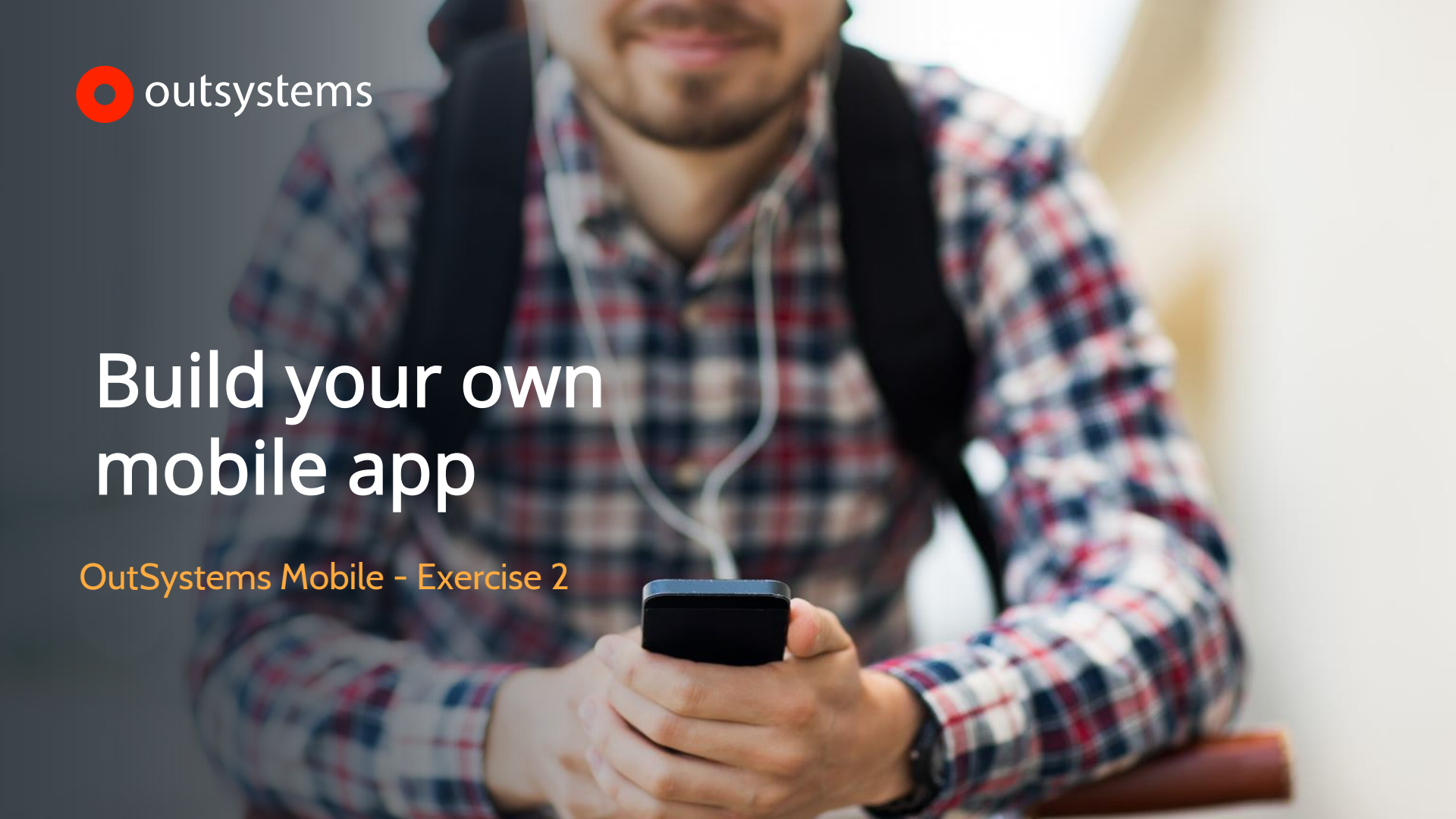


outsystems

Build your own mobile app

OutSystems Mobile - Exercise 2

outsystems

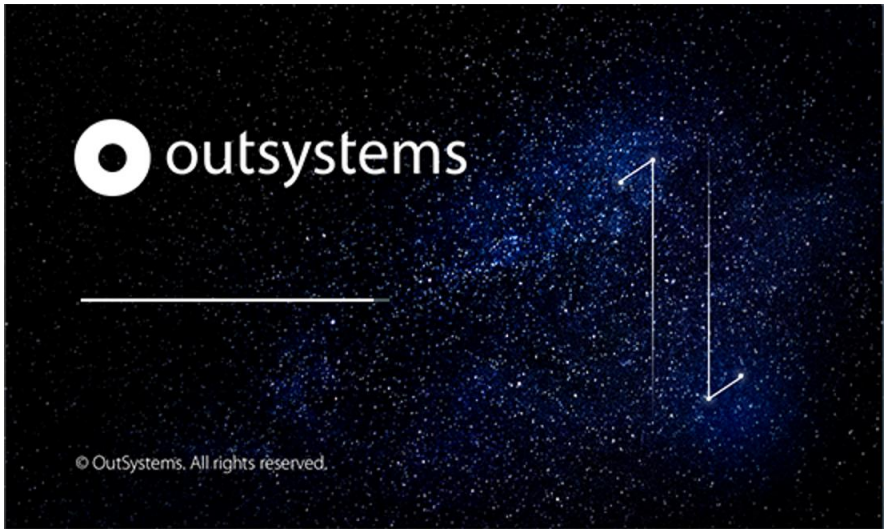
Get your environment ready!

Start your development studio

Development Studio

Open and start

1. Open **Development Studio (Service Studio)**
2. Be awesome and **build your own mobile app**.



outsystems

Section 1

Mobile Application Foundation

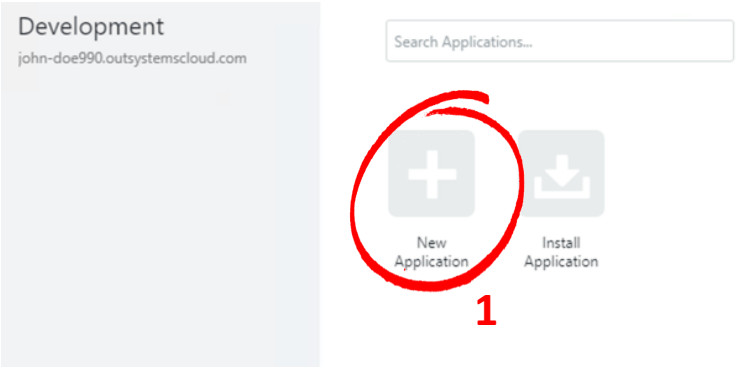
In this exercise, you will be creating a Mobile Application that lists the products you created from Exercise 1.

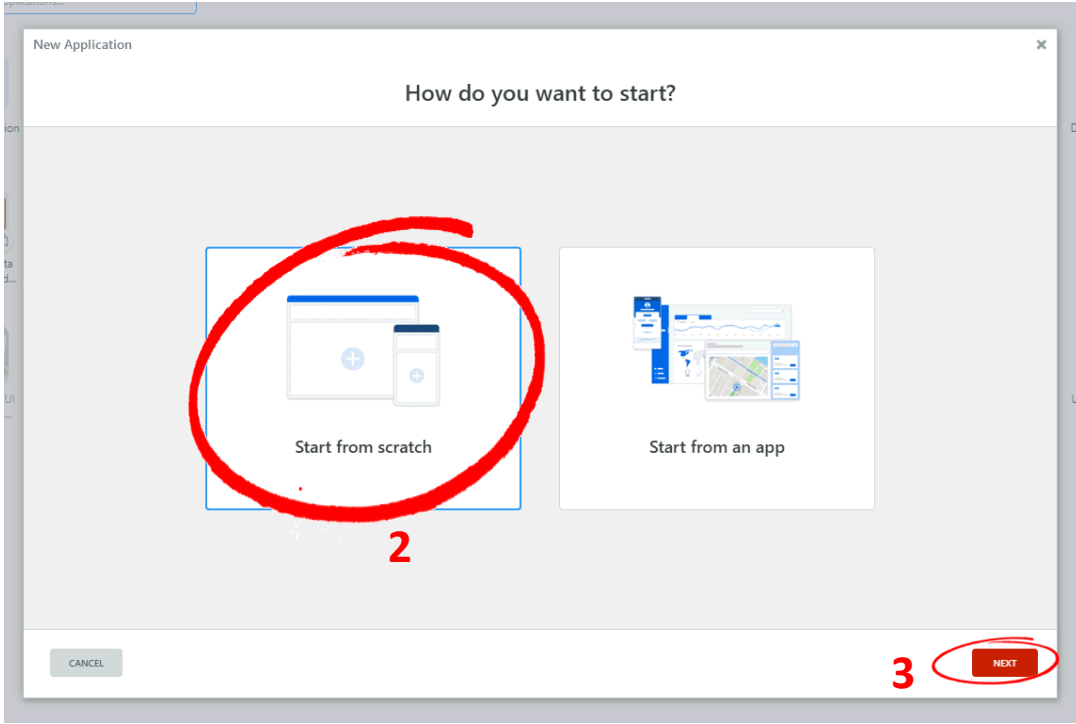
You will learn the basic differences between web and mobile apps, using a QR Code Scanner plugin and mobile templates.

The following Forge Component is required for this Exercise:

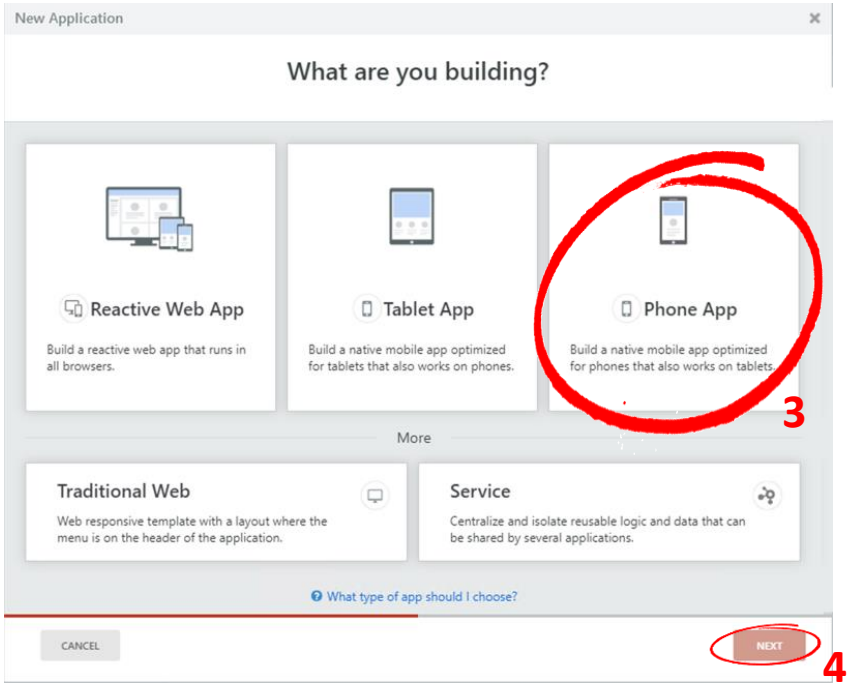
- QR Code Scanner Plugin

Create a new mobile app





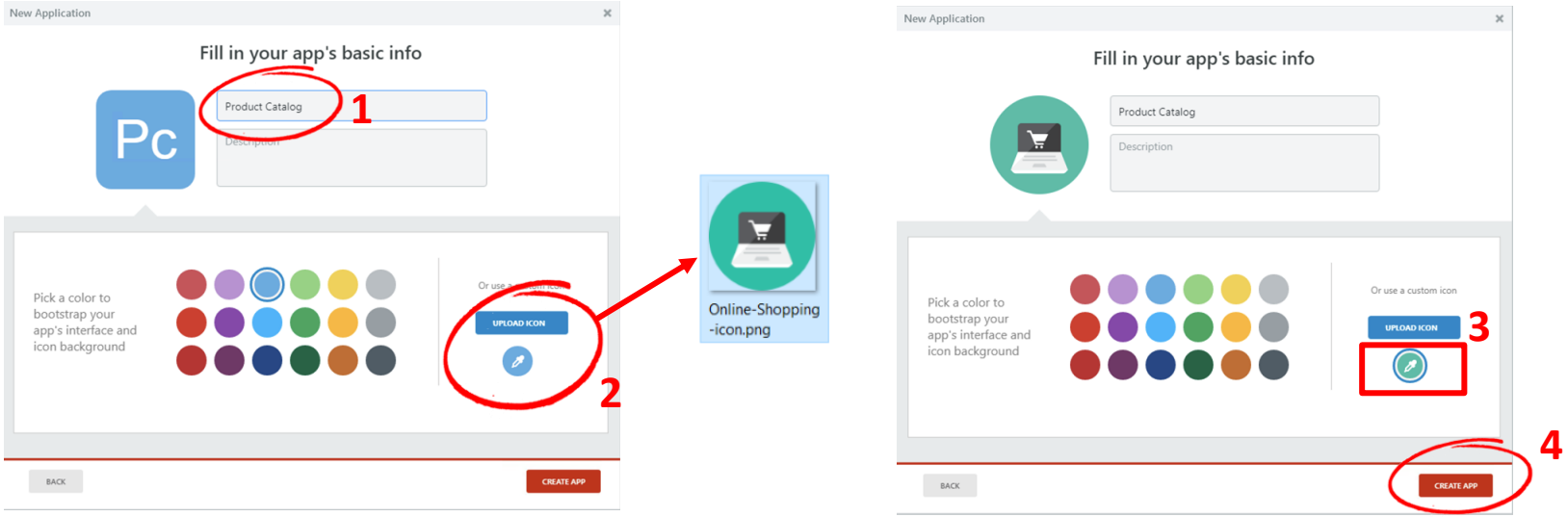
Create a new mobile app



Create a mobile app when you want to create an optimized native experience for mobile devices, with touch friendly behaviors, offline capability and/or leverage the device features such as fingerprint authentication, geo-location, camera, etc.

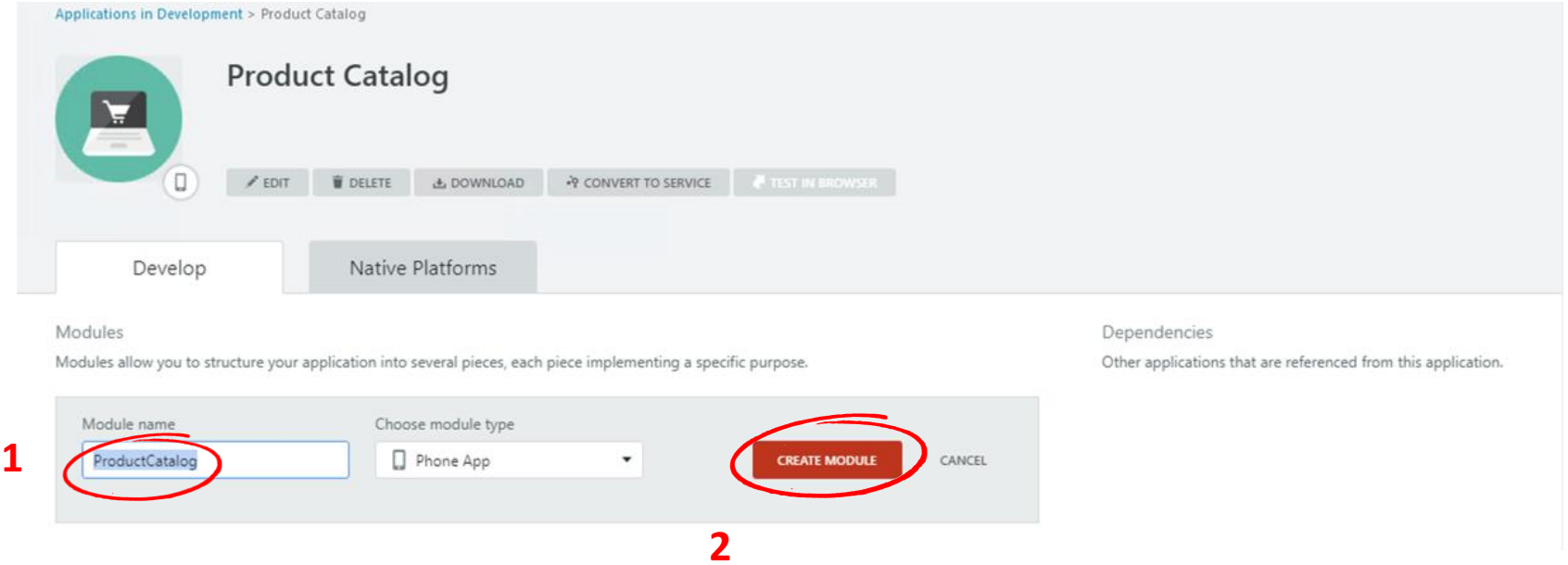
New Application

1. Type **Name** of your application “**Product Catalog**”
2. Upload the **icon** provided from Jumpstart Resources Material folder
3. Color palette will be automatically determine based on icon color. Select a **color palette**, if you want to change.
4. Click “Create App”



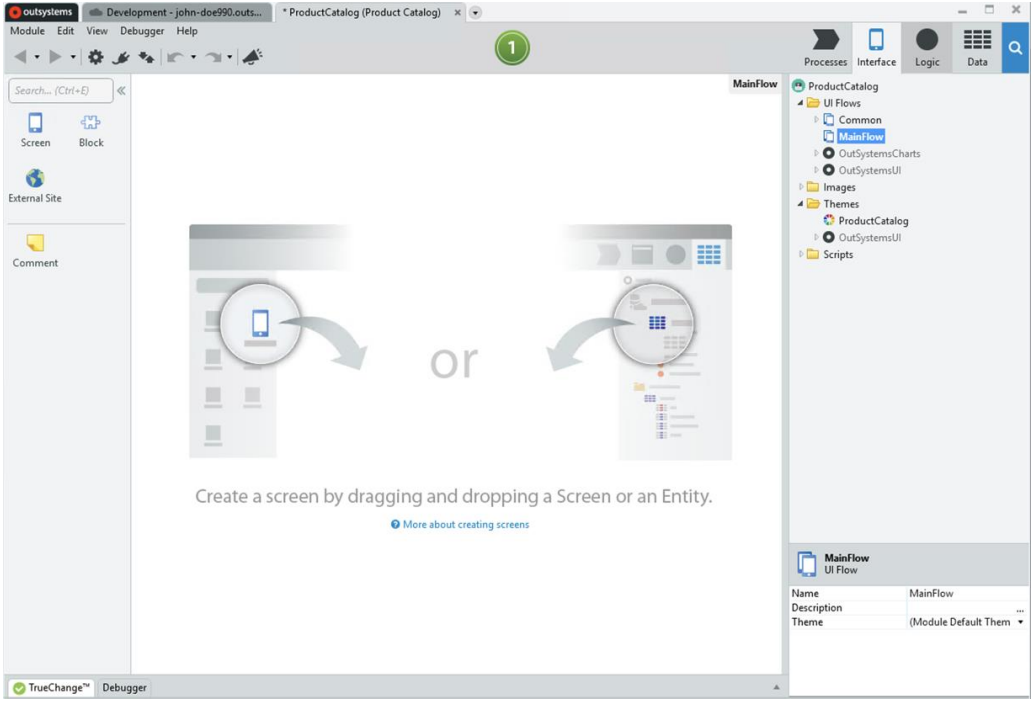
Create Phone App Module

1. Leave the default Module Name
2. Click Create Module button



Create Phone App Module

Your mobile app canvas is ready. You will be navigated to your new Mobile module

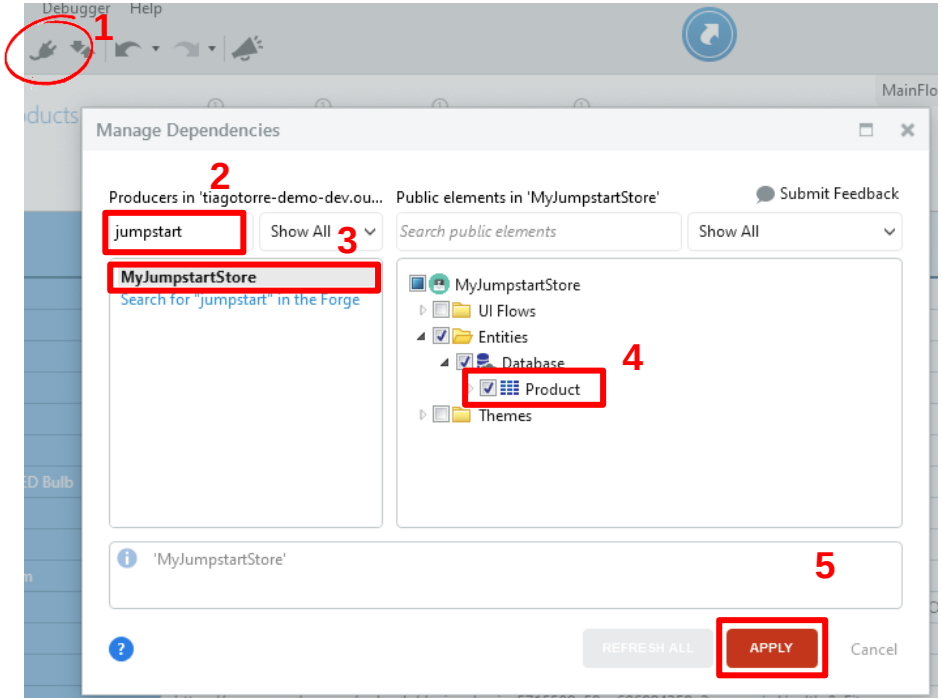


In the next steps we will add a dependency to our new mobile application. Dependencies are similar to “References” in .Net & Java.

This enables you to reuse Forge components, or pre-created templates, plugins, objects, methods, etc from existing applications.

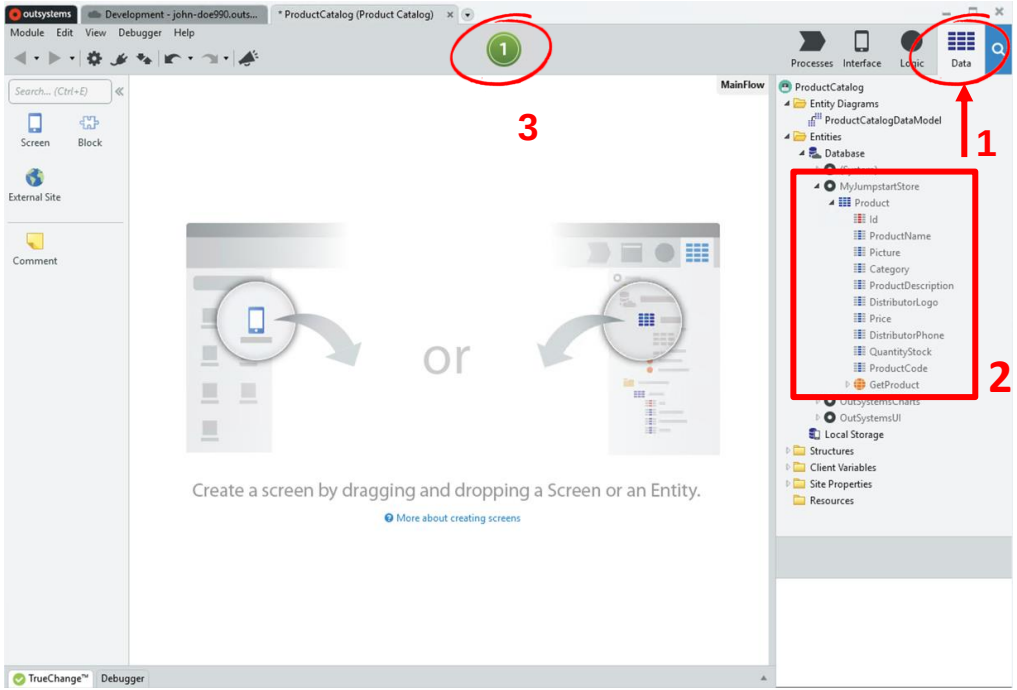
Product Entity

1. Click on the **Manage Dependencies** icon ( )
2. Search for the **<Name of your web application project in Exercise 1>**.
3. Add a reference to the **Product** entity table that we created in the first exercise.

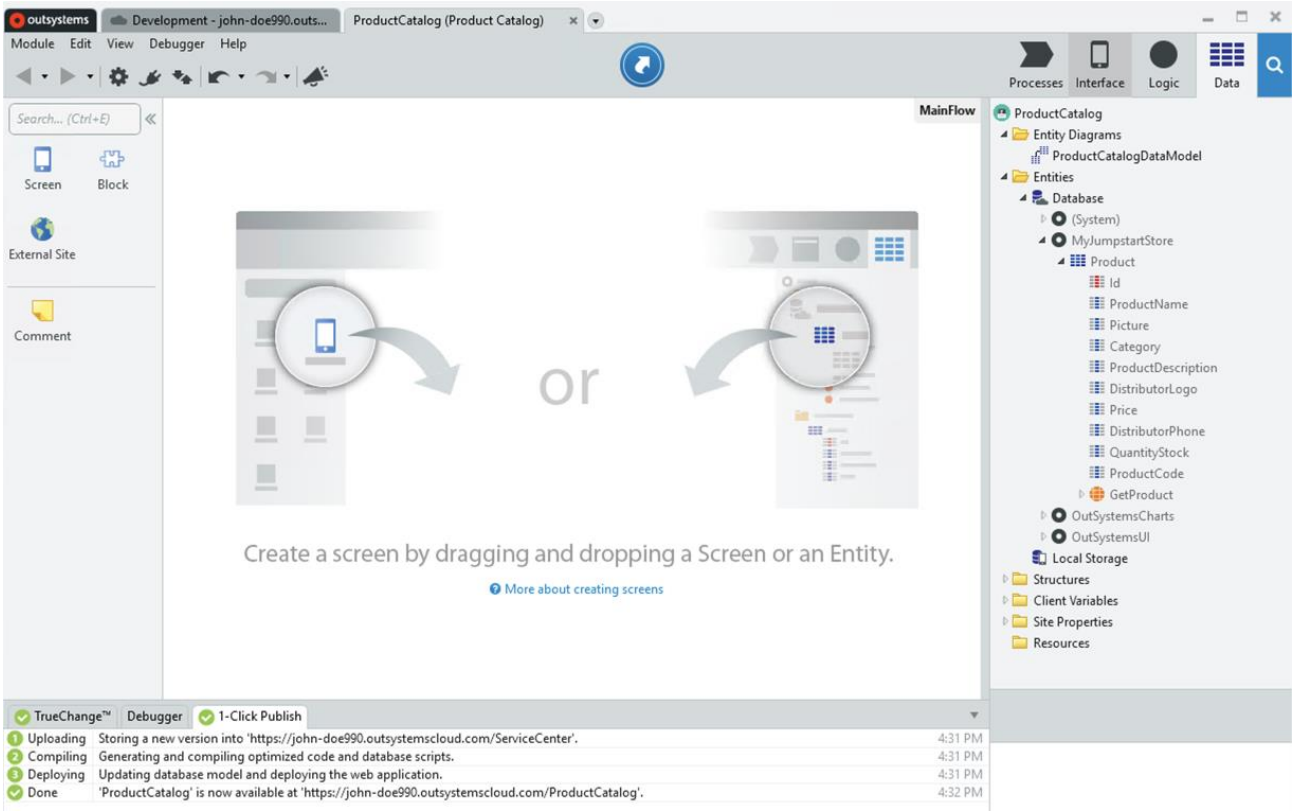


You can only see the Product entity here and not other entities because it was the only one we marked as “Public” in the previous exercise.

1. Go to the Data Tab
2. After adding the references, you can now see the **Product** entity in your project.
3. Publish your application



End of Section 1 - Mobile Application Foundation



Section 2

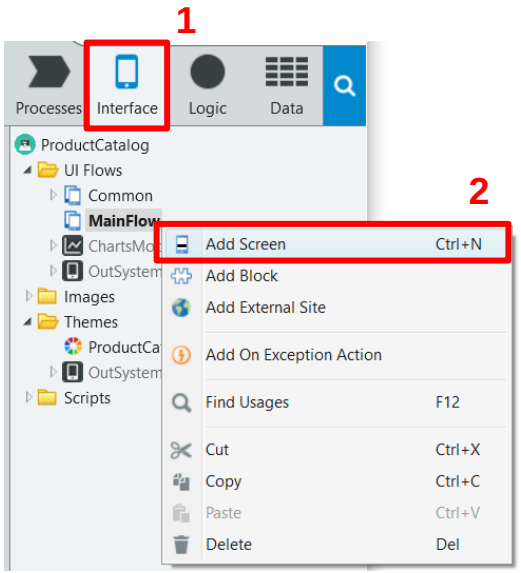
Product Gallery

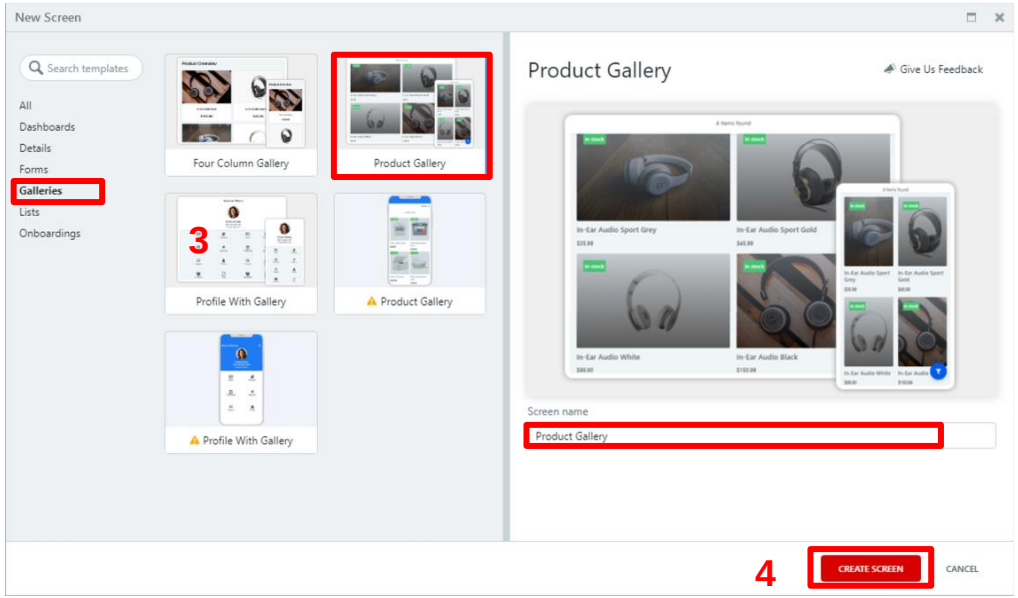
A. Mobile Templates - Product Gallery

The previous section has taught you how to add dependencies (references) into your application so that you can use downloaded plugins and reuse elements created from other applications.

In this exercise, we will create a mobile **Product Gallery screen** that contains a **filter** based on the **Outsystems UI templates**.

Create a new Screen





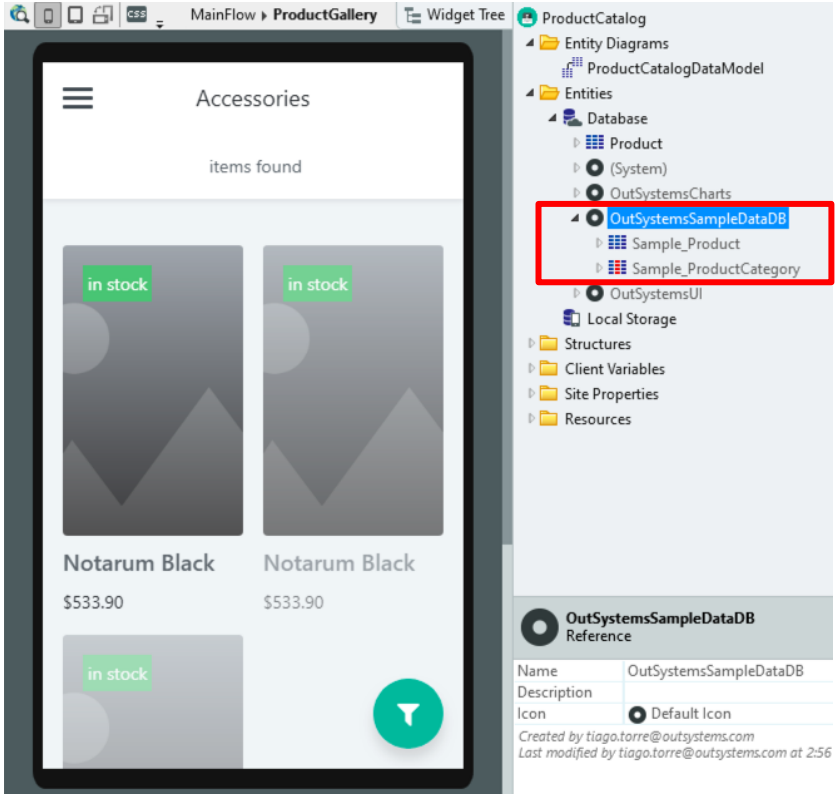
1. Go to the interface tab
2. Right-click on **MainFlow** and **Add Screen**
3. Select the **Product Gallery** Template
4. Keep the screen name as “**Product Gallery”**, and click **Create Screen**

Sample data was automatically added

A screen with pre-built template was automatically created.

Notice that it's using Sample Data Entities "**Sample_Product**" and "**Sample_ProductCategory**".

You can find these under **Data** section > Entities > OutSystemsSampleDataDB



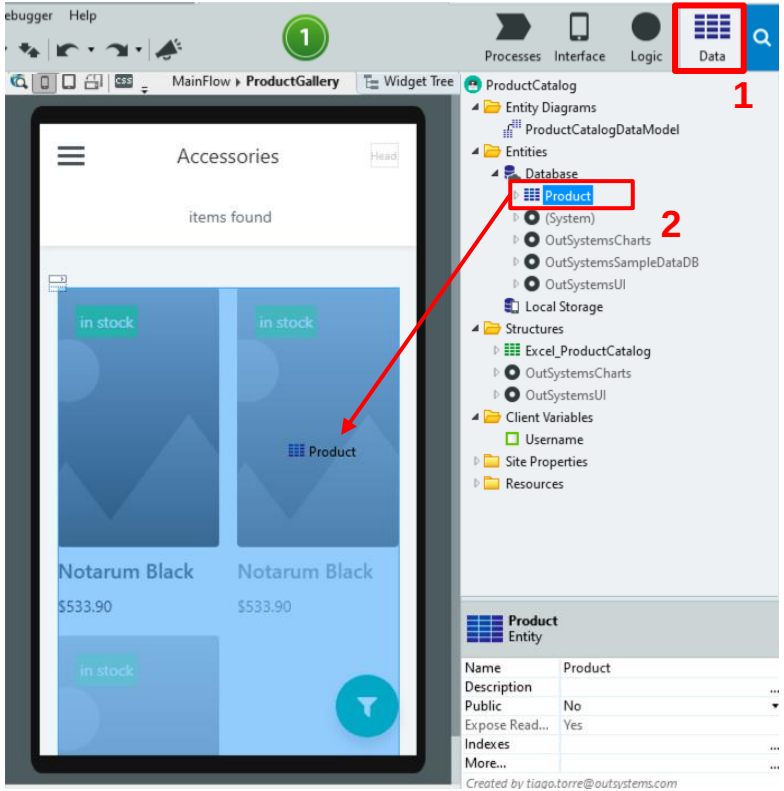
The Outsystems platform provides screen templates to help developers create user interfaces easily. These Templates contain **UI design**, include **pre-created actions** to accelerate development of the screen actions (such as filters) and **pre-created sample data** so you don't have to start from scratch.

In the next exercise, we will replace the Sample Data with our own data and adapt the pre-built backend rules (such as filters) that reference the Sample Data.

Use product entity data

1. Navigate to the **Data** tab

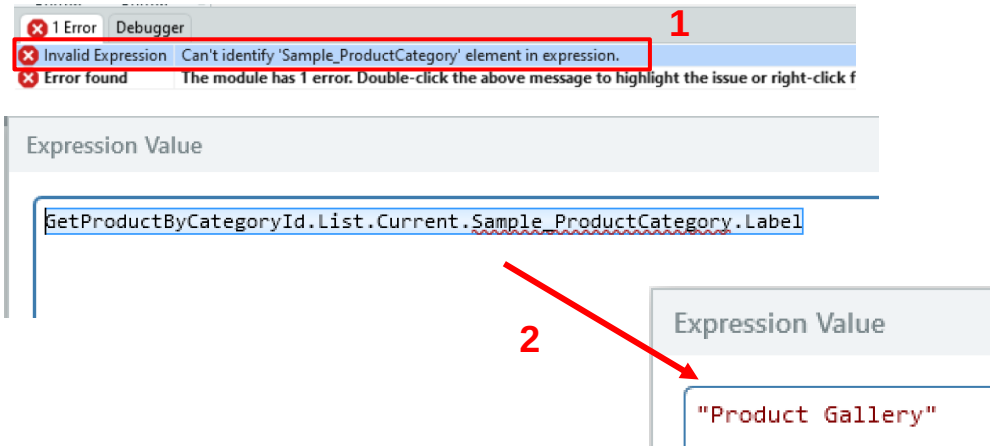
1. Drag and drop the **Product** entity into the **Gallery area**. Make sure you see "**Replace data with Product**" before letting go of the mouse click.

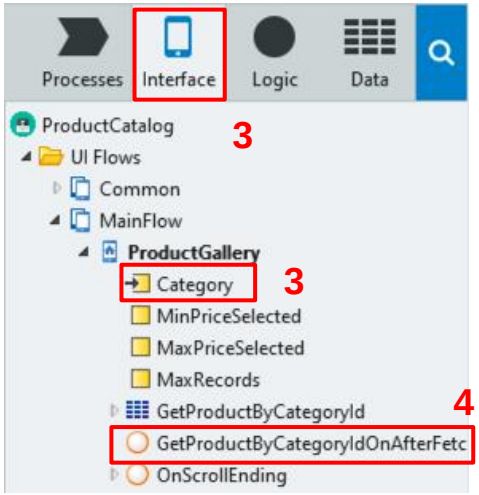


After replacing the list with our data, some variables may still need to be replaced manually.

Go to the next page to fix these errors

Can't identify 'Sample_ProductCategory' element in expression.





1. Double click on the error "*Unknown object 'Sample_ProductCategory' in expression.*"

1. This expression is the title of the screen. Delete it and change to just "**Product Gallery"** with the double quotes and click **Done**.

3. Navigate to the **Interface** tab and Delete the **Category** input parameter

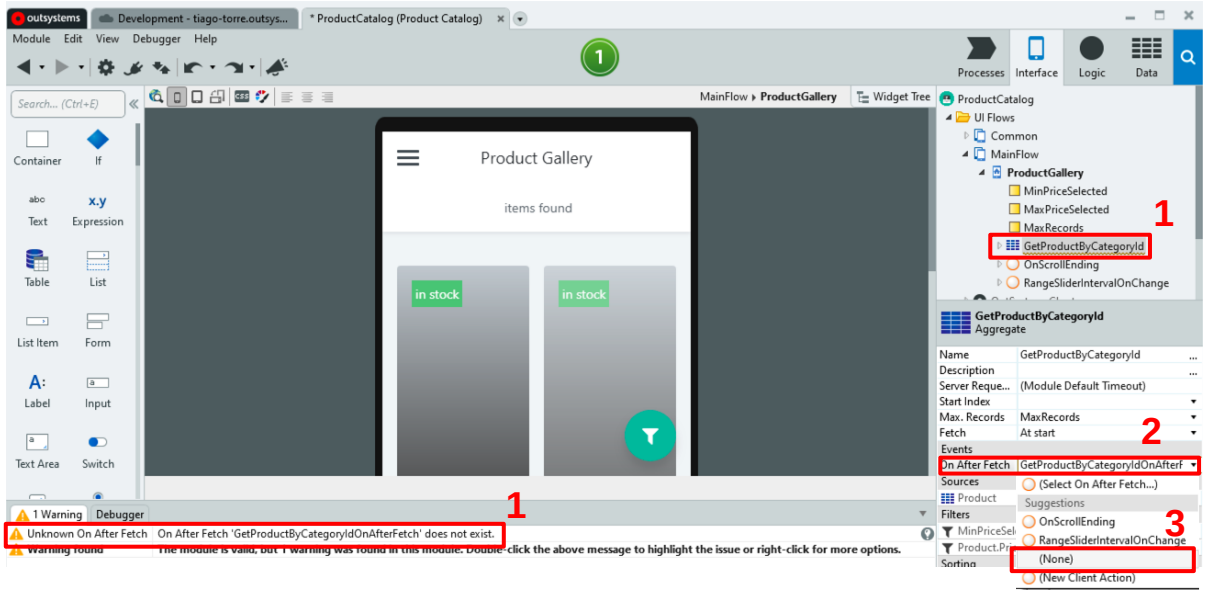
4. Delete the **GetProductByCategoryIdOnAfterFetch** Client Action. We don't need these.

Delete unused data & logic

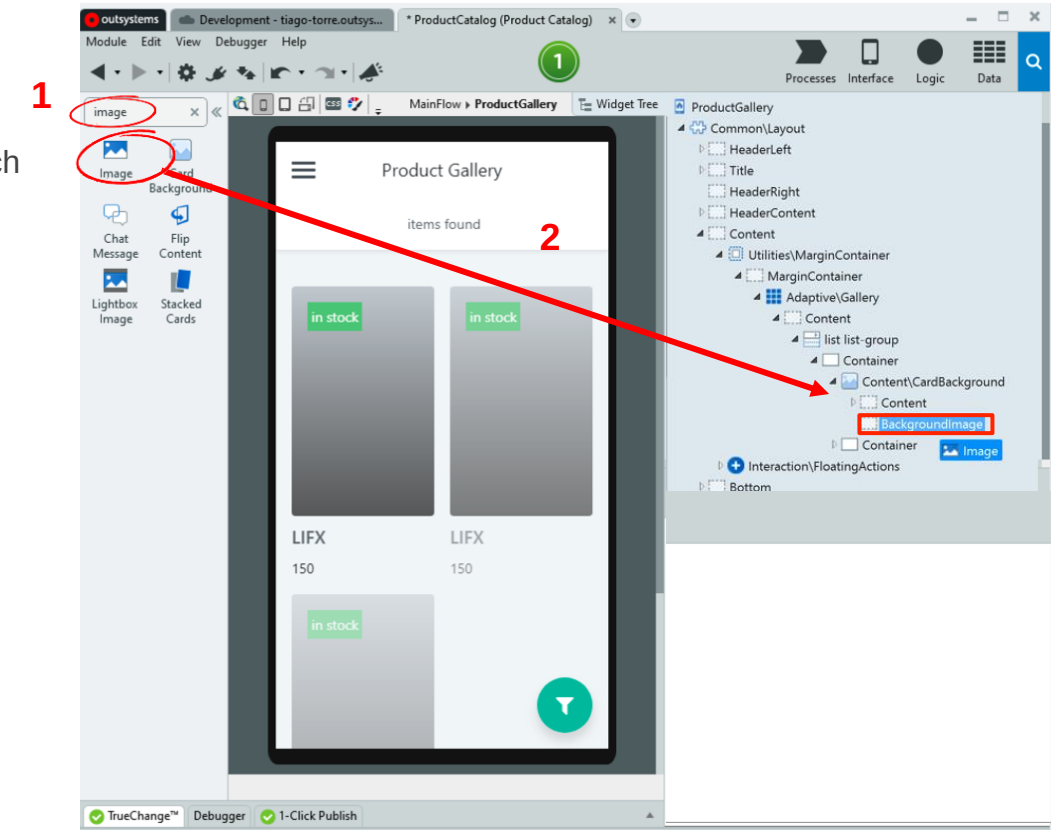
1. Double click on the “**Unknown On After Fetch”** warning or click on the **GetProductByCategoryId** aggregate

1. Expand the **On After Fetch** Event

1. Update the value of the **On After Fetch** event to **(None)**



1. Go to the toolbox search bar (top left) and search for “Image”.
2. Then, drag the Image widget and drop it on the “BackgroundImage” placeholder as illustrated.



QUICK TIP

Use the **Widget Tree** to easily locate elements in the screen structure.

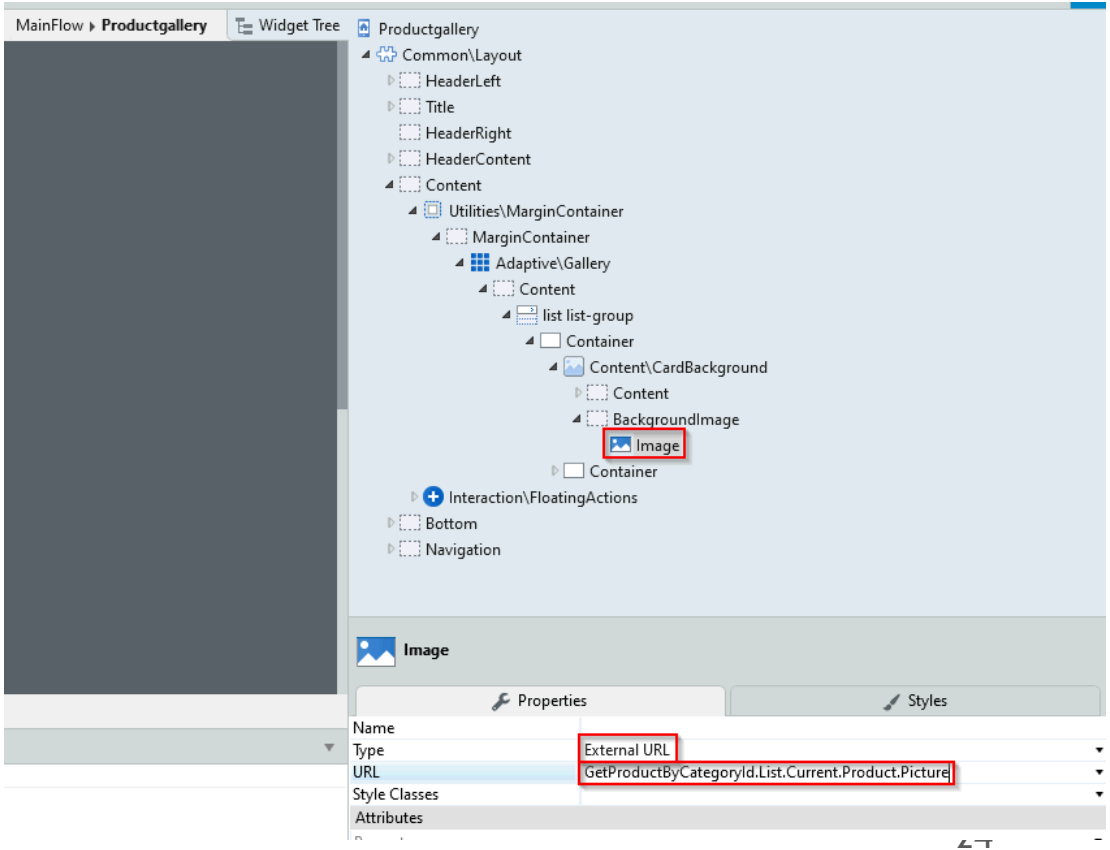
It can be manually opened by clicking the upper right icon (Widget Tree

It also **opens automatically** when you drag a widget onto the screen!

1. Select the newly created image and open the properties area.

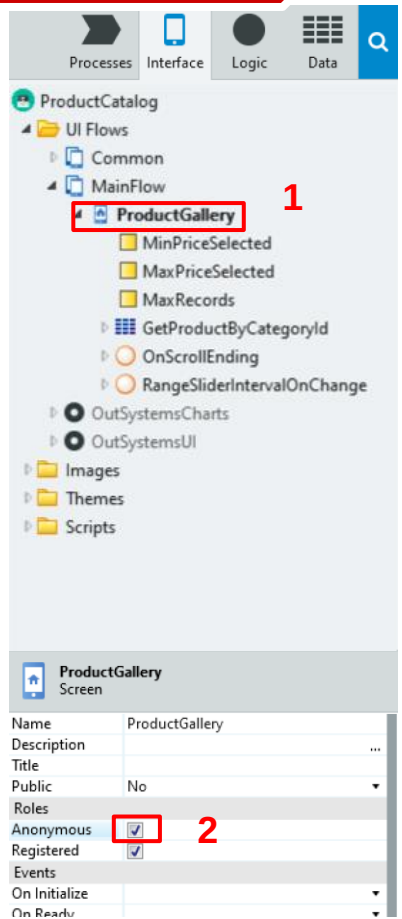
2. Change Type to "External URL".

3. Select "GetProductByCategoryId.List.Current.Product.Picture" as URL.



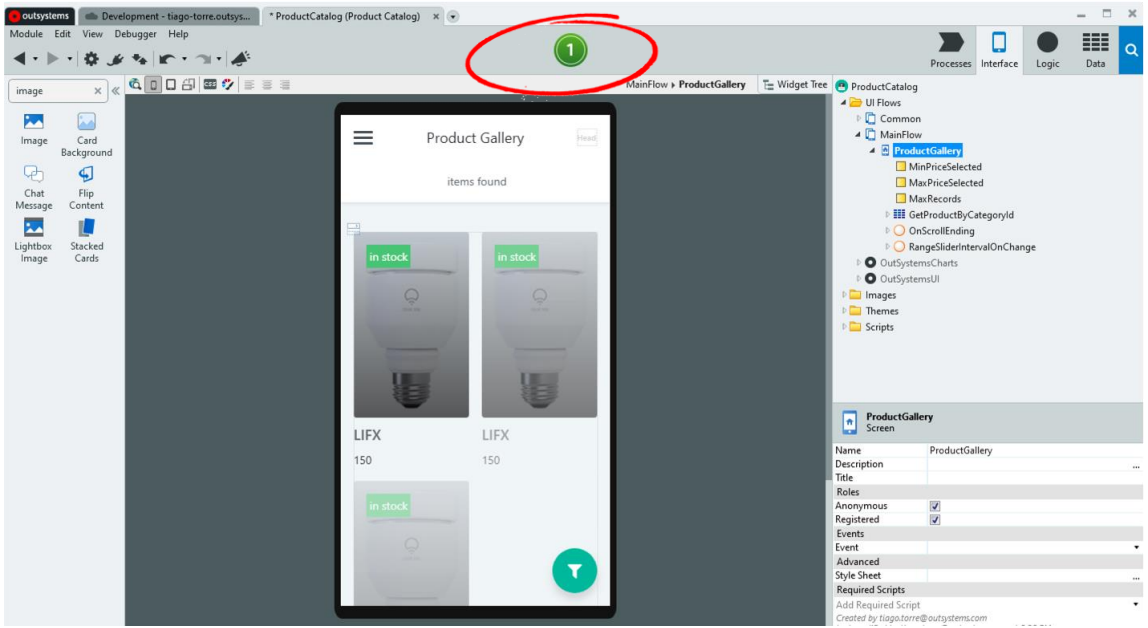
1. Click on the “ProductGallery” screen.

1. Mark the **ProductGallery** screen to be accessible by **Anonymous** users.



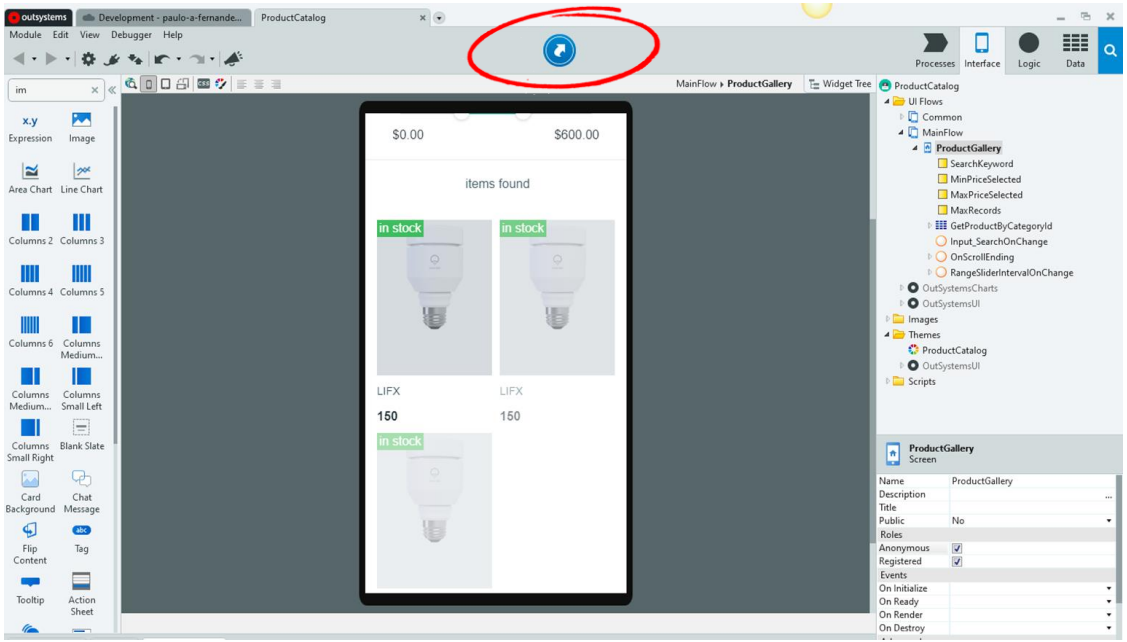
Deploy and Test it

1. At this point you can deploy your application by clicking the 1-click-publish button ( ). This will generate the mobile app.



Deploy and Test it

1. Open it in the **simulator** (and test your app.



outsystems

Section 3

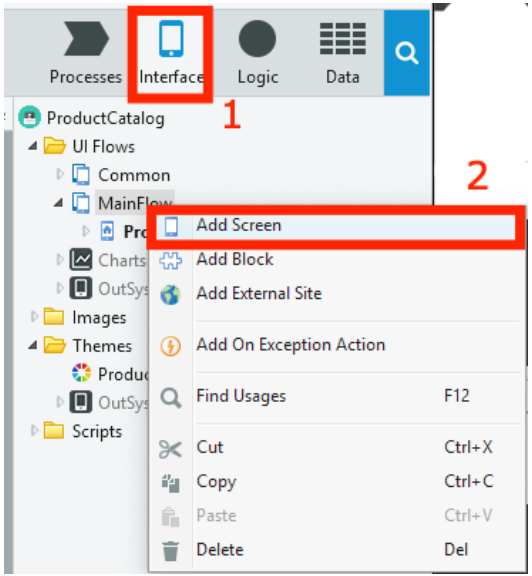
Product Detail

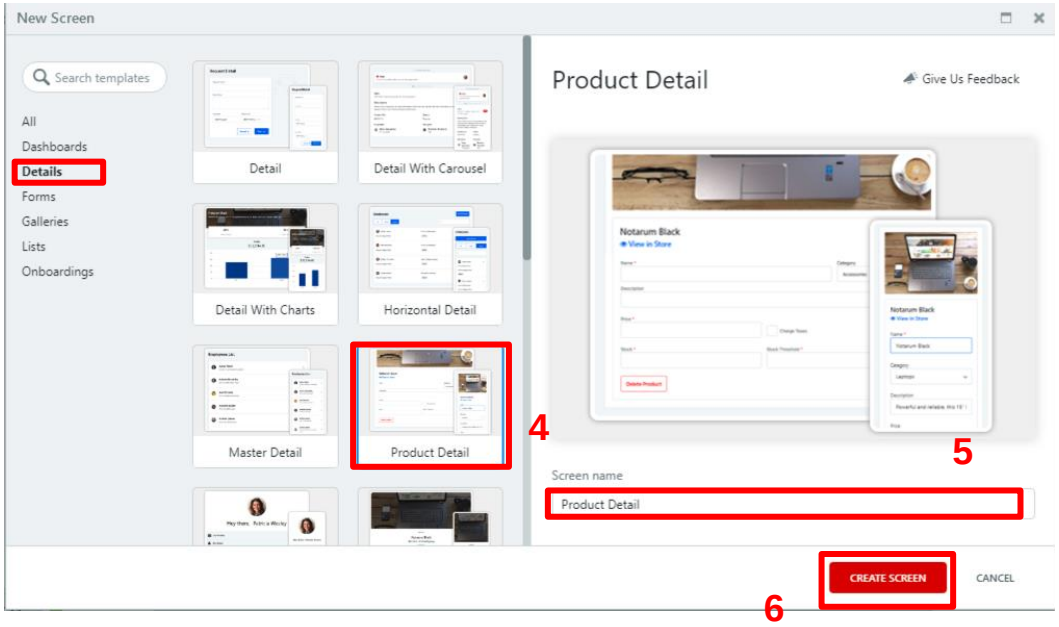
A. Product Detail

In the previous section we have create a Product Gallery screen and replaced the sample data with the Product Entity data referenced from the web application build in Exercise 1.

In this section we will create a Product Detail screen using the Outsystems UI templates, replace the sample data with our own data and link the Product Gallery to the Product Detail screen.

Product Detail





1. Go to the interface tab

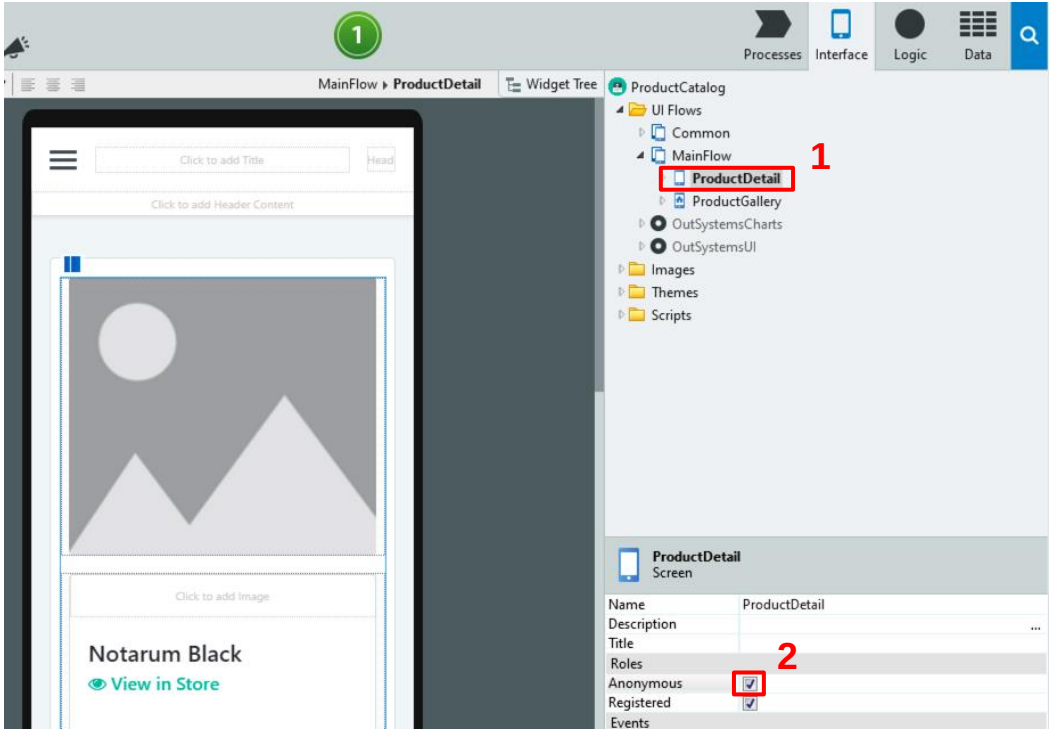
2. Right-click on **MainFlow** and **Add Screen**

3,4,5. Select the **Product Detail** Template and name your screen as **Product Detail.**

6. Click on **Create Screen**

Set it as anonymous

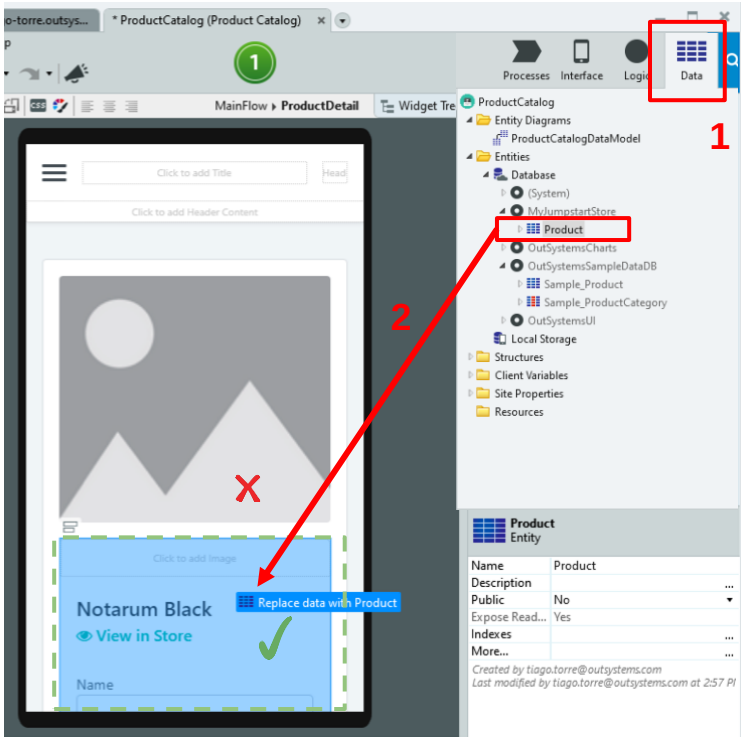
1. Select the “ProductDetail” screen from the MainFlow
2. From the screen properties, Check the **Anonymous Role**



Use the Product Entity data

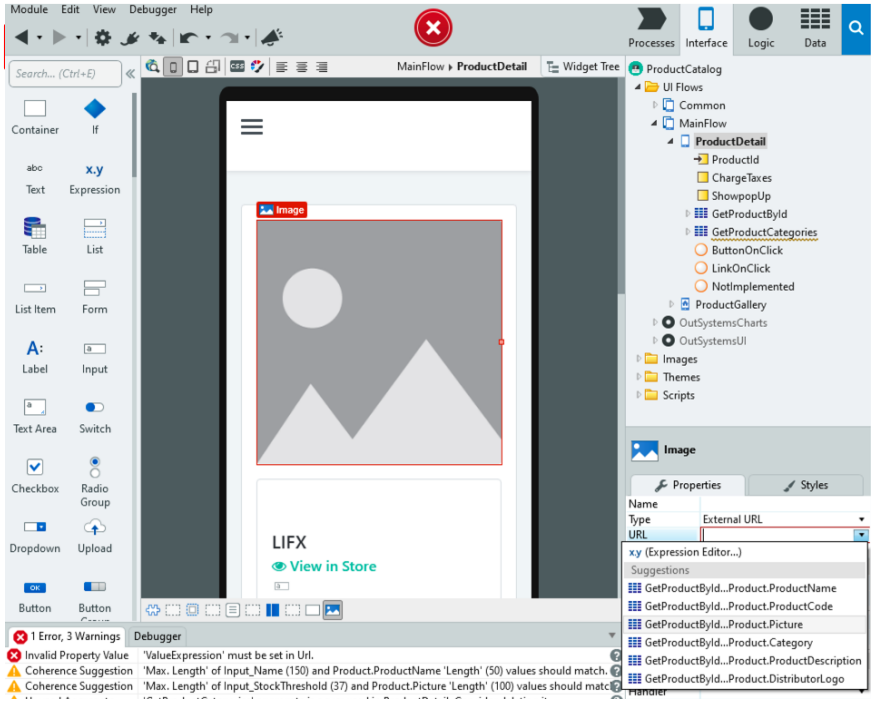
1. Go to the **Data tab**

1. Replace the data with the **Product** entity, by dragging the Product entity onto the lower part of the screen.
   Make sure you see "**Replace data with Product**" before letting go of the mouse button.



Fix error *Can't identify 'Image' element in the expression*

1. Select the "Image" widget of the ProductDetail screen.
2. On the Image Properties tab set the following attributes:
   - **Type**: External URL
   - **URL**: Select the suggested GetProductById…Product.Picture



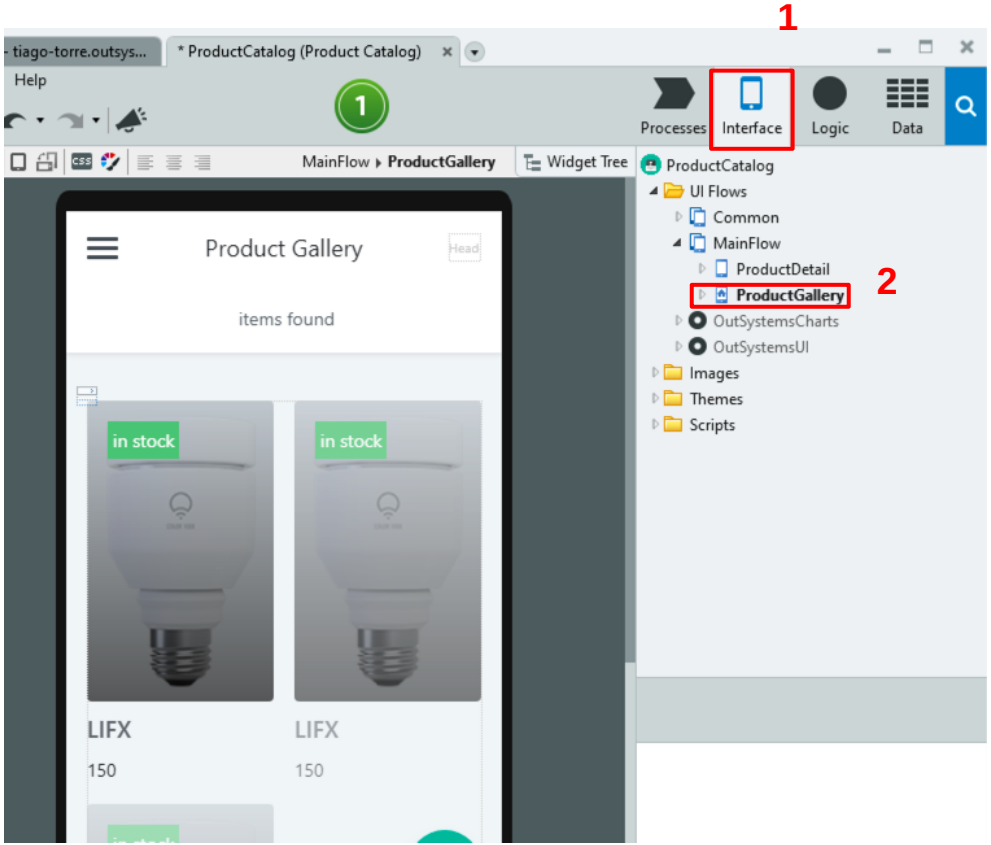
B. Link Product Gallery to Product Detail

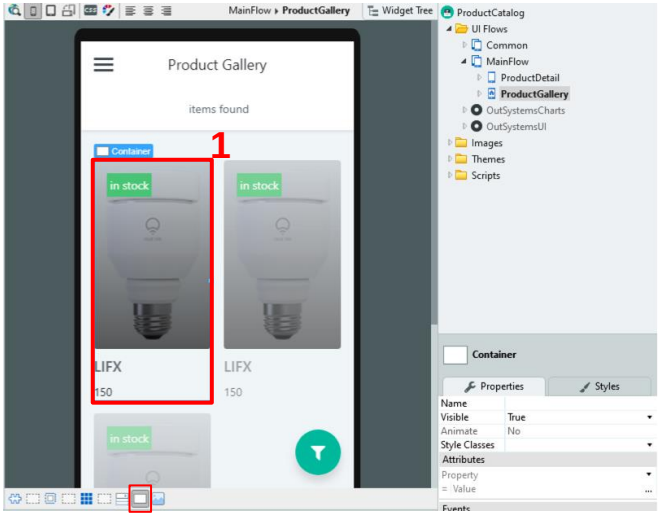
Now that we have replaced the Product Detail screen with the Product Entity data, we want to enable users to choose a product from the Product Gallery and see its details.

Open the screen

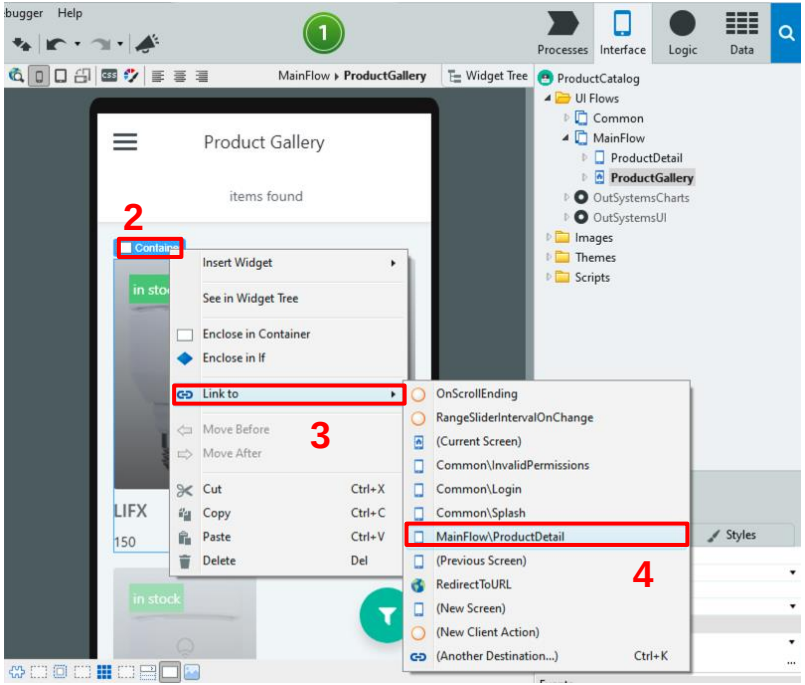
1. Go to the **Interface** tab

1. Double click the **Product Gallery** screen





1. Select the **Container** widget



2. Right-click in the **Container** (Container) icon
3. Select **Link to** and then
4. **MainFlow\ProductDetail screen**.

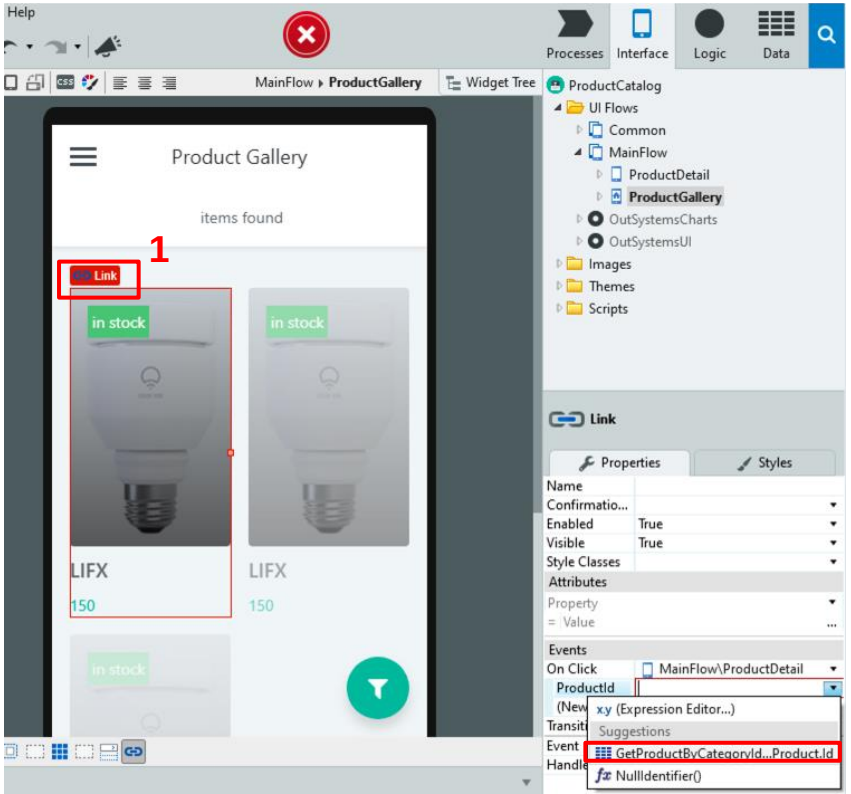
QUICK TIP

To help you select an enclosing element, you can use the **breadcrumbs** on the bottom of the main view:

Assign Product Id

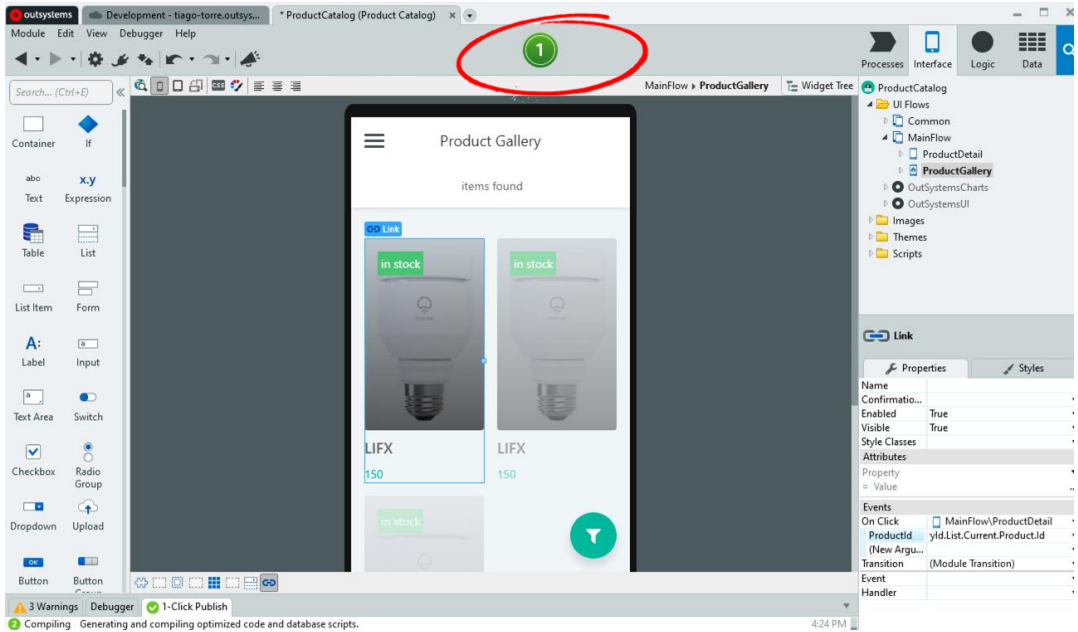
1. Make sure the **Link** widget is selected

1. On the **On Click** event assign to the **ProductId** the value **GetProductByCategoryId….Product.Id**



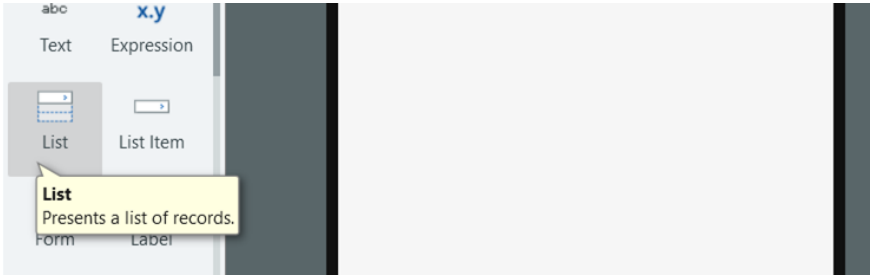
Deploy and Test it

1. At this point you can deploy your application by clicking the 1-click-publish button ( ) a test it ( ).

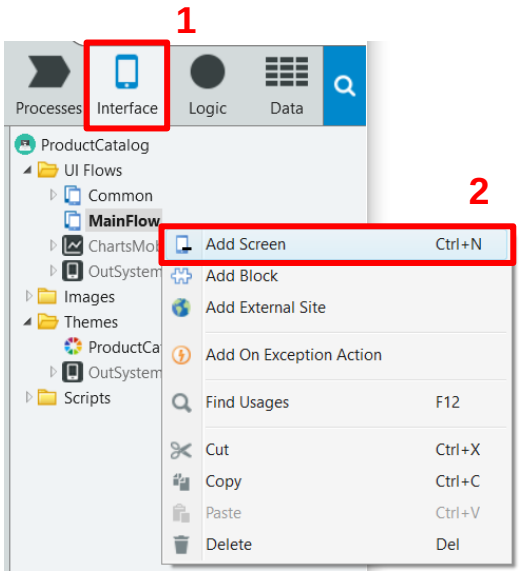



outsystems
Section 4
Product Listing

Similar to building apps in Web, in mobile, there are a few ways of building a listing screen. The manual way is to drag the List widget into the screen, and then add a List Item widget. Then build your preparation function that queries the details from the database.

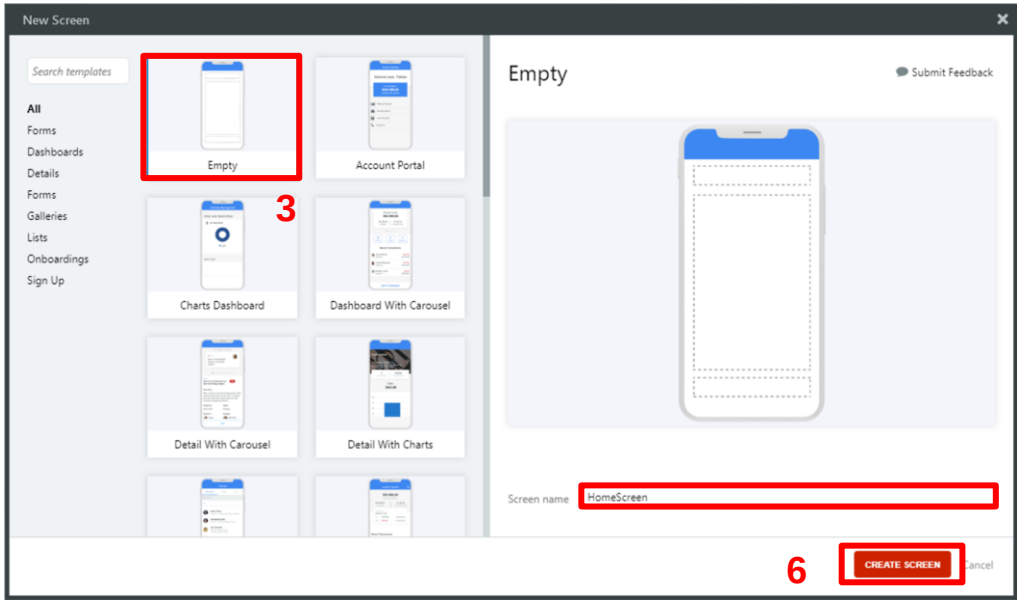


Within this exercise, we will instead use OutSystems accelerators to automate building the listing screen.



1. Go to the interface tab

2. Right-click on **MainFlow** and **Add Screen**

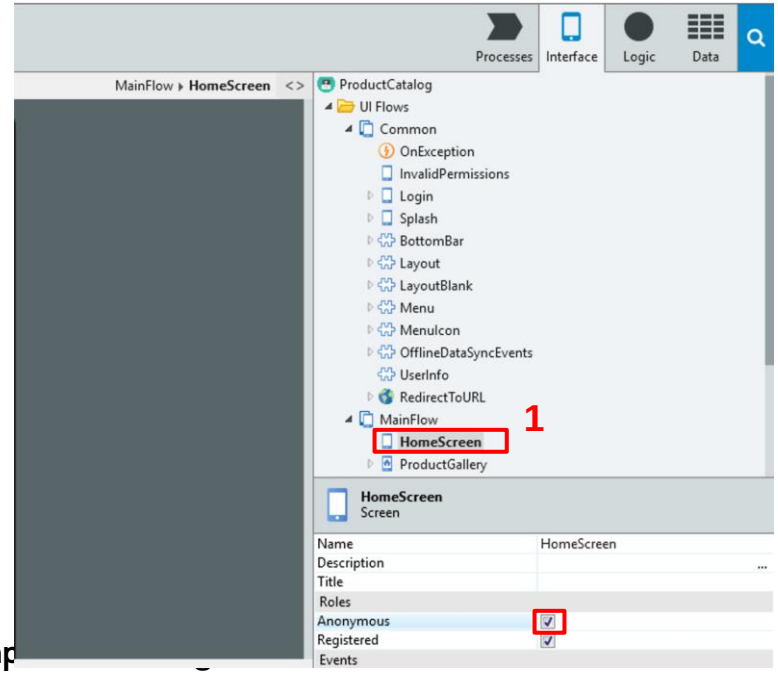


3. Select the **Empty Template** and name your screen as **HomeScreen.**

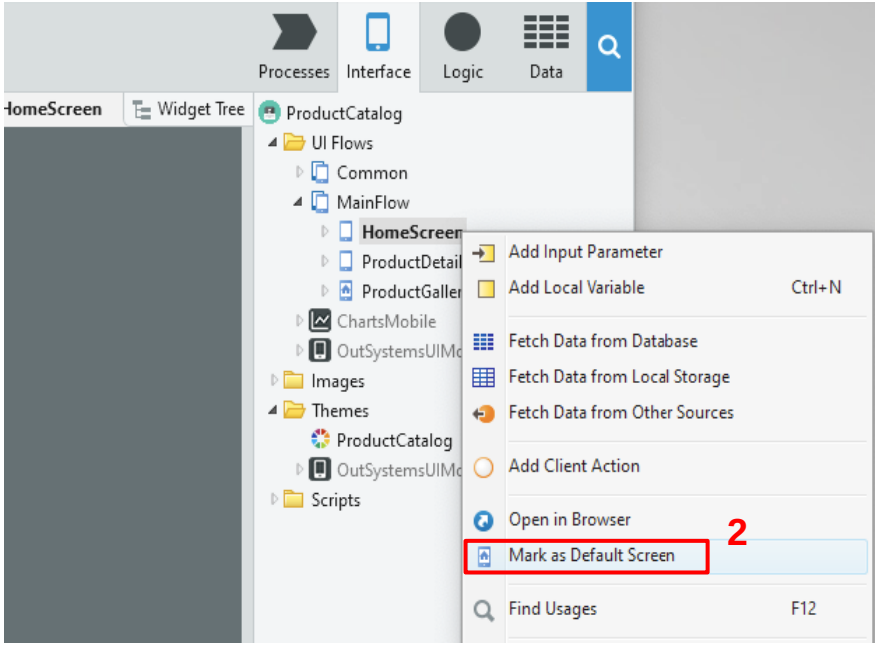
4. Click on **Create Screen**

1. Go to the interface tab, under the **MainFlow**

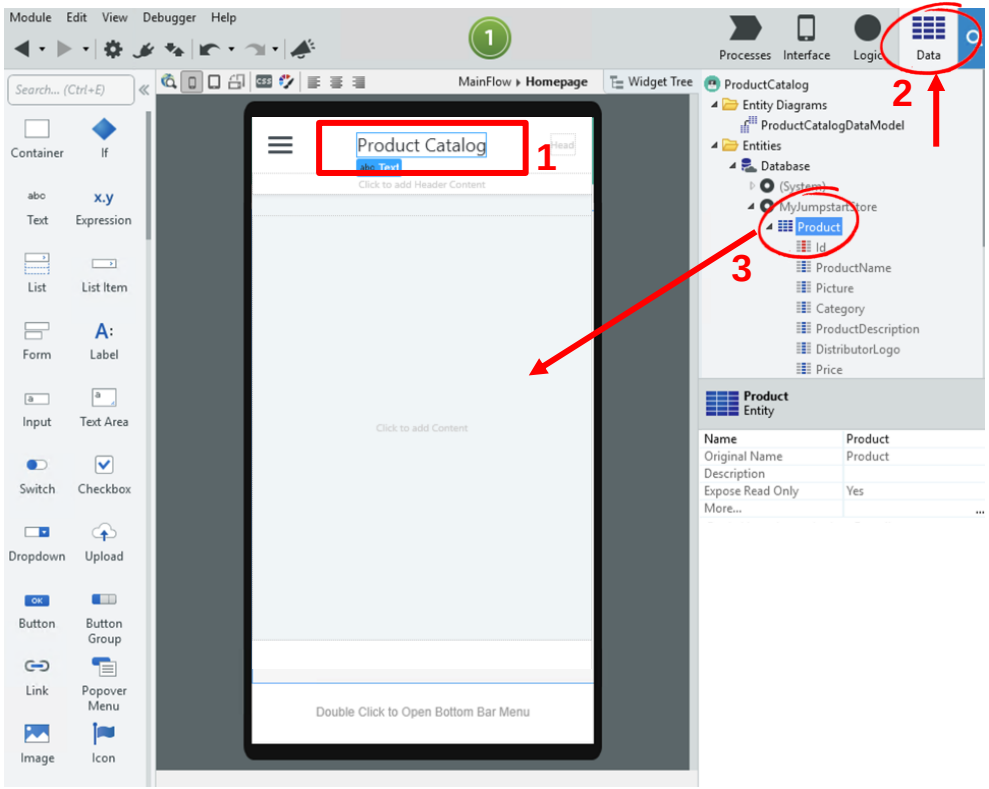
   Set the “HomeScreen” screen **to Anonymous**



2. Right-click “HomeScreen” and select **Mark as Default Screen**



1. Add a title. Type **Product Catalog** in the Title area of your mobile app.
2. Click Data tab.
3. Drag and drop the **Product** entity into the Content area.



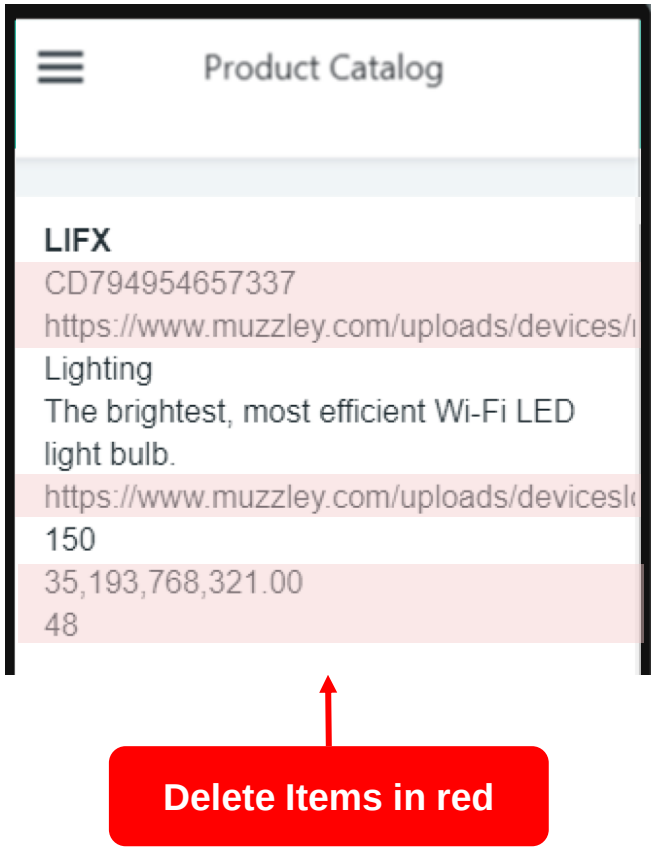
This will automatically create the listing widgets within the screen and the preparation logic that queries the data from the Product database.

Unnecessary data can be removed

We now have a list of products!

1. You can remove unnecessary attributes that we do not need.

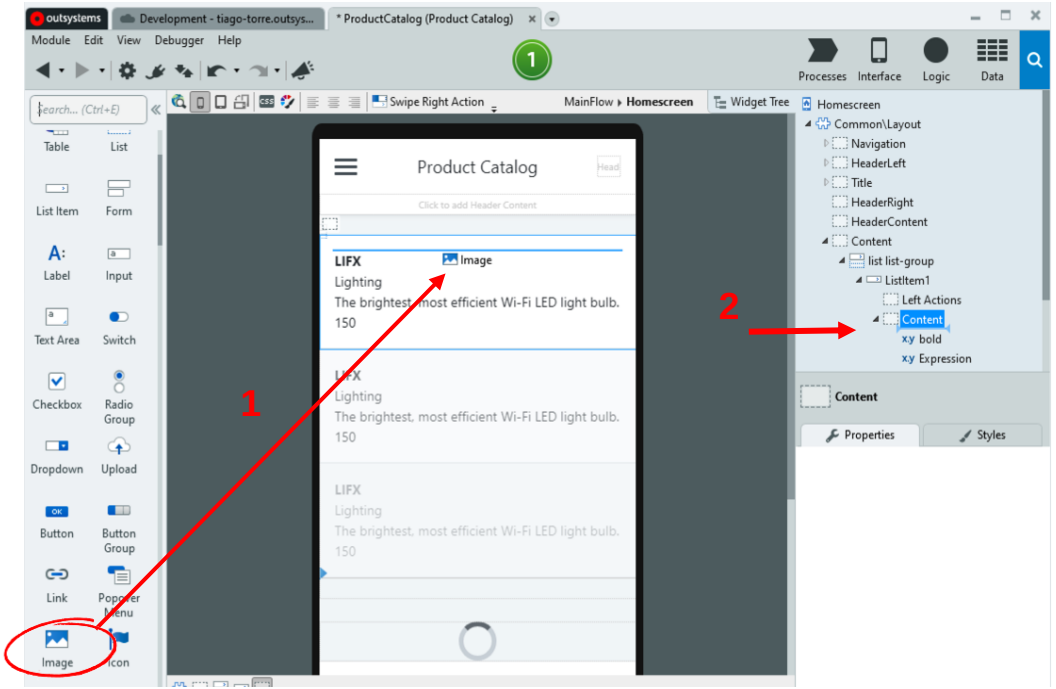
1. Multi-select the items using **CTRL+click**, then press **Delete**.



Drag and Drop the image widget

1. Add an **Image widget** into the top of the first attribute

1. Make sure the **image widget** is inserted on the correct position



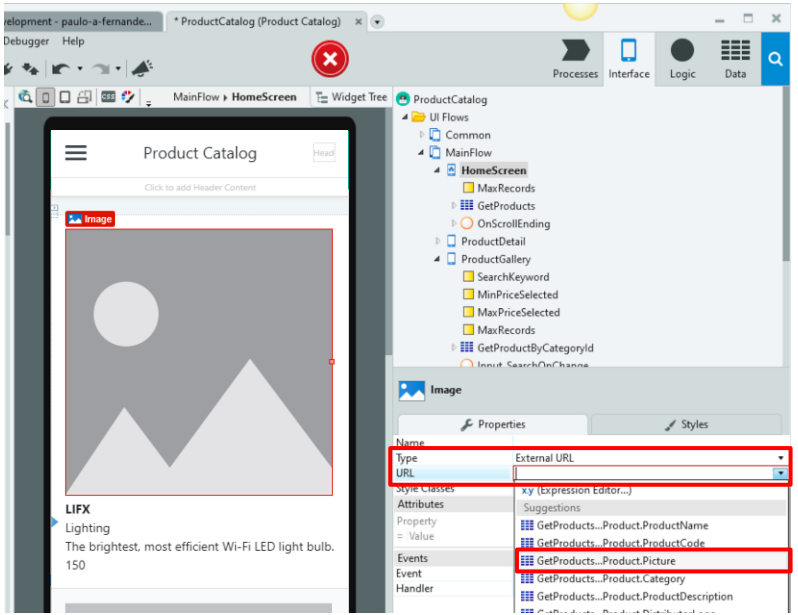
QUICK TIP

The **Widget Tree** is a great way of locating elements in the screen structure.

It can be manually opened by clicking the upper right icon (Widget Tree

It also **opens automatically** when you drag a widget onto the screen!

1. On the Image Properties tab set the following attributes:
   - Type: External URL
   - URL: Select the suggested GetProducts…Product.Picture



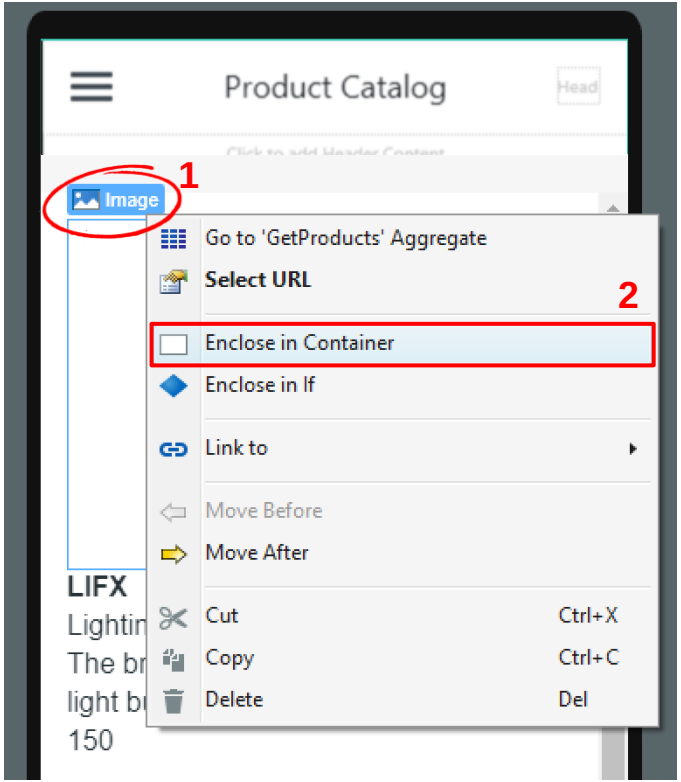
We will format the list items to show 2 columns. We want to have the product image on the left column, and the product description on the right column.

To do this, we will enclose the objects into Containers (similar to in HTML)

Enclose image in container

1. Right-click on the **image widget**

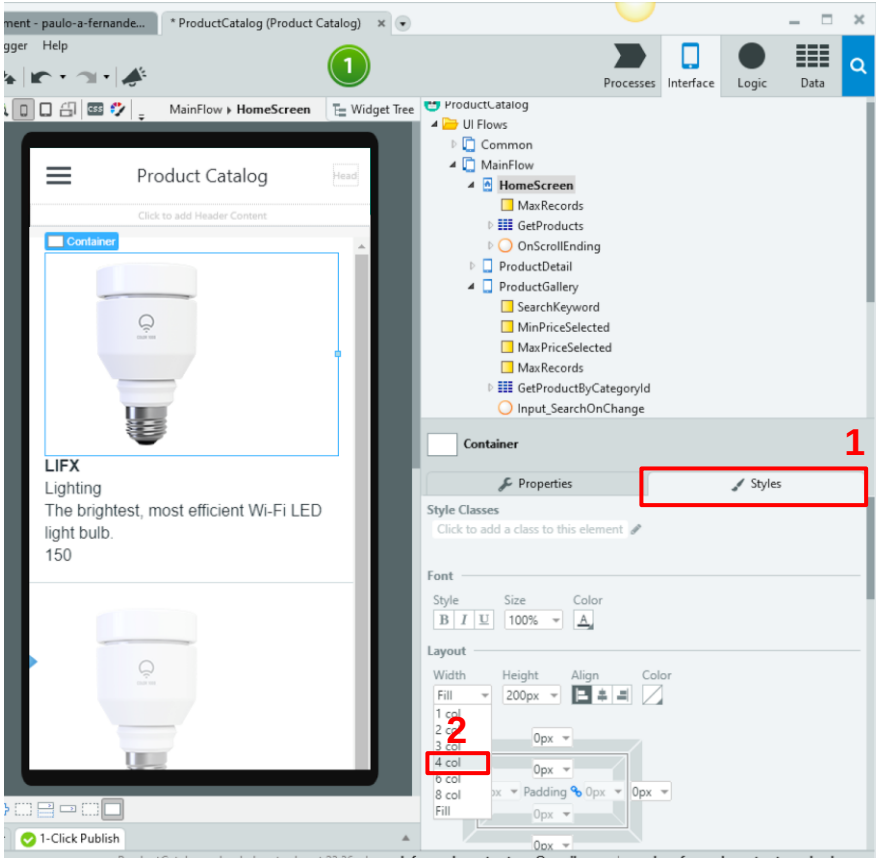
1. Click on **Enclose in Container**



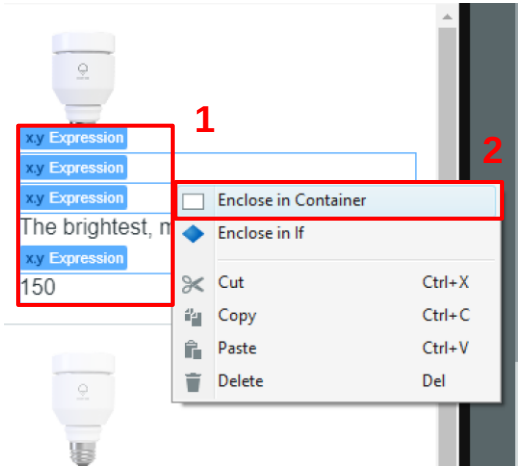
Set container width

1. With the container still selected, toggle to the Style Editor ()

1. Set the **width** to **4 col**

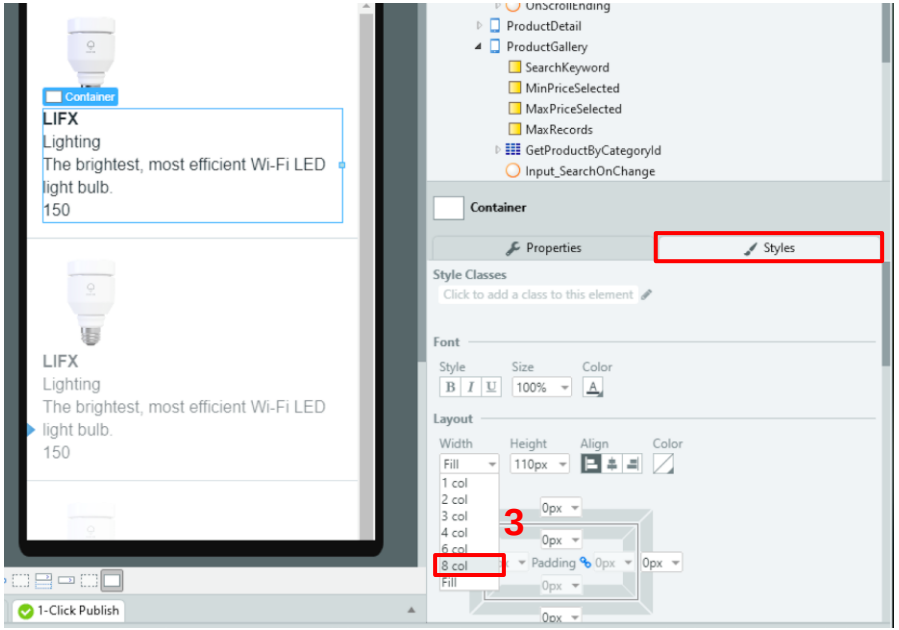


Adapt text attributes look and feel



1. Select all the text attributes **(Shift + Click)**

1. Right click > **Enclose in Container**



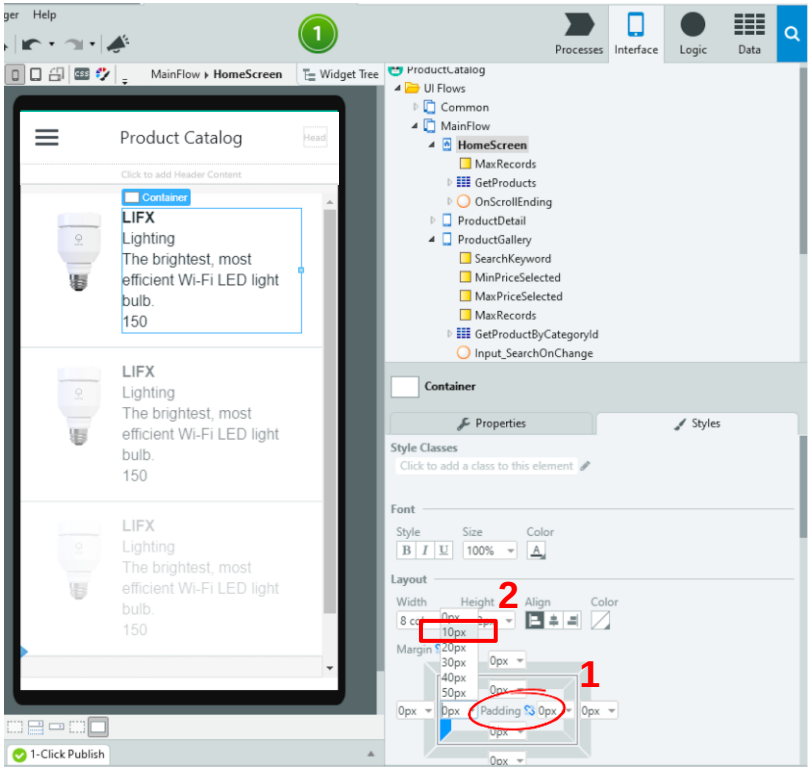
3. With the container still selected, toggle to the **Style Editor** ( ) and se **width** to **8 col**

Add space between image and text

You now have 2 columns. But there's no space in between the image and the text.

1. With the same **container** still selected, click the **chain icon** ( ) unlock the padding lock.

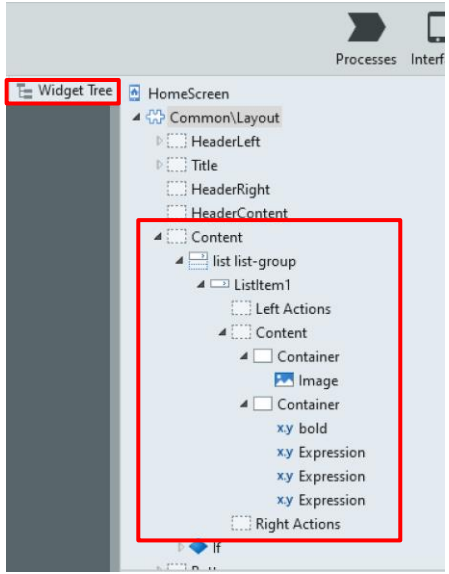
1. Then change the style attributes to have a **Padding-left** of **10px**.

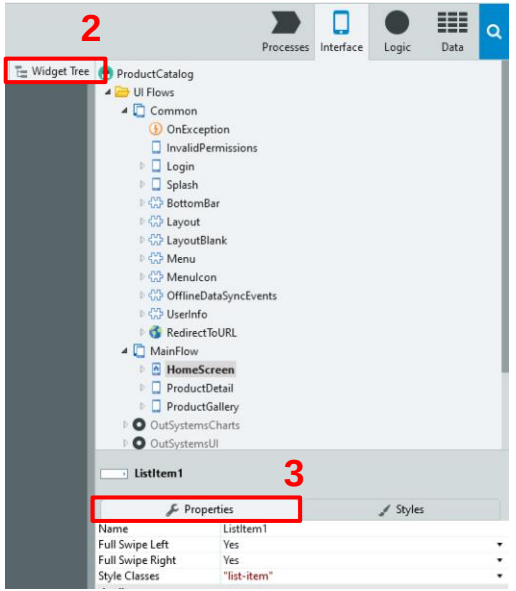


1. Navigate to the image block in the **widget tree** and make sure your structure looks like the following (see above)

1. Click on the **widget tree** to go back to the Navigation pane.

1. Click on the **Properties** tab.

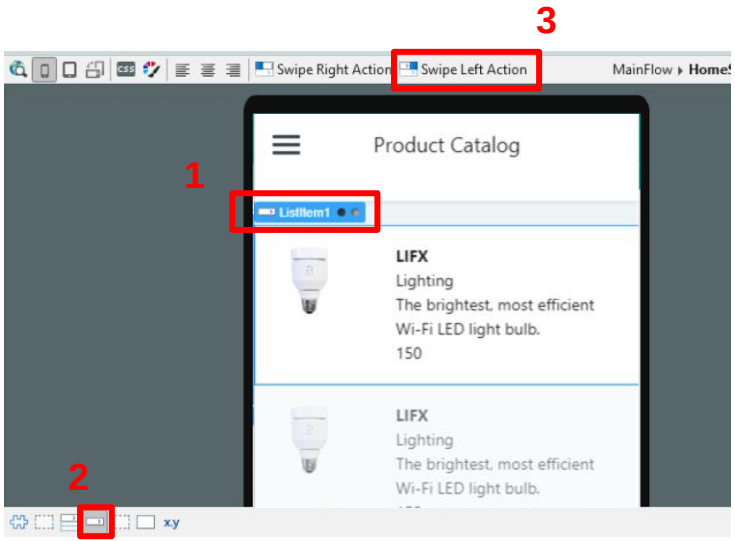




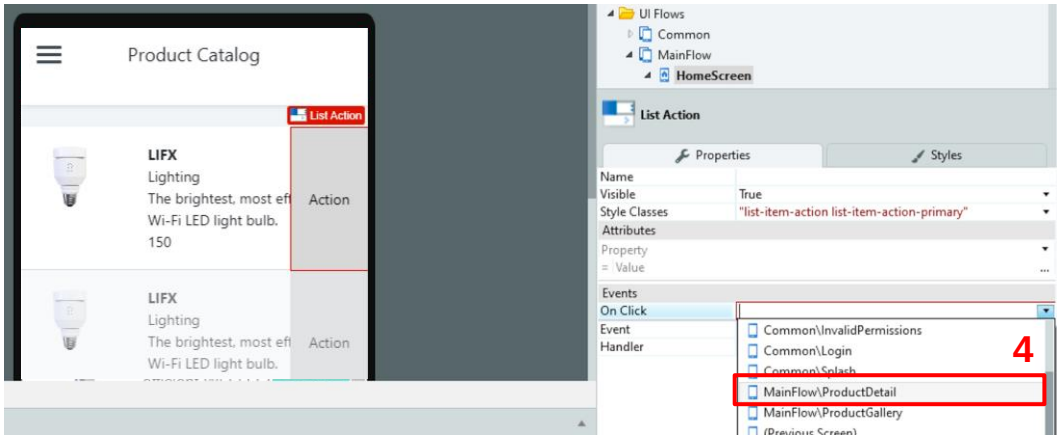
Now that we have a product listing screen, we want users to be able to “swipe” each item to view its detail screen.

To do this, we will add a swipe action to each list item so that users can see the product details when swiping left.

Add swipe action to list item



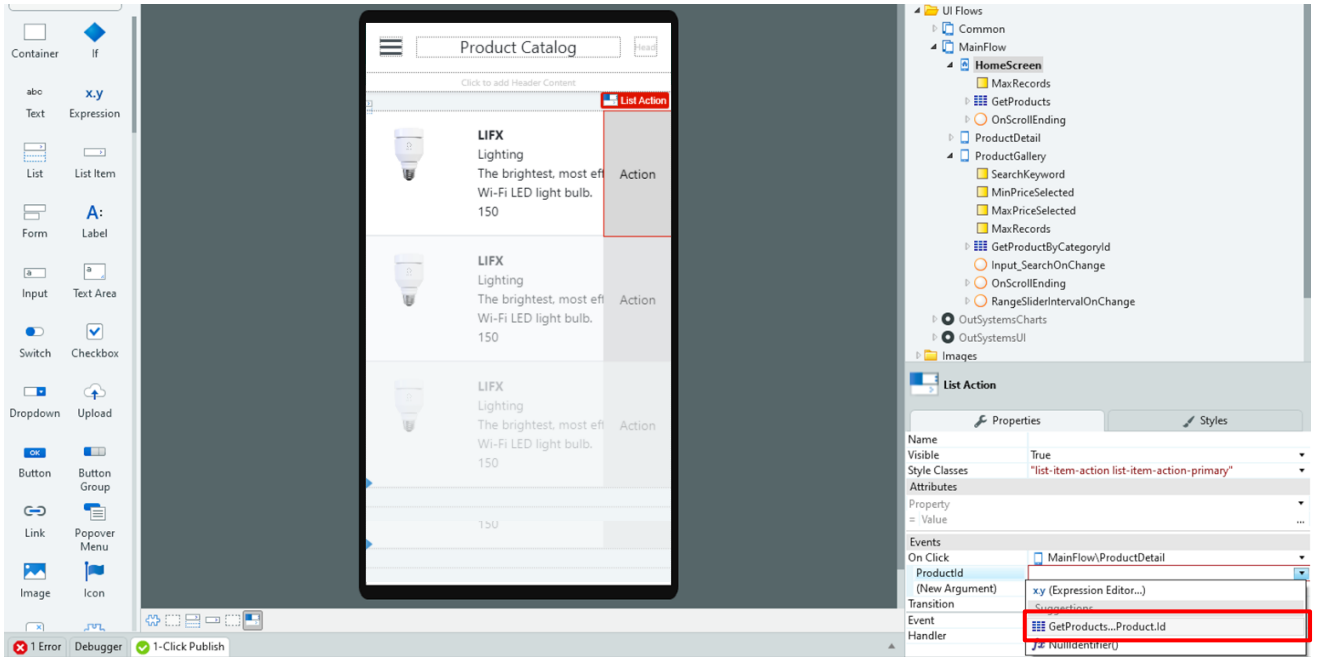
1,2,3. Select the List Item (make sure the Ic… ” is selected), and then **Swipe Left Action**.



4. In the **On Click** event, select **MainFlow\ProductDetail** to link to our Detail Screen.

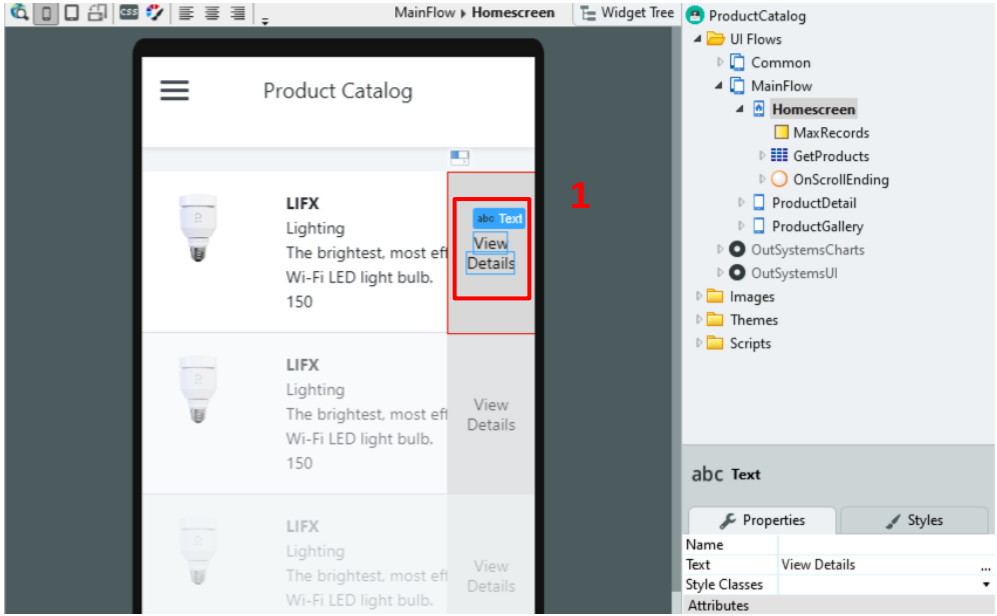
Add missing parameter

1. Add the missing input parameter **GetProducts….Product.Id.**



Add swipe action to list item

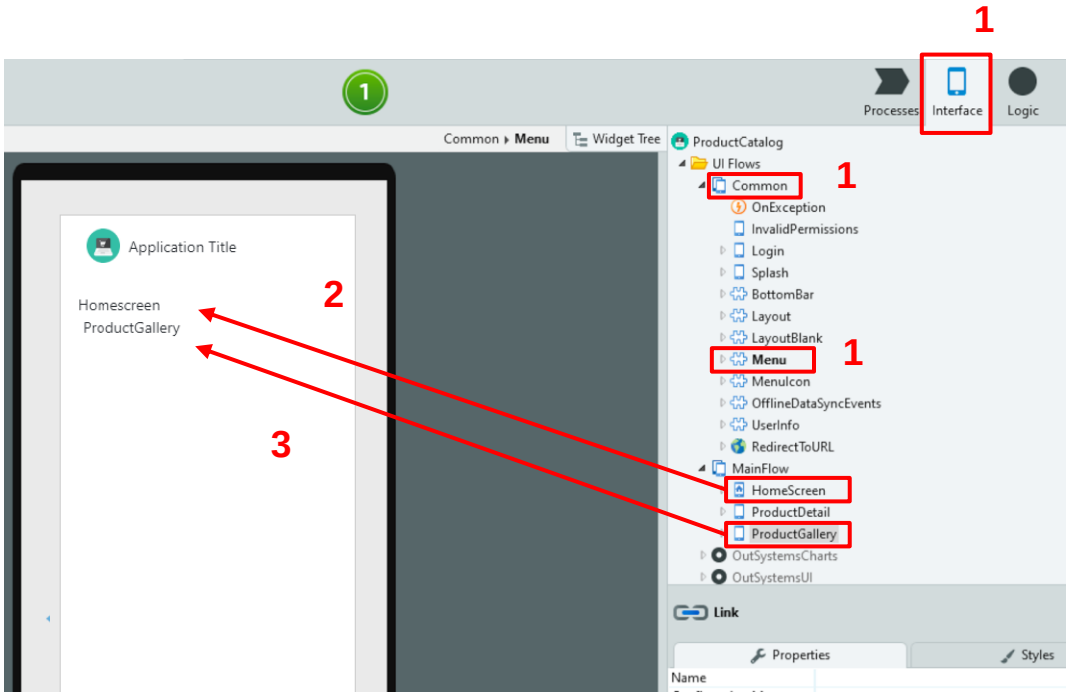
1. Change the action name from “Action” to “View Details”



1. Go to the **Interface** tab, open the **Common** flow, and double click the **Menu** block

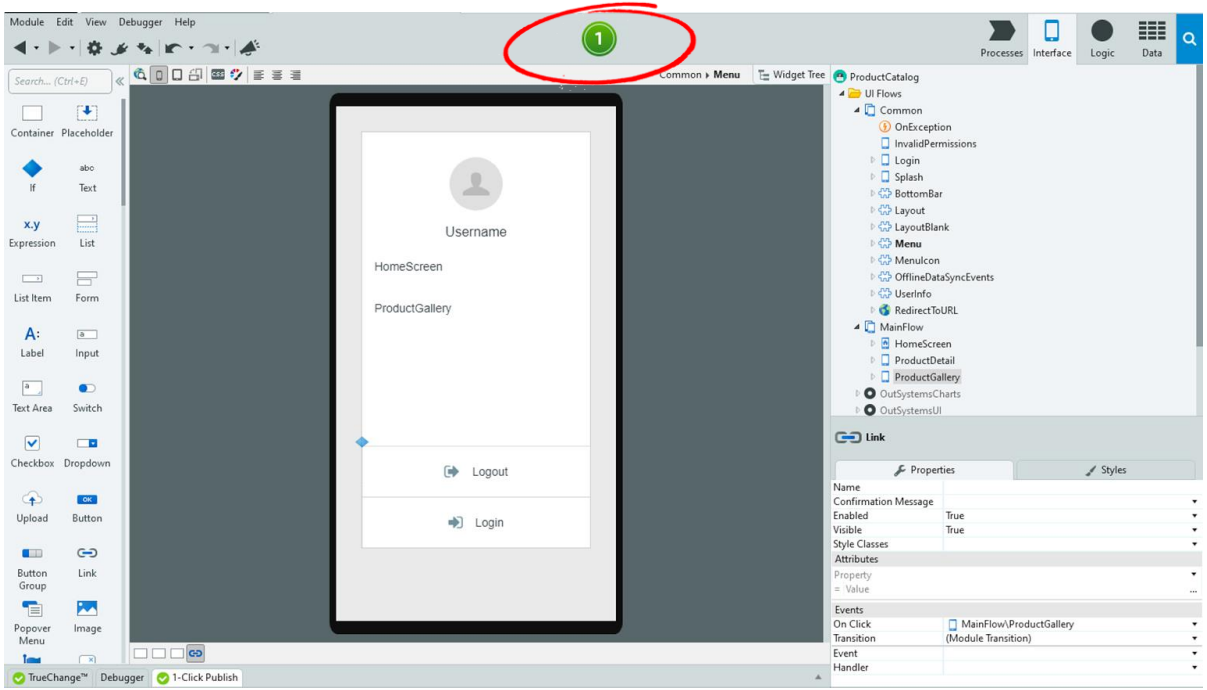
1. Drag the **HomeScreen** screen to the **Menu** block

1. Drag the **ProductGallery** screen to the **Menu** block



Deploy and Test it

1. At this point you can deploy your application by clicking the 1-click-publish button ( ) and test it ( ).



Section 5

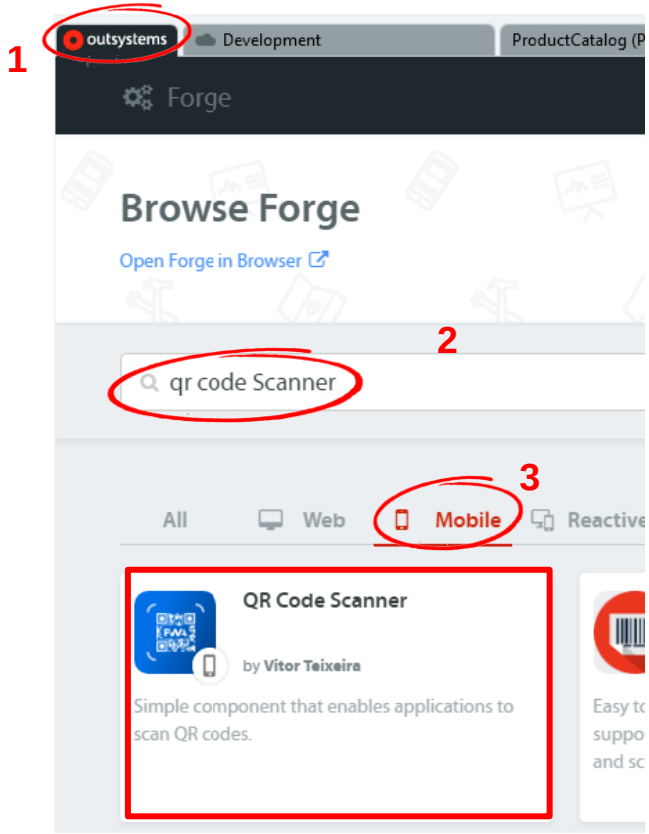
Mobile QR Code Scanner

We now have a list of products. But how can we filter the products based on a QR Code, using the mobile camera?

The first thing we need to do is to install the QR Code Scanner plugin on our environment.

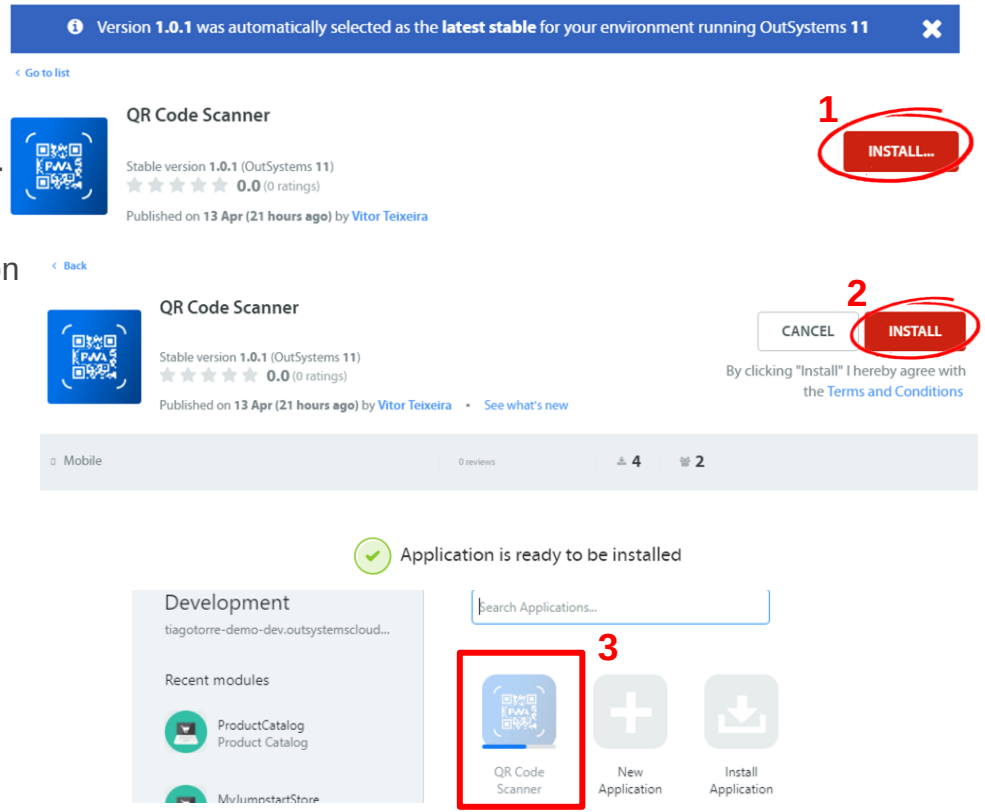
QR Code Scanner Plugin

1. In Service Studio, navigate to the OutSystems tab and make sure you are logged in (or navigate to https://www.outsystems.com/forge/) .
2. Search for the application “QR Code Scanner”
3. Filter Mobile Component
4. Click “QR Code Scanner” Plugin



QR Code Scanner Plugin

1. Click Install to install the component.
2. It will search for dependencies. Click Install again.
3. The application will install, together with any associated dependencies. Wait for your installation to complete.
4. You just installed your new Forge component!



Later you will learn how you can use this component in your apps by adding it as a dependency.

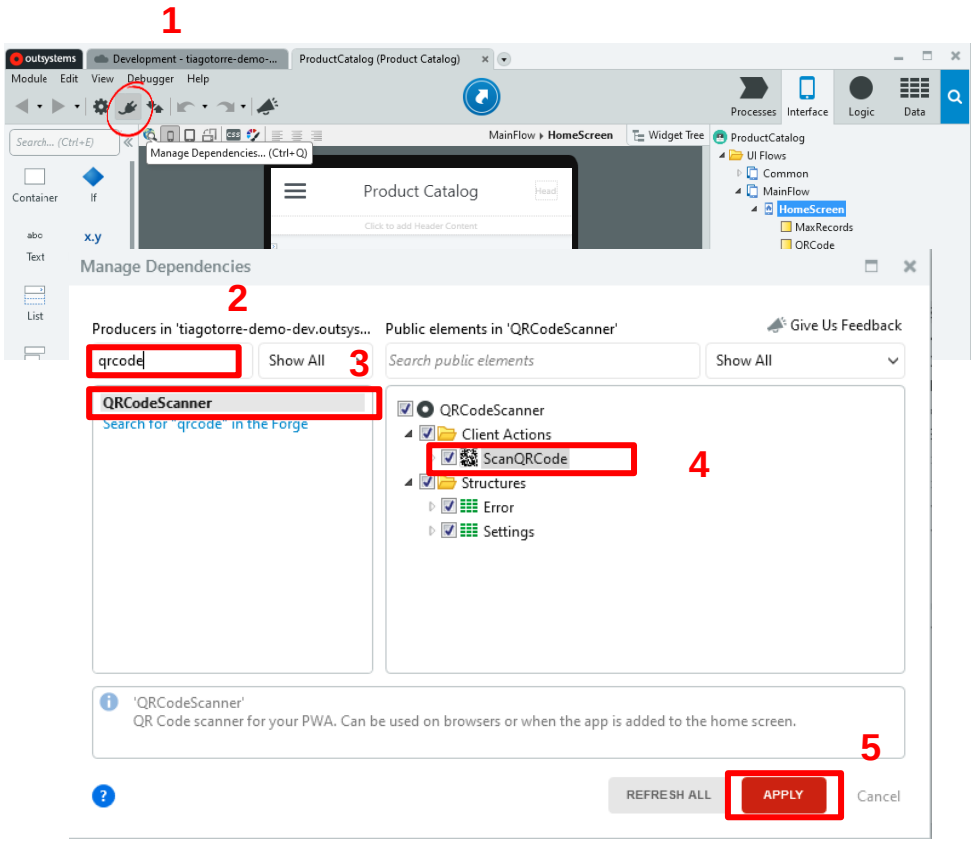
Now that we have the QR Code Scanner plugin installed in our environment lets include it in our mobile application by adding a new dependency.

Remember dependencies are similar to “References” in .Net & Java enabling you to reuse Forge components, or pre-created templates, plugins, objects, methods, etc from existing applications.

Add QR Code Scanner Plugin

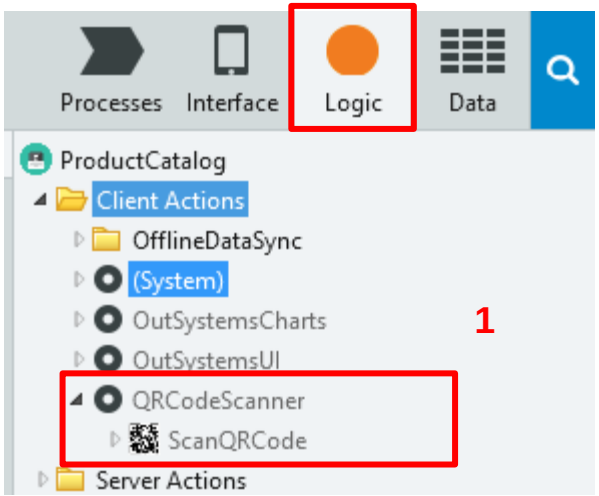
Let's use the Forge Component installed in the previous steps.

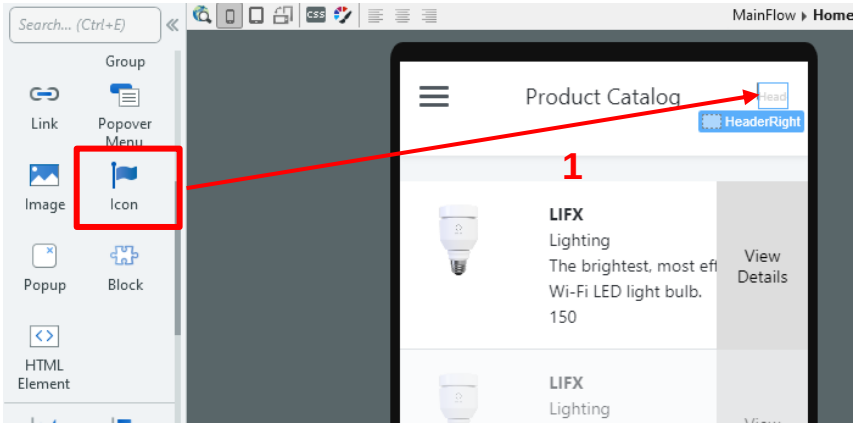
1. Click the **Manage Dependencies** icon
1. Under Producers, search for the **QRCodeScanner.**
1. Select the **QRCodeScanner.**
1. In the elements, select:
   - **ScanQRCode**
1. Click the **"Apply"** button

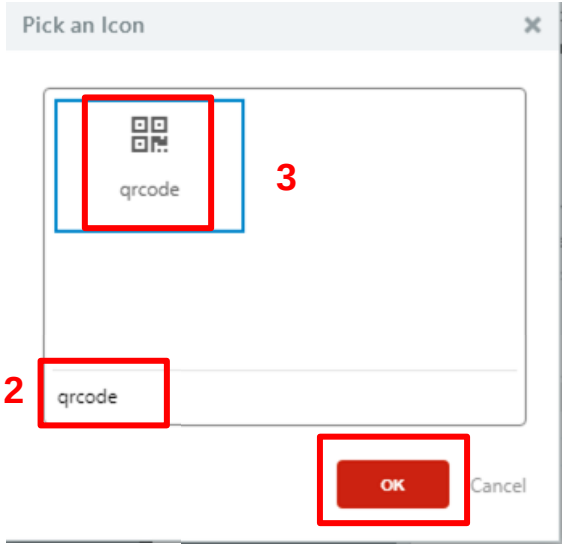


Review Dependencies Added

1. After adding the references, you should now see the **ScanQRCode** actions in the logic tab of your project, under **Client Actions > QRCodeScanner**





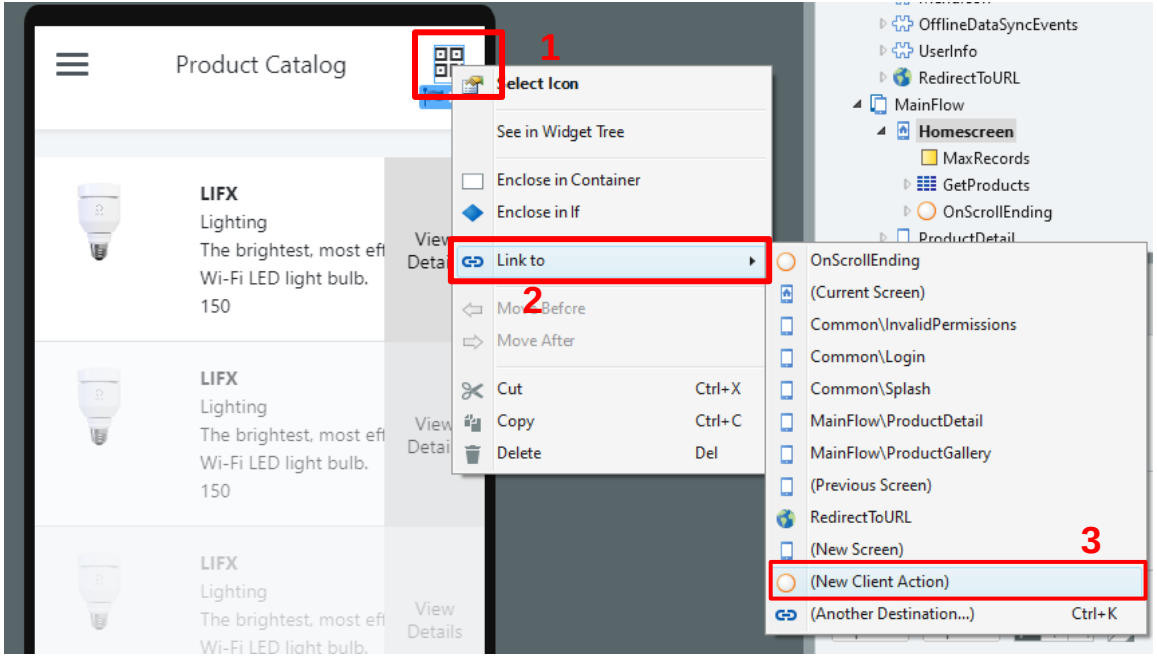


Let's create a button to trigger the QR Code.

1. On the **Interface** tab open the **HomeScreen** screen and drag and drop an **Icon** widget into the **HeaderRight** area.
2. When prompted for the Icon type, search for **qrcode**
3. Select the **qrcode icon** and click **OK.**

New Client Action

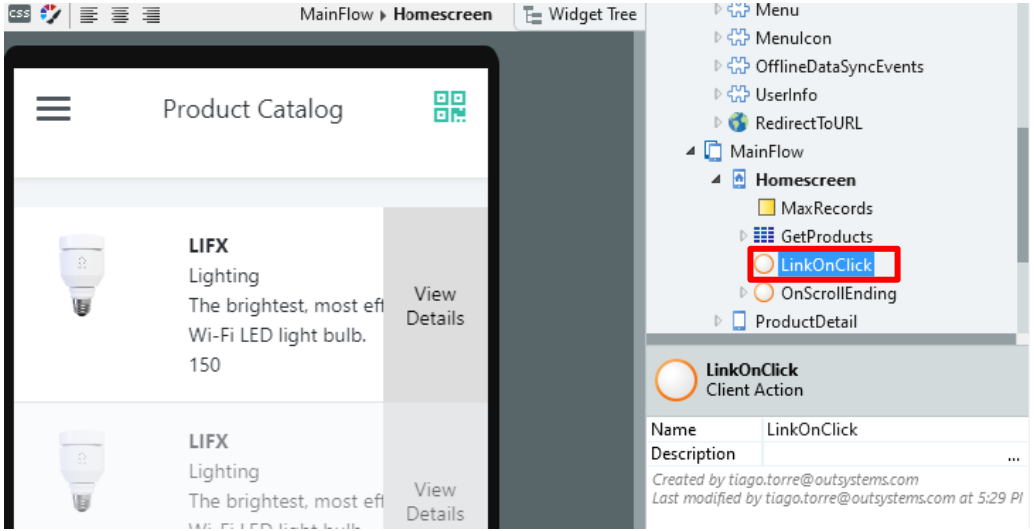
1. **Right click** the new QR code icon
2. Click on **Link to**
3. Click on **New Client Action)**



New Client Action

1. Notice that the “**LinkOnClick**” client action has been created. This action will be triggered when clicking the icon.

1. Double-click the “**LinkOnClick**” client action



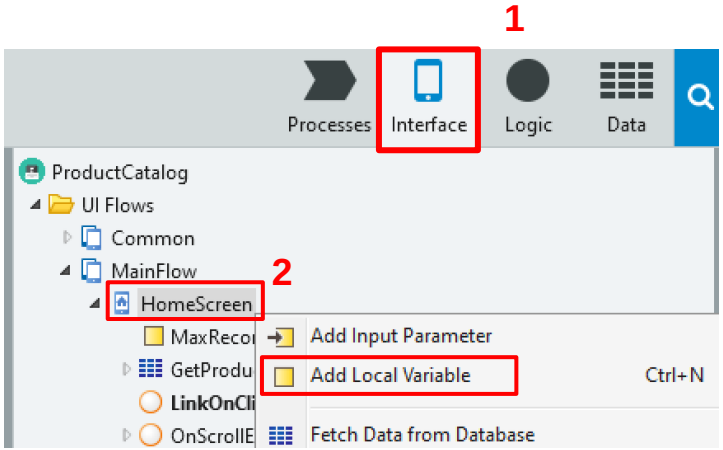
ScanQRCode Action

Look for the **ScanQRCode action** that you have added as a dependency earlier.

1. This is found under **Logic** > **Client Actions** > **QRCodeScanner Plugin** > **ScanQRCode**.
2. **Drag and drop** this into your **Action Flow**.

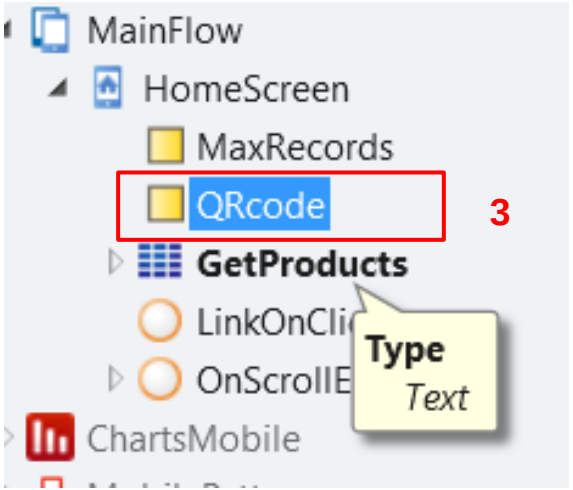
This action will trigger the user's mobile device camera to scan a QR code.

QRCode Variable



We need to store the QR code scan results into a variable, so that we could use the variable to filter the list results.

1. To add local variables, Click on the **Interface tab**
2. **right click** the **Products** screen > **Add Local Variable**

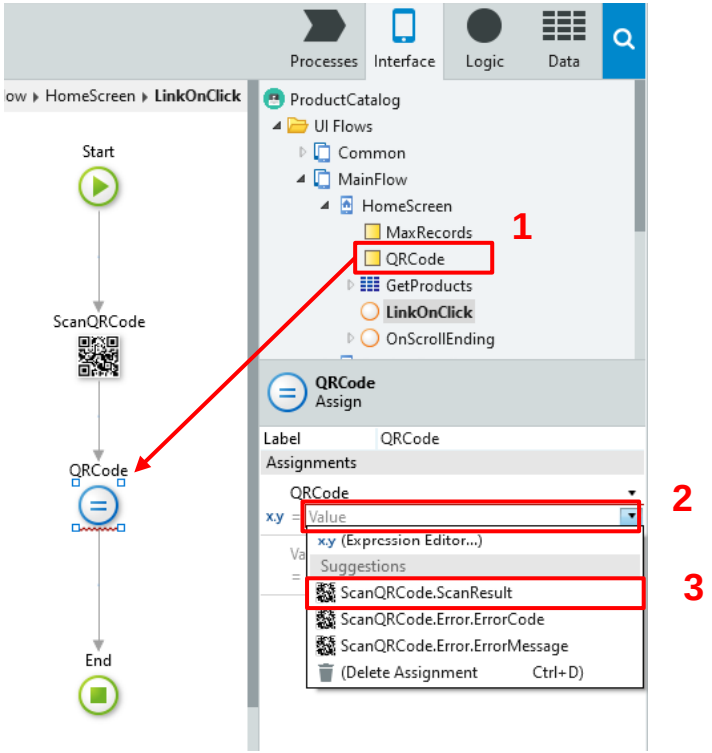


3. Name the variable as **QRcode**. Make sure its Data Type is **Text.**

We need to assign the scan results into the new **QRcode** variable.

1. To do this, drag and drop the **QRcode** variable into the action flow

1. Set the Assignment to **QRcode = ScanQRCode.ScanResult**



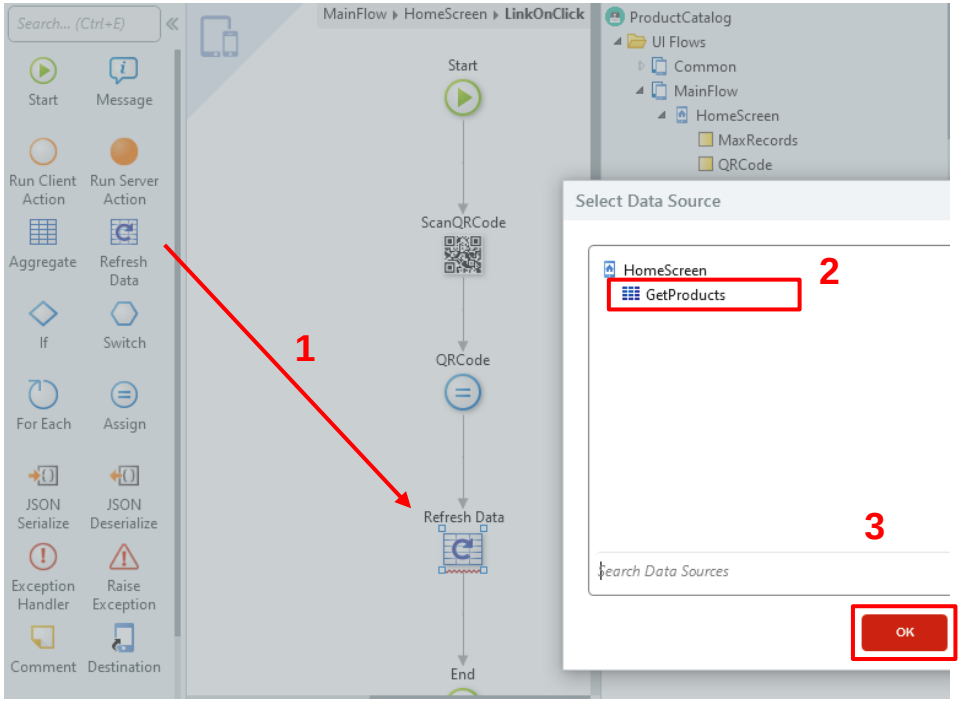
Refresh Products List

After scanning the QR Code, we need to refresh the Products list in the screen to reflect our filter (we will do the list filters later).

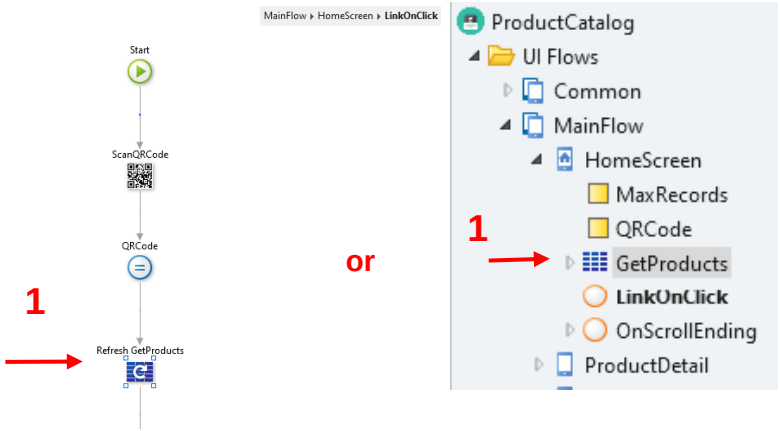
1. Use the **Refresh Data** widget.

1. When prompted for the data source to be refreshed, select **GetProducts**.

1. Press **OK**

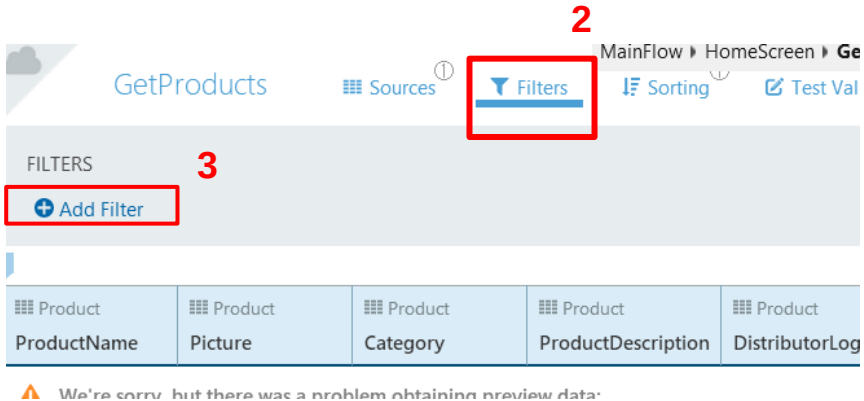


Add Aggregate Filter



We need to re-configure our query to ensure that the list is filtered based on the QRcode value.

1. Open aggregate editor by double clicking the **GetProducts** aggregate.



2. Click **Filters**, and **Add filter**

The GetProducts aggregate is the query that the HomeScreen's List gets its values from

Add Query

1. Enter the following query:

   QRcode="" or Product.ProductCode = QRcode

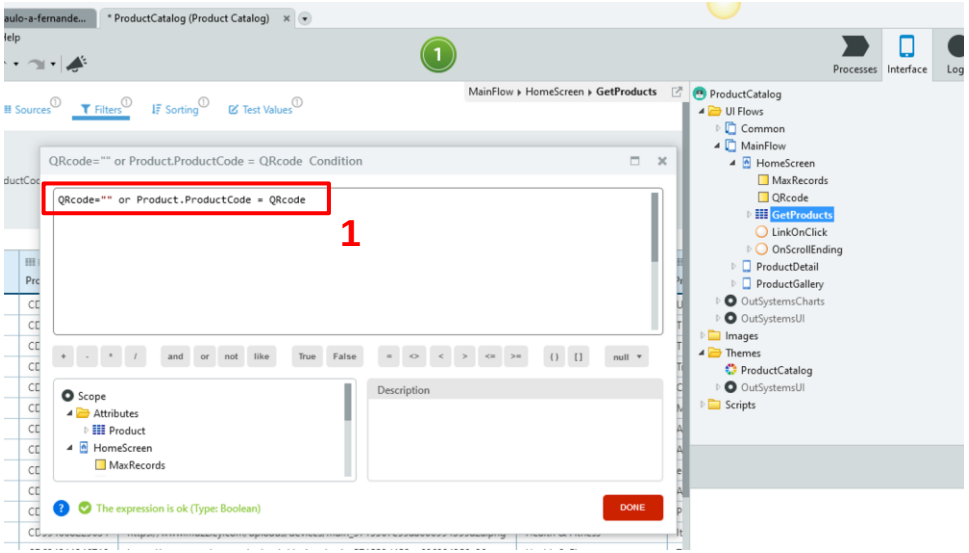
Explanation of the query:

- QRcode =""

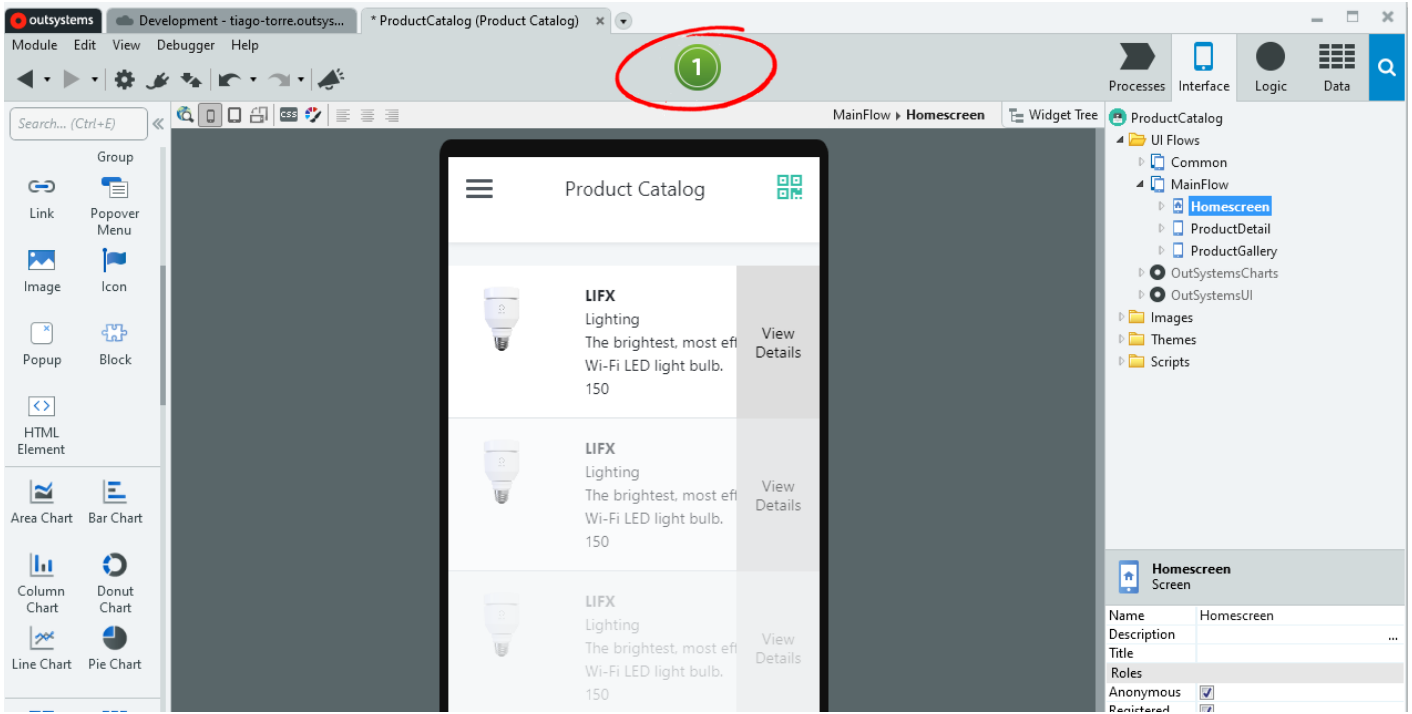
  This query will ensure that the list will not filter the query if there is no value in the **QRcode**.

- or Product.ProductCode = QRcode

  Otherwise if the **QRcode** is not empty, we will filter based on the **Product Code**.

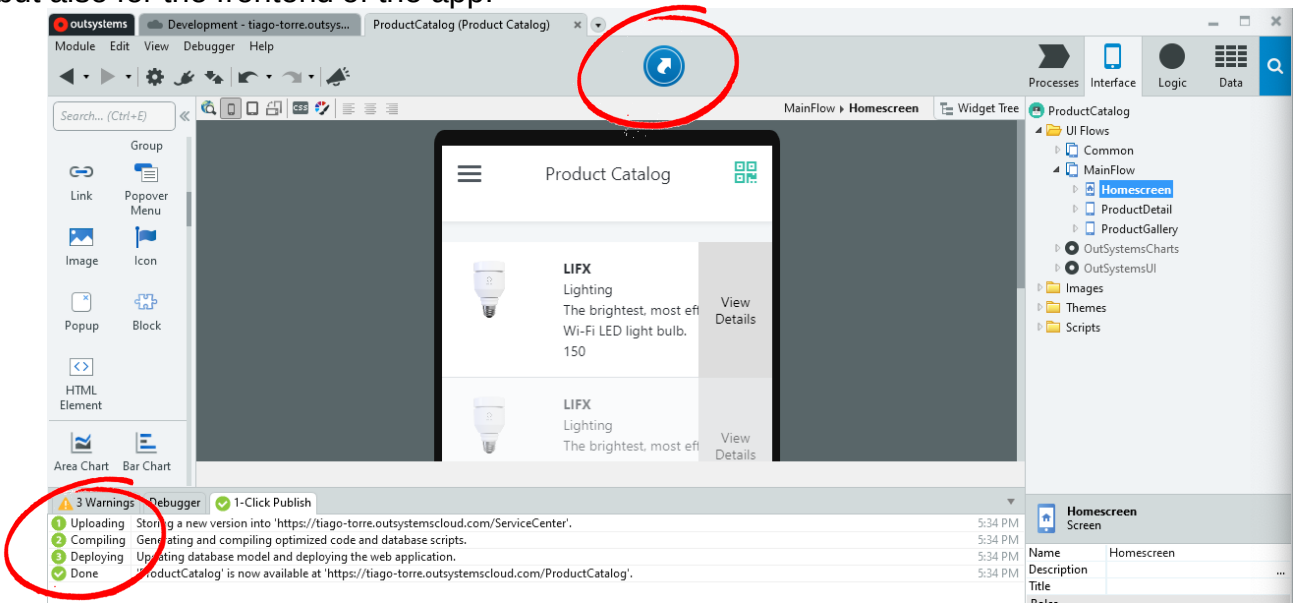


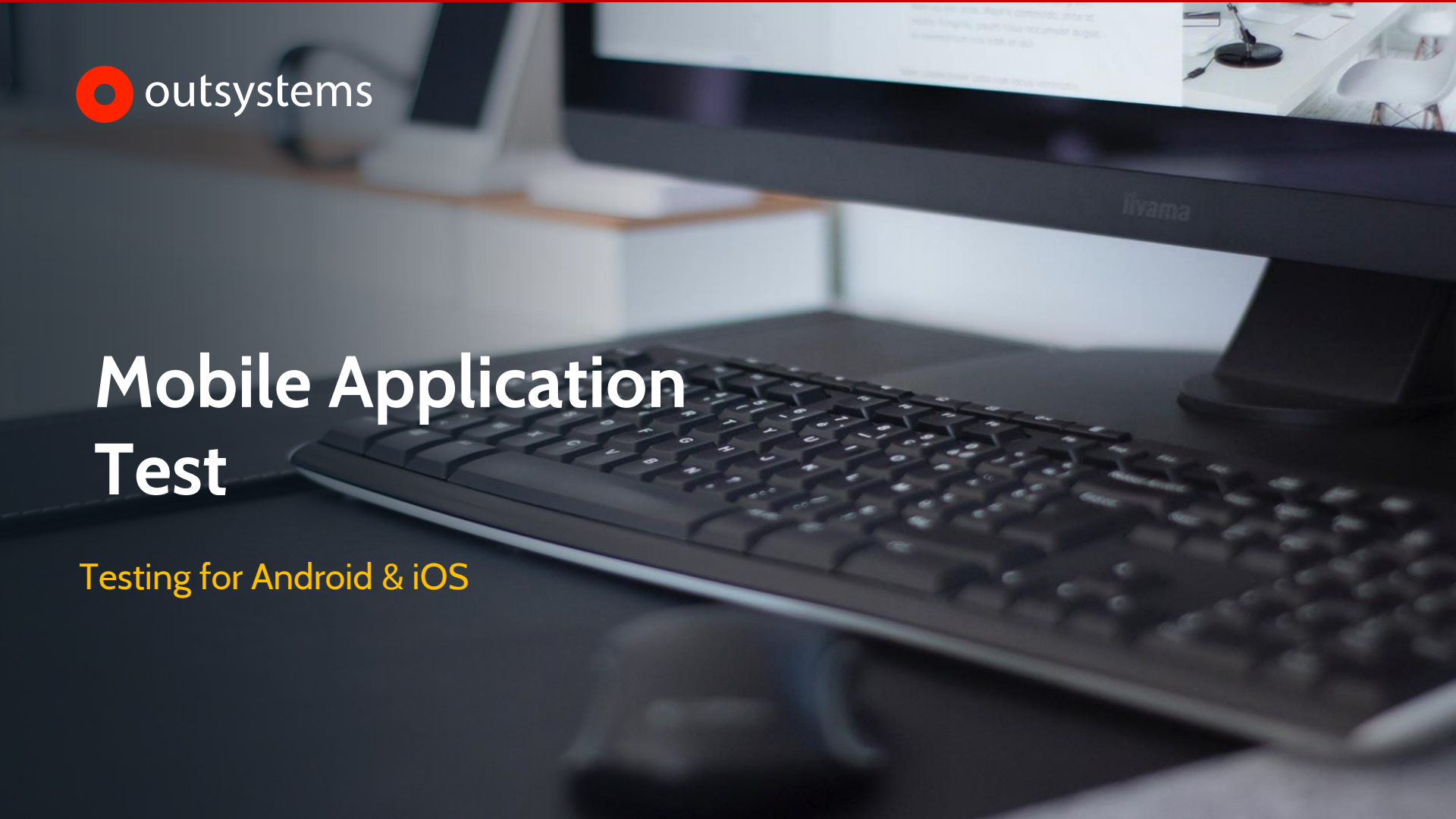
Publish Application



Test Mobile Application

1. While deploying the system will **generate standard code (.NET , HTML5, CSS3, React, JS, SQL,...)**, not only for the backend but also for the frontend of the app.

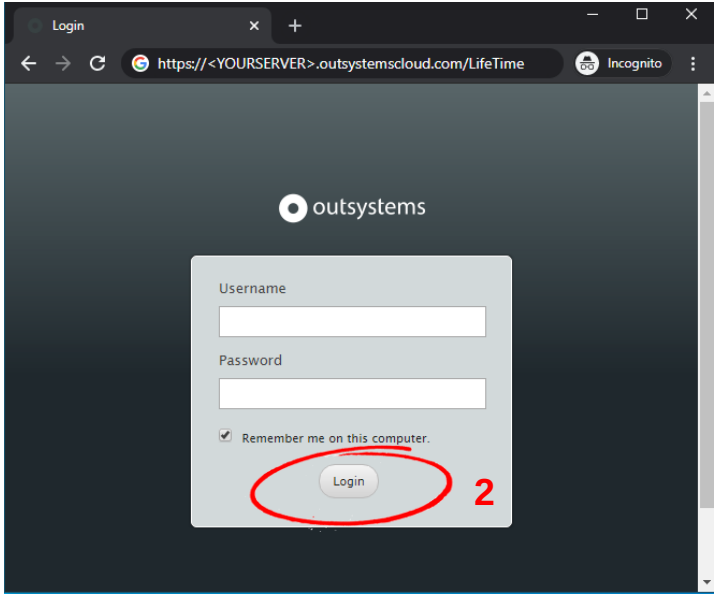


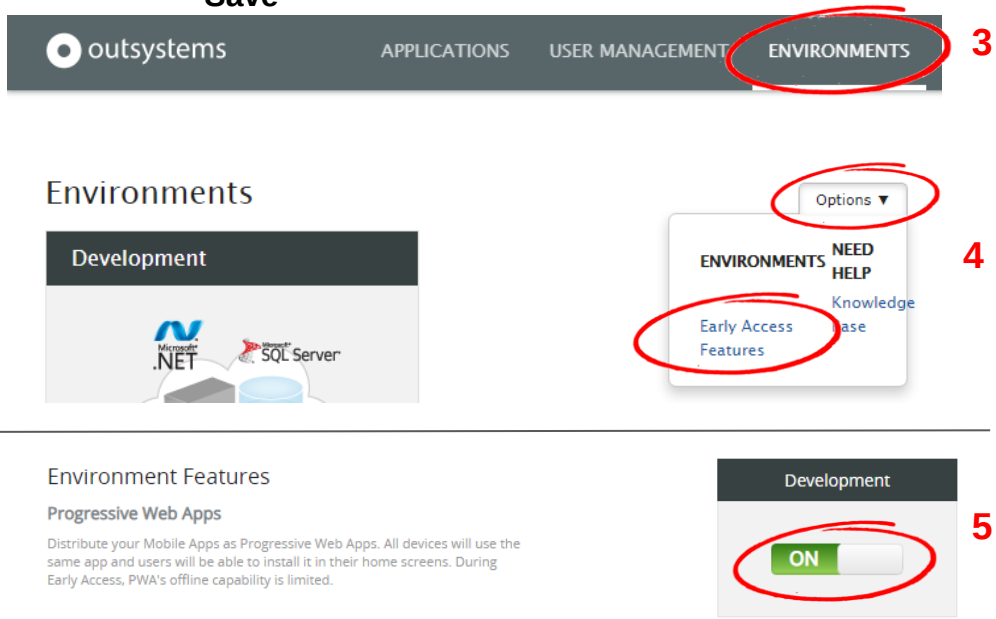

outsystems
Mobile Application Test
Testing for Android & iOS

Ready your environment

Possibly you already did this when following
“0 Jumpstart Prerequisites - Environment Setup.pdf”

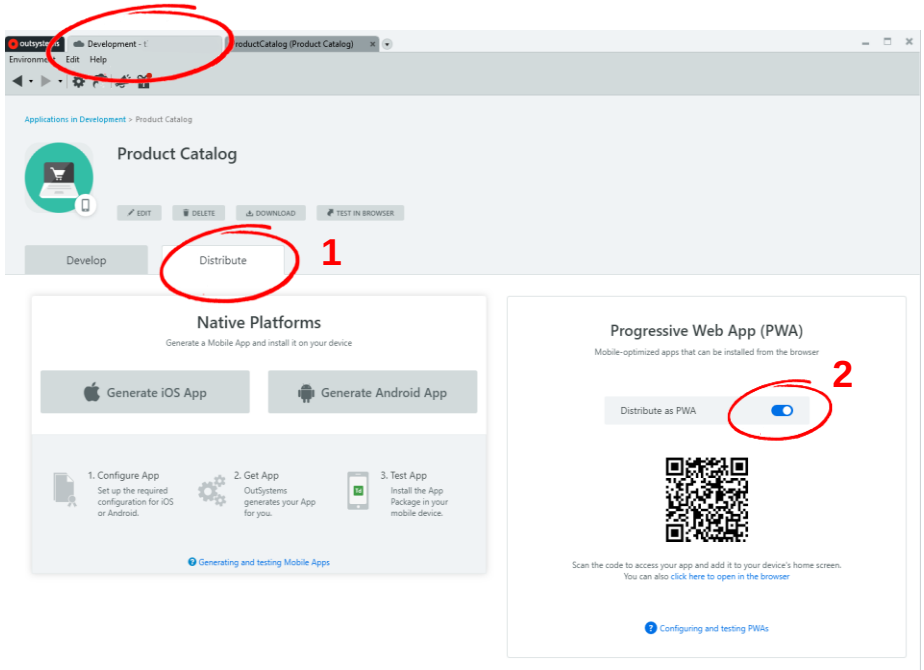
1. Access Lifetime on your environment:
   https://<YOURSERVER>.outsystemscloud.com/Lifetime
2. **Login** using your OutSystems account
3. Navigate to **Environments** in the top menu
4. Under **Options**, select **Early Access Features**
5. Enable **Progressive Web Apps**, and click **Save**





Test it on Your Device

1. Navigate to the **Distribute** tab in Service Studio.
1. Enable **Distribute as PWA**
3. **Scan QR Code** from using your mobile phone
4. Add App to homescreen (See next screen)



3

iOS device?

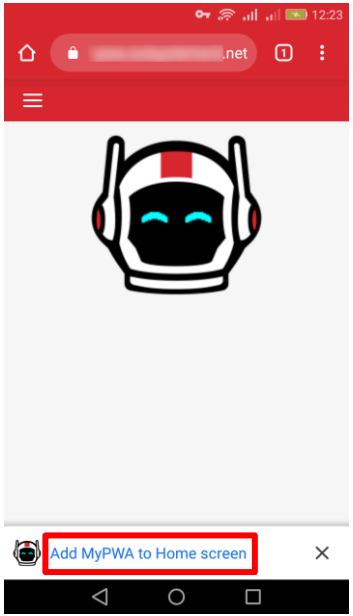
If you're running iOS 13 and later, iOS requires enabling Web SQL for PWAs. Go to **Settings > Safari > Advanced > Experimental Features** and make sure to **switch off "Disable Web SQL"**.

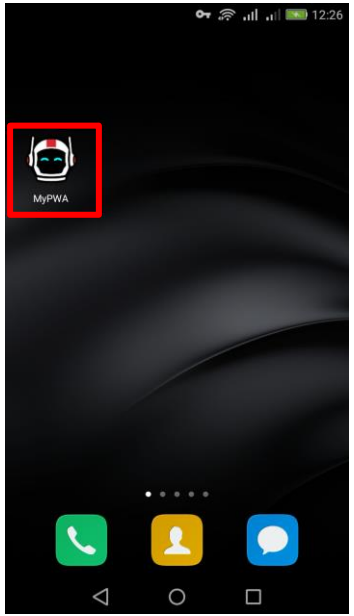
More info

Test it on Your Device

Android

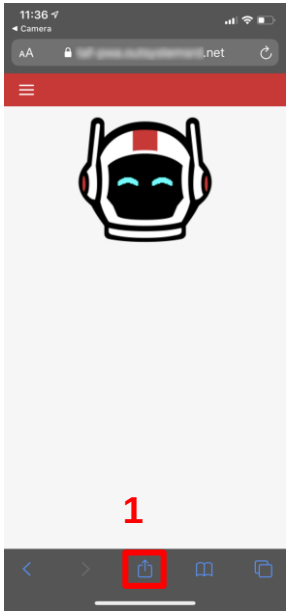
1. Tap the banner Add **<My app>** to Home screen.

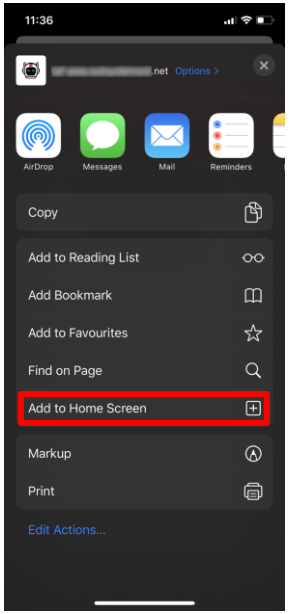




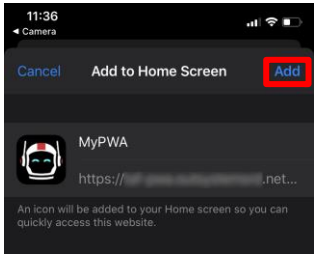
iOS

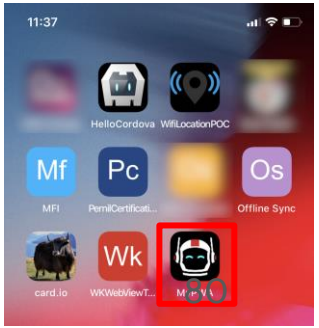
1. Tap the Share button ( )
2. Tap **Add to Home Screen**.
3. In the confirmation screen, tap **Add**.





3





Test your Mobile App

Open and start testing

Test your app and play around. Take a minute and consider: How long would it have taken to build this app in traditional code for both Android and IOS?

Do you want to test the QR codes? You can find them on the next section!


outsystems
Test QR Codes
Start scanning the product QR Codes

Amazon Echo

iOS device?

Below iOS 13.4, due to a bug in iOS, the scanning of barcodes will work best directly in Safari (as opposed to standalone PWA mode)

More info

Philips Hue

FitBit Blaze

iHealth Blood Pressure Monitor

outsystems

Bonus Exercise

Consume REST API

A. Bonus Exercise

Consume Rest API

Bonus exercise:

- If you would like to consume the REST API you have exposed in the previous exercise, you can find the steps in the following video:
  - https://youtu.be/RnOo4Ap7Oto

- This video also contains instruction on how to create a listing screen using other Outsystems accelerators, animations and others.


outsystems
Congratulations
You have successfully built a mobile app!

outsystems

Jumpstart training