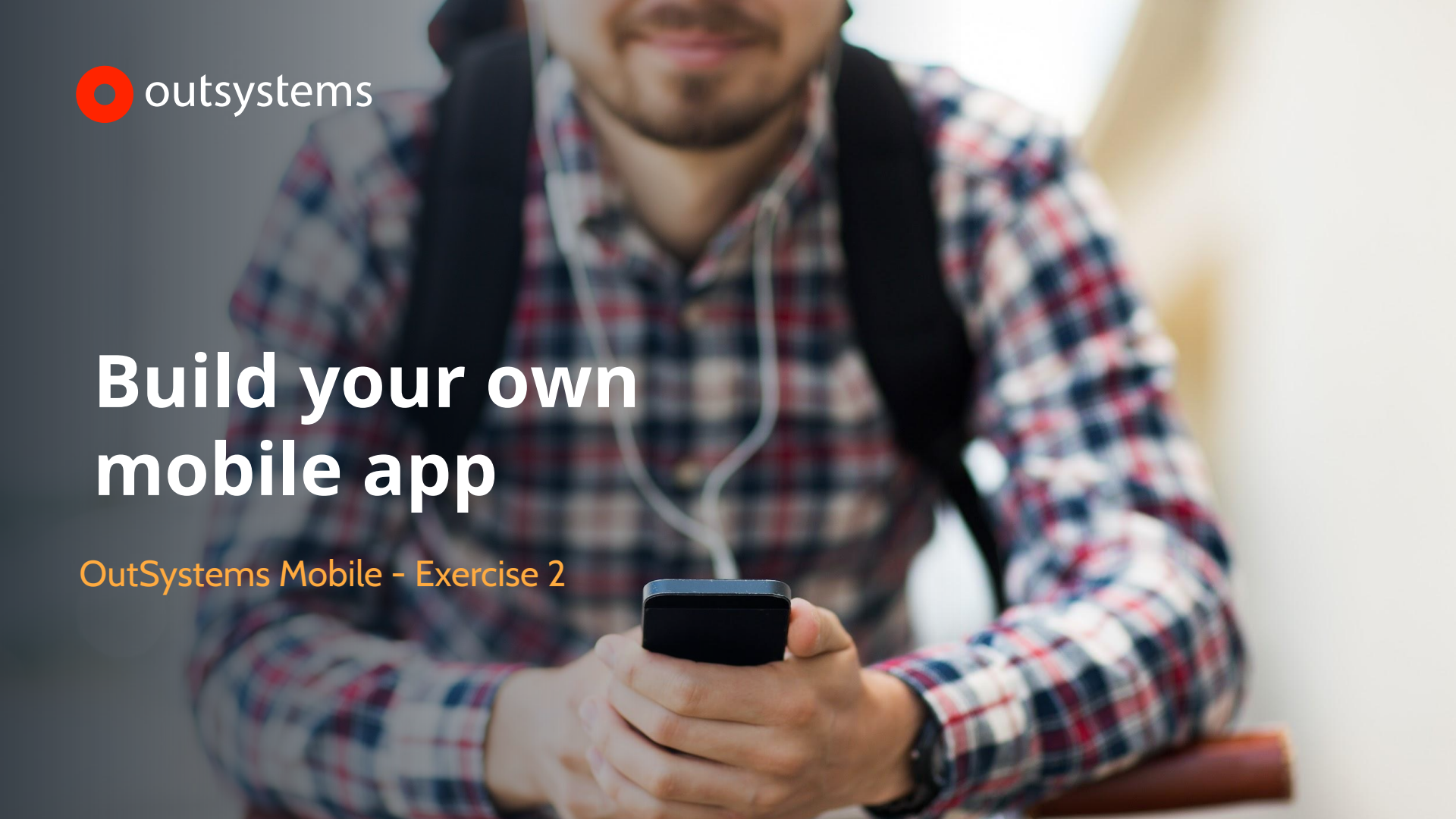


outsystems

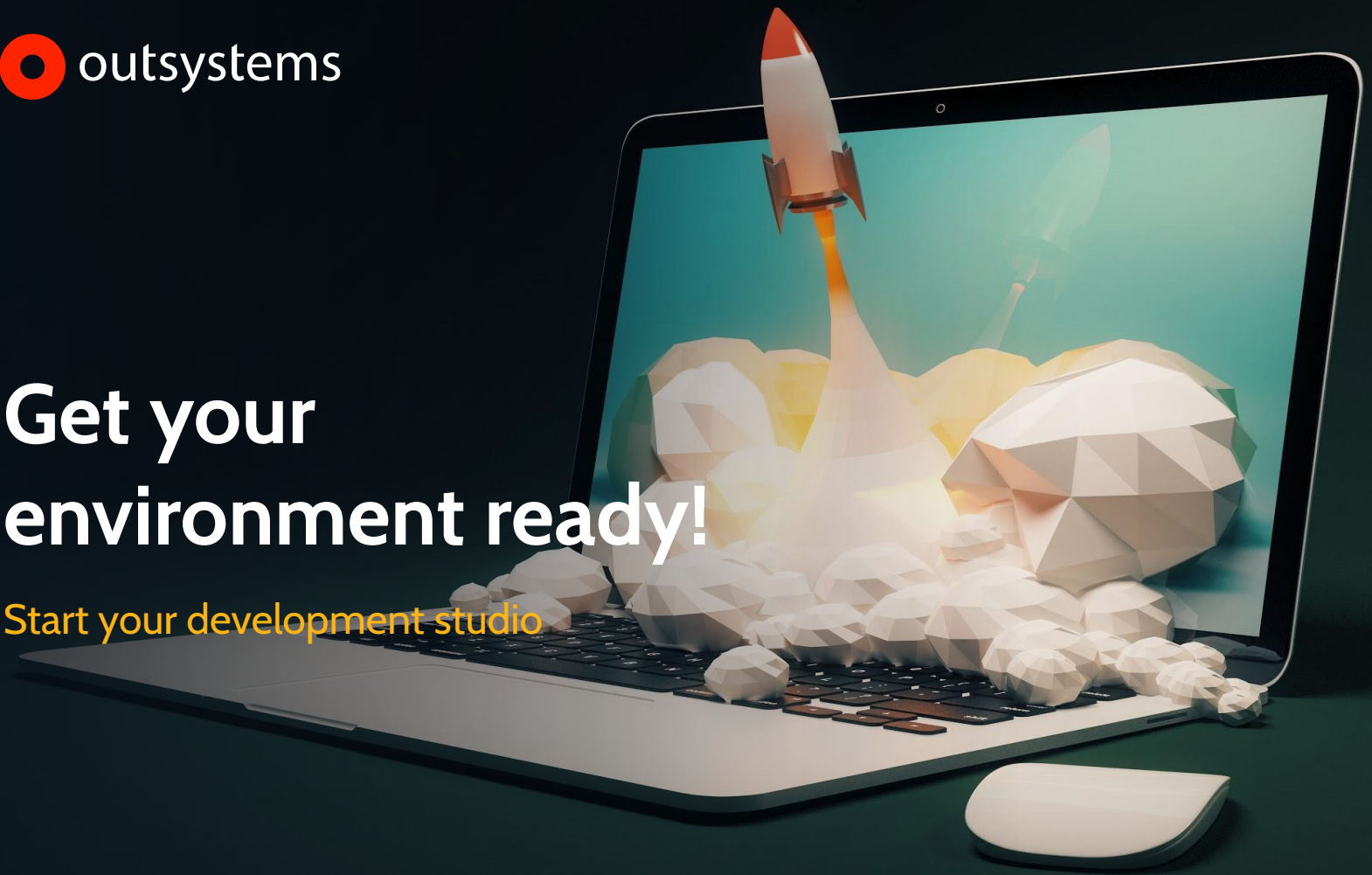
Build your own mobile app

OutSystems Mobile - Exercise 2

outsystems

Get your environment ready!

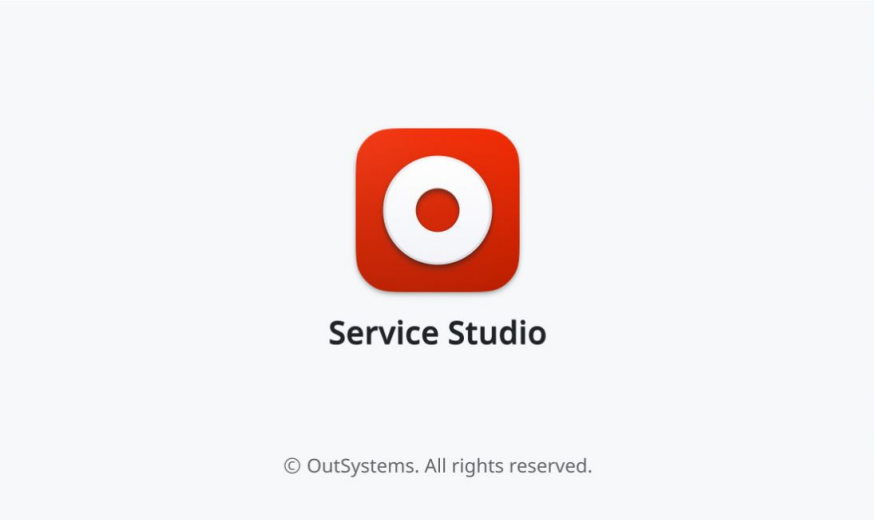
Start your development studio



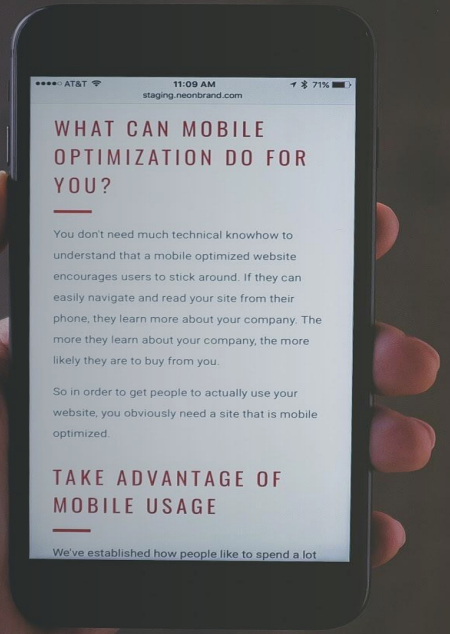
Development Studio

Open and start

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



outsystems



Section 1

Mobile Application Foundation

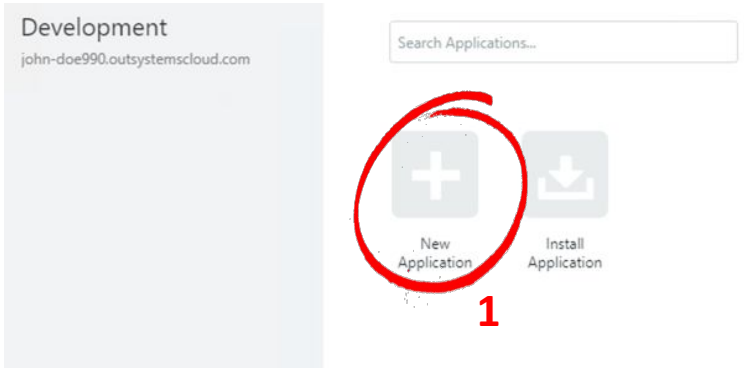
- In this exercise, you will be creating a Mobile Application that lists products hosted in the application demonstrated in the build up demo.

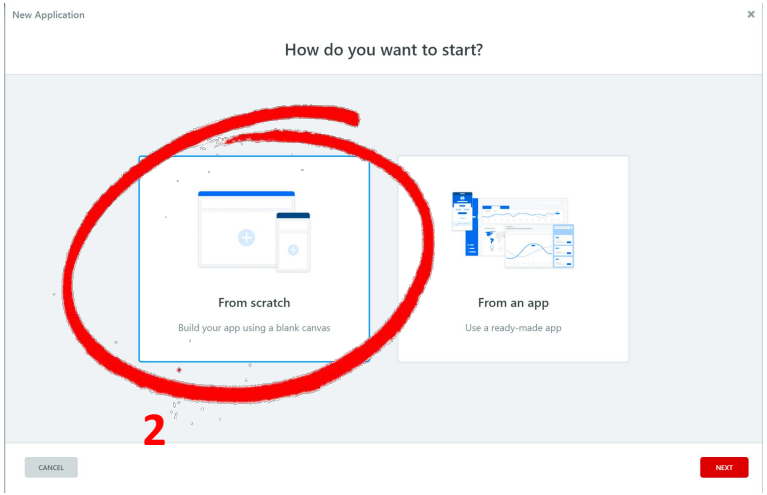
- You will learn the basic differences between web and mobile apps, using a barcode plugin and mobile templates.

- The following Forge Component will be downloaded during this Exercise:
  - Barcode Plugin (Plugin for scanning barcodes and QR codes)

Create a new mobile app

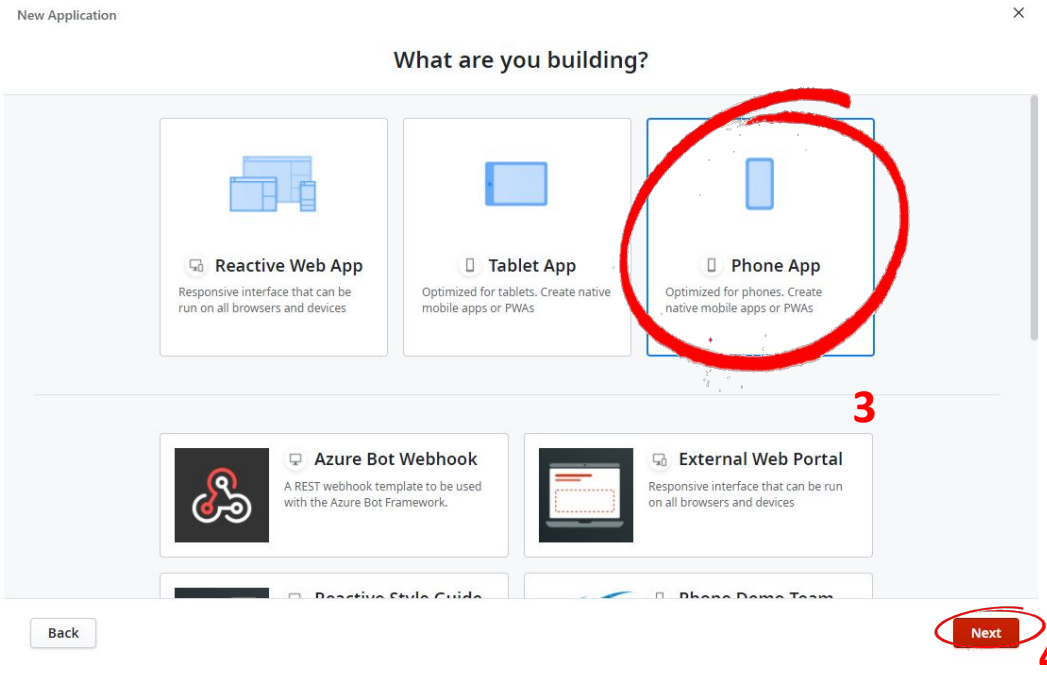
1. Click **“New Application”**
2. Select **“Start from scratch”**





Create a new mobile app

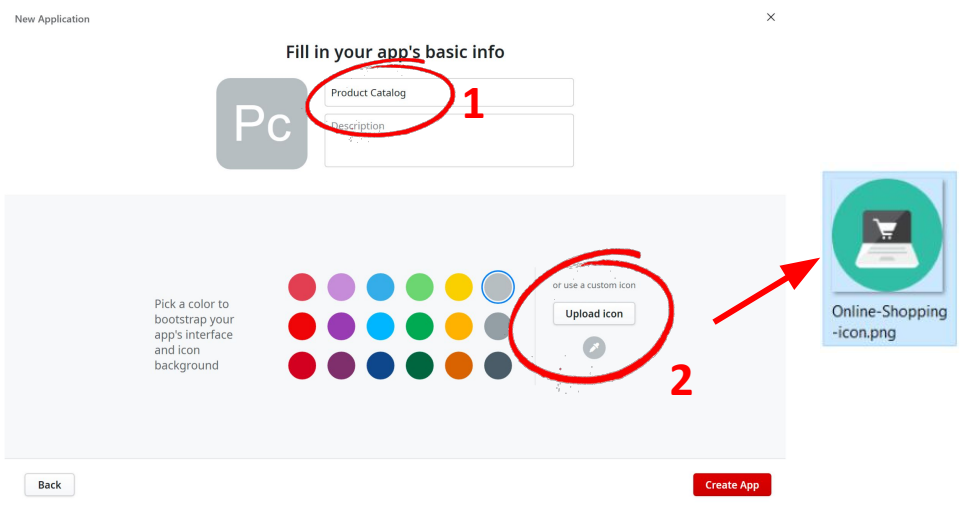
3. Select **Phone App**
4. Click **Next**

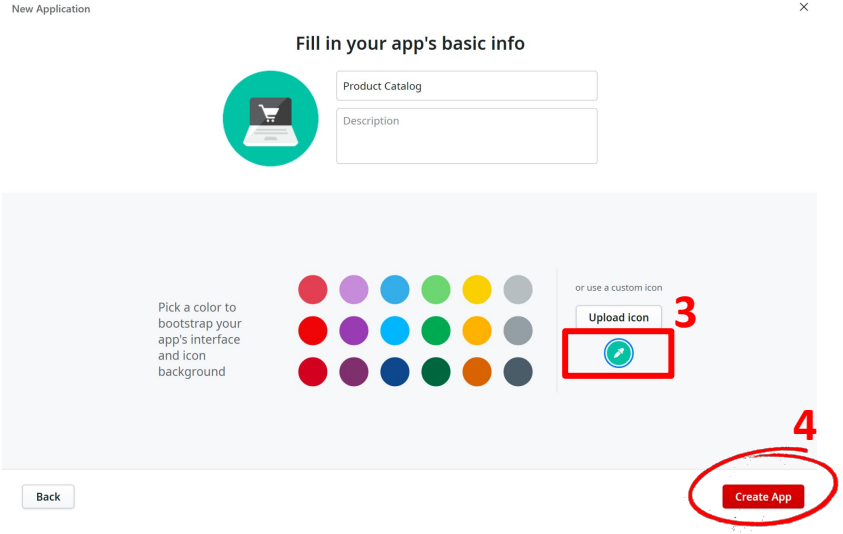


Create a mobile app when you want to create an optimized native experience for mobile devices, with touch friendly behaviors, offline capability and/or leverage the device features such as fingerprint authentication, geo-location, camera, etc.

New Application

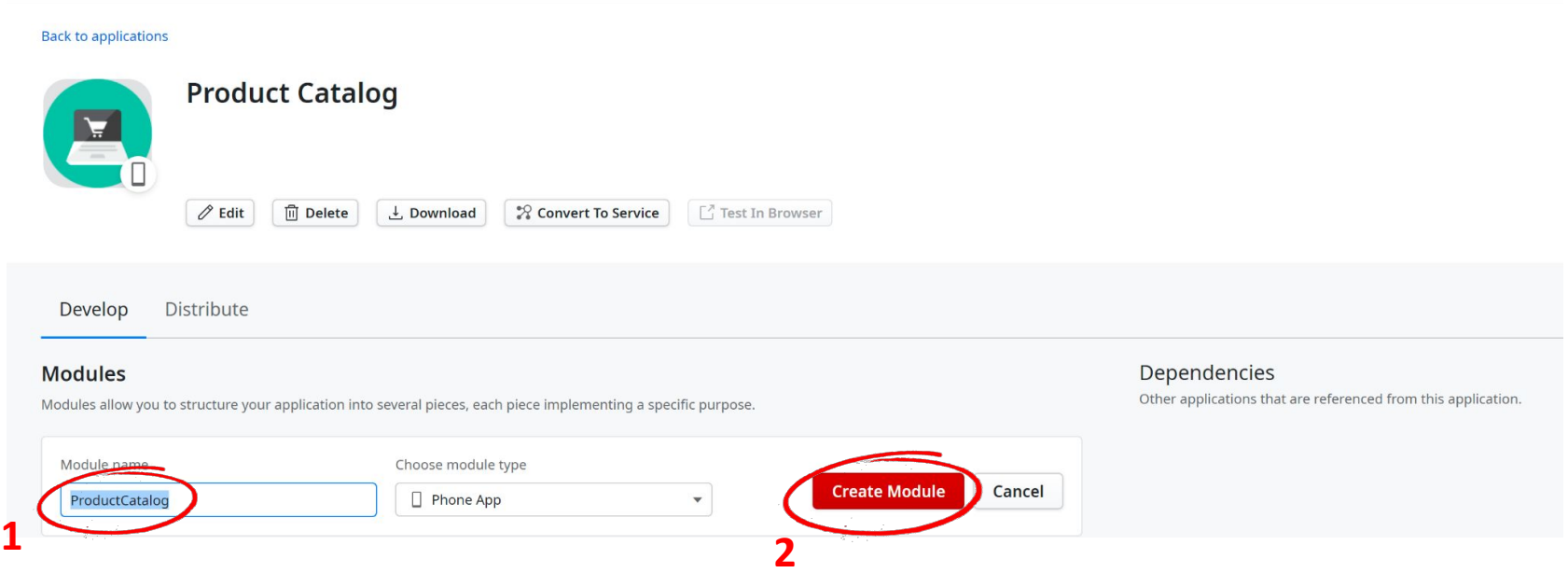
1. Type **Name** of your application “**Product Catalog**”
2. Upload the **icon** provided from Jumpstart Resources Material folder
3. Color palette will be automatically determine based on icon color. Select a **color palette**, if you want to change.
4. Click “Create App”





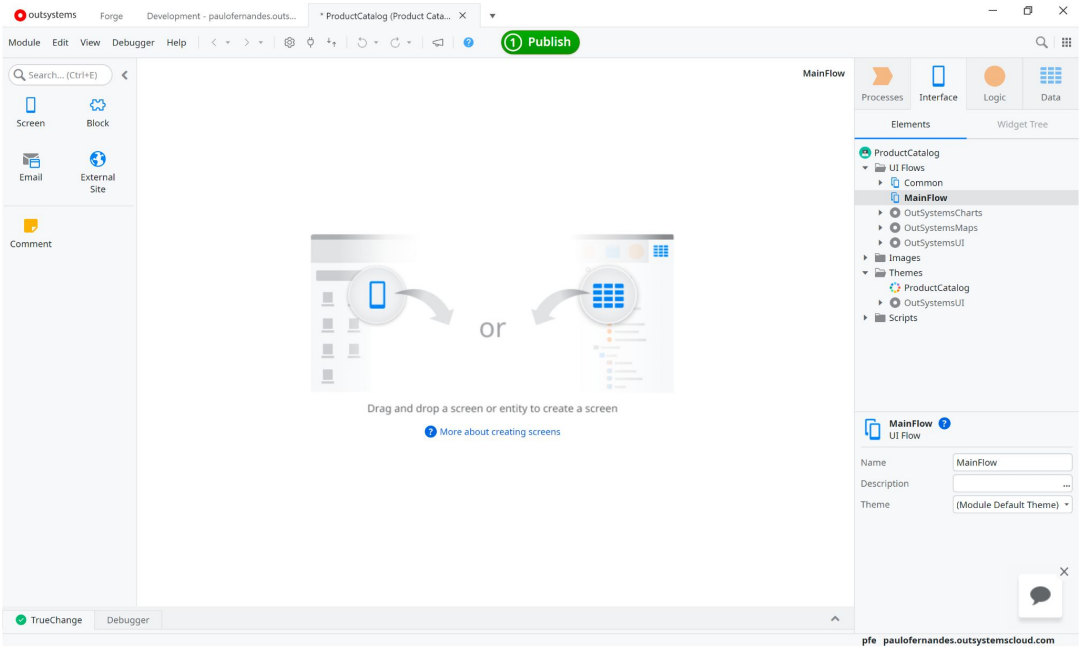
Create Phone App Module

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



Create Phone App Module

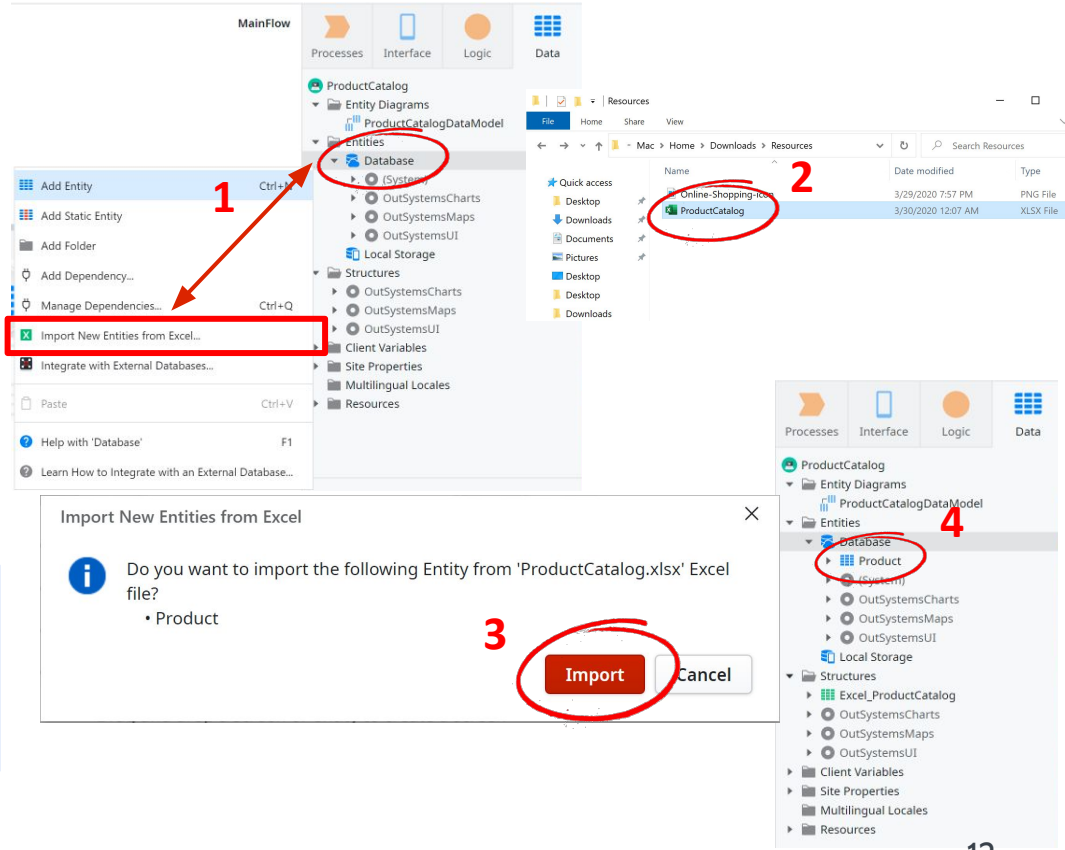
Your mobile app canvas is ready. You will be navigated to your new Mobile module



- Let’s start by creating our data model. In Outsystems tables are known as **Entities** while columns are known as **attributes**.

- In this part, we will create a Product entity, by quickly importing an Excel file. This process is also known as **Bootstrap**.

1. Navigate to the Data tab, right click **Database** and click on **“Import New Entities from Excel”**.
2. Select the **“ProductCatalog.xlsx”**.
3. Click **Import** when prompted for confirmation.
4. You’ll notice that the “Product” entity is automatically created.



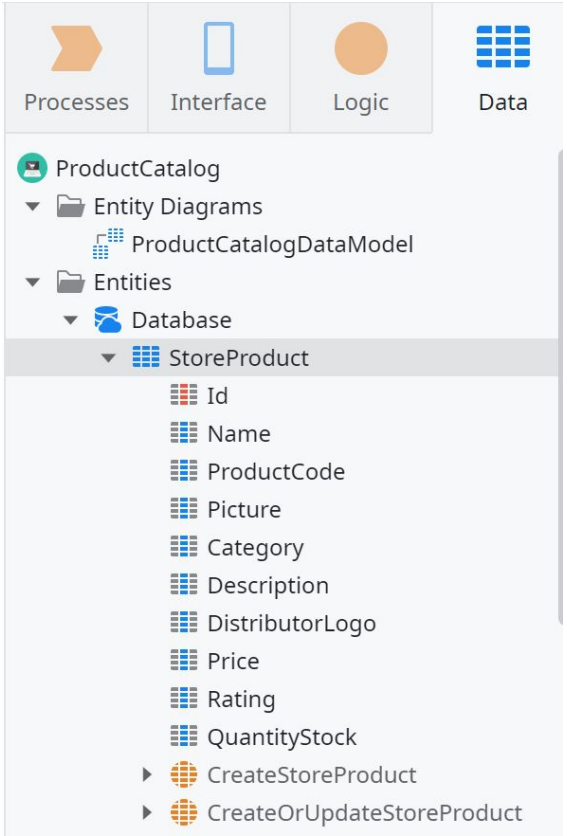
The OutSystems platform provides development **accelerators** to increase productivity on common, patterned and repeating development tasks (**scaffolding**).

Rename Products Entity to StoreProduct

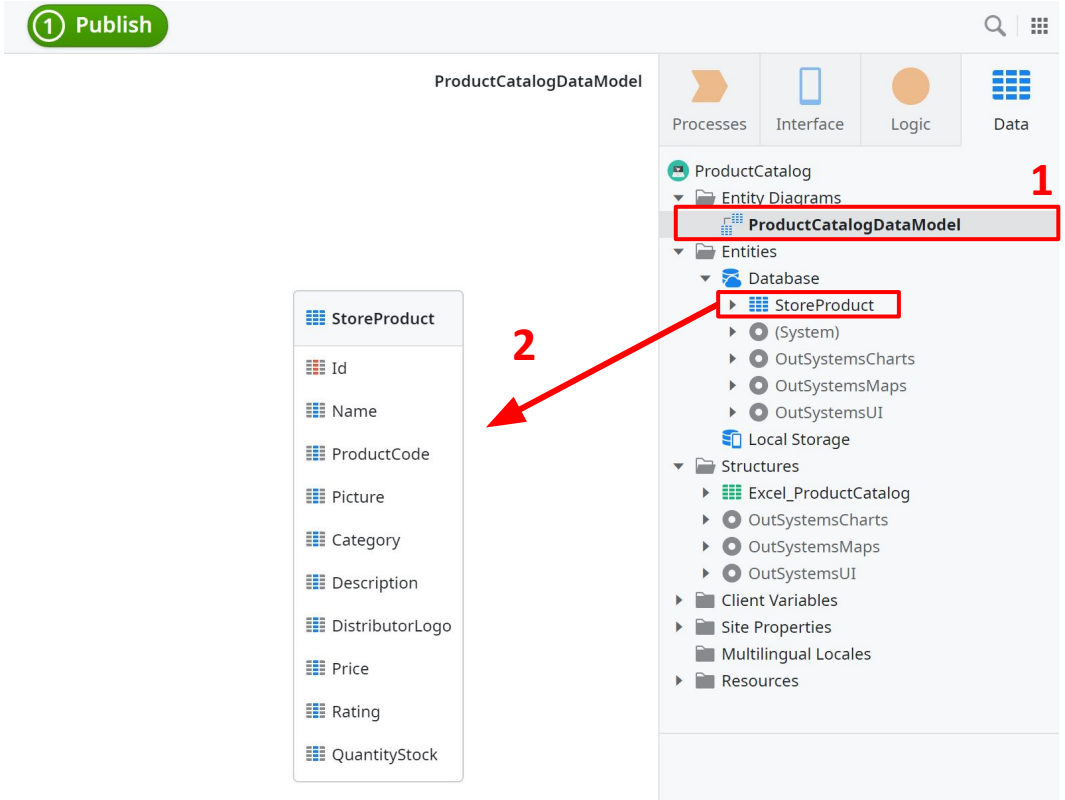
1. Navigate to the **Data** tab

2. Right Click on “Product”, select rename, and name it “StoreProduct”

After importing the data, you can rename entities as desired.

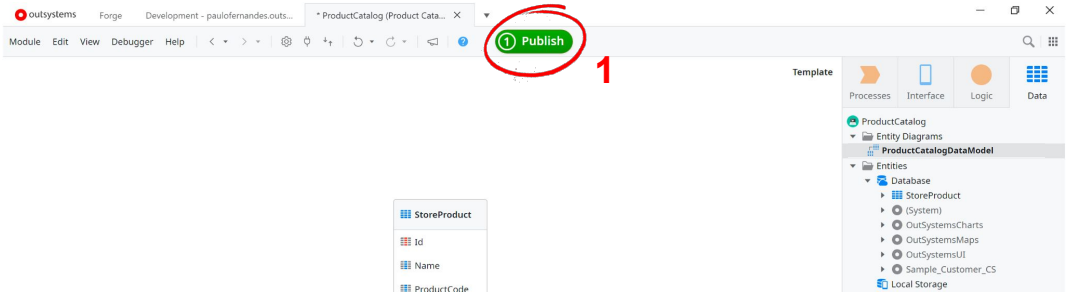


1. Double-click the Entity Diagram “**ProductCatalogDataModel**” to open it
2. Drag the “**StoreProduct**” entity into the canvas.



Entity Diagrams are a powerful way of quickly visualizing your data entities and their relationship

1. Press the **1-Click-Publish** (1) button to publish your application

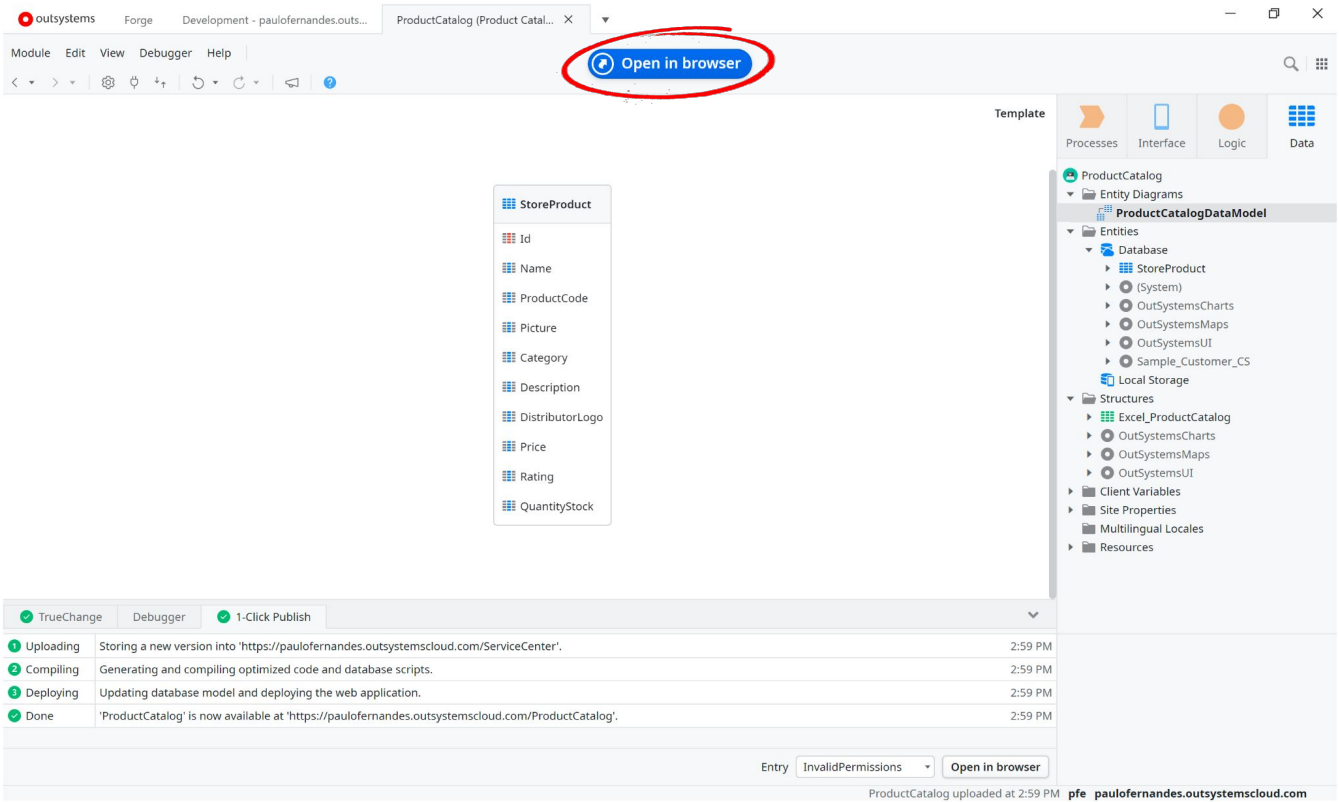


With a single click, the 1-Click-Publish button triggers the following steps:

1. Upload of the visual model to the environment, and versioning
2. Compilation and generation of optimized code
3. Deploy in the database and web server

TrueChange	Debugger	1-Click Publish
1 Uploading	Storing a new version into 'https://paulofernandes.outsystemscloud.com/ServiceCenter'.	
2 Compiling	Generating and compiling optimized code and database scripts.	
3 Deploying	Updating database model and deploying the web application.	
Done	'ProductCatalog' is now available at 'https://paulofernandes.outsystemscloud.com/ProductCatalog'.	

End of Section 1 - Mobile Application Foundation



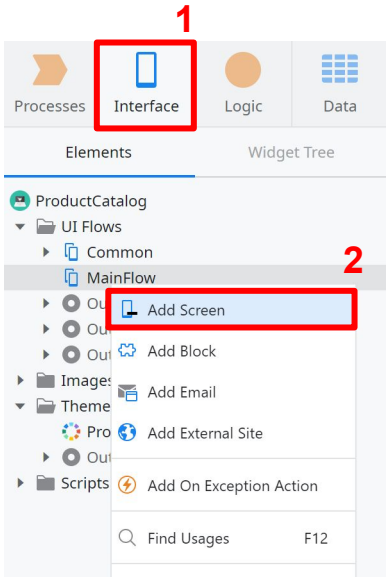
Section 2

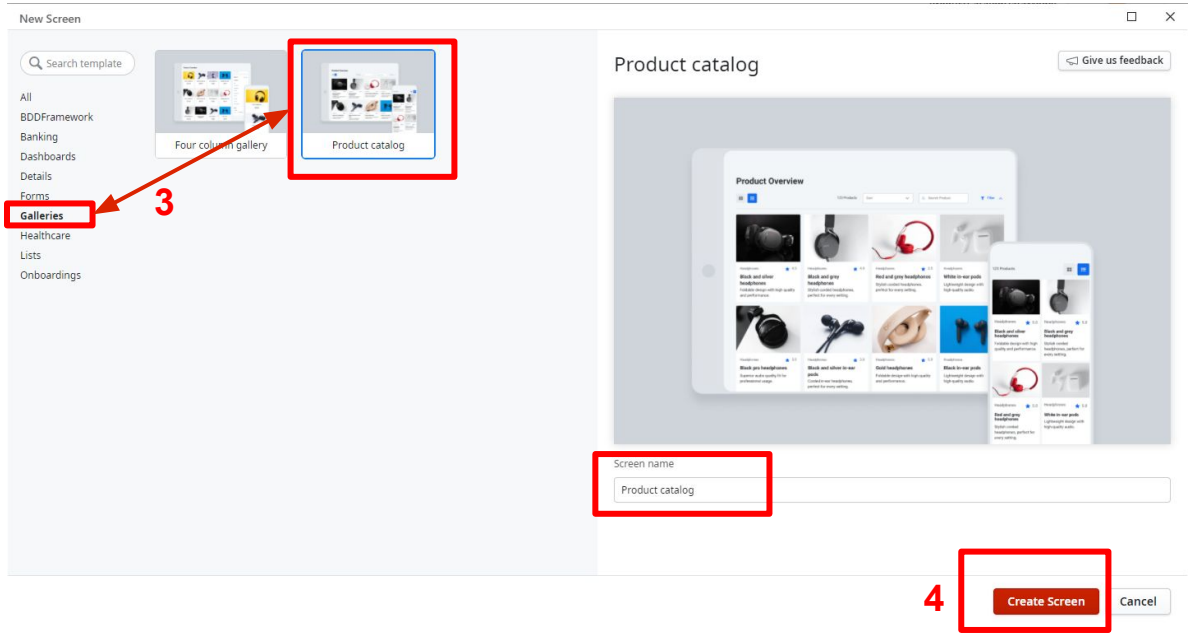
Product Gallery

- 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





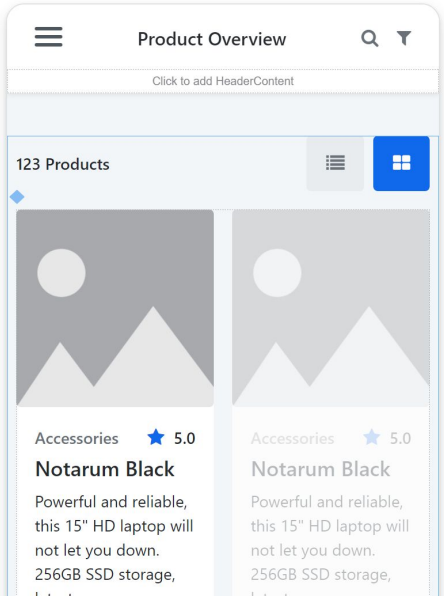
1. Go to the interface tab
2. Right-click on **MainFlow** and **Add Screen**
3. Go to “Galleries” and select the **Product catalog** Template
4. Keep the screen name as “**Product catalog”**, and click **Create Screen**

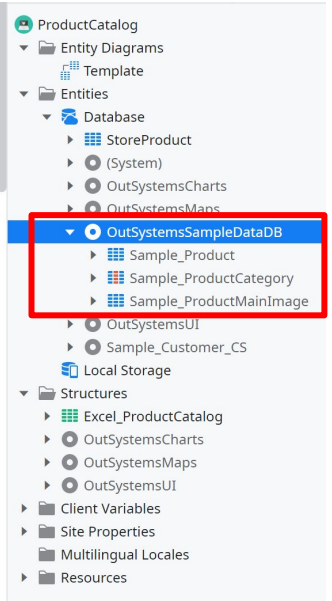
Sample data was automatically added

A screen with pre-built template was automatically created.

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

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





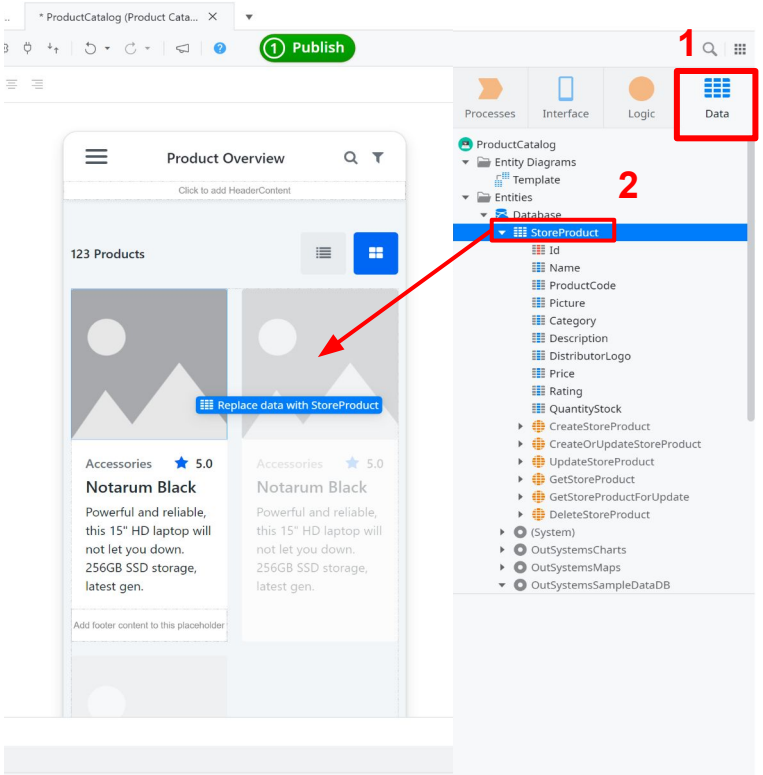
- The Outsystems platform provides screen templates to help developers create user interfaces easily. These Templates contain **UI design**, include **pre-created actions** to accelerate development of the screen actions (such as filters) and **pre-created sample data** so you don't have to start from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

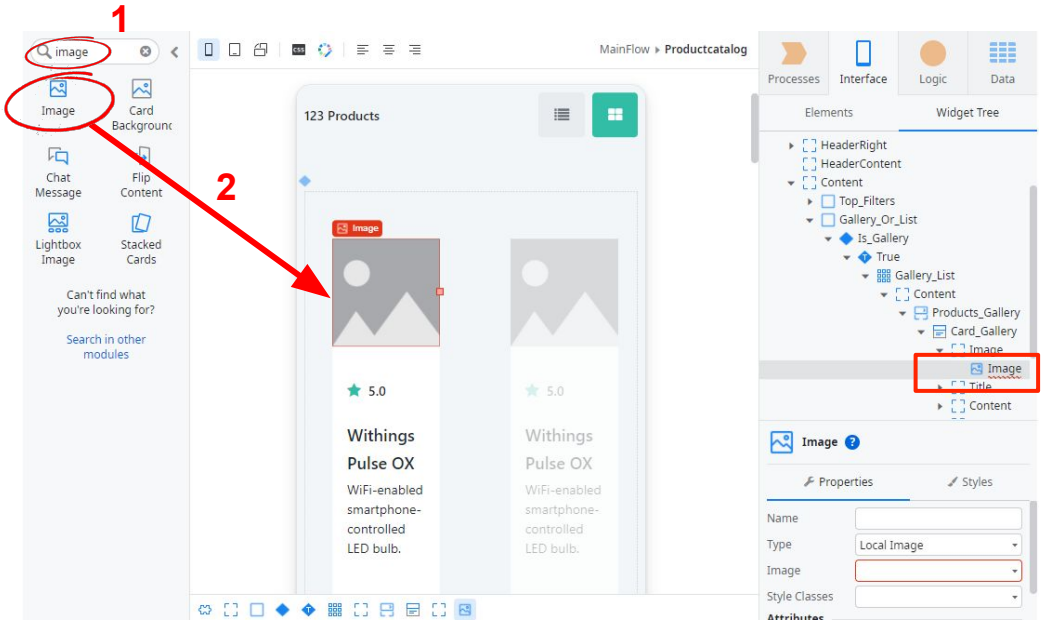
Use product entity data

1. Navigate to the **Data** tab

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



1. Go to the toolbox search bar (top left) and search for “Image”.
2. Then, drag the Image widget and drop it on the “Image” placeholder as illustrated. Make sure the indicator is placed under the Image Placeholder of the widget tree.



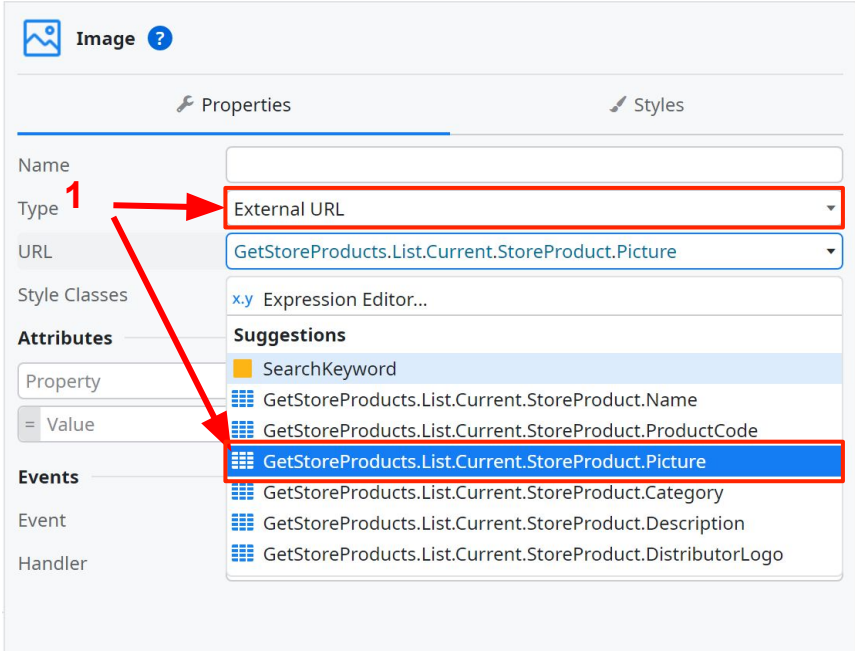
QUICK TIP

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

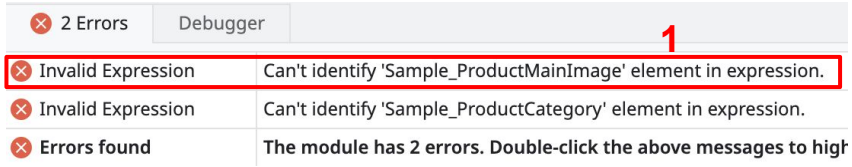
It can be manually opened by clicking the right tab “Widget Tree”

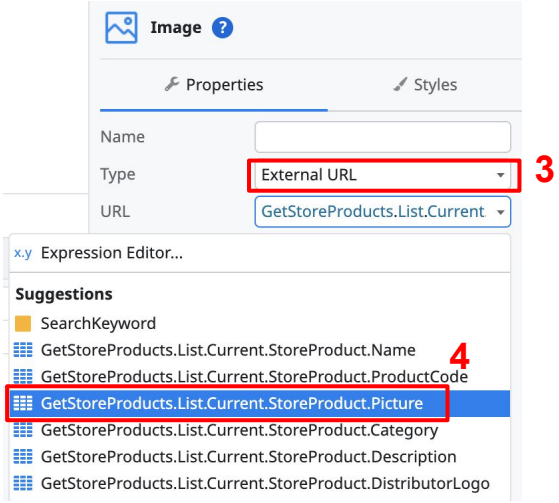
It also **opens automatically** when you drag a widget onto the screen!

1. On the Image Properties tab set the following attributes:
   - **Type**: **External URL**
   - **URL**: Select the suggested **GetStoreProducts.List.Current.StoreProduct.Picture**



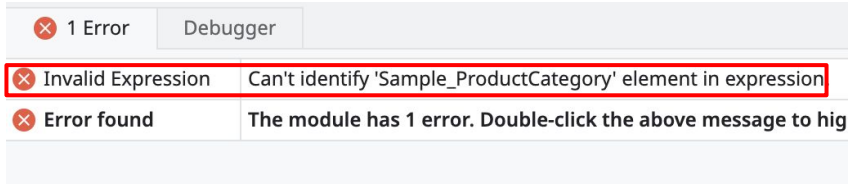
Can't identify ‘Sample_ProductMainImage’ element in expression.





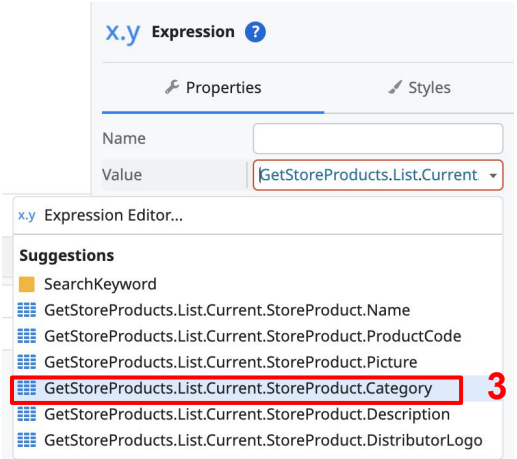
1. Double click on the error “*Can’t identify 'Sample_ProductMainImage' element in expression*”;
2. Close the pop-up dialog;
3. Navigate to the Properties on the bottom right of the screen and change the property “Type” from **Binary Data** to **External URL**;
4. Change the property URL with the value **GetStoreProducts.List.Current.StoreProduct.Picture**.

Can't identify ‘Sample_ProductCategory’ element in expression.



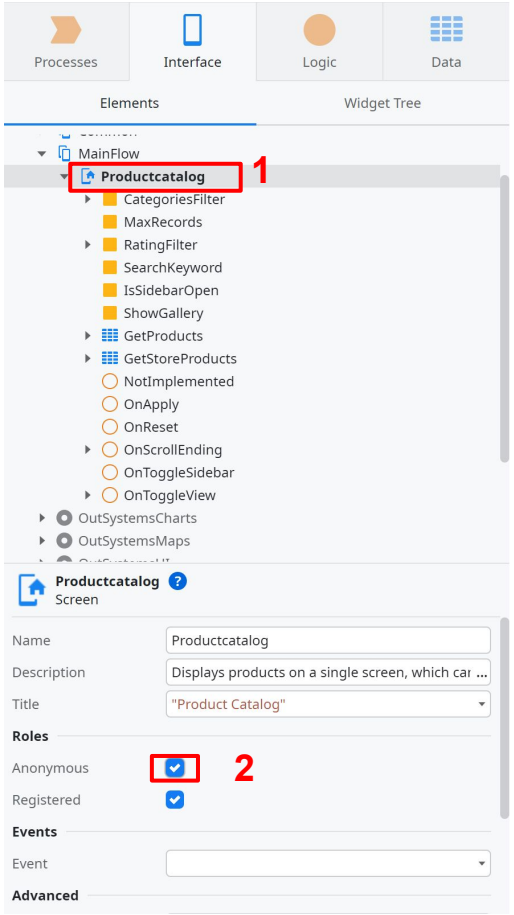
1. Double click on the error “*Can’t identify 'Sample_ProductCategory' element in expression.*” and close the pop-up dialog;

2. Navigate to the Properties on the bottom right of the screen and change the property “Value” to **GetStoreProducts.List.Current.StoreProduct.Category**.



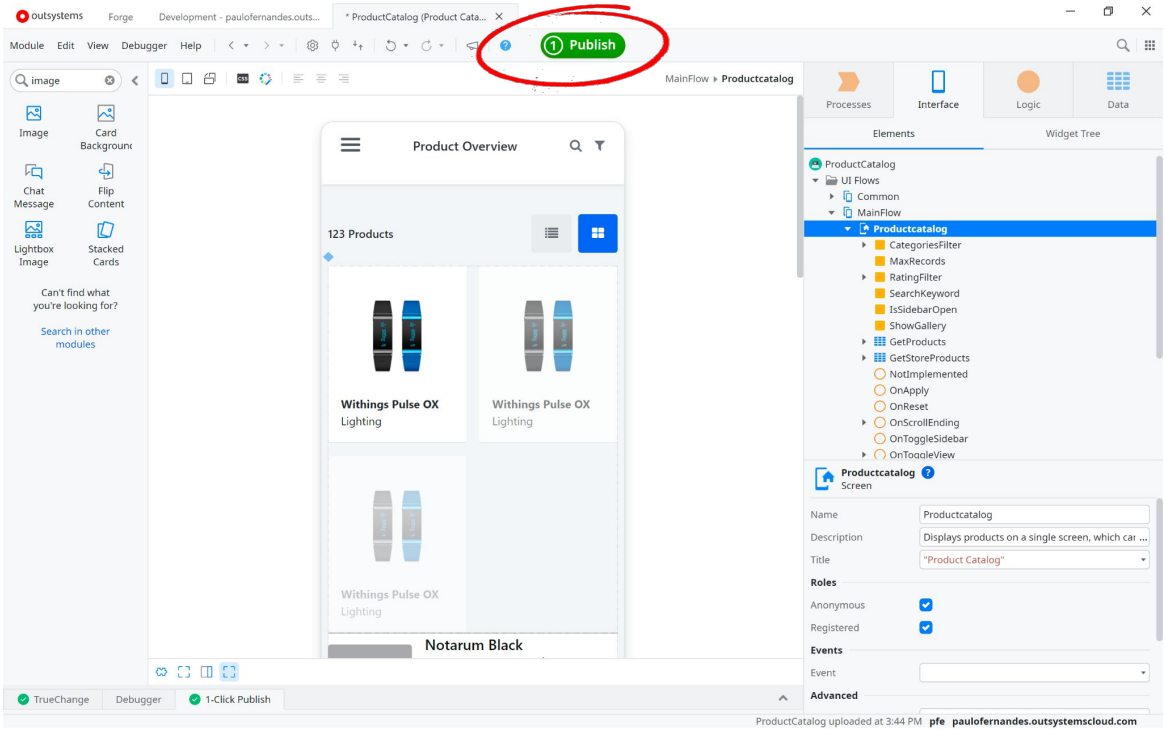
1. Click on the “Productcatalog” screen;

2. Mark the **Product Catalog** screen to be accessible by **Anonymous** users.



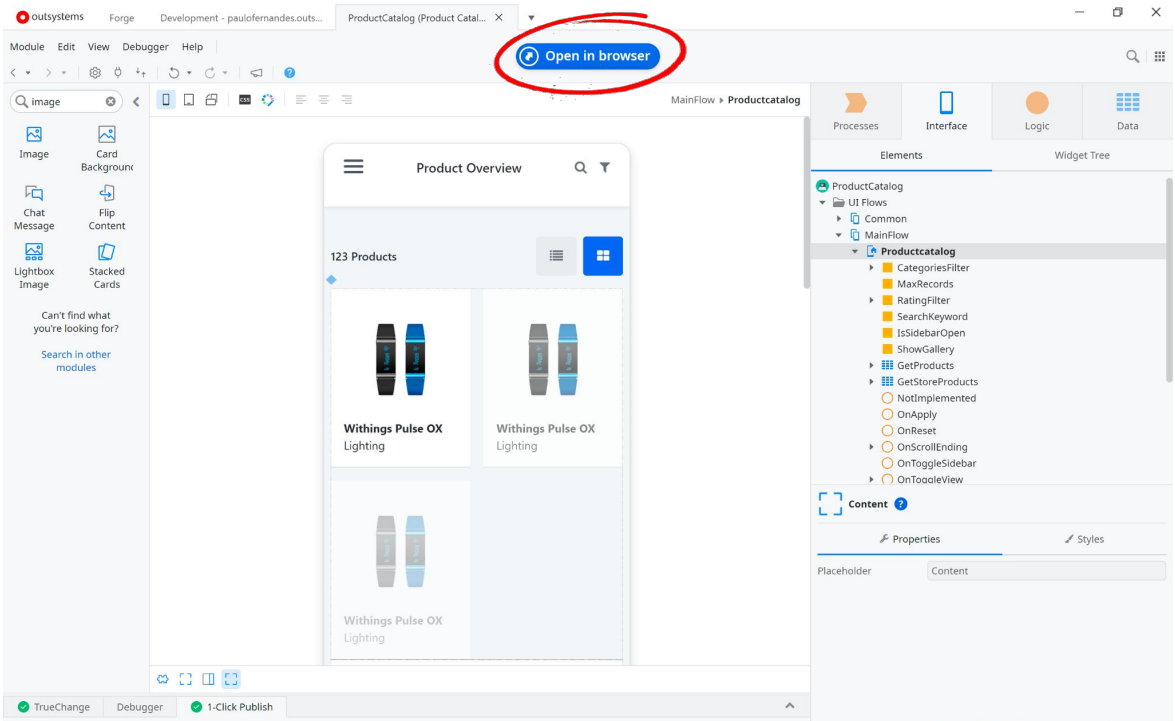
Deploy and Test it

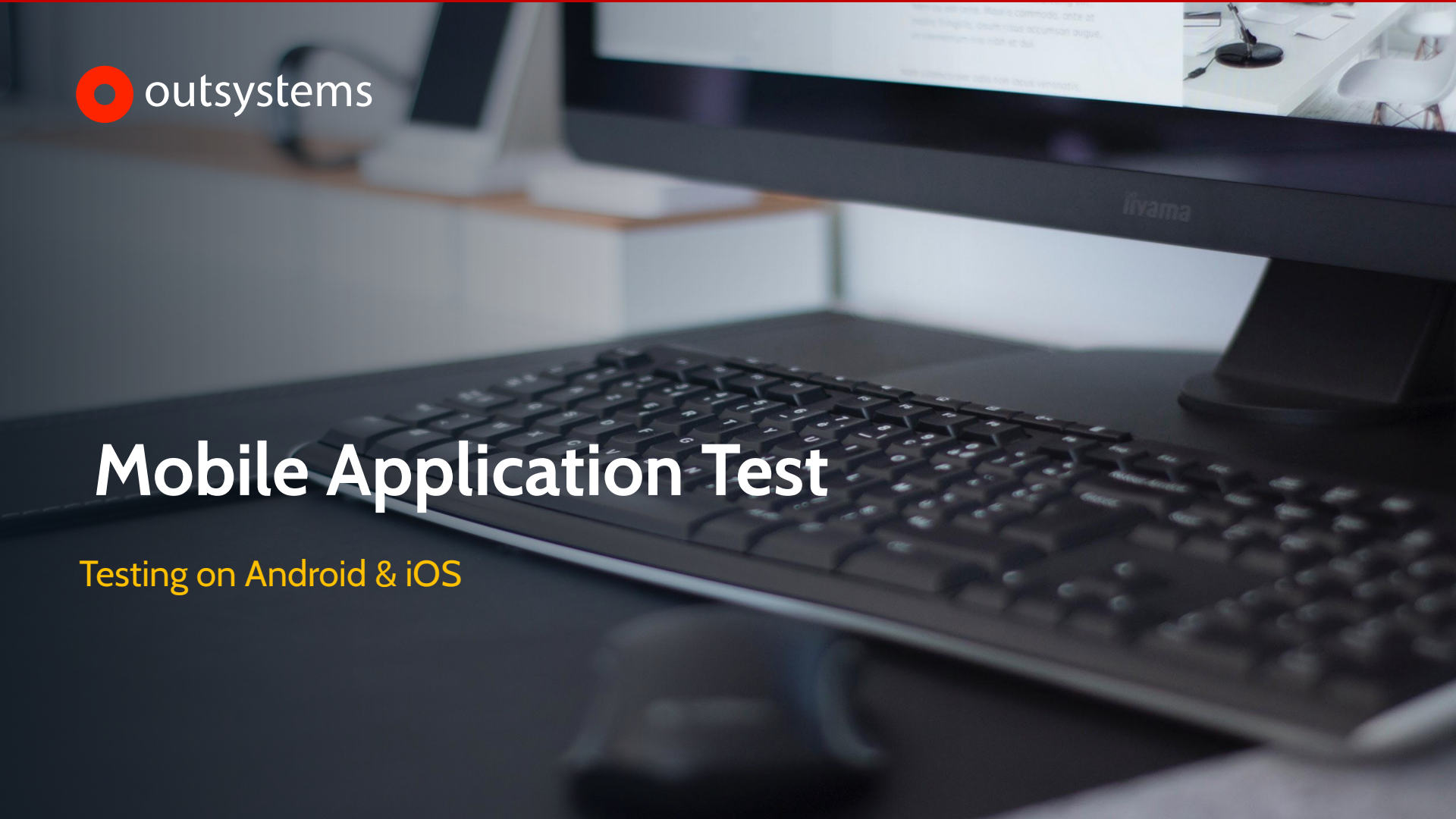
1. At this point you can deploy your application by clicking the 1-click-publish button (1 Publish). This will generate the mobile app.



Deploy and Test it

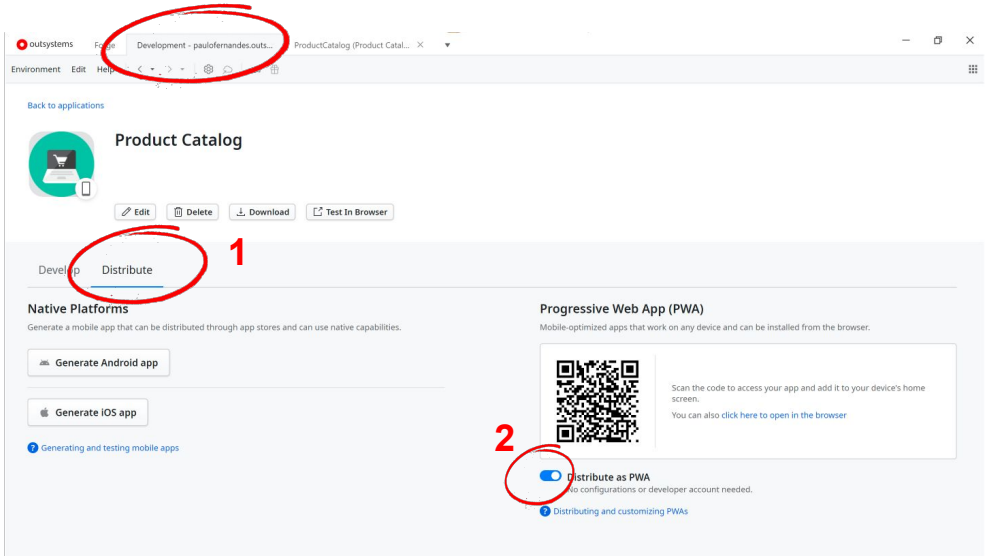
1. Open it in the **simulator** (Open in browser) and test your app.




outsystems
Mobile Application Test
Testing on Android & iOS

Test it on Your Device

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

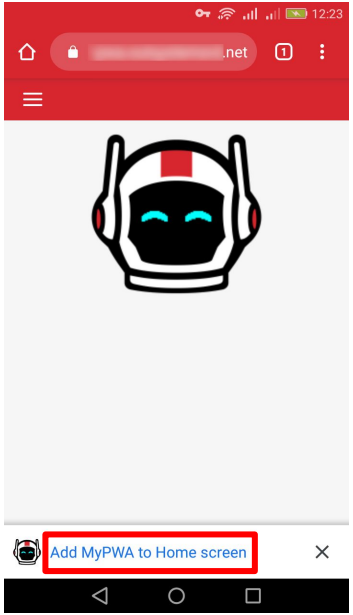


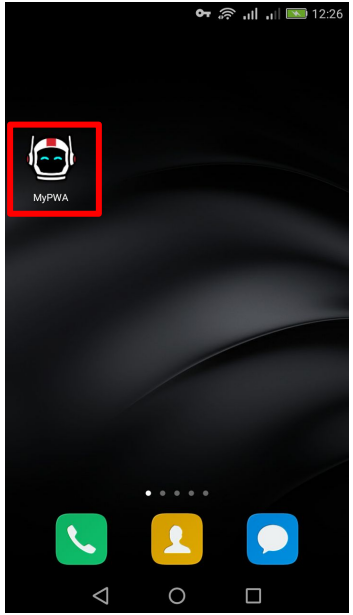
3

Test it on Your Device

Android

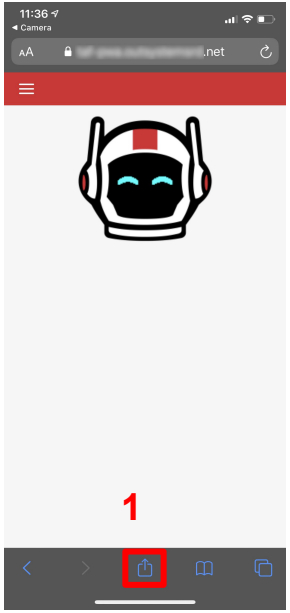
1. Tap the banner Add **<My app>** to Home screen.

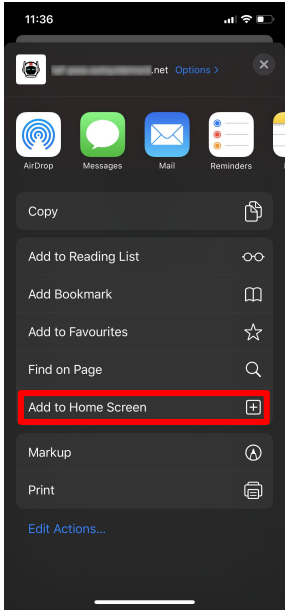




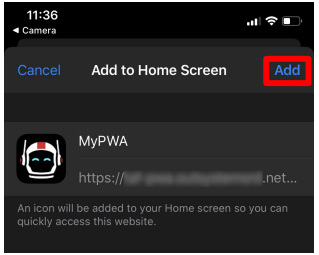
iOS

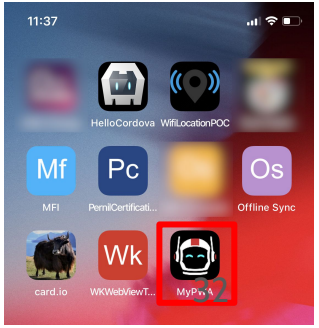
1. Tap the Share button ( )
2. Tap **Add to Home Screen**.
3. In the confirmation screen, tap **Add**.





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?

outsystems

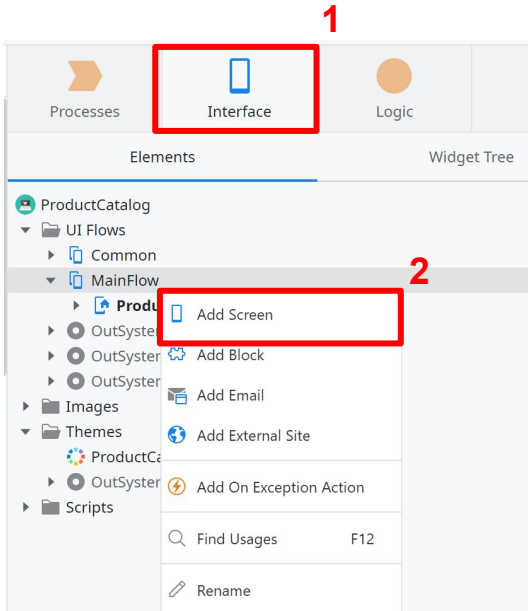
Section 3

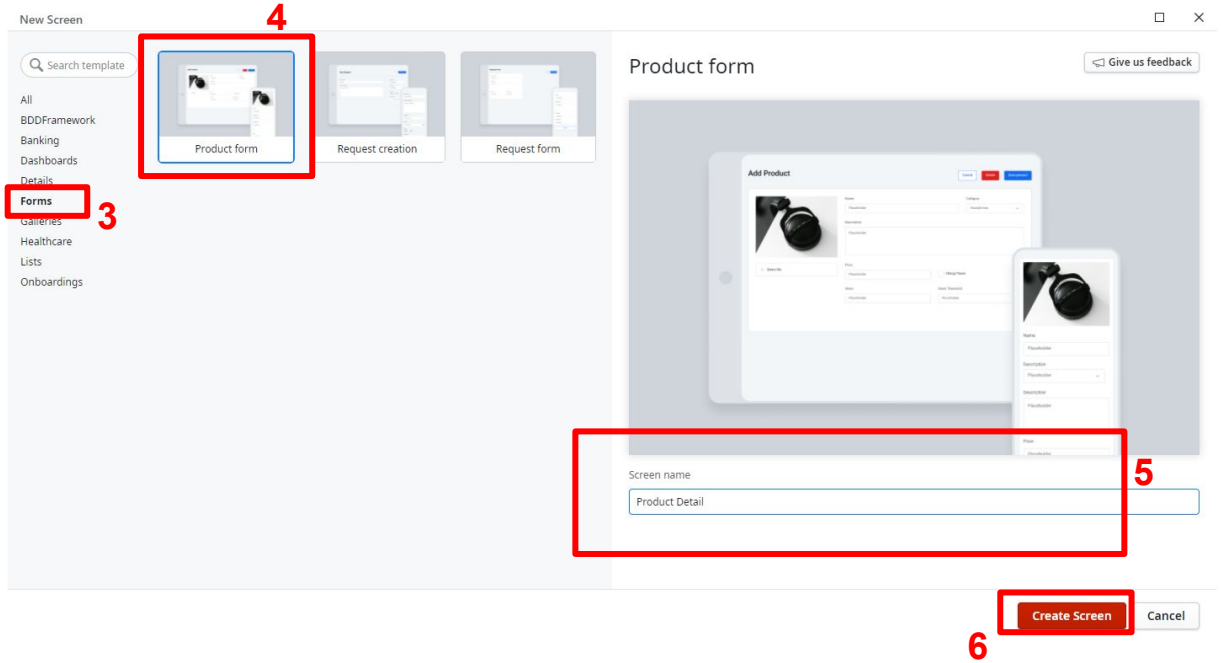
Product Detail

- In the previous section we have created a Product Gallery page and replaced the sample data with the Product Entity data.

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

Product Detail

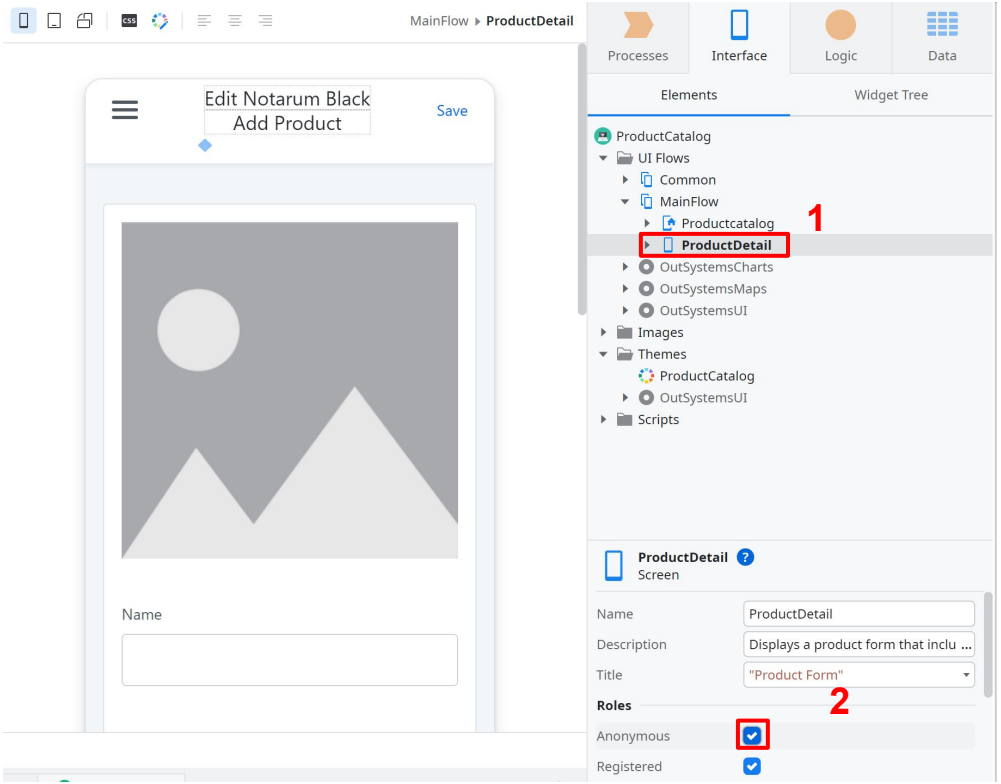




1. Go to the interface tab;
2. Right-click on **MainFlow** and **Add Screen**;
3, 4. Select **Forms > Product form**;
5. Name your screen as **Product Detail**;
6. Click on **Create Screen**.

Set it as anonymous

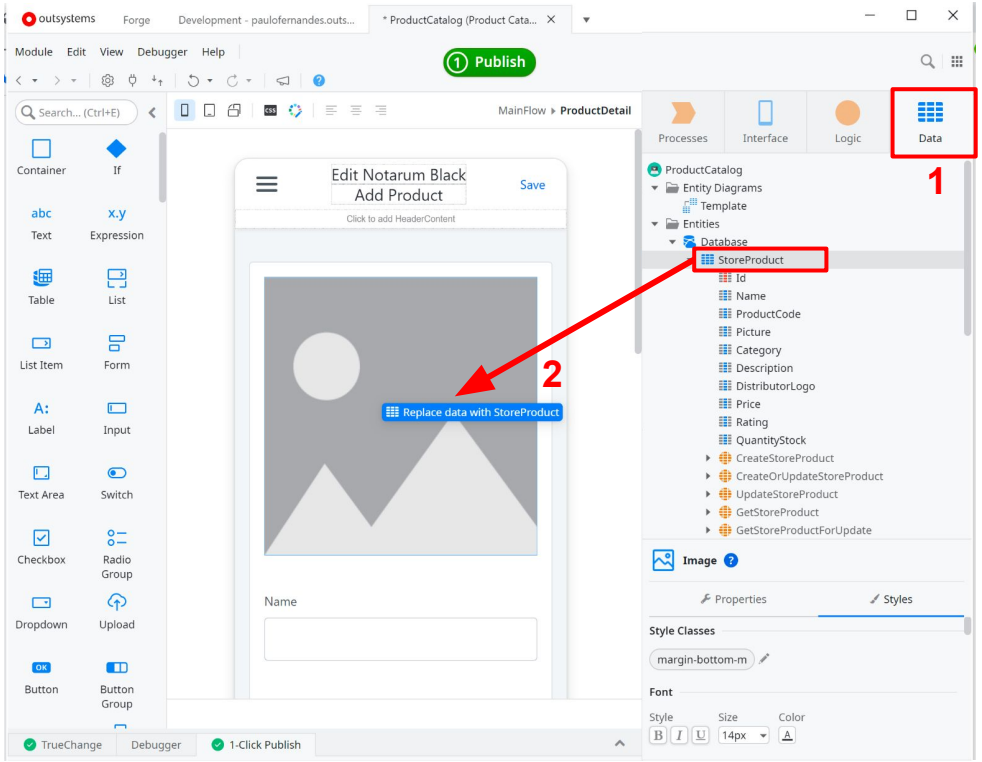
1. Select the “ProductDetail” screen from the MainFlow
2. From the screen properties, Check the **Anonymous Role**



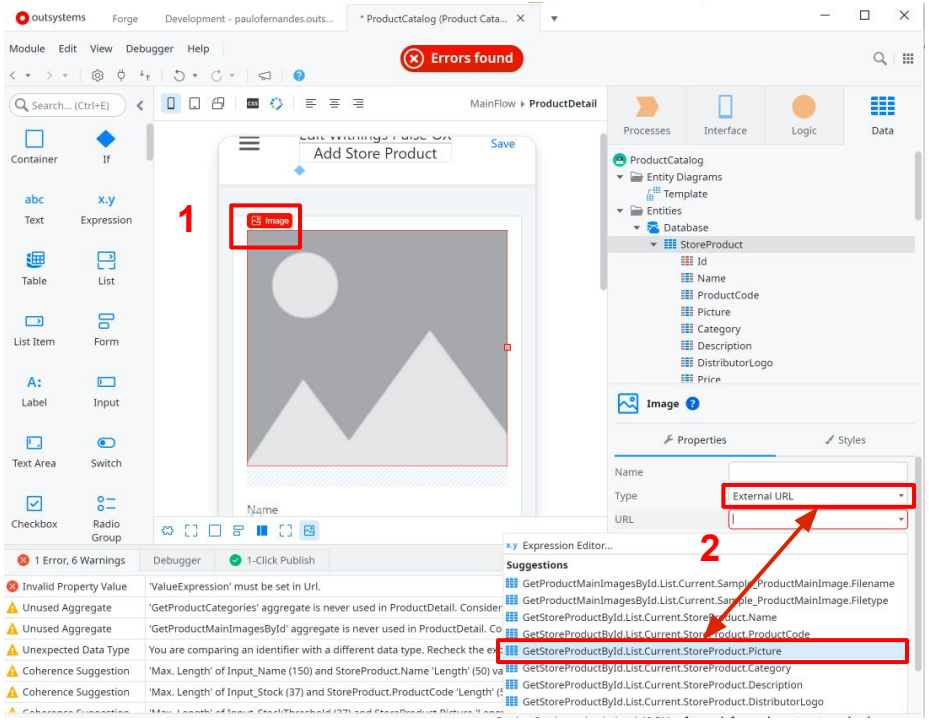
Use the Product Entity data

1. Go to the **Data tab**

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



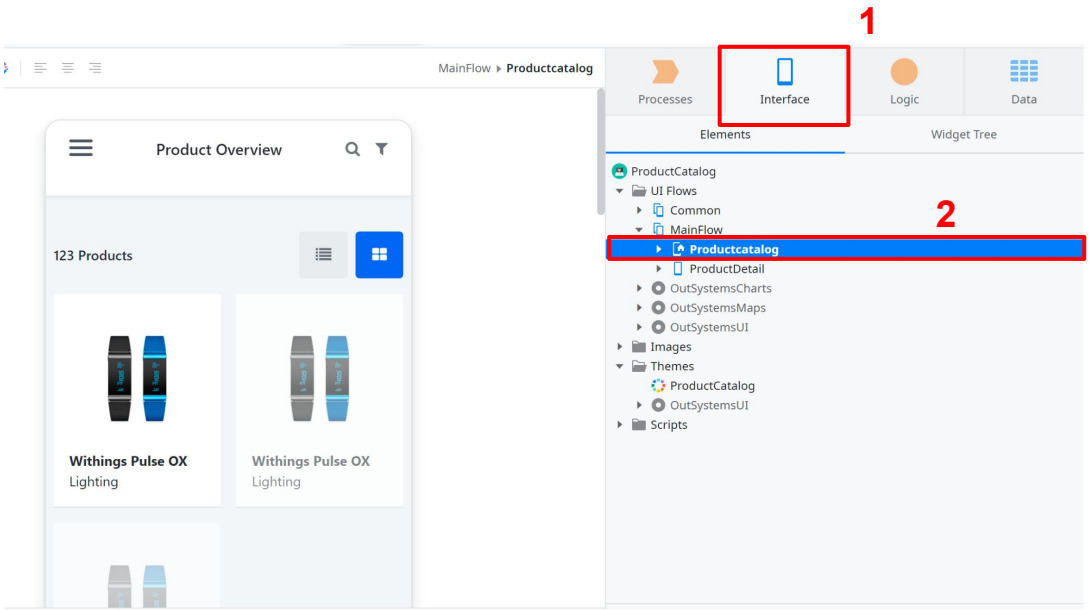
1. Select the “Image” widget of the ProductDetail screen;
2. On the Image Properties tab set the following attributes:
   - **Type**: External URL;
   - **URL**: Select the suggested **GetStoreProductById.List.Current.StoreProduct.Picture**.

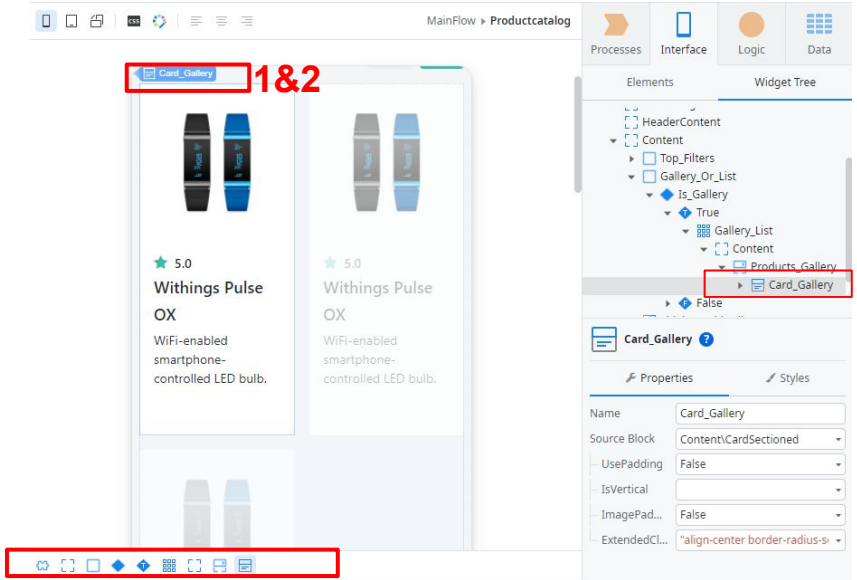


- Now that we have replaced the Product Detail screen with the Product Entity data, we want to enable users to choose a product from the Product Gallery and see its details.

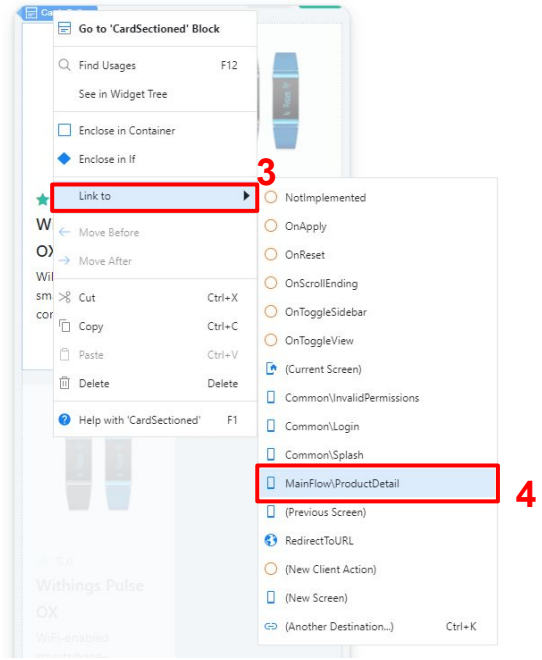
Open the screen

1. Go to the **Interface** tab
2. Double click the **Product Catalog** screen





1. Select the **Card_Gallery** widget;



QUICK TIP

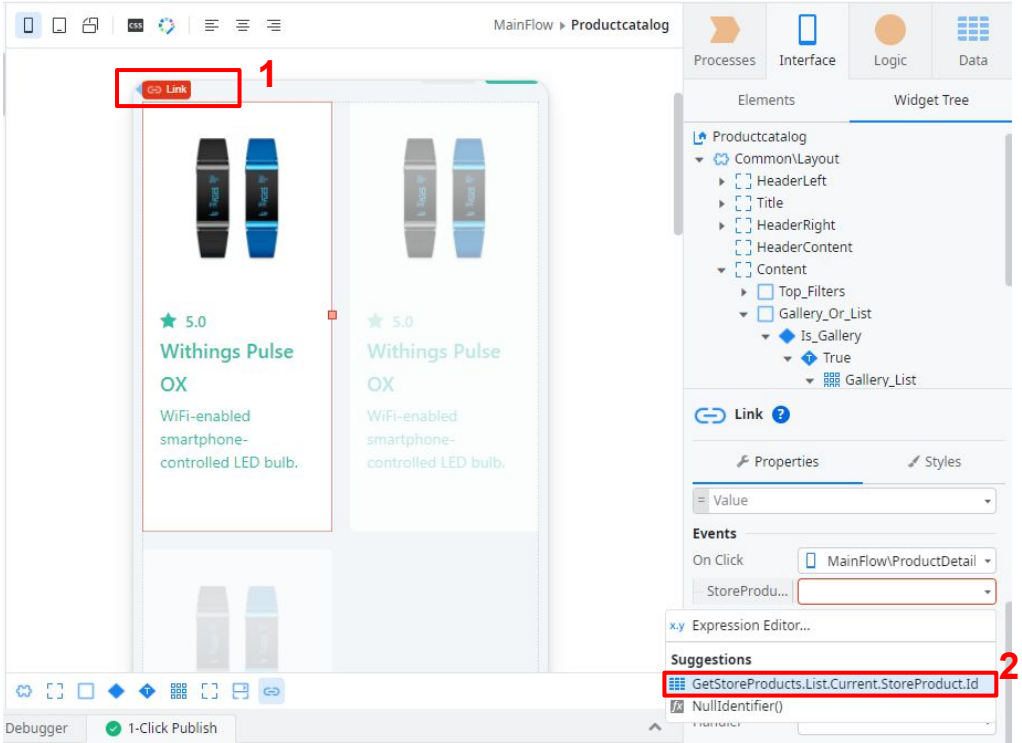
To help you select an enclosing element, you can use the **breadcrumbs** on the bottom of the main view:

2. Right-click on the **Card_Gallery**;
3. Select **Link to** and then;
4. **MainFlow\ProductDetail screen**.

Assign Product Id

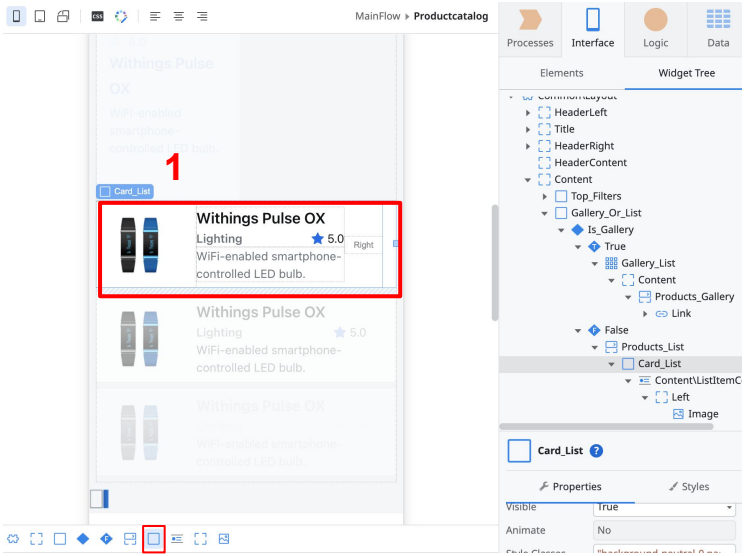
1. Make sure the **Link** widget is selected;

2. On the **On Click** event assign to the **StoreProductId** the value **GetStoreProducts.List.Current.StoreProduct.Id**.



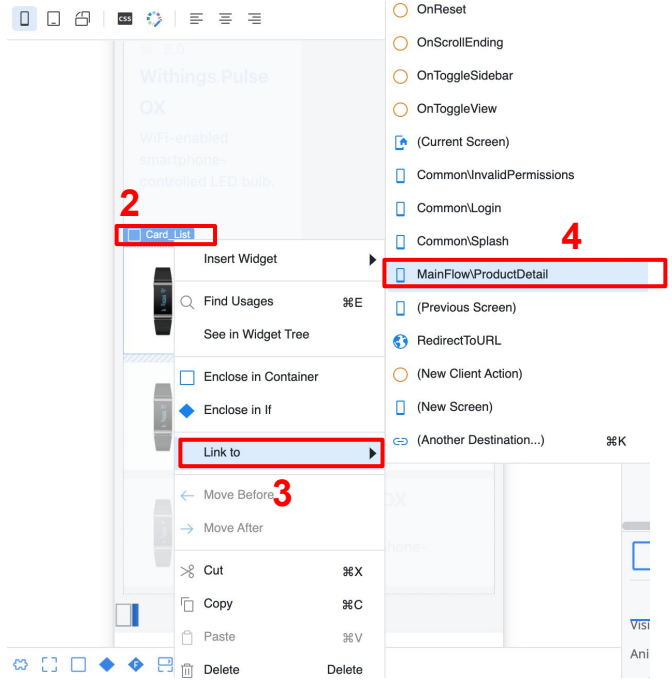
- Now let’s do the same for the Product List that is below the Product Gallery. Remember, we want to enable users to choose a product and see its details.

Section 3 C 1. Link Image to Product Detail Screen



1. Scroll down until you see the Product List;

2. Select the **Card_List** widget;



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

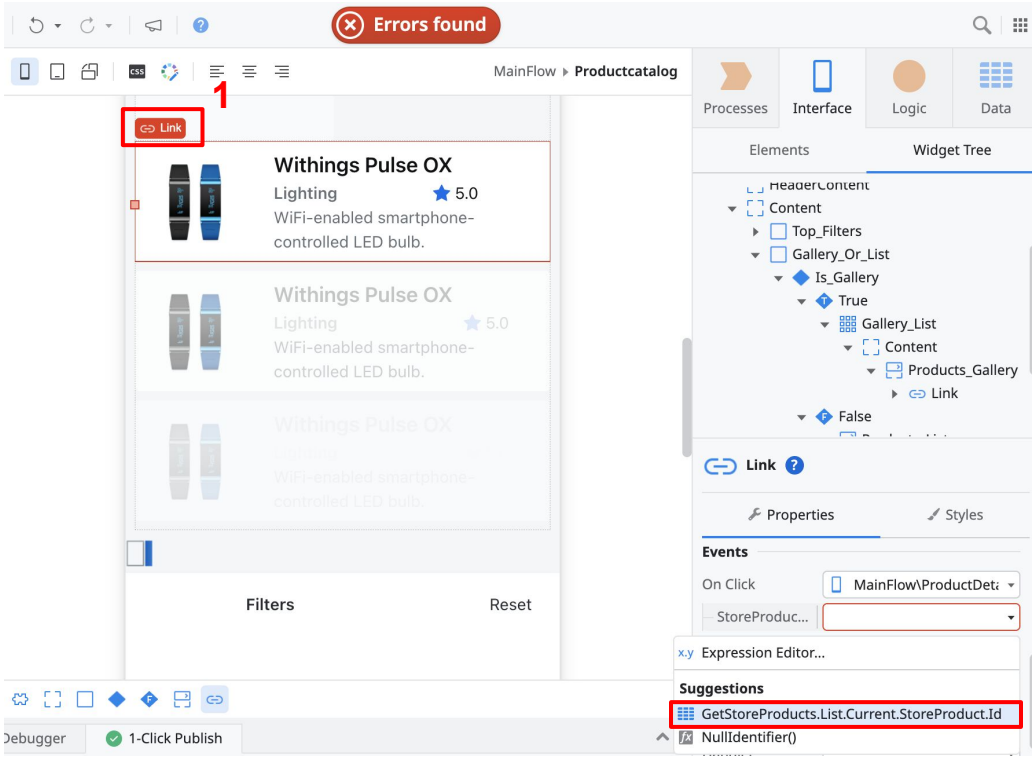
QUICK TIP

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

Assign Product Id

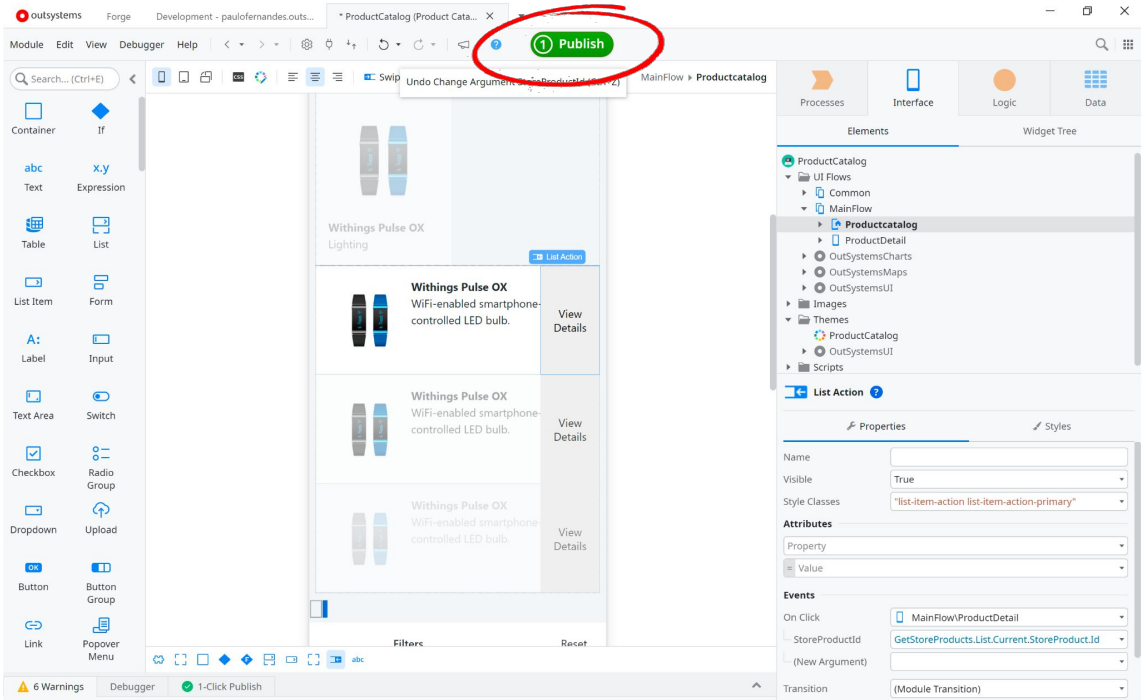
1. Make sure the **Link** widget is selected;

2. On the **On Click** event assign to the **ProductId** the value **GetStoreProducts.List.Current.StoreProduct.Id**.



Deploy and Test it

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



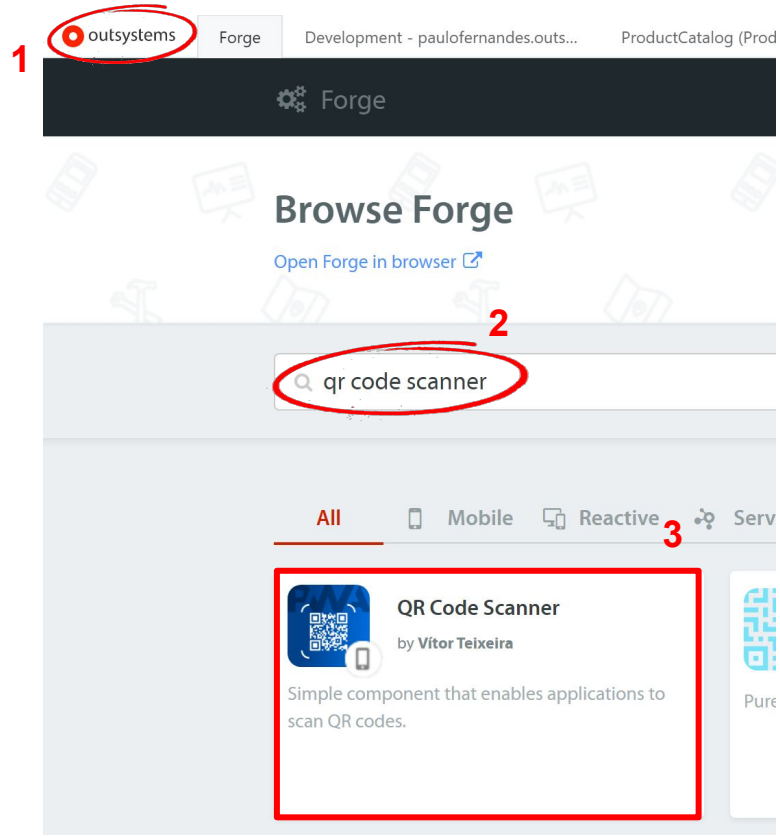
Section 4

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.

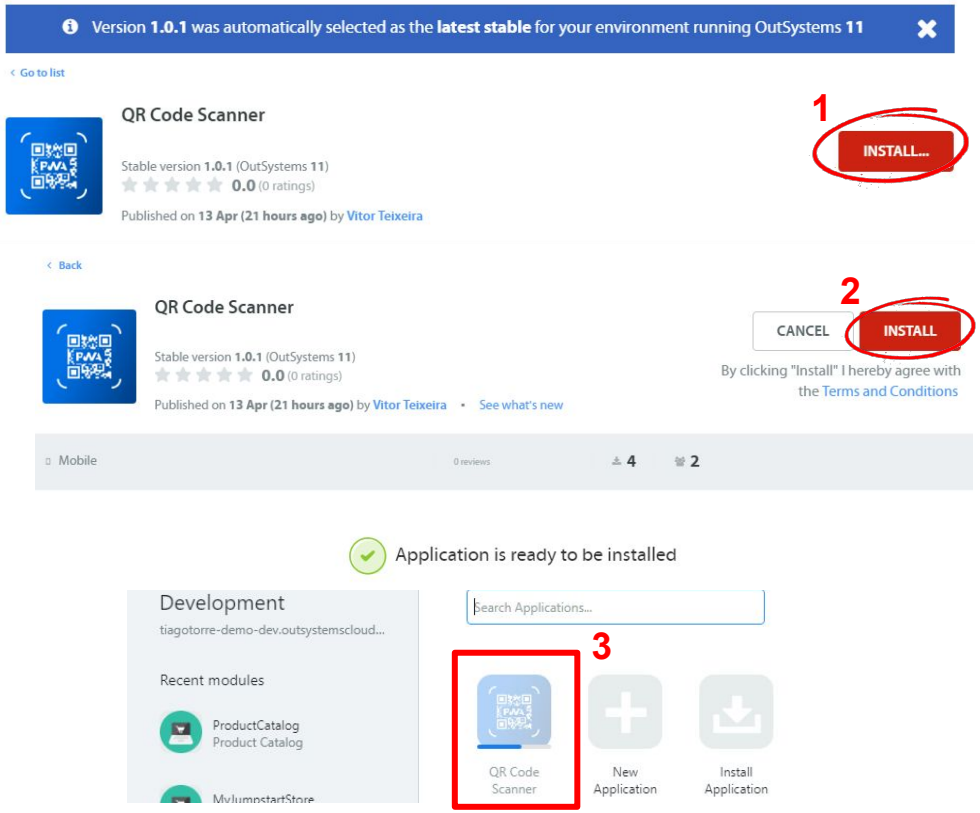
QR Code Scanner Plugin

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. 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.

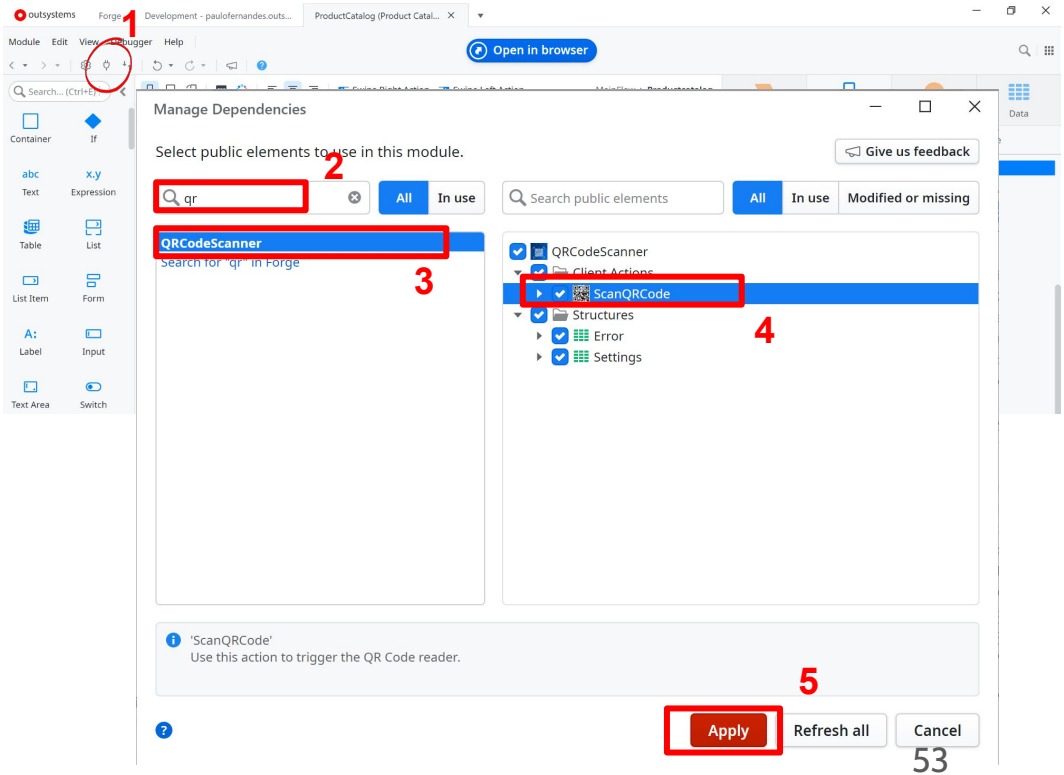
- Now that we have the QR Code Scanner plugin installed in our environment lets include it in our mobile application by adding a new dependency.

- 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

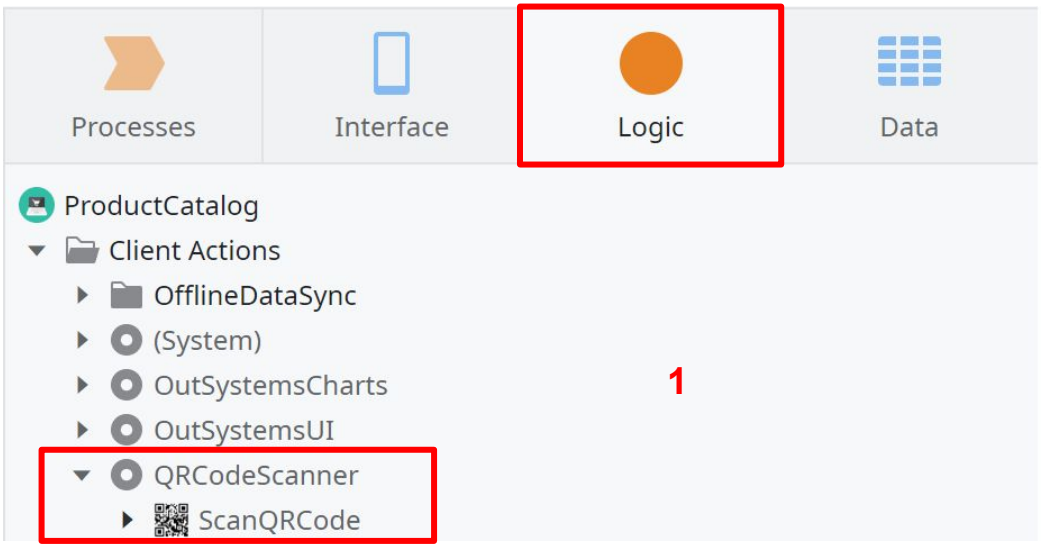
Let's use the Forge Component installed in the previous steps.

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

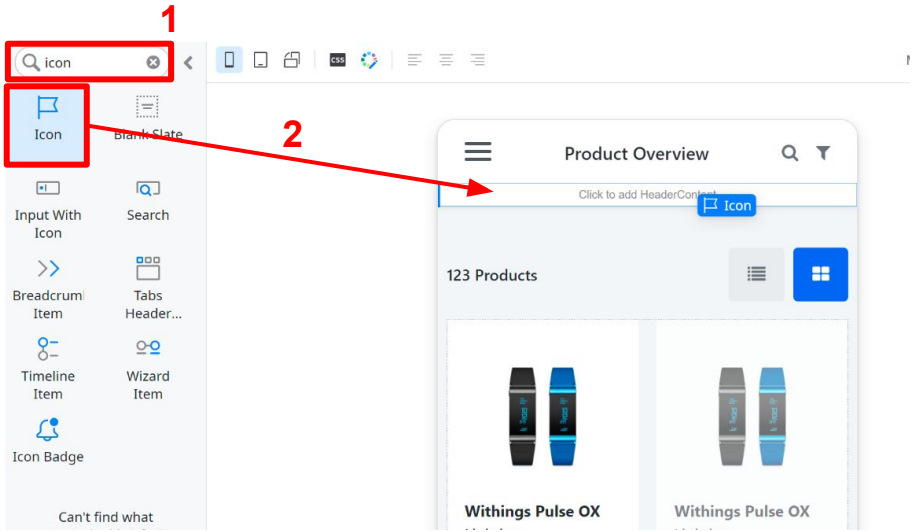


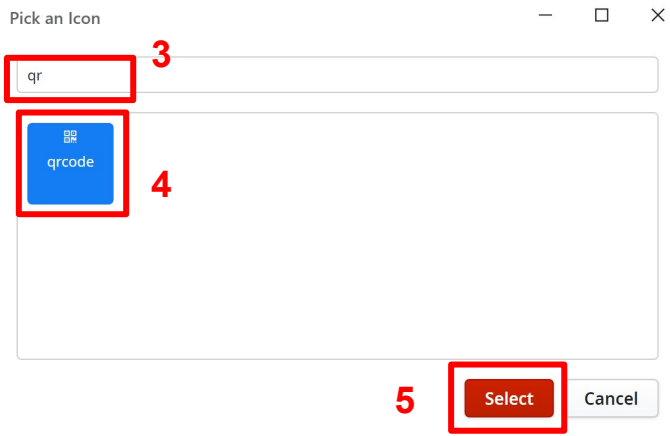
Review Dependencies Added

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



Section 4 B 3. Add QR Code Icon to Screen





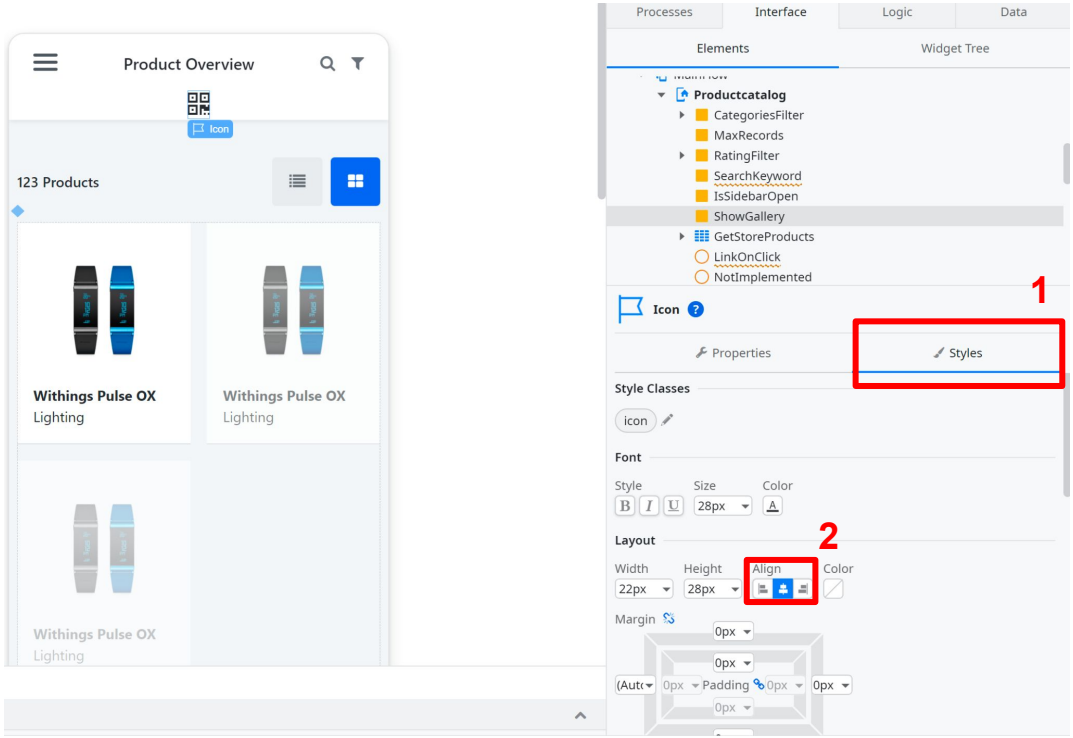
Let's create a button to trigger the QR Code.

1. On the **Widget Toolbox** on the left search for the word "icon"
2. Drag and drop an **Icon** widget into the **HeaderContent** area.
3. When prompted for the Icon type, search for **qrcode**
4. Select the **qrcode icon** and click **Select.**

New Client Action

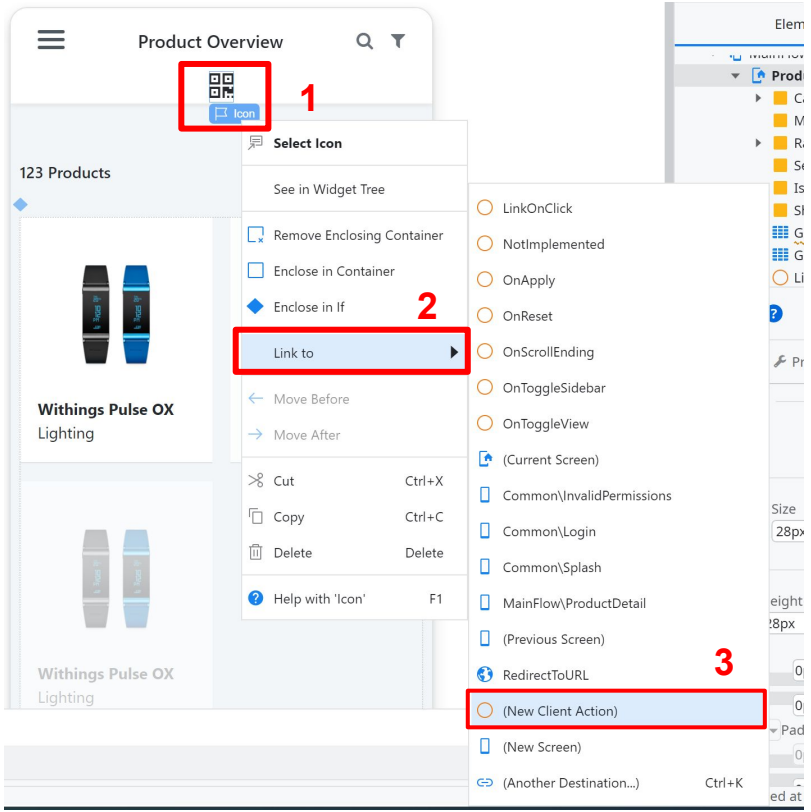
1. With the **icon** still selected, Click on the **Styles** tab

2. Align **Center**



New Client Action

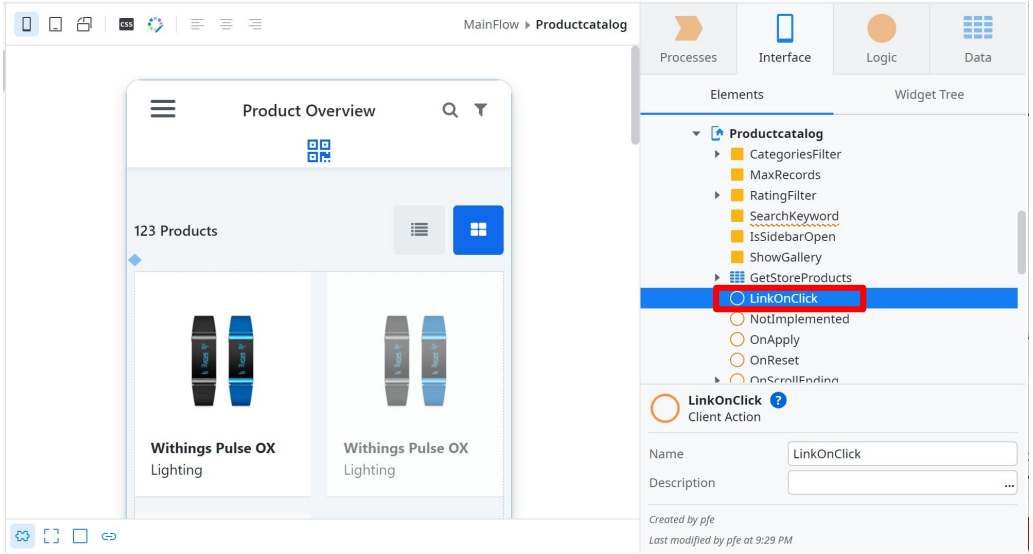
1. **Right click** the new QR code icon
2. Click on **Link to**
3. Click on (**New Client Action**)



New Client Action

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

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

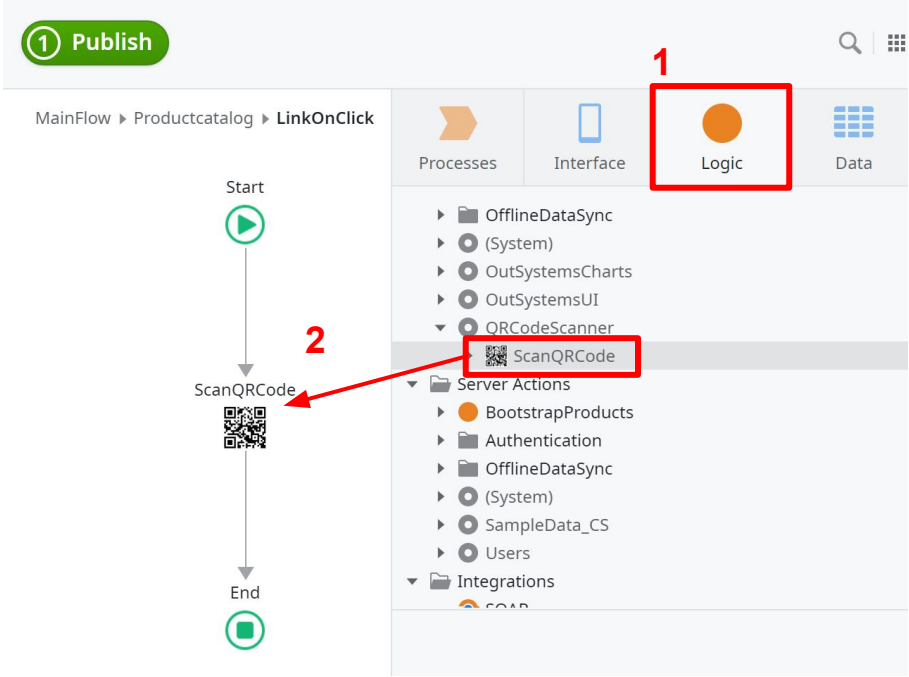


ScanQRCode Action

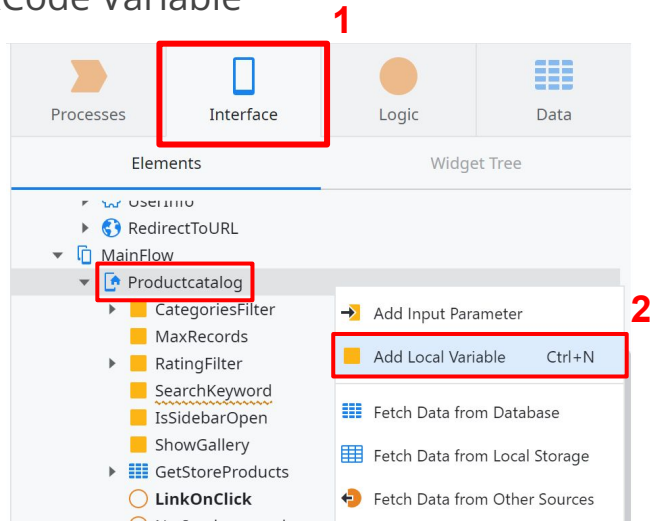
Look for the **ScanQRCode action** that you have added as a dependency earlier.

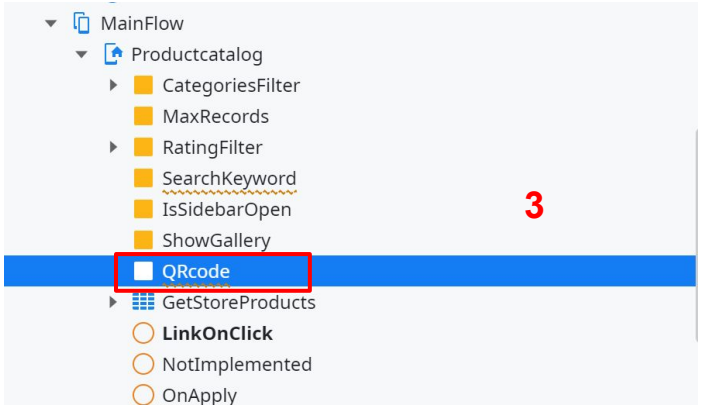
1. This is found under **Logic** > **Client Actions** > **QRCodeScanner Plugin** > **ScanQRCode**.
2. **Drag and drop** this into your **Action Flow**.

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



QRCode Variable





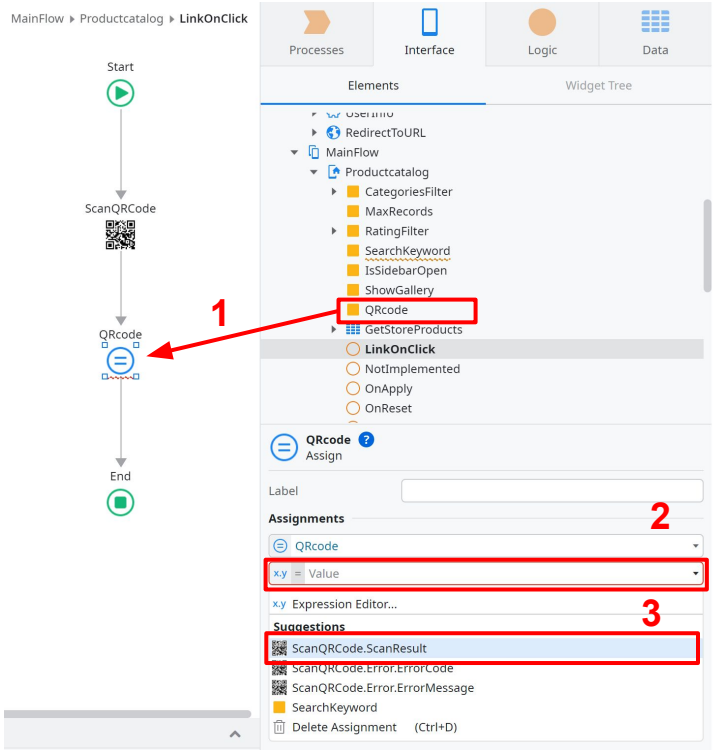
We need to store the QR code scan results into a variable, so that we could use the variable to filter the list results.

1. To add local variables, Click on the **Interface tab**
2. **right click** the **ProductCatalog** 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

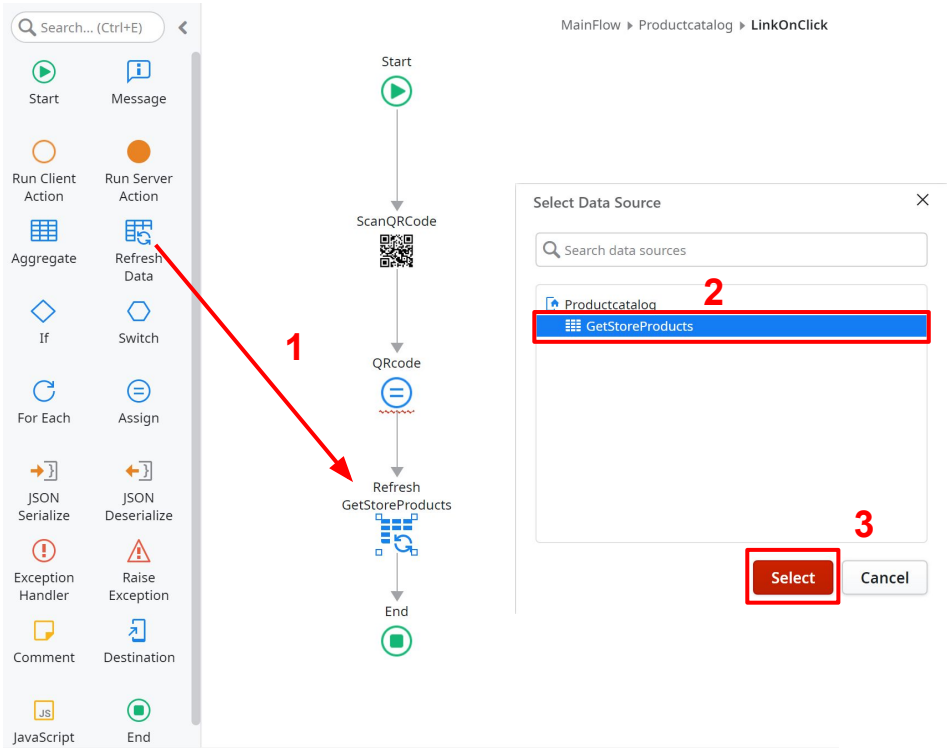
2. Set the Assignment to **QRcode = ScanQRCode.ScanResult**



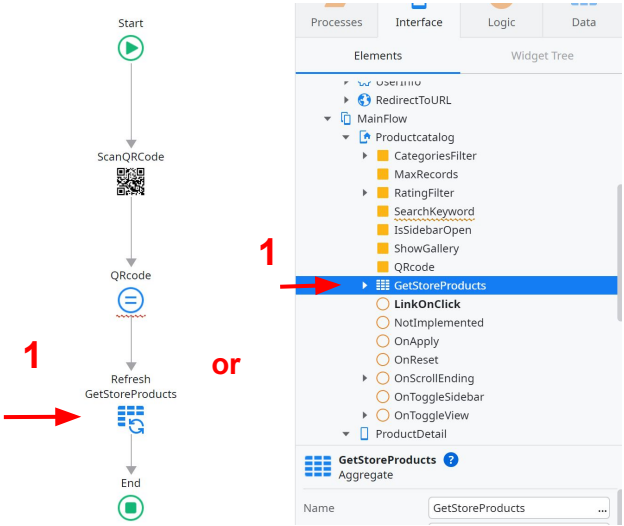
Refresh Products List

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.
2. When prompted for the data source to be refreshed, select **GetStoreProducts**.
3. Press **Select**

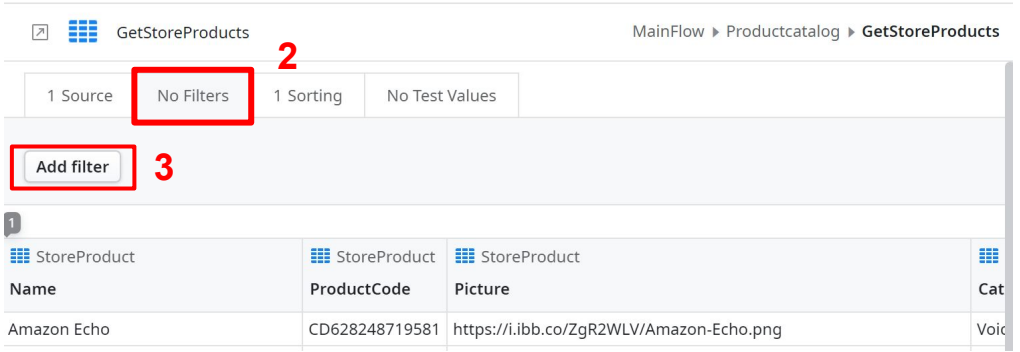


Add Aggregate Filter



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 **GetStoreProducts** aggregate.



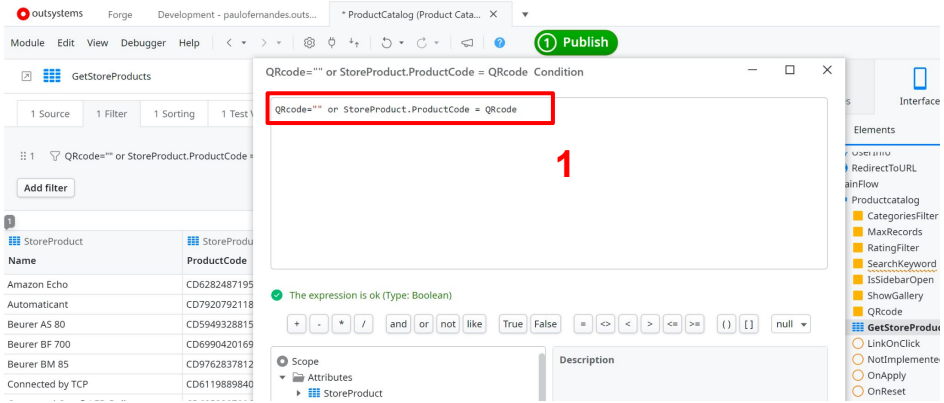
2. Click **Filters**, and **Add filter**

The GetStoreProducts aggregate is the query that the ProductCatalog Gallery and List gets its values from

Add Query

1. Enter the following query:

   QRcode="" or StoreProduct.ProductCode = QRcode



Explanation of the query:

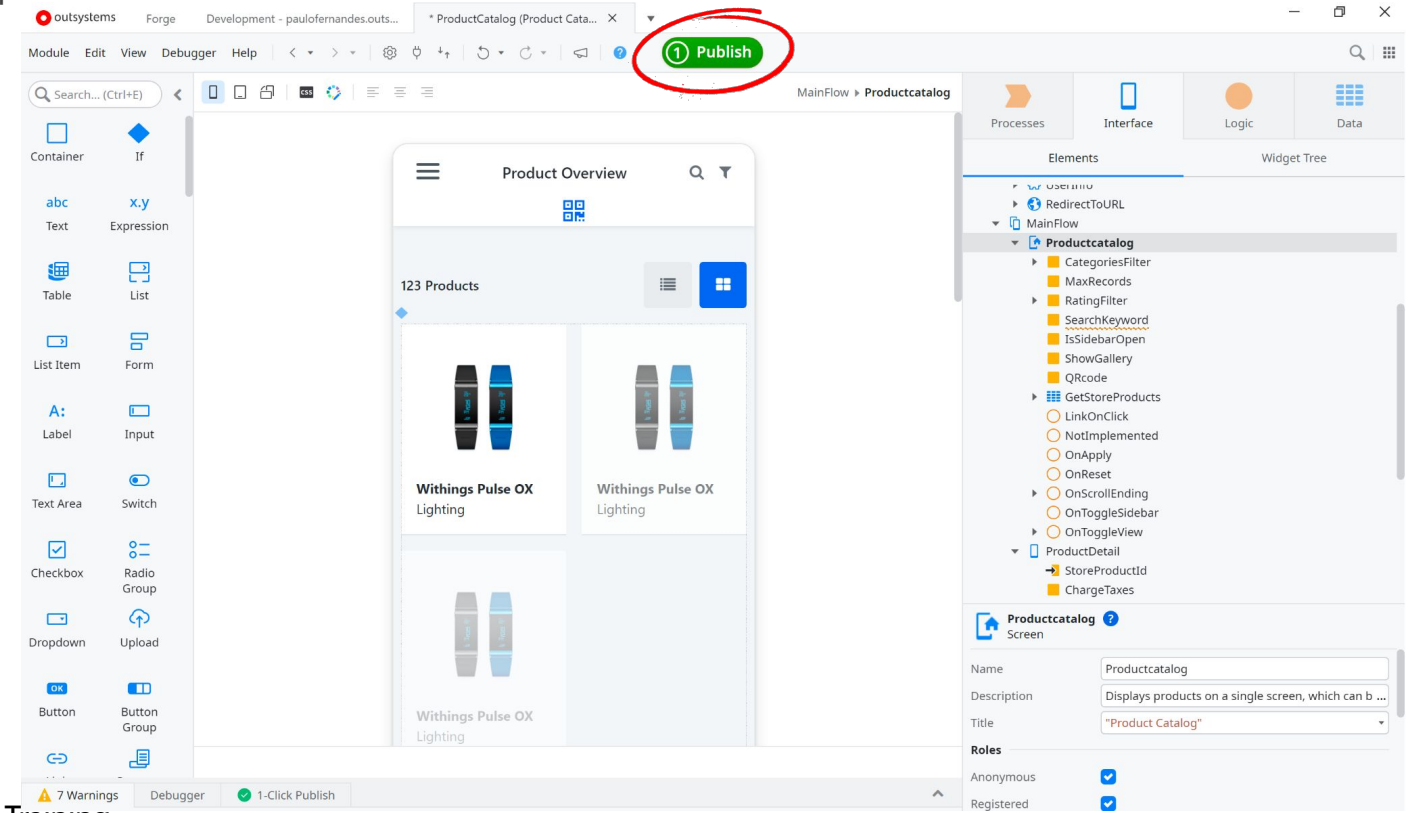
- QRcode =""

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

- or StoreProduct.ProductCode = QRcode

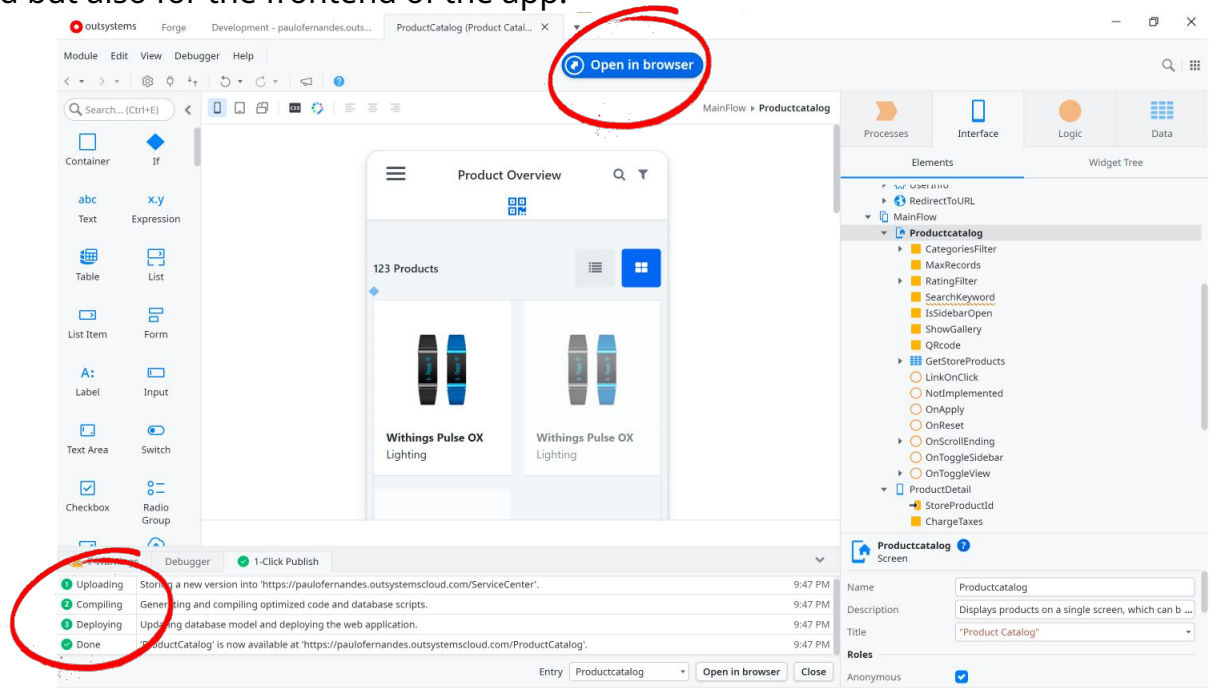
  Otherwise if the **QRcode** is not empty, we will filter based on the **Product Code**.

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
Test QR Codes
Start scanning the product QR Codes

Test your Mobile App

QR codes

Inside your app, tap the QR Code icon to scan the QR codes in the following slides.

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

Save and Delete Products - Add Logic

Bonus exercise:

- In the Product Detail add the logic on the “Save” button to allow products to be updated on the Products entity in the database.

- Also add the logic on the “Delete Product” button to allow products to be deleted from the database. Tip: In here you will need to access the pop-up window that comes up when you click that button.

Expose Rest API

Bonus exercise:

- To allow other systems to access the products you have on the database, you would need to expose a REST or SOAP API. Let's use a REST API in this exercise.

- In this exercise, lets expose a REST API with two methods:
  - GetStoreProducts - Allowing third parties to obtain the list of products in the database. This method doesn't receive input parameters but returns a products list.
  - DeleteStoreProduct - Allowing third parties to delete a specific product from the database. This method receives a ProductID and returns a text message.

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