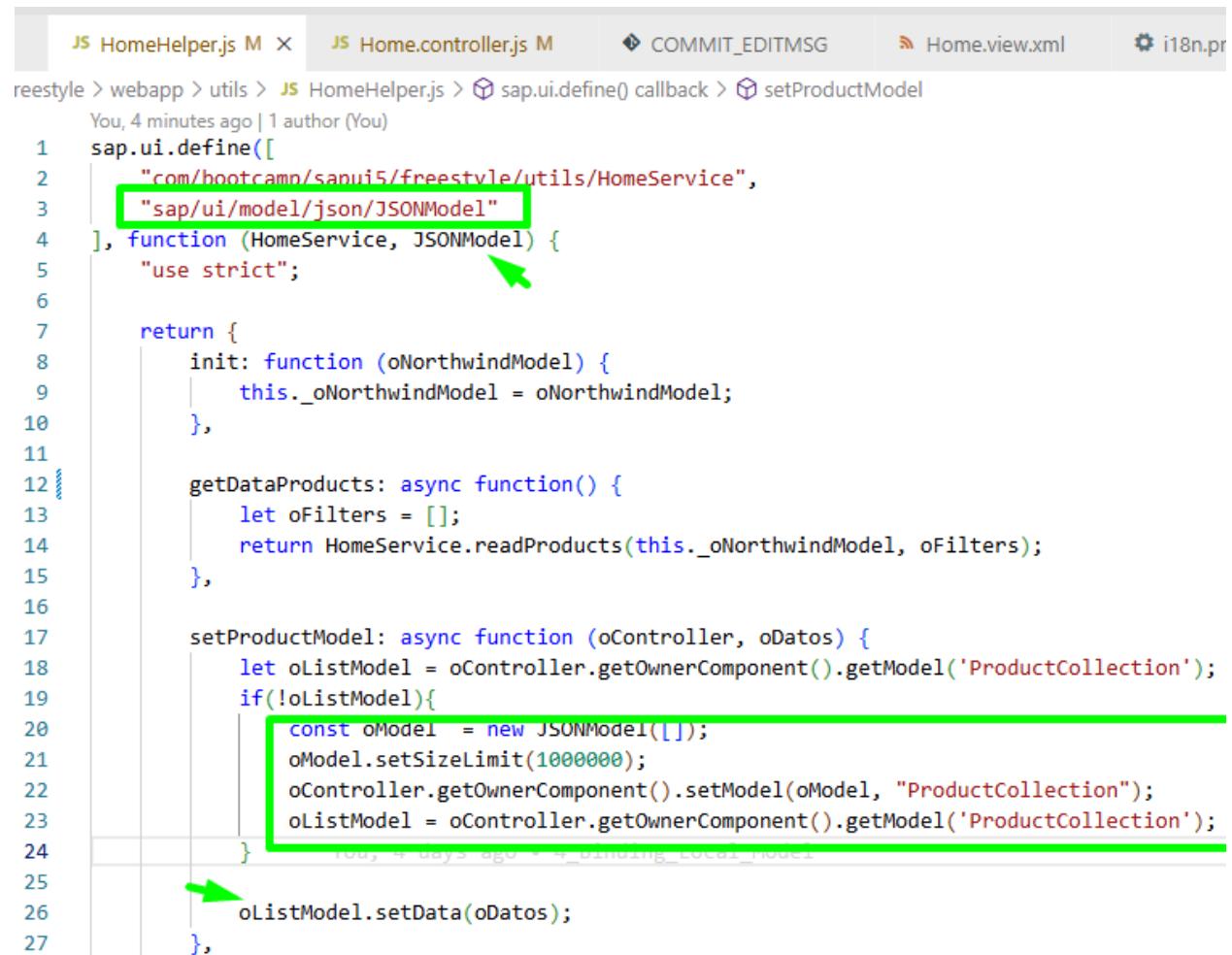


## Solicitud de Productos (Parte 2)

Ahora vamos a crear un modelo local para asignar la lista de productos previamente consultados desde el botón Buscar.

Agregamos la función **setProductModel** en el Helper **HomeHelper.js**:



```

JS HomeHelper.js M X JS Home.controller.js M COMMIT_EDITMSG Home.view.xml i18n.pr
reestyle > webapp > utils > JS HomeHelper.js > sap.ui.define() callback > setProductModel
You, 4 minutes ago | 1 author (You)
1 sap.ui.define([
2   "com/hootcamp/sanui5/freestyle/utils/HomeService",
3   "sap/ui/model/json/JSONModel"
4 ], function (HomeService, JSONModel) {
5   "use strict";
6
7   return {
8     init: function (oNorthwindModel) {
9       this._oNorthwindModel = oNorthwindModel;
10      },
11
12      getDataProducts: async function() {
13        let oFilters = [];
14        return HomeService.readProducts(this._oNorthwindModel, oFilters);
15      },
16
17      setProductModel: async function (oController, oDatos) {
18        let oListModel = oController.getOwnerComponent().getModel('ProductCollection');
19        if(!oListModel){
20          const oModel = new JSONModel([ ]);
21          oModel.setSizeLimit(1000000);
22          oController.getOwnerComponent().setModel(oModel, "ProductCollection");
23          oListModel = oController.getOwnerComponent().getModel('ProductCollection');
24        }
25        oListModel.setData(oDatos);
26      },
27    }
28  }
29
30  You, 4 days ago * 4 Binding Local model

```

En este punto usamos la librería [\*\*JSONModel\("sap/ui/model/json/JSONModel"\)\*\*](#)

JavaScript

```

setProductModel: async function (oController, oDatos) {
  let oListModel =
oController.getOwnerComponent().getModel('ProductCollection');

```

```

if(!oListModel){
    const oModel = new JSONModel([]);
    oModel.setSizeLimit(1000000);
    oController.getOwnerComponent().setModel(oModel, "ProductCollection");
    oListModel =
oController.getOwnerComponent().getModel('ProductCollection');
}

oListModel.setData(oDatos);
},

```

Ahora Invocamos la función desde el controlador de la vista **Home.controller.js**:

```

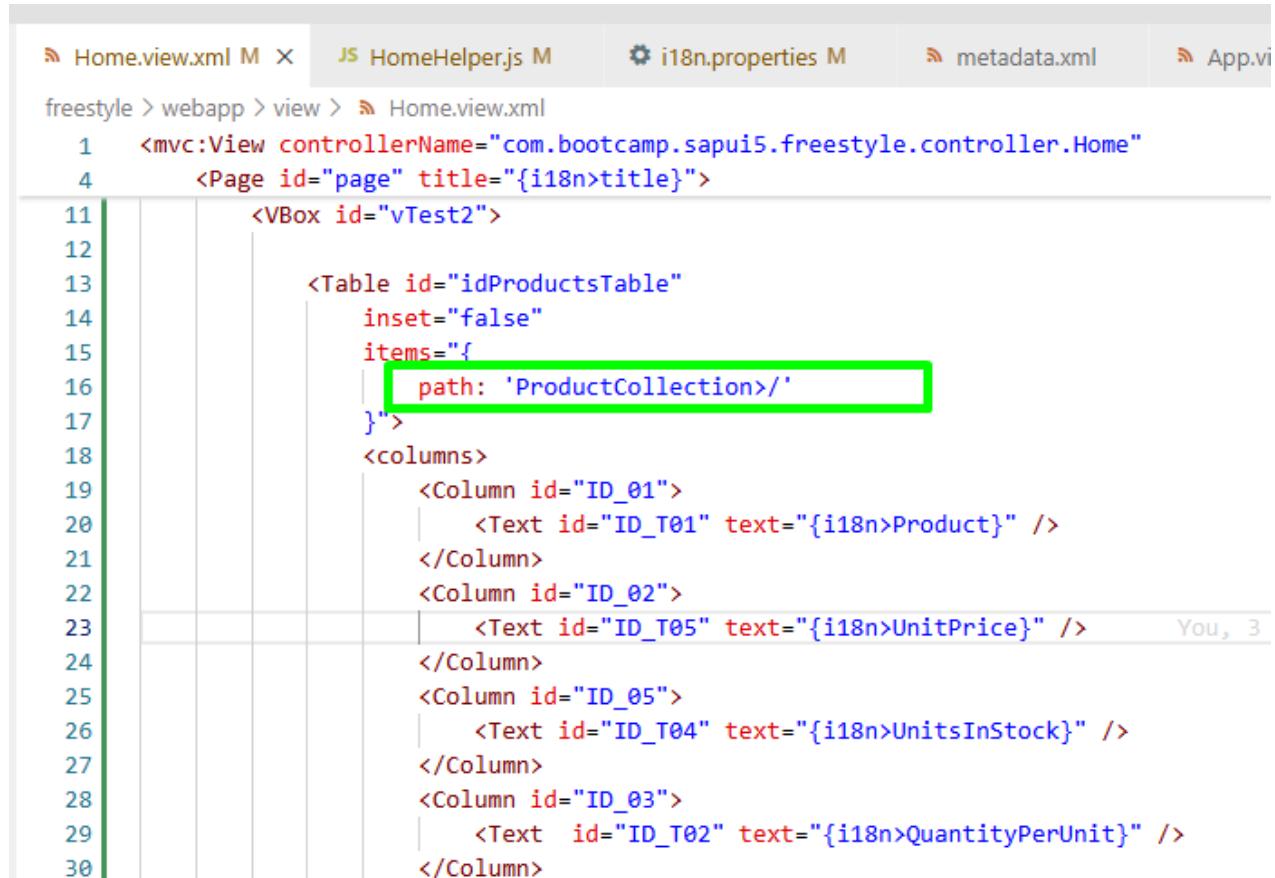
EXPLORER           ... /le   JS Home.controller.js M X JS HomeHelper.js M COMMIT_EDITM
BOOTCAMP-SAP-FIORI
> .vscode
< freestyle
  > node_modules
< webapp
  < controller
    JS App.controller.js
JS Home.controller.js M
  > CSS
  < i18n
    i18n.properties
  < localService
    metadata.xml
  > model
  > test

freestyle > webapp > controller > JS Home.controller.js > sap.ui.define() callback
You, 6 minutes ago | 1 author (You)
1 sap.ui.define([
2   "sap/ui/core/mvc/Controller",
3   "com/bootcamp/sapui5/freestyle/utils/HomeHelper"
4 ], [Controller, HomeHelper] => {
5   "use strict";
6
7   return Controller.extend("com.bootcamp.sapui5.freesty
8     OnInit() {
9       },
10
11     onPress: async function(){
12       let oDatos = await HomeHelper.getDataProducts(
13         await HomeHelper.setProductModel(this, oDatos
14       );
15     });
16   ...

```

Ahora agregamos en la vista Home una [tabla sap.m.table](#):

A esta tabla le asignamos en la propiedad items el modelo local creado previamente ([ProductCollection](#))



```
freestyle > webapp > view > Home.view.xml
1   <mvc:View controllerName="com.bootcamp.sapui5.freestyle.controller.Home"
4       <Page id="page" title="{i18n>title}">
11      <VBox id="vTest2">
12          <Table id="idProductsTable"
13              inset="false"
14              items="{
15                  path: 'ProductCollection'/' You, 3
16              }">
17              <columns>
18                  <Column id="ID_01">
19                      <Text id="ID_T01" text="{i18n>Product}" />
20                  </Column>
21                  <Column id="ID_02">
22                      <Text id="ID_T05" text="{i18n>UnitPrice}" />
23                  </Column>
24                  <Column id="ID_05">
25                      <Text id="ID_T04" text="{i18n>UnitsInStock}" />
26                  </Column>
27                  <Column id="ID_03">
28                      <Text id="ID_T02" text="{i18n>QuantityPerUnit}" />
29                  </Column>
30          </columns>

```

Se hace el binding con las diferentes columnas configuradas así:

freestyle > webapp > view > Home.view.xml

```

1  <mvc:View controllerName="com.bootcamp.sapui5.freestyle.controller.Home"
4      <Page id="page" title="{i18n>title}">
11     <VBox id="vTest2">
13         <Table id="idProductsTable"
51             </columns>
32             <items>
33                 <ColumnListItem id="ID_06" vAlign="Middle">
34                     <cells>
35                         <ObjectIdentifier id="ID_07"
36                             title="{ProductCollection>ProductID}"
37                             text="{ProductCollection>ProductName}" />
38                         <ObjectNumber id="ID_11"
39                             number={
40                                 parts:[{path:'ProductCollection>UnitPrice'}, {pat
41                                     type: 'sap.ui.model.type.Currency',
42                                     formatOptions: {showMeasure: true}
43                                 }]
44                                 unit="USD" />
45                         <Text id="ID_09"
46                             text="{ProductCollection>UnitsInStock}" />
47                         <Text id="ID_08"
48                             text="{ProductCollection>QuantityPerUnit}" />
49                     </cells>
50                 </ColumnListItem>
51             </items>
52         </Table>
53     </VBox>
54 </Page>
55 </mvc:View>
```

Java

```

<Table id="idProductsTable"
    inset="false"
    items="{
        path: 'ProductCollection>'
    }">
    <columns>
        <Column id="ID_01">
            <Text id="ID_T01" text="{i18n>Product}" />
```

```

        </Column>
        <Column id="ID_02">
            <Text id="ID_T05" text="{i18n>UnitPrice}" />
        </Column>
        <Column id="ID_05">
            <Text id="ID_T04" text="{i18n>UnitsInStock}" />
        </Column>
        <Column id="ID_03">
            <Text id="ID_T02" text="{i18n>QuantityPerUnit}" />
        </Column>
    </columns>
    <items>
        <ColumnListItem id="ID_06" vAlign="Middle">
            <cells>
                <ObjectIdentifier id="ID_07"
                    title="{ProductCollection>ProductID}"
                    text="{ProductCollection>ProductName}" />
                <ObjectNumber id="ID_11"
                    number="{

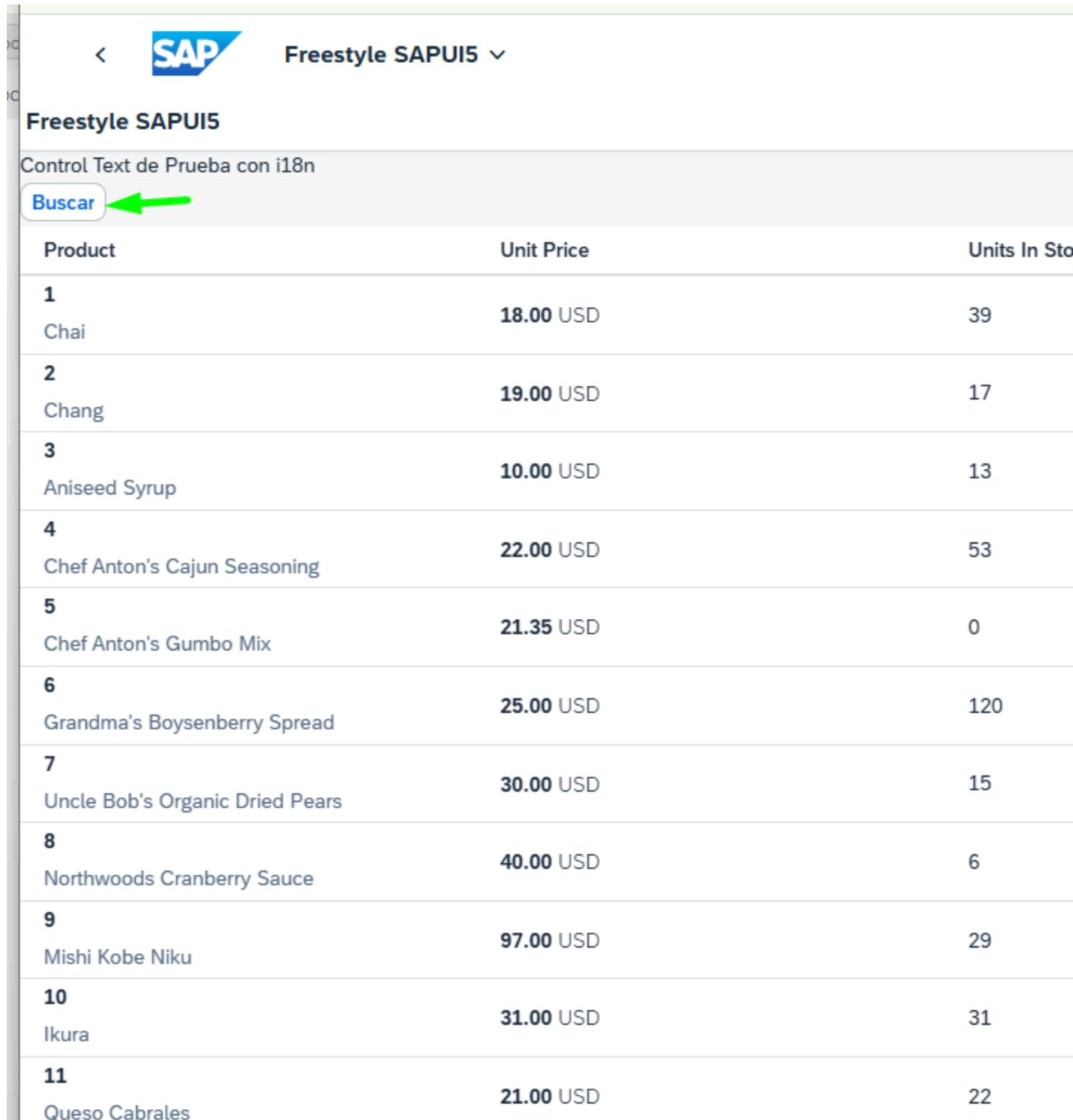
```

**parts:**[ {path: 'ProductCollection>UnitPrice' }, {path: '' }],

```

                    type: 'sap.ui.model.type.Currency',
                    formatOptions: {showMeasure: true}
                }
                unit="USD" />
                <Text id="ID_09"
                    text="{ProductCollection>UnitsInStock}" />
                <Text id="ID_08"
                    text="{ProductCollection>QuantityPerUnit}" />
            </cells>
        </ColumnListItem>
    </items>
</Table>
</VBox>
```

Ejecutamos la aplicación y damos click en el Botón Buscar:



SAP Freestyle SAPUI5

### Freestyle SAPUI5

Control Text de Prueba con i18n

Buscar

Product	Unit Price	Units In Stock
1 Chai	18.00 USD	39
2 Chang	19.00 USD	17
3 Aniseed Syrup	10.00 USD	13
4 Chef Anton's Cajun Seasoning	22.00 USD	53
5 Chef Anton's Gumbo Mix	21.35 USD	0
6 Grandma's Boysenberry Spread	25.00 USD	120
7 Uncle Bob's Organic Dried Pears	30.00 USD	15
8 Northwoods Cranberry Sauce	40.00 USD	6
9 Mishi Kobe Niku	97.00 USD	29
10 Ikura	31.00 USD	31
11 Queso Cabrales	21.00 USD	22

# Fragment

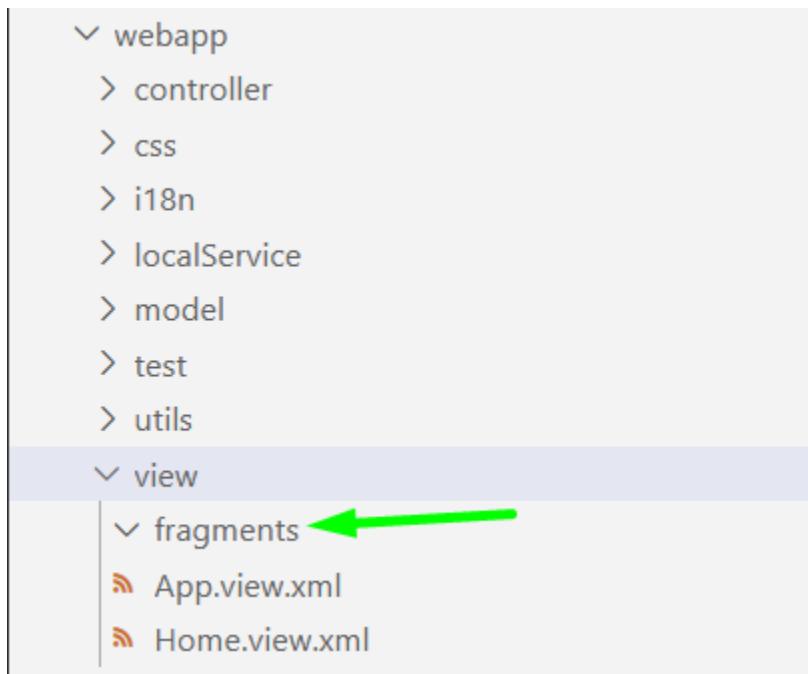
Un Fragment en SAPUI5 es una unidad reutilizable de interfaz de usuario que se utiliza para crear componentes de UI de manera modular. Los **fragments** permiten definir partes de una vista (view) que pueden ser reutilizadas en diferentes vistas o en el mismo contexto sin necesidad de duplicar código.

## Características de los Fragments:

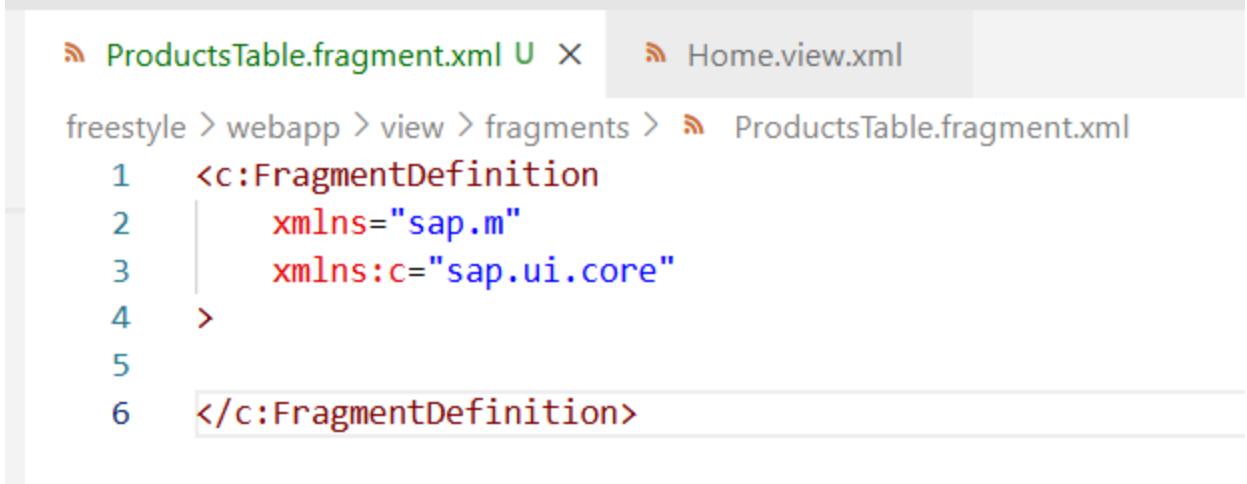
- **Reutilización:** Los fragments pueden ser utilizados en múltiples vistas, lo que ayuda a mantener el código limpio y DRY (Don't Repeat Yourself).
- **No Tienen Controlador Propio:** A diferencia de las vistas, los **fragments** no tienen su propio controlador. En su lugar, utilizan el controlador de la vista que los incluye.
- **Definición en XML o HTML:** Pueden ser definidos en archivos XML o HTML, lo que permite una fácil integración con el resto de la aplicación.

## Apliquemos a nuestro proyecto:

Creamos una carpeta llamada **fragments** en la ruta **webapp/view/fragments**



- Creamos un nuevo archivo **ProductsTable.fragment.xml** y agregamos la definición inicial del fragment así:



The screenshot shows a SAP Studio interface with two tabs: "ProductsTable.fragment.xml" (active) and "Home.view.xml". The breadcrumb navigation indicates the current location is "freestyle > webapp > view > fragments > ProductsTable.fragment.xml". The code editor displays the XML fragment definition:

```
1  <c:FragmentDefinition
2      xmlns="sap.m"
3      xmlns:c="sap.ui.core"
4  >
5
6  </c:FragmentDefinition>
```

JavaScript

```
<c:FragmentDefinition
    xmlns="sap.m"
    xmlns:c="sap.ui.core"
>

</c:FragmentDefinition>
```

- Ahora copiamos todo el contenido de la definición de la tabla de la vista Home al **fragment** que acabamos de definir:

JavaScript

```
<c:FragmentDefinition
    xmlns="sap.m"
    xmlns:c="sap.ui.core"
>

<Table id="idProductsTable"
    inset="false"
    items="{
        path: 'ProductCollection'
    }">
    <columns>
        <Column id="ID_01">
            <Text id="ID_T01" text="{i18n>Product}" />
        </Column>
```

```

<Column id="ID_02">
    <Text id="ID_T05" text="{i18n>UnitPrice}" />
</Column>
<Column id="ID_05">
    <Text id="ID_T04" text="{i18n>UnitsInStock}" />
</Column>
<Column id="ID_03">
    <Text id="ID_T02" text="{i18n>QuantityPerUnit}" />
</Column>
</columns>
<items>
    <ColumnListItem id="ID_06" vAlign="Middle">
        <cells>
            <ObjectIdentifier id="ID_07"
                title="{ProductCollection>ProductID}"
                text="{ProductCollection>ProductName}" />
            <ObjectNumber id="ID_11"
                number="{
                    parts:[{path:'ProductCollection>UnitPrice'},{path:''}],
                    type: 'sap.ui.model.type.Currency',
                    formatOptions: {showMeasure: true}
                }"
                unit="USD" />
            <Text id="ID_09"
                text="{ProductCollection>UnitsInStock}" />
            <Text id="ID_08"
                text="{ProductCollection>QuantityPerUnit}" />
        </cells>
    </ColumnListItem>
</items>
</Table>
</c:FragmentDefinition>

```

La definición del Fragment queda ya con toda la adición de la tabla [sap.ui.table](#)



```
ProductsTable.fragment.xml U X Home.view.xml M manifest.json M
freestyle > webapp > view > fragments > ProductsTable.fragment.xml

1 <c:FragmentDefinition
2   xmlns="sap.m"
3   xmlns:c="sap.ui.core"
4 >
5   <Table id="idProductsTable"
6     inset="false"
7     items="{
8       path: 'ProductCollection/'
9     }">
10    <columns>
11      <Column id="ID_01">
12        <Text id="ID_T01" text="{i18n>Product}" />
13      </Column>
14      <Column id="ID_02">
15        <Text id="ID_T05" text="{i18n>UnitPrice}" />
16      </Column>
17      <Column id="ID_05">
```

Ahora vamos modificar nuestra vista **Home** para que se cargue el fragment que definimos así:

ProductsTable.fragment.xml U      Component.js      Home.view.xml X      Story

freestyle > webapp > view > Home.view.xml  
You, 41 minutes ago | 1 author (You)

```

1  <mvc:View controllerName="com.bootcamp.sapui5.freestyle.controller.Home"
2      xmlns:mvc="sap.ui.core.mvc"
3      xmlns:c="sap.ui.core"
4      xmlns="sap.m">
5      <Page id="page" title="{i18n>title}"> You, 3 weeks ago •
6
7          <VBox id="vTest">
8              <Text id="textPrueba" text="{i18n>textPruebai18n}"></Text>
9              <Button id="btnBuscar" text="{i18n>textBuscar}" press="onPress"/>
10         </VBox>
11
12         <VBox id="vTest2">
13             <c:Fragment
14                 fragmentName="com.bootcamp.sapui5.freestyle.view.fragment"
15                 type="XML"
16             />
17         </VBox>
18     </Page>
19 </mvc:View>
```

## JavaScript

```

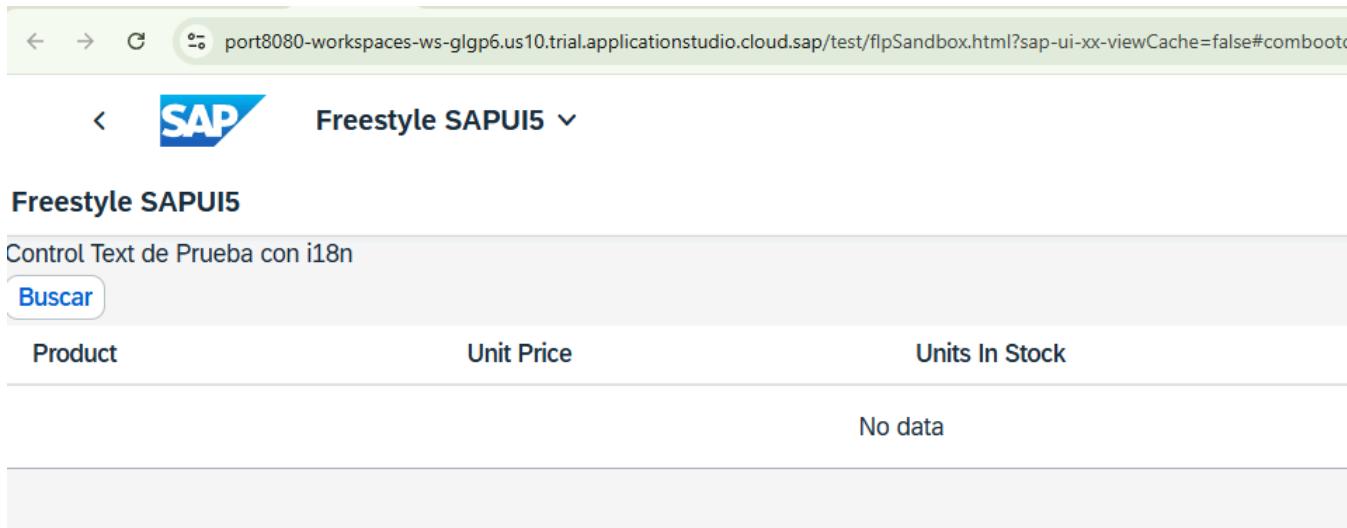
<mvc:View controllerName="com.bootcamp.sapui5.freestyle.controller.Home"
    xmlns:mvc="sap.ui.core.mvc"
    xmlns:c="sap.ui.core"
    xmlns="sap.m">
    <Page id="page" title="{i18n>title}">

        <VBox id="vTest">
            <Text id="textPrueba" text="{i18n>textPruebai18n}"></Text>
            <Button id="btnBuscar" text="{i18n>textBuscar}" press="onPress"/>
        </VBox>

        <VBox id="vTest2">
```

```
<c:Fragment  
fragmentName="com.bootcamp.sapui5.freestyle.view.fragments.ProductsTable"  
type="XML"  
/>  
</VBox>  
</Page>  
</mvc:View>
```

Al ejecutar la aplicación se presenta de la misma manera al usuario:



The screenshot shows a web browser displaying a SAP Fiori application titled "Freestyle SAPUI5". The URL in the address bar is "port8080-workspaces-ws-glgp6.us10.trial.applicationstudio.cloud.sap/test/flpSandbox.html?sap-ui-xx-viewCache=false#comboot". The page features a SAP logo and a search bar with the placeholder "Control Text de Prueba con i18n" and a "Buscar" button. Below the search bar is a table with three columns: "Product", "Unit Price", and "Units In Stock". A message "No data" is displayed in the first row of the table.

Product	Unit Price	Units In Stock
		No data