Develop a Smart SAP Catalog

1. Introduction

This tutorial shows you how to develop a Mendix app which uses, as its source, data which is held on an SAP back-end system.

You will create an app to maintain a catalog of products which integrates seamlessly with the SAP S/4 system. The app will support the admin and user roles.

You will then add the ability to search the catalog using SAP Leonardo Machine Learning image classification.

Specifically, you will learn how to:

- Create a Mendix app based on a template
- Generate the app's domain model based on an OData service coming from an SAP S/4 system
- Set up Mendix so that you can consume an SAP OData service using pre-built components from the Mendix App Store
- Adding more functionality to your app by modeling in Mendix
- Make your app 'smart' by leveraging the SAP Leonardo Machine Learning Foundation Connector
- Set security in your app
- Deploy your app to SAP Cloud Platform

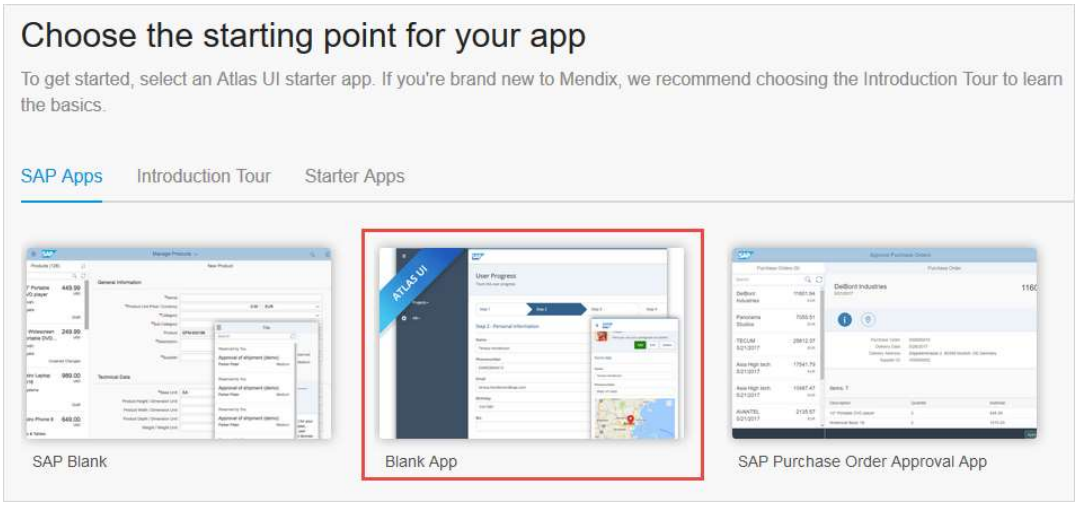
2. Prerequisites

You are provided with all the information and software which you need to perform this tutorial.

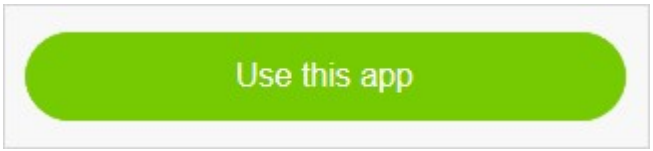
3. Starting a New SAP App

The first step is to create a Mendix app. You need to link this to SAP Cloud Platform so that you can run the app there.

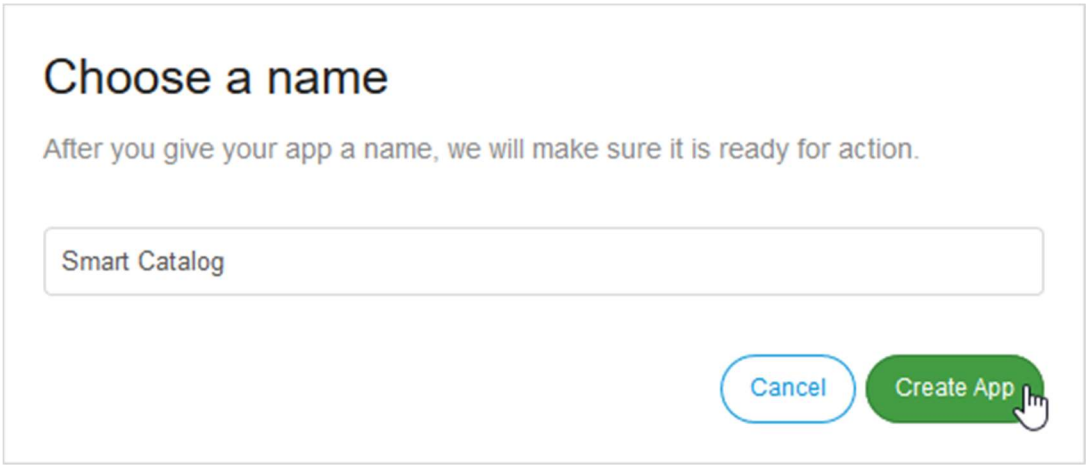
1. Choose the **Blank App** with Atlas UI styling.



2. Click **Use this app** to confirm that this is the one you want.



3. Enter *Smart Catalog*, as the name and click **Create App**.

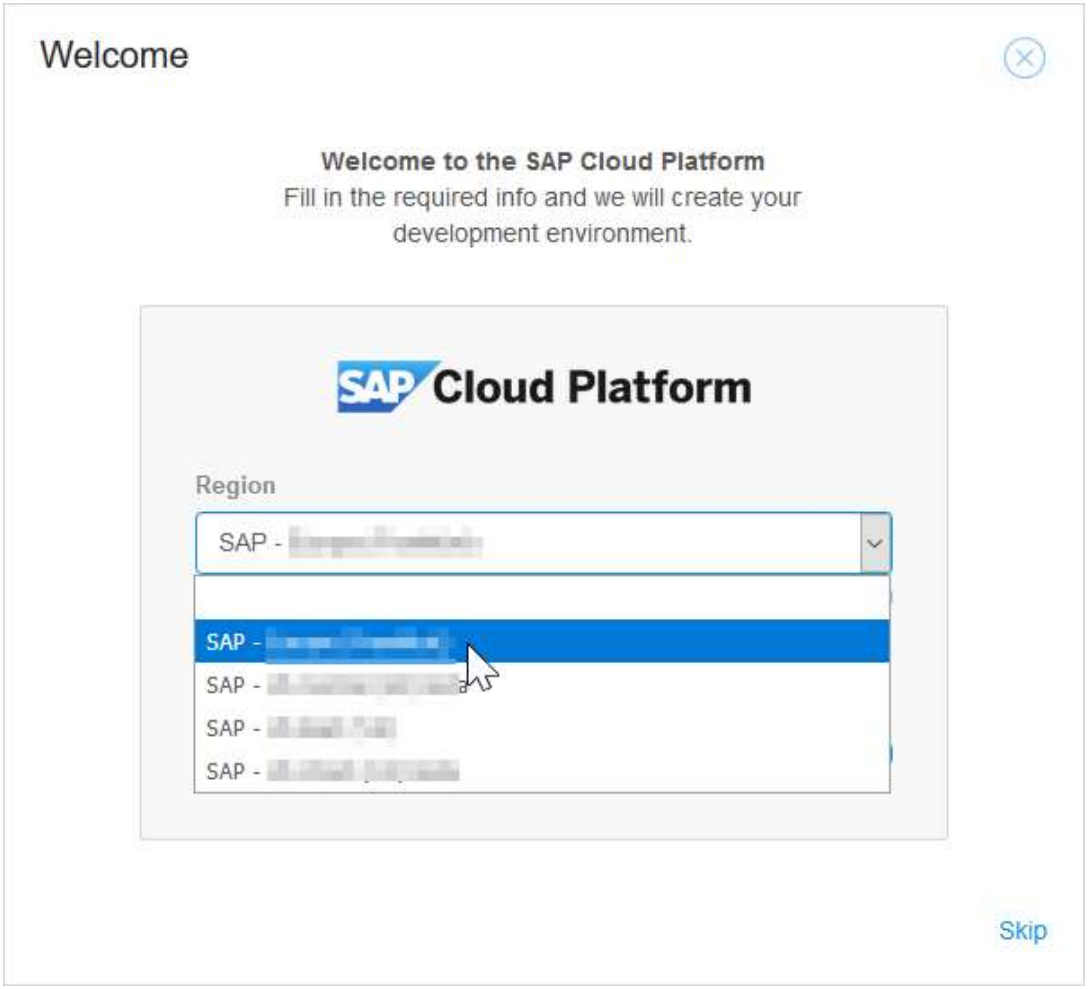


4. Creating an SAP Cloud Platform Environment

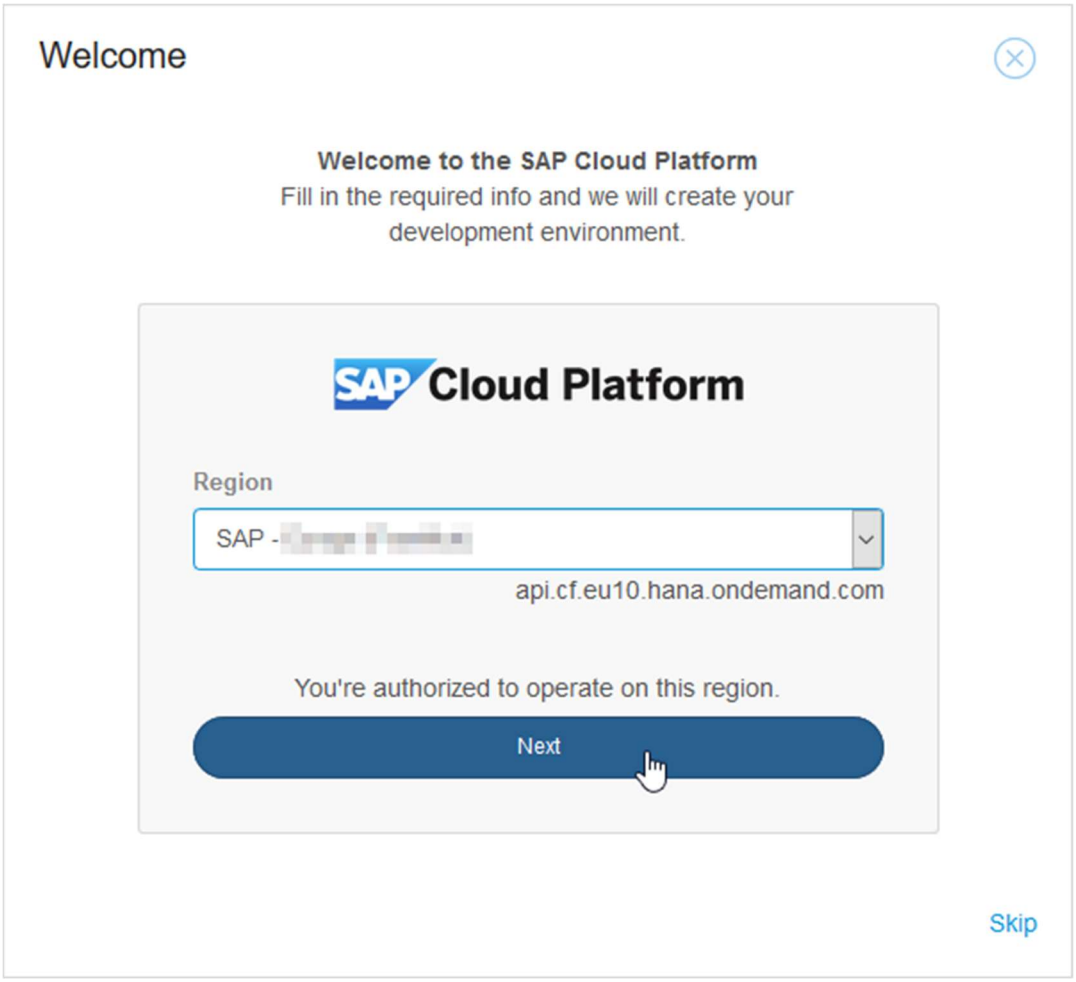
SAP needs to know exactly which region, account, subaccount, and space you will be using to deploy this app. Mendix will take you through the process step-by-step.

Depending on whether you have used SAP from Mendix before, and whether you are currently signed in to SAP you may have to log in to SAP and confirm that SAP and Mendix can share information about the app.

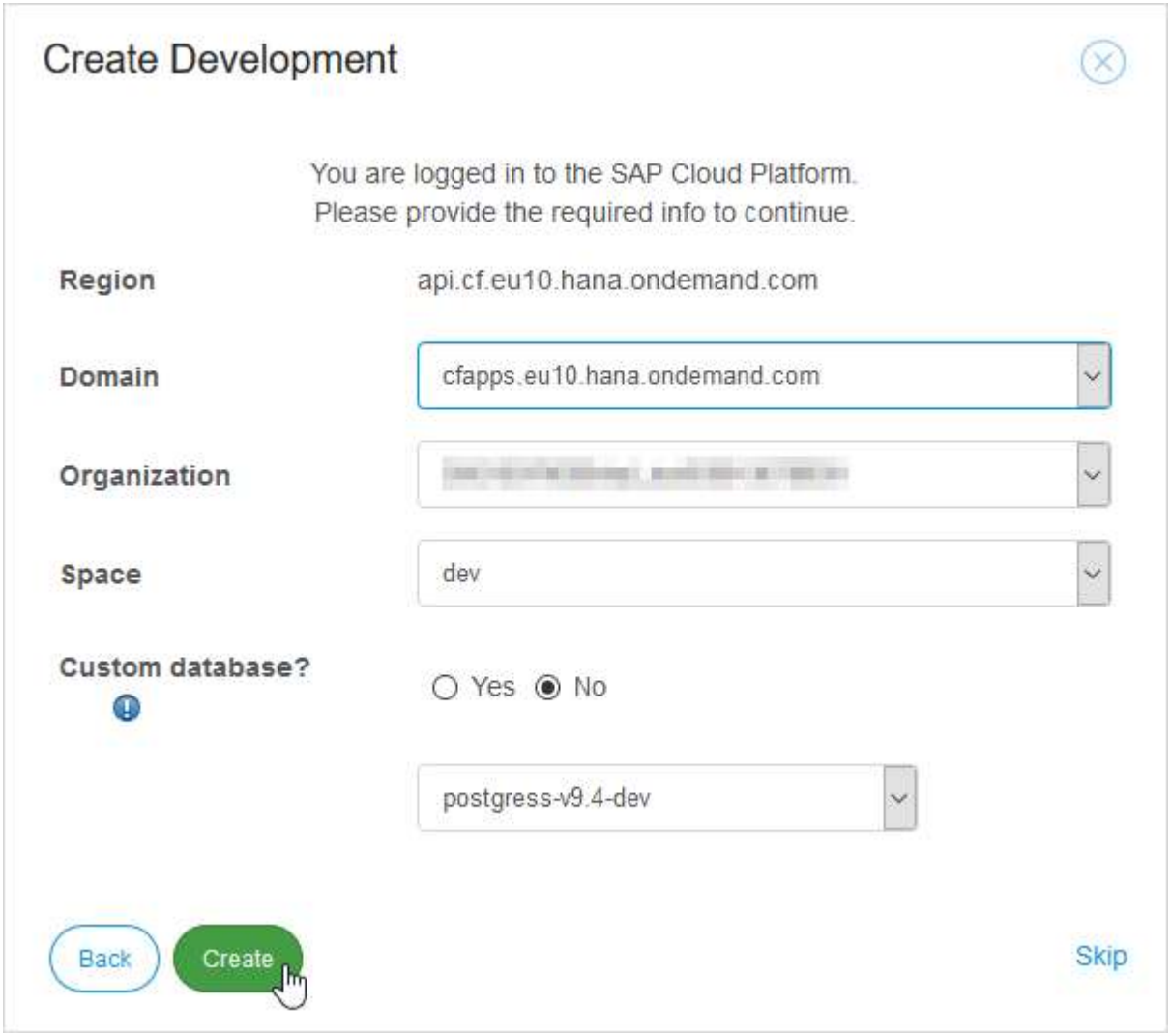
1. Select the **Region** of your SAP account.



2. Click **Next**.

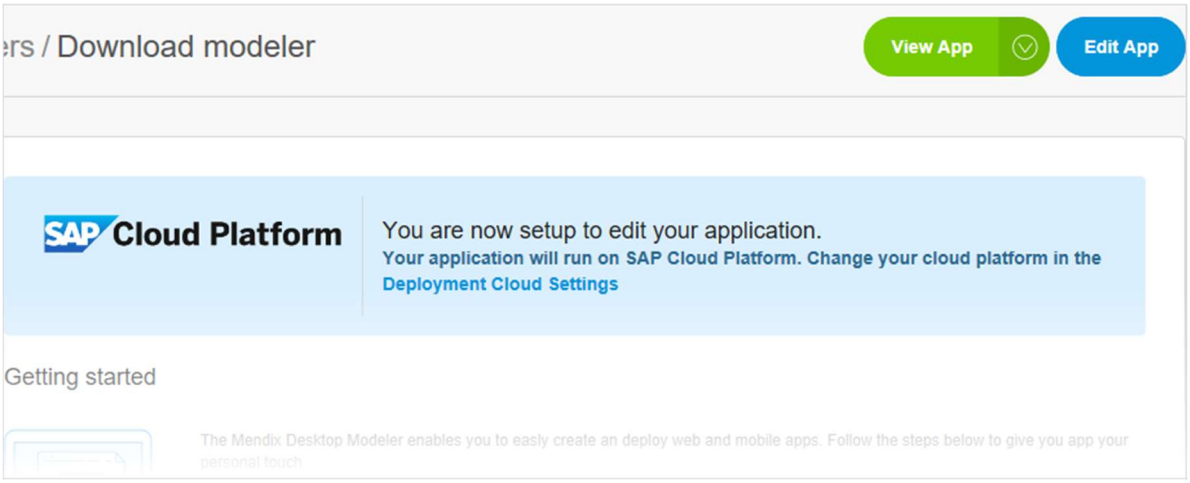

Welcome
Welcome to the SAP Cloud Platform
Fill in the required info and we will create your development environment.
SAP Cloud Platform
Region
SAP -
api.cf.eu10.hana.ondemand.com
You're authorized to operate on this region.
Next
Skip

3. Select your **Domain**, **Organization** (subaccount), and **Space**. You will only be able to select options which are accessible to your account.
4. Select No for Custom Database?.
5. Leave the **Postgress-v9.4-dev** [sic] database selected.
6. Click **Create**.

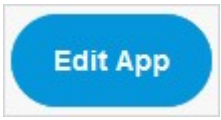


SAP now creates the environment for you, with all the runtime resources which are needed by your app.

You are shown a confirmation page and can now edit your app.

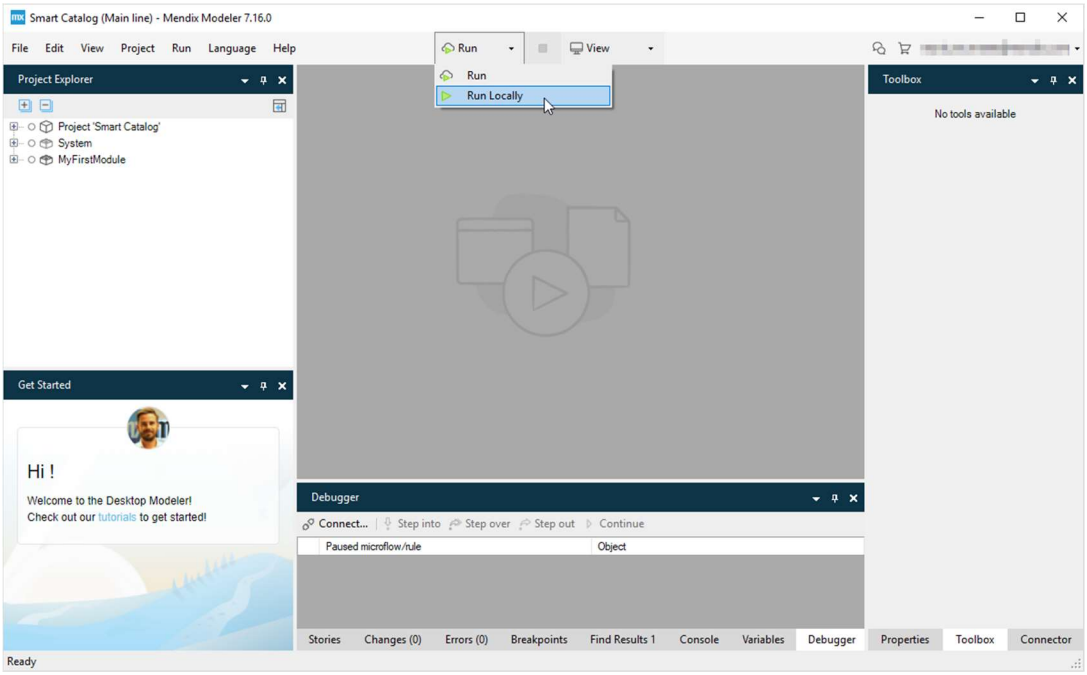


7. Click **Edit App**.

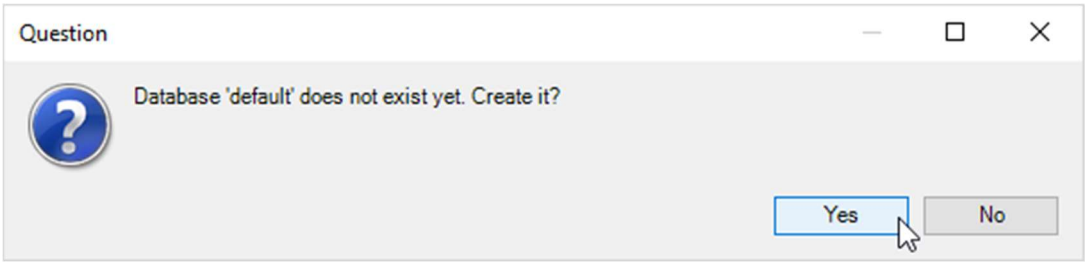


The correct version of the Mendix Desktop Modeler will open, automatically download your app to your computer, and open it.

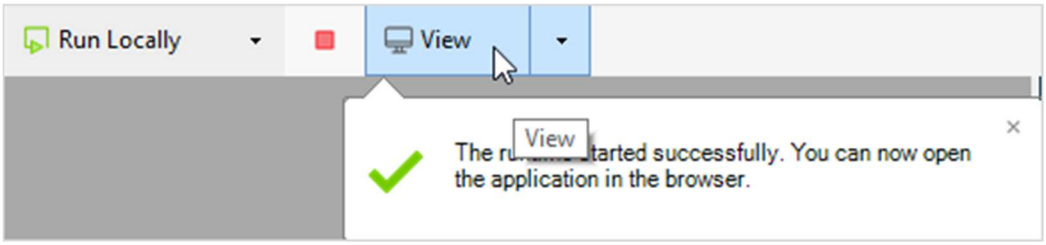
8. Click the *arrow* next to the **Run** button in the Desktop Modeler.
9. Click **Run Locally**.



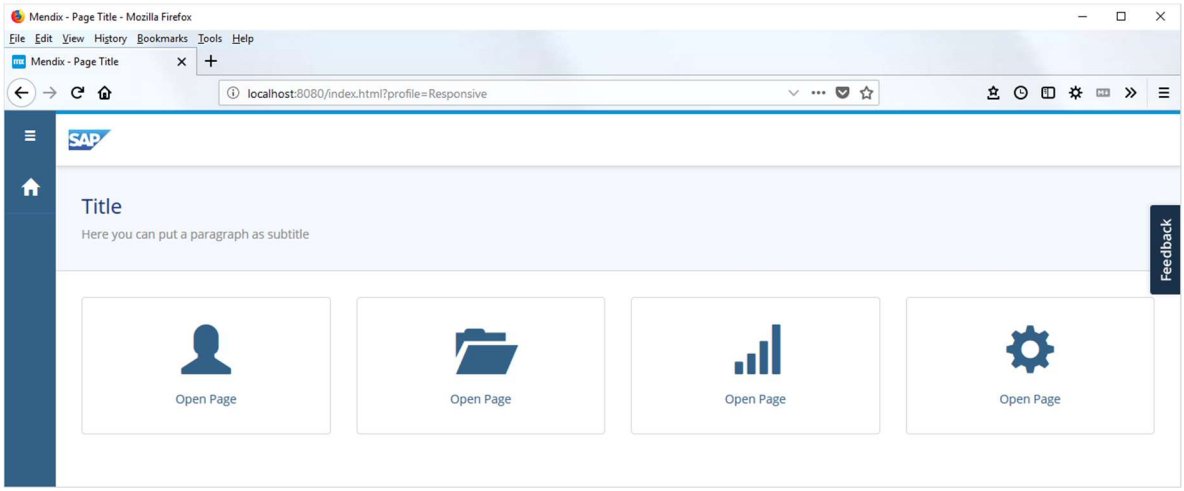
10. Click **Yes** to confirm that you want to create a database for the app.



11. Wait for the app to be started.
12. Click **View** to view the app in a browser.



The App looks like this:

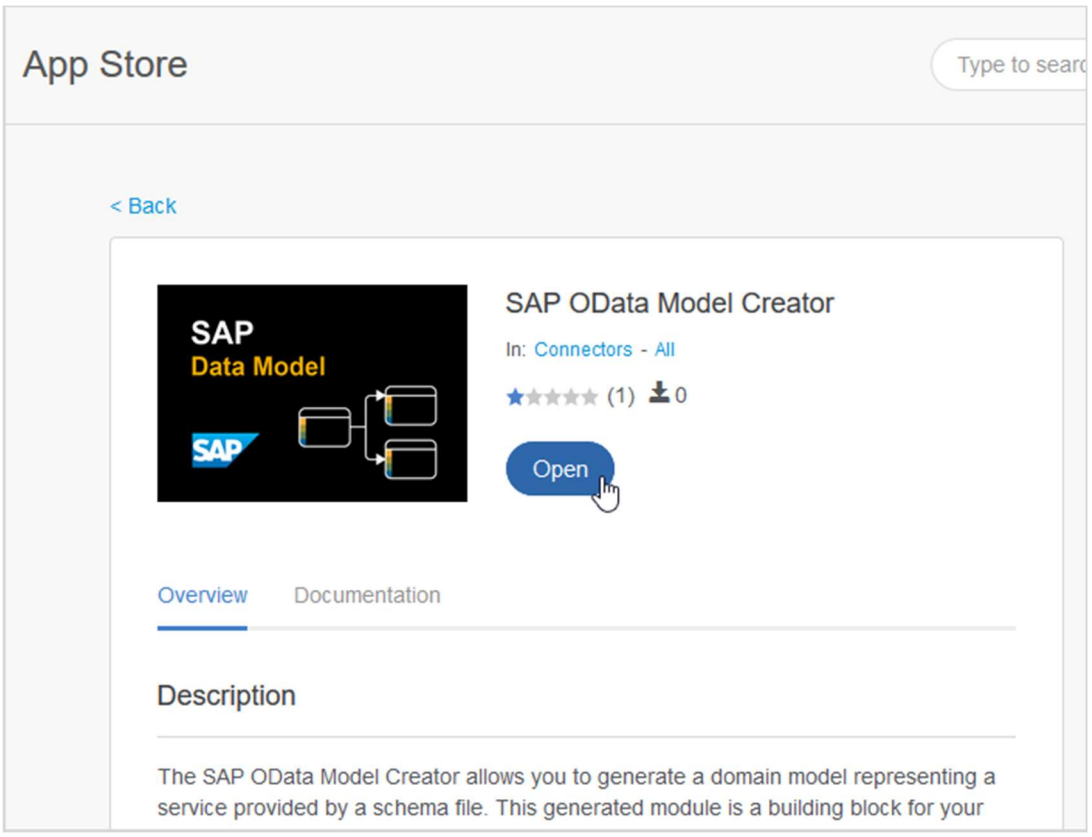


5. Creating an OData Data Model for the Product Management OData service

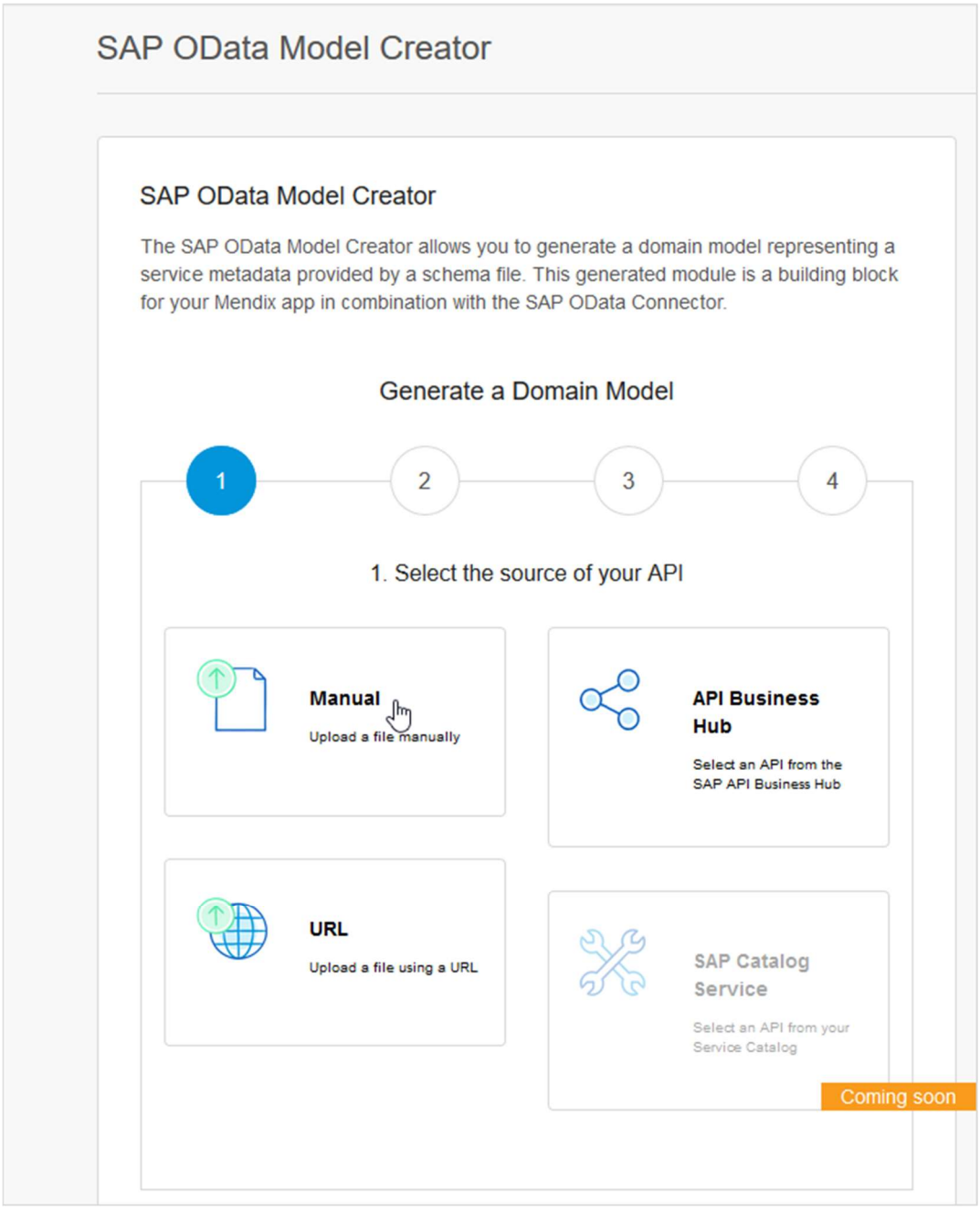
You are going to use a product catalog which is held in the SAP back-end system. These have been exposed via an OData service called **EPM_REF_APPS_PROD_MAN_SRV** is provided in the *student share* folder.

Your Mendix app needs to know the details of the EPM_REF_APPS_PROD_MAN_SRV OData service before you can get data from it. You do this by creating a data model using the SAP OData Model Creator in the Mendix App Store.

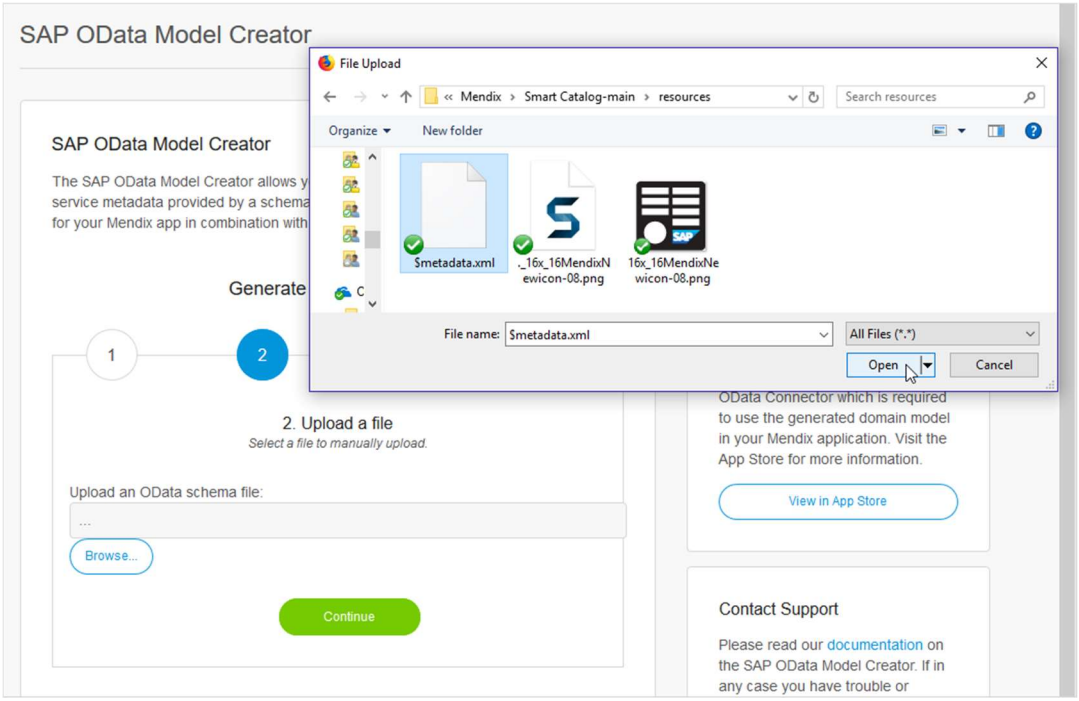
1. Open the SAP OData Model Creator App Store page (https://appstore.home.mendix.com/link/app/105622/) in your browser.
2. Click **Open** to start the SAP OData Model Creator.



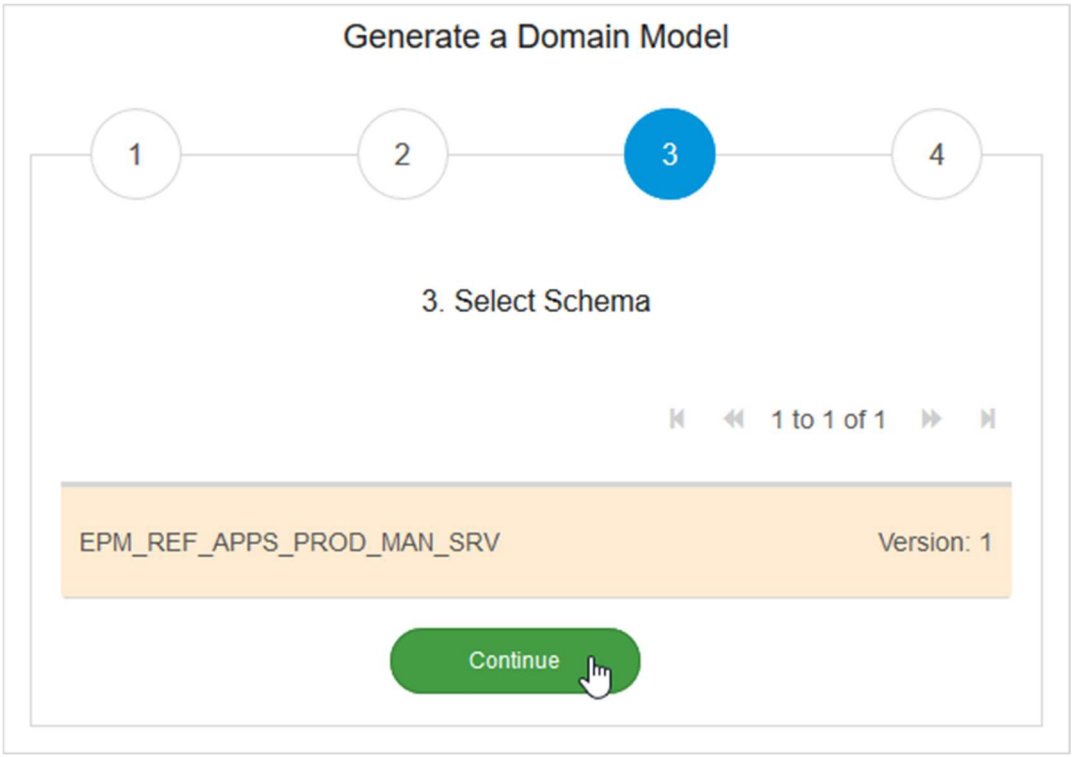
3. Click **Manual** as the source for the API.



4. Select the $metadata.xml file in the session folder in the *student share* and click **OK**.

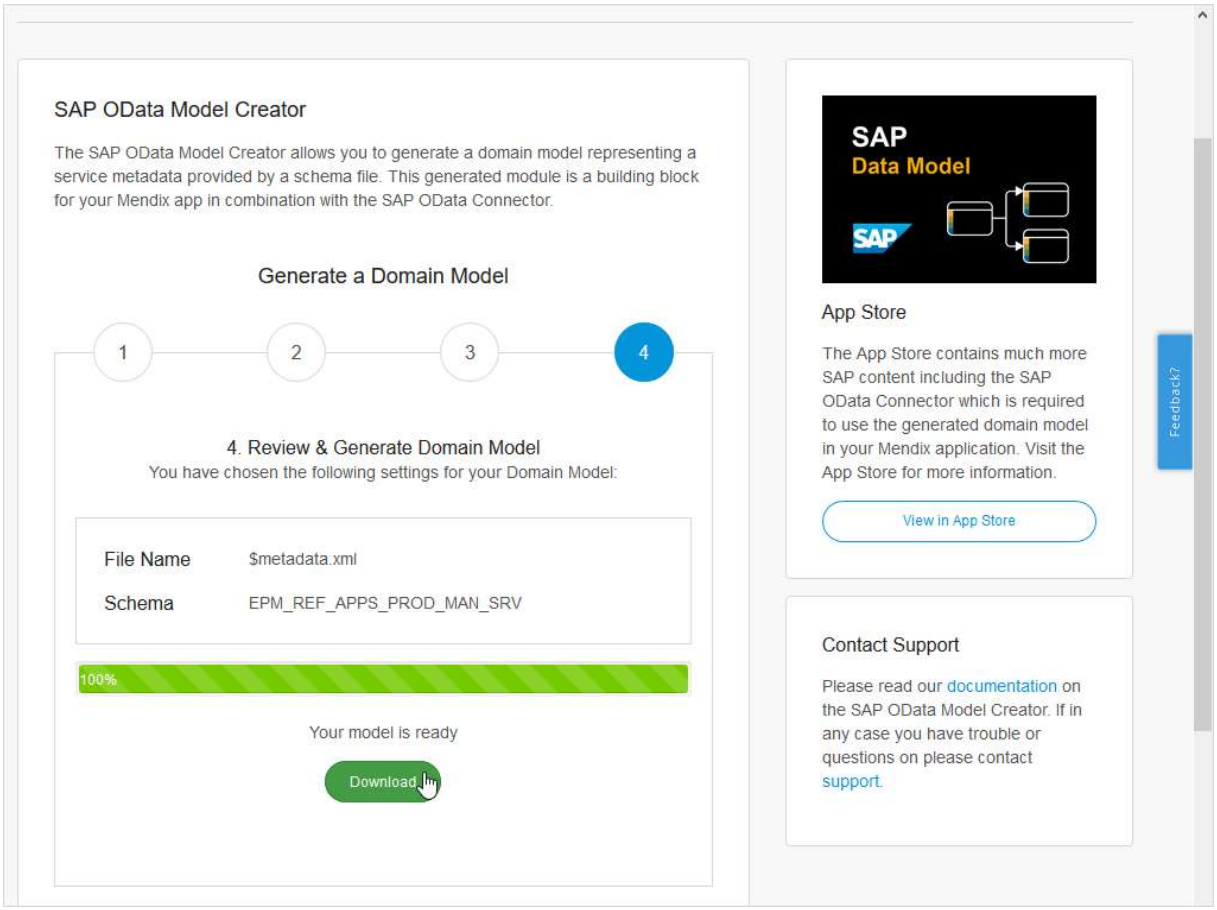


5. Click Continue.
6. Select the EPM_REF_APPS_PROD_MAN_SRV schema and click **Continue**.



7. Click **Generate .mpk**.

8. Once the generation is complete, the **Download** button appears.

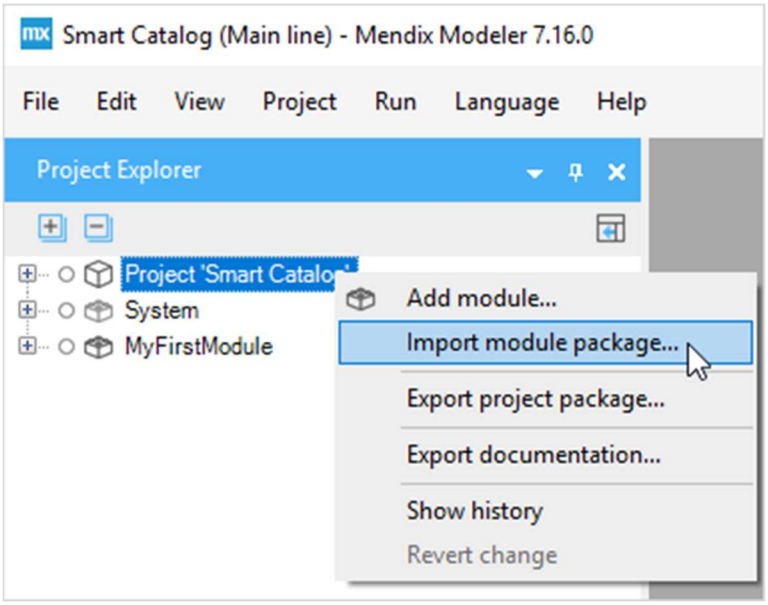


9. Click **Download** to save the file on your local drive.

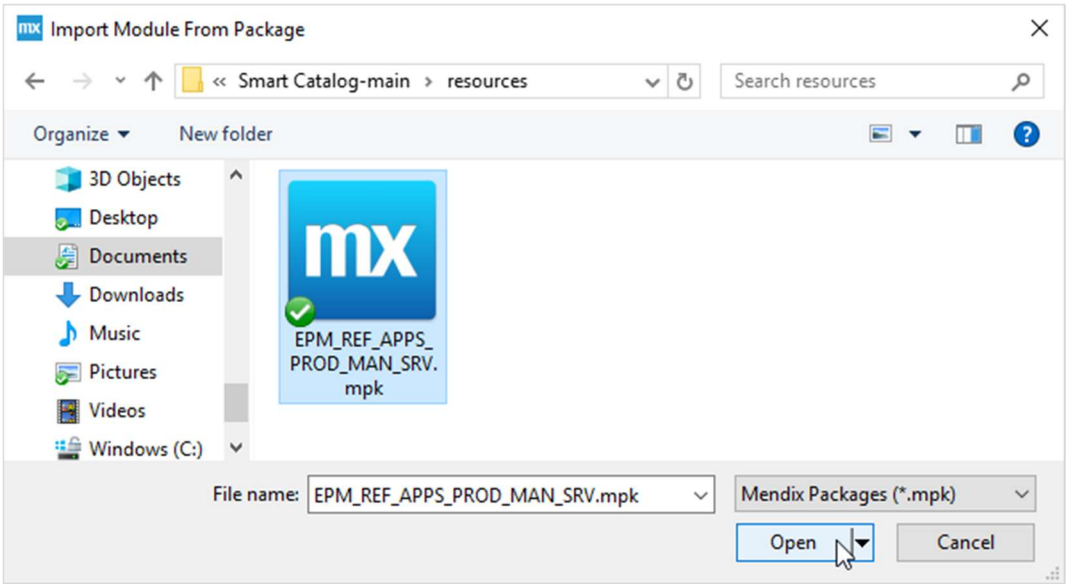
6. Importing the Data Model into your App

You now have all the information which Mendix needs to get your data out of the SAP back-end using the EPM_REF_APPS_PROD_MAN_SRV OData service. Now all you have to do is to import it into your Smart Catalog app, so that you can use it there.

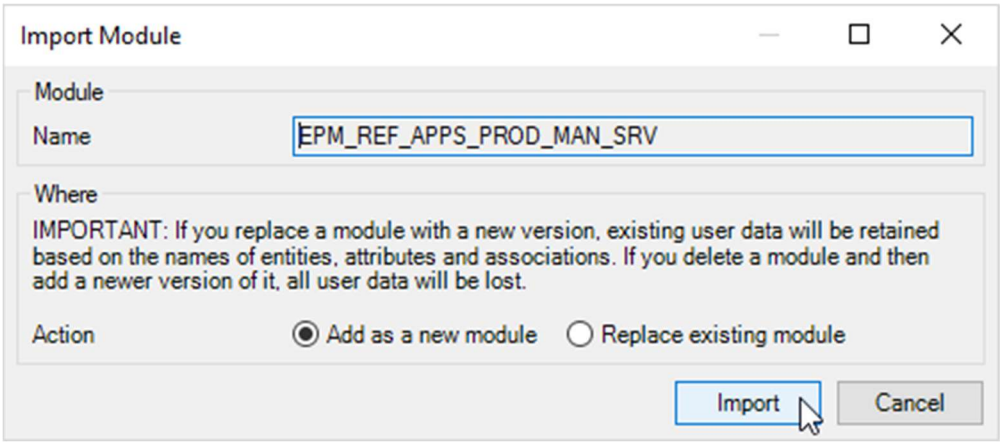
1. Return to your *Smart Catalog* app in the Desktop Modeler.
2. Right-click the project name and choose **Import module package...**.



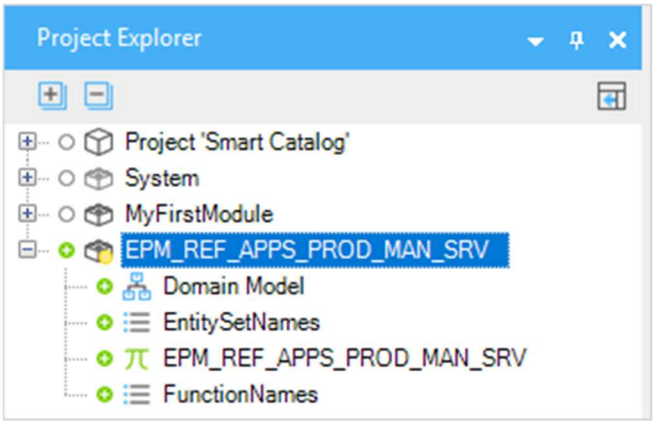
3. Find the module file you have just generated with the SAP OData Model Creator.
4. Click **Open**.



5. Select **Add as a new module** and click **Import**.



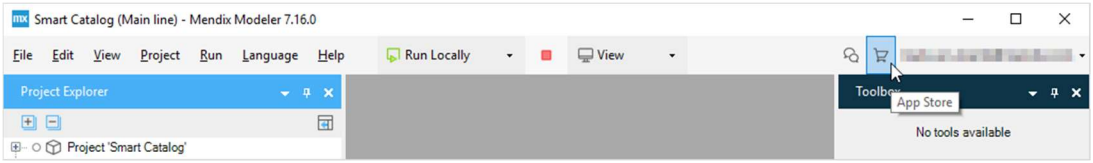
Your module will now be part of your Project:



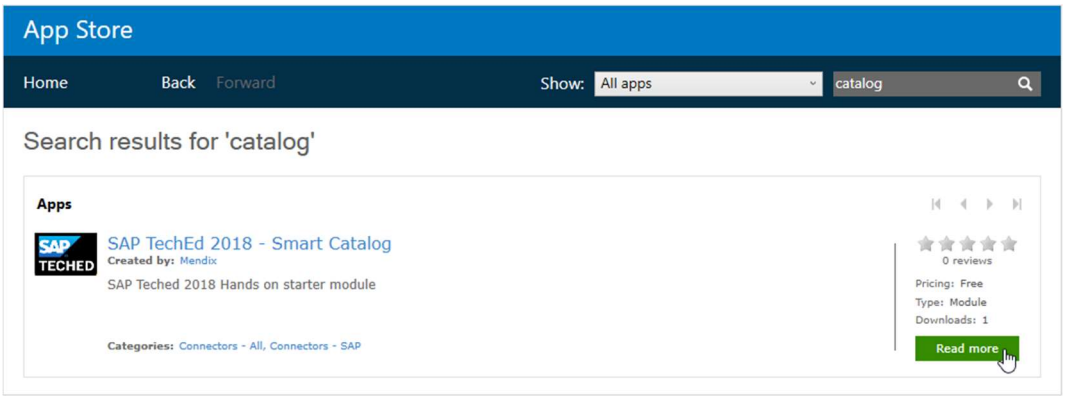
7 Importing Required Modules into the Mendix App

Some of the app has been written already, so you need to import those pieces into your app.

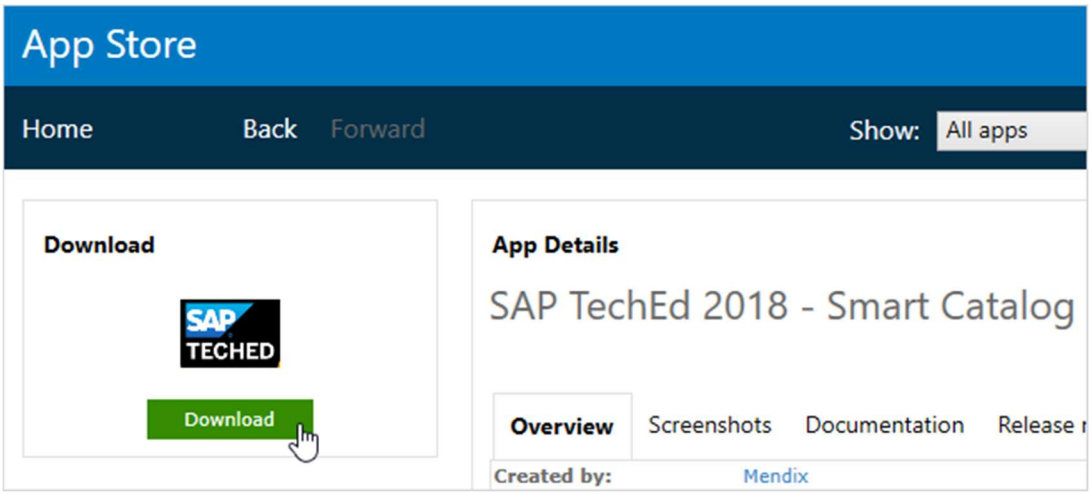
1. Click the App Store icon (the shopping basket) in the Desktop Modeler.



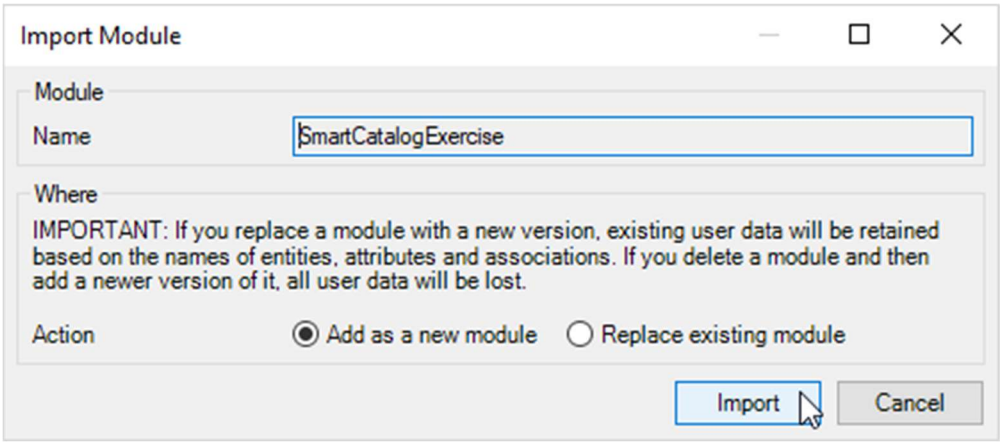
2. Enter *Catalog* in the search box and click the magnifying glass.
3. Click **Read more** next to *SAP TechEd 2018 - Smart Catalog*.



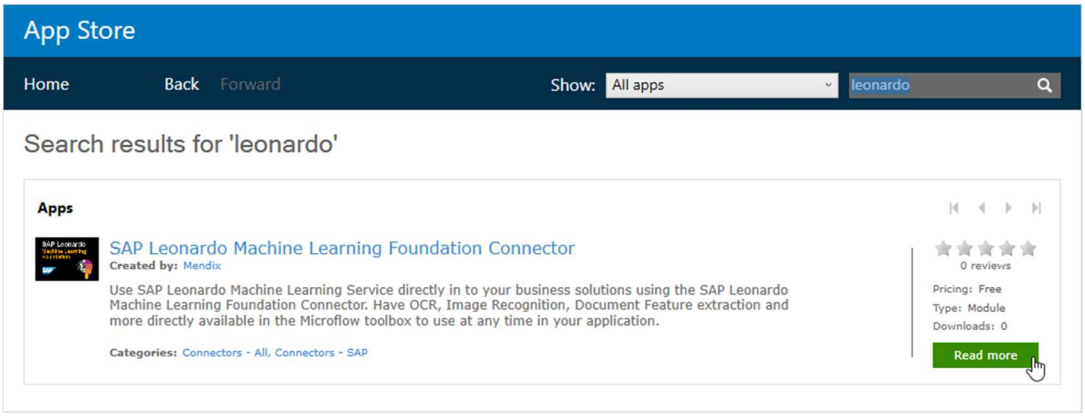
4. Click **Download** to add the module to your project.



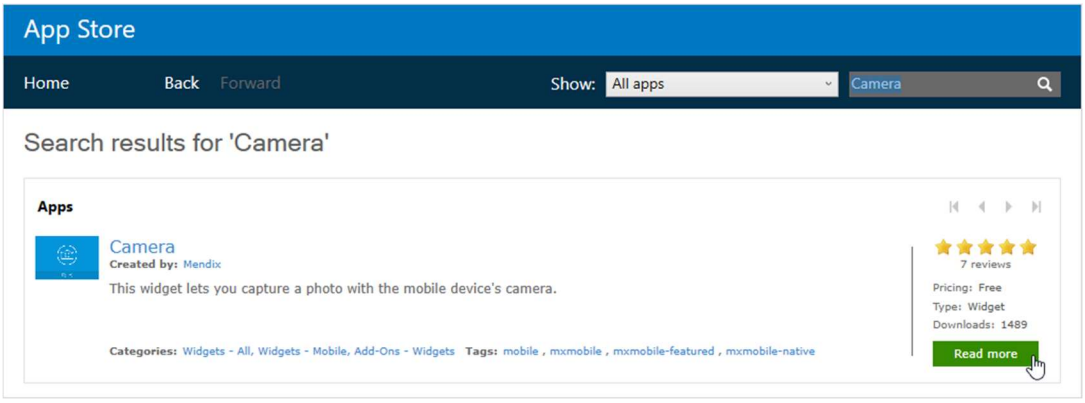
5. Click **Import** to confirm that you want to import the module.



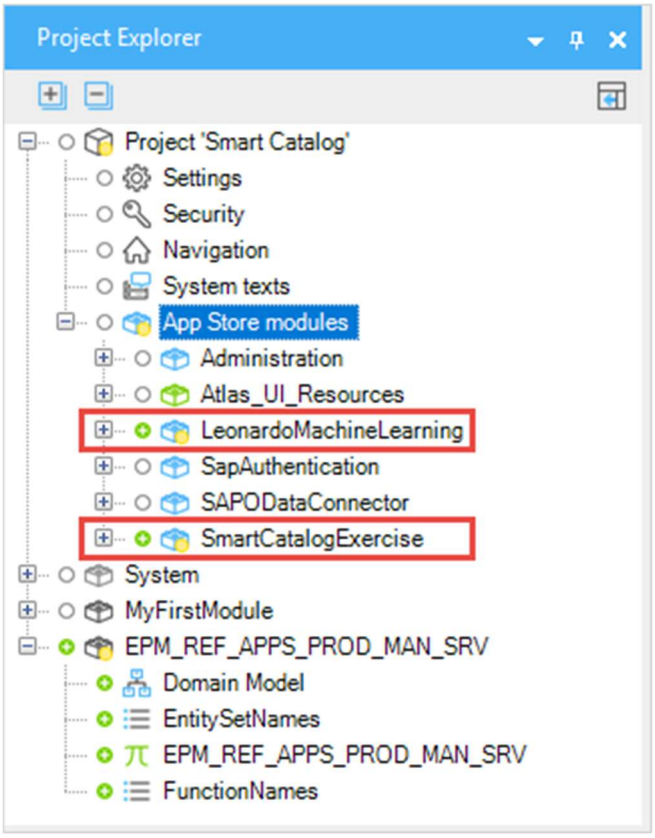
6. Repeat steps 3 through 6 to find and download the **SAP Leonardo Machine Learning Foundation Connector** module.



7. Repeat steps 3 through 6 to find and download the **Camera** module.



8. You can see the *LeonardoMachineLearning* and *SmartCatalogExercise* modules you have imported, along with other modules, by expanding the tree structure in the **Project Explorer**. The *Camera* module has created a new camera widget which you will use later in the exercise.



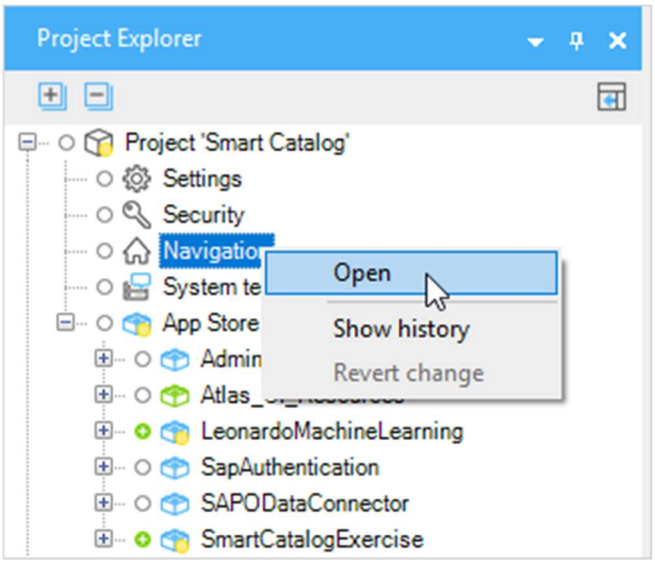
- The **SAP TechEd 2018 - Smart Catalog** module contains the initial modeling for your app
- The **SAP Leonardo Machine Learning Foundation Connector** allows your app to connect to SAP Leonardo Machine Learning Foundation services
- The **Camera** module allows your app to access the camera on your device

8. Setting up Application Navigation

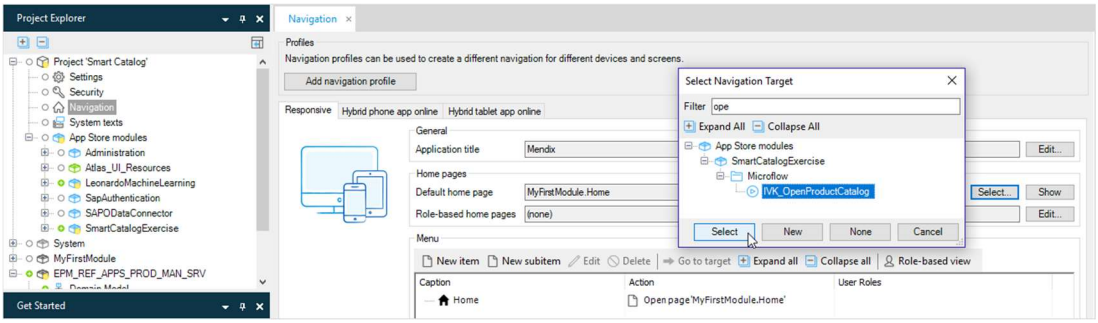
To ensure the application is using the imported module, the navigation of the application needs to be adjusted.

Mendix apps work by showing pages to the user. You can define which page should be the Home page: the first page the user sees. Each page in your Mendix app can also have a menu bar. You can define which pages appear in this.

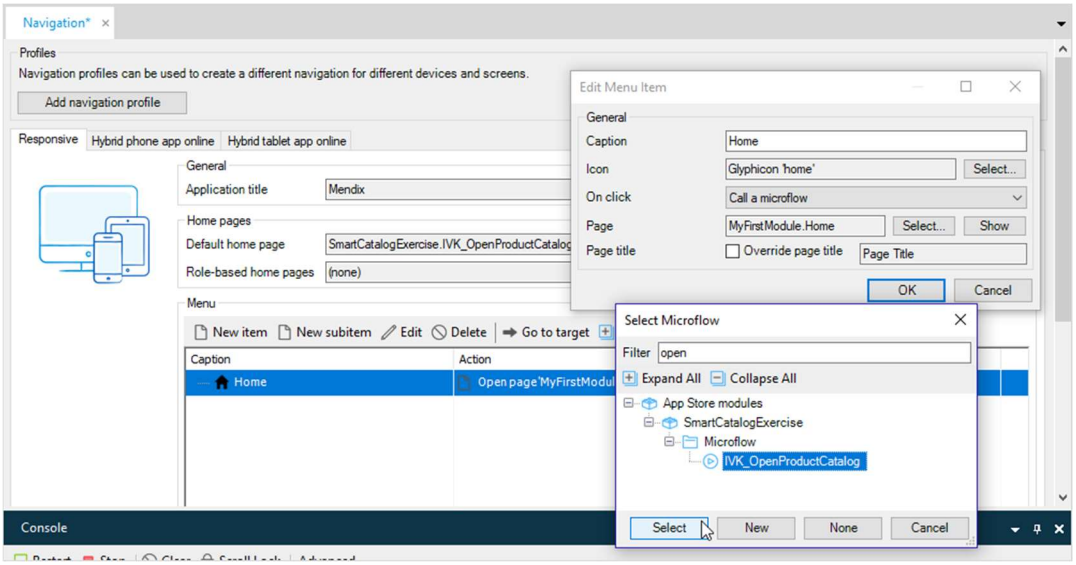
1. Right-click **Project 'Smart Catalog' > Navigation** and click **Open**.



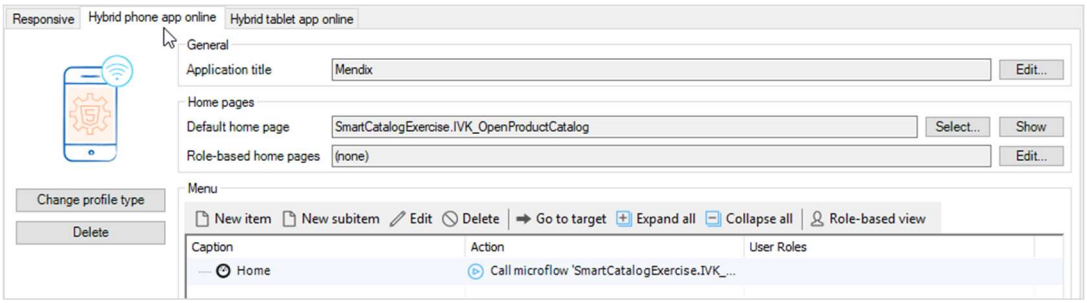
2. Click **Select...** next to **Default home page**.
3. Enter *open* in the **Filter**.
4. Select **App Store modules > SmartCatalogExercise > Microflow > IVK_OpenProductCatalog** as the new home page.
5. Click **Select**.



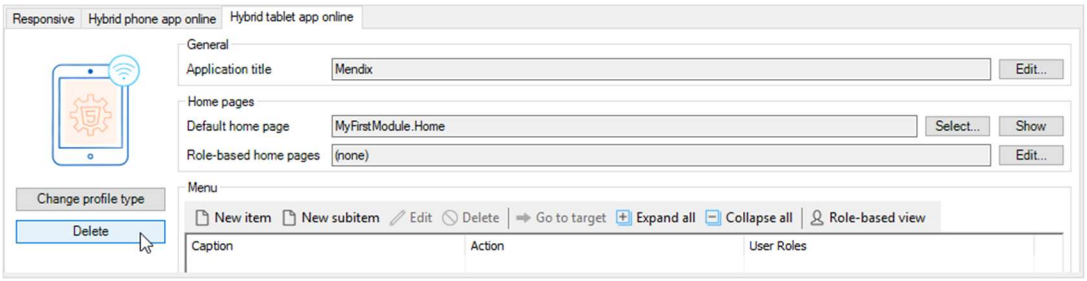
6. Click the **Home** page in the **Menu** section of the **Navigation**.
7. Click **Edit**.
8. Select *Call a microflow* for **On click**.
9. Select the **IVK_OpenProductCatalog** microflow using the filter.
10. Click **Select**.



11. Click **OK** to confirm the change.
12. Click the **Hybrid phone app online** tab on the Navigation dialog.
13. Repeat steps 2 through 11 for this tab.



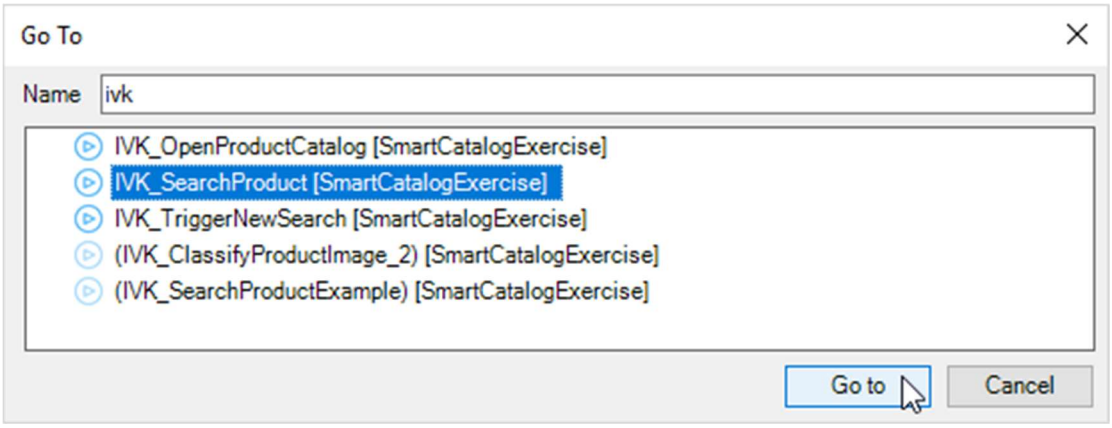
14. Click the **Hybrid tablet app online** tab on the Navigation dialog.
15. Click **Delete** to delete this navigation profile.



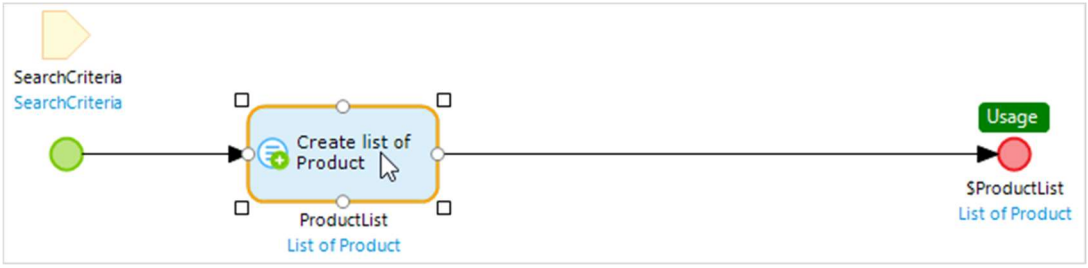
9 Retrieving Data from S/4HANA

The **SmartCatalogExercise** module comes with no products. You need to replace a retrieve from the local database with an OData **Get** action against the **EPM_REF_APPS_PROD_MAN_SRV** service.

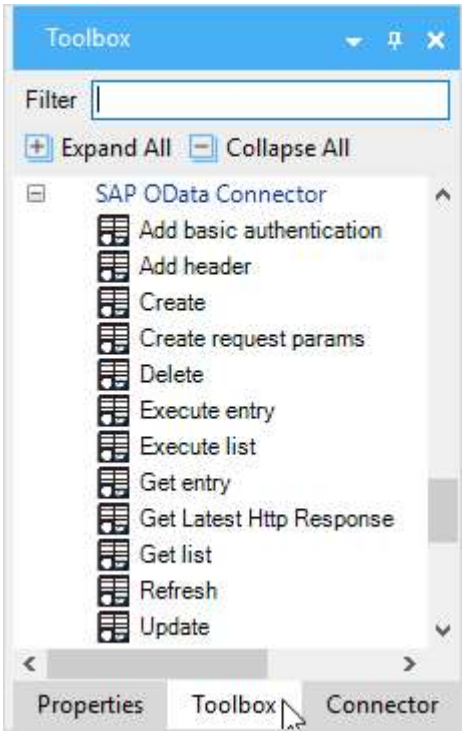
1. Press `Ctrl` + `G` to open the **Go To** dialog.
2. Search for the microflow **IVK_SearchProduct**.
3. Click **Go to**.



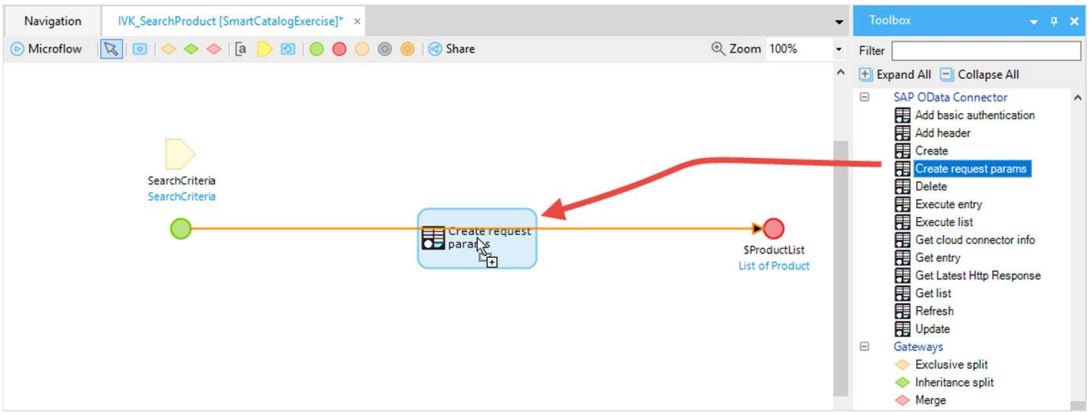
4. Click the action Create list of Product.
5. Press `Delete` to delete this action.



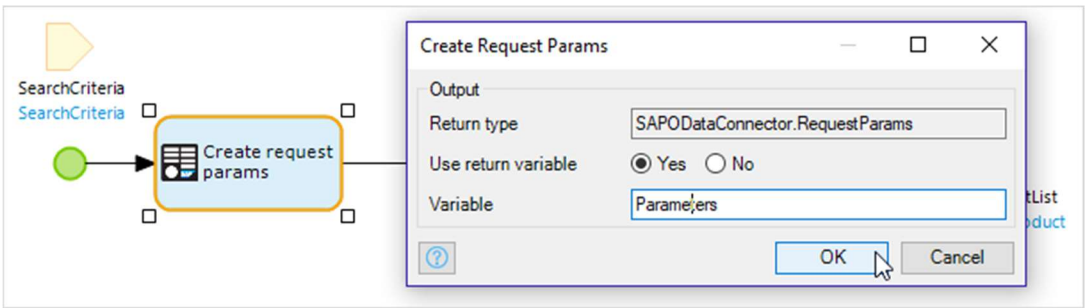
6. Click the **Toolbox** tab in the right-hand pane to switch to the microflow toolbox.



7. Drag an SAP OData Connector > Create request params into the microflow.



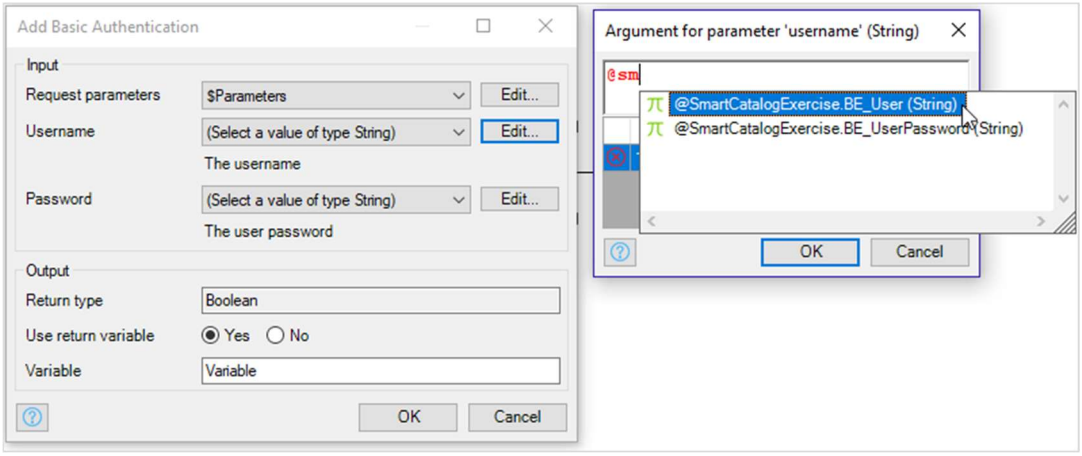
8. Double-click the **Create request params** action to open the properties.
9. Enter *Parameters* as **Variable** name.
10. Click **OK**.



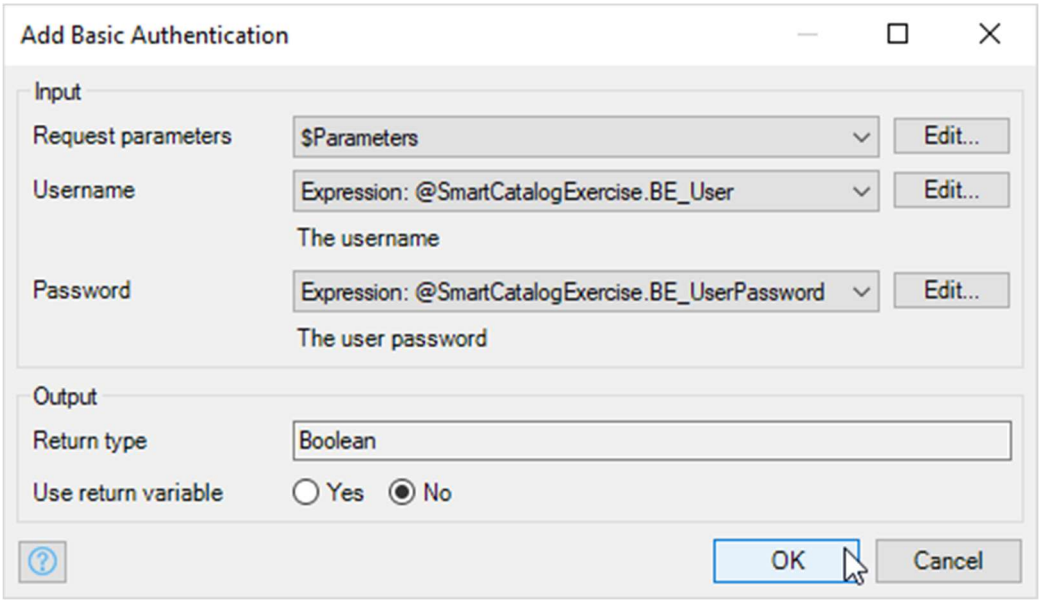
11. Drag an **SAP OData Connector > Add basic authentication** action after the **Create request params** action in the microflow.
12. Double-click this new action.
13. Select **$Parameters** as the **Request parameters**.
14. Click **Edit...** next to **Username**.
15. Enter **@sm**.
16. Click **@SmartCatalogExercise.BE_User (String)** to select it.

(This is a constant containing a user for the S/4HANA system which has been set up in advance).

17. Press `Enter` to confirm your selection. (Check that the full name of the parameter appears in the dialog).



18. Click **OK**.
19. Repeat steps 14 through 17 to set **Password** to *@SmartCatalogExercise.BE_UserPassword (String)*. This is the password for the user which was set up on the S/4HANA system.
20. Select *No* for **User return variable**.
21. Click **OK** to confirm the basic authentication action.

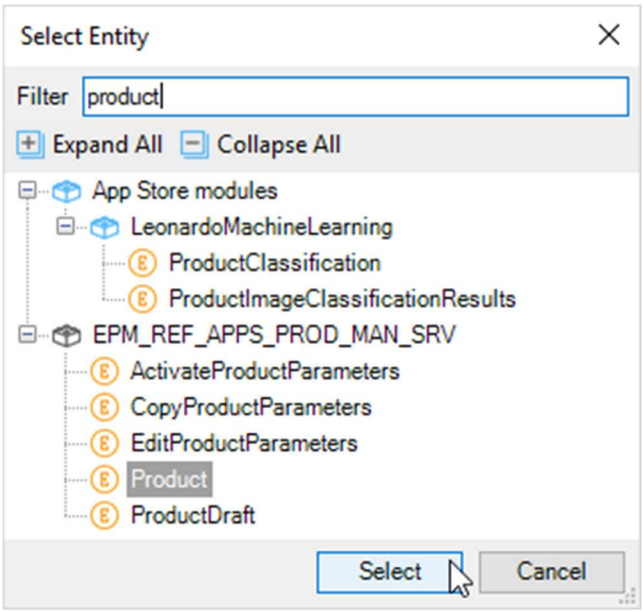


22. Drag **SAP OData Connector > Get list** action after the **Add basic authentication** action in the microflow.
23. Double-click on the **Get list** action.
24. Click **Edit...** next to the **Query**.
25. Enter (or copy and paste) the following query in the Edit box.

```
@EPM_REF_APPS_PROD_MAN_SRV.EPM_REF_APPS_PROD_MAN_SRV + '/' + toString
(EPM_REF_APPS_PROD_MAN_SRV.EntitySetNames.Products) +
'/?$filter=substringof(SubCategoryName,''' +
urlEncode($SearchCriteria/Term) + ''')'
```

This is the OData query which retrieves a list of products which contain the search term which is entered on the product list page. The first part of the query identifies the S/4HANA service list of products. This is then filtered to only return products where the search term appears in the product's SubCategoryName.

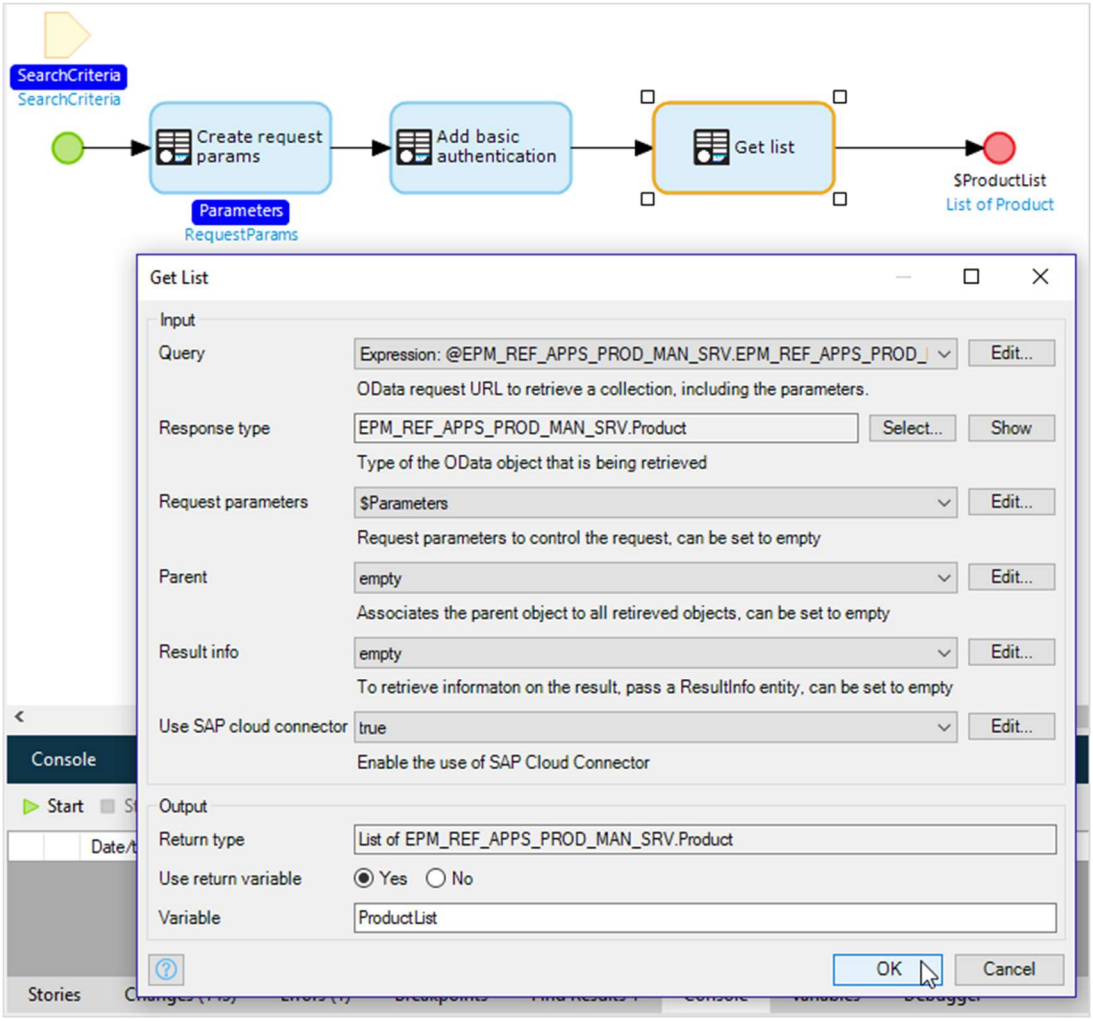
26. Click **OK**.
27. Click **Select...** next to the **Response type**.
28. Search for *Product*.
29. Select **EPM_REF_APPS_PROD_MAN_SRV > Product**.
30. Click **Select**.



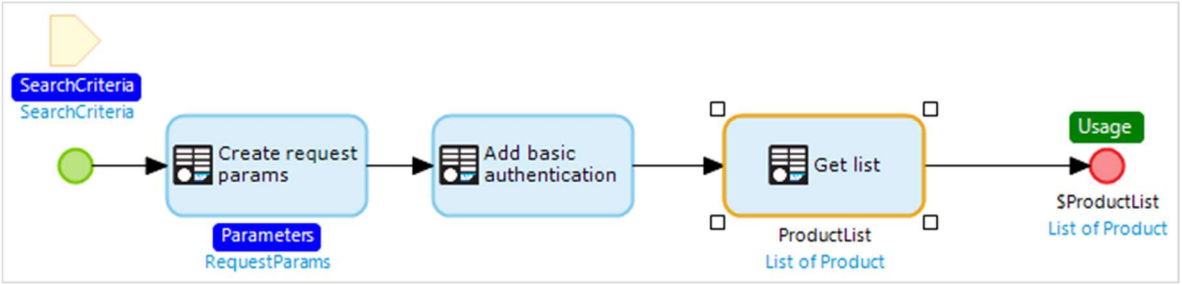
31. Select *$Parameters* from the dropdown for **Request Parameters**.
32. Select *empty* from the dropdown for **Parent**.
33. Select *empty* from the dropdown for **Result info**.
34. Select *true* from the dropdown for **Use SAP cloud connector**.

The SAP Cloud Connector has been configured in advance. It means that you can connect to the S/4HANA system (which is running on-premises and is not publicly available) just by setting this value to *true*. Mendix then makes the connection automatically for you, behind the scenes.

35. Enter *ProductList* for the **Variable** name.
36. Click **OK**.

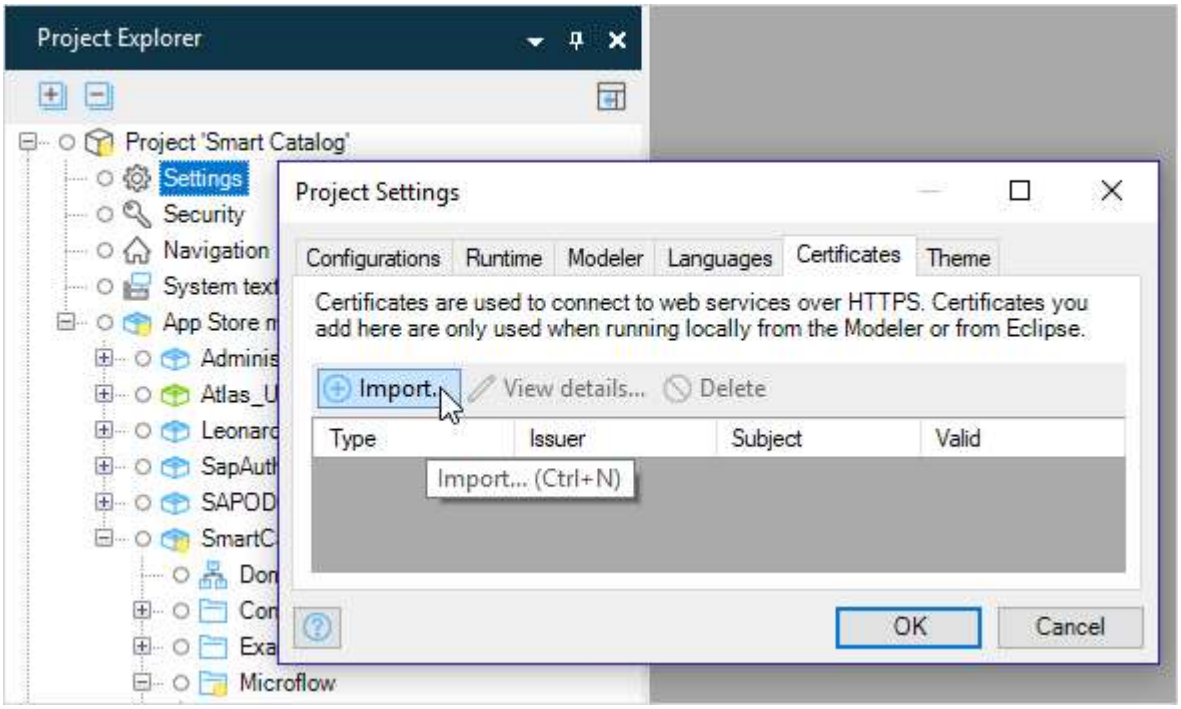


The microflow now looks like this:

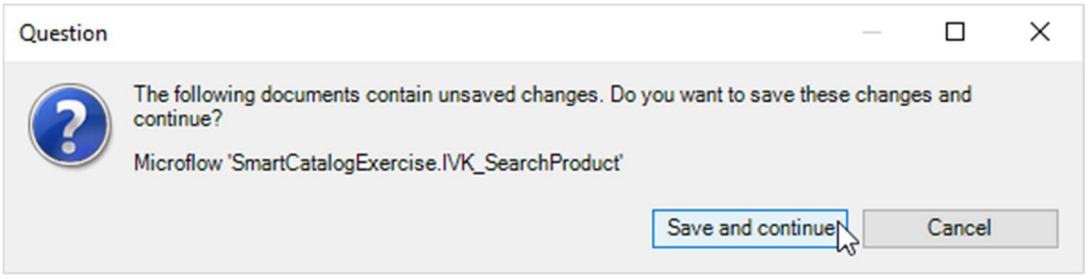


10 Testing the Integration with SAP

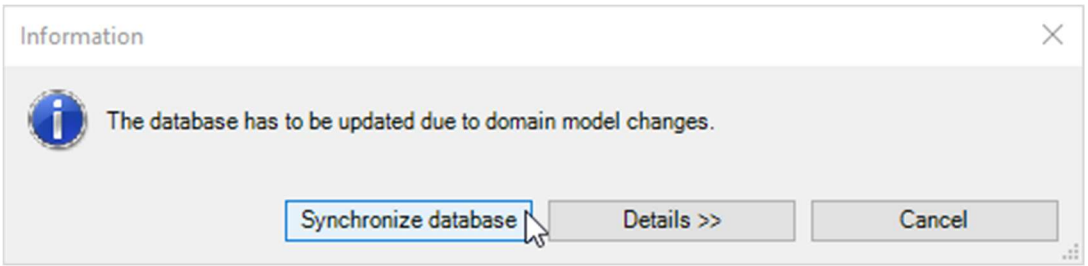
1. Right-click **Project 'Smart Catalog' > Settings** and click **Open** to open the project settings.
2. In the **Certificates** tab click **Import**.



3. Browse to the "Student (Local)" folder > CNA369.
4. Select the certificate file.
5. Click **Open**.
6. Click **OK** to close the dialog.
7. Click **Run Locally**.
8. Click **Save and continue** if you are asked to save changes and continue.

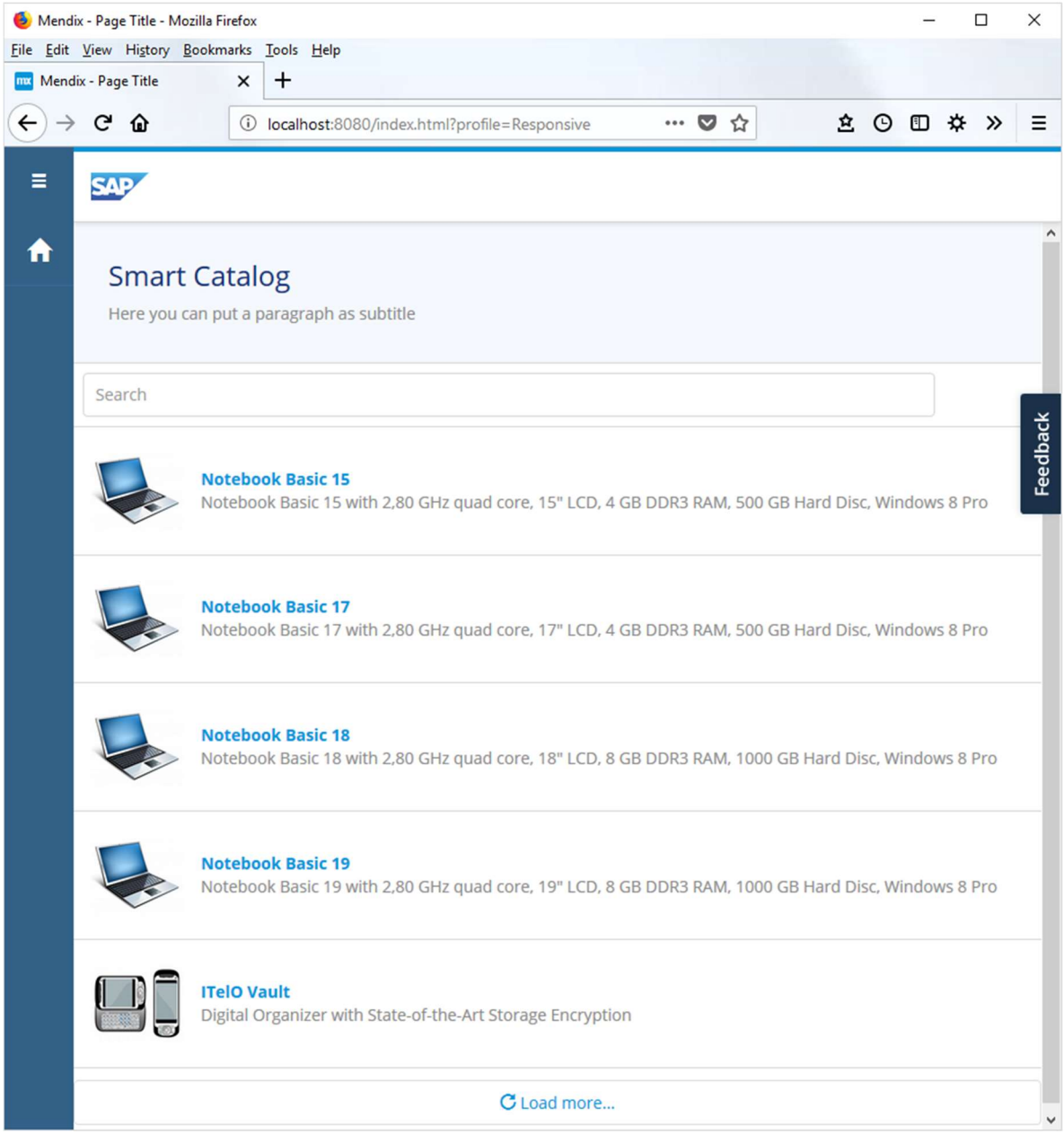


9. Click **Synchronize database** if you are told that the database has to be updated.



10. Wait until the runtime has been started successfully.
11. Click **View** to see the app running in a browser.

The app looks like this:

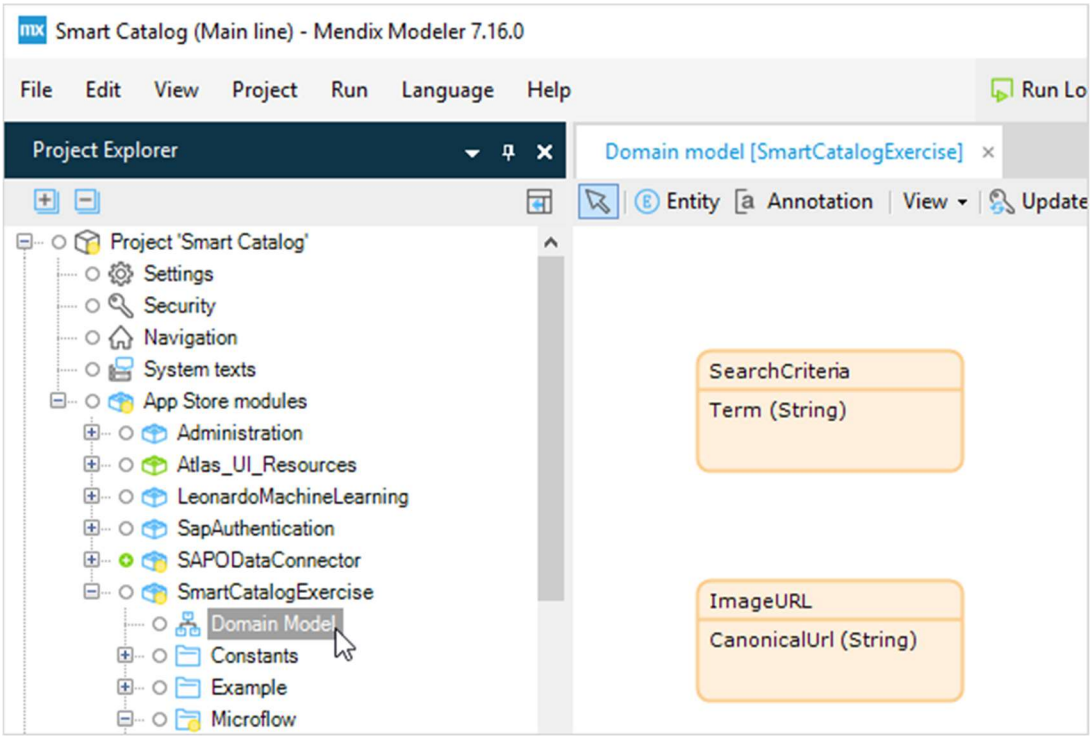


11 Adding Image-based Searching

You will now add the ability to search the catalog by image.

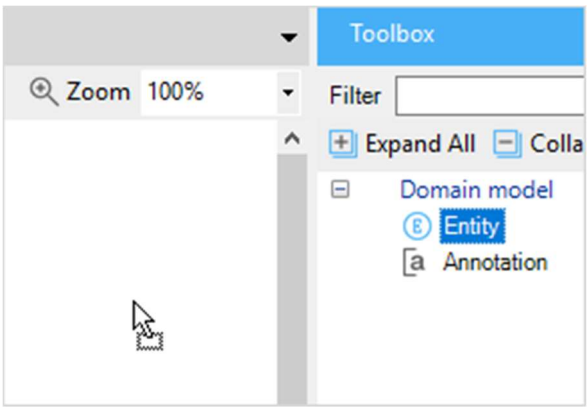
11.1 Adding Images to the App

1. Double-click **SmartCatalogExercise > Domain Model** to open the module's domain model.

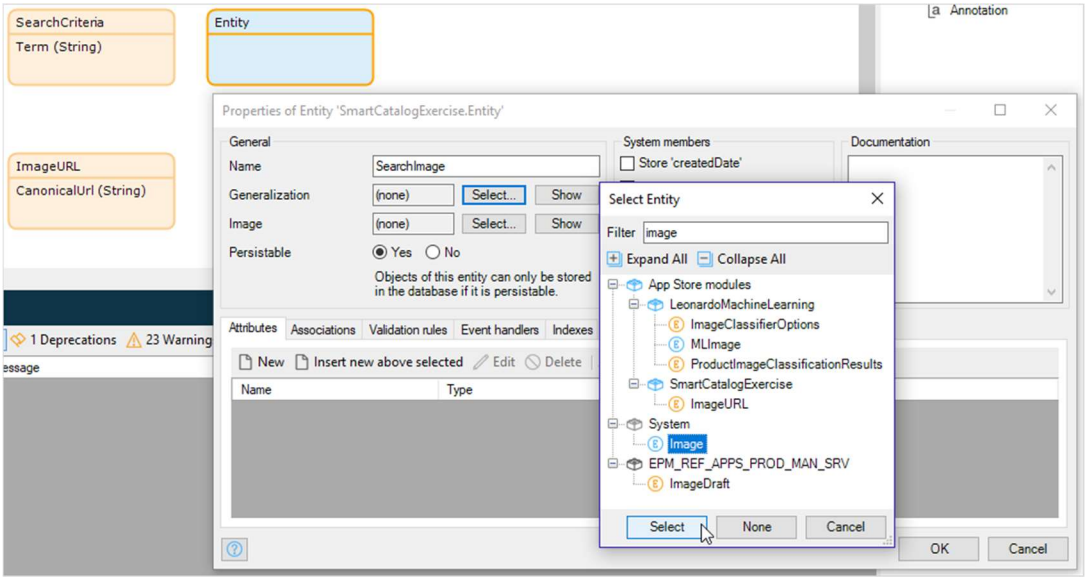


Firstly, you need to add an entity to the domain model to store the image you are going to use later to search the product list.

2. Drag a new **Entity** into the domain model.



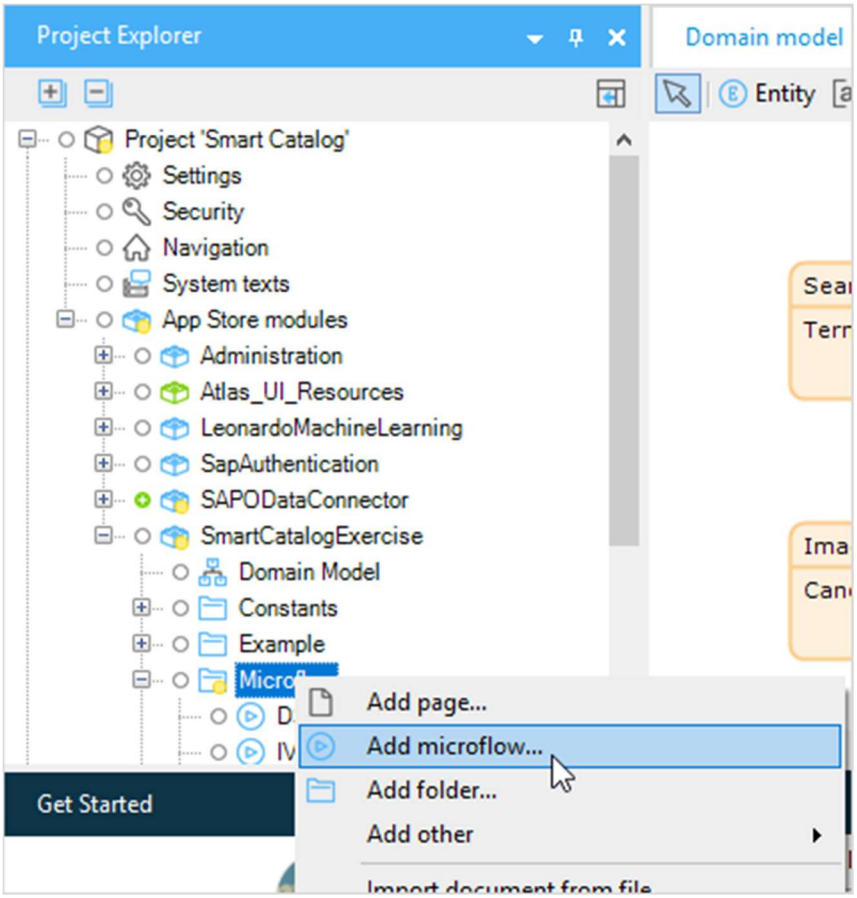
3. Double-click the new entity to open the properties.
4. Enter *SearchImage* as the **Name**.
5. Click **Select...** for the **Generalization**.
6. Select **System > Image**.
7. Click **Select**.



8. Click **OK** to close the properties dialog box.

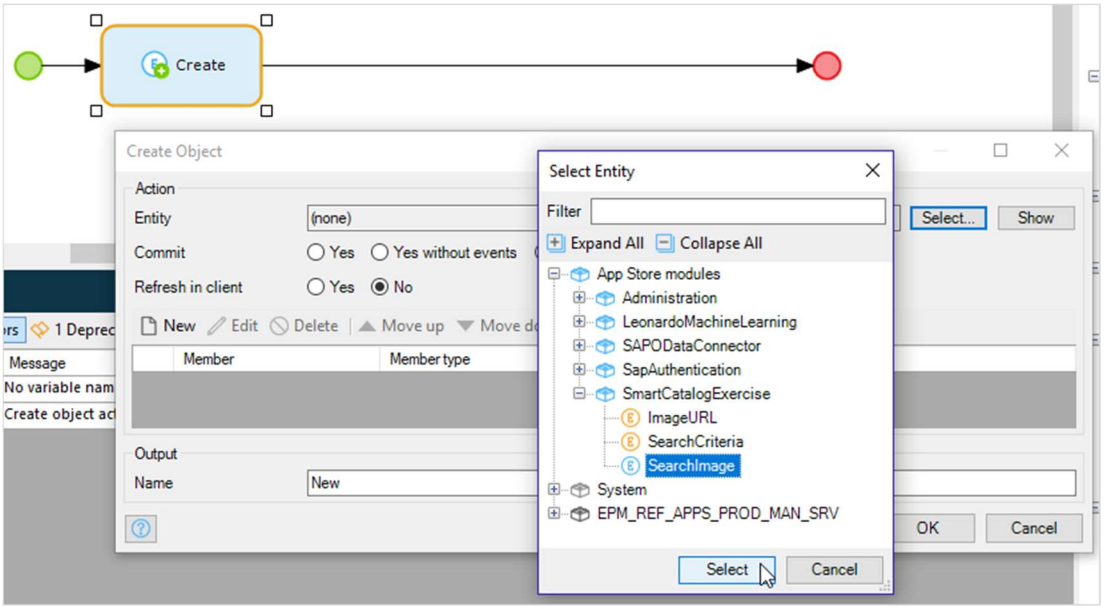
Now you create an empty *SearchImage* object which will then have the image put into it.

9. Right-click **Project 'Smart Catalog' > App Store modules > SmartCatalogExercise > Microflow** in the **Project Explorer**.
10. Click **Add microflow...**.



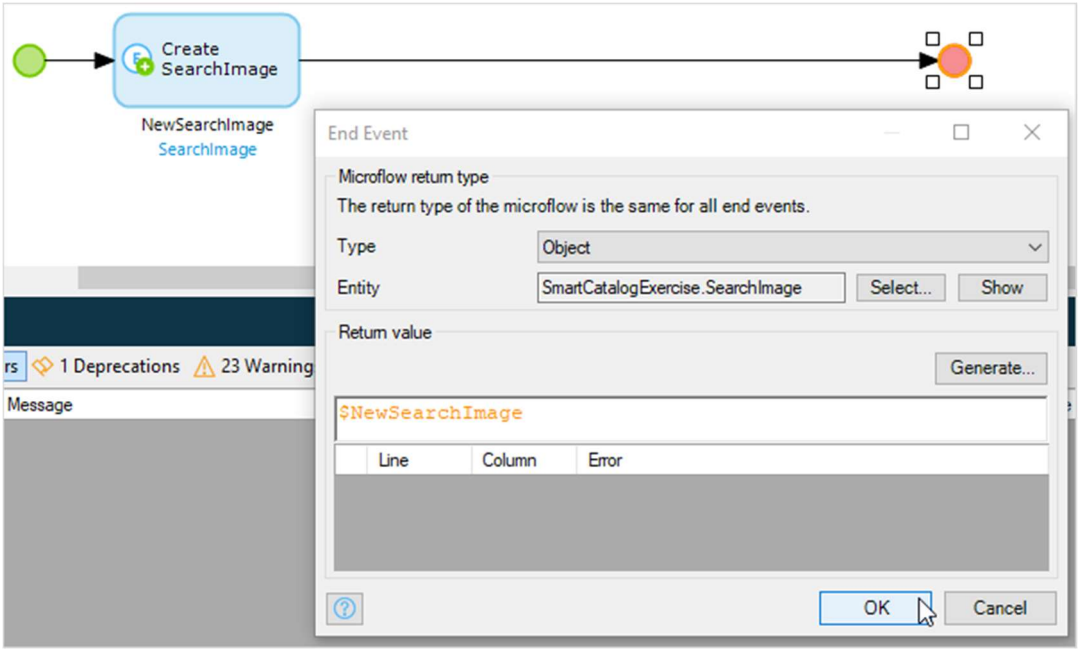
11. Enter *DS_NewEmptyImage* as the **Name**. (The DS_ prefix indicates that the microflow is acting as a *Data Source* - this is good practice to help future developers who extend your app).
12. Click **OK**.
13. Drag a **Create Object** action into the microflow.

14. Double-click the **Create** action.
15. Click **Select** for the **Entity**.
16. Select **App Store modules > SmartCatalogExercise > SearchImage**.
17. Click **Select**.

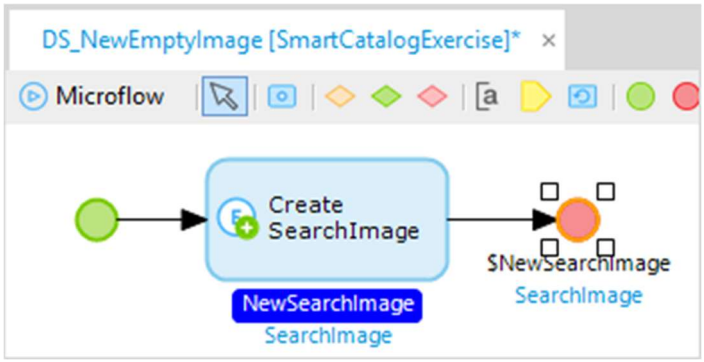


The output **Name** will be changed to **NewSearchImage**.

18. Click **OK**.
19. Double-click the red dot (the end event) of the microflow.
20. Select **Object** from the **Type** dropdown.
21. Select *App Store modules* > *SmartCatalogExercise* > *SearchImage* for the **Entity**.
22. Enter *$NewSearchImage* as the **Return Value**.
23. Click **OK**.



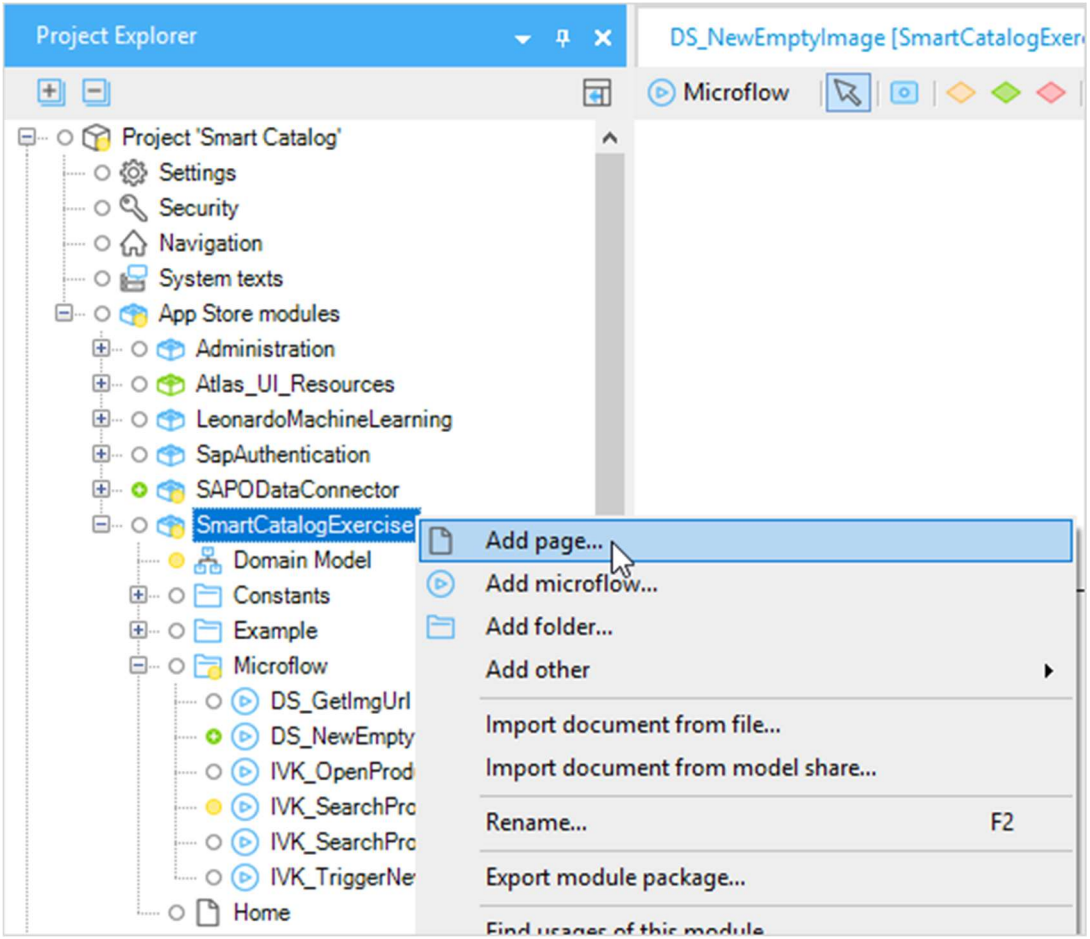
The microflow to create the empty *SearchImage* looks like this:



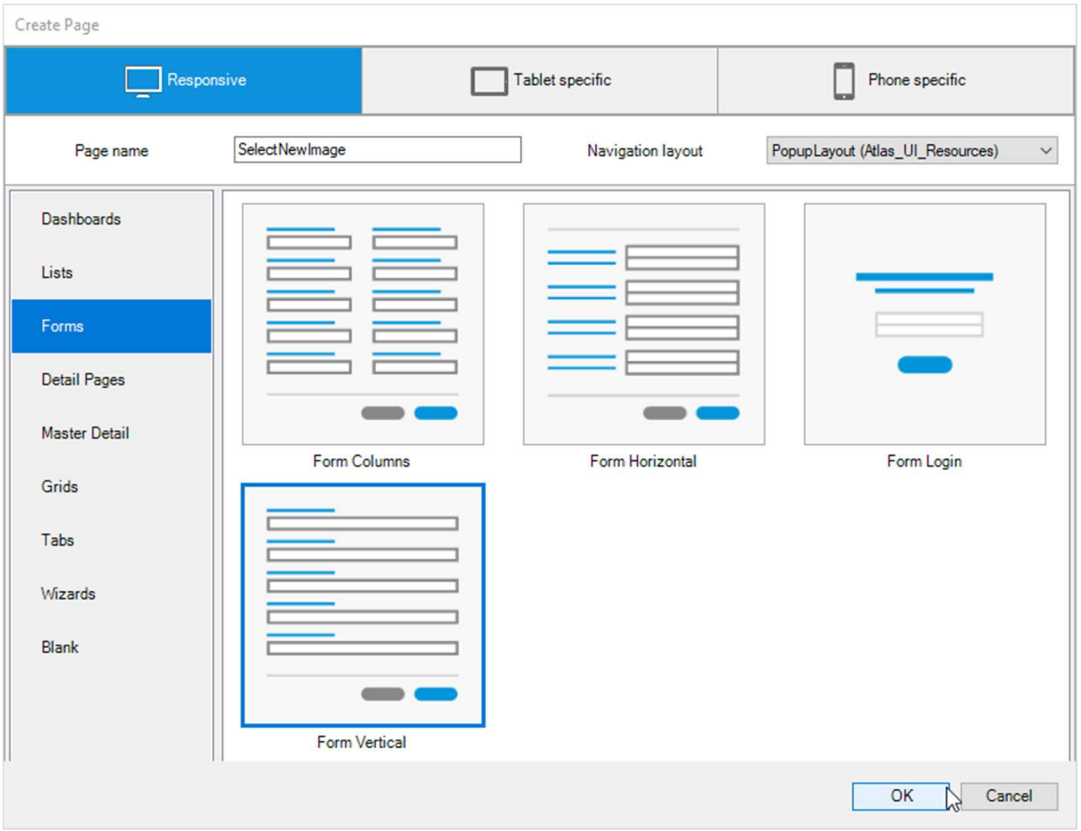
24. Right-click **Project 'Smart Catalog' > App Store modules > SmartCatalogExercise** in the **Project Explorer**.

Now you are going to add a new page which will be used when the user chooses to search by image.

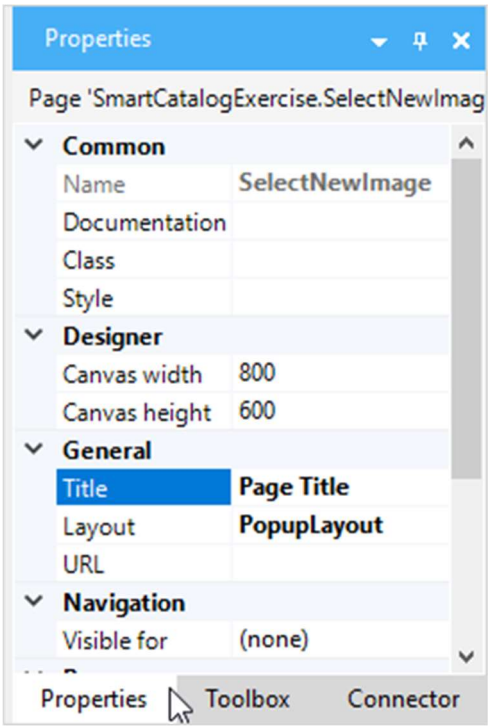
25. Click **Add page....**



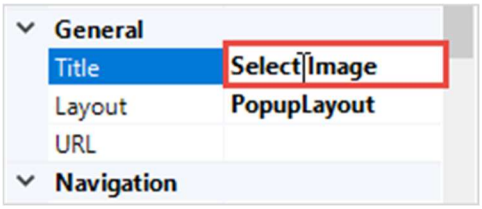
26. Enter *SelectNewImage* as **Page name**.
27. Select *PopUpLayout (Atlas_UI_Resources)* as **Navigation layout**.
28. Select **Forms > Form Vertical**.
29. Click **OK**.



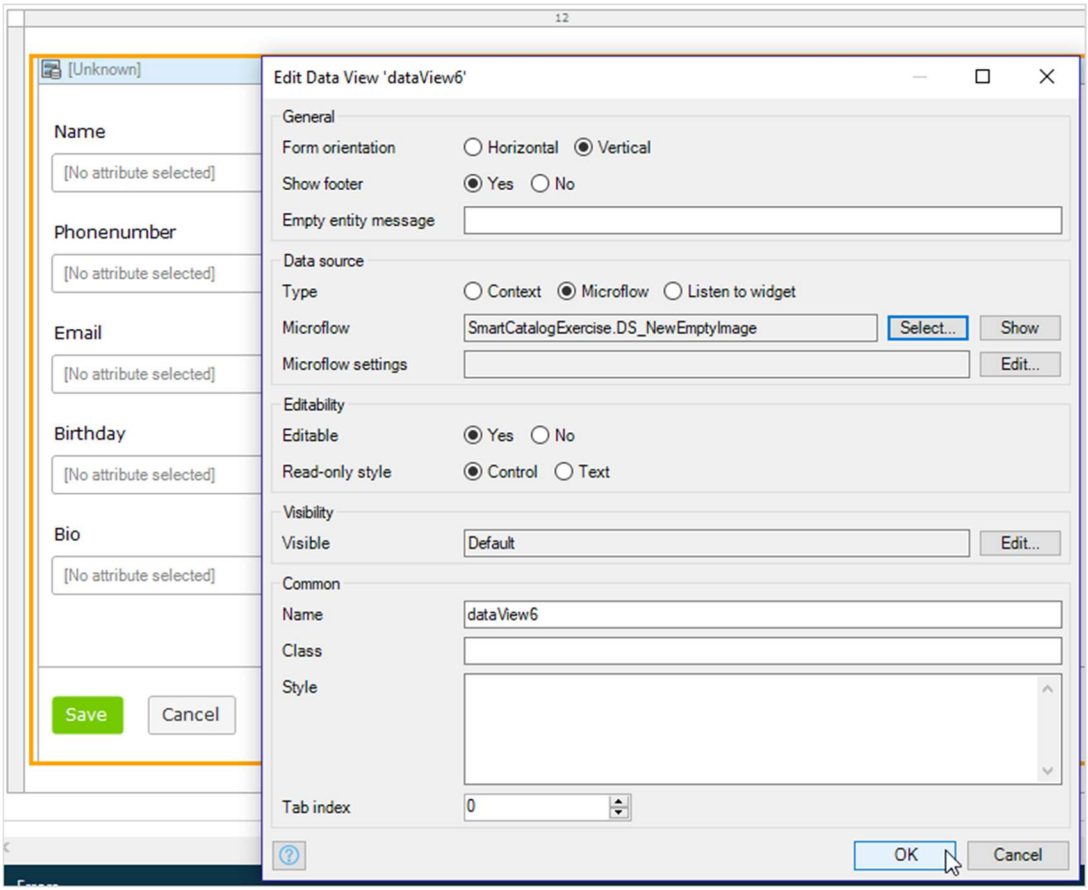
30. Click **Properties** in the right-hand pane.



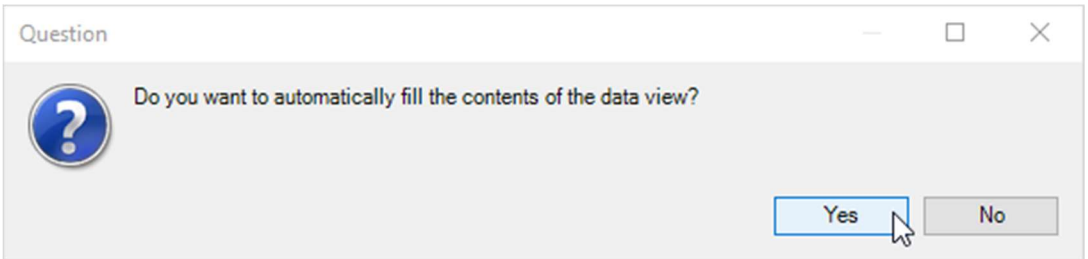
31. Enter *Select Image* as the **Title**.



32. Double-click the data view on the page (the blue heading currently displaying **(Unknown)**).
33. Select **Microflow** as the **Data source > Type**.
34. Select the **DS_NewEmptyImage** microflow.
35. Click **OK**.



36. Click **Yes** for the question **Do you want to automatically fill the contents of the data view?.**



37. Click the **Name** field on the page.

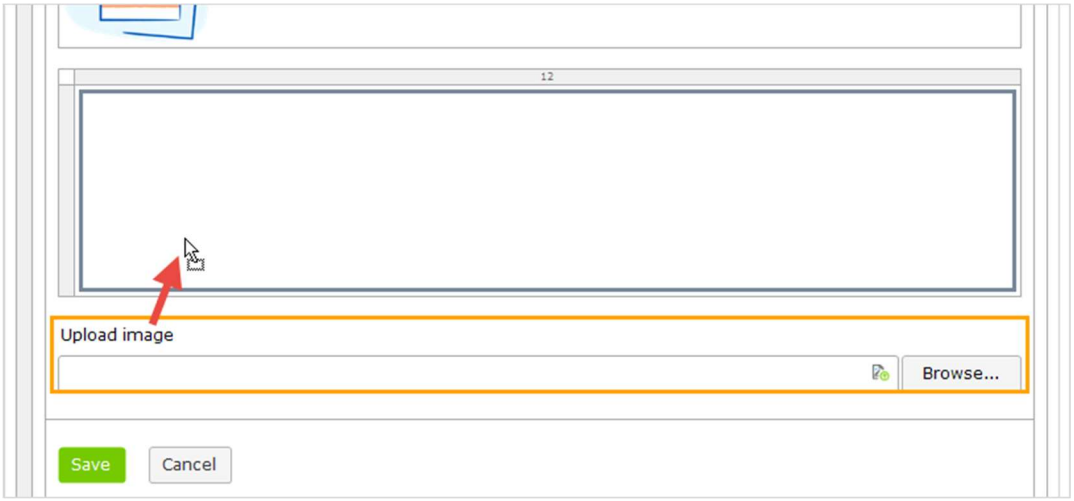
38. Press `Delete` to delete this field.
39. Delete the **Size** field.
40. Switch back to the **Toolbox** in the right-hand pane.

You need to add a camera widget to the page, so that users who are using a mobile device can use their camera to capture an image.

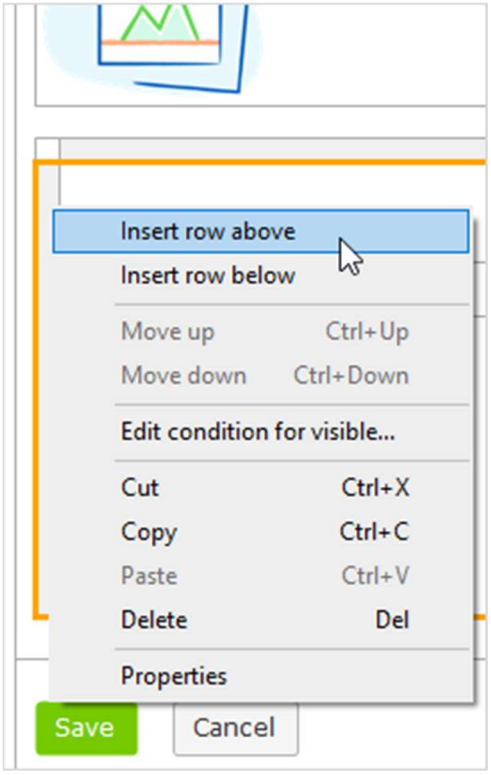
41. Drag a **Layout grid** into the drop zone under the image.

42. Select the **full** grid format.

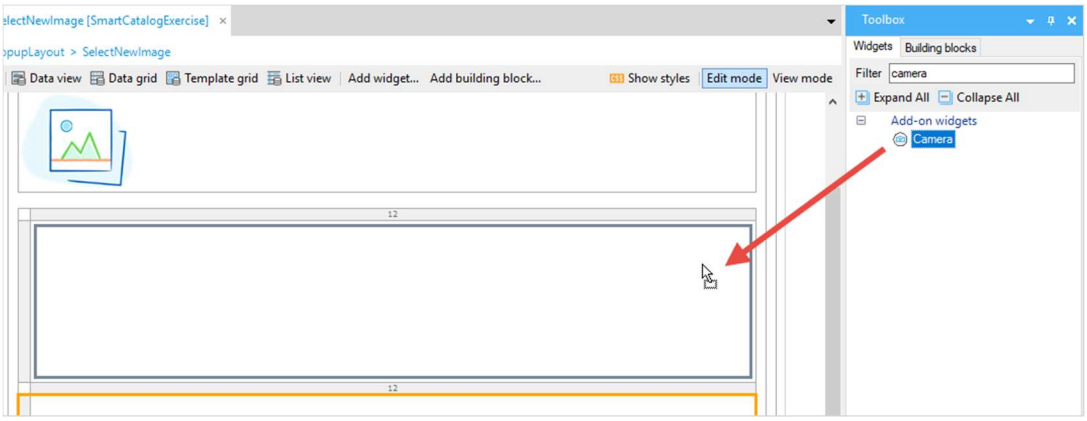
43. Drag the **Upload image** widget on the screen into the layout grid.



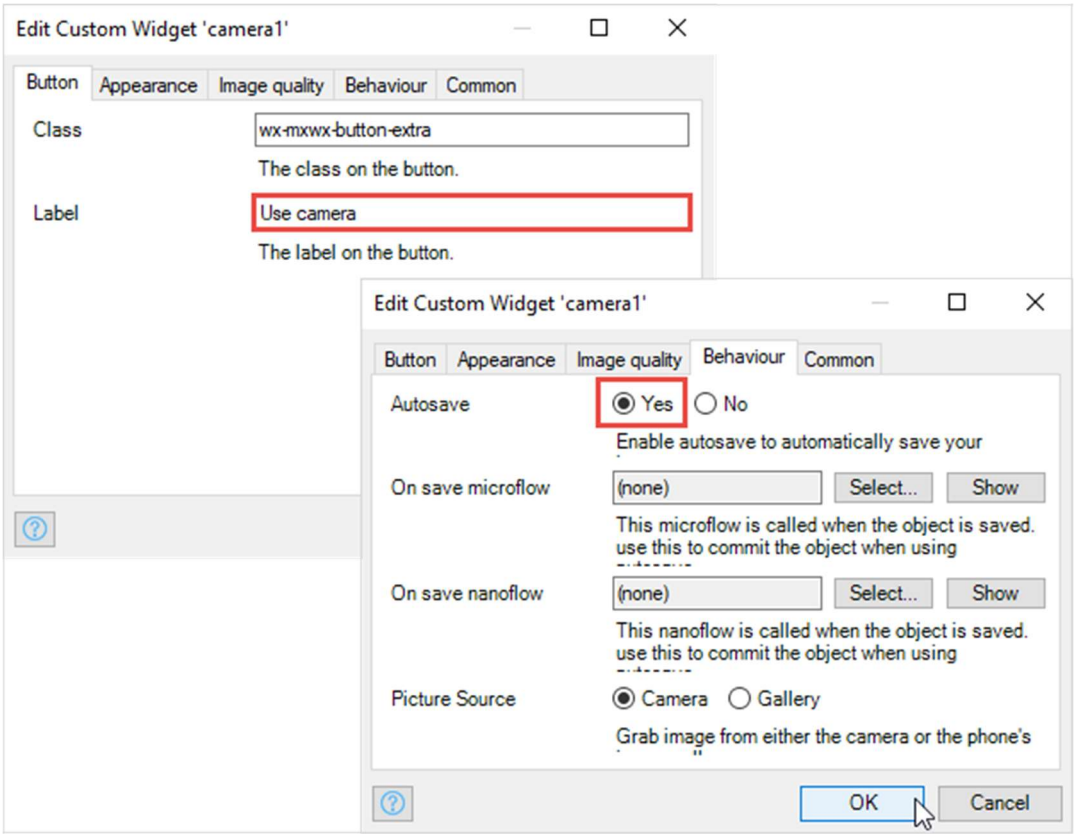
44. Right-click the left-hand side of the layout grid.
45. Click **Insert row above**.



46. Click **Full**.
47. Drag a **Camera** widget into the new row.



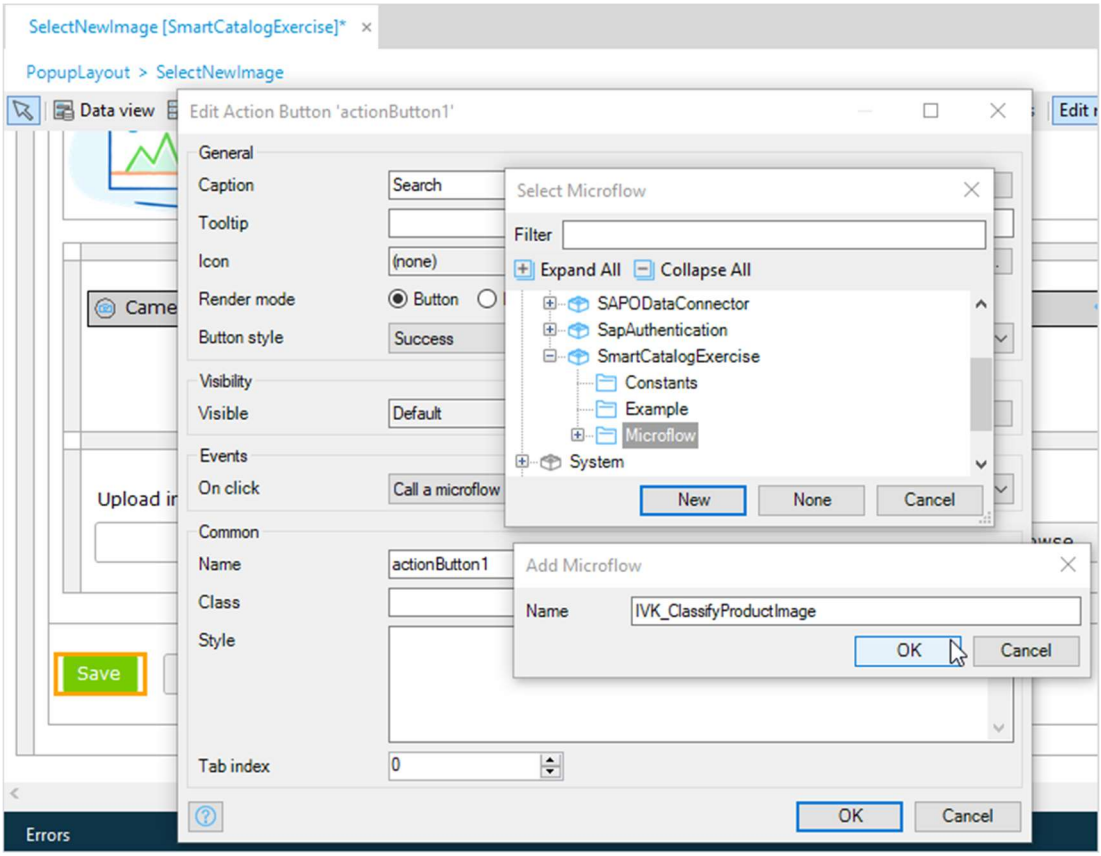
48. Double-click the **Camera** widget to open the properties.
49. Enter *Use Camera* as the **Label** in the **Button** tab.
50. Select **Yes** for **Auto save** in the **Behavior** tab.
51. Click **OK**.



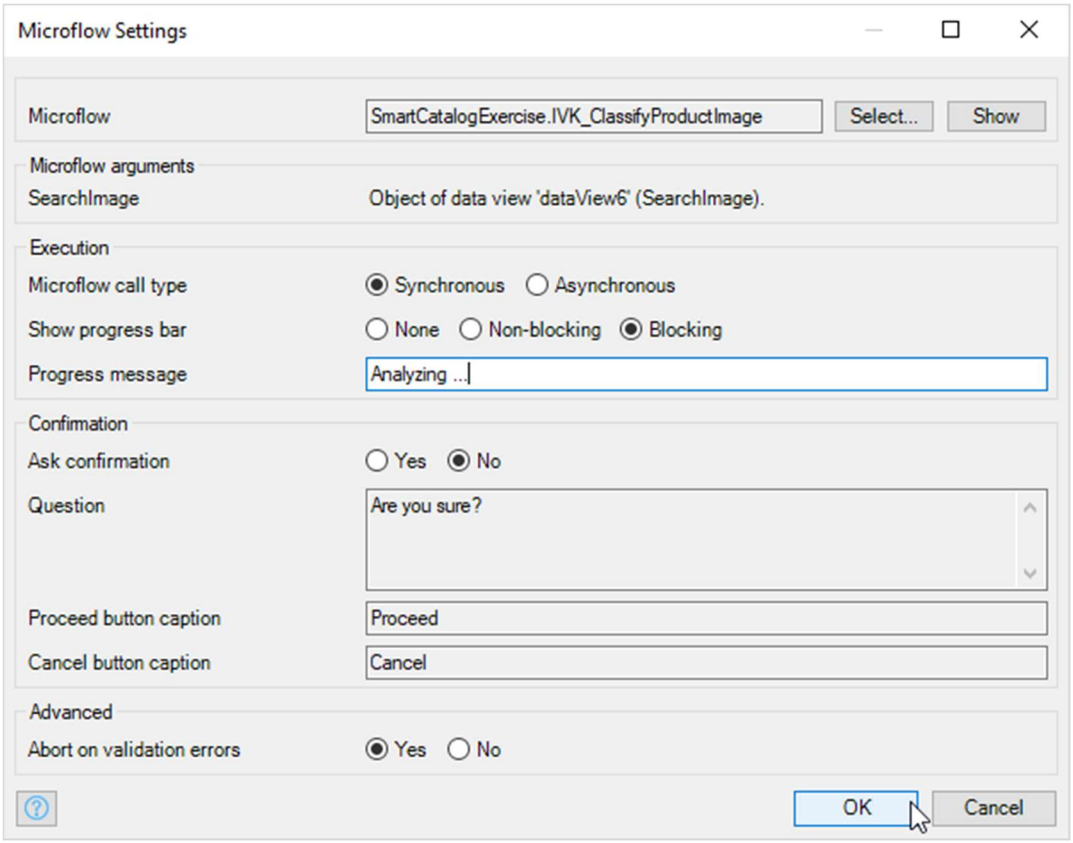
52. Double-click the **Save** button.
53. Enter *Search* as the **Caption** on the button.
54. Select **Call a microflow** for the **On click** action.
55. Click the **SmartCatalogExercise > Microflow** folder.

Now you will create a microflow to use SAP Leonardo Machine Learning Foundation services to classify the image.

56. Select a **New** microflow.
57. Enter *IVK_ClassifyProductImage* as the **Name**.
58. Click **OK**.

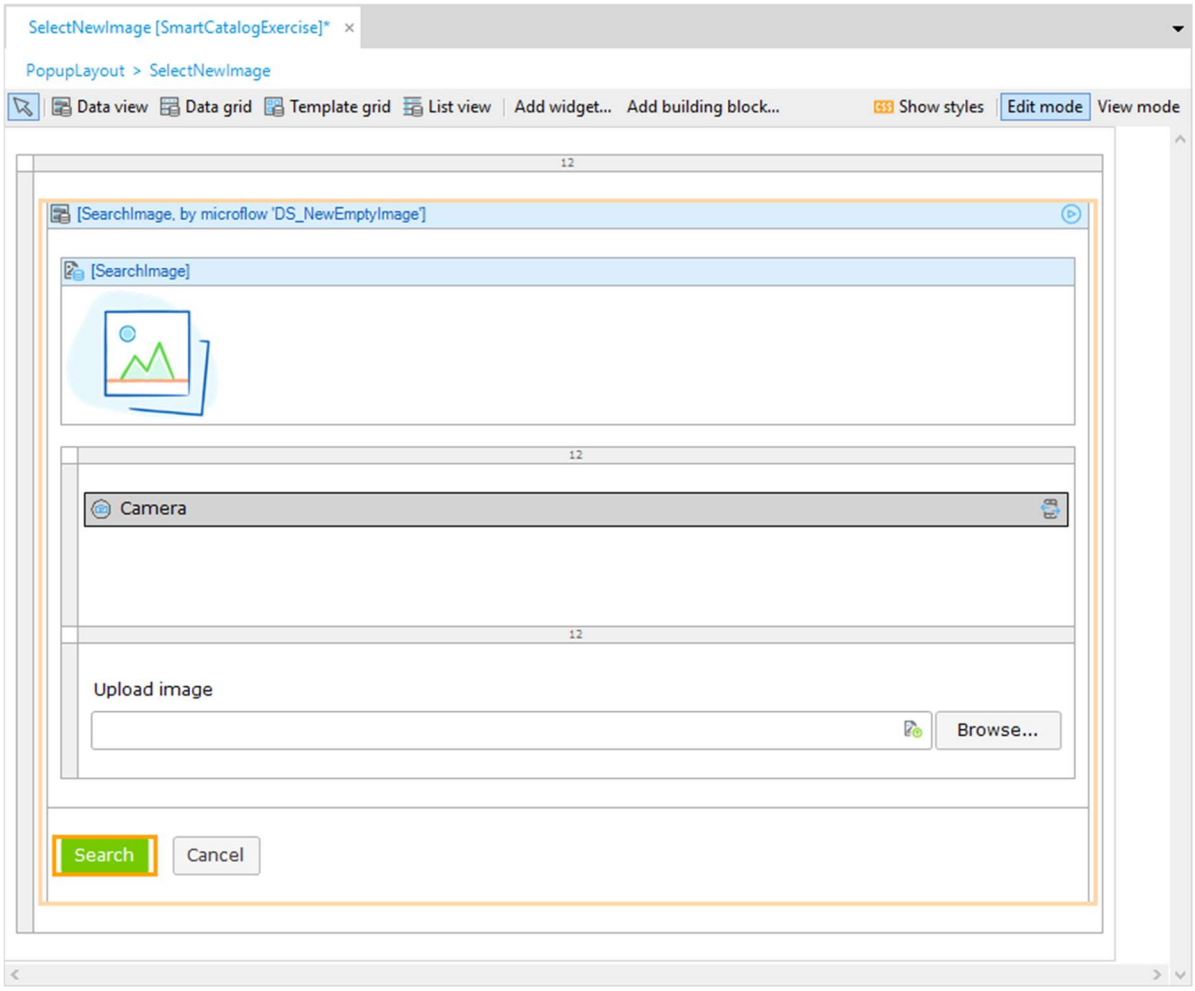


59. Click Edit... for Microflow Settings.
60. Select Blocking for Show progress bar.
61. Enter *Analyzing ...* as the Progress message.
62. Click OK.

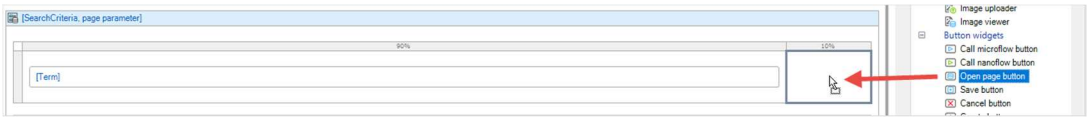


63. Click **OK** to close the **Edit Action Button** dialog.

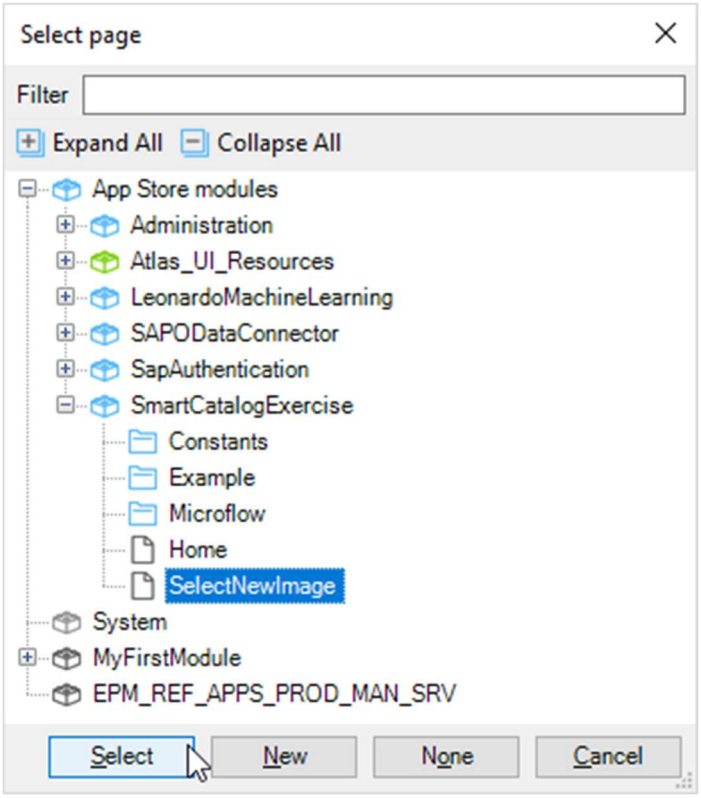
The page will look like this:



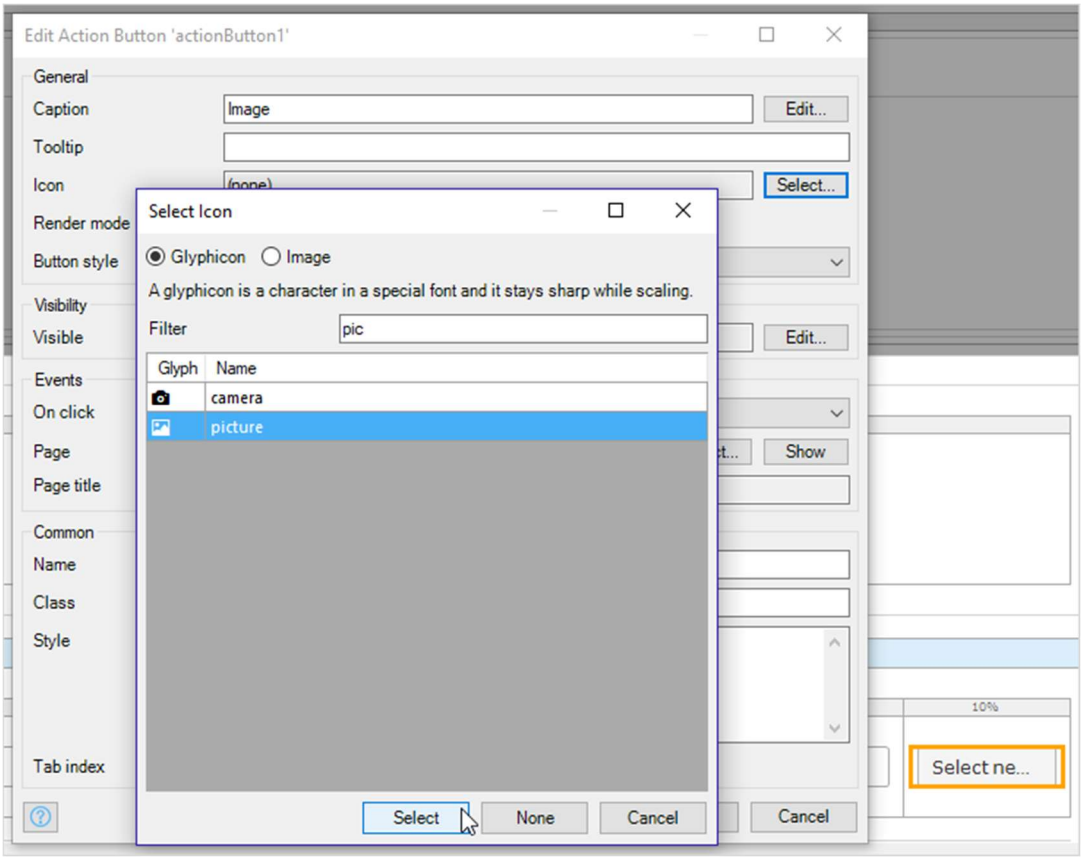
64. Double-click **Project 'Smart Catalog' > App Store modules > SmartCatalogExercise > Home** in the Project Explorer to open it.
65. Drag an **Open page button** next to the search bar.



66. Click **App Store modules > SmartCatalogExercise > SelectNewImage**.
67. Click **Select**.

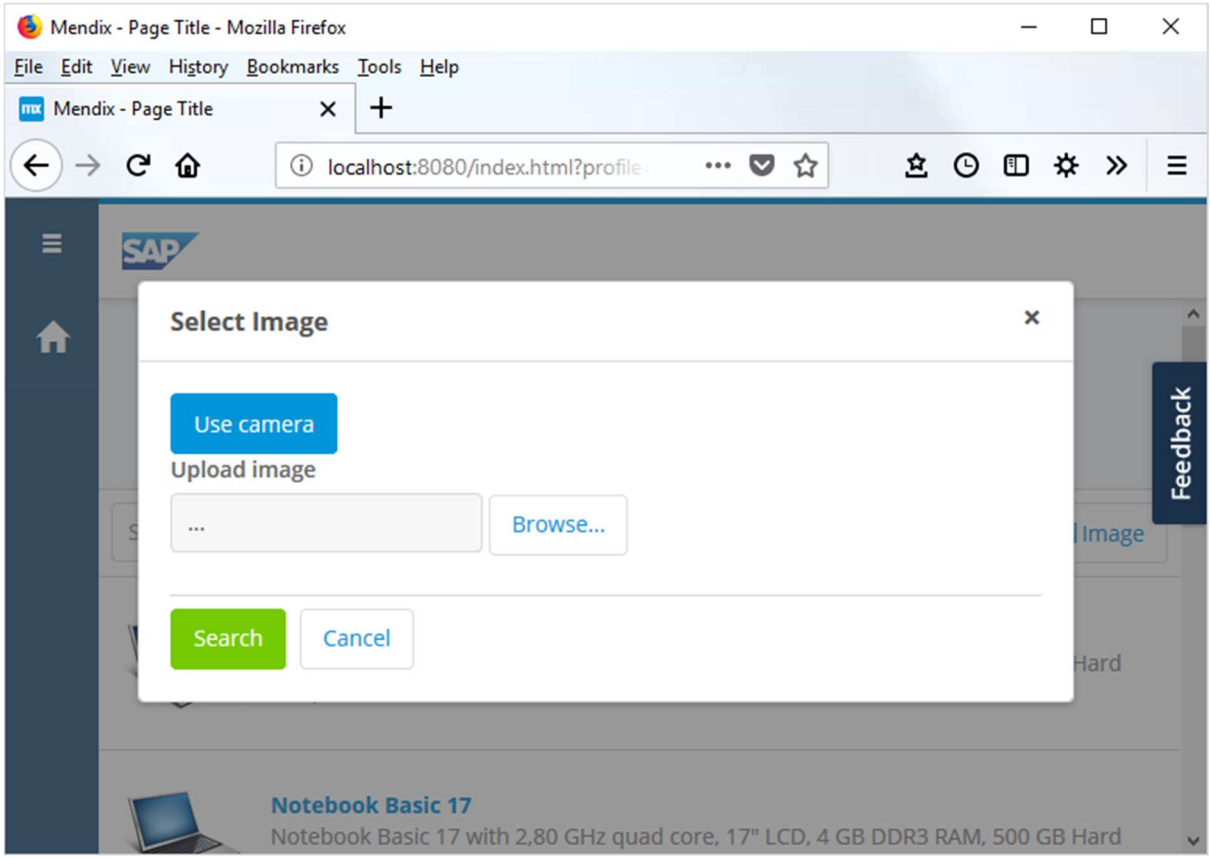


68. Double-click the new button.
69. Enter *Image* as the **Caption**.
70. Click **Select...** for the **Icon**.
71. Select the **picture** icon.
72. Click **Select**.



73. Click **OK**.
74. Click **Run Locally**.
75. Click **View** when the runtime has been started successfully.

You can now see the changes to the app. There is an **Image** button which you can click to see the **Select Image** pop-up.



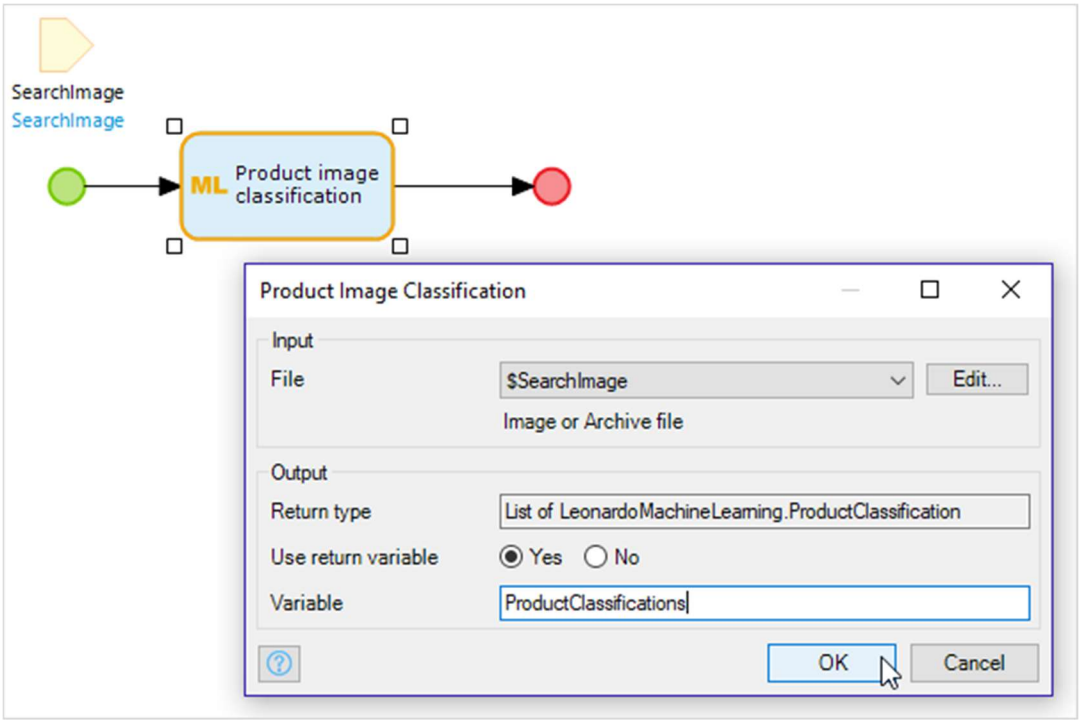
Now you need to use SAP Leonardo Machine Learning to make your app 'smart'.

11.2 Making your App 'Smart' Using the SAP Leonardo Machine Learning Foundation Service

1. Double-click the microflow **IVK_ClassifyProductImage** to open it.
2. Drag an **SAP Leonardo machine learning (API Business Hub) > Product Image Classification** action into the microflow.

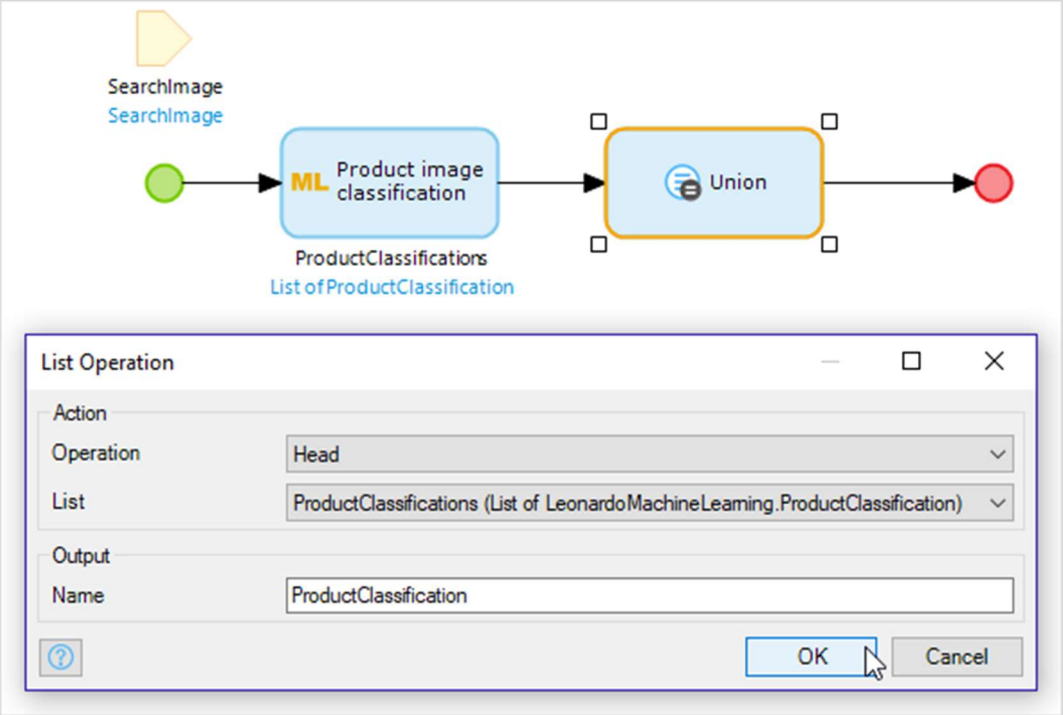
The *SAP Leonardo machine learning* widgets in the *API Business Hub* section are only available via the *SAP API Business Hub*. Other *SAP Leonardo machine learning* widgets are available both in the *SAP API Business Hub* and in the cloud.

3. Double-click the **Product Image Classification** action.
4. Select *$SearchImage* as the **File**.
5. Enter *ProductClassifications* as the **Variable**.
6. Click **OK**.



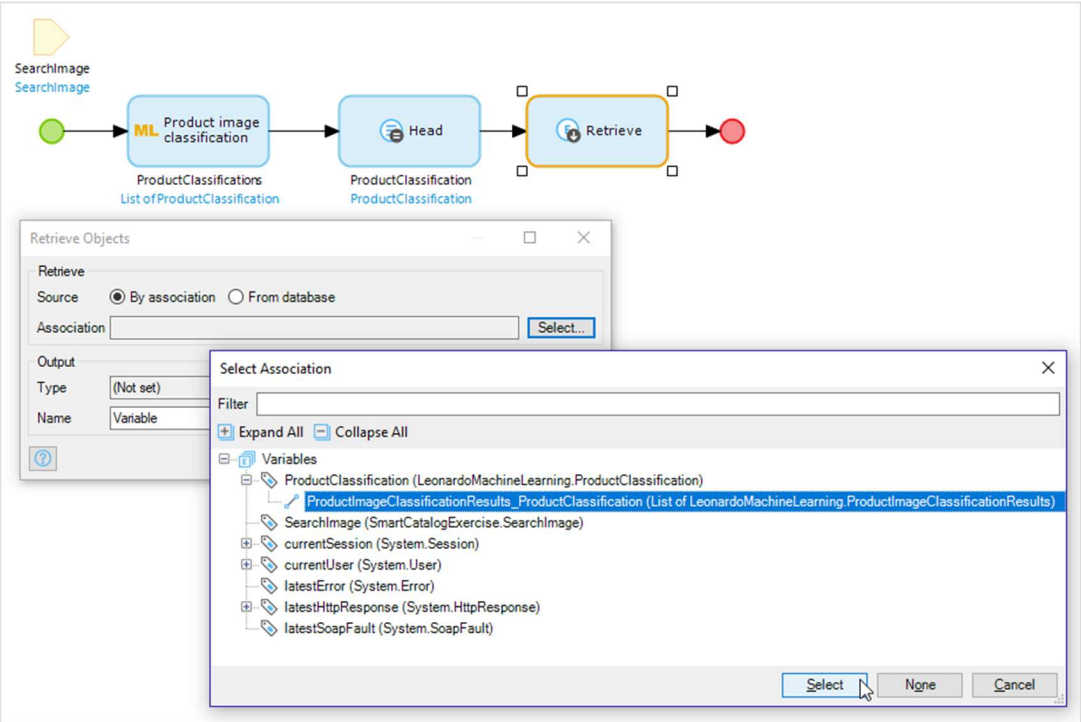
7. Drag a **List activities > List operation** action as the second action in the microflow. (It will be labeled **Union**).

8. Double-click the list operation.
9. Select *Head* as the **Operation**. This selects just the first item in the list.
10. Select *ProductClassifications (List of LeonardoMachineLearning.ProductClassification)* as the **List**.
11. Click **OK**.



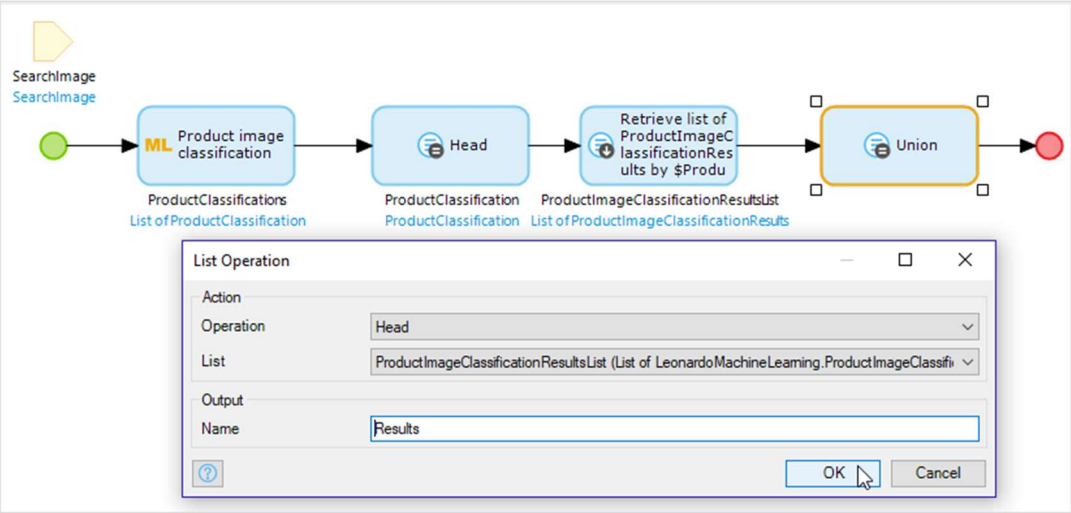
12. Drag an **Object activities > Retrieve** action as the third action in the microflow.

13. Double-click the **Retrieve** action.
14. Select **By association** as **Source**.
15. Click **Select...** for **Association**
16. Select **Variables > ProductClassification (LeonardoMachineLearning.ProductClassification) > ProductImageClassiciationResults_ProductClassification (List of LeonardoMachineLearning.ProductImageClassificationResults)**.
17. Click **Select**.

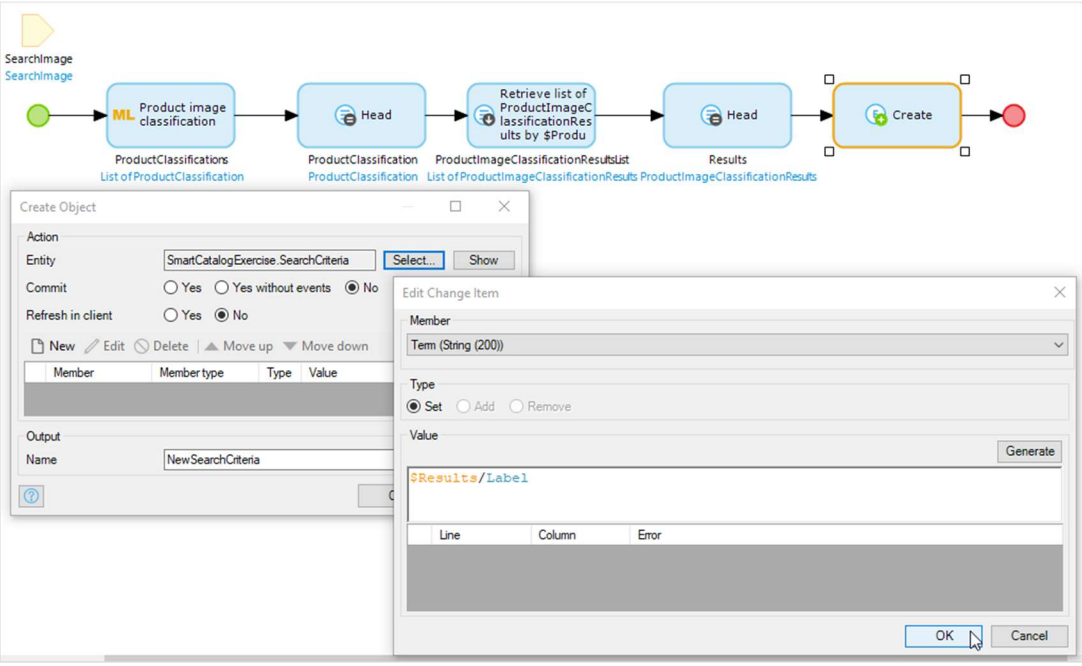


18. Click **OK** to close the dialog.
19. Drag a **List activities > List operation** action as the fourth action in the microflow.

20. Double-click the list operation.
21. Select *Head* as the **Operation**.
22. Select *ProductImageClassificationResultsList (List of LeonardoMachineLearning.ProductImageClassificationResults)* as the **List**.
23. Enter *Results* as the **Name**.
24. Click **OK**.

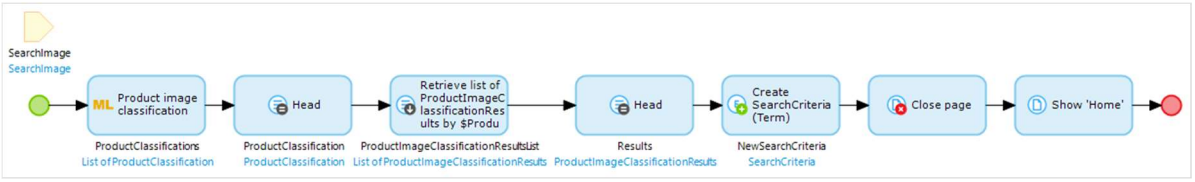


25. Drag a new **Create object** action as the fifth action in the microflow.
26. Double-click the **Create** action.
27. Click **Select...** for the **Entity**.
28. Select *App Store modules* > *SmartCatalogExercise* > *SearchCriteria*.
29. Click **Select**.
30. Click **New**.
31. Select *Term* as the **Member**.
32. Enter *$Results/Label* as the **Value**.
33. Click **OK**.



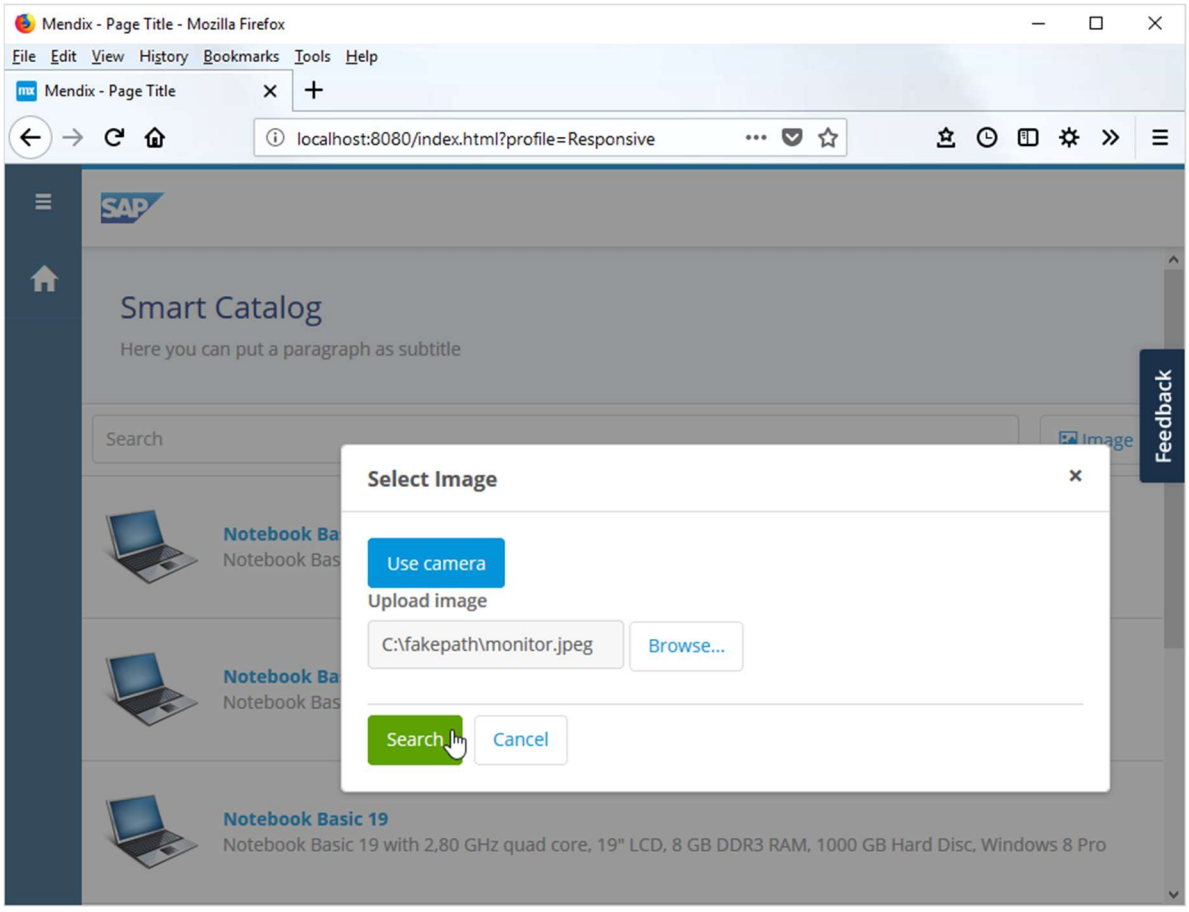
34. Click **OK** to close the **Create Object** dialog.
35. Drag a **Close page** action as the sixth action in the microflow.
36. Drag a **Show page** action as the last action in the microflow.
37. Double-click the **Show page** action.
38. Select **NewSearchCriteria (SmartCatalogExercise.SearchCriteria)** from the **Object to pass** dropdown.
39. Click **Select...** for the **Page**.
40. Select the **App Store modules > SmartCatalogExercise > Home** page.
41. Click **Select**.
42. Click **OK**.

Your microflow now looks like this:



43. Click Run Locally.
44. Click View when the runtime has been started successfully.

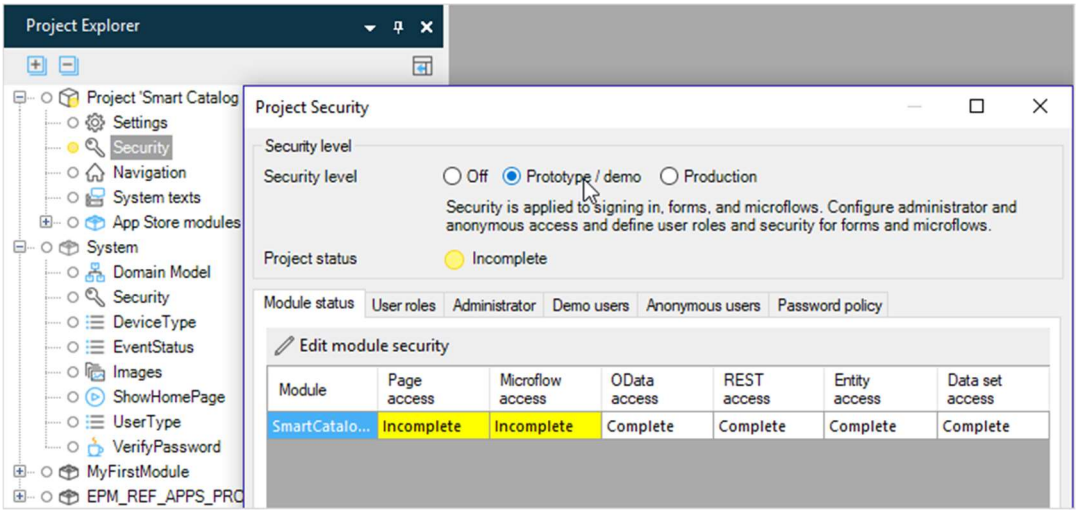
You can now see the changes to the app. Running in a browser, you cannot access a camera, but you could upload an image for SAP Leonardo Machine Learning Foundation Services to analyze.



12 Setting Security in your App

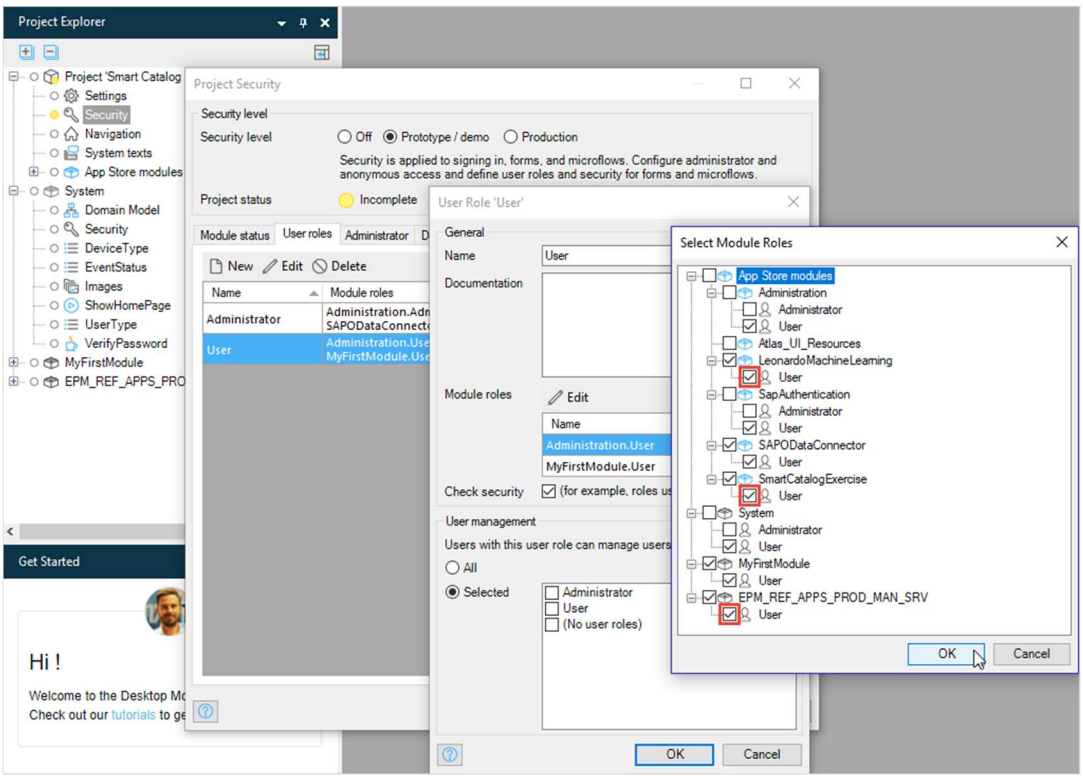
To deploy the app to SAP Cloud Platform and use the SAP Cloud Connector the security has to be set correctly. If the app is deployed with no security, it is not possible to use SAP credentials to logon and you cannot use non-public endpoints via the SAP Cloud Connector.

1. Double-click **Project `Smart Catalog' > Security** in the Project Explorer.
2. Click **Prototype / demo**.



You now need to set the *User role* **User** to have access to all the modules you have added to the app.

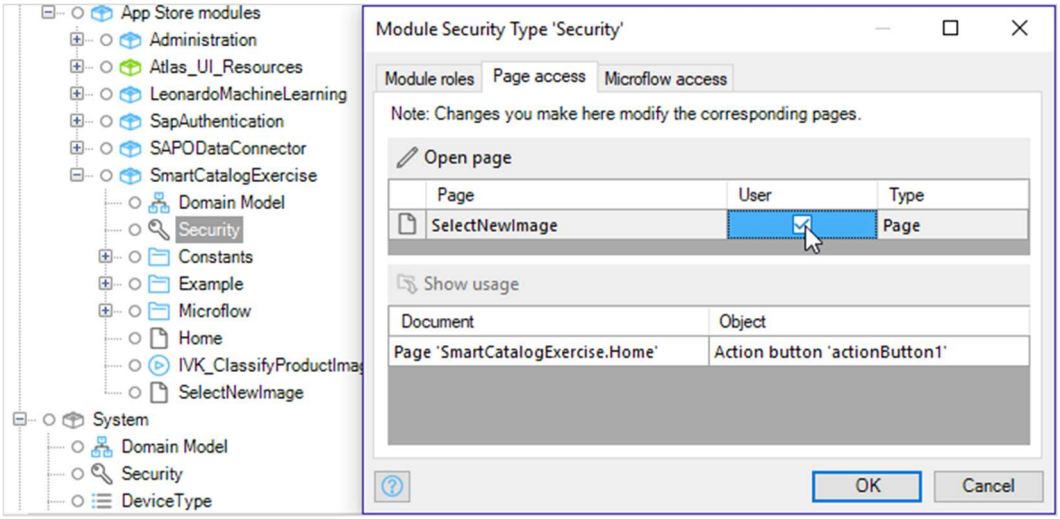
3. Click the **User roles** tab.
4. Select the role named **User**.
5. Click **Edit**.
6. Click **Edit** for **Module roles**.
7. Activate the **User** role for all the modules by clicking the tick boxes. The modules you need to change are:
    - LeonardoMachineLearning
    - SmartCatalogExercise
    - EPM_REF_APPS_PROD_MAN_SRV
8. Click **OK**



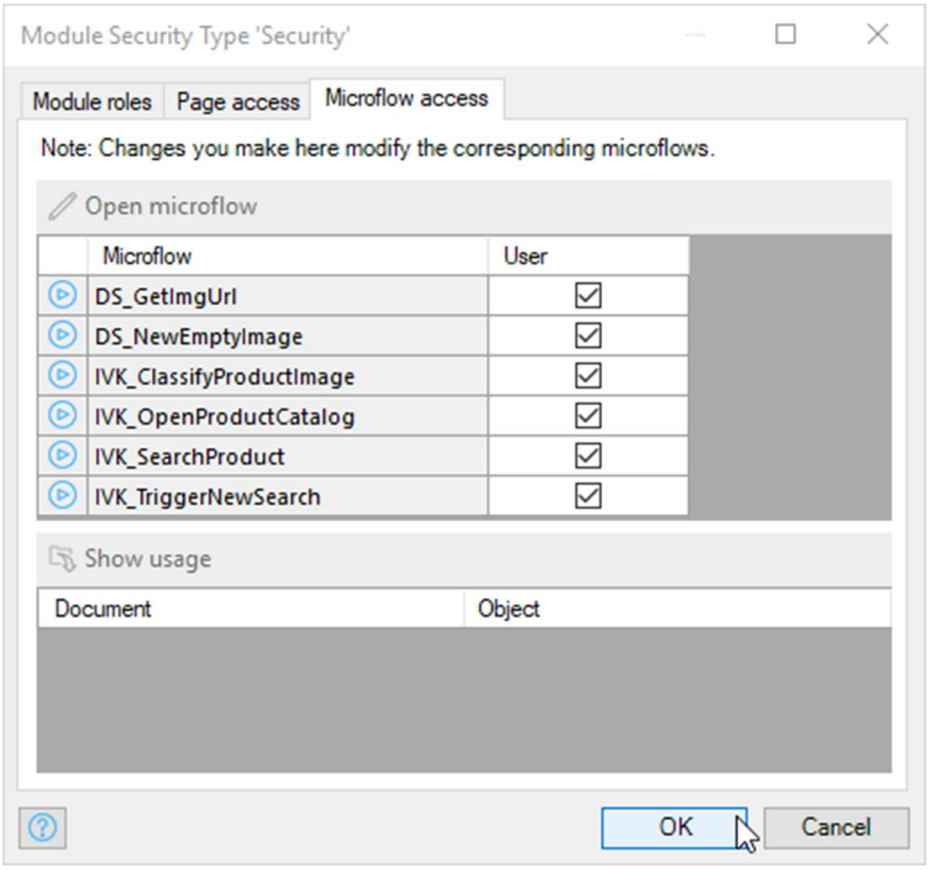
9. Click **OK** to close the **User Role 'User'** dialog.
10. Click **OK** to close the **Project Security** dialog.

You will see that the project now has some errors as the security settings will not allow some of the pages and microflows to be reached.

11. Double-click **Project 'Smart Catalog' > App Store modules > SmartCatalogExercise > Security**
12. Click the **Page access** tab.
13. Activate the **User** role for the *SelectNewImage* page by clicking the tick box.



14. Click the **Microflow access** tab.
15. Activate the **User** role for the *DS_NewEmptyImage* and *IVK_ClassifyProductImage* microflows by clicking the tick boxes.
16. Click OK to close the Module Security... dialog.



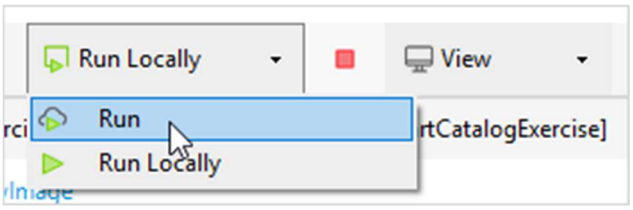
Your app is now error-free!

13 Deploying and Configuring Your App

You have tested your app running on your local machine. It is now time to deploy your app to SAP Cloud Platform. This will allow you to share your app with other users and utilize features such as SAP single sign-on and SAP Cloud Connector.

13.1 Deploying Your App

1. Click the arrow next to **Run Locally**.
2. Click **Run**.



This will deploy your app to SAP Cloud Platform.

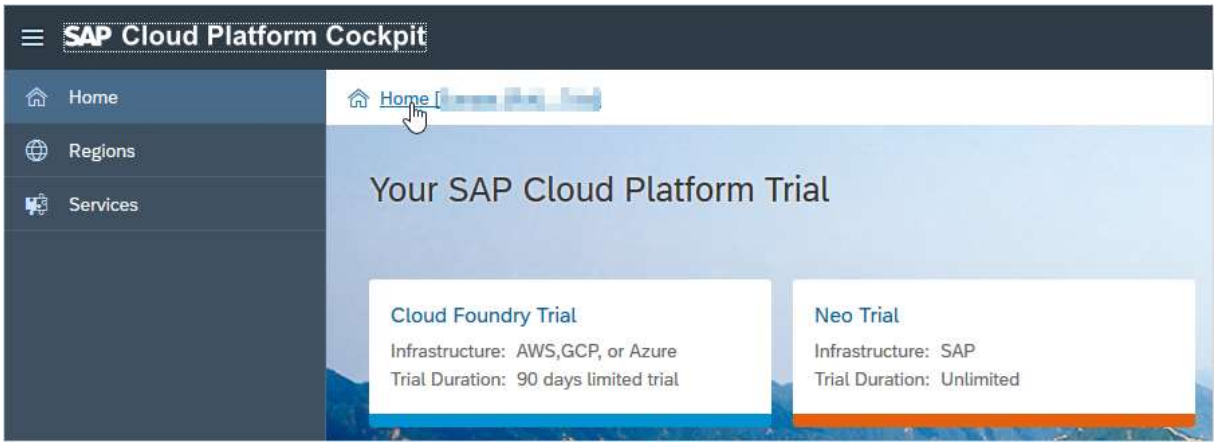
13.2 Configuring Your App

Because this is the first time you have deployed your app, there are a couple of items which need to be configured from within the SAP Cloud Platform Cockpit.

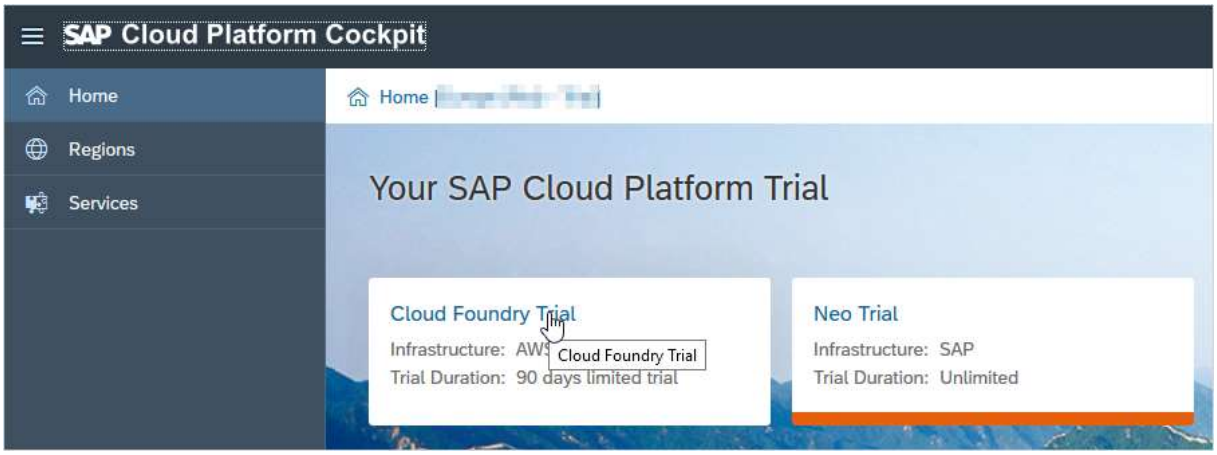
13.2.1 Configuring Single Sign On

No users have been given access to your app, so you need to add them.

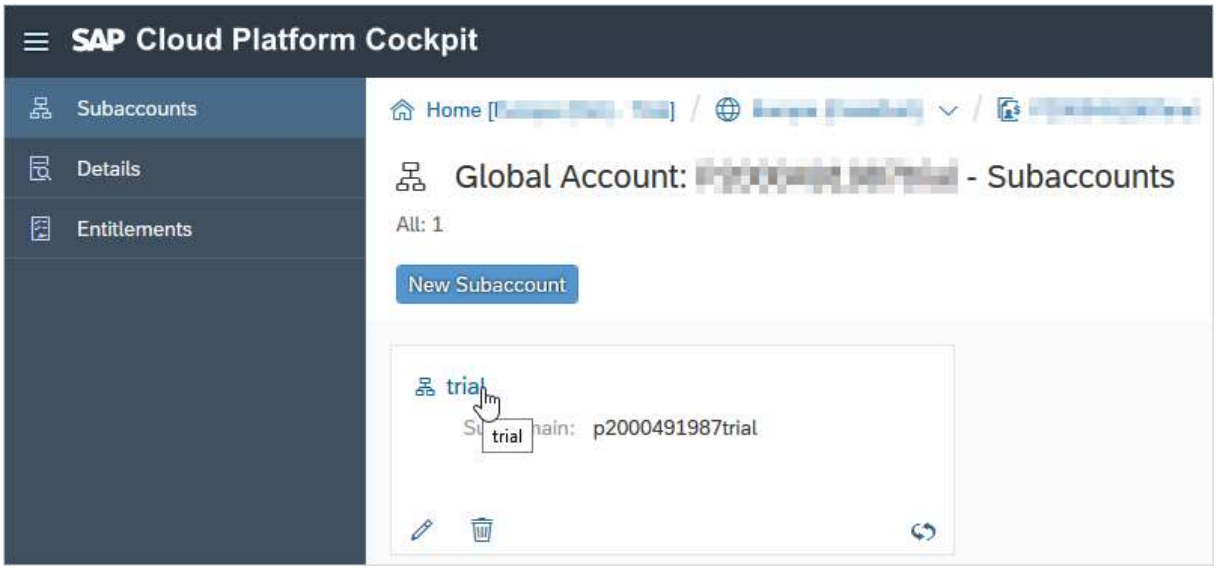
1. Return to the browser tab with the SAP Cloud Platform Cockpit open. If you have this, jump to step 4.
2. Open the SAP Cloud Platform Cockpit here: https://account.hanatrial.ondemand.com/cockpit.
3. Log On with your student credentials, if required.
4. Click **Home...** if you are not on the home page.



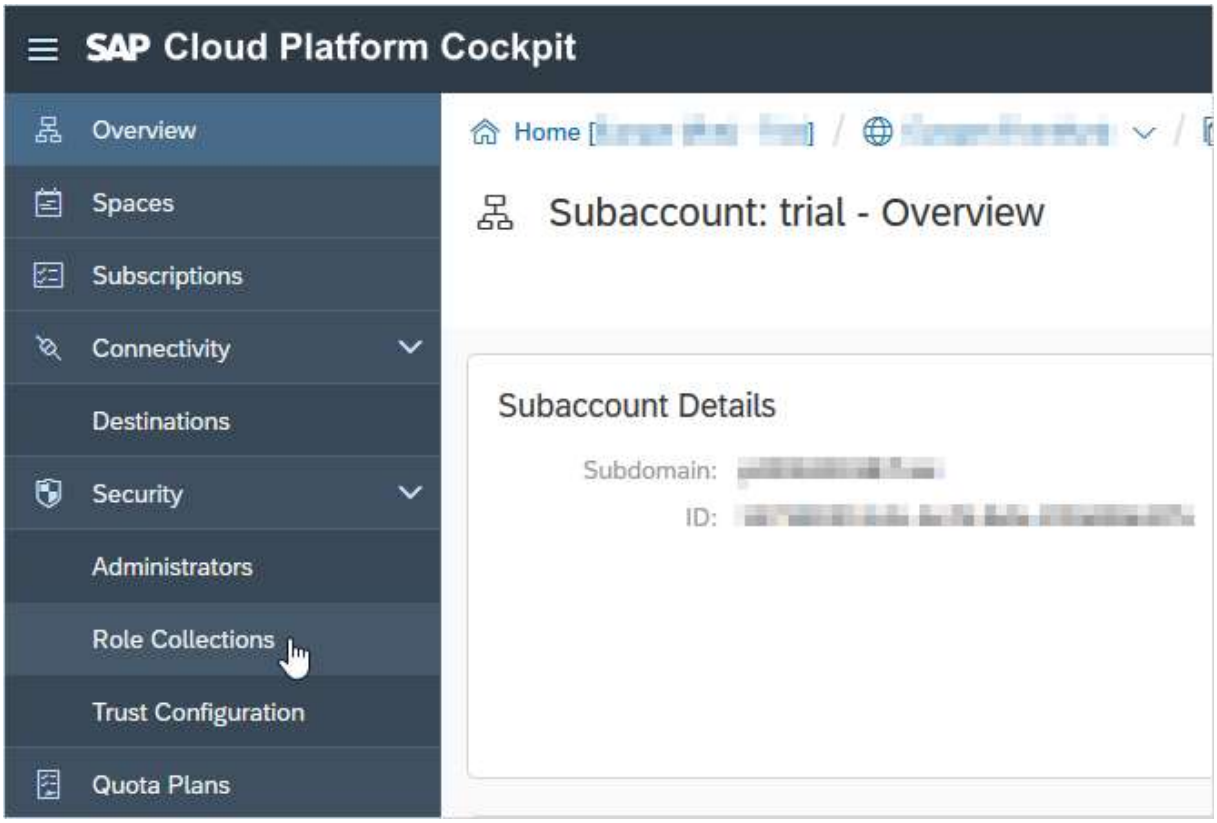
5. Click **Cloud Foundry Trial**.



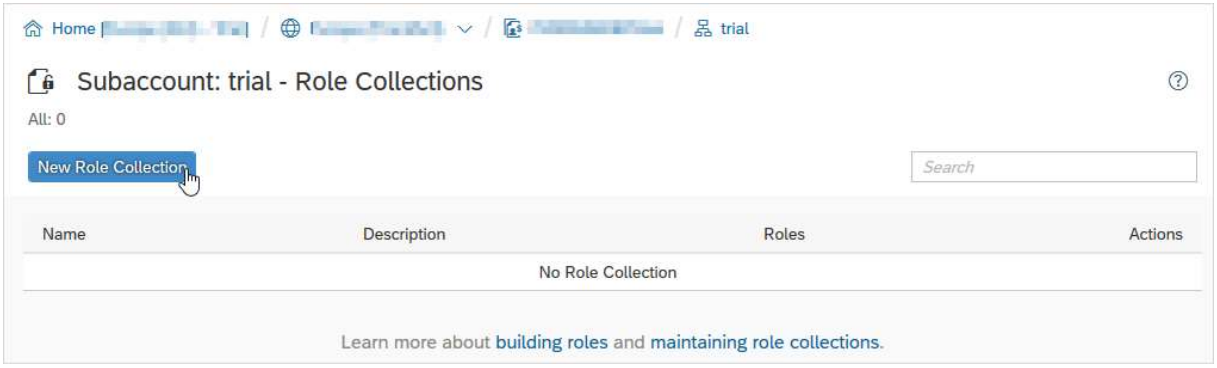
6. Click the **trial** subaccount.



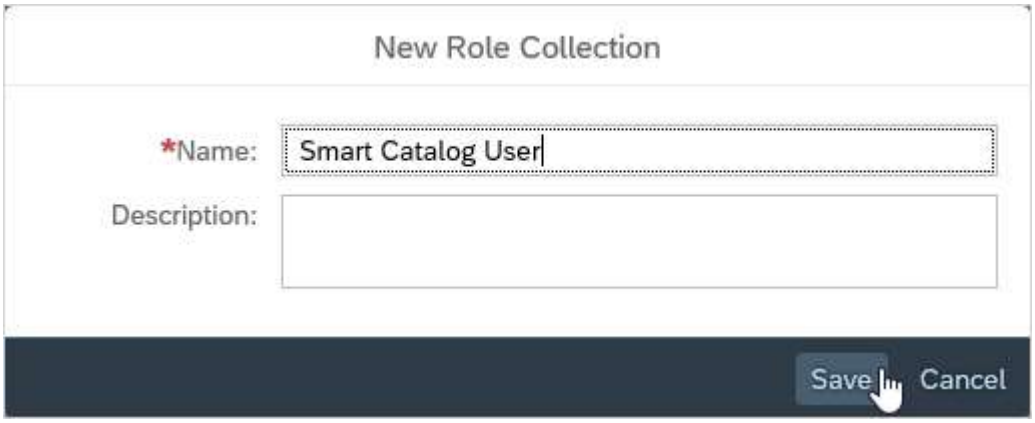
7. Click **Security > Role Collections** in the menu.



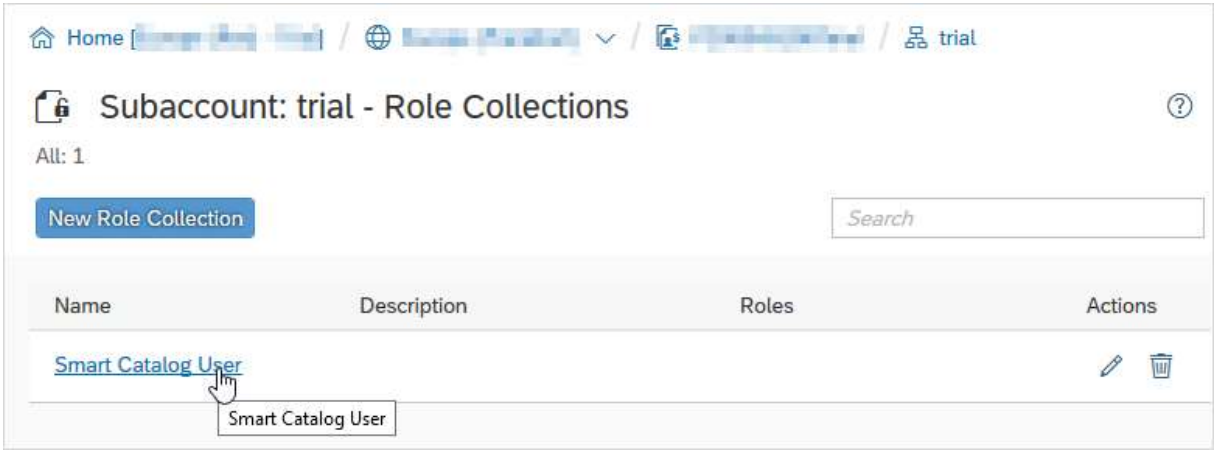
8. Click **New Role Collection.**



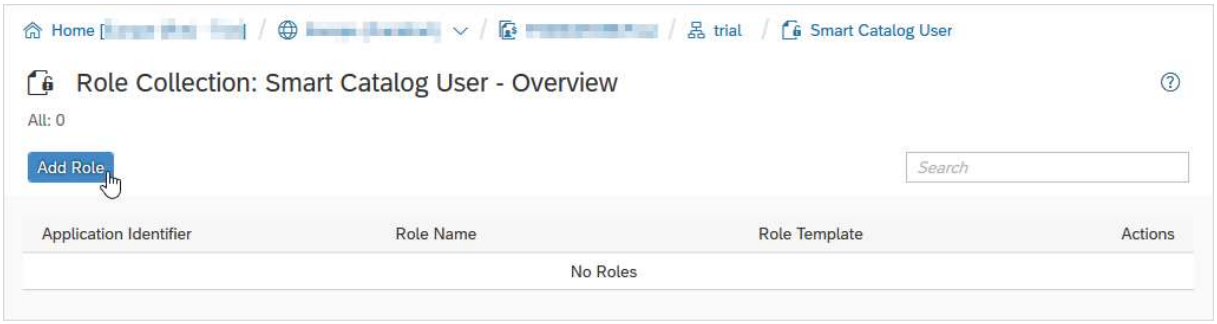
9. Enter *Smart Catalog User* as the **Name.**
10. Click **Save.**



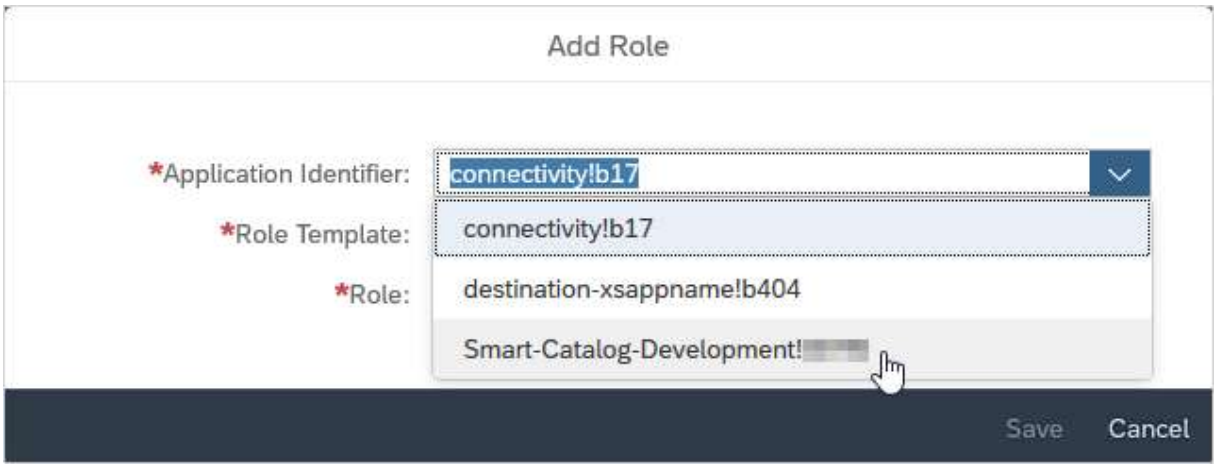
11. Click the **Smart Catalog User** role collection.



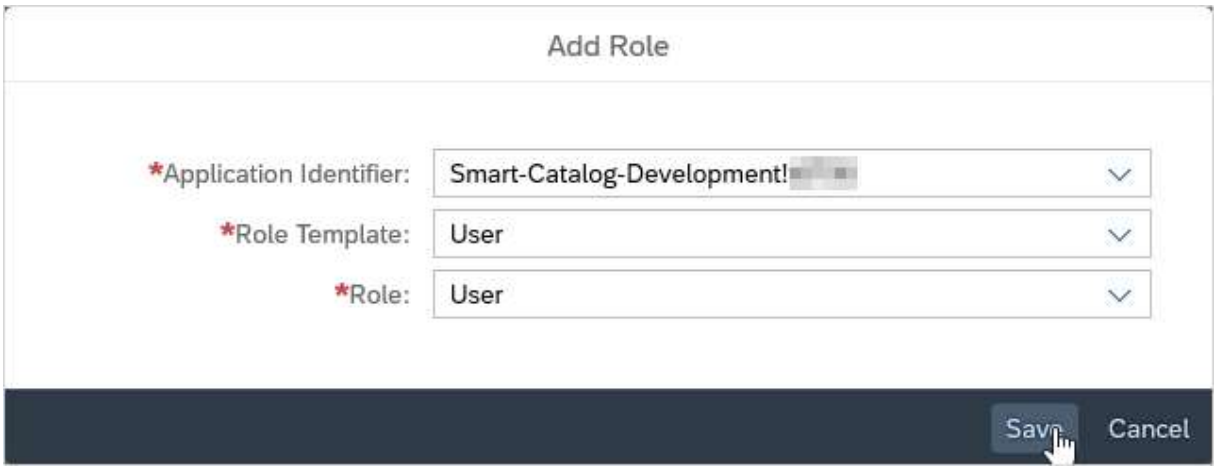
12. Click **Add Role**.



13. Select the **Application Identifier** for your app (starting *Smart-Catalog...*).

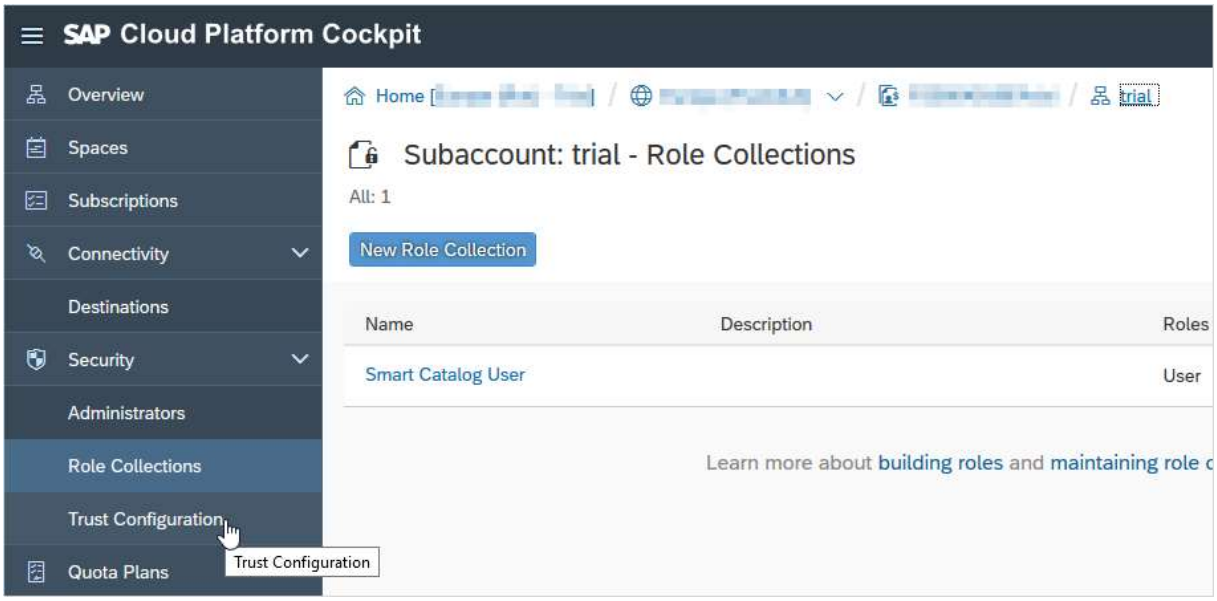


14. Select *User* for the **Role Template**.
15. Select *User* for the **Role**. (This is the *User* role as defined in your app).
16. Click **Save**.

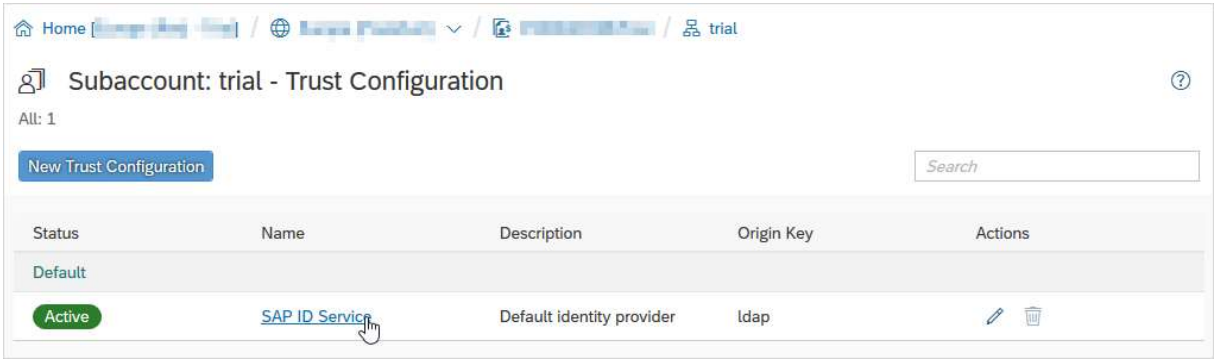


17. Click **trial** in the breadcrumb navigation to return to the trial subaccount.

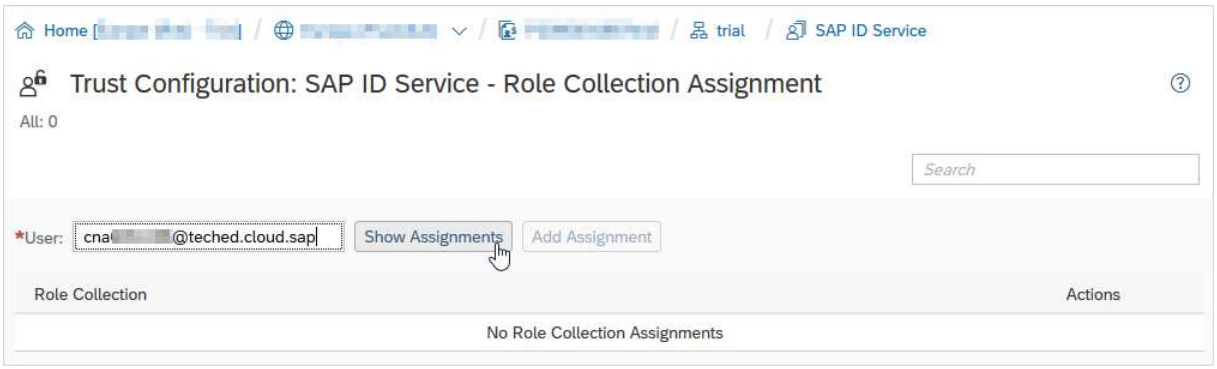
18. Click **Security > Trust Configuration** in the menu.



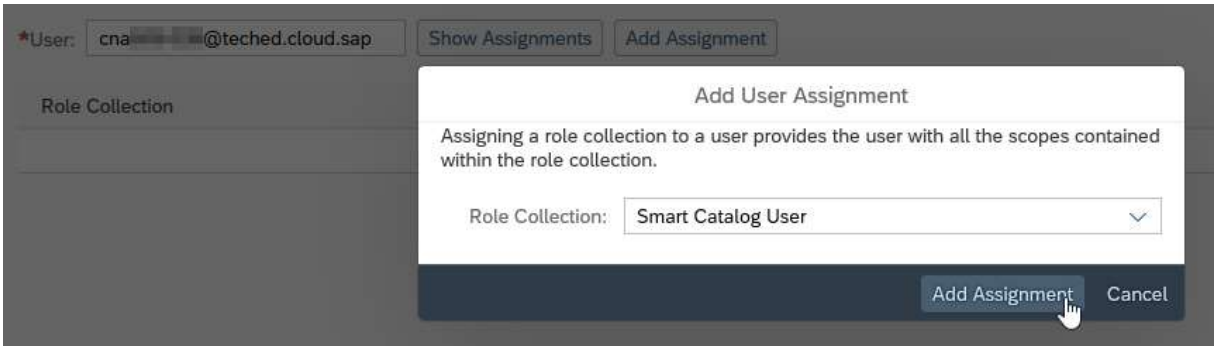
19. Click the **SAP ID Service** configuration.



20. Type your student username as the **User**
21. Click **Show Assignments**.



22. Click **Add Assignment**.
23. Check that the **Role Collection** is *Smart Catalog User*.
24. Click **Add Assignment**,

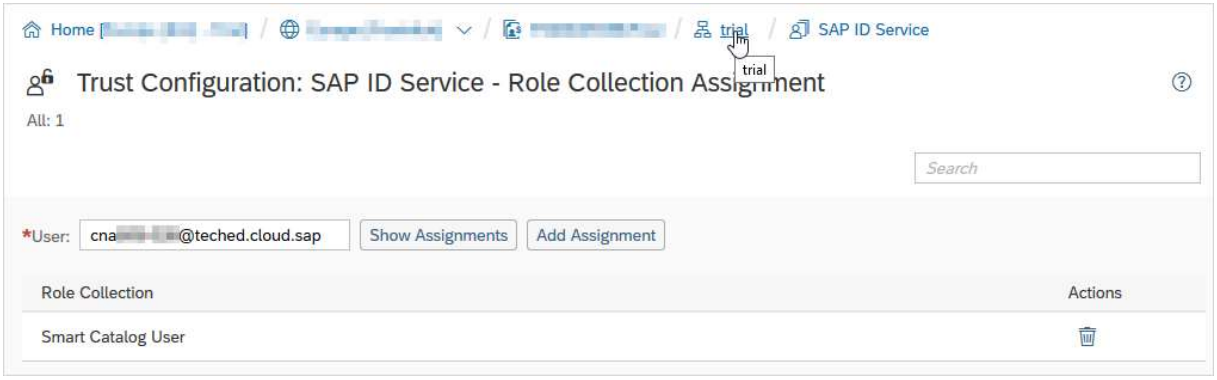


Your student username is now assigned to the User role in the app (via the *Smart Catalog User* role collection) and can be used to log in to your app.

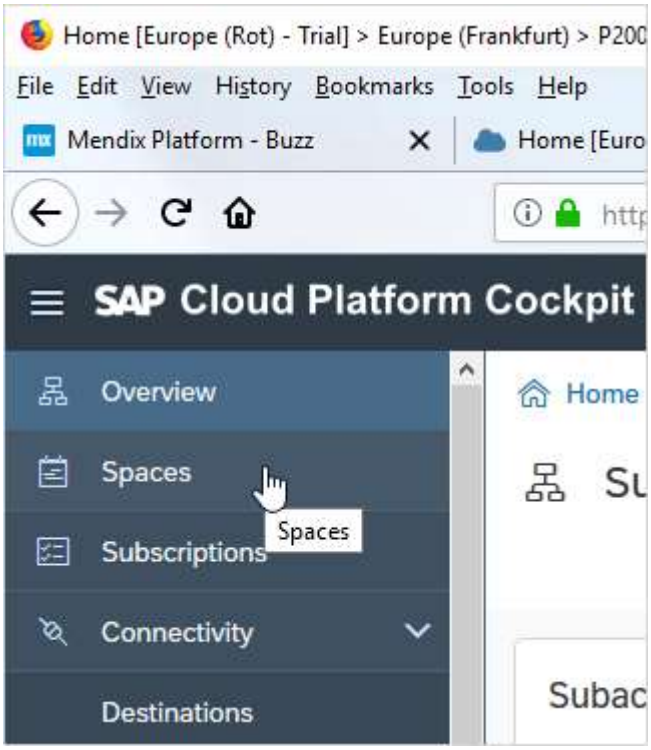
13.2.2 Cloud Connector Configuration

The end point for the cloud connector does not support HTTPS, We, therefore, have to alter the value of the variable containing the endpoint in the app when it is running in the cloud.

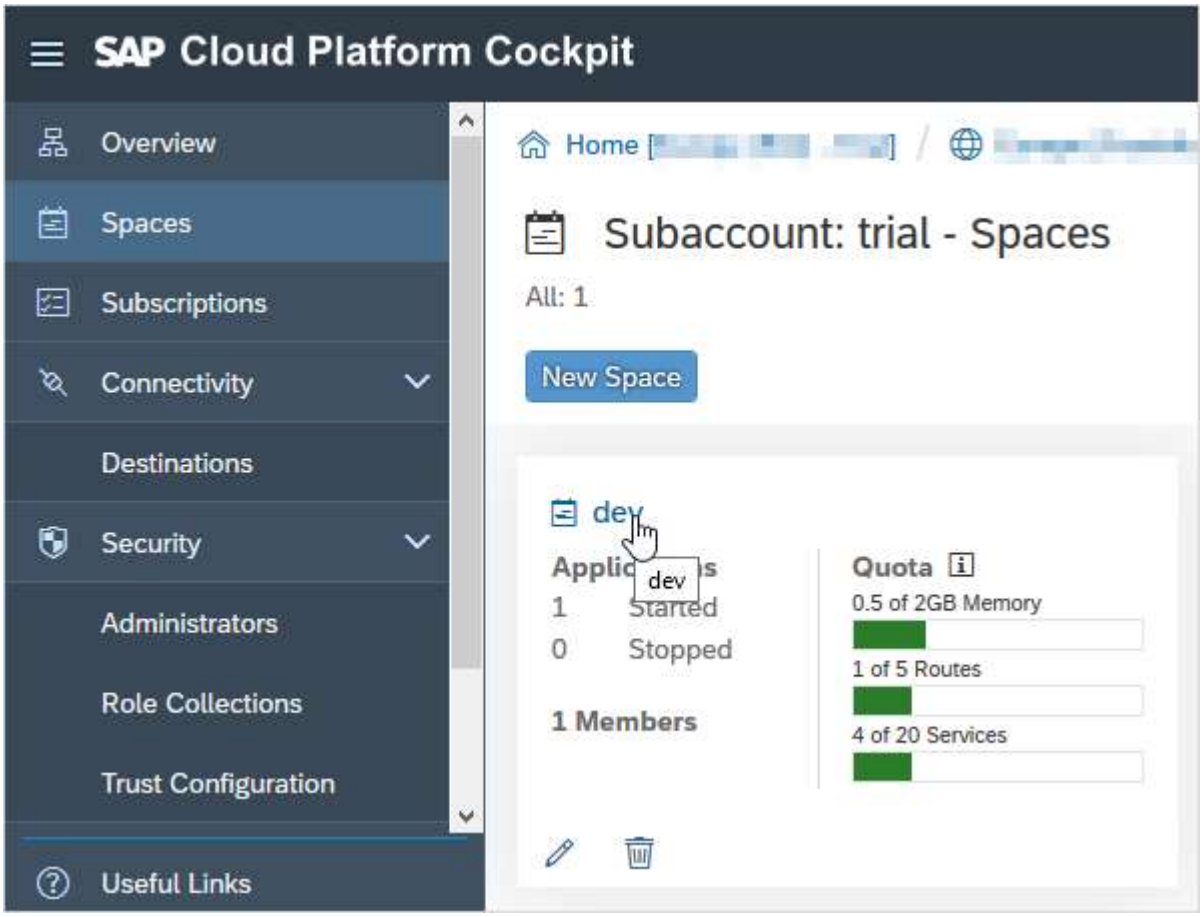
1. Click **trial** in the breadcrumb navigation to return to the trial subaccount.



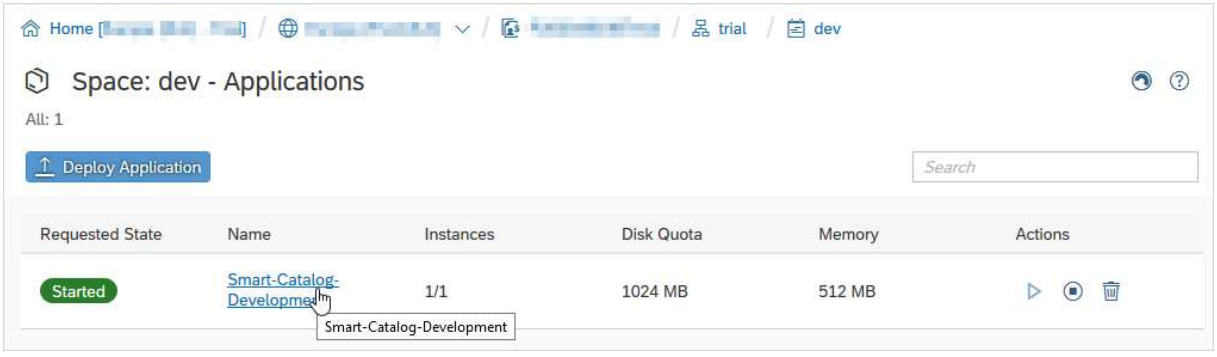
2. Click **Spaces** in the menu.



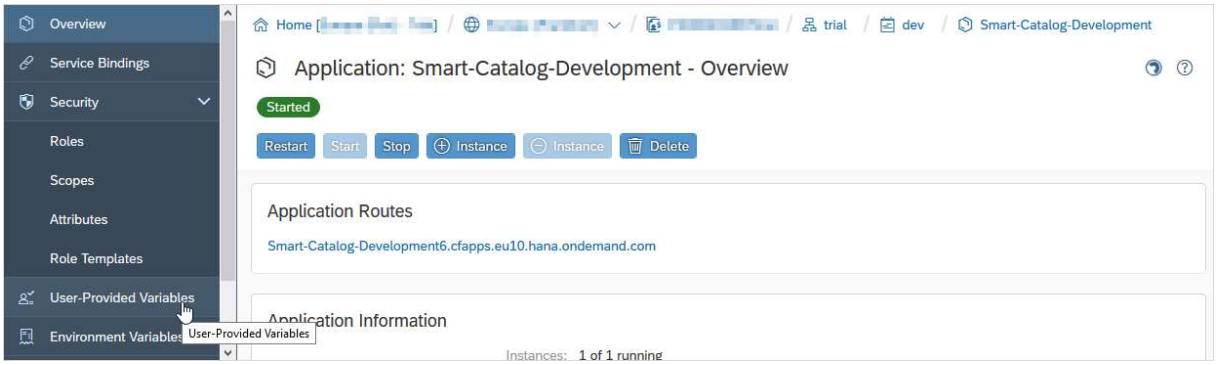
3. Click the **dev** space.



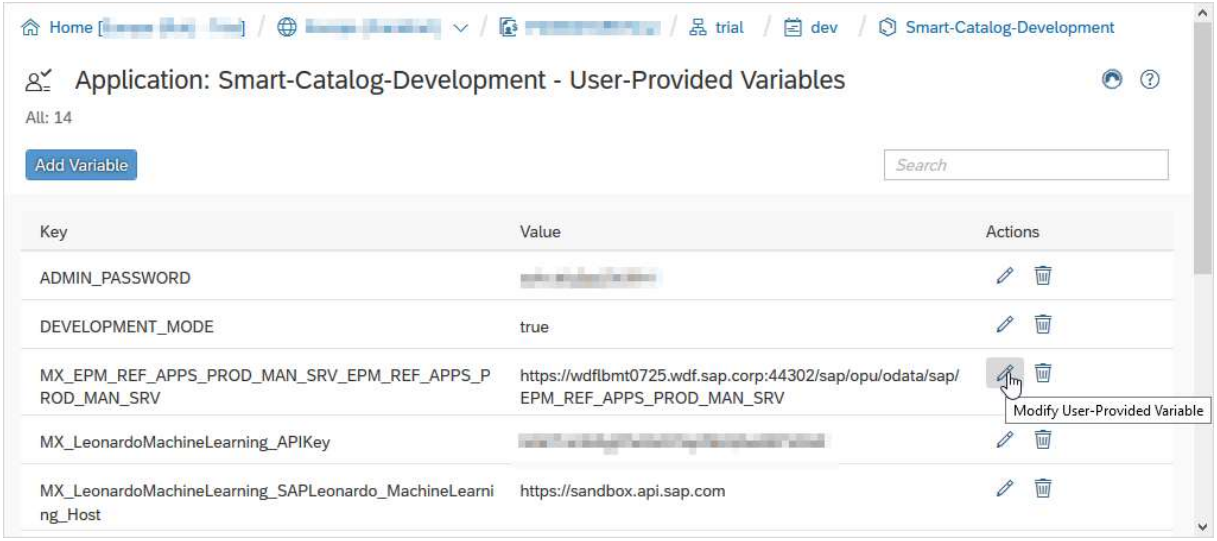
4. Click the **Smart-Catalog-Development** application.



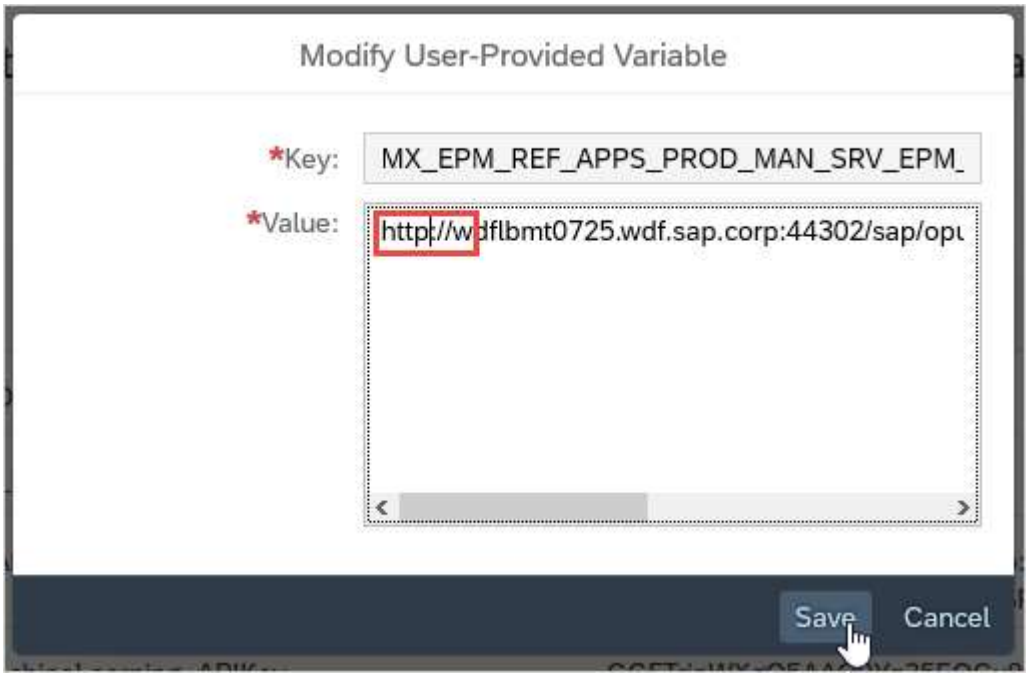
5. Click **User-Provided Variables** in the menu.



6. Click the edit pencil next to the variable **MX_EPM_REF_APPS_PROD_MAN_SRV_EPM_REF_APPS_PROD_MAN_SRV**.

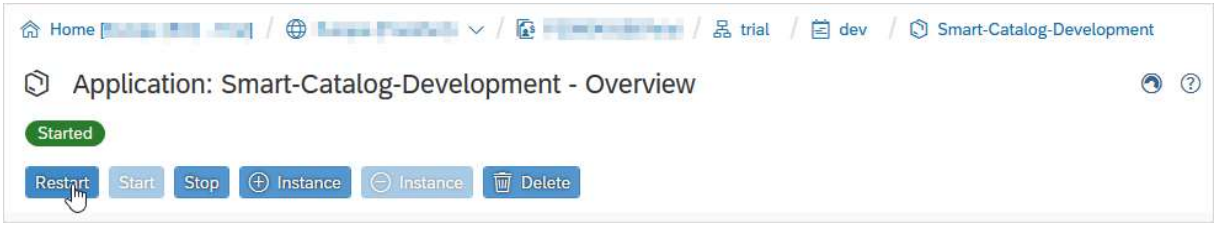


7. Change the **Value** from *https://...* to *http://...*.
8. Click **Save**.



9. Click **Overview** in the menu.

10. Click **Restart**.

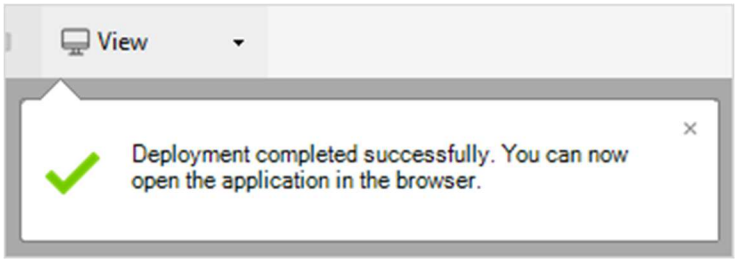


11. Wait for the application to restart and the status to return to **Started**.

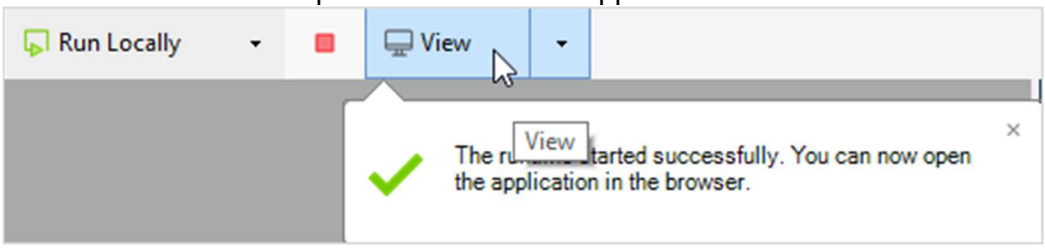
You have made all the configuration changes which are needed. These changes only have to be made the first time the app is deployed – your SAP environment will remember the settings.

13 Viewing Your App

You can now view your app running in SAP Cloud Platform and confirm that the SAP single-signon works.

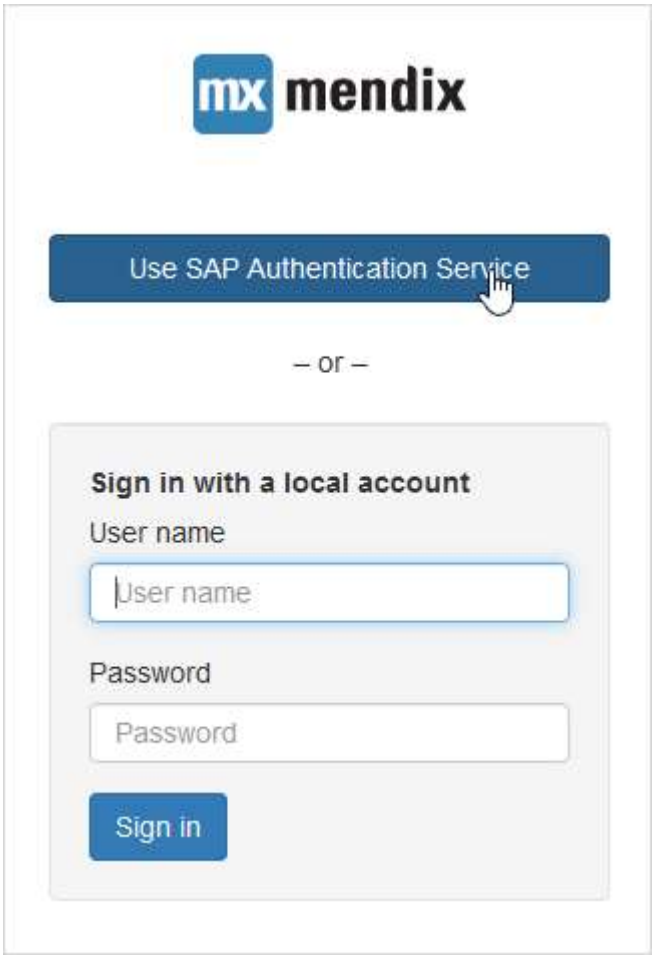


1. Click **View** in the desktop modeler when the application status is **Started**



OR
Click the URL in the SAP Cloud Platform Cockpit when the application status is **Started.**

2. Click **Use SAP Authentication Service** to login to the app.



3. Use your SAP credentials to login to the app.
4. Select an image from the student share (or the internet) and use it to search the catalog.

Congratulations! You have written a Mendix app to pull together SAP S/4HANA and SAP Leonardo Machine Learning Foundation Services.