1 Goal of this tutorial

Set up the SAP WebIDE Eclipse based development environment and
use it with Chrome browser (http://localhost:8080/webide/index.html).
Create a simple Hello World Application, upload it to the ABAP server and test it.

2 General

In the following examples replace XXX by your id and Vorname / Nachname by your name – you can use all materials, internet, work in teams of two, but you have to create your own results.

Collect the artefacts of your result as requested below in a result document `SAPUI5_01_<name>_<vorname>.docx` [Total: 19P].

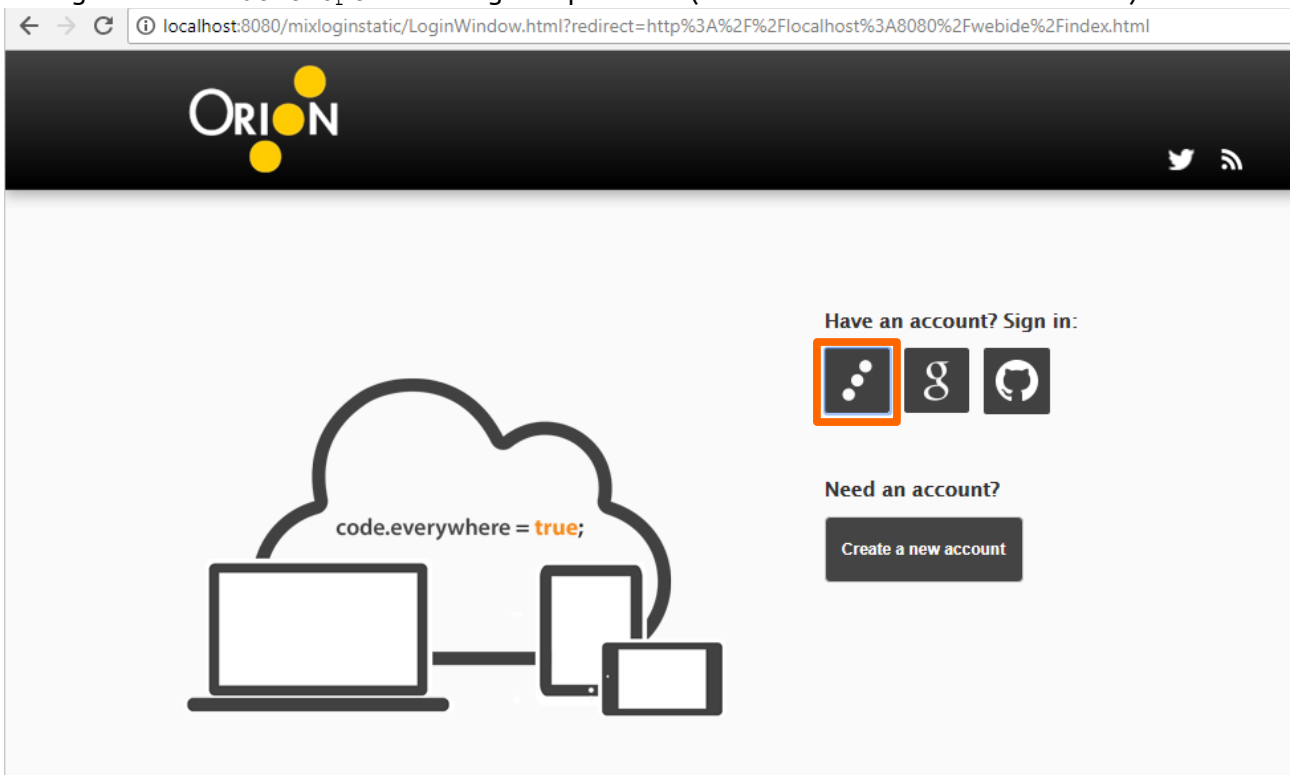
3 Setup of IDE

1 Extract the `sapWebIDE-PE.zip` (on class share) inside your VM on your Desktop as directory `sapWebIDE-PE`

2 Start `sapWebIDE-PE/orion.exe` which will open a command shell

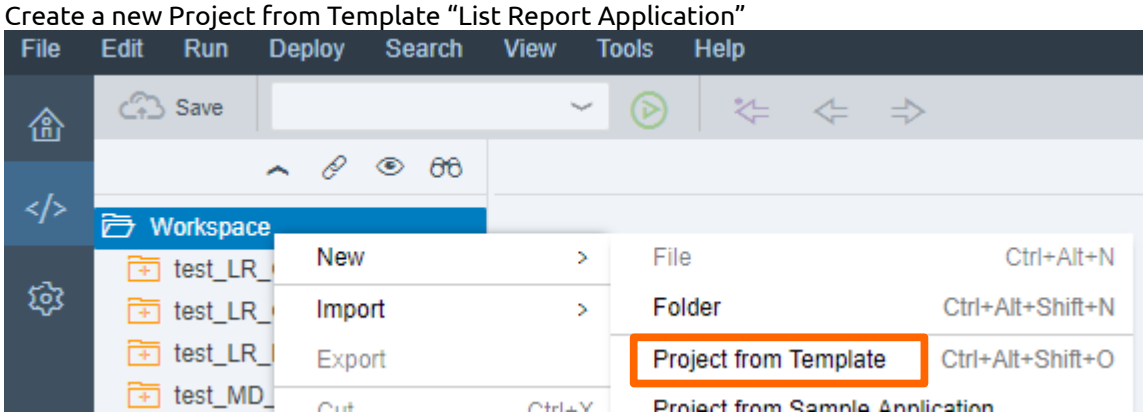
3 Open the link "Chrome – SAPWebIDE.." in Chrome

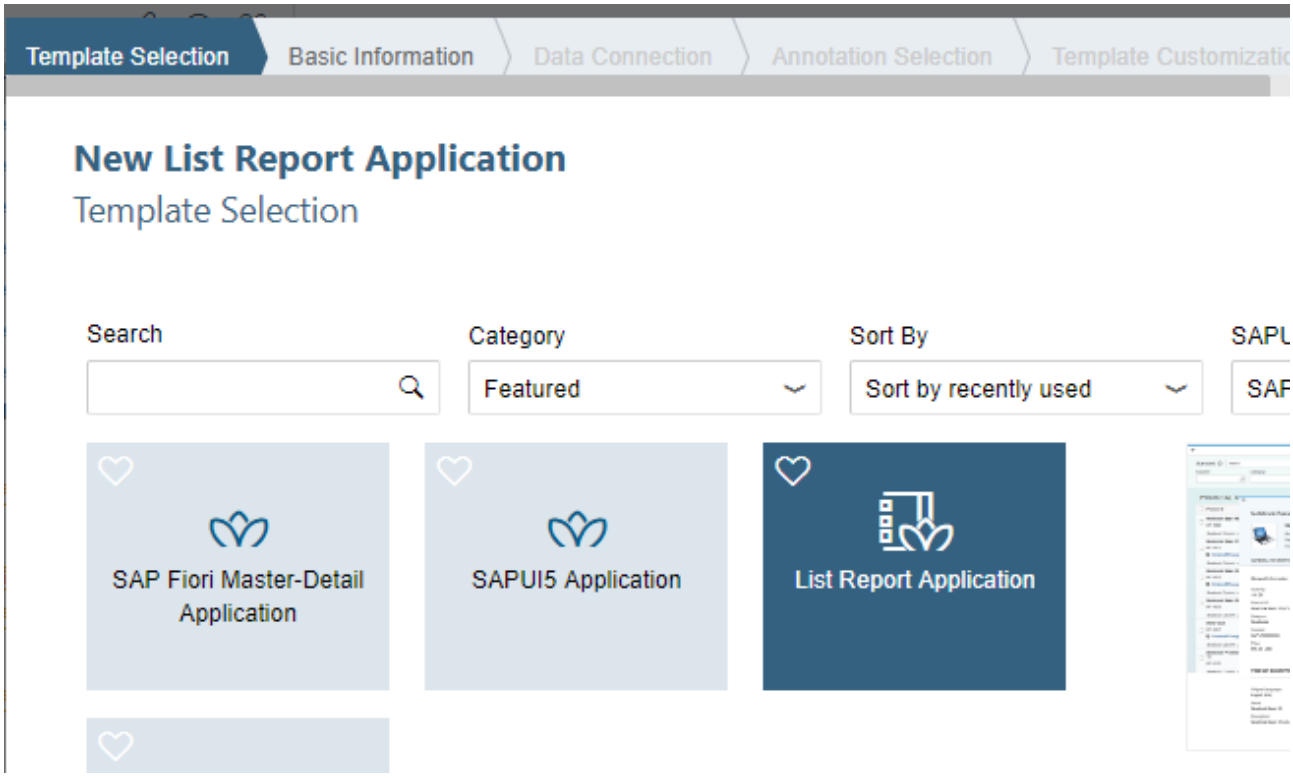
4 Log in with user `developer` and the gibbix password (click on the icon with the three dots):

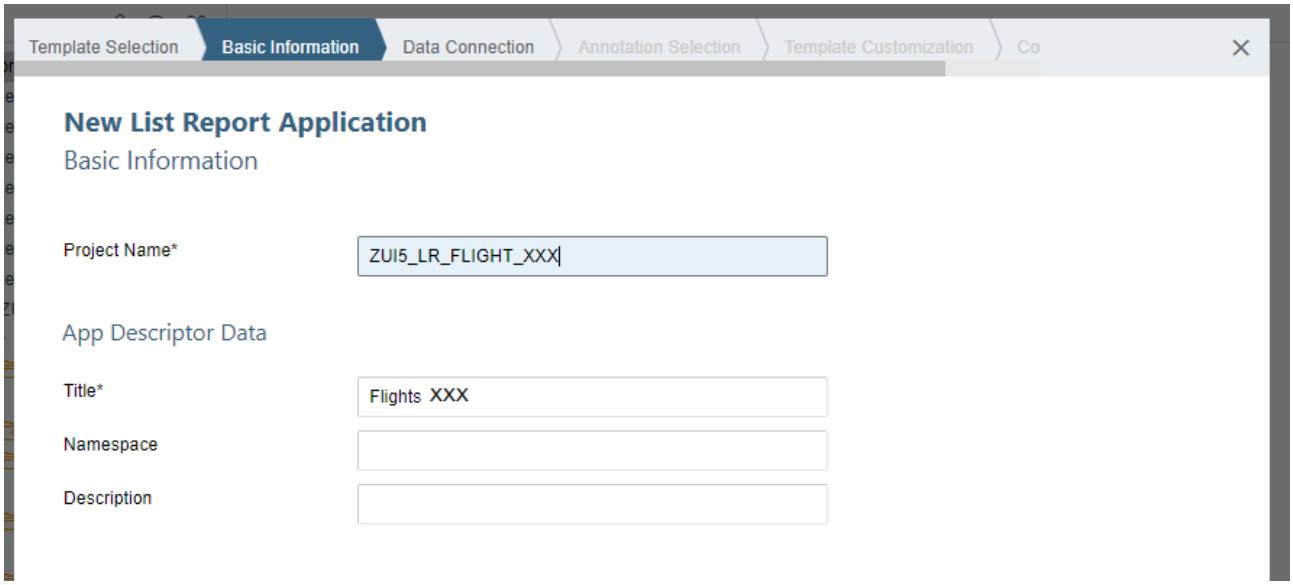


5 First: Generating List Report Application using an OData-Service

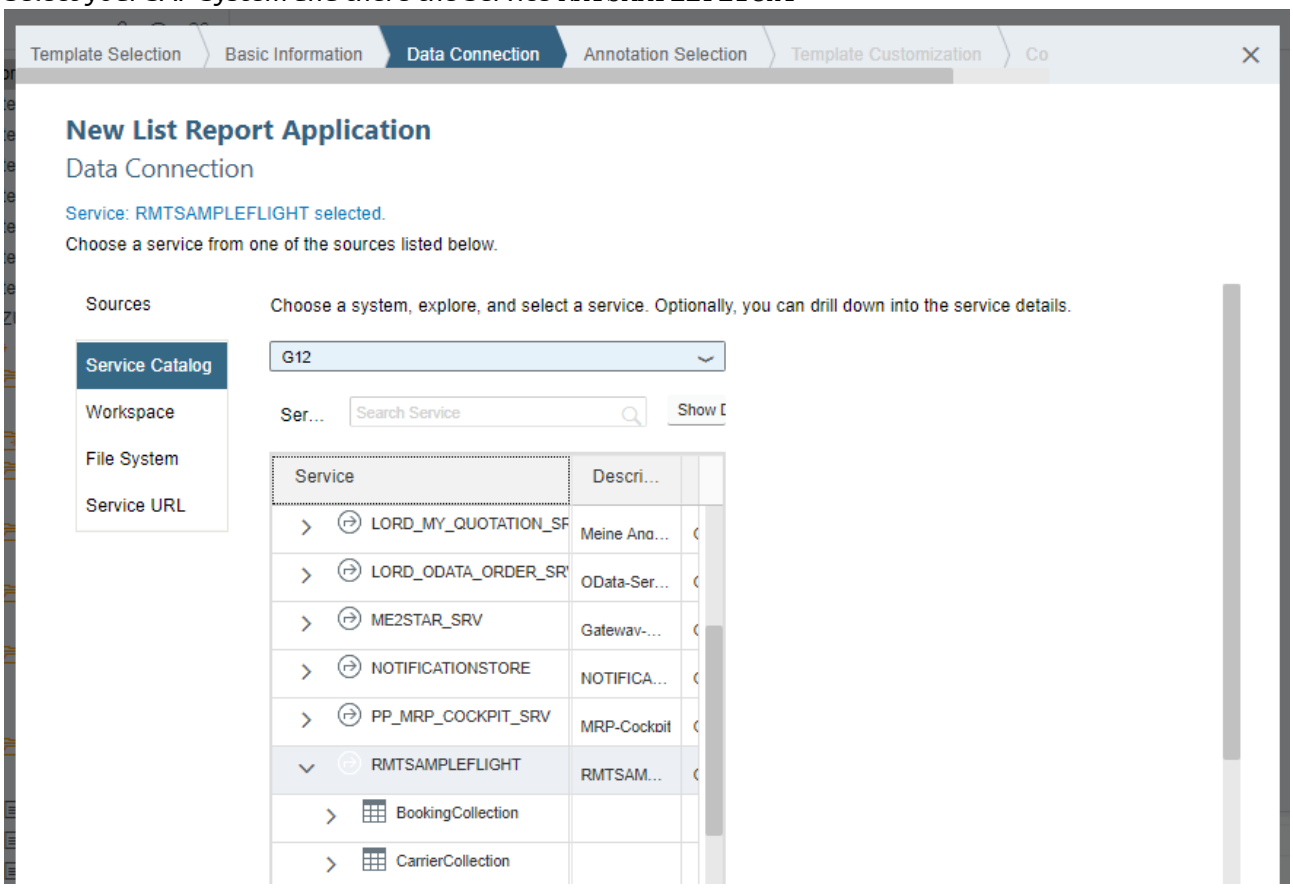
Create a new Project from Template “List Report Application”

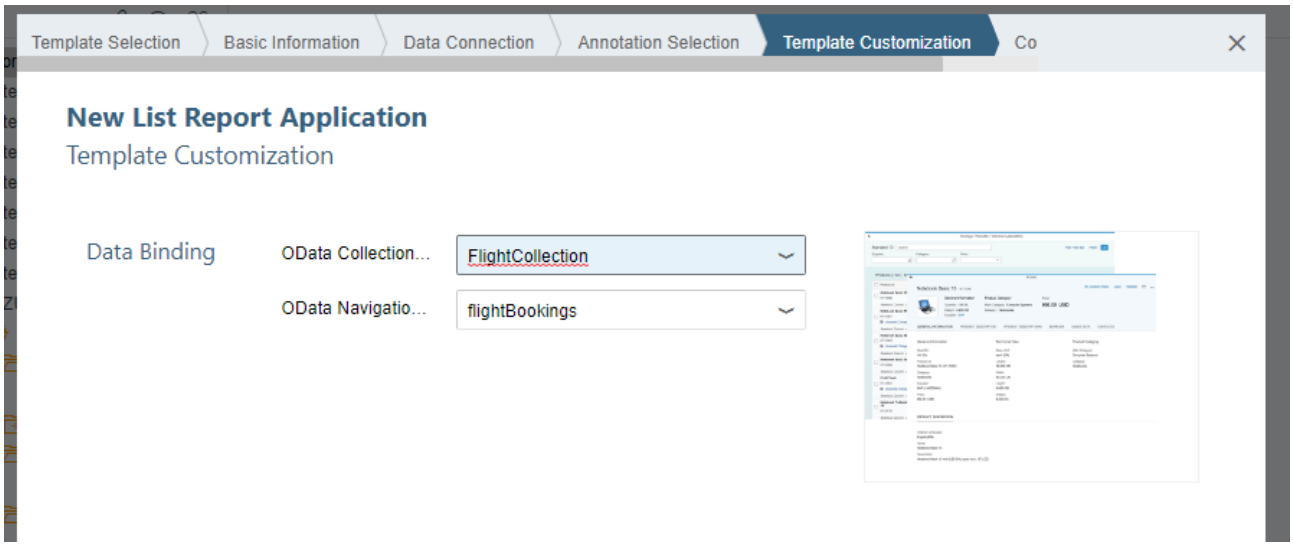




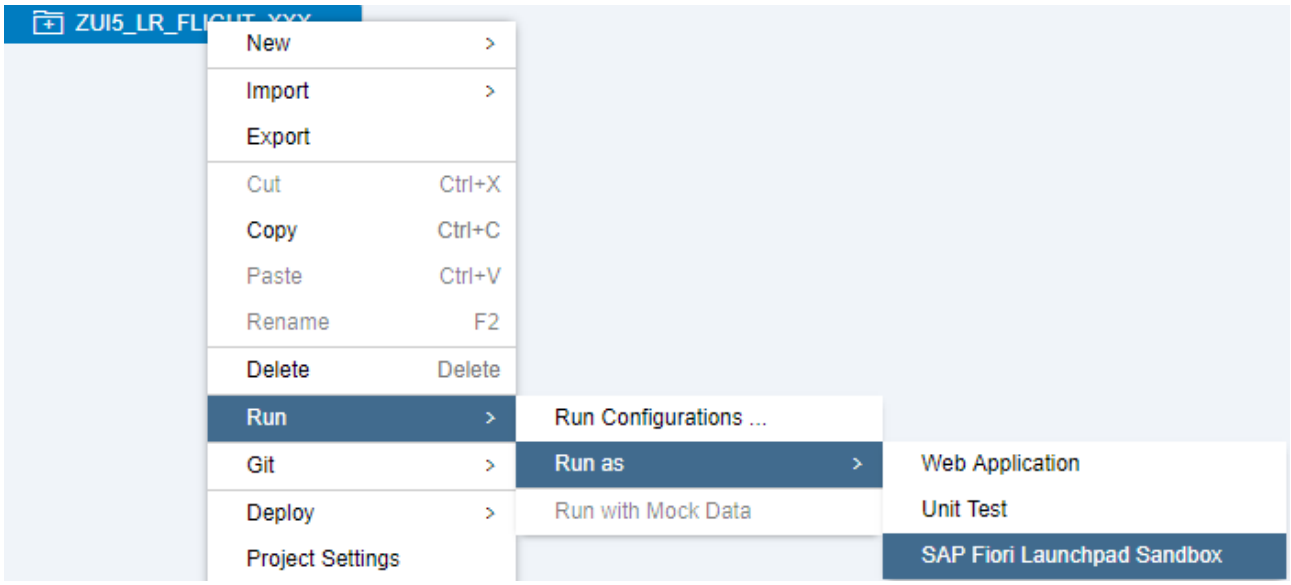


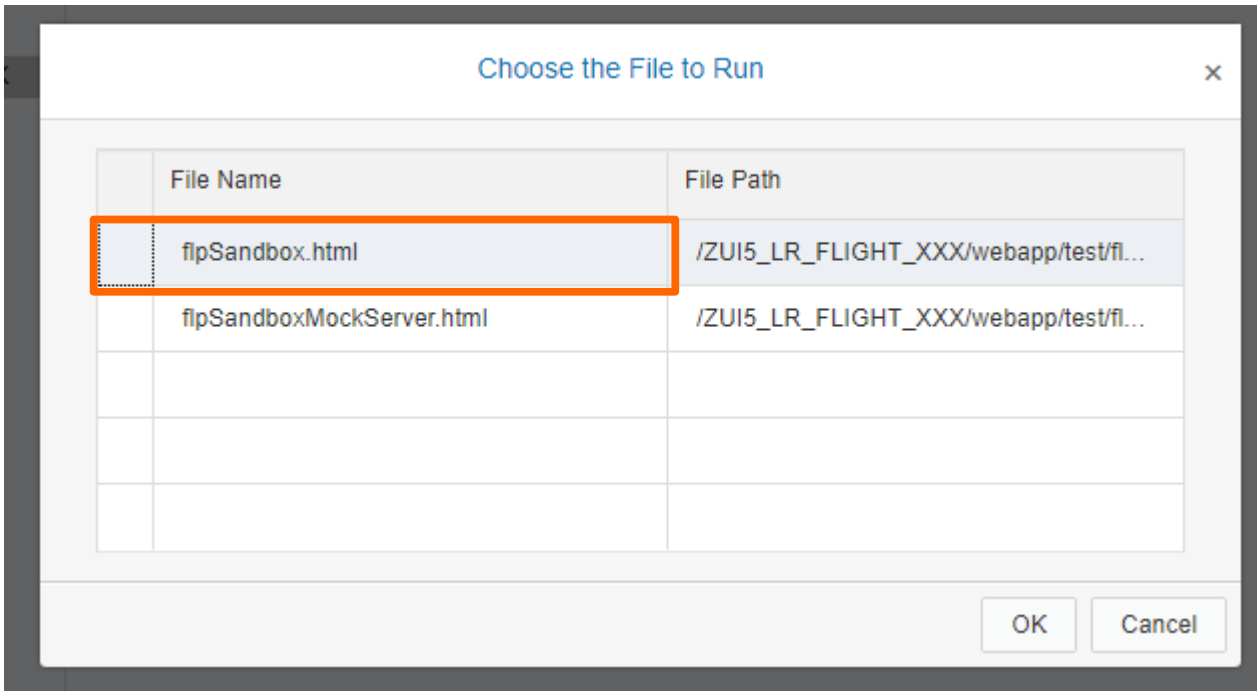
Select your SAP System and there the Service `RMTSAMPLEFLIGHT`

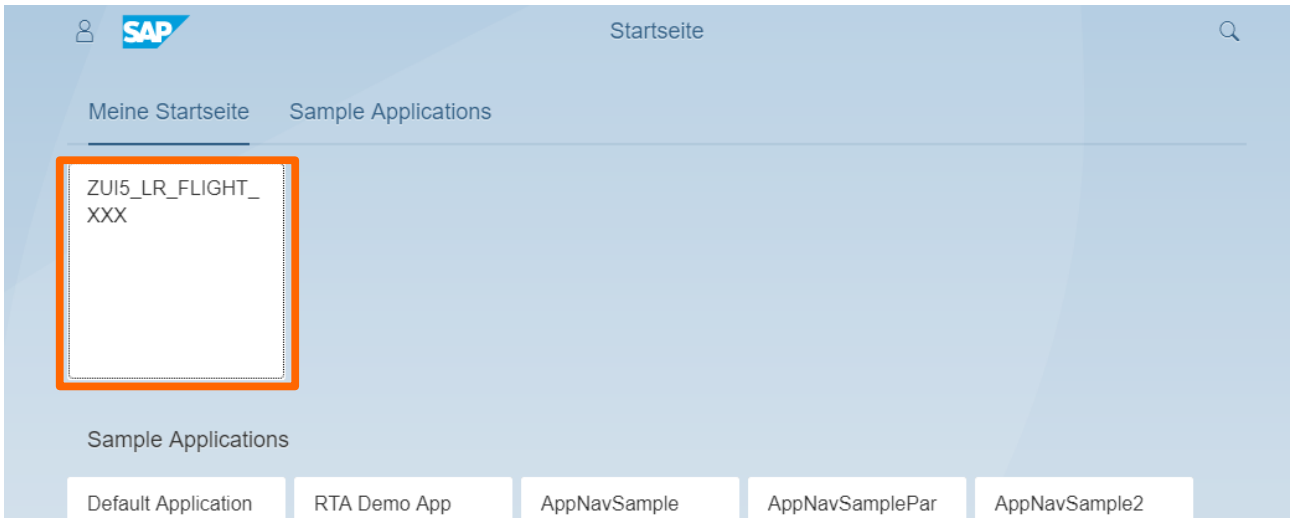


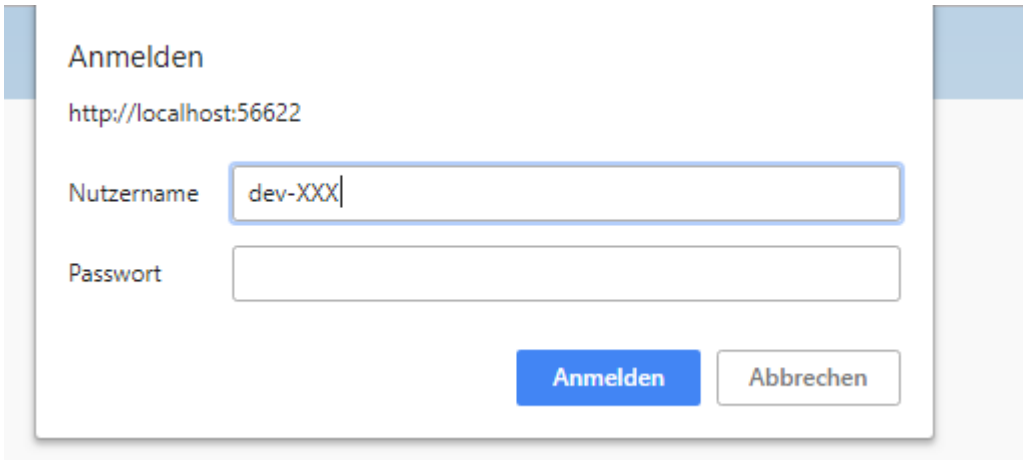


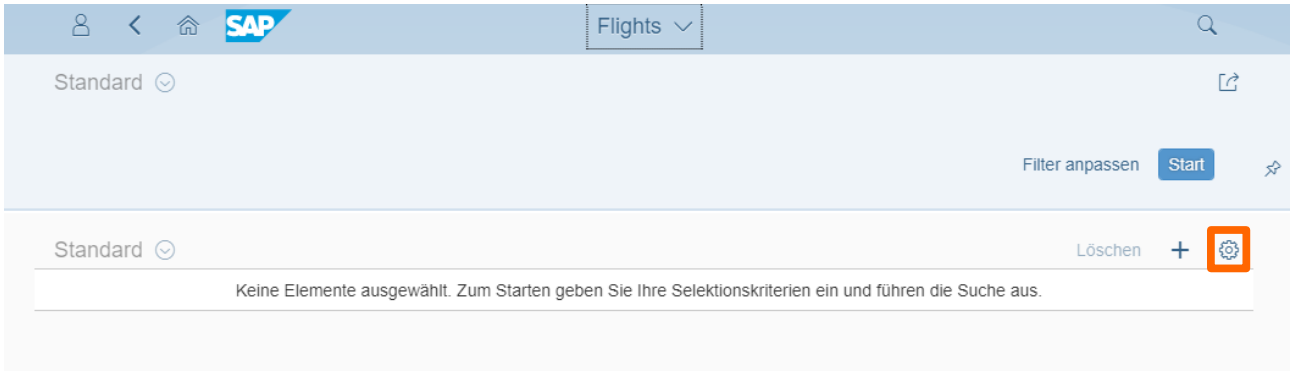
..for showing the flights in the first list and the bookings for each flight in a second list.


ZUI5_LR_FLIGHT_XXX
New
Import
Export
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Rename F2
Delete Delete
Run
Git
Deploy
Project Settings
Run Configurations ...
Run as
Run with Mock Data
Web Application
Unit Test
SAP Fiori Launchpad Sandbox

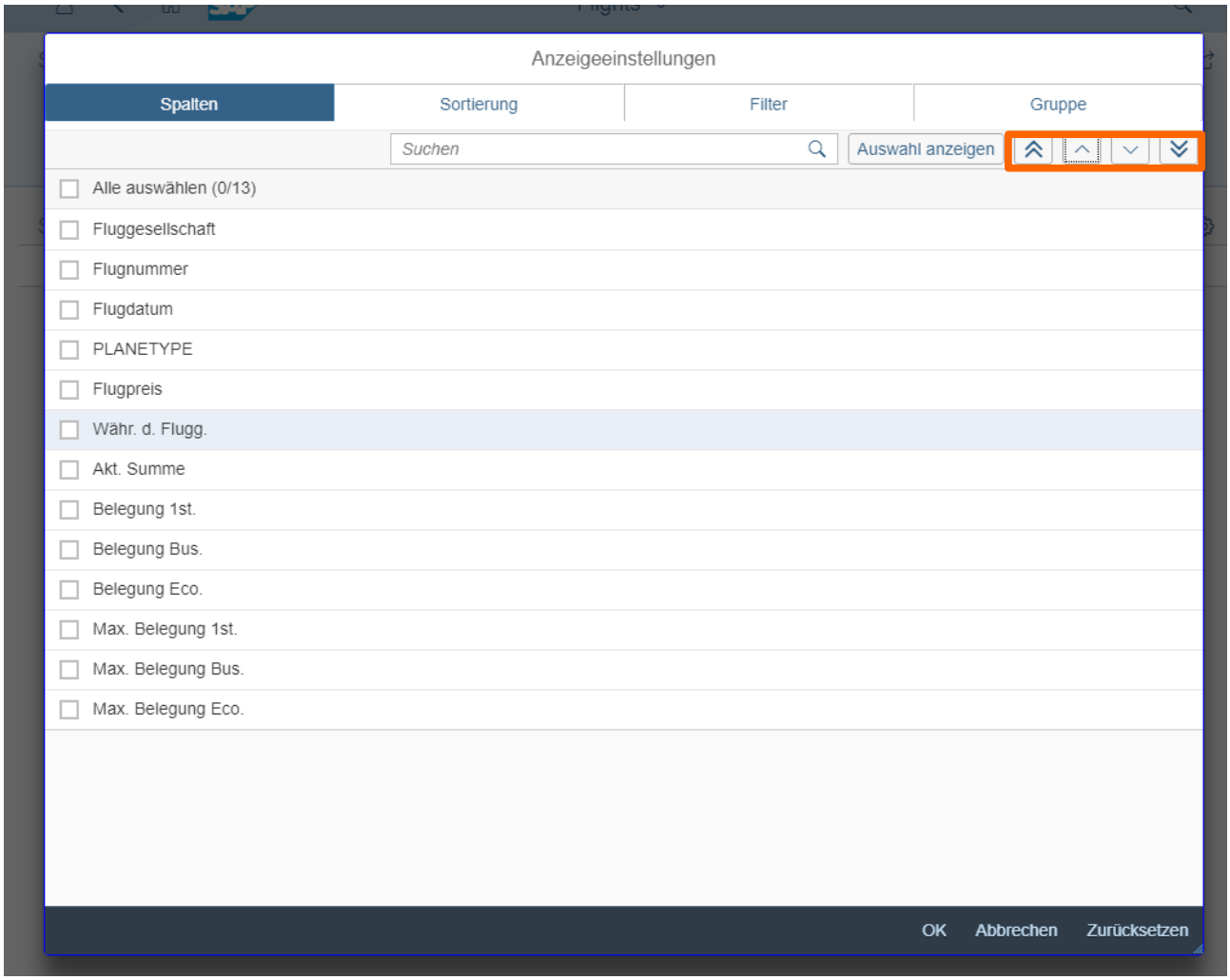

Choose the File to Run
File Name
File Path
flpSandbox.html
/ZUI5_LR_FLIGHT_XXX/webapp/test/fl...
flpSandboxMockServer.html
/ZUI5_LR_FLIGHT_XXX/webapp/test/fl...
OK
Cancel


Startseite
Meine Startseite
Sample Applications
ZUI5_LR_FLIGHT_XXX
Sample Applications
Default Application
RTA Demo App
AppNavSample
AppNavSamplePar
AppNavSample2



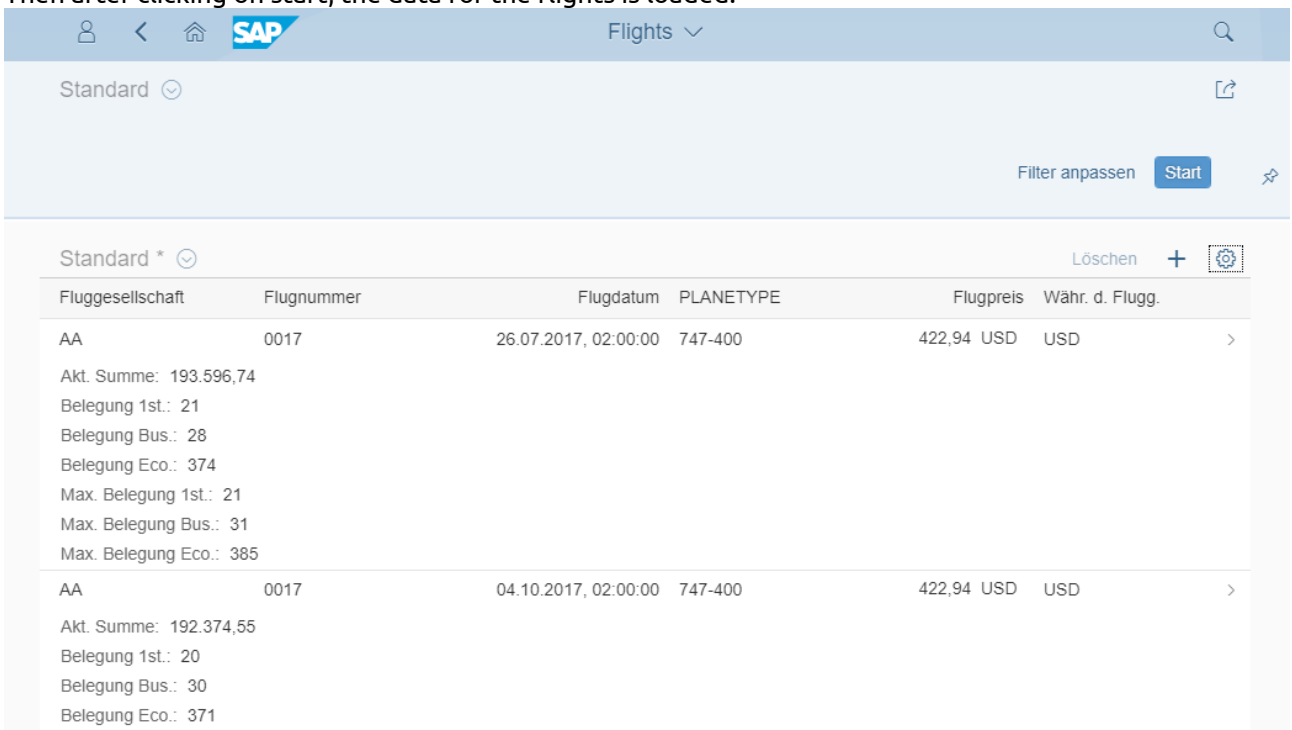


Select the wheel-icon to choose the data-columns/attributes to show.

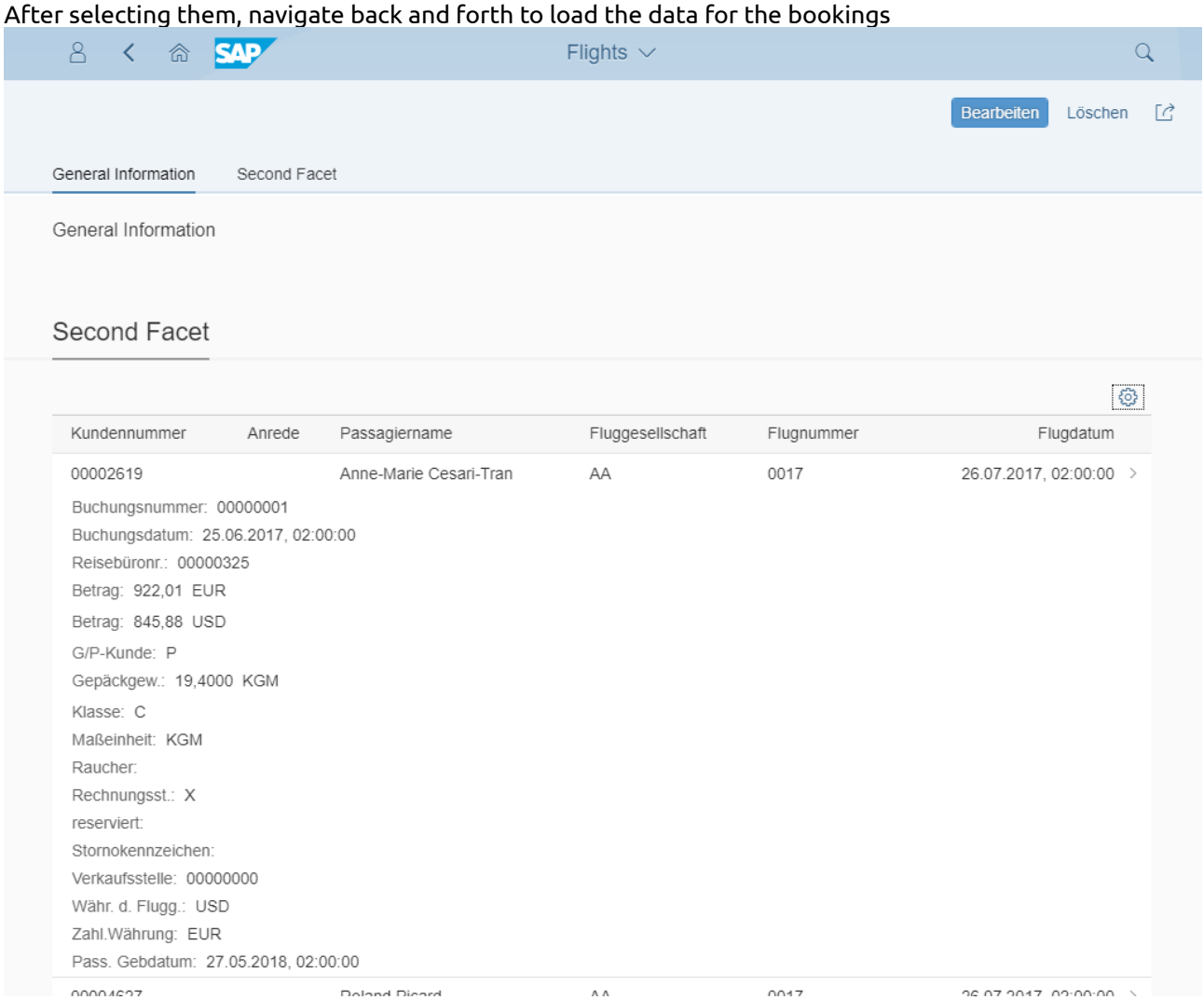


You can also arrange the sequence of the attributes

Then after clicking on start, the data for the flights is loaded:

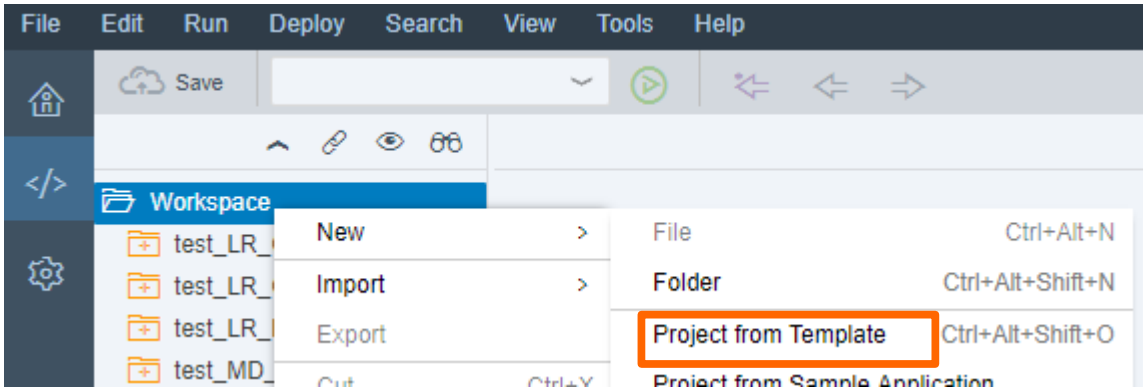


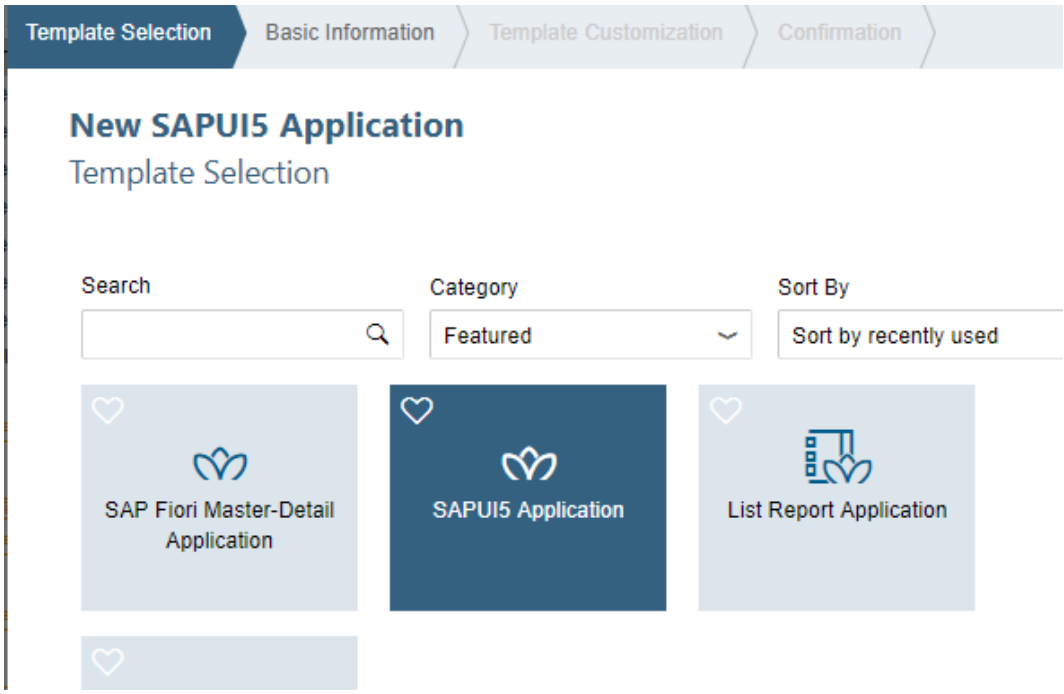
On the second screen you can also select the attributes to show.
After selecting them, navigate back and forth to load the data for the bookings

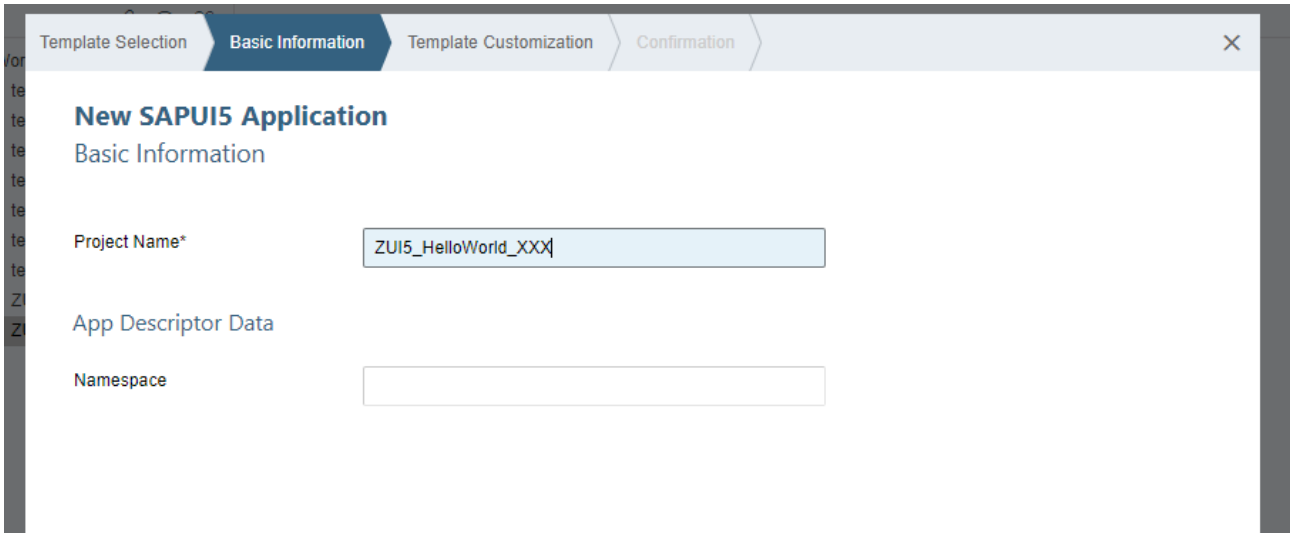


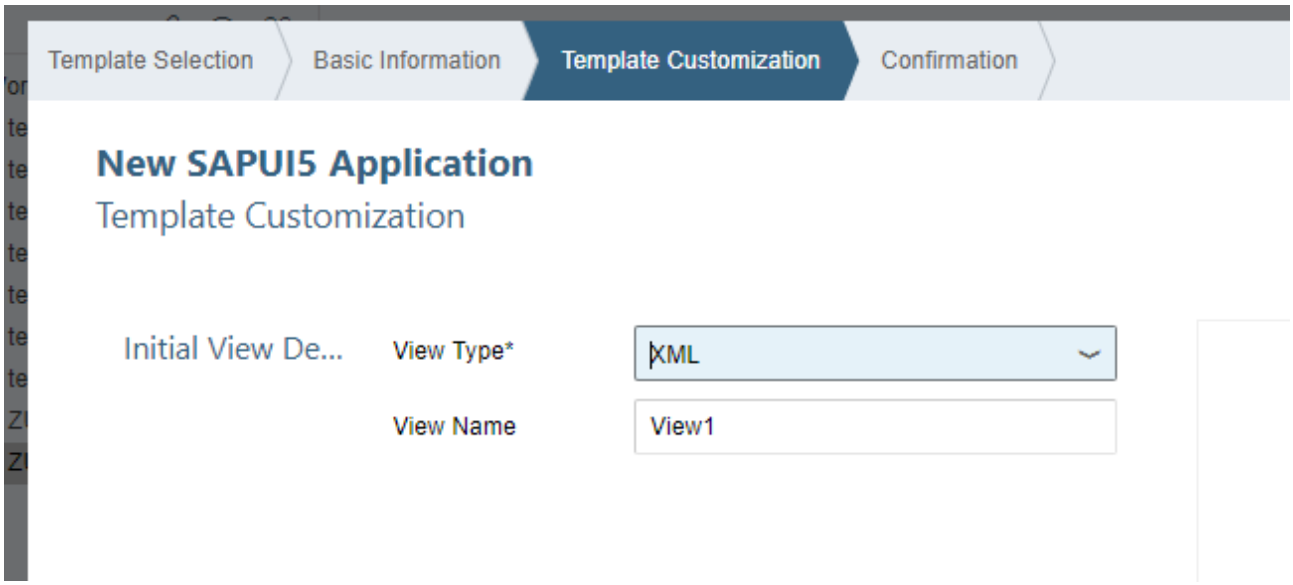
→ *Create a screenshot of the flights view and the booking view and collect it in the result document [4P]*

6 Second: Create a UI5 Application “HelloWorld”



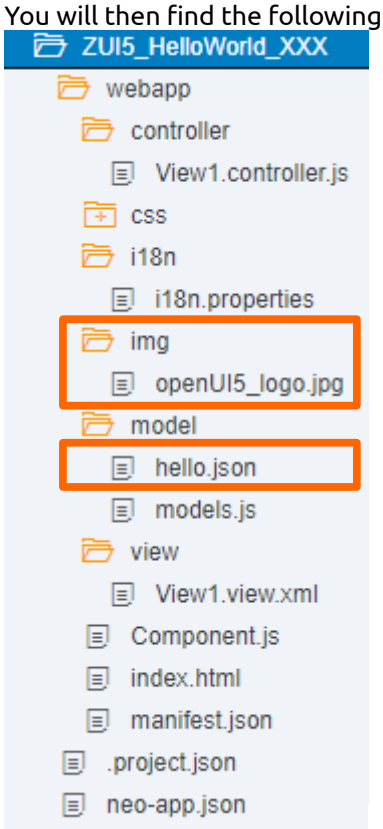






Choose XML for the view type.

You will then find the following structure:



The orange files need to be created as follows.

1. Create folder `img` and import logo-image (from WebIDE-Directory)

2. Create `hello.json` in model folder, using your name

hello.json

```
{
    "user" : {
        "name" : "Vorname Nachname"
    }
}
```

3. To use the `hello.json` as model, define the usage in `manifest.json` (blue)

manifest.json

```
        },
        "models": {
            "i18n": {
                "type": "sap.ui.model.resource.ResourceModel",
                "settings": {
                    "bundleName": "ZUI5_HelloWorld_MSC.i18n.i18n"
                }
            },
            "hello": {
                "type": "sap.ui.model.json.JSONModel",
                "uri": "model/hello.json"
            }
        },
        "resources": {
```

4. Add translatable texts in `i18n.properties`

i18n.properties

```
title=My Title
appTitle = App Title
appDescription=App Description

/* hello example */
helloButtonText=Say Hello!
helloMsg=Hello {0}
```

Now we are prepared for the next steps.

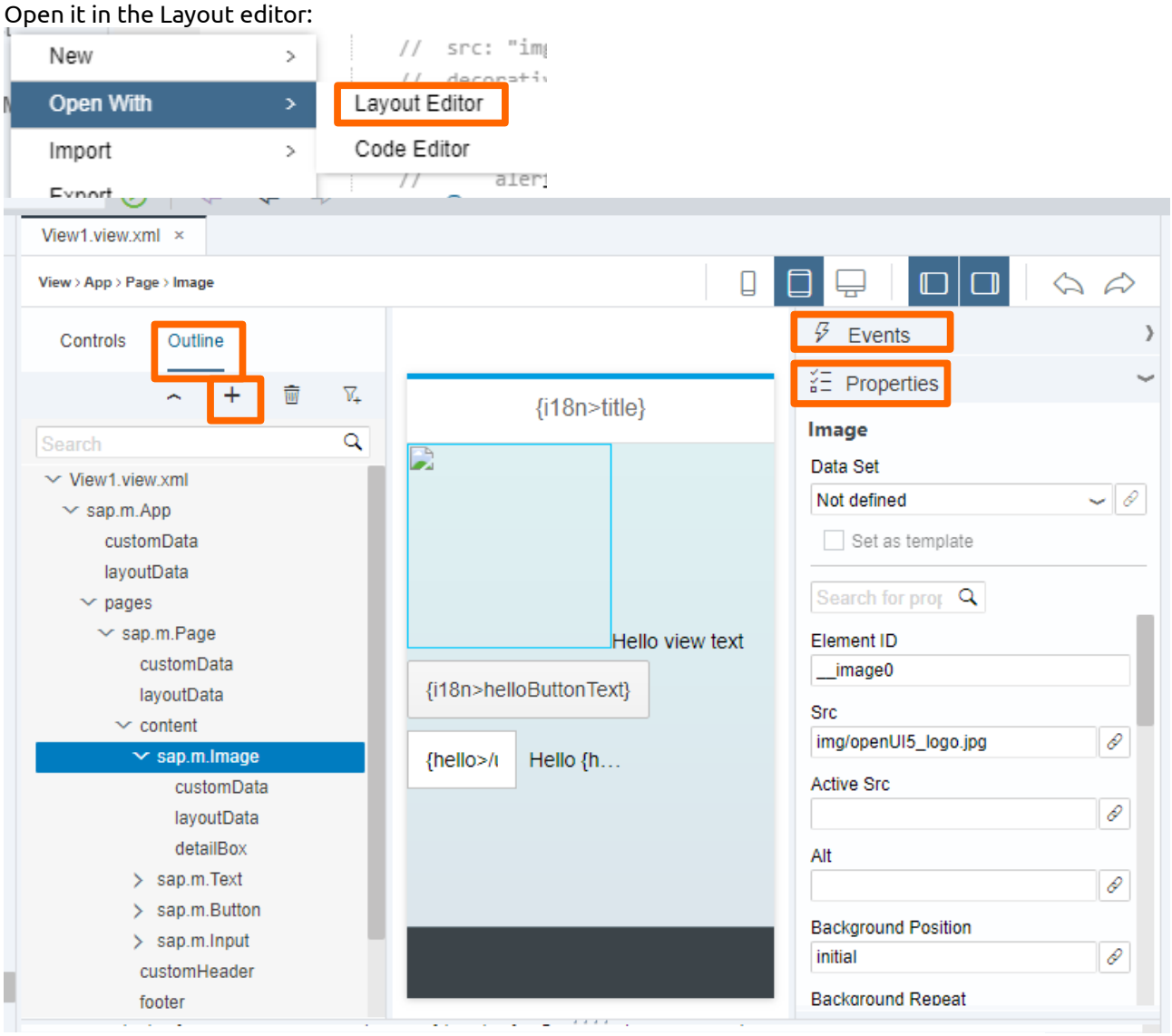
First we will only use the `index.html`. First comment out the part called “third:” and add the blue code.

```
20        <script>
21            sap.ui.getCore().attachInit(function() {
22
23                /* third: comment first and second */
24                new sap.m.Shell({
25                    app: new sap.ui.core.ComponentContainer({
26                        height : "100%",
27                        name : "ZUI5_HelloWorld_XXX"
28                    })
29                }).placeAt("content");
30
31                /* second: display an image with click-handler */
32                var oImage = new sap.m.Image({
33                    src: "img/openUI5_logo.jpg",
34                    decorative: false,
35                    alt: "SAPUI5 Logo",
36                    press: function(){
37                        alert("Hello User XXX :)");
38                    }
39                }).placeAt("content");
40
41                /* first: write some text */
42                var oTxt = new sap.m.Text();
43                oTxt.setText("Hello UI5 World \n on a new line");
44                oTxt.placeAt("content");
45
46            });
47        </script>
48    </head>
```

Test it by running it as web application. It should show the text and the image. Clicking on the image, it should show the message as popup.

→ *Create a screenshot after executing and collect it in the result document [3P]*

Then comment out the blue code and activate the part called “third:”. Now we continue preparing the view.
Open it in the Layout editor:



Add an image control and a text control. The code marked blue below will be added in the view.

Reference: https://sapui5.hana.ondemand.com/#/api

Add a press event to the image element and create a new function `onShowHelloView` for that. Implement the function that has been added in the controller (marked blue below) and remember to add the MessageToast module in line 3 and 4 of the controller.

Test it by running it as web application. It should show the text and the image. Clicking on the image, it should show the message as popup and then as MessageToast.

→ *Create a screenshot after executing and collect it in the result document [6P]*

View1.view.xml ×

```xml
<mvc:View xmlns:html="http://www.w3.org/1999/xhtml" xmlns:mvc="sap.ui.core.mvc" xmlns="sap.m" controllerName=
    <App>
        <pages>
            <Page title="{i18n>title}">
                <content>

                <Image width="140px" height="140px" id="__image0" src="img/openUI5_logo.jpg" press="onShowHel
                <Text text="Hello view text" id="__text5"/>

                <Button
                    text="{i18n>helloButtonText}"
                    press="onShowHelloButton"/>
                <Input
                    value="{hello>/user/name}"
                    description="Hello {hello>/user/name}"
                    valueLiveUpdate="true"
                    width="60%"/>

                </content>
            </Page>
        </pages>
    </App>
</mvc:View>
```

*View1.controller.js ×

```javascript
sap.ui.define([
    "sap/ui/core/mvc/Controller",
    "sap/m/MessageToast" /* loading MessageToast module*/
    ], function(Controller, MessageToast) {
    "use strict";
    return Controller.extend("ZUI5_HelloWorld_MSC.controller.View1", {
        /**
         *@memberOf ZUI5_HelloWorld_MSC.controller.View1
         */
        onShowHelloView: function() {
            //This code was generated by the layout editor.
            alert("Hello XXX from view controller");
            MessageToast.show("Hello nice World");
        },

        onShowHelloButton : function () {
            // read msg from i18n model
            var oBundle = this.getView().getModel("i18n").getResourceBundle();
            var sHello = this.getView().getModel("hello").getProperty("/user/name");
            var sMsg = oBundle.getText("helloMsg", [sHello]);
            // show message
            MessageToast.show(sMsg);
        }
    });
});
```

Then add the code for the button and input element in the view and add the function `onShowHelloButton` in the controller (code shown above)

Test it by running it as web application. It should a new button, input field with the text from the json-model and its (dynamically updating) description text to the right.

Type “Bern” after the content in the input field.

Clicking on the button should show a greeting message as MessageToast.

→ *Create a screenshot after executing and collect it in the result document [6P]*