Practice

<https://www.tcodesearch.com/tcodes/search?q=netweaver+gateway+version+tcode>

sicm – services –host name and port no.

1. Launchpad – Group and Catalog created, add a tiles and roles are authorized

* Deploy the app – SA38
* Create Roles – mentioned below a (1st Pt.)
* Create target and semantic object
* Fiori designer – Create catalog and group
* PFCG Roles
  1. Launchpad Configuration
  + Create Launchpad

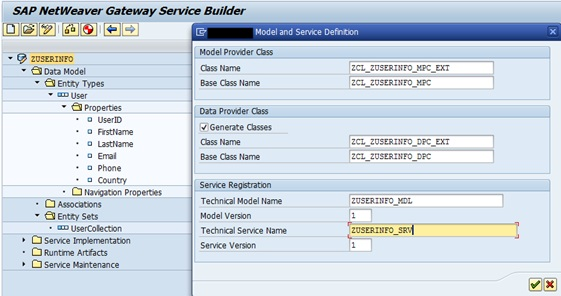
LPD\_Cust – Create Launchpad – Fill All detail – Role, Instance (Type of App) and Description- Write the namespace from component. Click on ok

* + Register Application

Navigated to Different Screen – Click on New Application – Link Text – App URL (will get from SICF), Enter App Alias name (Write any name) – Write Component Name in Additional Text Like – SAPUI5.Component=COmponentname

* + Create TR Customising TR se01
  + /UI2/SEMOBJ/ - to create semantic Object
  + Add Semantic Object – Write Semantic Obj and Description
  + Start Launchpad Designer /UI2/FLPD\_CUST and assign TR, get the package name from se80
  + Create Role with transaction start – PFCG – Write Roles
  + Add Group and Catalog to Roles using PFGC, Menu , Transaction, group and Catalog option selected and enter details respectively

1. How does cross app navigation work?s
2. Tell us about any Smart Control
3. How does smart filter works on smart table
4. Abap All Functions--- <https://blogs.sap.com/2014/03/06/let-s-code-crudq-and-function-import-operations-in-odata-service/>
   1. Go to SEGW
   2. Create Project
   3. Import a table using import – DDIC structure (Data dictionary)
   4. Enter Field wrt to field path, map key, enter object (which will be entity name)(User)
   5. Create Entity set by adding create on right click of entity set. Provide Entity Set name (eg: UserCollection) and also enter entity name(User) as created earlier
   6. Generate Run Time Artifacts- by clicking on generate runtime objects. Popup arrives, keep all the classes name default and save

In short 4 classes will be generated DPC’s and MPC’s. Copy Technical Service Name

* 1. Register service name click on register by clicking on folder Service Maintenance
  2. On successful registration, will open service catalog to call gateway client. Click on gateway client u can test the service (it can be done using /IWFND/GW\_CLIENT)
  3. Amend $metadata and execute to test the service
  4. QUERY: Right click on the dpc\_ext class and select ‘go to abap workbench’, select edit mode and right on get\_entityset and select redefine method

1. - T = internal table
2. - S = structure
3. - D = field

--- SE11 – Create or Vi

ew or edit Structure

--- SEGW to create Odata object - Create Entity – Create Entity Set- Create Run Time objects

---SE24 for checking MPC and DPC class

--IWFND/MAINT\_SERVICE Activate and Maintain Service – Add Service – (For Hub System and Backend System is Different maintain backend system name in system alias)- list of all service will be visible – double click on service to register or add service- go to SICF node

Gateway client inside add service page or /IWFND/GW\_CLIENT to test all service

-- SICF – Create service in SICF space – ie path – sap/opu/odata/sap/service\_Name

-- SMICM – for host and port no.

-- spro to check system alias – SAP Netwaver – configuration – Connection Setting – Manage Sap System Alias – List of all sap alias

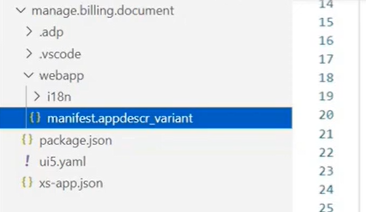
* Sa38 – to run a program
* Se16 – for table content

1. Extension Point----

<https://learning.sap.com/learning-journeys/develop-sapui5-applications/implementing-view-extension-modification-and-replacement_e990e88c-9483-48d6-98fd-cf60c97adf73>

BAS:

* Your manifest.json file need to have the correct settings for flexEnabled property and viewID target properties.
* Create Adaptation Project from Template
* Select necessary details like project name, extended project name,



* Config.json will be created with all imp information about the project, here we can change the sapui5 version as well
* The file manifest.appdescr\_variant is the manifest-file for the application variant, it contains the App id of the application variant. Right click and select SAPUI5 visual editor and will be able to see the view of the project with the possibilities of extension points

1. Learned about Creation of Association and Navigation properties of OData services, Extension to Standard Fiori/UI5 Application.
2. Viz Frame charts with SAPUI5 Application
3. Cloud ---

Create MTA app- Create Project from Template- MTA Application – mta.yaml file will be created which wlll contain all th configuration- right click on MTA.mtar- create MTA module from Template – approuter for CAP module and Sap Fiori Geneartor for SAPUI5 application

Build is necessary since It creates mtar file which is used for deployment

Create Multi target application, inside create html5 module, web app folder will include html application, Mta file will be generated,

Click on Application name, select project setting, then space, enter endpoints, space and organization then install builder,

This will install builder in the space and new application will be generated in CF space

Click on Build and new folder with project name will be generated and inside will include .mtar file which will be used to deploy in SCP. Right click on .mtar file, deploy, select deploy to Cloud foundry, Select space, endpoints and organization and done.

Inside CF cockpit, new application will be generated and when u click on that u will get application’s url link, click on view the application

1. HAT-----
2. Cordova----
3. Offline App and Data---

--Write sap.Mobile function in manifest..add

1. MockServer
2. Manifest and neo-app json
3. [Link to Docs](https://ind01.safelinks.protection.outlook.com/?url=https%3A%2F%2Flntinfotech.sharepoint.com%2Fsites%2FSAP%2Flearning%2FSAP%2520UI5%2520and%2520SAP%2520Fiori%2520Certification%2520Course%2FForms%2FAllItems.aspx%3Fviewid%3D852b22ea-316e-47f8-8ec4-c8fcd2d665c9&data=04%7C01%7CRicha.Mishra%40lntinfotech.com%7Cd05b5d653aee44d44c6c08d8b6fd6755%7C02aa9fc118bc4798a020e01c854dd434%7C0%7C0%7C637460546762354655%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C1000&sdata=FnnXDkbhXCoYIgOB8NyUryye1TcqRSP7G8%2BYJUUMuZ4%3D&reserved=0)

Refer to Format Date: <http://userguide.icu-project.org/formatparse/datetime>

1. Fiori App Development on SCP
2. Cloud Application Programming (CAP) with Node.js - https://developers.sap.com/tutorials/xsa-cap-create-project.html
3. HANA XSJS
4. SAP HANA Service
5. SCP Workflow Service
6. Cloud Transport Management
7. Cardova Plugins
8. Developing Apps with Offline Capability
9. Job Scheduling Service
10. To check version of SAP – Ctrl+Shift+Alt+P
11. Develop new SAP Fiori and SAPUI5 apps
12. Extend SAP Fiori apps
13. Develop SAPUI5 mobile hybrid apps (HAT plug-in)
14. Extend the SAP Web IDE with new plug-ins and templates
15. Basic Knowledge about common business process, SAP Workflow, ABAP programming, Basis and Authorization
16. What is SAP ERP
17. Installation of Gateway
18. Installation of Fiori components ie Procurement Approval, Leave Booking, Timesheet options in gateway
19. Activation of Fiori components in gateway ERP and SRM
20. Installation and configuration of Launchpad
21. Activate OData components in gateway
22. Extending, customizing, implementing Fiori standard components
23. Sap cloud platform and Fiori architecture on cloud and on premise deployment
24. Deploy Fiori in existing SAP Infrastructure (on premise deployment)

* Go to SE38
* Execute ‘/UI5/UI5\_REPOSITORY\_LOAD’
* Enter Name of SAPUI5 Application
* Upload
* Enter Package and TR
* Verify Deployed Application by entering sicf and search for service with application name go to files path ‘default\_host/sap/bc/ui5\_ui5/sap’. Right Click and Activate

1. How to update sapui5 version in ABAP

* Check the version
* Download the UI5 from the marketplace.
* Start transaction SA38 and execute program /UI5/UI5\_UPLOAD\_PATCH\_TO\_MIME
* Upload the downloaded file
* Execute the report first in test mode.
* And then to Dev by using TR

<https://community.sap.com/t5/technology-blogs-by-members/sap-ui5-1-60-patch-upgrade-abap/ba-p/13442032>

OPA Test:

1. **To perform a QUnit test, wha……………….k.kt do you include in a test page under your webapp folder?**  
   SAPUI5 bootstrap code
2. **What are the limitations of OPA5 test?**  
   Testing across more than one page is NOT possible

End-to-End tests are NOT recommended

1. **While testing an SAP Fiori app you discover that a button on the page delivers no response. Which of the following agile pyramid options do you use?**

**O**PA

1. **What activity does the QUnit Stub functionality perform?**

Returns values for method calls

1. **You need to explain the core functions of OPA5 for testing SAPUI5 apps. Which test options are offered by OPA5?**

Navigation tests

User interaction tests

1. **While testing an SAP Fiori app you discover that a button on the page delivers no response. Which of the following agile pyramid options do you use?**

**OPA**

1. **You perform a QUnit test with the following syntactical options. Which call returns true when you execute it?**

Hybrid App Toolkit Connector

1. **Hybrid App Toolkit Connector** which is a local server process that enables SAP Web IDE to connect to the local system's Cordova development environment and allows developers to create and manage a local Cordova project
2. Apache Cordova is a mobile application development framework created by Nitobi
3. **You need to strengthen the connection security between the Web IDE and the Hybrid App Toolkit Connector. Which of the following activities do you perform?**

Replace the default API key

1. **How do you integrate SAPUI5 in a Kapsel/Hybrid app?**

Add the sap-mobile-hybrid.js file to the Hybrid app

1. **Which dependent plugin must you enable to include the Push plugin when you create a hybrid mobile app?**

Dialog Notification

1. **How do you integrate SAPUI5 in a Kapsel/Hybrid app?**

Add the sap-mobile-hybrid.js file to the Hybrid app

1. **Why do you use the Cordova and SAP Kapsel plug-ins when using the SAP Web IDE, SAP HAT add-on?**

To integrate device APIs

To use offline OData

1. **Which file of the SAP HAT Connector stores the WebIDEHosts URL and can be used for troubleshooting?**

Config.json

Web IDE

1. Which properties do you need to configure when you use the Mock server?

Service Url & Module path

1. **Why does SAP Fiori use an intent-based navigation?**

Apps targets can be deployed separately

1. **Which UI5 theme is replacing SAP Blue Crystal as the base Fiori theme?**

SAP Belize

1. **You need to implement an SAP Fiori transactional app for your customer. What benefits are available in the completed app?**

It allows UI changes without development privileges in the back-end

It allows access to the back end to edit document details

1. **Which information is included in the SAPUI5 app’s manifest.json file?**

App ID, Name and Version

SAPUI5 dependencies

Referenced data sources

1. **You need to bind data from a model to an SAPUI5 view control. Which of the following models are valid?**

one-way binding

two-way binding

one-time binding

1. **Which activities does the OData Model Editor support?**

Define Odata models

Test Odata models

Edit Odata models

1. **Your customer needs to securely connect the SAP HANA Cloud Connector to the Web IDE of an on-premise system. How does the SAP HANA cloud connector help do this?**  
   It creates connectivity by a reverse-invoke process on the on-premise system

It secures an SSL tunnel between the SAP Hana Cloud and the on-premise system

1. **Your customer requires an app to display flight information on a mobile device. Please see the screenshot for details. Which options can you use to display the same data in a readable format on a mobile device?**

column : <Column demandPopin=”true” minScreenWidth=”tablet”>

column: <Column demandPopin=”true” minScreenWidth=”tablet” popinDisplay=”Inline”>

1. **In the source code below, which of the following locations must reference the extension point?**  
   Please choose the correct answer.

<code:View xmlns:core=”sap.ui.core” xmlns:mvc=”sap.ui.core.mvc” xmlns=”sap.m” controllerName=”appextension.EmployeeData” xmlns:html=http://www.w3.org/1999/xhtml>  
<Page title=”Title”>  
<content>  
<Label text = “First name:”/>  
<Input/>  
<core:ExtensionPoint name=”forMiddleName:”/>  
<<Label text = “Last name:”/>  
<Input/>  
<Button text=”Get Info” press=”doGetInfo”/>  
</content>  
</Page>

Component.js

1. **Why do you explore the Problem Space in detail during a Design Thinking workshop?**

To understand user needs

1. **You develop an SAPUI5 app that can be extended. To allow your customer notifications, you set up a hook method. What activities are required to implement a hook method?**

**\*\*Hook methods** provide a way to extend behavior of programs at runtime

Identify a strategic location in the controller for the customized code

Add a code snippet to check if the method exists and execute it  
Define a new function name ensuring it is reserved for the extension

1. **What are the basic guidelines of the Extension Point concept of SAPUI5?**

The custom app and the parent app contain a component.js file

The custom app is located in a separate project

1. **Which SAP Fiori personalization elements holds all of the technical information needed to start an app?**

Tile

1. **What is the purpose of the SAP HANA Cloud connector?**

It connects the services of the SAP HANA Cloud Platform and on-premise system in the customer’s network

1. **In the design thinking phase of your customer project, one design is accepted as viable. What other attributes need to be valid for this design to be considered?**

Desirable & Feasible

1. **Which tile types can you use to display real-time data values read from an OData service?**

App Launcher – Dynamic tile

KPI based tile

1. **You develop an SAPUI5 app and implement a FacetFilter. What events are triggered when the user interacts with the FactFilter control?**

Reset & confirm

1. **Your customer extends an SAP Fiori app with View Replacement. Which activities do you perform?**

Verify that adding custom content in the Extension Points will NOT suffice

Create an SAPUI5 view that replaces the complete view

1. **Your customer asks you to demonstrate their app with localization changes. Which activity do you perform?**

Show the i18n.properties file translations

1. **Which element in the SAPUI5 architecture holds the data in JSON or XML?**

**Model**

1. **Which of the following pattern sequences are the QUnit tests based on?**

Arrange, Act, and Assert

1. **Which model type in SAPUI5 does NOT support one-way and two-way binding from model to view?**

Resource model

1. **Which features of the SAP Web IDE help you extend a standard SAP Fiori app?**

The Extensibility Pane to extend the control

A wizard to generate the files for each extension possibility

1. **Which properties do you need to configure when you use the Mock server?**

Module path & Service URL

1. **You need to configure a SAP Web IDE destination to extend an existing SAP Fiori app. Which configuration setting for the WebIDEUsage property is required?**

dev\_abap

1. **Which security features does the SAP Fiori provide to restrict the access to native device capabilities?**  
   Restricted device functionality using a client policy

Bridge for non-Fiori content

1. **Which application types can you develop and deploy on the SAP HANA Cloud platform?**  
   Java apps

HTML apps  
HANA XS apps

1. **Your customer requires an app to display flight information on a mobile device. Please see the screenshot for details. Which options can you use to display the same data in a readable format on a mobile device?**

column : <Column demandPopin=”true” minScreenWidth=”tablet”>

column: <Column demandPopin=”true” minScreenWidth=”tablet” popinDisplay=”Inline”>

1. **In the Git source code management system, what is executed initially to create the local Git repository in SAP Web IDE?**

**Clone**

1. **How can you extend an SAP Fiori app?**

Add customer view content in a predefined extension point

Enable merging of the standard and the custom controller at runtime

Modify the properties of the view control

1. **Which of the following sequences is the correct sequence for the Problem Space phase in a Design Thinking workshop?**

Scoping, 360’ Research, Synthesis

1. **What is the purpose of the Logon Plugin Data Vault of the SAP Fiori Client?**

Provide a reusable component for storing sensitive information on the device

1. **For which app type in SAP Fiori do you install the KPI framework?**

**Analytical**

1. **Where are the themes saved for the SAP Fiori launchpad?**

**Theme Repository**

1. **Which technologies are used to create Hybrid apps and SAPUI5 apps?**

**HTML5, JS and CSS**

1. **Your customer asks you for the advantages of the Model View Controller (MVC) design pattern. Which of the following answers do you provide?**

MVC artifacts can be transported and debugged separately

MVC is easy to implement for the project team members and supports quick coding

1. **You finalized your app development in the SAP Web IDE and need to deploy it. Which deployment targets are offered?**

SAP HANA Cloud Platform

SAPUI5 ABAP Repository  
Register to SAP Fiori launchpad

1. **Which of the following activities allow you to store your changes in a local Git repository?**

**Commit**

1. **You develop an SAPUI5 app that updates data for sales order and sales order items on the back-end system. What do you create to implement a deep insert?**

Create a nested structure for Sales Order and Sales Order Items

Create an object structure that defines the hierarchy

1. **Your customer wants to enable the SAP Fiori app to be accessed in multiple languages. Which of the following activities do you perform?**

Create a resource file containing key/value pairs

HTML -enable the special characters in the resource file

Create a resource file for each language

1. **What is the required content of the Component.js file of an extension project?**  
   The link to the parent app

The customizing section

1. Binding Type

Property, Element, Aggregation

1. **What are the advantages of the Sinon.JS framework?**

It provides support for Spies, stubs and mocks

It supplies higher level test doubles for timers and AJAX requests

1. **Which file is referred to as the App Descriptor and what is its function?**

The manifest.json file is referred to as App Descriptor Its function is to instantiate the model

1. **What result do you expect from the de-composition and re-composition phases?**

The prevention of irrelevant data being shown to the user

The break-down of a large transaction

A purpose-built app to support personas

1. **Which deployment options do you have for SAP Fiori UIs and OData Services regarding the software components?**

Two different deployment packages on a different from the business packages

One deployment package on the same system as the business system

1. **When SAP Web IDE deploys an SAPUI5 app to the ABAP server, which type of app is it deployed as?**

BSP

1. **You perform a QUnit test with the following syntactical options. Which call returns true when you execute it?**

Assert.strictEqual(0,-0,”true”);

1. **In which attribute do you specify the themes for an SAP Fiori app?.**

data-sap-ui-theme of <script> tag

1. **You need to maintain the mapping between an app URL and a destination in the SAPUI5 app. Which file do you maintain?**

neo-app.json

1. **In a typical SAP Fiori landscape, which server instance stores the SAP Fiori SAPUI5 application code for a transactional app?**

SAP Front-End Server (FES)

1. **In the screenshot, which element of the Arrange-Act-Assert pattern corresponds to the Act in a QUnit test?**

this.calculator.press(“1”)

1. **Your customer wants to extend an SAP Fiori app through a Controller Extension. When can a controller extension lead to a crash?**  
   If the original code required to run the application is overwritten.

If the extension code accesses parts of the original application that were removed.

1. **To generate SAPUI5 coding that is based on prototypes, which tool do you use?**

**WEB IDE**

1. **Which elements are part of the UX?**

User emotion

User effectiveness

1. **Which SAP Fiori application is launched by using the search functionality instead of using a tile?**

Factsheet

1. **What do you configure to import the BUILD prototype in the SAP Web IDE for development?**

The destination in the SAP HANA Cloud Platform

1. **What does coherence refer to in the SAP Fiori design principles?**

The same intuitive experience across the whole enterprise

A consistent interaction and visual design language

1. **You are extending an SAP Fiori app. Which of the extension properties are defined in the manifest.json?**

sap.ui.viewModifications  
sap.ui.controllerExtensions

1. **You need to consume the OData Service from the Gateway to create a new SAP Fiori app in the SAP Web IDE. What value do you configure in the WebIDEUsage property for the destination in the SAP HANA Cloud Platform**

odata\_abap

1. **What can your customer use a custom SAP Fiori client for?**

To apply custom branding

To wrap the application with SAP Mobile Secure

To add additional plugins to the application

1. **What is a prerequisite for configuring the SAP Fiori HAT Connector?**

Create an SAP HANA Cloud Platform (HCP) Web IDE account

1. **To perform a QUnit test, what do you include in a test page under your webapp folder?**

SAPUI5 bootstrap code

1. **Which properties are part of the design phase when you develop SAP Fiori apps?**  
   Ideation, Prototyping and Validation
2. **Which of the following components are part of the SAP Web IDE, hybrid app toolkit add-on?**

SAP Web IDE Plugin, Hybrid Companion App, Hybrid App Toolkit Connector

1. **For which operations will the SAP Gateway server compute and return a new ETag in a response header?**

POST

PUT

1. **Your customer requires that the SAPUI5 design of all apps reference a specific and consistent look and feel. What do you create?**

Create a model using the Web IDE implementing the specific look and feel and reference this model in the bootstrap of all apps  
Create a theme using the Theme Designer implementing the specific look and feel and reference this theme in the bootstrap of all apps  
Create a css file using the Web IDE implementing the specific look and feel and reference this css in the bootstrap of all apps

1. **What happens when you use the SAP Web IDE console during a deployment?**

You can view error messages when a deployment is NOT successful

You can debug the error messages in the console

1. **When do you select the custom plugins to build and package a hybrid app using the SAP HAT?**

During the Configure Path for Custom Plugins step of installing the SAP HAT

1. **You develop an SAPUI5 app that updates data on the SAP back-end system. What activities are required when you implement an XSRF token?**

Execute the Refresh method for an expired token  
Retrieve the token and send it with each service request

1. **Which methods can you use to bind data to the controls in SAPUI5?**

[**Property Binding**](https://sapui5.hana.ondemand.com/1.28.33/docs/guide/91f0652b6f4d1014b6dd926db0e91070.html)

Property binding allows properties of the control to get automatically initialized and updated from model data.

Eg: var oTextField = new sap.ui.commons.TextField({

value: "{/company/name}"

});

[**Aggregation Binding**](https://sapui5.hana.ondemand.com/1.28.33/docs/guide/91f057786f4d1014b6dd926db0e91070.html)

Aggregation binding is used to automatically create child controls according to model data.

The item aggregation of the list is bound to the root path Invoices of the JSON data.

In the items aggregation, we define the template for the list that will be automatically repeated for each invoice of our test data.

var oItemTemplate = new sap.ui.core.ListItem({text:"{name}"});

var oComboBox = new sap.ui.commons.ComboBox({

items: {

path: "/company/contacts",

template: oItemTemplate

}

});

[**Element Binding**](https://sapui5.hana.ondemand.com/1.28.33/docs/guide/91f05e8b6f4d1014b6dd926db0e91070.html)

Element binding allows to bind elements to a specific object in the model data, which will create a binding context and allow relative binding within the control and all of its children. This is especially helpful in master/detail scenarios.

oMatrixLayout.bindElement("/company");

oMatrixLayout.createRow(

new sap.ui.commons.Label({text: "Name:"}),

new sap.ui.commons.TextField({value: "{name}"})

);

**Data Types: Pre defines formatting options eg: currency will automatically set xx.xx this format**

**number="{**

**parts: [{path: 'invoice>ExtendedPrice'}, {path: 'view>/currency'}],**

**type: 'sap.ui.model.type.Currency',**

**formatOptions: {**

**showMeasure: false**

**}**

**}"**

**numberUnit="{view>/currency}"**

**Expression Binding – Sometimes the predefined types of SAPUI5 are not flexible enough and you want to do a simple calculation or formatting in the view - that is where expressions are really helpful**

**numberState="{= ${invoice>ExtendedPrice} > 50 ? 'Error' : 'Success' }"**

1. **Your customer requests a special create method to enable the creation of the full OData hierarchy. Which OData service method can you use?**

CREATE\_DEEP\_ENTITY

1. **Which SAP Fiori app types require the SAP HANA database?**

Analytical

Factsheet

1. **When do you select the custom plugins to build and package a hybrid app using the SAP HAT?**  
   During the Configure Path for Custom Plugins step of installing the SAP HAT
2. **Which of the following pattern sequences are the QUnit tests based on?**

Arrange, Act, and Assert

1. **Which features of the SAP Web IDE help you extend a standard SAP Fiori app?**

A wizard to generate the files for each extension possibility

The Extensibility Pane to extend the control

1. **In the screenshot, which element of the Arrange-Act-Assert pattern corresponds to the Act in a QUnit test?**

this.calculator.press(“1”)

1. Design Thinking

Design Thinking is human approach that draws from designer toolkit to integrate people needs, possibility of technologies and requirement for business success

It is having 3 phases

1. Discover the problem

Steps:

1. Common Understanding (Scoping) – Addressing the problem without Anticipating the real solution
2. 360 (degree) Research - 360° Research is about research, discover, explore and capture data. Yet there is no Solution (Unstructured and Raw Data)
3. Synthesis - Unstructured to Structured Data which is collected in 360 Research

Note: Persona is fictional character to represent a user

1. Design the prototype / solution
2. Ideation - Ideation is about brainstorming ideas. At this point the team starts to imagine possibilities, with the goal of generating as many ideas as possible. Do not check for feasibility and viability at the beginning of ideation; this is done later during prioritization.
3. Prototyping - Prototyping is the first step to actually feel an idea. Initial Prototype is Mock or Lo-fi Prototype
4. Validation is about gathering feedback on concepts and prototypes.
5. Deliver the product/ service
   1. Development
   2. Testing
   3. Implementation - Implementation is about realizing a validated prototype

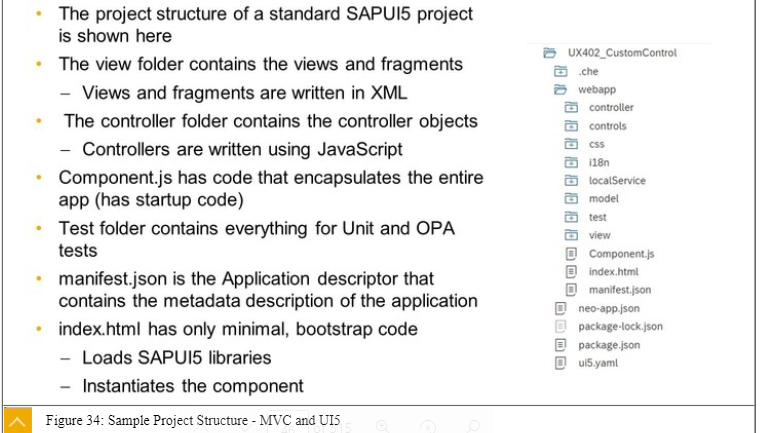
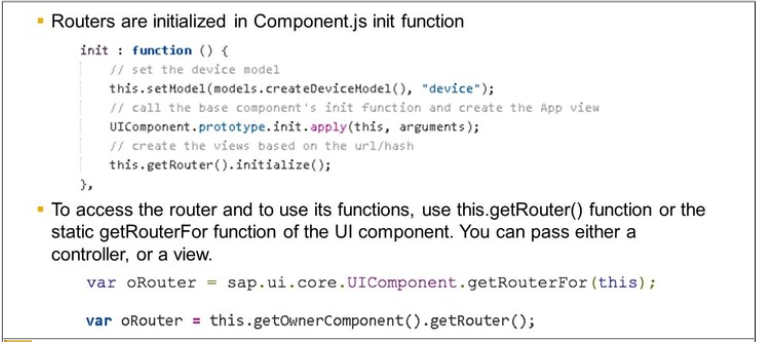
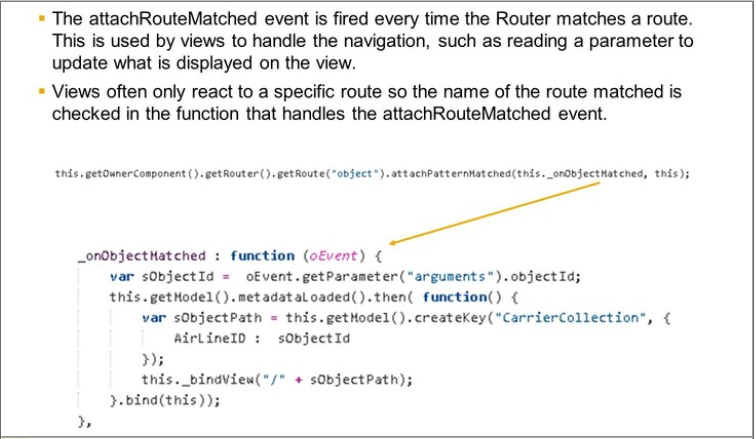
Design Thinking Components:

1. People,
2. Environment and Material,
3. Approach
4. **The original design template of SAP Fiori, which was called Blue Crystal, has been adapted to cover native apps and renamed to Belize**
5. **SAPUI5 Control Library**
6. **Sap.m – Mobile and Works everywhere**
7. **Sap.ui.commons – basically for desktop**
8. **Sap.ui.table – full feature of table control**
9. **Sap.viz – Charts and graph**
10. **Sap.ui.ux3 – ux 3.0 guideline control**
11. **Sap.ui.layout – layout control**
12. **ODATA is based on REST with latest version 4 used by IBM, SAP, etc**
13. **SAP Gateway is used by business suite to publish ODATA on ABAP application server**
14. **The service builder is a useful transaction that allows you to develop OData services for the SAP Business Suite. Service builder allows you to leverage an existing remote function call (RFC), Business Application Programming Interface (BAPI), or search help, and perform a service implementation by mapping**
15. **OData is an open standard originally developed by Microsoft but now managed by the Oasis Organization. It is based on the Atom Publishing and Atom Syndication standards, which in turn are based on XML and HTTP(S)**
16. **SAP Cloud Platform is a platform-as-a-service (PaaS) for extending, integrating and building apps to meet new challenges, attract new customers, and drive new business.**
17. **Segw – to create new Service**

**Se80 -** object navigator where u can see all objects like reports , repository objects , packages , function gro up , BSP applications .

**/Iwfnd/maint\_Service – activates services or adding system aliases**

**/iwfnd/gw\_client – test the odata service like read, etc**

1. **For Destination we need to add additional property WEBIDEUsage**
   1. odata\_abap (for consuming OData)
   2. dev\_abap (for extending existing Fiori apps)
   3. ui5\_execute\_abap (for executing SAPUI5 apps)
2. 
3. To use Expression Binding, you need to enable Extended Binding Syntax via the configuration setting xx-bindingSyntax set to complex.
4. With the introduction of the function sap.ui.define in the version 1.28, SAPUI5 has introducedthe support for Asynchronous Module Definition (AMD).
5. A module is a JavaScript file that can be loaded and executed in a browser.
6. Asynchronous Module Definition (AMD) is a JavaScript API which specifies a way to define amodule and its dependencies in such a way that they can be loaded asynchronously withoutworrying about the loading order.
7. Route with mandatory parameter:You can define mandatory parameters for the pattern by placing the parameter in curlybrackets ({parameter ID}).For example, if you define the pattern product/{id}, the hashes product/5 and product/3 (where 3 and 5 are product IDs) match the pattern. The matched event handlergets5 or 3 passed on with the key id in its arguments. But hash product/ does not match thepattern because the mandatory parameter is missing.Route with optional parameter:You can define optional parameters for the pattern by placing the parameter between colons(:parameter ID:).For example, if you define a pattern product/{id}/detail/:detailId:, the detailId parameter isoptional, whereas id is mandatory. Both hashes product/5/detail andproduct/3/detail/2 match the pattern.
8. 
9. 
10. displayBlock="true" prevents vertical scrollbars appearing with Views that are set to 100%height. This is used in app.view
11. JavaScript Promises The following are further information about promises ●A promise is an object that may produce a single value sometime in the future. A Promise can be in one of 3 states:

-Pending- the Promise’s outcome hasn’t yet been determined, because the asynchronous operation that will produce its result hasn’t completed yet.-Fulfilled- the asynchronous operation has completed, and the Promise has a value.-

Rejected- the asynchronous operation failed, and the Promise will never be fulfilled. In the rejected state, a Promise has a reason that indicates why the operation failed.

1. Using the bypassed property, you tell the Router to display the master target and the notFound target in case no route was matched to the current hash. The notFound targetconfigures a NotFound view.
2. Aggregation:●A relationship between two controls●A parent/child relationship●Example: table rows (parent) and cells (children)Association:●A relationship between two controls●

Association -Not a parent/child relationship●Represent a loose coupling●Example: a label is associated with a text field

1. Method Naming

on...: are used for event handlers

-init: are used to initialize the control, after its instantiation■This is a private method only called by the SAPUI5 core-

renderer: The renderer is responsible for creating the HTML structure for the control

1. What is Control

A UI component (usually a visible one) that can react to user activities and expose properties, methods and events to the application (and other controls) via JS API

Technically, a UI5 control at design time consists of:

The **Control APIdefinition** which defines the properties, events, methods and sometimesthe associations and aggregations

The **Control Renderer** which is responsible for creating the HTML string defining thestructure of the control

The **Control Behavior** which is the JavaScript code taking care of the control interactivity,reacting on user events, firing control events, handling method calls and propertychanges

The **Control Style** defining the visuals of the control (usually a control has different visualdesigns, bundled as "Themes")

Same Rules Apply to Custom Controls

1. Define Catalog, Groups and Roles, Tiles

A catalog is a set of Tiles / Applications you want to make available for one role. Using catalog and role user can define what app to be defined in entry points

A group is a subset of Applications from one or more catalogs

1. Which SAPUI5 versions are available on an Application Server ABAP is provided via https://<host>:<port>/sap/public/bc/ui5\_ui5/index.html.The version a running app is using is visible in the technical information dialog opened viaCTRL+ALT+SHIFT+P.
2. The Cloud Connector is a standalone software that is available free of charge to connect the services in the SAP CP with the on-premise systems in the customer network. Once installed in the customer network, it establishes a secure SSL Virtual Private Network (VPN)connection to the SAP CP. It is not needed for SAP Web IDE Personal Edition or Visual StudioCode.
3. A reverse proxy offers a single point of communication holding routing rules to forward requests to the correct target systems.
4. **A reverse proxy is a type of proxy server.  Unlike a traditional proxy server, which is used to protect clients, a reverse proxy is used to protect servers. A reverse proxy is a server that accepts a request from a client, forwards the request to another one of many other servers, and returns the results from the server that actually processed the request to the client as if the proxy server had processed the request itself. The client only communicates directly with the reverse proxy server and it does not know that some other server actually processed its request.**
5. $fomat=JSON
6. onInit and onAfterRendering in the extension are called after onInit and onAfterRenderingof the standard/original controller●onExit and onBeforeRendering in the extension are called before onExit andonBeforeRendering of the standard/original controller
7. XS Advanced is a renamed and enhanced version of Cloud Foundry.

Cloud Foundry is an open-source, multi-cloud application platform governed by the Cloud Foundry Foundation.Cloud Foundry is currently the application service of choice for the SAP Cloud Platform, whileXS Advanced is provided with the on premises version of SAP HANA.In most cases, you will find that an application running on XS Advanced will also run in CloudFoundry on the SAP Cloud Platform.

1. The full-stack application is made of the following three layers:●The UI layer: This layer uses the SAPUI5 JavaScript library and runs in the web browser.●The application server layer: The application logic that runs in this layer can be written indifferent languages, such as, Node.js, Java, or Python.●The database layer: The persistency layer is defined via Core Data Services (CDS). Thedata intensive logic is developed in SQLScript and runs within the SAP HANA Database.

|  |  |  |
| --- | --- | --- |
|  | ***Front-End server (T-code)*** | ***Description*** |
| 1 | SICF  (Path: /sap/bc/ui5\_ui5/sap/) | UI5 Application Path |
| 2 | SICF  (Path: /sap/opu/odata/sap/) | Odata Service Path |
| 3 | SEGW | Create Odata Service |
| 4 | /n/iwfnd/maint\_service | Odata Service Registration |
| 5 | /n/iwfnd/gw\_client | Test Odata Service |
| 6 | /n/iwfnd/error\_log | Error Log |
| 7 | SE38 (Report: /UI5/UI5\_REPOSITORY\_LOAD) | To upload/Download/Delete UI5 Application |
| 8 | /n/UI2/SEMOBJ | Semantic objects by SAP |
| 9 | LPD\_CUST | Launchpad Role |
| 10 | PFCG | PFCG Role |

1. OData: a standardized method (Open Data Protocol) for interacting with data on a server, ensuring interoperability between sources

OData annotations are metadata that define attributes and relationships

Backend Layer Can be – SAP s/4 HANA, SAP BW, SAP Cloud Platform, Other Backend

1. Structured Query Language (SQL) is a standardized language for communicating with arelational database. SAP HANA database is a relational database and fully supports SQL.SQL is used to retrieve, store or manipulate data in the SAP HANA database.
2. SAP HANA XS Classic – Web application server with server side JS engine embedded in database

SAP HANA XS Advance (and Cloud Foundry) – platform for microservices based polyglot web application on premises (and in Cloud)

SAP HANA Deployment Infrastructure (HDI) – service that helps to deploy database development artifacts to containers

SAP HANA HDI Container – consist of design time and respective run time container

1. A multi-target application includes multiple so-called “modules” which are the equivalent of Cloud Foundry applications.
2. HDI Container contains

●Design-time container – An isolated environment for design-time files

●Runtime container – Stores the deployed objects built according to the specificationstored in the corresponding design-time artifacts

1. When using containers, you do not explicitly reference a schema name when referring todatabase object in your SQLScript. As long as the development objects are in the samecontainer, they will find each other. Only when you need to reference database objects inother external containers or catalog schemas will you need to take action. In that case, youfirst need to build synonyms in your container that point to the external database object youneed access to.
2. SAP Logon is s4 hana screen
3. STC01 – Configure tasks
4. DEVELOPMENT S/4HANA 1909 FPS01 Current Version
5. SAP HANA Cloud allows you to consume data from SAP HANA database from Cloud applications running on SAP Cloud Platform, and also from running from somewhere else using SAP HANA Clients. Every instance of SAP HANA hits its own SAP HANA Database
6. How to change version of Application

* Go to manifest – change version and minimum version field to th desired version
* Go to Project Setting – General – SAPUI5 – version changed to desired one

1. Distribution folder i.e. dist will be created once deployed
2. App is inside the folder – sap/bc/ui5\_ui5/sap/app\_Name

How to get the file path – SICF- service name = App\_Name – Execute – get the path of app

1. In the example above we get a reference to the SAPUI5 core by calling sap.ui.getCore() and register an anonymous callback function by calling attachInit(…) on the core. In SAPUI5 callback functions are often referred to as handlers, listener functions, or simply listeners. The core is a Singleton and can be accessed from anywhere in the code.Our anonymous callback function is executed when the bootstrap of SAPUI5 is finished and displays a native JavaScript alert.There are two versions of the SAPUI5 core: one with jQuery and one without jQuery. In our case we load the core bundled with jQuery so that we can make use of jQuery within the app.
2. Each control extends sap.ui.core.Element which in turn extends sap.ui.base.ManagedObject. A control can typically be rendered. Instances of sap.ui.core.Element cannot be rendered. Please refer to the API documentation to learn more about the inheritance hierarchy of controls.
3. We tell SAPUI5 core that resources in the sap.ui.demo.wt namespace are located in the same folder as index.html. This is, for example, necessary for apps that run in the SAP Fiori launchpad.
4. From this step onwards, it is necessary to run the app on a Web server. We structure the app with multiple files that are loaded from the local file system. Without a Web server, this is prevented by the browser due to security reasons. If the error message "sap is not defined" appears in the developer tools of the browser, you need to check the resource path in the bootstrap.

We extend the array of required modules with the fully qualified path to sap.m.MessageToast. Once both modules, Controller and MessageToast, are loaded, the callback function is called and we can make use of both objects by accessing the parameters passed on to the function.

This Asynchronous Module Definition (AMD) syntax allows to clearly separate the module loading from the code execution and greatly improves the performance of the application. The browser can decide when and how the resources are loaded prior to code exection.

**Conventions**

* Use sap.ui.define for controllers and all other JavaScript modules to define a global namespace. With the namespace, the object can be addressed throughout the application.
* Use sap.ui.require for asynchronously loading dependencies but without declaring a namespace, for example code that is directly executed and is not referenced in other places.
* Use the name of the artifact to load for naming the function parameters (without namespace).

1. We create an initial Component.js file in the webapp folder that will hold our application setup. The init function of the component is automatically invoked by SAPUI5 when the component is instantiated. Our component inherits from the base class sap.ui.core.UIComponent and it is obligatory to make the super call to the init function of the base class in the overridden init method.
2. The helper method sap.ui.core.ComponentContainer instantiates the component by searching for a Component.js file in the namespace that is passed in as an argument. The component is automatically loading the root view that we have defined above and displays it. If you now call the index.html file, the app should still look the same, but is now packaged into a UI component.
3. Use of Manifest

All application-specific configuration settings will now further be put in a separate descriptor file called manifest.json. This clearly separates the application coding from the configuration settings and makes our app even more flexible. For example, all SAP Fiori applications are realized as components and come with a descriptor file in order to be hosted in the SAP Fiori launchpad.The launchpad acts as an application container and instantiates the app without having a local HTML file for the bootstrap. Instead, the descriptor file will be parsed and the component is loaded into the current HTML page. This allows several apps to be displayed in the same context. Each app can define local settings, such as language properties, supported devices, and more. And we can also use it to load additional resources and instantiate models like our i18n resource bundle.

1. In order to display the app control correctly on mobile phones, we add the displayBlock attribute with the value true to the view.
2. Asynchronous Loading – Sap.ui.define, does not support other file type
3. Synchoronous – sap.ui.require
4. Steps to create SAPUI5 in CF

Create empty project in SAP Business Studio

Make Sure project locally

Add northwind or ant odata source

Modify Approuter

Add some basic UI

Make sure app runs properly with mock destination

Deploy to CF

Add destination to CF

Make sure app runs in CF with real Destinations

1. Proxy server that forwards the request of your app to the UI5 server and the other to your backend server A proxy is simply a local script that connects to the remote server and returns the response to our app. In a productive scenario this would of course be a direct connectionThe web page or web application becomes a vehicle to deliver the malicious script to the user’s browser is called cross site scripting.

Avoiding the domain it used the application to run the application locally

1. HSTS is all about embedding the security header in the URL and specifying the expire time for which the browser(client) will not need confirmation from the server repeatedly for converting HTTP to HTTPS avoiding the man-in-the-middle attack (MITM).

Browser normally follow same origin policy and blocks so called CORS request by default

CORS means cross origin resource sharing and enable flexible sharing to fetch data from other host than actual web server the app is running on. Proxy allows certain request and add certain CORS header so that browser accept odata request

1. In SAP NetWeaver Gateway, entities are defined and connected in data models. These data models can be exposed as OData services during runtime. Runtime processing is handled by data provider classes (DPC) that are assigned to the object models.

Entity

It is something which has real existence. Like tuple1 contains information about Ram(id, name and Age) which has existence in real world . So the tuple1 is an entity. So we may say **each tuple is an entity**.

Entity Type

It is collection of entity having common attribute. As in Student table each row is an entity and have common attributes. So **STUDENT** is an entity type which contains entities having attributes id, name and Age.Also each entity type in a database is described by a name and a list of attribute.So we may say **a table is an entity set while table name specifies the entity type**

Entity SET

It is a set of entities of same entity type. so a set of one or more entities of Student Entity type is an Entity Set.

ATTRIBUTE

It is a property of an entity. For example, in table STUDENT id,name and Age are properties of an entity of entity type student. Hence these are attributes.

**SAP S/4HANA can be deployed on two delivery platforms:**

* 1. On Premises
  2. Cloud
  3. Hybrid (Both)

When we change the Model or implement the Service in GW Service builder, every time run time object gets generated, hence DPC and MPC existing code gets replaced with new sets of code. If we implement any code in DPC/MPC while generating runtime object the code gets deleted.

**MPC** - This is used to define model. you can use the method Define to create entity, properties etc using code based implementation. ... **DPC** - used to code your CRUDQ methods as well as function import methods.

ODATA is restful API – style of software architecture for distributed system ie world wide web

JSON is lightweight as compared to XML since XML have many line added in XML tags

SAP\_GWFND – access to backend system

SWGW – accesss to Gateway service builder

Web services for Rest is synchronous

Se11 – open the table create entity set, entity, mpc, dpc

Se24 - to view all menthod of mpc and extensions and dpc extension

System alias is required to map backend system ie it points to trusted RFC

Sicf – service – can be seen path of service ie created services node will seen here

Spro – to check system alias from backend system

Since system alias connect hub to backend.it will help to know which system is it coming from

/Iwfnd/maint\_service – maintain and activate service

/iwfnd/gw\_client

Se01 – create user

SAP Cloud Application Programming Model is a framework of tools, languages, and libraries to efficiently build enterprise-grade services and applications. This framework is both open and opinionated,

Its is basically SAAS while SAP HANA on premises is internal platform

SAPUI5 applications are build with the MVC pattern and therefore consist of multiple files, like views or controllers, With fragments, it’s possible to split up parts of views even further into smaller logical units.

Although having a lot of difference files based on their functionality is good during the development, it’s bad when running on a production server. Every necessary file fires a separate network request. You have to wait until all network requests are completed before the app is usable. you may not notice this delay in your corporate network[which is not in my case   :P], but you’d notice in on a mobile phone with a moderate connection speed.

To solve this issue, you can package different files into a single file, a process called ***bundling.***through bundling, your app only has to load a single file instead of all the separate files, which speeds up the loading time of your application because your app has to wait only for a single file to load.

In SAPUI5, this single file is called ***Component-preload.js***. this file contains all the code of your application in a minified version. **Minification** means removing all unnecessary white-space and comments, Which also decreases the file size. In addition, the javascript source code is also uglified. ***Ugification***means that your source code will not be as readable as before.

|  |  |
| --- | --- |
| ABAP Platform | <https://developers.sap.com/topics/abap-platform.html#tutorials/cjm9k6acpbukk0932znlglyzl> |
| oData/gateway | <https://sapyard.com/tutorials-on-odata-sap-netweaver-gateway/> |

Symmentic WSS Agent

1. The neo-app.json file contains all project settings for SAP Web IDE and is created in the root folder of your project. It is a JSON format file consisting of multiple configuration keys. The most important setting for you to configure is the path where the SAPUI5 runtime is located when starting the app.

You do this using the “routes” key and defining an array of resource objects. For running an SAPUI5 tutorial, you only need two entries –

* 1. configures SAPUI5 to be available with the path /resources
  2. configures the test resources needed for the SAP Fiori launchpad integration with the path /test-resources.

Create two configuration objects that contain a path, a target, and a description attribute with more configuration settings. The path and the entryPath values will point to the location on the server where the SAPUI5 resources will be stored.

SAP Web IDE reads these settings automatically when running the app. You can see the whole configuration file in the code block below. Optionally, you can add the key welcomeFile to configure the entry point to your app. In web applications, this is typically the index.html file.

{

"welcomeFile": "index.html",

"routes": [

{

"path": "/resources",

"target": {

"type": "service",

"name": "sapui5",

“version”: “snapshot”,

"entryPath": "/resources"

},

"description": "SAPUI5 Resources"

},

{

"path": "/test-resources",

"target": {

"type": "service",

"name": "sapui5",

"entryPath": "/test-resources"

},

"description": "SAPUI5 Test Resources"

}

]

}

* **SAP BAPI** (Business Application Programming Interface) is a standard interface to the business object models in **SAP** products. **BAPIs** are the primary method through which customer code and third-party applications interact with **SAP** products. It Supports the Remote Function Call (RFC) protocol

The SAP business objects stored in the Business Object Repository (BOR) encapsulate their data and processes. External access to the data and processes is only possible by means of specific methods - **BAPIs** (Business Application Program Interfaces).

1. Function modules are procedures that are defined in special ABAP programs only, so-called function groups, but can be called from all ABAP programs. Function groups act as containers for function modules that logically belong together. You create function groups and function modules in the ABAP Workbench using the [Function Builder](https://help.sap.com/saphelp_nw73/helpdata/en/d1/801e9a454211d189710000e8322d00/frameset.htm) .

Function modules allow you to encapsulate and reuse global functions in the SAP System. They are managed in a central function library

RFC (Remote Function Call) are remote enabled function module for communication between SAP systems and external systems

BAPI is implemented as RFC enabled function module and are created in function builder through ABAP workbench

A BAPI function is a function module that can be called remotely using the RFC technology.

1. To upload the application enter /UI5/UI5\_REPOSITORY\_LOAD
2. <https://help.sap.com/doc/saphelp_ssb/1.0/en-US/34/4e3d5269a2b610e10000000a44176d/content.htm?no_cache=true>
3. Customiing layer and configuration layer in launchpad
4. Difference between configuration and customizing layer in launchpad
5. Difference between association, navigation and expand in odata

***Associations****define the relationship between two or more Entity Types (for example, Employee WorksFor Department). Instances of associations are grouped in****Association Sets****.*

***Navigation Properties****are special properties on Entity Types which are bound to a specific association and can be used to refer to associations of an entity.*

*Finally, all instance containers (Entity Sets and Association Sets) are grouped in an****Entity Container****.*

Association – gives associated (dependent) entity / entity set using navigation property

In Expand – gives associated (dependent) entity / entity set + Principal entity / entity set using navigation property

1. Technically, the files of the SAPUI5 application are stored as a BSP application on the server.

All put together,  if you want to use a custom namespace, then you need to create a child node for ui5\_ui5 named as your namespace and add a child node to your namespace node with just the name of your BSP application without namespace prefix. The BSP application name itself nevertheless must contain the namespace prefix:

* SICF node for the SAPUI5 application in custom namespace: */sap/bc/ui5\_ui5/iprocon/testui5*
* BSP application name in custom namespace: /IPROCON/TESTUI5

If you do not want to use a custom namespace, you can simply pick any BSP application name from the customer namespace and create a sicf node with your application name as child of */sap/bc/ui5\_ui5/sap*.

* SICF node for the SAPUI5 application: */sap/bc/ui5\_ui5/sap/ztestui5*
* BSP application name: ZTESTUI5

The reason for this comes from the logic in the http handler classes for SAPUI5, that are registered with the node /sap/bc/ui5\_ui5.

1. Changes to cross-client Customizing objects and to Repository objects are recorded in Workbench requests

In **workbench (Configuration)** request .... it can be programs (repository objects) and cross client data (ie not specific to one client).

Changes to client-specific Customizing objects are recorded in **Customizing** requests.

Customer customized data designed for a specific client is transported to another client using customizing request. Most of the cases these requests are something like changing values in tables (that too client specific)

1. factory function :
2. **Device Responsive - demandPopin="true", minScreenWidth = ‘Tablet’**
3. Function Import from SAPUI5 :

oDataModel.callFunction("/TestFunctionImport", { // function import name  
 method: "POST", // http method  
 urlParameters: {"parameter1" : "value1" }, // function import parameters   
 sucess: function(oData, response) { }, // callback function for success  
 error: function(oError){ } // callback function for error

});

What is WEB IDE

SAP Web IDE is a web-based development environment that is optimized for developing SAPUI5 complex apps using the latest innovations, developing and extending SAP Fiori apps, developing mobile hybrid apps, and extending SAP Web IDE with plug-ins and templates.

App Development from WEB IDE

* Start a trail account
* After Logging in go to Service and click on WEB IDE Full Stack Tile
* Start Neo-app.JSON - The neo-app.json file contains all project settings for SAP Web IDE and is created in the root folder of your project. It is a JSON format file consisting of multiple configuration keys. The most important setting for you to configure is the path where the SAPUI5 runtime is located when starting the app.
* To use SAPUI5 features in your HTML page, you have to load and initialize the SAPUI5 library.

SAP Cloud Application Programming Model

<https://community.sap.com/t5/technology-blogs-by-sap/introducing-the-cloud-application-programming-model-cap/ba-p/13354172>

* *To not spend most of the time worrying about things like implementing metadata providers, serving $metadata requests as well as each individual CRUD requests, i18n and localization, filling in auditing data, code lists and value helps, extensibility and customization, multi-tenancy, resilience, and so on, instead of focusing on her actual tasks.*
* Following the golden path of the programming model helps you implement data models, services and UIs in order to develop stand-alone business applications or extend other cloud solutions, like SAP S/4 HANA or SAP SuccessFactors.

SAP Fiori elements is a framework that comprises the most commonly used floorplan templates and is designed to:

* Speed up development by reducing the amount of front-end code needed to build SAP Fiori apps
* Ensure stable, optimized UI code out of the box
* Deliver high-quality SAPUI5 applications to end users
* Drive UX consistency and compliance with the latest SAP Fiori design guidelines

App developers can use SAP Fiori elements to create SAP Fiori applications based on OData services and annotations that don't need JavaScript UI coding.

The current templates for SAP Fiori elements are as follows:

* List Report
* Object Page
* Analytic List Page
* Overview Page
* Worklist Page

Basic Steps for SAPfiori Elements

* Step1

A screenshot of a computer

Description automatically generated

* Step 2

A screenshot of a computer

Description automatically generated

* Step 3

A screenshot of a computer

Description automatically generated

* Step 4
* A screenshot of a computer

  Description automatically generated
* Step 5

A screenshot of a computer

Description automatically generated

* Step 6
* A screenshot of a computer program

  Description automatically generated
* Step – 7
* A screenshot of a computer

  Description automatically generated
* Step 9
* A screenshot of a computer

  Description automatically generated
* - Step 10
* A screenshot of a computer

  Description automatically generated

Started BAS Teminal command –

* Cd .. back to previous folder

How to get the data through AJAX call

* Create Destination - Enter Name, Type, Description, URL, Authentication
* Make Entry level for Destination in Neo-app.json
* {
* "path": "/zeenews",
* "target": {
* "type": "destination",
* "name": "zeenews"
* },
* "description": "News Feedservice"
* }
* AJAX call
* var aData = jQuery.ajax({
* type: "GET",
* contentType: "application/xml",
* url: "/zeenews/rss/world-news.xml",
* dataType: "xml",
* async: false,
* success: function(data, textStatus, jqXHR) {
* ;
* oModel.setData(data);
* alert("success to post");
* }

**Building and deploying the app**

**Building the app**

Do a right click on the “mta.yaml” file and select the option “Build MTA Project“.

**Deploying the app**

After successfully building the mta project a new folder “mta\_archives” with a “.mtar” file is created.

A screenshot of a computer

Description automatically generated

Do a right click on the “.mtar” file and select the option “Deploy MTA Archive” to start the deployment.  
  
**It can be necessary to provide your Cloud Foundry endpoint, your SAP credentials and the organization and the space where you want to deploy your app!**

**Steps to Create MTA Application , Deploy and view the app**

<https://community.sap.com/t5/technology-blogs-by-sap/calling-a-sap-business-technology-platform-rest-api-in-a-sapui5-application/ba-p/13518385>

**SAP Business Technology Platform is an innovation platform optimized for SAP applications in the cloud.**

1. From the “Welcome” page navigate to “Start from template”

1. Create a “Basic Multitarget Application”

* + Select the tile “Basic Multitarget Application”

* + Find a project name

* + Select button “Finish”

1. After the app was created successfully find the “mta.yaml” file

1. Do a right click on the “mta.yaml” file and select “Create MTA Module from Template”

1. Create an “Approuter Configuration”

* + Select the tile “Approuter Configuration”

* + Choose the HTML5 Application Runtime “Standalone Approuter”

* + **CAUTION: Please pay attention to select the correct approuter. The described steps will not work with the “Managed Approuter”!**

* + Select “Yes” to decide that you want to add an authentication and that you plan to add an UI

* + Click “Next”

1. Wait until the approuter is created successfully

1. From the “Welcome” page navigate to “Start from template”

1. Create a “SAP Fiori application”

* + Select the tile “SAP Fiori application”

* + Select Application Type “SAPUI5 freestyle”

* + Choose Floorplan “SAPUI5 Application” and click “Next”

* + Select Data Source “None”

* + Find a view name

* + Find a project name, an application title and an application namespace

* + Choose as folder path the folder of your created "Basic Multitarget Application"

* + Select “Yes” as deployment configuration

* + Keep the default selection of the radio buttons for “FLP configuration” (No) and “Advanced options” (No) and click “Next”

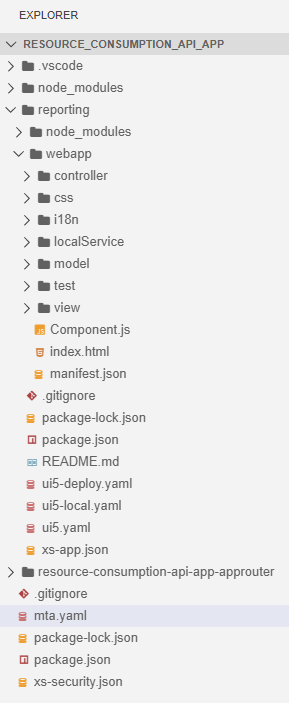
* + Choose “Cloud Foundry” as target

* + Select Destination “None” and click “Finish”

It can take some time until your project was created successfully. A message toast indicates that the creation of all files is finished.



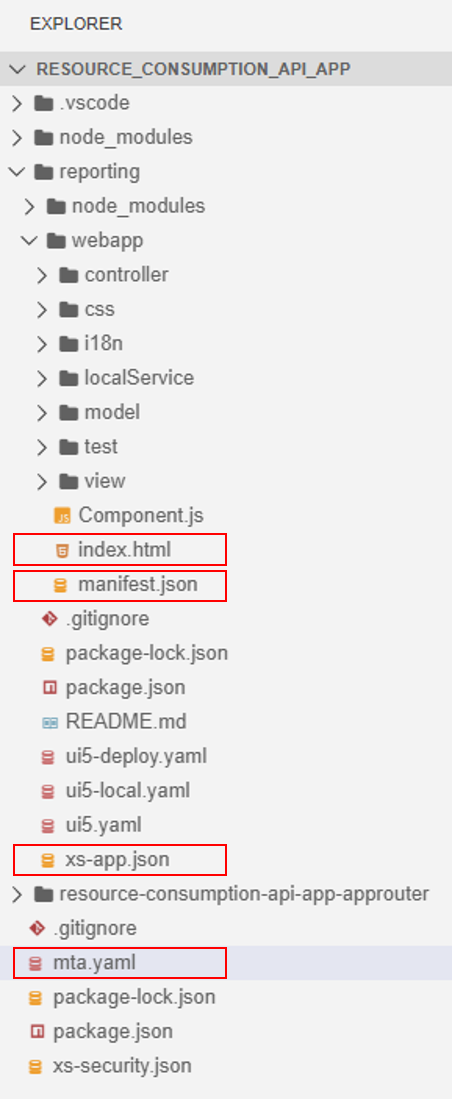
Now the app should be set up completely and the folder structure should look similar to the structure seen in this screenshot:



In the next section I will show you which code snippets you have to add to the corresponding files to receive the data from the REST API.  
  
*Tipp: To beautify your code and therefore getting a better overview, please have a look into following blog:*[*SAP Business Application Studio – How to beautify xml and javascript code*](https://blogs.sap.com/2021/04/30/sap-business-application-studio-how-to-beautify-xml-and-javascript-code/)

**Calling the REST API in the SAPUI5 application**

In the following screenshot I have marked the files which will be adjusted in the next steps.



**Change file “index.html”**

Change the source (src) to following:

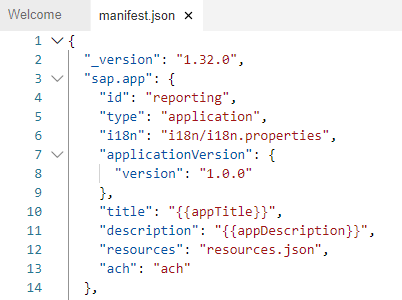


src=https://sapui5.hana.ondemand.com/resources/sap-ui-core.js​

**Change file “manifest.json”**

Add the following code snippet into the “sap.app” section:



First you need to add a comma after "ach": "ach" and secondly you can paste following code snippet below that:

"dataSources": {

"mainService": {

"uri": "reporting/reports/v1/monthlySubaccountsCost?fromDate=202101&toDate=202109",

"type": "JSON"

}

}

The result should look like this:



Add the following code snippet into the “models” section:

A screenshot of a computer code

Description automatically generated

First you need to add a comma after the second closing curly bracket and secondly you can paste the following code snippet below that:

"reporting": {

"type": "sap.ui.model.json.JSONModel",

"dataSource": "mainService",

"settings": {

"defaultBindingMode": "TwoWay",

"defaultCountMode": "Inline",

"refreshAfterChange": false

}

}

The result should look like this:

****

**Change file “xs-app.json”**

Delete the automatically generated first two routes:



Add the following code snippet as a new route to the file:

{

"source": "^/reporting/(.\*)$",

"target": "$1",

"service": "com.sap.cloud.udm",

"endpoint":"reporting",

"csrfProtection": false,

"authenticationType": "xsuaa"

},

The result should look like this:



**Change file “mta.yaml”**

Add the following code snippet into the "modules --> requires" section:

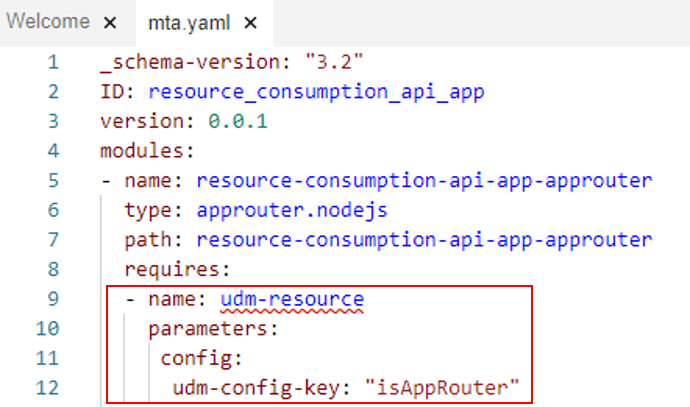
- name: udm-resource

parameters:

config:

udm-config-key: "isAppRouter"

The result should look like this:



**CAUTION: The line inserts must be according to the code snippet to prevent errors. Be aware that the “udm-resource” is red underlined until the next code snippet is added to the “resources” section (see next step).**  
  
Add the following code snippet into the "resources" section:

- name: udm-resource

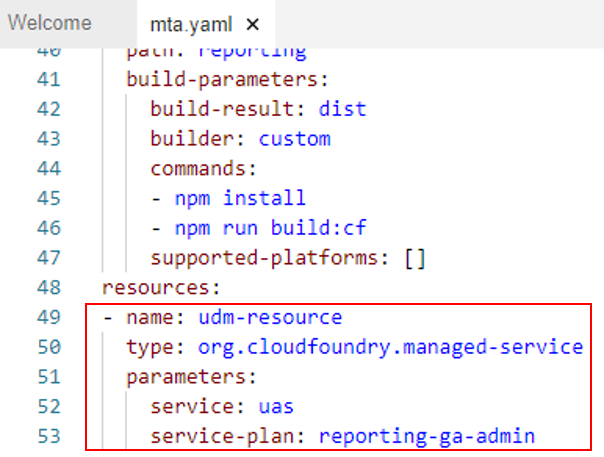
type: org.cloudfoundry.managed-service

parameters:

service: uas

service-plan: reporting-ga-admin

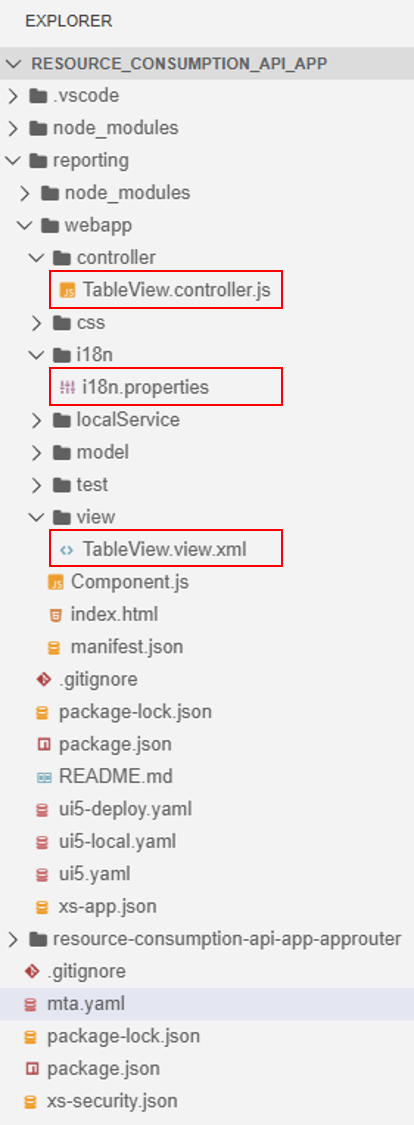
The result should look like this:



**CAUTION: The line inserts must be according to the code snippet to prevent errors.**  
  
Now all files are adjusted correctly, and it is possible to receive the data from the API. In the next step I will describe how to display the data from the “Subaccount Cost” API in a simple TableView.

**Displaying the received data in a simple UI**

To display the data in a TableView we need to change the files marked in the following screenshot.



*Tipp: To beautify your code and therefore getting a better overview, please have a look into following blog:*[*SAP Business Application Studio – How to beautify xml and javascript code*](https://blogs.sap.com/2021/04/30/sap-business-application-studio-how-to-beautify-xml-and-javascript-code/)

**Change the title of the app (Optional)**

Change the title of the app in the file “i18n.properties” (for e.g., the name of the Global Account of which you are receiving and displaying the data). This step is optional, but it improves the clarity of the app.

**Add a table to your view file - “<ViewName>.view.xml”**

Replace the “content” item with following code snippet:



<content>

<Table id="table1" items="{path: 'DataModel>/content'}" class="sapUiSizeCompact">

<columns>

<Column>

<Label text="Subaccount"/>

</Column>

<Column>

<Label text="Date"/>

</Column>

<Column>

<Label text="Service"/>

</Column>

<Column>

<Label text="Usage"/>

</Column>

<Column>

<Label text="Metric"/>

</Column>

<Column>

<Label text="Cost"/>

</Column>

<Column>

<Label text="Currency"/>

</Column>

</columns>

<ColumnListItem>

<Text text="{DataModel>subaccountName}"/>

<Text text="{DataModel>reportYearMonth}"/>

<Text text="{DataModel>serviceName}"/>

<Text text="{DataModel>usage}"/>

<Text text="{DataModel>metricName}"/>

<Text text="{DataModel>cost}"/>

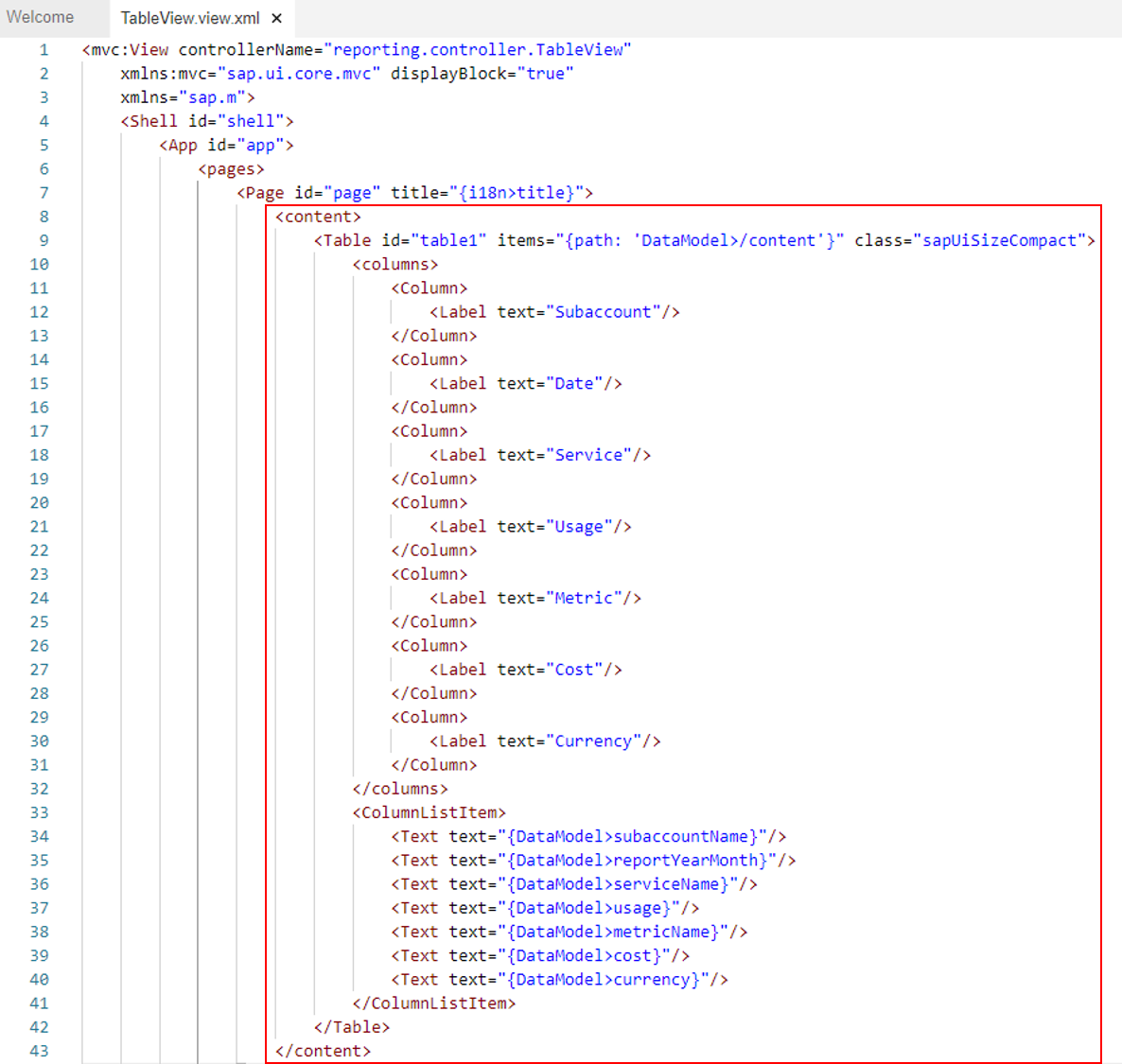
<Text text="{DataModel>currency}"/>

</ColumnListItem>

</Table>

</content>

The result should look like this:



**Add functions to your controller file - “<ControllerName>.controller.json”**

Paste the following code snippet into the “onInit” function:

var dataModel = this.getOwnerComponent().getModel("reporting");

this.getView().setModel(dataModel, "DataModel");

The result should look like this:

A screenshot of a computer program

Description automatically generated

Now all the components for the UI are defined and you are ready to build, deploy and start the app.

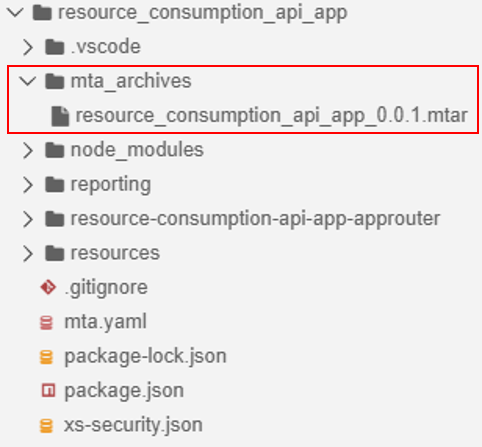
**Building and deploying the app**

**Building the app**

Do a right click on the “mta.yaml” file and select the option “Build MTA Project“.

**Deploying the app**

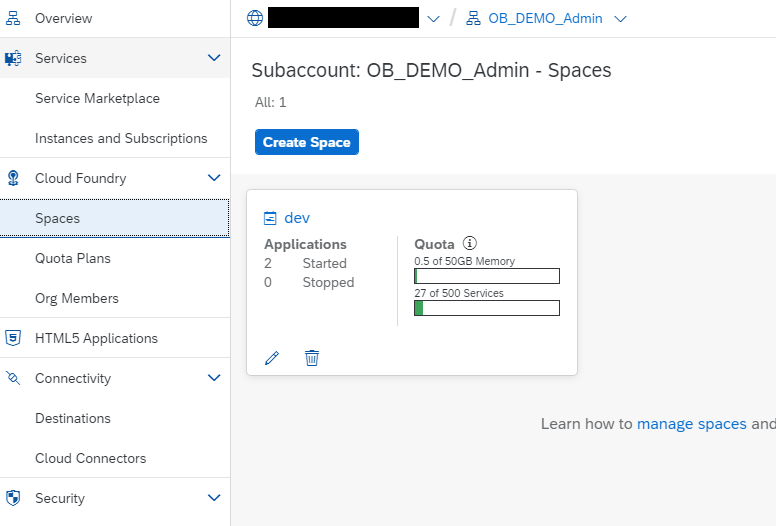
After successfully building the mta project a new folder “mta\_archives” with a “.mtar” file is created.



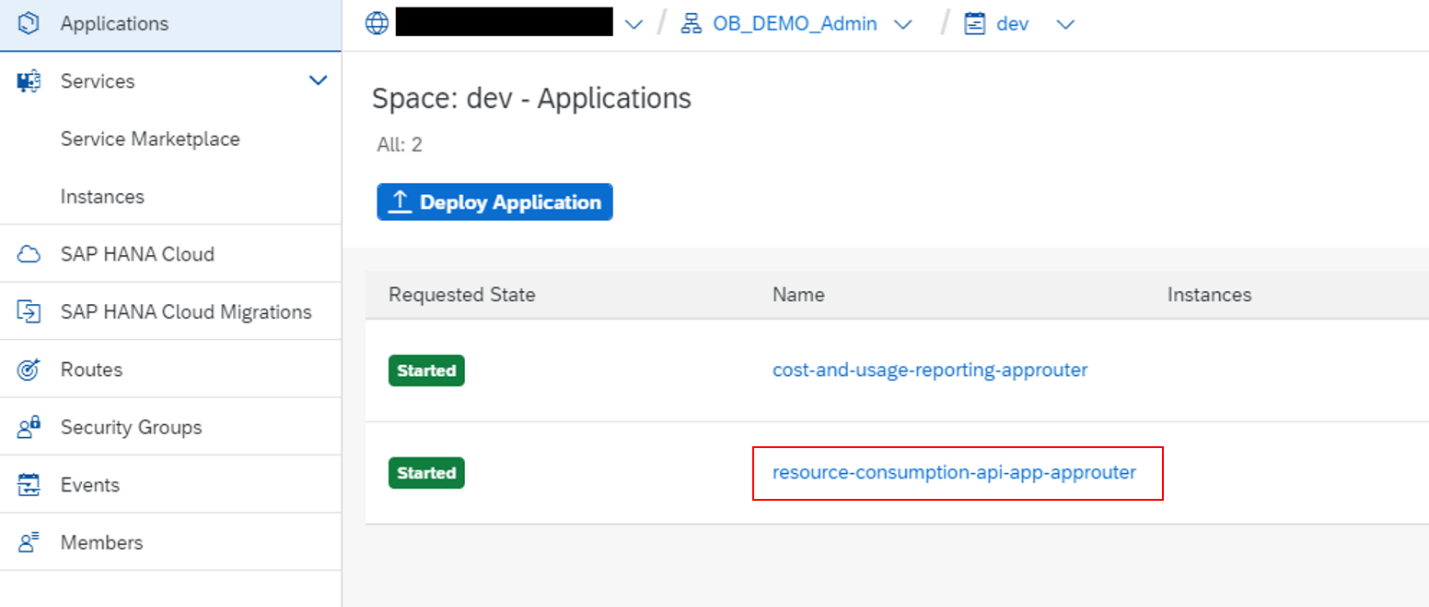
Do a right click on the “.mtar” file and select the option “Deploy MTA Archive” to start the deployment.  
  
**It can be necessary to provide your Cloud Foundry endpoint, your SAP credentials and the organization and the space where you want to deploy your app!**  
  
The information “Process Finished” indicates that the deployment was successful (this can take some moments).

**Starting the app in the SAP Business Technology Platform Cockpit**

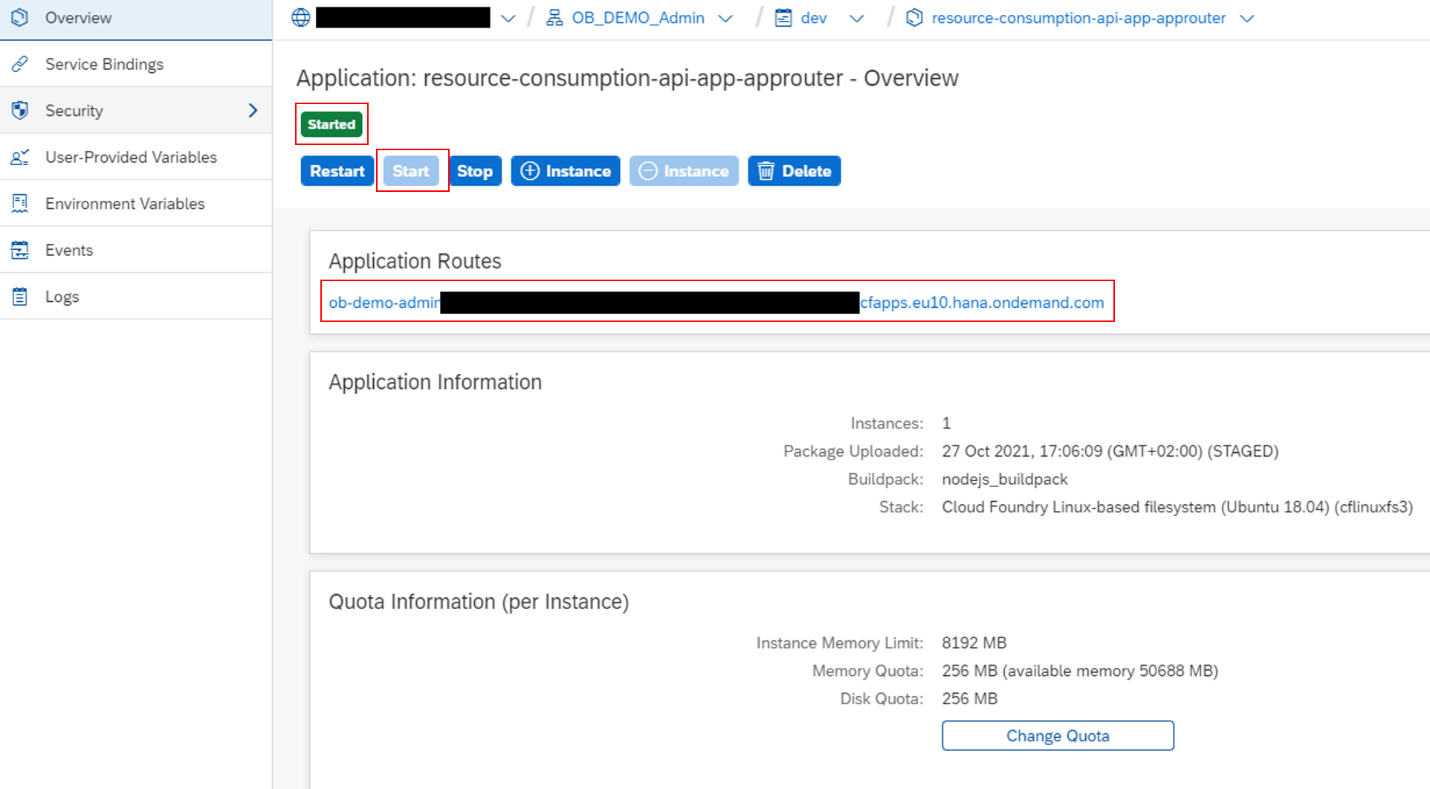
Enter your cockpit and navigate to the section “Spaces” under “Cloud Foundry”. Here you get an overview over the spaces you have created. Select the space, where you have deployed your app.



In the space you see all your deployed apps. Select the app you have just created.



After you have entered the overview page of your deployed app, first you need to make sure if the app is already started. The badge in the left corner will show you this information. If the badge is green and the text “started” can be seen, everything works fine. Otherwise please click on the button “start” to start the application manually.  
  
To enter the app you have to select the provided URL in the "Application Routes" section.



   
  
Now everything should work fine and you should see the data from the “Subaccount Cost” API in the constructed table.

A screenshot of a computer

Description automatically generated

**Conclusion**

A short recap, what have we achieved?  
  
With the described steps in this blog post we are able to create a SAPUI5 application from a template in SAP Business Application Studio, with some code adaption we can connect a SAP Business Technology REST API to the app and display the received data in a TableView, we manage to build and deploy the developed app and in the end we can see the data after starting the finished app from the SAP Business Technology Platform cockpit.

SAP Core Data Services (CDS) is a framework for data modeling that allows users to define and use data models on the database server instead of the application server

1. Right-click on the project main folder, the webapp folder, or the manifest.appdescr\_variant file and click **Preview Application**. Click **start-editor**.

The application opens in the canvas in UI Adaptation mode.

1. Expand the outline tree. You can see which element is a possible extension point. Then, select the element in the tree and you will see its parent marked and highlighted in the visual part of the editor. Right click on the highlighted element and click **Add fragment at extension point**.

In the dialog box, the default extension point is selected. Choose the target extension point and the index from the list where you want to add the fragment. You cannot reuse the same fragment multiple times.

1. To create a fragment:
   1. Enter a name for the fragment and click **Create**.

A fragment.xml file is created in the folder, **Your project name**  **webapp**  **changes**  **fragments** and opens in the editor.

* 1. Define the fragment. Save and close the .xml file. For more information, see [SAPUI5 documentation](https://sapui5.hana.ondemand.com/#/topic/2c677b574ea2486a8d5f5414d15e21c5).

An associated addXML.change file is created for every fragment in the folder, **Your project name**  **webapp**  **changes**. This change file contains the reference to the fragment.xml file, selected target aggregation, and index.

1. Navigate to the canvas to see the changes that you made.

To load the added change, you need to reload the browser tab for the adaptation project editor and application preview (if opened in a separate tab).

**Why use SAP Fiori elements?**

* High **development efficiency** to cover what 80% of all apps need

You do not need to build the UI over and over again. Just reuse the common features required by most applications. They are provided by the SAP Fiori elements floorplans.

* **Design consistency**

Predefined floorplans, views, and controllers ensure UI consistency within and across similar apps. Apps created using SAP Fiori elements are kept up-to-date as they are automatically adapted to the most recent design guidelines.

* **Decoupling of UI and business logic**

The metadata-driven development model uses semantic annotations and significantly reduces the amount of front-end code. Developers can focus on the business logic.

**Note**

The initial effort for creating an app using SAP Fiori elements might be higher than creating a freestyle SAP Fiori app. However, you will be richly rewarded for this effort after you've created more apps this way because your apps will benefit from using the framework and the included features, as described below.

You can create apps using the following SAP Fiori elements floorplans:

* [List Report and Object Page](https://sapui5.hana.ondemand.com/1.71.69/#/topic/c0eec49db81a441e878f528c8f3d28de.html)

SAP Fiori elements contain predefined templates for list reports and object pages. A list report lets users filter, view, and work with items (objects) organized in list (table) format. The list report is typically used in conjunction with an object page. This object page lets users work with objects, providing functions for viewing, editing, and creating objects.

* [Worklist](https://sapui5.hana.ondemand.com/1.71.69/#/topic/d1d588f1061b4bac96a1facb80d3f3a2.html)

A worklist displays a collection of items to be processed by the user. There is no need for sophisticated filtering. Working through the item list usually involves reviewing details of the list items and taking action. In most cases, the user has to either complete a work item or delegate it.

* [Overview Pages](https://sapui5.hana.ondemand.com/1.71.69/#/topic/c64ef8c6c65d4effbfd512e9c9aa5044.html)

An overview page is a data-driven SAP Fiori app for organizing large amounts of information. Information is visualized in a card format in an attractive and efficient way. Different cards are used for different types of content. The user-friendly experience makes viewing, filtering, and acting on data quick and easy. While presenting the big picture, business users can focus on the most important tasks enabling faster decision making as well as immediate action.

* [Analytical List Page](https://sapui5.hana.ondemand.com/1.71.69/#/topic/3d33684b08ca4490b26a844b6ce19b83.html)

Analytical list page is a SAP Fiori elements application for detailed analytics. It lets you analyze data from different perspectives, to investigate a root cause, and to act on transactional content. You can identify relevant areas within data sets or significant single instances using data visualization and business intelligence. All this can be done seamlessly on one page.

Navigation –

**Conventions**

* Configure the router in the manifest.json descriptor file
* Initialize the router exactly once
* Initialize the router in the component
* The init function of the component is automatically invoked by SAPUI5 when the component is instantiated. Our component inherits from the base class sap.ui.core.UIComponent and it is obligatory to make the super call to the init function of the base class in the overridden init method.
* **// call the init function of the parent**
* **UIComponent.prototype.init.apply(this, arguments);**
* **// create the views based on the url/hash**
* **this.getRouter().initialize();**
* We override the init function and call the parent’s init function first. We get a reference to the router and call initialize() on it. The router is instantiated automatically with the configuration loaded in the descriptor. The routing events and our configuration in the descriptor are now automatically enabled in the app.

Inside manifest.json – sap.ui5

"rootView": {

"viewName": "sap.ui.demo.nav.view.App",

"type": "XML",

"async": true,

"id": "app"

},

**"routing": {**

**"config": {**

**"routerClass": "sap.m.routing.Router",**

**"viewType": "XML",**

**"viewPath": "sap.ui.demo.nav.view",**

**"controlId": "app",**

**"controlAggregation": "pages",**

**"transition": "slide",**

**"async": true**

**},**

**"routes": [{**

**"pattern": "",**

**"name": "appHome",**

**"target": "home"**

**}],**

**"targets": {**

**"home": {**

**"viewId": "home",**

**"viewName": "Home",**

**"viewLevel" : 1**

**}**

**}**

**}**

**"bypassed": {**

**"target": "notFound"**

**},**

Initialize –

**getRouter : function () {**

**return UIComponent.getRouterFor(this);**

**},**

Q. Use of Manifest.json

This clearly separates the application coding from the configuration settings and makes our app even more flexible.

Q. Why addDependent is used while defining the fragment

// connect dialog to the root view of this component (models, lifecycle)

oView.addDependent(oDialog);

Filter:

**onFilterInvoices : function (oEvent) {**

**// build filter array**

**var aFilter = [];**

**var sQuery = oEvent.getParameter("query");**

**if (sQuery) {**

**aFilter.push(new Filter("ProductName", FilterOperator.Contains, sQuery));**

**}**

**// filter binding**

**var oList = this.byId("invoiceList");**

**var oBinding = oList.getBinding("items");**

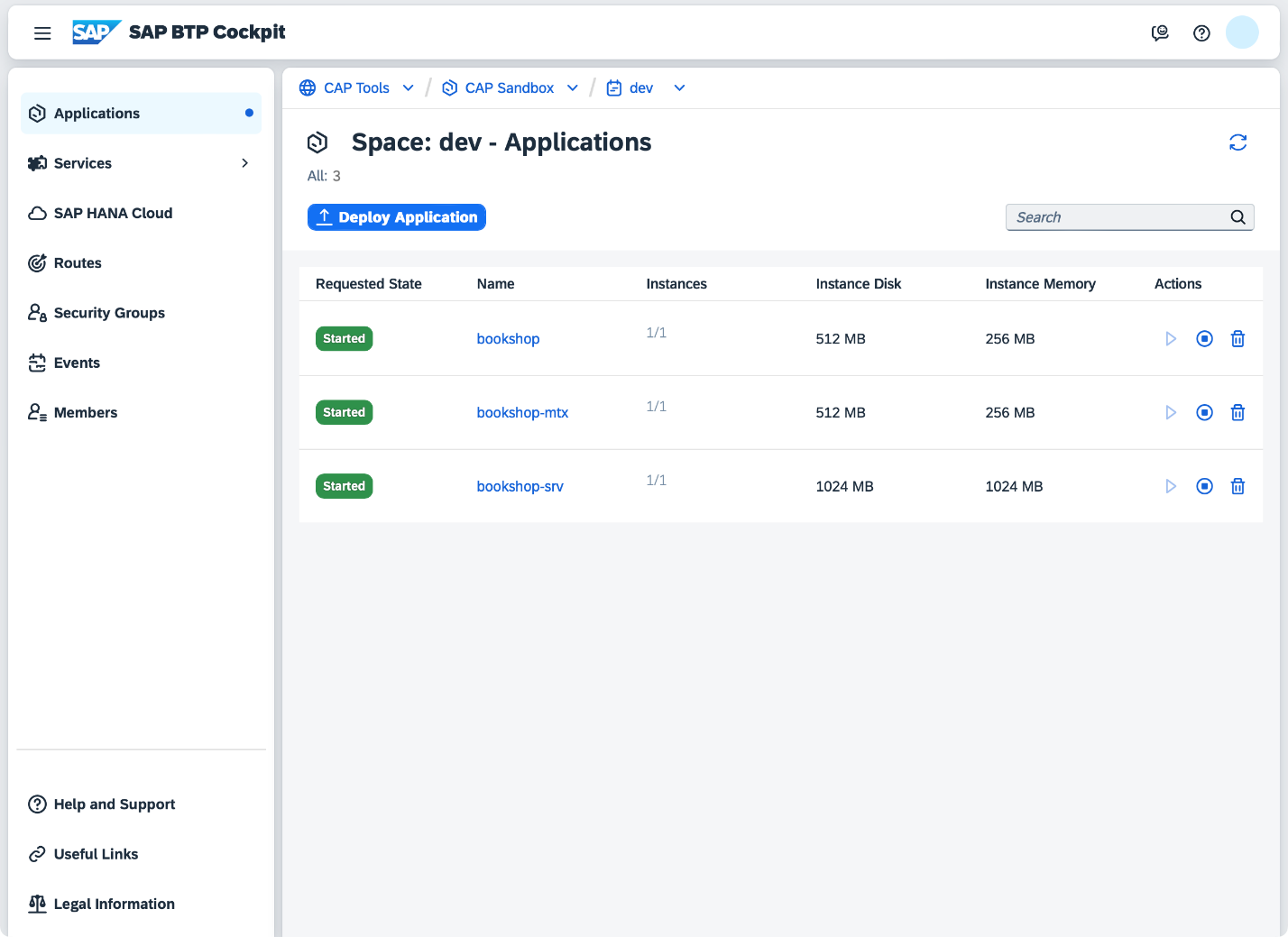
**oBinding.filter(aFilter)**

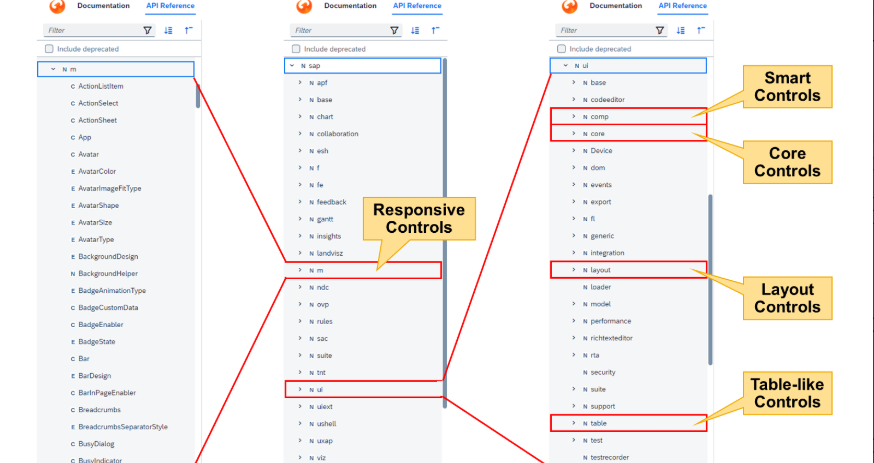
Q. What is the use of SAP Fiori Elements

SAP Fiori elements is a framework that comprises the most commonly used floorplan templates and is designed to:

* Speed up development by reducing the amount of front-end code needed to build SAP Fiori apps
* Ensure stable, optimized UI code out of the box
* Deliver high-quality SAPUI5 applications to end users
* Drive UX consistency and compliance with the latest SAP Fiori design guidelines

OData is a standard protocol for creating and consuming data by using simple HTTP and REST APIs for create, read, update, delete (CRUD) operations.

Go to Space in BTP Cockpit – Application – list of all deployed application - URL



A screenshot of a computer

Description automatically generated

https://www.odata.org/ - To refer about odata

A diagram of software development

Description automatically generated

SAPUI5 provides the following two types of components:

* *Faceless components* (class: sap.ui.core.Component)

Faceless components have no user interface and are used for coding when no UI elements are needed, such as for a service that provides data from a back-end system.

* *UI components* (class: sap.ui.core.UIComponent)

UI components extend faceless components and add rendering functionality to the component. They represent a screen area or element on the UI along with the respective settings and metadata. This component type will be covered in the course.

A diagram of a software application

Description automatically generated