

Build a Detail Screen Exercise

Table of Contents

Outline.....	2
Resources	2
Scenario	3
How-To.....	8
Getting Started	8
EmployeeDetail Screen	12
ProjectDetail Screen	19

Outline

In this exercise, we'll build two screens. The first will be used to display and edit the details about an Employee. The second will do the same, but for Projects.

These screens are already created, so we just need to build two Forms, one on each Screen, with the attributes that the user will be able to see and edit.

In sum, here's what we'll do:

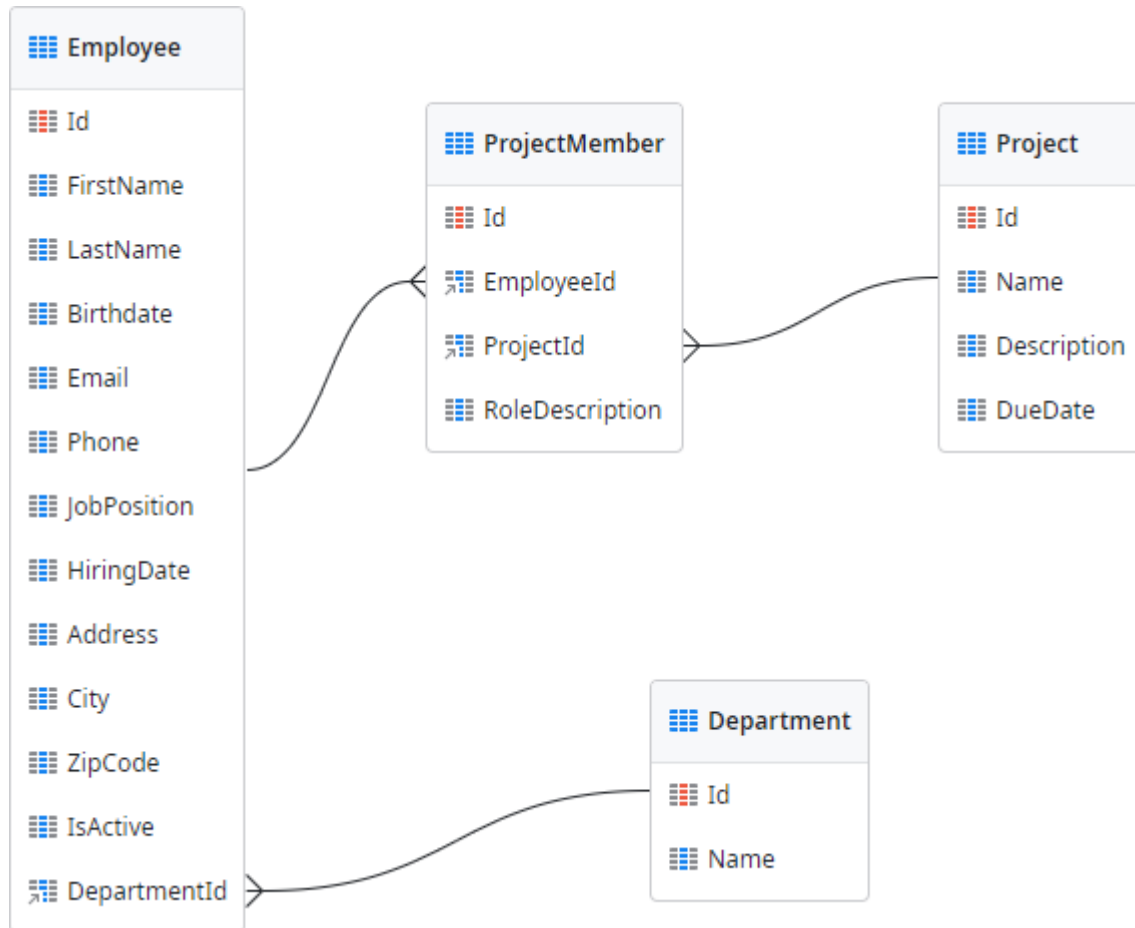
- In the EmployeeDetail screen, create a Form with an input widget per attribute of the Employee.
- In the EmployeeDetail screen, build the logic to create or update an employee in the database.
- In the ProjectDetail screen, create a Form with an input widget per attribute of the Project.
- In the ProjectDetail screen, build the logic to create or update a project in the database.

Resources

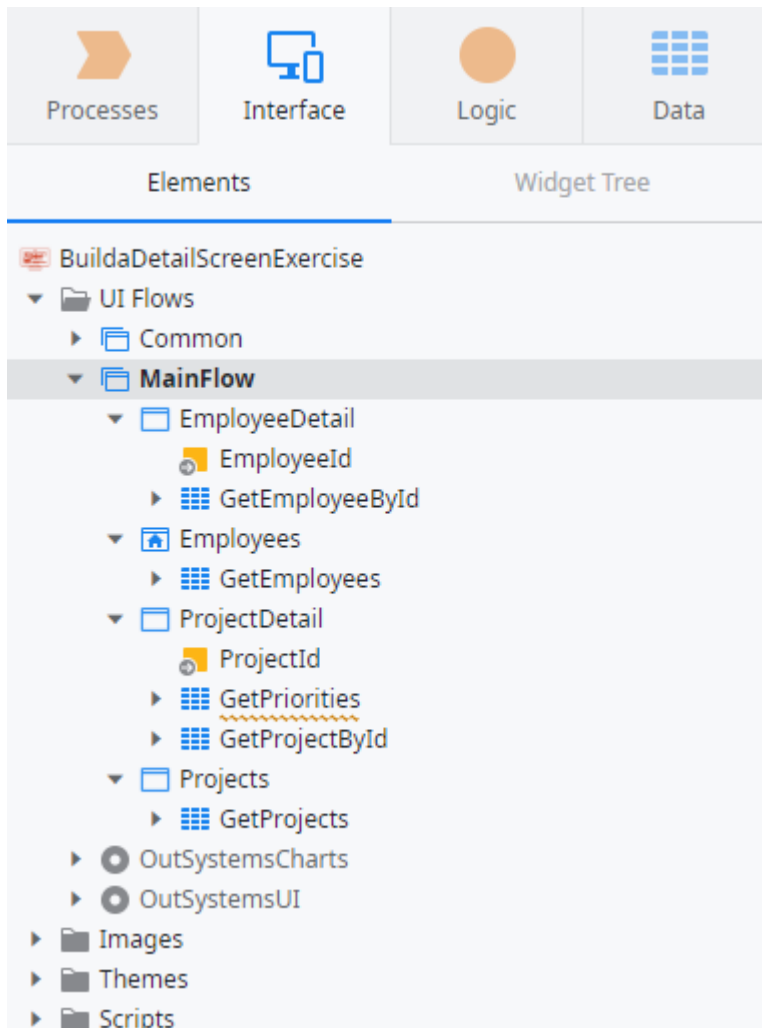
This exercise has a Quickstart application already created. This application has everything needed to start the exercise. This quickstart application can be found in the Resources folder of this exercise, with the name **Build a Detail Screen Exercise.oap**.

Scenario

In this exercise, we'll start from an existing application with one module. Inside that module, we have a data model already defined, with four Entities.



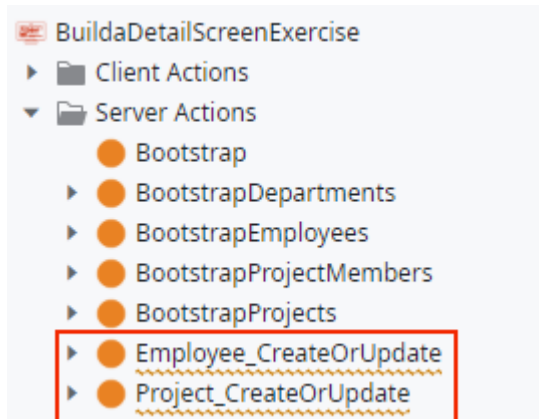
The module also has four screens created: Employees, EmployeeDetail, Projects, and ProjectDetail. The screens have already some Aggregates defined.



The Employees and Projects screens are fully built, so we don't need to touch them. The screens list the employees and projects respectively, and each record has a link to the respective Detail screen.

The EmployeeDetail screen has the **GetEmployeeById** Aggregate, which fetches the employee that matches the Id being passed as parameter. A similar scenario exists for the projects in the ProjectDetail screen with the **GetProjectById** Aggregate. In this case, besides the project, the Aggregate also fetches the Priority. Also, the ProjectDetail screen has the **GetPriorities** Aggregate, that simply fetches all the different priorities stored in the database.

There are also two Server Actions with the logic to create or update Employees and Projects: **Employee_CreateOrUpdate** and **Project_CreateOrUpdate**.



Starting from this application, in this exercise we want to:

- In the EmployeeDetail screen, create a Form with an input widget per attribute of the Employee.
- In the EmployeeDetail screen, build the logic to create or update an employee in the database.
- In the ProjectDetail screen, create a Form with an input widget per attribute of the Project.
- In the ProjectDetail screen, build the logic to create or update a project in the database.

At the end, the EmployeeDetail should look like the following screenshots:

Patricia Wesley

First Name *

Patricia

Last Name *

Wesley

Birthdate *

12/25/1986

Email *

patricia.wesley@example.com

Phone *

1-555-723-3191

Job Position *

Sales Manager West

Hiring Date

02/23/2015

Address *

333 George St

City *

Sydney

Zip Code *

NSW 2000

Is Active

☒

Department

Sales Intelligence

Save

The ProjectDetail should look like the following screenshot:

Task Manager

Name *

Description *

Due Date *

Priority *

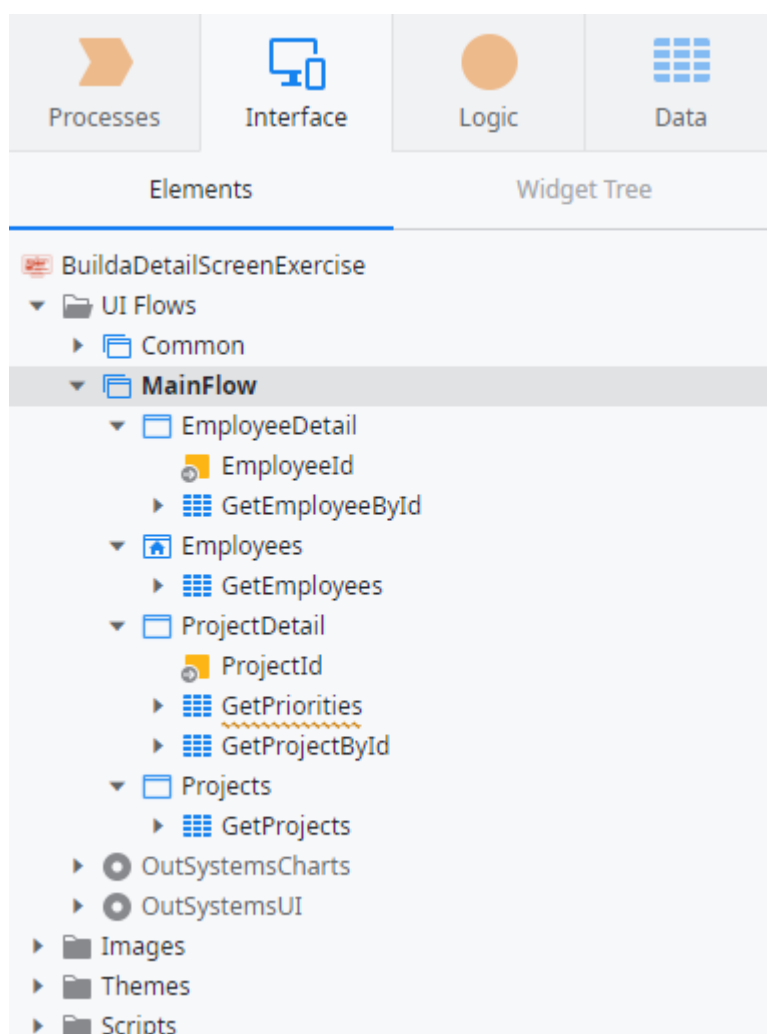
Save

How-To

In this section, we'll show you how to do this exercise, with a thorough step-by-step description. **If you already finished the exercise on your own, great! You don't need to do it again.** If you didn't finish the exercise, that's fine! We are here to help you.

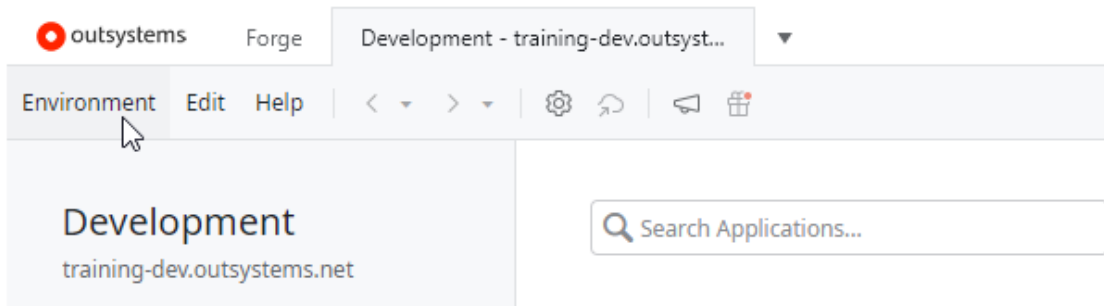
Getting Started

To start this exercise, we need to install the Quickstart file: **Build a Detail Screen Exercise.oap**. This file has an application with two screens, one for listing the projects and one with the details of a project. The screens already have the necessary Aggregates defined, so we'll only need to focus on building the screens with this data.

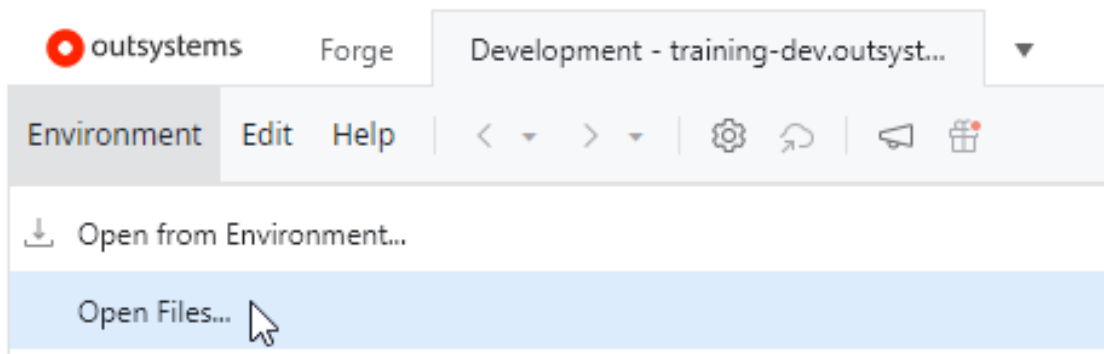


The first step that we need to take is to install the Quickstart application in our development environment. Before proceeding, you must have Service Studio opened and connected to an OutSystems Environment (e.g. Personal Environment).

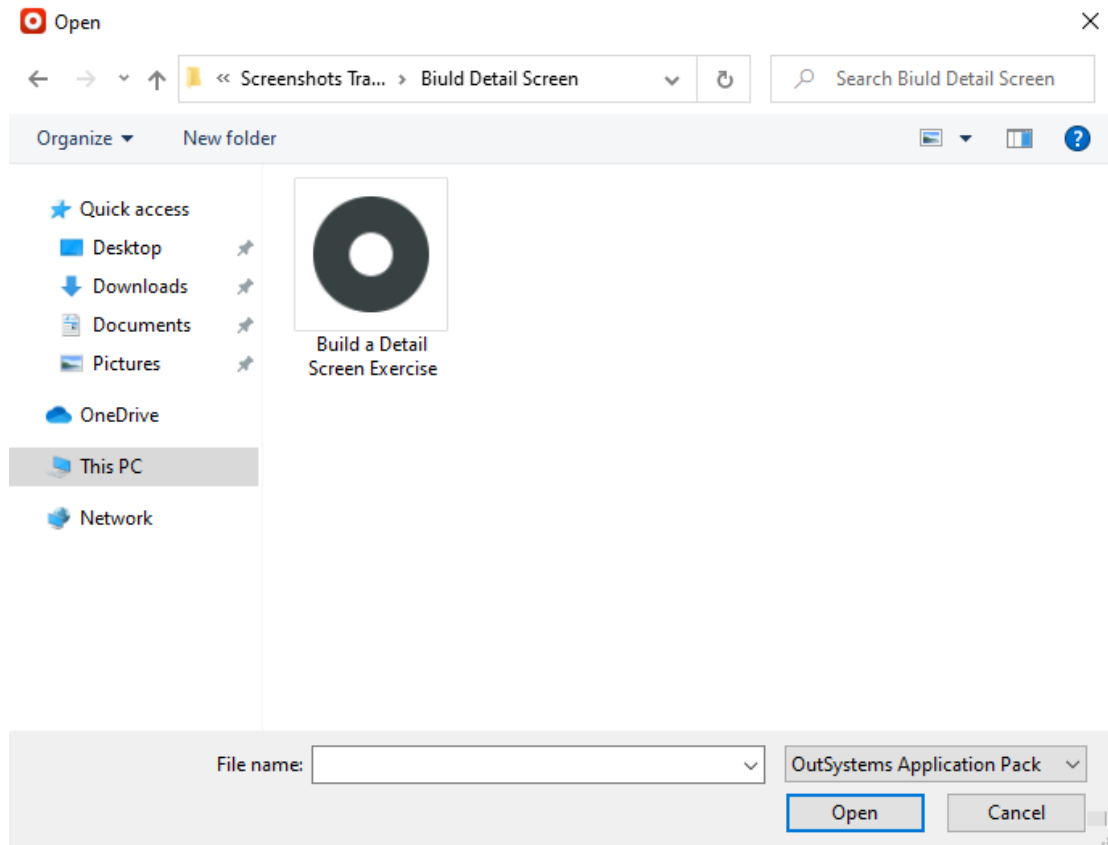
- 1) In Service Studio's main window, select the **Environment** menu on the top left.



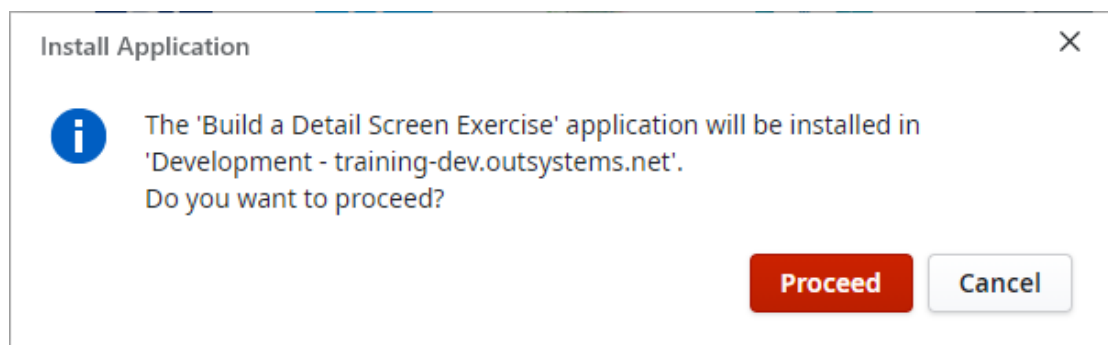
- 2) Select **Open Files...**



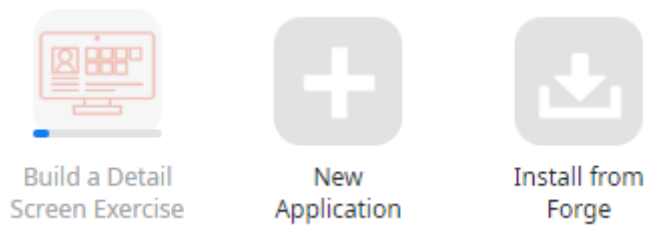
- 3) In the following dialog, change the file type to OutSystems Application Pack (.oap), find the location of the Quickstart and open the file named **Build a Detail Screen Exercise.oap**.



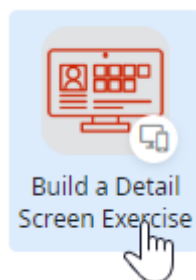
- 4) In the new confirmation dialog, select **Proceed**.



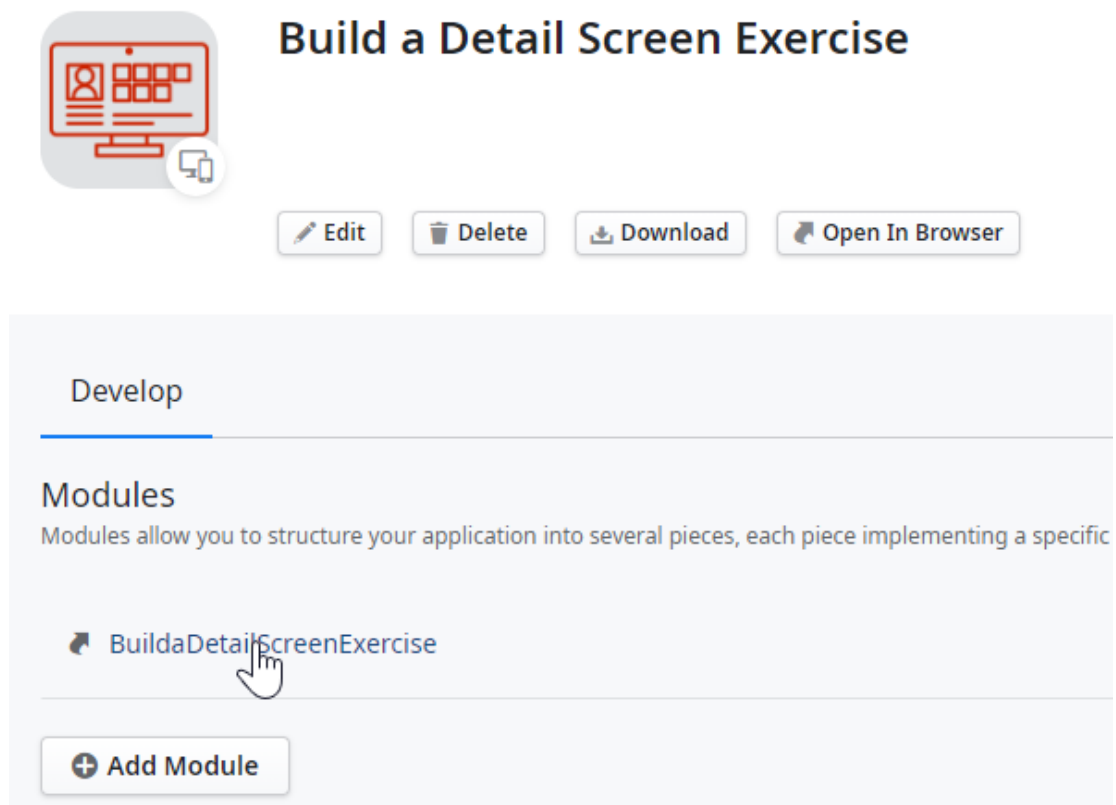
- 5) The application will begin installing automatically. When it's finished, we're ready to start!



- 6) Open the application in Service Studio.



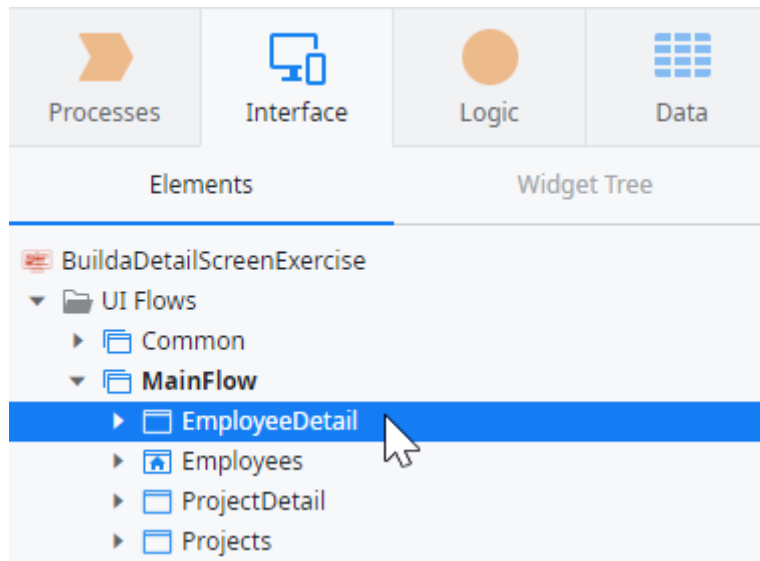
- 7) The application has only one module. Let's open it!



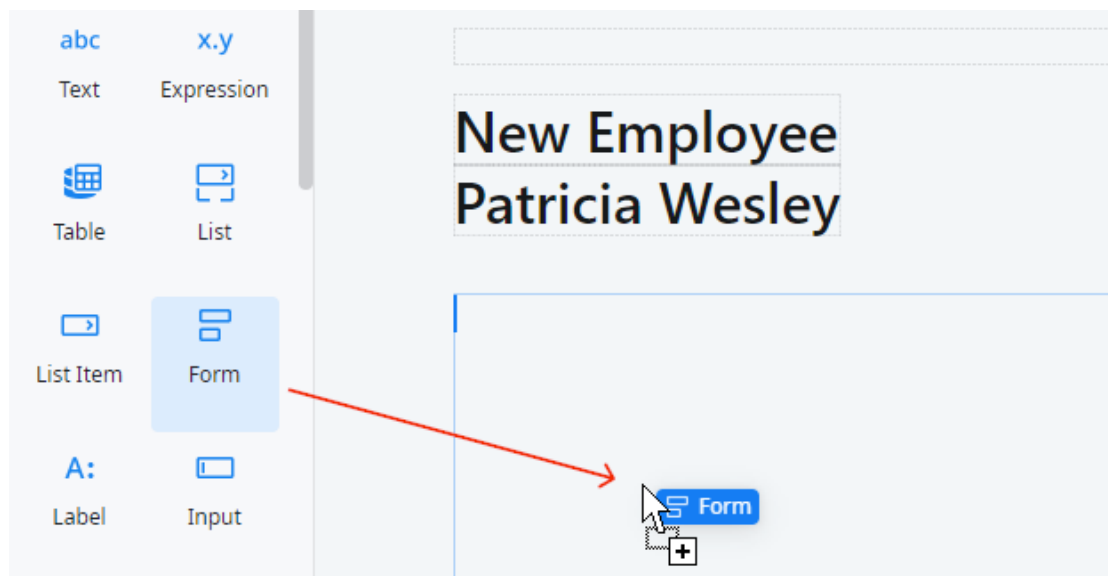
EmployeeDetail Screen

We'll start the exercise by defining the EmployeeDetail screen. The details of an employee should appear on a Form, with all its attributes (except the Id). The Form should have a Save Button that triggers a **SaveOnClick** Action. This Action will create or update an employee in the database.

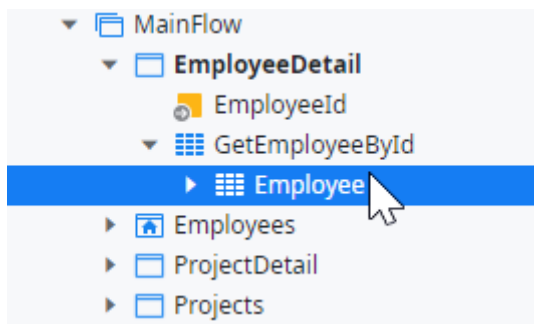
- 1) Open the EmployeeDetail screen.



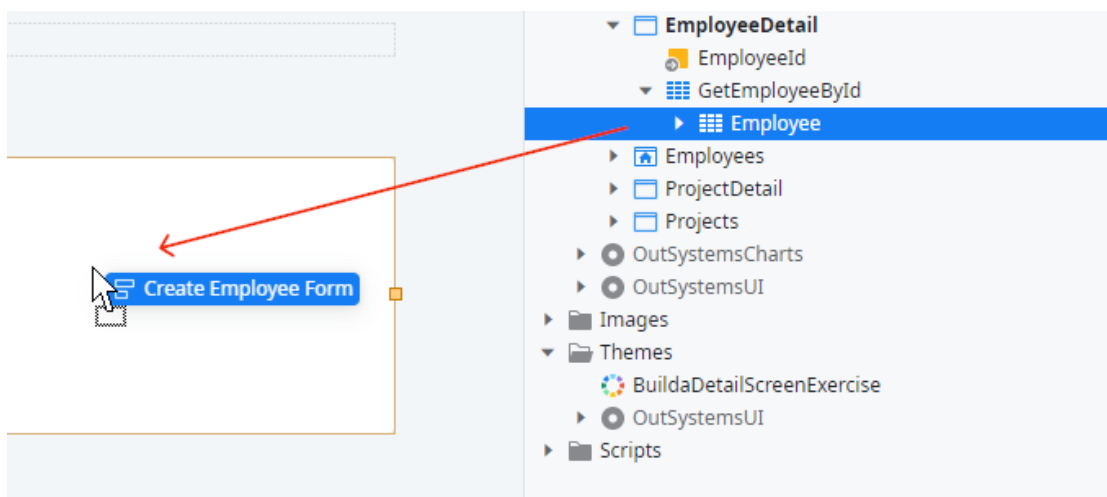
- 2) Drag a **Form** and drop it in the **MainContent** area of the screen.



- 3) On the right, under the EmployeeDetail screen, expand the **GetEmployeeById** Aggregate, and select the **Employee** Entity.



- 4) Drag the **Employee** Entity and drop it on the **Form**.



- 5) Notice that a large Form with all the attributes (except the Id) was created. Each attribute was created with an Input widget that is related to its Data Type. For

instance, the DepartmentId was created with a **Dropdown**, to select the Department, and the IsActive with a **Checkbox**, since its a Boolean attribute.

The screenshot shows a form with the following fields and controls:

- Hiring Date**: A text input field with a placeholder "mm/dd/yyyy".
- Address**: A text input field.
- City**: A text input field.
- Zip Code**: A text input field.
- Is Active**: A checkbox that is currently checked.
- Department**: A dropdown menu with "Human Resources" selected.
- Save**: A red button.

Despite this Form being created automatically, we can tweak it by adding new inputs or by replacing the widgets automatically added by others.

- 6) Our module still has an error. Notice that the **Save** button was added automatically to the Form. This Button expects an **OnClick** Destination.

OK Button

⚙ Properties

✍ Styles

Name

Confirmation Message

Enabled

True

Is Form Default

Yes

Visible

True

Style Classes

"btn btn-primary"

Attributes

Property

= Value

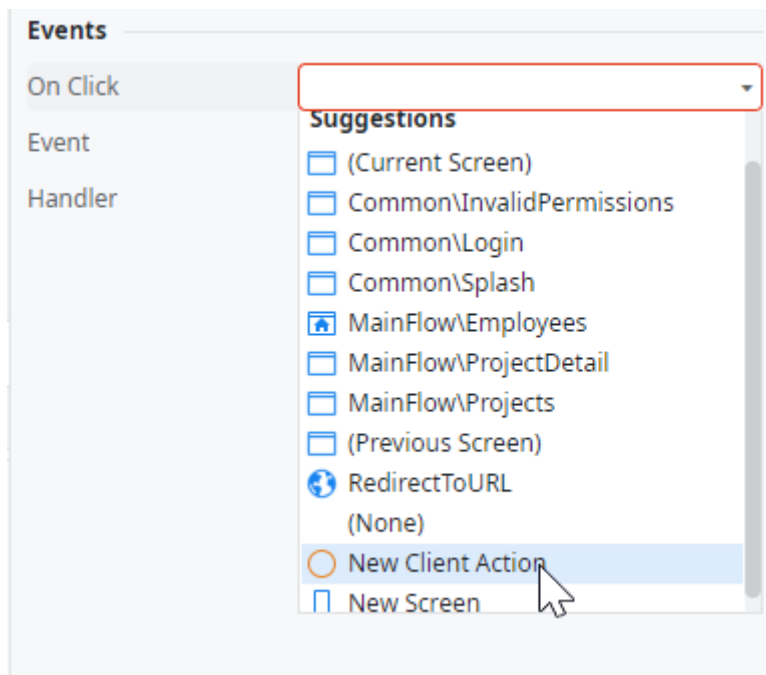
Events

On Click

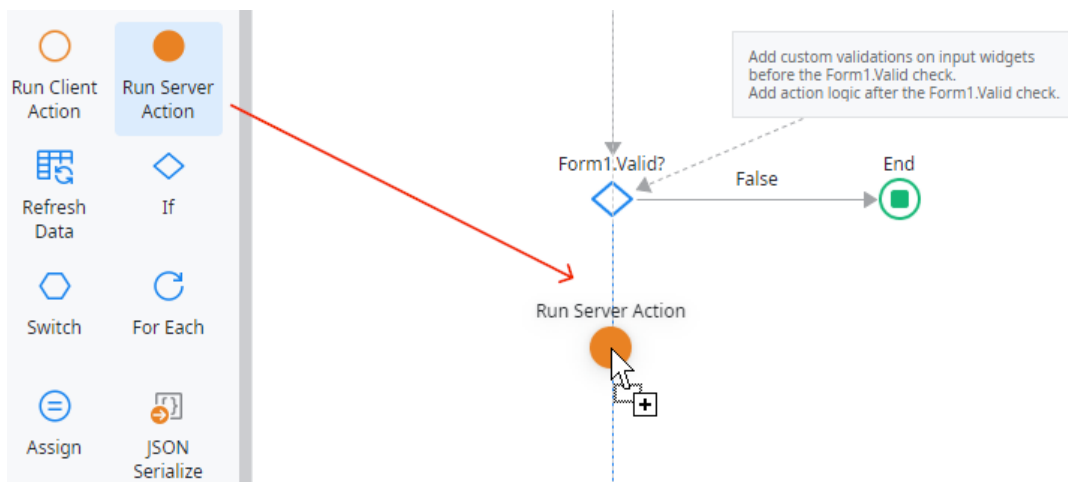
Event

Handler

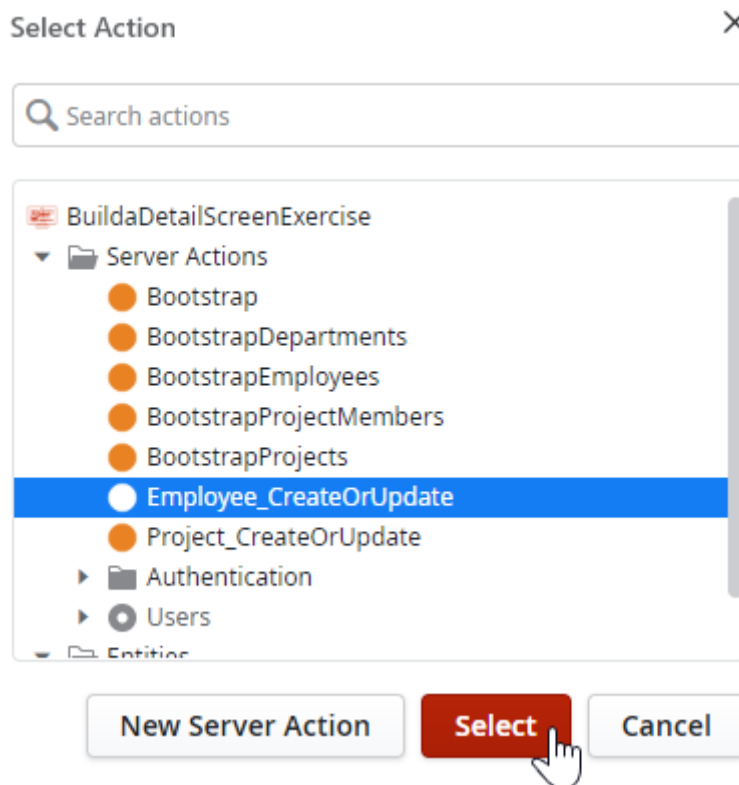
- 7) Set the **OnClick** Destination to *(New Client Action)* to create the *SaveOnClick* Action.



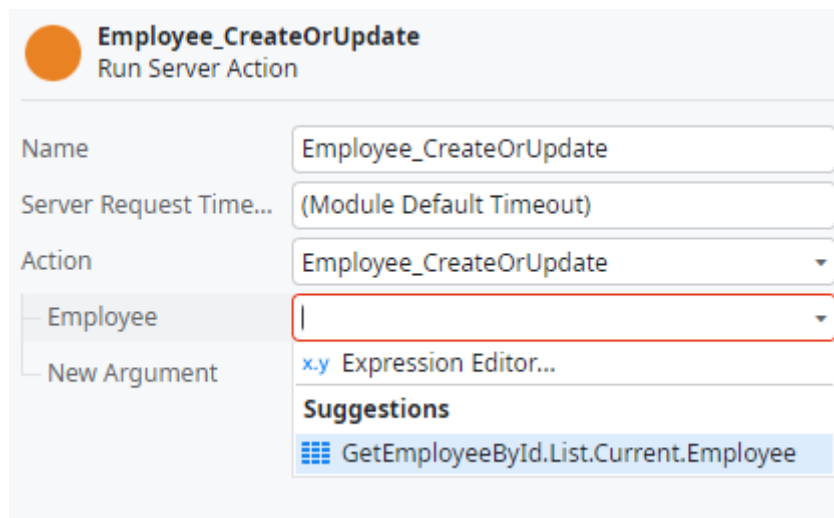
- 8) Drag and drop a Run Server Action to the True branch of the SaveOnClick Action flow.



- 9) In the new dialog, select the **Employee_CreateOrUpdate** Action.



- 10) Set the **Employee** input parameter of the Action to *GetEmployeeById.List.Current*.




- 11) Publish the module to the server to save the latest changes.

- 12) Open the application in the browser. The application opens in the Employees Screen.

Employees			
First Name ↕	Birthdate ↕	Email ↕	Job Position ↕
Patricia Wesley	25 Dec 1986	patricia.wesley@example.com	Sales Manager West
Edward Williams	9 Oct 1980	edward.williams@example.com	Human Resources Manager
Andrea McCarthy	14 Dec 1986	andrea.mccarthy@example.com	Services Manager West
Ann Olivarria	18 Aug 1978	ann.olivarria@example.com	Services Representative
Bridget Hernandez	10 Nov 1982	bridget.hernandez@example.com	Services Representative

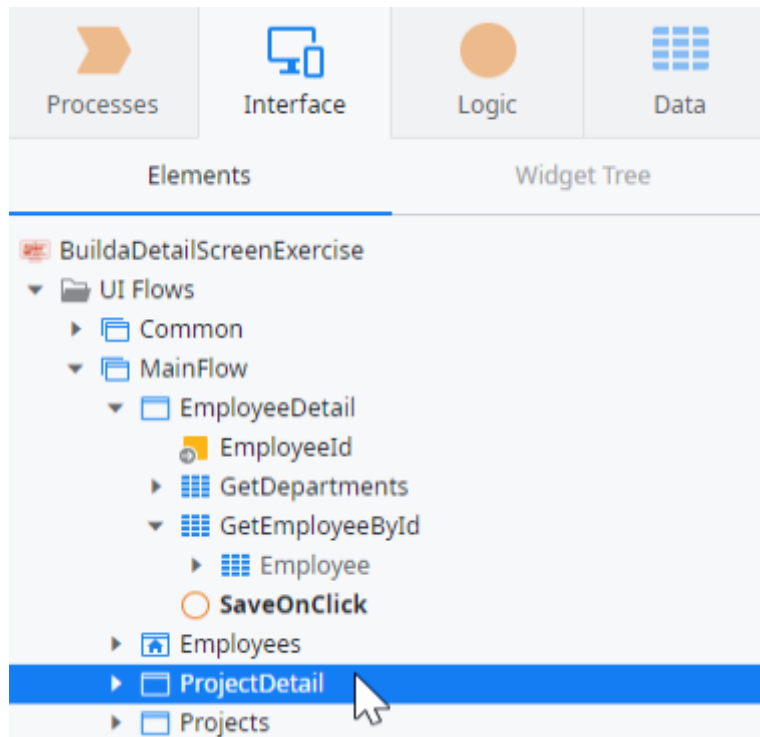
- 13) Select an Employee from the list to see its details. Make sure everything appears as expected in the Form.

Patricia Wesley	
First Name *	<input type="text" value="Patricia"/>
Last Name *	<input type="text" value="Wesley"/>
Birthdate *	<input type="text" value="12/25/1986"/> 
Email *	<input type="text" value="patricia.wesley@example.com"/>
Phone *	<input type="text" value="1-555-723-3191"/>

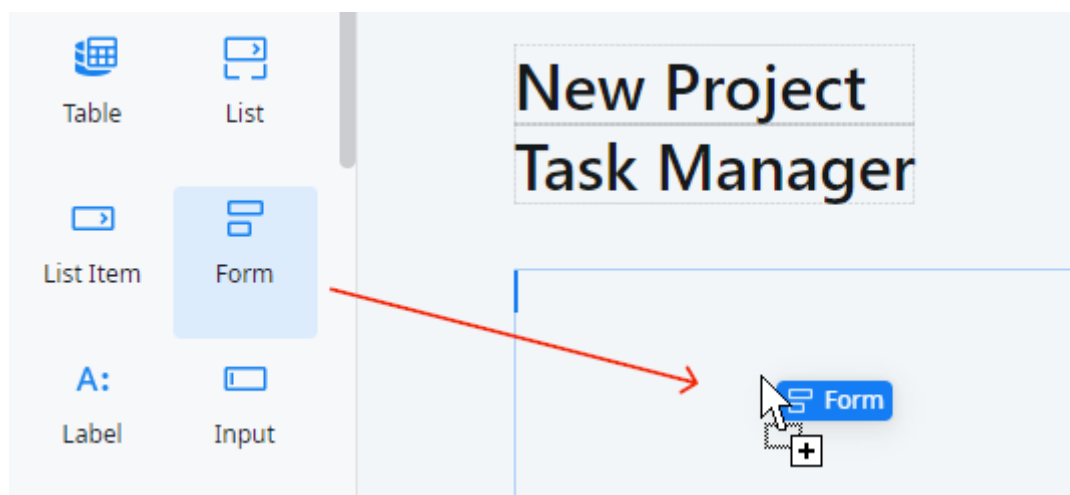
ProjectDetail Screen

In this section, we will implement the ProjectDetail screen and the logic to create or update a project in the database, much like we did for the EmployeeDetail screen.

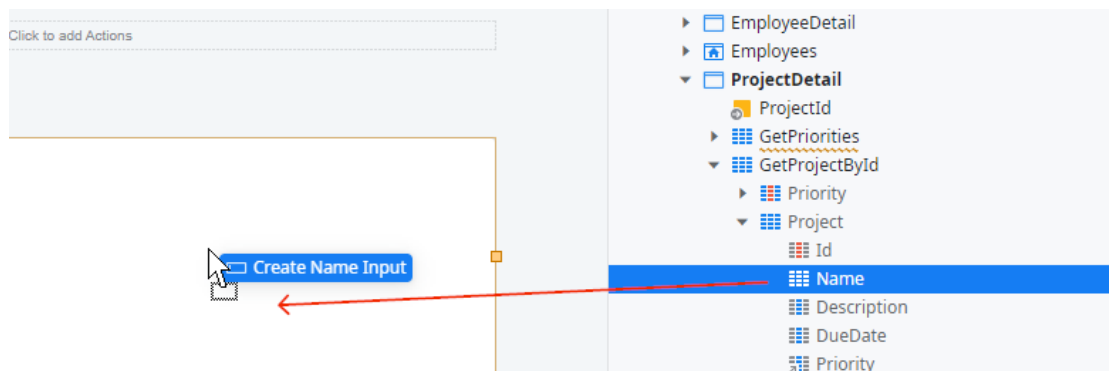
- 1) Double-click the **ProjectDetail** screen to open it.



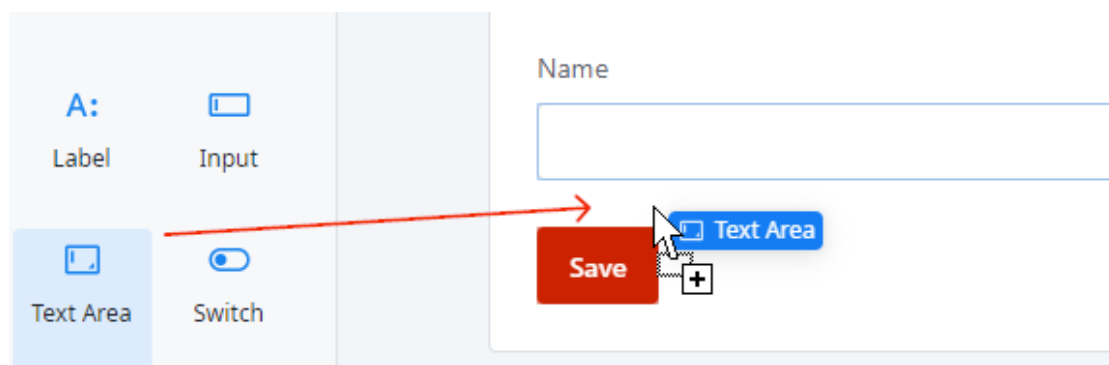
- 2) Drag a **Form** and drop it in the **MainContent** area of the screen.



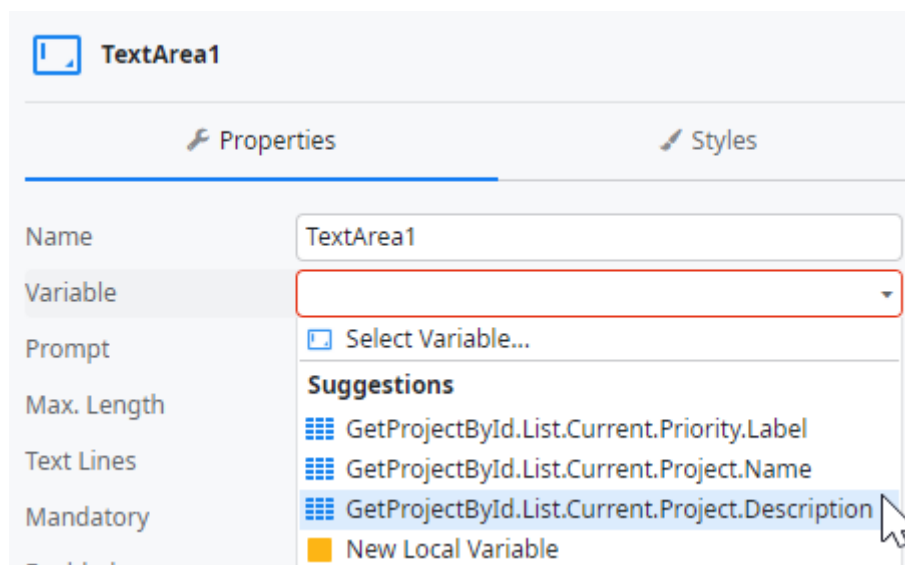
- Expand the **GetProjectById** Aggregate under the ProjectDetail screen. Expand the Project entity, drag the **Name** attribute and drop it on the Form.



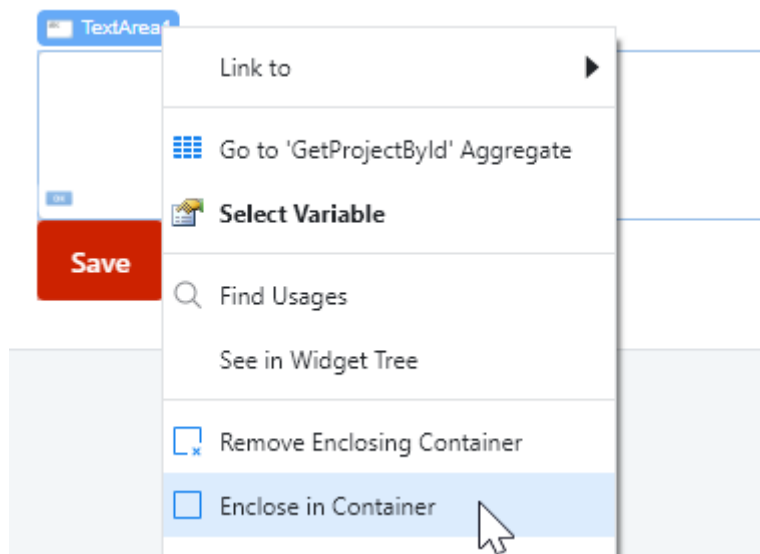
- Drag a **Text Area** widget and drop it between the input for the Name and the Save button.



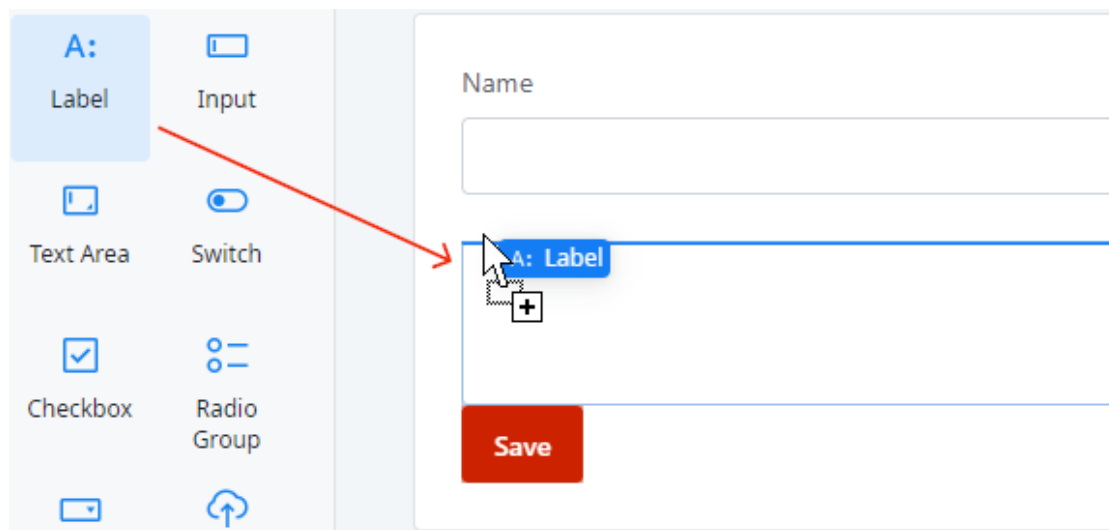
- Set its **Variable** property to be *GetProjectById.List.Current.Project.Description*.



- 6) Right-click on the Text Area and select **Enclose in Container**.

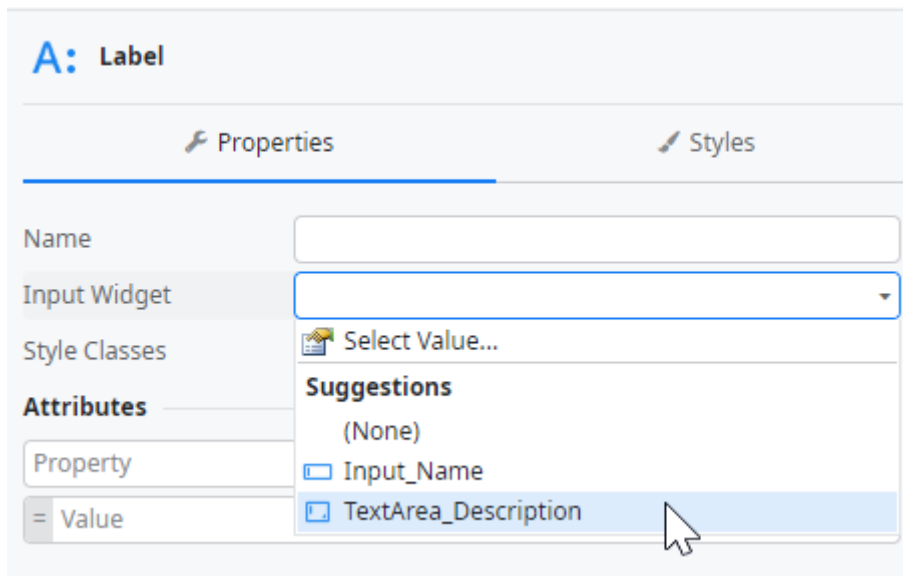


- 7) Drag a **Label** widget before the Text Area, but inside the new container.

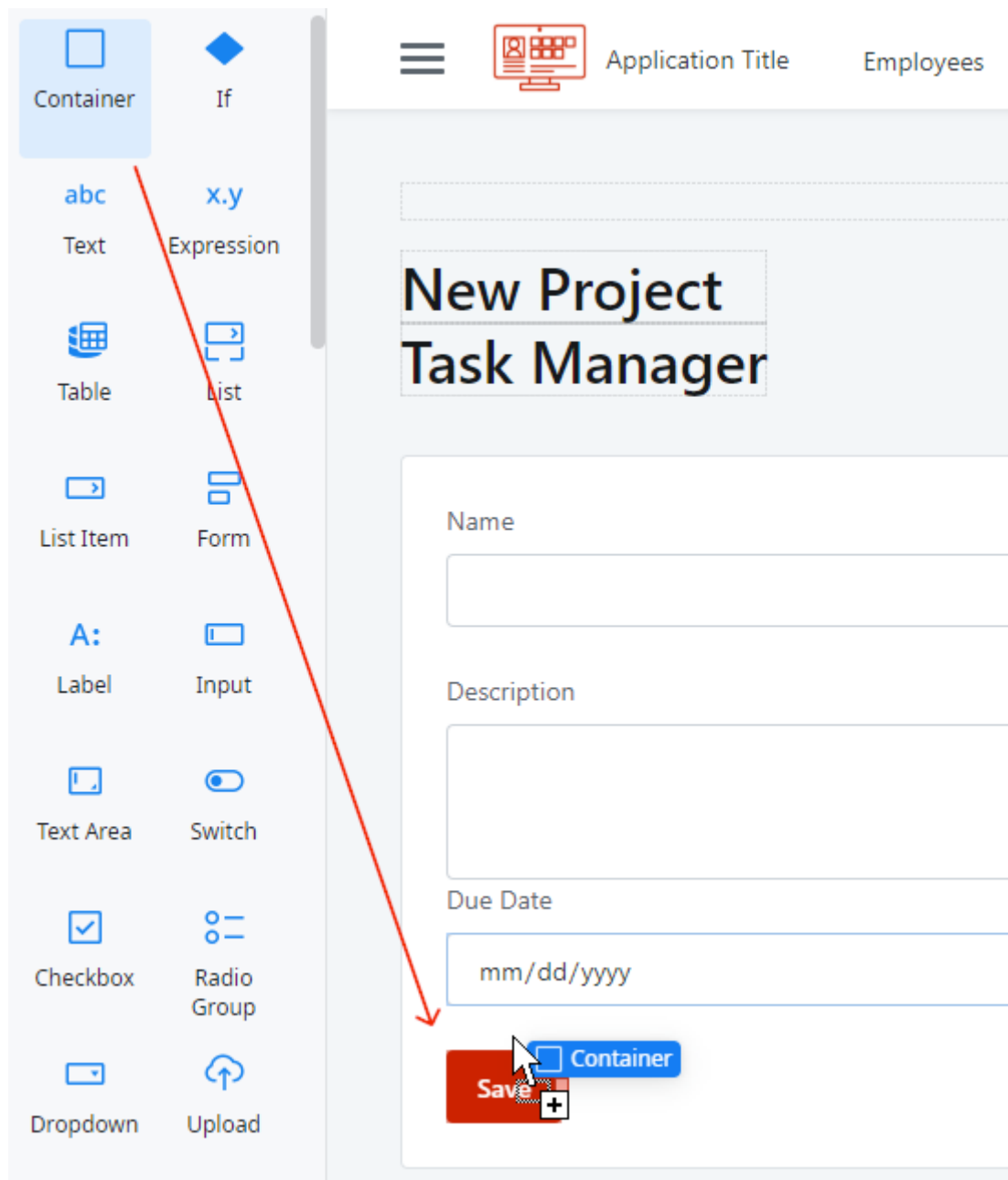


- 8) Set the text of the label to be *Description*.

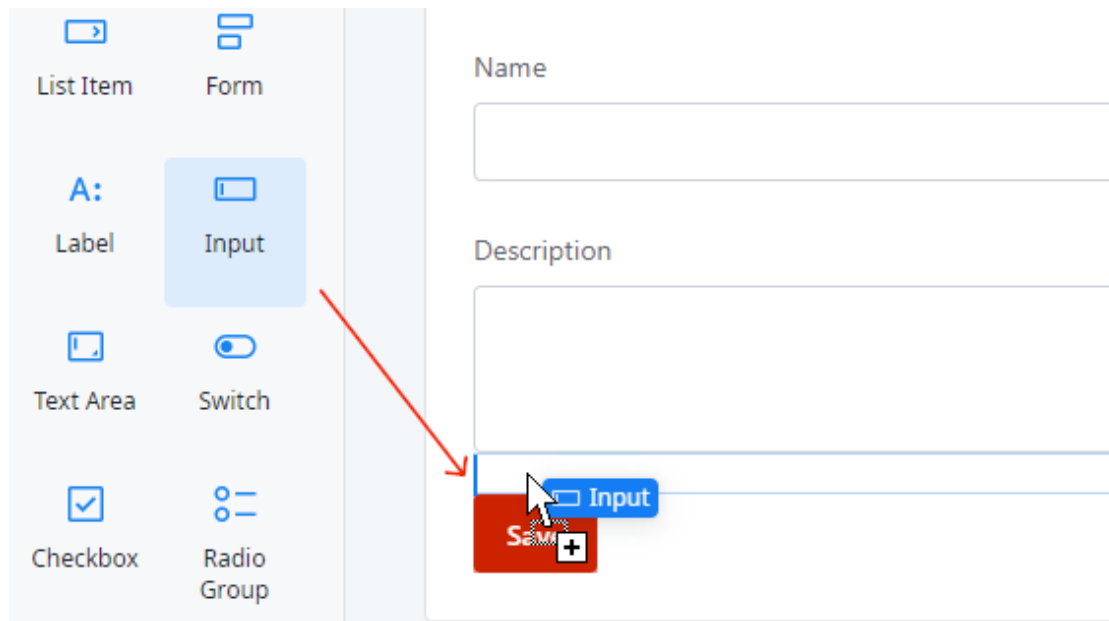
- 9) Set the **Input Widget** property to be *TextArea_Description*, to associate the label with the Text Area.



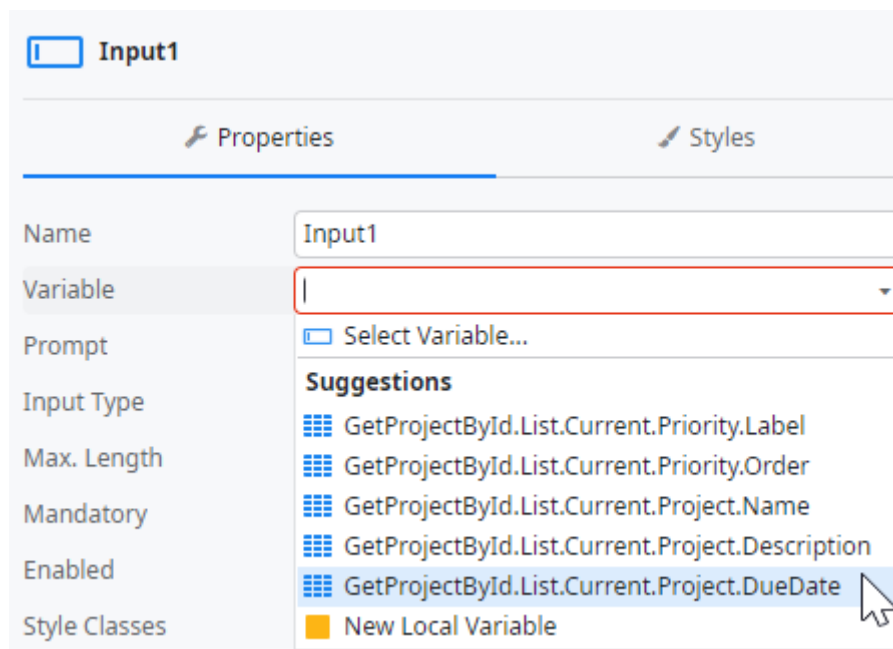
- 10) Drag a new **Container** below the container surrounding the Description label and input, but before the Save button.



11) Drag an **Input** widget and drop it inside the recently created container.



12) Set the **Variable** property to be *GetProjectById.List.Current.Project.DueDate*.



13) Just like for the Description, drag a new **Label** right before the new Input widget, but inside the Container, set its text to be Due Date and set its **Input**

Widget property to be *Input_DueDate* (the name of the recently created Input field).

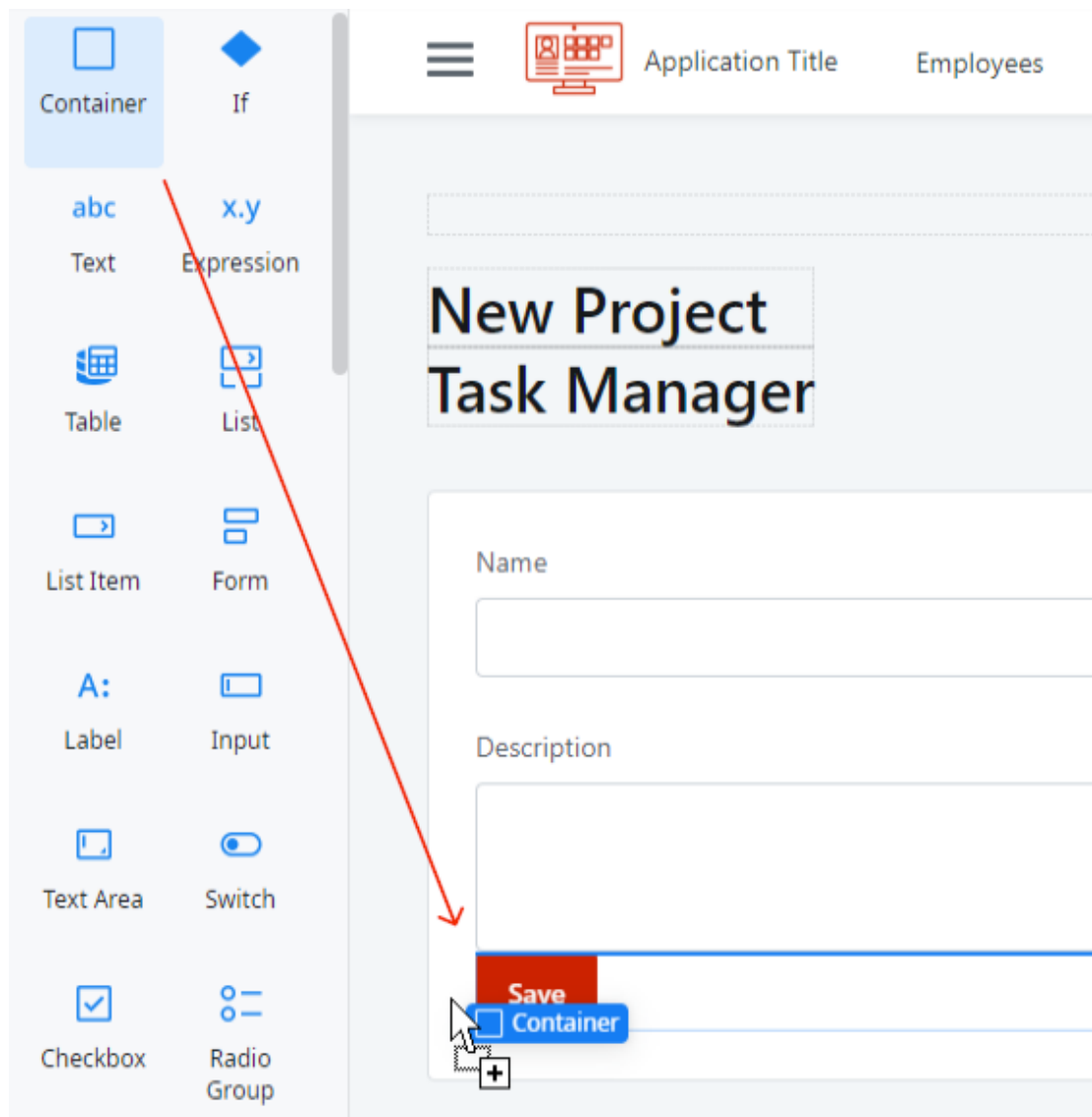
A: Label

Due Date

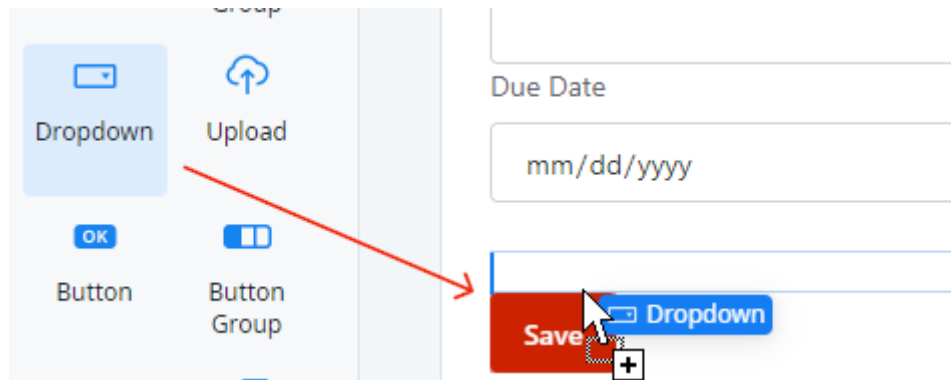
mm/dd/yyyy

Save

- 14) Drag a final **Container** between the container around the Due Date input field and label, but before the Save button.



15) Drag a **Dropdown** widget inside the new container.



16) In the Dropdown, we want to select the **Priority** of the project. For that, the Dropdown must list all the priorities. Since the Priority attribute is of type Priority Identifier, the value chosen by the user must have the same data type. Set the following properties of the Dropdown accordingly

Variable: GetProjectById.List.Current.Project.Priority

List: GetPriorities.List

Options Text: Priority.Label

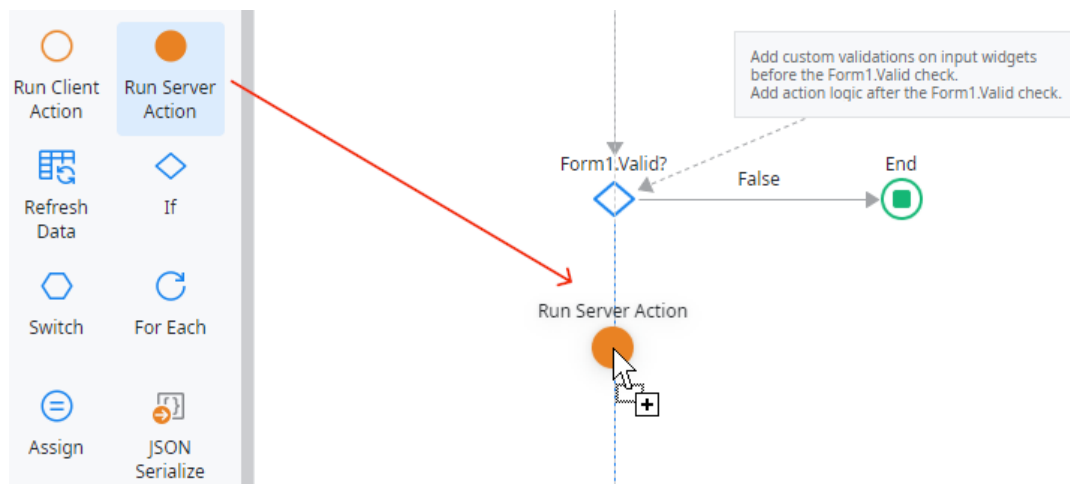
Options Value: Priority.Id

Dropdown1	
Properties	Styles
Name	Dropdown1
Variable	GetProjectById.List.Current.Project.Priority
List	GetPriorities.List
Options Content	Text Only
Options Text	Priority.Label
Options Value	Priority.Id
Mandatory	True
Enabled	True
Empty Text	
Style Classes	"dropdown"

- 17) Just like we did for the Description and Due Date, define the **Label** for the Dropdown and set its name to Priority, and set its Input Widget property.



- 18) Double-click on the **Save** button to create the *SaveOnClick* Action. Then drag a Run Server Action to the True branch.



- 19) Choose the **Project_CreateOrUpdate** Action and set the **Project** input parameter accordingly.

Project_CreateOrUpdate Run Server Action	
Name	Project_CreateOrUpdate
Server Request Time...	(Module Default Timeout)
Action	Project_CreateOrUpdate
Project	GetProjectById.List.Current.Project
New Argument	

- 20) Publish the module to the server to save the latest changes.

- 21) Open the application in the browser and navigate to the Projects Screen using the app menu.



- 22) Open a project and see its details with the widgets we added in this section.

Task Manager

Name *

Task Manager

Description *

Web application for customers to manage their tasks. This application will allow people to submit tasks, set priorities and manage the teams working on those tasks.

Due Date *

12/01/2020

Priority *

High

Save