element has specific properties that are displayed using a context menu, which is enabled when selecting the element. Properties are grouped according to their functionality and are detailed for each element type.

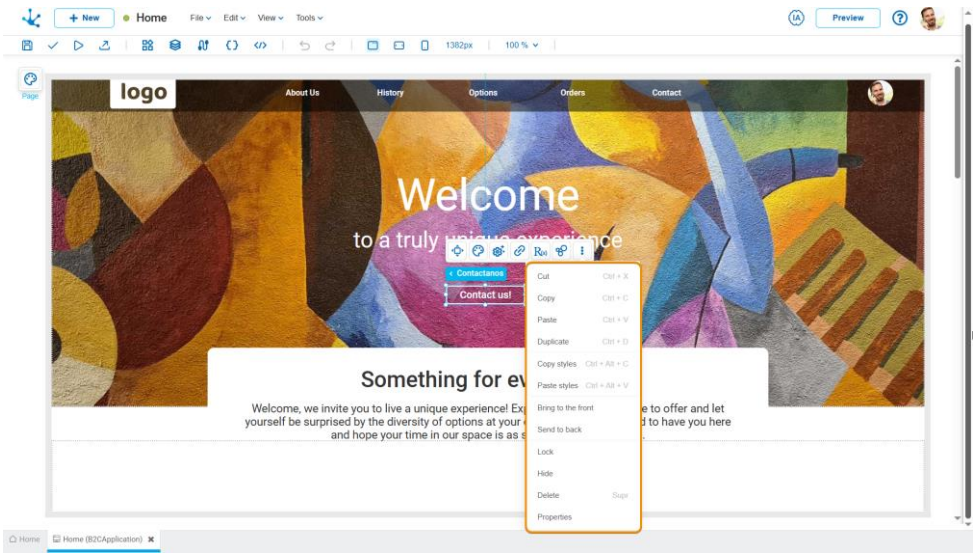
Context Menu

When selecting an element, a palette of icons opens, where each icon represents a group of specific properties of the selected element. The name of the element is also displayed and when the mouse is slid over it, the path from the page to the element can be seen, indicating the superior elements where the content is found.



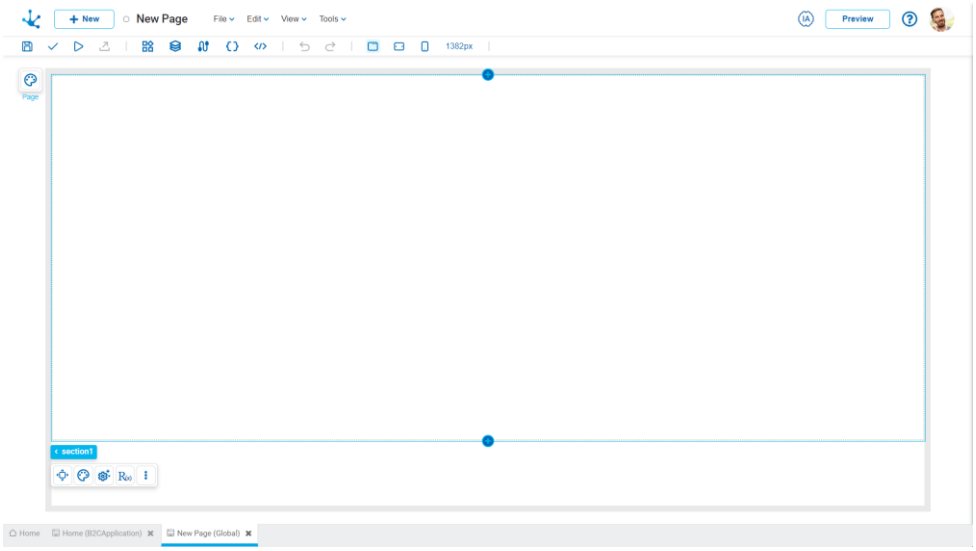
Clicking the right button of the mouse on an element expands a second context menu whose options correspond to operations that can be performed on the selected element. The same menu is displayed by pressing the icon on the palette.

- Cut (Ctrl+X)
- Copy (Ctrl+C)
- Paste (Ctrl+V)
- Duplicate (Ctrl+D)
- Copy styles (Ctrl+Alt+C)
- Paste styles (Ctrl+Alt+V)
- Bring to the front
- Send to back
- Lock/Unlock When an element is locked, it cannot be moved until it is unlocked.
- Hide: Hides the element in the breakpoint that is being used and in minor breakpoints. Once hidden it can be displayed again from Layers.
- Delete (Supr)
- Properties



Section

Pages are divided into sections where elements can be included. By default, the page has a defined section.



Operations

Add

It allows adding a new section by clicking on the icon of an existing section.

Modify Size

The section height can be adjusted from its bottom border, by clicking on it and dragging it. Another way to modify its size is from the structure panel of the section.

Move

A section can be moved using the functionality of layers.

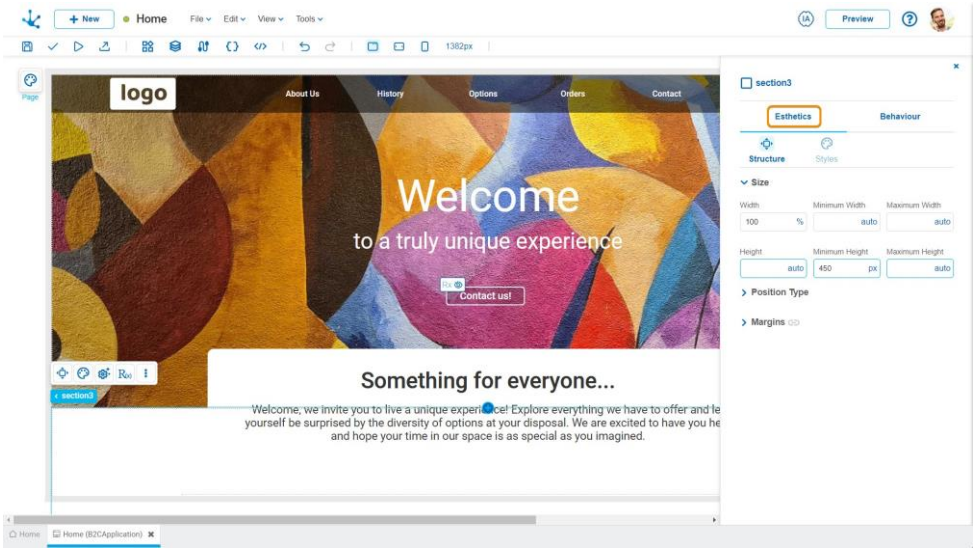
Classification of Properties

Properties are divided into two groups: esthetic properties and behavior properties.

Esthetic Properties

In the esthetic properties panel, the following are grouped:

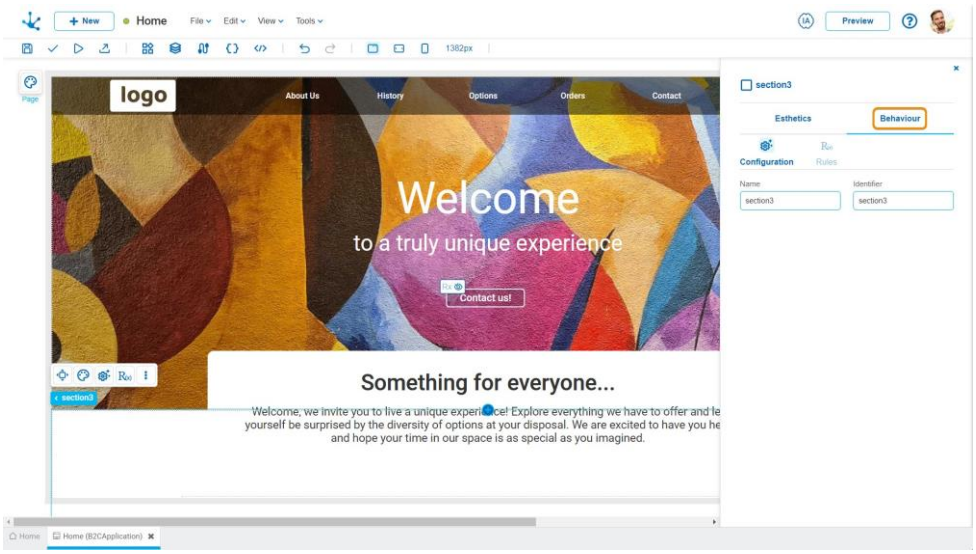
Structure Properties
Style Properties



Behavior Properties

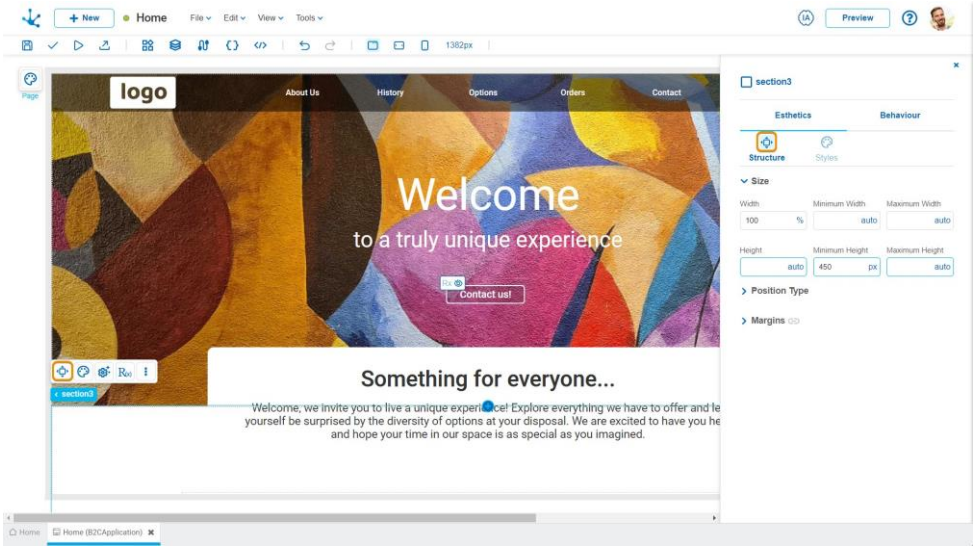
In the behavior properties panel, the following are grouped:

- Configuration Properties
- Rule Properties

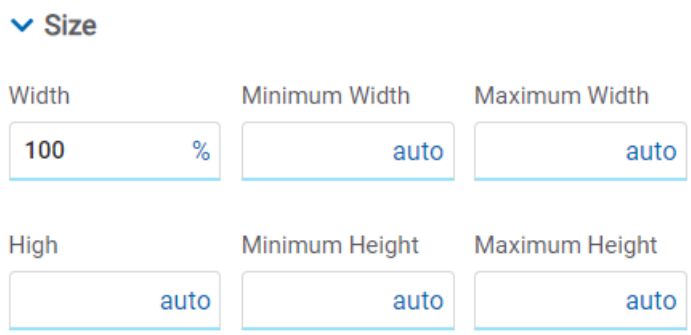


Structure Properties

The structure properties panel of a section opens when selecting the icon  of its context menu.



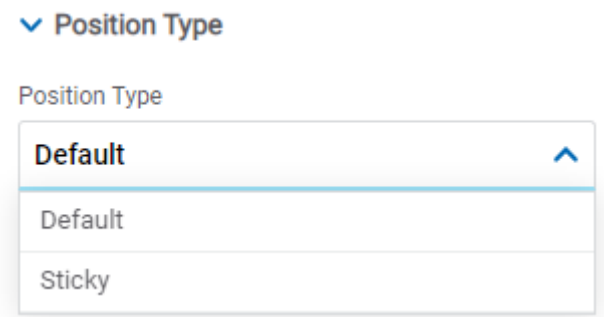
Size



All size properties can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh). Additionally, for Width and Height properties, the “auto” option is added.

> *Sections do not allow modifying the Width property.*

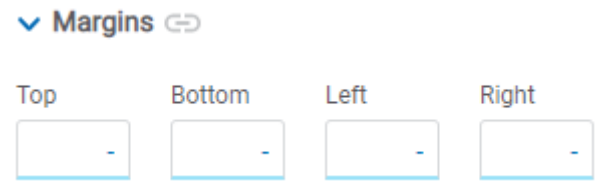
Position Type



Possible Values

- Default: The element has a relative position to the superior element where it was placed (container or section).
- Sticky: It is available only in sections, it allows the section to remain visible while users scroll through it. At first, the section is seen where it has been positioned in the entity, but it "sticks" to the screen as it is scrolled down.

Margins



It allows to define the distance of elements from the borders of their top element. The behavior of margins depends on the docking of the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Top
Distance to the top border of the highest ranking element.

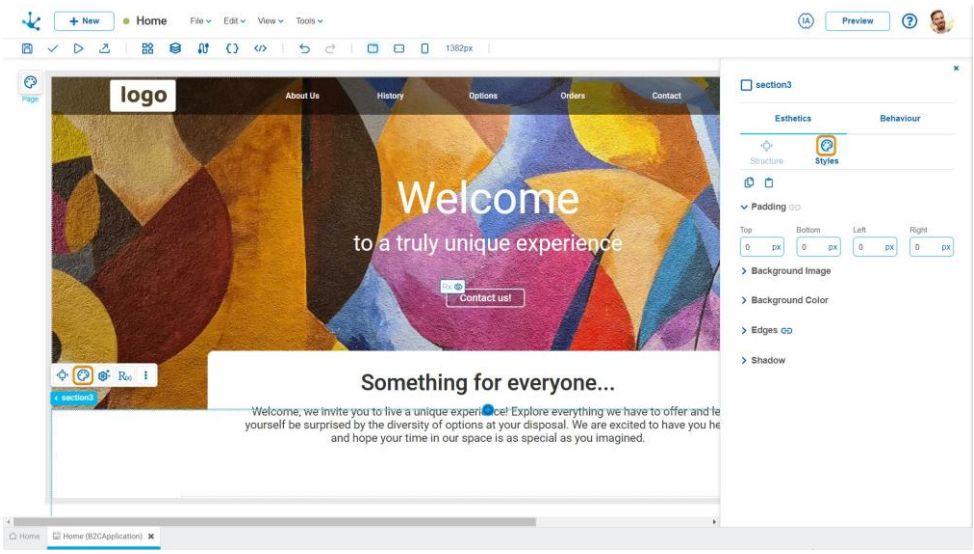
Bottom
Distance to the bottom border of the highest ranking element.

Left
Distance to the left border of the highest ranking element.

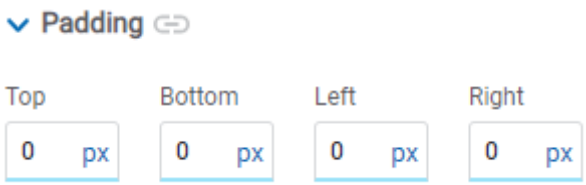
Right
Distance to the right border of the highest ranking element.

Style Properties

The style properties panel of a section opens when selecting the icon of its context menu.



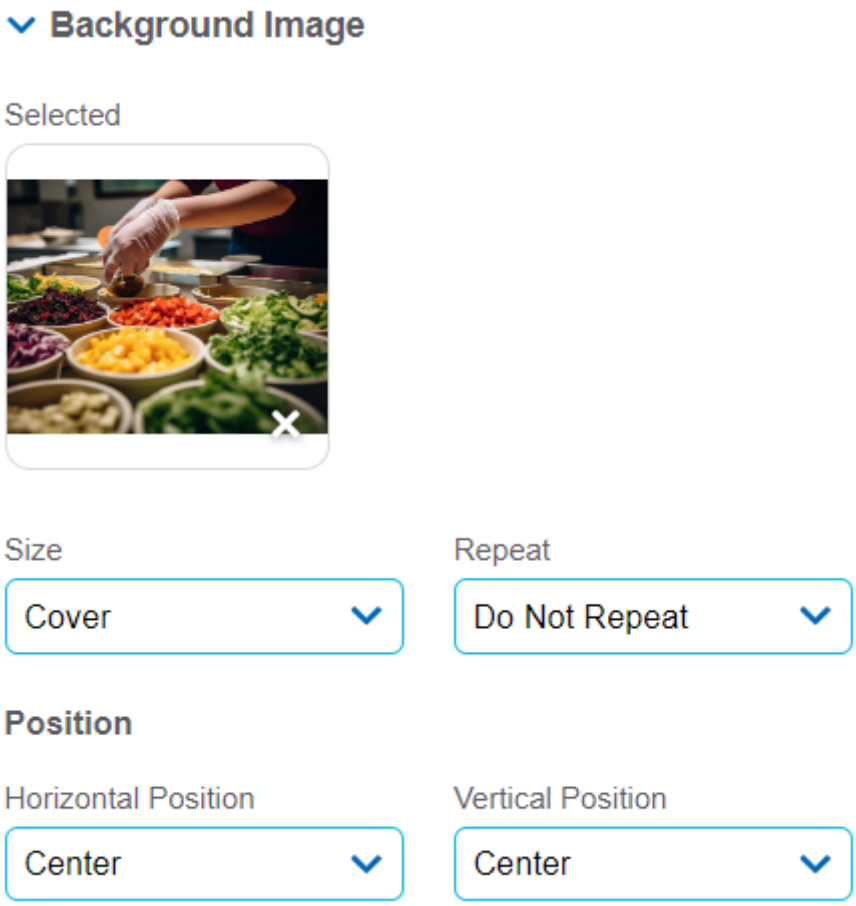
Padding



All padding space properties allow to create a space around the element borders (top, bottom, sides) and its content. The padding space configured is inward, while the margin is configured relative to a top element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Background Image

The element properties are represented by icons on its context menu, where its operations are also available.
PropsEstiloImagenFondo3



It allows to add a background image to the section.

Size

Possible Values

- Cover
- Content
- Auto

Repeat

Possible Values

- Repeat
- Repeat Hor.
- Repeat Vert.
- Spacing
- Rounded
- Do Not Repeat

Horizontal and Vertical Position

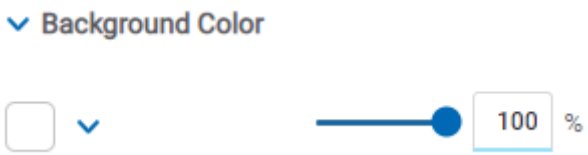
Possible Values

- Center
- Left
- Right

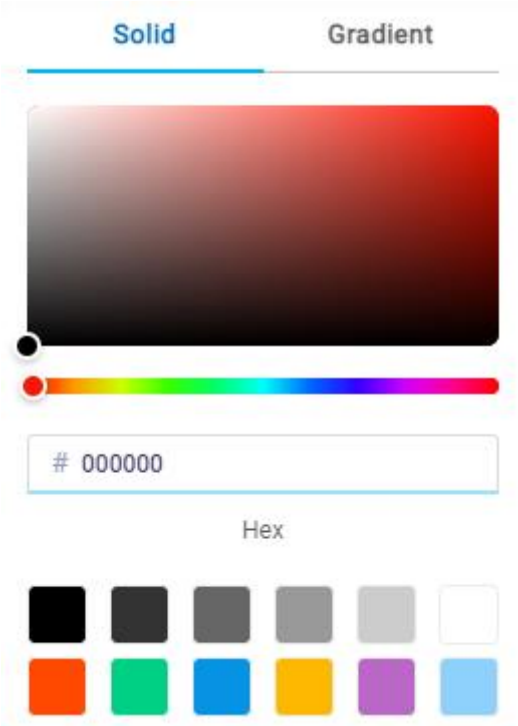
Background Color

It allows to model a background color for the element, it can be solid or gradient and different properties are defined for each one.

Solid



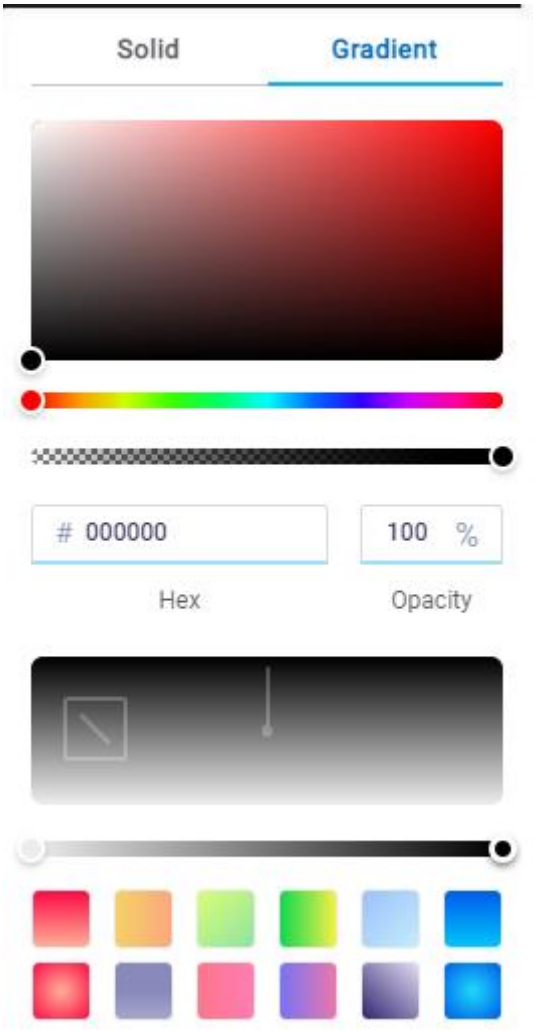
This option enables not only to select the color from a palette but also to define the degree of transparency.



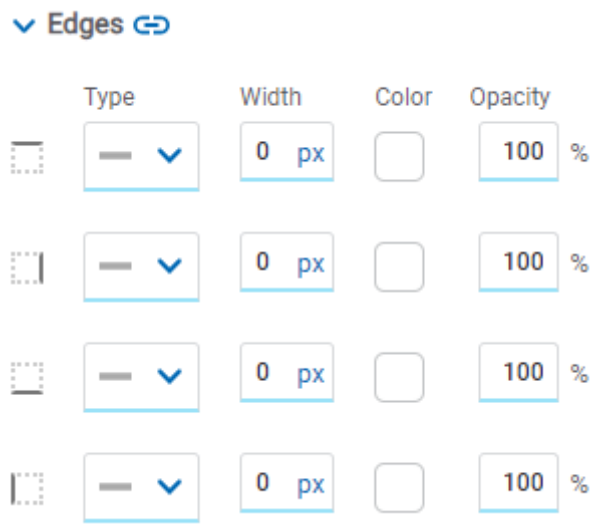
Gradient



This option enables to select not only the color from a palette but also to define its opacity and intensity according to the selected angle.

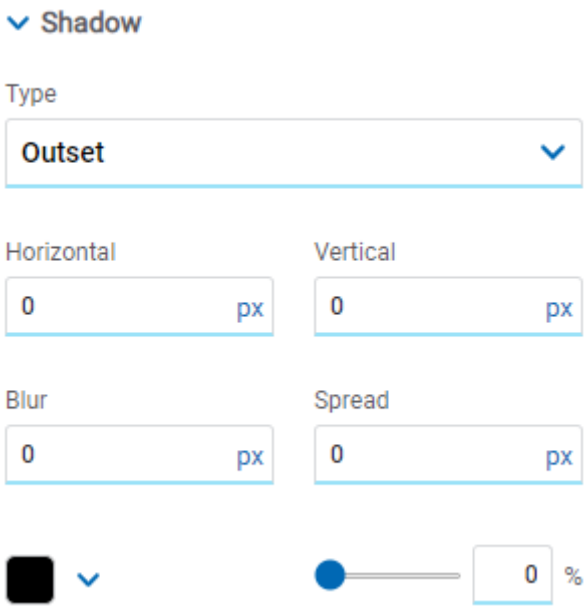


Edges



It allows to define the style of borders. Each one has its type, width, color and opacity defined, the latter as a percentage.

Shadow



Allows to define a shadow effect around the element.

Horizontal
Horizontal size of the shadow to the right of the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Vertical
Vertical size of the shadow below the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

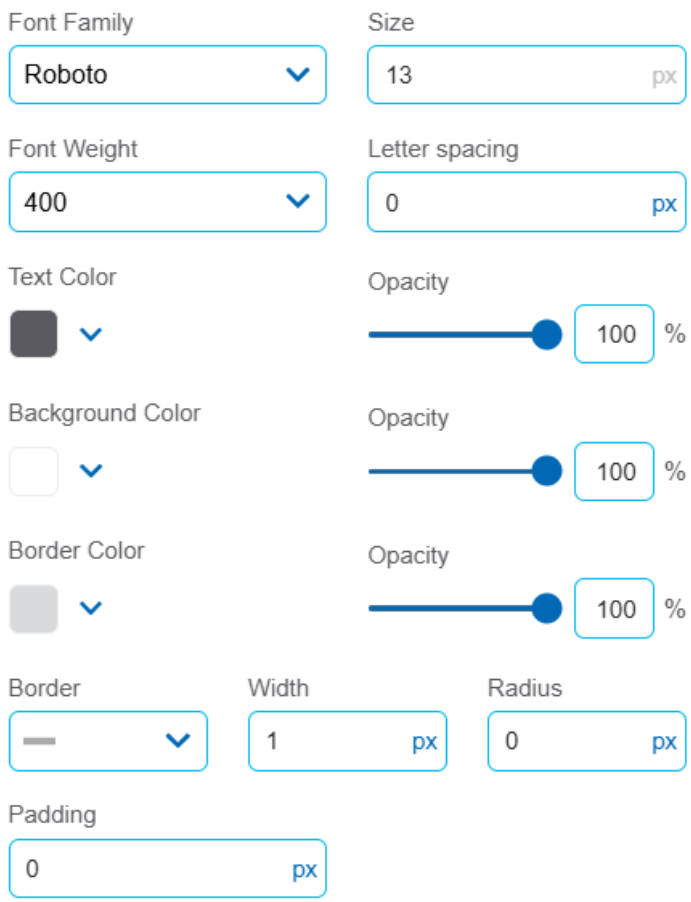
Blur
The greater this value is, the greater and lighter the shadow becomes. If not specified, its value is 0 and the shadow border is darker. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Spread
Positive values will cause the shadow to expand and grow bigger, while negative values will cause the shadow to shrink. If not specified, its value is 0 and the shadow will have the same size as the element.

Color
Defines the color of the shadow.

Transparency
Defines the transparency of the shadow.

Tooltip



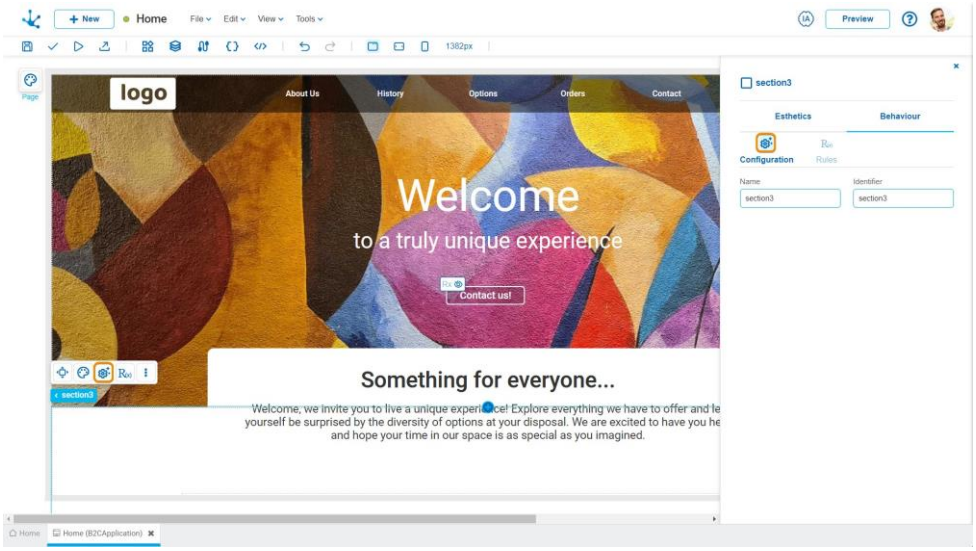
Source
It allows customizing the font, size, thickness and letter spacing of the pop-up message.

Colors
The color and opacity of the text, background and border can be defined. Borders are set uniformly for all sides of the tooltip.

Border
The border type, width, and border radius can be configured.

Configuration Properties

The configuration properties panel of a section opens when selecting the icon of its context menu.



Name
Name used to reference the section during modeling.

Identifier
Uniquely identifies the section. It is used in the Javascript SDK.

Tooltip
Pop-up message that appears when hovering over an element, providing additional information.

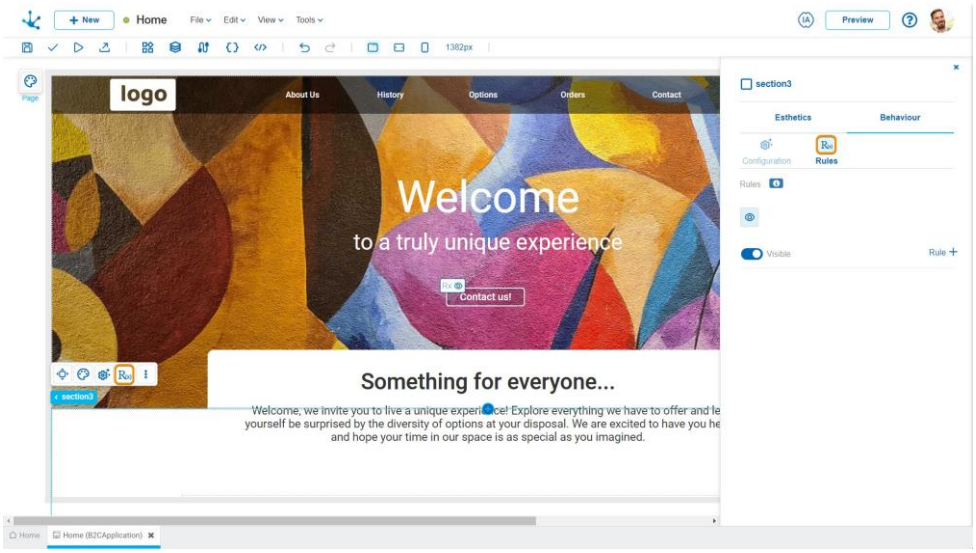


Text
Content of the message being displayed.

Direction
Position of the message in relation to the element (top, bottom, left, or right).

Rule Properties

The embedded rules properties panel of a section opens when selecting the icon of its context menu.



Properties

Rules

It is possible to define embedded rules for behavior, validation, and calculation associated with a section using the wizard (ctrl + space).

Displays syntax examples to write rules.

Visible

Indicates whether the section is visible. If this property is not checked, the section is not displayed on the page.

Visible (default) Not visible

Rule + It opens an editing area to define a rule and determine the visibility condition. If a rule is defined, the icon is displayed with a light blue border.

Operations when defining the rule:

Saves the new or modified rule

Cancels the operation

Operations once the rule is defined:

Edits the existing rule

Deletes the rule

Events

Sections allow the use of different events.

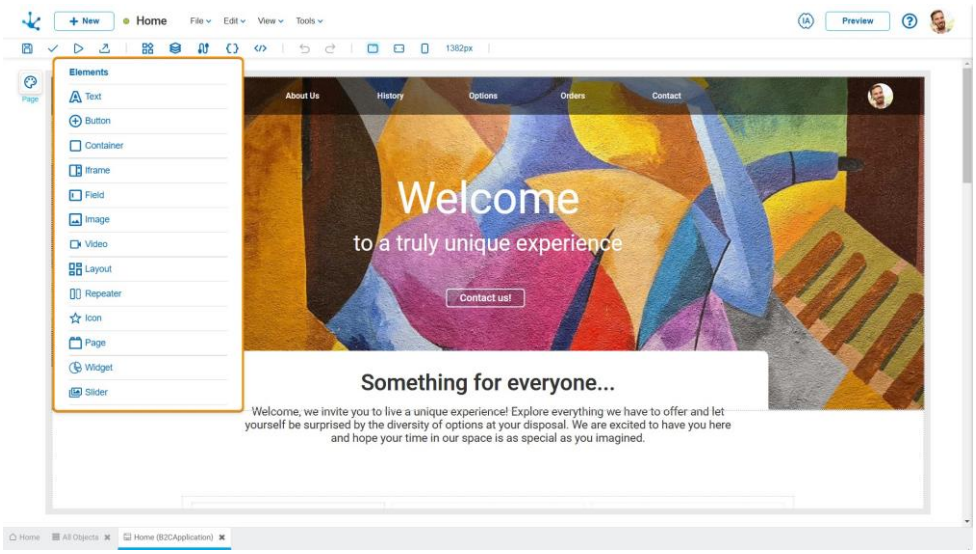
Event	Description
onMouseIn()	It is executed when the cursor is positioned on the element.
onMouseOut()	It is executed when the cursor moves out of the element.
onClick()	It is executed when clicking on the element.
onDoubleClick()	It is executed when clicking twice on the element.
onInit()	It is executed before the element is loaded.
afterViewInit()	It is executed after viewing the element.
onViewportEnter()	It is executed when the element is visible.
onViewportLeave()	It is executed when the element is no longer visible.

Element Types

The elements define the page content and can be of different types.

- Text
- Button
- Container
- Iframe
- Field
- Image
- Video
- Layout
- Repeater
- Icon
- Page
- Widget
- Slider

For each of these element types their properties can be configured in structure, style, configuration, hyperlink and embedded rules panels.



Text

Texts can be included on a page, defining the characteristics by modeling its properties.

The element properties are represented by icons on its context menu, where its operations are also available.

Subtypes

By selecting the "Text" option from the icon on the top toolbar, a list with the different subtypes of this element is displayed, which can be dragged to the modeling area. Each subtype has modeled element properties, predefined in a specific way.

- Title 1
- Title 2
- Subtitle 1
- Subtitle 2
- Paragraph
- Normal Text
- Rich Text

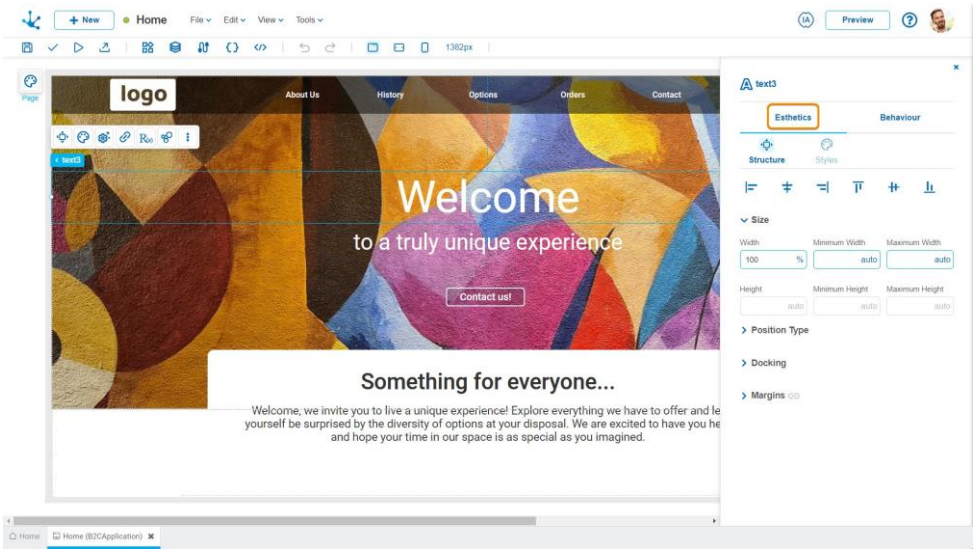
Classification of Properties

Properties are divided into two groups: esthetic properties and behavior properties.

Esthetic Properties

In the esthetic properties panel, the following are grouped:

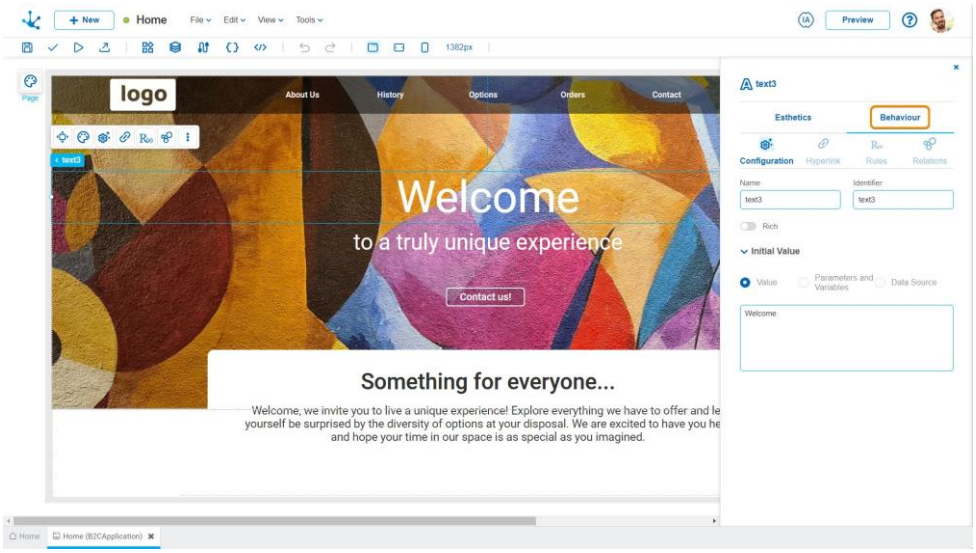
- Structure Properties
- Style Properties



Behavior Properties

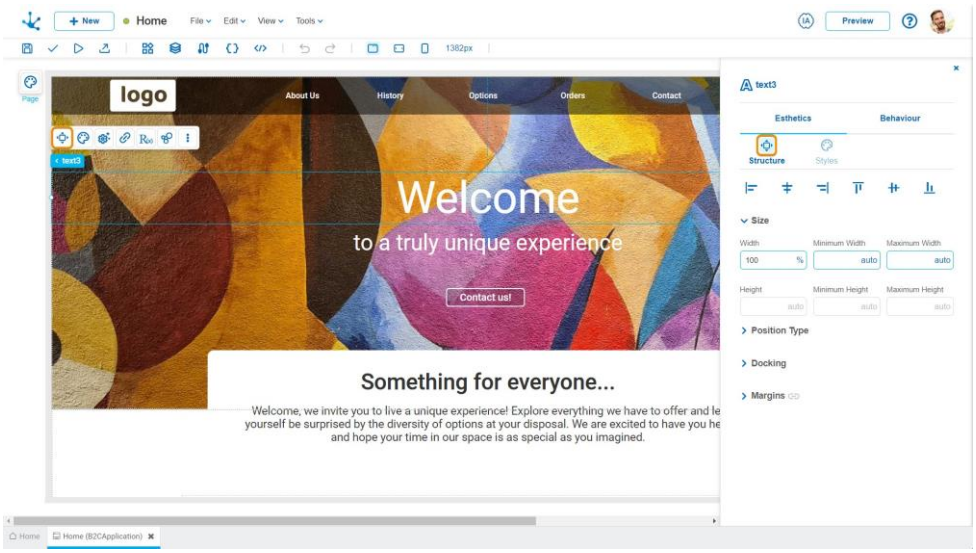
In the behavior properties panel, the following are grouped:

- Configuration Properties
- Hyperlink Properties
- Rules Properties
- Relations Properties



Structure Properties

The structure properties panel of an element opens when selecting the icon of its context menu.

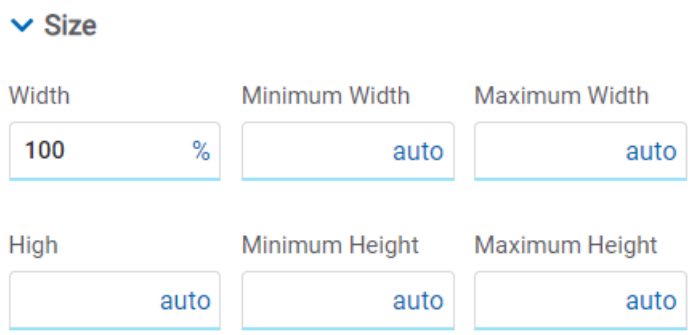


Alignment

Allows to align the element by selecting one of the available icons.

- Align to the left.
- Align to horizontal center.
- Align to the right.
- Align up.
- Align vertical center.
- Align bottom.

Size

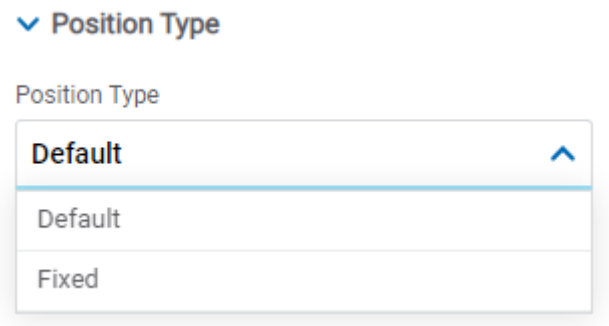


All size properties can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh). Additionally, for Width and Height properties, the “auto” option is added.

> *If selecting percentage (%) for the width and height properties, the size is calculated relative to the top element.*

Position Type

It determines if at the time of execution the element remains fixed on the page or if it moves as the scroll bar moves forward or backward.



Possible Values

- Default: The element has a relative position with respect to the superior element where it was placed (container or section).
- Fixed: Fixes the element to the page so that it is always visible in the same position. It allows to increase the visibility of important content.

Docking



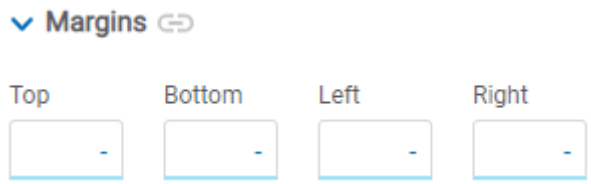
Determines the vertical and horizontal position of elements within the page section or container, when the screen is resized.

When an element is added or moved within the section or container, the modeler automatically docks it to the nearest corners or edges. It can also be configured by clicking on the arrows of the side to be docked to or on the center.

Once the element is docked, its exact position is indicated in the Margins property. The direction of the docking arrows determine the values that are automatically displayed for margins.

The docking position is indicated in the modeling area by dotted lines on the element.

Margins



It allows to define the distance of elements from the borders of their top element. The behavior of margins depends on the docking of the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Top
Distance to the top border of the highest ranking element.

Bottom
Distance to the bottom border of the highest ranking element.

Left
Distance to the left border of the highest ranking element.

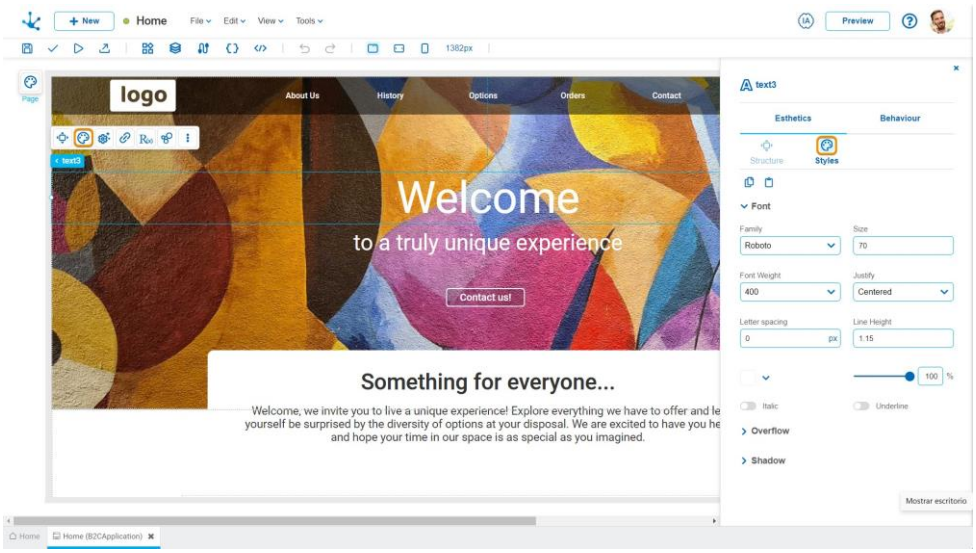
Right
Distance to the right border of the highest ranking element.

Allows the value entered in one of the margins to be copied to the other ones automatically.

Allows to indicate different values for each margin.

Style Properties

The style properties panel of an element opens when selecting the icon of its context menu.

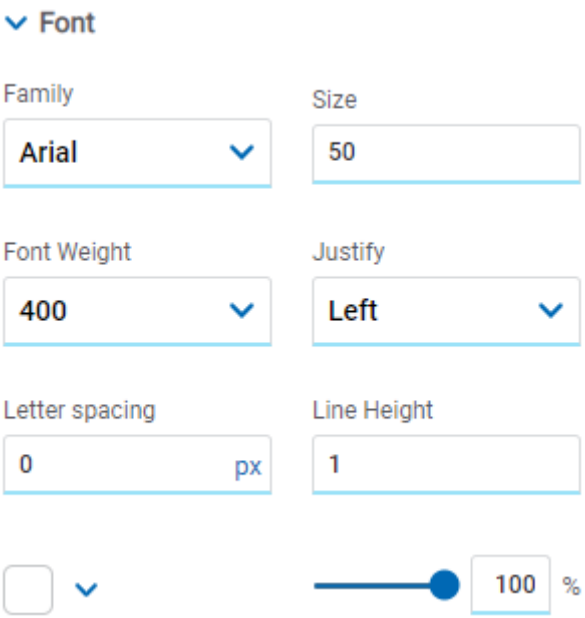


Styles

It allows copying the entire current configuration of the style panel for the selected element (Ctrl+Alt+C).

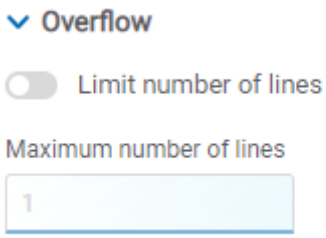
It allows applying the previously copied configuration to another element of the same type (Ctrl+Alt+V).

Source



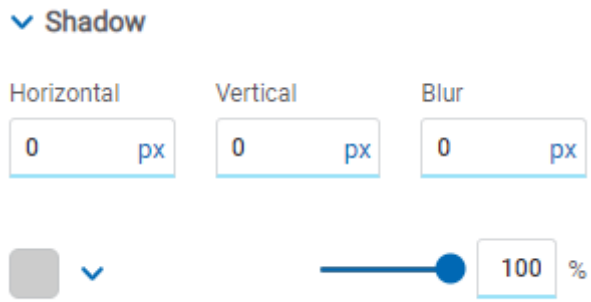
It allows for defining the text style.

Overflow



Allows limiting the number of lines that contain the text, indicating the maximum number of lines. If the text exceeds the number of lines indicated in this property, then it is truncated and displays the icon ⋯.

Shadow



Allows defining a shadow effect around the element.

Horizontal
Horizontal size of the shadow to the right of the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

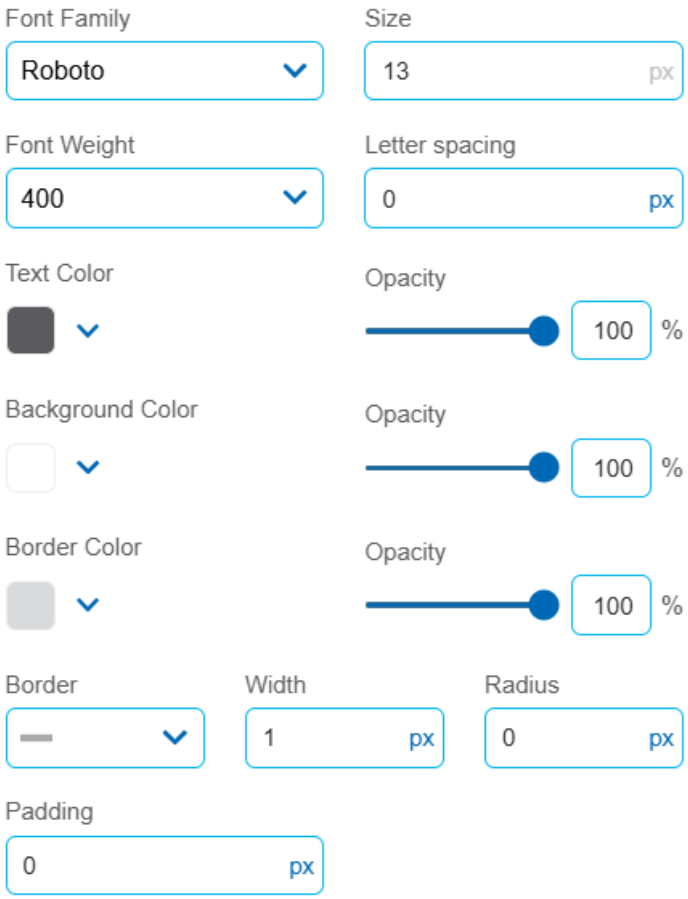
Vertical
Vertical size of the shadow below the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Blur
The greater this value is, the greater and lighter the shadow becomes. If not specified, its value is 0 and the shadow border is darker. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Color
Defines the color of the shadow.

Transparency
Defines the transparency of the shadow.

Tooltip



Source

It allows customizing the font, size, thickness and letter spacing of the pop-up message.

Colors

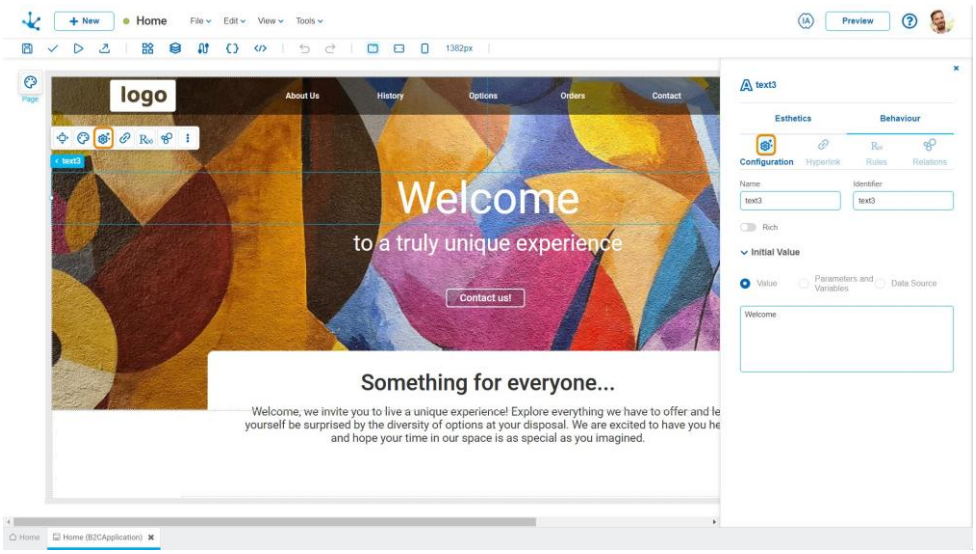
The color and opacity of the text, background and border can be defined. Borders are set uniformly for all sides of the tooltip.

Border

The border type, width, and border radius can be configured.

Configuration Properties

The configuration properties panel of an element opens when clicking the icon  of its context menu.



Name
Name used to reference the element during modeling.

Identifier
Uniquely identifies the element. It is used in the Javascript SDK.

Tooltip
Pop-up message that appears when hovering over an element, providing additional information.

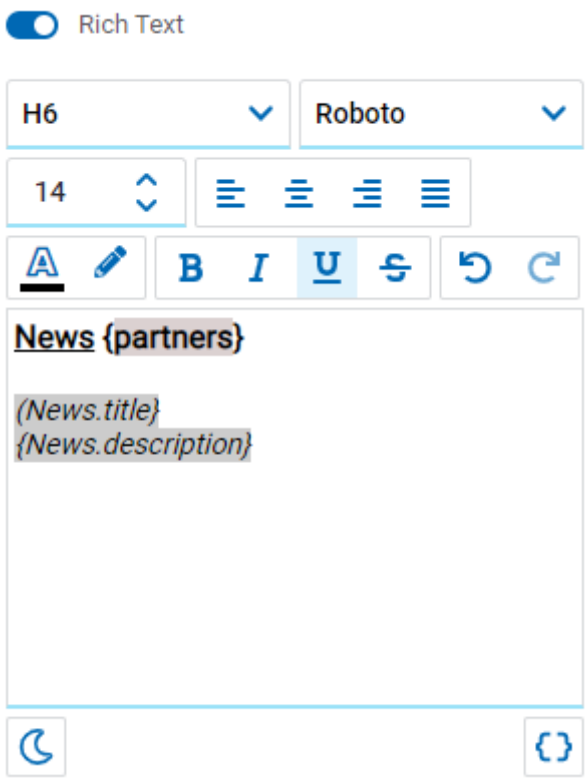


Text
Content of the message being displayed.

Direction
Position of the message in relation to the element (top, bottom, left, or right).

Rich Text

It allows defining formats that modify the appearance of the text element's content. Different styles can be combined in the same text, some parts can be highlighted, and more than one data source, parameter, or variable can be added.



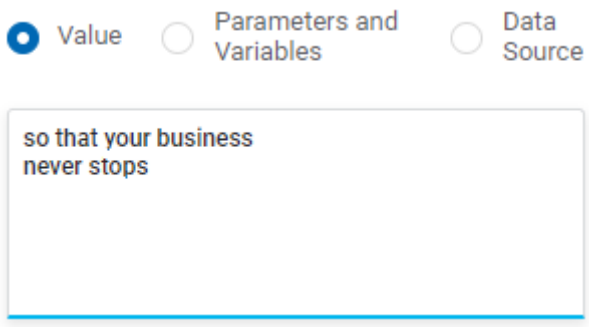
Changes the style of the code editor to dark mode.

Changes the style of the code editor to the previous mode.

It allows the incorporation of the content of variables, parameters, and information from different data sources into the text.

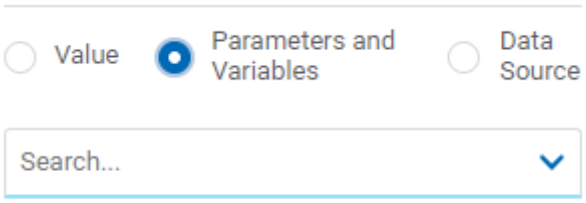
Initial Value

It allows selecting the source of the element's content and optionally defining the text style.



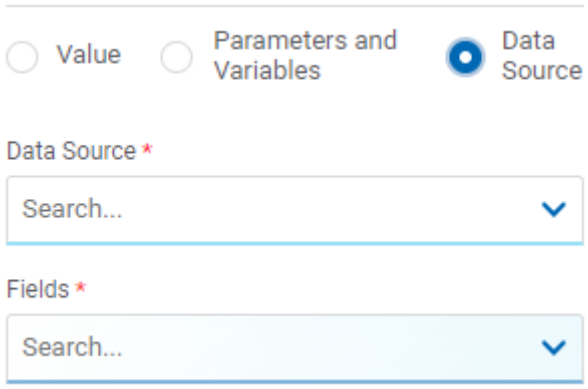
Value
Allows to enter a text that is displayed in the element.

Parameters and Variables



Parameters and Variables
Allows to select a parameter or a variable from, whose value is displayed in the element.

Data Source

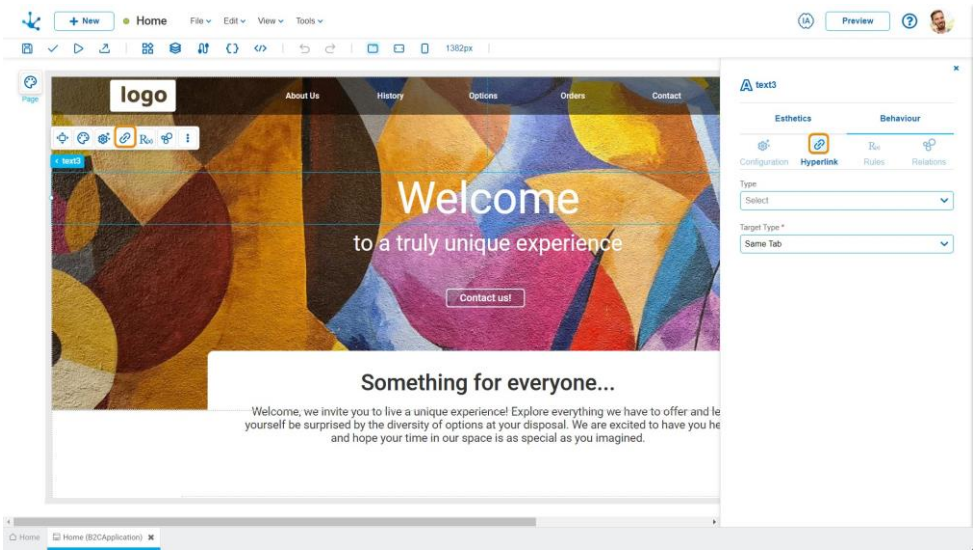


Data Source
It allows selecting a previously defined data source within the object.

Fields
It allows selecting a field from the data retrieved in the chosen data source, whose value is displayed in the element.

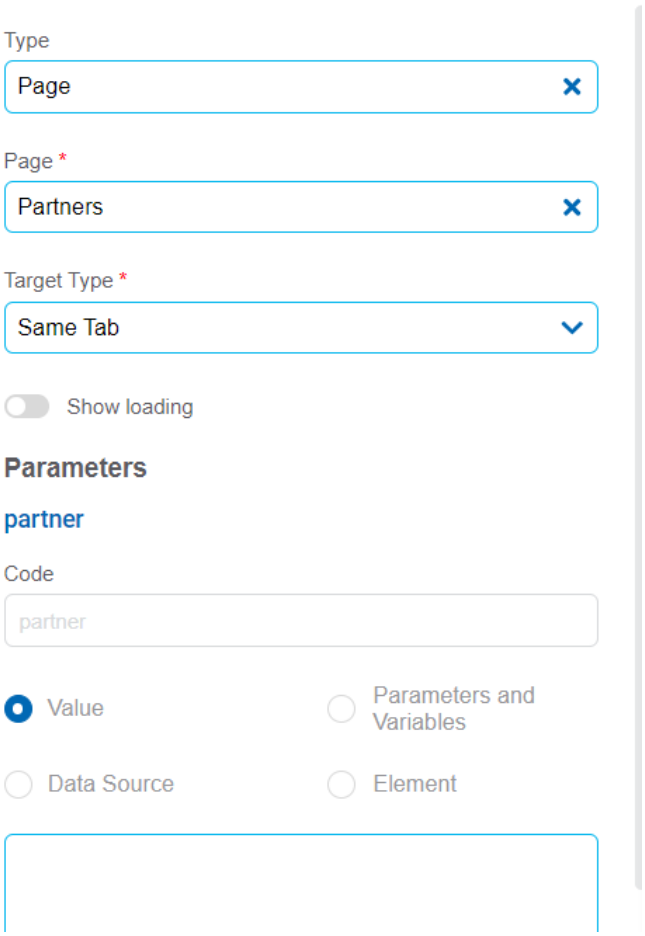
Hyperlink Properties

The hyperlink properties panel of an element opens when clicking the icon  of its context menu.



Define the behavior of the element when selecting it. Different properties are enabled depending on the type of object selected.

Page



Page
The pages modeled in the environment are displayed.

Destination Type
The available options for opening the object are displayed.

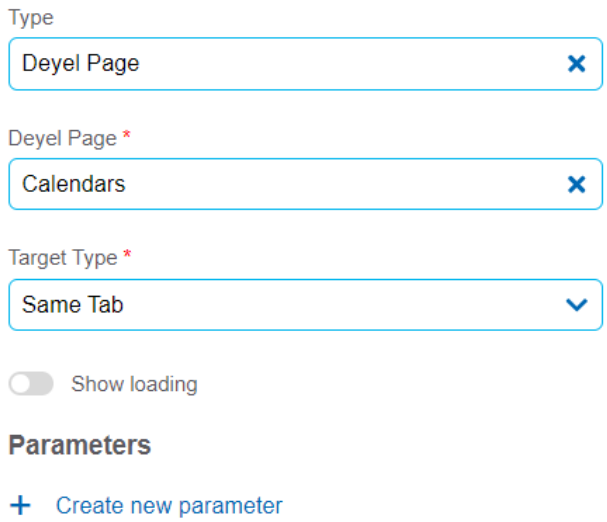
Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  - Modal Horizontal Size
    Defines its width.
  - Modal Vertical Size
    Defines its height.
- Iframe: selecting this option enables an additional property.
  - Destination Iframe
    Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Parameters
If there are parameters, their value can be specified with text, parameters and variables or data source.

Deyel Page



Deyel Page
The pages belonging to **Deyel** are displayed.

Destination Type
The available options for opening the object are displayed.

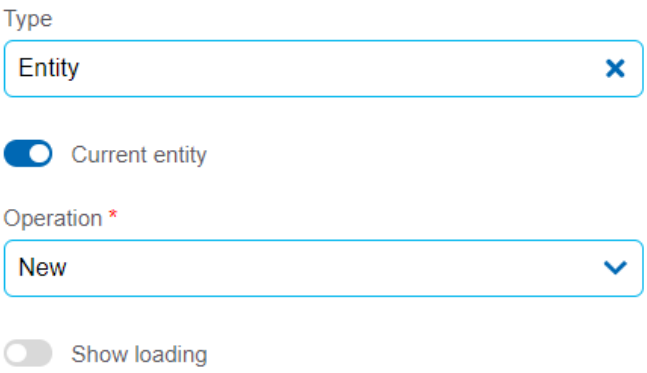
Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  Modal Horizontal Size
  Defines its width.
  Modal Vertical Size
  Defines its height.
- Iframe: selecting this option enables an additional property.
  Destination Iframe
  Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Parameters
Allows sending parameters to the selected object type.

Entity



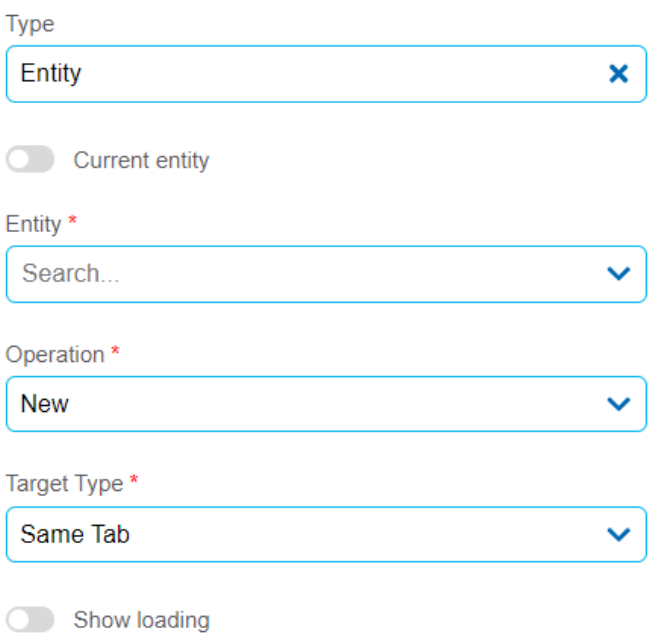
Current Entity
If this property is checked, the operation applies directly to the entity being modeled.

Operation
Defines the operation made when selecting the object.

Possible Values

- New: Indicates that the current entity panel is opened for the creation of an instance.
- Modify: Indicates that the current entity panel is opened for the update of an instance.

- Delete: Indicates that the current entity panel is opened for the deletion of an instance.



Entity
If the property Current Entity, is not checked, the entities modeled in the environment are displayed.

Operation
Defines the operation made when selecting the object.

Possible Values

- New: The panel of the entity selected in the previous property is opened to create an instance.
- Update The panel of the entity selected in the previous property is opened to update an instance.
- Show: The panel of the entity selected in the previous property is opened to show an instance.
- Grid: Indicates that the grid of the entity selected in the previous property is opened.

Destination Type
The available options for opening the object are displayed.

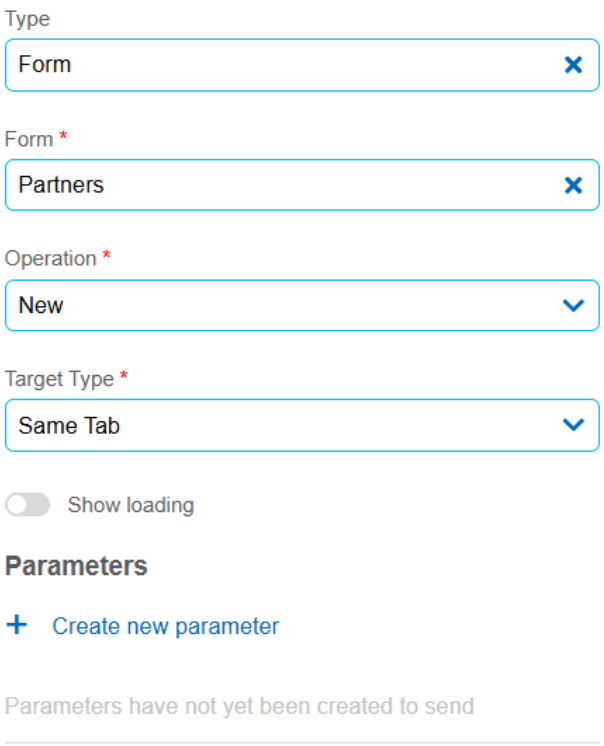
Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  Modal Horizontal Size
  Defines its width.
  Modal Vertical Size
  Defines its height.
- Iframe: selecting this option enables an additional property.
  Destination Iframe
  Expands the iframes previously defined on the object.

- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Forms



Forms
The forms modeled in the environment are displayed.

Operation
Defines the operation made when selecting the object.

Possible Values

- Create: Indicates that the panel of the object selected in the previous property is opened for the creation of an instance.
- Update: Indicates that the panel of the object selected in the previous property is opened for the update of an instance.
- Show: Indicates that the panel of the object selected in the previous property is opened for the query of an instance.
- Grid: Indicates that the grid of the object selected in the previous property is opened.

Destination Type
The available options for opening the object are displayed.

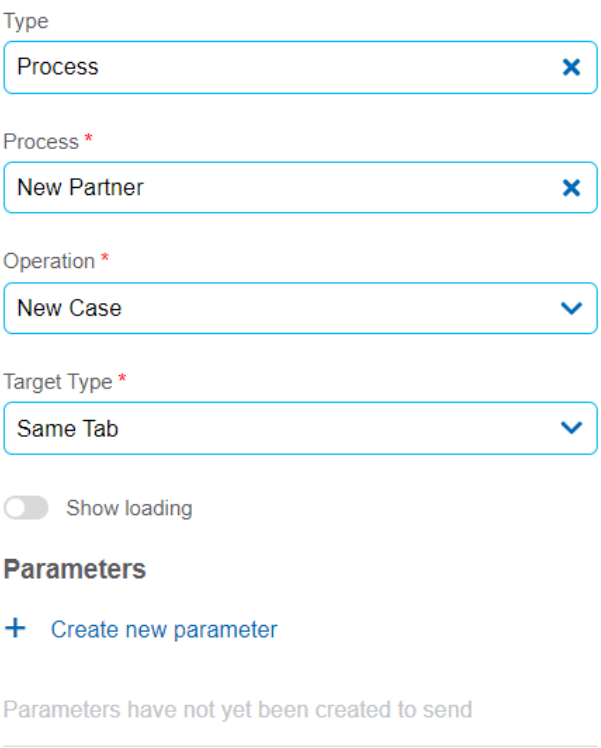
Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  Modal Horizontal Size
  Defines its width.
  Modal Vertical Size
  Defines its height.
- Iframe: selecting this option enables an additional property.
  Destination Iframe
  Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Parameters
Allows sending parameters to the selected object type.

Process



Process
The processes modeled in the environment are displayed.

Operation
Defines the operation made when selecting the object.

Destination Type
The available options for opening the object are displayed.

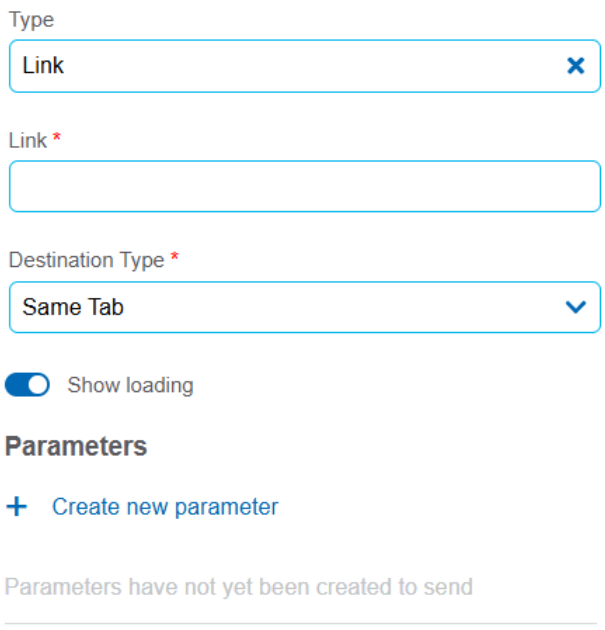
Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  - Modal Horizontal Size
    Defines its width.
  - Modal Vertical Size
    Defines its height.
- Iframe: selecting this option enables an additional property.
  - Destination Iframe
    Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Parameters
Allows sending parameters to the selected object type.

Link



Link

Allows to enter any link.

Destination Type
The available options for opening the object are displayed.

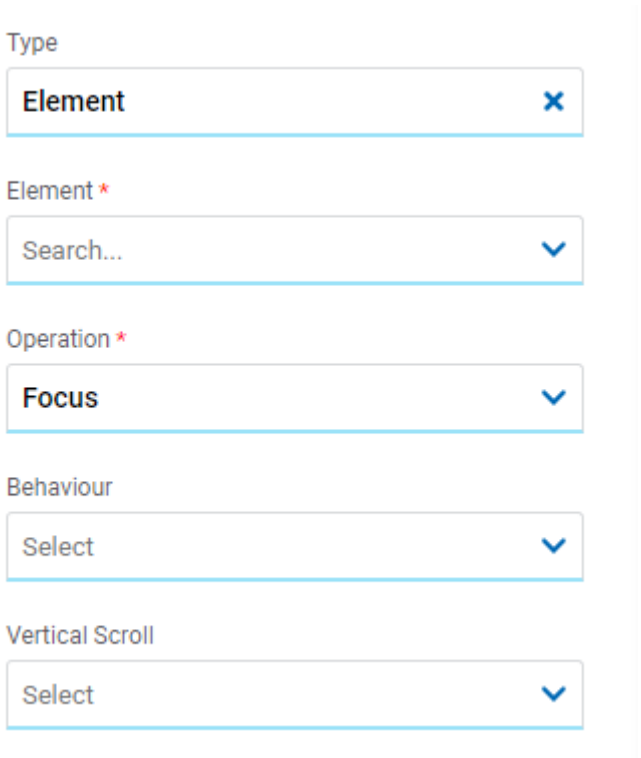
Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  - Modal Horizontal Size
    Defines its width.
  - Modal Vertical Size
    Defines its height.
- Iframe: selecting this option enables an additional property.
  - Destination Iframe
    Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Parameters
Allows sending parameters to the selected object type.

Element



Element
The modeled elements in the object are displayed.

Operation
Defines the operation made when selecting the element.

Possible Values

- Focus
- Show
- Hide
- Show/Hide

Behaviour
Establishes the transition of the animation. It is only modelable if the "Focus" operation was selected.

Possible Values

- Auto
- Smooth

Vertical Scroll
It is only modelable if the "Focus" operation was selected.
Possible Values

- Start: Moves to the start of the selected element.
- Closest: Moves to the closest position to the selected element from the element the event fires.
- Center: Moves to the center of the selected element.
- End: Moves to the end of the selected element.

Repeater



Repeater
The modeled repeaters from the object are displayed.

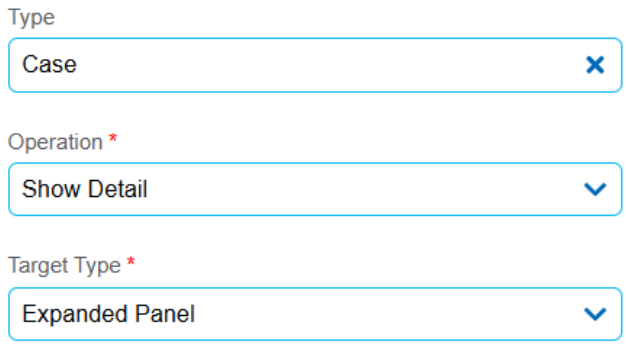
Operation

Defines the operation made when selecting the element.

Possible Values

- New: Allows for the creation of an item within the repeater. If this option is selected the additional properties are enabled.
  Creation Order
  It indicates in which position the new item of the container is added.
  Possible Values
  - Home
  - End
- Delete: Allows for the deletion of an item within the repeater.

Case



Case
Show case information.

Operation
Defines the operation made when selecting the element.

Possible Values

- Show Detail
- Show Graphic Execution

Destination Type
The available options for opening the object are displayed.

Possible Values

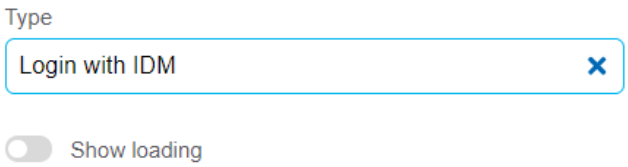
- Expanded Panel
- Modal: if this option is selected the additional properties are enabled.
  Modal Horizontal Size
  Define its width.
  Modal Vertical Size
  Define its height.

Back



It allows associating the event to go back in the browser to the element.

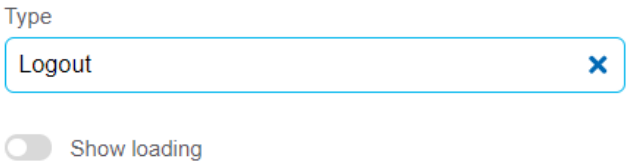
LogIn with IDM



Allows login with IDM.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

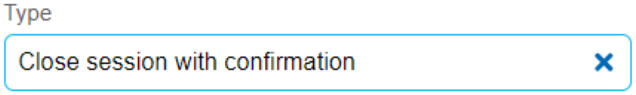
Logout



Allows logging the user out.

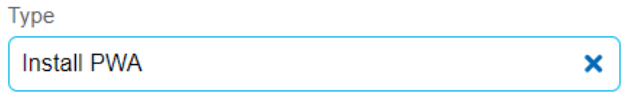
Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Close Session with Confirmation



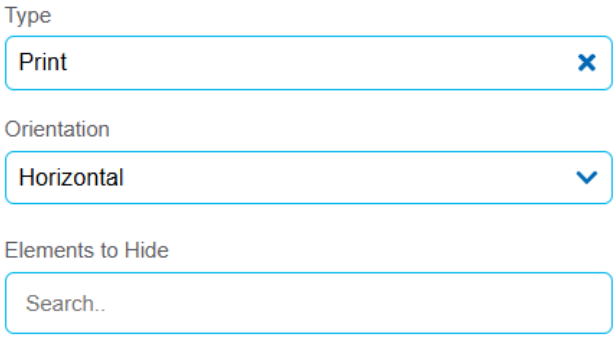
Displays a confirmation modal to logout the user.

Install PWA



Allows to install the application in the browser.

Print



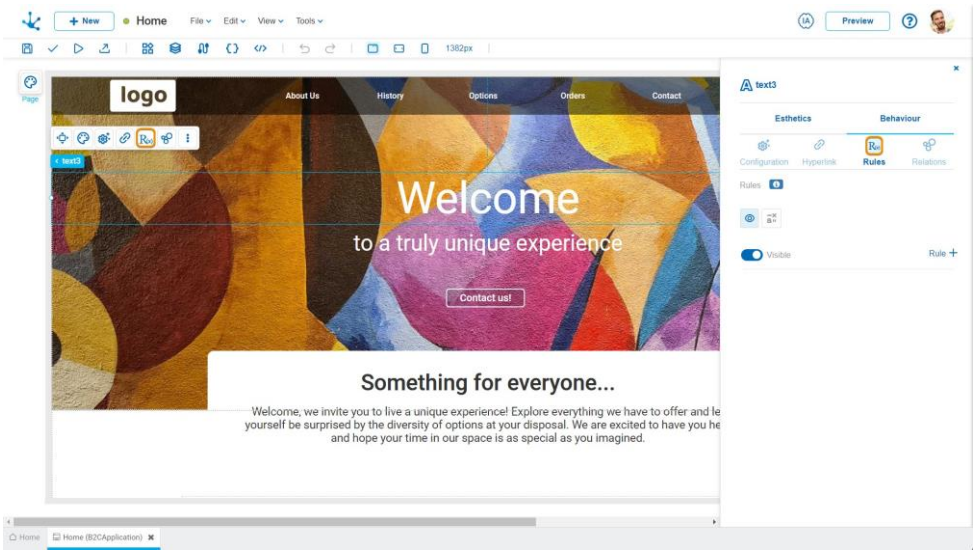
This type of hyperlink associated with an element allows printing the content of the entity or page that is being executed.

Orientation
Determines the layout of the content to be printed.
Possible Values
- Horizontal
- Vertical

Elements to hide
It allows selecting which elements of the entity or page should not be included in the printout.

Rule Properties

The rule properties panel of an element opens when clicking the icon  of its context menu.



Properties

Rules

Embedded rules on behavior, validation and calculation can be defined, associated with an element, by using the wizard (ctrl + space).

Shows syntax examples for writing the rules.

Visible

Indicates whether the element is visible. If this property is not checked, the element is not displayed in the page.

Visible (default) Not visible

Rule + Opens an edit area where a rule to determine the visibility condition can be defined. If a rule is defined, the icon is displayed with light blue borders.

Calculation

Rule + Opens an edit area where you can define the expression to be executed to calculate the element value. If a rule is defined, the icon is displayed with light blue borders.

Operations when defining the rule:

Saves the new or modified rule

Cancels the operation

Operations once the rule is defined:

Edits the existing rule

Deletes the rule

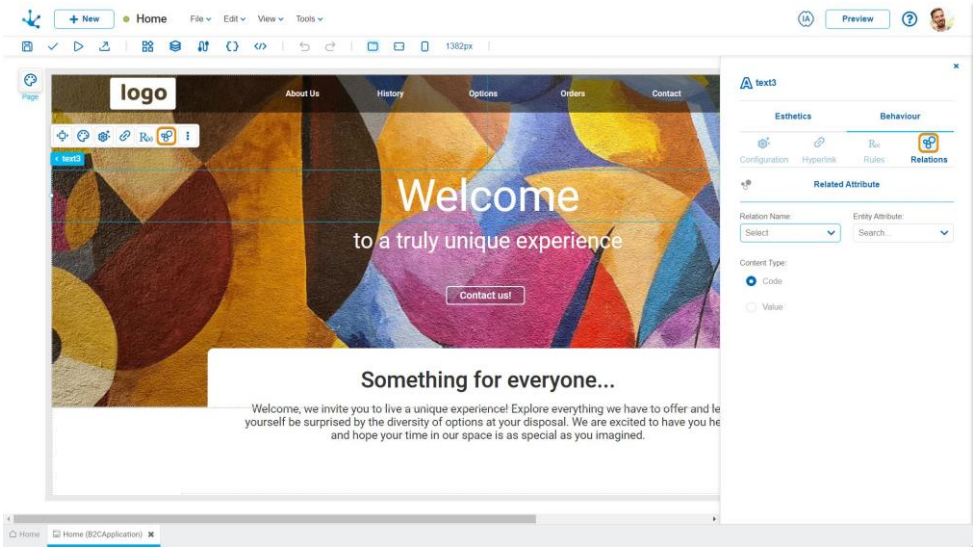
Events

Texts allow the use of different events.

Event	Description
onMouseIn()	It is executed when the cursor is positioned on the element.
onMouseOut()	It is executed when the cursor moves out of the element.
onClick()	It is executed when clicking on the element.
onDoubleClick()	It is executed when clicking twice on the element.
onInit()	It is executed before the element is loaded.
afterViewInit()	It is executed after viewing the element.
onViewportEnter()	It is executed when the element is visible.
onViewportLeave()	It is executed when the element is no longer visible.

Relation Properties

The relation properties panel of an element opens when clicking the icon  of its context menu.



Related Attribute

A related attribute indicates that the field value is retrieved from the attribute value of another entity. One of the relations defined on the page must be selected, and then it should be specified which attribute of the related entity it links to.

A field can be modeled as a related attribute and it can also have a relation defined to an entity, a value list, or a rule. This allows the value of the related attribute to be retrieved on this field and the functionality provided by the relation to be used.

Properties

Relation Name
The list of relations defined on the page is displayed, allowing the selection of one of them.

Entity Attribute
The list of attributes of the selected entity compatible with the field type is displayed. Attributes can have a relation modeled with an entity, a value list, or a rule in the selected entity.

Content Type
The element content varies depending on whether the Code or Value property is checked when:

- The entity attribute has a modeled relation.
- The field has the option "Entity" selected in the Values Obtained from property.

Code
When this property is checked, if the relation is modeled with a:

- Related entity, the identifier of the entity is displayed.
- Value list, the value list code is displayed.
- Rule, the code returned by the rule's execution is displayed.

Value
When this property is checked, if the relation is modeled with a:

- Related entity, the short description of the entity is displayed.
- Value list, the value list description is displayed.
- Rule, the description returned by the rule's execution is displayed.

Button

A button can be included on a page, defining the characteristics by modeling its properties.

The element properties are represented by icons on its context menu, where its operations are also available.

Subtypes

By selecting the “Button” option from the icon on the top toolbar, a list with the different subtypes of this element is displayed, which can be dragged to the modeling area. Each subtype has modeled element properties, predefined in a specific way.

- Primary
- Secondary
- Tertiary

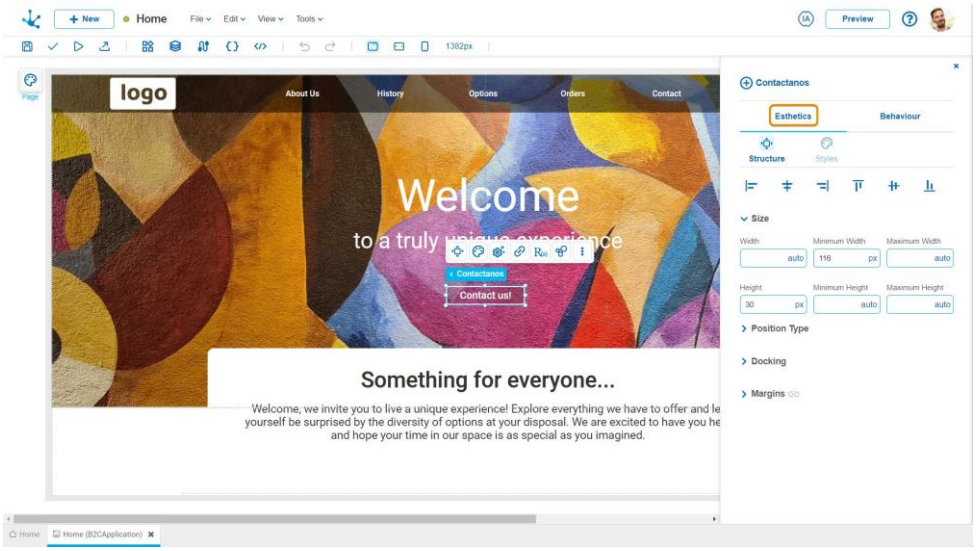
Classification of Properties

Properties are divided into two groups: esthetic properties and behavior properties.

Esthetic Properties

In the esthetic properties panel, the following are grouped:

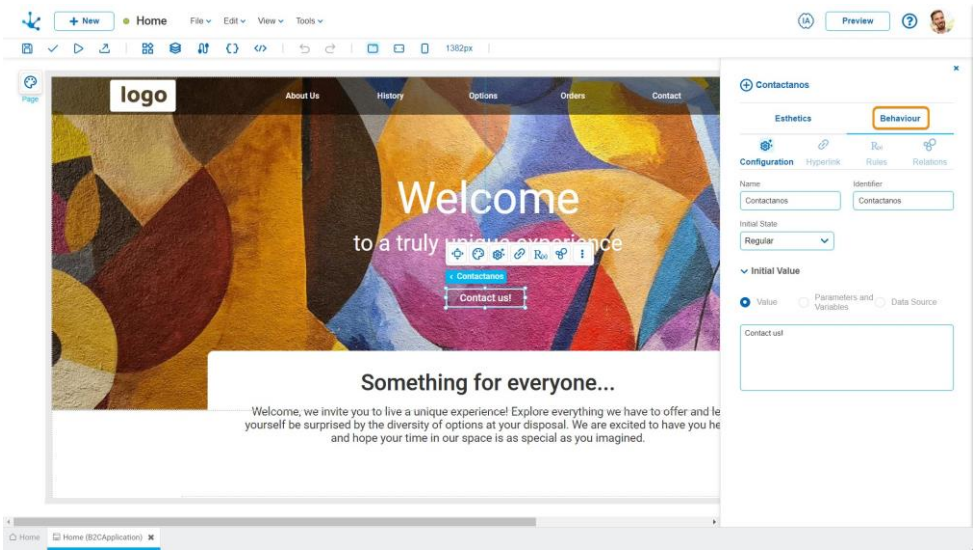
- Structure Properties
- Style Properties



Behavior Properties

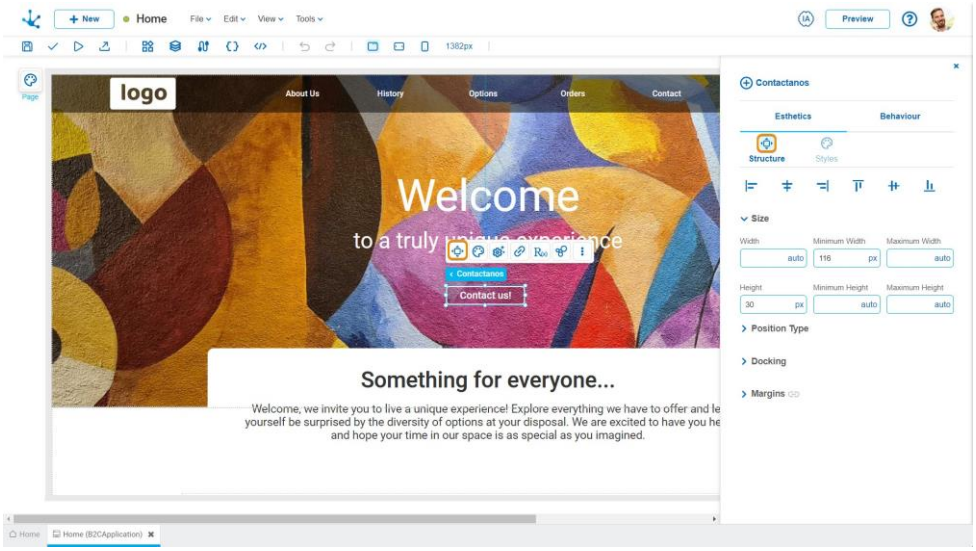
In the behavior properties panel, the following are grouped:

- Configuration Properties
- Hyperlink Properties
- Rules Properties
- Relations Properties



Structure Properties

The structure properties panel of an element opens when selecting the icon  of its context menu.



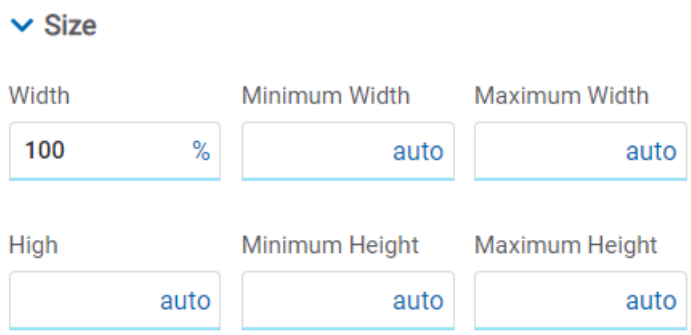
Alignment

Allows to align the element by selecting one of the available icons.

-  Align to the left.
-  Align to horizontal center.
-  Align to the right.
-  Align up.

- Align vertical center.
- Align bottom.

Size

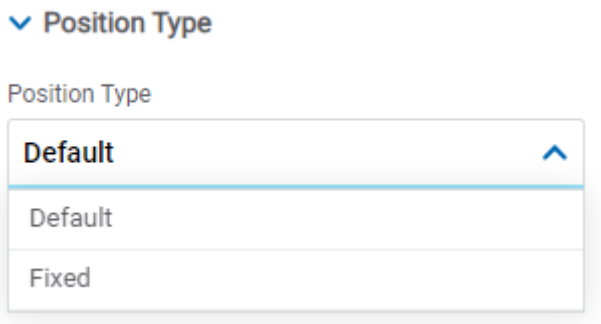


All size properties can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh). Additionally, for Width and Height properties, the "auto" option is added.

> *If selecting percentage (%) for the width and height properties, the size is calculated relative to the top element.*

Position Type

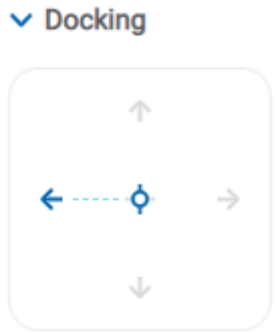
It determines if at the time of execution the element remains fixed on the page or if it moves as the scroll bar moves forward or backward.



Possible Values

- Default: The element has a relative position with respect to the superior element where it was placed (container or section).
- Fixed: Fixes the element to the page so that it is always visible in the same position. It allows to increase the visibility of important content.

Docking



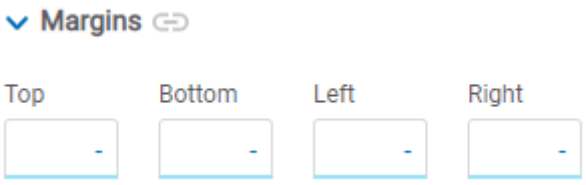
Determines the vertical and horizontal position of elements within the page section or container, when the screen is resized.

When an element is added or moved within the section or container, the modeler automatically docks it to the nearest corners or edges. It can also be configured by clicking on the arrows of the side to be docked to or on the center.

Once the element is docked, its exact position is indicated in the Margins property. The direction of the docking arrows determine the values that are automatically displayed for margins.

The docking position is indicated in the modeling area by dotted lines on the element.

Margins



It allows to define the distance of elements from the borders of their top element. The behavior of margins depends on the docking of the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Top
Distance to the top border of the highest ranking element.

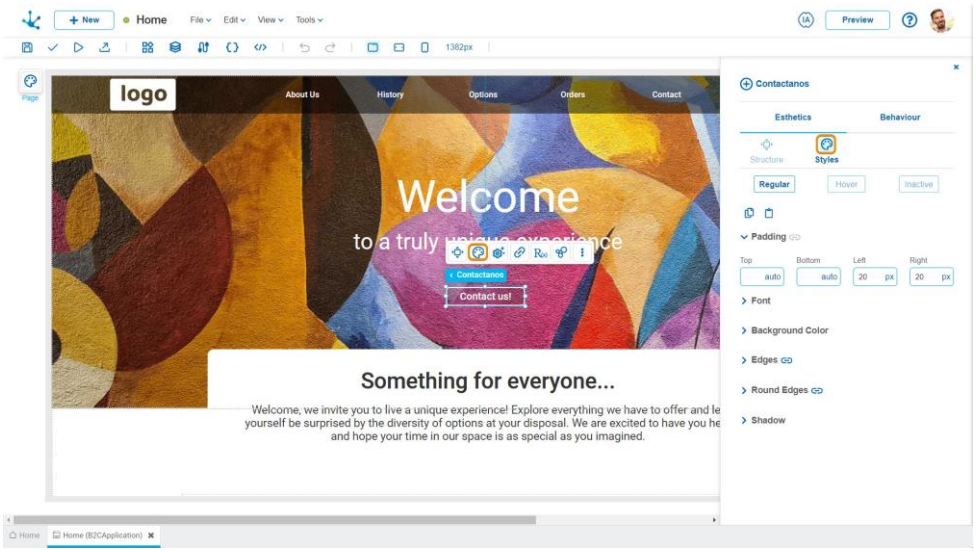
Bottom
Distance to the bottom border of the highest ranking element.

Left
Distance to the left border of the highest ranking element.

Right
Distance to the right border of the highest ranking element.

Allows the value entered in one of the margins to be copied to the other ones automatically.
Allows to indicate different values for each margin.

The style properties panel of an element opens when selecting the icon of its context menu.



This type of element may take different states and for each of them different values for its properties may be modeled.



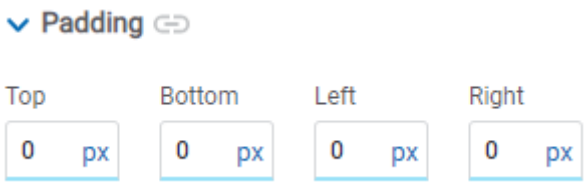
- Regular: the default state of the element, when it is enabled and the mouse pointer is not over it.
- Hover: the mouse pointer is over the element, indicating that it is interactive.
- Focus: the element has been selected using the keyboard (for example, with the Tab key) and is ready to receive an action.
- Active: the element is being pressed, either with the mouse or through a keyboard action.
- Inactive: the element is disabled and cannot receive any user interaction.

Styles

It allows copying the entire current configuration of the style panel for the selected element (Ctrl+Alt+C).

It allows applying the previously copied configuration to another element of the same type (Ctrl+Alt+V).

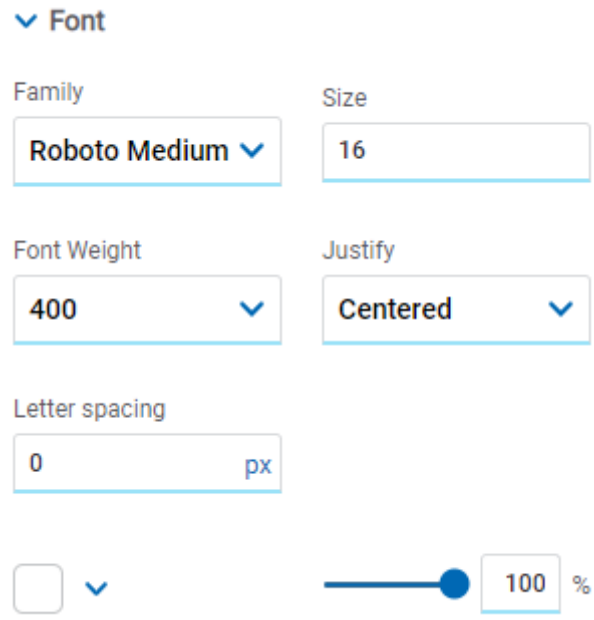
Padding



All padding space properties allow to create a space around the element borders (top, bottom, sides) and its content. The padding space configured is inward, while the margin is configured relative to a top element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Allows values entered in one of the paddings to be copied to the other ones automatically.
Allows to indicate different values for each padding.

Font

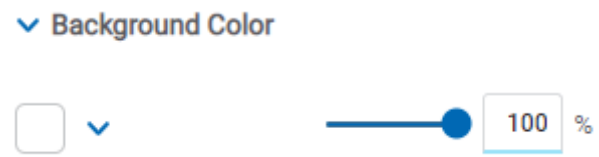


It allows to define the text style.

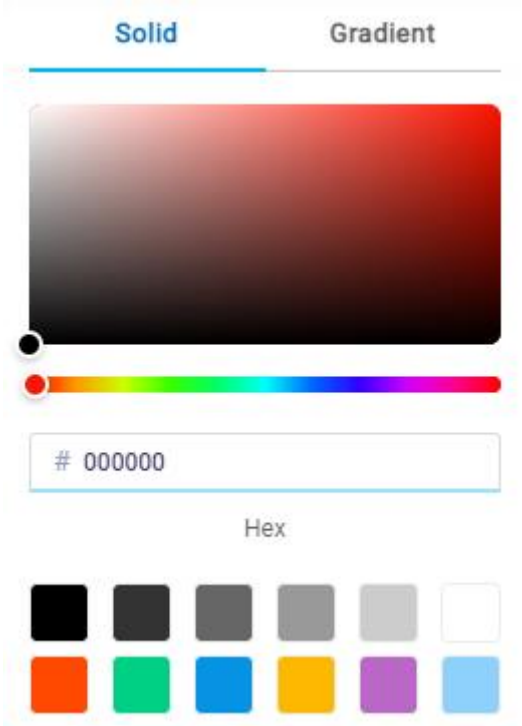
Background Color

It allows to model a background color for the element, it can be solid or gradient and different properties are defined for each one.

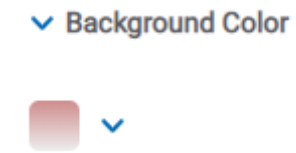
Solid



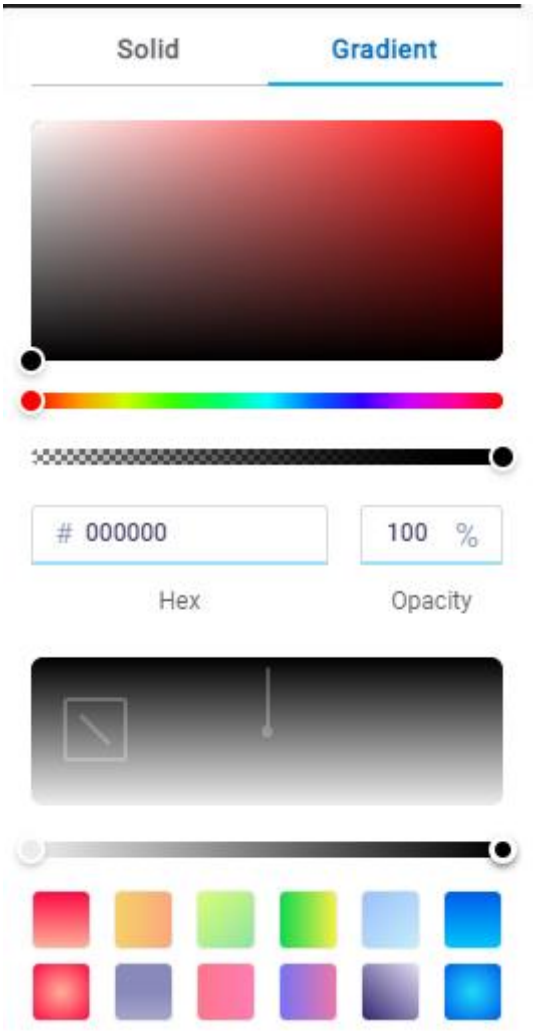
This option enables not only to select the color from a palette but also to define the degree of transparency.



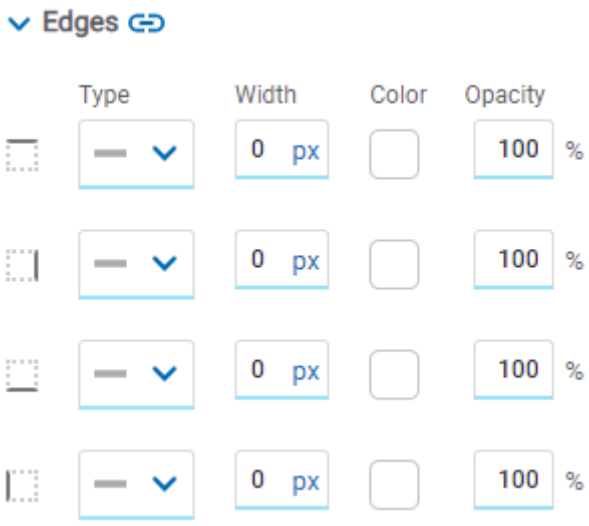
Gradient



This option enables to select not only the color from a palette but also to define its opacity and intensity according to the selected angle.



Edges



It allows to define the style of borders. Each one has its type, width, color and opacity defined, the latter as a percentage.

Allows the value entered in one of the borders to be copied to the other ones automatically.
Allows to indicate different values for each border.

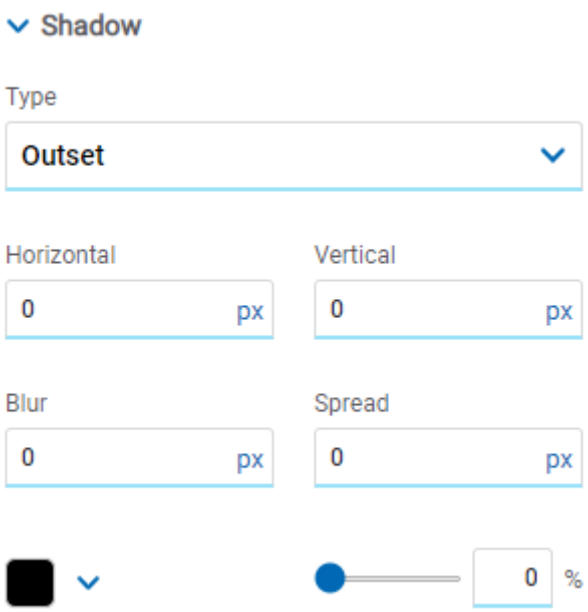
Round Edges



It allows to define the round edges at the corners of elements.
It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Allows the value entered in one of the borders to be copied to the other ones automatically.
Allows to indicate different values for each border.

Shadow



Allows to define a shadow effect around the element.

Type

Possible Values

- Outset
- Inset

Horizontal
Horizontal size of the shadow to the right of the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Vertical
Vertical size of the shadow below the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

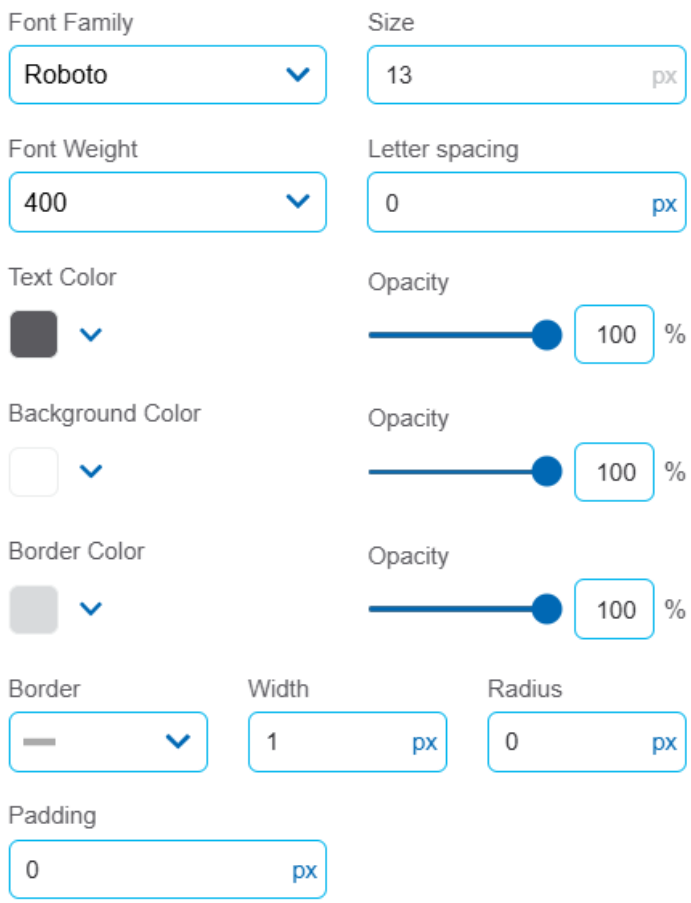
Blur
The greater this value is, the greater and lighter the shadow becomes. If not specified, its value is 0 and the shadow border is darker. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Spread
Positive values will cause the shadow to expand and grow bigger, while negative values will cause the shadow to shrink. If not specified, its value is 0 and the shadow will have the same size as the element.

Color
Defines the color of the shadow.

Transparency
Defines the transparency of the shadow.

Tooltip



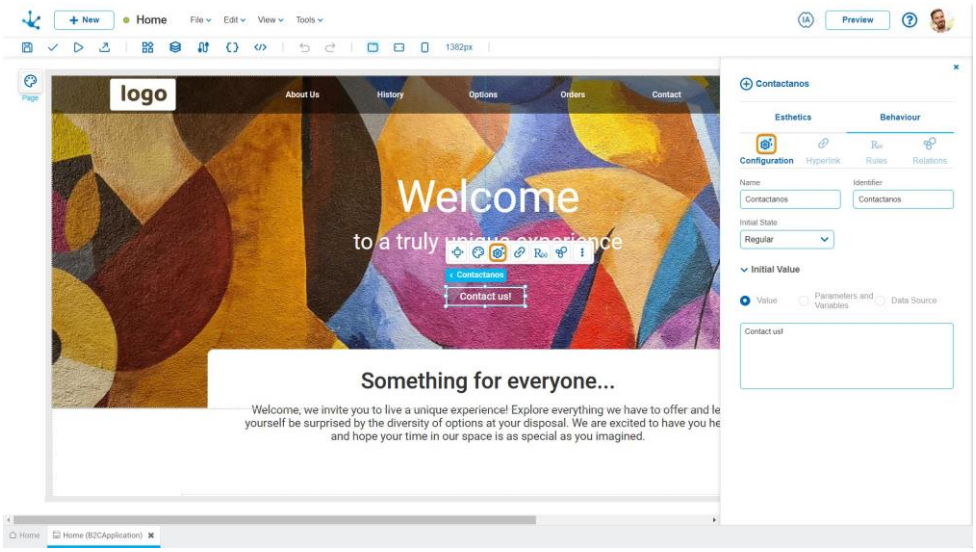
Source
It allows customizing the font, size, thickness and letter spacing of the pop-up message.

Colors
The color and opacity of the text, background and border can be defined. Borders are set uniformly for all sides of the tooltip.

Border
The border type, width, and border radius can be configured.

Configuration Properties

The configuration properties panel of an element opens when clicking the icon  of its context menu.





Name
Name used to reference the element during modeling.

Identifier
Uniquely identifies the element. It is used in the Javascript SDK.

Tooltip
Pop-up message that appears when hovering over an element, providing additional information.



Text
Content of the message being displayed.

Direction
Position of the message in relation to the element (top, bottom, left, or right).

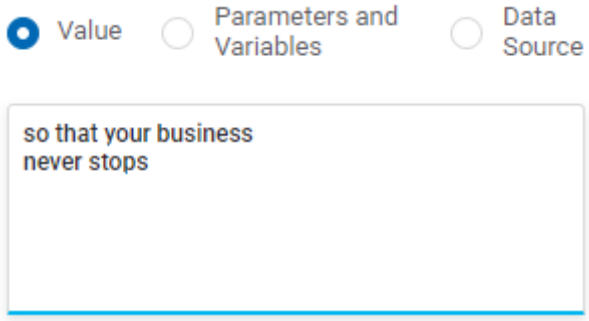
Initial State
It allows defining the state of the button when the page is loaded for the first time, ensuring that its initial behavior is as expected.

Possible Values

- Regular: The button is enabled, allowing immediate user interaction. It is used for actions that are available from the start.
- Inactive The button is disabled until certain conditions are met or a specific action is performed. Used for flows that require prior validations before enabling the button action.

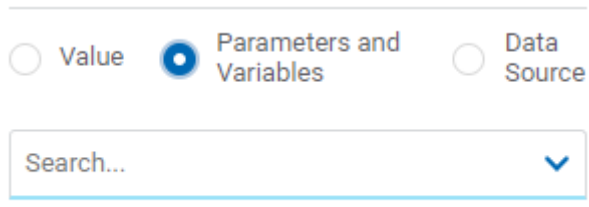
Initial Value

It allows selecting the source of the element's content and optionally defining the text style.



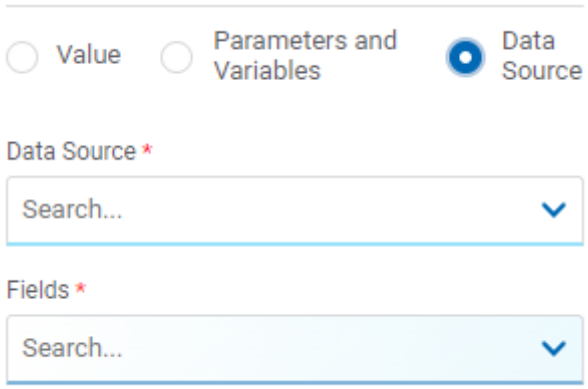
Value
Allows to enter a text that is displayed in the element.

Parameters and Variables



Parameters and Variables
Allows to select a parameter or a variable from, whose value is displayed in the element.

Data Source



Data Source

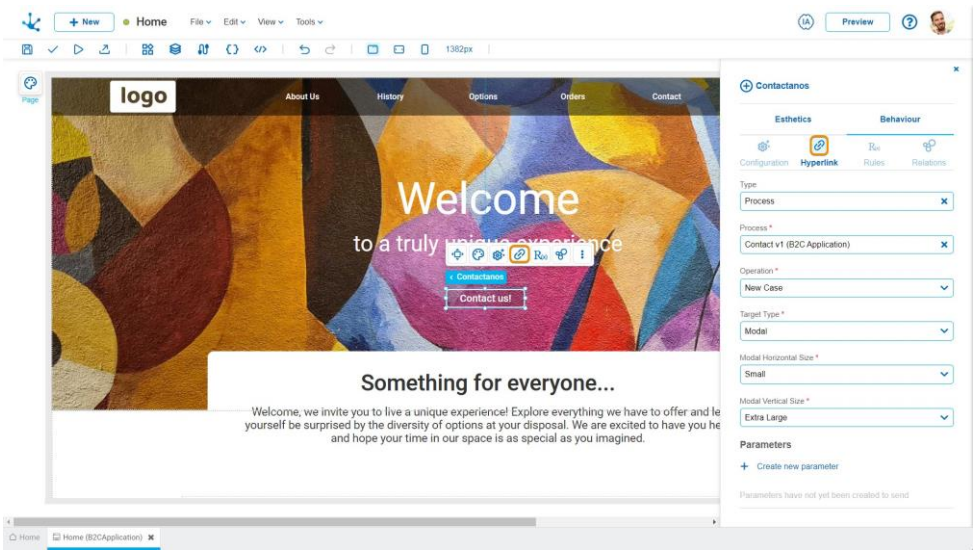
It allows selecting a previously defined data source within the object.

Fields

It allows selecting a field from the data retrieved in the chosen data source, whose value is displayed in the element.

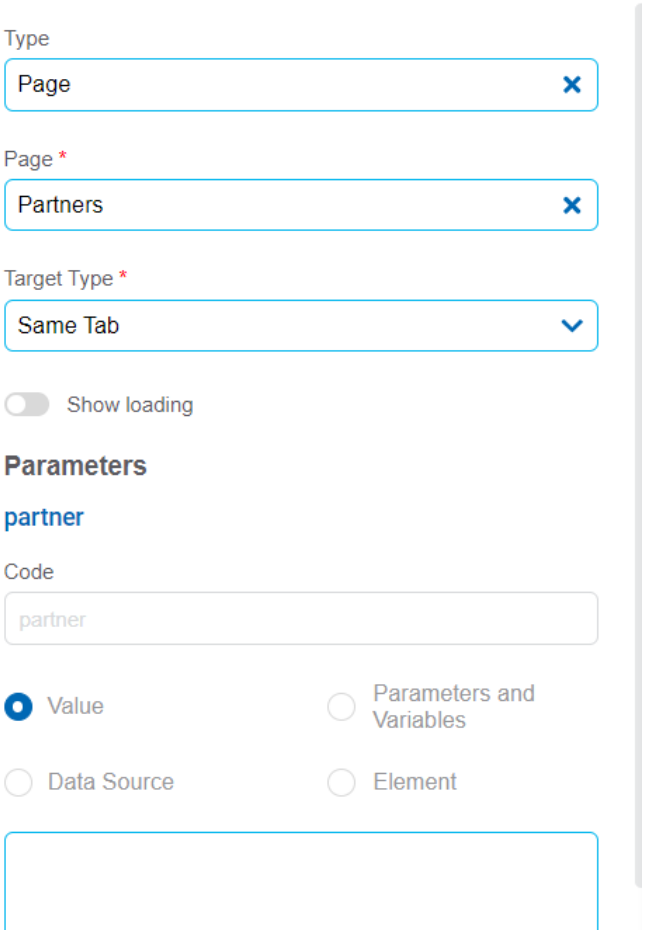
Hyperlink Properties

The hyperlink properties panel of an element opens when clicking the icon of its context menu.



Define the behavior of the element when selecting it. Different properties are enabled depending on the type of object selected.

Page



Page
The pages modeled in the environment are displayed.

Destination Type
The available options for opening the object are displayed.

Possible Values

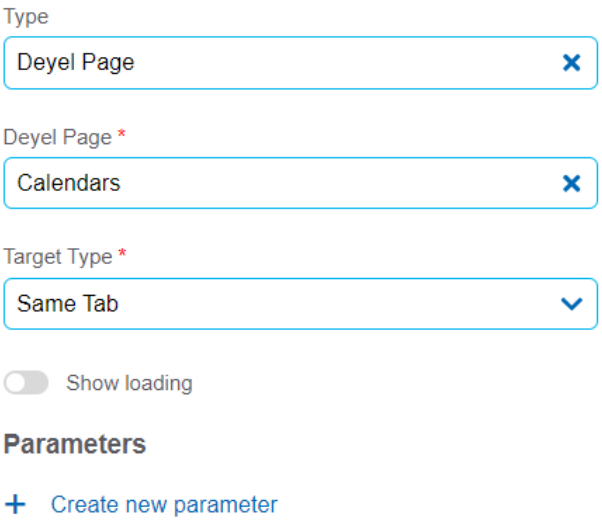
- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  Modal Horizontal Size
  Defines its width.
  Modal Vertical Size
  Defines its height.
- Iframe: selecting this option enables an additional property.
  Destination Iframe
  Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading

Shows the loading icon and disables the user interaction until the redirection is over.

Parameters
If there are parameters, their value can be specified with text, parameters and variables or data source.

Deyel Page



Deyel Page
The pages belonging to **Deyel** are displayed.

Destination Type
The available options for opening the object are displayed.

Possible Values

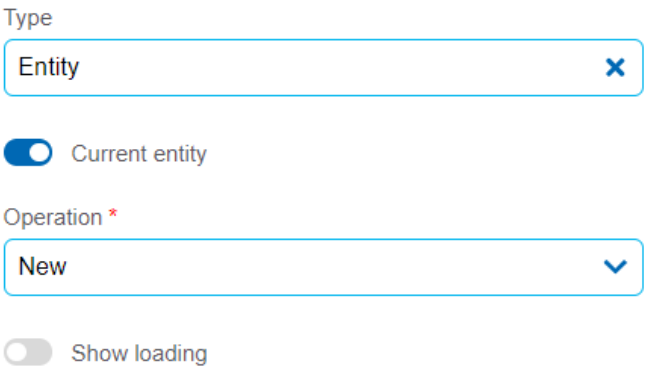
- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  - Modal Horizontal Size
    Defines its width.
  - Modal Vertical Size
    Defines its height.
- Iframe: selecting this option enables an additional property.
  - Destination Iframe
    Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Parameters

Allows sending parameters to the selected object type.

Entity



Current Entity

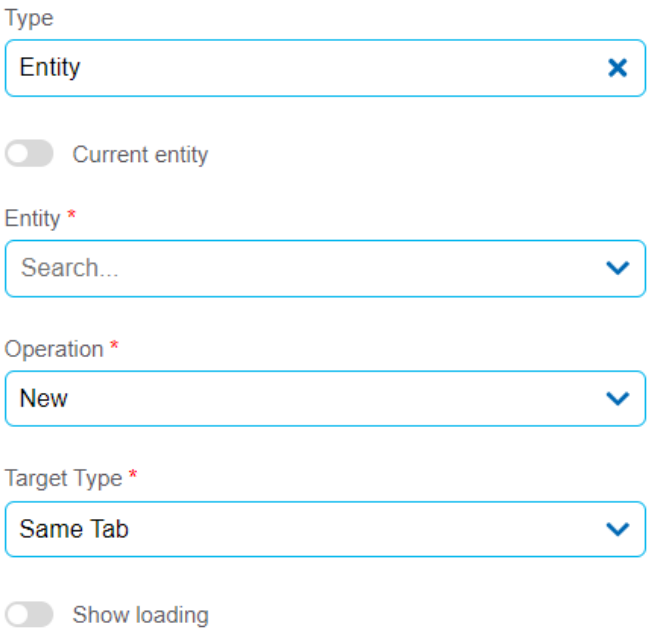
If this property is checked, the operation applies directly to the entity being modeled.

Operation

Defines the operation made when selecting the object.

Possible Values

- New: Indicates that the current entity panel is opened for the creation of an instance.
- Modify: Indicates that the current entity panel is opened for the update of an instance.
- Delete: Indicates that the current entity panel is opened for the deletion of an instance.



Entity
If the property Current Entity, is not checked, the entities modeled in the environment are displayed.

Operation
Defines the operation made when selecting the object.

Possible Values

- New: The panel of the entity selected in the previous property is opened to create an instance.
- Update The panel of the entity selected in the previous property is opened to update an instance.
- Show: The panel of the entity selected in the previous property is opened to show an instance.
- Grid: Indicates that the grid of the entity selected in the previous property is opened.

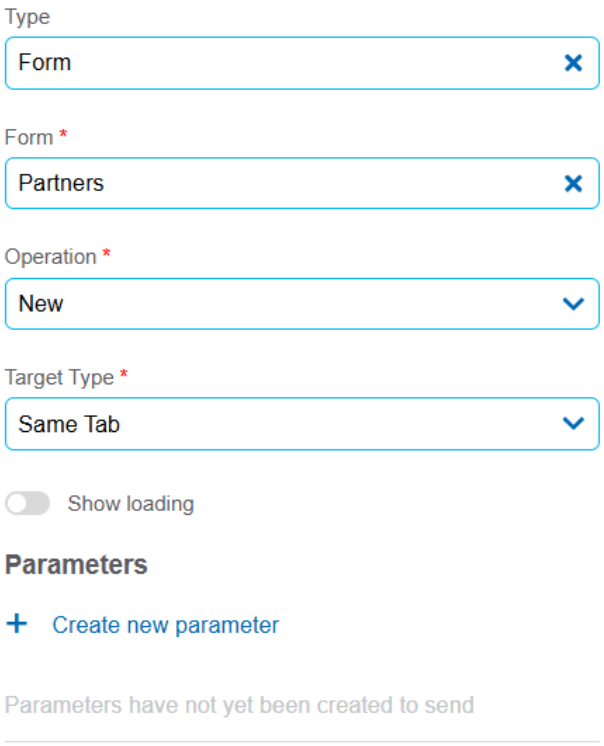
Destination Type
The available options for opening the object are displayed.

Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  Modal Horizontal Size
  Defines its width.
  Modal Vertical Size
  Defines its height.
- Iframe: selecting this option enables an additional property.
  Destination Iframe
  Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Forms



Forms
The forms modeled in the environment are displayed.

Operation
Defines the operation made when selecting the object.

Possible Values

- Create: Indicates that the panel of the object selected in the previous property is opened for the creation of an instance.
- Update: Indicates that the panel of the object selected in the previous property is opened for the update of an instance.
- Show: Indicates that the panel of the object selected in the previous property is opened for the query of an instance.
- Grid: Indicates that the grid of the object selected in the previous property is opened.

Destination Type
The available options for opening the object are displayed.

Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  - Modal Horizontal Size
    Defines its width.
  - Modal Vertical Size
    Defines its height.
- Iframe: selecting this option enables an additional property.
  - Destination Iframe

Expands the iframes previously defined on the object.

- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

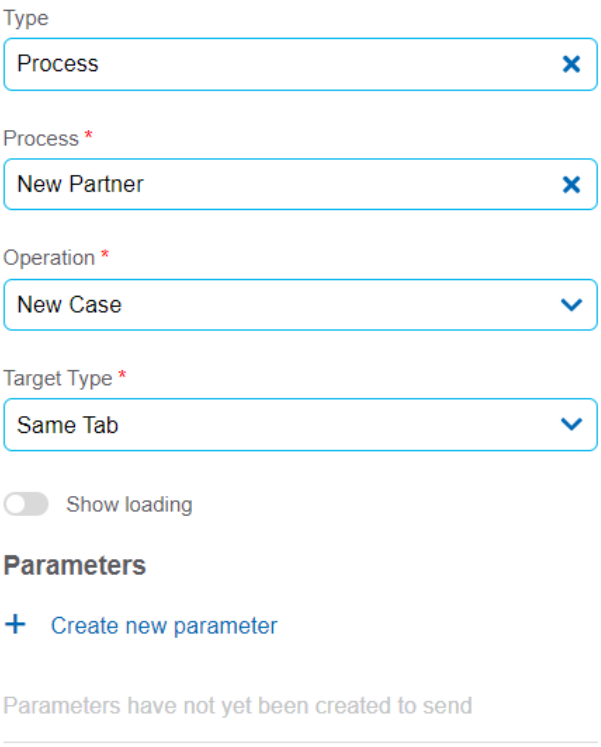
Show loading

Shows the loading icon and disables the user interaction until the redirection is over.

Parameters

Allows sending parameters to the selected object type.

Process



Process

The processes modeled in the environment are displayed.

Operation

Defines the operation made when selecting the object.

Destination Type

The available options for opening the object are displayed.

Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.

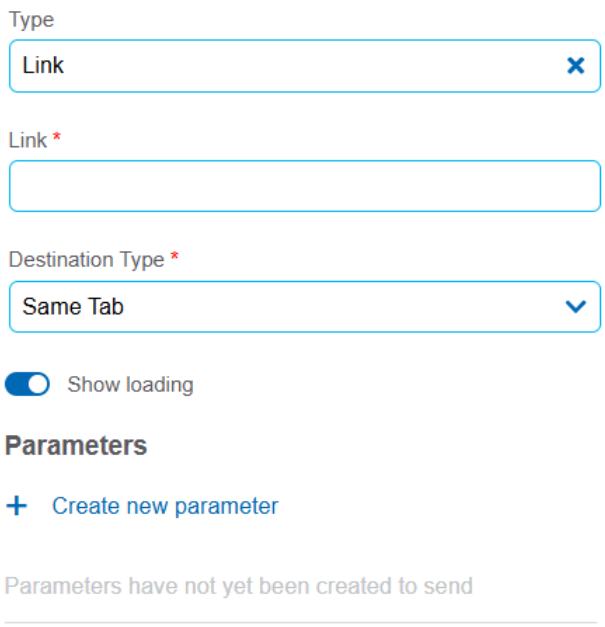
Modal Horizontal Size
Defines its width.
Modal Vertical Size
Defines its height.

- Iframe: selecting this option enables an additional property.
  Destination Iframe
  Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Parameters
Allows sending parameters to the selected object type.

Link



Link
Allows to enter any link.

Destination Type
The available options for opening the object are displayed.

Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  Modal Horizontal Size

Defines its width.

Modal Vertical Size

Defines its height.

- Iframe: selecting this option enables an additional property.

  Destination Iframe

  Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

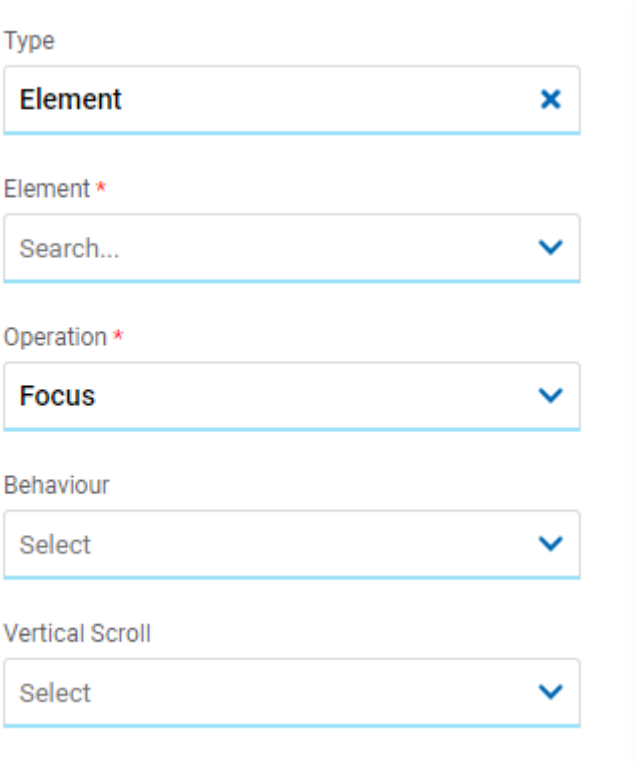
Show loading

Shows the loading icon and disables the user interaction until the redirection is over.

Parameters

Allows sending parameters to the selected object type.

Element



Element

The modeled elements in the object are displayed.

Operation

Defines the operation made when selecting the element.

Possible Values

- Focus
- Show
- Hide

- Show/Hide

Behaviour
Establishes the transition of the animation. It is only modelable if the "Focus" operation was selected.

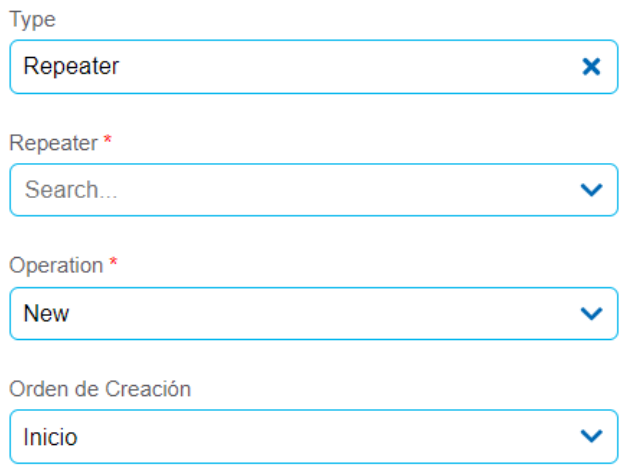
Possible Values

- Auto
- Smooth

Vertical Scroll
It is only modelable if the "Focus" operation was selected.
Possible Values

- Start: Moves to the start of the selected element.
- Closest: Moves to the closest position to the selected element from the element the event fires.
- Center: Moves to the center of the selected element.
- End: Moves to the end of the selected element.

Repeater



Repeater
The modeled repeaters from the object are displayed.

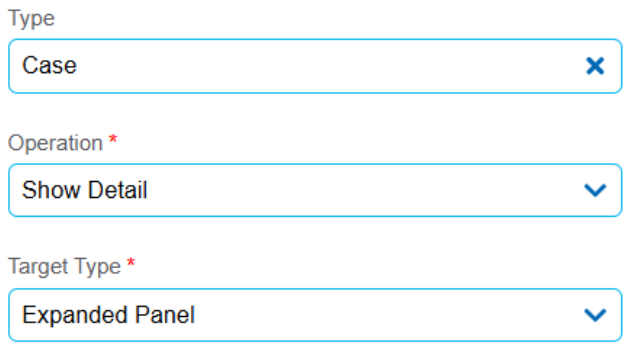
Operation
Defines the operation made when selecting the element.

Possible Values

- New: Allows for the creation of an item within the repeater. If this option is selected the additional properties are enabled.
  ### Creation Order
  It indicates in which position the new item of the container is added.
  Possible Values
  - Home
  - End

- Delete: Allows for the deletion of an item within the repeater.

Case



Case
Show case information.

Operation
Defines the operation made when selecting the element.

Possible Values

- Show Detail
- Show Graphic Execution

Destination Type
The available options for opening the object are displayed.

Possible Values

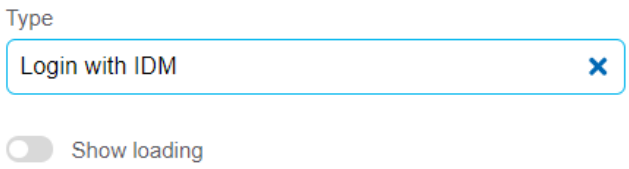
- Expanded Panel
- Modal: if this option is selected the additional properties are enabled.
  Modal Horizontal Size
  Define its width.
  Modal Vertical Size
  Define its height.

Back



It allows associating the event to go back in the browser to the element.

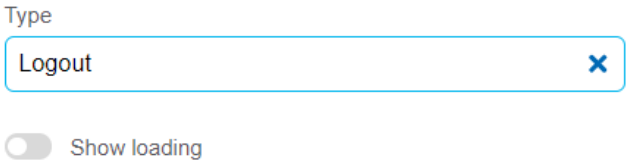
LogIn with IDM



Allows login with IDM.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

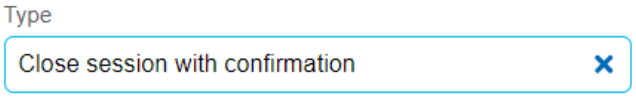
Logout



Allows logging the user out.

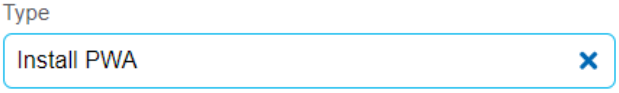
Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Close Session with Confirmation



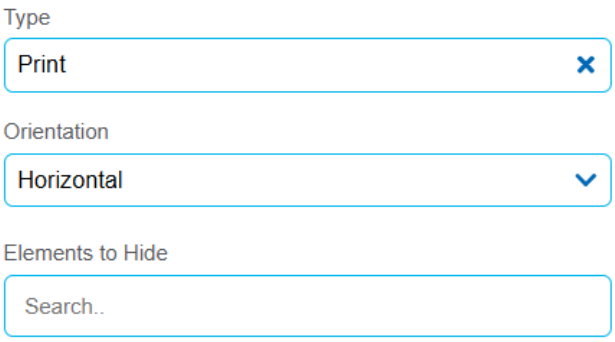
Displays a confirmation modal to logout the user.

Install PWA



Allows to install the application in the browser.

Print



This type of hyperlink associated with an element allows printing the content of the entity or page that is being executed.

Orientation

Determines the layout of the content to be printed.

Possible Values

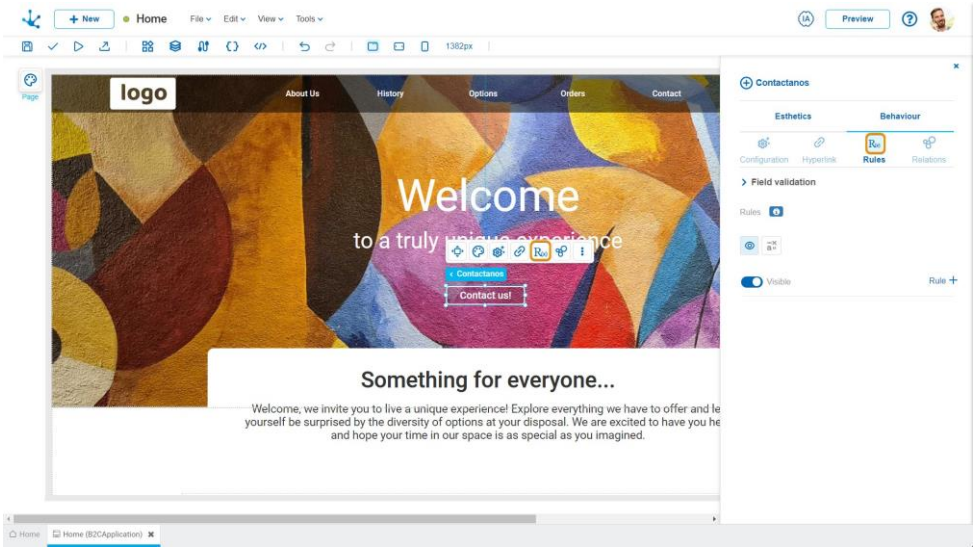
- Horizontal
- Vertical

Elements to hide

It allows selecting which elements of the entity or page should not be included in the printout.

Rule Properties

The rule properties panel of an element opens when clicking the icon  of its context menu.



Properties

Rules

Embedded rules on behavior, validation and calculation can be defined, associated with an element, by using the wizard (ctrl + space).

Shows syntax examples for writing the rules.

Visible

Indicates whether the element is visible. If this property is not checked, the element is not displayed in the page.

Visible (default) Not visible

Rule + Opens an edit area where a rule to determine the visibility condition can be defined. If a rule is defined, the icon is displayed with light blue borders.

Calculation

Rule + Opens an edit area where you can define the expression to be executed to calculate the element value. If a rule is defined, the icon is displayed with light blue borders.

Operations when defining the rule:

Saves the new or modified rule

Cancels the operation

Operations once the rule is defined:

Edits the existing rule

Deletes the rule

Events

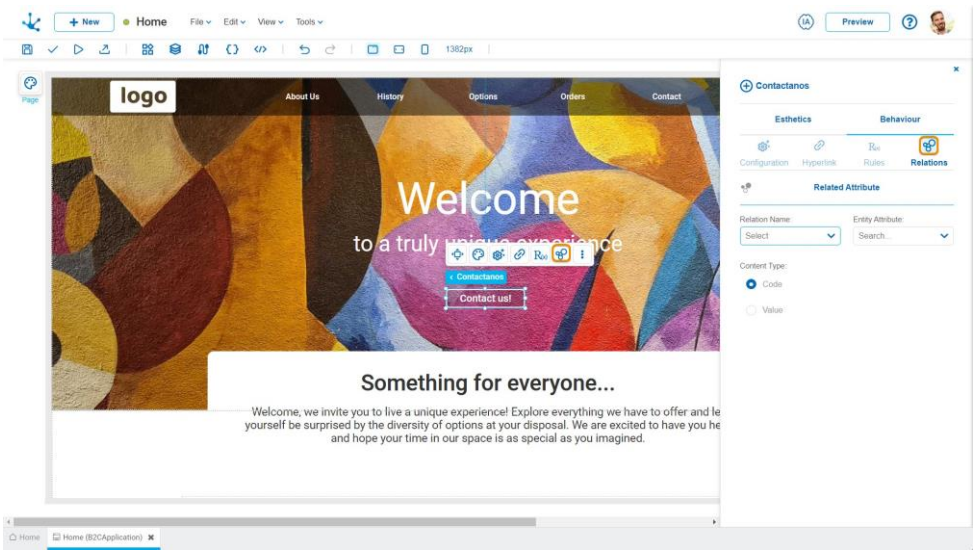
Buttons allow the use of different events.

Event	Description
onMouseIn()	It is executed when the cursor is positioned on the element.
onMouseOut()	It is executed when the cursor moves out of the element.
onClick()	It is executed when clicking on the element.
onDoubleClick()	It is executed when clicking twice on the element.
onInit()	It is executed before the element is loaded.

Event	Description
afterViewInit()	It is executed after viewing the element.
onViewportEnter()	It is executed when the element is visible.
onViewportLeave()	It is executed when the element is no longer visible.

Relation Properties

The relation properties panel of an element opens when clicking the icon of its context menu.



Related Attribute

A related attribute indicates that the field value is retrieved from the attribute value of another entity. One of the relations defined on the page must be selected, and then it should be specified which attribute of the related entity it links to.

A field can be modeled as a related attribute and it can also have a relation defined to an entity, a value list, or a rule. This allows the value of the related attribute to be retrieved on this field and the functionality provided by the relation to be used.

Properties

Relation Name
The list of relations defined on the page is displayed, allowing the selection of one of them.

Entity Attribute
The list of attributes of the selected entity compatible with the field type is displayed. Attributes can have a relation modeled with an entity, a value list, or a rule in the selected entity.

Content Type

The element content varies depending on whether the Code or Value property is checked when:

- The entity attribute has a modeled relation.
- The field has the option "Entity" selected in the Values Obtained from property.

Code

When this property is checked, if the relation is modeled with a:

- Related entity, the identifier of the entity is displayed.
- Value list, the value list code is displayed.
- Rule, the code returned by the rule's execution is displayed.

Value

When this property is checked, if the relation is modeled with a:

- Related entity, the short description of the entity is displayed.
- Value list, the value list description is displayed.
- Rule, the description returned by the rule's execution is displayed.

Container

To create an adaptive and structured design for the page, container boxes are used. Drag the elements inside the container and its style is then arranged for different displays. Elements are automatically attached to the container so they change size in direct relation to the container.

The page modeler also includes other structural elements that facilitate the fields modeling by speeding up the modeling of a page, without worrying about the aesthetic aspects of the fields. All aesthetic aspects of these elements are defined by default to adapt to different breakpoints.

The element properties are represented by icons in its context menu, where its operations are also available.

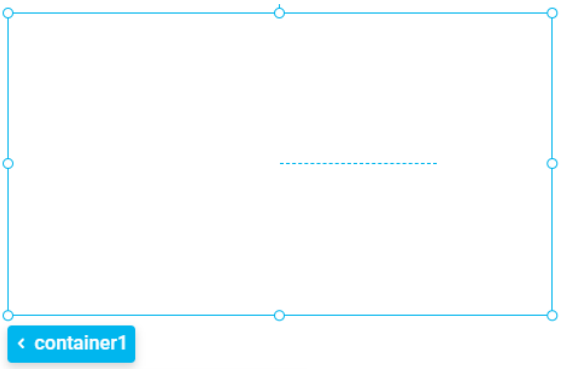
Subtypes

By selecting the "Container" option from the icon on the top toolbar, a list with the different subtypes of this element is displayed, which can be dragged to the modeling area. Each subtype has modeled element properties, predefined in a specific way.

- Simple
- Background Color
- Border
- Shadow
- Collapsible
- Repetitive Group Fields
- Collapsible Repetitive

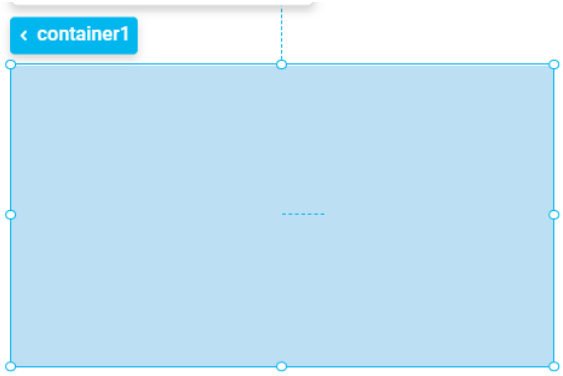
Simple

It allows modeling a simple-style container. The content can be any element.



Background Color

It allows modeling a container with a preset background color, which can be modified. The content can be any element.



Border

It allows modeling a container with a preset background color and a border, which can be modified. The content can be any element.

Shadow

It allows modeling a container with a preset background color, a border, and a shadow, which can be modified. The content can be any element.

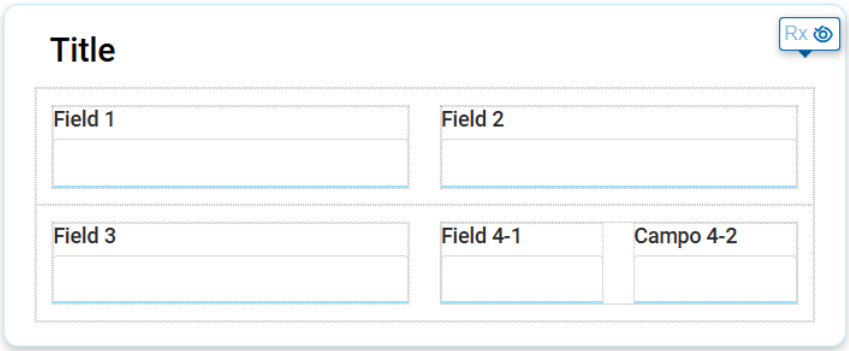


Collapsible

It allows grouping fields and offer drop-down behavior. This means that its content can be easily expanded or closed, by clicking on the corresponding icon, ↓ or ↑, depending on the state of the container.

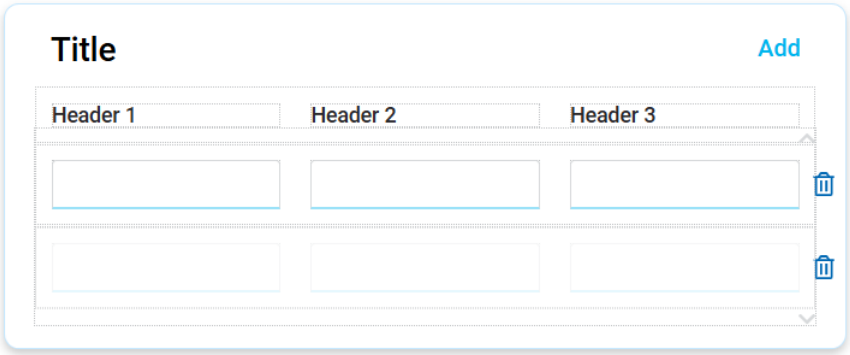
This container not only groups fields, but can also contain any other element, providing great flexibility in design.

The collapsible container is primarily used within design elements, to visually organize entities with a large number of fields. By expanding or collapsing the container, information can be viewed in a more organized manner, making it easier to navigate through data groups.



Repetitive Group of Fields

It allows organizing repetitive field structures in a grid format, with descriptions for each field and headers for columns. Additionally, being a repetitive structure, it includes the facility to add and delete rows within the grid.

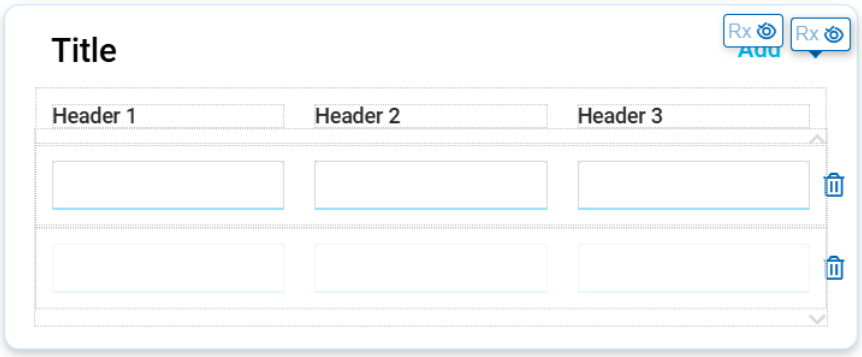


Collapsible Repetitive Card Group

It allows organizing repetitive field structures in a grid format, with descriptions for each field and headers for the columns. Additionally, being a repetitive structure, it includes the facility to add and delete rows within the grid.

This container has a fixed size, and items are scrolled page by page within the repeater, using the icons and . In the repeater configuration properties panel, it is modeled as subtype "card" and the number of rows to paginate.

The collapsible repetitive tape group is ideal for saving multiple instances of the same type of information while making it easier to display entities with many fields.

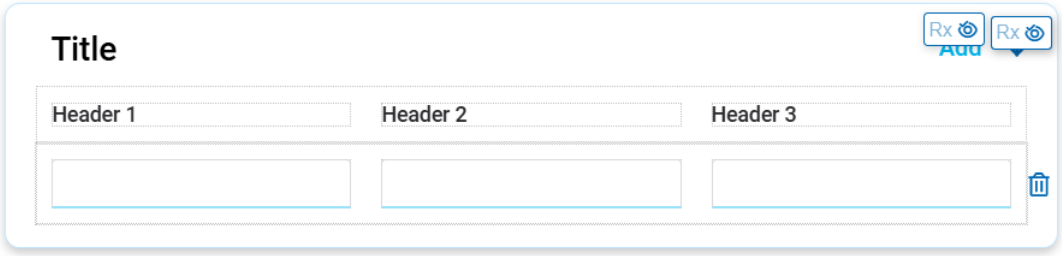


Collapsible Repetitive Tape Group

It allows organizing repetitive field structures in a grid format, with descriptions for each field and headers for the columns. Additionally, being a repetitive structure, it includes the facility to add and delete rows within the grid.

This container expands indefinitely as new items are added, offering greater flexibility. In the repeater configuration properties panel, it is modeled as "tape" subtype.

The collapsible repetitive tape group is ideal for saving multiple instances of the same type of information while making it easier to display entities with many fields.



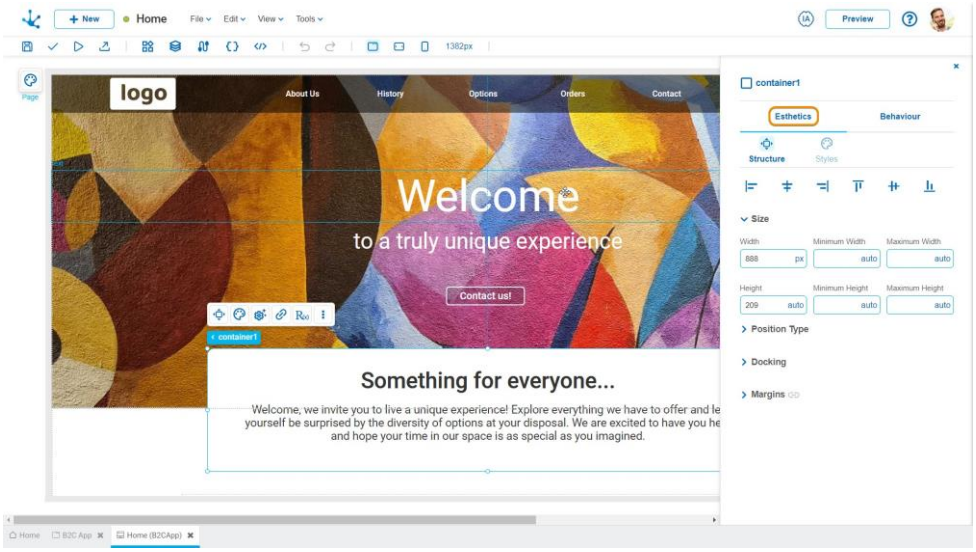
Classification of Properties

Properties are divided into two groups: esthetic properties and behavior properties.

Esthetic Properties

In the esthetic properties panel, the following are grouped:

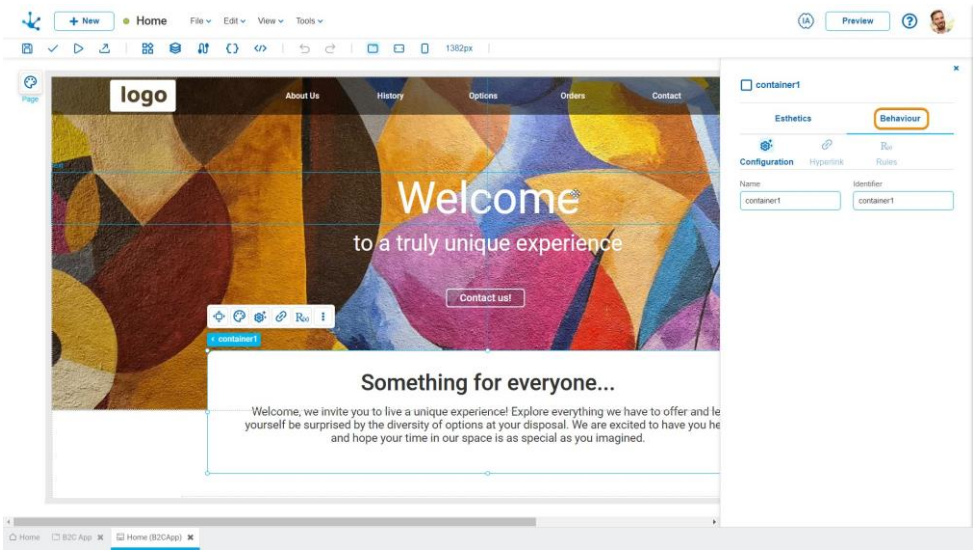
- Structure Properties
- Style Properties



Behavior Properties

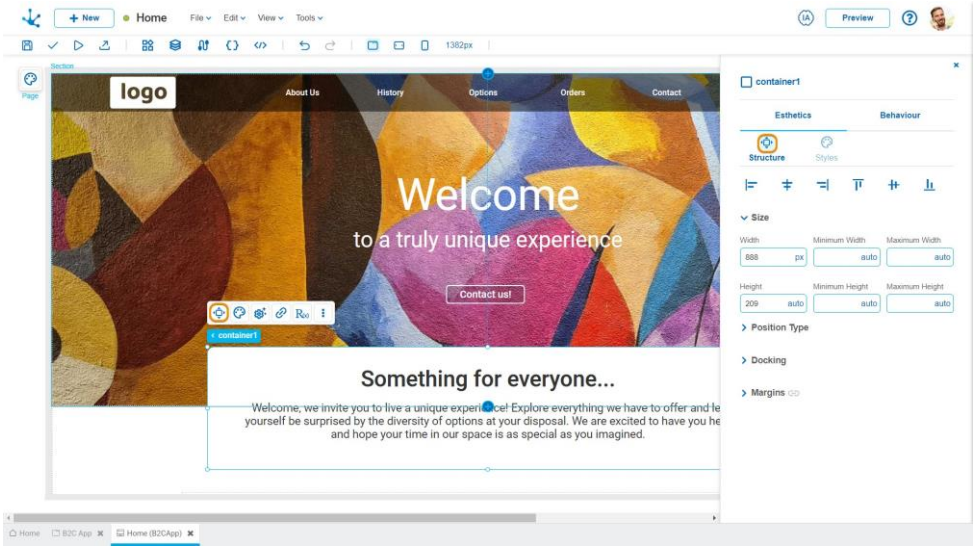
In the behavior properties panel, the following are grouped:

- Configuration Properties
- Hyperlink Properties
- Rules Properties



Structure Properties

The structure properties panel of an element opens when selecting the icon  of its context menu.



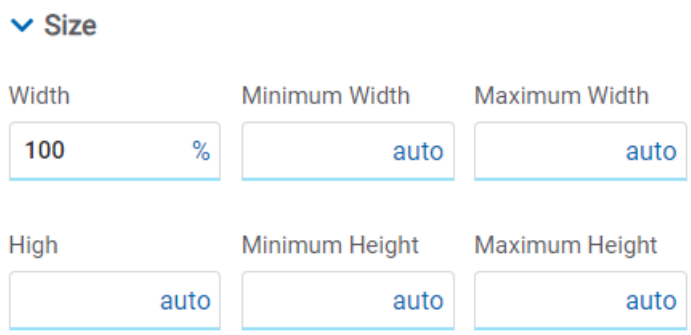
Alignment

Allows to align the element by selecting one of the available icons.

-  Align to the left.
-  Align to horizontal center.
-  Align to the right.
-  Align up.

- Align vertical center.
- Align bottom.

Size

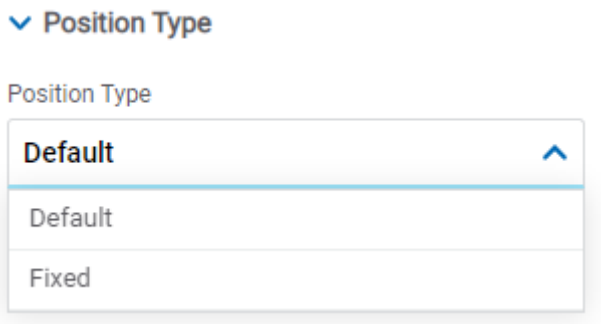


All size properties can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh). Additionally, for Width and Height properties, the "auto" option is added.

> *If selecting percentage (%) for the width and height properties, the size is calculated relative to the top element.*

Position Type

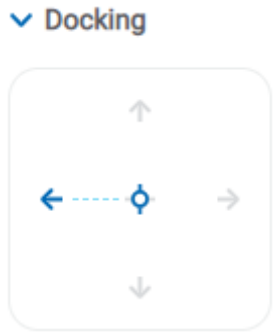
It determines if at the time of execution the element remains fixed on the page or if it moves as the scroll bar moves forward or backward.



Possible Values

- Default: The element has a relative position with respect to the superior element where it was placed (container or section).
- Fixed: Fixes the element to the page so that it is always visible in the same position. It allows to increase the visibility of important content.

Docking



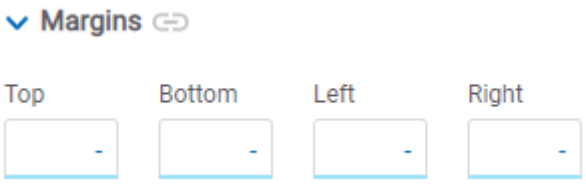
Determines the vertical and horizontal position of elements within the page section or container, when the screen is resized.

When an element is added or moved within the section or container, the modeler automatically docks it to the nearest corners or edges. It can also be configured by clicking on the arrows of the side to be docked to or on the center.

Once the element is docked, its exact position is indicated in the Margins property. The direction of the docking arrows determine the values that are automatically displayed for margins.

The docking position is indicated in the modeling area by dotted lines on the element.

Margins



It allows to define the distance of elements from the borders of their top element. The behavior of margins depends on the docking of the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Top
Distance to the top border of the highest ranking element.

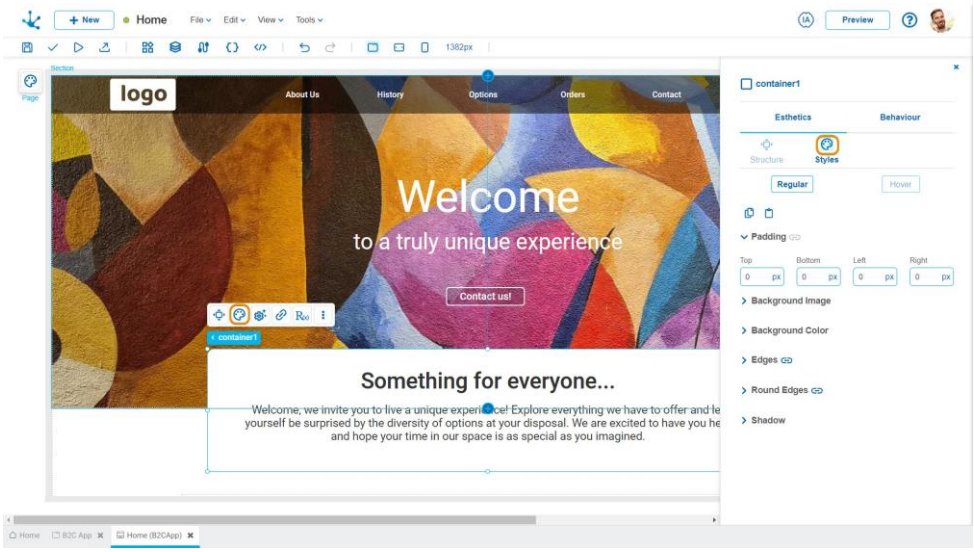
Bottom
Distance to the bottom border of the highest ranking element.

Left
Distance to the left border of the highest ranking element.

Right
Distance to the right border of the highest ranking element.

Allows the value entered in one of the margins to be copied to the other ones automatically.
Allows to indicate different values for each margin.

The style properties panel of an element opens when selecting the icon of its context menu.



This type of element may take different states and for each of them different values for its properties may be modeled.



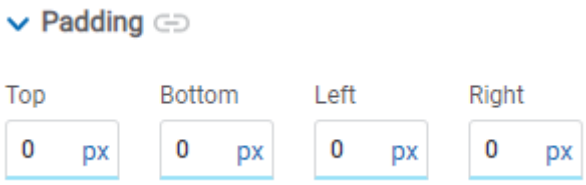
- Regular: The mouse pointer is not over the element.
- Hover: The mouse pointer is over the element.

Styles

It allows copying the entire current configuration of the style panel for the selected element (Ctrl+Alt+C).

It allows applying the previously copied configuration to another element of the same type (Ctrl+Alt+V).

Padding

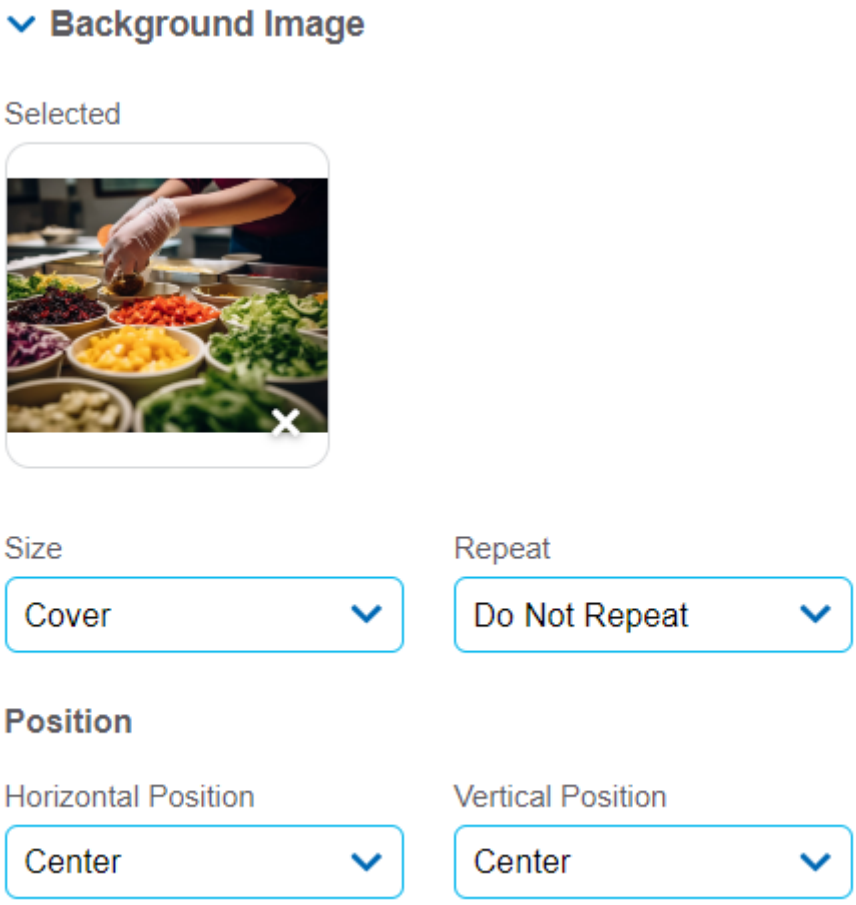


All padding space properties allow to create a space around the element borders (top, bottom, sides) and its content. The padding space configured is inward, while the margin is configured relative to a top element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Allows values entered in one of the paddings to be copied to the other ones automatically.
Allows to indicate different values for each padding.

Background Image

The element properties are represented by icons on its context menu, where its operations are also available.
PropsEstiloImagenFondo3

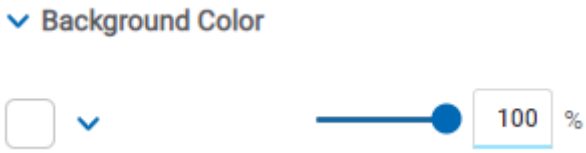


It allows to add a background image to the section.

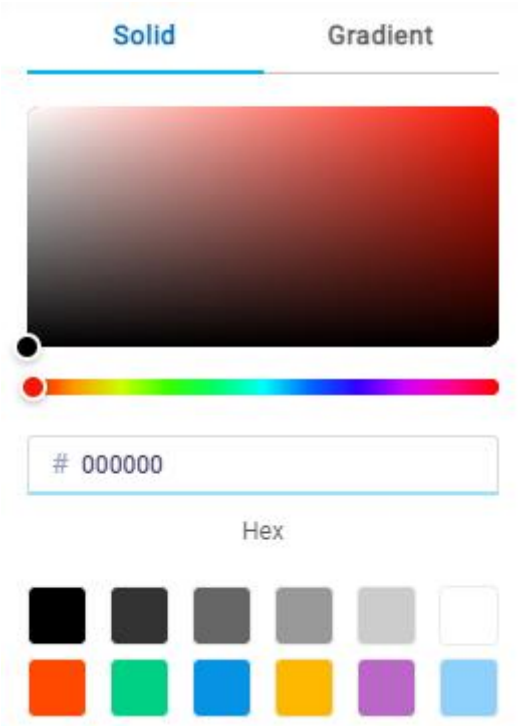
Background Color

It allows to model a background color for the element, it can be solid or gradient and different properties are defined for each one.

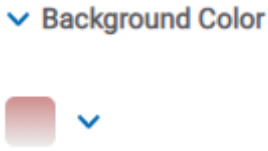
Solid



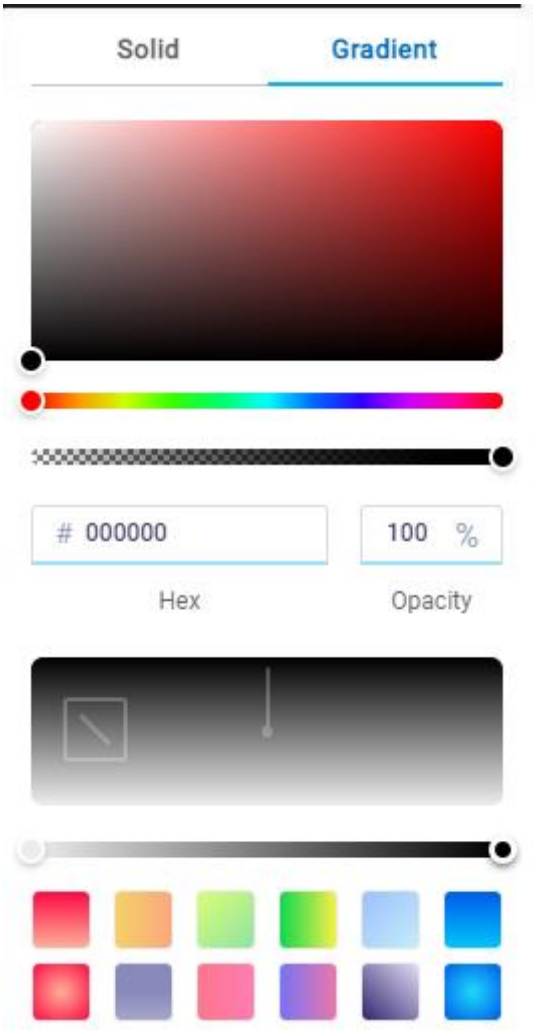
This option enables not only to select the color from a palette but also to define the degree of transparency.



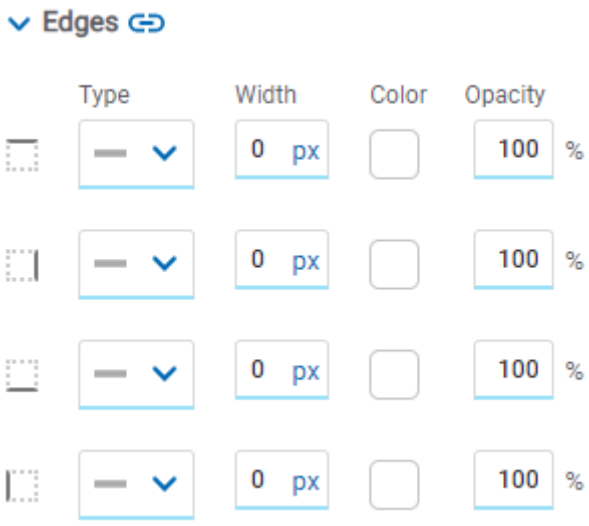
Gradient



This option enables to select not only the color from a palette but also to define its opacity and intensity according to the selected angle.



Edges



It allows to define the style of borders. Each one has its type, width, color and opacity defined, the latter as a percentage.

Allows the value entered in one of the borders to be copied to the other ones automatically.

Allows to indicate different values for each border.

Round Edges

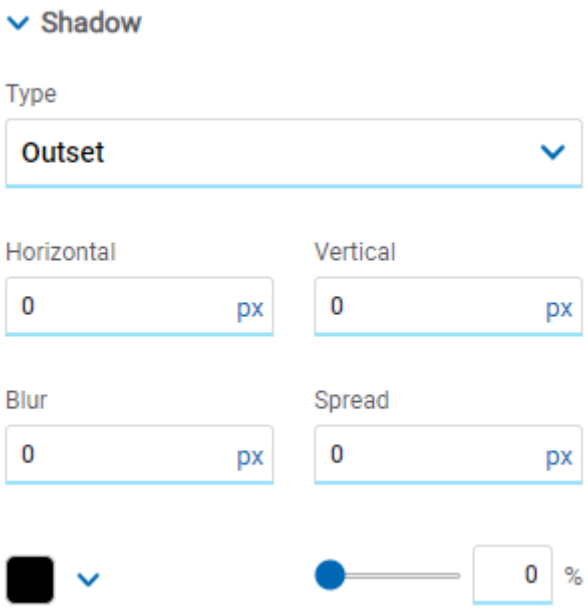


It allows to define the round edges at the corners of elements.
It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Allows the value entered in one of the borders to be copied to the other ones automatically.

Allows to indicate different values for each border.

Shadow



Allows to define a shadow effect around the element.

Horizontal
Horizontal size of the shadow to the right of the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Vertical
Vertical size of the shadow below the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

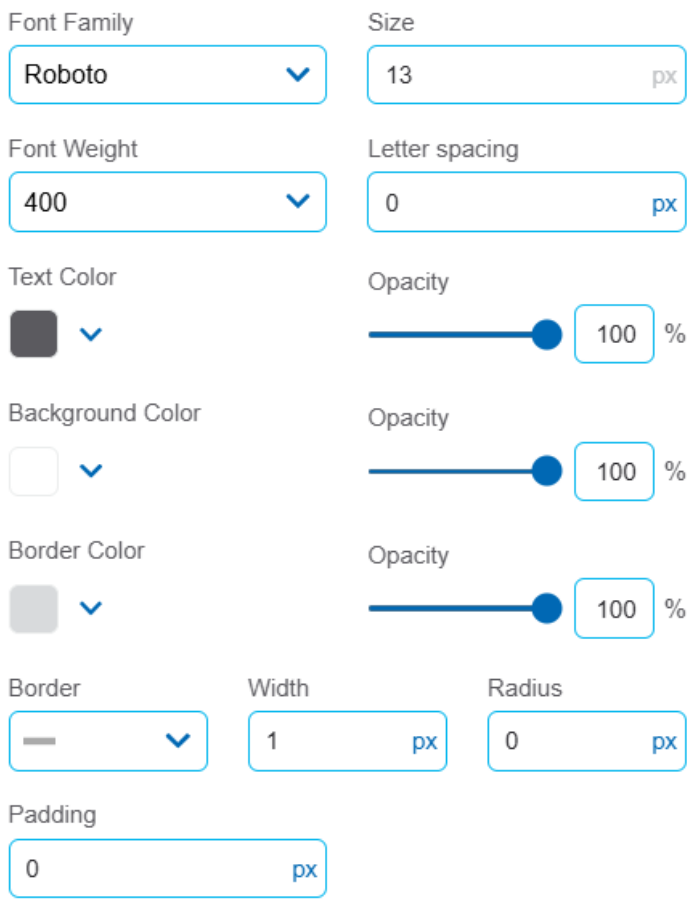
Blur
The greater this value is, the greater and lighter the shadow becomes. If not specified, its value is 0 and the shadow border is darker. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Spread
Positive values will cause the shadow to expand and grow bigger, while negative values will cause the shadow to shrink. If not specified, its value is 0 and the shadow will have the same size as the element.

Color
Defines the color of the shadow.

Transparency
Defines the transparency of the shadow.

Tooltip



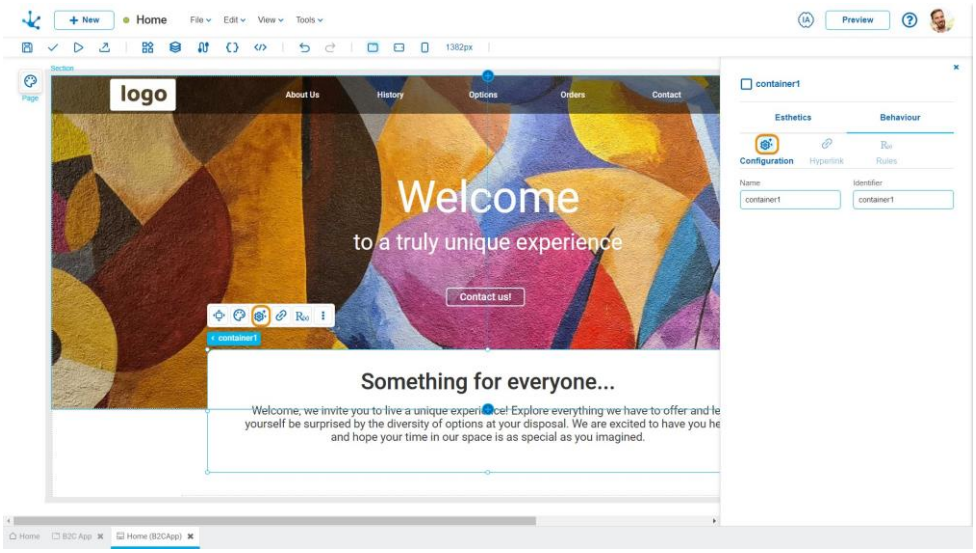
Source
It allows customizing the font, size, thickness and letter spacing of the pop-up message.

Colors
The color and opacity of the text, background and border can be defined. Borders are set uniformly for all sides of the tooltip.

Border
The border type, width, and border radius can be configured.

Configuration Properties

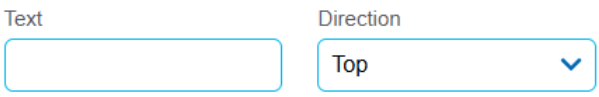
The configuration properties panel of an element opens when clicking the icon  of its context menu.



Name
Name used to reference the element during modeling.

Identifier
Uniquely identifies the element. It is used in the Javascript SDK.

Tooltip
Pop-up message that appears when hovering over an element, providing additional information.

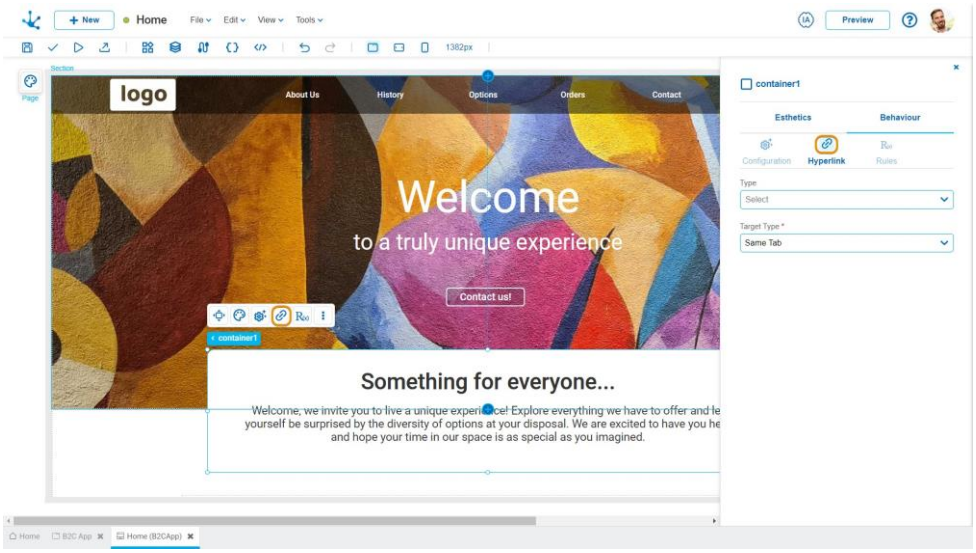


Text
Content of the message being displayed.

Direction
Position of the message in relation to the element (top, bottom, left, or right).

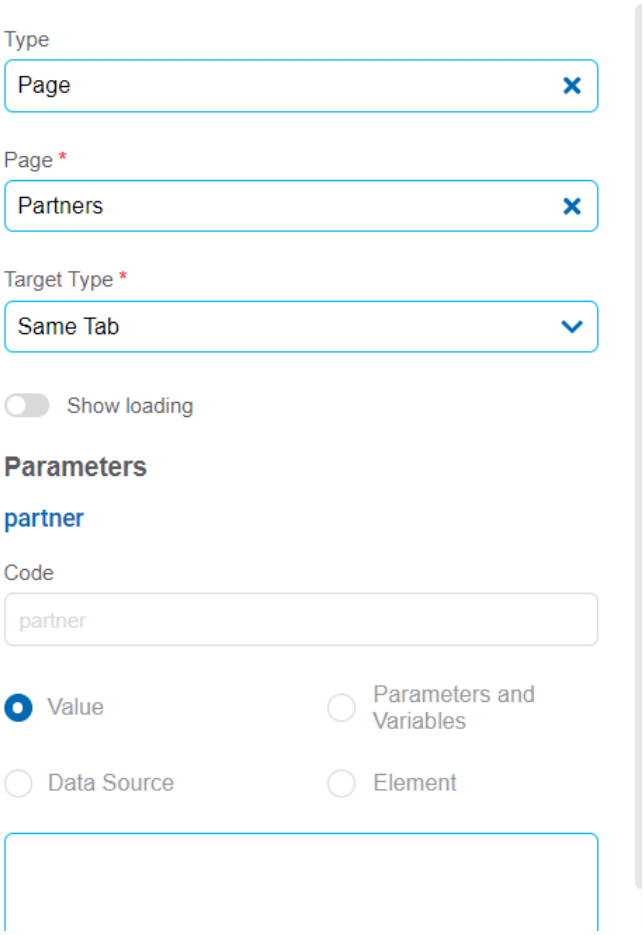
Hyperlink Properties

The hyperlink properties panel of an element opens when clicking the icon  of its context menu.



Define the behavior of the element when selecting it. Different properties are enabled depending on the type of object selected.

Page



Page
The pages modeled in the environment are displayed.

Destination Type
The available options for opening the object are displayed.

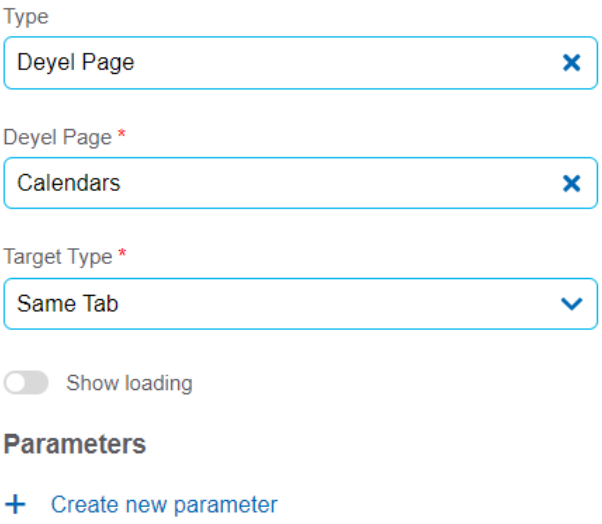
Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  - Modal Horizontal Size
    Defines its width.
  - Modal Vertical Size
    Defines its height.
- Iframe: selecting this option enables an additional property.
  - Destination Iframe
    Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Parameters
If there are parameters, their value can be specified with text, parameters and variables or data source.

Deyel Page



Deyel Page

The pages belonging to **Deyel** are displayed.

Destination Type
The available options for opening the object are displayed.

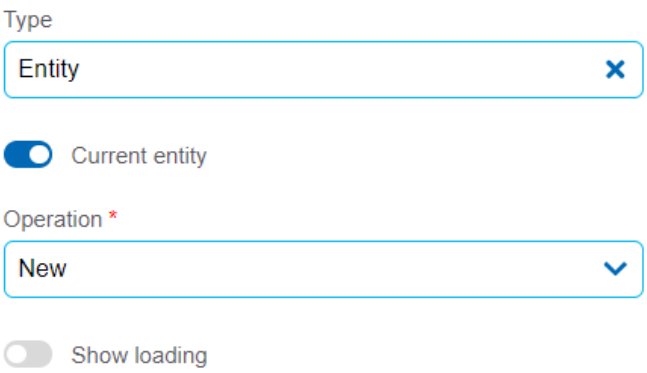
Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  Modal Horizontal Size
  Defines its width.
  Modal Vertical Size
  Defines its height.
- Iframe: selecting this option enables an additional property.
  Destination Iframe
  Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Parameters
Allows sending parameters to the selected object type.

Entity

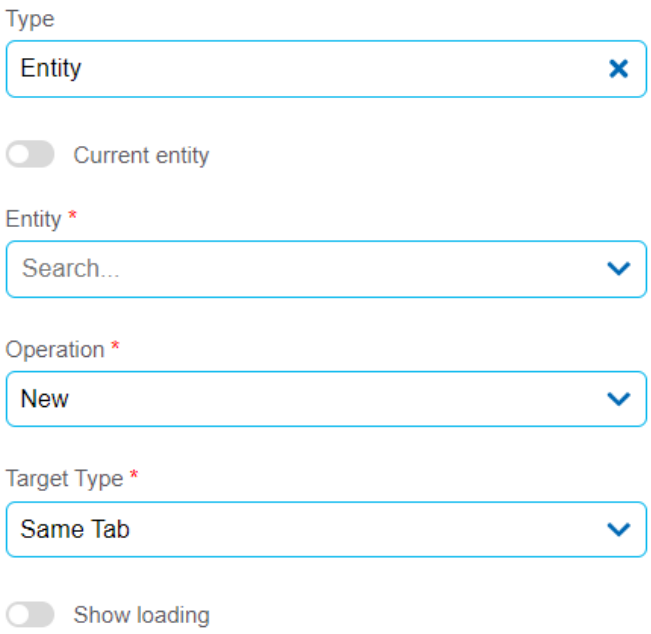


Current Entity
If this property is checked, the operation applies directly to the entity being modeled.

Operation
Defines the operation made when selecting the object.

Possible Values

- New: Indicates that the current entity panel is opened for the creation of an instance.
- Modify: Indicates that the current entity panel is opened for the update of an instance.
- Delete: Indicates that the current entity panel is opened for the deletion of an instance.



Entity
If the property Current Entity, is not checked, the entities modeled in the environment are displayed.

Operation
Defines the operation made when selecting the object.

Possible Values

- New: The panel of the entity selected in the previous property is opened to create an instance.
- Update The panel of the entity selected in the previous property is opened to update an instance.
- Show: The panel of the entity selected in the previous property is opened to show an instance.
- Grid: Indicates that the grid of the entity selected in the previous property is opened.

Destination Type
The available options for opening the object are displayed.

Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  Modal Horizontal Size
  Defines its width.
  Modal Vertical Size
  Defines its height.
- Iframe: selecting this option enables an additional property.

Destination Iframe

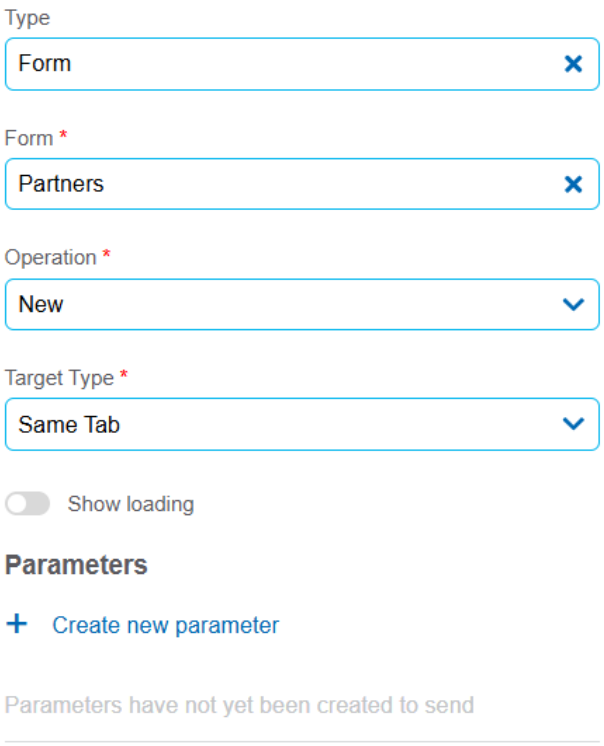
Expands the iframes previously defined on the object.

- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading

Shows the loading icon and disables the user interaction until the redirection is over.

Forms



Forms

The forms modeled in the environment are displayed.

Operation

Defines the operation made when selecting the object.

Possible Values

- Create: Indicates that the panel of the object selected in the previous property is opened for the creation of an instance.
- Update: Indicates that the panel of the object selected in the previous property is opened for the update of an instance.
- Show: Indicates that the panel of the object selected in the previous property is opened for the query of an instance.
- Grid: Indicates that the grid of the object selected in the previous property is opened.

Destination Type

The available options for opening the object are displayed.

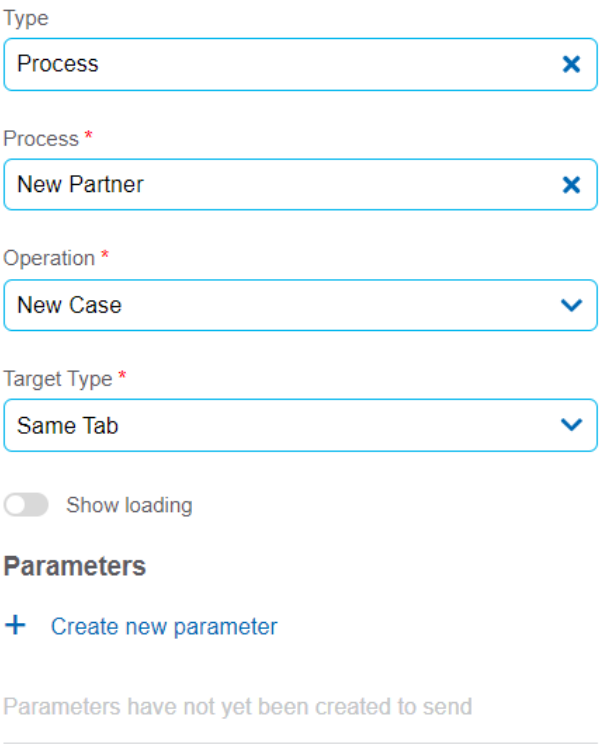
Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  Modal Horizontal Size
  Defines its width.
  Modal Vertical Size
  Defines its height.
- Iframe: selecting this option enables an additional property.
  Destination Iframe
  Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Parameters
Allows sending parameters to the selected object type.

Process



Process

The processes modeled in the environment are displayed.

Operation
Defines the operation made when selecting the object.

Destination Type
The available options for opening the object are displayed.

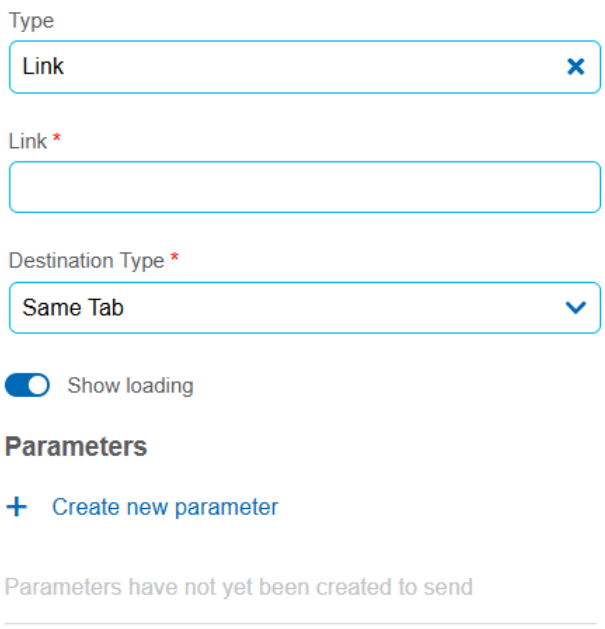
Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  - Modal Horizontal Size
    Defines its width.
  - Modal Vertical Size
    Defines its height.
- Iframe: selecting this option enables an additional property.
  - Destination Iframe
    Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Parameters
Allows sending parameters to the selected object type.

Link



Link
Allows to enter any link.

Destination Type
The available options for opening the object are displayed.

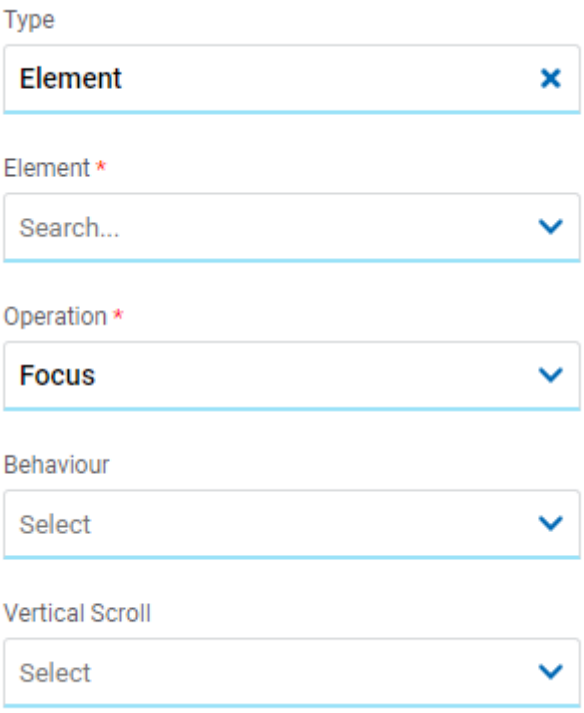
Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  - Modal Horizontal Size
    Defines its width.
  - Modal Vertical Size
    Defines its height.
- Iframe: selecting this option enables an additional property.
  - Destination Iframe
    Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Parameters
Allows sending parameters to the selected object type.

Element



Element
The modeled elements in the object are displayed.

Operation
Defines the operation made when selecting the element.

Possible Values

- Focus
- Show
- Hide
- Show/Hide

Behaviour
Establishes the transition of the animation. It is only modelable if the "Focus" operation was selected.

Possible Values

- Auto
- Smooth

Vertical Scroll
It is only modelable if the "Focus" operation was selected.
Possible Values

- Start: Moves to the start of the selected element.
- Closest: Moves to the closest position to the selected element from the element the event fires.
- Center: Moves to the center of the selected element.
- End: Moves to the end of the selected element.

Repeater



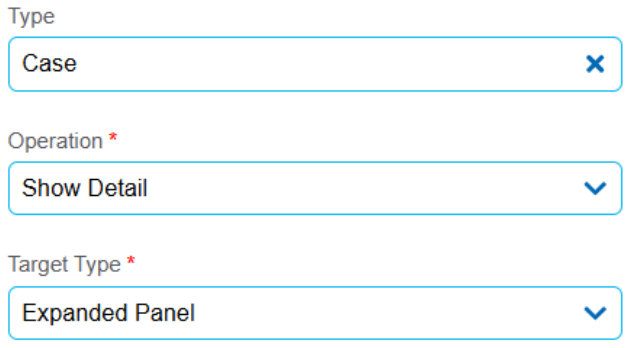
Repeater
The modeled repeaters from the object are displayed.

Operation
Defines the operation made when selecting the element.

Possible Values

- New: Allows for the creation of an item within the repeater. If this option is selected the additional properties are enabled.
  Creation Order
  It indicates in which position the new item of the container is added.
  Possible Values
  - Home
  - End
- Delete: Allows for the deletion of an item within the repeater.

Case



Case
Show case information.

Operation

Defines the operation made when selecting the element.

Possible Values

- Show Detail
- Show Graphic Execution

Destination Type
The available options for opening the object are displayed.

Possible Values

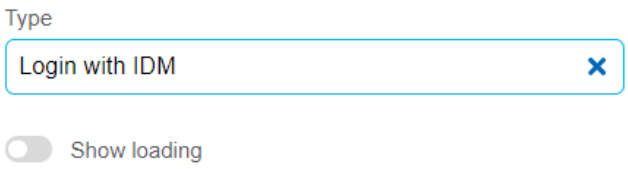
- Expanded Panel
- Modal: if this option is selected the additional properties are enabled.
  Modal Horizontal Size
  Define its width.
  Modal Vertical Size
  Define its height.

Back



It allows associating the event to go back in the browser to the element.

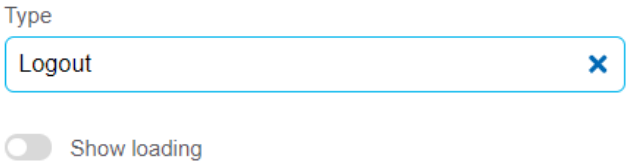
LogIn with IDM



Allows login with IDM.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

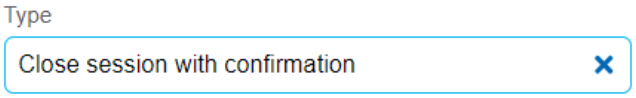
Logout



Allows logging the user out.

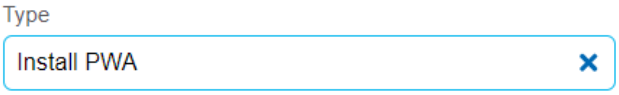
Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Close Session with Confirmation



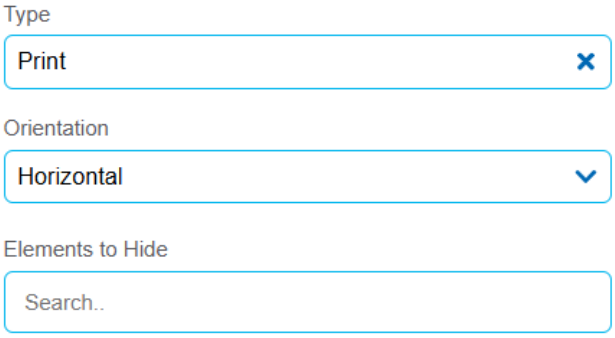
Displays a confirmation modal to logout the user.

Install PWA



Allows to install the application in the browser.

Print



This type of hyperlink associated with an element allows printing the content of the entity or page that is being executed.

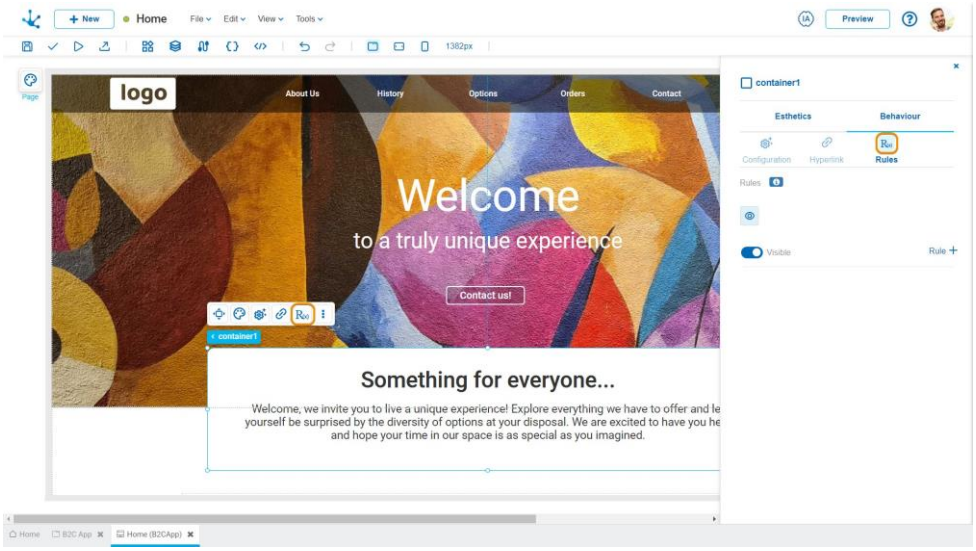
Orientation
Determines the layout of the content to be printed.

Possible Values
- Horizontal
- Vertical

Elements to hide
It allows selecting which elements of the entity or page should not be included in the printout.

Rule Properties

The rule properties panel of an element opens when clicking the icon  of its context menu.



Properties

Rules
Embedded rules on behavior, validation and calculation can be defined, associated with an element, by using the wizard (ctrl + space).

Shows syntax examples for writing the rules.

Visible

Indicates whether the element is visible. If this property is not checked, the element is not displayed in the page.

Visible (default) Not visible

Rule + Opens an edit area where a rule to determine the visibility condition can be defined. If a rule is defined, the icon is displayed with light blue borders.

Operations when defining the rule:

Saves the new or modified rule

Cancels the operation

Operations once the rule is defined:

- Edits the existing rule
- Deletes the rule

Events

Containers allow the use of different events.

Event	Description
onMouseIn()	It is executed when the cursor is positioned on the element.
onMouseOut()	It is executed when the cursor moves out of the element.
onClick()	It is executed when clicking on the element.
onDoubleClick()	It is executed when clicking twice on the element.
onInit()	It is executed before the element is loaded.
afterViewInit()	It is executed after viewing the element.
onViewportEnter()	It is executed when the element is visible.
onViewportLeave()	It is executed when the element is no longer visible.

Iframe

This element is used to show objects **Deyel** or external web pages. For example, it can contain forms, pages, cases, etc. Its position and size must be selected so that it responds to different display windows.

The element properties are represented by icons in its context menu, where its operations are also available.

Subtypes

By selecting the "Iframe" option from the icon on the top toolbar, a list with the different subtypes of this element is displayed, which can be dragged to the modeling area. Each subtype has modeled element properties, predefined in a specific way.

- Iframe with Scroll

- Adaptable Iframe

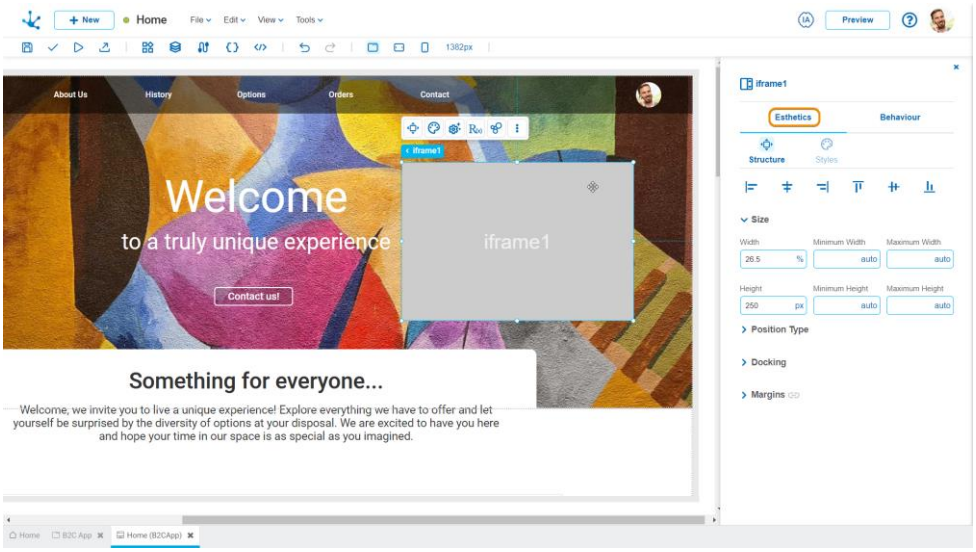
Classification of Properties

Properties are divided into two groups: esthetic properties and behavior properties.

Esthetic Properties

In the esthetic properties panel, the following are grouped:

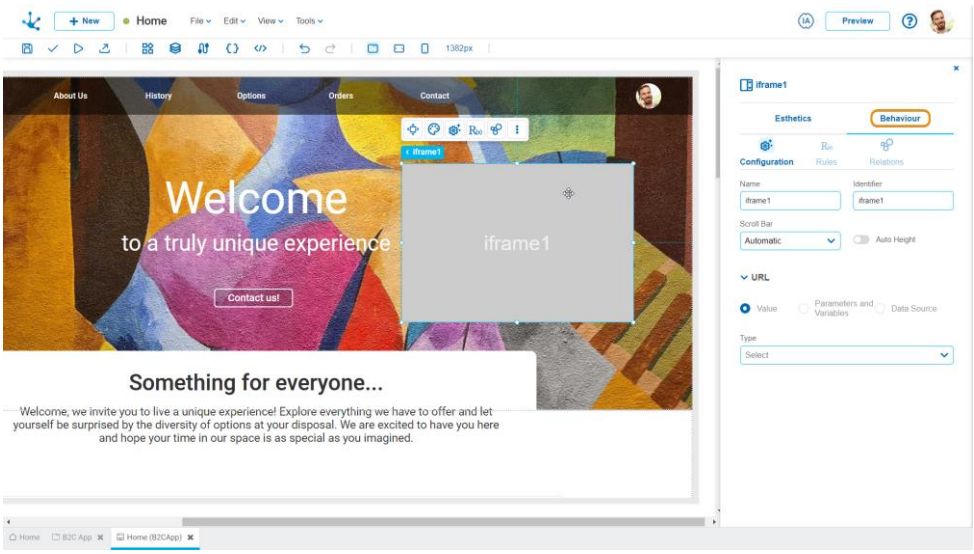
- Structure Properties
- Style Properties



Behavior Properties

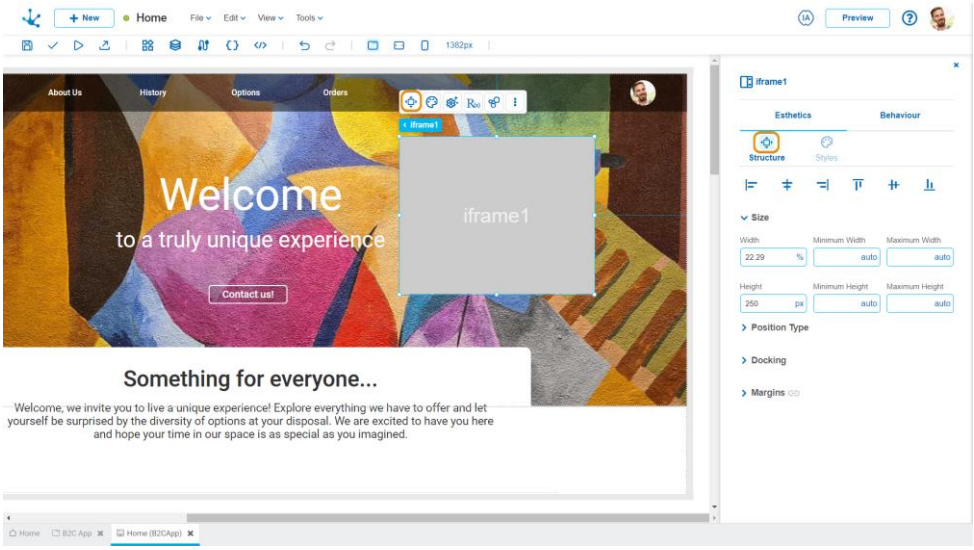
In the behavior properties panel, the following are grouped:

- Configuration Properties
- Rules Properties
- Relations Properties



Structure Properties

The structure properties panel of an element opens when selecting the icon  of its context menu.



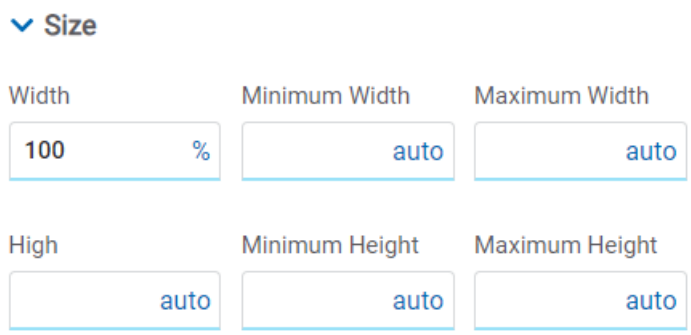
Alignment

Allows to align the element by selecting one of the available icons.

- Align to the left.
- Align to horizontal center.
- Align to the right.
- Align up.

- Align vertical center.
- Align bottom.

Size

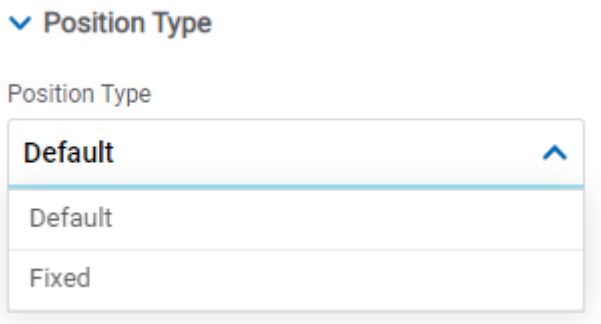


All size properties can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh). Additionally, for Width and Height properties, the "auto" option is added.

> *If selecting percentage (%) for the width and height properties, the size is calculated relative to the top element.*

Position Type

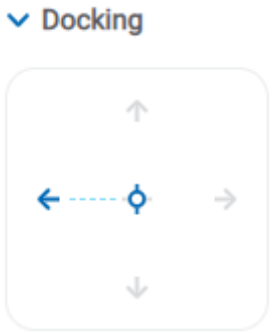
It determines if at the time of execution the element remains fixed on the page or if it moves as the scroll bar moves forward or backward.



Possible Values

- Default: The element has a relative position with respect to the superior element where it was placed (container or section).
- Fixed: Fixes the element to the page so that it is always visible in the same position. It allows to increase the visibility of important content.

Docking



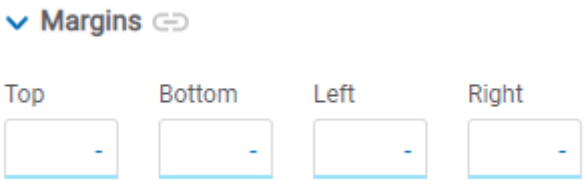
Determines the vertical and horizontal position of elements within the page section or container, when the screen is resized.

When an element is added or moved within the section or container, the modeler automatically docks it to the nearest corners or edges. It can also be configured by clicking on the arrows of the side to be docked to or on the center.

Once the element is docked, its exact position is indicated in the Margins property. The direction of the docking arrows determine the values that are automatically displayed for margins.

The docking position is indicated in the modeling area by dotted lines on the element.

Margins



It allows to define the distance of elements from the borders of their top element. The behavior of margins depends on the docking of the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Top
Distance to the top border of the highest ranking element.

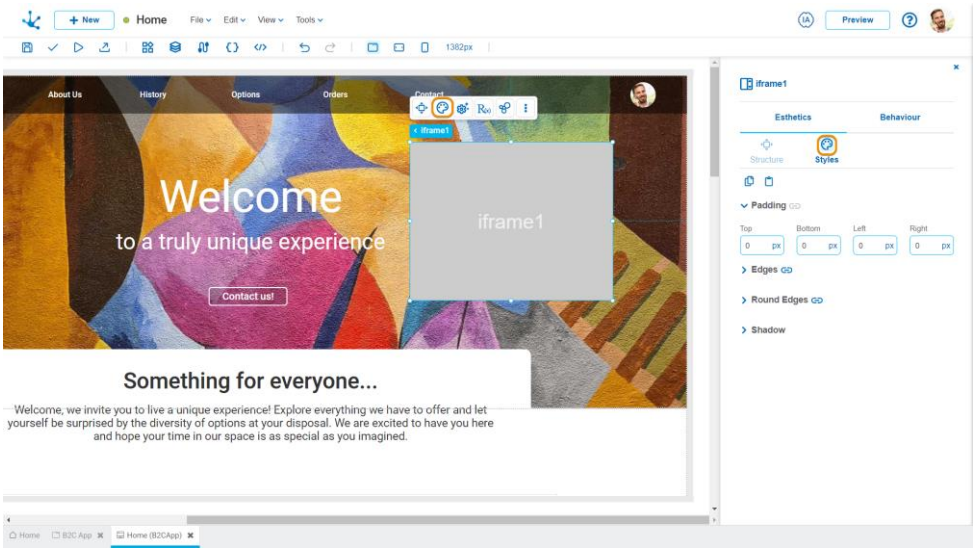
Bottom
Distance to the bottom border of the highest ranking element.

Left
Distance to the left border of the highest ranking element.

Right
Distance to the right border of the highest ranking element.

Allows the value entered in one of the margins to be copied to the other ones automatically.
Allows to indicate different values for each margin.

The style properties panel of an element opens when selecting the icon of its context menu.

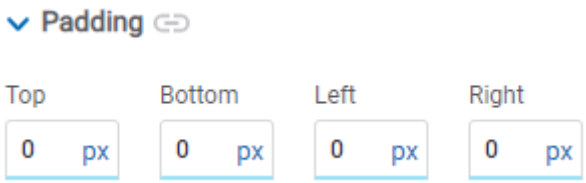


Styles

It allows copying the entire current configuration of the style panel for the selected element (Ctrl+Alt+C).

It allows applying the previously copied configuration to another element of the same type (Ctrl+Alt+V).

Padding

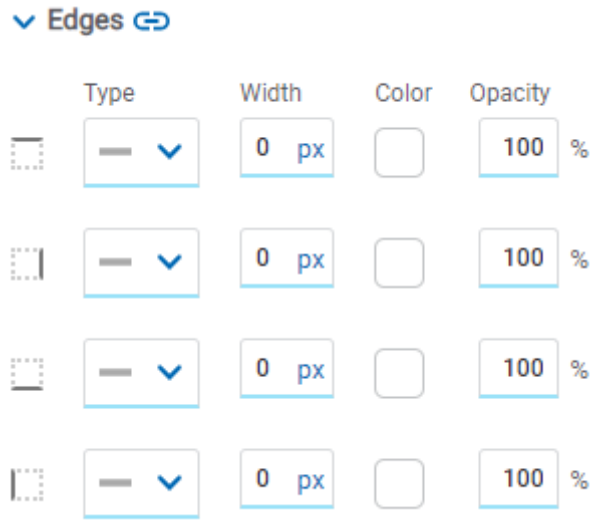


All padding space properties allow to create a space around the element borders (top, bottom, sides) and its content. The padding space configured is inward, while the margin is configured relative to a top element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Allows values entered in one of the paddings to be copied to the other ones automatically.

Allows to indicate different values for each padding.

Edges



It allows to define the style of borders. Each one has its type, width, color and opacity defined, the latter as a percentage.

Allows the value entered in one of the borders to be copied to the other ones automatically.
Allows to indicate different values for each border.

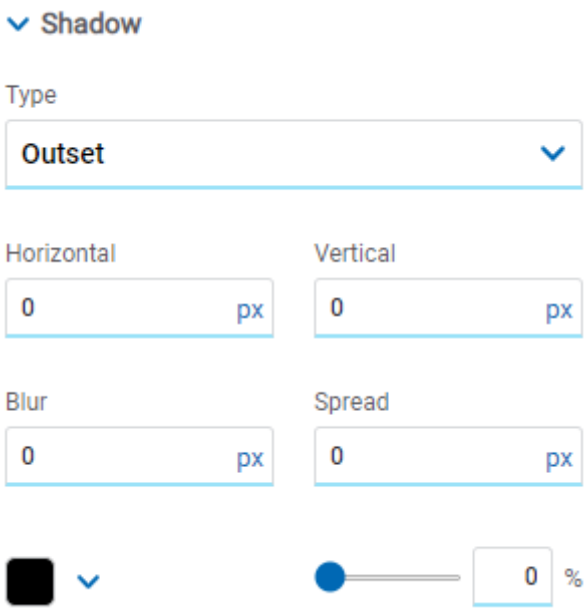
Round Edges



It allows to define the round edges at the corners of elements.
It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Allows the value entered in one of the borders to be copied to the other ones automatically.
Allows to indicate different values for each border.

Shadow



Allows to define a shadow effect around the element.

Type

Possible Values

- Outset
- Inset

Horizontal
Horizontal size of the shadow to the right of the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Vertical
Vertical size of the shadow below the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

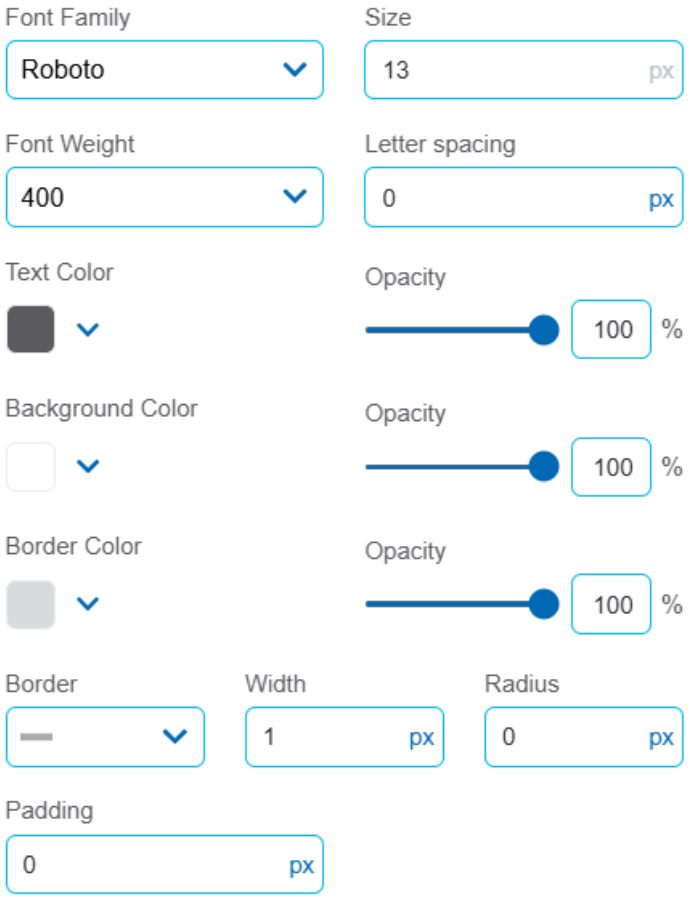
Blur
The greater this value is, the greater and lighter the shadow becomes. If not specified, its value is 0 and the shadow border is darker. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Spread
Positive values will cause the shadow to expand and grow bigger, while negative values will cause the shadow to shrink. If not specified, its value is 0 and the shadow will have the same size as the element.

Color
Defines the color of the shadow.

Transparency
Defines the transparency of the shadow.

Tooltip



Source

It allows customizing the font, size, thickness and letter spacing of the pop-up message.

Colors

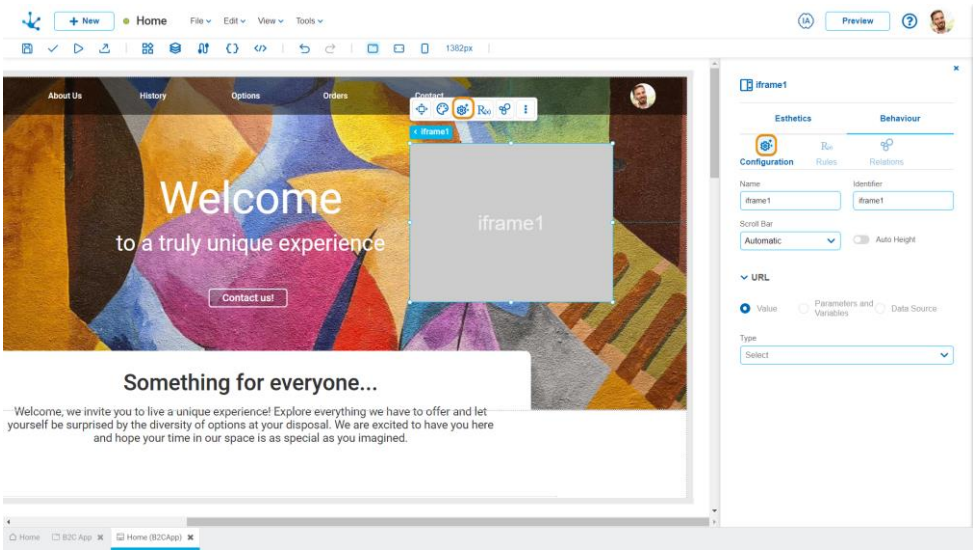
The color and opacity of the text, background and border can be defined. Borders are set uniformly for all sides of the tooltip.

Border

The border type, width, and border radius can be configured.

Configuration Properties

The configuration properties panel of an element opens when clicking the icon  of its context menu.



Properties



Name
Name used to reference the element during modeling.

Identifier
Uniquely identifies the element. It is used in the Javascript SDK.

Tooltip
Pop-up message that appears when hovering over an element, providing additional information.



Text
Content of the message being displayed.

Direction
Position of the message in relation to the element (top, bottom, left, or right).

Scroll



Auto Height

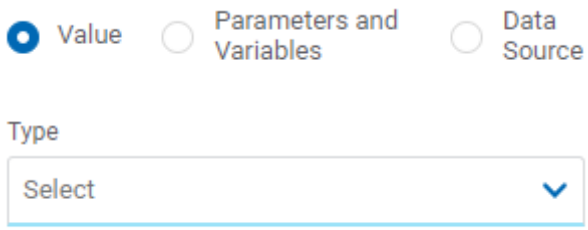
It allows the height of the iframe to adjust to the size of its content in case the latter exceeds the size of the iframe. If this property is not checked, the different values of the Scroll Bar property can be selected.

Scroll Bar
Indicates whether the scroll on the element is displayed: always, never, or according to the size of the iframe content.

URL

Allows to select the source of the element's content.

Value



- Page
  Page
  It allows selecting the external page that is displayed in the iframe. Once it is selected, its parameters are displayed if it has any.

- Deyel Page
  Deyel Page
  It allows selecting the **Deyel** page displayed in the iframe. Once it is selected, its parameters are displayed if available, or else a parameter can be added.

- Entity
  Entity
  It allows selecting a name for the entity displayed on the iframe. Once selected, its parameters are displayed if available, or else a parameter can be added.

- Form
  Form
  It allows selecting a name for the form displayed on the iframe. Once selected, its parameters are displayed if available, or else a parameter can be added.

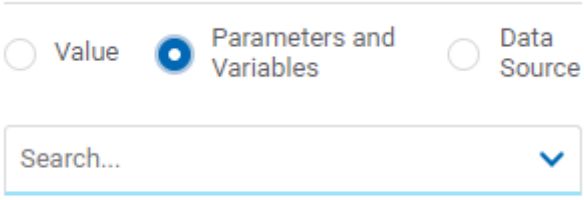
  Operation
  It allows selecting whether to view the creation of an instance or the form grid.

- Process
  Process
  It allows selecting the name of the process that is executed in the iframe. Once selected, its parameters are displayed if available, or else a parameter can be added.

Operation
Indicates that the creation of a case is displayed.

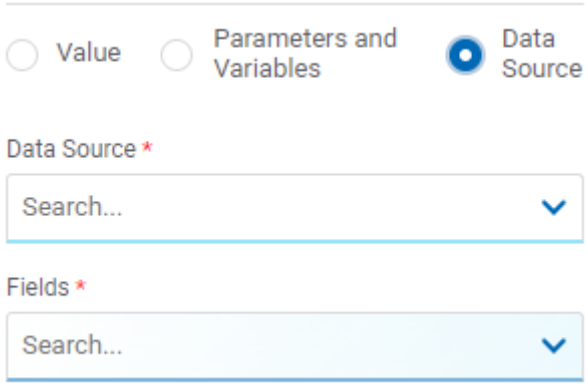
- Link
  Link
  It allows the entry of the link to the page that executes in the iframe.

Parameters and Variables



Parameters and Variables
Allows to select a parameter or a variable from, whose value is displayed in the element.

Data Source

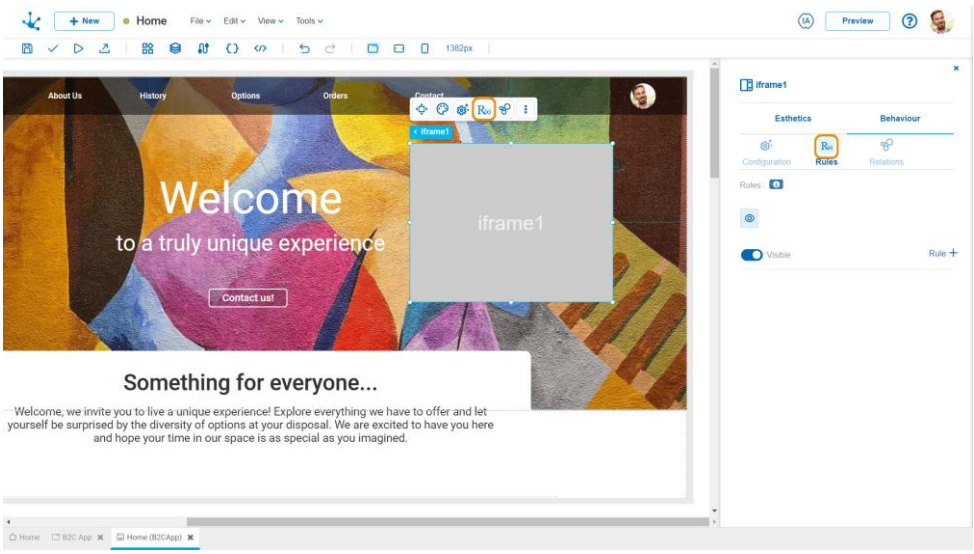


Data Source
It allows selecting a previously defined data source within the object.

Fields
It allows selecting a field from the data retrieved in the chosen data source, whose value is displayed in the element.

Rule Properties

The rule properties panel of an element opens when clicking the icon  of its context menu.



Properties

Rules

Embedded rules on behavior, validation and calculation can be defined, associated with an element, by using the wizard (ctrl + space).

Shows syntax examples for writing the rules.

Visible

Indicates whether the element is visible. If this property is not checked, the element is not displayed in the page.

Visible (default) Not visible

Rule + Opens an edit area where a rule to determine the visibility condition can be defined. If a rule is defined, the icon is displayed with light blue borders.

Operations when defining the rule:

Saves the new or modified rule

Cancels the operation

Operations once the rule is defined:

Edits the existing rule

Deletes the rule

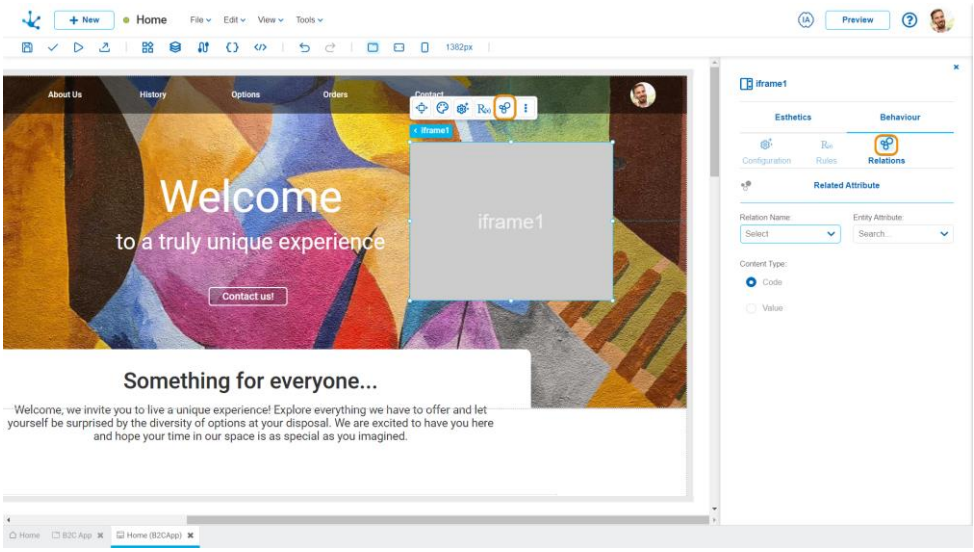
Events

Iframes allow the use of an event.

Event	Description
onLoad()	It is executed once all elements of the iframe are loaded.

Relation Properties

The relation properties panel of an element opens when clicking the icon of its context menu.



Related Attribute

A related attribute indicates that the field value is retrieved from the attribute value of another entity. One of the relations defined on the page must be selected, and then it should be specified which attribute of the related entity it links to.

A field can be modeled as a related attribute and it can also have a relation defined to an entity, a value list, or a rule. This allows the value of the related attribute to be retrieved on this field and the functionality provided by the relation to be used.

Properties

Relation Name
The list of relations defined on the page is displayed, allowing the selection of one of them.

Entity Attribute
The list of attributes of the selected page compatible with the field type is displayed. Attributes can have a relation modeled with an entity, a value list, or a rule in the selected entity.

Content Type
The element content varies depending on whether the Code or Value property is checked when:

- The entity attribute has a modeled relation.

- The field has the option "Entity" selected in the Values Obtained from property.

Code

When this property is checked, if the relation is modeled with a:

- Related entity, the identifier of the entity is displayed.
- Value list, the value list code is displayed.
- Rule, the code returned by the rule's execution is displayed.

Value

When this property is checked, if the relation is modeled with a:

- Related entity, the short description of the entity is displayed.
- Value list, the value list description is displayed.
- Rule, the description returned by the rule's execution is displayed.

Fields

This element is used for data entry.

The element properties are represented by icons in its context menu, where its operations are also available.

Subtypes

When selecting the option "Field" from the icon in the top toolbar, a list opens with the different subtypes of this element, which can be dragged and dropped into the modeling area.

- Text
- Number
- Multiline
- Date and Time
- Options Group
- Toggle
- File

The subtype of a field can be selected in the property Content Format.

File Subtype

It is possible to attach files to fields both from the computer and from Google Drive. This functionality is automatically available when configuring and publishing the Google Drive adapter, without requiring changes to the modeling.

When executing the entity, a file can be added by clicking on the modeled field.

The following options are available:

- From the computer: allows selecting a local file.
- From Google Drive: a window will open to log in to Google (if necessary) and then select the file from a Google account.

Once selected and confirmed, the file is displayed in the field.

Whenever a file is uploaded, the icon is displayed to the right of the field, sometimes automatically and other times when hovering the cursor over it.

When clicked, the following actions are displayed:

- Preview
- Download
- Delete

When hovering the cursor over the file, the preview option is displayed directly, regardless of the field configuration.

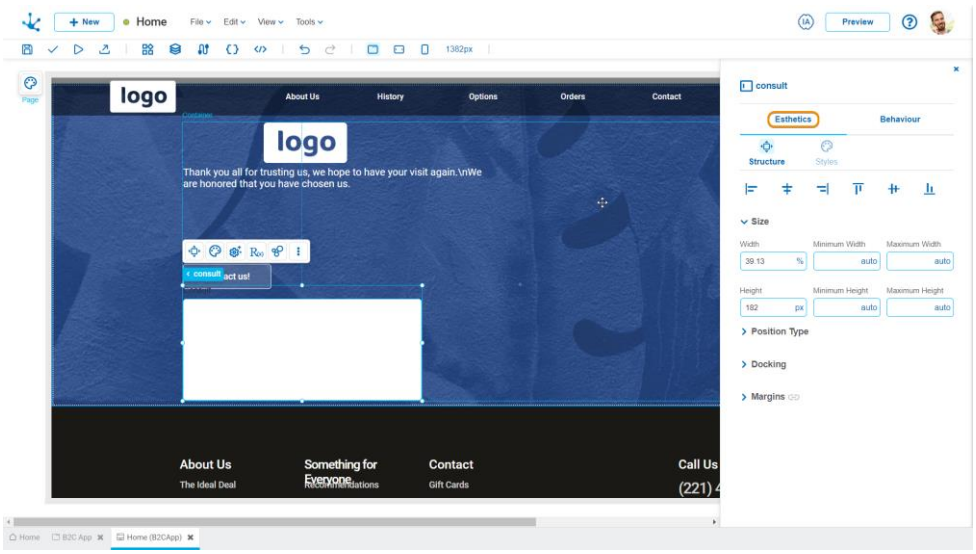
Classification of Properties

Properties are divided into two groups: esthetic properties and behavior properties.

Esthetic Properties

In the esthetic properties panel, the following are grouped:

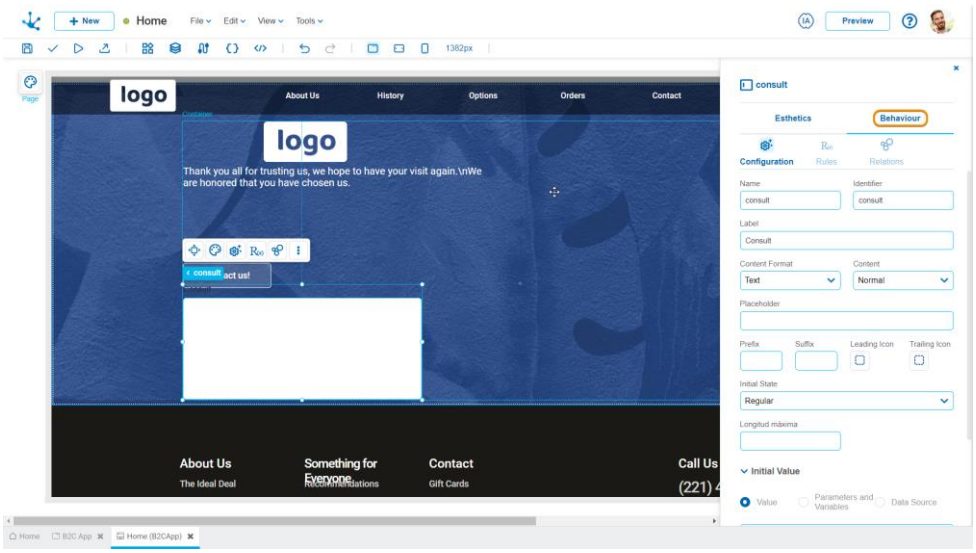
- Structure Properties
- Style Properties



Behavior Properties

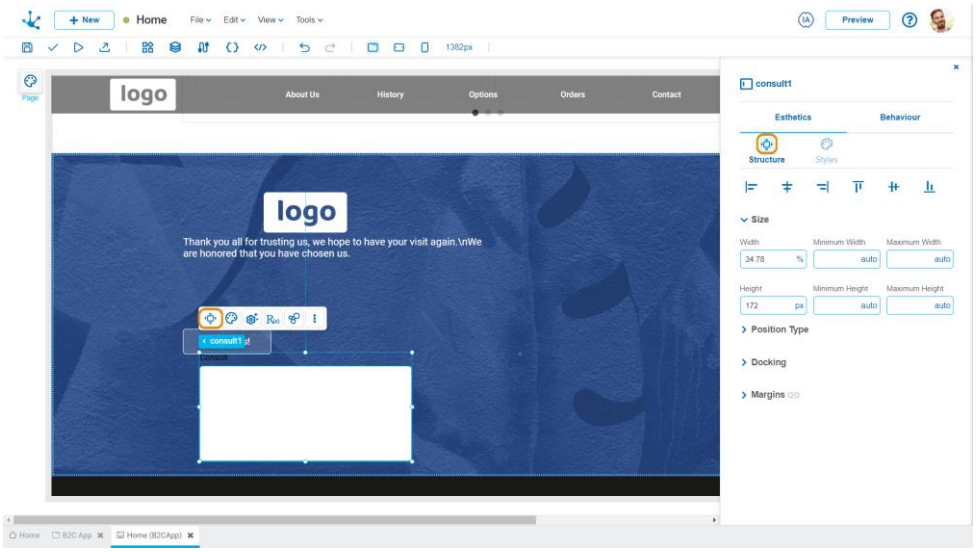
In the behavior properties panel, the following are grouped:

- Configuration Properties
- Rule Properties
- Relation Properties



Structure Properties

The structure properties panel of an element opens when selecting the icon  of its context menu.



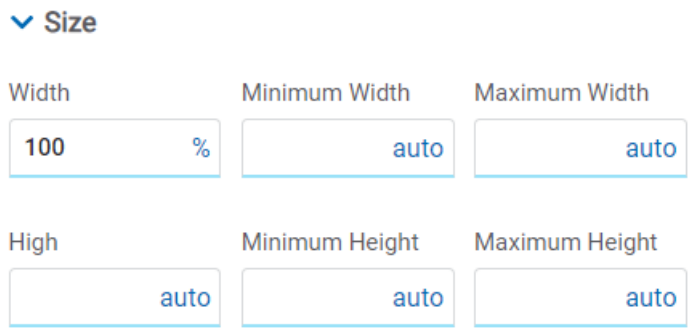
Alignment

Allows to align the element by selecting one of the available icons.

-  Align to the left.
-  Align to horizontal center.
-  Align to the right.

- Align up.
- Align vertical center.
- Align bottom.

Size

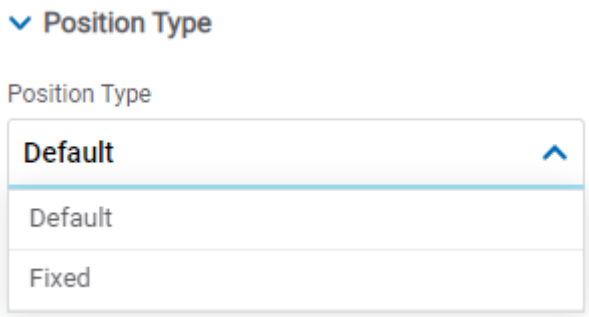


All size properties can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh). Additionally, for Width and Height properties, the "auto" option is added.

> *If selecting percentage (%) for the width and height properties, the size is calculated relative to the top element.*

Position Type

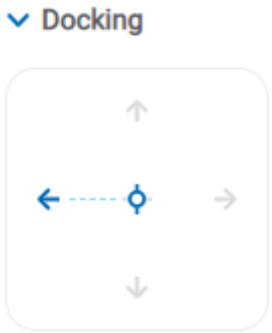
It determines if at the time of execution the element remains fixed on the page or if it moves as the scroll bar moves forward or backward.



Possible Values

- Default: The element has a relative position with respect to the superior element where it was placed (container or section).
- Fixed: Fixes the element to the page so that it is always visible in the same position. It allows to increase the visibility of important content.

Docking



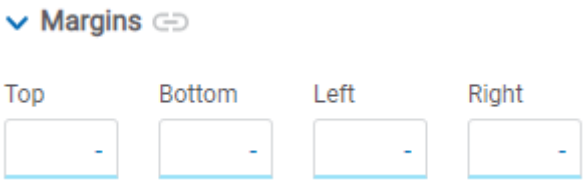
Determines the vertical and horizontal position of elements within the page section or container, when the screen is resized.

When an element is added or moved within the section or container, the modeler automatically docks it to the nearest corners or edges. It can also be configured by clicking on the arrows of the side to be docked to or on the center.

Once the element is docked, its exact position is indicated in the Margins property. The direction of the docking arrows determine the values that are automatically displayed for margins.

The docking position is indicated in the modeling area by dotted lines on the element.

Margins



It allows to define the distance of elements from the borders of their top element. The behavior of margins depends on the docking of the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Top
Distance to the top border of the highest ranking element.

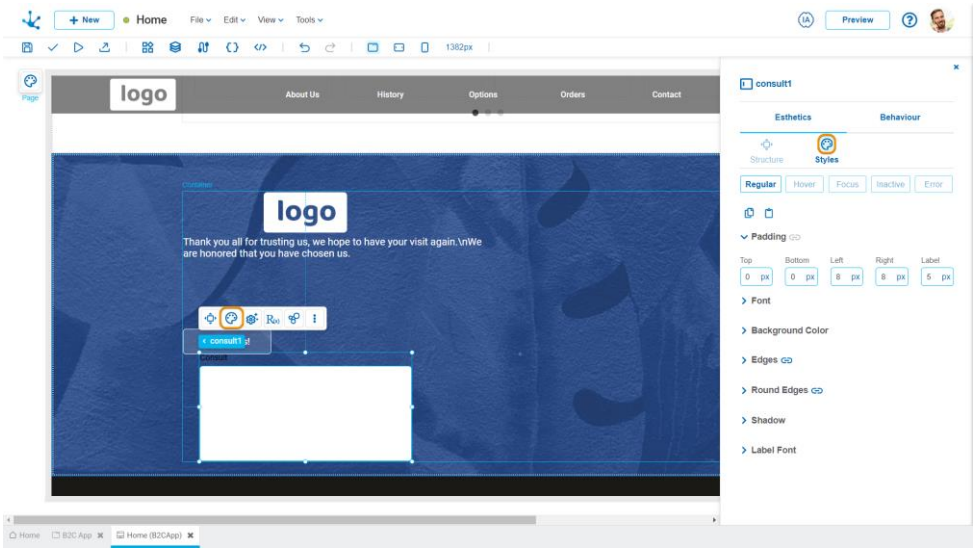
Bottom
Distance to the bottom border of the highest ranking element.

Left
Distance to the left border of the highest ranking element.

Right
Distance to the right border of the highest ranking element.

Allows the value entered in one of the margins to be copied to the other ones automatically.
Allows to indicate different values for each margin.

The style properties panel of an element opens when selecting the icon of its context menu.



This type of element may take different states and for each of them different values for its properties may be modeled.



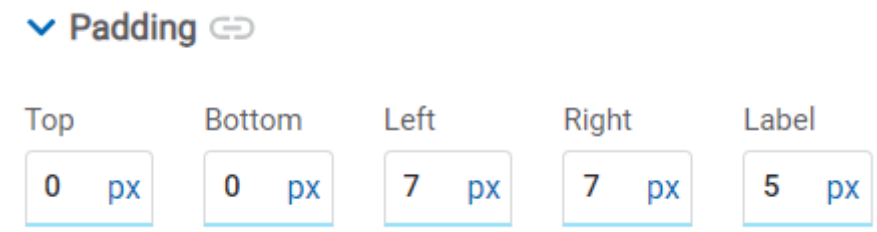
- Regular: the mouse pointer is not over the element.
- Hover: the mouse pointer is over the element.
- Focus: the element is pressed.
- Inactive: the element is not active.
- Error: the element is in error.

Styles

It allows copying the entire current configuration of the style panel for the selected element (Ctrl+Alt+C).

It allows applying the previously copied configuration to another element of the same type (Ctrl+Alt+V).

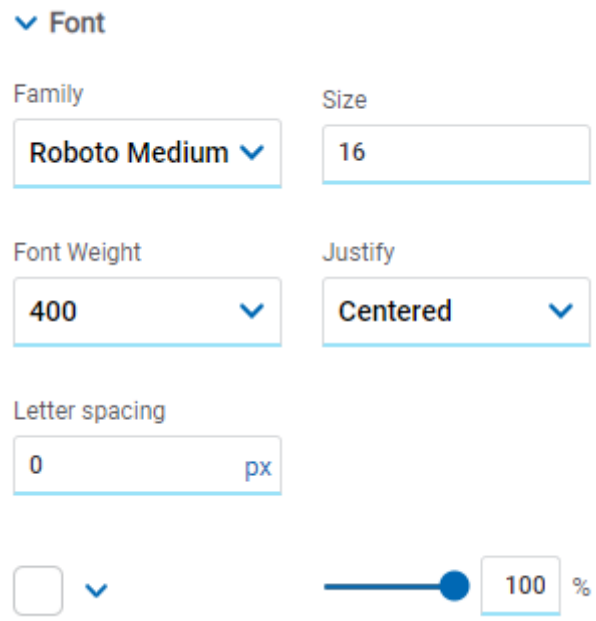
Padding



All padding space properties allow to create a space around the element borders (top, bottom, sides) and its content. The padding space configured is inward, while the margin is configured relative to a top element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Allows values entered in one of the paddings to be copied to the other ones automatically.
Allows to indicate different values for each padding.

Font

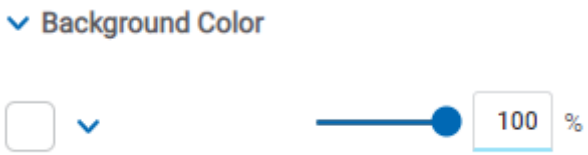


It allows to define the text style.

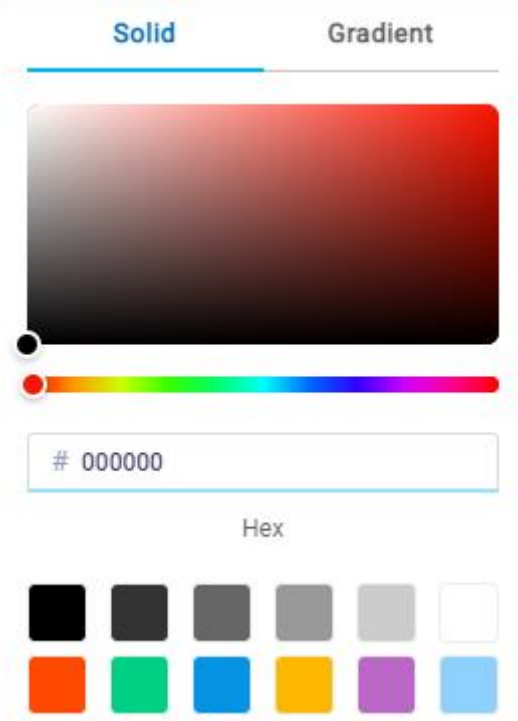
Background Color

It allows to model a background color for the element, it can be solid or gradient and different properties are defined for each one.

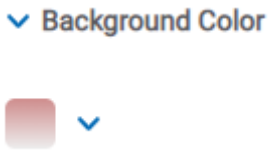
Solid



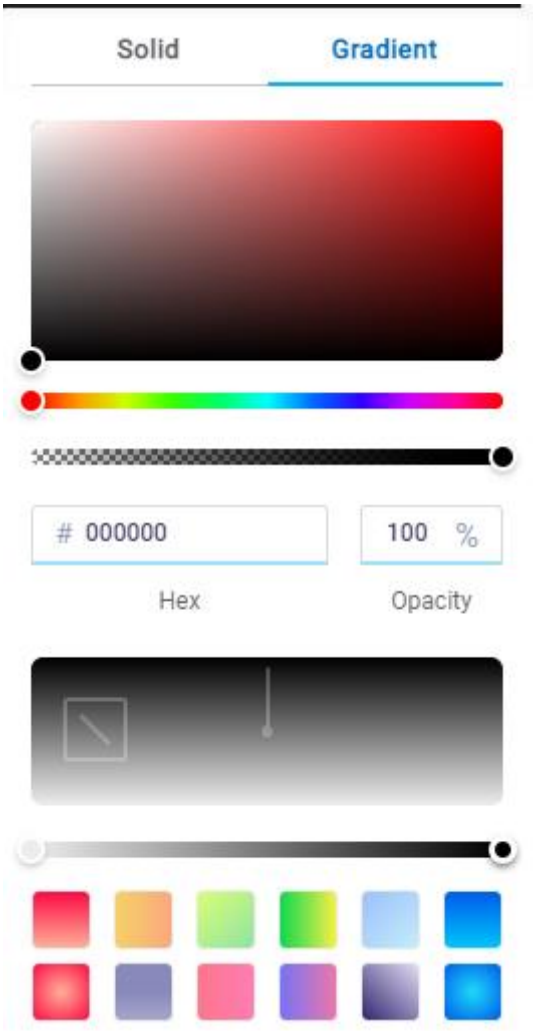
This option enables not only to select the color from a palette but also to define the degree of transparency.



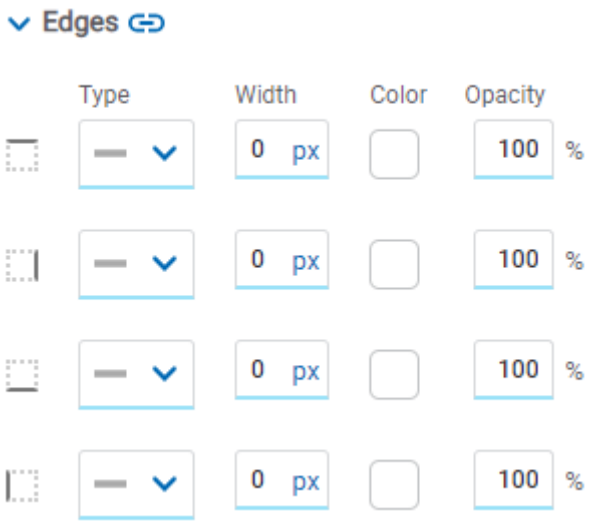
Gradient



This option enables to select not only the color from a palette but also to define its opacity and intensity according to the selected angle.



Edges



It allows to define the style of borders. Each one has its type, width, color and opacity defined, the latter as a percentage.

Allows the value entered in one of the borders to be copied to the other ones automatically.
Allows to indicate different values for each border.

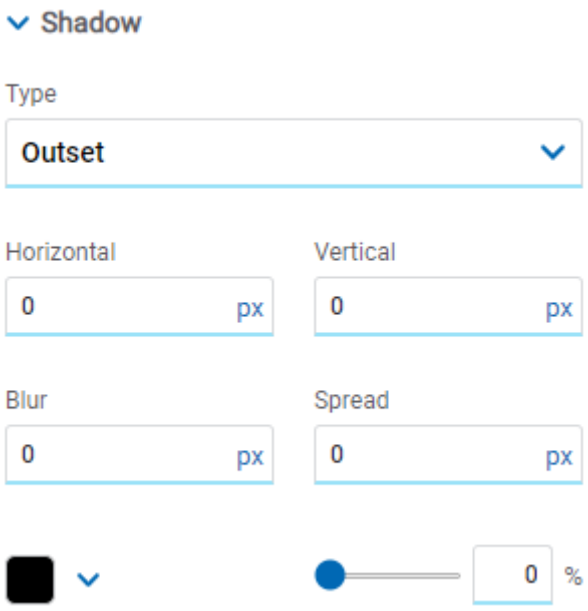
Round Edges



It allows to define the round edges at the corners of elements.
It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Allows the value entered in one of the borders to be copied to the other ones automatically.
Allows to indicate different values for each border.

Shadow



Allows to define a shadow effect around the element.

Type

Possible Values

- Outset
- Inset

Horizontal
Horizontal size of the shadow to the right of the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Vertical
Vertical size of the shadow below the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

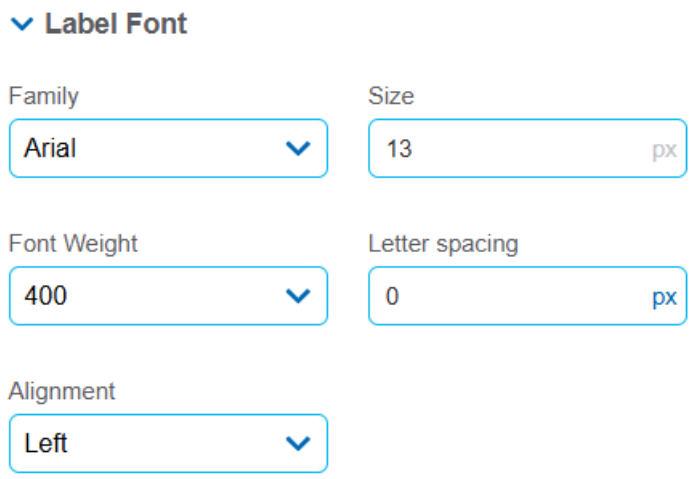
Blur
The greater this value is, the greater and lighter the shadow becomes. If not specified, its value is 0 and the shadow border is darker. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Spread
Positive values will cause the shadow to expand and grow bigger, while negative values will cause the shadow to shrink. If not specified, its value is 0 and the shadow will have the same size as the element.

Color
Defines the color of the shadow.

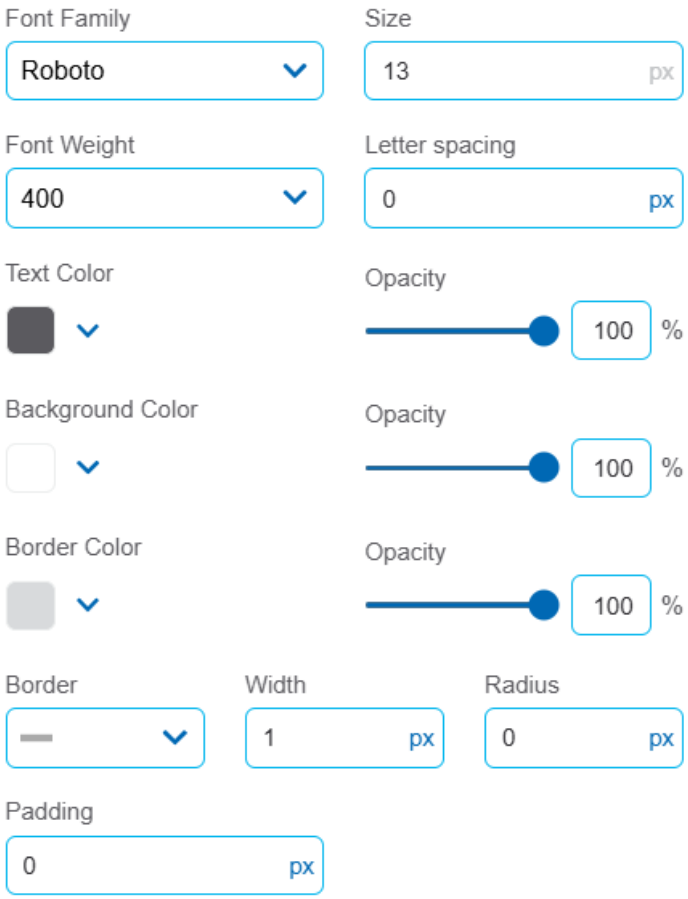
Transparency
Defines the transparency of the shadow.

Label Font



It allows to define the label style.

Tooltip



Source

It allows customizing the font, size, thickness and letter spacing of the pop-up message.

Colors

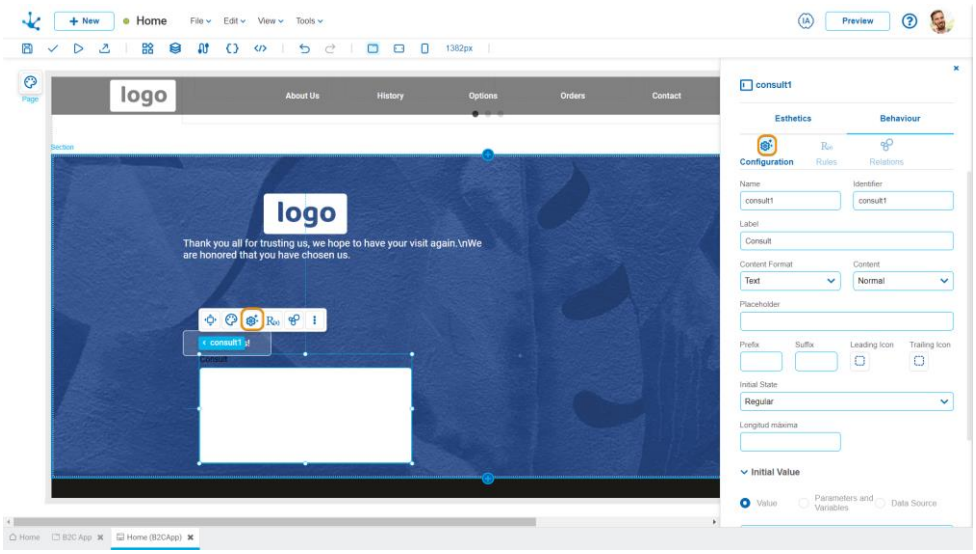
The color and opacity of the text, background and border can be defined. Borders are set uniformly for all sides of the tooltip.

Border

The border type, width, and border radius can be configured.

Configuration Properties

The configuration properties panel of an element opens when clicking the icon  of its context menu.



Name

Name assigned to reference a field when modeling, allowing the field to be uniquely identified within the page. Used in rule wizards to refer to the field within the conditions. It is generated automatically from the Label property, can be modified by the user and does not allow spaces or special characters.

Identifier

It is the name assigned to reference a field in the programming code, used to refer to the field within Java code in the "Execution Code" tab of advanced rules and the JavaScript code in the "Page Code" tab of the page modeler. It allows to uniquely identify the field within the modeled page. It can be modified by the user, as long as no data has been loaded into the page, and it does not allow spaces or special characters.

Tooltip

Pop-up message that appears when hovering over an element, providing additional information.



Text

Content of the message being displayed.

Direction

Position of the message in relation to the element (top, bottom, left, or right).

Label

It allows entering the text that is displayed on the field. Supports white space.

Content Format

It allows modifying the selected field type.

Possible Values

- Text
- Number
- Multiline

- Date and Time
- File

Content

Options available for this property depend on the value of the property Content Format. For each possible value, the options are different:

- Text: the options are "Normal, "Password", "Phone Number" and "Mail".
- Number: the options are "Decimal" and "Integer".
- Date and Time: the options are "Date and Time", "Local Date and Time", "Time", "Local Time", "Date" and "Local Date".
- File: the options for images are "*", "jpeg", "png", "gif", "svg+xml", "bmp", "webp", "tiff" and "x-icon". The options for audio are "*", "mpeg", "wav", "ogg", "midi", "aac" and "x-ms-wma". Video options are "*", "mp4", "quicktime", "webm", "x-msvideo", "x-flv", "3gpp". Application options are "*", "pdf", "msword", "vnd.ms-excel", "vnd.ms-powerpoint", "zip", "x-rar-compressed", "x-tar", "x-gzip", "x-bzip2", "jason"and "xml". The options for text are "*", "plain", "csv" and "xml".

If this option is selected, an additional property is enabled.

Display Mode

It allows modifying the way the file uploaded by the user is presented.

Possible Values

- File Name: Shows only the file name when the user uploads it, without an additional preview.
- Previewer: Presents a preview of the uploaded file, providing the user with a more user-friendly and interactive view.

Placeholder

It allows guiding the user about the content when entering the element. The placeholder text is displayed within the element.

Picker

It allows defining an interval picker that helps the user to increase or decrease the numeric field value according to a defined interval. Only available if the Field Type property is of type "Number".

Possible Values

- Align to the left: The picker is displayed on the left side of the numeric field.
- Align to the right: The picker is displayed on the right side of the numeric field.
- Hide: The picker is not displayed in the field; the arrow keys must be used to increase or decrease the value.

Interval

It allows defining the value that is increased or decreased each time the user interacts with the picker. For example, if the interval value is set to 5, clicking the increment icon increases the numeric field value by 5 units. Similarly, clicking the decrement icon decreases the field value by 5 units.

Prefix

It allows incorporating a text or symbol that is displayed on the left side of the field, before the entered value. It is used to provide additional context immediately, making it easier to understand the purpose or format of the field. For example, the $ sign can be set for monetary fields, indicating that a value in a specific currency is expected.

Suffix

It allows incorporating a text or symbol that is displayed on the right side of the field, immediately after the entered value. It is used to complement the information provided, offering details that indicate the context or units of the value entered. The meaning of the value can be specified by including

units of measurement, percentages, or frequencies—for example, "kg" in a field intended to record weight.

Leading Icon

It allows incorporating an icon on the left side of the field, before the value that is entered or displayed. This leading icon can graphically represent the expected data type, making it easier to quickly identify the purpose of the field. For example, a phone icon can be used to indicate that a contact number should be entered.

Trailing Icon

It allows incorporating an icon on the right side of the field, immediately after the value that is entered or displayed. This trailing icon is intended to complement the meaning of the field through a visual element, making it easier to understand the context in which it is located. For example, a "%" icon can be used to indicate that a percentage value should be entered.

Initial State

Defines the visual and functional state of the field when the page is loaded for the first time. This setting allows determining whether a field is enabled and ready to receive interaction or, on the contrary, is disabled, waiting for an action or precondition to be activated.

Possible Values

- Regular: The field is enabled by default. Allows the user to interact with the field immediately upon loading the entity.
- Inactive: the field is shown as disabled and does not allow interaction.

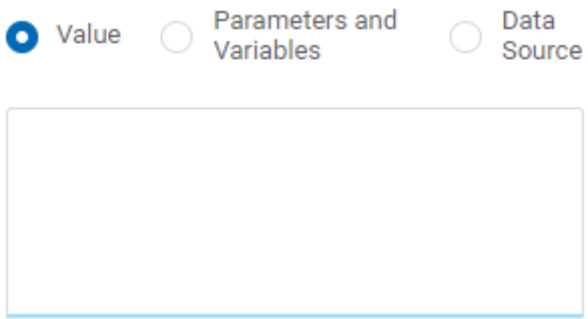
Maximum Length

It allows defining the maximum number of characters that can be entered in a text field.

Initial Value

Allows to select the source of the element's content.

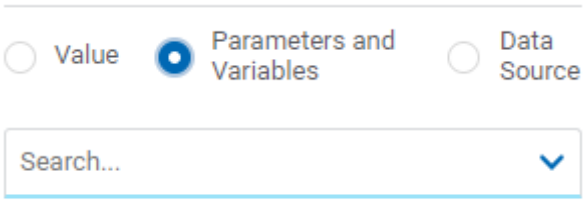
Value



Value

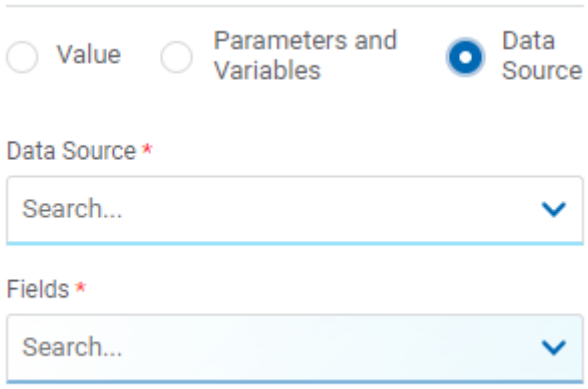
Allows to enter a text that is displayed in the element.

Parameters and Variables



Parameters and Variables
Allows to select a parameter or a variable from, whose value is displayed in the element.

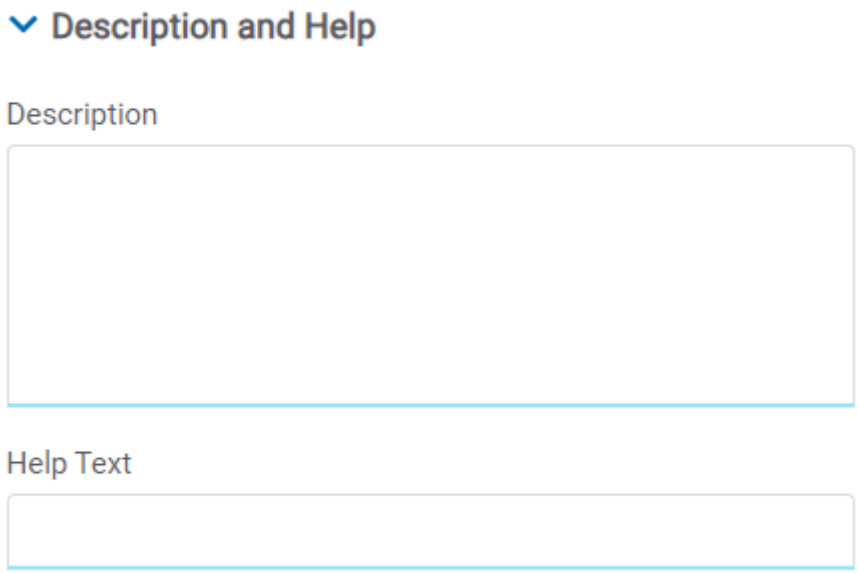
Data Source



Data Source
It allows selecting a previously defined data source within the object.

Fields
It allows selecting a field from the data retrieved in the chosen data source, whose value is displayed in the element.

Description and Help

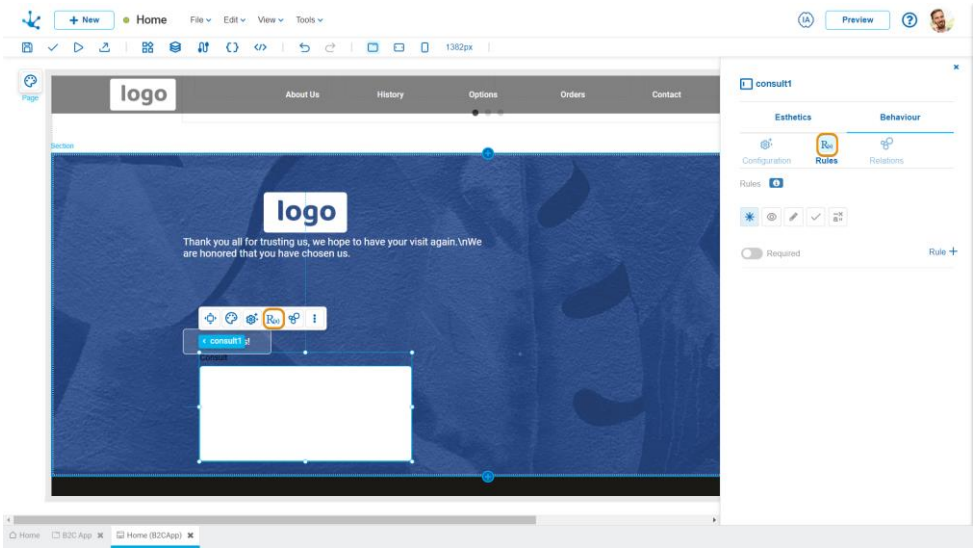


Description
Text that defines the field and optionally its content.

Help Text
This function is to guide the user about the content to be entered in the field. The text entered as help is displayed when the user hovers the cursor over the field.

Rule Properties

The rule properties panel of an element opens when clicking the icon  of its context menu.



Properties

Rules

Embedded rules on behavior, validation and calculation can be defined, associated with an element, by using the wizard (ctrl + space).

Shows syntax examples for writing the rules.

Required

Indicates whether the element is required on the page.

Required Not required (default)

Rule + Opens an edit area where a rule to determine the required condition can be defined. If a rule is defined, the icon is displayed with light blue borders.

Visible

Indicates whether the element is visible. If this property is not checked, the element is not displayed in the page.

Visible (default) Not visible

Rule + Opens an edit area where a rule to determine the visibility condition can be defined. If a rule is defined, the icon is displayed with light blue borders.

Editable

Indicates if the element is editable. If this property is not checked, the user cannot enter or modify values in the element.

Editable (default) Not editable

Rule + Opens an edit area where a rule to determine the editability condition can be defined. If a rule is defined, the icon is displayed with light blue borders.

Validation

Rule + Opens an edit area where you can define the condition that determines if the element value is correct or not. It is possible to define more than one rule. If rules are defined, the icon is displayed with light blue borders.

Calculation

Rule + Opens an edit area where you can define the expression to be executed to calculate the element value. If a rule is defined, the icon is displayed with light blue borders.

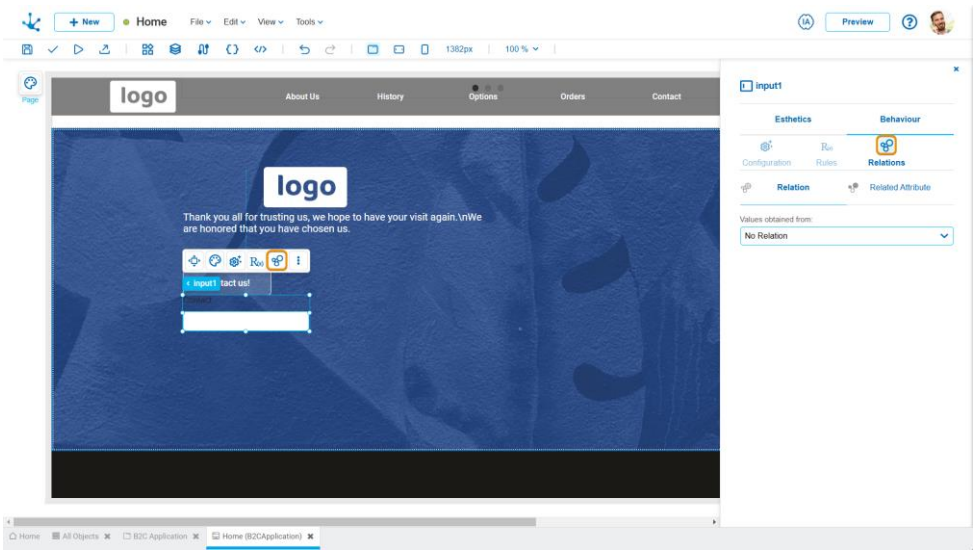
Operations when defining the rule:

- Saves the new or modified rule
- Cancels the operation

Operations once the rule is defined:

- Edits the existing rule
- Deletes the rule

The relation properties panel of an element opens when clicking the icon of its context menu.



Properties are divided into two groups, relation properties and related attribute properties.

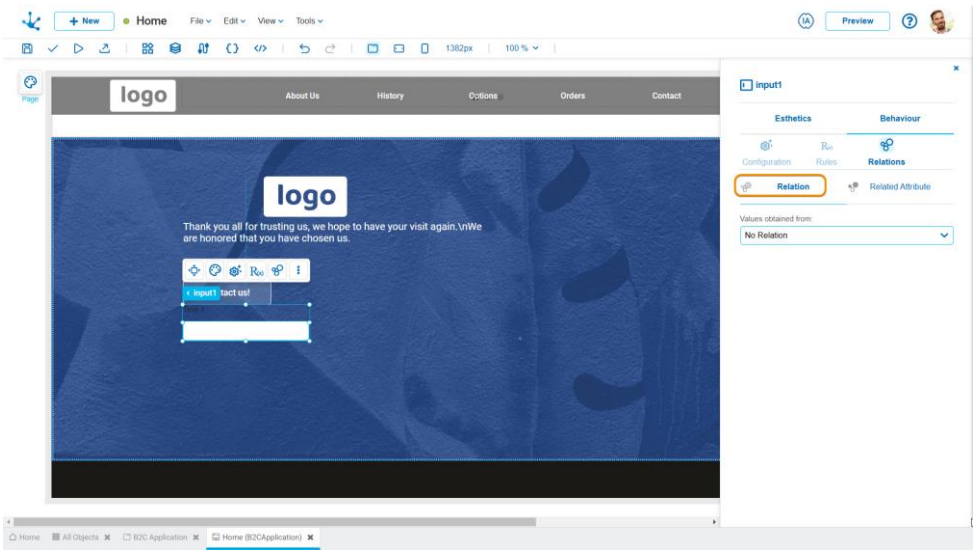
Relation

A relation allows obtaining values from different sources for the element being modeled. Within the "Relation" tab, the property Values obtained from is defined, along with the specific settings for each type of source. If the element has no relation, the default option “No Relation” should be modeled.

Possible Values

- No Relation
- Entity
- Value List
- Rule

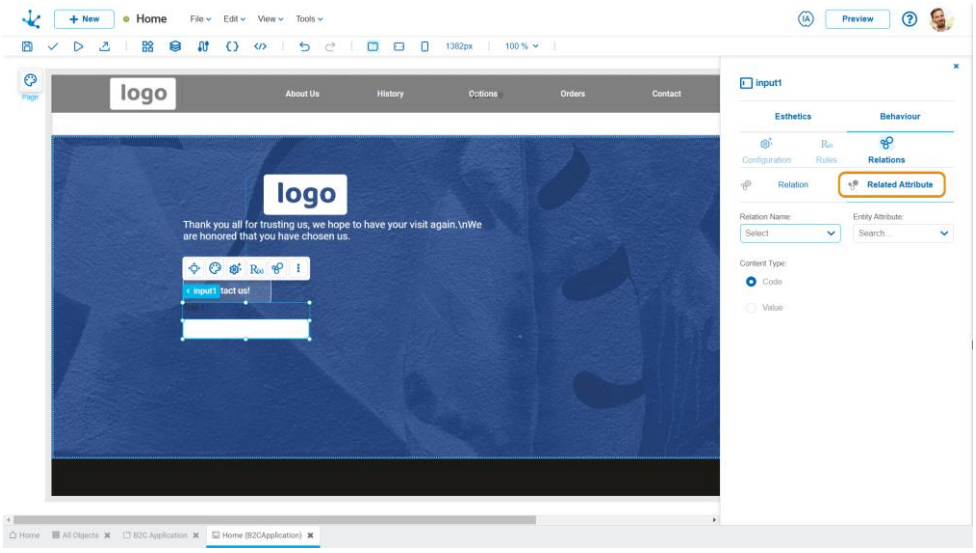
The relation is only valid for text-type fields.



Related Attribute

A related attribute indicates that the field value is retrieved from the attribute value of another entity. One of the relations defined on the page must be selected, and then it should be specified which attribute of the related entity it links to.

A field can be modeled as a related attribute and it can also have a relation defined to an entity, a value list, or a rule. This allows the value of the related attribute to be retrieved on this field and the functionality provided by the relation to be used.



Properties

Relation Name

The list of relations defined on the page is displayed, allowing the selection of one of them.

Entity Attribute

The list of attributes of the selected entity compatible with the field type is displayed. Fields can have a relation modeled with an entity, a value list, or a rule in the selected entity.

Content Type

The element content varies depending on whether the Code or Value property is checked when:

- The entity attribute has a modeled relation.
- The field has the option "Entity" selected in the Values Obtained from property.

Code

When this property is checked, if the relation is modeled with a:

- Related entity, the identifier of the entity is displayed.
- Value list, the value list code is displayed.
- Rule, the code returned by the rule's execution is displayed.

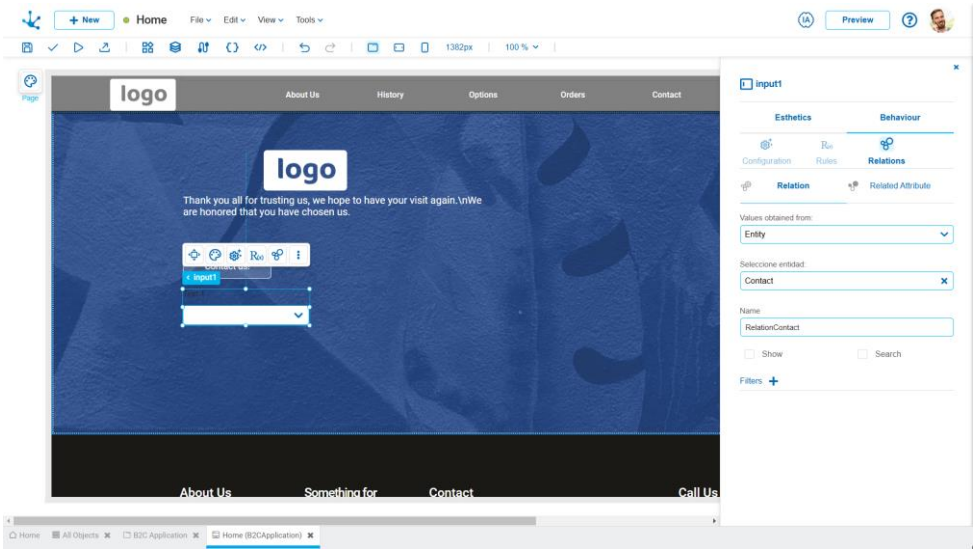
Value

When this property is checked, if the relation is modeled with a:

- Related entity, the short description of the entity is displayed.
- Value list, the value list description is displayed.
- Rule, the description returned by the rule's execution is displayed.

Entities

When relating a field to an entity, the field can take a value that is obtained from any instance of the entity, specifically from the content modeled in its short description.



To define a relation with an entity, first select the entity to establish the relation with, and then complete a set of properties.

Properties

Name

Name of the relation between both entities; it is a modeling-oriented property. It does not allow blanks and must be unique for each modeled page.

Permissions

The relation permissions define which functionalities are enabled for the field.

Show

Check this property to allow the user of the page to display the instance of the entity with which the relation is established.

Search

Check this property to allow the page user to access the results grid and search for the instance of the entity with which the relation is established.

Filters

It allows for narrowing the search results on the related entity.
This filter is applied both in the field's autocomplete and in the show through the magnifying glass when hovering the mouse over the field.
To create a filter, click on the icon + and a panel opens to complete the following properties:

Filtered Attribute

It allows the selection of an attribute of the related entity.

Condition

It allows the selection of a condition as part of the filter.

Type

The possible values to select are "Value" and "Field".

Value

It allows the input of fixed values.

Field

It allows the selection of a page field. It must be considered that its content depends on the Content property of the related attribute

Example of Filtered Entities

In this example, it is described how to model a page that includes a field related to an entity, and how to apply a filter on that field based on the value of another field in the related entity, meeting a specific condition.

1. Entity modeling

A "Product Catalog" entity is modeled with the following fields.

- Code: Unique product identifier.
- Description: Product detail or name.
- Category: Product classification (e.g. vehicles, real estate, household appliances).

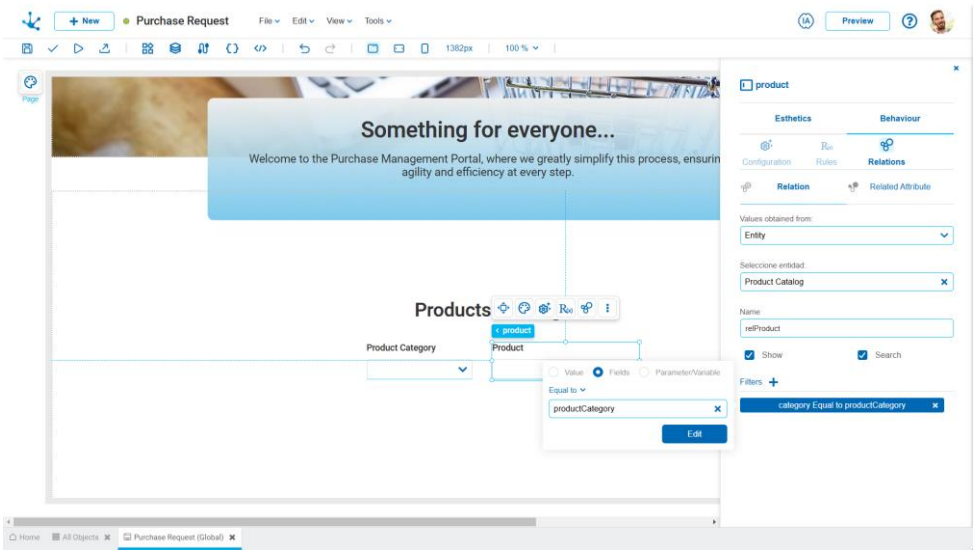
2. Page modeling

On the "Purchase Order" page, the following fields are included.

- Product Category: related to a value list.
- Product: related to the "Product Catalog" entity.

3. Filter configuration on the "Product" field

- Modeling the relation with the "Product Catalog" entity.
- Add a filter in order to define that the "Category" field of the "Product Catalog" entity matches the value selected in the "Product Category" field of the page.

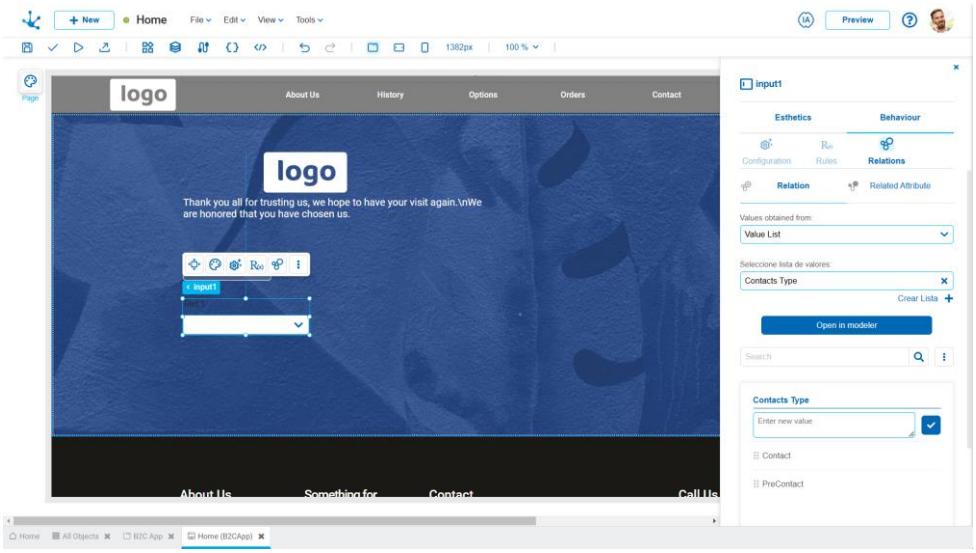


When executing the page, if a category is selected, in the "Product" field will only display catalog values corresponding to that category.

Lista de Valores

When a value list relates to a field, a set of options organized under a specific criterion is defined. This allows the application user to display the possible values that the field can take.

From the modeler, new lists can be created, edited, and linked to entity fields.



To create a new list of values, you must click on the icon +.
A panel opens where the list name is entered, the application it belongs to is selected, and if it is specified to be sorted alphabetically, the descending order can be selected. Clicking the "Accept" button creates the value list, subsequently allowing the entry of new values.

The value list can be used in any other field of the same or another page.

Property

Select value list
It allows selecting a specific list from the set of existing lists with the option to filter the set by entering text into the search field located above the list.

Other Elements

Open in modeler
It allows using the value lists modeler for the definition, instead of the field properties panel.

 Allows to filter values from the list based on the characters entered. If a list is very long it helps users to easily visualize the desired values.
 Enables an option that allows to add the internal code to the list values.

Operations on Values

-  Allows adding each entered value to the list of values.
-  It is displayed if the list has the Icons property modeled. It allows to associate icons to the list values.

- Double click: Allows to modify a value in the list.
- Move: Allows to change the position of a value within the list by dragging the value with the mouse.

Hovering the cursor over each of the values entered, a set of icons is displayed and this allows to perform different operations.

-  Allows to delete a value from the list of values. Once deleted, it is displayed in gray and crossed out.
-  Allows to restore a previously deleted value.

Display the Selected Line

-  Hides the icons that are displayed.
-  Shows hidden icons.

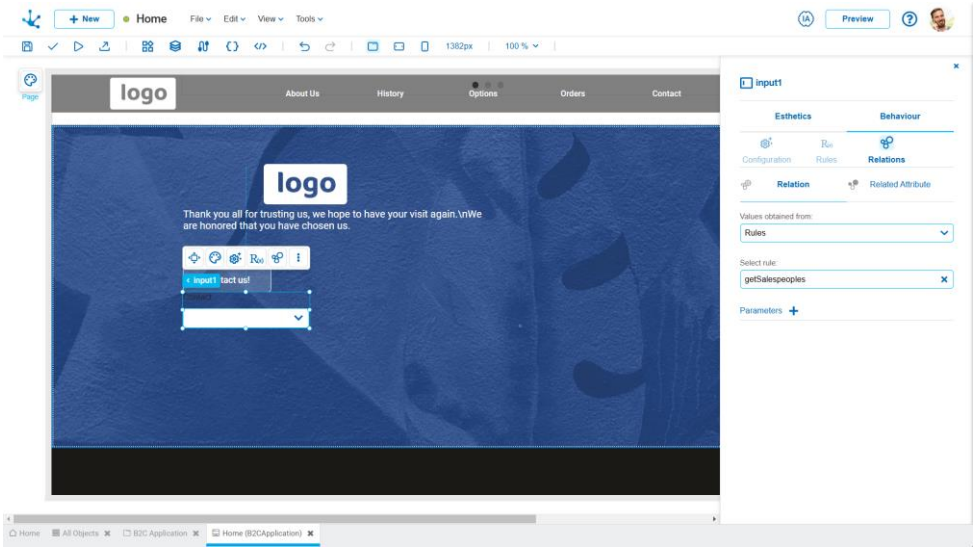
Rules

The rules that relate to fields aim to implement business logic to dynamically define the values that the field can offer as selection options in a dropdown.

Relating a field to a business rule provides greater flexibility than binding to a value list, allowing to apply custom logic, add search filters, and dynamically adjust results based on different criteria.

Once the rule is implemented, it can be associated with a field. To do this, it is necessary to activate the Enable Relation to a Field property.

> *Rules can only relate to text fields.*



Properties

Select rule

It allows selecting a rule from the set of rules modeled in the environment, with the option to filter the set by entering text into the search field.

Parameters

If it is necessary to pass input parameters to the rule, they must be selected and assign values to them, which can be fixed values, variables or fields of the same entity.

Example

A previously modeled rule "getUserByName" can be related and enter a fixed value "perez" as a parameter. In this way, all the users that contain the text 'perez' in their last name.

Image

Allows to add any multimedia content of visual type.

The element properties are represented by icons in its context menu, where its operations are also available.

Subtypes

By selecting the “Image” option from the icon on the top toolbar, a list with the different subtypes of this element is displayed, which can be dragged to the modeling area. Each subtype has modeled element properties, predefined in a specific way.

- Rounded Image
- Square Image
- Small Image
- Medium Image
- Large Image

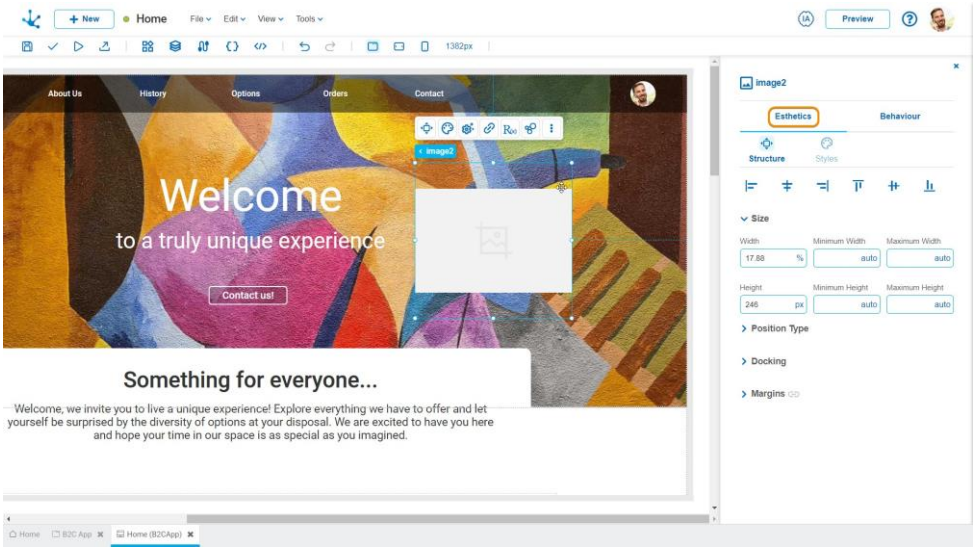
Classification of Properties

Properties are divided into two groups: esthetic properties and behavior properties.

Esthetic Properties

In the esthetic properties panel, the following are grouped:

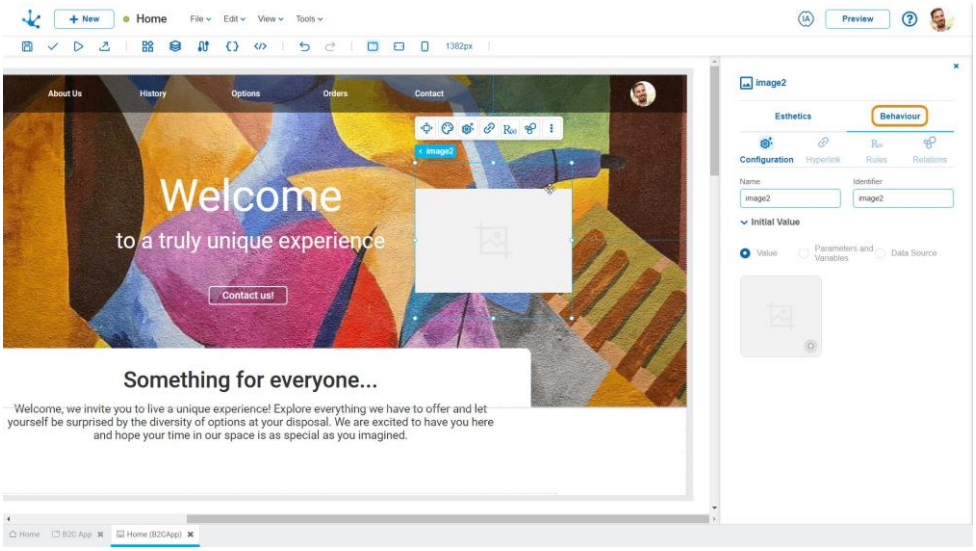
- Structure Properties
- Style Properties



Behavior Properties

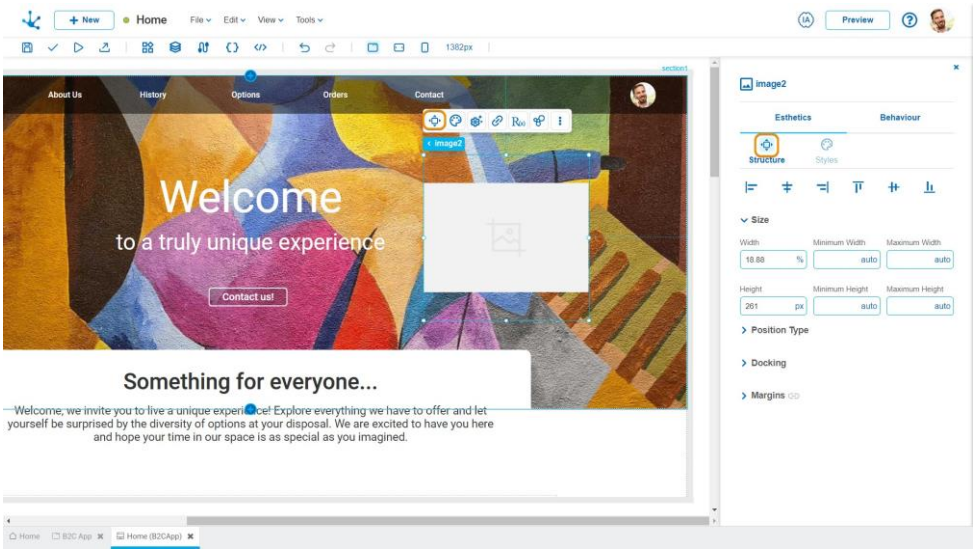
In the behavior properties panel, the following are grouped:

- Configuration Properties
- Hyperlink Properties
- Rules Properties
- Relations Properties



Structure Properties

The structure properties panel of an element opens when selecting the icon  of its context menu.



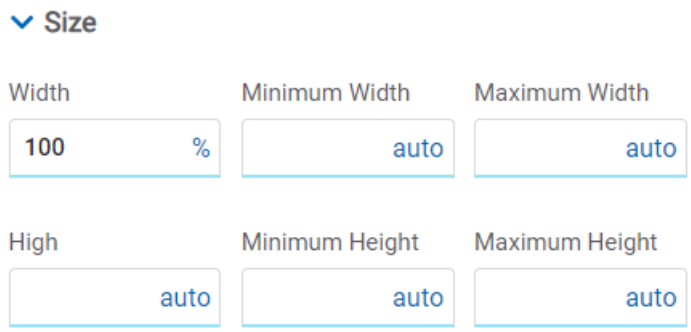
Alignment

Allows to align the element by selecting one of the available icons.

-  Align to the left.
-  Align to horizontal center.
-  Align to the right.

- Align up.
- Align vertical center.
- Align bottom.

Size

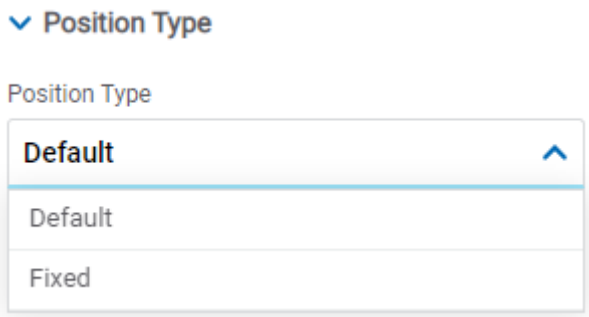


All size properties can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh). Additionally, for Width and Height properties, the "auto" option is added.

> *If selecting percentage (%) for the width and height properties, the size is calculated relative to the top element.*

Position Type

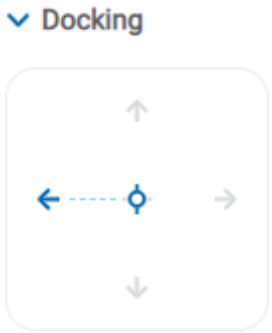
It determines if at the time of execution the element remains fixed on the page or if it moves as the scroll bar moves forward or backward.



Possible Values

- Default: The element has a relative position with respect to the superior element where it was placed (container or section).
- Fixed: Fixes the element to the page so that it is always visible in the same position. It allows to increase the visibility of important content.

Docking



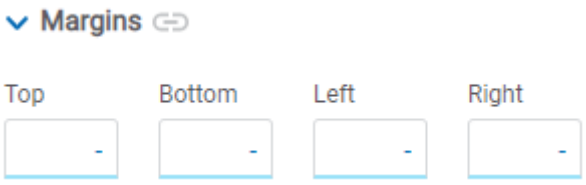
Determines the vertical and horizontal position of elements within the page section or container, when the screen is resized.

When an element is added or moved within the section or container, the modeler automatically docks it to the nearest corners or edges. It can also be configured by clicking on the arrows of the side to be docked to or on the center.

Once the element is docked, its exact position is indicated in the Margins property. The direction of the docking arrows determine the values that are automatically displayed for margins.

The docking position is indicated in the modeling area by dotted lines on the element.

Margins



It allows to define the distance of elements from the borders of their top element. The behavior of margins depends on the docking of the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Top
Distance to the top border of the highest ranking element.

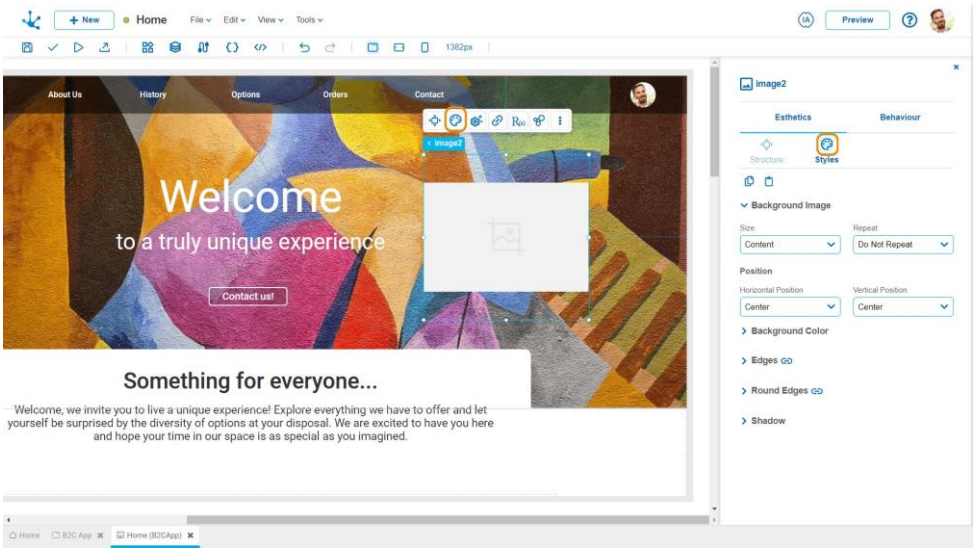
Bottom
Distance to the bottom border of the highest ranking element.

Left
Distance to the left border of the highest ranking element.

Right
Distance to the right border of the highest ranking element.

Allows the value entered in one of the margins to be copied to the other ones automatically.
Allows to indicate different values for each margin.

The style properties panel of an element opens when selecting the icon of its context menu.



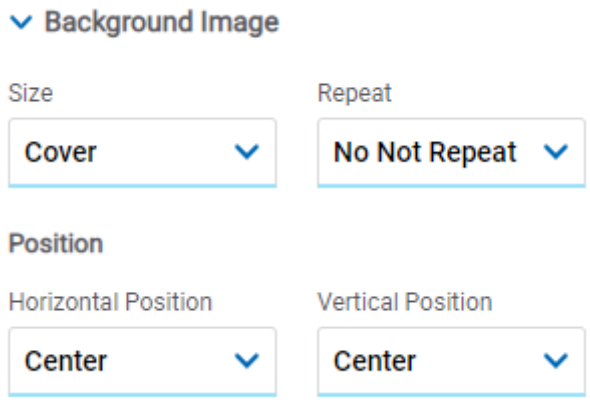
Styles

It allows copying the entire current configuration of the style panel for the selected element (Ctrl+Alt+C).

It allows applying the previously copied configuration to another element of the same type (Ctrl+Alt+V).

Background Image

Defines the properties of the background image.



Size

Possible Values

- Cover
- Content
- Auto

Repeat

Possible Values

- Repeat
- Repeat Hor.
- Repeat Vert.
- Spacing
- Rounded
- Do Not Repeat

Horizontal and Vertical Position

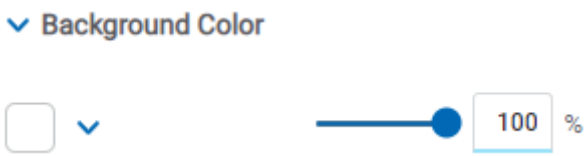
Possible Values

- Center
- Left
- Right

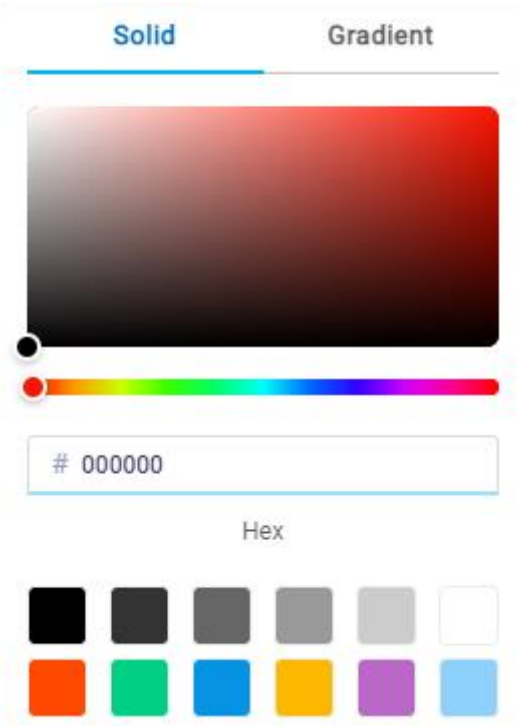
Background Color

It allows to model a background color for the element, it can be solid or gradient and different properties are defined for each one.

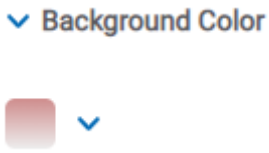
Solid



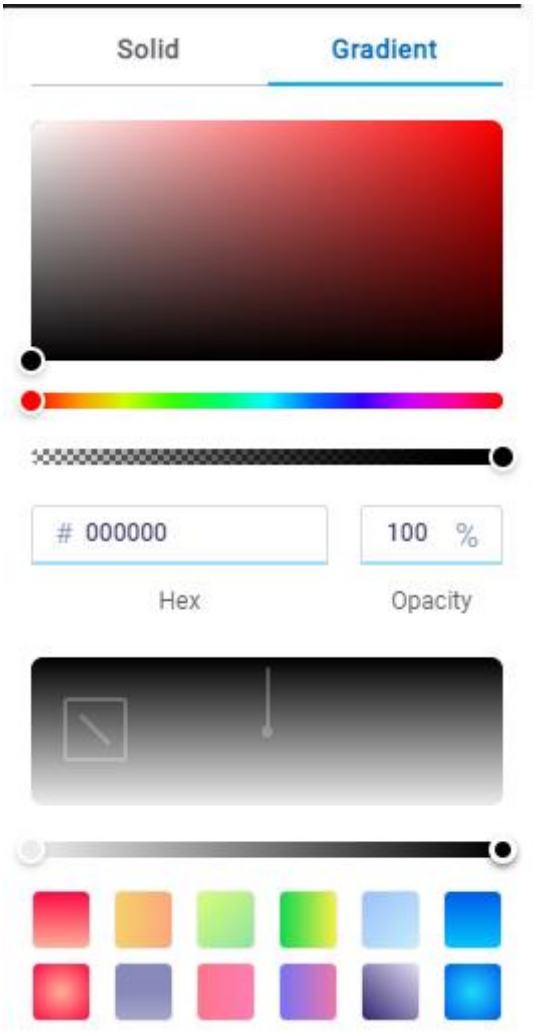
This option enables not only to select the color from a palette but also to define the degree of transparency.



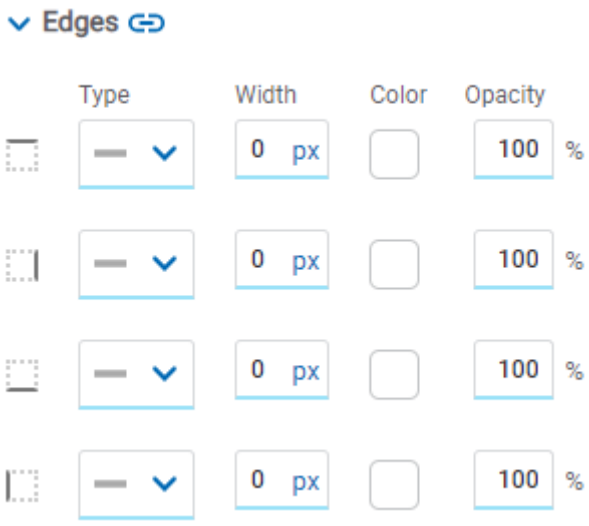
Gradient



This option enables to select not only the color from a palette but also to define its opacity and intensity according to the selected angle.



Edges



It allows to define the style of borders. Each one has its type, width, color and opacity defined, the latter as a percentage.

Allows the value entered in one of the borders to be copied to the other ones automatically.
Allows to indicate different values for each border.

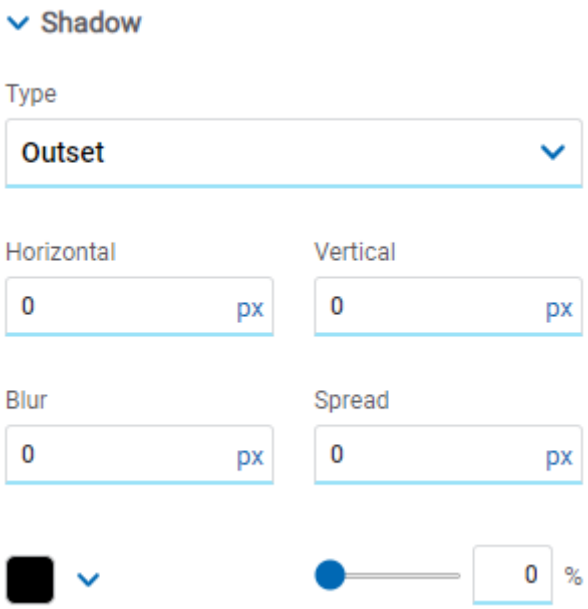
Round Edges



It allows to define the round edges at the corners of elements.
It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Allows the value entered in one of the borders to be copied to the other ones automatically.
Allows to indicate different values for each border.

Shadow



Allows to define a shadow effect around the element.

Type

Possible Values

- Outset
- Inset

Horizontal
Horizontal size of the shadow to the right of the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Vertical
Vertical size of the shadow below the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

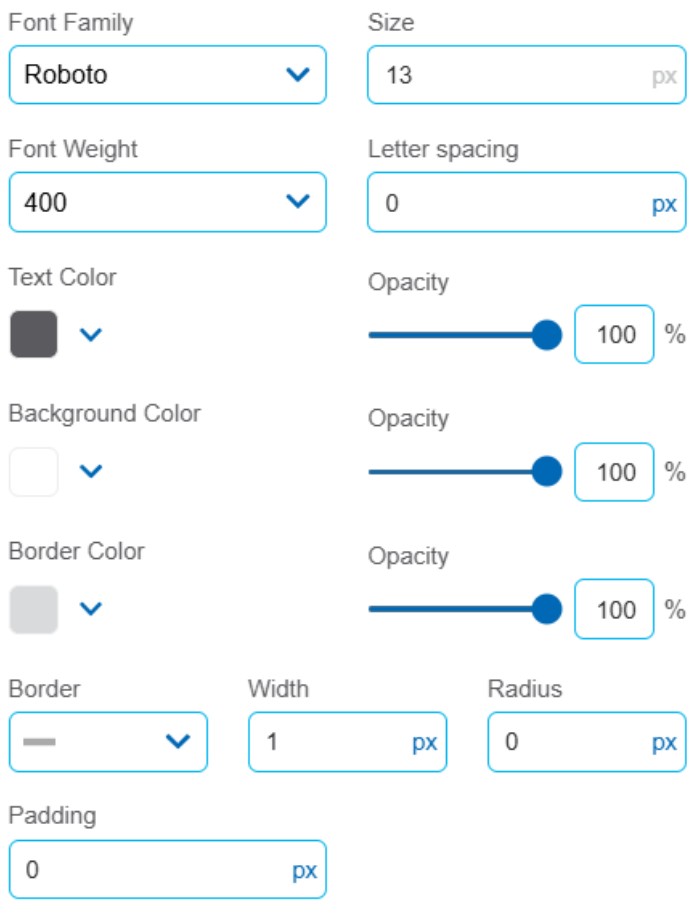
Blur
The greater this value is, the greater and lighter the shadow becomes. If not specified, its value is 0 and the shadow border is darker. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Spread
Positive values will cause the shadow to expand and grow bigger, while negative values will cause the shadow to shrink. If not specified, its value is 0 and the shadow will have the same size as the element.

Color
Defines the color of the shadow.

Transparency
Defines the transparency of the shadow.

Tooltip



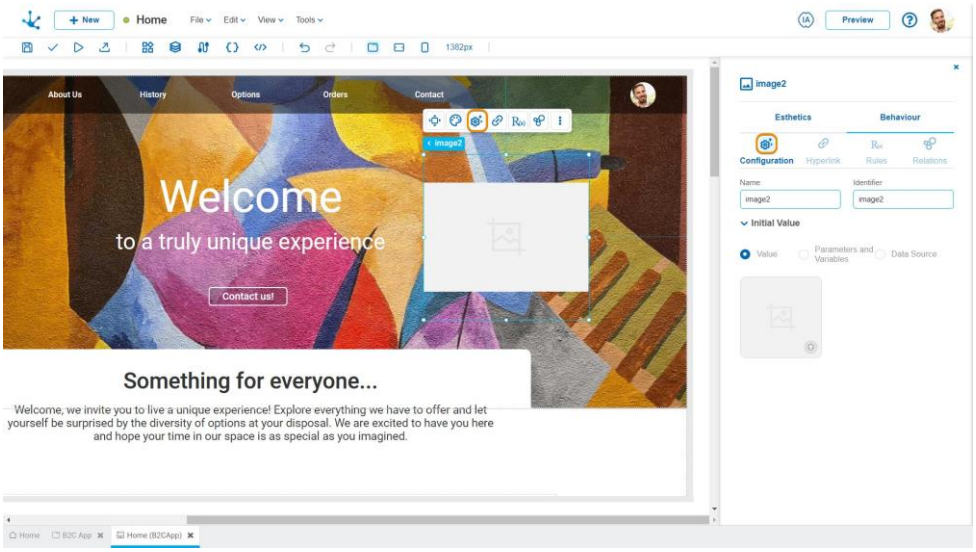
Source
It allows customizing the font, size, thickness and letter spacing of the pop-up message.

Colors
The color and opacity of the text, background and border can be defined. Borders are set uniformly for all sides of the tooltip.

Border
The border type, width, and border radius can be configured.

Configuration Properties

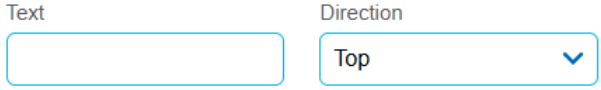
The configuration properties panel of an element opens when clicking the icon  of its context menu.



Name
Name used to reference the element during modeling.

Identifier
Uniquely identifies the element. It is used in the Javascript SDK.

Tooltip
Pop-up message that appears when hovering over an element, providing additional information.

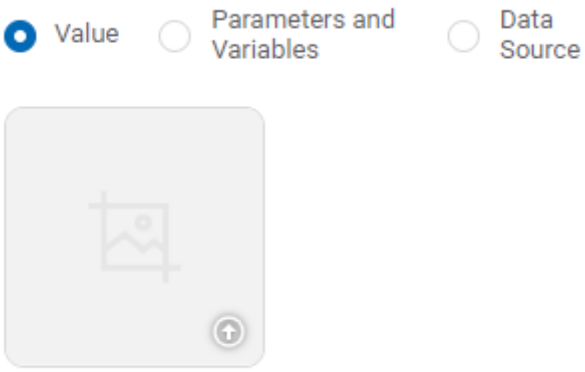


Text
Content of the message being displayed.

Direction
Position of the message in relation to the element (top, bottom, left, or right).

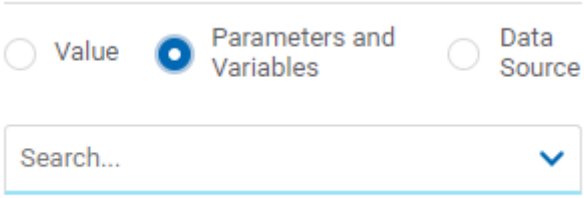
Allows to select the source of the element's content.

Initial Value



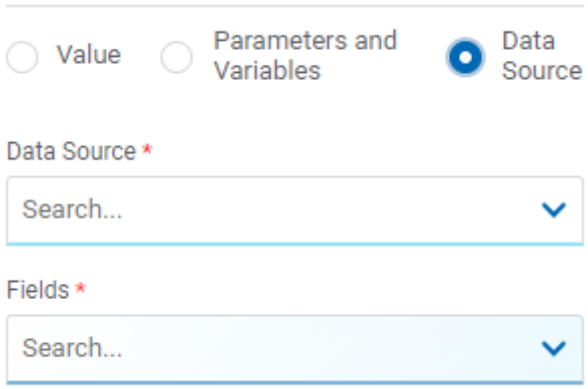
It allows adding an image from a file on the user's computer.

Parameters and Variables



Parameters and Variables
Allows to select a parameter or a variable from, whose value is displayed in the element.
This value can be a valid url.

Data Source



Data Source

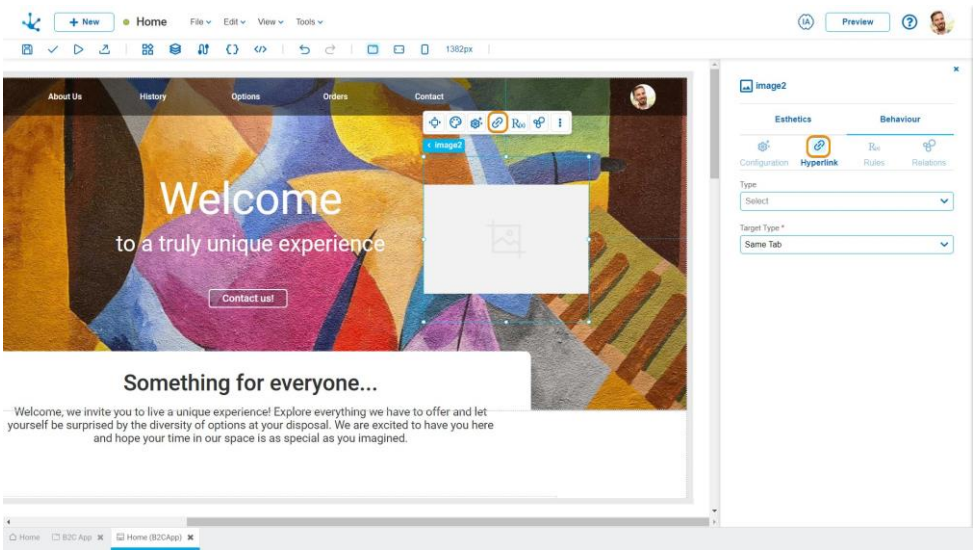
It allows selecting a previously defined data source within the object.

Fields

It allows selecting a field from the data retrieved in the chosen data source, whose value is displayed in the element.

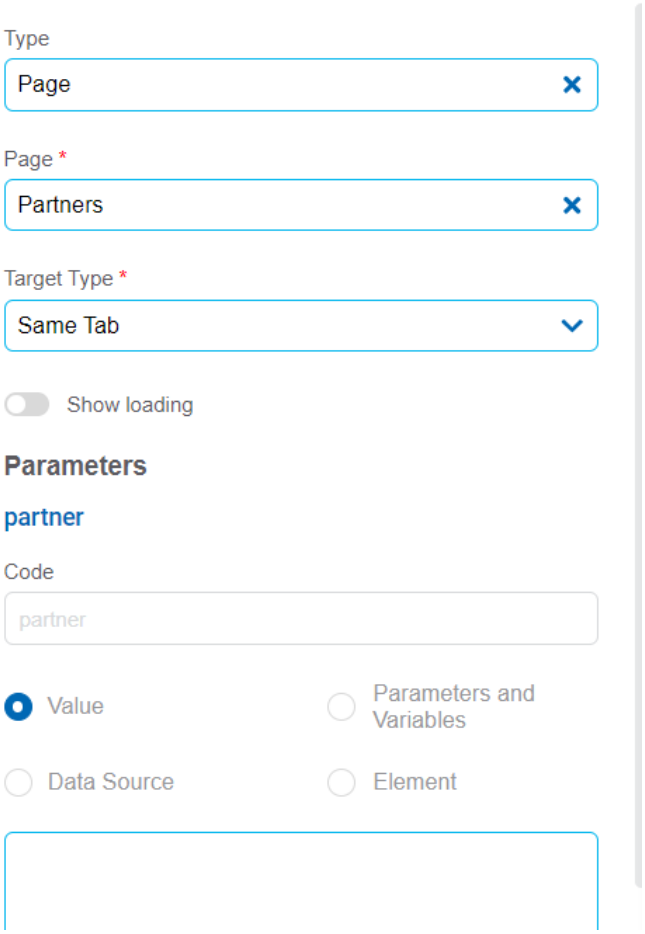
Hyperlink Properties

The hyperlink properties panel of an element opens when clicking the icon of its context menu.



Define the behavior of the element when selecting it. Different properties are enabled depending on the type of object selected.

Page



Page

The pages modeled in the environment are displayed.

Destination Type

The available options for opening the object are displayed.

Possible Values

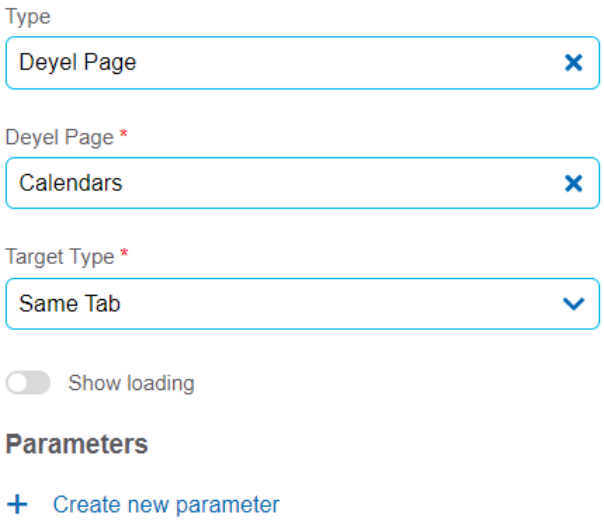
- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  - Modal Horizontal Size
    Defines its width.
  - Modal Vertical Size
    Defines its height.
- Iframe: selecting this option enables an additional property.
  - Destination Iframe
    Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading

Shows the loading icon and disables the user interaction until the redirection is over.

Parameters
If there are parameters, their value can be specified with text, parameters and variables or data source.

Deyel Page



Deyel Page
The pages belonging to **Deyel** are displayed.

Destination Type
The available options for opening the object are displayed.

Possible Values

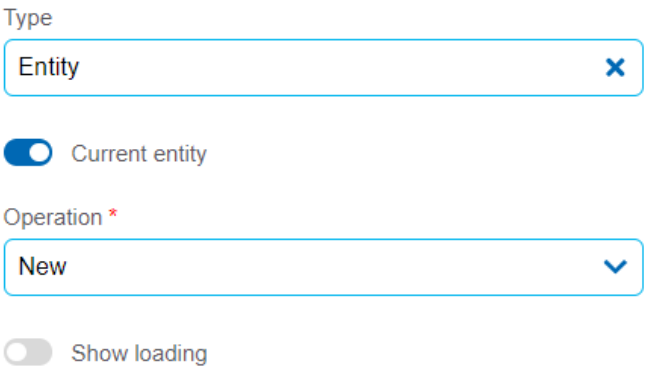
- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  - Modal Horizontal Size
    Defines its width.
  - Modal Vertical Size
    Defines its height.
- Iframe: selecting this option enables an additional property.
  - Destination Iframe
    Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Parameters

Allows sending parameters to the selected object type.

Entity



Current Entity

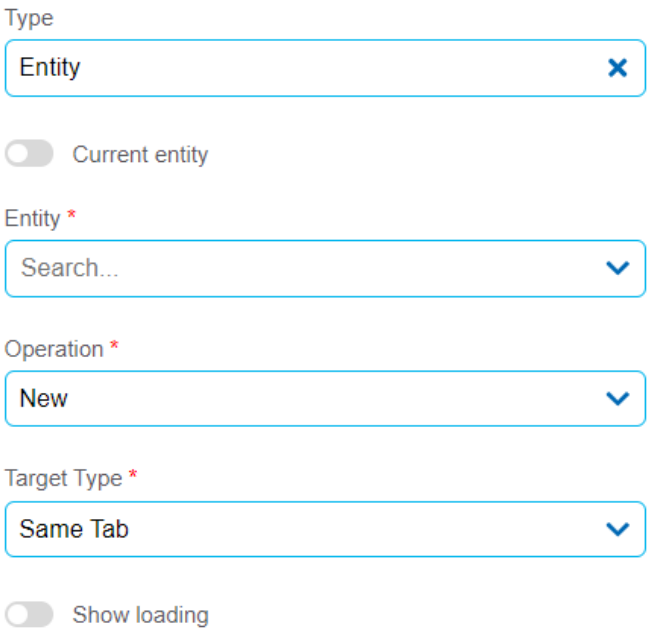
If this property is checked, the operation applies directly to the entity being modeled.

Operation

Defines the operation made when selecting the object.

Possible Values

- New: Indicates that the current entity panel is opened for the creation of an instance.
- Modify: Indicates that the current entity panel is opened for the update of an instance.
- Delete: Indicates that the current entity panel is opened for the deletion of an instance.



Entity
If the property Current Entity, is not checked, the entities modeled in the environment are displayed.

Operation
Defines the operation made when selecting the object.

Possible Values

- New: The panel of the entity selected in the previous property is opened to create an instance.
- Update The panel of the entity selected in the previous property is opened to update an instance.
- Show: The panel of the entity selected in the previous property is opened to show an instance.
- Grid: Indicates that the grid of the entity selected in the previous property is opened.

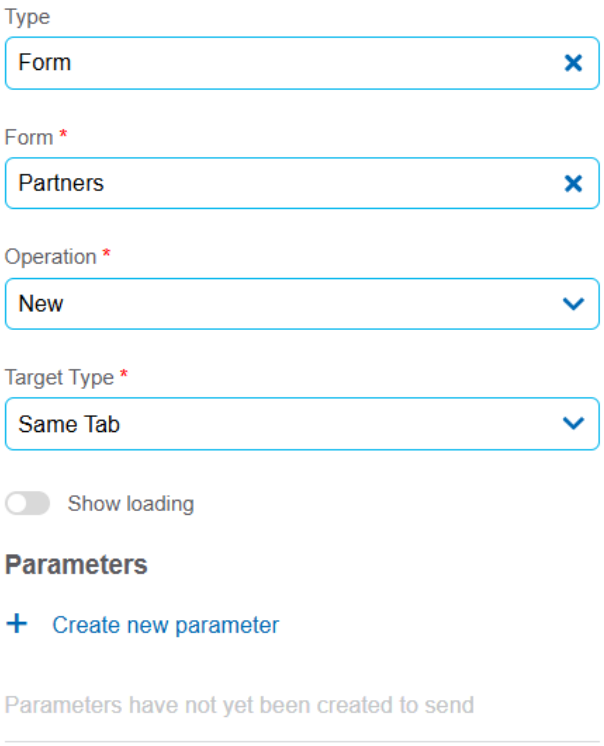
Destination Type
The available options for opening the object are displayed.

Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  Modal Horizontal Size
  Defines its width.
  Modal Vertical Size
  Defines its height.
- Iframe: selecting this option enables an additional property.
  Destination Iframe
  Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Forms



Forms
The forms modeled in the environment are displayed.

Operation
Defines the operation made when selecting the object.

Possible Values

- Create: Indicates that the panel of the object selected in the previous property is opened for the creation of an instance.
- Update: Indicates that the panel of the object selected in the previous property is opened for the update of an instance.
- Show: Indicates that the panel of the object selected in the previous property is opened for the query of an instance.
- Grid: Indicates that the grid of the object selected in the previous property is opened.

Destination Type
The available options for opening the object are displayed.

Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  - Modal Horizontal Size
    Defines its width.
  - Modal Vertical Size
    Defines its height.
- Iframe: selecting this option enables an additional property.
  - Destination Iframe

Expands the iframes previously defined on the object.

- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

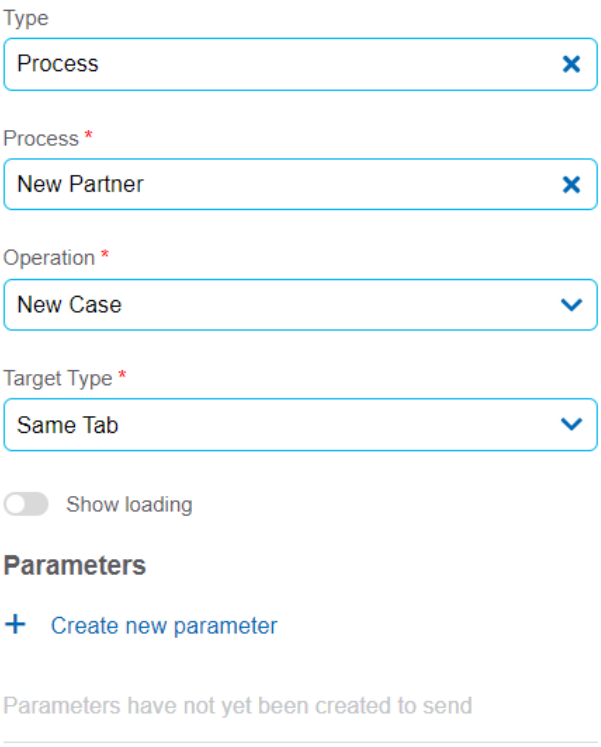
Show loading

Shows the loading icon and disables the user interaction until the redirection is over.

Parameters

Allows sending parameters to the selected object type.

Process



Process

The processes modeled in the environment are displayed.

Operation

Defines the operation made when selecting the object.

Destination Type

The available options for opening the object are displayed.

Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.

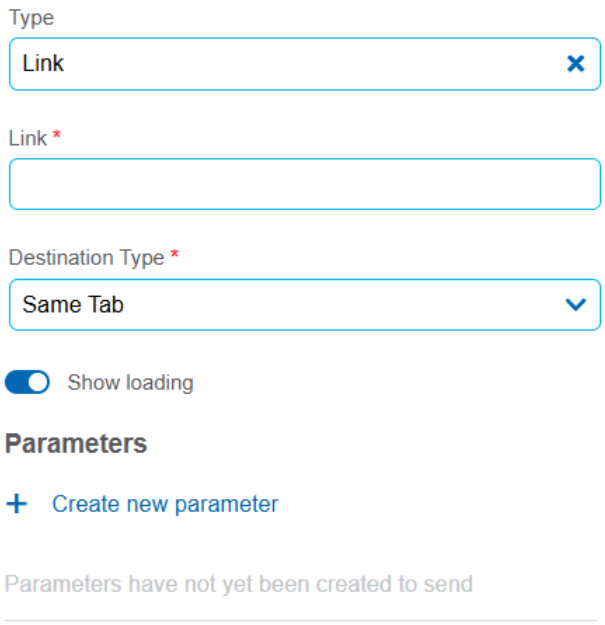
Modal Horizontal Size
Defines its width.
Modal Vertical Size
Defines its height.

- Iframe: selecting this option enables an additional property.
  Destination Iframe
  Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Parameters
Allows sending parameters to the selected object type.

Link



Link
Allows to enter any link.

Destination Type
The available options for opening the object are displayed.

Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  Modal Horizontal Size

Defines its width.

Modal Vertical Size

Defines its height.

- Iframe: selecting this option enables an additional property.

  Destination Iframe

  Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

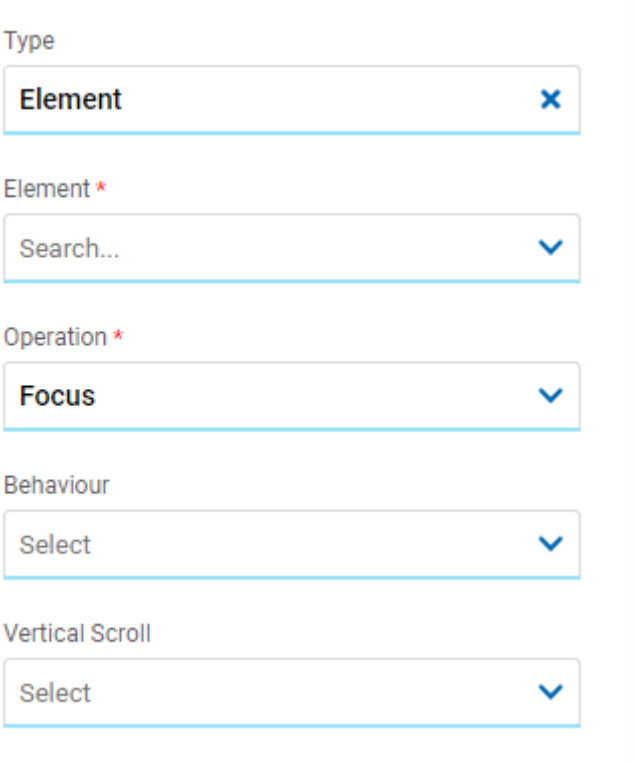
Show loading

Shows the loading icon and disables the user interaction until the redirection is over.

Parameters

Allows sending parameters to the selected object type.

Element



Element

The modeled elements in the object are displayed.

Operation

Defines the operation made when selecting the element.

Possible Values

- Focus
- Show
- Hide

- Show/Hide

Behaviour
Establishes the transition of the animation. It is only modelable if the "Focus" operation was selected.

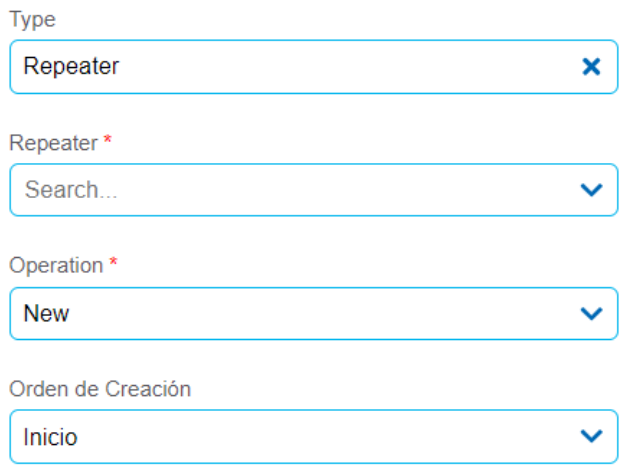
Possible Values

- Auto
- Smooth

Vertical Scroll
It is only modelable if the "Focus" operation was selected.
Possible Values

- Start: Moves to the start of the selected element.
- Closest: Moves to the closest position to the selected element from the element the event fires.
- Center: Moves to the center of the selected element.
- End: Moves to the end of the selected element.

Repeater



Repeater
The modeled repeaters from the object are displayed.

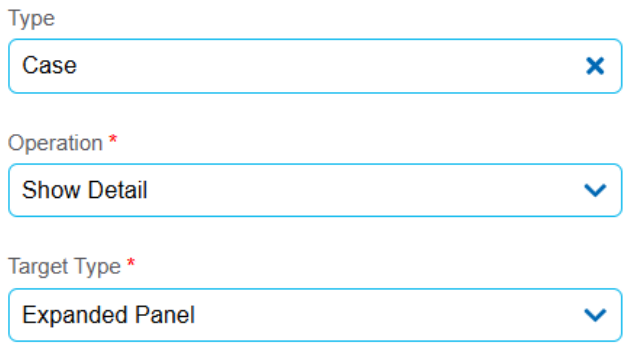
Operation
Defines the operation made when selecting the element.

Possible Values

- New: Allows for the creation of an item within the repeater. If this option is selected the additional properties are enabled.
  Creation Order
  It indicates in which position the new item of the container is added.
  Possible Values
  - Home
  - End

- Delete: Allows for the deletion of an item within the repeater.

Case



Case
Show case information.

Operation
Defines the operation made when selecting the element.

Possible Values

- Show Detail
- Show Graphic Execution

Destination Type
The available options for opening the object are displayed.

Possible Values

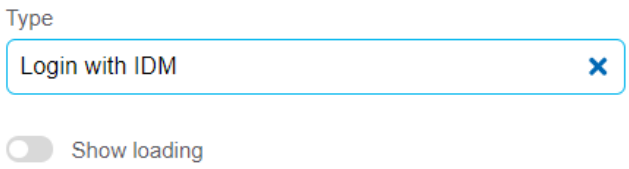
- Expanded Panel
- Modal: if this option is selected the additional properties are enabled.
  Modal Horizontal Size
  Define its width.
  Modal Vertical Size
  Define its height.

Back



It allows associating the event to go back in the browser to the element.

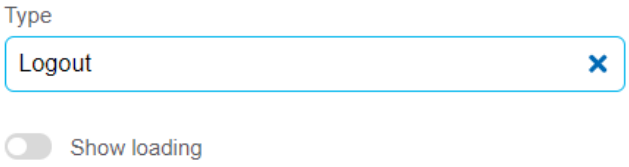
LogIn with IDM



Allows login with IDM.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

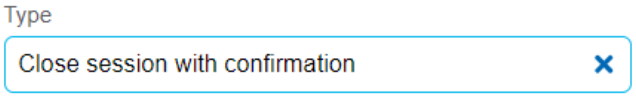
Logout



Allows logging the user out.

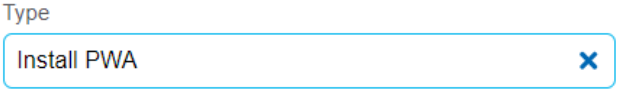
Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Close Session with Confirmation



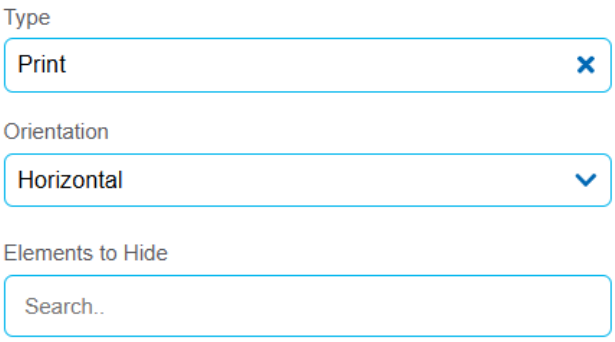
Displays a confirmation modal to logout the user.

Install PWA



Allows to install the application in the browser.

Print



This type of hyperlink associated with an element allows printing the content of the entity or page that is being executed.

Orientation

Determines the layout of the content to be printed.

Possible Values

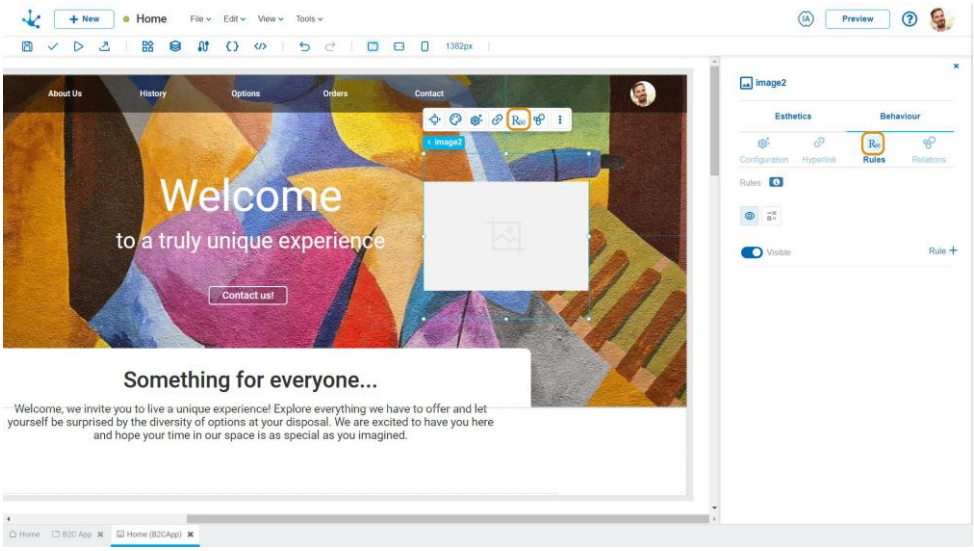
- Horizontal
- Vertical

Elements to hide

It allows selecting which elements of the entity or page should not be included in the printout.

Rule Properties

The rule properties panel of an element opens when clicking the icon R(x) of its context menu.



Properties

Rules

Embedded rules on behavior, validation and calculation can be defined, associated with an element, by using the wizard (ctrl + space).

Shows syntax examples for writing the rules.

Visible

Indicates whether the element is visible. If this property is not checked, the element is not displayed in the page.

Visible (default) Not visible

Rule + Opens an edit area where a rule to determine the visibility condition can be defined. If a rule is defined, the icon is displayed with light blue borders.

Calculation

Rule + Opens an edit area where you can define the expression to be executed to calculate the element value. If a rule is defined, the icon is displayed with light blue borders.

Operations when defining the rule:

Saves the new or modified rule

Cancels the operation

Operations once the rule is defined:

Edits the existing rule

Deletes the rule

Events

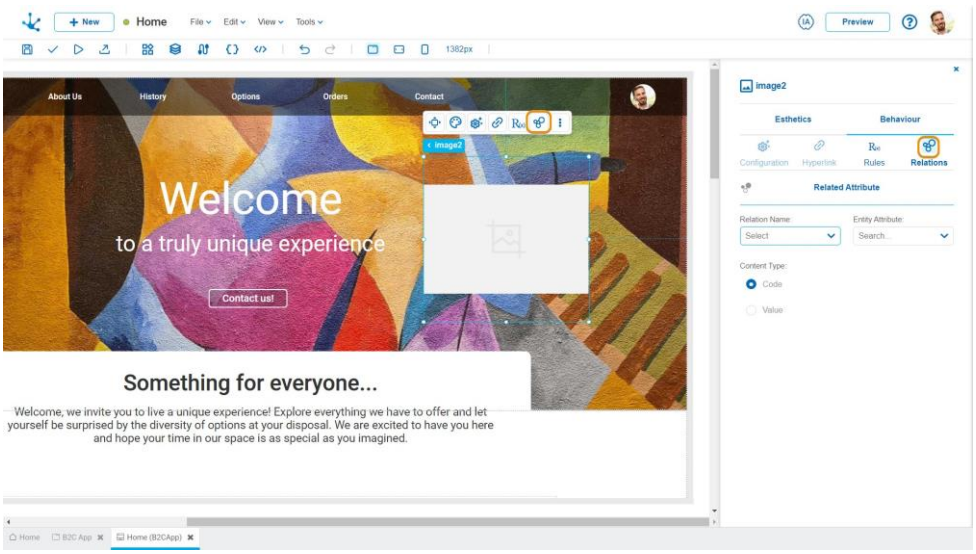
Images allow the use of different events.

Event	Description
onMouseIn()	It is executed when the cursor is positioned on the element.
onMouseOut()	It is executed when the cursor moves out of the element.
onClick()	It is executed when clicking on the element.
onDoubleClick()	It is executed when clicking twice on the element.
onInit()	It is executed before the element is loaded.

Event	Description
afterViewInit()	It is executed after viewing the element.
onViewportEnter()	It is executed when the element is visible.
onViewportLeave()	It is executed when the element is no longer visible.

Relation Properties

The relation properties panel of an element opens when clicking the icon of its context menu.



Related Attribute

A related attribute indicates that the field value is retrieved from the attribute value of another entity. One of the relations defined on the page must be selected, and then it should be specified which attribute of the related entity it links to.

A field can be modeled as a related attribute and it can also have a relation defined to an entity, a value list, or a rule. This allows the value of the related attribute to be retrieved on this field and the functionality provided by the relation to be used.

Properties

Relation Name
The list of relations defined on the page is displayed, allowing the selection of one of them.

Entity Attribute
The list of attributes of the selected entity compatible with the field type is displayed. Attributes can have a relation modeled with an entity, a value list, or a rule in the selected entity.

Content Type
The element content varies depending on whether the Code or Value property is checked when:

- The entity attribute has a modeled relation.
- The field has the option "Entity" selected in the Values Obtained from property.

Code
When this property is checked, if the relation is modeled with a:

- Related entity, the identifier of the entity is displayed.
- Value list, the value list code is displayed.
- Rule, the code returned by the rule's execution is displayed.

Value
When this property is checked, if the relation is modeled with a:

- Related entity, the short description of the entity is displayed.
- Value list, the value list description is displayed.
- Rule, the description returned by the rule's execution is displayed.

Video

Allows to add any multimedia content of video type.

The element properties are represented by icons in its context menu, where its operations are also available.

Subtypes

By selecting the "Video" option from the icon on the top toolbar, a list with the different subtypes of this element is displayed, which can be dragged to the modeling area. Each subtype has modeled element properties, predefined in a specific way.

- Small Video
- Medium Video
- Large Video

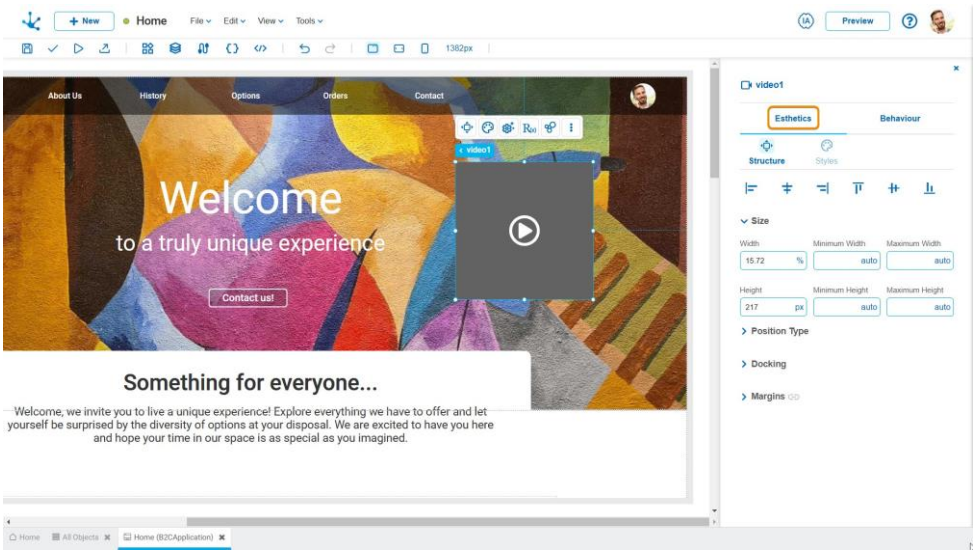
Classification of Properties

Properties are divided into two groups: esthetic properties and behavior properties.

Esthetic Properties

In the esthetic properties panel, the following are grouped:

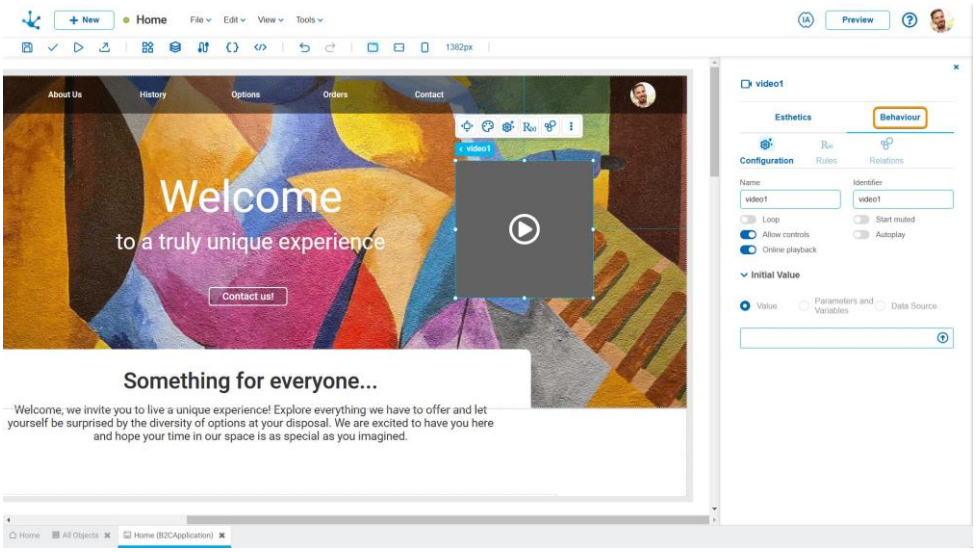
- Structure Properties
- Style Properties



Behavior Properties

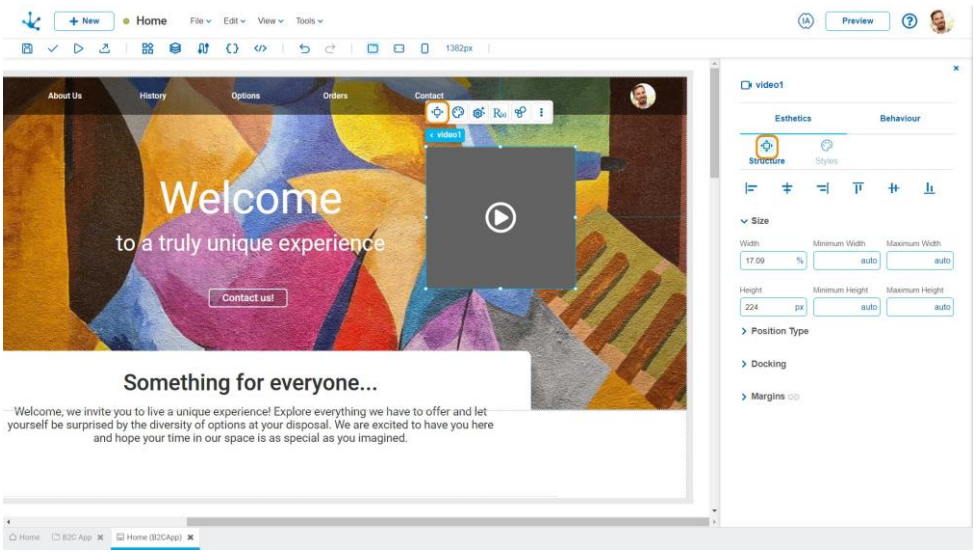
In the behavior properties panel, the following are grouped:

- Configuration Properties
- Rules Properties
- Relations Properties



Structure Properties

The structure properties panel of an element opens when selecting the icon  of its context menu.

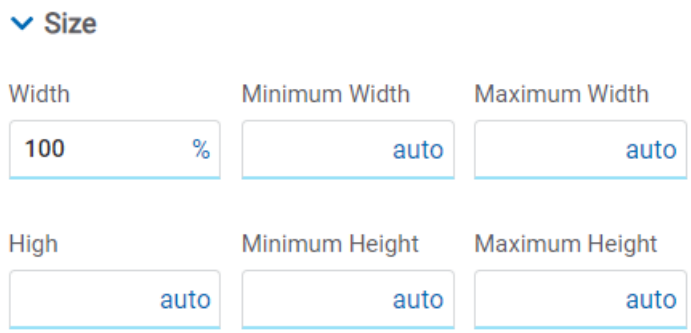


Alignment

Allows to align the element by selecting one of the available icons.

- Align to the left.
- Align to horizontal center.
- Align to the right.
- Align up.
- Align vertical center.
- Align bottom.

Size

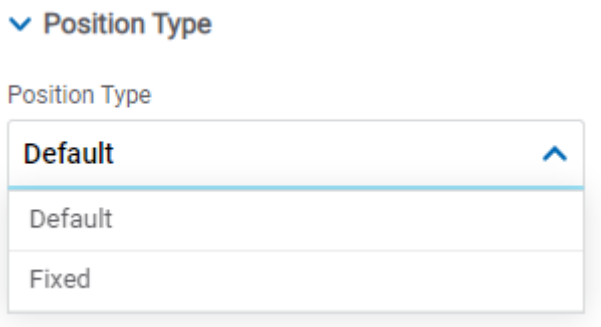


All size properties can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh). Additionally, for Width and Height properties, the “auto” option is added.

> *If selecting percentage (%) for the width and height properties, the size is calculated relative to the top element.*

Position Type

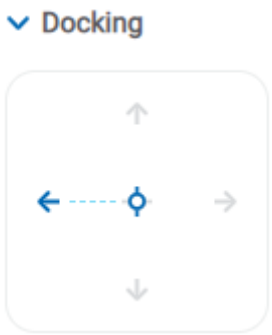
It determines if at the time of execution the element remains fixed on the page or if it moves as the scroll bar moves forward or backward.



Possible Values

- Default: The element has a relative position with respect to the superior element where it was placed (container or section).
- Fixed: Fixes the element to the page so that it is always visible in the same position. It allows to increase the visibility of important content.

Docking



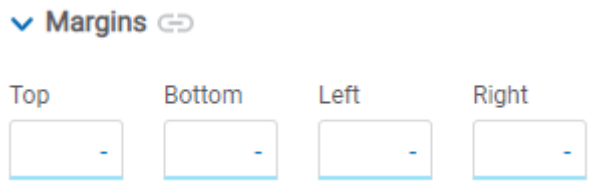
Determines the vertical and horizontal position of elements within the page section or container, when the screen is resized.

When an element is added or moved within the section or container, the modeler automatically docks it to the nearest corners or edges. It can also be configured by clicking on the arrows of the side to be docked to or on the center.

Once the element is docked, its exact position is indicated in the Margins property. The direction of the docking arrows determine the values that are automatically displayed for margins.

The docking position is indicated in the modeling area by dotted lines on the element.

Margins



It allows to define the distance of elements from the borders of their top element. The behavior of margins depends on the docking of the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Top
Distance to the top border of the highest ranking element.

Bottom
Distance to the bottom border of the highest ranking element.

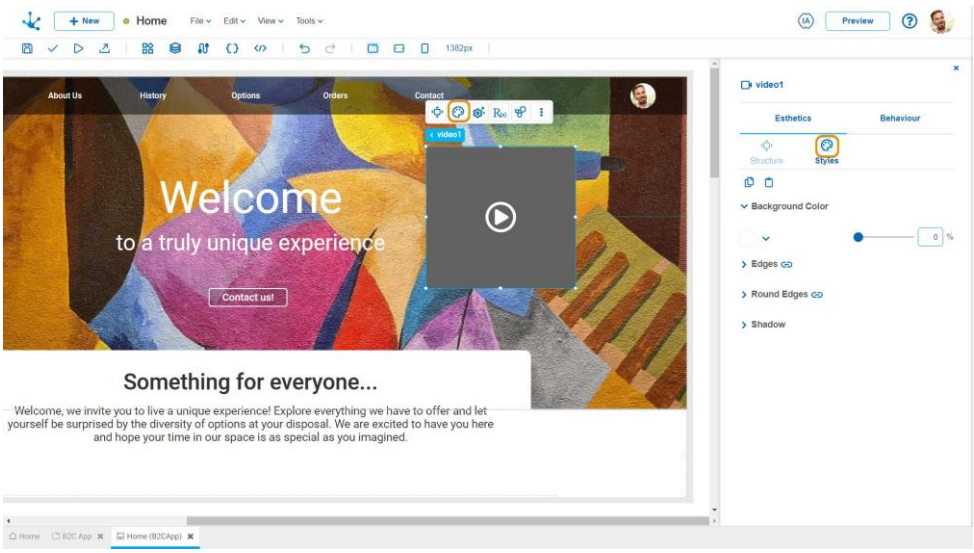
Left
Distance to the left border of the highest ranking element.

Right
Distance to the right border of the highest ranking element.

Allows the value entered in one of the margins to be copied to the other ones automatically.
Allows to indicate different values for each margin.

Style Properties

The style properties panel of an element opens when selecting the icon of its context menu.



Styles

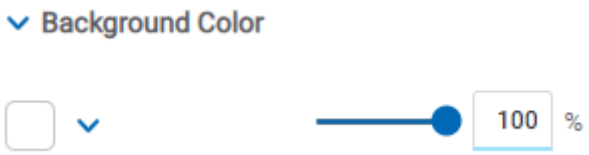
It allows copying the entire current configuration of the style panel for the selected element (Ctrl+Alt+C).

It allows applying the previously copied configuration to another element of the same type (Ctrl+Alt+V).

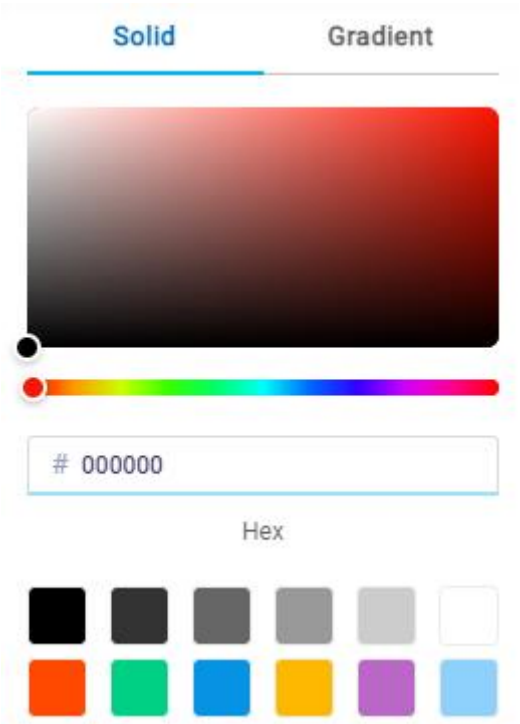
Background Color

It allows to model a background color for the element, it can be solid or gradient and different properties are defined for each one.

Solid



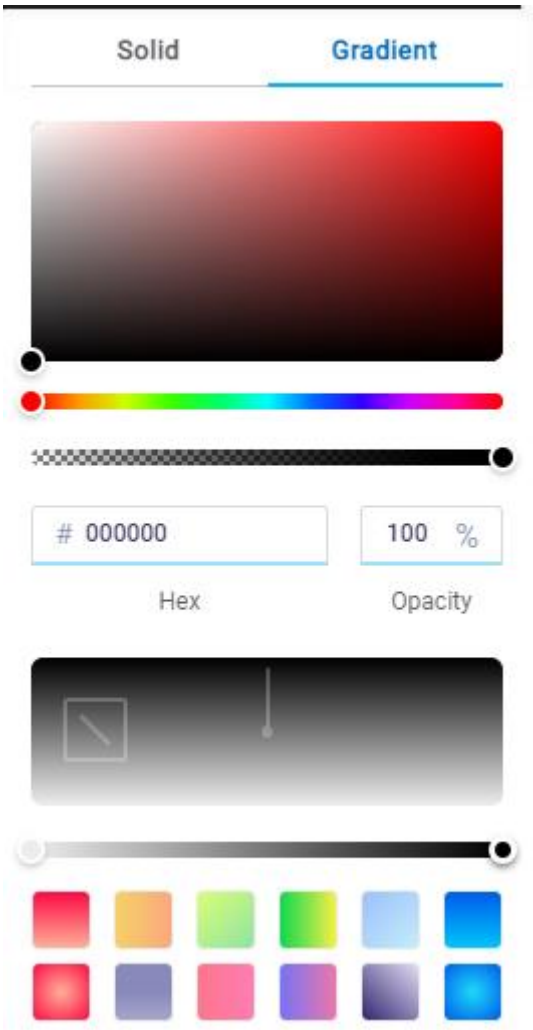
This option enables not only to select the color from a palette but also to define the degree of transparency.



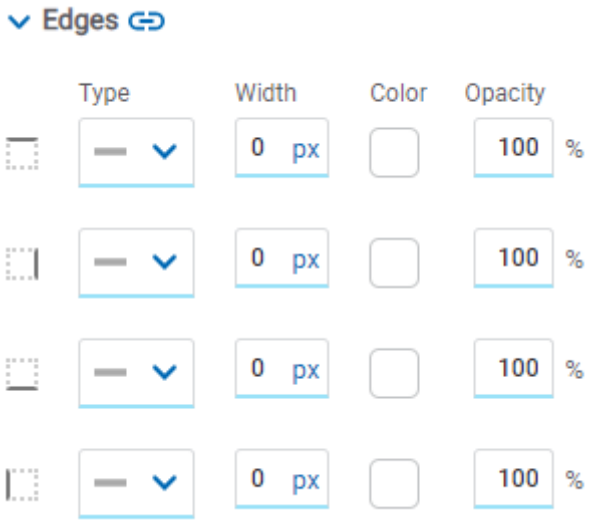
Gradient



This option enables to select not only the color from a palette but also to define its opacity and intensity according to the selected angle.



Edges



It allows to define the style of borders. Each one has its type, width, color and opacity defined, the latter as a percentage.

Allows the value entered in one of the borders to be copied to the other ones automatically.
Allows to indicate different values for each border.

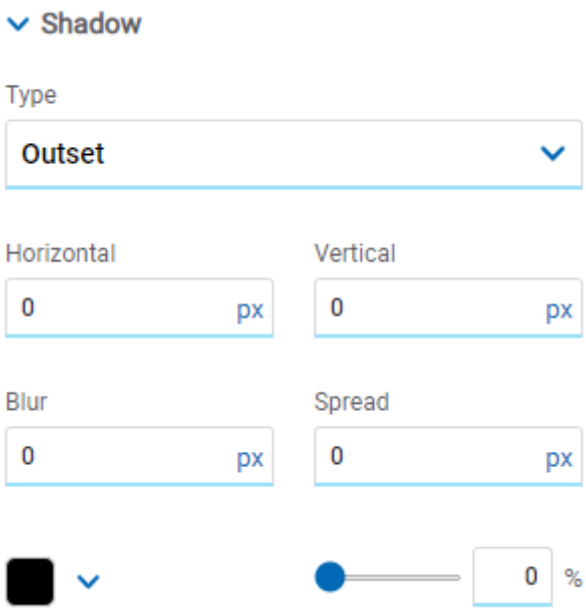
Round Edges



It allows to define the round edges at the corners of elements.
It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Allows the value entered in one of the borders to be copied to the other ones automatically.
Allows to indicate different values for each border.

Shadow



Allows to define a shadow effect around the element.

Type

Possible Values

- Outset
- Inset

Horizontal

Horizontal size of the shadow to the right of the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Vertical

Vertical size of the shadow below the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Blur

The greater this value is, the greater and lighter the shadow becomes. If not specified, its value is 0 and the shadow border is darker. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Spread

Positive values will cause the shadow to expand and grow bigger, while negative values will cause the shadow to shrink. If not specified, its value is 0 and the shadow will have the same size as the element.

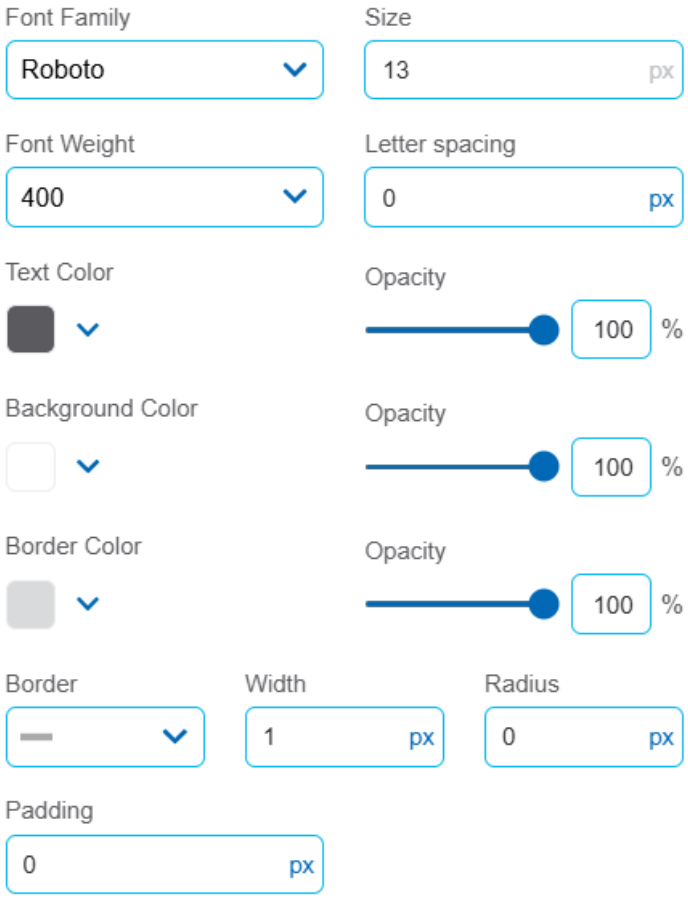
Color

Defines the color of the shadow.

Transparency

Defines the transparency of the shadow.

Tooltip



Source

It allows customizing the font, size, thickness and letter spacing of the pop-up message.

Colors

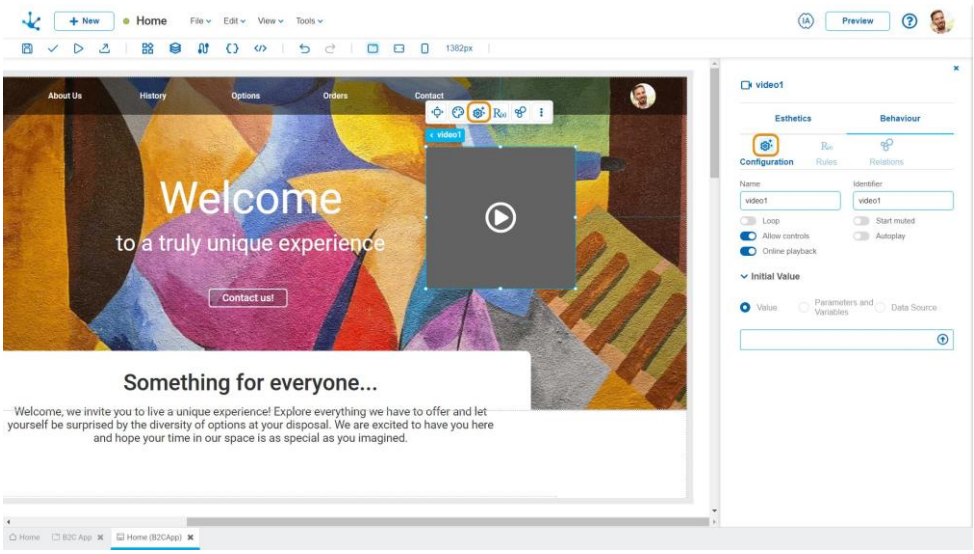
The color and opacity of the text, background and border can be defined. Borders are set uniformly for all sides of the tooltip.

Border

The border type, width, and border radius can be configured.

Configuration Properties

The configuration properties panel of an element opens when clicking the icon  of its context menu.



Name
Name used to reference the element during modeling.

Identifier
Uniquely identifies the element. It is used in the Javascript SDK.

Tooltip
Pop-up message that appears when hovering over an element, providing additional information.



Text
Content of the message being displayed.

Direction
Position of the message in relation to the element (top, bottom, left, or right).

Loop
It allows the video to play on a continuous loop. When the video ends, it automatically restarts from the beginning and continues playing indefinitely.

Start muted
It allows the video to start without sound. The user can enable audio manually if desired.

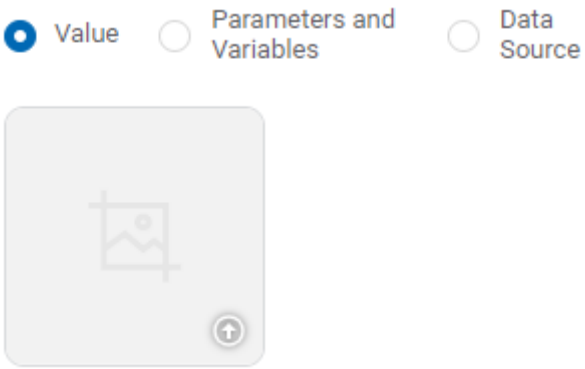
Allow controls
Enables video playback controls such as play and pause buttons, progress bar, volume, and fullscreen option.

Autoplay
It allows the video to start playing automatically as soon as the page is loaded. It can be combined with the Start muted property to prevent unexpected audio playback.

Online playback
It allows the video to play online within the page, rather than opening in a fullscreen player. It is especially useful on mobile devices, where videos often play full screen by default.

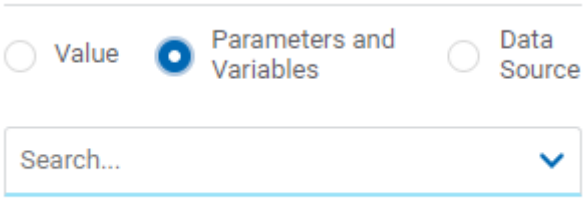
Initial Value

Allows to select the source of the element's content.



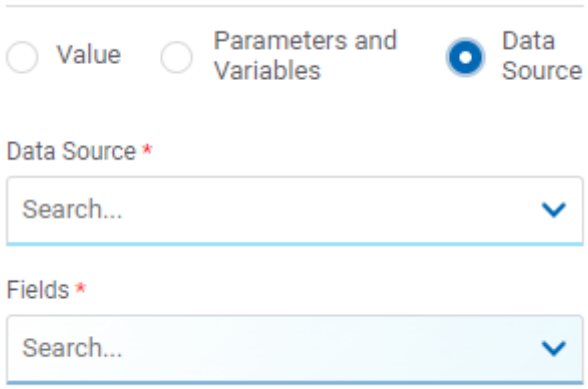
It allows adding a video from a file on the user's computer.

Parameters and Variables



Parameters and Variables
Allows to select a parameter or a variable from, whose value is displayed in the element.
This value can be a valid url.

Data Source

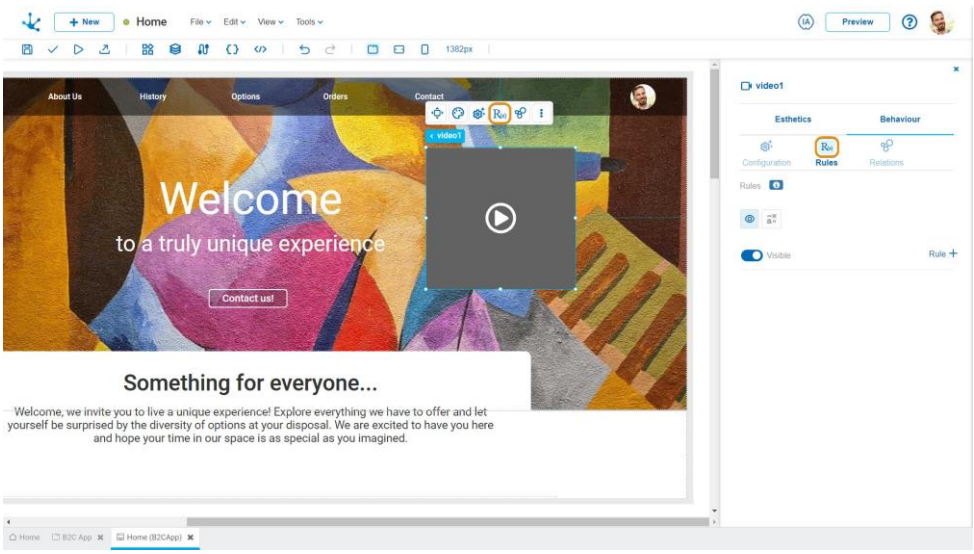


Data Source
It allows selecting a previously defined data source within the object.

Fields
It allows selecting a field from the data retrieved in the chosen data source, whose value is displayed in the element.

Rule Properties

The rule properties panel of an element opens when clicking the icon Rx of its context menu.



Properties

Rules
Embedded rules on behavior, validation and calculation can be defined, associated with an element, by using the wizard (ctrl + space).

Shows syntax examples for writing the rules.

Visible

Indicates whether the element is visible. If this property is not checked, the element is not displayed in the page.

Visible (default) Not visible

Rule + Opens an edit area where a rule to determine the visibility condition can be defined. If a rule is defined, the icon is displayed with light blue borders.

Calculation

Rule + Opens an edit area where you can define the expression to be executed to calculate the element value. If a rule is defined, the icon is displayed with light blue borders.

Operations when defining the rule:

- Saves the new or modified rule
- Cancels the operation

Operations once the rule is defined:

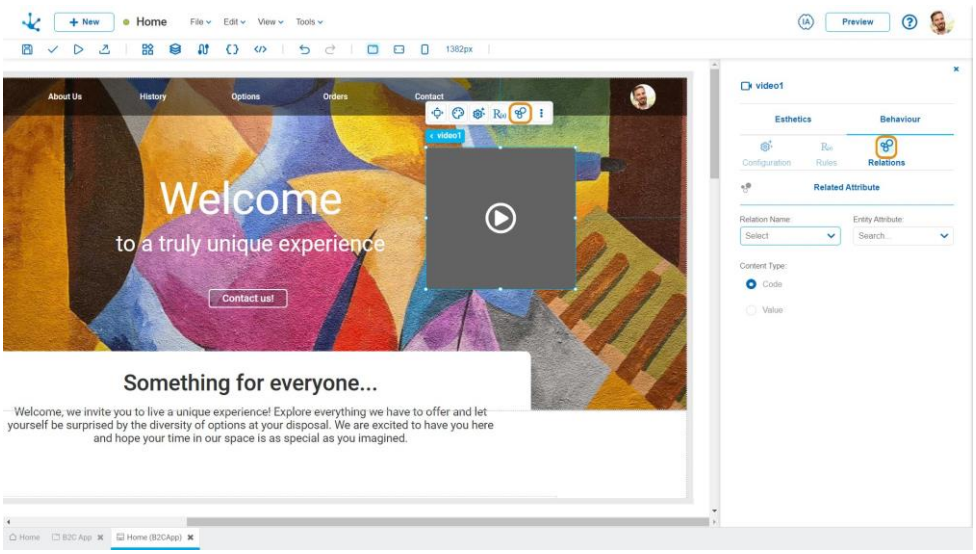
- Edits the existing rule
- Deletes the rule

Events

Videos allow the use of different events.

Event	Description
onMouseIn()	It is executed when the cursor is positioned on the element.
onMouseOut()	It is executed when the cursor moves out of the element.
onClick()	It is executed when clicking on the element.
onDoubleClick()	It is executed when clicking twice on the element.
onInit()	It is executed before the element is loaded.
afterViewInit()	It is executed after viewing the element.
onViewportEnter()	It is executed when the element is visible.
onViewportLeave()	It is executed when the element is no longer visible.

The relation properties panel of an element opens when clicking the icon of its context menu.



Related Attribute

A related attribute indicates that the field value is retrieved from the attribute value of another entity. One of the relations defined on the page must be selected, and then it should be specified which attribute of the related entity it links to.

A field can be modeled as a related attribute and it can also have a relation defined to an entity, a value list, or a rule. This allows the value of the related attribute to be retrieved on this field and the functionality provided by the relation to be used.

Properties

Relation Name

The list of relations defined on the page is displayed, allowing the selection of one of them.

Entity Attribute

The list of attributes of the selected entity compatible with the field type is displayed. Attributes can have a relation modeled with an entity, a value list, or a rule in the selected entity.

Content Type

The element content varies depending on whether the Code or Value property is checked when:

- The entity attribute has a modeled relation.
- The field has the option "Entity" selected in the Values Obtained from property.

Code

When this property is checked, if the relation is modeled with a:

- Related entity, the identifier of the entity is displayed.
- Value list, the value list code is displayed.

- Rule, the code returned by the rule's execution is displayed.

Value

When this property is checked, if the relation is modeled with a:

- Related entity, the short description of the entity is displayed.
- Value list, the value list description is displayed.
- Rule, the description returned by the rule's execution is displayed.

Layout

This element is a structure built of a set of Items and allows to structure them based on different styles such as column, rows, mosaic and stripe. Such items are containers, which allows to incorporate them to other elements. These elements can be moved along with each item when the layout structure is resized.

The page modeler also includes other structural elements that facilitate the fields modeling by speeding up the modeling of a page, without worrying about the aesthetic aspects of the fields. All aesthetic aspects of these elements are defined by default to adapt to different breakpoints.

Both groups of structural elements are used by subtype selection.

The element properties are represented by icons in its context menu, where its operations are also available.

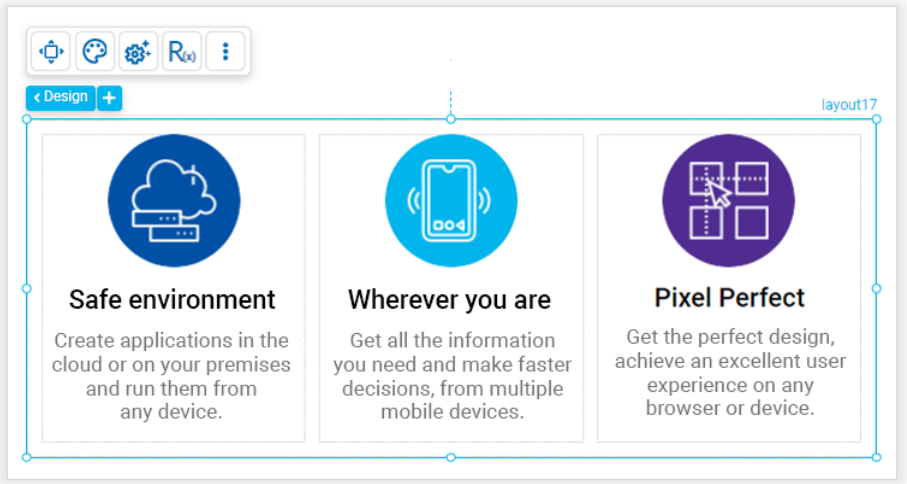
Subtypes

By selecting the "Layout" option from the icon on the top toolbar, a list with the different subtypes of this element is displayed, which can be dragged to the modeling area. Each subtype has modeled element properties, predefined in a specific way.

- Row
- Column
- Stripe
- Mosaic
- Columns and Rows
- Rows and Columns
- Field Group
- Fields in Rows and Columns

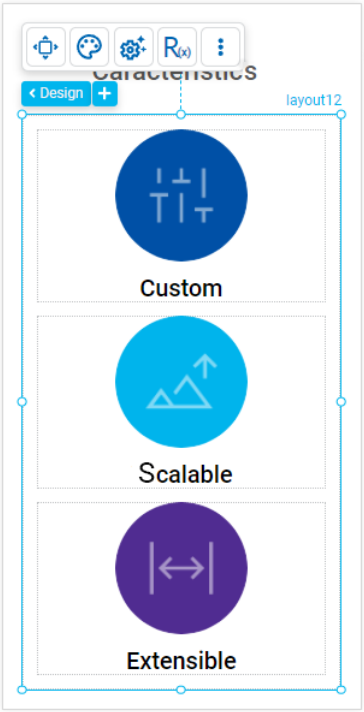
Row

It allows to define items arranged in rows.Within each row item, any type of element can be dragged.



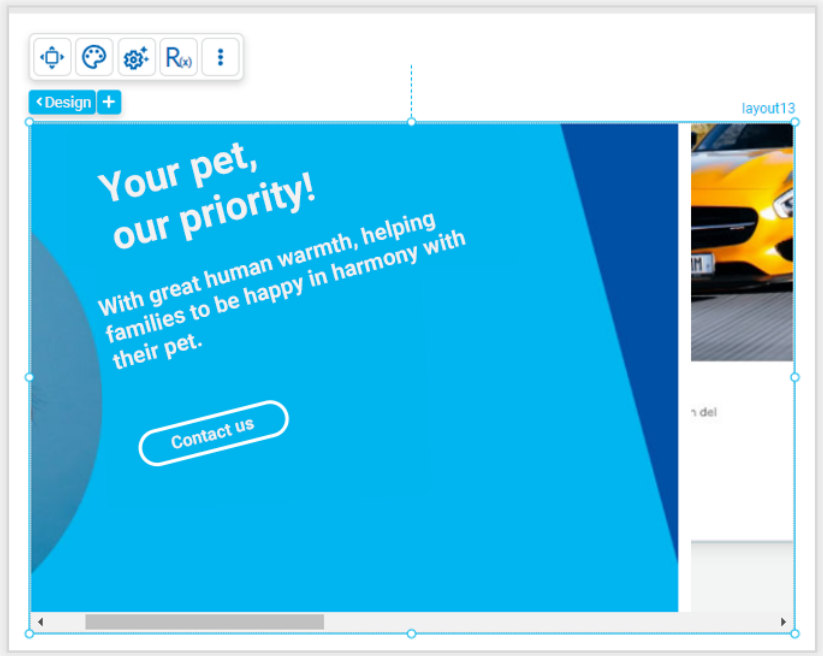
Column

It allows to define items arranged in columns. Within each column item, any type of element can be dragged.



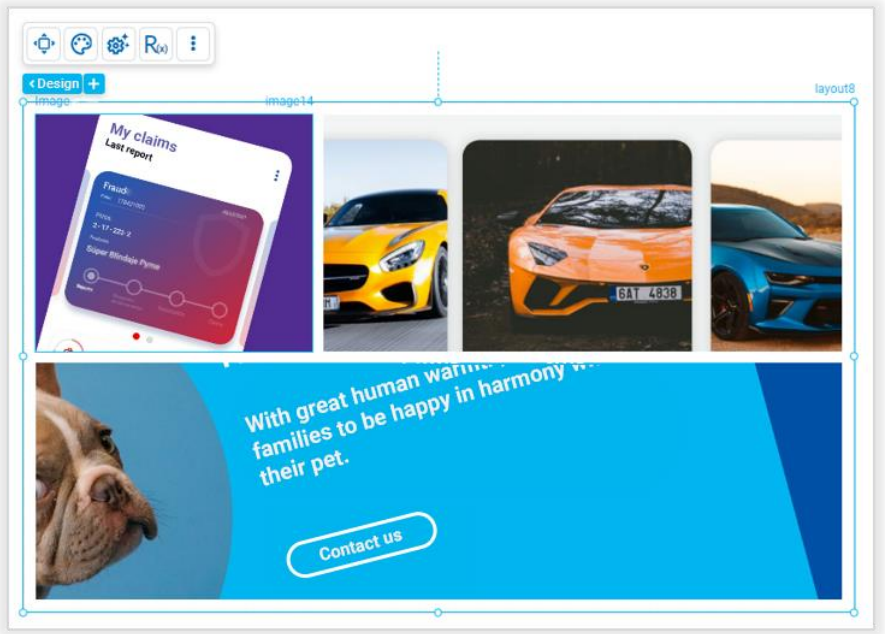
Stripe

It allows to define items arranged in a horizontal strip that covers the entire available width, incorporating a scroll bar to move. Within each stripe item, any type of element can be dragged.



Mosaic

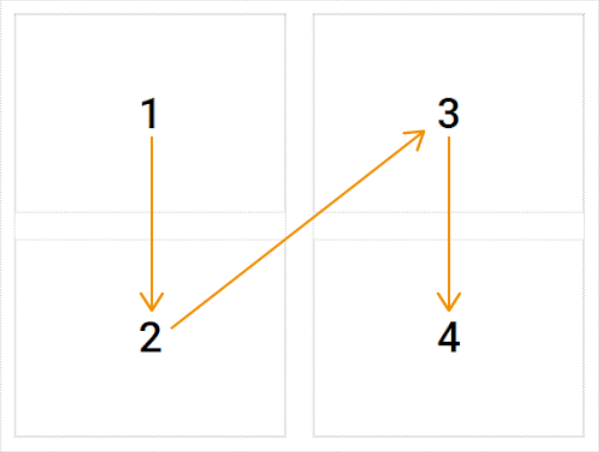
It allows to define items arranged in a grid, within each item any type of element can be dragged. Mosaics can be uniform or varied in size, depending on the desired style.



Columns and Rows

It allows organizing elements within cells, structured in columns and rows. Each cell is called a row column item and can contain any other element.

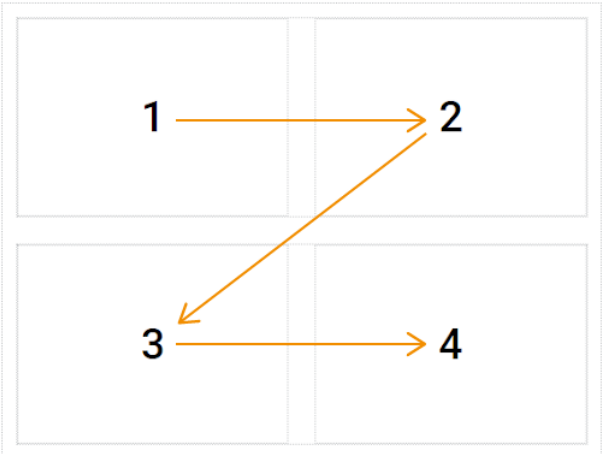
This layout subtype is especially useful when cells need to automatically adapt to the mobile breakpoint, stacking entire columns from left to right.



Rows and Columns

It allows organizing elements within cells, structured in rows and columns. Each cell is called a row column item and can contain any other element.

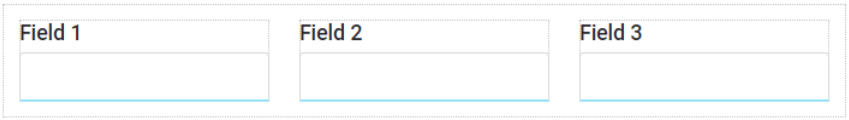
This layout subtype is especially useful when cells need to automatically adapt to the mobile breakpoint, stacking from left to right within each row.



Field Group

It allows grouping one or more fields, organizing them in a row or a column, as needed.

This layout subtype is especially useful when fields have to be grouped together, and easily relocated within other layouts.



Fields in Rows and Columns

It allows defining fields arranged in rows and columns. They can be modeled with one or multiple rows and with one or multiple columns. One or more fields can be defined in each cell.

This layout subtype is ideal for use directly on a section, especially when fields need to be arranged from left to right in each row, and to be arranged in vertical columns, top to bottom, across the entire screen. Additionally, in the mobile breakpoint, the layout adapts by stacking the content of each row from left to right, to ensure a clear display.



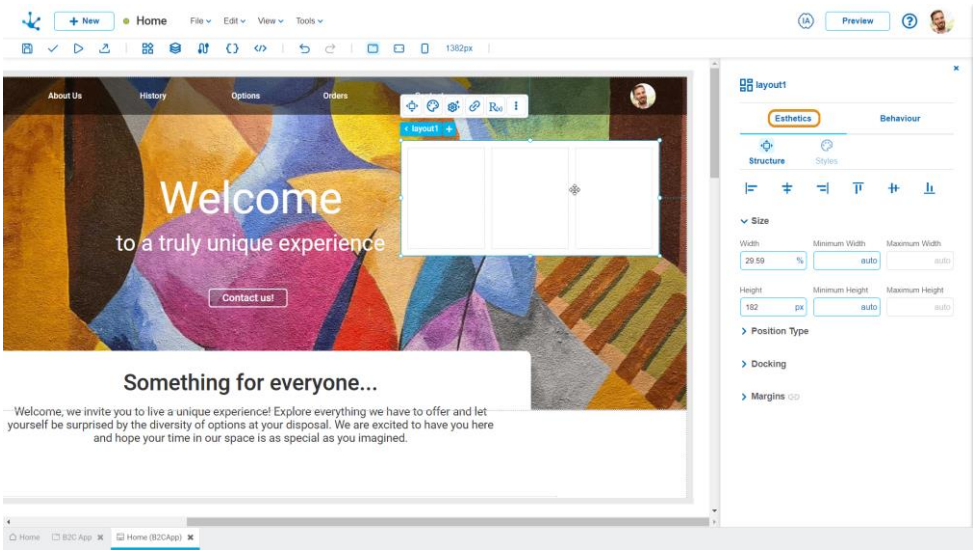
Classification of Properties

Properties are divided into two groups: esthetic properties and behavior properties.

Esthetic Properties

In the esthetic properties panel, the following are grouped:

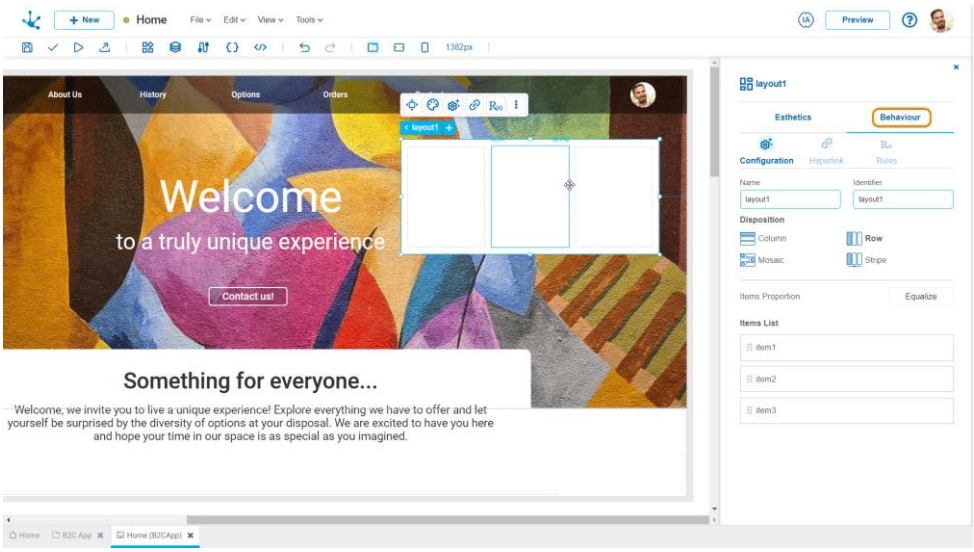
- Structure Properties
- Style Properties



Behavior Properties

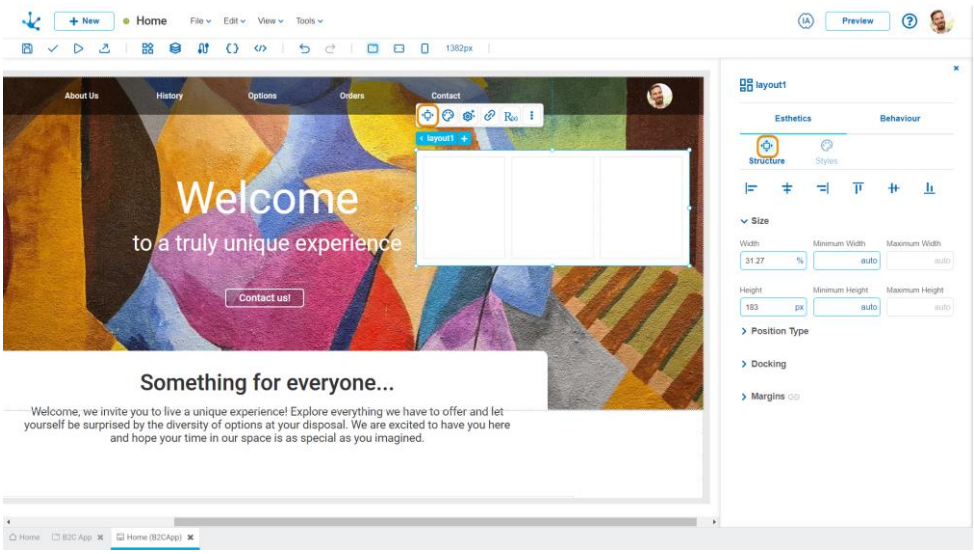
In the behavior properties panel, the following are grouped:

- Configuration Properties
- Hyperlink Properties
- Rules Properties



Structure Properties

The structure properties panel of an element opens when selecting the icon of its context menu.

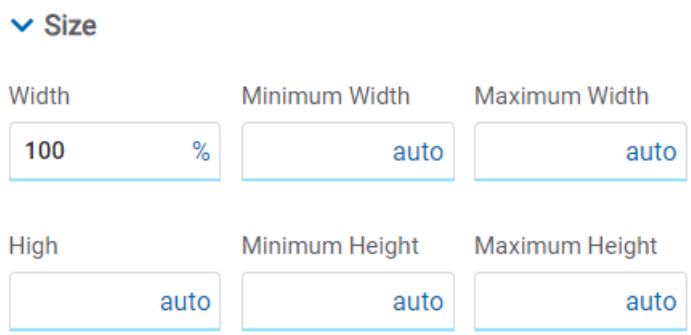


Alignment

Allows to align the element by selecting one of the available icons.

- Align to the left.
- Align to horizontal center.
- Align to the right.
- Align up.
- Align vertical center.
- Align bottom.

Size

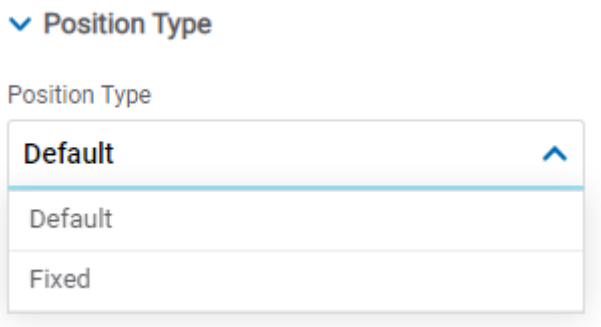


All size properties can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh). Additionally, for Width and Height properties, the “auto” option is added.

> *If selecting percentage (%) for the width and height properties, the size is calculated relative to the top element.*

Position Type

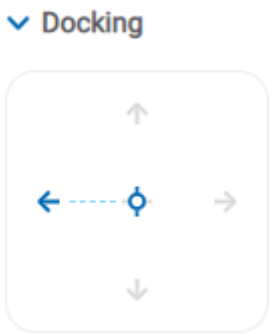
It determines if at the time of execution the element remains fixed on the page or if it moves as the scroll bar moves forward or backward.



Possible Values

- Default: The element has a relative position with respect to the superior element where it was placed (container or section).
- Fixed: Fixes the element to the page so that it is always visible in the same position. It allows to increase the visibility of important content.

Docking



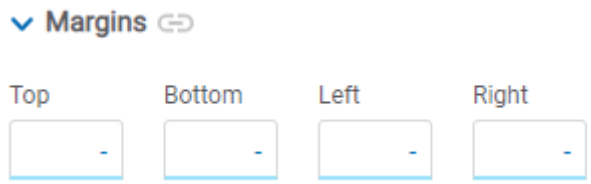
Determines the vertical and horizontal position of elements within the page section or container, when the screen is resized.

When an element is added or moved within the section or container, the modeler automatically docks it to the nearest corners or edges. It can also be configured by clicking on the arrows of the side to be docked to or on the center.

Once the element is docked, its exact position is indicated in the Margins property. The direction of the docking arrows determine the values that are automatically displayed for margins.

The docking position is indicated in the modeling area by dotted lines on the element.

Margins



It allows to define the distance of elements from the borders of their top element. The behavior of margins depends on the docking of the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Top
Distance to the top border of the highest ranking element.

Bottom
Distance to the bottom border of the highest ranking element.

Left
Distance to the left border of the highest ranking element.

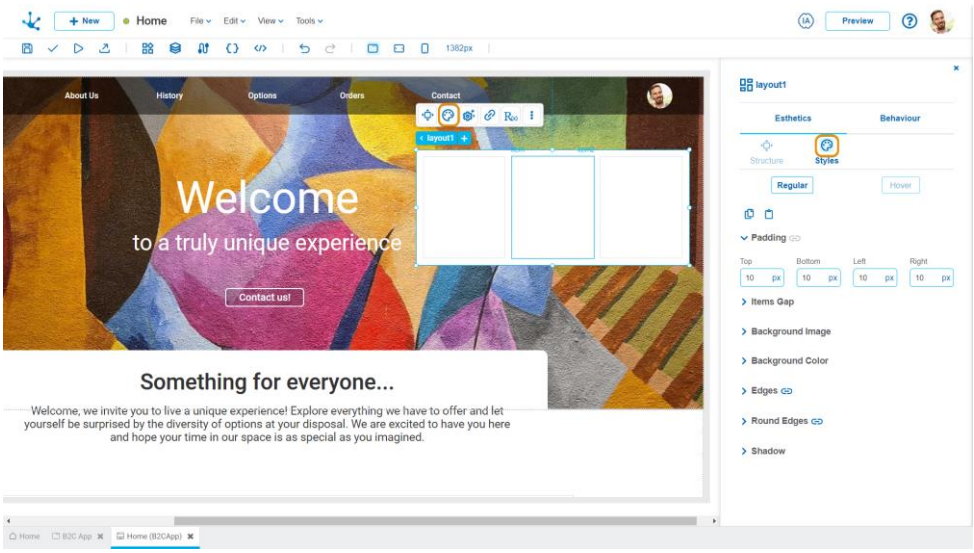
Right
Distance to the right border of the highest ranking element.

Allows the value entered in one of the margins to be copied to the other ones automatically.

Allows to indicate different values for each margin.

Style Properties

The style properties panel of an element opens when selecting the icon of its context menu.



This type of element may take different states and for each of them different values for its properties may be modeled.



- Regular: The mouse pointer is not over the element.
- Hover: The mouse pointer is over the element.

Styles

It allows copying the entire current configuration of the style panel for the selected element (Ctrl+Alt+C).

It allows applying the previously copied configuration to another element of the same type (Ctrl+Alt+V).

Padding

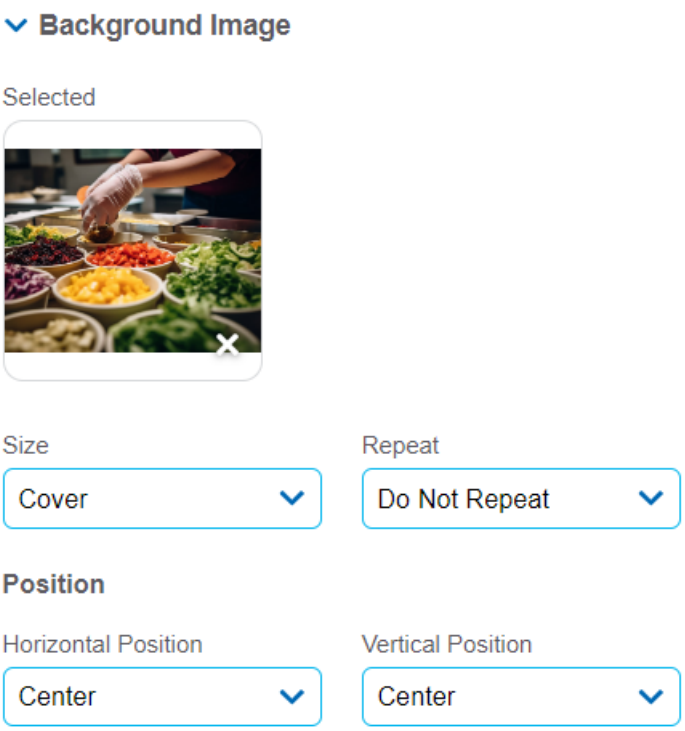


All padding space properties allow you to create a space around the borders (top, bottom, sides) and the bottom elements. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Allows values entered in one of the paddings to be copied to the other ones automatically.
Allows to indicate different values for each padding.

Items Gap
It defines the spacing between the items in the layout.

Background Image



Allows to add a background image to the element.

Selected

An image can be uploaded from the computer where it is being modeled.

Size

Possible Values

- Cover
- Content
- Auto

Repeat

Possible Values

- Repeat
- Repeat Hor.
- Repeat Vert.
- Spacing
- Rounded
- Do Not Repeat

Horizontal Position

Possible Values

- Center

- Left
- Right

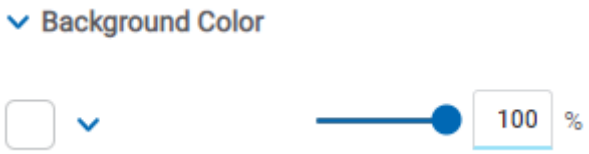
Vertical Position

- Center
- Top
- Botton

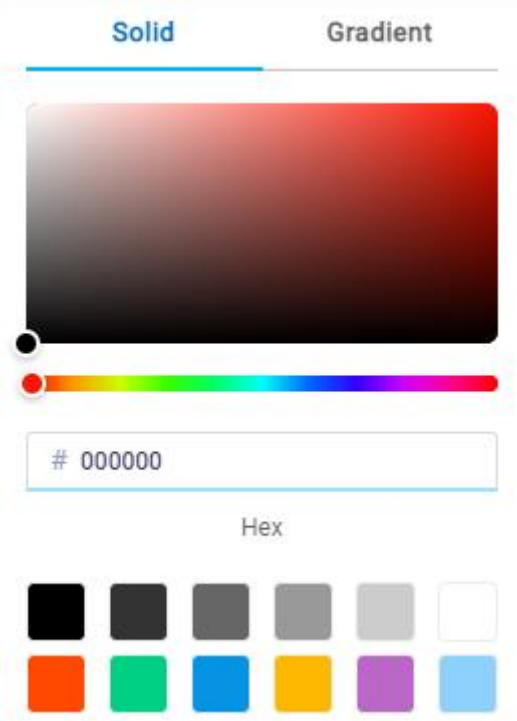
Background Color

It allows to model a background color for the element, it can be solid or gradient and different properties are defined for each one.

Solid



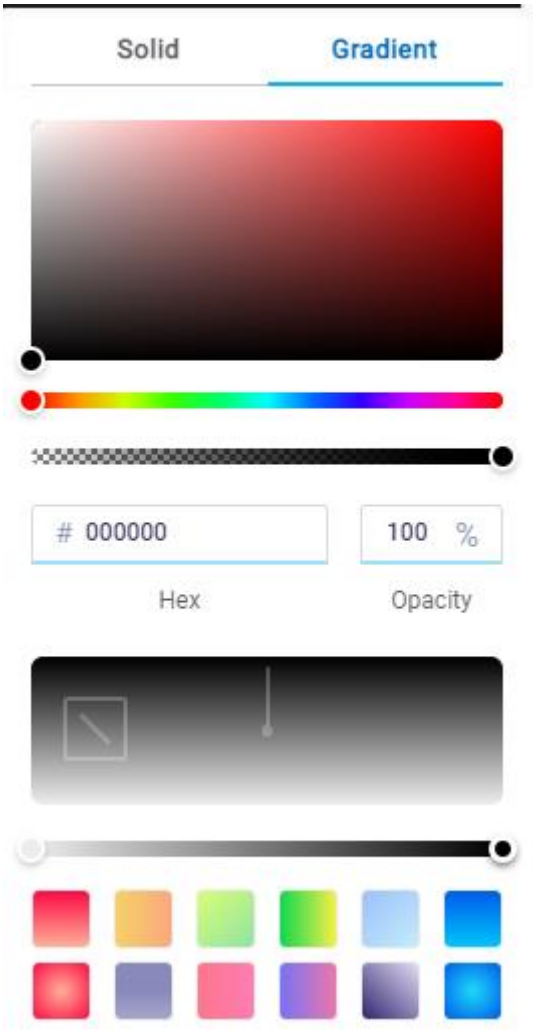
This option enables not only to select the color from a palette but also to define the degree of transparency.



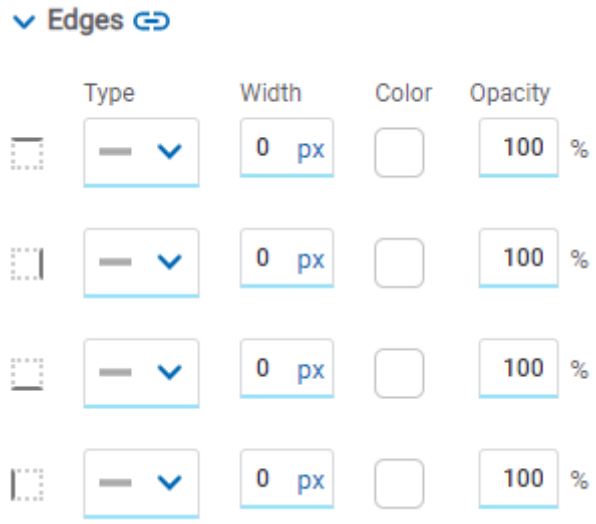
Gradient



This option enables to select not only the color from a palette but also to define its opacity and intensity according to the selected angle.



Edges



It allows to define the style of borders. Each one has its type, width, color and opacity defined, the latter as a percentage.

Allows the value entered in one of the borders to be copied to the other ones automatically.
Allows to indicate different values for each border.

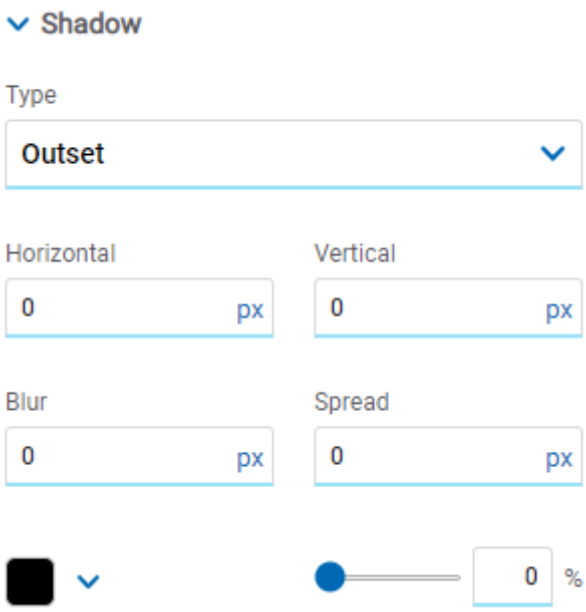
Round Edges



It allows to define the round edges at the corners of elements.
It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Allows the value entered in one of the borders to be copied to the other ones automatically.
Allows to indicate different values for each border.

Shadow



Allows to define a shadow effect around the element.

Horizontal
Horizontal size of the shadow to the right of the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Vertical
Vertical size of the shadow below the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

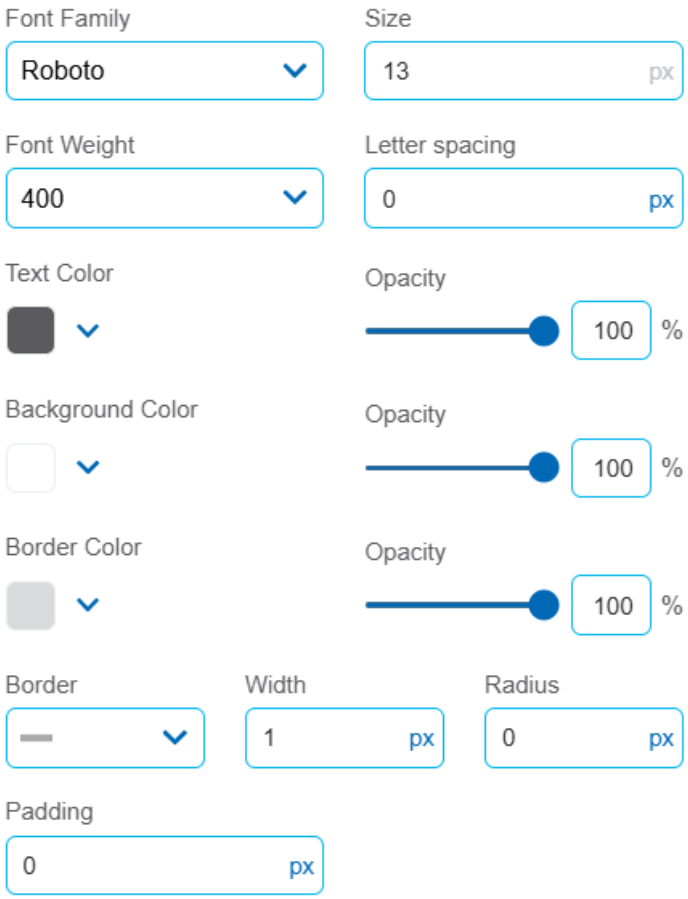
Blur
The greater this value is, the greater and lighter the shadow becomes. If not specified, its value is 0 and the shadow border is darker. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Spread
Positive values will cause the shadow to expand and grow bigger, while negative values will cause the shadow to shrink. If not specified, its value is 0 and the shadow will have the same size as the element.

Color
Defines the color of the shadow.

Transparency
Defines the transparency of the shadow.

Tooltip



Source

It allows customizing the font, size, thickness and letter spacing of the pop-up message.

Colors

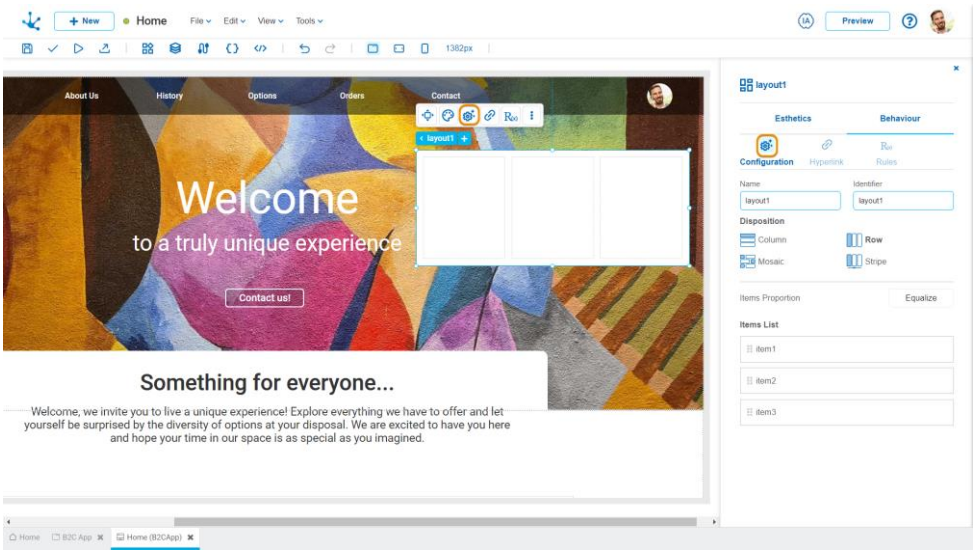
The color and opacity of the text, background and border can be defined. Borders are set uniformly for all sides of the tooltip.

Border

The border type, width, and border radius can be configured.

Configuration Properties

The configuration properties panel of an element opens when clicking the icon  of its context menu.



Name
Name used to reference the element during modeling.

Identifier
Uniquely identifies the element. It is used in the Javascript SDK.

Tooltip
Pop-up message that appears when hovering over an element, providing additional information.

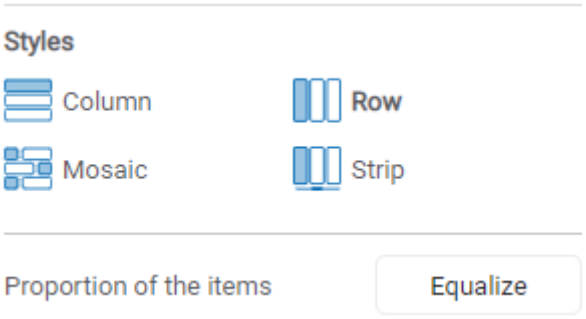


Text
Content of the message being displayed.

Direction
Position of the message in relation to the element (top, bottom, left, or right).

Styles

Indicates the way in which items are grouped within the layout.



Column
Items are placed vertically within the layout.

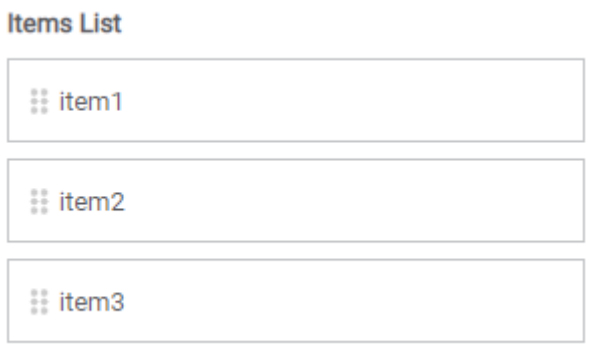
Row
Items are placed horizontally within the layout.

Mosaic
Combines the vertical and horizontal placement of items based on their size within the layout.

Strip
Items are placed horizontally within the layout. Allows scrolling.

Items Proportion
Pressing the “Equal” button adjusts the size of the items proportionally within the layout.

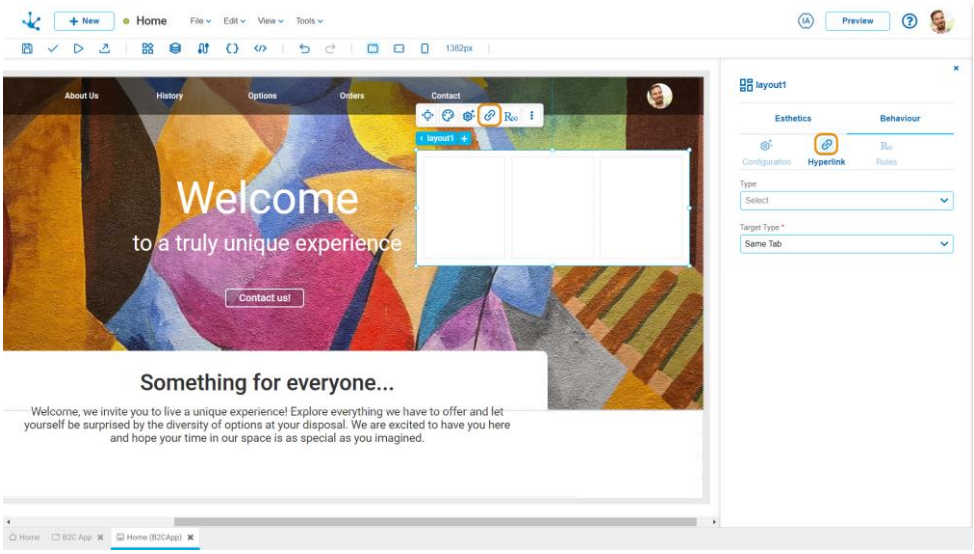
Items List



The items contained in the layout are displayed, and their position can be changed by dragging them within the list.

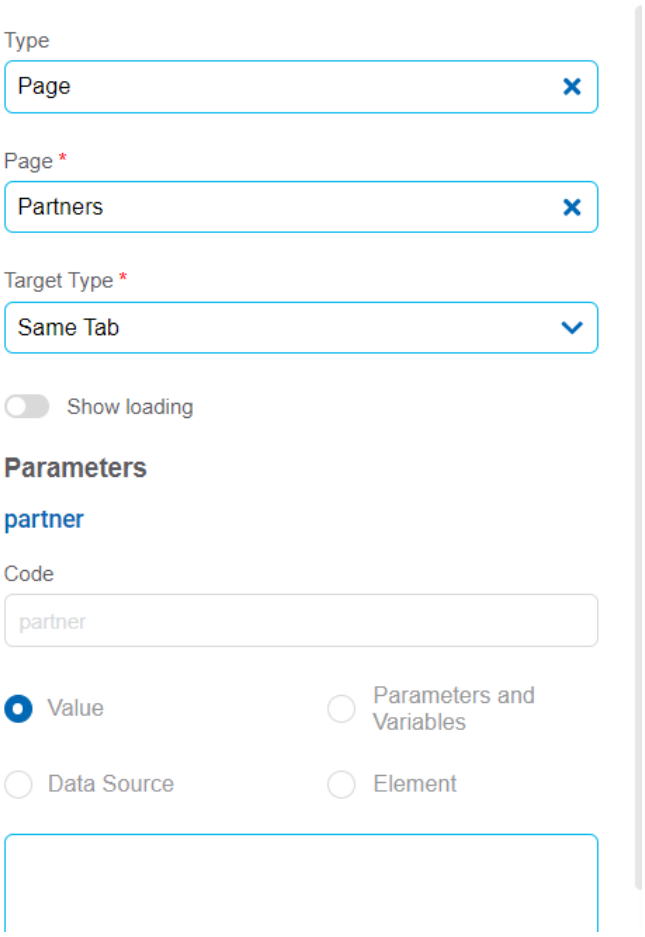
Hyperlink Properties

The hyperlink properties panel of an element opens when clicking the icon of its context menu.



Define the behavior of the element when selecting it. Different properties are enabled depending on the type of object selected.

Page



Page
The pages modeled in the environment are displayed.

Destination Type
The available options for opening the object are displayed.

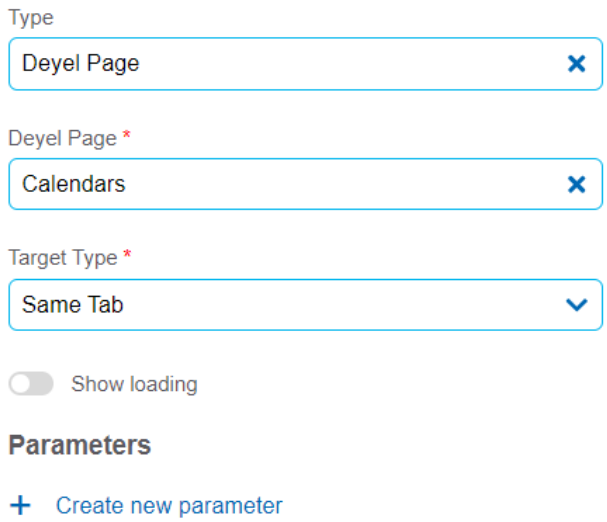
Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  - Modal Horizontal Size
    Defines its width.
  - Modal Vertical Size
    Defines its height.
- Iframe: selecting this option enables an additional property.
  - Destination Iframe
    Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Parameters
If there are parameters, their value can be specified with text, parameters and variables or data source.

Deyel Page



Deyel Page
The pages belonging to **Deyel** are displayed.

Destination Type
The available options for opening the object are displayed.

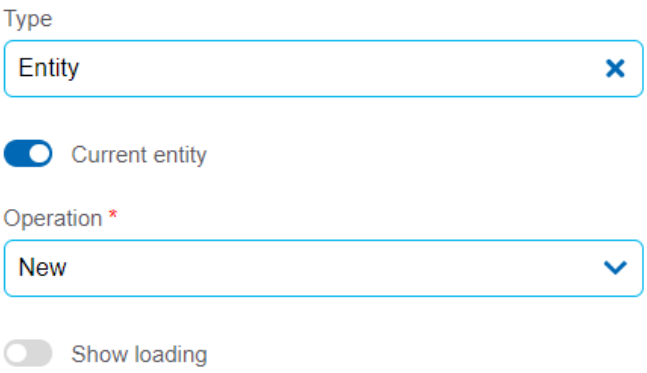
Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  Modal Horizontal Size
  Defines its width.
  Modal Vertical Size
  Defines its height.
- Iframe: selecting this option enables an additional property.
  Destination Iframe
  Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Parameters
Allows sending parameters to the selected object type.

Entity



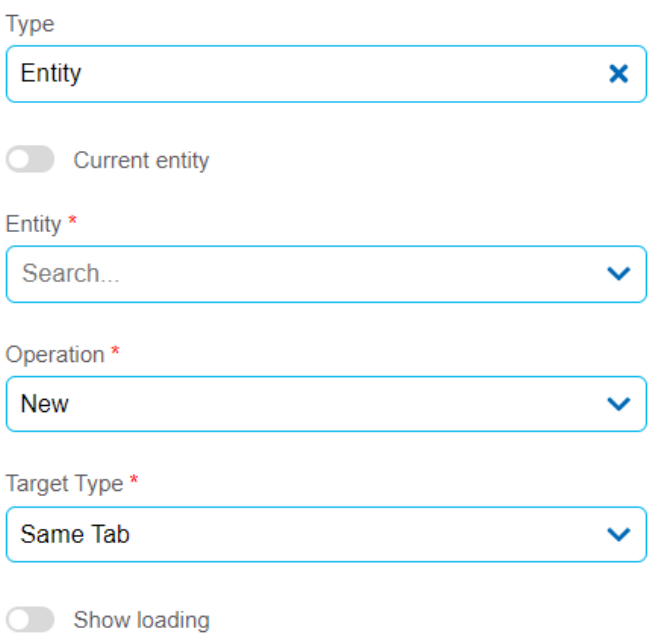
Current Entity
If this property is checked, the operation applies directly to the entity being modeled.

Operation
Defines the operation made when selecting the object.

Possible Values

- New: Indicates that the current entity panel is opened for the creation of an instance.
- Modify: Indicates that the current entity panel is opened for the update of an instance.

- Delete: Indicates that the current entity panel is opened for the deletion of an instance.



Entity
If the property Current Entity, is not checked, the entities modeled in the environment are displayed.

Operation
Defines the operation made when selecting the object.

Possible Values

- New: The panel of the entity selected in the previous property is opened to create an instance.
- Update The panel of the entity selected in the previous property is opened to update an instance.
- Show: The panel of the entity selected in the previous property is opened to show an instance.
- Grid: Indicates that the grid of the entity selected in the previous property is opened.

Destination Type
The available options for opening the object are displayed.

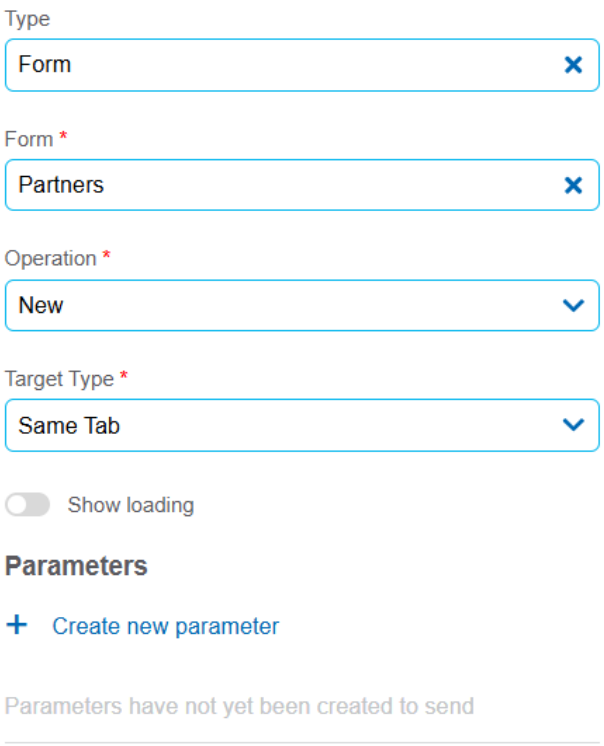
Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  Modal Horizontal Size
  Defines its width.
  Modal Vertical Size
  Defines its height.
- Iframe: selecting this option enables an additional property.
  Destination Iframe
  Expands the iframes previously defined on the object.

- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Forms



Forms
The forms modeled in the environment are displayed.

Operation
Defines the operation made when selecting the object.

Possible Values

- Create: Indicates that the panel of the object selected in the previous property is opened for the creation of an instance.
- Update: Indicates that the panel of the object selected in the previous property is opened for the update of an instance.
- Show: Indicates that the panel of the object selected in the previous property is opened for the query of an instance.
- Grid: Indicates that the grid of the object selected in the previous property is opened.

Destination Type
The available options for opening the object are displayed.

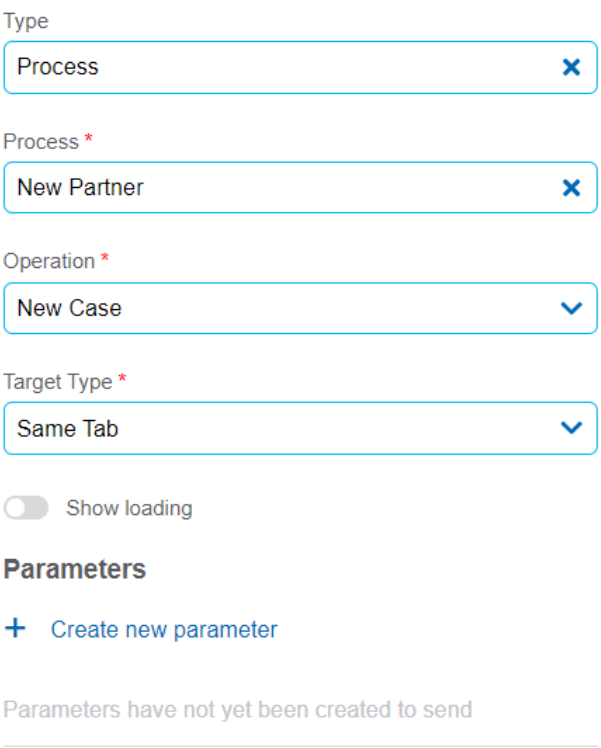
Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  Modal Horizontal Size
  Defines its width.
  Modal Vertical Size
  Defines its height.
- Iframe: selecting this option enables an additional property.
  Destination Iframe
  Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Parameters
Allows sending parameters to the selected object type.

Process



Process
The processes modeled in the environment are displayed.

Operation
Defines the operation made when selecting the object.

Destination Type
The available options for opening the object are displayed.

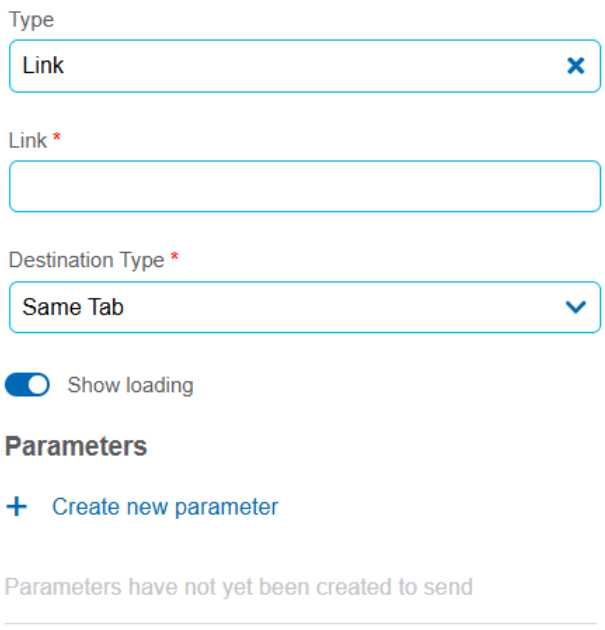
Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  - Modal Horizontal Size
    Defines its width.
  - Modal Vertical Size
    Defines its height.
- Iframe: selecting this option enables an additional property.
  - Destination Iframe
    Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Parameters
Allows sending parameters to the selected object type.

Link



Link

Allows to enter any link.

Destination Type
The available options for opening the object are displayed.

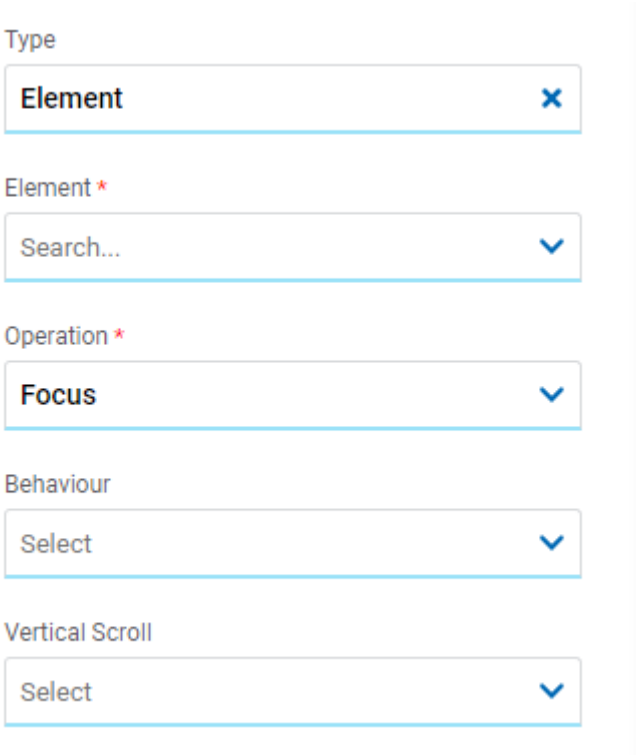
Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  Modal Horizontal Size
  Defines its width.
  Modal Vertical Size
  Defines its height.
- Iframe: selecting this option enables an additional property.
  Destination Iframe
  Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Parameters
Allows sending parameters to the selected object type.

Element



Element
The modeled elements in the object are displayed.

Operation
Defines the operation made when selecting the element.

Possible Values

- Focus
- Show
- Hide
- Show/Hide

Behaviour
Establishes the transition of the animation. It is only modelable if the "Focus" operation was selected.

Possible Values

- Auto
- Smooth

Vertical Scroll
It is only modelable if the "Focus" operation was selected.
Possible Values

- Start: Moves to the start of the selected element.
- Closest: Moves to the closest position to the selected element from the element the event fires.
- Center: Moves to the center of the selected element.
- End: Moves to the end of the selected element.

Repeater



Repeater
The modeled repeaters from the object are displayed.

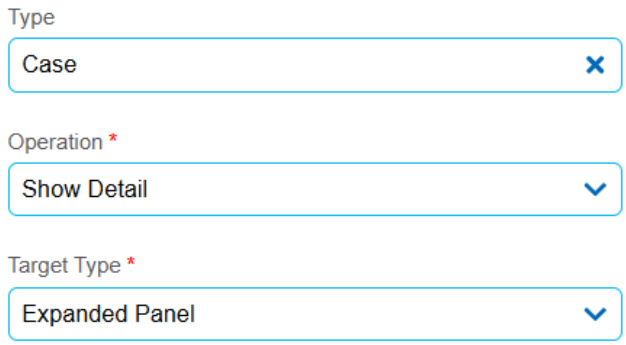
Operation

Defines the operation made when selecting the element.

Possible Values

- New: Allows for the creation of an item within the repeater. If this option is selected the additional properties are enabled.
  Creation Order
  It indicates in which position the new item of the container is added.
  Possible Values
  - Home
  - End
- Delete: Allows for the deletion of an item within the repeater.

Case



Case
Show case information.

Operation
Defines the operation made when selecting the element.

Possible Values

- Show Detail
- Show Graphic Execution

Destination Type
The available options for opening the object are displayed.

Possible Values

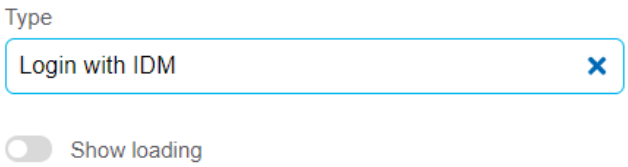
- Expanded Panel
- Modal: if this option is selected the additional properties are enabled.
  Modal Horizontal Size
  Define its width.
  Modal Vertical Size
  Define its height.

Back



It allows associating the event to go back in the browser to the element.

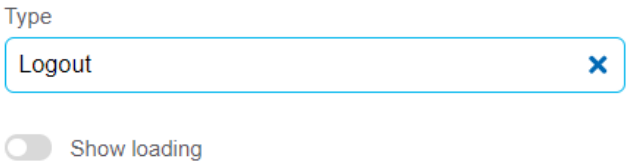
LogIn with IDM



Allows login with IDM.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

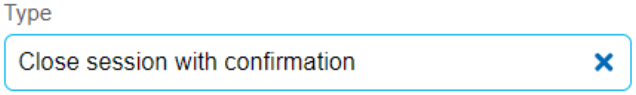
Logout



Allows logging the user out.

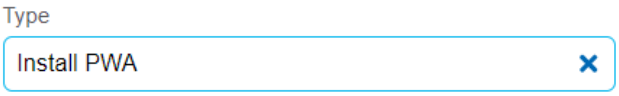
Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Close Session with Confirmation



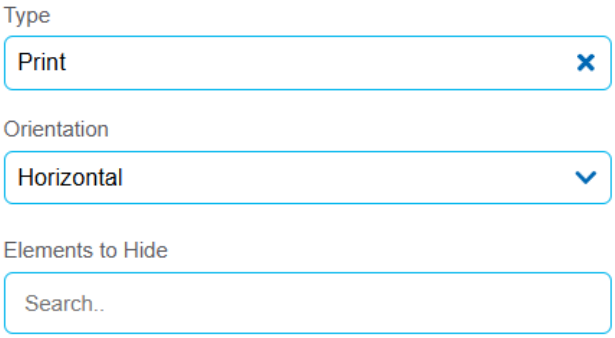
Displays a confirmation modal to logout the user.

Install PWA



Allows to install the application in the browser.

Print



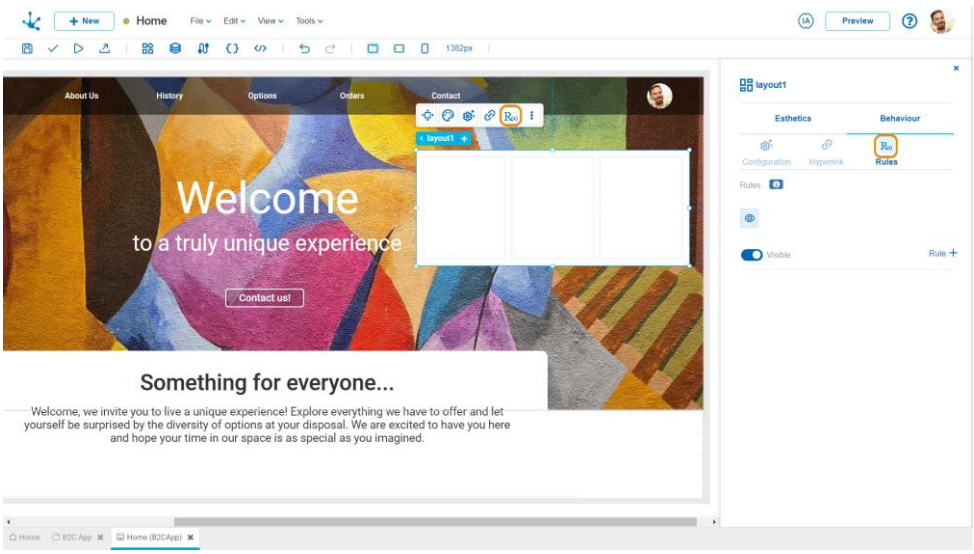
This type of hyperlink associated with an element allows printing the content of the entity or page that is being executed.

Orientation
Determines the layout of the content to be printed.
Possible Values
- Horizontal
- Vertical

Elements to hide
It allows selecting which elements of the entity or page should not be included in the printout.

Rule Properties

The rule properties panel of an element opens when clicking the icon  of its context menu.



Properties

Rules

Embedded rules on behavior, validation and calculation can be defined, associated with an element, by using the wizard (ctrl + space).

Shows syntax examples for writing the rules.

Visible

Indicates whether the element is visible. If this property is not checked, the element is not displayed in the page.

Visible (default) Not visible

Rule + Opens an edit area where a rule to determine the visibility condition can be defined. If a rule is defined, the icon is displayed with light blue borders.

Operations when defining the rule:

Saves the new or modified rule

Cancels the operation

Operations once the rule is defined:

Edits the existing rule

Deletes the rule

Item

More items can be included to the design and define their characteristics by modeling its properties.

Operations

Add

Allows to add an item to the design clicking on the icon from the element´s context menu.

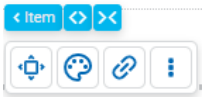


Move:

The change of position for an item is made in the design configuration properties panel.

Change Size

The size of an item can be changed from its context menu using the icons to enlarge their size and to reduce it, as well as from its structure properties panel.



Context Menu

Clicking the right button of the mouse on an element expands a second context menu whose options correspond to operations that can be performed on the selected item. The same menu is expanded pressing the icon from the palette.

- Cut (Ctrl+X)
- Copy (Ctrl+C)
- Paste (Ctrl+V)
- Duplicate (Ctrl+D)
- Lock/Unlock When an item is locked, it cannot be moved until it is unlocked.
- Hide Hides the item in the breakpoint that is being used and in minor breakpoints. Once it is hidden it can be shown again from Layers.
- Delete (Supr)
- Properties

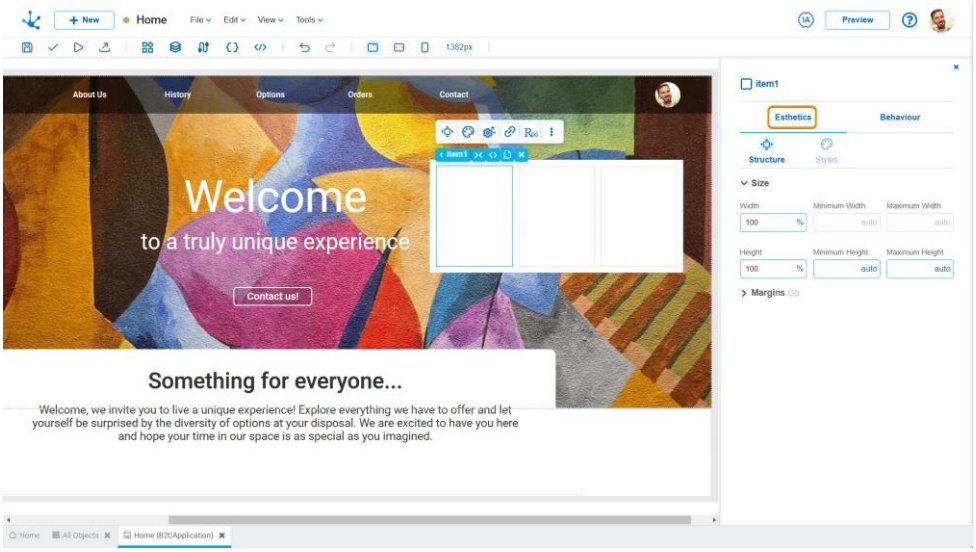
Classification of Properties

Properties are divided into two groups: esthetic properties and behavior properties.

Esthetic Properties

In the esthetic properties panel, the following are grouped:

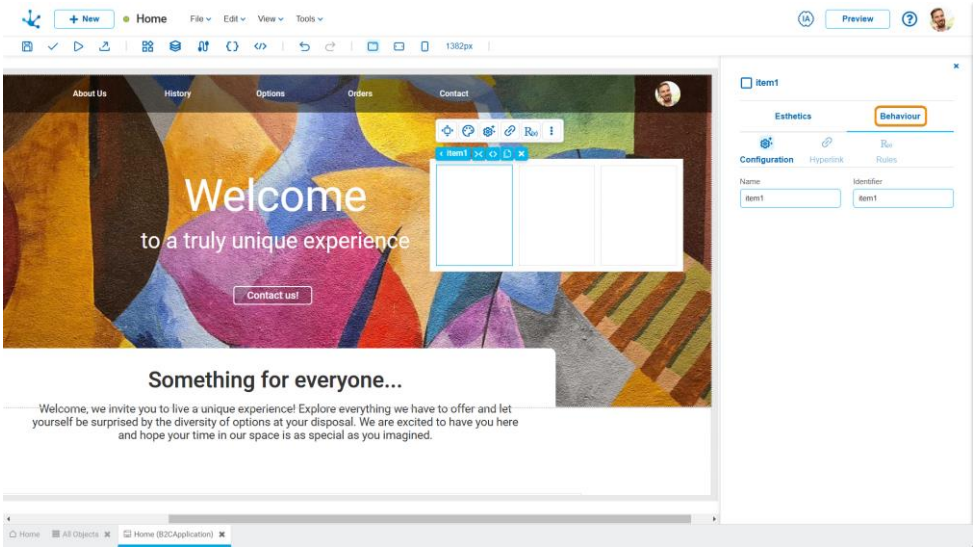
- Structure Properties
- Style Properties



Behavior Properties

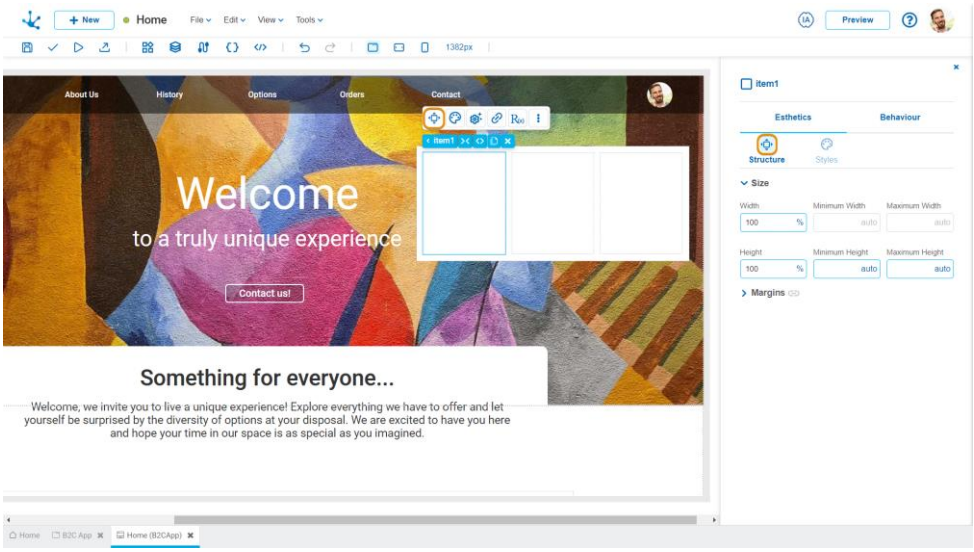
In the behavior properties panel, the following are grouped:

- Configuration Properties
- Hyperlink Properties
- Rules Properties

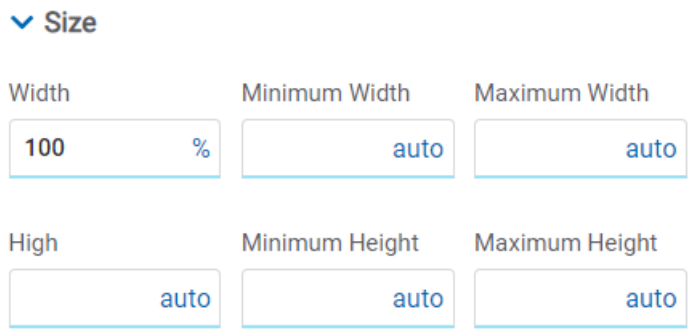


Structure Properties

The structure properties panel of an element opens when selecting the icon  of its context menu.



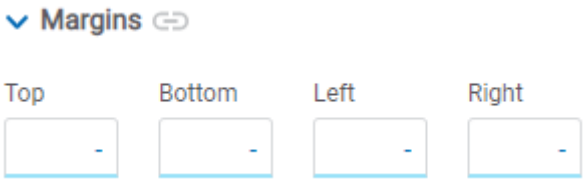
Size



All size properties can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh). Additionally, for Width and Height properties, the "auto" option is added.

> *If selecting percentage (%) for the width and height properties, the size is calculated relative to the top element.*

Margins



Allows to define the distance among items and also from items to the borders of the containing element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Top

Distance to the top border of the containing element.

Bottom
Distance to the bottom border of the containing element.

Left
Distance to the left border of the containing element or to the previous item.

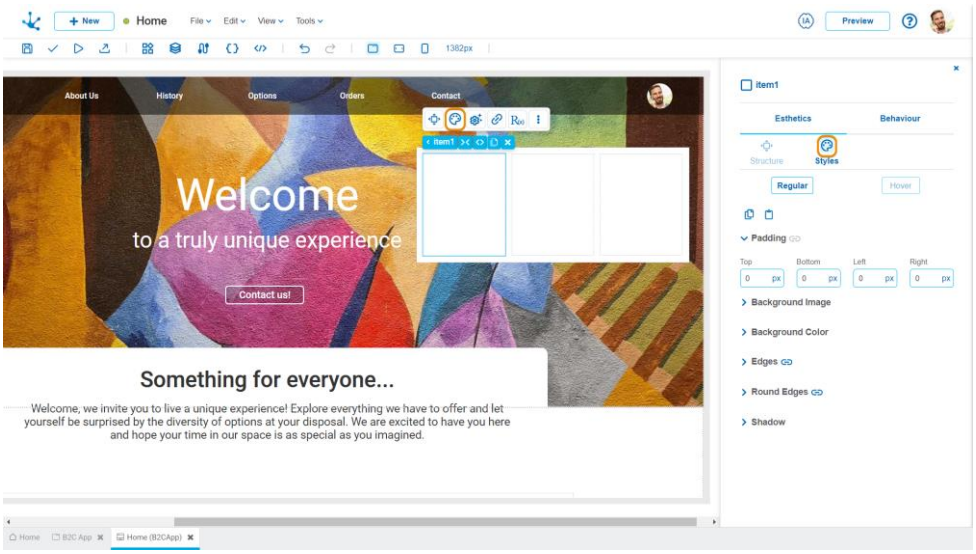
Right
Distance to the right border of the containing element or to the following item.

Allows the value entered in one of the margins to be copied to the other ones automatically.

Allows to indicate different values for each margin.

Style Properties

This type of element may take different states and for each of them different values for its properties may be modeled.





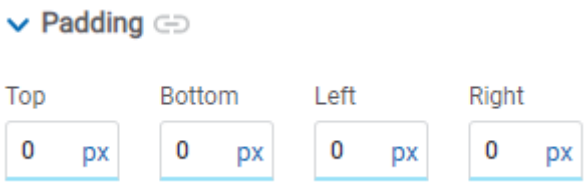
- Regular: The mouse pointer is not over the element.
- Hover: The mouse pointer is over the element.

Styles

It allows copying the entire current configuration of the style panel for the selected element (Ctrl+Alt+C).

It allows applying the previously copied configuration to another element of the same type (Ctrl+Alt+V).

Padding

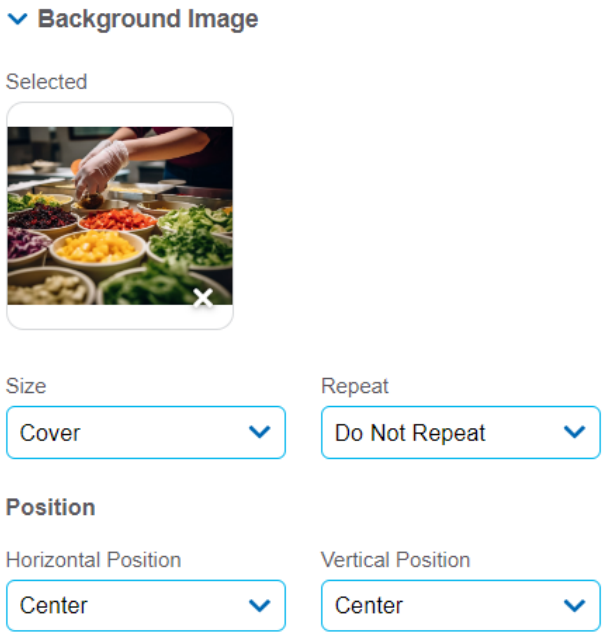


All padding space properties allow to create a space around the item borders (top, bottom, sides) and its content. The padding space configured is inward, while the margin is configured relative to a top item. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Allows values entered in one of the paddings to be copied to the other ones automatically.
Allows to indicate different values for each padding.

Background Image

It allows the addition of a background image to the item.

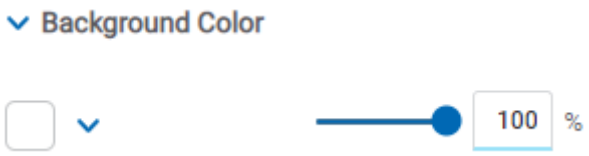


Selected
An image can be uploaded from the computer where it is being modeled.

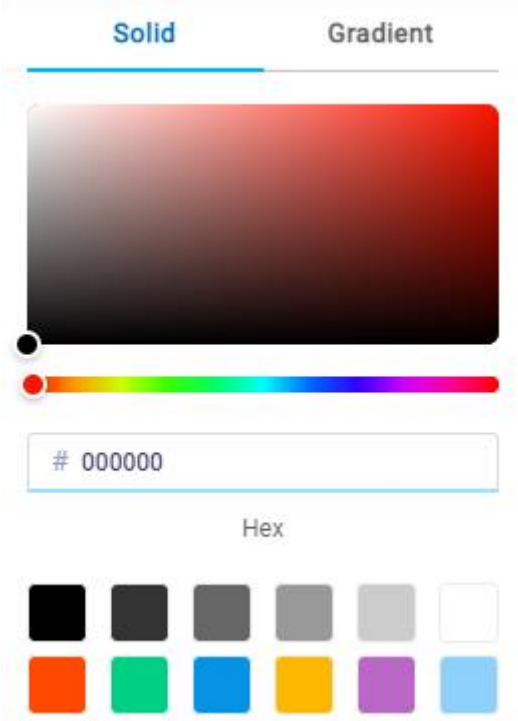
Background Color

It allows to model a background color for the element, it can be solid or gradient and different properties are defined for each one.

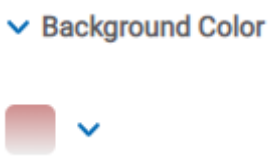
Solid



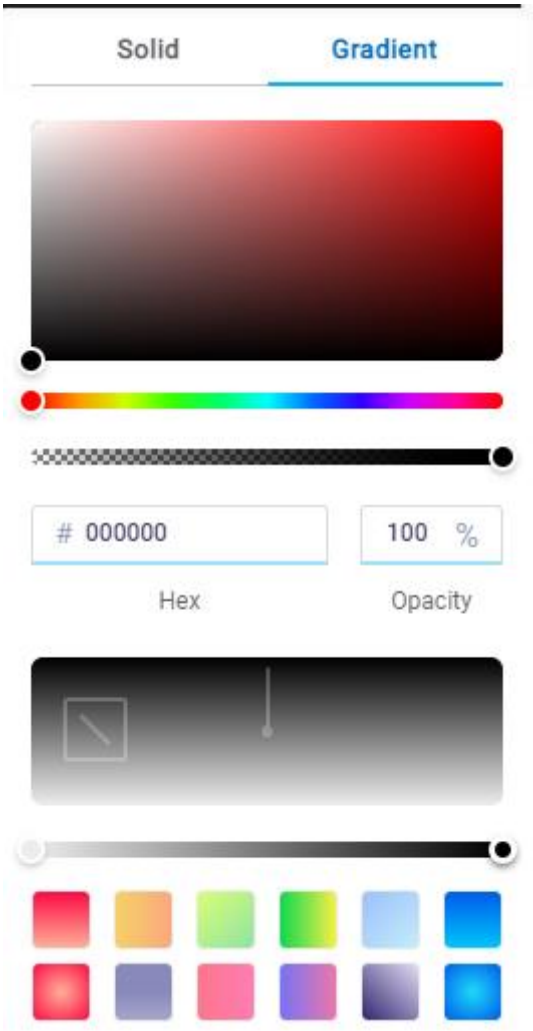
This option enables not only to select the color from a palette but also to define the degree of transparency.



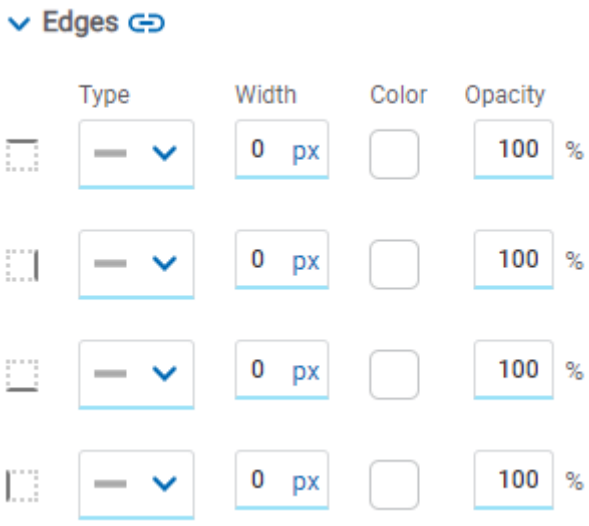
Gradient



This option enables to select not only the color from a palette but also to define its opacity and intensity according to the selected angle.



Edges



It allows to define the style of borders. Each one has its type, width, color and opacity defined, the latter as a percentage.

Allows the value entered in one of the borders to be copied to the other ones automatically.

Allows to indicate different values for each border.

Round Edges

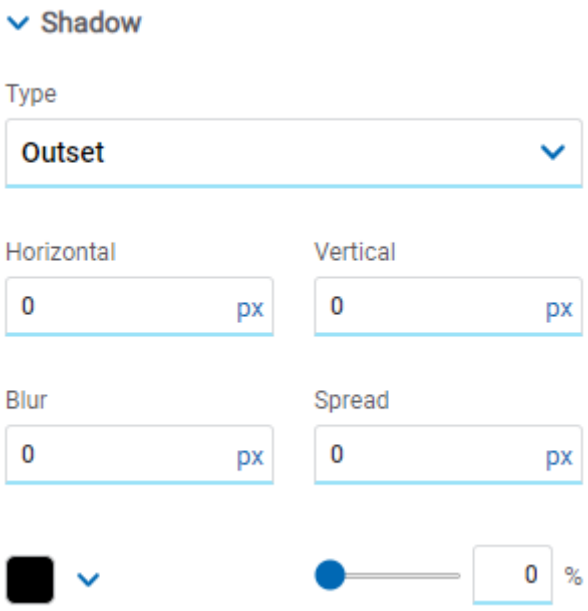


It allows to define the round edges at the corners of items.
It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Allows the value entered in one of the borders to be copied to the other ones automatically.

Allows to indicate different values for each border.

Shadow



Allows to define a shadow effect around the item.

Type

Possible Values

- Outset
- Inset

Horizontal
Horizontal size of the shadow to the right of the item. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Vertical
Vertical size of the shadow below the item. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

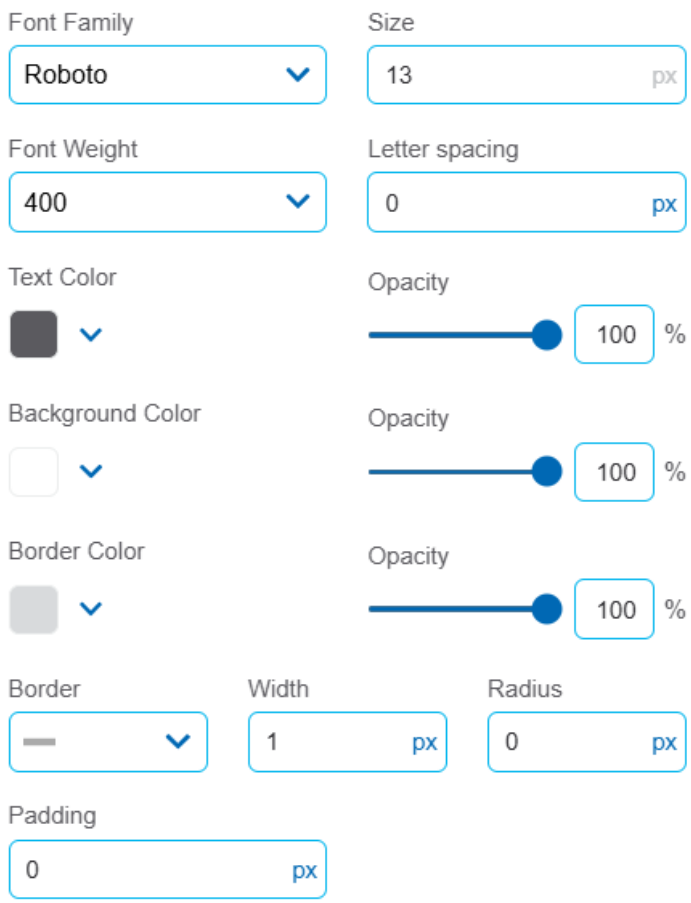
Blur
The greater this value is, the greater and lighter the shadow becomes. If not specified, its value is 0 and the shadow border is darker. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Spread
Positive values will cause the shadow to expand and grow bigger, while negative values will cause the shadow to shrink. If not specified, its value is 0 and the shadow will have the same size as the item.

Color
Defines the color of the shadow.

Transparency
Defines the transparency of the shadow.

Tooltip



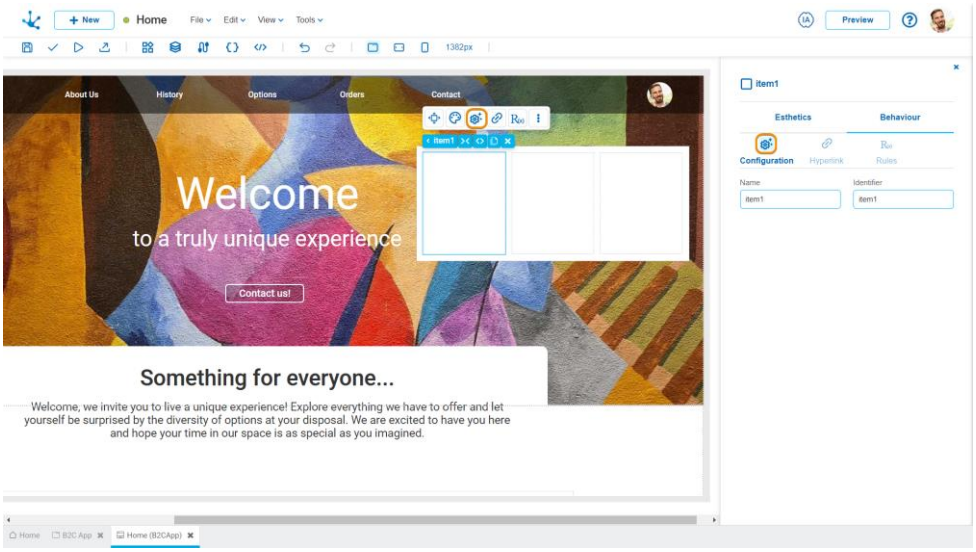
Source
It allows customizing the font, size, thickness and letter spacing of the pop-up message.

Colors
The color and opacity of the text, background and border can be defined. Borders are set uniformly for all sides of the tooltip.

Border
The border type, width, and border radius can be configured.

Configuration Properties

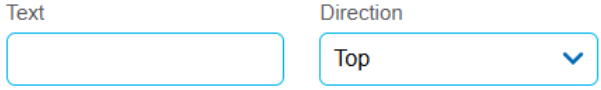
The configuration properties panel of an element opens when clicking the icon  of its context menu.



Name
Name used to reference the element during modeling.

Identifier
Uniquely identifies the element. It is used in the Javascript SDK.

Tooltip
Pop-up message that appears when hovering over an element, providing additional information.



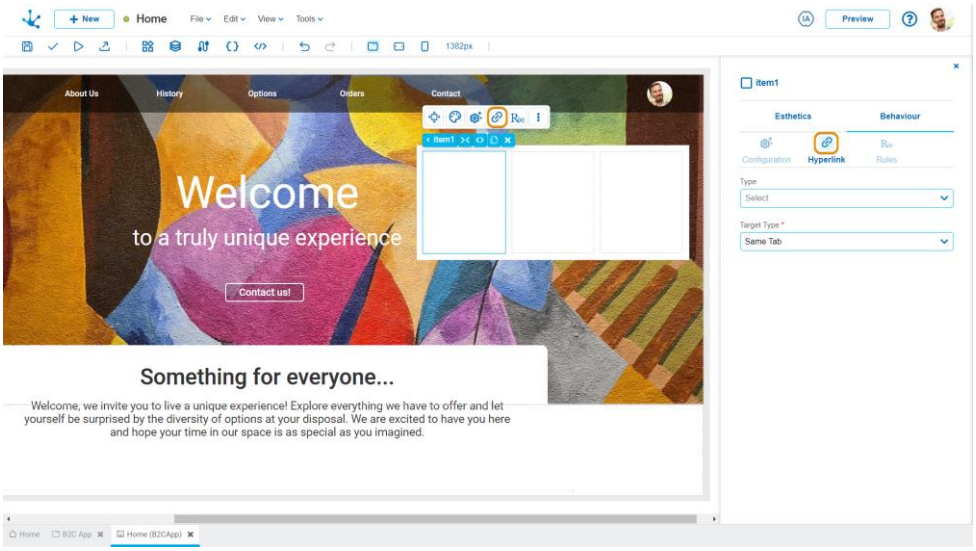
Text
Content of the message being displayed.

Direction

Position of the message in relation to the element (top, bottom, left, or right).

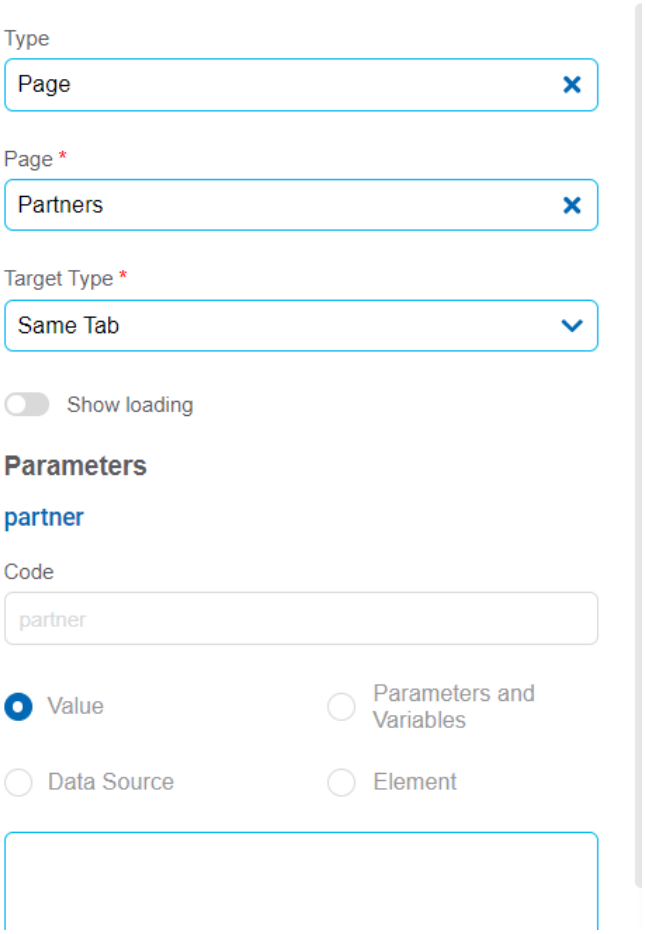
Hyperlink Properties

The hyperlink properties panel of an element opens when clicking the icon  of its context menu.



Define the behavior of the element when selecting it. Different properties are enabled depending on the type of object selected.

Page



Page
The pages modeled in the environment are displayed.

Destination Type
The available options for opening the object are displayed.

Possible Values

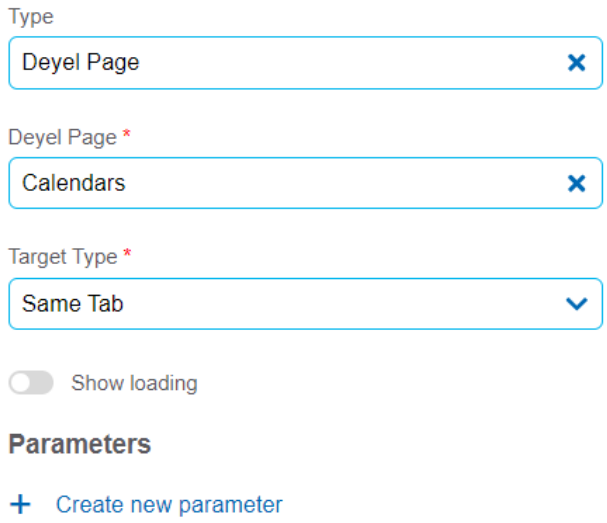
- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  Modal Horizontal Size
  Defines its width.
  Modal Vertical Size
  Defines its height.
- Iframe: selecting this option enables an additional property.
  Destination Iframe
  Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading

Shows the loading icon and disables the user interaction until the redirection is over.

Parameters
If there are parameters, their value can be specified with text, parameters and variables or data source.

Deyel Page



Deyel Page
The pages belonging to **Deyel** are displayed.

Destination Type
The available options for opening the object are displayed.

Possible Values

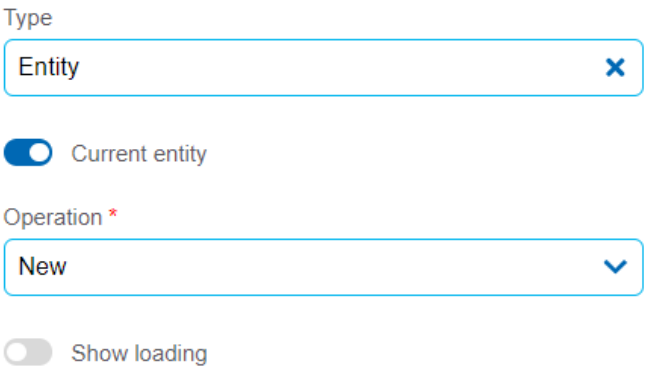
- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  - Modal Horizontal Size
    Defines its width.
  - Modal Vertical Size
    Defines its height.
- Iframe: selecting this option enables an additional property.
  - Destination Iframe
    Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Parameters

Allows sending parameters to the selected object type.

Entity



Current Entity

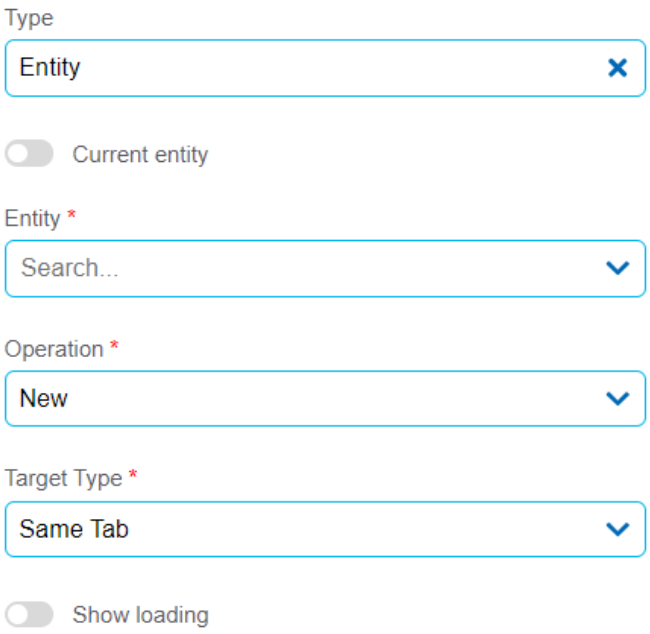
If this property is checked, the operation applies directly to the entity being modeled.

Operation

Defines the operation made when selecting the object.

Possible Values

- New: Indicates that the current entity panel is opened for the creation of an instance.
- Modify: Indicates that the current entity panel is opened for the update of an instance.
- Delete: Indicates that the current entity panel is opened for the deletion of an instance.



Entity
If the property Current Entity, is not checked, the entities modeled in the environment are displayed.

Operation
Defines the operation made when selecting the object.

Possible Values

- New: The panel of the entity selected in the previous property is opened to create an instance.
- Update The panel of the entity selected in the previous property is opened to update an instance.
- Show: The panel of the entity selected in the previous property is opened to show an instance.
- Grid: Indicates that the grid of the entity selected in the previous property is opened.

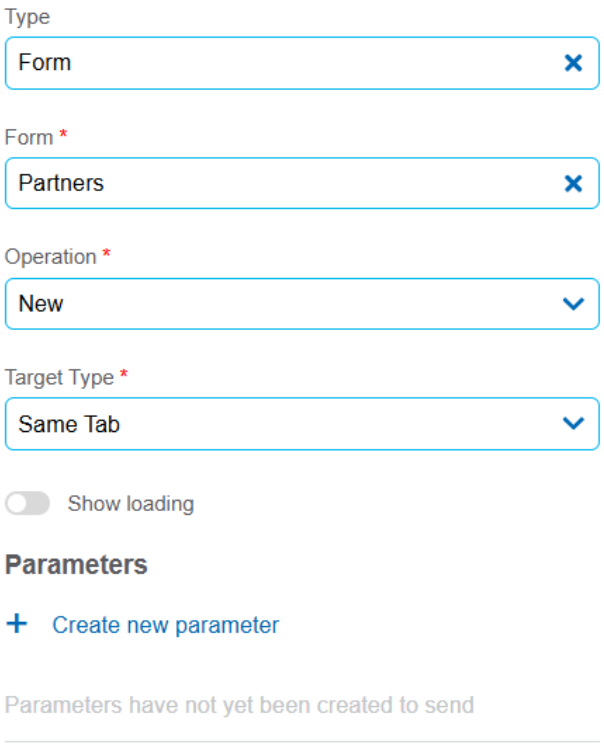
Destination Type
The available options for opening the object are displayed.

Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  Modal Horizontal Size
  Defines its width.
  Modal Vertical Size
  Defines its height.
- Iframe: selecting this option enables an additional property.
  Destination Iframe
  Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Forms



Forms
The forms modeled in the environment are displayed.

Operation
Defines the operation made when selecting the object.

Possible Values

- Create: Indicates that the panel of the object selected in the previous property is opened for the creation of an instance.
- Update: Indicates that the panel of the object selected in the previous property is opened for the update of an instance.
- Show: Indicates that the panel of the object selected in the previous property is opened for the query of an instance.
- Grid: Indicates that the grid of the object selected in the previous property is opened.

Destination Type
The available options for opening the object are displayed.

Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  - Modal Horizontal Size
    Defines its width.
  - Modal Vertical Size
    Defines its height.
- Iframe: selecting this option enables an additional property.
  - Destination Iframe

Expands the iframes previously defined on the object.

- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

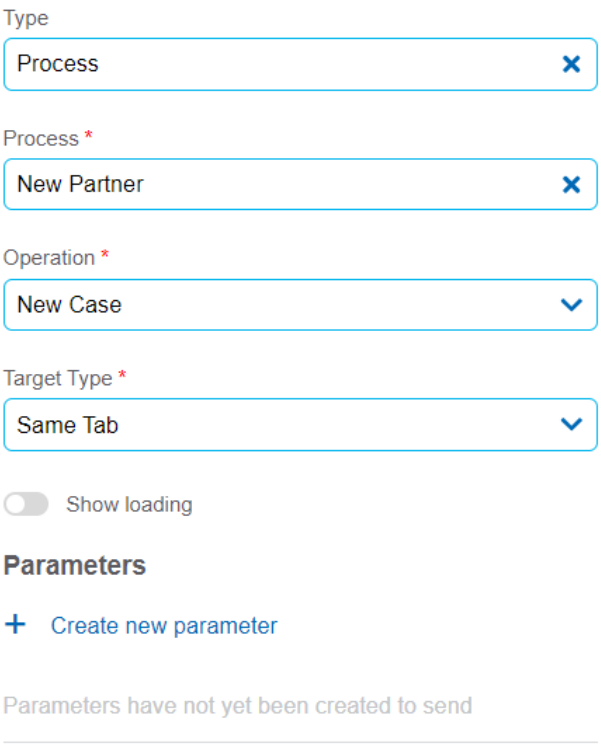
Show loading

Shows the loading icon and disables the user interaction until the redirection is over.

Parameters

Allows sending parameters to the selected object type.

Process



Process

The processes modeled in the environment are displayed.

Operation

Defines the operation made when selecting the object.

Destination Type

The available options for opening the object are displayed.

Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.

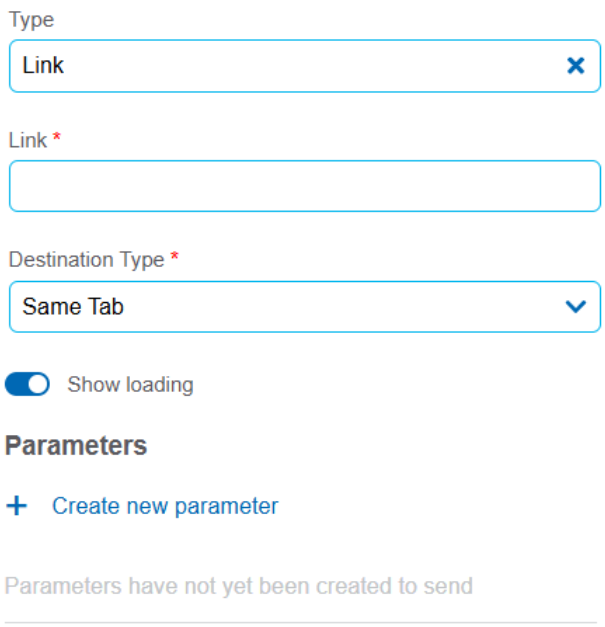
Modal Horizontal Size
Defines its width.
Modal Vertical Size
Defines its height.

- Iframe: selecting this option enables an additional property.
  Destination Iframe
  Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Parameters
Allows sending parameters to the selected object type.

Link



Link
Allows to enter any link.

Destination Type
The available options for opening the object are displayed.

Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  Modal Horizontal Size

Defines its width.

Modal Vertical Size

Defines its height.

- Iframe: selecting this option enables an additional property.

  Destination Iframe

  Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

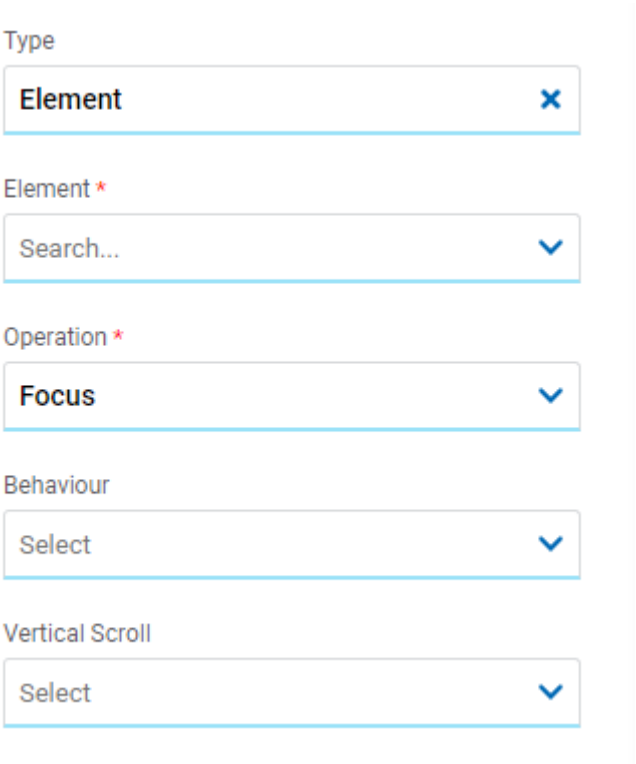
Show loading

Shows the loading icon and disables the user interaction until the redirection is over.

Parameters

Allows sending parameters to the selected object type.

Element



Element

The modeled elements in the object are displayed.

Operation

Defines the operation made when selecting the element.

Possible Values

- Focus
- Show
- Hide

- Show/Hide

Behaviour
Establishes the transition of the animation. It is only modelable if the "Focus" operation was selected.

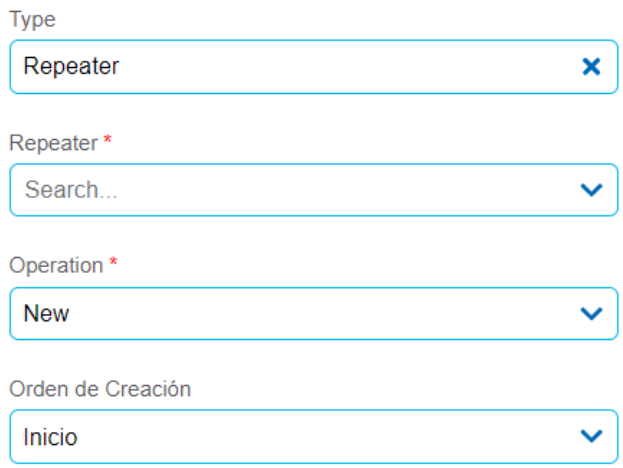
Possible Values

- Auto
- Smooth

Vertical Scroll
It is only modelable if the "Focus" operation was selected.
Possible Values

- Start: Moves to the start of the selected element.
- Closest: Moves to the closest position to the selected element from the element the event fires.
- Center: Moves to the center of the selected element.
- End: Moves to the end of the selected element.

Repeater



Repeater
The modeled repeaters from the object are displayed.

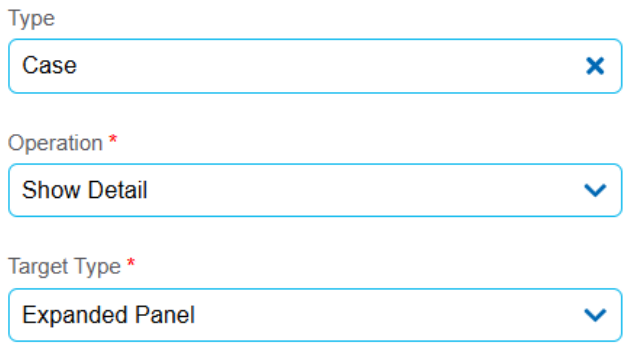
Operation
Defines the operation made when selecting the element.

Possible Values

- New: Allows for the creation of an item within the repeater. If this option is selected the additional properties are enabled.
  ### Creation Order
  It indicates in which position the new item of the container is added.
  Possible Values
  - Home
  - End

- Delete: Allows for the deletion of an item within the repeater.

Case



Case
Show case information.

Operation
Defines the operation made when selecting the element.

Possible Values

- Show Detail
- Show Graphic Execution

Destination Type
The available options for opening the object are displayed.

Possible Values

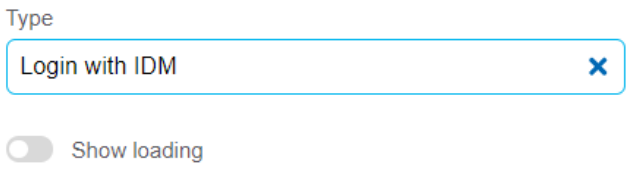
- Expanded Panel
- Modal: if this option is selected the additional properties are enabled.
  Modal Horizontal Size
  Define its width.
  Modal Vertical Size
  Define its height.

Back



It allows associating the event to go back in the browser to the element.

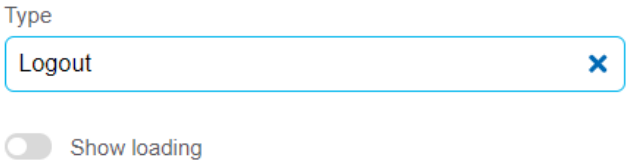
LogIn with IDM



Allows login with IDM.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

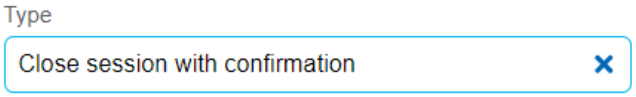
Logout



Allows logging the user out.

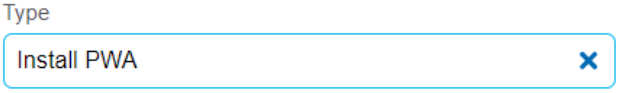
Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Close Session with Confirmation



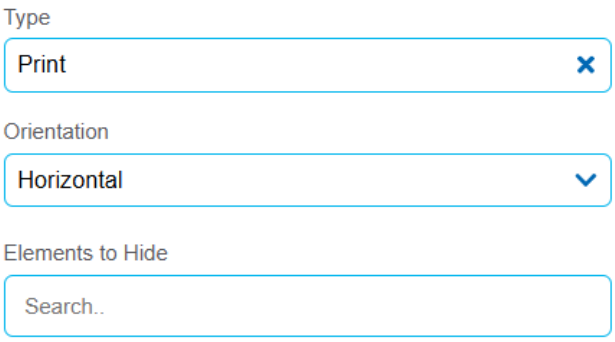
Displays a confirmation modal to logout the user.

Install PWA



Allows to install the application in the browser.

Print



This type of hyperlink associated with an element allows printing the content of the entity or page that is being executed.

Orientation

Determines the layout of the content to be printed.

Possible Values

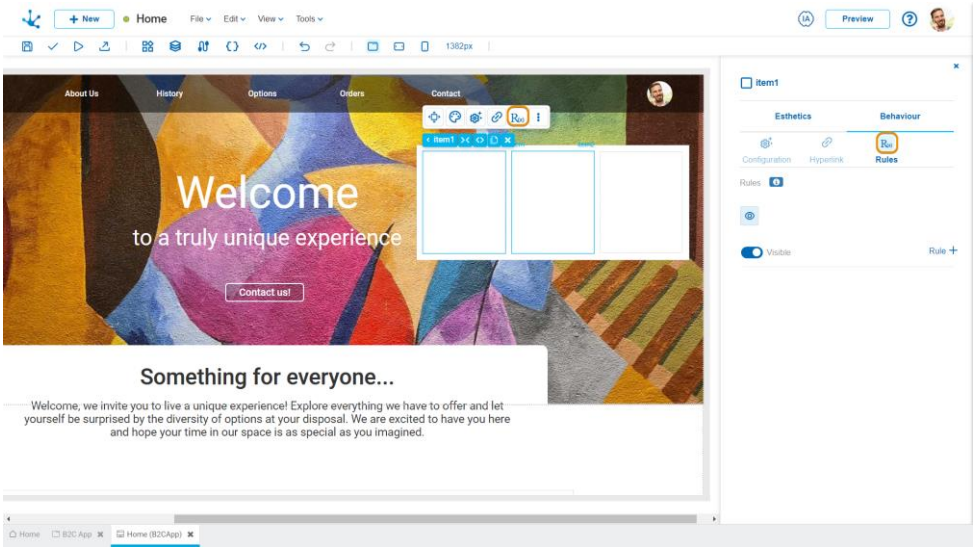
- Horizontal
- Vertical

Elements to hide

It allows selecting which elements of the entity or page should not be included in the printout.

Rule Properties

The rule properties panel of an element opens when clicking the icon  of its context menu.



Properties

Rules

Embedded rules on behavior, validation and calculation can be defined, associated with an element, by using the wizard (ctrl + space).
Shows syntax examples for writing the rules.

Visible
Indicates whether the element is visible. If this property is not checked, the element is not displayed in the page.
Visible (default) Not visible
Rule + Opens an edit area where a rule to determine the visibility condition can be defined. If a rule is defined, the icon is displayed with light blue borders.

Events

Items allow the use of different events.

Event	Description
onMouseIn()	It is executed when the cursor is positioned on the element.
onMouseOut()	It is executed when the cursor moves out of the element.
onClick()	It is executed when clicking on the element.
onDoubleClick()	It is executed when clicking twice on the element.
onInit()	It is executed before the element is loaded.
afterViewInit()	It is executed after viewing the element.
onViewportEnter()	It is executed when the element is visible.
onViewportLeave()	It is executed when the element is no longer visible.

Repeater

The repeater element allows to display a set of retrieved instances by a data source, unlike other elements that only allow a single instance to be displayed.

Visually it is similar to the layout element but only the first item can be modified and a data source must be defined. Such item repeats as many times as instances are obtained from the data source.

If the items included in the repeater show information on the instances of the data source defined in it, such source must be previously configured.

The element properties are represented by icons in its context menu, where its operations are also available.

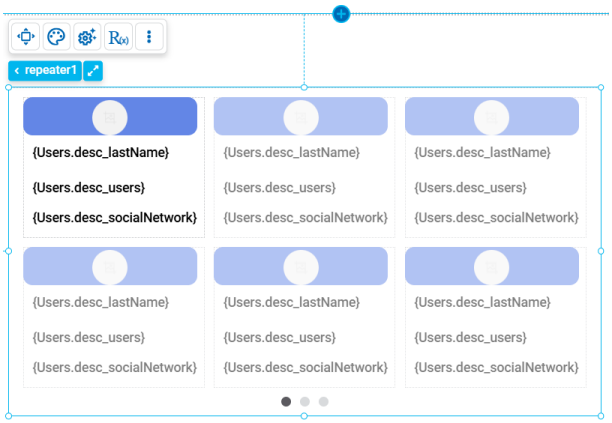
Subtypes

When selecting the option "Repeater" from the icon in the top toolbar, a list opens with the different subtypes of this element, which can be dragged and dropped into the modeling area. Each subtype has the element's properties modeled and built in a specific way.

- Cards
- Tape
- Slider

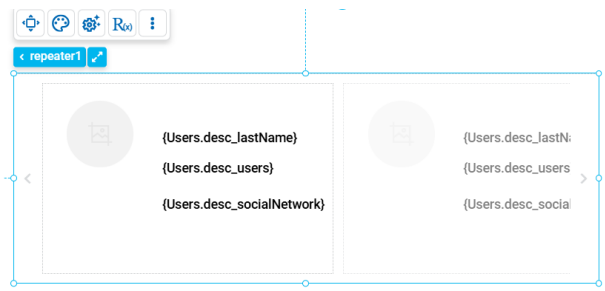
Cards

This type of repeater organizes its content in a visual card format, where each card acts as an independent unit that may contain different elements with values obtained from a data source. This type of repeater is ideal for representing structured content in an attractive and functional way.



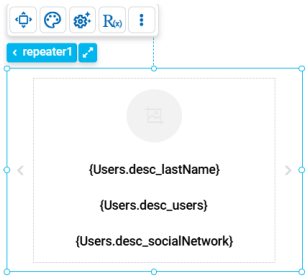
Tape

This type of repeater organizes elements in a horizontally movable structure, as if they were part of a continuous tape. It is ideal for displaying content that may come from a data source, in a limited space, while allowing users to explore more elements using browsing controls.



Slider

This type of repeater organizes elements in an interactive format that can be scrolled horizontally or vertically. It is ideal for displaying featured content, such as images, products or news, in an interactive and compact way, with values obtained from a data source.



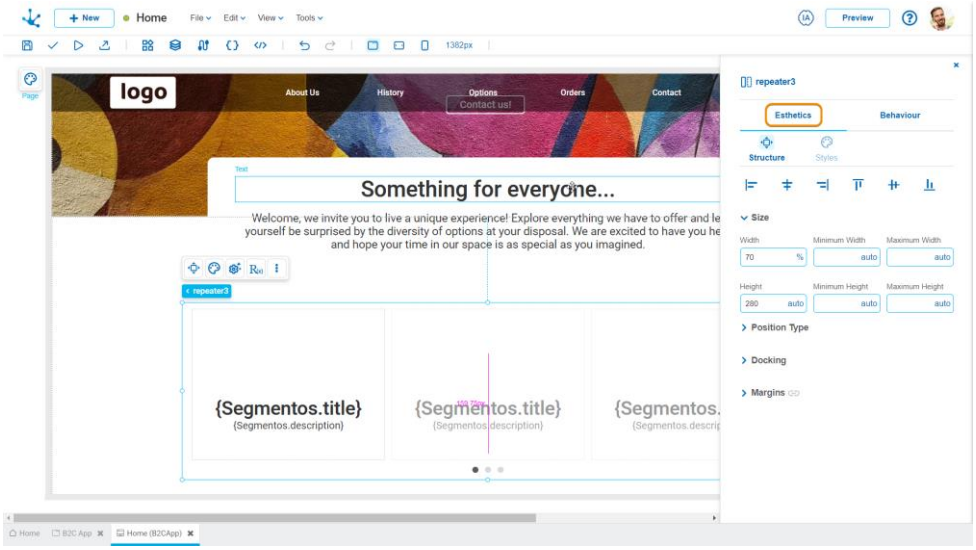
Classification of Properties

Properties are divided into two groups: esthetic properties and behavior properties.

Esthetic Properties

In the esthetic properties panel, the following are grouped:

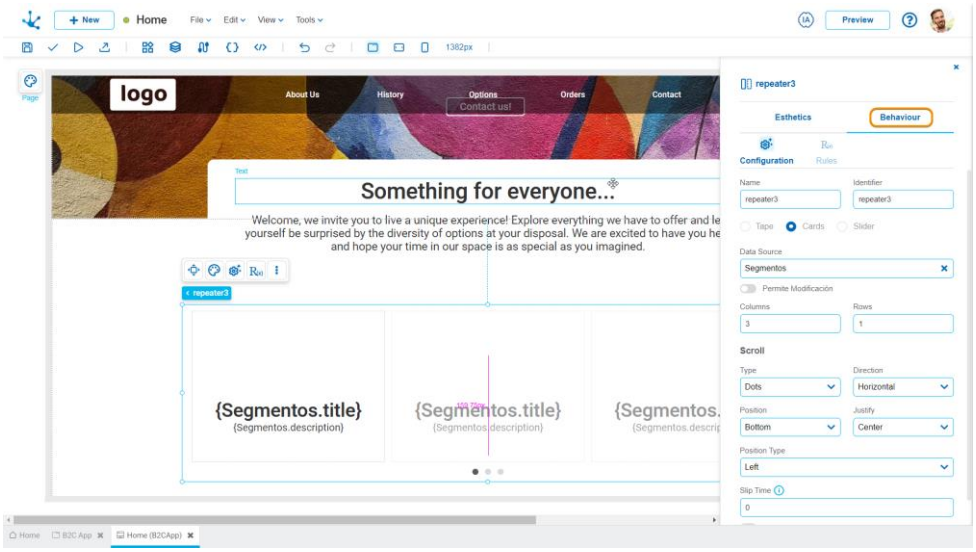
- Structure Properties
- Style Properties



Behavior Properties

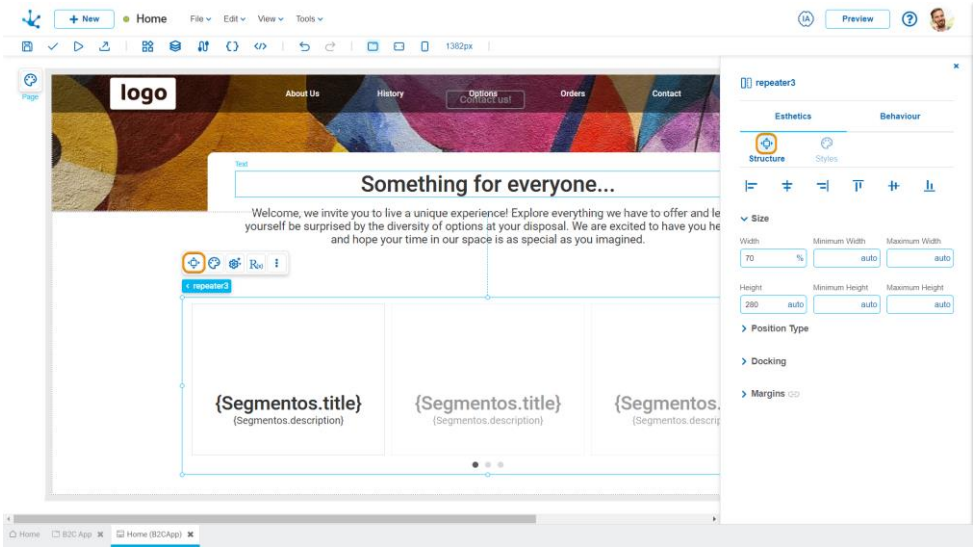
In the behavior properties panel, the following are grouped:

- Configuration Properties
- Rules Properties

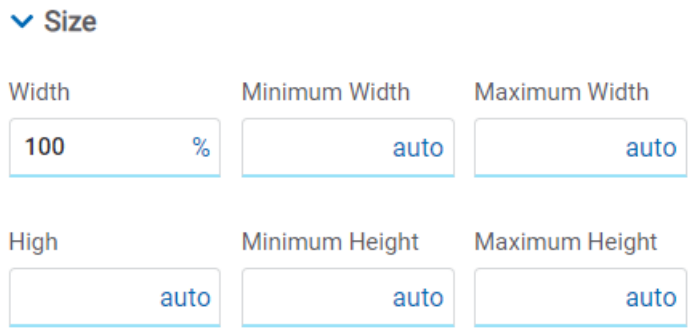


Structure Properties

The structure properties panel of an element opens when selecting the icon  of its context menu.



Size

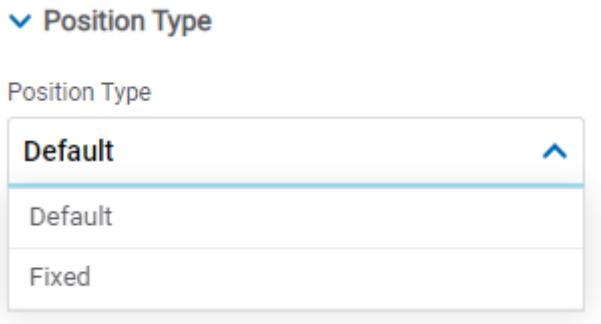


All size properties can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh). Additionally, for Width and Height properties, the "auto" option is added.

> *If selecting percentage (%) for the width and height properties, the size is calculated relative to the top element.*

Position Type

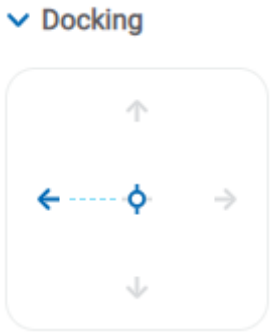
It determines if at the time of execution the element remains fixed on the page or if it moves as the scroll bar moves forward or backward.



Possible Values

- Default: The element has a relative position with respect to the superior element where it was placed (container or section).
- Fixed: Fixes the element to the page so that it is always visible in the same position. It allows to increase the visibility of important content.

Docking



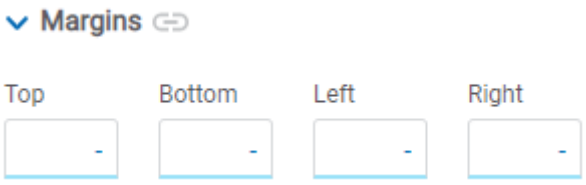
Determines the vertical and horizontal position of elements within the page section or container, when the screen is resized.

When an element is added or moved within the section or container, the modeler automatically docks it to the nearest corners or edges. It can also be configured by clicking on the arrows of the side to be docked to or on the center.

Once the element is docked, its exact position is indicated in the Margins property. The direction of the docking arrows determine the values that are automatically displayed for margins.

The docking position is indicated in the modeling area by dotted lines on the element.

Margins



It allows to define the distance of elements from the borders of their top element. The behavior of margins depends on the docking of the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Top
Distance to the top border of the highest ranking element.

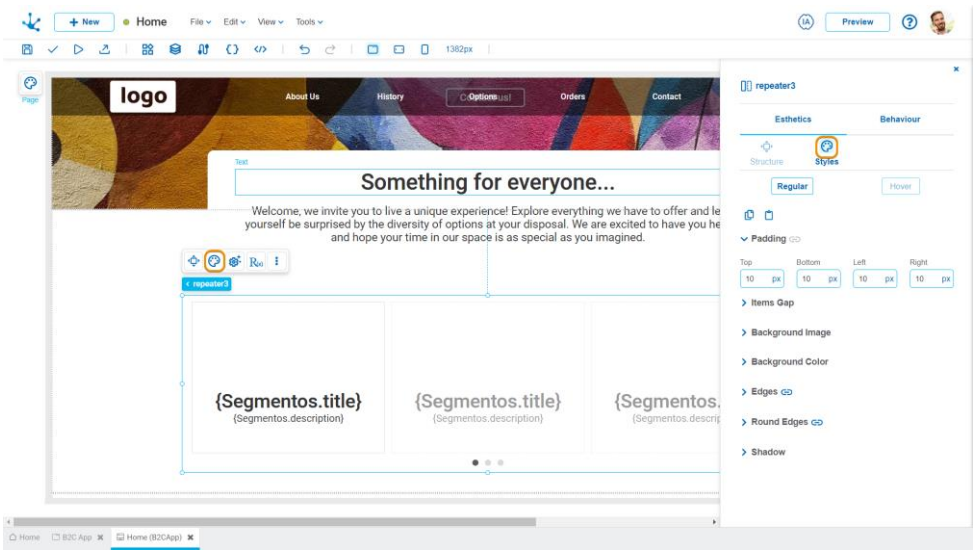
Bottom
Distance to the bottom border of the highest ranking element.

Left
Distance to the left border of the highest ranking element.

Right
Distance to the right border of the highest ranking element.

Allows the value entered in one of the margins to be copied to the other ones automatically.
Allows to indicate different values for each margin.

This type of element may take different states and for each of them different values for its properties may be modeled.



This type of element may take different states and for each of them different values for its properties may be modeled.



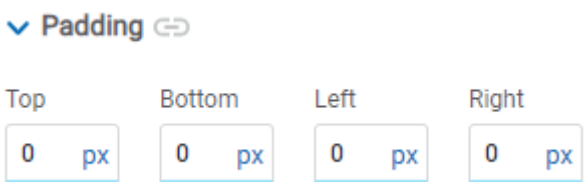
- Regular: The mouse pointer is not over the element.
- Hover: The mouse pointer is over the element.

Styles

It allows copying the entire current configuration of the style panel for the selected element (Ctrl+Alt+C).

It allows applying the previously copied configuration to another element of the same type (Ctrl+Alt+V).

Padding

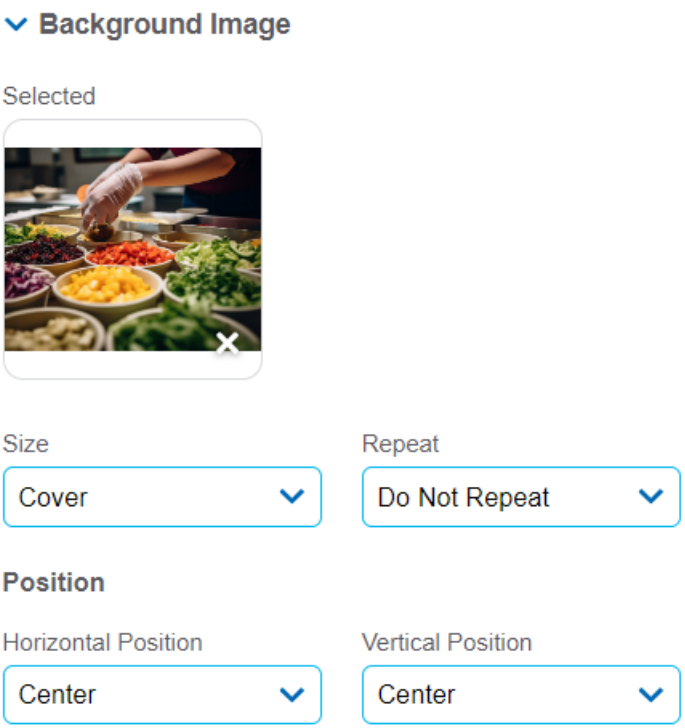


All padding space properties allow to create a space around the element borders (top, bottom, sides) and its content. The padding space configured is inward, while the margin is configured relative to a top element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Allows values entered in one of the paddings to be copied to the other ones automatically.
Allows to indicate different values for each padding.

Item Spacing
It defines the spacing between the items in the repeater.

Background Image



Allows to add a background image to the element.

Selected
An image can be uploaded from the computer where it is being modeled.

Size

Possible Values

- Cover
- Content
- Auto

Repeat

Possible Values

- Repeat
- Repeat Hor.
- Repeat Vert.
- Spacing
- Rounded
- Do Not Repeat

Horizontal Position

Possible Values

- Center
- Left
- Right

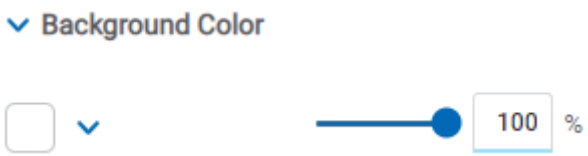
Vertical Position

- Center
- Top
- Botton

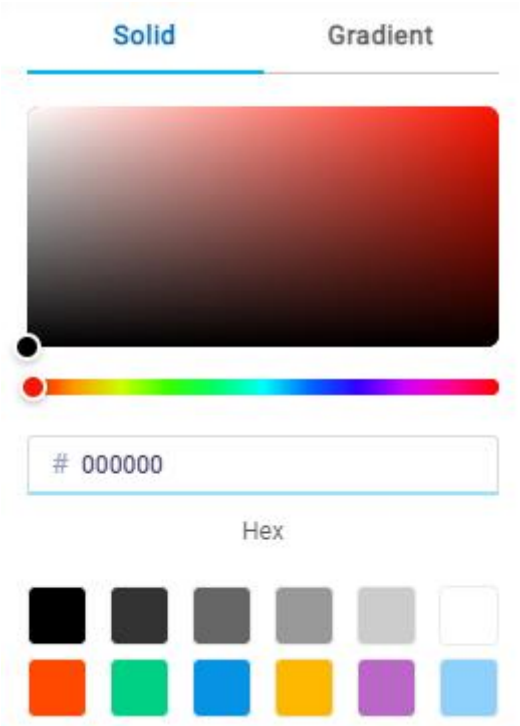
Background Color

It allows to model a background color for the element, it can be solid or gradient and different properties are defined for each one.

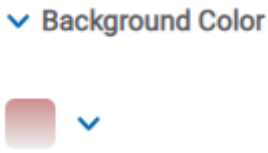
Solid



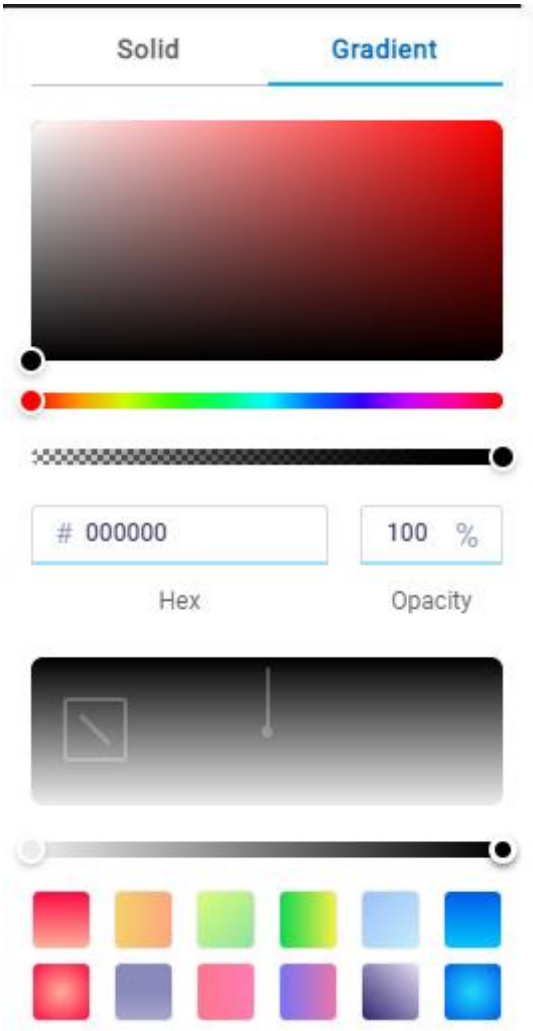
This option enables not only to select the color from a palette but also to define the degree of transparency.



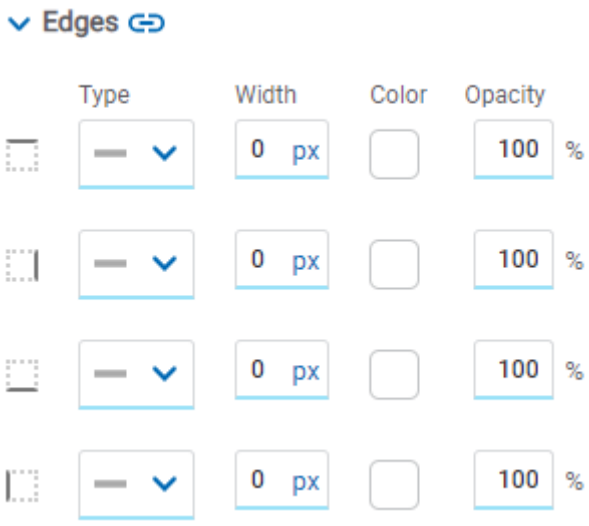
Gradient



This option enables to select not only the color from a palette but also to define its opacity and intensity according to the selected angle.



Edges



It allows to define the style of borders. Each one has its type, width, color and opacity defined, the latter as a percentage.

Allows the value entered in one of the borders to be copied to the other ones automatically.

Allows to indicate different values for each border.

Round Edges

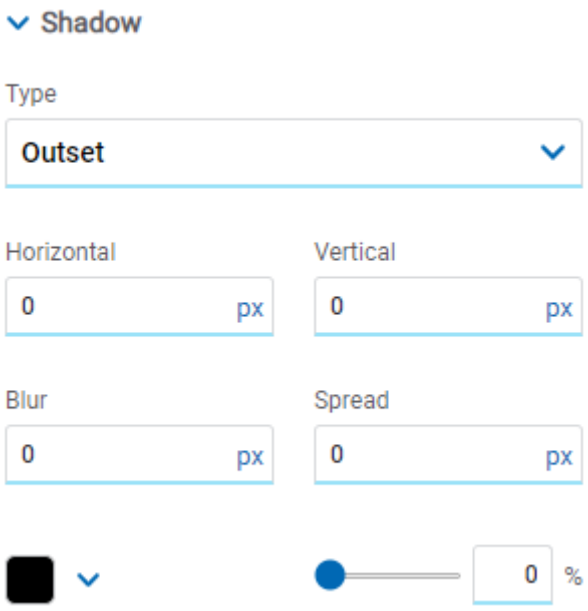


It allows to define the round edges at the corners of elements.
It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Allows the value entered in one of the borders to be copied to the other ones automatically.

Allows to indicate different values for each border.

Shadow



Allows to define a shadow effect around the element.

Horizontal
Horizontal size of the shadow to the right of the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Vertical
Vertical size of the shadow below the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Blur
The greater this value is, the greater and lighter the shadow becomes. If not specified, its value is 0 and the shadow border is darker. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Spread
Positive values will cause the shadow to expand and grow bigger, while negative values will cause the shadow to shrink. If not specified, its value is 0 and the shadow will have the same size as the element.

Color
Defines the color of the shadow.

Transparency
Defines the transparency of the shadow.

Pagination

The pagination properties depend on the type of scrolling modeled in the repeater's configuration properties and can be adjusted to achieve the desired behavior.

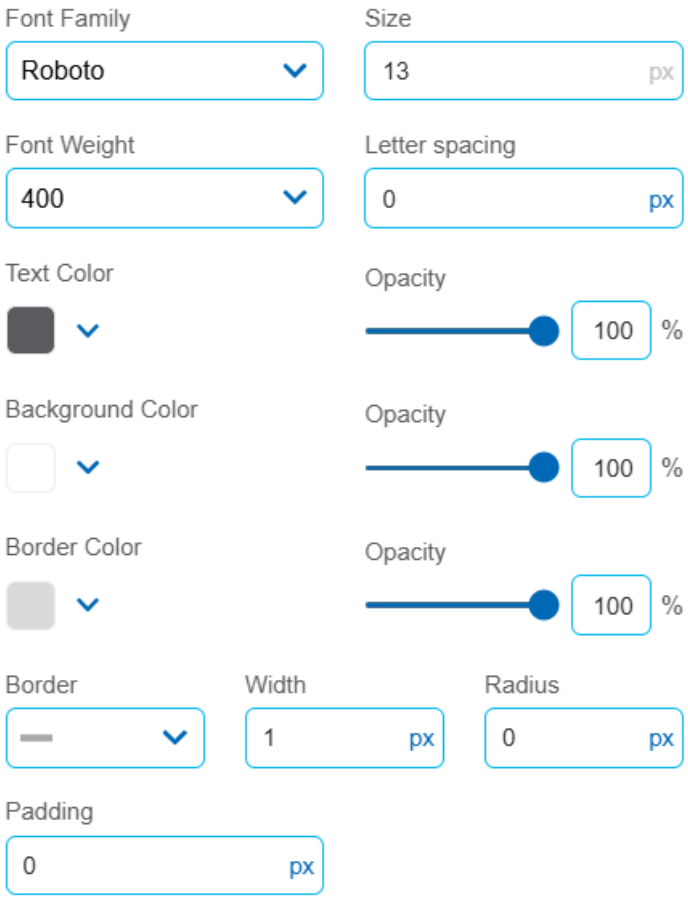
Arrow Scrolling
Arrows: size, color, background, disabled arrow color, and disabled background color.

> *The background color and the disabled background color only have an effect when the Position property is modeled as 'Inside'.*

Dot Scrolling
Dots: size, spacing, color and color of the selected dot.

Page Scrolling
Font: family, size and color
Buttons: size, spacing
Colors: background and font for normal, selected, and disabled states. It also allows setting the background color on hover.

Tooltip



Source

It allows customizing the font, size, thickness and letter spacing of the pop-up message.

Colors

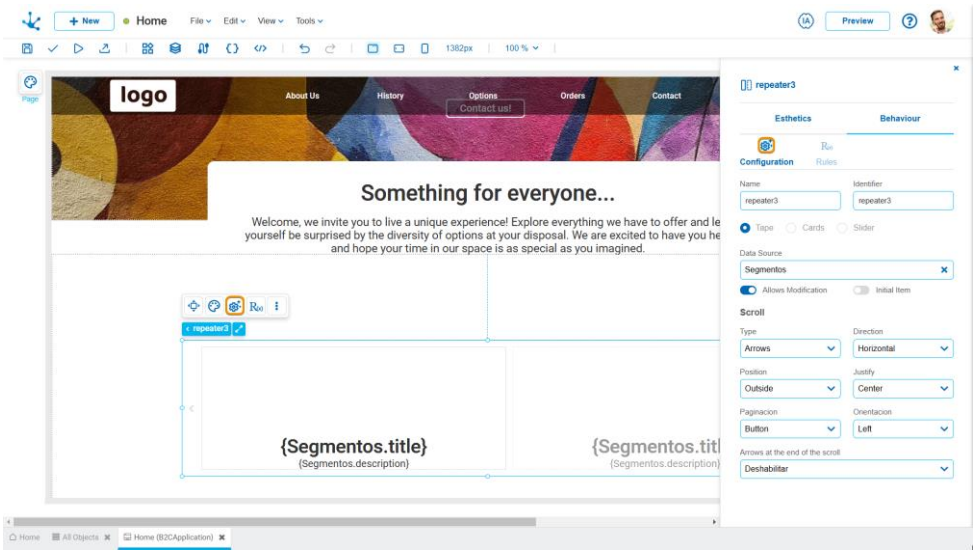
The color and opacity of the text, background and border can be defined. Borders are set uniformly for all sides of the tooltip.

Border

The border type, width, and border radius can be configured.

Configuration Properties

The configuration properties panel of an element opens when clicking the icon  of its context menu.



Name
Name used to reference the element during modeling.

Identifier
Uniquely identifies the element. It is used in the Javascript SDK.

Tooltip
Pop-up message that appears when hovering over an element, providing additional information.



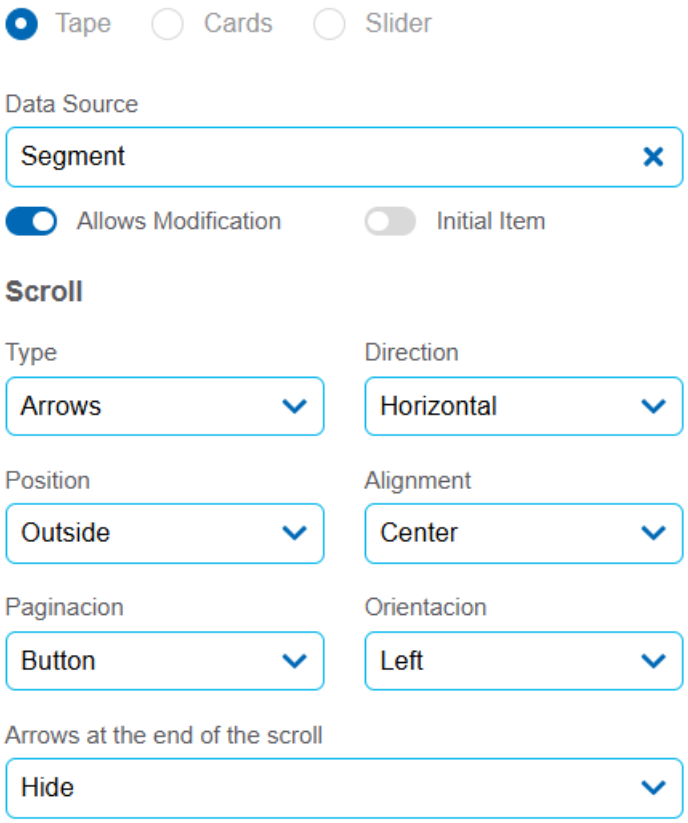
Text
Content of the message being displayed.

Direction
Position of the message in relation to the element (top, bottom, left, or right).

Items can be repeated in tape, card, or slider format.

Tape

When the page is executed, the items scroll within the repeater as the forward and backward icons are pressed. The number of items initially displayed on the tape depends on the size of the repeater.



Data Source

It allows selecting the source from which the data displayed on the repeater is retrieved. Once the source is selected, the Allows Modification property is enabled.

Allows Modification

It allows adding or removing items from the repeater using a modeled hyperlink. If the property is not activated, the number of items is fixed and cannot be modified. When activated, the Initial Item property is displayed.

Initial Item

It allows adding an empty item to the repeater using a modeled hyperlink.

> *If the repeater is empty at runtime, the initial item is automatically added, without the user needing to use the hyperlink.*

Type

If the page is going to be used from a desktop, the 'Arrows' option must be selected to enable the icons < *and* > indicate the forward or backward movement of the items on the tape. If it is going to be used from a touch breakpoint, the "None" option can be selected to enable scrolling.
If the selected value is "Arrows" the enabled properties are: Position and Alignment.

Possible Values

- Arrows
- None

Direction

Allows to select the direction in which the items within the element are displayed.

Possible Values

- Horizontal
- Vertical

Position and Alignment

Select the position of the scroll icons within the repeater.

Possible Values for Position

- Top
- Bottom
- Inside
- Outside

Possible Values for Alignment

- If the “Inside” or “Outside” values are selected for the Position property, the possible values of the Alignment property are: “Top”, “Center”, “Bottom”.
- If the “Top” or “Bottom” values are selected for the Position property, the possible values of the Alignment property are: “Left”, “Center”, “Right”.

Pagination

Defines how the elements of a repeater are loaded and displayed in the entity.

Possible Values

- Button: Loading more items is triggered manually, the user must click on a button labeled "Upload more" to show additional elements.
- Infinite: elements load automatically as the user scrolls down the entity. This behavior is known as infinite scroll.

Orientation

Defines the position at which the pagination element appears within the repeater container.

Possible Values

- Left: the pagination element is aligned to the left side of the container.
- Right: the pagination element is aligned to the right side of the container.

Arrows at the end of the scroll

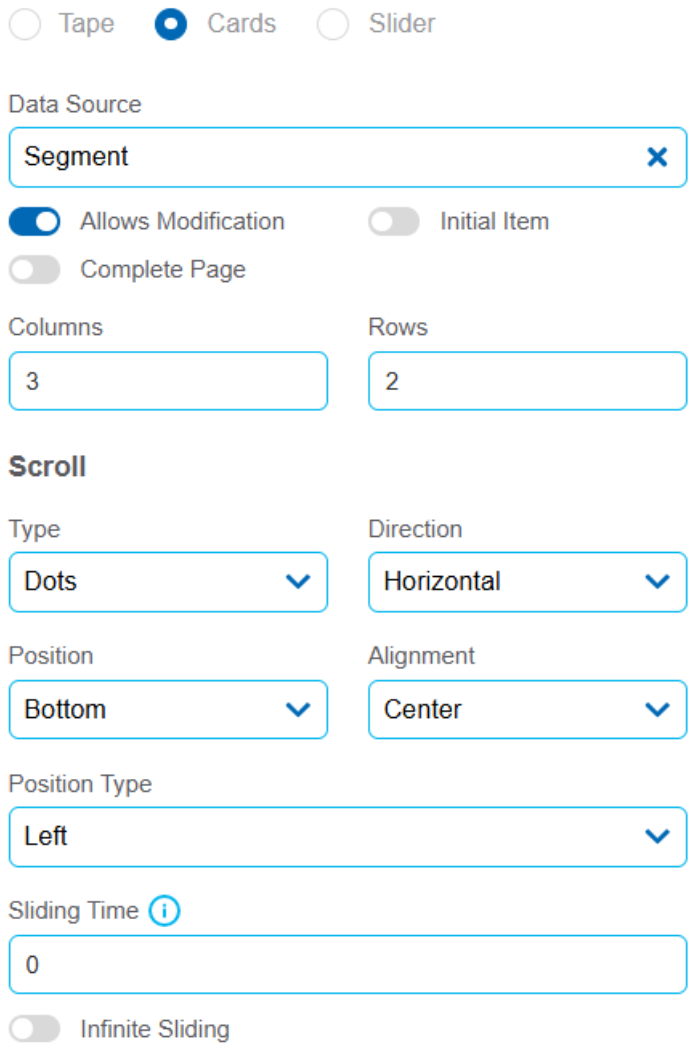
Defines the behavior of the navigation arrows when the last element is reached.

Possible Values

- Disable: The scroll arrows are disabled, indicating that there is no more content.
- Hide: Scroll arrows disappear completely, avoiding unnecessary navigation elements to be presented.

Cards

Unlike the tape, the items paginate within the repeater as the forward and backward icons are pressed. The number of items that are initially displayed in the repeater depends on the values modeled in the Columns and Rows properties.



Data Source

It allows selecting the source from which the data displayed on the repeater is retrieved. Once the source is selected, the Allows Modification property is enabled.

Allows Modification

It allows adding or removing items from the repeater using a modeled hyperlink. If the property is not activated, the number of items is fixed and cannot be modified. When activated, the Initial Item and Complete Page properties are displayed.

Initial Item

It allows adding an empty item to the repeater using a modeled hyperlink.

Complete Page

Automatically adds the necessary empty items to the repeater to reach the total quantity defined in its modeling, via a hyperlink

> *If the repeater is empty at execution, the initial item is automatically added or the defined number of items is completed and the user does not need to use the hyperlink.*

Columns
It allows defining the number of columns in the repeater.

Rows
It allows defining the number of rows in the repeater.

Type
Defines the type of control to move forward or backward between the groups of items visible in the repeater.

Possible Values

- Dots: each dot represents a group of visible items and they can be selected to change the view.
- Arrows: icons on the sides that allow moving forward or backward through groups of items..
- Pages: combines arrows with page numbers. The user can navigate between pages using arrows or access one directly. Up to two pages before and after the current one are displayed, with ellipses if there are more. The active page is highlighted to indicate the current position.

Direction
Allows to select the direction in which the items within the element are displayed.

Possible Values

- Horizontal
- Vertical

Position and Alignment
Select the position of the scroll icons within the repeater.

Possible Values for Position

- Top
- Bottom
- Inside Top
- Inside Bottom

Possible Values for Alignment

- Left
- Center
- Right

Position Type
It allows defining how cards are aligned within their container.

Possible Values

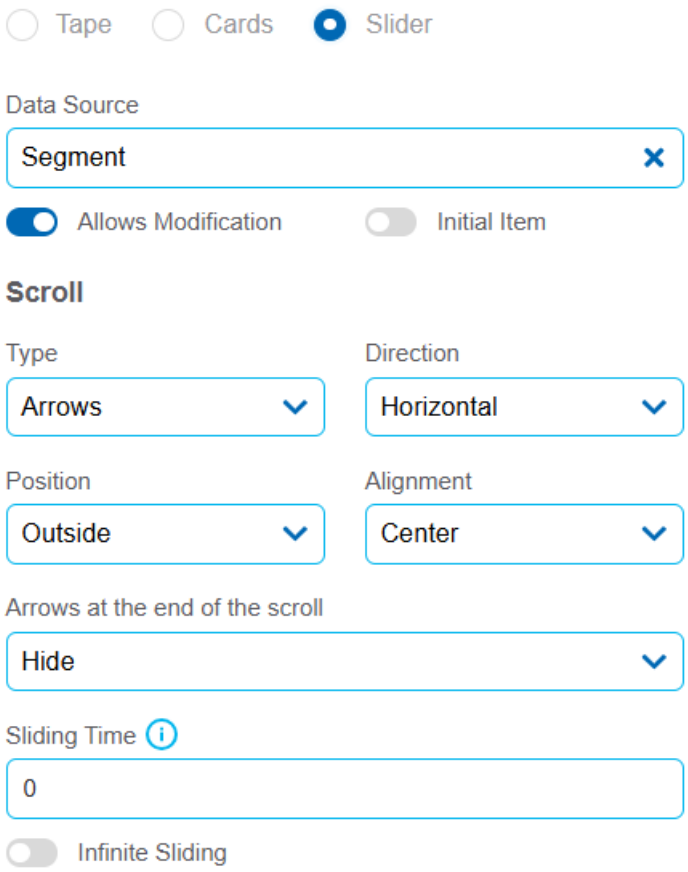
- Left
- Centered
- Right

Sliding Time
It allows defining the interval in seconds for the automatic transition between cards.

Infinite Sliding
It allows the card type repeater to slide continuously, returning to the beginning once the end is reached.

Slider

It behaves similarly to the Tape-type repeater, offering additional features to customize its behavior. It is designed to show and focus on a single element at a time.



Data Source

It allows selecting the source from which the data displayed on the repeater is retrieved. Once the source is selected, the Allows Modification property is enabled.

Allows Modification

It allows adding or removing items from the repeater using a modeled hyperlink. If the property is not activated, the number of items is fixed and cannot be modified. When activated, the Initial Item property is displayed.

Initial Item

It allows adding an empty item to the repeater using a modeled hyperlink.

> *If the repeater is empty at runtime, the initial item is automatically added, without the user needing to use the hyperlink.*

Type

Defines the type of control to move forward or backward between items in the repeater.

Possible Values

- Dots: each dot represents an item and they can be selected to change the view.
- Arrows: icons on the sides that allow moving forward or backward one item at a time.
- Pages: combines arrows with page numbers. The user can navigate between items using arrows or go directly to a specific one. Up to two items before and after the current one are displayed, with ellipses if there are more. The active item is highlighted to indicate the current position.

Direction
Allows to select the direction in which the items within the element are displayed.

Possible Values

- Horizontal
- Vertical

Position and Alignment
Select the position of the scroll icons within the repeater.
Possible Values for Position
- Top
- Bottom
- Inside
- Outside

Possible Values for Alignment
- If the "Inside" or "Outside" values are selected for the Position property, the possible values of the Alignment property are: "Top", "Center", "Bottom".
- If the "Top" or "Bottom" values are selected for the Position property, the possible values of the Alignment property are: "Left", "Center", "Right".

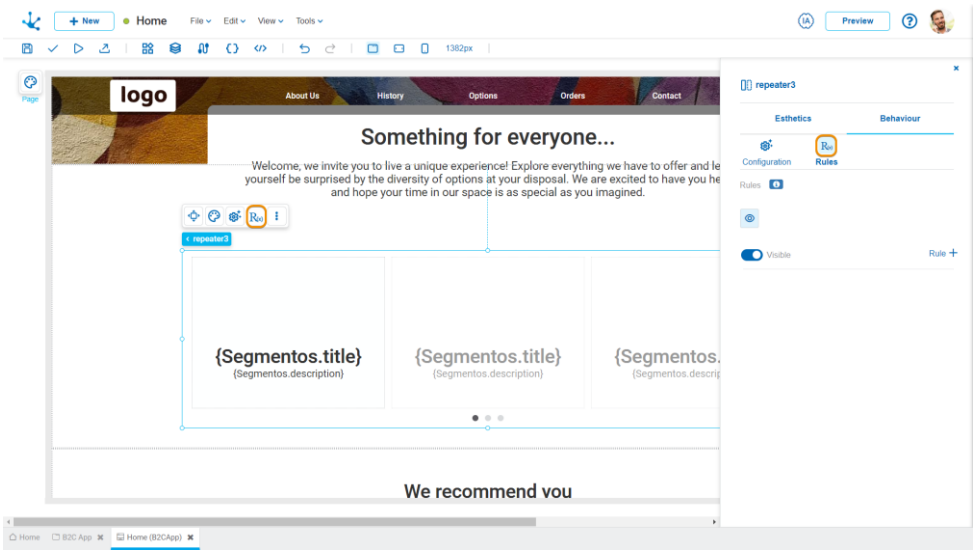
Arrows at the end of the scroll
Defines the behavior of the navigation arrows when the last element is reached.
Possible Values
- Disable: The scroll arrows are disabled, indicating that there is no more content.
- Hide: Scroll arrows disappear completely, avoiding unnecessary navigation elements to be presented.

Sliding Time
It allows defining the interval in seconds for the automatic transition between cards.

Infinite Sliding
It allows the card type repeater to slide continuously, returning to the beginning once the end is reached.

Rule Properties

The rule properties panel of an element opens when clicking the icon  of its context menu.



Properties

Rules

Embedded rules on behavior, validation and calculation can be defined, associated with an element, by using the wizard (ctrl + space).

Shows syntax examples for writing the rules.

Visible

Indicates whether the element is visible. If this property is not checked, the element is not displayed in the page.

Visible (default) Not visible

Rule + Opens an edit area where a rule to determine the visibility condition can be defined. If a rule is defined, the icon is displayed with light blue borders.

Operations when defining the rule:

Saves the new or modified rule

Cancels the operation

Operations once the rule is defined:

Edits the existing rule

Deletes the rule

Item

Items are defined within the repeater and only the first one can be modified since the changes are applied to the rest.

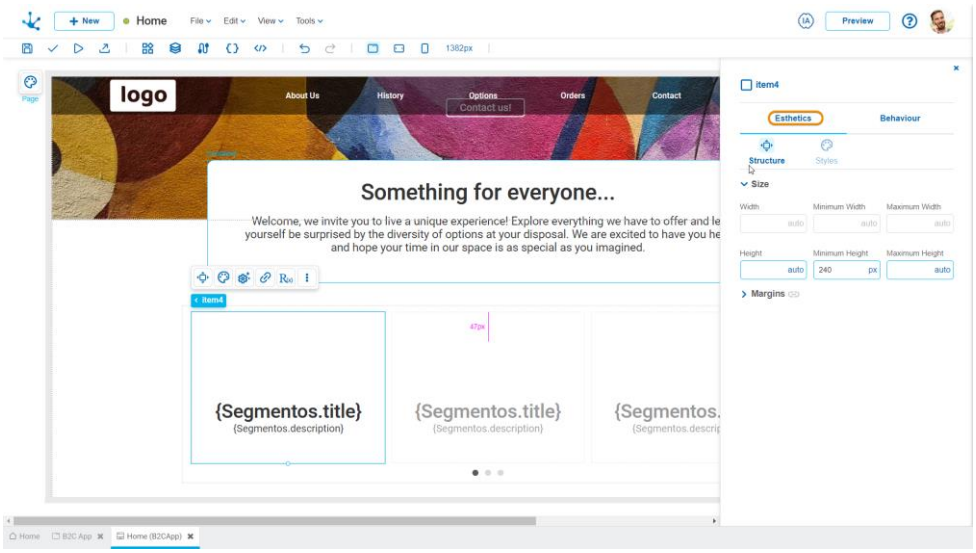
Classification of Properties

Properties are divided into two groups: esthetic properties and behavior properties.

Esthetic Properties

In the esthetic properties panel, the following are grouped:

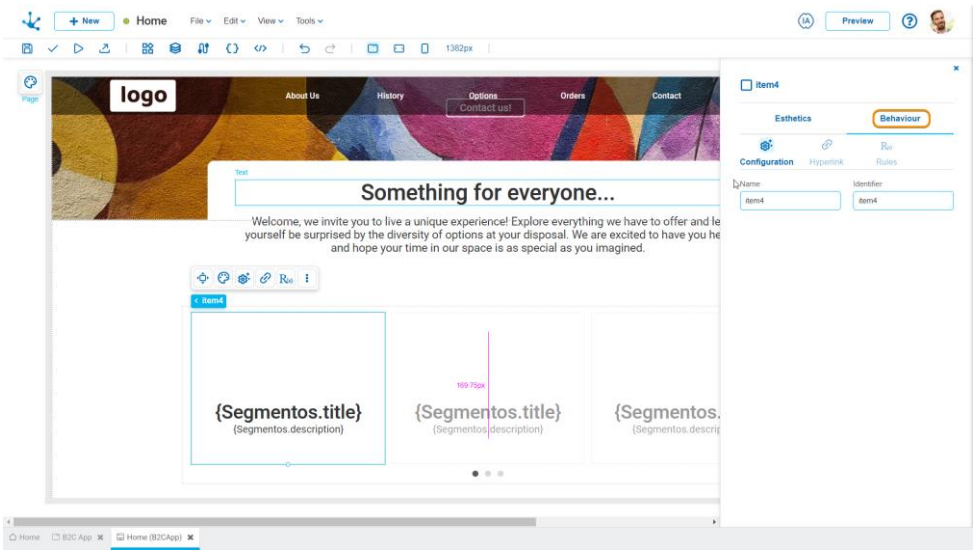
- Structure Properties
- Style Properties



Behavior Properties

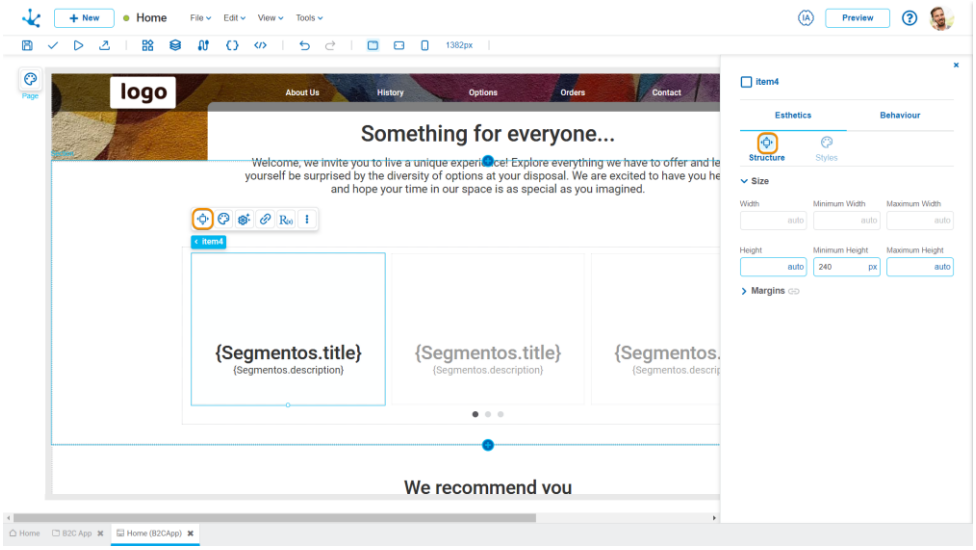
In the behavior properties panel, the following are grouped:

- Configuration Properties
- Hyperlink Properties
- Rules Properties

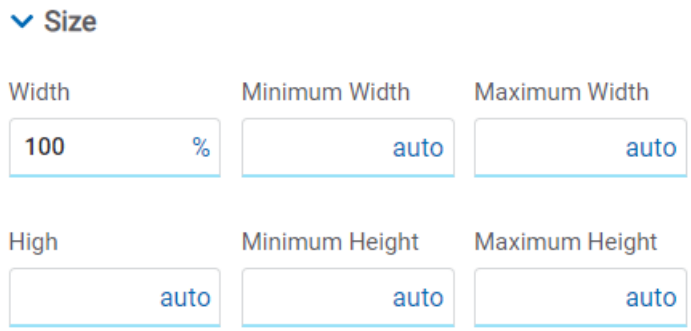


Structure Properties

The structure properties panel of an element opens when selecting the icon  of its context menu.



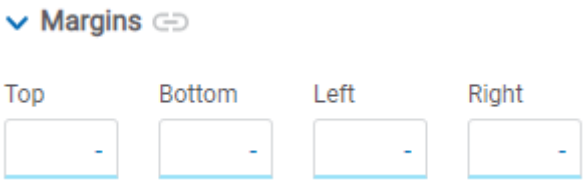
Size



All size properties can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh). Additionally, for Width and Height properties, the "auto" option is added.

> *If selecting percentage (%) for the width and height properties, the size is calculated relative to the top element.*

Margins



Allows to define the distance among items and also from items to the borders of the containing element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Top
Distance to the top border of the containing element.

Bottom
Distance to the bottom border of the containing element.

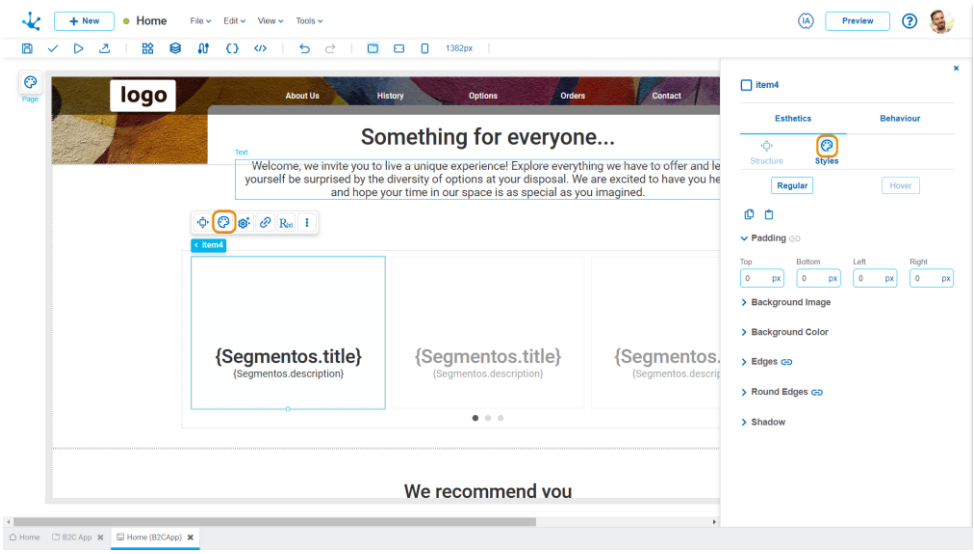
Left
Distance to the left border of the containing element or to the previous item.

Right
Distance to the right border of the containing element or to the following item.

Allows the value entered in one of the margins to be copied to the other ones automatically.
Allows to indicate different values for each margin.

Style Properties

This type of element may take different states and for each of them different values for its properties may be modeled.



This type of element may take different states and for each of them different values for its properties may be modeled.



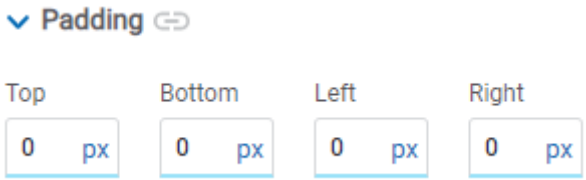
- Regular: The mouse pointer is not over the element.
- Hover: The mouse pointer is over the element.

Styles

It allows copying the entire current configuration of the style panel for the selected element (Ctrl+Alt+C).

It allows applying the previously copied configuration to another element of the same type (Ctrl+Alt+V).

Padding



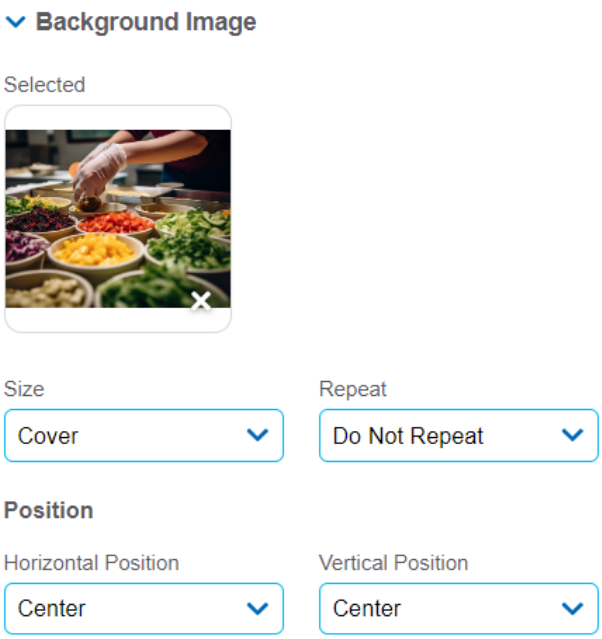
All padding space properties allow to create a space around the item borders (top, bottom, sides) and its content. The padding space configured is inward, while the margin is configured relative to a top item. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Allows values entered in one of the paddings to be copied to the other ones automatically.

Allows to indicate different values for each padding.

Background Image

It allows the addition of a background image to the item.

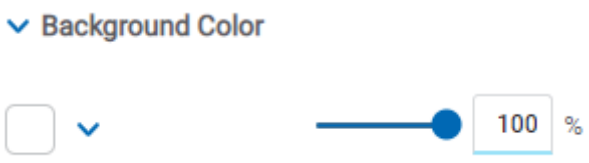


Selected
An image can be uploaded from the computer where it is being modeled.

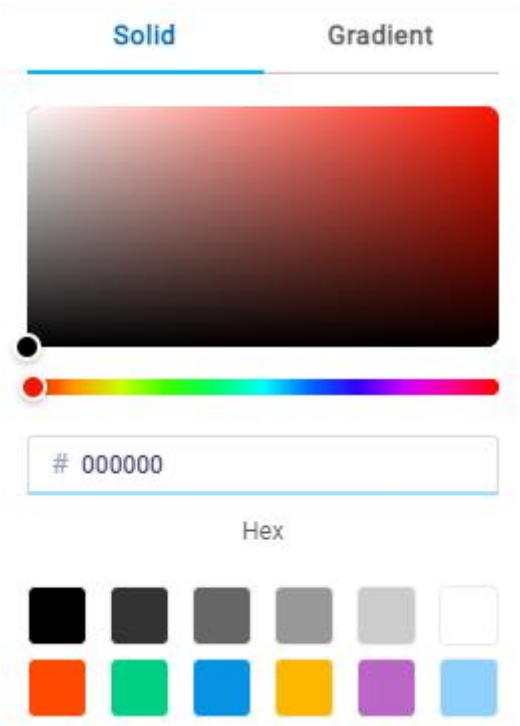
Background Color

It allows to model a background color for the element, it can be solid or gradient and different properties are defined for each one.

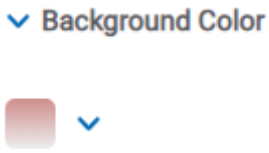
Solid



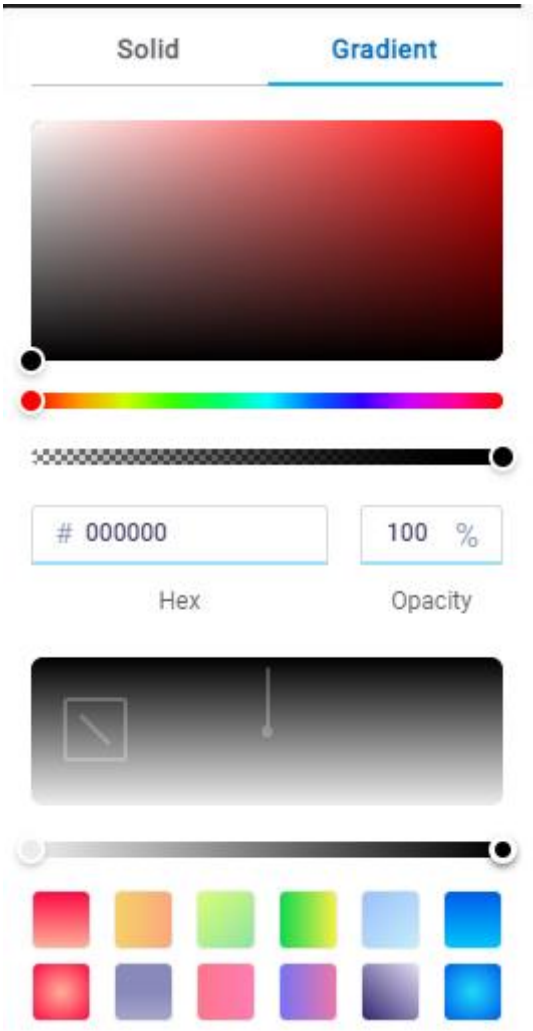
This option enables not only to select the color from a palette but also to define the degree of transparency.



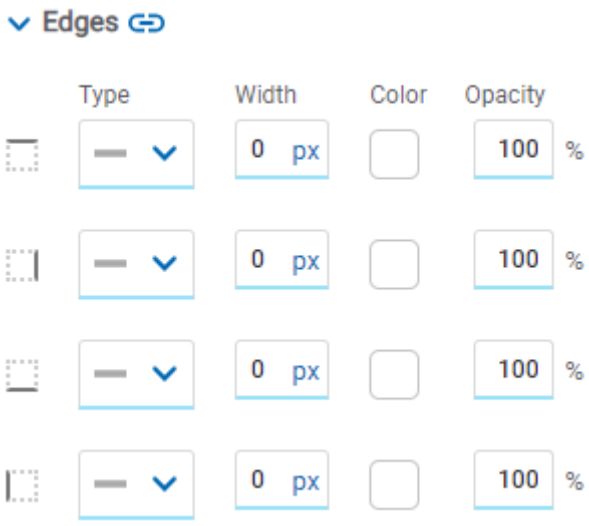
Gradient



This option enables to select not only the color from a palette but also to define its opacity and intensity according to the selected angle.



Edges



It allows to define the style of borders. Each one has its type, width, color and opacity defined, the latter as a percentage.

Allows the value entered in one of the borders to be copied to the other ones automatically.
Allows to indicate different values for each border.

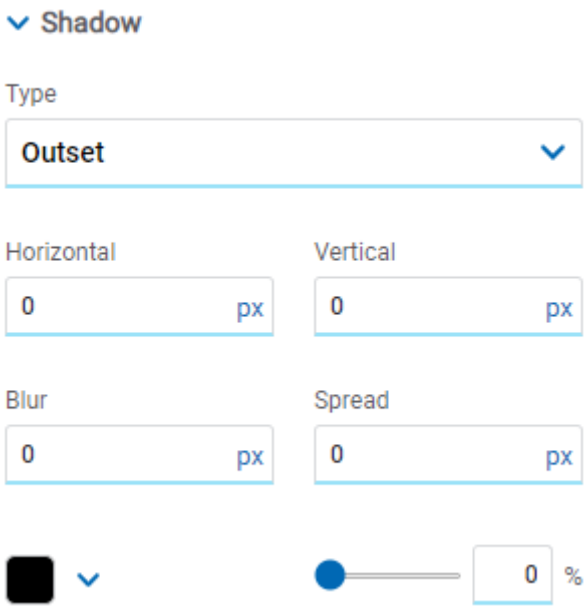
Round Edges



It allows to define the round edges at the corners of items.
It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Allows the value entered in one of the borders to be copied to the other ones automatically.
Allows to indicate different values for each border.

Shadow



Allows to define a shadow effect around the item.

Type

Possible Values

- Outset
- Inset

Horizontal
Horizontal size of the shadow to the right of the item. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Vertical
Vertical size of the shadow below the item. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

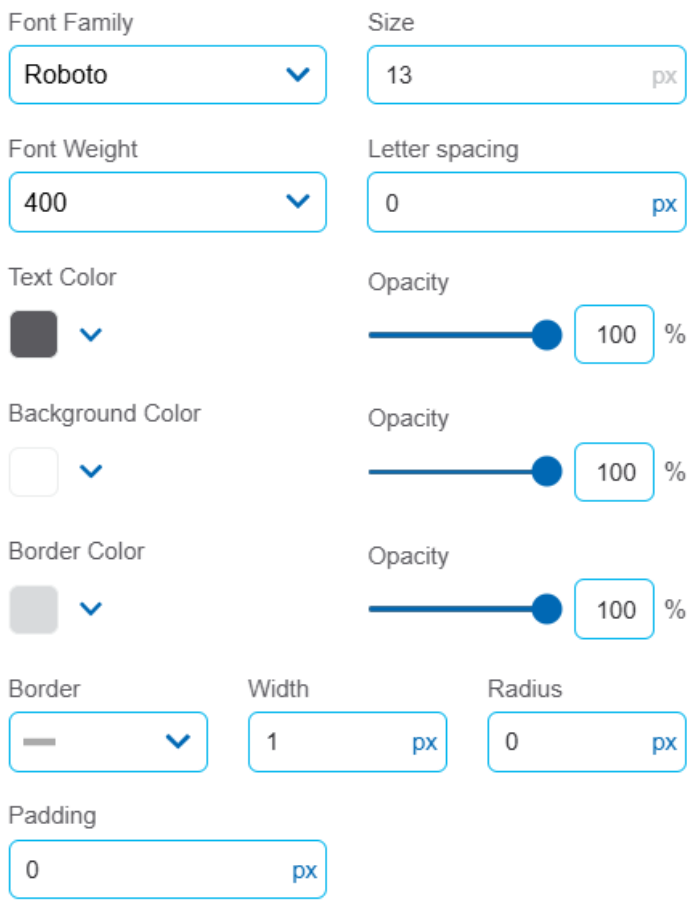
Blur
The greater this value is, the greater and lighter the shadow becomes. If not specified, its value is 0 and the shadow border is darker. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Spread
Positive values will cause the shadow to expand and grow bigger, while negative values will cause the shadow to shrink. If not specified, its value is 0 and the shadow will have the same size as the item.

Color
Defines the color of the shadow.

Transparency
Defines the transparency of the shadow.

Tooltip



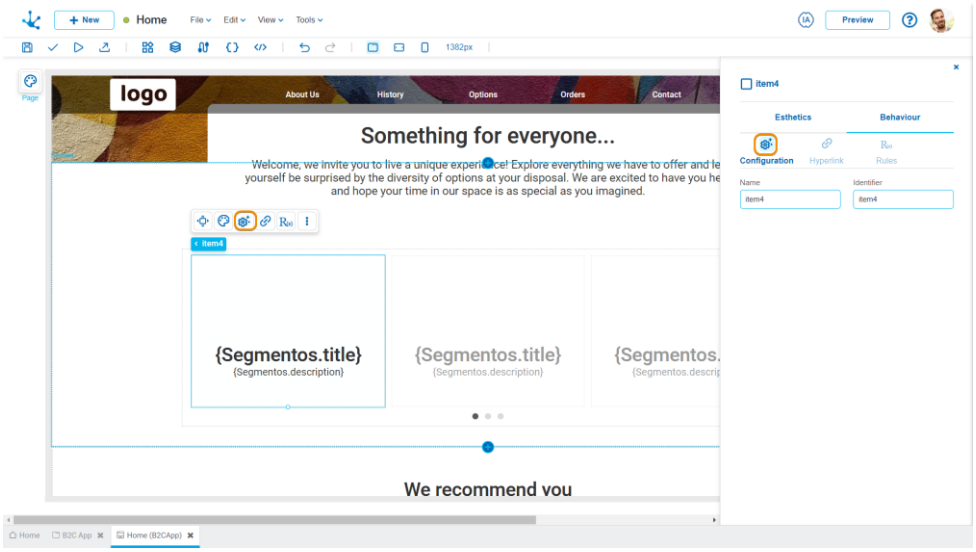
Source
It allows customizing the font, size, thickness and letter spacing of the pop-up message.

Colors
The color and opacity of the text, background and border can be defined. Borders are set uniformly for all sides of the tooltip.

Border
The border type, width, and border radius can be configured.

Configuration Properties

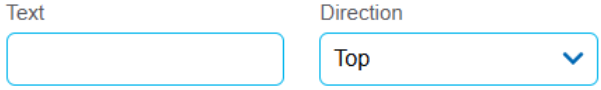
The configuration properties panel of an element opens when clicking the icon of its context menu.



Name
Name used to reference the element during modeling.

Identifier
Uniquely identifies the element. It is used in the Javascript SDK.

Tooltip
Pop-up message that appears when hovering over an element, providing additional information.



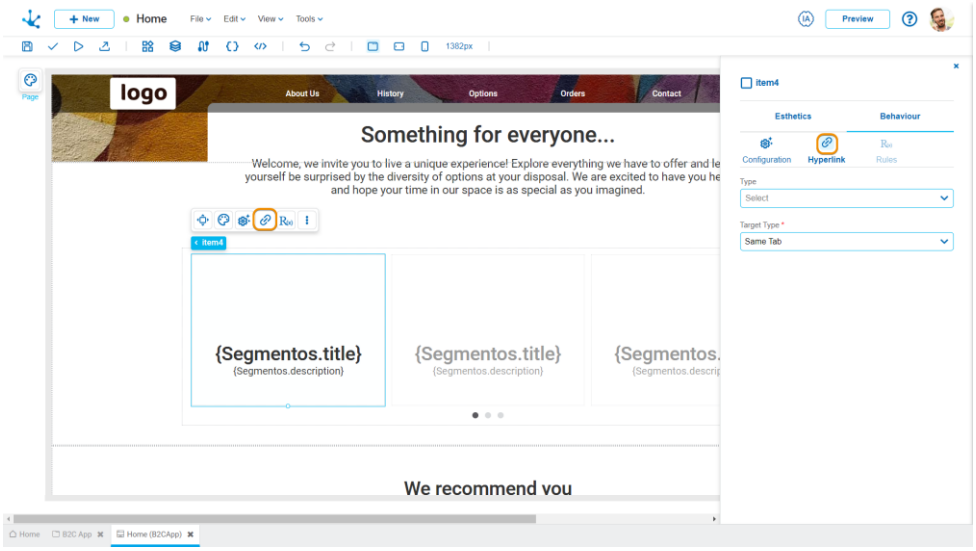
Text
Content of the message being displayed.

Direction

Position of the message in relation to the element (top, bottom, left, or right).

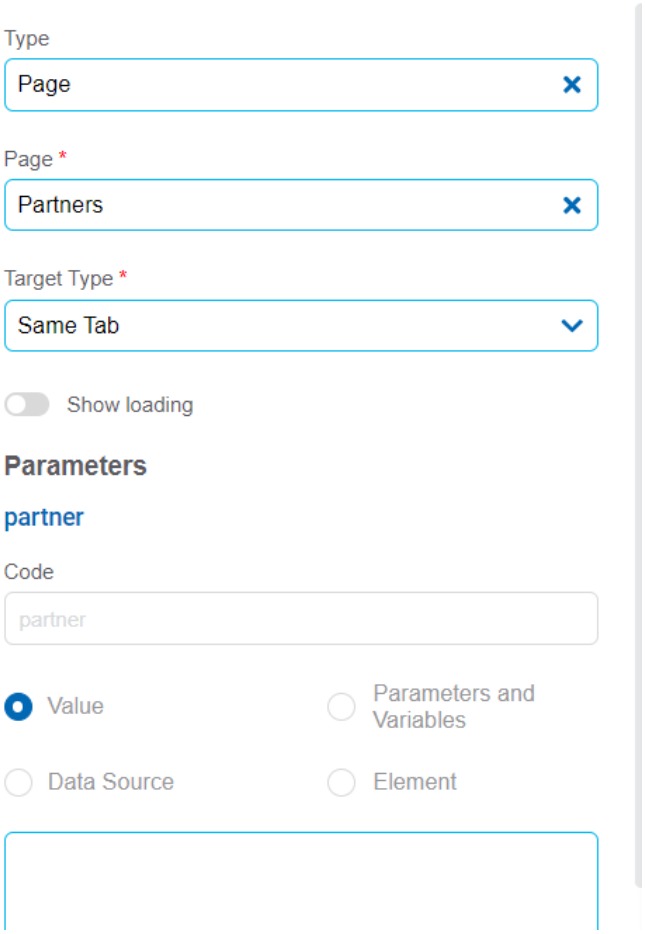
Hyperlink Properties

The hyperlink properties panel of an element opens when clicking the icon  of its context menu.



Define the behavior of the element when selecting it. Different properties are enabled depending on the type of object selected.

Page



Page
The pages modeled in the environment are displayed.

Destination Type
The available options for opening the object are displayed.

Possible Values

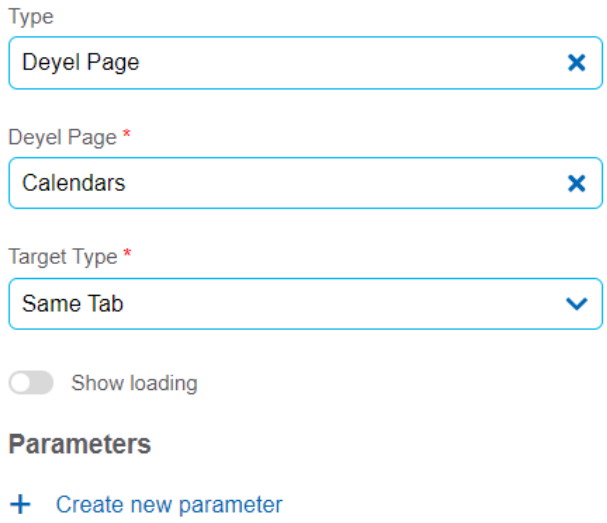
- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  Modal Horizontal Size
  Defines its width.
  Modal Vertical Size
  Defines its height.
- Iframe: selecting this option enables an additional property.
  Destination Iframe
  Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading

Shows the loading icon and disables the user interaction until the redirection is over.

Parameters
If there are parameters, their value can be specified with text, parameters and variables or data source.

Deyel Page



Deyel Page
The pages belonging to **Deyel** are displayed.

Destination Type
The available options for opening the object are displayed.

Possible Values

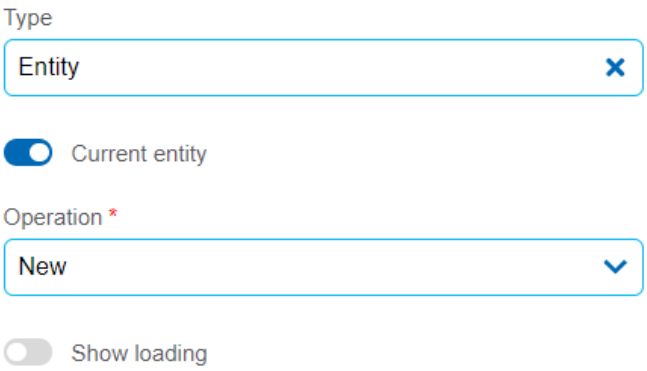
- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  - Modal Horizontal Size
    Defines its width.
  - Modal Vertical Size
    Defines its height.
- Iframe: selecting this option enables an additional property.
  - Destination Iframe
    Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Parameters

Allows sending parameters to the selected object type.

Entity



Current Entity

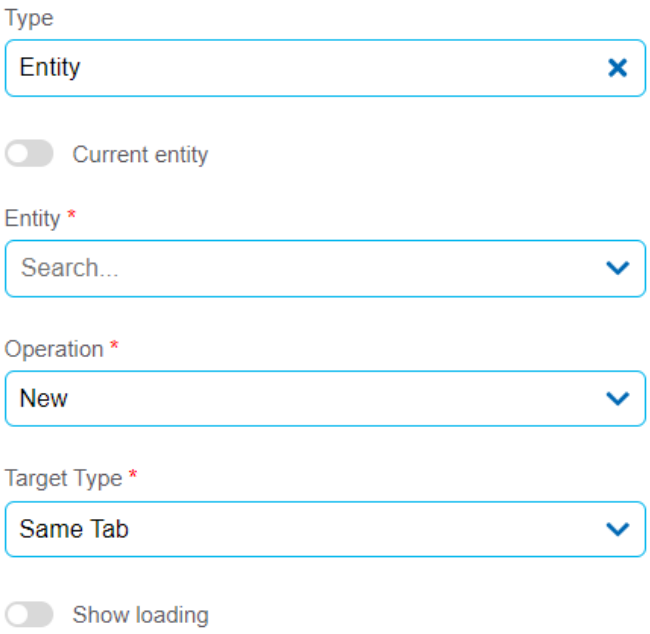
If this property is checked, the operation applies directly to the entity being modeled.

Operation

Defines the operation made when selecting the object.

Possible Values

- New: Indicates that the current entity panel is opened for the creation of an instance.
- Modify: Indicates that the current entity panel is opened for the update of an instance.
- Delete: Indicates that the current entity panel is opened for the deletion of an instance.



Entity
If the property Current Entity, is not checked, the entities modeled in the environment are displayed.

Operation
Defines the operation made when selecting the object.

Possible Values

- New: The panel of the entity selected in the previous property is opened to create an instance.
- Update The panel of the entity selected in the previous property is opened to update an instance.
- Show: The panel of the entity selected in the previous property is opened to show an instance.
- Grid: Indicates that the grid of the entity selected in the previous property is opened.

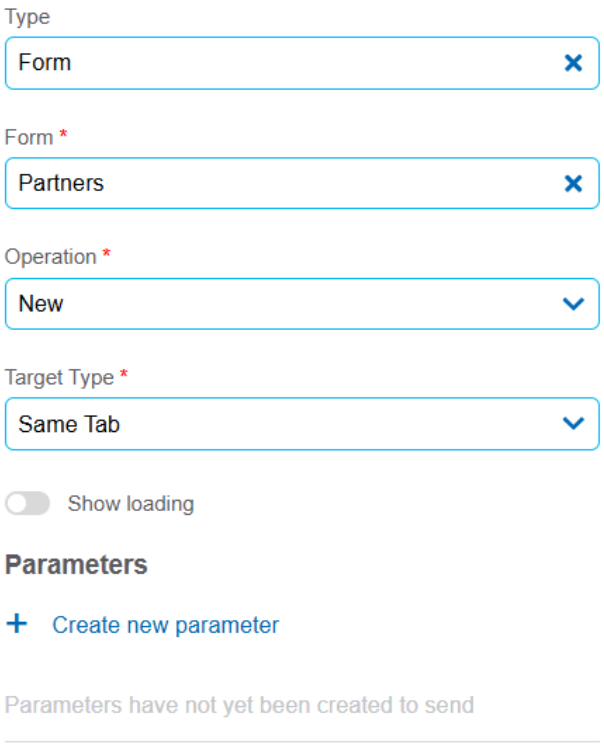
Destination Type
The available options for opening the object are displayed.

Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  - Modal Horizontal Size
    Defines its width.
  - Modal Vertical Size
    Defines its height.
- Iframe: selecting this option enables an additional property.
  - Destination Iframe
    Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Forms



Forms
The forms modeled in the environment are displayed.

Operation
Defines the operation made when selecting the object.

Possible Values

- Create: Indicates that the panel of the object selected in the previous property is opened for the creation of an instance.
- Update: Indicates that the panel of the object selected in the previous property is opened for the update of an instance.
- Show: Indicates that the panel of the object selected in the previous property is opened for the query of an instance.
- Grid: Indicates that the grid of the object selected in the previous property is opened.

Destination Type
The available options for opening the object are displayed.

Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  - Modal Horizontal Size
    Defines its width.
  - Modal Vertical Size
    Defines its height.
- Iframe: selecting this option enables an additional property.
  - Destination Iframe

Expands the iframes previously defined on the object.

- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

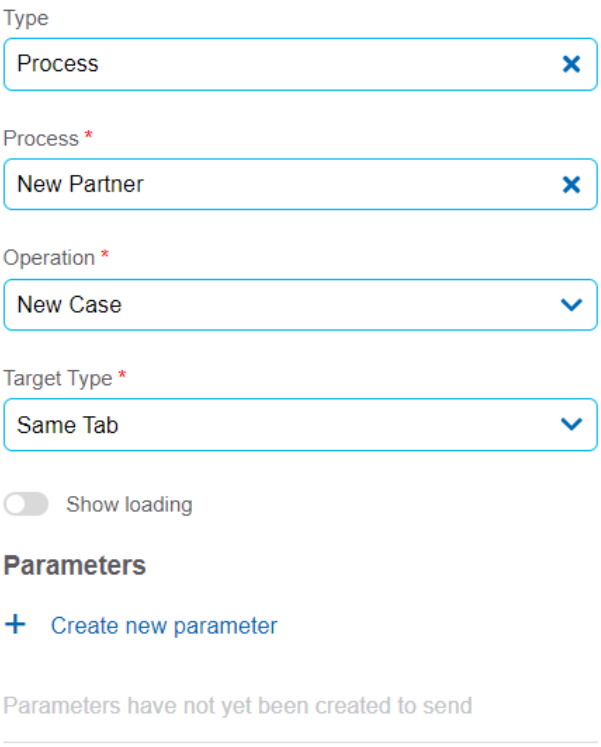
Show loading

Shows the loading icon and disables the user interaction until the redirection is over.

Parameters

Allows sending parameters to the selected object type.

Process



Process

The processes modeled in the environment are displayed.

Operation

Defines the operation made when selecting the object.

Destination Type

The available options for opening the object are displayed.

Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.

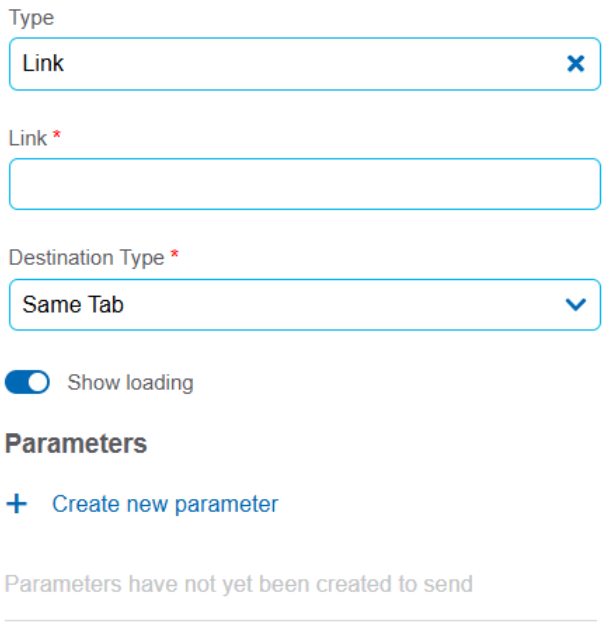
Modal Horizontal Size
Defines its width.
Modal Vertical Size
Defines its height.

- Iframe: selecting this option enables an additional property.
  Destination Iframe
  Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Parameters
Allows sending parameters to the selected object type.

Link



Link
Allows to enter any link.

Destination Type
The available options for opening the object are displayed.

Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  Modal Horizontal Size

Defines its width.

Modal Vertical Size

Defines its height.

- Iframe: selecting this option enables an additional property.

  Destination Iframe

  Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

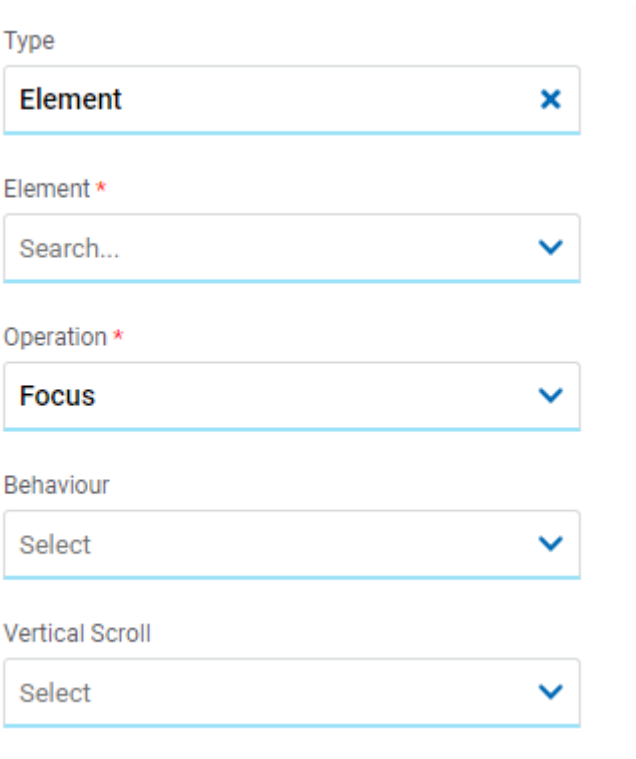
Show loading

Shows the loading icon and disables the user interaction until the redirection is over.

Parameters

Allows sending parameters to the selected object type.

Element



Element

The modeled elements in the object are displayed.

Operation

Defines the operation made when selecting the element.

Possible Values

- Focus
- Show
- Hide

- Show/Hide

Behaviour
Establishes the transition of the animation. It is only modelable if the "Focus" operation was selected.

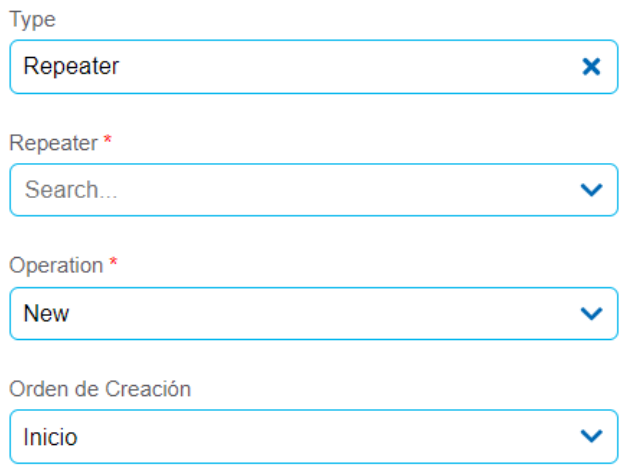
Possible Values

- Auto
- Smooth

Vertical Scroll
It is only modelable if the "Focus" operation was selected.
Possible Values

- Start: Moves to the start of the selected element.
- Closest: Moves to the closest position to the selected element from the element the event fires.
- Center: Moves to the center of the selected element.
- End: Moves to the end of the selected element.

Repeater



Repeater
The modeled repeaters from the object are displayed.

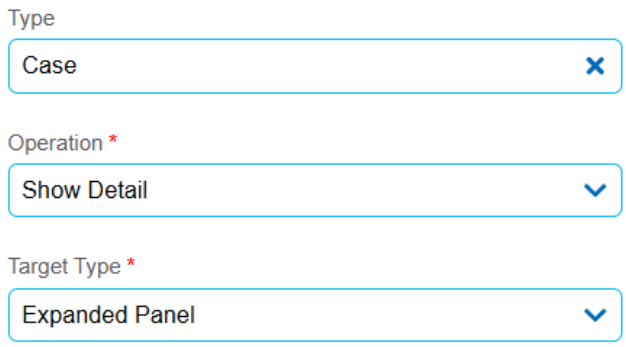
Operation
Defines the operation made when selecting the element.

Possible Values

- New: Allows for the creation of an item within the repeater. If this option is selected the additional properties are enabled.
  ### Creation Order
  It indicates in which position the new item of the container is added.
  Possible Values
  - Home
  - End

- Delete: Allows for the deletion of an item within the repeater.

Case



Case
Show case information.

Operation
Defines the operation made when selecting the element.

Possible Values

- Show Detail
- Show Graphic Execution

Destination Type
The available options for opening the object are displayed.

Possible Values

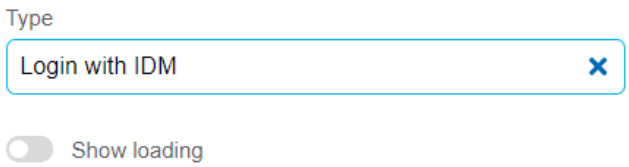
- Expanded Panel
- Modal: if this option is selected the additional properties are enabled.
  Modal Horizontal Size
  Define its width.
  Modal Vertical Size
  Define its height.

Back



It allows associating the event to go back in the browser to the element.

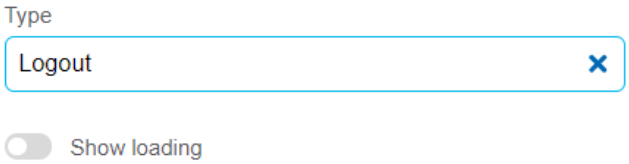
LogIn with IDM



Allows login with IDM.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

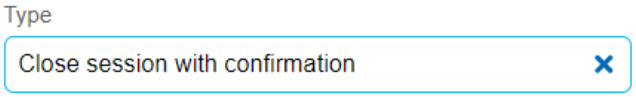
Logout



Allows logging the user out.

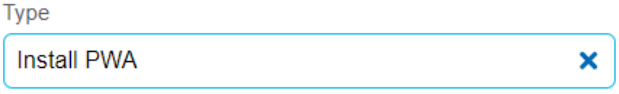
Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Close Session with Confirmation



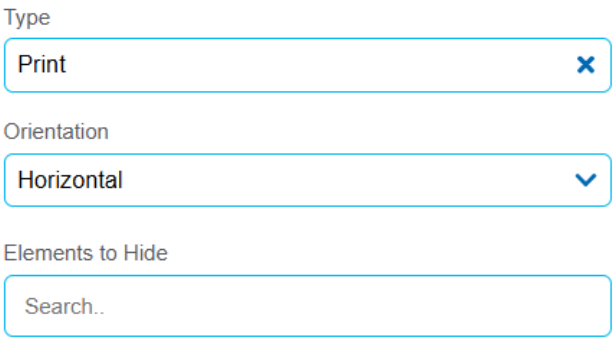
Displays a confirmation modal to logout the user.

Install PWA



Allows to install the application in the browser.

Print



This type of hyperlink associated with an element allows printing the content of the entity or page that is being executed.

Orientation

Determines the layout of the content to be printed.

Possible Values

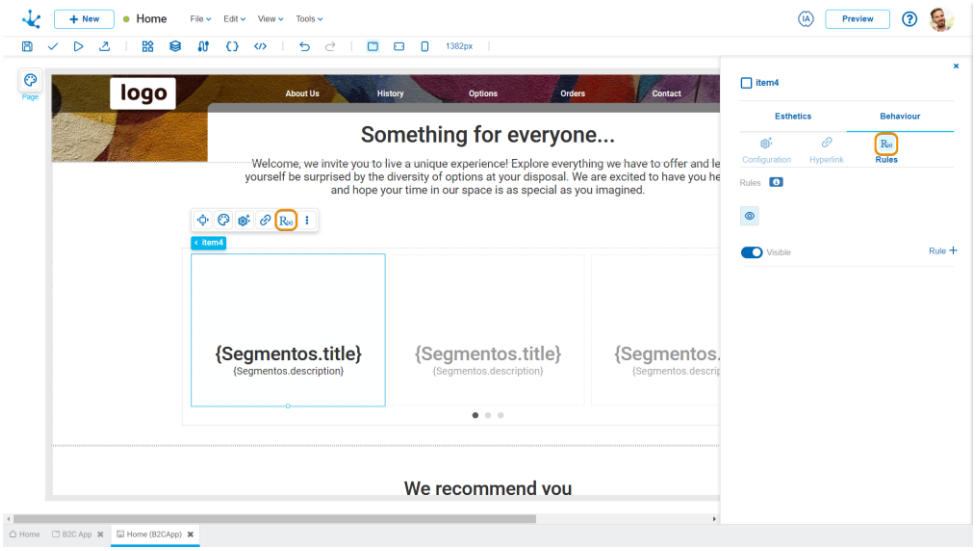
- Horizontal
- Vertical

Elements to hide

It allows selecting which elements of the entity or page should not be included in the printout.

Rule Properties

The rule properties panel of an element opens when clicking the icon  of its context menu.



Properties

Rules

Embedded rules on behavior, validation and calculation can be defined, associated with an element, by using the wizard (ctrl + space).

Shows syntax examples for writing the rules.

Visible

Indicates whether the element is visible. If this property is not checked, the element is not displayed in the page.

Visible (default) Not visible

Rule + Opens an edit area where a rule to determine the visibility condition can be defined. If a rule is defined, the icon is displayed with light blue borders.

Operations when defining the rule:

Saves the new or modified rule

Cancels the operation

Operations once the rule is defined:

Edits the existing rule

Deletes the rule

Events

Items allow the use of different events.

Event	Description
onMouseIn()	It is executed when the cursor is positioned on the element.
onMouseOut()	It is executed when the cursor moves out of the element.
onClick()	It is executed when clicking on the element.
onDoubleClick()	It is executed when clicking twice on the element.
onInit()	It is executed before the element is loaded.
afterViewInit()	It is executed after viewing the element.
onViewportEnter()	It is executed when the element is visible.
onViewportLeave()	It is executed when the element is no longer visible.

Icon

Allows to incorporate any icon type.

The element properties are represented by icons in its context menu, where its operations are also available.

Subtypes

By selecting the "Icon" option from the icon on the top toolbar, a list with the different subtypes of this element is displayed, which can be dragged to the modeling area. Each subtype has modeled element properties, predefined in a specific way.

- Small Icon
- Medium Icon
- Large Icon

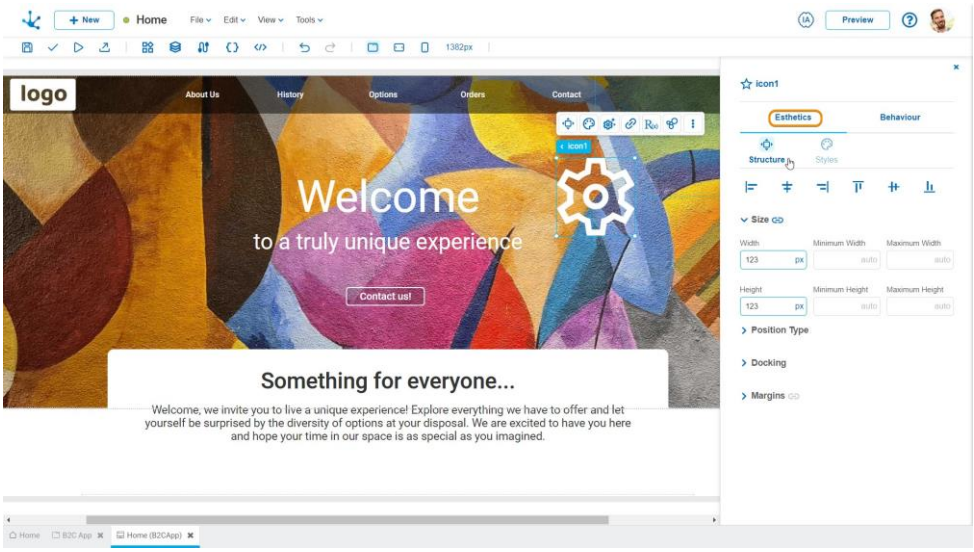
Classification of Properties

Properties are divided into two groups: esthetic properties and behavior properties.

Esthetic Properties

In the esthetic properties panel, the following are grouped:

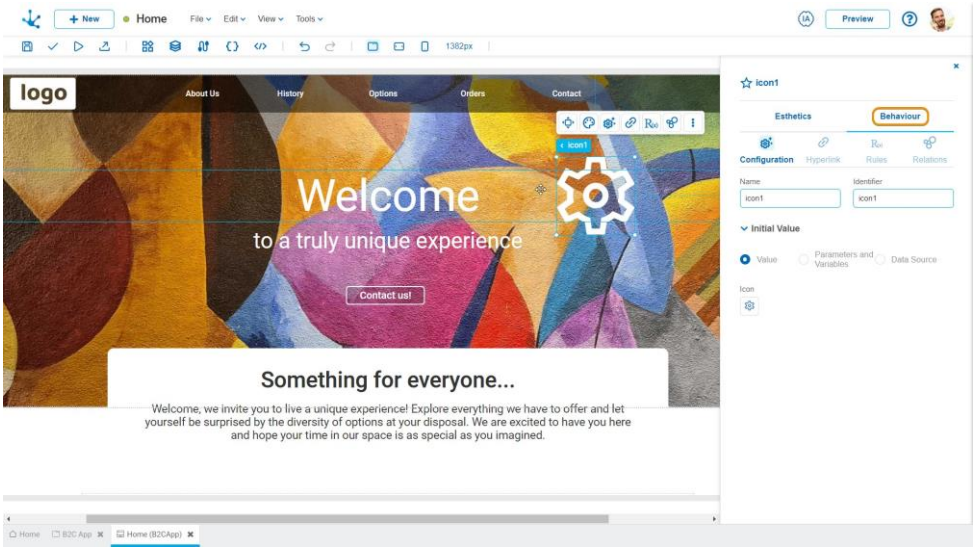
- Structure Properties
- Style Properties



Behavior Properties

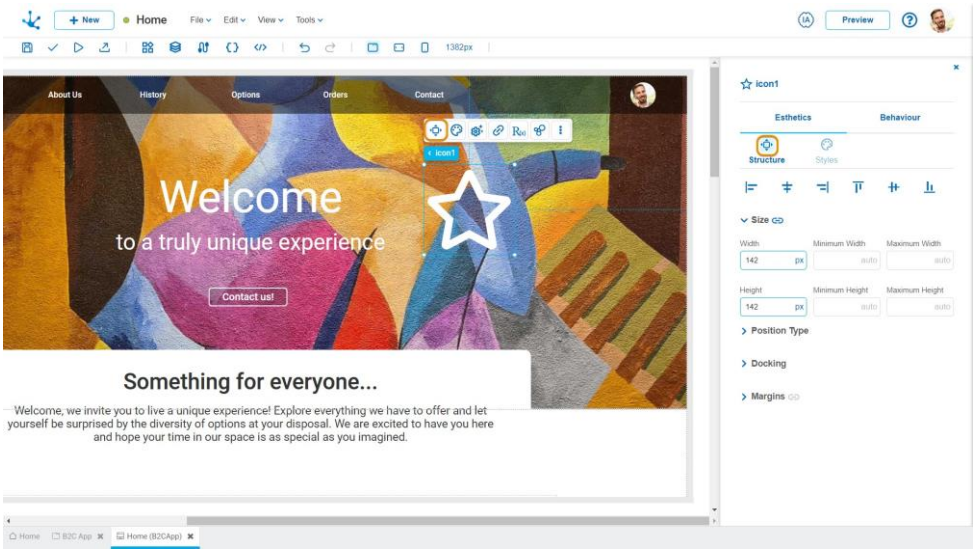
In the behavior properties panel, the following are grouped:

- Configuration Properties
- Hyperlink Properties
- Rules Properties
- Relations Properties



Structure Properties

The structure properties panel of an element opens when selecting the icon  of its context menu.



Alignment

Allows to align the element by selecting one of the available icons.

- Align to the left.
- Align to horizontal center.
- Align to the right.
- Align up.
- Align vertical center.
- Align bottom.

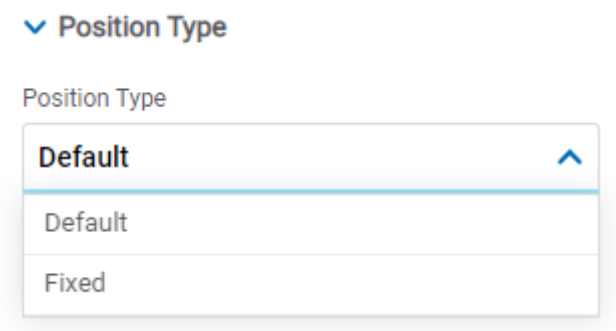
Size



It allows the input of Width and Height. The value entered in one of the properties is automatically copied to the other. These properties are expressed in pixels.

Position Type

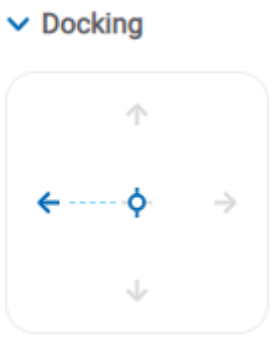
It determines if at the time of execution the element remains fixed on the page or if it moves as the scroll bar moves forward or backward.



Possible Values

- Default: The element has a relative position with respect to the superior element where it was placed (container or section).
- Fixed: Fixes the element to the page so that it is always visible in the same position. It allows to increase the visibility of important content.

Docking



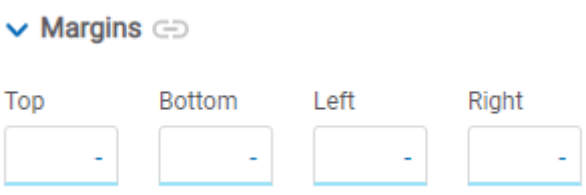
Determines the vertical and horizontal position of elements within the page section or container, when the screen is resized.

When an element is added or moved within the section or container, the modeler automatically docks it to the nearest corners or edges. It can also be configured by clicking on the arrows of the side to be docked to or on the center.

Once the element is docked, its exact position is indicated in the Margins property. The direction of the docking arrows determine the values that are automatically displayed for margins.

The docking position is indicated in the modeling area by dotted lines on the element.

Margins



It allows to define the distance of elements from the borders of their top element. The behavior of margins depends on the docking of the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Top
Distance to the top border of the highest ranking element.

Bottom
Distance to the bottom border of the highest ranking element.

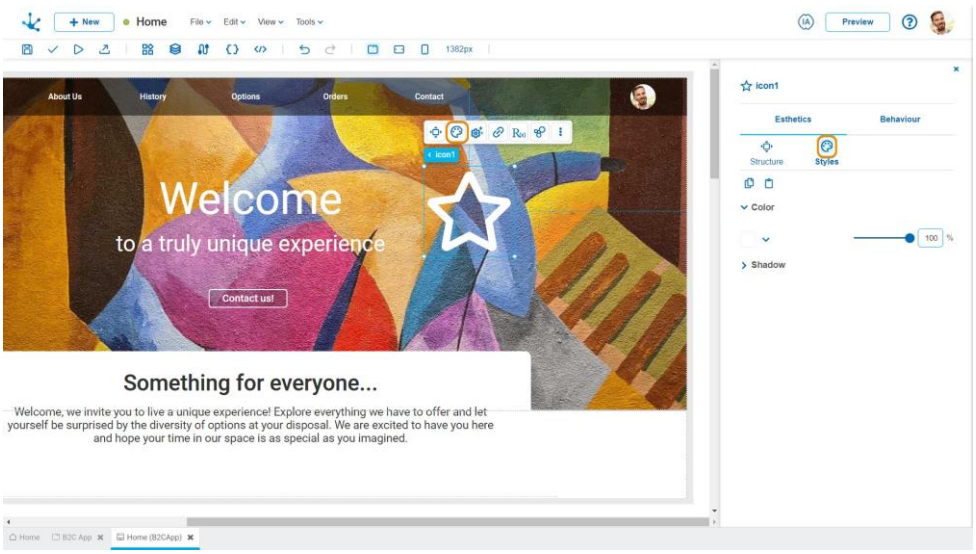
Left
Distance to the left border of the highest ranking element.

Right
Distance to the right border of the highest ranking element.

Allows the value entered in one of the margins to be copied to the other ones automatically.
Allows to indicate different values for each margin.

Style Properties

The style properties panel of an element opens when selecting the icon of its context menu.



Styles

It allows copying the entire current configuration of the style panel for the selected element (Ctrl+Alt+C).

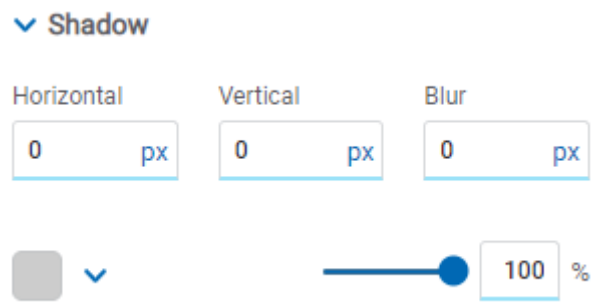
It allows applying the previously copied configuration to another element of the same type (Ctrl+Alt+V).

Color



It allows to define the color of the element.

Shadow



Allows defining a shadow effect around the element.

Horizontal
Horizontal size of the shadow to the right of the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

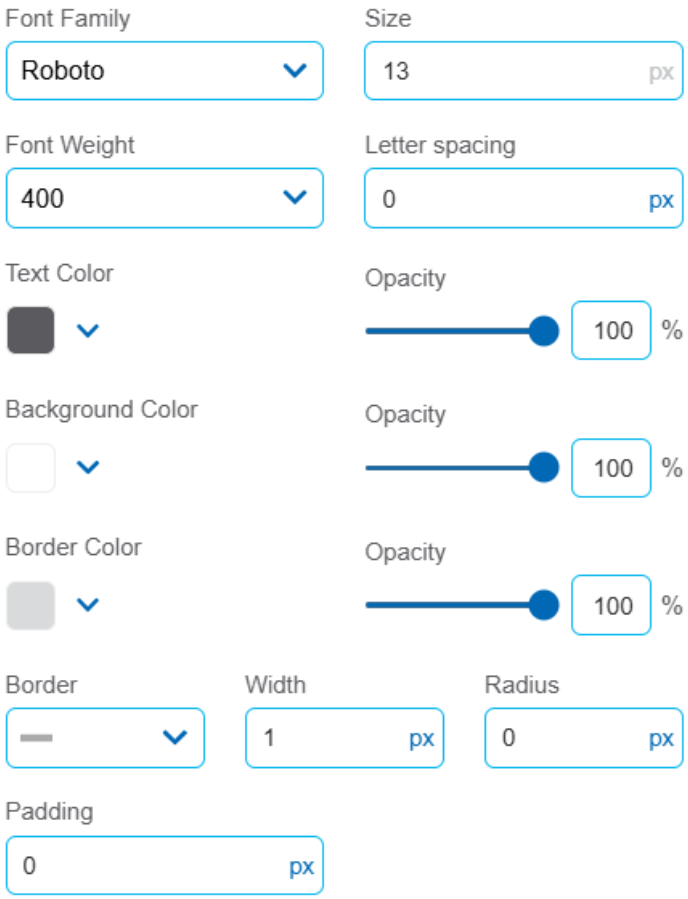
Vertical
Vertical size of the shadow below the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Blur
The greater this value is, the greater and lighter the shadow becomes. If not specified, its value is 0 and the shadow border is darker. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Color
Defines the color of the shadow.

Transparency
Defines the transparency of the shadow.

Tooltip



Source

It allows customizing the font, size, thickness and letter spacing of the pop-up message.

Colors

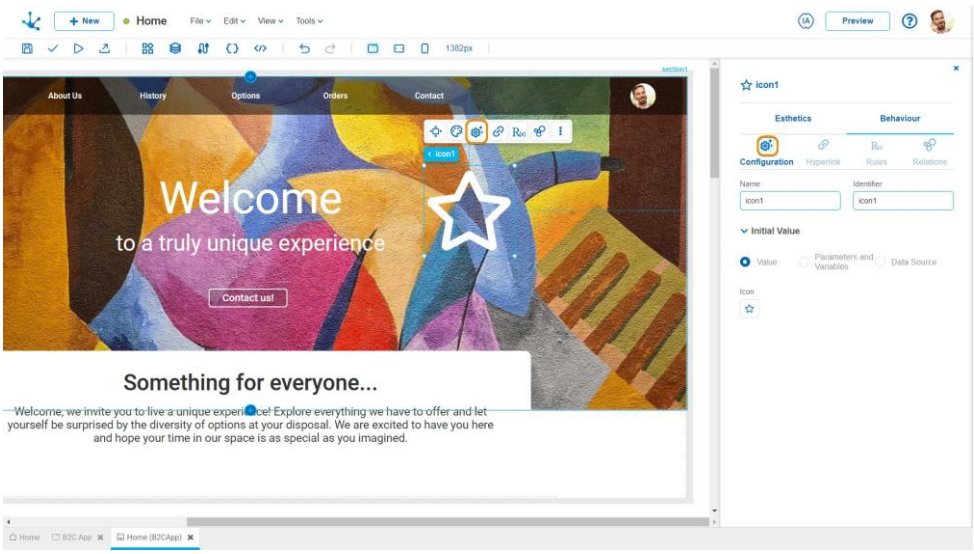
The color and opacity of the text, background and border can be defined. Borders are set uniformly for all sides of the tooltip.

Border

The border type, width, and border radius can be configured.

Configuration Properties

The configuration properties panel of an element opens when clicking the icon  of its context menu.



Name
Name used to reference the element during modeling.

Identifier
Uniquely identifies the element. It is used in the Javascript SDK.

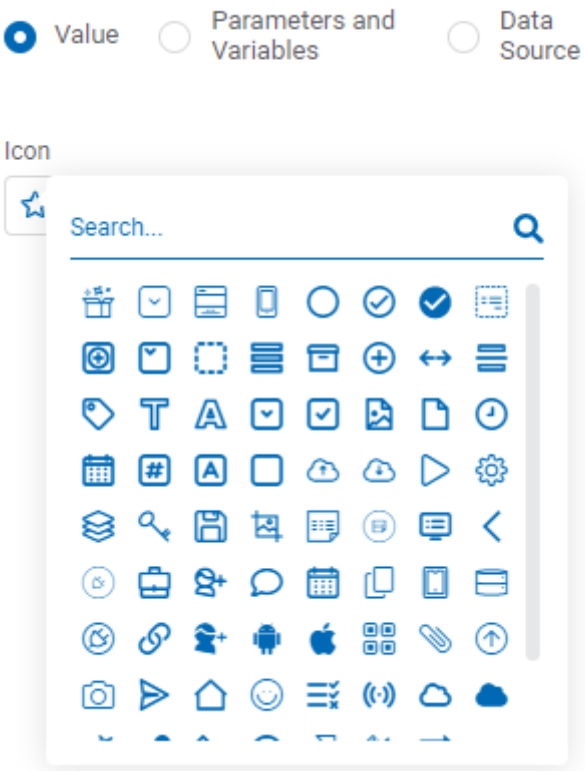
Tooltip
Pop-up message that appears when hovering over an element, providing additional information.



Text
Content of the message being displayed.

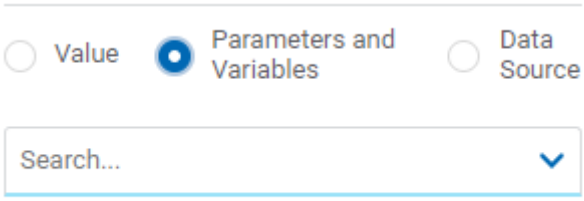
Direction
Position of the message in relation to the element (top, bottom, left, or right).

Initial Value



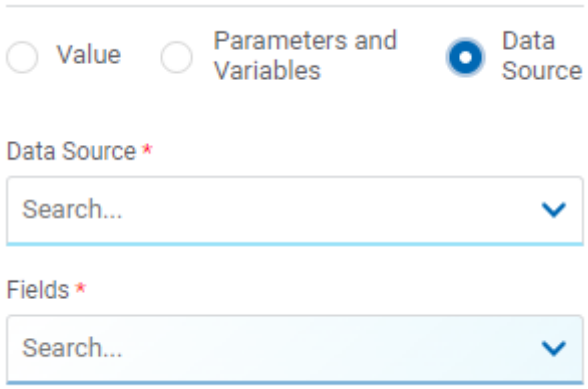
It allows selecting an icon from a palette.

Parameters and Variables



Parameters and Variables
Allows to select a parameter or a variable from, whose value is displayed in the element.
This value can be the code that represents the icon or a valid url.

Data Source



Data Source

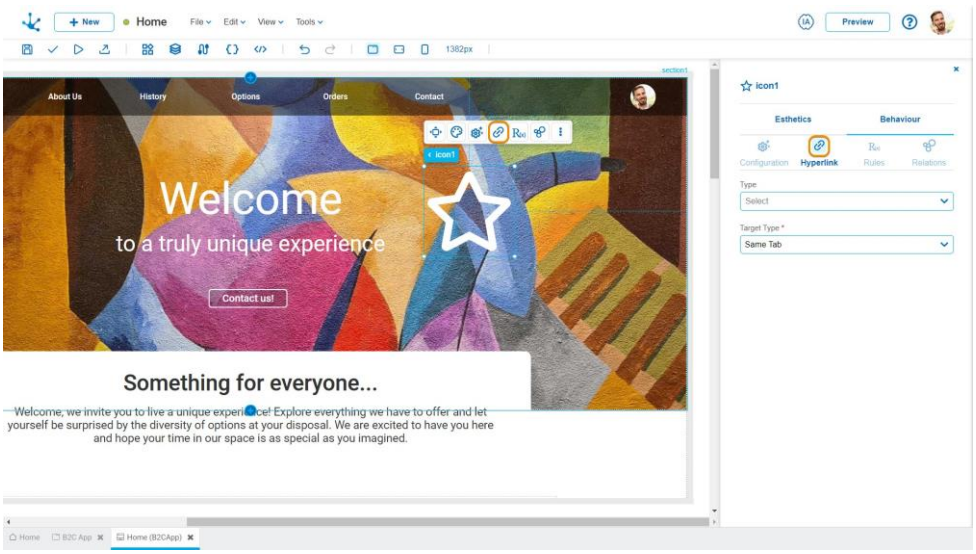
It allows selecting a previously defined data source within the object.

Fields

It allows selecting a field from the data retrieved in the chosen data source, whose value is displayed in the element.

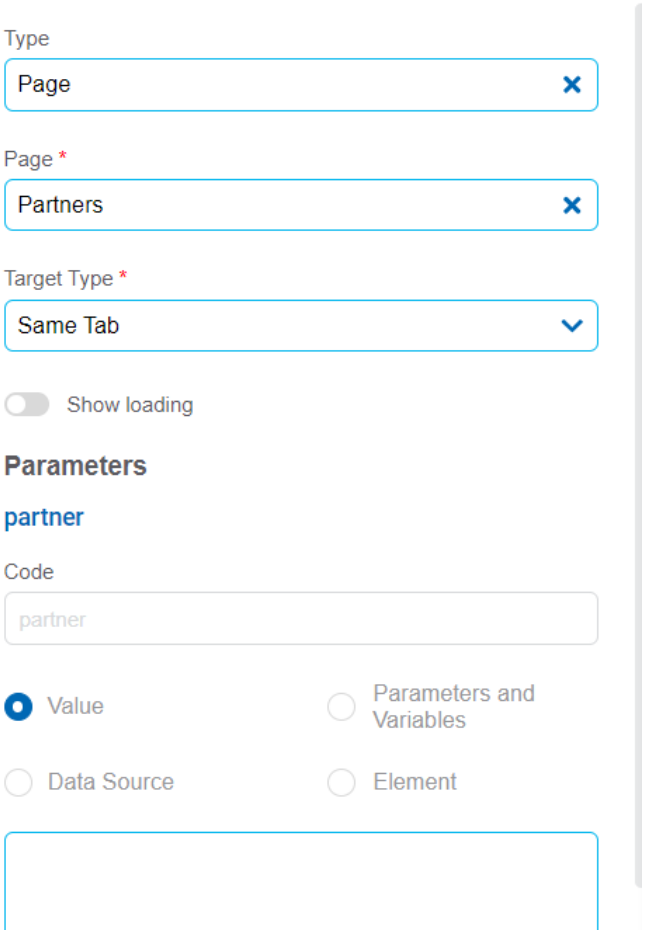
Hyperlink Properties

The hyperlink properties panel of an element opens when clicking the icon of its context menu.



Define the behavior of the element when selecting it. Different properties are enabled depending on the type of object selected.

Page



Page
The pages modeled in the environment are displayed.

Destination Type
The available options for opening the object are displayed.

Possible Values

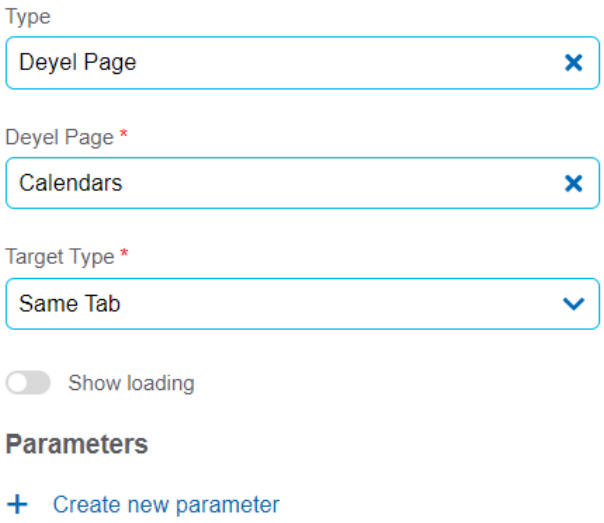
- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  Modal Horizontal Size
  Defines its width.
  Modal Vertical Size
  Defines its height.
- Iframe: selecting this option enables an additional property.
  Destination Iframe
  Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading

Shows the loading icon and disables the user interaction until the redirection is over.

Parameters
If there are parameters, their value can be specified with text, parameters and variables or data source.

Deyel Page



Deyel Page
The pages belonging to **Deyel** are displayed.

Destination Type
The available options for opening the object are displayed.

Possible Values

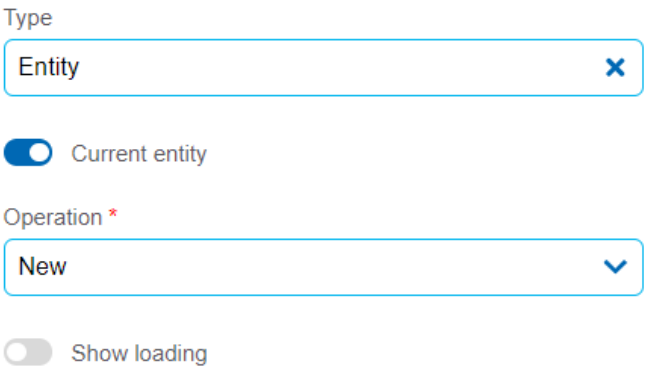
- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  - Modal Horizontal Size
    Defines its width.
  - Modal Vertical Size
    Defines its height.
- Iframe: selecting this option enables an additional property.
  - Destination Iframe
    Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Parameters

Allows sending parameters to the selected object type.

Entity



Current Entity

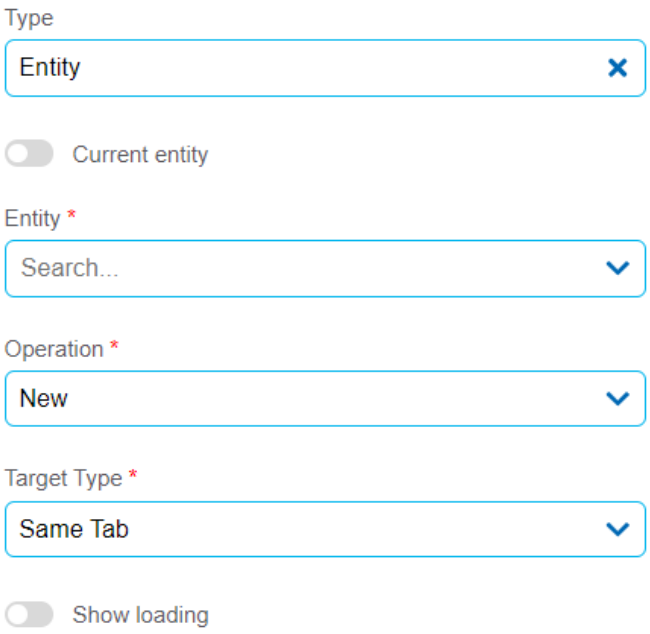
If this property is checked, the operation applies directly to the entity being modeled.

Operation

Defines the operation made when selecting the object.

Possible Values

- New: Indicates that the current entity panel is opened for the creation of an instance.
- Modify: Indicates that the current entity panel is opened for the update of an instance.
- Delete: Indicates that the current entity panel is opened for the deletion of an instance.



Entity
If the property Current Entity, is not checked, the entities modeled in the environment are displayed.

Operation
Defines the operation made when selecting the object.

Possible Values

- New: The panel of the entity selected in the previous property is opened to create an instance.
- Update The panel of the entity selected in the previous property is opened to update an instance.
- Show: The panel of the entity selected in the previous property is opened to show an instance.
- Grid: Indicates that the grid of the entity selected in the previous property is opened.

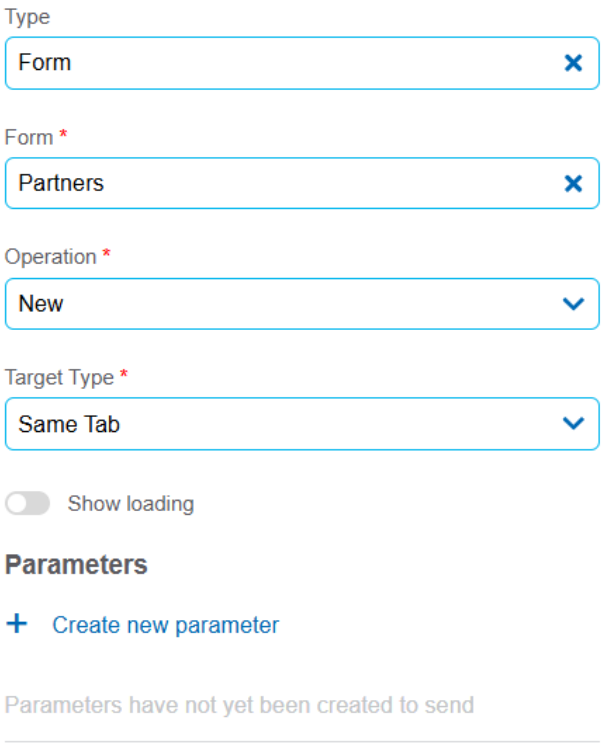
Destination Type
The available options for opening the object are displayed.

Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  Modal Horizontal Size
  Defines its width.
  Modal Vertical Size
  Defines its height.
- Iframe: selecting this option enables an additional property.
  Destination Iframe
  Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Forms



Forms
The forms modeled in the environment are displayed.

Operation
Defines the operation made when selecting the object.

Possible Values

- Create: Indicates that the panel of the object selected in the previous property is opened for the creation of an instance.
- Update: Indicates that the panel of the object selected in the previous property is opened for the update of an instance.
- Show: Indicates that the panel of the object selected in the previous property is opened for the query of an instance.
- Grid: Indicates that the grid of the object selected in the previous property is opened.

Destination Type
The available options for opening the object are displayed.

Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  - Modal Horizontal Size
    Defines its width.
  - Modal Vertical Size
    Defines its height.
- Iframe: selecting this option enables an additional property.
  - Destination Iframe

Expands the iframes previously defined on the object.

- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

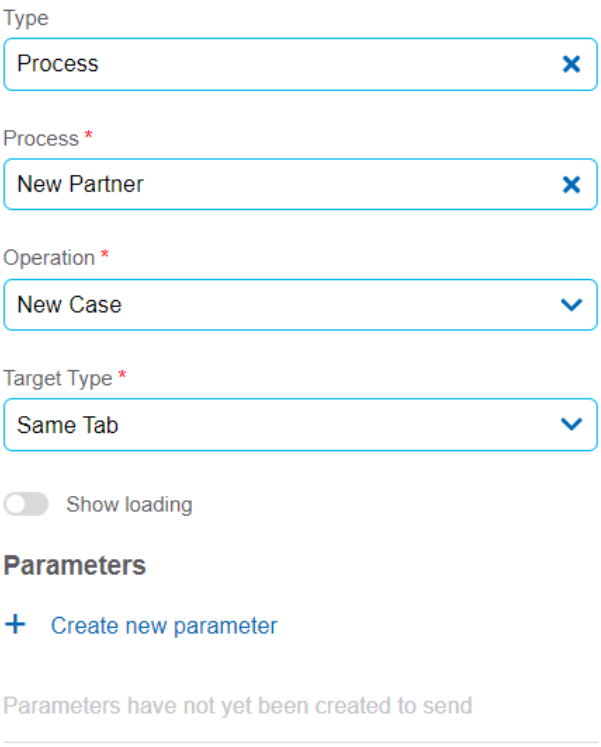
Show loading

Shows the loading icon and disables the user interaction until the redirection is over.

Parameters

Allows sending parameters to the selected object type.

Process



Process

The processes modeled in the environment are displayed.

Operation

Defines the operation made when selecting the object.

Destination Type

The available options for opening the object are displayed.

Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.

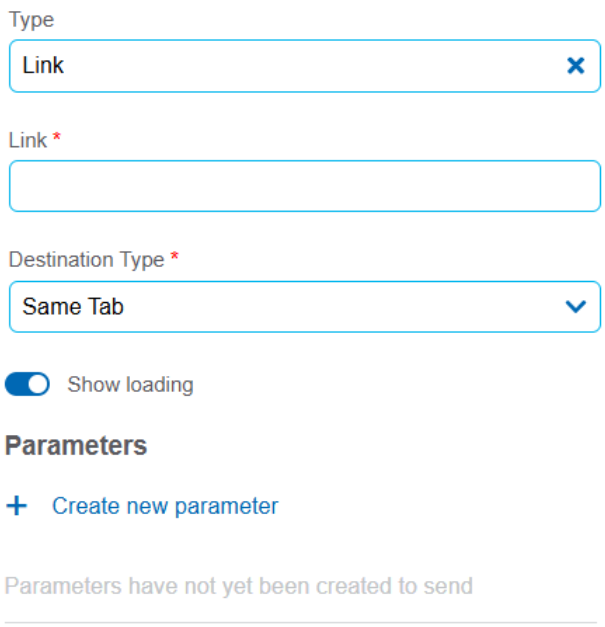
Modal Horizontal Size
Defines its width.
Modal Vertical Size
Defines its height.

- Iframe: selecting this option enables an additional property.
  Destination Iframe
  Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Parameters
Allows sending parameters to the selected object type.

Link



Link
Allows to enter any link.

Destination Type
The available options for opening the object are displayed.

Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  Modal Horizontal Size

Defines its width.

Modal Vertical Size

Defines its height.

- Iframe: selecting this option enables an additional property.

  Destination Iframe

  Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

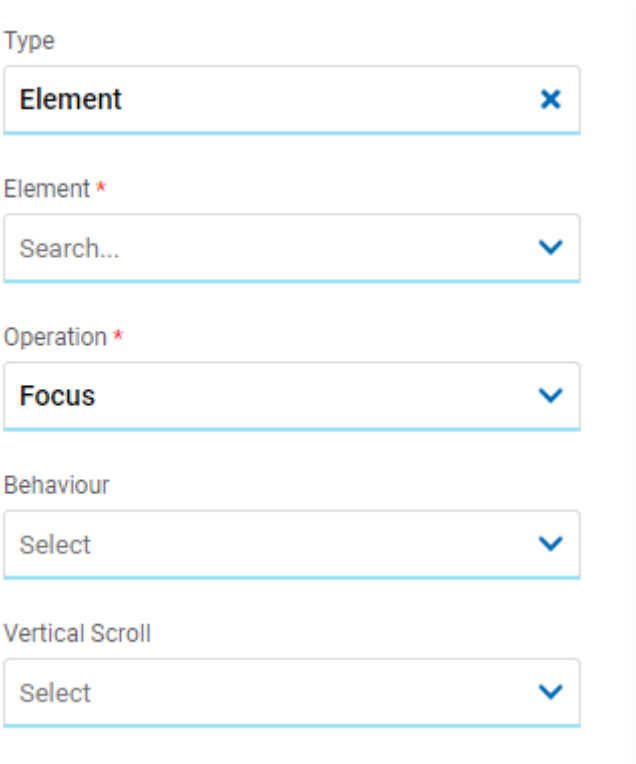
Show loading

Shows the loading icon and disables the user interaction until the redirection is over.

Parameters

Allows sending parameters to the selected object type.

Element



Element

The modeled elements in the object are displayed.

Operation

Defines the operation made when selecting the element.

Possible Values

- Focus
- Show
- Hide

- Show/Hide

Behaviour
Establishes the transition of the animation. It is only modelable if the "Focus" operation was selected.

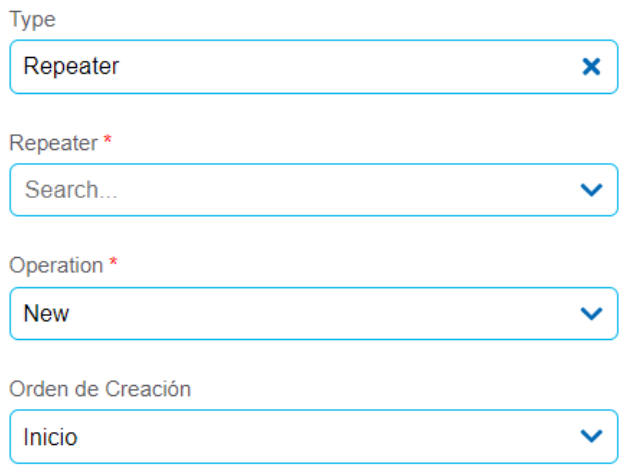
Possible Values

- Auto
- Smooth

Vertical Scroll
It is only modelable if the "Focus" operation was selected.
Possible Values

- Start: Moves to the start of the selected element.
- Closest: Moves to the closest position to the selected element from the element the event fires.
- Center: Moves to the center of the selected element.
- End: Moves to the end of the selected element.

Repeater



Repeater
The modeled repeaters from the object are displayed.

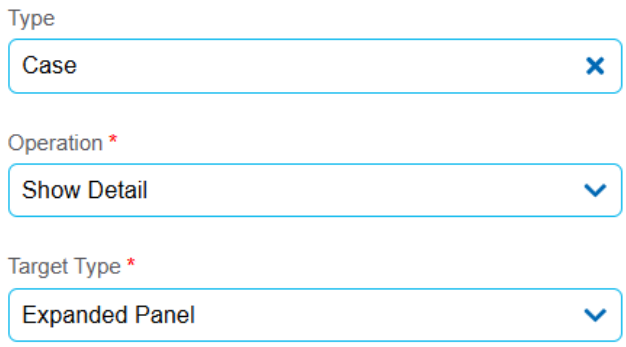
Operation
Defines the operation made when selecting the element.

Possible Values

- New: Allows for the creation of an item within the repeater. If this option is selected the additional properties are enabled.
  ### Creation Order
  It indicates in which position the new item of the container is added.
  Possible Values
  - Home
  - End

- Delete: Allows for the deletion of an item within the repeater.

Case



Case
Show case information.

Operation
Defines the operation made when selecting the element.

Possible Values

- Show Detail
- Show Graphic Execution

Destination Type
The available options for opening the object are displayed.

Possible Values

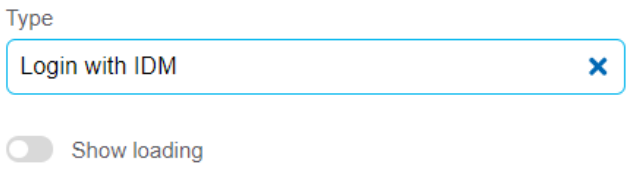
- Expanded Panel
- Modal: if this option is selected the additional properties are enabled.
  Modal Horizontal Size
  Define its width.
  Modal Vertical Size
  Define its height.

Back



It allows associating the event to go back in the browser to the element.

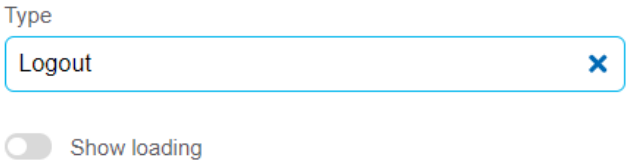
LogIn with IDM



Allows login with IDM.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

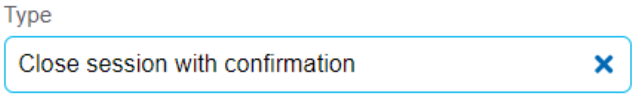
Logout



Allows logging the user out.

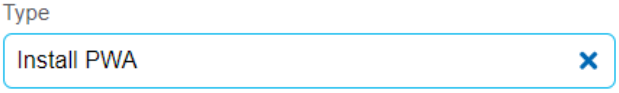
Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Close Session with Confirmation



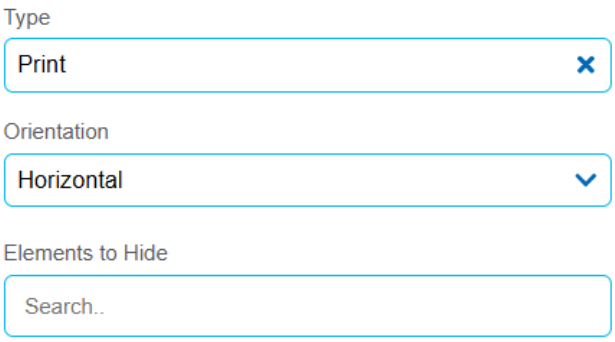
Displays a confirmation modal to logout the user.

Install PWA



Allows to install the application in the browser.

Print



This type of hyperlink associated with an element allows printing the content of the entity or page that is being executed.

Orientation

Determines the layout of the content to be printed.

Possible Values

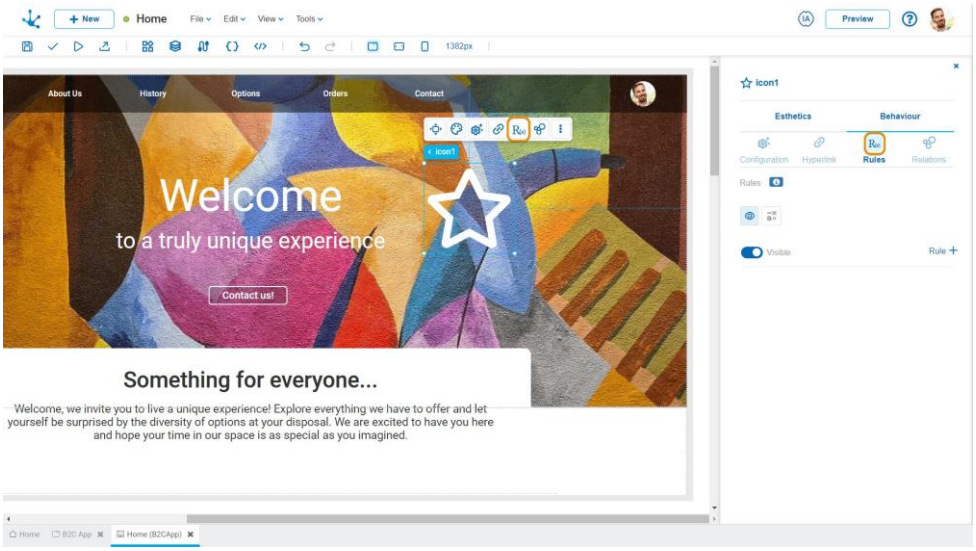
- Horizontal
- Vertical

Elements to hide

It allows selecting which elements of the entity or page should not be included in the printout.

Rule Properties

The rule properties panel of an element opens when clicking the icon  of its context menu.



Properties

Rules

Embedded rules on behavior, validation and calculation can be defined, associated with an element, by using the wizard (ctrl + space).
Shows syntax examples for writing the rules.

Visible
Indicates whether the element is visible. If this property is not checked, the element is not displayed in the page.
Visible (default) Not visible
Rule + Opens an edit area where a rule to determine the visibility condition can be defined. If a rule is defined, the icon is displayed with light blue borders.

Calculation
Rule + Opens an edit area where you can define the expression to be executed to calculate the element value. If a rule is defined, the icon is displayed with light blue borders.

Operations when defining the rule:

Saves the new or modified rule
Cancels the operation

Operations once the rule is defined:

Edits the existing rule
Deletes the rule

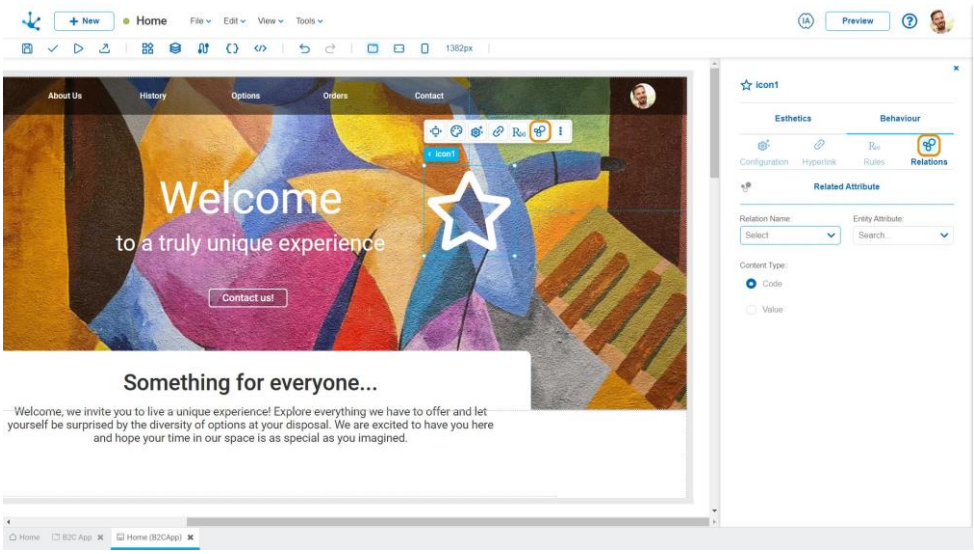
Events

Icons allow the use of an event.

Event	Description
onClick()	It is executed when clicking on the element.

Relation Properties

The relation properties panel of an element opens when clicking the icon of its context menu.



Related Attribute

A related attribute indicates that the field value is retrieved from the attribute value of another entity. One of the relations defined on the page must be selected, and then it should be specified which attribute of the related entity it links to.

A field can be modeled as a related attribute and it can also have a relation defined to an entity, a value list, or a rule. This allows the value of the related attribute to be retrieved on this field and the functionality provided by the relation to be used.

Properties

Relation Name
The list of relations defined in the entity is displayed, allowing the selection of one of them.

Entity Attribute
The list of attributes of the selected entity compatible with the field type is displayed. Attributes can have a relation modeled with an entity, a value list, or a rule in the selected entity.

Content Type
The element content varies depending on whether the Code or Value property is checked when:

- The entity attribute has a modeled relation.
- The field has the option "Entity" selected in the Values Obtained from property.

Code
When this property is checked, if the relation is modeled with a:

- Related entity, the identifier of the entity is displayed.
- Value list, the value list code is displayed.
- Rule, the code returned by the rule's execution is displayed.

Value
When this property is checked, if the relation is modeled with a:

- Related entity, the short description of the entity is displayed.
- Value list, the value list description is displayed.
- Rule, the description returned by the rule's execution is displayed.

Any reusable page can be included on a page, defining the characteristics by modeling its properties.

The element properties are represented by icons on its context menu, where its operations are also available.

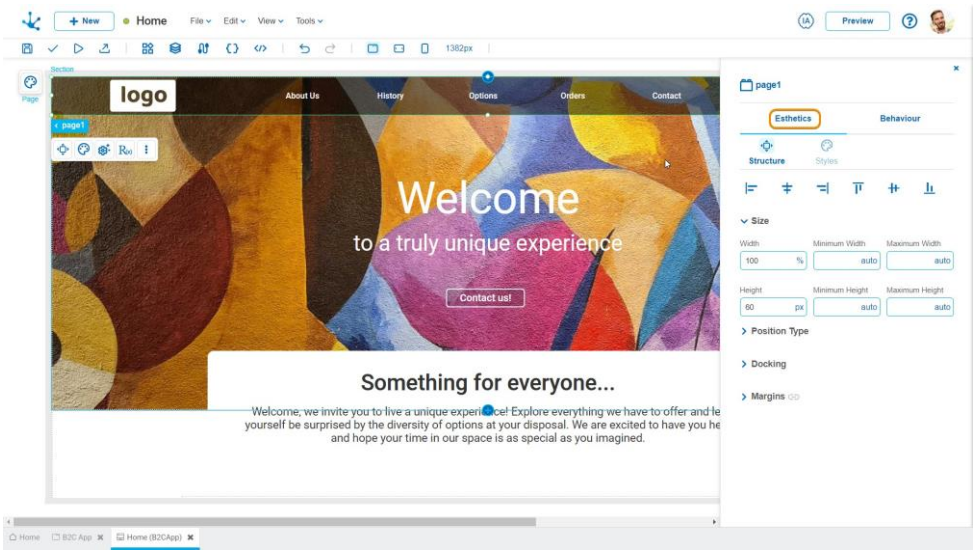
Classification of Properties

Properties are divided into two groups: esthetic properties and behavior properties.

Esthetic Properties

In the esthetic properties panel, the following are grouped:

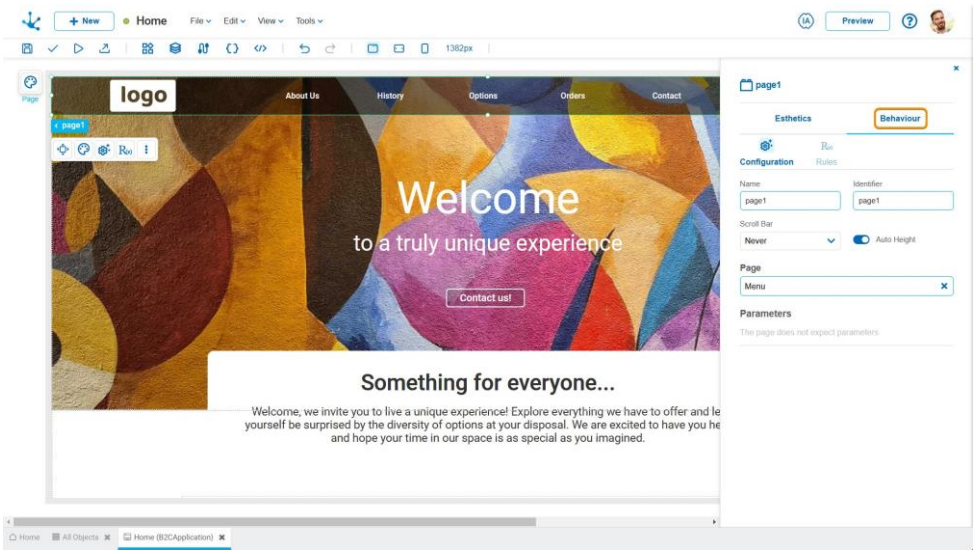
- Structure Properties
- Style Properties



Behavior Properties

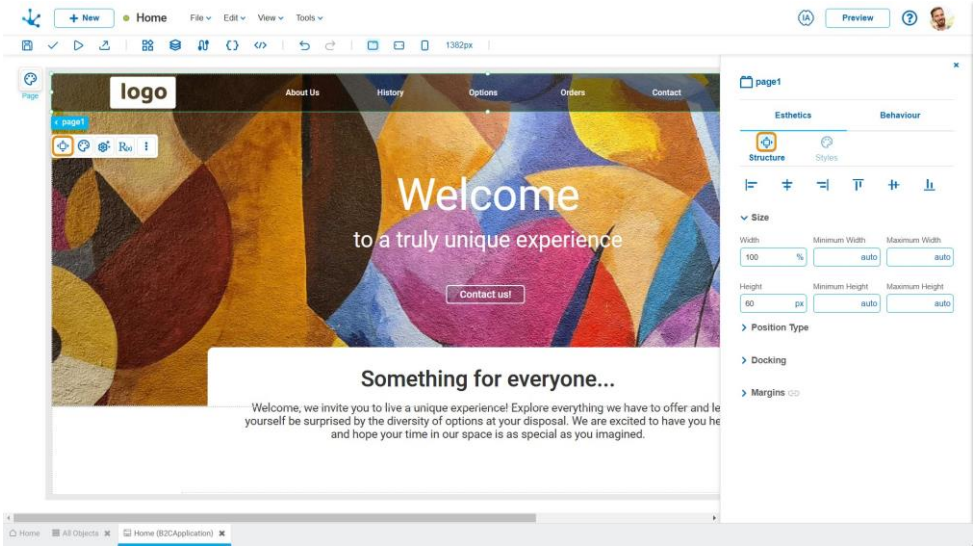
In the behavior properties panel, the following are grouped:

- Configuration Properties
- Rules Properties



Structure Properties

The structure properties panel of an element opens when selecting the icon  of its context menu.



Alignment

Allows to align the element by selecting one of the available icons.

-  Align to the left.
-  Align to horizontal center.
-  Align to the right.

- Align up.
- Align vertical center.
- Align bottom.

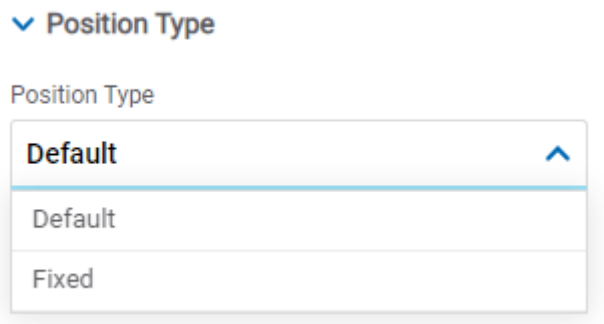
Size



It allows the input of Width and Height. The value entered in one of the properties is automatically copied to the other. These properties are expressed in pixels.

Position Type

It determines if at the time of execution the element remains fixed on the page or if it moves as the scroll bar moves forward or backward.



Possible Values

- Default: The element has a relative position with respect to the superior element where it was placed (container or section).
- Fixed: Fixes the element to the page so that it is always visible in the same position. It allows to increase the visibility of important content.

Docking



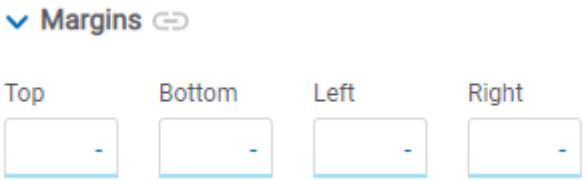
Determines the vertical and horizontal position of elements within the page section or container, when the screen is resized.

When an element is added or moved within the section or container, the modeler automatically docks it to the nearest corners or edges. It can also be configured by clicking on the arrows of the side to be docked to or on the center.

Once the element is docked, its exact position is indicated in the Margins property. The direction of the docking arrows determine the values that are automatically displayed for margins.

The docking position is indicated in the modeling area by dotted lines on the element.

Margins



It allows to define the distance of elements from the borders of their top element. The behavior of margins depends on the docking of the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Top
Distance to the top border of the highest ranking element.

Bottom
Distance to the bottom border of the highest ranking element.

Left
Distance to the left border of the highest ranking element.

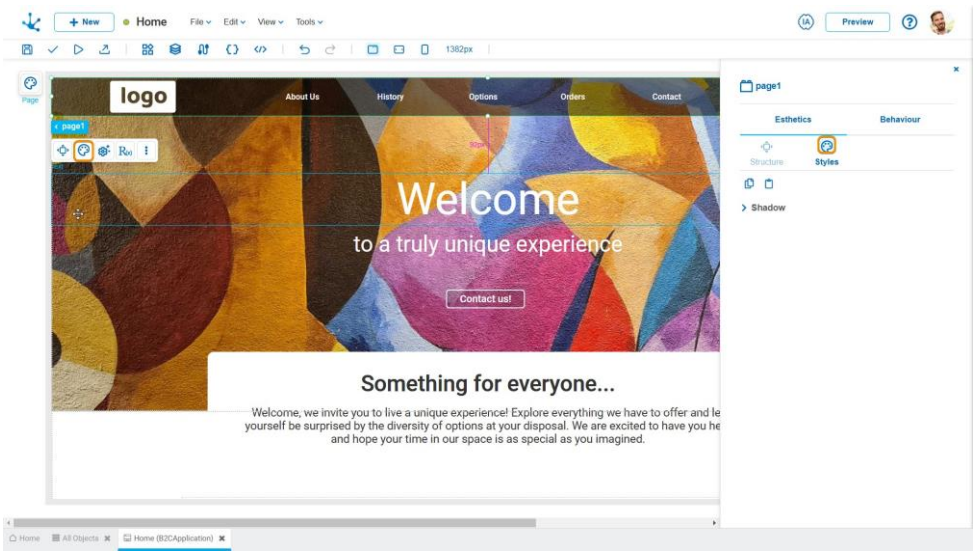
Right
Distance to the right border of the highest ranking element.

Allows the value entered in one of the margins to be copied to the other ones automatically.

Allows to indicate different values for each margin.

Style Properties

The style properties panel of an element opens when selecting the icon of its context menu.

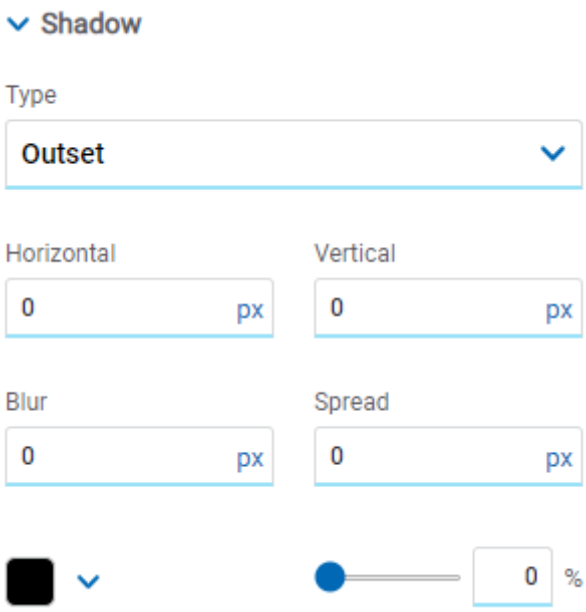


Styles

It allows copying the entire current configuration of the style panel for the selected element (Ctrl+Alt+C).

It allows applying the previously copied configuration to another element of the same type (Ctrl+Alt+V).

Shadow



Allows to define a shadow effect around the element.

Type

Possible Values

- Outset
- Inset

Horizontal
Horizontal size of the shadow to the right of the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Vertical
Vertical size of the shadow below the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

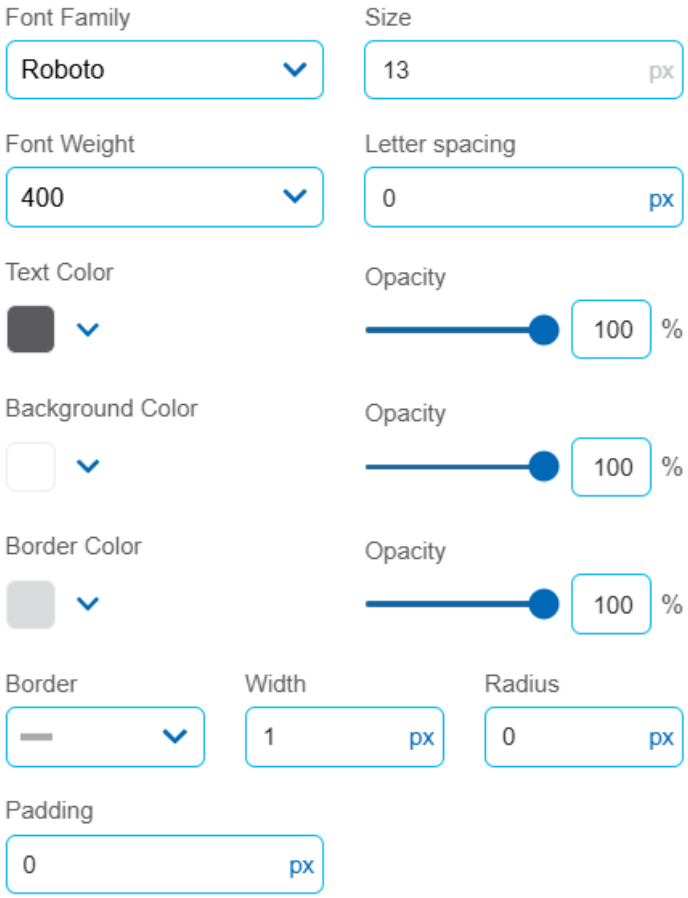
Blur
The greater this value is, the greater and lighter the shadow becomes. If not specified, its value is 0 and the shadow border is darker. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Spread
Positive values will cause the shadow to expand and grow bigger, while negative values will cause the shadow to shrink. If not specified, its value is 0 and the shadow will have the same size as the element.

Color
Defines the color of the shadow.

Transparency
Defines the transparency of the shadow.

Tooltip



Source

It allows customizing the font, size, thickness and letter spacing of the pop-up message.

Colors

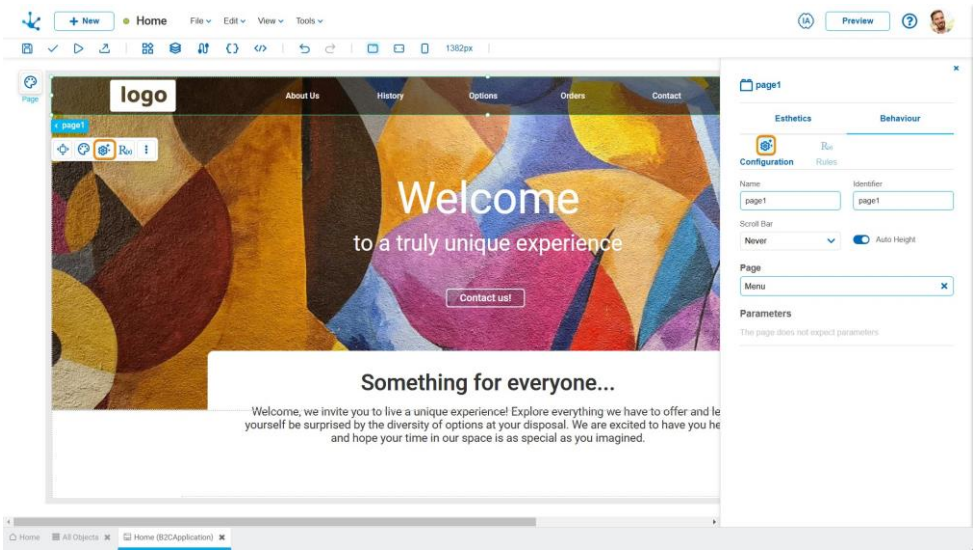
The color and opacity of the text, background and border can be defined. Borders are set uniformly for all sides of the tooltip.

Border

The border type, width, and border radius can be configured.

Configuration Properties

The configuration properties panel of an element opens when clicking the icon  of its context menu.



Name
Name used to reference the element during modeling.

Identifier
Uniquely identifies the element. It is used in the Javascript SDK.

Tooltip
Pop-up message that appears when hovering over an element, providing additional information.



Text
Content of the message being displayed.

Direction
Position of the message in relation to the element (top, bottom, left, or right).

Scroll Bar
It allows defining how the scroll bar behaves inside the reusable page container.

Possible Values

- Automatic: The scroll bar appears only when the contents of the container exceed its visible size.
- Never: The scroll bar does not appear, regardless of the content size.

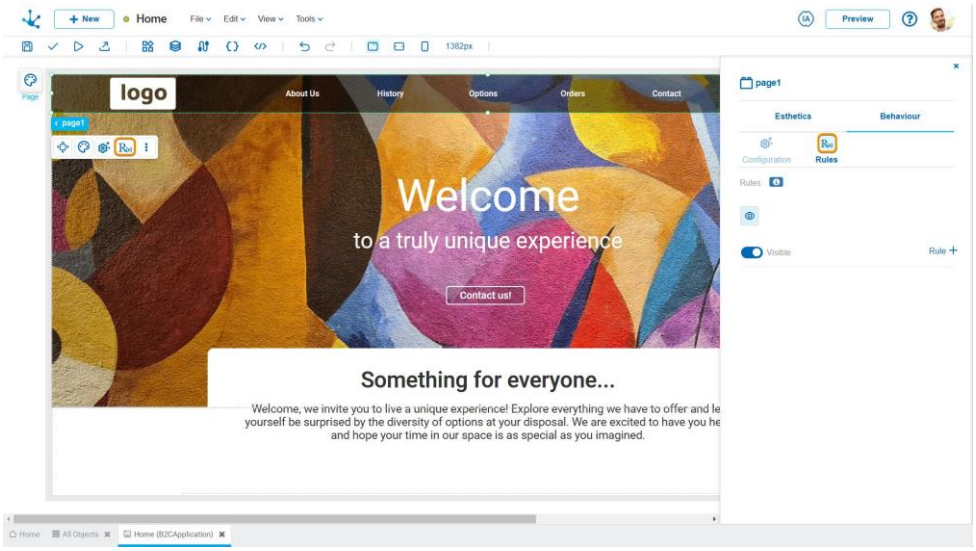
Auto Height
Determines whether the reusable page container automatically adjusts its height to fit the size of the content it contains.
If this property is checked, the container dynamically grows or shrinks in height based on the loaded content, eliminating the need for scrolling. In this case, the value of the Scroll Bar property automatically adjusts to "Never", since the container does not require sliding.

Page
The pages modeled in the environment are displayed.

Parameters
It allows sending parameters to the selected page.

Rule Properties

The rule properties panel of an element opens when clicking the icon  of its context menu.



Properties

Rules
Embedded rules on behavior, validation and calculation can be defined, associated with an element, by using the wizard (ctrl + space).
Shows syntax examples for writing the rules.

Visible
Indicates whether the element is visible. If this property is not checked, the element is not displayed in the page.
Visible (default) Not visible
Rule + Opens an edit area where a rule to determine the visibility condition can be defined. If a rule is defined, the icon is displayed with light blue borders.

Operations when defining the rule:

Saves the new or modified rule

Cancels the operation

Operations once the rule is defined:

Edits the existing rule

Deletes the rule

Widget

Indicators can be included on a page, defining the characteristics by modeling its properties.

The element properties are represented by icons on its context menu, where its operations are also available.

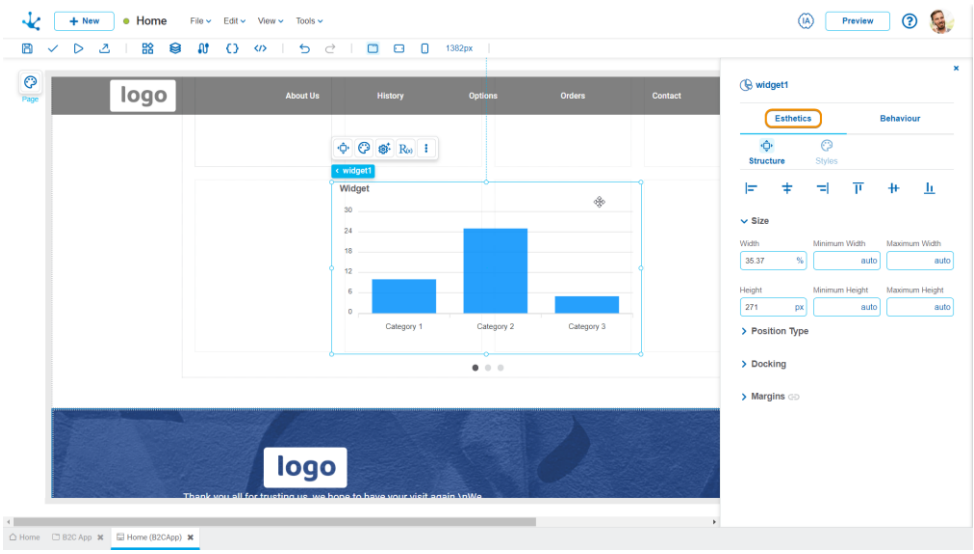
Classification of Properties

Properties are divided into two groups: esthetic properties and behavior properties.

Esthetic Properties

In the esthetic properties panel, the following are grouped:

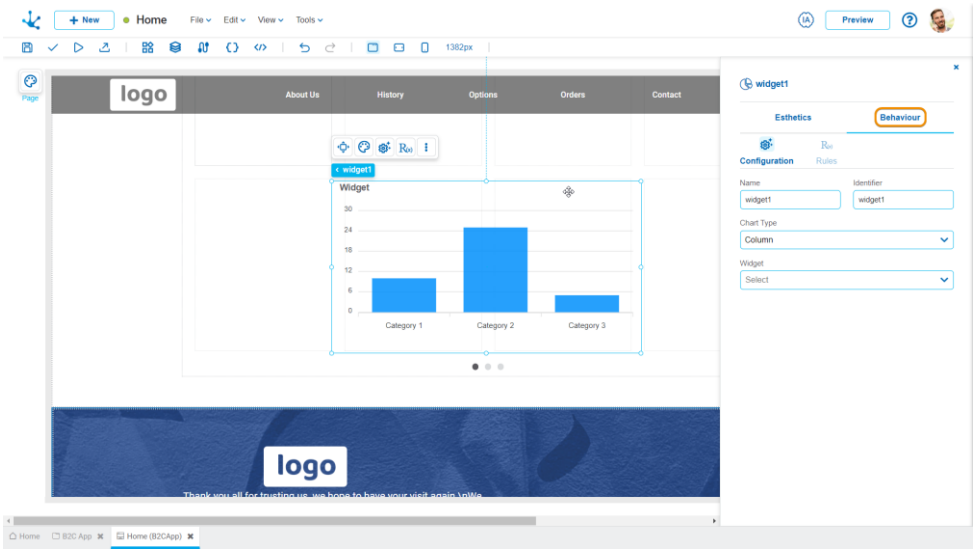
- Structure Properties
- Style Properties



Behavior Properties

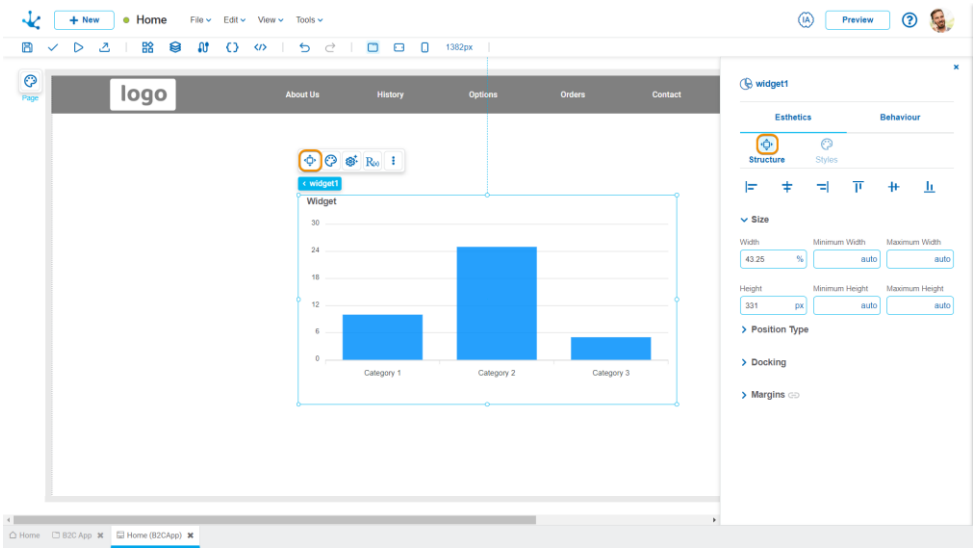
In the behavior properties panel, the following are grouped:

- Configuration Properties
- Rules Properties



Structure Properties

The structure properties panel of an element opens when selecting the icon  of its context menu.



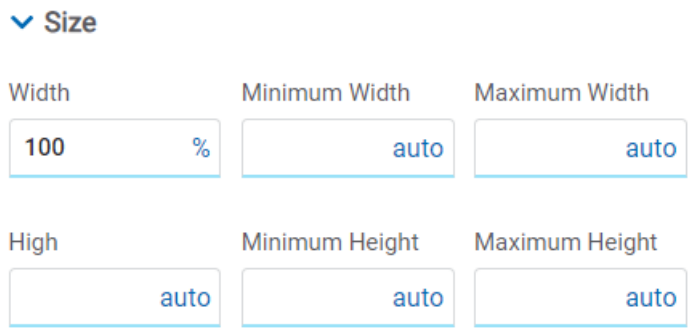
Alignment

Allows to align the element by selecting one of the available icons.

-  Align to the left.
-  Align to horizontal center.
-  Align to the right.

- Align up.
- Align vertical center.
- Align bottom.

Size

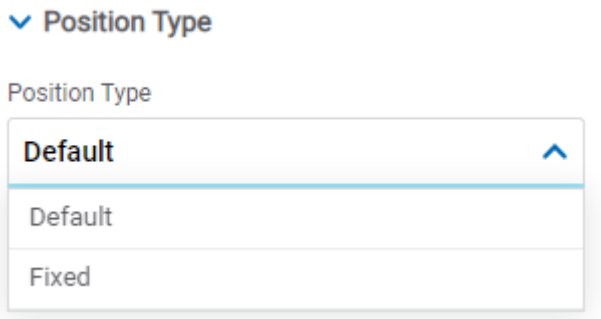


All size properties can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh). Additionally, for Width and Height properties, the "auto" option is added.

> *If selecting percentage (%) for the width and height properties, the size is calculated relative to the top element.*

Position Type

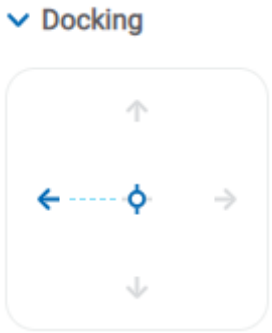
It determines if at the time of execution the element remains fixed on the page or if it moves as the scroll bar moves forward or backward.



Possible Values

- Default: The element has a relative position with respect to the superior element where it was placed (container or section).
- Fixed: Fixes the element to the page so that it is always visible in the same position. It allows to increase the visibility of important content.

Docking



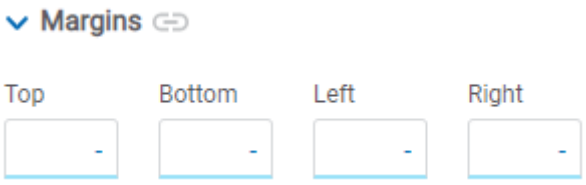
Determines the vertical and horizontal position of elements within the page section or container, when the screen is resized.

When an element is added or moved within the section or container, the modeler automatically docks it to the nearest corners or edges. It can also be configured by clicking on the arrows of the side to be docked to or on the center.

Once the element is docked, its exact position is indicated in the Margins property. The direction of the docking arrows determine the values that are automatically displayed for margins.

The docking position is indicated in the modeling area by dotted lines on the element.

Margins



It allows to define the distance of elements from the borders of their top element. The behavior of margins depends on the docking of the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Top
Distance to the top border of the highest ranking element.

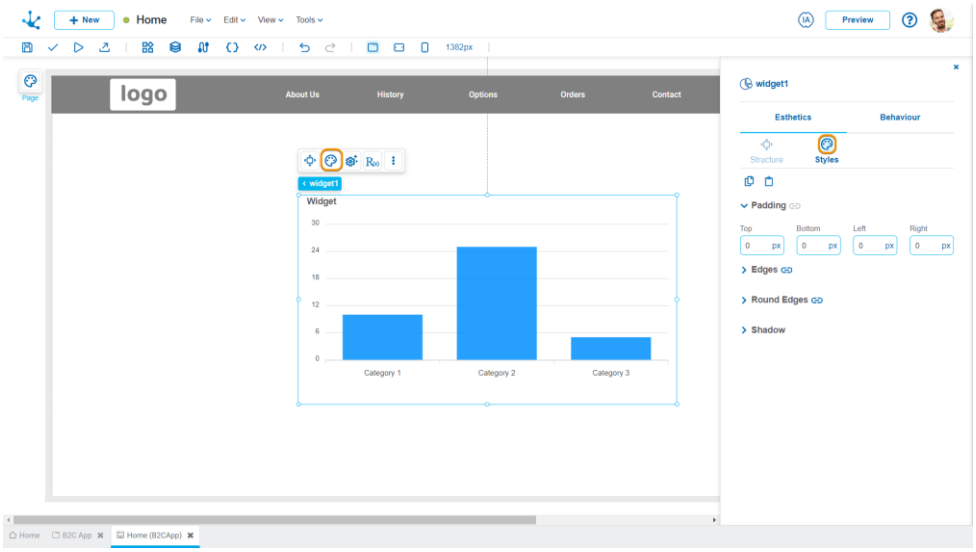
Bottom
Distance to the bottom border of the highest ranking element.

Left
Distance to the left border of the highest ranking element.

Right
Distance to the right border of the highest ranking element.

Allows the value entered in one of the margins to be copied to the other ones automatically.
Allows to indicate different values for each margin.

The style properties panel of an element opens when selecting the icon of its context menu.

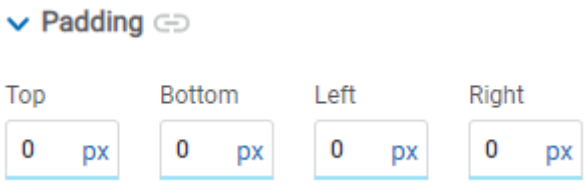


Styles

It allows copying the entire current configuration of the style panel for the selected element (Ctrl+Alt+C).

It allows applying the previously copied configuration to another element of the same type (Ctrl+Alt+V).

Padding

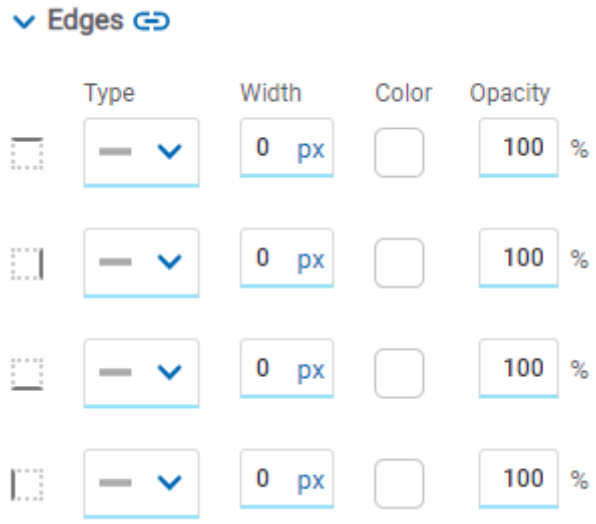


All padding space properties allow to create a space around the element borders (top, bottom, sides) and its content. The padding space configured is inward, while the margin is configured relative to a top element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Allows values entered in one of the paddings to be copied to the other ones automatically.

Allows to indicate different values for each padding.

Edges



It allows to define the style of borders. Each one has its type, width, color and opacity defined, the latter as a percentage.

Allows the value entered in one of the borders to be copied to the other ones automatically.
Allows to indicate different values for each border.

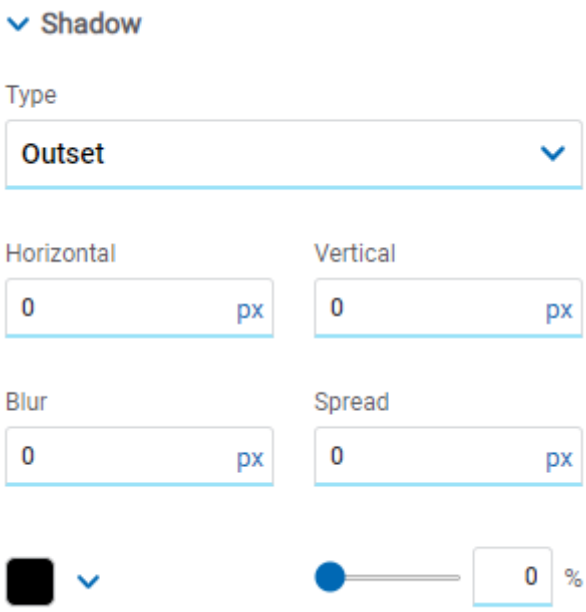
Round Edges



It allows to define the round edges at the corners of elements.
It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Allows the value entered in one of the borders to be copied to the other ones automatically.
Allows to indicate different values for each border.

Shadow



Allows to define a shadow effect around the element.

Type

Possible Values

- Outset
- Inset

Horizontal
Horizontal size of the shadow to the right of the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Vertical
Vertical size of the shadow below the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

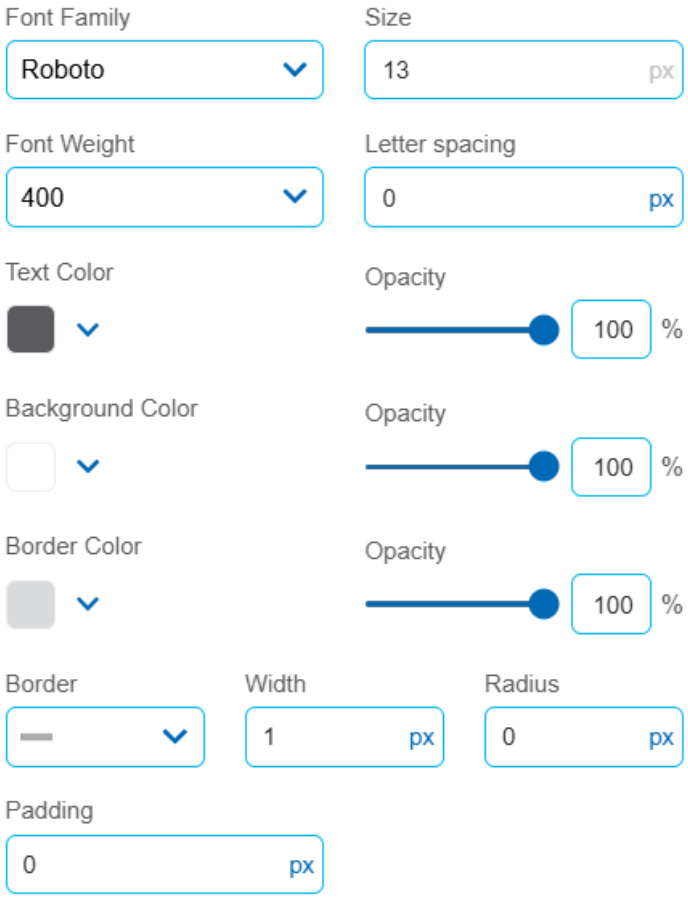
Blur
The greater this value is, the greater and lighter the shadow becomes. If not specified, its value is 0 and the shadow border is darker. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Spread
Positive values will cause the shadow to expand and grow bigger, while negative values will cause the shadow to shrink. If not specified, its value is 0 and the shadow will have the same size as the element.

Color
Defines the color of the shadow.

Transparency
Defines the transparency of the shadow.

Tooltip



Source

It allows customizing the font, size, thickness and letter spacing of the pop-up message.

Colors

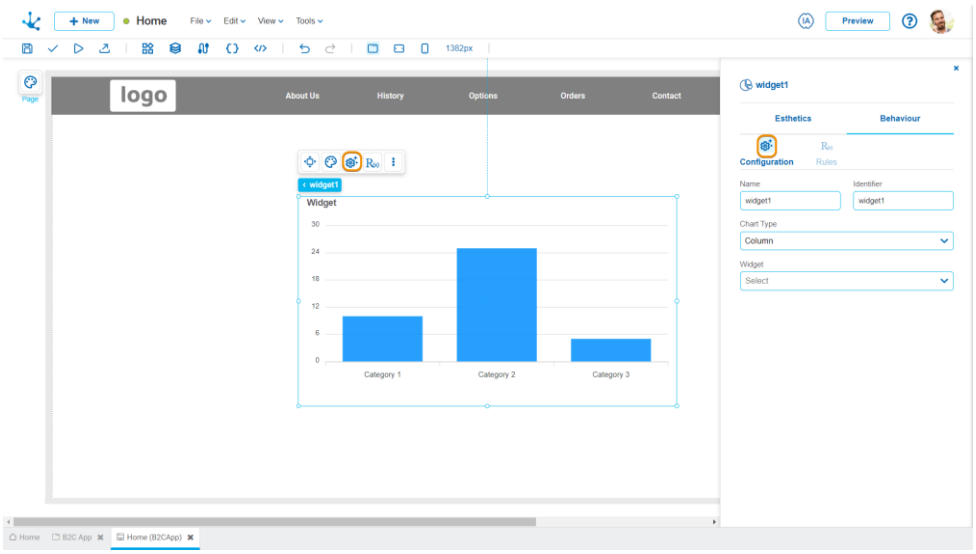
The color and opacity of the text, background and border can be defined. Borders are set uniformly for all sides of the tooltip.

Border

The border type, width, and border radius can be configured.

Configuration Properties

The configuration properties panel of an element opens when clicking the icon  of its context menu.



Name
Name used to reference the element during modeling.

Identifier
Uniquely identifies the element. It is used in the Javascript SDK.

Tooltip
Pop-up message that appears when hovering over an element, providing additional information.



Text
Content of the message being displayed.

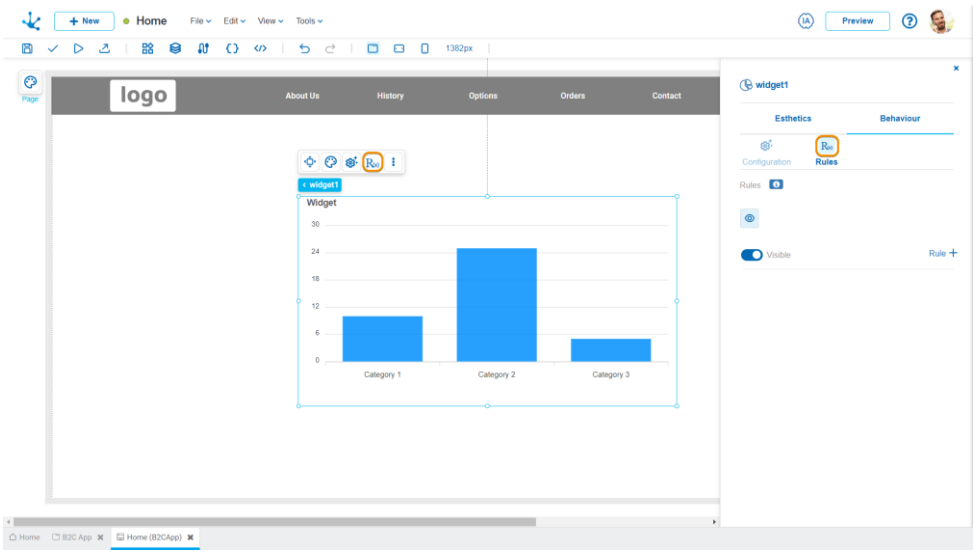
Direction
Position of the message in relation to the element (top, bottom, left, or right).

Chart Type
The types of charts for the modeled widgets are displayed in the environment.

Widget
The widgets modeled in the environment are displayed, according to the type of chart defined in the previous property.

Rule Properties

The rule properties panel of an element opens when clicking the icon $R_{(x)}$ of its context menu.



Properties

Rules

Embedded rules on behavior, validation and calculation can be defined, associated with an element, by using the wizard (ctrl + space).

Shows syntax examples for writing the rules.

Visible

Indicates whether the element is visible. If this property is not checked, the element is not displayed in the page.

Visible (default) Not visible

Rule + Opens an edit area where a rule to determine the visibility condition can be defined. If a rule is defined, the icon is displayed with light blue borders.

Operations when defining the rule:

Saves the new or modified rule

Cancels the operation

Operations once the rule is defined:

Edits the existing rule

Deletes the rule

Slider

It allows the sequential display of various elements, such as images, videos or any other content in a horizontal sequence. Users can scroll horizontally through the set of elements, either manually using

browsing controls or automatically. This improves the user experience, offering a more dynamic and content personalized interaction.

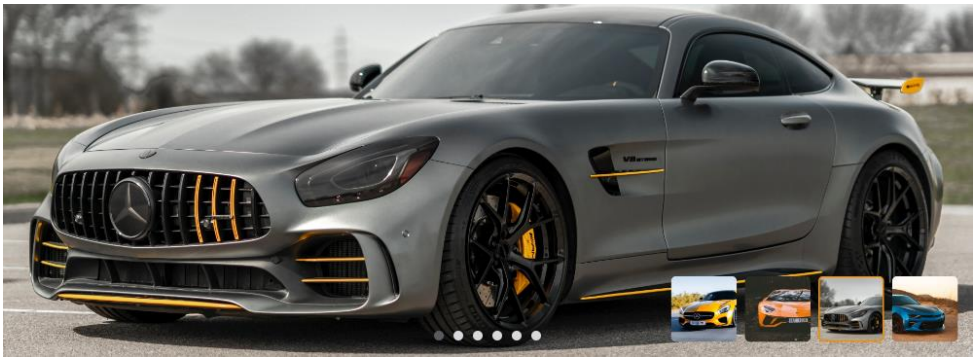
Subtypes

By selecting the “Slider” option from the icon on the top toolbar, a list with the different subtypes of this element is displayed, which can be dragged to the modeling area. Each subtype has modeled element properties, predefined in a specific way.

- Dots
- Arrows

Dots

This type of slider uses browsing dots, usually located at the bottom. Each dot represents an element of the set, allowing the user to display and directly select by pressing the corresponding dot.



Arrows

This type of slider incorporates browsing arrows located on both sides of it. The arrows allow the user to move forward or backward in the sequence, moving between items one at a time.



The element properties are represented by icons on its context menu, where its operations are also available.

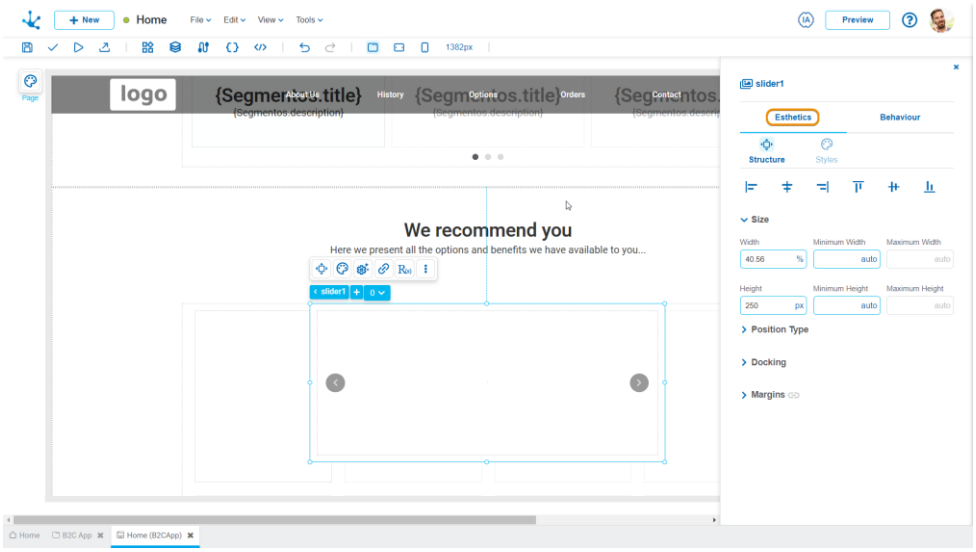
Classification of Properties

Properties are divided into two groups: esthetic properties and behavior properties.

Esthetic Properties

In the esthetic properties panel, the following are grouped:

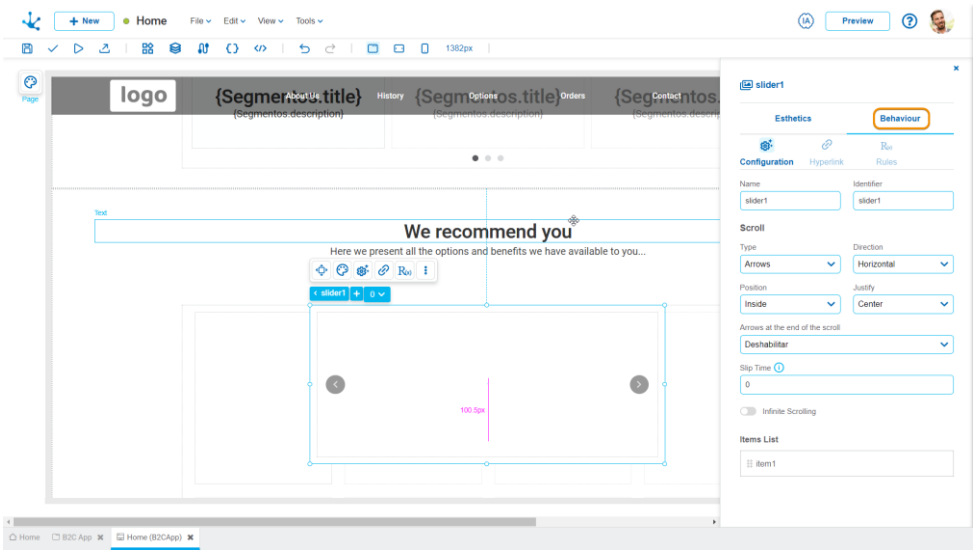
- Structure Properties
- Style Properties



Behavior Properties

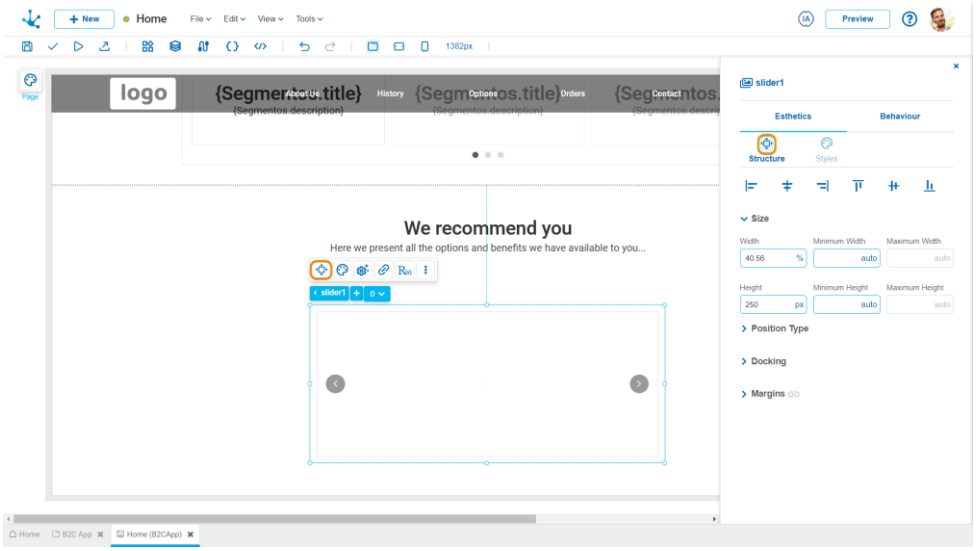
In the behavior properties panel, the following are grouped:

- Configuration Properties
- Hyperlink Properties
- Rules Properties



Structure Properties

The structure properties panel of an element opens when selecting the icon  of its context menu.



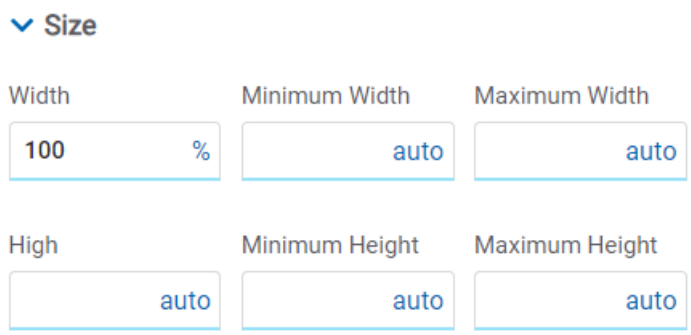
Alignment

Allows to align the element by selecting one of the available icons.

-  Align to the left.
-  Align to horizontal center.
-  Align to the right.
-  Align up.

- Align vertical center.
- Align bottom.

Size

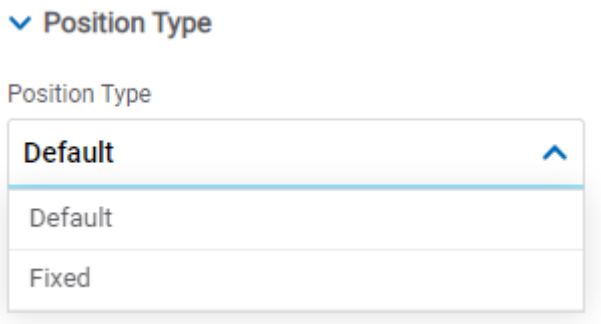


All size properties can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh). Additionally, for Width and Height properties, the "auto" option is added.

> *If selecting percentage (%) for the width and height properties, the size is calculated relative to the top element.*

Position Type

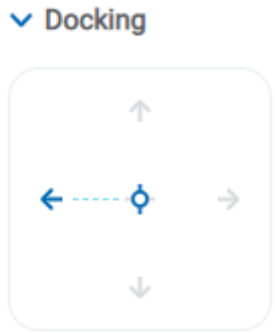
It determines if at the time of execution the element remains fixed on the page or if it moves as the scroll bar moves forward or backward.



Possible Values

- Default: The element has a relative position with respect to the superior element where it was placed (container or section).
- Fixed: Fixes the element to the page so that it is always visible in the same position. It allows to increase the visibility of important content.

Docking



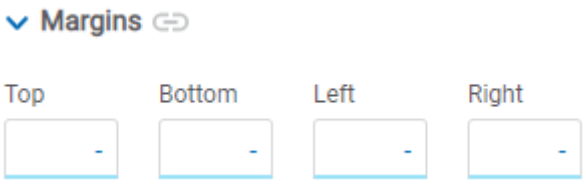
Determines the vertical and horizontal position of elements within the page section or container, when the screen is resized.

When an element is added or moved within the section or container, the modeler automatically docks it to the nearest corners or edges. It can also be configured by clicking on the arrows of the side to be docked to or on the center.

Once the element is docked, its exact position is indicated in the Margins property. The direction of the docking arrows determine the values that are automatically displayed for margins.

The docking position is indicated in the modeling area by dotted lines on the element.

Margins



It allows to define the distance of elements from the borders of their top element. The behavior of margins depends on the docking of the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Top
Distance to the top border of the highest ranking element.

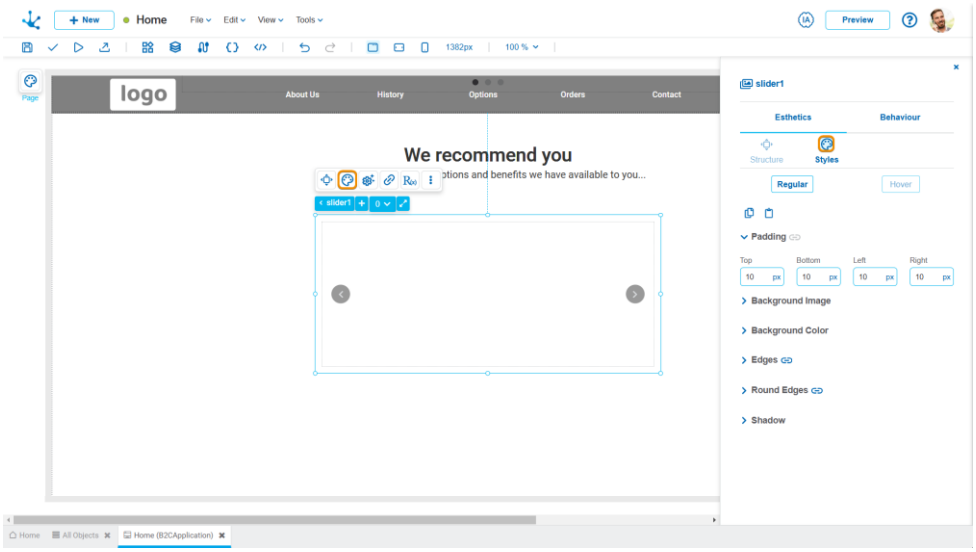
Bottom
Distance to the bottom border of the highest ranking element.

Left
Distance to the left border of the highest ranking element.

Right
Distance to the right border of the highest ranking element.

Allows the value entered in one of the margins to be copied to the other ones automatically.
Allows to indicate different values for each margin.

The style properties panel of an element opens when selecting the icon of its context menu.

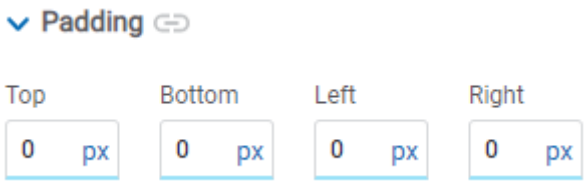


Styles

It allows copying the entire current configuration of the style panel for the selected element (Ctrl+Alt+C).

It allows applying the previously copied configuration to another element of the same type (Ctrl+Alt+V).

Padding

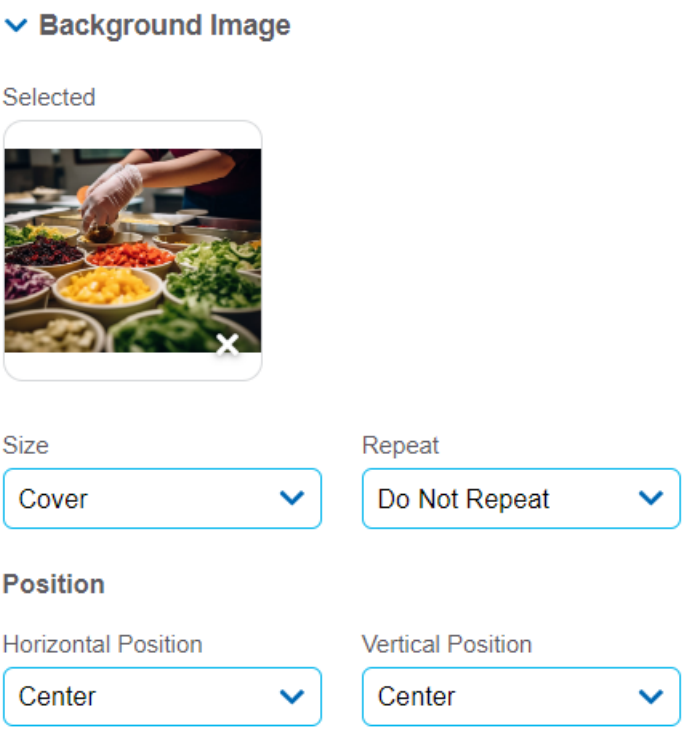


All padding space properties allow to create a space around the element borders (top, bottom, sides) and its content. The padding space configured is inward, while the margin is configured relative to a top element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Allows values entered in one of the paddings to be copied to the other ones automatically.

Allows to indicate different values for each padding.

Background Image



Allows to add a background image to the element.

Selected

An image can be uploaded from the computer where it is being modeled.

Size

Possible Values

- Cover
- Content
- Auto

Repeat

Possible Values

- Repeat
- Repeat Hor.
- Repeat Vert.
- Spacing
- Rounded
- Do Not Repeat

Horizontal Position

Possible Values

- Center

- Left
- Right

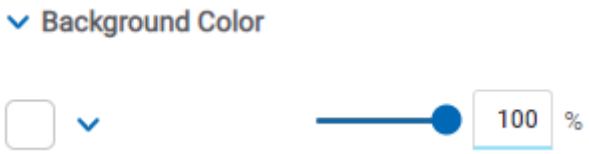
Vertical Position

- Center
- Top
- Botton

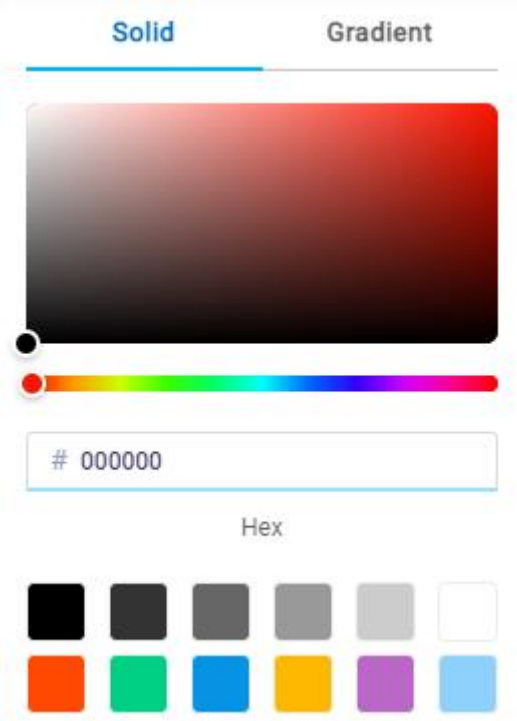
Background Color

It allows to model a background color for the element, it can be solid or gradient and different properties are defined for each one.

Solid



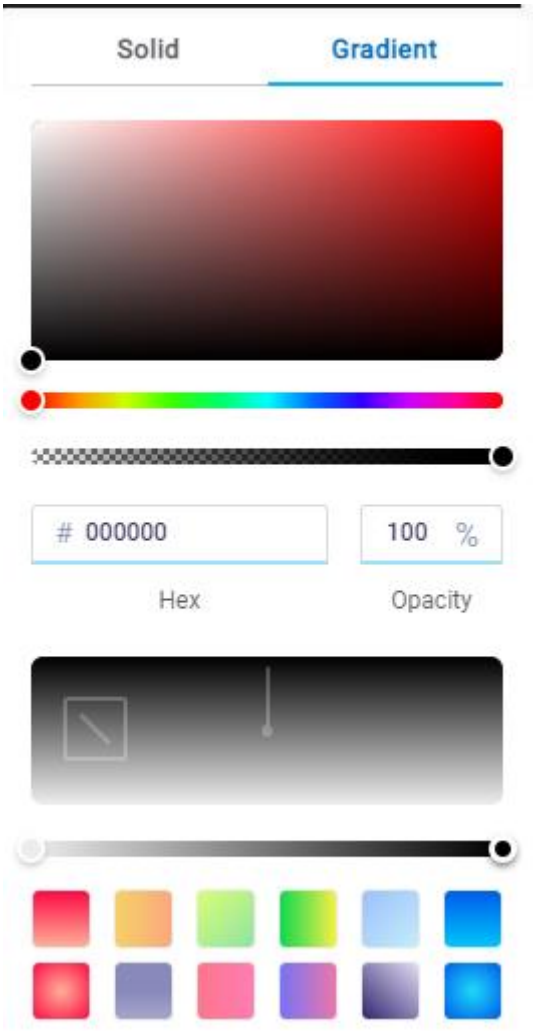
This option enables not only to select the color from a palette but also to define the degree of transparency.



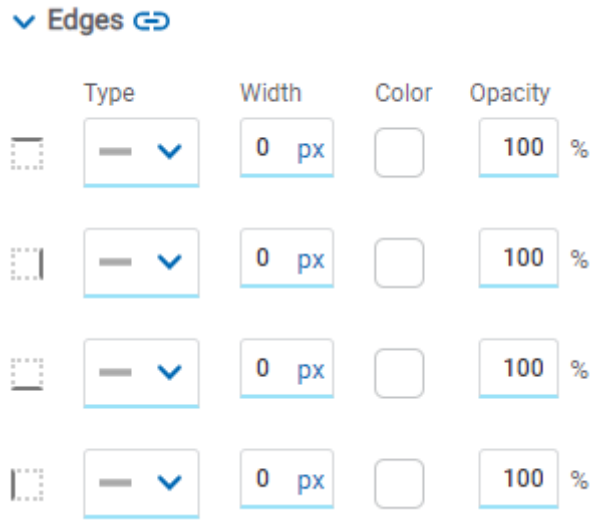
Gradient



This option enables to select not only the color from a palette but also to define its opacity and intensity according to the selected angle.



Edges



It allows to define the style of borders. Each one has its type, width, color and opacity defined, the latter as a percentage.

Allows the value entered in one of the borders to be copied to the other ones automatically.
Allows to indicate different values for each border.

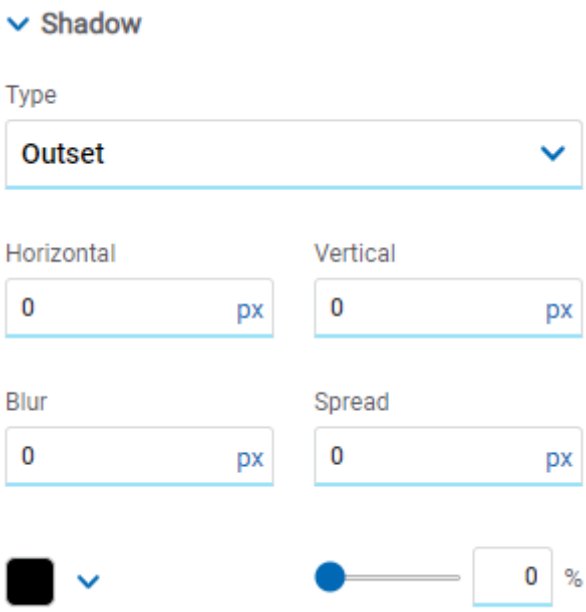
Round Edges



It allows to define the round edges at the corners of elements.
It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Allows the value entered in one of the borders to be copied to the other ones automatically.
Allows to indicate different values for each border.

Shadow



Allows to define a shadow effect around the element.

Horizontal
Horizontal size of the shadow to the right of the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Vertical
Vertical size of the shadow below the element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

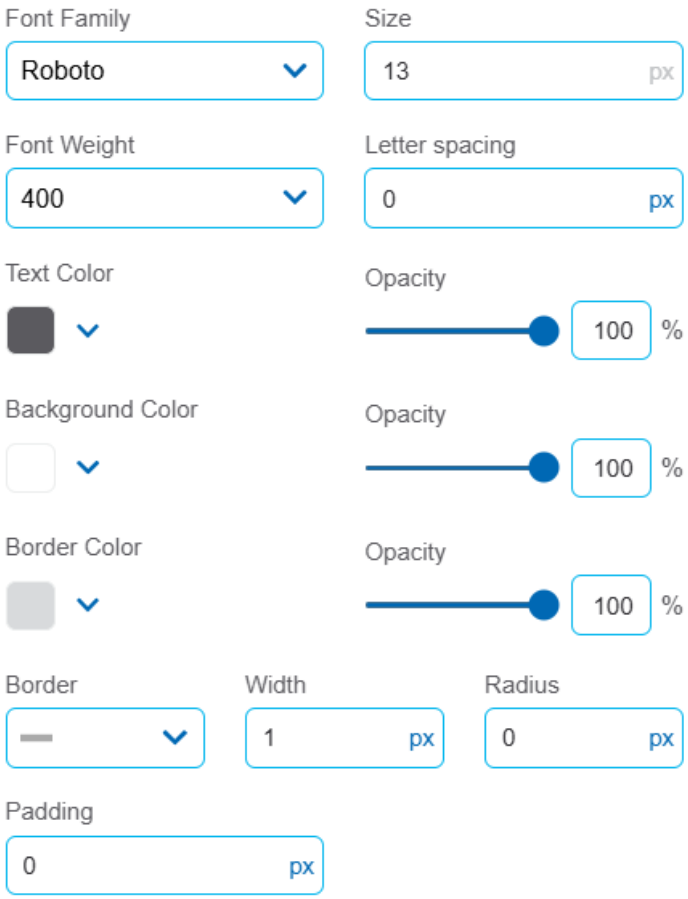
Blur
The greater this value is, the greater and lighter the shadow becomes. If not specified, its value is 0 and the shadow border is darker. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Spread
Positive values will cause the shadow to expand and grow bigger, while negative values will cause the shadow to shrink. If not specified, its value is 0 and the shadow will have the same size as the element.

Color
Defines the color of the shadow.

Transparency
Defines the transparency of the shadow.

Tooltip



Source

It allows customizing the font, size, thickness and letter spacing of the pop-up message.

Colors

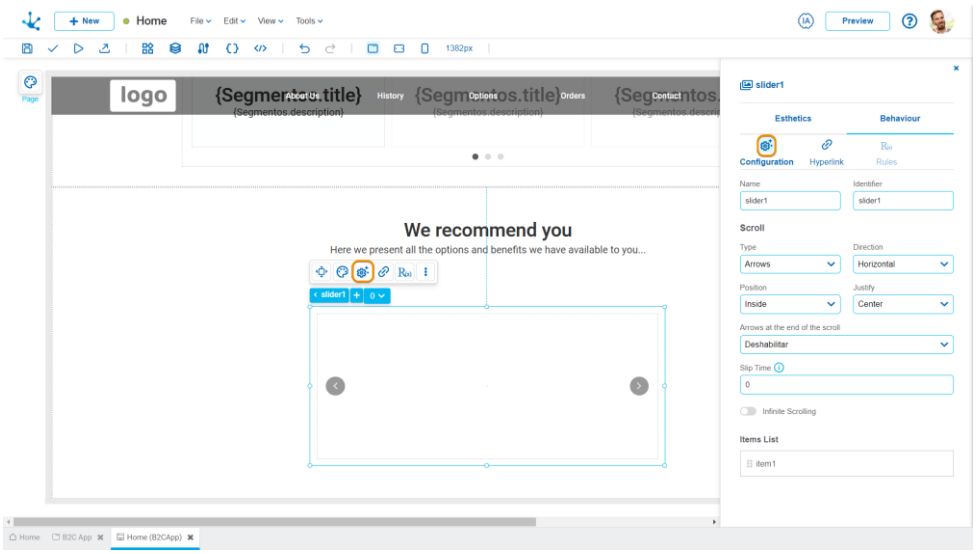
The color and opacity of the text, background and border can be defined. Borders are set uniformly for all sides of the tooltip.

Border

The border type, width, and border radius can be configured.

Configuration Properties

The configuration properties panel of an element opens when clicking the icon  of its context menu.



Name
Name used to reference the element during modeling.

Identifier
Uniquely identifies the element. It is used in the Javascript SDK.

Tooltip
Pop-up message that appears when hovering over an element, providing additional information.



Text
Content of the message being displayed.

Direction
Position of the message in relation to the element (top, bottom, left, or right).

Scroll

Type
It allows selection between types of sliding, either by dots or by arrows.

Direction
It allows for the sliding direction, horizontal or vertical, to be defined, depending on the needs of the content.

Position
It allows defining the position of the slider controls, either inside or outside the slider, placing them at the top or bottom.

Justify

The slider controls can be aligned to the left, center, or right of it, according to the desired layout.

Arrows at the End of the Scroll

It allows modeling the sliding controls when reaching the end of the sequence. This option is only applicable to arrow-type sliders.

Possible Values

- Disable
- Hide

Sliding Time

It allows adjusting the speed of the automatic sliding, expressed in seconds, to enhance the user experience and adapt to the type of content.

Infinite Sliding

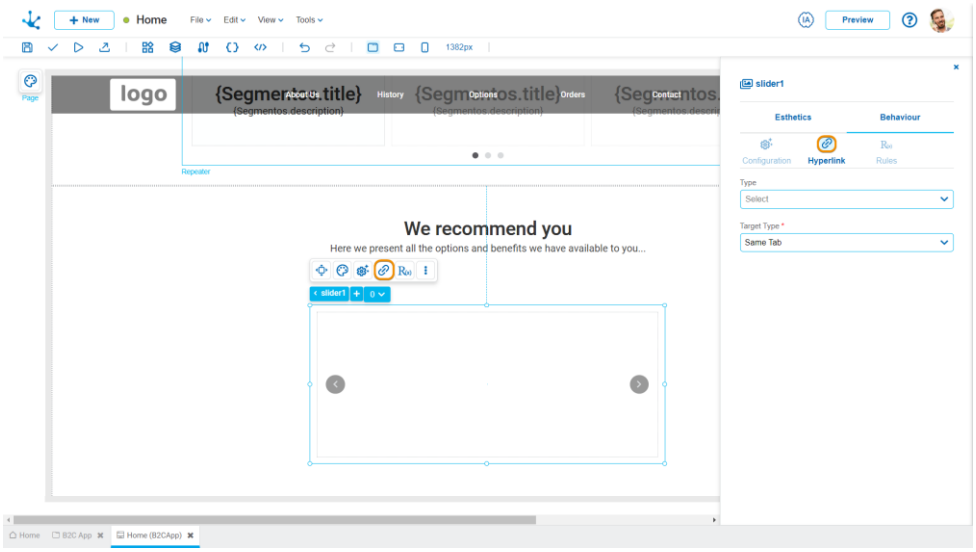
It allows enabling or disabling the option to restart the sequence once the end is reached.

Items List

It allows changing the order of items in the sequence.

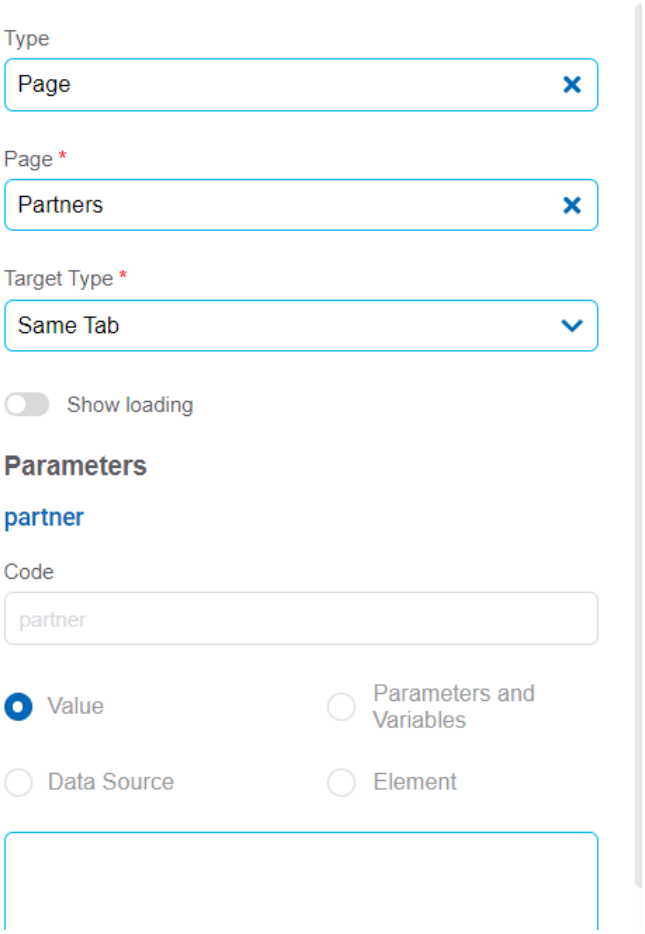
Hyperlink Properties

The hyperlink properties panel of an element opens when clicking the icon  of its context menu.



Define the behavior of the element when selecting it. Different properties are enabled depending on the type of object selected.

Page



Page

The pages modeled in the environment are displayed.

Destination Type

The available options for opening the object are displayed.

Possible Values

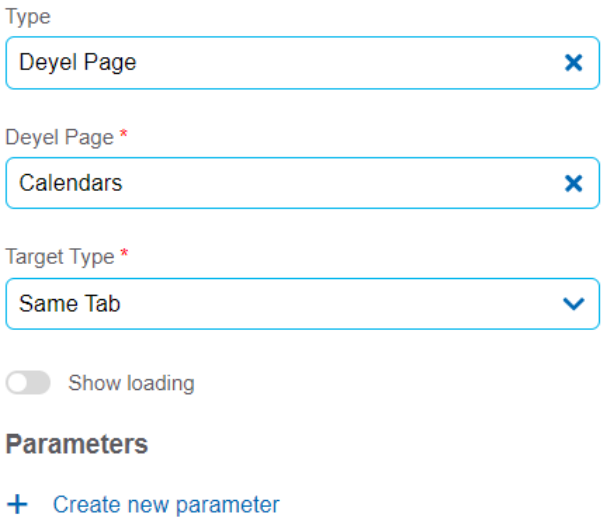
- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  - Modal Horizontal Size
    Defines its width.
  - Modal Vertical Size
    Defines its height.
- Iframe: selecting this option enables an additional property.
  - Destination Iframe
    Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading

Shows the loading icon and disables the user interaction until the redirection is over.

Parameters
If there are parameters, their value can be specified with text, parameters and variables or data source.

Deyel Page



Deyel Page
The pages belonging to **Deyel** are displayed.

Destination Type
The available options for opening the object are displayed.

Possible Values

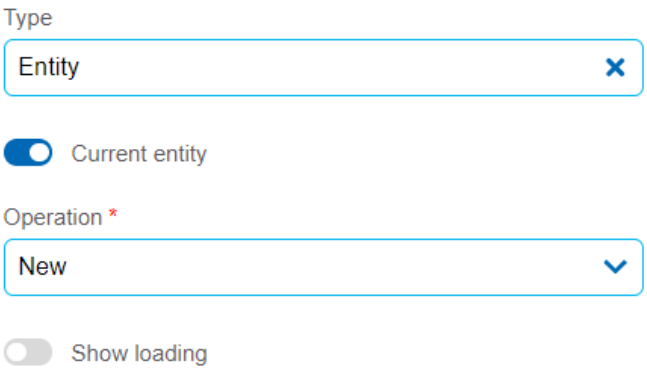
- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  - Modal Horizontal Size
    Defines its width.
  - Modal Vertical Size
    Defines its height.
- Iframe: selecting this option enables an additional property.
  - Destination Iframe
    Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Parameters

Allows sending parameters to the selected object type.

Entity



Current Entity

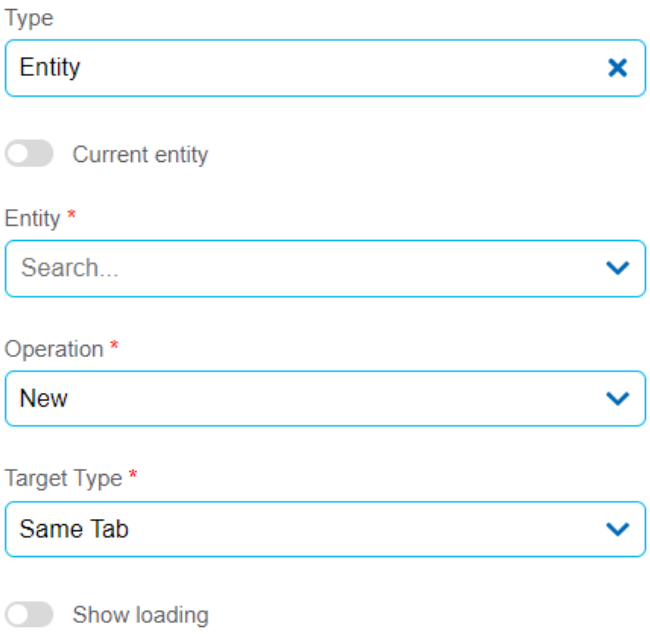
If this property is checked, the operation applies directly to the entity being modeled.

Operation

Defines the operation made when selecting the object.

Possible Values

- New: Indicates that the current entity panel is opened for the creation of an instance.
- Modify: Indicates that the current entity panel is opened for the update of an instance.
- Delete: Indicates that the current entity panel is opened for the deletion of an instance.



Entity
If the property Current Entity, is not checked, the entities modeled in the environment are displayed.

Operation
Defines the operation made when selecting the object.

Possible Values

- New: The panel of the entity selected in the previous property is opened to create an instance.
- Update The panel of the entity selected in the previous property is opened to update an instance.
- Show: The panel of the entity selected in the previous property is opened to show an instance.
- Grid: Indicates that the grid of the entity selected in the previous property is opened.

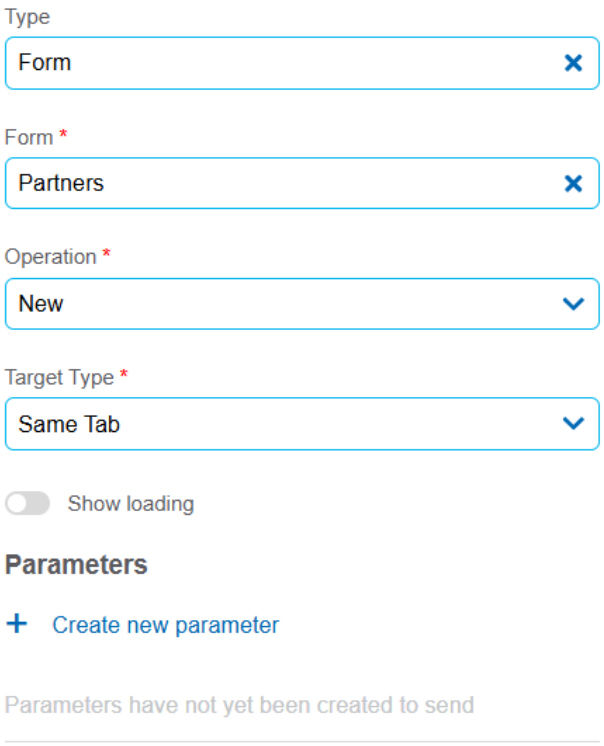
Destination Type
The available options for opening the object are displayed.

Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  - Modal Horizontal Size
    Defines its width.
  - Modal Vertical Size
    Defines its height.
- Iframe: selecting this option enables an additional property.
  - Destination Iframe
    Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Forms



Forms
The forms modeled in the environment are displayed.

Operation
Defines the operation made when selecting the object.

Possible Values

- Create: Indicates that the panel of the object selected in the previous property is opened for the creation of an instance.
- Update: Indicates that the panel of the object selected in the previous property is opened for the update of an instance.
- Show: Indicates that the panel of the object selected in the previous property is opened for the query of an instance.
- Grid: Indicates that the grid of the object selected in the previous property is opened.

Destination Type
The available options for opening the object are displayed.

Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  - Modal Horizontal Size
    Defines its width.
  - Modal Vertical Size
    Defines its height.
- Iframe: selecting this option enables an additional property.
  - Destination Iframe

Expands the iframes previously defined on the object.

- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

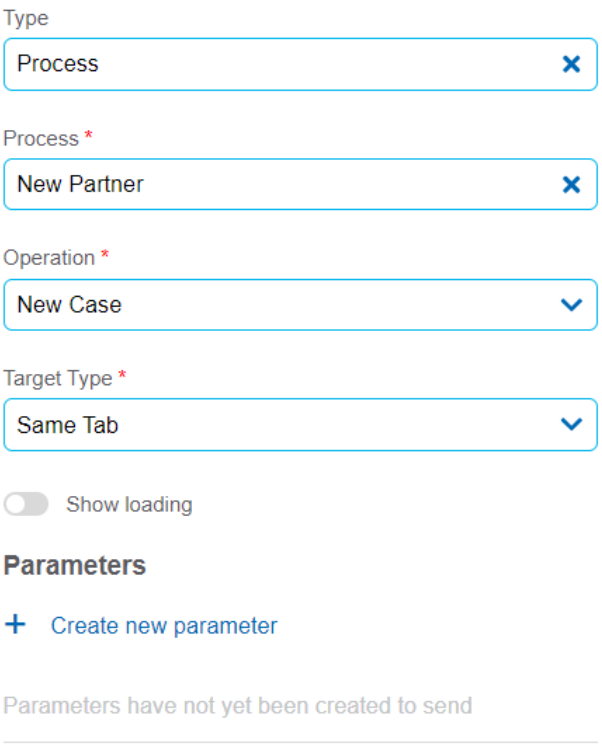
Show loading

Shows the loading icon and disables the user interaction until the redirection is over.

Parameters

Allows sending parameters to the selected object type.

Process



Process

The processes modeled in the environment are displayed.

Operation

Defines the operation made when selecting the object.

Destination Type

The available options for opening the object are displayed.

Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.

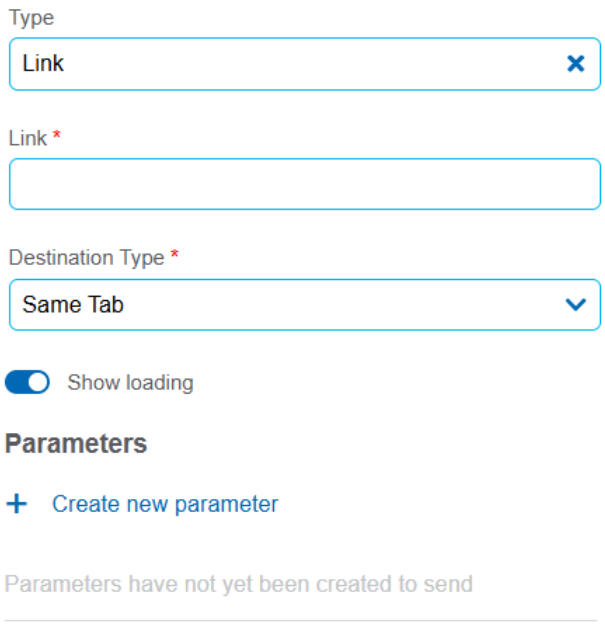
Modal Horizontal Size
Defines its width.
Modal Vertical Size
Defines its height.

- Iframe: selecting this option enables an additional property.
  Destination Iframe
  Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Parameters
Allows sending parameters to the selected object type.

Link



Link
Allows to enter any link.

Destination Type
The available options for opening the object are displayed.

Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  Modal Horizontal Size

Defines its width.

Modal Vertical Size

Defines its height.

- Iframe: selecting this option enables an additional property.

  Destination Iframe

  Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

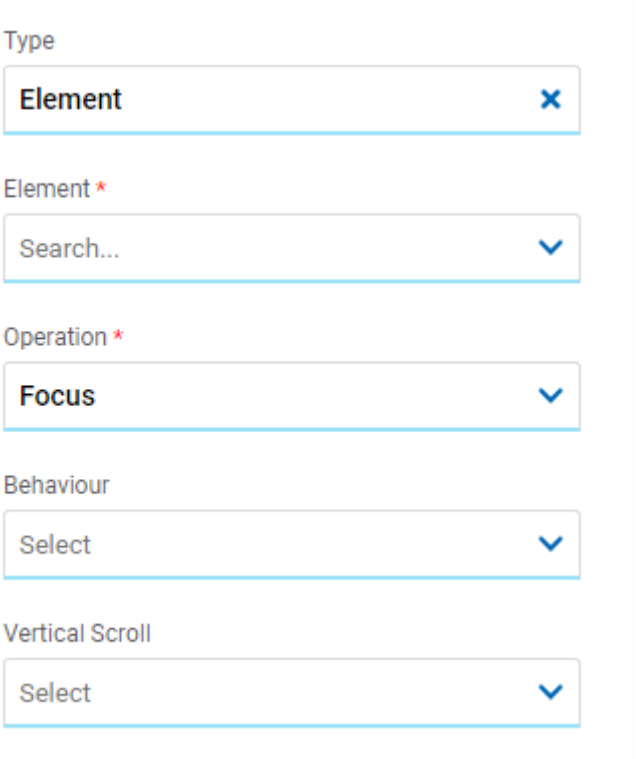
Show loading

Shows the loading icon and disables the user interaction until the redirection is over.

Parameters

Allows sending parameters to the selected object type.

Element



Element

The modeled elements in the object are displayed.

Operation

Defines the operation made when selecting the element.

Possible Values

- Focus
- Show
- Hide

- Show/Hide

Behaviour
Establishes the transition of the animation. It is only modelable if the "Focus" operation was selected.

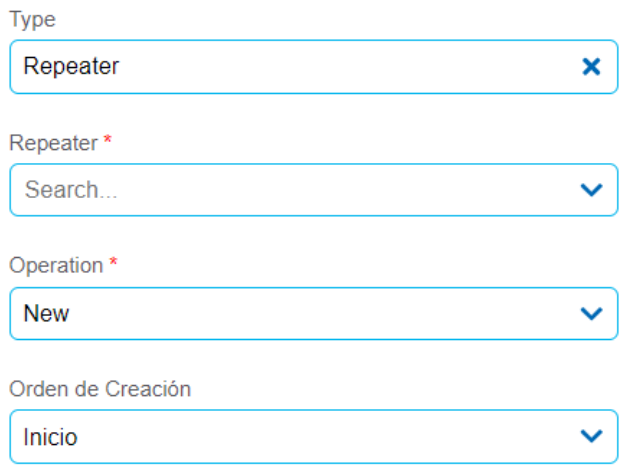
Possible Values

- Auto
- Smooth

Vertical Scroll
It is only modelable if the "Focus" operation was selected.
Possible Values

- Start: Moves to the start of the selected element.
- Closest: Moves to the closest position to the selected element from the element the event fires.
- Center: Moves to the center of the selected element.
- End: Moves to the end of the selected element.

Repeater



Repeater
The modeled repeaters from the object are displayed.

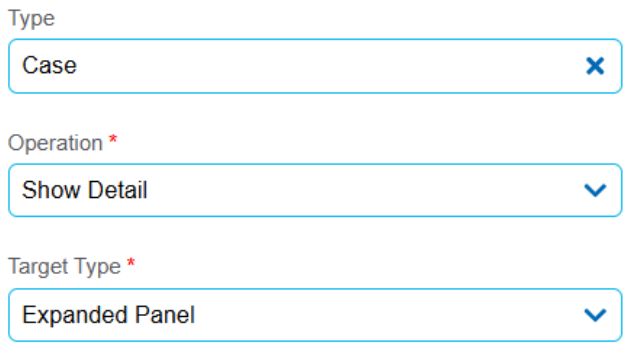
Operation
Defines the operation made when selecting the element.

Possible Values

- New: Allows for the creation of an item within the repeater. If this option is selected the additional properties are enabled.
  ### Creation Order
  It indicates in which position the new item of the container is added.
  Possible Values
  - Home
  - End

- Delete: Allows for the deletion of an item within the repeater.

Case



Case
Show case information.

Operation
Defines the operation made when selecting the element.

Possible Values

- Show Detail
- Show Graphic Execution

Destination Type
The available options for opening the object are displayed.

Possible Values

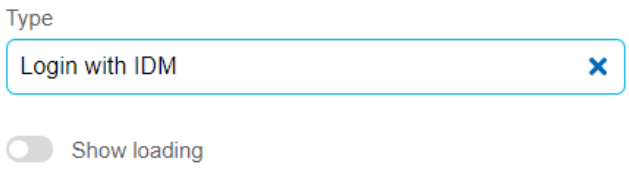
- Expanded Panel
- Modal: if this option is selected the additional properties are enabled.
  Modal Horizontal Size
  Define its width.
  Modal Vertical Size
  Define its height.

Back



It allows associating the event to go back in the browser to the element.

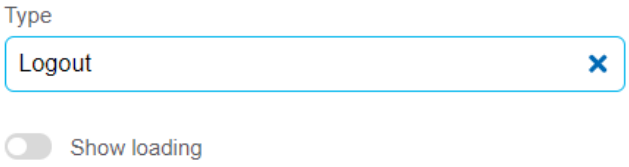
LogIn with IDM



Allows login with IDM.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

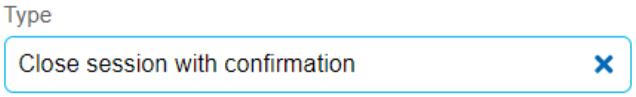
Logout



Allows logging the user out.

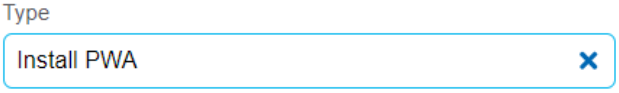
Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Close Session with Confirmation



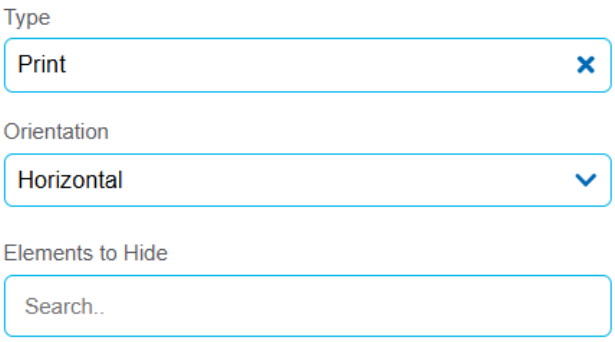
Displays a confirmation modal to logout the user.

Install PWA



Allows to install the application in the browser.

Print



This type of hyperlink associated with an element allows printing the content of the entity or page that is being executed.

Orientation

Determines the layout of the content to be printed.

Possible Values

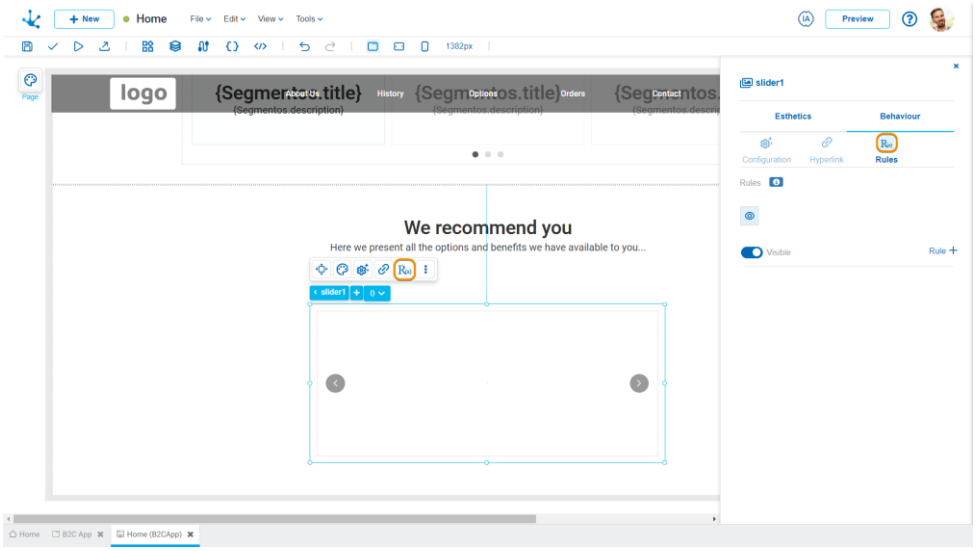
- Horizontal
- Vertical

Elements to hide

It allows selecting which elements of the entity or page should not be included in the printout.

Rule Properties

The rule properties panel of an element opens when clicking the icon R(x) of its context menu.



Properties

Rules

Embedded rules on behavior, validation and calculation can be defined, associated with an element, by using the wizard (ctrl + space).
Shows syntax examples for writing the rules.

Visible
Indicates whether the element is visible. If this property is not checked, the element is not displayed in the page.
Visible (default) Not visible
Rule + Opens an edit area where a rule to determine the visibility condition can be defined. If a rule is defined, the icon is displayed with light blue borders.

Operations when defining the rule:

Saves the new or modified rule

Cancels the operation

Operations once the rule is defined:

Edits the existing rule

Deletes the rule

Item

Items are defined within the slider and only the first one can be modified since the changes are applied to the rest.

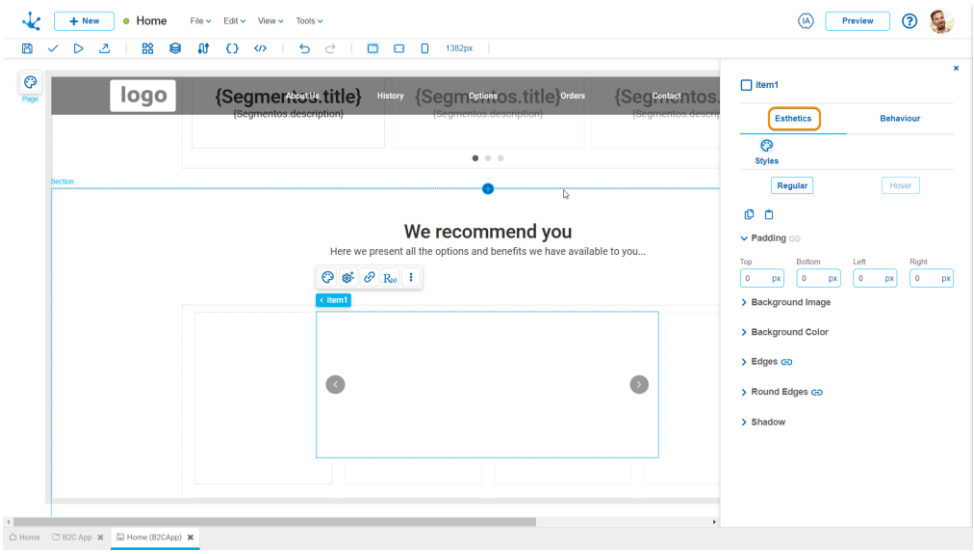
Classification of Properties

Properties are divided into two groups: esthetic properties and behavior properties.

Esthetic Properties

In the esthetic properties panel, the following are grouped:

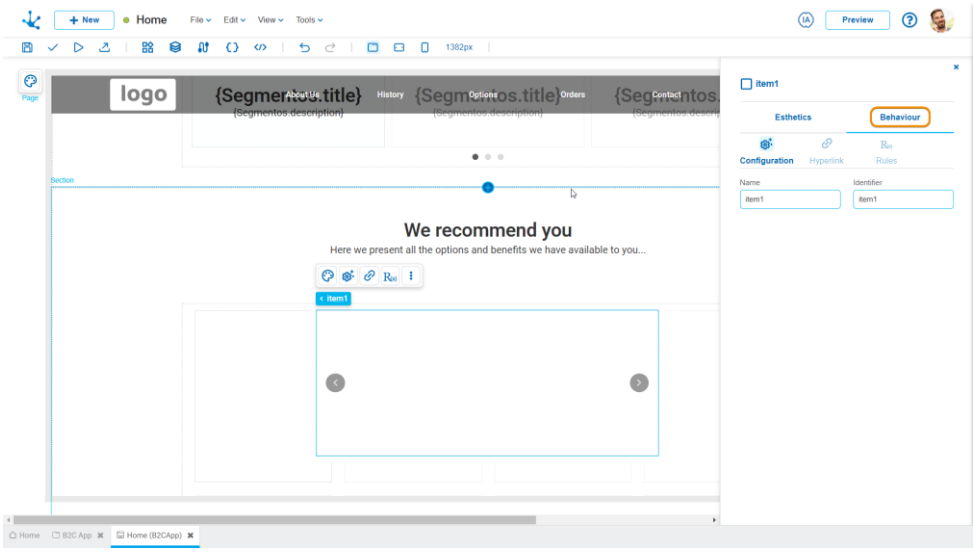
- Structure Properties
- Style Properties



Behavior Properties

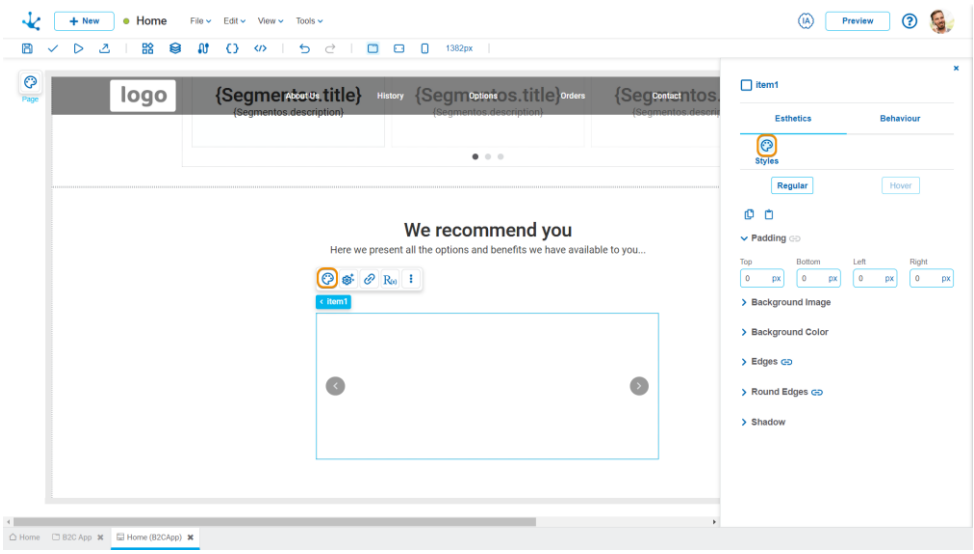
In the behavior properties panel, the following are grouped:

- Configuration Properties
- Hyperlink Properties
- Rules Properties



Style Properties

This type of element may take different states and for each of them different values for its properties may be modeled.



This type of element may take different states and for each of them different values for its properties may be modeled.



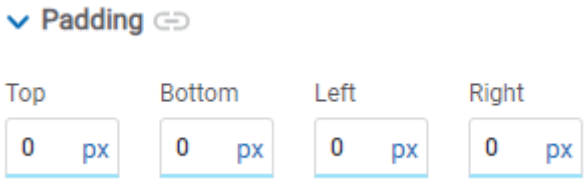
- Regular: The mouse pointer is not over the element.
- Hover: The mouse pointer is over the element.

Styles

It allows copying the entire current configuration of the style panel for the selected element (Ctrl+Alt+C).

It allows applying the previously copied configuration to another element of the same type (Ctrl+Alt+V).

Padding



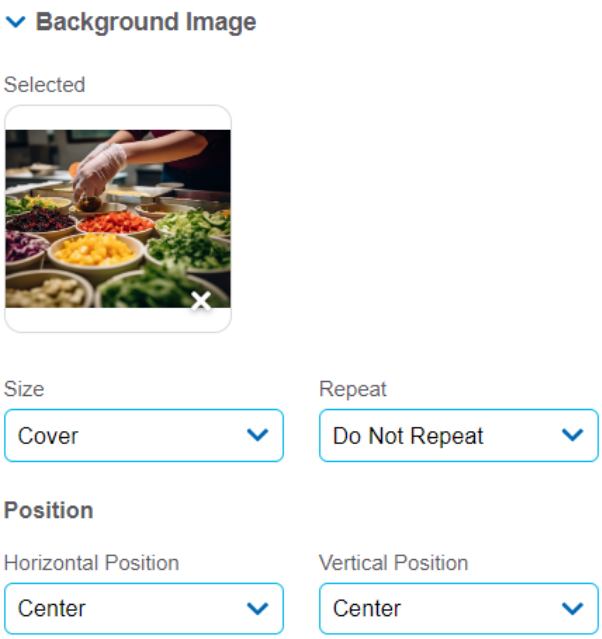
All padding space properties allow to create a space around the item borders (top, bottom, sides) and its content. The padding space configured is inward, while the margin is configured relative to a top item. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Allows values entered in one of the paddings to be copied to the other ones automatically.

Allows to indicate different values for each padding.

Background Image

It allows the addition of a background image to the item.

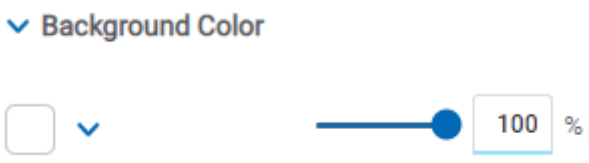


Selected
An image can be uploaded from the computer where it is being modeled.

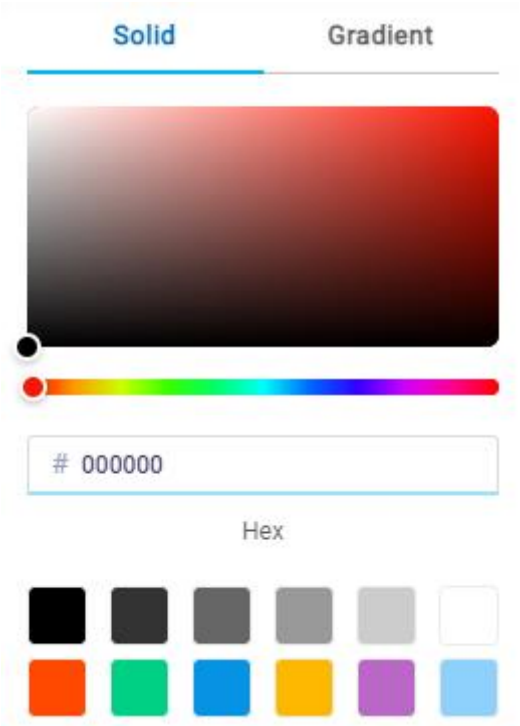
Background Color

It allows to model a background color for the element, it can be solid or gradient and different properties are defined for each one.

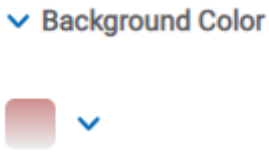
Solid



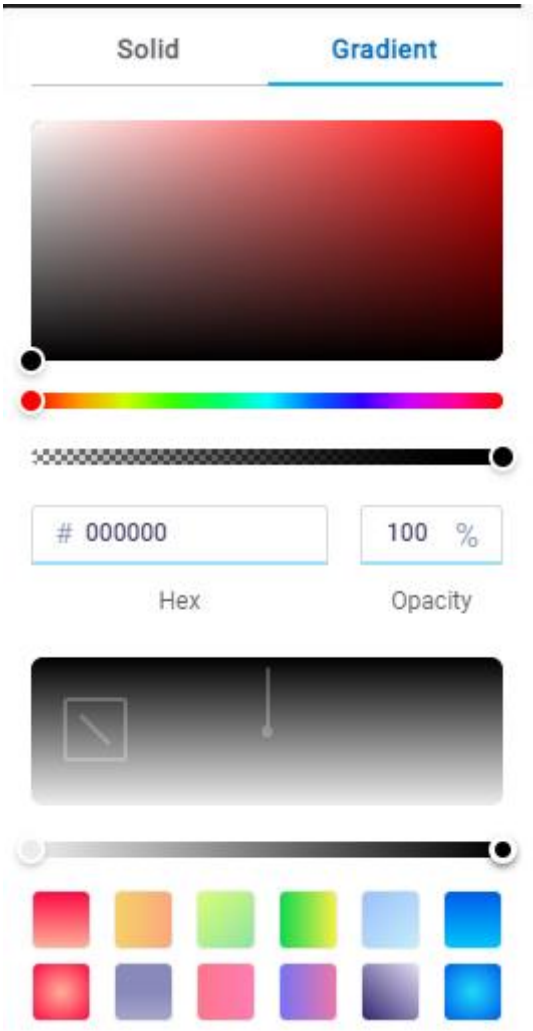
This option enables not only to select the color from a palette but also to define the degree of transparency.



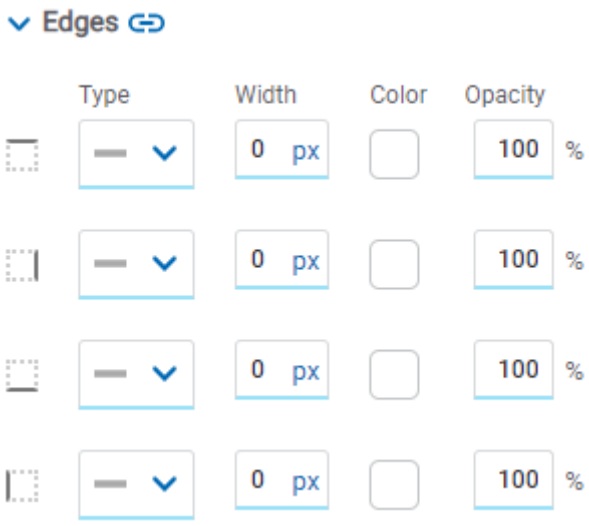
Gradient



This option enables to select not only the color from a palette but also to define its opacity and intensity according to the selected angle.



Edges



It allows to define the style of borders. Each one has its type, width, color and opacity defined, the latter as a percentage.

Allows the value entered in one of the borders to be copied to the other ones automatically.
Allows to indicate different values for each border.

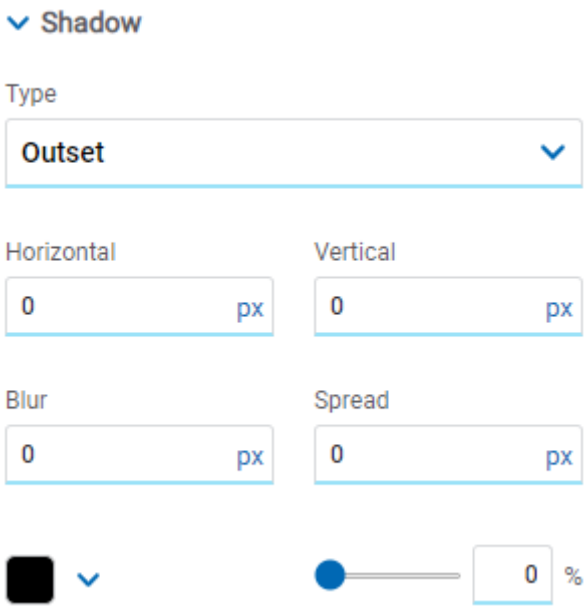
Round Edges



It allows to define the round edges at the corners of items.
It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Allows the value entered in one of the borders to be copied to the other ones automatically.
Allows to indicate different values for each border.

Shadow



Allows to define a shadow effect around the item.

Type

Possible Values

- Outset
- Inset

Horizontal
Horizontal size of the shadow to the right of the item. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Vertical
Vertical size of the shadow below the item. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

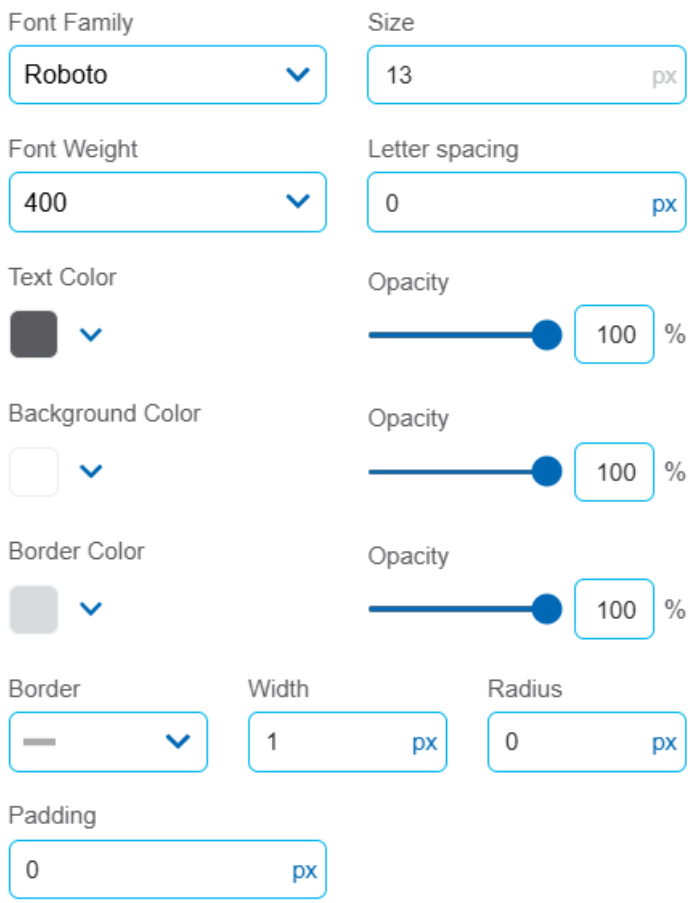
Blur
The greater this value is, the greater and lighter the shadow becomes. If not specified, its value is 0 and the shadow border is darker. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).

Spread
Positive values will cause the shadow to expand and grow bigger, while negative values will cause the shadow to shrink. If not specified, its value is 0 and the shadow will have the same size as the item.

Color
Defines the color of the shadow.

Transparency
Defines the transparency of the shadow.

Tooltip



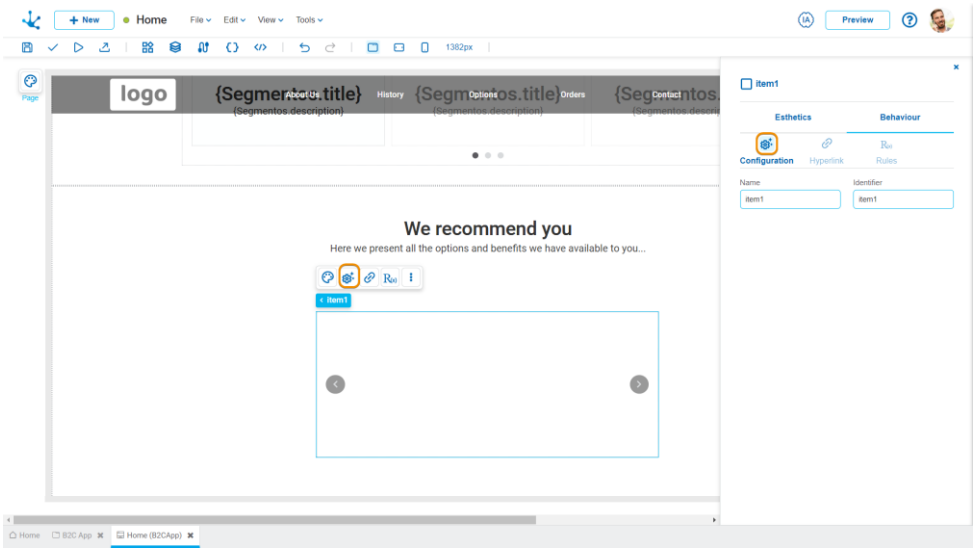
Source
It allows customizing the font, size, thickness and letter spacing of the pop-up message.

Colors
The color and opacity of the text, background and border can be defined. Borders are set uniformly for all sides of the tooltip.

Border
The border type, width, and border radius can be configured.

Configuration Properties

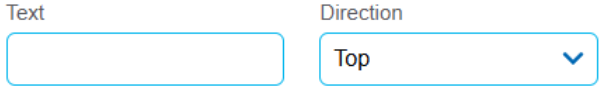
The configuration properties panel of an element opens when clicking the icon  of its context menu.



Name
Name used to reference the element during modeling.

Identifier
Uniquely identifies the element. It is used in the Javascript SDK.

Tooltip
Pop-up message that appears when hovering over an element, providing additional information.



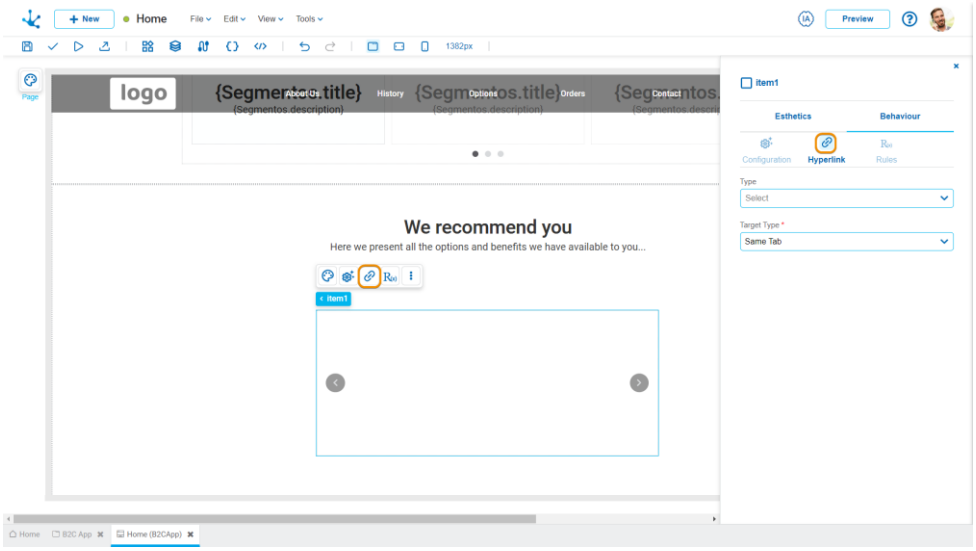
Text
Content of the message being displayed.

Direction

Position of the message in relation to the element (top, bottom, left, or right).

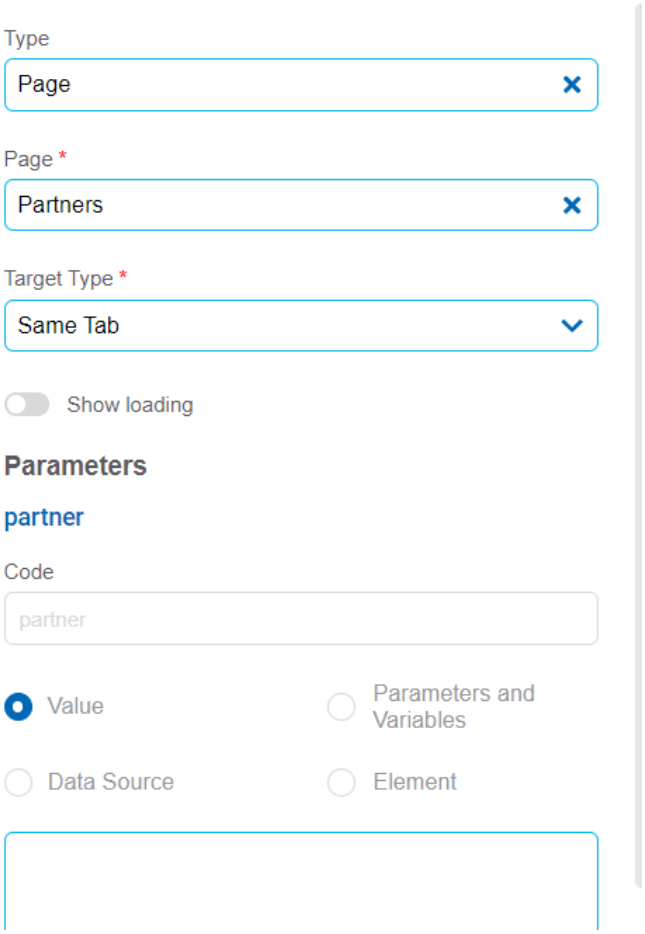
Hyperlink Properties

The hyperlink properties panel of an element opens when clicking the icon  of its context menu.



Define the behavior of the element when selecting it. Different properties are enabled depending on the type of object selected.

Page



Page
The pages modeled in the environment are displayed.

Destination Type
The available options for opening the object are displayed.

Possible Values

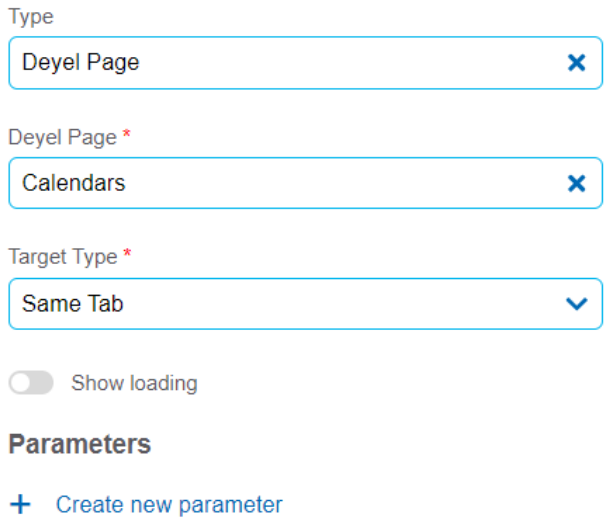
- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  Modal Horizontal Size
  Defines its width.
  Modal Vertical Size
  Defines its height.
- Iframe: selecting this option enables an additional property.
  Destination Iframe
  Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading

Shows the loading icon and disables the user interaction until the redirection is over.

Parameters
If there are parameters, their value can be specified with text, parameters and variables or data source.

Deyel Page



Deyel Page
The pages belonging to **Deyel** are displayed.

Destination Type
The available options for opening the object are displayed.

Possible Values

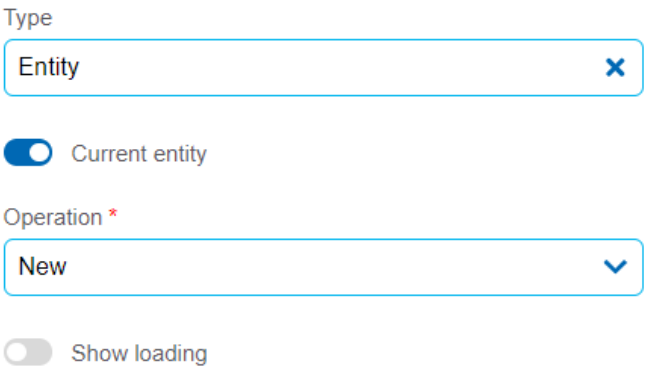
- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  - Modal Horizontal Size
    Defines its width.
  - Modal Vertical Size
    Defines its height.
- Iframe: selecting this option enables an additional property.
  - Destination Iframe
    Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Parameters

Allows sending parameters to the selected object type.

Entity



Current Entity

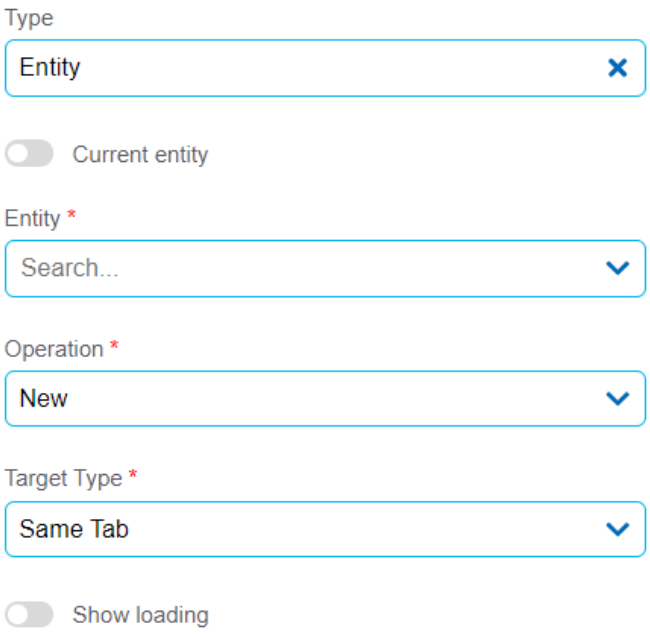
If this property is checked, the operation applies directly to the entity being modeled.

Operation

Defines the operation made when selecting the object.

Possible Values

- New: Indicates that the current entity panel is opened for the creation of an instance.
- Modify: Indicates that the current entity panel is opened for the update of an instance.
- Delete: Indicates that the current entity panel is opened for the deletion of an instance.



Entity
If the property Current Entity, is not checked, the entities modeled in the environment are displayed.

Operation
Defines the operation made when selecting the object.

Possible Values

- New: The panel of the entity selected in the previous property is opened to create an instance.
- Update The panel of the entity selected in the previous property is opened to update an instance.
- Show: The panel of the entity selected in the previous property is opened to show an instance.
- Grid: Indicates that the grid of the entity selected in the previous property is opened.

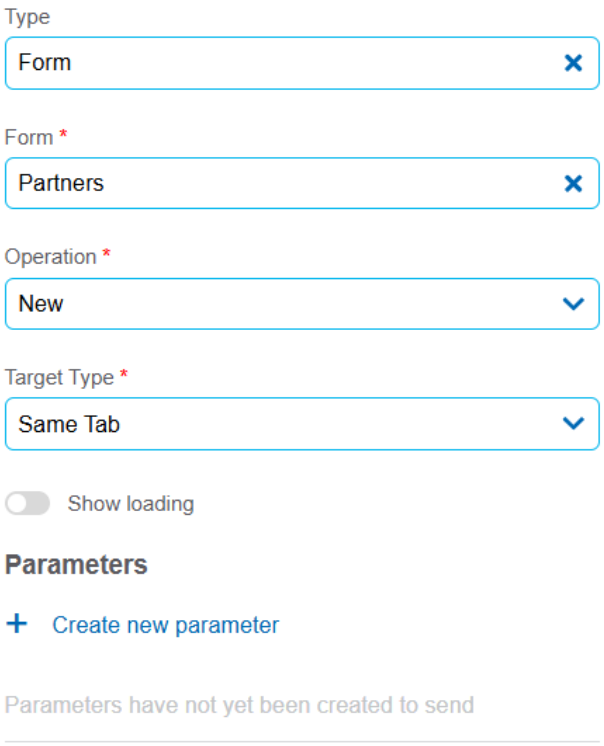
Destination Type
The available options for opening the object are displayed.

Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  Modal Horizontal Size
  Defines its width.
  Modal Vertical Size
  Defines its height.
- Iframe: selecting this option enables an additional property.
  Destination Iframe
  Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Forms



Forms
The forms modeled in the environment are displayed.

Operation
Defines the operation made when selecting the object.

Possible Values

- Create: Indicates that the panel of the object selected in the previous property is opened for the creation of an instance.
- Update: Indicates that the panel of the object selected in the previous property is opened for the update of an instance.
- Show: Indicates that the panel of the object selected in the previous property is opened for the query of an instance.
- Grid: Indicates that the grid of the object selected in the previous property is opened.

Destination Type
The available options for opening the object are displayed.

Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  - Modal Horizontal Size
    Defines its width.
  - Modal Vertical Size
    Defines its height.
- Iframe: selecting this option enables an additional property.
  - Destination Iframe

Expands the iframes previously defined on the object.

- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

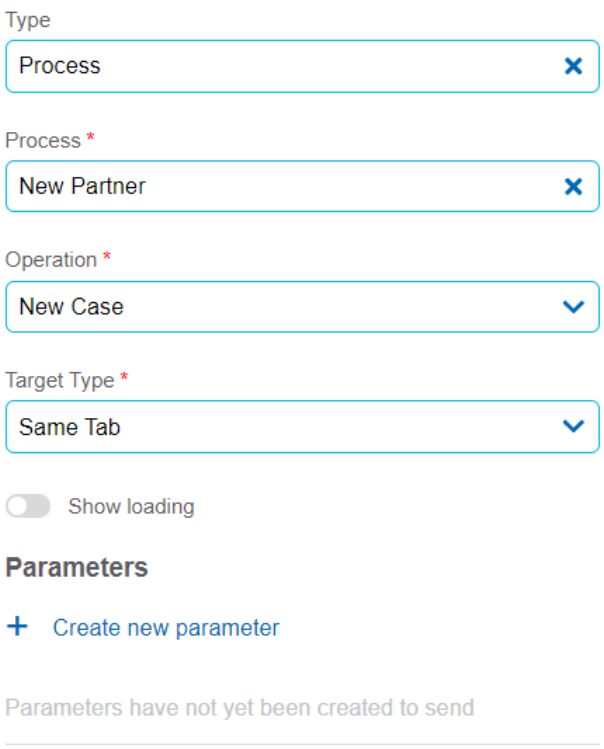
Show loading

Shows the loading icon and disables the user interaction until the redirection is over.

Parameters

Allows sending parameters to the selected object type.

Process



Process

The processes modeled in the environment are displayed.

Operation

Defines the operation made when selecting the object.

Destination Type

The available options for opening the object are displayed.

Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.

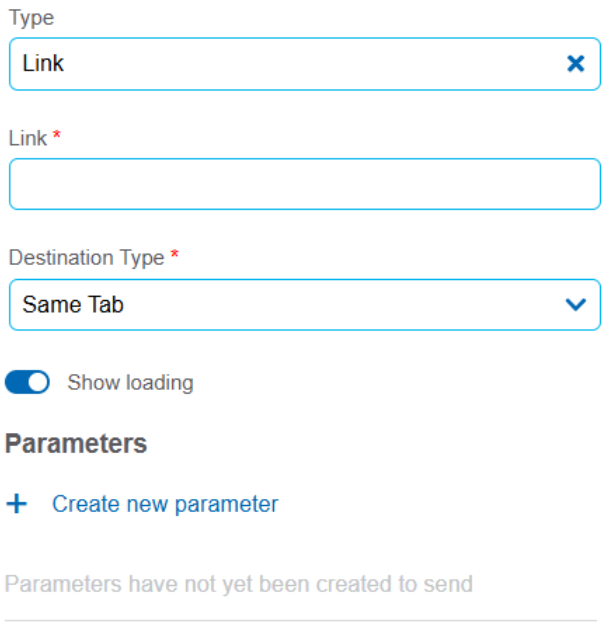
Modal Horizontal Size
Defines its width.
Modal Vertical Size
Defines its height.

- Iframe: selecting this option enables an additional property.
  Destination Iframe
  Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Parameters
Allows sending parameters to the selected object type.

Link



Link
Allows to enter any link.

Destination Type
The available options for opening the object are displayed.

Possible Values

- Same Tab
- New Window
- Modal: selecting this option enables additional properties.
  Modal Horizontal Size

Defines its width.

Modal Vertical Size

Defines its height.

- Iframe: selecting this option enables an additional property.

  Destination Iframe

  Expands the iframes previously defined on the object.
- Top Iframe: defines the iframe that contains the specific element where the hyperlink is being applied as destination.
- Parent Iframe: defines the iframe that contains the entire current object as destination.

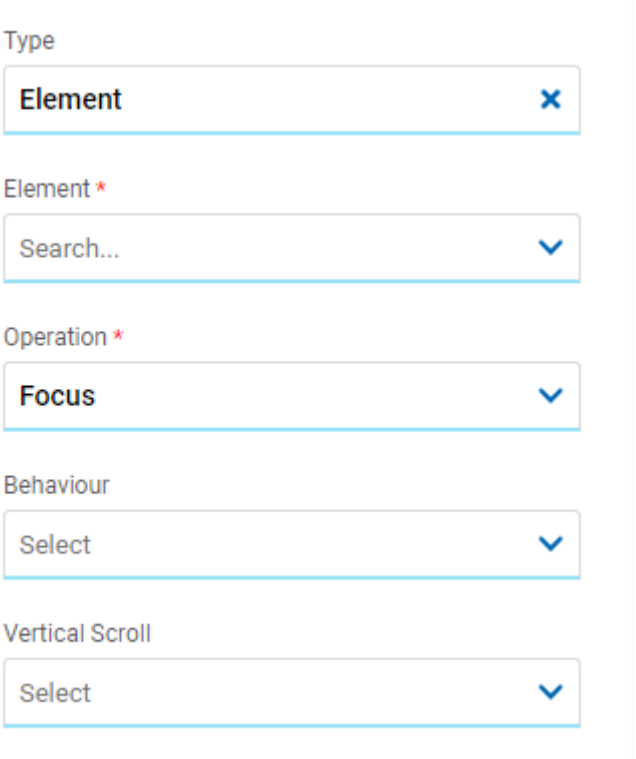
Show loading

Shows the loading icon and disables the user interaction until the redirection is over.

Parameters

Allows sending parameters to the selected object type.

Element



Element

The modeled elements in the object are displayed.

Operation

Defines the operation made when selecting the element.

Possible Values

- Focus
- Show
- Hide

- Show/Hide

Behaviour
Establishes the transition of the animation. It is only modelable if the "Focus" operation was selected.

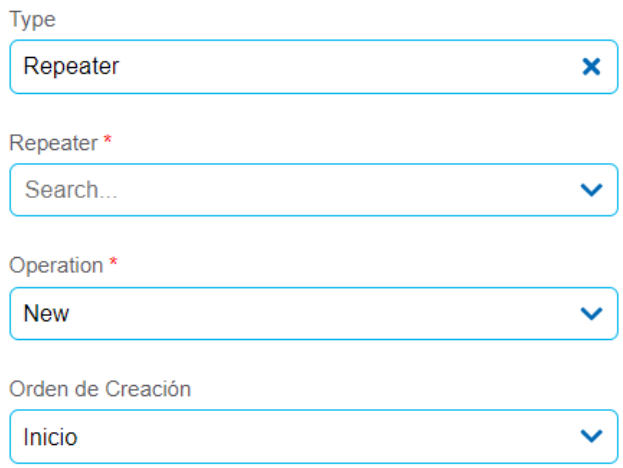
Possible Values

- Auto
- Smooth

Vertical Scroll
It is only modelable if the "Focus" operation was selected.
Possible Values

- Start: Moves to the start of the selected element.
- Closest: Moves to the closest position to the selected element from the element the event fires.
- Center: Moves to the center of the selected element.
- End: Moves to the end of the selected element.

Repeater



Repeater
The modeled repeaters from the object are displayed.

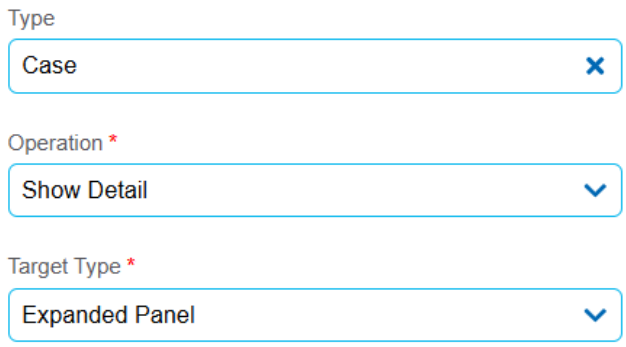
Operation
Defines the operation made when selecting the element.

Possible Values

- New: Allows for the creation of an item within the repeater. If this option is selected the additional properties are enabled.
  ### Creation Order
  It indicates in which position the new item of the container is added.
  Possible Values
  - Home
  - End

- Delete: Allows for the deletion of an item within the repeater.

Case



Case
Show case information.

Operation
Defines the operation made when selecting the element.

Possible Values

- Show Detail
- Show Graphic Execution

Destination Type
The available options for opening the object are displayed.

Possible Values

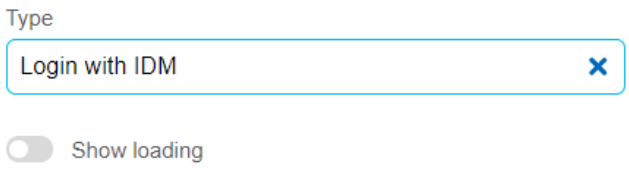
- Expanded Panel
- Modal: if this option is selected the additional properties are enabled.
  Modal Horizontal Size
  Define its width.
  Modal Vertical Size
  Define its height.

Back



It allows associating the event to go back in the browser to the element.

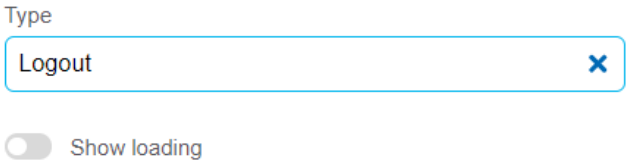
LogIn with IDM



Allows login with IDM.

Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

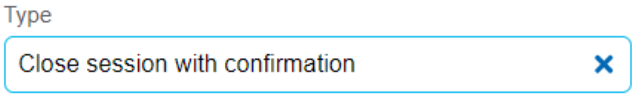
Logout



Allows logging the user out.

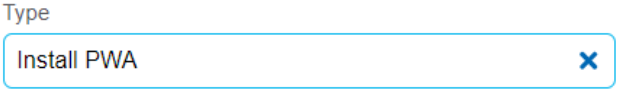
Show loading
Shows the loading icon and disables the user interaction until the redirection is over.

Close Session with Confirmation



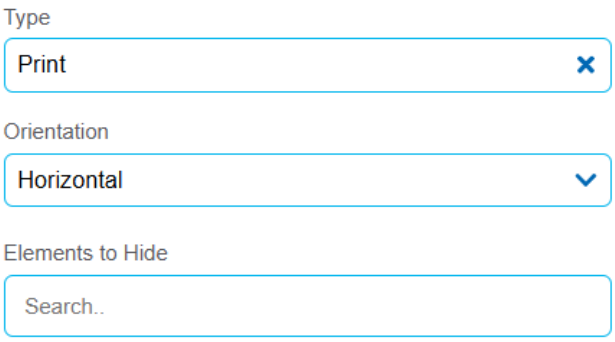
Displays a confirmation modal to logout the user.

Install PWA



Allows to install the application in the browser.

Print



This type of hyperlink associated with an element allows printing the content of the entity or page that is being executed.

Orientation

Determines the layout of the content to be printed.

Possible Values

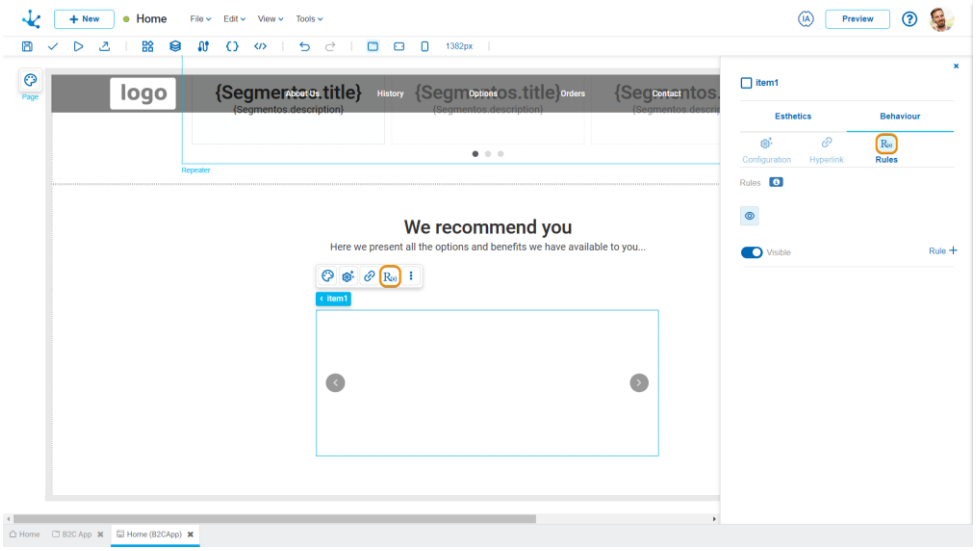
- Horizontal
- Vertical

Elements to hide

It allows selecting which elements of the entity or page should not be included in the printout.

Rule Properties

The rule properties panel of an element opens when clicking the icon R(x) of its context menu.



Properties

Rules

Embedded rules on behavior, validation and calculation can be defined, associated with an element, by using the wizard (ctrl + space).

Shows syntax examples for writing the rules.

Visible

Indicates whether the element is visible. If this property is not checked, the element is not displayed in the page.

Visible (default) Not visible

Rule + Opens an edit area where a rule to determine the visibility condition can be defined. If a rule is defined, the icon is displayed with light blue borders.

Operations when defining the rule:

Saves the new or modified rule

Cancels the operation

Operations once the rule is defined:

Edits the existing rule

Deletes the rule

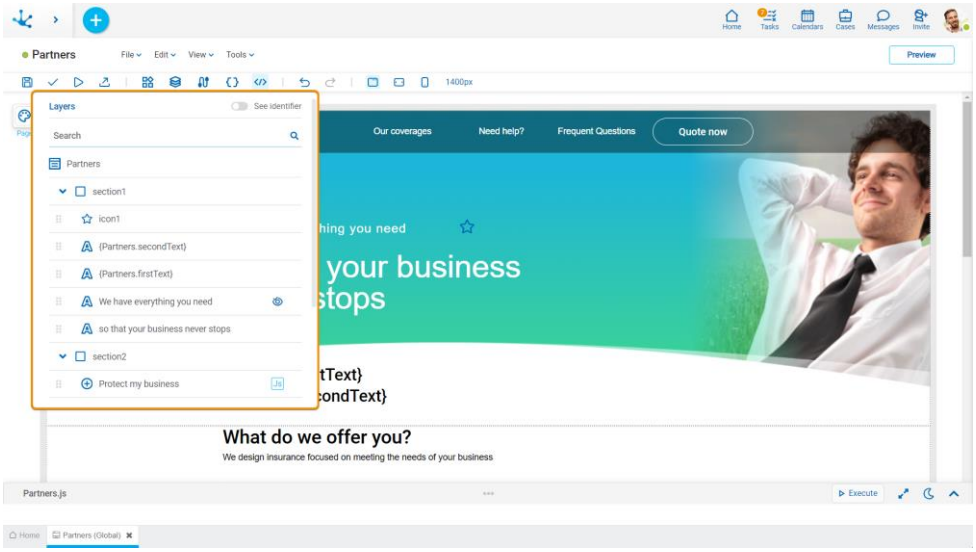
Events

Items allow the use of different events.

Event	Description
onMouseIn()	It is executed when the cursor is positioned on the element.
onMouseOut()	It is executed when the cursor moves out of the element.
onClick()	It is executed when clicking on the element.
onDoubleClick()	It is executed when clicking twice on the element.
onInit()	It is executed before the element is loaded.
afterViewInit()	It is executed after viewing the element.
onViewportEnter()	It is executed when the element is visible.
onViewportLeave()	It is executed when the element is no longer visible.

3.4.13.1.3.2. Layers

It allows to display the page elements organized hierarchically. First, sections are displayed, ordered as they are located on the page and within each one, all its elements can be seen.



Operations

Locate Elements

Hovering over an element in the layers panel highlights such element in the modeling area, making it easier to identify without the need to select it. If selected, in addition to being highlighted, its context menu is displayed.

When an element is selected in the modeling area and the layers panel is opened, the chosen element is displayed in the hierarchy.

Filter Elements

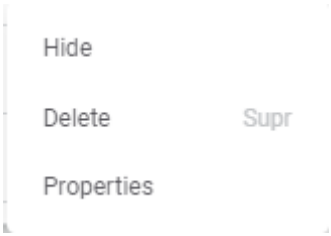
When performing a search, matching elements are highlighted in the layers menu, while those that do not meet the search criteria collapse their hierarchical display, if they had one.

Locate and Show Hidden Elements

Hidden elements on the page are displayed in gray in the layers panel next to the icon . When pressing this icon, the element becomes visible. When clicking on the name instead, the element is temporarily displayed in the modeling area.

Elements Panel

Selecting the icon displays a panel with different options.



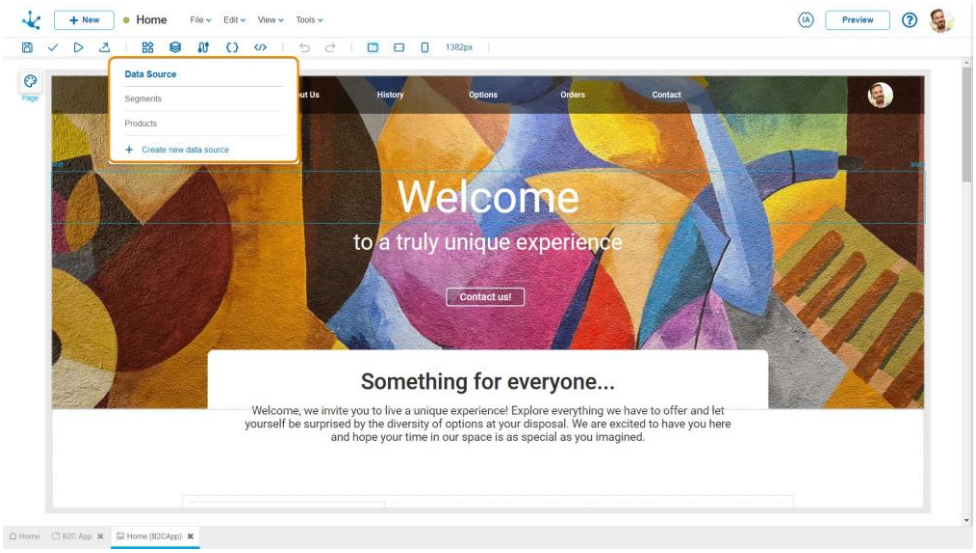
Reorder Elements

It allows to easily order the elements within the same superior element, dragging them in the list to the desired position.

3.4.13.1.3.3. Data Source

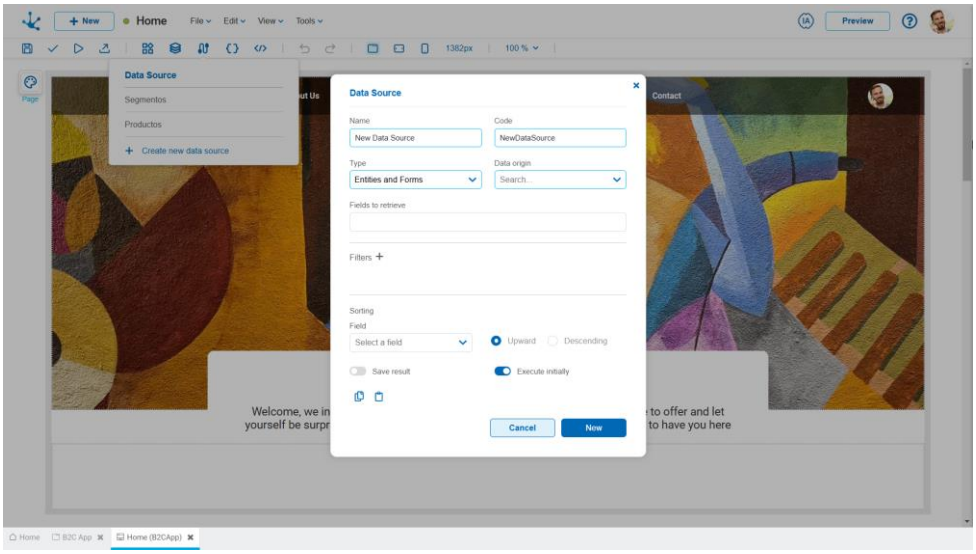
A data source retrieves data from entities and forms or by executing advanced rules depending on the properties modeled on it.

Through the icon , a set of data sources to be used on the page can be defined.



Operations can be performed on each line of the existing sources. Click on the icons and that delete or update it respectively.

A new data source can be created from the icon . When defining a new data source or selecting an existing one, a panel with its information opens.



Properties

Data sources can originate in entities, forms or in rules, in all cases, they share some properties and have specific ones.

Name
Name used in the modeler to reference the data source.

Code
It is used internally to reference the data source.

Type
Determines the type of object from which the data is to be taken. The possible values for this property are “Entities and Form” and “Rule”.

Data origin
It allows to select the object from which the data will be taken.

Sorting

Field
It allows to select the field the information will be sorted by in ascending or descending order.

Save result
It allows defining whether the data obtained by a data source should be stored temporarily. This functionality is useful for optimizing page performance and reducing unnecessary requests to the server.

Initially execute
It allows configuring whether the data source is executed automatically when the page is loaded.

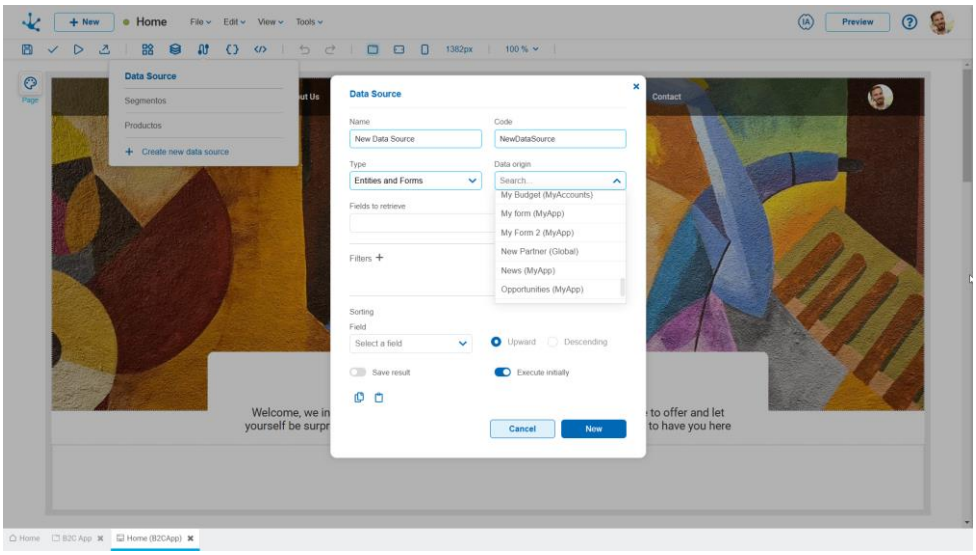
Copy data source

Allows copying the entire current definition of the selected data source to another page.

Paste data source
Allows applying the entire current definition of the copied data source to another page.

Entities and Forms as Data Source

Specific properties must be modeled when the data source is of entity type or form type.



Properties

Fields to retrieve
It allows the fields of the selected object to be selected.

Filter
The icon + allows to expand a panel with the available fields to define the selection criteria of the data.

Depending on the type of field, filter conditions are different, fixed values, variables or parameters can be entered and combined by means of operators. Once the filter criteria have been defined, the "Add" button should be pressed so as to define the filter.

Filter Conditions

Numeric Fields
The value entered should be numeric.
Search criteria:

- Greater than
- Greater equal to
- Less than
- Less equal to
- Between
- With Data

- No Data

Alphanumeric Fields

Enter a text to search for.
Search criteria:

- Contains
- Equal to
- Starts with
- Does not start with
- No Data
- With Data

Date Fields

A calendar opens to select the date and it can be filtered using different search criteria.
Options:

- Today
- Last 7 days
- Current Month
- Current Year
- Last Month
- Last Year
- From (Requires selection of a start date)
- To (Requires selection of an end date)
- Range (Requires the selection of a start date and an end date)
- Equal (Requires selection of a date)

Date Time Fields

A calendar opens to select the date and time, it can be filtered using different search criteria.

- From
- Until
- Equal
- Range

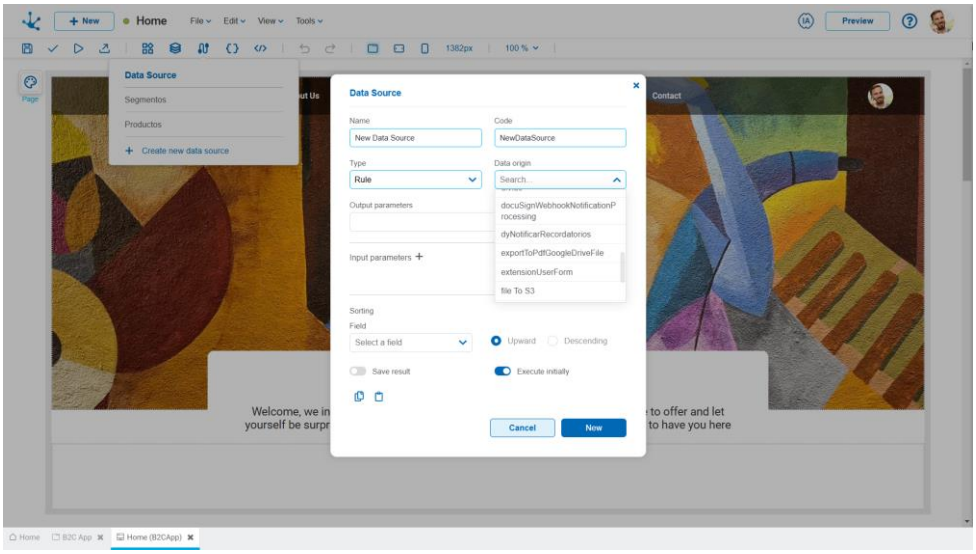
Value Lists Fields

The list values are displayed.
Search criteria:

- Included
- Not Included
- With Data
- No Data

Rule as Data Source

When the data source is of the rule type, specific properties must be modeled.



Properties

Output parameters
The output parameters of the selected rule are displayed.

Input parameters
The icon + allows expanding a panel with all the input parameters available for selection.
Page parameters, fixed values or variables can be entered for each chosen parameter.

3.4.13.1.3.4. Parameters and Variables

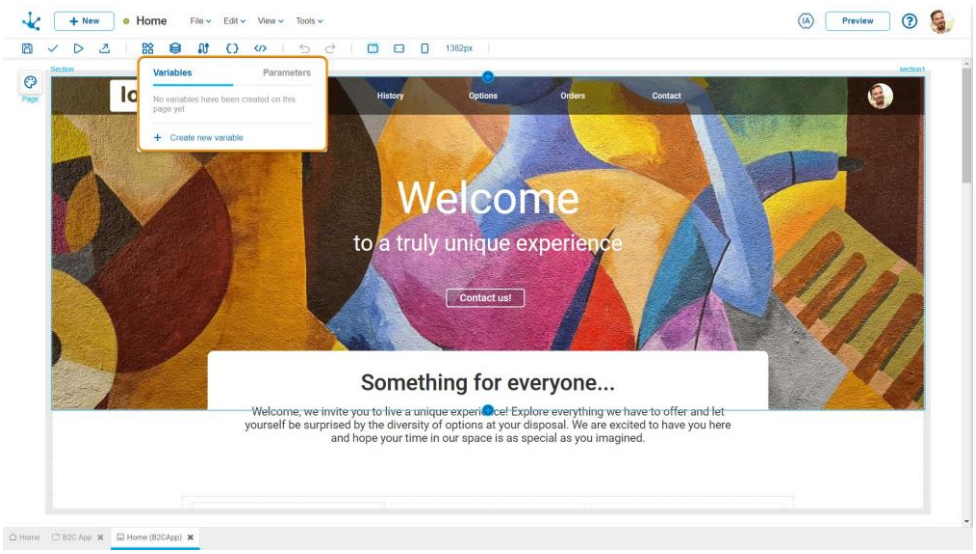
The modeled page allows to define variables and input parameters by means of the icon {}. Once the panel is opened, variables are displayed by default, parameters can also be selected.

Variables

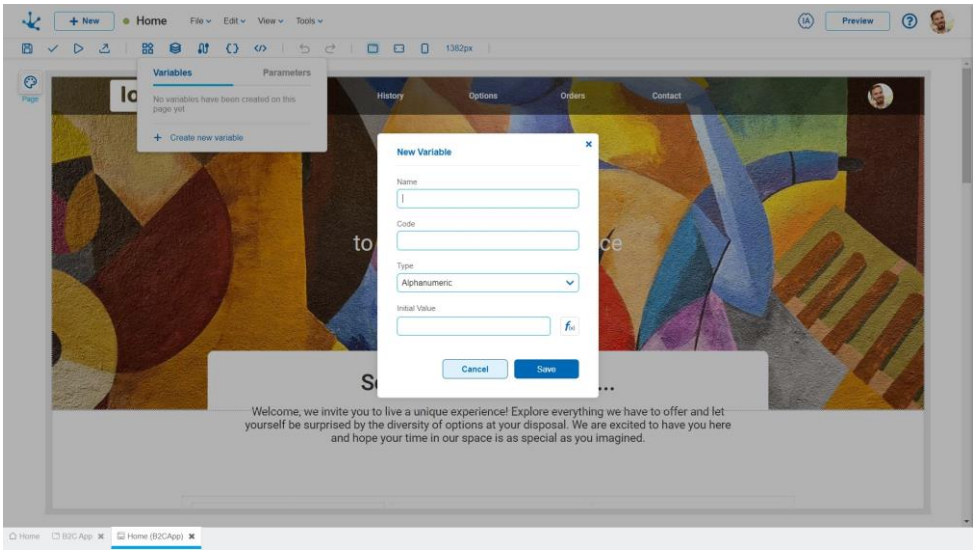
Variables used in pages are defined.

They allow to define data that can be used as the value of elements and in the filters of data sources. They can only be modified by code.

Operations can be performed on variables. Click on the icons 🗑 and ✎ on each row to delete or update it respectively.



A new variable can be created from the icon +. When a new variable is defined or an existing one is selected, a panel with its information opens.

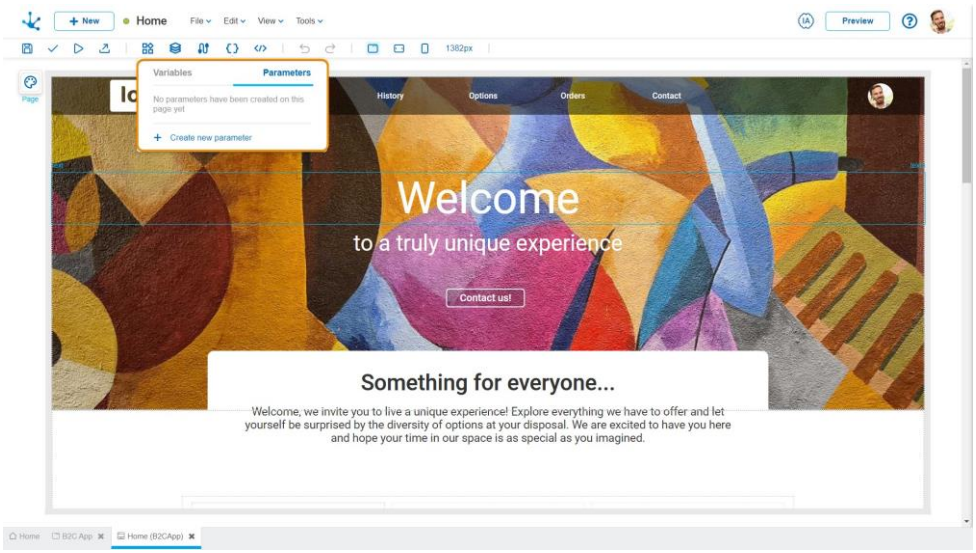


Parameters

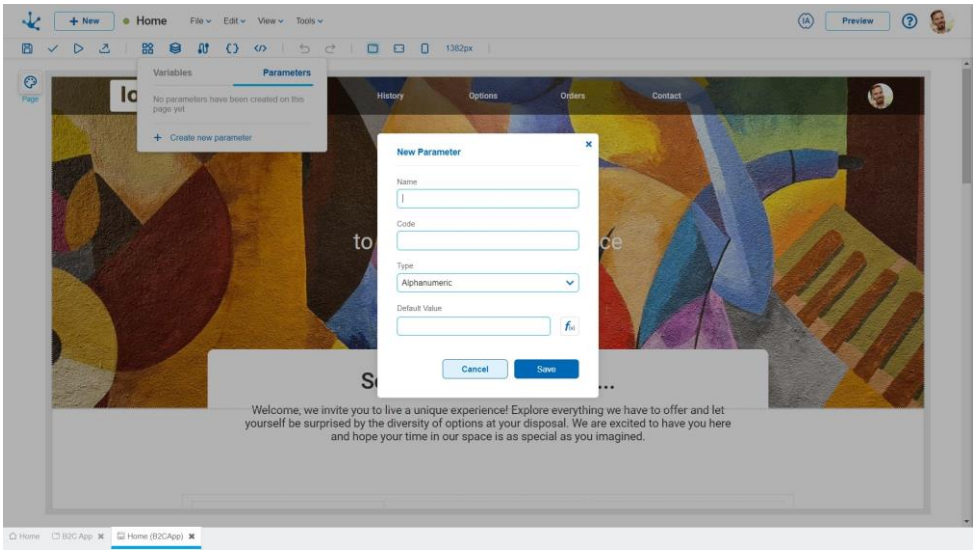
The parameters used in pages are defined.

Data received by the page that cannot be modified. As with variables, they can be used in elements and data sources.

Operations can be performed parameters. Click on the icons and on each row to delete or update it respectively.



A new parameter can be created from the icon + . When defining a new parameter or selecting an existing one, a panel with its information opens.



> *Once the page is published, it can be accessed from a web browser with the full access path, followed by the sign "?", the name of each parameter, the sign "=" and its value. If there is more than one parameter, they are concatenated by the sign "&". For example "myenvironment.deyel.com/pages/<URL property content>?<parameter name>=<parameter value>".*

3.4.13.1.3.5. Code Editing

It is the area within the modeler where the functionality of the page and its elements can be extended through code.

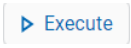
This area contains two tabs:

- JavaScript Tab: It allows writing custom JavaScript functions to add interactivity to the page, dynamically handle data, manage events, and integrate with other services or scripts.

- CSS Tab (Cascading Style Sheets): It is used to define custom style rules that modify the appearance of the page and its elements. Here, specific styles such as colors, sizes, borders, and animations can be applied, ensuring that the design meets the project's requirements.

The code editing area is displayed at the bottom of the page, following a bar with icons. The icon in the center of the bar allows you to modify the size of the area, while the icon on the far right allows to close it.

Editor Options



Performs the same operation as the preview.


Increases the panel size and covers the page modeling area.

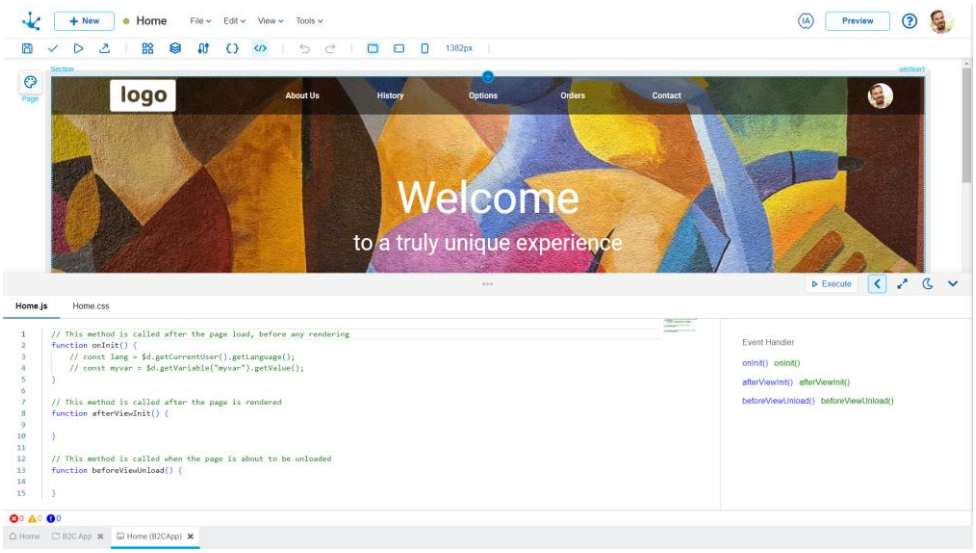

Returns the panel to its previous size.


Changes the style of the code editor to dark mode.


Changes the style of the code editor to the previous mode.

JavaScript Tab

Each element has a list of predefined events that depend on the type of element.



Sections

- Javascript code editor of the page: It allows adding and updating functions associated with the events of elements or pages. Includes an auto-complete functionality that speeds up writing by automatically suggesting SDK functions.
- Event handler: The event handler is displayed to the right of the code editor, it contains the identifier and the list of events of the selected element. If no element is selected, the list of page events expands.
- Message monitoring console: Located below the code editor, this advanced tool makes it easy to monitor and manage messages during development.

JavaScript Function Operations

Associate

The page has predefined JavaScript functions associated with its events, unlike its elements, which do not have them.

To create and associate a JavaScript function with an event, select the event in the event handler, enabling the new function name to be entered. By default, a name is generated consisting of the element identifier concatenated with the event name. Clicking on the icon + creates the JavaScript function associated with the event in the code area, and the name of the function is added to the event list.

If the name of the JavaScript function already exists within the code, a number is automatically suffixed to the name.

Select

Selecting a JavaScript function in the event handler positions the cursor over it in the code editor.

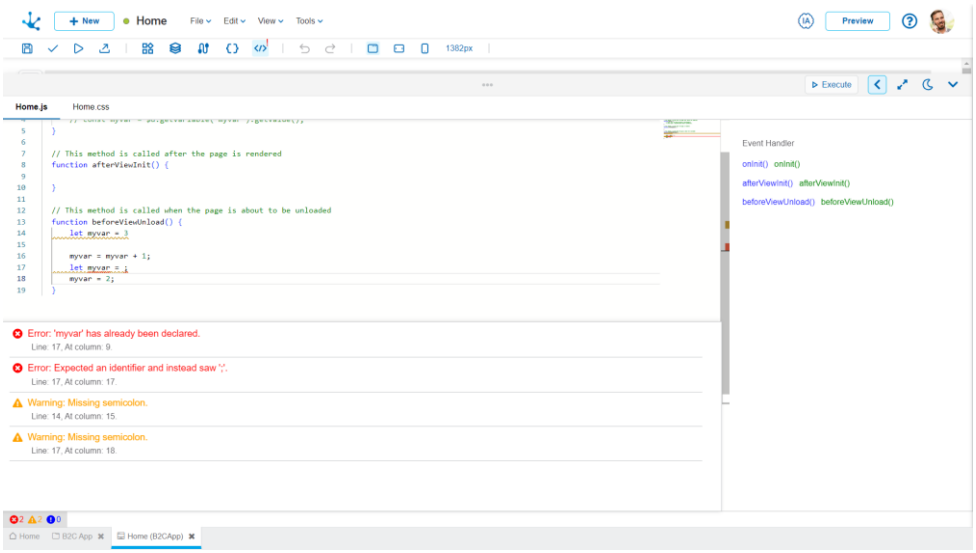
Disassociate

To disassociate a JavaScript function from an event, click on the icon that is displayed when hovering the cursor over it in the event list. In this way, the association of the function with the event is deleted but it is not deleted from the code.

> *If the JavaScript function is deleted from the code area, it still appears associated with the event handler. However, the event does not execute any function and does not throw an error.*

Message Monitoring Console

You can access this console by clicking on the icons corresponding to the messages, which are located in the bottom bar of the code editor.



Upon accessing the console, the messages are displayed in an organized and categorized manner. Details about each message are provided, including its location in the code and possible solutions. The messages are updated in real time, with continuous monitoring that allows you to visualize errors and warnings as they are detected.

Message Types

Syntax Errors: Detects and notifies syntax errors in the code.

Warnings: Highlights potential issues that could affect the performance or quality of the code, providing alerts on aspects that may need attention or review.

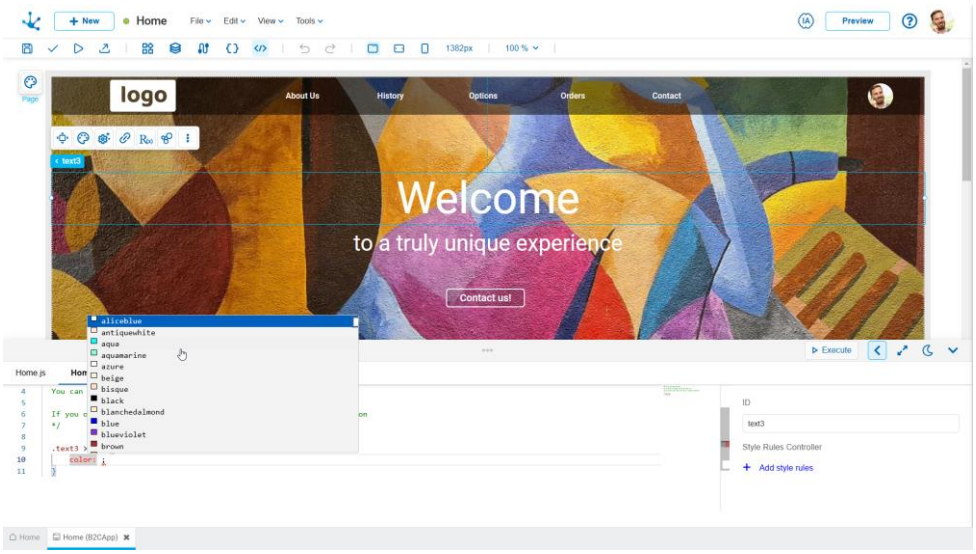
Usage Information: Identifies situations in the implementation of methods and functions, offering recommendations to improve the code.

CSS Tab

The Styles tab allows applying CSS rules to change the appearance of elements within the modeling tool.

Sections

- CSS code editor: Rules can be written manually or by selecting attributes using a wizard, making it easier to configure colors, fonts, sizes, margins, and other visual settings.
- Rule controller: The rule controller is displayed to the right of the code editor, which allows managing and applying specific styles to selected elements. If no element is selected, no associated rules are displayed, but if an element is selected, custom styles can be added, modified, or deleted for that element.



Steps to change the style of an element

- Navigate to the CSS tab: Initially, the panel does not contain applied style rules.
- Selecting an element: When clicking on an element within the modeling area, its ID is displayed in the interface. In the Style Rule Controller section, the option "+ Add style rule" is displayed.
- Add style rule: When this option is clicked, a CSS rule is automatically generated in the code editor, named according to the identifier of the selected element. Multiple rules can be created for each element.
- Edit the rule: Within the rule body, press "CTRL + spacebar" to invoke the wizard. Select the attribute and the value to be modified so as to customize the design as needed.
- Apply changes: Click on the button "Execute" from the editor to display the updates.

Eventos

The pages, their sections, and their elements have different events defined. These events are displayed in the area of page code, where code can be entered, allowing it to execute when a specific event occurs.

Page

Event	Description
onInit()	It is executed before loading the page.
afterViewInit()	It is executed after rendering the page.

Event	Description
beforeViewUnload()	It is executed before closing the page.

Sections

Event	Description
onMouseIn()	It is executed when the cursor is positioned on the element.
onMouseOut()	It is executed when the cursor moves out of the element.
onClick()	It is executed when clicking on the element.
onDoubleClick()	It is executed when clicking twice on the element.
onInit()	It is executed before the element is loaded.
afterViewInit()	It is executed after viewing the element.
onViewportEnter()	It is executed when the element is visible.
onViewportLeave()	It is executed when the element is no longer visible.

Elements

The events shared by all elements are detailed.

Event	Description
onMouseIn()	It is executed when the cursor is positioned on the element.
onMouseOut()	It is executed when the cursor moves out of the element.
onClick()	It is executed when clicking on the element.
onDoubleClick()	It is executed when clicking twice on the element.
onInit()	It is executed before the element is loaded.

Event	Description
afterViewInit()	It is executed after viewing the element.
onViewportEnter()	It is executed when the element is visible.
onViewportLeave()	It is executed when the element is no longer visible.

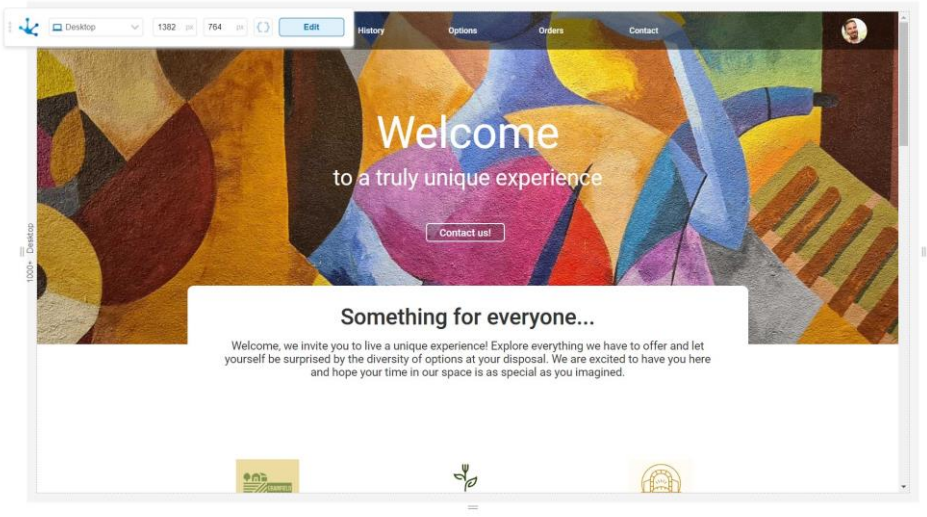
Button

The events shared by all fields are detailed.

Event	Description
onInit()	It is executed before loading the entity.
afterViewInit()	It is executed after rendering the entity.
beforeViewUnload()	It is executed before closing the entity.

3.4.13.1.3.6. Preview

It provides a view of the modeled page, showing how it would look on different devices.



Bar Items

Breakpoints and Devices

The type of device on which the preview of the modeled page will be displayed is selected

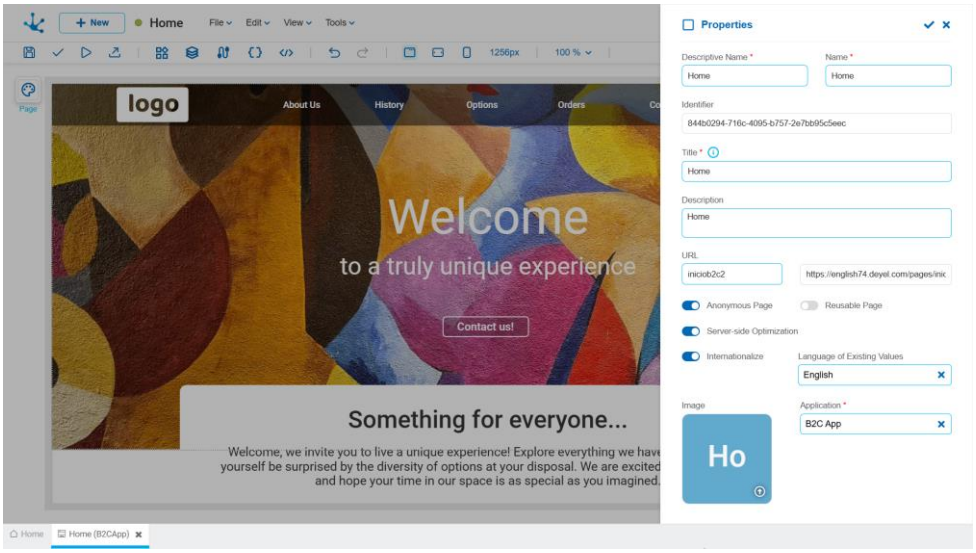
Pixels
It is displayed automatically according to the selected device.

No parameters defined on the page.
It allows to enter a value for each parameter defined on the page.

Edit
It allows to go back to code editing.

3.4.13.2. General Properties

To enter the properties panel of a page, click on the icon from the top toolbar.



Properties

Descriptive Name
Name used by the user to reference the page.

Name
It is used at the modeling level to reference the page.

Identifier
Uniquely identifies the page.

Title
It should summarize the page content and is displayed in the search results.

Description
Text that defines the page describing its functionality.

URL
Page identifier in the URL.

Absolute Path
When defining the identifier, the access path to the page is automatically filled in.

> *Once the page is published, it can be accessed from a web browser with the absolute path. If parameters are defined, their names and values should be added below.*
> *The published page can also be accessed from the modeler by clicking on this property.*

Anonymous Page
If this property is enabled, the user does not need to be logged into the application to access the page.

Reusable Page
If this property is enabled, it indicates that the page can be referenced from other pages or entities.

Server-Side optimization
By enabling this property, page rendering is performed on the server (SSR, - Server Side Rendering), which eliminates unnecessary waiting times and improves loading speed.

Internationalize
Enables the translation of object properties and its elements into different languages. When activated, it enables the option "Internationalization" from the "Tools" drop-down menu and the property Language of Existing Values, which is displayed on the right.

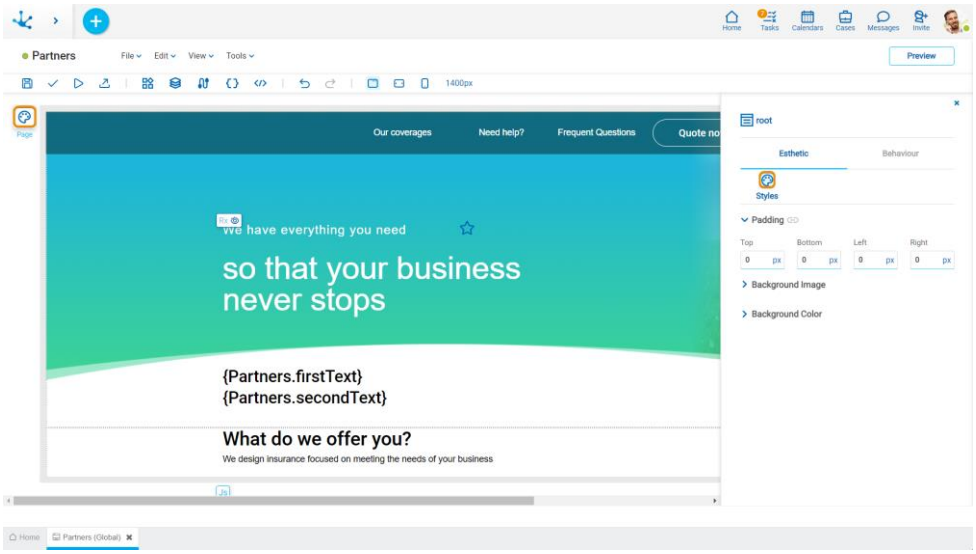
Language of Existing Values
Displayed when the property Internationalize is activated for the first time and indicates the original language. Its selection is required and the available options are: "Spanish (Argentina)", "Spanish (Mexico)", "English" and "Portuguese".

Application
It allows to define the application to which the object belongs.

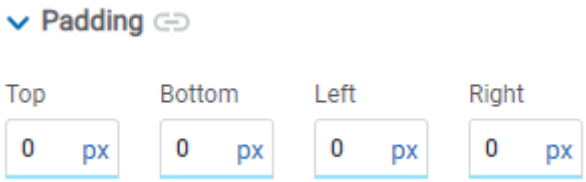
Image
It allows to customize the page by adding an image. This image represents the page in the applications modeler if it is selected as the home page.

3.4.13.3. Style Properties

The style properties panel of a page opens when selecting the icon  of the context menu.



Padding

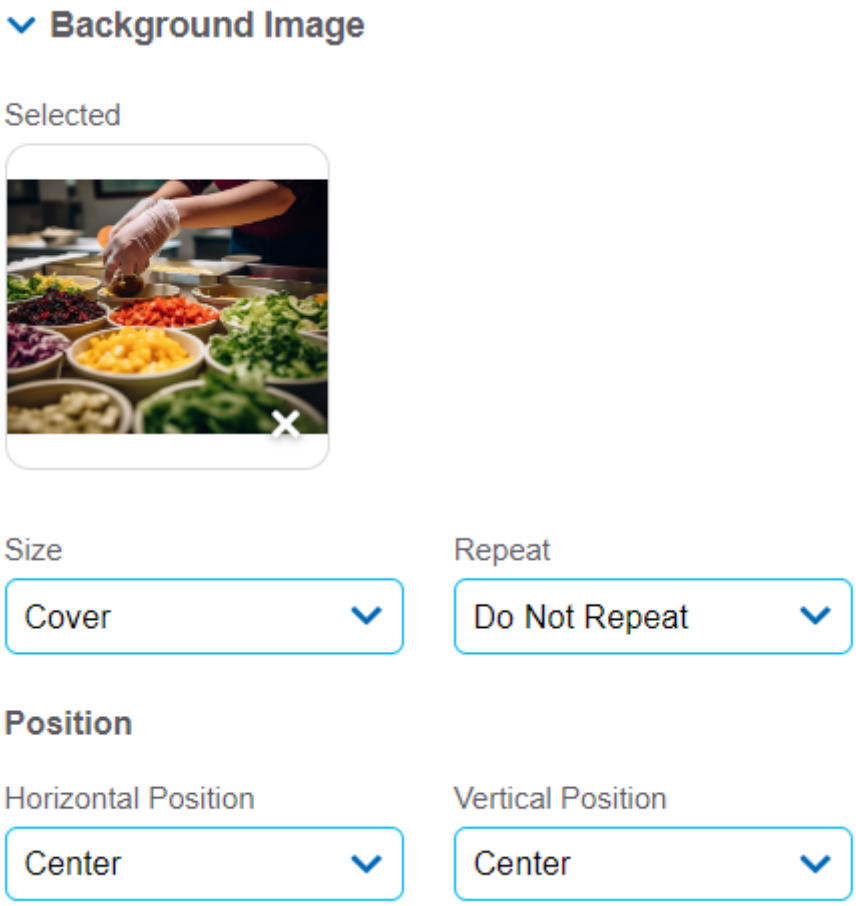


All padding space properties allow to create a space around the element borders (top, bottom, sides) and its content. The padding space configured is inward, while the margin is configured relative to a top element. It can be expressed in pixels (px), percentage (%), viewport width (vw), and viewport height (vh).
o

Allows values entered in one of the paddings to be copied to the other ones automatically.

Allows to indicate different values for each padding.

The element properties are represented by icons on its context menu, where its operations are also available.
PropsEstiloImagenFondo3



It allows to add a background image to the section.

Size

Possible Values

- Cover
- Content
- Auto

Repeat

Possible Values

- Repeat
- Repeat Hor.
- Repeat Vert.
- Spacing
- Rounded
- Do Not Repeat

Horizontal and Vertical Position

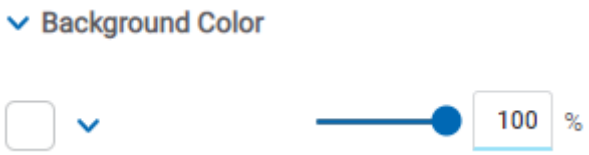
Possible Values

- Center
- Left
- Right

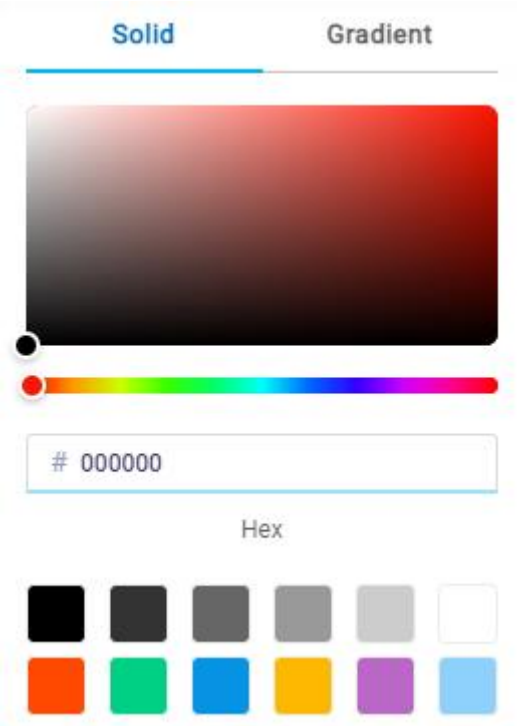
Background Color

It allows to model a background color for the element, it can be solid or gradient and different properties are defined for each one.

Solid



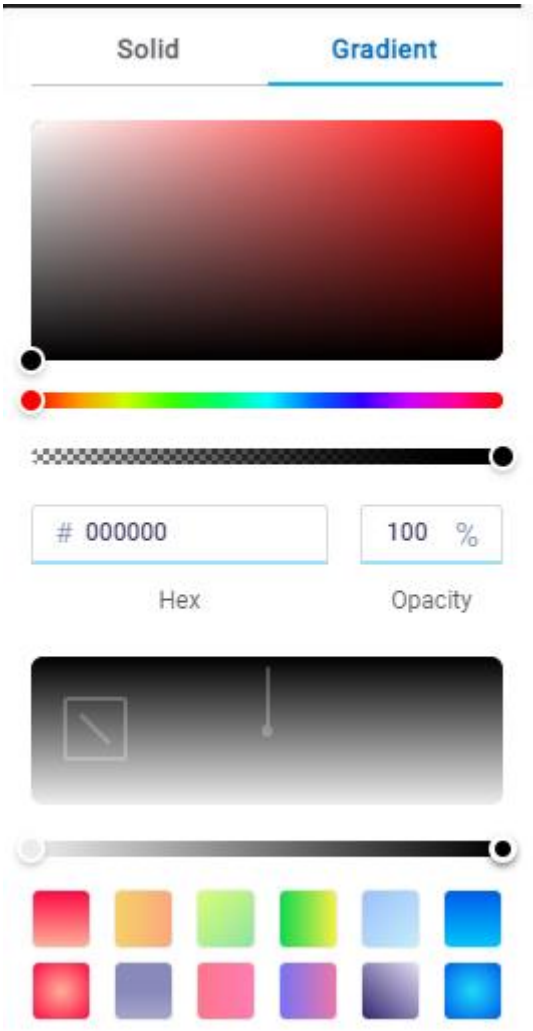
This option enables not only to select the color from a palette but also to define the degree of transparency.



Gradient



This option enables to select not only the color from a palette but also to define its opacity and intensity according to the selected angle.



3.4.13.4. Deyel SDK for Javascript

Using Deyel SDK (Software Development Kit), a set of development tools for pages is incorporated, based on JavaScript, which allows extending the functionality of the pages modeled within **Deyel**. There is available a set of methods that help manage the modeled elements and facilitate the incorporation of behavior.

To access the global methods of the Deyel SDK for JS, the reserved namespace $d should be used.

Example:

```
$d.getVariable("name");
```

Methods of the Pages SDK

The Deyel SDK for JS methods are grouped under the following classification:

- Element
- Repeater
- Widget
- Data Source
- Variable
- Parameter
- User
- Rule Execution
- Embedded Rules
- Redirection
- File Usage
- API
- Root
- Miscellaneous

3.4.13.4.1. Element

Access methods are used to retrieve items or access specific functionalities.

Class methods are specific to each element, facilitating their management and incorporating their own behavior.

Access Method

Name	Description	Result
$d.getElement(idElement)	Searches and gets an element by identifier. Parameter • String **idElement**: Element identifier	PageElement
$d.getButton(idElement)	Searches and gets a button by identifier. Parameter • String **idElement**: Button identifier	Button
$d.getColumnItem(idElement)	Searches and gets a column of an item. Parameter	ColumnItem

Name	Description	Result
	• String **idElement**: Column identifier	
$d.getContainer(idElement)	Searches and gets a container by identifier. Parameter • String **idElement**: Container identifier	container
$d.getEnrichedText(idElement)	Searches and gets a rich text by identifier. Parameter • String **idElement**: Rich text identifier	EnrichedText
$d.getStripItem(idElement)	Searches and gets a strip of the item by identifier. Parameter • String **idElement**: Strip identifier	PageStripItem
$d.getIcon(idElement)	Searches and gets an icon by identifier. Parameter • String **idElement**: Icon identifier	Icon
$d.getIframe(idElement)	Searches and gets an iframe by identifier. Parameter • String **idElement**: Iframe identifier	iframe
$d.getImage(id Element)	Searches and gets an image by identifier. Parameter • String **idElement**: Image identifier	Image
$d.getInput(idElement)	Searches and gets an input by identifier. Parameter • String **idElement**: Input identifier	input
$d.getItem(idElement)	Searches and gets an item by identifier. Parameter • String **idElement**: Item identifier	Item
$d.getLayout(IdElement)	Searches and gets a layout by identifier. Parameter	Layout

Name	Description	Result
	• String **idElement**: Layout identifier	
$d.getMosaicItem(IdElement)	Searches and gets an item mosaic by identifier. Parameter • String **idElement**: Item mosaic identifier	MosaicItem
$d.getRepeater(idElement)	Searches and gets a repeater by identifier. Parameter • String **idElement**: Repeater identifier	Repeater
$d.getRowItem(idElement)	Searches and gets an item row by identifier. Parameter • String **idElement**: Item row identifier	Row
$d.getSection(idElement)	Searches and gets a section by identifier. Parameter • String **idElement**: Section identifier	Section
$d.getText(idElement)	Searches and gets a text by identifier. Parameter • String **idElement**: Text identifier	Text
$d.getWidget(idElement)	Searches and gets a widget by identifier. Parameter • String **idElement**: widget identifier	Widget

PageElement Class Methods

Name	Description	Result
getDOMElement()	Gets the element within the DOM.	DOMElement
getMargin()	Gets the margin value.	String
getMarginBottom()	Gets the bottom margin value.	String

Name	Description	Result
getMarginLeft()	Gets the left margin value.	String
getMarginRight()	Gets the right margin value.	String
getMarginTop()	Gets the top margin value.	String
getPosition()	Gets the value of the element's position.	String
getPositionBottom()	Gets the value of the element's bottom position.	String
getPositionLeft()	Gets the value of the element's left position.	String
getPositionRight()	Gets the value of the element's right position.	String
getPositionTop()	Gets the value of the element's top position.	String
getVisible()	Gets if the element is visible.	Boolean
setMargin(margin)	Parameter • String **margin**: Margin value	PageElement
setMarginBottom(margin)	Parameter • String **margin**: Bottom margin value	PageElement
setMarginLeft(margin)	Parameter • String **margin:** Gets the left margin value	PageElement
setMarginRight(margin)	Parameter • String **margin:** Gets the right margin value	PageElement
setMarginTop(margin)	Parameter • String **margin**: Top margin value	PageElement
setPosition(position)	Parameter • String **position**: Position value	PageElement
setPositionBottom(position)	Parameter • String **position**: Bottom position value	PageElement
setPositionLeft(position)	Parameter • String **position**: Left position value	PageElement
setPositionRight(position)	Parameter • String **position**: Right position value	PageElement

Name	Description	Result
setPositionTop(position)	Parameter • String **position**: Top position value	PageElement
setVisible(visible)	Sets if the page is visible. Parameter • Boolean **visible**: Determines whether it is visible or not	PageElement

3.4.13.4.2. Repeater

Access methods are used to retrieve items or access specific functionalities.

Class methods are specific to each element, facilitating their management and incorporating their own behavior.

Access Method

Name	Description	Result
$d.getRepeater(idElement)	Searches and gets a repeater by identifier. Parameter • String **idElement**: Repeater identifier	Repeater

Class Methods

Name	Description	Result
getColumns()	Gets the number of columns of the repeater.	Number
getData()	Gets repeater data.	Object[]
getDataSource()	Gets the repeater datasource.	Datasource
getPage()	Gets the number of pages (pagination) of the repeater.	Number
getPerPage()	Gets the total number of items per page	Number

Name	Description	Result
	(pagination) of the repeater.	
getRows()	Gets the number of rows of the repeater.	Number
getTotalPages()	Gets the total number of pages (pagination) of the repeater.	Number
getTotalItems()	Gets the total number of items of the repeater.	Number
refreshRepeater()	Refreshes the repeater, that is, it requests data from the configured datasource again.	Repeater
setColumns(columns)	Sets a number of columns to the repeater. Parameter • Number **columns**: Number of columns	Repeater
setDataSource(idDataSource)	Sets a datasource to the repeater. Parameter • String **idDataSource**: Datasource identifier	Repeater
setPage(page)	Sets the page (pagination) of the repeater. Parameter • Number **page**: Page number	Repeater
setRows(rows)	Sets a number of rows to the repeater. Parameter • Number **rows**: Number of columns	Repeater

3.4.13.4.3. Widget

Access methods are used to retrieve items or access specific functionalities.

Class methods are specific to each element, facilitating their management and incorporating their own behavior.

Access Method

Name	Description	Result
$d.getWidget(idElement)	Searches and gets a widget by identifier. Parameter • String **idElement**: Widget identifier	Widget

Class Methods

Name	Description	Result
getWidgetType()	Gets the type of widget	String
getMockupJSON(type)	Gets an example JSON, which contains the internal format of widgets with their respective type. Parameter • String **type**: Widget type	Object
getValue()	Gets the JSON that contains the internal format of the widgets.	Object
setJSONValue(widgetJSON, index)	Sets the JSON that contains the internal format of a widget and instantiates it. Parameters • Object **widgetJSON**: JSON of the widget • Number **index**: Instance number	Widget
setValue()	Sets the id of a pre-existing widget and instantiates it. Parameters • Number **idWidget**: widget id • Number **index**: Instance number	Widget

3.4.13.4.4. Data Source

Access methods are used to retrieve items or access specific functionalities.

Class methods are specific to each element, facilitating their management and incorporating their own behavior.

Access Method

Name	Description	Result
setDataSource(idDataSource)	Gets a DataSource object, with the ID sent by parameter. Parameter • String **idDataSource**: Datasource identifier.	Datasource

Class Methods

Name	Description	Result
afterExecute(callback)	It is used to execute a function after invoking the Datasource. Parameter • (data: Object?) => **Object callback**: Function to be invoked	-
createDataSource(idDatasource, executeCallback)	It allows to create a datasource with a specific identifier. Parameters • String **idDataSource**: Datasource identifier • Callback **executeCallback**: Function to be executed upon completion of the operation Example: `$d.createDataSource(idDataSource: string, execute: (nuPage: number, qtRows: number) => { instances: Record<string, any>; paging?: { pages: number, "page-number": number; }; } | Record<string, any> | Record<...>[]):`	Datasource

Name	Description	Result
execute(callback, nuPage, qtRows)	It executes a datasource and upon receiving a response from the server it calls the callback function. Parameters • Function **callback**: Function to be executed upon completion of the operation • Number **nuPage** (optional): indicates the page to search. If not defined, it is the home page • Number **qtRows** (optional): Number of records to read, if not specified, will be 20 Example: `$d.getDataSource("business").afterExecute((data) => { isOwner = data.instances[0]?.owner activeButton("profileButton"); if (isOwner) { hideEditOption(true); } else { hideEditOption(false); } });`	-
getData()	Gets the data from the datasource.	Object
getFilters()	Gets the filters from the datasource. Gets an array of objects with code, operation, type and value.	Array<{ code: string; type: "LITERAL" | "VARIABLE" | "PARAMETER"; operation: string[]; value: FilterValue; }>
getName()	Gets the name of the datasource.	String
getOrder()	Gets the object that represents the order modeled in the datasource.	{ code: string; order: "ASC" | "DESC";

Name	Description	Result
		] | null
getSelection()	Gets the selected form fields or rule output parameters.	String[]
getSource()	Gets the form identifier or rule name.	String
getType()	Gets the type. It can be rule or form.	String
setFilter(code, operation, type, value)	Sets a filter to the datasource. Parameters • String **code**: Field identifier or rule or form parameter. • String | **String[]** | null operation (optional): Comparison operator (Logical operators). If null, the filter is cleared. • "LITERAL" | "VARIABLE" | "PARAMETER" **type** (Optional): Type of Operation. • **Any value** (Optional): Value to set Example: `$d.getDataSource("SuccessfulPaymentInformation").setFilter("relatedSale", "eq", "LITERAL", $d.getDataSource("DigitalSale").getData()[0].procedureNumber);`	-
setOrder(code, order)	Sets an order to the datasource. Parameters • String **code**: Field identifier or rule or form parameter. • "ASC" | "DESC" **order**: Indicates the type of order with which the operation will be carried out	-

3.4.13.4.5. Variable

Access methods are used to retrieve items or access specific functionalities.

Class methods are specific to each element, facilitating their management and incorporating their own behavior.

Access Method

Name	Description	Result
$d.getVariable(idVariable)	Searches the interface for a variable by identifier. Parameter • String **idVariable**: Variable identifier	Variable

Class Methods

Name	Description	Result
getValue(description)	Returns the value of the variable. If the variable has no value, it returns its initial value. Parameter • Boolean **description** (Optional): Returns if the variable description is taken into account Example: `$d.getVariable("variable name").getValue()`	String | Number | Boolean
setValue(value)	Sets the value of the variable. Parameter • String | Number | Boolean **value**: Value to assign to the variable Example: `$d.getVariable("variable name").setValue("Argentina")`	-

3.4.13.4.6. Parameter

Access methods are used to retrieve items or access specific functionalities.

Class methods are specific to each element, facilitating their management and incorporating their own behavior.

Access Method

Name	Description	Result
$d.getParameter(idParameter)	Searches the interface for a parameter by identifier. Parameter • String **idParameter**: Parameter identifier	Parameter

Class Methods

Name	Description	Result
getValue()	Gets the parameter value.	String | Number | Boolean

3.4.13.4.7. User

Access methods are used to retrieve items or access specific functionalities.

Class methods are specific to each element, facilitating their management and incorporating their own behavior.

Access Method

Name	Description	Result
$d.getCurrentUser()	Returns the current user.	User

Class Methods

Name	Description	Result
getLanguage()	Returns the configured language of the user.	“es_AR” | “en_US” | “pt_BR”

Name	Description	Result
getLastName()	Returns the user last name.	String
getName()	Returns the username.	String
getUsername()	Returns the code of the logged in user.	String

3.4.13.4.8. Rule Execution

Access methods are used to retrieve items or access specific functionalities.

Access Method

Name	Description	Result
$d.executeRule(name, version, params, callback)	It allows to execute an advanced Rest API-type rule. Parameters • String **name**: Rule name • Number **version**: Rule version • Object **params**: Object where the rule parameters are sent • Callback **callback**: Function that is executed after performing the operation Example: `$d.executeRule("generateResponseAboutDocumentation", 1, { 'pQuery': query, 'pTemplate': template, 'user': user, 'pEnvironment': environment, 'pResponse': answer, }, (response, error) => {});`	--

3.4.13.4.9. Embedded Rules

Access methods are used to retrieve items or access specific functionalities.

Access Method

Name	Description	Result
$d.validateElements(elements ,onSuccess, onFailure)	Executes validation and requirement rules for the elements passed as parameter. Parameters • String [] **elements**: Elements identifiers • Callback **onSuccess**: Function that is executed if all validations were successful • Callback **onFailure**: Function that is executed if any of the validations failed	

3.4.13.4.10. Redirection

Access methods are used to retrieve items or access specific functionalities.

Access Method

Name	Description	Result
$d.redirectToCalendar(target, params)	Redirection to the user's calendar. Parameters • String **target**: Navigation Url • Object **params** (optional): Configuration object	-
$d.redirectToCase(idCase, target, params)	Redirection to a process case. Parameters • String **idCase**: Case identifier • String **target**: Navigation url • Object **params** (optional): Parameter configuration object as modeled. It is used to send the parameters to the destination, in the key format: value Example: `params?: { [key: string]: string; }`	-
$d.redirectToCaseExecution(i	Redirection to a process activity.	-

Name	Description	Result
dCase, target, params)	Parameters • String **idCase**: Case identifier • String **target**: Navigation url • Object **params** (optional): Parameter configuration object as modeled. It is used to send the parameters to the destination, in the key format: value Example: `params?: { [key: string]: string; }`	
$d.redirectToCaseStart(idCase, target, params)	Redirection to the process start. Parameters • String **idCase**: Case identifier • String **target**: Navigation url • Object **params** (optional): Parameter configuration object as modeled. It is used to send the parameters to the destination, in the key format: value Example: `params?: { [key: string]: string; }`	-
$d.redirectToDashboard(target, params)	Redirection to the user's dashboard. Parameters • String **target**: Navigation url • Object **params** (optional): Parameter configuration object as modeled. It is used to send the parameters to the destination, in the key format: value Example: `params?: { [key: string]: string; }`	-
$d.redirectToError(message)	Redirection to the error page modeled in the application. Parameter • String **message**: Content of the message to be displayed on the page	-
$d.redirectToFormCreate(idForm, target, params)	Redirection to a form creation.	-

Name	Description	Result
	Parameters • String **idForm**: Form identifier • String **target**: Destination type • Object **params** (optional): Parameter configuration object as modeled. It is used to send the parameters to the destination, in the key format: value Example: `params?: { [key: string]: string; }`	
$d.redirectToFormUpdate(idForm, idKey, idInstance, target, params)	Redirection to a form update. Parameters • String **idForm**: Form identifier • String **idKey**: Document key • String **idInstance**: Identifier of the instance • String **target**: Destination type • Object **params** (optional): Parameter configuration object as modeled. It is used to send the parameters to the destination, in the key format: value Example: `params?: { [key: string]: string; }.`	-
$d.redirectToFormDelete(idForm, idKey, idInstance, target, params)	Redirection to a form deletion. Parameters • String **idForm**: Form identifier • String **idKey**: Document key • String **idInstance**: Identifier of the instance • String **target**: Destination type • Object **params** (optional): Configuration object	-
$d.redirectToFormShow(idForm, idKey, idInstance, target, params)	Redirection to a form show. Parameters • String **idForm**: Form identifier • String **idKey**: Document key • String **idInstance**: Identifier of the instance • String **target**: Destination type	-

Name	Description	Result
	• Object **params** (optional): Parameter configuration object as modeled. It is used to send the parameters to the destination, in the key format: value Example: `params?: { [key: string]: string; }`	
$d.redirectToFormSearch(idForm, target, params)	Redirection to a form search. Parameters • String **idForm**: Form identifier • String **target**: Destination type • Object **params** (optional): Parameter configuration object as modeled. It is used to send the parameters to the destination, in the key format: value Example: `params?: { [key: string]: string; }`	-
$d.redirectToPage(idPage, target, params)	Redirection to a page. Parameters • String **idPage**: Destination page identifier • String **target**: Destination type • Object **params** (optional): Parameter configuration object as modeled. It is used to send the parameters to the destination, in the key format: value Example: `params?: { [key: string]: string; }`	-
$d.redirectToTasks(idTasks, target, params)	Redirection to the user's tasks. Parameters • String **idTasks**: Task identifier • String **target**: Destination type • Object **params** (optional): Parameter configuration object as modeled. It is used to send the parameters to the destination, in the key format: value Example: `params?: { [key: string]: string;`	-

Name	Description	Result
	} 	
$d.redirectToUrl(url, target, params)	Redirection to a specific URL. Parameters • String **url:** Navigation url • String **target**: Destination type • Object params (optional): Parameter configuration object as modeled. It is used to send the parameters to the destination, in the key format: value Example: `params?: { [key: string]: string; }`	-

3.4.13.4.11. File Usage

Access methods are used to retrieve items or access specific functionalities.

Access Method

Name	Description	Result
$d.selectFile(callback, accept)	It allows selecting files. When executed, the file explorer opens so the user can choose one or more files. Parameters • Function **callback**: Function to be executed upon completion of the operation • String **accept**: Type of file that will be allowed to be selected. If null or undefined is sent, it is assumed that all file types are accepted Example: `$d.selectFile((event, files, input) => { console.log(files[0]); }, 'image/png')`	-
$d.uploadFile(file, callback,	Allows uploading a file to **Deyel**.	

Name	Description	Result
objectType)	Parameters • File **file**: Javascript Object that represents a file • Function **callback**: Function to be executed upon completion of the operation • String **objectType**: Destination type of object in **Deyel**. Example, "FORM" | "RULE". Example: `$d.uploadFile(files[0], (result, error) => { console.log(result); }, "FORM");`	

3.4.13.4.12. API

Access methods are used to retrieve items or access specific functionalities.

Class methods are specific to each element, facilitating their management and incorporating their own behavior.

Access Method

Name	Description	Result
$d.getAPI()	Gets an API object	API

Class Methods

Name	Description	Result
delete(url)	Makes a DELETE request to the url, using the online user's credentials (token). Parameter • String **url**: Rest API path Example:	Promise<Object>

Name	Description	Result
	`$d.delete("https://hostname.com/API/id")`	
get(url)	Makes a GET request to the url, using the online user's credentials (token). Parameter • String **url**: Rest API path Example: `$d.get("https://hostname.com/API")`	Promise<Object>

patch(url, data)	Makes a PATCH request to the url, using the credentials (token) of the online user and sending the object received in the "data" parameter in JSON format as a payload.
	
	Parameters
	• String **url**: Rest API path
	• Object **data**: Object to send in the body of the request
	
	Example:
	
	`$d.patch("https://hostname.com/API/id", {`
	`  commentId: 1,`
	`  commentContent: "Hello!"`
	`})`
		Promise<Object>

post(url, data)	Makes a POST request to the url, using the credentials (token) of the online user and sending the object received in the "data" parameter in JSON format as a payload. Parameters • String **url**: Rest API path • Object **data**: Object to send in the body of the request Example:	Promise<Object>

Name	Description	Result

	`$d.post("https://hostname.com/API/id", {`
	`  commentId: 1,`
	`  commentContent: "Hello!"`
	`})`	

put(url, data)	Makes a PUT request to the url, using the credentials (token) of the online user and sending the object received in the "data" parameter in JSON format as a payload.
	
	Parameters
	• String **url** : Rest API path
	• Object **data**: Object to send in the body of the request
	
	Example:
	
	`$d.put("https://hostname.com/API/id", {`
	`  commentId: 1,`
	`  commentContent: "Hello!"`
	`})`
		Promise<Object>

3.4.13.4.13. Root

Represents the main styles and configurations of the page.

Access methods are used to retrieve items or access specific functionalities.

Class methods are specific to each element, facilitating their management and incorporating their own behavior.

Access Method

Name	Description	Result
$d.getRoot()	Gets the main configuration of the page.	Root

Class Methods

Name	Description	Result
getBackgroundColor()	Gets the color of the page background.	String
getBackgroundPosition()	Gets the position of the page background.	String
getBackgroundPositionX()	Gets the position of the background on the horizontal axis.	String
getBackgroundPositionY()	Gets the position of the background on the vertical axis.	String
getBackgroundRepeat()	Gets the background repeat configuration.	String
getBackgroundSize()	Gets the background size.	String
getDOMElement()	Gets the Javascript node	Javascript DOM
getVisible()	Gets if the element is visible.	Boolean
getPadding()	Gets the configured padding space.	String
getPaddingBottom()	Gets the configured bottom padding space.	String
getPaddingLeft()	Gets the configured left padding space.	String
getPaddingRight()	Gets the configured right padding space.	String
getPaddingTopt()	Gets the configured top padding space.	String
setBackgroundColor(color, state)	Sets the background color. Parameters • String **color**: Color in hexadecimal • String **state**: Color state	Root
setBackgroundPosi-tion(position, state)	Sets the position of the page background. Parameters • Object **position**: Object where the X and Y coordinates are configured • String **state**: Position state	Root

Name	Description	Result
setBackgroundPositionX(position, state)	Sets the background position on the horizontal axis of the page. Parameters • Object **position**: Object where the X and Y coordinates are configured • String **state**: Position state	Root
setBackgroundPositionY(position, state)	Sets the background position on the vertical axis of the page. Parameters • Object **position**: Object where the X and Y coordinates are configured • String **state**: Position state	Root
setBackgroundRepeat(repeat, state)	Sets the page background repeat configuration. Parameters • String **repeat**: Repetition type • String **state**: Repetition state	Root
setBackgroundSize(size, state)	Sets the size of the page background. Parameters • String **size**: Background size • String **state**: Size status	Root
setPadding(padding, state)	Sets the page padding space. Parameters • String **padding**: Padding space • String **state**: Padding status	Root
setPaddingBottom(paddingBottom, state)	Sets bottom page padding space. Parameters • String **paddingBottom**: Padding space • String **state**: Padding status	Root
setPaddingLeft(paddingLeft, state)	Sets left page padding space. Parameters • String **paddingLeft**: Padding space • String **state**: Padding status	Root

Name	Description	Result
setPaddin-gRight(paddingRight, state)	Sets right page padding space. Parameters • String **paddingRight**: Padding space • String **state**: Padding status	Root
setPaddingTop(paddingTop, state)	Sets top page padding space. Parameters • String **paddingTop**: Padding space • String **state**: Padding status	Root
setVisible(visible)	Sets if the page is visible. Parameter • Boolean **visible**: Determines whether it is visible or not	Root

3.4.13.4.14. Authentication

Access methods are used to retrieve items or access specific functionalities.

Access Method

Name	Description	Result
$d.createGoogleLoginButton (config, container, onLogin, onNewUSer)	Creates the Google login button in a specific container. After successfully executing the login, it redirects to the home page configu-red in the application where it is running. Parameters • Object **config**: Object specified by Google to configure the button • DomElement string **container**: Container where the Google button will be displayed • Callback **onLogin**: Function that is execu-ted when the user logs in • Callback **noNewUSer**: Function that is exe-cuted when a new user is registered	-
$d.createGoogleLoginButton WithoutRedirection(config,	Creates the Google login button in a specific container. After successfully executing the	-

Name	Description	Result
container, onLogin, on-NewUSer)	login, it does not perform any redirection. Parameters • Object **config**: Object specified by Google to configure the button • DOMElement string **container**: Container where the Google button will be displayed • Callback **onLogin**: Function that is executed when the user logs in • Callback **noNewUSer**: Function that is executed when a new user is registered	
$d.logout()	Logs out of the current user.	-

3.4.13.4.15. Miscellaneous

Access methods are used to retrieve items or access specific functionalities.

Access Method

Name	Description	Result
$d.createSelect(configuration, container)	Creates a selection list, within a container, with the values specified in the configuration sent by parameter. Parameters • Object **configuration**: List configuration object • DOMElement string **container**: Container where the list will be displayed Example `let itemsList = [ { value: 'Argentina' id: '1' }, { value: 'Chile:', id: '2' }, {`	UpdateSelectFunctions: Object with functions

Name	Description	Result

	<pre>value: 'Brazil',
	id: '3'
	},
	{
	value: 'Paraguay',
	id: '4'
	}
	];
	$d.createSelect({
	idKey: "id",
	valueKey: "value",
	list: itemsList,
	test_id: "types",
	onHandleSelectItem: handleChangeTypeFilter,
	placeholder: 'Select Item'
	}, 'container1');
	
	function handleChangeTypeFilter(value) {
	console.log("value ",value);}</pre>
	Notes
	
	• The format of the list must be as specified.
	• The value of the **idKey** attribute is the identifier of each item.
	• The value of the **valueKey** attribute is the attribute on which to filter.
	• In the **onHandleSelectItem** attribute, the function to be executed is sent when an item is selected. In the example, the function handleChangeTypeFilter was passed, which prints the value on the screen.	

$d.createSelectMultiple(configuration, container)	Creates a multiple selection list, within a container, with the values specified in the configuration sent by parameter.
	
	Parameters
	• Object **configuration**: List configuration object
	• DOMElement string **container**: Container where the list will be displayed
	
	Example
	
	<pre>let itemsList = [
	{
	value: 'CRM',</pre>
		UpdateSelectMultipleFunctions: Object with functions

Name	Description	Result

	id: '1'
	}
	];
	
	function showSelect() {
	componentSelect =
	$d.createSelectMultiple({
	idKey: "id",
	valueKey: "value",
	list: itemsList,
	selects: selectedApps,
	placeholder: " Select
	application...",
	functions: {
	onChange: handleChan-
	geTypeFilter,
	},
	}, "selectApps");
	
	function handleChangeTypeFil-
	ter(value) {
	console.log("value
	",value);}
	}
	}	

$d.createTextEditor(configura tion, container)	Creates a rich text editor within a specified container. Parameters • Object **configuration**: List configuration object • DOMElement string **container**: Container where the list will be displayed Example let textEditorProps = { initialValue: "text", onChange: handleChangeTex- tEditor, options: ["blockType", "fontFamily", "fontSize", "textAlign", "colorPicker", "inline", "history",], inlineOptions: ["bold", "ita- lic", "underline", "striketh- rough"],	-

Name	Description	Result
	`blockTypeOptions: ["Normal", "H1", "H2", "H3", "H4", "H5", "H6"], fontFamilyOptions: ["Arial", "Brush Script MT, cursive", "Calibri", "Courier New", "Georgia", "Lucida Sans Typewriter, monospace", "Roboto Medium", "Roboto", "Tahoma", "Times New Roman", "Trebuchet MS",], }; $d.createTextEditor(textEditorProps, "container1");`	
$d.getAttributeDataType(idElement, attributeName)	Gets the attribute data sent by parameter of a given element. Parameters • String **idElement**: Element identifier • String **attributeName**: Name of the attribute to show Possible Values: "value" | "editable" | "visible"	String
$d.getCurrentBreakpoint()	Gets the current breakpoint.	String "DESKTOP" | "TABLET" | "MOBILE"

3.4.13.4.16. Examples of SDK Use

In this SDK use example, a rule is executed to add a new comment in a restaurant app.

It consists of associating an onClick function with a comment button, and its objective is to execute a rule to add a comment about the application.

The rule parameters are obtained from an input element entered by the logged-in user and from a page parameter.
After executing the rule, the callback method is triggered where the result is processed. If there is no error, it clears the value of the input element and updates the repeater iterating over the comments.

```
function inputComment_onClick(event) {
```

```
        const comment = $d.getInput("newComment").getValue();
        const user = $d.getCurrentUser().getUsername();
        const idMenu = $d.getParameter("idMenu").getValue();
        $d.executeRule("menuCommentSDK", 1, {
            commentDescription: comment,
            nameUser: user,
            menu: idMenu,
        }, callback);

    function callback(result, error) {
            $d.getInput("newComment").setValue("");
            $d.getRepeater("repeater1").refreshRepeater();
        }
    }
```

3.4.13.5. Server Side Rendering

Server Side Rendering (SSR) is a technique that allows pages to be processed and rendered directly on the server before being sent to the browser. This ensures that the content is fully ready to be displayed from the first interaction, eliminating unnecessary waiting times and improving the user experience.

To enable rendering on the server, the Server-Side optimization property must be activated.

Benefits

One of the main advantages of SSR is the immediate display of content, avoiding blank screens that often occur while scripts are loading in the browser. The user can interact with the page quickly, resulting in a smoother and more efficient experience.

Additionally, SSR ensures that the entire page, including the dynamic content, is available from the first request. This prevents the user from seeing placeholders or incomplete content while the data is being processed.

On mobile devices or environments with slow connections, SSR significantly reduces the browser's workload. By delivering the page already rendered from the server, the impact of hardware limitations and unstable networks is minimized. This offers an optimized experience even in resource-constrained contexts.

Another notable advantage of SSR is its positive impact on search engine optimization (SEO - Search Engine Optimization). By delivering fully rendered pages, search engines can index the content more efficiently, improving the visibility and ranking of web applications in search results.

3.4.13.6. Best Practices

The best practices for the pages modeling in **Deyel** consist of a set of recommendations, to be met whenever possible.

They allow to build clearer and more understandable page models, standardizing the use of the graphic elements that compose them.

On the one hand, good practices recommended for the construction of pages at a general level are described and, on the other hand, some specific ones associated with the page modeler of **Deyel**.

- General Considerations
- Modeling

3.4.13.6.1. General Considerations

Structure and patterns specific to each device

For each device, the elements have specific characteristics such as headers, footers, sidebars, menus, modals and buttons, among others.

- These features or patterns that solve common interaction problems in interfaces must be reusable to ensure consistency and an effective user experience. In **Deyel** is solved with the concept of reusable pages.

Visual consistency

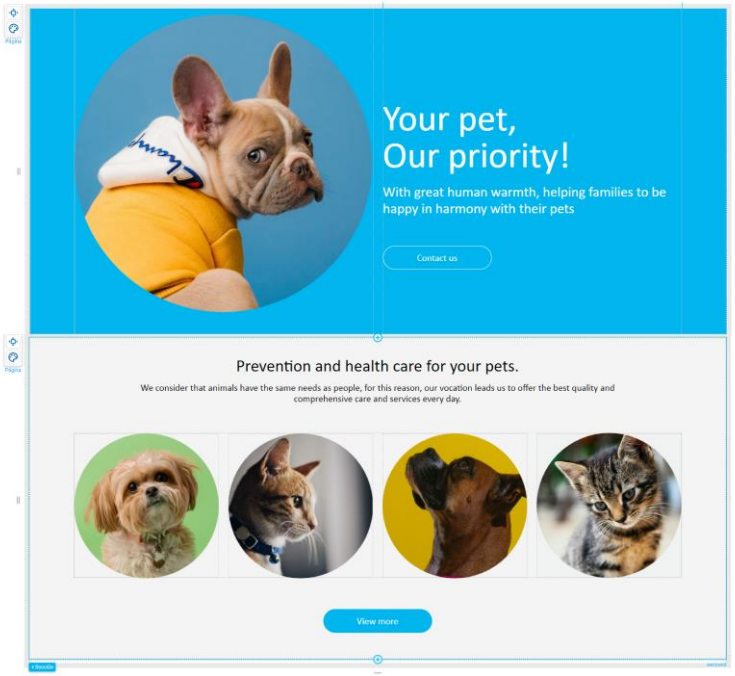
- The user interface must have visual consistency, to do so, graphic backgrounds must be reviewed, such as the company's site, in order to maintain color palettes, fonts, sizes and image treatment.

3.4.13.6.2. Modeling

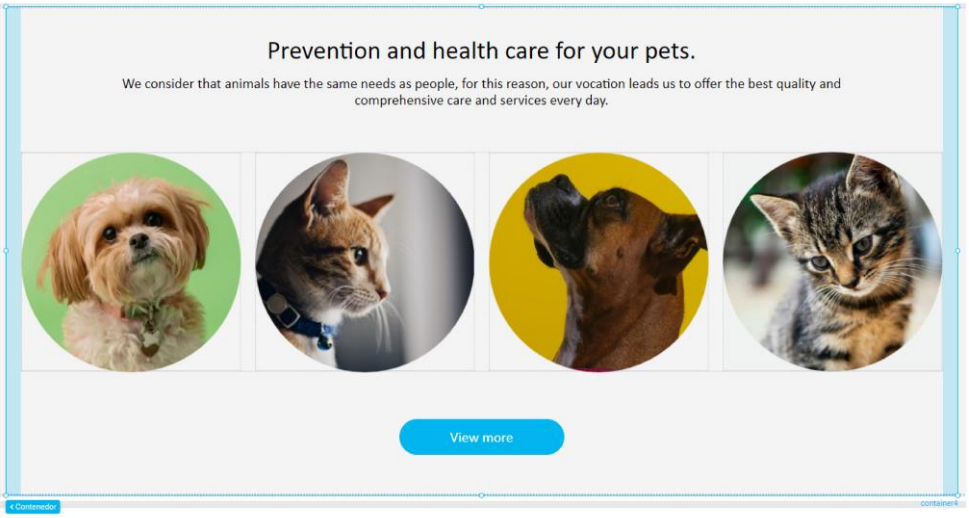
Below are some good practices that should be considered when modeling pages, aimed at information blocks, margins, breakpoints, images, fonts, buttons, padding spaces and docking, among others.

Information blocks in top elements

- In the user Interface, group and generate blocks of information within sections for greater control of the elements that make up each section and thus facilitate the necessary adjustments for each breakpoint.



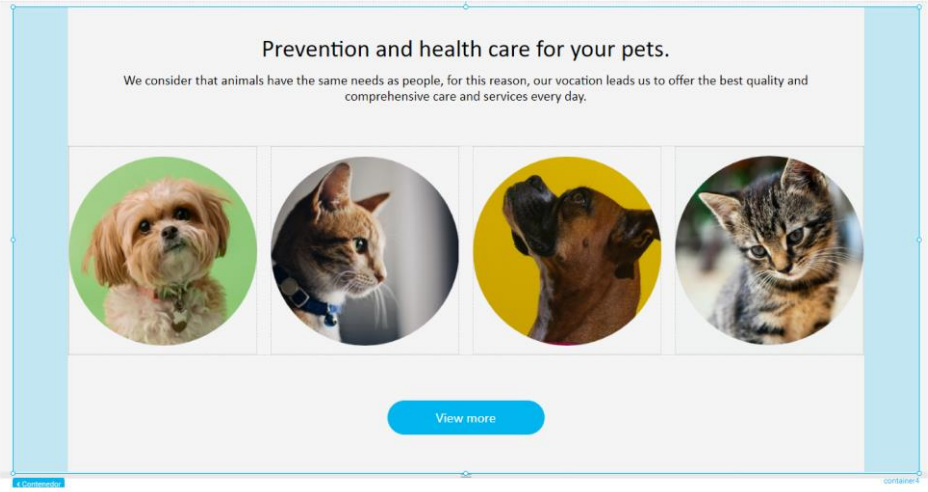
- The same applies to blocks of information inside containers.



Set width to container

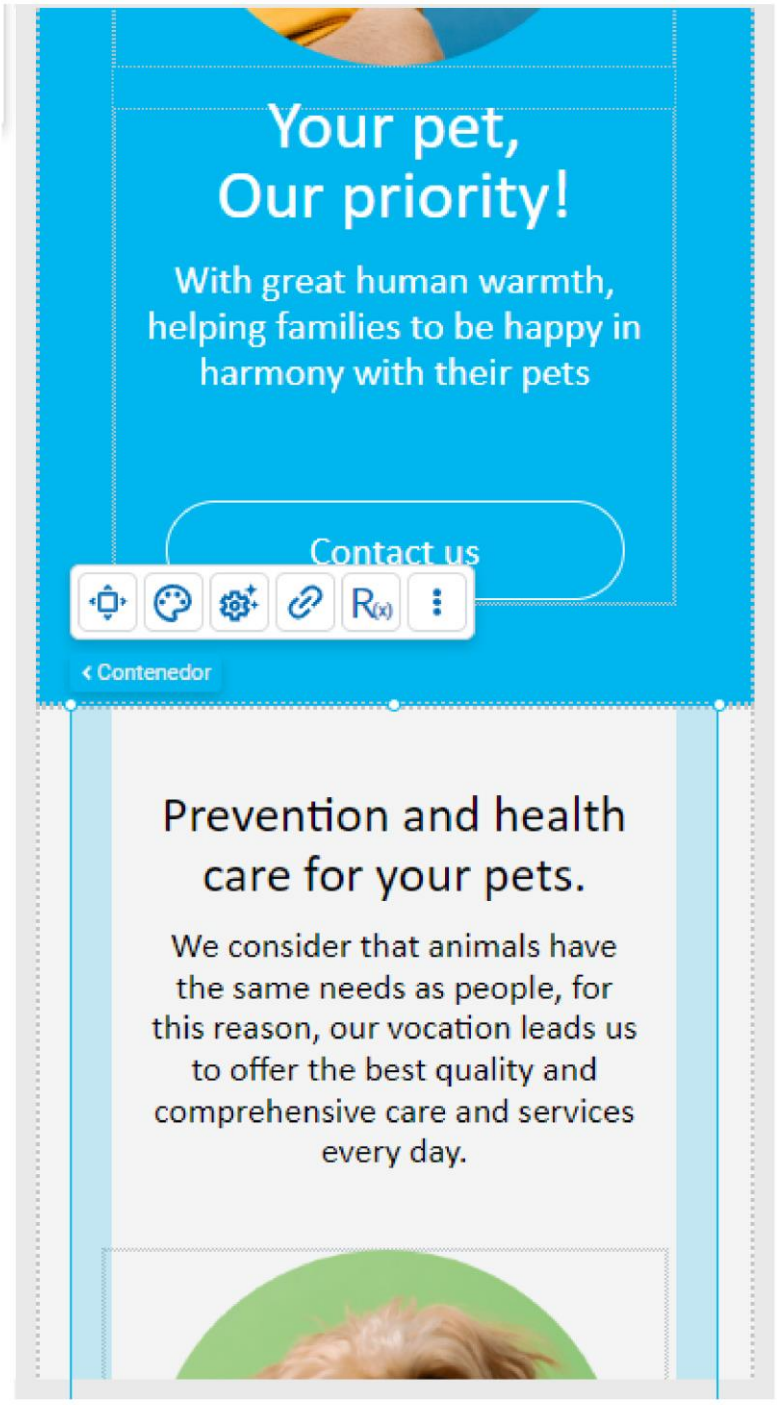
- Set a specific width to a container allows working with more facilities the elements that it contains.

In this example the container has specific width and all the elements it contains have their width at 100%. This makes it easier to adjust the display of elements at the remaining breakpoints.



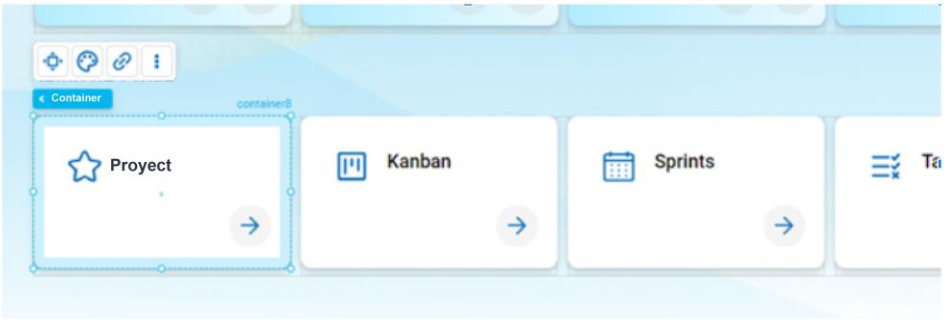
Consistent margins

- Working with consistent margins between sections or groups of information generates visual order and the necessary separation of content.



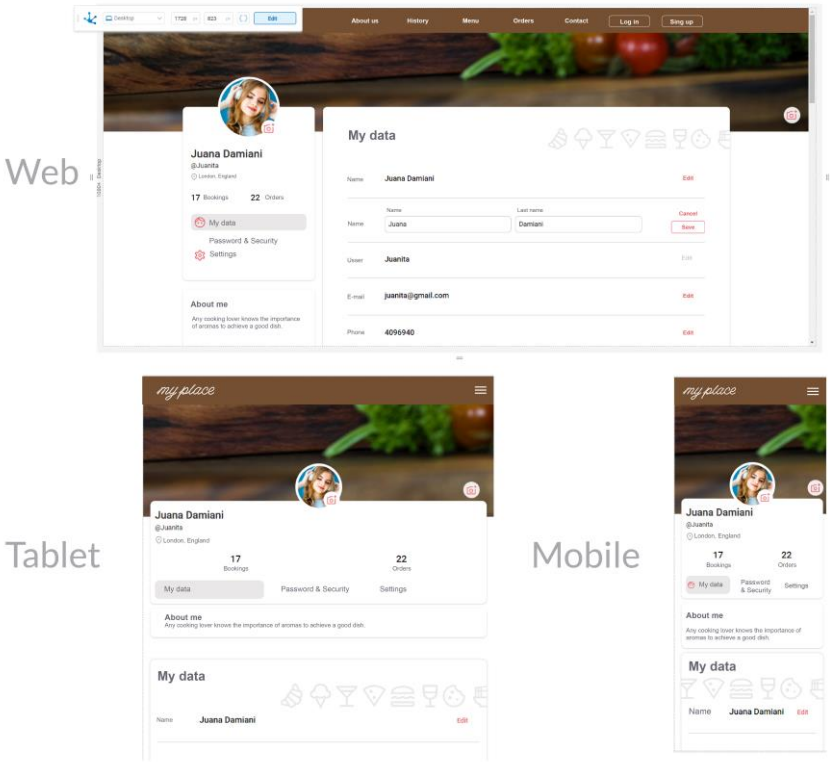
- Likewise, within sections and containers, both margins and padding spaces should be consistent.
- Equality of margins must be maintained on similar elements when they are repeated in the modeling and are not within a repeater.

- For example, to model exactly the same cards, a good tip is to adjust the first card and once you achieve the desired design, copy and paste to finally edit each one in particular.



Control breakpoints gradually by section

- When modeling, as the content grows, it is advisable to work on the breakpoints gradually. This makes it easier to control the correct position of the elements and adjust the breakpoints as the modeling progresses.

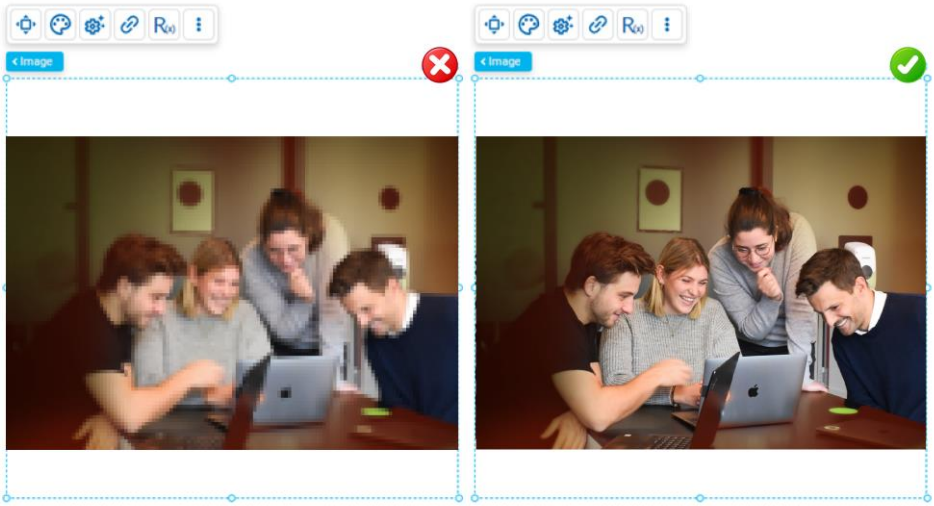


Images: sizes and resolutions

- When modeling an image element, the size at which it will be used on the page must be equal to or smaller than the original.

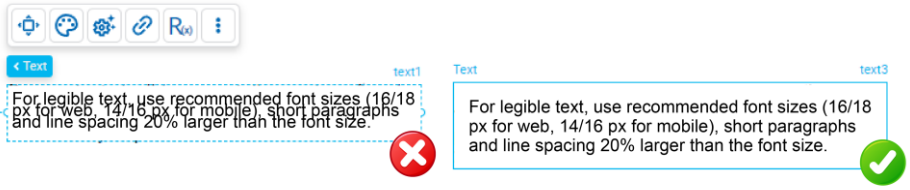
For example, if the size of a modeled image is small and it is used as the background of a section on a desktop computer, it will not look sharp and the result will not be as expected.

- Ideally, images should be in jpg or png format and use a resolution of 72dpi, to optimize the weight of the page.
- For icons, the best is the svg format because it does not lose definition, for photos the best is jpg because the files are lighter and if the user wants to use transparency, the use of png is recommended.
- Images must be optimized for use on the web, the most common formats are JPG, PNG, GIF and their resolution must be 72 dpi.
- The JPG format is ideal for photographs, PNG for images with transparency, GIF for short animations. On the other hand, for icons or logos, the use of SVG is recommended since they can be scaled without loss of definition.
- There are new image formats such as WebP and AVIF that offer greater compression and quality images with lighter files. While most browsers support them, it is advisable to check compatibility before use.



Source

- When modeling a text element, it must be able to be read clearly. To do this, it is advisable not to use sizes lower than those recommended for both a web interface (for text body 16/18 px) and for a mobile interface (for text body 14/16 px).
- Blocks of text should be short and concise since long paragraphs are difficult to read on the screen. If they are too wide they make reading difficult.
- The line spacing must be adequate, the recommended proportion is 20% larger than the font size.



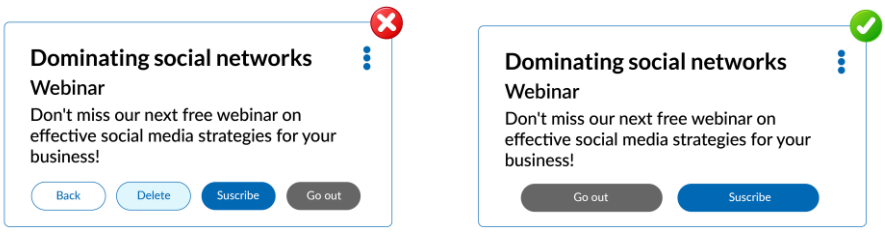
Contrast

- We must always ensure a good contrast between the text and the background to ensure readability.



Buttons

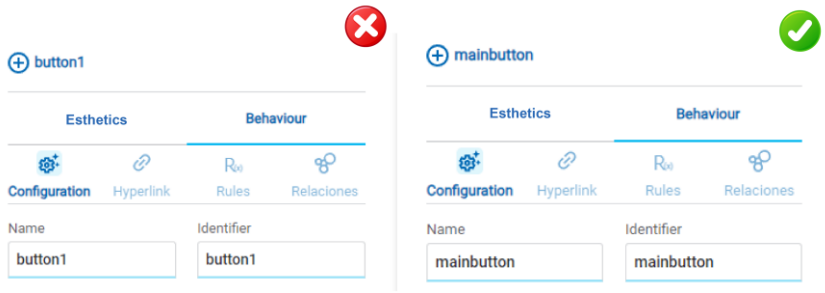
- The user should not add too many buttons per groups of information and they should differentiate which one corresponds to the main call to action.
- Try to generate the fewest steps and clicks in the interaction with the user.



Element names and identifiers

- The identifiers and names of the elements of a page must be descriptive, allowing easy queries when using the layers menu and in embedded rules modeling. It can also be useful when modeling behavior through the use of scripting.

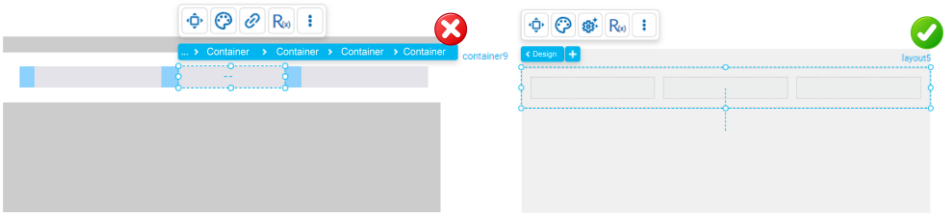
- The names and identifiers of elements on a page must be descriptive. Names are used to reference elements within the modeling while identifiers are used when referencing fields in the Deyel SDK for JavaScript. When using the layers facility, items can be searched by both name and identifier.



Simple structure

- Avoid excessive nesting of elements on a page.

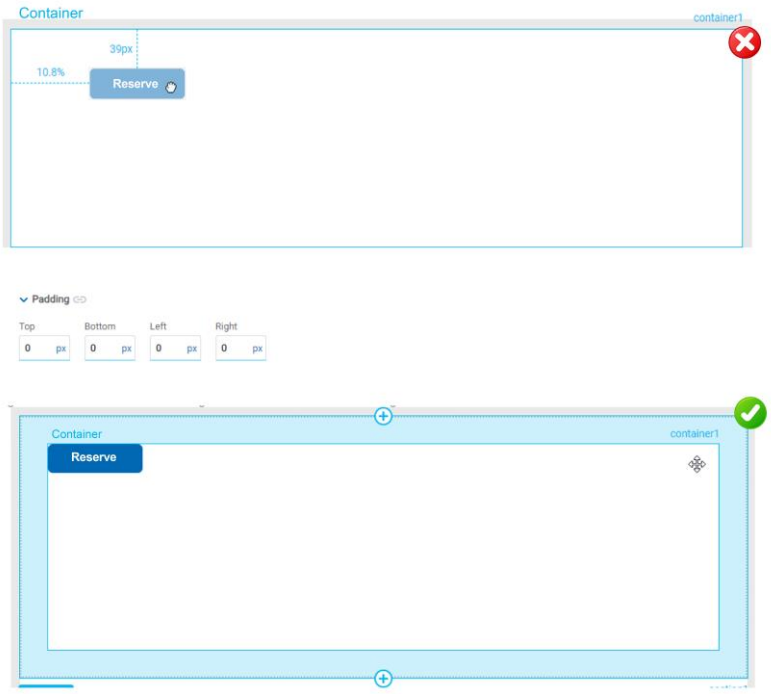
For example, adding unnecessary containers makes modeling difficult. If a container occupies the same size as the section, container or item that contains it, then it can be removed.



Padding

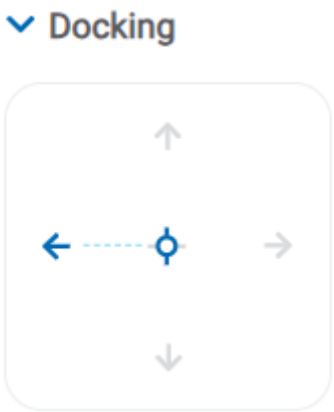
- Align elements within a container using “padding”, rather than manually. Besides it allows to control spaces easily.

For example, use:



Docking

- Dock the elements to the side that must remain fixed when it changes size.

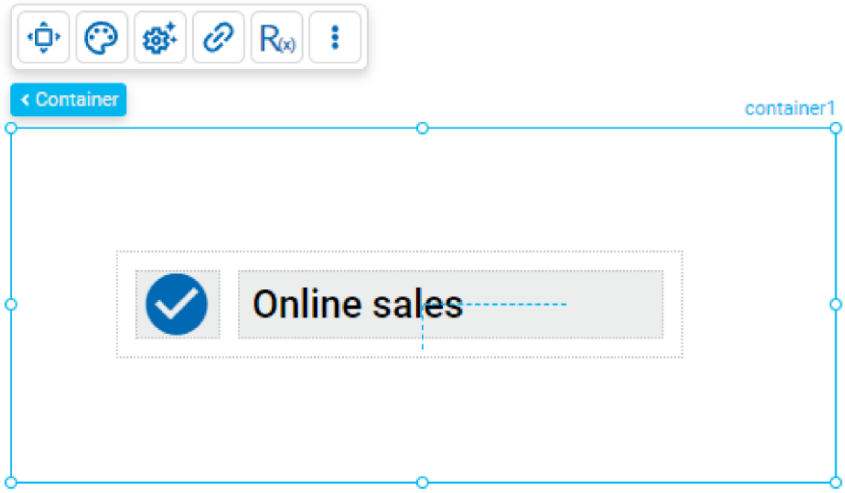


Alignment icons can also be useful, to dock to a specific position and automatically remove the margin if necessary.



Elements in the same row

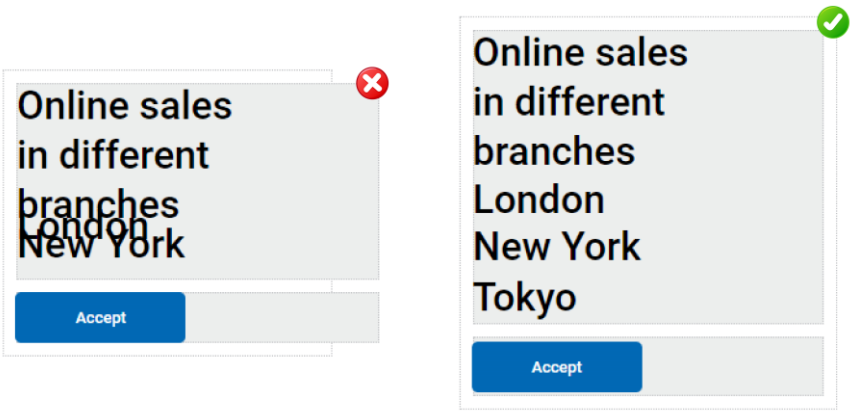
When modeling two or more horizontally oriented elements it is recommended to use a row type “Layout” element to contain them. This allows the page to be responsive and these elements do not overlap.



Elements in a same column

- When modeling two or more vertically oriented elements it is recommended to use a column type “Layout” element to contain them.

If there are items within the design that are of variable size, the height of the item must be indicated as “automatic”. This ensures they do not overlap.



Minimum size

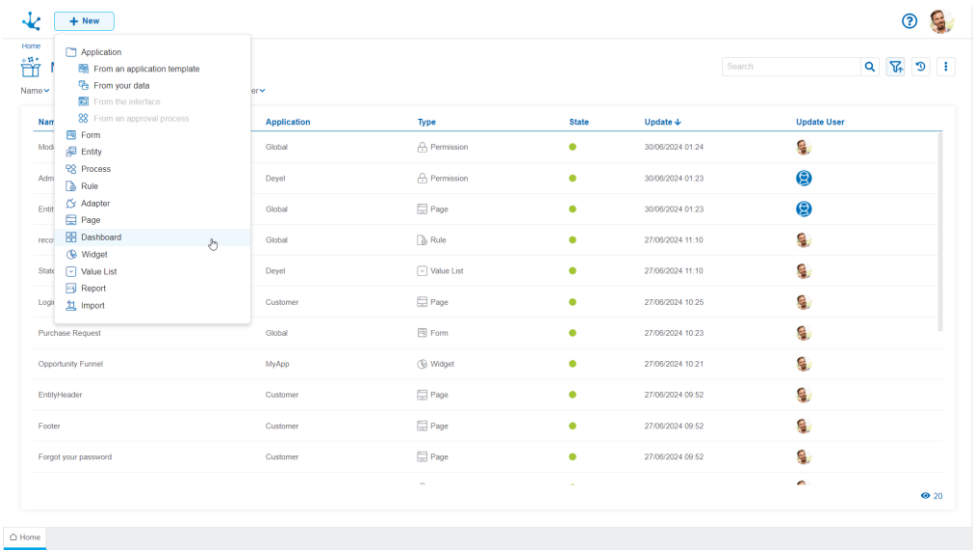
- The “Minimum Height” and “Minimum Width” properties of an element should be modeled to prevent its content from overflowing or displaying undesiredly if the size of the browser window is reduced. This improves the user experience and ensures that the page layout remains readable and functional across different browser window sizes.



3.4.14. Dashboards Modeling

The dashboard modeler is a tool that allows to model in an easy and intuitive way the applications dashboards and to publish them. The widgets included in the dashboards are modeled with the widget modeler.

From the portal, each business user can design their own dashboard, with the personalized information and their use preference, or they can use the dashboards created from the modeler.





This button is used to create a dashboard from the option  Dashboard.

The general characteristics of the dashboard modeling and the elements that make it up are described in the topics:

- Modeling Facilities
- Dashboards Properties

3.4.14.1. Modeling Facilities

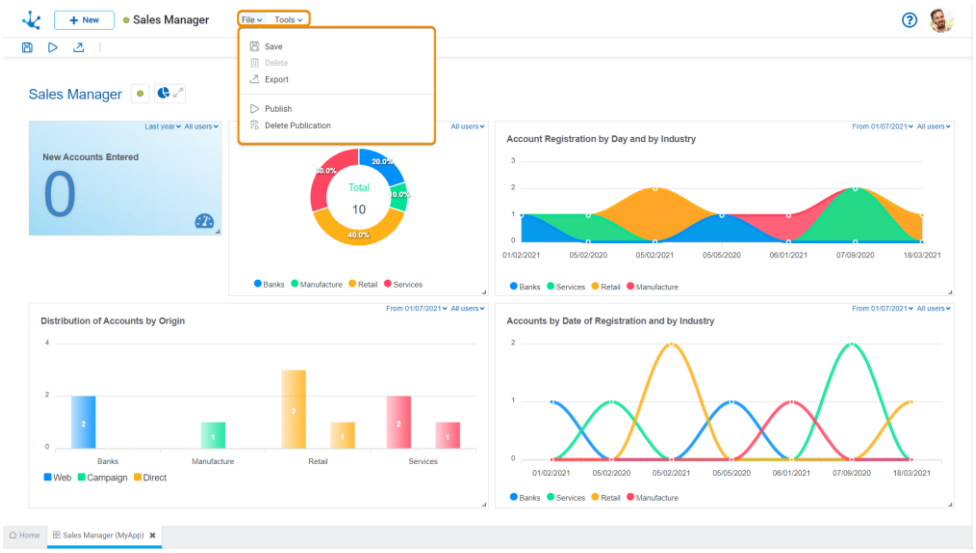
General characteristics of this modeler are specified below.

New Dashboard

The modeler user can define application dashboards. Once published, a dashboard becomes available to users who have permission in the application.

3.4.14.1.1. Expanded Menu

It is a horizontal list of options that contains vertical submenus with different operations on the dashboard or its modeling. Additionally, each submenu option can expand a dependant submenu.

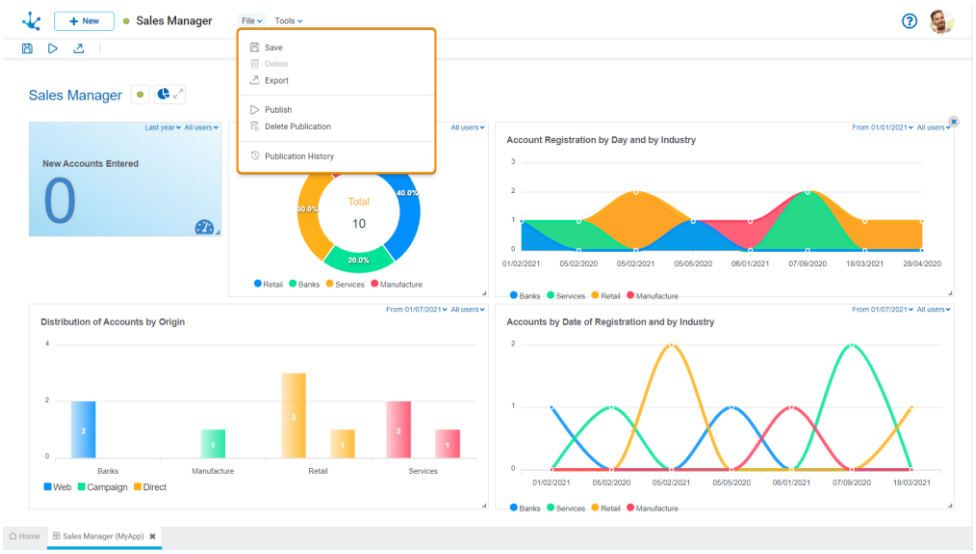


The expanded menu is made up of:

- Submenu File
- Submenu Tools

3.4.14.1.1.1. Submenu File

This submenu is opened by clicking the “File” option and it allows to make operations on the dashboard.



Save

This icon allows to save the object in the repository of **Deyel**, leaving its state as "Draft" or "Modified". If certain conditions are met, the modeler user receives a message indicating that the operation was performed correctly, otherwise they receive an explanatory message.

Conditions

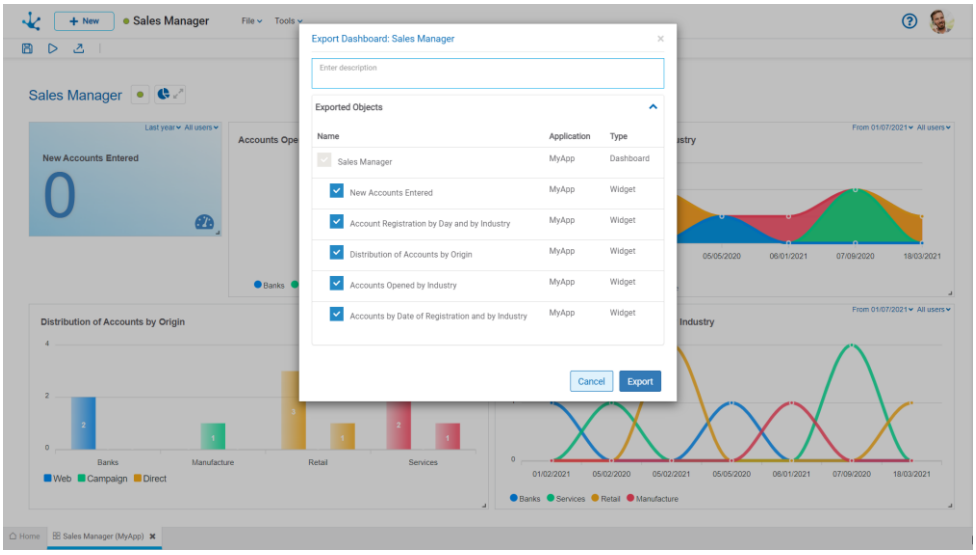
- There must be an object application.
- The name in the application must be unique.
- The object must not be locked by another user.
- There must be object's permissions.

Delete

It allows to delete the object only if it is in "Draft" state and does not have other associated objects of **Deyel** that were previously saved or published. When deleted, the tab it is on closes and the object is removed from the modeler's grid.

Export

This icon opens a window for the user to select and confirm the export of the object.



Description

In this property a text explaining the reason for the operation can be entered.
This text can be modified upon import and is displayed in the description column of the export record.

Exported Objects

By expanding the container, the indicators related to the dashboard being exported are shown. The ones not meant to be exported can be unchecked.
Press the "Cancel" button to leave the export without effect or the "Export" button to finish.

Publish

Through this icon the object changes to "Published" state and the modeler user receives the corresponding message, indicating the result of the operation. The condition for publishing is the same as for the "Save" operation.

Delete Publication

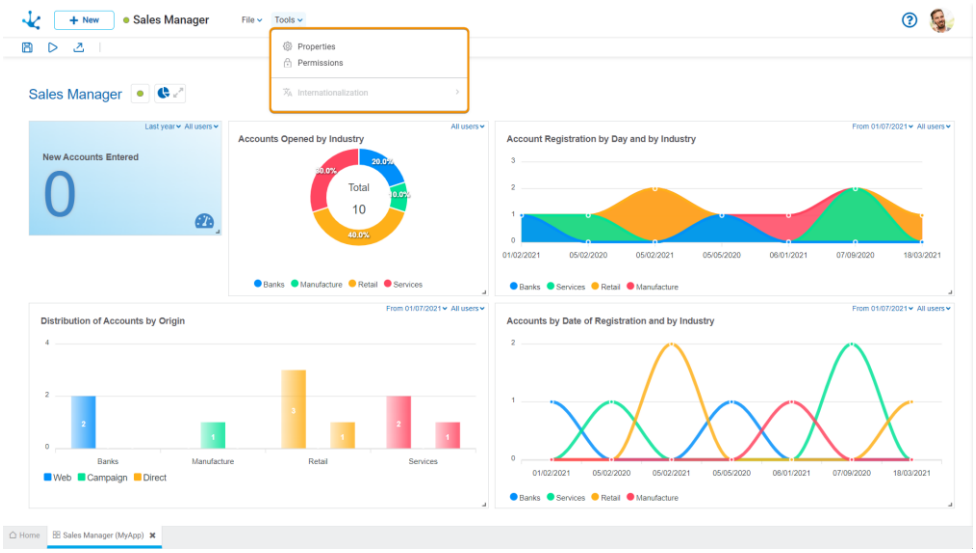
This icon allows to remove the use dashboard by returning it to the state "Draft", in addition to deleting the data.

Publication History

This icon allows managing versions of published objects, displaying their history, to have better control of the changes that each object has.

3.4.14.1.1.2. Submenu Tools

This submenu is opened by clicking on the "Tools" option, it allows to model dashboards properties, as well as their permissions and the internationalization.



Properties

Opens the dashboard properties panel.

Permissions

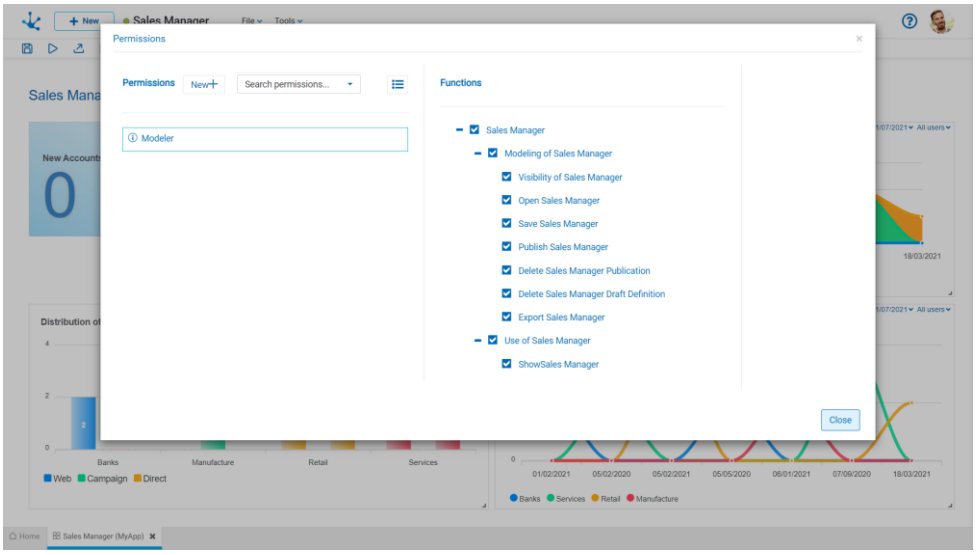
Allows to assign the security functions for use and modeling of the object to the existing permissions or to new permissions that the user creates, without having to go to the permission settings option from the menu.

Sections

- Permissions: Permissions to which object functions are assigned.
- Security Functions: Represents the total set of security functions, modeling and use of the object. Those that are marked are the ones included in the selected permission.

By default, all security functions for a new object are assigned to the permission Modeler.

Users who are assigned the permissions have access to the functions included in it.



New Opens a panel to create a new permission and once created, the security functions included in it must be selected .

Search permissions... Allows to select a permission from a list and enables the input of characters to filter the values in the list.

Opens the wizard to select a permission and once chosen, the necessary object security functions must be checked.

To unrelate a permission from the object, hover the cursor over the permission and press the icon . If there are functions selected for that permission, they must be unchecked in order to delete it.

Modeling Security Functions

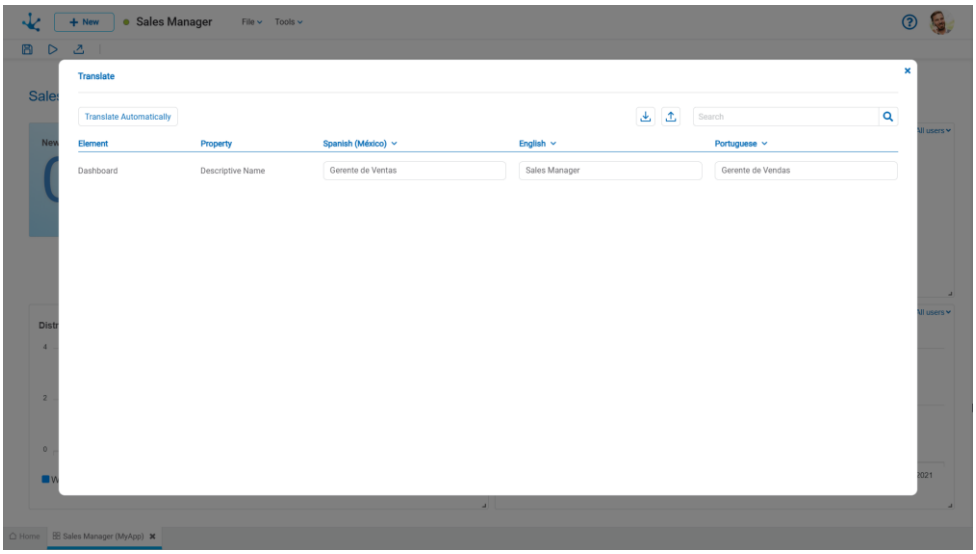
- Visibility: Allows to display the object in the Deyel modeler grid.
- Open: Allows to open the object from the Deyel modeler.
- Save: Enables the operation of saving modifications made to the object.
- Publish: Enables the operation of publishing the object leaving its state as "Published".
- Delete publication: Enables the operation of deleting the object publication leaving its state as "Draft".
- Delete draft definition: Enables the operation of deleting the object.
- Export: Enables the operation to export the object from the tools submenu of the expanded menu.

Use Security Functions

- Show: Allows to show the object.

Internationalization

The "Internationalization" option is enabled when the property Internationalize is activated. To access the translation panel, it is necessary to select "Internationalization" and then choose "Translate".



In this panel, all the translatable properties of the object are displayed, along with their content in the original language, and translations in the other available languages can be entered manually.

If a value is modified in the original language column, the change is applied directly to the object's property.

In the header of each language column, a menu that allows selecting another language can be displayed, updating the column with its contents.

Options Available in the Top Margin

Translate Automatically Generates translations from the original language to other languages using artificial intelligence.

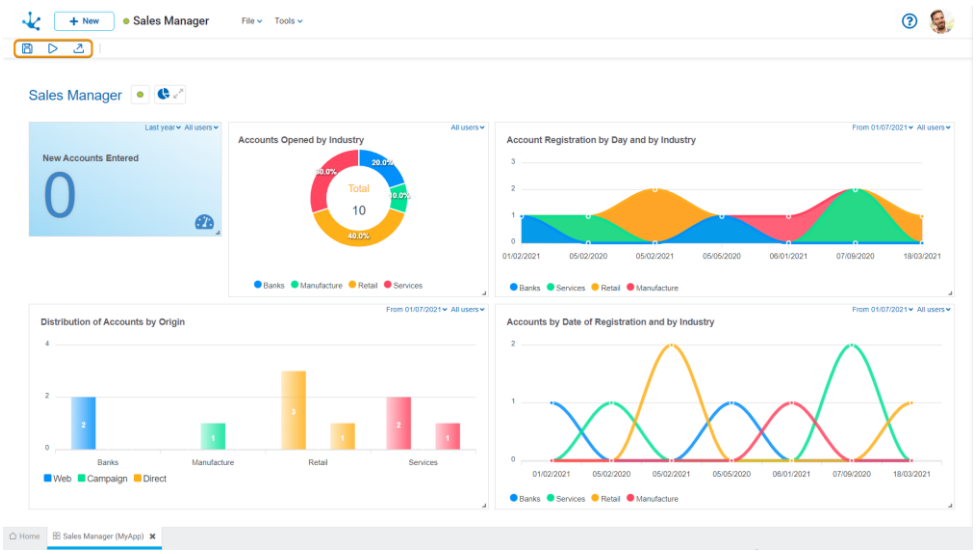
Downloads an Excel file with the contents of the translation panel.

Allows selecting an Excel file to import translations, making it easier to manage large volumes of content.

Filters the contents of columns based on the characters entered in the search field.

3.4.14.1.2. Top Toolbar

Contains icons for the quick access to the most used operations from the expanded menu.

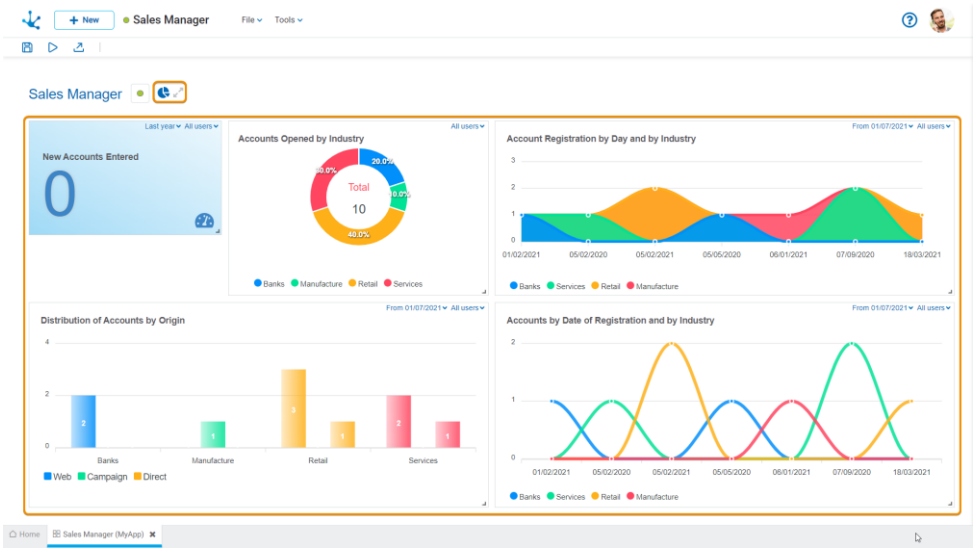


File

- Save
- Publish
- Export

3.4.14.1.3. Graphic Modeling Area

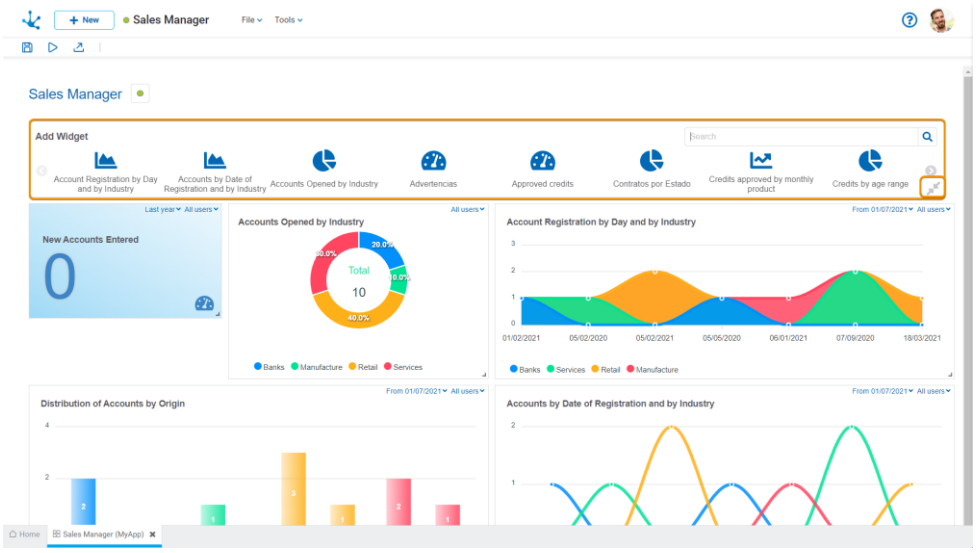
The graphic modeling area is the space where the different widgets that make up the dashboard are dragged.



Operations

Add Widgets

 Displays the gallery of published widgets on which the user has permission. For each widget its name is visualized as well as its icon.
The "Search" option, in the gallery allows to filter widgets by the entered text.



 Closes the widgets gallery.

Modify Widgets

The modifications that can be applied on the widgets of a dashboard, have impact only in the visualization of such dashboard.

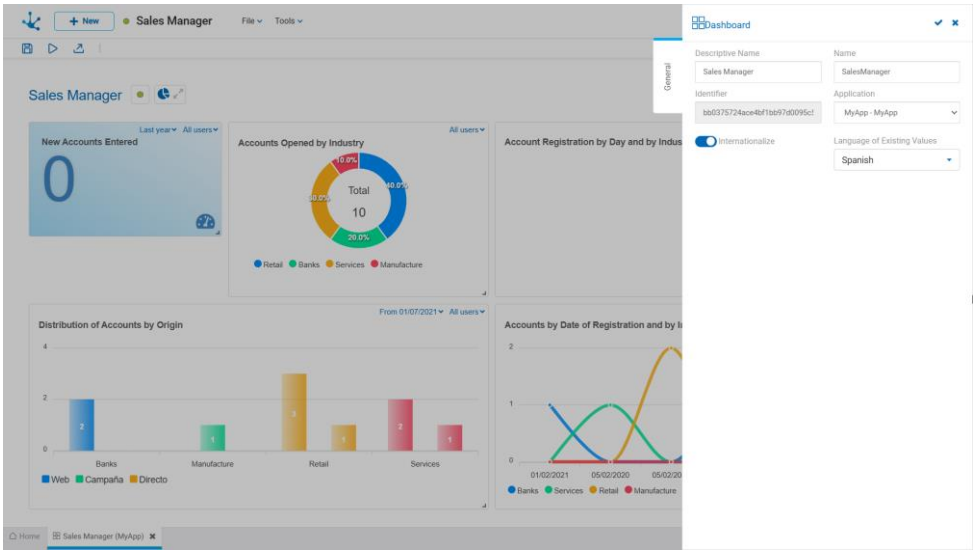
- Modify the widgets size. In order to secure the suitable display, each widget is assigned a minimum size to respect.
- Modify the location of the widgets in the dashboard area.
- Change user filters.
- Change date filters.
- Delete dashboard widget.

> *Operations modify size and modify location are done with the "Drag and Drop" facility.*

3.4.14.2. Dashboard Properties

The properties of the dashboards can be entered both at the time of their creation and when modifying an existing one.

Entering the dashboard properties panel is done using the icon  which is in the top toolbar.



General Tab

Properties

Descriptive Name
It is the name visualized from the dashboard.

Name
It is the dashboard´s name. It does not allow blanks or special characters. It is unique and required.

Identifier
It is the internal dashboard´s name. It is automatically generated.

Application
Allows to define the application to which the object belongs. If no application is reported, the object is assigned to the application "Global".

Internationalize
Enables the translation of object properties and its elements into different languages. When activated, it enables the option "Internationalization" from the "Tools" drop-down menu and the property Language of Existing Values, which is displayed on the right.

Language of Existing Values
Displayed when the property Internationalize is activated for the first time and indicates the original language. Its selection is required and the available options are: "Spanish", "English" and "Portuguese".

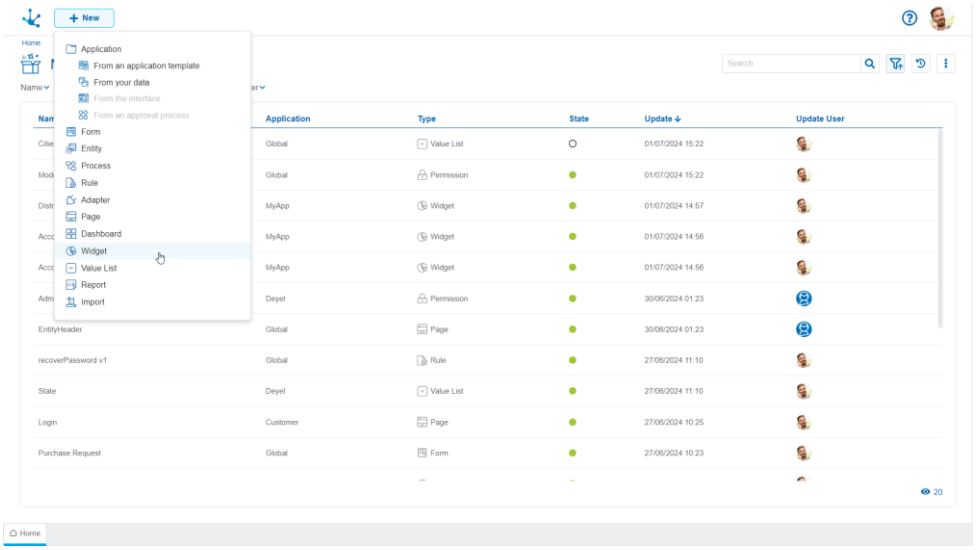
Actions

The icon ✔ is used to confirm the modifications made in the properties panel.

The icon ✖ is used to close the properties panel, if it was not previously saved, changes are discarded.

3.4.15. Widgets Modeling

The widgets modeling can be used to design and implement in an easy and intuitive way, the required widgets to display the relevant information for the user, as management issues, tasks and calendars.



New This button is used to create a widget from the option Widget.

The general characteristics of the widgets modeling and the elements that make it up are described in the topics:

- Modeling Facilities
- Widgets Properties

3.4.15.1. Modeling Facilities

General characteristics of this modeler are specified below.

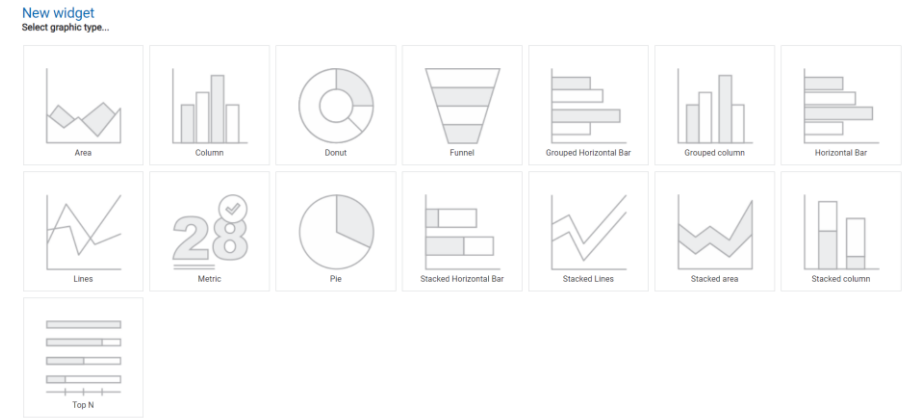
New Widget

The modeler user can design a new widget, that after being published, is available to be used on an application dashboard or a user dashboard.

Steps for Creating a New Widget

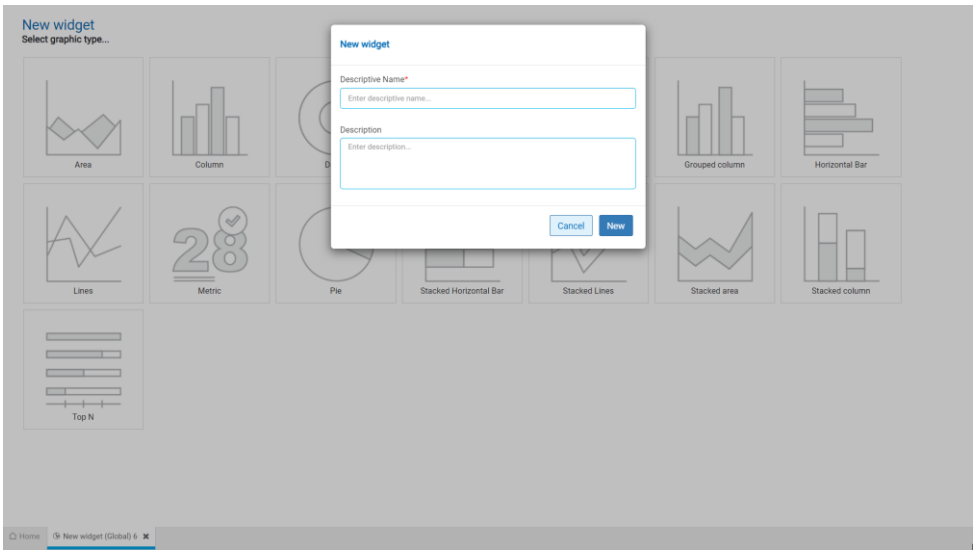
Step 1

Select the graphic type.



Step 2

Assign the name, optionally write the description and press the “Create” button.



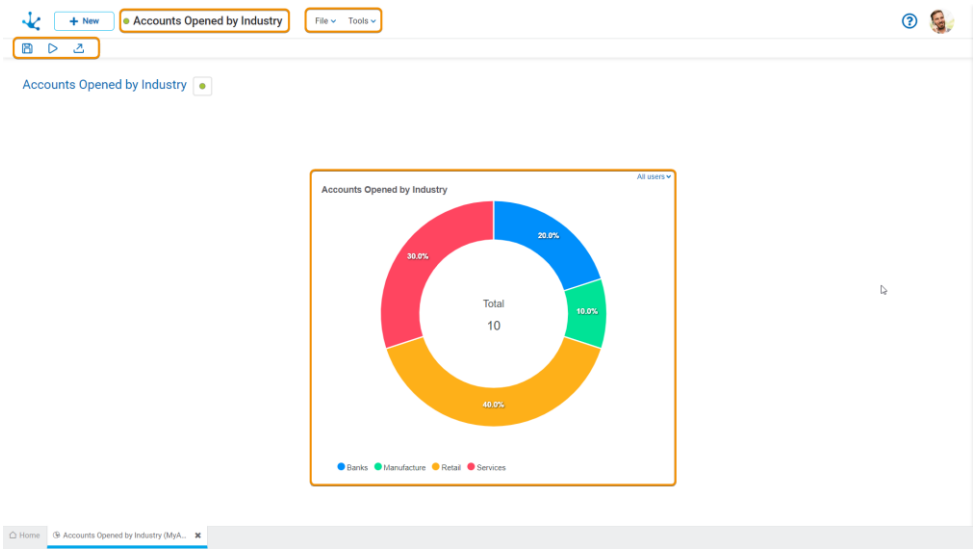
Step 3

Complete the widget using the sections of the workspace.

Workspace Sections

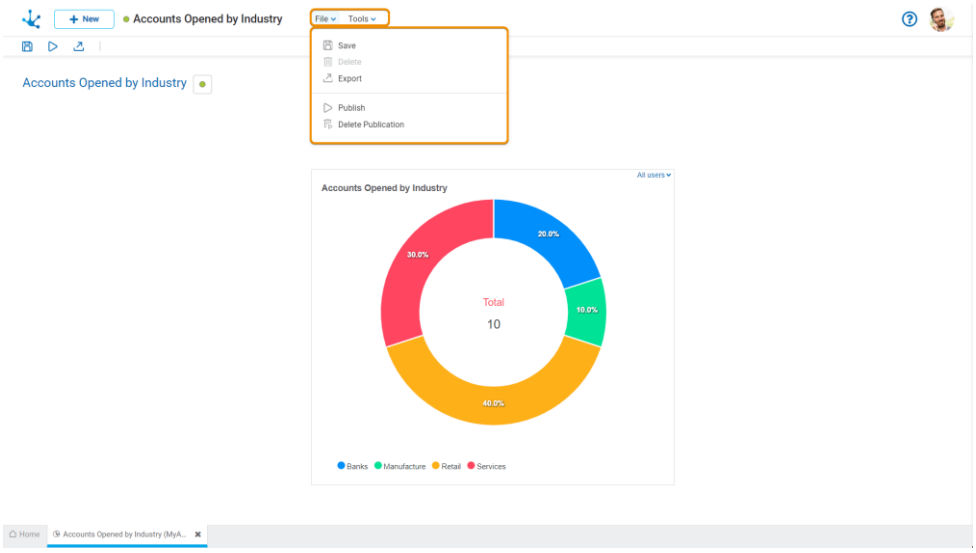
- Widget Information
  - State
  - Name

- Expanded Menu
- Top Toolbar
- Graphic Modeling Area



3.4.15.1.1. Expanded Menu

It is a horizontal list of options that contains vertical submenus with different operations on the widget or its modeling. Additionally, each submenu option can expand a dependant submenu.

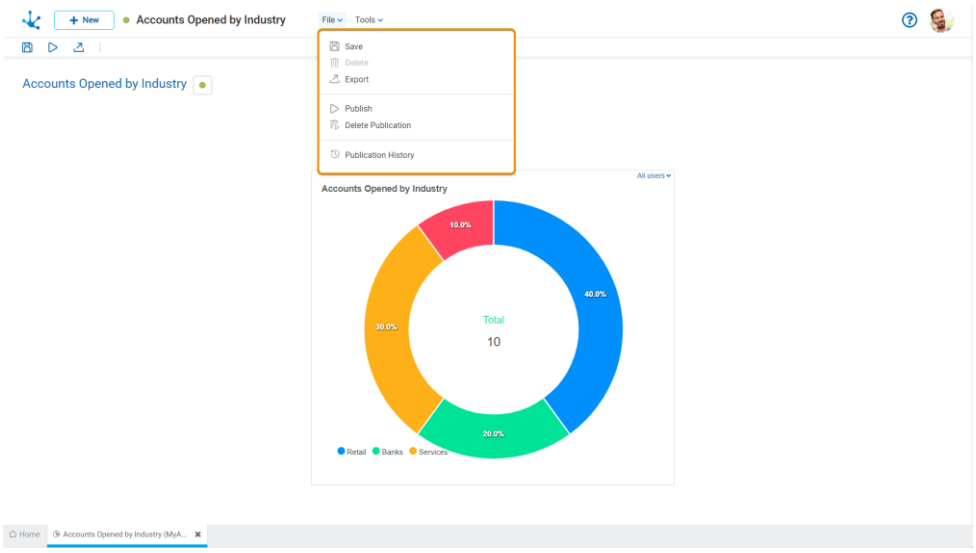


The expanded menu is made up of:

- Submenu File
- Submenu Tools

3.4.15.1.1.1. Submenu File

This submenu is opened by clicking the “File” option and it allows to make operations on the widget.



Save

This icon allows to save the object in the repository of **Deyel**, leaving its state as "Draft" or "Modified". If certain conditions are met, the modeler user receives a message indicating that the operation was performed correctly, otherwise they receive an explanatory message.

Conditions

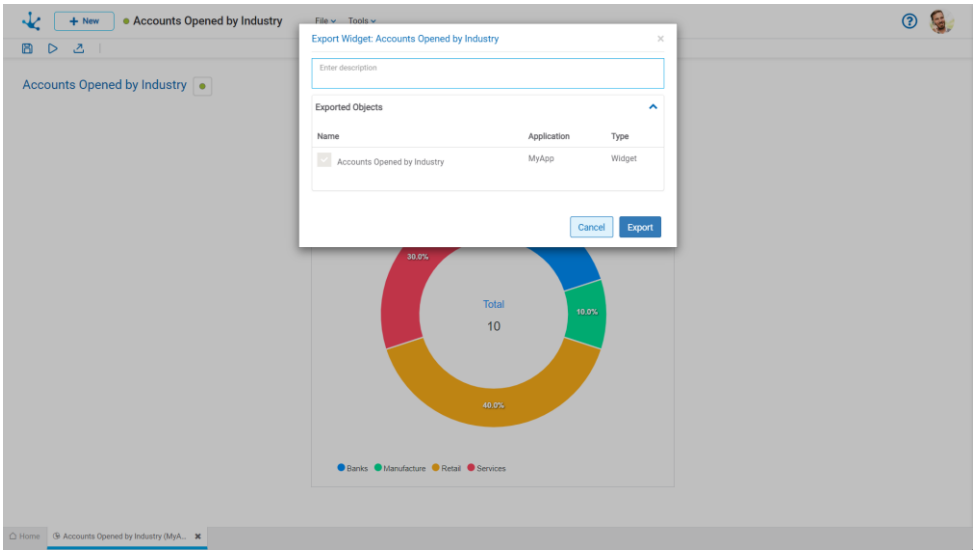
- There must be an object application.
- The name in the application must be unique.
- The object must not be locked by another user.

Delete

It allows to delete the object only if it is in "Draft" state and does not have other associated objects of **Deyel** that were previously saved or published. When deleted, the tab it is on closes and the object is removed from the modeler’s grid.

Export

This icon opens a window for the user to select and confirm the export of the object.



Description

In this property a text explaining the reason for the operation can be entered.
This text can be modified upon import and is displayed in the description column of the export record.

Exported objects

By expanding the container, the object being exported is displayed.

Press the "Cancel" button to leave the export without effect or the "Export" button to finish.

Publish

Through this icon the object changes to "Published" state and the modeler user receives the corresponding message, indicating the result of the operation. The condition for publishing is the same as for the "Save" operation.

Delete Publication

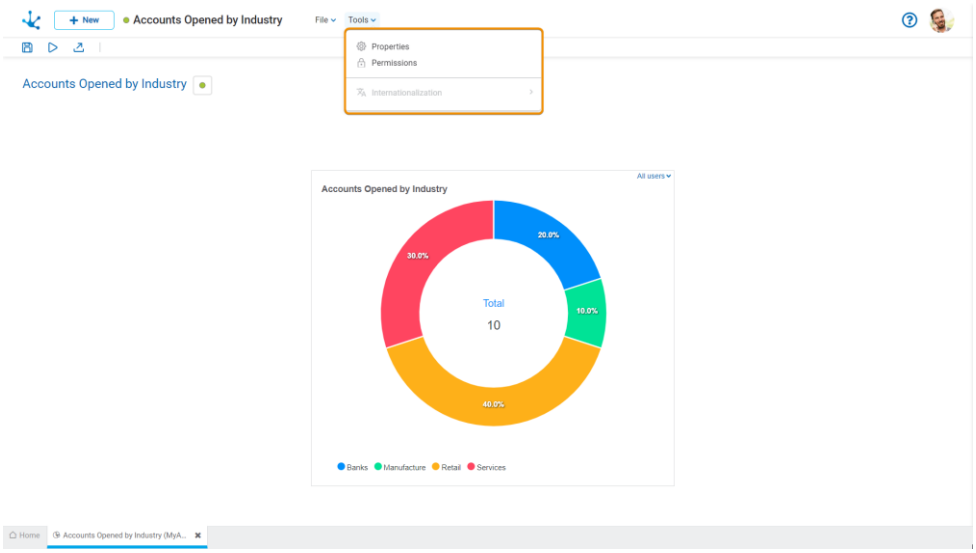
Allows to remove the use widget by returning it to the state "Draft", in addition to deleting the data.

Publication History

This icon allows managing versions of published objects, displaying their history, to have better control of the changes that each object has.

3.4.15.1.1.2. Submenu Tools

This submenu is opened by clicking on the "Tools" option and it allows to model widget properties, as well as their permissions and the internationalization.



Properties

Opens the widget properties panel.

Permissions

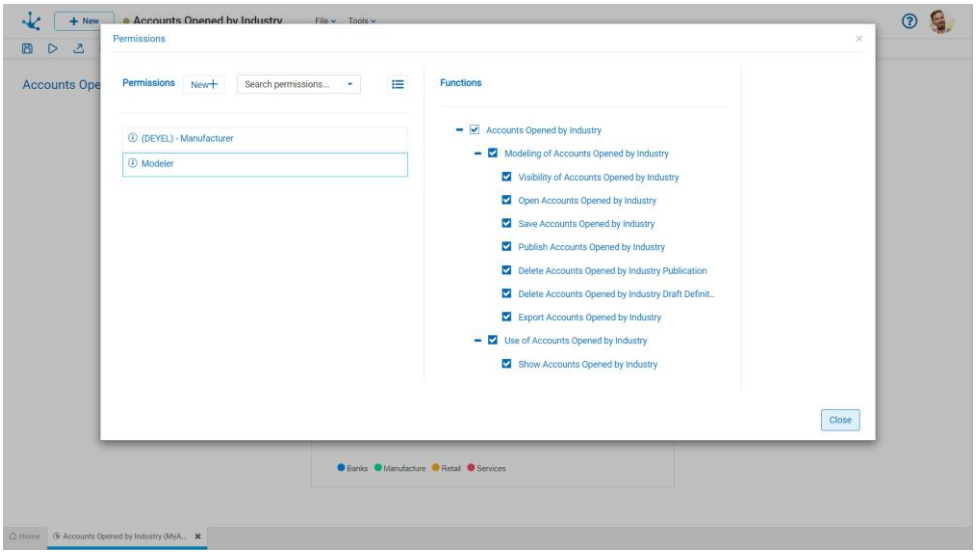
Allows to assign the security functions for use and modeling of the object to the existing permissions or to new permissions that the user creates, without having to go to the permission settings option from the menu.

Sections

- Permissions: Permissions to which object functions are assigned.
- Security Functions: Represents the total set of security functions, modeling and use of the object. Those that are marked are the ones included in the selected permission.

By default, all security functions for a new object are assigned to the permission Modeler.

Users who are assigned the permissions have access to the functions included in it.



New+ Opens a panel to create a new permission and once created, the security functions included in it must be selected .

Search permissions... Allows to select a permission from a list and enables the input of characters to filter the values in the list.

Opens the wizard to select a permission and once chosen, the necessary object security functions must be checked.

To unrelate a permission from the object, hover the cursor over the permission and press the icon ✖. If there are functions selected for that permission, they must be unchecked in order to delete it.

Modeling Security Functions

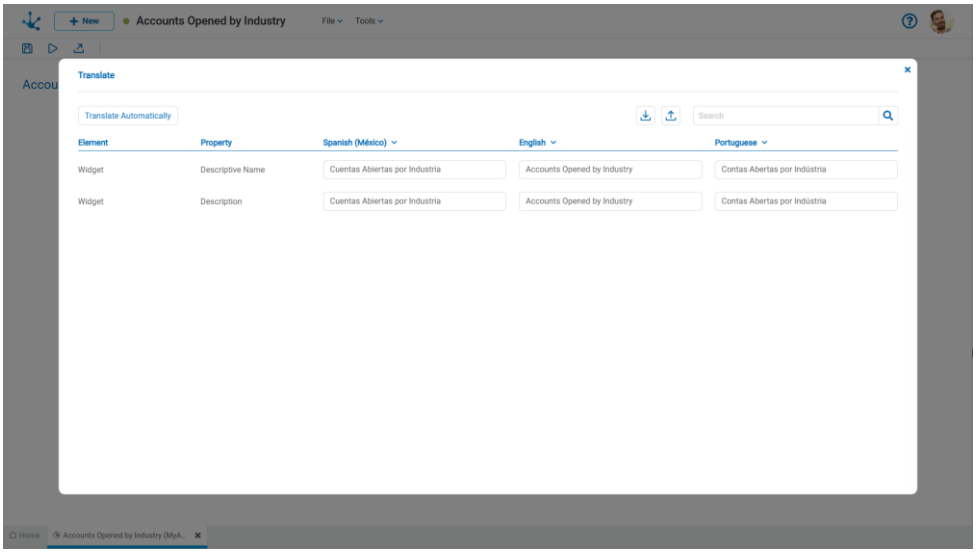
- Visibility: Allows to display the object in the Deyel modeler grid.
- Open: Allows to open the object from the Deyel modeler.
- Save: Enables the operation of saving modifications made to the object.
- Publish: Enables the operation of publishing the object leaving its state as "Published".
- Delete publication: Enables the operation of deleting the object publication leaving its state as "Draft".
- Delete draft definition: Enables the operation of deleting the object.
- Export: Enables the operation to export the object from the tools submenu of the expanded menu.

Use Security Functions

Show: Enables the show object operation.

Internationalization

The "Internationalization" option is enabled when the property Internationalize is activated. To access the translation panel, it is necessary to select "Internationalization" and then choose "Translate".



In this panel, all the translatable properties of the object are displayed, along with their content in the original language, and translations in the other available languages can be entered manually.

If a value is modified in the original language column, the change is applied directly to the object's property.

In the header of each language column, a menu that allows selecting another language can be displayed, updating the column with its contents.

Options Available in the Top Margin

Translate Automatically Generates translations from the original language to other languages using artificial intelligence.

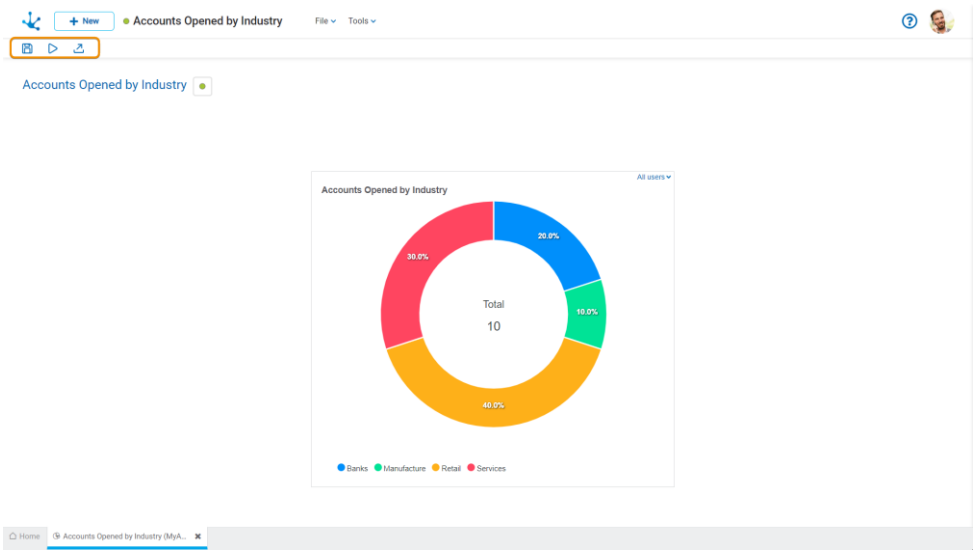
Downloads an Excel file with the contents of the translation panel.

Allows selecting an Excel file to import translations, making it easier to manage large volumes of content.

Filters the contents of columns based on the characters entered in the search field.

3.4.15.1.2. Top Toolbar

Contains icons for the quick access to the most used operations from the expanded menu.

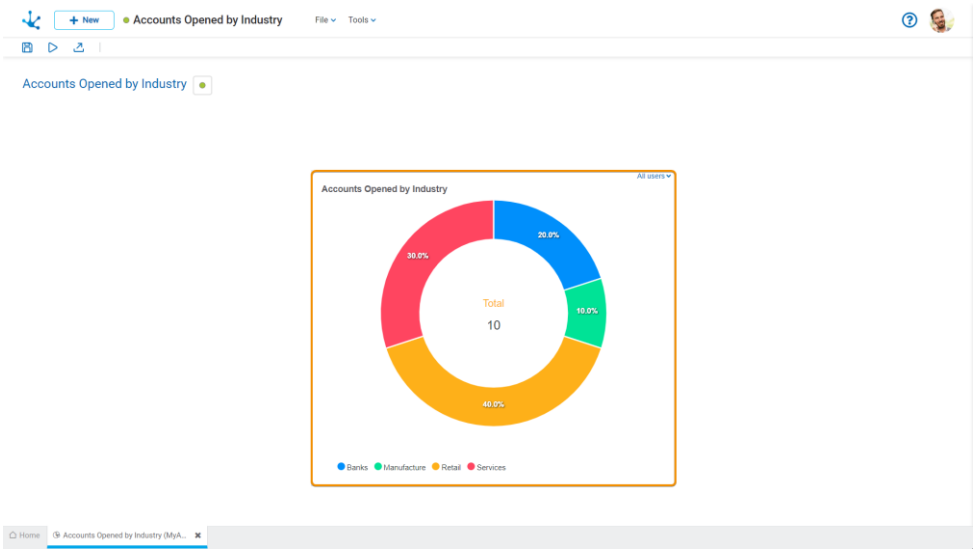


File

- Save
- Publish
- Export

3.4.15.1.3. Graphic Modeling Area

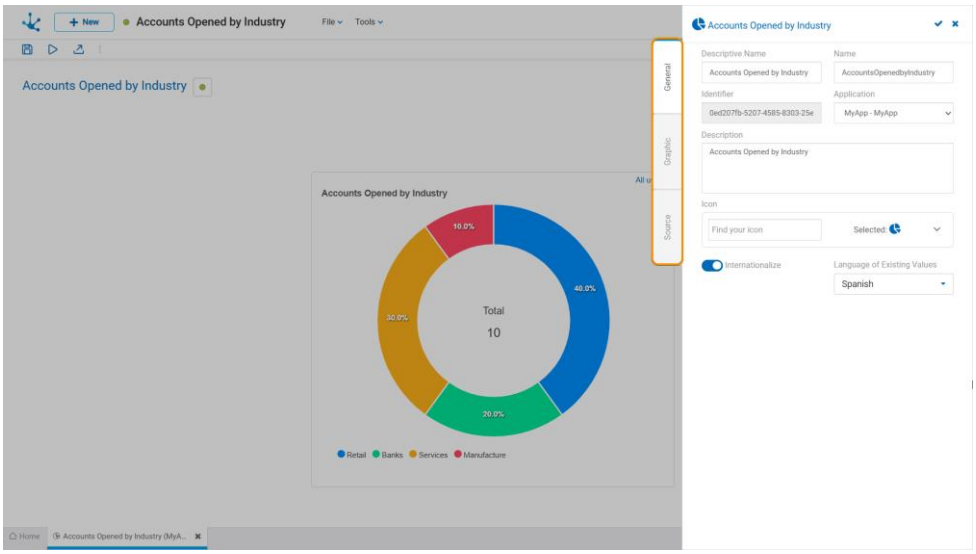
The graphic modeling area is the space where the widget is displayed according to the properties defined and by hovering over it the data values graphically represented can be seen.



3.4.15.2. Widget Properties

The properties of the widgets can be entered both at the time of their creation, and when modifying an existing one.

Entering the widget properties panel is done using the icon which is in the top toolbar.

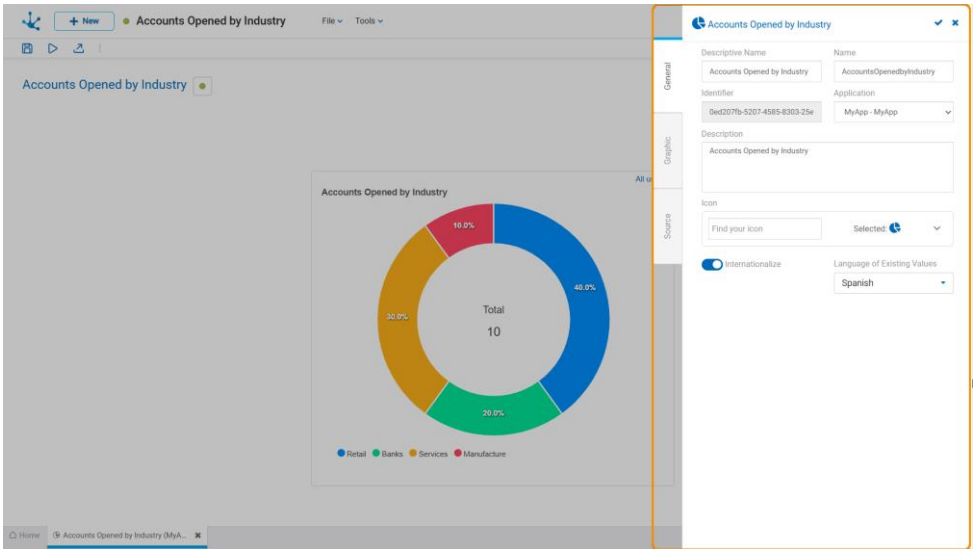


Tabs

- General
- Graphic
- Source

3.4.15.2.1. General

The properties panel is displayed on the right side of the widget modeler, where the first tab corresponds to general information.



Properties

Descriptive Name
It is the name used by users to reference the widget, for example in the modeler's grid and in the dashboards. It is required.

Name
It is internally used to reference the widget. It does not allow blanks or special characters. It is unique and required.

Identifier
Uniquely identifies the widget. It is automatically generated.

Application
Allows to define the application to which the object belongs. If no application is reported, the object is assigned to the application "Global".

Description
Text that describes the topic the widget allows to analyze, when incorporated into a dashboard.

Icon
Allows to incorporate an icon that visually identifies the widget when the operation is made Add widget, from the use of the dashboard.
The list of available icons is displayed to select one of them. The list is reduced as the name is written on the text "Find your icon".

Internationalize
Enables the translation of object properties and its elements into different languages. When activated, it enables the option "Internationalization" from the “Tools” drop-down menu and the property Language of Existing Values, which is displayed on the right.

Language of Existing Values
Displayed when the property Internationalize is activated for the first time and indicates the original language. Its selection is required and the available options are: “Spanish”, “English” and “Portuguese”.

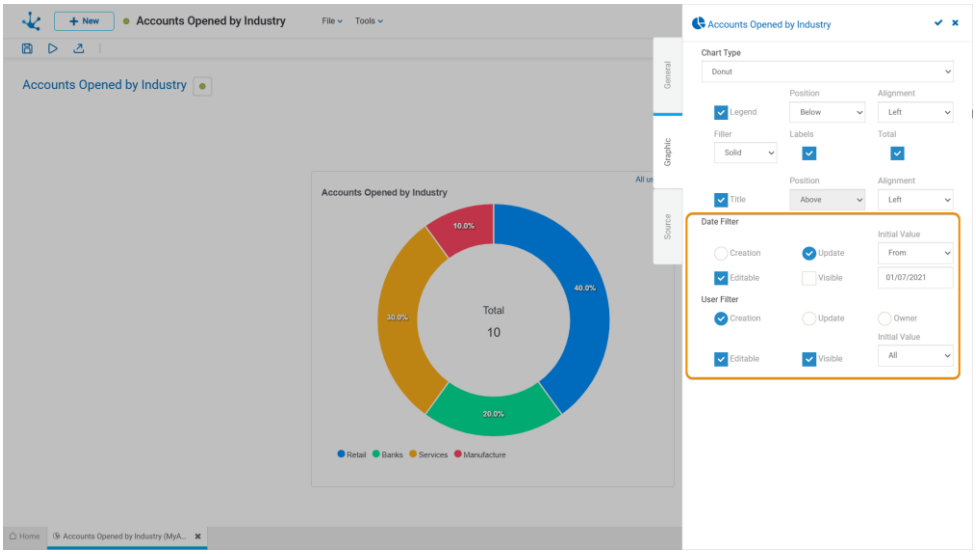
Actions

The icon ✔ is used to confirm the modifications made in the properties panel.

The icon ✖ is used to close the properties panel, if it was not previously saved, changes are discarded.

3.4.15.2.2. Graphic

The second tab of the side panel corresponds to the charts information. Allows to do the management of the properties of each chart type.



Properties

In addition to these shared properties, each type of widget has specific properties, which are detailed for each chart.

- Metrics
- Areas
- Lines
- Bars and Columns
- Pies and Donuts
- Funnel

Title

If this property is selected, when using the widget on a dashboard, the descriptive name is displayed. In this case it must also indicate:

Position

Allows to locate the title in the upper or lower sector of the widget area.

Justify

Allows to position the title to the right, center, or left, within the widget area.

Date Filter

Creation

Indicates that the date filter is applied on the creation date.

Update

Indicates that the date filter is applied on the modification date.

Editable

Indicates if the user can modify the filter conditions when using the widget from the dashboard, only if they have permission to do it.

If the user makes modifications and saves the dashboard, the new filter conditions established are maintained.

Visible

Indicates whether the filter is visualized or not when using the widget from the dashboard.

Initial Value

Indicates the value displayed from the dashboard by default.

Possible Values:

- Today
- From
- Last 7 days
- Current month
- Current year
- Last month
- Last year

User Filter

Creation

Indicates that the user filter is applied on the creation user.

Update

Indicates that the user filter is applied on the modification user.

Owner

Indicates that the user filter is applied on the instance owner.

Editable

Indicates if the user can modify the filter conditions when using the widget from the dashboard, only if they have permission to do it.

If the user makes modifications and saves the dashboard, the new filter conditions established are maintained.

Visible

Indicates whether the filter is visualized or not when using the widget from the dashboard.

Initial Value

Indicates the value displayed from the dashboard by default.

Possible Values:

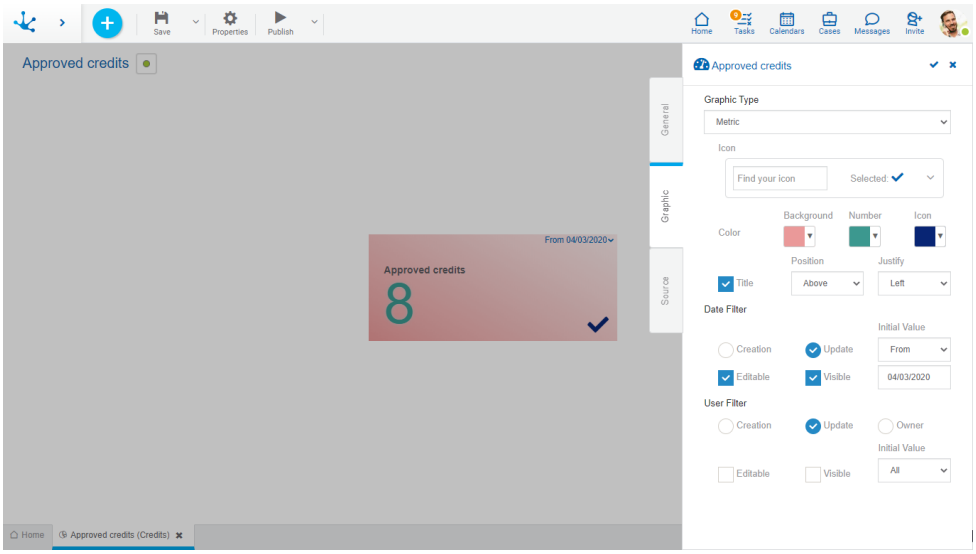
- Current user
- All
- My Team

> *My team is made up by all of the users that belong to the organizational unit for which the logged in user is the administrator and all users of a role for which they are coordinator.*

3.4.15.2.2.1. Metrics

Metric type widgets represent a numerical value, a quantitative measure that is used to measure and compare.

To the properties shared by all the widgets those specific to metrics are added.



Properties

Icon

Allows to incorporate an image displayed within the widget. It is incorporated in the same way as the icon of the "General" tab.

Color

Background

Allows to modify the background color of the widget, selecting it from the color palette.

Number

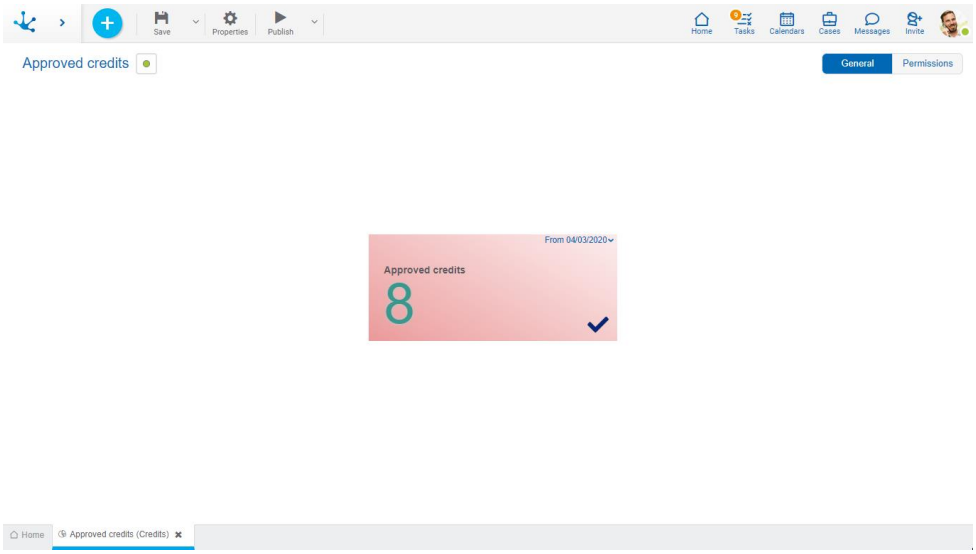
Allows to modify the value color of the widget, selecting it from the color palette.

Icon

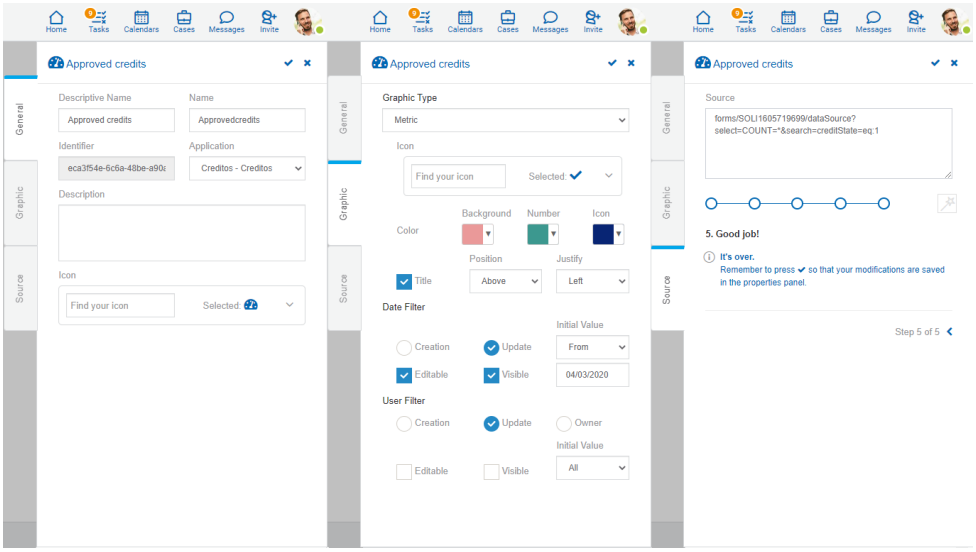
Allows to modify the icon color, selecting it from the color palette.

Example of Use

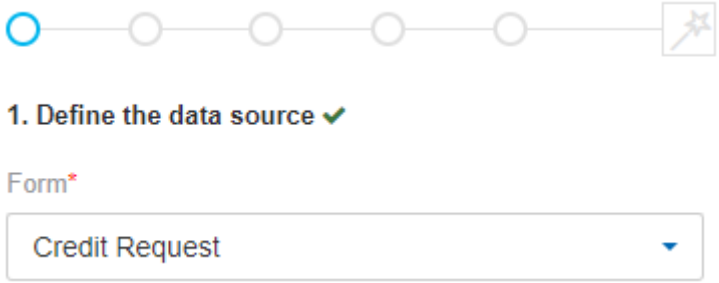
This metric chart shows the number of accounts entered during a selected period and for the users indicated in the filter.



The source code to model this chart is detailed.



Steps of the Modeling Wizard



The "Credit Request" form from which the data displayed in the chart is taken should be selected.



2. Define the data you want to display ✔

Series*

count(ALL)

The “count(ALL)” function is used to determine the number of loans.



3. Define display criteria

Select by

equal(creditStatus, 1)

A condition is added to display all requests with "creditState=1" where "creditState" corresponds to a value list with numerical code "1" and descriptive text "Approved".

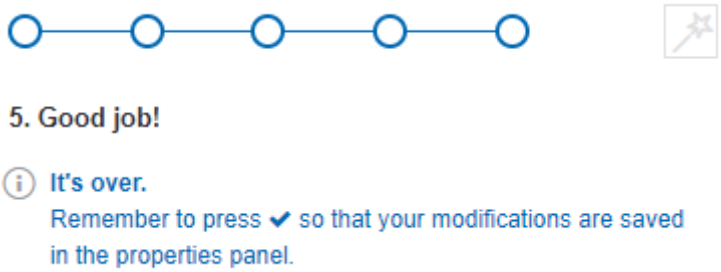


4. Define how to group the results

Group by

'Ctrl + Space' to open assistant.

In this type of widget the results are not grouped.



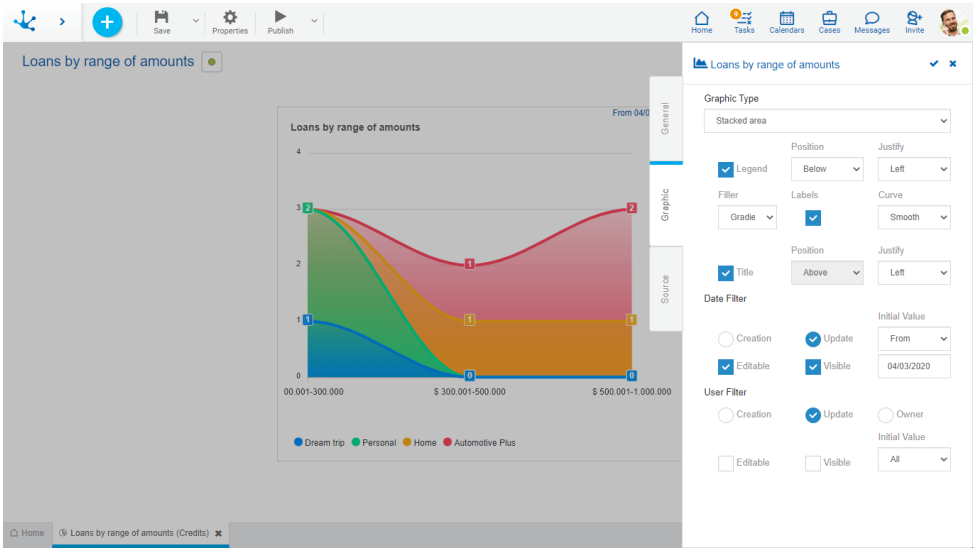
Step 5 reporting that all steps are met is completed .

3.4.15.2.2.2. Areas

Widgets of area type, comprehend the area and stacked area charts.

They can show changes over time, general trends, and continuity in a data set. But, while they may work in the same way as row widgets, the space between the row and the axis is filled in, indicating volume.

To the properties shared by all the widgets are added those specific to areas.



Properties

Legend

Indicates the incorporation of the data series description.
If this property is selected, it must also indicate:

Position

Select above or below.

Justify

If above or below position is selected, it allows to indicate right, center or left alignment.

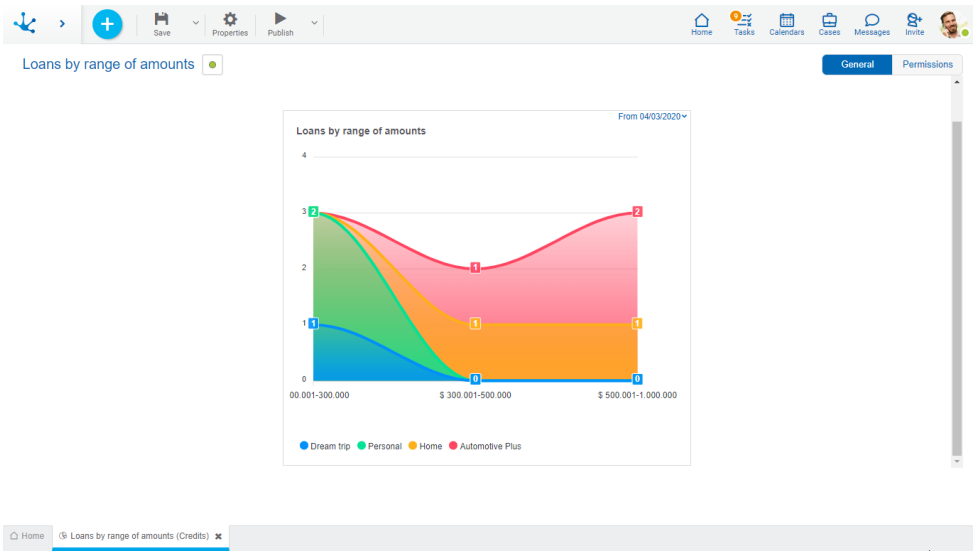
Filler
Allows to highlight the area of data series in a gradient or solid way.

Labels
Allows to indicate whether the labels corresponding to the data series are displayed.

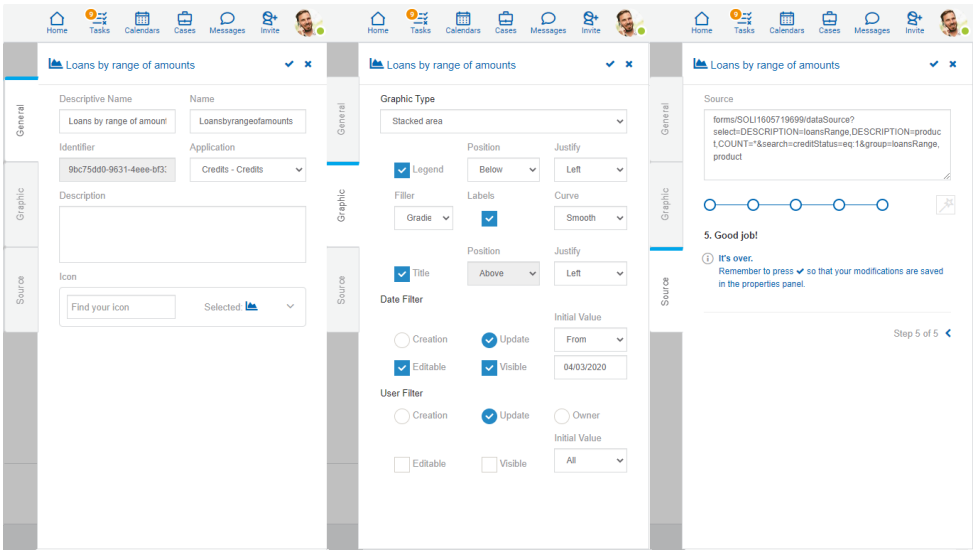
Curve
Indicates whether data lines are displayed as smooth, straight, or stepwise curve.

Example of Use

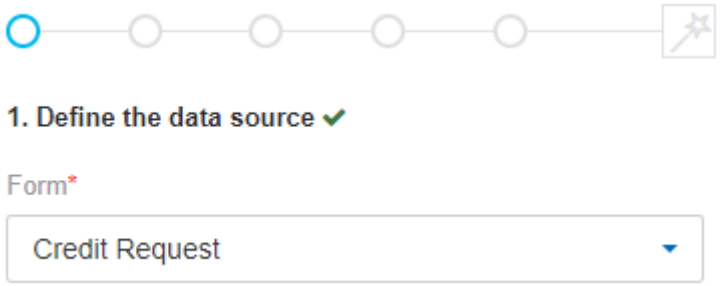
This stacked area widget shows the number of credits approved by range of expenses during a selected period and for the users indicated in the filter.
The ranges of expenses are represented on the horizontal axis and the number of credits approved for each range on the vertical axis.
The percentage of each product is given by the colored surface, and the grouping of all surfaces represents the total number of approved credits.



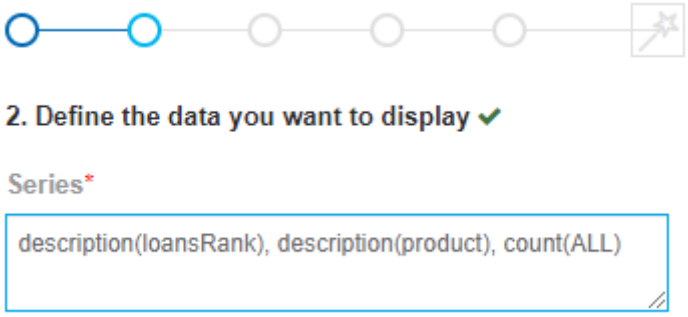
Below are the properties modeled to get this chart and the source code that is automatically generated when completing the wizard steps for widgets modeling.



Steps of the Modeling Wizard

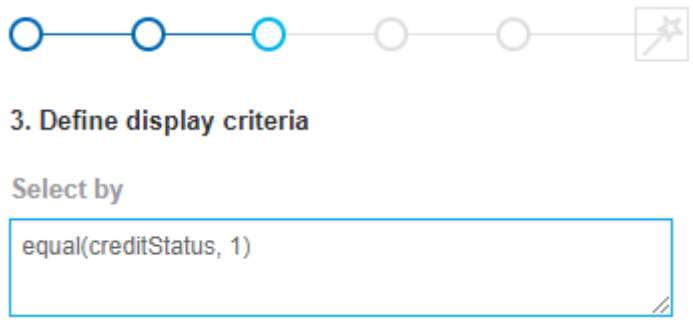


The "Credit Request" form from which the data displayed in the chart is taken should be selected.

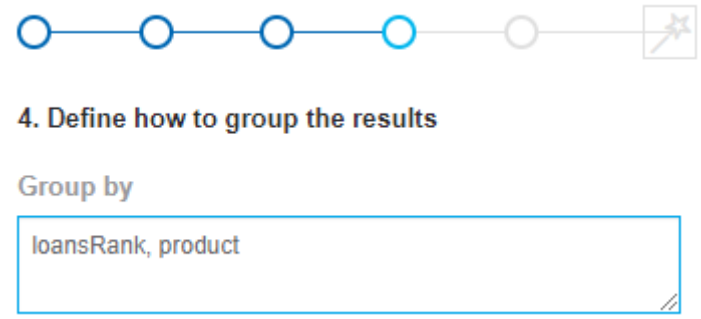


First the "loansRank" field of the form visualized on the horizontal axis of the chart is indicated. Secondly the "product" field that corresponds to the colored surfaces is indicated. And lastly the "count(ALL)" function is used to determine the number of loans per product that are displayed on the vertical axis.

> *For fields that use value lists, the "description()" function must be used to display every list value with its descriptive text, otherwise the numerical code* is displayed.

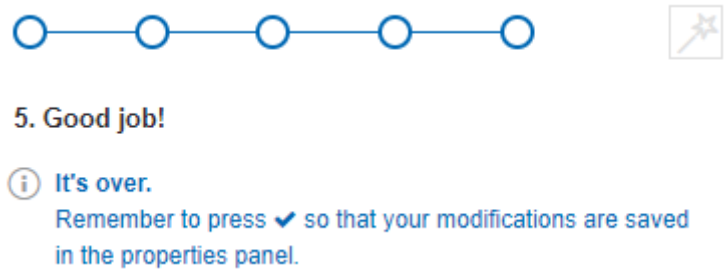


A condition is added to display all requests with "creditState=1" where "creditState" corresponds to a value list with numerical code "1" and descriptive text "Approved".



Grouping the results by the same form fields that were modeled in step 2 is defined.

> *In this step, the names of fields must be used without the "description()" function, despite being value lists.*



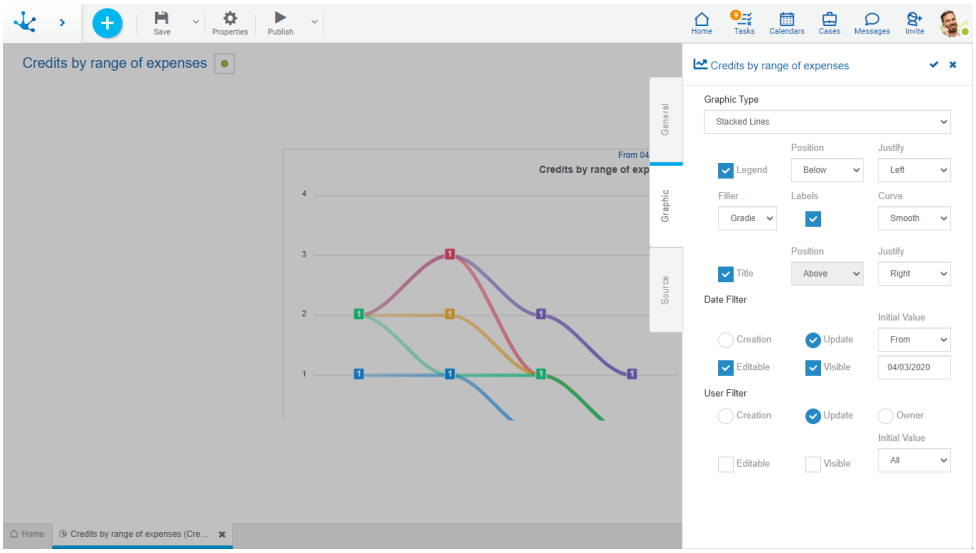
Step 5 reporting that all steps are met is completed .

3.4.15.2.2.3. Lines

Widgets of line type comprehend line and stacked line charts.

They are useful to show tendencies through time and compare different data series.

To the properties shared by all the widgets are added those specific to lines.



Properties

Legend

Indicates the incorporation of the data series description.
If this property is selected, it must also indicate:

Position

Select above or below.

Justify

If above or below position is selected, it allows to indicate right, center or left alignment.

Filler

Allows to highlight the area of data series in a gradient or solid way.

Labels

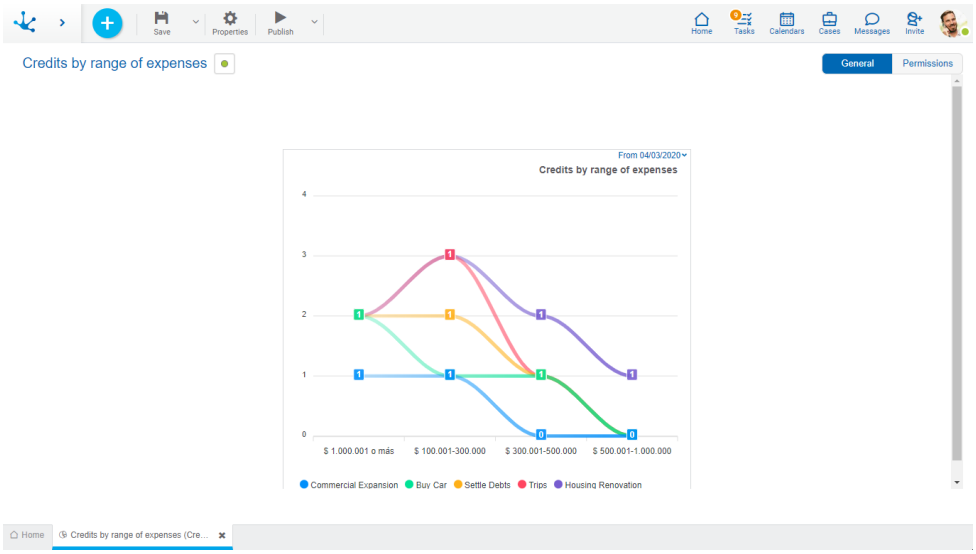
Allows to indicate whether the labels corresponding to the data series are displayed.

Curve

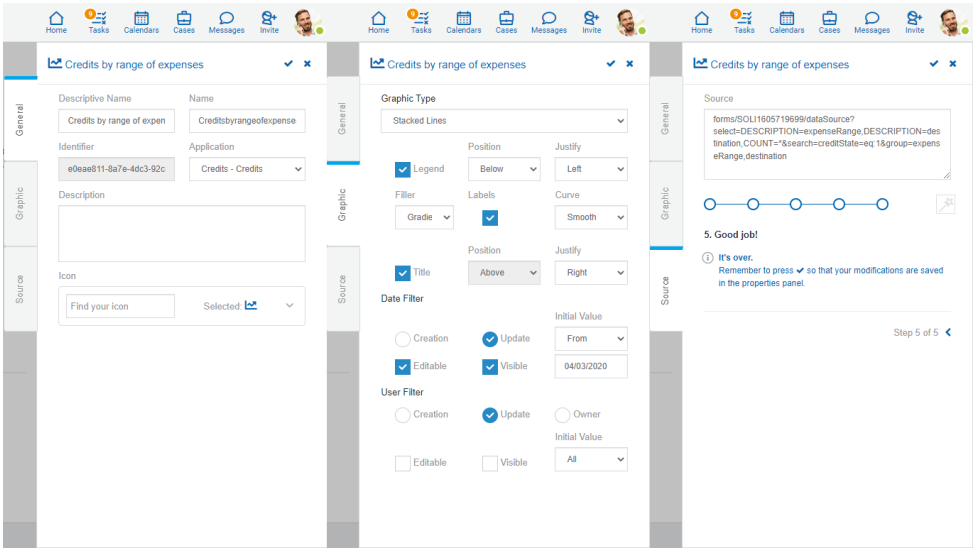
Indicates whether data lines are displayed as smooth, straight, or stepwise curve.

Example of Use

This stacked lines widget shows the number of credits approved by range of expenses during a selected period and for the users indicated in the filter.
The ranges of expenses are represented on the horizontal axis and the number of credits approved for each range on the vertical axis.
The height of the line for each destination indicates the total number of credits approved per rank.



The source code to model this chart is detailed.



Steps of the Modeling Wizard



The "Credit Request" form from which the data displayed in the chart is taken should be selected.



2. Define the data you want to display ✔

Series*

description(expensesRange), description(destination), count(ALL)

First the “expensesRange” field of the form visualized on the horizontal axis of the chart is indicated. Secondly, the "destination" field that corresponds to the height of the line is indicated. And lastly, the “count(ALL)” function is used to determine the number of loans per destination that are displayed on the vertical axis.

> *For fields that use value lists, the "description()" function must be used to display every list value with its descriptive text, otherwise the numerical code* is displayed.



3. Define display criteria

Select by

equal(creditStatus, 1)

A condition is added to display all requests with "creditState=1" where "creditState" corresponds to a value list with numerical code "1" and descriptive text "Approved".



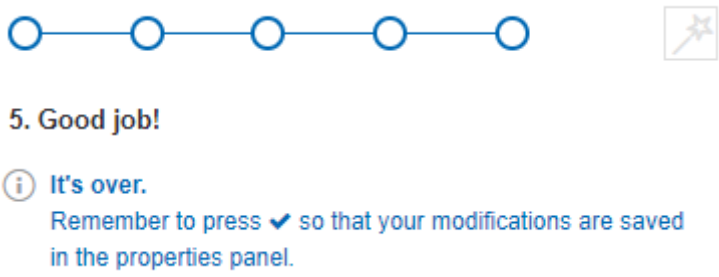
4. Define how to group the results

Group by

expensesRange, destination

Grouping the results by the same form fields that were modeled in step 2 is defined.

> *In this step, the names of fields must be used without the "description()" function, despite being value lists.*



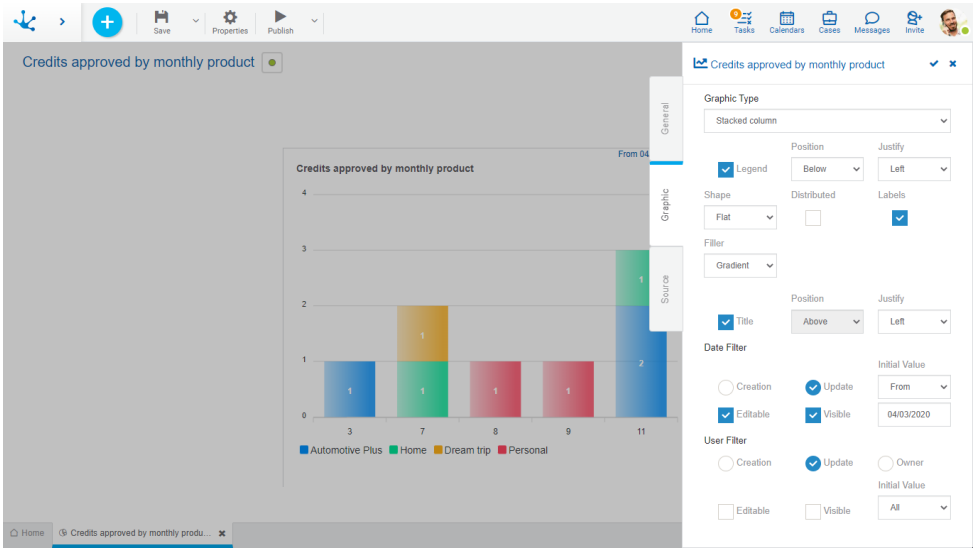
Step 5 reporting that all steps are met is completed .

3.4.15.2.2.4. Bars and Columns

The bar and column widgets, comprehend the charts of type: bar, grouped bar and stacked bar, column, grouped column and stacked column and N Top.

They are used to compare different data series.

To the properties shared by all the widgets are added those specific to bars and columns.



Properties

Legend

Indicates the incorporation of the data series description.
If this property is selected, it must also indicate:

Position

Select above or below.

Justify

If above or below position is selected, it allows to indicate right, center or left alignment.

Shape
Allows to select if the shape has a plane or rounded border.

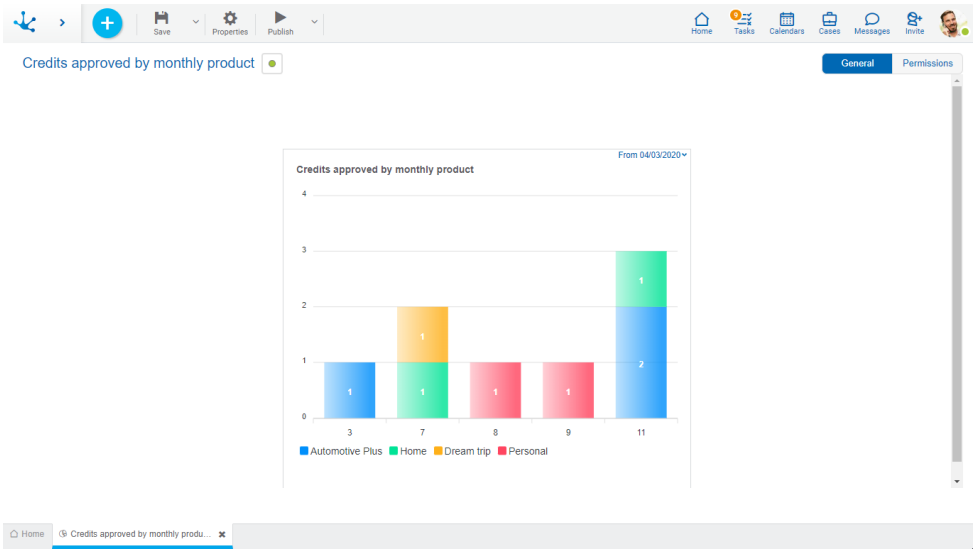
Distributed
Indicates whether the series are grouped or interleaved.

Labels
Allows to indicate whether the labels corresponding to the data series are displayed.

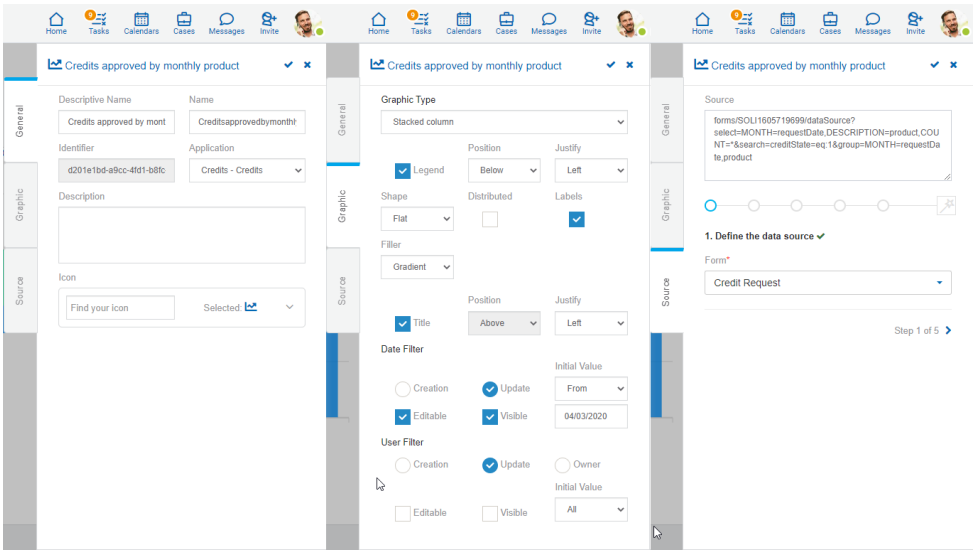
Filler
Allows to highlight the area of data series in a gradient or solid way.

Example of Use

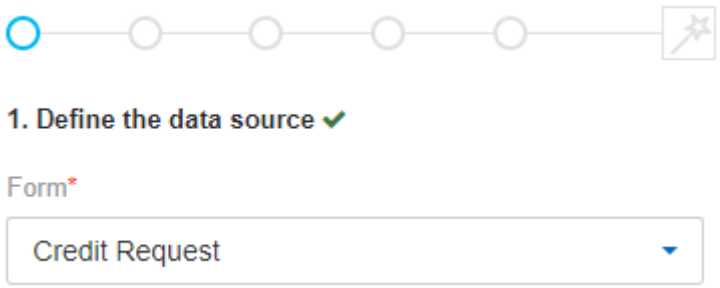
This stacked column widget shows the number of credits approved by range of expenses, during a selected period and for the users indicated in the filter.
The months are represented on the horizontal axis and the number of credits approved for each month is represented on the vertical axis
The percentage of each product is given by the colored surface, and the grouping of all surfaces represents the total number of approved credits.



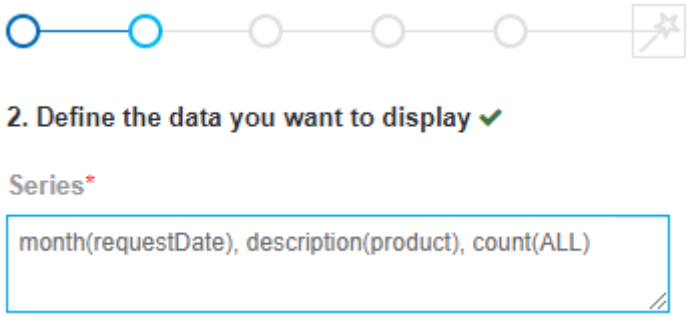
The source code to model this chart is detailed.



Steps of the Modeling Wizard



The "Credit Request" form from which the data displayed in the chart is taken should be selected.



First the “month(requestDate)” function is used to determine the month of the loan request that is displayed on the horizontal axis of the chart. Secondly the "product" field that corresponds to the height of the column is indicated. And lastly the “count(ALL)” function is used to determine the number of loans per product per month that are displayed on the vertical axis.

> *For fields that use value lists, the "description()" function must be used to display every list value with its descriptive text, otherwise the numerical code* is displayed.



3. Define display criteria

Select by

equal(creditStatus, 1)

A condition is added to display all requests with "creditState=1" where "creditState" corresponds to a value list with numerical code "1" and descriptive text "Approved".



4. Define how to group the results

Group by

month(requestDate), product

Grouping the results by the same form fields that were modeled in step 2 is defined.

> *In this step, the names of fields must be used without the "description()" function, despite being value lists.*



5. Good job!

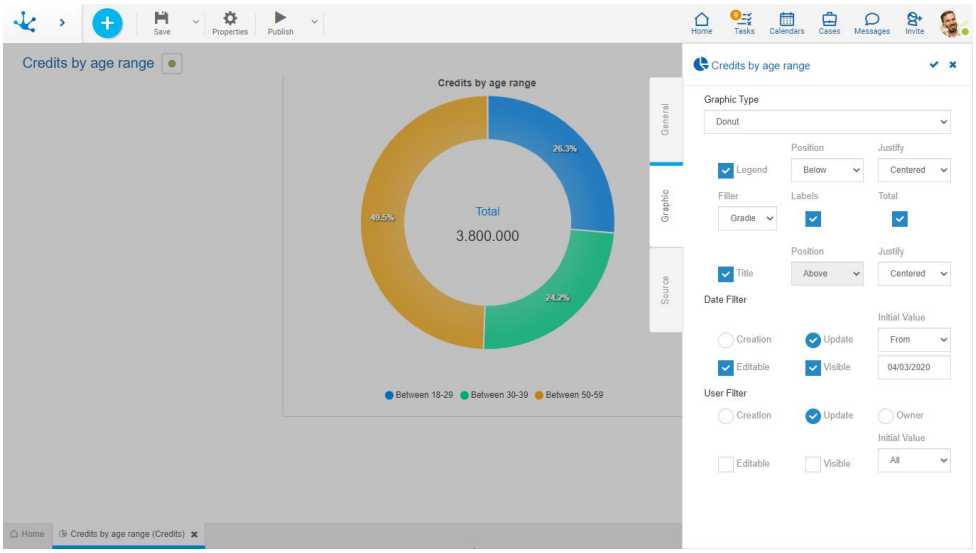
It's over.
Remember to press ✔ so that your modifications are saved in the properties panel.

Step 5 reporting that all steps are met is completed .

3.4.15.2.2.5. Pies and Donuts

Widgets of pies and donuts type are used to show parts of a whole. They represent the amount of segments in percentages and the total sum is equal to 100 %.

To the properties shared by all the widgets are added those specific to pies and donuts.



Properties

Legend

Indicates the entry of data series description.
If this property is selected, also indicate:

Position

Select above, below, left, right.

Justify

If above or below position is selected, it allows to justify right, center or left.
If right or left position is selected, the legend is displayed top right or left as indicated.

Filler

Allows to highlight the area of data series in a gradient or solid way.

Labels

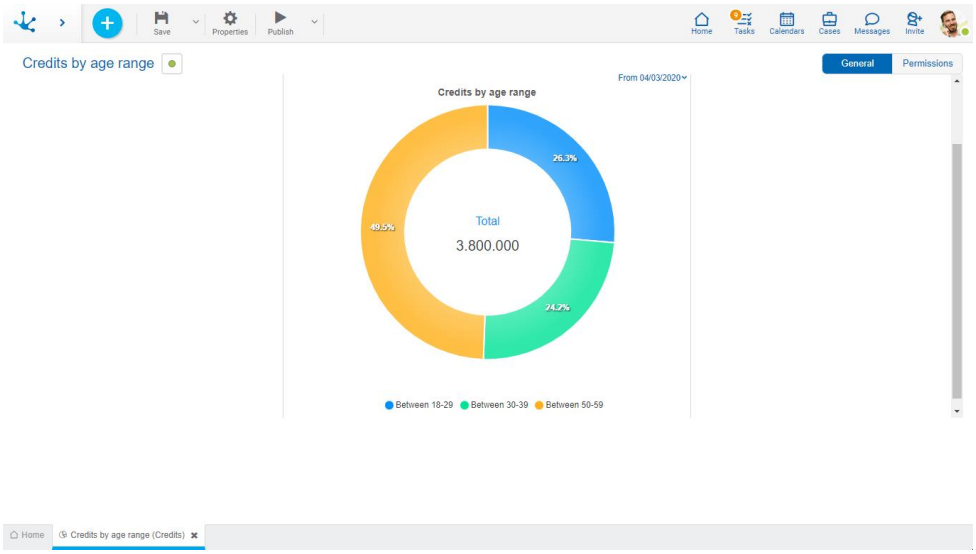
Allows to indicate whether the labels corresponding to the data series are displayed.

Total

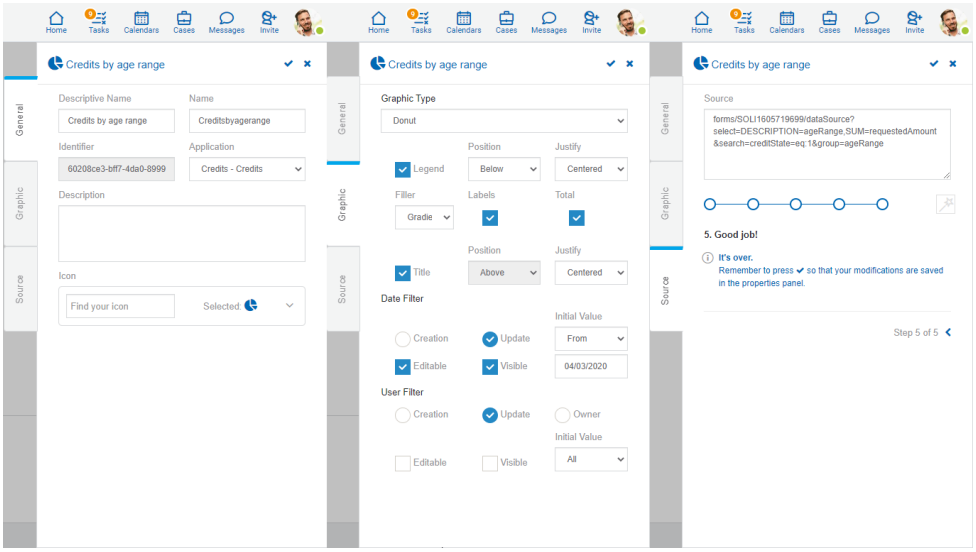
Indicates whether or not the total of all series is displayed.

Example of Use

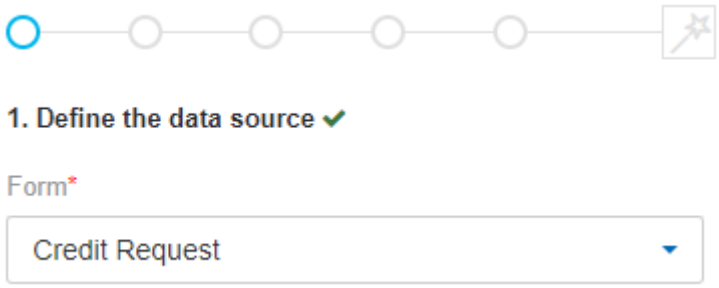
In this donut widget the total amount requested grouped by age range is visualized, for a selected period and to the users indicated in the filter.



The source code to model this chart is detailed.



Steps of the Modeling Wizard



The "Credit Request" form from which the data displayed in the chart is taken should be selected.



2. Define the data you want to display ✔

Series*

description(ageRange), sum(requestedAmount)

First the “ageRange” field of the form visualized on each chart portion is indicated. And lastly the “sum(requestedAmount)” function to determine the total amount of loans requested by age range is used.

> *For fields that use value lists, the "description()" function must be used to display every list value with its descriptive text, otherwise the numerical code* is displayed.



3. Define display criteria

Select by

equal(creditStatus, 1)

A condition is added to display all requests with "creditState=1" where "creditState" corresponds to a value list with numerical code "1" and descriptive text "Approved".



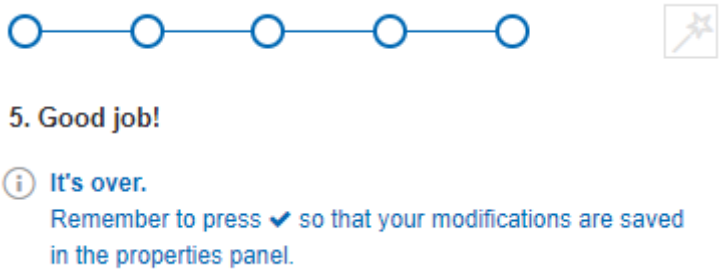
4. Define how to group the results

Group by

ageRange

Grouping the results by the same form fields that were modeled in step 2 is defined.

> *In this step, the names of fields must be used without the "description()" function, despite being value lists.*



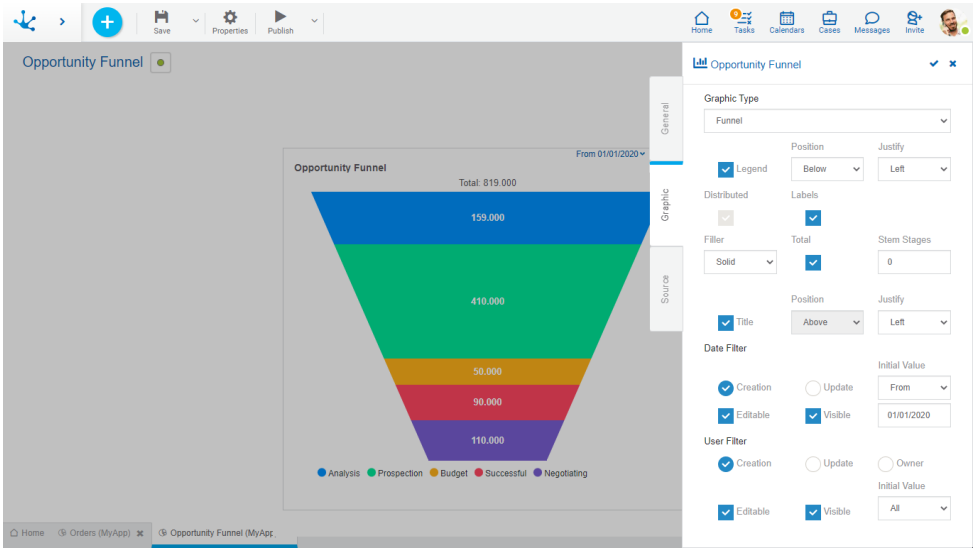
Step 5 reporting that all steps are met is completed .

3.4.15.2.2.6. Funnel

Widgets of funnel type show values in gradually decreasing proportions. In this type of charts the data is shown as a value for each stage.

To the properties shared by all the widgets are added those specific to funnels.

This chart type also incorporates stages modeling to identify each represented value and define its location.



Properties

Legend

Indicates the entry of data series description.
If this property is selected, also indicate:

Position

Select above, below, left, right.

Justify

If above or below position is selected, it allows to justify right, center or left.
If right or left position is selected, the legend is displayed top right or left as indicated.

Distributed
Indicates whether the series are grouped or interleaved.

Labels
Allows to indicate whether the labels corresponding to the data series are displayed.

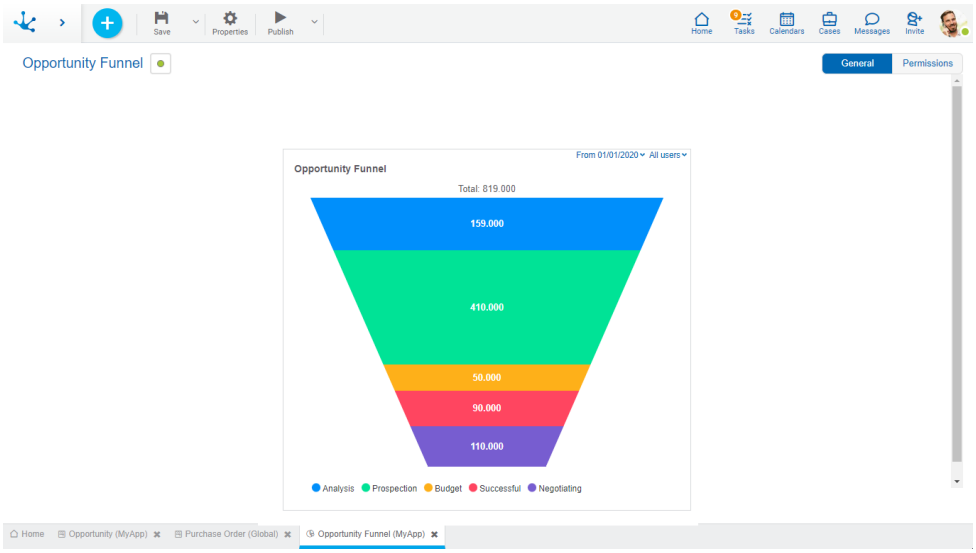
Filler
Allows to highlight the area of data series in a gradient or solid way.

Total
Indicates whether or not the total of all series is displayed.

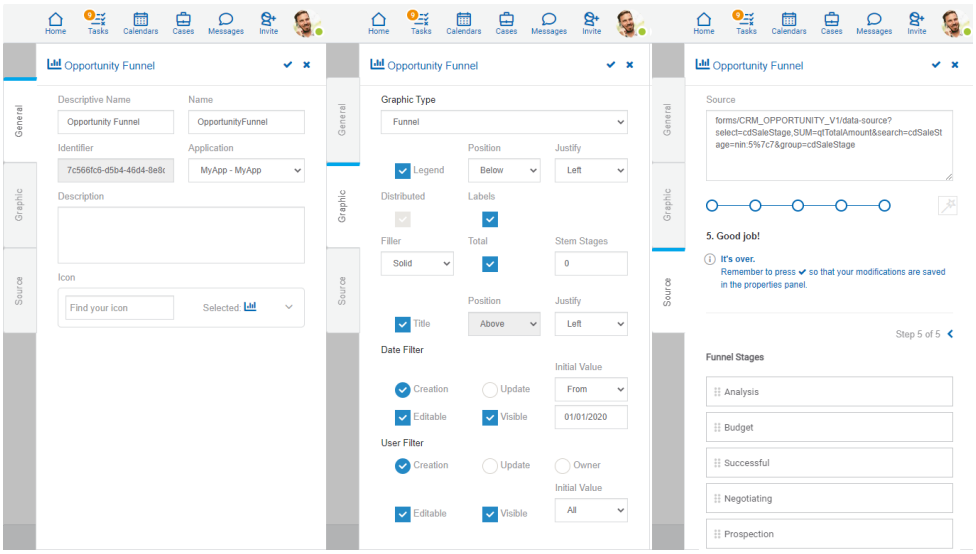
Stem Stages
Indicates the number of stages or bars that are displayed in the stem of the funnel.
If this property is zero the stages are part of the funnel and it does not have a stem.

Example of Use

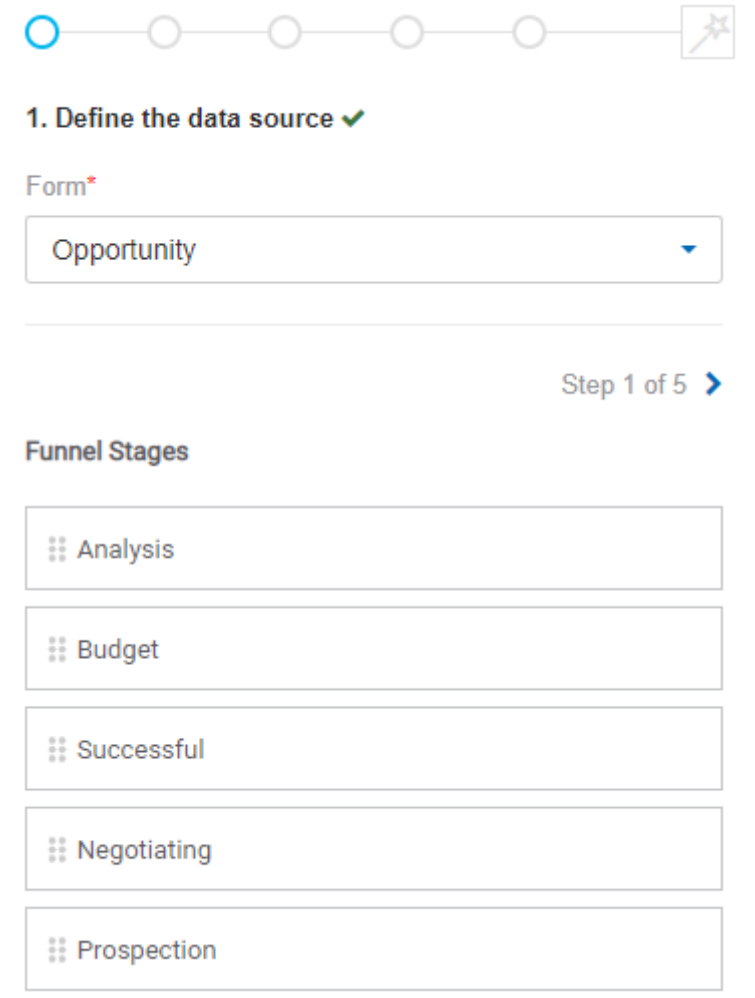
In this funnel widget the estimated total amount of CRM opportunities grouped by Sales stage is displayed, during a selected period and for the users indicated in the filter.



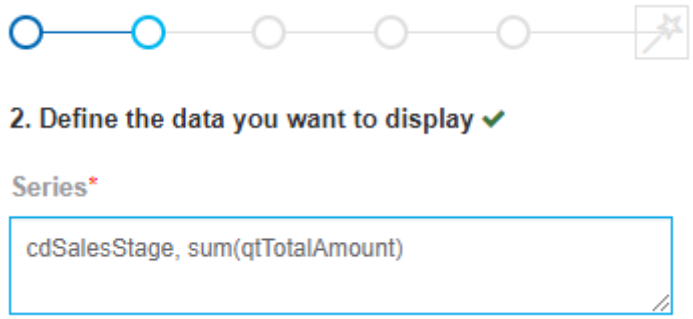
The source code to model this chart is detailed.



Steps of the Modeling Wizard

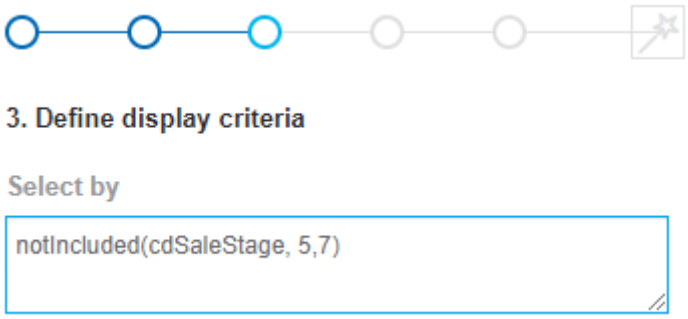


The "Opportunity" form is selected from which the data displayed in the chart is taken. And the order of the stages of the funnel is defined.

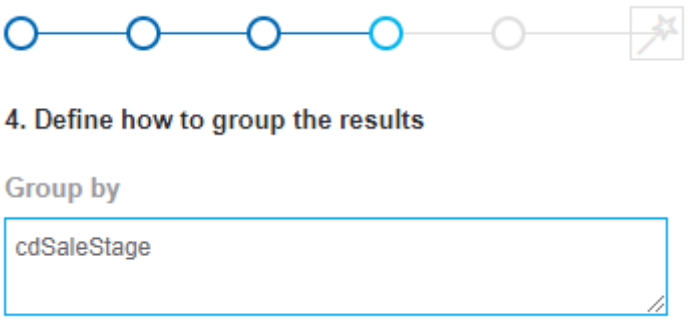


First the “cdSalesStage” field of the form visualized on each funnel stage is indicated. And finally the function "sum(qtTotalAmount)" is used to determine the total amount estimated for each stage.

> *In this step, the names of the fields must be used without the "description()" function, despite being values lists.*

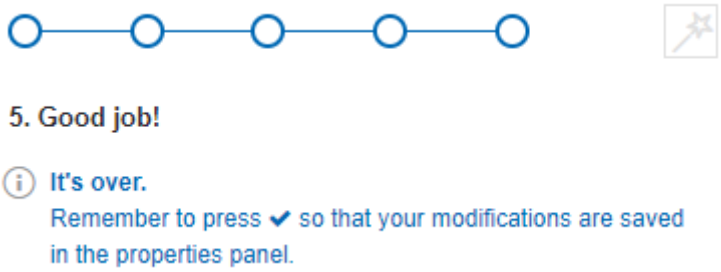


The condition to display all opportunities is added, excluding the stages with numerical code “5” and “7” and descriptive text “Lost” and “Stand-By”.



Grouping the results by the same form fields that were modeled in step 2 is defined.

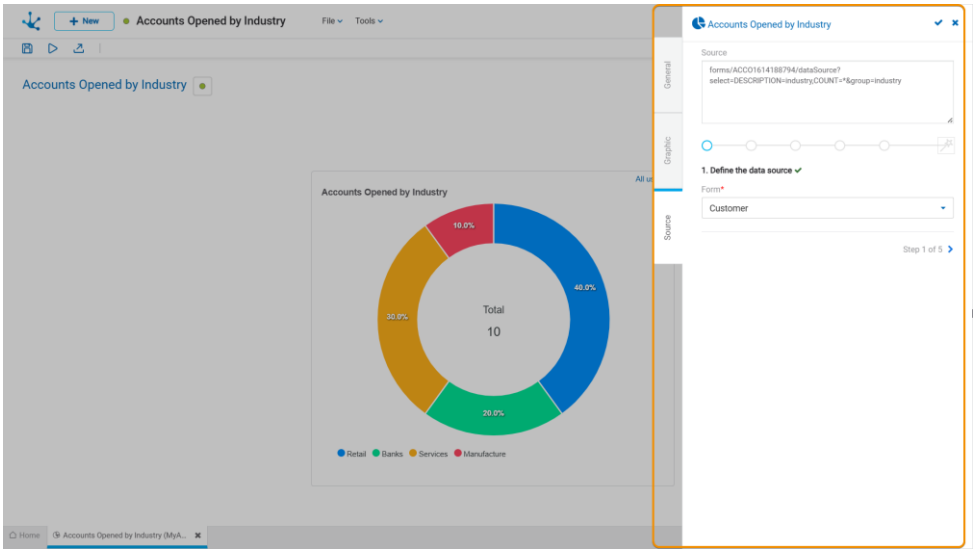
> *In this step, the names of fields must be used without the "description()" function, despite being value lists.*



Step 5 reporting that all steps are met is completed .

3.4.15.2.3. Source

The third tab of the side panel corresponds to the Rest API for obtaining the data of the selected chart and the modeling wizard for its automatic generation, following a series of steps.



Properties

Source

Corresponds to the URI on the Rest API to obtain the data displayed on the selected chart. It can be codified or automatically generated through the modeling wizard steps.

Wizard for Widgets Modeling

Allows to automatically generate the URI of the Rest API in the area of the source following the sequence of steps represented by the lines of circles.

Each step is identified by a number and accompanied by an informative paragraph, represented by the "i" icon.

The required steps are represented by the red circle and once they are completed they are displayed in light blue, indicating that the part corresponding to the source code has been completed. The circles corresponding to the steps not yet completed are displayed in grey.

The pre-established order must be respected and advance by selecting the next circle or by pressing the paging icon, located at the bottom of the wizard. The completed steps can be navigated through by pressing the circles or by using the paging icon.

Wizard Steps for Widgets Modeling

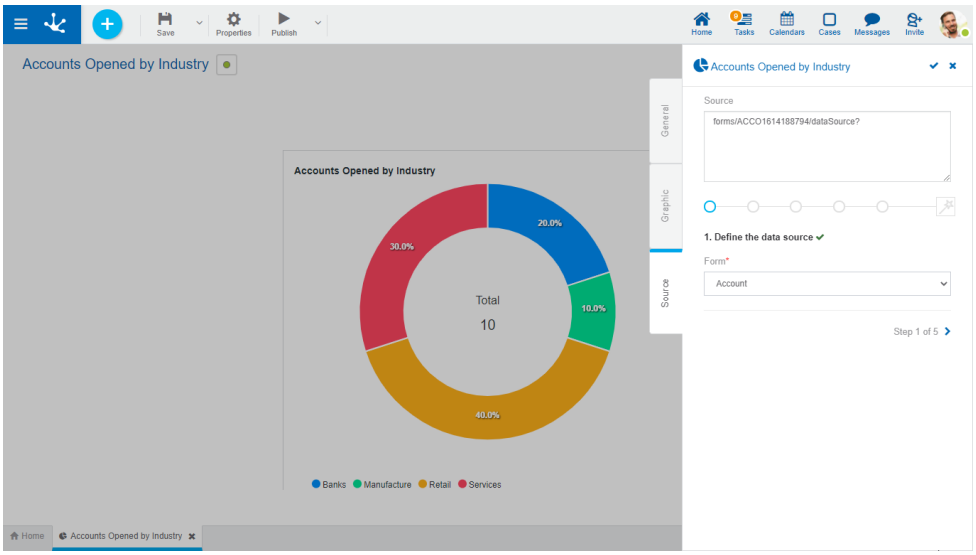
Step 1: Define the data source

This step is required.

Form

Allows to select the entity from which the data to be displayed on the chart will be taken. Once it is selected there is a check mark to the right of the step description, indicating that it was completed successfully. When finishing the step, the generated code is displayed on the source.

> *When expanding the available form list a search field is activated. As the user enters characters into it, the number of forms displayed is reduced to those whose names contain such characters.*



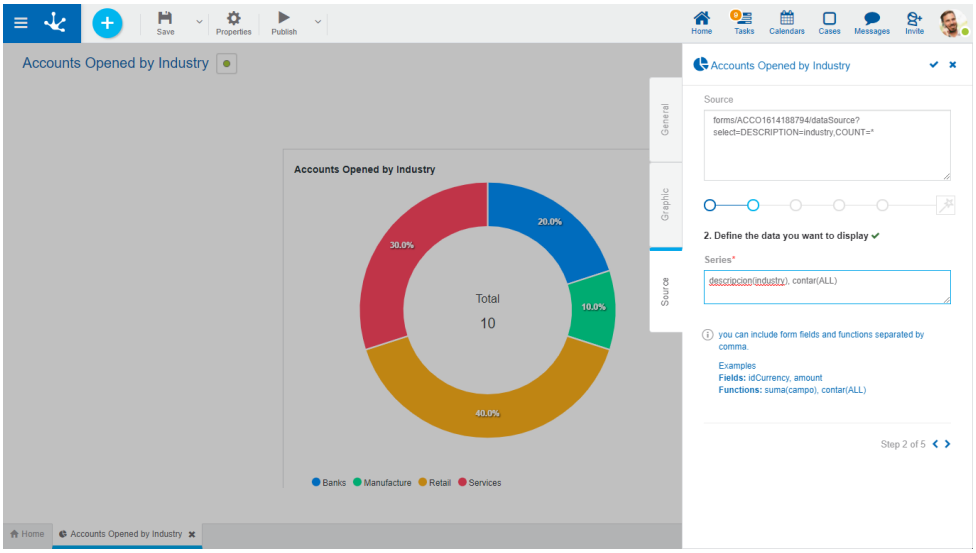
Step 2: Define the data you want to display.

This step is required.

Series

It must be indicated which elements are going to be displayed on the chart, being able to use the help of the wizard. Once the series are entered, there is a check mark to the right of the step description,

indicating that it was completed successfully. When completing the series, the added code is displayed on the source.

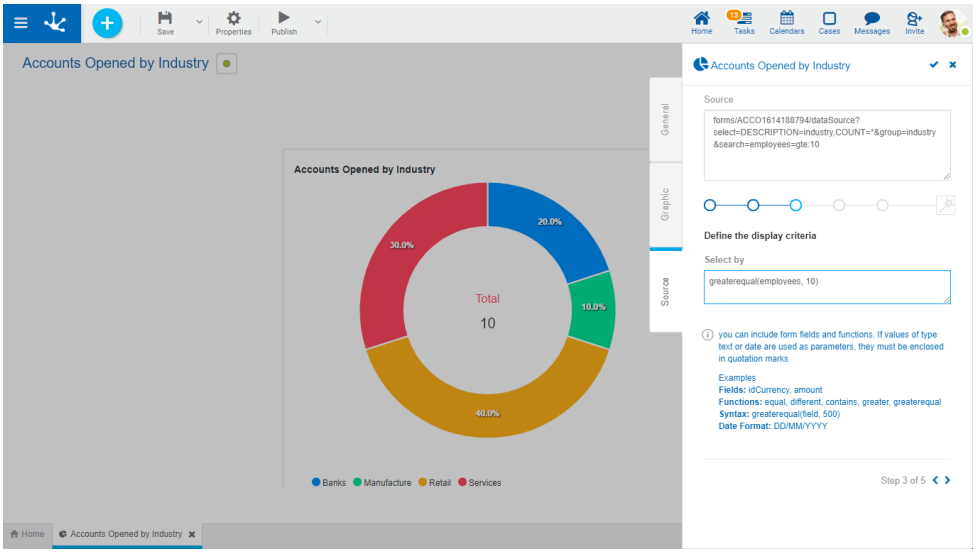


Step 3: Define the display criteria.

This step is optional.

Select by

Allows to indicate with the help of the wizard the elements that must meet a condition for the data to be displayed on the chart. When completing this step, the added code is displayed on the source.

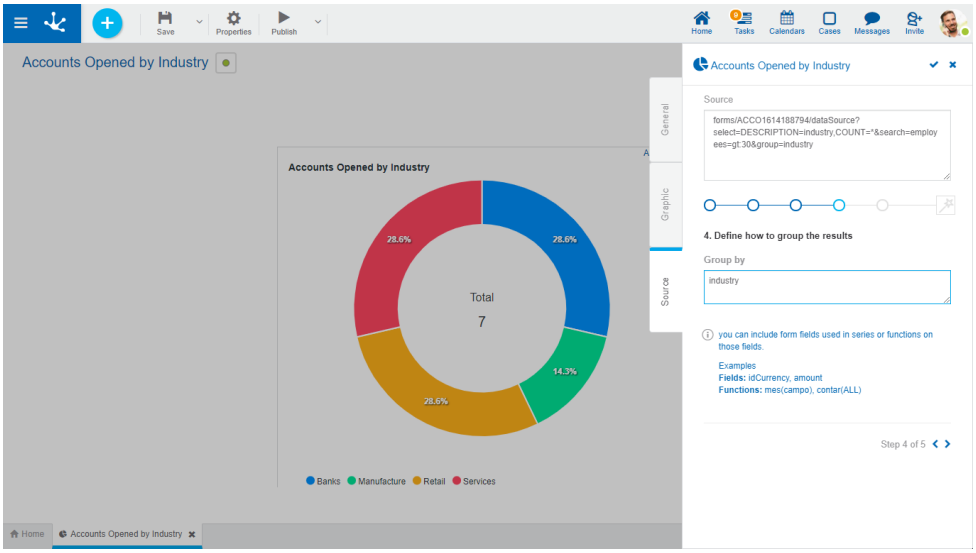


Step 4: Define how to group the results.

This step is optional.

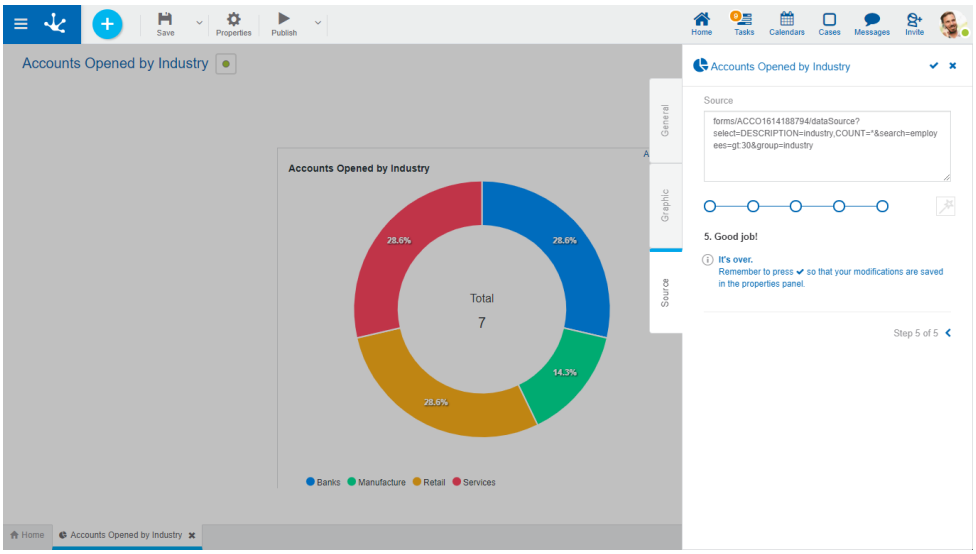
Group by

Allows to indicate with the help of the wizard the elements of the series, by which the results will be grouped. When completing this step, the added code is displayed on the source.



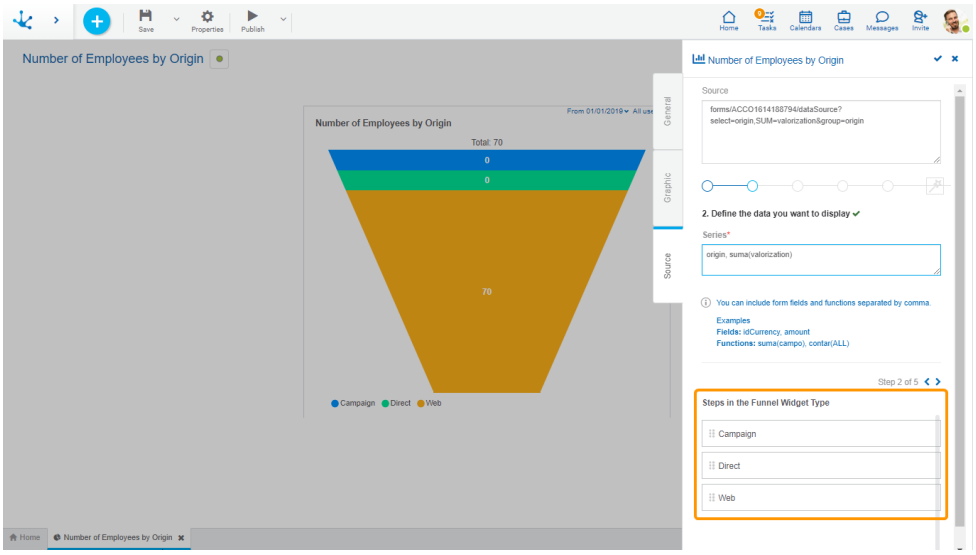
Step 5: Good job.

Informs that all the steps have been completed, and reminds that changes made on the widget properties panel must be saved.



Stages in the Funnel Type Widget

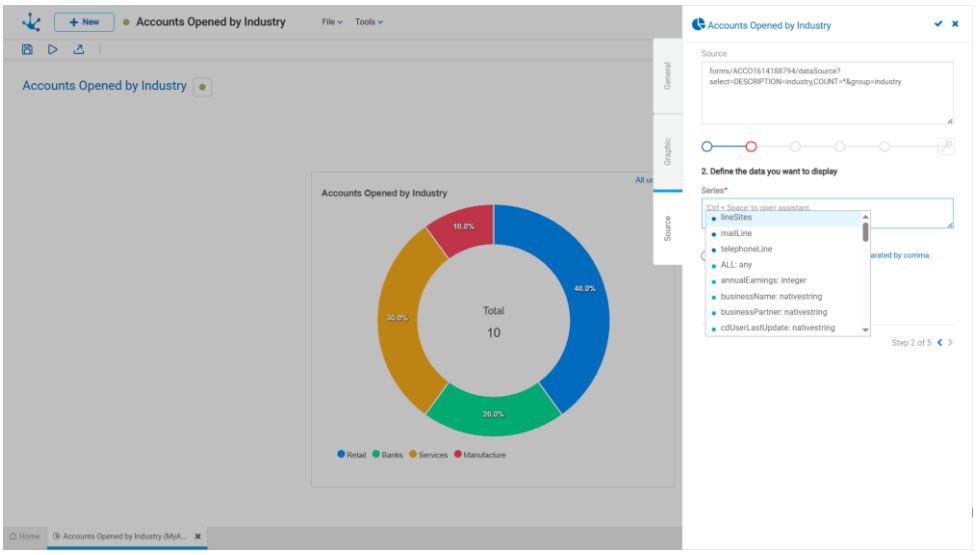
When modeling a funnel type widget a form must be selected where the value of one of its fields originates in a value list or a rule. Once the series field is indicated, the stages are displayed as a list in all the steps of the wizard, being able to sort or delete them.



3.4.15.2.3.1. Editing Wizard

In the editing area of each of the steps you can use a wizard for the automatic generation of the source.

Such wizard is activated pressing the "Ctrl + Space" keys, using the functionality of autocompleting, that is, as text is entered in the wizard, only are shown the elements which names start with the text the user writes.
In case of not using autocomplete the complete list of elements available for the step you are working on opens.



3.4.15.2.3.2. Elements Used in the Wizard

Steps can be composed of different types of elements, separated by a ".".

- Form fields

The form fields are used by identifying them through the property Name . In the wizard they are displayed with ● on their left.

- Iterative containers
  The container Name property is shown. Whenever an iterative container is used, indicate the field separated by a ".". In the wizard they are displayed with ● on their left.
  Example: Items.Quantity

- Functions
  In the wizard they are displayed with ● on their left.

In all cases, the data type they represent is indicated on the right, and in the case of functions, the data type they receive as a parameter is shown.

In the event of a syntax error, a message indicating the problem is displayed.

Functions in the Wizard

These functions are displayed in the wizard, with a brief description of each function and suggested steps in the wizard.

Function	Description	Wizard (Step)	Return Type
count(field)	Counts the occurrences of a given field.	Series and grouping	Numeric
descrip-tion(field)	Returns the description of the field.	Series	Text
max(field)	Returns the maximum value.	Series and grouping	Field type used
month(field)	Month of the year. It is used for date type fields.	Series	Numeric
min(field)	Returns the minimum value.	Series and grouping	Field type used
average(field)	Returns the average value.	Series and grouping	Numeric
Add(field)	Adds the entered field values.	Series and grouping	Numeric Deci-mal
equals(field, field or value)	Compares the first field with another field or value for equality.	Display criteria	Boolean

Function	Description	Wizard (Step)	Return Type
distinct(field, field or value)	Compares the first field with another field or value by inequality.	Display criteria	Boolean
contains(field, value)	Checks if the field contains a certain value.	Display criteria	Boolean
startsWith(field,value)	Checks if the field starts with a certain value.	Display criteria	Boolean
doesnotStartWith(field,value)	Checks if the field does not start with a certain value.	Display criteria	Boolean
greater(field, field or value)	Checks if the first field is greater than the second field or entered value.	Display criteria	Boolean
greaterEqual(field, field, or value)	Checks if the first field is greater than or equal to the second field or entered value.	Display criteria	Boolean
less(field, field or value)	Checks if the first field is less than the second field or entered value.	Display criteria	Boolean
lessEqual(field, field, or value)	Checks if the first field is less than or equal to the second field or entered value.	Display criteria	Boolean
included(field, value1, value2, ...)	Checks if the field value exists in the comma-separated list of values.	Display criteria	Boolean
notIncluded(field, value1, value2, ...)	Checks if the field value does not exist in the comma-separated list of values.	Display criteria	Boolean

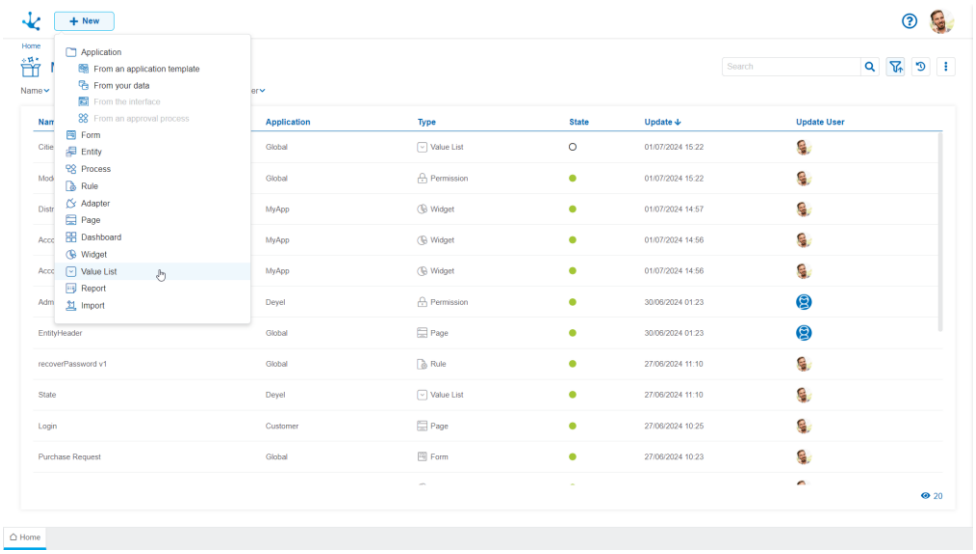
Operands

To add a condition of the type “contains value” or “does not contain value” without specifying a specific value, the operands 'NOTNULL' and 'ISNULL' can be used, respectively.
Example: Evaluates whether the actualAuthorizationDate field contains no value, or whether it has a value assigned.

equals(actualAuthorizationDate, 'ISNULL')
equals(actualAuthorizationDate, 'NOTNULL')

3.4.16. Value Lists Modeling

The value list modeler allows to define a set of possible values for a field, grouped under some criteria. These values can be associated to entity fields, page fields and form fields.



The value list modeler allows to administer each list, respecting their progress through the different states of a modeled object.

- Save
- Publish
- Delete
- Export
- Delete Publication

A new value list can be created from:

- The button  , from the option  Value Lists.
- The entities modeler, in the “Relation” type property of a field, selecting the value lists option.
- The form modeler, in the “Relation” tab of a field selecting the value list option.
- The form modeler, through the control Value List.

The general characteristics of the list of values modeling environment are described in the topics:

- Modeling Facilities
- Value Lists Properties

3.4.16.1. Modeling Facilities

General characteristics of this modeler are specified below.

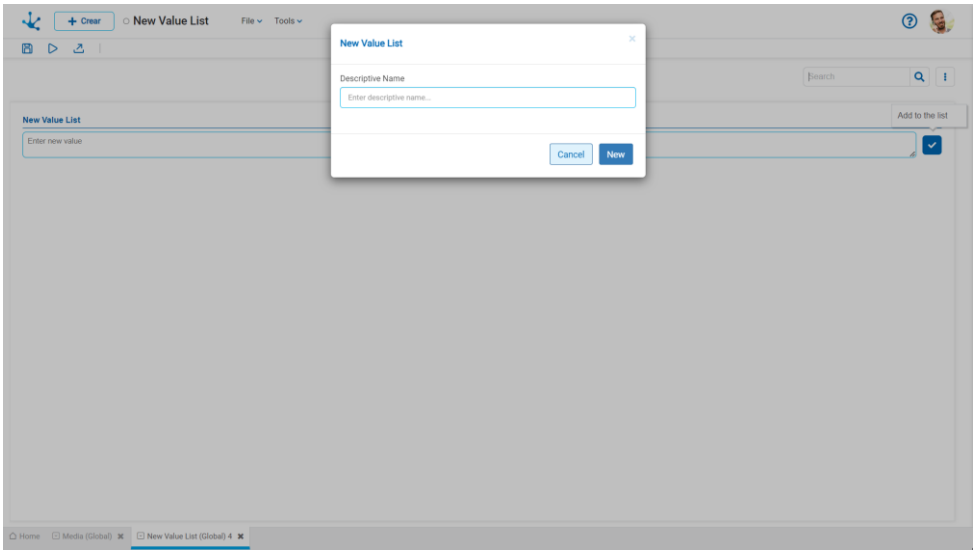
New Value List

The modeler user can design a new value list, available to be used from any form after being published.

Steps for Creating a New Value List

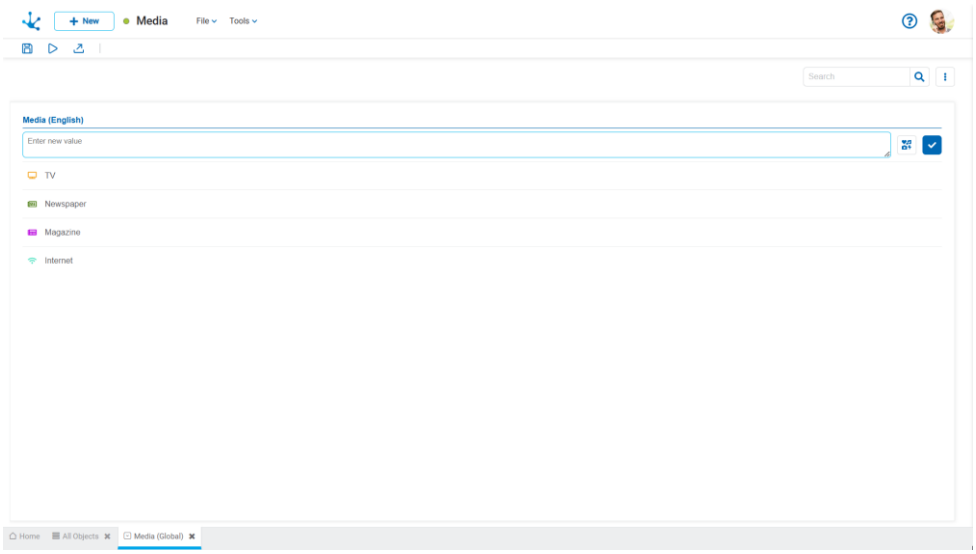
Step 1

Assign the name and press the "Create" button.



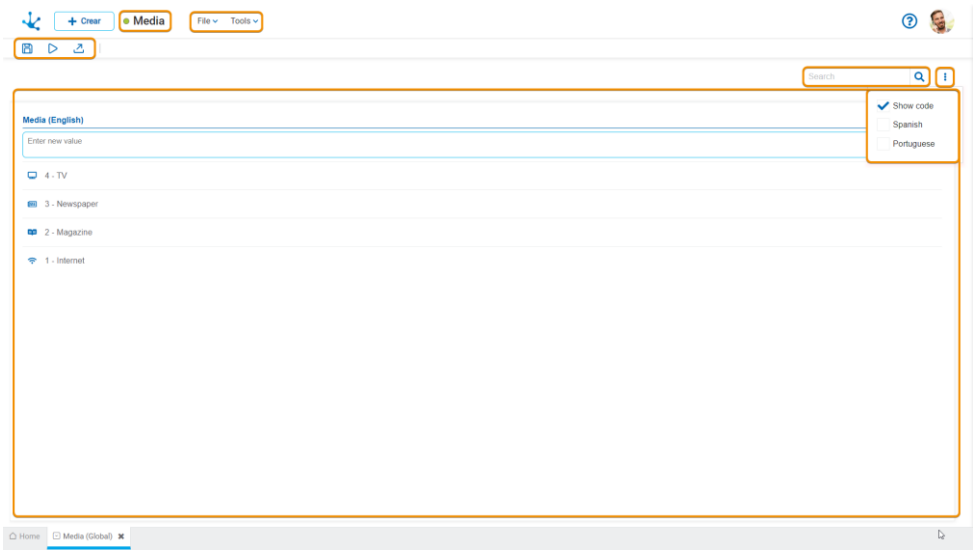
Step 2

Enter a new value and press the accent mark icon or the enter key. Repeat this step until all the desired values are entered.



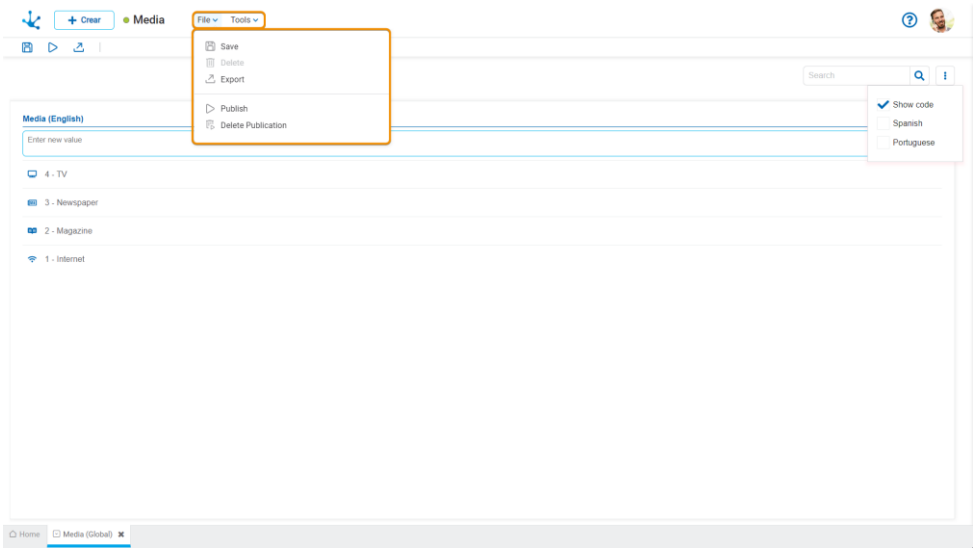
Workspace Sections

- Value List Information
  - State
  - Name
- Expanded Menu
- Top Toolbar
- Allows to filter values from the list based on the characters entered. If a list is very long it helps users to easily visualize the desired values.
- Enables an option that allows to visualize the internal code from the list values. If the property Internationalize is active the available languages are enabled. When selecting a language a column is added for the entry of the value list translations.
- Modeling Area



3.4.16.1.1. Expanded Menu

It is a horizontal list of options that contains vertical submenus with different operations on the value list or its modeling. Additionally, each submenu option can expand a dependant submenu.

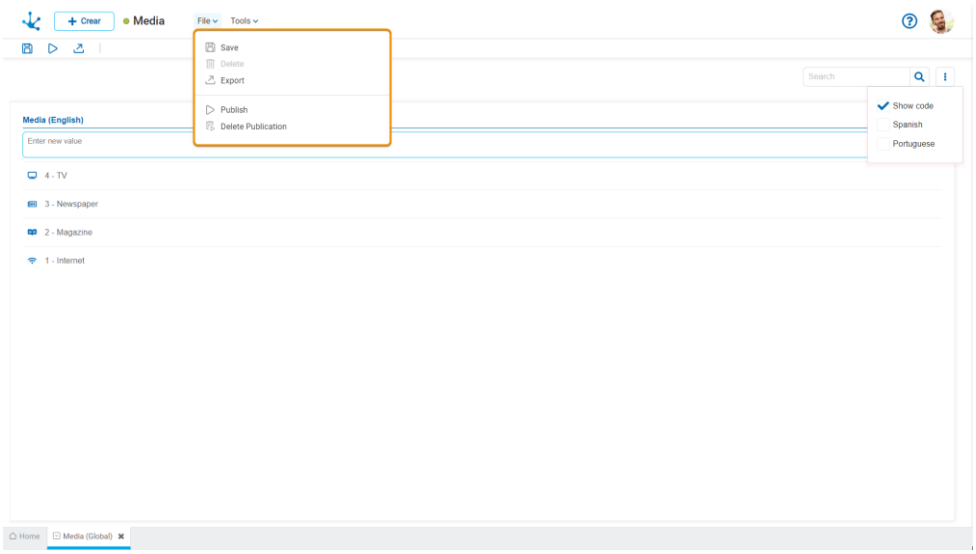


The expanded menu is made up of:

- Submenu File
- Submenu Tools

3.4.16.1.1.1. Submenu File

This submenu is opened by clicking the “File” option and it allows to make operations on the value list.



Save

This icon allows to save the object in the repository of **Deyel**, leaving its state as "Draft" or "Modified". If certain conditions are met, the modeler user receives a message indicating that the operation was performed correctly, otherwise they receive an explanatory message.

Conditions

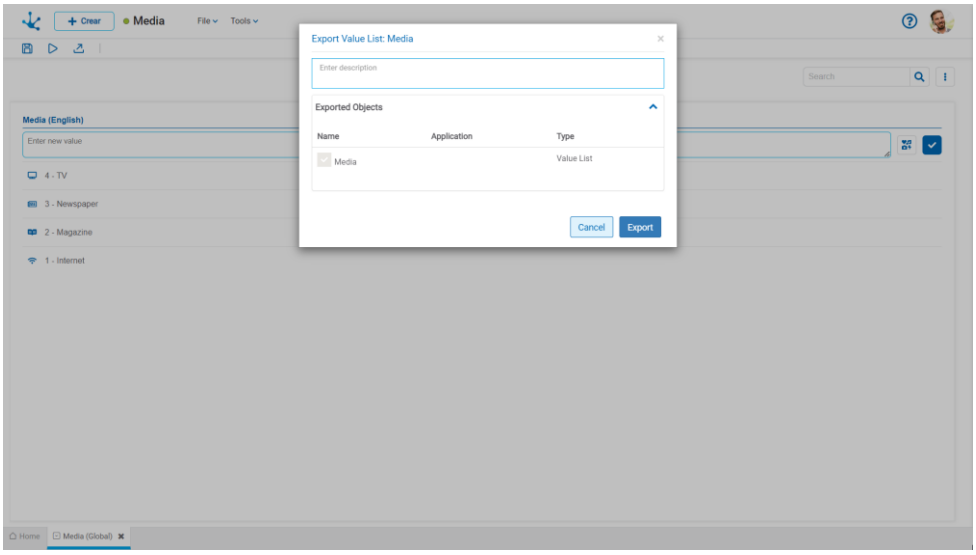
- There must be an object application.
- The name in the application must be unique.
- The object permissions must exist if the user selects the Models Security property in the “General” tab of the properties panel.

Delete

It allows to delete the object only if it is in "Draft" state and does not have other associated objects of **Deyel** that were previously saved or published. When deleted, the tab it is on closes and the object is removed from the modeler’s grid.

Export

This icon opens a window for the user to select and confirm the export of the object.



Description

In this property a text explaining the reason for the operation can be entered.
This text can be modified upon import and is displayed in the description column of the export record.

Exported objects

By expanding the container, the object being exported is displayed.

Press the “Cancel” button to leave the export without effect or the “Export” button to finish.

Publish

Through this icon the object changes to “Published” state and the modeler user receives the corresponding message, indicating the result of the operation. The condition for publishing is the same as for the "Save" operation.

Delete Publication

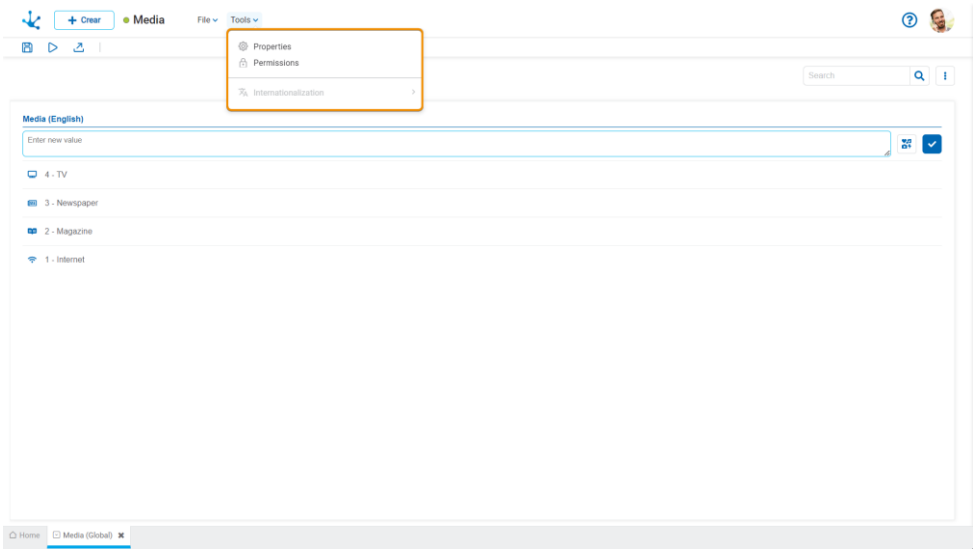
This icon allows to remove the use value list by returning it to the state “Draft”, in addition to deleting the data.

Publication History

This icon allows managing versions of published objects, displaying their history, to have better control of the changes that each object has.

3.4.16.1.1.2. Submenu Tools

This submenu is opened by clicking on the “Tools” option and it allows to model value list properties, their permissions and the internationalization.



Properties

Opens the value list properties panel.

Permissions

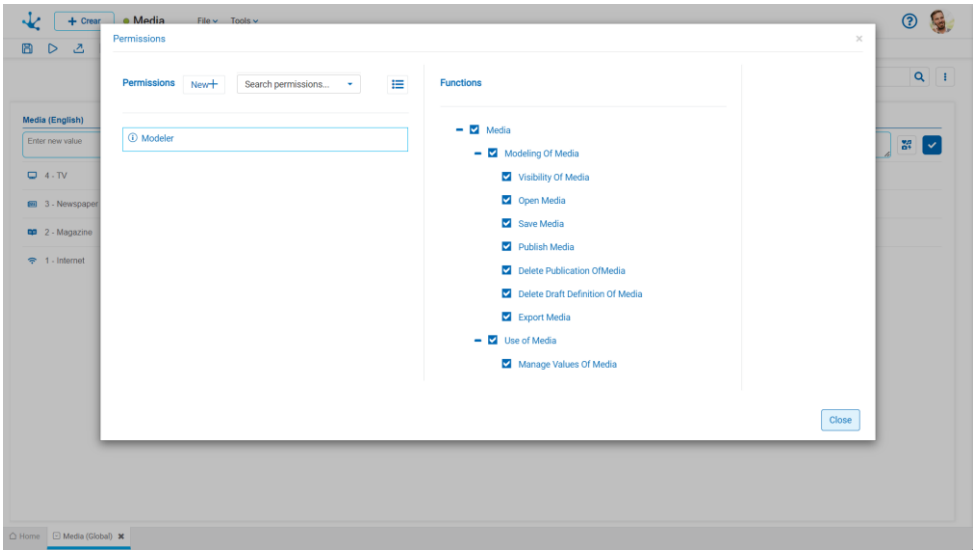
Allows to assign the security functions for use and modeling of the object to the existing permissions or to new permissions that the user creates, without having to go to the permission settings option from the menu.

Sections

- Permissions: Permissions to which object functions are assigned.
- Security Functions: Represents the total set of security functions, modeling and use of the object. Those that are marked are the ones included in the selected permission.

By default, all security functions for a new object are assigned to the permission Modeler.

Users who are assigned the permissions have access to the functions included in it.



New Opens a panel to create a new permission and once created, the security functions included in it must be selected .

Search permissions... Allows to select a permission from a list and enables the input of characters to filter the values in the list.

Opens the wizard to select a permission and once chosen, the necessary object security functions must be checked.

To unrelate a permission from the object, hover the cursor over the permission and press the icon . If there are functions selected for that permission, they must be unchecked in order to delete it.

Modeling Security Functions

- Visibility: Allows to display the object in the Deyel modeler grid.
- Open: Allows to open the object from the Deyel modeler.
- Save: Enables the operation of saving modifications made to the object.
- Publish: Enables the operation of publishing the object leaving its state as "Published".
- Delete publication: Enables the operation of deleting the object publication leaving its state as "Draft".
- Delete draft definition: Enables the operation of deleting the object.
- Export: Enables the operation to export the object from the tools submenu of the expanded menu.

.

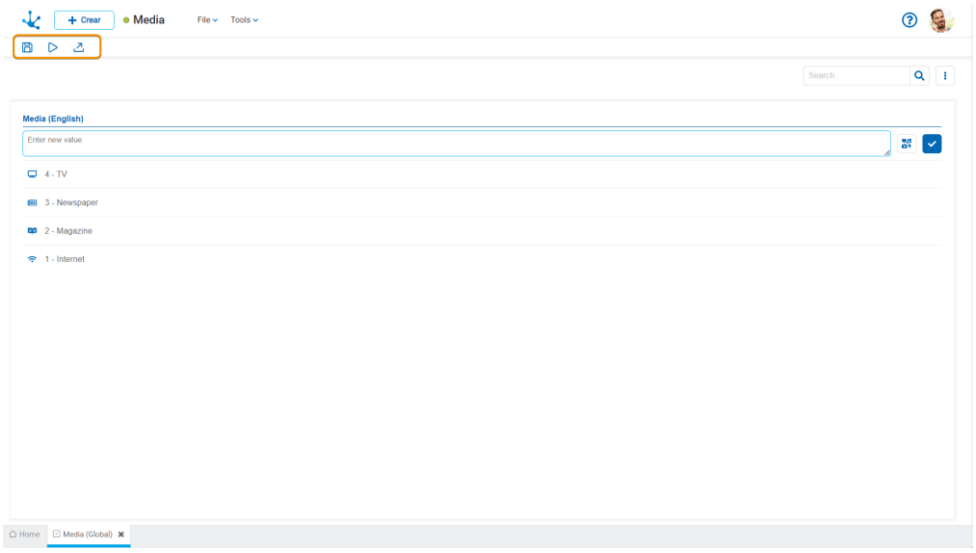
Use Security Functions

- Manage Values Enables the operation to enter, modify or delete values in the list.

> *The design "Permissions" option is only displayed if the property Models Security has been selected.*

3.4.16.1.2. Top Toolbar

Contains icons for the quick access to the most used operations from the expanded menu.



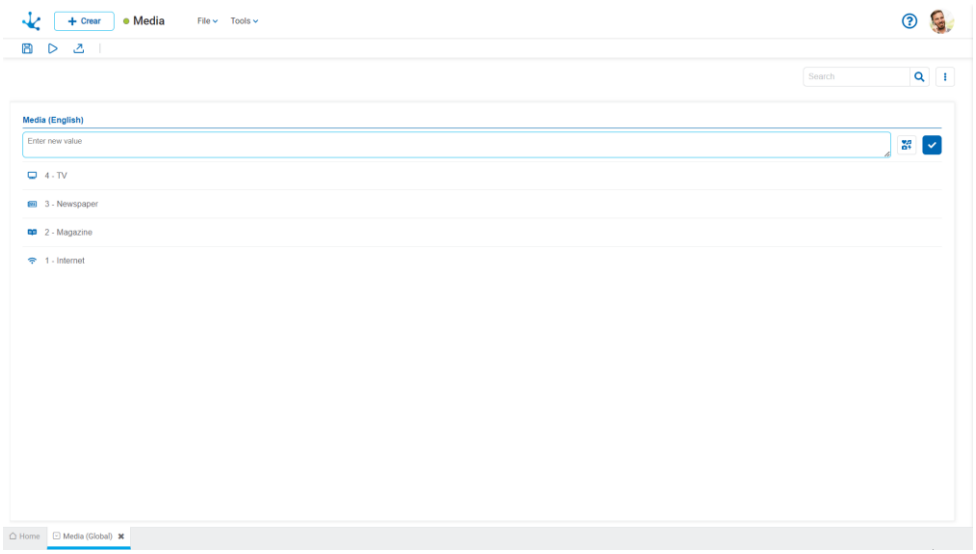
File

- Save
- Publish
- Export

3.4.16.1.3. Modeling Area

The modeling area is the section where the values can be entered, modified or deleted from the lists.

Once a list of values is published, the updates to its values are reflected in the fields that use them, both in the modeling and in the execution of pages, entities and forms. Such updates are also reflected on the value list configuration.



Operations on Values

- Allows adding each entered value to the list of values.
- It is displayed if the list has the Icons property modeled. It allows to associate icons to the list values.

- Double click: Allows to modify a value in the list.
- Move: Allows to change the position of a value within the list by dragging the value with the mouse.

Hovering the cursor over each of the values entered, a set of icons is displayed and this allows to perform different operations.

- Allows to delete a value from the list of values. Once deleted, it is displayed in gray and crossed out.
- Allows to restore a previously deleted value.

Display the Selected Line

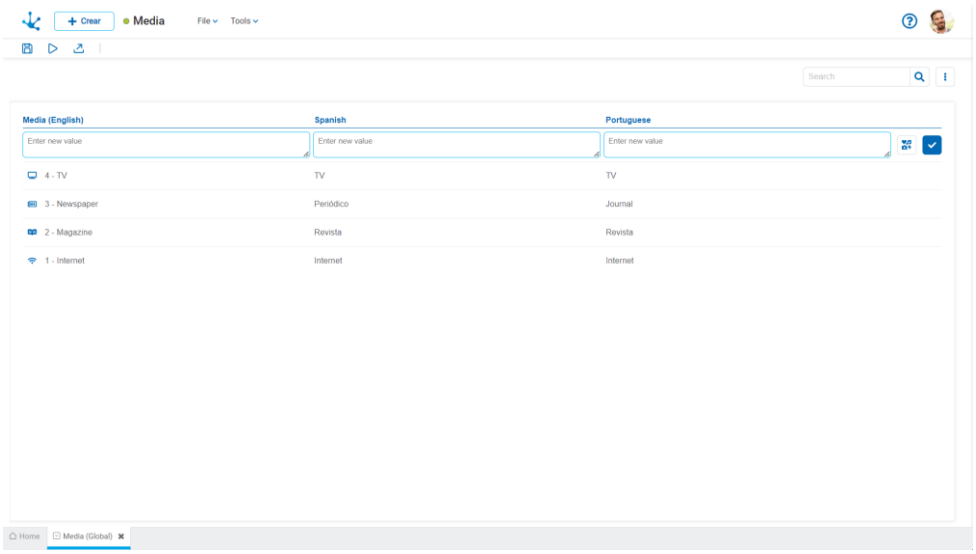
- Hides the icons that are displayed.
- Shows hidden icons.

Values Internationalization

If the value list has the Internationalize property active, the language configured for the user is displayed in the header of the list. For each language selected from the icon a new column is added.

The user can load the corresponding translation for each value.

Removing the check mark of a language deletes the column corresponding to that language.

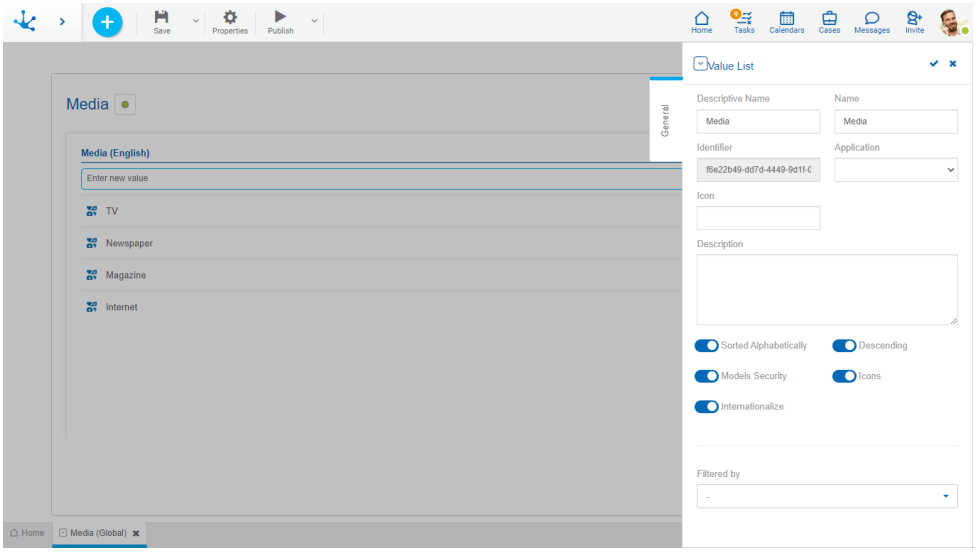


> *Once the internationalization has been modeled, it is suggested to load values to the list in all the selected languages.*

3.4.16.2. Value List Properties

The properties of the value list can be entered both at the time of their creation and when modifying an existing one.

Entering the value list properties panel is done using the icon  which is in the Top Toolbar.



General Tab

Properties

Descriptive Name

It is the name used by the user to reference the value list.

Name
It is used internally to reference the value list. No blanks or special characters allowed. It is unique and required.

Identifier
It identifies unequivocally the value list. It is generated automatically.

Application
Allows to define the application to which the object belongs. If no application is reported, the object is assigned to the application "Global".

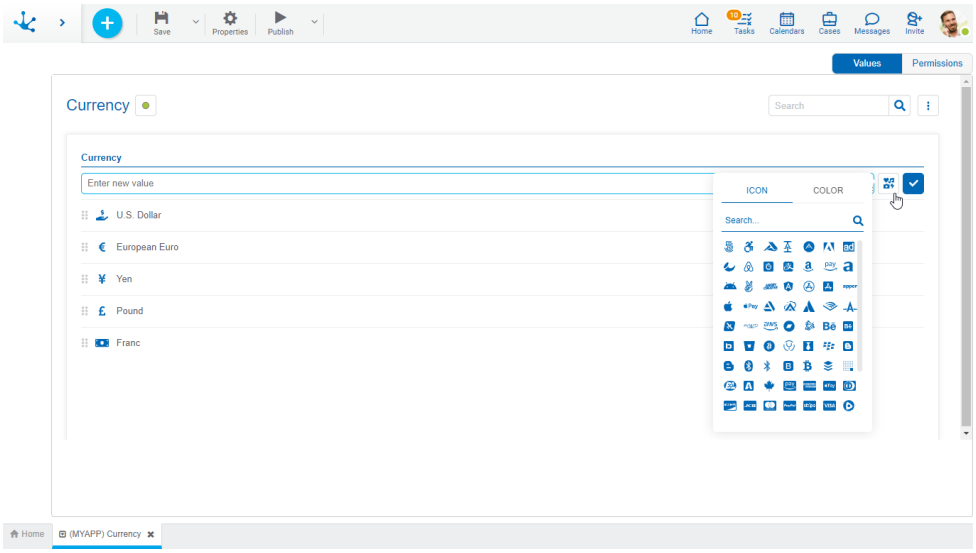
Description
Text which defines the value list and its respective content.

Sorted Alphabetically
Checking this property indicates that the values in the value list are sorted alphabetically, and allows to select if the order will be descending.

Descending
Indicates that the order of the values in the value list is descending. This property can be selected if it was previously indicated that the list is sorted alphabetically.

Models Security
Allows to add security to the value list, creating the security functions of modeling and of use to be assigned in the design option "Permissions" or in the permission configuration.

Icons
Allows to associate an icon to each value list when such values are incorporated.



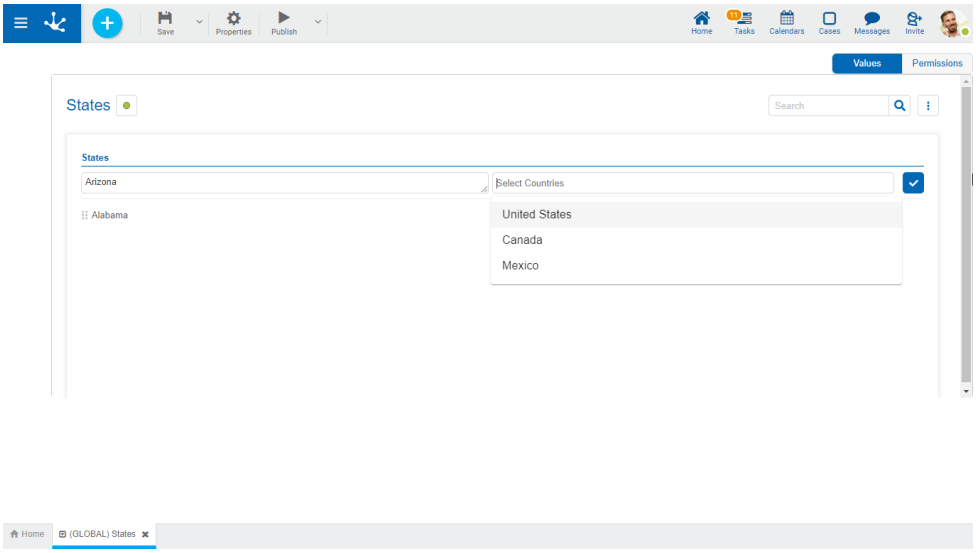
Internationalize

If this property is deactivated, the value list can be used in one language only. If activated values can be entered in different languages available in **Deyel**.

- When activating it:
  The property Language of Existing Values is enabled to select the language in which the values were entered or in which they will be entered if the list were new.
- When deactivating it:
  The property Language of Existing Values is enabled to select the language in which the value list are going to be kept.

Filtered by

Allows to select the value list to be related to the list being modeled. When a value is entered in the modeled list its corresponding is indicated in the list selected in this property.
Example: If the modeled list it Provinces and the list selected in the property Filtered by is Countries, when entering a province you must indicate which country it corresponds to.



Actions

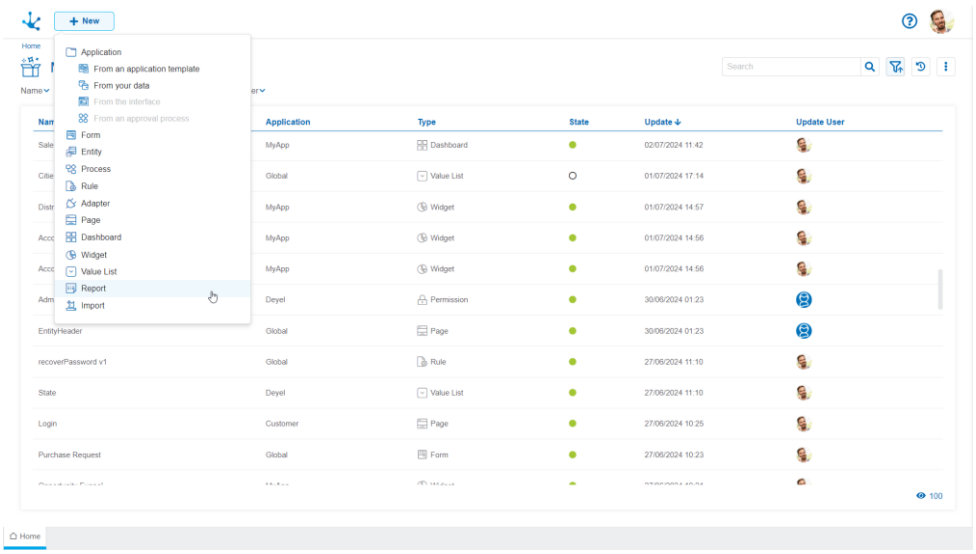
The icon ✔ is used to confirm the modifications made in the properties panel.

The icon ✖ is used to close the properties panel, if it was not previously saved, changes are discarded.

3.4.17. Reports Modeling

The report modeler allows to easily and intuitively model application reports in order to get detailed information on one or more related entities.

Each business user can design their own reports from the portal, using custom information and their preferred output format.



New This button is used to create a report from the option Report.

The general features of the report modeler and the elements that compose it are described in the following topics:

- Modeling Facilities
- Report Properties

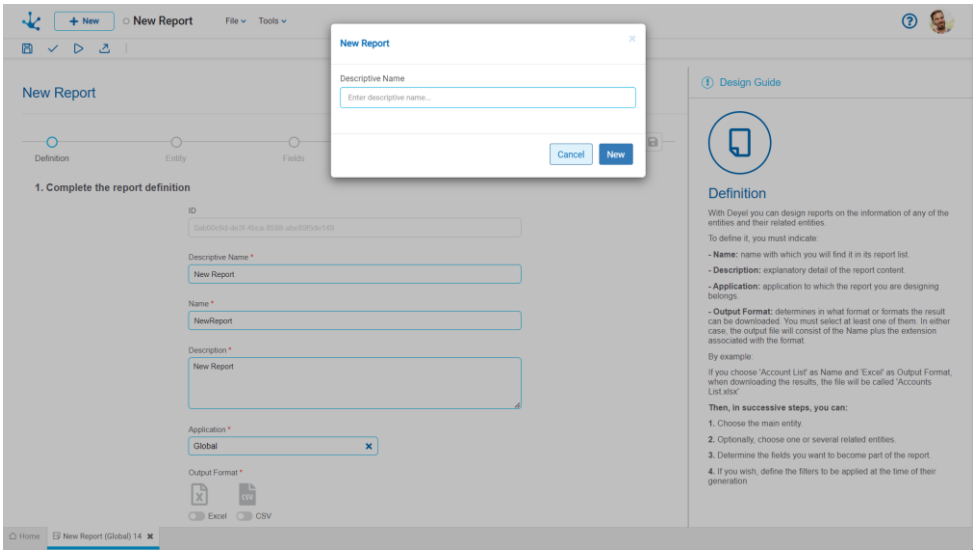
3.4.17.1. Modeling Facilities

General characteristics of this modeler are specified below.

New Report

The modeler user defines a new report, which after being published is available to be used in the portal by business users.

When the report is created, a panel opens and allows to enter its name.

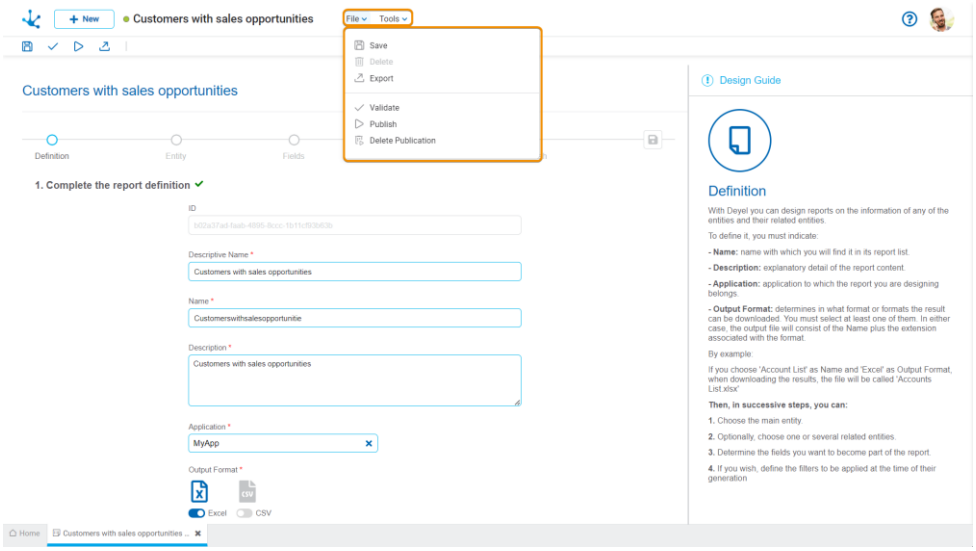


Workspace Sections

- Report Information
  - State
  - Name
- Expanded Menu
- Top Toolbar
- Modeling Wizard

3.4.17.1.1. Expanded Menu

It is a horizontal list of options containing vertical submenus with different operations on the report or on its modeling. In turn, each option in a submenu may expand a dependent submenu.

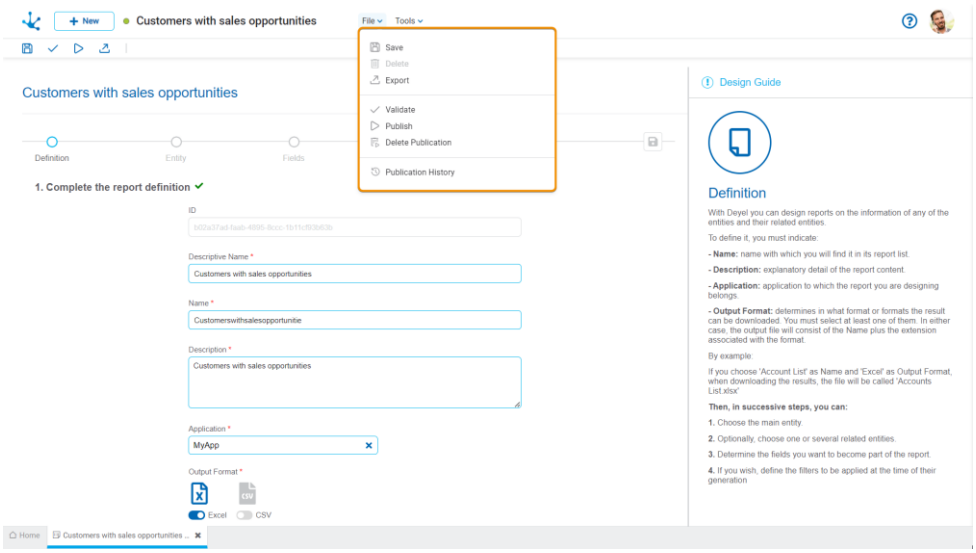


The expanded menu consists of:

- File Submenu
- Tools Submenu

3.4.17.1.1.1. File Submenu

This submenu opens by clicking on the “File” option and allows the performance of operations on the report.



Save

This icon allows to save the object in the repository of **Deyel**, leaving its state as "Draft" or "Modified". If certain conditions are met, the modeler user receives a message indicating that the operation was performed correctly, otherwise they receive an explanatory message.

Conditions

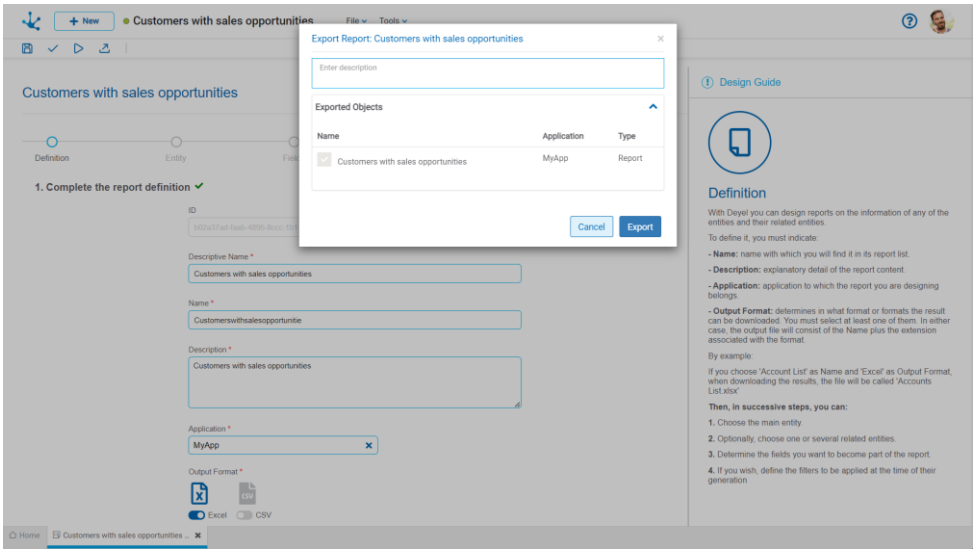
- The object application is required.
- The name in the application must be unique.
- The object permissions are required.

Delete

It allows to delete the object only if it is in "Draft" state and does not have other associated objects of **Deyel** that were previously saved or published. When deleted, the tab it is on closes and the object is removed from the modeler’s grid.

Export

This icon opens a window for the user to select and confirm the export of the object.



Description

In this property a text explaining the reason for the operation can be entered.
This text can be modified upon import and is displayed in the description column of the export record.

Exported Objects

By expanding the container, the object being exported is displayed. Objects not meant to be exported can be unchecked.
Click on the "Cancel" button to undo export or click on the "Export" button to finish.

Validate

This icon allows to validate if the object is ready to be published, that is, the same validations are carried out as when publishing and the result of them is reported.

Publish

Through this icon the object changes to “Published” state and the modeler user receives the corresponding message, indicating the result of the operation. The condition for publishing is the same as for the "Save" operation.

Delete Publication

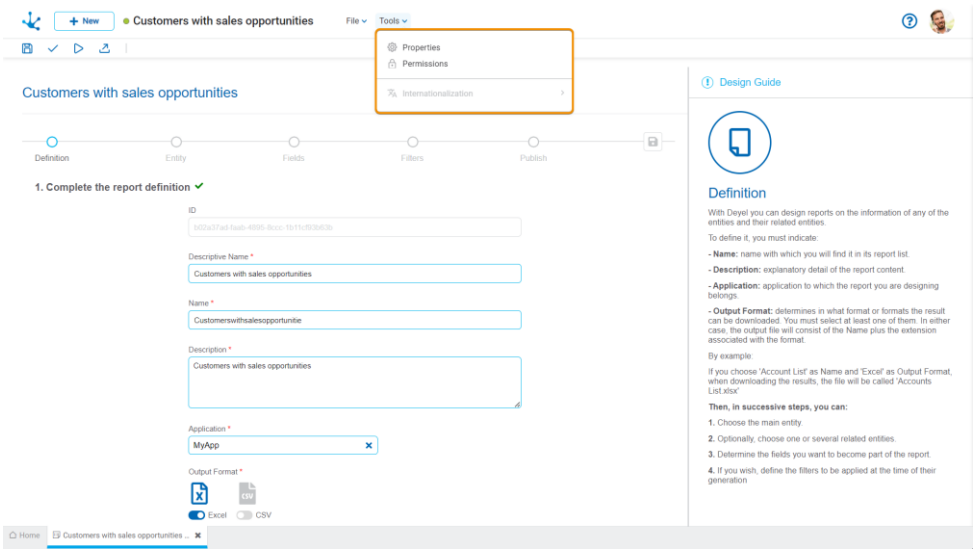
This icon allows disabling the report for use by returning it to the state “Draft”, in addition to deleting the data.

Publication History

This icon allows managing versions of published objects, displaying their history, to have better control of the changes that each object has.

3.4.17.1.1.2. Tools Submenu

This submenu opens by clicking on the “Tools” option and it allows modeling report properties, as well as its permissions and internationalization.



Properties

Opens the panel of report properties.

Permissions

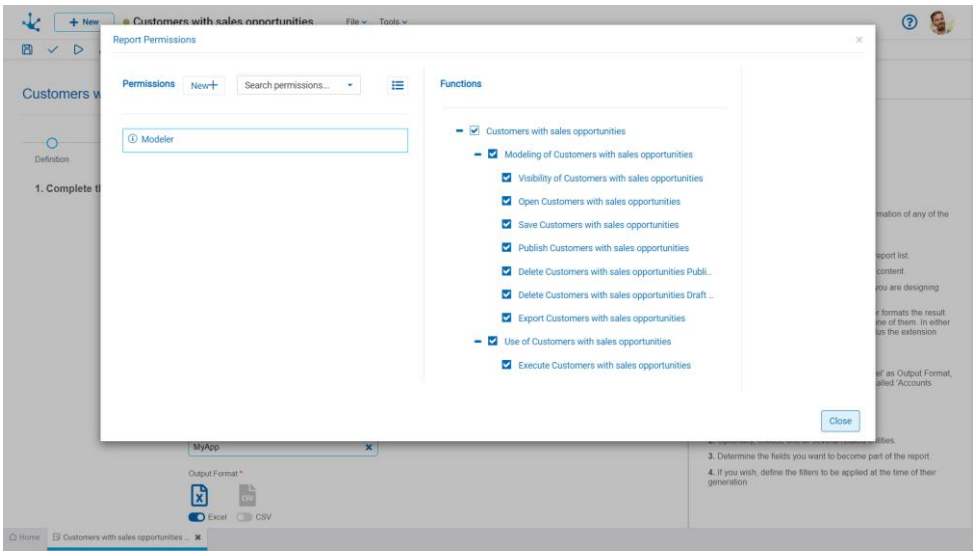
Allows to assign the security functions for use and modeling of the object to the existing permissions or to new permissions that the user creates, without having to go to the permission settings option from the menu.

Sections

- Permissions: Permissions to which object functions are assigned.
- Security Functions: Represents the total set of security functions, modeling and use of the object. Those that are marked are the ones included in the selected permission.

By default, all security functions for a new object are assigned to the permission Modeler.

Users who are assigned the permissions have access to the functions included in it.



New Opens a panel to create a new permission and once created, the security functions included in it must be selected .

Search permissions... Allows to select a permission from a list and enables the input of characters to filter the values in the list.

Opens the wizard to select a permission and once chosen, the necessary object security functions must be checked.

To unrelate a permission from the object, hover the cursor over the permission and press the icon . If there are functions selected for that permission, they must be unchecked in order to delete it.

Modeling Security Functions

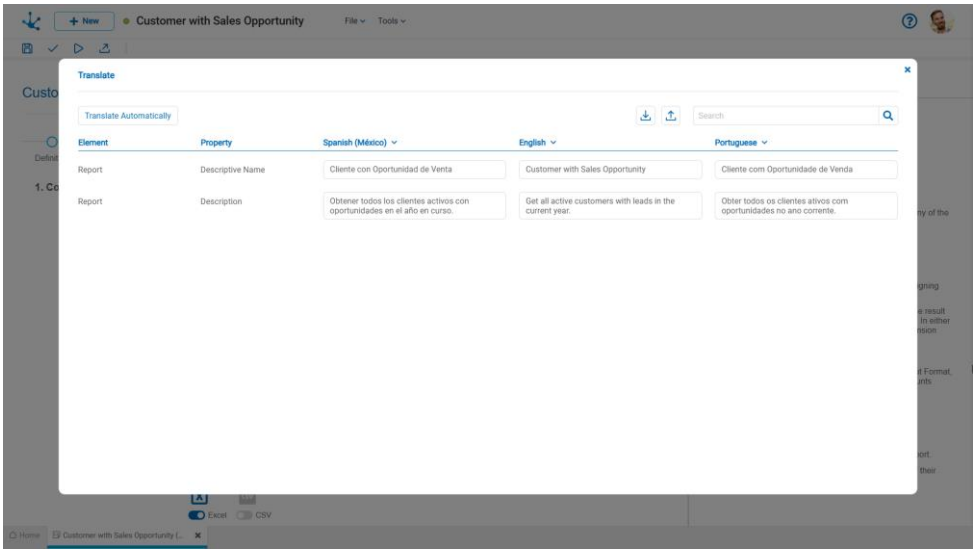
- Visibility: Allows to display the object in the Deyel modeler grid.
- Open: Allows to open the object from the Deyel modeler.
- Save: Enables the operation of saving modifications made to the object.
- Publish: Enables the operation of publishing the object leaving its state as "Published".
- Delete publication: Enables the operation of deleting the object publication leaving its state as "Draft".
- Delete draft definition: Enables the operation of deleting the object.
- Export: Enables the operation to export the object from the tools submenu of the expanded menu.

Use Security Functions

- Execute: It allows the execution of the object from the portal.

Internationalization

The "Internationalization" option is enabled when the property Internationalize is activated. To access the translation panel, it is necessary to select "Internationalization" and then choose "Translate".



In this panel, all the translatable properties of the object are displayed, along with their content in the original language, and translations in the other available languages can be entered manually.

If a value is modified in the original language column, the change is applied directly to the object's property.

In the header of each language column, a menu that allows selecting another language can be displayed, updating the column with its contents.

Options Available in the Top Margin

Translate Automatically Generates translations from the original language to other languages using artificial intelligence.

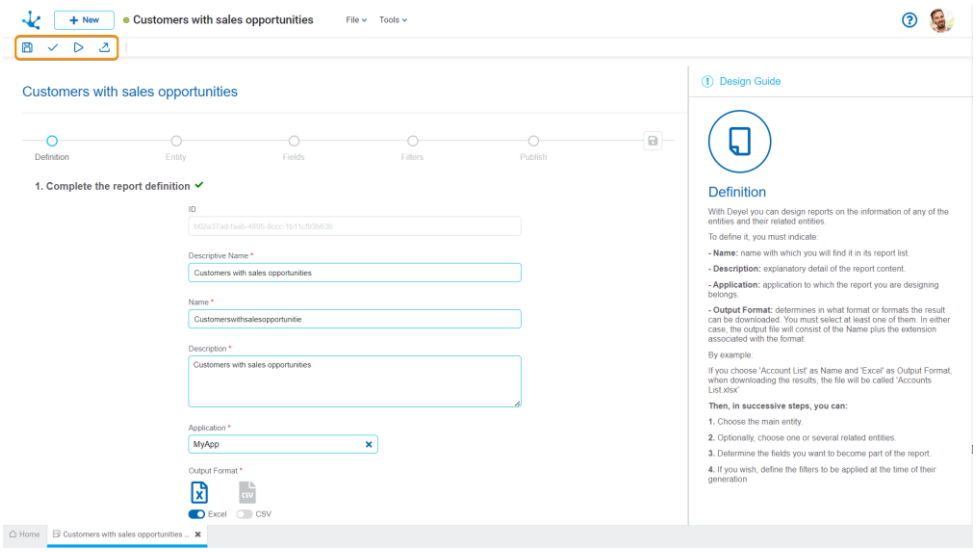
Downloads an Excel file with the contents of the translation panel.

Allows selecting an Excel file to import translations, making it easier to manage large volumes of content.

Filters the contents of columns based on the characters entered in the search field.

3.4.17.1.2. Top Toolbar

Contains icons for quick access to the most used operations of the expanded menu.

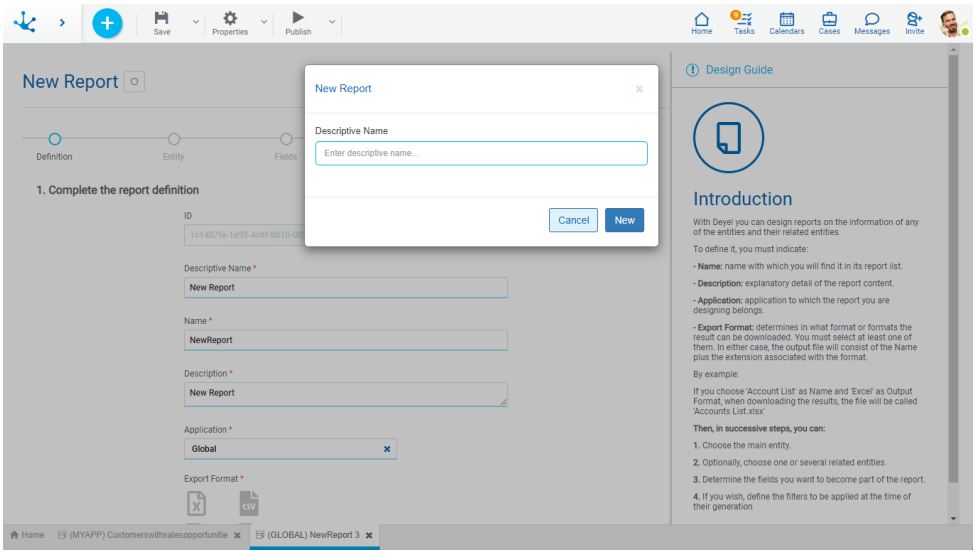


File

- Save
- Validate
- Publish
- Export

3.4.17.1.3. Modeling Wizard

The reports modeler uses a wizard that guides the user by indicating the necessary steps for modeling. First, enter the report name, then complete the necessary steps for modeling and finally publish it.



In each step of the wizard, useful information for each step of the modeling is displayed in the right panel, indicating how the requested fields should be completed.

The set of wizard steps is displayed by a bar where each step is represented by a circle. Once each step has been completed, a checkmark appears to the right of the description, indicating that it was completed successfully.

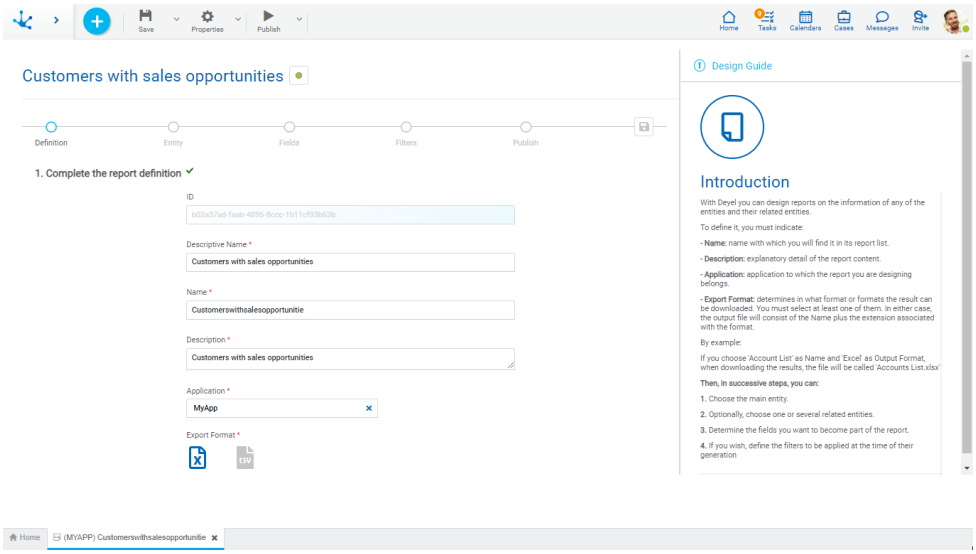
It is possible to advance to the next step by pressing the next circle only if the previous steps have been completed, it is also possible to go back to the previous ones.

3.4.17.1.3.1. Definition

In addition to the descriptive name that is entered at the time of report creation, the required properties must be entered, in addition to its output format.

Output Format

Determines the format(s) in which the report result can be downloaded. They can be Excel or CSV and at least one of them must be selected.



An asterisk "" on the label indicates that the property is required.*

3.4.17.1.3.2. Entity

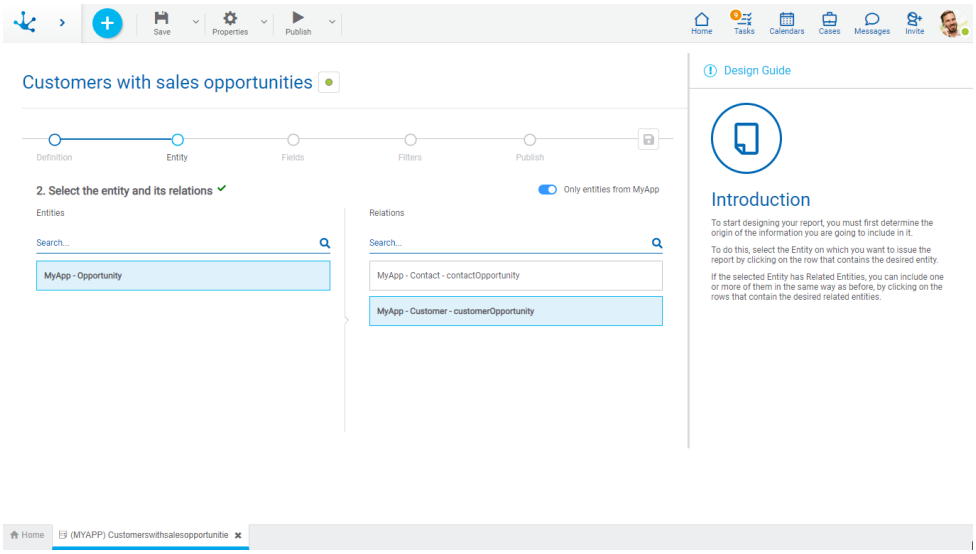
To start designing a report, the origin of the information included in it must first be determined. For this, select the entity for which the report is generated, by clicking on the row that contains it. If the selected entity has related entities, they are displayed on the right. One or more entities can be included within the same report, by clicking on the rows of the desired related entities.

For each related entity, the name of the field that establishes the relation is displayed on the same line.

To delete a selection, click on the line.

Only entities from <report application>

If this property is checked, only the entities of the application to which the report belongs are displayed.

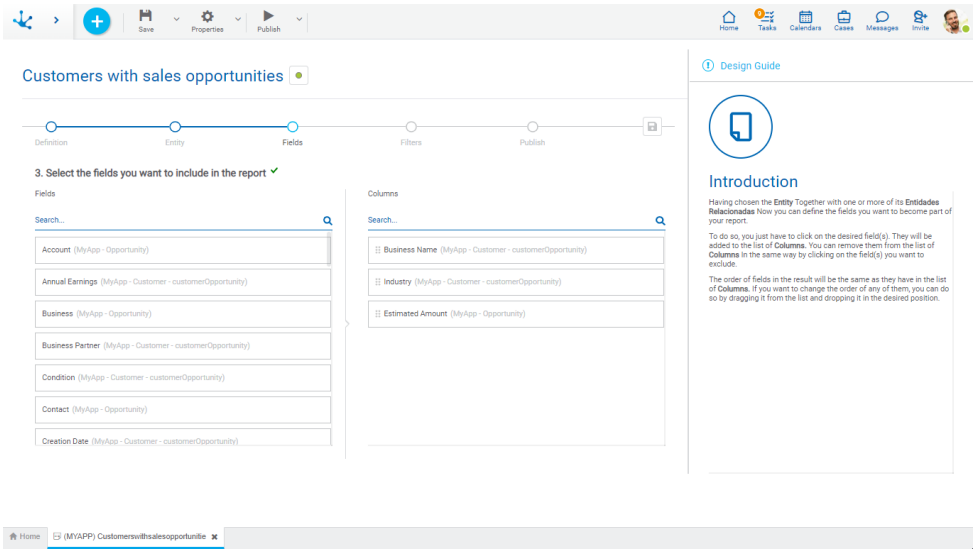


3.4.17.1.3.3. Fields

Once the main entity has been selected along with those related to it, the fields that become report columns must be defined. The fields of all the entities chosen for the report are available on the left to be selected.

By clicking on the desired field(s), they go to the columns area of the report. While clicking on the column(s) that are to be excluded from the report, they go to the fields area.

The order of fields in the report corresponds to the order in which they are seen in the columns area. If the order of any of them has to be changed, just drag it to the chosen position.



3.4.17.1.3.4. Filters

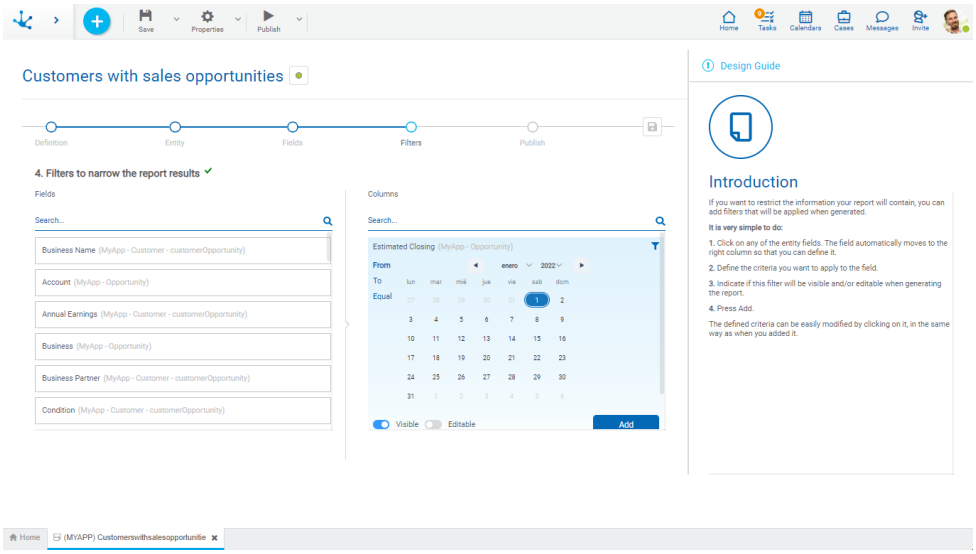
The fields of all the entities chosen for the report are available in the left area to be selected as filters, as in the previous step. For each selected field, conditions can be defined to filter the information shown in the report.

By clicking on the desired field(s), they go to the report columns area so that the conditions available for each field can be added.

For each filter, it is possible to indicate if it is visible at the time the user generates the report. Its editability by the user can also be configured in order to change the filter values.

The defined criteria can be easily updated by clicking on the filter line or deleted from the list by pressing the “X” icon.

Depending on the type of field, filter conditions are different and operators can be entered. Once the filter criteria have been defined, the "Add" button should be pressed so as to define the filter.



Filter Conditions

Numeric Fields

The value entered should be numeric.
Search criteria:

- Greater than
- Greater equal to
- Less than
- Less equal to
- Between
- With Data
- No Data

Alphanumeric Fields

Enter a text to search for.
Search criteria:

- Contains
- Equal to
- Starts with
- Does not start with
- No Data
- With Data

Date Fields

A calendar opens to select the date and it can be filtered using different search criteria.
Options:

- From
- Until
- Equal

DateTime Field

A calendar opens to select the date and time, it can be filtered using different search criteria.

- From
- Until

- Equal
- Range

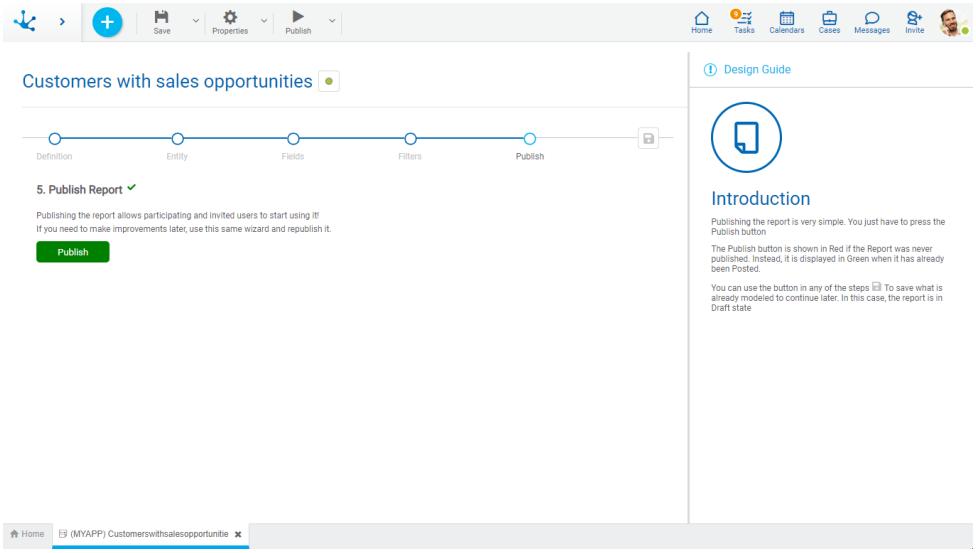
Value Lists Fields
The list values are displayed.
Search criteria:
- Included
- Not Included
- With Data
- No Data

3.4.17.1.3.5. Publish

Publishing a report allows users who have been assigned permissions to start using them.

The button to publish the form turns from red to green once the first publication has been made.

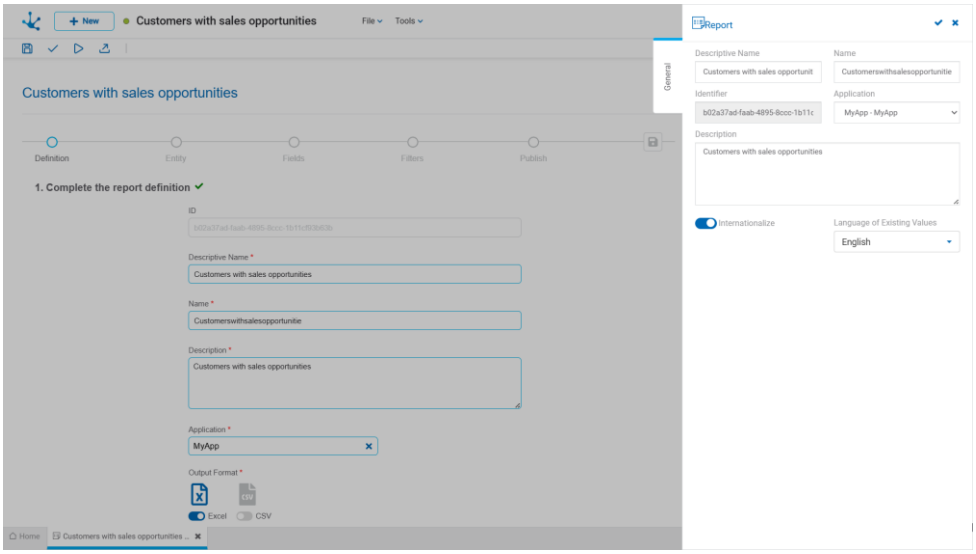
The icon can be used at any step in order to save what has been modeled without making a publication. In this case, the report remains in state "Draft".



3.4.17.1.4. Report Properties

Report properties can be entered both at the time of their creation and when updating an existing one.

To enter the report properties panel, use the icon which is in the top toolbar.



General Tab

Properties

Descriptive Name
It is the report name being displayed.

Name
It is the report name. No blanks or special characters allowed. It is unique and required.

Identifier
It is the report internal name. It is generated automatically.

Application
Allows to define the application to which the object belongs. If no application is reported, the object is assigned to the application "Global".

Description
Defines the report by describing its content.

Internationalize
Enables the translation of object properties and its elements into different languages. When activated, it enables the option "Internationalization" from the “Tools” drop-down menu and the property Language of Existing Values, which is displayed on the right.

Language of Existing Values

Displayed when the property Internationalize is activated for the first time and indicates the original language. Its selection is required and the available options are: “Spanish”, “English” and “Portuguese”.

Actions

The icon ✔ is used to confirm the modifications made in the properties panel.

The icon ✖ is used to close the properties panel, if it was not previously saved, changes are discarded.

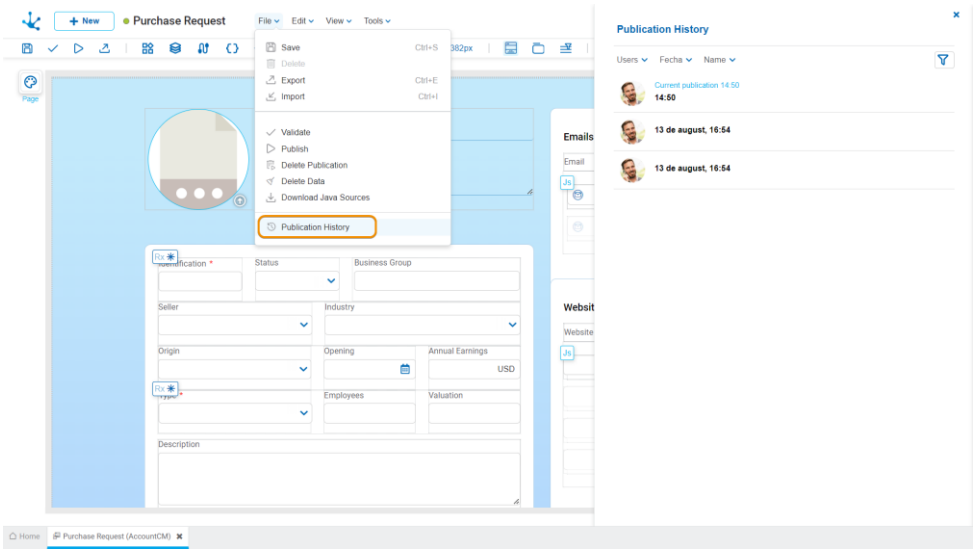
3.4.18. Publication History

The publication history allows managing versions of the published objects, to better track the changes that each object has undergone.

A version is a snapshot of the current state of an object, which can be used to compare it with past or future versions, and thus observe the changes made or analyze differences between versions.

The objects on which versions can be managed are all those that can be published:

- Application
- Entity
- Page
- Process
- Rule
- Form
- Dashboard
- Widget
- Report
- Value List
- Adapter



A version is generated only when an object is published. Version generation is completely transparent to the user, since **Deyel** is responsible for versioning the object automatically.

Each time an object is published, a new version is generated. So the version list of an object can contain 0 or more versions. If the object has never been published, the versions list is empty, and for each publication, a new version is generated in the list. There is no maximum limit to the number of ver-

sions. In other words, the version list contains the history of all publications of the object. The latest version is the current one and is the one that **Deyel** uses to execute, that is, the current version corresponds to the object's execution definition.

Version Properties

Creation user
Assigned by **Deyel** at the time of publishing the object.

Creation date and time
Assigned by **Deyel** at the time of publishing the object. It is displayed if the name is modified.

Name
By default it contains the creation date and time. It can optionally be modified to customize the published version of the object.

Description
Optionally, it can be entered to detail a user story, a ticket, or a more extensive explanation describing the content of the published version of the object.

Access to the Version List

The list of published versions is accessed using the “Publication History” option, from the expanded menu "Archive" within each object modeler.

This list opens in a new tab and contains information about its properties.

- Creation user
- Creation date and time
- Name
- Description

When an object is exported, the publication history is not included in the object export. When the object is imported, the publication history of the imported object remains unchanged.

Operations on a Version

A set of operations can be performed on each published version.

Search the publications grid

A grid or ordered list of published versions, with information about their properties can be used for each object.

The grid is sorted in descending order by creation date and time, so that the current version, which is the last publication of the object, is shown first.

The grid can be filtered by some of its properties:

- Creation date
- Creation user
- Name

Show

In the versions grid, each row can be selected to show the information of the chosen version. **Deyel** opens the object modeler in show mode in a new tab, with the version selected, where the object's operations are disabled, for example, save, publish, or delete. The tab name is formed by the combination of the object name, the date and time of creation of the published version and the name of the application to which the object belongs.

Modify

This operation allows the entry or update of the version name and its description.

3.5. Admin

The administration console centralizes the management of all resources and configurations necessary to ensure the correct functioning of applications within the environment.

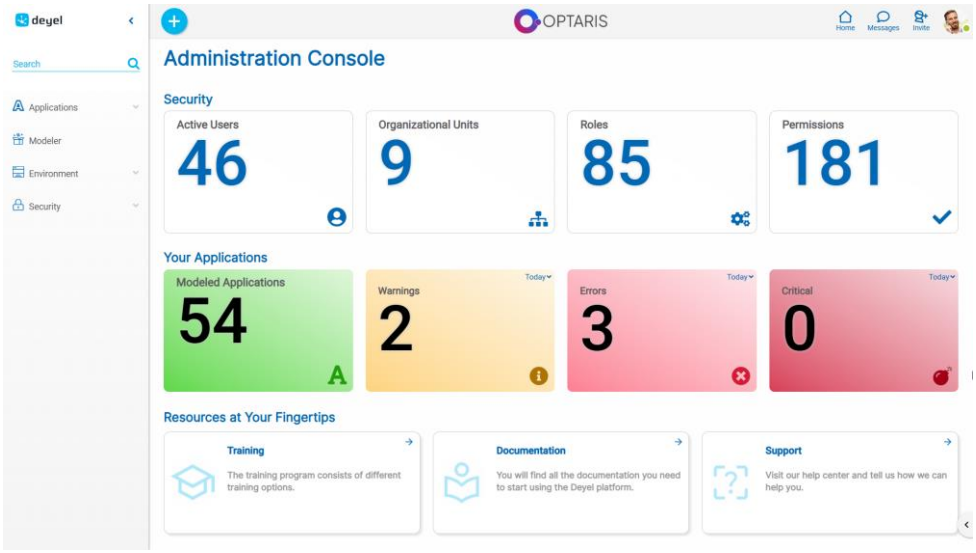
From this console, administrators, modelers, and security administrators have all the necessary tools to create, modify, and install applications, and to configure the various properties of the environment in accordance with the defined security policies.

To start using the administration console, it is neecessary to access the URL of the corresponding environment, for example:
https://myenvironment.deyel.com/admin.

The administration console features a main panel with indicators that summarize the overall state of the environment. Information related to security management is displayed, including users, organizational units, roles and permissions, as well as the state of modeled applications, including warnings, errors and critical events.

From the “Resources at Your Fingertips” section of the main panel, access to support resources is available, including:

- Training: access to the available training
- Documentation: access to the platform documentation
- Assistance: access to the ticket management



Management from the Console

The administration console allows managing fundamental aspects of the environment and thus maintain comprehensive and centralized control of the environment, ensuring that both the operation and development of applications are carried out with the required security and efficiency.

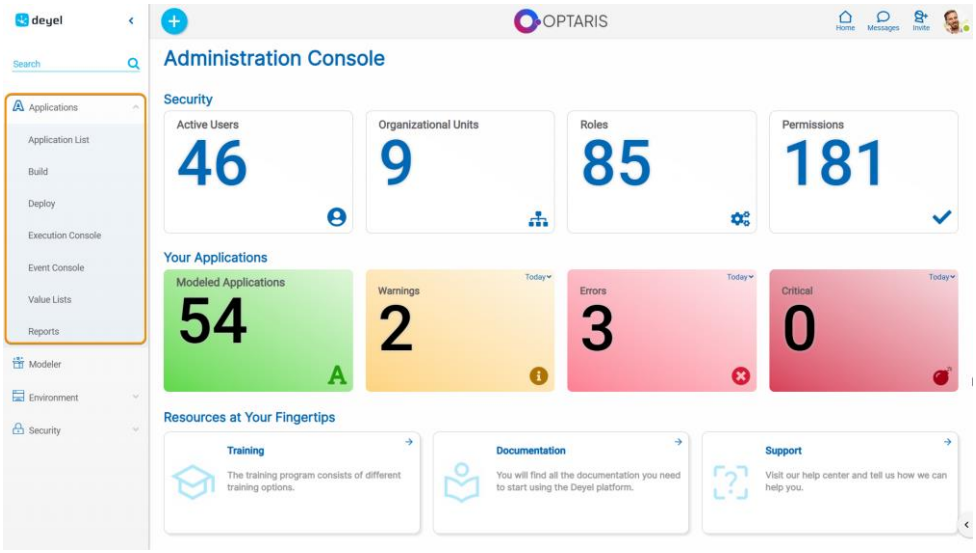
The side menu allows navigating between the different functionalities of the platform:

- Applications
- Modeler
- Environment
- Security

3.5.1. Applications

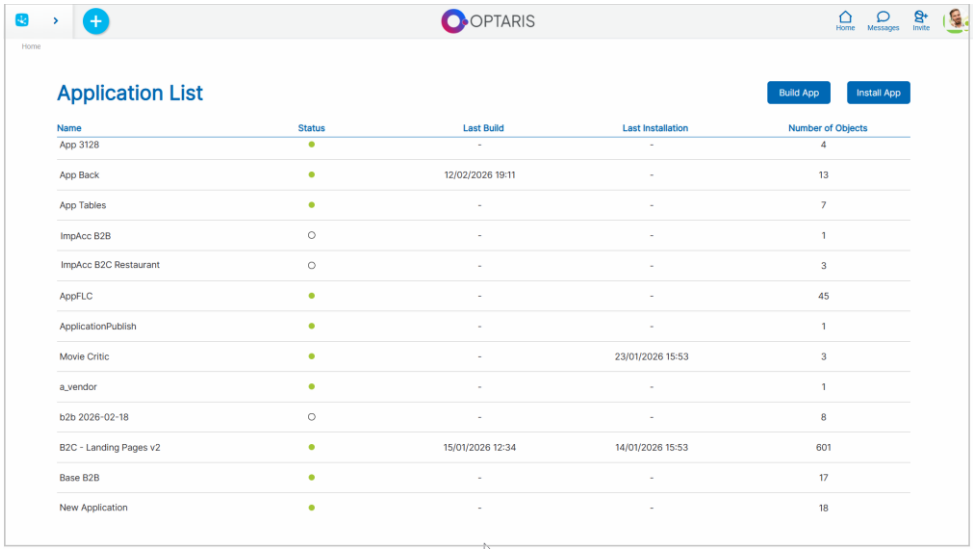
Centralizes the functionalities associated with the comprehensive management of the modeled applications, including their visualization, build, installation, monitoring and administration of related information.

- List of applications
- Build of Applications and Changes
- Build of Applications and Changes
- Execution Console
- Events Console
- Value List
- Reports



3.5.1.1. List of Applications

The list of applications allows the administrator to build, install, and view data from the applications in their environment.



Properties

Name

Identifies the application within the environment.

State

Indicates the application state. Allows quick identification of which applications are available for use.

Last build

Date and time when the most recent application build was performed. If no information is displayed, it means that the application has no recorded builds in the current environment.

Last installation
Date and time when the most recent application installation was performed. If no information is displayed, it means that the application has not recorded installations in the current environment, since it was created directly within it.

Number of Objects
Total number of objects that make up the application.

Actions

The following buttons are available in the top right corner.



It allows selecting an application and generating a new build for its subsequent installation in another environment.



It allows installation , in the current environment, of an application previously built in another environment.

3.5.1.2. App Build and Changes

Through the building and deploying applications functionality, **Deyel** facilitates continuous application integration by enabling an organized promotion of changes across environments. The result of building an application, whether complete or just changes, is a package (.zip file), which will later be used in the installation.

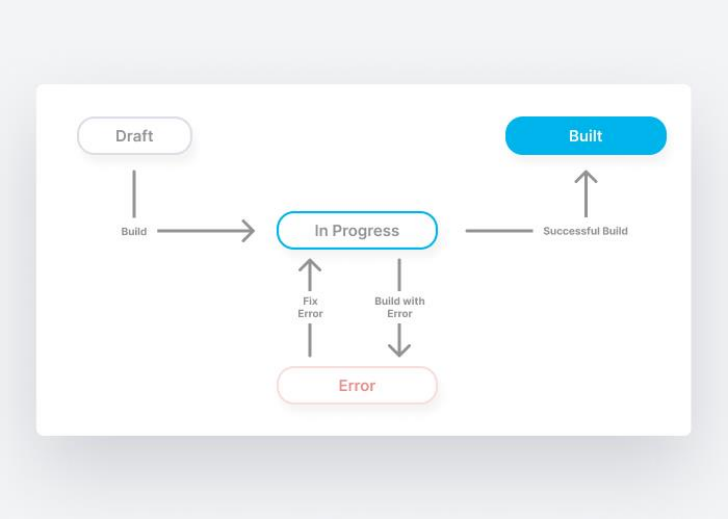
The build generates an installable package that includes the configuration data and objects selected by the user.

A build is a collection of selected objects and entity data that needs to be installed in an environment.

A construction may contain:

- One or more complete applications, with all their objects.
- Selected objects, belonging to one or more applications
- Entity data.

State Diagram in a Build



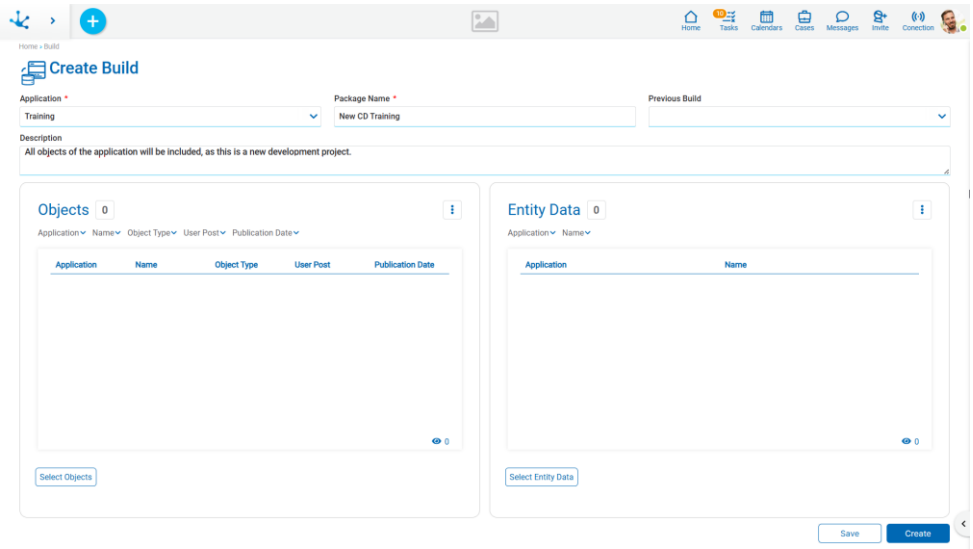
Possible states of a build:

- Draft: the build was partially saved by the user.
- In Progress: **Deyel** is processing the request and generating the package.
- Built: build was completed successfully.
- Error: The build failed and an error occurred.

3.5.1.2.1. New Build

To create a new build, hover the mouse pointer over the icon and select .

The user executing the build must have the 'Building Applications and Changes' permission and can only create builds for applications for which they have defined security permissions.



Properties

Application
Drop-down list showing the published applications for which the user has permissions. This is a required and informative field that allows the build to be associated with a specific application, although the package may include objects from other applications.

Package Name
Name that identifies the build within the application.

Previous Build
Enables selecting another build as the base for this build.

Description
Free text field for entering relevant information about the build.

Object Selection Grids and Entity Data

The grids display the objects and data of entities included in the build.

- Filters can be applied to columns to display selected objects and entity data included in the build.
- It is possible to customize the visible columns from the icon  .
- The number of displayed objects is shown with the icon  in the bottom right corner.
- When hovering over an included object, the option to delete it is displayed.

Only objects with “Published” state can be included in a build.
An object can appear in the entity data grid without being included in the objects grid. In other words, the build can contain only the data of an entity, without including its complete definition.

The objects grid can be empty only when entities have been added to the entity data grid.
Likewise, the entity data grid can be empty only when objects have been added to the objects grid.

Therefore, elements can be added to one grid, the other, or both, but both grids cannot be empty.

The grids contain:

- Objects
- Entity Data

Save and Complete

By clicking the “Save” button, it is possible to partially save the build for later retrieval.

To complete the build, press the “Create” button. Deyel Bot sends a message when the execution starts and reports the build result upon completion.

Once the build creation is complete, the .zip file package becomes available for download. The build name is unique per application.

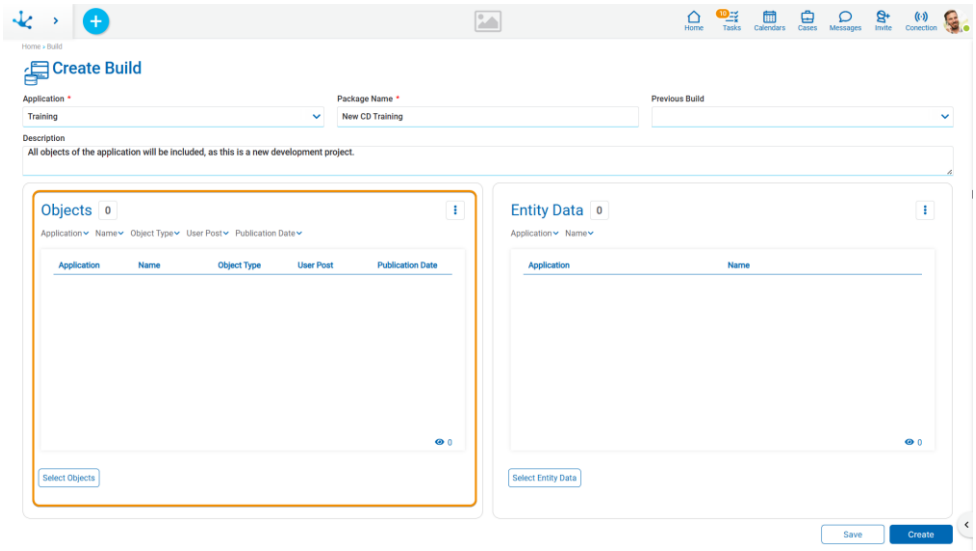
The build consists of including objects, adding entity data, and finally generating the .zip file package. This may take a few minutes, depending on the number of objects. The execution is performed unattended and, upon completion, Deyel Bot sends a message notifying the build state.

If the build fails, the error is reported so that it can be corrected. The error details can be consulted from the operation "Show" that can be accessed from the build grid.

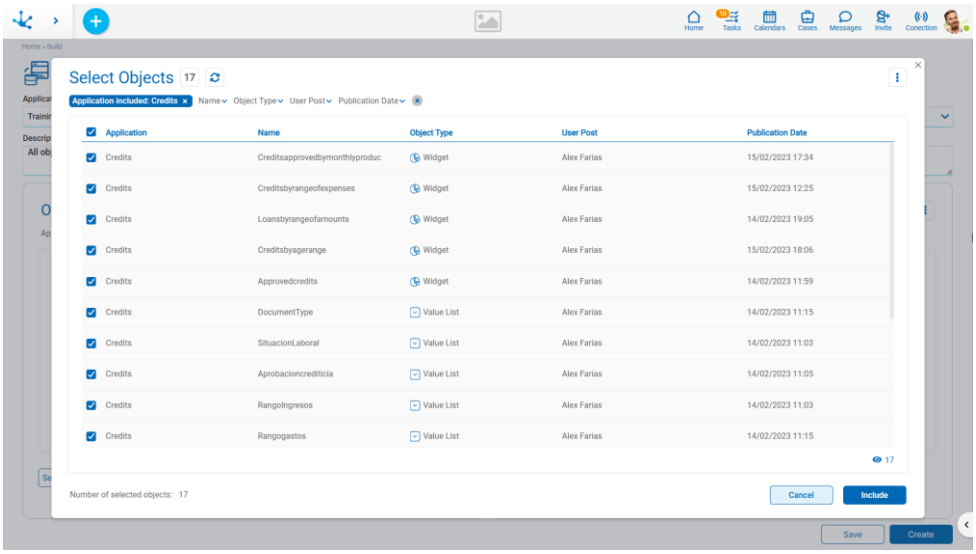
3.5.1.2.1.1. Objects

This grid displays the objects included in the build. Only published objects from published applications are allowed to be chosen.

However, when selecting precedents or dependents, published objects from unpublished applications can be chosen.



By clicking the 'Select Objects' button, a panel opens where objects can be searched, filtered, and selected to be included in the build.



- Every object has a checkbox.
- The check next to the title allows selecting all objects.
- Selecting at least one object enables the “Include” button.

When “Include” is clicked, a message appears prompting verification of precedents and dependents. If confirmed, the objects are added to the build and the “Select Precedents” and “Select Dependents” buttons are enabled.

Selection of Precedents and Dependents

The build offers assistance to identify precedents and dependents according to the objects Included. Once at least one object has been selected, the “Select Precedents” and “Select Dependents” buttons are displayed, allowing these objects to be added to the build and included in the package.

The build can be performed without including preceding or dependent objects, for example, when it is certain that they already exist in the target environment.

Is it necessary to include precedents in the build?

- No, if the precedent is exactly the same in the target environment.
- Yes, if the precedent does not exist in the target environment.
- Yes, if the precedent exists and is different in the target environment, and if the changes made to the precedent need to be carried over to the target environment.

The verification of precedents can be performed as many times as necessary, until no pending precedents remain to be included. This implies that precedents of precedents can be included recursively, until the list of unincluded precedents is empty.

When analyzing required precedents, it is common to find precedents that belong to another application, such as the global application. This can occur when an object modeled for one application is unintentionally assigned to another. In these cases, it is possible to correct the assignment so that all objects corresponding to an application are correctly grouped within it.

Clicking the “Select Precedents” button opens a panel where the precedents to include can be searched, filtered, and selected. Clicking on Include opens a confirmation message. If confirmed, the selected precedents are added to the objects grid of the build.

Is it necessary to include dependents in the build?

- It is not necessary to include dependents if the application does not exist in the target environment.
- Dependents must be included if the application already exists in the target environment and any of the selected objects have at least one of the following changes:
  - In an entity: deletion of a field, or changes to the application, class name, identifier, or data type of a field.
  - In a rule: deletion of a parameter, or changes to the application, class name, or data type of a parameter.
  - In a process: deleting an entity relation, or changing the application or class name.

The verification of dependents can be performed as many times as necessary, until no pending dependents remain to be included. This allows including dependents of dependents recursively, until the list of excluded dependents is empty.

Clicking on the “Select Dependents” button opens a panel where the dependents to include can be searched, filtered, and selected. Clicking on “Include” opens a confirmation message. If confirmed, the selected dependents are added to the objects grid of the build.

In summary, it is likely that precedents will need to be included in a build, whereas including dependents is uncommon, since the conditions that justify it are rarely met.

Key Note:
It is essential to carefully evaluate whether it is really necessary to include all the precedents and dependents, or only some of them. Including unnecessary objects can result in larger packages than required, affecting both build and installation times and costs. Therefore, it is recommended to incorporate only those objects whose function is clearly understood and whose inclusion is justified, avoiding adding elements only 'just in case'.

Objects Excluded from Build

Some objects are excluded from application builds because they are an essential part of the target environment and must be configured directly there, such as test or production.
Deyel neither proposes them as precedents or dependents nor allows adding them to the object list or the entity data list.

Excluded items:

- Users, organizational units and agents:
  They must be previously configured in the target environment for the installation to be successful. Otherwise, the installation will fail and be rejected.

- Adapters

Deyel allows exporting adapters directly from the modeler, but only the following properties are exported: identifier, name, descriptive name, and description.
Other configuration properties (such as server, database, job, user, password, etc.) are not exported and must be manually configured in the target environment.

- Job positions:
  They are not exportable between environments and require manual configuration in the target environment.
  If a job position is not defined in the target environment, the installation will be successful but incomplete, as a part that was modeled in the development environment will be missing.

Partially included objects:

- Roles:
  Deyel includes the role only with its identifier, application, name, descriptive name and description.
  Actors, coordinators, and role permissions are not exported between environments.
  The installation will be successful, but for the application to function properly, role configuration must be completed in the target environment.

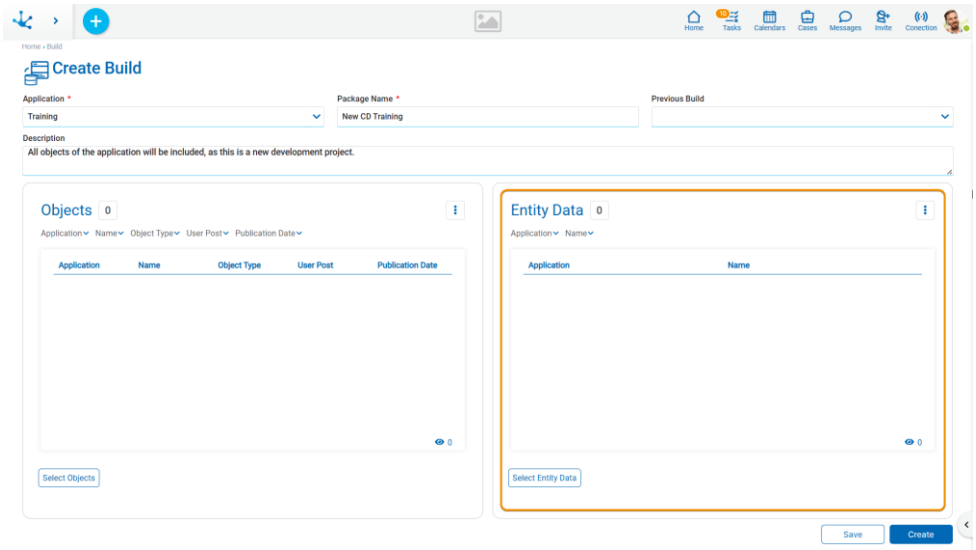
Build a Complete Application

To build a complete application, starting from the selection of objects, the following steps must be followed:

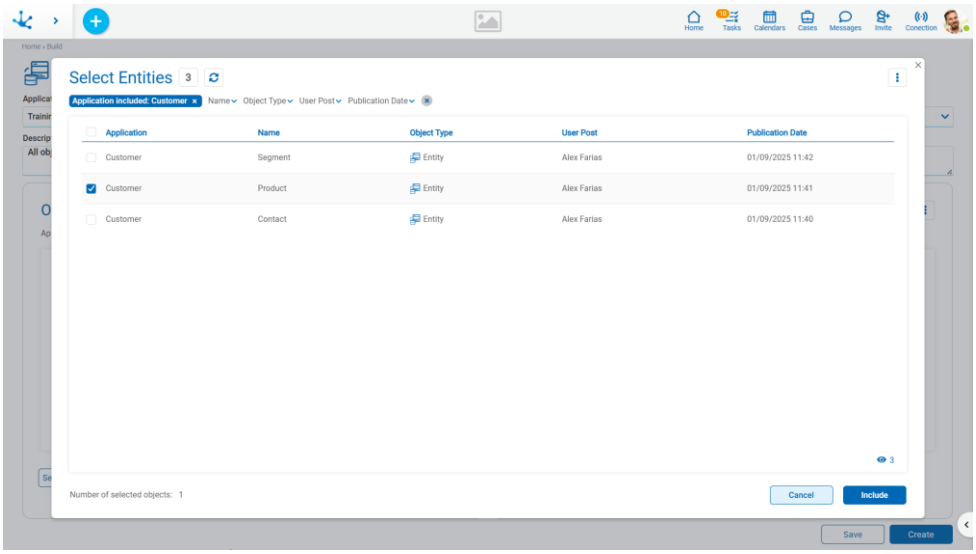
- Filter by the desired application.
- Scroll down to the bottom of the grid.
- Select all objects.
- Check the number of selected objects.
- Click the “Include” button.

3.5.1.2.1.2. Entity Data

This grid displays the entities included for data migration.



By pressing the 'Select Entity Data' button, a panel opens where entities can be searched, filtered, and selected to include their data in the build.



- Each entity has a checkbox.
- The check next to the title allows selecting all entities.
- Selecting at least one entity enables the “Include” button.

Clicking on "Include" opens a message with the option to cancel or confirm. If confirmed, the entities are added to the build.

3.5.1.2.2. Grid

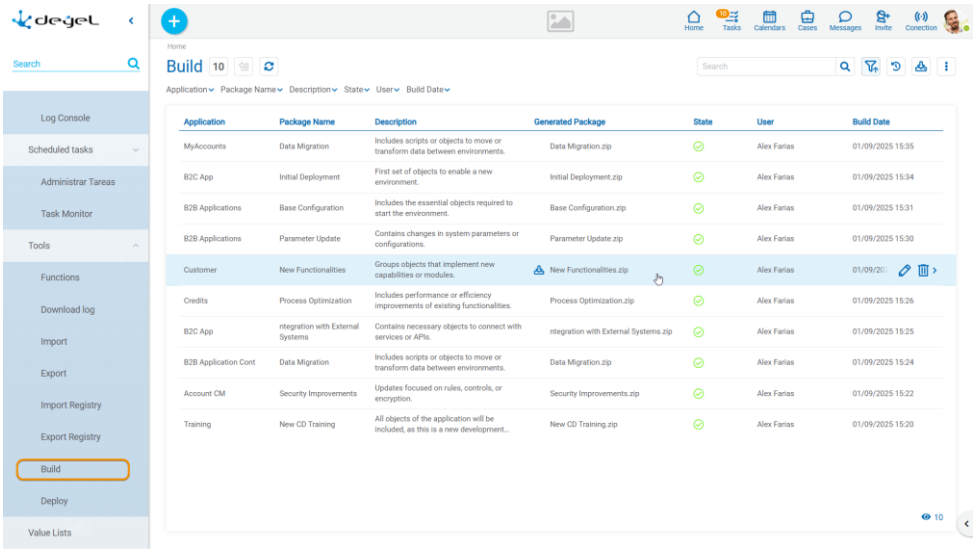
The build grid can be accessed from:

- The “Configuration” menu, selecting "Tools" and then “Build”.

- The search facility.

Previous builds are displayed in a grid format, following the standard layout of the results grids, using the following facilities:

- Sorting
- Search and Filters Bar
- Data Download
- Operations



The following properties of the builds are displayed as grid columns:

- Application
- Package Name
- Description
- Generated Package
- State (Possible values: Draft, In Progress, Built, Error)
- User
- Build Date

The following properties can be used as filters:

- Application
- Package Name
- Description
- State
- User
- Build Date

It is possible to perform operations on each grid row of the build. By clicking on the row, the selected build is shown, while through the icons and an update or deletion is performed respectively.

Also, hovering the mouse pointer over a build in the column "Generated Package" displays the icon  ; by clicking this icon, the .zip file generated for the build is downloaded.

3.5.1.2.3. Operations

Operations can be performed on each build, depending on the security permissions defined for the logged-in user. Hovering the mouse pointer over each row of the grid displays the icons corresponding to the available operations.

Except for the operation "Create", which is carried out exclusively from the context menu.

The view that opens when performing an operation from the grid displays, in addition to the columns already visible, the following information:

- Build detail: log that displays the progress of the build.
- Error description: is only displayed if the build state is "Error".
- List of included objects: details the objects included in the build.
- List of configuration entity data :shows the entities whose data is migrated during the build process.

Show

Opens a view with the information of the selected build. The attributes shown in the view are not editable. According to the security permissions of the user, the available buttons to operate on that build are enabled.

Modify

Opens a view with the information of the selected build, displaying as editable only those attributes that can be modified. The modification is performed by clicking the "Create" button, after which the user receives a message confirming the operation.

This action is mainly useful to:

- Resume the build creation in "Draft" state.
- Retry the creation of a build that is in "Error" state.

In the latter case, it is recommended to first correct the reported error to prevent the build from failing again.

The attributes that can be modified are the same as those available when creating a build:

- Application (Required)
- Package Name (Required)

- Description
- Previous Build
- List of Objects
- Entity data List

Additionally, **Deyel** offers assistance through:

- List of not included Precedents
- List of not included Dependents

Delete

Opens a view with the information of the selected build, where the attributes are not editable. The deletion is performed by clicking the "Delete" button, after which the user receives a message confirming the operation.

> *This action is irreversible and cannot be undone.*

Download

Once Deyel Bot confirms the successful completion of the build, it is possible to download the package. This operation can be repeated multiple times; in every case, the generated .zip file contains the same information and is saved in the specified local or network location.

3.5.1.3. App Deploy and Changes

Implementing new applications, or applying changes and modifications to existing ones, requires transferring the new application functionalities securely and reliably from the development environment to testing, staging, and finally to production.

Through the building and deploying applications functionality, **Deyel** facilitates continuous application integration by enabling an organized promotion of changes across environments. The result of building an application, whether complete or just changes, is a package (.zip file), which will later be used in the installation.

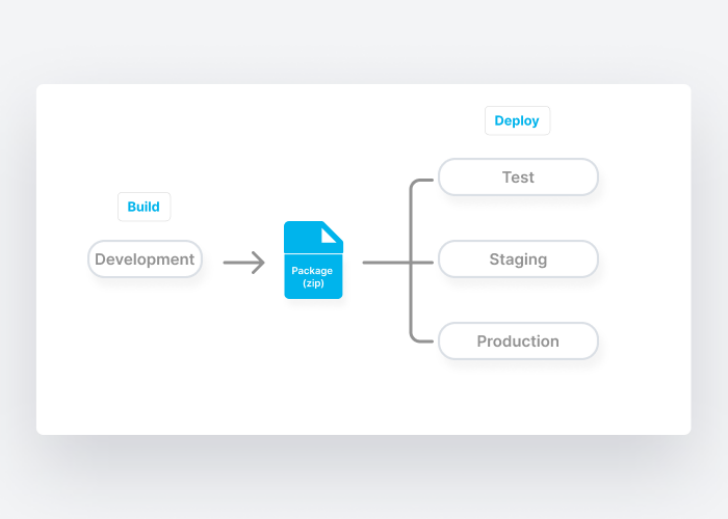
The Installation process takes the contents of the package and installs it in the target environment (test, staging or production), providing detailed results.

Regarding the installation of configuration data, for each instance of the entity included in the build, a search by ID in the target environment is performed. Depending on the result of this search, the following procedure is followed:

- If the instance does not exist, it will be created.
- If the instance already exists, it will be updated.
- If the instance exists on the target but is not included in the package to be installed, no action is taken.

Deyel keeps a record of the builds and installations carried out in each environment:

- In the source environment, the builds are recorded along with their contents.
- In the target environments, packages are installed and each installation is logged.



From a package, its deploy can be executed in a target environment (e.g., test or production). The same package can be installed more than once in the same environment. **Deyel** independently records the information associated with each of these facilities.

The installation is atomic, meaning that the package is either installed completely or not installed at all. If an error occurs during installation, no changes are applied, ensuring the integrity of the target environment.

Processing During a Build

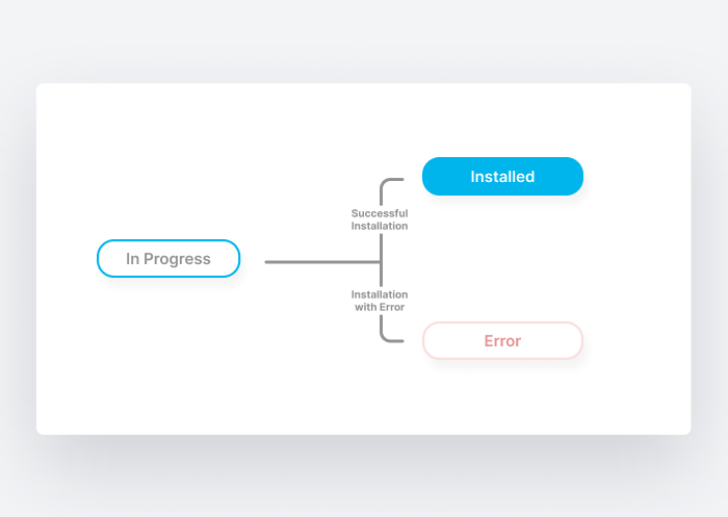
In **Deyel,** a build involves processing that may take several minutes to complete, due to the complexity of the tasks required to ensure a correct and stable installation.

During this process, **Deyel** performs a series of actions:

- Analysis of the state of dependent objects before starting the installation.
- Integrity validations, related to object modeling.
- Verification of required precedent objects.
- Compatibility evaluation of dependent objects.
- Import of objects
- Publication of objects without validations.
- Publication of objects with validations.

This processing is performed unattended. Once completed, DeyelBot sends a message notifying the completion of the process along with its state.

State Diagram of a Deploy



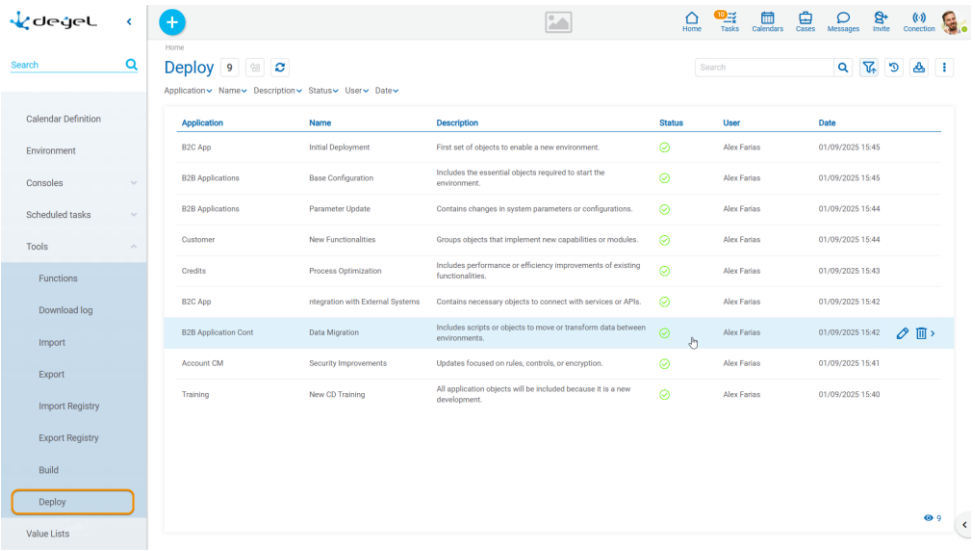
Possible states of a deploy:

- In Progress: **Deyel** is performing package installation.
- Installed: The installation completed successfully.
- Error: The installation did not complete due to an error.

3.5.1.3.1. New Deploy

To create a new deploy, hover the mouse pointer over the icon and select .

To install a package, it is necessary to have the “Application and Change Deployment” permission assigned, which enables access to this functionality.



Properties

Package
Allows selecting the package to install, either by loading it from the device or importing it from Google Drive. The fields "Application" and "Name" are automatically completed based on the package information.

Description
Free text field for entering relevant information about the installation

Installation Start and Pre-Validation

By pressing the button "Create", the package installation process starts. During this process, **Deyel** performs a series of validations to ensure that the target environment meets all necessary requirements. If errors are detected, each one is reported in detail so that they can be corrected before proceeding.

Validations Performed

Validation of modeling conditions

The target environment is checked for the required modeling configurations in the following elements:

- Users
- Organizational units
- Agents
- Adapters
- Roles

Validation of required applications

It is verified that the applications associated with the package objects exist in the target environment. Otherwise, such applications must be included in the package as objects. For example:

- If the package contains all objects of application 'A' but the application does not exist in the target environment, it must be included as an object in the package.
- If the package includes a single object of application "B" and this application does not exist in the target, it must also be added as an object.

Validation of required precedents

It is validated that all required preceding objects (that are not included in the package) already exist in the target environment.

If the installation completes successfully, all objects in the package are published and productive, ready to be executed on **Deyel**.

> *For roles, it may be necessary to complete their configuration in the target environment to ensure the application functions correctly.*

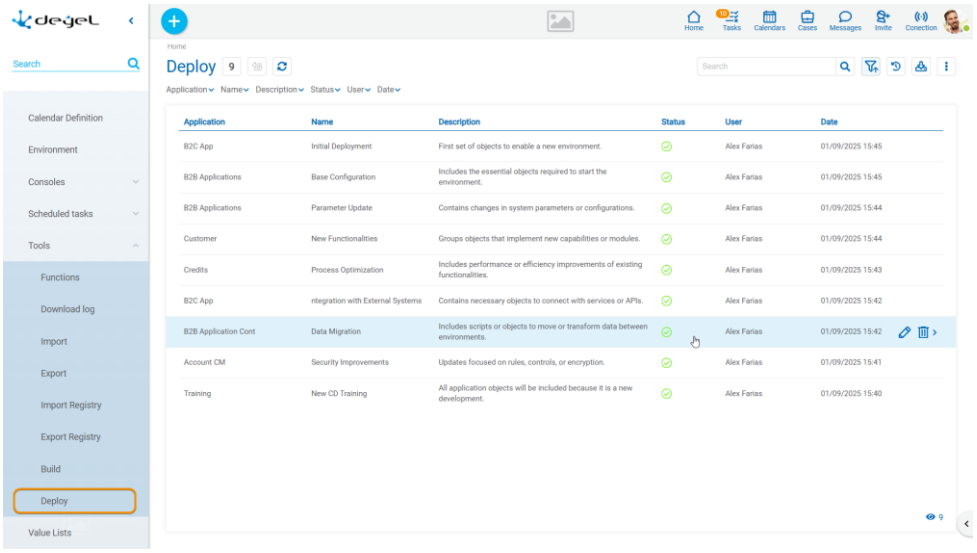
3.5.1.3.2. Grid

The facilities grid can be accessed from:

- The “Congiguration” menu, selecting "Tools" and then “Deploy”.
- The search facility.

Previous installations are displayed in a grid format, following the standard layout of the results grid, using the following facilities:

- Sorting
- Search and Filters Bar
- Data Download
- Operations



The following properties of installations are displayed as grid columns:

- Application
- Name
- Description
- State (Possible values: Draft, In Progress, Built, Error)
- User
- Installation Date

The following properties can be used as filters:

- Application
- Name
- Description

- State
- User
- Installation Date

It is possible to perform operations on each grid row of the installation. By clicking on the row, the selected installation is shown, while through the icons  and  an update or deletion is performed respectively.

3.5.1.3.3. Operations

Operations can be performed on each installation, depending on the security permissions defined for the logged-in user. Hovering the cursor over each row of the grid , icons corresponding to the available operations are displayed.

The exception is the operation "Create", which is carried out exclusively from the context menu.

Show

Opens a view with the information of the selected installation. The attributes displayed in the view are not editable. According to the security permissions of the user, the available buttons to operate on that installation are enabled.

In addition to the columns already visible in the grid, the following properties can be viewed:

- Package
- Installation detail
- Error description: visible only when the installation state is "Error".
- List of objects: is displayed only if the installation was successful.
- List of entity data: is displayed only if the installation was successful.

Modify

Opens a view with the information of the selected installation, displaying as editable only those attributes that can be modified. The modification is performed by clicking the “Modify” button, after which the user receives a message confirming the operation.

Delete

Opens a view with the information of the selected installation, where the attributes are not editable. The deletion is performed by clicking the “Delete” button, after which the user receives a message confirming the operation.

> *This action only deletes the installation record.*

3.5.1.4. Application Execution Console

This console allows to review the execution of the applications modeled with **Deyel** by recording the actions performed when executing the modeled objects.
To carry out the analysis, different levels of registration can be defined, which allow knowing the behavior of the applications according to the needs of the modeler user. For example, in development environments everything that has been executed can be known, while in productive environments it is necessary to focus on performance improvements or check if there are errors.

For each action, the executions register a start and an end event, indicating in each one the date and time of completion, the type and object involved, the executing user and additional information about each action, among other things.

The end events also indicate whether the execution ended correctly, with warnings or with an error, with the details in each case.

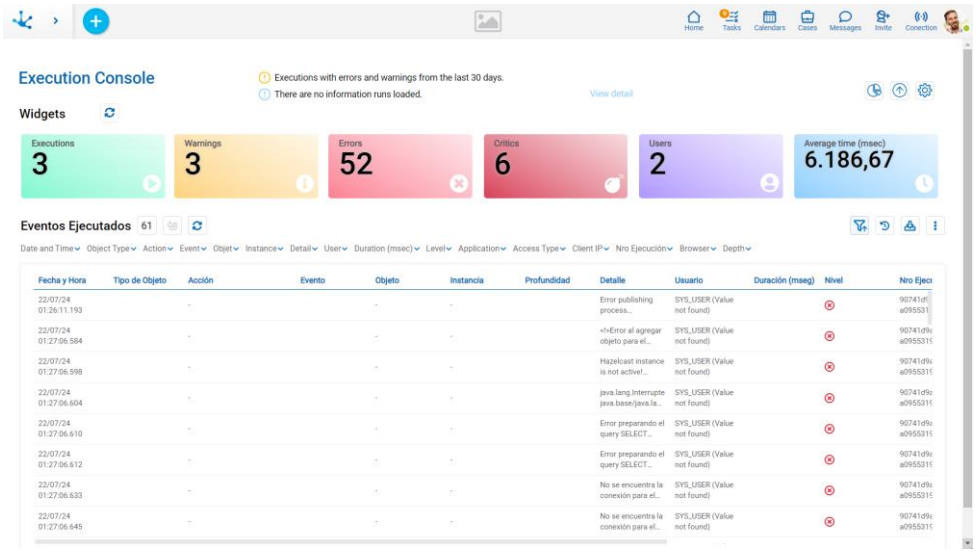
Starting an execution can generate executions on other objects. For these cases, the start and end events of these executions are presented nested within the start and end events of the initial execution.

The console incorporates indicators that provide an overview of how many executions were performed, how many users participated, whether there were warnings for slow executions or errors for a given time range.

To deepen the analysis, the execution event grid that has different search criteria can be used, where the user can filter by applications, by object type, by user, by IP address, as well as by the content of the modeled embedded rules. So that the user can have complete detail of how, when and who executed their applications.

To configure and use the console, the following steps must be performed:

- Configure the register level that determines what is going to be saved.
- Execute the objects to register the actions of the defined register level.
- Make the records loading in the records grid.
- Analyze the indicators and check the details of the records from the records grid.



When entering the console, the widgets and the grid are displayed with the records loaded on the dates indicated at the top.

These dates can be modified by entering the desired range and clicking on the icon  the widgets and the grid are updated.

The values of the widgets and the grid records include the executions made within the period indicated in the dates and times entered.
The dates and times included in the analyzed period have as default value, the time period from the oldest execution to the last one loaded into the console.
In the case that there are periods that are not loaded in the console, a message is displayed indicating which periods are missing.

> *The dates and times of the executions are shown in UTC-3.*

Widgets

The widgets provide an overview of the executions made for the indicated time period.

They can be hidden clicking on the icon  and then to view them again the same icon is used.

Executions
Indicates the total number of execution end events recorded for all levels.

Warning
Indicates the total of executions with end events that exceed the maximum expected duration. The duration time is defined in the environment property Maximum time.

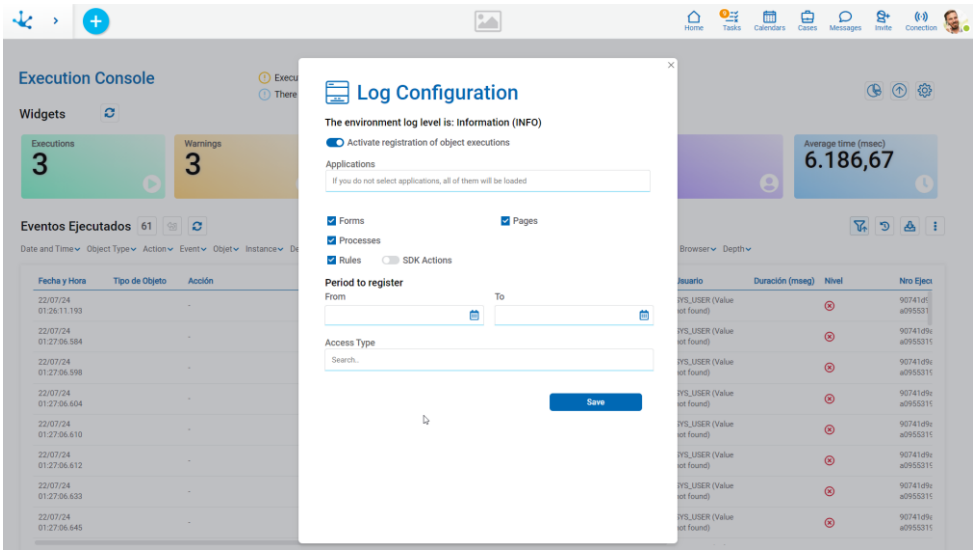
Errors
Indicates the total number of executions with end events that have an error; these errors can be ERROR or CRITICAL level.

Users
Indicates the number of users that performed the executions.

Average time
Indicates the average time of all the executions.

3.5.1.4.1. Register Level

Clicking the icon  opens a panel where the register level is configured and when the configuration is saved, the executions of the selected objects and applications will begin to be recorded, becoming available to upload to the console.



Configuration Criteria

By entering the configuration panel, a message is displayed indicating the register level that the environment has active.
This level is configured in the environment property Execution log level and can take the values "Info", "Warning" or "Error".

Activate the register of object executions

Only if the environment register level is Information (INFO), the register of executions of objects and applications can be activated.

> *If the "Warning" or "Error" level is defined in the environment, all records with the configured level are available to load into the console.*

Applications

By default all applications are registered. One or more applications can be selected.

Forms, Processes, Rules

The different objects to register depending on the active application(s) can be selected.

SDK Actions

If the rule registry is activated, the details of all the Deyel SDK methods can be included, recording for each rule executed the SDK methods used in the rule.

From, To

The dates are optional, if they are entered, only the executions in this time range are recorded.

Access type

More than one access type can be selected.
Possible values:

- WEB
- REST
- DEYEL

- SDK

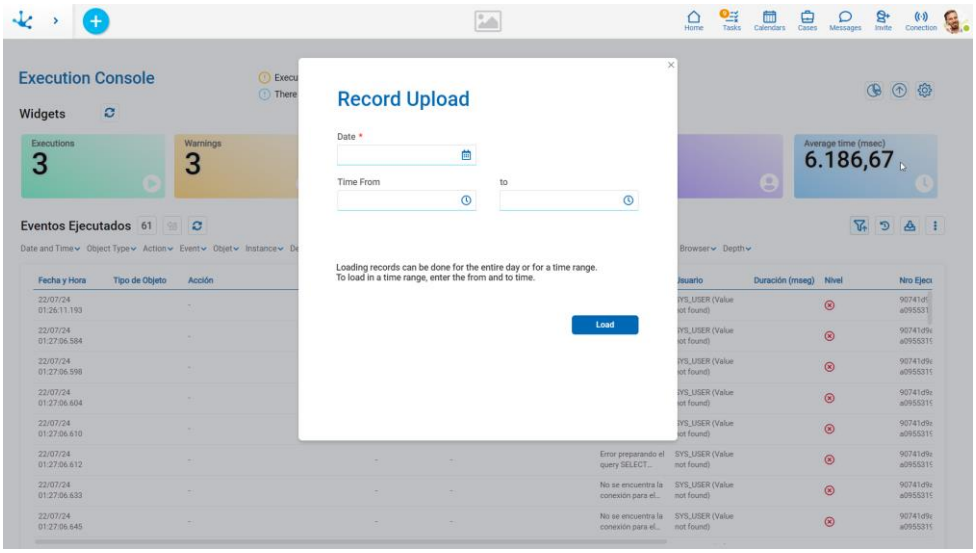
Configuration Recommendations

Depending on the type of environment being used, there are different levels of detail that are recommended to be configured.

- In the development environment, the register level with all objects enabled is used to register all the details of the executions. With such detail, failures in the executions can be detected, analyzing the errors that occurred and the performance of the executions can be evaluated.
- In the test environment, the register level with all objects enabled is used to register all the details of the executions during the test period and then disable it.The results of user tests are analyzed to proactively detect errors and performance improvements.
- In the production environment, it is recommended to configure executions at "Warning" level, this way errors can be detected and performance analyzed. If necessary for a specific situation, the level of detail of the executions for certain objects or applications can be raised for a certain time.

3.5.1.4.2. Records Loading

Clicking the icon opens a panel that allows entering the date to load the records into the grid.



Filters

Date

It is required to select the date for which the execution records will be loaded.

Time from, to

The time range of the selected date can be entered, for which the execution records will be loaded.

Loading Execution

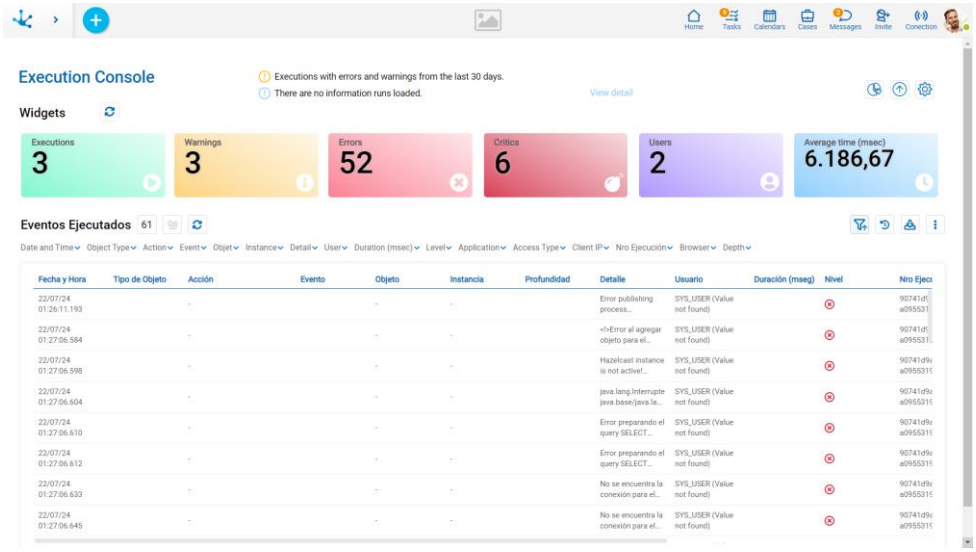
By pressing the “Load” button, the execution records are loaded into the records grid, this update may eventually take a few seconds depending on the volume of information generated.

When the records are finished loading, Deyel Bot sends a message to the user's chat to notify the completion.

The oldest date available for loading records depends on the value defined in the environment property “Log files persistence”. After this time the records to load to the console, as well as the console data, are deleted.

3.5.1.4.3. Records Grid

The records are displayed chronologically in ascending order. The user can change the initial sorting and also set different search criteria.



Columns

Date and Time

Indicates the date and time when the action event of the modeled object is executed.

Object Type

The object types can be: "Form", "Process" and "Advanced rule".

Action

The action defined for each object type.

Event

Determines when each action starts and ends (START / END). A record is displayed with “Start” and another with “End”.

Object
It is the name of the object that made the action.

Instance
Depending on the action performed, the object identifier is displayed, for example in processes it is the case number.

Detail
Includes the descriptive messages depending on the action performed, in different situations.
The descriptive message is logged for failed completion events. It can be the result of a validation or an unexpected error, in the latter case its stack is displayed.

User
It is the user that performs the action.

Duration
It is the duration in milliseconds of the action executed. Values are only displayed when the event is "Ends".

Level
Displays the type of level of the actions executed.
Possible Values:
- INFO: shows all information actions.
- WARNING: shows actions that took longer than expected.
- ERROR and CRITIC: show actions with errors.

Application
It is the name of the application of the object to which it belongs.

Access Type
It is the type of access that the user uses. This type of access can be from: Web portal (WEB), Rest API (REST), SDK or internal workflow calls (DEYEL).

IP Client
It is the IP with which the user accesses **Deyel** from the Web browser

Execution
It is a unique identifier automatically generated for each execution performed by the user and for each of the actions that may have been triggered by such execution. All actions and events generated by the action requested by the user are identified by the same value in this column.

Browser
It is the name of the browser used by the user.

3.5.1.4.4. Actions by Objects

The actions performed by modeled objects in an application are grouped by the type of object.

Form

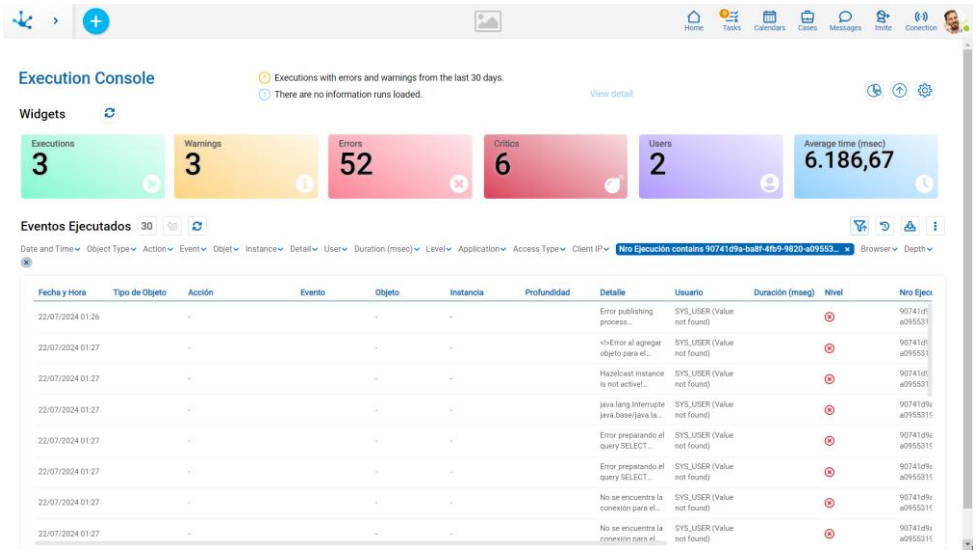
Action	Description	Detail
new	Show the form to create an instance from the web portal or the mobile application.	-
create	Create a form instance from the web portal, the mobile application, Rest or SDK.	-
read	Show a form instance from the web portal, the mobile application, Rest or SDK.	-
update	Modify a form instance from the web portal, the mobile application, Rest or SDK.	-
delete	Delete a form instance from the web portal, the mobile application, Rest or SDK.	-
search	Search of forms instances from the web portal, the mobile application, Rest or SDK.	In the start and end event

Action	Description	Detail
		the search filter used is displayed.
executeEmbed-dedRule	Execute an embedded rule modeled in the form or in the process (pro-cess activities, gateway or flows).	Embedded

Action	Description	Detail
		d r u l e s d e t a i l .
validation	Execute the validation rules of a form.	
autocomplete	Execute a call from a form field to resolve autocomplete functionality.	
filterField	Execute a call from a form field to solve a filtered field.	
<Deyel SDK para Java>	If it is a Deyel SDK execution performed from an advanced rule, the operation of the Service Class that is executed.	

Example of creating a form instance with embedded calculation and validation rules

The following image of the console shows the sequence of events generated when the user clicks on an "OK" button to create an instance of the "Purchase Request" form.



The form creation has two embedded calculation rules and one embedded validation rule.
The registration of events is initiated by the user Valentin Pereira with the “create” action for the "Purchase Request" form. As part of the creation, the 3 embedded rules are executed, they are registered in the order of execution with their start and end events.

The first embedded rule that is executed is a calculation rule and corresponds to the “total” field of the “lines” container of multiple occurrences, for its first occurrence identified with “(0)”. This rule multiplies the “quantity” field by the “unitPrice” field.
In the column of grid detail it is visualized:

```
Origin: lines/total(0), CALC, Expression: lines.quantity * lines.UnitPrice
```

Next, the embedded calculation rule for the “estimatedTotal” field is executed, which adds the “total” field of the lines.
In the column of grid detail it is visualized:

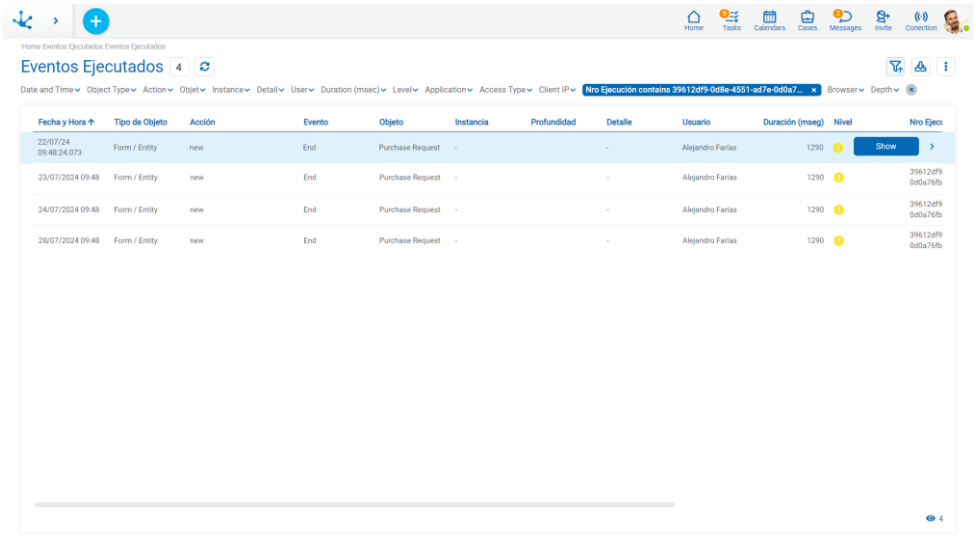
```
Origin: ”totalEstimated”, Type: CALC, Expression: sum(lines.total)
```

Finally, the embedded validation rule is executed, which evaluates that if the “unitPrice” field of the multiple occurrence container is entered, it is required to enter the corresponding “quantity” field.
In the column of grid detail it is visualized:

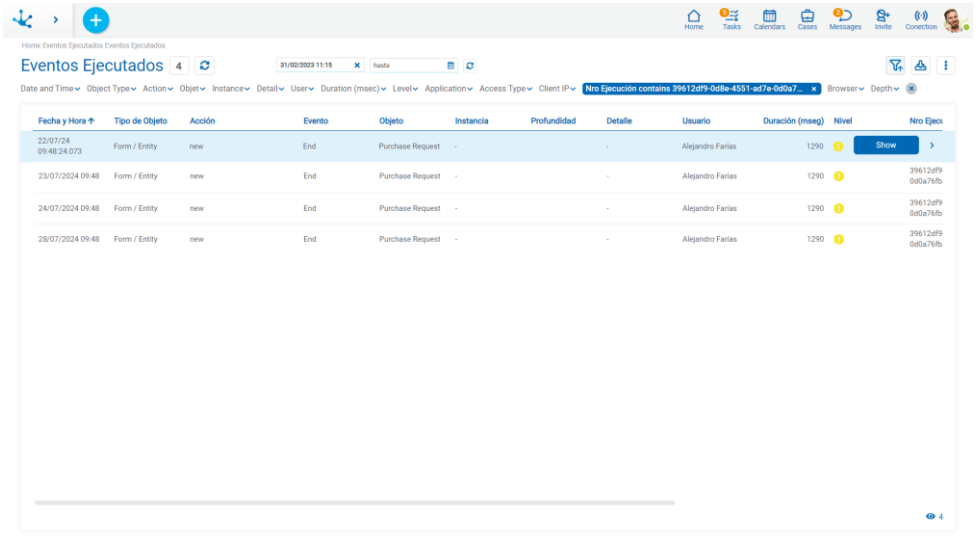
```
Origin: lines/quantity(0), COND, Expression: IF (AND (lines.unitPrice > 0, lines.quantity <= 0), "If you enter the the price you must enter the quantity")
```

The execution ends with the end event of the “create” action of the form instance. All execution records are grouped under the same “#Execution” identifier value.

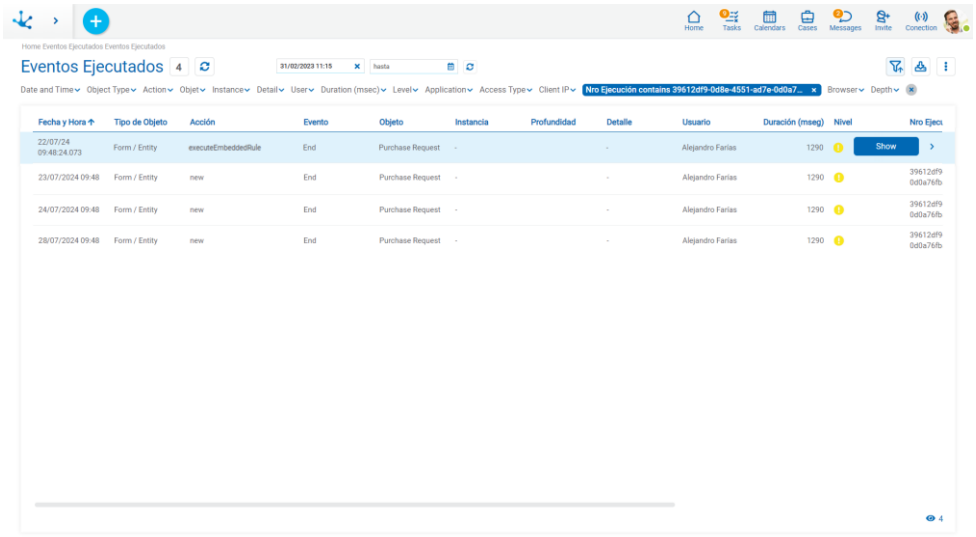
In the following image the cursor is positioned on the line corresponding to the first embedded calculation rule.



Next, the cursor is positioned on the line corresponding to the second embedded calculation rule.



Finally, the cursor is positioned on the line corresponding to the embedded validation rule.



Processes

Action	Description	Detail
new	Show the first activity.	-
showActivity	Show activity with or without data (for case creation) from the web portal, mobile app, Rest or SDK.	Activity name.
executeActivity	Execute a case activity from the web portal, mobile app, Rest, or SDK.	Activ

Action	Description	Detail
		ity name.
executeEmbed-dedRule	Execute an embedded rule modeled in activities, in logic gateways or in process flows.	Embedded rules detail.
executeGateway	Execute a logic gateway of the case.	Gateway na

Action	Description	Detail
		me.
executeAutomaticAction	Execute an automatic action, in the execution of a case activity (at the beginning or at the end of the activity).	Values of the input parameters for message

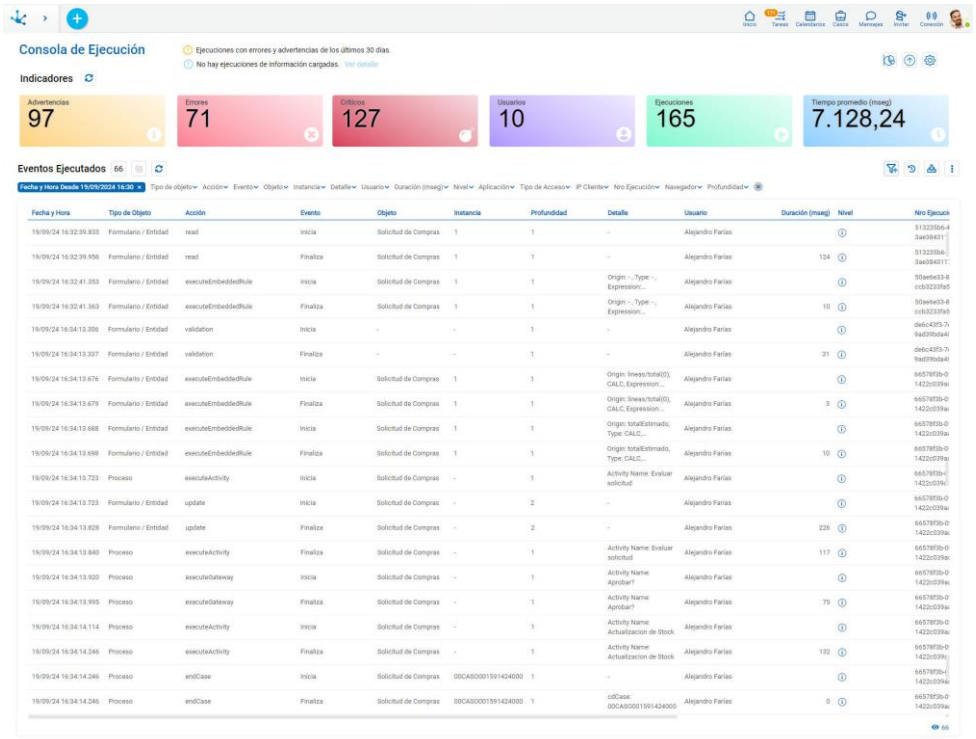
Action	Description	Detail
		sending ormail sending.
showCase	Show a case from the web portal, the mobile application, Rest or SDK.	-
showTasksList	Show tasks list.	Search filters us

Action	Description	Detail
		edin the task list.
getLastTask	Reading the last 3 tasks.	-
<Deyel SDK para Java>	If it is a Deyel SDK execution performed from an advanced rule, the operation of the Service Class that is executed.	

Example of execution of the “Evaluate request” activity with a gateway that evaluates whether it is approved or not

When the request is approved, an advanced rule is executed that updates the product stock using the “update” method of the Deyel SDK of the forms.
The example displays the sequence of events generated when the user clicks the “Approve” button to finish the “Evaluate request” activity.

The execution grid is shown filtering by the identifier “#Execution” to display the 20 events included in the execution of the activity.



The execution begins with the update of the form that corresponds to the execution of the activity "Evaluate request".

As part of the update of the form, 3 embedded rules are executed, of which 2 are calculation and one validation, as detailed in the previous example corresponding to the form.
Next, the execution of the activity begins with an automatic action that contains an advanced rule that uses Deyel SDK to read and modify the "Products" form. This is detailed in the next example corresponding to advanced rules.

The gateway evaluates whether the request is approved or not and then the execution of the condition and the activity is completed. Finally, the execution ends with a warning because the default value of 1 second for its duration is exceeded.

Advanced Rules

Action	Description	Detail
execute	Execute advanced rules on forms, processes (process activity actions or flows), scheduled tasks, SDKs, or Rest API invocations.	In

Action	Description	Detail
		the start event the input parameters are displ

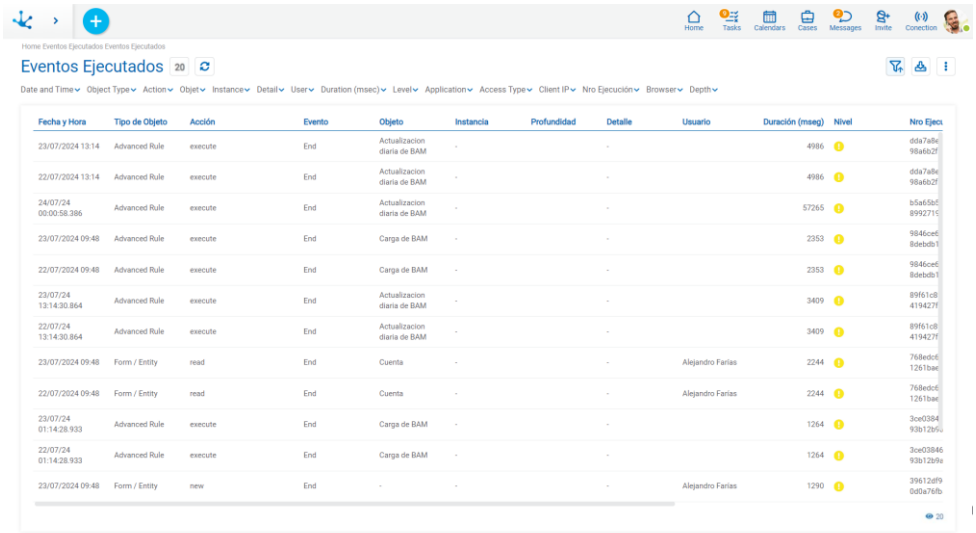
Action	Description	Detail
		ayed. In the end event the output parameters a

Action	Description	Detail
		red displayed.

Example of executing an advanced rule with Deyel SDK actions, modeled as an automatic action of the “Evaluate request” activity

The example includes the events that are executed in the “Evaluate request” activity, which initiates an automatic action that executes the advanced rule “update stock”.
In the advanced rule, an instance of the “Products” form is read by executing a “read” action and then the stock of the read product is updated by executing an “update” action.

In the following image, the “Instance” and “Application” columns were hidden in the grid so that the “Access Type” column could be displayed, without performing the horizontal scroll.



In the “Access Type” column it is seen that the activity is initiated from the user portal, following **Deyel** starts the automatic action and from Deyel SDK the “read” and “update” methods of the “Products” form are executed.

3.5.1.4.5. Embedded Rules Detail

The properties of the detail are described below.

Origin
The value can be a form field, a conditional flow, or page elements.

Type:
It is the embedded rule type.
Possible values:

- OBLIGATORY: for the requirement of a form field.
- EDITION: for the edition of a form field.
- COND: for the validation of a form field.
- CALC: for a calculation rule of a form field.
- RULE_RELATION: for a relation to an advanced rule in a form field.
- VALUE_LIST_RELATION: for a relation to a value list in a form field.
- ENTITY_RELATION: for a relation to an entity in a form field.

Expression
The modeled rule expression is displayed.

Examples

Several examples of embedded rules are included below

Requirement

The project field is required if the newProject field and task are empty.

```
{ Origin: "project", Type: "OBLIGATORY", Expression:
"Y(ISBLANK(newProject), ISBLANK(task))"  }
```

Validation

The from field must be less than the date function().

```
{ Origin: "dtBegin", Type: "COND", Expression: "IF(from > date(), "Must be
before or equal to today")"  }
```

The hours field must be greater than 0 if the totalHours field is less than or equal to 0.

```
{ Origin: "hours", Type: "COND", Expression: "IF(totalHours <=0, "Must in-
form the number of hours")"  }
```

If the Develop button is pressed, the assignedProgrammerCd field should be loaded.

```
{ Origin: "assignedProgrammerCd", Type: "COND", Expression:
"IF(AND(lastButtonPressed()=="Develop", ISBLANK(assignedProgrammerCd)),
"Must enter a Programmer")"  }
```

It is evaluated with an embedded rule if the owner is not white and with an advanced rule called ContainsCrmPermissions it is evaluated if it does not have a CRM license.

```
{ Origin: "", Type: "COND", Expression:
"IF(AND(NO(ISBLANK(owner)),NO(containsCrmPermissions( owner))),"The owner
does not have CRM application permissions")"  }
```

Calculation

Example of a calculation rule with an advanced rule. The autUser field obtains the authorizer by calling the following rule:

```
{ Origin: "autUser", Type: "CALC", Expression: "getAuthorizingU-
ser(requestingUser)"  }
```

10 days are added to the dateMondayWeek field.

```
{ Origin:  "dateMondayWeek", Type: "CALC", Expression: "ad-
dDays(startDate,10)"  }
```

The totalEstimated field has the sum of the iterative lines.total.

```
{ Origin: "totalEstimated", Type: "CALC", Expression: "sum(rows.total)"  }
```

Calculation in an iterative field of the form

Performs the calculation for the first occurrence of the iterative of the rows/total field where it multiplies the rows.amount field by rows.unitPrice.

```
{ Origin: "rows/total (0)", Type: "CALC", Expression: "rows.amount *
rows.unitPrice"}
```

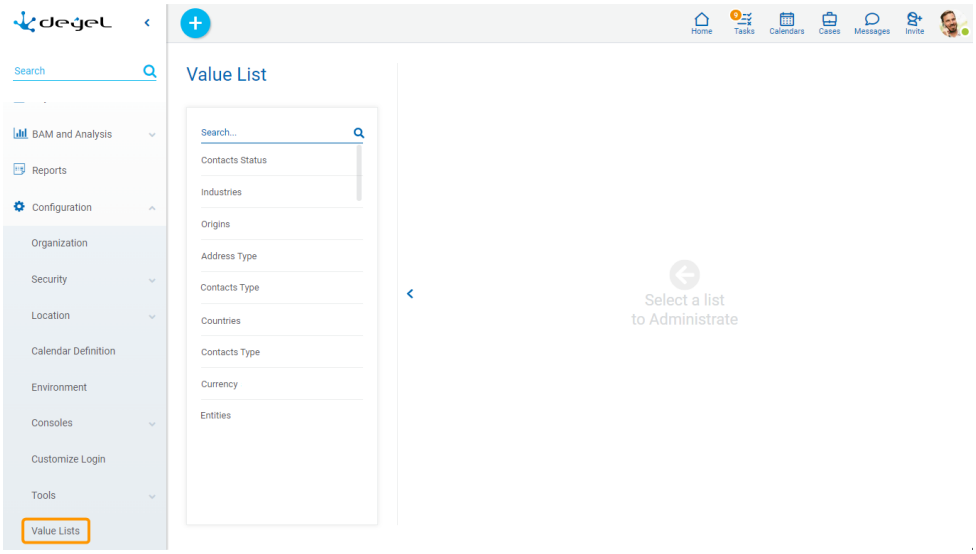
Performs the calculation for the second occurrence of the iterative of the rows/total field where it multiplies the rows.amount field by rows.unitPrice.

```
{ Origin: rows/total(1)", Type: "CALC", Expression: " rows.amount *
rows.unitPrice"}
```

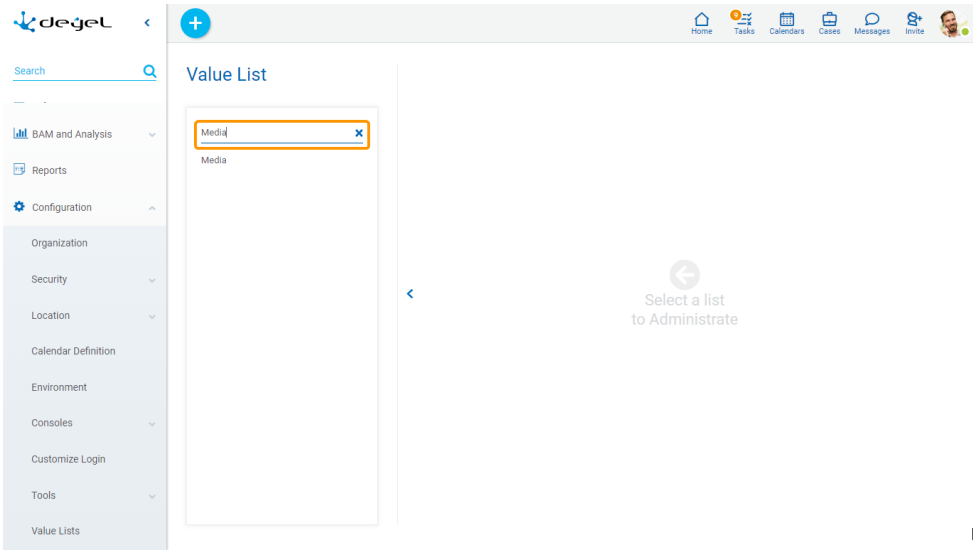
3.5.1.5. Value Lists

The "Value Lists" option of the menu allows application users to administrate the lists values, if they have the corresponding permissions assigned.

The names of the value lists in the different applications are displayed.



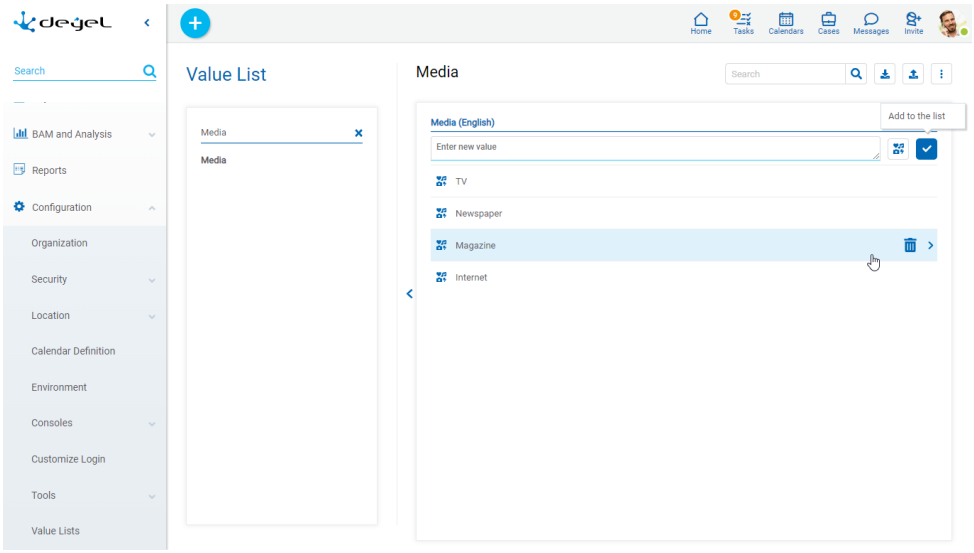
Allows to filter values from the list based on the characters entered. If a list is very long it helps users to easily visualize the desired values.



When selecting a value list, the values of the selected list are displayed in the right panel, thus making it possible to search, enter, delete and restore their values.

These updates are reflected in the value lists used:

- In the execution of forms from the user portal.
- In the forms modeler.
- In the value lists modeler.



It allows to export the value list in an Excel file. This file contains the following columns:

- CODE: Internal code of the list values.
- VALUE: Contains the list values if the Internationalize property is not active.
- ORDER
- DELETED: Indicates if it has low logic.
- FILTER_CODE: Contains the code that establishes the correspondence between the value of the list being downloaded and the list that filters it.

If the Icons property is active, values are reported in the following columns:

- ICON_COLOR
- ICON

If the Internationalize property is active, values are reported in the following columns:

- VALUE SP-AR: Contains the values of the list in Spanish.
- VALUE EN-US: Contains the values of the list in English.
- VALUE PT-BR: Contains the values of the list in Portuguese.

It allows to import a value list from an Excel file. A panel opens where the location and name of the same is selected.

It enables an option that allows displaying the internal code of the list values. If the Internationalize property is active, this enables the available languages for which the translation of the values can be entered.

Conditions for Export and Import

- The download file format must be respected.

- New values must have a blank Code column.
- Blank rows must be ignored.
- Partial downloading and uploading of values is not allowed.

Operations on Values

- Allows adding each entered value to the list of values.
- It is displayed if the list has the Icons property modeled. It allows to associate icons to the list values.

- Double click: Allows to modify a value in the list.
- Move: Allows to change the position of a value within the list by dragging the value with the mouse.

Hovering the cursor over each of the values entered, a set of icons is displayed and this allows to perform different operations.

- Allows to delete a value from the list of values. Once deleted, it is displayed in gray and crossed out.
- Allows to restore a previously deleted value.

Display the Selected Line

- Hides the icons that are displayed.
- Shows hidden icons.

Internationalization of Values

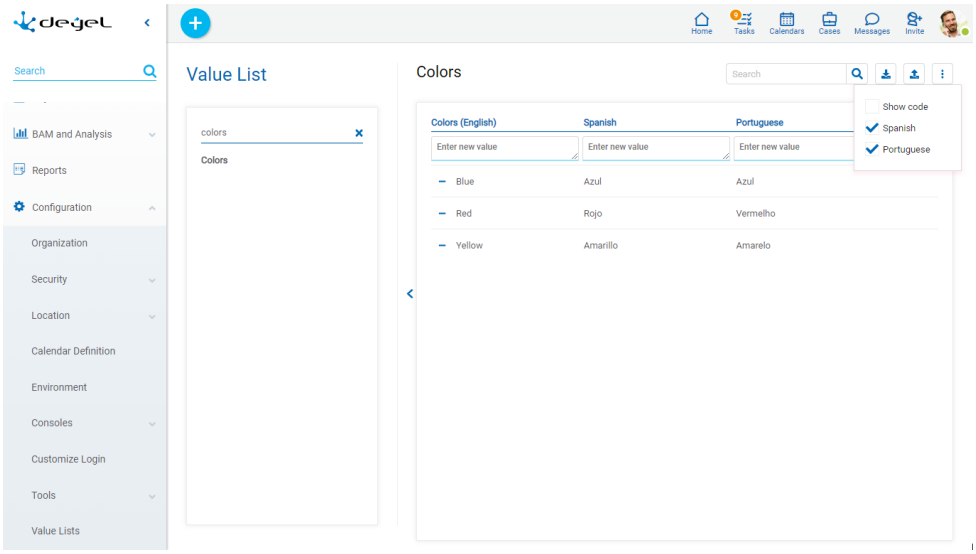
Using internationalized value lists enables to load their values in the different available languages, so that users with different language preferences configured can see list values in their own language.

If the value list has the Internationalize property active, the language configured for the user is displayed in the header of the list. For each language selected from the icon a new column is added.

The user can load the corresponding translation for each value.

Removing the check mark of a language deletes the column corresponding to that language.

For each available language, a new column is added to the Excel file, which is used to import or export the list values.



> *Once the internationalization has been modeled, it is suggested to load values to the list in all the selected languages.*

3.5.1.6. Reports

Reports allow users to get detailed information on one or more related entities, with different output presentations.

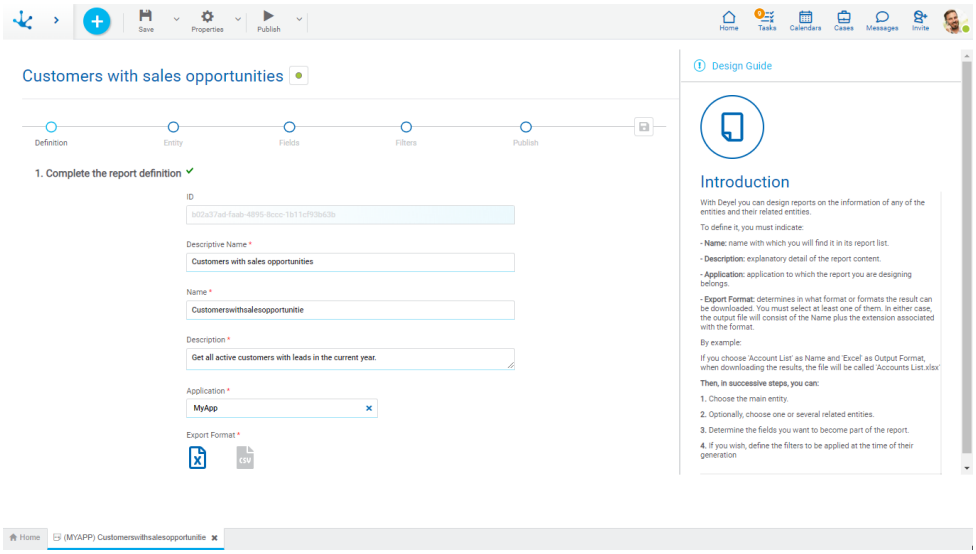
Their design has a self-explaining wizard for the selection of the main entity and its related entities, its fields, filters, presentation and permissions.

The execution of reports can be done in the foreground or be received by chat or email.

Reports can be:

- Application Reports
  They are modeled and published through the reports modeler, by IT modeler users. They can be customized and executed by all users from the portal grid.
  An application report can be saved by a user with a different name using the "Save As" operation and it automatically becomes that user's own report.

- User Reports
  All users can design and execute their own reports using customized information and output format preferences from the portal grid.
  They can use a wizard equivalent to the one used by the reports modeler by modeling a subset of properties and using the facility to save the report instead of publishing it.
  A user report can be saved with the same name using the "Save" operation or with a different name using the "Save as" operation.
  This type of report is only visible in the grid of the user who created it, that is, it can only be modified and executed by such user.



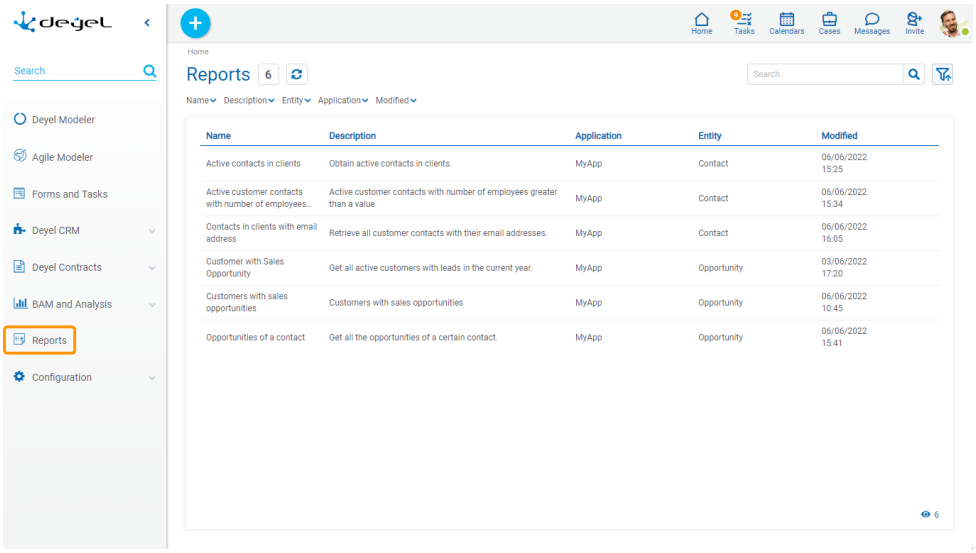
3.5.1.6.1. Grid

To create and perform operations on reports, access the grid from the menu of **Deyel**.

- From the "Reports" option.
- From the search facility.

Reports are displayed in the form of a grid, with the standard presentation of the results grid, using the following facilities:

- Search and Filters
- Operations



The following report properties are displayed as grid columns:

- Name

- Description
- Application
- Entity
- Modified

Filters can be applied by the following properties:

- Name
- Description
- Entity
- Application
- Modified

It is possible to perform operations on each row of the reports grid. Through icons  ,  and  its update, generation or deletion is performed respectively, depending on the type of report and the security permissions.

3.5.1.6.2. Operations

The user can perform operations on each report depending on the type of report and the security permissions they have defined. Hovering the cursor over each row of the grid , icons corresponding to the available operations are displayed.

With the exception of the operation "To create", which can only be done for user reports. It can be done from the context menu in different ways.

- Hover over the icon  and select the icon to your right  corresponding to the option "Report".
- Click on the icon  and select the option "Report" in the expanded vertical panel.

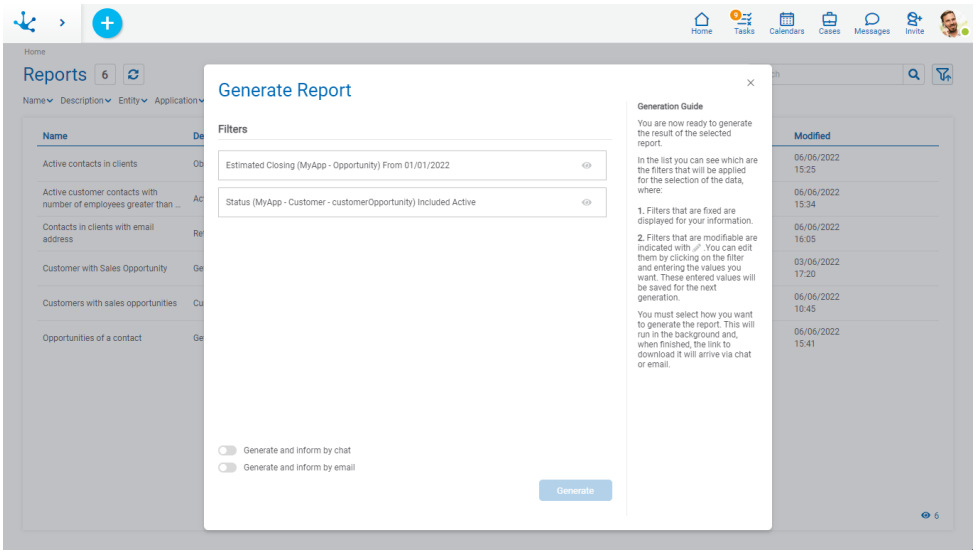
Update

By clicking the icon  in the line of the selected report, the modeling wizard opens for the user, allowing updates to be made to the values entered for all steps.

If it is an application report, the user has the possibility to save it with another name, while if it is a user report, they can save it with the same name or change it. They can also update it, execute it and not save the changes. Updates to application reports cannot be published by the business user.

Generate

By clicking the icon  to generate a report, a panel opens where the modeled filters for data selection are displayed.



A build guide is displayed on the right side of the panel to assist the user.

Filters that were modeled as editable are indicated by the icon and they can be updated before generating the report and the new filter criteria entered are defined for the report. On the other hand, filters that were modeled as non-editable can only be informed.

- Generate and inform by chat
  If this option is selected, the user receives a notification from Deyel Bot with a compressed file to be downloaded, containing the report in the modeled format.

- Generate and inform by email
  If this option is selected, the user receives an email with a compressed file, containing the report in the modeled format.

Clicking the "Generate" button starts the execution of the report in the background and at the end the user is informed according to the selected method.

An icon on a report row in the grid indicates that it is executing and if this icon is pressed, the report can be stopped.

The file name that the user receives is made up of the name of the report concatenated with the extension associated with the selected format.

Delete

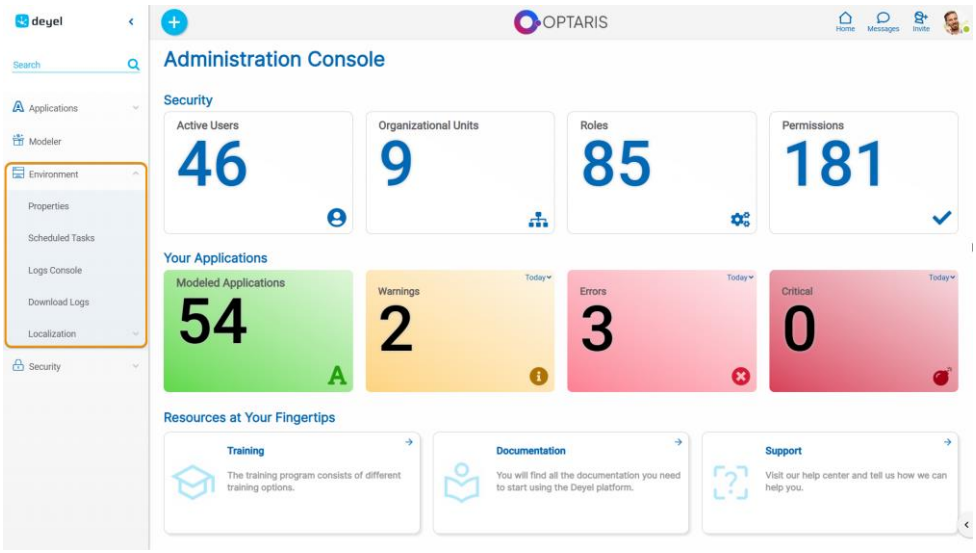
By clicking the icon , a confirmation window is displayed on the selected report line and upon accepting, the user receives a message indicating that the report has been deleted.

Only user reports can be deleted.

3.5.2. Environment

It groups the functionalities intended for the configuration and administration of the execution environment, allowing the management of technical aspects and general properties that impact the operation of applications.

- Properties
- Scheduled Tasks
- Log Console
- Location



3.5.2.1. Properties

In the **Deyel** environment, a set of properties can be defined to adapt the operation of the product according to specific needs of the client.

Each user may select the values for some properties using the facility "My Preferences".

Precedence for properties is as follows:

- Preferred value of the connected user.
- Value configured in the environment.
- Default value of the property.

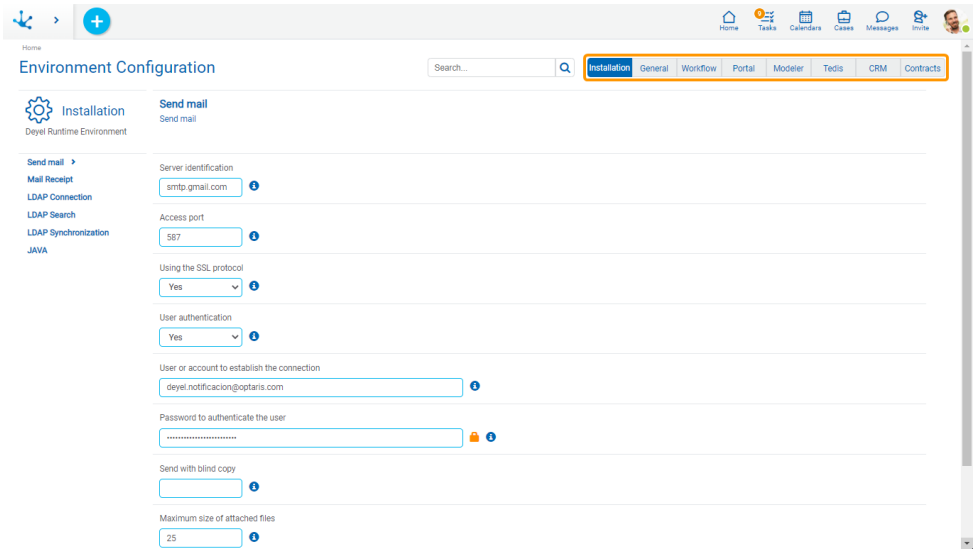
Categories

The tabs at the top right define the groups into which the configurable properties are categorized.

- Installation
- General
- Workflow
- Portal
- Modeler

- Tedis
- CRM

Selecting each tab expands a menu for each category, containing options that group properties by theme.



Configurable Properties

For each property, its descriptive name and current value are displayed, which can be modified by the user. In some cases this value can be entered and sometimes it is selected from a list of possible values.

All properties have a default value, which can be null.

When the user updates a property value, they can press the "Undo Change" button to cancel the modification. This option to undo a change is available until the "Apply Configuration" button is pressed. This is when all the properties are stored in the repository of **Deyel** and the user is notified with a successful operation message.

When a property has a value other than the default value, the "Remove preference" button is displayed in order to change the current value to the default value.

Some properties that can be updated dynamically but in other properties the new value is applied when the environment is restarted.

Properties that require the environment to be restarted are identified by the icon .

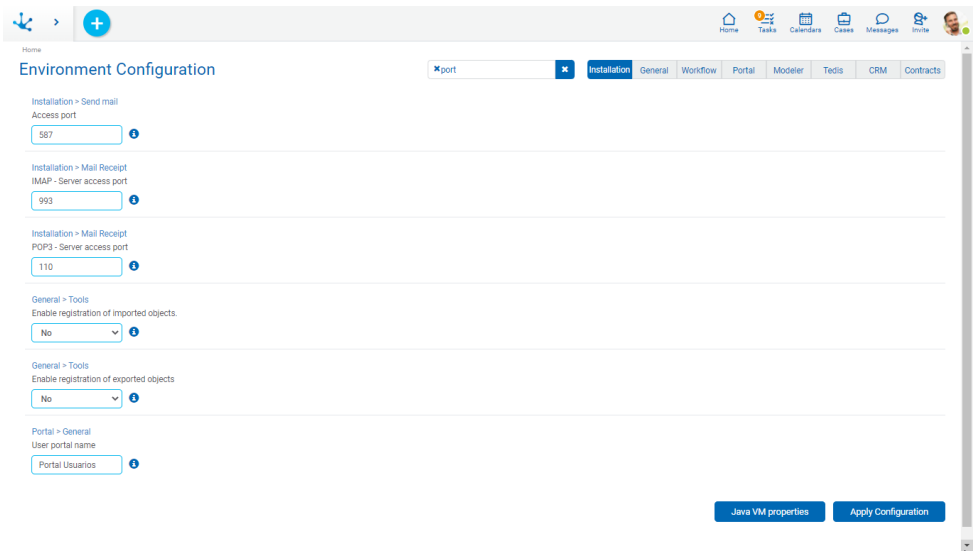
Some properties are encrypted and their content is displayed protected. They are displayed with the icon , for example user passwords.

Pressing the icon gives access to a detailed description of the property.

Search

The search facility can be used indicating the code, name or description of the property sought, in full or in part.

A breadcrumb is displayed on each one of the properties that indicates to which category the property belongs. This breadcrumb can be used for positioning in the indicated category.



Java VM Properties

They are displayed by pressing the "Java VM Properties" button and correspond to the Java Virtual Machine of the application server in use, for example: IBM Websphere or Apache Tomcat. These properties cannot be updated and depend on the Java version that is being used, for example Java for Linux, Java for Windows, etc.

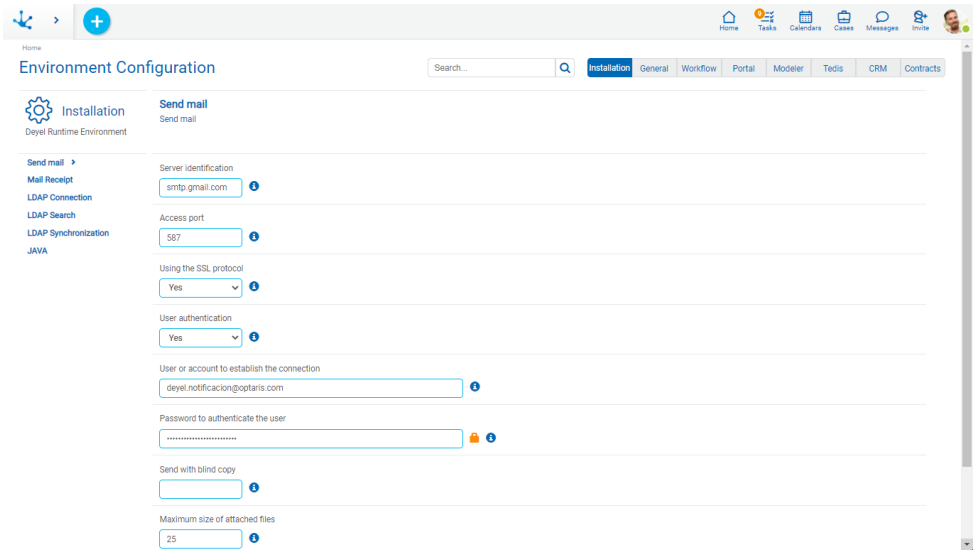
3.5.2.1.1. Installation

It allows configuring the execution environment properties of **Deyel**.

These are properties that establish how the product is installed and how it integrates with other software components, for example, with the email server or with an LDAP server.

Options

- Sending Mail
LDAP Connection
- LDAP Search
- LDAP Synchronization
- Java



3.5.2.1.1.1. Sending mail

Access to mail server and account used to send emails.

> *IMPORTANT.*
> *If the **SendMailServer** property is not informed, **Deyel** does not send notifications by mail.*
> *When the property is informed, **Deyel** tries to connect to the mail server and authenticate using the indicated username and password.*
> *In this way, the administrator is assured that when configuring, access to the mail server works correctly.*

Configurable Properties

Server identification

Indicates the name or IP address of the mail server used for sending Emails. For example, your.domain.com

Name	ServerSendMail
Code	MAIL_SERVER
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—

Dynamic	✔
Encrypted	—
Default Value	your.domain.com

Access Port

Indicates the access port to the mail server. For example, 25.

Name	PortSendMail
Code	MAIL_SERVER_PORT
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	—
Default Value	25

Using the SSL protocol

Determines if you use SSL for sending emails (enable for Gmail or similar accounts).

Name	SSLEmail
Code	SMTP_USES_SSL
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	—
Possible Values	• Yes • No (Predetermined)

User authentication

Indicates whether the mail server requires user authentication.

Name	AuthenticationSendMail
Code	MAIL_SERVER_ AUTHENTICATE
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	—
Possible Values	• Yes (Predetermined) • No

User or account to establish the connection

User or account with which the mail is sent.

Name	UserSendMail
Code	MAIL_SERVER_USER
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	—
Default Value	your@user.com

Password to authenticate the user

Password used to authenticate the user.

Name	PasswordSendMail
Code	MAIL_SERVER_PASSWORD
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	✔
Default Value	

Send with blind copy

Enter the email addresses (separated by ';') to which a blind copy of each email sent by **Deyel** should be sent.

> *Emails sent to deliver portal access keys to users are excluded.*

Name	SendWithBlindCopy
Code	BCC_ADDRESSES_FOR_OUTGOING_MAIL
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	—
Default Value	

Maximum size of attached files

Sets the maximum size in MB of the attached files in an email. If multiple files are attached, their total size cannot exceed this value. The maximum value allowed for this property is 50MB.

Name	MaximumSizeOfAttachedFiles
Code	EMAIL_ATTACHMENT_SIZE_LIMIT
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	—
Default Value	25 Mb

Sending emails

When email sending is enabled, **Deyel** records every email in a mailing list. The scheduled task of sending emails is responsible for connecting to the server and sending each of the emails registered in the list. When sending is disabled, **Deyel** does not update the list and therefore no new emails are sent.

> *This property is intended to be used in development environments in which sending emails is not necessary.*

Name	SendingEmails
Code	EMAIL_SENDING
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	—
Default Value	• Yes (Predetermined) • No

3.5.2.1.1.2. LDAP Connection

Configuration of access to LDAP server. All properties are required to be able to activate the LDAP authentication.

Configurable Properties

URL to connect to the LDAP server

It is required in order to enable LDAP authentication.
Example: ldap: // localhost: 389 /

Name	LDAPUrl
Code	LDAP_PROVIDER_URL
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	—
Default Value	

User to connect to LDAP

Identification of the user with which Deyel accesses LDAP to make the queries.
The distinguished name (DN) of that user is indicated. Example: cn

Name	LDAPUser
Code	LDAP_USER
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔

Encrypted	—
Default Value	

Password for connection to LDAP

Name	LDAPPassword
Code	LDAP_PASSWORD
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	—
Default Value	

3.5.2.1.1.3. LDAP Search

Configuration of user search within the LDAP directory.
Sets the search subtree, the LDAP attributes that are considered search keys, and additional user selection filters.

Configurable Properties

LDAP directory root node

Sets the root node of the LDAP tree considered by **Deyel** for user search. Example: ou

Name	LDAPDirectoryRoot
Code	LDAP_ROOT_DIR
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔

Encrypted	—
Default Value	

User search filter

Deyel uses the user code and/or alias as default search fields.
This property allows to establish additional conditions to the search criteria.
To enter a condition in which operators and user attributes participate.
The value must be indicated according to the LDAP standard. For example: (objeCtClass=inetOrgPerson)

Name	LDAPUserFilter
Code	LDAP_USER_FILTER
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	—
Default Value	

User code

Indicates the LDAP attribute whose value is mapped to the **Deyel** user code.
The informed attribute must uniquely identify the user within the LDAP directory.
If this property is not informed, then the user code must be entered in **Deyel**.

Name	LDAPUsercode
Code	LDAP_USER_FIELD_cdUsuario
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔

Encrypted	—
Default Value	

User aliases

Indicates the LDAP attribute whose value is mapped to the **Deyel** user alias.
The informed attribute must uniquely identify the user within the LDAP directory.
There cannot be two users with the same alias.
If this property is not informed, then the user alias can be entered in **Deyel**.

Name	LDAPAliasUser
Code	LDAP_USER_FIELD_dsAlias
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	—
Default Value	

3.5.2.1.1.4. LDAP Synchronization

Configuring properties of **Deyel** users that are synchronized with attributes of LDAP.

Configurable Properties

Surname

Indicates the LDAP attribute whose value is mapped to the **Deyel** user last name.
If not informed, then the user last name must be entered in **Deyel**.

Name	LDAPSurnameUser
Code	LDAP_USER_FIELD_dsApellido
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—

• User	—
Dynamic	✔
Encrypted	—
Default Value	sn

Name

Indicates the LDAP attribute whose value is mapped to the **Deyel** user name.
If not informed, then the user name must be entered in **Deyel**.

Name	LDAPUsername
Code	LDAP_USER_FIELD_dsNombre
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	—
Default Value	givenName

E-mail

Indicates the LDAP attribute whose value is mapped to the **Deyel** user email.
There cannot be two users with the same email.
If this property is not informed, then the user email can be entered in **Deyel**.

Name	LDAPEmailUser
Code	LDAP_USER_FIELD_dsEmail
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔

Encrypted	—
Default Value	email

Condition

Indicates the LDAP attribute whose value is mapped to the **Deyel** user state.
If not informed, then the user state can be entered in **Deyel**.

Name	LDAPUserState
Code	LDAP_USER_FIELD_dsActivo
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	—
Default Value	

Telephone

Indicates the LDAP attribute whose value is mapped to the **Deyel** user phone.
If not informed, then the user phone can be entered in **Deyel**.

Name	LDAPPhone
Code	LDAP_USER_FIELD_dsTelefono
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	—
Default Value	

Organizational Unit

Indicates the LDAP attribute whose value is mapped to the **Deyel** user organizational unit.
The value must be an existing unit code in **Deyel**.
If not informed, then the user unit must be entered in **Deyel**.

Name	LDAPUnitUser
Code	LDAP_USER_FIELD_cd_oRG_Unit
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	—
Default Value	

3.5.2.1.1.5. JAVA

Java virtual machine used by **Deyel** to compile advanced rules.

Configurable Properties

JDK Installation Directory

Used for online compilation of advanced rules.

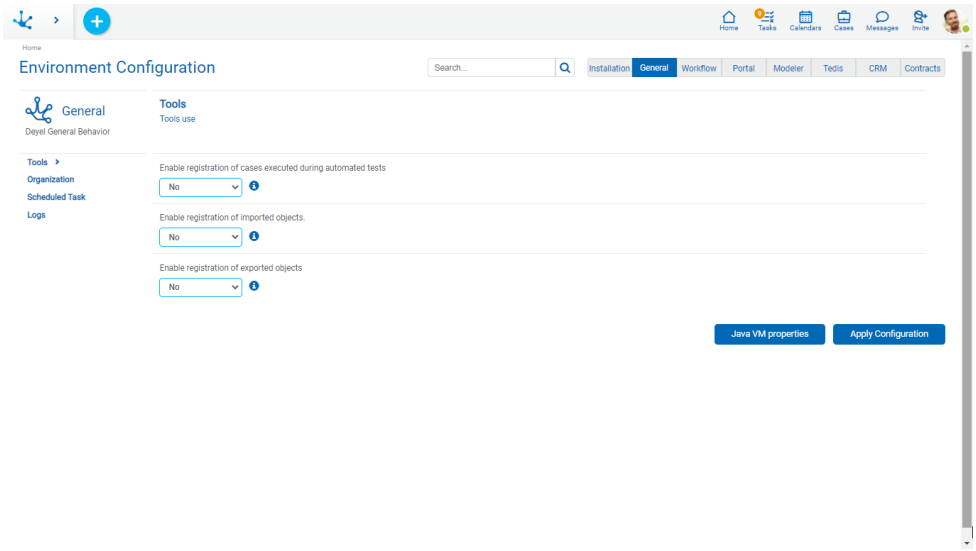
Name	DirectoryInstalationJDK
Code	JDK_HOME
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	—
Default Value	

3.5.2.1.2. General

It allows configuring properties that define the general behavior of **Deyel**.

Options

- Tools
- Organization
- Scheduled Tasks
- Logs
- Content Security Policy (CSP)
- Control de Acceso HTTP (CORS)



3.5.2.1.2.1. Tools

Configuring properties related to the use of **Deyel** tools.

Configurable Properties

Enable registration of cases executed during automated tests

When the automated test cases are executed, this property determines whether the information of each executed case should be saved in the User Case Execution (UEC) form.

Name	EnableCaseExecutionRegister
Code	ENABLE_STRESS
Configuration Levels	
• Installation	✔
• Application	—

• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	—
Default Value	• Yes • No (Predetermined)

Enable registration of imported objects

By enabling this property, each import operation that is carried out is recorded in the Import Registration form and can be viewed in detail.

Name	ImportRegistration
Code	REGISTER_IMPORTS
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	—
Default Value	• Yes • No (Predetermined)

Enable registration of exported objects

By enabling this property, each export operation that is carried out is recorded in the Export Registration form and can be viewed in detail.

Name	EnableRegisterExported Objects
Code	REGISTER_EXPORTS
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—

• User	—
Dynamic	✔
Encrypted	—
Default Value	• Yes • No (Predetermined)

3.5.2.1.2.2. Organization

Configuring properties related to the administration of the organizational structure.

Configurable Properties

Organizational unit coding type

It allows to choose the way in which the code of the organizational units is generated.

Name	TypeCodingUnits
Code	AUTOMATIC_ORG_UNIT_ID
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	—
Possible Values	• Manual • Semi Automatic • Automatic (Default)

Possible Values

- Manual: Must be informed by the user.
- Semi Automatic: Can be informed by the user. If it is not informed it is coded automatically.
- Automatic - The code is assigned by the **Deyel**.

3.5.2.1.2.3. Scheduled Tasks

These properties define the automatic start of scheduled tasks and the parameters used by specific scheduled tasks of **Deyel**.

Configurable Properties

Automatic start of scheduled tasks

Defines whether when starting the execution of **Deyel** automatically starts the execution of scheduled tasks.
When No is indicated, scheduled tasks must be started manually, by accessing the scheduled tasks monitor.

Name	HomeAutomaticProgrammedTasks
Code	TASK_SCHEDULER
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	—
Possible Values	• Si (Predeterminado) • No

Activation period for the debugger for temporary files associated with forms and cases

Activation interval (expressed in seconds) of the debugging process for files uploaded to the repository.
This is one of the system's scheduled tasks, whose execution periodicity is defined with this property.

Name	PeriodActivationDebuggerArchives
Code	CLEAN_UPLOADED_FILES_PROCESS_PERIODICITY
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	—

Default Value	18.000 (5 hours)

Minimum backup time considered by the debugger for temporary files associated with forms and cases

Minimum time (expressed in seconds) during which a file uploaded to **Deyel** cannot be deleted.
Even if the file purging process detects that the file is not related to any case or form, will not be deleted until the time indicated in this property has elapsed.

Name	MinimumTime
Code	UPLOADED_FILES_MIN_TIME
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	—
Default Value	3.600 (1 hour)

Persistence of scheduled task execution history

The execution history of scheduled tasks is cleared daily.
This property sets the number of days that must pass before a record from the history can be deleted.
It can take values between 1 and 30.

Name	PersistenceScheduledTaskExecutionHistory
Code	SCHEDULE_TASK_HISTORY_OLD
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	—

Default Value	5

3.5.2.1.2.4. Logs

Configures the automatic generation and debugging of log files.

Configurable Properties

Directorio de registro de logs

Directorio en el cual se generan los archivos de log.
Si no se informa valor, se utiliza el directorio **/logs**, dentro del contexto.

> *Solamente se puede configurar en instalaciones On-Premise*

Name	LogDirectory
Code	LOG_DIRECTORY
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	—
Encrypted	—
Default Value	

Log files persistence

The generated log files are automatically deleted by the system's log purging process.
This property sets the number of days that should persist, in order to delete them.

> *In Cloud environments, this property must assume values greater than zero and less than 30.*
> *In On-Premise environments, a value greater than 30 days can be indicated. If the value is zero, files are never deleted.*

Name	LogFilesPersistence
Code	LOG_FILES_DAYS_OLD
Configuration Levels	

• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	—
Default Value	15

Execution log level

Indicates the type of information that **Deyel** records in the log console and in the execution console.

Name	LogLevelEnabled
Code	LOG_DATA_LEVEL
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	—
Default Value	• NONE • ERROR • WARNING • INFO (Predeterminado)

- NONE
No information is logged.

- ERROR
Errors are logged.

- WARNING
Errors and warnings are logged.

- INFO
Errors, warnings, and event information are logged.

Maximum execution time of business rules

Defines a time, expressed in milliseconds. When the execution of a business rule exceeds this threshold, input and output parameters are saved in a log file. If the defined value is 0, the execution time of the rules is not controlled.

Name	RulesTimeMaximumExecution
Code	LOG_LOGIC_MAX_TIME
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	—
Default Value	1000

Information persistence in the logs console

The information stored in the logs console is automatically removed by the environment log console scrubbing process. This property defines the amount of hours that the information must persist, before being deleted. If the value is zero, the information is never deleted.

Name	LogConsolePersistence
Code	LOG_DATA_OLD
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	—
Default Value	372

3.5.2.1.2.5. Content Security Policy

Additional security layer that helps prevent and mitigate some types of attacks, including Cross Site Scripting.

All configurable directives can be filled according to specification https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Security-Policy/Sources#sources

Configurable Properties

Directive default_src

The default-src directive serves as a fallback for the other CSP directives. For each missing directive, the user agent looks up the directive and uses its value.

Name	DirectiveDefault_Src
Code	DEFAULT_SRC
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	—
Default Value	

Directive style_src

The directive specifies valid sources for style sheets. If this directive is absent, the user agent will look for the default-src directive. One or more sources are allowed for the style-src directive.

Name	DirectiveStyle_Src
Code	STYLE_SRC
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	—
Default Value	

Directive font_src

The directive specifies valid sources for the loaded text fonts. If this directive is absent, the user agent will look for the default-src directive. One or more sources are allowed for the font-src directive.

Name	DirectiveFont_Src
Code	FONT_SRC
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	—
Default Value	

Directive script_src

The directive specifies valid JavaScript sources. If this directive is absent, the user agent will look for the default-src directive. One or more sources are allowed for the
script-src directive.

Name	DirectiveScript_Src
Code	SCRIPT_SRC
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	—
Default Value	

Directive img_src

The directive specifies valid image sources. If this directive is absent, the user agent will look for the default-src directive. One or more sources are allowed for the
img-src directive.

Name	DirectiveImg_Src
Code	IMG_SRC
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	—
Default Value	data

Directive frame_src

The directive specifies valid sources for loading nested browsing contexts using elements like <frame> and <iframe>. One or more sources are allowed for the frame-src directive.

Name	DirectiveFrame_Src
Code	FRAME_SRC
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	—
Default Value	

Directive frame_ancestors_src

The directive specifies valid sites that can embed the **Deyel** portal using <frame>, <iframe>, <object>, <embed> o <applet>. It differs from the frame-src directive as the latter specifies where iframes can be loaded from in the **Deyel** portal. One or more sources are allowed for the frame-ancestor-src directive.

Name	DirectiveFrame_ancestors_Src
Code	FRAME_ANCESTORS_SRC
Configuration Levels	

• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	—
Default Value	

Directive form_action_src

The directive determines the URLs that can be used from **Deyel** as destination in the <form> tags of HTML. One or more sources are allowed for the form-action-src directive.

Name	DirectiveForm_action_Src
Code	FORM_ACTION_SRC
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	—
Default Value	*.google.com

Directive connect_src

The directive determines the URLs that can be loaded via script interfaces. The APIs not allowed in the directive are: <a> ping, fetch(), XMLHttpRequest, WebSocket,EventSource and Navigator.sendBeacon(). If this directive is absent, the user agent will look for the default-src directive. One or more sources are allowed for the connect-src directive.

Name	DirectiveConnect_Src
Code	CONNECT_SRC
Configuration Levels	
• Installation	✔
• Application	—

• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	—
Default Value	https://www.cloudflare.com/cdn-cgi/trace data

3.5.2.1.2.6. HTTP Access Control

Mechanism based on HTTP headers that allows loading resources in browsers from a domain, HTTP schema or port of origin other than the one owned by **Deyel**.
The configurable directive can be filled according to the following specification
https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS

Configurable Property

Access_Control_Allow_Origin Directive

The directive indicates which origin can access the resource. Only one origin can be specified.

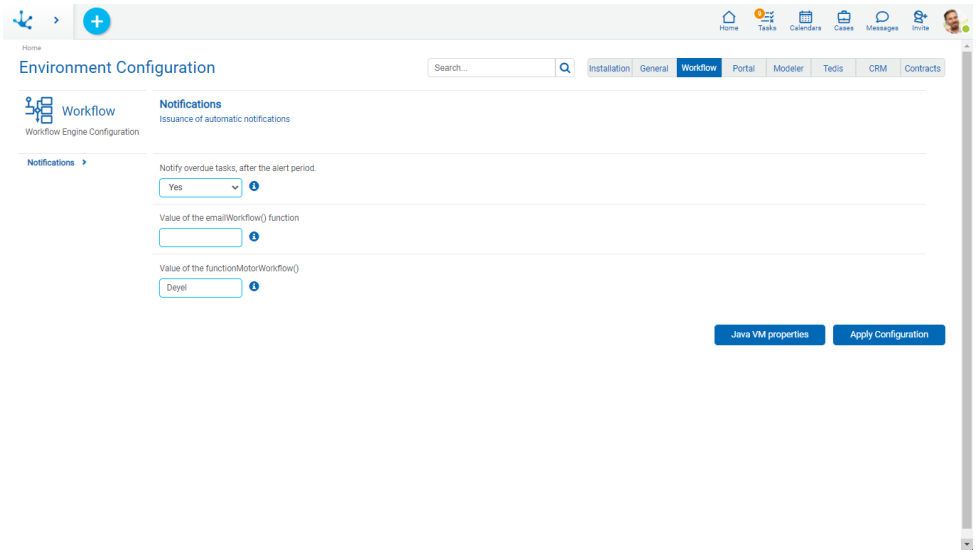
Name	Access_Control_Allow_Origin
Code	ACCESS_CONTROL_ALLOW_ORIGIN
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	—
Default Value	

3.5.2.1.3. Workflow

It allows configuring the properties related to the workflow engine.

Options

- Notifications



3.5.2.1.3.1. Notifications

Configuring automatic notifications sent by the workflow engine.

Configurable Properties

Notify overdue tasks, after the alert period.

Indicates if an email should be sent for overdue or expired alerts. When the scheduled task for sending alerts runs because it detects an overdue task whose warning period has expired and which has not been previously processed, it sends an email only if this property is activated.

Name	NotificationAutomaticActionsCancelled
Code	ADDRESSES_FOR_CANCELED_ACTIVITY
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	—
Default Value	

Value of the emailWorkflow() function

When defining automated email actions or alerts, the emailWorkflow() function can be used to model the notification sender. When the mail is received, the modeled value is displayed as sender, instead of displaying the actual sender, which is the one configured in the UserSendEMail property.It may happen that some mail servers, such as Google for example, do not allow displaying a different value from the actual sender.

Name	EmailWorkflow
Code	WORKFLOW_MAIL
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	—
Default Value	deyel@deyel.com

Value of the functionMotorWorkflow()

The functionWorkflowMotorName() returns an alphanumeric value indicating the name of the Workflow engine. It is commonly used in notification modeling. The default value is Deyel Workflow Engine, but a different value can be set if necessary.

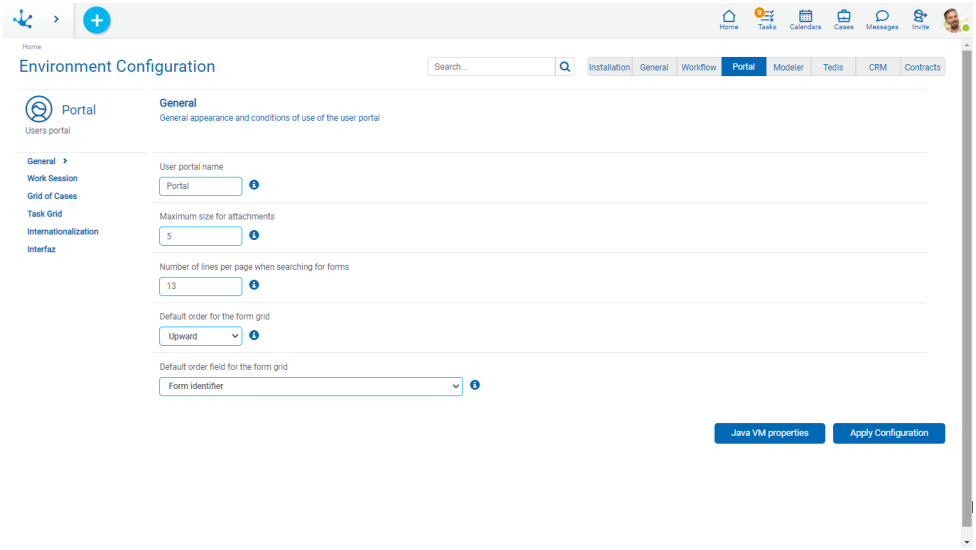
Name	workflowEngineName
Code	WORKFLOW_NAME
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	—
Default Value	Deyel Workflow Engine

3.5.2.1.4. Portal

It allows configuring properties that define the general appearance and behavior of the user portal.

Options

- General
- Work Session
- Cases Grid
- Tasks Grid
- Internationalization



3.5.2.1.4.1. General

General appearance and use conditions of the user portal.

Configurable Properties

User portal name

Text displayed in the browser tabs, preceding the user data. For example, **Deyel** or Deyel Portal. It allows to quickly identify the browser tabs that are accessing the portal.

Name	PortalName
Code	SYSTEM_NAME
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔

Encrypted	—
Default Value	Portal

Maximum file size stored in database

When the user attaches files to the system, either as form fields or as documents attached to a case, their size is controlled not to exceed the limit established in this property.
A number is indicated, expressed in MB, which must be between 0 and 20.

Name	UploadDocumentsMaximumSize
Code	MAX_UPLOAD_SIZE
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	—
Default Value	5

Number of lines per page in form search

Sets the number of lines per page when form instances are shown.

Name	SearchFormsQuantityLinesPerPage
Code	DEFAULT_PAGE_SIZE
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	—
Default Value	13

Default order for forms grid

Allows to define the default order for the forms grid.

Name	OrderForms
Code	BROWSE_FORM_ORDER
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	—
Possible Values	• Apward • Descending (Default)

Default order field for the forms grid

Allows to define the field used for the default order of the forms grid.

Name	FieldOrderForms
Code	BROWSE_FORMS_ORDER_FIELD
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	—
Possible Values	• Creation Date (Default). • Last modification. • Form identifier.

Connection monitor

Activate the connection monitor. This property is configured from the user preferences.

Name	ConnectionMonitor
Code	CONNECTION_MONITOR
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	—
Possible Values	• Yes • No (Predetermined)

Maximum file size stored in AWS S3

When the user uploads files to Amazon S3 repository, either as form fields, in Tedis conversations, or as documents attached to a case or form, the size of the file should be checked not to exceed the limit established in this property.
A number expressed in MB is indicated. The maximum value is 5000MB.

Name	MaximumUploadSizeForFilesInAmazonS3
Code	MAX_UPLOAD_SIZE_AS3
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	—
Default Value	500

Expiration time for Amazon S3 pre-signed URL

When the user uploads or downloads files from the Amazon S3 repository, pre-signed links are generated with an expiration time set in this property.

A number expressed in seconds is indicated. The maximum value is 21600.(6 hours).

Name	ExpirationTimeForAma-zonS3PreSignedUrl
Code	PRESIGNED_EXPIRATION_AS3
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	—
Default Value	10

Maximum file size to process in the environment

When files from the Amazon S3 repository are processed within the server, either to be used in a rule or downloaded from the file download endpoint, the size of the files should be checked not to exceed the limit set in this property.
A number expressed in MB is indicated. The maximum value is 25.

Name	MaximumFileSizeInAma-zonS3ToBeProcessedInTheEnvironment
Code	MAX_PROGRAMATIC_SIZE_AS3
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	—
Default Value	10

3.5.2.1.4.2. Work Session

Work session characteristics.

Configurable Properties

Authentication type

Defines the authentication method used by **Deyel**.

Name	AuthenticationType
Code	TP_AUTHENTICATION_LOGIN
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	—
Possible Values	• Native (Predetermined) • LDAP • Google • Personalized • Federated IDM • Mixed • Azure AD

Maximum inactivity time

Maximum inactivity time, expressed in minutes, to keep the user session active in the browser.
If that time is exceeded, the session expires and the user has to access the portal again.
The value -1 must be indicated so that the session does not expire.

Name	MaximumIdleTime
Code	MAX_SESSION_INACTIVITY_TIME
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	—

Possible Values	-1 (Default), indicates that sessions do not expire due to inactivity. It may contain numeric values that represent a number of minutes.

Allow multiple user sessions with the same browser

When this property is enabled, it is possible to maintain multiple work sessions, with different users, within the same browser.

Name	AllowMultipleSessions
Code	ALLOW_MULTISESSION
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	—
Possible Values	• Yes • No (Predetermined)

Mixed authentication method

These properties are used to configure the mixed authentication mechanism.
They establish which authentication methods are enabled and the order in which they should be used.

There are 3 properties that work in a similar way:

- First mixed authentication method
- Second mixed authentication method
- Third mixed authentication method

Name	AuthenticationMixta1 AuthenticationMixta2 AuthenticationMixta3
Code	AUTHENTICATION_MIXTA_1 AUTHENTICATION_MIXTA_2 AUTHENTICATION_MIXTA_3
Configuration Levels	

• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	—
Possible Values	• Native • LDAP • Google • Personalized • Federated IDM • Azure AD

When configuring the use of mixed authentication, the following should be validated:

- At least one of these properties is indicated.
- There are no repetitions in the established values.
- If any property assumes the value:

  "IDM Federated", the adapter "IDMAuthorizationCode" is validated as published.

  "LDAP", all LDAP related properties are validated.

  LDAP - Server Connection
  LDAP - User Search
  LDAP - Attribute Synchronization

  "Google", all Google related properties are validated.

  "Customized", all properties related to custom authentication are validated.

  "Azure AD", the "Azure AD" adapter is validated as published.

Configuration can only be applied when all validations are correct.

Google authentication - OAuth credentials - Client identification

To use Google's authentication services, **Deyel** must present credentials that identify it as an OAuth 2.0 client.
This property sets the Client ID, which is part of those credentials.

> *It can only be configured in On-Premise environments*

Name	GoogleClientID
Code	GOOGLE_CLIENT_ID
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	✔
Default Value	

Google authentication - OAuth credentials - Client Secret

To use Google's authentication services, **Deyel** must present credentials that identify it as an OAuth 2.0 client.
This property sets the Client Secret, which is part of those credentials.

> *It can only be configured in On-Premise environments*

Name	GoogleClientSecret
Code	GOOGLE_CLIENT_SECRET
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	✔
Default Value	

Custom authentication

Name of the rule defined to run Custom authentication.

Name	CustomAuthentication
Code	CUSTOM_AUTHENTICATION
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	—
Default Value	

The existence of the rule is validated by configuring the type of custom or mixed authentication that contains it.
If the rule exists, it is validated to have the required input and output parameters.

Authorized domains to send user invitations

The email addresses to which invitations are sent must belong to one of the domains reported in this property. Several domains can be indicated, separated by semicolons. For example: 'mycompany.com ; optis.com'. If no value is reported, then invitations can be sent to any email address.

Name	AuthorizedDomains
Code	AUTHORIZED_DOMAINS
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	—
Possible Value	They must be valid domain names. They can be separated by semicolons.

Require password change

When a user uses 'Forgot your password' and logs in for the first time with the assigned password, **Deyel** may require a password change depending on the value of this property:

- Not Required - Deyel allows the user to continue using the assigned password.
- Optional - The password change screen is presented prompting the user to update it. You can do so or indicate that you keep the assigned password.
- Required - The user is forced to change the password in order to continue.

Name	RequireChangePassword
Code	PASSWORD_CHANGE
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	✔
Possible Values	• Not Required (Default) • Optional • Required

Maximum duration of user session

Maximum duration of user session, expressed in minutes. If that time is exceeded, the session expires and the user has to log in again. By indicating the value -1 the sessions do not expire.

Name	MaximumSessionDuration
Code	MAX_SESSION_EXPIRATION_TIME
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	—
Possible Values	-1 (Default), indicates that sessions do not expire for a maximum time.

	It may contain numeric values that represent a number of minutes.

Maximum number of simultaneous sessions per user

Allows to limit the number of simultaneous sessions that a user can have active. Indicating the value -1 the number of sessions is unlimited.

Name	NumberofSimultaneousSessions
Code	MAX_SESSIONS_BY_USER
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	—
Possible Values	-1 (Default), indicates that the number of simultaneous sessions is unlimited.

Maximum number of failed logins

Sets the maximum allowable number of failed authentications due to incorrect passwords. When the user exceeds this number of consecutive failed attempts, their account becomes inactive. If the value -1 is set, there is no limitation on the number of failed logins.
This control only applies when using the native authentication mechanism, in which **Deyel** is responsible for verifying the user's password.

Name	MaximumAmountOfFailedAccesses
Code	INVALID_PASS_ATTEMPTS
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔

Encrypted	—
Possible Values	-1 (Default), indicates that there is no limitation on the number of failed logins.

Maximum user locked time

The account locking by number of failed logins is set for a maximum period of time, indicated in minutes, in this property. After that time, the user will be able to access again, if they enter correctly with their password.

If -1 is indicated then the account remains locked indefinitely until one of the planned unlocking mechanisms is executed:

- The security administrator activates the account again.
- The user receives an email with a link that allows them to activate their account again.

Name	MaximumUserLockoutTime
Code	MAX_USER_BLOQUED_TIME
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	—
Possible Values	-1 (Default), indicates that the account remains blocked.

Password history

When the user registers their password, **Deyel** verifies that it is different from the previous N.
The number N, defined in this property, can assume values between 0 and 100.
If 0 is indicated, then **Deyel** does not compare the new password with the previous ones. Repetitions are allowed.

Name	PasswordHistory
Code	PASSWORD_HISTORY
Configuration Levels	
• Installation	✔

• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	—
Possible Values	- 0 (Default), indicates that password repetition is not enforced. - N between 1 and 100.

This mechanism is applied when access passwords are managed by **Deyel**, using native authentication.

When the user enters a new key password, **Deyel** verifies the following:

- If N=0, the new password is not compared with the previous ones.
- If N=1, the new password cannot be the same as the current one.
- If N>1, the new password cannot be the same as the current one, and neither can it be the same as the previous N-1 passwords.

For example, if N=3, the new password must be different from the current one and the 2 previous ones recorded in history.

When the password is invalid, the user receives a message “You have used that password previously. You must report a different one.”

To implement this control, **Deyel** keeps a history of all passwords used by the user. Values are kept in chronological order and encrypted in the database and are not accessible by the user.

Two step authentication

When using native or custom authentication, it is possible to enable two-step authentication.
This property sets the desired behavior.

Name	TwoFactorAuthentication
Code	2FA
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔

Encrypted	—
Possible Values	• Optional (Default) • Required • Not Enabled

Password expiration time

Indicates the number of consecutive days that must pass for a password to expire.
If the value -1 is specified, then passwords do not expire.

Name	PasswordExpirationTime
Code	PASSWORD_EXPIRATION_TIME
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	—
Possible Values	-1 (Default), indicates that passwords do not expire. It may contain numeric values that represent a number of days.

This control is applied when using the native authentication, where passwords are managed by **Deyel**.

Each time the user sets a new password, the time at which this operation is performed is recorded and the expiration time can be calculated.

For example, if the value 1 is reported, passwords that were set on that day expire the day after.

Password expiration notification

Indicates the number of days in advance that users must be notified of the expiration of their password.

- If the value -1 is indicated, then no notification is issued.
- If a value other than -1 is specified, then it is checked to be less than the property Password expiration time.

The scheduled task must be active Notify Password Expiration.

Name	PasswordExpirationNotification
Code	PASSWORD_EXPIRATION_ALERT_TIME
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	—
Possible Values	-1 (Default), indicates that no notification is issued. It may contain numeric values that represent a number of days.

3.5.2.1.4.3. Cases Grid

Set of properties that allow defining the columns that are initially displayed in the cases grid.

All of them have the following general characteristics.

Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	✔
Dynamic	✔
Encrypted	—
Possible Value	• Yes (Predetermined) • No

Configurable Properties

Process name

Name	GridCaseProcess
Code	SHOW_PROCESS_CASES

Case description

Name	GridCaseDescription
Code	SHOW_DESCRIPTION_CASES

Chat

Visualizar acceso a la conversación asociada al caso.

Name	GridCaseChat
Code	SHOW_CHAT_CASES

Name of the current activity

Name	GridCaseActivity
Code	SHOW_ACTIVITY_CASES

Case start date

Name	GridCaseHome
Code	SHOW_BEGIN_CASES

Responsible for current activity

Name	GridCaseResponsible
Code	SHOW_RESPONSIBLE_CASES

Expiration of the current activity

Name	GridCaseExpiration
Code	SHOW_EXPIRED_CASES

Case priority

Name	GridCasePriority
Code	SHOW_PRIORITY_CASES

Case completion

Fecha de finalización del caso.

Name	GridCaseEnd
Code	SHOW_ENDED_CASES

Start of current activity

Name	GridCaseHomeActivity
Code	SHOW_BEGINACT_CASES

Case status

Name	GridCaseState
Code	SHOW_STATE_CASES

Case identification

Name	GridCaseCase
Code	SHOW_CASE_CASES

3.5.2.1.4.4. Tasks Grid

Set of properties that allow defining the columns that are initially displayed in the tasks grid.

All of them have the following general characteristics.

Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—

• User	✔
Dynamic	✔
Encrypted	—
Possible Value	• Yes (Predetermined) • No

Configurable Properties

Case identification

Name	GridTasksCase
Code	SHOW_CASE_TODOLIST

Process name

Name	GridTasksProcess
Code	SHOW_PROCESS_TODOLIST

Case description

Name	GridTasksDescriptionCase
Code	SHOW_DESCRIPTION_TODOLIST

Chat

Visualizar acceso a la conversación asociada al caso.

Name	GridTasksChat
Code	SHOW_CHAT_TODOLIST

Name of the homework

Name	GridTasksName
Code	SHOW_ACTIVITY_TODOLIST

Task start

Name	GridTasksHome
Code	SHOW_BEGIN_TODOLIST

Responsible for the task

Name	GridTasksResponsible
Code	SHOW_RESPONSIBLE_TODOLIST

Task due

Name	GridTasksExpiration
Code	SHOW_EXPIRED_TODOLIST

Task priority

Name	GridTasksPriority
Code	SHOW_PRIORITY_TODOLIST

3.5.2.1.4.5. Internationalization

Properties that allow configuring the language and numeric fields or date formats that are most appropriate for you.

Configurable Properties

Deyel language

Sets the language used by the Deyel platform tools. For example, user portal, modelers, etc.

Name	BTRB.Locale
Code	BTRB.Locale
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	✔
• User	✔

Dynamic	✔
Encrypted	—
Possible Values	• Spanish (Default) • English • Portuguese

Date display

Indicates the date display mask. For example, mmddyyyy or mm/dd/yyyy.

Name	DateDisplayMask
Code	DATE_VIEW_MASK
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	✔
• User	✔
Dynamic	✔
Encrypted	—
Possible Values	• DD/MM/YYYY (Default) • DDMMYYYY • DD-MM-YYYY • DD MM YYYY • MMDDYYYY • MM/DD/YYYY • MM-DD-YYYY • MM DD YYYY

Time display

Indicates the time display mask. For example, HHMM or HH:MM.

Name	Time Display Mask
Code	Hour_View_Mask
Configuration Levels	
• Installation	✔

• Application	—
• Organizational Unit	✔
• User	✔
Dynamic	✔
Encrypted	—
Possible Values	• HH:MM (Default) • HHMM

Decimal symbol - Editing numeric values

Indicates the symbol that separates the integer part from the decimal part for output format. For example: 10.99.
This property must be informed so that the numerical values can be displayed correctly.
The value of the field can be entered without typing the integer part separator, but if a separator is typed, it must be the one specified in this property.

Name	DecimalSymbolEditing
Code	DECIMAL_CHARACTER_SEPARATOR_IN
Configuration Levels	
• Installation	✔
• Application	✔
• Organizational Unit	—
• User	✔
Dynamic	✔
Encrypted	—
Possible Values	, (Predetermined) .

Thousands symbol - Editing numeric values

Indicates the symbol that separates groups of thousands (1,000, 1,000,000, etc.) for input format. For example: 999,999.
The field value can be entered without typing a thousands separator, but if a separator is typed, must be the one specified in this property.

Name	ThousandsSymbolEditing
Code	THOUSANDS_CHARACTER_SEPARATOR_IN

Configuration Levels	
• Installation	✔
• Application	✔
• Organizational Unit	—
• User	✔
Dynamic	✔
Encrypted	—
Possible Values	, . (Predetermined)

Decimal symbol - Display of numerical values

Indicates the symbol that separates the integer part from the decimal part for output format. For example: 10.99.
This property must be reported so that the numeric values can be displayed correctly.

Name	DecimalSymbolDisplay
Code	DECIMAL_CHARACTER_SEPARATOR_OUT
Configuration Levels	
• Installation	✔
• Application	✔
• Organizational Unit	—
• User	✔
Dynamic	✔
Encrypted	—
Possible Values	, (Predetermined) .

Thousands symbol - Display of numerical values

Indicates the symbol that separates groups of thousands (1,000, 1,000,000, etc.) for output format. For example: 999,999.
This property must be informed so that the numerical values can be displayed correctly.

Name	ThousandsSymbolDisplay
Code	THOUSANDS_CHARACTER_SEPARATOR_

	OUT
Configuration Levels	
• Installation	✔
• Application	✔
• Organizational Unit	—
• User	✔
Dynamic	✔
Encrypted	—
Possible Values	, . (Predetermined)

3.5.2.1.4.6. Interface

Set of properties that allow to configure the user interface.

Configurable Properties

Theme

Defines the theme used by the users portal.

Name	Theme
Code	THEME_DIR
Configuration Levels	
• Installation	✔
• Application	✔
• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	—
Possible Values	• Red • Green • Blue • White (Default) • White y Green • White y Brown • Blue y Light Blue

Menu initial state

Initial state of the menu when the user enters the portal.

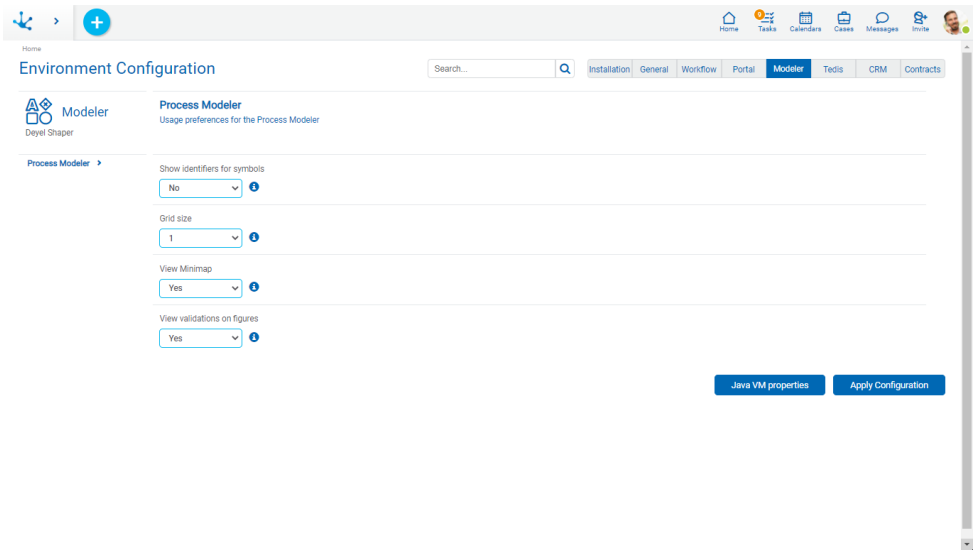
Name	estadoInicialMenu
Code	INITIAL_MENU_STATUS
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	✔
Dynamic	—
Encrypted	—
Possible Values	• Expanded • Collapsed (Default)

3.5.2.1.5. Modeler

It allows configuring properties related to the modelers of **Deyel**.

Options

- Processes Modeler
- Applications Modeler



3.5.2.1.5.1. Processes Modeler

Usage preferences for the process modeler.

Configurable Properties

Show identifiers for symbols

Activating this property, its unique identifier is displayed along with the name of each symbol.

Name	ShowIdentifiers
Code	SHOW_IDS
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	✔
Dynamic	✔
Encrypted	—
Possible Values	• Yes • No (Predetermined)

Grid size

Number of pixels used to move symbols on the diagram.

Name	GridSize
Code	GRID_SIZE
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	✔
Dynamic	✔
Encrypted	—
Possible Values	• 1 (Predetermined)

	• 10 • 20

View Minimap

View the minimap in the lower right of the modeling area.

Name	ViewMinimap
Code	SHOW_MINIMAP
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	✔
Dynamic	✔
Encrypted	—
Possible Values	• Yes (Predetermined) • No

View validations on figures

Displays the exclamation point on the figures that have validation errors.

Name	ValidationsOnFigures
Code	INLINE_VALIDATIONS
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	✔
Dynamic	✔
Encrypted	—
Possible Values	• Yes (Predetermined) • No

3.5.2.1.5.2. Applications Modeler

Usage preferences for the applications modeler.

Configurable Properties

Building Environment

Allows to update the path to the environment that executes the build service of a mobile application.

Name	BuildingEnvironment
Code	APPBUILDER_ENV
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	—
Possible Value	• https://appbuilder.deyel.com/ • https://appbuilderdev.deyel.com/

Add KeyStore

Allows to update the Android KeyStore file defined in base64 format.
It is a binary file that contains cryptographic keys, such as private keys and digital certificates, that are used to sign and authenticate Android applications before they are distributed.

Name	AddKeystore
Code	KEYSTORE
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	—

Possible Value	File in user-defined Base64 format.

KeyStore Password

Allows to update the password in order to sign the application within the KeyStore.
It is the password used to access and unlock the keys stored within the keystore file.

Name	KeystorePassword
Code	KEYSTORE_PASSWORD
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	—
Possible Value	User-defined text.

KeyStore Alias

Allows to update the alias associated with the KeyStore.
It is the alias assigned to a private key and its corresponding certificate within the keystore file.

Name	KeystoreAlias
Code	KEYSTORE_ALIAS
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	—
Possible Value	User-defined text.

Store Password

Allows to update the password that enables the use of the KeyStore file.
It is the password used to protect the keystore file.

Name	StorePassword
Code	STORE_PASSWORD
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	—
Possible Value	User-defined text.

Apple ID

Allows to update the Apple identifier associated with the developer account.

Name	AppleID
Code	APPLE_ID
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	—
Possible Value	File in user-defined Base64 format.

Apple Team Identifier

Allows to update the team identifier associated with the Apple developer account.

Name	AppleTeam

Code	APPLE_TEAM
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	—
Possible Value	User-defined text.

Password for Apple Services

Allows to update the password to use Apple services.

Name	AppleServicesPassword
Code	APPLE_SERVICES_PASSWORD
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	—
Possible Value	User-defined text.

Apple Certificate

Allows to update the Apple certificate file with p12 extension in base64 format.

Name	AppleCertificate
Code	CERTIFICATE
Configuration Levels	
• Installation	✔
• Application	—

• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	—
Possible Value	File in user-defined Base64 format.

Certificate Password

Allows to update the password associated with the Apple certificate.

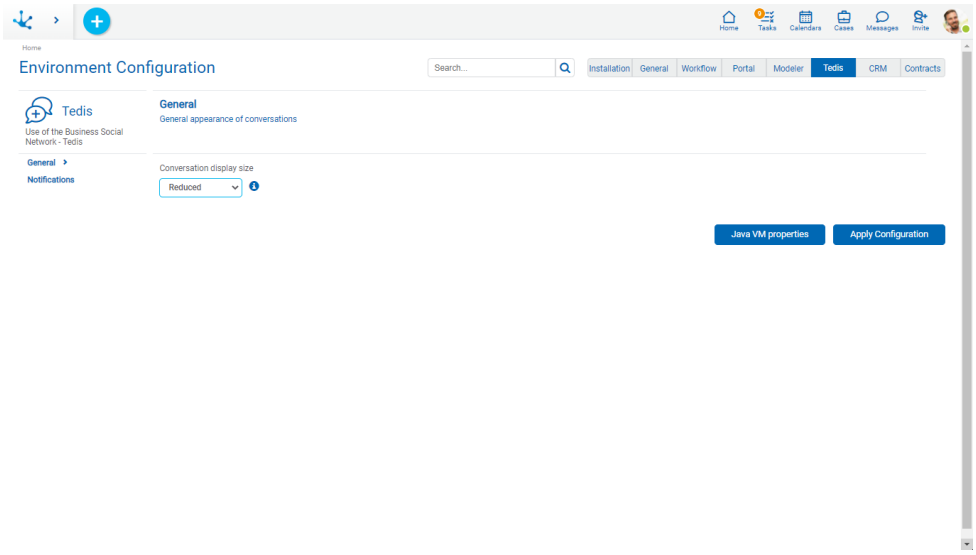
Name	CertificatePassword
Code	CERTIFICATE_PASSWORD
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	—
Possible Value	User-defined text.

3.5.2.1.6. Tedis

It allows configuring properties that define the appearance and behavior of Tedis Business Social Networking.

Options

- General
- Notifications



3.5.2.1.6.1. General

General appearance of conversations in Tedis.

Configurable Properties

Conversation display size

Initial size of conversations. Each of them can be initially displayed in a reduced or expanded window.

Name	SizeConversations
Code	INITIAL_CONV_SIZE
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	—
Possible Values	• Reduced (Default) • Expanded

Show users by organizational unit

Shows the users in its organizational unit and subunits.

Name	UsersByOrganizationalUnit
Code	TEDIS_USERS_BY_UO
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	—
Possible Values	• No (Default) • Yes

Show users by application

Shows the users who have permission to use the application.

Name	ShowUsersPerApplication
Code	TEDIS_USERS_BY_APP
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	—
Dynamic	✔
Encrypted	—
Possible Values	• No (Default) • Yes

3.5.2.1.6.2. Notifications

Sending automatic notifications from Tedis.

Configurable Properties

Notification of new messages

Receive a notification when a new message is received.

Name	NotificationNewMessages
Code	ALERT_ON_NEW_MESSAGE
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	✔
Dynamic	✔
Encrypted	—
Possible Values	• Yes (Predetermined) • No

Notification of connected users

Receive a notification when a user connects to the portal.

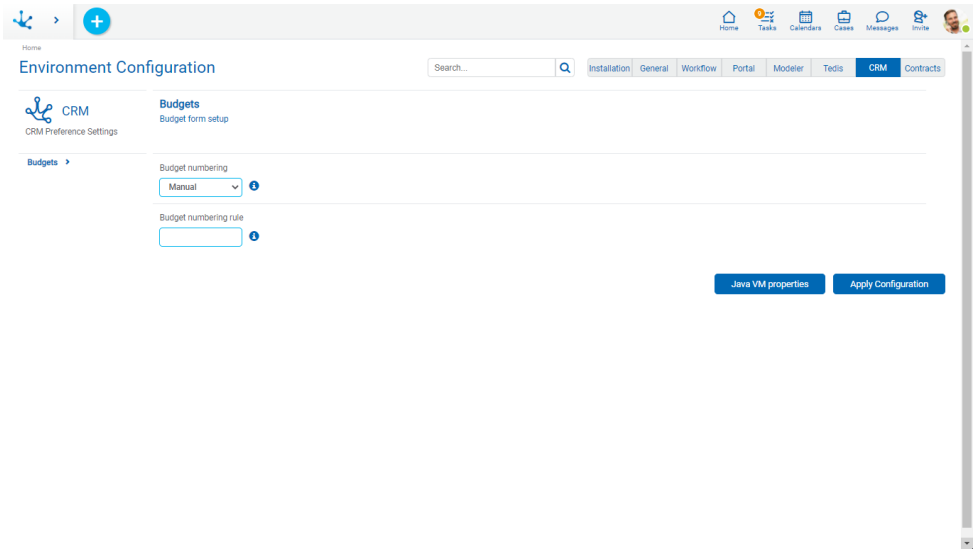
Name	NotificationUsersConnected
Code	ALERT_USER_ONLINE
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	✔
Dynamic	✔
Encrypted	—
Possible Values	• Yes (Predetermined) • No

3.5.2.1.7. CRM

It allows configuring properties that define specific behaviors and use preferences of the CRM solution.

Options

- Budgets



3.5.2.1.7.1. Budgets

Configuring the CRM Quote form.

Configurable Properties

Budget numbering

It allows to choose the way in which the budget numbering is generated.

Name	NumberingQuote
Code	QUOTE_NUMBER
Configuration Levels	
• Installation	✔
• Application	—
• Organizational Unit	—
• User	✔
Dynamic	—
Encrypted	—
Possible Values	• Manual (Default) • Automatic • Rule

Possible Values

- Manual

Must be informed by the user.

- Automatic

Numbering is auto-increment

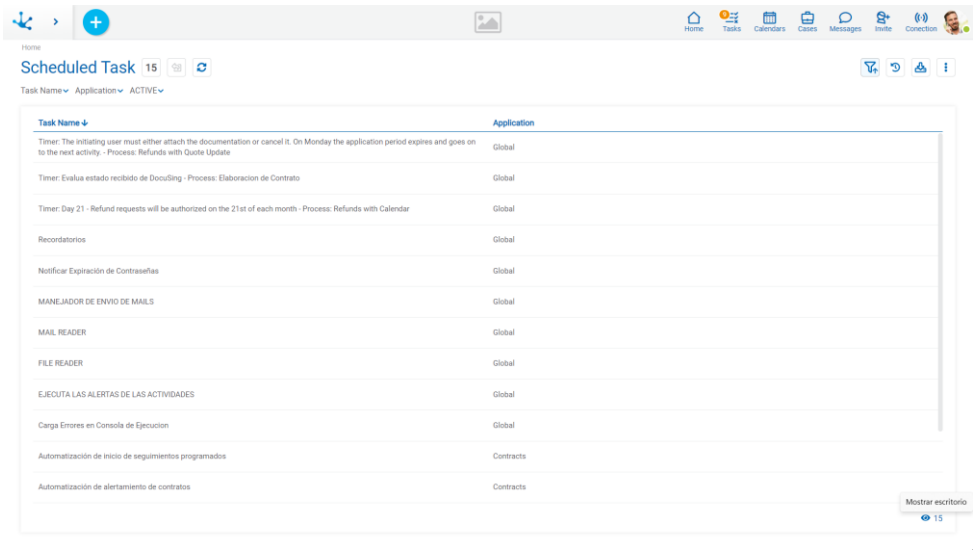
- Rule

Numbering is obtained from the rule entered in the Budget numbering rule property.

3.5.2.2. Scheduled Tasks

A scheduled task is an action configured to execute automatically at a specific time or time interval, without the need for manual intervention. It is executed automatically by the SYSUSER user.

Each task defines the frequency with which it should be executed.

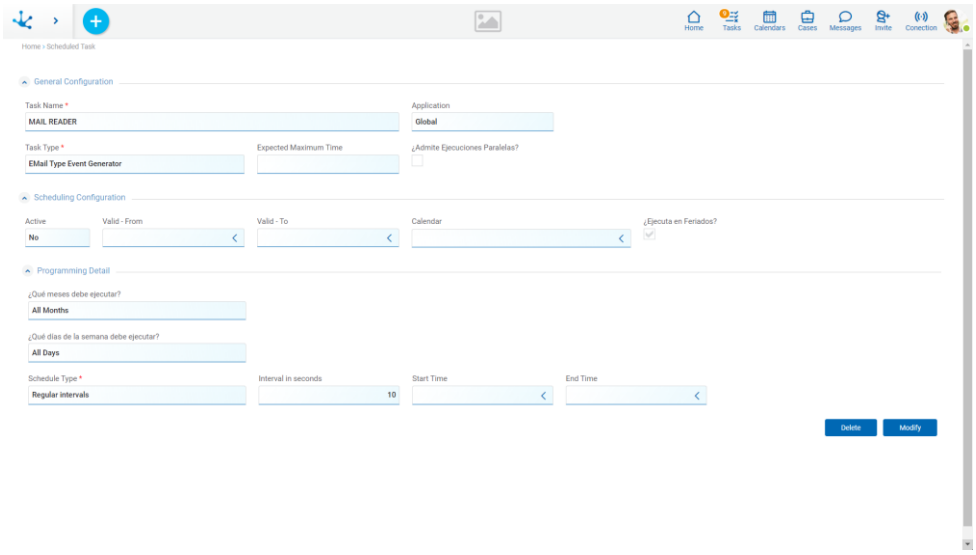


It is possible to select the following options:

- Administrate Tasks
- Task Monitor

3.5.2.2.1. Administrate Tasks

Deyel uses scheduled tasks to automatically execute tasks on the server.



An asterisk "" on the label indicates that the property is required.*

Properties

General configuration

Task Name

It must contain a text that describes the task objective.
This text appears as the task name when monitoring scheduled tasks or when showing the task execution history.

Application

Indicates to which application the scheduled task belongs.
Notice that the scheduled tasks of the (licensed applications) solutions can only be activated when the use license is valid in the environment.

Task Type

It allows to select the scheduled task type from a set of options defined in the environment

- Based on Business Rule Execution
- EMail type Event Generator
- File type Event Generator
- Rule type Event Generator
- Alert generator for activities

Maximum Expected Time

The maximum expected execution time is defined and expressed in milliseconds.
This data is optional and allows delays to be detected during task execution.

In the task monitor, when displaying a task in progress, the state can be:

Running When the execution time is less than or equal to the maximum expected time.

 Running - Delayed When the execution time exceeds the maximum expected time.

Does it support Parallel Executions?
When **Deyel** is used in High Availability (HA) mode, where there may be multiple installations of **Deyel** environment, this property determines whether the scheduled task can be executed simultaneously or in different installations of **Deyel** environment.

> *A scheduled task never begins its execution in a **Deyel** environment installation if the previous execution has not yet finished..*

Rule Configuration

This section is displayed when the task type is "Based on Business Rule Execution", this allows configuring the corresponding business rule.

Rule to Execute
A list of the business rules defined in the environment is presented.
The rule to use should be selected.

Parameters
The parameters that can be moved on to the task are reported. One parameter per row should be entered, indicating its name and the corresponding value.

Scheduling Configuration

It allows defining the scheduling to execute the task. When clicking on the "Accept" button, the changes made to the task schedule are immediately applied.

Active
Determines whether the scheduled task should be executed or not.

Valid From / Valid To
An optional effective period can be set for the scheduled task. This means that the task will only be executed within that period.

Calendar
Indicates the calendar that must be considered to interpret the properties Valid From y Valid To.

> *The values of these properties are of date and time type and therefore should be interpreted considering a specific time zone or usage. This time zone or usage is the one corresponding to the selected calendar. If no calendar is indicated, the standard calendar is considered.*

Does it execute on Holidays?
Indicates whether the task should be executed normally on holidays.
The definition of holidays is also determined by considering the reported calendar or the standard calendar when no calendar is reported.

Programming Details

This section allows defining when the task should be executed.

What months should it execute?

Possible values are:

- All months (Default)
- Indicated months
  The months to execute are selected. At least one month must be chosen.

What days should it execute?

Possible values are:

- All days (Default)
- Indicated days

The days of the week to execute are selected. At least one day must be chosen.

Schedule Type

Indicates the time the task should be executed.
Possible values are:

- Daily at a specific time
  #### Execution Time
  Time at which the scheduled task should be executed.
  The time zone set in the Calendar field is considered.

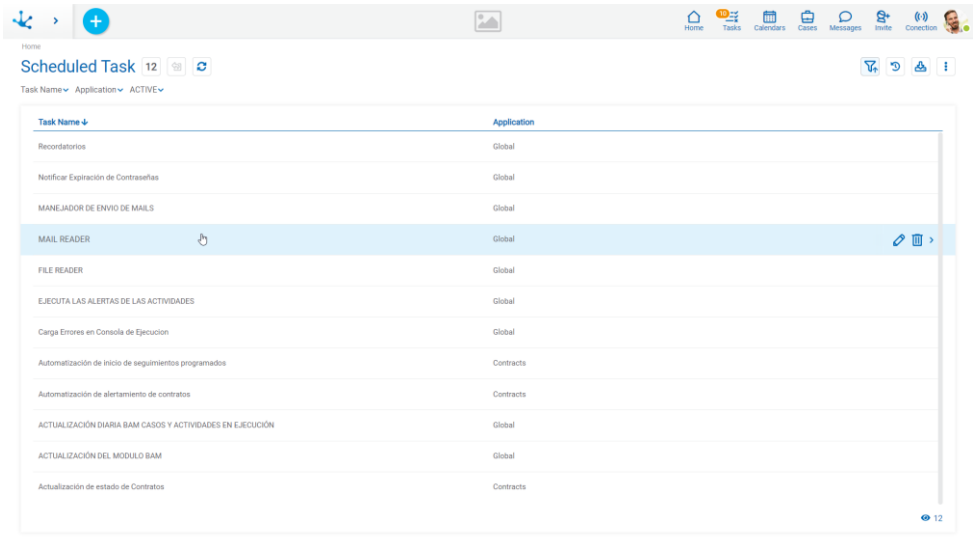
- Regular intervals.
  #### Interval in Seconds
  Number of seconds that elapse between task executions.

  #### Start Time / End Time
  They indicate the time of the day during which the task should be executed. If these properties are not specified, the task executes according to the selected frequency but without time restrictions. The time zone set in the Calendar field is considered.

3.5.2.2.1.1. Grid

By selecting the "Administrate Tasks" button In the main menu, they are displayed in grid form, with the standard presentation of the results grid, using the following facilities:

- Sorting
- Search and Filters Bar
- Data Download
- Operations



The following properties of scheduled tasks are displayed as grid columns:

- Task Name
- Application

Filters can be applied by the following properties:

- Task Name
- Application
- Active

Operations can be performed on each grid row. By clicking on the row, the selected task is shown, while hovering over it enables the icons and that allow the update or delete operations respectively.

3.5.2.2.1.2. Operations

The operations that can be performed on each task depend on the security permissions defined for the logged-in user. When hovering the cursor over each grid row, the icons corresponding to the available operations are displayed.

The exception is the operation "Create", which can be performed from the context menu in different ways.

- Hover over the icon and select the icon to your right that corresponds to the option "Scheduled Task".
- Click on the icon and select the option "Scheduled Task" in the expanded vertical panel.

Create

Opens the properties panel of the new scheduled task, where the properties are editable. To create, press any of the available buttons and the user receives a message indicating that data was saved.

Buttons

- Accept: Confirms the creation of a new task.
- Accept and Create: Confirms the creation of a task and opens a new panel to create another one.

Show

Opens the properties panel of the selected task, where the properties are not editable and depending on the security permissions of the user, the buttons available to operate with the task being shown are enabled.

Buttons

- Modify
- Delete

Modify

Opens the properties panel of the selected task, only those properties that can be modified are editable. To update, click on the "Accept" button and the user receives a message indicating that data was saved.

> *The scheduled tasks of **Deyel** solutions are only configurable when the use license for that solution is valid. In that case, it is not only allowed to configure whether the task is active or not, and to modify its scheduling.*

Delete

Opens the properties panel of the selected task, where the properties are not editable. To delete, click on the "Accept" button and the user receives a message indicating that data was deleted.

> *The scheduled tasks of "Solutions" cannot be deleted.*

3.5.2.2.1.3. Parameters

Some types of scheduled tasks support the use of parameters, this is because they execute advanced rules.

- Based on Business Rule Execution
- Rule-Type Event Generator

Each parameter must be reported on a separate row, indicating its name and value, separated by the "=" sign.

The name corresponds to the name of the parameter specified in the advanced rule, respecting upper-case and lowercase letters.

The value must respect a format, depending on the data type of the rule parameter.

Valid Formats

Valid formats for parameter values in scheduled tasks are detailed below.

Alphanumeric

The string should be provided without the need to enter quotation marks.
Example:
pSurname=Lopez

Numeric

Integer or decimal values should be provided.

Examples:
- pAge=22
- pID=20201120
- pWeight=70.5

Logical

The values “true” or “false” should be provided.

Example:
- pcdActive=true

Other

If it is necessary to report a type other than the detailed formats, such as a date, the alphanumeric or numeric format can be used, but the rule must include the appropriate conversion.

3.5.2.2.1.4. Types of Scheduled Tasks

There are different types of scheduled tasks defined in the environment.

Based on Business Rule Execution

This task type allows scheduling the automatic execution of a business rule.

It is necessary to authorize the system user, SYSUSER, for the execution of this rule. For this, authorization must be included in the SYSUSER login permission.

When configuring the scheduled task, specify the business rule to execute.

Deyel offers some predefined scheduled tasks of this type, which allow for different processes to be performed.

eMail-type Event Generator

Deyel can connect to an email server to read messages received in a specific account.

To configure the integration with this email server, the Mail Reader adapter is used.

When activating a task of this type, it verifies the reception of emails in the account configured in this adapter.

When this task is executed, **Deyel** checks the receipt of emails in the configured account. For each email received, an email-type event is triggered and announced to all processes that are waiting for the occurrence of such events..

Deyel offers such a task within the scheduled tasks that are predefined.

It is possible to define new scheduled tasks of this type to allow for different schedules. For example, a first task scheduled to run from Monday to Friday from 9 a.m. to 6 p.m. and a second task scheduled to run on Saturdays from 9 a.m. to 1 p.m.

Rule-Type Event Generator

In the process modeler, start events or border events that are of "Rule" type can be modeled. The occurrence of these events is directly linked to a scheduled task of this type. When defining the event, the scheduled task whose execution generates the events is selected.

This type of task executes a business rule and with the results of that execution, a rule-type event is triggered, which is announced to all processes waiting for the occurrence of such events.

The business rule to be executed is specified when creating and configuring the task.

File-Type Event Generator

In the process modeler, start events or border events that are of "File" type can be modeled. This type of event occurs when a new file appears that meets the conditions modeled in the event.

When a task of this type is activated, **Deyel** checks for the appearance of new files in a given folder.

For each new file detected, an event of "File" type is triggered, which is announced to all processes waiting for the occurrence of such events.

Deyel offers such a task within the scheduled tasks that are predefined.

New scheduled tasks of this type can be defined to allow for different schedules. For example, a first task scheduled to run from Monday to Friday from 9 a.m. to 6 p.m. and a second task scheduled to run on Saturdays from 9 a.m. to 1 p.m.

Generates Alerts for Activities

When a task of this type is activated, the activities that are being executed are verified and the need to issue alerts is analyzed. Considering the definition of alerts modeled in each activity, the corresponding notifications are sent by email.

There is a task of this type within the scheduled tasks that are predefined.

Timer-Type Event Generator

Scheduled tasks of this type are automatically created by **Deyel** when a process with an active timer is published. Likewise, they are automatically deleted when the timer is removed or deactivated in the process.

Each of these tasks is associated with a timer-type event in a specific process.

The functionality of this scheduled task is to control the elapsed time and, according to the frequency indicated in that event, communicate its occurrence to the process.

3.5.2.2.1.5. Built-in Scheduled Tasks

A set of scheduled tasks specific to the **Deyel** environment is described below.

Environment Scheduled Tasks

These tasks are executed as required by **Deyel** to ensure its proper functioning. They start automatically along with **Deyel** and cannot be paused by the environment administrator.

The task monitor displays their execution only when there is an error, so it can be noticed by the administrator.

Debugger for Files Uploaded to Repositories

This task is responsible for permanently deleting content that is no longer in use. For example, files or images that were contained within a form field and were replaced by another. Or files of different types that were attached to a case and are no longer so. When this task is executed, it detects such situations and deletes the contents that can be considered obsolete or unused.

Old Environment Log Debugger

This task deletes old log files. The age of the data to be cleaned can be specified in the environment configuration.

LogFilePersistence: Sets the age in days of the application log files that should be deleted by this task.

In cloud environments, this property must assume values greater than zero and less than 30. In On-Premise environments, an age greater than 30 days can be indicated. If the value is zero, files are never deleted.

Log Console Debugger

This task deletes items from the log console. The age of the data to be cleaned can be specified in the environment configuration.

LogConsolePersistence: Sets the age in hours of the log console items that should be deleted by this task. If the value is zero, items are never deleted.

Environment Temporary File Debugger

This task deletes the temporary directories of **Deyel** (Temp, BusinessRuleDeployment, BusinessRuleXML, EventFiles, Export, Import, IntegrationDeployment, IntegrationXML, and Upload) It executes every 24 hours, starting from the environment startup time or the scheduled tasks restart time.

Report Manager

This task ensures the execution of reports in the background and the delivery of results to the user.

User Token Debugger

This task deletes user token records for terminated sessions. It executes every 12 hours, starting from the environment startup time or the scheduled tasks restart time.

Debugger of Scheduled Task Execution History

This task deletes the execution history records of scheduled tasks.
It executes daily and considers the property ScheduledTaskHistoryPersistence, which establishes the number of days that should pass before the information is deleted.

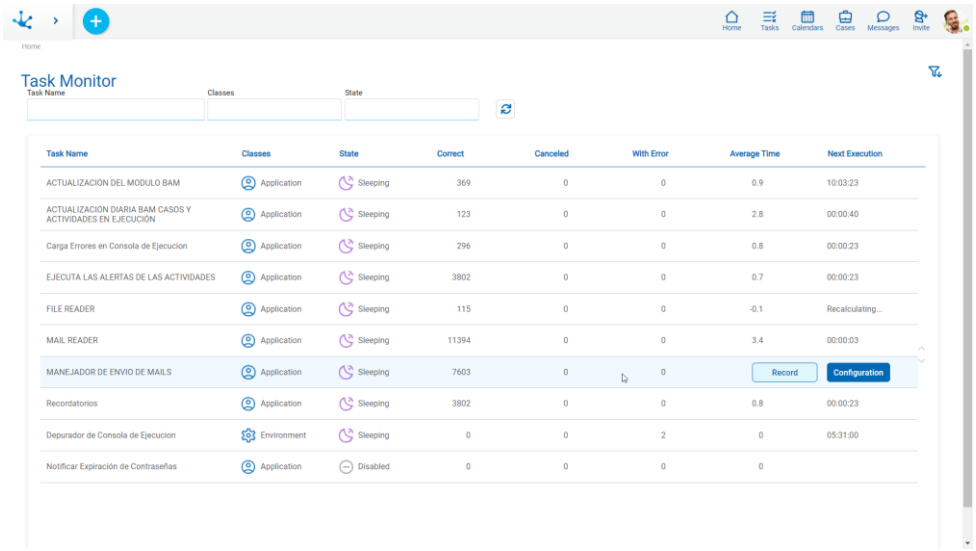
Optional Scheduled Tasks

These tasks allow the environment administrator to activate, deactivate, or configure the execution frequency, according to their needs..

Mail Sending Manager

This task ensures the sending of emails from the various facilities of **Deyel** that can generate them, such as automatic email-sending actions, advanced rules, and notifications, among others. If there are connection errors with the email server, a retry scheme is implemented to overcome failures and ensure that emails are sent.

For **Deyel** to effectively send emails, this scheduled task must be running, and email sending must also be enabled in the environment configuration.



Mail Reader

This task generates email type events. It is initially inactive in the environment of **Deyel**. It should be activated when modeling processes that use this type of event. **Deyel** checks the receipt of emails in the account indicated in the configuration of email reception.

File Reader

This task generates File type events. It is initially inactive in the environment of Deyel. It should be activated when modeling processes that use this type of event.

Execution of Activity Alerts

This task generates alerts for activities. It is initially inactive in the environment of **Deyel**. It must be activated if the execution of the alerts modeled in the activities and the sending of the relevant notifications via email is required.

BAM Module Update

This task is based on business rules. It is initially inactive in the environment of **Deyel** and executes every 10 minutes. Its goal is to take all updates on cases and activities recorded since the last execution of this task and automatically update the information in the module of BAM (Business Activity Monitoring). The execution of this task causes both the BAM information (cases and activities in execution) and the Process Analysis information (completed cases and activities) to be updated.

Information on new or modified activities and cases is detected by this task so that it incrementally impacts the BAM module. Therefore, every 10 minutes, any change or new development is detected and transmitted to the BAM module, to obtain recent and updated information when BAM reports are displayed.

Daily Update of BAM Cases and Activities in Progress

This task is based on business rules. It is initially inactive in the environment of **Deyel** and runs only once a day. However, this task can be scheduled to run more frequently, such as every 4 hours, as required. Its objective is to keep the duration (execution time) and the status (delayed, at risk, on time) of all ongoing cases and activities updated, according to the frequency with which the task was scheduled: daily or every 4 hours. So, this task only includes the update of cases and activities in progress, which are not covered by the previous task that runs every 10 minutes.

For example, an ongoing case that was uploaded yesterday and was taken by the incremental update every 10 minutes will remain without updated duration and status until it is modified again and detected by the incremental load. To prevent information from becoming outdated, this task is responsible for sweeping all ongoing information and updating its duration and status. Thus, this task is complemented by the previous one to display BAM reports and obtain complete and updated information.

Notify Password Expiration

This task is based on business rules. Notifies users in advance of their password expiration. The administrator must trigger the execution of this scheduled task, as it is initially inactive in the environment.

Once activated, it runs daily and analyzes the configuration of the Password expiration time and Password expiration notification environment properties.

If either property is set to -1, no notifications are issued and the task ends. Otherwise, for each active user, the task calculates the alert date, which is the password expiration date minus the number of days in advance that the notification should be sent. If the calculated alert date corresponds to the current day, then the user is notified by email about the upcoming expiration of their password.

It is important to consider the configuration of the emailing so that these notifications can be issued.

3.5.2.2.2. Task Monitor

This functionality is intended to display the execution state of active scheduled tasks in the system.

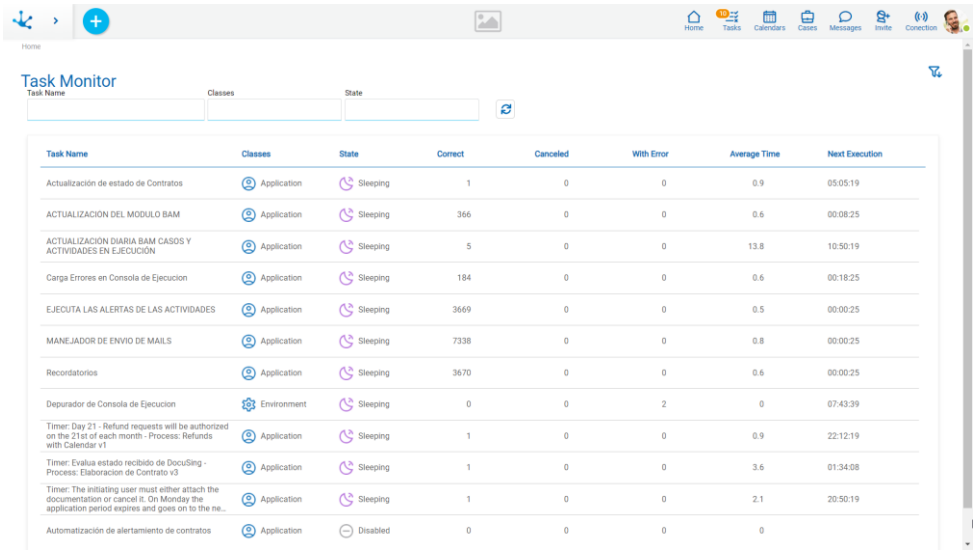
The complete list of defined scheduled tasks is displayed, considering that:

- Scheduled tasks for **Deyel** solutions are only displayed if the use license for that solution is active in the environment.

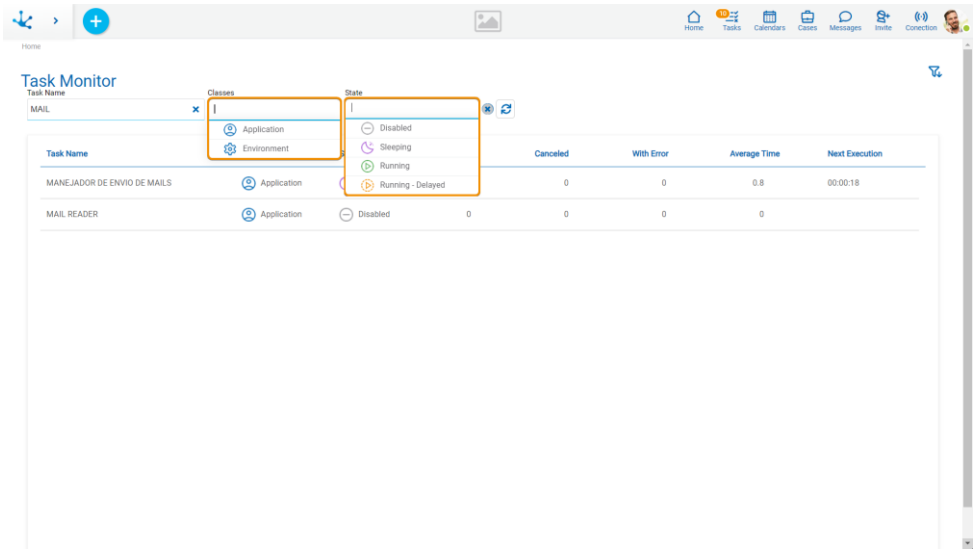
There are different alternatives to access this function:

- Click on the “Configuration” option in the main menu and within the "Scheduled Tasks" submenu, click on the "Task Monitor" option.

- Enter “Task Monitor” in the menu search and click on the listed option.



At the top, you can select values to define filters that condition the content of the list. These filters can be shown or hidden using the icons and respectively.



The resulting scheduled tasks list displays those tasks that meet the established criteria, considering the order specified below:

1. Environment Scheduled Tasks (when execution errors have occurred).
2. Application Scheduled Tasks, which are Running (Running or Running - Delayed).
3. Application Scheduled Tasks that are waiting to execute (Sleeping).
4. Application Scheduled Tasks that are disabled {Disabled).

And within each group, the scheduled tasks are sorted alphabetically.

Each item in the list displays information about a scheduled task.

Task Name
It is taken from the definition of the scheduled task.

Class
Indicates whether this is an environmental or application scheduled task.

State
The state of the scheduled task is displayed at the time of show.

Successful
Indicates the number of successful executions in the last 48 hours.
The information is retrieved from the execution history.

Canceled
Indicates the number of executions canceled in the last 48 hours.
The information is retrieved from the execution history.

When the administrator of **Deyel** uses the option to cancel the execution of a scheduled task, it is recorded in the execution history.

With Error
Indicates the number of executions failed in the last 48 hours.
The information is retrieved from the execution history.

When **Deyel** detects an error during the execution of a scheduled task, it records the erroneous completion in the task execution history.

Average Time
All successful executions of the task that have occurred in the last 48 hours are retrieved from the history and the average execution time, expressed in seconds, is calculated.

Next Execution
This column has value for active tasks. Indicates the time remaining until the task executes again. It is expressed in hours, minutes and seconds.
It is calculated by considering the time at which the previous execution started and recalculating the time for the next execution.

Operations

When hovering the cursor over each row, buttons are displayed to operate on the corresponding scheduled task.



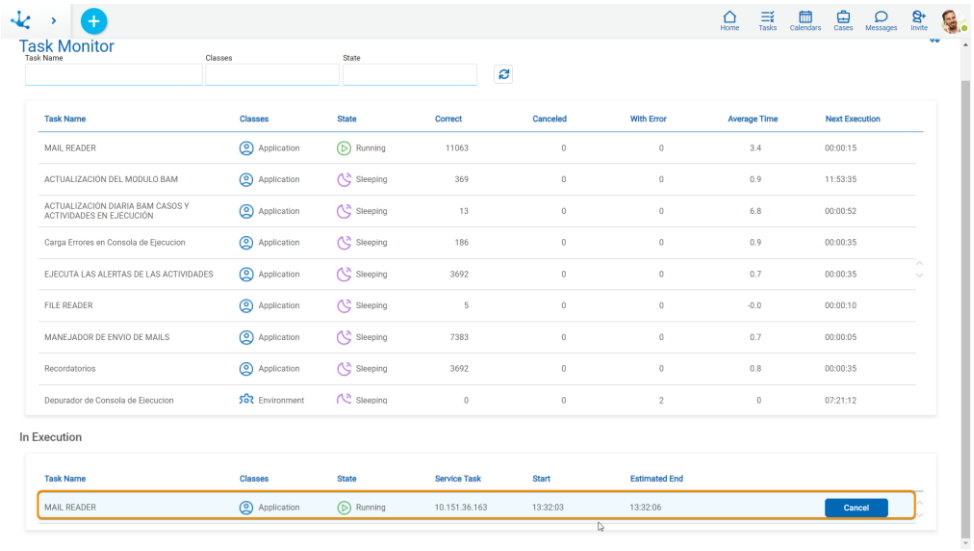
It allows access to the execution history of that scheduled task.



It allows access to the definition of this task to modify its properties. Only available for application tasks.

Tasks in Progress

If there are scheduled tasks that are being executed, they are displayed at the end of the monitor list.



Task Name, Class and State

These columns are described in the previous panel.

Service Task

It is **Deyel** service task, represented by its IP address, where the scheduled task is executing.

In High Availability (HA) execution environments, where there are multiple service tasks, it is possible for the same scheduled task to be executing in more than one service task. This happens because the scheduled task allows for parallel executions.
In these cases, multiple rows are displayed for the scheduled task. Each row reports on the execution of a service task (IP).

Start Time

Reports the time at which the scheduled task execution started.

Estimated End

It reports the estimated time of completion for the scheduled task.
It is calculated by adding the "Average Time" of execution in the last 48 hours to the start date of the task.

Operation

When hovering the cursor over each row, a button is displayed to operate on the corresponding scheduled task.

Cancel It allows the administrator to cancel the execution of the scheduled task.

When the "Cancel" button is clicked, **Deyel** checks whether the scheduled task is still executing on the corresponding IP and, if so, it immediately interrupts its execution. In addition, it records the cancellation of that execution in the history.

3.5.2.2.2.1. Execution History

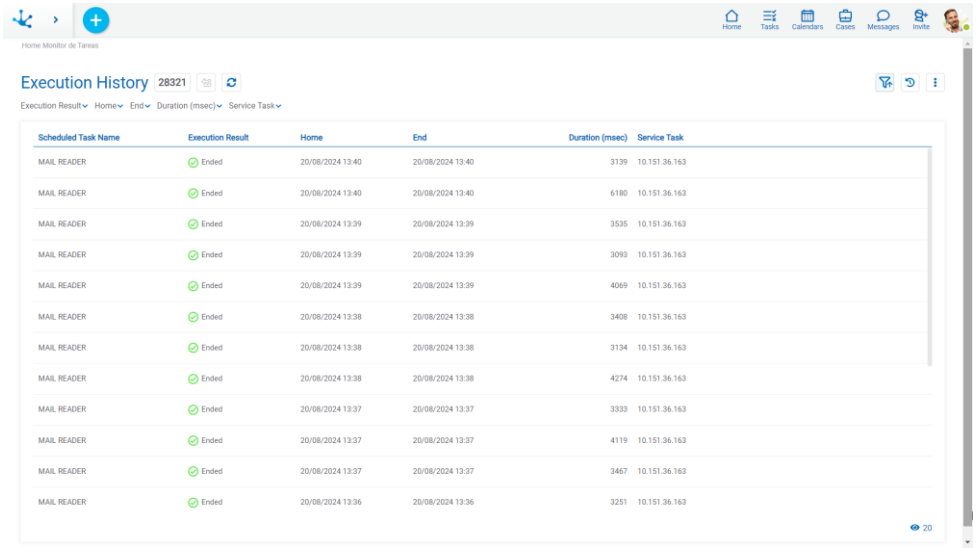
This functionality allows showing the executions of a scheduled task.

Deyel adds information into this log every time it runs a scheduled task, recording all the data described below.

Deyel also automatically cleans up this history by deleting data that is considered too old.

Record This button allows access to the history from the task monitor to review the executions of a scheduled task.

A grid is displayed where each element represents an execution of the task. They are arranged chronologically, showing the most recent first.



At the top, you can select values to define filters that condition the content of the list. These filters can be displayed or hidden using the icons and respectively.

Scheduled Task Name

This is the name of the scheduled task, as seen in the task monitor.

Execution Result

In the execution history, it is usually observed that most tasks have completed successfully. But it can also be seen the ones that ended with error or were cancelled. Occasionally, those that are still in progress are also found.

Home

It indicates the date and time of the start of the execution.

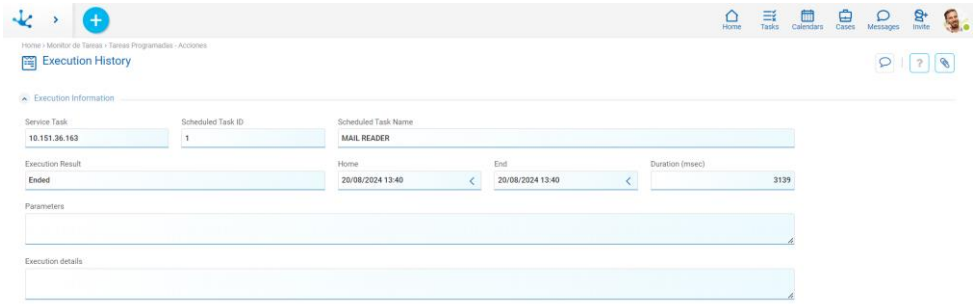
End
It indicates the date and time of the end of the execution.
If the task is running, this column has no value.

Duration (Msec)
Number of milliseconds that elapse from the start to the end of the execution.
If the task is running, this column has no value.

Service Task
Indicates the IP of the service task where the execution was performed.

Detailed Information

Selecting an item from the grid provides access to detailed information about the task execution.



Scheduled Task ID
This property is the identifier of the scheduled task.
The task name can be changed, but this ID remains constant over time and allows all previous executions to be recovered.

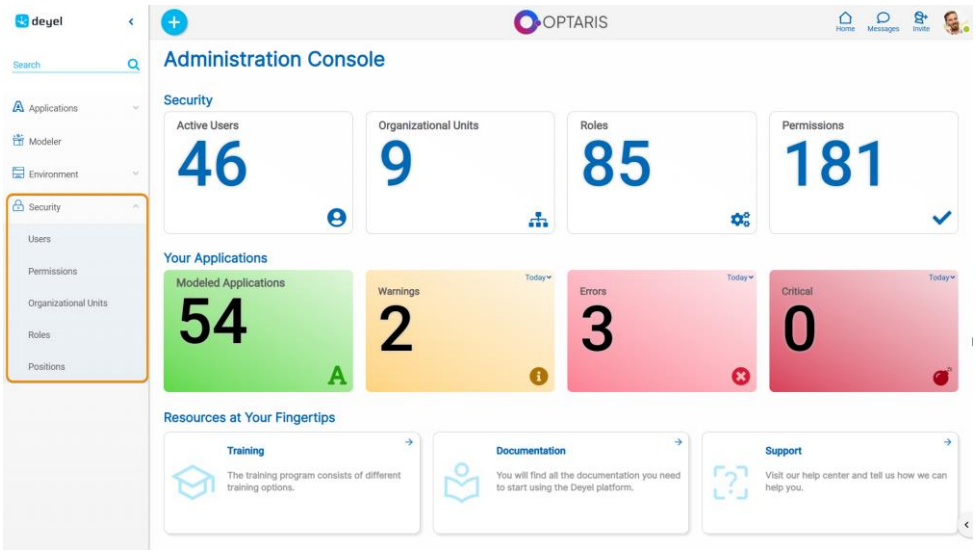
Parameters
These are the parameters that were reported to execute the scheduled task. They are obtained from the definition of the task.

Execution Details
When the task ends with an error, this field details the problem detected.

3.5.3. Security

It centralizes the administration of users, roles, permissions, and organizational structures within the environment, allowing the definition of access to applications and functionalities, thus ensuring a secure and organized control scheme.

- Users
- Permissions
- Organizational Units
- Roles
- Positions



3.5.3.1. Users

A user is a person or entity that uses a system, service, or application. In Deyel, a user is someone who interacts with the platform or any of its associated applications.

Nominated User

In **Deyel**, all users are nominated. This means that in order to identify themselves and use the product, they must be individually registered within a user catalog where their credentials, access permissions, personal and professional information, preferences, and other additional data are collected.

Each of the nominated users has accesss permissions assigned that define which are the specific functionalities the user can execute.

Person

Every person using **Deyel** or any of its applications based on this platform are called nominated users. As such, they have a code that uniquely identifies them, an access key which allows them to authenticate and initiate the work session and a user profile, that collects their personal information, working information, their preferences, additional data, etc.

Smart Thing

There is a type of user that does not represent a person, called "Smart Thing". It is about devices that have the capacity of communicating with other devices, reacting to events and executing specific functions. Smart things can participate of the business processes, as initiators of such or as responsible users of executing activities, so as to optimize the execution of such processes. For example, the execution of daily tasks or calculus tasks can be delegated to them.

Examples

- Chatbot: A chatbot can communicate with other users using the business social network Tedis, interpreting specific messages. Each chatbot can define a process to model the way in which it responds to each of the messages received. The answer can be simple, for example responding with a predefined sentence, or it can be more complex and shoot the execution of a business process. A chatbot is defined to interact with users through messages, delegating this way the execution of tasks. Human users can use chatbots as their assistants, so that they reply to their messages and automate actions.

- Rest API Client: A Rest API client is used so that the external applications can be integrated with **Deyel**, logging in with a user of this type and consuming the resources exposed in Deyel REST API, by sending credentials for authentication and generating the corresponding access token.

Anonymous User

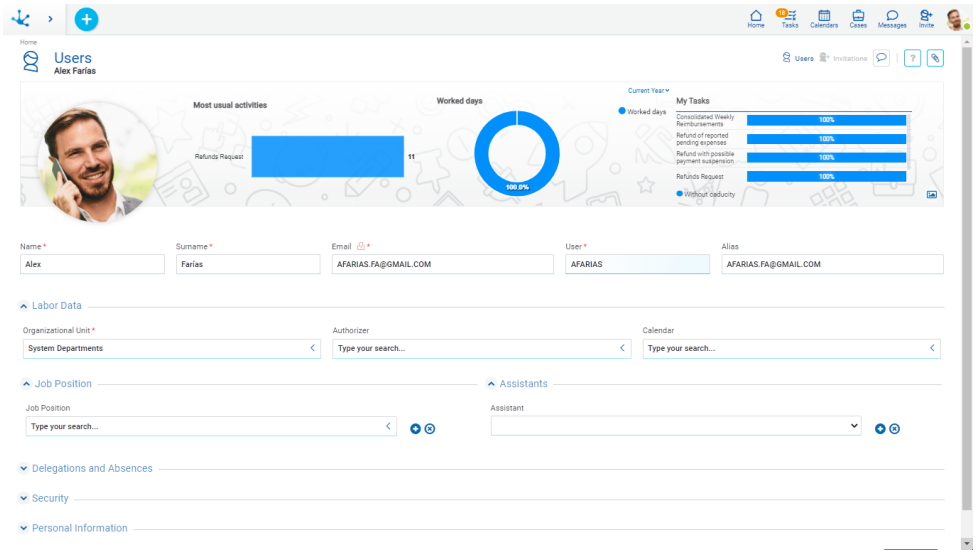
An anonymous user is someone who can use an application without needing to authenticate or log in. This type of user can execute any functionality from an application, as long as such functionality is contained within the "Anonymous" access permission.

System User

To make internal operations automatically, **Deyel** uses a predefined user called SYSTEM USER (SYSUSER).
This user is displayed in the user grid, it is not possible to modify or deactivate it and it does not require user licenses.
The initial load of the use license or its subsequent update or the execution of scheduled tasks, among others, are operations whose execution is made using this user and so it remains registered in the audit trails.
Logging in by authenticating with this user is not possible. **Deyel** uses it only for the doing and registration of internal tasks.

Properties

For each user, a profile is maintained with information through the configuration of properties that are grouped into different sections.



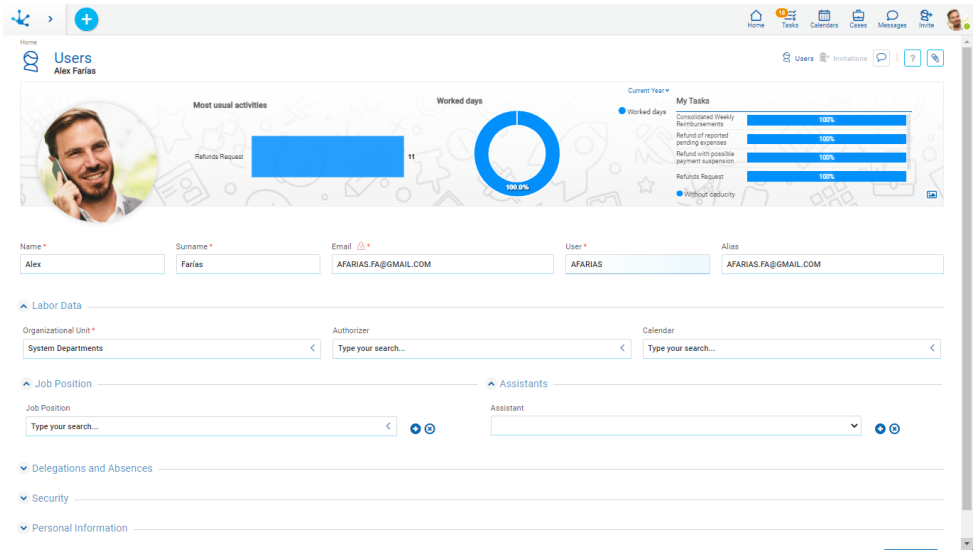
Invitations

In the upper right section the user relationship with the invitations that they have done is displayed and it is allowed to show those invitations.

3.5.3.1.1. Properties

For each user, a profile defined by properties is maintained, which are grouped into different sections.

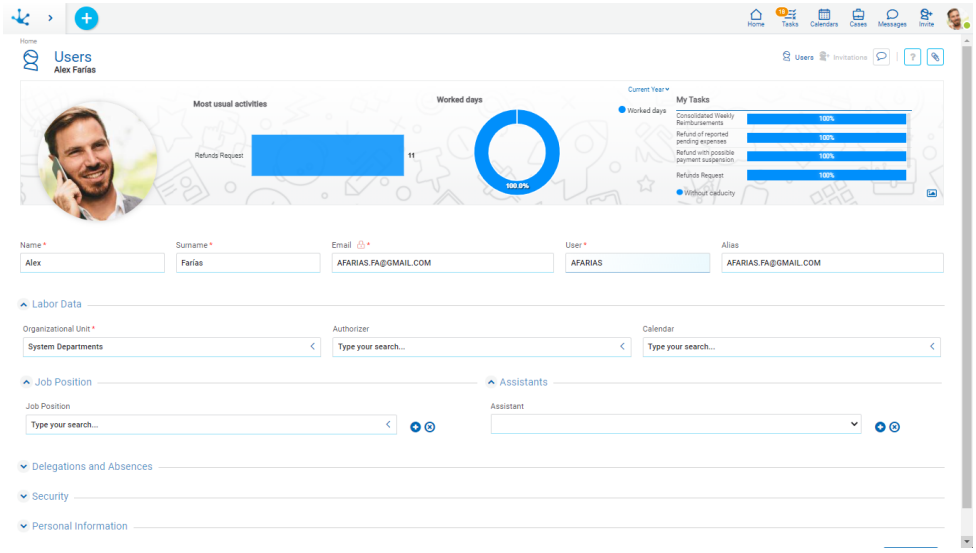
- User Identification
- Labor Data
- Delegations and Absences
- Security
- Personal Information
- Configuration
  For users of type "Smart Thing", this group of properties must be defined.



*An asterisk*on the label indicates that the property is required.*

3.5.3.1.1.1. Identification

The identification properties section contains user-specific information related to their identity within the environment of **Deyel**.



An asterisk "" on the label indicates that the property is required.*

Properties

Profile Picture

Each user can upload an image from a file to their profile. If not provided, an image containing the user's initials is automatically generated.

User

Code that uniquely identifies the user.

Alias

The value entered in this property acts as a user alternative code. For example, the email address or any other encoding that uniquely identifies the user can be used. **Deyel** checks that no two users have the same alias.
When accessing the portal, the user can enter either their code or their alias to authenticate.

Behavior Widgets

These widgets are often visualized through dashboards that display information such as habitual tasks, days worked, and recent user activities.

- The "Most usual activities" widget displays a ranked list of the most performed activities.
- The "Days worked" widget provides insights into the user's attendance. By clicking on this widget, the profile form expands to display absences and leaves, including their types and corresponding periods.
- "My Tasks" displays a summary of assigned tasks, along with their due date state. By clicking on a task, the task grid with its associated cases can be viewed.

Cover Image

It can be customized by selecting an image to use as a background from the icon  .

First Name - Last Name

These properties may not be reported for smart things.

Email

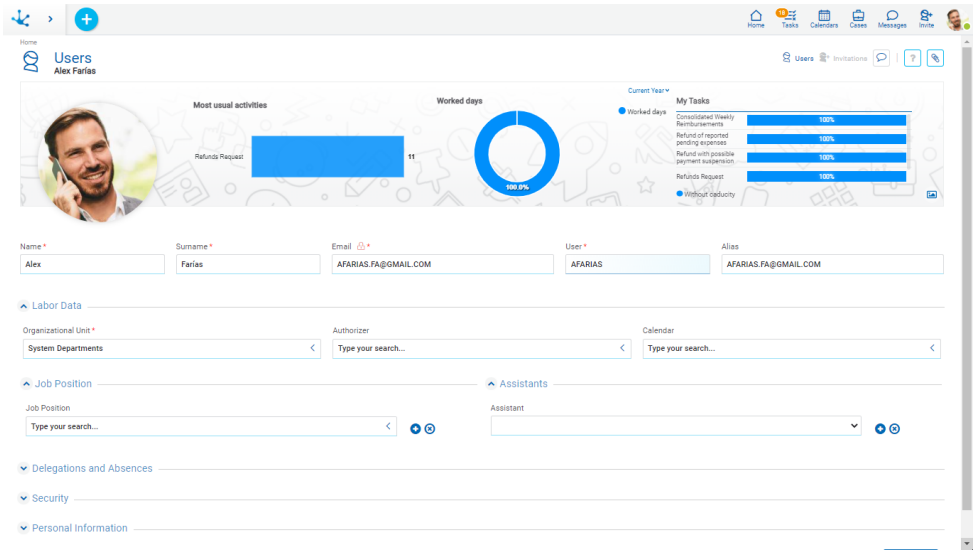
The value entered in this field serves as an alternative option for the user to enter the environment. **Deyel** validates that no two users have the same email.

When a user accesses their profile, it displays the icon  to the right of the property Email. Pressing it opens an additional field where the Application Password can be entered. This password is a unique key password generated from the Google email account settings or from the Outlook email account. It is designed to allow external applications, such as **Deyel**, to securely connect to the email account, without having to use the main password. This setting is required for the user to be able to send emails from entities or forms.

> *To ensure that emailing functions correctly, the user and mail server must match the configuration specified in the environment properties. .*

3.5.3.1.1.2. Labor Data

The work related properties section contains information regarding the user's position and activities within the company.



*An asterisk*on the label indicates that the property is required.*

Properties

Organizational Unit
Indicates the organizational unit in which the user works.

Authorizer
Indicates the one in charge of authorizing or approving the processes started by the user.
It is an optional property, that makes reference to another user belonging to the same organizational unit or to a different one.
This property can be recovered from the business processes so as to set the participant responsible to those activities that require an authorization, by using an agent Authorizer.

Calendar
Identifies the calendar that establishes the dates and working hours of the user.
This property can be defined for each user individually.
When the user has not a special calendar defined, then the current calendar at the level of the organizational unit to which they belong is considered. If there is no definition of a calendar in such unit, it is sought at the higher levels of the organizational structure, until detecting the calendar to apply.
The root organizational unit of all the hierarchy has a predetermined calendar defined.

Job Position

Optionally the job positions which correspond to the users can be indicated.

Position
Defines the job position occupied by the user.

Assistants

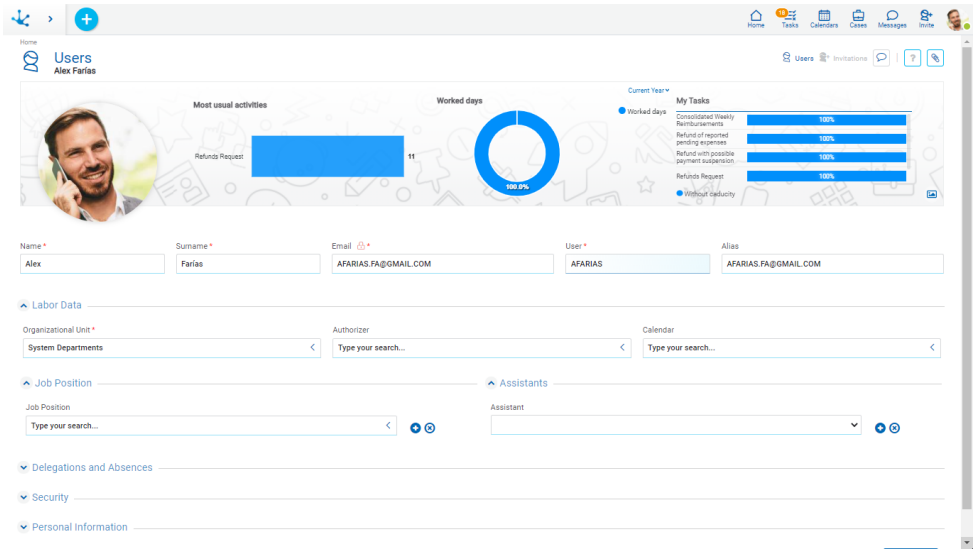
Optionally the chatbots can be selected to work as assistants of the user.

Assistant
A user can define one or more chatbots as their assistants.

Each time the user receives a message, each of their assistants verify if they recognize that message as a command they can execute.
Each chatbot exposes the commands they can execute and defines which participants can invoke their execution.
On the other hand, those who communicate with this user can see the commands they can use by pressing the command icon on the chat window.

3.5.3.1.1.3. Delegations and Absences

The delegations and absences properties section contains information about whom the user delegates their tasks to in their absence and who delegates tasks to the user, as well as information about the type and date of the absence.



> *An asterisk*on the label indicates that the property is required.*

Properties

Delegates

The list of delegated users is indicated and the period during which the delegations of tasks in effect.
When the user is inactive or absent then those current delegates can:

- Do the tasks assigned to the user.
- Initiate the processes authorized for the user.

Delegate of

Informs the list of users that have delegated tasks in the user and the corresponding period.
This list is dynamic and is completed at the moment of opening the user profile.

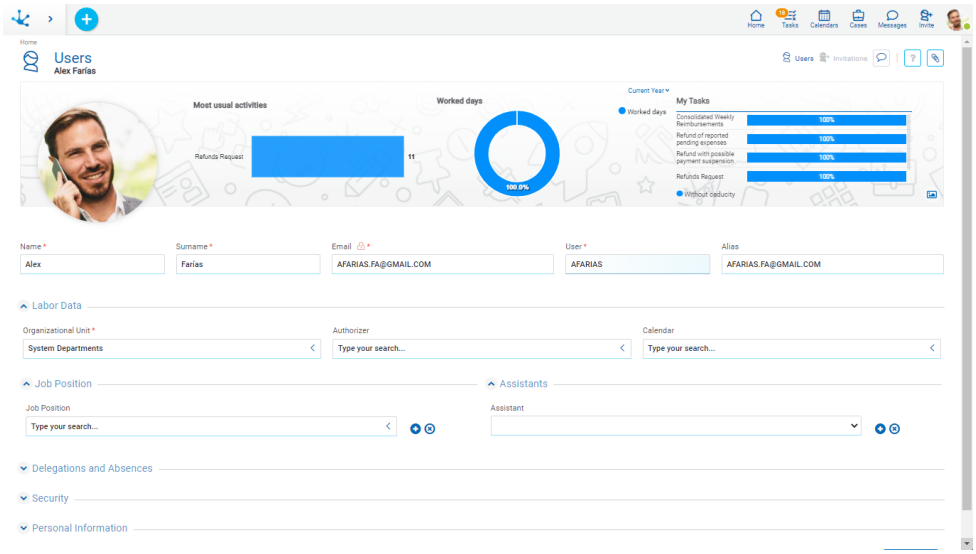
Absences

Each element of the list indicates the type of absence and the corresponding period.
During these periods of absence the mechanisms of tasks districts are activated.

Absences can be indicated considering a start and end time.

3.5.3.1.1.4. Security

The security properties section contains information related to the user's status, the licenses of the products they have installed in the environment, their access permissions, the roles they are part of, and two-factor authentication, if they have chosen to implement it.



*An asterisk*on the label indicates that the property is required.*

Properties

State and Expiration

User State

Indicates if the user is active or not.
An inactive user cannot enter the environment as their account is deactivated.

Expiration Date

The user account is automatically deactivated when the date entered in this property is reached. This automatic deactivation does not use the tasks delegations scheme.
This property can be used when the authentication of users is performed by **Deyel**.

Failed Accesses

Displays the number of failed accesses due to incorrect password.
It is incremented every time user authentication fails due to an incorrect password and returns to zero when the user authenticates successfully.
When the user exceeds the Maximum number of failed accesses their account is locked.

Lockout Date

Displays the date and time when the user's account is locked.
It is used to control the Maximum lockout time of the user's account.

Licenses

Each environment of **Deyel** has use licenses that enable the use of licensed applications during a period of time and by a certain number of users.
In this section the list of licensed application the user can use is indicated and which type of user license they have assigned to do this.

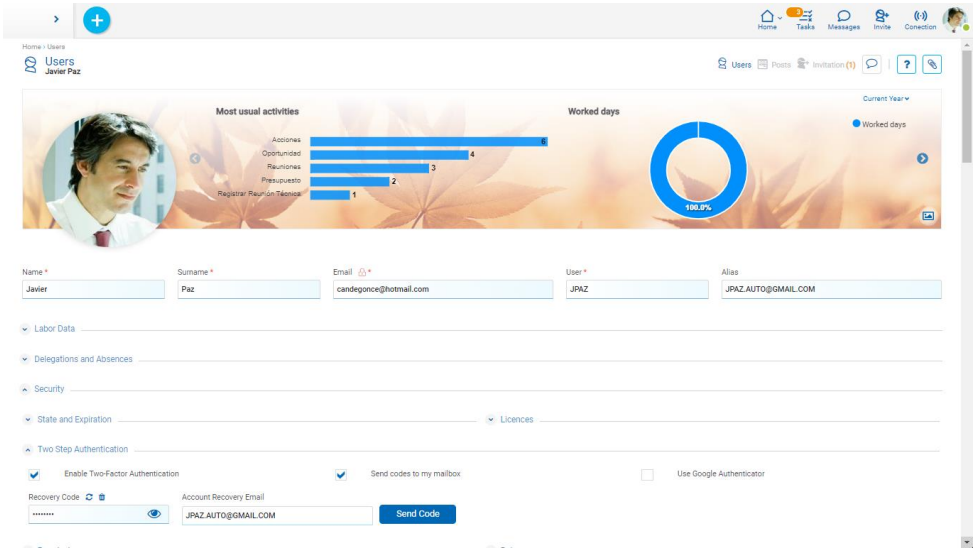
Product
Indicates the licensed application the user can use.

License Type
Indicates the type of user license that is assigned to the user to use the product.

Two Step Authentication

When the use of double authentication factor is enabled in the environment, this section is visualized.

If the Two Step Authentication configurable property is set to “Optional”, the user can decide whether to use 2FA to authenticate. On the other hand, if the value is "Mandatory", the user must configure the use of 2FA. When the user is authenticated for the first time, **Deyel** sends a first TOTP code by email and in turn generates an account recovery code.



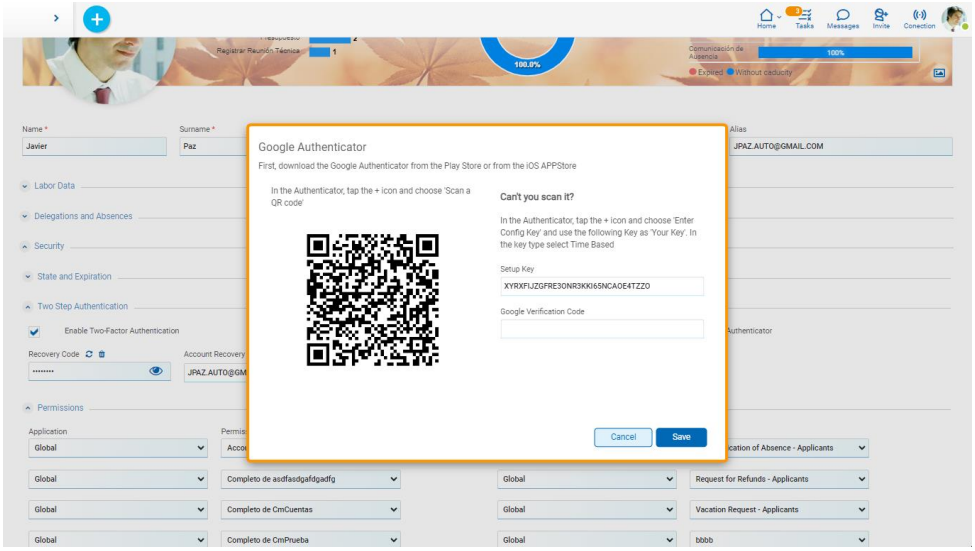
For security, to prevent improper access to this information by an unauthorized person, the user is required to confirm his identity by entering his password correctly to display this information.

Enable Two Step Authentication
The user can enable or disable the use of 2FA. If it is enabled, it indicates how they want to receive the TOTP codes, being able to choose any of the following options:

Send Codes to my Mailbox
If this option is selected **Deyel** sends the access codes to the user's email account, which has been reported in the profile.

Use Google Authenticator

If the use of the Google authenticator is selected, a panel is displayed that tells the user how to configure such application.



A QR code can be scanned or enter a key that is generated by **Deyel**.

The user must follow the indicated steps and to confirm that they have correctly installed and configured the authenticator, they are asked to enter a TOTP code generated by that application. Only when the code is verified correctly it is possible to save the use configuration of the authenticator.

When the authenticator has been configured and is in use, a configuration icon is displayed next to the field. By clicking, the same panel that allows to configure the use of the application on the same or another device is accessed.

> *The security administrator cannot access a user's 2FA settings. They can only disable 2FA to contemplate particular situations but not re-enable it, only the user himself can do it.*

Recovery Code

When the user cannot complete their two-step authentication because they do not receive their TOPT verification codes, it is necessary to offer a mechanism that allows them to overcome this inconvenience, using a recovery code.

When the user activates 2FA, a recovery code is generated. that can be consulted only by the user himself. To visualize the value, the icon must be used.

The icon allows to generate a new recovery code, the previous one is no longer valid.

The icon allows to remove the recovery code, indicating that this mechanism is not going to be used.

On the home page, when the user needs to pass 2FA, they can request access with this recovery code. If the correct code is reported, the 2FA is validated and the user can enter.

Account Recovery Email

When the user does not receive the TOTP verification codes and also does not remember his recovery code, another mechanism is provided for him to receive the access codes.

The user can optionally register an alternative email address, which must be different from their institutional account.

To verify that the user has access to the email account, an email is sent to that address, containing a TOPT validation code that must be entered in an attached field.

When accept is pressed and there was some change in the recovery email, it is verified that the code is valid. Otherwise, it is not allowed to register the indicated email account.

Permissions

The access permissions the user has assigned are listed. Each element from the list indicates the name of the permission and the application they belong to.

In the first positions the permissions the user inherits for belonging to an organizational unit, to a role or to a job position are displayed. These elements are displayed protected and it is not possible to delete them from the list. Hovering over each of these elements, it is displayed from which object the permission is inherited.

Following the inherited permissions, the permissions assigned directly to the user are shown. When creating or modifying the user the elements from the list can be added or deleted.

First the application is selected and then the permission of such application to be assigned to the user is selected.

When permissions from a licensed application are assigned, this one must be able to be used by the user. If the application is not available in the use license of the environment, the permission is protected and it is possible to delete it from the permissions list, but not modify the element.

> *The permission "Account Manager" can only be assigned to a unique user.*
> *In On-Premise installations of **Deyel**, this permission can be assigned or removed from a user by the security administrator.*
> *In Cloud installations, the site of **Deyel** must be used to change the user defined as "Account Manager".*

In every installation there exists a set of predefined permissions that can be assigned to the users.

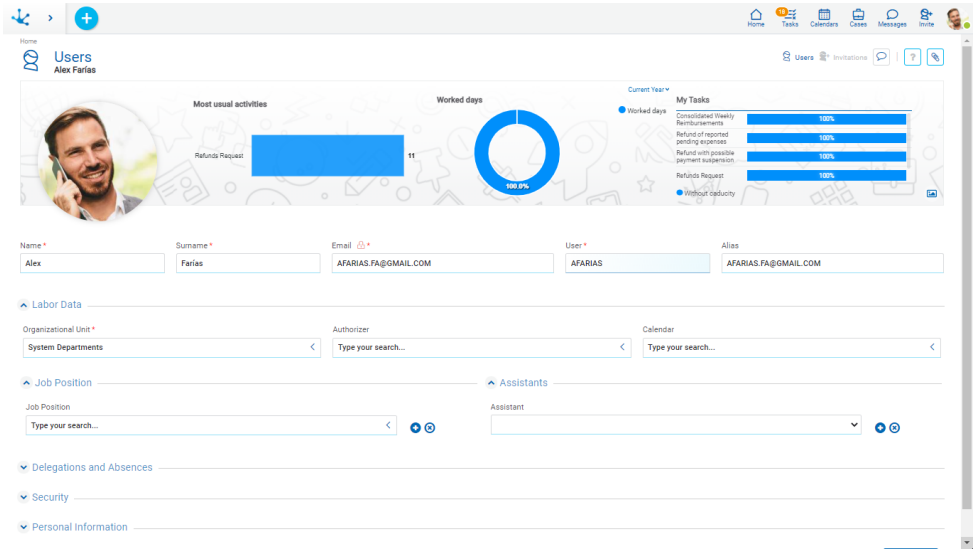
Roles

The roles the user performs are listed. The list is dynamically conformed when accessing the user profile, recovering the roles where the user or their organizational unit are actors. Each element indicates the role name and the application it belongs to.

In the first positions the roles the user inherits for belonging to a unit are displayed and then the roles where the user is an actor.

3.5.3.1.1.5. Personal Information

The personal data properties section contains information related to the user's individual characteristics, such as their ID, address, phone number, or whether they use social networks.



*An asterisk*on the label indicates that the property is required.*

Properties

The user profile includes the following properties:

Nationality
Birth Date
Identification Type and ID number
Phone Number / Extension

Addresses

The user can have multiple addresses, with the following properties for each of them:

Country
State
City
Postal Code
Street
Number
Department

Social Networks

The user profile includes information about the identity of the user in the different social networks (LinkedIn, Twitter, Facebook, YouTube and Skype).

In the case of Twitter, when publishing the corresponding adapter "Sign in with Twitter" option is enabled.
If the user has their Twitter session opened in the browser, they can grant permission to **Deyel** to publish tweets directly on behalf of the user. If not logged in, the user is redirected to the Twitter site to log in first.

When the authorization process is finished, from the user profile the authorized account is displayed, below their image.

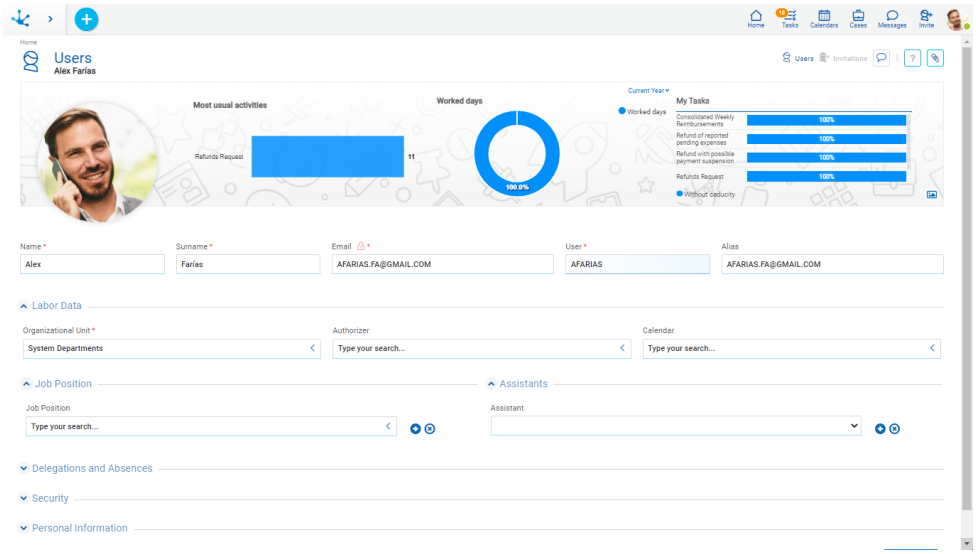
Users can unlink their Twitter account in different ways:
- From their profile, clicking on the icon ✖ visualized to the right of the Twitter property.
- From Twitter, the authorization for **Deyel** to publish tweets on behalf of the user is removed.

Additional Information

Remarks
Allows to load complementary data of the user.

3.5.3.1.1.6. Configuration

The configuration properties section contains information related to users of the 'Smart Thing' type. This section is only available for users of this type".



*An asterisk*on the label indicates that the property is required.*

Properties

Definition

Visible

Indicates if the chatbot is visible in the users list of the business social network Tedis.
When a chatbot is not visible, it can work as an assistant, but it is not possible to send it messages directly in a chat. For “Rest API Client” type this parameter cannot be modified.

Thing Type

Indicates the smart thing type that represents the user. It can take “Chatbot” or “Rest API Client” values.

Message Processor

This property is exclusive for “Chatbot” smart thing type. The business process that implements the processing of the messages the chatbot receives can be defined. Each time a chatbot receives a message, it starts a case of this process to generate the corresponding response.

Commands

This section is exclusive for “Chatbot” smart thing type and defines the list of commands the chatbot can interpret and respond to.
Executing a command implies executing the business process that implements it.

Participants

This section is exclusive for “Chatbot” smart thing type and defines the participants that can use the chatbot as an assistant.
If an organizational unit is selected, then any user from that unit can use the chatbot.

WebHooks

This section is exclusive for “Chatbot” smart thing type. Webhooks are interfaces that allow to integrate applications, that is, they allow to connect and exchange data between applications.
A chatbot can use multiple webhooks, each of them defines the URL used to contact other application, sending it data in JSON format.
When a chatbot receives a message, it forwards it to each of its webhooks.

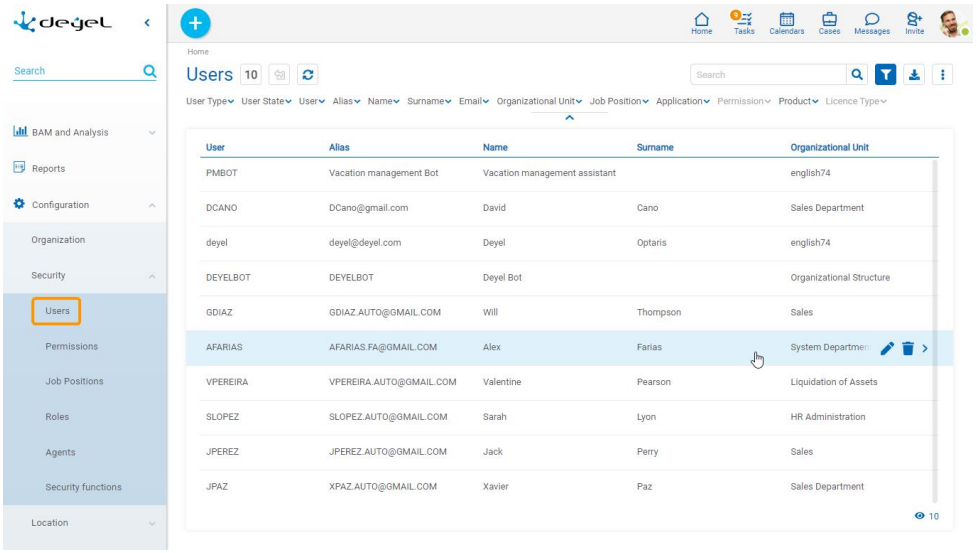
3.5.3.1.2. Grid

To do the creation and administration of users, the grid can be accessed from **Deyel** menu.

- From the option “Configuration”, selecting "Security" and then “Users”.
- From the search facility.

The users are visualized as a grid with the standard presentation of results grids, with facilities of:

- Sorting
- Search Bar and Filters
- Download Data
- Operations



The following properties of the users are visualized as grid columns:

- User
- Alias
- Name
- Surname
- Organizational Unit

Filters can be applied by the following properties:

- User Type
- User State
- User
- Alias
- Name
- Surname
- Email
- Organizational Unit
- Job Position
- Application
- Permission
- Product
- License Type

It is possible to perform operations on each line of the users grid. By clicking on the line the display of the selected user is done, while by means of the icons and its modification or elimination is done, respectively.

3.5.3.1.3. Operations

It is possible to do operations on each user, depending on the security permissions that the connected user has defined. When hovering over each of the lines of the grid the icons with available operations are visualized.

The exception is the operation "Create" that can be executed from the context menu in different ways.

- Hover the cursor over the icon  and select one of the icon on its right  or  corresponding to the option "Users" or "Smart Things".
- Click on the icon  and select the option "Users" o "Smart Thing" on the vertical panel displayed.

Create

Opens properties panel of the new user or smart thing with all properties as editable. The creation is done by pressing any of the buttons available and the user receives a message indicating that data have been saved.

> ***Deyel*** *can do users authentication against LDAP directory, in this case some of the user properties can be disabled, as such information can be recovered from LDAP directory.*

Buttons

- Accept: Confirms the creation of a new user or smart thing.
- Accept and Create: Confirms the creation of a user or smart thing and opens a new panel to create another one.

Considerations

When creating users some considerations are applied to entered properties.

Name - Surname
These properties are not informed for user type "Smart Thing".

Email
There cannot be more than one user with the same email. This property is optional for user type "Smart Thing".

User
The user code is made up automatically based on the name and surname. It is made up of the first initial of the name followed by a maximum of nine letters of the surname. It can be modified by the security administrator. There cannot be more than one user with the same code.

Alias
Takes the same value the property Email does. It can be modified by the security administrator. There cannot be more than one user with the same alias.

Profile Image
The profile image is entered by the users themselves. It is not a property that the security administrator can define when creating the user. **Deyel** assigns by default an image with the user´s initials.

Labor Data

Organizational Unit.

The organizational unit "My Company" is proposed by default. It can be modified by the security administrator.

Security

State and Expiration

Due Date
It must be greater than or equal to the current date.

User Licenses

In the properties Product and License Type the first product available in the use license is initially proposed. The security administrator can change this proposed value.

> *The list of applications only includes licensed applications available in the use license.*
> *When creating a new user, **Deyel** verifies that the amount of enabled users in the use license is not exceeded.*

Permission

Considering the values entered for the properties Product and License Type, the initial permission is proposed by **Deyel**.

Type of Use License	Initially Proposed Permission
Deyel - Participant	Deyel.EndUser
Deyel - Agile Modeler	Deyel.AgileModeler
Deyel - Deyel Modeler	Deyel.DeyelModeler
CRM - Administrator	CRM.Administrator
CRM - Participant	CRM.Seller
Contracts - Administrator	Contracts.Administrator
Contracts - Participant	Contracts.Seller

The security administrator can change the permission initially proposed.

> *The list of applications only includes licensed applications if they are available in the use license.*
> *The list of license types only shows the ones available in the use license.*

When permissions of a licensed application are assigned, it is verified that:

- The licensed application is in the list of applications the user can use.
- The type of the user license is compatible with the one required by the access permission.

> *The permission "Account Manager" can only be assigned to a unique user.*

Show

Opens the properties panel of the selected user, where properties are not editable. Depending on the security permissions, buttons available to operate with that user are enabled.

- Update
- Delete

Update

Opens properties panel of the selected user, with those properties that can be modified as editable. The update is done by pressing the button “Accept” and the user receives a message indicating that data have been saved.

> ***Deyel** can do users authentication against LDAP directory, in this case some of the user properties can be disabled, as such information can be recovered from LDAP directory.*

Considerations

Users can update their profile, modifying properties like their profile image, their delegated users and inform absences, among others. However some properties can only be modified by the security administrator, who has permission to execute the security function "Deyel - Modify users data".

Name - Surname
These properties are not informed for user type "Smart Thing".

Email
There cannot be more than one user with the same email. This property is optional for user type "Smart Thing".

Alias
There cannot be more than one user with the same alias.

Profile Image
The profile image is entered by the user themselves. It is not a property that the security administrator can define when modifying the user.

Delegations and Absences

Delegates
When a user is modified:

- Delegations occurred in the past, cannot be modified nor deleted.
- Current delegations, that is the ones that have the current date within the period, can be modified but the date must be kept within the period.
- Future delegations can be freely deleted or modified.

Absences
This list can be administrated by the users themselves or by the administrator user. Each element contains the absence type and its period.
Only the absences occurred during the last year are visualized, however the administrator user can see all of the absences.
When a user is modified:
- The absences occurred in the past can be modified or deleted only by the security administrator.
- Current absences, that is the ones that have the current date within the period, can be modified but the date must be kept within the period. The security administrator can delete or modify current absences.
- Future absences can be freely deleted or modified.

Security

State and Expiration

Due Date.
It must be greater than or equal to the current date.

User Licenses

Elements from the list of licensed applications can be added o deleted when modifying a user.

> *If an element is created or modified, only licensed applications available in the use license can be selected.*
> *Those elements from the list that make reference to applications not available in the environment are displayed protected, it is possible to delete them from the list but it is not possible to modify them.*
> ***Deyel** verifies that the amount of enabled users in the use license are no exceeded..*

Permissions

Elements from the list can be added or deleted when modifying the user.

The Application must be selected first and then the Permission for such application.

When permissions of a licensed application are assigned, it is verified that:
- If the application is not available in the environment at the moment of the modification, the permission is seen as protected and it is possible to delete it from the permissions list, but not modify it.
- If the application is available in the environment at the moment of the modification, it is verified:
  - The licensed application is one of the products the user can use.
  - The type of user license is compatible with the one required by the security permission.

> *The permission "Account Manager" can only be assigned to a unique user.*

Delete

Opens the properties panel of the selected user or smart thing, where their properties are not editable. The deletion is done by pressing the button “Accept” and the user receives a message indicating that data have been deleted.

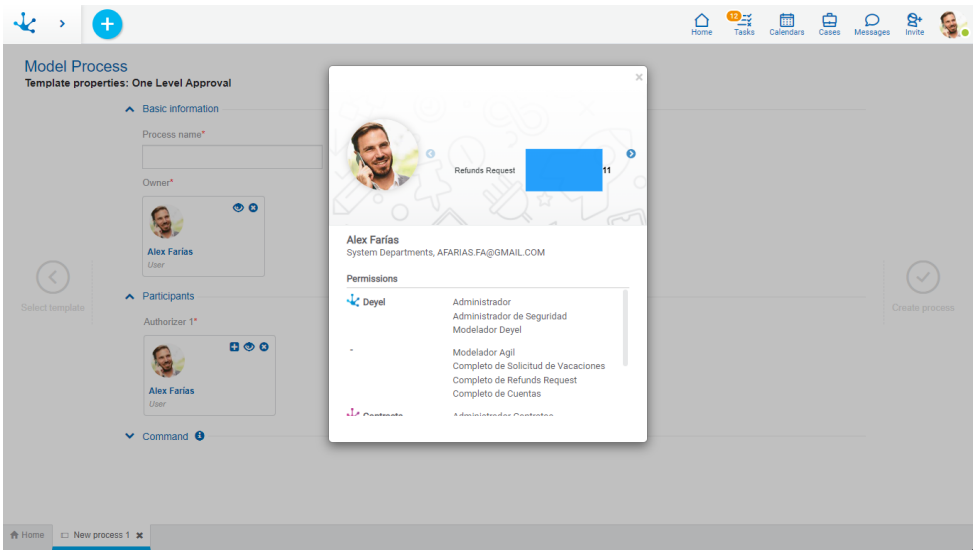
Considerations

When deleting users some verifications are made.

- The administrator of an organizational unit cannot be deleted.
- A coordinator or actor of a role cannot be deleted.
- A user with permission "Account Manager" cannot be deleted.
- Responsible or participant users of a process in use cannot be deleted.
- A logged in user cannot delete its own profile.

3.5.3.1.4. Show User Profile

From the different modelers, as well as from execution of processes and forms the query of the user profile can be made by pressing the icon and an informative panel is displayed.



Panel Elements

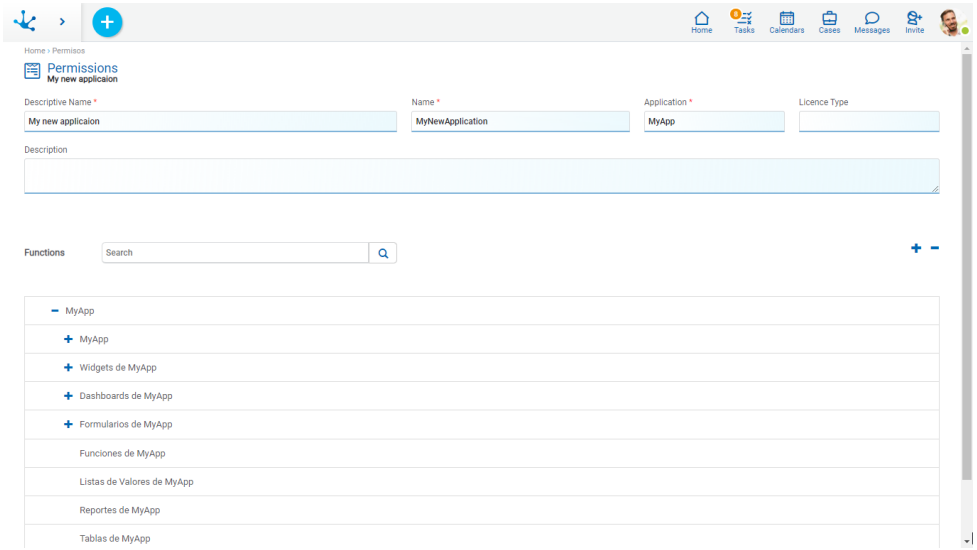
Some of the properties which were defined for the user are displayed, such as Name, Surname, Organizational Unit, Email and Permissions. These last are seen within a container where the scroll bar can be used.

At the top of the panel behavior indicators are seen, the icon allows to advance to the following indicator and the icon allows to return to the previous indicator.

3.5.3.2. Permissions

An access permission is an object that groups security functions.

Each user has assigned one or more security permissions that indicate which functionality they can execute within each application. This way, the operations that a user can execute are determined by the access permissions that they have been assigned.



> *An asterisk "*" on the label indicates that the property is required.*

Properties

Descriptive Name

Complete name of the permission. This text is the one visualized on the permissions grid.

Name

It is an abbreviated or reduced name. It is used when referring to the permission in an error message or in any mention that needs to abbreviate the descriptive name. This property does not admit blank spaces.

Application

Defines the application the permission belongs to, through the selection grom a list of available applications. The permission is made up of security functions of such application.
The licensed applications are included in such list only if the use license in the environment allows their use.
Access permissions of a licensed application, can only be assigned to users that have user license of such application. When a permission is created or modified, **Deyel** verifies that the authorized security functions are compatible with the license type required by the permission. For example, a permission that requires only a participant license cannot authorize functionalities that are typical of an agile modeler or a **Deyel** modeler.

When a permission belongs to a not licensed application, it can include functionalities of **Deyel** or of a solution, but they can be executed by a participant user.

> *Any permission can contain security functions of **Deyel** application.*
> *Any permissions of **Deyel** application can contain global security functions.*

Description
Details the information about the permission, extending what is expressed in the property Descriptive Name. Allows to enter a description of the facilities that the permission grants, without analyzing in detail the list of authorized functions.

Functions
Panel in which the set of security functions defined by the permission are authorized. They are visualized with the defined hierarchical structure.

Search
On top of the panel the bar of search of functions is visualized, from where filters can be applied to the function names.

The icon  allows to delete the defined selection criteria and all lines on the panel are displayed again without highlighting.

 Expands the hierarchical structure of security functions.

 Reduces the hierarchical structure of security functions only to higher level ones.

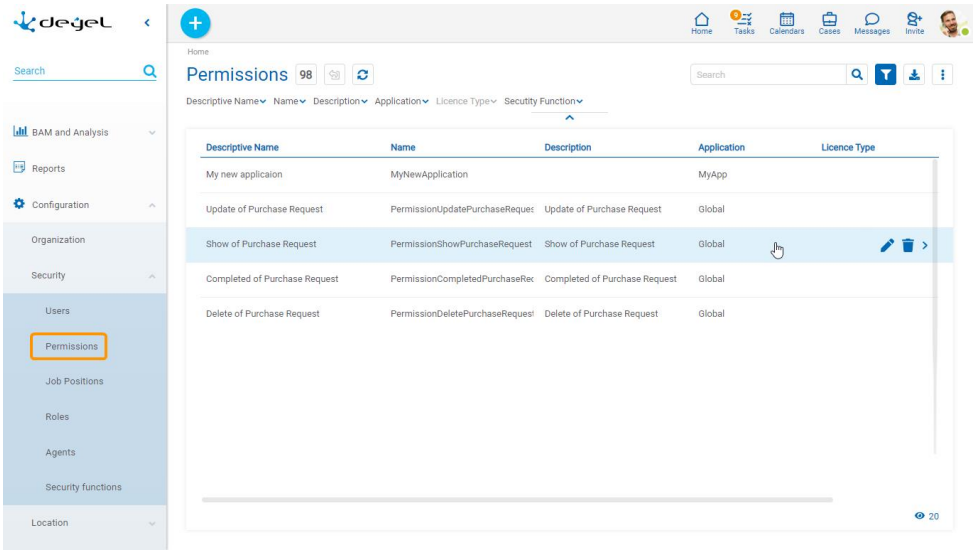
3.5.3.2.1. Grid

For creating and managing permissions, it is posible to access the grid from the menu of **Deyel**.

- From the option “Configuration”, selecting "Security" and then “Permissions”.
- From the search facility .

Permissions are visualized as a grid with the standard presentation of results grids, with facilities of:

- Sorting
- Search Bar and Filters
- Data Download
- Operations



The following properties of permissions are visualized as grid columns:

- Descriptive Name
- Name
- Description
- Application
- License Type

Filters can be applied by the following properties:

- Descriptive Name
- Name
- Description
- Application
- License Type
- Security Function

It is possible to perform operations on each line of the permissions grid. By clicking on the line the display of the selected permissions is done, while by means of the icons and its modification or elimination is done, respectively.

3.5.3.2.2. Operations

It is possible to perform operations on each permission, depending on the security permissions that the connected user has defined. When hovering over each of the lines of the grid the icons with available operations are visualized.

The exception is the operation "Create" that can be executed from the context menu in different ways.

- Hover over the icon and select the icon on its right , corresponding to the option "Permissions".
- Click on the icon and select the option "Permissions" on the vertical panel displayed.

Create

Opens properties panel of the new permission with all properties as editable. The creation is done by pressing any of the buttons available and the user receives a message indicating that data have been saved.

> *Functions*
> *In the panel in which the set of security functions authorized through the permission is defined, functions included in the panel can be selected.*

Buttons

- Accept: Confirms the creation of a new permission.
- Accept and Create: Confirms the creation of a permission and opens a new panel to create another one.

Show

Opens the properties panel of the selected permission, where properties are not editable. Depending on the security permissions of the user, buttons available to operate with that permission are enabled.

Buttons

- Update
- Copy
- Delete

Update

Opens properties panel of the selected permission, with those properties that can be modified as editable. The update is done by pressing the button "Accept" and the user receives a message indicating that data have been saved.

Copy

Opens a panel with the same properties of the selected permission, with its properties as editable. The necessary modifications must be done and when pressing the button "Accept", the user receives a message indicating that data have been saved. When the permission corresponds to a licensed application, it can only be copied if its license is valid.

> *Descriptive Name*
> *This property takes the value of the origin permission, adding a suffix indicating the number of copy generated. However this value can be modified.*

> *Name*
> *This property takes the value of the origin permission, adding by default a suffix indicating the number of copy generated. However this value can be modified.*

Delete

Opens the properties panel of the selected permission, where the properties are not editable. The deletion is done by pressing the button “Accept” and the user receives a message indicating that data have been deleted.

> *The permission cannot be deleted if there are users or roles who have it assigned..*
> *The permission "Account Manager" is a protected permission and it cannot be deleted.*

3.5.3.2.3. Predefined Permissions

In **Deyel** there are predefined security permissions with different characteristics.

> *This predefined permissions are protected by **Deyel** and cannot be modified, imported or deleted.*

Deyel

End User

- Accesses the user portal, where they find the tools which allow them to initiate cases, show the state of the cases they have initiated, show their to-do list and execute them, use messages, etc.
- Uses forms to administrate business entities, being able to show or modify each entity according to the access right they have on the entity.

Agile Modeler

- Includes the feature of the end user.
- Uses forms modeling wizard to build, document and implement the forms, processes and business entities of the organization.
- Defines roles and parameterization entities.
- Define business rules to include logic in their processes.
- Uses tools to export and import their models and information of the environment.

Deyel Modeler

- Includes the feature of the end user.
- Uses tools of graphical forms modeling and processes modeling to build, document and implement the process and business entities of the organization.
- Defines roles, agents, and parameterization entities.
- Implements more complex business logic using wizards that allow them to incorporate Java components.

- Implements the integration with other applications or external data bases.
- Implements behavior or more complex validations in forms, using the advanced edition of scripting.
- Uses tools to export and import their models and information of the environment.

Tester

- Includes the feature of the end user.
- Has tools that allow them to define and maintain the test plans of the different processes, record individual test cases and define random scenarios.
- Uses tools to automate the execution of test plans and monitor the results.

BAM

- User of BAM module of **Deyel**.

Guest in Tedis

- Includes the feature of the end user.
- It is used to limit the use of this collaborative platform. for example, a guest user can participate in conversations, but not initiate them.

Security Administrator

- Includes the feature of the end user.
- It has tools that allow them to administrate the organizational structure, the users and the roles they play.
- It can define functions and security permissions and assign them to the users and roles.

Administrator

- Includes the feature of the end user.
- Uses tools to configure:
  - The properties of the installation (DB connection, integration with mail server, etc.).
  - The use and upgrade of cache memories.
  - The parameterization entities of **Deyel**.
  - Holiday calendars and special dates
  - Glossaries
- Uses habitually the following functions:
  - Monitoring and reboot of scheduled tasks.
  - Events console and logs.
  - System log download.
  - Execution of tuning tools, such as case debugging.
  - Tools to export and import information of the environment.

Environment Configuration

- It only allows entry to the environment configuration.

Global

Account Manager

Allows to identify the user that performs as "Account Manager".
This is a protected security permission, that cannot be removed from the installation or modified. Only a user can have this security permission assigned. This is controlled in the creation and in the modification of users.

> *In On-Premise installations, this permission can be assigned or removed from a user by the security administrator.*
> *In Cloud installations, the site of **Deyel** must be used to change the user that performs as " Account Manager".*

Besides, this permission cannot be assigned to roles or job positions.

Modeler

Groups all modeling and usage security functions of the objects, giving agility to the assignment of permissions particularly at the moment of creating objects. Every object created has its security functions assigned to this permission.

Anonymous

This permission defines a set of functionalities that are authorized for every user.

The anonymous user has this permission assigned and every nominated user inherit the security functions from this permission implicitly

CRM

Seller

- Includes the features of **Deyel** end users, without access to agile forms.
- Includes access permissions for CRM sellers.

Sales Manager

- Includes the features of **Deyel** end users.
- Includes access permissions for CRM sales managers.

Administrator

- Includes the feature of the end user of **Deyel**.
- Includes access permissions for administration functions of Security and of CRM.

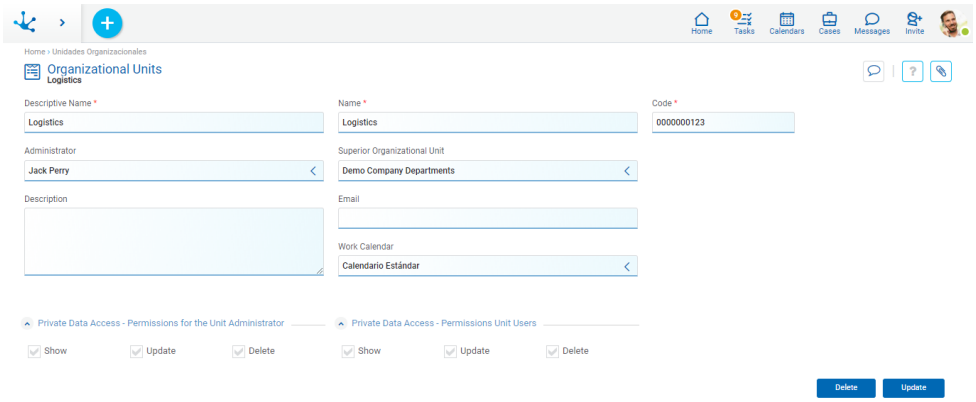
3.5.3.3. Organizational Units

Phase 8: Configuration > Organization

The organizational structure of Deyel defines the areas or the offices from the company and the relation of hierarchy between them.

Such areas or offices correspond to the organizational units that can be defined such as owner, initiator and participant when modeling a process.

When defining the users the organizational unit in which each one belongs must be indicated.



An asterisk "" on the label indicates that the property is required.*

Properties

Descriptive Name

Complete name of the organizational unit. This text is the one visualized in the organizational structure and as a grid column.

Name

It is an abbreviated or reduced name. It is used when referring to the unit in an error message or when there is the need to abbreviate the descriptive name. No blank spaces admitted. This name must be unique.

Code

Identification code of the organizational unit. This property is visible if the environment property Organizational unit coding type takes values "Manual" or "Semi Automatic", while it is not visible if the value is "Automatic".

Administrator

Optionally an user that performs as coordinator or head of the unit can be indicated.

Superior Organizational Unit
Allows to identify the superior organizational unit inside the hierarchical structure.

Description
Property where the use of the organizational unit can be detailed so as to extend what is expressed in the Descriptive Name.

Email
Email associated to the organizational unit.

Work Calendar
Allows to select the work calendar associated to the organizational unit. If a calendar is not defined for the organizational unit, **Deyel** uses the one of the closest higher-level unit with a defined calendar. The root unit of all the hierarchy has defined a calendar by default.
In the query of an organizational unit, when hovering over this property, a legend is displayed informing if the calendar is defined in the unit or in one of its superior units, or if it uses the default calendar.

Private Data Access - Permissions for the Unit Administrator
For the unit administrator, the access permits to the private data of their work teams are established.

Show
Can the unit administrator see the private data of their subordinates? (Yes/No)

Modify
Can the unit administrator modify the private data of their subordinates? (Yes/No)

Delete
Can the unit administrator delete the private data of their subordinates? (Yes/No)

Private Data Access - Permissions Unit Users
For a user that is part of the unit which establishes the access permits to private data of their colleagues.

Show
Can a unit user see the private data of their colleagues? (Yes/No)

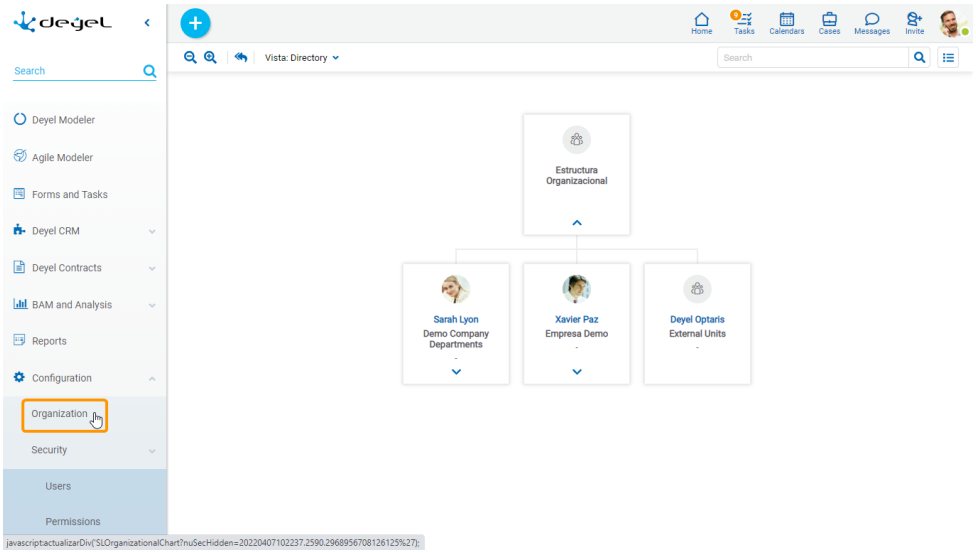
Modify
Can a unit user modify the private data of their colleagues? (Yes/No)

Delete
Can a unit user delete the private data of their colleagues? (Yes/No)

3.5.3.3.1. Organizational Structure

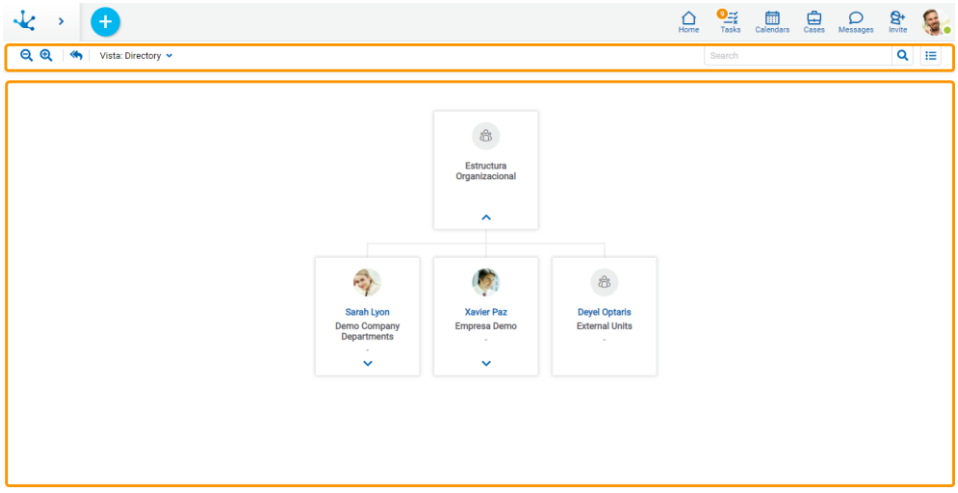
The Deyel menu can be accessed, to create and administer the organizational units.

- From the “Configuration” option, selecting “Organization”.
- From the search facility.



Sections

- Top Bar
- Grid



3.5.3.3.1.1. Top Bar

From the top bar different options related to the content and presentation of the organization chart can be selected.

Zoom Out

Decreases the display size of the diagram.

Zoom In

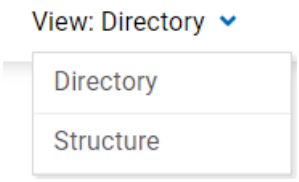
Increases the display zoom of the diagram.

Return to Initial State

The selected view is returned to the initial display, the search is cleared and the expanded units are closed.

View

A panel to select the view is displayed.





Fast Search

When entering a text in the bar and pressing the icon or "enter" key, all the organizational units that contain in the Descriptive Name the entered text, are displayed in the list.

The chosen unit is highlighted with a blue border and with its dependent units expanded.

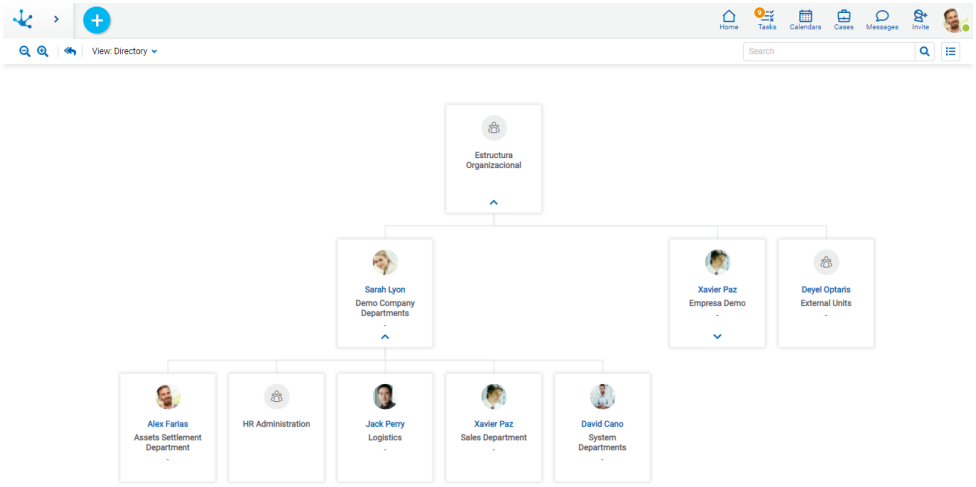
Change to Grid Mode

Allows to change the display to grid mode.

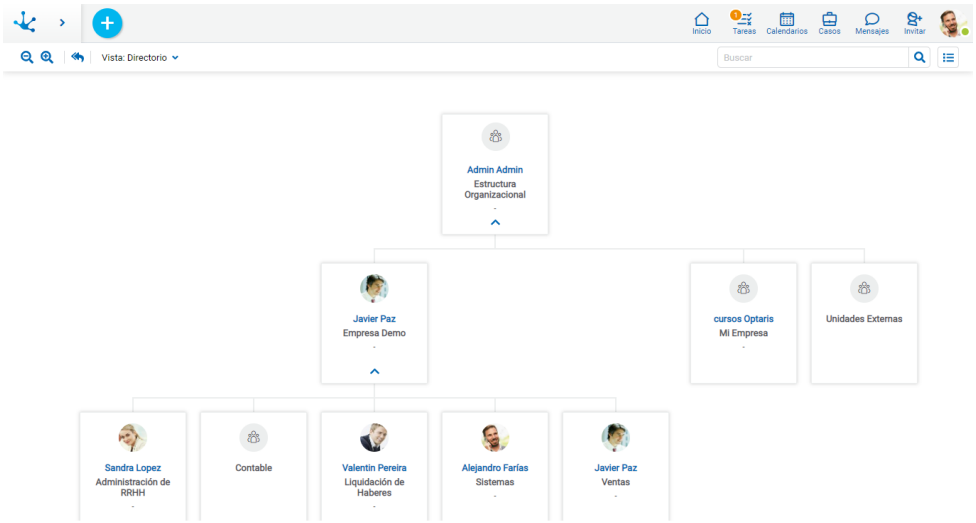
3.5.3.3.1.2. Organization Chart

The organizational units are displayed hierarchically. In the superior panel the "Structure" and "Directory" view types can be selected. The location within the structure depends on the definitions made in the Superior Organizational Unit property of each unit.

In the "Structure" view, each unit is identified by its Descriptive Name.



In the "Directory" view, the name and surname of the administrator of each unit, with their profile image, are also included.



Allows to display the superior level of the structure, when an element is selected as a search result.

Allows to display the direct dependent units.

Allows to hide the direct dependent units.

Allows to access the detailed query of the organizational unit properties.

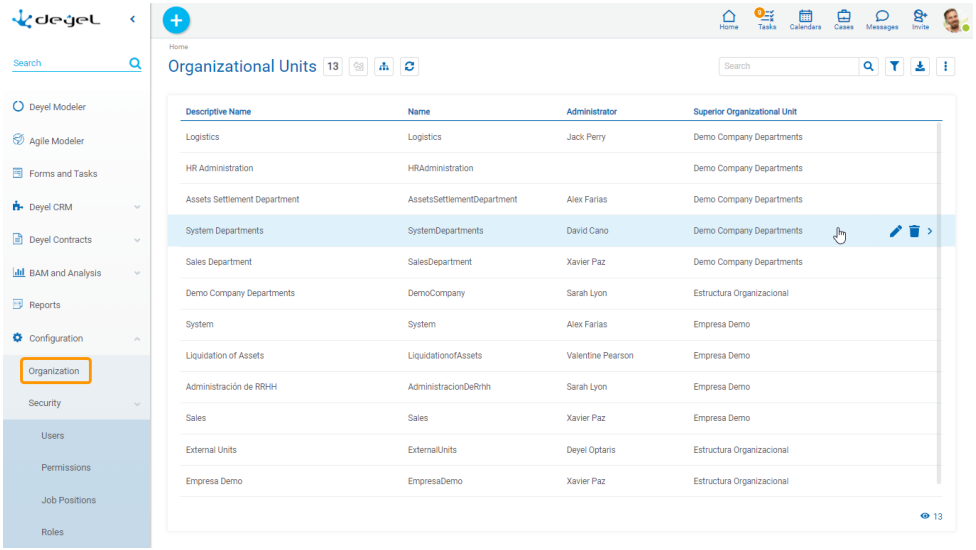
> *The organizational chart image can be enlarged or reduced using the mouse wheel. The organizational chart can be moved by clicking and dragging it.*

3.5.3.3.2. Grid

The grid of organizational units allows to visualize them with standard presentation of results grids, with facilities of:

- Sorting
- Search Bar and Filters
- Download Data
- Operations

The icon  allows to change the display to grid mode.



The following properties of the organizational units are visualized as grid columns:

- Descriptive Name
- Name
- Administrator
- Superior Organizational Unit

Filters can be applied by the following properties:

- Descriptive Name
- Name
- Administrator
- Superior Organizational Unit

It is possible to make operations on each grid row of the organizational units. By clicking on the row a query of the selected unit can be made and through the icons  and  it can be modified or deleted respectively.

3.5.3.3.3. Operations

Both from the organizational chart and from the grid operations can be done on each organizational unit.

Hovering the cursor over each element of the organization chart displays the icon that allows to show the properties of the selected unit. While hovering it over each of the lines of the grid icons corresponding to the operations available for each unit are displayed.

Create

The operation "Create" can be done from the context menu in different ways.

- Hover over the icon and select the icon on its right , corresponding to the option "Organizational Units".
- Click on the icon and select the option "Organizational Units" on the vertical panel displayed.

Opens properties panel of the new organizational unit with all the properties as editable. The creation is done by pressing any of the buttons available and the user receives a message indicating that data have been saved.

> *Code*
> *Depends on property configuration Organizational unit coding type, if the value is "Manual" the code is mandatory, if it is "Semi Automatic" the code is optional and if it is "Automatic" the code is not visible.*

Buttons

- Accept: Confirms the creation of a new organizational unit.
- Accept and Create: Confirms the creation of an organizational unit and opens a new panel to create another one.

Show

Opens the properties panel of the selected organizational unit, where properties are not editable. Depending on the security permissions of the user, buttons available to operate with that organizational unit are enabled.

Buttons

- Update
- Delete

Update

Opens properties panel of the selected organizational unit, with those properties that can be modified as editable. The update is done by pressing the button “Accept” and the user receives a message indicating that data have been saved.

> *Administrator*
> *When the unit has an administrator defined and the unit has been modeled as owner or participant in a process that is in use, this property can be modified entering a new administrator but cannot be defined with no value.*

Delete

Opens the properties panel of the selected organizational unit, where properties are not editable. The deletion is done by pressing the button “Accept” and the user receives a message indicating that data have been deleted.

> *There should be no user belonging to the unit.*
> *The unit should not be actor in any role.*
> *The unit should not be owner nor participant in any process that is in states "Published" or "Modified".*

3.5.3.4. Roles

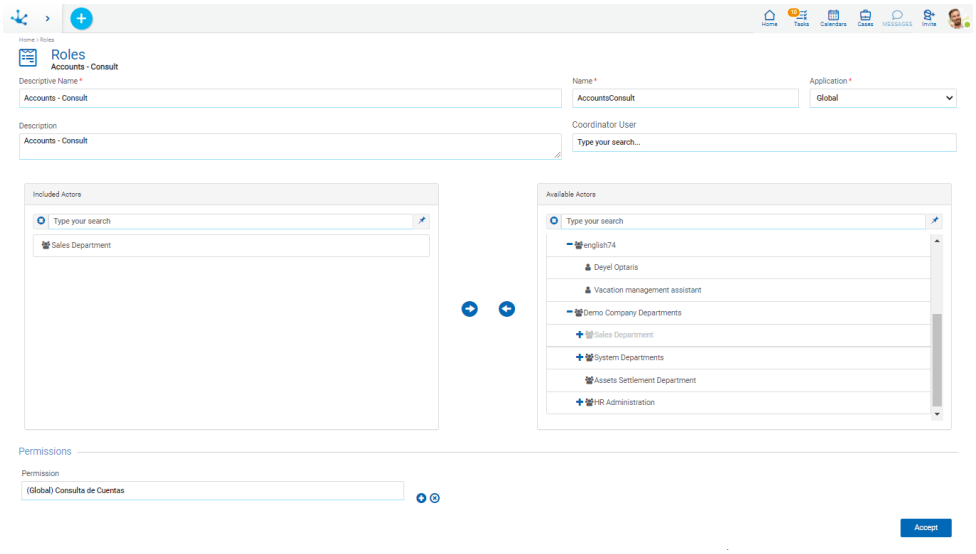
Phase 8: Configuration > Roles and job positions

While the organizational structure of **Deyel** defines the areas or offices of the company, the hierarchical relations between them and the actors that make it up, the roles allow to define groups of actors that have activities and common responsibilities, but are cross or parallel to the organizational structure.

The actors of a role can be users, organizational units or users that belong to these units. Each role can have defined a set of security permissions, that are assigned to the users comprising such role or to the users whose organizational unit integrates the role.

In the modeling of business processes, this concept is used to define that an activity can be executed by any of the members of the role. In agile forms, roles can be used to define who can do operations on form instances.

An organizational unit and a user can belong to multiple roles.



An asterisk "" on the label indicates that the property is required.*

Properties

Descriptive Name

Complete name of the role. This text is the one visualized on the roles grid.

Name

It is an abbreviated or reduced name. It is used when referring to the role in an error message or in any mention that needs to abbreviate the descriptive name. This property does not admit blank spaces.

Application

Defines the application the role belongs to, through the selection of a list of available applications. The licensed applications are included in such list only if the use license of the application is valid in the environment.

Description

Details the information about the role, extending the property Descriptive Name. Allows to document the purpose for which the role has been generated, in which process is used, etc.

Coordinator User

A coordinator user of the role is optionally defined, doing the selection from the user search.

Current Actors

Panel in which the users and organizational units that are role actors, can be defined and visualized as a grid.
The role must have at least a current actor, that is the list of current actors cannot be empty.
On top of the panel a search bar of actors is visualized, from where filters can be applied to the user names and organizational units to reduce the grid.
It is possible to move elements from this panel from or to the panel Available Actors, as detailed under the title Transfer of Actors.

Available Actors

Panel from where you can select users and/or organizational units available to be defined as role actors.
On top of the panel a bar of search of actors is visualized, from where filters can be applied to the user names and organizational units to reduce the grid.
It is possible to move elements from this panel from or to the panel Current Actors, as detailed under the title Transfer of Actors.

Permissions
This property allows to define security permissions to the role actors, that is that actors who integrate the role or that play in units which are role actors, inherit the security permissions defined here. It is an indirect way of assigning security permissions.

> *It is not allowed for the permission "Account Manager" to be associated to roles.*

Actors Transfer

The transfer of actors between the panels Current Actors and Available Actors is done by selecting the actors to transfer in the source panel and pressing the icon corresponding to the direction.

 Transfers the selected actors in the panel of available actors to the panel of current actors.
 Transfers the selected actors in the panel of current actors to the panel of available actors.

The icon  allows to select all the actors of the corresponding panel, or remove the selection.

Actors Search

The search of actors is done by clicking on the selected lines, or using the search facility of each panel of actors. If a text is entered and the key "Enter" is pressed, the panel is updated showing only the actors whose name contains the text entered. The results are highlighted in yellow.
The icon  allows to delete the defined selection criteria and all lines are displayed again on the panel.

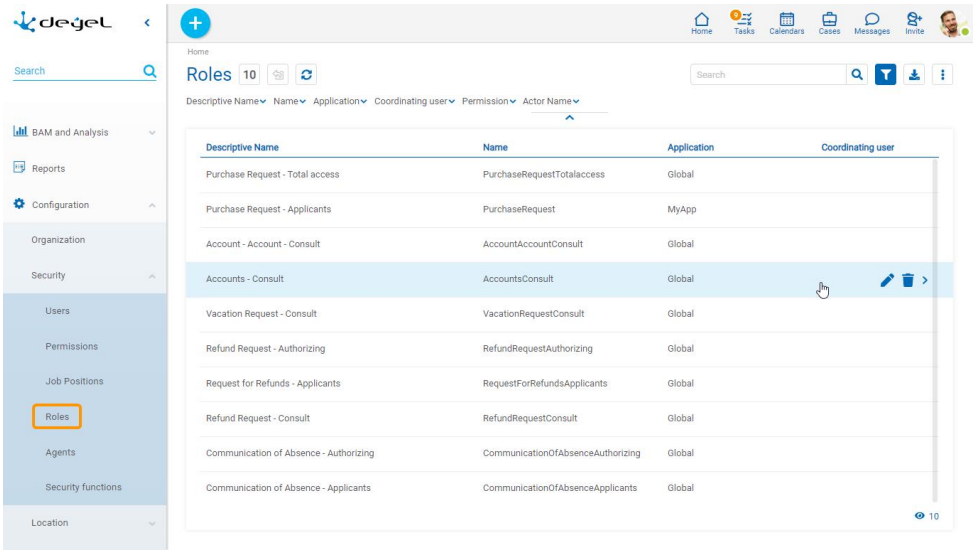
3.5.3.4.1. Grid

For creating and managing roles, it is posible to access the grid from the menu of **Deyel**.

- From the option “Configuration”, selecting "Security" and then “Roles”.
- From the search facility.

Roles are visualized as a grid with the standard presentation of results grids, with facilities of:

- Sorting
- Search Bar and Filters
- Data Download
- Operations



The following properties of roles are visualized as grid columns:

- Descriptive Name
- Name
- Application
- Coordinator User

Filters can be applied by the following properties:

- Descriptive Name
- Name
- Application
- Coordinator User
- Actor Name

It is possible to perform operations on each line of the roles grid. By clicking on the line the display of the selected role is done, while by means of the icons and its modification or elimination is done, respectively.

3.5.3.4.2. Operations

It is possible to do operations on each role, depending on the security permissions that the connected user has defined. When hovering over each of the lines of the grid the icons with available operations are visualized.

The exception is the operation "Create" that can be executed from the context menu in different ways.

- Hover the cursor over the icon and select the icon on its right corresponding to the option "Roles".
- Click on the icon and select the option "Roles" on the vertical panel displayed.

Create

Opens properties panel of the new role with all properties as editable. The creation is done by pressing any of the buttons available and the user receives a message indicating that data have been saved.

Buttons

- Accept: Confirms the creation of a new role.
- Accept and Create: Confirms the creation of a role and opens a new panel to create another one.

Show

Opens properties panel of the selected role, where properties are not editable and the icons for the transfer of actors are not visualized, however the search of actors is enabled. Depending on the security permissions of the user, buttons available for operating with the role that is being consulted are enabled.

Buttons

- Update
- Delete

Update

Opens properties panel of the selected role, where properties that can be modified are editable. The update is done by pressing the button "Accept" and the user receives a message indicating that data have been saved.

> *Application*
> *It is not allowed to modify the role if the application is licensed and its use license is not valid.*
> *Coordinator User*
> *When the role is responsible or participant of a process that is in use, it is not possible to delete the definition of the coordinator, although it is possible to define a new one.*

Delete

Opens properties panel of the selected role, where properties are not editable and the icons for the transfer of actors are not visualized, however the search of actors is enabled. The deletion is done by pressing the button "Accept" and the user receives a message indicating that data have been deleted.

> *A role cannot be deleted if it is responsible or participant of a process in use.*

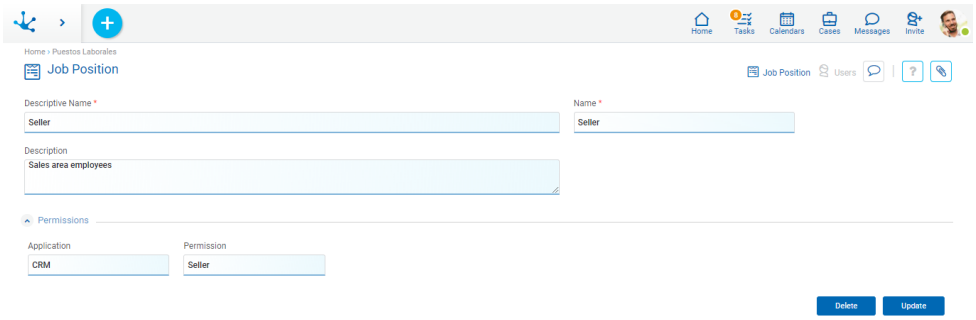
3.5.3.5. Positions

Phase 8: Configuration > Roles and job positions

A job position defines in **Deyel** the activities that a user or group of users develop within the organization.

Each job position can have assigned a set of security permissions, that are inherited by the users who occupy that position, which allows to simplify assigning permissions to users.

This concept is used to associate users to the job position they occupy and know which are their skills. From a job position it can be visualized the relation with the users in the upper right section, with the number of users that are associated to the job position consulted. If you click on the name of the relation “Users”, a grid with the users associated to the job position is visualized if the number is greater than 1, or the properties panel of the user in case that such number is 1.



An asterisk "" on the label indicates that the property is required.*

Properties

Descriptive Name
Complete name of the job position. This text is the one visualized on the grid of job positions.

Name
It is an abbreviated or reduced name. It is used when referring to the job position in an error message or in any mention that needs to abbreviate the descriptive name. This property does not admit blank spaces.

Description
Details the information about the job position, extending the property Descriptive Name.
Allows to include a detailed description of the tasks that the user occupying that job position does.

Assigned permits
Security permissions can be added to the job position.

Application

Allows to define the application list that can be used by the users who have the job position assigned within their profile.
Permits
Allows to define a predefined security permission for each application entered in the property Application.

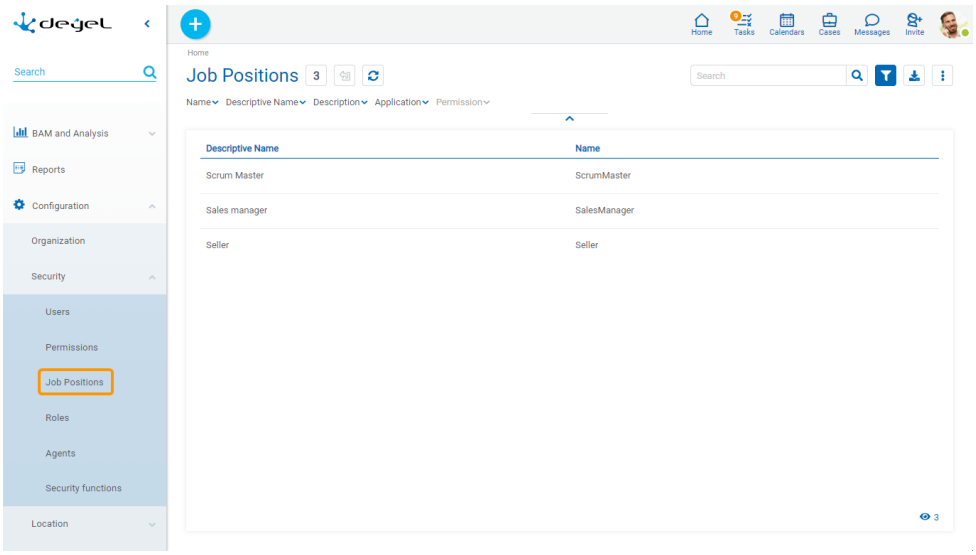
3.5.3.5.1. Grid

For creating and managing job positions, it is posible to access the grid from the menu of **Deyel**.

- From the option “Configuration”, selecting "Security" and then “Job Positions”.
- From the search facility.

Job positions are visualized as a grid with the standard presentation of results grids, with facilities of:

- Sorting
- Search Bar and Filters
- Data Download
- Operations



The following properties of job positions are visualized as grid columns:

- Name
- Descriptive Name

Filters can be applied by the following properties:

- Name
- Descriptive Name
- Description
- Application
- Permission

It is possible to perform operations on each line of the job positions grid. By clicking on the line the display of the selected job position is done, while by means of the icons  and  its modification or elimination is done, respectively.

3.5.3.5.2. Operations

It is possible to perform operations on each job position, depending on the security permissions that the connected user has defined. When hovering over each of the lines of the grid the icons with available operations are visualized.

The exception is the operation "Create" that can be executed from the context menu in different ways.

- Hover the cursor over the icon  and select the icon on its right  corresponding to the option "Job Position".
- Click on the icon  and select the option "Job Position" on the vertical panel displayed.

Create

Opens properties panel of the new job position, where properties are editable. The creation is done by pressing any of the buttons available and the user receives a message indicating that data have been saved.

Buttons

- Accept: Confirms the creation of a new job position.
- Accept and Create Confirms the creation of a job position and opens a new panel to create another one.

Show

Opens the properties panel of the selected job position, where properties are not editable. Depending on the security permissions of the user, buttons available to operate with that job position are enabled.

Buttons

- Update
- Delete

Update

Opens properties panel of the selected job position, with those properties that can be modified as editable. The update is done by pressing the button “Accept” and the user receives a message indicating that data have been saved.

Delete

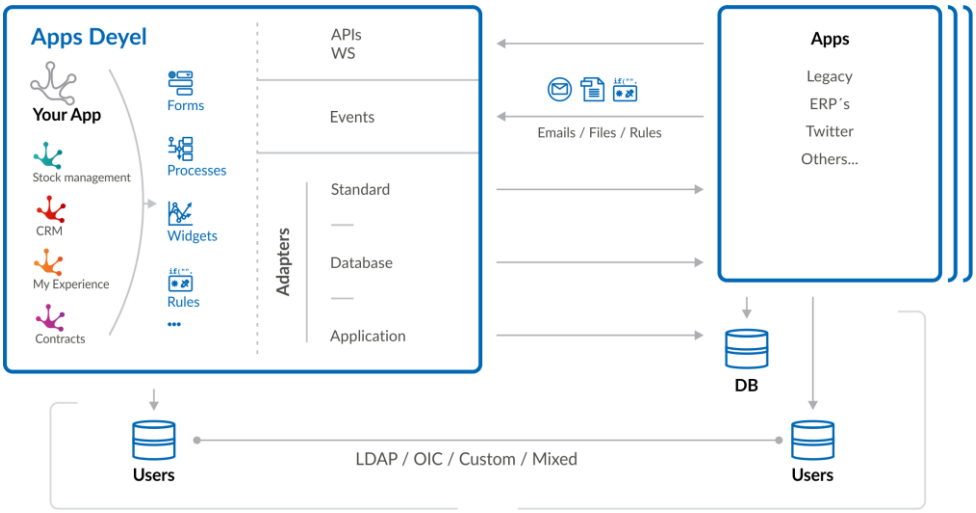
Opens the properties panel of the selected job position, where its properties are not editable. The deletion is done by pressing the button “Accept” and the user receives a message indicating that data have been deleted.

3.6. Integration

Applications developed with **Deyel** can interact with other applications, by using different tools and services of the platform.

On the one hand, **Deyel** allows to integrate its applications with external applications and existing data sources, that is, Legacy systems, databases, Twitter or other external applications, making use of adapters based on web services, Java and JDBC.

It also has a web services API based on the Richardson maturity model design guides, level 3, which allows any external application to **Deyel** to interact with applications developed with **Deyel**, or with solutions, such as CRM, Contracts or Stock Management, among others. It also allows the use of emails, rules, files and other events.



Applications developed with **Deyel** have their own user directory and if necessary they can be integrated through LDAP, OIC, in a personalized and mixed way, that is, allowing groups of users with different authentications.

Integration Facilities

Applications can be integrated using different platform tools and services.

Integration with External Applications

Applications developed with **Deyel** can be integrated with other applications by different means, either consulting or updating their information.

- Rules based on standard adapters

This integration is done using advanced rules that use the SDK facilities of Deyel and adapters that allow access to web services from external providers.

- Rules based on database adapters that use the Java JDBC protocol

This integration is done using adapters provided by **Deyel** for the most popular database engines on the market.

- Rules based on application adapters

This integration is done using adapters that allow defining access to applications on the market that expose services.

Integration from External Applications

Other applications can be integrated with applications developed with **Deyel**, mainly using Deyel Rest API. There are also other means that can be very useful, depending on the characteristics of the applications to be integrated.

- Use of web services API
  Through the use of Deyel Rest API other applications can interact with applications developed with **Deyel**, having access to forms and cases through web services.

- Other media
  Integration can be done by using emails, rules and files.

Authentication Schemas

The authentication procedure ensures that any use of **Deyel** and the use of its solutions are performed by "someone" or "something" that can be identified as a user, within the user registry.

Deyel uses different authentication schemas to respond to this request.

Using the Integration

Integration rules can be used from the different objects of the applications developed with **Deyel**.

- Rules in forms and processes
  Advanced adapter-based rules can be used using embedded rules.

- Automatic actions of process activities
  Advanced adapter-based rules can be used in the modeling of automatic actions of process activities.

- Scheduled tasks and events
  Advanced adapter-based rules can also be used in the modeling of events and scheduled tasks.

3.6.1. Deyel Rest API

Deyel Rest API allows developers and external applications to programmatically interact with **Deyel** in a simple way, achieving fast integration at low coding cost.

The layout is modeled as a hierarchy where each node is either a single resource or a collection resource. By convention, the former is called "resource" and the latter is called "collection".

Resources are known as API nouns.

A collection contains a list of resources of the same type.

Example: The users collection contains a list of users where each user is a resource.

According to the REST architecture these resources can be used with a small number of methods.

Methods are known as API verbs.

With the HTTP protocol, resource names map naturally to URLs, while methods map via the HTTP protocol:

- GET method
- POST method
- PUT / PATCH method
- DELETE method

The examples that are explained in each of the methods that are implemented on a project in Postman and the collection can be opened from here.

> *To use the endpoints included in the collection, it is necessary to replace the value "myenvironment" in the variables of the collection with the value corresponding to the environment to be used.*

Security

The Rest API security of Deyel is implemented using the standard OAuth 2.0, which allows access to the provided resources, using a single Bearer type access token.

To start using Deyel Rest API, it is first necessary to obtain the access token, which is used in methods that access and interact with resources.

3.6.1.1. GET method

The GET method uses the URL path to determine the collection or resource to get with, optionally, additional parameters that determine the set of results to get for the request.

- It uses the HTTP GET verb.
- There is no request body, that is, the configuration does not declare a "body" clause.

- The displayed resource or resources are contained in the response body.
- Each resource contained in the response body contains hypermedia links (HATEOAS), which allows the API consumer to dynamically navigate to related resources.

In all the examples, the 'Account' object is used, whether it's an entity or a form.

Example: Get all "Account" object instances.

```
GET /forms/CRM_ACCOUNT/instances/
```

Optional Parameters

If the request returns a collection of resources, the GET method URL may contain optional parameters that allow customizing the results obtained by that request.

None or several of these parameters may be included in each request, separated by the "&" sign. The general syntax is:

```
GET /collection-name?parameter=value[{&parameter=value}]
```

search

Through this parameter, the response content can be filtered using RHS notation. This notation is:

```
fieldName=operator:value
```

One or several filters separated by "," may be included within this parameter.

The "fieldName" value is that of the Identifier property of the resource on which the request is being made.

The possible values for "operator" are:

Value	Description
eq	Equal
neq	Not equal
like	Contains
sw	Starts with
nsw	Does not start with
ew	Ends with

Value	Description
new	Does not end with
gt	Greater than
gte	Greater than equal to
lt	Less than
lte	Less than equal to
in	Included in
nin	Not included in

Example 1: Search all "Account" object instances that have "JPerez" value in the field corresponding to the seller.

```
GET /forms/CRM_ACCOUNT/instances?search=cdInstanceOwner=eq:JPerez
```

Example 2: Search all "Account" object instances that have "JPerez" value in the field corresponding to the seller and the value of the field corresponding to the company name that starts with the “Acme” value.

```
GET
/forms/CRM_ACCOUNT/instances?search=cdInstanceOwner=eq:JPerez,dsCompany=sw:
Acme
```

sort

This parameter allows sorting the response content according to the needs of the API consumer. The syntax is:

```
[+/-]fieldName
```

One or several sorting criteria separated by “,” may be included within this parameter.

Prefixes "+" or "-" indicate whether it is an ascending or descending sorting. If omitted, the sorting is assumed to be ascending.

Example 1: Get the"Account" object instances in ascending sorting by the number of employees in the account.

```
GET /forms/CRM_ACCOUNT/instances?sort=qtEmployee
```

Example 2: Get the "Account" object instances in ascending sorting by the company name and within it, in descending sorting by the number of employees in the account.

```
GET /forms/CRM_ACCOUNT/instances?sort=dsCompany,-qtEmployee
```

per-page

The response to a request is paginated. It means that the API returns a page of the results obtained by the request.

The size of these pages can be determined using this parameter, that is, the maximum amount of resources that the request response contains. The syntax is:

```
per-page=20
```

This parameter, like the rest of them, is optional. If omitted, the API takes the following configurable property value Number of lines per page when searching for forms.

Example: Define the page size with value 20.

```
GET /forms/CRM_ACCOUNT/instances?per-page=20
```

page-number

In combination with the per-page parameter, this parameter allows to determine the subset of results corresponding to a request that are included in the response. The syntax is:

```
page-number=3
```

As this parameter is also optional, if omitted, the default value is 1.

Example 1: Return page 2 with pages having 10 resources, that is, resources from 11 to 20.

```
GET /forms/CRM_ACCOUNT/instances?page-number=2
```

Example 2: Return page 2 with pages having 30 resources, that is, resources from 31 to 60.

```
GET /forms/CRM_ACCOUNT/instances?per-page=30&page-number=2
```

HATEOAS

Each resource contained in the response to a given request includes hypermedia links (Hypermedia as the Engine of Application State) that allow the API consumer to dynamically navigate to resources and methods related to the resource being accessed.

In all cases, the response for each resource includes a list of the type:

- rel: HTTP verb to be invoked to access the related resource or method.
- label: Description of the related resource or method.
- href: URL of the related resource or method.

Example: Make a request to get data from instance 93 of the "Account" object.

```
GET /forms/CRM_ACCOUNT/instances/93
```

The response to this request includes, in addition to properties of the requested form, the information to execute the methods of related resources.


```
{
    "idForm": 93,
    "cdStatus": "1",
    "phone": null,
    "cdUserLastUpdate": "JPEREZ",
    "dtLastUpdate": "1575059822000",
    "email": null,
    "dsIndustry": "6",
    "cdUserStore": "JPEREZ",
    "dtStore": "1540846918000",
    "dsOwner": "Perez, Juan",
    "dtOpening": "1540782000000",
    "dsHolding": null,
    "links":[
        {
            "rel": "PUT",
            "label": "Edit",
            "href":
            "http://miAmbiente/v1.0/forms/CRM_ACCOUNT/instances/93"
        },
        {
            "rel": "PATCH",
            "label": "Partial Edit",
            "href":
            "http://miAmbiente/v1.0/forms/CRM_ACCOUNT/instances/93"
        },
        {
            "rel": "DELETE",
            "label": "Remove",
            "href":
            "http://miAmbiente/v1.0/forms/CRM_ACCOUNT/instances/93"
        }
    ]
```

```
}
```

This information dynamically allows:

- Editing instance 93 of the object
  `PUT /forms/CRM_ACCOUNT/instances/93`

- Updating instance 93 of the object
  `PATCH /forms/CRM_ACCOUNT/instances/93`

- Deleting instance 93 of the object
  `DELETE /forms/CRM_ACCOUNT/instances/93`

3.6.1.2. POST method

The POST method is used to create a new resource to a collection.

- It uses the HTTP POST verb.
- The url only contains the name of the collection in which a new resource is created.
- The values of the new resource attributes are specified in the request body, that is, the configuration declares a "body" clause.
- The response body contains the information of the resource identifier created in the form:
  name: Name of the resource identifier attribute (idObjectName).
  value: Value of the resource identifier attribute (idObjectValue).
  dtLastUpdate: Resource creation date.

Example: Create an "Account" object instance.

```
POST /forms/CRM_ACCOUNT/instances
```

The request body ("body" clause) contains:

```
{
    "cbIsPartner": false,
    "cdInstanceOwner": "JPerez",
    "cdStatus": "Active",
    "cdType": "1",
    "dsCompany": "Tenaris",
    "dsDescription": "",
    "dsHolding": "TECHINT",
    "dsIndustry": "4",
    "dsOwner": "Juan Perez",
   "dsSource": "Referred",
    "dtOpening": "1540782000000",
    "emails":"",
    "flLogo": null,
   "idPartnerCompany": null,
    "nuIdentifNumber": "30-00000000-5",
```

```
    "qtAnualRev":"1500000000",
    "qtEmployee":1000,
    "qtScore":90,
    "lsAddress":[
       {
          "lsAddress/dsAddress":"Della Paolera 333",
          "lsAddress/dsCity": "buenos aires",
          "lsAddress/dsCountry":"argentina",
          "lsAddress/dsState":"buenos aires",
          "lsAddress/tpAddress":"1"
       }
    ],
   "lsFiles":[
        {
            "lsFiles/dsFile":null,
            "_nuSequence":0
        }
    ],

    "webSiteLine":[
     {
           "webSiteLine/dsWebsite":"www.tenaris.com",
           "webSiteLine/tpWebsite":"1"
     }
   ],
    "phoneLine":[
      {
           "phoneLine/tpPhone":"1",
           "phoneLine/nrPhone":"01137489236"
      }
   ],
    "eMailLine":[
      {
           "eMailLine/tpEmail":"1",
           "eMailLine/dsEmail":"info@tenaris.com"
      }
    ]
}
```

3.6.1.3. PUT/PATCH Method

PUT and PATCH methods are used to update a resource. The difference between them is that the PUT method impacts all the attributes of the resource and the PATCH method impacts one or more of those attributes.

- They use the HTTP PUT and PATCH verbs respectively.
- The url contains the collection name and id of the resource to be updated.
- The new values of the resource attributes are specified in the request body, that is, the configuration declares a "body" clause.

In the PUT method, all the values for all resource attributes must be specified. On the other hand, in the PATCH method, only the values of the attributes that are to be updated must be specified.

- There is no response body.
- The modification date (dtLastUpdate) obtained with a query (GET) to the corresponding instance prior to making the modification must be specified.

Example: Instance update with Id = 93 from the "Account" object.

```
PUT /forms/CRM_ACCOUNT/instances/93
```

The request body ("body" clause) contains:

```
{
  "cbIsPartner": false,
  "cdInstanceOwner": "JPerez",
  "cdStatus": "Active",
  "cdType": "1",
  "dsCompany": "Tenaris",
  "dsDescription": "",
  "dsHolding": "TECHINT",
  "dsIndustry": "4",
  "dsOwner": "Juan Perez",
  "dsSource": "Referred",
  "dtLastUpdate":"1575059822000",
  "dtOpening": "1540782000000",
  "emails":"",
  "flLogo": null,
  "idPartnerCompany": null,
  "nuIdentifNumber": "30-00000000-5",
  "qtAnualRev": "1500000000",
  "qtEmployee": 1000,
  "qtScore": 90,
  "lsAddress": [
    {
      "lsAddress/dsAddress": "Della Paolera 333",
      "lsAddress/dsCity": "buenos aires",
      "lsAddress/dsCountry": "argentina",
      "lsAddress/dsState": "buenos aires",
      "lsAddress/tpAddress": "1"
    }
 ],
  "lsFiles": [
    {
      "lsFiles/dsFile": null,
```

```
      "_nuSequence": 0
    }
 ],
  "webSiteLine": [
    {
      "webSiteLine/dsWebsite": "www.tenaris.com",
      "webSiteLine/tpWebsite": "1"
    }
  ],
  "phoneLine": [
    {
      "phoneLine/tpPhone": "1",
      "phoneLine/nrPhone": "01137489236"
    }
  ],
  "eMailLine": [
    {
      "eMailLine/tpEmail": "1",
      "eMailLine/dsEmail": "info@tenaris.com"
    }
  ]
}
```

```
PATCH /forms/CRMACCOUNT/instances/93
```

The request body ("body" clause) contains:

```
{
  "dsDescription": "Steel company",
  "dsHolding": "TECHINT Co",
  "dsIndustry": "4" ,
  "dtLastUpdate":"1687549970000"
}
```

3.6.1.4. DELETE method

The DELETE method is used to delete a resource contained within the operation url.

- It uses the HTTP DELETE verb.
- The url contains the collection name and id of the resource to be deleted.
- There is no request body.
- There is no response body.

Example: Deletion of instance with Id = 93 from the "Account" object.

```
DELETE /forms/CRM_ACCOUNT/instances/93
```

3.6.1.5. Return Codes

All methods are implemented in such a way that the response contains not only the information returned by the request in the corresponding cases, but the return code that indicates the result, successful or not, of the operation performed.

If the result of the operation is not successful, in addition to the corresponding return code, a collection is returned with all the errors of **Deyel** that have occurred.

Although each method implements its own return codes for its operation, all possible HTTP return codes are described in the following table.

HTTP response code	Description
200	“OK” success code, for GET or HEAD request.
201	“Created” success code, for POST request.
202	"Updated" success code, for PUT request.
204	“No Content” success code, for DELETE request.
400	The request couldn't be understood, usually because the JSON or XML body contains an error.
401	The session ID or OAuth token used has expired or is invalid. The response body contains the message and errorCode.
403	The request has been refused. Verify that the logged-in user has appropriate permissions. If the error code is REQUEST_LIMIT_EXCEEDED, you've exceeded API request limits in your org.
404	The requested resource couldn't be found. Check the URI for errors, and verify that there are no sharing issues.
405	The method specified in the Request-Line isn't allowed for the resource specified in the URI.
415	The entity in the request is in a format that's not

HTTP response code	Description
	supported by the specified method.
500	An error has occurred within Deyel Platform, so the request could not be completed. Contact Deyel Customer Support.
503	Service unavailable. Contact Deyel Customer Support.

3.6.1.6. Data Type

Data types available in **Deyel** are detailed, with its equivalent representation in the JSON schemas used by the API.

	Data type	Type in JSON	Example
Text	Alpha-numeric (length)	String	"value"
	Alpha-numeric Uppercase (length)		
	Large Alpha-numeric		
	Enriched Text		
Integer	Number	Number	10
	Large Integer	String	"10"
	Decimal		"10.5"

	Data type	Type in JSON	Example
Time	Time	String (milliseconds from epoch)	"946748150000"
	Local time		
Date	Date	String (milliseconds from epoch)	"1606499064998"
	Date and Time		
	Local Date		
	Date and Local Time		
Image	Image in Database	String	"&cdRepositorio=IDB&dsFilename=2.....pg&dsOriginalFilename=photo.jpg"
File	File in Database	String	"&cdRepositorio=IDB&dsFilename=2.....pg&dsOriginalFilename=photo.jpg"
Check	Check	Boolean	true / false

3.6.1.7. Swagger Reference Guide

Within the documentation tools available on the market, **Deyel** uses Swagger, a collaborative platform that has established itself as a benchmark.

All endpoints available in the API are documented in the reference guide, along with the definition and explanation of their parameters, request schemas and data structure of each of them.

This is an interactive documentation that allows to explore and test API calls from a browser, showing parameters, headers, request schemas, and response body, as well as the duration of the request and the URL command that can be used to send the same request from the command line.

3.6.1.8. Security

Deyel implements the OAuth 2.0 standard to interact with its Rest API using a Bearer access token.

It is important to understand the concepts of authentication and authorization.

Authentication is defined as the process to verify who is connecting, that is, it refers to identification.

For example, when a user logs into an application by entering a username and password, the application authenticates that user. That is, the application verifies that the entered password matches the one associated with the user. As long as the password matches, the user can access the application. But if it doesn't match, the user is not allowed to log in because their ID could not be confirmed.

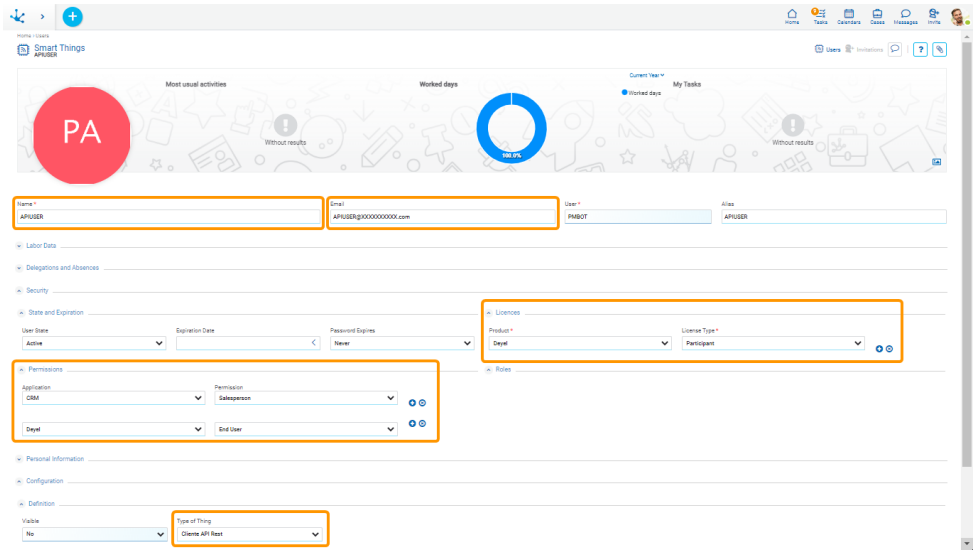
Authorization is defined as the process that verifies what the user has access to.

For example, in an application that offers different functionalities for free users and fee-paying users, free users are authorized to access only a limited set of functionalities until they pay the subscription and become users with better functionalities.

Registration of External Applications

In **Deyel**, applications that try to connect from an external environment are represented by a thing type user "Rest API Client". This means that in order to consume the resources exposed in Deyel Rest API, it is necessary to send the credentials of that user in order to authenticate and generate the corresponding access token, which must then be used for consumption by the different endpoints.

To configure a smart thing type user, it should be created from the "Configuration" option of the **Deyel** menu. A smart thing type user must be defined, configuring its properties to indicate name, user, type and permissions for each application.



The smart thing type user properties should be completed with the suggested values.

Name	APIUSER
User	APIUSER
Email	Email address where the user receives the password.
Type	Rest API Client
Product Type of License	CRM Participant
Product Type of License	Deyel Participant
Application Permission	CRM Salesperson
Application Permission	Deyel End User

Access Token Request Endpoint

To request the new access tokens and to access the different endpoints, it is necessary to first access the endpoint:

https://<ambiente>/oauth/token

To request the access token, the following configuration should be used:

- POST verb.
- Add parameters using the format:
"application/x-www-form-urlencoded".
- Access credentials ("client_id" and "client_secret") must be sent in “Basic xxxxxx” format where the xxxxxx represent Base64 encoded values.
- The parameters to be sent vary depending on the authorization flow, but the "grant_type" parameter must always be sent.

Deyel has an OAuth 2.0 module that is implemented internally and serves to perform user authorization and generates as a result a JWT token used to authorize.

Expiration and Renewal of Tokens

The generated access tokens are valid for 1 day, while the refresh tokens last for 14 days, users are asked to identify themselves again at the end of the period.

Information Stored in Tokens

The generated access token registers claims in the JWT body (payload): user code, organizational unit of the user and user permissions.

3.6.1.8.1. OAuth 2.0

OAuth 2.0 is an authorization protocol. It allows a third-party application to have access to an HTTP service, either on behalf of a resource owner, by arranging an approval interaction between the resource owner and the HTTP service, or by allowing the third-party application to have access on its own name.

OAuth 2.0 focuses on:

- Providing authorization flows to limit access to a service. Authorization flows are intended to be used by web applications, desktop applications, mobile phones, and room (bedroom, living room, etc) devices.

- Providing scopes to specify what access is requested by the application and what is authorized to be accessed by the owner of the accessed resource.

Authorization Flow (Flow Grant Types)

OAuth flows (grant types) refer to how an application obtains an access token that allows it to access resources exposed through an API. The standard uses these flows to solve all business scenarios that may arise in the consumption of APIs based on three variables:

- The type of consuming application.
- Its degree of confidence.
- How the resource owner interacts in the process.

Authorization Flows Available in Deyel

In **Deyel,** the following authorization flows are available:

- Client Credentials
  This flow is used to consume the objects exposed in the Rest API. The client can request an access token using only their client credentials when requesting access to protected resources under their control.

- Resource Owner Password
  Like the previous flow, it is used to consume the objects exposed in the Rest API. The flow is suitable for clients capable of obtaining the resource owner's credentials (username and password, typically via an interactive form) and working on their behalf with their protected resources.

- Refresh Token
  It is used to refresh access tokens. This means that those access tokens that expire or become invalid should be exchanged for a new one using this flow. It is used in authorizations with the Resource Owner Password flow.

3.6.1.8.1.1. Client Credentials

To make calls, the following configuration should be used:

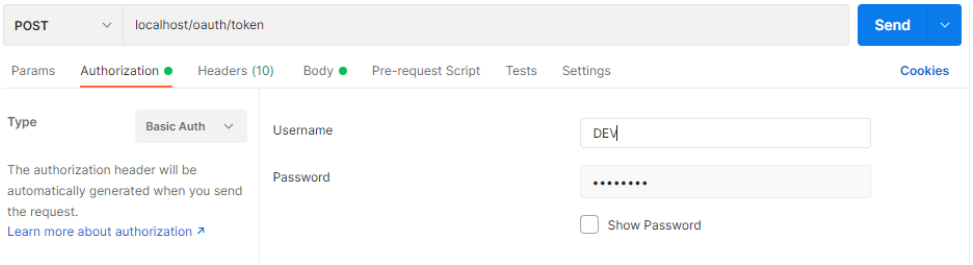
- POST verb.
- Add parameters using the format:

"application/x-www-form-urlencoded".

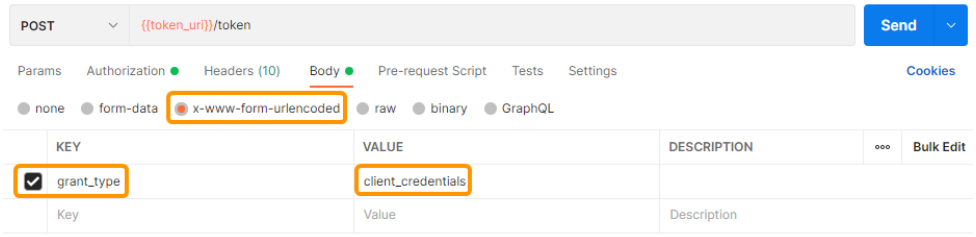
- Access credentials ("client_id" and "client_secret") must be sent in “Basic xxxxxx” format where the xxxxxx represent Base64 encoded values.
- Specify the "grant_type" parameter with “client_credentials” value.

Example of Use in Postman

- Specification of authorization with “Rest API thing” user keys.



- Specification of grant_type and parameter sending format.



Successful response of access token request:

```
{

"access_token": "eyJraWQiOiJrMSI-
sImFsZyI6IlJTMjU2In0.eyJpc3MiOiJEZXllbCIsImV4cCI6MTY1NDI2MTQ5MCwiY2R
PcmdVbml0IjoiMDAwMDAwMDAwMSIsImdyYW50X3R5cGUiOiJjbGllbnRfY3JlZGVudGl
hbHMiLCJwZXJ-
taXNzaW9ucyI6IntcImR0VmFsaWRpdHlcIjpcIkp1biAyLCAyMDIyLCAxMDowNDo1MCB
BTVwiLFwibHNQZXJ-
taXNzaW9uc1wiOlt7XCJpZEFwcGxpY2F0aW9uXCI6XCJERVlFTFwiLFwiY2RQZXJtaXN
```

```
zaW9uXCI6XCJVU1VBUklPRklOQUxcIn1dfSIsImNvbXBhbnkiOiJDU1QiLCJudVNlY0h
pZGRl-
biI6IjIwMjIwNjAyMTAwNDUwLjg2NjAuODA1Mjc1NzQ3NDc1NzUxOSIsImlkRGF0YUJh
c2VTZXNzaW9uIjoiREVZRUxCT1QqMjAyMjA2MDIxMDA0NTAuODY2MC44MDUyNzU3NDc0
NzU3NTE5IiwianRpIjoieDN1UnhXM09xbVZvMlBWdmsxWGZmdyIsImlhdCI6MTY1NDE3
NTA5MCwic3ViIjoiREVZRUxCT1QifQ.M8EHKFhMwJQsdJbg4VQkPcvlJxoRyG-
XO6h4eVWyMtkyvcCxN658hCqg8luu2yM5rEVMpUqOhNWKAoJdcvOvCMpAi21oYttkVlQ
xSdE-
NUVNu4fmhDavQshstzvqGrY3qu9gaxdCfcLjwf62bEf9227TpvDGbZyFP8jYLbA81gwo
1y6Q2NOaIE_yrChMB4LRm-bhEx5epdL1lMsE3JuwjMQ0tN2j-
EehA1dDPB3AY8461wTGOPULF66B81iGmnOvexvoeS0vZQU3cMXwpxy-yQ8ulJRIdj-
BprxmkFWkc3_MyD7SgkyqQwVSjiXDgfbJoFkCTPYHprZVDE362yBYW",
    "token_type": "Bearer",
    "expires_in": 86400
}
```


Failed response of access token request:

```
{
    "error_description": "Invalid client_id",
    "error": "Invalid_request"
}
```

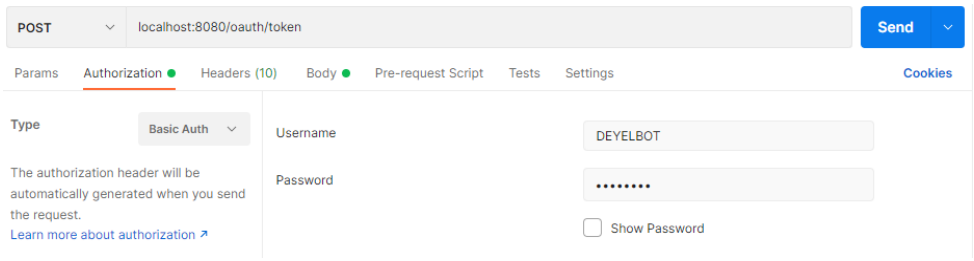
3.6.1.8.1.2. Resource Owner Password Credentials

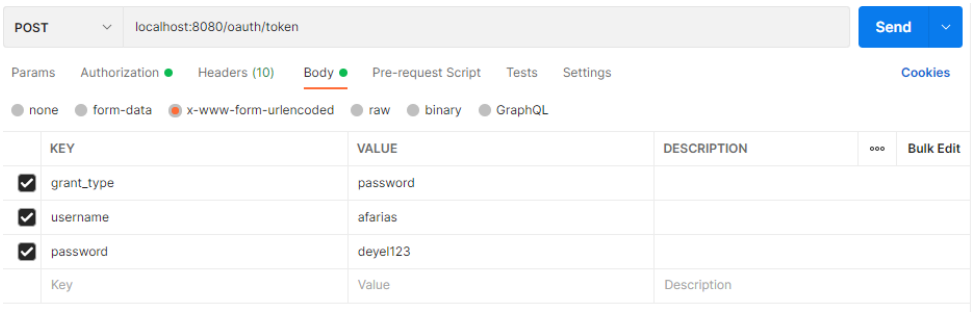
To make calls, the following configuration should be used:

- POST verb.
- Add parameters using the format:
"application/x-www-form-urlencoded".
- Access credentials ("client_id" and "client_secret") must be sent in “Basic xxxxxx” format where the xxxxxx represent Base64 encoded values.
- Specify the "grant_type" parameter with "password" value.
- Specify the "username" parameter with the value of the resource owner's username.
- Specify the "password" parameter with the value of the resource owner’s password.

Usage Example in Postman

The following is an example of an access token request from Postman.





The client makes the following HTTP request:

```
POST /oauth/token HTTP/1.1
Host: localhost:8080
Authorization: Basic REVZRUxCT1Q6Qk9UREVZRUwzMTIh
Content-Type: application/x-www-form-urlencoded
Content-Length: 54
grant_type=password&username=afarias&password=deyel123
```

If the access token request is valid and authorized, the authorization server issues an access token and a refresh token. If the client request failed authentication or is invalid, the authorization server returns an error response, such as for submitting a non-existent user.

Successful access token response:

```
{
    "access_token": "eyJraWQiOiJrMSI-
sImFsZyI6IlJTMjU2I….ezCK4IWR7YU1NlMAuCFKE7mJw",
    "refresh_token": "eyJraWQiOiJrMSI-
sImFsZyI6IlJTMjU….cL_mqernCGkQ",
    "token_type": "Bearer",
    "expires_in": 86400
}
```

Failed answer:

```
{
    "error_description": "Invalid client_id",
    "error": "Invalid_request"
```

```
}
```

3.6.1.8.1.3. Refresh Token

In the event that a refresh token or access token is denied by an HTTP 401 code, the access token request endpoint must be invoked with the following configuration:

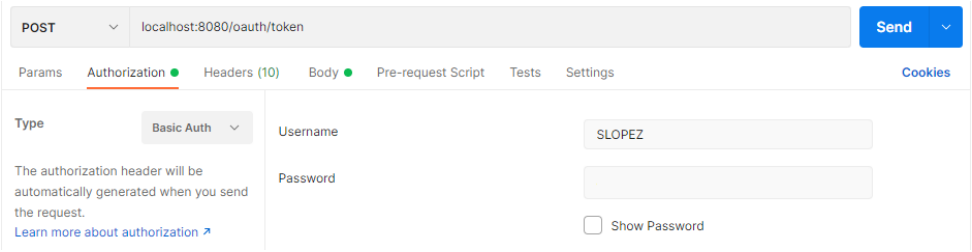
- POST verb.
- Add parameters using the format:

"application/x-www-form-urlencoded".

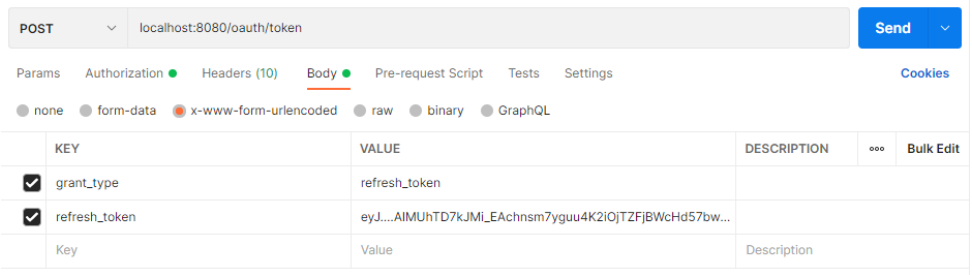
- The "client_id" access credential must be sent in “Basic xxxxxx” format where the xxxxxx represent Base64 encoded values.
- Specify the "grant_type" parameter with "refresh_token" value.
- Specify the "refresh_token" parameter with the value of the refresh token obtained when authenticating with **Deyel**.

Usage Example in Postman

- Authorization specification with user code.



- Specification of "grant_type", parameter sending format and refresh token value.



As a response, a new access token and the same refresh_token value are obtained.

Example of successful access token request:

```
{

"access_token":
```

```
"eyJ.AIMUhTD7kJMi_EAchnsm7yguu4K2iOjTZFjBWcHd57bwcTeghfhfwh_aIYJhGOd
CDwyIrkpmF-
tOOeW2VPG_9UGn2kRmEgLOAjqYJWXrVyjhs877k_cITkcvclAQnCh5TEkmhmE7IKgW1a
K53XSP_M4xIOnxCugq4Qvi-2cWI152BWhLRkICgJ4Q",

"refresh_token": "eyJraWQiO….4a4xGrq_6137hbsI53AgcPpbDAL6qC1-
YfvuzOKYxtwqhieCQSflY9DnTDDVa-
0Yy5lxeCfVGt1ra8pxLmhkzz_iaqoabqXprZIejKY9FJIhC0paMTV_w06YSgUh5XI0p3
YayrLTjik1YayrLTjik1YayrLT3",
      "token_type": "Bearer",
      "expires_in": 86400
}
```


Example of failed access token request:


```
{
    "errors": [
        {
            "message": "Unauthorized",
            "attribute-name": ""
        },
        {
            "message": "Invalid refresh token.",
            "attribute-name": "refresh_token"
        }
    ]
}
```


3.7. Business Social Network - Tedis

Tedis is the business social network used by **Deyel**, designed to improve communication and collaboration within the organization. It allows for fast, intuitive messaging integrated with **Deyel** applications.

Main Features

- **Chats**: They allow the exchange of instant notifications, streamlining user communication.
- **Conversations associated with objects**: They allow to leave messages in chats related to different **Deyel** objects.

Comments on cases: Comments on a specific case can be recorded, with this information remaining as part of the case and visible to all participants. This facilitates collaboration and makes processes more efficient. This mechanism allows, for example, making queries to other users without the need to hand the case over to them for formal intervention.

Comments on processes: They facilitate communication between the participants of a case and those responsible for such processes, allowing for suggestions and improvements.

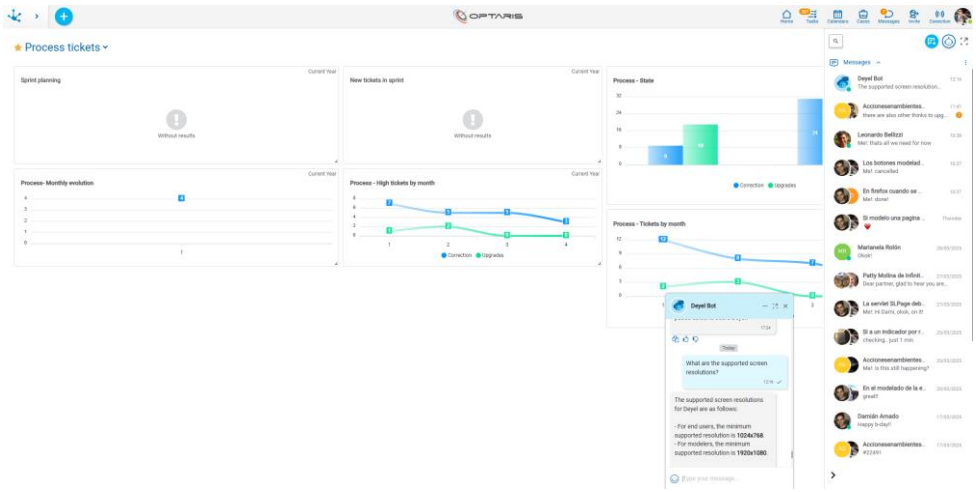
Comments on entities and forms: They allow users to interact, collaborate and discuss each instance. It is also possible to show conversations related to associated cases, with the aim of tracking entities or forms and understanding user interactions.

- **Subscriptions**: Through the subscriptions, each user may choose to receive news about comments related to the object of interest. This allows quick tracking of new comments without having to show a particular case, entity or form.

- **Commands**: It is possible to start processes from the chat using commands. To do this, all you have to do is define a process with a start event by command, and then associate the process command with one or more bots.

3.7.1. Chats

The chat bar is located on the far right side of the portal and provides quick access to all conversations within the Tedis business social network.



Chat Bar

The chat bar provides quick access to active conversations within the portal. It can be expanded or collapsed by clicking on the messages icon located at the top toolbar. The number above this icon indicates the user's total number of unread messages.

Header

When the bar is expanded, the following icons are displayed at the top:

 Search: It allows searching for messages by title or content.

 New message: It opens a window to select one or more recipients. If a single user is selected and a previous conversation already exists, it is resumed; otherwise, a new one is created. In case of multiple recipients, a group chat is automatically generated. By clicking “Create Chat”, the conversation window opens.

 Chatbot: It starts a conversation with Deyel Bot.

 Full screen: It opens the message panel in a new browser tab, where the chat bar is displayed on the left and the active message is on the right.

When the bar is reduced, the same icons are arranged vertically, one below the other.

Body

Below the header, the expanded bar displays two controls:

 Message view: It allows showing or hiding the container that expands the conversations.

 Filtered: It opens a menu with options to filter through pending messages or view all messages.

When the message container is expanded, the recent conversations can be seen. Each conversation displays the user's or group's image, the connection state (for individual users), the conversation title, and a snippet of the latest message.

The reduced bar only displays the user’s or group's image and, if applicable, their online state.

The possible connection states are: connected, unavailable, absent or disconnected, and they can be modified from the user profile.

Clicking on any of the messages automatically opens a chat window on the left side of the bar.

Foot

At the bottom of the bar are the icons ‹ and ›, they allow expanding, reducing or hiding the chat bar.

Chat Window

The chat window allows users to send and receive messages in conversations. Each conversation opens in a separate window when selected from the chat bar or when creating a new one.

Header

At the top of the chat window, the following is displayed:

 User profile or group image.

Title: Clicking on the conversation title allows expanding, collapsing, or minimizing the chat window. In group conversations, the title can be modified.

 It includes additional actions such as showing participants, going to the related object, modifying the title, adding users, unsubscribing, expanding or collapsing the window, and deleting the history. The available options may vary depending on whether it is an individual or group conversation.

 It opens the conversation in a new browser tab.

 It closes the chat window.

In group conversations, a legend is also displayed with the word "Participants" followed by the total number. Clicking on it displays images of the group members.

Message

Each message sent within the conversation displays a state icon to the right indicating its status:

 Sending

 Sent

 Read

In group conversations, the "Read" state" appears only when all participants have viewed the message.

In the conversation window, there are messages written by users and also attachments and messages from **Deyel**, such as when a user joins or leaves a group conversation, or when the development of an application is completed.

Messages are organized chronologically and separated by date, which facilitates their reading and tracking.

Foot

At the bottom of the window is the area to perform the following actions:

Write a message: It allows typing directly into the text field.

Insert emoticons: They can be selected from the palette that expands from the corresponding icon or write directly with symbols.

Attach files: Up to 6 files can be attached per message, except those with the .exe or .msi extension.

There are two ways to attach files:

- Drag and drop files directly into the window: the message is sent automatically once the upload is complete.
- Select files from the attachment icon: files must be uploaded first and then sent, with or without text. Attachments may be removed before sending.

Images include a preview, and if an error occurs during upload, an option to retry sending is offered.

The maximum size in MB for an attachment depends on the value of the Maximum size stored in database variable that is defined for the environment.

Commands: It allows running previously defined commands. A command is a specific instruction defined for a user or for a chatbot, initiating the execution of a specific process. To do this, the process must have been modeled with a command start event.

Chat Preferences

From the environment configuration, the user can customize their experience with Tedis. The initial size of chat windows can be set, as well as enable or disable notifications for new messages or when other users connect. User visibility can also be configured based on their organizational unit or by application, facilitating access to relevant contacts within the work environment.

Guest User Permission

The permission "Guest in Tedis", which determines the actions that a guest user can perform in the business social network.

A guest user can only reply to chats where they have previously received messages. Actions such as viewing the user list, creating new chats, adding or removing users in group chats, and modifying the title of a group chat are restricted.

> *If the guest user deletes the chat history, it cannot be accessed until they receive a new message.*

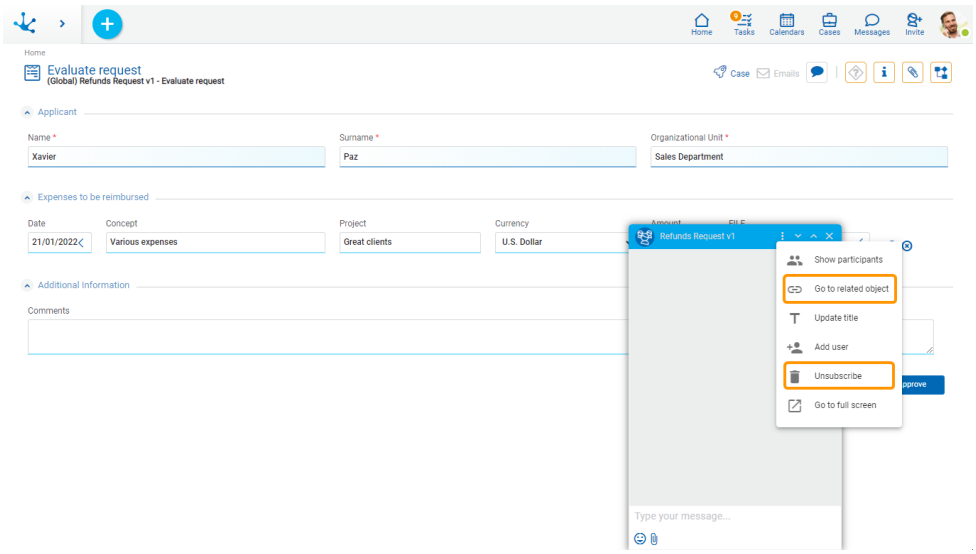
3.7.2. Conversations Associated with Objects

When a user is running a case or updating a form instance, they may need to make a query to other users. This communication and exchange of additional information usually takes place without using applications, for example through phone calls, emails, etc. **Deyel** suggests to use a chat during the execution of processes cases and during the use of forms instances, in order to associate comments with the application.

These conversations have an additional behavior to the group conversations.

- Go to related object
  It allows the user to show directly the object related to the conversation.

- Subscribe / Unsubscribe
  A user that can show an object may participate in the comments on it. Those users that wish to be informed about any news added to the comments, can use the subscription function.
  A user that has subscribed to the comments of a case or form is notified every time a new message is entered in such chat. Besides, other users can be added to the chat so that they can also participate and be notified of new comments. A subscribed user may also leave the chat if they no longer want to receive notifications.

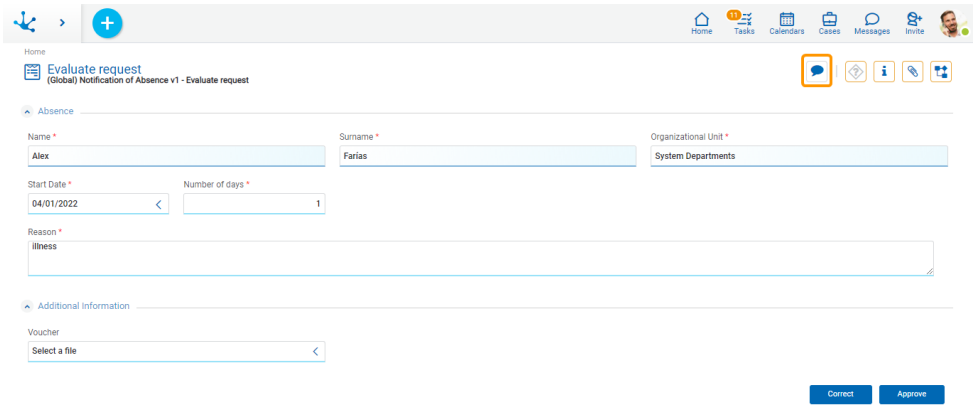


Comments Associated with Cases

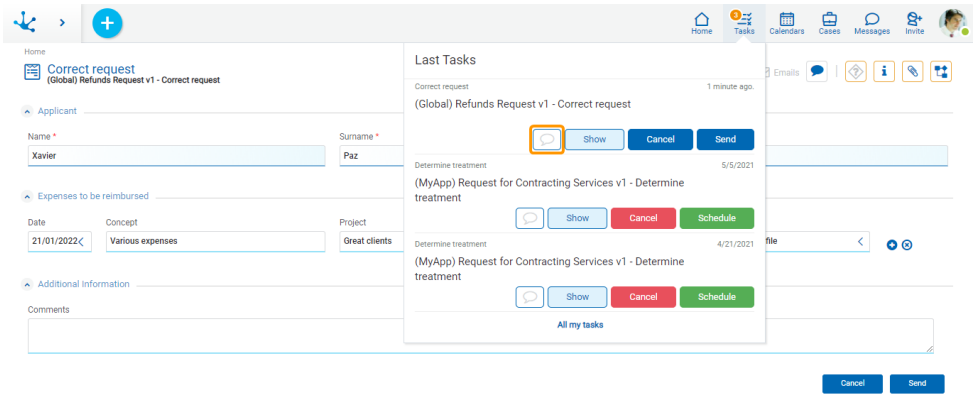
Having a case-specific conversation simplifies process modeling, as countless situations can be solved through user interaction rather than process modeling, as well as feeding the case with information used for decision making.

How to access comments associated with a case:

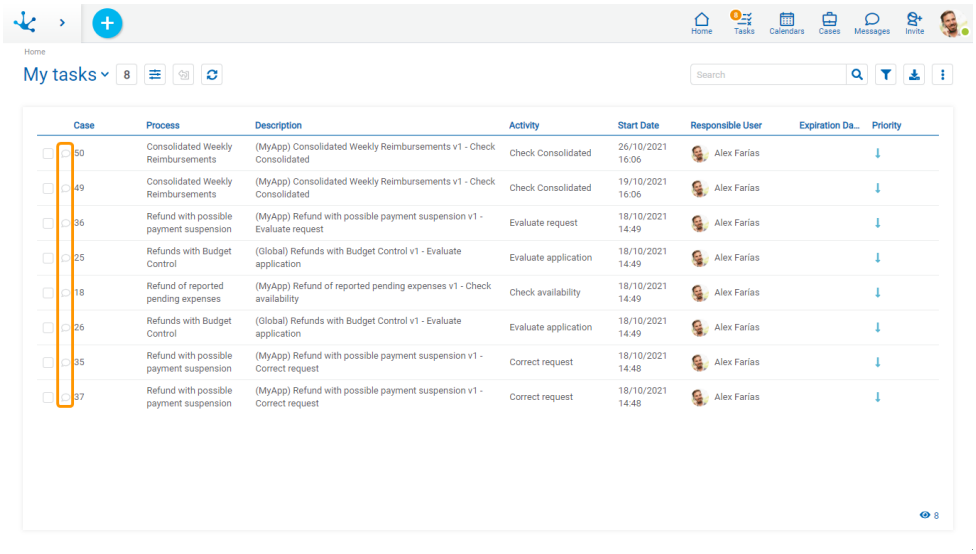
- Show of the case, selecting the "Case" option.



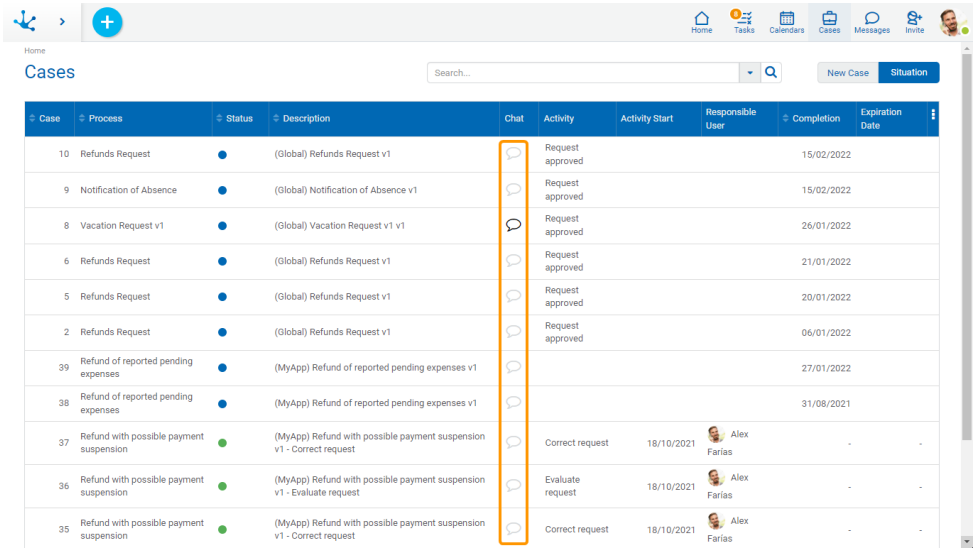
- From the last tasks panel.



- From the tasks grid.

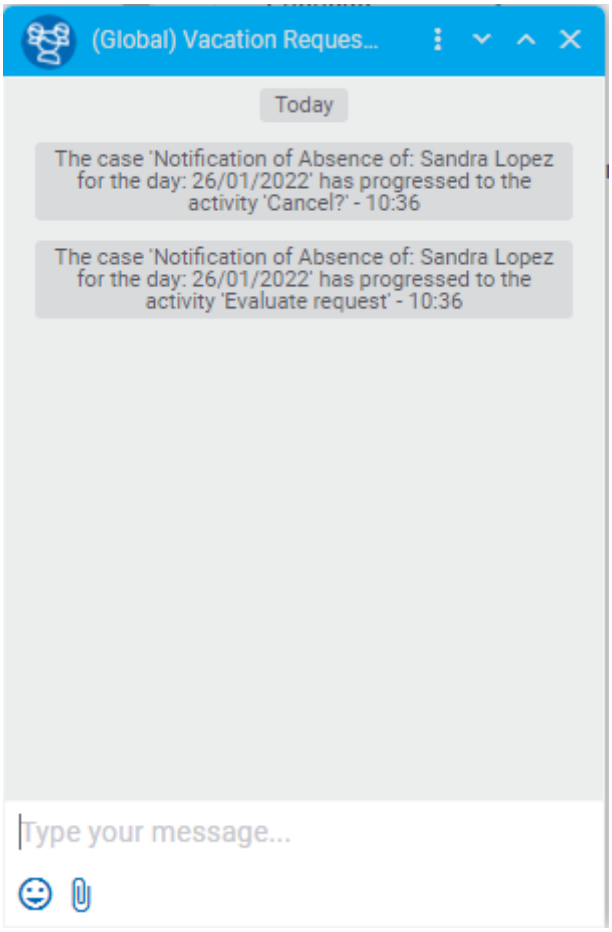


- From the cases grid



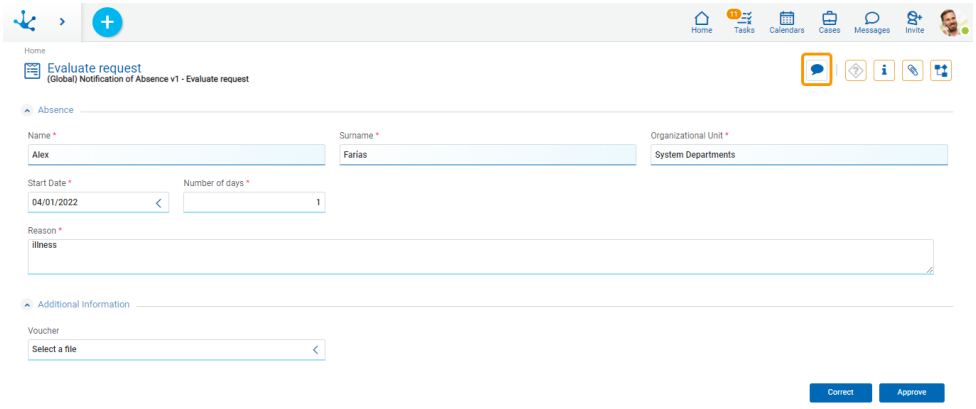
Events Generation

As the case progresses within the activities, events are generated informing of such progress.



Comments Associated with Cases

Within the continuous process improvement cycle, it is important for the responsible user or the owner of the process to be in contact with its users. For this, there is an exclusive chat for each process where its users can record messages related to modeling, thus allowing the modeling responsible users to receive these comments.

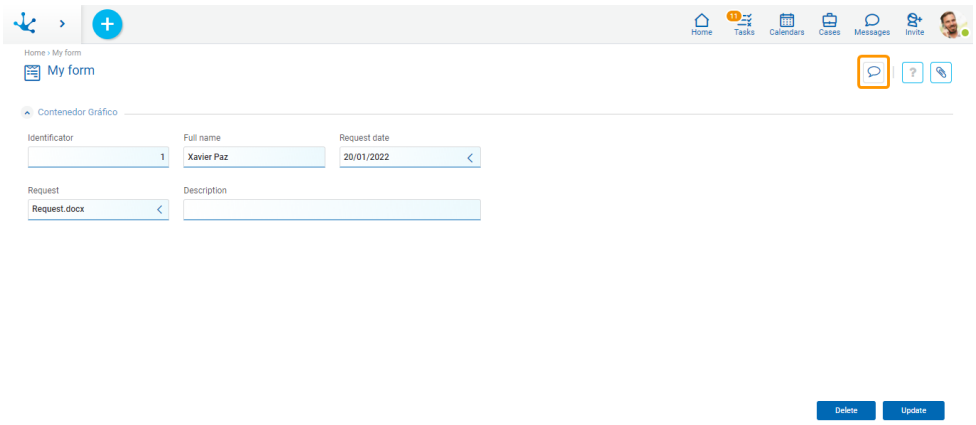


Comments Associated with Forms

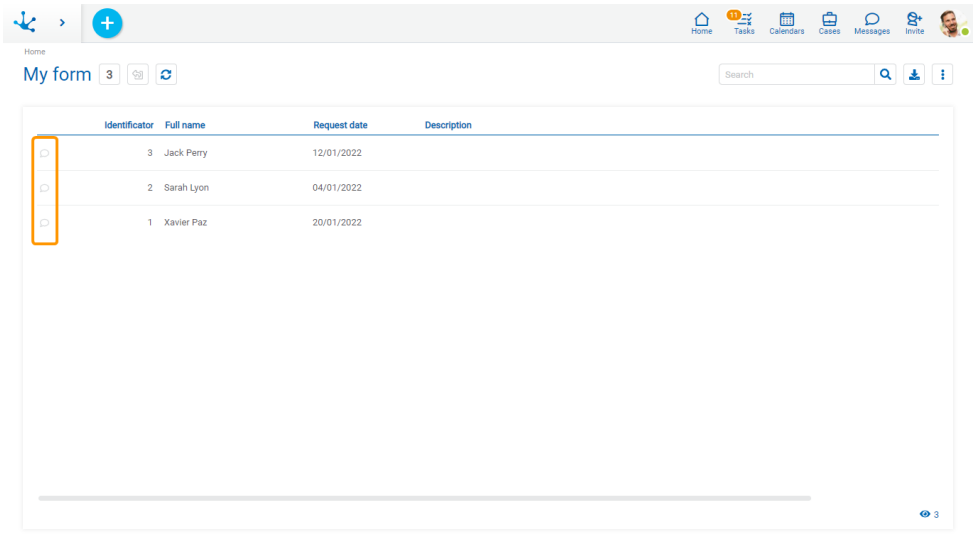
Deyel allows to create conversations associated with forms instances, so that users who use them can interact and collaborate on specific topics of the form. Users who have permissions on the form may subscribe to the conversation.

How to access comments associated with a form instance:

- Show of the form instance.



- From the grid of form instances.



3.8. BAM and Process Analysis

Phase 7: BAM

Deyel has a set of predefined reports that allow analyzing the operation of processes and their tasks, both historically (Analysis) and in real time (BAM). The information in these reports is updated through the scheduled system tasks.

Through these reports, it is possible to observe and understand the operation of processes and detect possible improvements.

Predefined reports summarize the behavior of processes and their tasks. They allow identifying what was done on time, behind schedule, deviations against maximum and expected durations, trends, participants performance, bottlenecks, etc.

The behavioral information is displayed in the form of a grid and through charts, always having the reference of the expected trend curves to contrast against reality. In this way, behavior can be graphically monitored. In all cases, the information can be drilled down, to reach the process level, task, executor, date range, case, etc.

Although the reports are predefined, multiple views of the information can be generated, when analyzing it by applying search criteria, such as time, priorities, participants, initiators, cases states and tasks, among others.

In the case of real time analysis, in addition to identifying the cases and tasks that are on time and behind schedule, those that are at risk are identified, providing tools to proceed on what is actually happening in the organization.

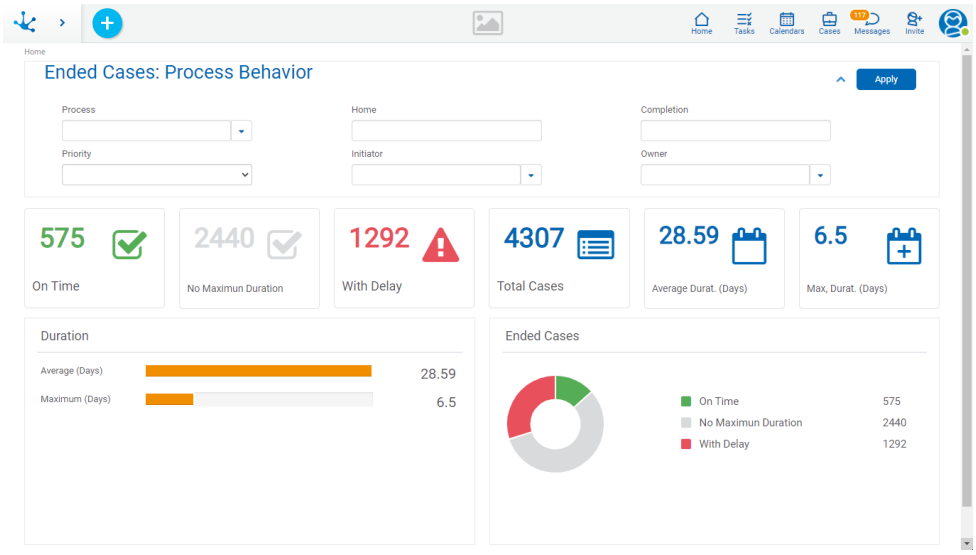
In addition to the predefined reports, user-defined reports can be incorporated based on existing or supplement them with specific characteristics of the organization.
Reports are accessed through the different menu options that are displayed by pressing "BAM and Analysis".

3.8.1. Process Analysis - Process Behavior

Phase 7: BAM > Process analysis

In this report, the behavior of processes is quickly displayed, indicating the cases that were completed on time, those behind schedule and the real average duration compared to the maximum duration.

The information can be displayed by applying the search filters shown in the upper section of the report.

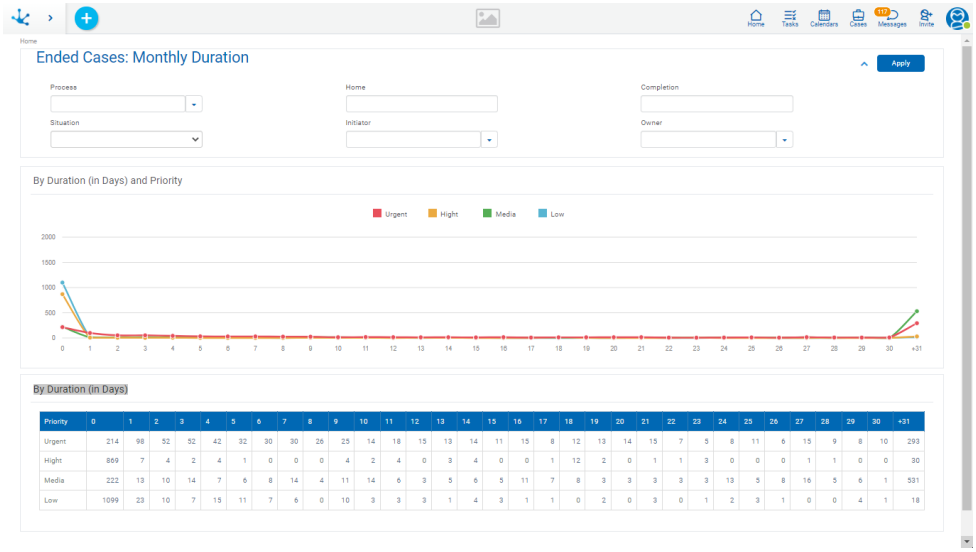


3.8.2. Process Analysis - Monthly Duration

Phase 7: BAM > Process analysis

The report on distribution of cases by duration must show a chart similar to a Gaussian bell, where the center of the bell is the average duration, provided that the behavior of the processes is adequate.
To analyze behavior in detail, drill down on the values of the grid in order to reach the cases that make up each value.

The information can be displayed by applying the search filters shown in the upper section of the report.



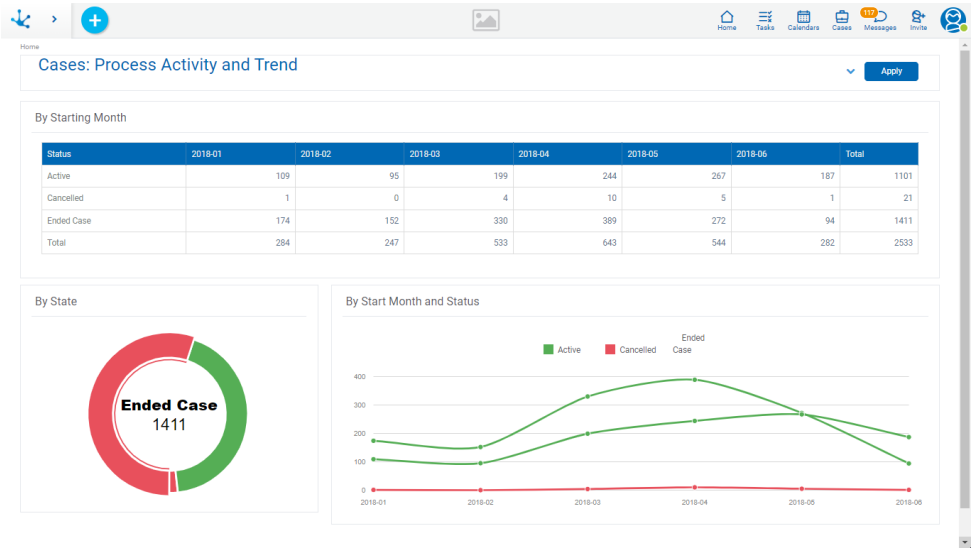
3.8.3. Process Analysis - Process Activity

Phase 7: BAM > Process analysis

In this report, the activity carried out by the processes is displayed, showing the values of cases initiated by periods, how many were completed, how many remain in execution, and the cases suspended and canceled.

If the information on process activity for the last months is analyzed, it should be observed that as months go by, the cases in execution tend to increase, and the cases already completed tend to decrease. In other words, during the first months, a greater number of completed cases should be observed, while during the last months, there are more cases in execution. Any anomaly, that is to say that the chart shows a different trend, can be investigated by drilling down on the values.

The information can be displayed by applying the search filters shown in the upper section of the report.

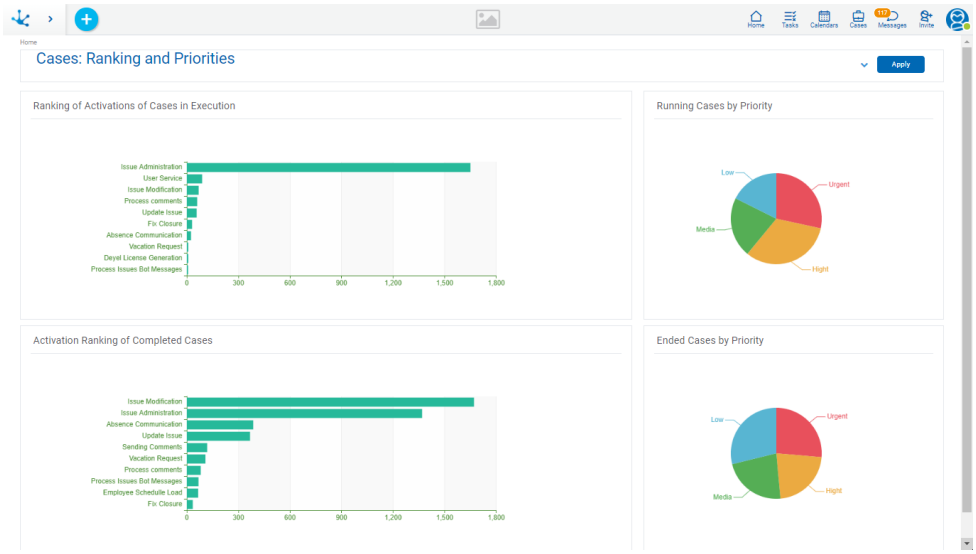


3.8.4. Process Analysis - Activation Ranking

Phase 7: BAM > Process analysis

This report shows a ranking of processes, through the number of cases initiated for each one, also considering their execution priority.

The information can be displayed by applying the search filters shown in the upper section of the report.

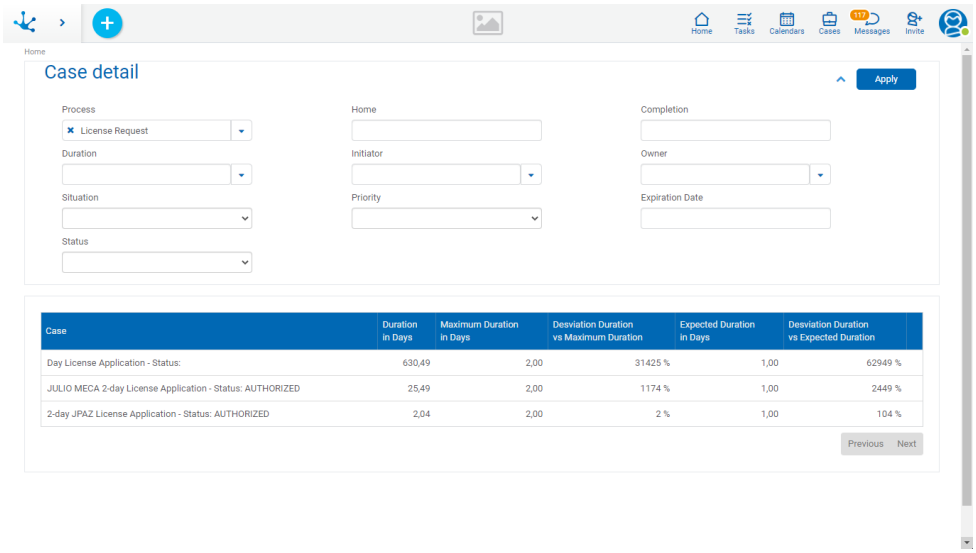


3.8.5. Process Analysis - Case Details

Phase 7: BAM > Process analysis

In this report, cases with their actual, maximum and expected duration can be displayed, together with comparisons of duration with maximum duration and duration with expected duration.

The information can be displayed by applying the search filters shown in the upper section of the report.

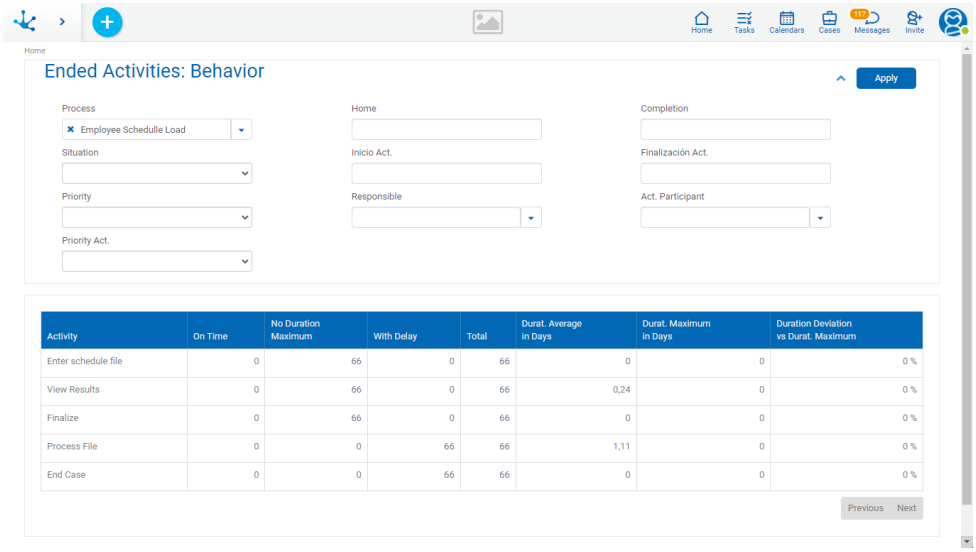


3.8.6. Activity Analysis - Activity Behavior

Phase 7: Activity BAM - Activity analysis

In this report, the behavior of a process completed activities can be analyzed (including or not all its versions). The amount of activities on time and behind schedule is detailed for each of them. Additionally, the average duration, the maximum duration and the deviation between both of them is mentioned.

The information can be displayed by applying the search filters shown in the upper section of the report.

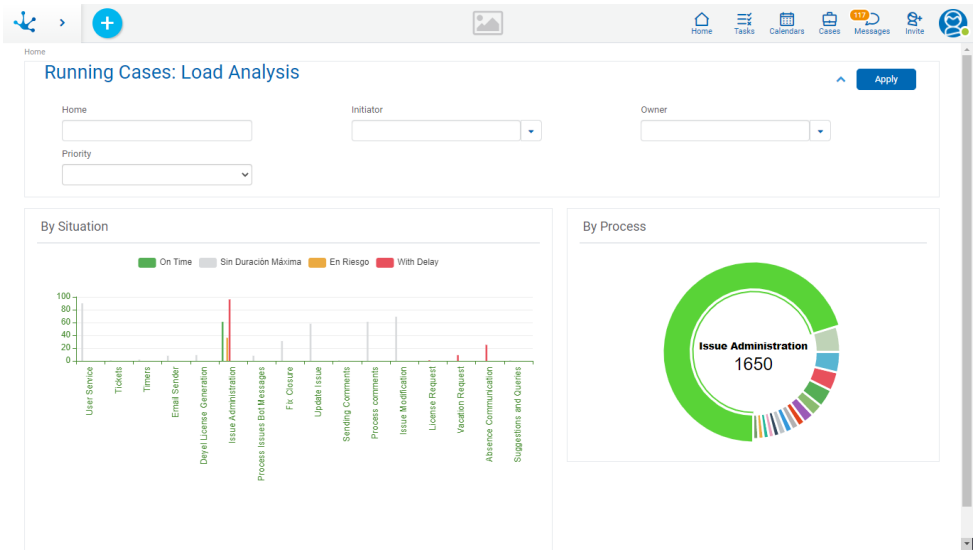


3.8.7. Process BAM - Load Analysis

Phase 7: BAM > Process monitoring

In this report, the current workload can be analyzed, considering the distribution of cases by process. For each process, it also shows how many cases are on time, how many at risk and how many are behind schedule.

The information can be displayed by applying the search filters shown in the upper section of the report.



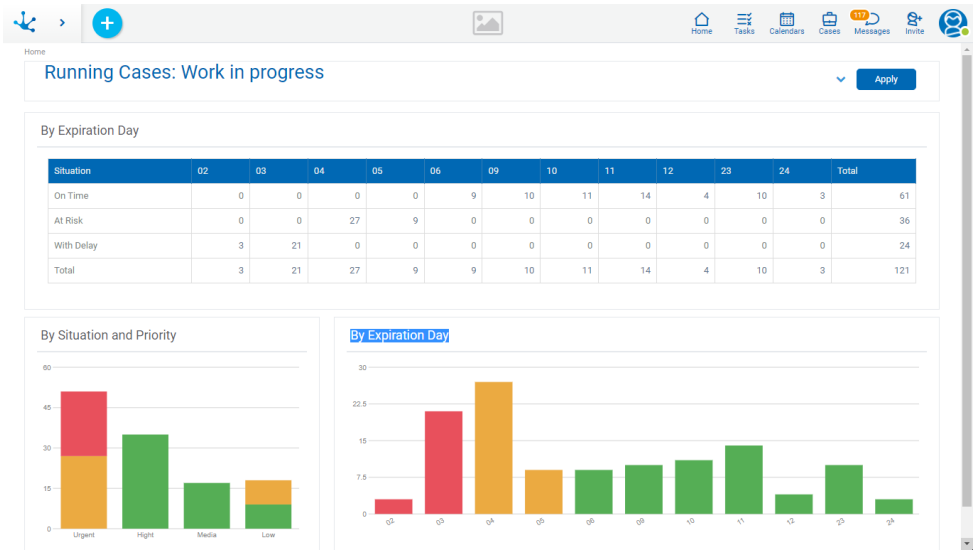
3.8.8. Process BAM - Work in Progress

Phase 7 > Process monitoring

This report shows the status of the cases that are in execution, together with their distribution according to their expected due date in periods of time.
Regarding the status of the cases, it is graphically displayed how many are on time, how many at risk and how many are behind schedule.

Regarding the distribution of cases according to due date, the chart indicates how many are scheduled to expire in the time period indicated, and it may or may not count those that are behind schedule. If those behind schedule are counted, they are shown in the first period considered in the chart.

The information can be displayed by applying the search filters shown in the upper section of the report.

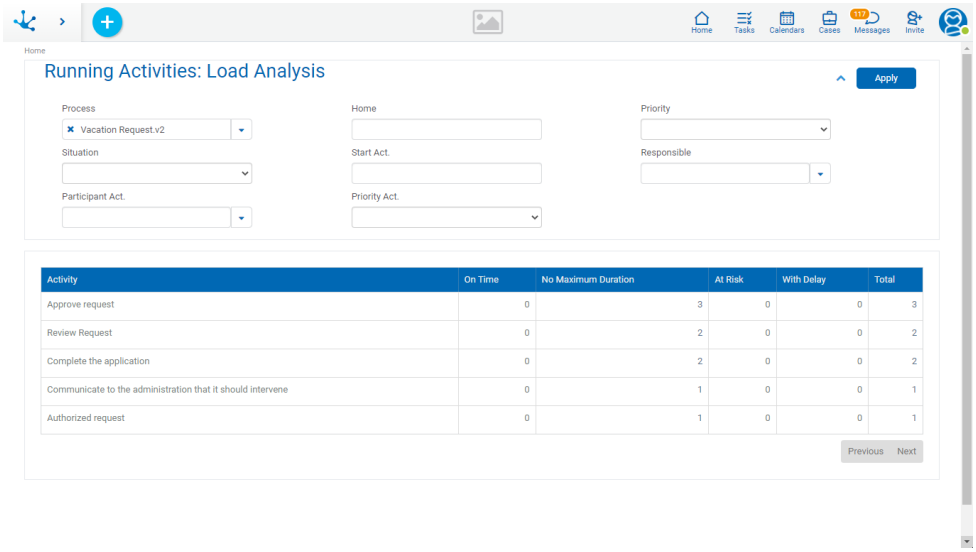


3.8.9. Activity BAM - Load Analysis

Phase 7 > Activity monitoring

In this report, the behavior of activities in progress of a process can be analyzed (including or not all its versions). The amount on time, at risk and behind schedule is detailed for each activity.

The information can be displayed by applying the search filters shown in the upper section of the report.



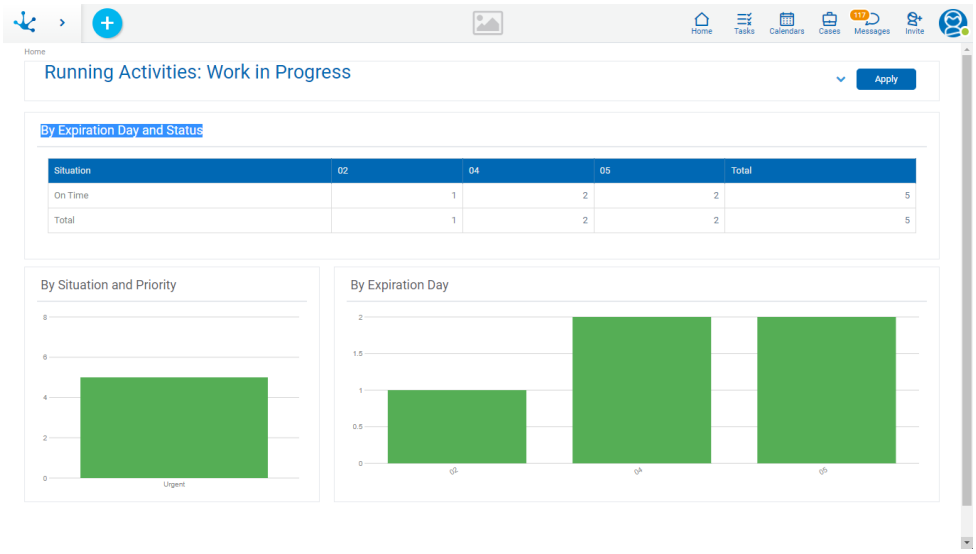
3.8.10. Activity BAM - Work in progress

Phase 7 > Activity monitoring

This report shows the status of the activities that are in execution, together with their distribution according to their expected due date in periods of time.
Regarding the status of the activities, it is graphically displayed how many are on time, how many at risk and how many are behind schedule.

Regarding the distribution of activities according to due date, the chart indicates how many are scheduled to expire in the time period indicated, and it may or may not count those that are behind schedule. If those behind schedule are counted, they are shown in the first period considered in the chart.

The information can be displayed by applying the search filters shown in the upper section of the report.



3.9. Configuration

Phase 8: Configuration

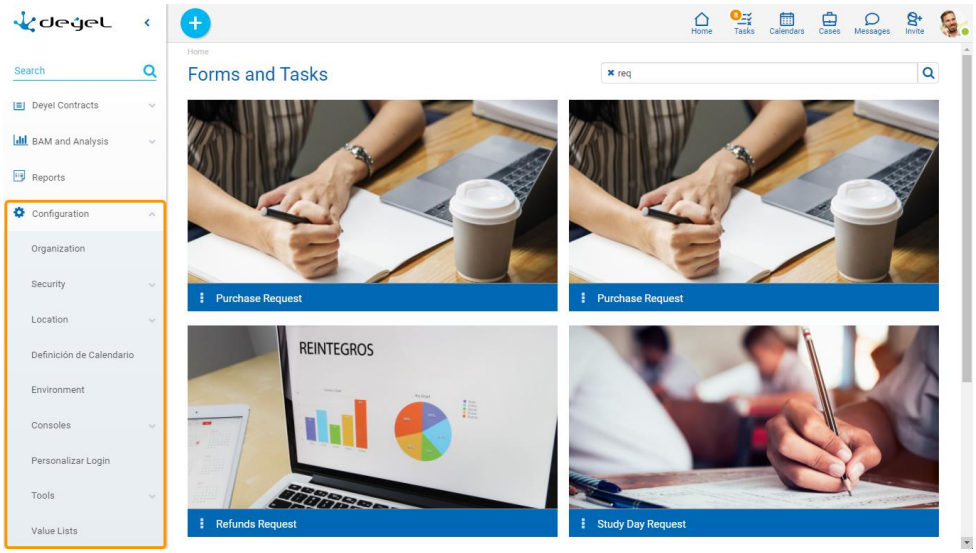
Configuration of **Deyel** is realized through the different menu options which are displayed by pressing "Configuration".

Organization
Allows to manage the organizational structure, defining the hierarchical relations between the different areas.

Security
Allows to define and administer the security scheme of **Deyel** through users, job positions, roles, agents, permissions and security functions.

Environment
Allows to define the preference settings for the use of **Deyel** through properties, that adapt the product operation to the specific requirements of the company.

List of Values
Allows to visualize the names on the lists of values of different applications and manage its values.



3.9.1. Security

Phase 8: Configuration > Security

Deyel uses a security scheme based on in two concepts:

- **Authentication** of the users, that allows to control who uses the product.
- **Authorization** for the specific functions that each user can use.

3.9.1.1. Authentication

Deyel keeps a register of all the users that are enabled to use the product, collecting their identifying data (user code, alias, email address) their personal and labor data, their state, etc.

The procedure of authentication ensures that all use of **Deyel** and its solutions are made by “someone” or “something” that can be identified as a user, inside the user register.

Also, to ensure that “someone” or “something” is who they say they are, their user or alias and their password must be informed.

Deyel uses different authentication methods but in each context only one of them should be used.

The mechanism of authentication to be used is defined in the configuration of the **Deyel** execution environment.

Native Authentication
This is the predetermined method of authentication. **Deyel** verifies that the user exists in its register and verifies the password, which is stored encrypted in its database.

LDAP Authentication
When an organization uses directory services to register its users, **Deyel** can be configured to delegate in LDAP the authentication process.

Federated Authentication IDM
Deyel implements this type of authentication by integrating with an external Identity Manager and delegating user authentication to it.

Federated Authentication
Deyel implements this type of authentication integrating with Google. When an organization uses Google accounts for their users, the authentication process can be delegated to Google.

Custom Authentication
This kind of authentication delegates on an advanced rule the users authentication.

Mixed Authentication
Allows an organization to use simultaneously different types of authentication.

Azure AD Authentication
Deyel implements this type of authentication integrating with Microsoft Azure Active Directory. When an organization uses Microsoft accounts for their users, the authentication process can be delegated to Azure Active Directory.

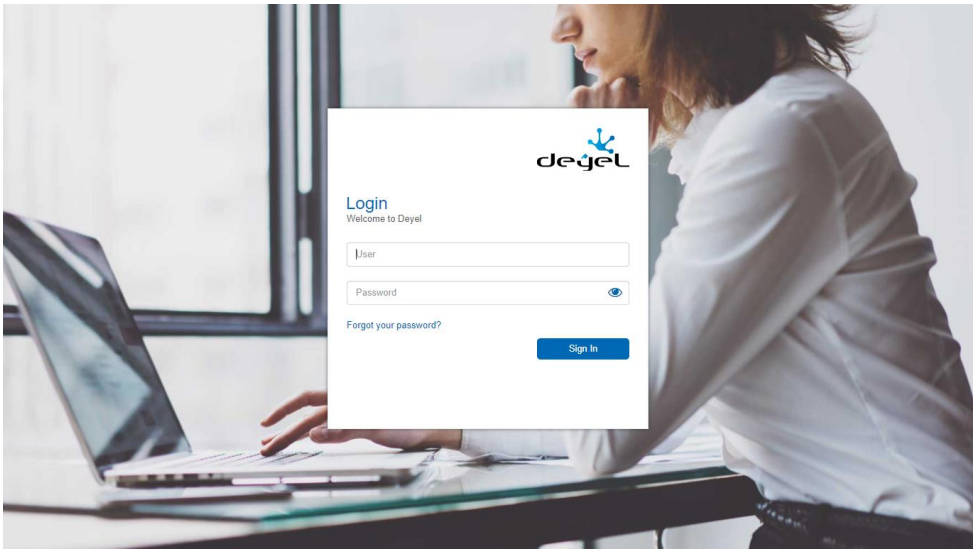
Multiple Factor Authentication
Deyel allows the use of a second authentication factor called 2FA for the native and personalized authentication methods.

3.9.1.1.1. Native Authentication

This is the predetermined mechanism of authentication.



The user enters the user code or alias and the password on the user portal access page.



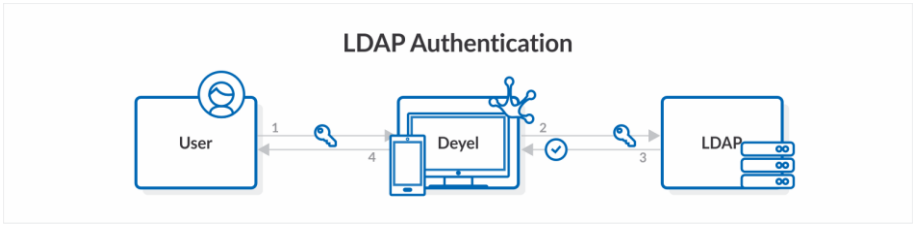
Deyel verifies that a user exists in its register, with the user code or alias informed and verifies the password. Access keys persist in **Deyel** database encrypted with SHA3.

If the user or the password are incorrect, the access to the product is denied.

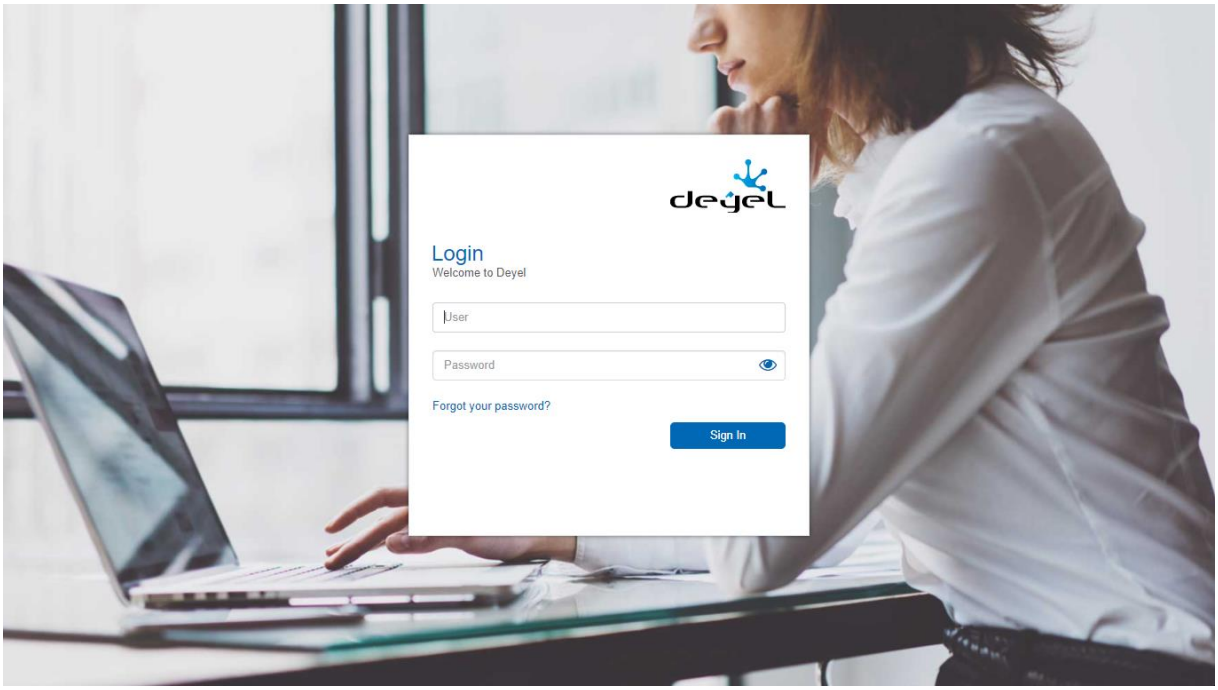
For this method, it is possible to optionally use a second authentication factor called 2FA, which is based on TOTP (Time-based One-Time Password algorithm) one-time keys.

3.9.1.1.2. LDAP Authentication

When an organization uses directory services to register its users, **Deyel** can be configured to delegate the authentication process to LDAP.



The user enters the user code or alias and the password on the user portal access page.



Deyel verifies that a user exists in its register, with the user code or alias entered and delegates the authentication to LDAP. If the user is registered in **Deyel** and LDAP reports a correct authentication, then the access is allowed.

> *IMPORTANT*
>
> *Before activating LDAP authentication it must be ensured that a registered user exists in Deyel and can authenticate correctly in LDAP.*
> *If none of the users meet these conditions, it is not possible to enter the portal.*
>
> *Deyel verifies that the administator that is configurating the environment can authenticate correctly in LDAP.*
> *When LDAP is used as one of the Mixed authentication methods, this last verification is not done.*
>
> *When Deyel can not establish communication with LPAD server, an administrator is allowed to enter the portal by using Native authentication.*
> *In these cases, the user does not have all administration options available, but can only access to the environment configuration, to reconfigure the mechanism of authentication.*
>
> *With this mechanism of authentication the option "Forgot your Password" is not available.*
> *If the user does not remember their password, they must observe the procedures that the organization determines to solve the problem.*

In the configuration of the execution environment of **Deyel**, different aspects of the integration with LDAP can be configured.

LDAP - Server Connection
Configuration of access to LDAP server. All properties are required to activate LDAP authentication.

LDAP - User Search
Configuration of users search in LDAP Directory. Establishes the search subtree, LDAP attributes which are considered search keys and additional user selection filters.

LDAP - Attribute Synchronization
Configuration of **Deyel** user properties that are synchronized with LDAP attributes

Attribute Synchronization

Deyel allows certain properties of the users can be recovered from attributes of LDAP directory, avoiding that these properties can be modified in **Deyel**.

When **Deyel** connects correctly and determines that the user exists in LDAP, it is marked as a "Synchronyzed User". This indicates that some of their attributes have been obtained from LDAP and can not be modified in **Deyel**. In the same way, **Deyel** automatically removes the "Synchronized User" mark when it determines that the user does not exist in LDAP. This way its attributes can be modified in **Deyel**.

There are different moments in which these attributes synchronization is done.

Login

When the user authenticates correctly against LDAP, attributes are synchronized. If **Deyel** detects an error that does not enable to do this synchronization, a register of this is left in the logs console and log in to the portal is not allowed.

User Creation

When a user is created, by informing the user code or the alias, the existence in LDAP is verified. If the user exists in LDAP, attributes are obtained and remain protected. They can not be modified in **Deyel.** If the values recovered from LDAP are incorrect or the user does not exist in LDAP, creating the user in **Deyel** is not allowed.

User Modification

When a user is modified, **Deyel** syncs again the attributes before showing the information on the screen. If **Deyel** detects an error that does not enable to do this synchronization, a register of this is left in the logs console and the synchronization is not done. Then, there is access to the user information, but the synchronized properties can not be modified from **Deyel**.

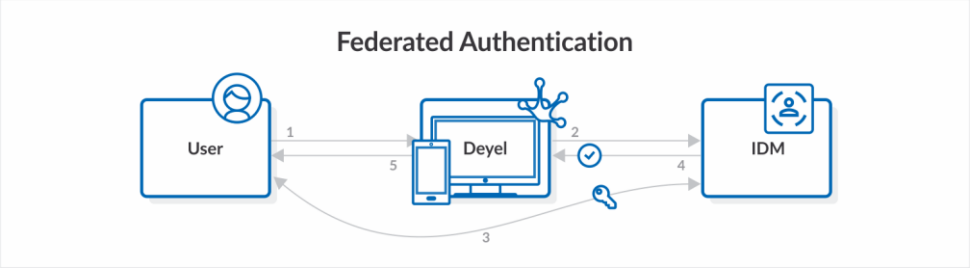
Query and Deletion of Users

When these operations are executed, an attributes synchronization is also done before showing the user information on the screen. If **Deyel** detects an error that does not enable to do this synchronization, a register of this is left in the logs console and the synchronization is not done.

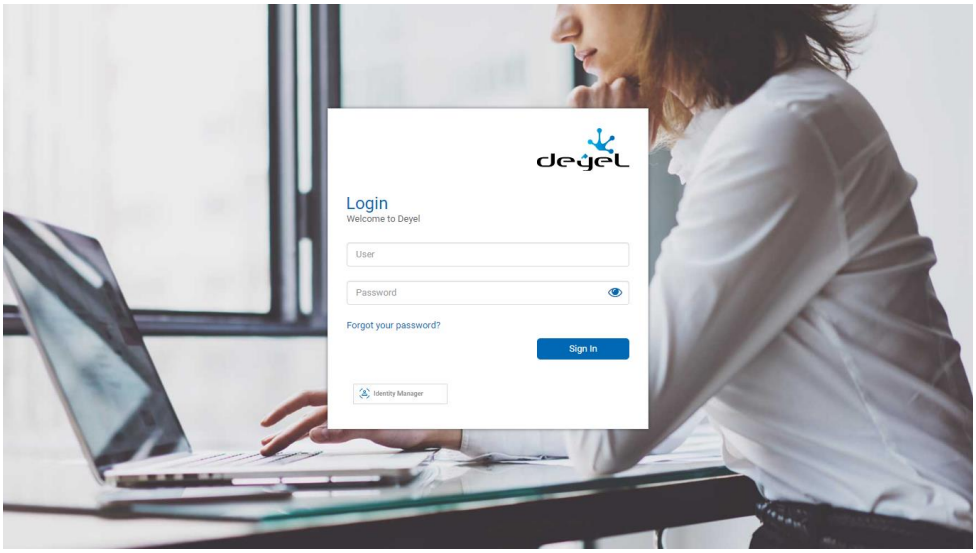
3.9.1.1.3. Federated Authentication IDM

An organization can use the services of an Identity Manager to do the authentication of its users. This way a single access mechanism is granted to the applications, preventing users from having to remember multiple access credentials.

Within **Deyel** use of this type of authentication can be made.



The access page to the user portal presents a button that allows to redirect users to the login configured in the IDMAuthorizationCode adapter to enter their credentials.



Pressing the "Identity Manager" button takes the user to the login panel where they enter their credentials.

When the user logs in, the Identity Manager informs **Deyel** that the user has been successfully authenticated and it gives back a token needed for the subsequent execution of a business rule that is configured in the mentioned adapter.

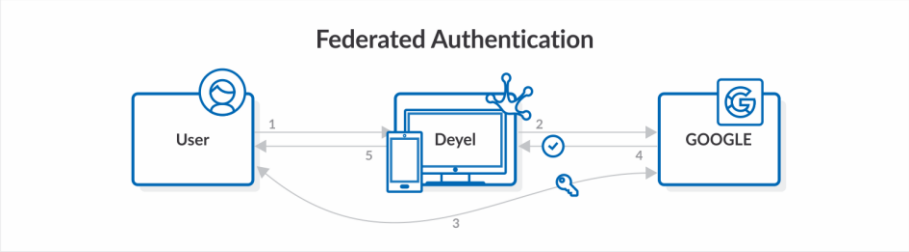
Deyel invokes such business rule to which the previously obtained token is passed as a parameter. Then, **Deyel** verifies that a user exists in their register, whose user name or alias coincides with the one obtained through the rule.

Only when the user is successfully authenticated and is registered in **Deyel** the product is allowed to be used.

3.9.1.1.4. Federated Authentication

An organization can use the services of an identity provider (IDP) to do the authentication of its users. In this way it grants a single access mechanism to the applications, avoiding users having to remember multiple login credentials.

Deyel implements this type of authentication integrating with Google. When an organization uses Google accounts for its users, the authentication process can be delegated to Google.



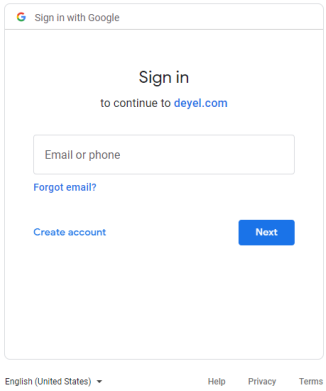
In the access page to the user portal there is a button that allows to redirect users to Google login in order to enter their credentials.

When users enter their account, Google informs **Deyel** that they have been successfully authenticated and the access code is never informed to **Deyel**.

Deyel only verifies that a user exists in its register, whose email account is the Google account entered. Users can use the product only when they are correctly authenticated to Google and they are defined in **Deyel**.



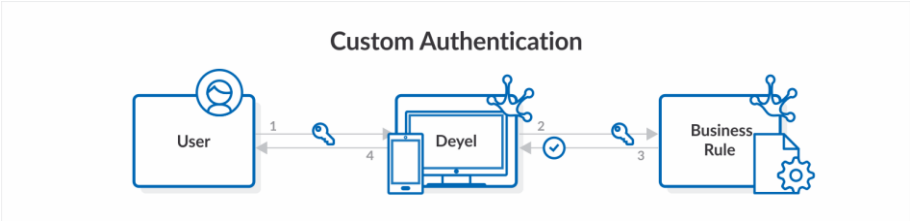
By pressing the button “Enter with Google” the users go to the login panel where they enter their credentials.



When users enter their account, Google informs **Deyel** that they have been successfully authenticated. **Deyel** verifies that the users are defined and their emails are the Google accounts they entered.

3.9.1.1.5. Custom Authentication

This kind of authentication uses an advanced rule to authenticate users.



The user informs their user code or their alias and password, in the access page to the user portal. **Deyel** invokes this advanced rule, passing the data entered by the user. The advanced rule implements the authentication logic and returns if it is correct or not.

When **Deyel** receives that the authentication is correct, it is verified that the user exists in **Deyel** and allows entry.Passwords do not persist in **Deyel**.

For this method, it is possible to optionally use a second authentication factor called 2FA, which is based on TOTP (Time-based One-Time Password algorithm) one-time keys.

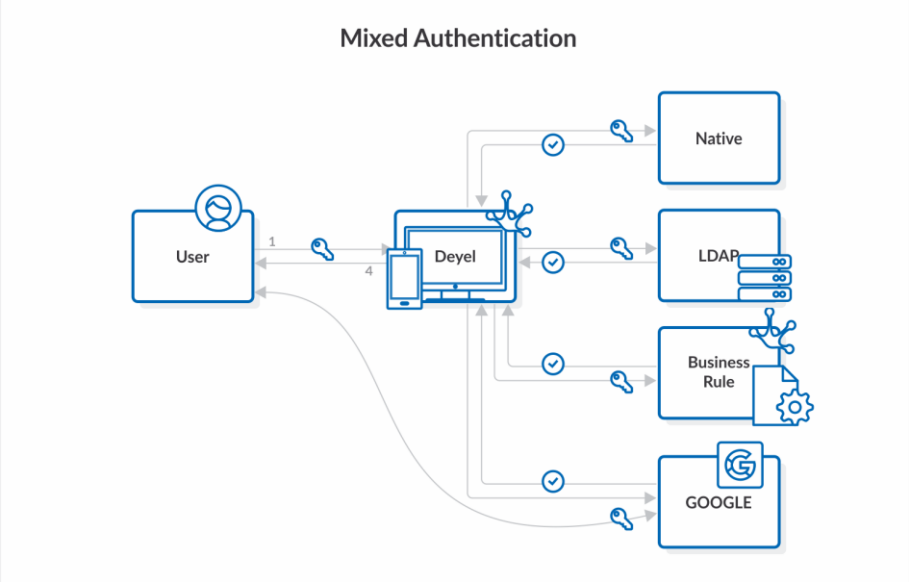
3.9.1.1.6. Mixed Authentication

The mixed authentication allows to combine multiple schemes of authentication, in a predetermined order.

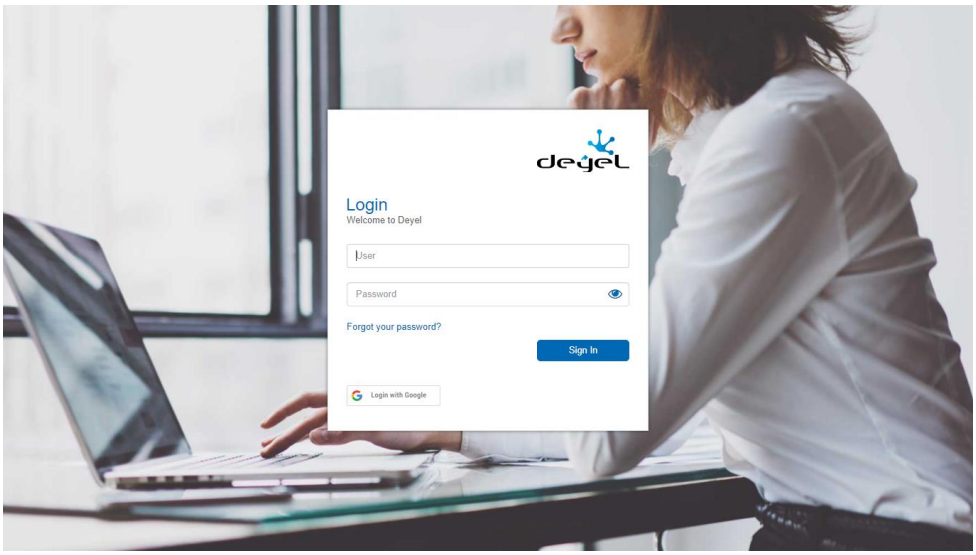
Let´s consider for example an organization that delegates in LDAP the authentication of its internal users, but needs to use addtionally the native authentication and the federated authentication (Google), to authenticate other groups of external users, which are not registered in the directory services.

This situation is solved by using the mixed authentication, indicating in the environment configuration which are the supported authentication methods and in what order should they be applied:

- First Method of Mixed Authentication = LDAP
- Second Method of Mixed Authentication = Native
- Third Method of Mixed Authentication = Google



The access page to the portal allows to use the different alternatives.



If the user presses the button “Enter with Google”, they are redirected to the Google login. The user can use the product only when they are correctly authenticated in Google and registered in **Deyel**.

If the user informs their user code or their alias and password, then **Deyel** tries to authenticate them first in LDAP and then by Native authentication.

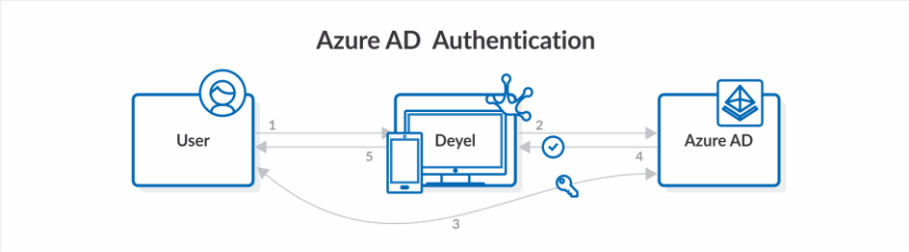
When a mechanism authenticates correctly, **Deyel** allows the login of the user if it exists.

If a mechanism fails, it continues with the next one and if every configurated mechanism fails, the user receives an authentication error and the product can not be used.

3.9.1.1.7. Autenticación Azure AD

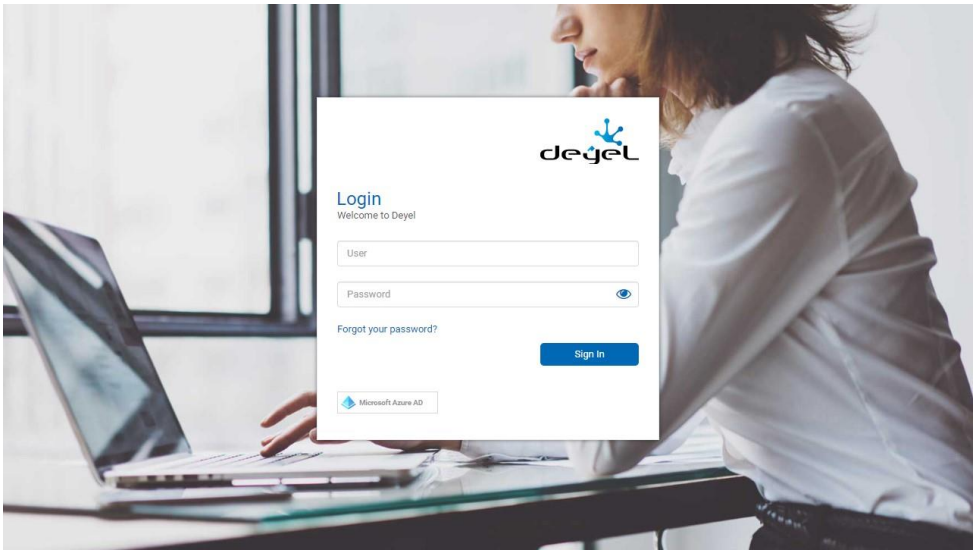
An organization can use the services of an Identity Manager to do the authentication of its users. This way a single access mechanism is granted to the applications, preventing users from having to remember multiple access credentials.

Deyel implements this type of authentication integrating with Microsoft. When an organization uses these type of accounts for their users, the authentication process can be delegated to Microsoft Azure AD.

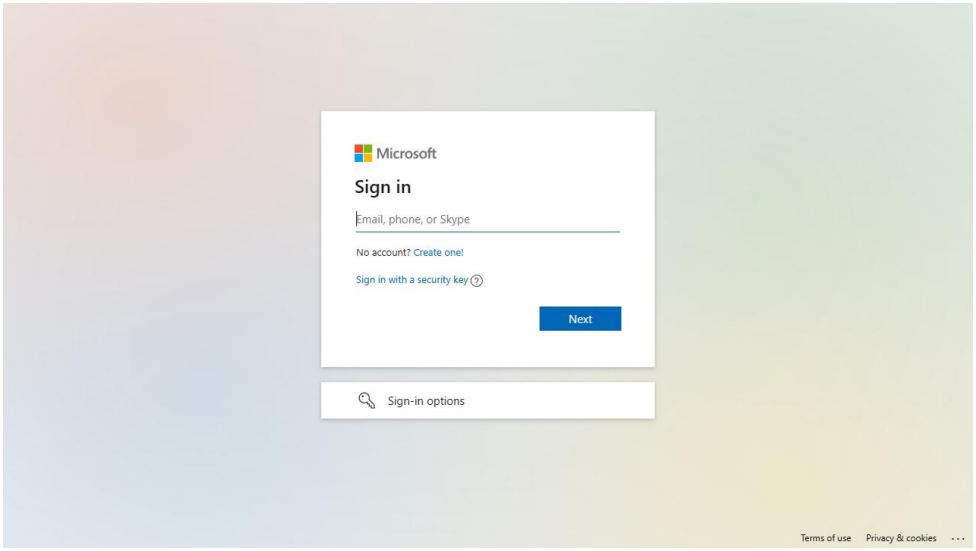


When users login, Microsoft informs **Deyel** that the user has been successfully authenticated and the access key is never informed to **Deyel**.

Deyel only verifies that a user exists in its register, whose email account is the Microsoft entered account. Users can use the product only when they are correctly authenticated in Microsoft and registered in **Deyel**.



Pressing the “Microsoft Azure AD” button opens the login panel where users enter their credentials.

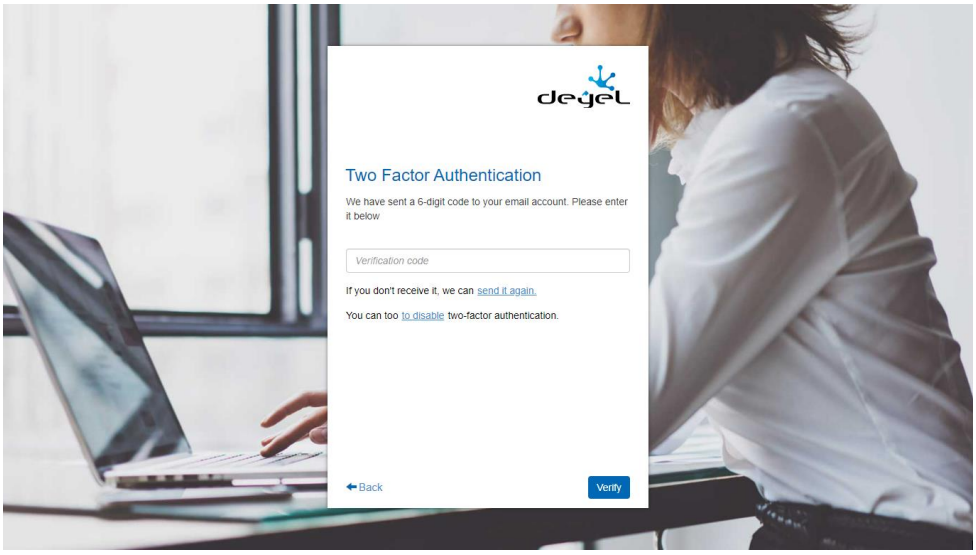


3.9.1.1.8. Multiple Factor Authentication

For the native and custom authentication methods, it is possible to use a second authentication factor called 2FA, based on the input of TOTP keys (Time-based One-Time Password algorithm).

When the user successfully authenticates with his username and password, he manages to pass the first authentication factor. Immediately, if that user has activated the double authentication factor, they must overcome it by entering one of these TOTP keys.

The Administrator from **Deyel** can enable or disable the use of the environment variable 2FA in the installation and each user can configure how to use such variable to access the portal.



When the user configures:

- using Google authenticator can enter the verification code that the authenticator generates. It is shown as the first alternative to avoid sending emails to the user.
- the verification codes to be sent to their institutional mailbox, they receive a new code in it.
- an alternative account to receive the verification codes, receive a code to the account recovery email.
- that wants to use the recovery code and the code entered is correct, the 2FA is validated and access is allowed.

By pressing the “disable” option to deactivate two-step verification, the user is asked to confirm the operation. Once confirmed, the following message is received:



And the administrator receives a request to disable the use of 2FA for that user, which is done by accessing the user's profile and modifying its configuration. These are manual user management procedures that can take time, so the applicant is prompted to try to identify themselves later.

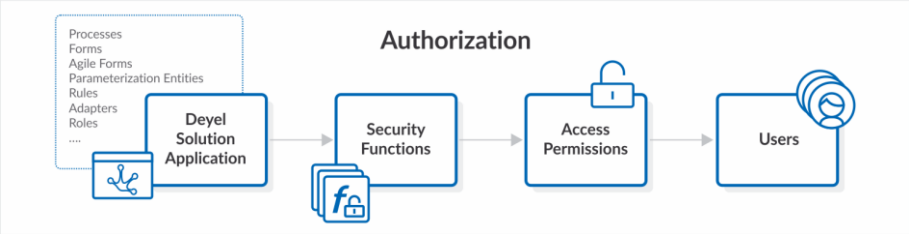
3.9.1.2. Authorization

The concept of authorization implies to verify which specific security functions each user can execute, preventing unauthorized actions to be carried out.

In the users profile they have predefined the access permissions and a user can only execute a security function, only if it is contained within their access permissions.

Modelers use **Deyel** to develop the applications the organizations needs and each application is modeled as a set of processes, business entities based on forms, rules, adapters, and roles, among other objects related to each other, with the objective of solving a determined requirement.

The design methodologies focused on the user contemplate the definition of the characters, to represent user groups that use the application similarly. Modelers can identify the characters in each application and determine which security functions they need to use. From this the access permissions assigned to the users are defined.

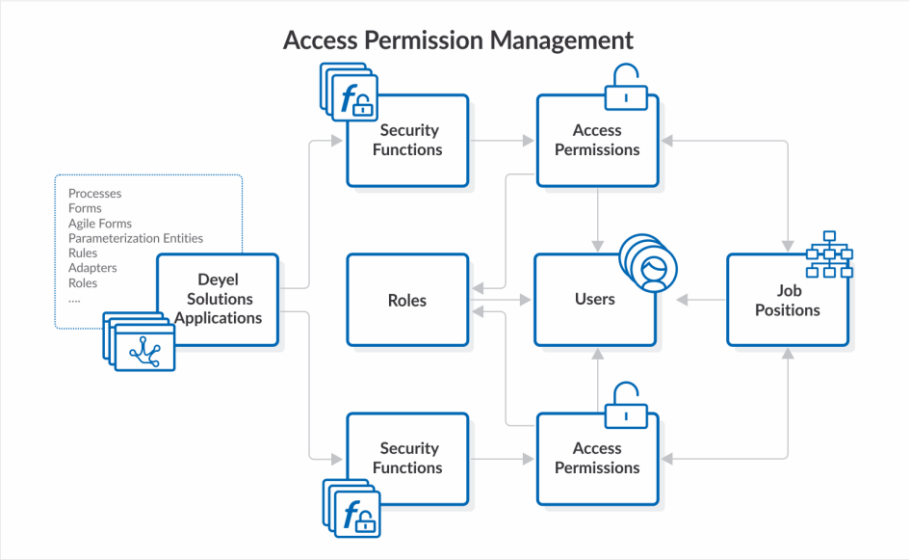


Access Permissions Management

Access permissions of each user need to adjust dynamically. The appearance of new applications, new features or changes on the user work assignments, imply that the access permissions that the user has assigned must be redefined.

The security administrator can add or remove access permissions to each of the users individually.

In organizations with great number of users or applications, this task can be complex. The concepts of roles and job positions allow to simplify it, assigning permissions to multiple users and not individually.



Predefined Access Permissions

Deyel and its solutions have a set of predefined access permissions, designed so that the different characters can use the feature they require.

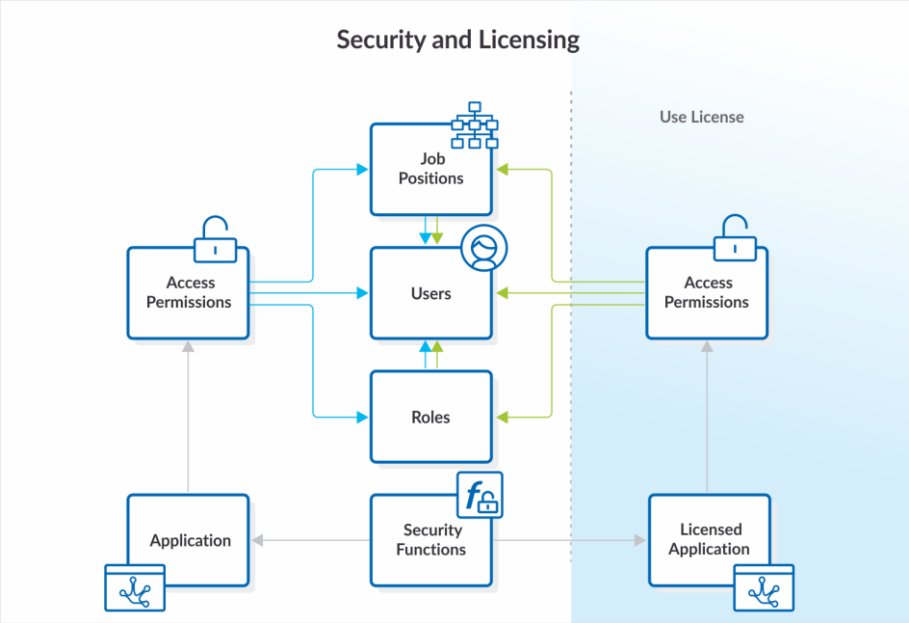
For example, **Deyel** has access permissions defined for end users, for business modelers, for IT modelers and for security administrators, among others. In the same way, **CRM** has access permissions designed for sellers and sales managers.

Predefined access permissions cannot be modified, in case of needing it a copy of them can be made.

3.9.1.3. Security and Licensing

Phase 8: Configuration > Licensing

The licensing scheme uses different concepts and affects the operation of the security scheme.



Licensed Application

A licensed application is the one that requires a use license to be used.

Use License

A use license is generated by the manufacturer and distributed in an XML file digitally signed. Each **Deyel** environment must import the corresponding use license.

A use license determines:

- The name of the company for whom the license is defined.
- Its identification.
- The platform or modality (Cloud, On Premise).
- The license type (Subscription, Perpetual, MyWay).
- The environment (Development, Test, Production, Trial).
- The validity period, with its issuance and renewal dates.
- The licensed applications can be used in the environment and for each one, the number of user licenses available.

User License

Each of the licensed applications defines different types of user licenses. The use license defines for each licensed application, the maximum quantity of user licenses of each type, that can be assigned in the environment.

In the case of **Deyel** the user license types are the following:

- Participant

A participant user uses **Deyel** to participate in the defined processes, as well as using forms, calendars, chats, etc. either initiating a case or performing some subsequent activity inside that same case.

- Agile Modeler
  An agile modeler is a business user that uses **Deyel** platform to define their web forms from predefined templates or doing its complete design. They can adequate them to the business needs, by adding a process or defining access permissions and its visibility, to finally publish them for its use.

- Deyel Modeler
  A Deyel modeler is an IT user that uses **Deyel** platform to design and build their object models for the development of applications of low code. Uses tools of graphic processes modeling and forms to build, document and implement the process and business entities of the organization. This type of modeler also defines the roles and agents, and can implement more complex business logic using assistants that allows them to incorporate Java components.
  They can implement the integration with other applications or basis of external data, behavior or more complex validations in the forms, using the advanced edition of the scripting. Uses tools to export and import their models and information of the environment.

In the case of solutions as CRM, Stock Management or Contracts, the user license types are the following:

- Participant
  A participant user can use the solution according to the access permissions they have assigned.

- Administrator
  An administrator of the solution is a user that uses **Deyel** platform to extend the solution.

Example:

- Enable the use of **Deyel**, considering up to 100 users of participant type, up to 50 of agile modeler type and up to 10 of Deyel modelers type.
- Enable the CRM use considering up to 100 participant users.

Access Permissions - Required User License

The access permissions of a licensed application establish what type of user license is required. In the case of **Deyel** there exist predefined permissions that require a license of Deyel modeler type, others require a license of an agile modeler and others only require a license of participant type.

Assignment of User Licenses and Access Permissions

On the profile of each user, it is indicated:

- The licensed applications that can be used and the type of user license that they are assigned to use them.
- The assigned access permissions.

When access permissions of a licensed application are assigned, it is controlled that:

- The licensed application can be used by the user.
- The type of user license is compatible with the required one by the access permission.

To authorize the execution of a security function, it is verified if it is in any of the access permissions assigned to the user.

> *The corresponding permissions to a licensed application, are ignored when:*
> - *The use license does not enable the use of such licensed application.*
> - *The use license does not enable the user license type required by the security permission.*
> - *The user does not have the user license type required by the security permission.*

MyWay Use License

All **Deyel** solutions have a licensing modality that allows them to be fully customized. Installing a "MyWay" type of license allows to transform the solutions available in the environment into non-licensed applications, so they can be modified as any other own application.

The solutions with use license from this type cannot be automatically updated when the version of **Deyel** is updated, but the adaptations made in the environment are strictly respected.

When a "MyWay" type of license is installed:

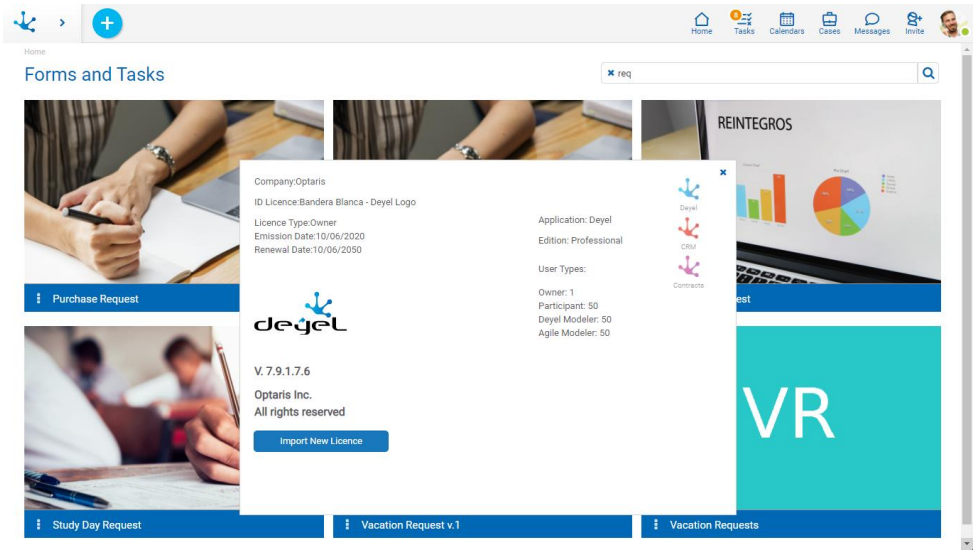
- All the solutions that the environment has licensed adopt the "MyWay" modality.
- Any user that has the "Global.Modeler" permission can model objects of the environment solutions to adapt them freely.
- Once this license type has been installed, the environment must remain in this licensing modality.

3.9.1.3.1. Showing Use License

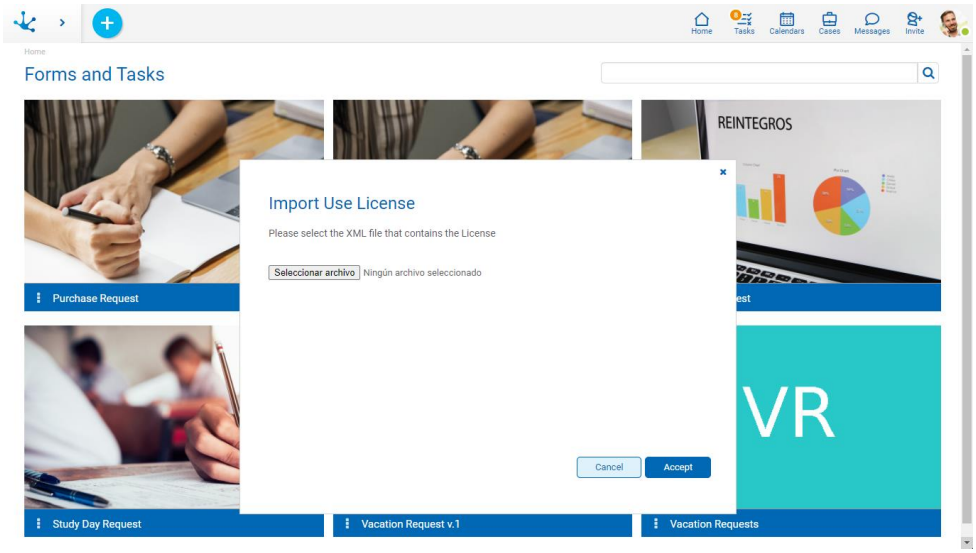
The information contained in the current use license can be shown from the functionality "About" on the top toolbar of the users portal.

About Deyel

When opening this panel the information about the use license can be seen.



When the user has access permission “Administrator”, they can import a new use license. Date and time when the user imported the current license are informed under the button "Import". When pressing this button, a window which allows to upgrade the use license of the environment is displayed.



The XML file that contains the use license must be selected before pressing the button “Accept”.

Deyel verifies the digital signature of the XML file to ensure its integrity and verifies that the new license has been generated in the environment and that it has the correct validity.

After importing the use license, a button that allows to restore the previous license is temporarily enabled. This option allows to restore situations where the use license was improperly imported.

3.9.1.3.2. Licensing Controls

Deyel controls the environment licensing and sends out alerts when users enter it.

Non-existent License

If the use license is not detected, a panel is opened to initially load the license.

Invalid License

The use license exists but the validity of the digital signature is not verified, then the license is invalid and the user is notified when entering the portal.

User Level Licensing

It is verified that the connected user has user licenses granted and are compatible with the use license. If the user has not user licenses assigned, they receive a message informing them of the situation.

Number of Licensed Users

For every licensed application it is controlled that the number of user licenses assigned do not exceed what is defined in the use license. This control is performed for each type of user license.

Expired Licenses

The use license has a specified renewal date. It is considered that the license expires the day after such date.

3.9.1.4. Agents

An agent is a type of participant that allows to dynamically determine a user, an organizational unit or a role, based on certain logic during the execution of a process.

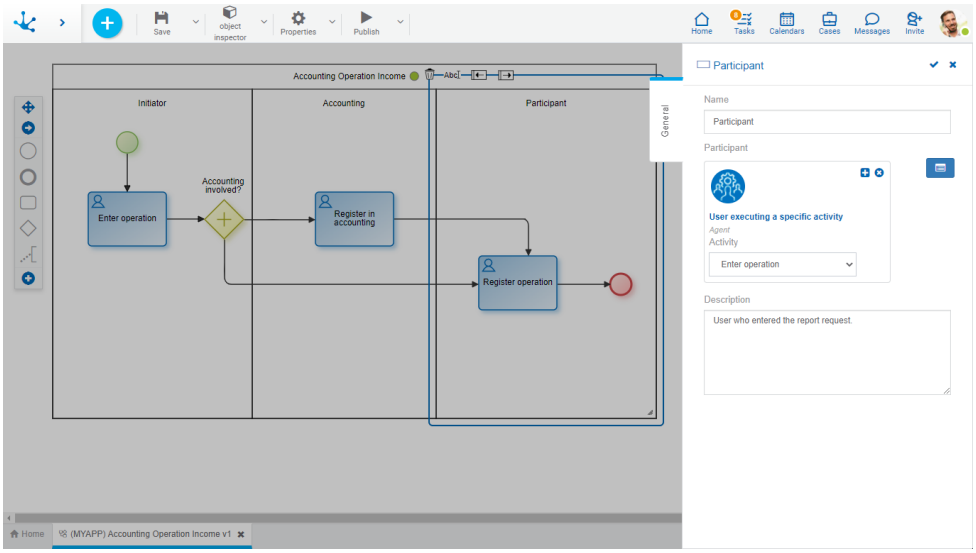
Predefined Agents

When modeling processes, a set of predefined agents can be used to establish who can execute the activities found in a lane.

User Executing a Specific Activity

This agent returns the user that executed a specific activity. This agent is used when there is the need to indicate that an activity is made by the same user that executed the activity defined.

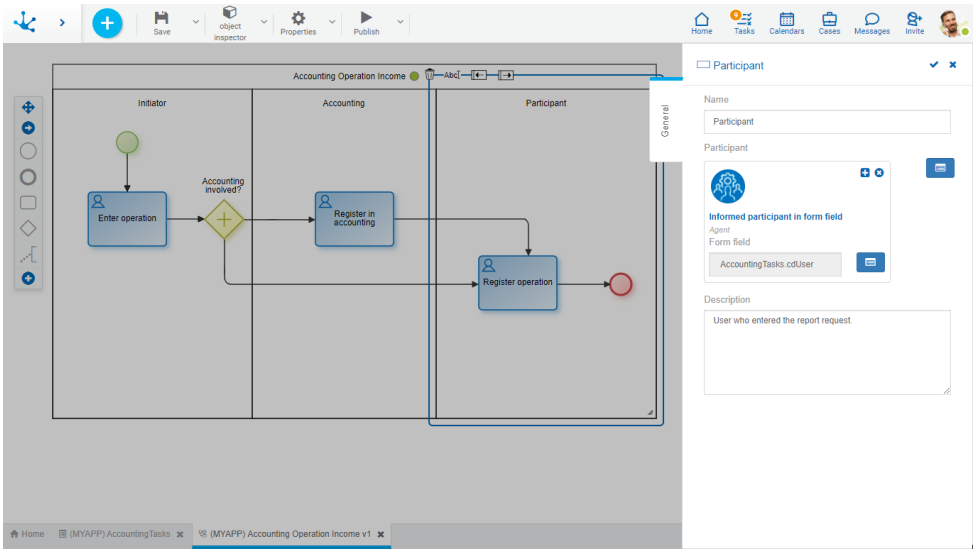
Usage Example: The activity “Register operation” is executed by the same user that executed the activity “Enter operation”.



Participant Informed in Form Field

This agent returns the participant from a form field. This agent is used when there is the need to indicate that an activity is made by the user indicated in the defined field of a form related to the process.

Usage Example: The activity “Register operation” is executed by the user shown in field “cdUser” of the form"TareasContables".



Administrator of the User OU

This agent returns the user that is administrator of the organizational unit the executing user of the last activity belongs to.

Any User

This agent returns the the user that is currently logged in. If a process has this agent as the participant of the initial lane, every **Deyel** user can initiate the case.

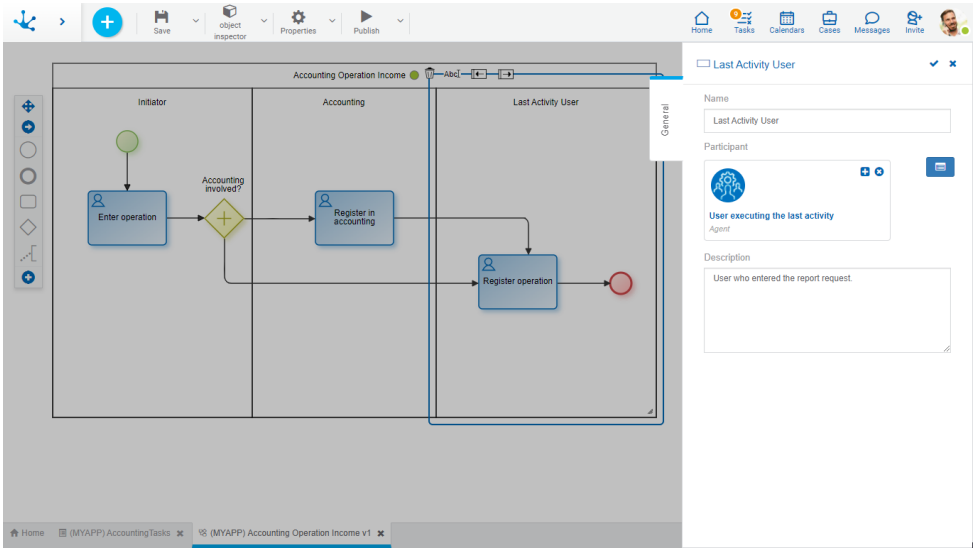
Authorizer

This agent returns the user defined as authorizer in the profile of the user that executed the last activity.

User Executing the Last Activity

This agent returns the user that initiated the execution of the last activity. This agent is used when there is the need to indicate that an activity is made by the same user that executed the activity that goes before.

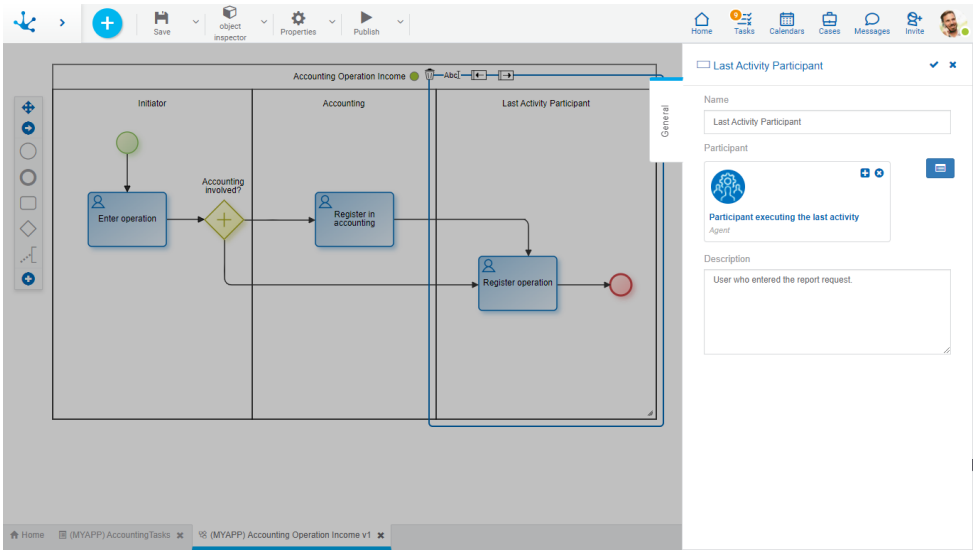
Usage Example: The activity “Register operation” is executed by the user that initiated the case or by the user that executed the task ”Register in accounting”, if the case involved the accounting department.



Participant Executing Last Activity

This agent returns the participant that initiated the execution of the last activity. It is used to indicate that an activity is done by the same participant..

Usage Example: The activity “Register operation” is executed by the same user that initiated the case or by the organizational unit “Accountant”, if the case involved the accounting department.



Initiator

This agent returns the user that executed the first activity of the process. It cannot be used in the first lane of a process.

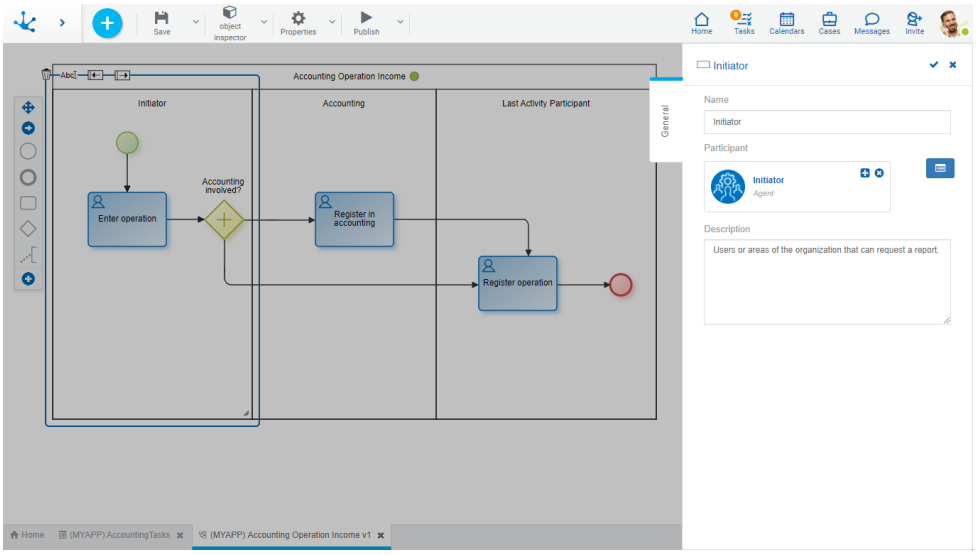
CRM User

Returns the user that is currently logged in as participant only if they have CRM Solution license. If a process has this agent as participant of the initial lane, every user with CRM license can initiate the case. It is recommended to use this agent only to define the participant of the initial lane of the process.

Initiator Restricted by Function

This agent evaluates if the current user has in their permissions the security function "Initiate Process" corresponding to the current process. This agent can only be used in the first lane of the process.

Usage Example: The activity "Register operation" must be executed by a user that has a permission permit with the security function "Start by agent Entering Accounting Operation" enabled.



3.9.1.5. Security Functions

A security function represents an operation that can be done on an object. It can be included in one or more permissions thus authorizing the users that have those permissions to do such operation.

Every operation available in **Deyel** or in its solutions has a corresponding security function defined. Besides, every object that is modeled has a set of security functions that are automatically created when saving and publishing the object.

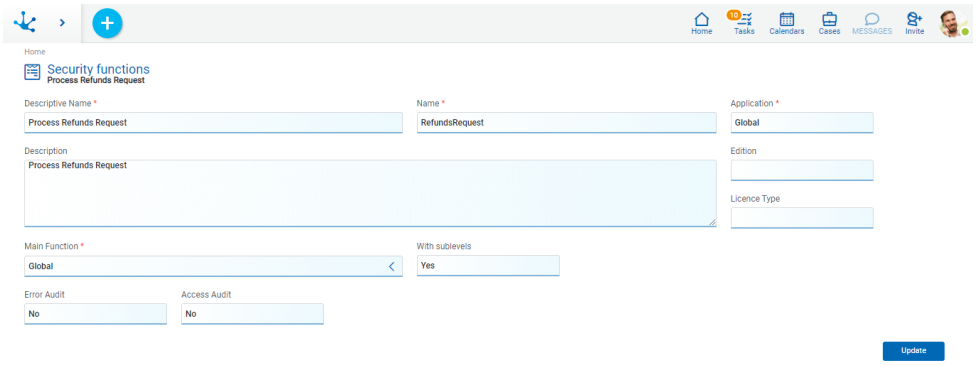
Security functions can be shown or assigned to permissions from the option "Permissions" on the menu and particularly security functions of the modeled objects, from the object modeler as well.

Of Modeling

They authorize the execution of the modeling operations, such as save, publish, export, delete. They are generated when saving the object and their assignment to permissions has an immediate impact when saving or publishing an object.

Of Use

They authorize the execution of the operations on the instances of the objects, like create, modify, delete, show, when they are used from the portal. They are generated when the object is published. The modifications in the assignment of use functions do not have a real impact on the use of the object, until it is published.



An asterisk "" on the label indicates that the property is required.*

Properties

Descriptive Name
Complete name of the security function.This text is visualized on security functions grid.

Name
It is an abbreviated name. It is used when referring to the security function in an error message or in any mention that needs to abbreviate the descriptive name.

Application
Indicates the applications to which the security function belongs to, it is the same as the one his Father Function.

Edition
It indicates the edition of **Deyel** or of the licensed application required for the execution of the function. Some functions are available only in the edition "Enterprise" of **Deyel** or "Professional" of the solutions.

License Type
Indicates the type of license the user needs to have assigned in order to execute the function. It is used for **Deyel** functions or licensed applications functions.

Description
Details information about the security function, extending what is expressed in the property Descriptive Name.

Superior Function
Allows to identify the superior security function inside the hierarchical structure. There exists a father function for each available solution and for each modeled application.

With Sublevels
Indicates that the security function has subordinated functions.

Audits for Access

When the value “YES” is defined an entry is generated in the audit registry each time a user executes the security function. Using the Logs Console these audit trails can be recovered and evaluated the use of the different security functions.

Audits for Errors

When the value “YES” is defined an entry is generated in the audit registry each time an error occurs during the execution of the security function. Using the Logs Console these audit trails can be recovered and evaluated the occurrence of these type of errors.

3.9.1.5.1. Functional Structure

For creating and managing security functions, it is possible to access the functional structure from the menu of **Deyel.**

- From the option “Configuration”, selecting “Security” and then “Security Functions”.
- From the search facility.

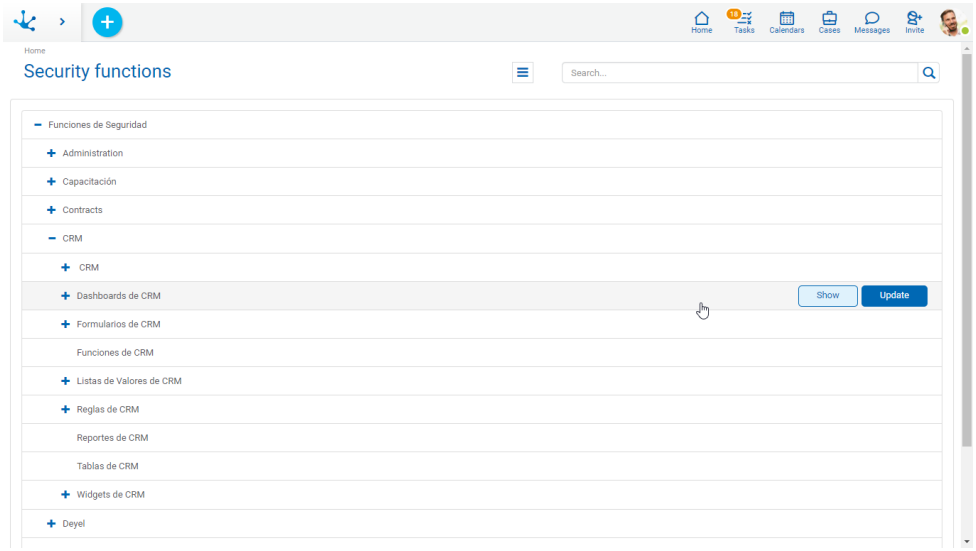
Security functions are identified from its property Descriptive Name and automatically generated when objects are modeled. They can be visualized as a hierarchical view or as an object grid.

It is possible to perform operations on each security function from hierarchical view and from the grid mode by using buttons.

Hierarchical View

Father element, which name is the name of the application, is located as sublevel of the top element “Security Functions” and security functions of the application and its objects are visualized under it. For each modeled object, its corresponding modeling security functions and of use are created.

The icon ☰ allows to change the visualization to grid mode.

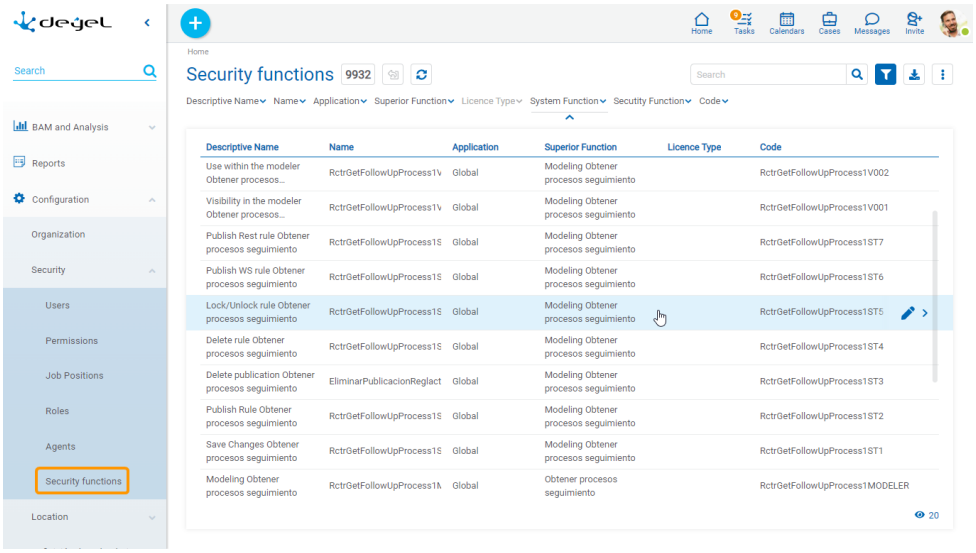


Grid

Security functions are visualized as a grid, with the standard presentation of results grids, with facilities of::

- Ordering
- Search Bar and Filters
- Data Download
- Operations

The icon allows to change the visualization to hierarchical view mode.



The following properties of security functions are visualized as grid columns:

- Descriptive Name
- Name
- Application
- Superior Function
- License Type
- Code

Filters can be applied by the following properties:

- Name
- Descriptive Name
- Application
- Superior Function
- Edition
- License Type
- System Function
- Security Function
- Code

It is possible to perform operations on each line of the security functions grid. By clicking on the line the display of the selected security function is done, while by means of the icon  the modification is done.

3.9.1.5.2. Operations

It is possible to perform operations of show and modification from hierarchical view and from grid mode, depending on the security permissions that the connected user has defined. When hovering over each of the lines the buttons or icons with available operations are visualized, depending on the visualization mode.

Show

Opens properties panel of the selected security function. Properties are visualized as not editable and depending on the security permission of the user, the button to modify that security function is enabled.

Update

Opens properties panel of the selected security function, with those properties that can be modified as editable. The update is done by pressing the button "Accept" and the user receives a message indicating that data have been saved.

> *When the security function belongs to a licensed application, only the attributes that define the audits for access and audits for errors can be modified.*

3.9.1.6. Security in Entities

Phase 2: Form Modeling > Advanced Tips > Entities Privacy

Security in an entity is related to the access the users have to the instances of such entity, through their user interface which is named form. An instance of an entity corresponds to a created form, that contains data in its fields.

Entities Classification According to the Security Level

Public
The only security that controls the access to a public entity is the one given by the user profile.

Private
An entity is private when in addition to the security definded in the user profile, there is one additional check per entity instance.

The verification takes into account the following concepts:

- Instance Owner
  Identification of the user that has access permissions to show, modify and delete the particular instance of an entity.

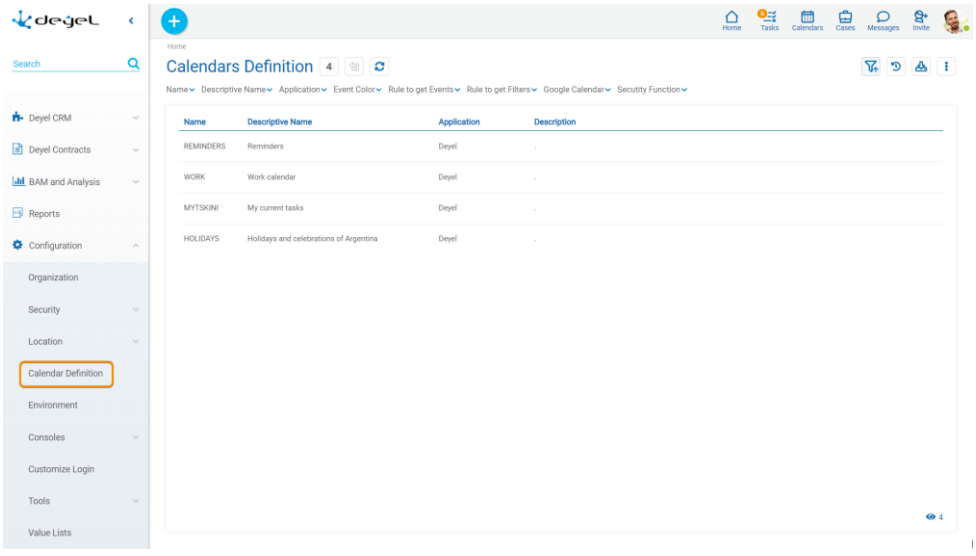
- Access according to Hierarchy
  The access to instance data in addition to the owner, takes into account the definition of the organizational unit which the owner user belongs to. If a user belongs to a unit that has private data access properties defined, the coordinator of such unit can have any of the accesses defined for the instances of his subordinates.

- Security to Functions
  Allows a user that is not an instance owner to access these instances despite being private, through security functions assigned to a permission. The user is allowed to show, modify or delete private instances.

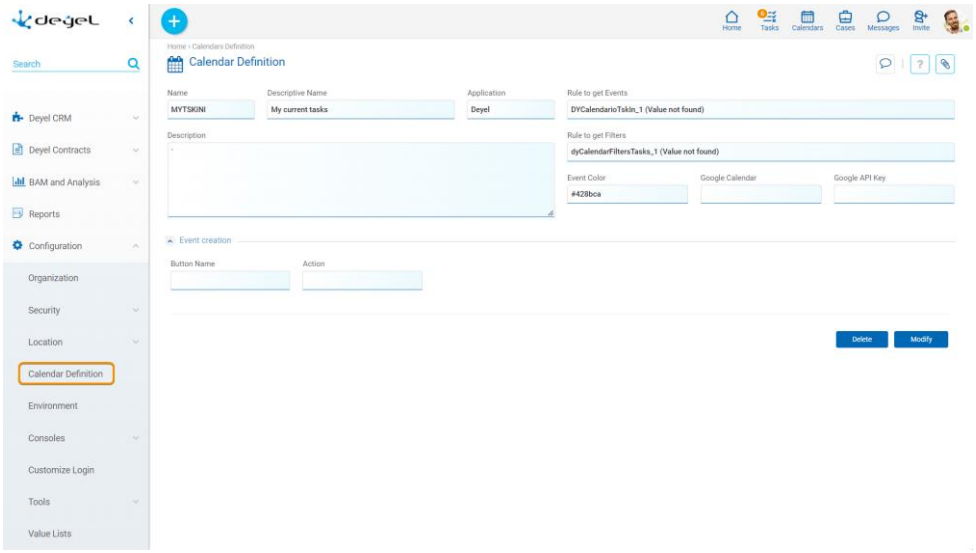
- Access from a Process
  Allows access to the instances of a form from a case even if the user does not have the necessary security functions for the form or the form has privacy defined

3.9.2. Calendars Definition

Deyel allows to define and administer the calendars which are visualized through the "Calendars Definition" option from the menu.



Operations can be performed on each grid row. By clicking on the row, the selected calendar is consulted, while by means of the icons and perform an update or deletion respectively.



Properties

Name
Abbreviated or reduced name that identifies the calendar.
There cannot be more than one calendar with the same name for the same application.

Descriptive Name
Full name of the calendar, which is displayed in the calendars selection panel.

Application
Application to which the calendar belongs.

Description
It is an optional text that allows to document the purpose of the calendar.

Rule to get events
Identification of the advanced rule that is executed to retrieve the events that correspond to the calendar. This rule must be previously developed and can be selected using a wizard.
This property is optional for all calendars, except those that are not imported from Google, in which case it is required.

Rule to get filters
Identification of the advanced rule that is executed to retrieve the filters that can be applied in the retrieval of calendar events. This rule must be previously developed and can be selected using a wizard.

Event Color
Color with which the events of the selected calendar are graphically represented.

Events creation
Defines the list of actions that can be executed in the window of creation of events for the selected calendar. For each element of the list the following properties are defined:

Button Name
This is the text that is displayed on the button. The maximum number of characters is 30.

Action
It allows to select the action that is executed when the button is pressed. Depending on the action selected, different properties are enabled to be completed.

- New Case
  Process
  The process to start must be selected from a list of available processes.

- Create Form
  Form
  The form to be created must be selected from a list of available forms.

Correlation of start and end of the event with the variables of the process or form
It can be optionally set how to transfer the data entered in the properties Start and End of the window creation of events, to the fields of the selected form or associated to the selected process.

Start Date
Enter the identifier of the field where the user wants to transfer the start date of the event. If the indicated field is of type "Date and Time", the value entered is transferred to the field Start. If the indicated field is of type "Date", the value entered is transferred to the field Start and the time is not considered.

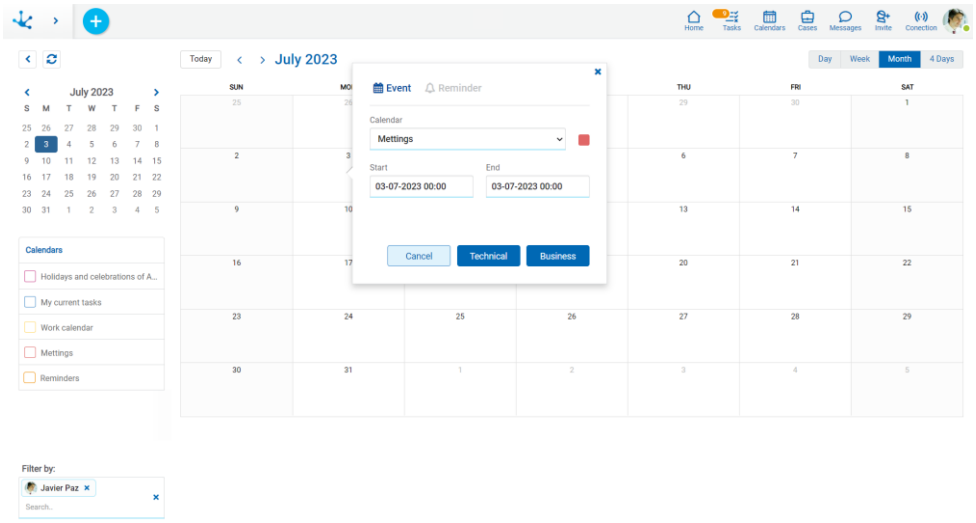
Start Time
Enter the identifier of the field where the user wants to transfer the start time of the event. It is validated that the indicated field is of type "Time".

End Date
Enter the identifier of the field where the user wants to transfer the end date of the event. If the indicated field is of type "Date and Time", the value entered is transferred to the field End. If the indicated field is of type "Date", the value entered is transferred to the field End and the time is not considered.

End Time
Enter the identifier of the field where the user wants to transfer the end time of the event. It is validated that the indicated field is of type "Time".

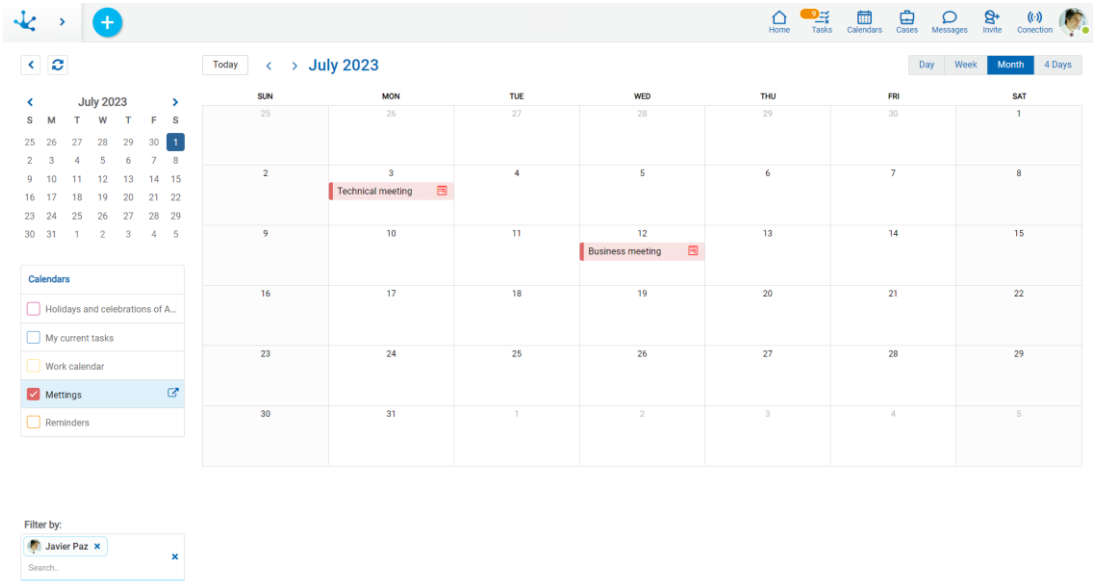
3.9.2.1. Creation of Events and Reminders

By clicking on a cell in the calendar events panel, a window is expanded which allows to create a new event or a new reminder.

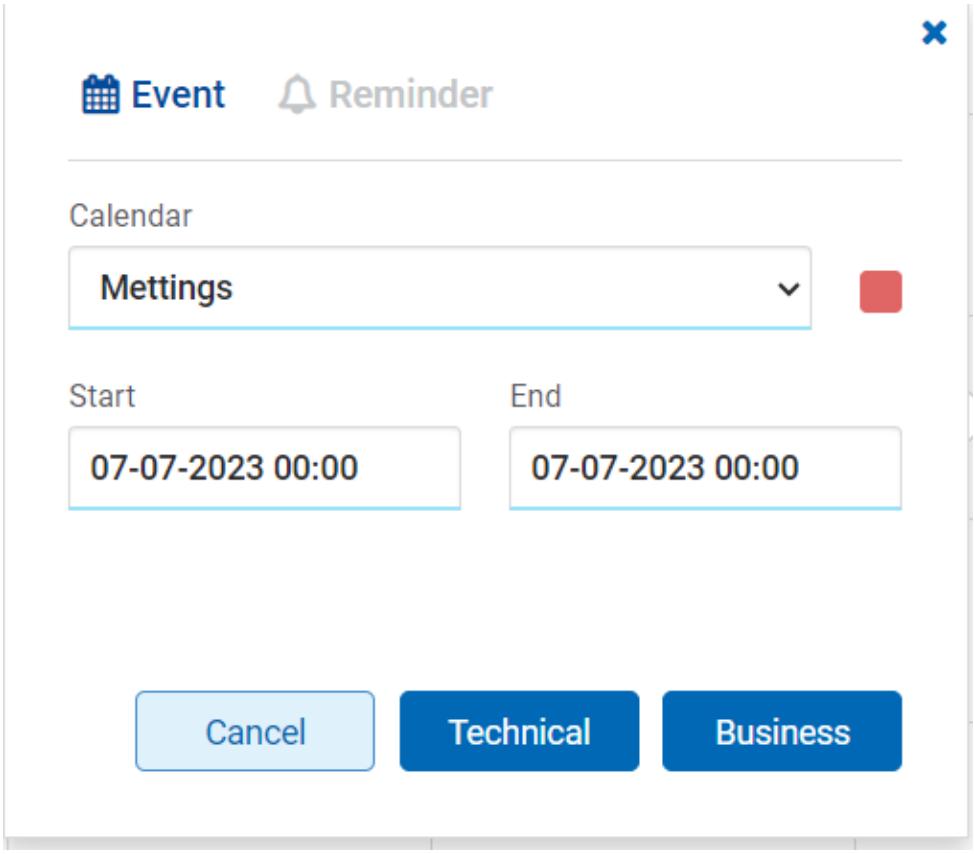


3.9.2.1.1. Creation of Events

Each user can define their own events and visualize them in the calendar.



To create a new event the calendar cell must be selected and the following window is expanded:



Calendar

The list of calendars that have some action configured to create events is displayed and that these actions are authorized to the user.

When the user selects one of the calendars from the list, its definition is accessed and the buttons for creating the event are displayed.

Only those selected in the calendar panel are displayed in the list.

Start and End

These fields determine the start and end of the event.

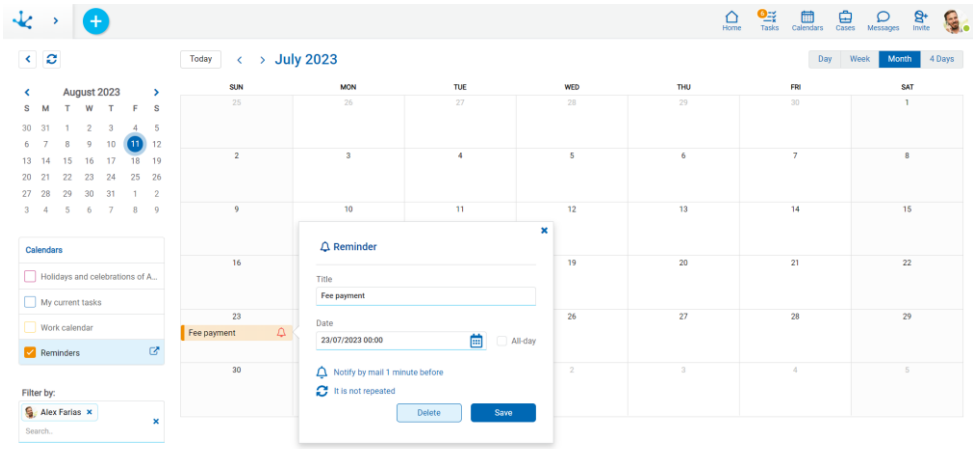
When the calendar is being displayed as “Day”, “Week” or “4 Days”, these fields are initialized according to the selected position of the events panel.

The date and time where the click was made is considered, on the other hand, when the display form is monthly, only the date is considered and the time is initially assumed as "00:00".

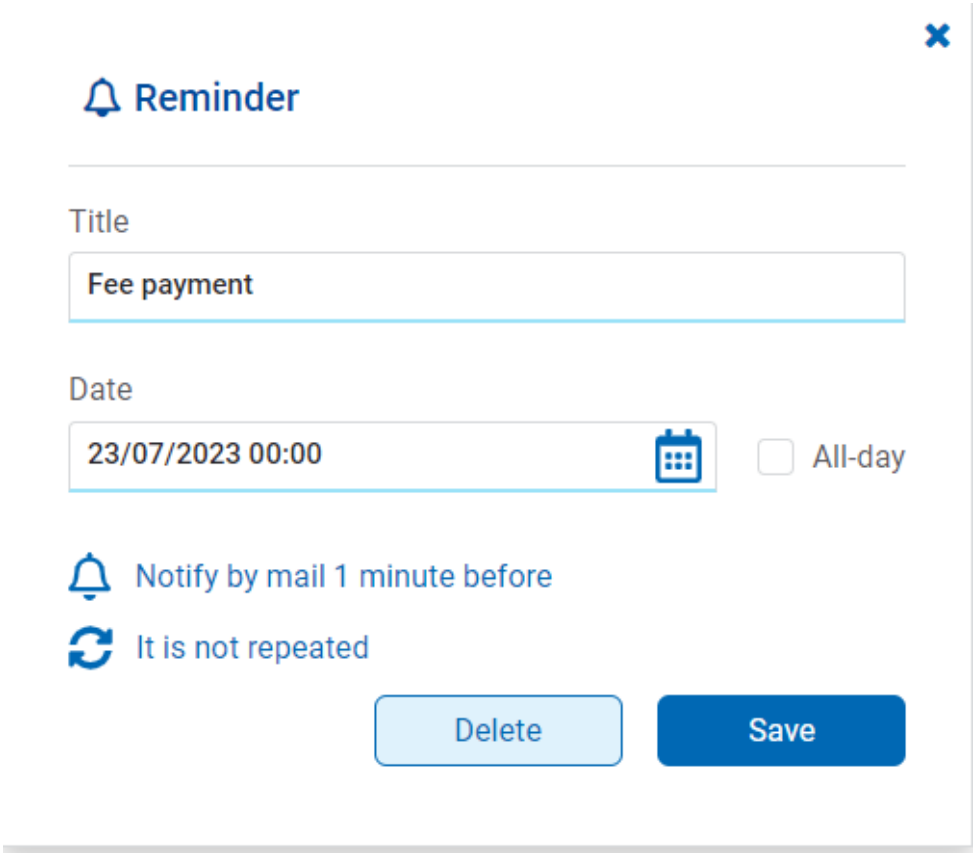
The user can change the proposed value for the "Start" and "End" fields.

3.9.2.1.2. Creation of Reminders

Each user can define their own reminders and visualize them in the calendar.



To create a new reminder the calendar cell must be selected and the following window is expanded:



Properties

Title

Text describing the reminder.

Date and Time

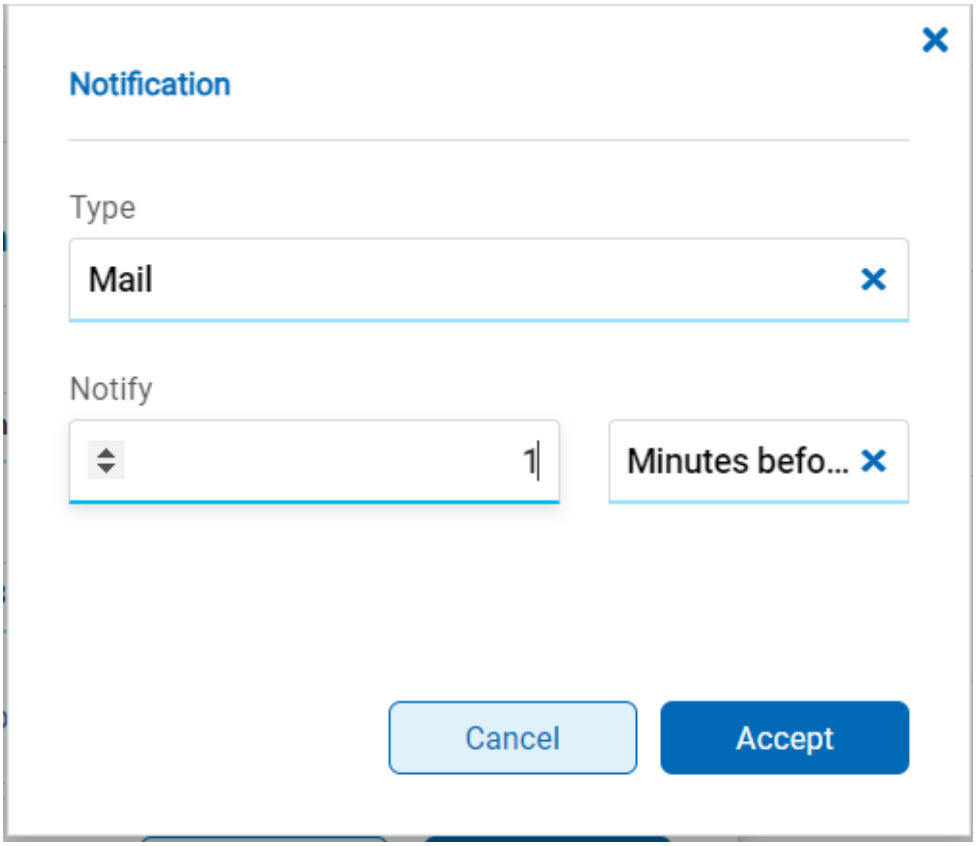
Date and Time in which the reminder must be shown.

All Day
Indicates that the reminder refers to the entire day.

Notification

Optionally, it can be configured that **Deyel** notifies the user in advance of this reminder.

Opens a panel to enter the notification data.



Type
Indicates the type of notification.

Possible Values

- Without Notification
- Email

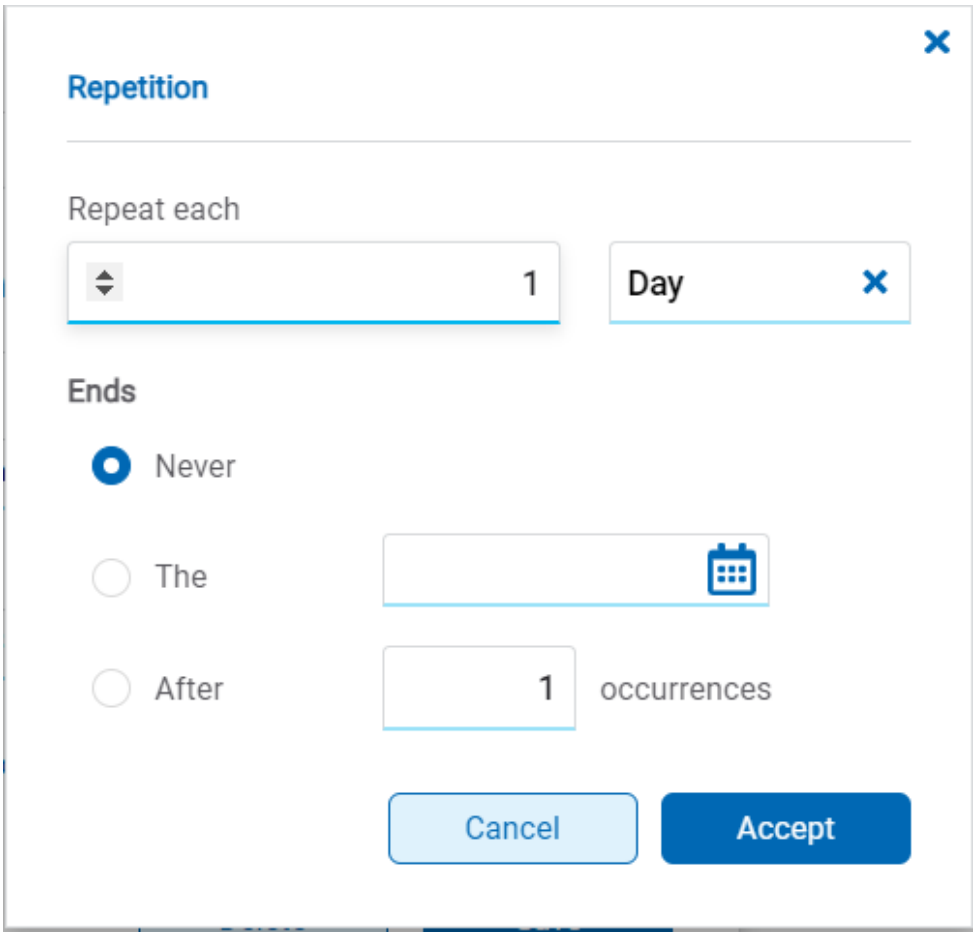
Notify
Allows to select the number of minutes, hours, days or weeks of anticipation with which to notify. When the notification All Day is indicated, only days or weeks can be selected in the preview period. The maximum anticipation that can be indicated is 30 days.

At the Time
When the notification All Day, the time when the notification is issued can be selected. Otherwise this property remains hidden.

Repetition

Optionally, a reminder can be defined as recurring.

Opens a panel to enter the repetition data.



Repeat each
Indicates the number of times the reminder should be repeated based on the selected period. Depending on the period, different properties are enabled to be completed.

- Week
  Is repeated
  Allows to select the days of the week on which the reminder should be repeated.

- Month
  Day

Depending on the selected cell where the repetition is being configured, it allows choosing the possible options to repeat the reminder.

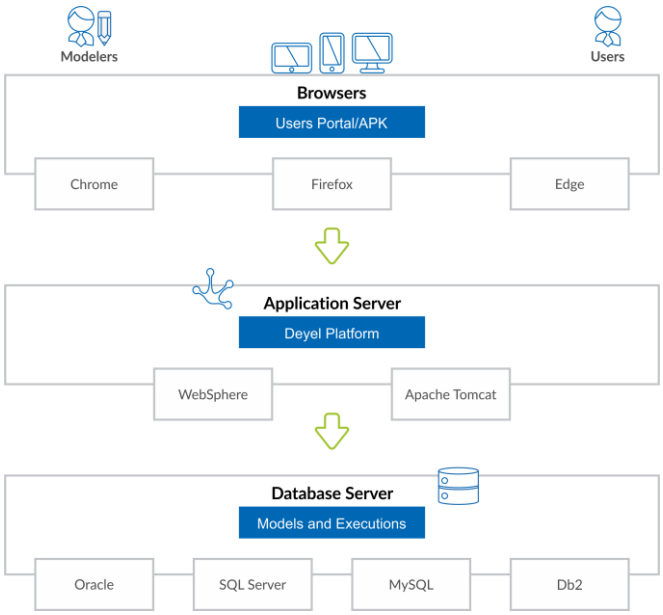
Ends
Defines when the repetition of the reminder should end.

Possible Values

- Never: It is repeated indefinitely.
- The: It is repeated until the indicated deadline.
- After: Indicates that the reminder ends after a certain number of repetitions.

3.10. Requirements

The following are the software requirements for the use and installation of **Deyel**.



Workstations (Cloud and On-Premise Modality)

HARDWARE	RAM	a minimum of 4 GB
	Screen resolution	• End user: a minimum of 1024x768 • Modeler: a minimum of 1920x1080
SOFTWARE	Browser	• Chrome, a minimum of 79.0 • Edge Chromium, a minimum of 79.0 • Firefox, a minimum of 74.0

		• Safari 13.1.3 and 14.1.2 (Only for User Portal)

> *It is advisable to keep browser versions up-to-date, since security vulnerabilities are regularly corrected.*

Application Server (On-Premise Modality)

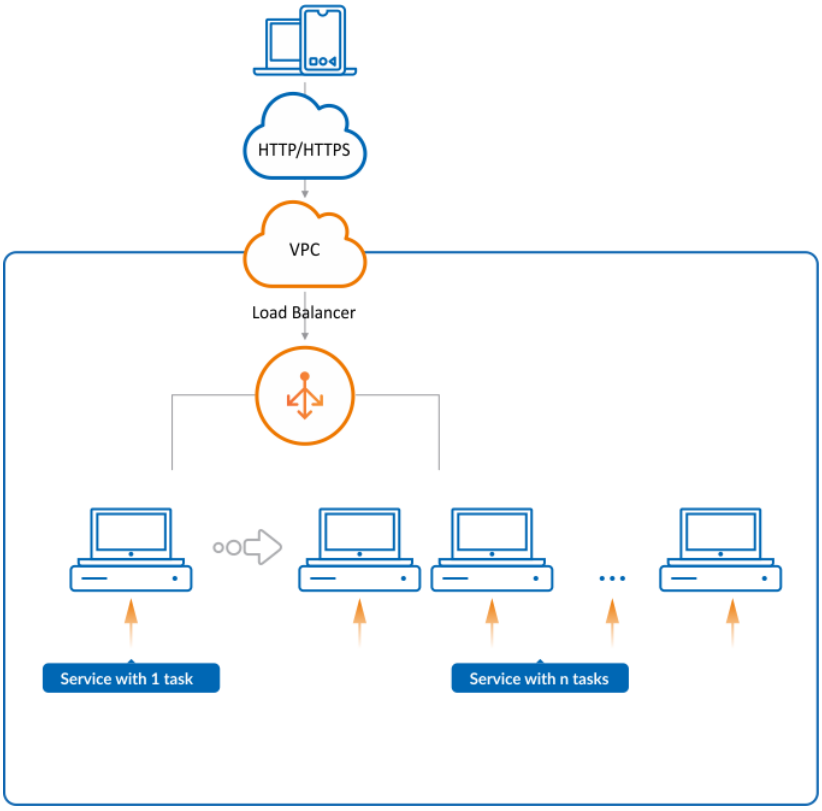
HARDWARE	RAM	a minimum of 6 GB
	HD	a minimum of 5 GB free space.
	Processor	64-bit / 2 cores / at least 2.4GHz
SOFTWARE	Web Service	• WebSphere 8 / WebSphere 9 • Apache Tomcat 8 / Apache Tomcat 9
	Java	Java 11

Database Server (On-Premise Modality)

HARDWARE	RAM	a minimum of 4 GB
	HD	a minimum of 10 GB free space
	Processor	64-bit / 2 cores / a minimum of 2.4GHz
SOFTWARE	Database	• Mysql 8 • DB2 9.5 or DB2 10.5 • Sql Server 2012 • Oracle 12c release 12.2 / Oracle 19c

3.11. Cloud Service Operation

Deyel Cloud allows to develop, test, deploy, and execute web and mobile applications without worrying about the operational aspects of the platform technology. Modeled applications and their data are hosted directly on the cloud infrastructure provided by Deyel Cloud.



The Deyel Cloud infrastructure uses Amazon Web Services (AWS), incorporating the latest technologies and techniques into its platform to offer a reliable, secure and high-performance cloud service.

Each client accesses each of their environments in isolation. Depending on the contracted license, the development, testing, quality and production environments can be accessed.

Features of Deyel Cloud

A set of features make Deyel Cloud's infrastructure robust and secure.

Execution Portal

Each environment has an execution portal where users can access to model and execute their applications.

Traffic is managed by the AWS Route 53 DNS service and AWS ALB (Traffic manager), which resolves and routes the URL to the environment. Each environment is deployed as a dockerized application running on a cluster using the Elastic Container Service (ECS) service. This service accesses an Aurora Mysql database schema and a file repository in AWS S3.

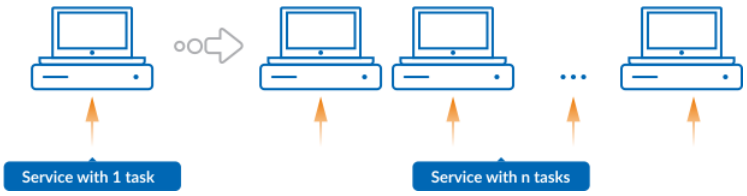
Security

Infrastructure incorporates protection against DDoS attacks using AWS Shield. AWS Shield protects web applications from common web attacks that can impact availability, jeopardize security, or con-

sume excessive resources. Incorporates the AWS Web Application Firewall (WAF). This protection is defined on the Load Balancer with a set of preconfigured rules managed by AWS WAF. AWS WAF includes controls such as protecting against OWASP (Open Web Application Security Project) top 10 security risks.

Scalability

Deyel Cloud implements a dockerized service for each environment that is executed in a cluster service with one or several tasks. The number of tasks is defined using automatic scaling, allocating more resources at times when the environment is most used.



To detect periods of high resource consumption, the CPU usage of the service is analyzed. If it exceeds the expected percentage, additional tasks for the service are automatically initiated. These new service tasks immediately increase the amount of resources, thus generating greater processing capacity. This ensures optimal response times during periods of increased demand. When the service detects less resource use, it automatically decreases the number of tasks. Increase or decrease in the number of service tasks is carried out transparently for the user.

To implement this scaling scheme, Target Tracking Scaling Policies from the AWS ECS service are used.

Types of Deployment

Deyel incorporates different types of deployment to connect with the services of each company. The deployment types define the service access mode based on security requirements.

Backups

Deyel incorporates daily backups of all information and files uploaded in the applications.

Data Encryption

All transactional data is hosted in an AWS RDS service Aurora database and encrypted at rest using the AES-256 algorithm. Data in transit is encrypted using TLS 1.2 / 1.3.

Users can only interact with data via https only through the Deyel Cloud portal web user interface, of the Deyel mobile application or through the Rest api. Users must have login credentials and the appropriate authorization to access the data.

Business Continuity

A Disaster Recovery Plan (DRP) is incorporated to ensure business continuity.

3.11.1. Backups

All the information and all the files uploaded in the applications and modeled objects are considered as data in the environment of **Deyel** Cloud. These data can be uploaded into the environment through the applications modeling, the use of both web and mobile applications, from massive imports, through advanced rules using the SDK or from the integration API.

The uploaded data is stored in an AWS Aurora MySQL service database and in a file repository hosted on the AWS S3 service.

Retention Period

The **Deyel** Cloud service performs a daily backup of the environment data for the last 7 days and a monthly backup for the last 3 months. These backups are performed daily between 3:00 and 6:00 am UTC-3.

Additionally, a backup copy of the environment data can be requested, recovering the information at a given time within the last 24 hours.

The generated backups are stored in an AWS S3 repository that is highly available, encrypted, and data redundant in more than one availability zone. This information is encrypted using AES 256.

Download

To obtain a backup, the complete database and stored files must be downloaded. The database downloading can be done from the daily automatic backup or from a backup generated at a given moment in time.

For the files uploaded to the environment, a compressed file is generated with all the files from the previous day, in this way the files can be downloaded for each working day. If required, the complete download of all existing files can be performed.

Depending on the type of plan contracted, a longer retention period can be modified and the number of monthly downloads increased.

Restore

To restore the information from the database, the operation must be done on a Mysql database as described in the first step of the On Premise Installation.

3.11.2. Disaster Recovery Plan (DRP)

It is essential for a company to guarantee business continuity. To do so, it needs a robust disaster recovery plan that allows it to remain operational in the face of natural disaster or malicious attacks.

The Deyel Cloud infrastructure on AWS mainly bases its DRP strategy on the advantages offered by the AWS services on which it is mounted.

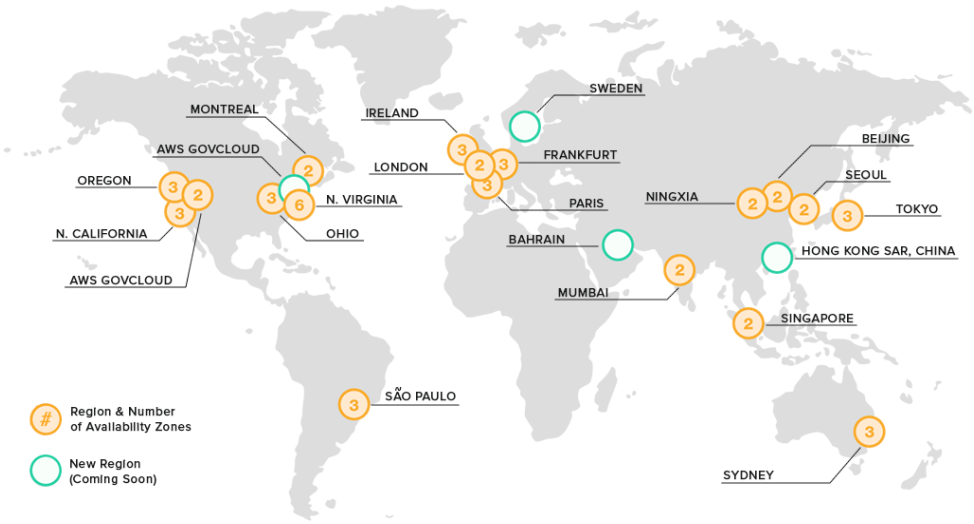
The two main axes of these advantages are the use of regions, availability zones and data centers distributed around the world, and the configuration of the services used to make use of this global infrastructure..

Use of AWS Regions, Availability Zones, and Data Centers

The Deyel Cloud infrastructure is based on AWS regions and availability zones.

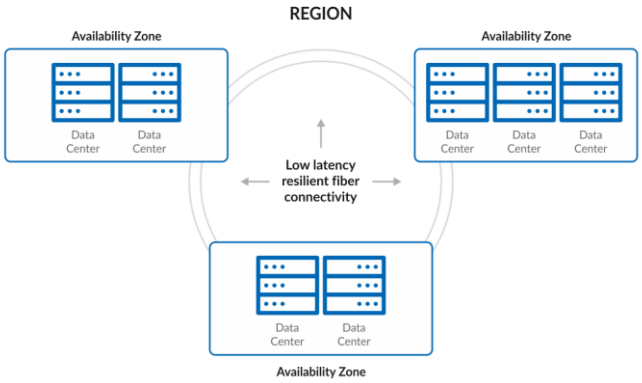
An AWS Region is a physical location in the world, which has multiple Availability Zones. These zones have one or more data centers, each with power, networks and connectivity.

These availability zones provide the ability to operate highly available, fault-tolerant, and scalable database and production applications. AWS has more than 100 availability zones and more than 30 geographic regions around the world.



https://www.infrastructure.aws/

Each AWS region is designed to be completely isolated from the other regions. This allows for greater fault tolerance and stability. Each availability zone is isolated, even though they are connected through low-latency links.



Each availability zone is designed as a separate fault zone, meaning that availability zones are physically separated within a typical metropolitan region and located on low-risk flood plains.

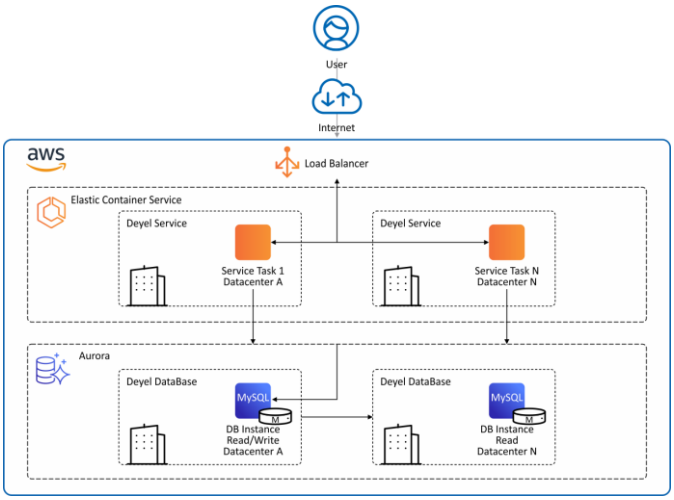
They also have uninterruptible power supplies (UPS) and onsite backups. Availability zones are connected with high-speed links (tier-1).

Configuration of AWS Services Used by Deyel Cloud to Use the Global Infrastructure

AWS services used in Deyel Cloud infrastructure are configured to use regions and availability zones.

Main Components of Deyel Cloud Infrastructure

The main components of Deyel Cloud infrastructure and how the AWS services that support them are configured are detailed below.



- Application data

Application data is stored in Amazon Relational Database (RDS) service Aurora clusters. These clusters are configured with read/write and read instances in different availability zones, which can be immediately swapped in the event of a failover.
To create backups, use the multiAZ option.

- Execution of applications

Applications are executed on AWS Elastic Container Service (ECS) clusters with EC2 instances distributed across availability zones, with autoscaling. Applications are executed on ECS services and can be multitask distributed across EC2 instances in different availability zones.

Clusters used by the Deyel Cloud infrastructure can be created in any AWS region globally in a matter of seconds.

- Critical infrastructure files replicated among regions.
  All the files necessary for assembly and creation infrastructure and for a disaster recovery are stored with AWS S3 and Amazon ECR service, using replication among regions.

- Infrastructure as code (IaC)
  The infrastructure and resources necessary for the execution of **Deyel** on AWS are created using the AWS CloudFormation service. Using this service complies with the good practices of standardizing infrastructure components and allows fast troubleshooting.
  It provides applications resources in a safe and repeatable way, allowing you to create and recreate infrastructure and applications, without having to perform manual actions or write custom scripts.
  Through this service, the Deyel Cloud infrastructure and its applications can be implemented in any AWS region globally.

RPO, RTO and failure events for Standard and Enterprise Editions

Tasks and database with their replica are in different availability zones (in one or more data centers) within a region.

Failure Event	Action	RPO	RTO
The server executing the application goes out of service. Example: hardware failure, motherboard, fonts, disks, etc.	If there are multiple application tasks, the load is shifted to the rest of the tasks. In all cases, an identical task is automatically started on another server in the same data center.	0	In the Enterprise edition the RTO is 0. In the Standard edition the RTO is less than 90 seconds.
The data center that contains the servers executing the application goes out of service service. Example: catastrophe in the data center city.	If there are multiple application tasks, the load is shifted to the rest of the tasks. In all cases, an identical task is automatically started on	0	In the Enterprise edition the RTO is 0. In the Standard edition the RTO is less than 90 seconds.

Failure Event	Action	RPO	RTO
	another server in the same data center.		
The server executing the database goes out of service. Example: hardware failure, motherboard, fonts, disks, etc.	The read-only instance is automatically converted to R/W.	0	In both editions the RTO is less than 90 seconds.
The R/W database goes out of service (structure breaks, engine problems).	The read-only instance is automatically converted to R/W.	0	In both editions the RTO is less than 90 seconds.

Read Replica Instance in a Region Different from the Main One

The read replica instance in a region different from the main one is another option for higher availability and fault tolerance.

When considering catastrophes of global magnitude where an entire region may stop operating, clusters are generated through the AWS CloudFormation service in another region while maintaining use of the interregional database, achieving an active synchronization and a higher fault tolerance based on a better geographic dispersion.

The minimum essential RTO is 3 hours, although customer verification tasks are recommended before enabling the use of the platform again, which can raise the overall RTO.

As an example, to do lists for these cases are detailed.

Tasks	RTO < 8 hours
Assessing the situation, verification of the non-operating regions where the platform was operating.	2hs
Generating the platform infrastructure in the third region.	Less than 30 minutes
Configuring the new infrastructure with the operational database in this region.	Less than 30 minutes
Customer verification of the platform.	Recommended. 3hs

3.11.3. Types of Deployments

Deyel has different options to access its environments, as well as to connect with the company's services.

Each environment is accessed with the “myenvironment.deyel.com” url defined for each environment as its standard access.

Deyel Cloud also allows to define an access with own domain to access with a name in the url “https://mycompany.com”.

Additionally, access with private VPN connections or access restricted by IP address can be defined.

Ways to Access the Environment

An environment can be accessed in different ways, depending on whether to open an application, a page, to do administration tasks or to open the modeler.

To access applications and pages, the url of the environment and its name are used. Additionally, parameters can be sent on pages to define a specific behavior. And to perform administrative tasks or access the modeler, access is done using the global application.

Example of each access

- Access to an application
  https://<*environment name*>.deyel.com/<*application name*>

- Access to a page
  https://<*environment name*>.deyel.com/pages/<*page name*>?<*parameter name*>=<*parameter value*>

  > *If there is more than one parameter, they are concatenated by the sign "&".*

- Access to the administration portal and modeler
  https://<*environment name*>.deyel.com/<*global application name*>

3.11.3.1. Standard Access

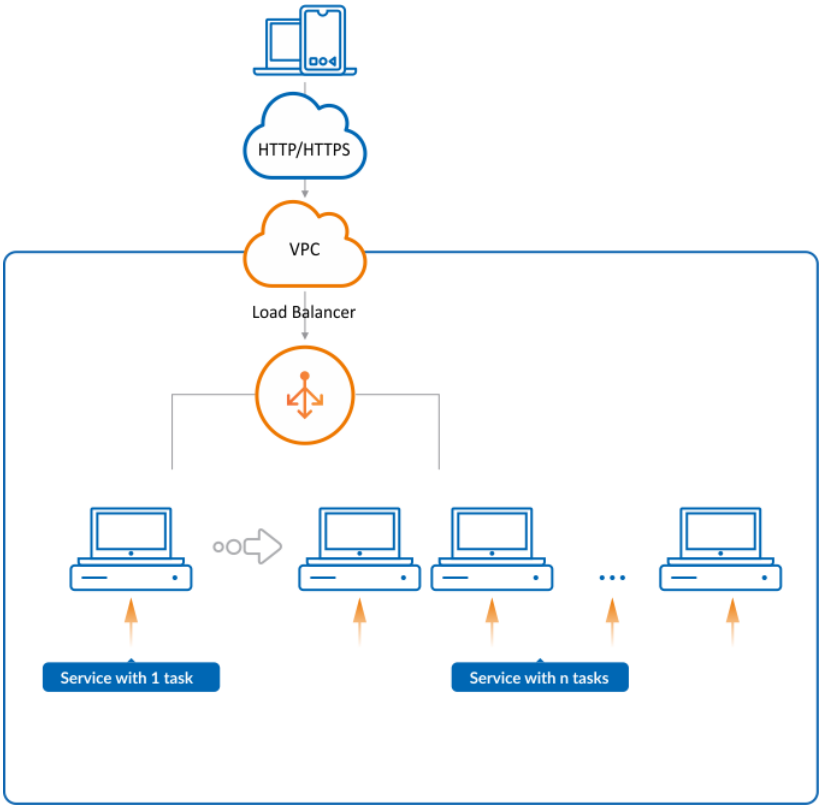
Deyel Cloud is accessed using the https protocol from any client connected to the internet using the url of its environment. If an attempt to log in with http, it is automatically redirected to https.

To access the productive environment, the url "myenvironment.deyel.com" is defined by default and for other environments, a suffix is added after the name of the environment. The suffix for development is "dev" and "test" for the test environment.

Environment URLS

The urls for each environment are:

- For the production environment: https://myenvironment.deyel.com
- For the test environment: https://myenvironmenttest.deyel.com
- For the development environment: https://myenvironmentdev.deyel.com



Using standard access to Deyel Cloud allows to assess the health of the environment.

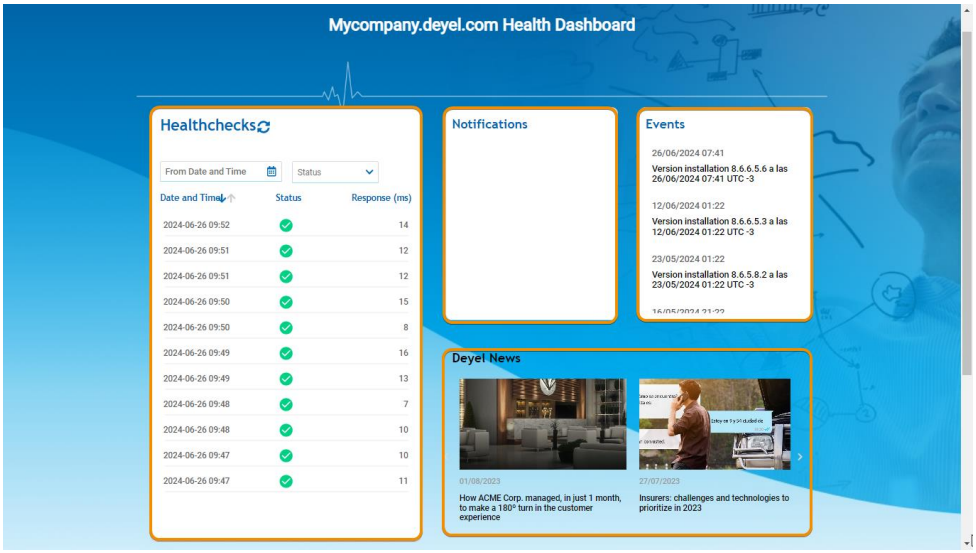
3.11.3.1.1. Health Dashboard

This administration dashboard allows monitoring the state of the environment, by checking health checks automatically performed every 30 seconds. It checks whether the results of these checks were successful and monitors the response time of the environment.

The health checks of the environment can be filtered based on the date and time of checking and by their state.

The information grouped into sections is displayed on the dashboard:

- A grid with the health checks corresponding to the last 30 days
- notifications related to the environment.
- events that occurred.
- updates from **Deyel**.



Dashboard Access

The dashboard is available for all Deyel Cloud environments and can be accessed with the following URL:

- https://portal.deyel.com/pages/health?environment=NAME-ENVIRONMENT

> *NAME-ENVIRONMENT must be replaced with the name of the environment being shown.*

For example, mycompany.deyel.com is accessed as follows:

- For the production environment:
  https://portal.deyel.com/pages/health?environment=mycompany
- For the testing environment:
  https://portal.deyel.com/pages/health?environment=mycompanytest
- For the development environment:
  https://portal.deyel.com/pages/health?environment=mycompanydev

Health Check Grid

The check results are displayed in this grid, they are ordered by date and time in descending order, also their state and the time it took to complete the check.

If the response time of a check is greater than 250 milliseconds, the state is considered erroneous.

As checks are carried out every 30 seconds, the icon can be used to update the grid.

The results can be filtered by date and time of checking or by its state, and can be sorted in ascending or descending order by clicking the icon .

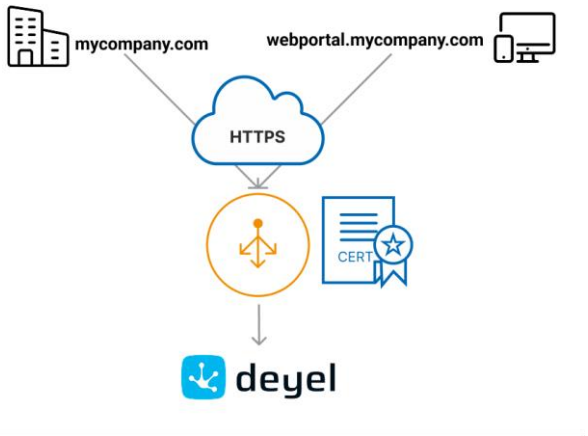
Notifications, Events and News

Version update or maintenance task notices are displayed in the notifications section. Version updates are performed automatically and do not affect the operation of the environment, while maintenance tasks are notified in advance as they may require downtime.

The updates made and the results of the maintenance tasks executed are displayed in the events section, while, in the news section, there are success stories of **Deyel** and information on new versions with their highlighted functionalities, among other topics.

3.11.3.2. Access with Own Domain

Deyel allows configuring access to the production environment through its own domain or subdomain, previously registered and managed by the client. This replaces the generic URL with a secure, customized address, thanks to the installation of a valid SSL certificate.

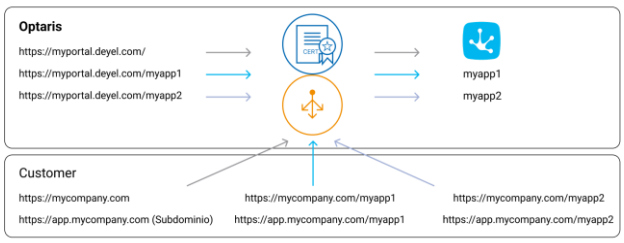


Access to Applications and Pages

Access to applications and pages is detailed below.

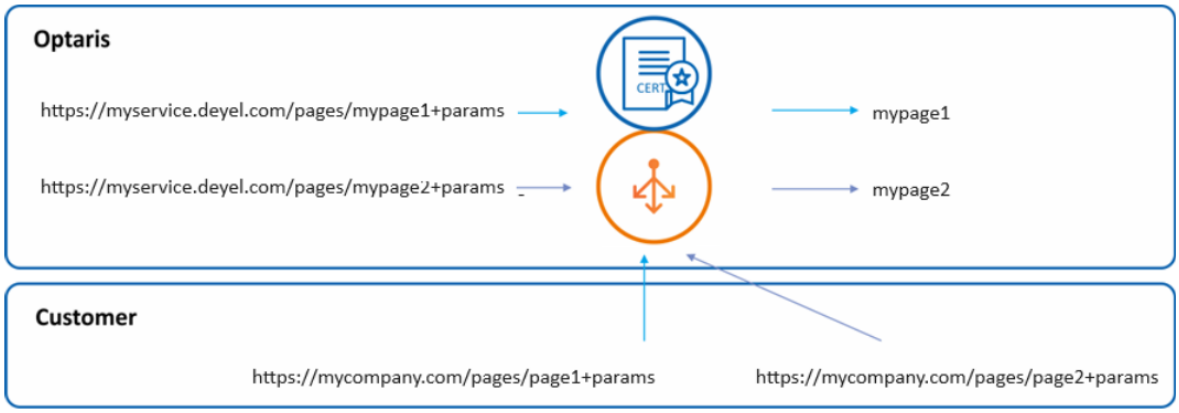
Access to Applications

Each application is accessible through a URL composed of the defined domain or subdomain, followed by the name of the application as modeled in **Deyel**.



Access to Pages

To access the pages, the domain name, then /pages/ with the name of the page and its parameters can be used, for example "mycompany.com/pages/mypage1?param1=value1&param2=value2".



Configuration

Linking a custom domain requires coordination between the domain owner and the technical team of **Deyel**.

Tasks in Charge of the Domain Owner

- Send to the technical team of **Deyel** the domain or subdomain you want to link.
- Upon receiving the .csv file with the technical data, configure two CNAME records in the DNS: one to validate the SSL certificate (temporary use) and another to link the domain with **Deyel** (permanent use), as follows:

  Record 1 – SSL certificate validation CNAME (temporary use).

  This record allows you to validate the ownership of the domain to issue the SSL certificate managed by the technical team of **Deyel**.

  - Domain Name: domain or subdomain to be linked
  - Record Name: generated by the technical team
  - Record Type: CNAME
  - Record Value: generated by the technical team

This record must remain active until the technical team of **Deyel** confirms that the SSL certificate was issued correctly.
Once this confirmation is received, the owner can delete this record without affecting the operation of the domain.

Record 2 - CNAME resolution to Deyel (permanent use).

This record ensures that the domain points correctly to the load balancer of **Deyel**.

- Domain Name: domain or subdomain to be linked
- Record Name: @ or the desired subdomain
- Record Type: CNAME
- Record Value: Load balancer DNS of **Deyel**

This record must be kept active at all times to ensure proper access to the platform.

Tasks in Charge of the Technical Team of Deyel

- Upon receiving the domain name, validate that it is registered, active, and allows DNS configuration.
- Generate and send the .csv file (AWS Certificate Manager) the domain owner with the data necessary to configure both CNAME records.
- Confirm SSL Certificate Issuance: Once AWS has validated the verification CNAME record and issued the certificate, notify the domain owner that they can delete the temporary validation record.

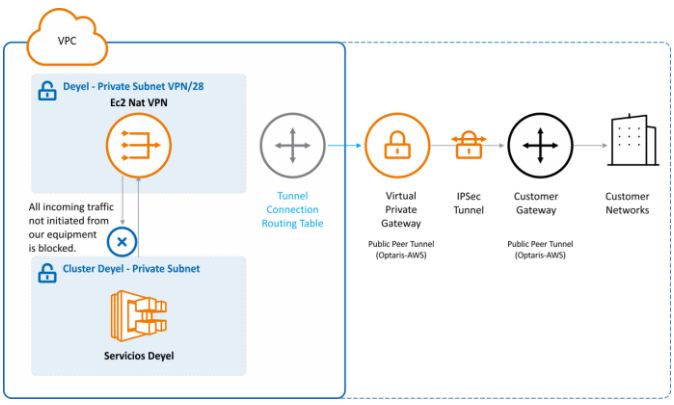
3.11.3.3. Access with Private Connections (VPN)

Deyel Cloud allows to establish "site-to-site" VPN connections to access from **Deyel** to web services located on customer's private network servers. The VPN service includes initial configuration, connectivity testing and technical support.

To use the VPN service, the customer needs a compatible VPN device located at their corporate premises, with an assigned public IP (IPv4) address and with the capabilities to be configured using the IPsec protocol.

Deyel uses the AWS Virtual Private Gateway service to establish IpSec tunnels with its customers and thus enable access to its web services, for example by performing integrations when using a Rest API.

This service enables to establish a "site-to-site" tunnel between the customer and the AWS Virtual Private Gateway service of **Deyel** with the Internet Key Exchange version 2 (IKEv2) protocol.



The configuration of this service supports different types of cryptography:

AES 256-bit encryption, SHA-2 hashing.

Diffie-Hellman groups:

Phase 1 can now use DH groups 2, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24

Phase 1 encryption algorithms AES128, AES256, AES128-GCM-16, AES256-GCM-16
Phase 1 integrity algorithms SHA-1, SHA2-256, SHA2-384, SHA2-512

Phase 2 can now use DH groups 2, 5, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24
Phase 2 encryption algorithms AES128, AES256, AES128-GCM-16, AES256-GCM-16
Phase 2 integrity algorithms SHA-1, SHA2-256, SHA2-384, SHA2-512

Customer Data

- Customer Gateway: Public customer Ip where to establish the tunnel.
- Local Customer IPv4 Network Cidr: (Destination network/IP of the customer to connect privately).
- Pre Shared Key: To establish encryption.

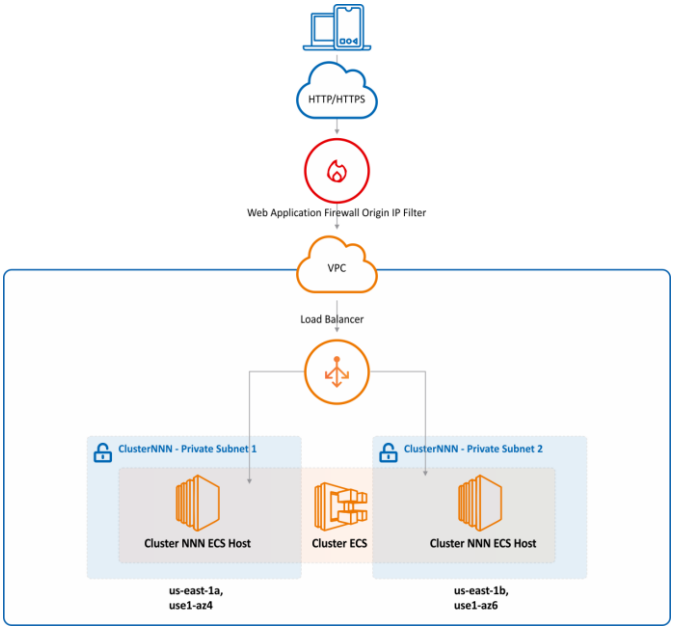
Deyel Data

- Outside IP Address: Public IP address of **Deyel** provided by AWS for "tunnel 1-2", from where the tunnel is established.
- Local IPv4 Network Cidr: Private network/private IP within the VPC where the customer receives connections privately *Network/IP of the EC2 Nat-VP.

3.11.3.4. Access Restricted by IP Addresses

Deyel Cloud offers the possibility of protecting access to its service by restricting it to only a specific IP address list. To perform this implementation, public access can be blocked using custom IP origin filters and thus control the access points allowed using the https protocol.

Once the allowed access IP addresses are configured, users attempting to connect to **Deyel** from outside of these IP addresses receive an error message, indicating that they must connect to the site through a VPN connection or from the company's corporate network.



3.12. On-Premise Installation

The following describes the steps to perform a new installation of **Deyel** or a version update in On-Premise modality.

Distribution

Deyel and its solutions are distributed through the following files:

Description	File
Web application	• deyel-**Version**.war • Docker image and the docker-compose file
Initial database	• Oracle: deyeldb-**Version**.dmp • Mysql: deyeldb-**Version**.sql • Sql Server: deyeldb-**Version**.bak • DB2: deyeldb-**Version**.zip
License	license.xml

> ***Version***
> *Each version of **Deyel** has the following V.R.F.B. structure where*
> *V - Version*
> *R - Release*

> *F - Fix*
> *B - Build*
> *Example: deyel-7.8.1.3.war*

Step 1: Create the Initial Database for Deyel

a. Install the database engine on which you want to install **Deyel**.

b. Create a user with permissions that can create tables, views, functions, store procedures, and triggers. The following variables must be known:

- **host**: engine ip address.
- **port**: engine port.
- **user**: user with the necessary permissions
- **password**: key password of the user with the necessary permissions

c. Restore the initial base of **Deyel** with the following commands:

- Oracle
  ```
  impdp user/password@host:port/deyel dumpfile=deyeldb-Version.dmp
  ```
- Mysql
  ```
  mysql -u user -p password -h host --port port --default-character-set=utf8 deyel < deyeldb-Versión.sql
  ```
- Sql Server
  ```
  SqlCmd -S host -U user -P password -Q "RESTORE DATABASE deyel FROM DISK ='deyeldb-Version.bak' WITH REPLACE"
  ```
- DB2
  Unzip deyeldb file-***Version***.zip
  Execute command from the folder where export files are located.
  ```
  db2move deyeldb import -io replace_create > restore_deyel.log
  ```

Step 2: Install Deyel on the Application Server

- Installation in Apache Tomcat
- Install in Docker
- Installation in AWS

Step 3: Configure Database

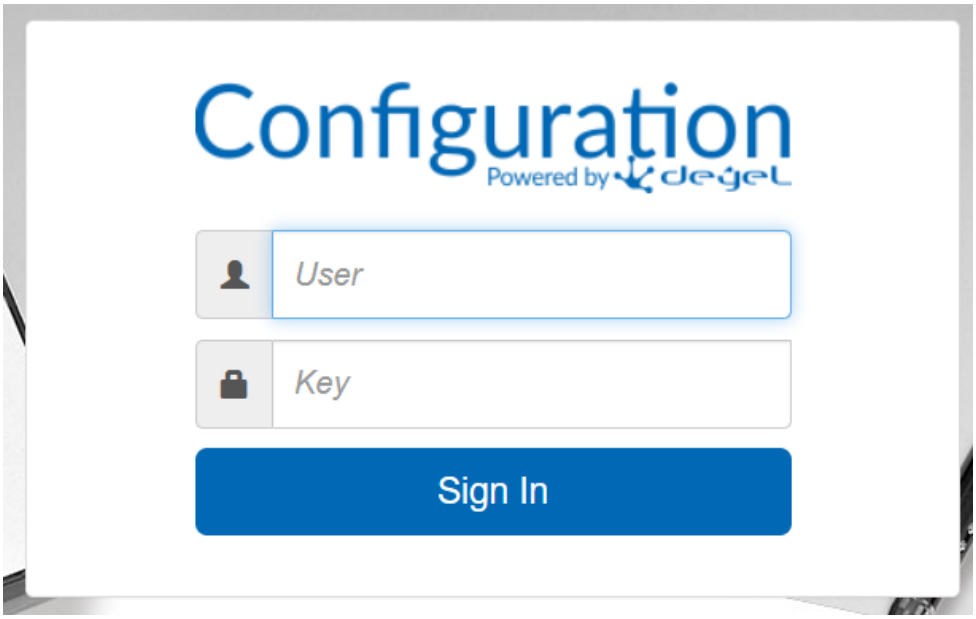
To configure the database, it is recommended to use the environment variable DEYEL_DB_PROPERTIES_PATH. This variable is used to define the directory where the file with the connection data is located. If the environment variable is not defined, the file is used within the WEB-INF/classes/ConsistEnv_es_AR.properties context.

The environment variable should be created before performing the context implementation.

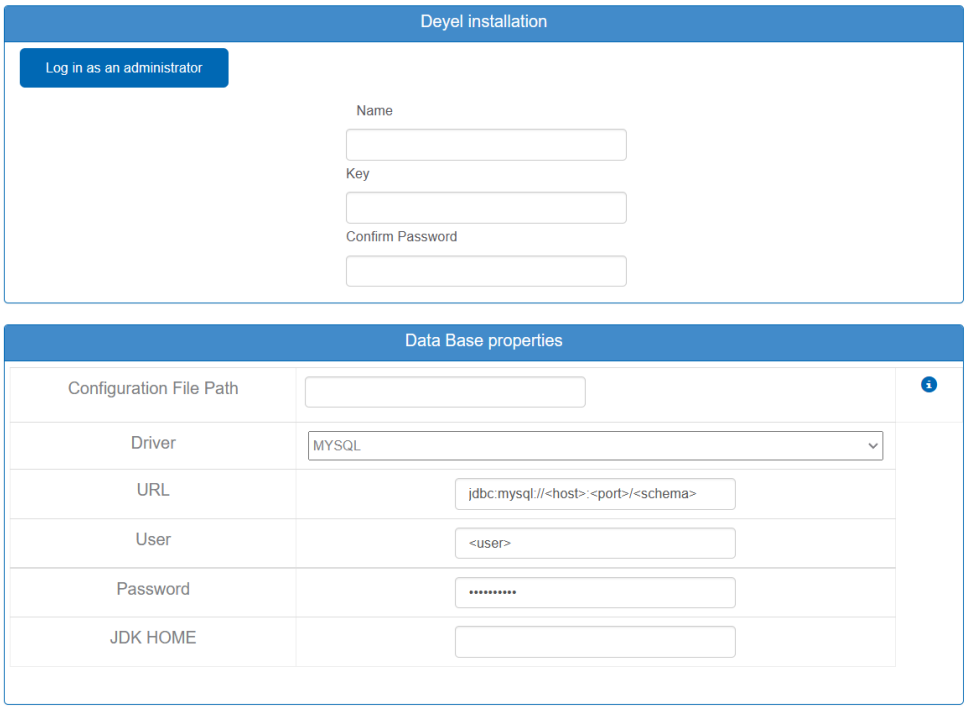
Instructions for Database Configuration

a. Enter https://**server:port**/deyel/Config.

> *Server: is the name of the server where the installation is performed or its IP address and access port.*
> *Example: http://**myteam**:8080/deyel*



> *It is accessed without username and password until one is defined. It is recommended to define a username and password when first accessing.*



> *Include the Unicode and Encoding parameters in the connection URL. Example with corresponding values:* ***jdbc:mysql://cluster-db-My8:3306/Deyel-schema?useUnicode=true&characterEncoding=latin1***

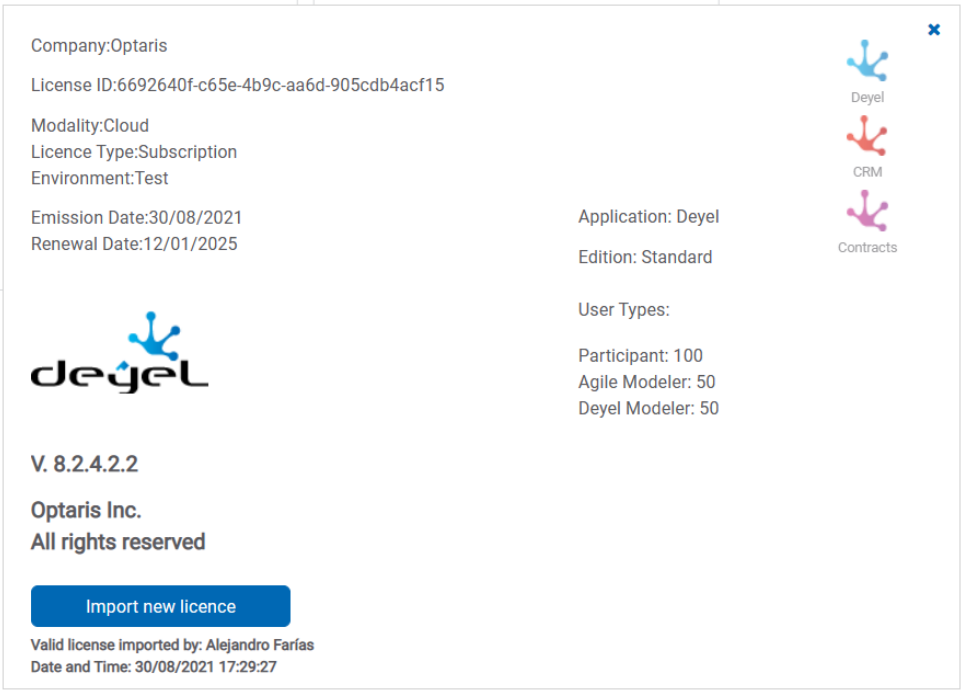
b. Complete the data of the Database Properties section and apply configuration. If the DEYEL_DB_PROPERTIES_PATH environment variable was defined, in the Configuration File Path property, the directory where the file with the connection data is located is shown. If this property is not reported, this means that the file is in the context.

c. Restart the context.

Step 4: Import License

a. Enter https://**server:port**/deyel.

b. Import the use license property is enabled.



c. Enter the **Deyel** environment with https://**server:port**/deyel using the example username "afarias" with "deyel123" key password.

Step 5 (optional): Configure the connection pool

Once the environment is licensed, the minimum and maximum number of connections the pool uses to communicate with the database can be configured.

Go to https://server:port/deyel/Config to access the following configuration menu:



- **Maximum connections**
  Sets the maximum number of simultaneous connections the pool can keep open. This parameter is key to managing system resources and preventing database overload.

- **Minimum connections**
  Defines the minimum number of connections that the system keeps open at all times. Properly adjusting this value enhances overall performance by ensuring the availability of connections based on the environment's demands. To determine an optimal value, the ConnectionsPool log available in the environment can be shown, where used connections are recorded.

By default, **Deyel** configures the connection pool with a maximum of 30 and a minimum of 2 connections.

Considerations to Keep in Mind

- If the application entities use files with a preview view, the workstations must have access to the *.google.com domain.
- If the network blocks this domain, the preview won't work because it relies on Google's "gview" library.

3.12.1. Directories Structure

Once the implementation of the file with war extension has been done, the following directory structure is defined.

Directory	Description
batch • log	Directory with Java classes that allow the execution of batch processes. • log: Directory where the result of the execution of classes is saved.
businessRuleDeployment	Directory where files generated by business rule implementations are temporarily stored.
businessRuleXML	Directory where business rules definitions are temporarily stored.
docGenerator • app • staticHTML • themes • themesDefault • themesResponsive • themesResponsiveMin	All files related to forms definition are stored. • app: Where the generated documents are located. Deprecated. • staticHTML: Basic files used to compose screens. • themes / themesDefault / themesResponsive / themesResponsiveMin: They correspond to the themes that are used when modeling forms.
eventFiles	Temporary directory of files linked to events that occur in processes (mails, File Reader files, etc.).
export	Temporary directory where the exported files are stored.
Import	Temporary directory where the imported files are stored.
integrationDeployment	Directory to temporarily store files generated by integration rules implementations (SQL and web services).

Directory	Description
integrationXML	Directory to temporarily store integration rules definitions (SQL and web services).
jasperReportFiles	XML files containing Jasper Report framework templates used in report printing and printing in general.
logs	Storage directory of **Deyel** logs. When installing, log types and the number of days they are kept can be specified.
META-INF	Information of **Deyel** version.
modeler version	Files with images, JavaScript, and web modeler styles.
script	JavaScript files that are used in the web portal.
Temp	Temporary directory of **Deyel**.
themes	Theme storage directory of **Deyel**.
updaters	Directory containing the files necessary to update the version of **Deyel**.
upload	Directory to temporarily store files uploaded to **Deyel**.
vendors	Directory of libraries used by **Deyel**.
WEB-INF	Storage directory of classes that define the operation of **Deyel**.
widget	Directory with templates of widgets used by the web portal.

Temporary directories (Temp, BusinessRuleDeployment, BusinessRuleXML, EventFiles, Export, Import, IntegrationDeployment, IntegrationXML, and Upload) are automatically removed every 24 hours. The 24 hours are calculated from the start time of the scheduled tasks.

3.12.2. Installation in Apache Tomcat

One of the options to install **Deyel** on an application server is under Apache Tomcat.

Apache Tomcat Configuration for Deyel

On Linux, Windows and on Dockerized Installations

The following minimum values can be defined in the setenv.sh file inside the Apache Tomcat bin folder:

- `XX:+UseConcMarckSweepGC`
- `Dfile.encoding=ISO-8859-1`

Use of Special Characters, Encoding and Post Size

Configure the following files with the lines indicated for each case.

Apache Tomcat 8.0

- conf/catalina.properties
  Include the line:
  `tomcat.util.http.parser.HttpParser.requestTargetAllow=|`

- conf/server.xml

```
<Connector port="8080" protocol="HTTP/1.1"
 connectionTimeou="20000"
     maxHttpHeaderSize="32768"
     maxPostSize="-1"
     redirectPort="8443"
     URIEncoding="UTF-8" />
```

Apache Tomcat 8.5 and higher

- conf/server.xml

```
<Connector port="8080" protocol="HTTP/1.1"
  connectionTimeout="20000"
  maxHttpHeaderSize="32768"
  maxPostSize="-1"
  redirectPort="8443"
  URIEncoding="UTF-8"
  relaxedPathChars ="[ \ ] { | }"
  relaxedQueryChars ="[ \ ] { | }"
  URIEncoding="UTF-8"
  compression="on"
  compressibleMimeTy-
  pe="text/html,text/xml,text/plain,text/css,text/javascript,application/j
  avascript,application/json,application/xml"
  compressionMinSize="2048"
  useSendfile="off"
  relaxedPathChars="[ \ ] { | }"
  />
```

> *The maximum required post size is disabled with the maxPostSize=parameter"-1" to save or publish processes.*

Clickjacking protection, HSTS header force and browser content sniffing

Enable the HTTPHeader security filter from Apache Tomcat in the conf/web.xml file. For Tomcat 8 and higher installations it should be:

```
<filter>
        <filter-name>httpHeaderSecurity</filter-name>
      <filter-
class>org.apache.catalina.filters.HttpHeaderSecurityFilter</filter-
class>
        <async-supported>true</async-supported>
        <init-param>
            <param-name>antiClickJackingOption</param-name>
            <param-value>SAMEORIGIN</param-value>
        </init-param>
        <init-param>
            <param-name>hstsEnabled</param-name>
          <param-value>true</param-value>
        </init-param>
</filter>

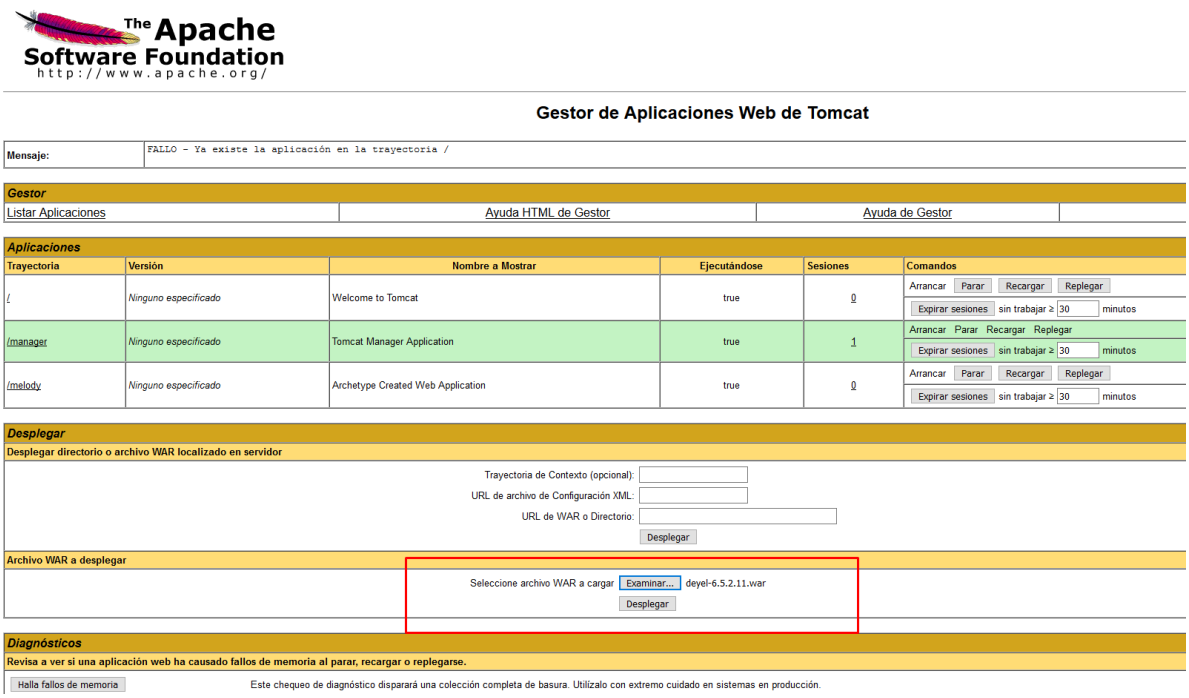
 <filter-mapping>
        <filter-name>httpHeaderSecurity</filter-name>
        <url-pattern>/*</url-pattern>
        <dispatcher>REQUEST</dispatcher>
</filter-mapping>
```

War file implementation

In order to implement in a context with the name "deyel", the file should be renamed with the extension war, assigning it the name "deyel.war".

Apache Tomcat has a default maximum for uploading files with a war extension of 50Mb, it should be increased to 500Mb in the <TOMCAT>\webapps\manager\WEB-INF\web.xml file.

```
52      -->
53      <multipart-config>
54        <!-- 500MB max -->
55        <max-file-size>524288000</max-file-size>
56        <max-request-size>524288000</max-request-size>
57        <file-size-threshold>0</file-size-threshold>
58      </multipart-config>
59    </servlet>
60    <servlet>
61      <servlet-name>Status</servlet-name>
```



Continue with installation step 3

3.12.2.1. Version Update

The following describes the steps to update the version of **Deyel** on On-Premise installations.

Distribution

The version update of **Deyel** and its solutions are distributed through the following files:

Description	File
Web application	deyel-**Version**.war

> ***Version***
> *Each version of **Deyel** has the following V.R.F.B. structure where*
> *V - Version*
> *R - Release*
> *F - Fix*
> *B - Build*
> *Example: deyel-7.8.1.3.war*

Step 1: Do Backup of Existing Database

- Oracle
  ```
  expdp deyel/password@host:port/deyel dumpfile=deyeldb.dmp schemas=deyel
  ```

- Mysql
  ```
  mysqldump -u root -p -h host --routines --skip-add-locks --single-
    transaction deyel > deyeldb.sql
  ```

- Sql Server
  ```
  BACKUP DATABASE deyel TO DISK = 'deyeldb.bak'"
  ```

- DB2
  ```
  db2move deyeldb export -aw -sn deyel > deyel_export.log
  ```
  The export files remain in the folder where the command is executed.
  A compressed file called deyeldb.zip should be generated with all the generated files.

Step 2: Do Backup of the Existing Context

- Compress all files in the existing context and move them out of the application server.
- Perform "undeploy" operation of the existing context on the application server.

Step 3: Install Deyel on the Application Server

- Installation on Apache Tomcat
- Installation in WAS

Step 4: Configure Database

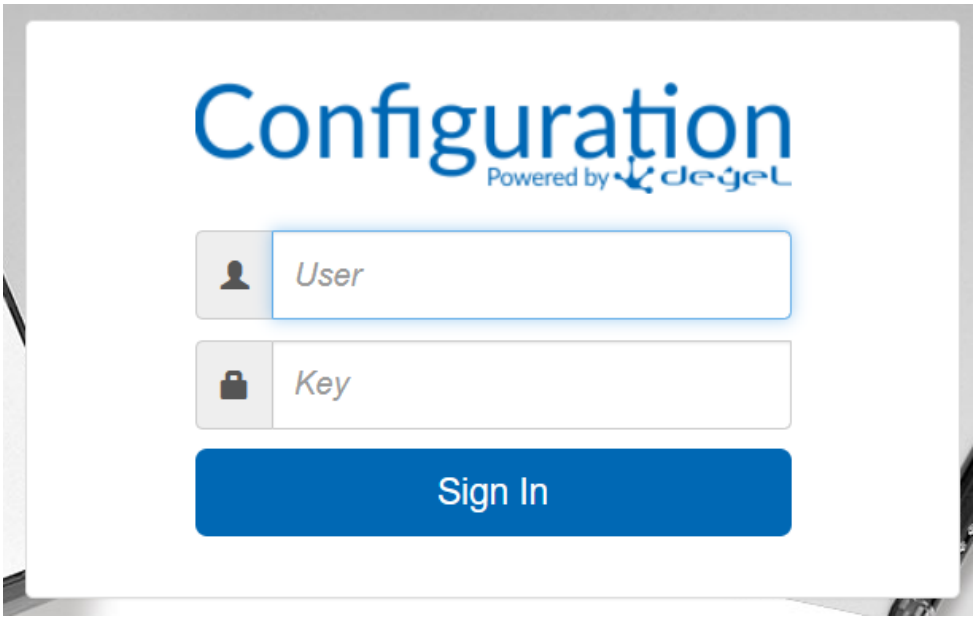
If the DEYEL_DB_PROPERTIES_PATH environment variable was defined with the directory where the file with connection data is located, it is not necessary to perform this step.

If the DEYEL_DB_PROPERTIES_PATH environment variable was not defined, the database properties configuration file is within the context. In this case, the configuration of **Deyel** should be entered so as to complete the values for the connection properties again.

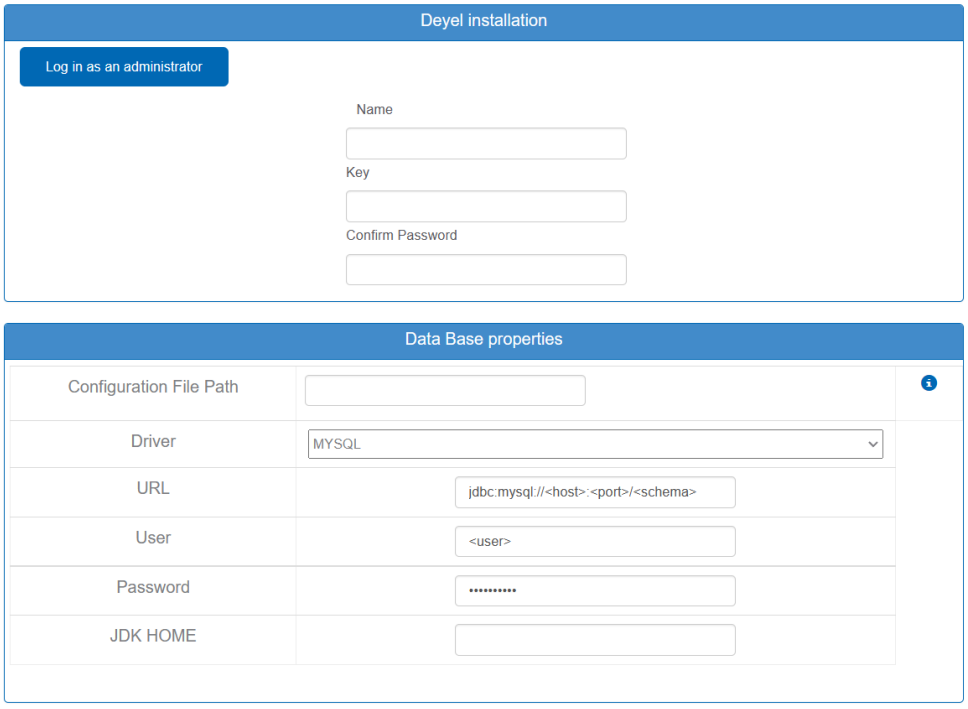
Instructive

a. Enter a https://**server:port**/deyel/Config.

> *Where server: server name where the installation is performed or its IP address and access port.*
> *Example: http://**myteam**:8080/deyel*



It is accessed without username and password until one is defined. It is recommended to define a username and password when first accessing.

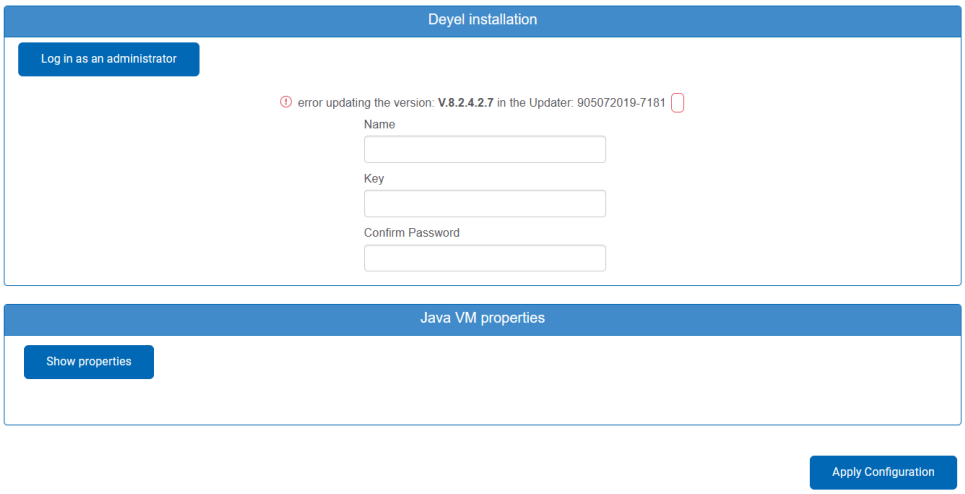


b. Complete the data of the Database Properties section and apply configuration.

c. Restart the context.

Step 5: Check if the Update is Successfully Performed

Enter a https://**server:port**/deyel/Config and check for any errors.



If an error occurred while updating, do the following:

- Download the log of the update that fails and the complete log of the day (SystemOut*.log) to send them to the **Deyel** team.
- Restore the backed up database in step 1.
- Stop the application server and copy all the context files backed up in step 2.

3.12.3. Installation in Docker

Below are the steps to install and update the version of **Deyel** using the docker-compose file.

Initial Installation

Step 1: Download the docker-compose file

- Ask the **Deyel** team for the credentials to download the image from AWS ECR and the link to download the backup with the initial database.
- Install AWS CLI following the installation instructions.
- Execute the following statements to login and download the docker image:

```
- aws configure
```

Enter the credentials to access the AWS ECR repository

```
-aws ecr get-login-password | docker login --username AWS --password-
stdin 955768960522.dkr.ecr.us-east-1.amazonaws.com
```

- `docker pull 955768960522.dkr.ecr.us-east-1.amazonaws.com/deyelonpremise:<8.1.0.5.3>`

  If the suffix indicates the version to install, it must be replaced by the corresponding one.

Step 2: Set the directory to save the installation properties file

Create the folder on the server where docker runs to define the volume, for example docker compose uses the ./app-conf folder.

Step 3: Set directory to save log files

Create the folder on the server where docker runs to define the volume, for example docker compose uses the ./app-logs folder.

Step 4: Execute:

```
docker-compose up -d
```

Continue with installation step 3

Version Update

In each delivery, the docker image is delivered to carry out the installation and the docker-compose file must be modified, replacing the name of the docker image (image:) using the downloaded version.

Then the update must be carried out by executing:

```
docker-compose up -d
```

3.12.4. Installation on AWS

The steps below are for installing a service on ECS or EKS within AWS services.

Service Properties

These are the properties that must be defined for the service to work correctly:

Parameter	Value	Description
CLIENT	myenvironment	Name of the environ-

Parameter	Value	Description
		ment. It is used to recover the secret.
MODALITY	cloud	If a secret is used, the value cloud must be set.
REGION_AWS	us-east-1	AWS region where to search for the secret.
INTERFACE_AWS	10.110.*.*	Network segment for tasks of the same Deyel service to communicate.
TAG_KEY_AWS	Name	Key that allows identifying the EC2 instances within the cluster.
TAG_VALUE_AWS	EC2-Server1	Value assigned

Parameter	Value	Description
		to the TAG_KEY_AWS key to locate the EC2s.
USE_HAZELCAST	true	Required value.
SQL_DRIVER	com.mysql.jdbc.Driver	JDBC driver used to connect to the database.
SQL_URL	jdbc:mysql://mydatabase.amazonaws.com:3306/myenvironment?useUnicode=true&characterEncoding=latin1	Connection URL to MySQL/Aurora database.
SQL_USER	myenvironment	User to access the database schema (if a secret is not used).
SQL_PASSWORD	mykey	Database user password (if a secret is not used).

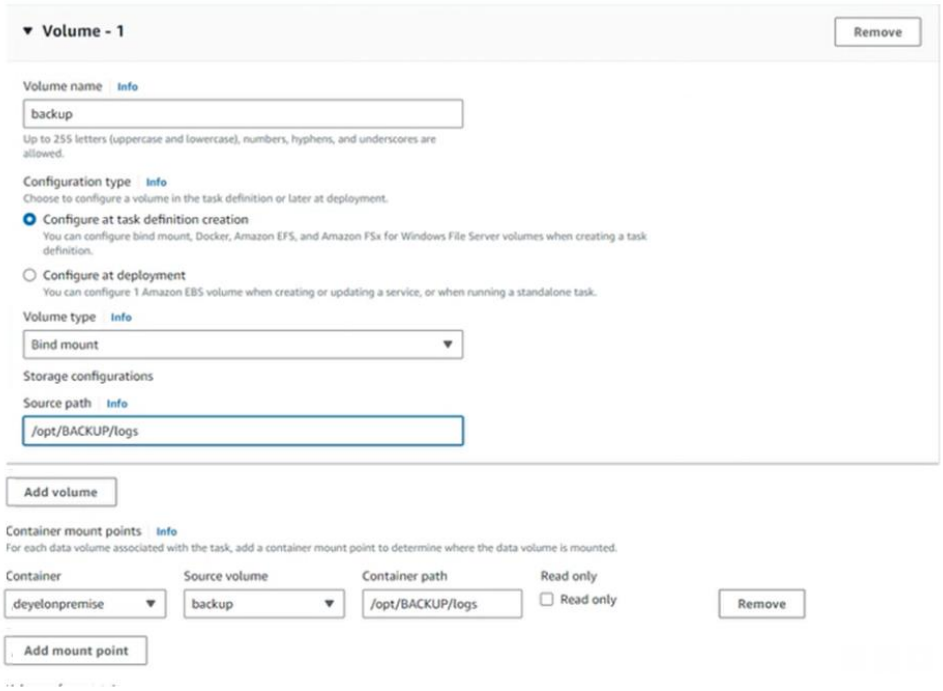
Log Configuration

In the property_value table of the database, configure the following properties:

Property	Value Example	Description
LOG_DIRECTORY	/opt/BACKUP/logs/myenvironment	Path where the logs generated by the application will be stored.
LOG_DIRECTORY_DOWNLOAD	/opt/BACKUP/logs/myenvironment	Path to download the generated logs.
LOG_OUTPUT	/opt/BACKUP/logs/myenvironment	Defines that logs will be written to a file.

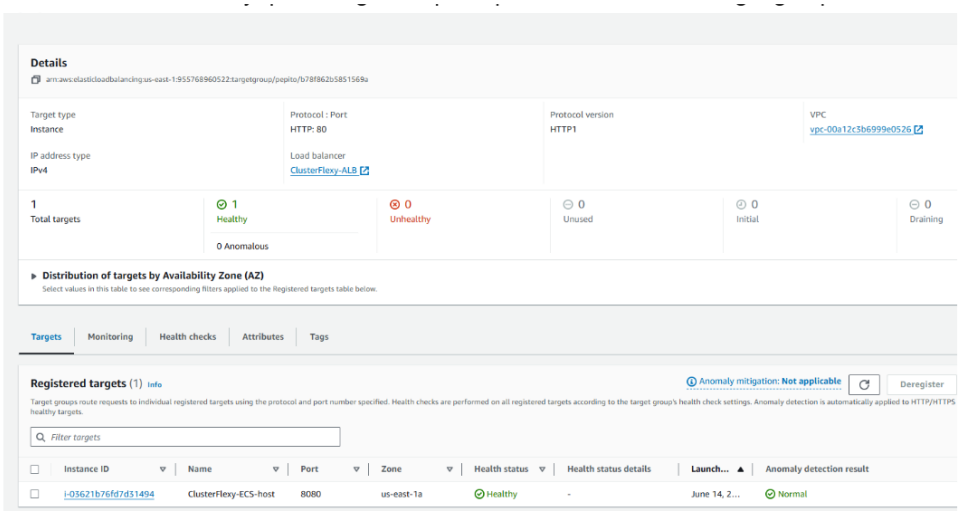
Configure logs in an EFS (Elastic File System) folder:

- On the EC2s of the cluster, define the path to the folder in EFS that will store the logs.
- In the cluster tasks, mount this folder so that the logs are saved correctly in EFS.

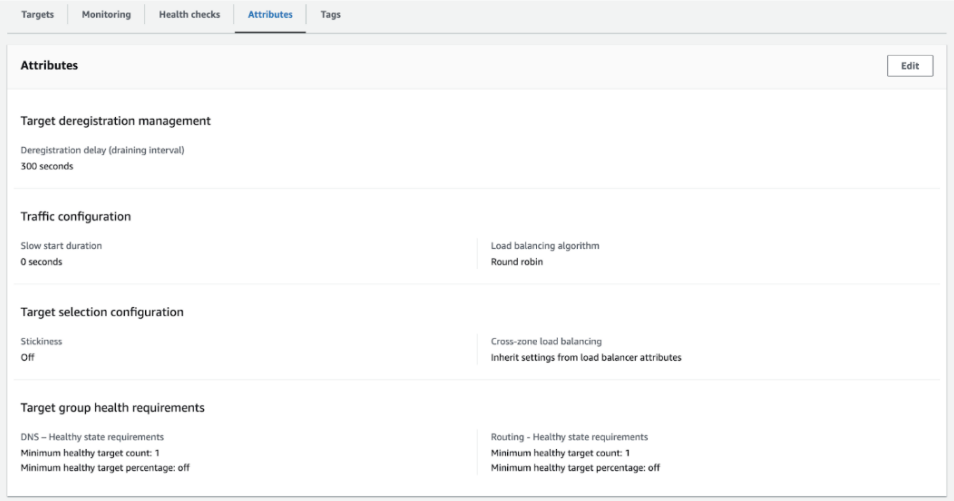


Configure the Target Group

Access to the service must be configured to go through the Target Group on port 8080, as shown in the image.

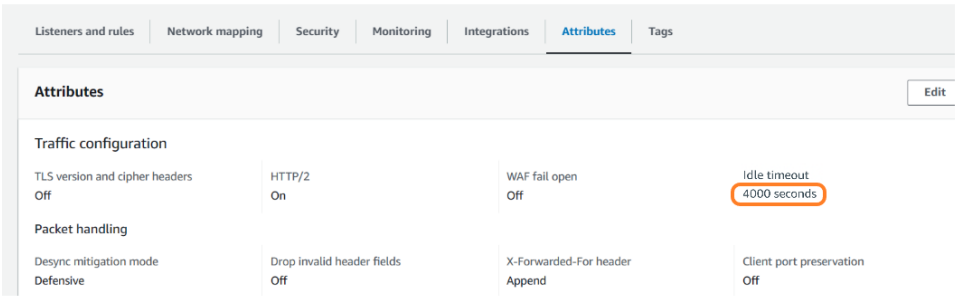


The attributes shown in the image must be configured.



Configure the Application Load Balancer (ALB)

Access the Load Balancer (ALB) and set the IDLE TIMEOUT parameter to 4000 seconds.
This setting is important to prevent connections from being automatically closed due to inactivity.



Configure the HTTPS Certificate

- If the certificate is public:
  Use AWS Certificate Manager (ACM) to request the certificate.
  Configure the certificate previously uploaded to ACM on the Application Load Balancer (ALB) to encrypt HTTPS connections.

- If the certificate is private (for restricted access):
  A script is responsible for installing the certificate when the Deyel service container is started.
  This script must be located in a folder mounted inside the container.

Steps to Configure the Certificate in the Container

1. **Define Volumes in Docker Compose**
   A volume must be defined in the docker-compose.yml file where the certificate installation script is located.
   Example:

```
volumes:
      - './app-logs:/opt/tomcat/webapps/ROOT/logs'
      - './app-scripts:/optaris/userScript'
```

2. **Location of the startup.sh script**
   The startup.sh script must be present in the app-scripts folder on the host server.
   This script will be executed when the container starts, before starting the Tomcat server.

3. **Contents of the startup.sh Script**
   The script must include the installation of the certificate in the container as follows:

```
#!/bin/bash
echo "$CERT_PATH" > /tmp/ca.crt
$JAVA_HOME/bin/keytool -importcert -file /tmp/ca.crt -alias ca-root-
banesco -keystore $JAVA_HOME/lib/security/cacerts    -storepass chan-
geit -noprompt

exec $TOMCAT_HOME/bin/catalina.sh run
```

Aurora MySQL 8 Database Configuration/Creation

When creating the database in Aurora, it must be configured so that table names are case-insensitive.

> *This parameter can only be set at cluster creation time.*

Steps to create the Aurora database

1. **Create new Parameter Groups**
   Create a DBClusterParameterGroup and a DBParameterGroup based on the family:
   Aurora MySQL - aurora-mysql8.0

2. **Modify the created DBClusterParameterGroup**
   Set the following values:

```
character_set_client: latin1
character_set_connection: latin1
character_set_database: latin1
character_set_filesystem: latin1
character_set_results: latin1
character_set_server: latin1
innodb_file_per_table: '0'
lower_case_table_names: '1'
long_query_time: '1'
slow_query_log: '1'
```

3. **Modify the created DBParameterGroup**
   Set the following values:
   innodb_print_all_deadlocks: '1'
   log_bin_trust_function_creators: '1'
   log_output: FILE
   max_allowed_packet: '1073741824'
   sql_mode: 'STRICT_TRANS_TABLES,NO_ENGINE_SUBSTITUTION'

4. **Create a new Aurora MySQL database**
   Use the Parameter Groups created previously.
   `Engine: 8.0.mysql_aurora`

3.12.5. High Availability

A high availability installation consists of having two or more installations of **Deyel** connected to each other, sharing the same database. Each of these installations is identified as a member and they all form a cluster, which is necessary to have a high availability installation.

Each member of the cluster includes internally a software that allows a high availability operation called Hazelcast. To make these members know each other and guarantee high availability, it is necessary to carry out a certain configuration.

Steps for the Configuration of each Member

Step 1: Activate the "HAZELCAST_USE_XML" property in the shared database

This property allows the configuration of Hazelcast in XML mode to specify the different ways of connecting among members.

To activate the property, it is necessary to include it in the "property_value" table using the following sentence:

```
insert into property_value (CD_PROPERTY, CD_LEVEL, CD_LEVEL_VALUE,
DS_VALUE, CD_USER_STORE, DT_STORE, CD_USER_UPDATE, DT_LAST_UPDATE) va-
lues('HAZELCAST_USE_XML','OWNER','0','true',NULL,NULL,NULL,NULL);
```

Step 2: Configure the Hazelcast XML file

Within the installation of each cluster member, the XML file "hazelcast.xml" located in the path <miInstalacion>\ WEB-INF\classes\ should be configured.

Its configuration consists of:

- Set the name of the cluster for each member to connect to, optionally the password can also be specified.

```
...
<hazelcast xmlns="http://www.hazelcast.com/schema/config"
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xsi:schemaLocation="http://www.hazelcast.com/schema/config
           http://www.hazelcast.com/schema/config/hazelcast-config-3.11.xsd">

    <group>
        <name>cluster</name>
        <password>cluster-pass</password>
    </group>
```

- Configure member discovery on the network. There are several ways to do this, such as multicast, TPC-IP, etc. You can choose the most convenient way for each installation, following the recommendations of the Hazelcast guide on their site

  https://docs.hazelcast.org/docs/3.11.7/manual/html-single/index.html#setting-up-clusters

Step 3: Restart

Once configured, members should be restarted so that they can connect to each other. To verify this, the application servers log should be shown and the connection of members should be verified.

> *Each member is considered to be installed on a full application server.*

Example: In Tomcat, the cluster can be shown in the "catalina" log, visualizing the connection of 6 members.

For more details of Hazelcast configurations, show the documentation of its site https://docs.hazelcast.org/docs/3.11.7/manual/html-single/index.html.

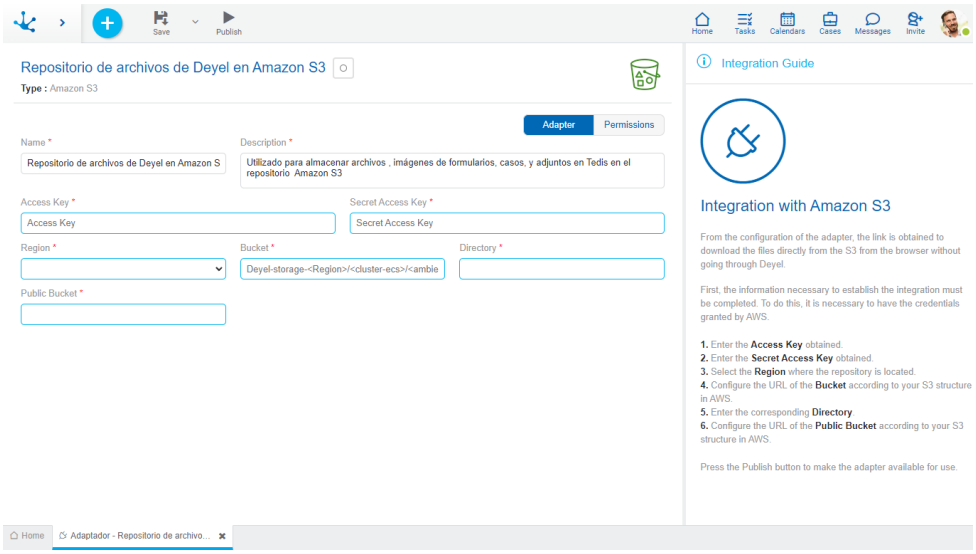
3.12.6. Amazon S3 Adapter

Integration with this adapter is available in installations with On-Premise licensing.

Having an Amazon account, the AWS S3 service can be used. This service allows to store the files in the repository and download them directly from the form that uses them.

Properties

In addition to the properties shared by adapters, such as Name and Description, there are also properties specific to the file repository of **Deyel** on Amazon S3.



> *An asterisk "*" on the label indicates that the property is required.*

Access Key
Identification key given in the user security section of the Amazon account.

Secret Access Key
Secret key associated to the Access Key property, given in the user security section of the Amazon account. They are used together when requesting access to the resources provided by the platform.

Region
AWS Region where the bucket is located.

Bucket
Bucket URL based on its S3 structure in AWS.

Directory
Name of the folder within the bucket.

Public Bucket
Public bucket URL based on its S3 structure in AWS.

3.12.6.1. Bucket Security Configuration

Each environment has a public and a private bucket configured. Access permissions to buckets can be assigned by adding credentials in the adapter or by setting permissions directly at the resource level.

Assigning permissions at the resource level

Resource-level permissions are assigned through a policy that must be assigned to the EC2 role where the container is executed. This policy can be assigned either to the adapter user or to the resource (for example, the EC2 container).

Policy Data

The policy must contain the following permissions (making sure to replace deyel and deyel-public with the correct bucket names):

```
{
	'Version': '2012-10-17',
	'Statement': [
	{
		'Sid': 'VisualEditor1',
		'Effect': 'Allow',
		'Action': 's3:ListBucket',
		'Resource': [
			'arn:aws:s3:::deyel ,
			'arn:aws:s3:::deyel-public'
		]
	},
	{
		'Sid': 'VisualEditor0',
		'Effect': 'Allow',
		'Action': [
			's3:GetObjectVersionTagging',
			's3:GetStorageLensConfigurationTagging',
			's3:GetObjectAcl',
			'logs:CreateLogStream',
			's3:GetIntelligentTieringConfiguration',
			's3:GetObjectVersionAcl',
			's3:DeleteObject',
			's3:PutAccessPointPolicyForObjectLambda',
			's3:GetObjectRetention',
			's3:GetBucketWebsite',
			's3:GetJobTagging',
			's3:GetMultiRegionAccessPoint',
			's3:GetObjectAttributes',
			's3:GetObjectLegalHold',
			'logs:CreateLogGroup',
			's3:GetReplicationConfiguration',
			's3:ListMultipartUploadParts',
			's3:PutObject',
			's3:GetObject',
			's3:GetAnalyticsConfiguration',
			's3:GetObjectVersionForReplication',
			's3:GetAccessPointForObjectLambda',
			's3:GetStorageLensDashboard',
			's3:CreateAccessPoint',
			's3:GetLifecycleConfiguration',
			's3:GetInventoryConfiguration',
			's3:GetBucketTagging',
			's3:GetAccessPointPolicyForObjectLambda',
			's3:ListBucketVersions',
			's3:ListBucket',
			's3:GetObjectVersionAttributes',
			's3:GetBucketPolicy',
			's3:GetEncryptionConfiguration',
			's3:GetObjectVersionTorrent',
			's3:GetBucketRequestPayment',
			's3:GetAccessPointPolicyStatus',
			's3:GetObjectTagging',
			's3:GetMetricsConfiguration',
```

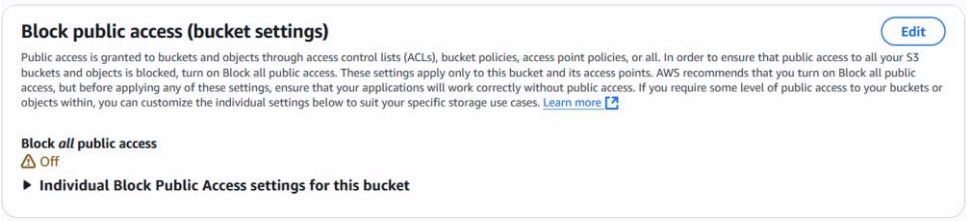
```
                'arn:aws:s3:::deyel/*',
                'arn:aws:s3:::deyel-public/*'
            ]
        }
        ]
}
```

CORS Configuration for Buckets

Buckets must be configured appropriately to allow or deny public access, as appropriate.

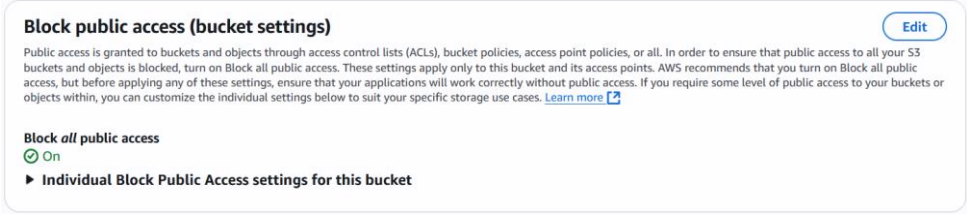
Public bucket configuration

Block all public access: OFF, (Block all public access = OFF)



Private bucket configuration

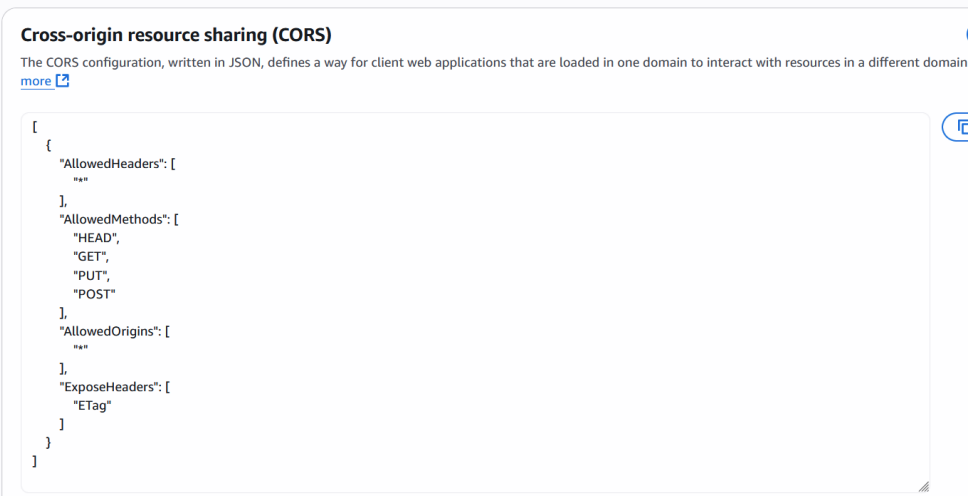
Block all public access: ON, (Block all public access = ON)



CORS Configuration

In the permissions section, enable the Cross-Origin Resource Sharing (CORS) option and configure the following methods:

```
[
        {
        'AllowedHeaders': [
                '*'
        ],
        'AllowedMethods': [
                'HEAD',
                 'GET',
                'PUT',
                'POST'
        ],
        'AllowedOrigins': [
                '*'
        ],
        'ExposeHeaders': [
                'ETag'
        ]
        }
]
```



3.13. Download PDF

The user manual is also available in PDF format. It may take a few minutes to open, as it is approximately 39MB in size.



www.deyel.com

Argentina: +54-11-5219-4600

Chile: +56-2-2233-5901

México: +52-55-5980-9528

Email: hola@deyel.com

Síguenos



Argentina +54-11-5219-4600

hola@optaris.com

www.optaris.com