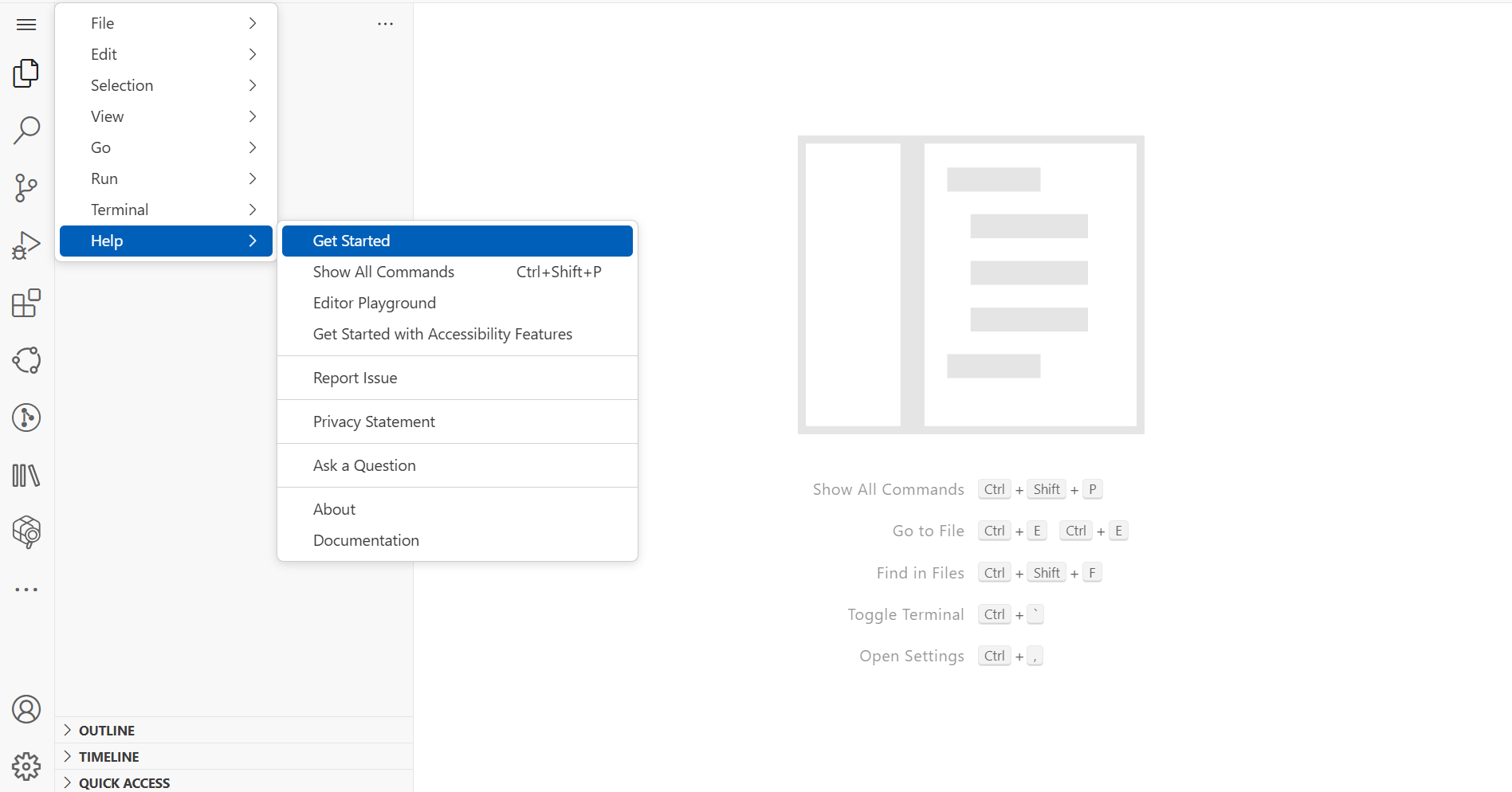
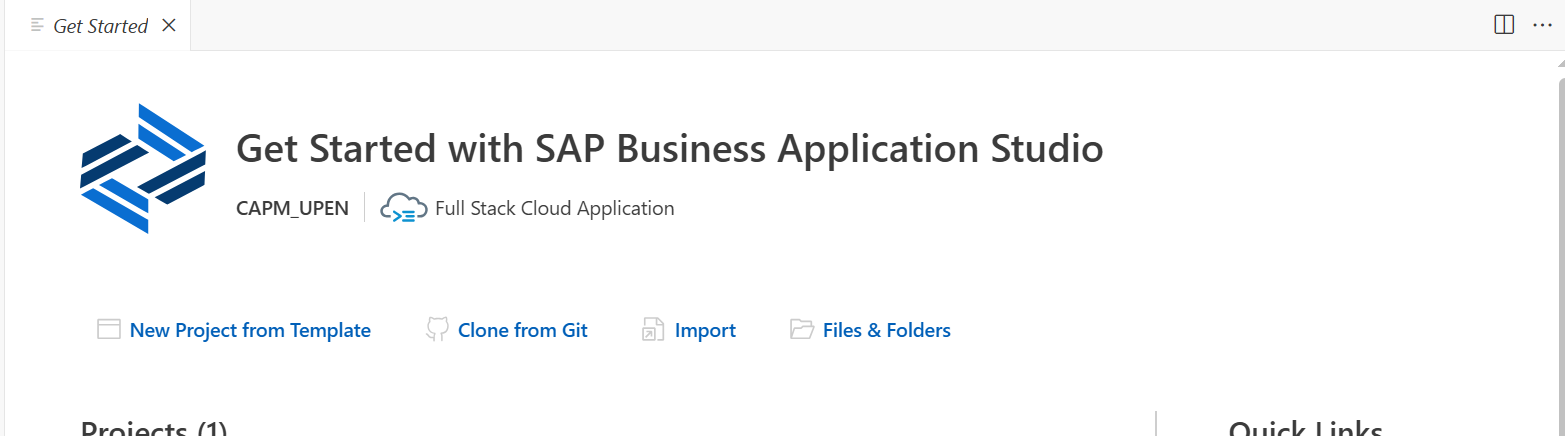
**BUILDING ANOTHER SAP CAPM APPLICATION ALONG WITH THE FIORI UI:**

**STEP-1:** click on the 3 horizontal lines 🡺Help 🡺Getstarted. ****

**STEP-2: Click on the New project from template option.**

****

**STEP-3 :**  click on the cap Project & StartA screenshot of a computer

AI-generated content may be incorrect.

**STEP-4:  
🡺**Give the details i.e project name & choose the cloud Foundry: MTA-Deployment as the path for your deployment & click on finish

A screenshot of a computer

AI-generated content may be incorrect.

**STEP-5:**  Check the files under the new cap application i.e Puchase\_Order\_Requisition.

**STEP-6:** Create a new file under the db folder named commons.cds[used for resuablity]

**A screenshot of a computer

AI-generated content may be incorrect.**

**Script for commons,cds:**namespace upen.common;

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

type Gender : String(1) enum{

    male = 'M';

    female = 'F';

    undisclosed = 'U';

};

type AmountT : Decimal(10,2)@(

    Semantics.amount.currencyCode: 'CURRENCY\_CODE',

    sap.unit:'CURRENCY\_CODE'

);

aspect Amount: {

    CURRENCY: Currency;

    GROSS\_AMOUNT: AmountT @(title : '{i18n>GrossAmount}');

    NET\_AMOUNT: AmountT @(title : '{i18n>NetAmount}');

    TAX\_AMOUNT: AmountT @(title : '{i18n>TaxAmount}');

}

type Guid: String(32);

type PhoneNumber: String(30)@assert.format : '^\s\*(?:\+?(\d{1,3}))?[-. (]\*(\d{3})[-. )]\*(\d{3})[-. ]\*(\d{4})(?: \*x(\d+))?\s\*$';

type Email: String(255)@assert.format : '^[a-zA-Z0-9.\_%+-][+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$](mailto:+@[a-zA-Z0-9.-]+\.%5ba-zA-Z%5d%7b2,%7d$)';

**STEP-7 :** Create another file under the dB folder named “**datamodel.cds”**.

**A screenshot of a computer

AI-generated content may be incorrect.**

**SCRIPT FOR DATAMODEL.CDS:**

namespace upen.db;

using { cuid, managed, temporal, Currency, Country } from '@sap/cds/common';

using { anubhav.common } from './commons';

context master {

entity businesspartner {

key NODE\_KEY: common.Guid ;

BP\_ROLE: String(2);

EMAIL\_ADDRESS: String(105);

PHONE\_NUMBER: String(32);

FAX\_NUMBER: String(32);

WEB\_ADDRESS: String(44);

ADDRESS\_GUID: Association to address;

BP\_ID: String(32);

COMPANY\_NAME: String(250) ;

}

entity address {

key NODE\_KEY: common.Guid;

CITY: String(44);

POSTAL\_CODE: String(8);

STREET: String(44);

BUILDING: String(128);

COUNTRY: String(44) @title : '{i18n>COUNTRY}';

ADDRESS\_TYPE: String(44);

VAL\_START\_DATE: Date;

VAL\_END\_DATE: Date;

LATITUDE: Decimal;

LONGITUDE: Decimal;

businesspartner: Association to one businesspartner on

businesspartner.ADDRESS\_GUID = $self;

}

entity product{

key NODE\_KEY: common.Guid;

PRODUCT\_ID: String(28);

TYPE\_CODE: String(2);

CATEGORY: String(32);

DESCRIPTION: localized String(255) ;

SUPPLIER\_GUID: Association to master.businesspartner;

TAX\_TARIF\_CODE: Integer;

MEASURE\_UNIT: String(2);

WEIGHT\_MEASURE: Decimal(5,2);

WEIGHT\_UNIT: String(2);

CURRENCY\_CODE: String(4);

PRICE:Decimal(15,2);

WIDTH:Decimal(5,2);

DEPTH:Decimal(5,2);

HEIGHT:Decimal(5,2);

DIM\_UNIT:String(2);

}

entity employees: cuid {

nameFirst: String(40);

nameMiddle: String(40);

nameLast: String(40);

nameInitials: String(40);

sex: common.Gender;

language: String(1);

phoneNumber: common.PhoneNumber;

email: common.Email;

loginName: String(12);

Currency: Currency;

salaryAmount: common.AmountT;

accountNumber: String(16);

bankId: String(8);

bankName: String(64);

}

}

context transaction {

entity purchaseorder: common.Amount, cuid{

//key NODE\_KEY: common.Guid ;

PO\_ID: String(40) ;

PARTNER\_GUID: Association to master.businesspartner ;

LIFECYCLE\_STATUS: String(1);

OVERALL\_STATUS: String(1) ;

Items: Composition of many poitems on Items.PARENT\_KEY = $self;

}

entity poitems: common.Amount, cuid{

//key NODE\_KEY: common.Guid ;

PARENT\_KEY: Association to purchaseorder;

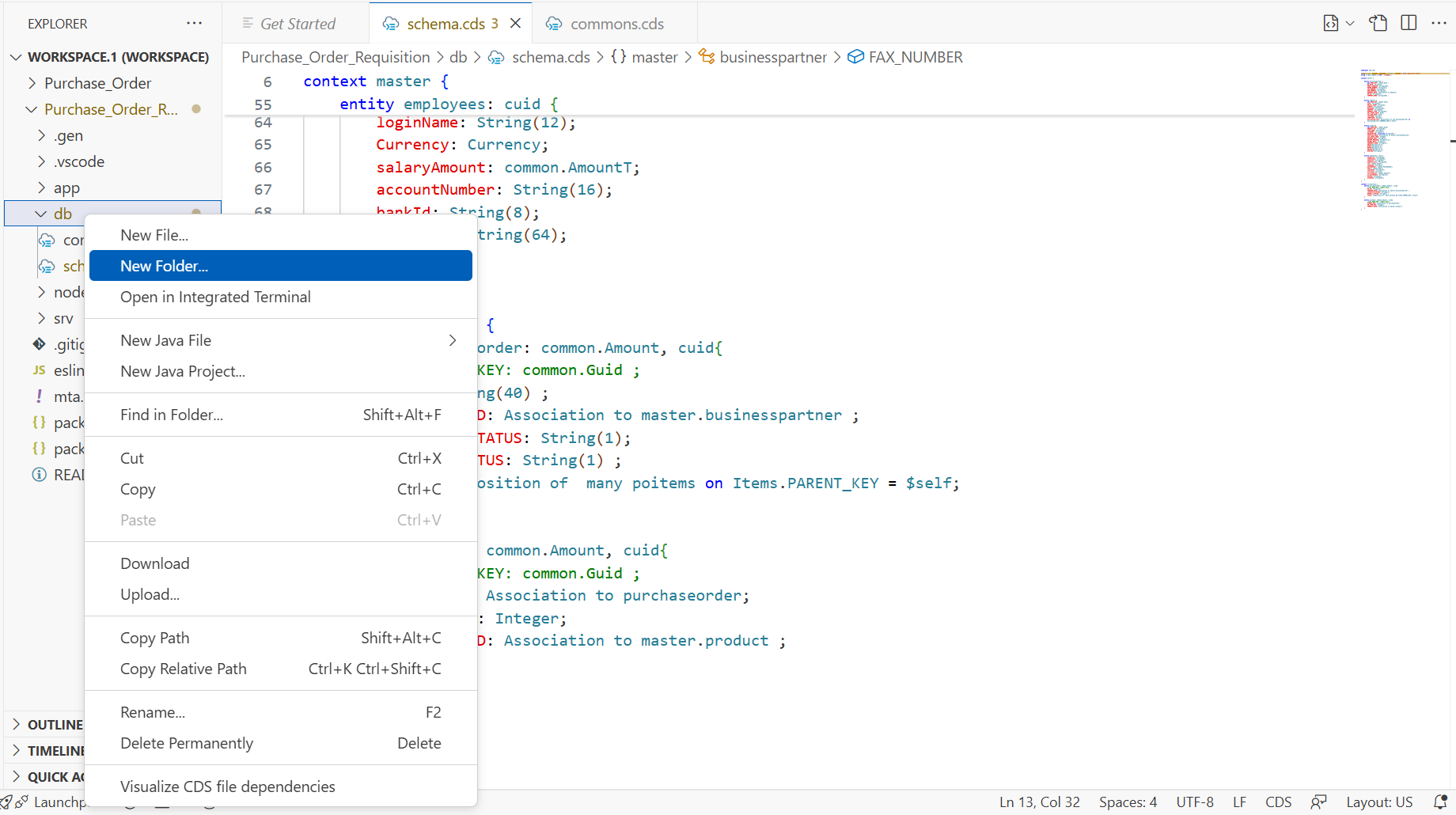
PO\_ITEM\_POS: Integer;

PRODUCT\_GUID: Association to master.product ;

}

}

**STEP-8 :** Create another folder named csv under the db folder for the insertion of records into our entities.



🡺Create the csv files for the database records.

**STEP-9:** Open the terminal and select the Purchase\_Order\_Requisition

A screenshot of a computer

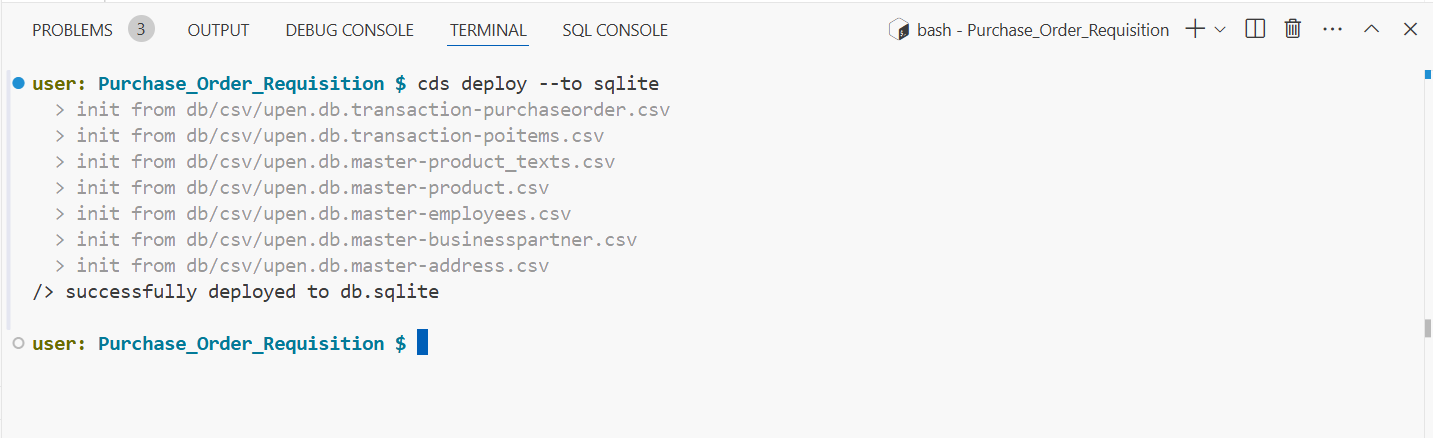
AI-generated content may be incorrect.

🡺Select the Purchase\_Order-Requisition from the drop-down menu.

A screenshot of a computer

AI-generated content may be incorrect.

🡺Give the command in the terminal 🡺cds deploy -–to sqlite.



🡺Copy the relative path of the db.sqlite file generated.

A screenshot of a computer

AI-generated content may be incorrect.

🡺click on the 3 dotted lines and select sql tools.A screenshot of a computer

AI-generated content may be incorrect.

🡺Click on add new connection & give the details i.e.

1)Name of the connection ,2) Relative Path in the Database-file column.

🡺Test & Save Connection.

A screenshot of a computer

AI-generated content may be incorrect.

🡺Click on connect now.

A screenshot of a computer

AI-generated content may be incorrect.

🡺Click on tables to see the records in the SQLite databse.

A screenshot of a computer

AI-generated content may be incorrect.

🡺**STEP-10 : create a new file under the srv i.e CatalogService.cds.**

A screenshot of a computer

AI-generated content may be incorrect.

**Script for the CatalogService.cds:**

**Note : @(odata.draft.enabled: true) this command is used to enable the draft state of the records and also ODATA-CRUD Operations out of the box from the sap capm framework.**

**using { upen.db.master, upen.db.transaction } from '../db/datamodel';**

**service CatalogService @(path: 'CatalogService') {**

**entity ProductSet as projection on master.product;**

**entity BusinessPartnerSet as projection on master.businesspartner;**

**entity BusinessAddressSet as projection on master.address;**

**// @readonly**

**entity EmployeeSet as projection on master.employees;**

**//@Capabilities : { Deletable: false }**

**entity POs @(odata.draft.enabled: true) as projection on transaction.purchaseorder{**

**\*,**

**Items,**

**case OVERALL\_STATUS**

**when 'P' then 'Pending'**

**when 'N' then 'New'**

**when 'A' then 'Approved'**

**when 'X' then 'Rejected'**

**end as OverallStatus : String(10),**

**case OVERALL\_STATUS**

**when 'P' then 2**

**when 'N' then 2**

**when 'A' then 3**

**when 'X' then 1**

**end as ColorCode : Integer,**

**}**

**actions{**

**@Common.SideEffects : {**

**TargetProperties : [**

**'in/GROSS\_AMOUNT',**

**]**

**}**

**action boost() returns POs**

**};**

**function largestOrder() returns array of  POs;**

**//definition of the function**

**function getOrderDefaults() returns POs;**

**entity POItems as projection on transaction.poitems;**

**}**

**🡺Create another file under the srv folder named CatalogService.js  
  
NOTE: The file name should be same as the file created before i.e CatalogService.cds and now we need need to create the CatalogService.js [case sensitive].  
  
🡺In the CatalogService.cds file we define the functionalities and in the CatalogService.js we write the implementation of our application.**

A screenshot of a computer

AI-generated content may be incorrect.

**Script for the catalogservice.js:**

**🡺The custome logic is written in the service.js file.**

**module.exports = cds.service.impl( async function(){**

**//Step 1: get the object of our odata entities**

**const { EmployeeSet, POs } = this.entities;**

**this.before('UPDATE', EmployeeSet, (req) => {**

**var salary = parseInt(req.data.salaryAmount);**

**if(salary >= 1000000){**

**req.error(500,"sorry no one can get this salary in my org");**

**}**

**});**

**this.after('READ', EmployeeSet, (data) => {**

**//res.data.push({"ID":"DUMMY"});**

**//return req.data;**

**data.push({**

**"ID": "CUSTOM",**

**"nameFirst": "Munna"**

**});**

**});**

**this.on('boost', async (req,res) => {**

**try {**

**//since its instance bound we will get the key of PO**

**const ID = req.params[0];**

**//Print on console the key**

**console.log("Hey Amigo, Your purchase order with id " + req.params[0] + " will be boosted");**

**//Start a db transaction suing cds ql - https://cap.cloud.sap/docs/node.js/cds-tx**

**const tx = cds.tx(req);**

**//UPDATE dbtab set grossamount = current + 20k WHERE ID = key**

**await tx.update(POs).with({**

**GROSS\_AMOUNT: { '+=' : 20000 }**

**}).where(ID);**

**} catch (error) {**

**return "Error " + error.toString();**

**}**

**});**

**this.on('getOrderDefaults', async (req,res) => {**

**return {**

**"OVERALL\_STATUS": "N",**

**"GROSS\_AMOUNT": 500,**

**"CURRENCY\_code": "EUR"**

**};**

**});**

**this.on('largestOrder', async (req,res) => {**

**try {**

**const tx = cds.tx(req);**

**var reply = [];**

**//SELECT \* UPTO 1 ROW FROM dbtab ORDER BY GROSS\_AMOUNT desc**

**reply = await tx.read(POs).orderBy({**

**GROSS\_AMOUNT: 'desc'**

**}).limit(2);**

**// const reply = await srv.run( SELECT.from(POs).orderBy({**

**// GROSS\_AMOUNT: 'desc'**

**// }).limit(1) )**

**return reply;**

**} catch (error) {**

**return "Error " + error.toString();**

**}**

**});**

**}**

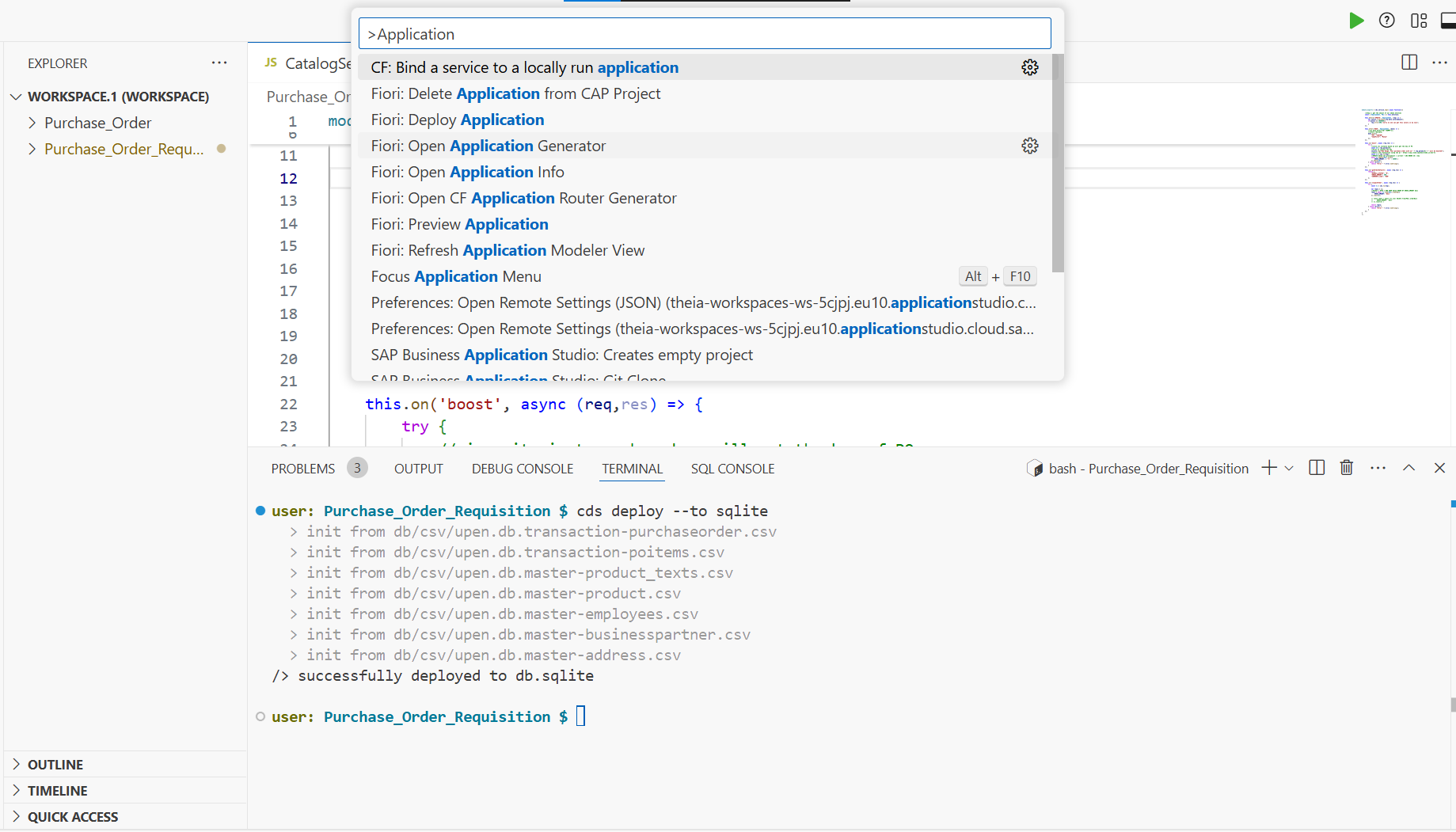
**);**

**NOTES:**

**🡺Till now we have worked on the backend of our application now we work on the front-end i.e building Fiori app.**

**GENERATING THE SAP-FIORI APP:**

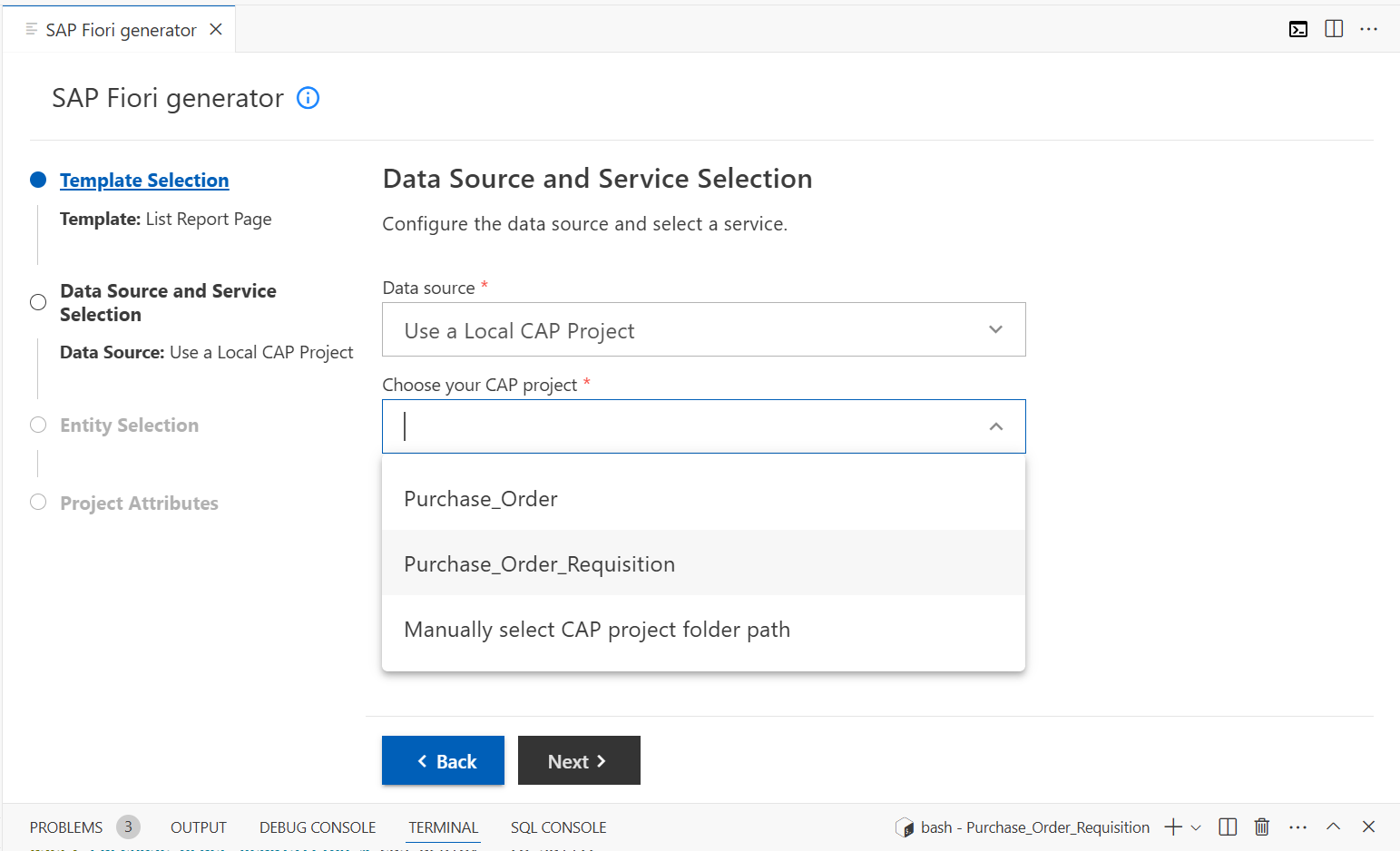
**🡺Open the above search bar and do search for the “Fiori:Open Application Generator”**



**🡺Click on the Fiori Open Application Generator.**

A screenshot of a computer

AI-generated content may be incorrect.



A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**🡺Check the files created under annotations.cds file under the app folder.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**🡺Open the annotations.cds file and write the below code for designing the Fiori ui.**

**SCRIPT FOR THE annotations.cds file:  
  
using CatalogService as service from '../../srv/CatalogService';**

**annotate service.POs with @(**

**Common.DefaultValuesFunction : 'getOrderDefaults',**

**UI.SelectionFields:[**

**PO\_ID,**

**PARTNER\_GUID.COMPANY\_NAME,**

**GROSS\_AMOUNT,**

**OVERALL\_STATUS**

**],**

**UI.LineItem:[**

**{**

**$Type : 'UI.DataField',**

**Value : PO\_ID,**

**},**

**{**

**$Type : 'UI.DataField',**

**Value : PARTNER\_GUID.COMPANY\_NAME,**

**},**

**{**

**$Type : 'UI.DataField',**

**Value : PARTNER\_GUID.ADDRESS\_GUID.COUNTRY,**

**},**

**{**

**$Type : 'UI.DataField',**

**Value : GROSS\_AMOUNT,**

**},**

**{**

**$Type : 'UI.DataFieldForAction',**

**Action : 'CatalogService.boost',**

**Label: 'boost',**

**Inline: true**

**},**

**{**

**$Type : 'UI.DataField',**

**Value : OverallStatus,**

**Criticality: ColorCode**

**},**

**],**

**UI.HeaderInfo:{**

**TypeName: 'Purchase Order',**

**TypeNamePlural: 'Purchase Orders',**

**Title: {Value : PO\_ID},**

**Description: {Value : PARTNER\_GUID.COMPANY\_NAME}**

**},**

**UI.Facets:[**

**{**

**$Type : 'UI.CollectionFacet',**

**Label: 'General Information',**

**Facets : [**

**{**

**$Type : 'UI.ReferenceFacet',**

**Label: 'Order Details',**

**Target : '@UI.Identification'**

**},**

**{**

**$Type : 'UI.ReferenceFacet',**

**Label: 'Configuration Details',**

**Target : '@UI.FieldGroup#Spiderman'**

**},**

**],**

**},**

**{**

**$Type : 'UI.ReferenceFacet',**

**Label: 'PO Items',**

**Target : 'Items/@UI.LineItem',**

**},**

**],**

**UI.Identification:[**

**{**

**$Type : 'UI.DataField',**

**Value : PO\_ID,**

**},**

**{**

**$Type : 'UI.DataField',**

**Value : PARTNER\_GUID\_NODE\_KEY,**

**},**

**{**

**$Type : 'UI.DataField',**

**Value : OVERALL\_STATUS,**

**},**

**],**

**UI.FieldGroup #Spiderman: {**

**Label : 'PO pricing',**

**Data : [**

**{**

**$Type : 'UI.DataField',**

**Value : GROSS\_AMOUNT,**

**},**

**{**

**$Type : 'UI.DataField',**

**Value : NET\_AMOUNT,**

**},**

**{**

**$Type : 'UI.DataField',**

**Value : TAX\_AMOUNT,**

**},**

**{**

**$Type : 'UI.DataField',**

**Value : CURRENCY\_code,**

**},**

**],**

**}**

**);**

**annotate service.POItems with @(**

**UI.LineItem:[**

**{**

**$Type : 'UI.DataField',**

**Value : PO\_ITEM\_POS,**

**},**

**{**

**$Type : 'UI.DataField',**

**Value : PRODUCT\_GUID\_NODE\_KEY,**

**},**

**{**

**$Type : 'UI.DataField',**

**Value : GROSS\_AMOUNT,**

**},**

**{**

**$Type : 'UI.DataField',**

**Value : CURRENCY\_code,**

**},**

**{**

**$Type : 'UI.DataField',**

**Value : PRODUCT\_GUID.CATEGORY,**

**},**

**],**

**UI.HeaderInfo:{**

**TypeName : 'PO Item',**

**TypeNamePlural: 'PO Items',**

**Title : {Value: PO\_ITEM\_POS},**

**Description: {Value: PRODUCT\_GUID.DESCRIPTION}**

**},**

**UI.Facets:[**

**{**

**$Type : 'UI.ReferenceFacet',**

**Label: 'More Info',**

**Target : '@UI.Identification',**

**},**

**],**

**UI.Identification:[**

**{**

**$Type : 'UI.DataField',**

**Value : PO\_ITEM\_POS,**

**},**

**{**

**$Type : 'UI.DataField',**

**Value : PRODUCT\_GUID\_NODE\_KEY,**

**},**

**{**

**$Type : 'UI.DataField',**

**Value : CURRENCY\_code,**

**},**

**{**

**$Type : 'UI.DataField',**

**Value : GROSS\_AMOUNT,**

**},**

**{**

**$Type : 'UI.DataField',**

**Value : NET\_AMOUNT,**

**},**

**{**

**$Type : 'UI.DataField',**

**Value : TAX\_AMOUNT,**

**}**

**]**

**);**

**annotate service.POs with {**

**PARTNER\_GUID@(**

**Common : {**

**Text : PARTNER\_GUID.COMPANY\_NAME,**

**},**

**ValueList.entity: CatalogService.BusinessPartnerSet**

**);**

**OVERALL\_STATUS @(**

**readonly,**

**)**

**};**

**annotate service.POItems with {**

**PRODUCT\_GUID@(**

**Common : {**

**Text : PRODUCT\_GUID.DESCRIPTION,**

**},**

**ValueList.entity: CatalogService.ProductSet**

**)**

**};**

**@cds.odata.valuelist**

**annotate service.BusinessPartnerSet with @(**

**UI.Identification:[{**

**$Type : 'UI.DataField',**

**Value : COMPANY\_NAME,**

**}]**

**);**

**@cds.odata.valuelist**

**annotate service.ProductSet with @(**

**UI.Identification:[{**

**$Type : 'UI.DataField',**

**Value : DESCRIPTION,**

**}]**

**);**

**🡺run the cds run command in the terminal and click on the “Open in a New Tab” pop up.**

**🡺You can also use the “cds watch” command in the terminal to run the app i.e the cds watch command when you make changes in the application while it is opened.**

**A screenshot of a computer

AI-generated content may be incorrect.**

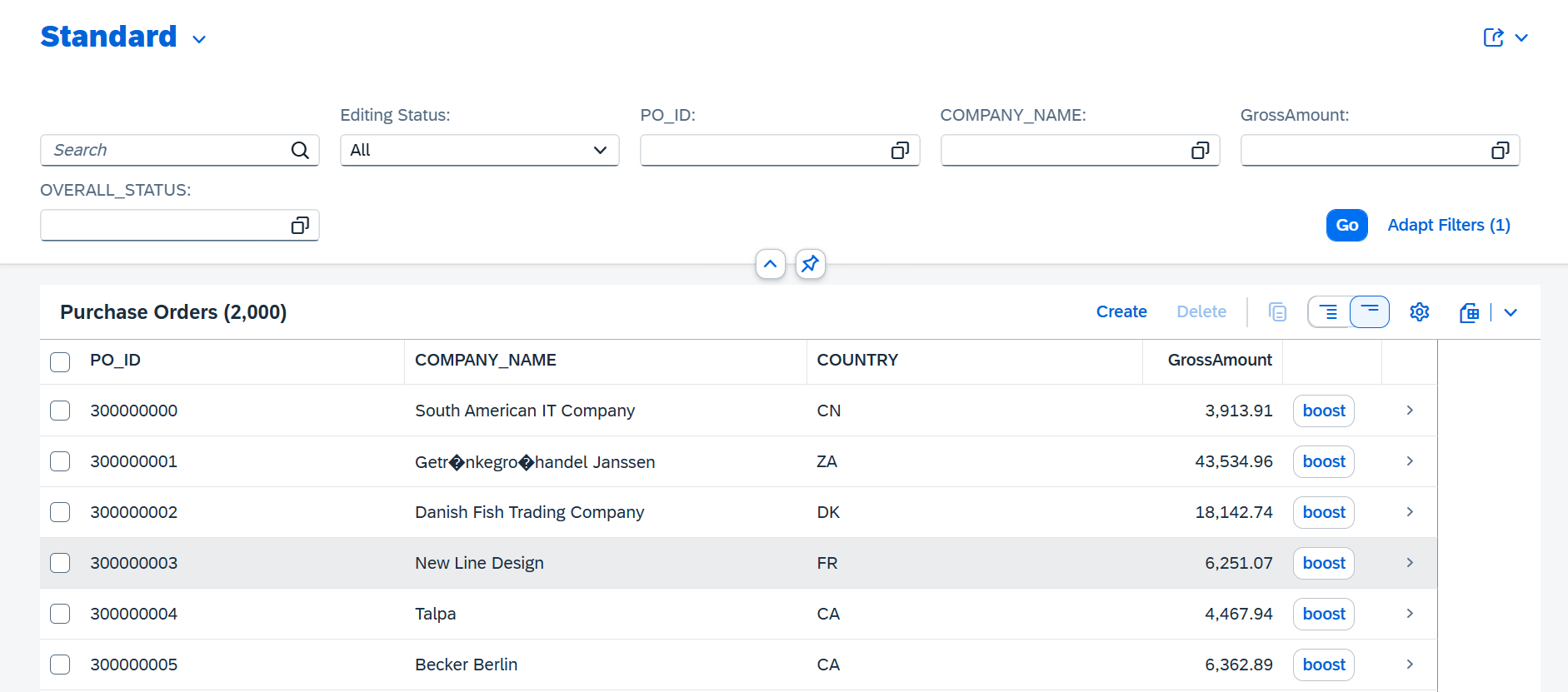
**🡺Click on the webapp that you have created.**

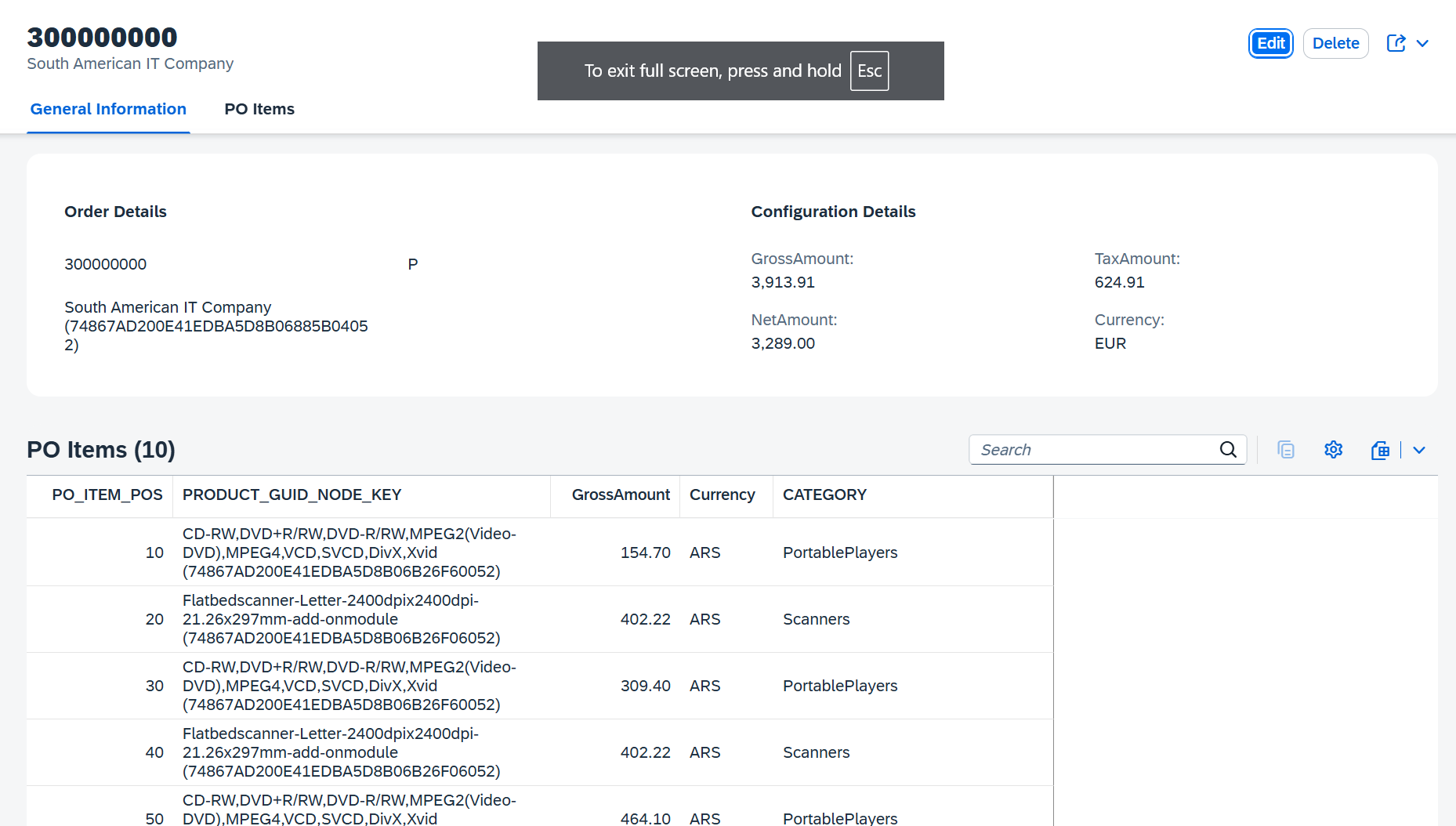
**A screenshot of a computer

AI-generated content may be incorrect.**

**🡺The ui takes time to load and it opens.**

**🡺Click on the any record to see the further drill-down.**

****

****