1 - 2. HÉT

ELŐKÉSZÜLETEK

Az első hét netes tananyagok tanulmányozásával telt:

- (https://help.sap.com/viewer/400066065a1b46cf91df0ab436404ddc/2.0.02/en-US/4f0a1567616541cbb7bd338fb1b64c35.html)

Próbáltam megismerkedni az SAP HANA-val mint környezettel, illetve mivel az SAP Cloud nem állt mindig rendelkezésre, ezért JavaScript tutorialokat olvastam (https://www.w3schools.com/js/).

FELADAT

A feladat egy kisebb SAP HANA XS alkalmazás létrehozása, amely egy megadott adathalmazt SAP HANA adatbázis formájában eltárol, ODATA segítségével a külvilág számára interfészt biztosít, és ennek az interfésznek a segítségével egy frontend adatokat jelenít meg, illetve tartalmaz legalább egy diagrammot is.

IMPORTÁLÁS

A projekt létrehozása után létrehoztam a projekten belül egy adatbázis modult db néven. Az adatbázis modulban szükség volt egy séma elkészítésére, ugyanis ez előfeltétele a CSV fájl beimportálásának. Ehhez készítettem a db/src mappában egy tarsasag.hdbcds nevű fájlt, amit feltöltöttem az adatbázis sémájával:

```
context Tarsasag {
    entity tarsasag_entity {
        TARS_ROV_NEV : String(255);
        ...
        ASZ_EVE      : Integer;
    };
};
```

Ezután buildeltem a db modult, és előállt az üres adatbázis, a sémában specifikált attribútumokkal. Ezek után Database Explorer-ben beimportáltam a CSV fájlt. Habár a varázsló helyesen összeegyeztette az attribútumok nagy részét, szükséges volt még kézzel összepárosítani pár oszlopot (MEGYE_KOD, JEGYZ_TOKE_ERT_HUF).

Normalizálás előtt még szükséges volt kiszedni az adattáblából a duplikátumokat, illetve minden kódhoz egyedi nevet rendelni (ugyanis a táblában sok helyen szerepelt ugyanolyan kód mellett két féle név, elgépelés miatt). Ezt SQL utasításokkal könnyedén meg lehetett oldani: először egy SELECT DISTINCT utasítással kód-név páronként kikerestem, hol szerepel ugyanolyan kód mellett két külön név, majd ezek után frissítettem a táblát.

NORMALIZÁLÁS

Most, hogy előállt a minden névhez egyedi kódot rendelő tábla, készen álltam a normalizálásra. Ehhez szétszedtem a nagy táblát sok kis, név - kód párokat tartalmazó táblákra. Először ezt SQL utasításokkal értem el, de későbbi lépések során, a kalkulációs nézet létrehozásához az SAP nem engedte ezeket a táblákat felhasználni, ezért ezek létrehozásához külön sémákat kellett létrehozni a db/tarsasag.hdbcds állományban, a következő módon:

```
context Tarsasag {
    entity tarsasag_entity {
        TARS_ROV_NEV : String(255);
        ...
        ASZ_EVE      : Integer;
    };
    entity cegall_entity {
        allapot : String(255);
        kod     : Integer;
    };
    ...
    entity telepules_entity {
        nev     : String(255);
        kod     : Integer;
    };
};
```

Létre kellett még hozni egy olyan táblát, amiben minden szerepel, csak a kódtáblában szereplő nevek nem:

```
entity tarsasag_norm_entity {
    TARS_ROV_NEV           : String(255);
    TARS_HOSZ_NEV          : String(511);
    CIM_EGYBEN             : String(255);
    TARS_TIPUS_KOD         : Integer;
    ADOSZAM                : String(255);
    GAZD_FORM_KOD          : Integer;
    CEGALL_KOD             : Integer;
    NEMGAZD_AG_KOD         : String(255);
    NEMGAZD_AGAZAT_KOD     : Integer;
    NEMGAZD_SZAKAGAZAT_KOD : Integer;
    JEGYZ_TOKE_ERT_HUF     : Double;
    ORSZAG_KOD             : String(255);
    REGIO_KOD              : Integer;
    MEGYE_KOD              : Integer;
    TELEPULES_KOD          : Integer;
    ASZ_EVE                : Integer;
};
```

KALKULÁCIÓS NÉZET LÉTREHOZÁSA

A kalkulációs nézet létrehozása előtt a `db/src/.hdiconfig` fájlban a verziószámot át kellett írni **2.0.30.0-ra**, ugyanis enélkül a kalkulációs nézet nem fog működni (mert nem enged össze JOIN-olni kettőnél több táblát). Mint minden módosítás után, most is újraépítettem a `db` modult, majd nekiláttam a kalkulációs nézet elkészítésének. A varázslóban a Data Category menüpontban először CUBE-ot választottam, de később kiderült hogy ez adatok aggregálására hasznos, és aggregáció nélkül nem is működik, ezért ezt később át kellett állítani SQL ACCESS ONLY-ra.

Ezek után készíteni kellett egy JOIN-t, és itt ki kellett választani az összes sémában specifikált kódtáblát, illetve a kódtáblákat nem tartalmazó táblát (fent: `tarsasag_norm_entity`), majd utána a grafikus szerkesztőben össze kellett kötni a megfelelő kódokat, majd a Mappping menüpont alatt az Output oszlobpa be kellett kötni a Data Sources név tagjait, illetve a kódtáblában neveket nem tartalmazó táblát is.

ODATA LÉTREHOZÁSA

Ezek után létrehoztam az Odata service-t. Ehhez készítettem egy Node.js modult `odata` néven a projekten belül, engedélyeztem az XSJS támogatást, és létrehoztam az `odata/lib/tarsasag.xsodata` fájlt:

```
service {
    "calc_view" as "tarsasag";
}
```

Szükség volt még az `odata/server.js` fájlban átírni a `redirectUrl`-hez tartozó stringet `/tarsasag.xsodata`-ra, illetve az `./mta.yaml` fájlban hozzáadni az adatbázist, mint odata függőség:

```
// ...
 - name: odata
   type: nodejs
   path: odata
   requires:
    - name: hdi_db
// ...
```

Ezek után vált futtathatóvá az Odata szolgáltatás (Run as Node.js Application). Sikeres futtatás után kaptam egy linket, és ezen keresztül már lekérdezhetővé vált a tábla.

KAPCSOLÓDÁS ODATÁHOZ ANGULAR SEGÍTSÉGÉVEL

Vasicsek Gábor segítségével előállt egy kezdetleges Angular projekt, amivel végre tudtunk hajtani egyszerű lekérdezéseket. Ahhoz hogy ez sikerülhessen, bizonyos módosításokat kellett eszközölni a projektben:

1) Át kellett írni a `./proxy.conf.js` fájlban a targetet a saját Odata linkemre (`https://ok-tnb132.inf.elte.hu:51065`), valamint a secure opciót false-ra állítani (ugyanis a szerver tanusítványát nem biztonságos harmadik fél állította ki az első két hét folyamán).
2) Az `angular.json` fájlban host-ként a `0.0.0.0` címet kellett megadni.
3) Létre kellett hozni egy odata service-t: `ng generate service odata`
4) Az `app.component.ts` fájlban végre kell hajtani a lekérdezést:

```
// ...
import { OdataService } from './services/odata.service';
// ...
constructor(private odata: OdataService) { }
ngOnInit(){
    this.odata.getData(['tars_rov_nev']).subscribe(
        res => this.data = res.d.results);
// ...
```

5) A `app.component.ts` fájlban meg kell jelenítenünk a lekérdezés eredményét:

```
<table>
<th>TARS_ROV_NEV</th>
<th>TARS_HOSZ_NEV</th>
<tr *ngFor='let item of data'>
    <td>{{item.TARS_ROV_NEV}}</td>
    <td>{{item.TARS_HOSZ_NEV}}</td>
</tr>
</table>
```

6) Meg kellett írni az odata.service-t is:

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
@Injectable()
export class OdataService {
    constructor(private http: HttpClient) { }
    configUrl = '/hana/tarsasag.xsodata' +
                '/tarsasag?$top=1000&$format=json&' +
                 '$orderby=JEGYZ_TOKE_ERT_HUF+desc';
    getData(params) {
        return this.http.get(this.configUrl);
    }
}
```

7) Ezek után elindítottam az Angular szervert (`npm start`), és `http://localhost:4200` címen keresztül sikerült is elérnem, valamint a lekérdezés eredményét is megkaptam HTML táblázatban.

FRONTEND TOVÁBBFEJLESZTÉSE

Következő célom egy olyan alkalmazás volt ahol két menüpont között lehet váltani, egy diagram és egy táblázat között. Először is jobban meg kellett ismerkednem az Angularral, ebben sokat segített az Angular Tour of Heroes tutorialja (`https://angular.io/tutorial`). Utána létre hoztam két külön Angular komponenst (`ng generate component {diagram,table}`), egy routing modult (ez vált két nézet között: `ng generate module app-routing`), majd a főkomponens HTML fájlához hozzáírni, hogy `<router-outlet></router-outlet>`.

Ezek után a table komponensben csak meg kell jeleníteni a táblázatot a fenti módon, a diagram komponensben pedig CanvasJS (`https://canvasjs.com/`) segítségével megjeleníthető a diagram:

```
// ./src/app/diagram/diagram.component.html
_____________________________________________________________________
<div id="chartContainer" style="height: 370px; width: 100%;"></div>
<div class="btn-group">
  <button type="button"
          value="2011"
          (click)="onChange($event.target.value)"
          class="btn btn-primary">2011</button>
<!-- ... ugyanez évszámokkal 2017-ig ... -->
</div>
```

A komponens TypeScript fájljában pedig megjeleníthető a diagram.