**Step 1. Hello World**

Index.html: Contine 2 sectiuni principale: Head si Body.

Head: e folosit de browser pt a procesa documentul, prin intermediul tagurilor meta

Body: aici se descrie aspectul paginii (putem pune paragrafe poze etc) tot cu ajutorul tagurilor(p, img)

*Pt rulare folosim: ui5 serve*

Port:8080 si trb sa avem “/document”

**Step 2. Bootstrap**

Adaugam framework ul SAPUI5 si modulele de baza.

Pt asta folosim tag-ul script in Head. In interiorul tag-ului folosim attribute precum:

<script>

src->spune browserului unde gaseste libraria SAPUI5

theme-> alegem tema

libs-> alegem libraria care are controle UI de un anumit tip(noi avem m)

compactVersion-> pt cea mai recenta versiune de SAPUI5, folosim “edge”

async-> setam pe true ca sa ruleze asincron( adica sa faca lucrurile in paralel, mai multe fire de executie)-> incarcarea simultana a resurselor pe ecran !!!

onInit-> modul prin care se Evita scrierea de cod js executabil direct in fisierul HTML(Pt Securitate) II SPUNEM UNDE SA GASEASCA FISIERUL ( II DAM RUTA) ULTIMUL E FISIERUL NOSTRU

resourceroots-> specificam ca resursele sunt localizate in acelasi folder ca index.html prin “./”

Cream fisierul index.js-> in el se gaseste logica. Este apelat in scipts

Sap.ui.define-> defineste un module cu un propriile id si dependinte care vor fi incarcate async

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

**Step 3. Controls**

**Controlele-folosite pt a define aspectul si comportamentul unor parti ale paginii**

In index.html la body stergem textul si punem sapUiBody-> adauga diferite teme pt stil

Dam id=”content”

A screenshot of a computer

Description automatically generated with medium confidence

**Step 4. XML Views**

**XML VIEWS- ofera modalitate prin care sa modularizam controalele folosite( adica sa separam)**

**Text

Description automatically generated**

App.view.xml-responsabil pt instantierea lui text control(la pasaul 3 era instantiate in index.js)

Referentiaza sap.m-> adica libraria.. sap.m e default namespace astfel putem folosi Text control

Text

Description automatically generated

Text

Description automatically generated

**Step 5. Controllers**

**Controllers-js files in ui5 app in care pui logica pt ui … practic fisierele controllers implementeaza logica pt XML VIEW**

Text

Description automatically generated

Graphical user interface, text, application

Description automatically generated

controllerName->ce controller sa foloseasca.. sap.ui….-> controller e numele folderului iar App numele controllerului

Button-> ce text sa afiseze si event press

Text

Description automatically generated

onShowHello-handler function

Text

Description automatically generated

**Step 6. Modules**

**Modules- orice js file care poate fi incarcat si executat de browser**

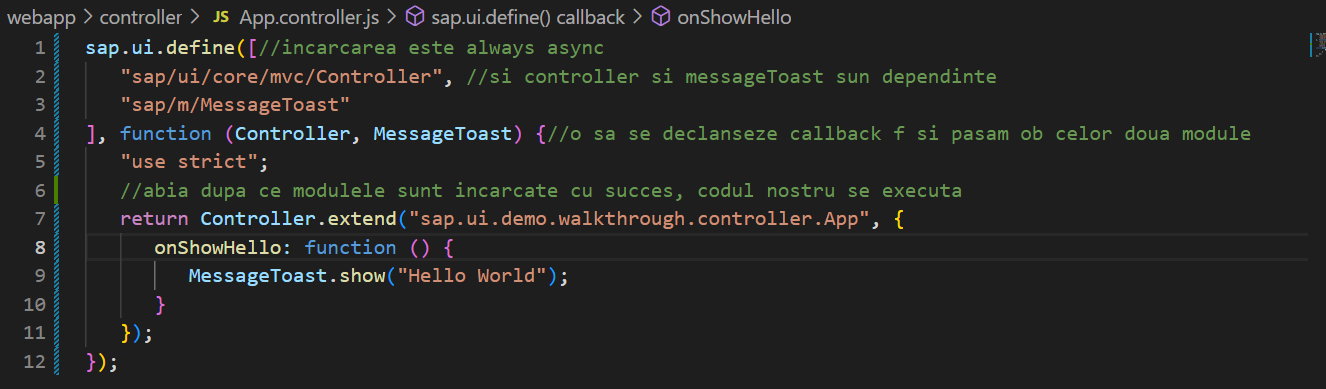
* **Poate continue orice tip de cod, nu are restrictii de genul**

**SE REFERA SA IMPARTIM CODUL IN MAI MULTE FISIERE(MODULE), SA NU AVEA UN SG FISIER CU O TONA DE COD)**

DE RETINUT!!

Fisierul App.controller.js este in sine un modul pt ca avem define

Si de asemenea ne folosim de modulele Controller si MessageToast(linia 2 si 3)



Text

Description automatically generated

1. Se refera ca putem folosi fisierul(modulul) peste tot in app

**Step 7. JSON Model**

**JSON Model- client side model, doar tine date, nu poate trimite date updatate, trb ca diferite metode din app sa faca asta**

Text

Description automatically generated

IN INPUT

Text

Description automatically generated

**Step 8. Translatable Texts**

**Avem nev de fisierul i18n**

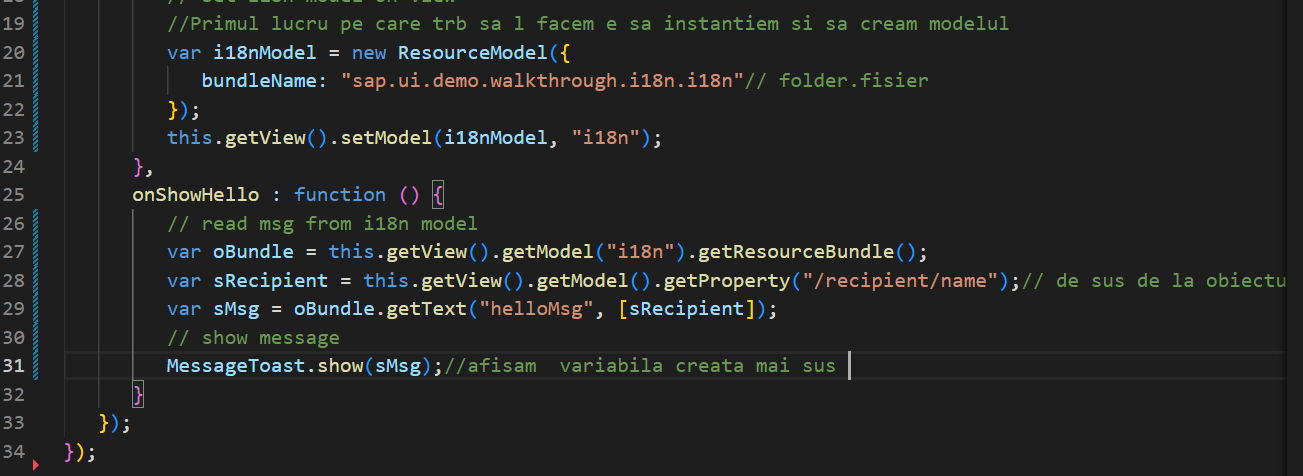
**Graphical user interface, text, application, chat or text message

Description automatically generated**

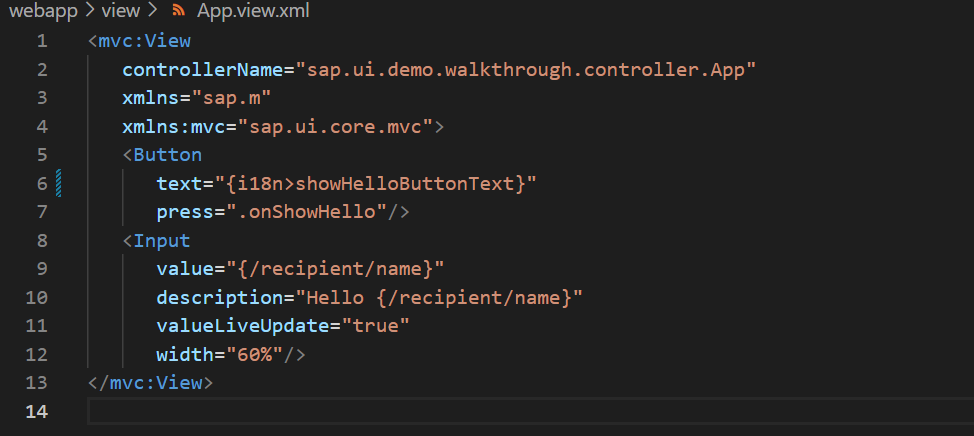
**Apoi in controller.. adaugam un nou modul ResourceModel**

****

**Facem ca la celelalte**

****

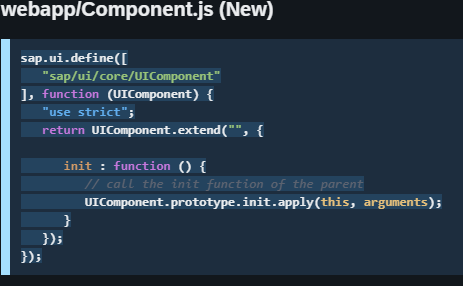
**Ca si butonul sa aiba un text din fifsierul i18n**

****

**Step 9. Component Configuration**

**Components- provide independent si reuseable parts in app**

**Creare componenta**

****

**Initial**

**Apoi**

**Text

Description automatically generated**

**Dupa in controller scoatem fct de onInit pt ca deja am pus o in componenta**

**Text

Description automatically generated**

**Schimbam si index.js pt ca acum folosim Comp, nu View**

**Text

Description automatically generated**

**Step 10. Descriptor for Applications**

**Step 11. Pages and Panels**

**App->pages->Page->content->Panel->B+I**

**In View adaugam un control de tip Page(sap.m)** apoi content, etc.. si punem Button si Input pe care deja le avem

Titlul paginii il setam tot de la Page(ca un css)

Propo display block=true-> e pt height view-ului

In pages putem sa avem mai multe Page

Panel-> folosit pt a grupa content similar

**Step 12. Shell Control as Container**

**Shell Container- se ocupa cu adaptarea vizuala a aplicatiei la dimensiunea ecranului**

**Se pune ca un root( In view, ca tree-ul de la react), punem App de la step 11 in el**

**Shell-> face letter box Graphical user interface, text, application, email

Description automatically generated**

**Inainte barile din dr si st nu erau, nici dunga deasupra**

**Step 13. Margins and Paddings**

**Adaugam in panel-  si width auto**

**La fel si la restul (Button, Input)**

**Introducem TextControl(ultimul)**

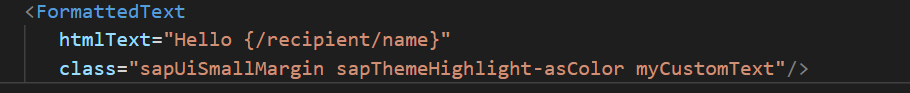
**Step 14. Custom Css si theme colors**

**Facem folder css cu fisier style.css**

**Text

Description automatically generated**

**Punem in manifest (resources)**

**In view am pus (ca in css facut) si** myCustoMtext la fel ****

**Step 15. Nested Views**

Practic pana acum aveam un sg view pt toata aplicatia. Acum am mai facut unul pt panel si am dat copy paste din primul in al doilea la partea de <Panel>

Facem la fel si la Partea de controller

Text

Description automatically generated with medium confidence Text

Description automatically generated

**Step 16. Dialogs and Fragments**

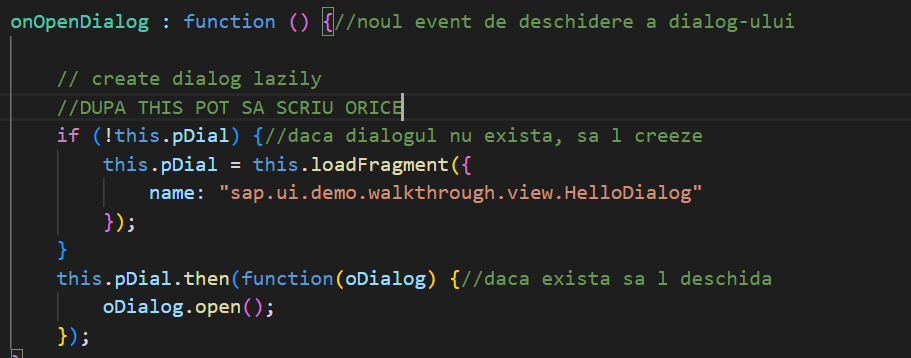
Cu fragmente putem face view-uri

FrAGMENTELE n au controllet ...cu ele cream reusable

Cream uj buton prin sa sa deschidem Dialog-ul prin eventHandler

Facem Dialogul in view si e de tip fragment

Acum facem eventhandlerul pt deschidere in Controller-ul Panelului



**Step 17. Fragments CallBacks**

Adaugam buton de inchider al dialogului

Adaugam functionalitatea in HelloPanel

**A screenshot of a computer

Description automatically generated with medium confidence**

Apoi cream butonul care o sa fie in dialog

**Graphical user interface, text, application

Description automatically generated**

Putem avea beginButton sau EndButton

**Step 18. Icon**

In ambele viewuri

**Step 19. Aggregation Binding**

Trb sa adaugam in manifest **Graphical user interface, text, website

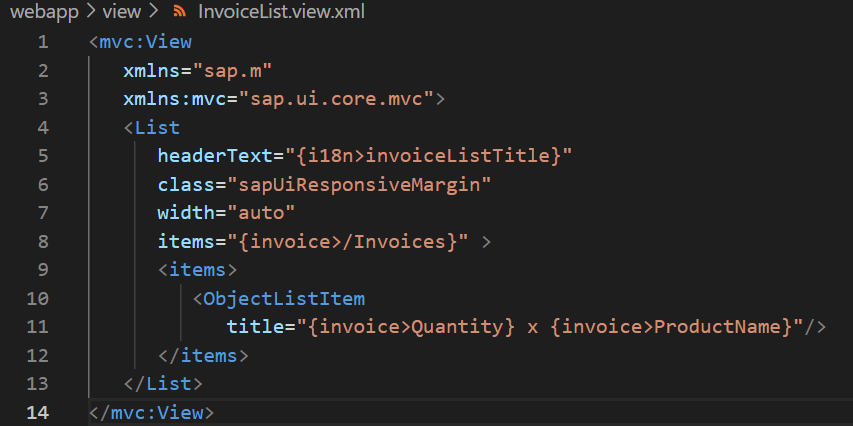
Description automatically generated**

Uri=calea

componenta noastră va instanția automat un nou JSONModel care încarcă datele facturii din fișierul Invoices.json

Mai facem un view nou in App ca sa punem facturile Text

Description automatically generated

Si acum ii facem propriul view 

**Step 20. Data Types**

**Text

Description automatically generated**

Deci in OLI, title, number, etc sunt proprietati deja definite

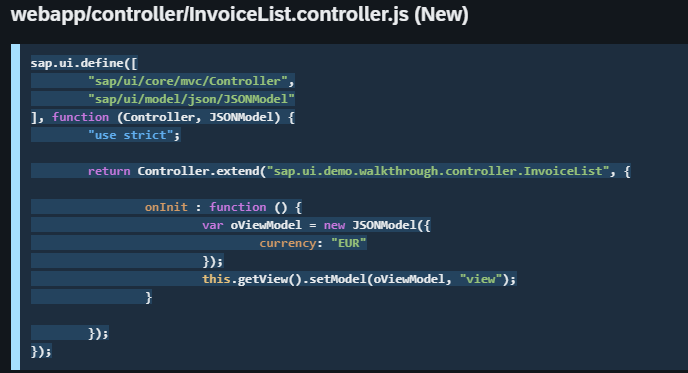
La path: specificam calea valorii cu care vrem sa lucram, iar currency ul vine din InvoiceListController

formatOptions ->showMeasure-> ar fi aratat EUR FIX LANGA

CU EL FALSE-> A picture containing text

Description automatically generated

Apoi am facut controllerul pt InvoiceList unde am facut si currency ul

****

**Step 21. Expression Binding**

Expression binding- useful adding little components of logic to your various attrubutes in ui5 controls to determine pif smth needs to happend

ex: o expresie ca sa zica daca preul e mai mare de 50 sa ne faca rosu si daca nu verde

**In OBI PUNEM**

****

Citire: daca val extendedPrice din Factura respectiva e mai mare decat 50->Error, daca nu->success

Culorile se pun automat