

# Creating an Item Catalog

## Table of Contents

<b>Outline.....</b>	<b>2</b>
Scenario	2
Items Catalog Screen	2
<b>How-to.....</b>	<b>5</b>
Getting Started	5
Outdated Dependencies	5
Creating the Items Screen	7
Using a UI Accelerator	10
Replacing Data	12
Bring Back the Image	13
Final touches	15
<b>Wrapping up.....</b>	<b>20</b>
References	20

# Outline

In this tutorial, you will continue to extend the Order Management application. This time with the creation of an Item Gallery Screen. This Screen will display all the Items in the database, as well as their main information, such as name, description, price, and image, in a gallery format.

You will implement some requirements from scratch, but also use some accelerators along the way.

## Scenario

So far, the Order Management app has two Screens:

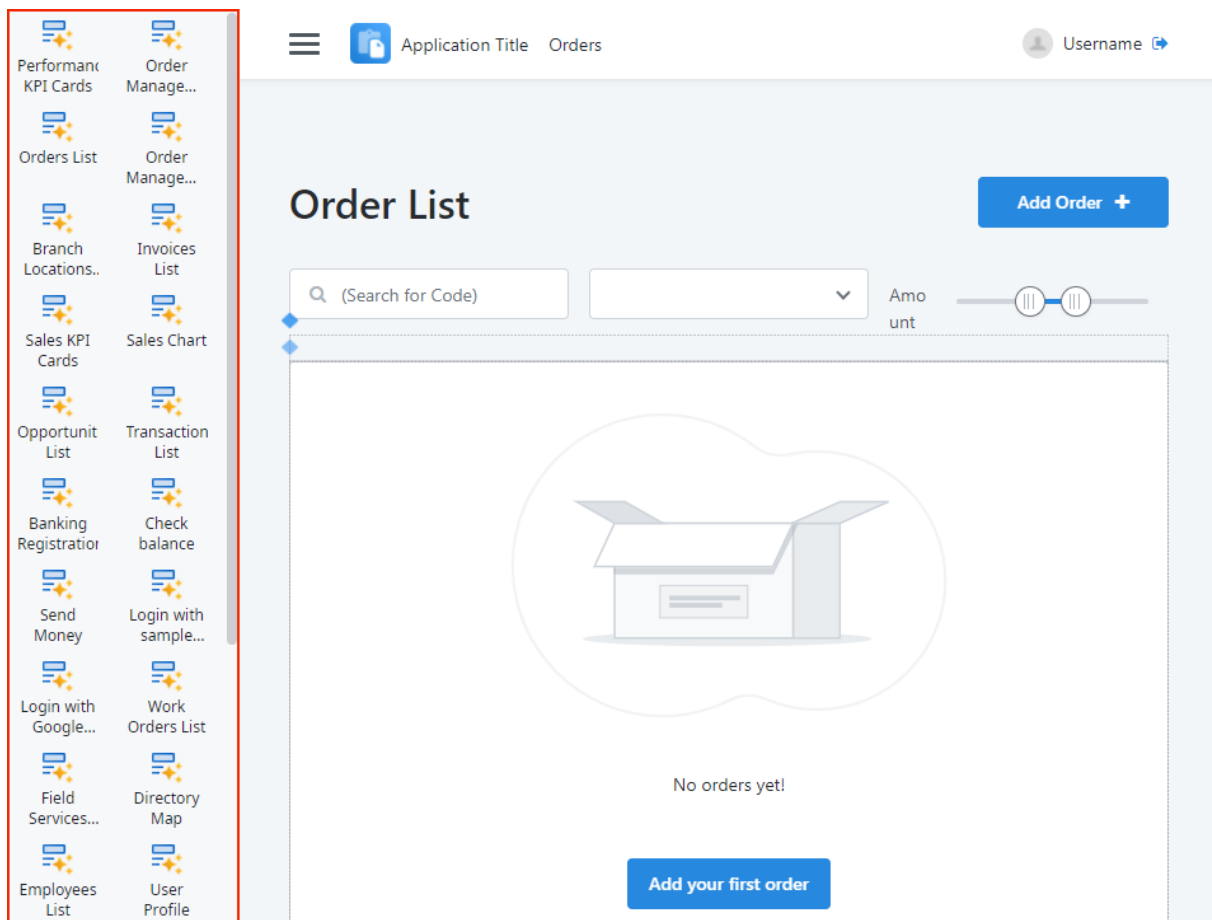
- Orders: lists all the orders and their most important information.
- Order Detail: has a Form that allows viewing and editing relevant information of an order, such as date, status, and order code.

In this tutorial you will create a new Screen to display all Items (products) in a gallery.

## Items Catalog Screen

The Items Screen will be created from scratch. This means you will start from an empty Screen and then add elements to it. But that does not mean more work! You will use an **Accelerator** to create complex and functional UIs with a few clicks.

The OutSystems Accelerators allow you to develop Reactive web and Mobile apps faster. Each Accelerator contains building blocks that you can add to your apps, and then modify them as you need.



In the Order Management application, you will need to do two major steps to achieve your objective:


- Create a new Screen and use a Product Gallery Accelerator
- Replace the sample data that is created by the accelerator with your Items.

The final outcome should look like this:

OrderManagement Orders **Items**

Andrea McKenzie


### Items gallery



**Camera Lens**

Wide-angle camera lens with a focal length of 35 mm


\$400,00



**Cocooil Body Oil**

Light oil specially formulated with pure Cocoa Butter and Vitamin E to soften and soothe rough and dry skin


\$25,00



**Cosmetic & Face Wash Set**

Set with 6 products, including charcoal cleaner, esfoliating rub and moisturizing balm

\$100,00



**Earbuds**

Lightweight design makes it ideal for listening while during exercise, travel or for everyday wear

\$210,00

# How-to

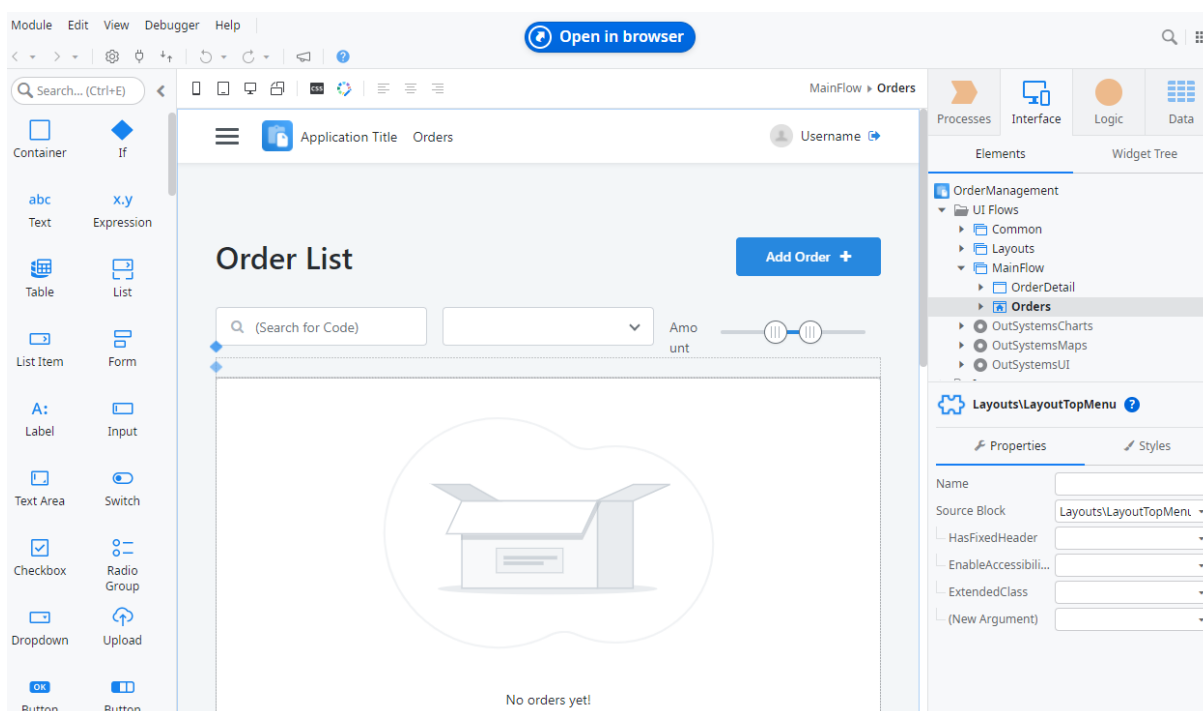
In this section, we'll show you a thorough step-by-step description of how to implement the scenario described in the previous section.

## Getting Started

In this tutorial, we assume that you have already followed the previous tutorial, and have the two Screens, the data model and the search filters implemented.

If you haven't done that so far, it is important to go back to the previous tutorials and follow them.

To start this exercise, you need the Service Studio with the module Order Management opened. You should see the Screen below with the source of our application.

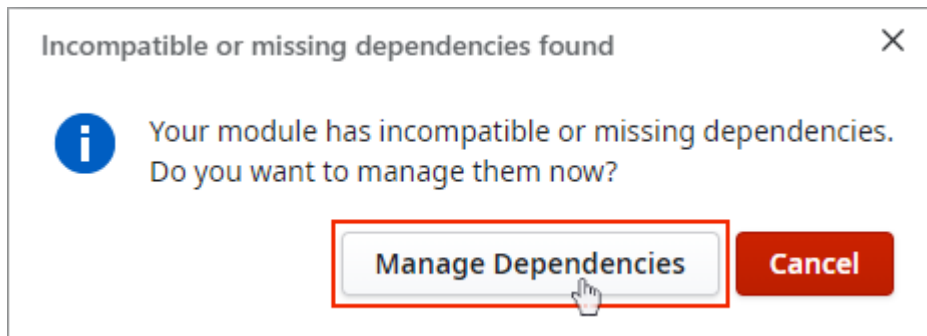


If you don't see it, make sure to open the application and module in Service Studio!

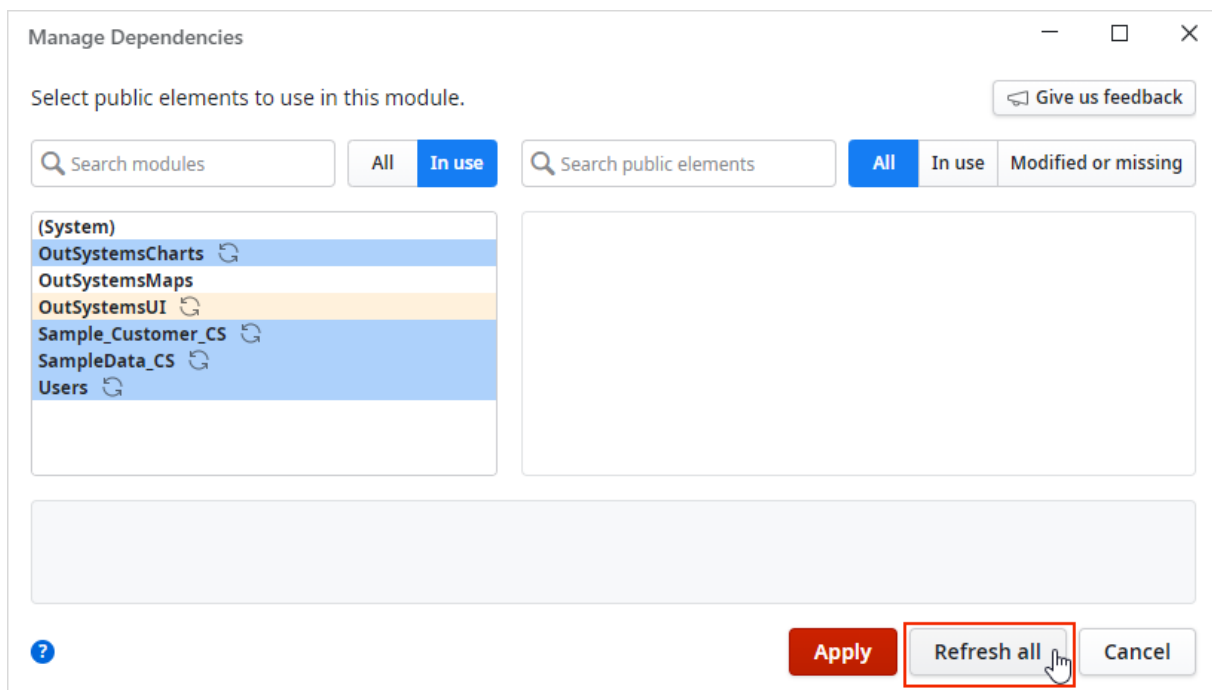
## Outdated Dependencies

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.



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



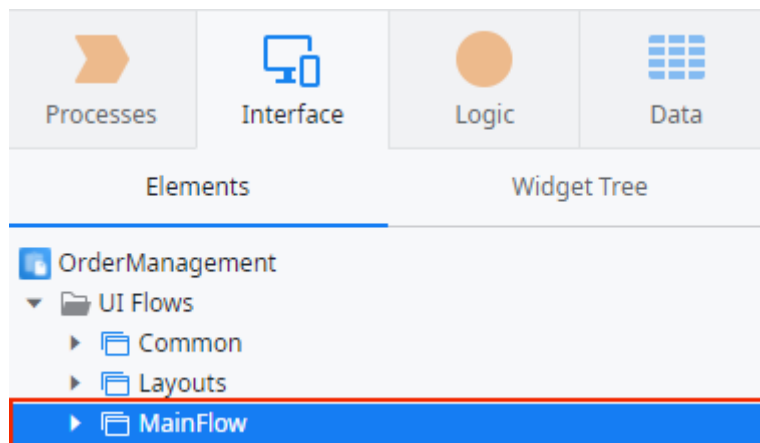
Now publish the module to update the project



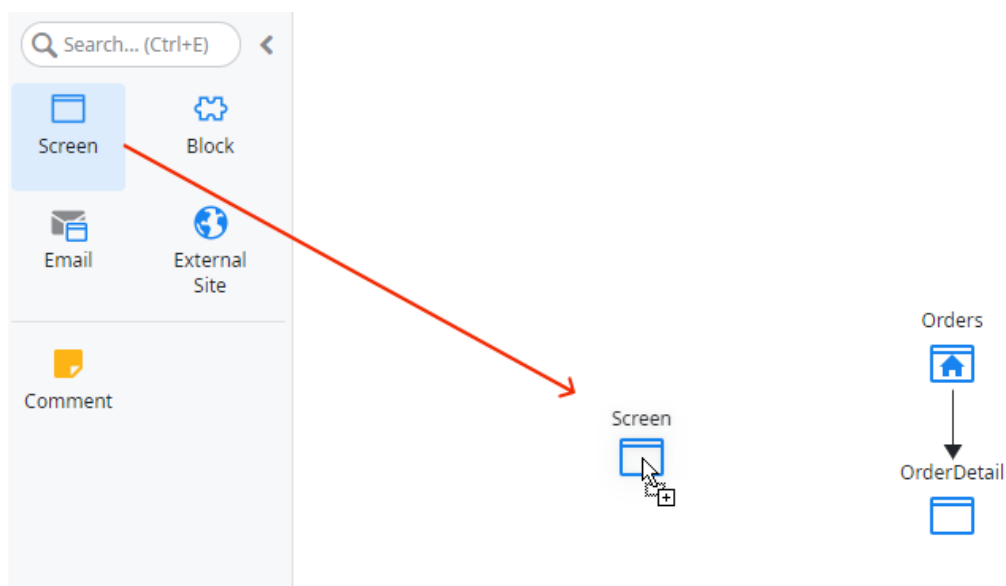
## Creating the Items Screen

In the previous tutorials you created Screens by dragging and dropping Entities to the MainFlow. In this section, you will create a new empty Screen and add it to the menu of the application.

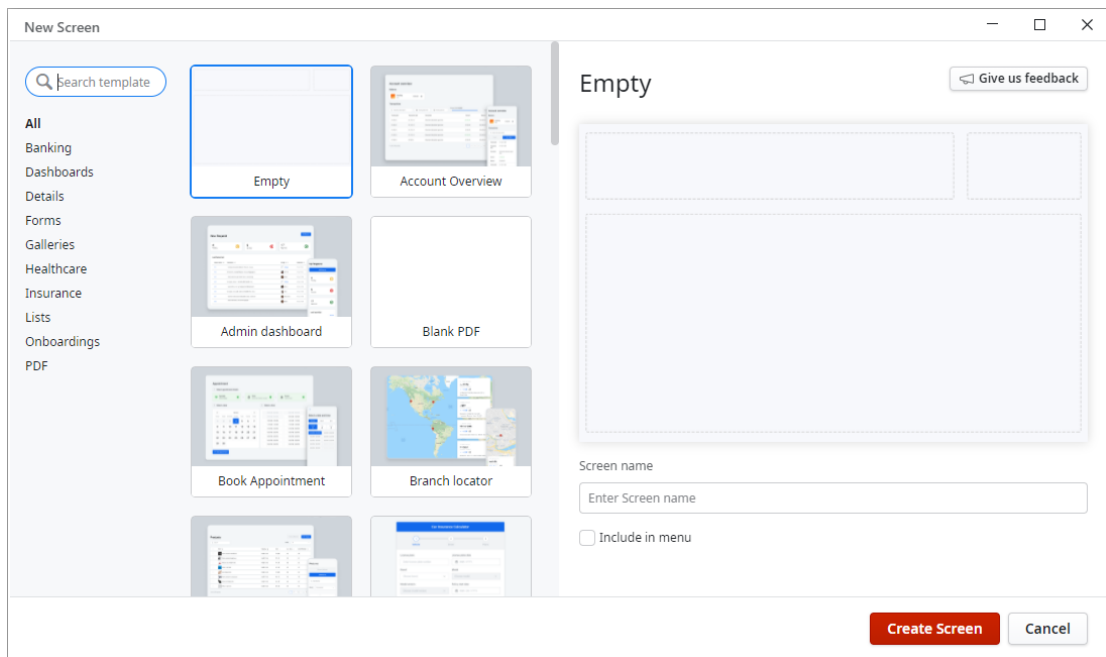
- 1) Under the Interface tab, open the **MainFlow** by double clicking it on it.



- 2) Drag a **Screen** from the left sidebar and drop it on the MainFlow.



- 3) In the New Screen dialog, select **Empty**, then click on the **Create Screen** button.

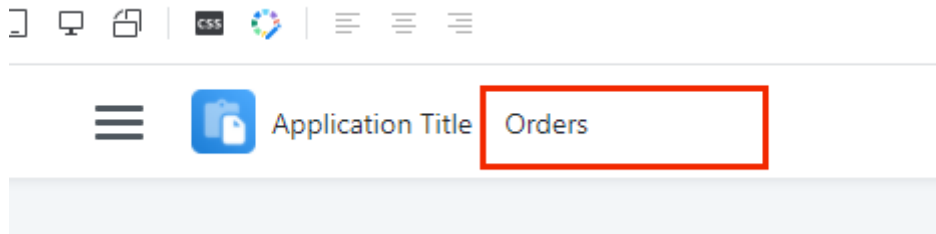


When you add a new Screen to the app you can choose to use Screen templates. You are not going to use them in this app, but don't forget to check our [documentation](#) to learn more about them.

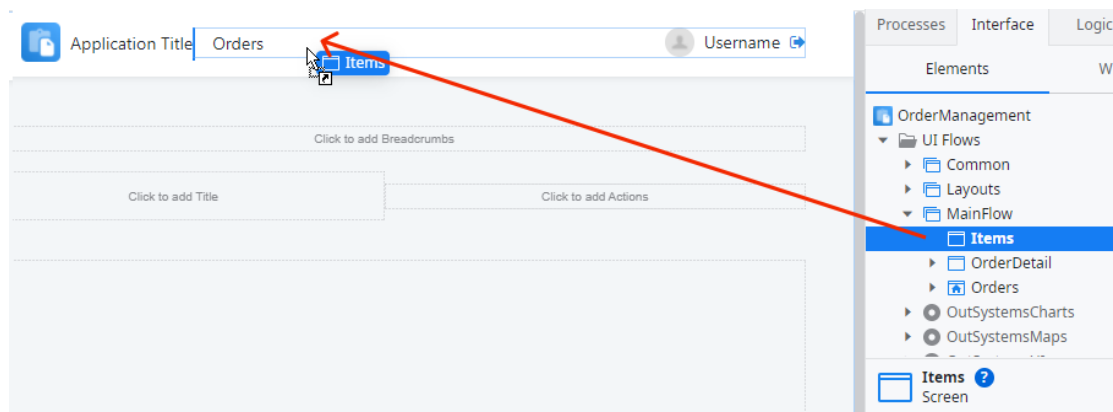
- 4) Rename the Screen to *Items*.



- 5) Double-click the Items Screen to open it. We have an empty Screen at this point. But, before we continue to actually populate the Screen with content, let's add it to the menu of the application.



- 6) Drag the Items Screen from the right sidebar and drop it right next to the Orders menu entry, on the top of the Screen.



OutSystems will do the rest for you!

- 7) Publish the module and open in the browser to test it.



- 8) Click on the Items Screen in the Menu to see the new Screen you created.

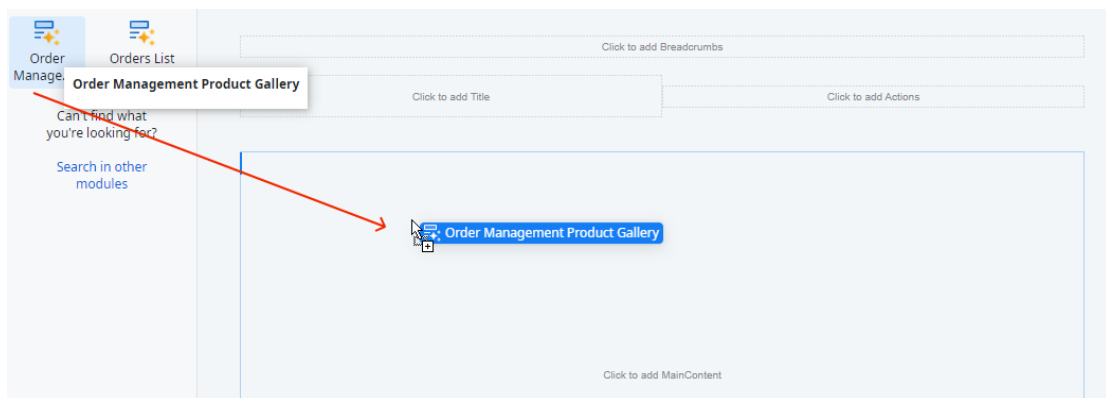


It's still empty at this point. Let's change that!

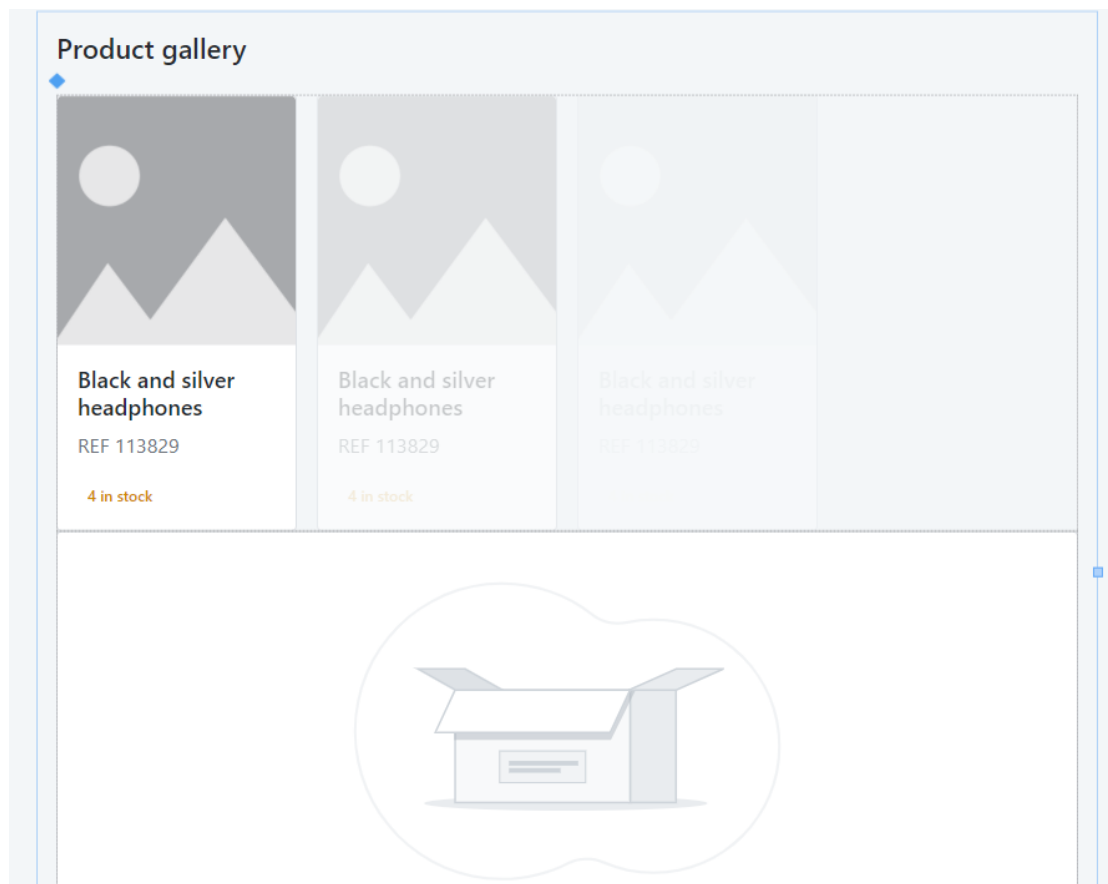
## Using a UI Accelerator

Your Items Screen is empty for now, but you can use UI Accelerators to quickly create beautiful and functional Screens for your app. So let's use one: the Order Management Products Gallery!

- 1) In the Items Screen, search for the **Order Management Products Gallery** accelerator on the Toolbox (left sidebar), then drag and drop it on the center of the Screen.



- 2) Wait a few seconds until the accelerator creates the interface with a Product Gallery. Your Screen should look like the one below:



The accelerators use sample data to fill out the elements in the Screen. But you want to use your data right?

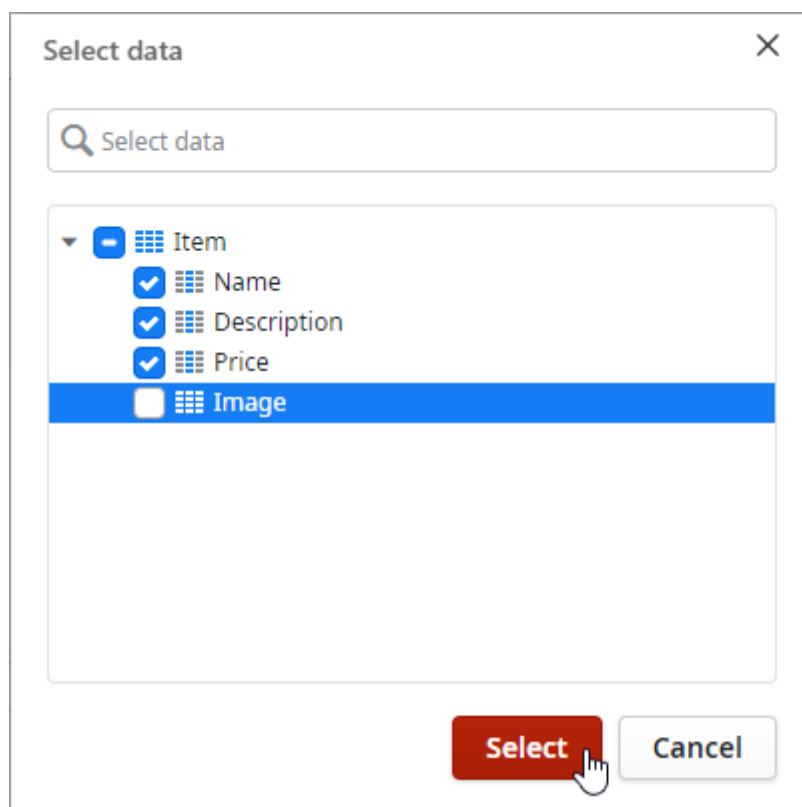
## Replacing Data

The accelerators do a lot of heavy-lifting, but you need to make sure that was created is actually using your data and not sample data. So, let's replace the sample data with data from the Item Entity.

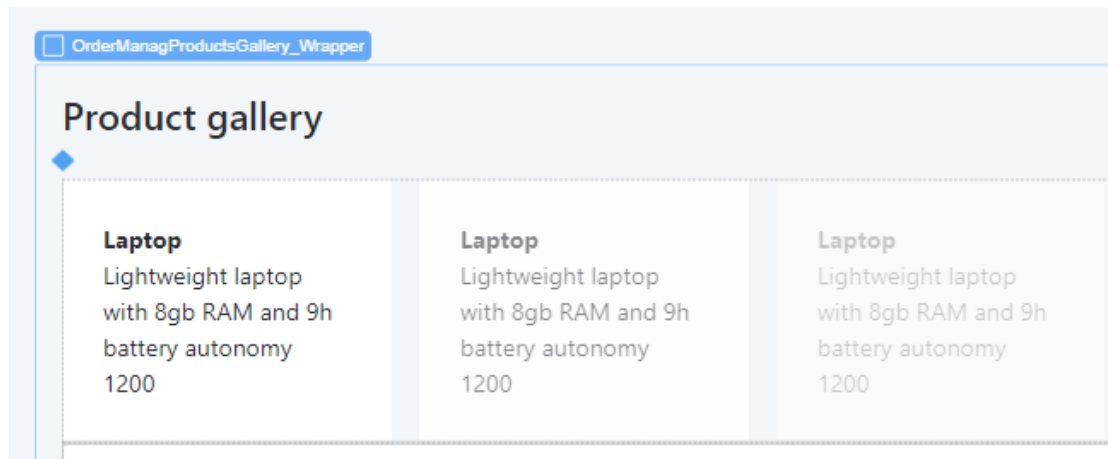
- 1) Click on the **Data** tab, then drag the **Item** Entity and drop it on the product card in the Items Screen, as shown in the picture.



- 2) A dialog with the Entity and its attributes will appear. Untick the **Image** attribute and click on **Select**.



The result will look like this:

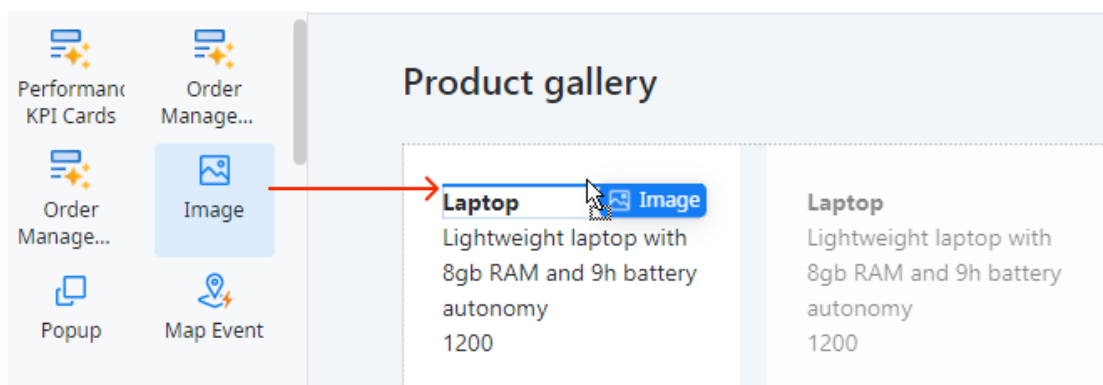


Right now, everything on the Screen is using your Items data, instead of the sample data. Amazing, right? The UI changed a bit, but that's expected since our Items Entity is not exactly the same as the sample data being used, and that's ok. OutSystems tried to adapt to the new data.

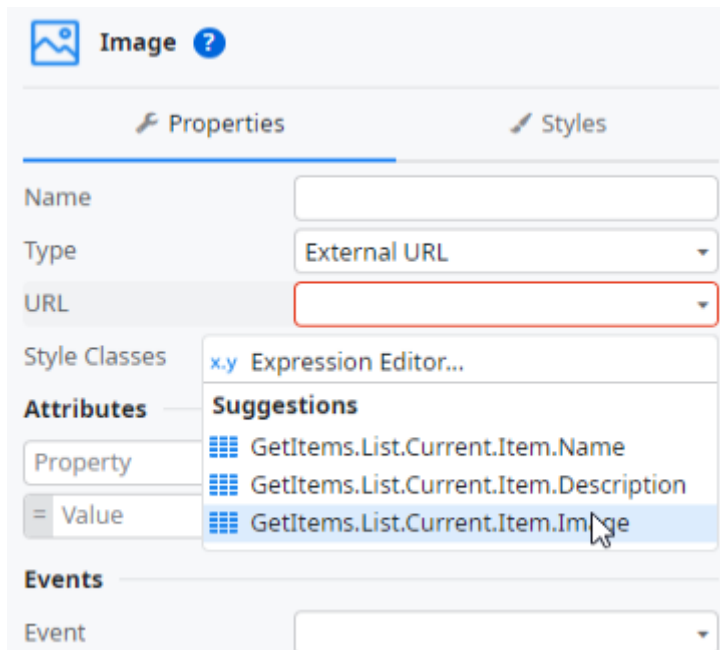
## Bring Back the Image

Ok, so why did we not include the Image when we replaced data? Because the Image in the Items Entity is a **text** with an URL that points to an image. If we selected it, then we would have an URL appearing on the Screen. And we don't want that!

- 1) Drag an **Image** from the Toolbox and drop it above the Item name.

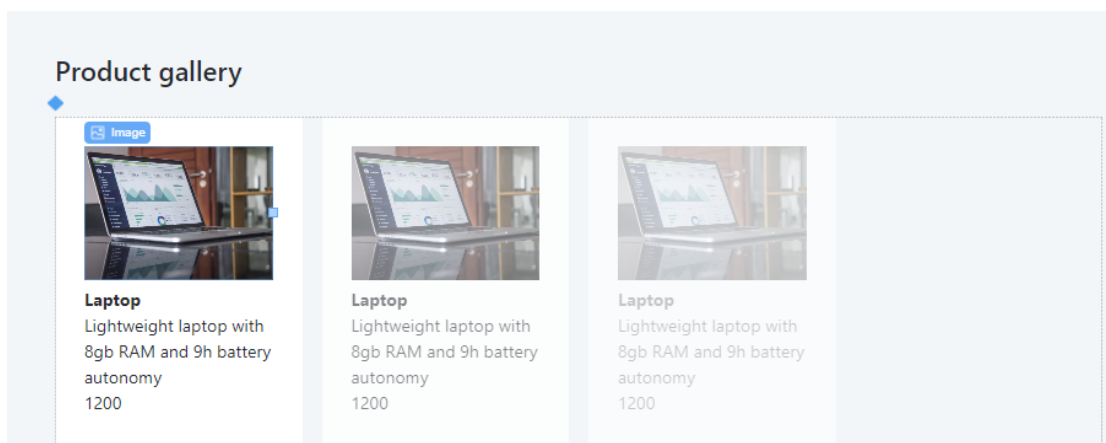


- 2) On the Image properties on the right sidebar, set the **Type** to **External URL**. In the **URL** property that appears below, select the *GetItems.List.Current.Item.Image*



So, since our Image attribute is an URL, we indicate that the image that will appear comes from an external URL. Then, we set the URL to the Image attribute that is fetched by the GetItems Aggregate.

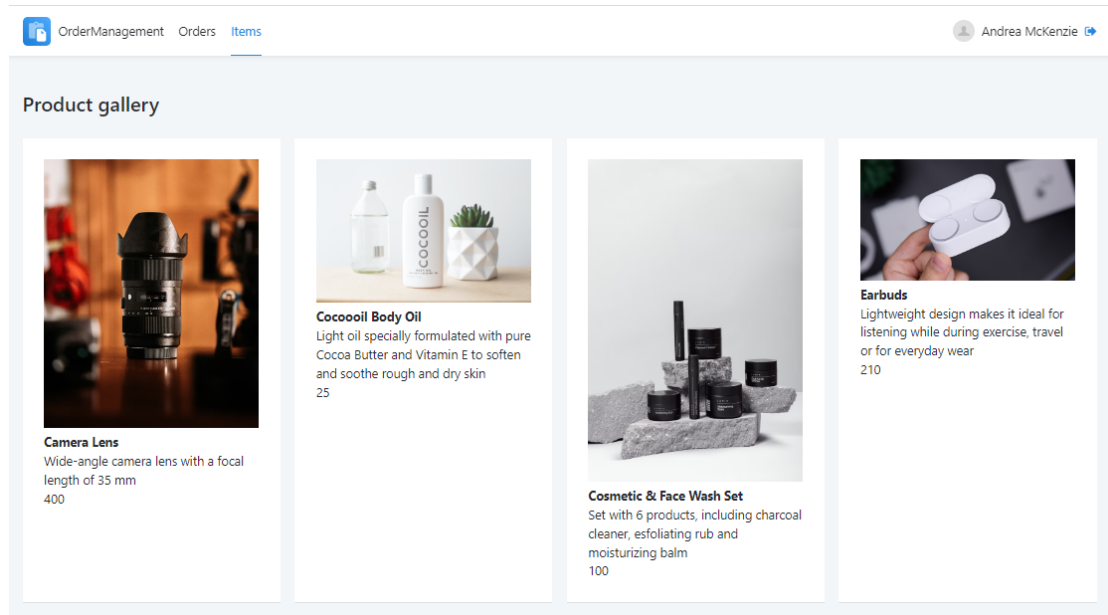
Your Screen should look like this:



- 3) Publish the module and open the app in the browser.



How does it look? Kind of strange right?

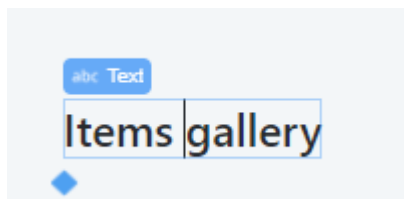


This happens because the images have different shapes and sizes. But don't worry, we can quickly improve this.

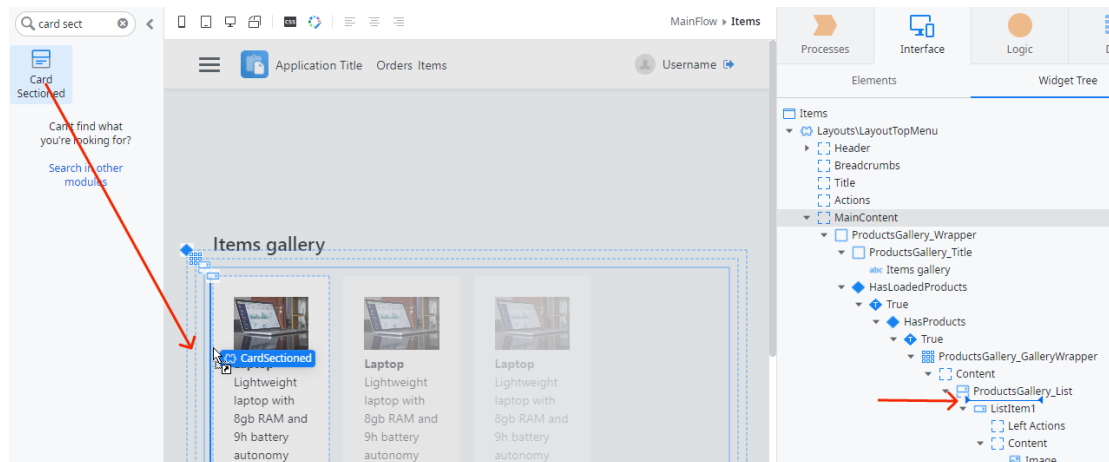
## Final touches

Let's do some final adjustments. You will change the Title of the Screen, use a new Widget for the Gallery called Card Sectioned that will create a clear separation between the data, and work on the style to make sure the images all look the same.

- 1) In the Items Screen, click on the *Products Gallery* text and type *Items Gallery* instead.

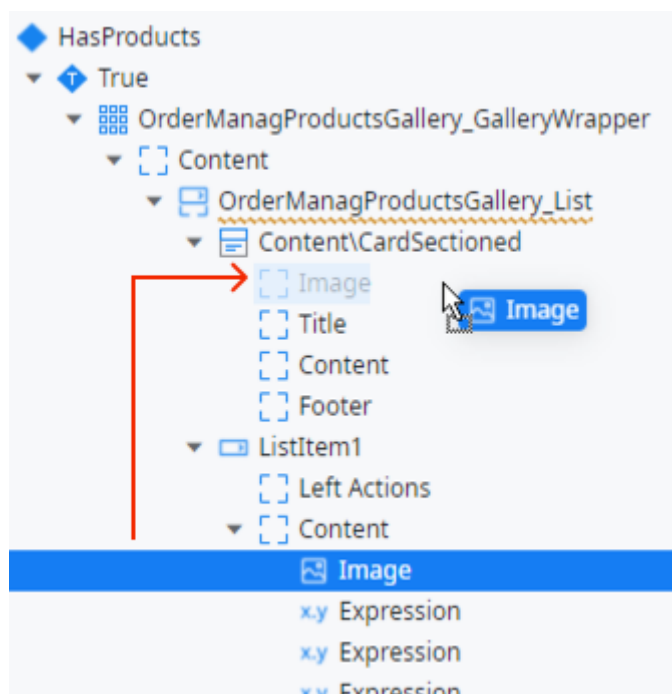


- 2) Drag a **Card Sectioned** Widget from the Toolbox to the Screen, and using the Widget Tree to help you, make sure it falls inside the **ProductsGallery\_List**.



This List is using a ListItem Widget to display the Item information, but you will replace that by the CardSectioned. The CardSectioned has four areas, or placeholders: Image, Title, Content and Footer. You will now separate the Item information in these four placeholders.

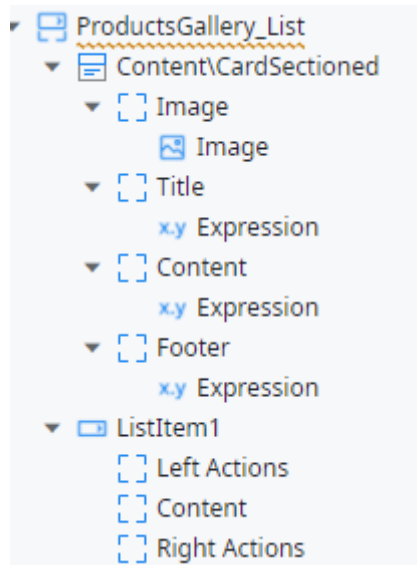
- 3) Drag the **Image** inside the ListItem and drop it on the Image placeholder of the CardSectioned.



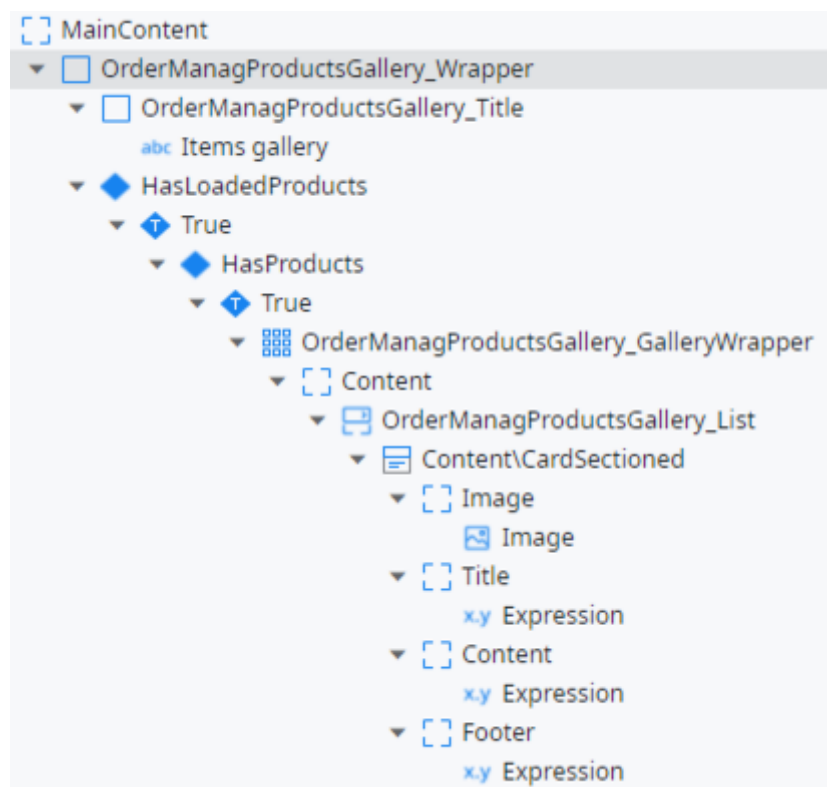
- 4) Drag each one of the remaining Expressions in the ListItem, in order, and drop them on the remaining placeholders of the CardSectioned. The first expression



will go to the **Title** placeholder, the second one to the **Content** placeholder and the third one to the **Footer** placeholder.

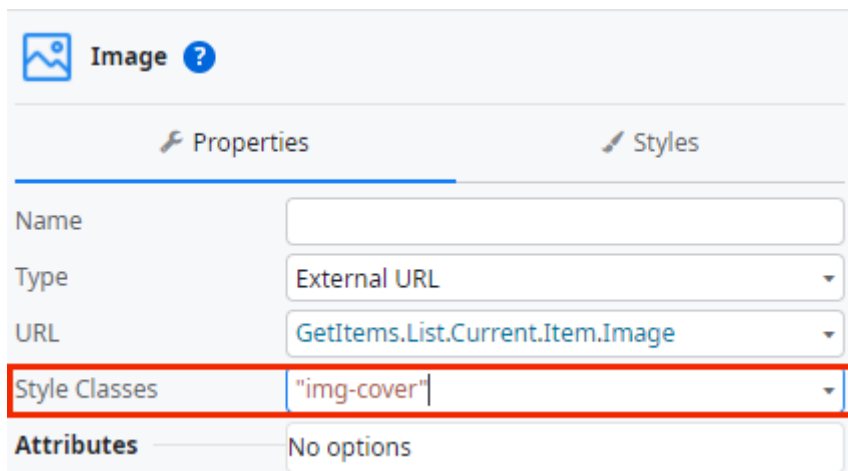


- 5) Delete the **ListItem** after you're done. Your Widget Tree should look like the image below.



If you look to your Screen, the information has more space between them.

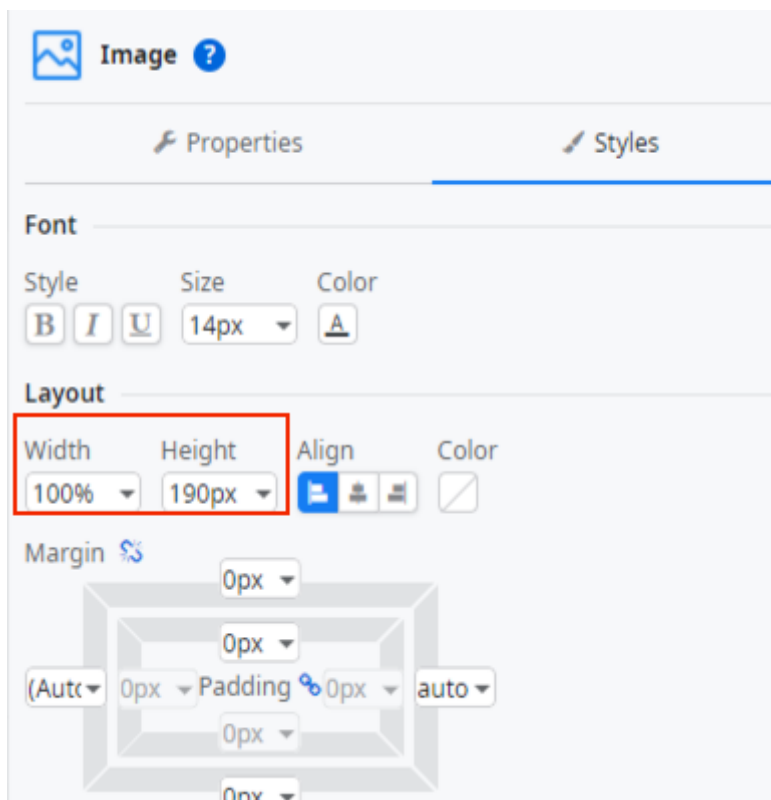
- 6) Now, let's work on the image. Click on the **Image** to open its properties and type `"img-cover"` in the **Style Classes**.



This is applying a style that already exists, don't worry about it at this point. We'll cover that later!

- 7) In the properties of the Image, switch to the **Styles** tab and type `100%` in the **Width** property and `190px` in the **Height**.

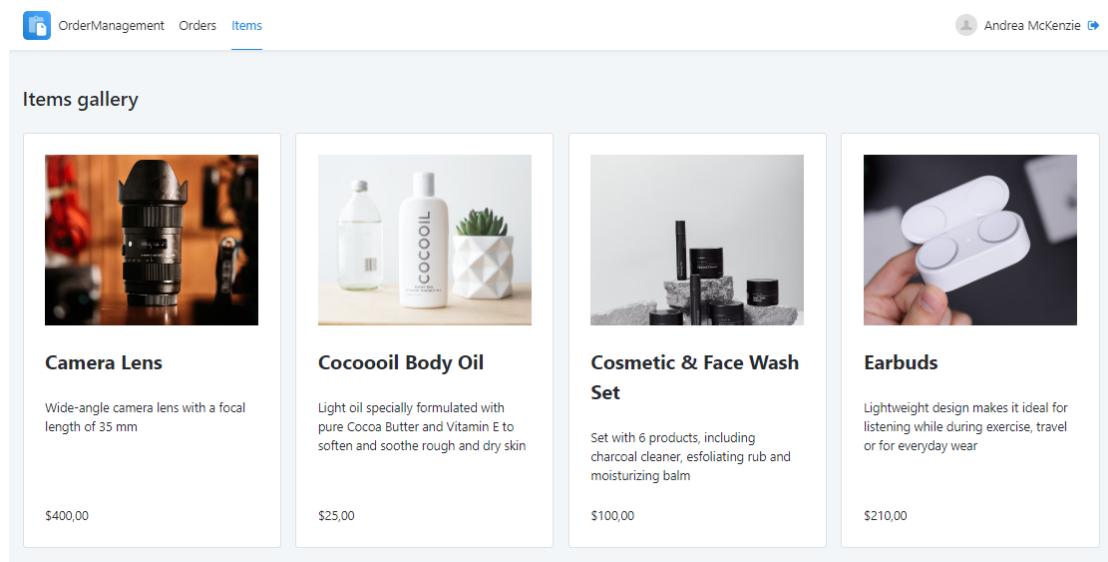
**Note:** If we keep the height at 100%, then the full height of the image would be used, causing the strange behavior you saw earlier. To fix that we're limiting it to 190 pixels.



8) Publish and open the app in the browser, so you can see the final result.



Better? Beauty is in the details!



## Wrapping up

Congratulations on finishing this tutorial. With this exercise, you had the chance to use more accelerators and do some fun stuff with the UI of the Items Screen, while learning 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 UI](#)
- 2) [Screen Templates](#)
- 3) [UI Accelerators](#)
- 4) [Replace sample data with real data](#)

**See you in the next tutorial!**