# Create a Sales Order App using a UCC S/4 HANA System

In this project you will create an app that retrieves Market data from an S/4 HANA ERPSim system and displays a table and graphs.

### Prerequisites

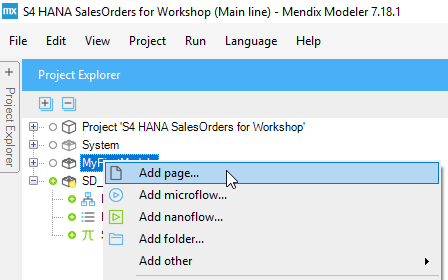
This case was prepared using version 7.21.0 of the desktop Mendix Modeler. You must also have an account in an ERPSim system on which a game has been played.

## Create the App

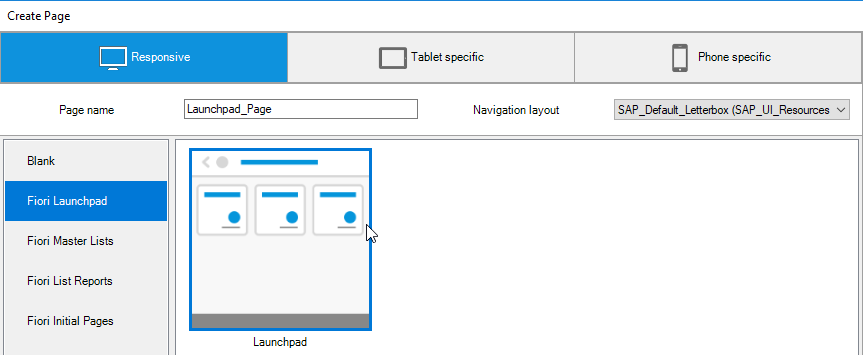
Create an app using the Fiori Blank SAP template. Name the app according to the format provided by your instructor.

## Create a Launchpad

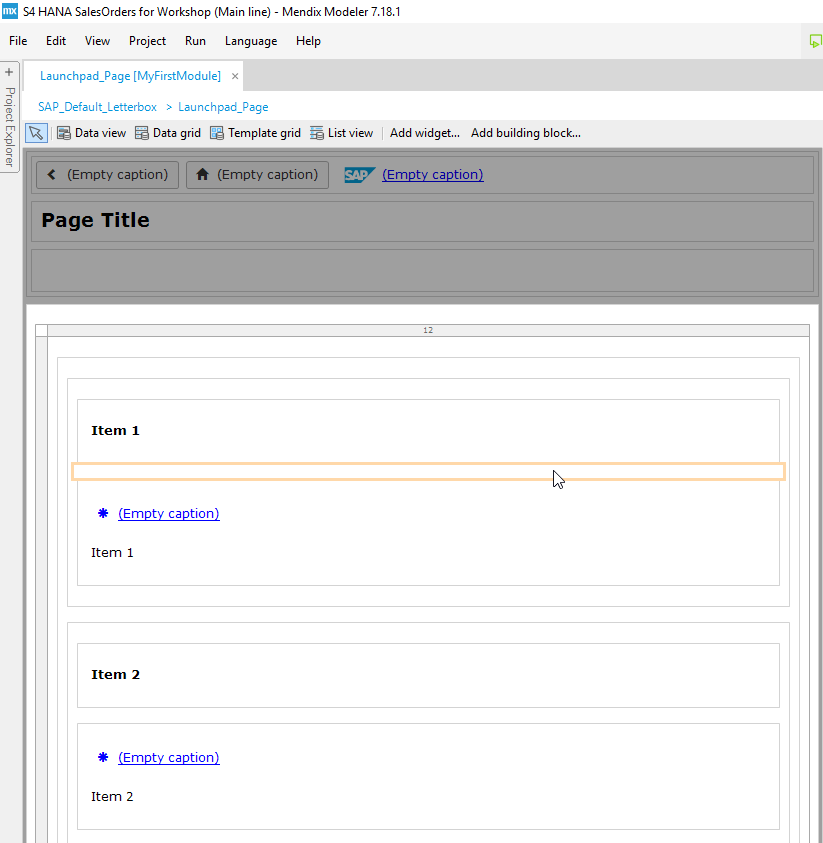
The home page that is created by the project creation wizard isn’t very useful so we’ll replace it with a Fiori launchpad. Right-click MyFirstModule in the Project Explorer and select **Add page…**



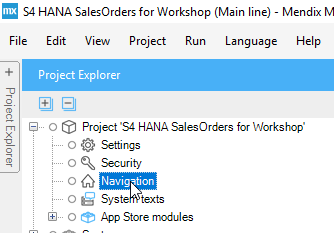
Enter the name **Launchpad\_Page**, select **SAP\_Default\_Letterbox** for the Navigation layout, select the **Fiori Launchpad** template category and select the **Launchpad** template. Create the page.



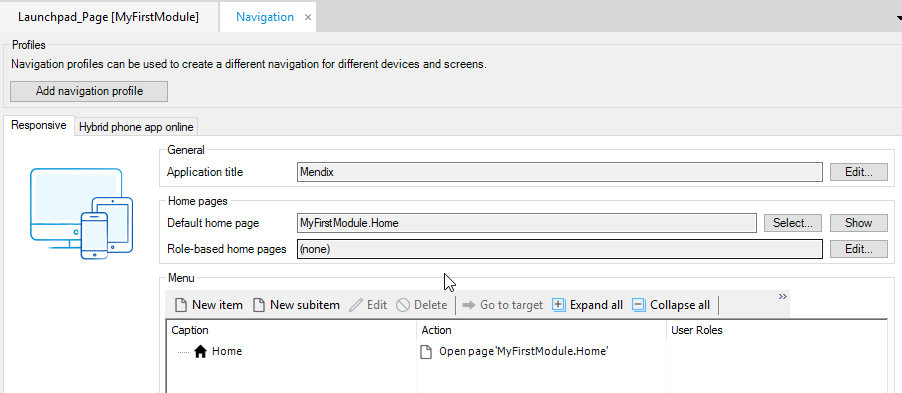
The launchpad template creates a page with three tiles.



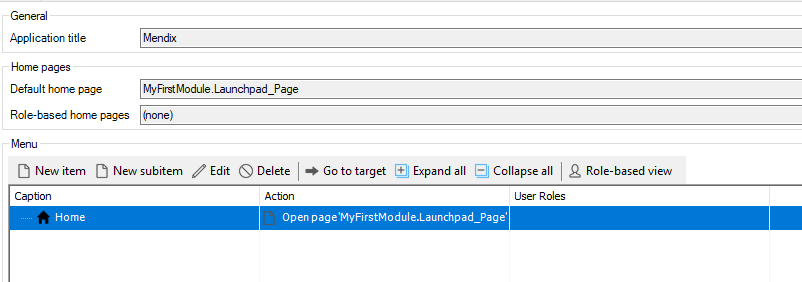
We need to make this the home page for the project. To do that, open Navigation located in the project root of the Project Explorer.



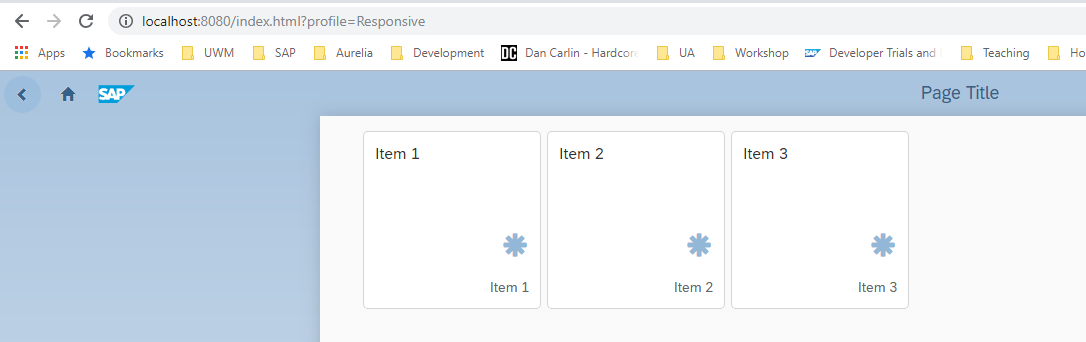
Make two changes to the navigation configuration. First, click **Select** next to the Default home page field and select the new page. Next, double-click Home in the Menu list and select the new page.



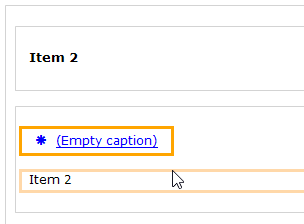
The configuration should look like this when you are done:



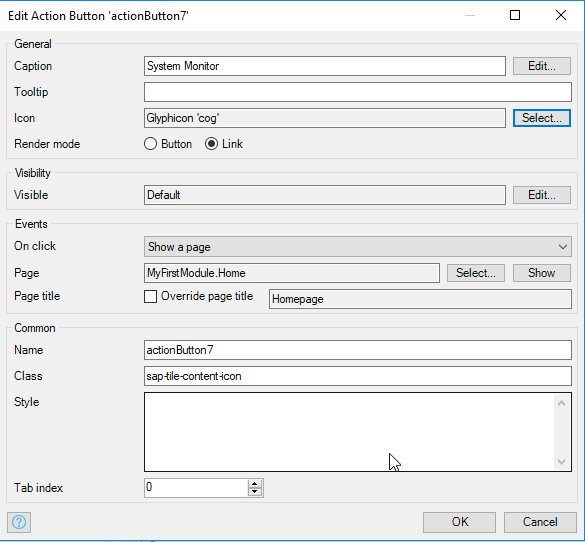
If you run the app now it should look like this:



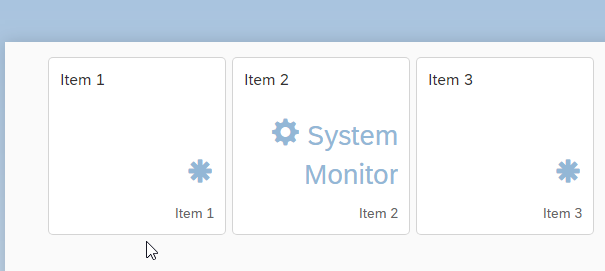
Let’s configure the second tile to open the old Home page. On the Launchpad\_Page, double-click (Empty caption) for Item 2.



Enter **System Monitor** for the Caption, change the Icon to the **cog**, select **Show a page** for the On click event and select the **Home page** for the page to show.



Run the app again to see the results. When you click the System Monitor caption, the app will navigate to the Home page. Click either the back arrow or home icon to return the Launchpad.

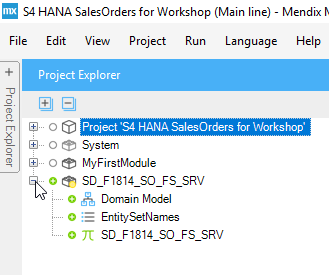


We’ll reserve the first tile for our Sales Order page.

## Import the OData Model

To access data in an SAP system using OData we must import the OData model into the Mendix Modeler as a Mendix module. To obtain the metadata, login to the Fiori launchpad with your S/4 HANA credentials then open a new tab in the browser and navigate to this URL (replace <HOST> with the host name for your server):

http://<HOST> /sap/opu/odata/sap/SD\_F1814\_SO\_FS\_SRV/$metadata/

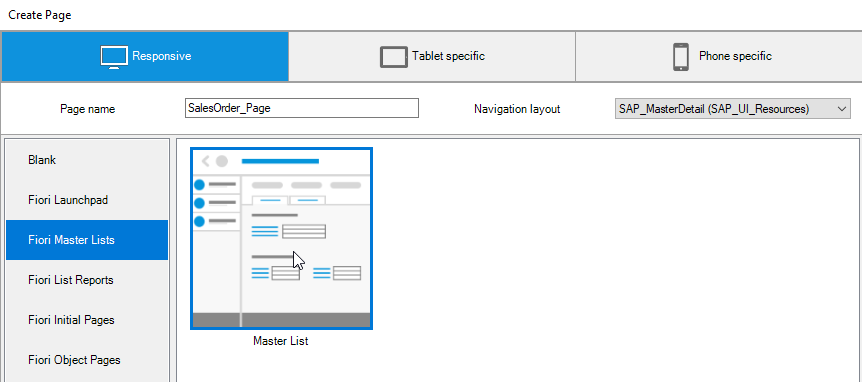


## Add a Sales Order Master/Detail page

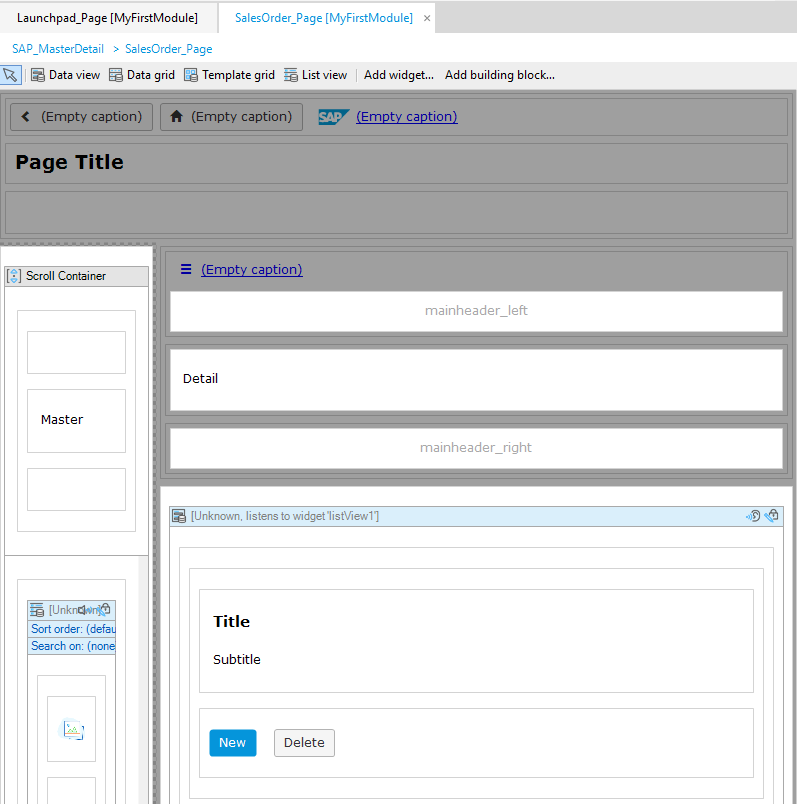
### Create the Page

Add another new page to the project. Configure it as shown below.

Page name: SalesOrder\_Page  
Navigation layout: SAP\_MasterDetail  
Template category: Fiori Master Lists  
Template: Master List



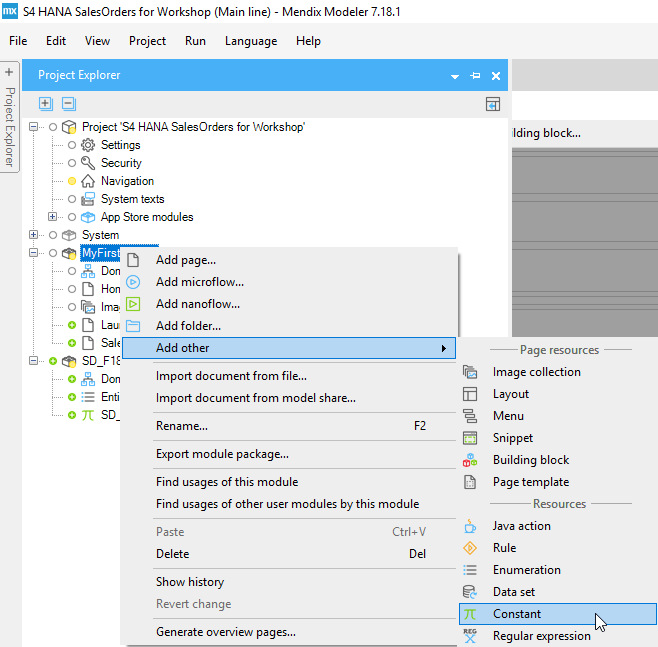
The page is created. The Scroll Container widget on the left will be used to display the master list (sales orders) and the Data View widget on the right will be used to show details of the selected sales order.



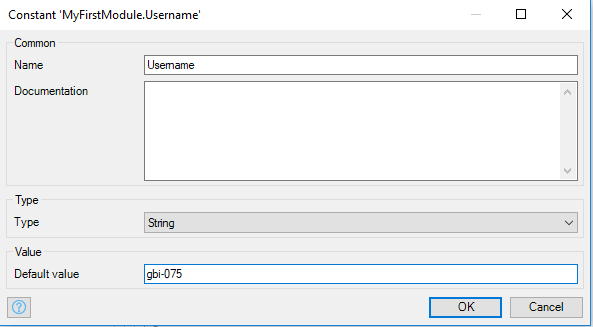
As the datasource for this page, we’ll create a microflow that retrieves sales orders from the S/4HANA system.

### Create the Sales Orders Microflow

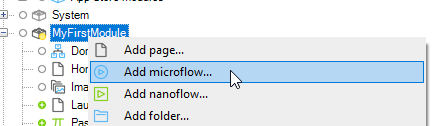
To simplify the project a bit we’ll store our credentials in constants. Right-click MyFirstModule in the Project Explorer and select Add other→Constant.



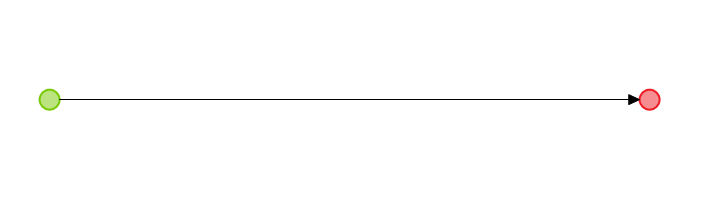
Name the constant Username and enter your S/4HANA username. Create a second constant called Password and set it equal to your S/4HANA password.



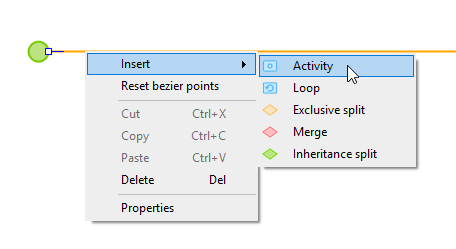
Right-click MyFirstModule and select Add microflow…



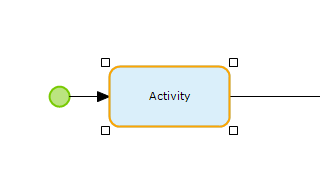
Name the microflow **DSO\_GetSalesOrders**. Initially, the microflow only has Start and End events.



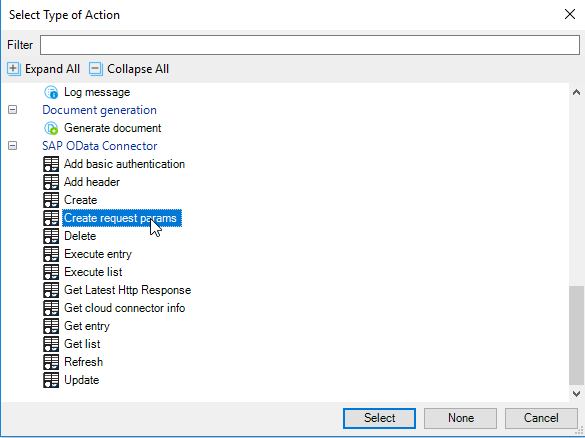
You can add activities either by dragging them from the toolbar at the top of the editor or by right-clicking the line where you want to insert the activity. Right-click the line to the right of the Start event and select Insert→Activity.



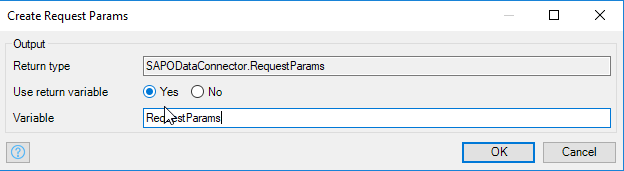
To configure the Activity, double-click it.



The list of activity types associated with the SAP OData Connector can be found at the bottom of the list of available activities. The SAP OData Connector is a Mendix add on that was added automatically to the project because we used a SAP project template to create the project. Select the Create request params activity type.

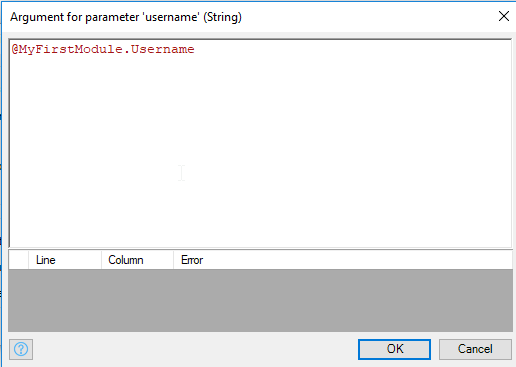


Change the variable name to RequestParams.

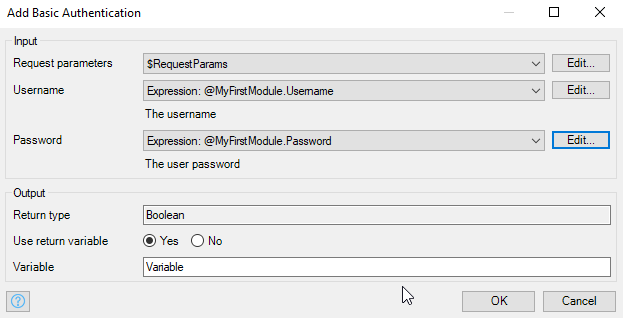


Add another activity to the right of the Create request params activity and select the Add basic authentication activity type.

Select RequestParams in the Request parameters drop down. For the Username, click Edit and enter the formula @MyFirstModule.Username. @MyFirstModule refers to the module visible in the Project Explorer. The .Username references the constant we created earlier.



Configure the Password field as well. You can leave the rest of the fields as they are.



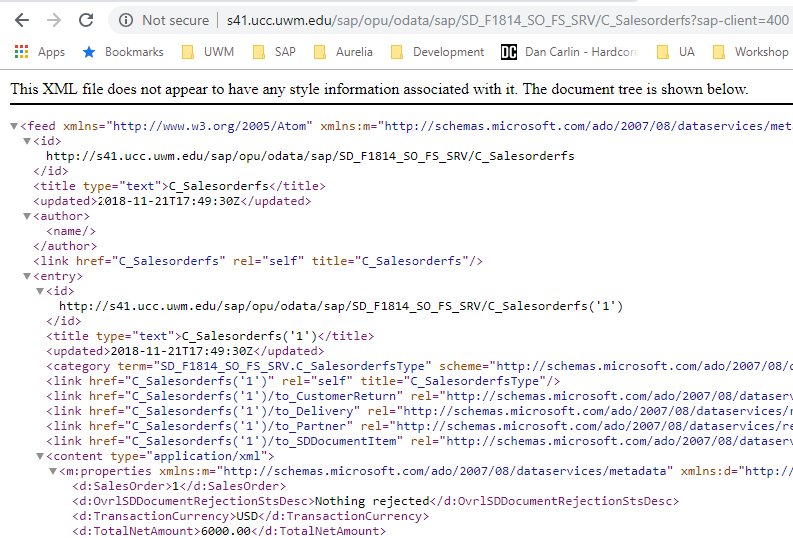
Next, add a Get List activity. Enter the following for the Query. Replace 400 with your client in the SAP system.

@SD\_F1814\_SO\_FS\_SRV.SD\_F1814\_SO\_FS\_SRV + '/' + toString(SD\_F1814\_SO\_FS\_SRV.EntitySetNames.C\_Salesorderfs) + '?sap-client=400'

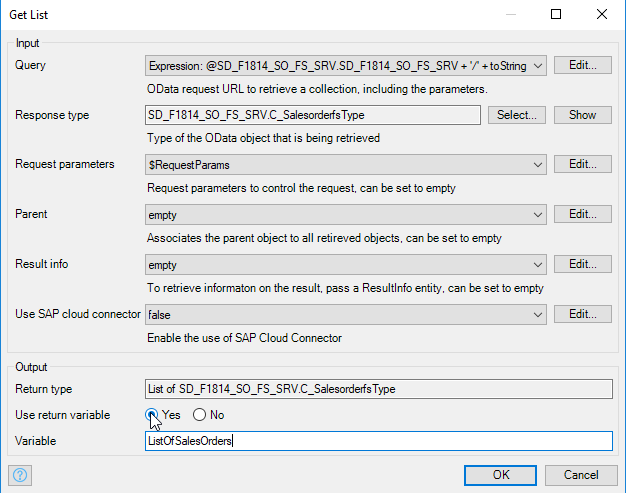
This query references the SD\_F1814\_SO\_FS\_SRV constant in the OData module we imported above which is set to the base URI of the service. SD\_F1814\_SO\_FS\_SRV.EntitySetNames.C\_Salesorderfs references the C\_Salesorderfs entity in the EntitySetNames enumeration and toString() converts it to a string. The resulting URI is:

http://<HOST> /sap/opu/odata/sap/SD\_F1814\_SO\_FS\_SRV/C\_Salesorderfs?sap-client=400

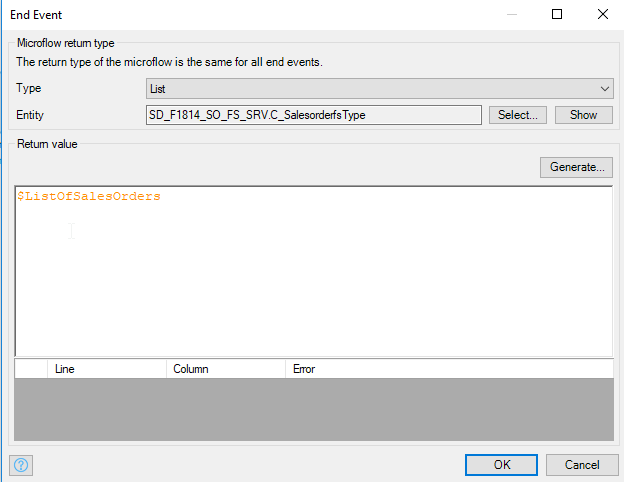
If you paste this URI into a browser, you’ll retrieve a list of sales orders (make sure you are logged into the Fiori launchpad in another tab).



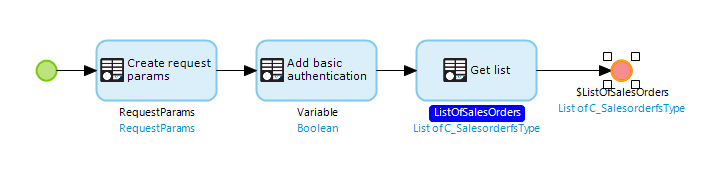
Now complete the configuration of the Get List activity as shown below.



Finally, double-click the End event to configure it to pass the ListOfSalesOrders list to the Sales Order page. Select List for the Type, the C\_SalesorerfsType entity for the Entity and $ListOfSalesOrders for the Return value. When you type the $, you will be provided a list of available variables.

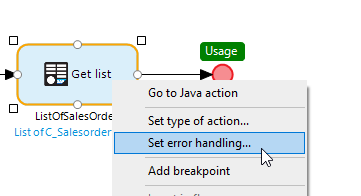


The Microflow looks like this and this is sufficient to retrieve the data and pass it to the SalesOrder page.

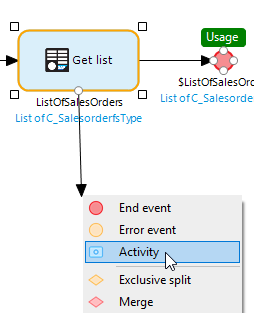


### Add Error Handling

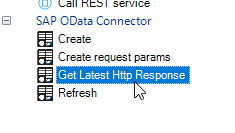
However, just in case there are problems retrieving the data, we’ll add error handling to the Get list activity. Right-click the Get list activity and select Set error handling…



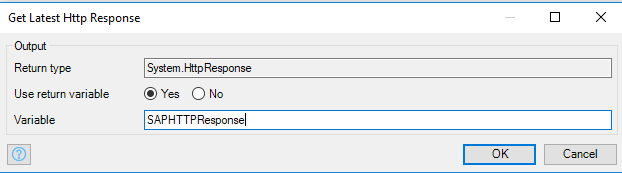
Set the error handling to **Custom with rollback**. Click on the border on the bottom of the Get list activity and drag down and release. Select Activity.



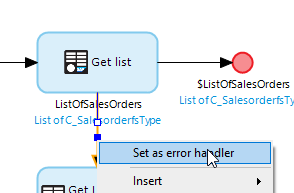
Configure the type as Get Latest Http Response…



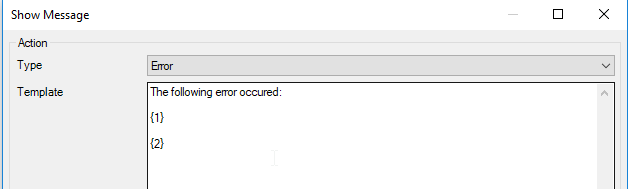
… and configure it as shown.



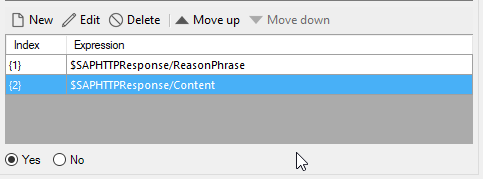
Set this Activity as the error handler for the Get list Activity by right-clicking the arrow between the two Activities and selecting Set as error handler.



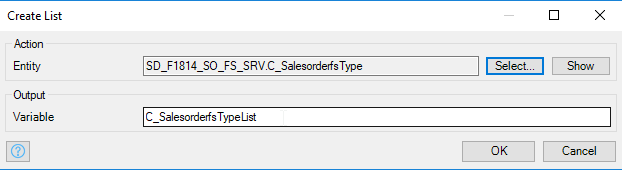
Drag from the border of the Activity you just created to add a new Activity. Configure it as a Show Message type. Enter the following for the template.



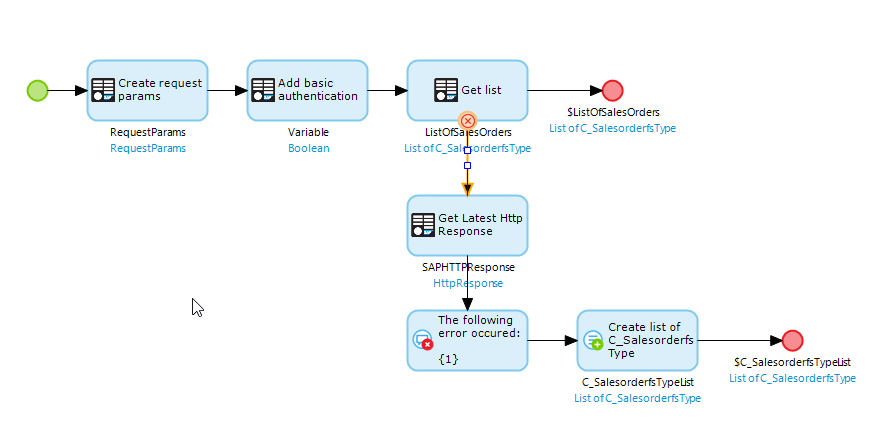
The {} brackets indicate parameters where data will be inserted. Configured two parameters as shown.



Now we have to create a new End event for this branch of the Microflow but, since the existing End event passes a list of type C\_SalesderfsType, we have to create an empty list of this type so the new End event has a list of the same type to pass. Drag from the border of the Show Message Activity and create a new Activity of type Create List. Configure it as shown:

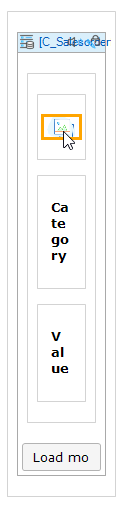


Finally, create a new End event that passes the newly created list. The final Microflow looks something like this:

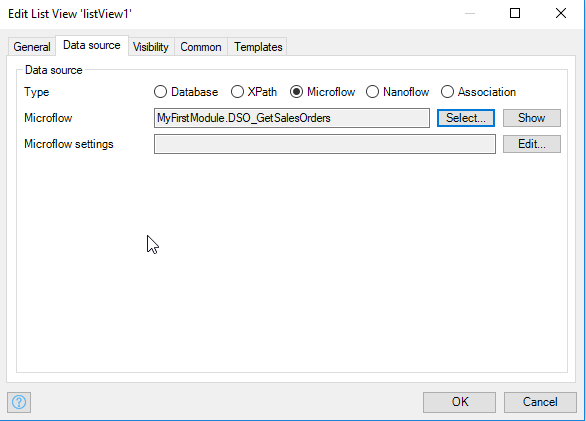


### Configure the Datasource for the SalesOrder Listview

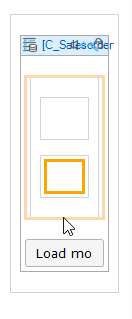
On the left side of the SalesOrders page is a ListView widget that will display the list of sales orders retrieved by the Microflow we just created. To set the Datasource of this Listview, double-click the header (not the Image widget that cursor is hovering over in the image below).



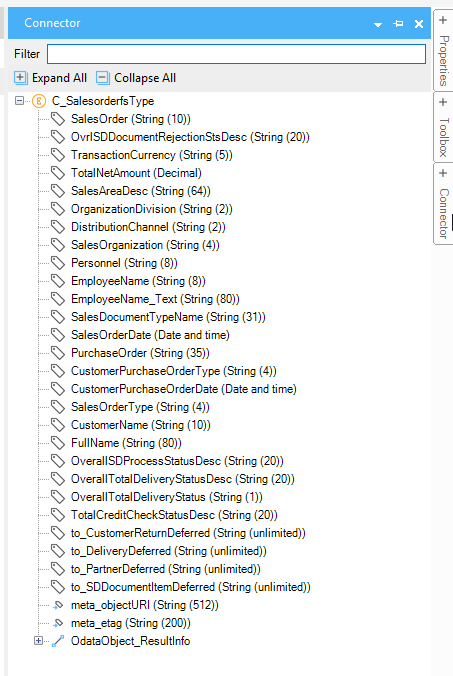
On the Data source tab, select Microflow and use the Select… button to select the DSO\_GetSalesOrders microflow.



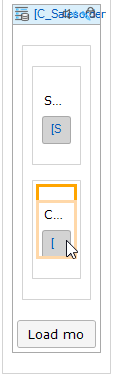
Now delete some of the widgets (the image widget and its container, the Category label and the Value label) in the Listview widget so that it looks like the image below.



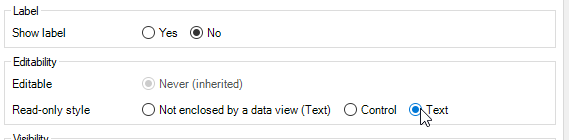
If you click inside the Listview widget and open the Connector tab (usually on the right side of the Modeler) you will see all the fields available in the C\_SalesderfsType entity.



Drag SalesOrder and CustomerName into the Listview in the locations shown below.

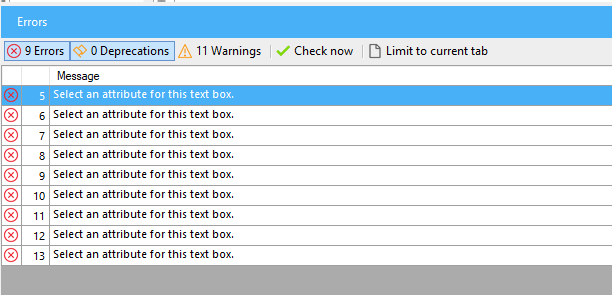


Double-click these newly added widgets and configure them to not show the label and to display as Text.



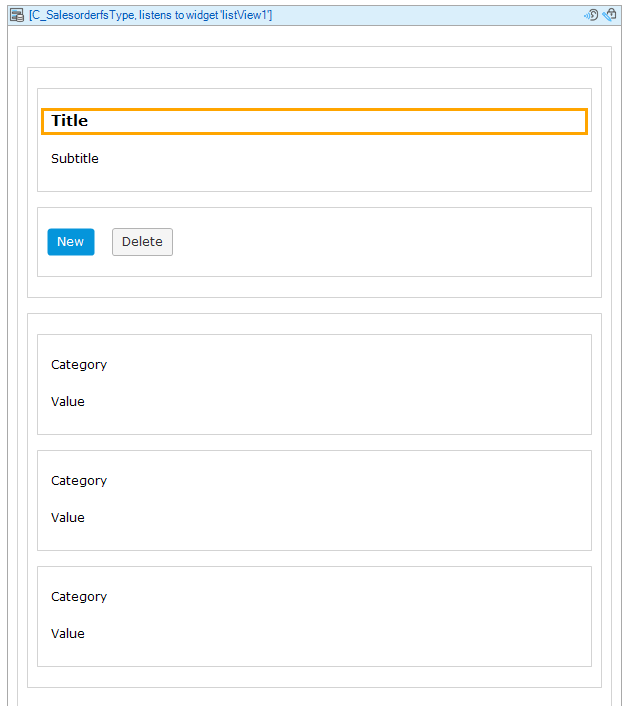
### Configure the Details

In the Error tab at the bottom of the Modeler you will notice there are now nine errors. These errors occur because we have configured a data source for the Listview but haven’t configured the data sources for the widgets on the detail side of the page.

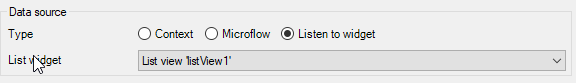


If you double-click an error message, the Modeler will attempt to highlight the location of the error in the project.

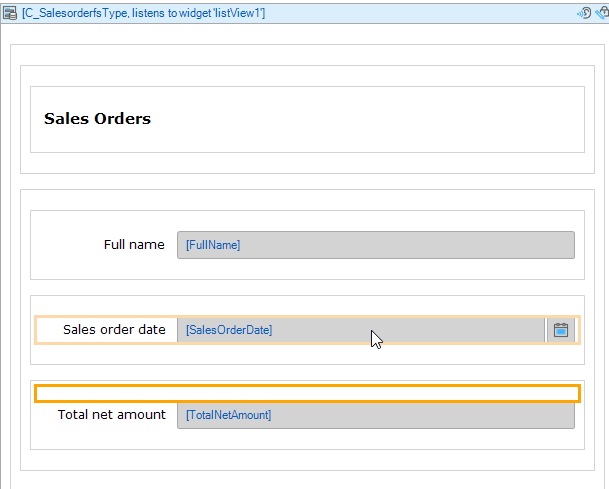
The Detail side of the page is divided into three sections. The top section looks like the image below.



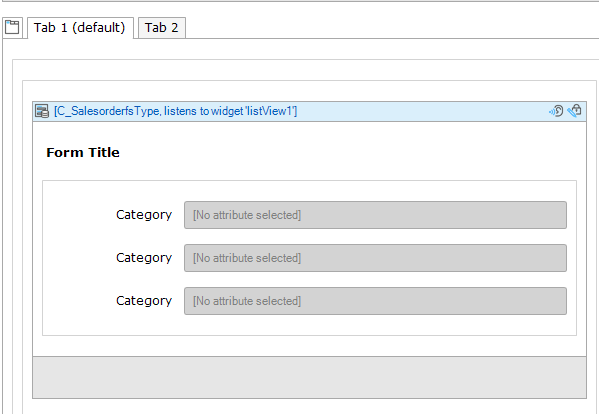
The containing widget is a Data view. Unlike the Listview, which can be bound to a list of objects, the Data view can be bound to a single object. Double-click the header of the Data view and look at how the Data source is configured. The Data view “listens” to the Listview and will be bound to whichever sales order object is selected.



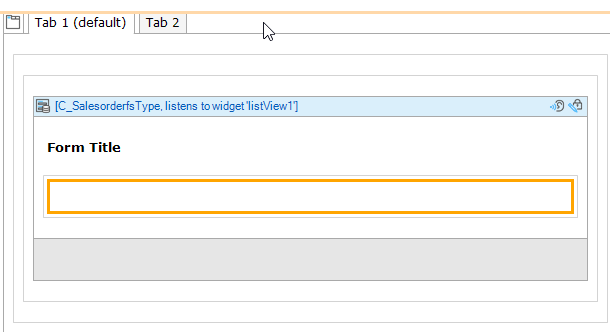
Configure the Data view as shown below. If you click inside the Data view, you can drag fields from the Connector tab onto the widget.



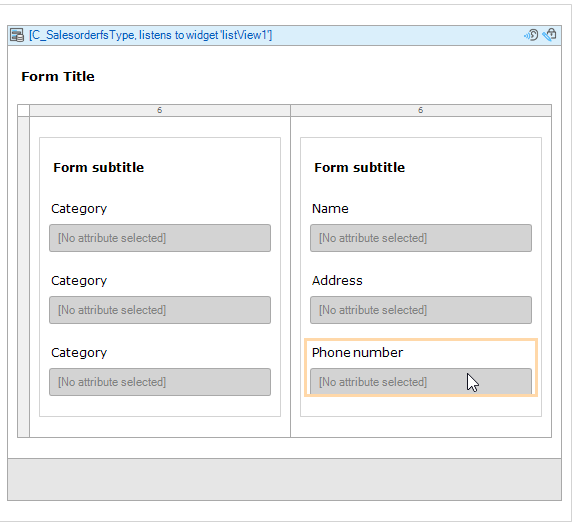
The second section has a Tab widget which we will use later. For now, just delete the three widgets shown on Tab 1.



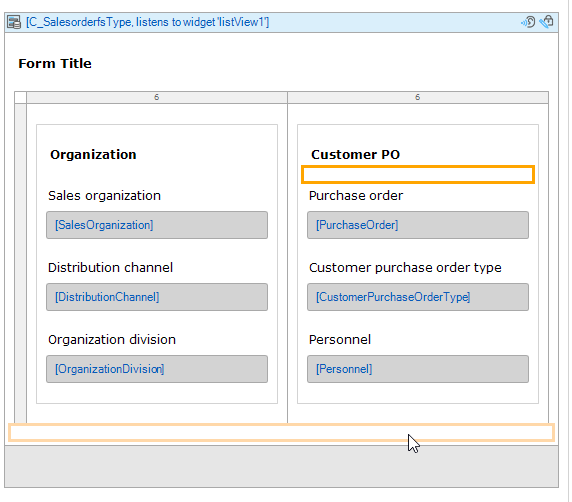
It should look like this:



Finally, the bottom section looks like this.

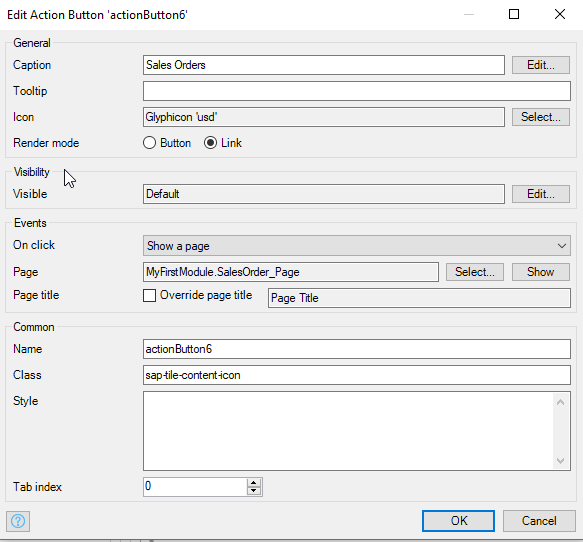


Configure it so it looks like the image below. You can drag fields from the Connector tab and drop them on the existing widgets to replace them.

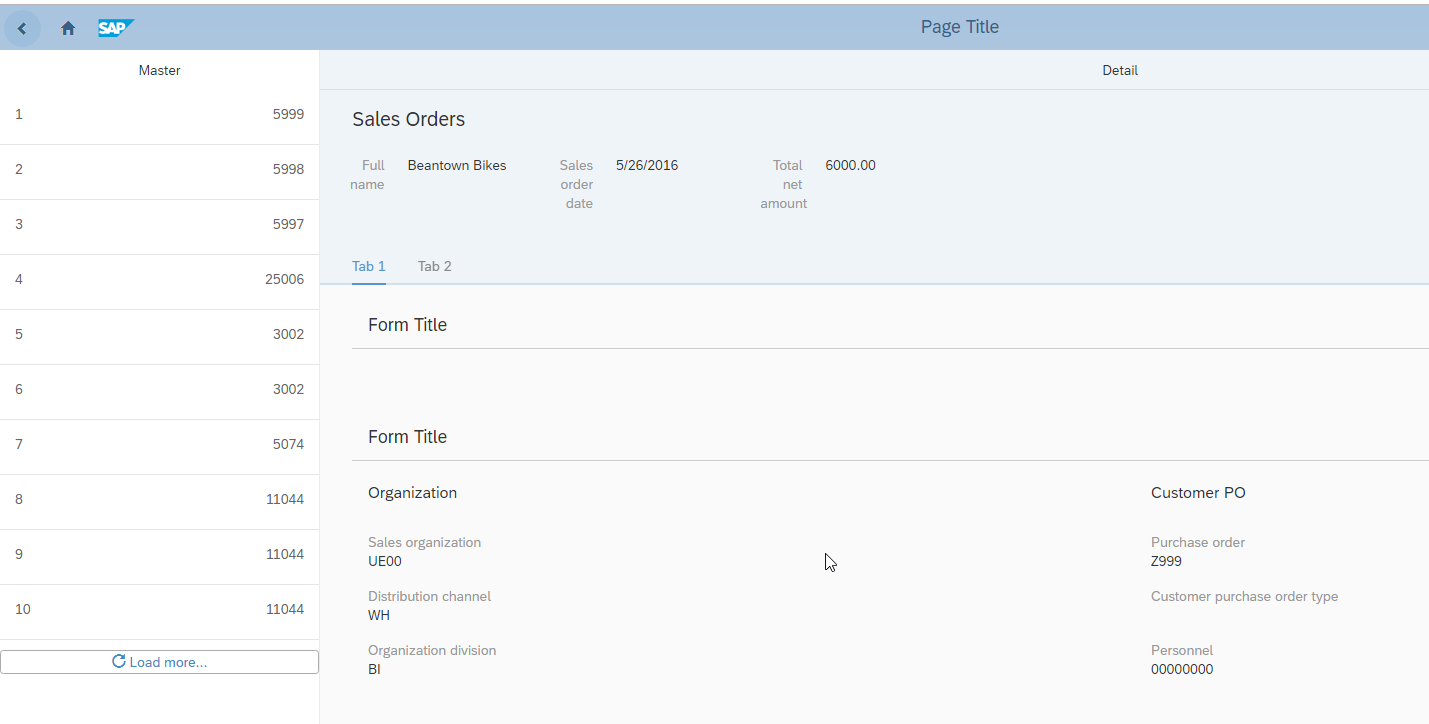


### Add the SalesOrder Page to the Launchpad

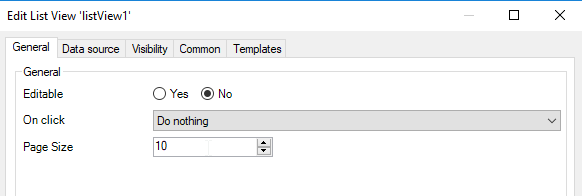
Configure the first tile on the Launchpad\_Page to open the SalesOrders\_Page.



Run the app and click on the Sales Order tile. If all goes well, the page will look like the image below showing the sales orders available in the S/4HANA system in your client.



If you create a sales order in the system (through the S/4HANA’s Fiori Launch pad or SAP GUI) you should see it on the list. You may have to configure the list to show more sales orders as we haven’t configured the Load more… button at the bottom of the Listview. Open the properties of the Listview widget and set the Page Size.

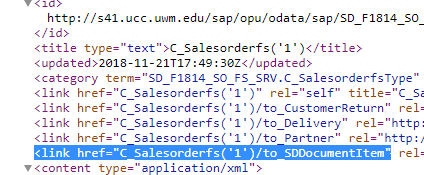


## Add the Sales Order Items

Next, we’ll add the sales order items.

### Update the Microflow

If you look at the data returned by the service we used to retrieve sales orders in a browser tab , you will see the highlighted service in the image below. This service returns the items for a sales order (sales order 1 in this case).



This service would be used like this to retrieve the items for sales order 1:

http://<HOST> /sap/opu/odata/sap/SD\_F1814\_SO\_FS\_SRV/C\_Salesorderfs(‘1’)/to\_SDDocumentItem?sap-client=400

One way to retrieve sales order items when the user selects a sales order on the list would be to create another Microflow. However, it is a bit simpler to retrieve the sales order items at the same time we retrieve the sales orders. The sales order items are embedded in the XML document returned by the system. This increases the amount of data retrieved but will work for our purposes since we don’t have a large number of orders. The URI to retrieve the sales orders along with their items is:

http://<HOST> /sap/opu/odata/sap/SD\_F1814\_SO\_FS\_SRV/C\_Salesorderfs?$expand=to\_SDDocumentItem&sap-client=400

The $expand keyword tells the service to retrieve the sales order items and embed them in the data returned. Try it in a browser to see the results.

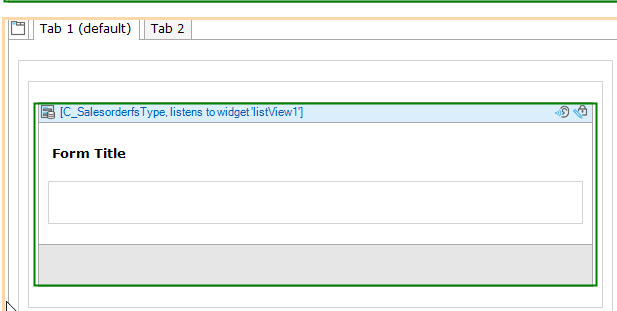
Edit the Get list Activity in the DSO\_GetSalesOrders microflow and change the Query to:

@SD\_F1814\_SO\_FS\_SRV.SD\_F1814\_SO\_FS\_SRV + '/' + toString(SD\_F1814\_SO\_FS\_SRV.EntitySetNames.C\_Salesorderfs) + '?$expand=to\_SDDocumentItem&sap-client=400'

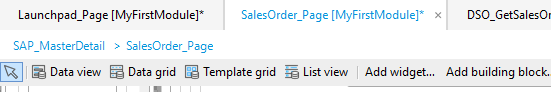
Remember to change the client number.

### Add the Sales Order Items Data Grid to the SalesOrders\_Page

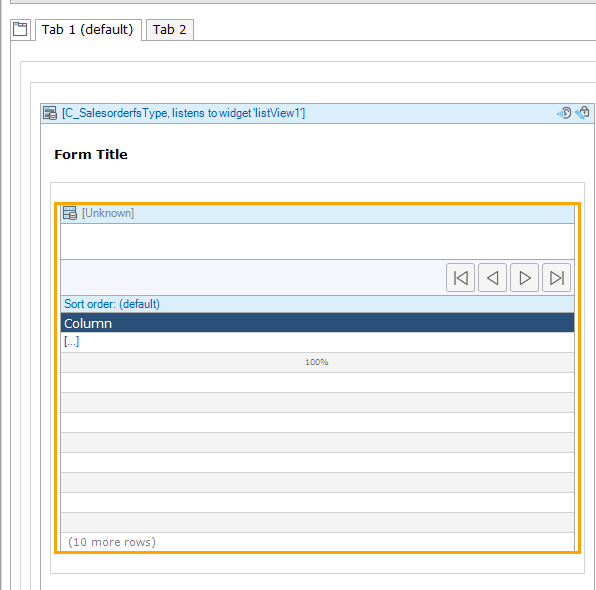
Locate the Tab widget on the SalesOrders\_Page.



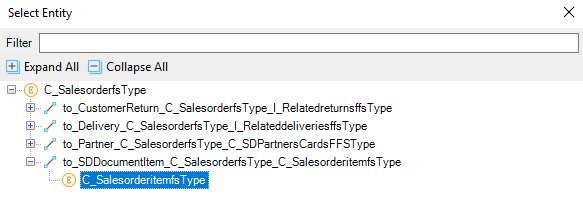
Click on Data grid in the toolbar at the top of the Modeler…



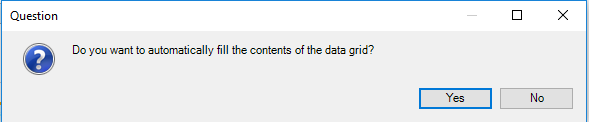
…then click inside the Container widget under Form Title.



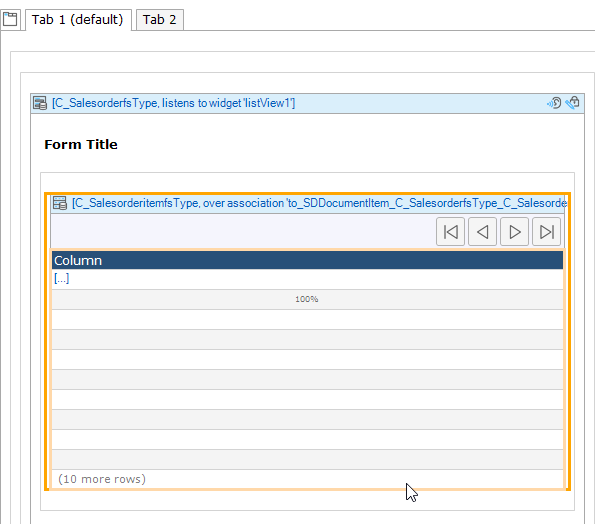
Double-click the header of the new Data grid and select Association for the Data source. Click the Select button next to the Entity field and configure the Entity as shown below. This takes advantage of the association between C\_SalesorderfsType and C\_SalesorderitemfsType that exists in the Domain Model.



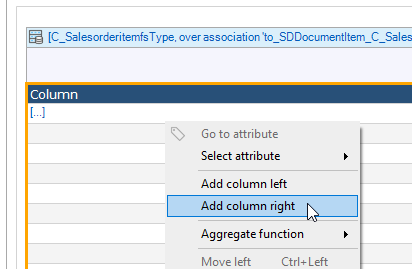
When you click OK the Modeler will ask you if you want to fill the grid automatically. Select No.



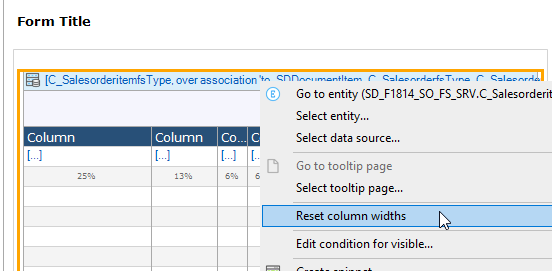
The Data grid looks like the image below.



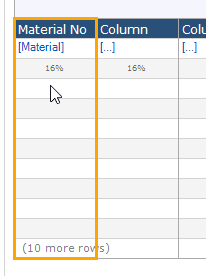
Add a column by right-clicking inside the existing column and selecting Add column right.



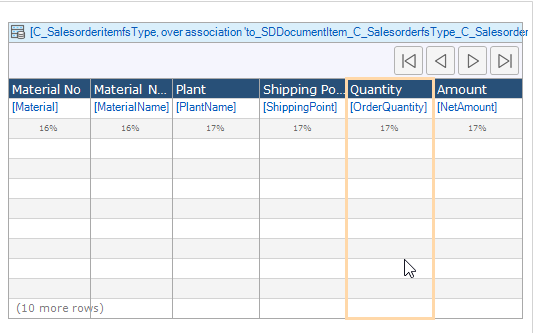
Add four more columns. Once you’ve added all the columns, you can right-click the Data grid header and select Reset column widths to equalize the widths.



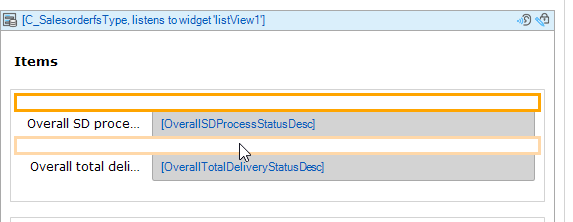
Now click the left-most column and start typing to set the header to Material No. Then drag the Material field from the Connector tab onto the column.



Configure the remaining columns with the fields Material Name, Plant, Shipping Point, OrderQuantity and NetAmount.

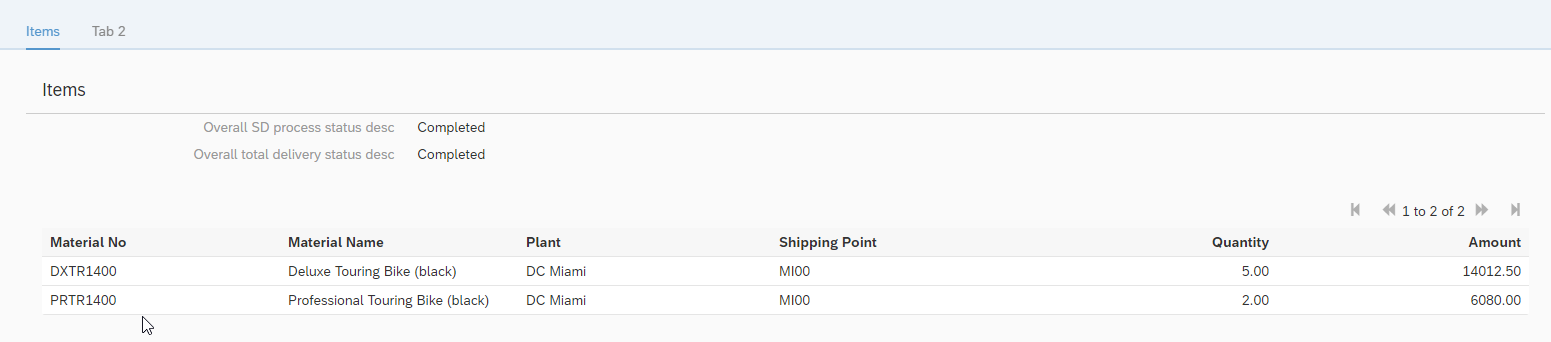


Drag a Container widget from the Toolbox tab to just above the Data grid then drag the fields OverallSDProcessStatusDesc and OverallTotalDeliveryStatusDesc into it.



You can change the Tab caption and the label as well.

Now when you run the app, the items are displayed.



## Add Customer Details

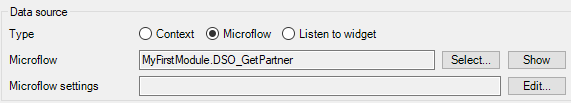
We’ll add the customer’s details to the second tab. Initially, the second tab is empty.



The first thing to add is a Data view. Click the Data view widget in the toolbar at the top…



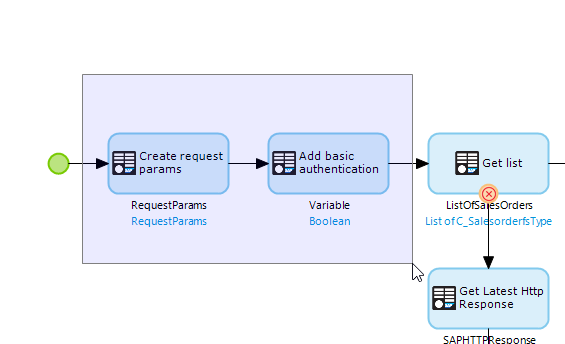
…and click on the content area for Tab 2. Double-click the header of this Data view and configure it to listen to the Listview widget. Next, add a second Data view inside the first. Double-click the header of the second Data view and select Microflow as the Data source. Click Select… to select a microflow then click New to create a new microflow. Name it DSO\_GetPartner.

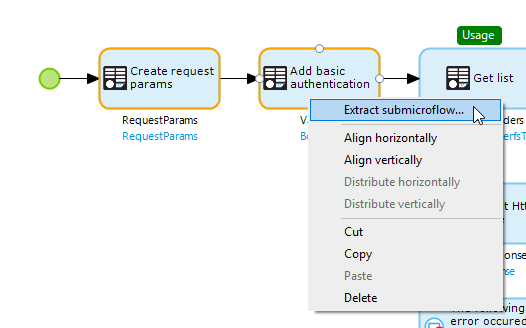


The microflow is similar to DSO\_GetSalesOrders. The reason we embedded one Data view in another is to make sure we had access to the selected sales order in the microflow. Notice the new microflow, in addition to the Start and End events has a parameter of type C\_SalesOrdersType. Mendix is smart enough to know we might want to have access to the sales order object the outer Data view is bound do inside the microflow.

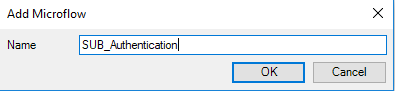


The first two Activities in this microflow are identical to the first two in DSO\_GetSalesOrders so we can extract those two Activities into a submicroflow. Open DSO\_GetSalesOrders and drag a box around the Create request params and Add basic authentication Activities.

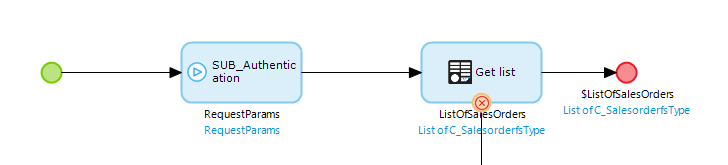


Right-click on of the selected Activities and select Extract submicroflow…

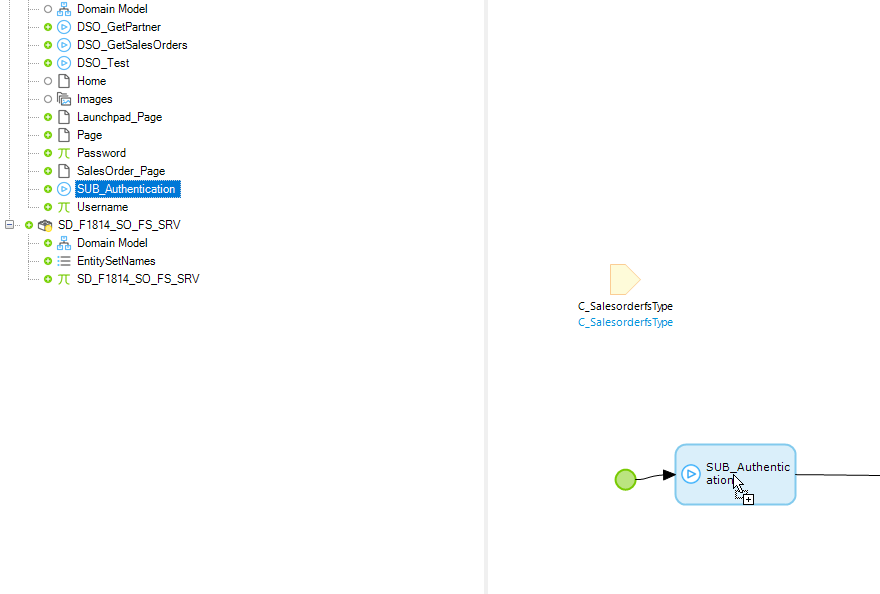
Name the submicroflow SUB\_Authentication.



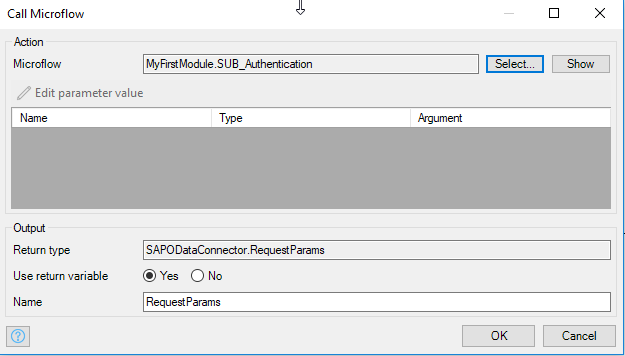
The submicroflow is created and added to the DSO\_GetSalesOrders microflow.



In the DSO\_GetPartner microflow, drag the submicroflow from the Project Explorer to the line immediately to the right of the Start event.



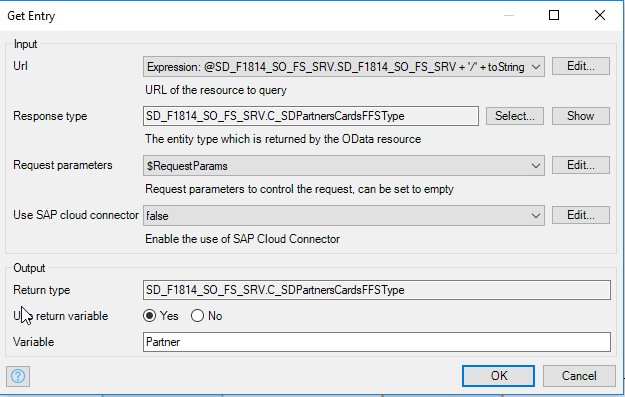
Double-click the submicroflow Activity and then click OK to create the RequestParams variable.



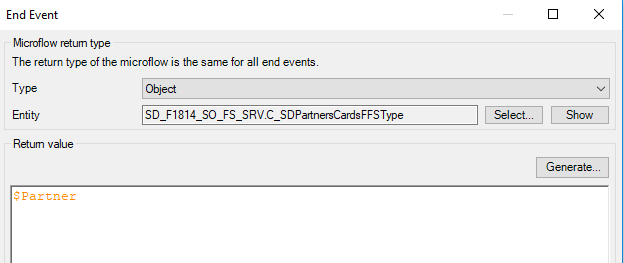
Add a Get entry Activity to the right of the submicroflow. We’re retrieving a single object in this case so we use the Get entry Activity instead of the Get list Activity. The Query should be:

@SD\_F1814\_SO\_FS\_SRV.SD\_F1814\_SO\_FS\_SRV + '/' + toString(SD\_F1814\_SO\_FS\_SRV.EntitySetNames.C\_SDPartnersCardsFFS) + '(SDDocument=%27' + $C\_SalesorderfsType/SalesOrder + '%27,SDDocumentItem=%27000000%27,PartnerFunction=%27SP%27)'

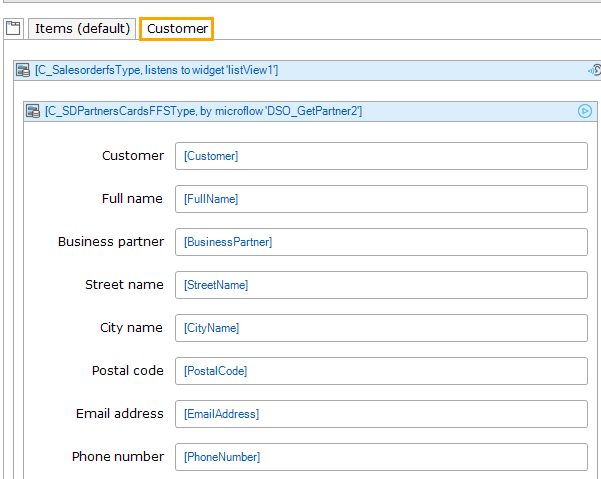
Note this URI uses the sales order number from the parameter passed into the microflow ($C\_SalesorderfsType/SalesOrder). The rest of the configuration looks like the image below.



Finally, configure the End event to pass the Partner to the Data view.



Return to the Tab 2, click inside the inner Data view and drag fields from the Connector tab onto the tab. You can also change the Tab caption.



Now run the app and select the Customer tab.

