



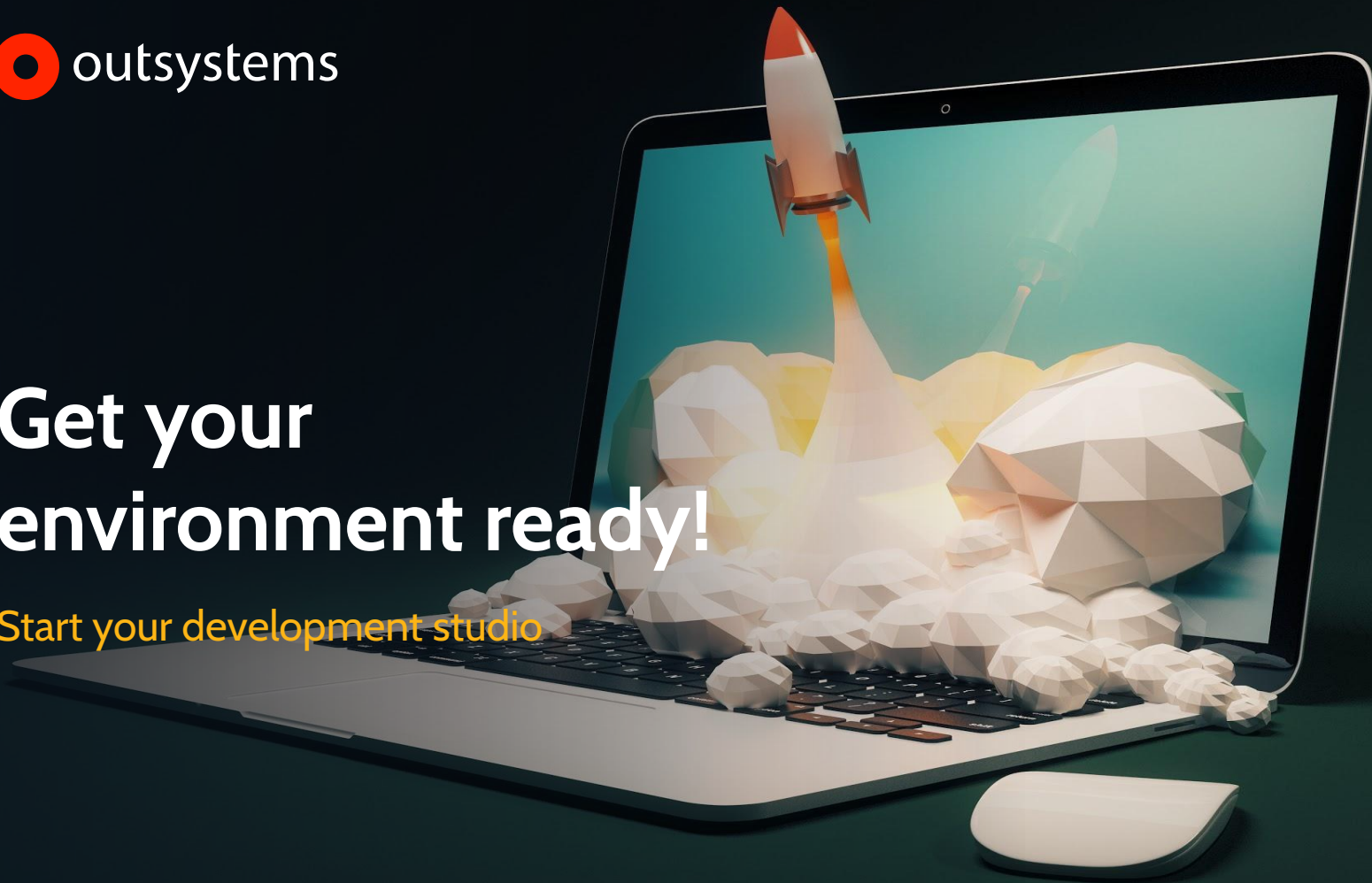
Build your own mobile app

OutSystems Mobile - Exercise 2



Get your environment ready!

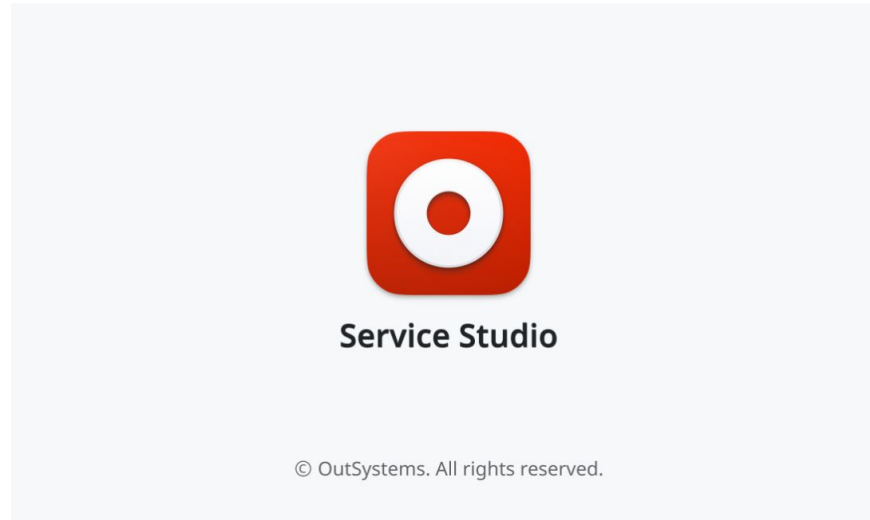
Start your development studio

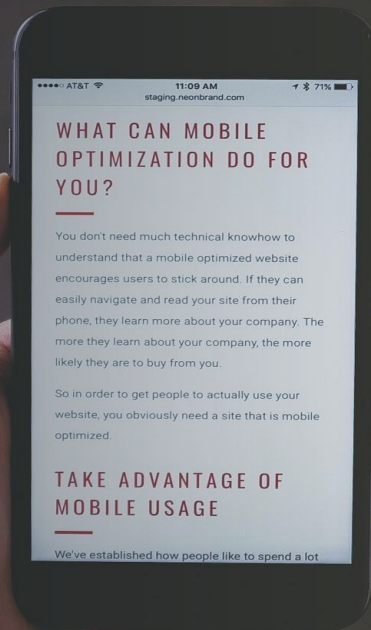


Development Studio

Open and start

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





Section 1

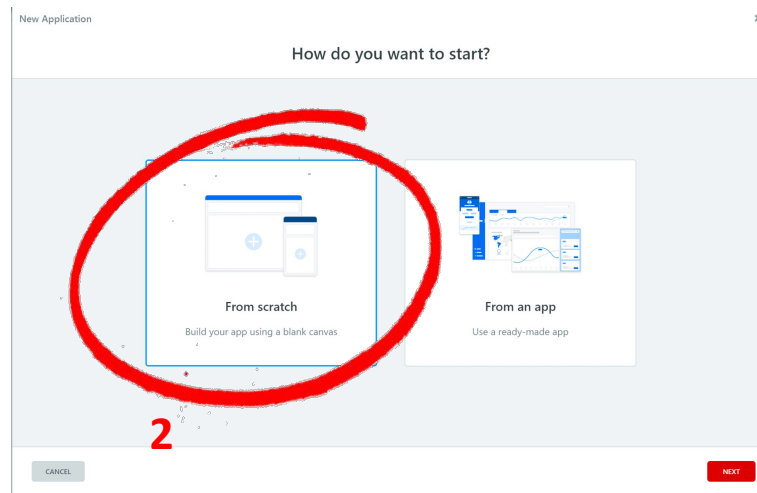
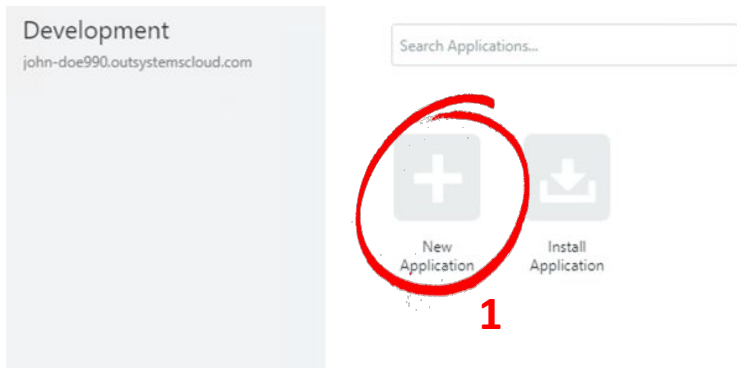
Mobile Application Foundation

- | 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)

Section 1 > A > 1. Starting Point

Create a new mobile app

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

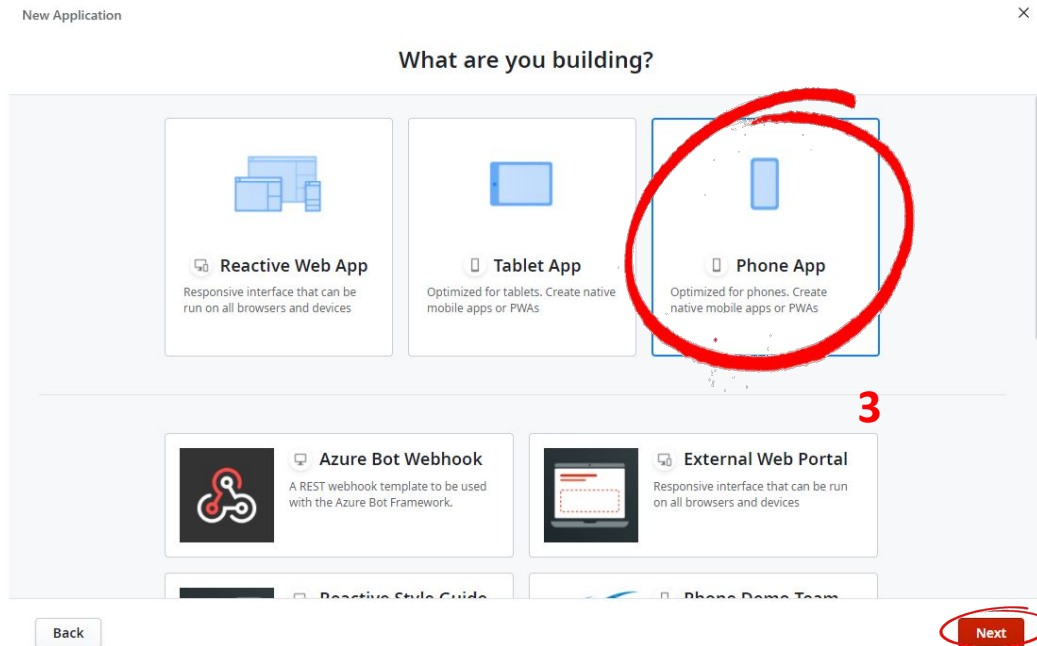


Section 1 > A > 2. Create a New App

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.



Section 1 > A > 3. Application Name & Icon

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”

The image shows two sequential screenshots of the 'New Application' form. The first screenshot shows the 'Fill in your app's basic info' section with the 'Product Catalog' name entered in the 'Product Catalog' field (marked with a red circle and '1'). Below this is a color palette and an 'Upload icon' button (marked with a red circle and '2'). A red arrow points from the 'Upload icon' button to a preview of the selected icon, 'Online-Shopping -icon.png'. The second screenshot shows the same form with the 'Product Catalog' name and 'Description' fields filled. The 'Upload icon' button is now highlighted with a red box (marked with '3'), and the 'Create App' button at the bottom right is highlighted with a red circle (marked with '4').

Section 1 > A > 4. Module Creation

Create Phone App Module

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

[Back to applications](#)



Product Catalog

[Edit](#) [Delete](#) [Download](#) [Convert To Service](#) [Test In Browser](#)

Develop [Distribute](#)

Modules

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

Dependencies

Other applications that are referenced from this application.

Module name

ProductCatalog

Choose module type

Phone App

Create Module

Cancel

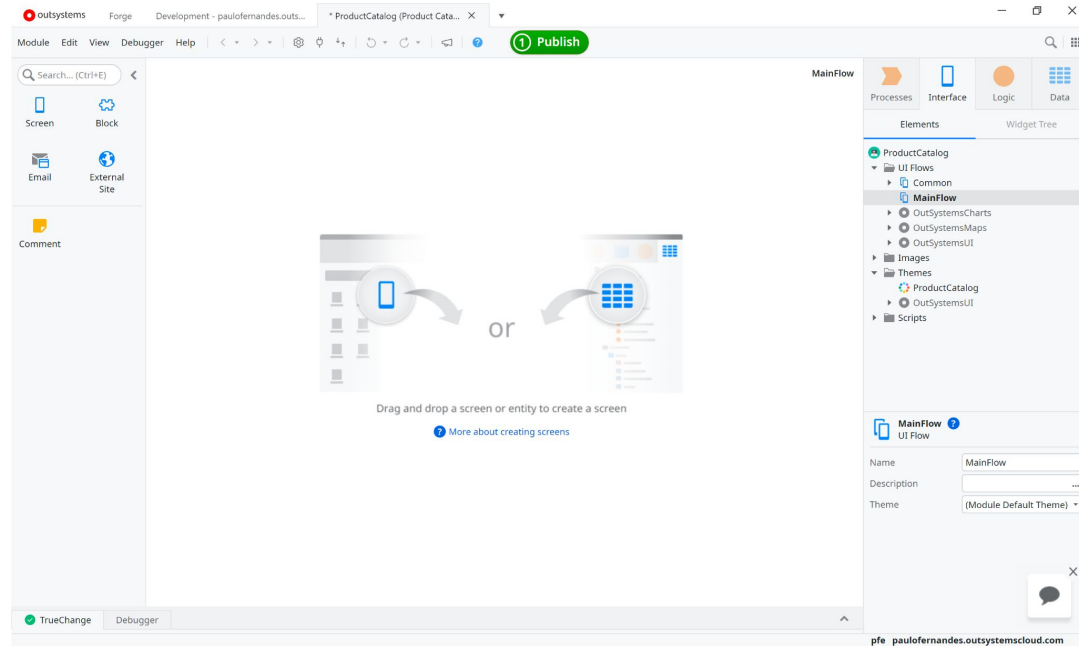
1

2

Section 1 > A > 5. Module Creation

Create Phone App Module


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

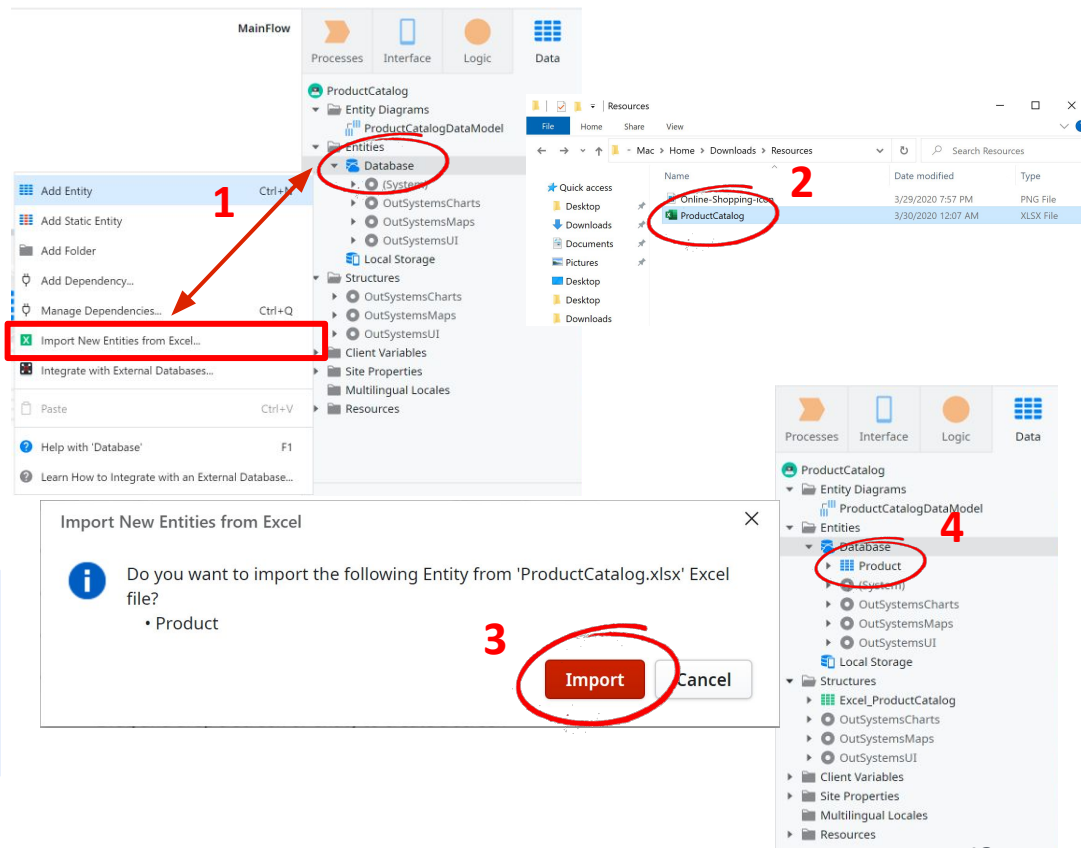


- | Let's start by creating our data model. In Outsystems tables are known as **Entities** while columns are known as **attributes**.
- | In this part, we will create a Product entity, by quickly importing an Excel file. This process is also known as **Bootstrap**.

Section 1 > B > 1. Create Product Entity

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**).



The screenshot illustrates the process of creating a product entity in the OutSystems IDE. It is divided into four numbered steps:

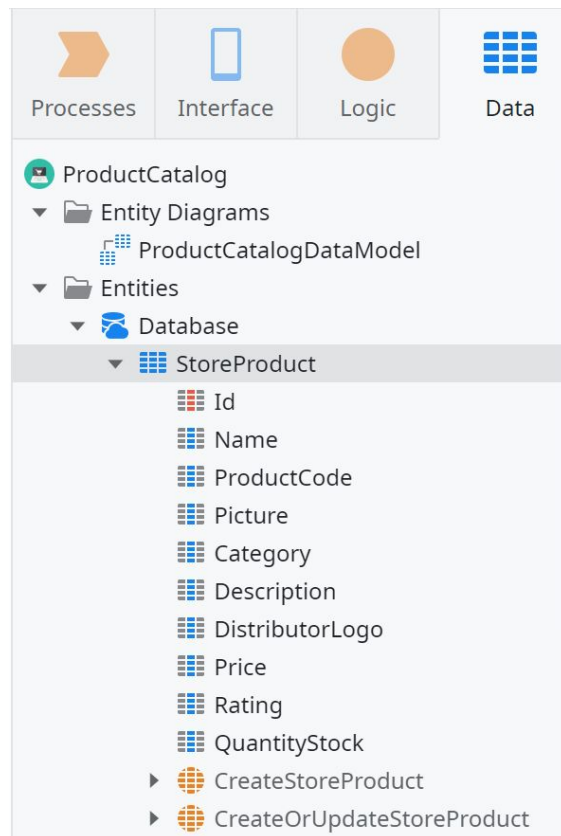
- Step 1:** The user navigates to the **Data** tab in the IDE. A right-click context menu is open over the **Database** folder, and the option **Import New Entities from Excel...** is highlighted with a red box and a red arrow labeled '1'.
- Step 2:** A file explorer window shows the **Resources** folder. The file **ProductCatalog.xlsx** is selected, highlighted with a red circle and labeled '2'.
- Step 3:** A confirmation dialog box titled **Import New Entities from Excel** is displayed. It asks, "Do you want to import the following Entity from 'ProductCatalog.xlsx' Excel file?" and lists the entity **Product**. The **Import** button is highlighted with a red circle and labeled '3'.
- Step 4:** The IDE's entity diagram is shown. The **Product** entity has been successfully created under the **Database** folder and is highlighted with a red circle and labeled '4'.

Section 1 > B > 2. Rename Entity Name

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.



Section 1

B

3. Add Product Entity to Diagram


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

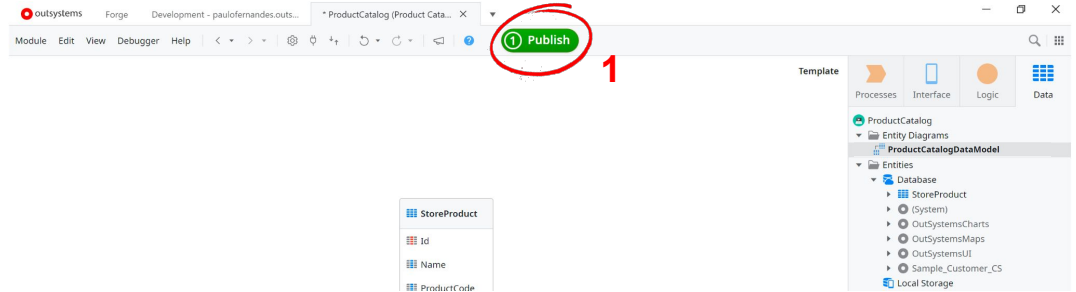



Entity Diagrams are a powerful way of quickly visualizing your data entities and their relationship

The screenshot shows the software interface for 'ProductCatalogDataModel'. At the top, there is a green 'Publish' button. Below it, there are four tabs: 'Processes', 'Interface', 'Logic', and 'Data'. The 'Data' tab is selected. In the right-hand pane, the 'Entity Diagrams' folder is expanded, and 'ProductCatalogDataModel' is highlighted with a red box and a red '1'. Below it, the 'Entities' folder is expanded, and the 'Database' folder is expanded, with 'StoreProduct' highlighted by a red box and a red '2'. A red arrow points from the 'StoreProduct' entity in the right pane to the 'StoreProduct' entity in the left pane. The left pane shows the 'StoreProduct' entity with the following attributes: Id, Name, ProductCode, Picture, Category, Description, DistributorLogo, Price, Rating, and QuantityStock.

Section 1 > B > 4. Publish App

1. Press the **1-Click-Publish** () 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

The screenshot shows the OutSystems IDE interface for a project named 'ProductCatalog'. The top navigation bar includes 'Module', 'Edit', 'View', 'Debugger', and 'Help'. A red circle highlights the 'Open in browser' button in the top right corner. The main workspace displays the 'ProductCatalog' data model with a list of attributes: Id, Name, ProductCode, Picture, Category, Description, DistributorLogo, Price, Rating, and QuantityStock. The right-hand side shows the 'ProductCatalogDataModel' structure, including 'Entity Diagrams', 'Entities', 'Database', 'Local Storage', and 'Structures'. The bottom status bar shows the deployment progress: 'Uploading', 'Compiling', 'Deploying', and 'Done'. The 'Done' step indicates that 'ProductCatalog' is now available at 'https://pauloferandes.outsystemscloud.com/ProductCatalog'.



Section 2

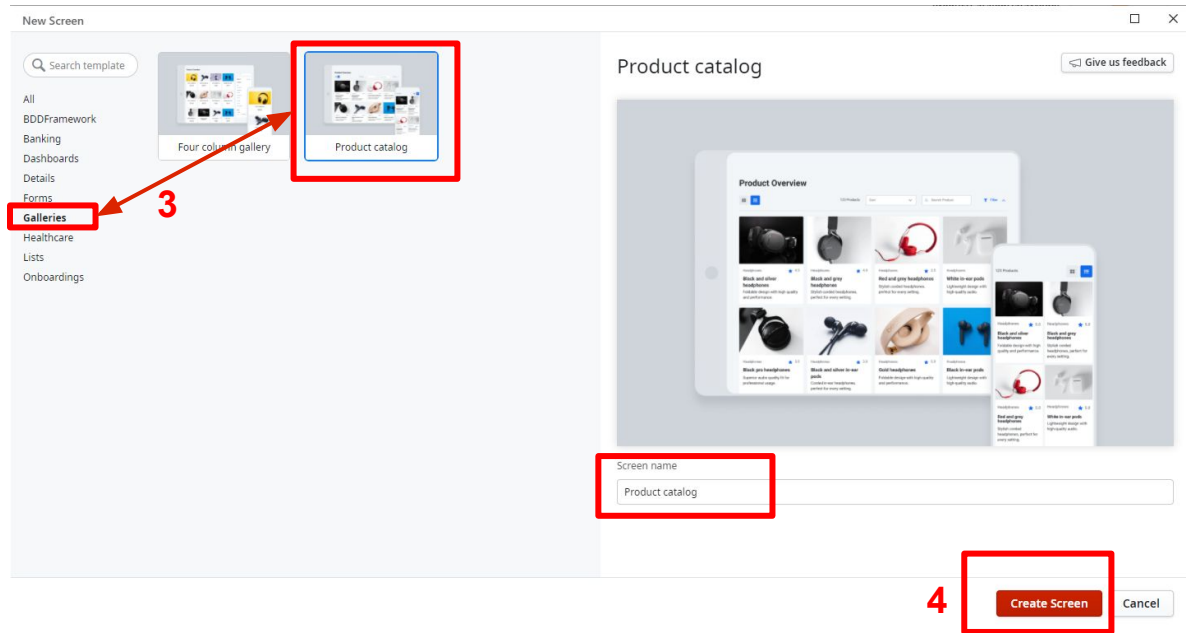
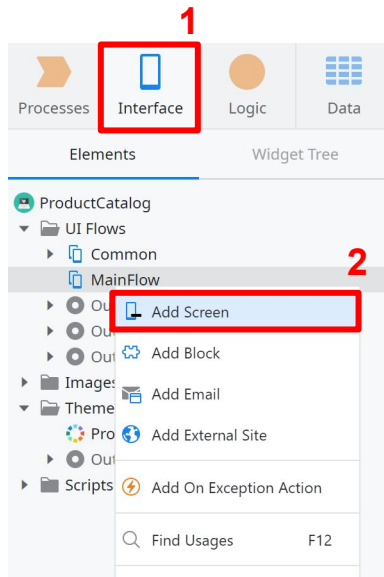
Product Gallery



- | 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**.

Section 2 > A > 1. Create a new Screen

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**

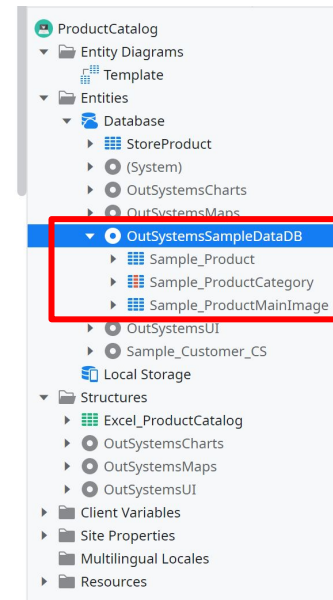
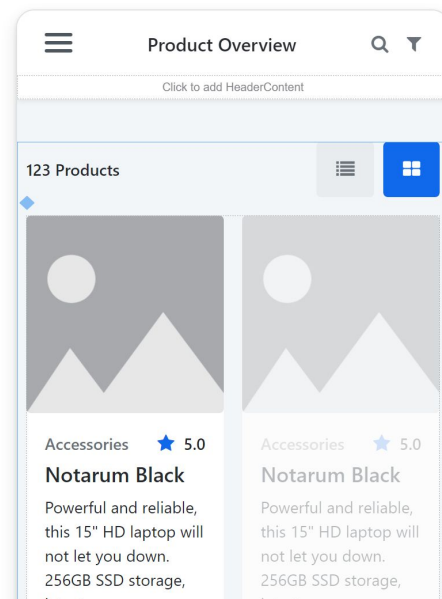
Section 2 > A > 2. Review Sample Data

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

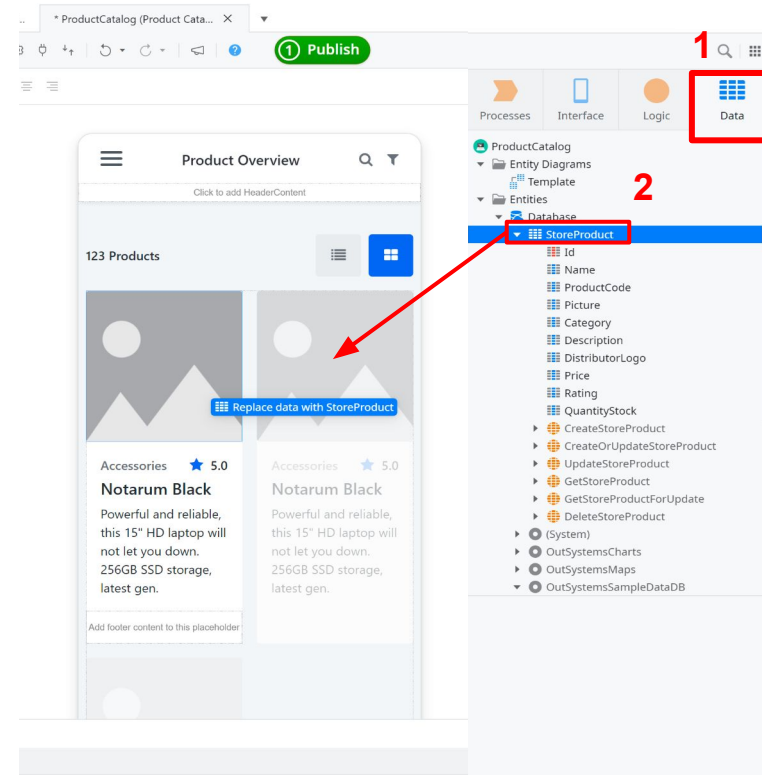


- | The Outsystems platform provides screen templates to help developers create user interfaces easily. These Templates contain **UI design**, include **pre-created actions** to accelerate development of the screen actions (such as filters) and **pre-created sample data** so you don't have to start from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

Section 2 > B > 1. Replace 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.



Section 2 > B > 2. Add image

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.

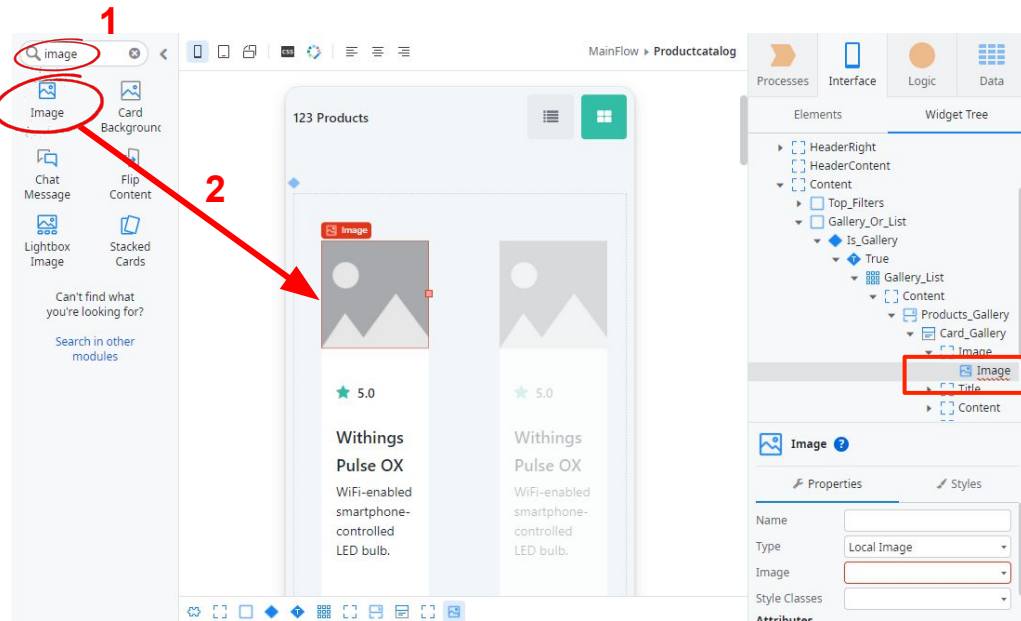


QUICK TIP

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

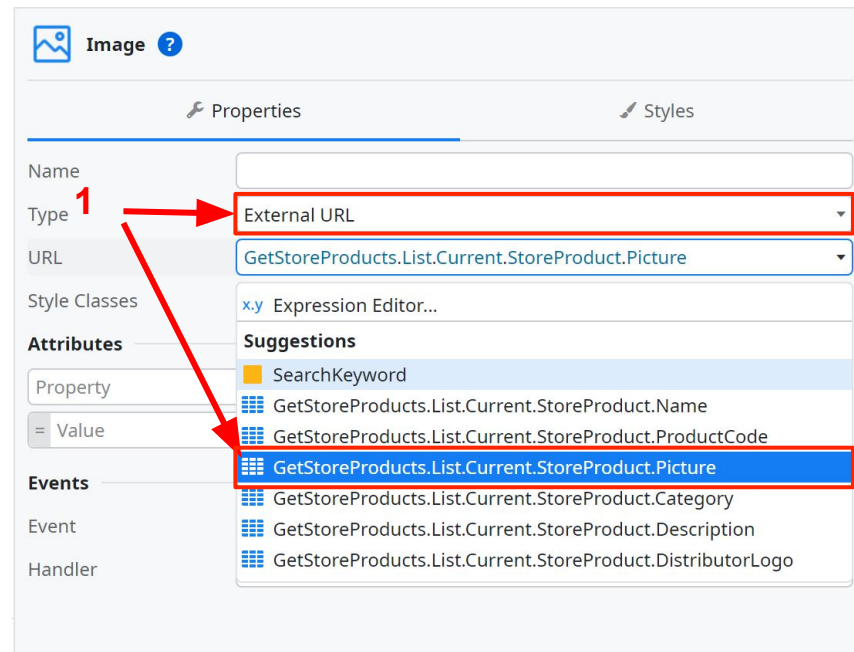
It can be manually opened by clicking the right tab “Widget Tree”

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



Section 2 > B > 3. Configure image properties

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



The screenshot shows the 'Image' properties panel. The 'Type' dropdown is set to 'External URL'. The 'URL' dropdown is set to 'GetStoreProducts.List.Current.StoreProduct.Picture'. The 'Suggestions' list includes 'SearchKeyword', 'GetStoreProducts.List.Current.StoreProduct.Name', 'GetStoreProducts.List.Current.StoreProduct.ProductCode', 'GetStoreProducts.List.Current.StoreProduct.Picture', 'GetStoreProducts.List.Current.StoreProduct.Category', 'GetStoreProducts.List.Current.StoreProduct.Description', and 'GetStoreProducts.List.Current.StoreProduct.DistributorLogo'. The 'GetStoreProducts.List.Current.StoreProduct.Picture' suggestion is highlighted with a blue background and a red border. A red arrow points from the 'Type' dropdown to the 'External URL' option, and another red arrow points from the 'URL' dropdown to the 'GetStoreProducts.List.Current.StoreProduct.Picture' option.

Section 2

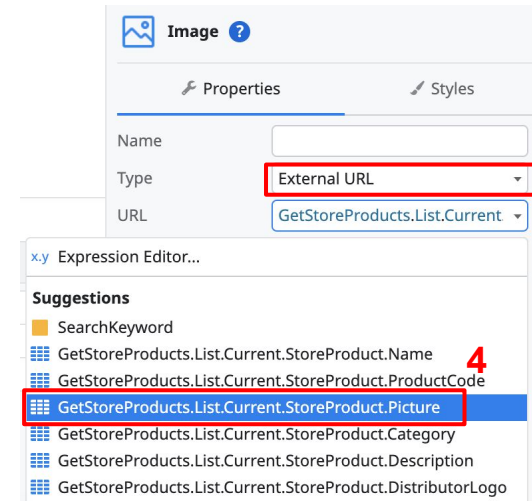
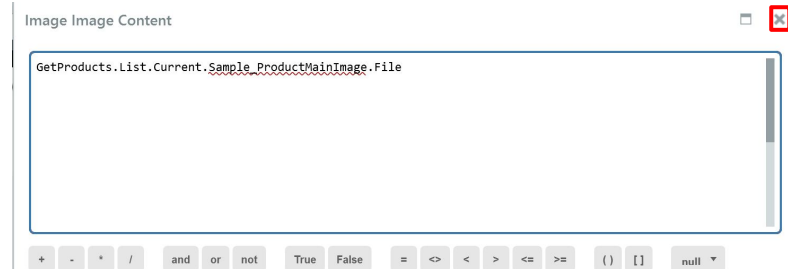
B

4. Fix Error

Can't identify 'Sample_ProductMainImage' element in expression.

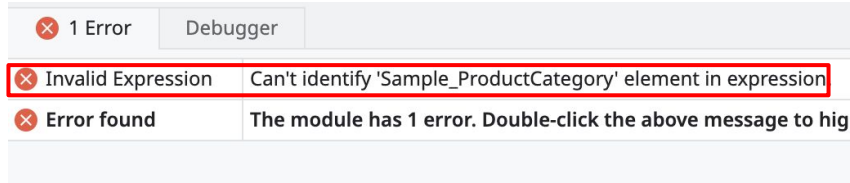
2 Errors	
Debugger	
Invalid Expression	Can't identify 'Sample_ProductMainImage' element in expression.
Invalid Expression	Can't identify 'Sample_ProductCategory' element in expression.
Errors found	The module has 2 errors. Double-click the above messages to high

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**.

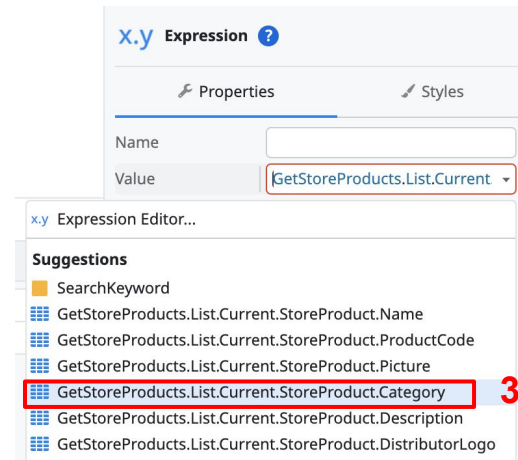
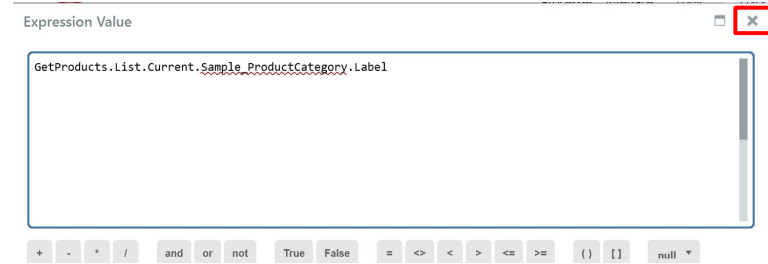


Section 2 > B > 5. Fix Error

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.**



Section 2 > B > 6. Make Screen Anonymous

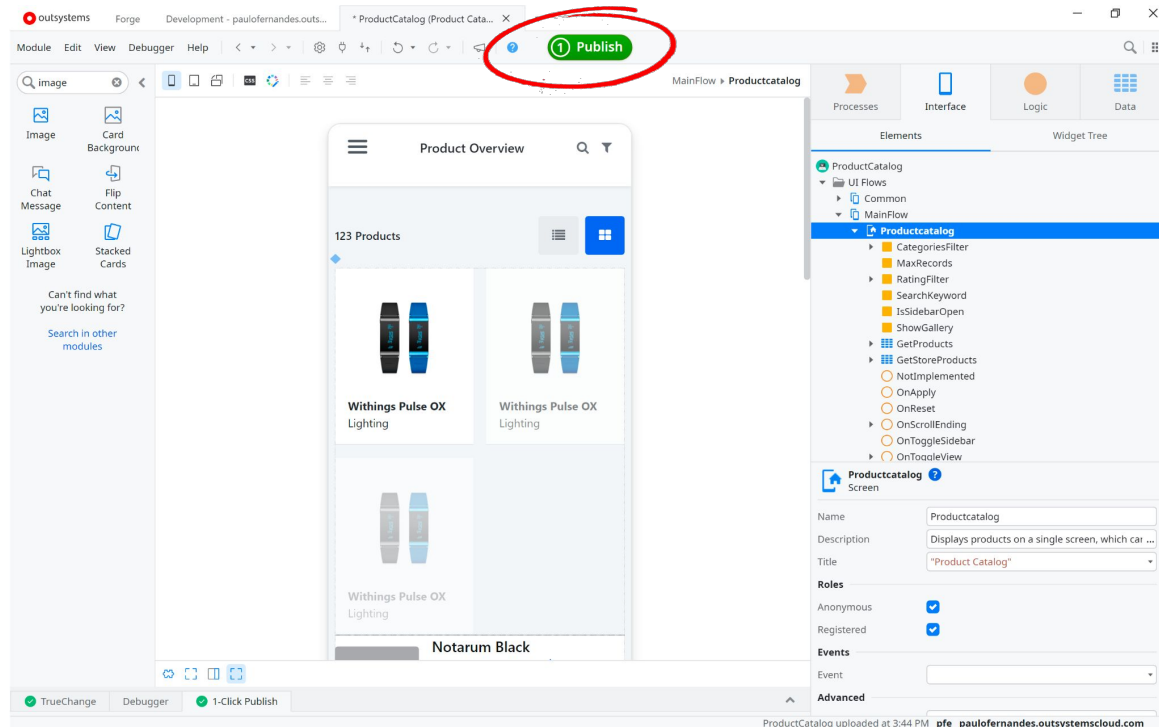
1. Click on the "Productcatalog" screen;
2. Mark the **Product Catalog** screen to be accessible by **Anonymous** users.

The screenshot displays the SAP Fiori Designer interface. At the top, there are four tabs: Processes, Interface, Logic, and Data. Below these are two main sections: Elements and Widget Tree. In the Elements section, the 'Productcatalog' screen is highlighted with a red box and a red number '1'. The Widget Tree section shows a hierarchical structure of components under 'MainFlow', including 'Productcatalog', 'CategoriesFilter', 'MaxRecords', 'RatingFilter', 'SearchKeyword', 'IsSidebarOpen', 'ShowGallery', 'GetProducts', 'GetStoreProducts', 'NotImplemented', 'OnApply', 'OnReset', 'OnScrollEnding', 'OnToggleSidebar', 'OnToggleView', 'OutSystemsCharts', and 'OutSystemsMaps'. Below the Widget Tree, the 'Productcatalog' screen configuration is shown. The 'Name' field is 'Productcatalog', the 'Description' is 'Displays products on a single screen, which car ...', and the 'Title' is 'Product Catalog'. Under the 'Roles' section, the 'Anonymous' checkbox is checked and highlighted with a red box and a red number '2', while the 'Registered' checkbox is also checked. The 'Events' section has an empty dropdown menu, and the 'Advanced' section is partially visible at the bottom.

Section 2 > B > 7. 1-Click Publish

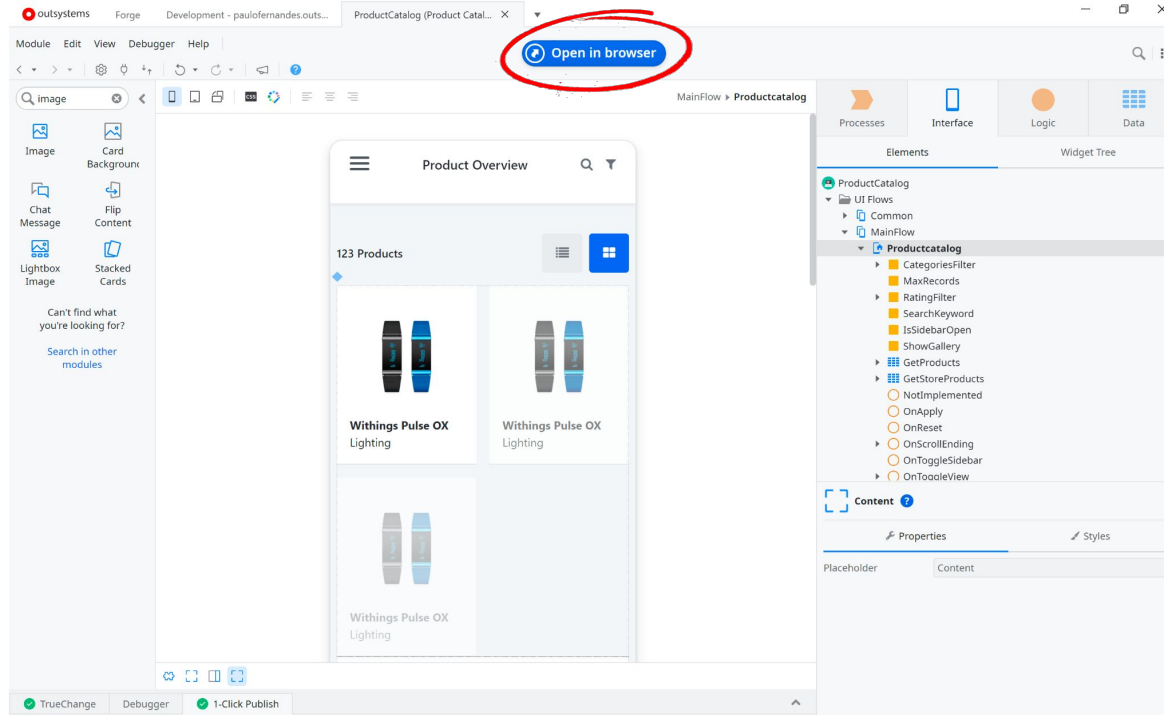
Deploy and Test it

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** () and test your app.



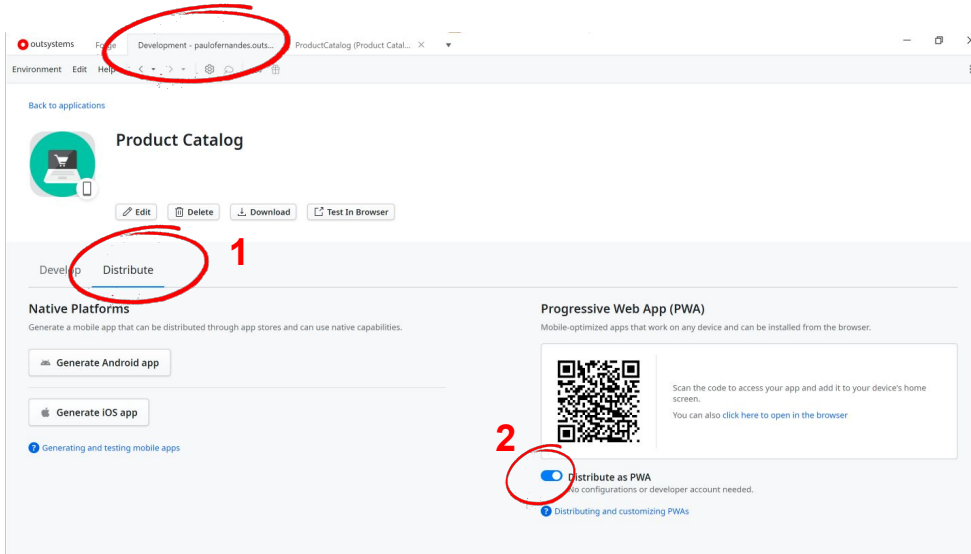


Mobile Application Test

Testing on Android & iOS

Test it on Your Device

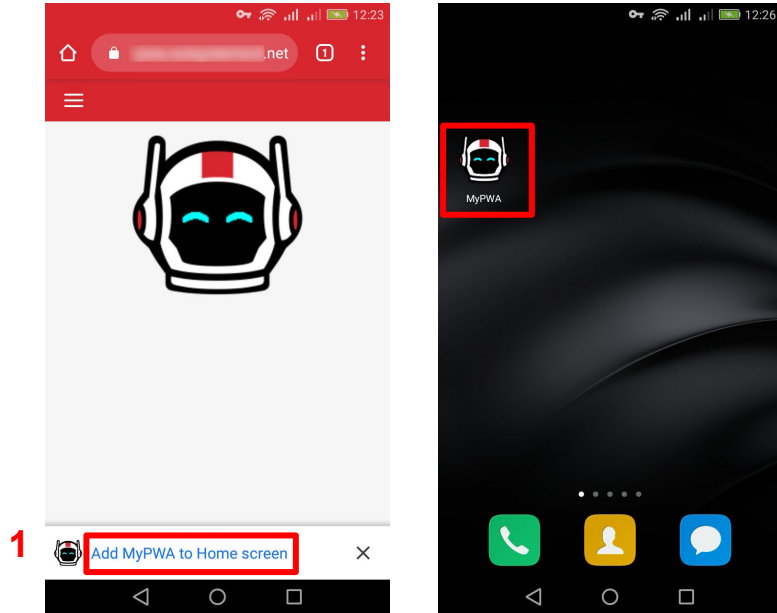
1. Navigate to the **Distribute** tab in Service Studio.
2. Enable **Distribute as PWA**
3. **Scan QR Code** from using your mobile phone
4. Add App to homescreen (See next screen)



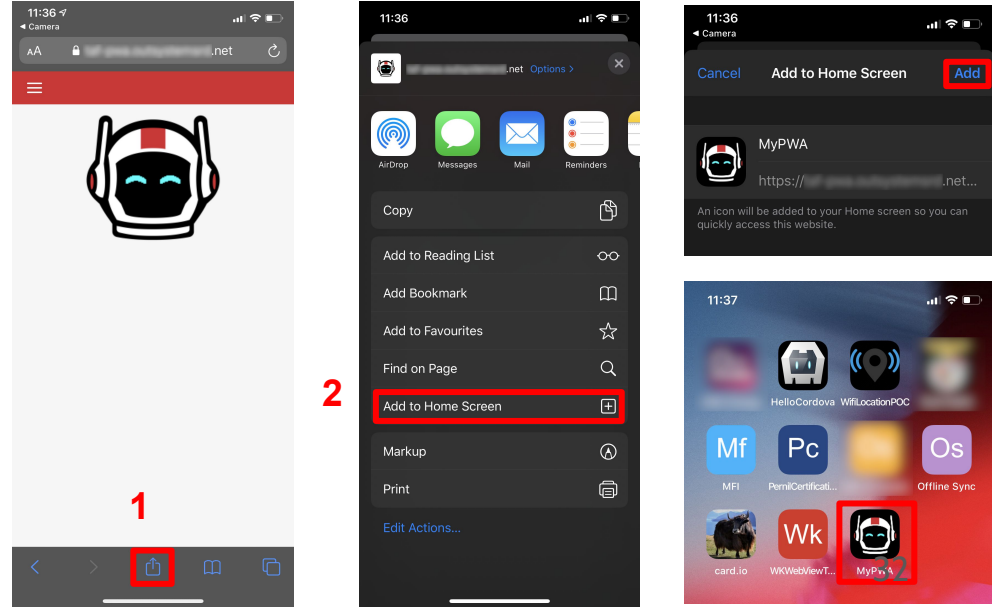
Test it on Your Device



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



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



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?



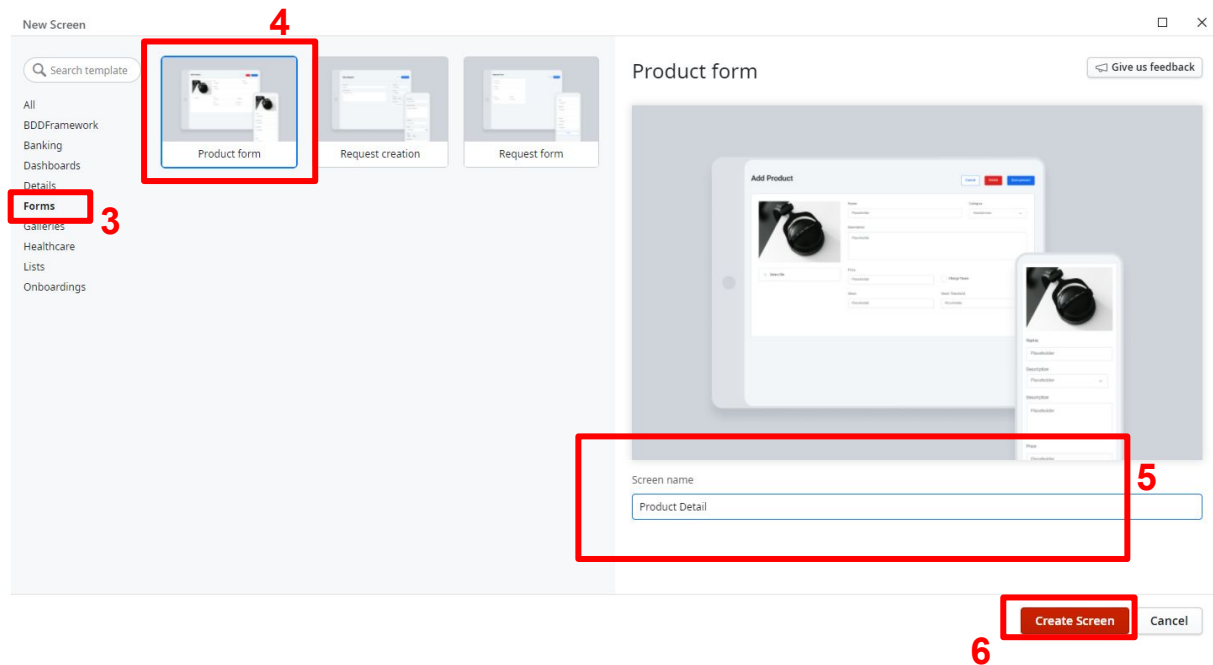
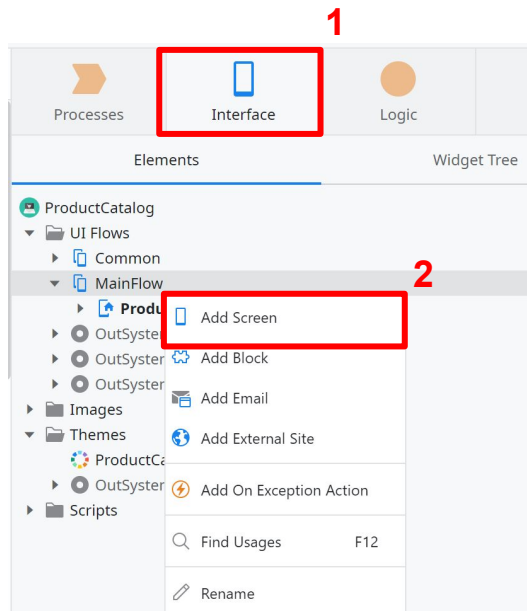
Section 3

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.

Section 3 > A > 1. Create a new Screen

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**.

Section 3 > A > 2. Configure Product Detail Screen

Set it as anonymous

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

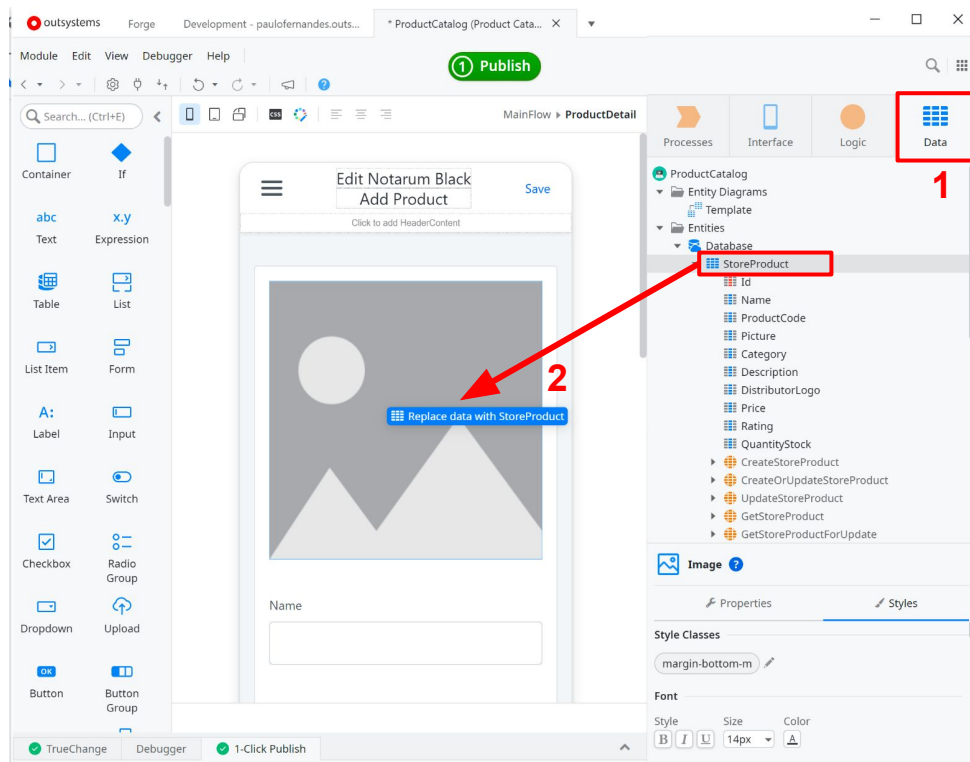
The screenshot displays the SAP Fiori Designer interface for configuring the 'ProductDetail' screen. The left pane shows a mobile device preview with a header 'Edit Notarum Black' and 'Add Product' button, and a 'Name' input field. The right pane shows the 'Widget Tree' with 'ProductDetail' selected (marked with a red box and '1'). Below it, the 'ProductDetail' screen properties are shown, with the 'Anonymous' role checked (marked with a red box and '2').

Category	Property	Value
General	Name	ProductDetail
	Description	Displays a product form that inclu ...
	Title	"Product Form"
Roles	Anonymous	<input checked="" type="checkbox"/>
	Registered	<input checked="" type="checkbox"/>

Section 3 > A > 3. Replace Sample Data

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.

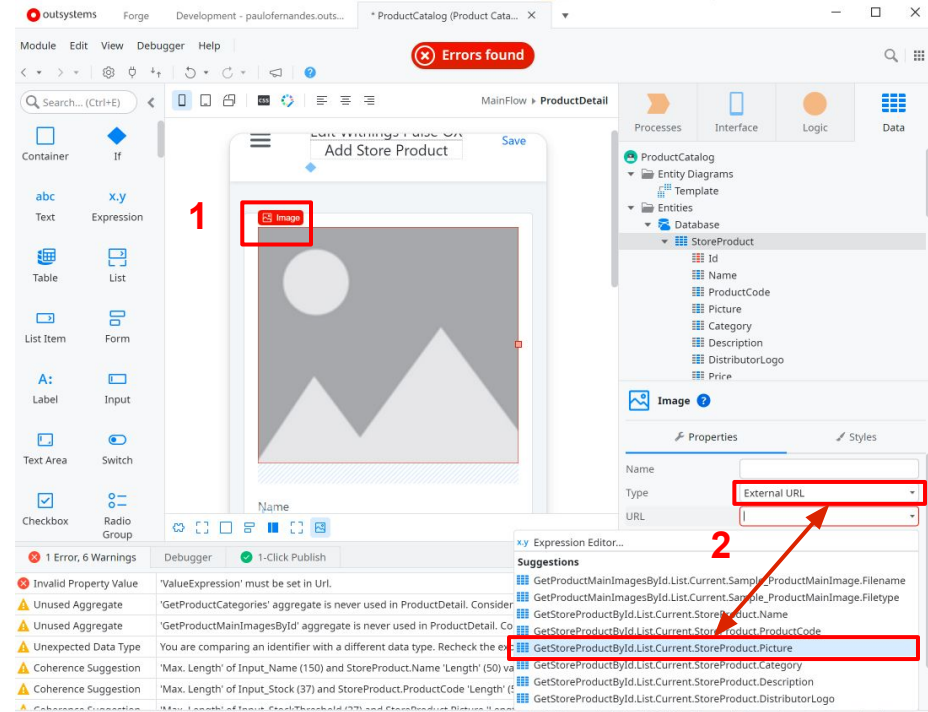


Section 3

A

4. Fix Image

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`.

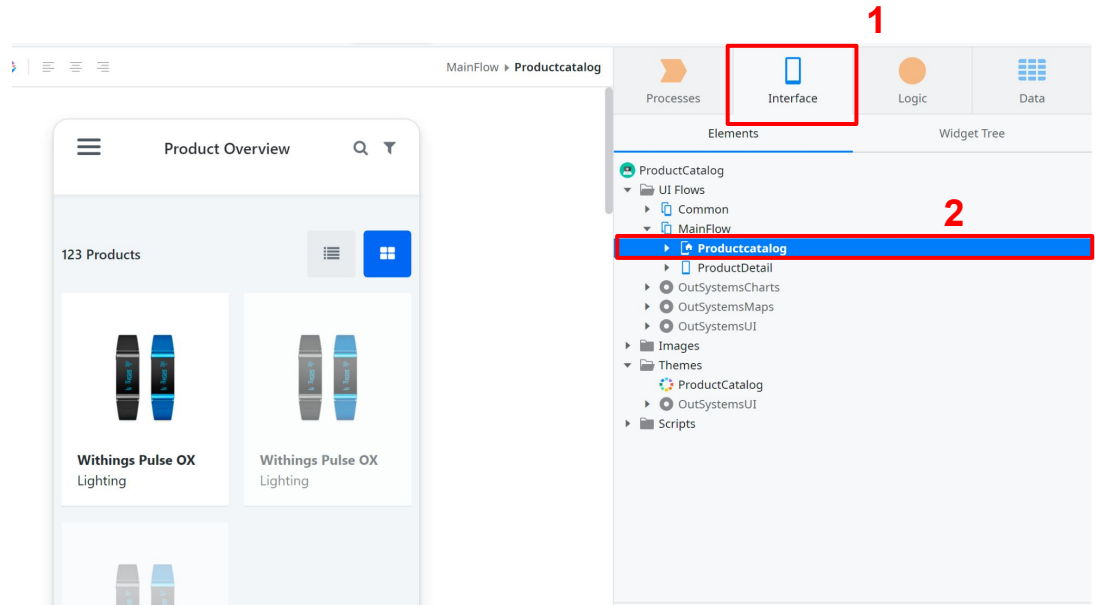


- | Now that we have replaced the Product Detail screen with the Product Entity data, we want to enable users to choose a product from the Product Gallery and see its details.

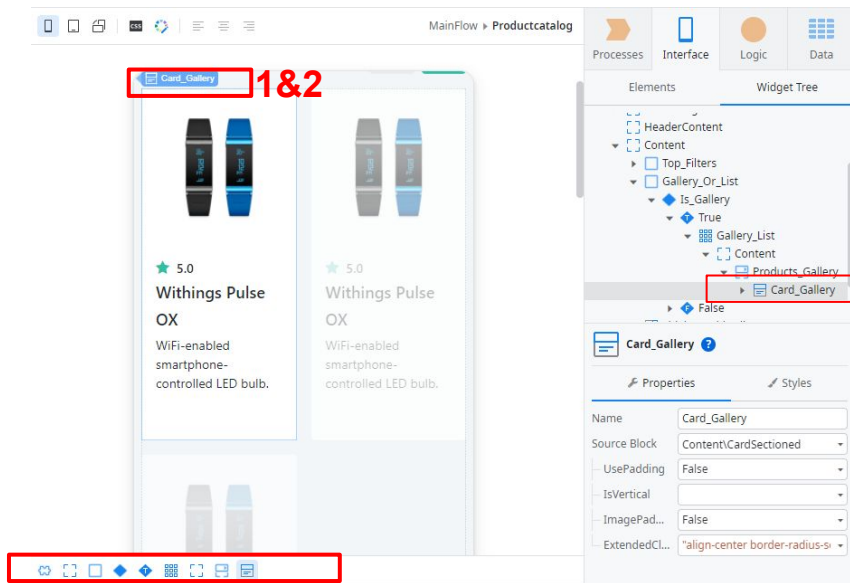
Section 3 > B > 1. Open Product Catalog Screen

Open the screen

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



Section 3 > B > 2. Link Gallery to Product Detail Screen

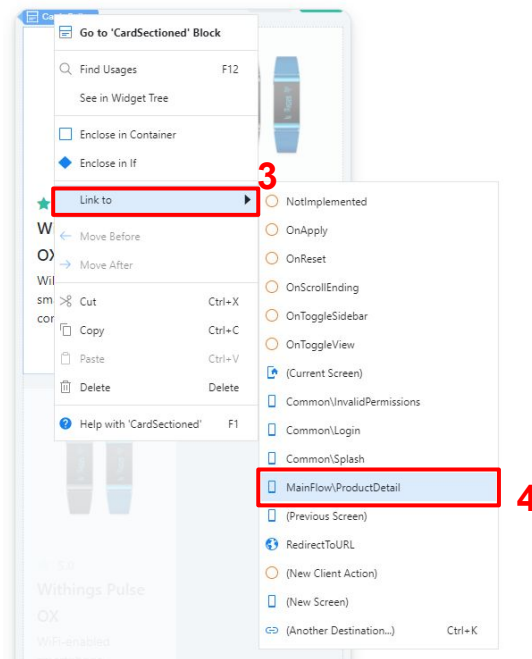


1. Select the **Card_Gallery** widget;



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.

Section 3

B

3. Link Image to Product Detail 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**.

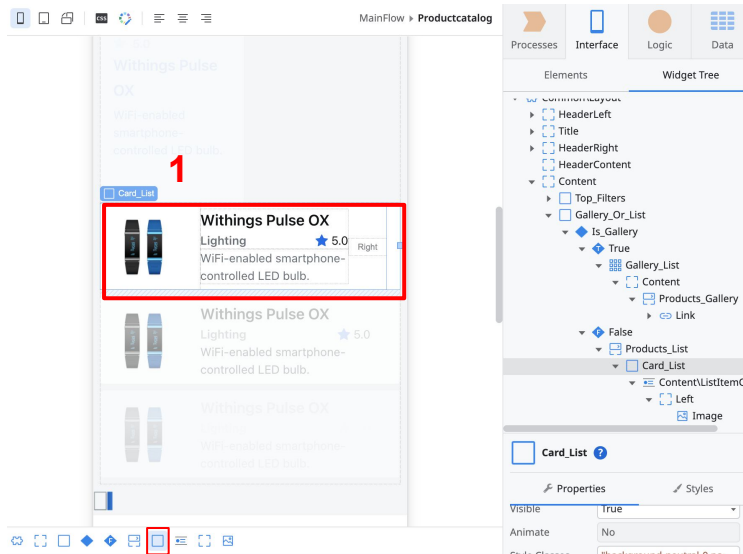
The screenshot shows a mobile app development IDE interface. The main canvas displays a product catalog with two product cards. The first card is titled "Withings Pulse OX" and features a star rating of 5.0. A red box labeled "1" highlights a "Link" widget at the top of the first product card. The right-hand side of the IDE shows the "Widget Tree" and "Properties" panels. The "Link" widget is selected, and its "On Click" event is configured to navigate to "MainFlow/ProductDetail". The "StoreProduct..." dropdown is open, and a red box labeled "2" highlights the suggestion "GetStoreProducts.List.Current.StoreProduct.Id" in the "Suggestions" list.

- | 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

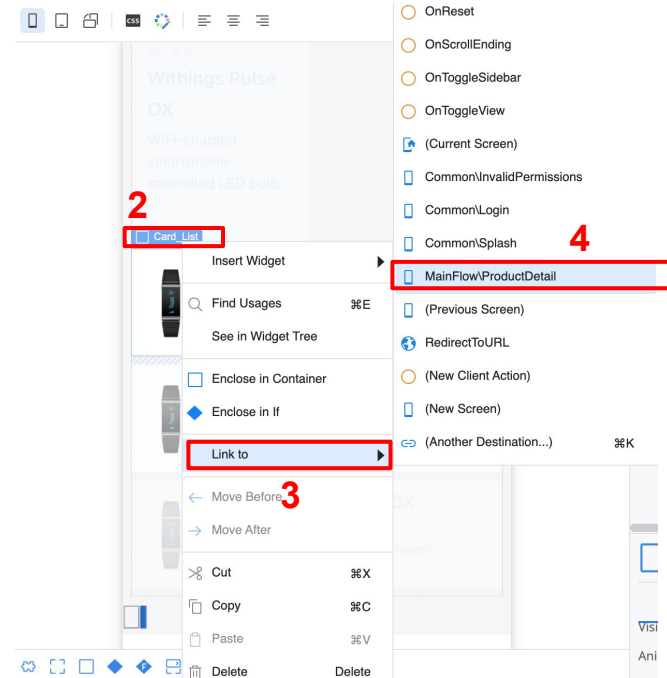


1. Scroll down until you see the Product List;
2. Select the **Card_List** widget;



QUICK TIP

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



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

Section 3 > C > 2. Link Image to Product Detail Screen

Assign Product Id

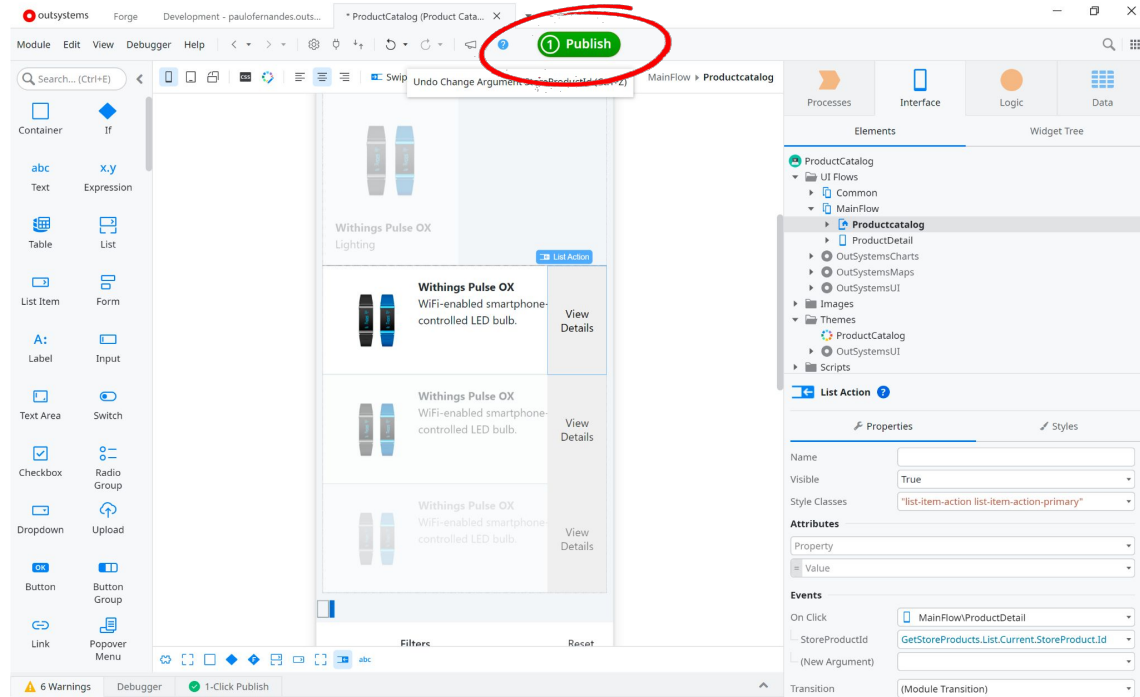
1. Make sure the **Link** widget is selected;
2. On the **On Click** event assign to the **ProductId** the value **GetStoreProducts.List.Current.StoreProduct.Id**.

The screenshot displays the Axure RP editor interface. The main canvas shows a mobile application screen with a list of 'Withings Pulse OX' products. A red box highlights the 'Link' widget icon in the top toolbar, with a red '1' next to it. The right-hand side of the editor shows the 'Widget Tree' and 'Properties' panels. In the 'Properties' panel, the 'On Click' event is set to 'MainFlowProductDet...', and a dropdown menu is open showing 'StoreProduct...'. A red box highlights the suggestion 'GetStoreProducts.List.Current.StoreProduct.Id' in the 'Suggestions' list, with a red '2' next to it. The 'Errors found' notification is visible at the top of the editor.

Section 3 > C > 3. 1-Click Publish

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

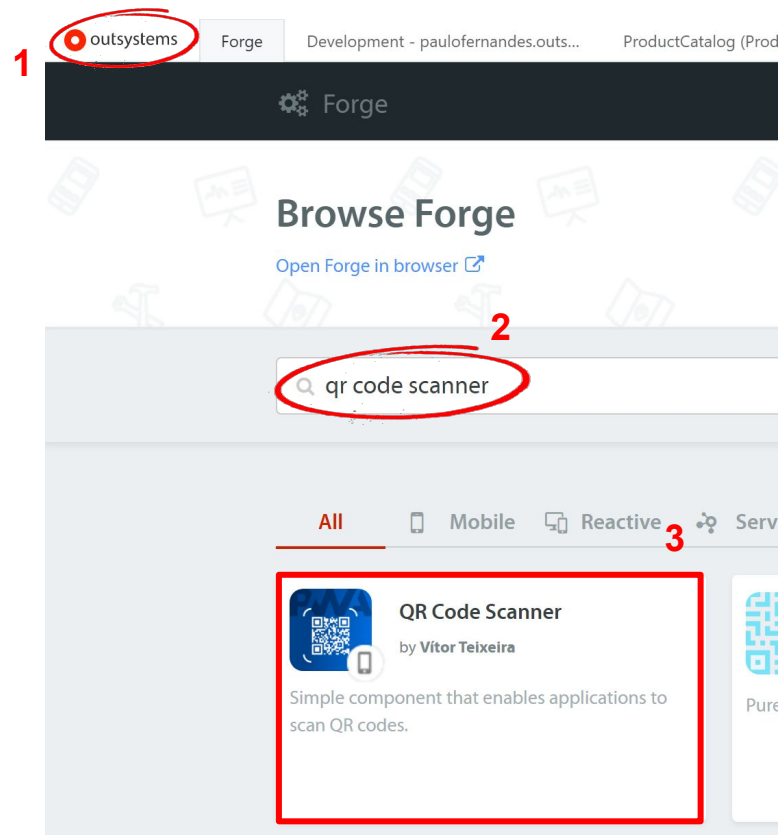


- | We now have a list of products. But how can we filter the products based on a QR Code, using the mobile camera?
- | The first thing we need to do is to install the [QR Code Scanner](#) plugin on our environment.

Section 4 > A > 1. Install a Forge Component

QR Code Scanner Plugin

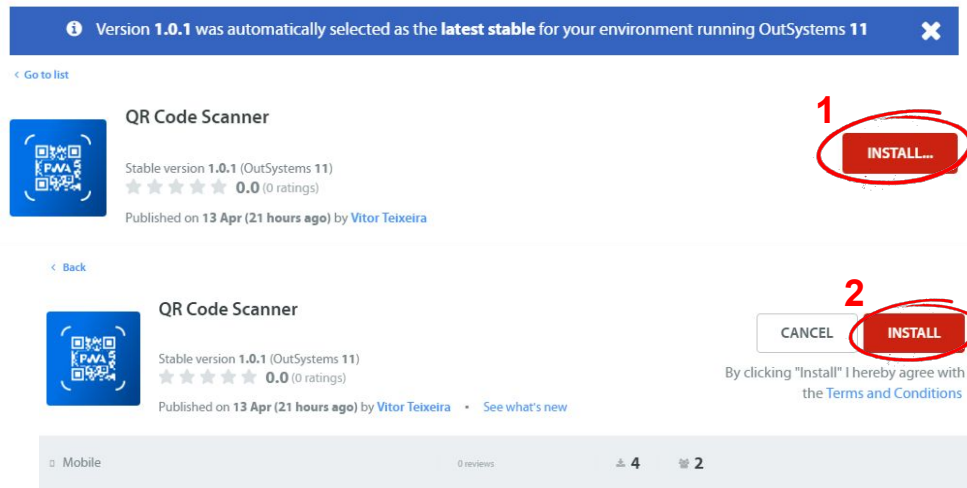
1. In Service Studio, navigate to the OutSystems tab and make sure you are logged in (or navigate to <https://www.outsystems.com/forge/>).
2. Search for the application “QR Code Scanner”
3. Click “QR Code Scanner” Plugin



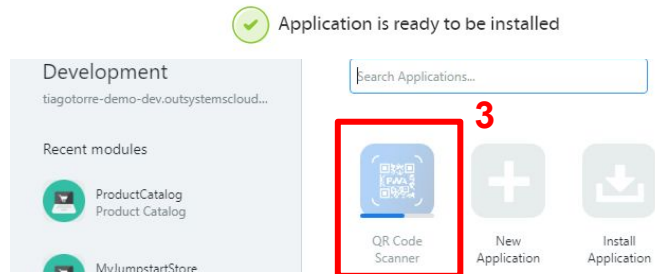
Section 4 > A > 2. Install a Forge Component

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.

Section 4 > B > 1. Add Forge Dependencies

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

The screenshot shows the 'Manage Dependencies' dialog in the Forge IDE. The search bar contains 'qr' (2). The 'QRCodeScanner' package is selected in the 'Producers' list (3). The 'ScanQRCode' element is selected in the 'Elements' list (4). The 'Apply' button is highlighted (5).

1. Manage Dependencies icon

2. Search for 'qr'

3. Select 'QRCodeScanner'

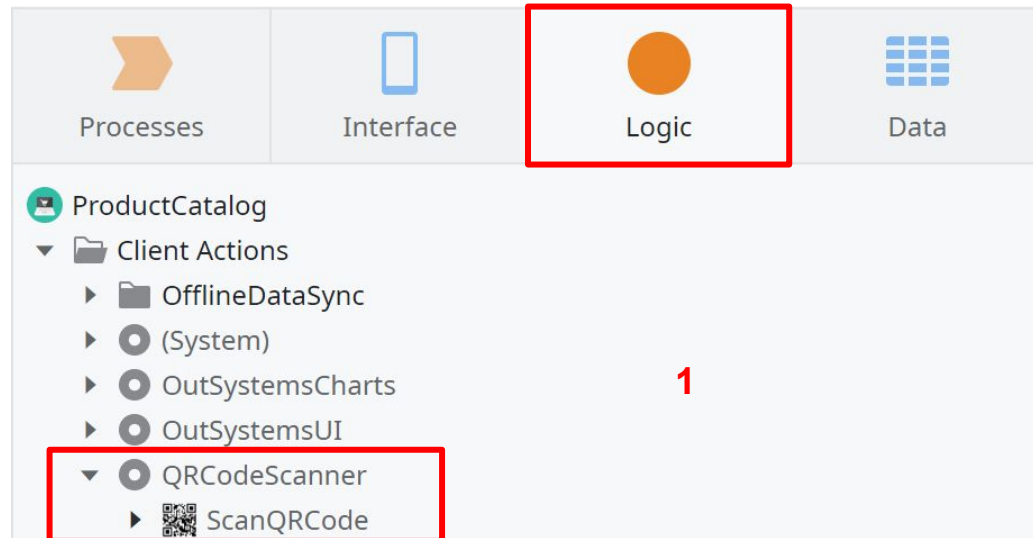
4. Select 'ScanQRCode'

5. Click 'Apply'

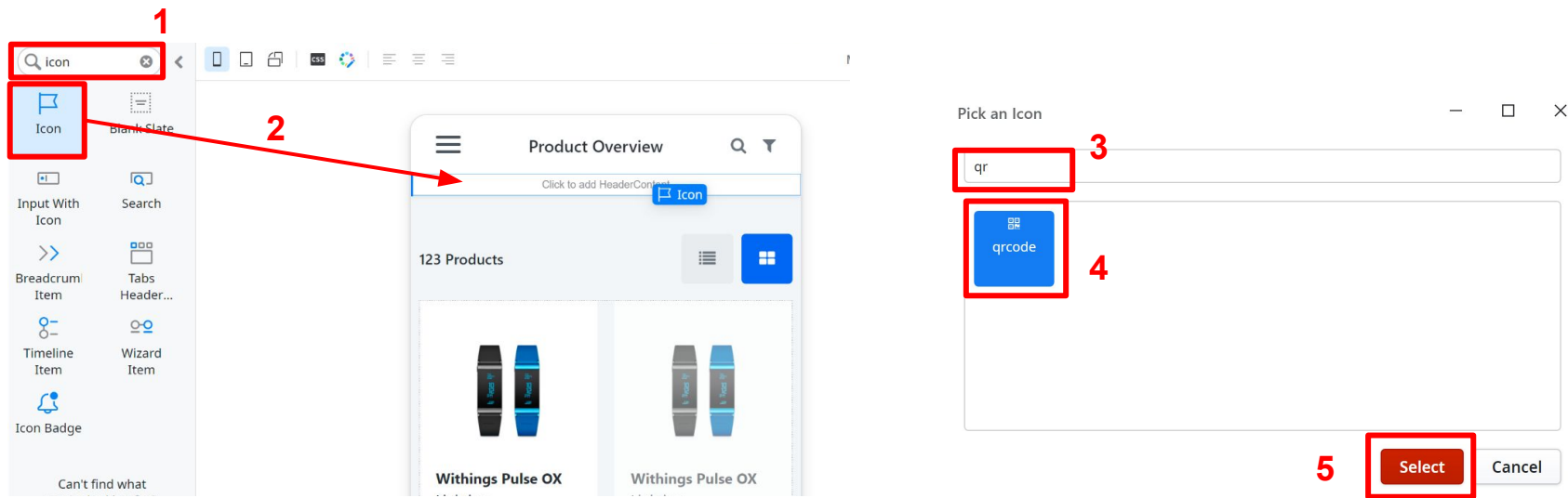
Section 4 > B > 2. Add Forge Dependencies

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



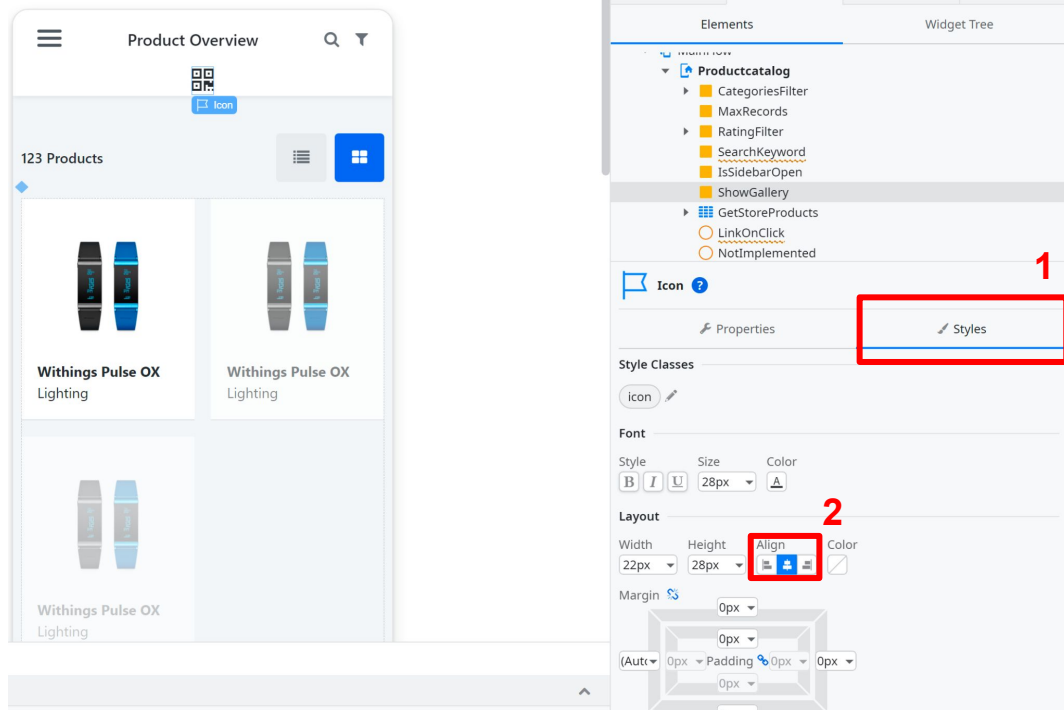
Let's create a button to trigger the QR Code.

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**.

Section 4 > B > 4. Center Icon on Page

New Client Action

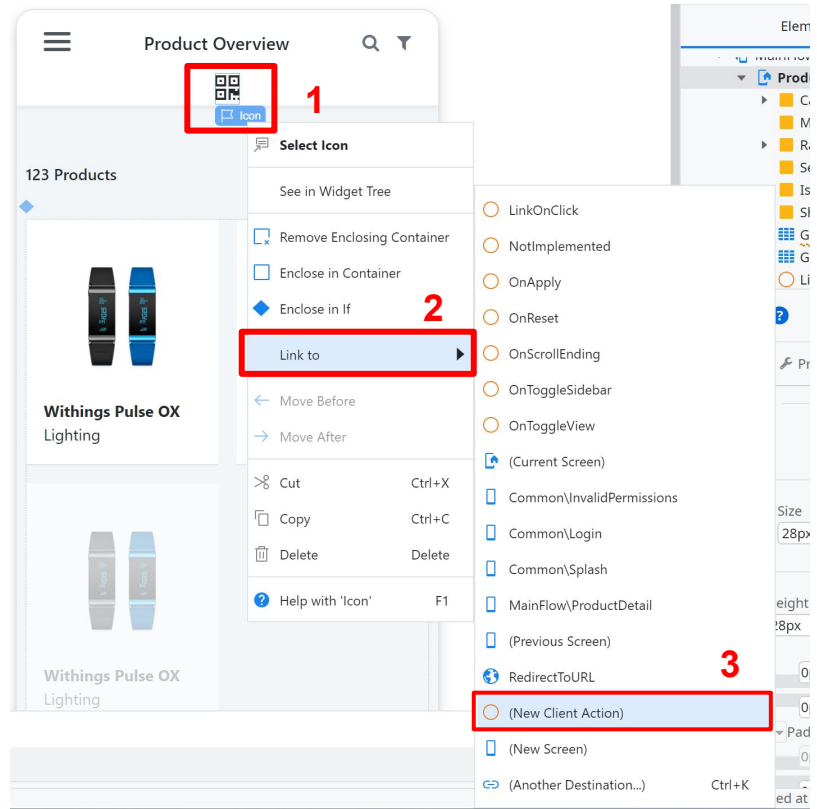
1. With the **icon** still selected, Click on the **Styles** tab
2. Align **Center**



Section 4 > B > 5. Link Icon to Client Action

New Client Action

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



Section 4 > B > 6. Open 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

The screenshot displays the SAP Fiori Designer interface. The main workspace shows a mobile application preview titled 'Product Overview' with a search bar and a list of '123 Products'. Two product cards are visible, both labeled 'Withings Pulse OX Lighting'. The right-hand side of the interface features a 'Widget Tree' and a 'Client Action' configuration panel. In the 'Widget Tree', the 'LinkOnClick' client action is highlighted with a red box. Below it, the 'Client Action' configuration panel is visible, showing the name 'LinkOnClick' and a description field. The status bar at the bottom right indicates 'Created by pfe' and 'Last modified by pfe at 9:29 PM'.

ScanQRCode Action

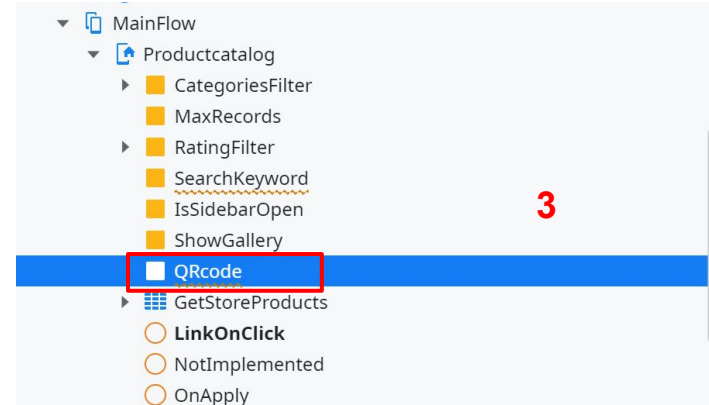
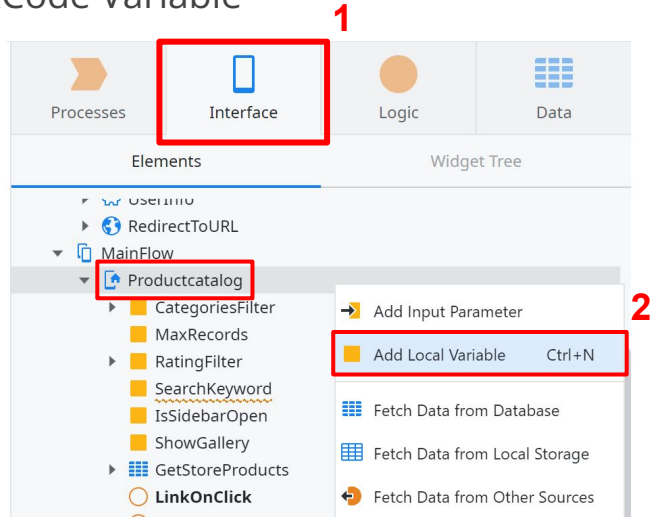
Look for the **ScanQRCode** action that you have added as a dependency earlier.

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.

The screenshot displays the SAP Fiori development tool interface. At the top, a green button labeled "1 Publish" is visible. Below it, the breadcrumb path "MainFlow > Productcatalog > LinkOnClick" is shown. The main workspace contains an action flow diagram with three steps: "Start" (a play button icon), "ScanQRCode" (a QR code icon), and "End" (a square icon). A red arrow labeled "2" points from the "ScanQRCode" action in the diagram to the "ScanQRCode" action in the right-hand pane. The right-hand pane is divided into four tabs: "Processes", "Interface", "Logic", and "Data". The "Logic" tab is selected and highlighted with a red box labeled "1". Under the "Logic" tab, a tree view shows various actions. The "QRCodeScanner" folder is expanded, and the "ScanQRCode" action is highlighted with a red box. Other actions visible in the tree include "OfflineDataSync", "(System)", "OutSystemsCharts", "OutSystemsUI", "Server Actions", "BootstrapProducts", "Authentication", "OfflineDataSync", "(System)", "SampleData_CS", "Users", and "Integrations".

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**

The screenshot displays the SAP Fiori Builder interface. On the left, a process flow diagram shows the sequence: Start → ScanQRCode → QRcode → End. A red arrow labeled '1' points from the 'QRcode' variable in the flow to the 'QRcode' variable in the 'Assign' action configuration panel on the right. In the 'Assign' panel, the 'Label' field is empty. The 'Assignments' section shows 'QRcode' selected in the dropdown, with 'x.y = Value' entered in the text field. A red box highlights this text field, with a red arrow labeled '2' pointing to it. Below the 'Assignments' section, the 'Suggestions' list shows 'ScanQRCode.ScanResult' selected, highlighted with a red box and a red arrow labeled '3'. The 'Widget Tree' on the right shows the 'MainFlow' structure with 'Productcatalog' expanded, containing various filters and actions, with 'QRcode' highlighted in yellow.

Section 4 > B > 10. Refresh Screen Data

Refresh Products List

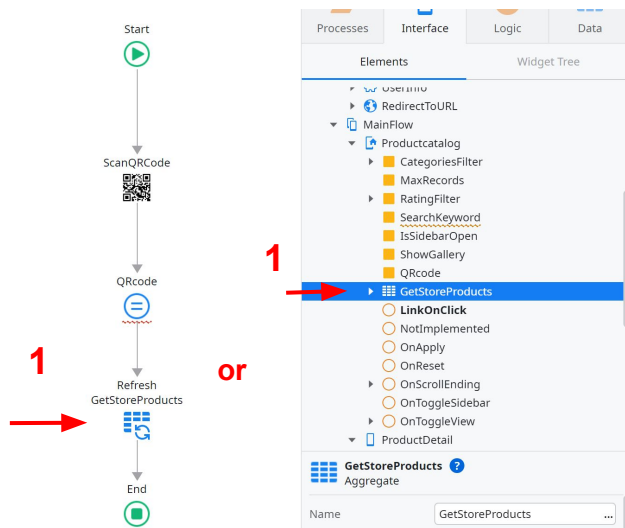
After scanning the QR Code, we need to refresh the Products list in the screen to reflect our filter (we will do the list filters later).

1. Use the **Refresh Data** widget.
2. When prompted for the data source to be refreshed, select **GetStoreProducts**.
3. Press **Select**

The image shows a workflow editor interface. On the left is a widget palette with a search bar and various actions. A red arrow labeled '1' points from the 'Refresh Data' widget in the palette to the 'Refresh GetStoreProducts' widget in the workflow. The workflow starts with 'Start', followed by 'ScanQRCode', 'QRcode', and 'Refresh GetStoreProducts', ending with 'End'. On the right, a 'Select Data Source' dialog box is open, showing a search bar and a list of data sources. A red box labeled '2' highlights 'GetStoreProducts' in the list. A red box labeled '3' highlights the 'Select' button at the bottom of the dialog.

Section 4 > B > 11. Reconfigure Data Query

Add Aggregate Filter




The screenshot shows the 'GetStoreProducts' aggregate editor. At the top, it displays '1 Source', 'No Filters', '1 Sorting', and 'No Test Values'. A red box highlights the 'No Filters' button, with a red '2' next to it. Below this, a red box highlights the 'Add filter' button, with a red '3' next to it. The main area shows a table with columns: 'Name', 'ProductCode', 'Picture', and 'Cat'. The table contains one row: 'Amazon Echo', 'CD628248719581', 'https://i.libb.co/ZgR2WLV/Amazon-Echo.png', and 'Voic'.

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

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

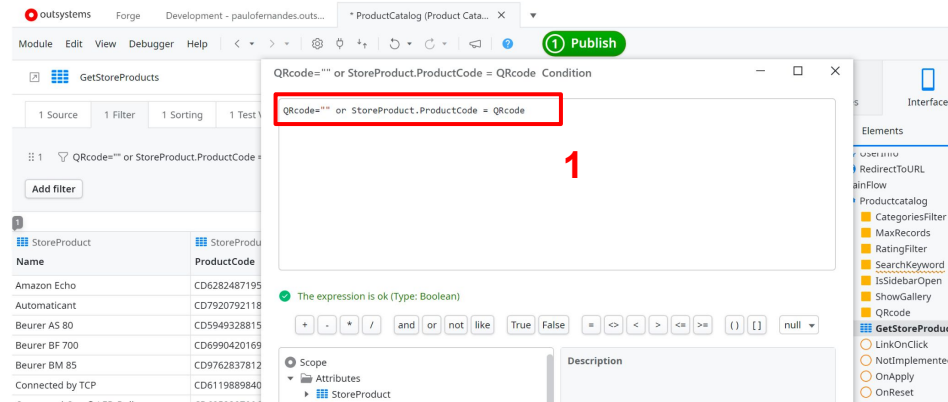
Section 4 > B > 11. Enter Query in Filter

Add Query

1. Enter the following query:
QRcode="" or StoreProduct.ProductCode = QRcode

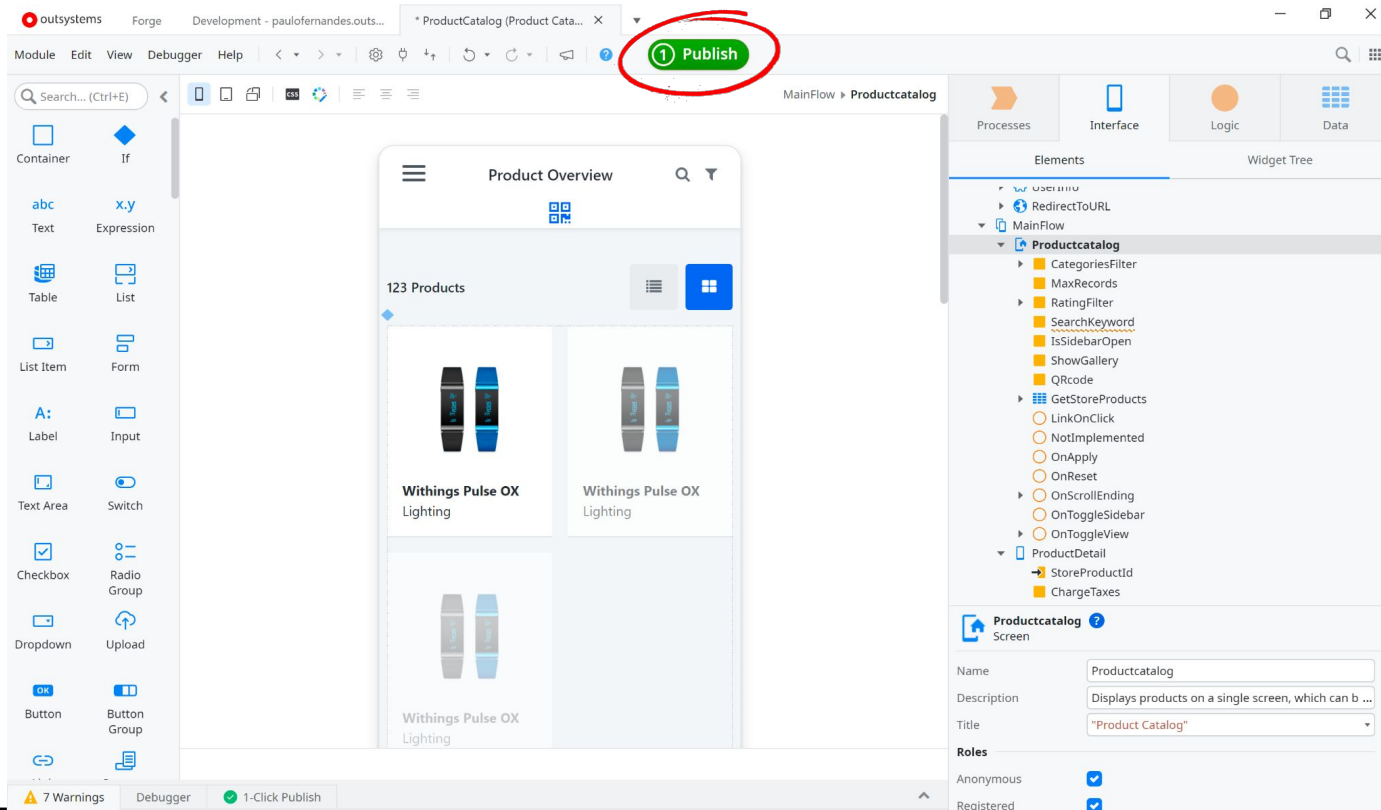
Explanation of the query:

- 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**.



Section 4 > B > 12. Deploy and Test it

Publish Application



Section 4 > B > 13. Deploy and Test it

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.

The screenshot shows the OutSystems IDE interface during the deployment of a mobile application. The main workspace displays a mobile application preview titled "Product Overview" showing a list of products, specifically "Withings Pulse OX Lighting". The "Deploy" button in the top right corner of the IDE is circled in red. Below the main workspace, the "Debugger" panel shows the deployment progress with four steps: "Uploading", "Compiling", "Deploying", and "Done". The "Done" step is also circled in red. The "Productcatalog" screen configuration is visible on the right side of the IDE, showing the name "Productcatalog" and a description "Displays products on a single screen, which can b ...".



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

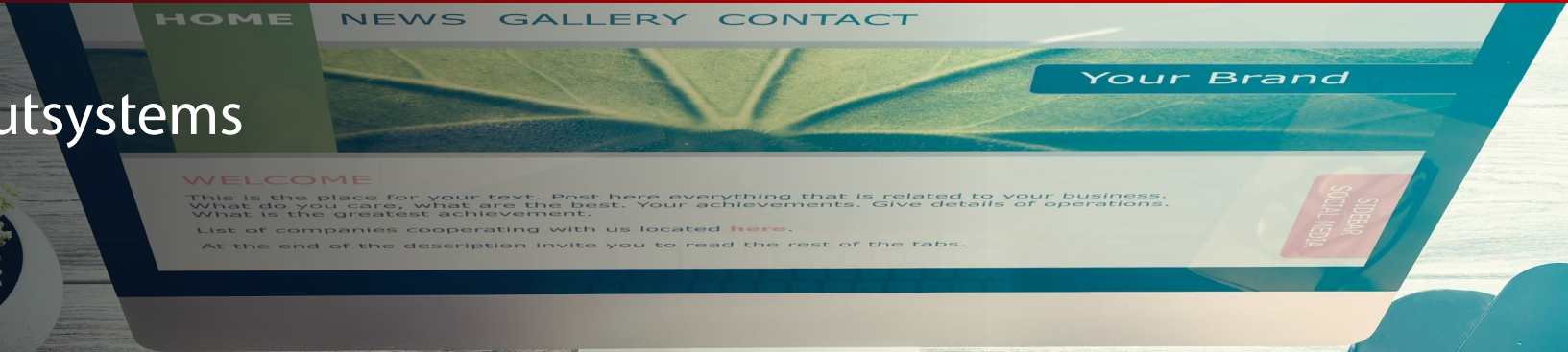


FitBit Blaze



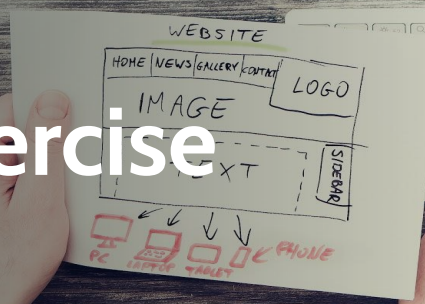
iHealth Blood Pressure Monitor





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, let's 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.



Congratulations

You have successfully built a mobile app!



Jumpstart training

This material is owned by OutSystems and may only be used in the ways described in this Copyright Notice:

You may take temporary copies necessary to read this document

You may print a single copy of this material for personal use

You must not change any of this material or remove any part of any copyright notice

You must not distribute this material in any shape or form