„Mini” SAP-HANA projekt dokumentációja

07.01

Végighallgattuk a bevezetést, felmértük a kezdeti érdeklődési köröket. Ismerkedtünk a SAP-HANA Cloud Foundryval.

07.02

Sikerült regisztrálni az SAP-HANA Cloud Foundry-ba, de egy egyszerű üres adatbázist nem sikerült buildelni. Próbálkoztam egy node.js modul létrehozásával, de az se sikerült. Végül egy egyszerű „Hello World” HTML5 modult sikerült csinálni.

Ennek a tutorial sorozatnak a végig olvasása, általános fogalmak megismerése (MTA, microservices, oszlop tömörítés, in memory).

https://blogs.sap.com/2017/09/04/xs-advanced-for-not-so-dummies/

https://blogs.sap.com/2017/09/05/xs-advanced-for-not-so-dummies-pt-2-multi-target-applications/

https://blogs.sap.com/2017/09/22/xs-advanced-for-not-so-dummies-pt-3-microservices/

https://blogs.sap.com/2018/02/16/xs-advanced-for-not-so-dummies-routing/

07.03

Kapott e-mail linkjeinek végigolvasása.

XSA dokumentáció

https://help.sap.com/viewer/4505d0bdaf4948449b7f7379d24d0f0d/2.0.03/en-US/d8226e641a124b629b0e8f7c111cd1ae.html

CDS dokumentáció

https://help.sap.com/viewer/4505d0bdaf4948449b7f7379d24d0f0d/2.0.03/en-US/d8226e641a124b629b0e8f7c111cd1ae.html

PAL dokumentáció

https://help.sap.com/viewer/2cfbc5cf2bc14f028cfbe2a2bba60a50/2.0.03/en-US/c9eeed704f3f4ec39441434db8a874ad.html

Mivel a szerver hibás működése miatt a kapott tutorialok kipróbálása sikertelen volt, más ismeretek megszerzése felé indultam el. Láttam, hogy a javascript egy nagyon fontos eleme ennek a projectnek, így az alábbi tutorialok segítségével megpróbáltam tapasztalatot szerezni js kód írásában.

https://www.youtube.com/watch?v=PkZNo7MFNFg

Adatvizualizáció d3.js könyvtár használatával. Ezután js-be kézzel bevitt kódokat már tudtam vizualizálni.

https://www.youtube.com/watch?v=nzshmMlOuwI

07.04

Megkaptuk csv fájlt, végighallgattuk a teendőket vele. Sajnos szerver oldalú hiba miatt a több óra munkával semmit sem értünk el. Miután ezt a tevékenységet feladtam, elkezdtem Thomas Jung SAP-HANA developer youtube tutorial videóit nézni. Ezekben levezeti, hogy miért van szükség erre a technológiára, milyen hardware fejlődések tették elérhetővé, és a fontosabb fogalmakat ábrákkal is vizualizálta. Sajnos, amikor gyakorlati részre ért, a nem működő szerverek miatt nem lehetett kipróbálni.

07.05

Megkaptuk az ELTE szervereket, és létrehoztuk az első adatbázist. Elkezdtük az adatok hibájával az ismerkedést.

Adatbázis létrehozása

1. Az egyetemi lokális szerverre bejelentkezés: https://oktnb132.inf.elte.hu:53075/watt/index.html; első bejelentkezés után módosítjuk a jelszavunkat.
2. Új Multi Target Applicationt hozunk létre.
3. Jobb klikk a munka mappánkra majd „New Project From Template”.
4. Ezen belül jobb klikkel „New SAP HANA database module”.
5. Így keletkezik egy src mappa, amin jobb klikkel: „New HDB CDS Artifact”.
6. Ezt megnyitva egy kontextust látunk, és azon belül hozzunk létre egy entityt, mindkettőnek tetszőleges nevet adhatunk.
7. Entityt megnyitva létrehozzuk az összes fejlécet, és mindegyiknek beállítjuk a típusát. A kapott csv fájl fejléceit hoztuk létre.
8. Majd buildeljük a projektet.
9. Database explorerben létrehozunk egy új adatbázist „Add a database to the Database Explorer” (plusz ikonnal jelölt) lehetőségre, és a felugró listából kiválasztjuk a MTA projectünk nevét.
10. Ezután a table fülön belül megtaláljuk a létrehozott adatbázis modellünket, amire jobb gombbal kattintva felugrik az „Import Data” lehetőség.
11. A felugró menüben kiválasztjuk a csv fájlunkat.
12. Ellenőrizzük, hogy az attribútumokat helyes felismerte-e, ha nem vagy hiányosak, akkor kézzel beállítjuk.
13. Végül a majdnem sikeres import után a két teljesen hibás sort az SQL console-ba írt kóddal töröltem.

07.08

Modulok létrehozását tanultam.

https://developers.sap.com/tutorials/xsa-connecting-webide.html

https://developers.sap.com/tutorials/xsa-html5-module.html

https://developers.sap.com/tutorials/xsa-hdi-module.html

https://developers.sap.com/tutorials/xsa-xsjs-xsodata.html

HDI container létrehozása, adatokkal feltöltése. Végül a front-end fejlesztését Angularban csináltam, így később nem kellett használni html5 modult. Megpróbáltam odata-t létrehozni, de jogosultságok miatt az nem sikerült. Továbbiakban az adathalamazt tisztítottam SQL kód segítségével, és létrehoztam további táblákat, hogy a felesleges ismétlődéseket elkerüljem.

Adat tisztítása és normalizálása

1. Az összes segéd-táblánknak először létrehozunk egy entityt. Buildelés után ez az entity megjelenik a Database Explorer Tables menüpontja alatt.
2. SQL kód segítségével feltöltjük adatokkal ezeket a táblákat.
3. A táblázatok létrehozásában a felesleges információk ismétlődésének minimalizálására törekedtem.
4. Következő segédtáblákat hoztam létre:
    a. Eredeti CEGEKTABLA tábla minden attribútummal
    b. TARSTIPUS(TARS_TIPUS_MEGNEV, TARS_TIPUS_KOD)
    c. GAZD_FORM(GAZD_FORM, GAZD_FORM_KOD)
    d. CEGALL(CEGALL, CEGALL_KOD)
    e. NEMGAZD_AG(NEMGAZD_AG_MEGNEV, NEMGAZD_AG_KOD)
    f. NEMGAZD_AGAZATNEMGAZD_AGAZAT_MEGNEV, NEMGAZD_AGAZAT_KOD)
    g. NEMGAZD_SZAKAGAZAT(NEMGAZD_SZAKAGAZAT_MEGNEV, NEMGAZD_SZAKAGAZAT_KOD)
    h. ORSZAG(ORSZAG, ORSZAG_KOD)
    i. REGIO(REGIO, REGIO_KOD)

j. MEGYE(MEGYE, MEGYE_KOD)
k. TELEPULES(TELEPULES, TELEPULES_KOD)
l. NCEGEKTABLA(TARS_ROV_NEV, TARS_HOSZ_NEV, CIM_EGYBEN, TARS_TIPUS_KOD, ADOSZAM, GAZD_FORM_KOD, CEGALL_KOD, NEMGAZD_AG_KOD, NEMGAZD_AGAZAT_KOD, NEMGAZD_SZAKAGAZAT_KOD, JEGYZ_TOKE_ERT_HUF, ORSZAG_KOD, REGIO_KOD, MEGYE_KOD, TELEPULES_KOD, ASZ_EVE)

07.09

Sikerült Odata-t létrehozni. Utána az Odatának a lekérdezésével js kódban próbálkoztam, de ez nem sikerült.

Odata létrehozása

1. Az eddig elkészített MTA-ban létrehozunk egy „New Node.js Module”-t.
2. „Enable XSJS support” lehetőséget ki kell pipálni.
3. Ebben az állományban lib mappájában készítünk egy új fájlt, aminek a kiterjesztése xsodata.
4. xsodata állományba a következőt írtam:

```
service {

    "CEGEKTABLA" as "tablazatodata"  keys generate local "ID";

    "kalkulacios" as "kalkulaciosodata"  keys generate local "ID";

    "CMEGYE" as "cmegyeodata";

    }
```

5. Ezzel a beállítással elérhető url-ben az eredeti tábla, a kalkulációs nézet, és egy egyedi lekérdezés táblám, „CMEGYE”, amiben megyénként a cégek száma van.
6. Server.js fájl-nak a redirectUrl utáni részt át kell írni „/<név>.xsodata”-ra.
7. A buildelés után kapunk egy URL linket, amin keresztül elérhetjük a service-ben szereplő táblákat.

07.10

Calculation View létrehozása

1. Az első létrehozott mappának a .hdiconfig kiterjesztésű fájlában a plugin_version-t átírjuk 2.0.30.0-ra.
2. A src-mappára jobb klikkel kattintva „New Calculation View”.
3. Create Jointtal létrehozunk egy joint, amit át is nevezhetünk.
4. Táblázatok megadása forrásként „Add Data Source” funkcióval.
5. Kétszer kattintunk a joinra, Join Definition fülben meghatározzuk az inner joinokat az adattagok összekötésével.
6. Mapping fülben mindegyik adattagok behúzzuk, de csak egyszer.
7. Bezárjuk a szerkesztőt, dupla kattintás az Aggregationra.
8. Mapping fülben is minden adatot behúzunk.
9. Build után már működni fog a Data Preview.

07.10-11

Végül az Angular sikeres telepítése után és egy hallgató kollégától kapott kezdetleges kód segítségével elkezdtem a front-end fejlesztését és a dokumentáció megírását.

Front-End, Angular

1. Angular telepítése console-ból npm install -g @angular/cli paranccsal
2. Ha ez a parancs nem működik, telepíteni kell a node.js-t
3. Az adatok vizualizációjához töltsük le a canvasjs könyvtárat.
4. app.component.ts állományba importáljuk a canvasjs állományt
5. services mappá belül, odata.service.ts-t hozunk létre és ezzel a kóddal beállítjuk az odata url-t.

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';

@Injectable()
export class OdataService {
  constructor(private http: HttpClient) {   }

configUrl = '/hana/myodata.xsodata/';

getData(params)
{
    return this.http.get(this.configUrl + params);
}
}
```

1. Odatáról az adatok megszerzése:

```
export class AppComponent implements OnInit {
```

ezen belül:

```
this.odata.getData('tablazatodata?$top=1000&$orderby=JEGYZ_TOKE_ERT_HUF+desc&distinct=true
').subscribe((res :any) => {
      this.data = res.d.results;
      show_graph(this.data,this.data2);
    });
```

majd ezután:

```
function show_graph(db,db2) {

      var asd1 = [];
      var asd2 = [];

      for (var i = 0; i < db.length; ++i) {
        asd1.push({ label: db[i].TARS_ROV_NEV, y: db[i].JEGYZ_TOKE_ERT_HUF / 1000000 })
      }
```

show_graph függvény végén megírjuk a kirajzolást a canvasjs állományban található példák alapján.