

CAPM Binding Data to a Table in a Basic SAP UI5 Application (Chapter 1 - READ)

Authored by:

Maresh L

SAP UI5 Fiori Associate Consultant

Email: lokkidimahesh6@gmail.com

This document provides a beginner-friendly guide to binding data to a table in a basic SAP UI5 application, focusing on integration with an SAP HANA database and deployment within the Cloud Foundry runtime environment. It will walk you through setting up your development environment, creating data models, and connecting them to your UI5 application.

What You Will Learn:

- Setting up the development environment for binding data to a UI5 table.
- Connecting an SAP HANA database with your SAP UI5 application.
- Creating an XML view control for the UI5 table.
- Binding data with OData V4 to populate the table in the UI.

Chapter 1: Implementing CRUD Operations – Read (Chapter 1)

This guide currently covers the **Read** operation (data retrieval) in the SAP UI5 table. The **Create**, **Delete**, and **Update** operations will be covered in the upcoming chapters of this series.

Prerequisites:

- Basic knowledge of JavaScript and Node.js.
- Familiarity with SAP BTP.
- Installed tools: Node.js, npm, and SAP Business Application Studio or Visual Studio Code.

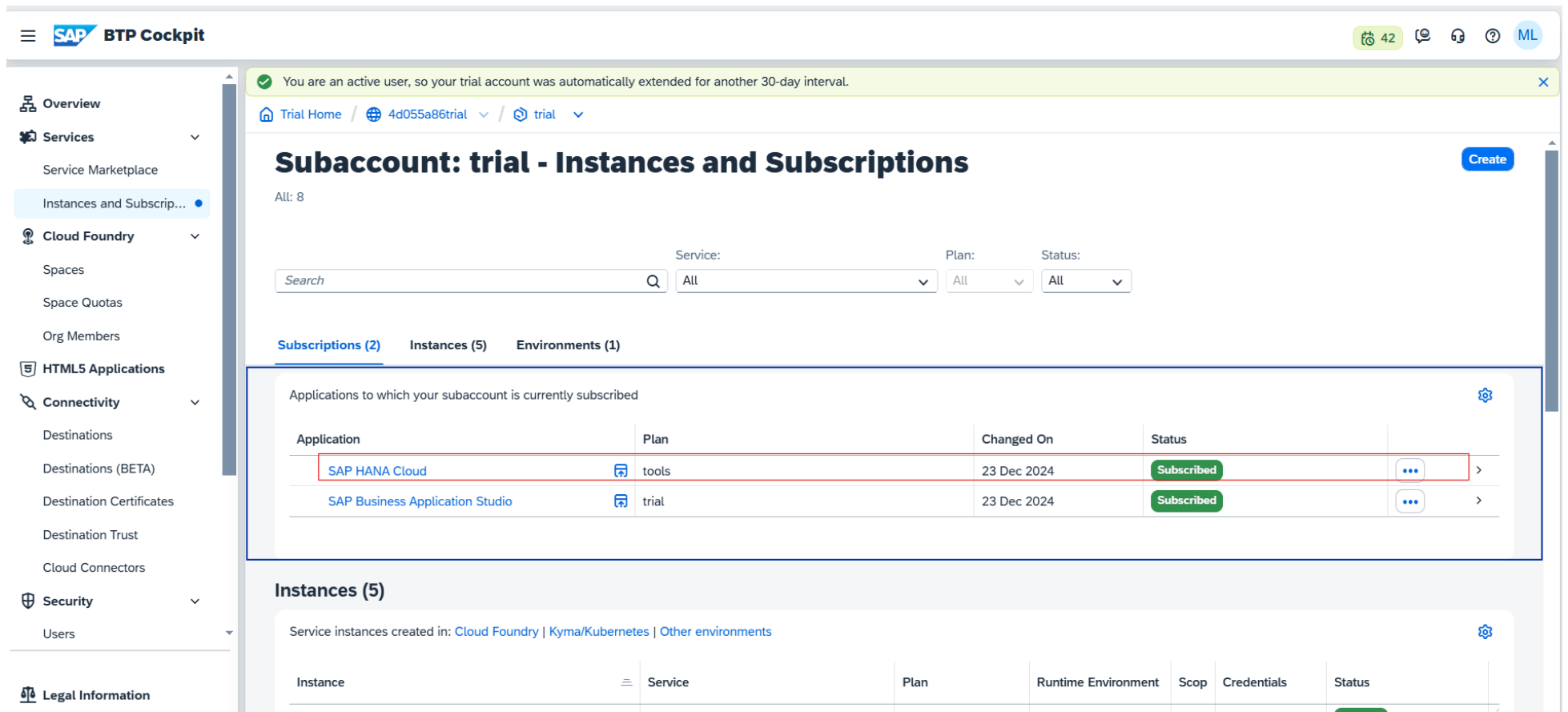
For detailed steps on database and instance creation in CAPM, please refer to the following document:

https://www.linkedin.com/posts/mahesh-lokkidi-8664b9135_sap-capm-steps-pdf-guide-activity-7278389861781315585-nKxN?utm_source=share&utm_medium=member_desktop

1. Open SAP HANA Cloud

If SAP HANA Cloud is not visible in your environment, please subscribe to SAP HANA Cloud first. Once subscribed, proceed with creating a Cloud Foundry instance.

For detailed instructions and reference, kindly refer to the document linked above.



The screenshot shows the SAP BTP Cockpit interface. The left sidebar contains navigation links: Overview, Services, Cloud Foundry, HTML5 Applications, Connectivity, Security, and Legal Information. The main content area is titled 'Subaccount: trial - Instances and Subscriptions'. It features a search bar and filters for Service, Plan, and Status. Below the filters, there are tabs for Subscriptions (2), Instances (5), and Environments (1). The 'Subscriptions (2)' tab is active, showing a table of applications to which the subaccount is currently subscribed. The table has columns for Application, Plan, Changed On, and Status. Two subscriptions are listed: 'SAP HANA Cloud' and 'SAP Business Application Studio'. The 'SAP HANA Cloud' subscription is highlighted with a red box, and its status is 'Subscribed'. The 'Instances (5)' tab is also visible, showing a table of service instances created in the Cloud Foundry environment.

Application	Plan	Changed On	Status
SAP HANA Cloud	tools	23 Dec 2024	Subscribed
SAP Business Application Studio	trial	23 Dec 2024	Subscribed

2.Start SAP HANA Cloud

Click on **Start** to initiate the SAP HANA Cloud service.

Note: For trial accounts, it is required to start the service daily. However, this step is not necessary for real-time (paid) accounts, as the service remains active.

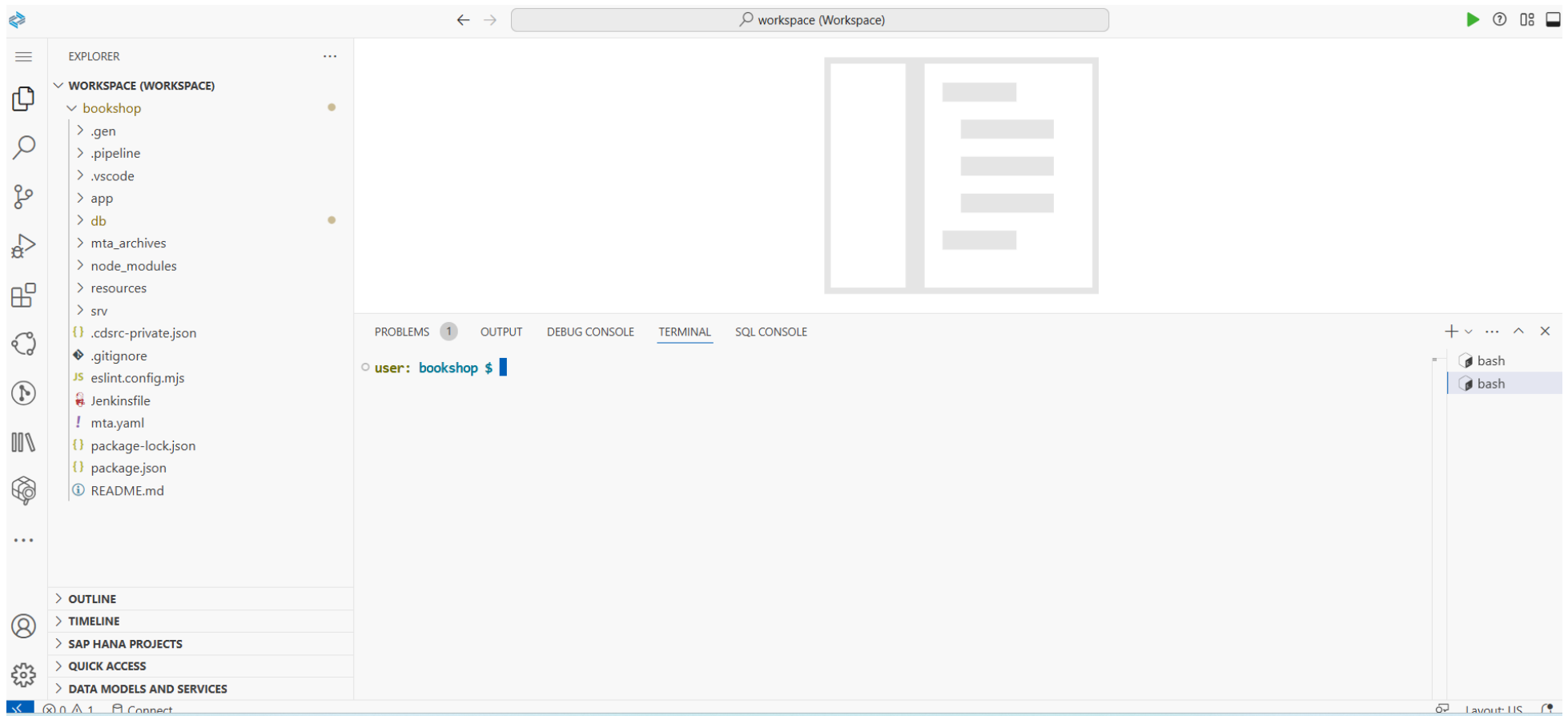
The screenshot shows the SAP HANA Cloud Central 'Instances' page. The header includes the SAP logo, 'HANA Cloud Central', and a 'Search Commands' bar. The left sidebar contains navigation icons. The main area is titled 'All Instances' and includes filters for Search, State, Notifications, Type, and Version. A table lists instances, with one instance 'SHDB' in a 'Stopped' state. A context menu is open over the 'Start' button in the table row, showing options like 'Manage Configuration', 'Add Data Lake', 'Copy SQL Endpoint', 'Copy Instance ID', 'Copy Configuration', 'Apply Patch', 'Upgrade', 'Create Template to Clone Instance', 'Start', and 'Delete'. The 'Start' option is highlighted with a red box.

State	Name	Type	Notifications	Runtime Environment	Memory	Storage	Compute	Actions
Stopped	SHDB	SAP HANA Database	1	Cloud Foundry	16 GB	80 GB	1 vCPUs	1 node 0 replicas Start

3. Open SAP Business Application Studio

Open the **SAP Business Application Studio**.

In this example, I have already created a **CAPM Bookshop** project. For your reference, I have provided the document link on the first page. The project I created is named **Supermarket**, which is similar to the Bookshop project.

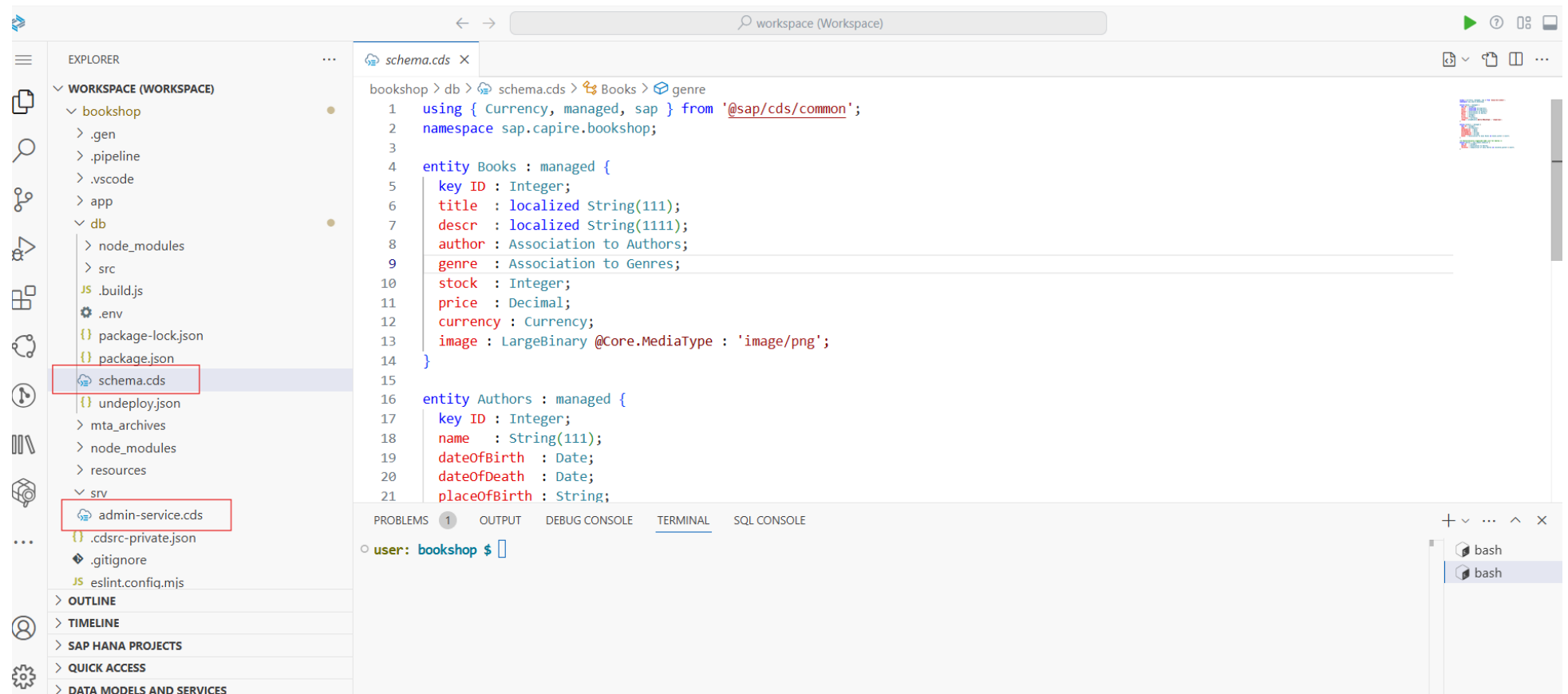


4. Create CDS Files for Database and Service

I have created the following CDS files:

1. **schema.cds** – This file defines the database schema and tables for the application.
2. **admin-service.cds** – This service file defines the OData services for interacting with the database.

These files are essential for setting up the database structure and exposing the necessary APIs for CRUD



schema.cds

```
using { Currency, managed, sap } from '@sap/cds/common';
namespace sap.capire.bookshop;

entity Books : managed {
  key ID : Integer;
  title : localized String(111);
  descr : localized String(111);
  author : Association to Authors;
  genre : Association to Genres;
  stock : Integer;
  price : Decimal;
  currency : Currency;
  image : LargeBinary @Core.MediaType : 'image/png';
}

entity Authors : managed {
  key ID : Integer;
  name : String(111);
  dateOfBirth : Date;
  dateOfDeath : Date;
  placeOfBirth : String;
  placeOfDeath : String;
  books : Association to many Books on books.author = $self;
}

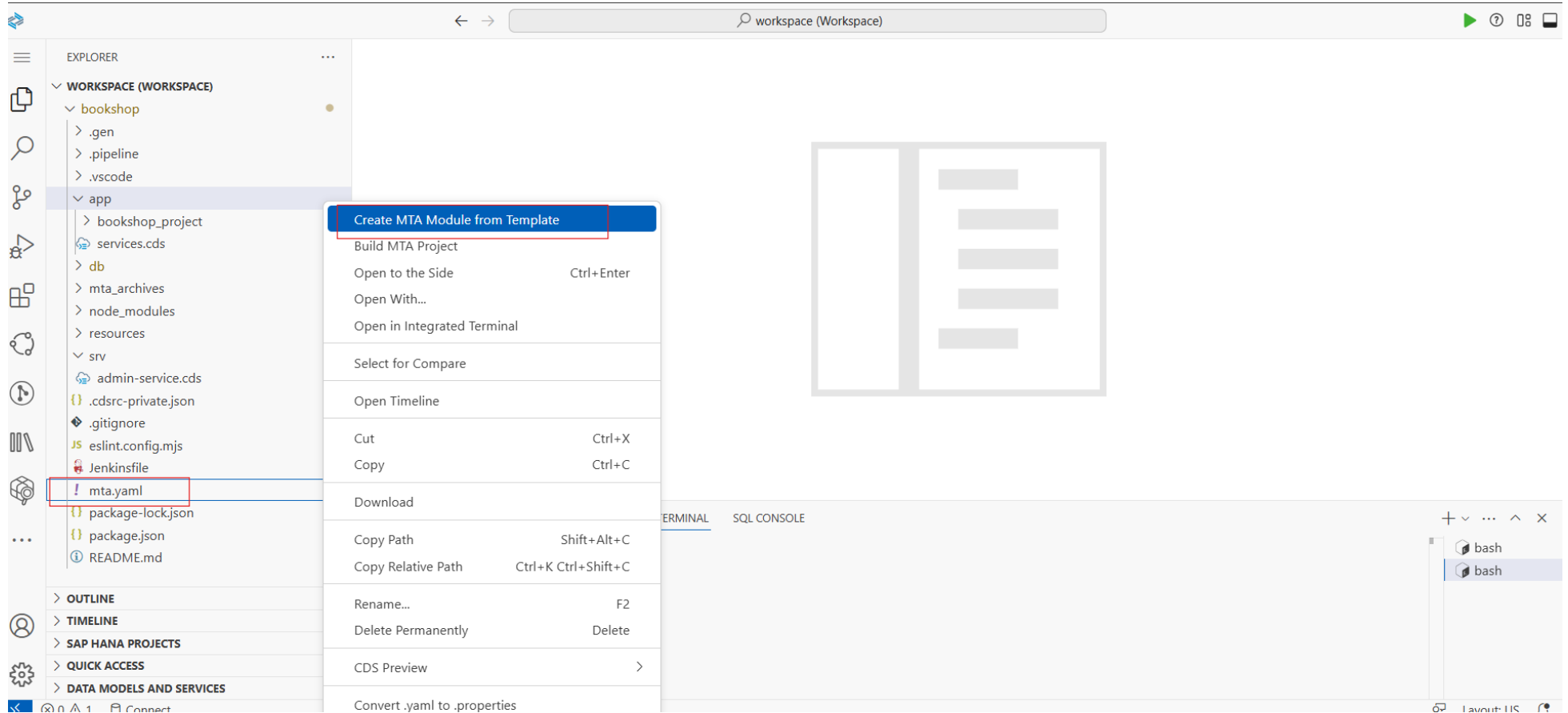
/** Hierarchically organized Code List for Genres */
entity Genres : sap.common.CodeList {
  key ID : Integer;
  parent : Association to Genres;
  children : Composition of many Genres on children.parent = $self;
}

admin-service.cds

using { sap.capire.bookshop as my } from './db/schema';
service AdminService @(requires:'any') {
  entity Books as projection on my.Books;
  entity Authors as projection on my.Authors;
}
```

5. Create MTA Module from Template

In the **MTA (Multi-Target Application)** file, right-click and select **Create MTA Module from Template**.



6.Create a Freestyle UI5 Application with SAP Fiori Generator

The screenshot displays the SAP Business Studio interface. On the left, the **EXPLORER** pane shows the project structure under **WORKSPACE (WORKSPACE)**. The **bookshop** folder is expanded, showing subfolders like **.gen**, **.pipeline**, and **.vscode**. The **app** folder is selected, revealing its contents, including **bookshop_project**, **services.cds**, **db**, **mta_archives**, **node_modules**, **resources**, **srv**, **admin-service.cds**, **.cdsrc-private.json**, **.gitignore**, **eslint.config.mjs**, **Jenkinsfile**, **mta.yaml**, **package-lock.json**, **package.json**, and **README.md**. The **OUTLINE**, **TIMELINE**, **SAP HANA PROJECTS**, **QUICK ACCESS**, and **DATA MODELS AND SERVICES** panes are also visible.

The main workspace area shows the **New MTA Module From Template** wizard. The **Select Module Template** step is active, showing the **SAP Fiori generator** template. The path to the selected project's 'mta.yaml' file is `/home/user/projects/bookshop/mta.yaml. The wizard offers three templates:`

- Approuter Configur...**: Add an approuter configuration to your multitarget application (MTA) project. [More Information](#)
- SAP HANA Database ...**: A database module contains design-time database artifacts. On deployment, a database module is assigned an HDI container to which the corresponding run-time database objects are deployed. [More Information](#)
- SAP Fiori generator**: Create an SAPUI5 application using SAP Fiori elements or a freestyle approach. [More Information](#)

The **SAP Fiori generator** template is highlighted. A **Start >** button is located at the bottom of the wizard.

7. Select Basic and Next

The screenshot displays the SAP Fiori generator web application interface. On the left, the 'EXPLORER' sidebar shows a project structure under 'WORKSPACE (WORKSPACE)' with folders like '.gen', '.pipeline', '.vscode', and 'app'. The 'app' folder is expanded, showing sub-items like 'bookshop_project', 'services.cds', 'db', 'mta_archives', 'node_modules', 'resources', 'srv', 'admin-service.cds', '.cdsrc-private.json', '.gitignore', 'eslint.config.mjs', 'Jenkinsfile', 'mta.yaml', 'package-lock.json', 'package.json', and 'README.md'. Below the explorer are sections for 'OUTLINE', 'TIMELINE', 'SAP HANA PROJECTS', 'QUICK ACCESS', and 'DATA MODELS AND SERVICES'.

The main content area is titled 'SAP Fiori generator' and shows the 'Template Selection' step. The 'Select Module Template' option is selected, with 'SAP Fiori generator' as the chosen template. Below this, a list of options is shown: 'Template Selection' (selected), 'Data Source and Service Selection', 'Entity Selection', and 'Project Attributes'. The 'Template: Basic' option is highlighted.

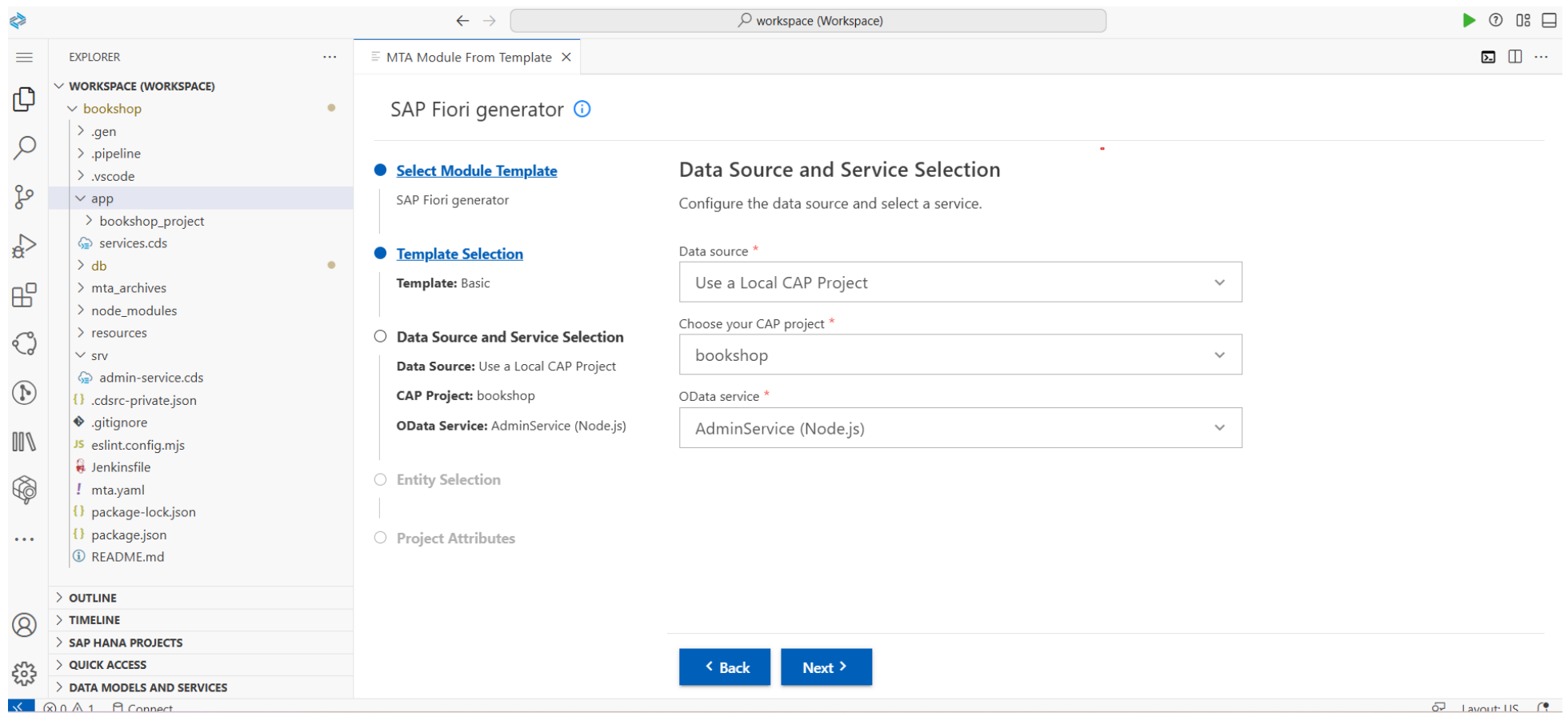
The 'Template Selection' section prompts the user to 'Choose your application template.' and asks 'Which template do you want to use? *'. Three templates are presented as cards:

- Basic**: Create a freestyle application, starting with an empty page.
- Custom Page**: Create an SAP Fiori elements application containing a custom page based on the flexible programming model.
- List Report Page**: Create an SAP Fiori elements application containing a list report and an object page.

Each card includes a 'More Information' link and a preview image. At the bottom, there are two blue buttons: '< Start Over' and 'Next >'.

8. Select Local CAP Project and Service

1. Choose the **service** (such as admin-service.cds) that you defined within the CAP project.



9. Enter Project Attributes

EXPLORER

WORKSPACE (WORKSPACE)

bookshop

> .gen

> .pipeline

> .vscode

app

bookshop_project

services.cds

db

mta_archives

node_modules

resources

srv

admin-service.cds

.cdsrc-private.json

.gitignore

eslint.config.mjs

Jenkinsfile

mta.yaml

package-lock.json

package.json

README.md

OUTLINE

TIMELINE

SAP HANA PROJECTS

QUICK ACCESS

MTA Module From Template X

SAP Fiori generator

Select Module Template

SAP Fiori generator

Template Selection

Template: Basic

Data Source and Service Selection

Data Source: Use a Local CAP Project

CAP Project: bookshop

OData Service: AdminService (Node.js)

Entity Selection

View Name: HomePage

Project Attributes

Module Name: retail_bookstore

Application Title: Retail BookStore Project

Application Namespace: com.bookstore

Description: An SAP Fiori application.

More...

Deployment Configuration

Project Attributes

Configure the main project attributes.

Module name [?] *

retail_bookstore

Application title [?]

Retail BookStore Project

Application namespace [?]

com.bookstore

Description [?]

An SAP Fiori application.

Minimum SAPUI5 version [?] *

1.132.1

Enable TypeScript

☐ Yes ☒ No

Add deployment configuration to MTA project (/home/user/projects/bookshop/mta.yaml)

☒ Yes ☐ No

Add FLP configuration

☐ Yes ☒ No

Configure advanced options [?]

☐ Yes ☒ No

< Back

Next >

[?] The generated project will not open in a stand-alone folder.

10. Select Cloud Foundry (earlier which is used in CAP creation)

The screenshot displays the SAP Fiori generator interface. On the left, the 'EXPLORER' pane shows a project structure under 'WORKSPACE (WORKSPACE)' with folders like '.gen', '.pipeline', '.vscode', 'app', 'bookshop_project', 'services.cds', 'db', 'mta_archives', 'node_modules', 'resources', 'srv', 'admin-service.cds', and various configuration files. The main area is titled 'MTA Module From Template' and 'SAP Fiori generator'. It features a sidebar with steps: 'Select Module Template', 'Template Selection', 'Data Source and Service Selection', 'Entity Selection', 'Project Attributes', and 'Deployment Configuration'. The 'Deployment Configuration' step is active, showing 'Target Type: Cloud Foundry' and 'Destination Name: Local CAP Project API (Instance Based Destination)'. A 'Finish' button is visible, along with a note: 'The generated project will not open in a stand-alone folder.'

EXPLORER

WORKSPACE (WORKSPACE)

- bookshop
 - .gen
 - .pipeline
 - .vscode
 - app
 - bookshop_project
 - services.cds
 - db
 - mta_archives
 - node_modules
 - resources
 - srv
 - admin-service.cds
 - .cdsrc-private.json
 - .gitignore
 - eslint.config.mjs
 - Jenkinsfile
 - mta.yaml
 - package-lock.json
 - package.json
 - README.md

OUTLINE

TIMELINE

SAP HANA PROJECTS

QUICK ACCESS

DATA MODELS AND SERVICES

MTA Module From Template

SAP Fiori generator

Select Module Template

SAP Fiori generator

Template Selection

Template: Basic

Data Source and Service Selection

Data Source: Use a Local CAP Project

CAP Project: bookshop

OData Service: AdminService (Node.js)

Entity Selection

View Name: HomePage

Project Attributes

Module Name: retail_bookstore

Application Title: Retail BookStore Project

Application Namespace: com.bookstore

Description: An SAP Fiori application.

More...

Deployment Configuration

Target Type: Cloud Foundry

Destination Name: Local CAP Project API (Instance Based Destination)

Back Finish

The generated project will not open in a stand-alone folder.

11.Application is Ready

The screenshot displays the SAP Fiori Tools workspace interface. On the left, the Explorer pane shows the project structure under 'WORKSPACE (WORKSPACE)'. The 'app' folder is expanded, and the 'retail_bookstore' sub-project is highlighted with a red rectangle. The main area is titled 'Application Information' and contains a 'Project Details' table.

Property	Value	Property	Value	Property	Value
Identifier	com.bookstore.retailbookstore	Type	SAPUI5 freestyle	Pages	Page_Map
Title	Retail BookStore Project	Min UI5 Version	1.132.1		View HomePage
Namespace	com.bookstore	CAP Project Type	CAP Node.js		
Location	/home/user/projects/bookshop/app...	Main Service	admin (V4.0)		
Files	package.json manifest.json				

Below the table, a 'Status' section shows a green checkmark icon. Under the heading '</> What you can do', there are eight actionable cards:

- Preview Application**: Choose from start scripts to run the application preview.
- Open Page Map**: Open the page map that shows application pages and navigation paths.
- Add Deploy Config**: Add deploy configuration.
- Add Fiori Launchpad Config**: Add Flp configuration to the application.
- Deploy**: Deploy according to the configuration by default stored in 'ui5-deploy.yaml'.
- Undeploy**: Remove a deployed artifact from backend or cloud.
- Check Node Modules**: Execute command that checks all node module dependencies for newer versions.
- Create Archive**: Zip the project excluding the node_modules for sharing (e.g. support cases).
- Change minUI5Version**: Change the minimum version of
- Run UI5 Linter**: Run a static code analysis to
- Convert Preview Config**: Convert the configuration to

A notification bar at the bottom right states: 'The files have been generated.'

12. Adding the dataSources in manifest.json:

Adding the dataSources section in manifest.json binds your SAP UI5 application to the OData service exposed by your CAP project. It defines the service URL, specifies the OData version (4.0), and sets up the necessary configurations for communication between frontend and backend. This is crucial for making CRUD operations and retrieving data.

```
"dataSources": { "mainService": { "uri": "/odata/v4/admin/", "type": "OData", "settings": { "annotations": [], "odataVersion": "4.0" } } }
```

The screenshot shows the SAP Studio IDE interface. The Explorer panel on the left displays the project structure, including the 'manifest.json' file. The main editor area shows the content of 'manifest.json', with the 'dataSources' section highlighted by a red box. The 'dataSources' section is defined as follows:

```
"dataSources": {
  "mainService": {
    "uri": "/odata/v4/admin/",
    "type": "OData",
    "settings": {
      "annotations": [],
      "odataVersion": "4.0"
    }
  }
}
```

The bottom panel shows the terminal output, indicating that the application is running and the OData service is being accessed successfully.

13.Screen Shots and code for xml view and controller

The screenshot displays the SAP Studio (VS Code) interface for a project named 'bookshop'. The Explorer sidebar on the left shows the project structure, with 'HomePage.view.xml' selected under the 'view' directory. The Editor window shows the XML code for 'HomePage.view.xml', which is a SAP UI5 View. The code defines a table with four columns: Title, Author ID, Price, and Stock. The Terminal window at the bottom shows the output of the 'cds watch' command, indicating that the watch process has ended.

```
bookshop > app > retail_bookstore > webapp > view > HomePage.view.xml
1 <mvc:View controllerName="com.bookstore.retailbookstore.controller.HomePage"
2   xmlns:mvc="sap.ui.core.mvc"
3   xmlns="sap.m">
4   <Page id="page" title="{i18n>title}">
5
6   <Table
7     id="booksTable"
8     items="{
9       path: '/Books'
10    }">
11   <columns>
12     <Column id="_IDGenColumn">
13       <Text id="_IDGenText1" text="Title" />
14     </Column>
15     <Column id="_IDGenColumn1">
16       <Text id="_IDGenText2" text="Author ID" />
17     </Column>
18     <Column id="_IDGenColumn2">
19       <Text id="_IDGenText3" text="Price" />
20     </Column>
21     <Column id="_IDGenColumn3">
22       <Text id="_IDGenText4" text="Stock" />
23     </Column>
24   </columns>
25 </Table>
26 </Page>
27 </mvc:View>
```

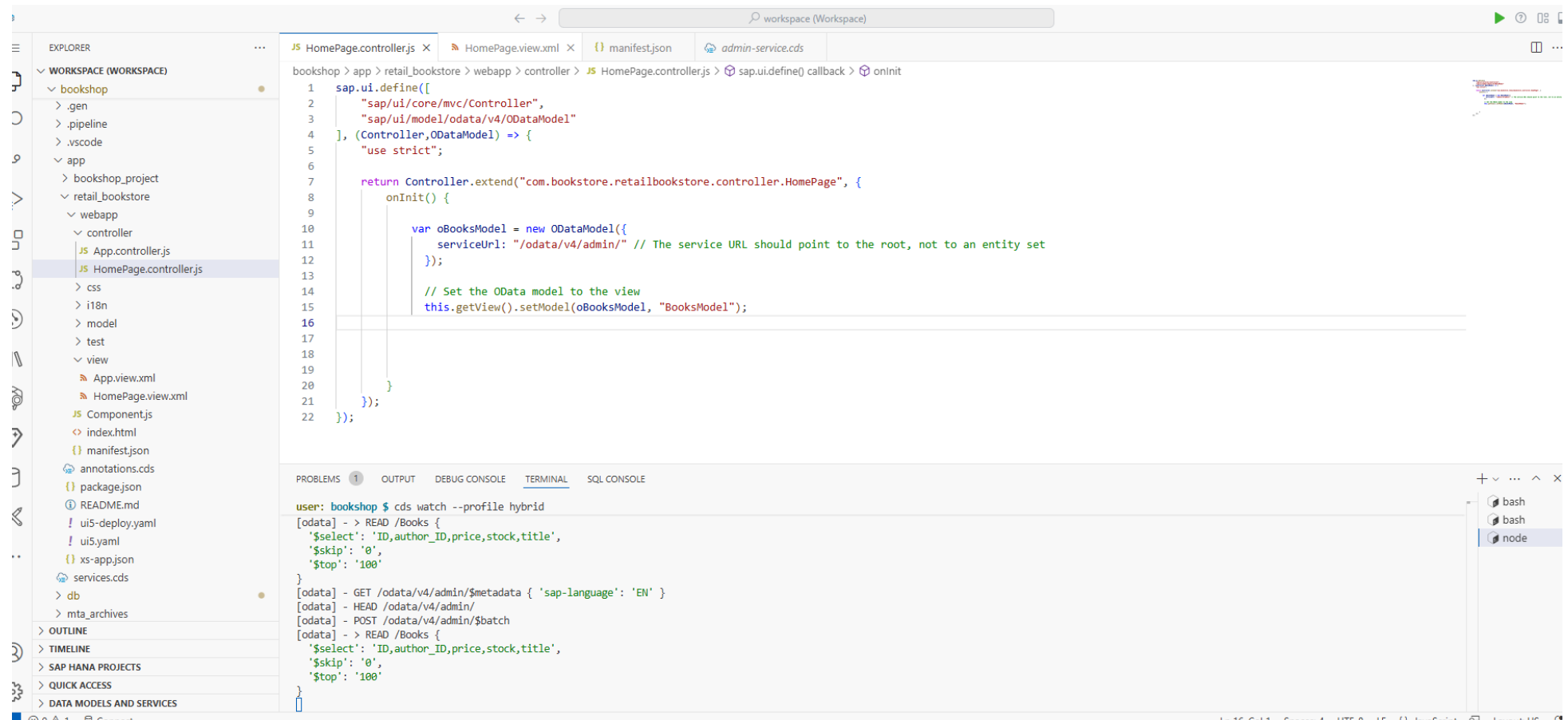
```
user: bookshop $ cds watch --profile hybrid
'$stop': '100'
}
[odata] - GET /odata/v4/admin/$metadata { 'sap-language': 'EN' }
[odata] - HEAD /odata/v4/admin/
[odata] - POST /odata/v4/admin/$batch
[odata] - > READ /Books {
  '$select': 'ID,author_ID,price,stock,title',
  '$skip': '0',
  '$top': '100'
}
^C
[cds] - my watch has ended.
user: bookshop $
```

To connect the OData service to your UI5 application, you can use the following code:

```
var oBooksModel = new ODataModel({
  serviceUrl: "/odata/v4/admin/" // The service URL should point to the root, not to an entity set
});

// Set the OData model to the view
this.getView().setModel(oBooksModel, "BooksModel");
```

This code creates a new **ODataModel** with the service URL pointing to the root of the OData service and binds it to the view with the model name BooksModel. This allows your UI to interact with the data from the backend OData service.



View.xml

```
<mvc:View controllerName="com.bookstore.retailbookstore.controller.HomePage"
  xmlns:mvc="sap.ui.core.mvc"
  xmlns="sap.m">
  <Page id="page" title="{i18n>title}">

  <Table
    id="booksTable"
    items="{
      path: '/Books'
    }">
    <columns>
      <Column id="_IDGenColumn">
        <Text id="_IDGenText1" text="Title" />
      </Column>
      <Column id="_IDGenColumn1">
        <Text id="_IDGenText2" text="Author ID" />
      </Column>
      <Column id="_IDGenColumn2">
        <Text id="_IDGenText3" text="Price" />
      </Column>
      <Column id="_IDGenColumn3">
        <Text id="_IDGenText4" text="Stock" />
      </Column>
    </columns>

    <items>
      <ColumnListItem id="_IDGenColumnListItem">
        <cells>
          <Text id="_IDGenText5" text="{title}" />
        </cells>
      </ColumnListItem>
    </items>
  </Table>
</Page>
</mvc:View>
```

```
<Text id="_IDGenText6" text="{author_ID}" />
<Text id="_IDGenText7" text="{price}" />
<Text id="_IDGenText8" text="{stock}" />
</cells>
</ColumnListItem>
</items>
</Table>
</Page>
</mvc:View>
```

Controller.js

```
sap.ui.define([
  "sap/ui/core/mvc/Controller",
  "sap/ui/model/odata/v4/ODataModel"
], (Controller, ODataModel) => {
  "use strict";

  return Controller.extend("com.bookstore.retailbookstore.controller.HomePage", {
    onInit() {

      var oBooksModel = new ODataModel({
        serviceUrl: "/odata/v4/admin/" // The service URL should point to the root, not to an entity set
      });

      // Set the OData model to the view
      this.getView().setModel(oBooksModel, "BooksModel");

    }
  });
});
```

14.Application Ready – Click on Retail Bookstore

Once the application is fully set up and deployed, click on **Retail Bookstore** to access and interact with the application. This will launch the UI5 interface connected to your backend OData service.

Welcome to @sap/cds Server

Serving bookshop 1.0.0

These are the paths currently served:

Web Applications:

/bookshop_project/dist	
/bookshop_project/webapp	
/retail_bookstore/webapp	

Service Endpoints:

/odata/v4/admin / \$metadata	
Books	Fiori preview
Authors	Fiori preview
Genres	Fiori preview
Currencies	Fiori preview

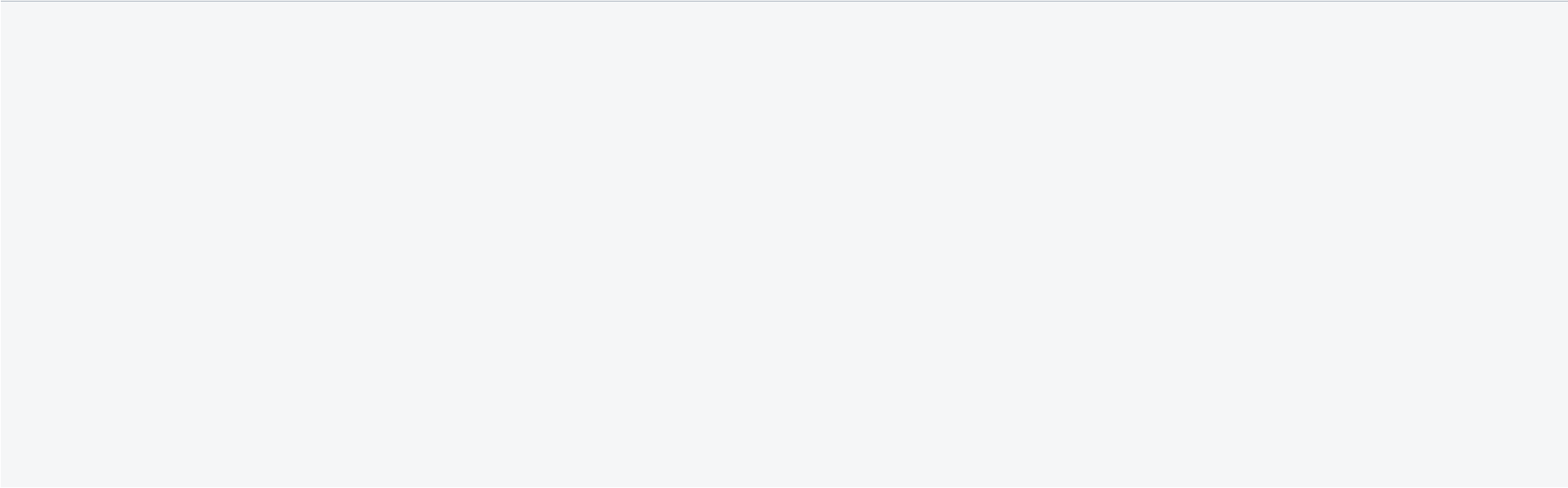
This is an automatically generated page.
You can replace it with a custom `./app/index.html`.

15.Application Final: Table View with All Data

Once the application is ready, the **Retail Bookstore** will display a table view populated with all the data from the backend OData service. This table will show a list of items, such as books, retrieved from the database, providing users with an interactive and dynamic view of the data.

Retail BookStore Project

Title	Author ID	Price	Stock
The Art of Coding	101	29.99	50
Database Essentials	102	19.99	30
Cloud Computing	103	39.99	20



Thank you for reading **Chapter 1: Implementing CRUD Operations – Read**.
The next chapter, **Create**, will be available soon. Stay tuned for more!

Thank You