**OutSystems JUMPSTART**

Exercise 2: Build your own mobile app

…………………………………………………………………………………………………………

|  |
| --- |
| In this exercise, you will be creating a Native Mobile Application that lists the products you created from *Exercise 1: Build your own Web App*.  You will learn the basic differences between web and mobile apps, using a barcode plugin, and using mobile templates. |

**TABLE OF CONTENTS**

[**Install a Forge Component**](#_z0cn8qryl09o) **1**

[**Create a Mobile App**](#_q5w5go28e2qi) **2**

[**Add Dependencies**](#_sn759r23s898) **5**

[**Create a gallery using a UI Template**](#_cbv37fzb4u67) **7**

[**Create a page without a template**](#_j7c1m1nuyfy4) **13**

[**Create a List populating from a data source**](#_ff2n63o95os7) **14**

[**Add Product’s Picture into the list**](#_2jdpqgnnfax5) **16**

[**Add links into the sidebar Menu**](#_7lq2he7rqk14) **22**

[**Swipe item to view Detail Page**](#_panve8wwi1z7) **23**

[**Add a barcode scanner**](#_zgdoy6i416yw) **28**

[**Test your mobile app in your phone**](#_ipezdw53fgkt) **35**

# Install a Forge Component

|  |
| --- |
| OutSystems Forge (<https://www.outsystems.com/forge/>) is an open-source library, where developers can browse components (Connectors, UI Widgets, Templates, Native Device Plugins, Extensions, Sample Applications, etc), and install in their environment.  This section explains the steps of installing a component from the OutSystems Forge. |

The following Forge Component is required for this Exercise:

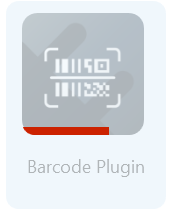
1. **Barcode Plugin**Plugin for scanning barcodes and QR codes.
   1. In Service Studio, navigate to the leftmost OutSystems tab and make sure you are logged in, *or* navigate to <https://www.outsystems.com/forge/>.
   2. Search for theapplication **Barcode Plugin**

|  |
| --- |
|  |

* 1. **Install** the component.  
     It will search for dependencies. Click **Install** again. 

|  |
| --- |
|  |

* 1. The application and any dependencies will automatically install.

  
*.*

Wait for your installation to complete (no icons showing the red progress bars).

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.

# Create a Mobile App

* 1. Navigate to the **Development** tab.

|  |
| --- |
|  |

* 1. Create a **New Application**, choose **Mobile App** and 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.*

* 1. Pick the template: **Phone**

|  |
| --- |
|  |



*FYI: You can also create custom templates - which is a to package and reuse consistent branding and architectural patterns such as authentication, across your development teams. A template can include:*

* *Branding*
* *Security*
* *More Patterns*
* *Predefined links to core services*
  1. Specify your Application Name as **Product Catalog** and **upload an icon.** You can find the icon in the Resources folder you downloaded before this exercise.

|  |
| --- |
|  |

* 1. Create a **Mobile** module for your application called **ProductCatalog**.

|  |
| --- |
|  |

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

|  |
| --- |
|  |

# Add **Dependencies**

|  |
| --- |
| 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. |

* 1. Click the **Manage Dependencies** icon

|  |
| --- |
|  |

* 1. Let’s use the Forge Component we installed earlier.   
     Under Producers, search for **BarcodePlugin.**   
     In the elements, select **CheckBarcodePlugin** (for checking if the device supports barcode scanning) & **ScanBarcode** (for triggering the device’s scan barcode action).

|  |
| --- |
| *Don’t* click OK yet. |

* 1. Add a reference to the **Product** entity table that we created in the first exercise (found under **<Name of your web application project in Exercise 1>**).

|  |
| --- |
|  |

*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. After clicking OK and adding the references, you can now see the **ScanBarcode** action under the **Logic** tab and the **Product** entity under the **Data** tab in your project.

|  |
| --- |
|  |

This exercise 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.

# Create a gallery using a UI Template

|  |
| --- |
| The platform provides screen templates to help developers create world-class User Interfaces easily. Not only do the Templates contain UI design, it also has pre-created actions to accelerate development of the page actions such as filters.  OutSystems UI templates pre-creates your screens with sample data so you don’t have to start from scratch.  In this chapter, we will create a mobile Product Gallery page that contains a filter. |

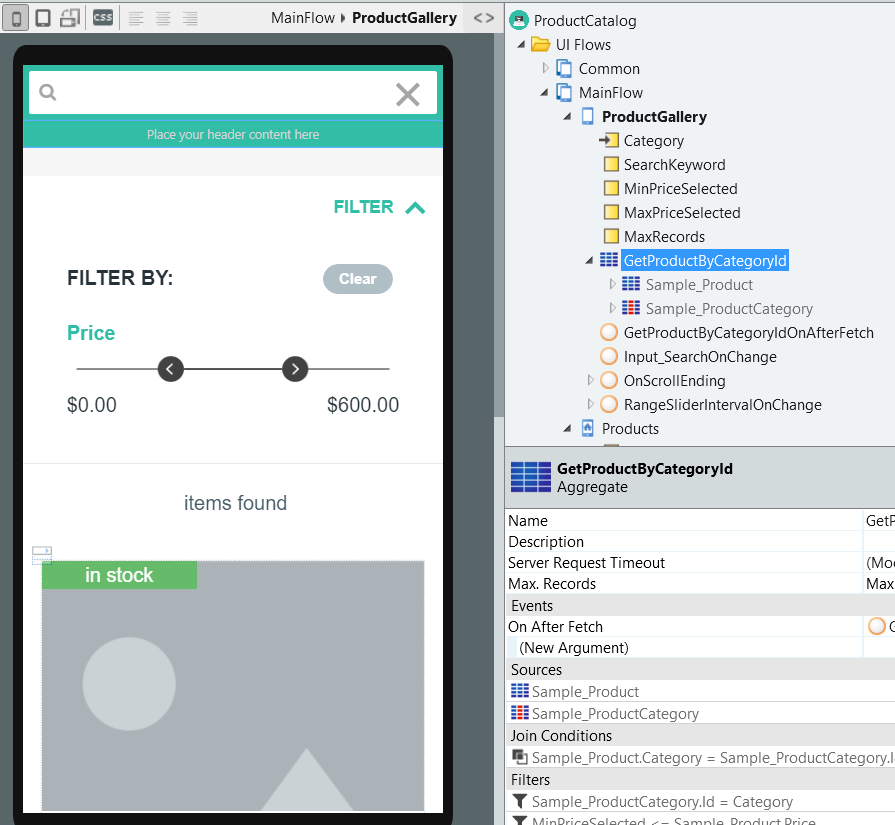
* 1. Create a new page in **Interface** → **MainFlow** by dragging an empty **Screen** from the left to the middle, or right-clicking on **MainFlow** and selecting **Add Screen**

|  |
| --- |
|  |

* 1. Select the **Product Gallery** Template.  
     Name your page as **ProductGallery.**

|  |
| --- |
|  |

* 1. A page with pre-built template is automatically created.  
     Notice that it’s using Sample Data “**Sample\_Product**” and “**Sample\_ProductCategory**” for the **GetProductByCategoryId** query within the selected template.



*With UI Templates, not only is the User Interface pre-built, it also includes the backend actions related to the User Interface (in this case, filter actions), and it also comes with* ***Sample Data*** *as a placeholder.*

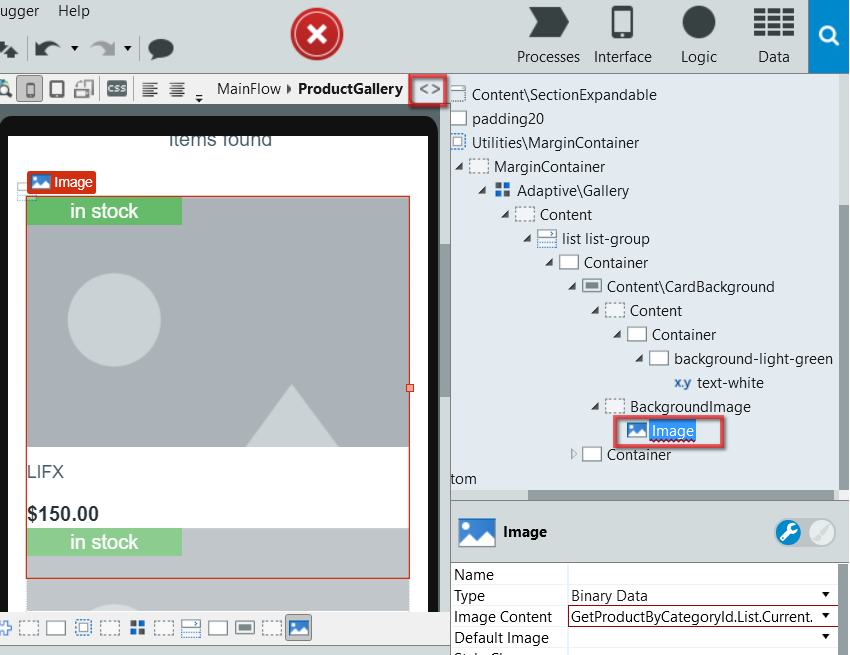
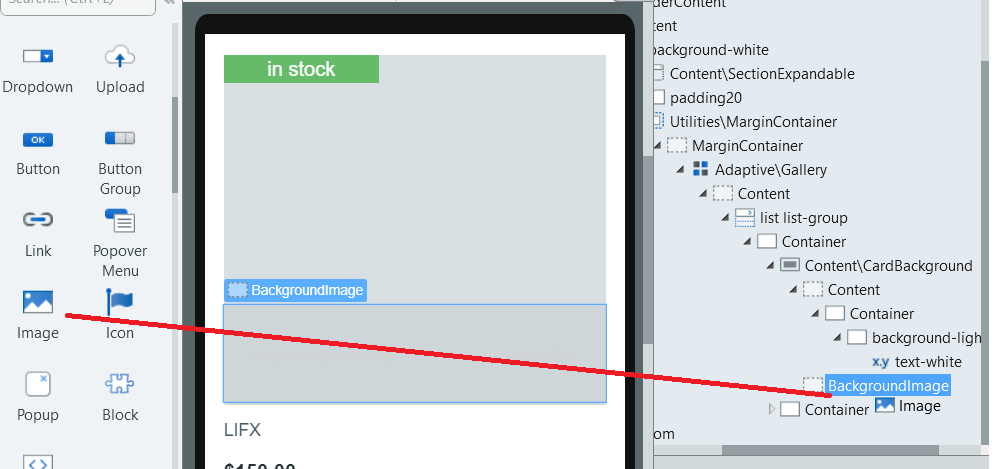
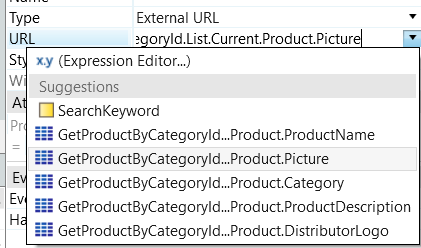
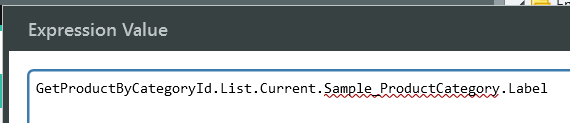
*In actual projects, we would need to replace the Sample Data with our own data, including pre-built backend rules (such as filters) that reference to Sample Data.*

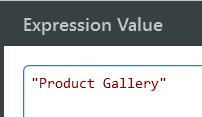
* 1. Drag and drop the **Product** entity into the Gallery area. Make sure you see “**Replace data with Product**” before letting go of the mouse click.

|  |
| --- |
|  |

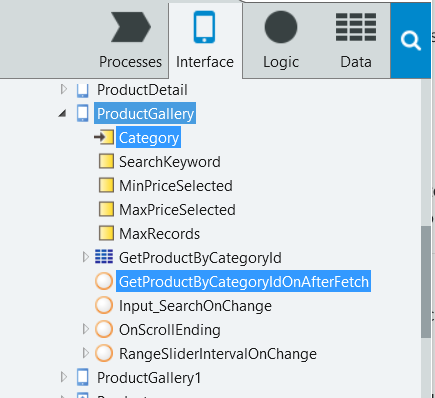
* 1. However even after replacing the list with our data, some variables from the Template may still need to be replaced manually.

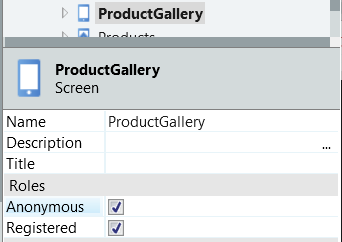
|  |
| --- |
| Next we will fix these errors. |

* 1. Error “*Unknown Object ‘Image’ in expression*”  
     Navigate to the image block in the widget tree.  
     
  2. **Delete** the image block.
  3. Add a new **Image** control in the same position.  
       
     Update the following properties:   
     **Type: External URL  
     URL: ...Product.Picture**  
     
  4. *[if you get the the following error, go through this step]* Error “*Unknown object 'Sample\_ProductCategory' in expression.*”  
     

This is the title of the page. Change this value to **“Product Gallery”** with the quotations.  


* 1. Under **Interface** Tab→ **ProductGallery** screen, Delete the **Category** input parameter and the **GetProductByCategoryIdOnAfterFetch** Client Action. We don’t need these .



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

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

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

# Create a page without a template

* 1. Under the **Interface** tab, right click **MainFlow,** select **Add Screen**.

|  |
| --- |
|  |

* 1. Select the **Empty** template.  
     Enter the **Screen name: *Products***

|  |
| --- |
|  |

* 1. You now have your empty *Products* page.

|  |
| --- |
|  |

# Create a List populating from a data source

|  |
| --- |
| Similar to building apps in Web, in mobile, there are a few ways of building a listing page.  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 page. |

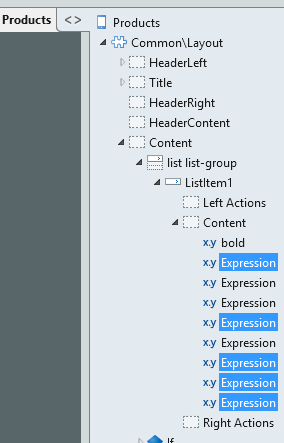
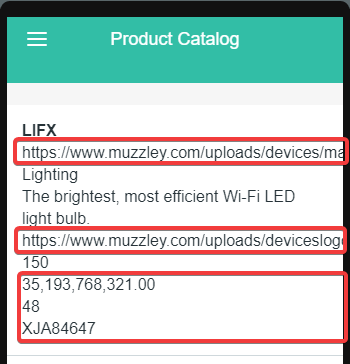
* 1. Add a title. Type **Product Catalog** in the Title area of your mobile app.

|  |
| --- |
|  |

* 1. Drag and drop the **Product** entity into the Content area.

|  |
| --- |
|  |

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

* 1. We now have a list of products!  
     You can remove unnecessary attributes that we do not need.  
     Multi-select the items using **CTRL+click**, then press **Delete**.  
     

# Add Product’s Picture into the list

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

|  |
| --- |
|  |

* 1. We will format the list items to show 2 columns. Image on the left column, and the descriptions on the right column.   
     To do this, we will enclose the objects into **Containers** (similar to <div> in HTML)  
     Right click the image > **Enclose in Container**

|  |
| --- |
|  |

* 1. With the container still selected, toggle to the Style Editor () and set the **width** to **4 col**

|  |
| --- |
|  |

* 1. Select all the text attributes (Shift + Click), Right click > **Enclose in Container**

|  |
| --- |
|  |

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

|  |
| --- |
|  |

* 1. You now have 2 columns. But there’s no space in between the image and the text.   
     With the same container still selected, click the chain icon () to unlock the padding lock, and change the style attributes to have a **Padding-left** of **10px**.

|  |
| --- |
|  |

* 1. Make sure your structure looks like the following:

|  |
| --- |
|  |

* 1. We will specify the image source to the **Picture** attribute of our **Product** entity.   
     Select the image and set the following attributes:  
     **Type: External URL  
     URL:** Select the suggested **GetProducts….Picture**

|  |
| --- |
|  |

# Add links into the sidebar Menu

* 1. In the **Interface** Tab, Navigate to **Common > Menu** to open the side menu block.

Drag and drop the **Products** page and the **ProductGallery** page into the panel to automatically add the hyperlinks.

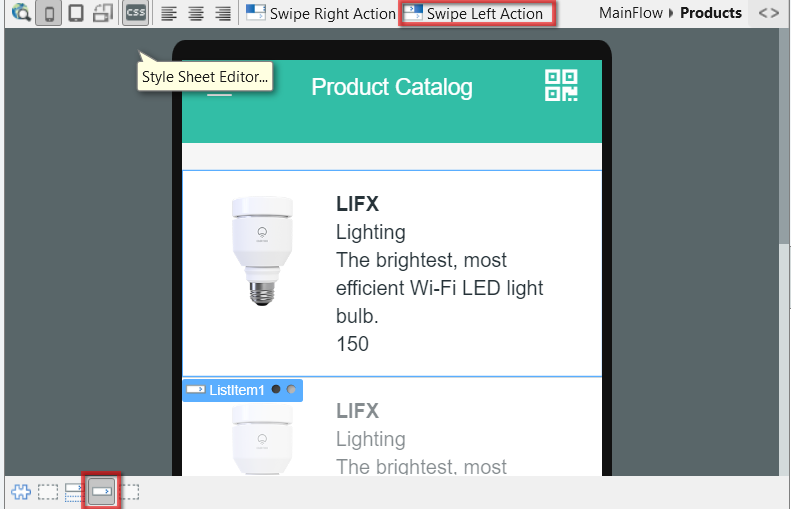
|  |
| --- |
|  |

* 1. Feel free to deploy () and test ().

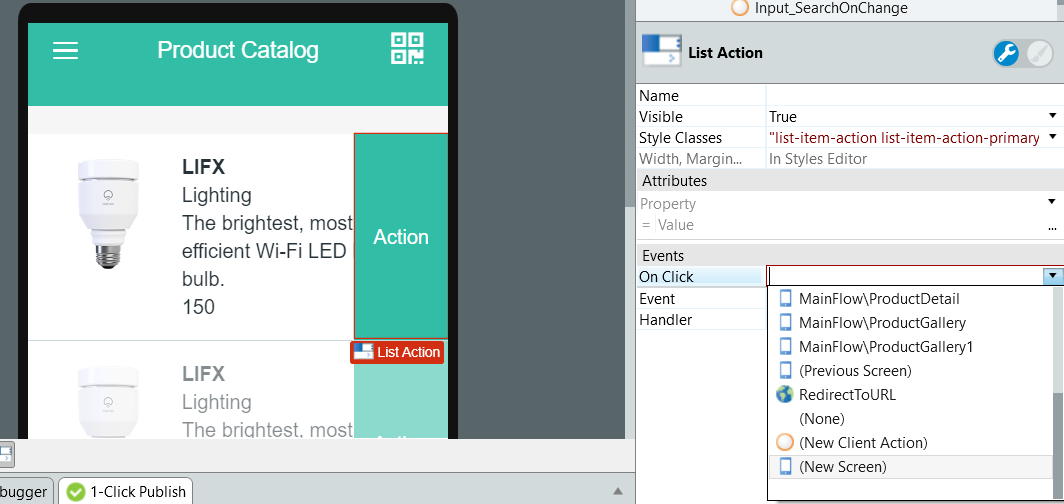
# Swipe item to view Detail Page

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

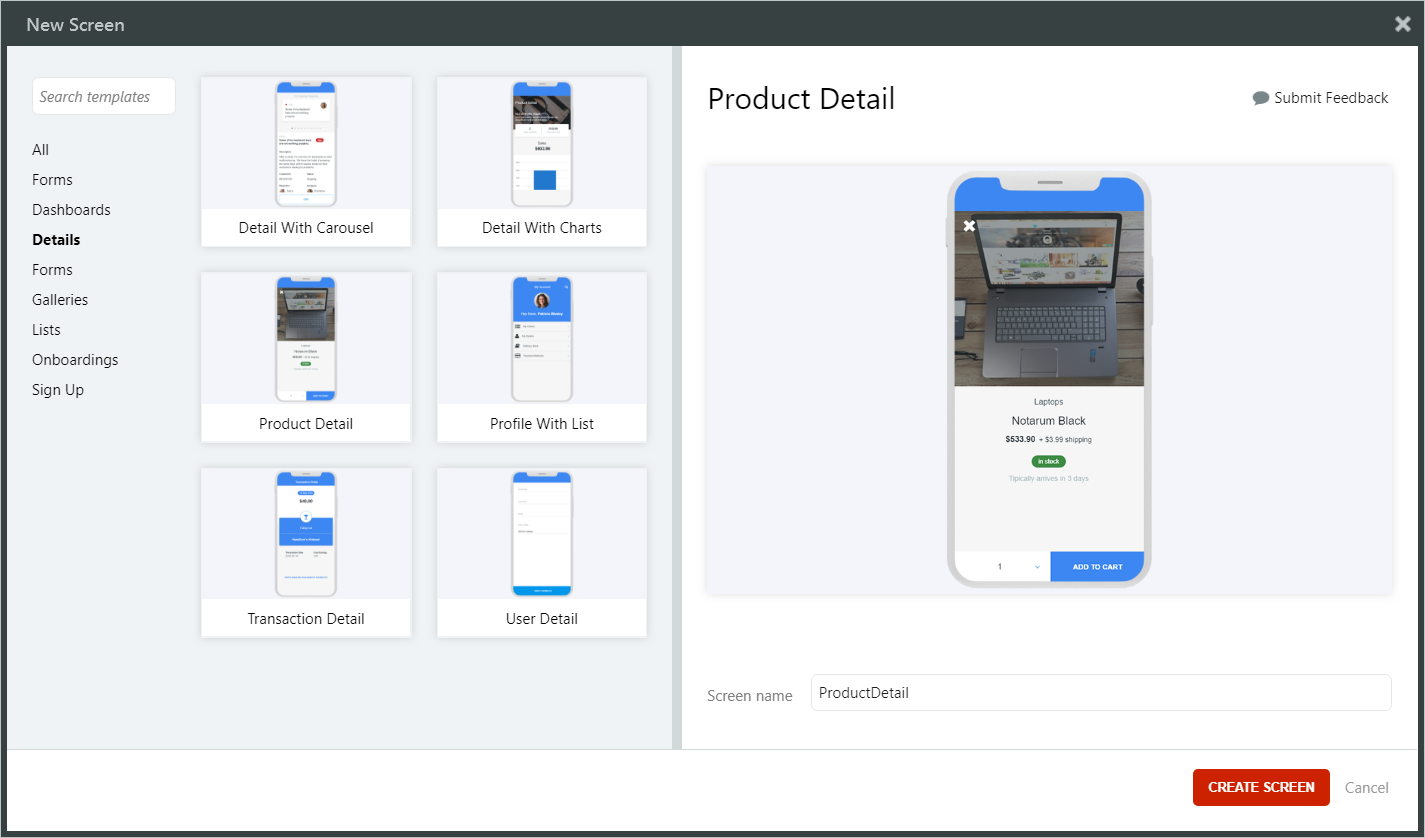
* 1. Back to the **Interface** Tab → **MainFlow** → **Products** Page, select the *List Item* (make sure the Icon “” is selected), and click **Swipe Left Action**.

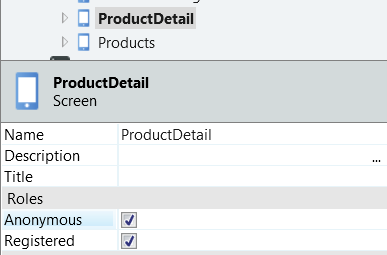


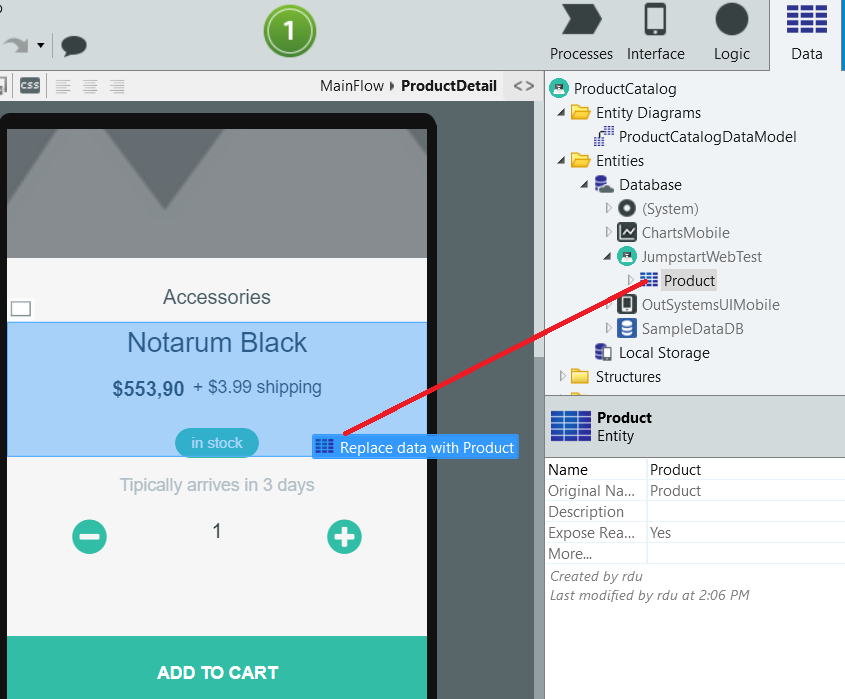
* 1. In the **On Click** event, select (**New Screen)** to create our Detail Page.



* 1. In the UI Templates, select **Details** > **Product Detail**. Name the page as **ProductDetail.**



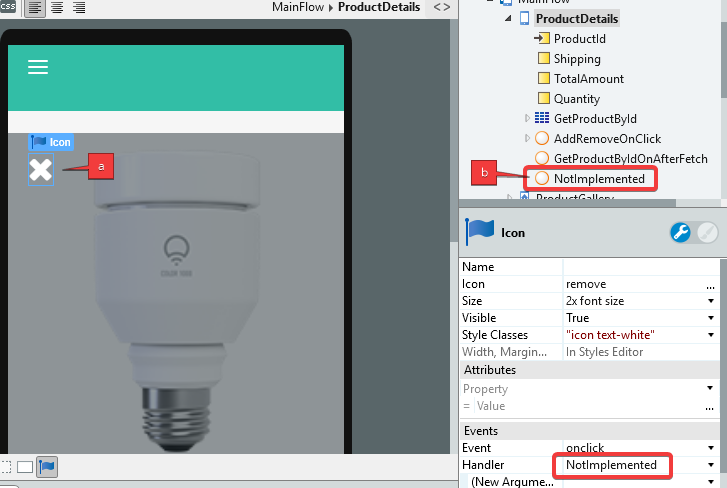
* 1. Mark the **ProductDetail** screen to be accessible by **Anonymous** users.  
       
     
  2. Replace the data with the **Product** entity by dragging it from the **Data** Tab onto the Screen until it says “**Replace data with Product**”.



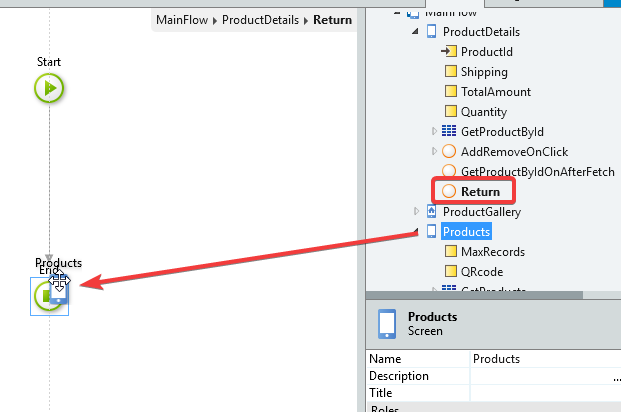
* 1. Now it’s up to you to do the next steps, similar to what we did for the Product Gallery page.   
     *Hint:* Delete image widget and create a new one pointing to Image URL.
  2. Map the **Expression** highlighted to the **Product.Category** attribute.



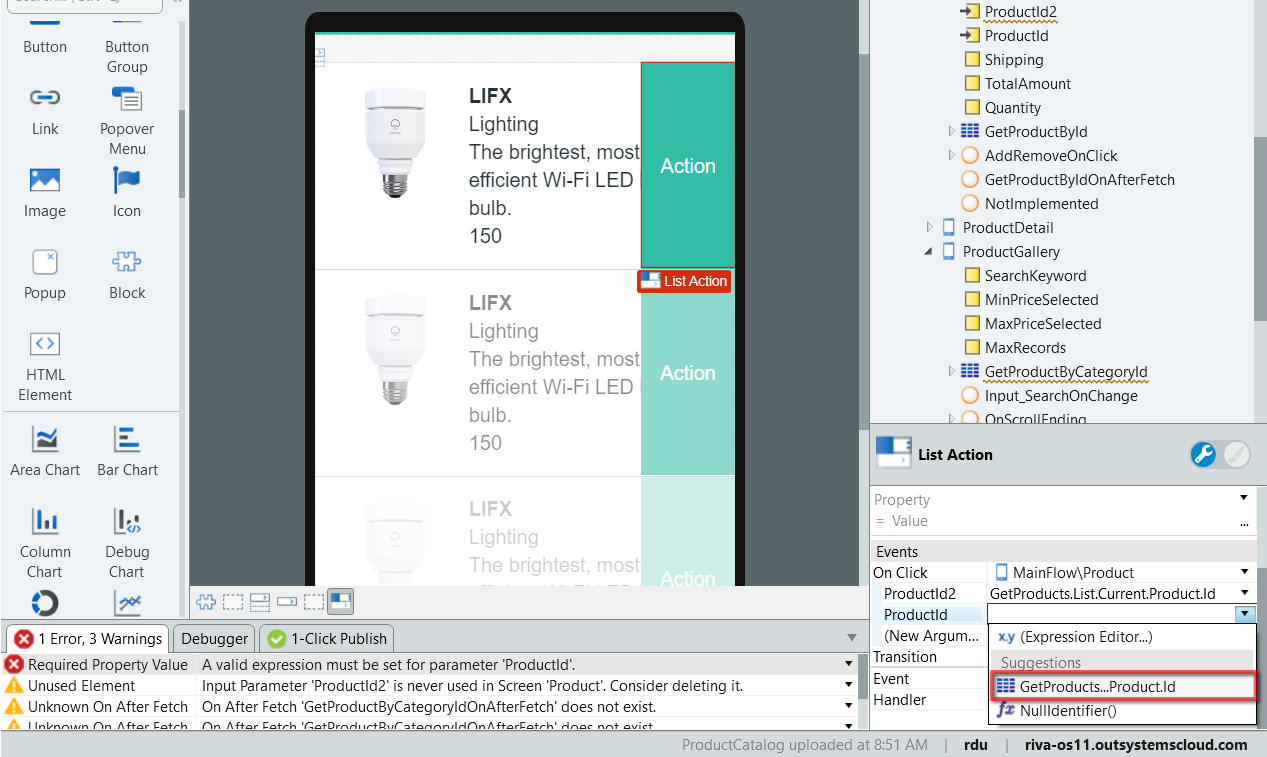
* 1. Select the X mark at the upper-left of the Product Image. Notice that the **onclick** handler of that Icon calls the **NotImplemented** Client Action included from the Product Details template. Double-click the **NotImplemented** Client Action to modify it to go back to the Product listing screen.



* 1. Rename the **NotImplemented** Action to **Return**, and in the Action flow, delete the **Not implemented message** Message, then drag-and-drop the **Products** screen over the **End** icon in the flow.



* 1. Back in the **Products** page, add the missing input parameter **Product.Id** for the **Swipe Left Action**.

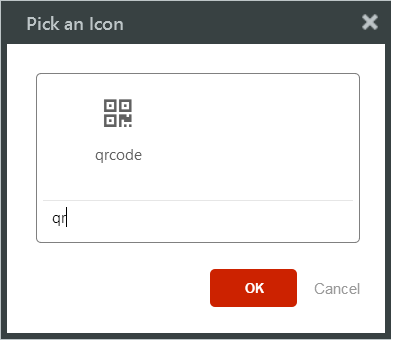


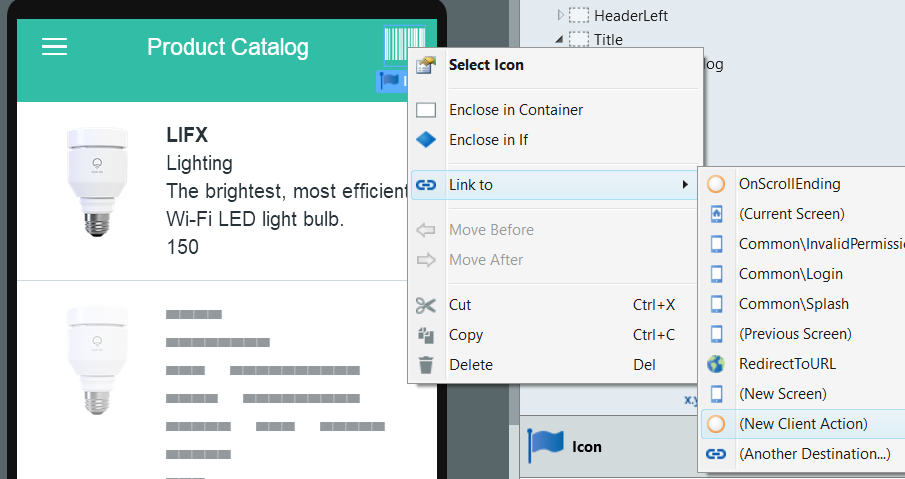
# Add a barcode scanner

|  |
| --- |
| We now have a list of products. But how can we filter the products based on a barcode value that we scanned using the mobile camera?  Note; the barcode scanner works for both QR code and barcodes! |

* 1. Let’s create a button to trigger the barcode.   
     Drag and drop the **Icon** widget into the **Actions** area in the upper rightqr.

|  |
| --- |
|  |

* 1. When prompted for the Icon type, search for **QRcode** or **Barcode** and select the icon.  
     

* 1. **Right click** the new QR code icon > **Link to** > **(New Client Action)**  
     

This creates a client action that triggers on click of the icon.

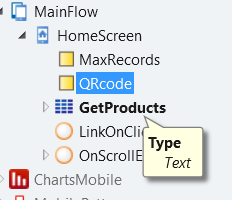
* 1. Look for the **ScanBarcode** action that you have added as a dependency earlier.  
     This is found under **Logic** > **Client Actions** > **Barcode Plugin** > **ScanBarcode**.   
     Drag and drop this into your Action.

|  |
| --- |
|  |

This action will trigger the user’s mobile device camera to scan a QR code.  
  
*Note: For best practices, we should use the* ***CheckBarcodePlugin*** *to check if the client device supports barcode scanning before triggering the* ***ScanBarcode.*** *However, for simplicity of this exercise, we will skip that portion.*

* 1. We need to store the QR code scan results into a **variable** in this screen, so that we could use the variable to filter the list results.   
     To add local variables, **right click** the **Products** page **> Add Local Variable**

|  |
| --- |
|  |

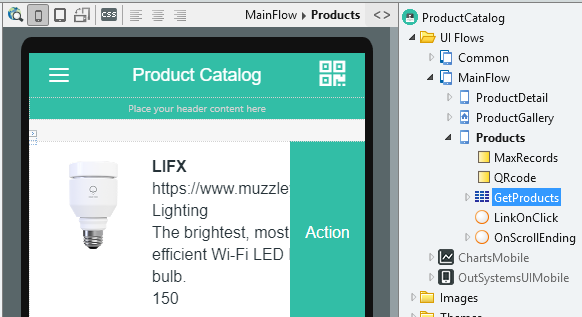
Name the variable as **QRcode.**  


* 1. We need to assign the scan results into the new **QRcode** variable.  
     To do this, drag and drop the **QRcode** variable into the action, and set the Assignment to **QRcode** = **ScanBarcode.ScanResult**

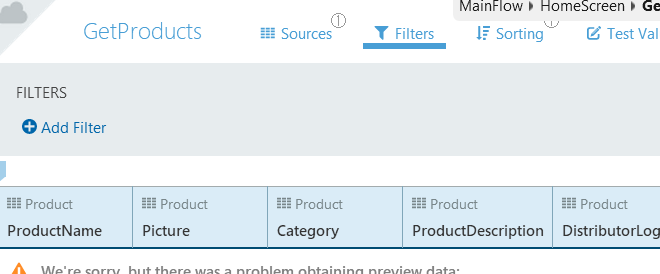
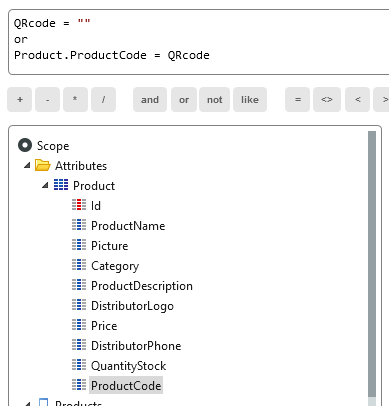
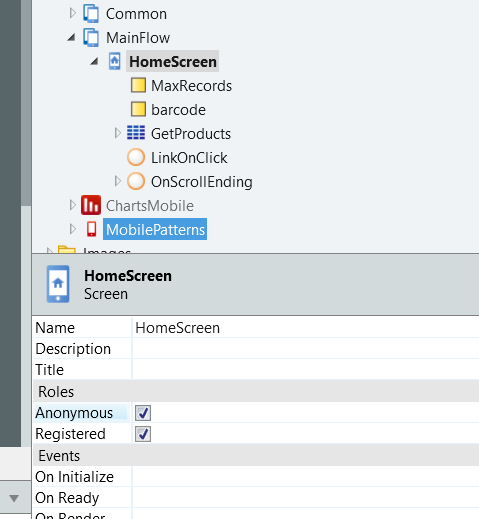
|  |
| --- |
|  |

* 1. After scanning the barcode, we need to refresh the Products list in the screen to reflect our filter (we will do the list filters later).   
     Use the **Refresh Data** widget.   
     When prompted for the data source to be refreshed, select **GetProducts**.

|  |
| --- |
|  |

* 1. We need to re-configure our query to ensure that the list is filtered based on the QRcode value.   
     Open aggregate editor by Double clicking the **GetProducts** aggregate.  
       
     

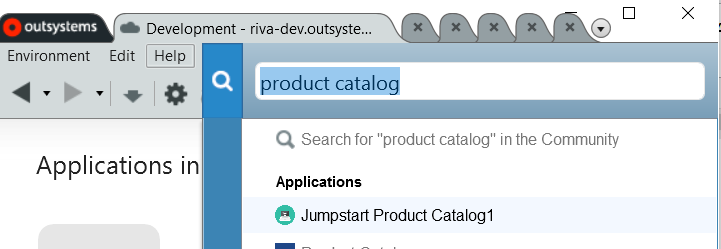
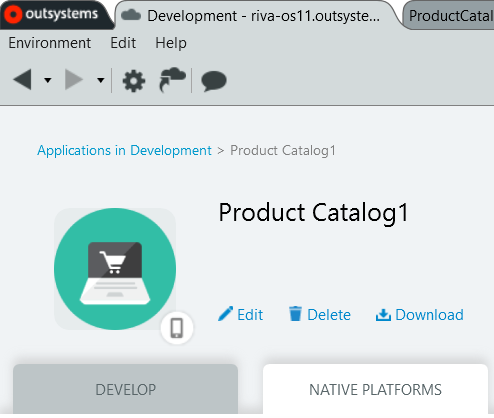
*Note: the GetProducts aggregate is the query that the Products Screen gets its values from.*

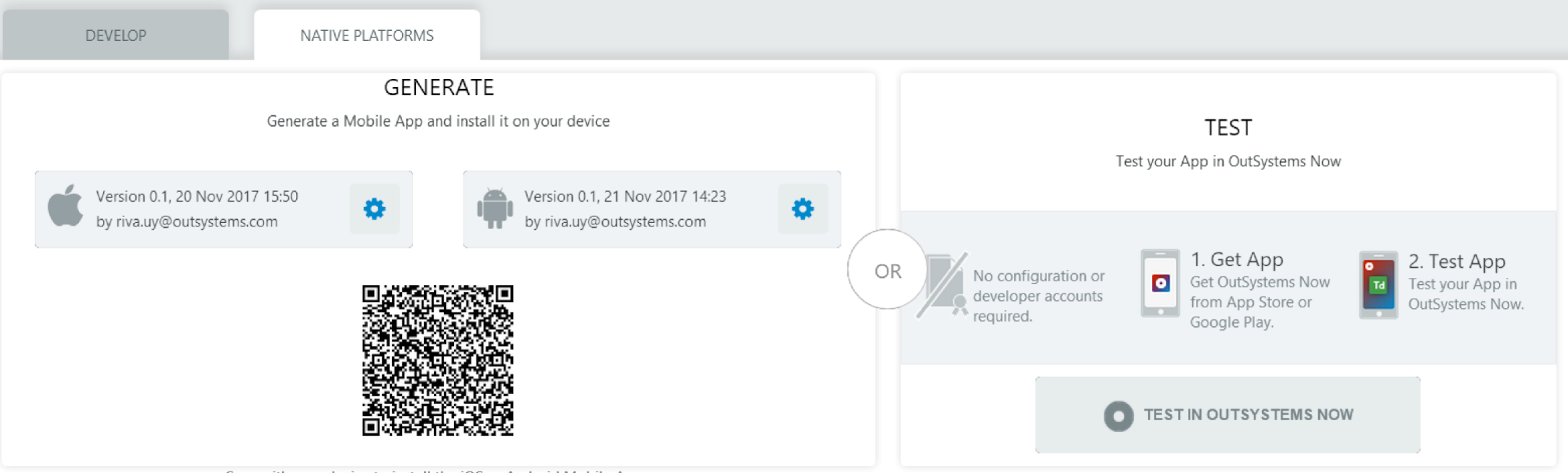
* 1. Click **Filters**, and **Add filter**
  2. Enter the following query:  
       
     Explanation of the query:
     + - QRcode = “”   
         This query will ensure that the list will not filter the query if there is no value in the **QRcode**.
       - or Product.ProductCode = QRcode   
         Otherwise if the **QRcode** is not empty, we will filter based on the **Product Code** (in this exercise, the QR codes are based on a unique Product Code attribute).
  3. Mark the **Products** page to be accessible by Anonymous users by selecting the screen and ticking “**Anonymous**” in the screen properties.  
     
  4. **Update** your application by clicking the 1-click-publish button.

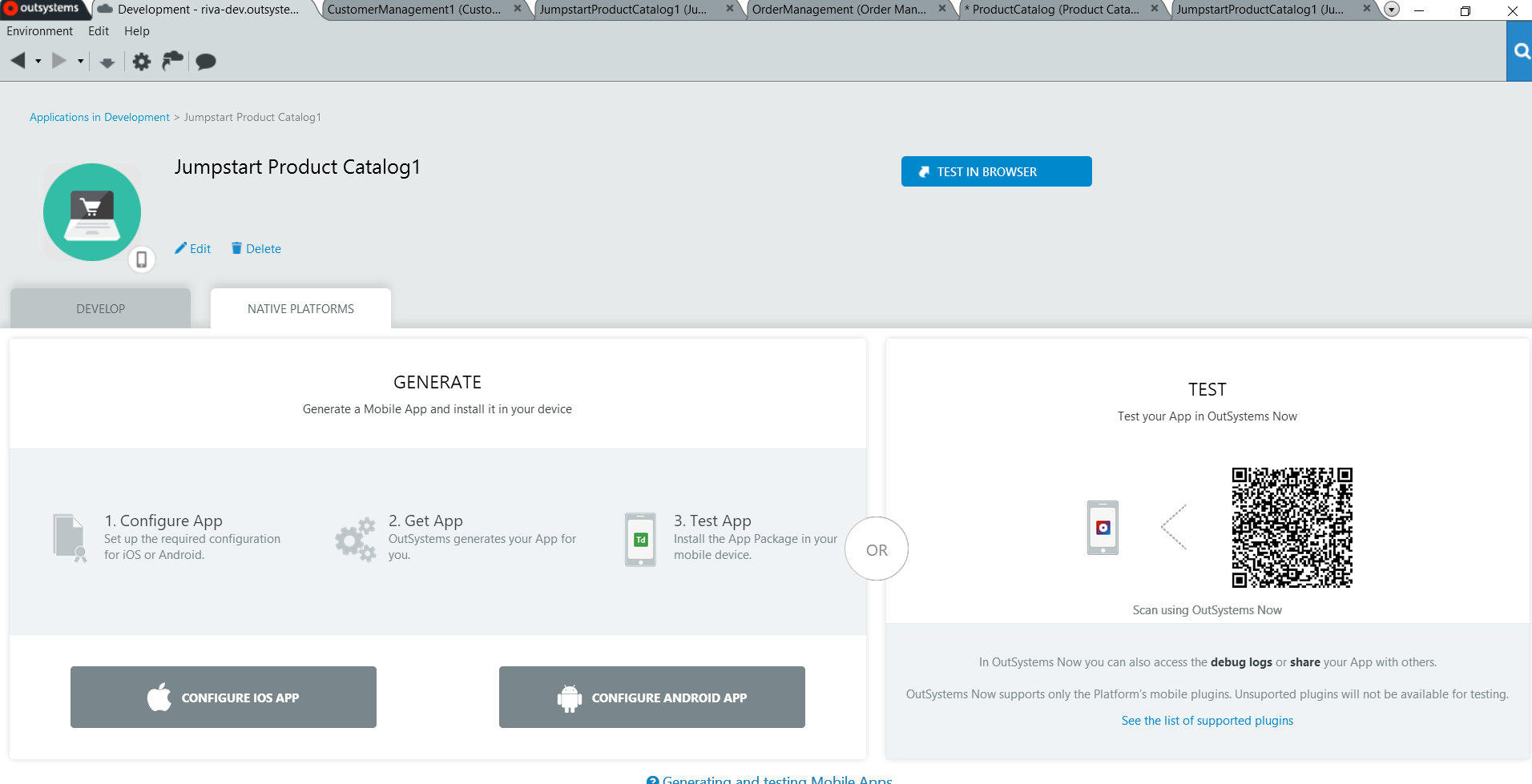
# Test your mobile app in your phone

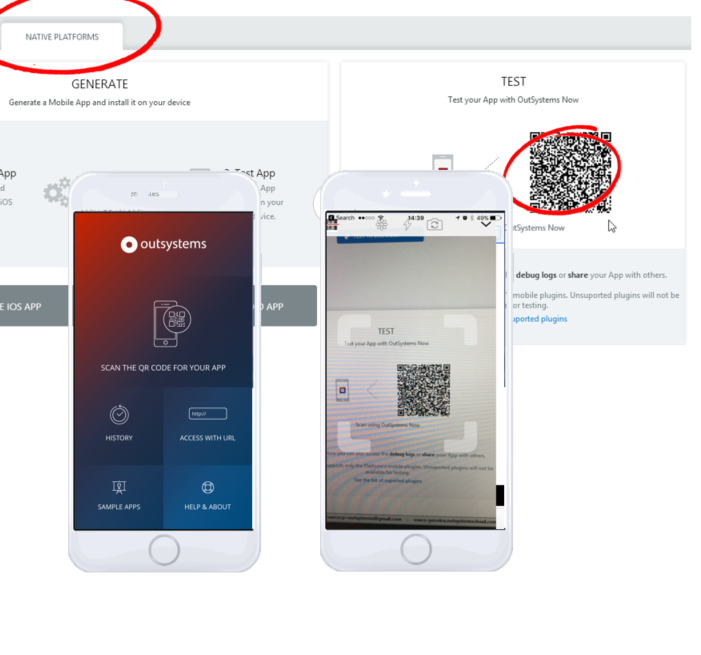
|  |
| --- |
| We now have a list of products. But how can we filter the products based on a barcode value that we scanned using the mobile camera?  Note; the barcode scanner works for both QR code and barcodes! |

* 1. Go to the **Development** Tab, and open your mobile application’s detail page.  
     If it’s not open already, you can also search for it.

* 1. Click **Native Platforms** Tab. If not expanded yet, expand the right side of the screen and click **TEST IN OUTSYSTEMS NOW.**

  
  
*Tip: On the left part of the screen, you can also generate the Android APK, and the IOS installer (needs the iOS developer certificate) to install it in your device.*

* 1. In your **iOS or Android device**, download **OutSystems Now** from the app store.  
     
  2. In the **OutSystems Now** app, scan the QR code on your screen.  
     This will navigate to your app and simulate your app being installed in your phone.   
     

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? It’s time to stand up, stretch, and start scanning the QR Codes pasted on the walls!

Congratulations! You have completed this exercise.

**Bonus exercise:**If you would like to consume the REST API you have exposed in the previous exercise, you can find the steps here: <https://www.youtube.com/watch?v=RnOo4Ap7Oto&t=0s&list=PLaxrSw3Eft4FziLc6fNPKjPhkHrroD4bc&index=2>