Workshop NATO

Workshop for NATO ABAP/Fiori Elements

Setup Workstation for Workshop.

Download latest version of Eclipse IDE for Enterprise Java and Web Developers (2022-03)

Install ADT (ABAP Development Tools) for Eclipse.

Create SAP Trial Account and create a SAP BTP ABAP Environment Trial User.

Exercises

1. Create Package for Workshop

2. Create Dev Space in SAP Business Application Studio for Workshop

3. Create Simple CDS

4. Create Basic List Report

5. Build & Deploy our Application and Configure our Fiori Cloud Launchpad

6. Add Search and Filters to List Report

7. Create List Report Object Page

8. External Navigation and Visualiztion

9. Overview Page

10. Analytical List Page

11. Extend Fiori Elements List Report Object Page Application

12. Extend Fiori Elements Overview Page Application

13. RAP Basics

14. RAP Basics Delete, Update, Create

15. RAP with Draft version

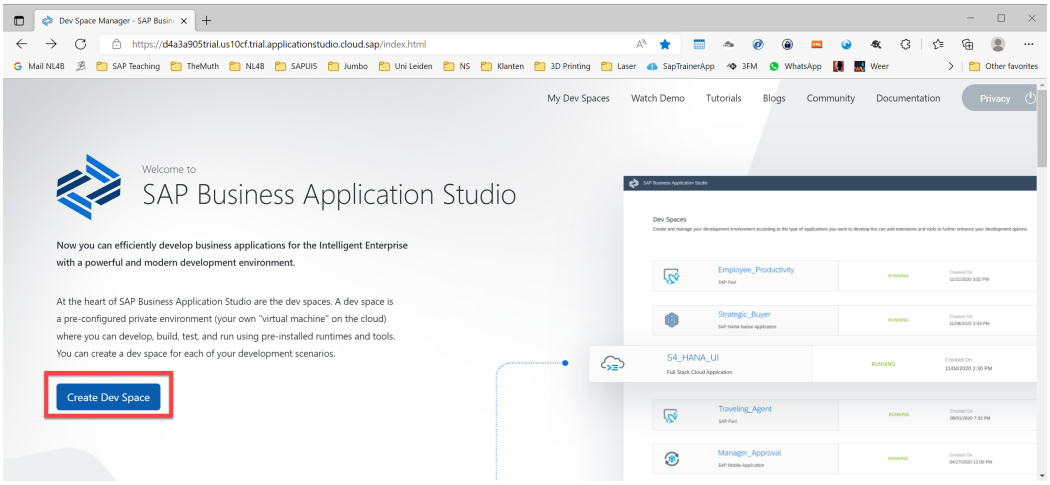
Create Package for Workshop

- Start Eclipse and open your ABAP cloud project.
- Right click and select *New* **ABAP Package**.
- Enter Name **Z_WKSP_##**, where ## is your assigned number.
- Enter Description *WKSP Exercises ##*
- Select `Next` and then `Finish`

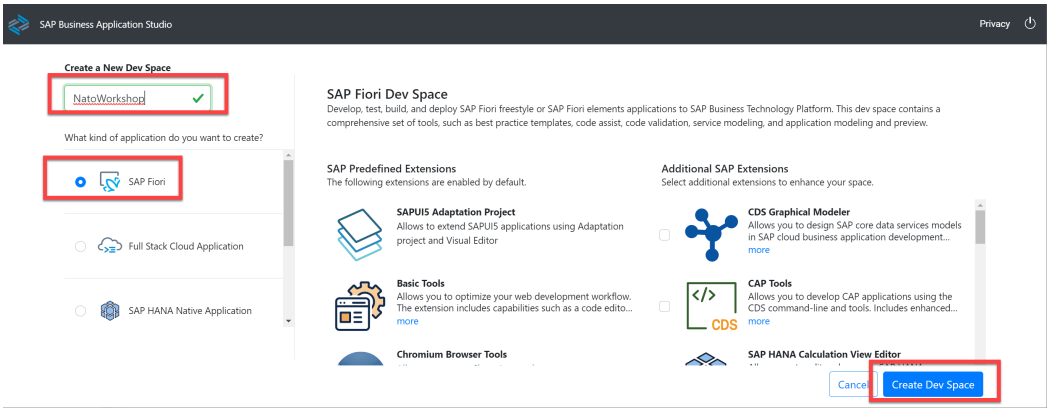
Create Dev Space in SAP Business Application Studio for Workshop

Start your SAP Business Application Studio.

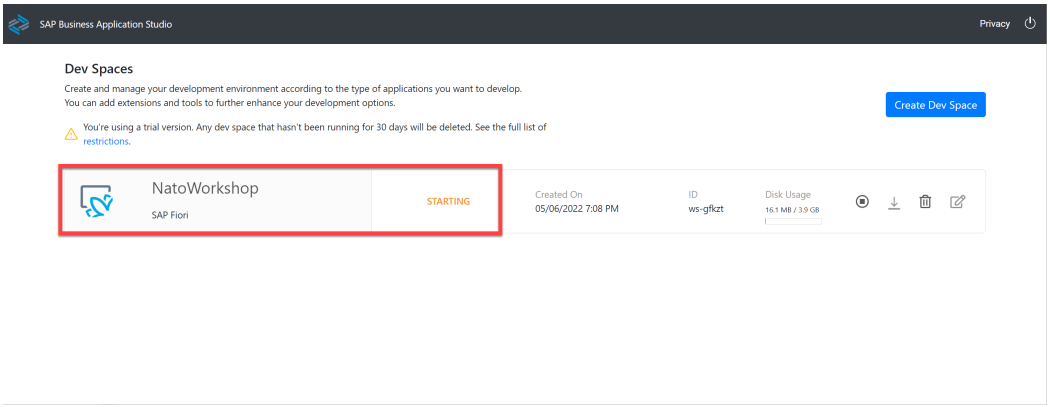
- Press the Create Dev Space button.



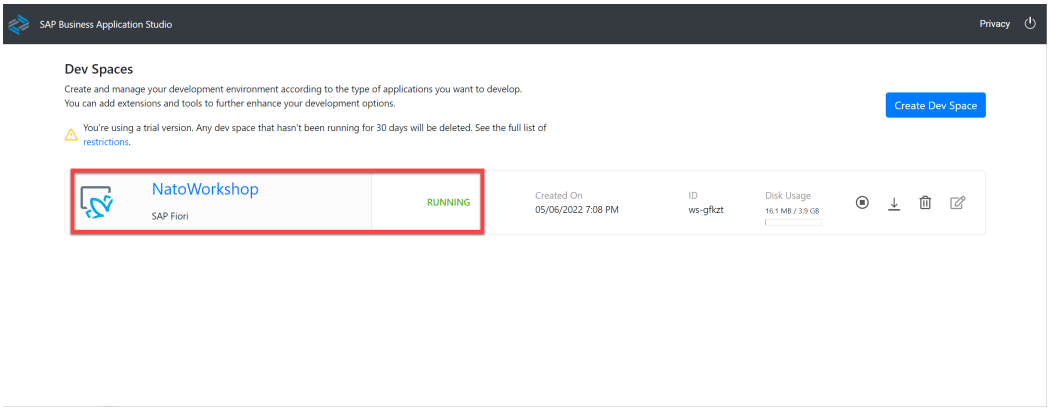
- Enter the name for your Dev Space, e.g.: **NatoWorkshop**, select **SAP Fiori** and pres the Create Dev Space button.

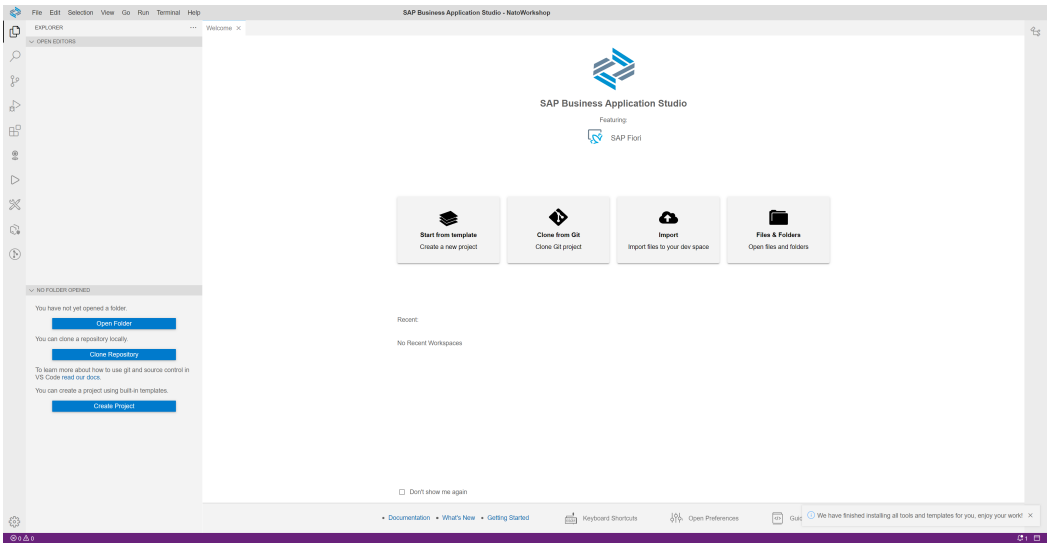


- This will create and start the Dev Space.



- When the Dev Space is running you can click on the name and the Business Application Studio is opened.





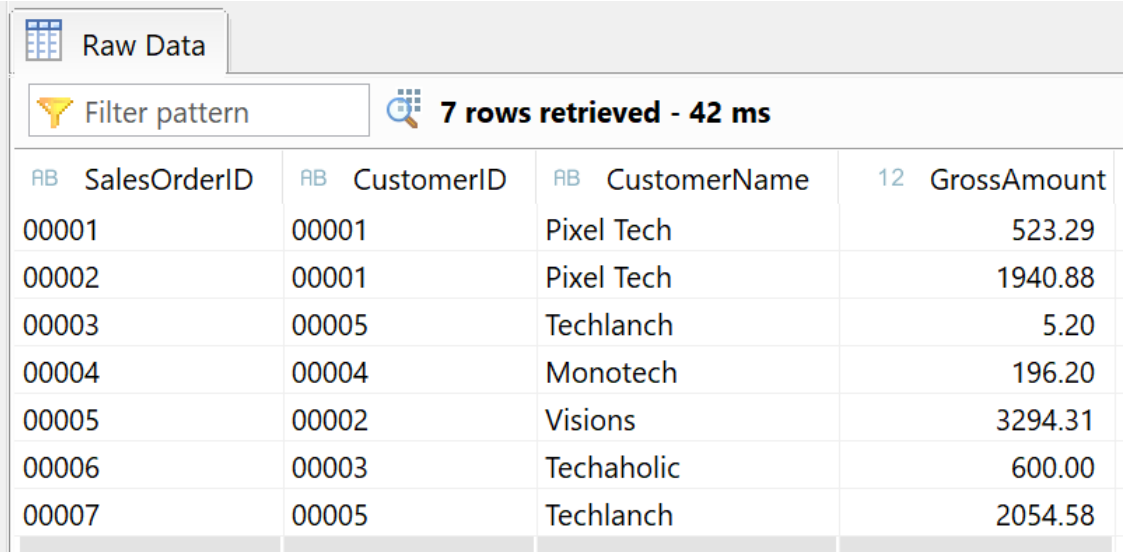
Create a simple CDS

Create a simple CDS with name **ZWS##_CDS_SIMPLE** where ## is your number.
Add the fields: Id, CustomerId, _Customer.name and GrossAmount.

> Template:

```
@AbapCatalog.sqlViewName: 'ZWK##CDSSIMPLE'
@AbapCatalog.compiler.compareFilter: true
@AbapCatalog.preserveKey: true
@AccessControl.authorizationCheck: #CHECK
@EndUserText.label: 'Simple CDS'
define view ZWS##_CDS_Simple as select from ztmcds9_i_so {
    key Id as SalesOrderID,
    CustomerId as CustomerID,
    _Customer.name as CustomerName,
    GrossAmount
}
```

Activate your CDS using CTRL-F3.
Test your CDS by opening it with **Data Preview**. This should result in a list of records.


Raw Data

Filter pattern

7 rows retrieved - 42 ms

SalesOrderID	CustomerID	CustomerName	GrossAmount
00001	00001	Pixel Tech	523.29
00002	00001	Pixel Tech	1940.88
00003	00005	Techlanch	5.20
00004	00004	Monotech	196.20
00005	00002	Visions	3294.31
00006	00003	Techaholic	600.00
00007	00005	Techlanch	2054.58

Create Service Definition

You are going to create a Service Definition in which you can list all the CDS views you want to expose in a OData service.
Right click on your Package and select *New* **Other ABAP Repository Object**, then type

Service and select **Service Definition**.
Enter Name **ZUI_WKSP_##** and enter a description.

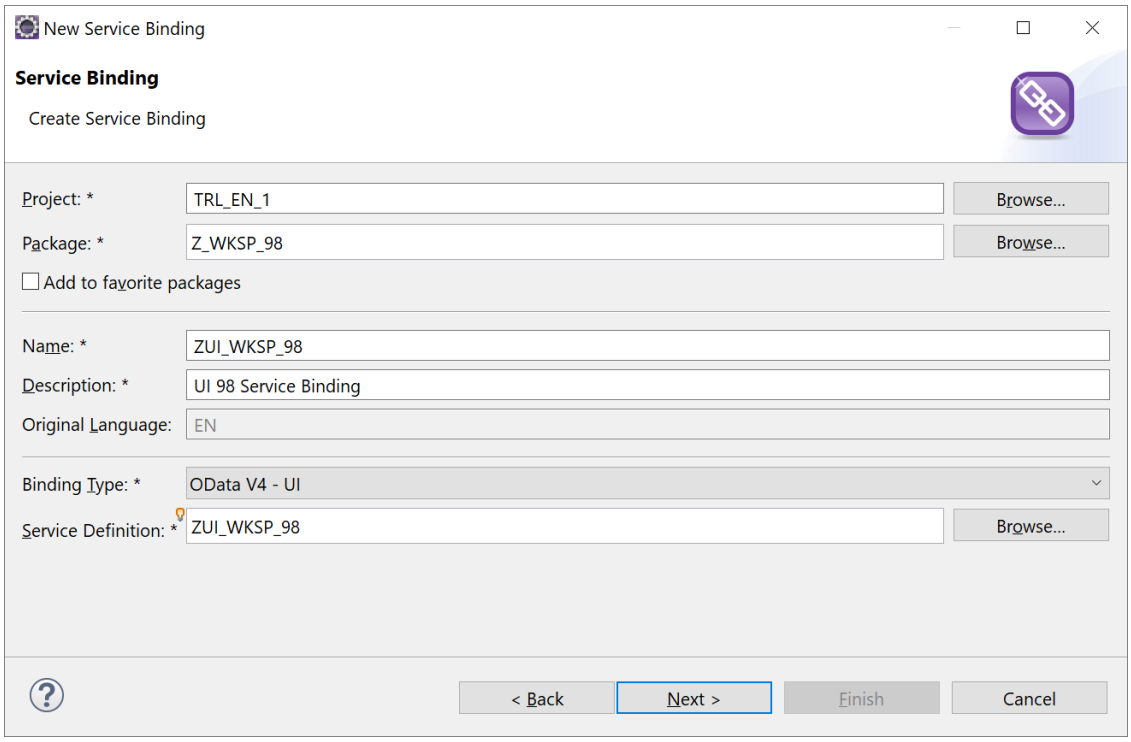
> Hint: Naming conventions are often used to specify if the service definition is used as a UI or API service.

Add your CDS as the entity name and give it a nice name like SimpleCDS.

```
@EndUserText.label: 'UI ## Service Definition'
define service ZUI_WKSP_## {
    expose ZWS##_CDS_Simple as SimpleCDS;
}
```

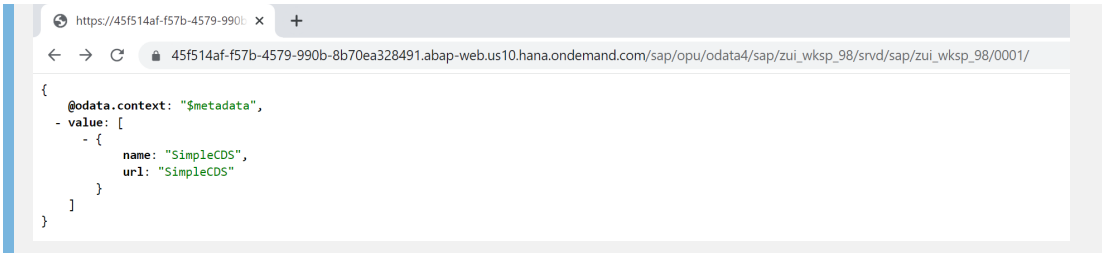
Create Service Binding

Now we are going to expose the Service Definition by creating a Service Binding.
Right click on your Package and select *New* **Other ABAP Repository Object**, then type *Service* and select **Service Binding**.
Enter Name **ZUI_WKSP_##** and enter a description. Then select *Binding Type* **OData V4 - UI**, and then enter your Service Definition. You can start typing the first letters of your Service Definition and then use CTRL+SPACE for Auto Complete, or use the `Browse` button.

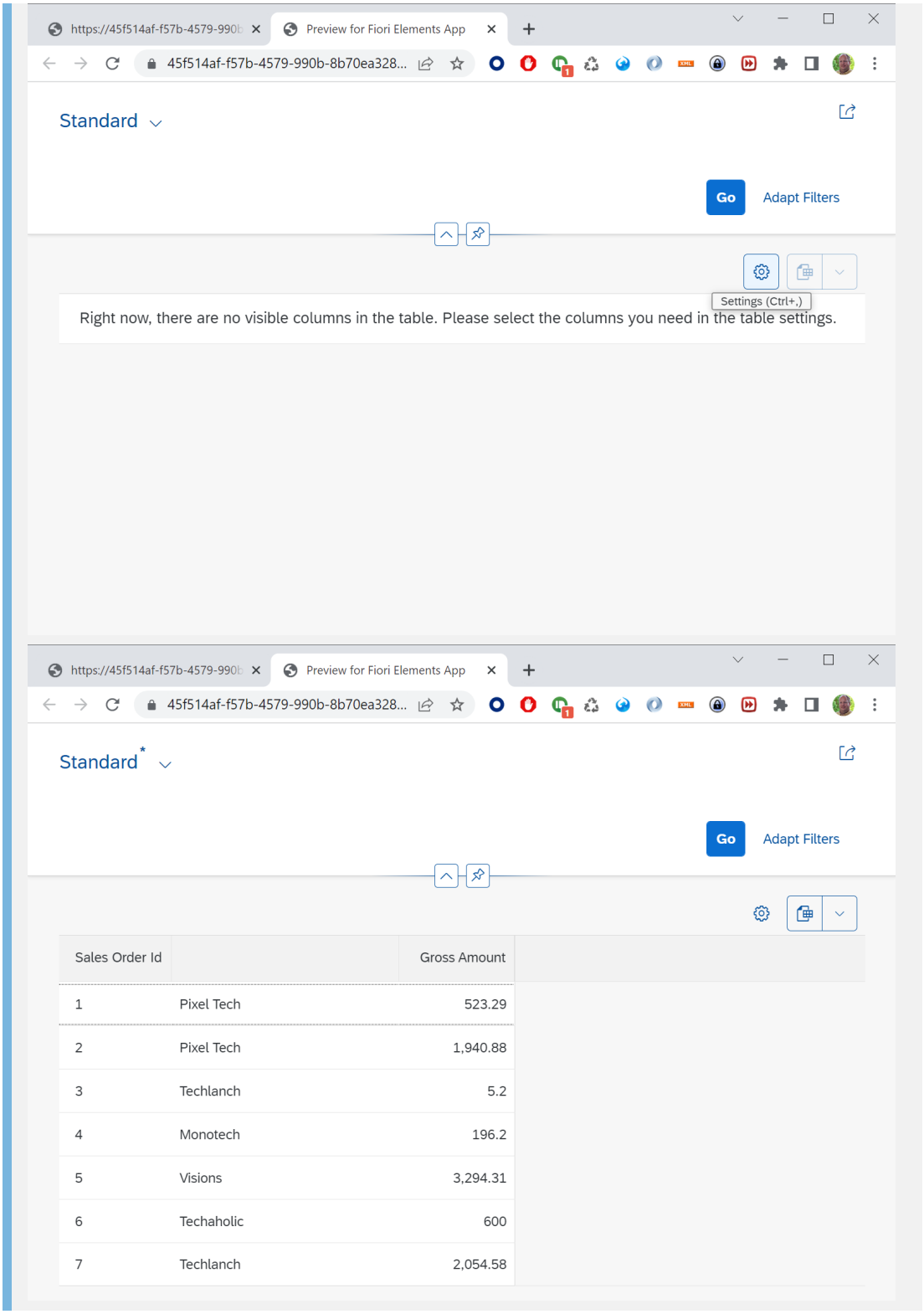


Activate your Service Binding and then push Publish (this may take a few seconds). Your Service Binding is now ready to be used.

If you press the ServiceURL a browser will be started and it will open the ROOT URL for your service.



If you select an Entity Set and press the `Preview` button, it will open a browser with an example application. But you still need to goto the settings to select the columns you want to display.



Add Some UI Annotiations

Now add some annotiations to make the Preview application display lines directly without have to go to the settings.
Add Header to table and select by default the Sales Order Id, CustomerName and

GrossAmount.

> Hint: @UI.headerInfo, @UI.lineItem

When you have added the annotations to your CDS, open the Preview again and see the result.

Option 1, Annotations in your CDS

```
@AbapCatalog.sqlViewName: 'ZWK##CDSSIMPLE'
@AbapCatalog.compiler.compareFilter: true
@AbapCatalog.preserveKey: true
@AccessControl.authorizationCheck: #CHECK
@EndUserText.label: 'Simple CDS'
@UI.headerInfo.typeName: 'Sales Order'
@UI.headerInfo.typeNamePlural: 'Sales Orders'
define view ZWS##_CDS_Simple as select from ztmcds9_i_so {
@UI.lineItem: [{position: 10 }]
    key Id as SalesOrderID,
    CustomerId as CustomerID,
@UI.lineItem: [{position: 20 },{ label: 'Customer Name' }]
    _Customer.name as CustomerName,
@UI.lineItem: [{position: 30 }]
    GrossAmount
}
```

Option 2, use a Metadata Extension

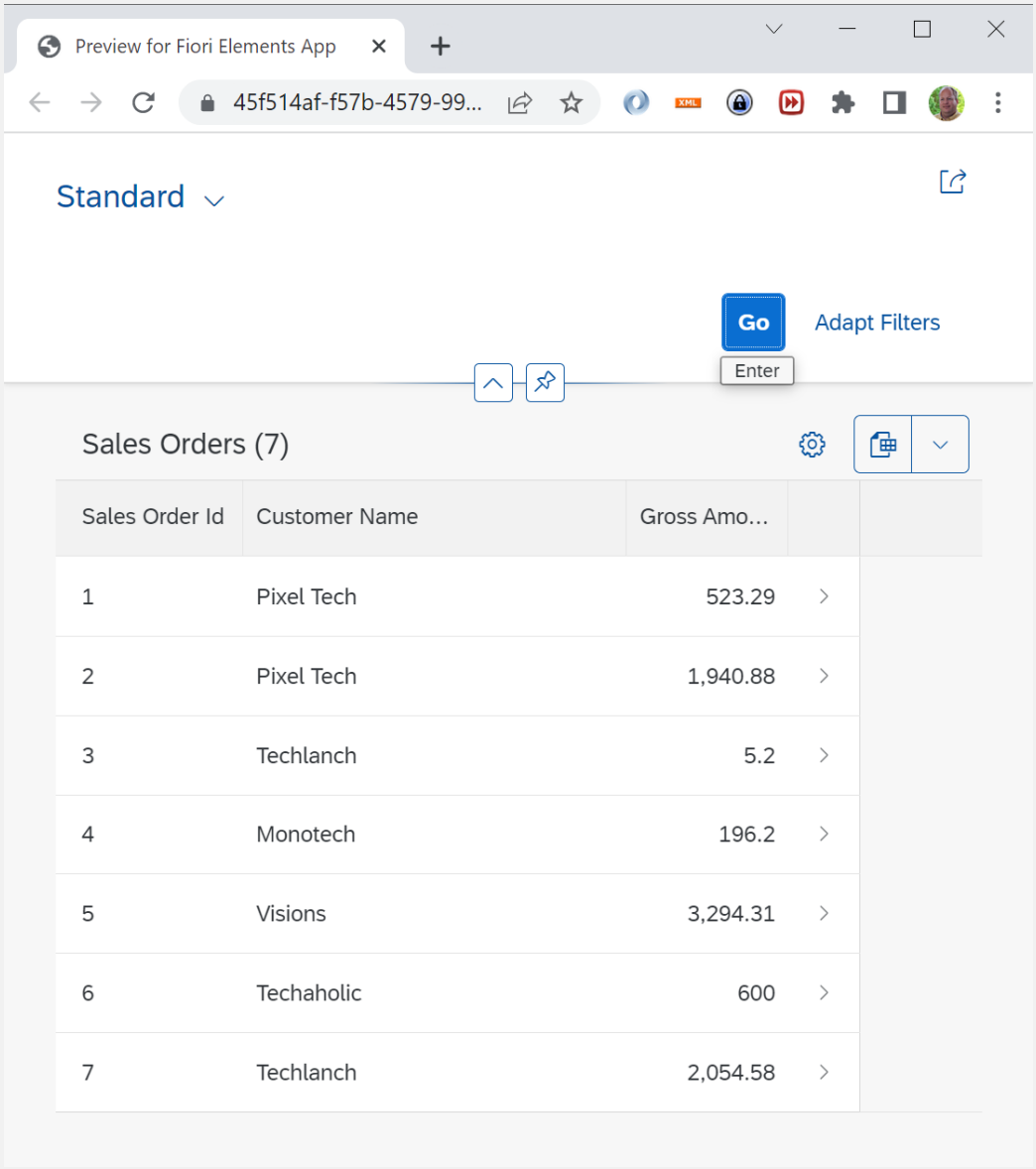
Create a Metadata Extension for your CDS. Add the annotation **@Metadata.allowExtensions: true** to your Simple CDS above the *define* line.

```
@AbapCatalog.sqlViewName: 'ZWS##CDSBASIC'
@AbapCatalog.compiler.compareFilter: true
@AbapCatalog.preserveKey: true
@AccessControl.authorizationCheck: #CHECK
@EndUserText.label: 'Basic CDS'
@Metadata.allowExtensions: true
define view ZWS##_CDS_Basic as select from ztmcds9_i_so {
    key Id as SalesOrderID,
    CustomerId as CustomerID,
    _Customer.name as CustomerName,
    GrossAmount
}
```

Right click on your Package and select *New* **Other ABAP Repository Object**, then type *Core* and select **Metadata Extension**.
And add the annotations for the fields to this file.

```
@Metadata.layer: #CUSTOMER
@UI.headerInfo.typeName: 'Sales Order'
@UI.headerInfo.typeNamePlural: 'Sales Orders'
annotate view ZWS##_CDS_Basic with
{
  @UI.lineItem: [{position: 10 }]
  SalesOrderID;
  @UI.lineItem: [{position: 20 },{ label: 'Customer Name' }]
  CustomerName;
  @UI.lineItem: [{position: 30 }]
  GrossAmount;
}
```

Result:


Preview for Fiori Elements App
45f514af-f57b-4579-99...
Standard
Go
Adapt Filters
Enter
Sales Orders (7)
Sales Order Id
Customer Name
Gross Amo...
1 Pixel Tech 523.29
2 Pixel Tech 1,940.88
3 Techlanch 5.2
4 Monotech 196.2
5 Visions 3,294.31
6 Techaholic 600
7 Techlanch 2,054.58

Create Basic List Report

Copy your *ZWS##_CDS_SIMPLE* to **ZWS##_CDS_LIST**.
Remember to change the @AbapCatalog.sqlViewName and @EndUserText.label values.

```
@AbapCatalog.sqlViewName: 'ZWK##CDSLIST'
@AbapCatalog.compiler.compareFilter: true
@AbapCatalog.preserveKey: true
@AccessControl.authorizationCheck: #CHECK
@EndUserText.label: 'List Report'
@UI.headerInfo.typeName: 'Sales Order'
@UI.headerInfo.typeNamePlural: 'Sales Orders'
define view ZWS##_CDS_LIST as select from ztmcds9_i_so {
@UI.lineItem: [{position: 10 }]
    key Id as SalesOrderID,
    CustomerId as CustomerID,
@UI.lineItem: [{position: 20 },{ label: 'Customer Name' }]
    _Customer.name as CustomerName,
@UI.lineItem: [{position: 30 }]
    GrossAmount
}
```

Activate the CDS and add it to the Service Binding as **BasicList**

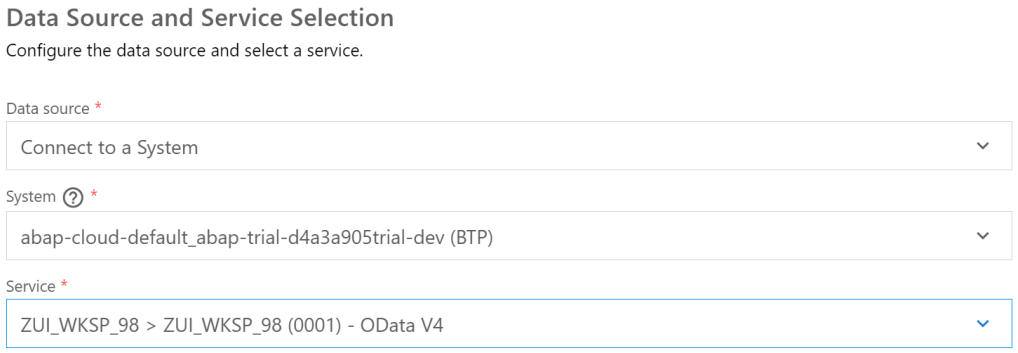
```
@EndUserText.label: 'UI 98 Service Definition'
define service ZUI_WKSP_98 {
  expose ZWS##_CDS_Simple as SimpleCDS;
  expose ZWS##_CDS_Basic as BasicCDS;
  expose ZWS##_CDS_LIST as BasicList;
}
```

Start your SAP Business Application Studio.
Open/Create your Dev Space.

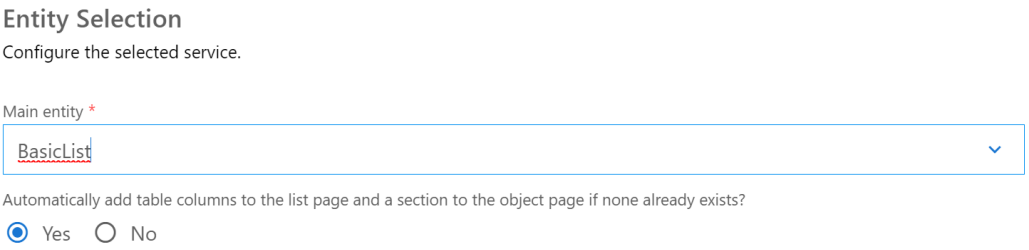
There are two options to start the Fiori Template Wizard.

1. From the Welcome page you select *Start from template* and then select the **SAP Fiori application**
2. From the menu select *View* then *Find Command...* *(`Ctrl+shift+P`)* and then type **Fiori: Open Application Generator**

- Select the Application Type: **SAP Fiori elements**, and select the **List Report Object Page** and press `Next`.
  In the Data source select *Connect to a System*, in the System select *your abap-cloud-default..(BTP)*. It will now look for the OData services on the Trial ABAP Cloud system, this may take a minute.
  In the Service select your Service Binding ZUI_WKSP_##.



- Press `Next`
- Select your **BasicList** as the *Main entity*.



- Press `Next`
  Enter the Projects Attributes

Name	Value
Module name	basic-list
Application title	Basic List
Application namespace	nato.workshop
Description	Basic List
Project folder path	/home/user/projects
Minimum SAPUI5 version	leave as is
Add deployment configuration	Yes
Add FLP configuration	Yes
Configure advanced options	No

Project Attributes

Configure the main project attributes.

Module name *

basic-list

Application title

Basic List

Application namespace

nato.workshop

Description

Basic List

Project folder path *

/home/user/projects

Minimum SAPUI5 version *

1.96.7 (Source system version)

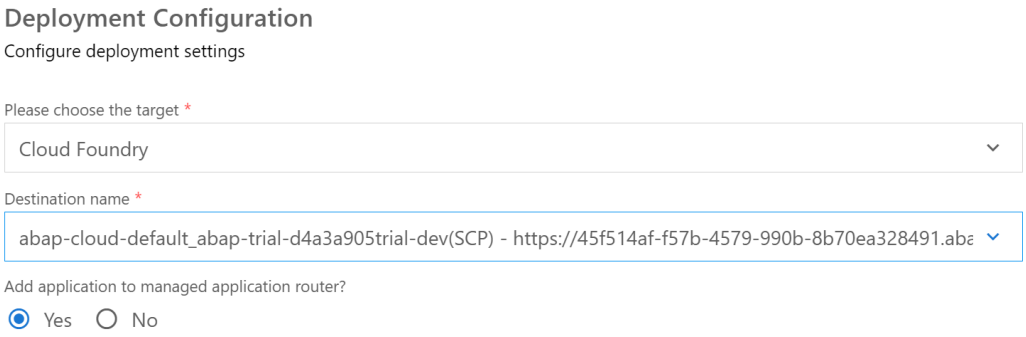
Add deployment configuration

(•) Yes ( ) No

Add FLP configuration

(•) Yes ( ) No

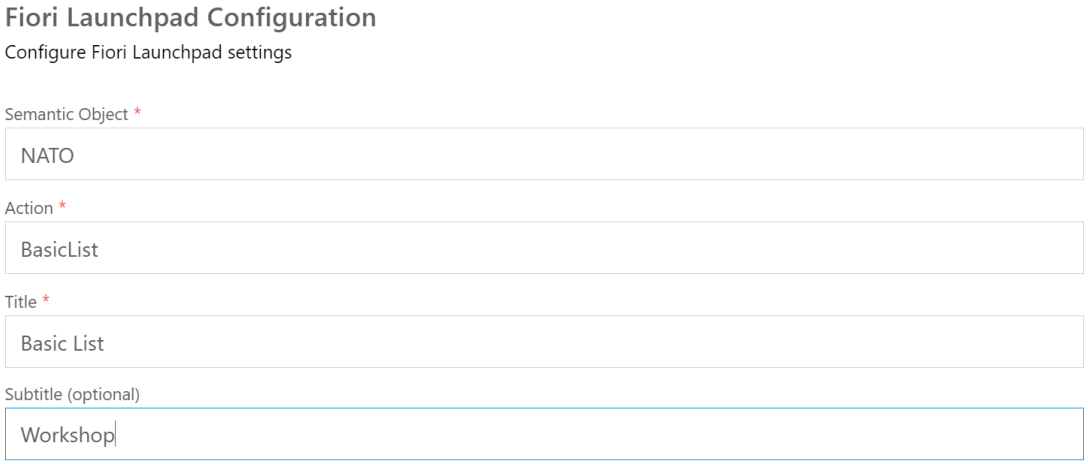
- Press `Next`



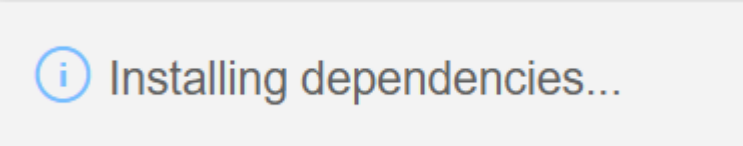
- Press `Next`

Enter Fiori Launchpad settings

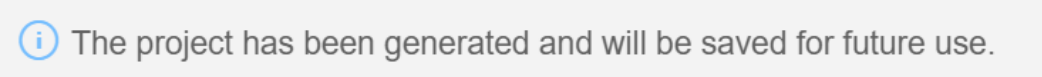
Name	Value
Semantic Object	NATO
Action	BasicList
Title	Basic List
Subtitle (optional)	Workshop



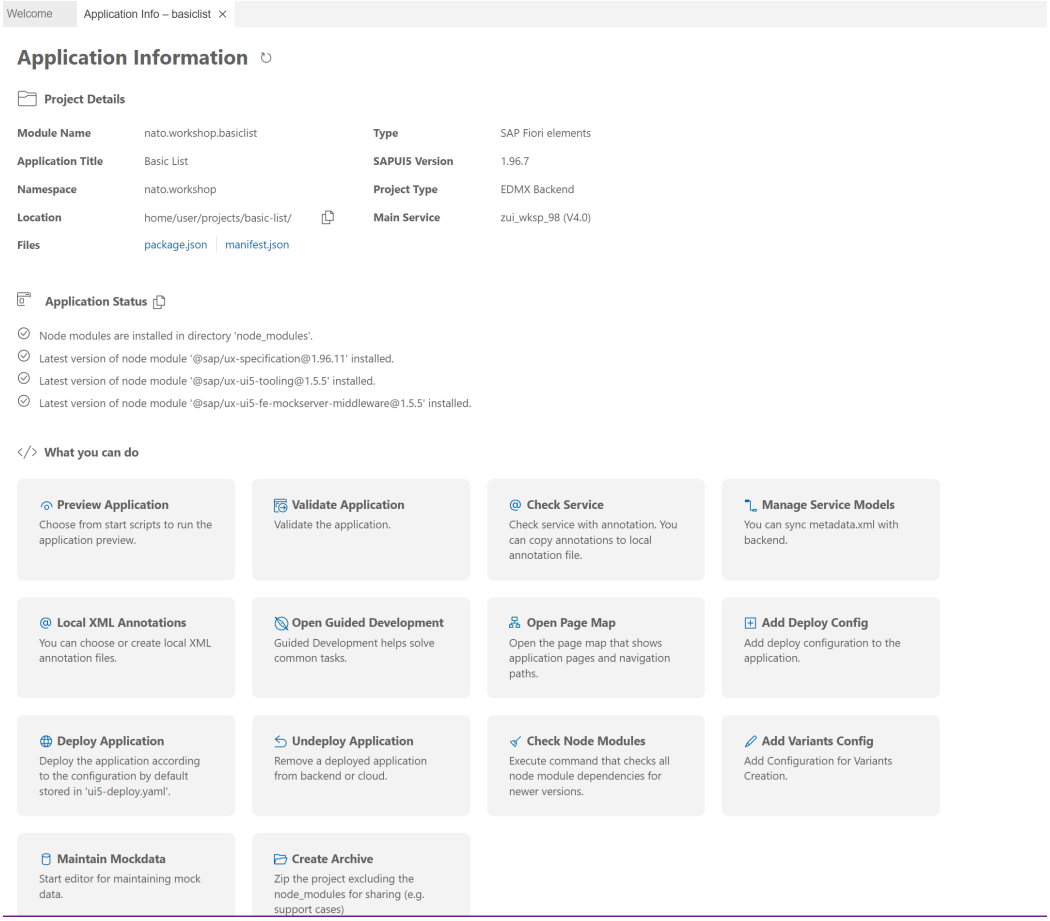
- Press `Finish`



- The installation of the dependencies may take a few moments.



- When is is done you will see a page with Application Information.

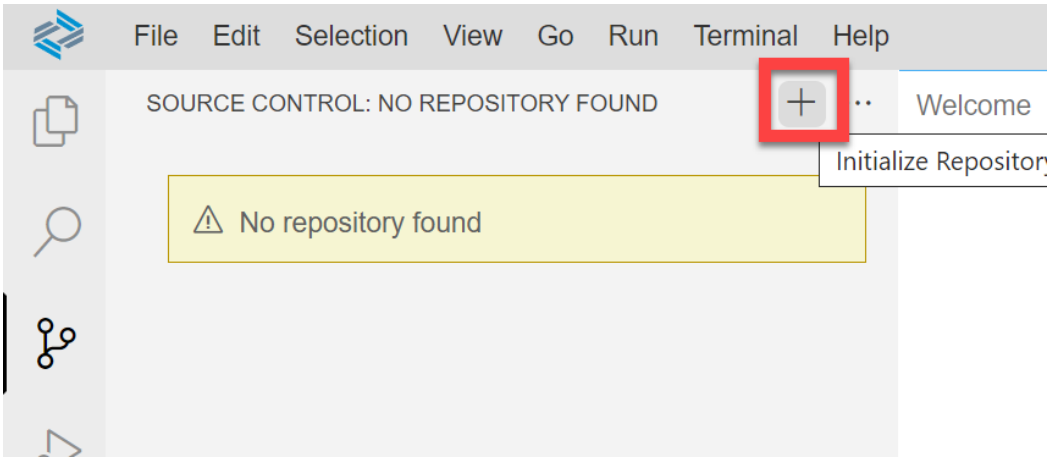


- Press *Preview Application* to start the application.

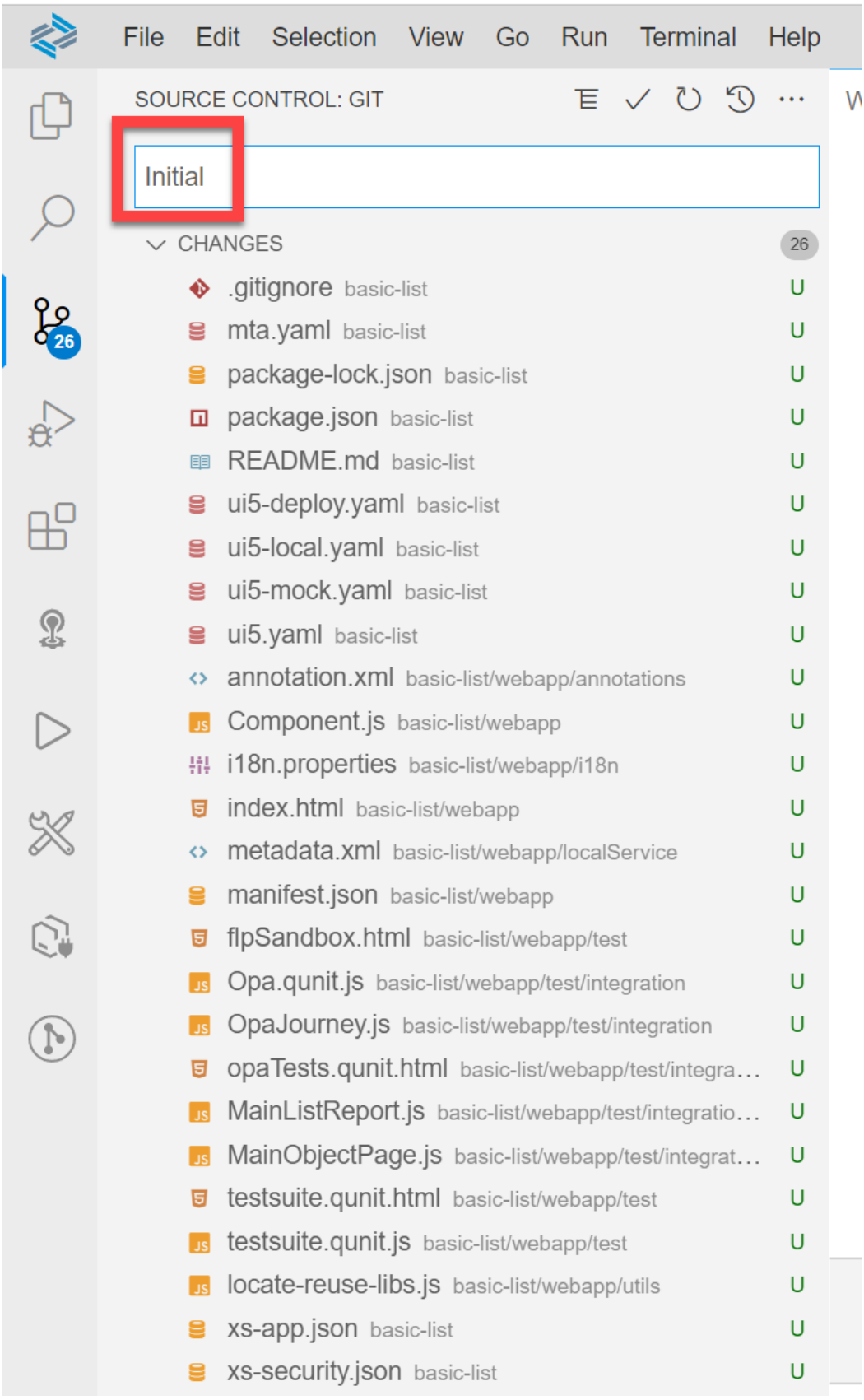
Open the project folder and activate GIT

- Open the projects folder.

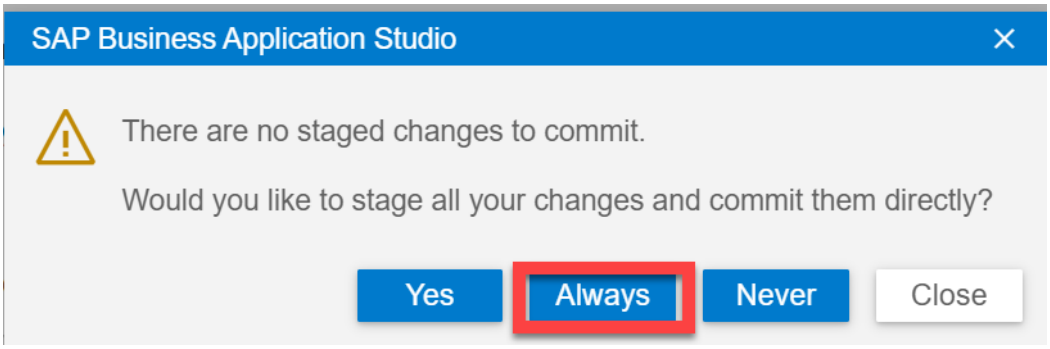
- Click on the *GIT* tab.  and press the + Plus button.



- This will instantiate the GIT source control. Enter a message, e.g.: **Initial** and press **CTRL+Enter**.



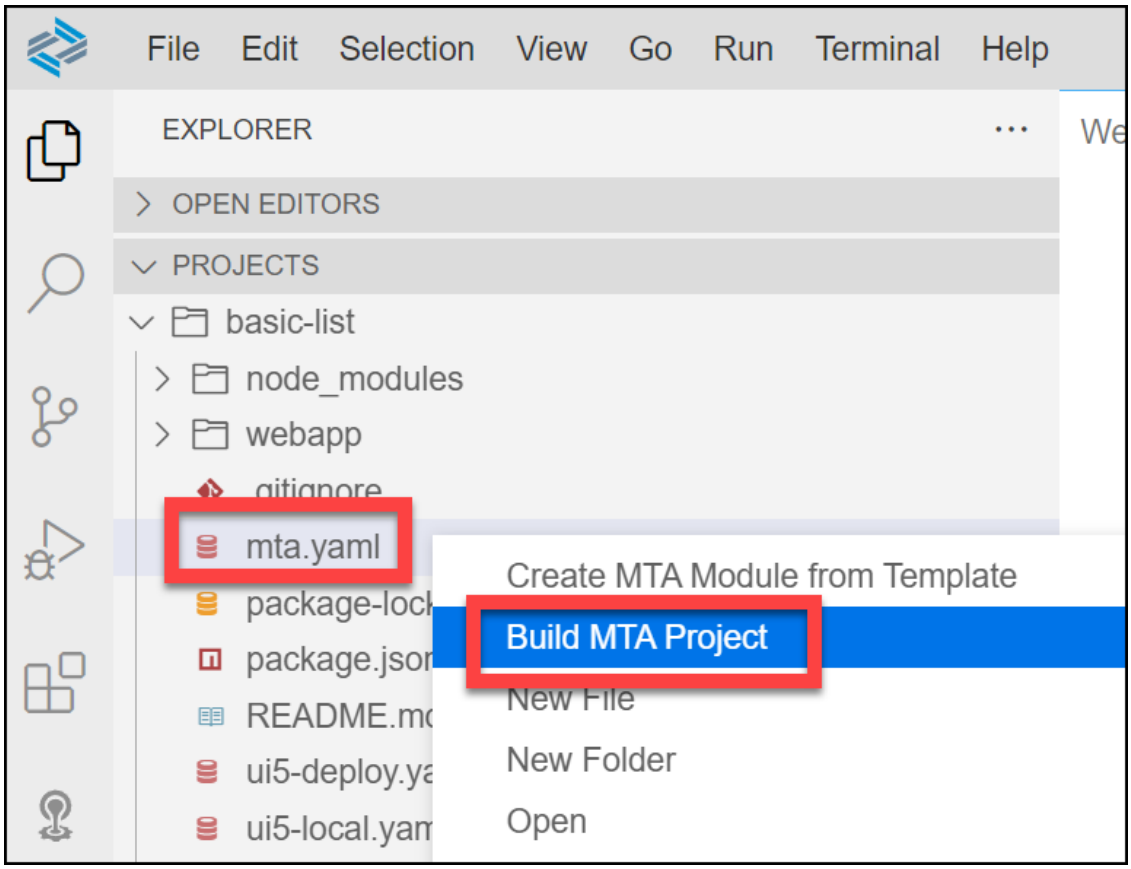
Select *Always* in the popup.


SAP Business Application Studio
There are no staged changes to commit.
Would you like to stage all your changes and commit them directly?
Yes
Always
Never
Close

Build & Deploy Application and Configure Fiori Cloud Launchpad

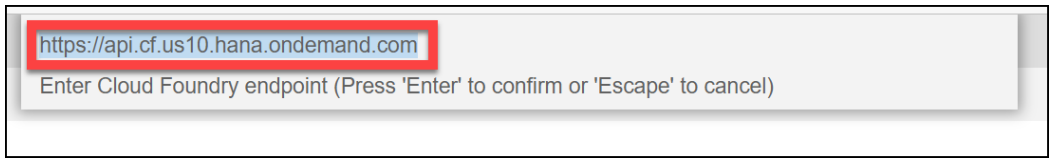
Build and Deploy

- Right click on the **mta.yaml** file and select *Build MTA Project*



This will start a terminal and it will create 3 new directories: *dist*, *mta_archives* and *resources*.

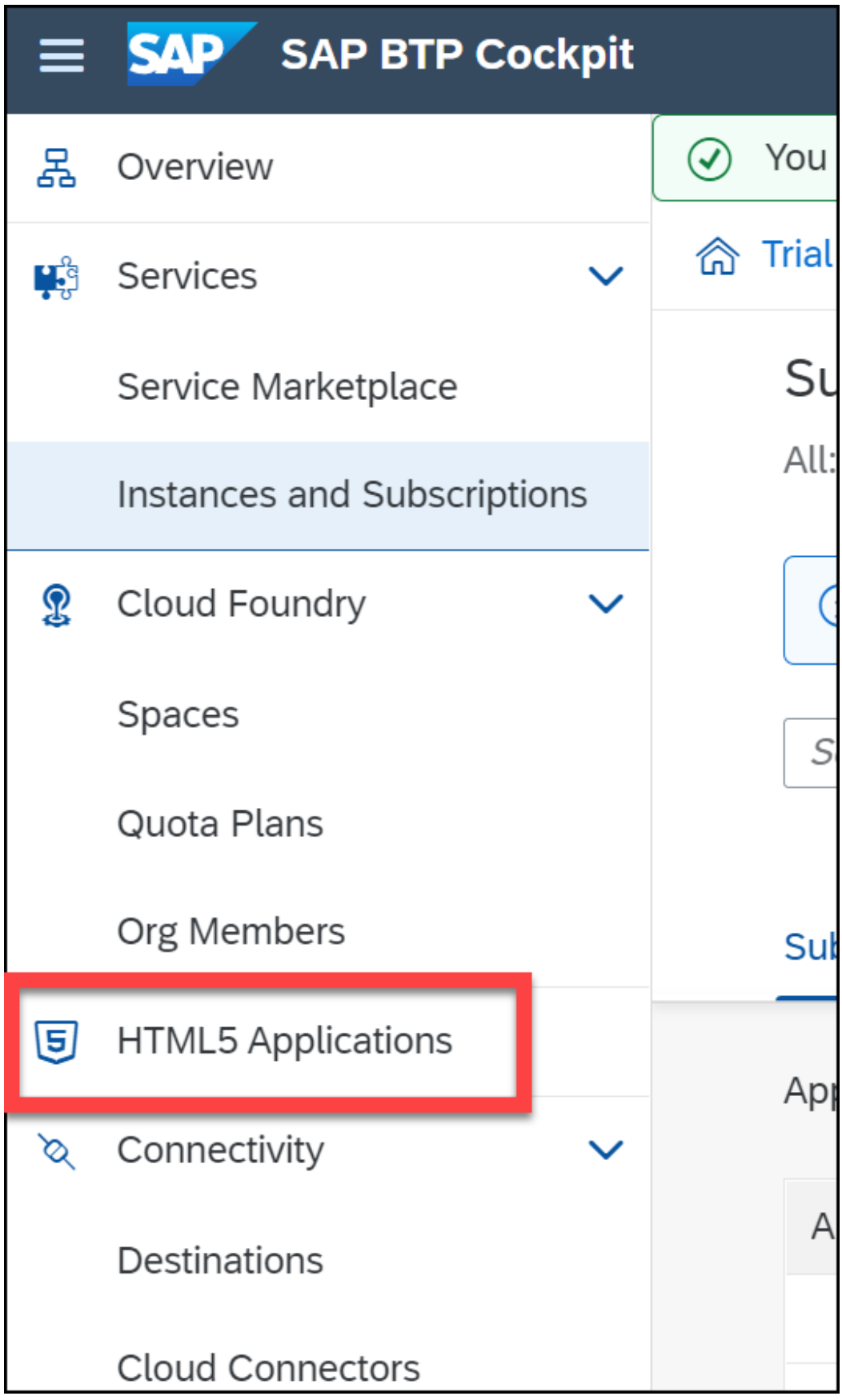
- Open the *mta_archives* folder and right click the **.mtar** file and select *Deploy MTA Archive*
- Check your Cloud Foundry Endpoint and press *Enter* to confirm.



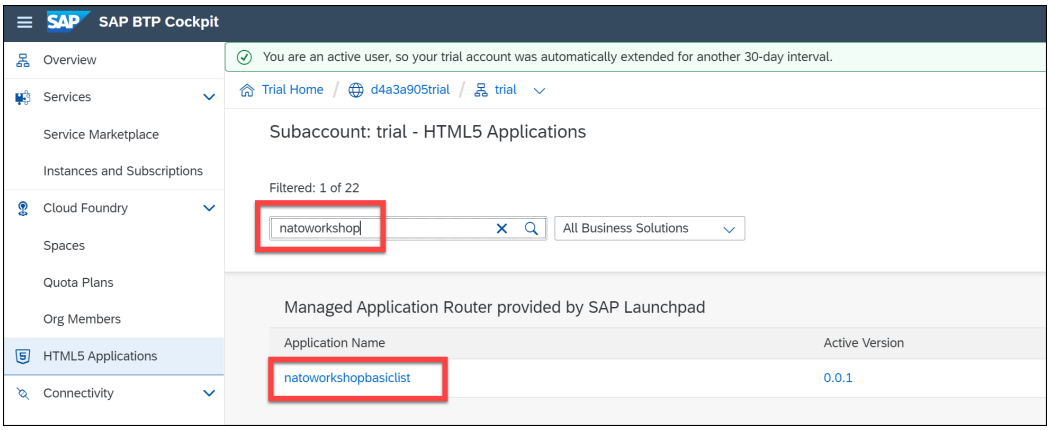
- Enter your e-mail address and password of your trial account.
- Select your Organisation and Space

This will open another terminal that will deploy the **.mtar** to the cloud. This can take a moment.

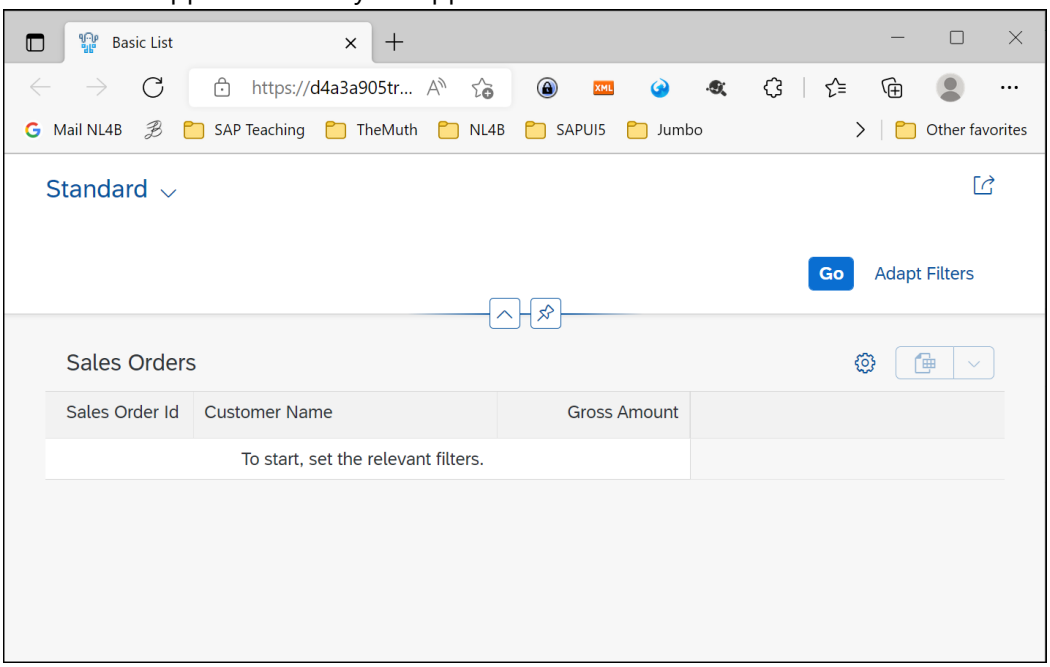
- To see if all went well you can goto your SAP BTP Trial Cockpit and the option **HTML5 Applications** on the left side.


SAP BTP Cockpit
Overview
Services
Service Marketplace
Instances and Subscriptions
Cloud Foundry
Spaces
Quota Plans
Org Members
HTML5 Applications
Connectivity
Destinations
Cloud Connectors

- If you filter on *natoworkshop* you should see your application.

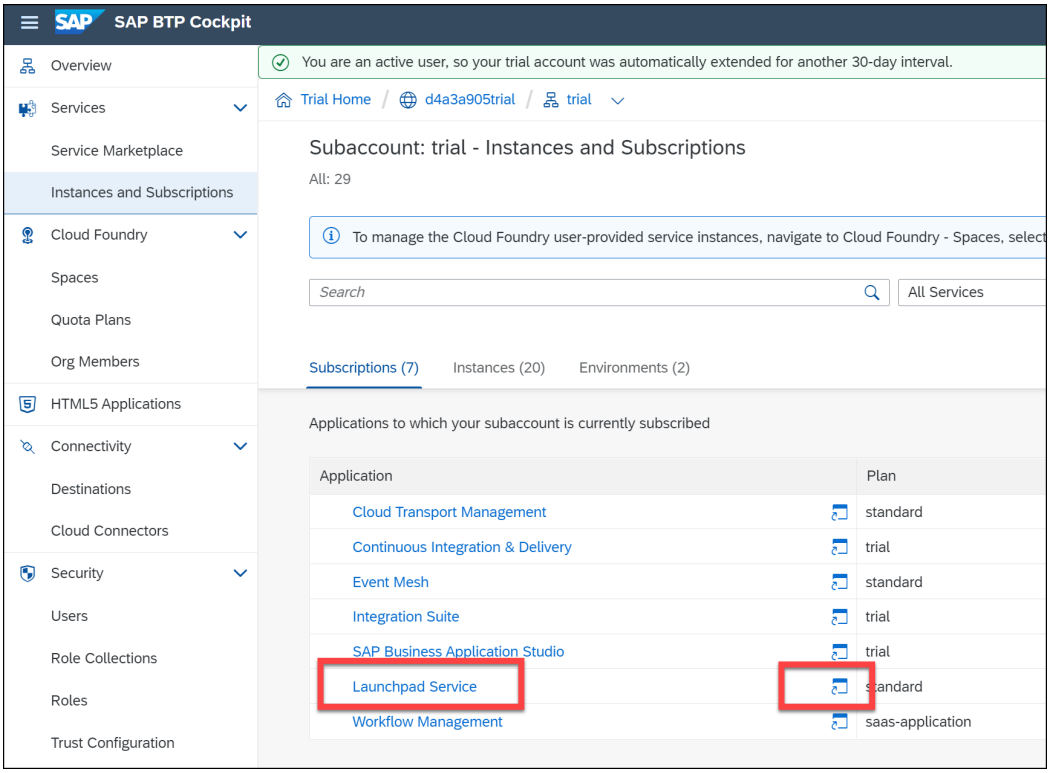


- Click on the application and your application will start.

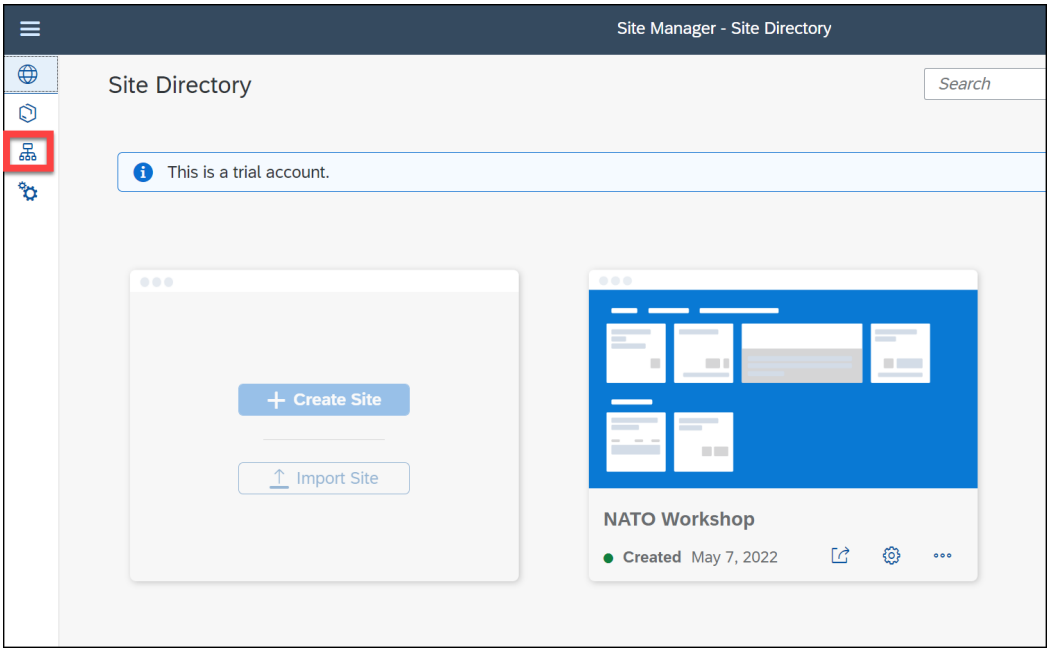


Configure Fiori Launchpad Cloud

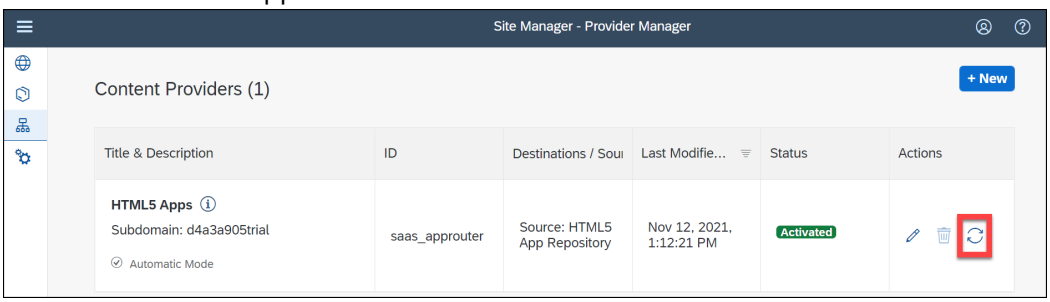
- Goto *Instances and Subscriptions* in your BTP Cockpit and goto the Launchpad Service



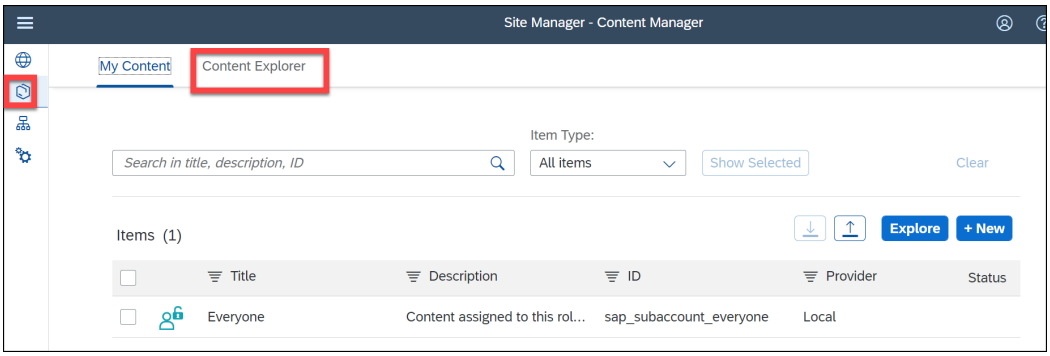
- In the Site Directory select the **provider manager**



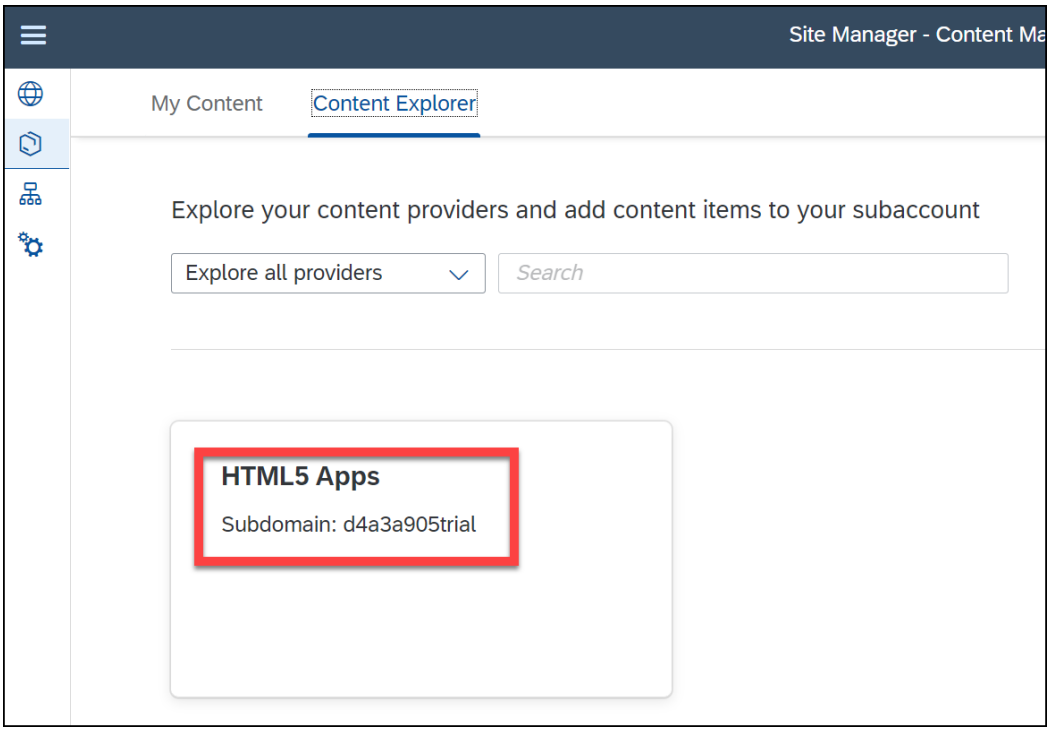
- Refresh the HTML5 Apps



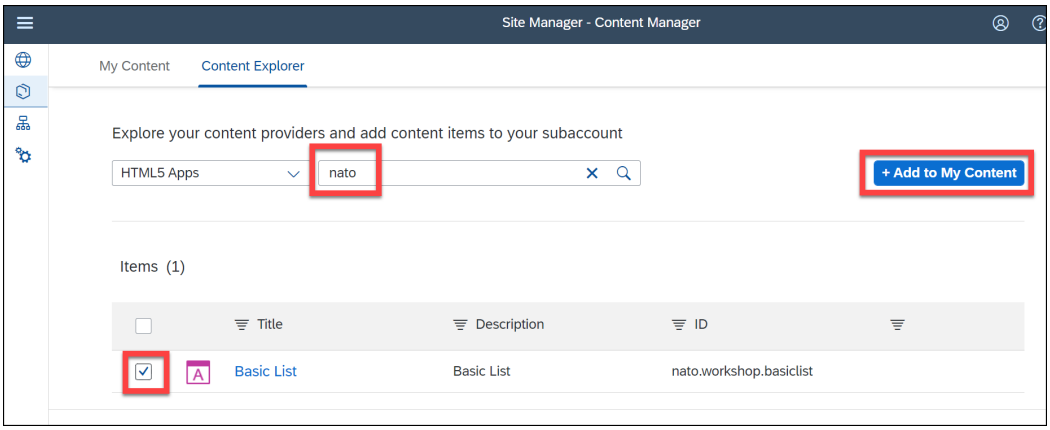
- Goto the **Content Manager** and click the tab *Content Explorer*



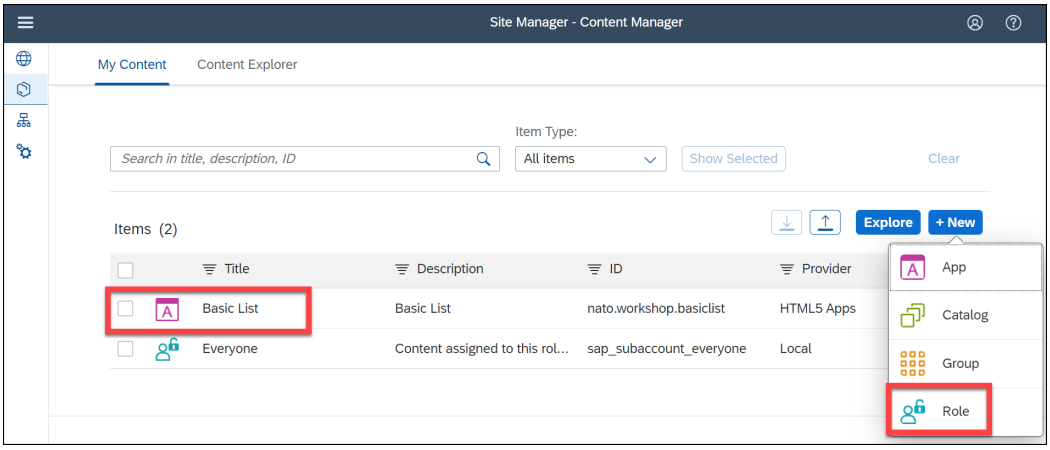
- Click the **HTML5 Apps**



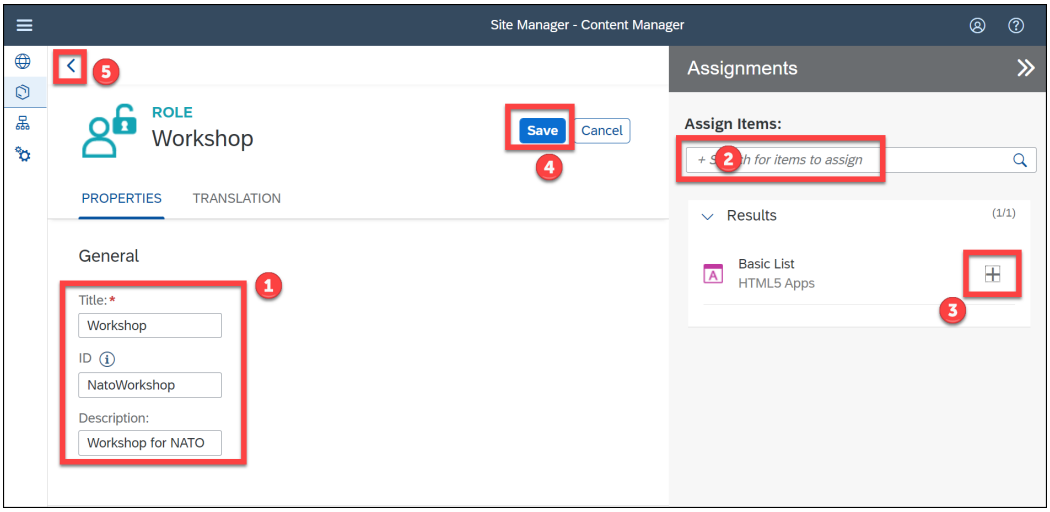
- User filter **nato** to list your **Basic List** application and add it to My Content



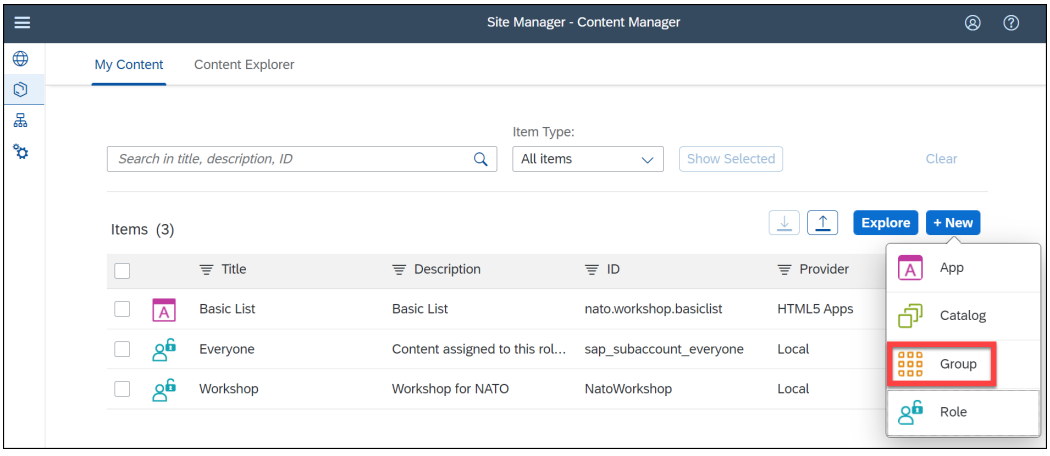
- Goto the tab *My Content*, here you see your **Basic List** application.
  Now lets create a Role.



- Enter the *Title*, *ID* and *Description*, then click on the search block under Assign Items:, you now see a your application. Press the + button behind your application, then *Save* and then *Back*

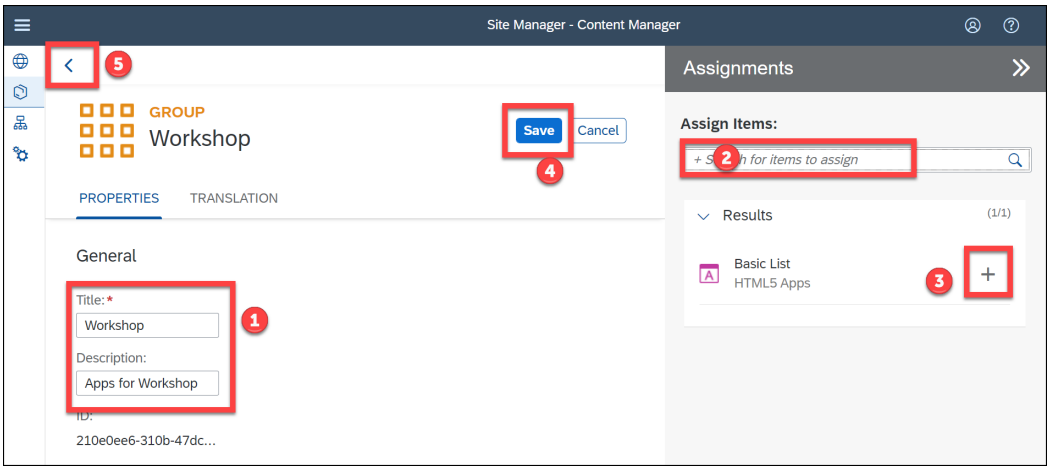


- We now have a new Role
  Now lets create a Group.

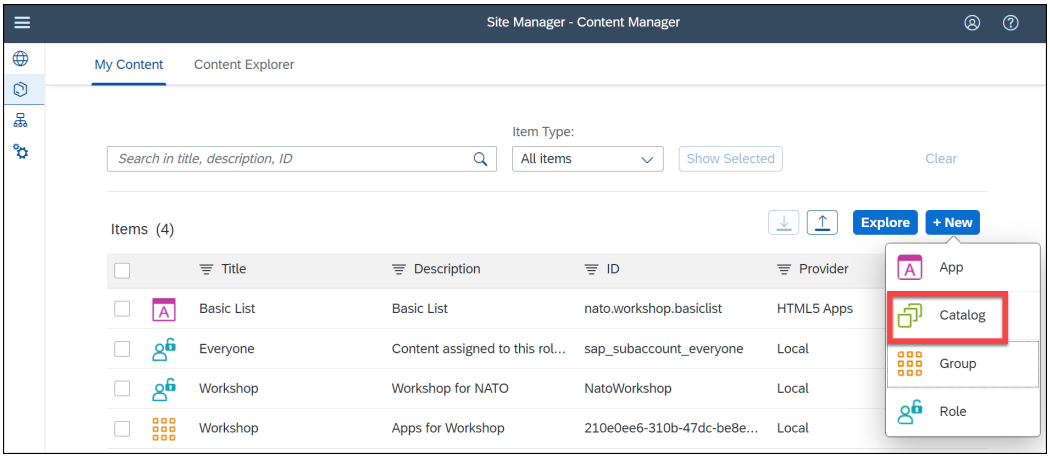


- Enter the *Title* and *Description*, then click on the search block under Assign Items:, you now see a your application. Press the + button behind your application, then

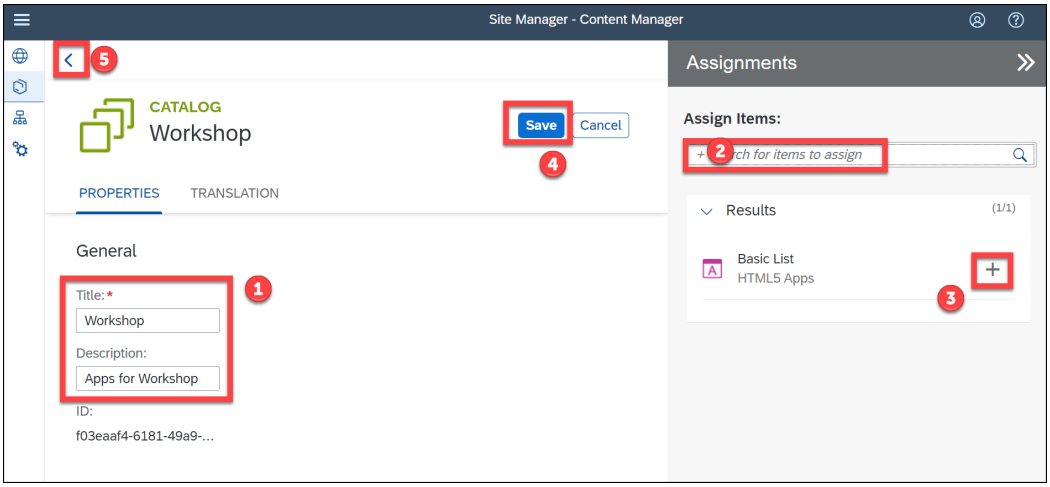
Save and then *Back*



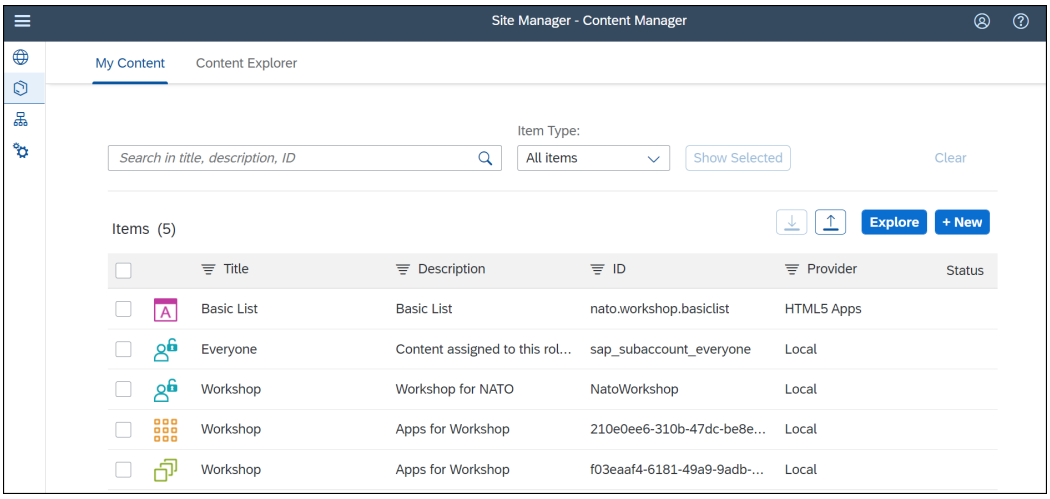
- We now have a new Group
  Now lets create a Catalog



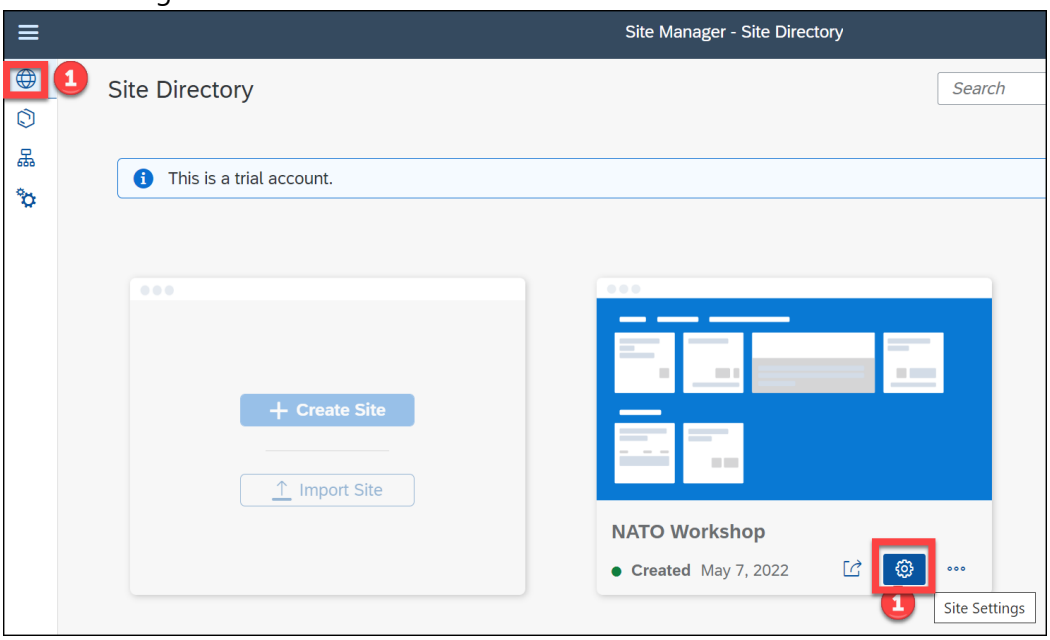
- Enter the *Title* and *Description*, then click on the search block under Assign Items:, you now see a your application. Press the + button behind your application, then *Save* and then *Back*!



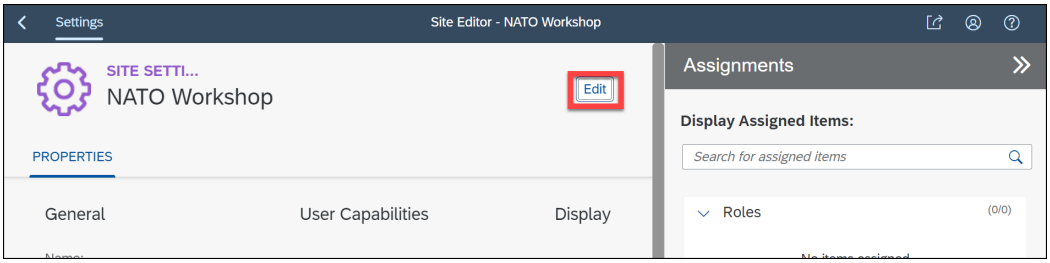
- We now have a new Catalog



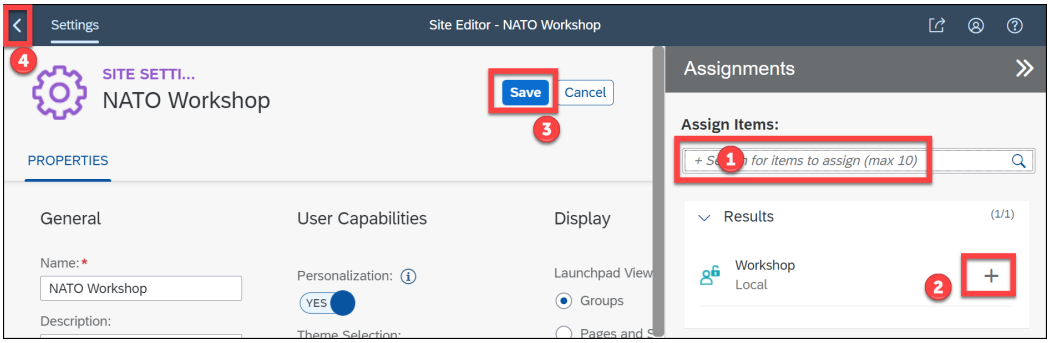
- Next we assign the created Role to our Site



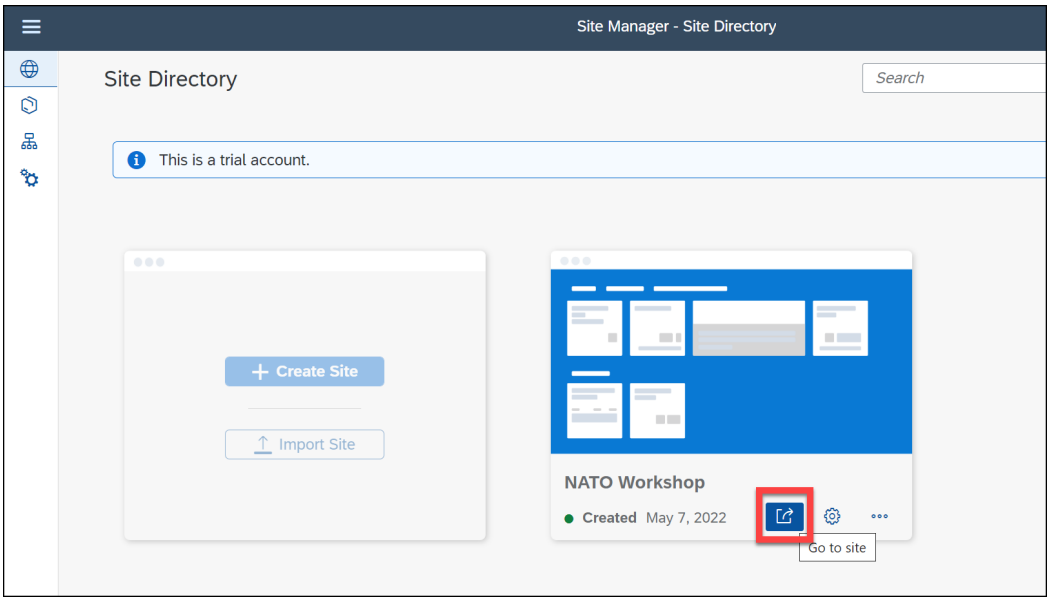
- We *Edit* the Site



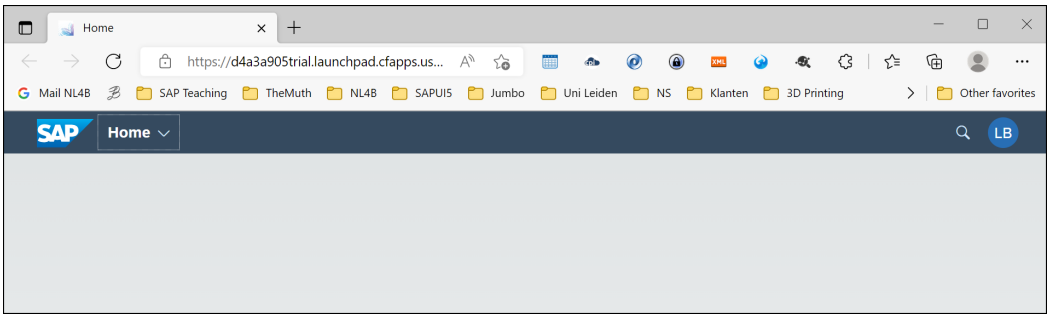
- Click on the search block under Assign Items:, you now see a your Role. Press the + button behind your role, then *Save* and then *Back*



- Goto the Site/Launchpad

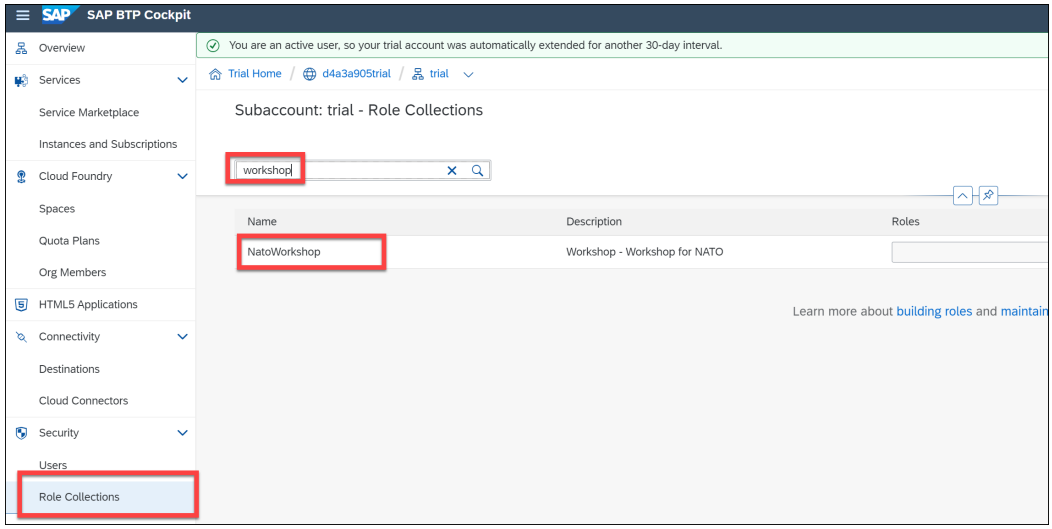


- Our Launchpad is still empty.

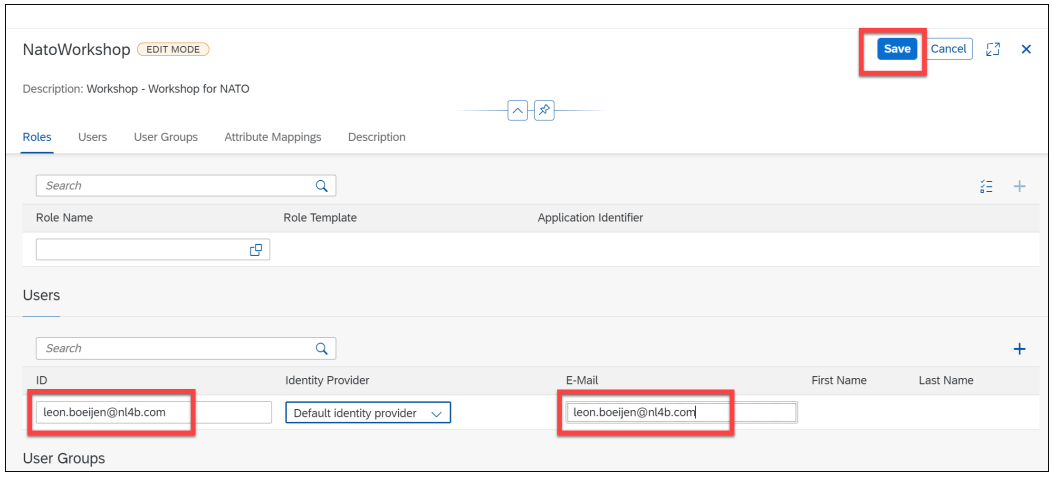


This is because we have not yet assigned the new Role to our user.

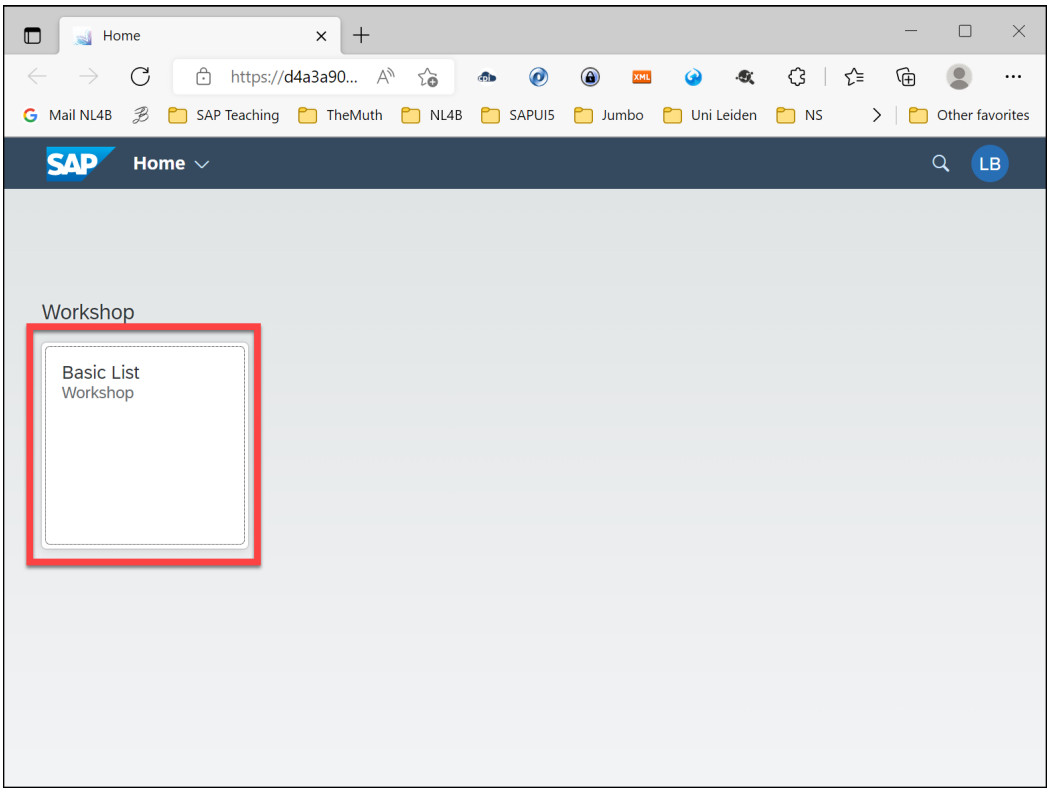
- Goto your BTP Cockpit and select *Role Collections*, Filter on **workshop** and click on your Role



- Goto *Edit* mode and enter your e-mail address of your trial account in the fields **ID** and **E-Mail** and press *Save*



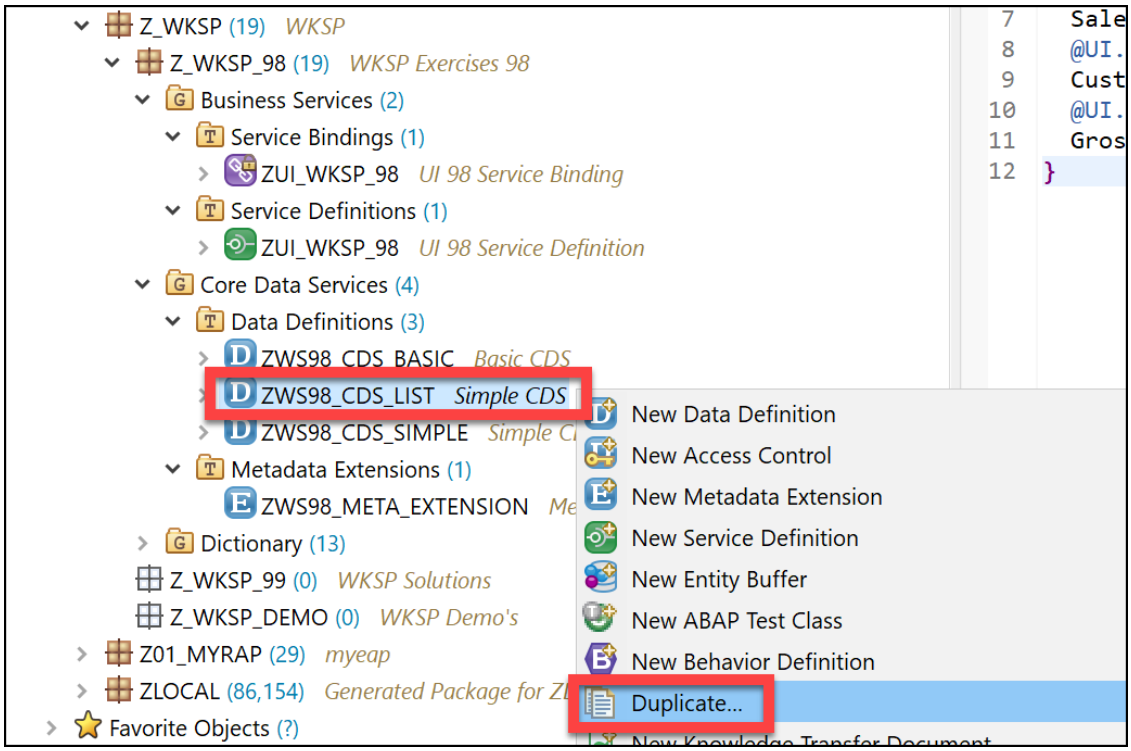
- Now lets start our Launchpad again.



> Hint: it is sometimes needed to start a browser in *Private/Incognito mode* and lauch the Fiori Launchpad again, to make your changes available.

Add Search and Filters to List Report

Task 1: Duplicate CDS **ZWS##_CDS_LIST** to **ZWS##_CDS_LIST_SEARCH**



Make sure you change the *@AbapCatalog.sqlViewName*.

Add extra field for CreationDate and Customer Email.

```
@AbapCatalog.sqlViewName: 'ZWK##CDSLISTSRCH'
@AbapCatalog.compiler.compareFilter: true
@AbapCatalog.preserveKey: true
@AccessControl.authorizationCheck: #CHECK
@EndUserText.label: 'List Report'
@UI.headerInfo.typeName: 'Sales Order'
@UI.headerInfo.typeNamePlural: 'Sales Orders'
define view ZWS##_CDS_LIST_SEARCH
  as select from ztmcds9_i_so
{
      @UI.lineItem: [{position: 10 }]
  key Id
as SalesOrderID,
      CustomerId
as CustomerID,
      @UI.lineItem: [{position: 20 },{ label: 'Customer Name' }]
      _Customer.name
```

```
as CustomerName,
      @UI.lineItem: [{position: 30 },{ label: 'E-mail' }]
      _Customer.email
as CustomerEmail,
      @UI.lineItem: [{position: 40 }]
      GrossAmount,
      @UI.lineItem: [{position: 50},{ label: 'Creation Date' }]
      cast( left(cast(CreationDateTime as abap.char( 30)),8) as
abap.dats) as CreationDate
}
```

Task 2: Add the new CDS to your *Service Definition*

```
@EndUserText.label: 'UI ## Service Definition'
define service ZUI_WKSP_## {
  expose ZWS##_CDS_Simple as SimpleCDS;
  expose ZWS##_CDS_Basic as BasicCDS;
  expose ZWS##_CDS_LIST as BasicList;
  expose ZWS##_CDS_LIST_SEARCH as BasicSearch;
}
```

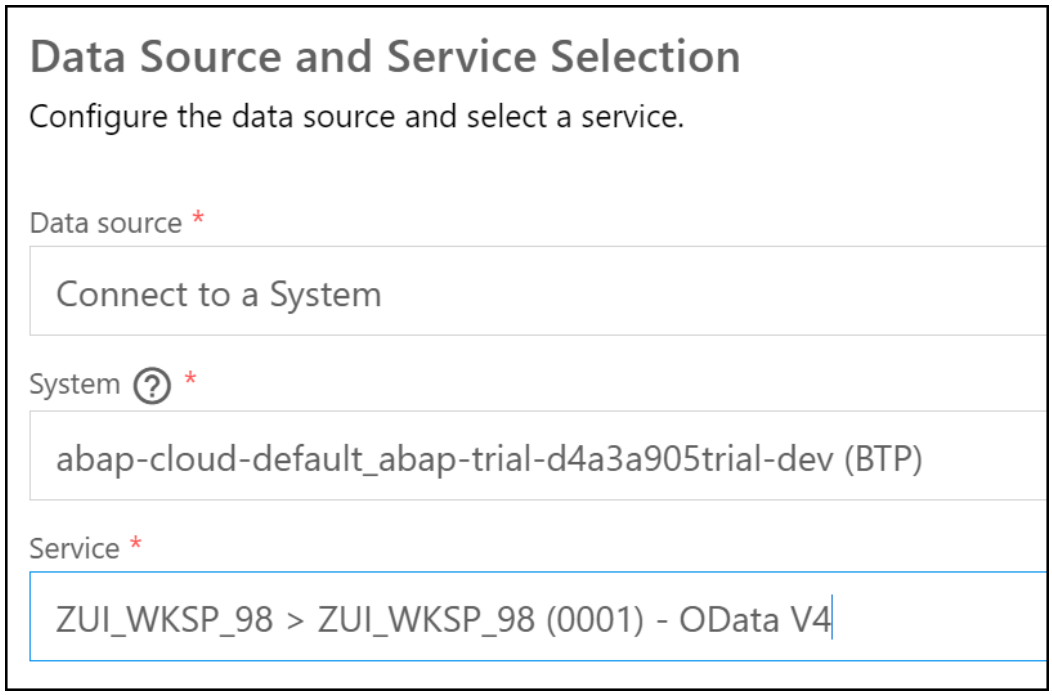
Task 3: Create a new Fiori elements application with the new CDS

Create a new Fiori Application using the Template Wizard for a *List Report*

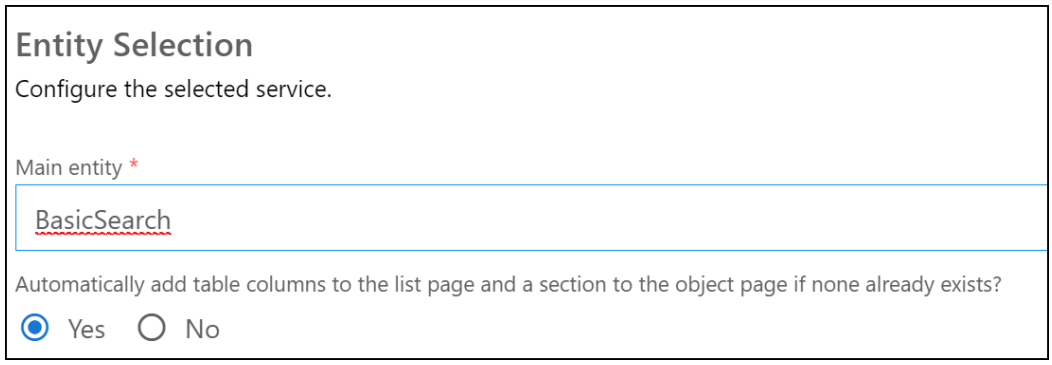
Field	**Value**
Data source	Connect to a System
System	abap-cloud-default_xx-dev (BTP)
Service	ZUI_WKSP_##
Main entity	BasicSearch
Module name	basic-search
Application title	Basic List with Search
Application namespace	nato.workshop
Description	Basic List with Search
Project folder path	/home/user/projects
Add deployment configuration	Yes

Field	Value
Add FLP configuration	Yes
Deployment Target	Cloud Foundry
Destination name	abap-cloud-default_xx(SCP)
Add application to managed application router	Yes
Semantic Object	NATO
Action	BasicSearch
Title	Basic List with Search
Subtitle	Workshop

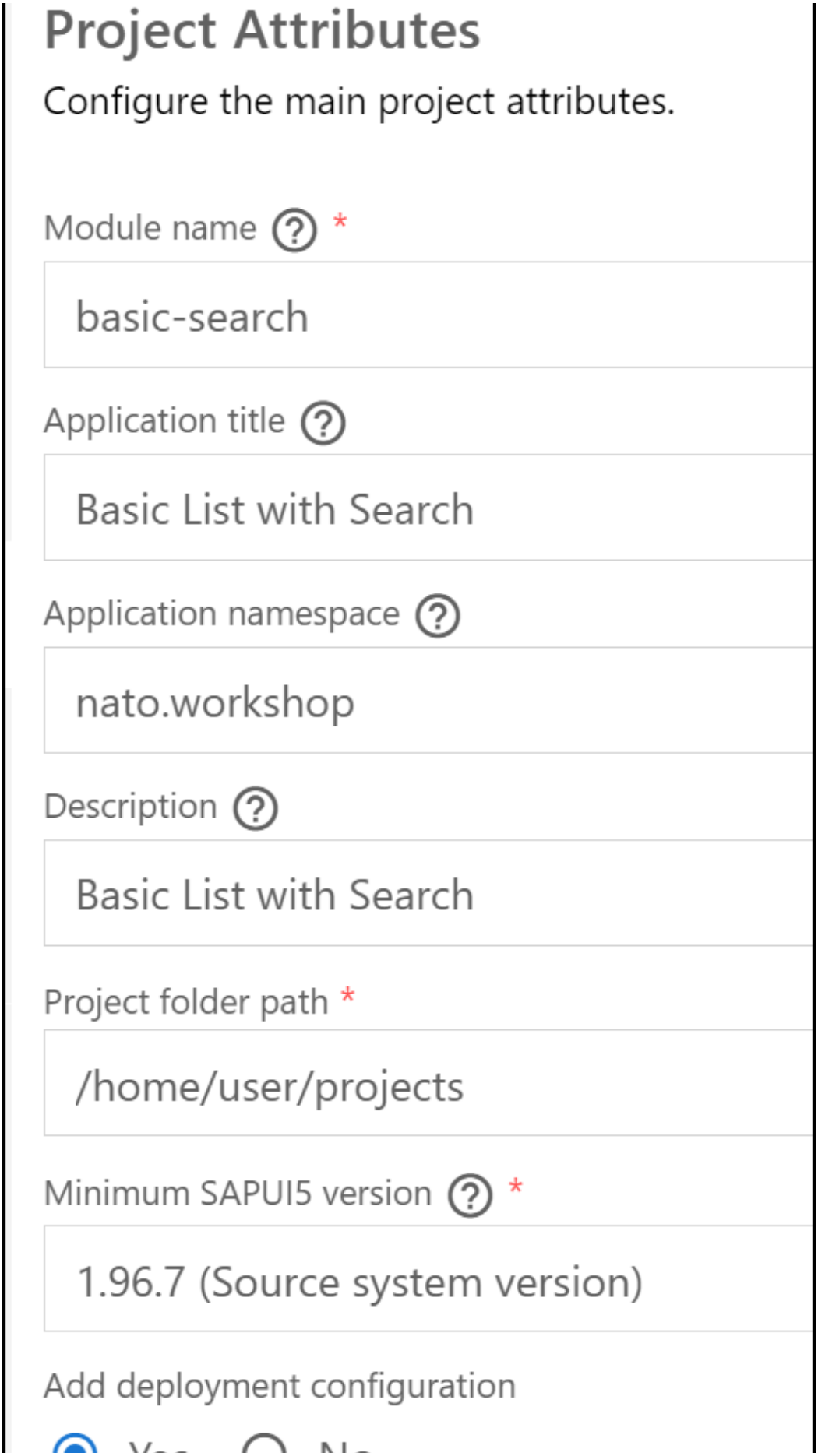
- Data Source and Service Selection

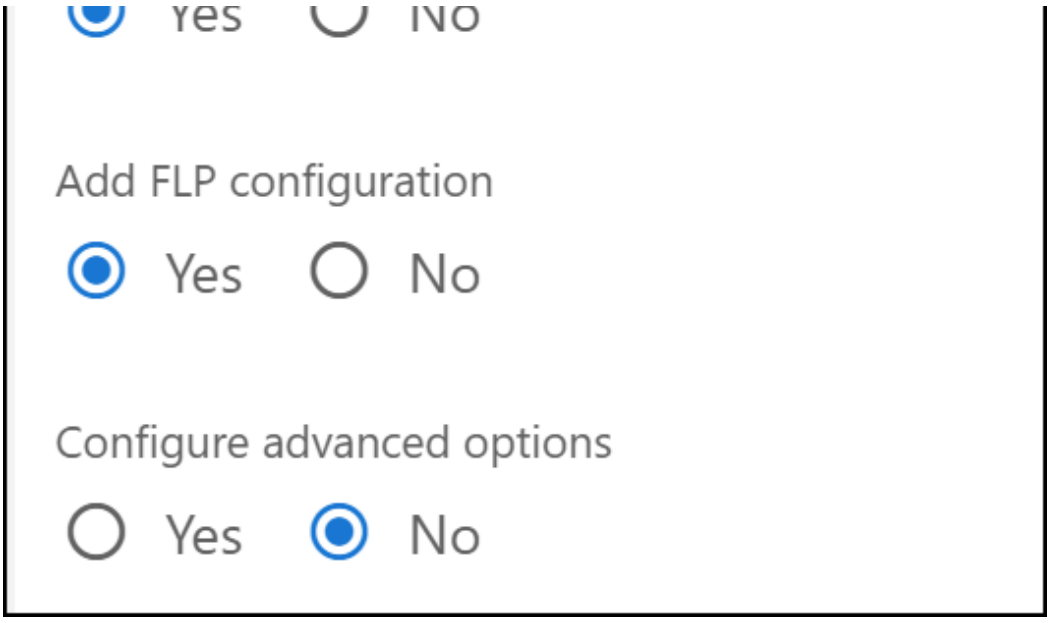


- Entity Selection

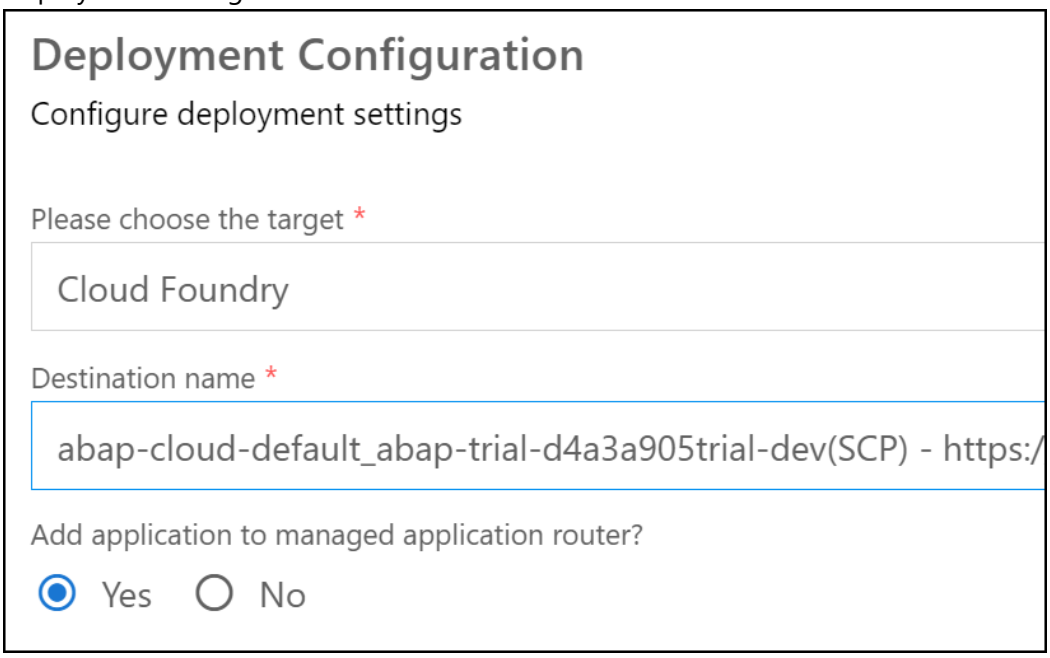


- Project Attributes

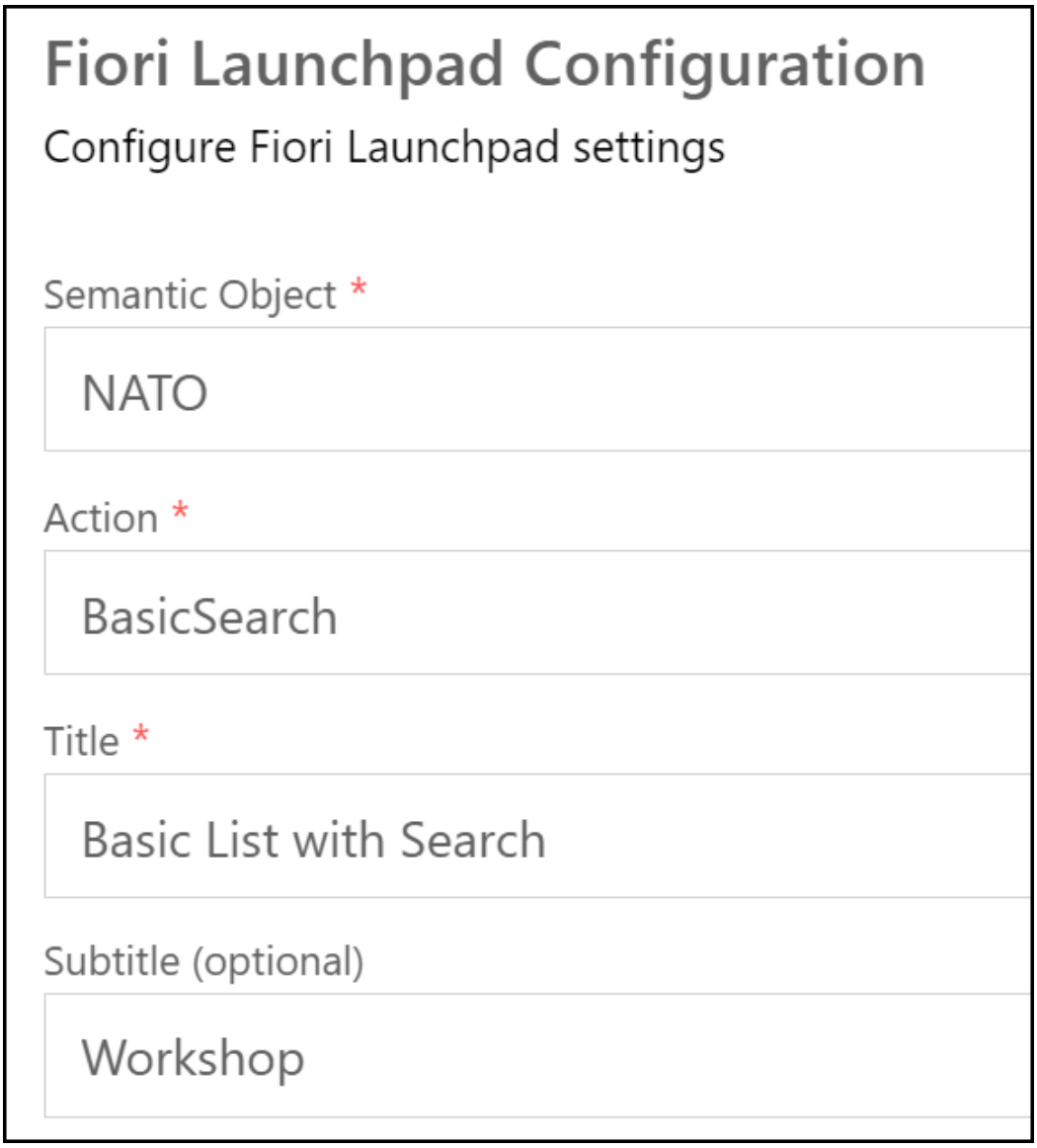

Project Attributes
Configure the main project attributes.
Module name *
basic-search
Application title
Basic List with Search
Application namespace
nato.workshop
Description
Basic List with Search
Project folder path *
/home/user/projects
Minimum SAPUI5 version *
1.96.7 (Source system version)
Add deployment configuration



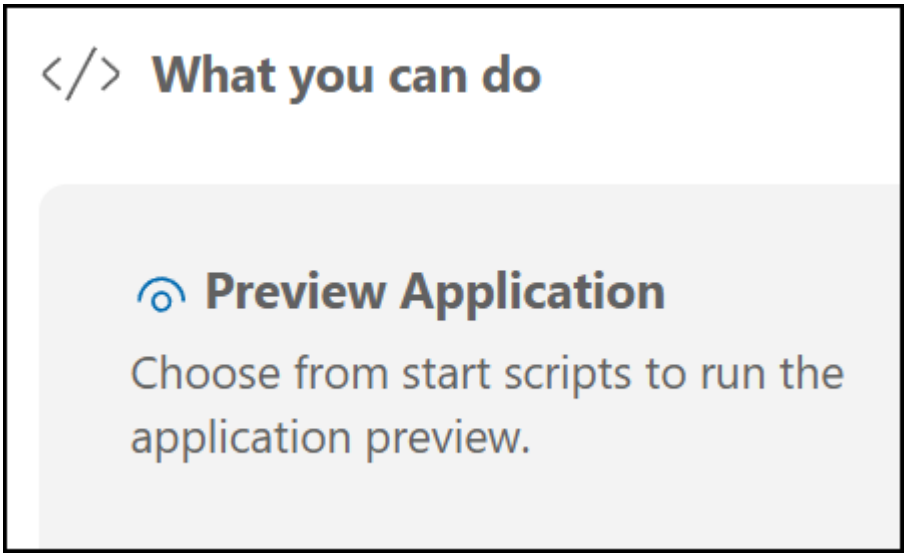
- Deployment Configuration



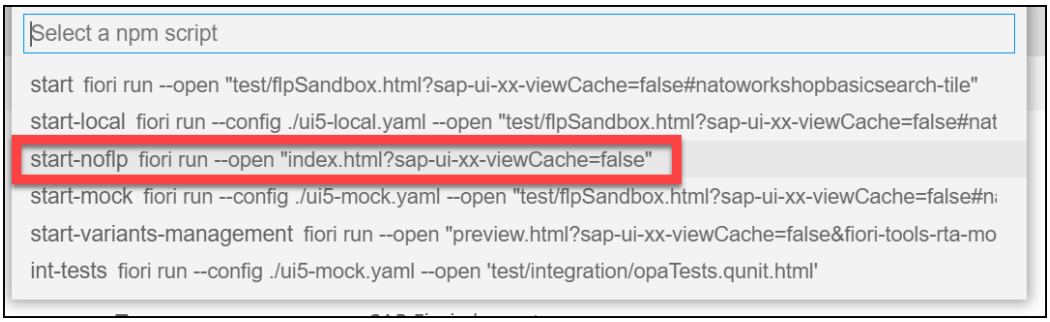
- Fiori Launchpad Configuration



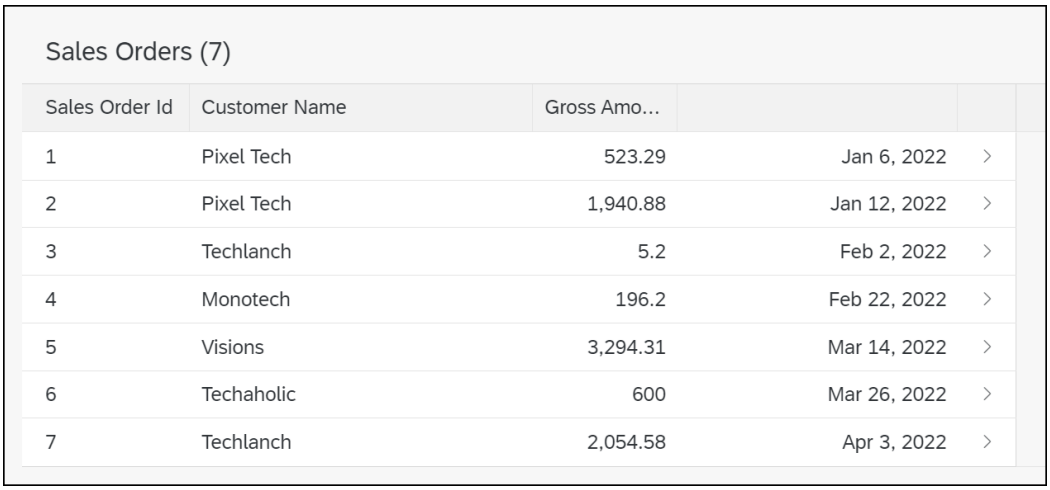
- Preview Application



- Select start-noflp, this is the fastest test.



- Result



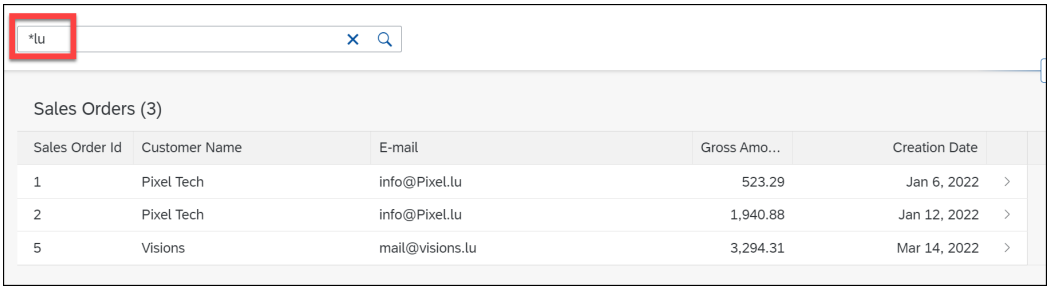
Task 4: Add Search to CDS view

- Add a *@Search.searchable: true* annotation to your CDS view before the `define view...`.
- Set fields *CustomerName* and *CustomerEmail* as default search elements.
- Set Fuzzy search level at 60%.
- Activate your CDS and Reload your fiori application to see the results.

```
@AbapCatalog.sqlViewName: 'ZWK##CDSLISTSRCH'
@AbapCatalog.compiler.compareFilter: true
@AbapCatalog.preserveKey: true
@AccessControl.authorizationCheck: #CHECK
@EndUserText.label: 'List Report'
@UI.headerInfo.typeName: 'Sales Order'
@UI.headerInfo.typeNamePlural: 'Sales Orders'
@Search.searchable: true
define view ZWS##_CDS_LIST_SEARCH
  as select from ztmcds9_i_so
{
      @UI.lineItem: [{position: 10 }]
  key Id
as SalesOrderID,
```

```
      CustomerId
as CustomerID,
      @UI.lineItem: [{position: 20 },{ label: 'Customer Name' }]
      @Search.defaultSearchElement: true
      _Customer.name
as CustomerName,
      @UI.lineItem: [{position: 30 },{ label: 'E-mail' }]
      @Search.defaultSearchElement: true
      @Search.fuzzinessThreshold : 0.6
      _Customer.email
as CustomerEmail,
      @UI.lineItem: [{position: 40 }]
      GrossAmount,
      @UI.lineItem: [{position: 50},{ label: 'Creation Date' }]
      cast( left(cast(CreationDateTime as abap.char( 30)),8) as
abap.dats) as CreationDate
}
```

- Result



- Play a bit with the fuzzy search levels.

Task 4: Add Selection fields for Filtering

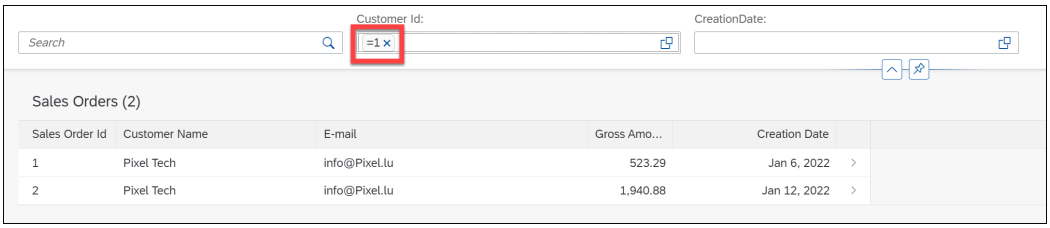
- Add filters for fields *CustomerId* and *CreationDate*.

```
@AbapCatalog.sqlViewName: 'ZWK##CDSLISTSRCH'
@AbapCatalog.compiler.compareFilter: true
@AbapCatalog.preserveKey: true
@AccessControl.authorizationCheck: #CHECK
@EndUserText.label: 'List Report'
@UI.headerInfo.typeName: 'Sales Order'
@UI.headerInfo.typeNamePlural: 'Sales Orders'
@Search.searchable: true
define view ZWS##_CDS_LIST_SEARCH
  as select from ztmcds9_i_so
{
      @UI.lineItem: [{position: 10 }]
  key Id
```

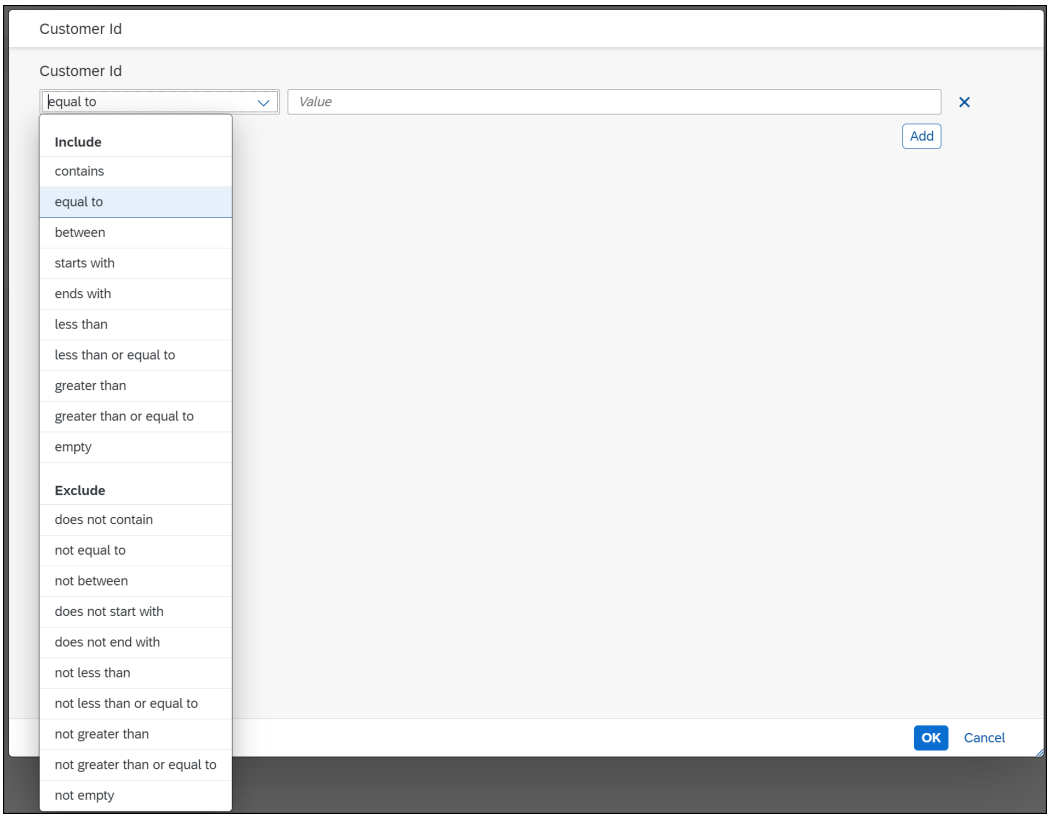
```
as SalesOrderID,
      @UI.selectionField: [{position: 10}]
      CustomerId
as CustomerID,
      @UI.lineItem: [{position: 20 },{ label: 'Customer Name' }]
      @Search.defaultSearchElement: true
      @Search.fuzzinessThreshold : 0.6
      _Customer.name
as CustomerName,
      @UI.lineItem: [{position: 30 },{ label: 'E-mail' }]
      @Search.defaultSearchElement: true
      _Customer.email
as CustomerEmail,
      @UI.lineItem: [{position: 40 }]
      GrossAmount,
      @UI.lineItem: [{position: 50},{ label: 'Creation Date' }]
      @UI.selectionField: [{position: 20}]
      cast( left(cast(CreationDateTime as abap.char( 30)),8) as
abap.dats) as CreationDate
}
```

- Result

- With active filter



- Basic Filter Screen



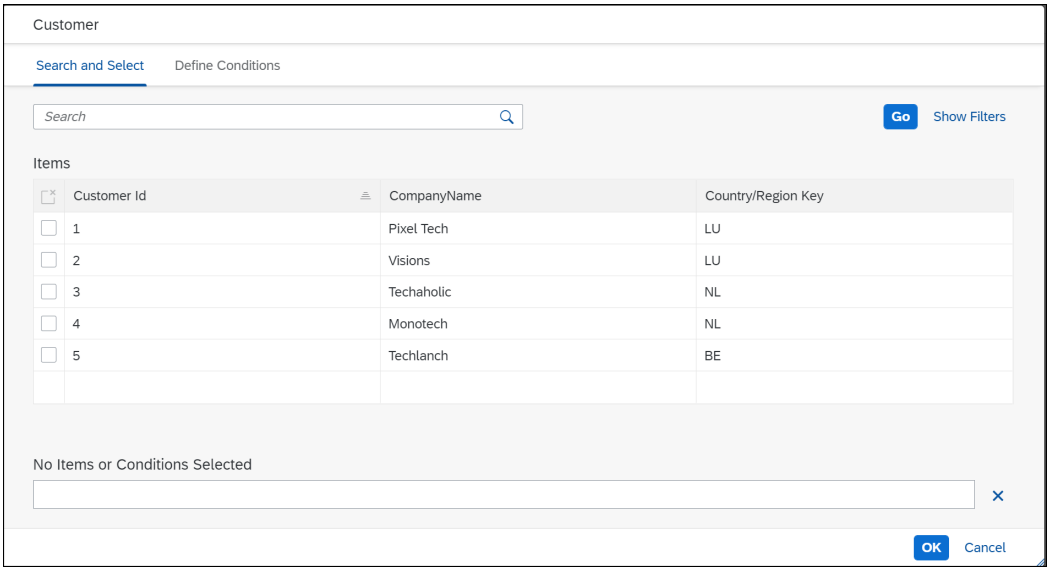
Task 5: Add Value Helps to Filters

- Have a look at CDS **ZTMCDS9_C_VH_CUST** in package **Z_TMC**.
- Use this CDS as Value Help for the *CustomerId*.
- Add *@Consumption.filter.selectionType: #INTERVAL* to the *CreationDate* filter and see the result.

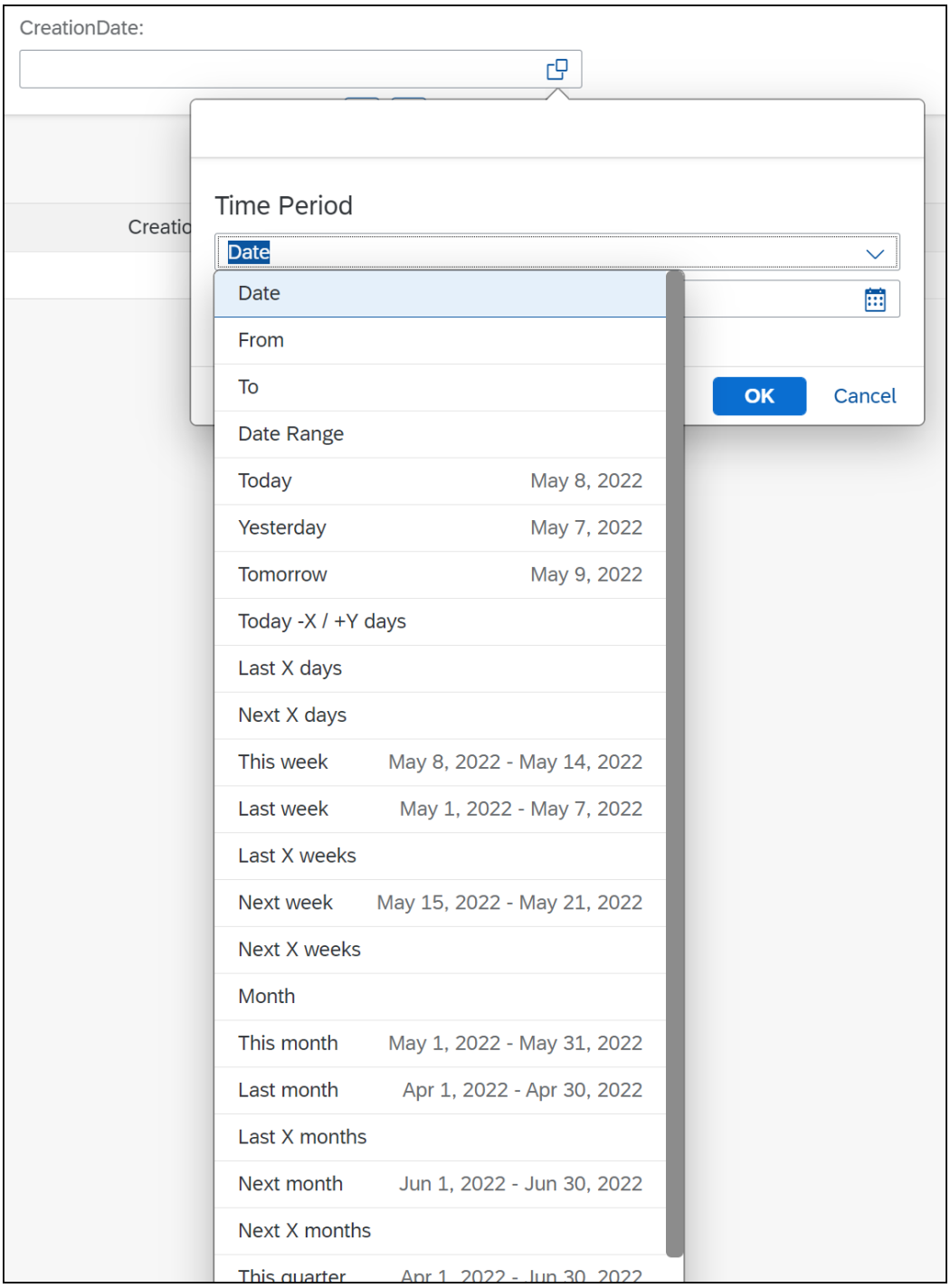
```
@AbapCatalog.sqlViewName: 'ZWK##CDSLISTSRCH'
@AbapCatalog.compiler.compareFilter: true
@AbapCatalog.preserveKey: true
@AccessControl.authorizationCheck: #CHECK
@EndUserText.label: 'List Report'
@UI.headerInfo.typeName: 'Sales Order'
@UI.headerInfo.typeNamePlural: 'Sales Orders'
@Search.searchable: true
define view ZWS##_CDS_LIST_SEARCH
  as select from ztmcds9_i_so
  association [1] to ztmcds9_c_vh_cust as _CustomerVH
    on $projection.CustomerID = _CustomerVH.SoldToParty
{
      @UI.lineItem: [{position: 10 }]
  key Id
as SalesOrderID,
```

```
      @UI.selectionField: [{position: 10}]
      @Consumption.valueHelpDefinition: [{ entity:{ name:
'ZTMCDS9_C_VH_CUST', element: 'SoldToParty' }}]
      CustomerId
as CustomerID,
      @UI.lineItem: [{position: 20 },{ label: 'Customer Name' }]
      @Search.defaultSearchElement: true
      @Search.fuzzinessThreshold : 0.6
      _Customer.name
as CustomerName,
      @UI.lineItem: [{position: 30 },{ label: 'E-mail' }]
      @Search.defaultSearchElement: true
      _Customer.email
as CustomerEmail,
      @UI.lineItem: [{position: 40 }]
      GrossAmount,
      @UI.lineItem: [{position: 50},{ label: 'Creation Date' }]
      @UI.selectionField: [{position: 20}]
      @Consumption.filter.selectionType: #INTERVAL
      cast( left(cast(CreationDateTime as abap.char( 30)),8) as
abap.dats) as CreationDate,
      _CustomerVH
}
```

- Filter for *CustomerId*



- Filter for *CreationDate*



Create List Report Object Page

Task 1: Duplicate CDS **ZWS##_CDS_LIST_SEARCH** to **ZWS##_CDS_LIST_OBJECT**

Task 2: Add the new CDS to your *Service Definition*

```
@EndUserText.label: 'UI ## Service Definition'
define service ZUI_WKSP_## {
  expose ZWS##_CDS_Simple as SimpleCDS;
  expose ZWS##_CDS_Basic as BasicCDS;
  expose ZWS##_CDS_LIST as BasicList;
  expose ZWS##_CDS_LIST_SEARCH as BasicSearch;
  expose ZWS##_CDS_LIST_OBJECT as ListObject;
}
```

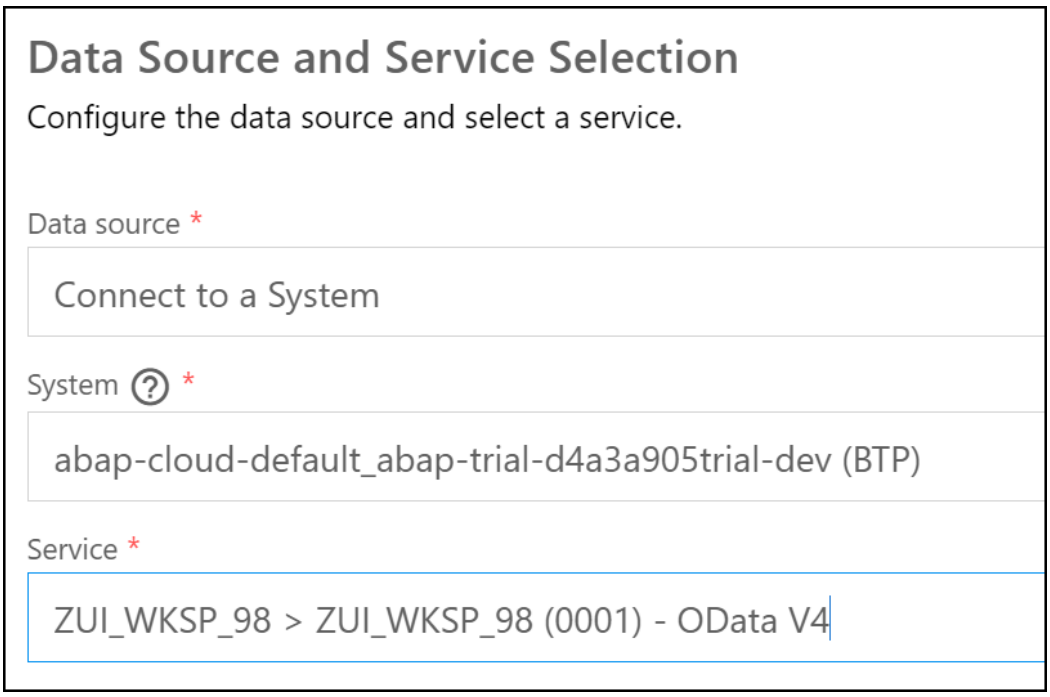
Task 3: Create a new Fiori elements application with the new CDS

Create a new Fiori Application using the Template Wizard for a *List Report*

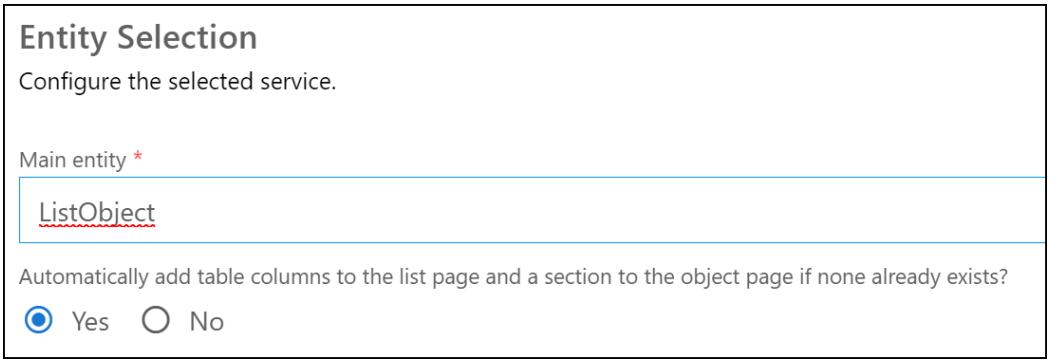
Field	**Value**
Data source	Connect to a System
System	abap-cloud-default_xx-dev (BTP)
Service	ZUI_WKSP_##
Main entity	ListObject
Module name	basic-object
Application title	Basic Object
Application namespace	nato.workshop
Description	Basic Object
Project folder path	/home/user/projects
Add deployment configuration	Yes
Add FLP configuration	Yes
Deployment Target	Cloud Foundry

Field	Value
Destination name	abap-cloud-default_xx(SCP)
Add application to managed application router	Yes
Semantic Object	NATO
Action	BasicObject
Title	Basic List Object
Subtitle	Workshop

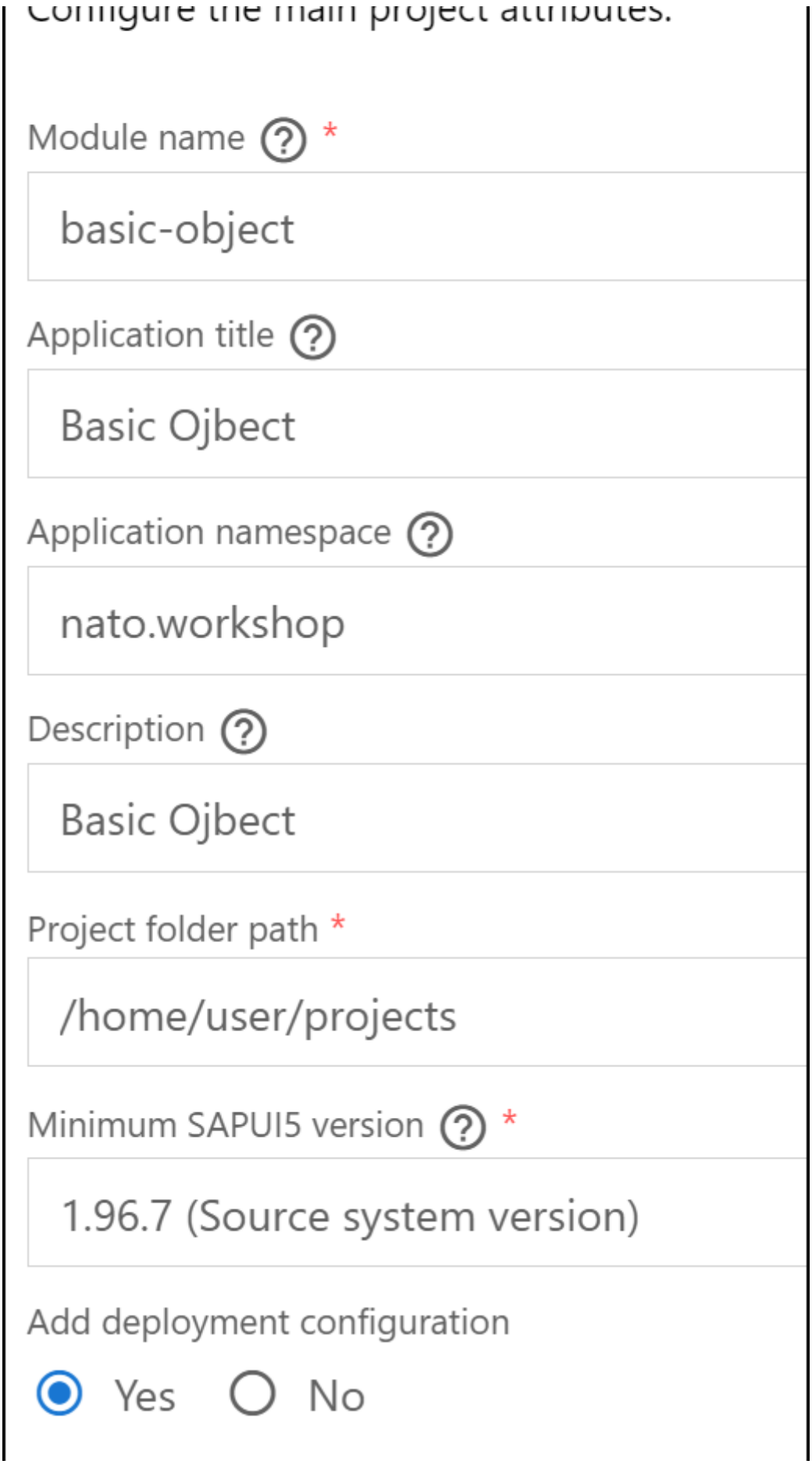
- Data Source and Service Selection

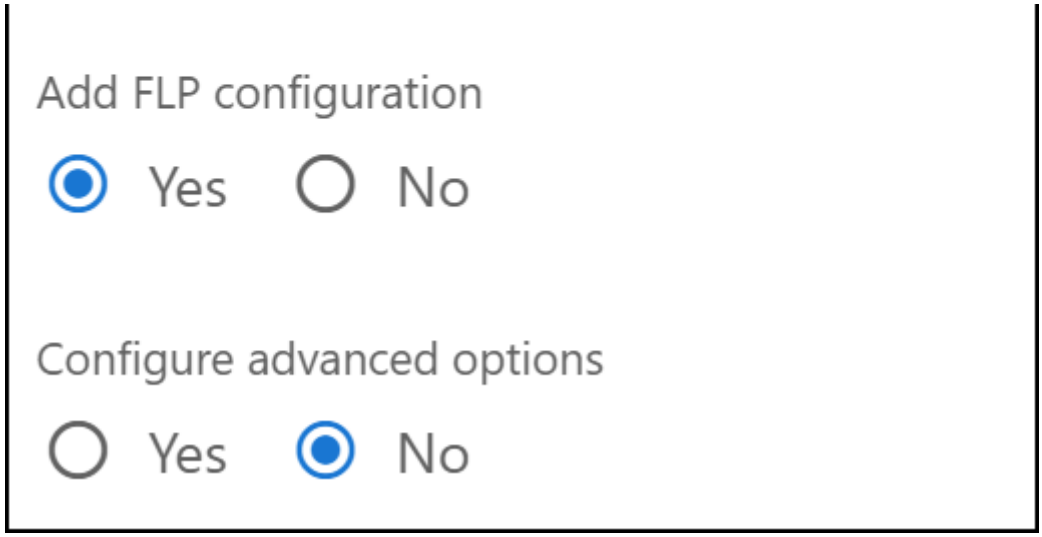


- Entity Selection

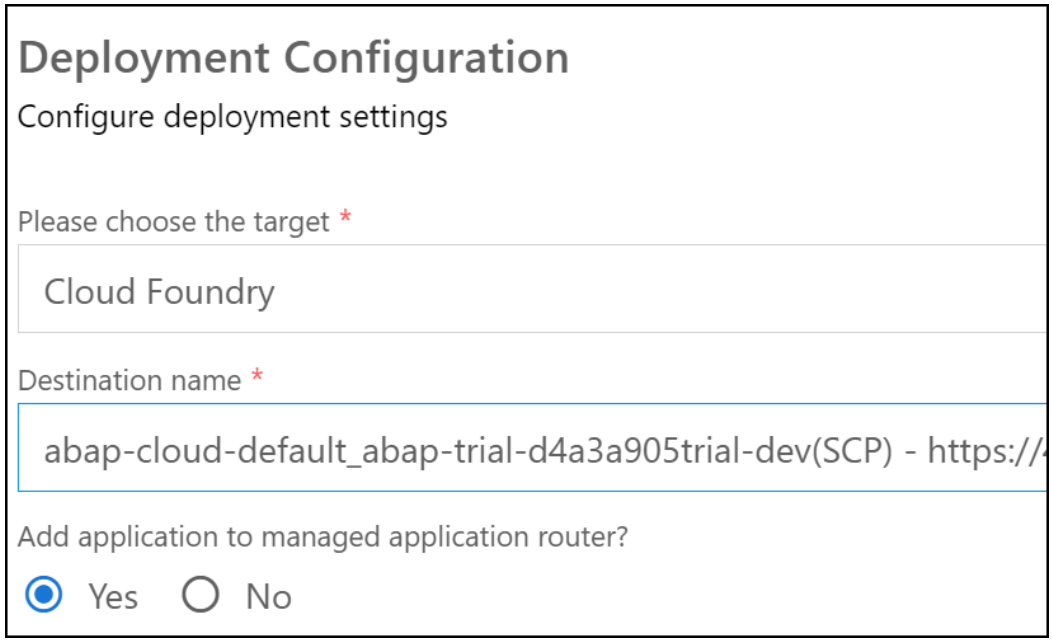


- Project Attributes

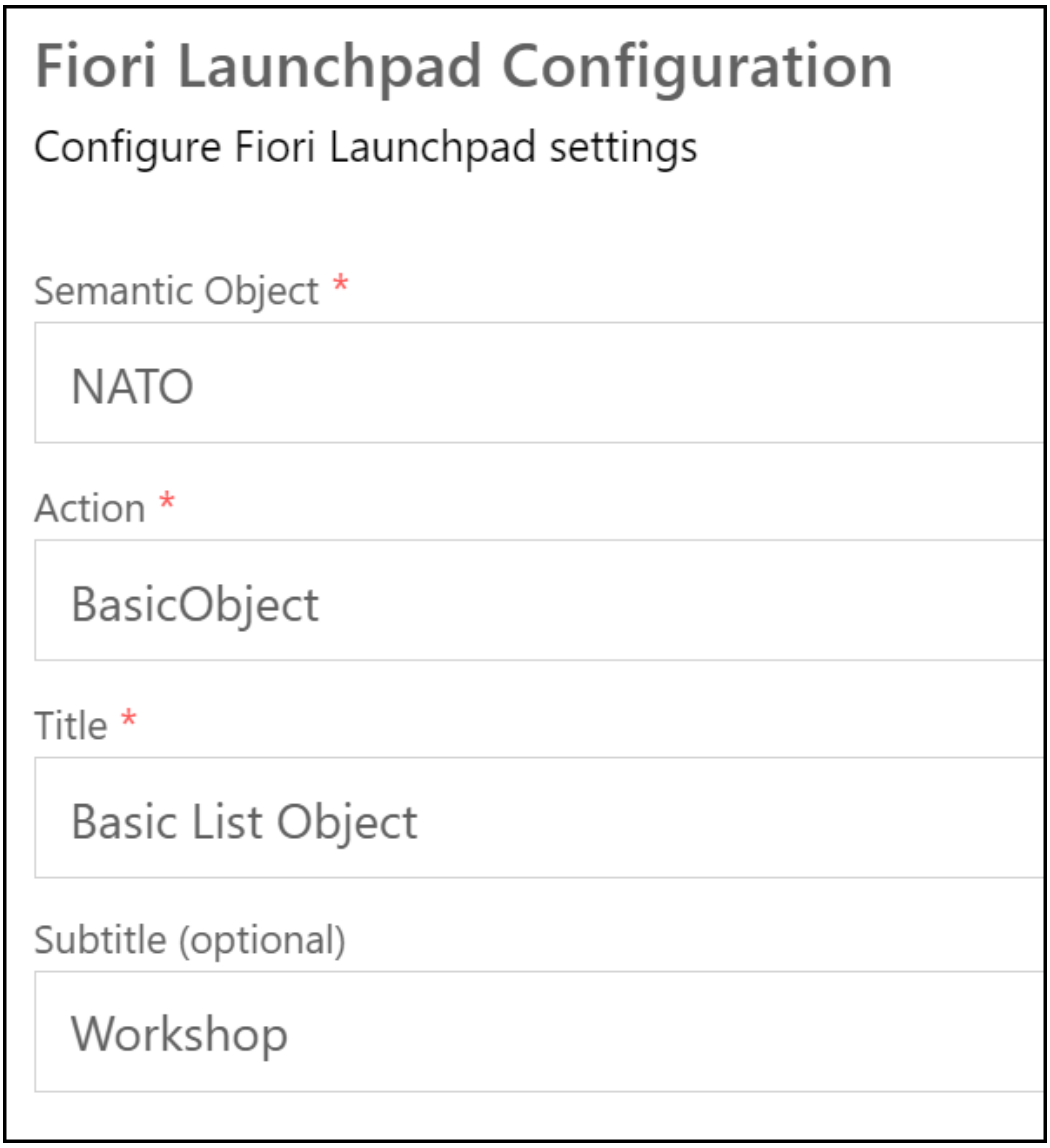

Configure the main project attributes.
Module name *
basic-object
Application title
Basic Ojbect
Application namespace
nato.workshop
Description
Basic Ojbect
Project folder path *
/home/user/projects
Minimum SAPUI5 version *
1.96.7 (Source system version)
Add deployment configuration
Yes
No



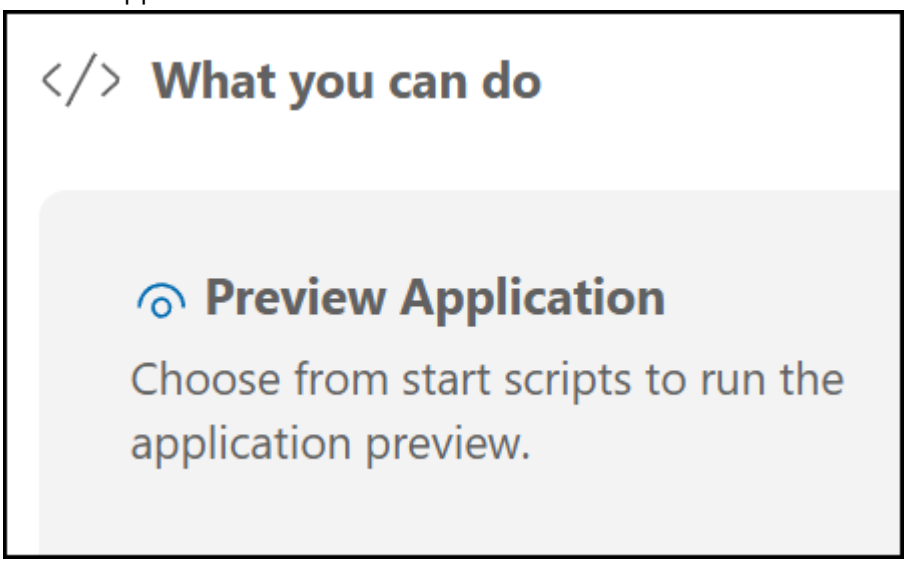
- Deployment Configuration



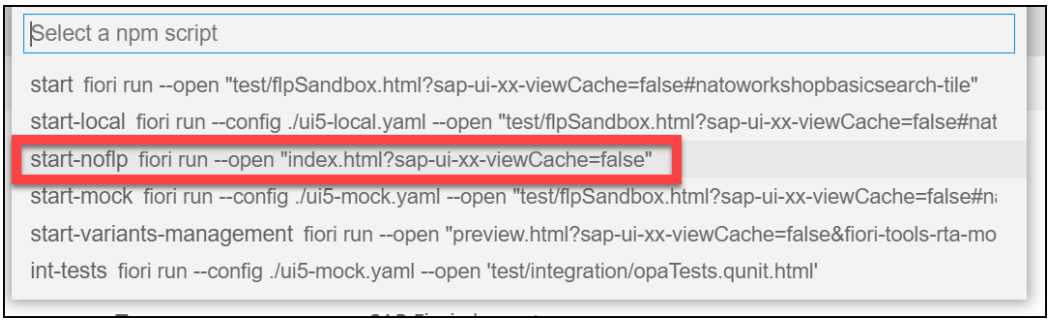
- Fiori Launchpad Configuration



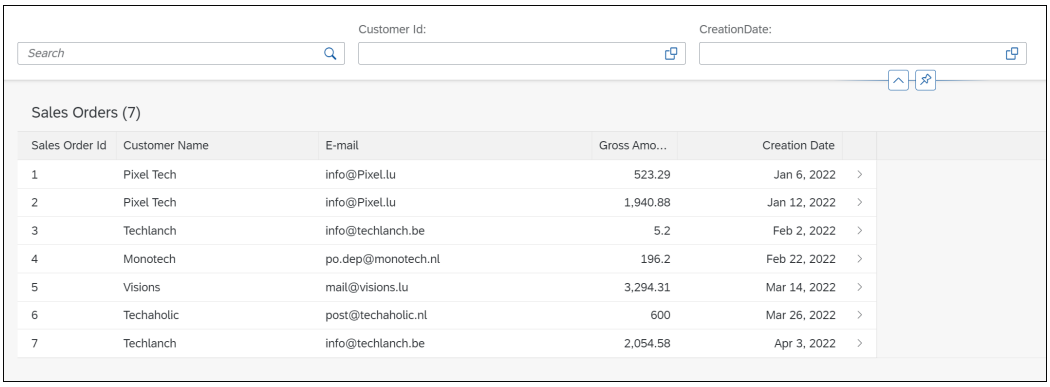
- Preview Application



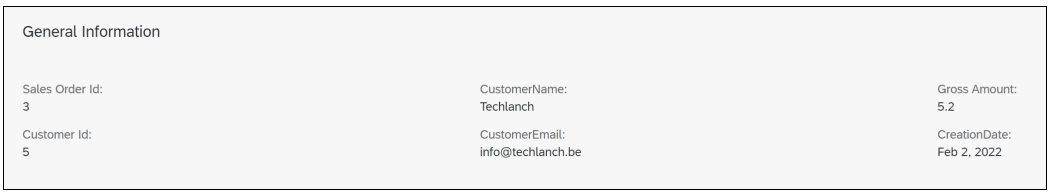
- Select start-noflp, this is the fastest test.



- Result



- Select a line



Task 4: Basic Object Page

- Set the title of every single sales order to the value of field **SalesOrderID**.
- Use *@UI.identification* and *@UI.facet* to add more fields to the object page

Position	Value
10	SalesOrderID
20	CustomerID
30	CustomerName
40	CustomerEmail

```
@AbapCatalog.sqlViewName: 'ZWK##CDSLISTOBJT'
@AbapCatalog.compiler.compareFilter: true
@AbapCatalog.preserveKey: true
@AccessControl.authorizationCheck: #CHECK
```

```
@EndUserText.label: 'Object Page'
@UI.headerInfo.typeName: 'Sales Order'
@UI.headerInfo.typeNamePlural: 'Sales Orders'
@UI.headerInfo.title.value: 'SalesOrderID'
@Search.searchable: true
define view ZWS##_CDS_LIST_OBJECT
  as select from ztmcds9_i_so
  association [1] to ztmcds9_c_vh_cust as _CustomerVH on
$projection.CustomerID = _CustomerVH.SoldToParty
{

      @UI.facet: [
       {
         id: 'COLLFAC1',
         label: 'General Information',
         type: #COLLECTION,
         position: 10
       },
       {
         label: 'General Information',
         type: #IDENTIFICATION_REFERENCE,
         parentId: 'COLLFAC1',
         position: 10
       }
      ]

      @UI.lineItem: [{position: 10 }]
      @UI.identification: [{position: 10}]
  key Id
as SalesOrderID,
      @UI.selectionField: [{position: 10}]
      @UI.identification: [{position: 20}]
      @Consumption.valueHelpDefinition: [{ entity:{ name:
'ZTMCDS9_C_VH_CUST', element: 'SoldToParty' }}]
      CustomerId
as CustomerID,
      @UI.lineItem: [{position: 20 },{ label: 'Customer Name' }]
      @UI.identification: [{position: 30}]
      @Search.defaultSearchElement: true
      @Search.fuzzinessThreshold : 0.6
      _Customer.name
as CustomerName,
      @UI.lineItem: [{position: 30 },{ label: 'E-mail' }]
      @UI.identification: [{position: 40}]
      @Search.defaultSearchElement: true
      _Customer.email
as CustomerEmail,
      @UI.lineItem: [{position: 40 }]
```

```
      GrossAmount,
      NetAmount,
      TaxAmount,
      DeliveryStatus,
      BillingStatus,
      @UI.lineItem: [{position: 50},{ label: 'Creation Date' }]
      @Consumption.filter.selectionType: #INTERVAL
      cast( left(cast(CreationDateTime as abap.char( 30)),8) as
abap.dats) as CreationDate,
      CreationUser,
      @UI.hidden: true
      CurrencyCode,
      _CustomerVH
}
```

- Result

Task 5: Add Header Facets

- Create two header facets: *AMOUNT* and *STATUS*

Group	Position	Field
AMOUNT	10	GrossAmount
AMOUNT	20	NetAmount
AMOUNT	30	TaxAmound
STATUS	10	DeliveryStatus
STATUS	20	BillingStatus

```
@AbapCatalog.sqlViewName: 'ZWK##CDSLISTOBJT'
@AbapCatalog.compiler.compareFilter: true
@AbapCatalog.preserveKey: true
@AccessControl.authorizationCheck: #CHECK
@EndUserText.label: 'Object Page'
@UI.headerInfo.typeName: 'Sales Order'
@UI.headerInfo.typeNamePlural: 'Sales Orders'
@UI.headerInfo.title.value: 'SalesOrderID'
```

```
@Search.searchable: true
define view ZWS##_CDS_LIST_OBJECT
  as select from ztmcds9_i_so
  association [1] to ztmcds9_c_vh_cust as _CustomerVH on
$projection.CustomerID = _CustomerVH.SoldToParty
{

      @UI.facet: [
      {
      id: 'COLLFAC1',
      label: 'General Information',
      type: #COLLECTION,
      position: 10
      },
      {
      label: 'General Information',
      type: #IDENTIFICATION_REFERENCE,
      parentId: 'COLLFAC1',
      position: 10
      },
      {
      label: 'Amount Information',
      purpose: #HEADER,
      type: #FIELDGROUP_REFERENCE,
      targetQualifier: 'AMOUNT',
      position: 10
      },
      {
      label: 'Status Information',
      purpose: #HEADER,
      type: #FIELDGROUP_REFERENCE,
      targetQualifier: 'STATUS',
      position: 10
      }
      ]

      @UI.lineItem: [{position: 10 }]
      @UI.identification: [{position: 10}]
  key Id
as SalesOrderID,
      @UI.selectionField: [{position: 10}]
      @UI.identification: [{position: 20}]
      @Consumption.valueHelpDefinition: [{ entity:{ name:
'ZTMCDS9_C_VH_CUST', element: 'SoldToParty' }}]
      CustomerId
as CustomerID,
      @UI.lineItem: [{position: 20 },{ label: 'Customer Name' }]
      @UI.identification: [{position: 30}]
```

```
      @Search.defaultSearchElement: true
      @Search.fuzzinessThreshold : 0.6
      _Customer.name
as CustomerName,
      @UI.lineItem: [{position: 30 },{ label: 'E-mail' }]
      @UI.identification: [{position: 40}]
      @Search.defaultSearchElement: true
      _Customer.email
as CustomerEmail,
      @UI.lineItem: [{position: 40 }]
      @UI.fieldGroup: [{position: 10, qualifier: 'AMOUNT'}]
      GrossAmount,
      @UI.fieldGroup: [{position: 20, qualifier: 'AMOUNT'}]
      NetAmount,
      @UI.fieldGroup: [{position: 30, qualifier: 'AMOUNT'}]
      TaxAmount,
      @UI.fieldGroup: [{position: 10, qualifier: 'STATUS'}]
      DeliveryStatus,
      @UI.fieldGroup: [{position: 20, qualifier: 'STATUS'}]
      BillingStatus,
      @UI.lineItem: [{position: 50},{ label: 'Creation Date' }]
      @Consumption.filter.selectionType: #INTERVAL
      cast( left(cast(CreationDateTime as abap.char( 30)),8) as
abap.dats) as CreationDate,
      CreationUser,
      @UI.hidden: true
      CurrencyCode,
      _CustomerVH
}
```

- Result

Task 6: Add Sections

- Create a collection facet for *General information*.
- Create a collection facet for *More information*.
- Assign qualifiers **AMOUNT** and **STATUS** to collection *More information*.

```
@AbapCatalog.sqlViewName: 'ZWK##CDSLISTOBJT'
@AbapCatalog.compiler.compareFilter: true
@AbapCatalog.preserveKey: true
@AccessControl.authorizationCheck: #CHECK
@EndUserText.label: 'Object Page'
@UI.headerInfo.typeName: 'Sales Order'
@UI.headerInfo.typeNamePlural: 'Sales Orders'
@UI.headerInfo.title.value: 'SalesOrderID'
@Search.searchable: true
define view ZWS##_CDS_LIST_OBJECT
  as select from ztmcds9_i_so
  association [1] to ztmcds9_c_vh_cust as _CustomerVH on
$projection.CustomerID = _CustomerVH.SoldToParty
{

  @UI.facet: [
   {
     id: 'COLLFAC1',
     label: 'General Information',
     type: #COLLECTION,
     position: 10
   },
   {
     id: 'COLLFAC2',
     label: 'More Information',
     type: #COLLECTION,
     position: 20
   },
   {
     label: 'General Information',
     type: #IDENTIFICATION_REFERENCE,
     parentId: 'COLLFAC1',
     position: 10
   },
   {
     label: 'Amount Information',
     parentId: 'COLLFAC2',
     type: #FIELDGROUP_REFERENCE,
     targetQualifier: 'AMOUNT',
     position: 10
   },
   {
     label: 'Status Information',
     parentId: 'COLLFAC2',
     type: #FIELDGROUP_REFERENCE,
     targetQualifier: 'STATUS',
     position: 20
```

```
   }
  ]

      @UI.lineItem: [{position: 10 }]
      @UI.identification: [{position: 10}]
  key Id
as SalesOrderID,
      @UI.selectionField: [{position: 10}]
      @UI.identification: [{position: 20}]
      @Consumption.valueHelpDefinition: [{ entity:{ name:
'ZTMCDS9_C_VH_CUST', element: 'SoldToParty' }}]
      CustomerId
as CustomerID,
      @UI.lineItem: [{position: 20 },{ label: 'Customer Name' }]
      @UI.identification: [{position: 30}]
      @Search.defaultSearchElement: true
      @Search.fuzzinessThreshold : 0.6
      _Customer.name
as CustomerName,
      @UI.lineItem: [{position: 30 },{ label: 'E-mail' }]
      @Search.defaultSearchElement: true
      _Customer.email
as CustomerEmail,
      @UI.lineItem: [{position: 40 }]
      @UI.identification: [{position: 40}]
      @UI.fieldGroup: [{position: 10, qualifier: 'AMOUNT'}]
      GrossAmount,
      @UI.identification: [{position: 50}]
      @UI.fieldGroup: [{position: 20, qualifier: 'AMOUNT'}]
      NetAmount,
      @UI.identification: [{position: 60}]
      @UI.fieldGroup: [{position: 30, qualifier: 'AMOUNT'}]
      TaxAmount,
      @UI.fieldGroup: [{position: 10, qualifier: 'STATUS'}]
      DeliveryStatus,
      @UI.fieldGroup: [{position: 20, qualifier: 'STATUS'}]
      BillingStatus,
      @UI.lineItem: [{position: 50},{ label: 'Creation Date' }]
      @UI.selectionField: [{position: 20}]
      @Consumption.filter.selectionType: #INTERVAL
      cast( left(cast(CreationDateTime as abap.char( 30)),8) as
abap.dats) as CreationDate,
      CreationUser,
      @UI.hidden: true
      CurrencyCode,
      _CustomerVH
}
```

- Result Result

Task 7: Move Annotations to Metadata Extension (Optional)

- Duplicate **ZWS##_CDS_LIST_OBJECT** to **ZWS##_CDS_C_LO**
- Add annotation *@Metadata.allowExtensions: true*
- Create a *Metadata Extenstion* **ZWS##_ME_CDS_C_LO**
- Move all UI annotations to the *Metadata Extenstion*
- Add ZWS##_CDS_C_LO** to your *Service Definition*
- Create a new Fiori Application using the Template Wizard for a *List Report*

```
@AbapCatalog.sqlViewName: 'ZWK98CDSCLISTOBJ'
@AbapCatalog.compiler.compareFilter: true
@AbapCatalog.preserveKey: true
@AccessControl.authorizationCheck: #CHECK
@EndUserText.label: 'Object Page'

@Metadata.allowExtensions: true
define view ZWS98_CDS_C_LO
  as select from ztmcds9_i_so
  association [1] to ztmcds9_c_vh_cust as _CustomerVH on
$projection.CustomerID = _CustomerVH.SoldToParty
{

  key Id
as SalesOrderID,
      CustomerId
as CustomerID,
      _Customer.name
as CustomerName,
      _Customer.email
as CustomerEmail,
      GrossAmount,
      NetAmount,
      TaxAmount,
      DeliveryStatus,
      BillingStatus,
      cast( left(cast(CreationDateTime as abap.char( 30)),8) as
abap.dats) as CreationDate,
      CreationUser,
      CurrencyCode,
      _CustomerVH
}
```

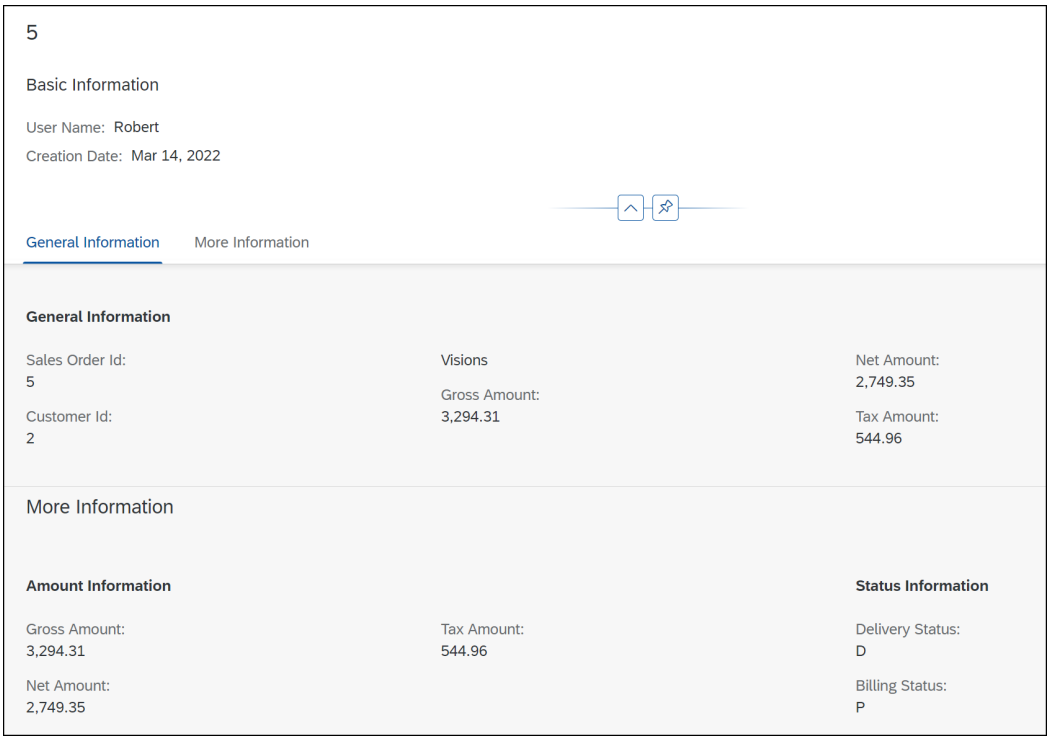
```
@Metadata.layer: #CUSTOMER
@UI.headerInfo.typeName: 'Sales Order'
@UI.headerInfo.typeNamePlural: 'Sales Orders'
@UI.headerInfo.title.value: 'SalesOrderID'
@Search.searchable: true
annotate view ZWS98_CDS_C_LO
    with
{
  @UI.facet: [
   {
     id: 'COLLFAC1',
     label: 'General Information',
     type: #COLLECTION,
     position: 10
   },
   {
     id: 'COLLFAC2',
     label: 'More Information',
     type: #COLLECTION,
     position: 20
   },
   {
      label: 'Basic Information',
      purpose: #HEADER,
      type: #FIELDGROUP_REFERENCE,
      targetQualifier: 'BASIC',
      position: 10
      },
   {
     label: 'General Information',
     type: #IDENTIFICATION_REFERENCE,
     parentId: 'COLLFAC1',
     position: 10
   },
   {
     label: 'Amount Information',
     parentId: 'COLLFAC2',
     type: #FIELDGROUP_REFERENCE,
     targetQualifier: 'AMOUNT',
     position: 10
   },
   {
     label: 'Status Information',
     parentId: 'COLLFAC2',
     type: #FIELDGROUP_REFERENCE,
     targetQualifier: 'STATUS',
     position: 20
   }
```

```
  ]

      @UI.lineItem: [{position: 10 }]
      @UI.identification: [{position: 10}]
      SalesOrderID;
      @UI.selectionField: [{position: 10}]
      @UI.identification: [{position: 20}]
      @Consumption.valueHelpDefinition: [{ entity:{ name:
'ZTMCDS9_C_VH_CUST', element: 'SoldToParty' }}]
      CustomerID;
      @UI.lineItem: [{position: 20 },{ label: 'Customer Name' }]
      @UI.identification: [{position: 30}]
      @Search.defaultSearchElement: true
      @Search.fuzzinessThreshold : 0.6
      CustomerName;
      @UI.lineItem: [{position: 30 },{ label: 'E-mail' }]
      @Search.defaultSearchElement: true
      CustomerEmail;
      @UI.lineItem: [{position: 40 }]
      @UI.identification: [{position: 40}]
      @UI.fieldGroup: [{position: 10, qualifier: 'AMOUNT'}]
      GrossAmount;
      @UI.identification: [{position: 50}]
      @UI.fieldGroup: [{position: 20, qualifier: 'AMOUNT'}]
      NetAmount;
      @UI.identification: [{position: 60}]
      @UI.fieldGroup: [{position: 30, qualifier: 'AMOUNT'}]
      TaxAmount;
      @UI.fieldGroup: [{position: 10, qualifier: 'STATUS'}]
      DeliveryStatus;
      @UI.fieldGroup: [{position: 20, qualifier: 'STATUS'}]
      BillingStatus;
      @UI.lineItem: [{position: 50},{ label: 'Creation Date' }]
      @UI.selectionField: [{position: 20}]
      @Consumption.filter.selectionType: #INTERVAL
      @UI.fieldGroup: [{position: 20, qualifier: 'BASIC', label:
'Creation Date'}]
      CreationDate;
      @UI.fieldGroup: [{position: 10, qualifier: 'BASIC'}]
      CreationUser;
      @UI.hidden: true
      CurrencyCode;

}
```

```
@EndUserText.label: 'UI ## Service Definition'
define service ZUI_WKSP_## {
  expose ZWS##_CDS_Simple as SimpleCDS;
  expose ZWS##_CDS_Basic as BasicCDS;
  expose ZWS##_CDS_LIST as BasicList;
  expose ZWS##_CDS_LIST_SEARCH as BasicSearch;
  expose ZWS##_CDS_LIST_OBJECT as ListObject;
  expose ZWS98_CDS_C_LO as ListObjectME;
}
```

Field	Value
Data source	Connect to a System
System	abap-cloud-default_xx-dev (BTP)
Service	ZUI_WKSP_##
Main entity	ListObjectME
Module name	basic-object
Application title	Basic List with Object Page using Metadata Extension
Application namespace	nato.workshop
Description	Basic List with Object Page using Metadata Extension
Semantic Object	NATO
Action	BasicObjectME
Title	Basic List Object (ME)
Subtitle	Workshop

- Result


5
Basic Information
User Name: Robert
Creation Date: Mar 14, 2022
General Information
More Information
General Information
Sales Order Id:
5
Customer Id:
2
Visions
Gross Amount:
3,294.31
Net Amount:
2,749.35
Tax Amount:
544.96
More Information
Amount Information
Gross Amount:
3,294.31
Net Amount:
2,749.35
Tax Amount:
544.96
Status Information
Delivery Status:
D
Billing Status:
P

External Navigation and Visualiztion

Task 1: New CDS **ZWS##_CDS_NAV_EXT**

- Create a new CDS **ZWS##_CDS_NAV_EXT** based on **ztmdt9_products** with an assosiation to **ztmdt9_suppliers**
- Add fields

Position	Field	Name
10	id	ProductId
20	name	
30	supplier_id	
40	_Supplier.name	SupplierName

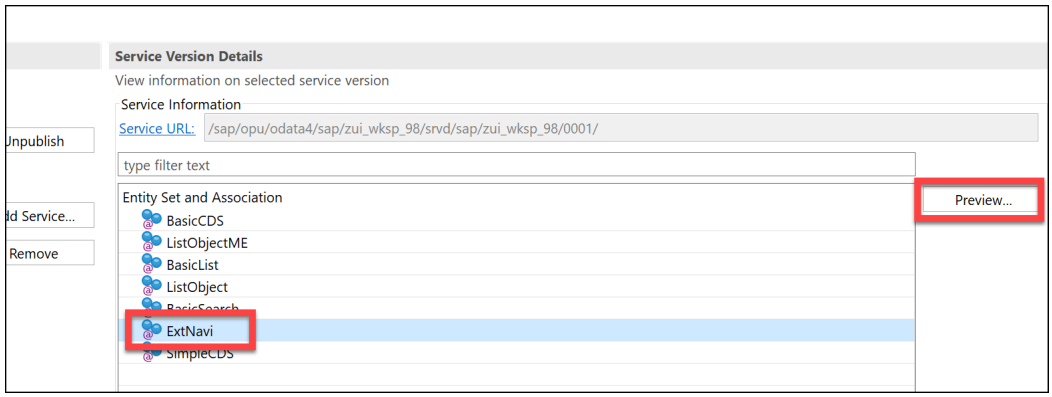
- Make the field SupplierName an external link to *https://www.google.com/search?q=**SupplierName***.
- Add the new CDS to your *Service Definition* and use the *Preview* option in your *Service Binding* to test your result.

```
@AbapCatalog.sqlViewName: 'ZWS##CDSNAVEXT'
@AbapCatalog.compiler.compareFilter: true
@AbapCatalog.preserveKey: true
@AccessControl.authorizationCheck: #CHECK
@EndUserText.label: 'Navigate to external link'
@UI.headerInfo.typeNamePlural: 'Products'
define view ZWS##_CDS_NAV_EXT
  as select from ztmdt9_products
  association [1] to ztmdt9_suppliers as _Supplier on _Supplier.id
= ztmdt9_products.supplier_id
{
      @UI.lineItem: [{ position: 10 }]
  key id
as ProductId,
      @UI.lineItem: [{ position: 20 }]
      name
as ProductDescription,
      @UI.lineItem: [{ position: 30 }]
      supplier_id
as SupplierId,
      @UI.lineItem: [{ position: 40, url: 'SearchURL', type:
#WITH_URL }]
```

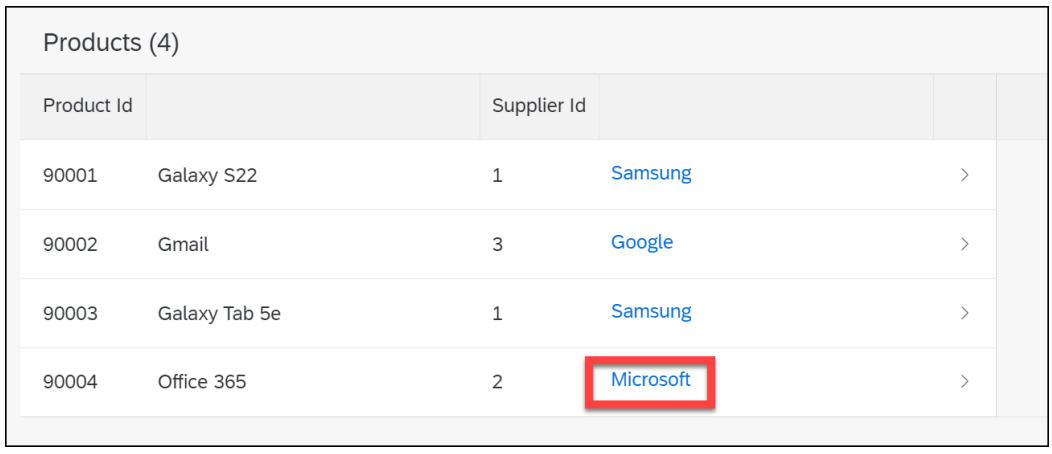
```
      _Supplier.name
as SupplierName,
      @UI.hidden: true
      concat('https://www.google.com/search?q=', _Supplier.name)
as SearchURL
}
```

```
@EndUserText.label: 'UI ## Service Definition'
define service ZUI_WKSP_## {
  expose ZWS##_CDS_Simple as SimpleCDS;
  expose ZWS##_CDS_Basic as BasicCDS;
  expose ZWS##_CDS_LIST as BasicList;
  expose ZWS##_CDS_LIST_SEARCH as BasicSearch;
  expose ZWS##_CDS_LIST_OBJECT as ListObject;
  expose ZWS##_CDS_C_LO as ListObjectME;
  expose ZWS##_CDS_NAV_EXT as ExtNavi;
}
```

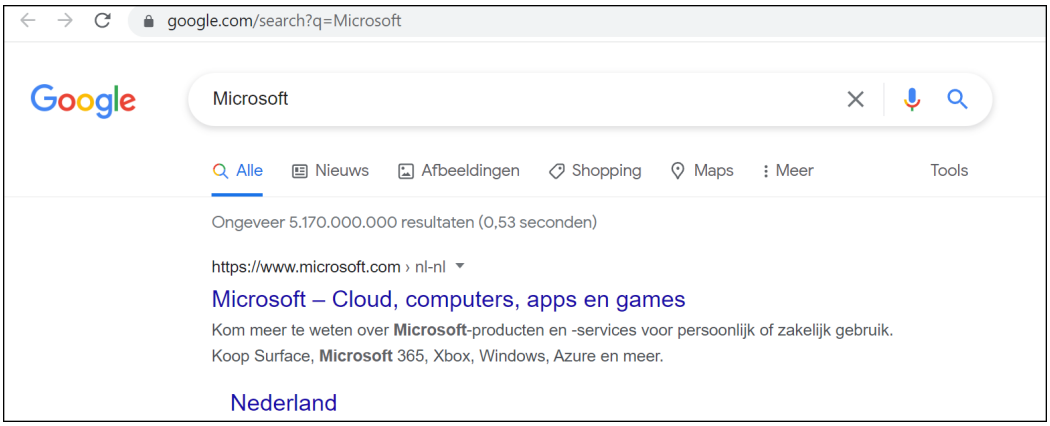
- Preview



- Result



- Navigation



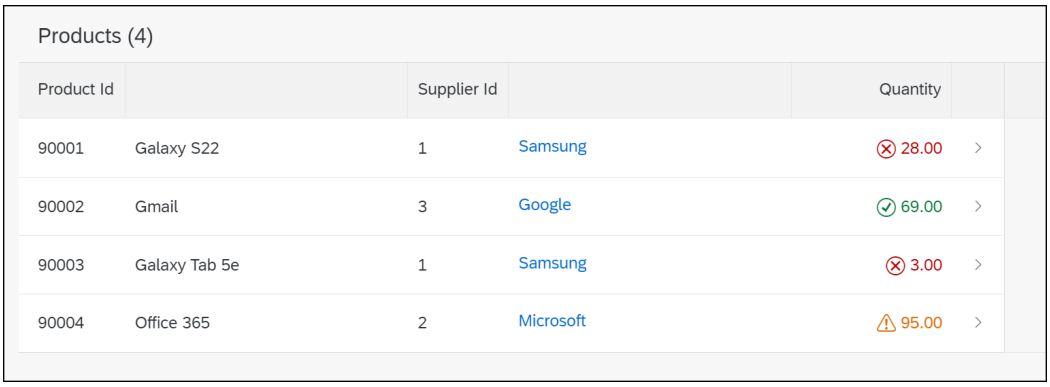
Task 2: Add visualization

- Add the fields *uom* and *stock* to the view
- When the stock is < 30 then color it Red
- When the stock is > 80 then color it Orange
- Else then color green

```
@AbapCatalog.sqlViewName: 'ZWS##CDSNAVEXT'
@AbapCatalog.compiler.compareFilter: true
@AbapCatalog.preserveKey: true
@AccessControl.authorizationCheck: #CHECK
@EndUserText.label: 'Navigate to external link'
@UI.headerInfo.typeNamePlural: 'Products'
define view ZWS##_CDS_NAV_EXT
  as select from ztmdt9_products
  association [1] to ztmdt9_suppliers as _Supplier on _Supplier.id
= ztmdt9_products.supplier_id
{
      @UI.lineItem: [{ position: 10 }]
  key id
as ProductId,
      @UI.lineItem: [{ position: 20 }]
      name
as ProductDescription,
      @UI.lineItem: [{ position: 30 }]
      supplier_id
as SupplierId,
      @UI.lineItem: [{ position: 40, url: 'SearchURL', type:
#WITH_URL }]
      _Supplier.name
as SupplierName,
      @UI.hidden: true
      concat('https://www.google.com/search?q=', _Supplier.name)
```

```
as SearchURL,
      @Semantics.unitOfMeasure: true
      uom,
      @UI.lineItem: [{ position: 50, criticality:
'StockLevelCritical' }]
      stock,
      @UI.hidden: true
      case when stock < 30 then 1
      when stock > 80 then 2
      else 3 end
as StockLevelCritical
}
```

- Result



Overview Page

Task 1: Create new CDS **ZWS##_CDS_OVP**

- Create a new CDS based on cds *ZTMCDS9_I_SO_ITEMS*
- Set annotation *@Metadata.allowExtensions: true*
- Add fields:

Field
SoId as SalesOrder
Id as SalesOrderItem
CustomerName
ProductId
ProductName
CurrencyCode
GrossAmount
NetAmount
'sap-icon://sales-order' as SAPIconUrl

```
@AbapCatalog.sqlViewName: 'ZWK##CDSOVP'
@AbapCatalog.compiler.compareFilter: true
@AbapCatalog.preserveKey: true
@AccessControl.authorizationCheck: #CHECK
@EndUserText.label: 'Overview Page'
@Metadata.allowExtensions: true
define view ZWS##_CDS_OVP
  as select from ZTMCDS9_I_SO_ITEMS
{
  key SoId as SalesOrder,
  key Id as SalesOrderItem,
      CustomerName,
      ProductId,
      ProductName,
      CurrencyCode,
      GrossAmount,
      NetAmount,
      'sap-icon://sales-order' as SAPIconUrl
}
```

Task 2: Create Metadata Extension **ZWS##_ME_CDS_OVP**

- Create new Metadata Extenstion
- Set annotations

Field	Annotation
SalesOrder	lineItem position 10
SalesOrder	selectionField position 10
SalesOrderItem	lineItem position 20
GrossAmount	lineItem position 30

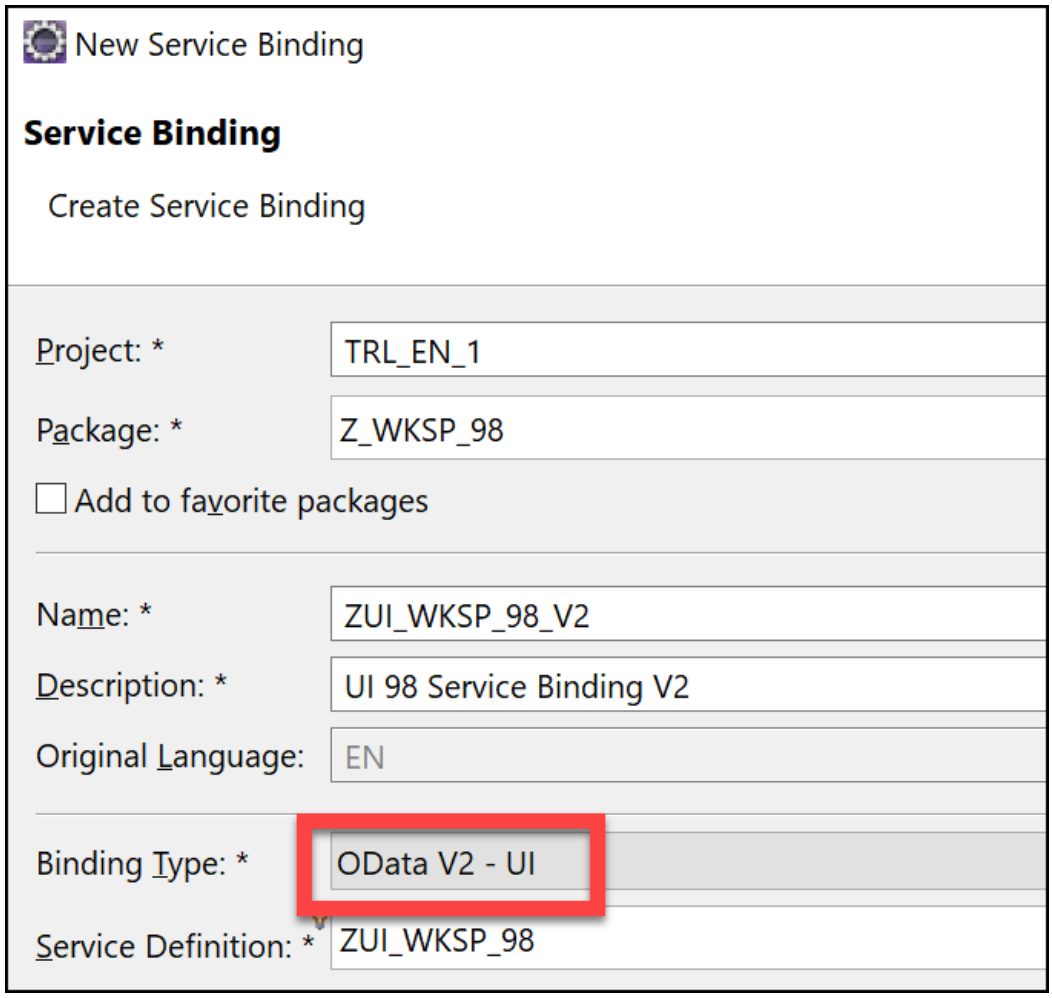
```
@Metadata.layer: #CUSTOMER
annotate view ZWS##_CDS_OVP with
{
  @UI.lineItem: [{ position: 10 }]
  @UI.selectionField: [{ position: 10 }]
  SalesOrder;
  @UI.lineItem: [{ position: 20 }]
  SalesOrderItem;
  @UI.lineItem: [{ position: 30 }]
  GrossAmount;
}
```

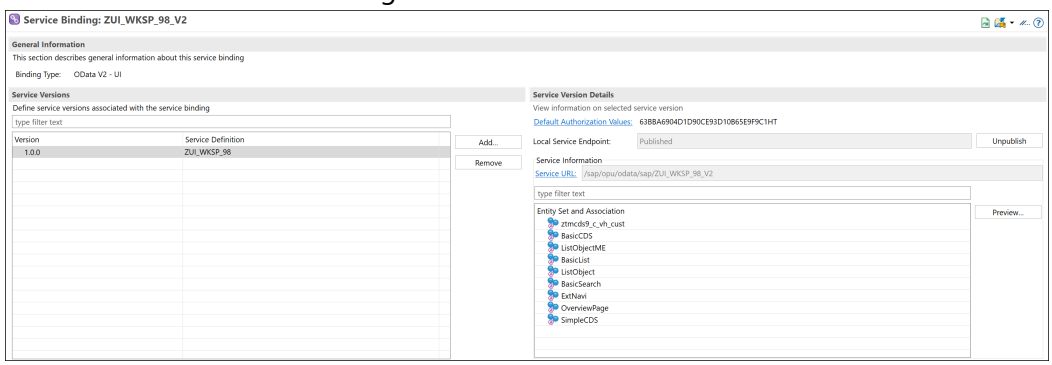
Task 3: Add **ZWS##_CDS_OVP** to *Service Definition*

- Add this new CDS to your *Service Definition* as OverviewPage

```
@EndUserText.label: 'UI ## Service Definition'
define service ZUI_WKSP_## {
  expose ZWS##_CDS_Simple as SimpleCDS;
  expose ZWS##_CDS_Basic as BasicCDS;
  expose ZWS##_CDS_LIST as BasicList;
  expose ZWS##_CDS_LIST_SEARCH as BasicSearch;
  expose ZWS##_CDS_LIST_OBJECT as ListObject;
  expose ZWS##_CDS_C_LO as ListObjectME;
  expose ZWS##_CDS_NAV_EXT as ExtNavi;
  expose ZWS##_CDS_OVP as OverviewPage;
}
```

Task 4: Create Service Binding OData V2 **ZUI_WKSP_##_V2**

- For Overview Pages we need to have a OData V2 service
- Create a new Service Binding with name **ZUI_WKSP_##_V2**
- Use *Binding Type* **OData V2 - UI**
- 

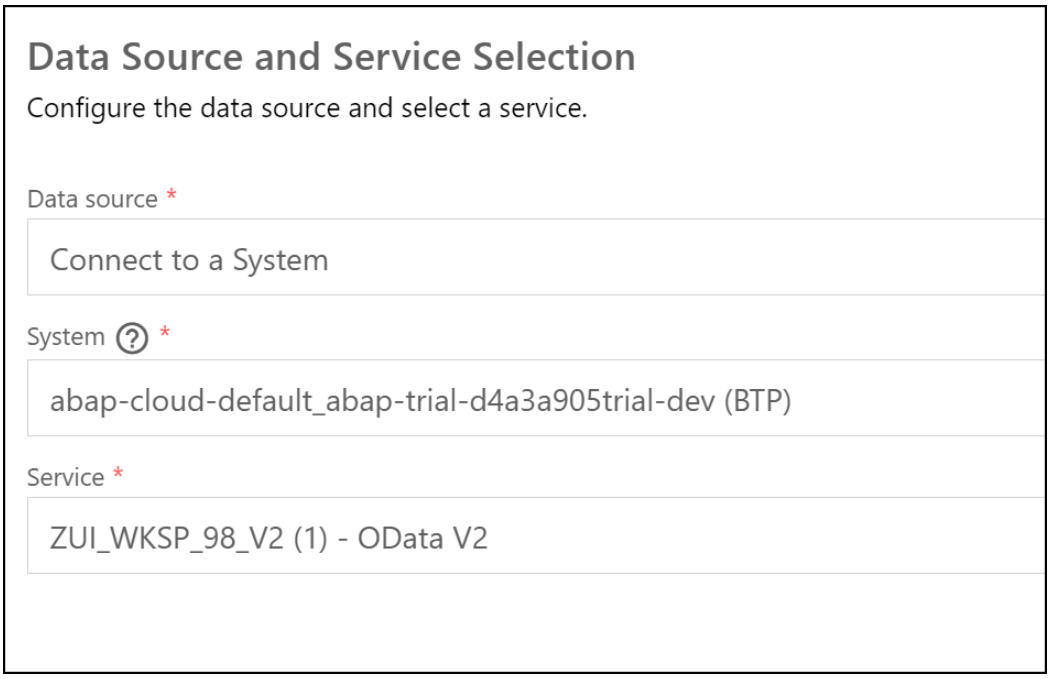
- Publish the new Service Binding
- 

Task 5: Create a new Fiori elements application with the new CDS/OData V2

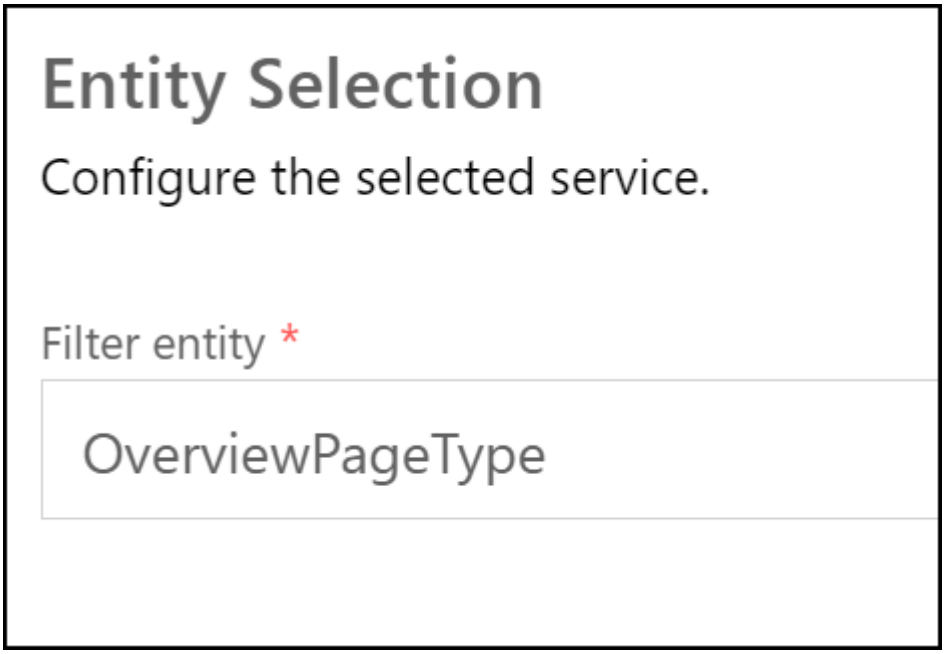
Create a new Fiori Application using the Template Wizard for a *Overview Page*

Field	Value
Data source	Connect to a System
System	abap-cloud-default_xx-dev (BTP)
Service	ZUI_WKSP_##_V2
Main entity	OverviewPageType
Module name	overview-page
Application title	Overview Page
Application namespace	nato.workshop
Description	Overview Page
Project folder path	/home/user/projects
Add deployment configuration	Yes
Add FLP configuration	Yes
Deployment Target	Cloud Foundry
Destination name	abap-cloud-default_xx(SCP)
Add application to managed application router	Yes
Semantic Object	NATO
Action	OverviewPage
Title	Overview Page
Subtitle	Workshop

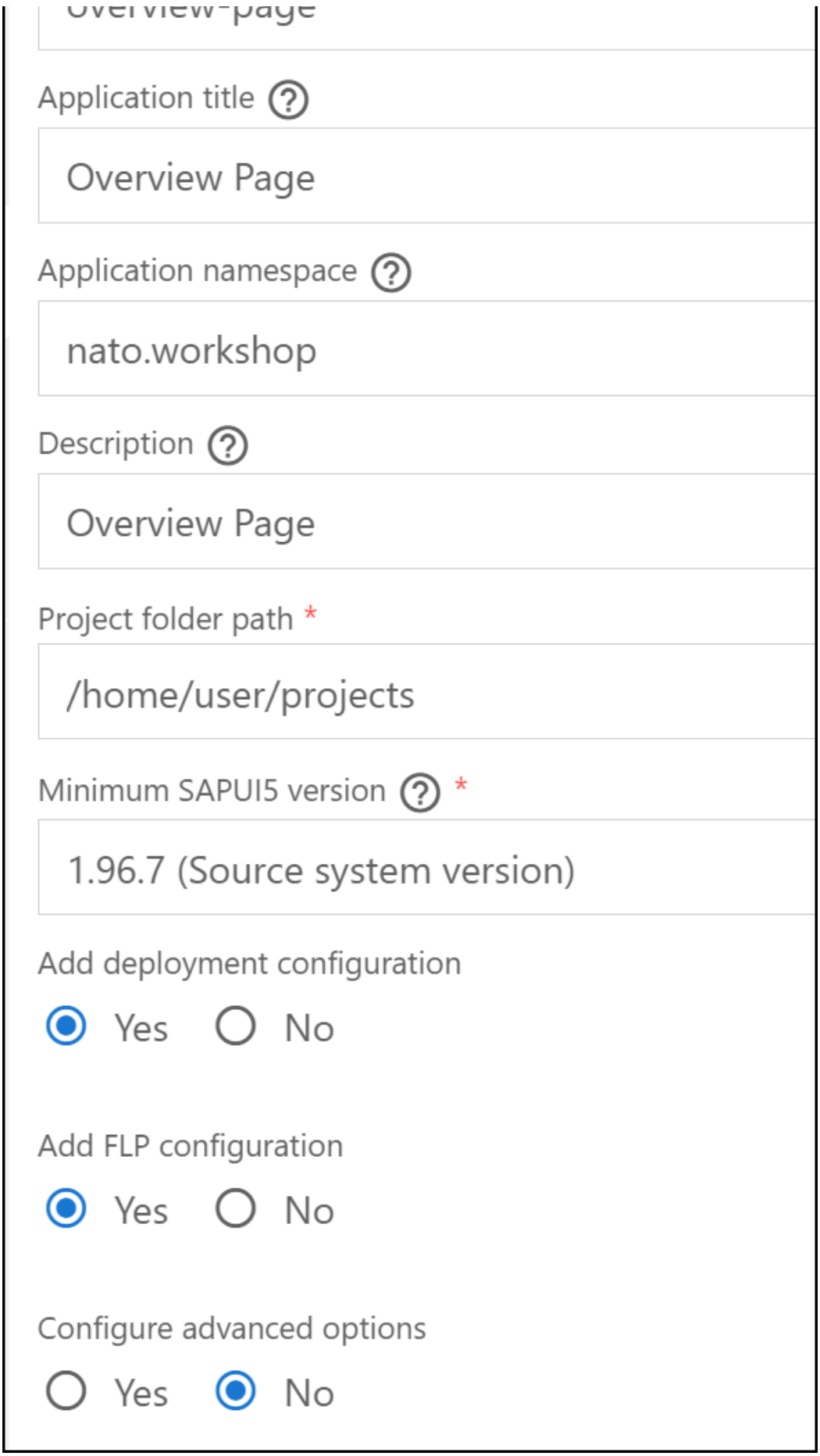
- Data Source and Service Selection



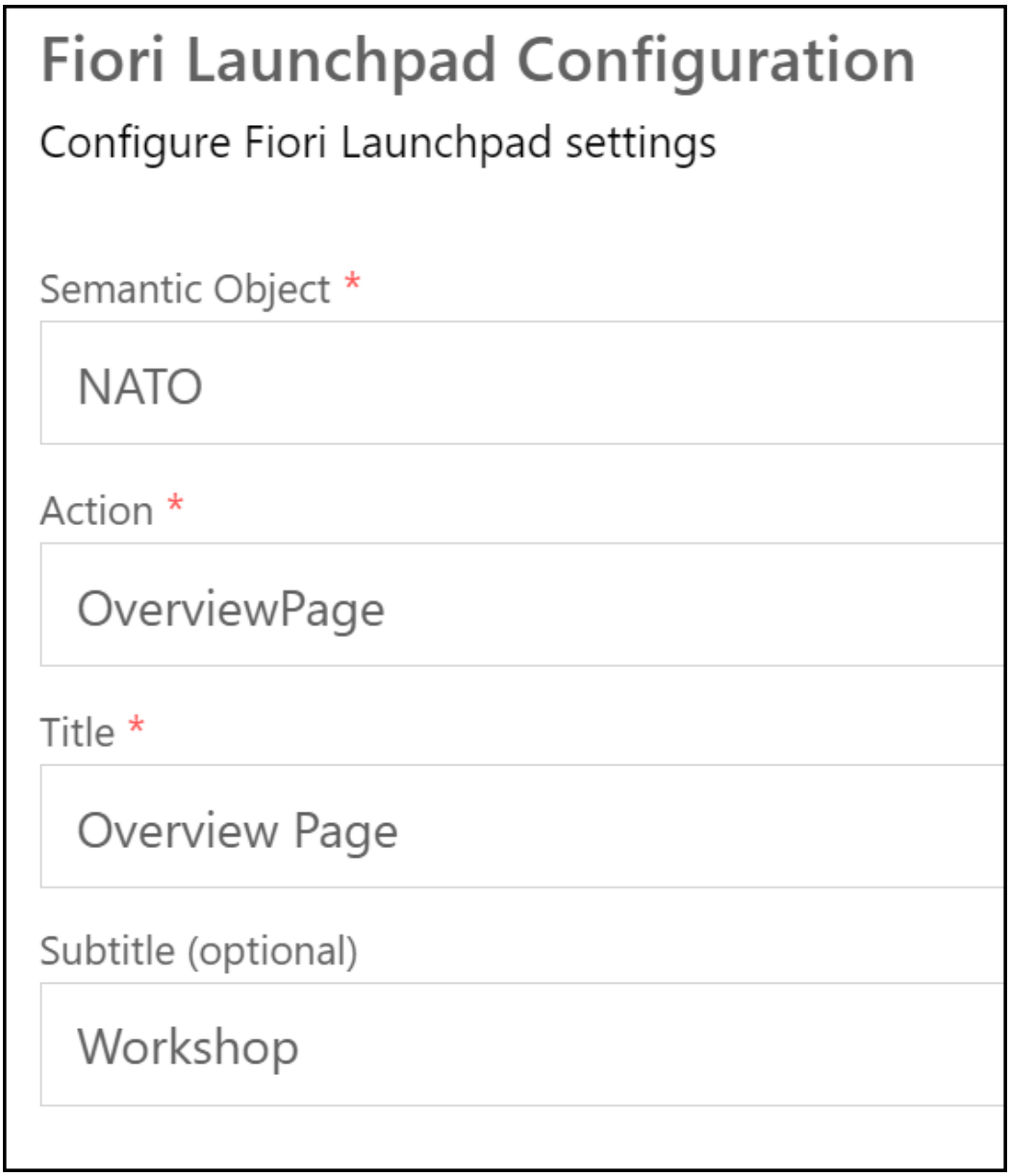
- Entity Selection



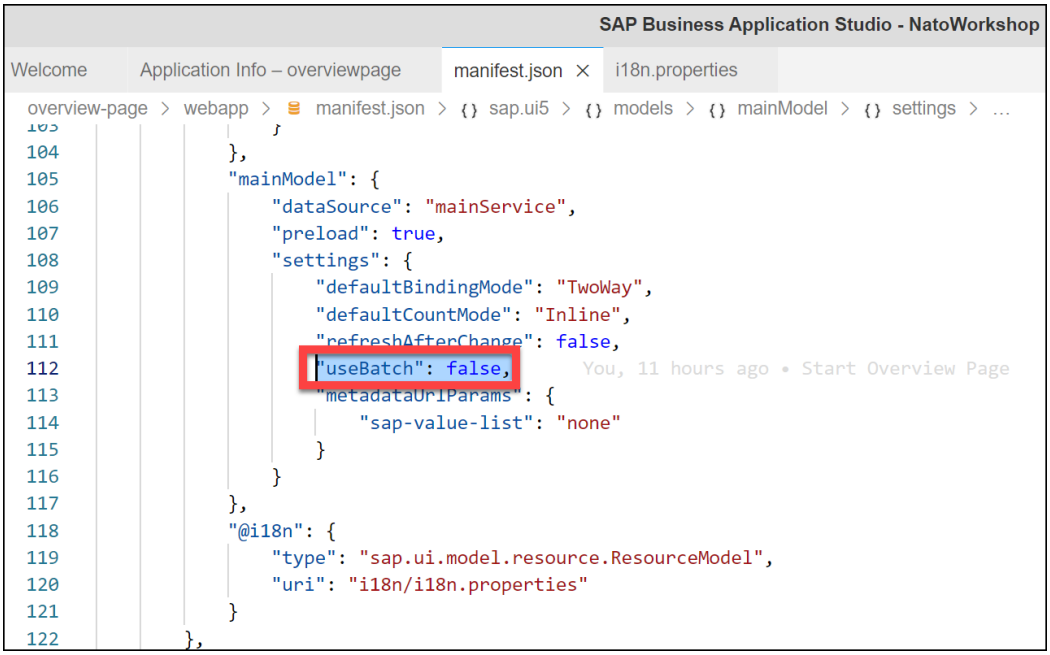
- Project Attributes


overview-page
Application title
Overview Page
Application namespace
nato.workshop
Description
Overview Page
Project folder path *
/home/user/projects
Minimum SAPUI5 version *
1.96.7 (Source system version)
Add deployment configuration
Yes
No
Add FLP configuration
Yes
No
Configure advanced options
Yes
No

- Fiori Launchpad Configuration

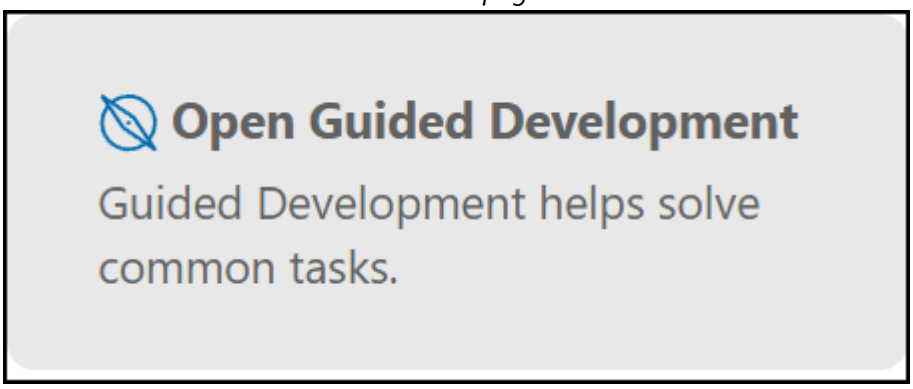


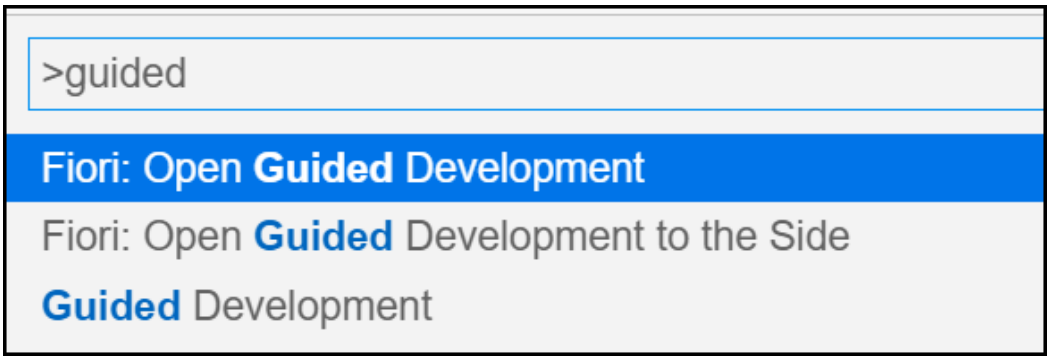
- Open the manifest.json file and add **"useBatch": false,** to the *settings* part of the *mainModel*

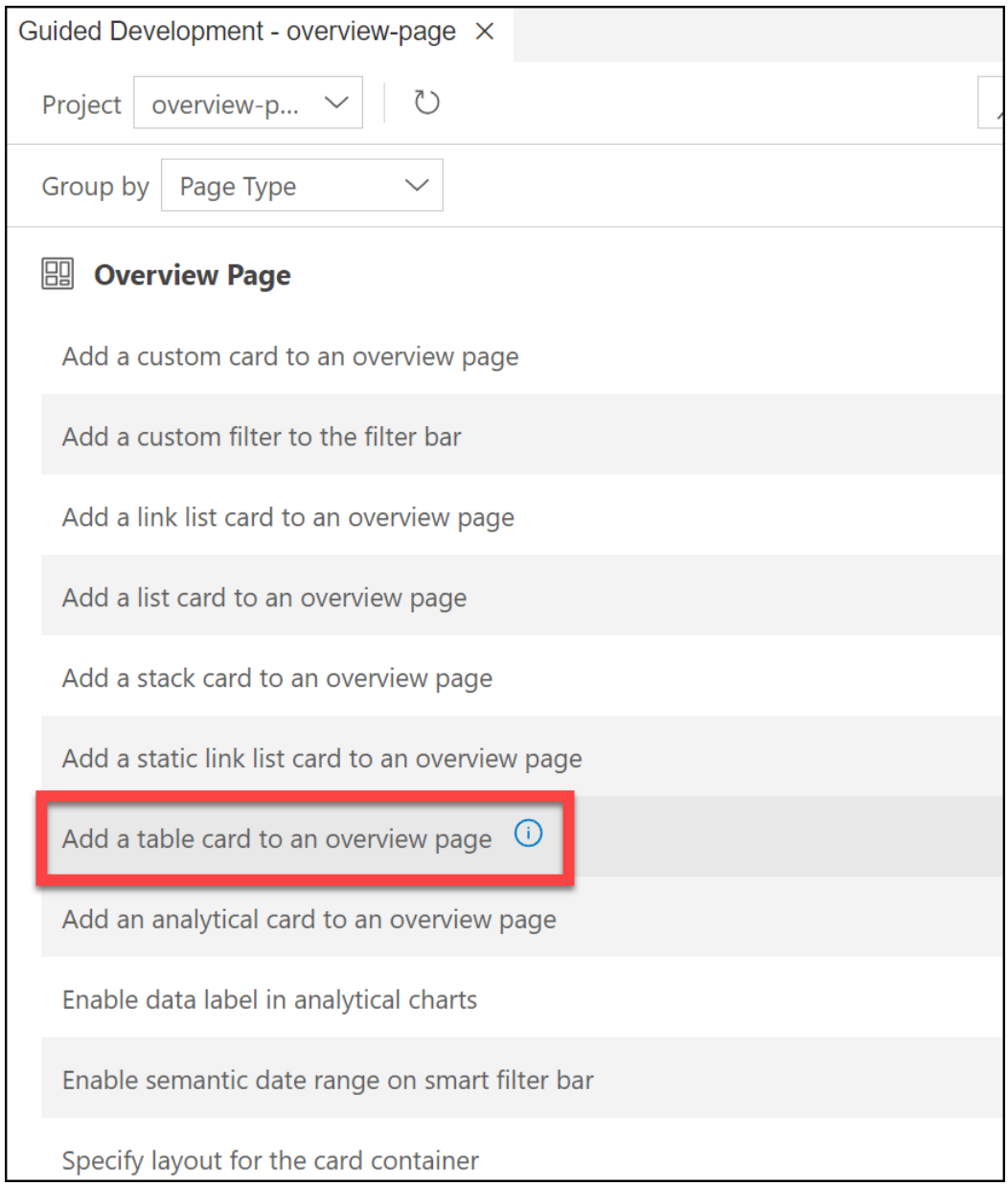


Task 6: Add Table Card

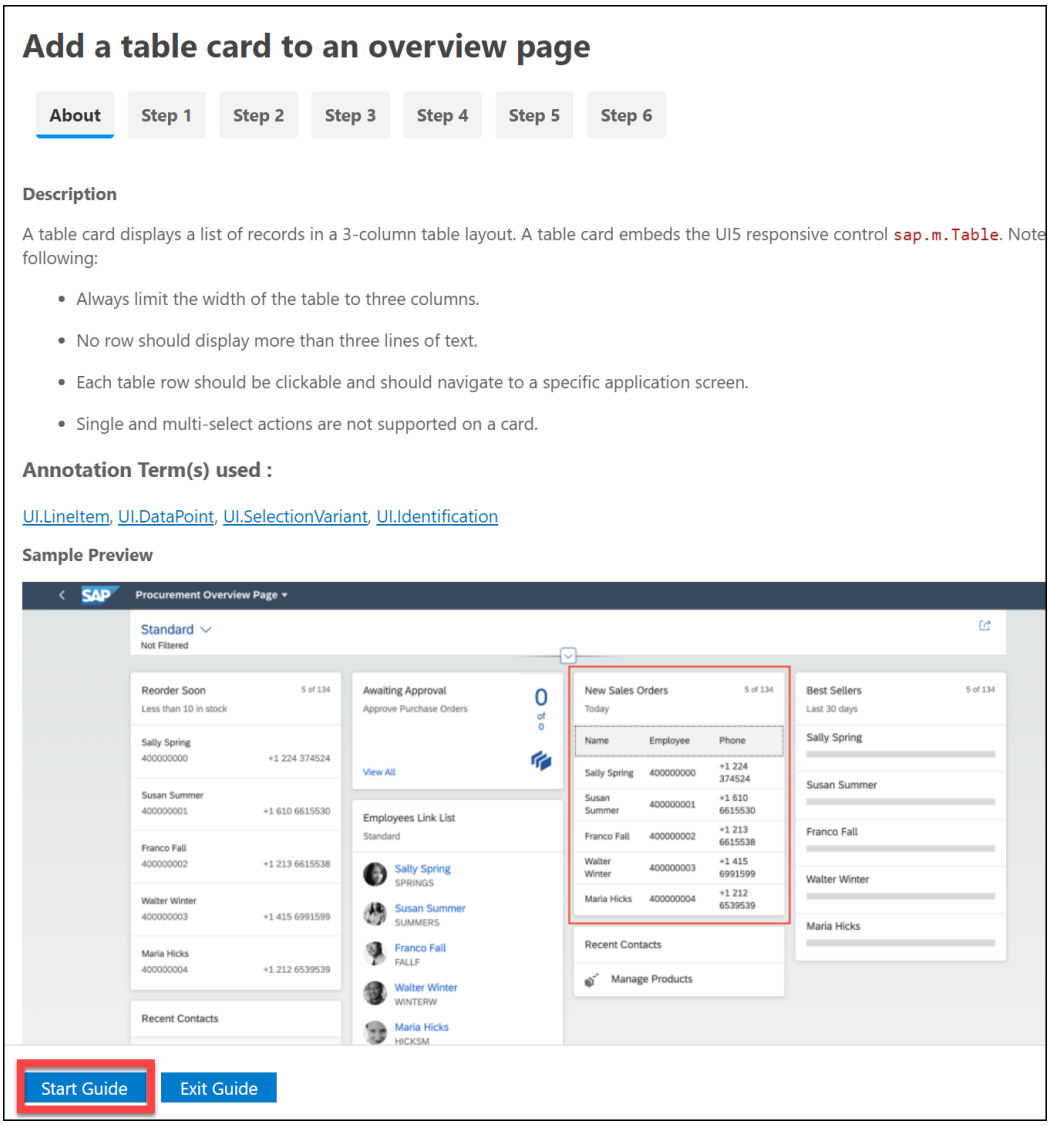
- From the *Application Info* page or via CTRL+SHIFT+P, *Open Guide Development* and select **Add a table card to an overview page*





- 
Guided Development - overview-page
Project
overview-p...
Group by
Page Type
Overview Page
Add a custom card to an overview page
Add a custom filter to the filter bar
Add a link list card to an overview page
Add a list card to an overview page
Add a stack card to an overview page
Add a static link list card to an overview page
Add a table card to an overview page
Add an analytical card to an overview page
Enable data label in analytical charts
Enable semantic date range on smart filter bar
Specify layout for the card container

- Start Guide



- Select *Step 5* and set *Model, Entity Set and Card ID* under bullet **New Card Parameters**

Name	Value
Model	mainModel
Entity Set	OverviewPage
Card ID	Card00

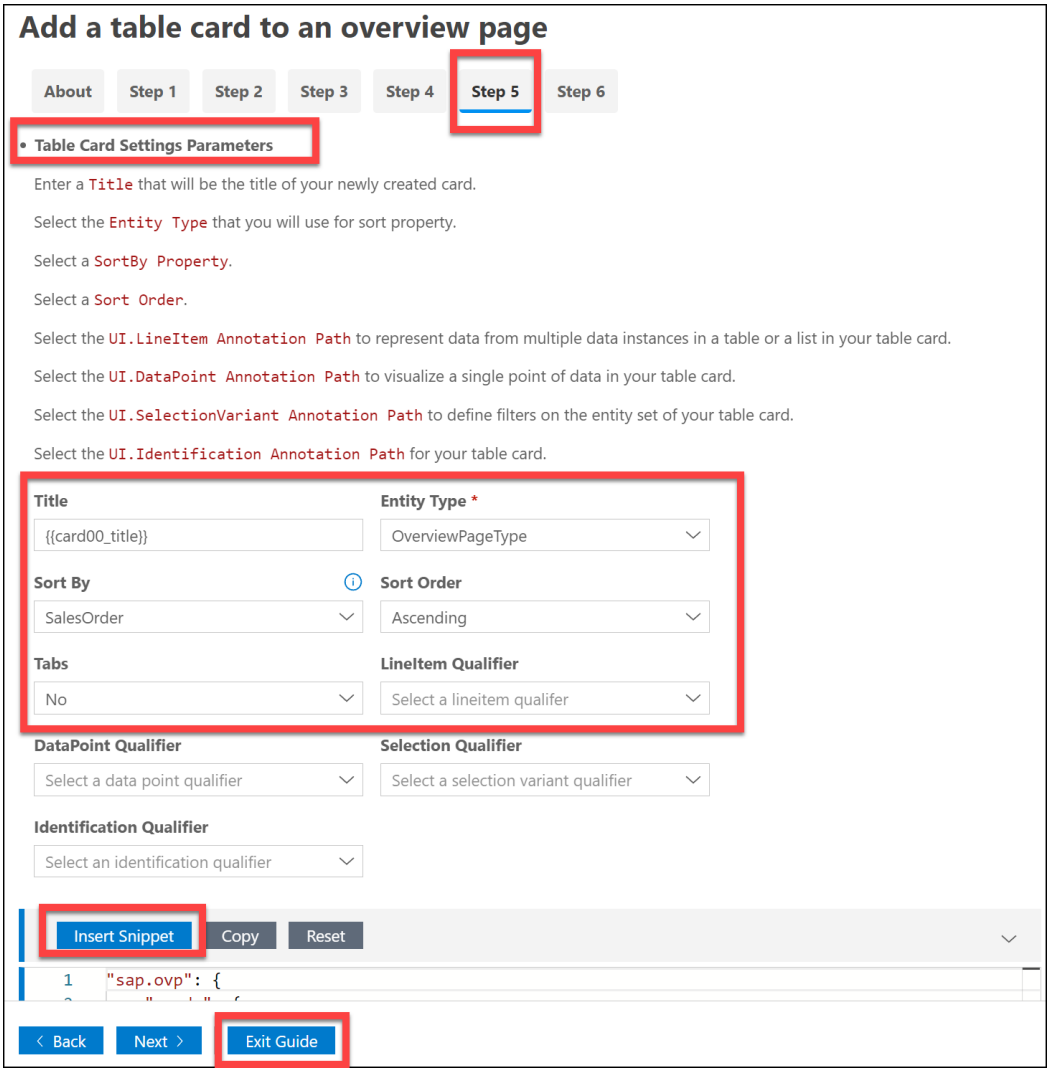
- New Card Parameters

- Scroll down to bullet **Table Card Settings Parameters** and enter following settings:

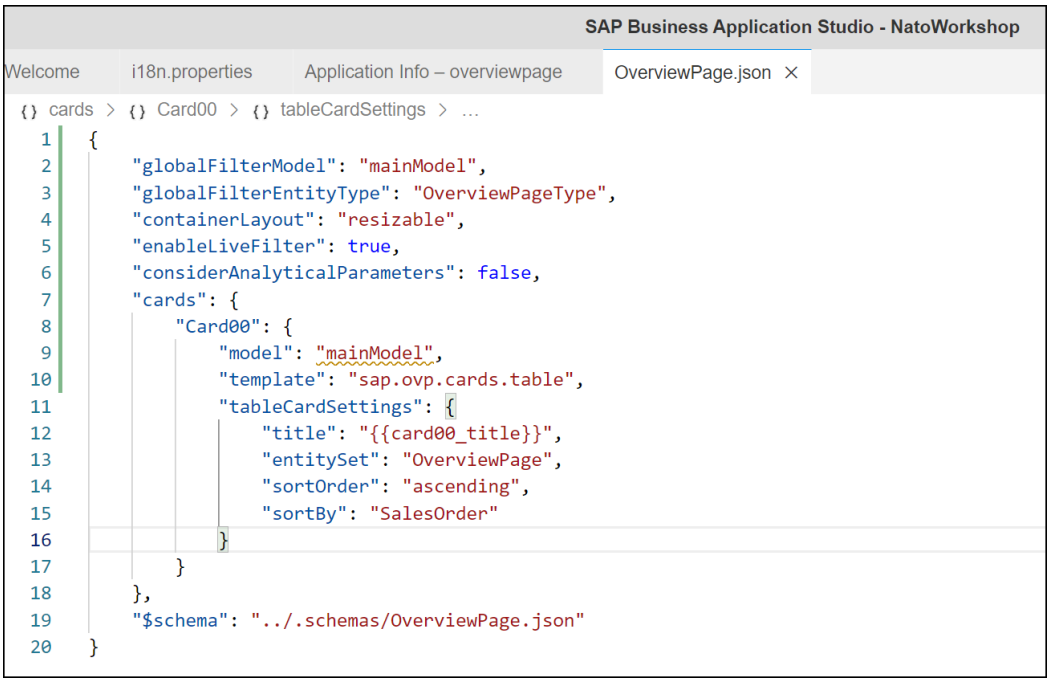
Name	Value
Title	{{card00_title}}

Name	Value
Entity Type	OverviewPageType
Sort By	SalesOrder
Sort Order	Ascending
Tabs	No

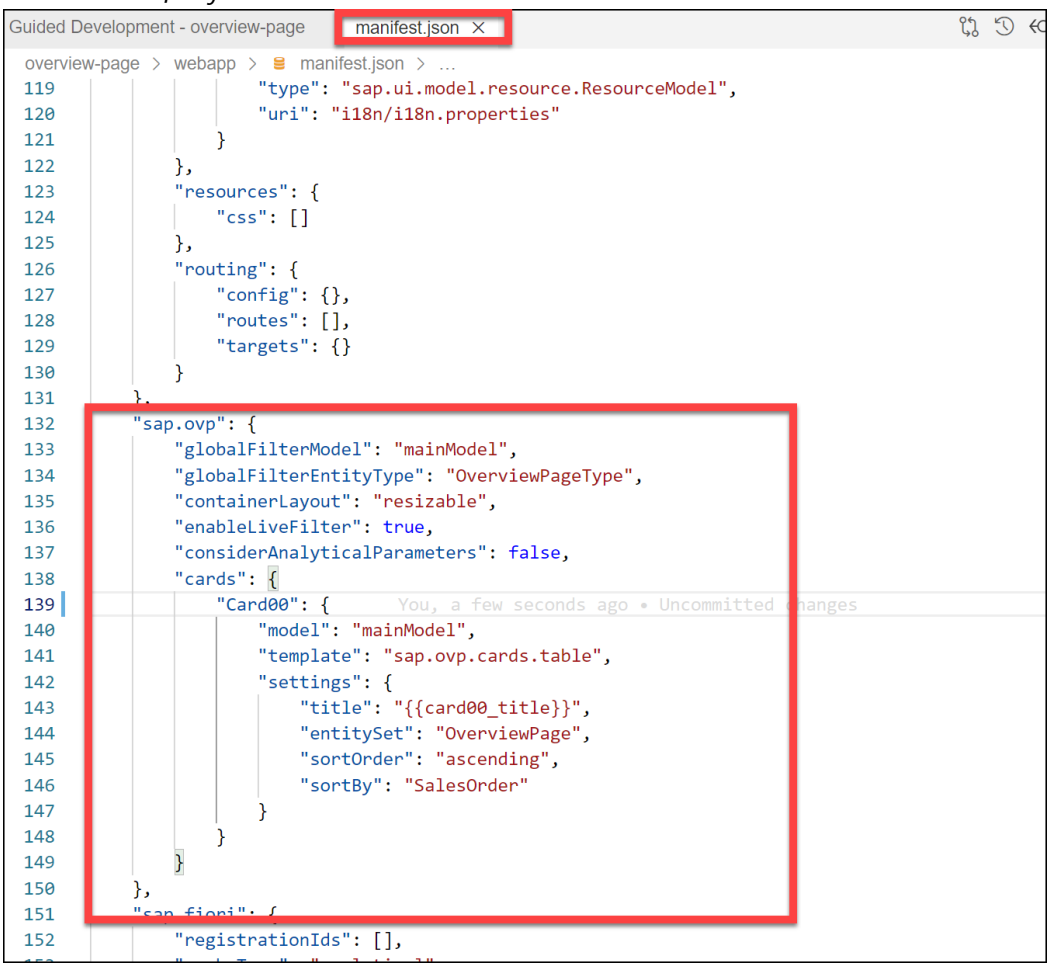
- Select **Insert Snippet** and *Exit Guide*



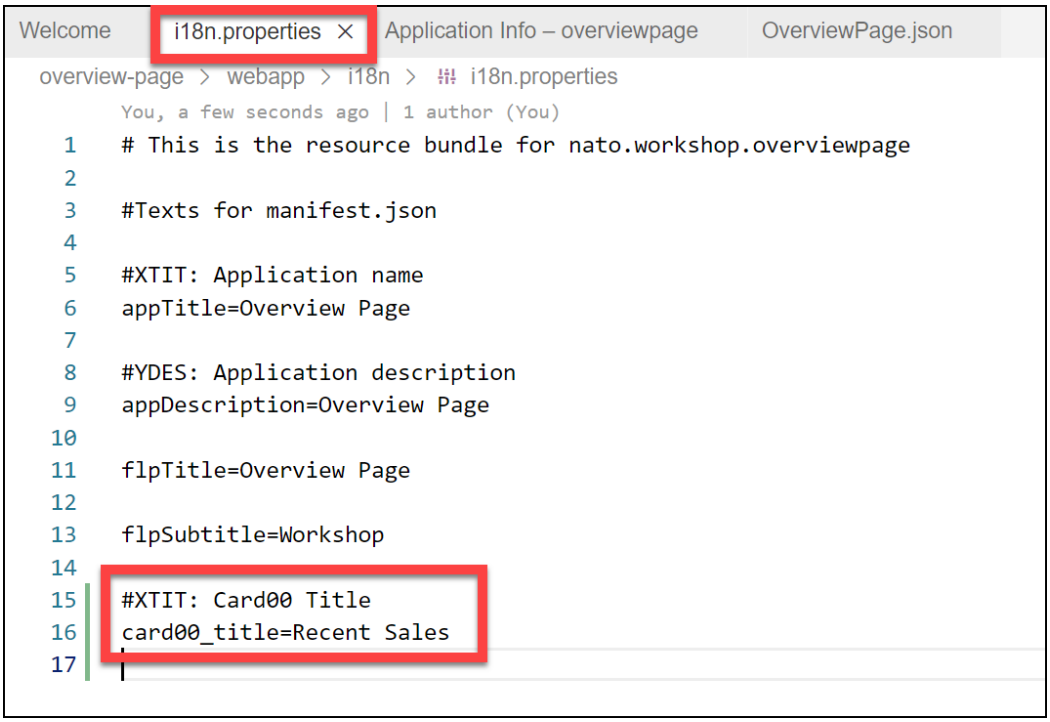
- OverviewPage.json



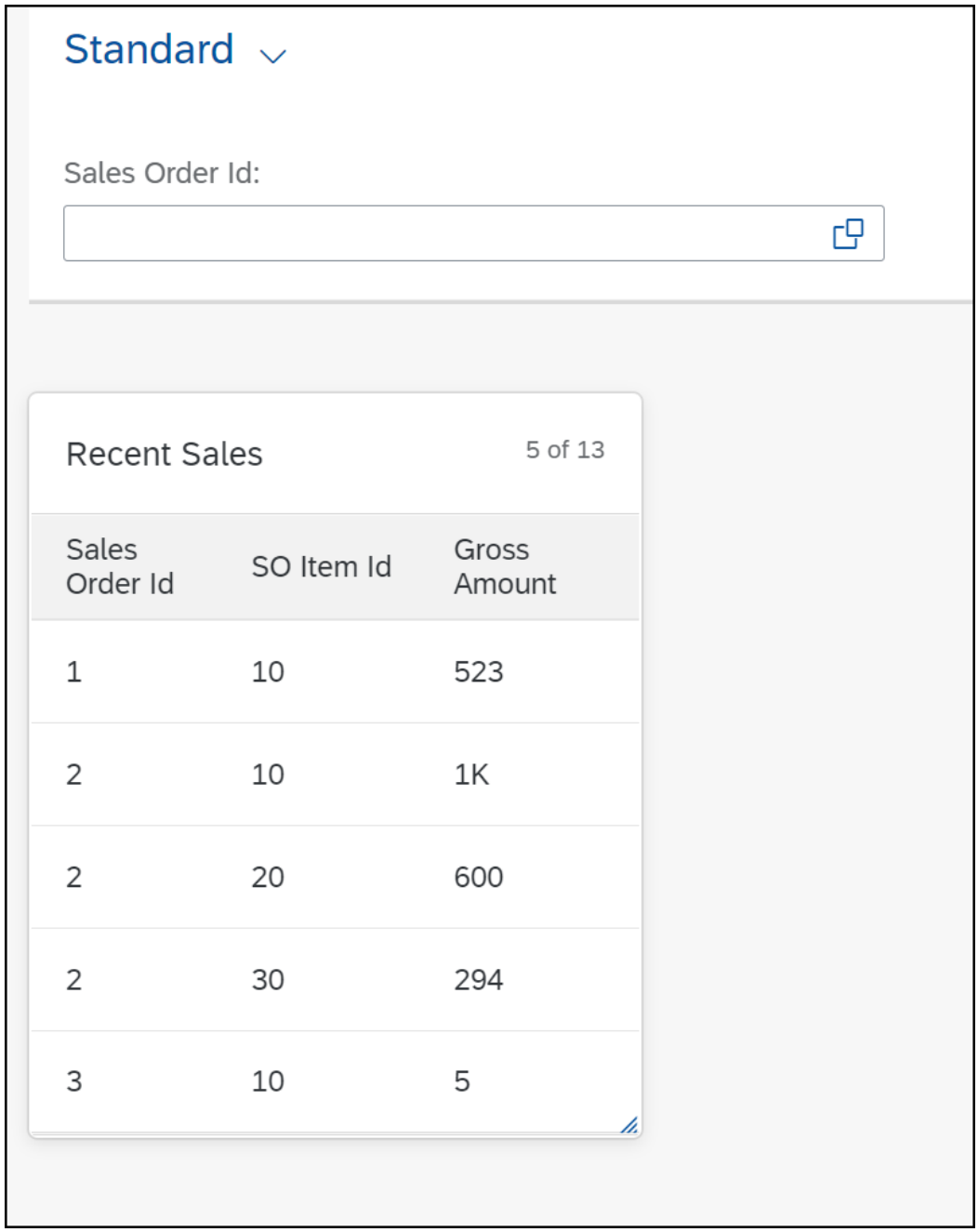
- Look at *manifest.json*



- Add **card00_title=Resent Sales** to *i18n.properties* file



- Now preview the application



Task 7: Add Analytical Card

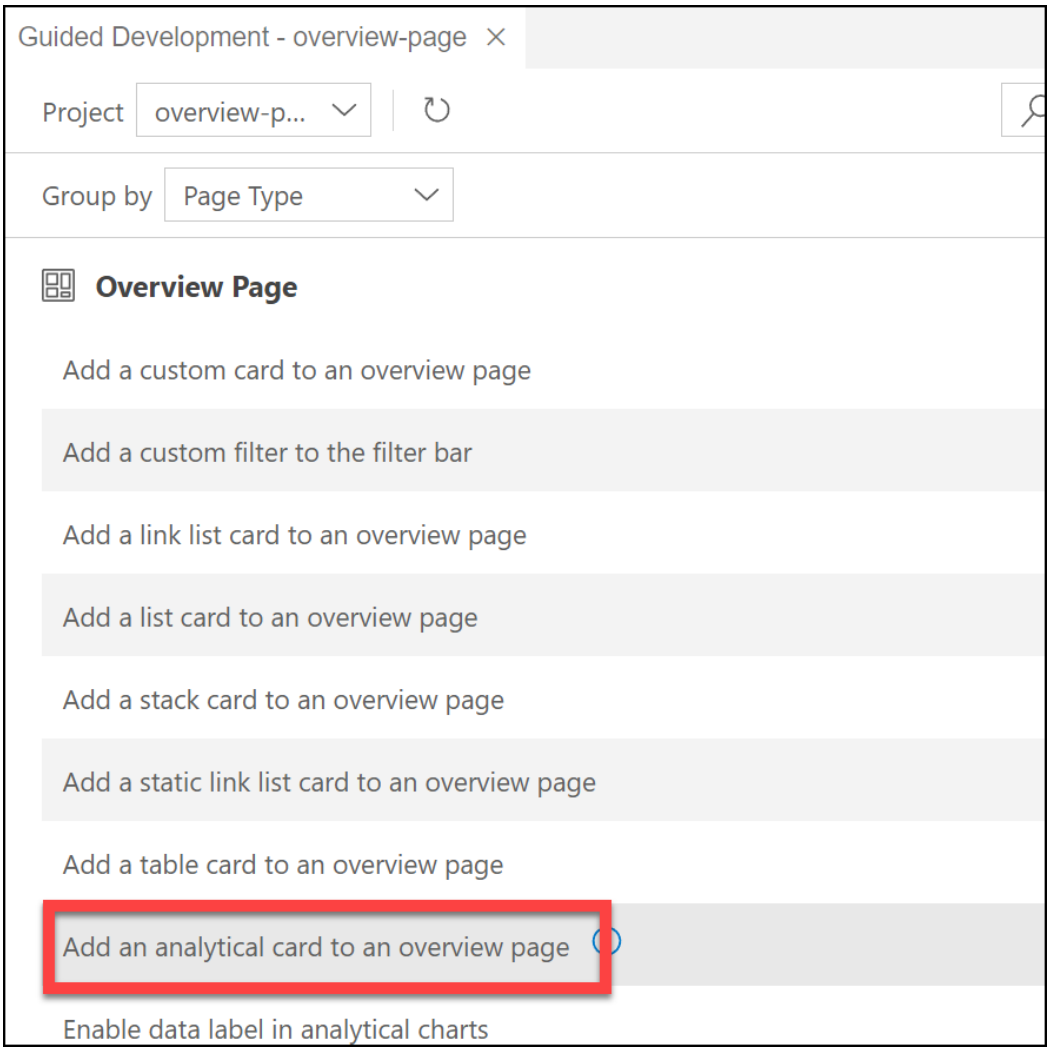
- Add annotations *@UI.chart* to the Metadata Extension

```
@Metadata.layer: #CUSTOMER

@UI.chart: [{
    title: 'Sales by Customer',
    chartType:#DONUT ,
    dimensions: [ 'CustomerName' ],
```

```
    measures: [ 'GrossAmount' ],
    dimensionAttributes: [{
        dimension: 'CustomerName',
        role: #CATEGORY
    }],
    measureAttributes: [{
        measure: 'GrossAmount',
        role: #AXIS_1,
        asDataPoint: false
    }]
}]

annotate view ZWS##_CDS_OVP with
{
  @UI.lineItem: [{ position: 10 }]
  @UI.selectionField: [{ position: 10 }]
  SalesOrder;
  @UI.lineItem: [{ position: 20 }]
  SalesOrderItem;
  @UI.lineItem: [{ position: 30 }]
  GrossAmount;
}
```

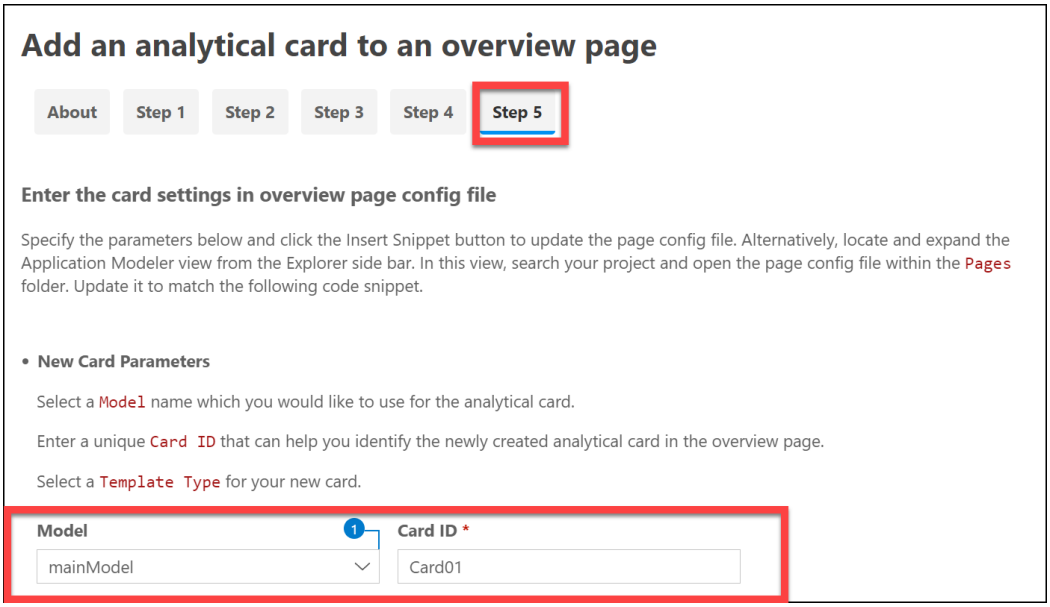
- Add **card01_title=Sales by Customer** to file *i18n.properties*

- Use *Guided Development* to **Add an analytical card to an overview page**



- Start the Guide
- Select *Step 5* and set *Model and Card ID* under bullet **New Card Parameters**

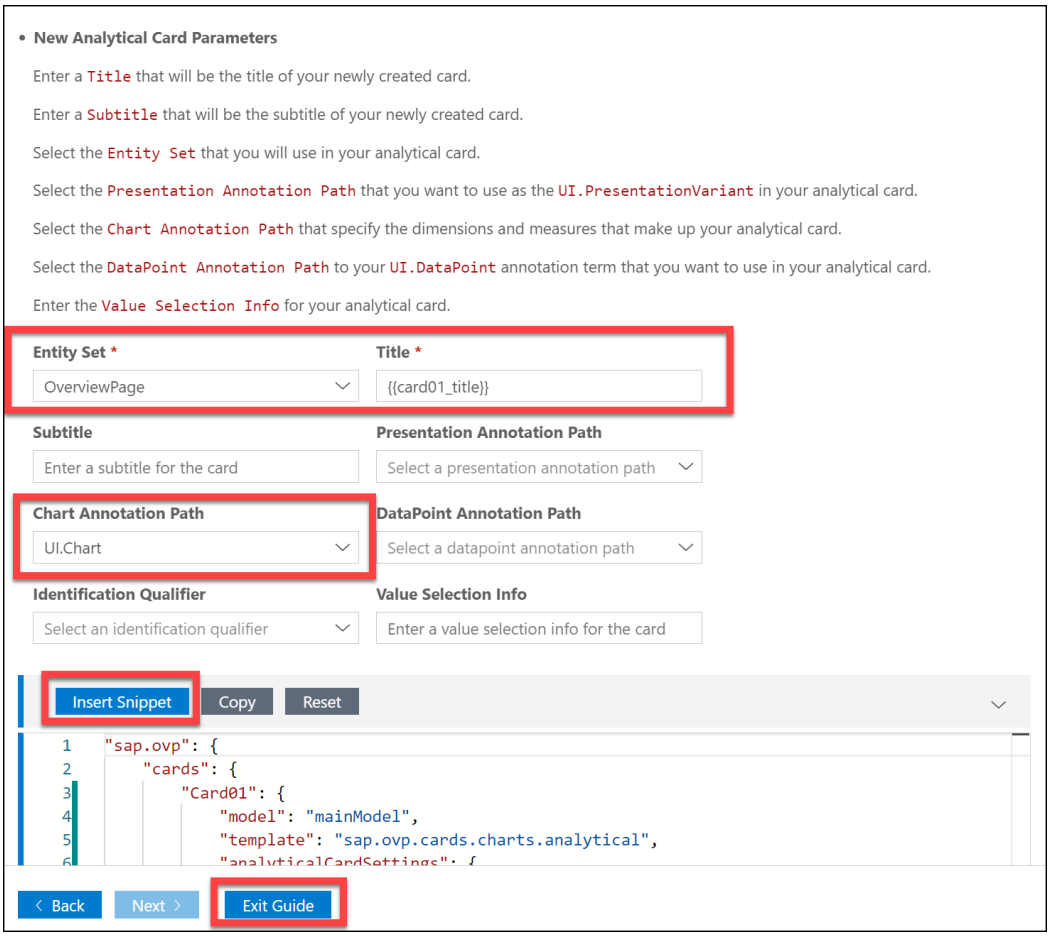
Name	Value
Model	mainModel
Card ID	Card01

- 

- Scroll down to bullet **Table Card Settings Parameters** and enter following settings:

Name	Value
Entity Set	OverviewPage
Title	{{card01_title}}
Chart Annotation Path	UI.Chart

- Select **Insert Snippet** and *Exit Guide*



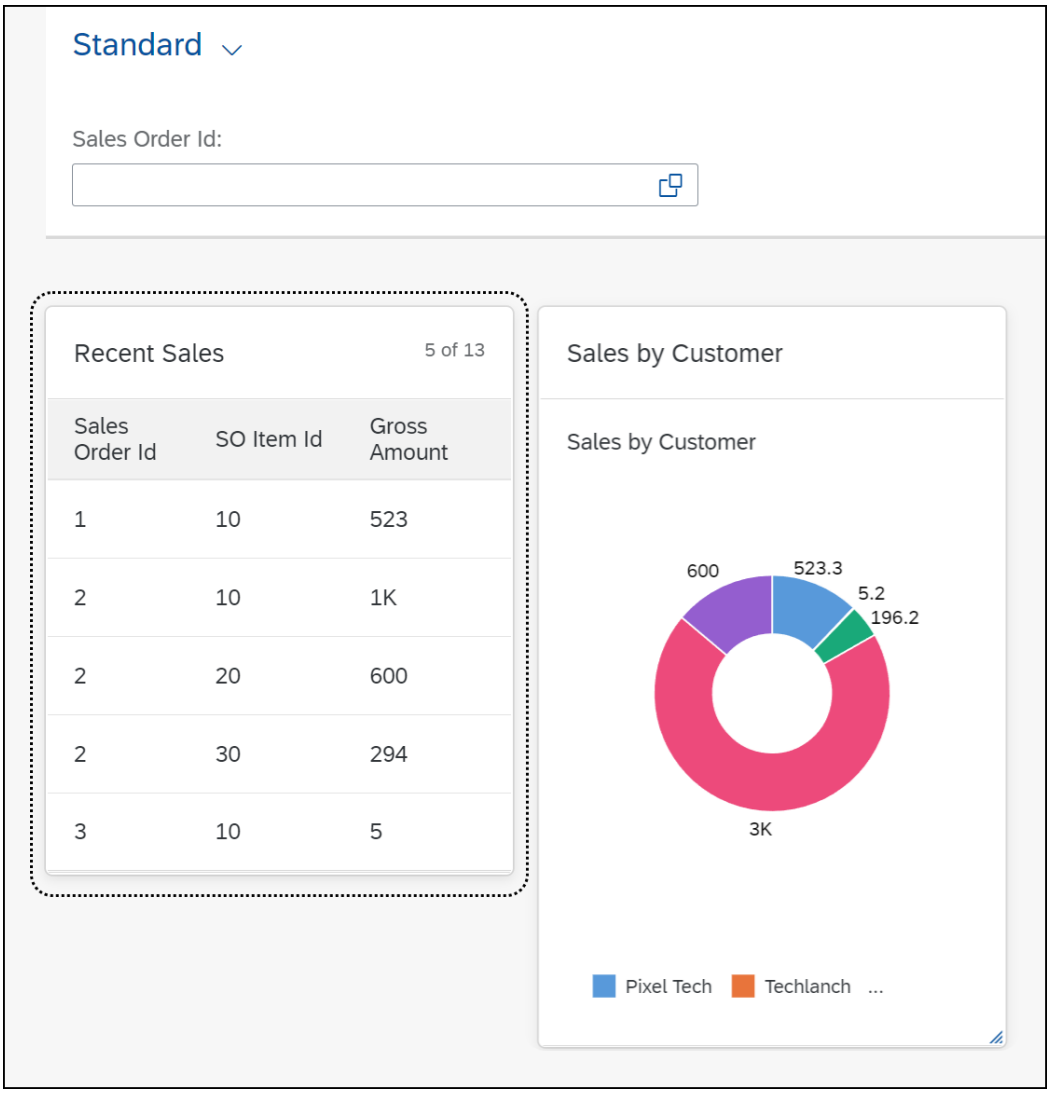
- OverviewPage.json

Welcome | i18n.properties | OverviewPage.json | Application Info – overviewpage

cards > ...

```
{
    "globalFilterModel": "mainModel",
    "globalFilterEntityType": "OverviewPageType",
    "containerLayout": "resizable",
    "enableLiveFilter": true,
    "considerAnalyticalParameters": false,
    "cards": {
        "Card00": {
            "model": "mainModel",
            "template": "sap.ovp.cards.table",
            "tableCardSettings": {
                "title": "{{card00_title}}",
                "entitySet": "OverviewPage",
                "sortOrder": "ascending",
                "sortBy": "SalesOrder"
            }
        },
        "Card01": {
            "model": "mainModel",
            "template": "sap.ovp.cards.charts.analytical",
            "analyticalCardSettings": {
                "title": "{{card01_title}}",
                "entitySet": "OverviewPage",
                "chartAnnotationPath": "com.sap.vocabularies.UI.v1.Chart"
            }
        }
    },
    "$schema": "../.schemas/OverviewPage.json"
}
```

- Now preview the application



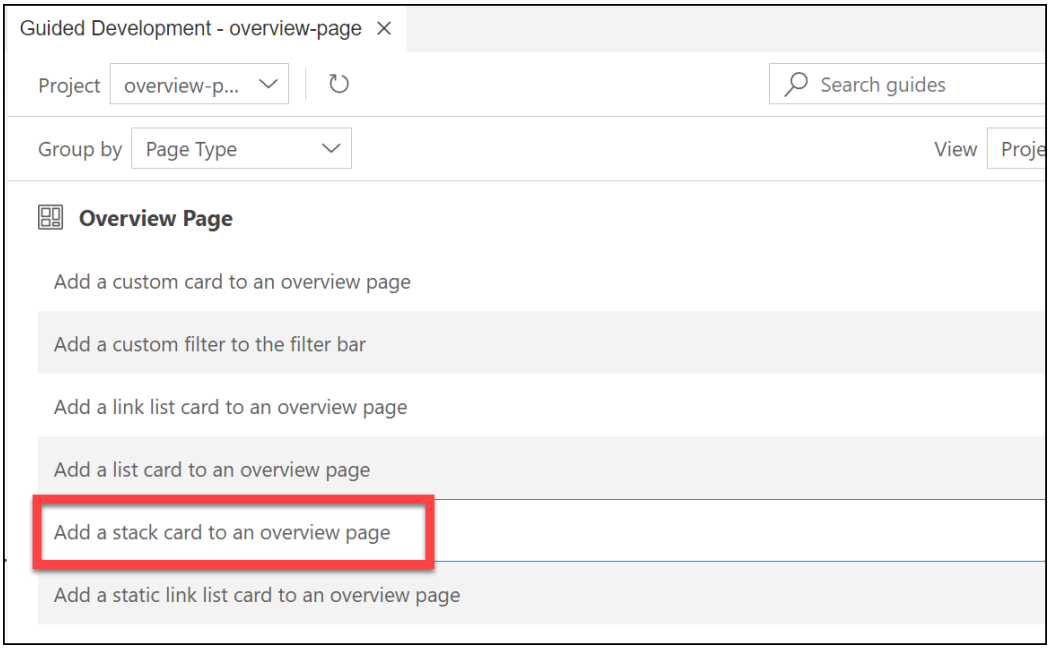
Task 8: Add Stack Card

- Add annotation *@UI.headerInfo* to the Metadata Extension
- Add annotation *@UI.facet*
- Add annotations *@UI.fieldGroup* with *qualifier* **DETAILD** for fields **CustomerName** and **ProductName**

```
@Metadata.layer: #CUSTOMER

@UI.chart: [{
    title: 'Sales by Customer',
    chartType:#DONUT ,
    dimensions: [ 'CustomerName' ],
    measures: [ 'GrossAmount' ],
    dimensionAttributes: [{
        dimension: 'CustomerName',
```

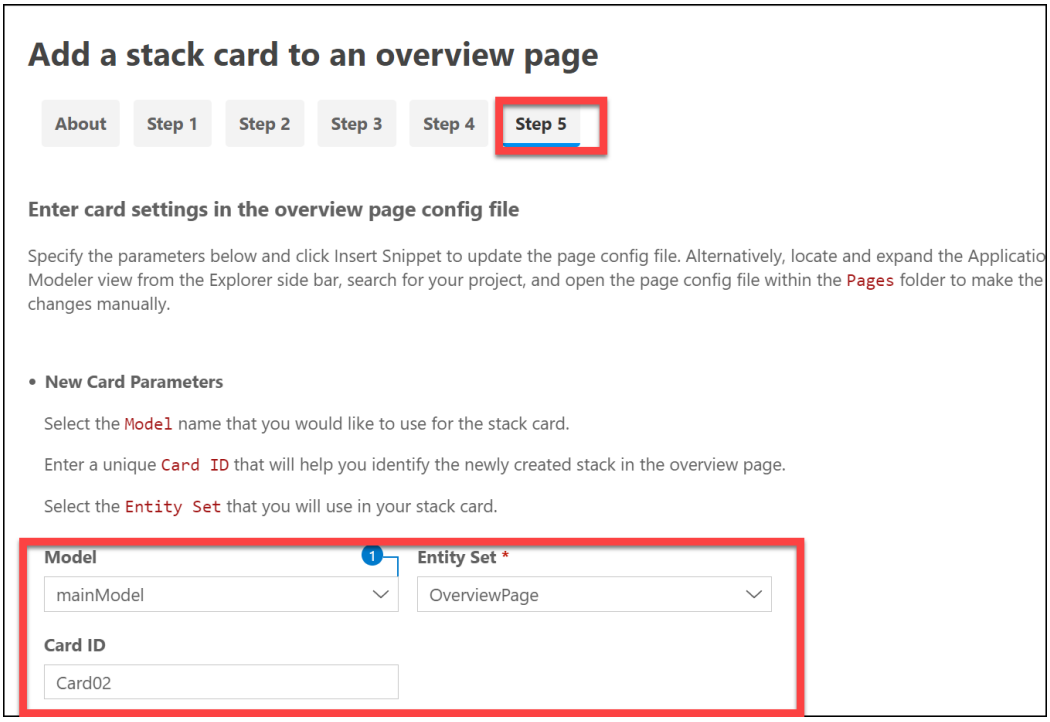
```
        role: #CATEGORY
    }],
    measureAttributes: [{
        measure: 'GrossAmount',
        role: #AXIS_1,
        asDataPoint: false
    }]
}]

@UI.headerInfo: {
  typeNamePlural: 'Sales Orders',
  typeName: 'Sales Order',
  imageUrl: 'SAPIconUrl',
  title: {
    label: 'Sales Order ID',
    value: 'SalesOrder'
  },
  description: {
    label: 'Sales Order Item',
    value: 'SalesOrderItem'
  }
}

annotate view ZWS##_CDS_OVP with
{

  @UI.facet: [{
          type: #FIELDGROUP_REFERENCE,
          targetQualifier: 'DETAILED',
          isSummary: true
      }]

  @UI.lineItem: [{ position: 10 }]
  @UI.selectionField: [{ position: 10 }]
  SalesOrder;
  @UI.lineItem: [{ position: 20 }]
  SalesOrderItem;
  @UI.lineItem: [{ position: 30 }]
  GrossAmount;
  @UI.fieldGroup:[{position: 10, qualifier: 'DETAILED', label:
'Customer'}]
  CustomerName;
  @UI.fieldGroup:[{position: 20, qualifier: 'DETAILED', label:
'Product'}]
  ProductName;
}
```

- Add **card02_title=Sales Orders** and **card02_subtitle=Item** to file *i18n.properties*
- Use *Guided Development* to **Add a stack card to an overview page**



- Start the Guide
- Select *Step 5* and set *Model, Entity Set and Card ID* under bullet **New Card Parameters**

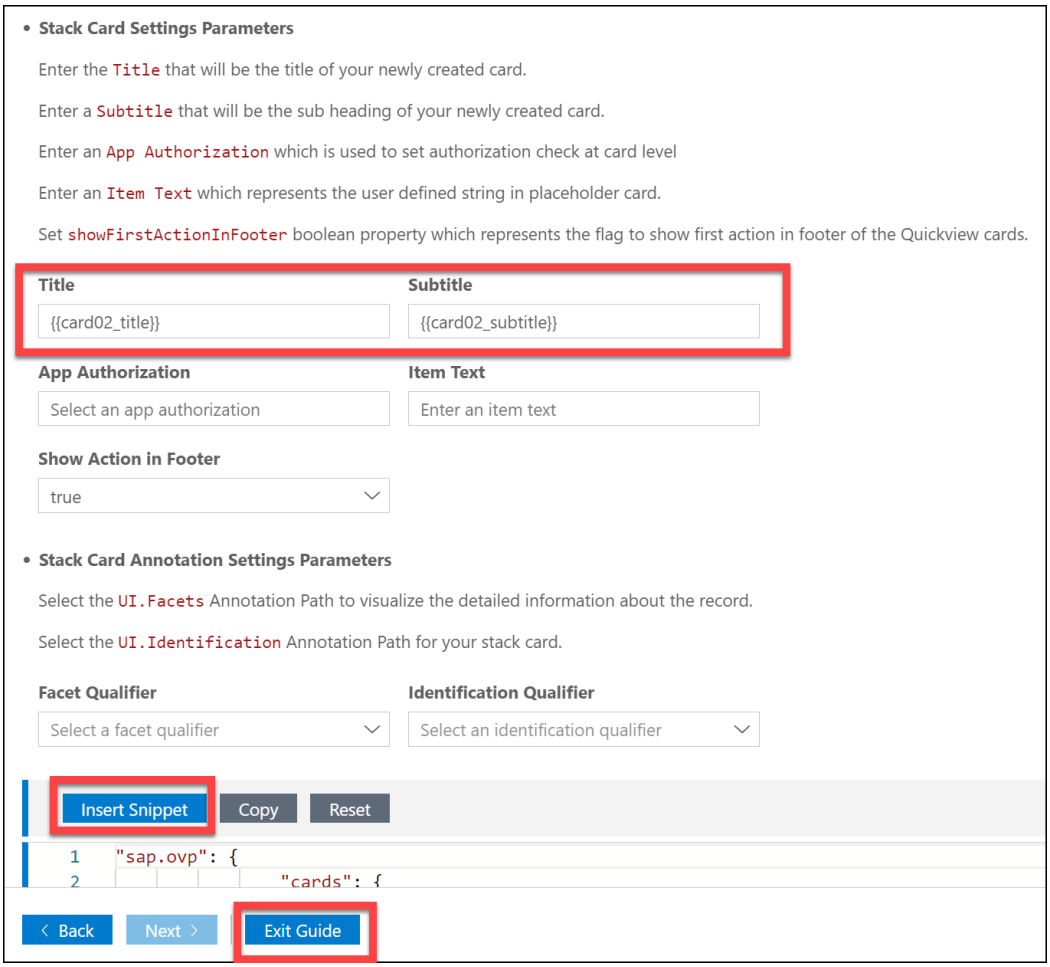
Name	Value
Model	mainModel
Entity Set	OverviewPage
Card ID	Card02

- 

- Scroll down to bullet **Stack Card Settings Parameters** and enter following settings:

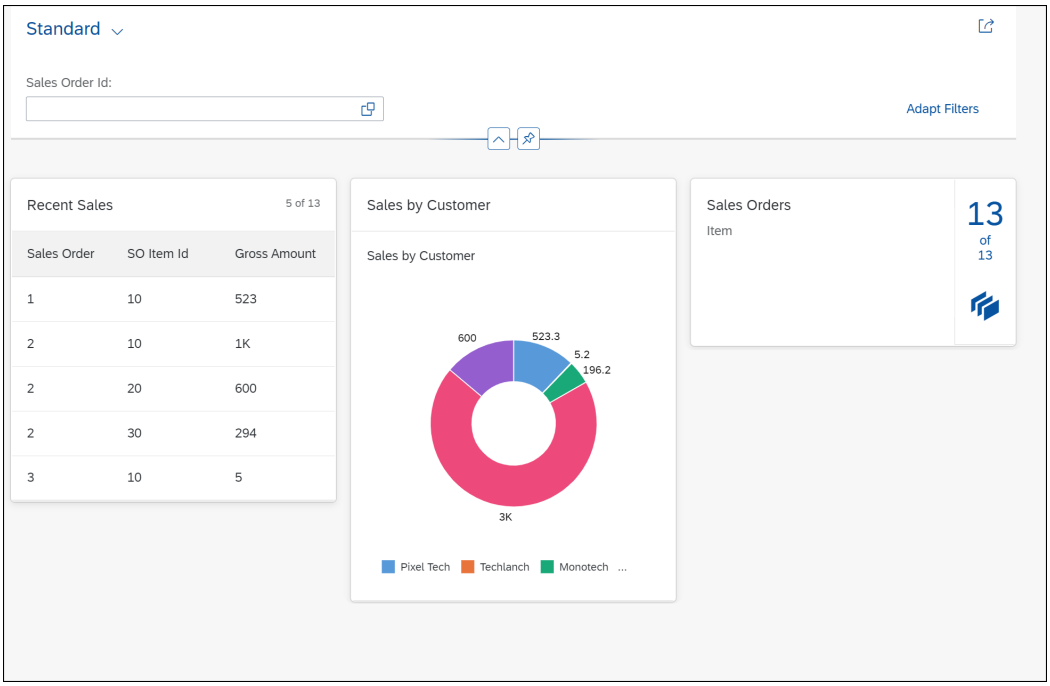
Name	Value
Title	{{card02_title}}
Subtitle	{{card02_subtitle}}

- Select **Insert Snippet** and *Exit Guide*

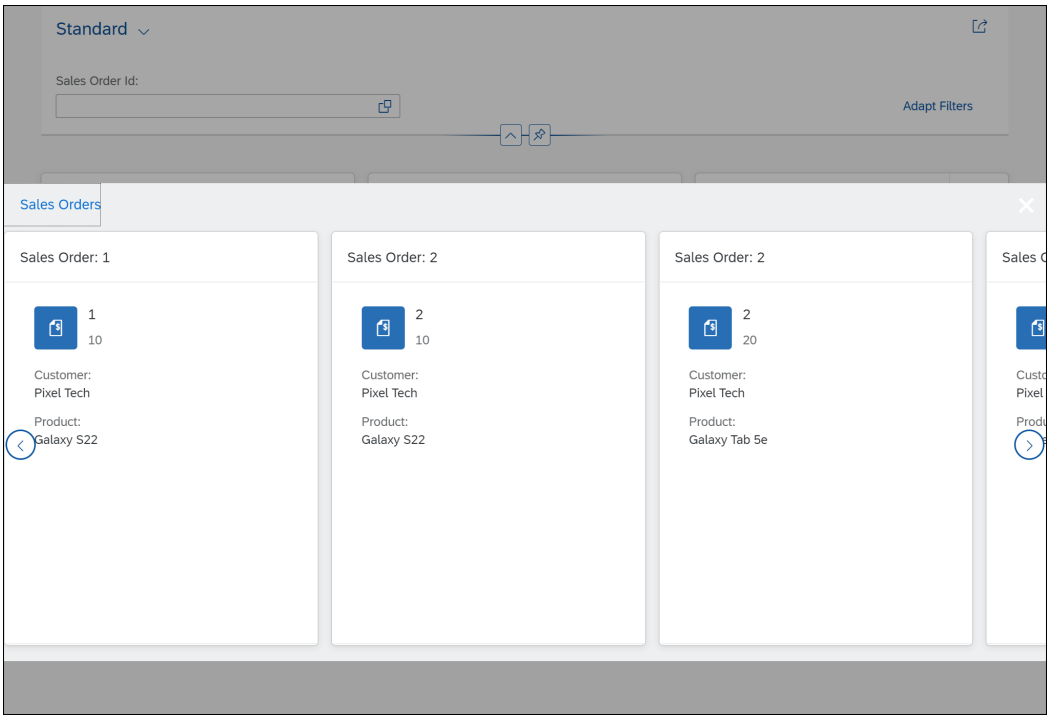


- OverviewPage.json

- Now preview the application



- Click on icon of *Stack Card* to show all **Sales Orders**



Analytical List Page

Task 1: Create a new CDS view **ZWS##_CDS_ALP**

- Create a new CDS base on CDS **ztmcds9_c_items_cube**
- Set the annotatin *@Metadata.allowExtensions: true*
- Add the following fields

Field	Annotation
SoId	
Id	
CustomerName	@AnalyticsDetails.query.display: #TEXT_KEY
ProductId	
ProductName	
CurrencyCode	
GrossAmount	@Aggregation.default: #SUM
NetAmount	@Aggregation.default: #SUM
Quantity	@Aggregation.default: #SUM

```
@AbapCatalog.sqlViewName: 'ZWS##CDSALP'
@AbapCatalog.compiler.compareFilter: true
@AbapCatalog.preserveKey: true
@AccessControl.authorizationCheck: #CHECK
@EndUserText.label: 'Analytical List Page'
@Metadata.allowExtensions: true
define view ZWS##_CDS_ALP as select from ztmcds9_c_so_items_cube {
    key SoId,
    key Id,
    @AnalyticsDetails.query.display: #TEXT_KEY
    CustomerName,
    ProductId,
    ProductName,
    CurrencyCode,
    @Aggregation.default: #SUM
    GrossAmount,
    @Aggregation.default: #SUM
    NetAmount,
    @Aggregation.default: #SUM
```

```
    Quantity
}
```

Task 2: Create new Metadata Extension **ZWS##_ME_CDS_ALP**

- Add annotation *@UI.lineitem: [{position}]* to all fields
- Make field *SoId* a filter selection field
- Add *@UI.headerInfo* to your CDS
- Add *@UI.selectionPresentationVariant, @UI.presntationVariant, @UI.chart and @UI.selectionVariant* to your CDS

```
@Metadata.layer: #CUSTOMER

@UI.headerInfo: { typeName: 'Sales Order Item',
                  typeNamePlural: 'Sales Order Items',
                  title.value: 'CustomerName',
                  description.value: 'ProductName'
                }

@UI.selectionPresentationVariant: [
    {
        qualifier: 'Default',
        presentationVariantQualifier: 'Default',
        selectionVariantQualifier: 'Default'
    }
]
@UI.presentationVariant: [
    {
        qualifier: 'Default',
        visualizations: [{
            type: #AS_CHART,
            qualifier: 'ChartDefault'
        }]
    }
]
@UI.chart: [
    {
        qualifier: 'ChartDefault',
        title: 'Sales by Customer',
        chartType: #COLUMN,
        dimensions: [ 'CustomerName', 'ProductName' ],
        measures: [ 'GrossAmount' ],
        dimensionAttributes: [
            {
```

```
                dimension: 'CustomerName',
                role: #CATEGORY
            },
            {
                dimension: 'ProductName',
                role: #CATEGORY
            }
        ],
        measureAttributes: [{
            measure: 'GrossAmount',
            role: #AXIS_1,
            asDataPoint: false
        }]
    }
]
@UI.selectionVariant: [
    {
        qualifier: 'Default',
        text: 'Default'
    }
]

annotate view ZWS##_CDS_ALP
    with
{
    @UI.lineItem : [{ position: 10}]
    @UI.selectionField: [{ position: 10 }]
    SoId;
    @UI.lineItem : [{ position: 20}]
    Id;
    @UI.lineItem : [{ position: 30}]
    CustomerName;
    @UI.lineItem : [{ position: 40}]
    ProductId;
    @UI.lineItem : [{ position: 50}]
    ProductName;
    @UI.lineItem : [{ position: 60}]
    CurrencyCode;
    @UI.lineItem : [{ position: 70}]
    GrossAmount;
    @UI.lineItem : [{ position: 80}]
    NetAmount;
    @UI.lineItem : [{ position: 90}]
    Quantity;
}
```

Task 3: Add the new CDS to your *Service Definition*

- Add your CDS as **AnalyticalListPage** to your *Service Definition*

```
@EndUserText.label: 'UI ## Service Definition'
define service ZUI_WKSP_## {
  expose ZWS##_CDS_Simple as SimpleCDS;
  expose ZWS##_CDS_Basic as BasicCDS;
  expose ZWS##_CDS_LIST as BasicList;
  expose ZWS##_CDS_LIST_SEARCH as BasicSearch;
  expose ZWS##_CDS_LIST_OBJECT as ListObject;
  expose ZWS##_CDS_C_LO as ListObjectME;
  expose ZWS##_CDS_NAV_EXT as ExtNavi;
  expose ZWS##_CDS_OVP as OverviewPage;
  expose ZWS##_CDS_ALP as AnalyticalListPage;
}
```

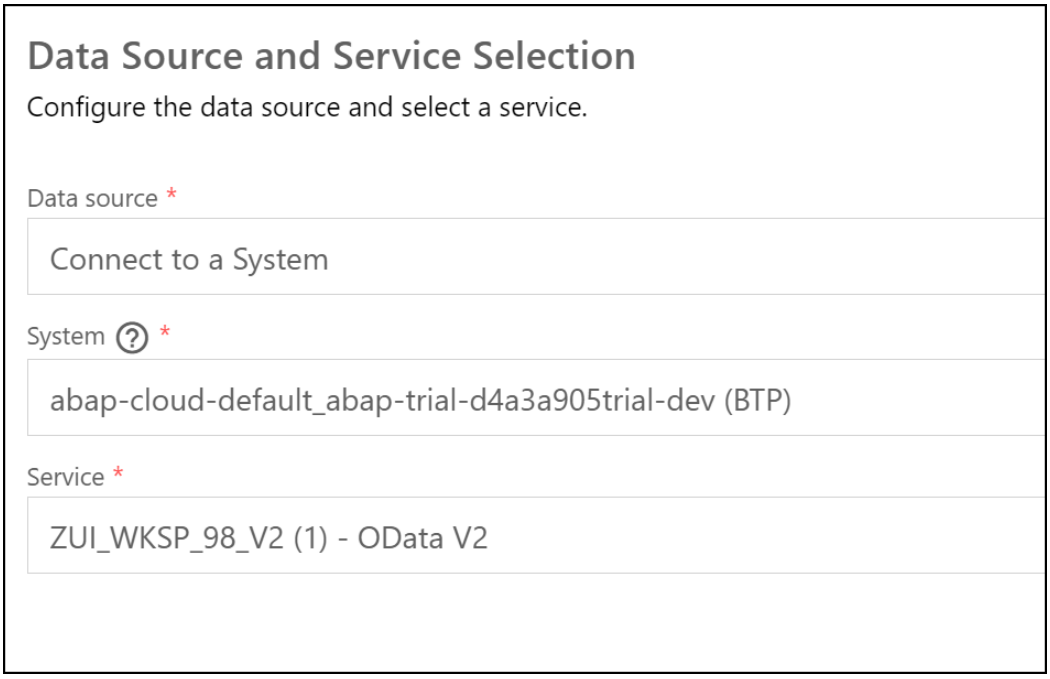
Task 4: Create a new Fiori elements application with the CDS/OData V2

Create a new Fiori Application using the Template Wizard for a *Analytical List Pagee*

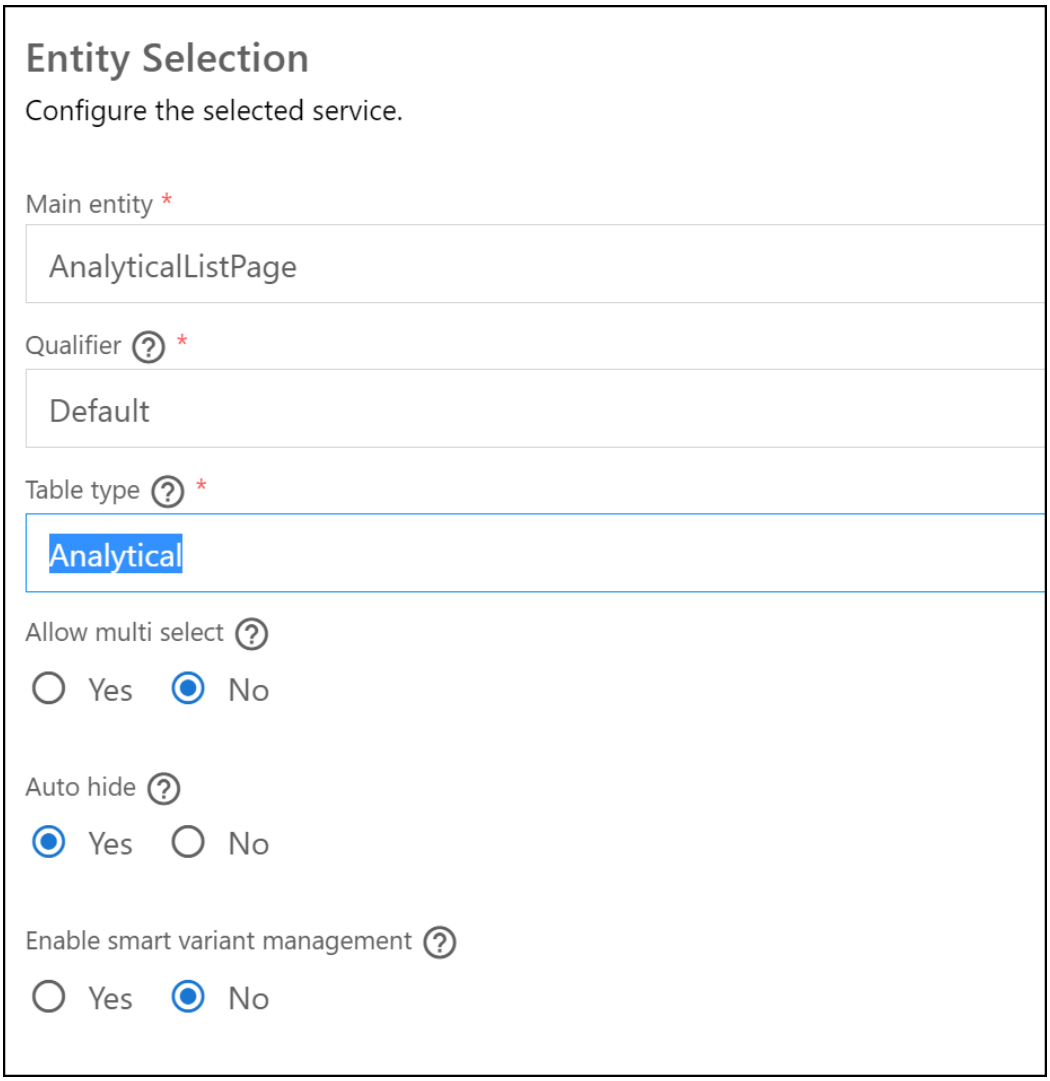
Field	**Value**
Data source	Connect to a System
System	abap-cloud-default_xx-dev (BTP)
Service	ZUI_WKSP_##_V2
Main entity	AnalyticalListPage
Qualifier	Default
Table type	Analytical
Allow multi select	No
Auto hide	Yes
enable smart variant management	No
Module name	analytic-page
Application title	Analytical List Page
Application namespace	nato.workshop
Description	Analytical List Page
Project folder path	/home/user/projects

Field	Value
Add deployment configuration	Yes
Add FLP configuration	Yes
Deployment Target	Cloud Foundry
Destination name	abap-cloud-default_xx(SCP)
Add application to managed application router	Yes
Semantic Object	NATO
Action	AnalyticPage
Title	Analytical List Page
Subtitle	Workshop

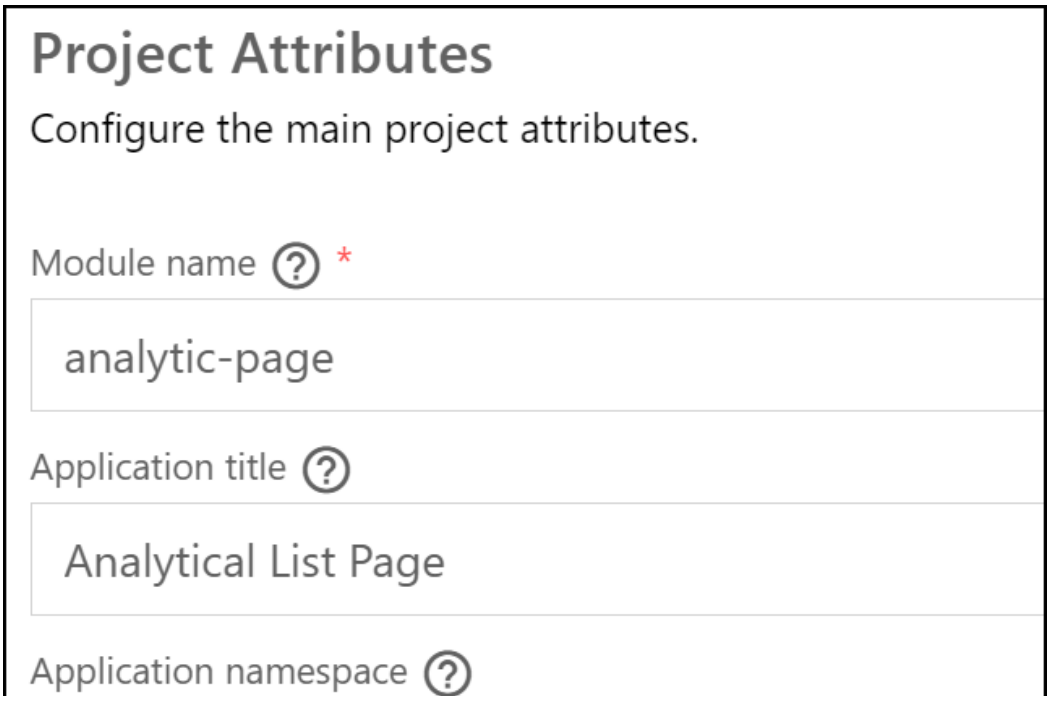
- Data Source and Service Selection

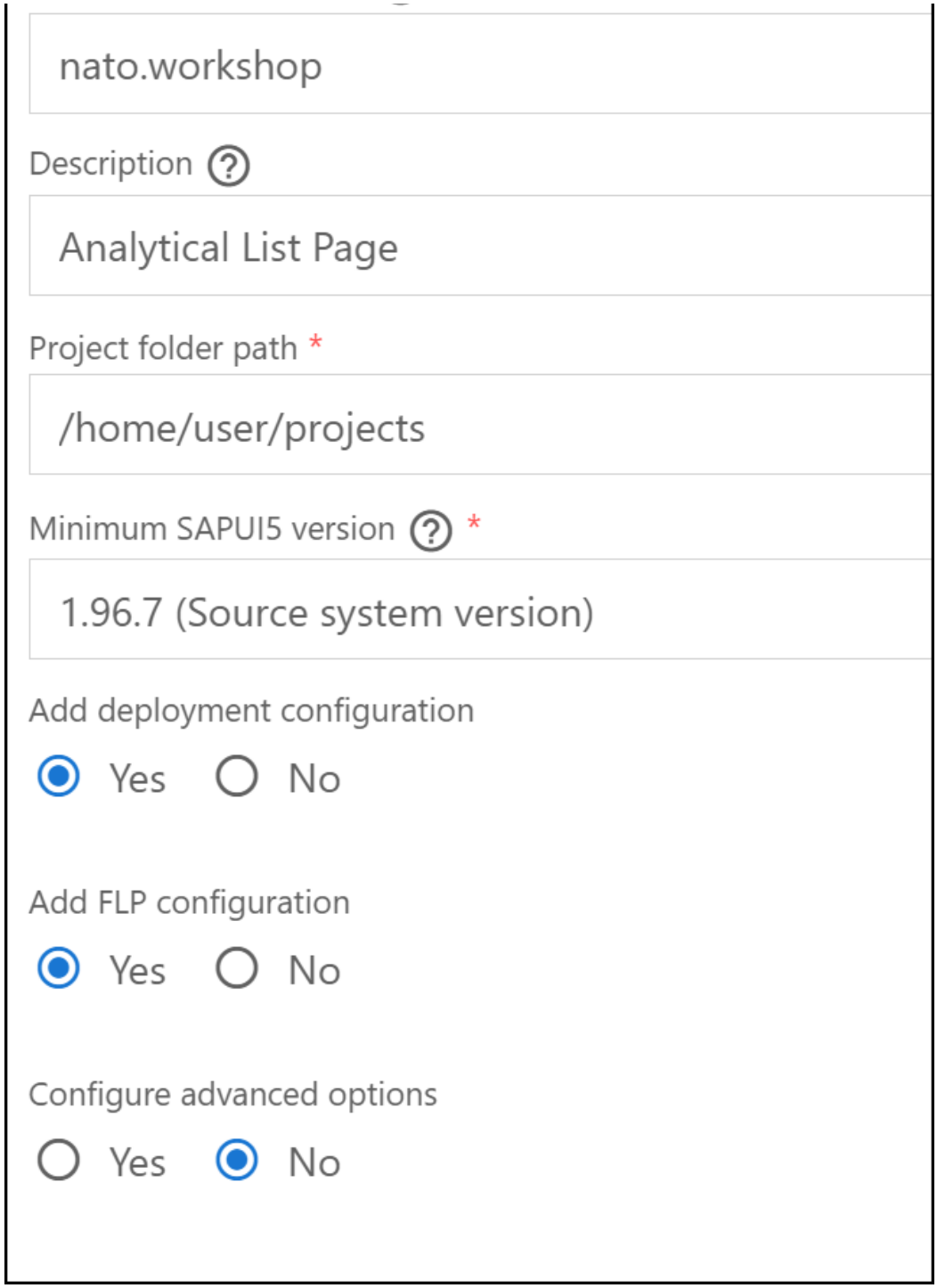


- Entity Selection

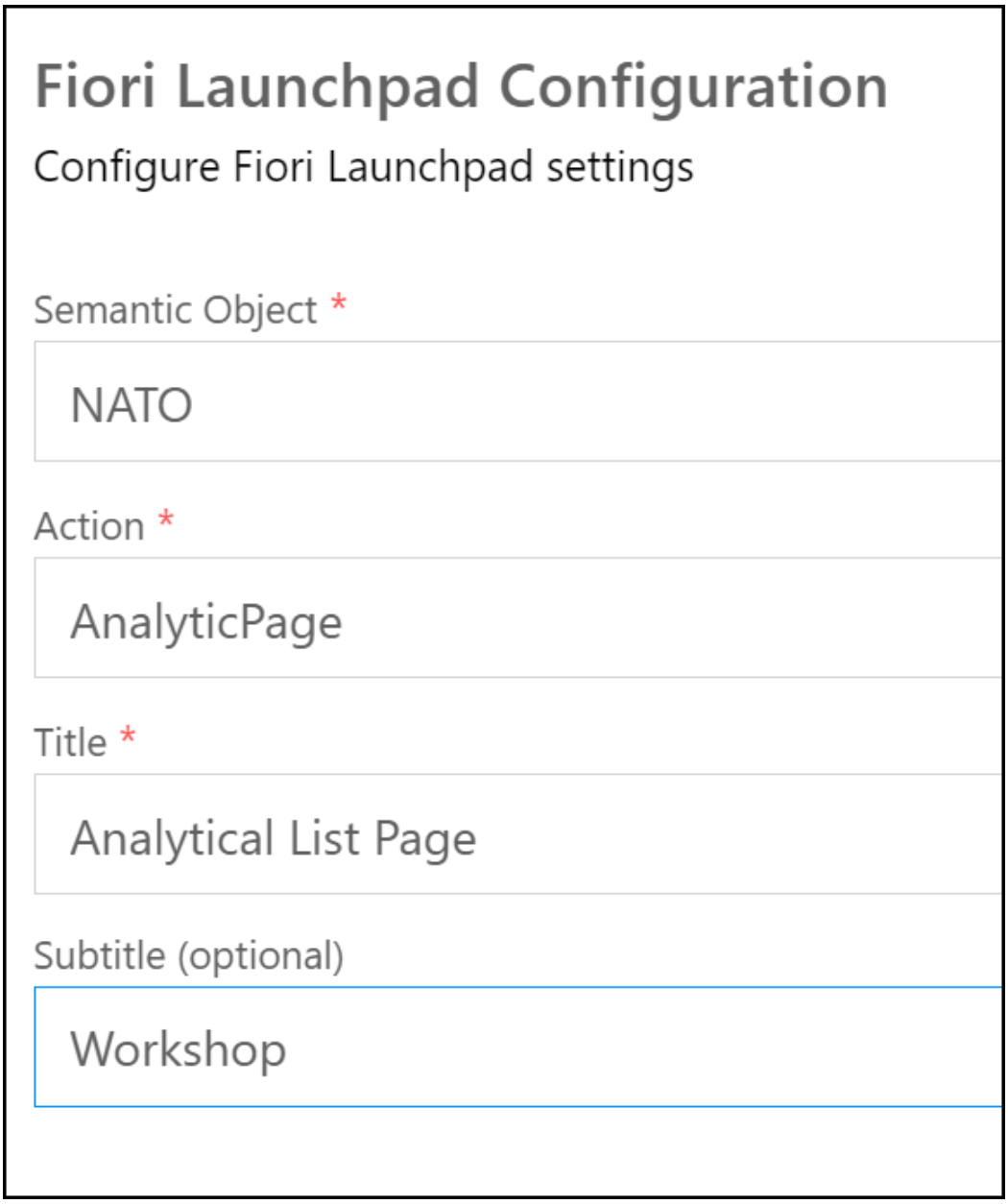


- Project Attributes

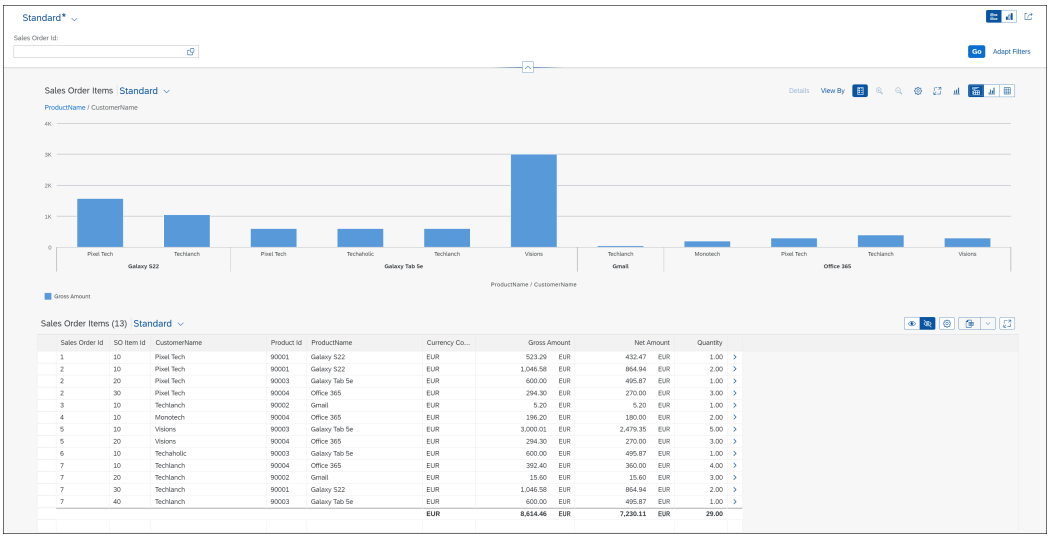



nato.workshop
Description
Analytical List Page
Project folder path *
/home/user/projects
Minimum SAPUI5 version *
1.96.7 (Source system version)
Add deployment configuration
Yes
No
Add FLP configuration
Yes
No
Configure advanced options
Yes
No

- Fiori Launchpad Configuration



- Preview the Application



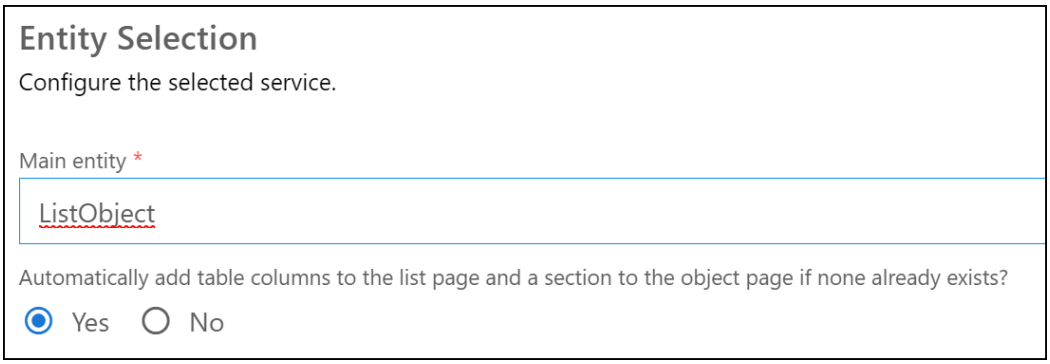
Extend Fiori Elements List Report Object Page Application

Task 1: Create a Fiori elements Extension application **

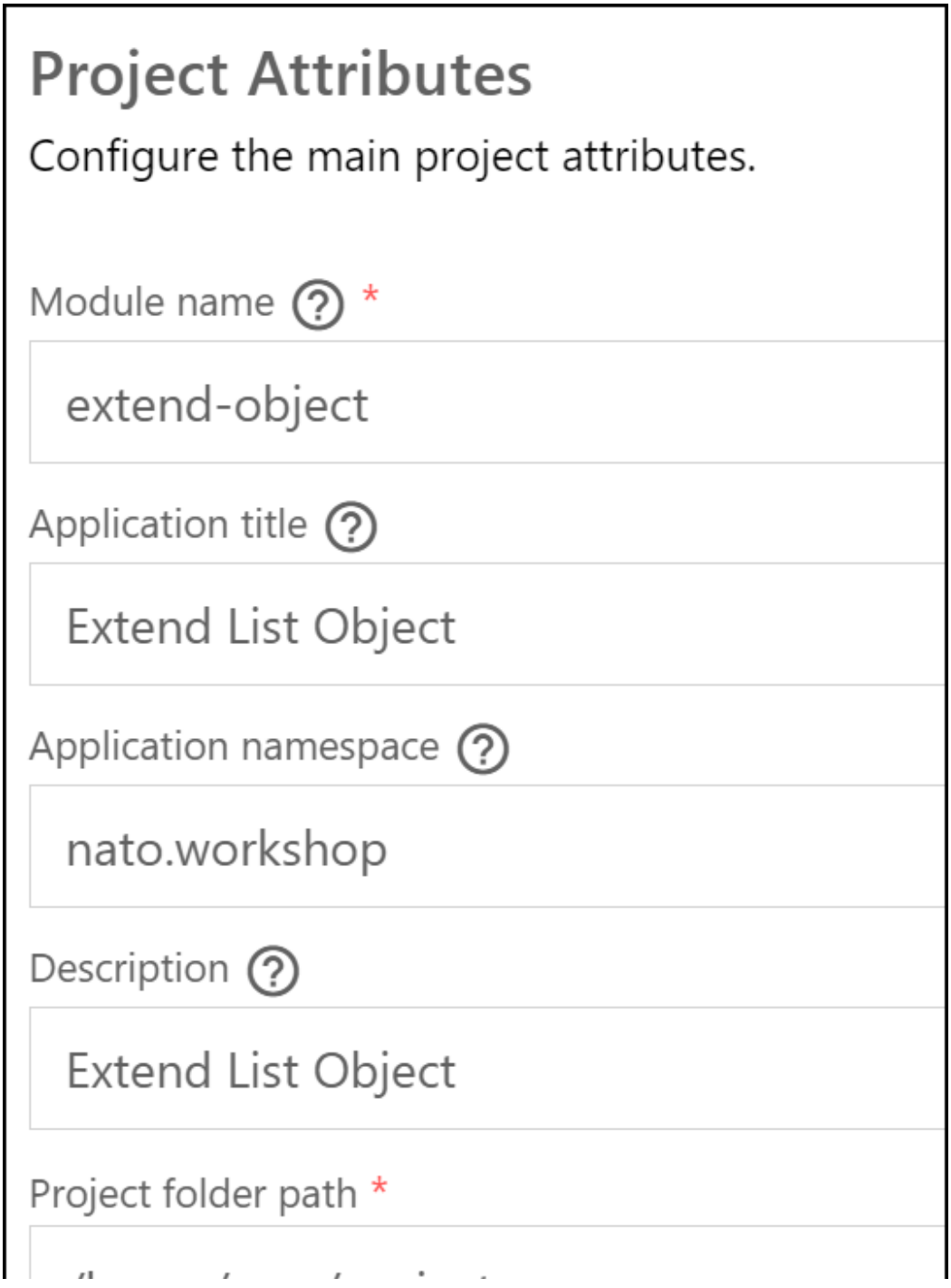
Create a new Fiori Application using the Template Wizard for a *List Report*

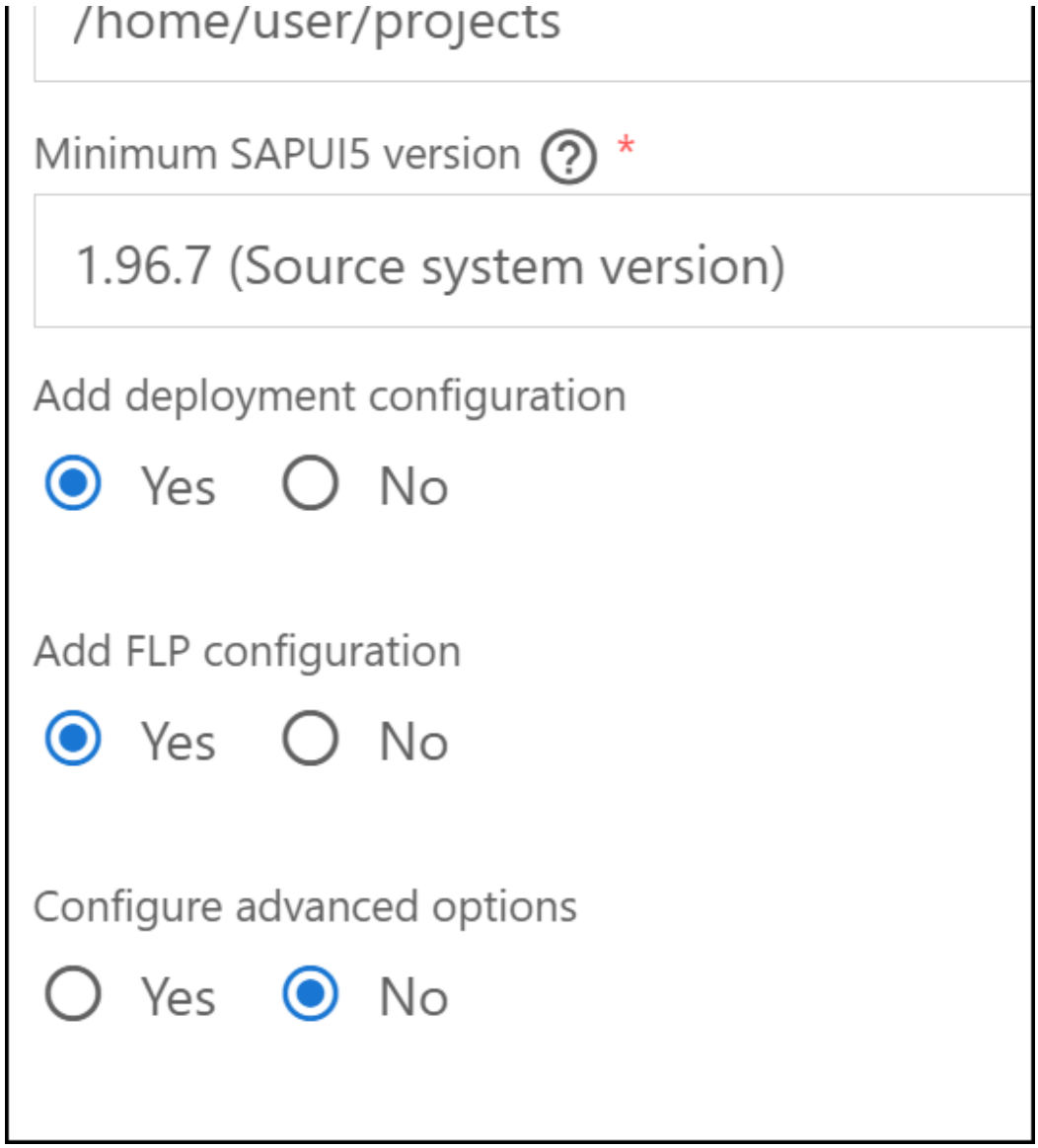
Field	**Value**
Data source	Connect to a System
System	abap-cloud-default_xx-dev (BTP)
Service	ZUI_WKSP_##
Main entity	ListObject
Module name	extend-object
Application title	Extend List Object
Application namespace	nato.workshop
Description	Extend List Object
Project folder path	/home/user/projects
Add deployment configuration	Yes
Add FLP configuration	Yes
Deployment Target	Cloud Foundry
Destination name	abap-cloud-default_xx(SCP)
Add application to managed application router	Yes
Semantic Object	NATO
Action	ExtendList
Title	Extend List Object
Subtitle	Workshop

- Entity Selection

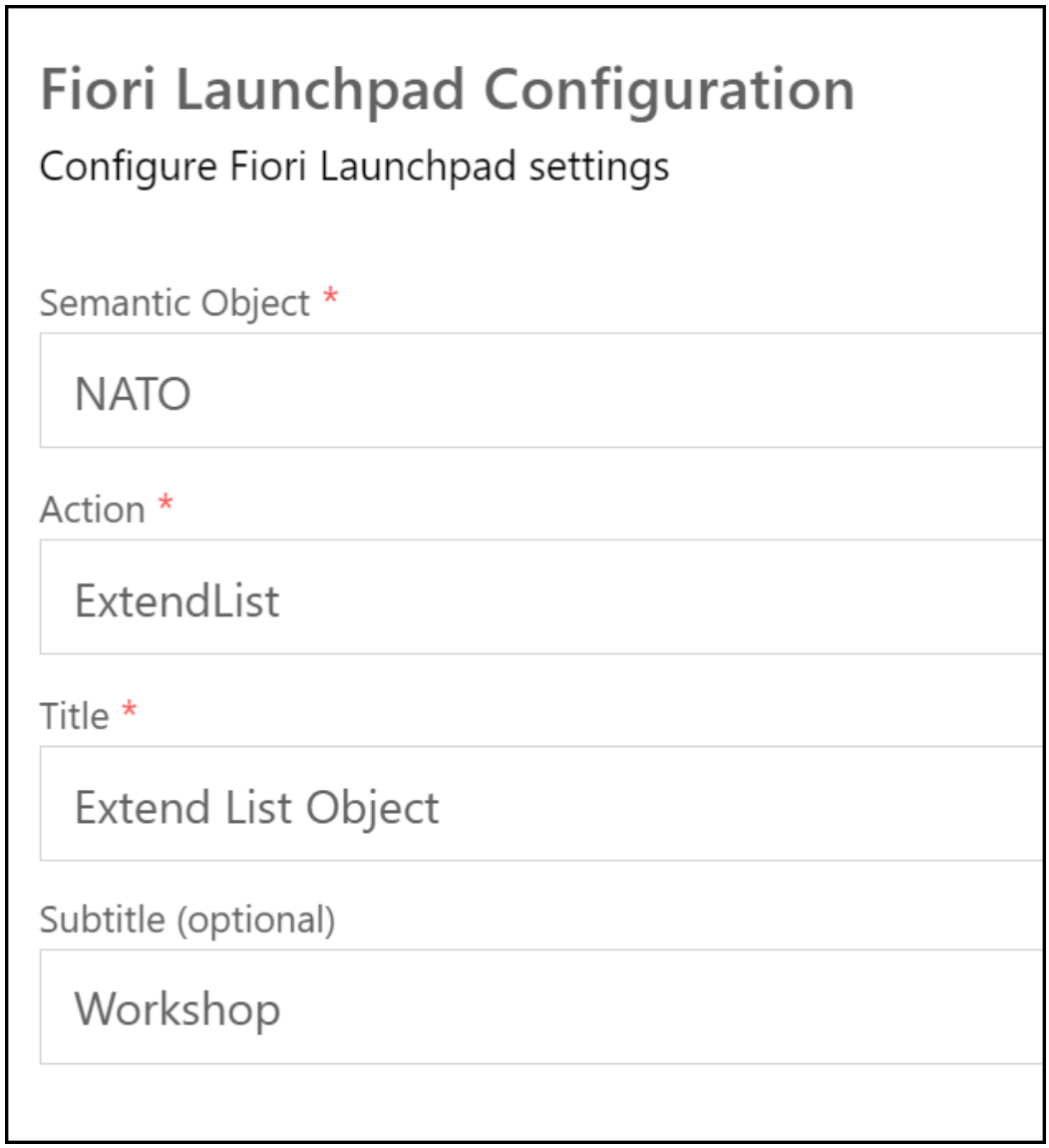


- Project Attributes

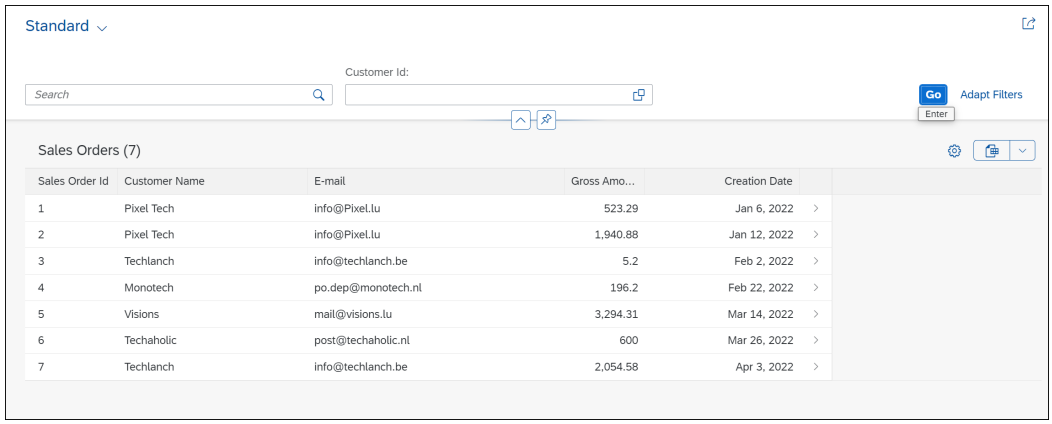



/home/user/projects
Minimum SAPUI5 version *
1.96.7 (Source system version)
Add deployment configuration
Yes
No
Add FLP configuration
Yes
No
Configure advanced options
Yes
No

- Fiori Launchpad Configuration

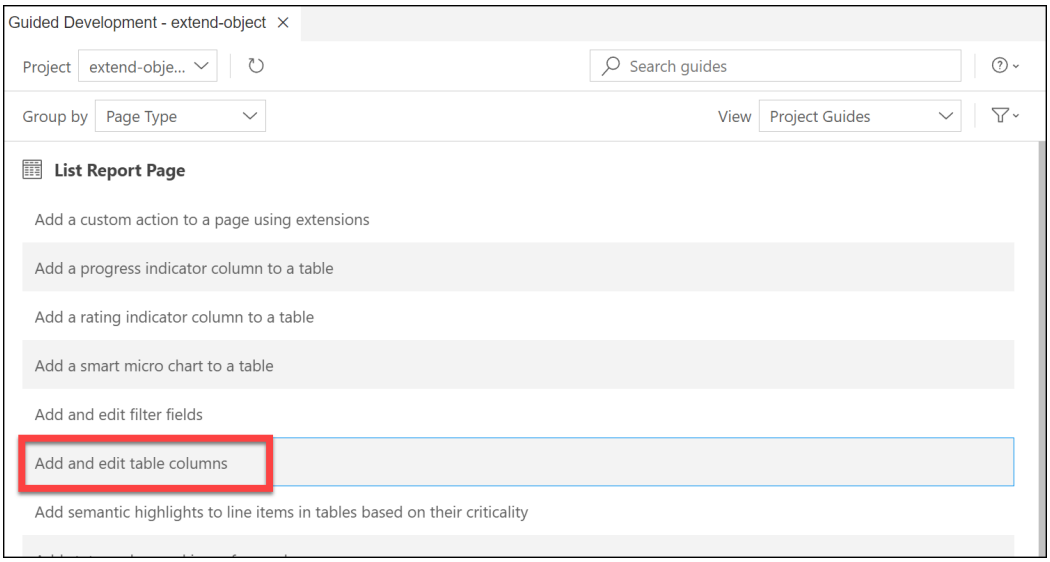


- Preview Application

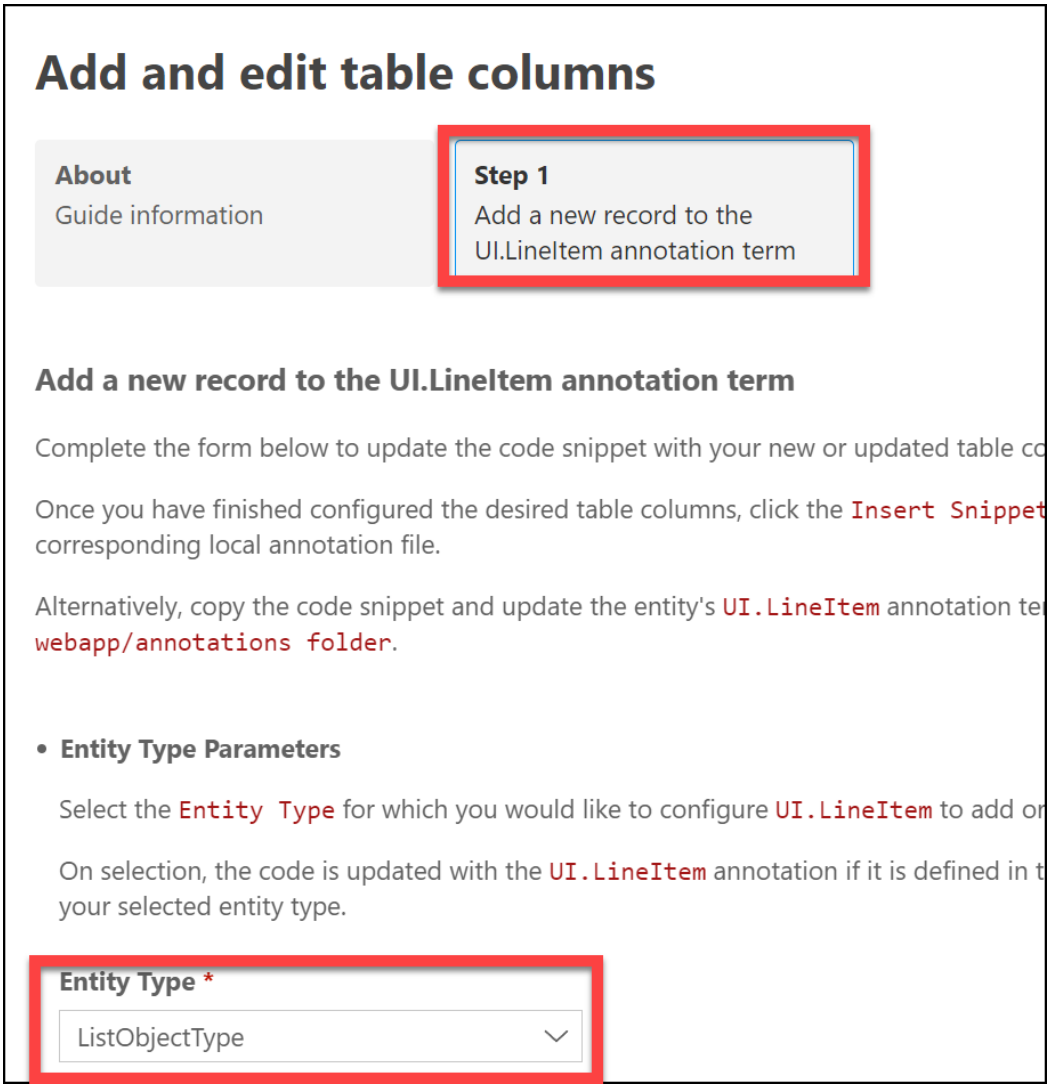


Task 2: Add extra column to Table

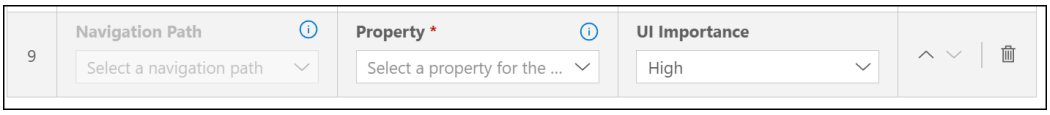
- Start the *Guided Development* and select **Add and edit table columns**



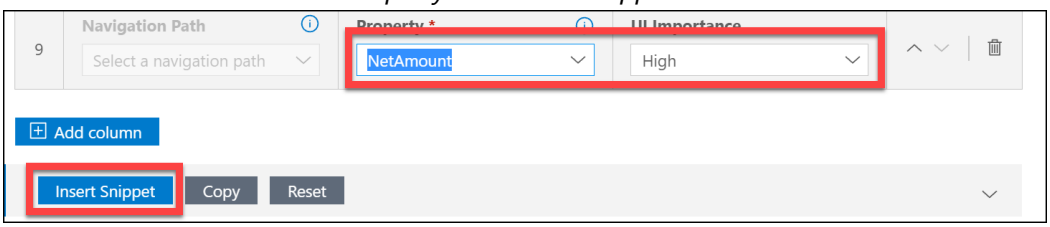
- Select **ListObjectType** as the *Entity Type*



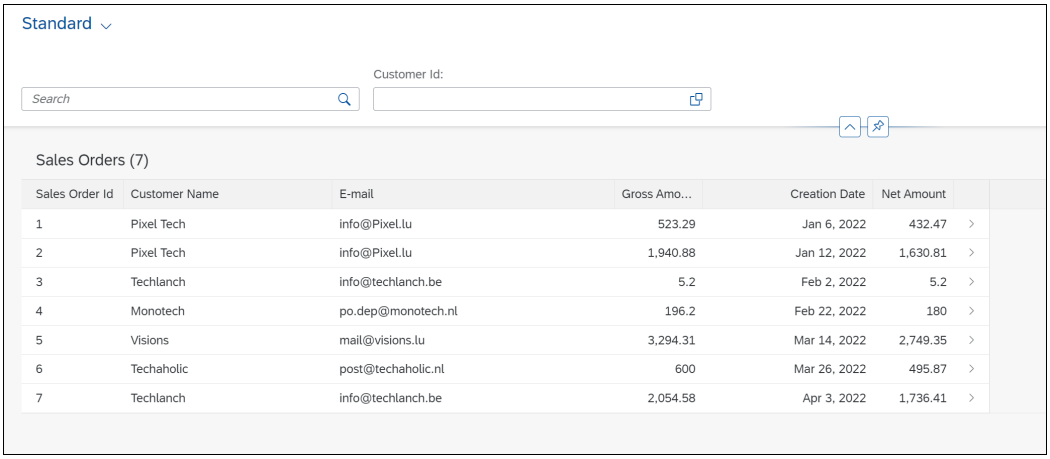
- Press button *Add Column*



- Enter **NetAmmount** as the *Property* and *Insert Snippet*

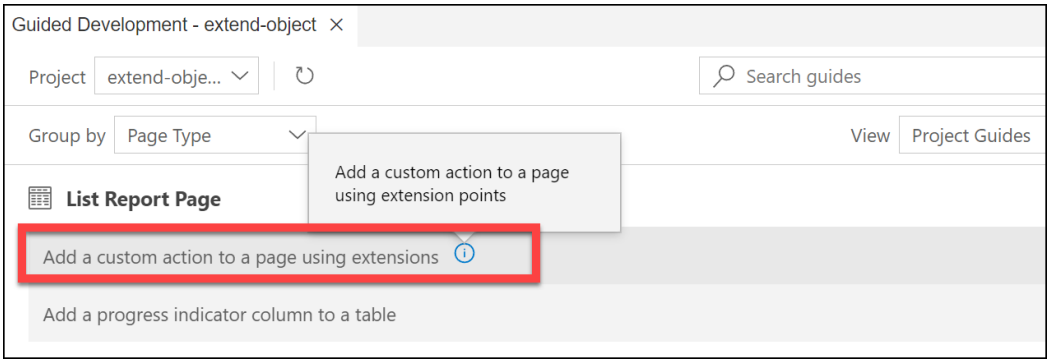


- Preview Application

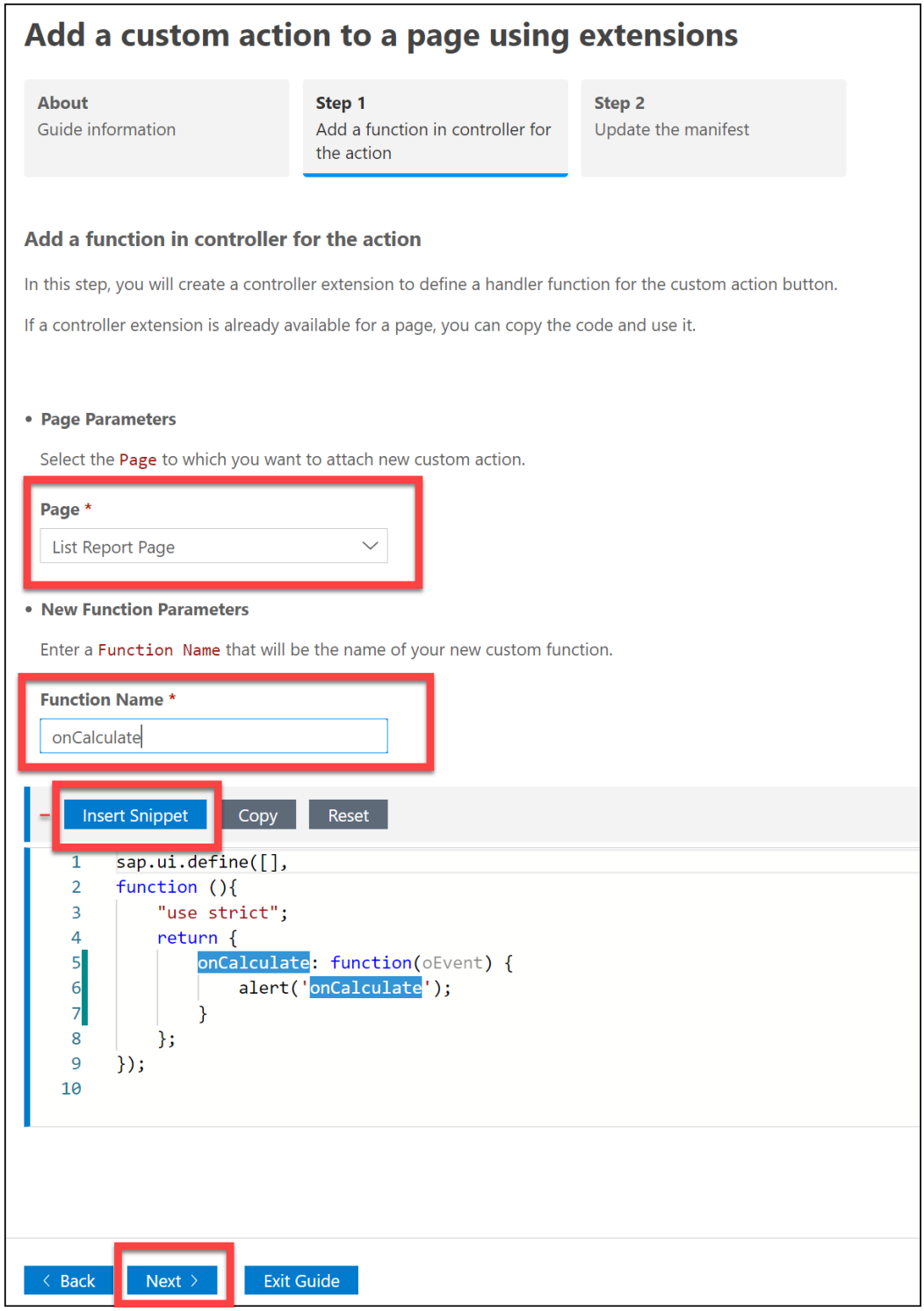


Task 3: Add a custom Action

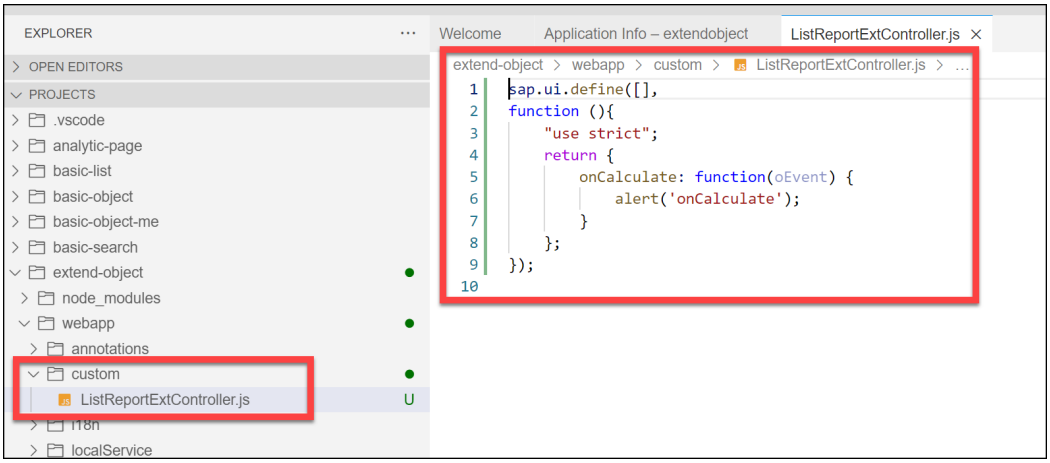
- Start the *Guided Development* and select **Add a custom action to a page using extensions**



- *Start Guide*
- Select **List Report Page** in *Page*, enter **onCalculate** as *Function Name* and Press *Insert Snippet* followed by *Next*

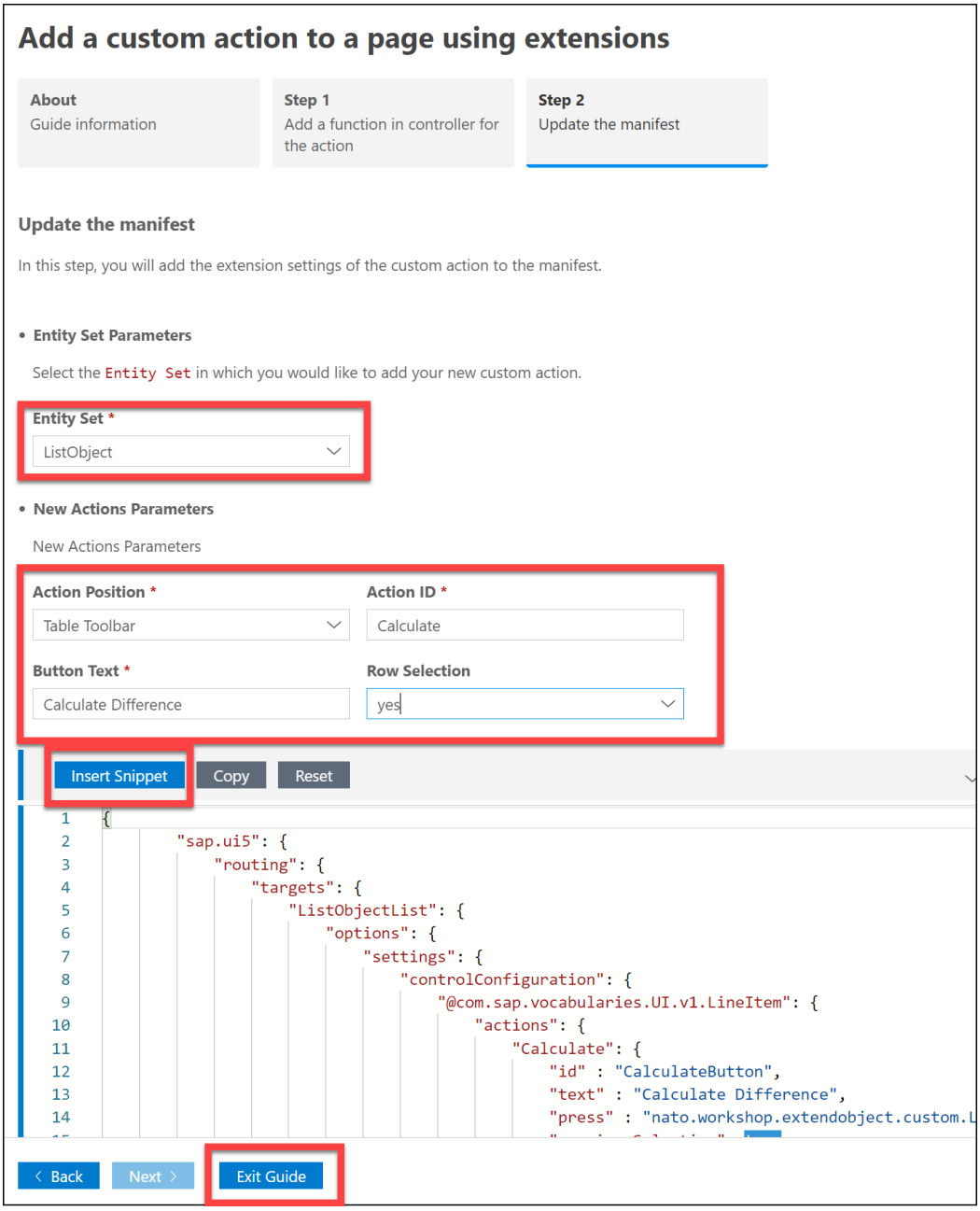


- This will add a new *folder* **custom** and new *file* **ListReportExtController.js** with a template snippet for the javascript code



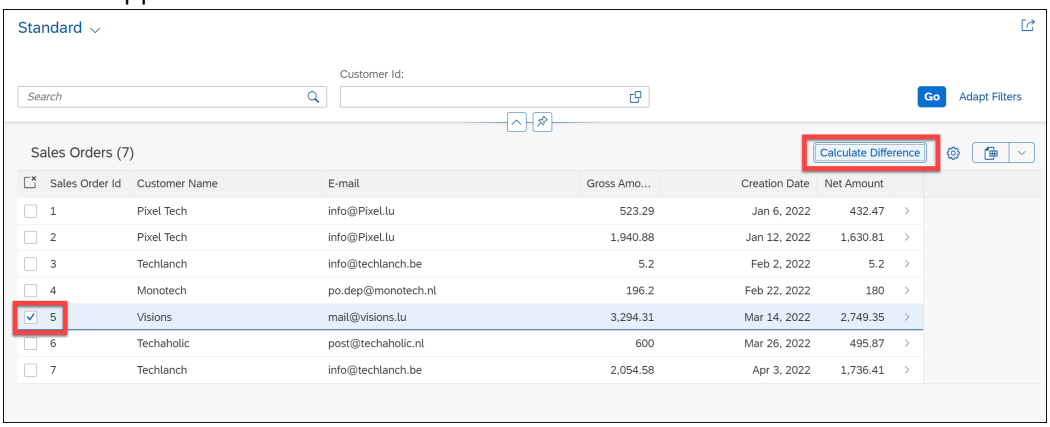
- At the next step set the following parameters, *Insert Snippet* and *Exit Guide*

Name	Value
Entity Set	ListObject
Action Position	Table Toolbar
Action ID	Calculate
Button Text	Calculate Difference
Row Selection	yes

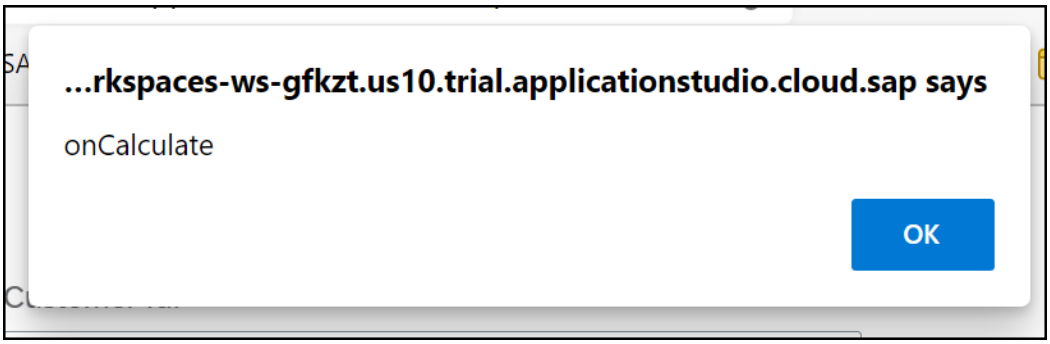

Add a custom action to a page using extensions
About
Guide information
Step 1
Add a function in controller for the action
Step 2
Update the manifest
Update the manifest
In this step, you will add the extension settings of the custom action to the manifest.
Entity Set Parameters
Select the Entity Set in which you would like to add your new custom action.
Entity Set *
ListObject
New Actions Parameters
New Actions Parameters
Action Position *
Table Toolbar
Action ID *
Calculate
Button Text *
Calculate Difference
Row Selection
yes
Insert Snippet
Copy
Reset
{
"sap.ui5": {
"routing": {
"targets": {
"ListObjectList": {
"options": {
"settings": {
"controlConfiguration": {
"@com.sap.vocabularies.UI.v1.LineItem": {
"actions": {
"Calculate": {
"id" : "CalculateButton",
"text" : "Calculate Difference",
"press" : "nato.workshop.extendobject.custom.L
Back
Next
Exit Guide

- Manifest.json file is updated

- Preview Application

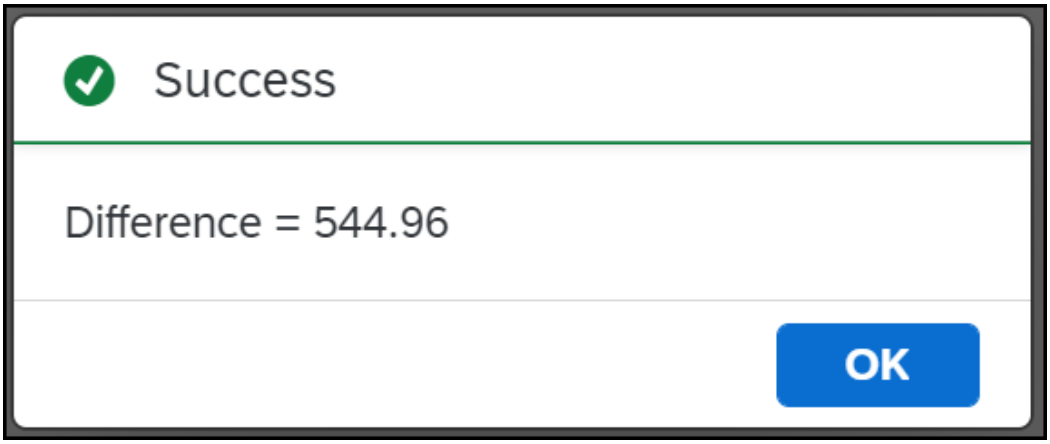


- Select a line and press the **Calculate Difference** button



- Now implement the javascript template in the new **ListReportExtController.js**, Calculate the difference between de *GrossAmount* and *NetAmount* and show it using a MessageBox

```
sap.ui.define([
    "sap/m/MessageBox"
],
function (MessageBox){
    "use strict";
    return {
        onCalculate: function(oEvent) {
            let nGrossAmount =
parseFloat(this.getSelectedContexts()[0].getObject().GrossAmount);
            let nNetAmount = parseFloat(this.getSelectedContexts()
[0].getObject().NetAmount);
            let nDiff = nGrossAmount - nNetAmount;
            MessageBox.success("Difference = " + nDiff);
        }
    };
});
```

- 

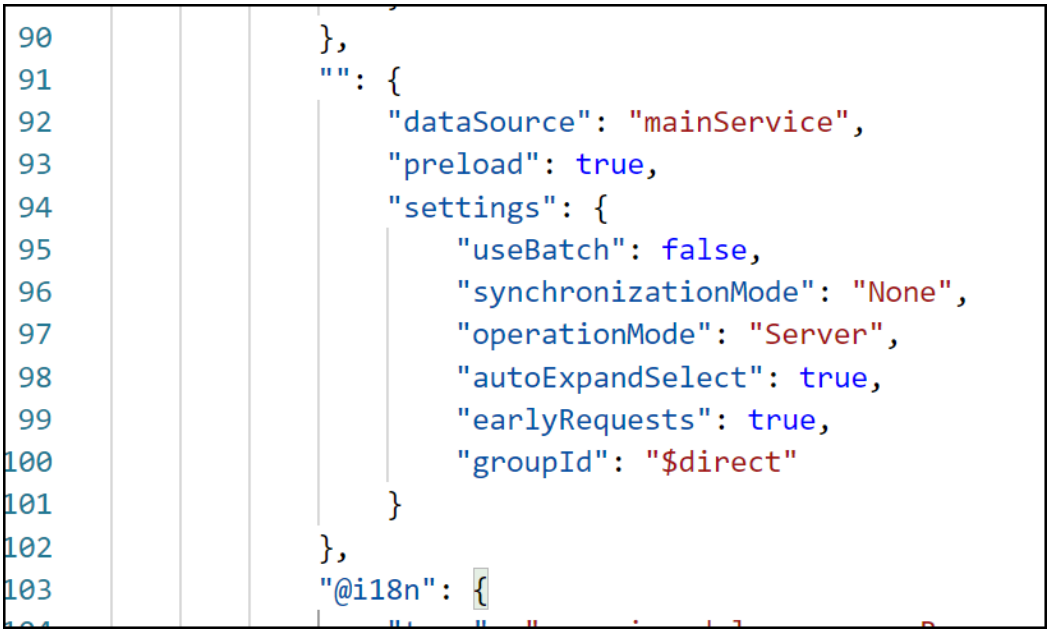
Extend Fiori Elements Overview Page Application

Task 1: Create a new Extension application CDS/OData V2

- Create a new Fiori Application using the Template Wizard for a *Overview Page*

Field	**Value**
Data source	Connect to a System
System	abap-cloud-default_xx-dev (BTP)
Service	ZUI_WKSP_##_V2
Main entity	OverviewPageType
Module name	extend-ovp
Application title	Extend Overview Page
Application namespace	nato.workshop
Description	Extend Overview Page
Project folder path	/home/user/projects
Add deployment configuration	Yes
Add FLP configuration	Yes
Deployment Target	Cloud Foundry
Destination name	abap-cloud-default_xx(SCP)
Add application to managed application router	Yes
Semantic Object	NATO
Action	ExtendOVP
Title	ExtendOVP
Subtitle	Workshop

- Open the manifest.json file and add **"useBatch": false,** to the *settings* part of the *mainModel*

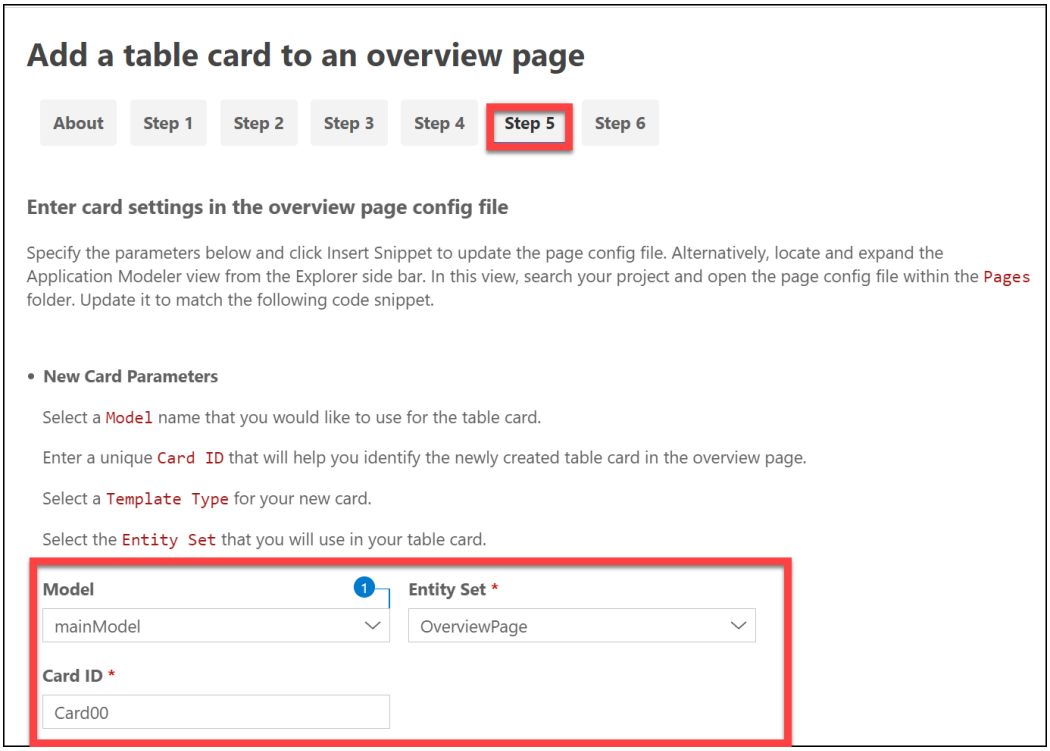


```
90          },
91          "": {
92              "dataSource": "mainService",
93              "preload": true,
94              "settings": {
95                  "useBatch": false,
96                  "synchronizationMode": "None",
97                  "operationMode": "Server",
98                  "autoExpandSelect": true,
99                  "earlyRequests": true,
100                 "groupId": "$direct"
101             }
102         },
103         "@i18n": {
```

Task 2: Add a table card

- Start a *Guided Development* for **Add a table card to an overview page**
- In *Step 5* use following **New Card Parameters**

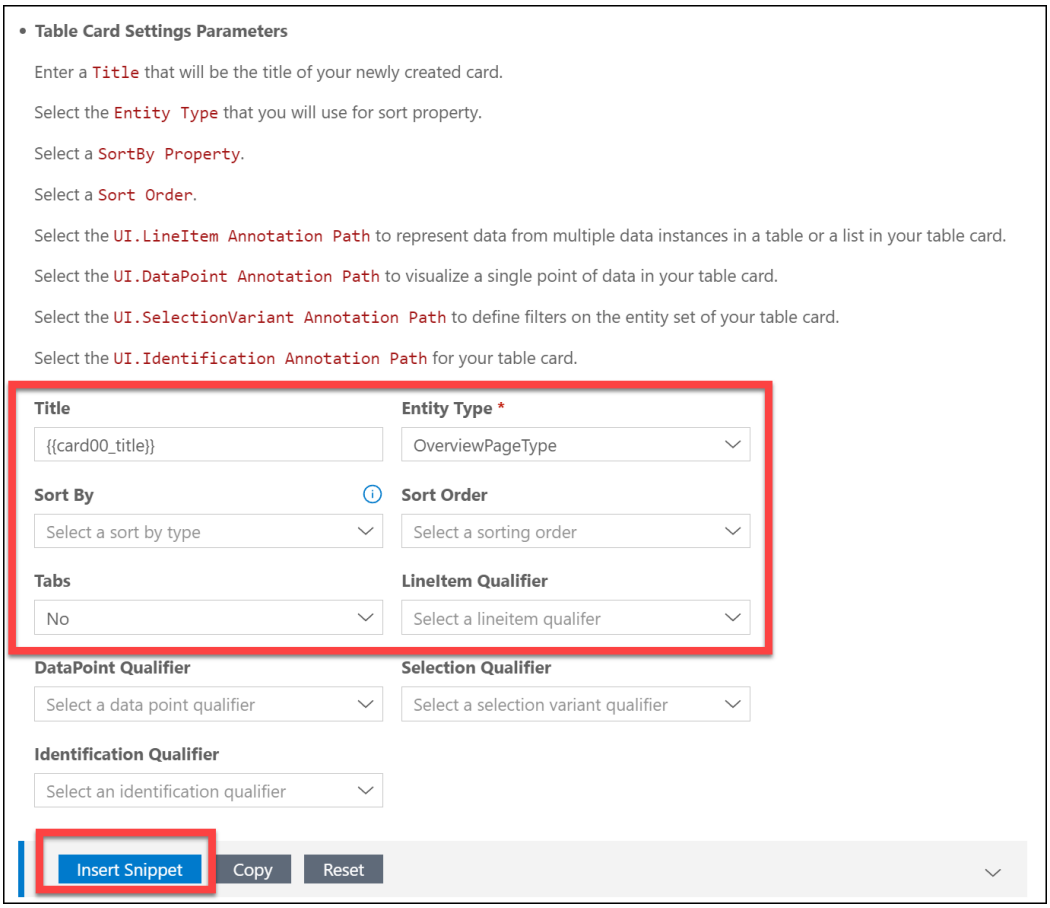
Name	Value
Model	mainModel
Entity Set	OverviewPage
Card ID	Card00



- Add following **Table Card Settings Paramters**

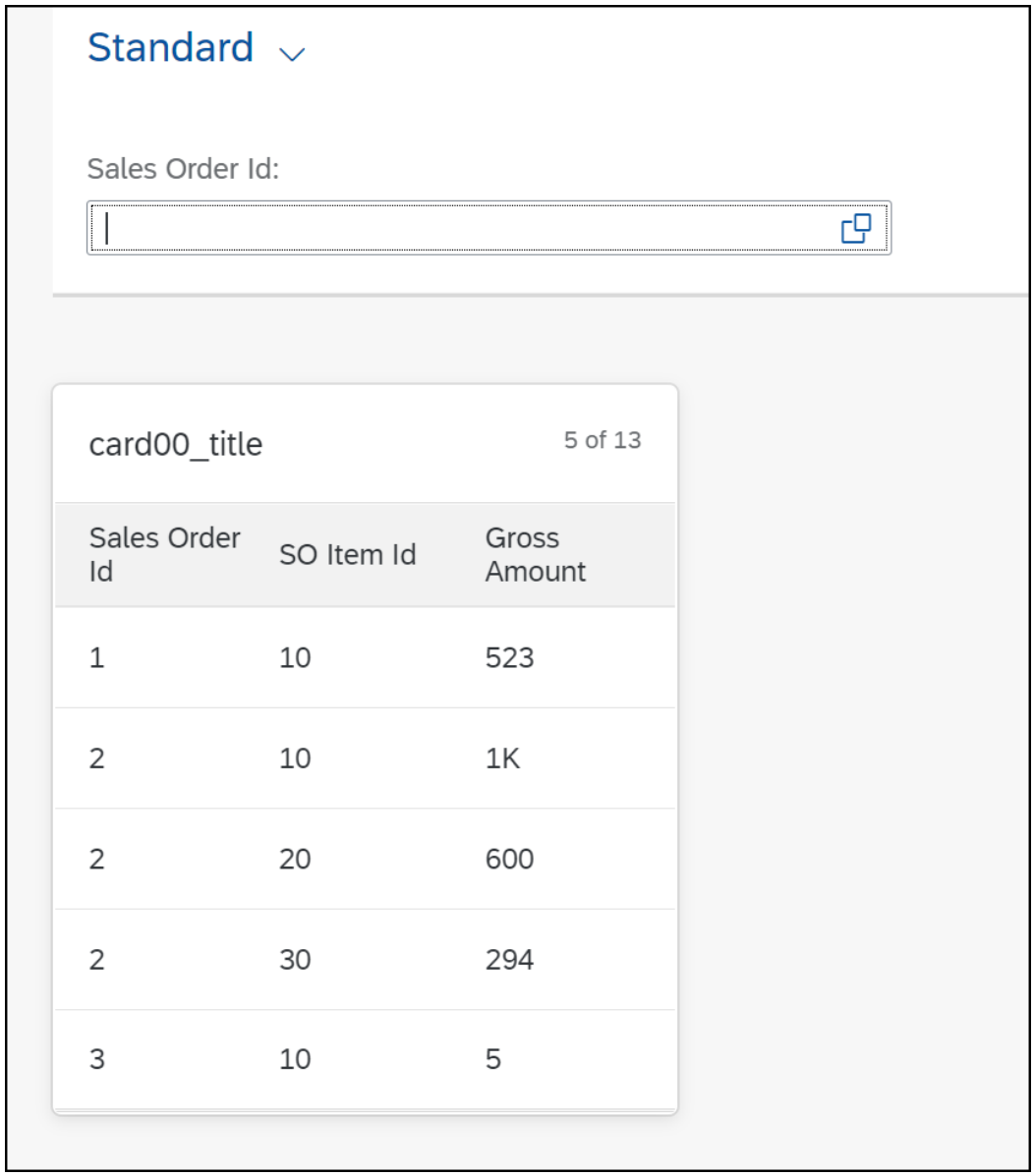
Name	Value
Title	{{card00_title}}
Entity Type	OverviewPageType
Tabs	No

- *Insert Snippet*



- OverviewPage.josn

- Preview Application

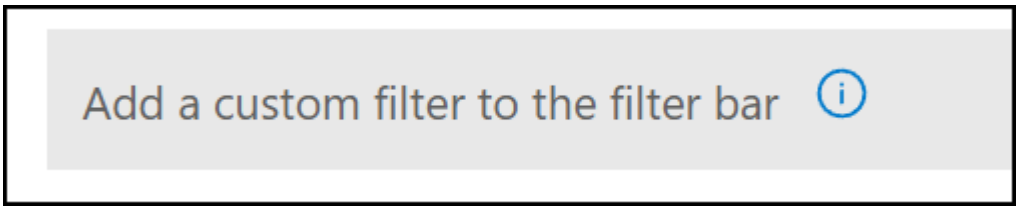


Task 3: Add a custom filter to the filter bar

- Add **card00_title=Recent Sales card01_title=Recent Products sold** to *i18n.properties*

```
#XTIT: Card00_title
card00_title=Recent Sales

#XTIT: Card01_title
card01_title=Recent Products sold
```

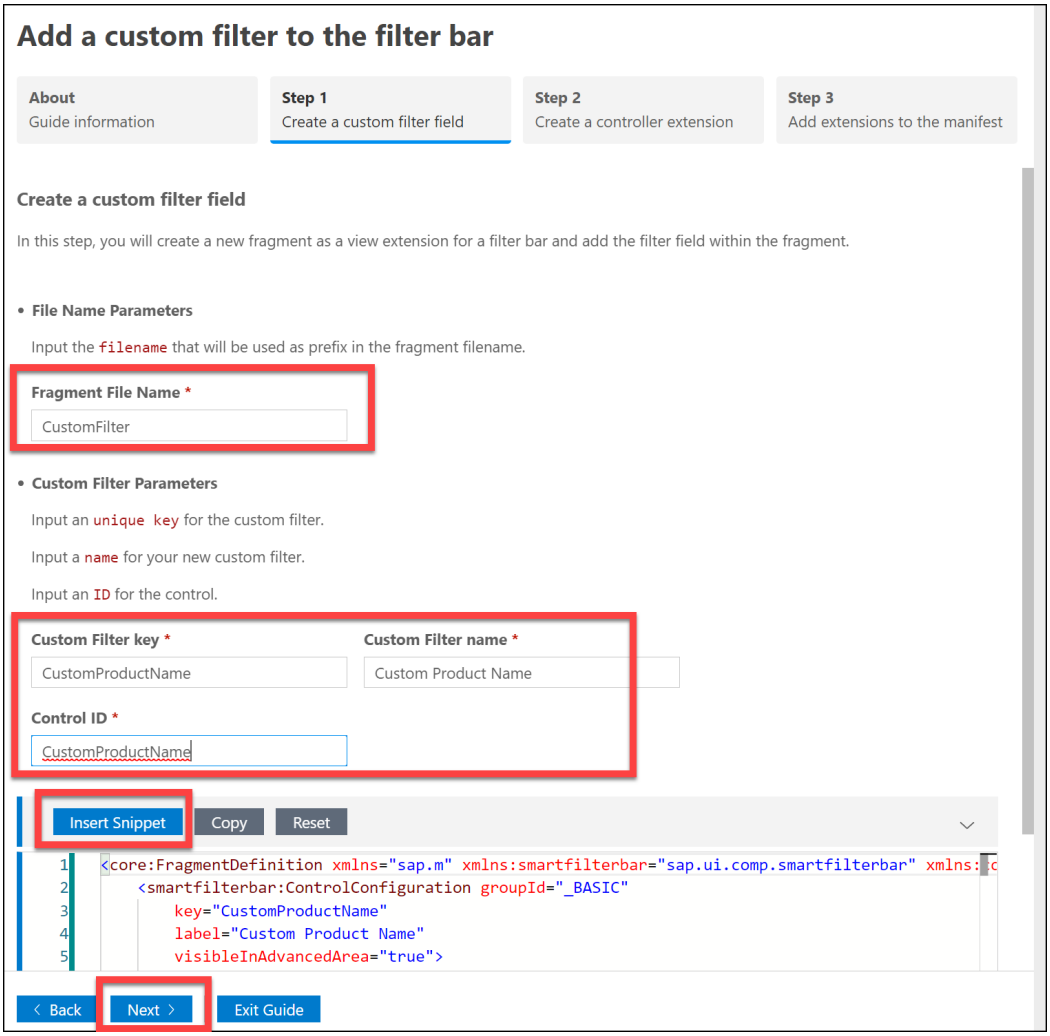
- Start a *Guided Development* for **Add a custom filter to the filter bar**



- In *Step 1* use following parameters

Name	Value
Fragment File Name	CustomFilter
Custom Filter Key	CustomProductName
Custom Filter Name	Custom Product Name
Control ID	CustomProductName

- *Insert Snippet* and *Next*

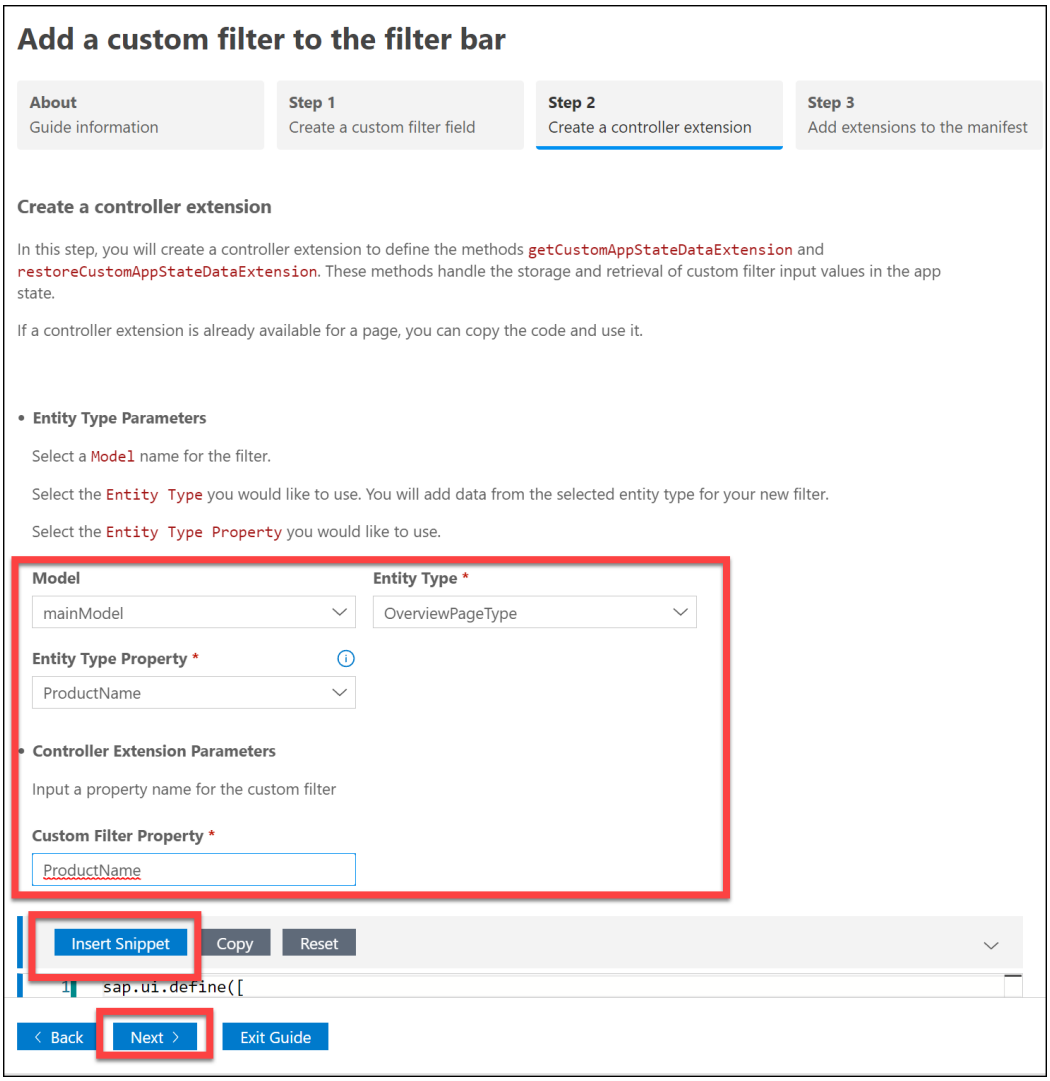


- This creates a new folder **ext/fragments** with file **CustomFilter.fragment.xml**

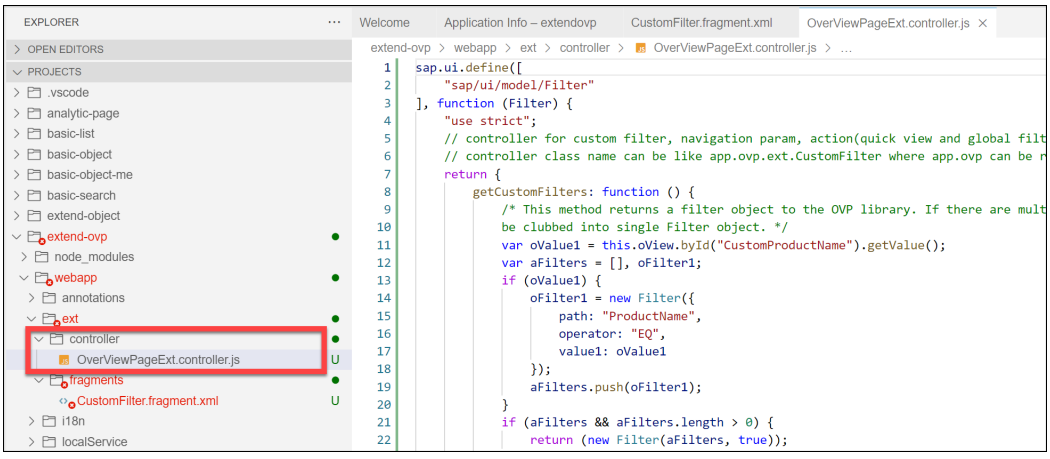
- In *Step 2* use following parameters

Name	Value
Model	mainModel
Entity Type	OverviewPageType
Entity Type Property	ProductName
Custom Filter Property	ProductName

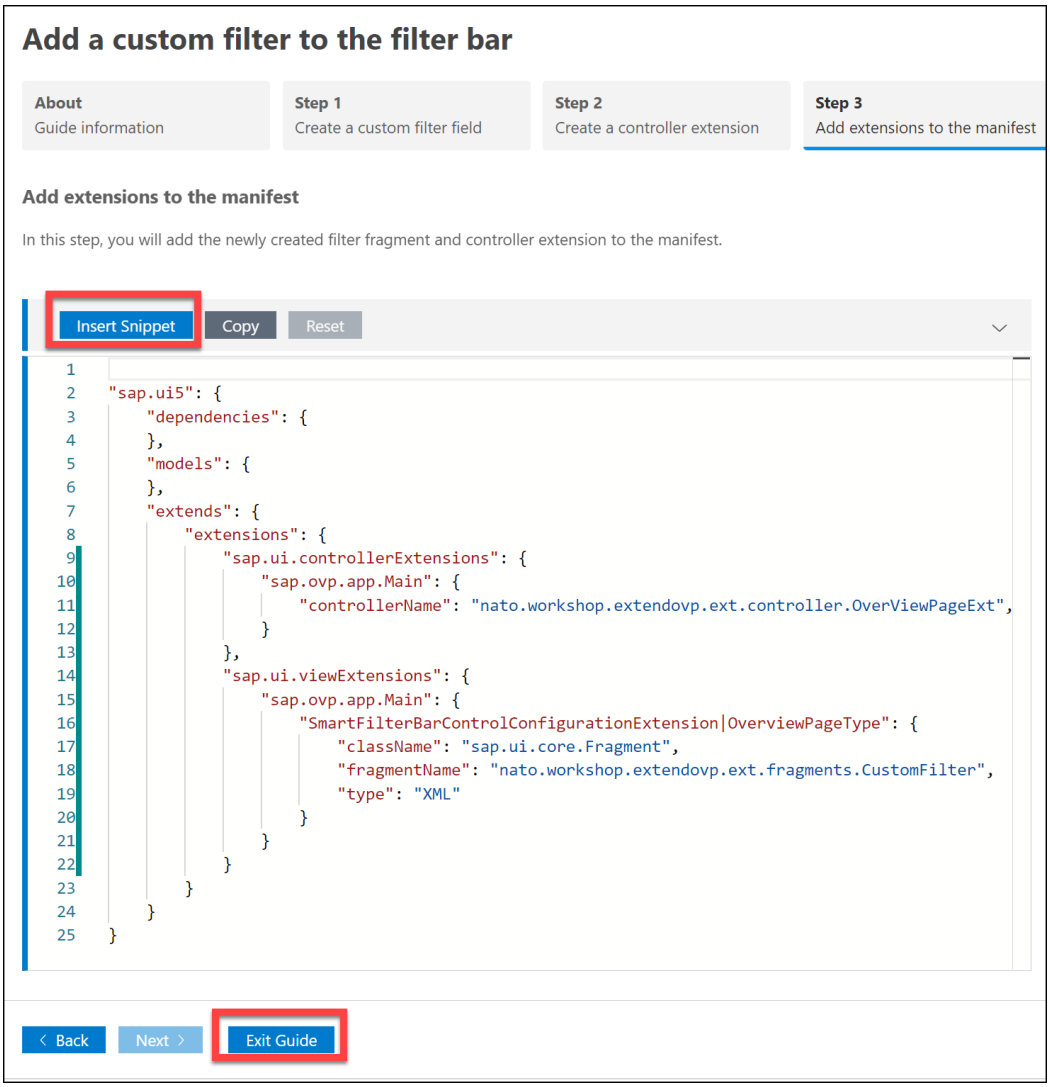
- *Insert Snippet* and *Next*



- This creates a new folder **ext/controller** with file **OverViewPageExt.controller.js**



- In *Step 3 Insert Snippet* and *Exit Guide*



- Implement **CustomFilter.fragment.xml** by adding a *Select*

```
<core:FragmentDefinition xmlns="sap.m"
xmlns:smartfilterbar="sap.ui.comp.smartfilterbar"
xmlns:core="sap.ui.core">
    <smartfilterbar:ControlConfiguration id="CustomFilter"
groupId="_BASIC"
        key="CustomProductName"
        label="Custom Product Name"
        visibleInAdvancedArea="true">
        <smartfilterbar:customControl>
            <Select id="CustomProductName">
                <core:Item id="all" text="All" key="All"/>
                <core:Item id="Gal" text="Galaxy" key="Gal"/>
                <core:Item id="Sub" text="Subscription"
key="Sub"/>
            </Select>
        </smartfilterbar:customControl>
```

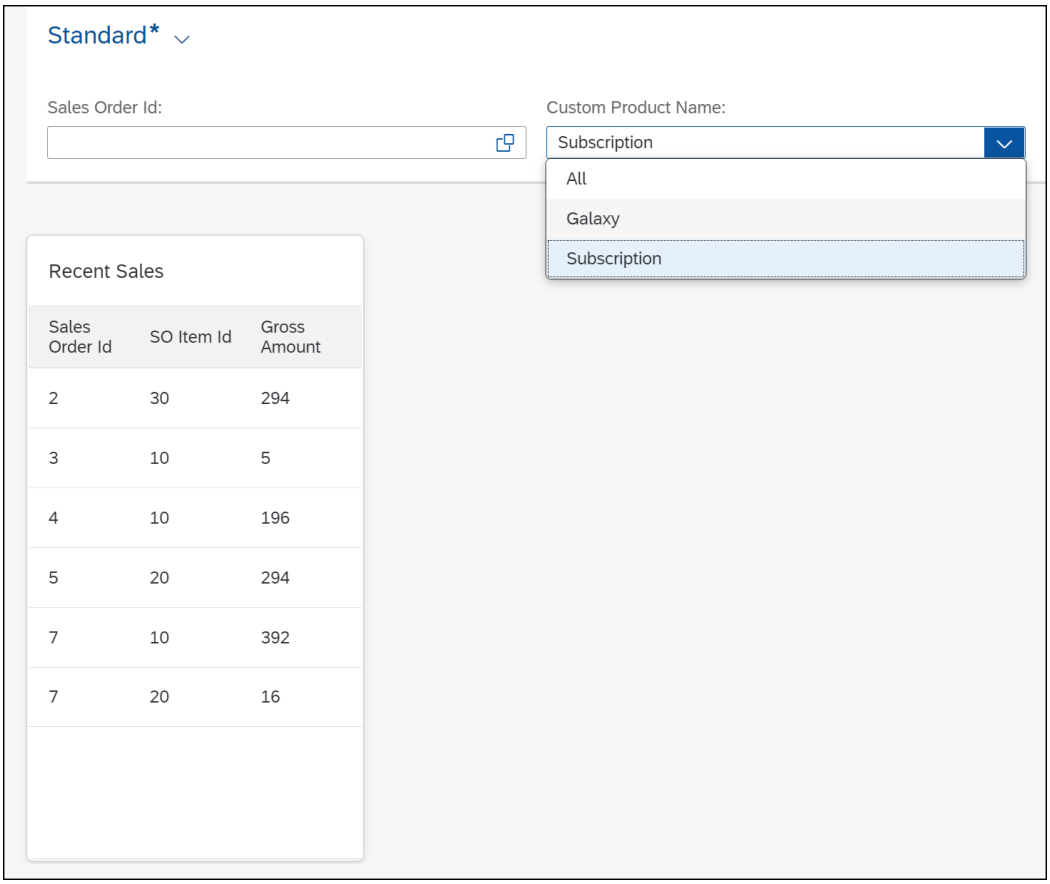
```
    </smartfilterbar:ControlConfiguration>
</core:FragmentDefinition>
```

- Implement **OverViewPageExt.controller.js**

```
sap.ui.define([
    "sap/ui/model/Filter",
    "sap/ui/model/FilterOperator"
], function (Filter,FilterOperator) {
    "use strict";
    // controller for custom filter, navigation param,
action(quick view and global filter), navigation target
    // controller class name can be like app.ovp.ext.CustomFilter
where app.ovp can be replaced with your application namespace
    return {
        getCustomFilters: function () {
            /* This method returns a filter object to the OVP
library. If there are multiple filters, they should
            be clubbed into single Filter object. */
            var sSelectedKey =
this.oView.byId("CustomProductName").getSelectedKey();
            var aFilters = [], oFilter1;

            switch (sSelectedKey){
                case "All":
                    break;
                case "Gal":
                    oFilter1 = new Filter({path:"ProductName",
operator: FilterOperator.StartsWith, value1: "Gal"})
                    aFilters.push(oFilter1);
                    break;
                case "Sub":
                    oFilter1 = new Filter({path:"ProductName",
operator: FilterOperator.NotStartsWith, value1: "Gal"})
                    aFilters.push(oFilter1);
                    break;
                }

            if (aFilters && aFilters.length > 0) {
                return (new Filter(aFilters, true));
            }
        },
        getCustomAppStateDataExtension: function (oCustomData) {
            //the content of the custom field will be stored in
the app state, so that it can be restored later, for example after
a back navigation.
```

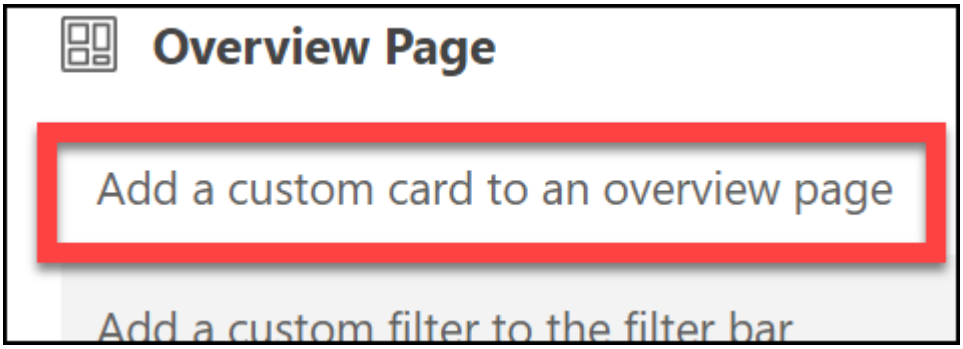
```
            //The developer has to ensure that the content of the
field is stored in the object that is returned by this method.
            if (oCustomData) {
                var oCustomField1 =
this.oView.byId("CustomProductName");
                if (oCustomField1) {
                    oCustomData.ProductName =
oCustomField1.getSelectedKey();
                }
            }
        },
        restoreCustomAppStateDataExtension: function (oCustomData)
{
            //in order to restore the content of the custom field
in the filter bar, for example after a back navigation,
            //an object with the content is handed over to this
method. Now the developer has to ensure that the content of the
custom filter is set to the control
            if (oCustomData) {
                if (oCustomData.ProductName) {
                    var oCustomField1 =
this.oView.byId("CustomProductName");

oCustomField1.setValue(oCustomData.ProductName);
                }
            }
        }
    }
});
```

- Preview Application



Task 4: Add a custom card

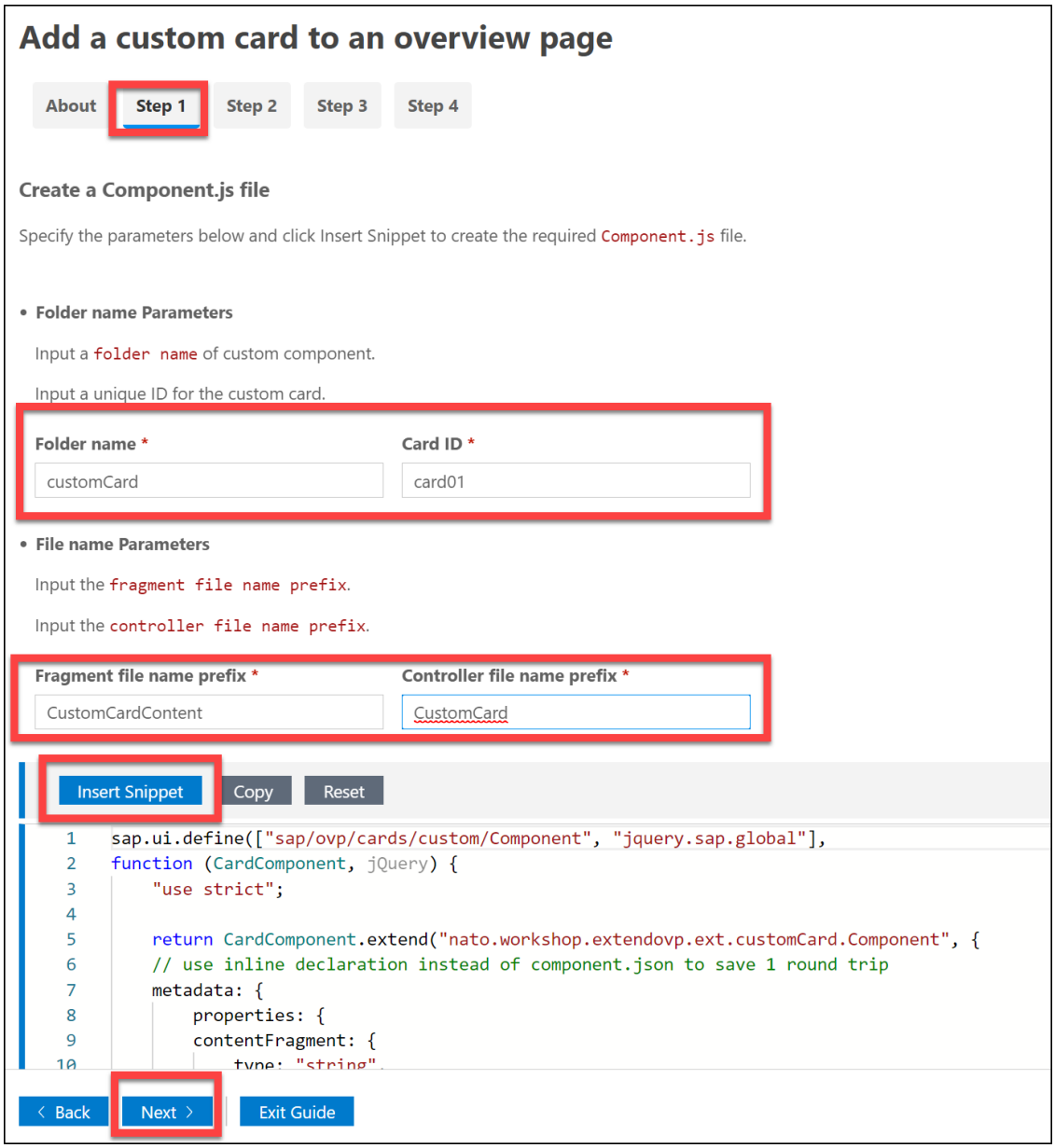
- Start a *Guided Development* for **Add a custom card to an overview page**



- In *Step 1* use following parameters

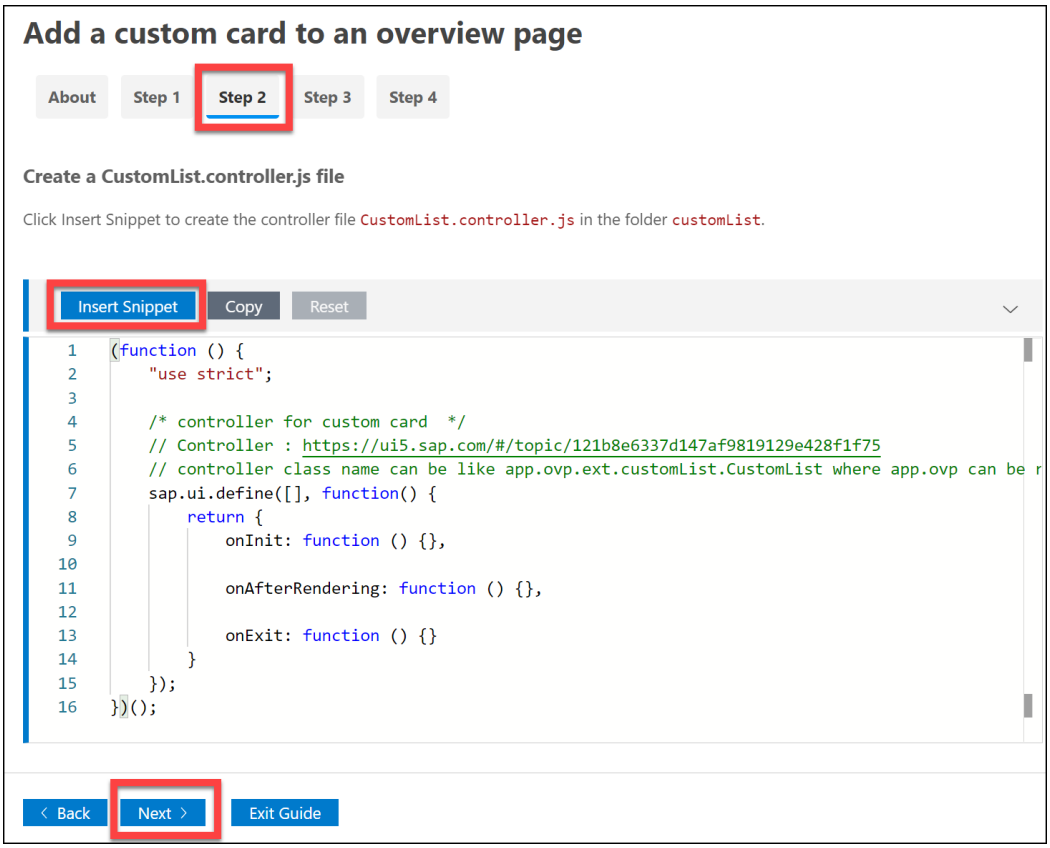
Name	Value
Folder name	customCard
Card ID	card01
Fragment file name prefix	CustomCardContent
Controller file name prefix	CustomCard

Insert Snippet and *Next*

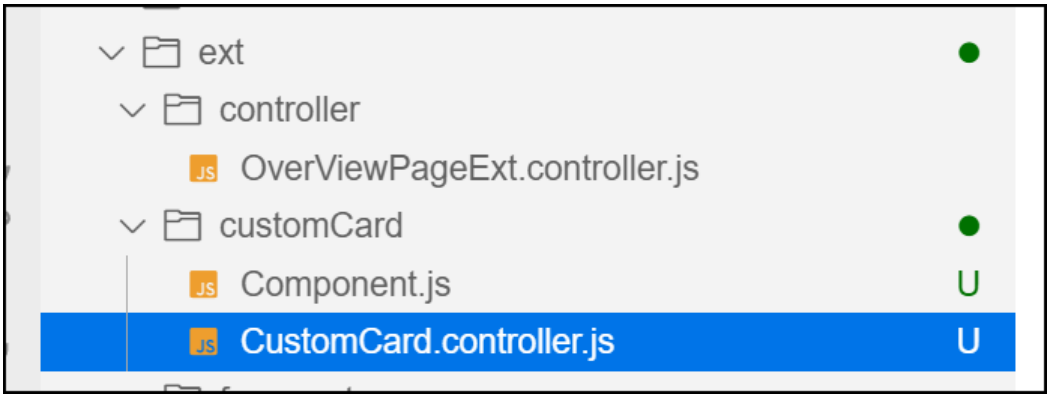


- This creates a new folder **customCard** with file **Component.js**

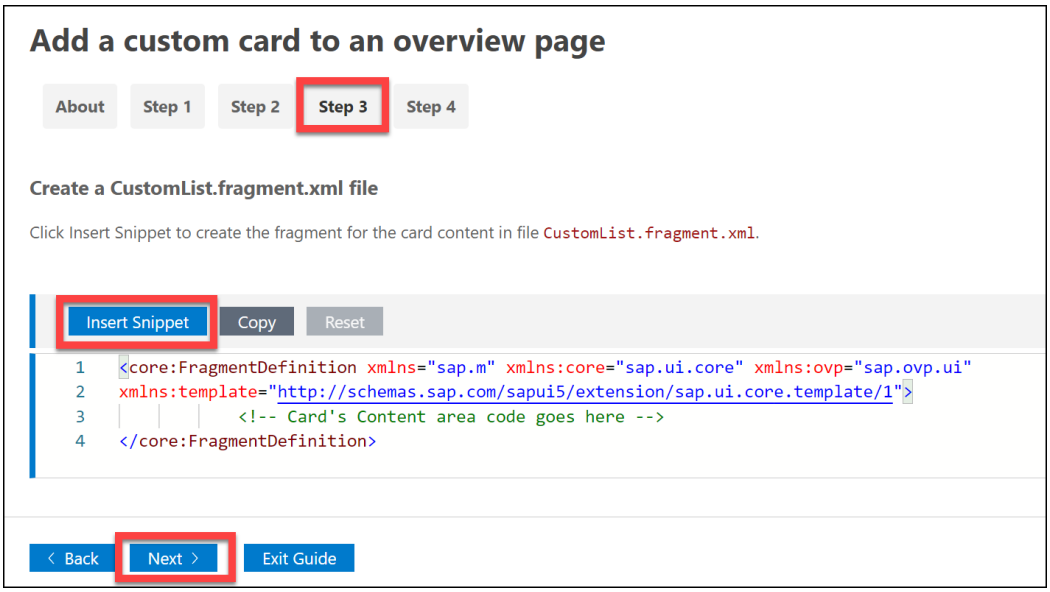
- In *Step 2 Insert Snippet* and *Next*



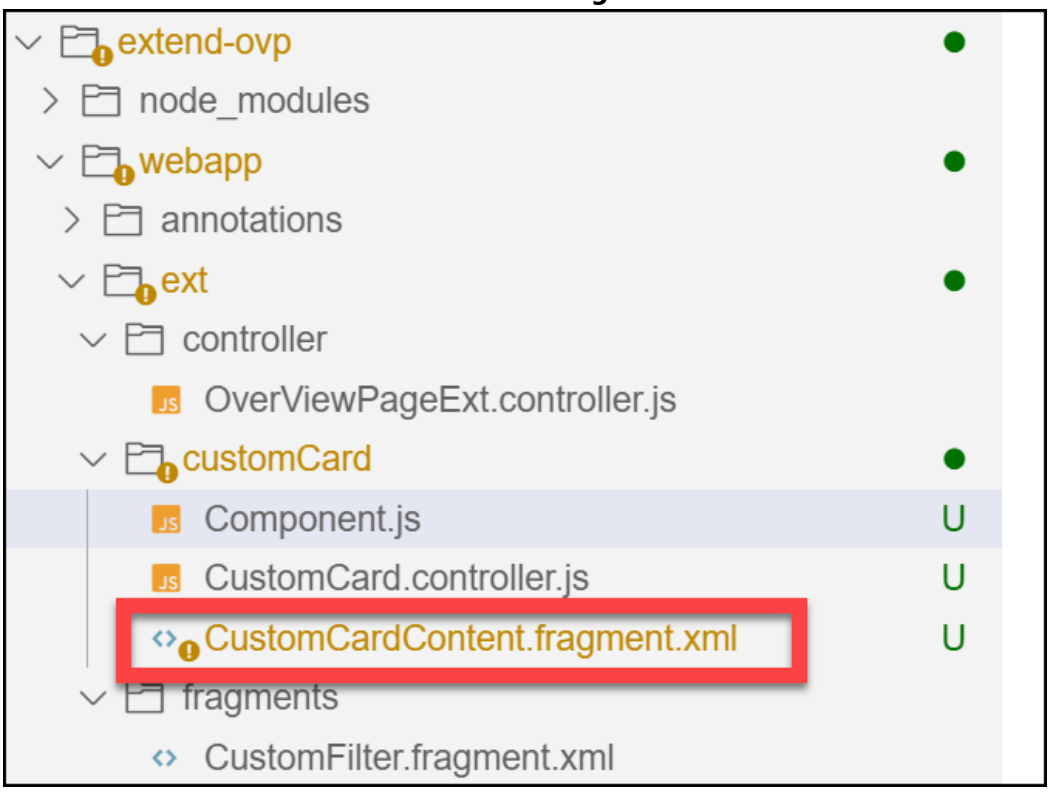
- This creates a new file **CustomCard.controller.js**



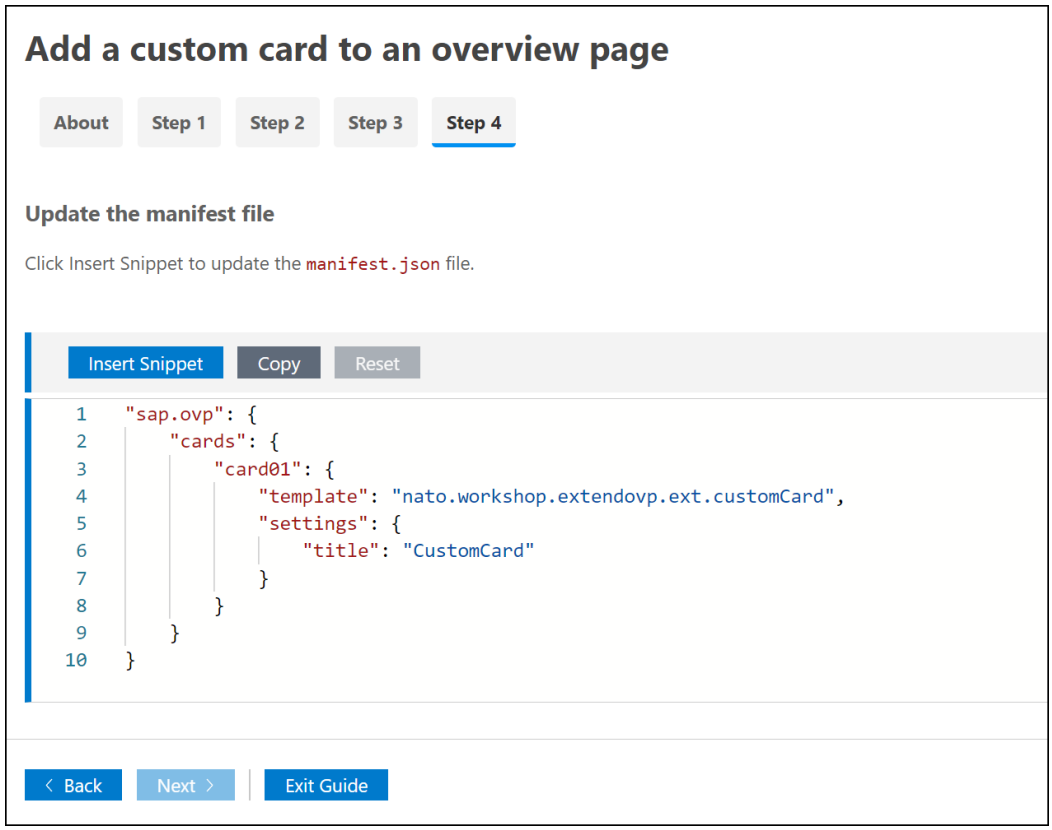
- In *Step 3 Insert Snippet* and *Next*



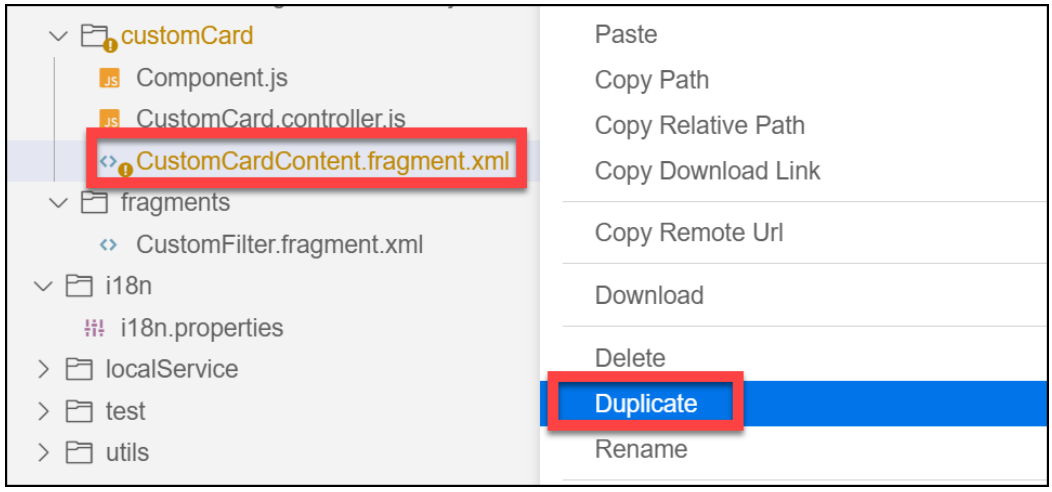
- This creates a new file **CustomCardContent.fragment.xml**



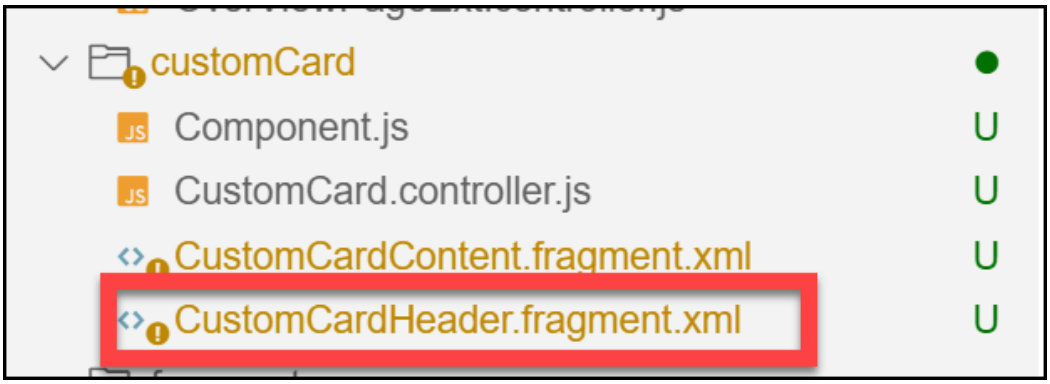
- In *Step 4 Insert Snippet* and *Exit Guide*



- Duplicate the **CustomCardContent.fragment.xml** file to **CustomCardHeader.fragment.xml**, because we also want a header in our custom card



- CustomCardHeader.fragment.xml



- in file **ext/customCard/Component.js** copy the *defaultValue* from **ContentFragment** to **headerFragment** and change it to **CustomCardHeader**

- Implement **CustomCardHeader.fragment.xml**

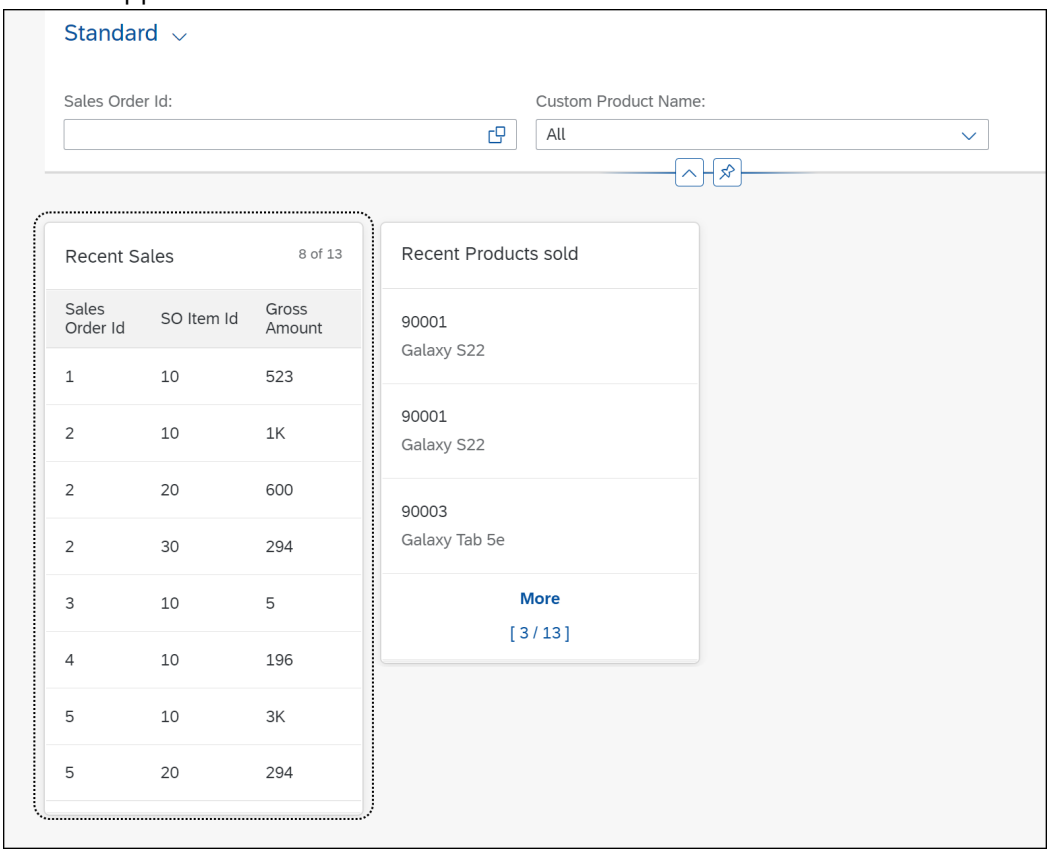
```
<core:FragmentDefinition xmlns="sap.m" xmlns:core="sap.ui.core"
xmlns:ovp="sap.ovp.ui"
xmlns:template="http://schemas.sap.com/sapui5/extension/sap.ui.cor
e.template/1">
    <VBox id="idVbox">
    <Title id="idTitle" text="{i18n>card01_title}"
class="sapUiSmallMargin"/>
    </VBox>
</core:FragmentDefinition>
```

*Implement **CustomCardContent.fragment.xml**

```
<core:FragmentDefinition xmlns="sap.m" xmlns:core="sap.ui.core"
xmlns:ovp="sap.ovp.ui"
```

```
xmlns:template="http://schemas.sap.com/sapui5/extension/sap.ui.cor
e.template/1">
    <List id="idList" items="{ path: '/OverviewPage',
templateShareable: false}"
          growing="true"
          growingThreshold="3">
        <items>
            <StandardListItem id="idSLI" title="{ProductId}"
description="{ProductName}"/>
        </items>
    </List>
</core:FragmentDefinition>
```

- Preview Application



RAP Basics

In this exercise we are going to create a RAP based Application.

- We will create a table for Customers, build a Class to fill the table with some sample data, then create a Root View CDS for that table.
- Next we will create a Service Definition for that CDS, and create 2 Service Bindings. A V2 UI and a V4 UI Service Binding. This to show that there are sometimes a few differences between V2 and V4.
- Next we will create a two UI5 Fiori elements applications, one for each version V2 and V4.

Next we are going to add **delete**, **update** and **create** functionality to the application.

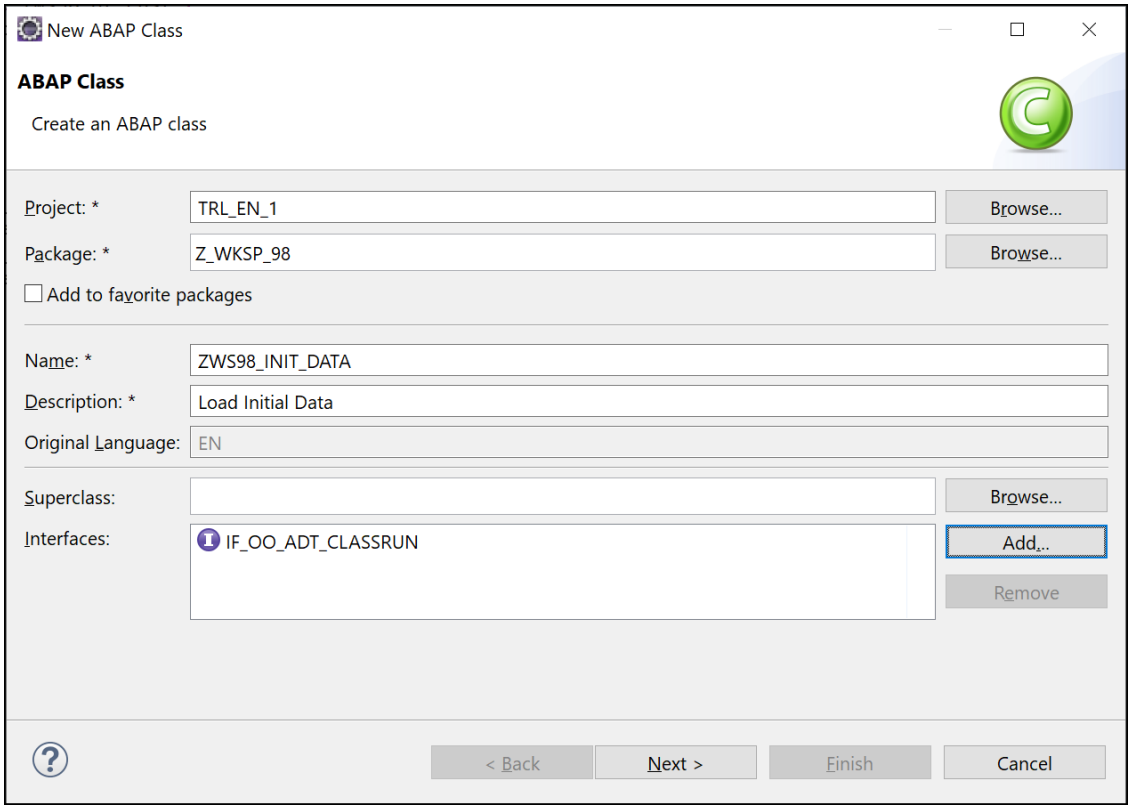
- We are going to add a **managed** *Behavior Definition* for the CDS.
- First the Delete
- Then the Edit
- Last the Create

Create a table for Customers **ZWS##_DT_CUST**

```
@EndUserText.label : 'Customers'
@AbapCatalog.enhancement.category : #NOT_EXTENSIBLE
@AbapCatalog.tableCategory : #TRANSPARENT
@AbapCatalog.deliveryClass : #A
@AbapCatalog.dataMaintenance : #RESTRICTED
define table zws##_DT_CUST {
  key client     : abap.clnt not null;
  key id         : ztmde9_customer_id not null;
  name           : abap.char(30);
  street         : abap.char(50);
  city           : abap.char(50);
  country        : land1;
  email          : abap.char(249);
  crea_date_time : timestampl;
  crea_uname     : syuname;
  lchg_date_time : timestampl;
  lchg_uname     : syuname;

}
```

Create a ABAP Class **ZWS##_INIT_DATA**

Create the ABAP Class and add the interface **IF_OO_ADT_CLASSRUN**, this will allow you to run the class from eclipse using the F9 function key.




```
CLASS zws##_init_data DEFINITION
  PUBLIC
  FINAL
  CREATE PUBLIC .

  PUBLIC SECTION.

    INTERFACES if_oo_adt_classrun .
    DATA: t_customers      TYPE STANDARD TABLE OF zws##_dt_cust.
    METHODS:
      delete_data,
      load_data.
  PROTECTED SECTION.
  PRIVATE SECTION.
ENDCLASS.


CLASS zws##_init_data IMPLEMENTATION.

  METHOD if_oo_adt_classrun~main.
    DATA: lv_reset  TYPE abap_boolean VALUE 'X',
          lv_delete TYPE abap_boolean VALUE ''.
    CASE abap_true.
```

```
      WHEN lv_reset.
        delete_data( ).
        load_data(  ).
        out->write( 'Data is reset!' ).
      WHEN lv_delete.
        delete_data( ).
        out->write( 'Data is deleted!' ).
    ENDCASE.
  ENDMETHOD.

  METHOD delete_data.
    DELETE FROM zws##_dt_cust WHERE id IS NOT NULL.
    COMMIT WORK AND WAIT.
  ENDMETHOD.

  METHOD load_data.
    DATA: lv_timestampl TYPE timestampl.
    GET TIME STAMP FIELD lv_timestampl.

    t_customers = VALUE #(
           ( id = '00001' name = 'Pixel Tech' street = 'Rue de la
Meuse 12' city = 'Arlon' country = 'LU' email = 'info@Pixel.lu'
              crea_date_time = lv_timestampl crea_uname = sy-uname
lchg_date_time = lv_timestampl lchg_uname = sy-uname )
           ( id = '00002' name = 'Visions' street = 'Rue Thomas
Edison 56' city = 'Luxembourg' country = 'LU' email =
'mail@visions.lu'
              crea_date_time = lv_timestampl crea_uname = sy-uname
lchg_date_time = lv_timestampl lchg_uname = sy-uname )
           ( id = '00003' name = 'Techaholic' street = 'De dam 15'
city = 'Amsterdam' country = 'NL' email = 'post@techaholic.nl'
              crea_date_time = lv_timestampl crea_uname = sy-uname
lchg_date_time = lv_timestampl lchg_uname = sy-uname )
           ( id = '00004' name = 'Monotech' street = 'Schoolstraat
1' city = 'Alphen' country = 'NL' email = 'po.dep@monotech.nl'
              crea_date_time = lv_timestampl crea_uname = sy-uname
lchg_date_time = lv_timestampl lchg_uname = sy-uname )
           ( id = '00005' name = 'Techlanch' street = 'Bronstraat
6' city = 'Brussels' country = 'BE' email = 'info@techlanch.be'
              crea_date_time = lv_timestampl crea_uname = sy-uname
lchg_date_time = lv_timestampl lchg_uname = sy-uname )
            ).
    MODIFY zws##_dt_cust FROM TABLE @t_customers.
    COMMIT WORK AND WAIT.
  ENDMETHOD.

ENDCLASS.
```

- Run the Class using F9, you will see the message *Data is reset!* in the Console of Eclipse.
- Check the content of the table by opening the table and then pressing F8 or right click on the table and select *Open With -> Data Preview*.

Create a CDS **ZWS$$_CDS_RAP_BASIC**

- Create a CDS with Template **Define View**
- Add the word **root** between *define* and *view*

```
define root view ...
```

- and add all the elements, but keep the names of the last 4 elements with the _.
- and set the @AccessControl.authorizationCheck to **#NOT_REQUIRED**.

```
@AbapCatalog.sqlViewName: 'ZWS##CDSRAPB1'
@AbapCatalog.compiler.compareFilter: true
@AbapCatalog.preserveKey: true
@AccessControl.authorizationCheck: #NOT_REQUIRED
@EndUserText.label: 'Basic RAP CDS'
define root view ZWS##_CDS_RAP_BASIC
  as select from zws##_dt_cust
{
  key id             as Id,
      name           as Name,
      street         as Street,
      city           as City,
      country        as Country,
      email          as Email,
      crea_date_time as Crea_Date_Time,
      crea_uname     as Crea_Uname,
      lchg_date_time as Lchg_Date_Time,
      lchg_uname     as Lchg_Uname
}
```

Add Basic Annotations for *lineitem* and *facet*

- Set *lineitem* & *identification* annotations for the fields Id, Name, Street, City, Country and Email.
- Add *@UI.facet*

```
@AbapCatalog.sqlViewName: 'ZWS##CDSRAPB1'
@AbapCatalog.compiler.compareFilter: true
@AbapCatalog.preserveKey: true
@AccessControl.authorizationCheck: #NOT_REQUIRED
@EndUserText.label: 'Basic RAP CDS'
@UI.headerInfo.typeName: 'Customer'
@UI.headerInfo.typeNamePlural: 'Customers'
define root view ZWS##_CDS_RAP_BASIC
  as select from zws##_dt_cust
{
      @UI.facet: [{id: 'Customer', purpose: #STANDARD, type:
#IDENTIFICATION_REFERENCE, label: 'Customer', position: 10 }]

      @UI: { lineItem: [{position: 10, importance: #HIGH, label:
'ID' }],
                  identification: [{position: 10, label: 'ID' }] }
  key id              as Id,
      @UI: { lineItem: [{position: 20, importance: #HIGH, label:
'Name' }],
                    identification: [{position: 20, label: 'Name'
}] }
      name            as Name,
      @UI: { lineItem: [{position: 30, importance: #HIGH, label:
'Street' }],
            identification: [{position: 30, label: 'Street' }] }
      street          as Street,
      @UI: { lineItem: [{position: 40, importance: #HIGH, label:
'City' }],
           identification: [{position: 40, label: 'City' }] }
      city            as City,
      @UI: { lineItem: [{position: 50, importance: #HIGH, label:
'Country' }],
            identification: [{position: 50, label: 'Country' }] }
      country         as Country,
      @UI: { lineItem: [{position: 60, importance: #HIGH, label:
'Email' }],
            identification: [{position: 60, label: 'Email' }] }
      email           as Email,
      crea_date_time as Crea_Date_Time,
      crea_uname     as Crea_Uname,
      lchg_date_time as Lchg_Date_Time,
      lchg_uname     as Lchg_Uname
}
```

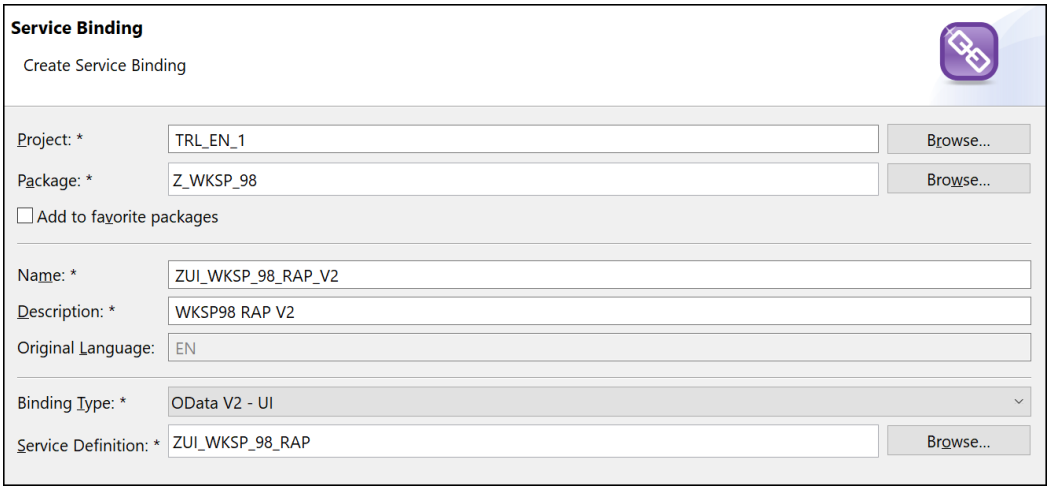
Create Service Definition **ZUI_WKSP_##_RAP**

- Create a Service Definition and expose your CDS as *Customers*

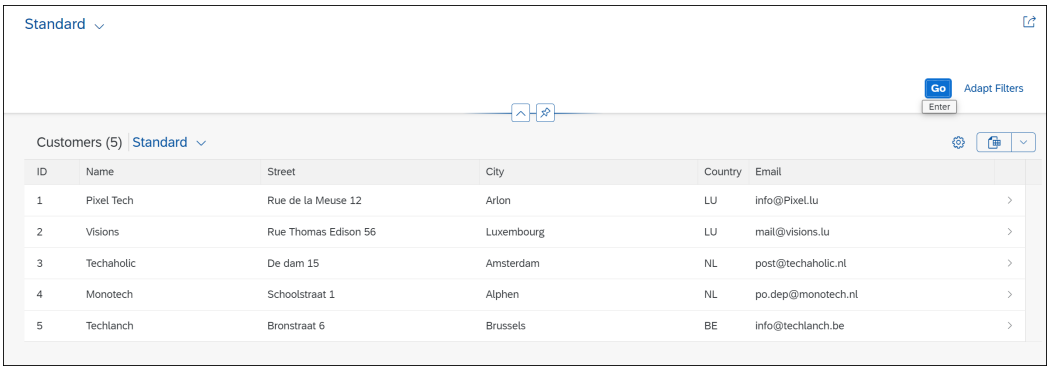
```
@EndUserText.label: 'Service definition Customers'
define service ZUI_WKSP_##_RAP {
    expose zws##_cds_rap_basic as Customers;
}
```

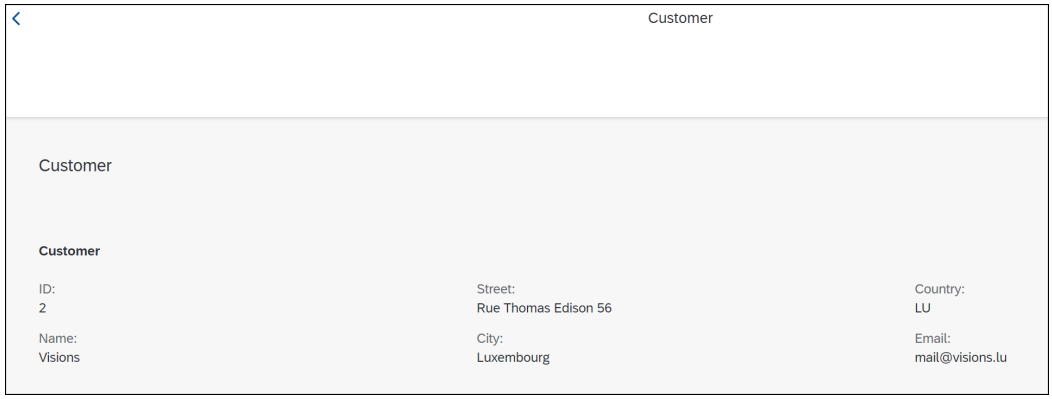
Create 2 Service Bindings **ZUI_WKSP_##_RAP_V2** and **ZUI_WKSP_##_RAP_V4**

- Create a Service Binding with *Binding Type* **ODAta V2 - UI** for *Service Definition* **ZUI_WKSP_##_RAP**

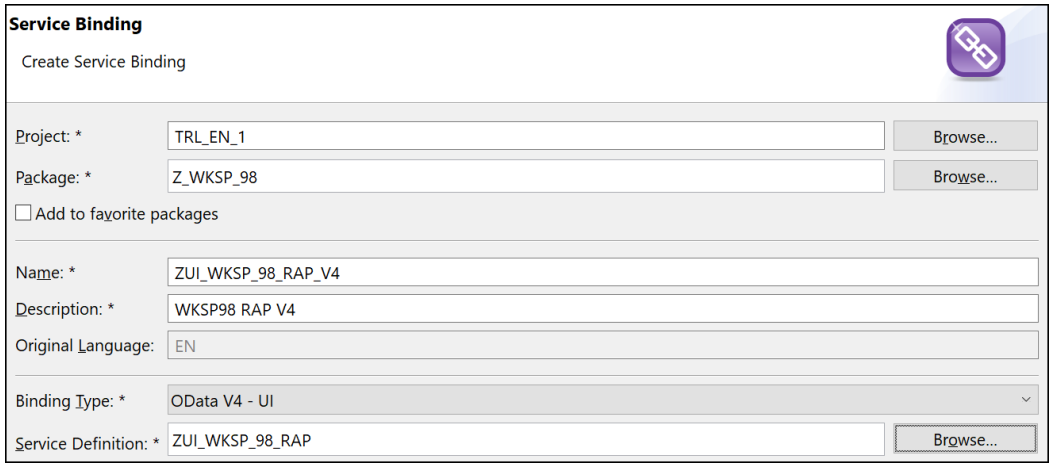


- Activate and Publish the Service Binding.
- Test/Preview the Entity Set **Customers**

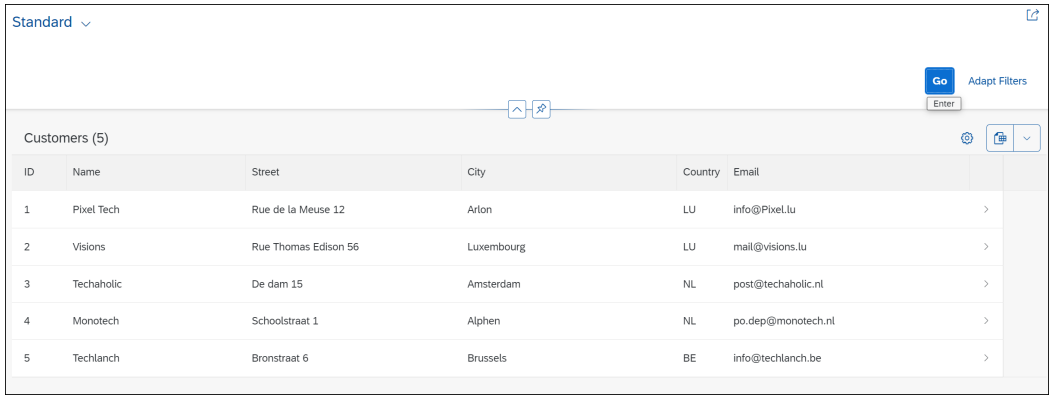




- Create a Service Binding with *Binding Type* **OData V4 - UI** for *Service Definition* **ZUI_WKSP_##_RAP**



- Activate and Publish the Service Binding.
- Test/Preview the Entity Set **Customers**



Create 2 Fiori Elements application with floorplan **List Report Object Page**, for the OData V2 and the OData V4

V2

- Logon to your SAP Business Application Studio and create a *New Project From Template*
- Create a Fiori Appliction with *SAP Fiori Elements* and floorplan **List Report Object Page**
- Select *Connect to a System* and select your **abap-cloud-default_abap-trial-xxx (BTP)**
- Select your **V2** Service Binding
- Select *Customers* as your main entity
- Set Project Attributes:

Name	Value
Module name	rap-basic-v2
Application title	RAP Basic V2
Application namespace	nato.workshop
Description	RAP Basic V2
Project folder path	/home/user/projects
Minimum SAPUI5 version	leave as is
Add deployment configuration	Yes
Add FLP configuration	Yes
Configure advanced options	No

- Set Deployment Configuration

- Enter Fiori Launchpad Configuration

Name	Value
Semantic Object	NATO
Action	RapBasicV2
Title	RAP Basic V2
Subtitle (optional)	Workshop

- Run the Preview of the application

V4

- Logon to your SAP Business Application Studio and create a *New Project From Template*
- Create a Fiori Appliction with *SAP Fiori Elements* and floorplan **List Report Object Page**
- Select *Connect to a System* and select your **abap-cloud-default_abap-trial-xxx (BTP)**
- Select your **V4** Service Binding
- Select *Customers* as your main entity
- Set Project Attributes:

Name	Value
Module name	rap-basic-v4
Application title	RAP Basic V4
Application namespace	nato.workshop
Description	RAP Basic V4
Project folder path	/home/user/projects
Minimum SAPUI5 version	leave as is
Add deployment configuration	Yes
Add FLP configuration	Yes
Configure advanced options	No

- Set Deployment Configuration

- Enter Fiori Launchpad Configuration

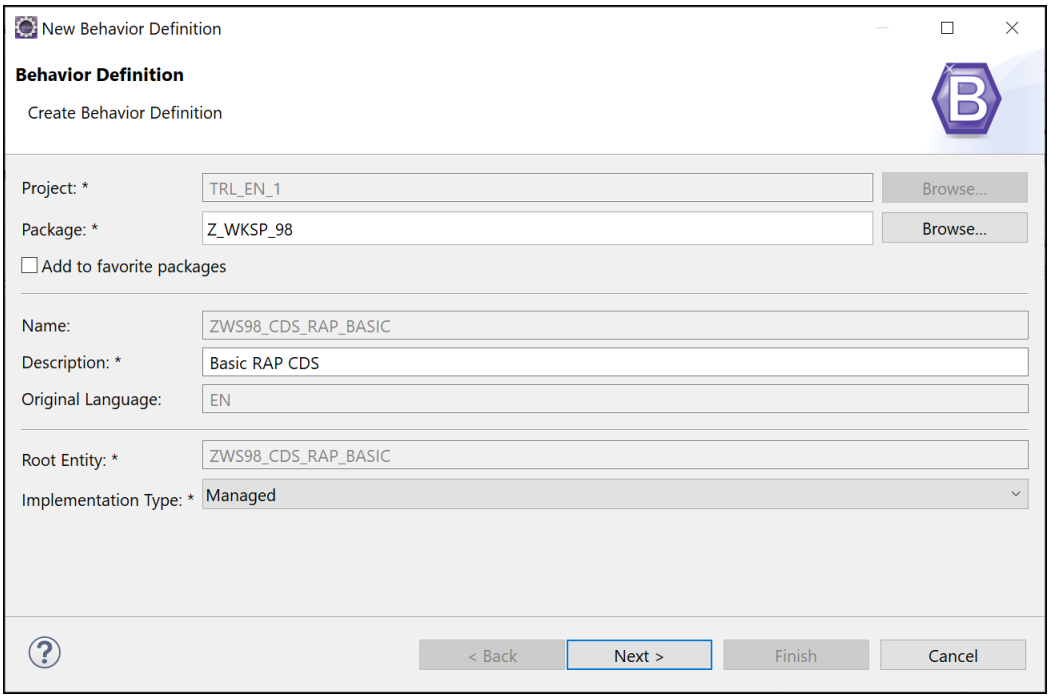
Name	Value
Semantic Object	NATO
Action	RapBasicV4
Title	RAP Basic V4
Subtitle (optional)	Workshop

- Run the Preview of the application

RAP Basics add Delete, Update, Create

Create a *Managed Behavior* for your CDS only for Delete

- Right click on your CDS and select **New Behavior Definition**



- Set the *alias* to **Customer*
- Remove the **create** and **update** options:

```
managed implementation in class zbp_ws##_cds_rap_basic unique;

define behavior for ZWS##_CDS_RAP_BASIC alias Customer
persistent table ZWS##_DT_CUST
lock master
authorization master ( instance )
//etag master <field_name>
{
  delete;
}
```

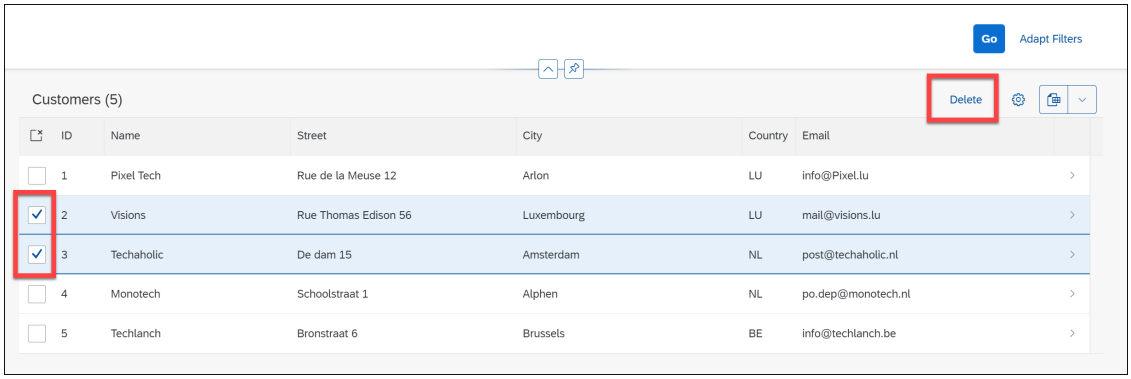
- Activate the Behavior
- Use **CTRL** + **SHIFT** + **1** to open the *Quick Assist* tab.
- Next click on the Class name in the Behavior and see the proposal in the *Quick Assist* tab.
- Double click the Proposal to generate the class

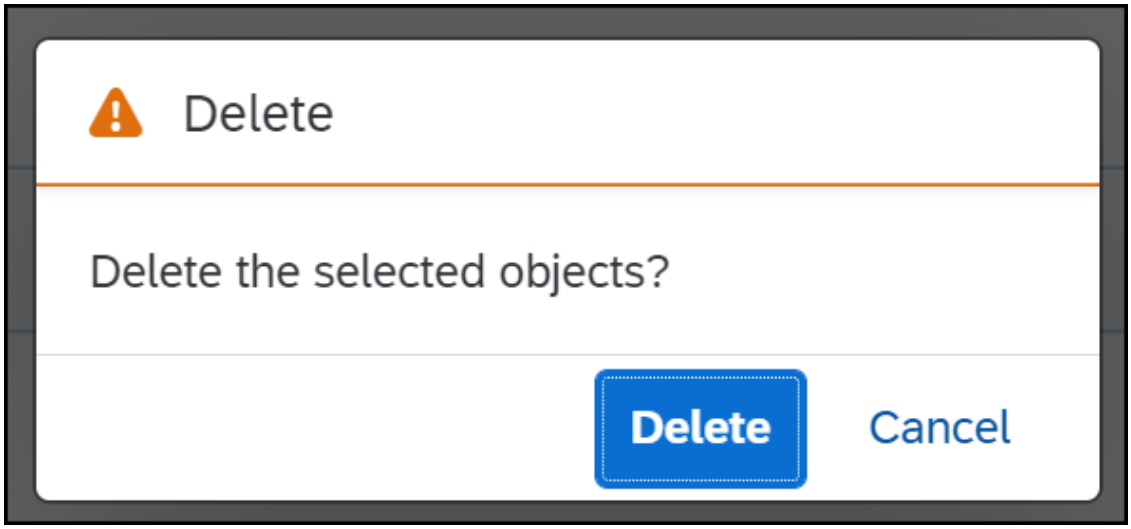
```
CLASS zbp_ws##_cds_rap_basic DEFINITION PUBLIC ABSTRACT FINAL FOR
BEHAVIOR OF zws##_cds_rap_basic.
ENDCLASS.

CLASS zbp_ws##_cds_rap_basic IMPLEMENTATION.
ENDCLASS.
```

```
CLASS lhc_Customer DEFINITION INHERITING FROM
cl_abap_behavior_handler.
  PRIVATE SECTION.

    METHODS get_instance_authorizations FOR INSTANCE AUTHORIZATION
      IMPORTING keys REQUEST requested_authorizations FOR Customer
RESULT result.

ENDCLASS.

CLASS lhc_Customer IMPLEMENTATION.

  METHOD get_instance_authorizations.
  ENDMETHOD.

ENDCLASS.
```

- Test the delete function from either the preview from Eclipse or the BAS application.
- There is now a **Delete** button, when you select one or more lines you can delete them. Check out the difference between the V2 and V4 applications.
- Or click on a line and use the **Delete** button in the Object Page.

> If you have deleted all your records, just run the Class **ZWS##_INIT_DATA** again with F9.





Add *Edit* Option to the Behavior Definition

- Add the **update** option to the Behavior Definition.

```
managed implementation in class zbp_ws##_cds_rap_basic unique;

define behavior for ZWS##_CDS_RAP_BASIC alias Customer
persistent table ZWS##_DT_CUST
lock master
authorization master ( instance )
//etag master <field_name>
{
  update;
  delete;
}
```

- Activate the Behavior Definition.
- Test the application again. See the difference between the V2 and V4 versions.

> The V4 is not displaying the **Edit** Button whereas the V2 is. This is because for the V4 version we need to add the **Draft** option. We will do this later. For now just test with the V2 application.

- When you press the edit button you are able to change the *key* field **ID**. That is not wat we want.
- To prevent this we add **field (readonly) ID;** to the Behavior Definition. This will make the ID field read only.

```

```

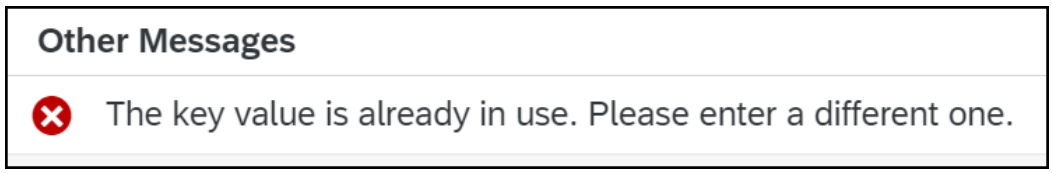
- Activate and the V2 application again.

Add *Create* Option to the Behavior Definition

- Add the **create** option to the Behavior Definition.

```
managed implementation in class zbp_ws##_cds_rap_basic unique;

define behavior for ZWS##_CDS_RAP_BASIC alias Customer
persistent table ZWS##_DT_CUST
lock master
authorization master ( instance )
etag master Lchg_Date_Time
{
  create;
  update;
  delete;

  field (readonly) ID;
}
```

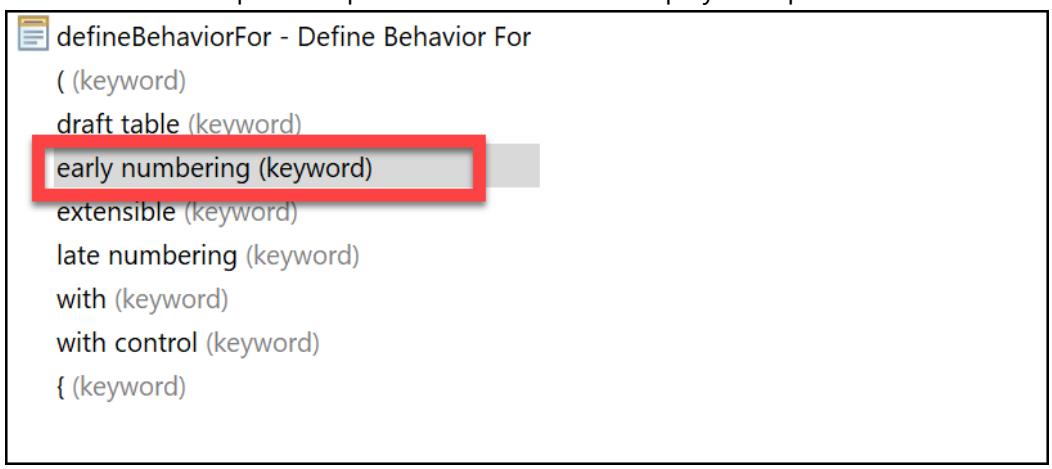
- Activate the Behavior Definition.
- Test the application again.
- You now see that you can only *once* add a new record, because the second new record that you want to make gives you an error:



- This can be solved by using **early numbering**

Add **early numbering** to the Behavior Definition

- Add the **early numbering** to the Behavior Definition
- Use the Code Completion option **CTRL**+**SPACE** to display the options



- Double click the word **create** to use the *Quick Assist* to generate a new method
- Implement the new method by checking the MAX number in the table and adding 1 to that max number.

```
METHOD earlynumbering_create.

    LOOP AT entities INTO DATA(entity) WHERE id IS NOT INITIAL.
      APPEND CORRESPONDING #( entity ) TO mapped-customer.
    ENDLOOP.

    DATA(entities_without_id) = entities.
    DELETE entities_without_id WHERE id IS NOT INITIAL.

    "Get max travel ID from standard table
    SELECT SINGLE FROM zws##_dt_cust FIELDS MAX( id ) INTO
@DATA(max_cust_id).

    "Set Customer Id
    LOOP AT entities_without_id INTO entity.
      max_cust_id += 1.
      entity-Id = max_cust_id.

      APPEND VALUE #( %cid = entity-%cid
                      %key = entity-%key
```

```
                    ) TO mapped-customer.
    ENDLOOP.

  ENDMETHOD.
```

- Activate the Behavior Definition.
- Test the application again.
- You now see that when you press save the record gets the next number in the table.

RAP with Draft Verion

Now we are going to add *Draft version* to the process. To keep the basic version separated from the new Draft version we need to copy/create a new CDS.

- Copy CDS **ZWS##_CDS_RAP_BASIC** to **ZWS##_CDS_RAP_BASIC_DRAFT**
- Change the *sqlViewName*
- Change the words *Customer* to *Client*, that way we can see which CDS we are using when we test the applications.

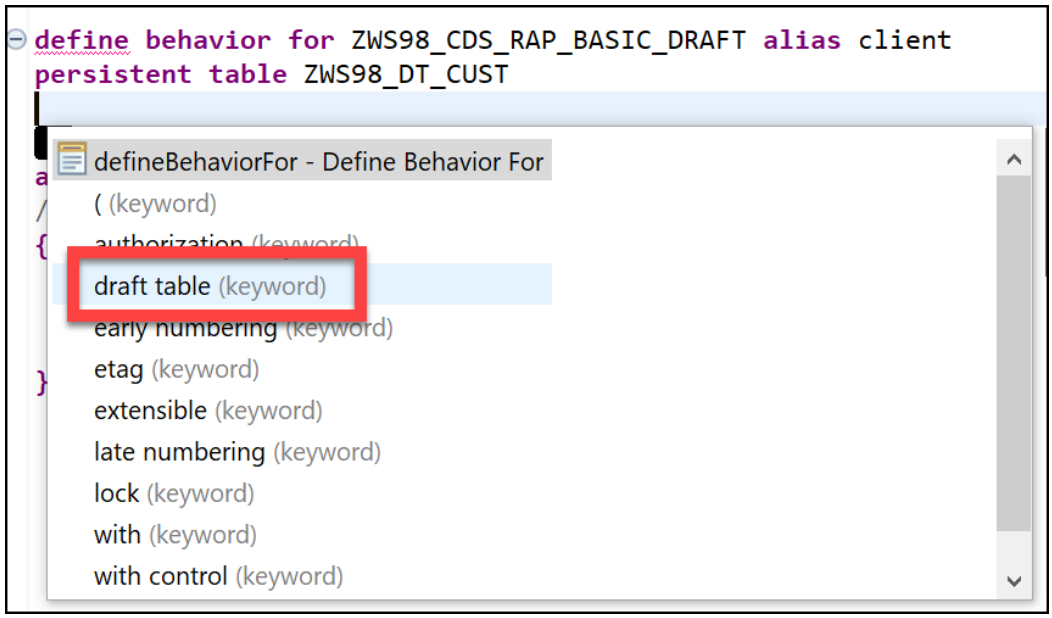
```
@AbapCatalog.sqlViewName: 'ZWS##CDSRAPB2'
@AbapCatalog.compiler.compareFilter: true
@AbapCatalog.preserveKey: true
@AccessControl.authorizationCheck: #NOT_REQUIRED
@EndUserText.label: 'Basic RAP CDS'
@UI.headerInfo.typeName: 'Client'
@UI.headerInfo.typeNamePlural: 'Clients'
define root view ZWS##_CDS_RAP_BASIC_DRAFT
  as select from zws##_dt_cust
{
      @UI.facet: [{id: 'Client', purpose: #STANDARD, type:
#IDENTIFICATION_REFERENCE, label: 'Client', position: 10 }]

      @UI: { lineItem: [{position: 10, importance: #HIGH, label:
'ID' }],
                  identification: [{position: 10, label: 'ID' }] }
  key id              as Id,
      @UI: { lineItem: [{position: 20, importance: #HIGH, label:
'Name' }],
                   identification: [{position: 20, label: 'Name'
}] }
      name            as Name,
      @UI: { lineItem: [{position: 30, importance: #HIGH, label:
'Street' }],
            identification: [{position: 30, label: 'Street' }] }
      street          as Street,
      @UI: { lineItem: [{position: 40, importance: #HIGH, label:
'City' }],
           identification: [{position: 40, label: 'City' }] }
      city            as City,
      @UI: { lineItem: [{position: 50, importance: #HIGH, label:
'Country' }],
            identification: [{position: 50, label: 'Country' }] }
      country         as Country,
      @UI: { lineItem: [{position: 60, importance: #HIGH, label:
```

```
'Email' }],
            identification: [{position: 60, label: 'Email' }] }
      email          as Email,
      crea_date_time as Crea_Date_Time,
      crea_uname     as Crea_Uname,
      lchg_date_time as Lchg_Date_Time,
      lchg_uname     as Lchg_Uname
}
```

- Add the new CDS to the Service Definition as *Clients*

```
@EndUserText.label: 'Service definition Customers'
define service ZUI_WKSP_##_RAP {
  expose ZWS##_CDS_RAP_BASIC as Customers;
  expose ZWS##_CDS_RAP_BASIC_DRAFT as Clients;
}
```

- Add a new *managed* **Behavior Definition** to the new CDS
- Add the option **with draft;**
- Set the Alias to **client**
- Under *persistent table* add **draft table ZWS##_DT_CUST_DR**, use the Code Completion



- Use the *Quick Assist* to generate the **Draft table** for you, See the newly added field **"%admin"**

```
@EndUserText.label : 'Draft table for entity
ZWS##_CDS_RAP_BASIC_DRAFT'
```

```
@AbapCatalog.enhancement.category : #EXTENSIBLE_ANY
@AbapCatalog.tableCategory : #TRANSPARENT
@AbapCatalog.deliveryClass : #A
@AbapCatalog.dataMaintenance : #RESTRICTED
define table zws##_dt_cust_dr {
  key mandt      : mandt not null;
  key id         : ztmde9_customer_id not null;
  name           : abap.char(30);
  street         : abap.char(50);
  city           : abap.char(50);
  country        : land1;
  email          : abap.char(249);
  crea_date_time : timestampl;
  crea_uname     : syuname;
  lchg_date_time : timestampl;
  lchg_uname     : syuname;
  "%admin"       : include sych_bdl_draft_admin_inc;

}
```

- Add the options: **lock master total etag xx** and **etag master xx**
- Set the *field (readonly) id;*
- Result of the Behavior Definition should look like this.

```
managed implementation in class zbp_ws##_cds_rap_basic_draft
unique;
with draft;

define behavior for ZWS##_CDS_RAP_BASIC_DRAFT alias client
persistent table ZWS##_DT_CUST
draft table ZWS##_DT_CUST_DR
lock master total etag Lchg_Date_Time
authorization master ( instance )
etag master Lchg_Date_Time
{
  create;
  update;
  delete;

  field ( readonly ) id;
}
```

- Put your cursor on the class name in the first line of the Behavior Definition and use the *Quick Assist* to Create the implementation class.

- Test your application using the preview of **Clients** in the *Service Binding*
- Delete works as expected
- Edit, you now have the option to see the **Display Saved Version** or the **New Draft Version**
- If you make a change you shortly see **Saving Draft** at the bottom and then **Draft saved**. This tells you if there is a draft version available.
- With the *Cancel* button you get the option to *Discard* all changes.
- Create works again only 1 time, after that you get an error that the Key Value is already in use.

Fix the Create

- To fix the option to create multiple draft versions you need to add the option **late numbering** to the Behavior Definition.

```
managed implementation in class zbp_ws##_cds_rap_basic_draft
unique;
with draft;
//strict; //Comment this line in to enable strict mode. The strict
mode is prerequisite to be future proof regarding syntax and to be
able to release your BO.

define behavior for ZWS##_CDS_RAP_BASIC_DRAFT alias client
late numbering
persistent table ZWS##_DT_CUST
draft table ZWS##_DT_CUST_DR
lock master total etag Lchg_Date_Time
authorization master ( instance )
etag master Lchg_Date_Time
{
  create;
  update;
  delete;

  field ( readonly ) id;
}
```

- When you activate the new Behavior Definition, you need to fix 2 things.
- First: use the *Quick Assist* to recreate the draft table, this because we need an extra key field.

```
@EndUserText.label : 'Draft table for entity
ZWS98_CDS_RAP_BASIC_DRAFT'
@AbapCatalog.enhancement.category : #EXTENSIBLE_ANY
@AbapCatalog.tableCategory : #TRANSPARENT
@AbapCatalog.deliveryClass : #A
@AbapCatalog.dataMaintenance : #RESTRICTED
define table zws98_dt_cust_dr {
  key mandt      : mandt not null;
  key id         : ztmde9_customer_id not null;
  key draftuuid  : sdraft_uuid;
  name           : abap.char(30);
  street         : abap.char(50);
  city           : abap.char(50);
  country        : land1;
  email          : abap.char(249);
  crea_date_time : timestampl;
  crea_uname     : syuname;
  lchg_date_time : timestampl;
  lchg_uname     : syuname;
  "%admin"       : include sych_bdl_draft_admin_inc;

}
```

- To be able to generate the new definition of the Draft Table you first may have to delete the Draft Table or make sure there are no records in the Draft Table.
- Second: use the *Quick Assist* to add a new method for *late numbering*
- Implement the new method **adjust_numbers**, to get the latest number from the table

```
 METHOD adjust_numbers.
  IF mapped-client IS NOT INITIAL.
      "  Get max travel ID from standard table
      SELECT SINGLE FROM zws98_dt_cust FIELDS MAX( id ) INTO
@DATA(max_cust_id).
      max_cust_id += 1.
      mapped-client[ 1 ]-%key-Id = max_cust_id.
    ENDIF.
  ENDMETHOD.
```

Blogs and Tips and Tricks

Connecting from SAP Business Application Studio to SAP ABAP Environment in BTP

Set Up Local Development Using VS Code

If cf --version does not work after install of Cloud Foundry, then you need to make a symbolick link manually:

Open a CMD in administrator mode and execute following command:

```
mklink "c:\Program Files\Cloud Foundry\cf.exe" "c:\Program
Files\Cloud Foundry\cf8.exe"
```

Login to cf
Start a terminal and execute the following:

> cf login -a <API-URL>
> and enter your email, password and select your organisation and space.

```
cf login -a https://api.cf.us10.hana.ondemand.com
```

If Fiori Template wizard niet wil starten:
Uninstall the existing generator with the command: `npm uninstall -g @sap/generator-fiori`.
Then try and install the generator again by executing: `npm i -g @sap/generator-fiori`.

SAP Help, Develop Applications With RAP

Space in your SAP Business Application Studio is low.

- You can delete the **node_modules** folders out of you projects.
- Then clean the *Trash* folder, the *yarn cache* and *npm cache* folder, using the next commands in a terminal

```
rm -rf /home/user/.local/share/Trashfiles
rm -rf /home/user/.cache
rm -rf /home/user/.npm
```

Additional Online Workshops

Workshops about the ABAP RESTful Application Programming Model (RAP)

CodeJam - SAP Business Technology Platform, ABAP Environment & ABAP RESTful Application Programming Model