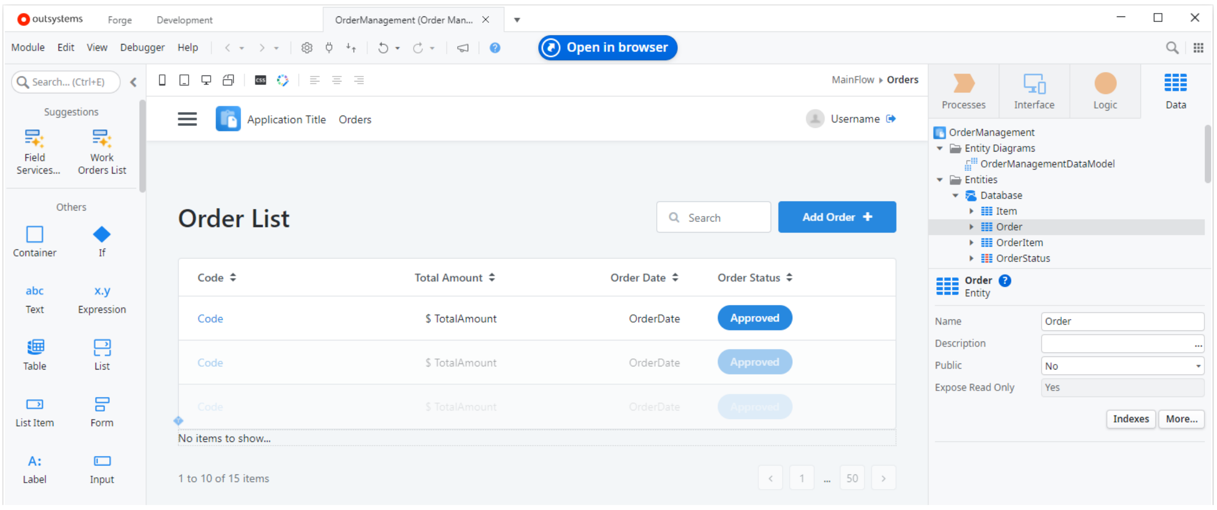
Create the Order Management App

Table of Contents

Outline

In this tutorial, you will build an Order Management application from scratch using Service Studio, the OutSystems IDE.

You will start by creating the **Order Management** application, defining its data model and then creating two Screens.



In the end, you will be able to open your app in the browser and try it out.

Scenario

We want to build a web application to manage customer orders, which includes controlling the process of receiving, tracking, and fulfilling them. In this tutorial, you will create the Order Management application and its first two Screens:

- Orders: lists all the orders and their most important information.
- Order Detail: has a Form that allows viewing and editing relevant information of an order, such as date, status, and order code.

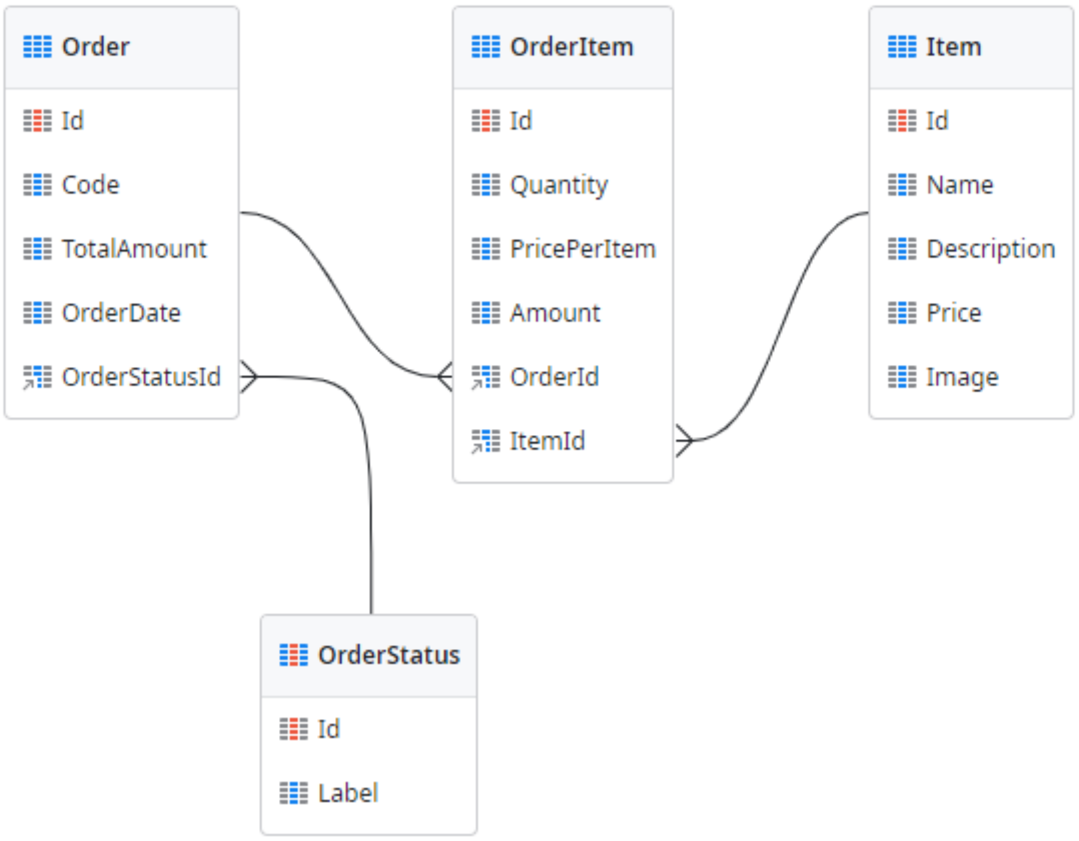
You will start from an empty app, with no data or Screens. The first step will then be to import product data from an Excel file.

Data Model

Every application needs to store and retrieve data that represents various business concepts.

In OutSystems, these concepts are modeled and referred to as Database Entities. This application will have four Database Entities:

- Item, which will hold all the items (products) and all its details, such as name and price, in the database;
- Order, with the order's details, such as date and code;
- Order Item, which is an auxiliary entity to create a relationship between an order and an item;
- Order Status, with all possible statuses of our orders.

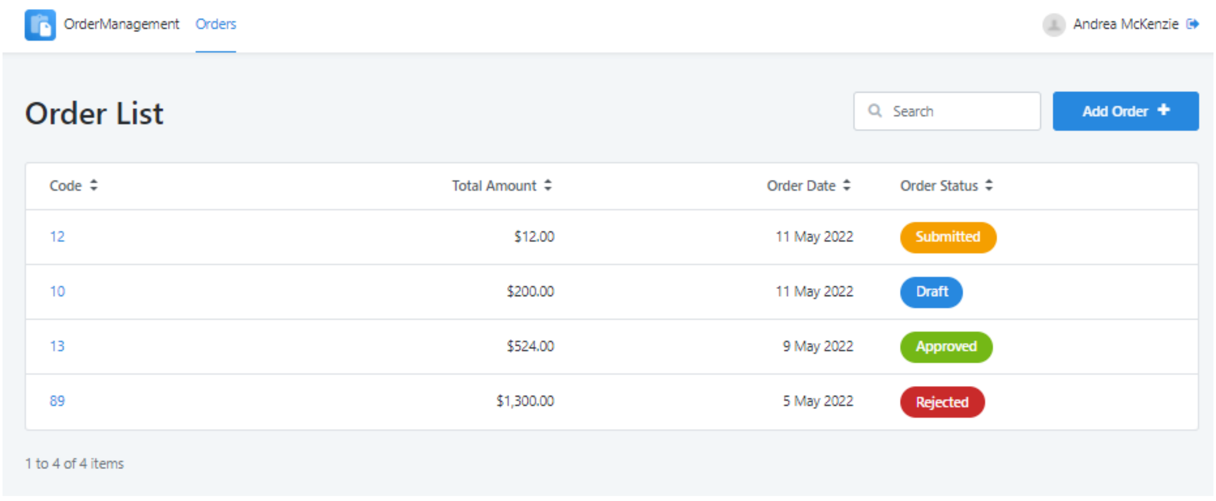


Orders List Screen

The Orders List Screen displays a list of all the orders in a tabular layout, with the code, total amount, date of the order, and order status.

The Screen will also have a text input field to filter the orders by code, right next to a button that will allow the user to add a new order.

The Orders Screen will look like the following image:



Order List

Code	Total Amount	Order Date	Order Status
12	$12.00	11 May 2022	Submitted
10	$200.00	11 May 2022	Draft
13	$524.00	9 May 2022	Approved
89	$1,300.00	5 May 2022	Rejected

1 to 4 of 4 items

Order Detail Screen

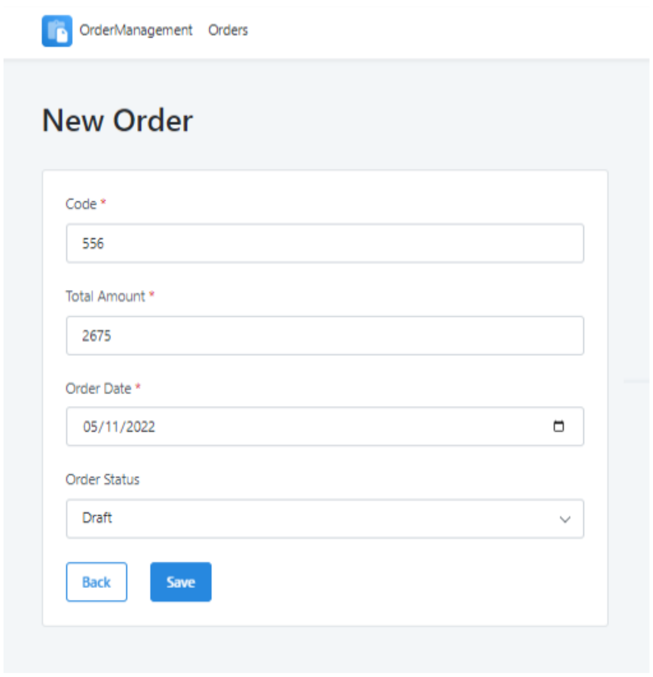
The Order Detail Screen displays all the fields of a specific order.

A user of this application can navigate to this Screen in two ways:

- when clicking on the code of the order in the Orders Screen;
- when clicking on the option to add a new order.

As mentioned before, this Screen will be used to view the details of the order, but will also allow a user to modify and save the order data. For that purpose, the Screen should also have a Save button that executes the logic to save the order information in the database. Right next to it, we will also have a Back button to return to the Orders Screen.

The Order Detail Screen will look like the image below:


OrderManagement Orders

New Order

Code *

556

Total Amount *

2675

Order Date *

05/11/2022

Order Status

Draft

How-To

Now that you know the scenario, let's build the Order Management app by following this how-to guide! Are you ready? Let's do it together!

Resources

This how-to guide has two resources that you will need:

- The **Order Management.oap** file, which is the quickstart for the application that you will create. Don't worry, we just created the empty app to get you started quickly, but didn't create any functionality yet;
- The **Items.xls** excel file with some sample data. Feel free to add more data if you want to, or even create your Excel file with your data. However, make sure you keep the same columns and column headers, at least for now, so you don't get a mismatch between your data and this guide.

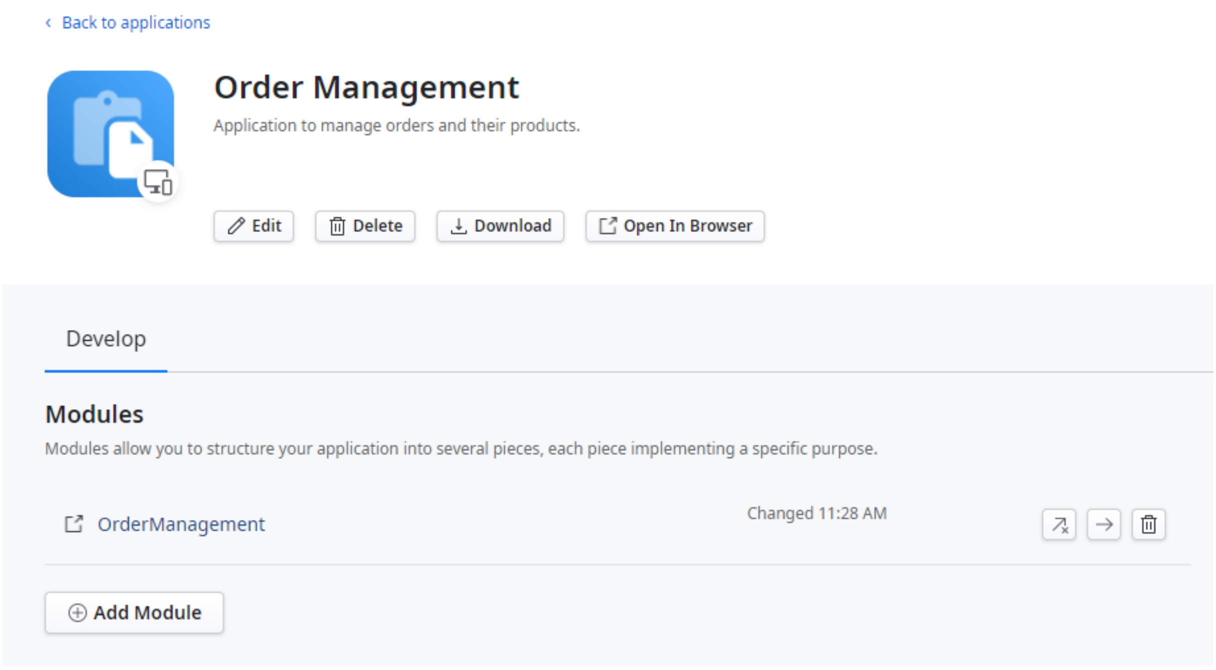
Please download them from the **Lesson materials** on the course page.

Getting Started

Let's start this tutorial by installing the Quickstart file, **Order Management.oap**.

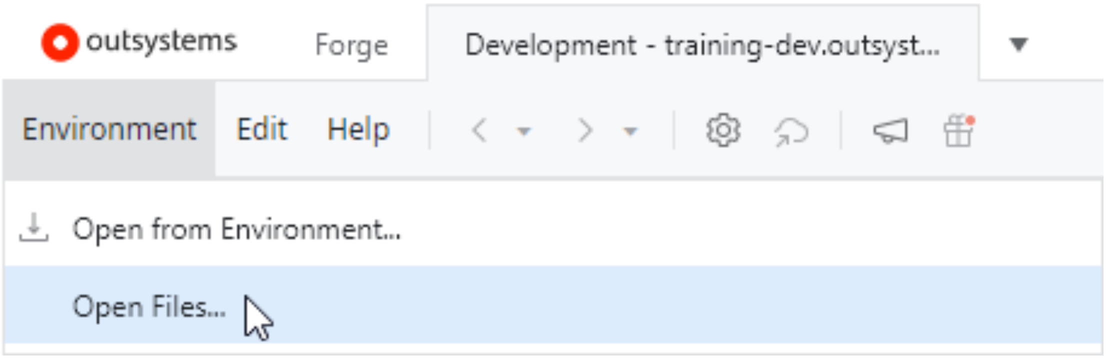
This file has an application with one module, named **OrderManagement**, one icon and a small **description**.

The module is pretty much empty at this point and it's where you will create the Screens and logic of the Order Management web application.

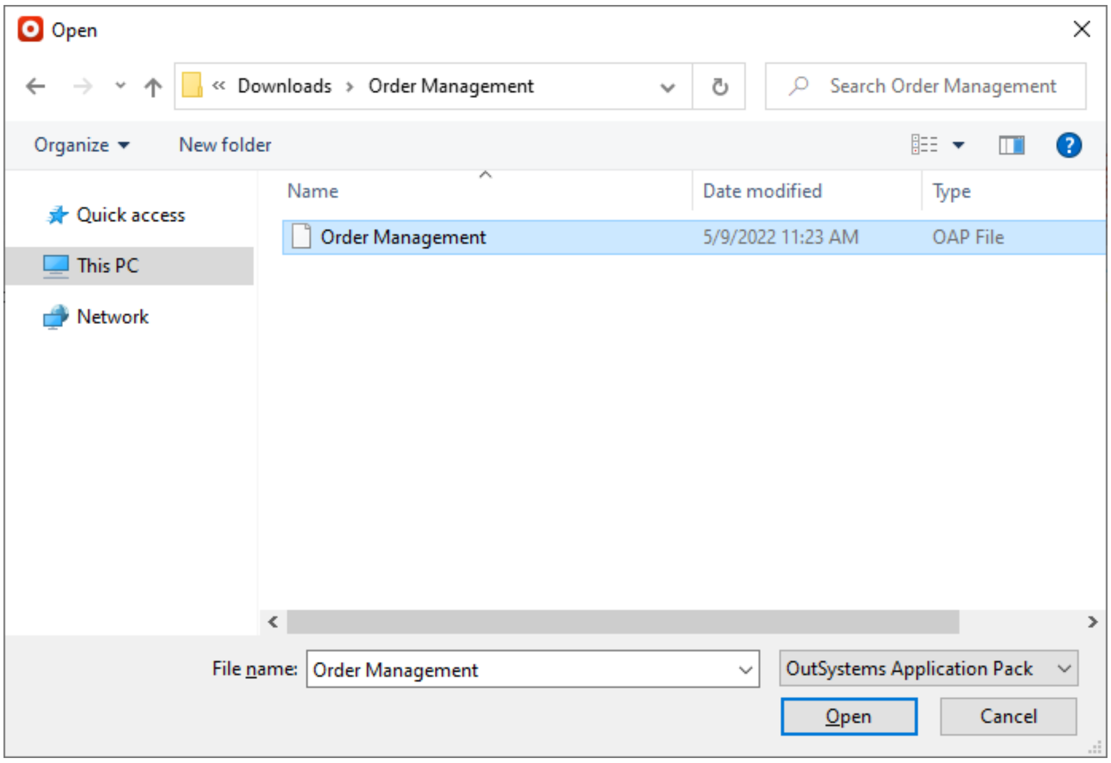

‹ Back to applications
Order Management
Application to manage orders and their products.
Edit
Delete
Download
Open In Browser
Develop
Modules
Modules allow you to structure your application into several pieces, each piece implementing a specific purpose.
OrderManagement
Changed 11:28 AM
Add Module

So, install Service Studio if you haven't done so yet, connect to your OutSystems account (in Service Studio) and download the exercise resources from the **Lesson Materials**.

1) In Service Studio's main window, click on the **Environment** menu on the top left of your window, then select **Open Files...**.



2) In the new dialog, navigate to the folder where you have the resource file and change the file type to OutSystems Application Pack (.oap). You should now see the file **Order Management.oap**. Double-click the file to open it.


Open

« Downloads › Order Management

Search Order Management

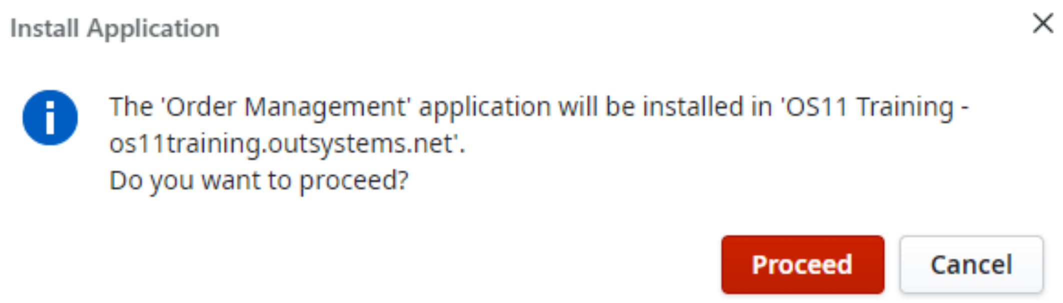
Organize ▾ New folder

Name	Date modified	Type
Order Management	5/9/2022 11:23 AM	OAP File

File name: Order Management

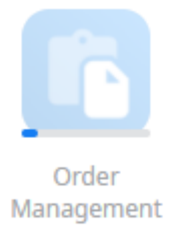
OutSystems Application Pack

3) In the new confirmation dialog that appears in Service Studio, select **Proceed**.

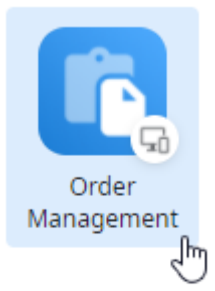

Install Application

The 'Order Management' application will be installed in 'OS11 Training - os11training.outsystems.net'.
Do you want to proceed?

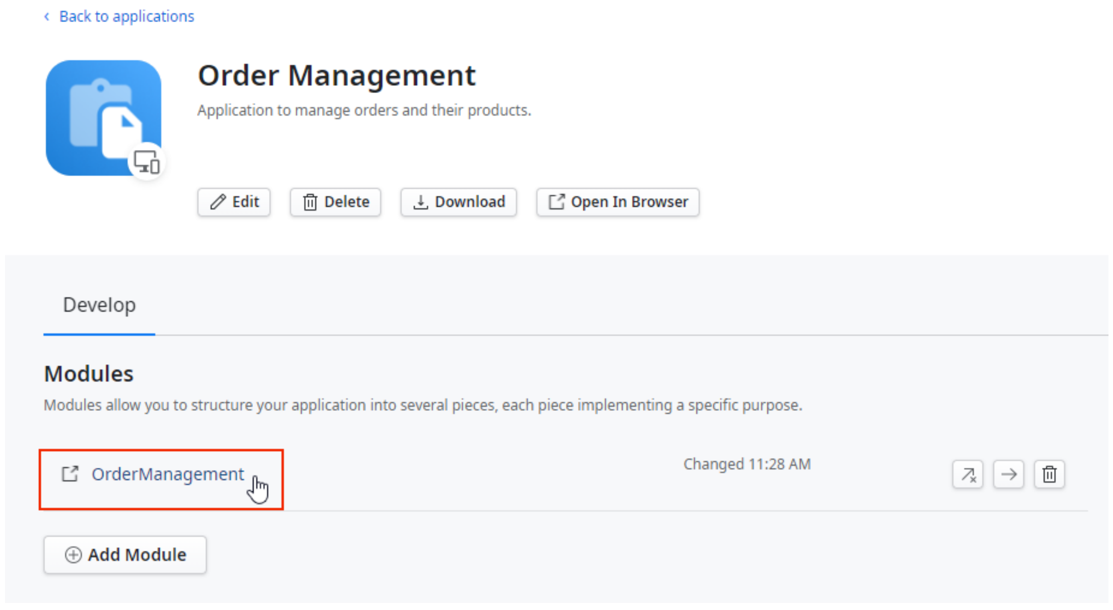
4) The application will begin installing automatically. Give it some time until it's finished.



5) When the progress bar finishes, click on the application icon to open it in Service Studio.



6) The application has only one module. Let's open it by clicking on the module's name!



Order Management

Application to manage orders and their products.

Modules

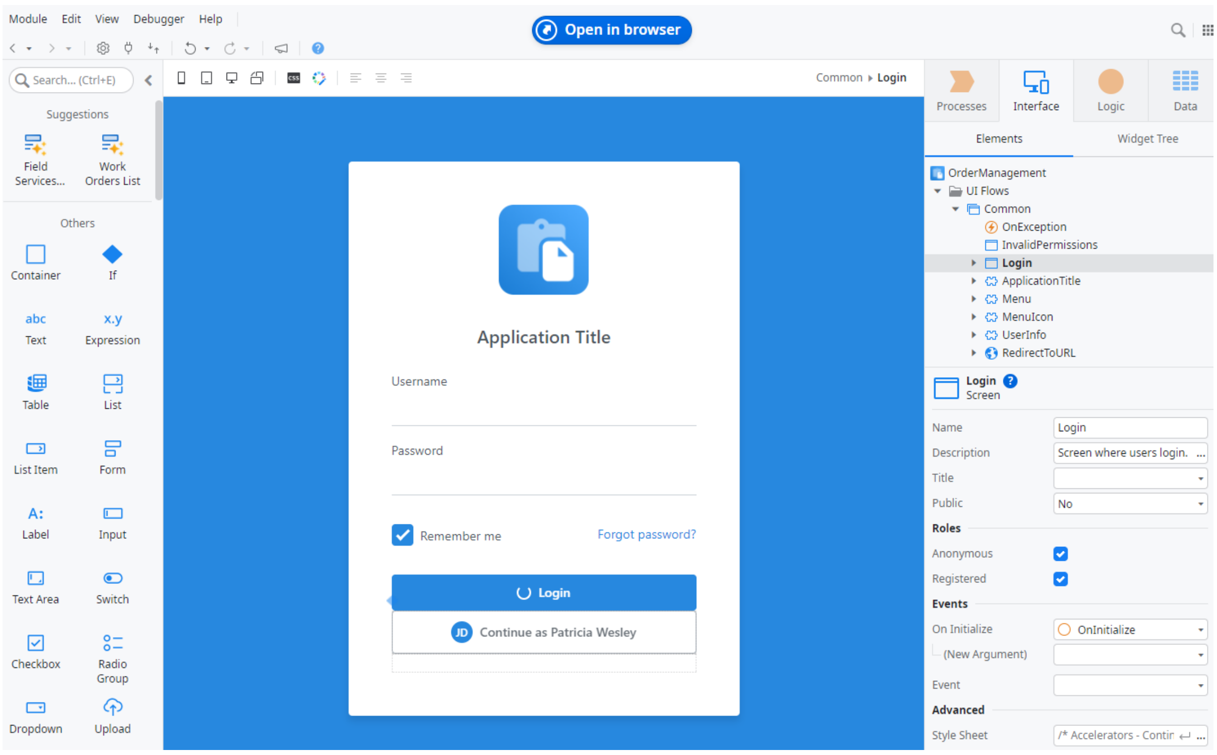
Modules allow you to structure your application into several pieces, each piece implementing a specific purpose.

OrderManagement — Changed 11:28 AM

An application is a set of modules and it is within the modules that you develop the elements in your applications, from UI and business logic to data.

Welcome to Service Studio

Now that you opened the module, you will see the Login Screen, which is automatically created for every new app.

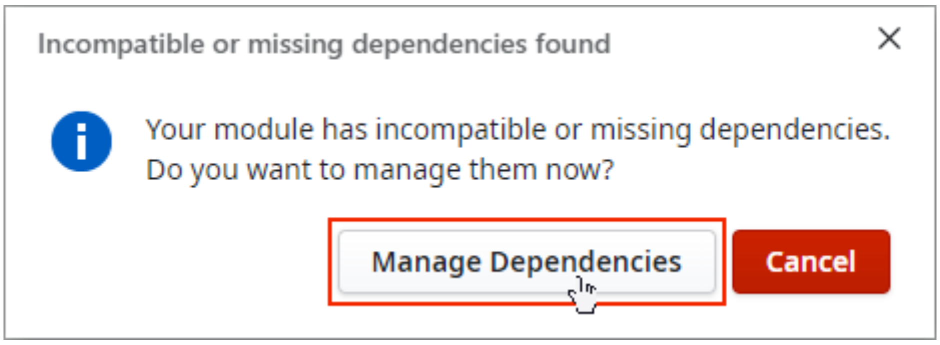


Name	Login
Description	Screen where users login.
Title	
Public	No
Roles	
Anonymous	✓
Registered	✓
Events	
On Initialize	OnInitialize
(New Argument)	
Event	
Advanced	
Style Sheet	/* Accelerators - Contir

Outdated Dependencies

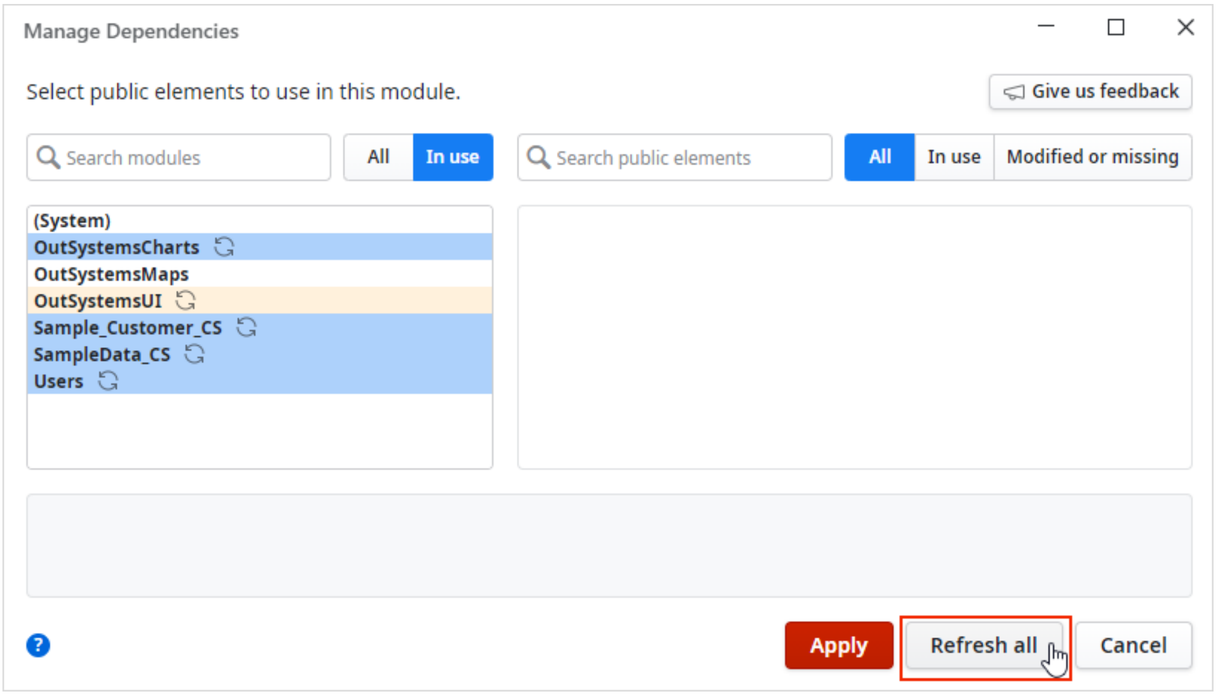
You might get a popup message informing that you have outdated dependencies. This is completely normal, since we are always trying to bring a new and updated version of our components!

If that happens, simply click on the button that says *"Manage Dependencies"* to see the outdated components.

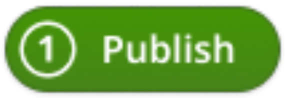

Incompatible or missing dependencies found

Your module has incompatible or missing dependencies. Do you want to manage them now?

Then, click on *"Refresh all"* to update everything at once and *"Apply"* when you are done.

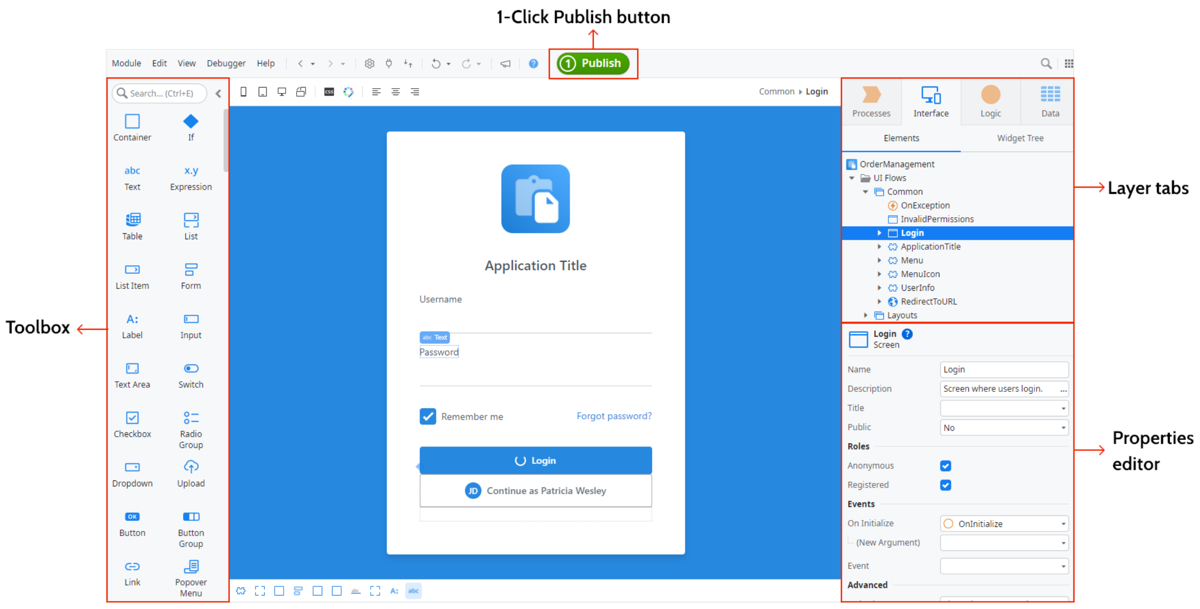


Now publish the module to update the project



The workspace of Service Studio is where you design, deploy, and debug the modules of your applications. This is where you are going to work to create your app.

Note: If you are new to Service Studio, we encourage you to take a good look at the different areas that comprise the workspace, such as the Toolbox, the 1-Click Publish button, the layer tabs, and the properties editor.

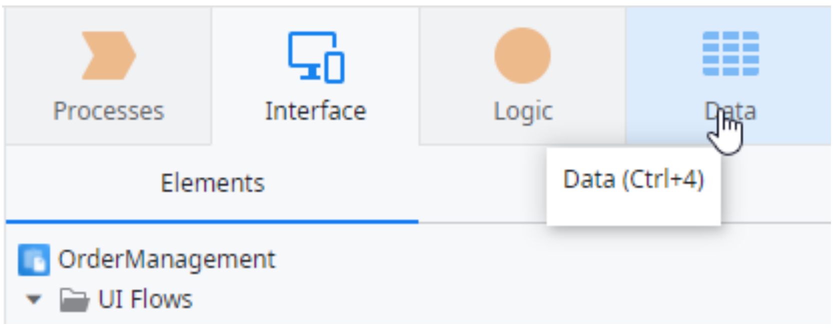


If you want to learn more about Service Studio, don't forget to check our documentation or to enroll in the Service Studio Overview course.

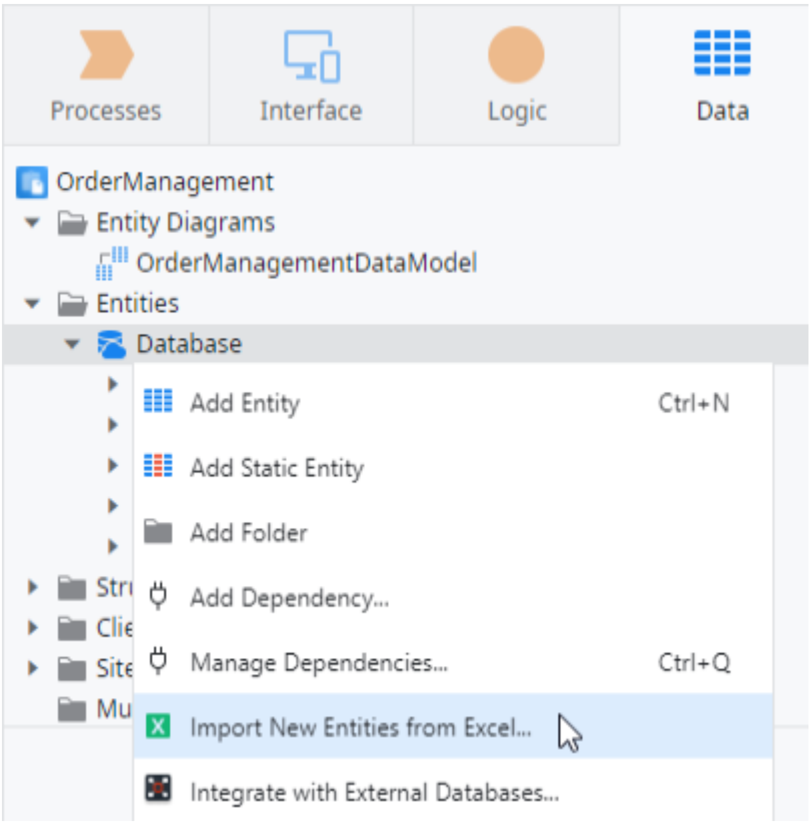
Import Data from an Excel File

Now, that you have the app installed, let's start by adding data to it. In this section, you will import an Excel file with the data for the items that will be available to add to your orders.

1) In Service Studio, select the Data tab in the top-right corner. This will open an area of Service Studio where we can see and define the data for our app.

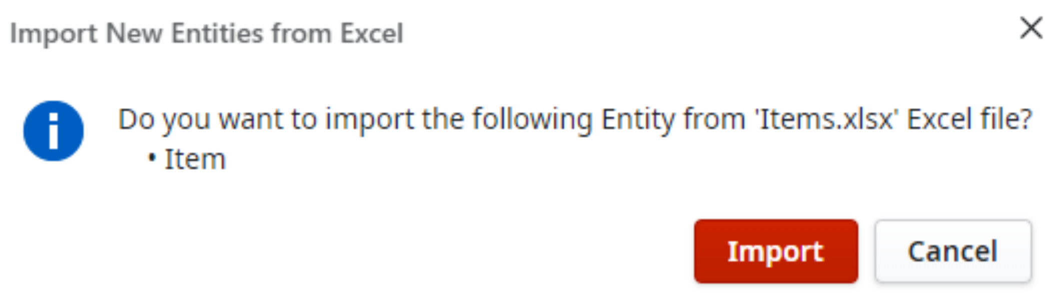


2) Expand the Entities folder, right-click the Database element and select **Import New entities from Excel...**



3) In the new dialog that appears, navigate to the folder where you have the **Items.xlsx** file and select it.

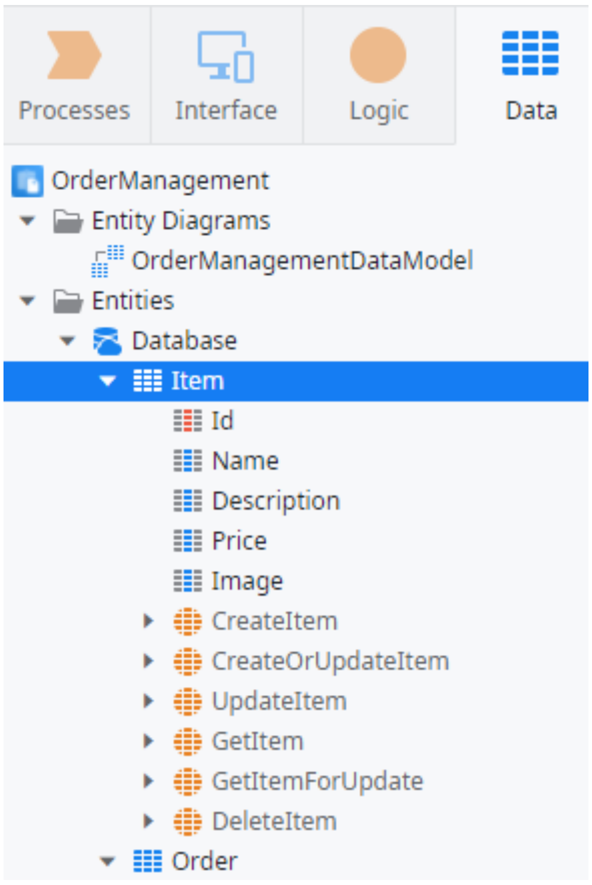
4) A dialog will appear showing the name of the Entity that will be created. Make sure the **Item** Entity appears here. Click **Import**.


Import New Entities from Excel

Do you want to import the following Entity from 'Items.xlsx' Excel file?
• Item

OutSystems analyzes the Excel file and creates one Entity per sheet in the Excel file, using the name of the sheet.

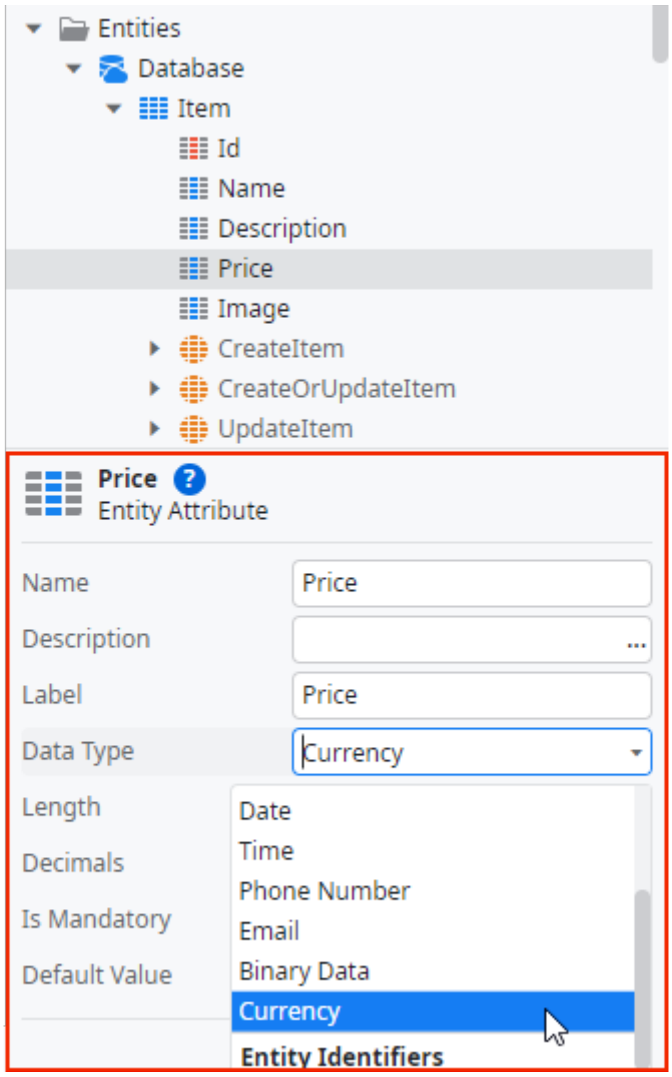
An Entity will be created in the Database, and you can expand it to see that the attributes were also automatically created.



An attribute refers to a column in the Excel file. OutSystems goes through each column, analyzes the content in each row, and creates an attribute with the name of the column and with the appropriate data type.

This is just one way to import data to OutSystems. You can create your data model from scratch, or even import data from other data sources. You can also edit or create your own Excel files. Bear in mind that the sheet and column names will also be the names of your Entities and attributes.

5) Click on the Price attribute to see its properties, then select **Currency** in the dropdown of the property *Data Type*.

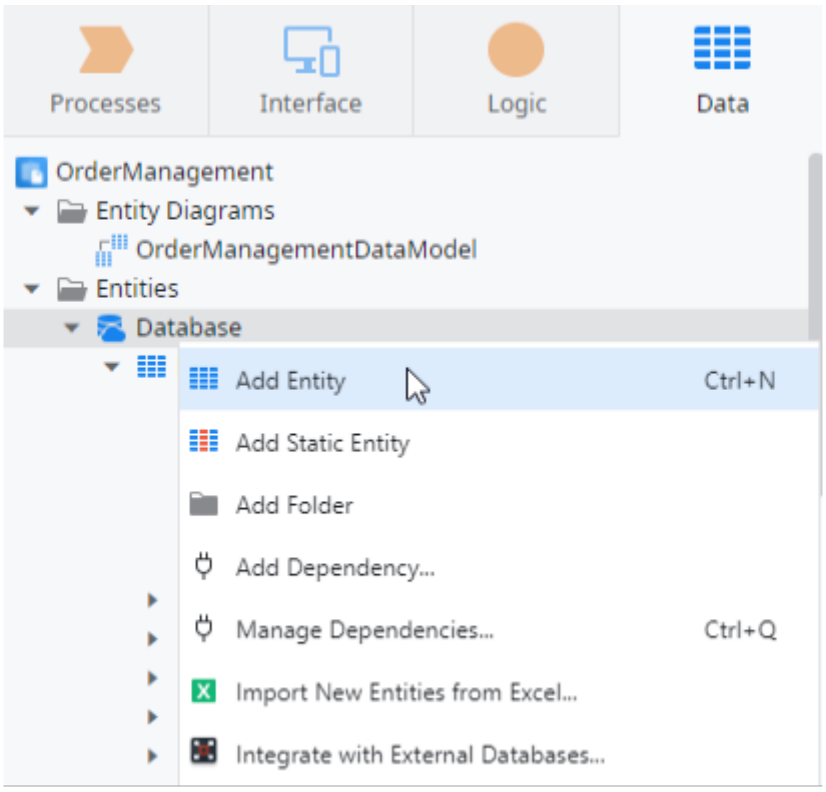


Name	Price
Description	
Label	Price
Data Type	Currency
Length	
Decimals	
Is Mandatory	
Default Value	

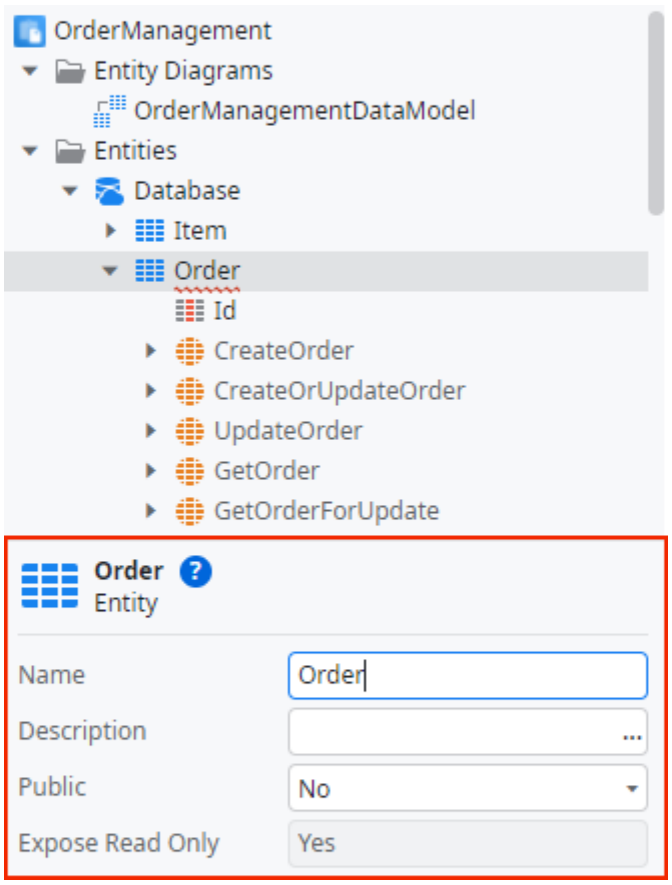
Create Entities from Scratch

You have used the Excel file to populate the database with items. But we need a few more Entities to store other important data, such as the Orders.

1) Right-click the Database and click on the option **Add Entity**

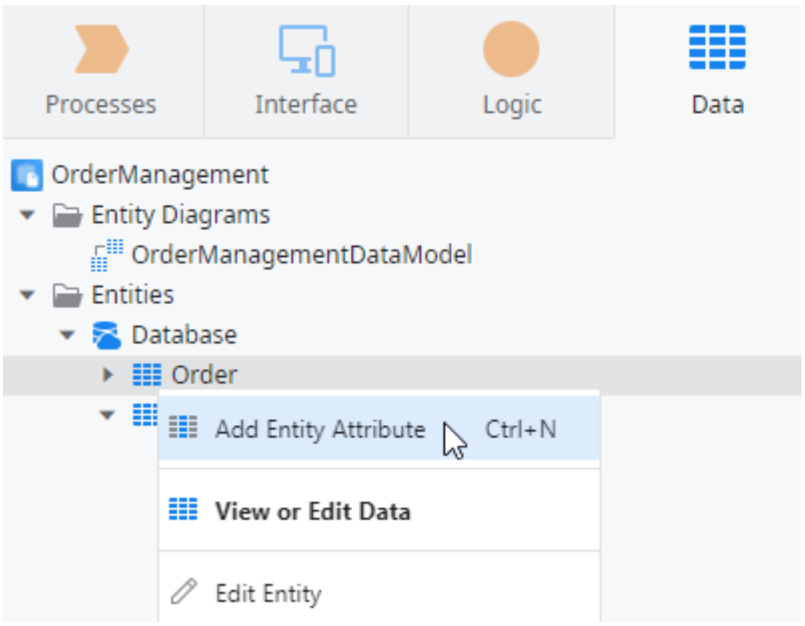


2) Click on the Properties editor and type *Order* as the name of the newly created Entity .



Name: Order
Description:
Public: No
Expose Read Only: Yes

3) Right-click the Order Entity and click on the option **Add Entity Attribute**

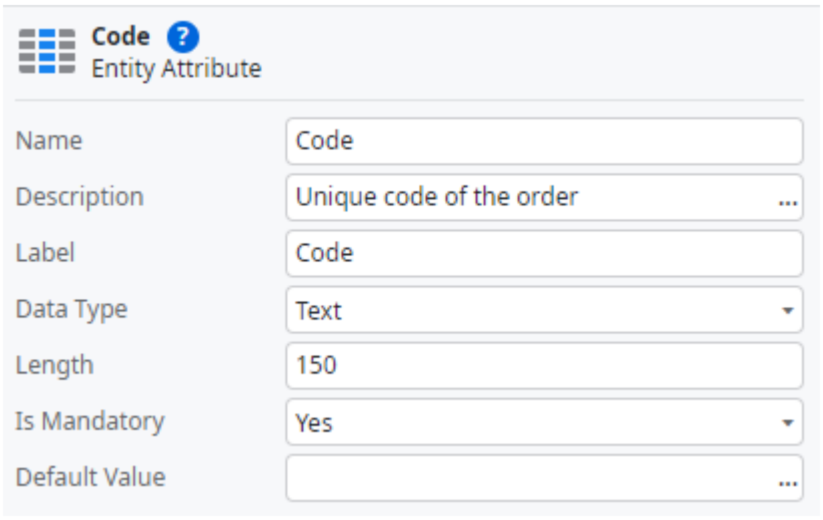


4) Click on attribute and type *Code* in the Name property.

5) Write a brief description of the attribute (this step is optional, but it will be helpful when you are looking at your project in the future, or if you a working in a team).

6) Click on the **Is Mandatory** property and change it to *Yes* in the dropdown.

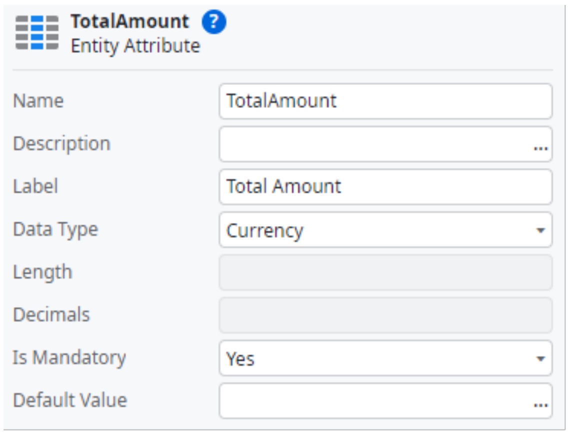
7) Make sure the **Data Type** is set to **Text**, and type the value *150* in the **Length** property. Your attribute should look like this:



Code
Entity Attribute

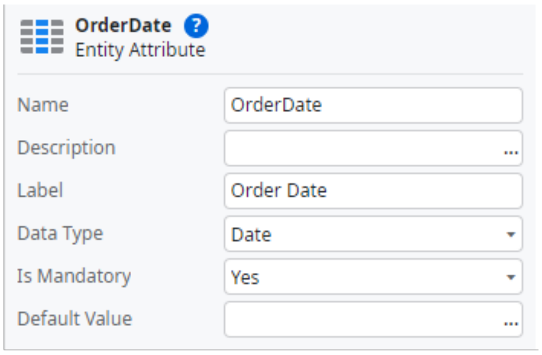
Name	Code
Description	Unique code of the order
Label	Code
Data Type	Text
Length	150
Is Mandatory	Yes
Default Value	

8) Repeat steps 3 to 7 and create the **mandatory** attributes *TotalAmount* and *OrderDate*, this time with data type set to **Currency** and **Date**.



TotalAmount
Entity Attribute

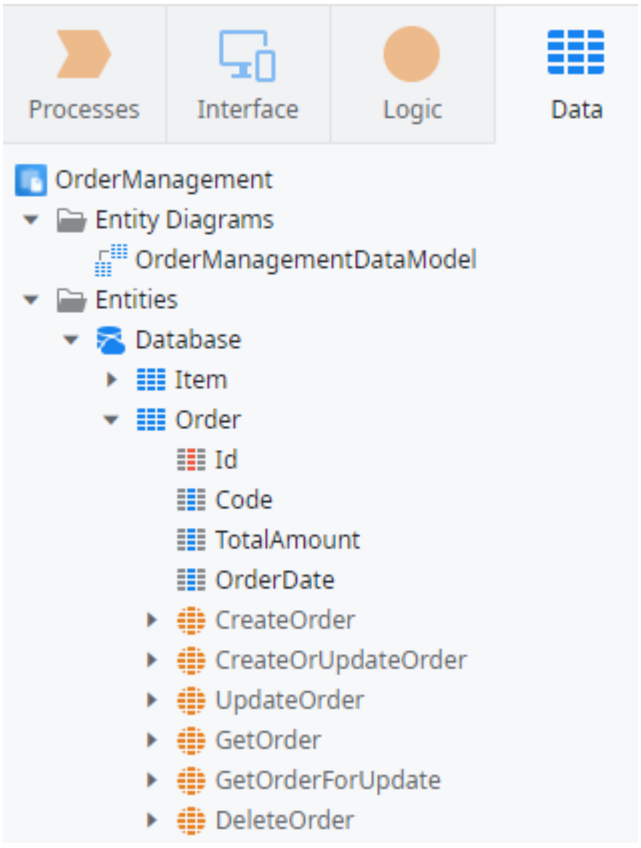
Name	TotalAmount
Description	
Label	Total Amount
Data Type	Currency
Length	
Decimals	
Is Mandatory	Yes
Default Value	



OrderDate
Entity Attribute

Name	OrderDate
Description	
Label	Order Date
Data Type	Date
Is Mandatory	Yes
Default Value	

Your Order Entity should look like this:

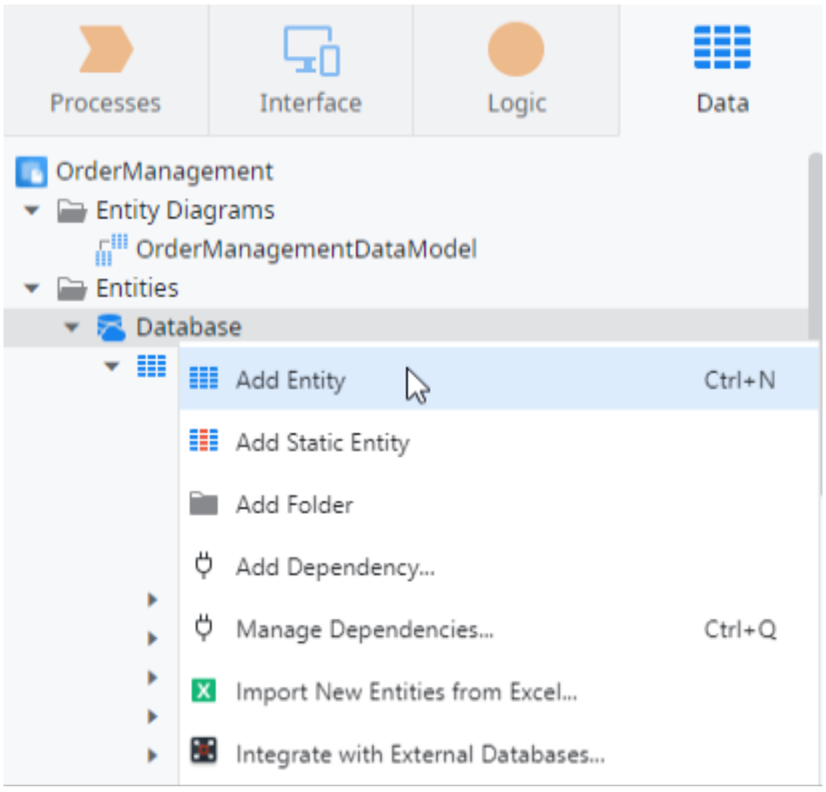


OrderManagement
- Entity Diagrams
  - OrderManagementDataModel
- Entities
  - Database
    - Item
    - Order
      - Id
      - Code
      - TotalAmount
      - OrderDate
      - CreateOrder
      - CreateOrUpdateOrder
      - UpdateOrder
      - GetOrder
      - GetOrderForUpdate
      - DeleteOrder

Create Relationships Between the Data

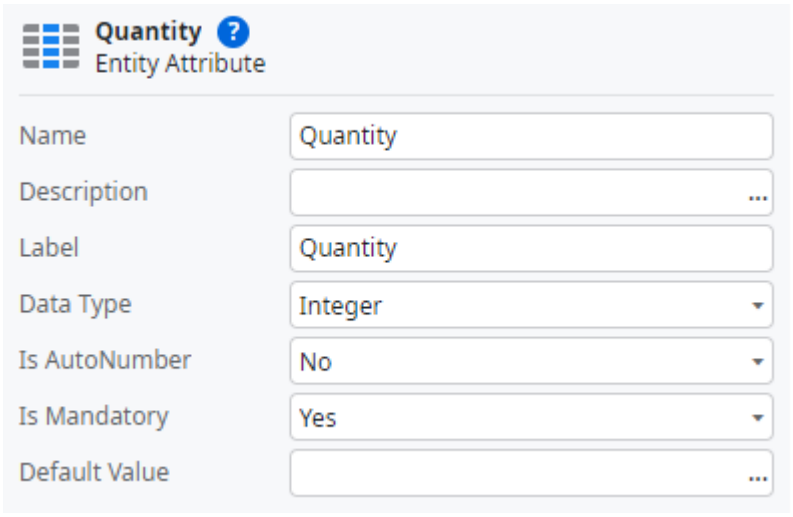
Now that you have created some Entities, it's time to create relationships between them. In this application, an Order can contain multiple Items, and an Item can belong to multiple Orders. So you will need to create an auxiliary Entity to create this relationship. We call this a *Many-to-Many* Relationship. You can learn more about Entity relationships in our documentation.

1) First, let's create a new Entity as we did in the previous section.



2) Name the new Entity *OrderItem*. This Entity will be used to connect one item to an Order.

3) Add a mandatory attribute called *Quantity*, with **Data Type** set to **Integer**.

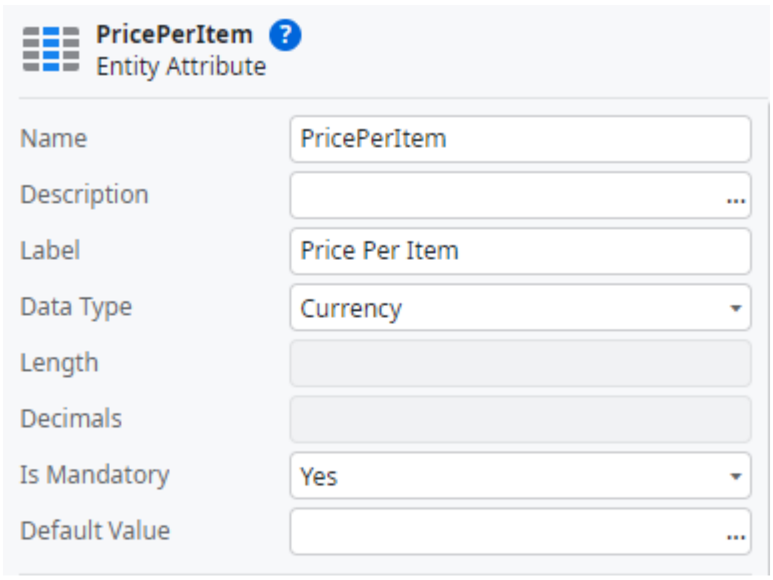


Quantity
Entity Attribute

Name	Quantity
Description	
Label	Quantity
Data Type	Integer
Is AutoNumber	No
Is Mandatory	Yes
Default Value	

This attribute will represent the quantity of a certain Item in an Order. For instance, an order can include 2 laptops or 3 sunglasses.

4) Add another mandatory attribute called *PricePerItem* with **Data Type** set as **Currency**.

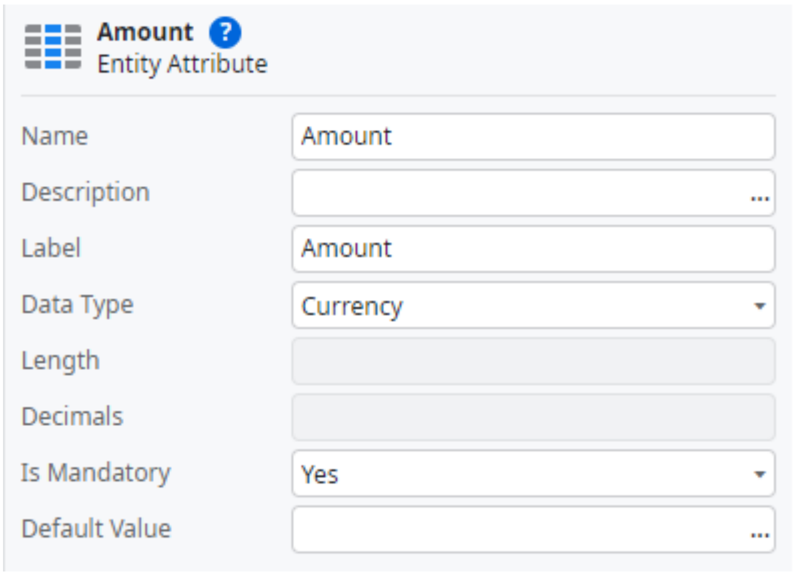


PricePerItem
Entity Attribute

Name	PricePerItem
Description	
Label	Price Per Item
Data Type	Currency
Length	
Decimals	
Is Mandatory	Yes
Default Value	

This attribute will store the unit price of each individual item. This will be useful later, when you create the logic to add products to an Order.

5) Add one last mandatory attribute called *Amount*, also with **Data Type** set to **Currency**.

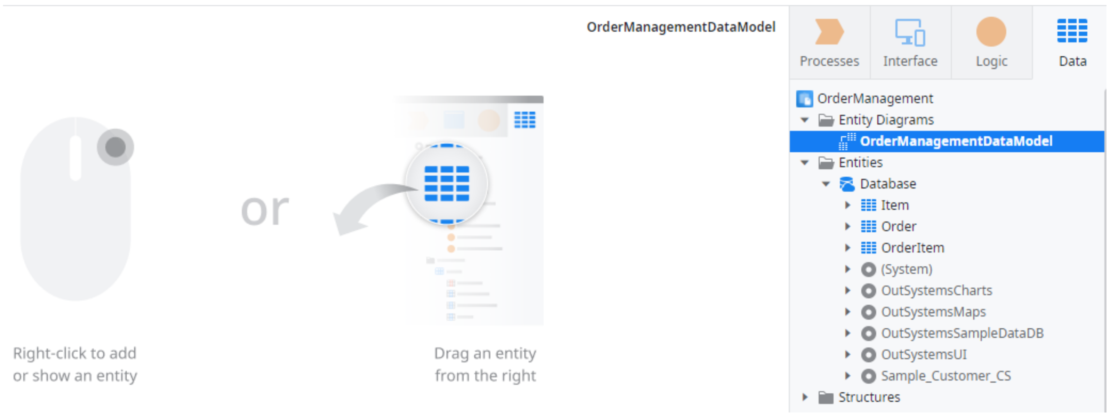


Amount
Entity Attribute

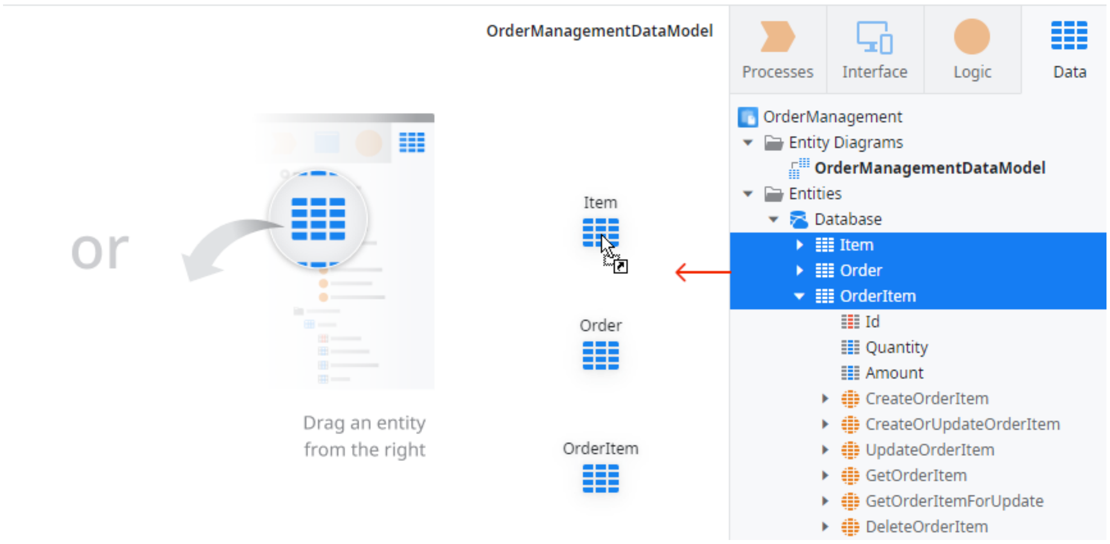
Name	Amount
Description	
Label	Amount
Data Type	Currency
Length	
Decimals	
Is Mandatory	Yes
Default Value	

This attribute will store the total price of that Item type in an order. Let's imagine you have an order with 2 speakers that cost $100 each. In this case the amount would be $200.

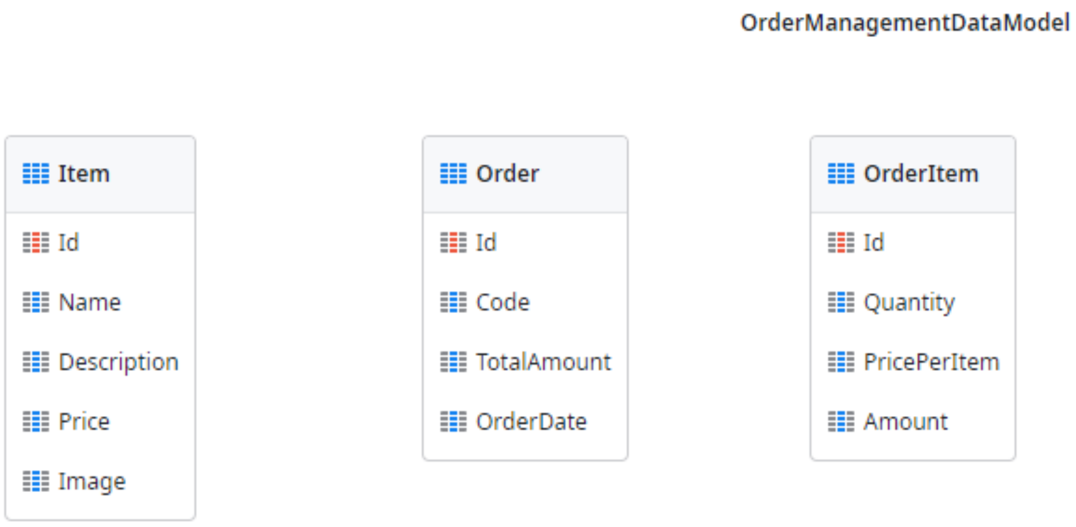
6) Still in the Data tab, expand the **Entity Diagrams** folder (if it is not expanded already). Double-click the **OrderManagementDataModel** to open it.



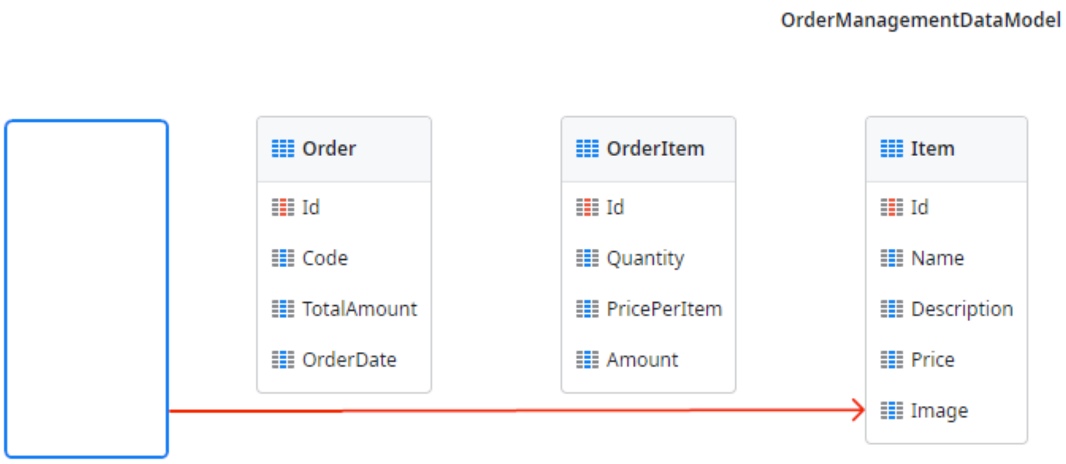
7) Select the **Order**, **OrderItem** and **Item** Entities (you can use CTRL+click to select several elements). Drag and drop the selected Entities to the empty data model you have just opened.



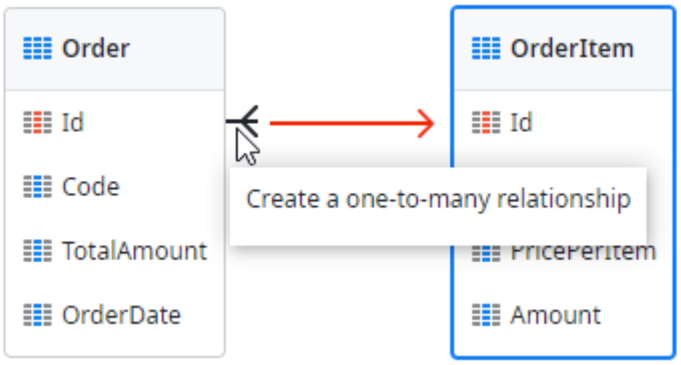
You will have something like the image below.



8) Click on the Item Entity in the diagram and drag it to the right side of the OrderItem Entity. This step is optional, but helps us visualize the data model better.

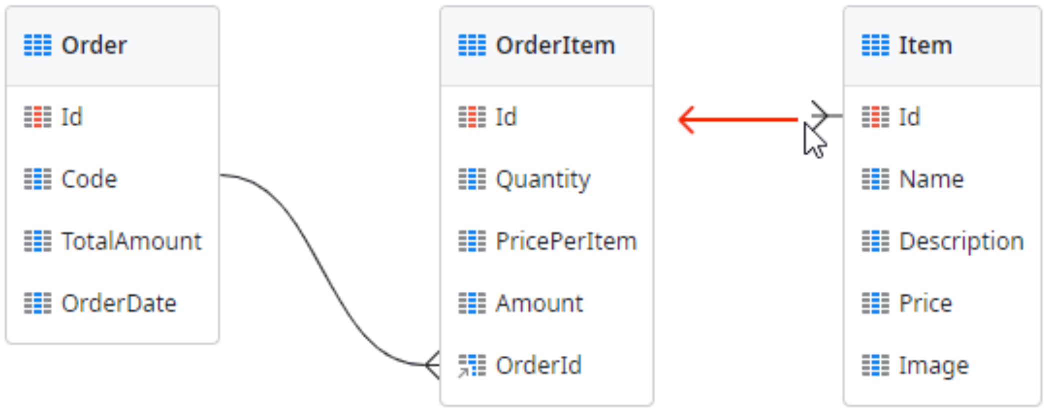


9) Hover the mouse in the right side of the Order Entity, select the connector that appears close to the **Id** attribute, and drag it to the OrderItem Entity.

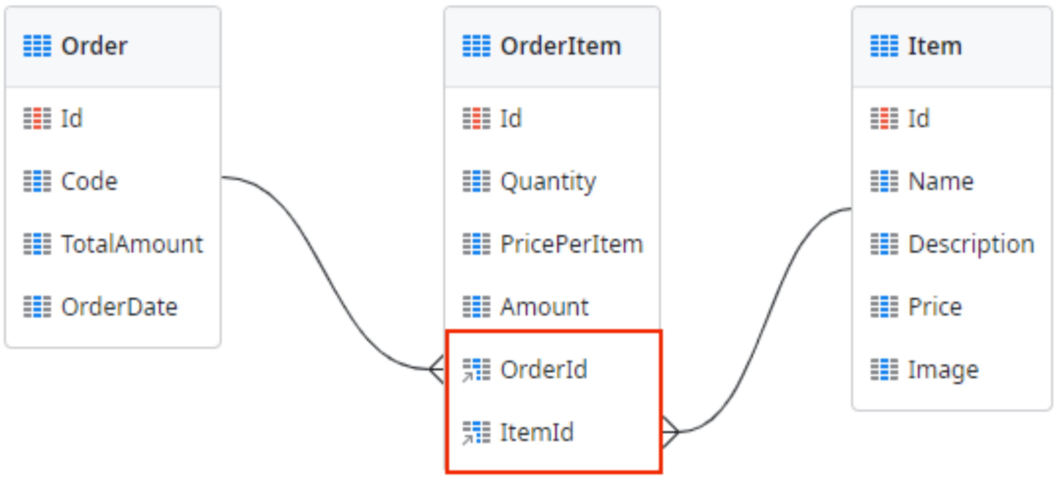


A new attribute was created in the OrderItem Entity called *OrderId*. This creates a relationship between the two Entities, meaning that an Item is associated with one Order.

10) Repeat the same step for the **Item** Entity. Don't forget to drag **from** the Item Entity **to** the OrderItem.



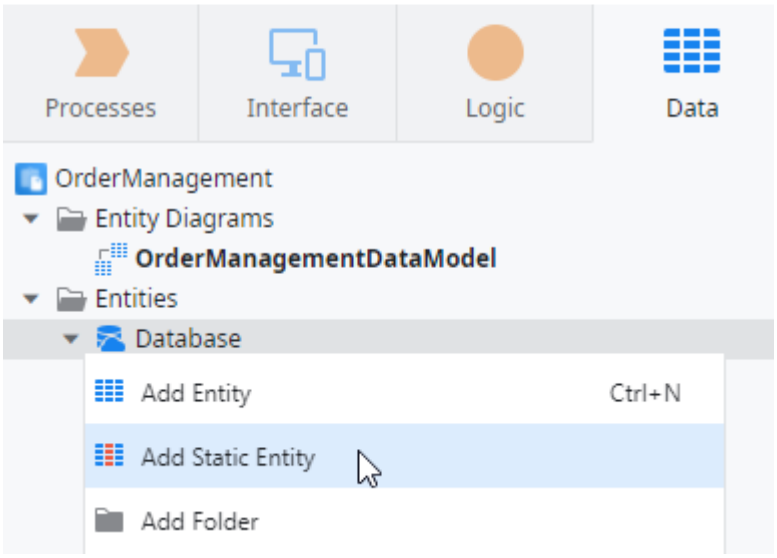
Notice that the icons in the two new attributes are slightly different from the others. This happens because they are attributes linked to another entity, also known as foreign keys. If you click on the attribute you can see they have special identifiers.



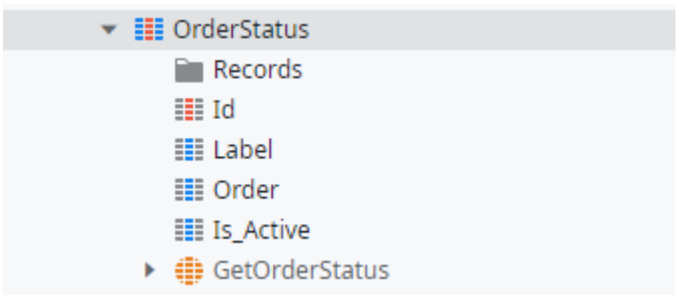
Add a Status to the Order

An Order can have different statuses, such as "Draft", "Submitted", "Approved", and "Rejected". Since these values are limited to the values we already know, we can use a Static Entity here. A Static Entity consists of a set of named values that cannot be changed during execution time.

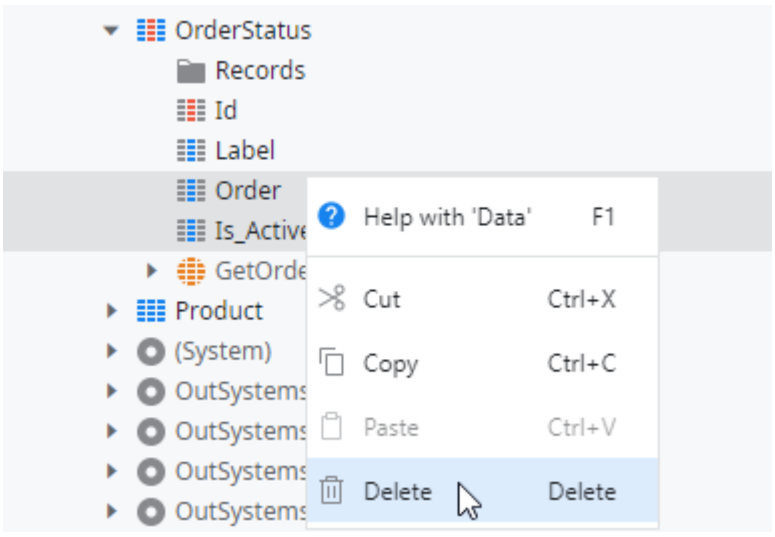
1) In the Data tab, right-click the Database and select **Add Static Entity**.



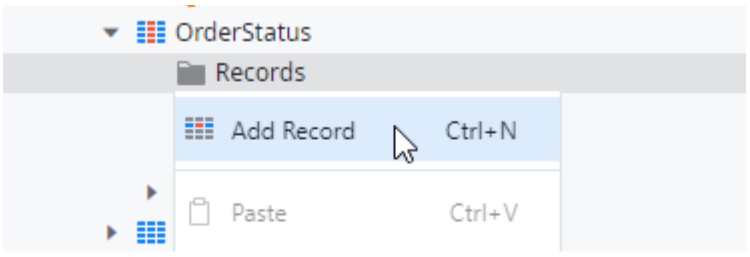
2) Name the Entity *OrderStatus*. Notice that four attributes are automatically created: Id, Label, Order, and Is_Active.



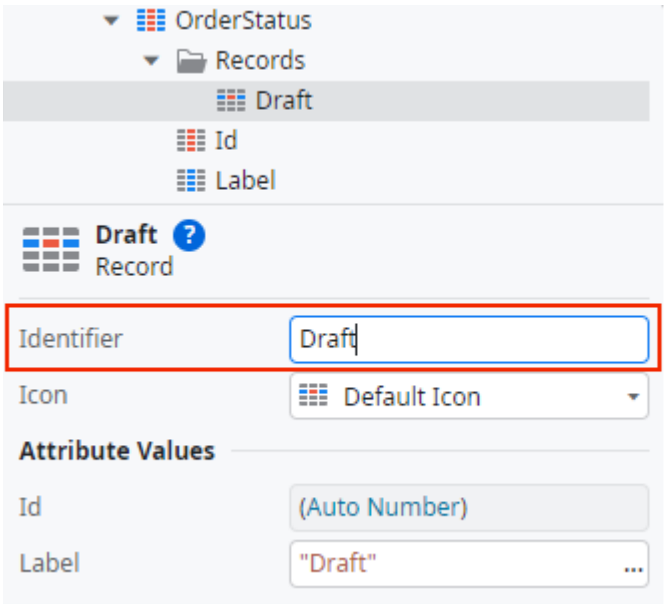
3) Select the **Order** and **Is_Active** attributes, right-click them and select **Delete**, sincet they won't be used. You can also use the delete key from your keyboard.



4) Now let's add some values to our statuses. Right-click the OrderStatus Entity and select **Add Record**.



5) Type **Draft** in the Identifier.



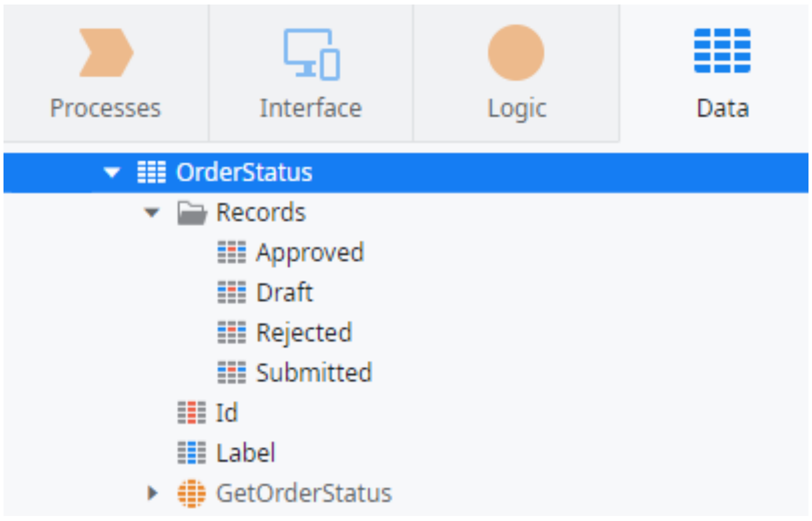
Draft Record

Identifier	Draft
Icon	Default Icon

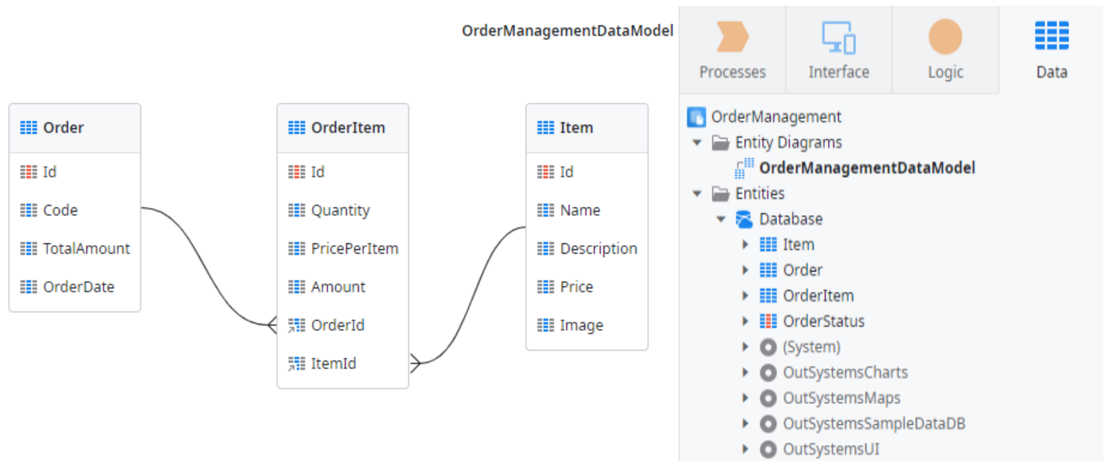
Attribute Values

Id	(Auto Number)
Label	"Draft"

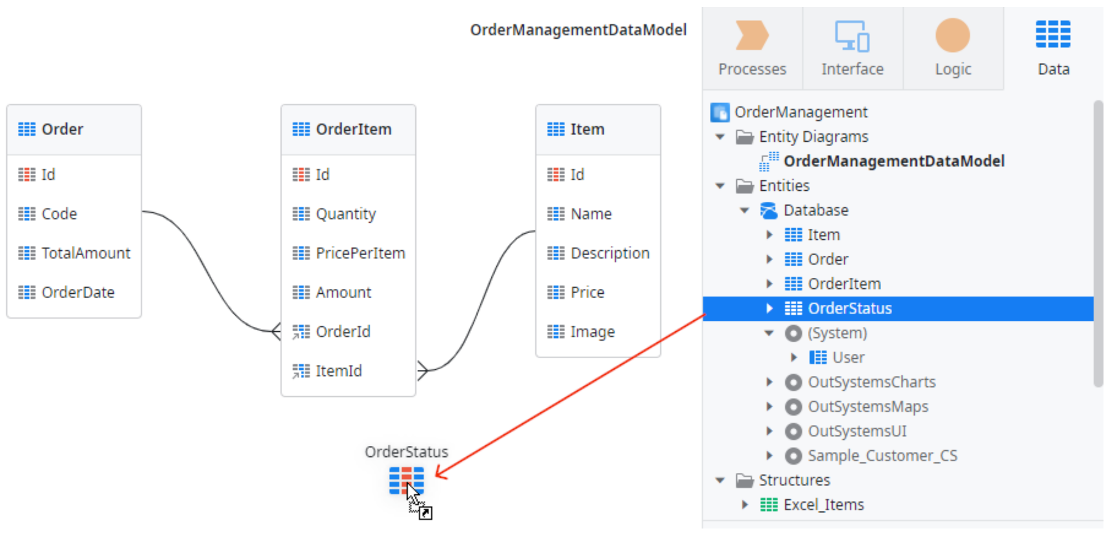
6) Repeat the previous 2 steps and create the records *Submitted*, *Approved* and *Rejected*.



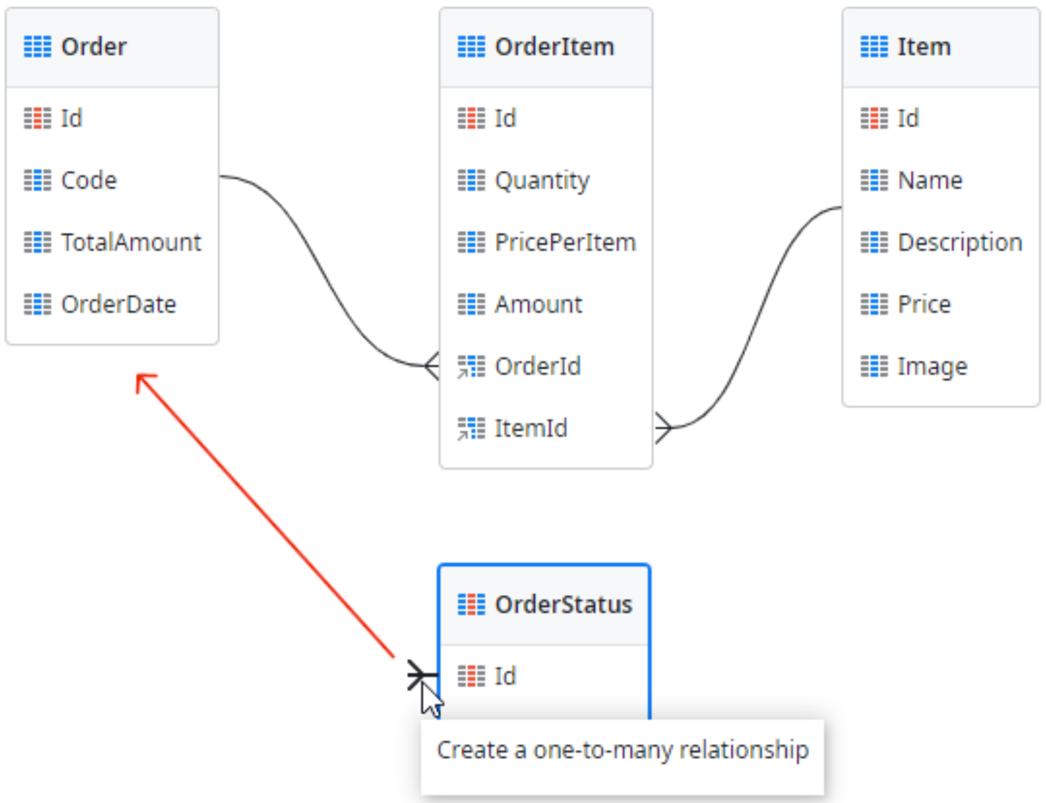
7) Open the Data Model again by double-clicking it.



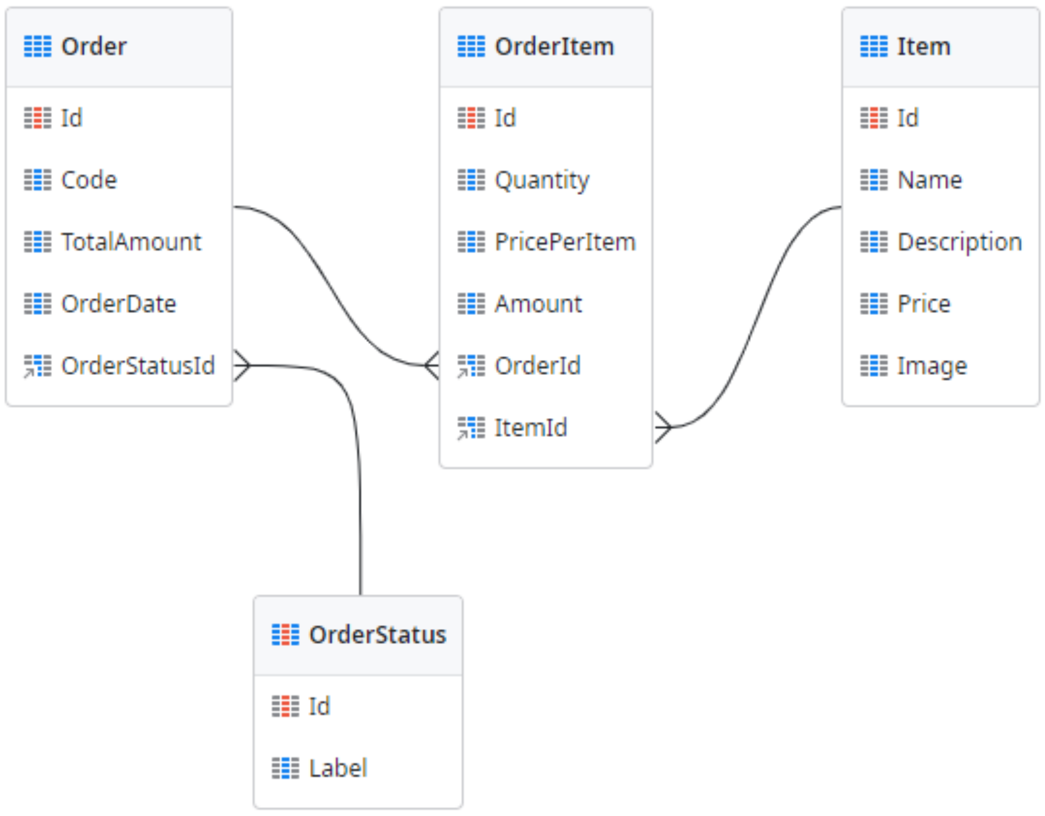
8) Drag the **OrderStatus** Entity to the data model.



9) Drag the connector **from** the OrderStatus Entity **to** the Order Entity to create a relationship between these two Entities.



10) Now each Order has one status. And we finished our data model!



Publish the Module

The data model is now ready, so now we need to publish this module in the OutSystems server to actually save the data in the database, and to be able to view it in our application.

1) Publish the module by clicking on the **Publish** button.

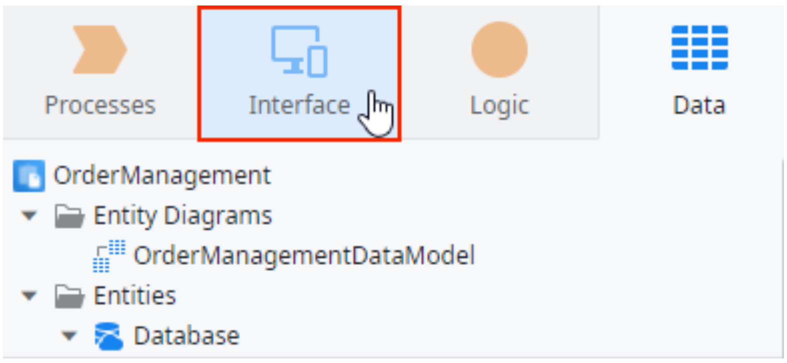
The publishing process will upload your module to the server, compile and generate the code, update the database and deploy it. When the process is over, the app is ready to be accessed by your users.

At this point we don't have any interface yet to actually see the data or interact with the app. Let's continue to the next section to add some Screens.

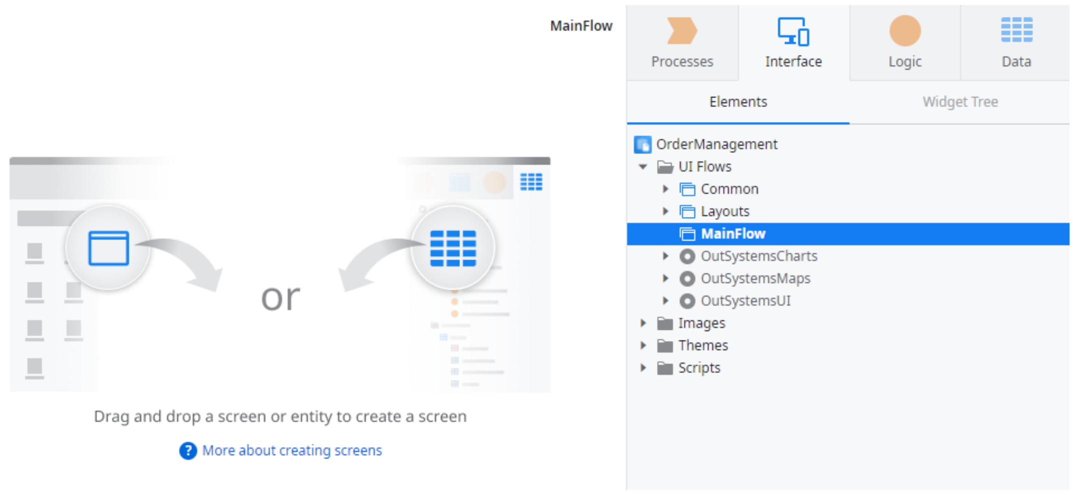
Create Your First Screens

Now that you have created your data model, it is time to work on the interface of the web application. Let's see how we can quickly create two screens in just a few simple steps!

1) In the top-right corner, switch to the Interface tab by clicking on the corresponding icon.

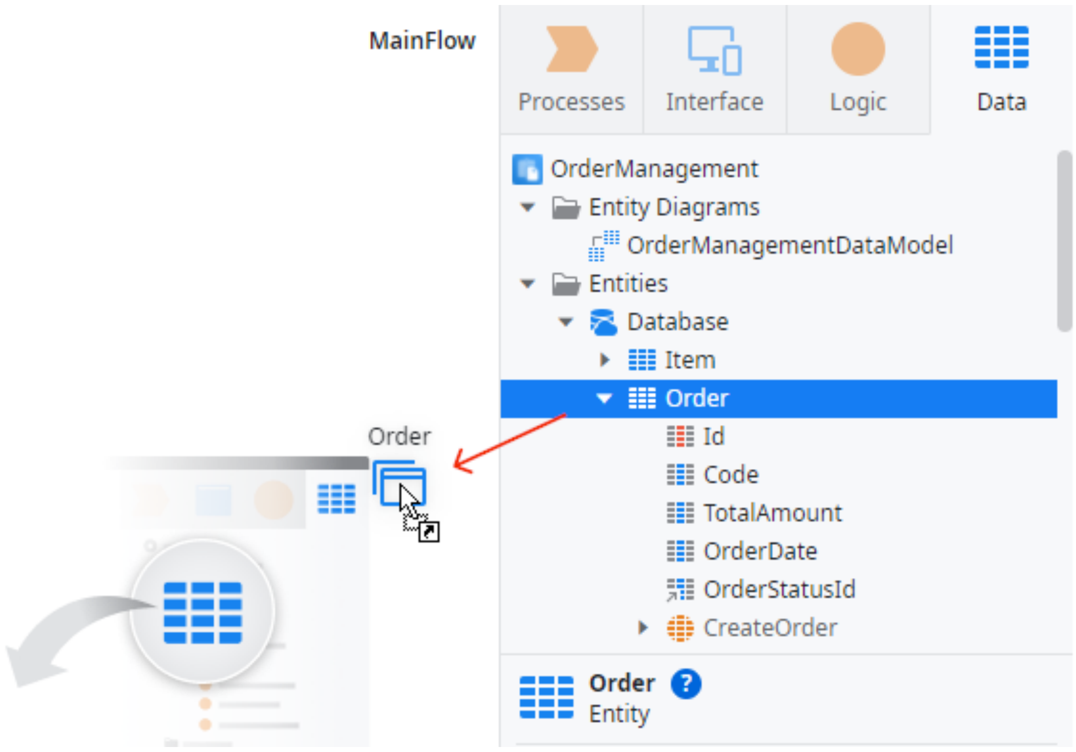


2) Double-click the **MainFlow** element

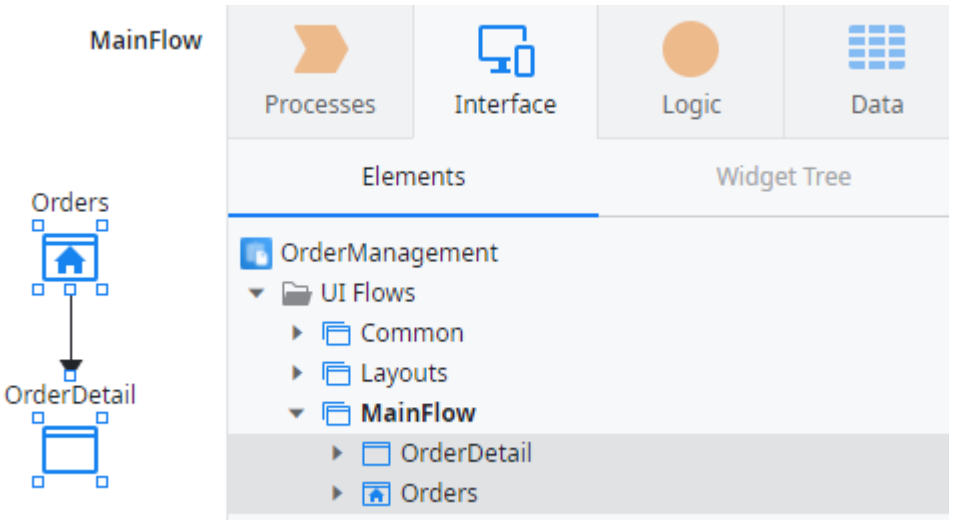


The MainFlow is where your custom Screens will be created. For now, we only have the Login Screen, but the MainFlow is empty.

3) Switch back to the **Data** tab, then drag and drop the **Order** Entity in the MainFlow area, just like the following image.

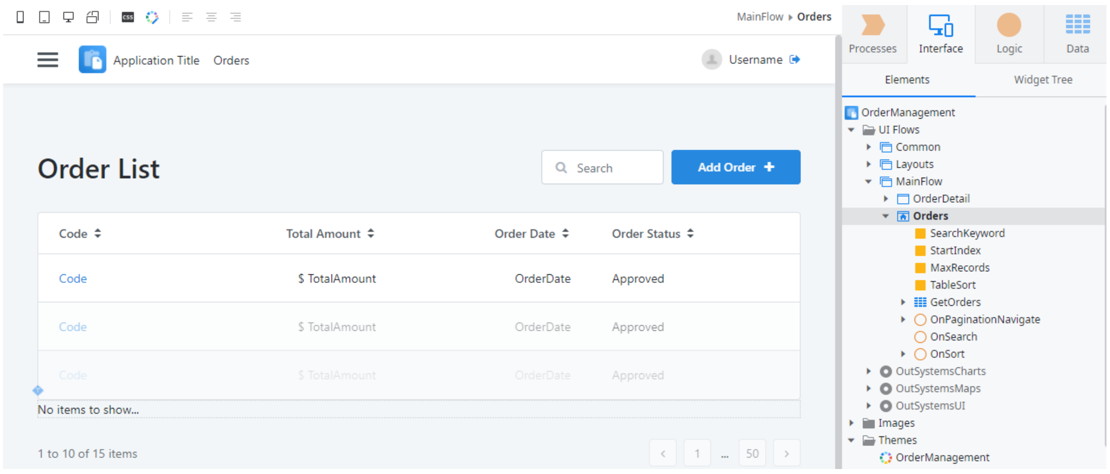


And just like that, you have created two Screens!

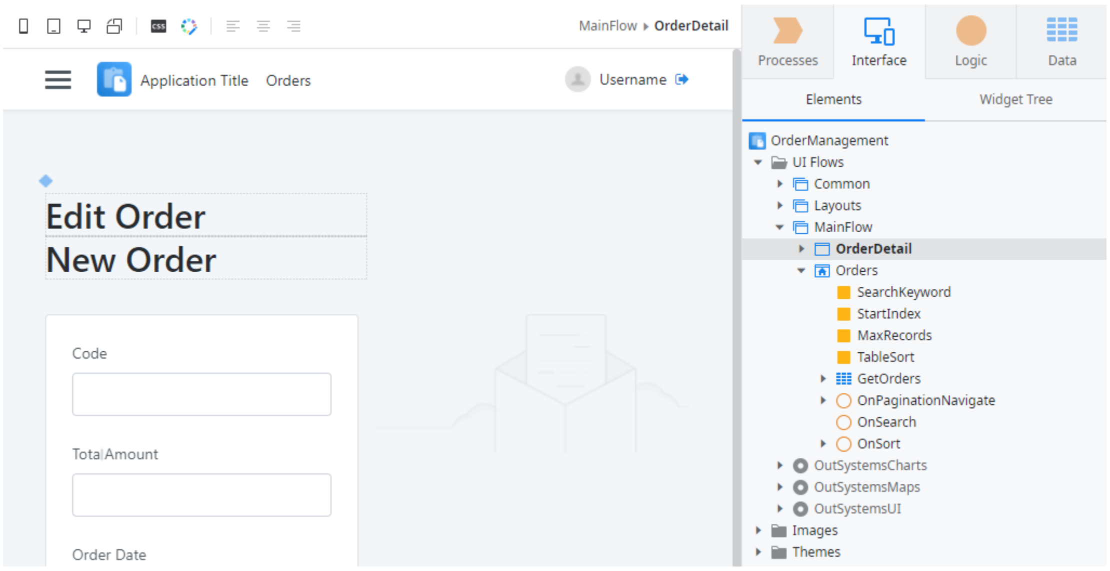


You can rename them if you want, but for now, we will use the automatically created names.

4) Double-click the **Orders** Screen to see what was created. You can see that the Screen was automatically created with all the functionality that we need: list of orders, search by name, pagination and the option to add a new order.



The same thing happens with the **OrderDetail** Screen.



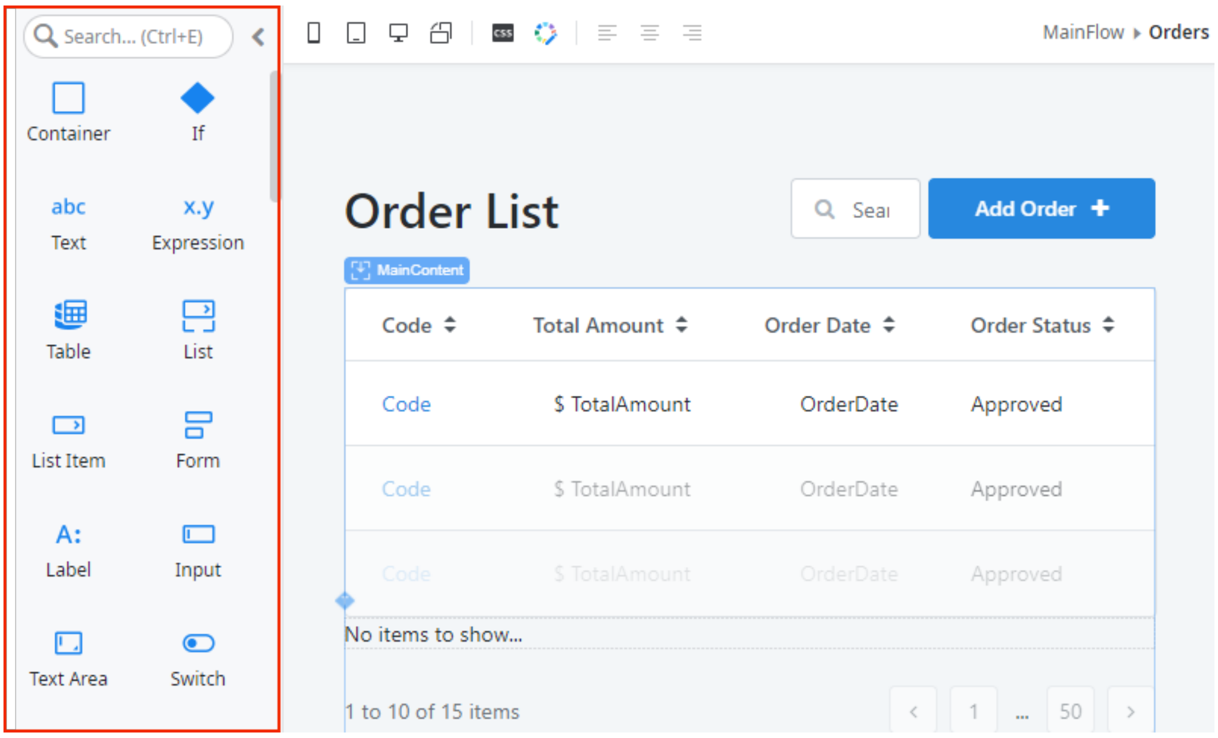
OutSystems uses the attributes in the Entity and creates a list and a detail Screen accordingly, following the application's layout. This saves us a lot of time and can be customized if needed.

5) **Publish** the module again to save the last changes you did.

Using a Color Code

In the previous step, you created two Screens automatically. Now you will customize some elements of the Orders Screen. More specifically, you will add a color code for different order statuses.

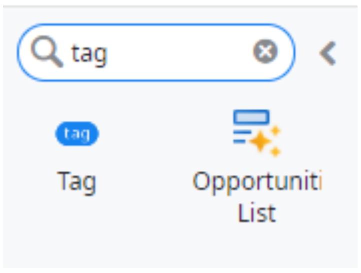
On the left side of Service Studio, you can see a bar with many elements that can help you build and customize your Screens. This bar is called **Toolbox** and the elements **Widgets**. You will use some Widgets in the next few steps.



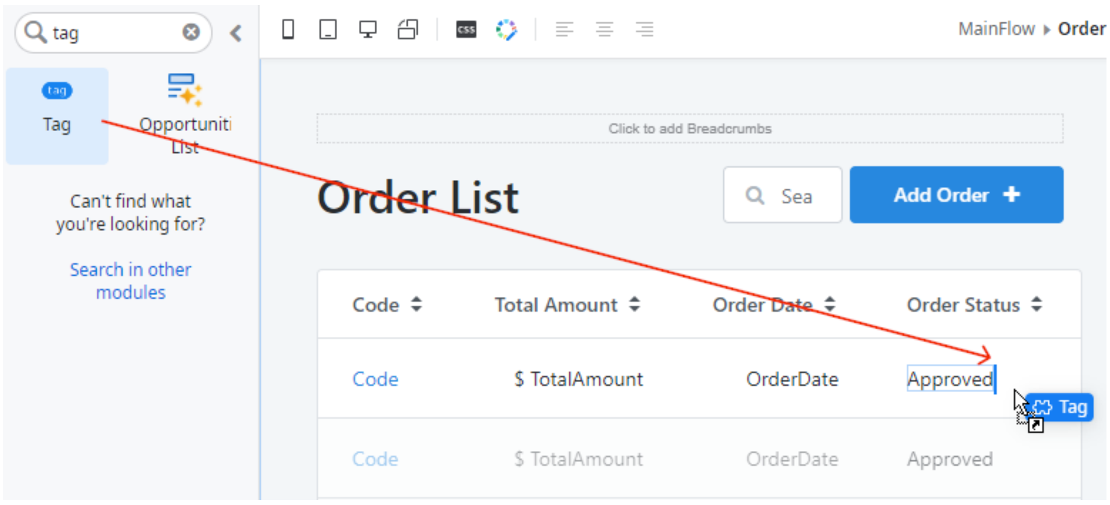
Code	Total Amount	Order Date	Order Status
Code	$ TotalAmount	OrderDate	Approved
Code	$ TotalAmount	OrderDate	Approved
Code	$ TotalAmount	OrderDate	Approved

Now you will customize the color of the status order using a color code for each type of status.

1) Search for a **Tag** Widget in the Toolbox.



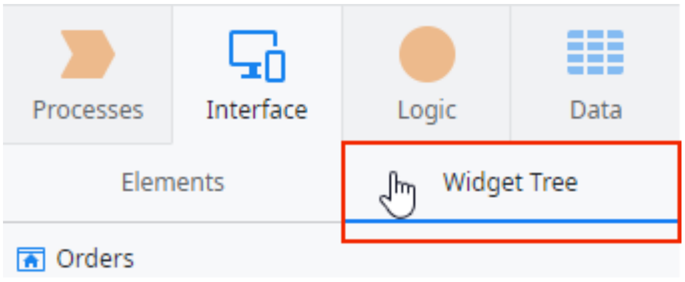
2) Drag and drop it on the right side of the first Order Status cell that says *Approved*.



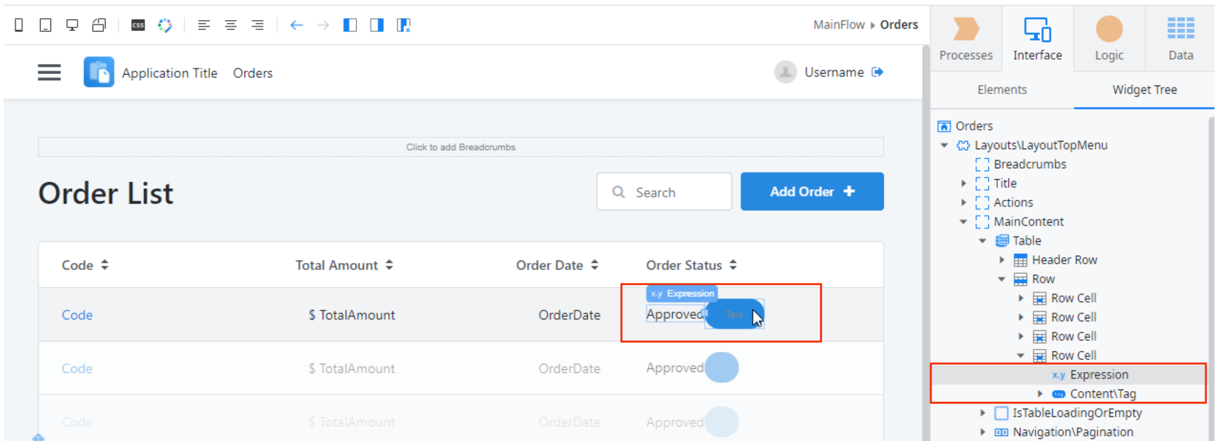
You will notice that the text is not inside the Tag. So you need to drag the text inside the tag. You can have more precision by using the **Widget Tree**.

The Widget Tree is available when we have a Screen opened on Service Studio. It shows the structure of all the elements inside the Screen, and it is a great helper to set the pieces right where they belong.

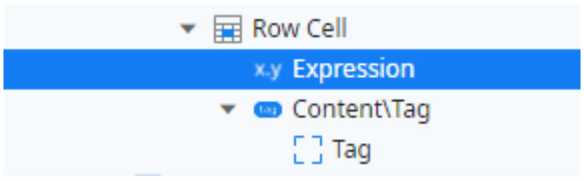
To do so, click on Widget Tree on the Interface tab.



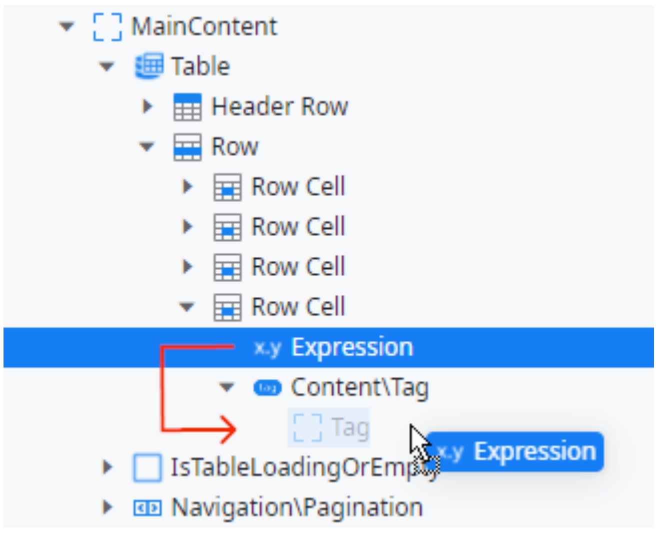
When you click on an element of the Screen, you can see it inside the Widget Tree as well.



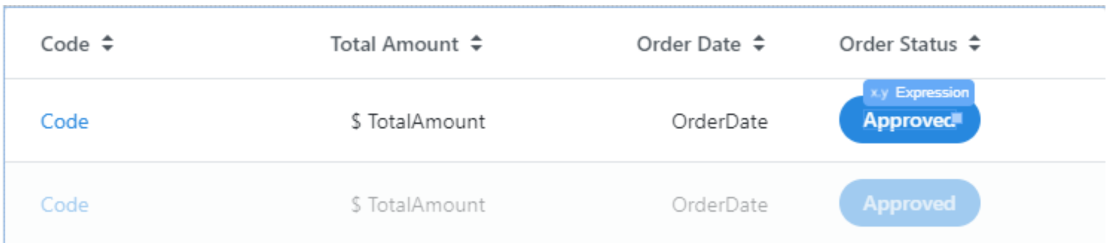
1) In the Widget Tree, expand the **Content\Tag** element so you can see its structure.



2) Drag the Expression (which is where the status text is) to the **Tag** element inside the **Content\Tag**.



Your Screen should look like this:

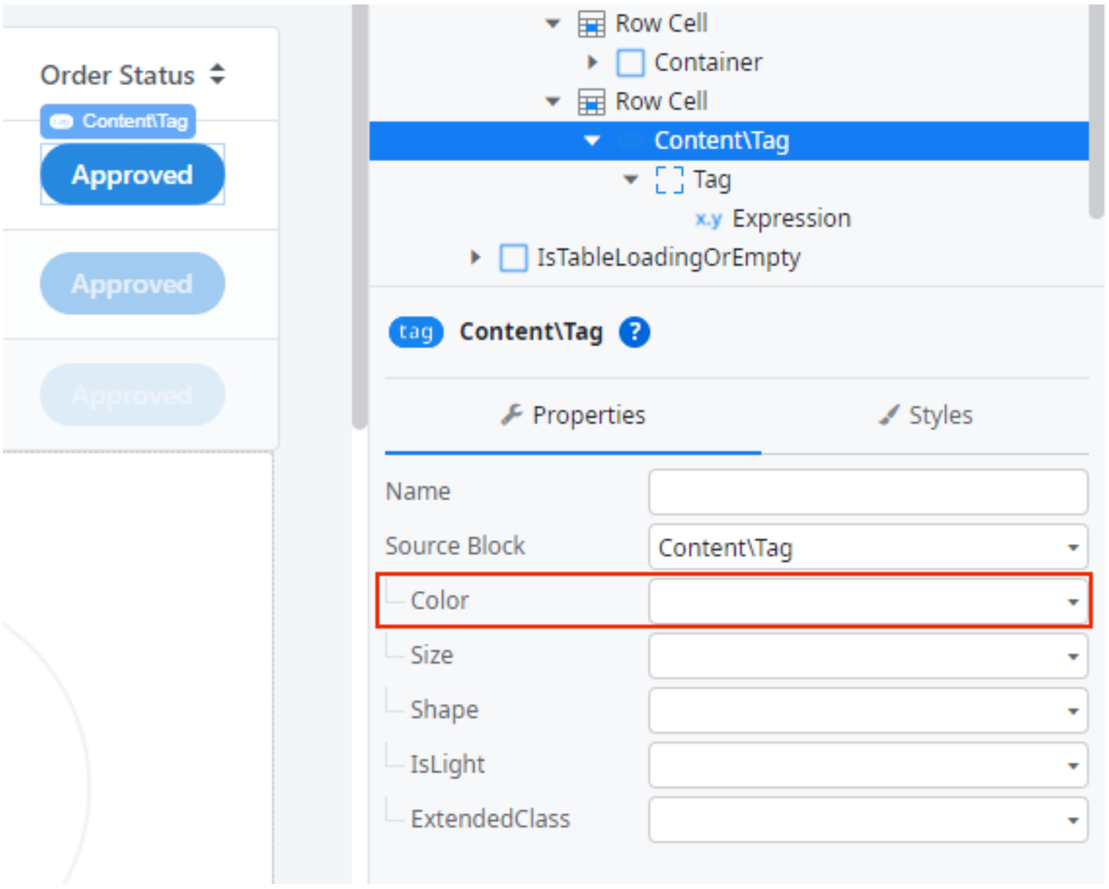


Code	Total Amount	Order Date	Order Status
Code	$ TotalAmount	OrderDate	Approved
Code	$ TotalAmount	OrderDate	Approved

Now you will create a condition to define different colors for different statuses. You can use the suggested color code below or a custom one.

- Draft: Primary color of the app (blue)
- Submitted: Yellow
- Approved: Green
- Rejected: Red

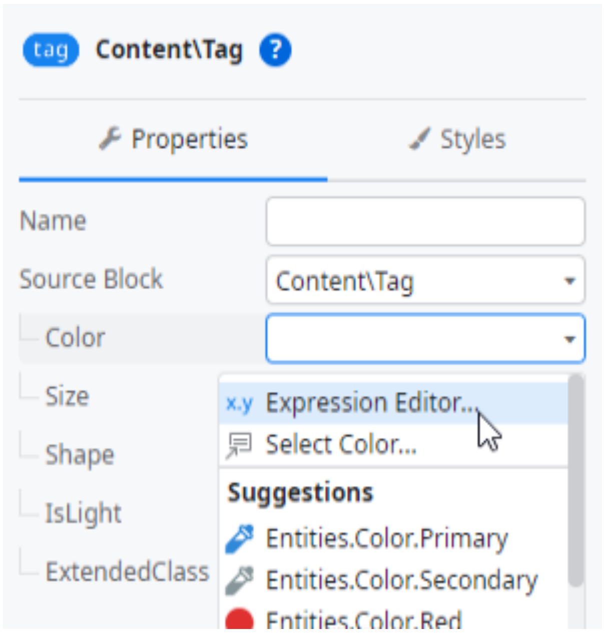
3) Click on the Tag to see its properties and find the **Color** property.



Content\Tag

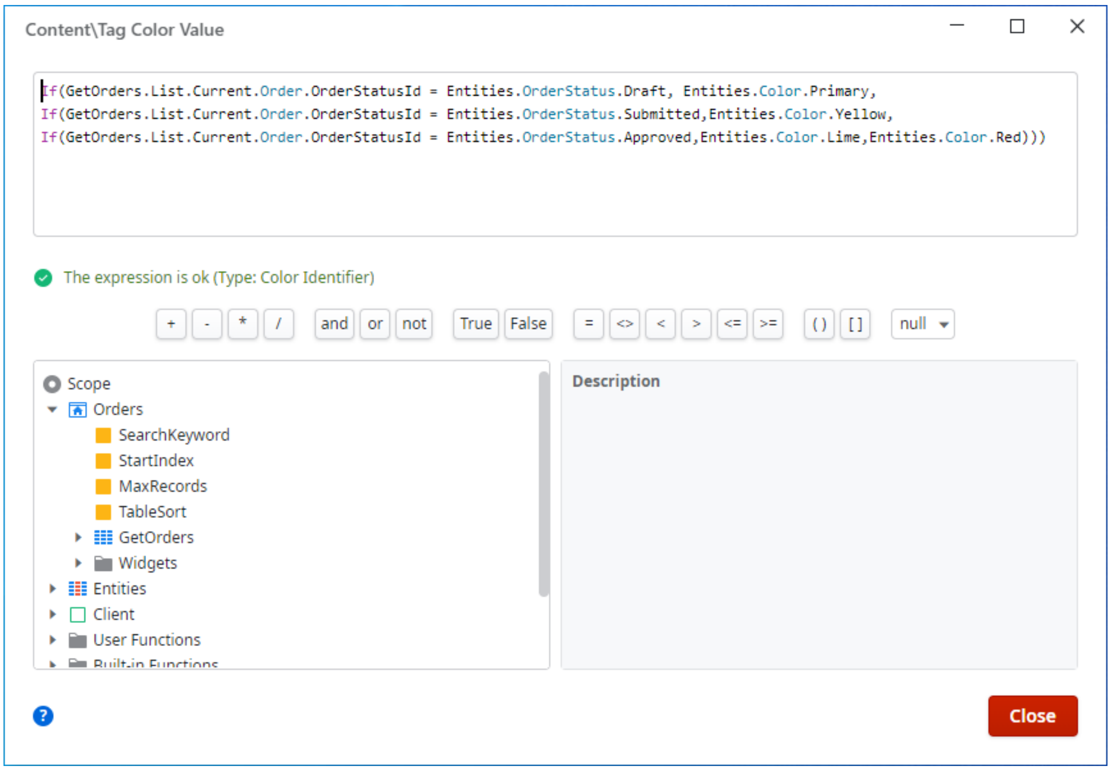
Name	
Source Block	Content\Tag
Color	
Size	
Shape	
IsLight	
ExtendedClass	

4) Expand the dropdown of the Color property, and select **Expression Editor...**.



5) Copy the following expression into the dialog that was opened:

```
If(GetOrders.List.Current.Order.OrderStatusId =
Entities.OrderStatus.Draft, Entities.Color.Primary,
If(GetOrders.List.Current.Order.OrderStatusId =
Entities.OrderStatus.Submitted,Entities.Color.Yellow,
If(GetOrders.List.Current.Order.OrderStatusId =
Entities.OrderStatus.Approved, Entities.Color.Lime,
Entities.Color.Red)))
```



```
If(GetOrders.List.Current.Order.OrderStatusId = Entities.OrderStatus.Draft, Entities.Color.Primary,
If(GetOrders.List.Current.Order.OrderStatusId = Entities.OrderStatus.Submitted,Entities.Color.Yellow,
If(GetOrders.List.Current.Order.OrderStatusId = Entities.OrderStatus.Approved,Entities.Color.Lime,Entities.Color.Red)))
```

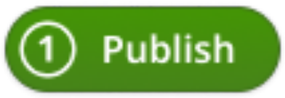
The expression is ok (Type: Color Identifier)

Wow! This expression is quite long! However, what we're saying is that if the status of the order is draft, the tag will have the Primary color of the app. If it's submitted, we will have the tag appearing in yellow, if it's approved the tag will be green and if it's rejected the tag will be red.

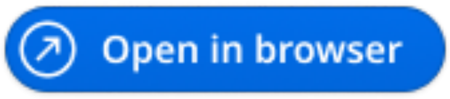
Testing in the Browser

Even though you changed some aspects of the Orders Screen, they will not be visible until you add orders to it. So let's do that.

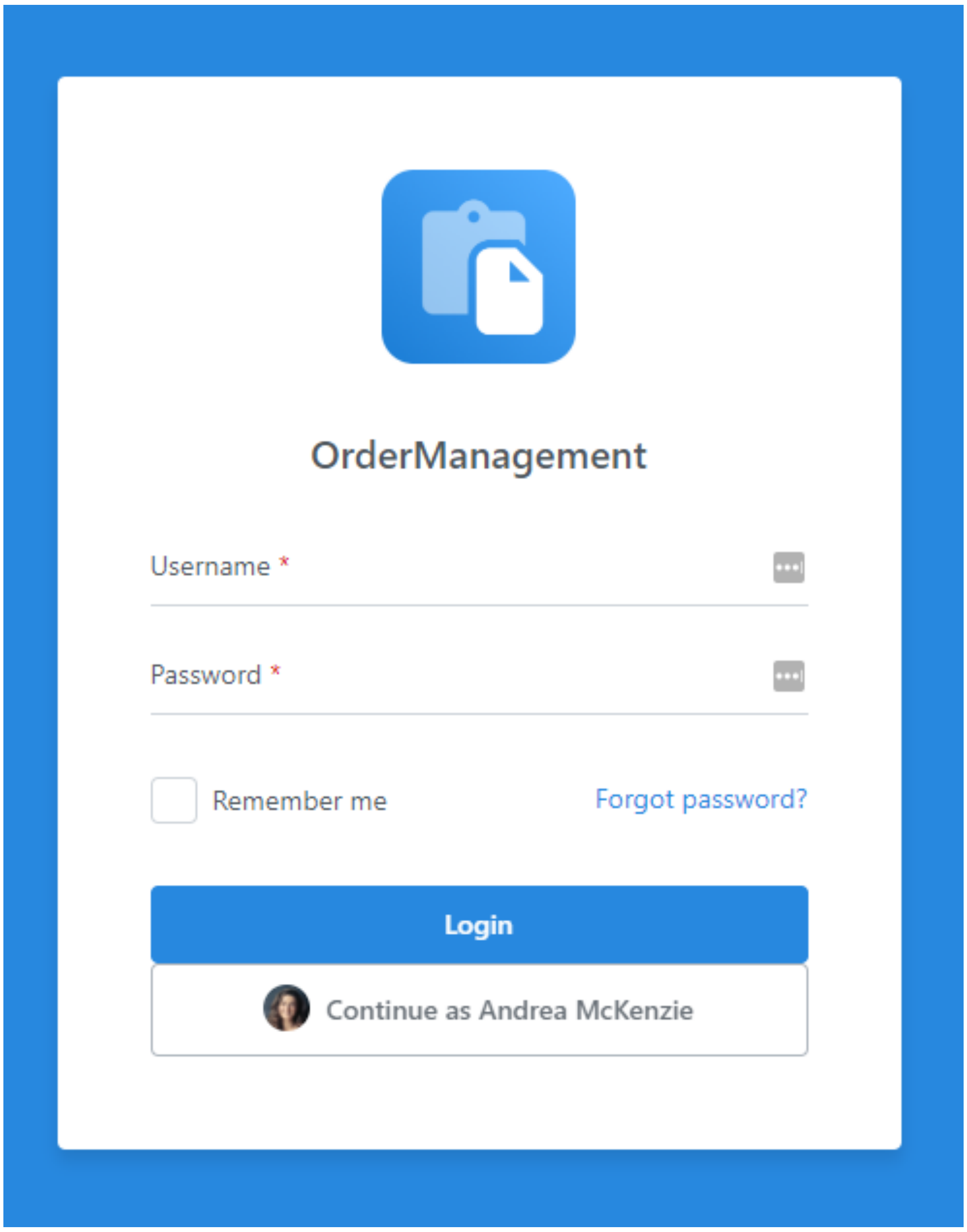
1) Click on the Publish button.



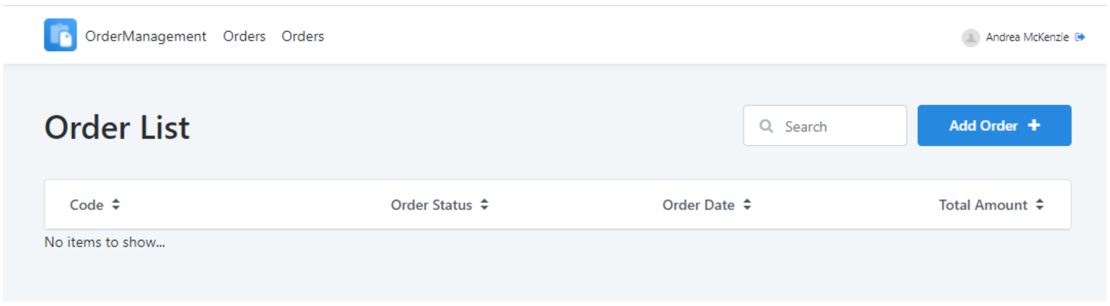
2) When it is done publishing, click on the blue button that appears where the Publish button was. This will open the app in the browser.



3) You should see a login screen. Use the sample login option available to log in. This sample user is used to emulate the app's behavior as a logged-in user.



Your app should look like this when the Order List is empty:

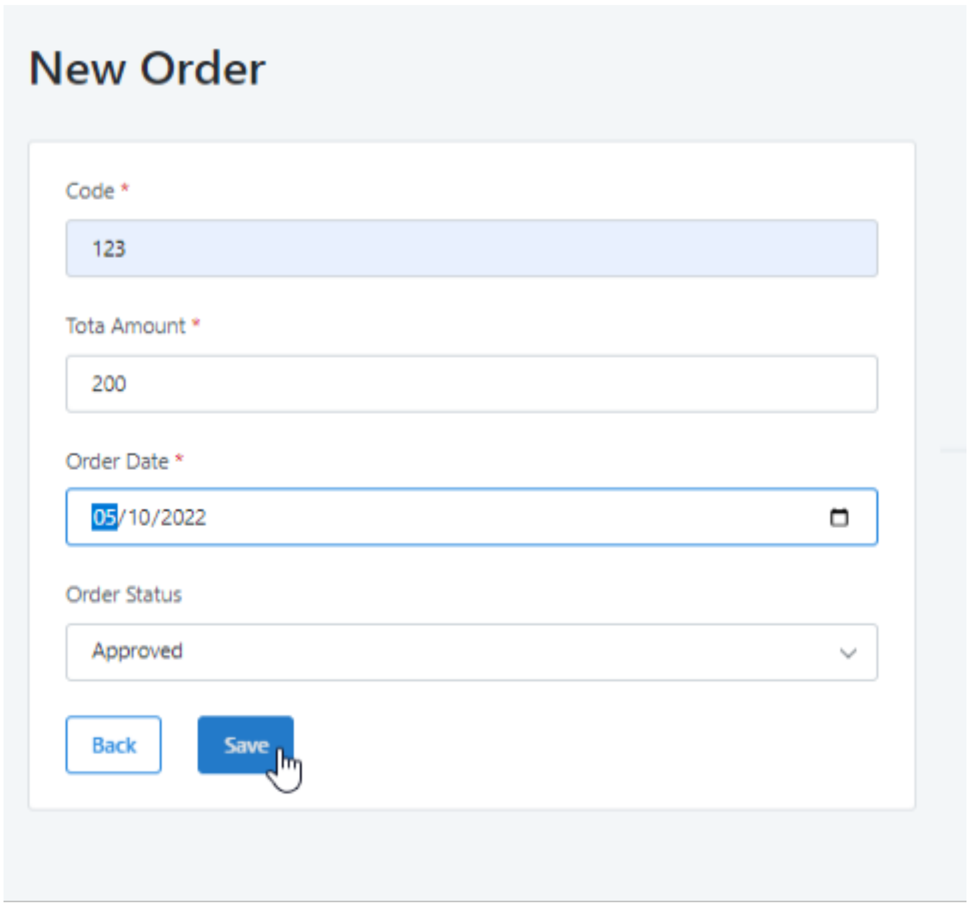


Code	Order Status	Order Date	Total Amount

No items to show...

4) Click on the **Add Order** button.

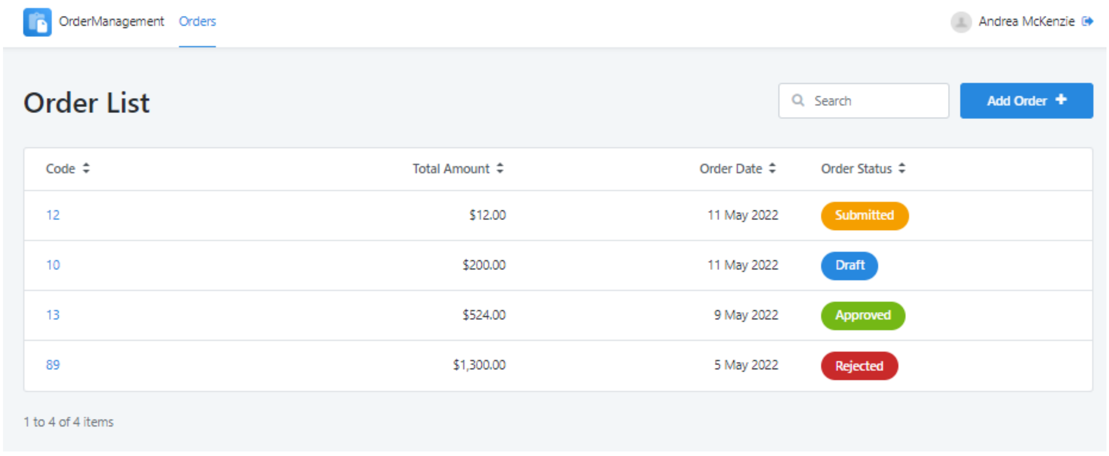
5) Write some test values and press the save Button to create a new Order.



New Order

Code *	123
Tota Amount *	200
Order Date *	05/10/2022
Order Status	Approved

6) Create one order for each order status to see your custom colors.



Order List

Code	Total Amount	Order Date	Order Status
12	$12.00	11 May 2022	Submitted
10	$200.00	11 May 2022	Draft
13	$524.00	9 May 2022	Approved
89	$1,300.00	5 May 2022	Rejected

1 to 4 of 4 items

Wrapping up

Congratulations on finishing this tutorial. With this exercise, we had the chance to go through some essential aspects of OutSystems, and even and get to know more about the platform.

References

If you want to deep dive into the subjects that we have seen in this exercise, we have prepared some useful links for you:

1) OutSystems Overview

2) Service Studio Overview

3) Intro to OutSystems Development

4) Modeling Data

5) Entity Relationships

6) Bootstrap and Entity Using an Excel File

7) Tag Widget

See you in the next tutorial!