1 Goal of this tutorial

Create an Application for navigating between two views (Master and Detail),
extend it to consume the OData-Service to read and show the list of employees on the Master view.

Then add functionality to show the details for a selected employee in the Detail view, add icons, i18n and functionality to create, save and delete data entered in the Detail view.

Finally test it locally in the IDE, upload it to the ABAP server and test it again.

Technically you will use the following building blocks of a UI5 application: index.html as starting page, component with manifest.json for application configuration, view (with sections for app, page and content with controls like table, form and buttons) and its assigned controller, model consuming a remote OData-service from SAP Netweaver backend, routing and navigation between views passing parameters, binding of data to UI controls (aggregation binding for the list and element binding for a single entry), using SAP icons, message and i18n for static texts.

2 General

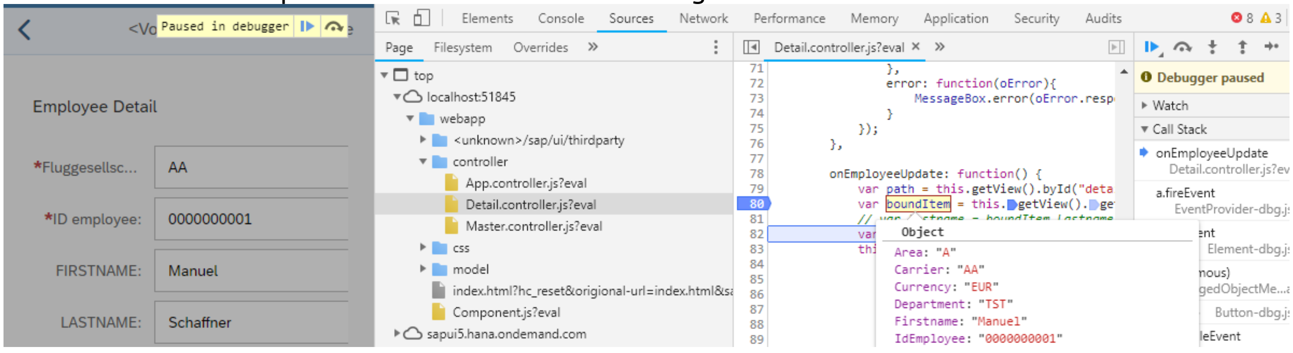
In the following examples replace XXX by your id and Vorname / Nachname by your name – you can use all materials, internet, work in teams of two, but you have to create your own results.

Collect the artefacts of your result as requested below in a result document `SAPUI5_02_<name>_<vorname>.docx` [Total: 33P].

Export the project from WebIDE into a zip-file.

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

You can use the development tools of chrome to see things at work


Employee Detail

*Fluggesellsc...: AA
*ID employee: 0000000001
FIRSTNAME: Manuel
LASTNAME: Schaffner

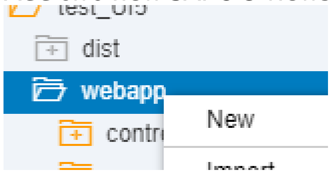
```
            },
            error: function(oError){
                MessageBox.error(oError.resp
            }
        });
    },

    onEmployeeUpdate: function() {
        var path = this.getView().byId("deta
        var boundItem = this.getView().ge
```

```
Object
  Area: "A"
  Carrier: "AA"
  Currency: "EUR"
  Department: "TST"
  Firstname: "Manuel"
  IdEmployee: "0000000001"
```

3 First: Create Application with Navigation

Create a new project from template “UI5 Application”, name it `ZUI5_EMP01_XXX` and name the view `App_XXX`.

Add two new SAPUI5 views by right-clicking on the webapp-folder:



One called `Master` and one called `Detail`.
The corresponding controllers will be added automatically.

Change the code in the **App_XXX.view.xml** as follows:
Replace everything inside the mvc-Tag by `<App id="app"/>`

Change the code in the **Master.view.xml** as follows:

1. Change the following attributes in the existing `Page` tag:
   `title="{i18n>masterViewTitle}"`

2. Add a button with press-event `onNavToDetail` to the content

Change the code in the **Master.controller.js** as follows:
Add/implement the event handler function `onNavToDetail` for the navigation event.
The number for the `detailId` is random and will be replaced later

```
 onNavToDetail : function (oEvent){
      var oId = 1;
      this.getOwnerComponent().getRouter().navTo("detail", {
            detailId : oId
      });
 }
```

Change the code in the **Detail.view.xml** as follows:
Add the following attributes to the existing `Page` tag:

```
title="{i18n>detailViewTitle}"
showNavButton="true"
navButtonPress="onNavToMaster"
```

Change the code in the **Detail.controller.js** as follows:
Add an event handler function:

```
onNavToMaster: function (oEvent){
      this.getOwnerComponent().getRouter().navTo("master");
}
```

Change the code in **manifest.json** as follows:
Add the following code for defining routing between master and detail view.
Insert after/below the “resources” property:

```
,
"routing": {
      "config": {
            "routerClass": "sap.m.routing.Router",
            "viewType": "XML",
            "viewPath": "ZUI5_EMP01_XXX.view",
            "controlId": "app",
            "controlAggregation": "pages",
            "transition": "slide",
            "async": true
      },
      "routes": [
            {
                  "pattern": "",
                  "name": "master",
                  "target": "master"
            },
            {
                  "pattern": "detail/{detailId}",
                  "name": "detail",
                  "target": "detail"
            }
      ],
      "targets": {
            "master": {
                  "viewId": "master",
                  "viewName": "Master",
                  "viewLevel": 1
            },
            "detail": {
                  "viewId": "detail",
                  "viewName": "Detail",
                  "viewLevel": 2
            }
      }
}
```

Change the code in **Component.js** as follows:
Initialize the router in the `init` function by adding
`this.getRouter().initialize();`

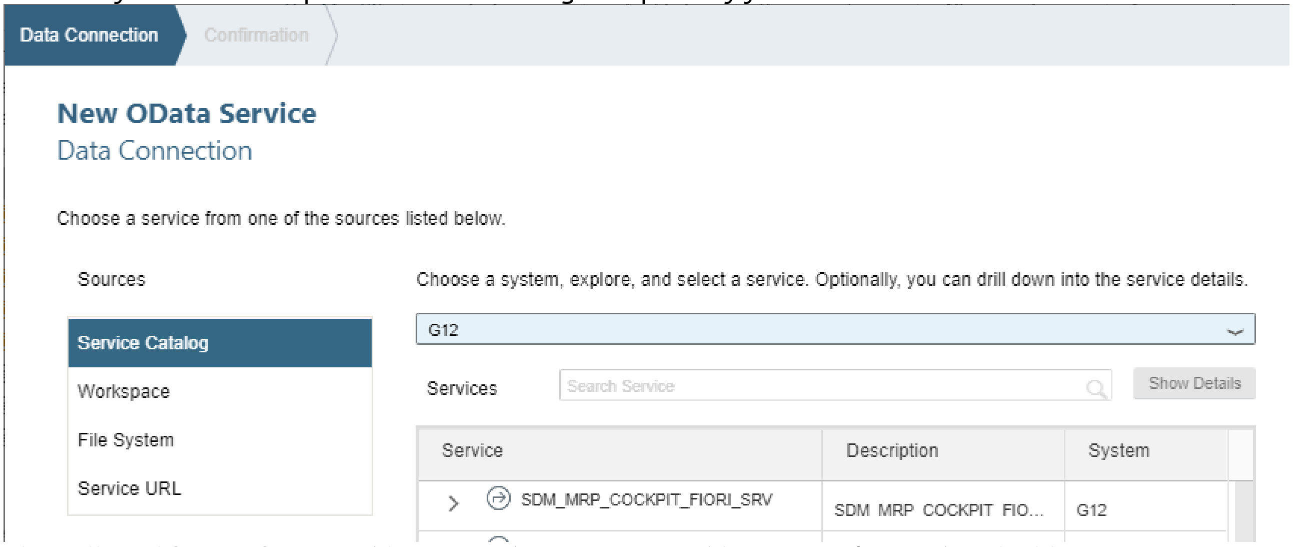
Add the following properties to **i18n.properties** (using your first- and lastname):

```
masterViewTitle=<Vorname> <Nachname>: Master
detailViewTitle=<Vorname> <Nachname>: Detail
```

Test it as Web Application.
→ *Create a screenshot of the master and the detail screen including the URL and collect it in the result document [8P]*

4 Second: Connect Application to Odata-Service and show list of data

Add a new “OData Service” providing the data to the application by right-clicking on the `webapp` folder. Choose the service you have created before in the Backend using ABAP: Z_ODS_EMPLOYEE_XXX_SRV. If your service does not work, you might also use the service ZOD_EMPLOYEE_MSC_SRV as a workaround, but then you have to adapt this in the following codeparts by yourself.


Data Connection
Confirmation

New OData Service
Data Connection

Choose a service from one of the sources listed below.

Sources

Choose a system, explore, and select a service. Optionally, you can drill down into the service details.

Service Catalog
Workspace
File System
Service URL

G12

Services — Search Service — Show Details

Service	Description	System
SDM_MRP_COCKPIT_FIORI_SRV	SDM MRP COCKPIT FIO...	G12

This will modify manifest.json (dataSource) , neo-app.json (destination/system) and add localService/metadata.xml

Change the code in **manifest.json** as follows:
In the models-section add a model `employees` for your datasource.

```
,
                        "Employees": {
                              "type": "sap.ui.model.odata.v2.ODataModel",
                              "settings": {
                                    "defaultBindingMode": "TwoWay",
                                    "useBatch" : false
                              },
                              "dataSource": "Z_ODS_EMPLOYEE_XXX_SRV",
                              "preload": true
                        }
```

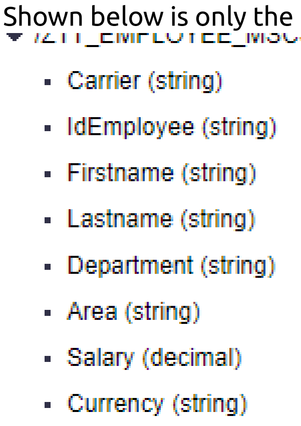
Change the code in **Component.js** as follows:

1. Add usage definition of module `"sap/ui/model/odata/v2/ODataModel"`, and pass it as additional parameter `ODataModel`, to the callback function.

Change the code in **Master.controller.js** as follows:
Make the model available in the view controller by adding access to it in the `onInit` function:

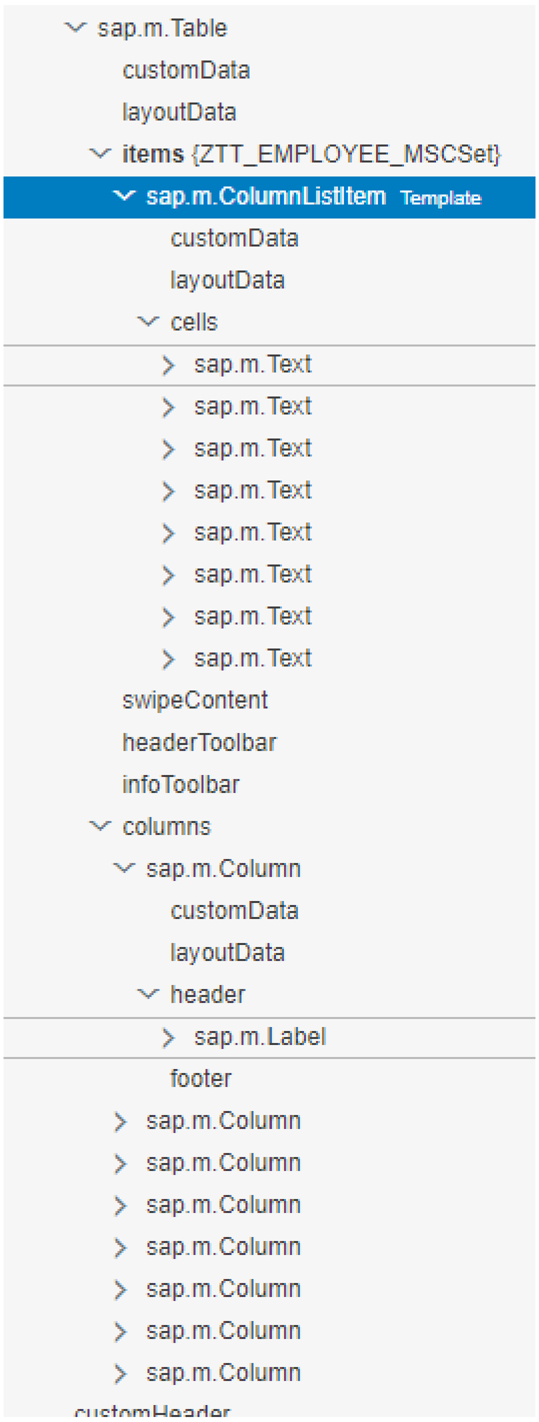
```
onInit: function() {
      this.getView().setModel(this.getOwnerComponent().getModel("Employees"));
},
```

Change the code in **Master.view.xml** as follows:
Add the table to hold the data to the content and bind the item-texts to the model and the label-texts to the label-fields of the model. The of the items list is the one used for the entity set in the OData service.
Shown below is only the code for one header and data column – replace ... by the other columns.



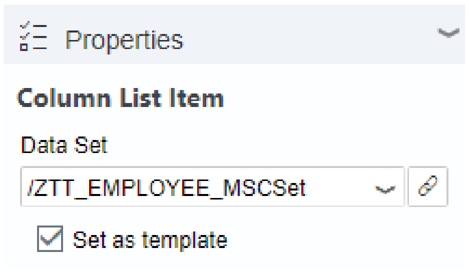
- Carrier (string)
- IdEmployee (string)
- Firstname (string)
- Lastname (string)
- Department (string)
- Area (string)
- Salary (decimal)
- Currency (string)

```
<Table
noDataText="Drop column list items here and columns in the area above"
id="listTable"
items="{/ZTT_EMPLOYEE_XXXSet}"
mode="SingleSelectMaster"
headerText="{i18n>employeeList}"
selectionChange="onNavToDetail">
    <items>
        <ColumnListItem id="__item0">
            <cells>
                <Text text="{Carrier}" id="__text0"/>
...
            </cells>
        </ColumnListItem>
      </items>
    <columns>
        <Column id="__column0">
            <header>
                <Label text="{/#ZTT_EMPLOYEE_XXX/Carrier/@sap:label}"
id="__label0"/>
            </header>
        </Column>
...
     </columns>
 </Table>
```



sap.m.Table
- customData
- layoutData
- items {ZTT_EMPLOYEE_MSCSet}
  - sap.m.ColumnListItem Template
    - customData
    - layoutData
    - cells
      - sap.m.Text
      - sap.m.Text
      - sap.m.Text
      - sap.m.Text
      - sap.m.Text
      - sap.m.Text
      - sap.m.Text
      - sap.m.Text
- swipeContent
- headerToolbar
- infoToolbar
- columns
  - sap.m.Column
    - customData
    - layoutData
    - header
      - sap.m.Label
    - footer
  - sap.m.Column
  - sap.m.Column
  - sap.m.Column
  - sap.m.Column
  - sap.m.Column
  - sap.m.Column
  - sap.m.Column

customHeader



Properties

Column List Item

Data Set: /ZTT_EMPLOYEE_MSCSet

[✓] Set as template

Change the code in **Master.controller.js** as follows:
To set/pass the keys using the URL for the entry to show, add the following code to the existing `onNavToDetail` event handler function (`var oId` is already existing):

```
var oItem = oEvent.getSource().getSelectedItem();
var oCtx = oItem.getBindingContext();
var oId = oCtx.getProperty("Carrier") + '-' + oCtx.getProperty("IdEmployee");
```

Add an `entry` for the employee list (table title) to the **i18n.properties**.

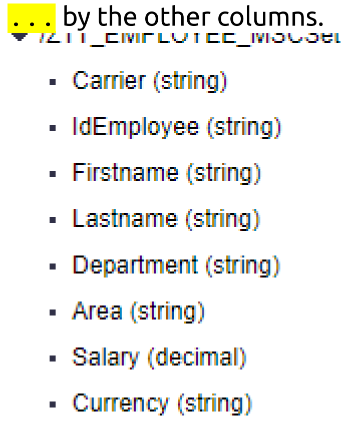
Test it as Web Application.
→ *Create a screenshot of the master screen including the URL and collect it in the result document [9P]*

5 Third: Show detailed data and add CRUD functionality

Change the code in **Detail.view.js** as follows:

Add the namespace for form `xmlns:form="sap.ui.layout.form"` to the View tag.

Add a simple form to the content. Shown below is only the code for one header and data column – replace `...` by the other columns.



- Carrier (string)
- IdEmployee (string)
- Firstname (string)
- Lastname (string)
- Department (string)
- Area (string)
- Salary (decimal)
- Currency (string)

```
<form:SimpleForm editable="true" layout="ResponsiveGridLayout" id="detailForm">
    <form:content>
        <core:Title text="{i18n>employeeDetail}" id="__title0"/>
        <Label text="{/#ZTT_EMPLOYEE_MSC/Carrier/@sap:label}" id="__label10"/>
        <Input width="100%" id="__input0" value="{Carrier}" required="true"/>
...
    </form:content>
</form:SimpleForm>
```

Add a toolbar with buttons below the form. Each button has an event and an icon assigned.

```
<Toolbar id="__toolbar1">
    <content>
        <Button text="{i18n>createButtonText}" id="__button0" icon="sap-
icon://create" press="onEmployeeCreate"/>
        <Button text="{i18n>updateButtonText}" id="__button1" icon="sap-
icon://save" press="onEmployeeUpdate"/>
        <Button text="{i18n>deleteButtonText}" id="__button2" icon="sap-
icon://delete" press="onEmployeeDelete"/>
    </content>
</Toolbar>
```

Change the code in **Detail.controller.js** as follows:

Make the model available in the view controller by adding access to it in the `onInit` function, also add access to the router:

```
onInit: function() {
	this.getView().setModel(this.getOwnerComponent().getModel("Employees"));
	// prepare access to router
	var oRouter = sap.ui.core.UIComponent.getRouterFor(this);
	oRouter.getRoute("detail").attachPatternMatched(this._onObjectMatched,
this);
},
```

Add a function to read the keys passed as arguments on the navigation from the router and bind the corresponding element in the model to the form.

```
/**
 * Read from URL parameter which entry to display and bind it to form
 */
_onObjectMatched: function (oEvent) {
	var aDetailId, selectedItemPath, detailForm;
	aDetailId = oEvent.getParameter("arguments").detailId.split('-');
	// extract keys for entry from arguments <Carrier>-<EmployeeId> in URL
	selectedItemPath = "/ZTT_EMPLOYEE_MSCSet(Carrier='" + aDetailId[0] +
"',IdEmployee='" + aDetailId[1] + "')";
	// = selection-path: /ZTT_EMPLOYEE_MSCSet(Carrier='X',IdEmployee='Y')
	detailForm = this.getView().byId("detailForm");
	detailForm.bindElement({path: selectedItemPath});
},
```

Add event handler functions for CRUD:

```
onEmployeeCreate: function() {
	var path ="/ZTT_EMPLOYEE_MSCSet";
	var boundItem =
this.getView().getModel().getProperty(this.getView().byId("detailForm").getEleme
ntBinding().getPath());
	var msg =
this.getView().getModel("i18n").getResourceBundle().getText("employeeCreated",
boundItem.IdEmployee);
	this.getView().getModel().create(path, boundItem, {
		success: function(){
			MessageBox.success(msg);
		},
		error: function(oError){
			MessageBox.error(oError.responseText);
		}
	});
},
onEmployeeUpdate: function() {
	var path =
this.getView().byId("detailForm").getElementBinding().getPath();
	var boundItem = this.getView().getModel().getProperty(path);
	// var lastname = boundItem.Lastname; // nok: getProperty()  getValue()
data() oEvent.getSource().data("IdEmployee")
	var msg =
this.getView().getModel("i18n").getResourceBundle().getText("employeeUpdated",
boundItem.IdEmployee);
	this.getView().getModel().update(path, boundItem, {
		success: function(){
			MessageBox.success(msg); // update does not return anything
		},
		error: function(oError){
			MessageBox.error(oError.responseText);
		}
	});
},

onEmployeeDelete: function() {
	var path =
this.getView().byId("detailForm").getElementBinding().getPath();
	var boundItem = this.getView().getModel().getProperty(path);
	var msg =
this.getView().getModel("i18n").getResourceBundle().getText("employeeDeleted",
boundItem.IdEmployee);
	this.getView().getModel().remove(path, {
		success: function(){
			MessageBox.success(msg);
		},
		error: function(oError){
			MessageBox.error(oError.responseText);
		}
	});
}
```

Add the following texts for the messages to **i18n.properties** – they contain a placeholder for the id:

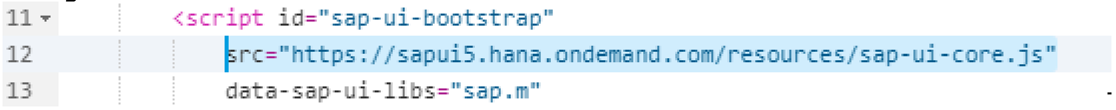
```
employeeCreated=Employee with ID {0} created!
employeeUpdated=Employee with ID {0} updated!
employeeDeleted=Employee with ID {0} deleted!
```

Add an `entry` for the employee detail (form title) to the **i18n.properties**.
Add `entries` for the button texts to to the **i18n.properties**.

Test it as Web Application.
→ *Create a screenshot of the detail and master screen including the URL initially calling the list, then after creating an entry, after changing an entry and after deleting an entry and collect it in the result document [12P]*

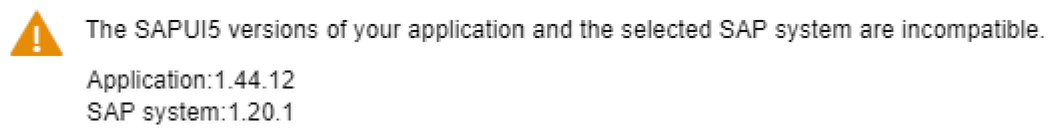
6 Fourth: Upload to SAP Netweaver ABAP

Change the code in the **index.html** as follows:



```
    <script id="sap-ui-bootstrap"
        src="https://sapui5.hana.ondemand.com/resources/sap-ui-core.js"
        data-sap-ui-libs="sap.m"
```

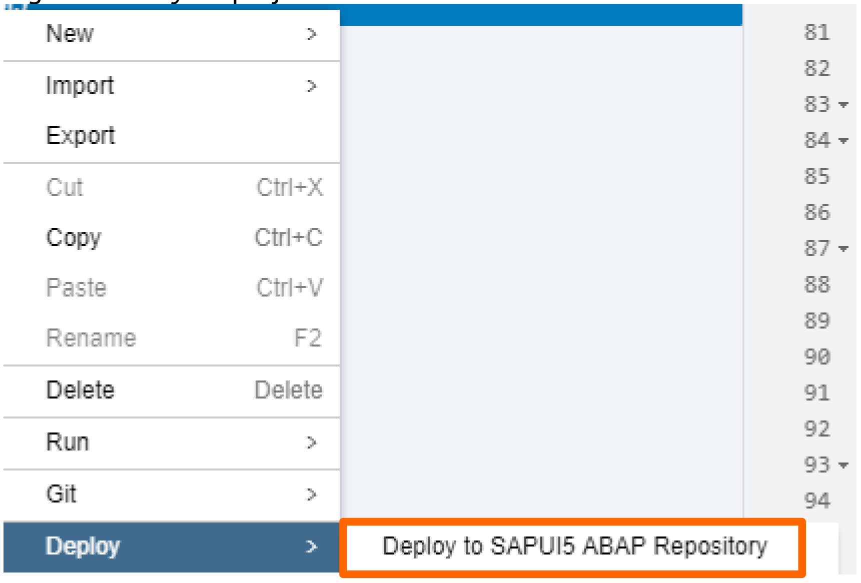
This is necessary, because the release of the libraries for UI5 available on the SAP Netweaver System G12 does not match with those used in the SAP Web IDE:

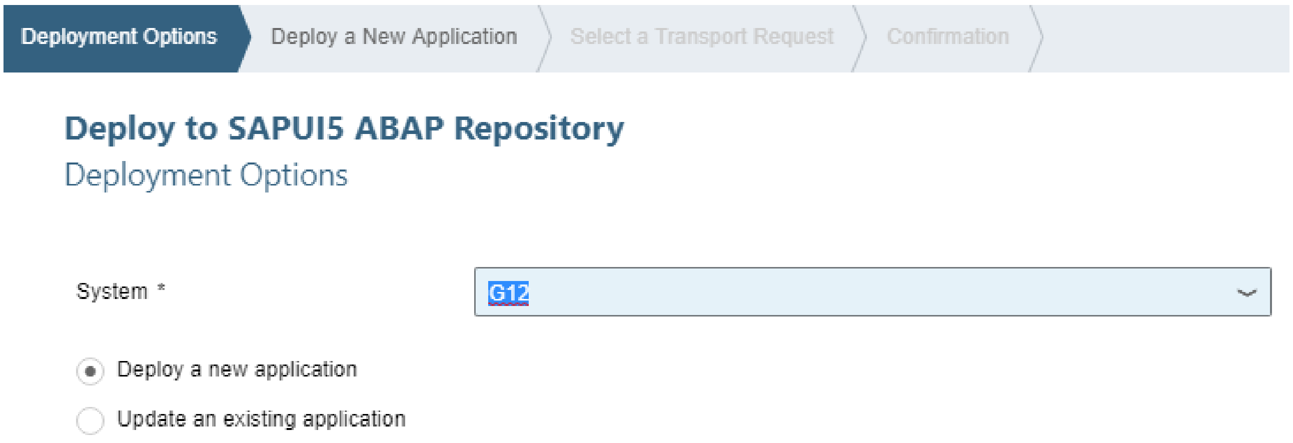


The SAPUI5 versions of your application and the selected SAP system are incompatible.

Application:1.44.12
SAP system:1.20.1

Right click on your project

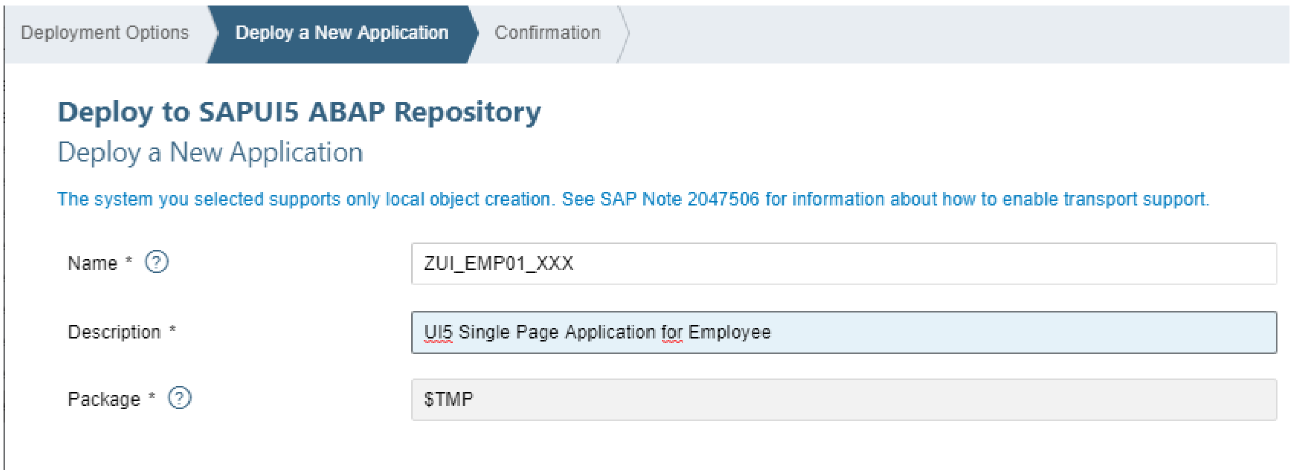




Deploy to SAPUI5 ABAP Repository
Deployment Options

System * G12

(●) Deploy a new application
( ) Update an existing application



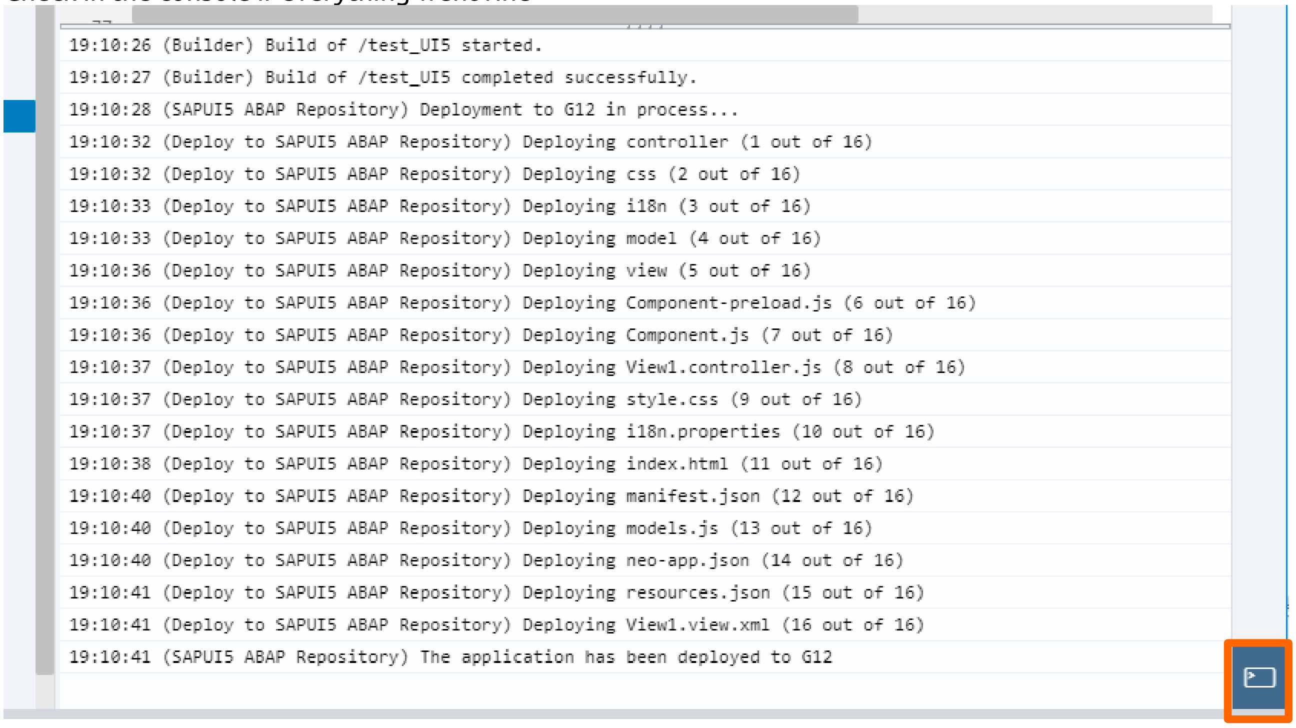
Deploy to SAPUI5 ABAP Repository

Deploy a New Application

The system you selected supports only local object creation. See SAP Note 2047506 for information about how to enable transport support.

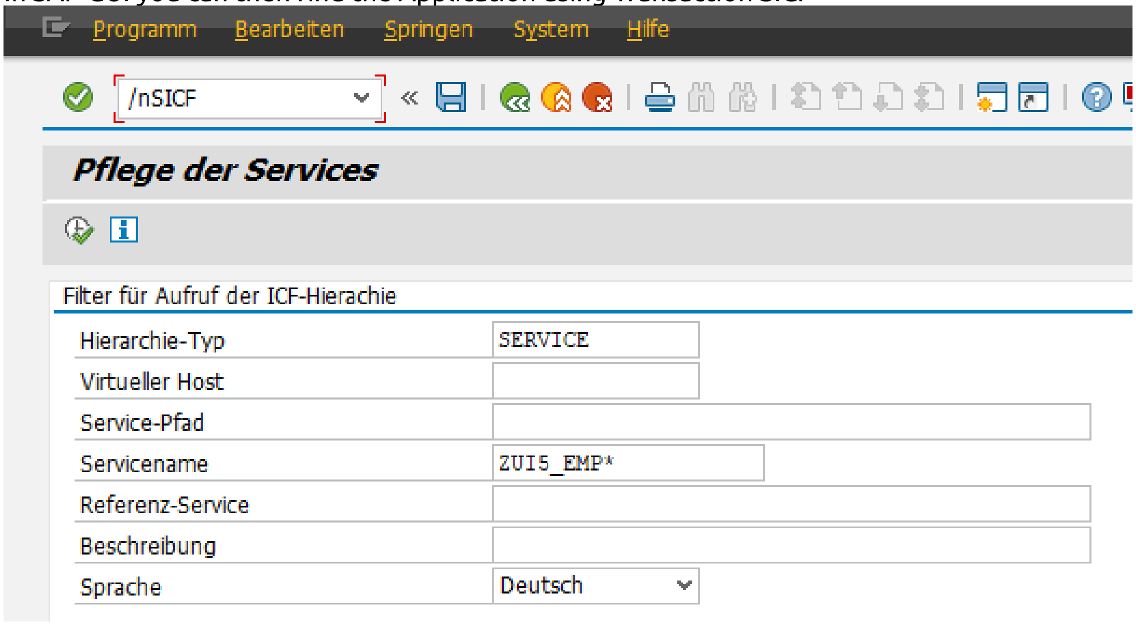
Name *	ZUI_EMP01_XXX
Description *	UI5 Single Page Application for Employee
Package *	$TMP

Check in the console if everything went fine



```
19:10:26 (Builder) Build of /test_UI5 started.
19:10:27 (Builder) Build of /test_UI5 completed successfully.
19:10:28 (SAPUI5 ABAP Repository) Deployment to G12 in process...
19:10:32 (Deploy to SAPUI5 ABAP Repository) Deploying controller (1 out of 16)
19:10:32 (Deploy to SAPUI5 ABAP Repository) Deploying css (2 out of 16)
19:10:33 (Deploy to SAPUI5 ABAP Repository) Deploying i18n (3 out of 16)
19:10:33 (Deploy to SAPUI5 ABAP Repository) Deploying model (4 out of 16)
19:10:36 (Deploy to SAPUI5 ABAP Repository) Deploying view (5 out of 16)
19:10:36 (Deploy to SAPUI5 ABAP Repository) Deploying Component-preload.js (6 out of 16)
19:10:36 (Deploy to SAPUI5 ABAP Repository) Deploying Component.js (7 out of 16)
19:10:37 (Deploy to SAPUI5 ABAP Repository) Deploying View1.controller.js (8 out of 16)
19:10:37 (Deploy to SAPUI5 ABAP Repository) Deploying style.css (9 out of 16)
19:10:37 (Deploy to SAPUI5 ABAP Repository) Deploying i18n.properties (10 out of 16)
19:10:38 (Deploy to SAPUI5 ABAP Repository) Deploying index.html (11 out of 16)
19:10:40 (Deploy to SAPUI5 ABAP Repository) Deploying manifest.json (12 out of 16)
19:10:40 (Deploy to SAPUI5 ABAP Repository) Deploying models.js (13 out of 16)
19:10:40 (Deploy to SAPUI5 ABAP Repository) Deploying neo-app.json (14 out of 16)
19:10:41 (Deploy to SAPUI5 ABAP Repository) Deploying resources.json (15 out of 16)
19:10:41 (Deploy to SAPUI5 ABAP Repository) Deploying View1.view.xml (16 out of 16)
19:10:41 (SAPUI5 ABAP Repository) The application has been deployed to G12
```

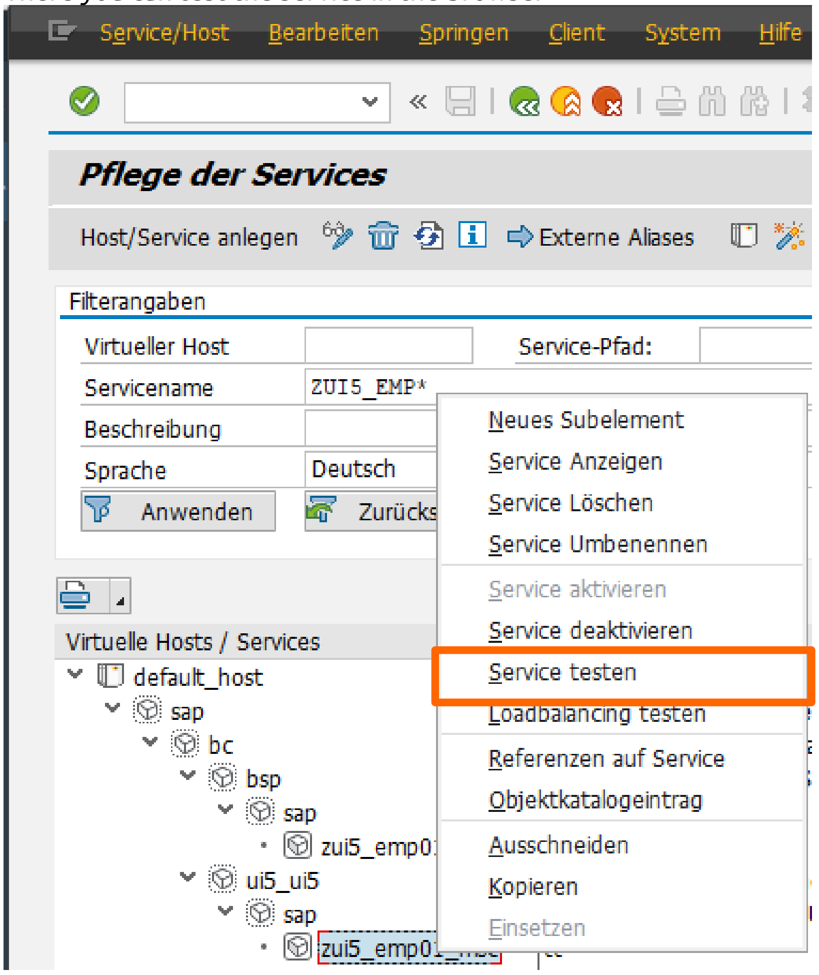
In SAP GUI you can then find the Application using Transaction SICF



Filter für Aufruf der ICF-Hierachie

Hierarchie-Typ	SERVICE
Virtueller Host	
Service-Pfad	
Servicename	ZUI5_EMP*
Referenz-Service	
Beschreibung	
Sprache	Deutsch

There you can test the service in the browser



Filterangaben

Virtueller Host		Service-Pfad:	
Servicename	ZUI5_EMP*		
Beschreibung			
Sprache	Deutsch		

http://vlhsapg12.hevs.ch:8012/sap/bc/ui5_ui5/sap/zui5_emp01_XXX/index.html?sap-client=200

→ *Create a screenshot of the master and detail screen including the URL and collect it in the result document [4P]*