## Prerequisites : -

1. Download and install [Node.js](https://nodejs.org/en/download/).
2. Download and install [SQLite3](https://www.sqlite.org/download.html).
3. Download [Visual Studio Code](https://code.visualstudio.com/) and add several extensions to streamlined development of Fiori Element applications.

**Development : -**

Step 1: - Define a Data Model

* Use CDS to define the relevant data types and entities of domain. The DB schema is stored in file **schema.cds** and needs to be created in **folder db.**
* Sample Code: -

namespace sap.capire.medical;

using { Currency } from '@sap/cds/common';

entity Medicines {

@Common.Label : 'UUID'

key ID : UUID;

identifier : String @Common.Label : 'Identifier';

title : localized String @( Common.Label : 'Medicine Name' );

availability : Integer;

price : Decimal(9,2);

currency : Currency;

}

Steps 2: - Define an oData Service

* Using the above data model, we can now build the oData Service.
* We need to create **srv folder** and create the file **cat-service.cds**
* Sample Code: -

using sap.capire.medical from '../db/schema';

service MainService {

entity Medicines as projection on medical.Medicines;

};

Steps 3: - Start the oData Server

* As **Data model and oData Service are ready**. We can start the NodeJs express and display the resulting metadata XML.
* **Command: - cds run**

Steps 4: - Add Data

* As service is running and metadata is loaded properly.

We can add some data and check the same in browser.

* But first we need to add [SQLite driver for Node.js] as a dev dependency to project
* npm add sqlite3 -D
* All data that is provide in CSV files will be deployed by CDS to the SQLite database. In our case we need data for medicines, currencies.
* To add the data **csv folder (** db/csv)**,** need to be created inside **db folder.**
* CSV files details are listed below: -
  1. Medicine CSV details: -
* Name: - sap.capire.medical-Medicines.csv
* Data: -

ID,IDENTIFIER,TITLE,AVAILABILITY,PRICE,CURRENCY\_CODE

844ede5c-071e-34ed-907f-f99cb3a4693d,PL-100,Paracetamol,200,2.65,INR

64d40922-ea0d-30f9-9b83-eb4448ee4c2e,NI-200,Nice,10,2.6,INR

2.Currencies CSV details: -

* Name: - sap.common-Currencies.csv
* Data: -

"SYMBOL";"DESCR";"NAME";"CODE"

"€";"Euro";"Euro";"EUR"

"£";"British Pound";"British Pound";"GBP"

"₹";"Indian Rupee";"Indian Rupee";"INR"

"$";"US Dollar";"US Dollar";"USD"

Steps 5: - Check the data

data can be viewed immediately in a browser when you run SQLite with option in-memory.

Command:

* Command: - cds run –in-memory

Steps 6: -

* With the backend OData server endpoint ready to use, we can now start the ***SAP UX App Generator*** to build your Fiori Application based on the OData entities.
* Create a new empty folder called **app** and select the last menu entry ***SAP UX - App Generator*** to start it.
* When the Fiori Elements app is generated successfully. The index.html file is located in subfolder webapp.
* Re-run the command cds run –in-memory. The application should be up and running.

<script src="https://veui5infra.dhcp.wdf.sap.corp/sapui5-sdk-internal/test-resources/sap/ushell/bootstrap/sandbox.js"></script>

<script

src="https://veui5infra.dhcp.wdf.sap.corp/sapui5-sdk-internal/resources/sap-ui-core.js"

Steps 7: - Enhance the cat-service.cds files

* Add the required UI Annotations and enable the CRUD operations
* Sample Code :-

using sap.capire.medical from '../db/schema';

service MainService {

@odata.draft.enabled

entity Medicines as projection on medical.Medicines;

};

annotate MainService.Medicines with @(

UI: {

HeaderInfo: {

TypeName: 'Medicine',

TypeNamePlural: 'Medicines',

Title: { $Type: 'UI.DataField', Value: title }

},

SelectionFields: [ identifier, title, availability, price],

LineItem: [

{$Type: 'UI.DataField', Value: identifier},

{$Type: 'UI.DataField', Value: title},

{$Type: 'UI.DataField', Value: availability},

{$Type: 'UI.DataField', Value: price}

],

HeaderFacets: [

{$Type: 'UI.ReferenceFacet', Target: '@UI.FieldGroup#MedicineDetail',Label: 'Medical Details'}

],

Facets: [

{

$Type: 'UI.CollectionFacet',

Label: 'Medicine Information',

Facets: [

{

$Type: 'UI.ReferenceFacet',

Target: '@UI.FieldGroup#DetailsOfMedicine',

Label: 'Detail Description'

}

]

}

],

FieldGroup#MedicineDetail: {

Data:[

{$Type: 'UI.DataField', Value: identifier},

{$Type: 'UI.DataField', Value: availability}

]

},

FieldGroup#DetailsOfMedicine: {

Data:[

{$Type: 'UI.DataField', Value: title},

{$Type: 'UI.DataField', Value: availability}

]

}

}

);

annotate MainService.Medicines with {

identifier @( Common.Label : 'Medicine' ) ;

availability @( Common.Label : 'Total Available Quantity' );

price @( Common.Label : 'Price', Measures.ISOCurrency: currency\_code );

}