


# Develop SAP Business One extensions on the SAP Cloud Platform



## TABLE OF CONTENTS

PREREQUISITES.....	4
i. Download and Install Development Tools.....	4
ii. Create a SAP Cloud platform Neo trial account .....	5
iii. Activate Web IDE Full Stack service.....	6
v. Activate Build service in SAP Cloud Platform cockpit .....	9
vi. Activate Build feature in WebIDE.....	9
vii. Activate a Cloud Foundry trial account .....	10
STEP 1: CREATE A BUILD HIGH FIDELITY PROTOTYPE .....	12
STEP 2: IMPORT YOUR BUILD PROTOTYPE INTO WEBIDE .....	12
i. Create the Project.....	12
ii. Test the project with mock data.....	15
iii. Create a destination pointing to your backend server .....	16
iv. Connect to your real B1 backend server .....	16
v. Extra SAP Business One backend configuration steps.....	18
STEP 3: CLONE A NODEJS APP .....	24
STEP 4: DEPLOY THE NODEJS APP INTO SAP CLOUD FOUNDRY.....	25
i. SAP Cloud Platform Cloud Foundry Environment.....	25
ii. Create the backing services.....	26
iii. Create the required destinations .....	27
iv. Deploy the smbmkmt app.....	28
i. Configuration for only B1 or ByD systems.....	29
ii. Configure the SMB Mktplace backend.....	30
iii. Initialize the SMB Mktplace backend .....	31
iv. Test the SMB Mktplace backend /SimilarItems API .....	31
STEP 5: CONSUME THE NODEJS APP FROM THE SAP FIORI APP .....	33
i. Create a destination pointing to your smbmkmt backend.....	33
ii. Create a new JSON model.....	34
iii. Change the Image control in the Page1.view.xml file. ....	34
iv. Create a FileUploader control. ....	35
v. Bind the Matching Items Table to our backend properties.....	36
vi. Implement the Page1.controller.js. ....	38
STEP 6: ADD A NEW SERVICE TO THE NODEJS APPLICATION .....	41
STEP 7: CALL THE NEW NODEJS SERVICE FROM YOUR SAP FIORI APP.....	44



The objective of this hands on is to put in practice how to develop SAP Business One extensions on SAP Cloud Platform.

The exercise will be composed by

- Step 1: Create a Build prototype connecting to B1
- Step 2: Import the Build prototype into a SCP WebIDE Fiori application and connect to your real B1 backend
- Step 3: Clone an existing NodeJS application
- Step 4: Deploy the server side NodeJS application to the Cloud Foundry environment
- Step 5: Modify the SAP Fiori app to consume the server side NodeJS application
- Step 6: Add a new service to the NodeJS application and consume it from SAP Fiori
- Step 7: Call the new NodeJS service from the SAP Fiori app

This hands-on exercise will require several steps, please follow them in the proposed order as each step is counting on the precedent steps.

## PREREQUISITES

### i. Download and Install Development Tools

Download and install git version control on your system from the following link

<https://git-scm.com/downloads>



We will also make use of the Cloud Foundry Environment.

To do so, we need the Cloud Foundry command line interface (CLI)

You can download it and install it if the CF CLI for your operating system on.

<https://github.com/cloudfoundry/cli#downloads>

#### Downloads

##### Installing using a package manager

Mac OS X and Linux using Homebrew via the [cloudfoundry tap](#):

```
brew install cloudfoundry/tap/cf-cli
```

Debian and Ubuntu based Linux distributions:

```
# ...first add the Cloud Foundry Foundation public key and package repository to your system
wget -q -O - https://packages.cloudfoundry.org/debian/cli.cloudfoundry.org.key | sudo apt-key add -
echo "deb https://packages.cloudfoundry.org/debian stable main" | sudo tee /etc/apt/sources.list.d/cloudfou
# ...then, update your local package index, then finally install the cf CLI
sudo apt-get update
sudo apt-get install cf-cli
```

Enterprise Linux and Fedora systems (RHEL6/CentOS6 and up):

```
# ...first configure the Cloud Foundry Foundation package repository
sudo wget -O /etc/yum.repos.d/cloudfoundry-cli.repo https://packages.cloudfoundry.org/fedora/cloudfoundry-
# ...then, install the cf CLI (which will also download and add the public key to your system)
sudo yum install cf-cli
```

#### Installers and compressed binaries

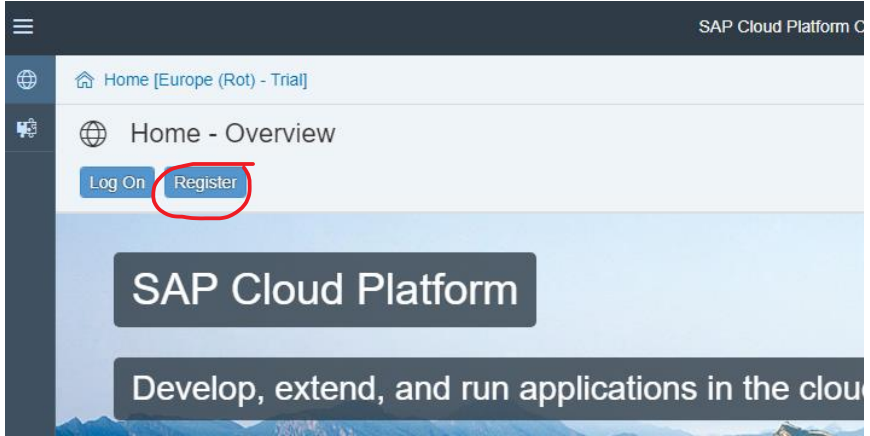
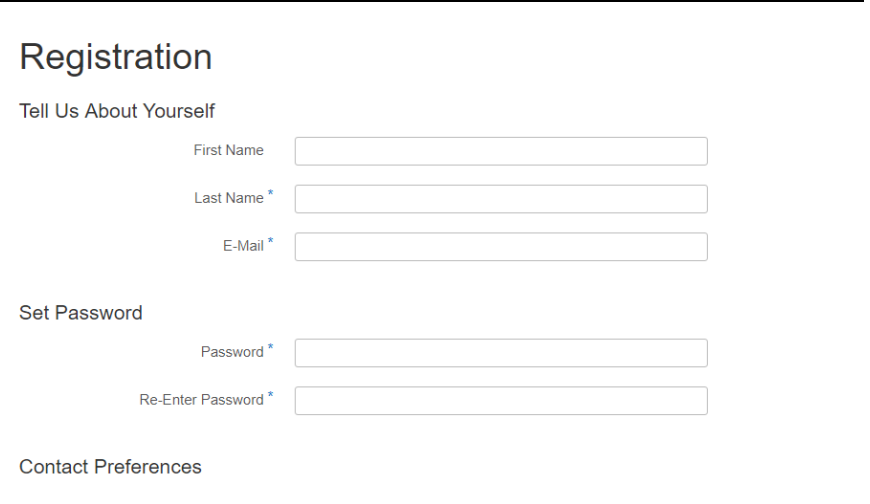
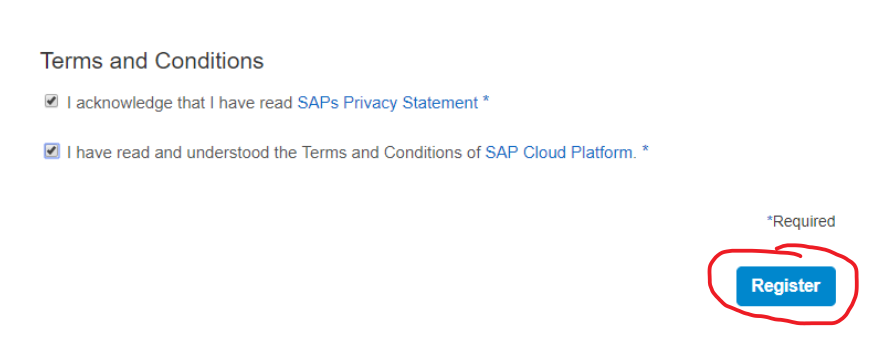
	Mac OS X 64 bit	Windows 64 bit	Linux 64 bit
Installers	<a href="#">pkg</a>	<a href="#">zip</a>	<a href="#">rpm / deb</a>
Binaries	<a href="#">tgz</a>	<a href="#">zip</a>	<a href="#">tgz</a>

## ii. Create a SAP Cloud platform Neo trial account

The exercises proposed in this hands on are implemented on top of the SAP Cloud Platform.

If you have already a trial SAP Cloud Platform account, you can skip this step.

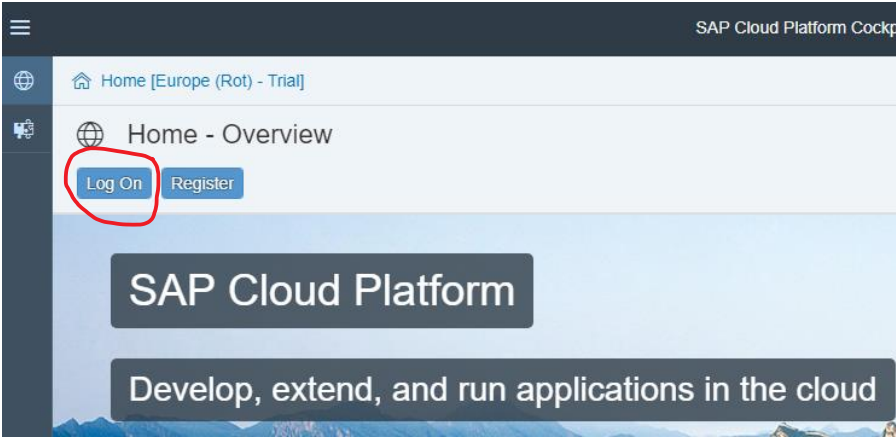
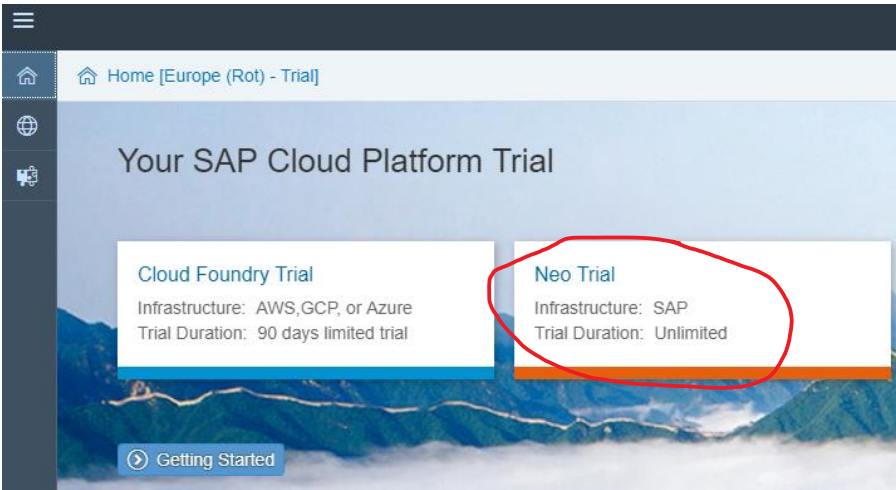
To create a trial SAP Cloud Platform account, go to the following link:

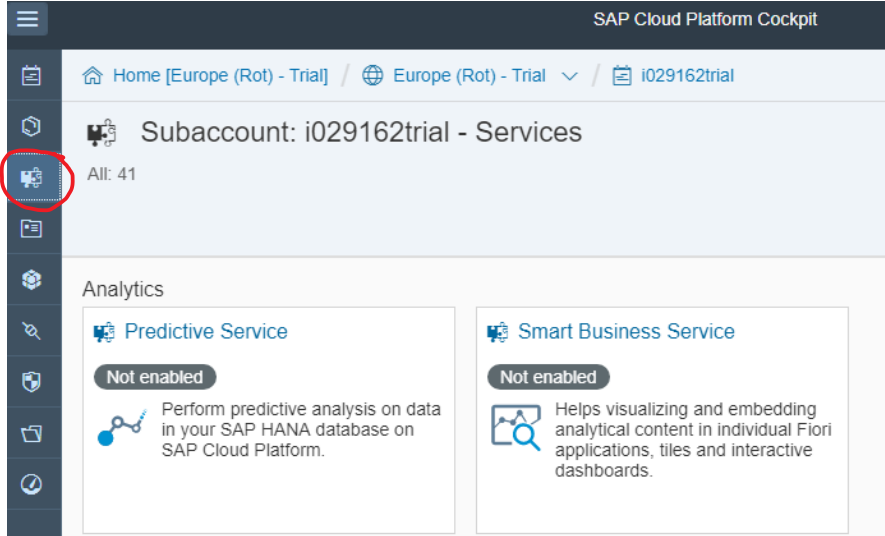
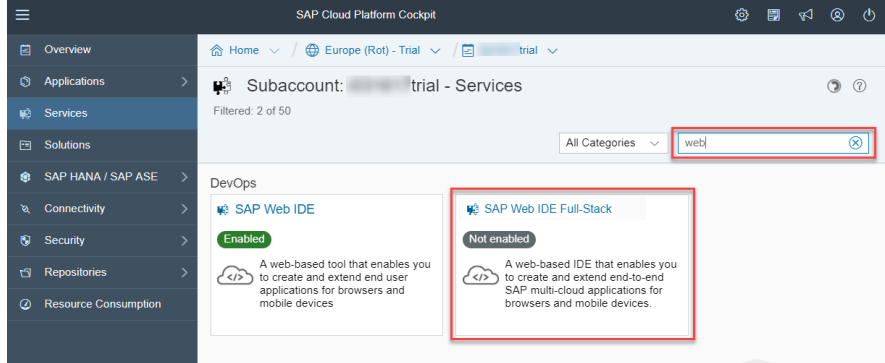
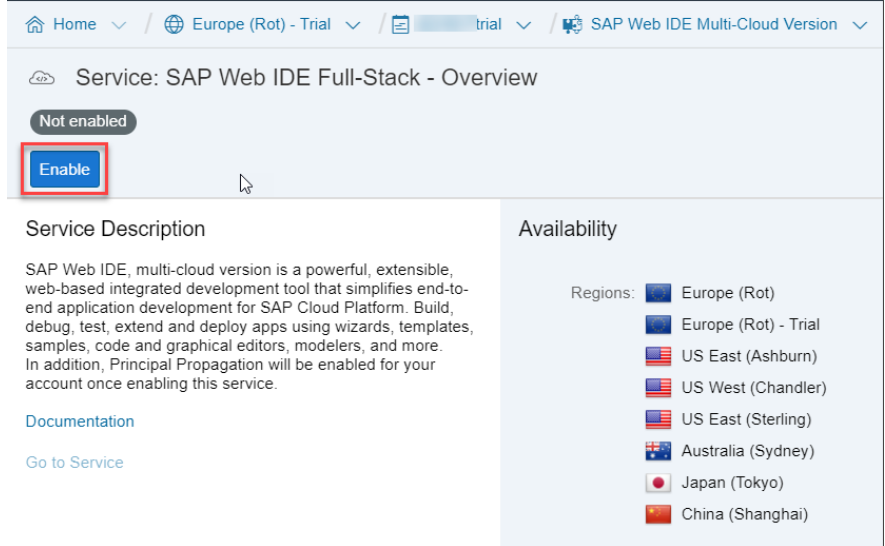
Explanation	Screenshot
<p>To create a trial SAP Cloud Platform account, go to the following link:</p> <p><a href="https://account.hanatrial.ondemand.com">https://account.hanatrial.ondemand.com</a></p> <p>Press the Register button</p>	 The screenshot shows the SAP Cloud Platform Home - Overview page. The 'Register' button is circled in red. The page includes a header with 'SAP Cloud Platform C', a navigation bar with 'Home [Europe (Rot) - Trial]', and a main content area with 'Home - Overview', 'Log On', and 'Register' buttons. Below this is a large banner with 'SAP Cloud Platform' and 'Develop, extend, and run applications in the cloud'.
<p>Enter all your details</p>	 The screenshot shows the SAP Cloud Platform Registration form. It includes sections for 'Tell Us About Yourself' (First Name, Last Name, E-Mail), 'Set Password' (Password, Re-Enter Password), and 'Contact Preferences'. The 'Register' button is circled in red.
<p>Accept the terms and conditions by checking both check boxes and press "Register".</p>	 The screenshot shows the SAP Cloud Platform Terms and Conditions page. It includes checkboxes for 'I acknowledge that I have read SAPs Privacy Statement' and 'I have read and understood the Terms and Conditions of SAP Cloud Platform'. The 'Register' button is circled in red.

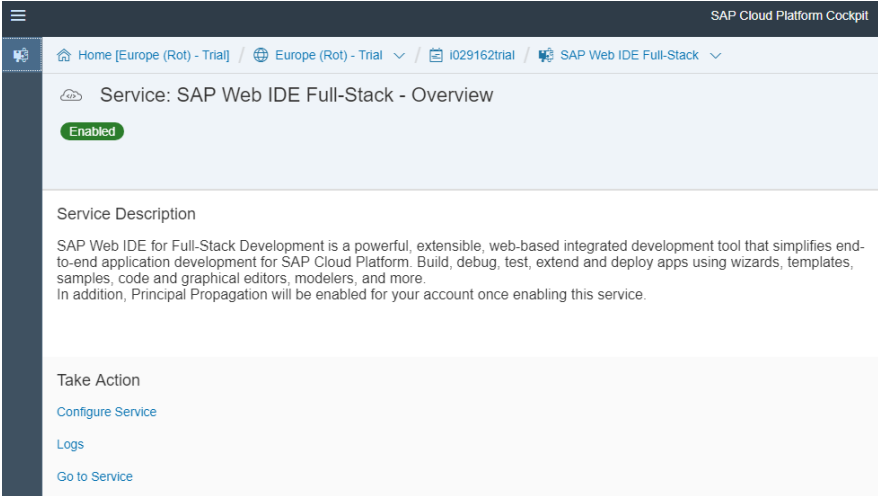
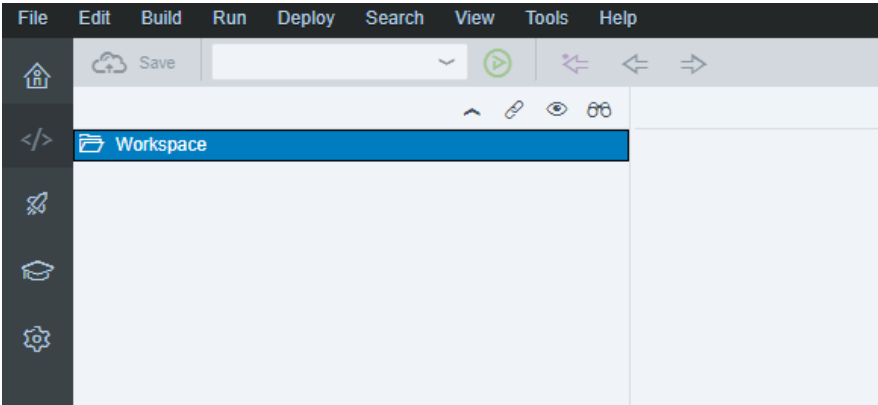
### iii. Activate Web IDE Full Stack service

We will use Web IDE Full Stack for the creation and implementation of our application.  
Web IDE is offered as a service on the SAP Cloud Platform.

To activate Web IDE Full Stack service please follow the steps here below, if you already have Web IDE Full Stack service active in your account please skip this step.

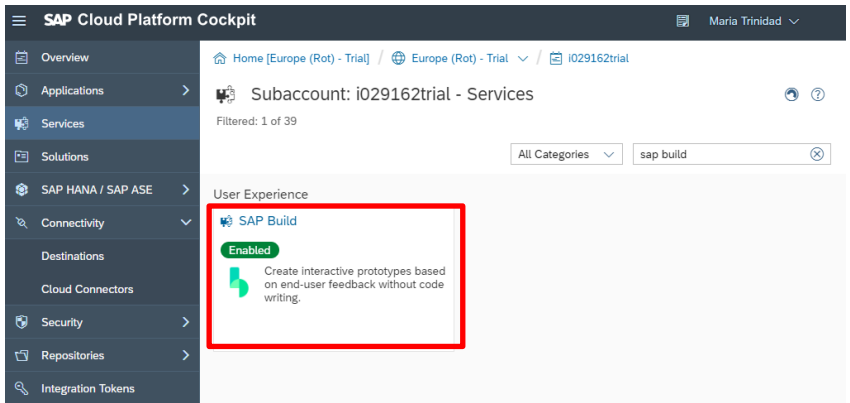
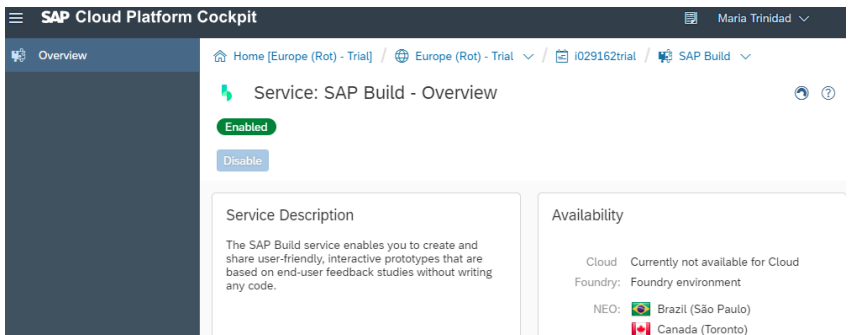
Explanation	Screenshot
Open your trial SAP Cloud Platform account from the following link: <a href="https://account.hanatrial.ondemand.com">https://account.hanatrial.ondemand.com</a>	
Press the Log On button if you are not automatically logged in	 The screenshot shows the SAP Cloud Platform Cockpit login page. The header includes a hamburger menu, a home icon, and the text 'Home [Europe (Rot) - Trial]'. Below this is a 'Home - Overview' section with a 'Log On' button circled in red and a 'Register' button. The main content area features the 'SAP Cloud Platform' logo and the tagline 'Develop, extend, and run applications in the cloud'.
After login if you are proposed between Cloud Foundry Trial and Neo Trial please choose Neo Trial.	 The screenshot shows the 'Your SAP Cloud Platform Trial' selection page. It features two trial options: 'Cloud Foundry Trial' (Infrastructure: AWS, GCP, or Azure; Trial Duration: 90 days limited trial) and 'Neo Trial' (Infrastructure: SAP; Trial Duration: Unlimited). The 'Neo Trial' option is circled in red. A 'Getting Started' button is visible at the bottom.

Explanation	Screenshot
Select the Services icon on the left side bar.	 <p>The screenshot shows the SAP Cloud Platform Cockpit interface. The left sidebar contains several icons, and the 'Services' icon (represented by a puzzle piece) is circled in red. The main area displays the 'Subaccount: i029162trial - Services' page with a list of services, including 'Predictive Service' and 'Smart Business Service', both marked as 'Not enabled'.</p>
Enter <b>Web</b> in the search edit text. Click on the SAP Web IDE Full Stack box.	 <p>The screenshot shows the SAP Cloud Platform Cockpit interface with the search bar at the top right containing the text 'web'. The search results show two services: 'SAP Web IDE' (marked 'Enabled') and 'SAP Web IDE Full-Stack' (marked 'Not enabled'). The 'SAP Web IDE Full-Stack' service box is highlighted with a red rectangle.</p>
Click <b>Enable</b> . This may take a few minutes.	 <p>The screenshot shows the 'Service: SAP Web IDE Full-Stack - Overview' page. The 'Not enabled' status is shown at the top, and the 'Enable' button is highlighted with a red rectangle. Below the button, the 'Service Description' and 'Availability' sections are visible. The 'Availability' section lists various regions, including Europe (Rot), Europe (Rot) - Trial, US East (Ashburn), US West (Chandler), US East (Sterling), Australia (Sydney), Japan (Tokyo), and China (Shanghai).</p>

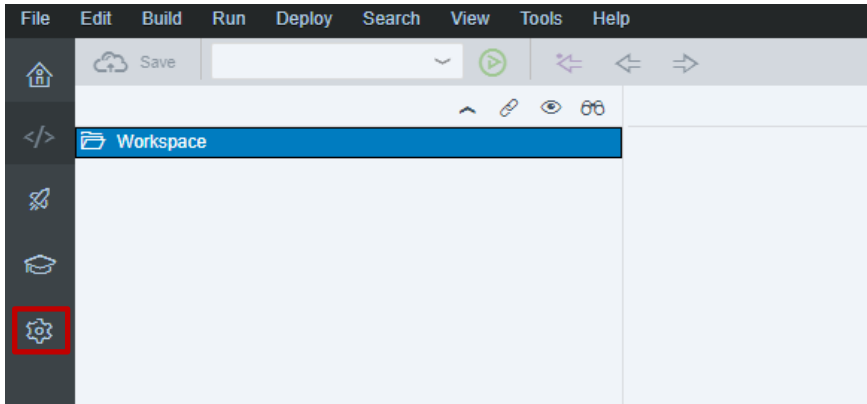
Explanation	Screenshot
<p>Once Enabled select the link <b>Go to Service</b> to open Web IDE Full Stack.</p>	 <p>The screenshot shows the SAP Cloud Platform Cockpit interface. The breadcrumb navigation at the top reads: Home [Europe (Rot) - Trial] / Europe (Rot) - Trial / i029162trial / SAP Web IDE Full-Stack. The main heading is 'Service: SAP Web IDE Full-Stack - Overview', followed by a green 'Enabled' status button. Below this is a 'Service Description' section stating: 'SAP Web IDE for Full-Stack Development is a powerful, extensible, web-based integrated development tool that simplifies end-to-end application development for SAP Cloud Platform. Build, debug, test, extend and deploy apps using wizards, templates, samples, code and graphical editors, modelers, and more. In addition, Principal Propagation will be enabled for your account once enabling this service.' At the bottom, under 'Take Action', there are three links: 'Configure Service', 'Logs', and 'Go to Service'.</p>
<p>Web IDE opens with an empty Workspace unless you already developed applications with Web IDE in the past.</p>	 <p>The screenshot displays the SAP Web IDE development environment. It features a top menu bar with 'File', 'Edit', 'Build', 'Run', 'Deploy', 'Search', 'View', 'Tools', and 'Help'. Below the menu is a toolbar with icons for 'Save', a search icon, a play icon, and navigation arrows. A left sidebar contains icons for home, code editor, explorer, and settings. The main area shows a file explorer with a folder named 'Workspace' selected, indicating an empty workspace.</p>

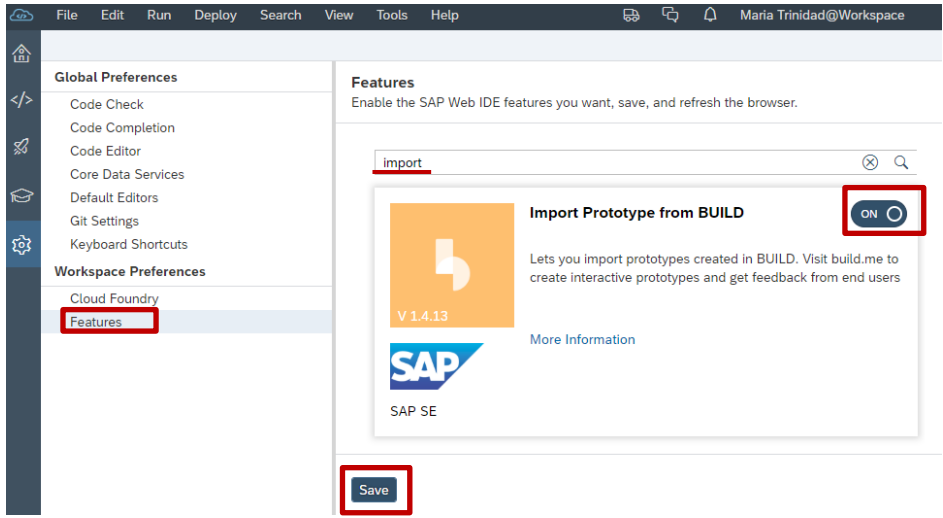


## v. Activate Build service in SAP Cloud Platform cockpit

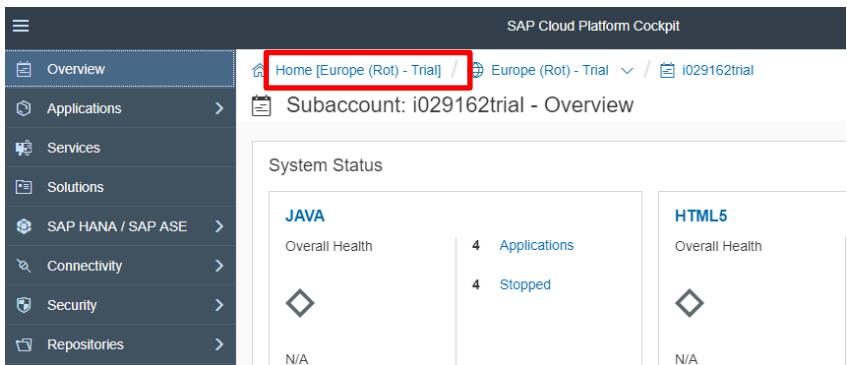
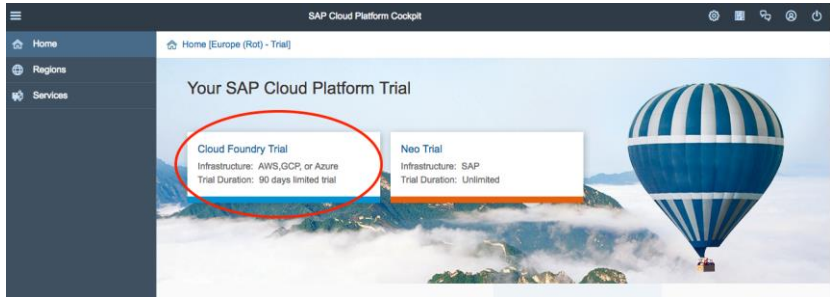
Explanation	Screenshot
<p>We need to activate the SAP Build service.</p> <p>Follow similar steps as in previous prerequisite to see the list of Services.</p> <p>Search for SAP Build service.</p>	
<p>If not active select it and click the button Enable to activate it (in my case it is already Enabled).</p>	

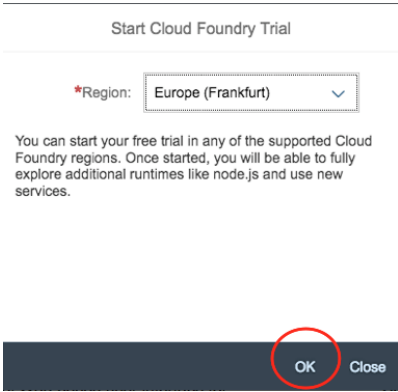
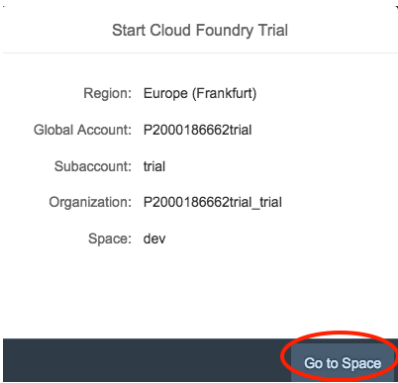
## vi. Activate Build feature in WebIDE

Explanation	Screenshot
<p>Open <b>SAP Web IDE Full Stack</b>.</p> <p>Check the prerequisites sections “Create a SAP Cloud platform trial account” and “Activate Web IDE Full Stack service” if you don’t know how to open WebIDE Full Stack.</p> <p>Click on the <b>Preferences</b> icon.</p>	

Explanation	Screenshot
<p>Select <b>Features</b> menu.</p> <p><b>Search</b> for <b>import</b>. Change the button to <b>ON</b> to get the <b>Import Prototype from BUILD</b> feature available in WebIDE.</p> <p>Click <b>Save!</b></p>	

#### vii. Activate a Cloud Foundry trial account

Explanation	Screenshot
<p>We need to activate the SAP Cloud Platform Cloud Foundry Environment.</p> <p>On the SAP Cloud Platform Dashboard click on the <b>HOME</b> link.</p>	
<p>From the SAP CP Home Screen, click on <b>Cloud Foundry Trial</b>.</p>	

Explanation	Screenshot
<p>Select the Trial Region that suits your location. And Click on <b>OK</b></p>	
<p>This will initialize your Cloud Foundry Trial and create a DEV space (where the solutions will be deployed).</p>	

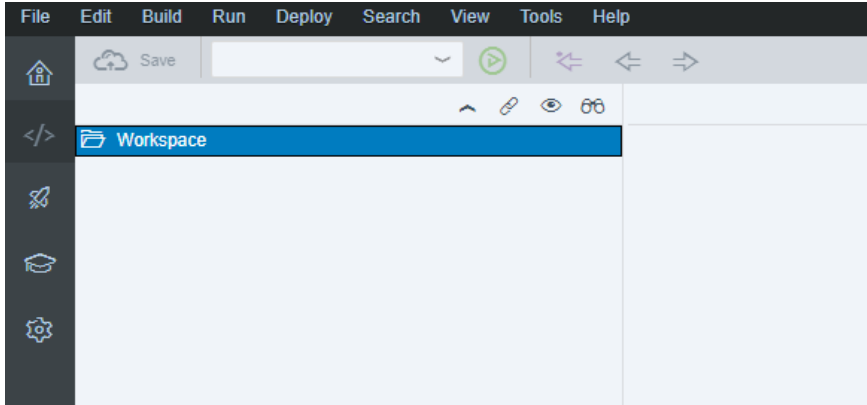
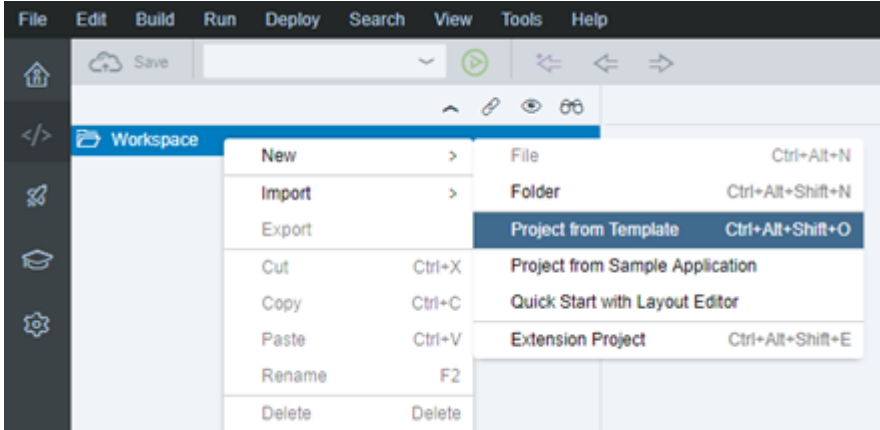
## STEP 1: CREATE A BUILD HIGH FIDELITY PROTOTYPE

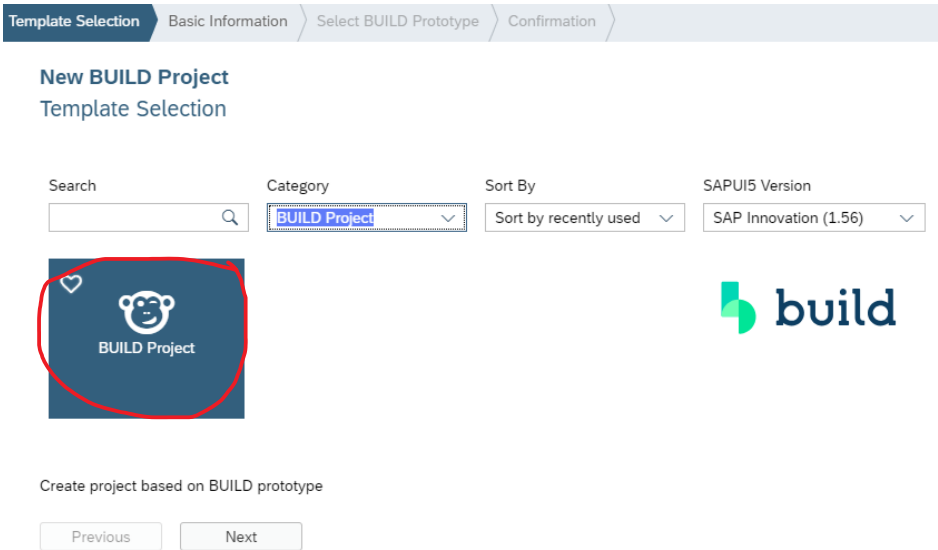
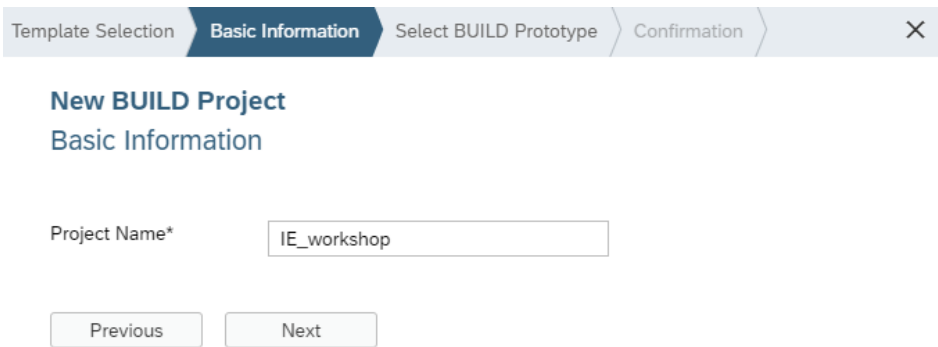
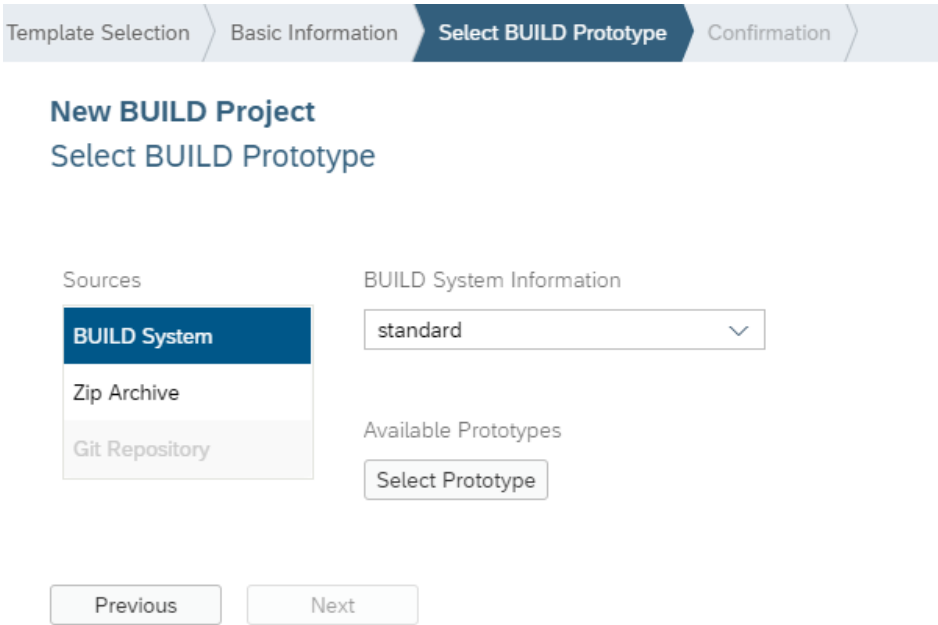
Follow the steps of the HandsOn\_Build\_B1\_Instructions.pdf document.

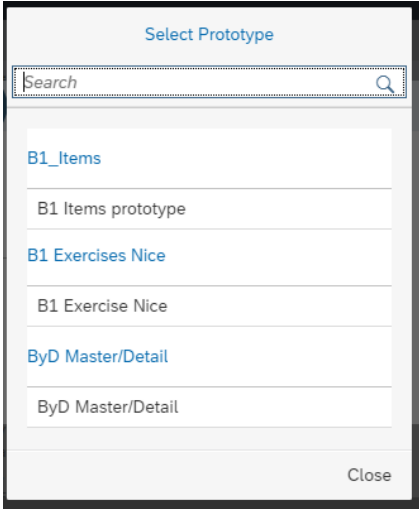
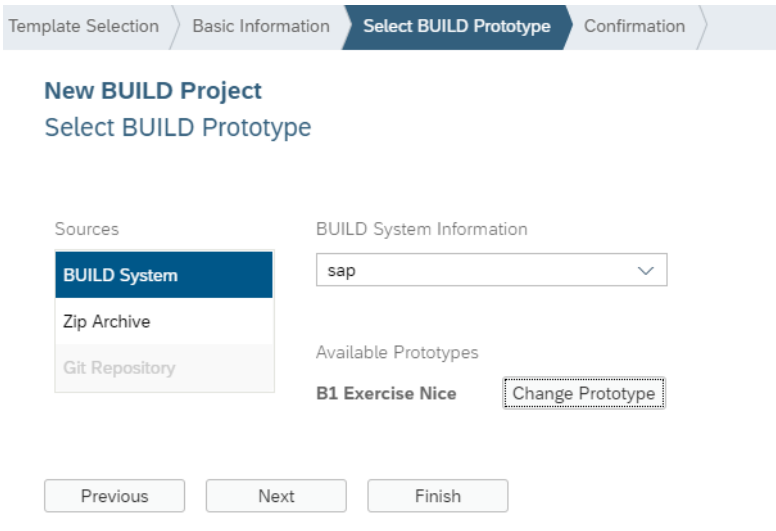
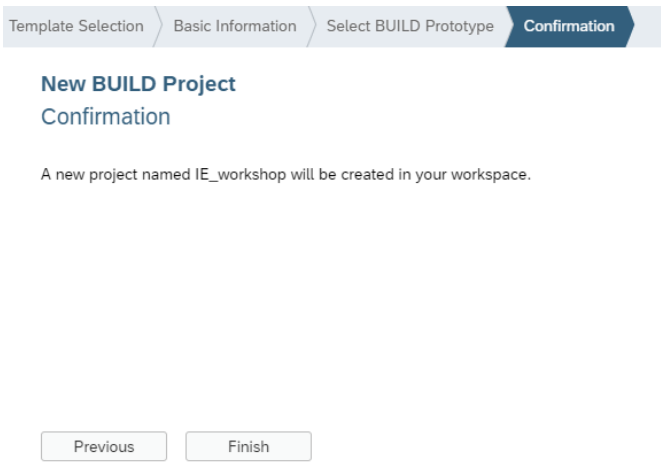
## STEP 2: IMPORT YOUR BUILD PROTOTYPE INTO WEBIDE

The objective of this first exercise is to create a SAP Fiori app from your Build prototype.

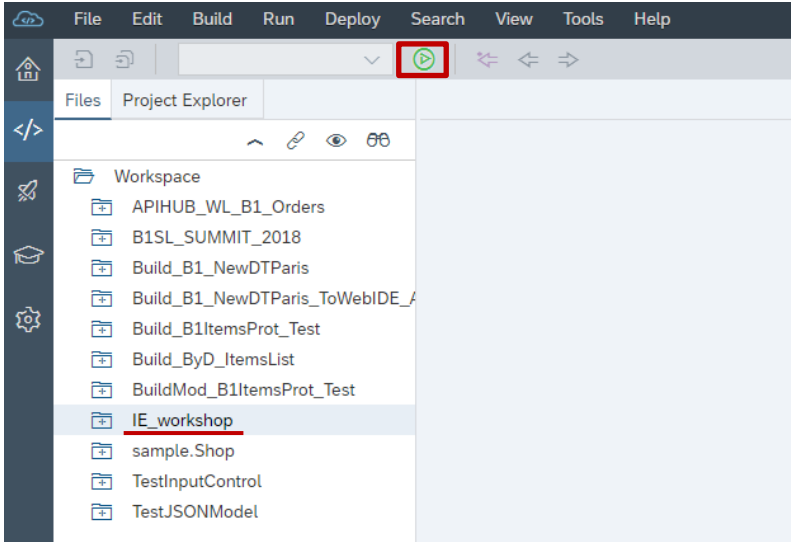
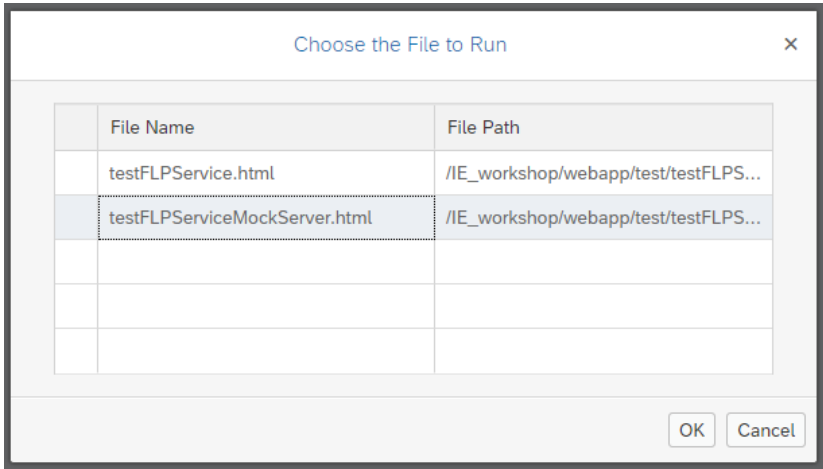
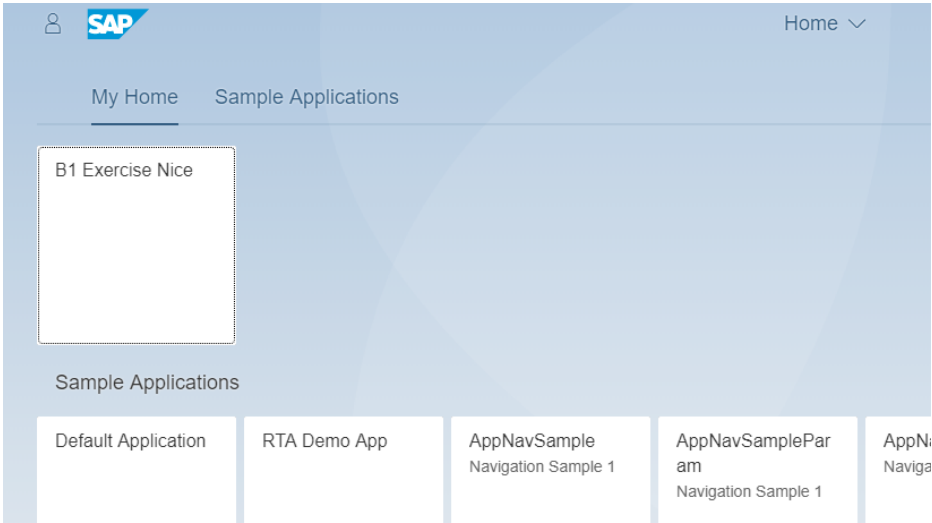
### i. Create the Project

Explanation	Screenshot
<p>Open SAP Web IDE Full Stack.</p> <p>Check the prerequisites sections “Create a SAP Cloud platform trial account” and “Activate Web IDE Full Stack service” if you don’t know how to open WebIDE Full Stack.</p>	 The screenshot shows the SAP Web IDE Full Stack interface. The top menu bar includes File, Edit, Build, Run, Deploy, Search, View, Tools, and Help. Below the menu bar is a toolbar with icons for Save, Run, and other actions. The main workspace area is currently empty, with a 'Workspace' tab selected in the left sidebar.
<p>Right click on your Workspace and select <b>New</b> -&gt; <b>Project from Template</b>.</p>	 The screenshot shows the SAP Web IDE Full Stack interface with the 'Workspace' tab selected. A right-click context menu is open over the workspace area. The menu options are: New (with a right arrow), Import (with a right arrow), Export, Cut (Ctrl+X), Copy (Ctrl+C), Paste (Ctrl+V), Rename (F2), and Delete. The 'New' option is expanded, showing a sub-menu with the following options: File (Ctrl+Alt+N), Folder (Ctrl+Alt+Shift+N), Project from Template (Ctrl+Alt+Shift+O), Project from Sample Application, Quick Start with Layout Editor, and Extension Project (Ctrl+Alt+Shift+E). The 'Project from Template' option is highlighted.

Explanation	Screenshot
<p>Select the <b>BUILD Project</b> template.</p> <p>Press <b>Next</b>.</p> <p>If you don't see this template, change the Category to <b>BUILD Project</b> or <b>All categories</b>.</p>	
<p>Enter a <b>Project Name</b>.</p> <p>Press <b>Next</b>.</p>	
<p>Select <b>BUILD System</b> source.</p> <p>Select <b>standard</b> BUILD System Information.</p> <p>Press <b>Select Prototype</b> to get the list of available BUILD prototypes.</p> <p><i>You might be prompted to enter your BUILD User Name and Password. Enter your credentials and press Log In.</i></p>	

Explanation	Screenshot
<p>Select the prototype you shared previously in BUILD.</p> <p>Press <b>Close</b>.</p>	
<p>Press <b>Next</b>.</p>	
<p>Press <b>Finish</b>.</p>	

## ii. Test the project with mock data

Explanation	Screenshot												
<p>Go to your workspace, the new project should be listed.</p> <p>Select your project and press the Run  button.</p>													
<p>Select the <b>testFLPServiceMockServer.html</b> to run the application with the mock data we prepared in BUILD.</p> <p>Press <b>OK</b>.</p>	 <table border="1"> <thead> <tr> <th>File Name</th><th>File Path</th></tr> </thead> <tbody> <tr> <td>testFLPService.html</td><td>/IE_workshop/webapp/test/testFLPS...</td></tr> <tr> <td><b>testFLPServiceMockServer.html</b></td><td>/IE_workshop/webapp/test/testFLPS...</td></tr> <tr> <td> </td><td> </td></tr> <tr> <td> </td><td> </td></tr> <tr> <td> </td><td> </td></tr> </tbody> </table>	File Name	File Path	testFLPService.html	/IE_workshop/webapp/test/testFLPS...	<b>testFLPServiceMockServer.html</b>	/IE_workshop/webapp/test/testFLPS...						
File Name	File Path												
testFLPService.html	/IE_workshop/webapp/test/testFLPS...												
<b>testFLPServiceMockServer.html</b>	/IE_workshop/webapp/test/testFLPS...												
<p>A new tab will be open and show an SAP Fiori launchpad.</p> <p><b>Select the tile on the launchpad that corresponds to the name your BUILD prototype application.</b></p>													

Explanation	Screenshot																				
Your application will now open and show the views you designed with BUILD.	<div><div><div><div><div></div><div>&lt;</div><div></div><div></div></div><div>B1 Exercise Nice ▾</div></div><div>Shoe Store</div><div>23 All Items50 Matching Items</div><div>Items (3) <div>↕</div> <div>🔍</div></div><table><tr><th>Item Code</th><th>Description</th><th>Quantity</th><th>Price</th><th>Currency</th></tr><tr><td>A00001</td><td>J.B. Officeprint 1420</td><td>197</td><td>200</td><td>GBP</td></tr><tr><td>A00002</td><td>J.B. Officeprint 1111</td><td>391</td><td>100</td><td>GBP</td></tr><tr><td>B0001</td><td>B0001</td><td>0</td><td>0</td><td></td></tr></table></div></div>	Item Code	Description	Quantity	Price	Currency	A00001	J.B. Officeprint 1420	197	200	GBP	A00002	J.B. Officeprint 1111	391	100	GBP	B0001	B0001	0	0	
Item Code	Description	Quantity	Price	Currency																	
A00001	J.B. Officeprint 1420	197	200	GBP																	
A00002	J.B. Officeprint 1111	391	100	GBP																	
B0001	B0001	0	0																		

### iii. Create a destination pointing to your backend server

On the SAP Community [From SAP API Business Hub to your SAP Business One system](#) blog dedicated to SAP API Business Hub it is explained how to create a destination in SAP Cloud Platform pointing to your SAP Business One backed server. Please check step number 1 of this blog to learn how to create a destination.

#### Destination Configuration

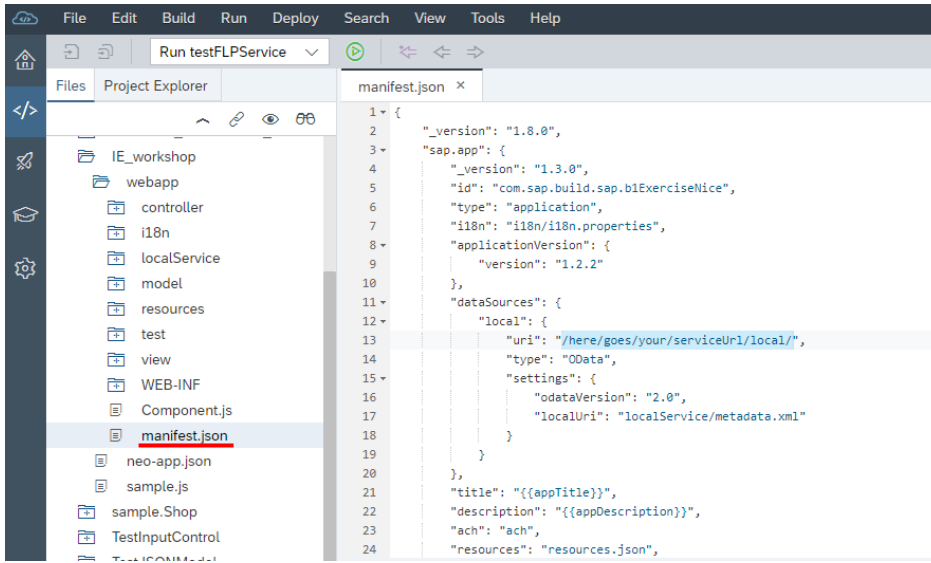

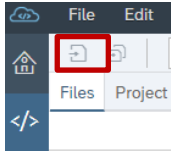
<p><b>*Name:</b> B1SL_workshopNice</p> <p><b>Type:</b> HTTP</p> <p><b>Description:</b> B1SL_workshopNice</p> <p><b>*URL:</b> https://[redacted]50000</p> <p><b>Proxy Type:</b> Internet</p> <p><b>Authentication:</b> BasicAuthentication</p> <p><b>User:</b> {"UserName":"manager","CompanyDB":"SBOD</p> <p><b>Password:</b> [redacted]</p>	<p><b>Additional Properties</b></p> <table border="1"> <tr> <td>TrustAll</td> <td>true</td> <td></td> </tr> <tr> <td>WebIDEEnabled</td> <td>true</td> <td></td> </tr> <tr> <td>WebIDEUsage</td> <td>ui5_execute</td> <td></td> </tr> </table>	TrustAll	true		WebIDEEnabled	true		WebIDEUsage	ui5_execute	
TrustAll	true									
WebIDEEnabled	true									
WebIDEUsage	ui5_execute									

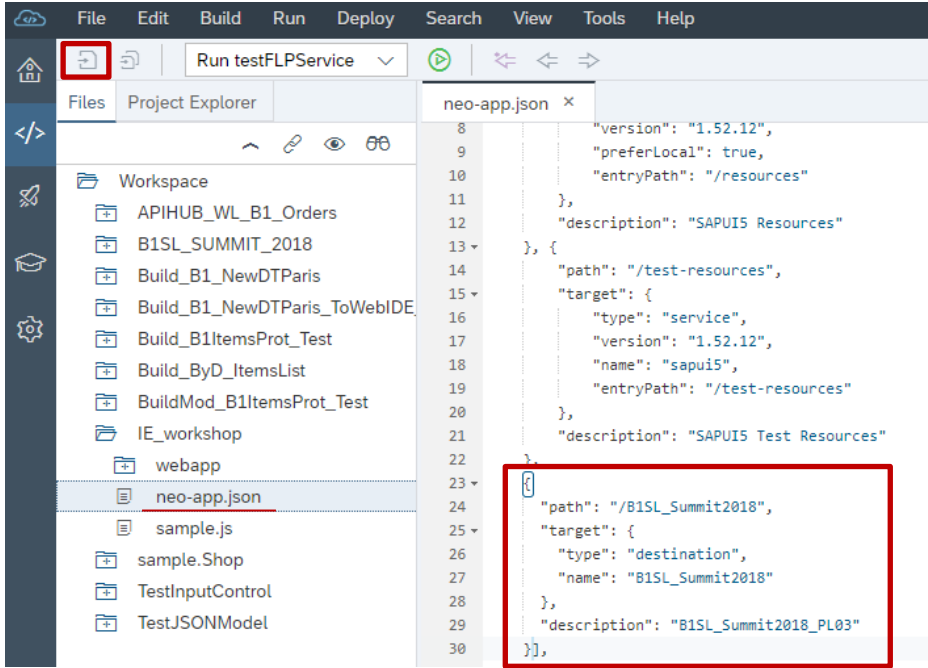
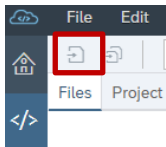
[Edit](#)
[Clone](#)
[Export](#)
[Delete](#)
[Check Connection](#)

### iv. Connect to your real B1 backend server

We have imported the BUILD prototype into a WebIDE SAP Fiori project, but we are still not connected to a real backend server. This section will show you how to modify the SAP Fiori project to connect to your real B1 backend server.



Explanation	Screenshot
<p>In SAP Web IDE workspace, expand your project.</p> <p>In the <b>webapp</b> folder, <b>open the manifest.json</b> file with the code editor.</p>	
<p>In the <b>manifest.json</b> file.</p> <p>Replace the <b>uri</b> property value under <b>dataSources</b> section with your backend OData service path.</p> <p>The uri is built from your <b>destination name</b> (in my case /B1SL_Summit2018) plus the <b>root Service Layer path for OData v4</b> (/b1s/v2).</p>	
<p>Press <b>Save</b> button.</p>	

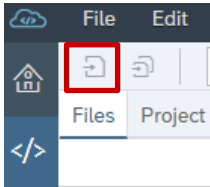
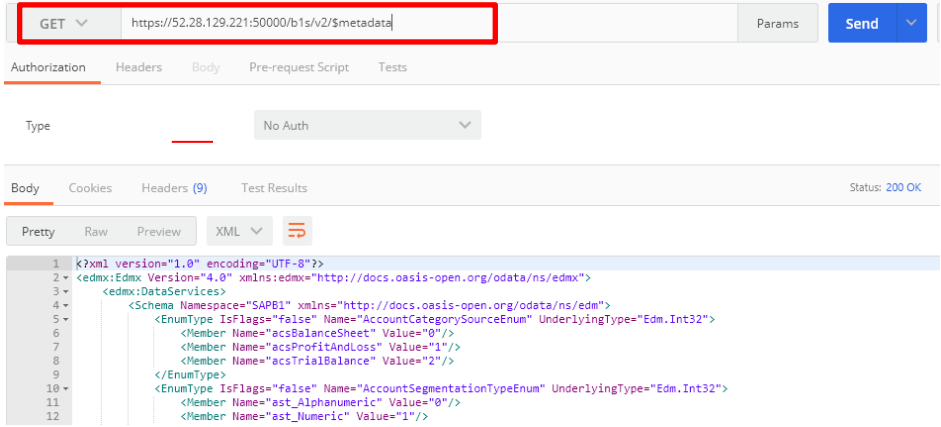
Explanation	Screenshot
<p>Open <b>neo-app.json</b> file.</p> <p>Add your <b>backend destination</b> entry to fetch data.</p> <pre> {   "path": "/yourdest",   "target": {     "type": "destination",     "name": "yourdest"   },   "description": "yourdesc" } </pre>	 <p>The screenshot shows the SAP WebIDE interface. The Project Explorer on the left lists the workspace contents, including 'neo-app.json'. The editor on the right displays the content of 'neo-app.json'. A new destination entry is added at the bottom of the file, highlighted with a red box:</p> <pre> {   "path": "/B1SL_Summit2018",   "target": {     "type": "destination",     "name": "B1SL_Summit2018"   },   "description": "B1SL_Summit2018_PL03" }], </pre>
<p>Press <b>Save</b> button.</p>	 <p>The screenshot shows the SAP WebIDE File menu. The 'Save' button, represented by a floppy disk icon, is highlighted with a red box.</p>

#### v. Extra SAP Business One backend configuration steps

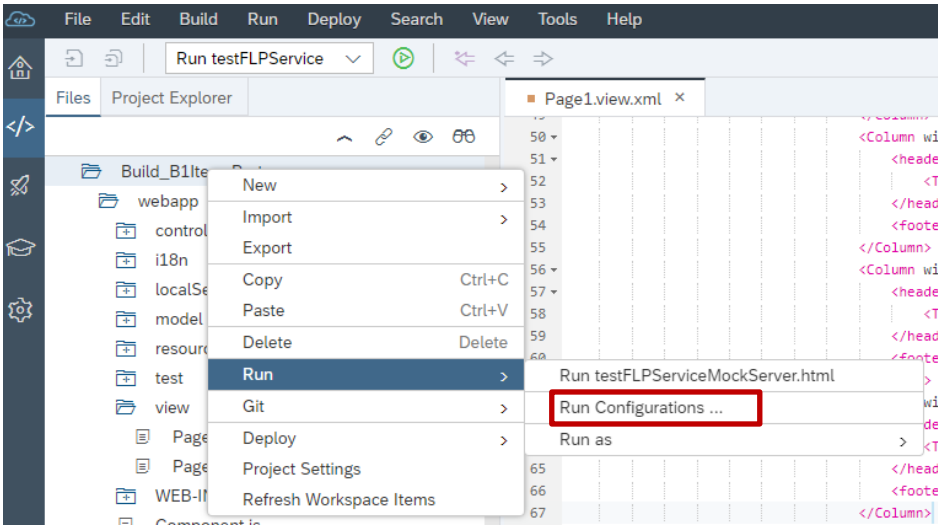
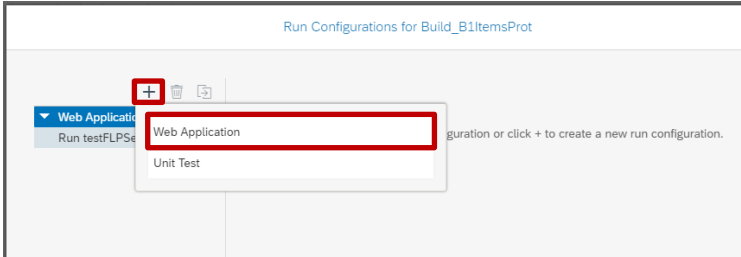
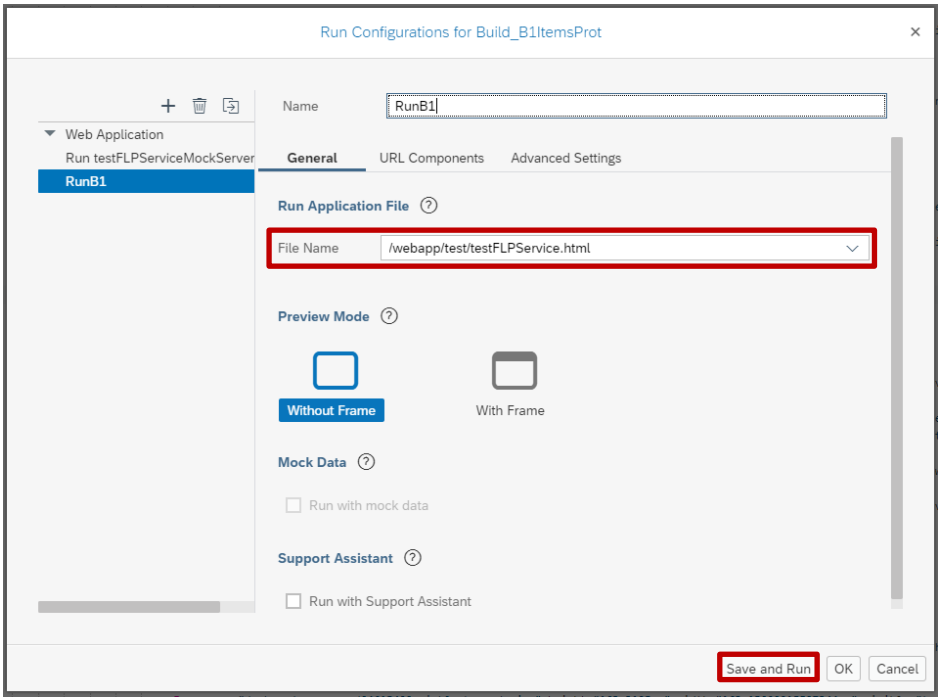
As at the time we have created this document SAP BUILD doesn't support yet OData v4 and SAP Business One Service Layer APIs are based on OData v4, to design our SAP Business One Build prototype we had to use a custom OData model in SAP Build to design our prototype. Therefore, the WebIDE project will not directly run after the changes done in previous steps but some extra steps will be required.

As SAP WebIDE supports OData v4 we can now replace the custom OData model we designed in SAP Build by the real SAP Business One Service Layer OData model to get SAP Business One data from our backend.

Explanation	Screenshot
<p>Open the <b>manifest.json</b> file.</p> <p>Change the “<b>settings</b>” “<b>odataVersion</b>” to <b>4.0</b>.</p>	 <pre> 1 { 2   "_version": "1.8.0", 3   "sap.app": { 4     "_version": "1.3.0", 5     "id": "com.sap.build.sap.b1ExerciseNice", 6     "type": "application", 7     "i18n": "i18n/i18n.properties", 8     "applicationVersion": { 9       "version": "1.2.2" 10    }, 11    "dataSources": { 12      "local": { 13        "uri": "/B1SL_Summit2018/b1s/v2/", 14        "type": "OData", 15        "settings": { 16          "odataVersion": "4.0", 17          "localUri": "localService/metadata.xml" 18        } 19      } 20    }, </pre>
<p>Search <b>models</b> element inside sap.ui5</p>	 <pre> 64 }, 65 "models": { 66   "i18n": { 67     "type": "sap.ui.model.resource.ResourceModel", 68     "uri": "i18n/i18n.properties" 69   }, 70   "": { 71     "dataSource": "local", 72     "type": "sap.ui.model.odata.v2.ODataModel", 73     "settings": { 74       "loadMetadataAsync": false, 75       "json": true, 76       "bJSON": true, 77       "defaultBindingMode": "TwoWay", 78       "defaultCountMode": "Inline", 79       "useBatch": true, 80       "refreshAfterChange": false, 81       "disableHeadRequestForToken": true 82     } 83   } 84 }, </pre>

Explanation	Screenshot
<p>Replace the <b>type</b> of the model with empty name by <b>sap.ui.model.odata.v4.ODataModel</b>.</p> <p>Change the <b>settings</b> (copy the values here below) and add <b>preload</b> property with value <b>true</b>.</p> <p>Pay attention you keep the <b>dataSource</b> value unchanged as it matches the <b>dataSource</b> value defined at the beginning of the file.</p> <pre> "settings": {   "operationMode": "Server",    "synchronizationMode": "None",    "groupId": "\$direct" }, "preload": true </pre>	<pre> "models": {   "i18n": {     "type": "sap.ui.model.resource.ResourceModel",     "uri": "i18n/i18n.properties"   },   "": {     "dataSource": "local",     "type": "sap.ui.model.odata.v4.ODataModel",     "settings": {       "operationMode": "Server",       "synchronizationMode": "None",       "groupId": "\$direct"     },     "preload": true   } }, </pre>
Press the <b>Save</b> button.	
<p>Retrieve the metadata file from SAP Business One Service Layer via <b>Postman</b> with the <b>GET</b> request <b><u><a href="https://your_b1sl_server:50000/b1s/v2/\$metadata">https://your_b1sl_server:50000/b1s/v2/\$metadata</a></u></b>.</p> <p><b>Save</b> the response as a file named <b>metadata.xml</b>.</p>	

Explanation	Screenshot
<p>Replace the <b>localService/metadata.xml</b> file imported from BUILD by the SAP Business One Service Layer metadata file saved in the previous step.</p> <p>To avoid conflicts as the Build metadata.xml file is already there you can rename the existing file as <b>build_metadata.xml</b>.</p>	 <p>The screenshot shows the SAP Studio interface. In the Project Explorer on the left, the file 'metadata.xml' under the 'localService' folder is highlighted. The main editor displays the XML content of 'metadata.xml', which is an EDMX file for SAP Business One services, including schema definitions for various data services.</p>
<p>Open the <b>Page1.view.xml</b> file, search for <b>ItemsSet</b> and replace it by <b>Items</b>.</p> <p>In the model we created in BUILD entities have the suffix Set, while in SAP Business One Service Layer we don't have it, we need to fix it to be able to directly connect to Service Layer.</p>	 <p>The screenshot shows the SAP Studio interface. In the Project Explorer on the left, the file 'Page1.view.xml' is selected. The main editor displays the XML content of 'Page1.view.xml'. The XML shows a table binding to 'ItemsSet' in the 'items' property, which is highlighted in blue. Below the table, there is an 'infoToolbar' section.</p>
<p>Open <b>Component.js</b> file.</p> <p>Replace <b>ItemsSet</b> by <b>Items</b> in the navigationWithContext definition.</p>	 <p>The screenshot shows the SAP Studio interface. In the Project Explorer on the left, the file 'Component.js' is selected. The main editor displays the JavaScript code for 'Component.js'. The code defines a navigationWithContext object, and the 'Items' property is highlighted with a red box.</p>

Explanation	Screenshot
<p>To run the application this time connecting to your real B1 backend right click on your project and choose <b>Run -&gt; Run Configurations...</b></p>	
<p>Press <b>+</b>.</p> <p>Select <b>Web Application</b>.</p>	
<p>Give a <b>Name</b> to the new configuration.</p> <p>Select <b>testFLPService.html</b> as File Name.</p> <p>Press <b>Save and Run</b>.</p>	

Explanation	Screenshot																																			
Now the data shown comes from B1 and not anymore from the Build sample data.	<div><div>B1 Exercise Nice</div><div>Shoe Store</div><div>23 All Items50 Matching Items</div><div>Items (3)<div>↑↓⌵⌶⚙</div><table><tr><th>Item Code</th><th>Description</th><th>Quantity</th><th>Price</th><th>Currency</th></tr><tr><td>I00002</td><td>Blu-Ray DL Disc 10-Pack</td><td>1,537</td><td></td><td>&gt;</td></tr><tr><td>R00001</td><td>Printer Paper A4 White</td><td>33,703</td><td></td><td>&gt;</td></tr><tr><td>D00002</td><td>Portable Hard Disk 2TB</td><td>806</td><td></td><td>&gt;</td></tr><tr><td>C00006</td><td>Gigabit Network Card</td><td>1,483</td><td></td><td>&gt;</td></tr><tr><td>P20001</td><td>4GB Memory Server</td><td>547</td><td></td><td>&gt;</td></tr><tr><td>A00005</td><td>Rainbow Color Printer 7.5</td><td>1,564</td><td></td><td>&gt;</td></tr></table></div></div>	Item Code	Description	Quantity	Price	Currency	I00002	Blu-Ray DL Disc 10-Pack	1,537		>	R00001	Printer Paper A4 White	33,703		>	D00002	Portable Hard Disk 2TB	806		>	C00006	Gigabit Network Card	1,483		>	P20001	4GB Memory Server	547		>	A00005	Rainbow Color Printer 7.5	1,564		>
Item Code	Description	Quantity	Price	Currency																																
I00002	Blu-Ray DL Disc 10-Pack	1,537		>																																
R00001	Printer Paper A4 White	33,703		>																																
D00002	Portable Hard Disk 2TB	806		>																																
C00006	Gigabit Network Card	1,483		>																																
P20001	4GB Memory Server	547		>																																
A00005	Rainbow Color Printer 7.5	1,564		>																																

You can also check the details page containing the different prices depending on their price list for a specific Item by clicking on one of the rows.	<div><div>Untitled Prototype</div><div>Title Subtitle</div><div>SUB SECTION TITLE<div>SUB SECTION TITLE</div>SUB SECTION TITLE</div><div>Items (3)<div>↑↓⌵⌶⚙</div><div>Some random text</div><table><tr><th>Price List No</th><th>Price</th></tr><tr><td>1</td><td>2 GBP</td></tr><tr><td>2</td><td>1 GBP</td></tr><tr><td>3</td><td>1.5 GBP</td></tr><tr><td>4</td><td>2.5 GBP</td></tr><tr><td>5</td><td>3 GBP</td></tr><tr><td>6</td><td>3.5 GBP</td></tr></table></div></div>	Price List No	Price	1	2 GBP	2	1 GBP	3	1.5 GBP	4	2.5 GBP	5	3 GBP	6	3.5 GBP
Price List No	Price														
1	2 GBP														
2	1 GBP														
3	1.5 GBP														
4	2.5 GBP														
5	3 GBP														
6	3.5 GBP														

Congratulations! You have imported a Build prototype to your WebIDE development environment and connected to your real SAP Business One backend server.

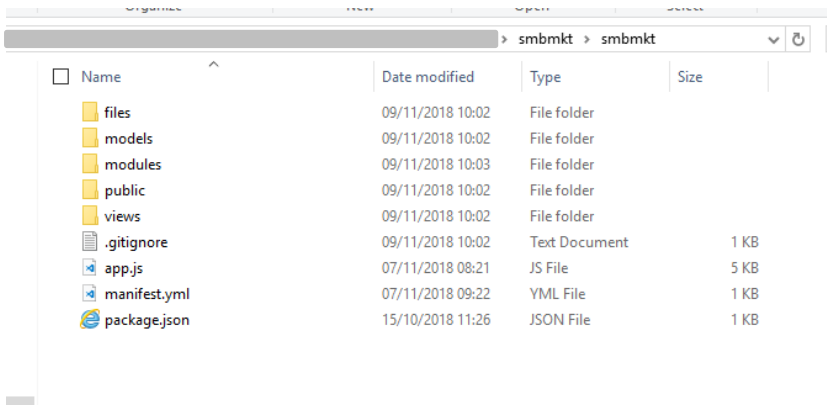
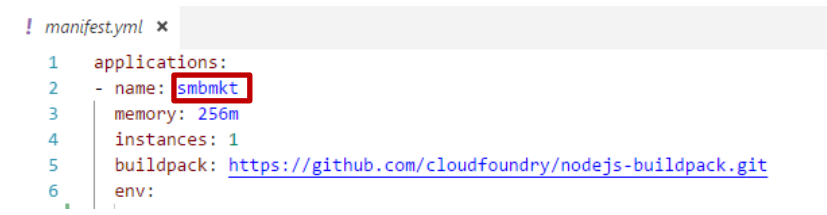
### STEP 3: CLONE A NODEJS APP

In this step we are going to deploy the backend of our application.

The application we are going to deploy is based on the SMB Marketplace proof of concept we shared in the [Digital Transformation for SMBs – the Intelligent Enterprise blog](#).

It will contain the business logic required to call SAP Leonardo services and get Item details from SAP Business One and SAP Business ByDesign erps.

The application is written in NodeJS and the source code is available on GitHub.

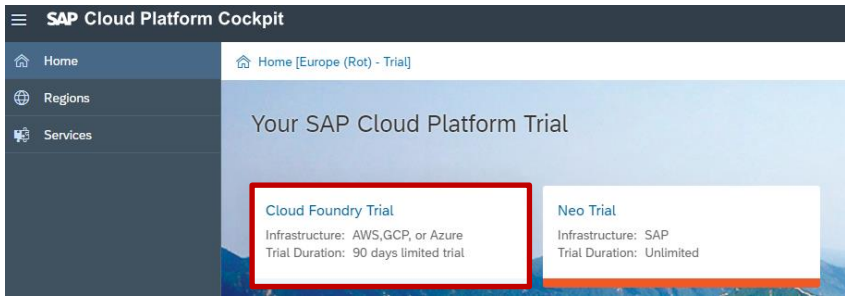
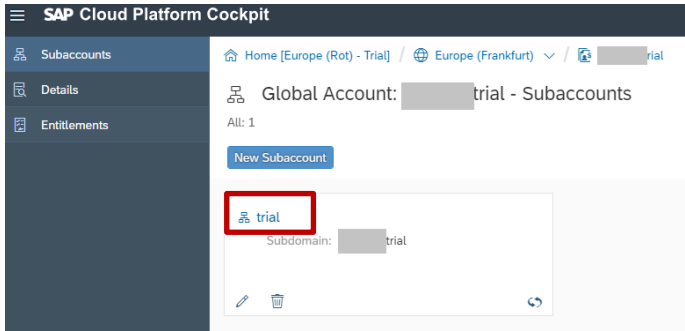
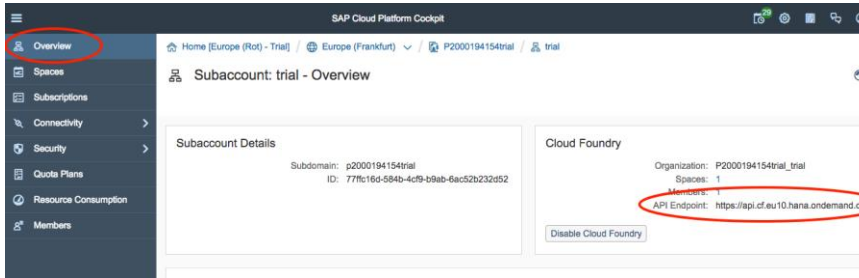
Explanation	Screenshot
<p>Once <b>git</b> is installed (according to the pre requisites), open your system <b>terminal</b> (cmd, bash..)</p> <p><b>Navigate</b> to a specific folder where you will download the sample application.</p> <p><b>Pay attention</b> what folder is it, we will access it later.</p>	
<p>Execute the following command to clone our solution:</p> <pre>\$ git clone -b CF_exercise https://github.com/BISA/smbmkt.git</pre>	
<p>Go to the directory smbmk/smbmk, this is the folder containing the code we will push into</p>	
<p>You can change the name of the app in <b>manifest.yml</b> file and set a unique name for your application, e.g.: <b>smbmk&lt;your Initials&gt;</b>.</p> <p>It is not a mandatory operation as we can generate a random url for our application to avoid conflicts with other accounts running the same app name, it will be shown in the next step.</p>	

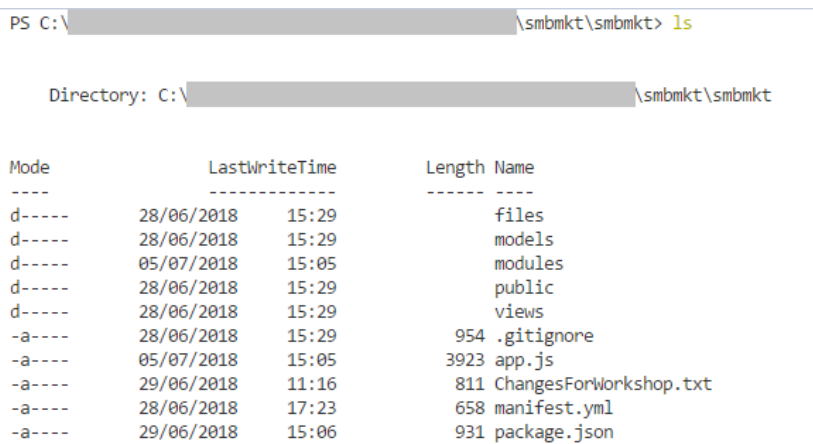



## STEP 4: DEPLOY THE NODEJS APP INTO SAP CLOUD FOUNDRY

In this step, we are going to deploy our SMB Marketplace app to SAP Cloud Platform Cloud Foundry.

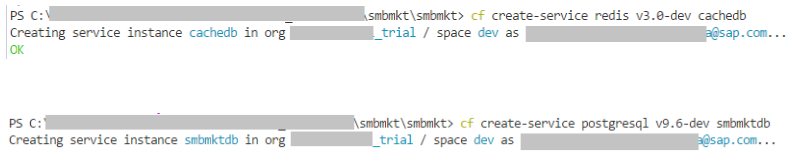
### i. SAP Cloud Platform Cloud Foundry Environment

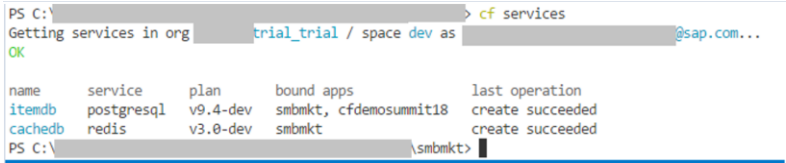
Go to your Cloud Foundry account.	 <p>The screenshot shows the SAP Cloud Platform Cockpit interface. The left sidebar has a menu with 'Home', 'Regions', and 'Services'. The main content area is titled 'Your SAP Cloud Platform Trial' and contains two cards: 'Cloud Foundry Trial' (highlighted with a red box) and 'Neo Trial'. The 'Cloud Foundry Trial' card lists 'Infrastructure: AWS, GCP, or Azure' and 'Trial Duration: 90 days limited trial'.</p>
Select your trial subaccount. Click on the <b>trial</b> link.	 <p>The screenshot shows the 'Subaccounts' page in the SAP Cloud Platform Cockpit. The left sidebar has a menu with 'Subaccounts', 'Details', and 'Entitlements'. The main content area shows 'Global Account: [redacted] trial - Subaccounts' and 'All: 1'. A 'New Subaccount' button is visible. Below it, a table lists subaccounts, with the first one 'trial' highlighted by a red box. The subdomain for 'trial' is '[redacted]trial'.</p>
Open the <b>Overview</b> option in the menu  <b>Select and copy your API Endpoint. E.g.</b>  <a href="https://api.cf.eu10.hana.ondemand.com">https://api.cf.eu10.hana.ondemand.com</a>	 <p>The screenshot shows the 'Overview' page for the 'trial' subaccount in the SAP Cloud Platform Cockpit. The left sidebar has a menu with 'Overview' (highlighted with a red circle), 'Spaces', 'Subscriptions', 'Connectivity', 'Security', 'Quota Plans', 'Resource Consumption', and 'Members'. The main content area is titled 'Subaccount: trial - Overview' and contains two panels: 'Subaccount Details' and 'Cloud Foundry'. The 'Cloud Foundry' panel shows 'Organization: P2000194154trial_trial', 'Spaces: 1', 'Members: 1', and 'API Endpoint: https://api.cf.eu10.hana.ondemand.com' (highlighted with a red circle). A 'Disable Cloud Foundry' button is also visible.</p>

<p>With the CLI installed (according to the pre-requisites), open your system <b>terminal</b> and navigate to the folder of the backend app cloned on STEP 3 of this guide</p>	 <pre> PS C:\[redacted]\smbmktd&gt; ls  Directory: C:\[redacted]\smbmktd  Mode                LastWriteTime         Length Name ----                - d-----          28/06/2018      15:29         files d-----          28/06/2018      15:29        models d-----          05/07/2018      15:05       modules d-----          28/06/2018      15:29       public d-----          28/06/2018      15:29       views -a-----          28/06/2018      15:29       954 .gitignore -a-----          05/07/2018      15:05      3923 app.js -a-----          29/06/2018      11:16     811 ChangesForWorkshop.txt -a-----          28/06/2018      17:23     658 manifest.yml -a-----          29/06/2018      15:06     931 package.json </pre>
<p>From that folder, login to Cloud foundry using the command</p> <pre>cf login -a &lt;API ENDPOINT&gt;</pre> <p>e.g.</p> <pre>\$ cf login -a api.cf.eu10.hana.ondemand.com</pre> <p>When prompted provide your SAP Cloud Platform email and password</p>	 <pre> PS C:\[redacted]\smbmktd&gt; cf login API endpoint: https://api.cf.eu10.hana.ondemand.com  Email&gt; [redacted]@sap.com  Password&gt; Authenticating... OK  Targeted org [redacted]_trial  Targeted space dev  API endpoint: https://api.cf.eu10.hana.ondemand.com (API version: 2.114.0) User: [redacted]@sap.com Org: [redacted]_trial Space: dev </pre>

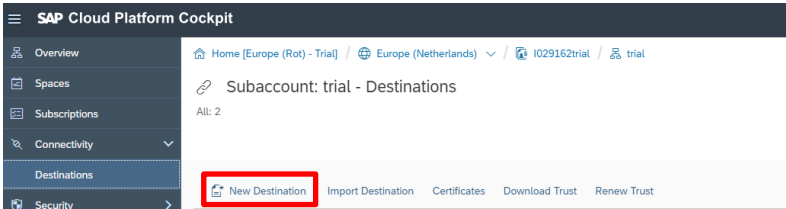
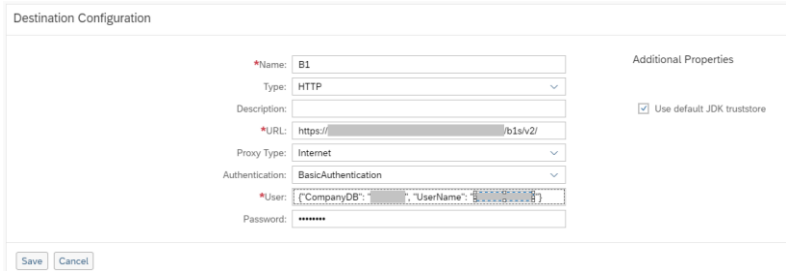
## ii. Create the backing services

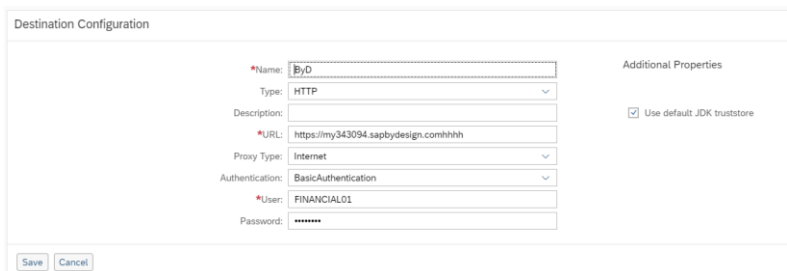

This app uses 2 [backing services](#) from SAP Cloud Platform. [Redis](#) for storing B1 Service Layer Sessions ID in cache and [PostgreSQL](#) to store [SAP Leonardo Feature Extraction Vectors](#). Here are the steps to create them:

Explanation	Screenshot
<p>Using the command terminal, navigate to the smbmktd directory.</p> <p>Execute the following commands to create the Redis and PostgreSQL services:</p> <pre>cf create-service redis v3.0-dev cachedb</pre> <pre>cf create-service postgresql v9.6-dev itemsdb</pre>	 <pre> PS C:\[redacted]\smbmktd&gt; cf create-service redis v3.0-dev cachedb Creating service instance cachedb in org [redacted]_trial / space dev as [redacted]@sap.com... OK  PS C:\[redacted]\smbmktd&gt; cf create-service postgresql v9.6-dev itemsdb Creating service instance itemsdb in org [redacted]_trial / space dev as [redacted]@sap.com... </pre>

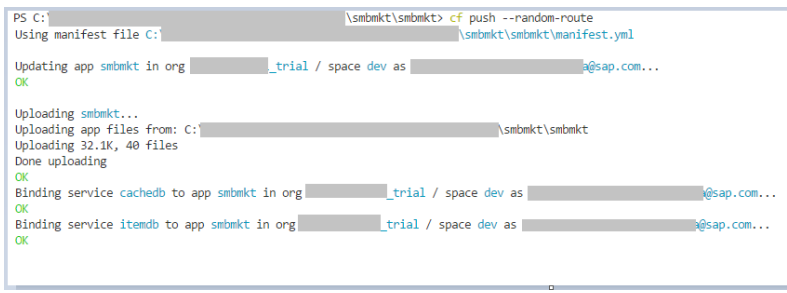
Explanation	Screenshot
<p>PS: When using a trial account some limitations apply. If you already had a postgresql or redis service you will not be able to create a second one, just reuse the one you have or delete your old one.</p>	
<p>You can check which services are active and the bound apps with the command:</p> <pre>cf services</pre> <p>(your services might not be bound to an app if just created now)</p>	 <pre>PS C:\&gt; cf services Getting services in org trial_trial / space dev as [redacted]@sap.com... OK name      service    plan      bound apps      last operation itemdb    postgresql v9.4-dev   smbkt, cfdemo   create succeeded cachedb   redis      v3.0-dev   smbkt           create succeeded PS C:\&gt;</pre>

### iii. Create the required destinations

Explanation	Screenshot
<p>To connect to our B1 and/or ByD ERP server we will use Cloud Foundry destinations.</p>	
<p>Go to your Cloud Foundry account. Select your trial subaccount.</p> <p>Select the <b>Destinations</b> menu.</p> <p>Click on the <b>"New Destination"</b> option.</p>	
<p>Create your B1 Destination.</p> <p>Enter a name for your B1 Destination. If you enter a different name than B1 pay attention to following steps as you will need to replace the Name in all references.</p> <p>Enter Type HTTP and BasicAuthentication.</p> <p>Enter your B1 ERP URL and credentials as well as any specific configuration you might require.</p> <p>Press <b>Save</b>.</p> <p>Note: Even if you only work with ByD please create B1 destination to avoid exceptions in the sample code.</p>	

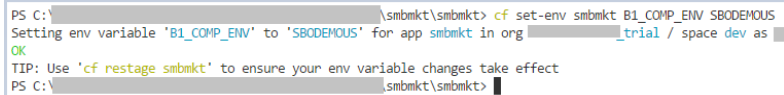
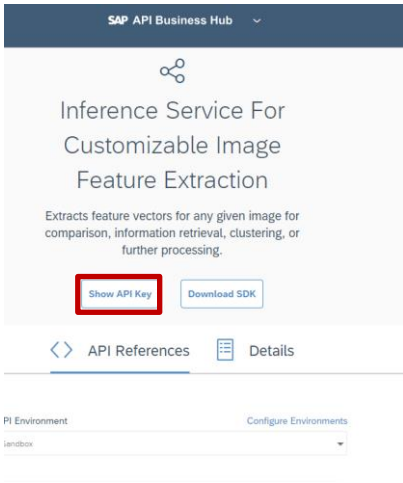
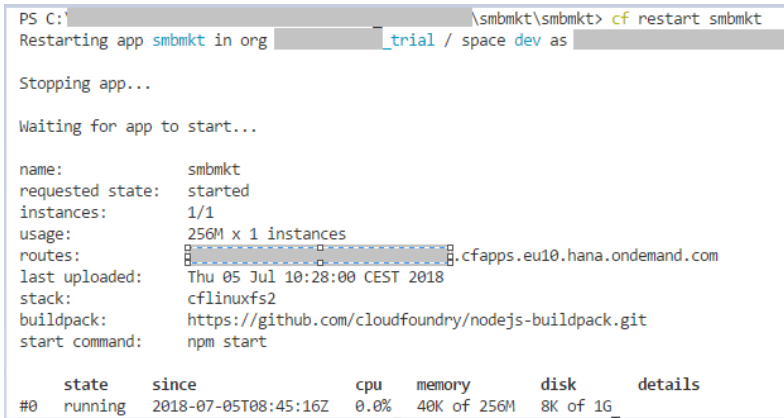
Explanation	Screenshot
<p>Create your ByD Destination.</p> <p>Enter a name for your ByD Destination. If you enter a different name than ByD pay attention to following steps as you will need to replace the Name in all references.</p> <p>Enter Type HTTP and BasicAuthentication.</p> <p>Enter your ByD ERP URL and credentials as well as any specific configuration you might require.</p> <p>Press <b>Save</b>.</p> <p>Note: Even if you only work with B1 please create ByD destination to avoid exceptions in the sample code.</p>	
<p>Open the <code>modules/dest/dest-app.json</code> file.</p> <p>Check the parameters specified for both routes are correctly pointing to your Destinations and their <code>entryPath</code> is correct.</p> <p>Check the blog <a href="#">Call SAP Cloud Platform destinations from your Node.js application</a> for more details on Destinations.</p>	

#### iv. Deploy the smbmktp app

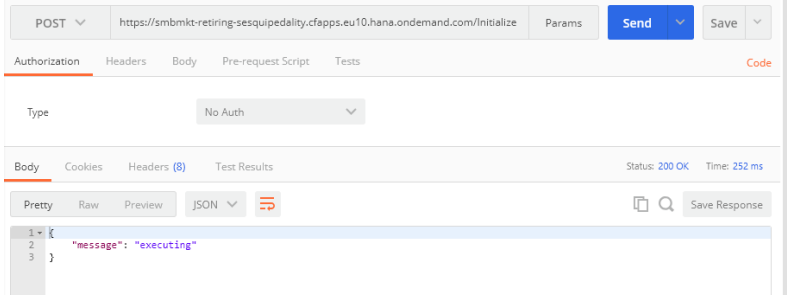
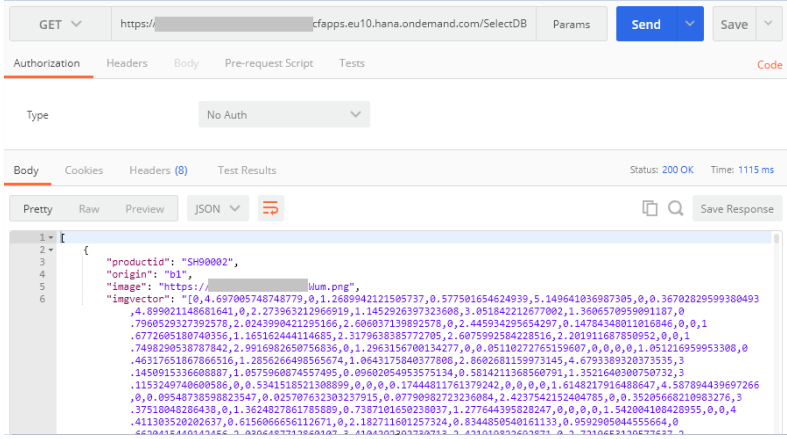
Explanation	Screenshot
<p>This app has 2 microservices (bot and smbmktp) that can be deployed at once or separately. Their specifications are detailed in the <a href="#">manifest.yml</a>.</p> <p>In this exercise we will only work with the smbmktp microservice as the other service is the one related to</p>	



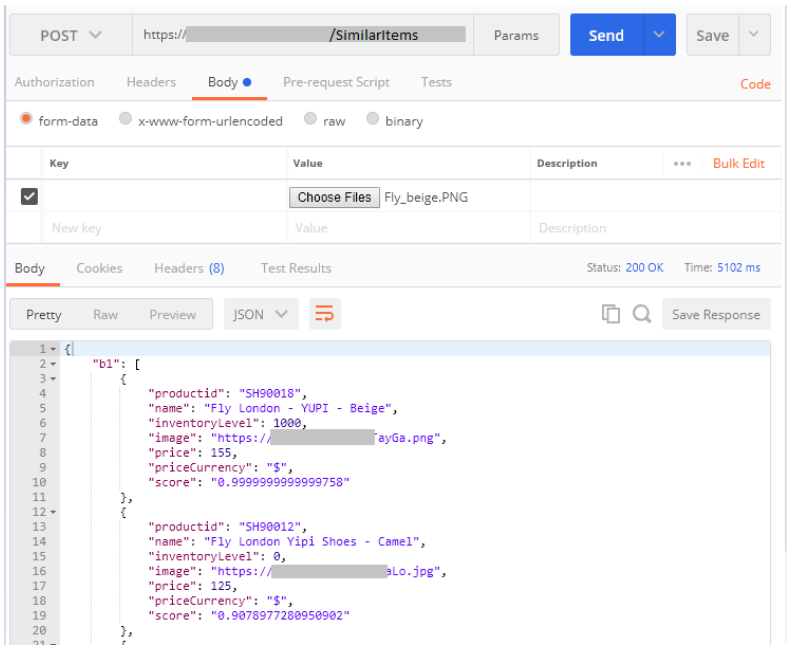
## ii. Configure the SMB Mktplace backend

Explanation	Screenshot														
<p><b>Set the following <u>Environment Variables</u></b> so the app can work properly.</p> <p>The ones marked in red are mandatory (B1 or ByD variable depending on your backend).</p> <p>Please note that you need at least one ERP system from both to be able to retrieve items data.</p> <p>If you don't have a B1 or ByD system available, you can then skip its corresponding environment variable.</p>	<p><b>B1_DEFAULT_BP</b>: &lt;A Business Partner Code for the B1 Sales Order&gt; <b>BYD_DEFAULT_BP</b>: &lt;A Business Partner Code for the ByD Sales Order&gt; <b>FILE_SEP</b>: _-_ <b>LEO_API_KEY</b>: &lt;SAP Leonardo API Key&gt; <b>TEMP_DIR</b>: files/tmp <b>VECTOR_DIR</b>: files/vectors</p>														
<p>Set one by one the environment variables with the command:</p> <pre>cf set-env smbmk B1_COMP_ENV SBODEMOUS</pre>															
<p>To obtain the SAP Leonardo API Key please open the following link <a href="#">SAP Leonardo Feature Extraction Vectors</a> and press the button “Show API Key”.</p>															
<p>Restart your application so it can get the new environment variables with the following command:</p> <pre>cf restart smbmk</pre>	 <table><thead><tr><th></th><th>state</th><th>since</th><th>cpu</th><th>memory</th><th>disk</th><th>details</th></tr></thead><tbody><tr><td>#0</td><td>running</td><td>2018-07-05T08:45:16Z</td><td>0.0%</td><td>40K of 256M</td><td>8K of 1G</td><td></td></tr></tbody></table>		state	since	cpu	memory	disk	details	#0	running	2018-07-05T08:45:16Z	0.0%	40K of 256M	8K of 1G	
	state	since	cpu	memory	disk	details									
#0	running	2018-07-05T08:45:16Z	0.0%	40K of 256M	8K of 1G										

### iii. Initialize the SMB Mktplace backend

Explanation	Screenshot
<p>To initialize the Postgresql database with the existing items from B1 and ByDesign as well as the vectors for each item please call the following API with Postman for example:</p> <p>POST <u>&lt;your backend url&gt;/Initialize</u></p>	
<p>After initialization you can check the Postgresql items table content with the following API:</p> <p>GET <u>&lt;your backend url&gt;/SelectDB</u></p> <p>If the Initialize command runs successfully an entry should be available for each one of your items containing the productid, origin, image and imgVector properties.</p>	

### iv. Test the SMB Mktplace backend /SimilarItems API

Explanation	Screenshot
<p>With Postman call the /SimilarItems API:</p> <p>POST <u>&lt;your backend url&gt;/SimilarItem</u> s</p> <p>In the <b>body</b> select <b>“form-data”</b> and choose a file containing the image of a shoe.</p>	



Congratulations! You have implemented and deployed your first Cloud Foundry application on SAP Cloud Platform!



## STEP 5: CONSUME THE NODEJS APP FROM THE SAP FIORI APP

Until now our SAP Fiori application hasn't been modified and reflects exactly what was designed in BUILD. In this step we are going to modify the tab "Matching Items" to consume the services provided by our NodeJS backend.

### i. Create a destination pointing to your smbmkmt backend

Your destination in your SAP Cloud Platform cockpit -> Connectivity -> Destinations should look like the one here, just replace the URL with your smbmkmt url.

Check the following tutorial [Create a Destination on SAP Cloud Platform](#) to learn more details about destinations.

Destination Configuration

\*Name: smbmkmt\_workshopNice

Type: HTTP

Description: smbmkmt\_workshopNice

\*URL: https://smbmkmt-...cfaf

Proxy Type: Internet

Authentication: NoAuthentication


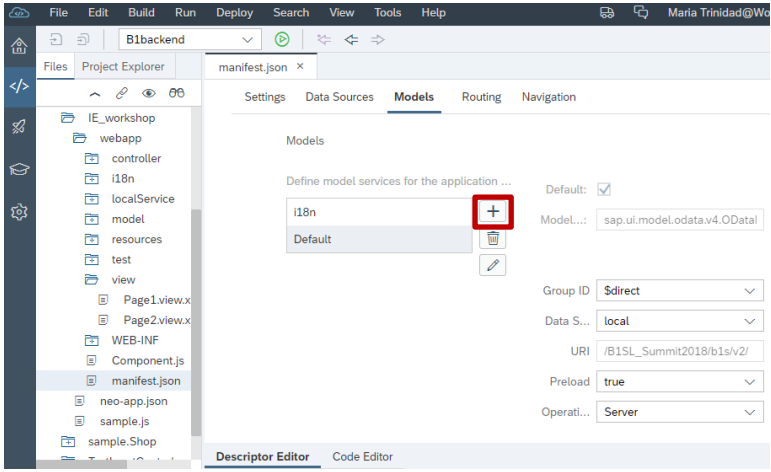
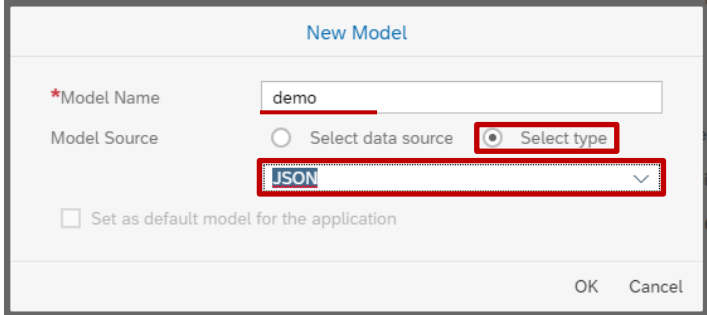
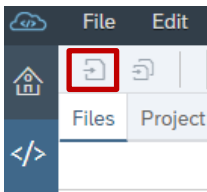
Additional Properties

TrustAll	true	
WebIDEEnabled	true	
WebIDESystem	smbmkmt_workNice	
WebIDEUsage	ui5_execute, odata_gen	

Buttons: Edit, Clone, Export, Delete, Check Connection

Explanation	Screenshot
<p>Open <b>neo-app.json</b> file.</p> <p>Add your <b>backend destination</b> entry to fetch data.</p> <pre>{   "path": "/yourdest",   "target": {     "type": "destination",     "name": "yourdest"   },   "description": "yourdesc" }</pre>	<pre>10      "entryPath": "/resources" 11    }, 12    "description": "SAPUI5 Resources" 13  }, { 14    "path": "/test-resources", 15    "target": { 16      "type": "service", 17      "version": "1.52.12", 18      "name": "sapui5", 19      "entryPath": "/test-resources" 20    }, 21    "description": "SAPUI5 Test Resources" 22  }, { 23    "path": "/B1SL_workshopNice", 24    "target": { 25      "type": "destination", 26      "name": "B1SL_workshopNice" 27    }, 28    "description": "B1SL_workshopNice" 29  }, { 30    "path": "/smbmkmt_workshopNice", 31    "target": { 32      "type": "destination", 33      "name": "smbmkmt_workshopNice" 34    }, 35    "description": "smbmkmt_workshopNice" 36  } 37 }</pre>
<p>Press <b>Save</b> button.</p>	

## ii. Create a new JSON model

Explanation	Screenshot
<p>Open the <b>manifest.json</b> file with the Descriptor Editor.</p> <p>Go to the <b>Models</b> tab.</p> <p>Press the  button to add a new model.</p>	
<p>Enter the <b>Model Name</b> of the model <b>demo</b> if you want to avoid changing the references in the following steps.</p> <p>Choose <b>Select type</b> as Model Source.</p> <p>Select <b>JSON</b> as type.</p> <p>Press <b>OK</b>.</p>	
<p>Press <b>Save</b> button.</p>	

## iii. Change the Image control in the Page1.view.xml file.

Explanation	Screenshot
<p>Open the <b>Page1.view.xml</b> file with the Code Editor.</p> <p>Search for the Image control and replace it with the following code:</p> <pre>&lt;Image id="img" tooltip="image" class="sapUiLargeMargin"</pre>	

Explanation	Screenshot
<pre>n" src="{demo}/fileURL}"/ &gt;</pre>	

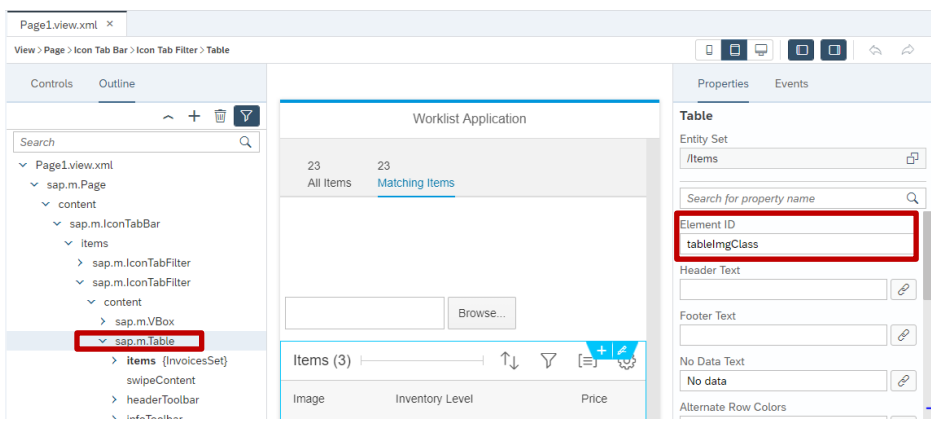
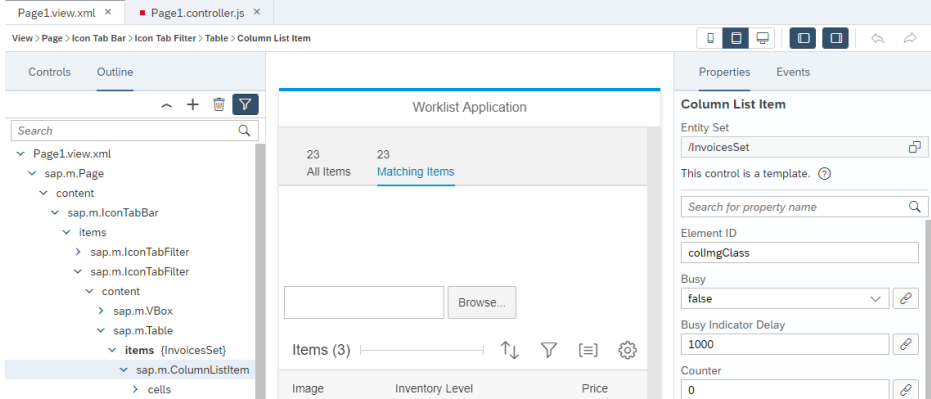
#### iv. Create a FileUploader control.

In BUILD we added a SearchField control as the FileUploader control was not available. We will now replace it with a FileUploader.

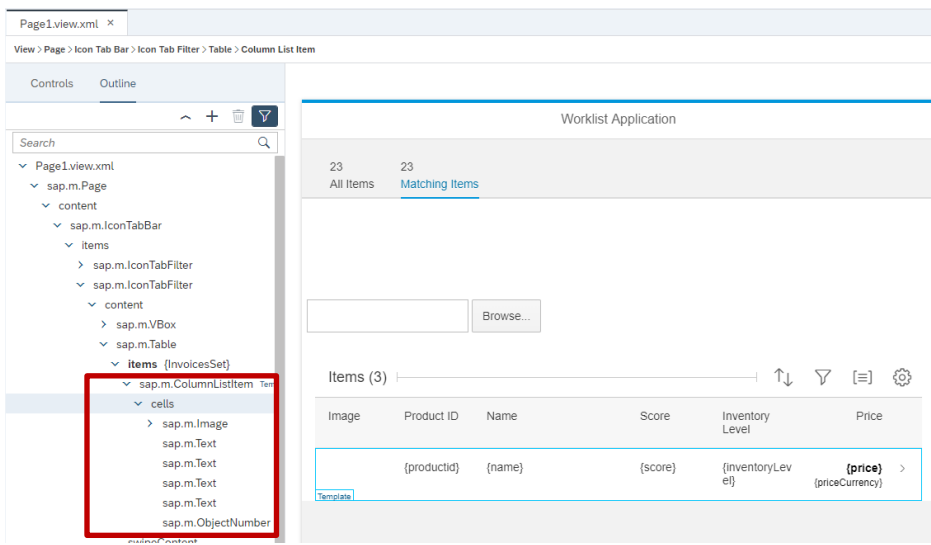
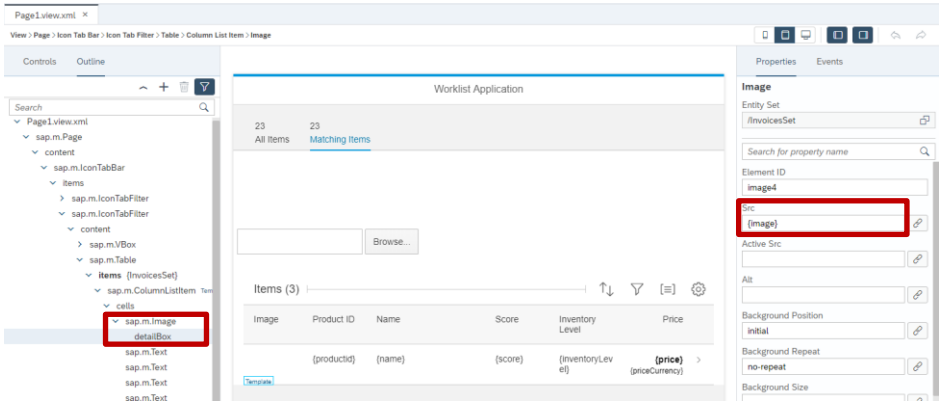
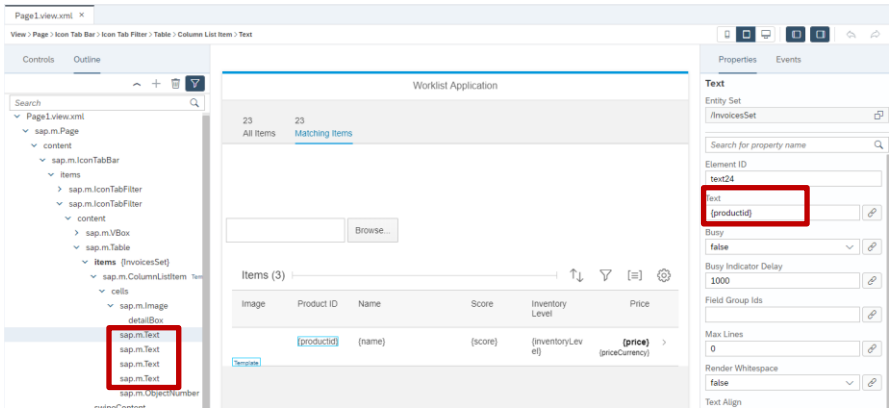
Explanation	Screenshot
<p>Open the <b>Page1.view.xml</b> file with the Code Editor.</p> <p>Search the SearchField control and replace it by the following code.</p> <p>We use the <b>smbmkt</b> destination created in a previous step to get the <b>SimilarItems</b> url.</p> <p><b>Replace <b>smbmkt_destination</b> with your <b>smbmkt</b> destination name.</b></p>	 <pre>&lt;IconTabFilter icon="" iconColor="Default" text="Matching Items" count="50" design="Vertical" showAll="false"   &lt;content&gt;     &lt;Image id="img" tooltip="image" class="sapUiLargeMargin" src="{demo}/fileURL}/&gt;     &lt;u:FileUploader id="fileUploader" name="files" uploadUrl="/smbmkt_workshopNice/SimilarItems" useMultipart="true"       &lt;u:headerParameters&gt;         &lt;u:FileUploaderParameter name="Accept" value="application/json"/&gt;       &lt;/u:headerParameters&gt;     &lt;/u:FileUploader&gt;   &lt;/content&gt;</pre> <pre>&lt;u:FileUploader id="fileUploader" name="files"   uploadUrl="/smbmkt_destination/SimilarItems" useMultipart="true"   sendXHR="true" uploadOnChange="true" tooltip="Upload your file to the   local server" fileType="jpg,png,gif" mimeType="application/x-zip-   compressed,application/zip,application/octet-   stream,image/png,image/jpg,image/jpeg,image/bmp,image/tiff"   change="fileUploadChange" uploadStart="fileUploadStart"   uploadComplete="fileUploadComplete"&gt;   &lt;u:headerParameters&gt;     &lt;u:FileUploaderParameter   name="Accept" value="application/json"/&gt;   &lt;/u:headerParameters&gt; &lt;/u:FileUploader&gt;</pre>
<p>Add the prefix <b>xmlns:u="sap.ui.unified"</b>, required by the FileUploader control, at the beginning of the Page1.view.xml file.</p> <p>Press <b>Save</b> button.</p>	 <pre>&lt;mvc:View xmlns:mvc="sap.ui.core.mvc" xmlns:u="sap.ui.unified" controllerName="com.sap.bui:   xmlns:layout="sap.ui.layout"&gt;   &lt;Page showHeader="true" title="Shoe Store" showFooter="true" showNavButton="false"&gt;     &lt;content&gt;</pre>

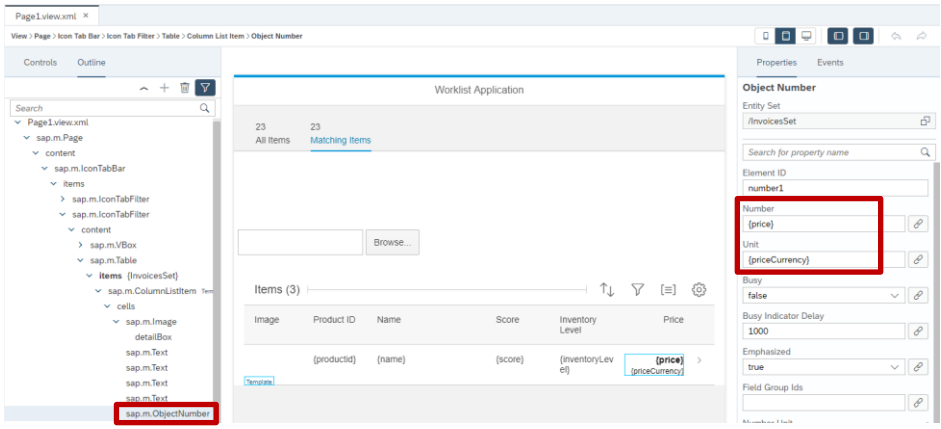
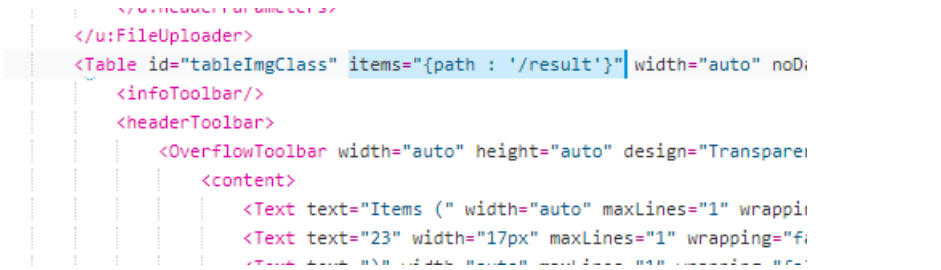
#### v. Bind the Matching Items Table to our backend properties

Let's define first the IDs of our Table and ColumnListItem controls, we will need them to further bind them to our backend response.

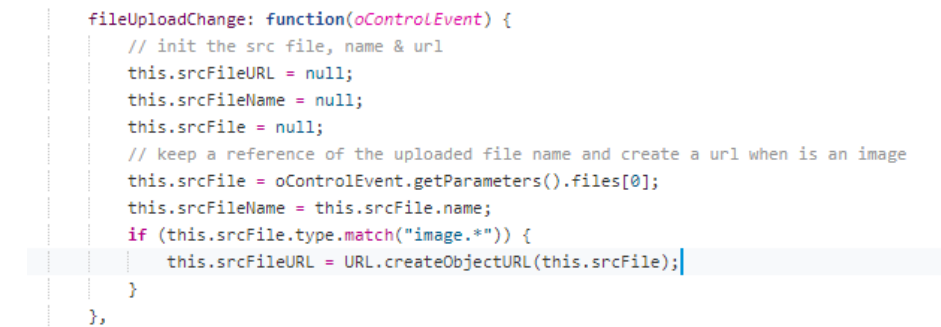
Explanation	Screenshot
<p>Open the <b>Page1.view.xml</b> file with the Layout Editor.</p> <p>In the Outline tab (left of the screen) open the second <b>IconTabFilter</b> content and select the <b>sap.m.Table</b>.</p> <p>In the Properties tab (right of the screen) enter <b>tablemgClass</b> as <b>Element ID</b>.</p>	
<p>In the Outline tab (left of the screen), inside the <b>sap.m.Table</b> we selected in previous step now select the <b>Items</b> -&gt; <b>sap.m.ColumnListItem</b> element.</p> <p>In the Properties tab (right of the screen) enter <b>collmgClass</b> as <b>Element ID</b>.</p> <p>If you have more than one <b>ColumnListItem</b> you can remove them.</p>	

Now let's map each column in the Table to our backend response properties.








Explanation	Screenshot
<p>Open the <b>Page1.view.xml</b> file with the Layout Editor.</p> <p>Open the Outline tab (left of the screen) and select the <b>sap.m.ColumnListItem</b> -&gt; <b>cells</b> element.</p> <p>Make sure you get 6 cells defined with the types as marked in the screen capture. Maybe the easier is to delete them and recreate them in the right order.</p>	
<p>For the <b>sap.m.Image</b> open the <b>detailBox</b> and enter <b>{image}</b> in the <b>Src</b> property.</p>	
<p>For the cells of type <b>sap.m.Text</b> go over them and set the Text property to the different properties names returned by the <b>smbmkt</b> backend:</p> <ul style="list-style-type: none"> <li>{productid}</li> <li>{name}</li> <li>{score}</li> <li>{inventoryLevel}</li> </ul>	

Explanation	Screenshot
<p>For the last cell of type <b>sap.m.ObjectNumber</b> set the property <b>Number</b> to <b>{price}</b> and <b>Unit</b> to <b>{priceCurrency}</b>.</p>	
<p>Open the <b>Page1.view.xml</b> file with the Code Editor.</p> <p>Search for the <b>Table</b> with id <b>"tableImgClass"</b> we updated in previous steps.</p> <p>Add the following property to indicate the path to the results from <b>SimilarItems</b>:</p> <pre>items="{path : '/result'}"</pre>	

## vi. Implement the Page1.controller.js.

Explanation	Screenshot
<p>Open the <b>Page1.controller.js</b> file.</p> <p>Implement the <b>fileUploadChange</b> function.</p> <p>This function will be called when a file has been selected.</p>	
<p>You can get the code from the following link:</p> <p><a href="https://github.com/B1SA/smbmkt/blob/CF_exercise/exercise/extras/STEP">https://github.com/B1SA/smbmkt/blob/CF_exercise/exercise/extras/STEP</a></p>	

Explanation	Screenshot
<a href="#">%205/Page1.controller.js_ext.txt</a>	<pre> // keep a reference of the uploaded file name and create a url out of that when this is an image  this.srcFile = oControlEvent.getParameters().files[0]; this.srcFileName = this.srcFile.name; if (this.srcFile.type.match("image.*")) {     this.srcFileURL = URL.createObjectURL(this.srcFile); } }, </pre>
<p>Now let's implement the <b>fileUploadComplete</b> function.</p> <p>This function will be called after the fileUploader uploadUrl ({demo&gt;/url}) has been called and a response returned.</p>	<pre> fileUploadComplete: function (oControlEvent) {     // get the current view     var oView = this.getView();     // smbmk backend     // clear previous results from the model     oView.getModel("demo").setProperty("/result", null);     var processResult = function (oController, data) {         oView = oController.getView();          // merge with existing results - working with B1 only on this case         var result = oView.getModel("demo").getProperty("/result");         if (result) {             result.push.apply(result, data.b1);         } else {             result = data.b1;         }         oView.getModel("demo").setProperty("/result", result);         oView.getModel("demo").setProperty("/fileURL", oController.srcFileURL);          // Set Model to Table         var oTable = oView.byId("tableImgClass");         oTable.setModel(oView.getModel("demo"));     };     if (oControlEvent.getParameters().status === 200) {         // get the response as JSON and process the results         processResult(this, JSON.parse(oControlEvent.getParameters().responseRaw));     } else {         MessageToast.show("Error " + oControlEvent.getParameters().status + " : " + JSON.parse(oControlEvent     } } </pre>
<p>You can get the code from the following link:</p> <p><a href="https://github.com/B1SA/smbmk/blob/CF_exercise/exercise/extras/STEP%205/Page1.controller.js_ext.txt">https://github.com/B1SA/smbmk/blob/CF_exercise/exercise/extras/STEP%205/Page1.controller.js_ext.txt</a></p>	<pre> fileUploadComplete: function (oControlEvent) {     // get the current view     var oView = this.getView();     // smbmk backend     // clear previous results from the model     oView.getModel("demo").setProperty("/result", null);     var processResult = function (oController, data) {         oView = oController.getView();          // merge with existing results - working with B1 only on this case         var result = oView.getModel("demo").getProperty("/result");         if (result) {             result.push.apply(result, data.b1);         } else {             result = data.b1;         }         oView.getModel("demo").setProperty("/result", result);     };     if (oControlEvent.getParameters().status === 200) {         // get the response as JSON and process the results         processResult(this, JSON.parse(oControlEvent.getParameters().responseRaw));     } else {         MessageToast.show("Error " + oControlEvent.getParameters().status + " : " + JSON.parse(oControlEvent     } } </pre>

Explanation	Screenshot																					
	<pre>oView.getModel("demo").setProperty("/fileURL", oController.srcFileURL);  // Set Model to Table var oTable = oView.byId("tableImgClass"); oTable.setModel(oView.getModel("demo"));  };  if (oControlEvent.getParameters().status === 200) {     // get the response as JSON and process the results     processResult(this, JSON.parse(oControlEvent.getParameters().responseRaw)); } else {     MessageToast.show("Error " + oControlEvent.getParameters().status + " : " + JSON.parse(oControlEvent.getParameters().responseRaw).error_description); }  }</pre>																					
<p>Add the MessageToast definition at the beginning of the file.</p> <pre>"sap/m/MessageToast", , MessageToast</pre>	<pre>sap.ui.define(["sap/ui/core/mvc/Controller", "sap/m/MessageBox", "sap/m/MessageToast", "./utilities", "sap/ui/core/routing/History" ], function (BaseController, MessageBox, MessageToast, Utilities, History) {     "use strict";</pre>																					
<p>Now the code should be complete.</p> <p>Run your SAP Fiori application to check all the new features.</p> <p>Press Browse and choose an image containing a shoe.</p> <p>The SimilarItems backed API is called and similar items result shown in the table.</p>	<div><div>All ItemsMatching Items</div><div></div><div><div>Fly_beige.PNG</div><div>Browse...</div></div><div>Items ( 23 )</div><table><tr><th>Image</th><th>Product Id</th><th>Name</th><th>Inventory Level</th><th>Score</th><th>Price</th><th></th></tr><tr><td></td><td>SH90018</td><td>Fly London - YUPI - Beige</td><td>0.99999999999999758</td><td>0</td><td>190 GBP</td><td>&gt;</td></tr><tr><td></td><td>SH90012</td><td>Fly London Yipi Shoes - Camel</td><td>0.9078977280950902</td><td>0</td><td>180 GBP</td><td>&gt;</td></tr></table></div>	Image	Product Id	Name	Inventory Level	Score	Price			SH90018	Fly London - YUPI - Beige	0.99999999999999758	0	190 GBP	>		SH90012	Fly London Yipi Shoes - Camel	0.9078977280950902	0	180 GBP	>
Image	Product Id	Name	Inventory Level	Score	Price																	
	SH90018	Fly London - YUPI - Beige	0.99999999999999758	0	190 GBP	>																
	SH90012	Fly London Yipi Shoes - Camel	0.9078977280950902	0	180 GBP	>																

Congratulations! You have just implemented your first full stack loosely coupled extension!

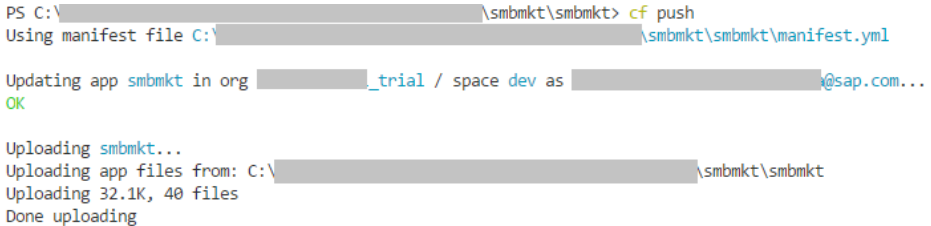
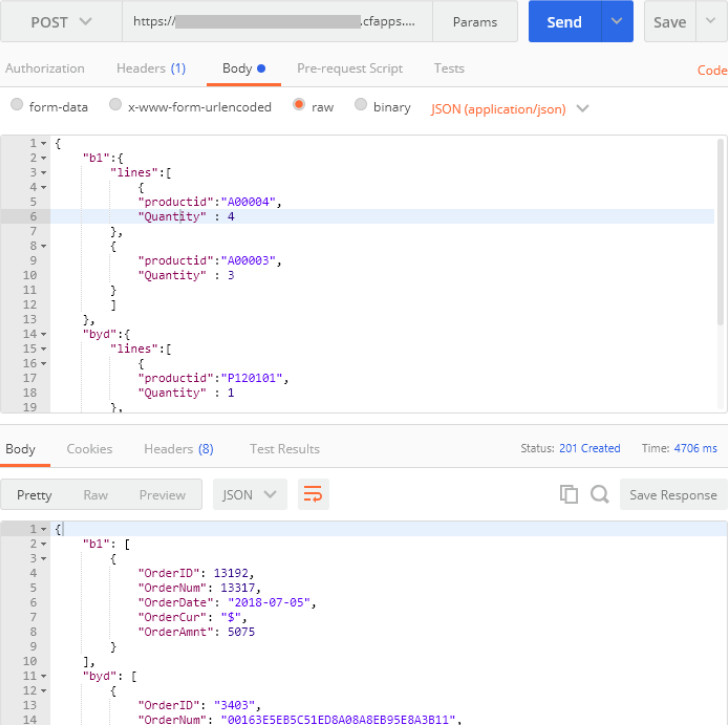


## STEP 6: ADD A NEW SERVICE TO THE NODEJS APPLICATION


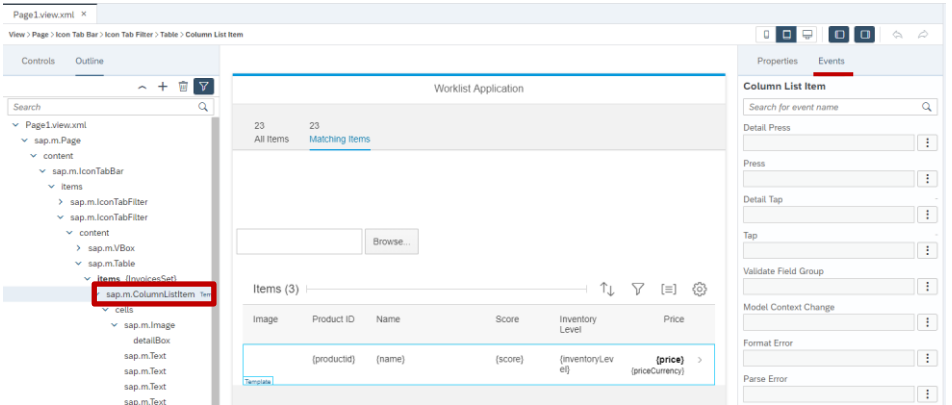
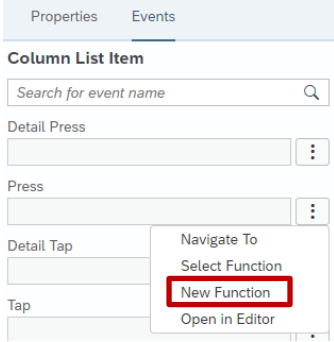
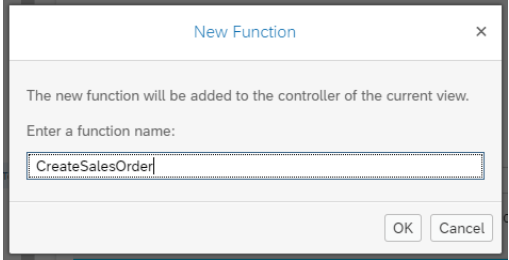
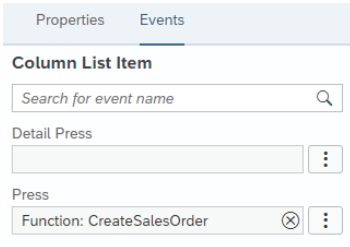
Let's add a new service to the NodeJS app that will create Sales Orders in the ERP system.

Explanation	Screenshot
<p>Go to the <b>smbmkt</b> folder you deployed before in Cloud Foundry.</p> <p>Open the <b>app.js</b> file with a Java Script editor (Visual Studio Code is an option).</p> <p>Add a post service called <b>/SalesOrders</b>.</p> <p>This service will call a function in the <b>biz</b> module.</p> <p>You can get the code from the following link:  <a href="https://github.com/B1SA/smbmkt/blob/CF_exercise/exercise/extras/STEP%206/app_ext.txt">https://github.com/B1SA/smbmkt/blob/CF_exercise/exercise/extras/STEP%206/app_ext.txt</a></p>	<pre> app.post('/SalesOrders', function (req, res) {   console.log("REQUEST: Create Sales Order")   biz.CreateSalesOrder(req.body, function (response) {     res.setHeader('Content-Type', 'application/json')     res.status(201)     res.send(response)   }) }); </pre>
<p>Open the <b>modules/biz.js</b> file.</p> <p>Add a function called <b>CreateSalesOrder</b>.</p> <p>You can get the code from the following link:  <a href="https://github.com/B1SA/smbmkt/blob/CF_exercise/exercise/extras/STEP%206/biz_ext.txt">https://github.com/B1SA/smbmkt/blob/CF_exercise/exercise/extras/STEP%206/biz_ext.txt</a></p>	<pre> function CreateSalesOrder(body, callback) {   /* Receives a body with all items from each erp */    var fResp = {};    call = 0;   for (key in body) {     var re = PostErpSalesOrder(key, body[key]).then(function (salesOrder) {       fResp[Object.keys(salesOrder)] = salesOrder[Object.keys(salesOrder)].values;       call++;       if (call == Object.keys(body).length) {         callback(fResp)       }     })   } } </pre>
<p>In the <b>modules/biz.js</b> file.</p> <p>Add a function called <b>PostErpSalesOrder</b>.</p> <p>This function will create a new sales order in the corresponding erp module (B1 or ByD) for each item ordered.</p> <p>You can get the code from the following link:  <a href="https://github.com/B1SA/smbmkt/blob/CF_exercise/exercise/extras/STEP%206/biz_ext.txt">https://github.com/B1SA/smbmkt/blob/CF_exercise/exercise/extras/STEP%206/biz_ext.txt</a></p>	<pre> let PostErpSalesOrder = function (origin, body) {   return new Promise(function (resolve, reject) {     var erp = eval(origin);      erp.PostSalesOrder(body, function (error, salesOrder) {       if (error) {         salesOrder = {};         salesOrder.error = error;       }       var output = {};       if (salesOrder.hasOwnProperty("value")) {         salesOrder = salesOrder.value       }        output[origin] = { values: salesOrder.error    salesOrder }       resolve(normalize.SalesOrders(output))     })   }) } </pre>

Explanation	Screenshot
<p>In the <b>modules/biz.js</b> file.</p> <p>Declare in module.exports the <b>CreateSalesOrder</b> function.</p> <p>You can get the code from the following link:</p> <p><a href="https://github.com/B1SA/smbmkt/blob/CF_exercise/exercise/extras/STEP%206/biz_ext.txt">https://github.com/B1SA/smbmkt/blob/CF_exercise/exercise/extras/STEP%206/biz_ext.txt</a></p>	<pre> module.exports = {   GetItems: function (query, callback) {     return (GetItems(query, callback))   },   GetSalesOrders: function (options, callback) {     return (GetSalesOrders(options, callback))   },   SimilarItems: function (body, callback) {     return (SimilarItems(body, callback))   },   CreateSalesOrder: function (body, callback) {     return (CreateSalesOrder(body, callback))   }, } </pre>
<p>Open the <b>erp/b1.js</b> file.</p> <p>Add a new function <b>PostSalesOrder</b>.</p> <p>You can get the code from the following link:</p> <p><a href="https://github.com/B1SA/smbmkt/blob/CF_exercise/exercise/extras/STEP%206/b1_ext.txt">https://github.com/B1SA/smbmkt/blob/CF_exercise/exercise/extras/STEP%206/b1_ext.txt</a></p>	<pre> function PostSalesOrder(body, callback) {   var options = {}    options.url = SLServer + "/Orders"   options.method = "POST"   options.body = {     "CardCode" : process.env.B1_DEFAULT_BP,     "DocDueDate" : moment().format('YYYY-MM-DD'),     "Comments": "Order created via SMB Mkt Place @" + moment.now(),     "DocumentLines":[]   }   options.body.DocumentLines = JSON.parse(b1Normalize(JSON.stringify(body.lines)))    options.body = JSON.stringify(options.body);    ServiceLayerRequest(options, function (error, response, body) {     if (!error &amp;&amp; response.statusCode == 201) {       console.log("Sales order created: " + body.DocEntry)       body = odata.formatResponse(JSON.parse(body));       callback(null, body);     } else {       callback(error);     }   }); } </pre>
<p>In the <b>erp/b1.js</b> file.</p> <p>Declare in module.exports the <b>PostSalesOrder</b> function.</p> <p>You can get the code from the following link:</p> <p><a href="https://github.com/B1SA/smbmkt/blob/CF_exercise/exercise/extras/STEP%206/b1_ext.txt">https://github.com/B1SA/smbmkt/blob/CF_exercise/exercise/extras/STEP%206/b1_ext.txt</a></p>	<pre> JS b1.js 1  /* Service Layer module to interact with B1 Data */ 2  /* Server Configuration and User Credentials set in environment varia 3  /* Session and Node ID stored in Redis cache database */ 4 5  var client; // Redis Client 6 7  module.exports = { 8    GetItems: function (options, callback) { 9      return (GetItems(options, callback)) 10    }, 11    GetOrders: function (options, callback) { 12      return (GetOrders(options, callback)) 13    }, 14    PostSalesOrder: function (body, callback) { 15      return (PostSalesOrder(body, callback)) 16    }, 17    setClient: function (inClient) { client = inClient; } 18  } </pre>

Explanation	Screenshot
<p>Go to your cmd line window.</p> <p>Run the cf command cf push</p> <p>To upload the changes you did to your app to Cloud Foundry.</p>	 <pre> PS C:\[redacted]\smbmkt\smbmkt&gt; cf push Using manifest file C:\[redacted]\smbmkt\smbmkt\manifest.yml  Updating app smbmkt in org [redacted]_trial / space dev as [redacted]@sap.com... OK  Uploading smbmkt... Uploading app files from: C:\[redacted]\smbmkt\smbmkt Uploading 32.1K, 40 files Done uploading </pre>
<p>You can test now your new service with Postman with the url of your app as follows:</p> <p><a href="https://smbmkt-YOURAPP/SalesOrders">https://smbmkt-YOURAPP/SalesOrders</a></p>	 <pre> POST https://[redacted].cfapps.... Authorization: [redacted] Headers: (1) Body:   form-data   x-www-form-urlencoded   raw (selected)   binary   JSON (application/json)  1 { 2   "b1":{ 3     "lines":[ 4       { 5         "productid":"A00004", 6         "Quantity" : 4 7       }, 8       { 9         "productid":"A00003", 10        "Quantity" : 3 11      } 12    ], 13    "byd":{ 14      "lines":[ 15        { 16          "productid":"P120101", 17          "Quantity" : 1 18        }, 19      ] 20    } 21  } 22 } </pre> <p>Status: 201 Created Time: 4706 ms</p> <pre> 1 { 2   "b1": [ 3     { 4       "OrderID": 13192, 5       "OrderNum": 13317, 6       "OrderDate": "2018-07-05", 7       "OrderCur": "\$", 8       "OrderAmt": 5075 9     }, 10    ], 11    "byd": [ 12      { 13        "OrderID": "3403", 14        "OrderNum": "00163E5EB5C51ED8A08A8EB95E8A3B11", </pre>

## STEP 7: CALL THE NEW NODEJS SERVICE FROM YOUR SAP FIORI APP

Explanation	Screenshot
<p>Open the <b>Page1.view.xml</b> file with the Layout Editor.</p> <p>Open the Outline tab and select <b>sap.m.ColumnsListItem</b> control we worked on previously (second TabFilter for Matching Items).</p> <p>Select the <b>Events</b> tab (right side).</p> <p>Click on the  button corresponding to the <b>Press</b> event.</p>	
<p>Choose the <b>New Function</b> option.</p>	
<p>Enter <b>CreateSalesOrder</b> as function name.</p> <p>Press <b>OK</b>.</p>	
<p>The new function will be indicated inside the Press event.</p>	

Explanation	Screenshot
<p>Open the <b>Page1.controller.js</b> file.</p> <p>A new empty function has been automatically created based on our last step.</p>	<pre> CreateSalesOrder: function (oEvent) {     //This code was generated by the layout editor. } </pre>
<p>Let's implement this function to call our smbmk backend nodejs /SalesOrder service.</p> <p>We use here the destination pointing to our smbmk backend.</p>	<pre> CreateSalesOrder: function (oEvent) {     // Get Data from ODataModel V4 /Orders     var body = {         "b1": {             "lines": [{                 "productid": oEvent.getSource().getBindingContext().getObject().productid,                 "Quantity": 1             }]         }     };      \$.ajax({         url: "/smbmk_workshopNice/SalesOrders",         type: "POST",         data: JSON.stringify(body),         contentType: "application/json",         success: function (data) {             MessageToast.show("B1 SalesOrder number " + data.b1[0].OrderNum + " created.");         },         error: function (jqXHR, textStatus, errorThrown) {             MessageToast.show("POST SalesOrders error: " + JSON.stringify(jqXHR.responseJSON));         }     }); } </pre>
<p>You can get the code from the following link:</p> <p><a href="https://github.com/B1SA/smbmk/blob/CF_exercise/exercise/extras/STEP%207/Page1.controller.js_ext.txt">https://github.com/B1SA/smbmk/blob/CF_exercise/exercise/extras/STEP%207/Page1.controller.js_ext.txt</a></p> <p><b>Replace smbmk_destination with your smbmk destination name.</b></p>	<pre> CreateSalesOrder: function (oEvent) {     // Get Data from ODataModel V4 /Orders     var body = {         "b1": {             "lines": [{                 "productid": oEvent.getSource().getBindingContext().getObject().productid,                 "Quantity": 1             }]         }     };      \$.ajax({         url: "/smbmk_destination/SalesOrders",         type: "POST",         data: JSON.stringify(body),         contentType: "application/json",         success: function (data) {             MessageToast.show("B1 SalesOrder number " + data.b1[0].OrderNum + " created.");         },         error: function (jqXHR, textStatus, errorThrown) {             MessageToast.show("POST SalesOrders error: " + JSON.stringify(jqXHR.responseJSON));         }     }); } </pre>

Congratulations! You have just implemented your first full stack loosely coupled extension!

**www.sap.com/contactsap**

© 2018 SAP SE or an SAP affiliate company. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP SE or an SAP affiliate company.

The information contained herein may be changed without prior notice. Some software products marketed by SAP SE and its distributors contain proprietary software components of other software vendors. National product specifications may vary.

These materials are provided by SAP SE or an SAP affiliate company for informational purposes only, without representation or warranty of any kind, and SAP or its affiliated companies shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP or SAP affiliate company products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

In particular, SAP SE or its affiliated companies have no obligation to pursue any course of business outlined in this document or any related presentation, or to develop or release any functionality mentioned therein. This document, or any related presentation, and SAP SE's or its affiliated companies' strategy and possible future developments, products, and/or platform directions and functionality are all subject to change and may be changed by SAP SE or its affiliated companies at any time for any reason without notice. The information in this document is not a commitment, promise, or legal obligation to deliver any material, code, or functionality. All forward-looking statements are subject to various risks and uncertainties that could cause actual results to differ materially from expectations. Readers are cautioned not to place undue reliance on these forward-looking statements, and they should not be relied upon in making purchasing decisions.

SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP SE (or an SAP affiliate company) in Germany and other countries. All other product and service names mentioned are the trademarks of their respective companies. See <http://www.sap.com/corporate-en/legal/copyright/index.epx> for additional trademark information and notices.