Első két hét eredménye

Feladat

- Bármilyen front-end technológiával adatok vizualizálása *SAP Hana* adatbázisból nyert adatokkal *ODATA* segítségével.

Előkészületek

- Sikeresen létrehoztam egy *SAP Hana Trial* fiókot, amivel használni tudtam a SAP Web IDE-t, először *SAP* szerveren, majd technikai problémák miatt később saját szerveren.
- Még mielőtt nekikezdtem volna a teszt projekt létrehozásának, előtte több napig oktatóanyagokat olvastam magáról a környezetről, az architektúráról (https://help.sap.com/viewer/4505d0bdaf4948449b7f7379d24d0f0d/2.0.03/en-US/d8226e641a124b629b0e8f7c111cd1ae.html). Tudatában, hogy a projektben adatbázisra van szükségünk, eddigi tudásaimat kiegészítettem ebben a témakörben (http://scs.web.elte.hu/Work/DW/adattarhazak.htm). Ezáltal elsajátítottam rengeteg fogalmat, mint például az adatkocka, pivot, delta adattöltés jelentéseit.
- Technikai problémák miatt nem tudtuk használni a *SAP Web IDE*-t, illetve még ekkor nem tudtuk, hogy milyen témakörben fogunk dolgozni, ezért további ismeretek elsajátításába kezdtem:
    - Először is egy *Python* gyorstalpalót kezdtem, pár nap alatt sikerült segédanyag mellett megtanulni, illetve begyakorolni (https://www.w3schools.com/python/python_intro.asp).
    - Ezután kicsit front-end technológiák iránt kezdtem el érdeklődni, például a *React* (https://www.w3schools.com/react/), és az *Angular* (https://www.w3schools.com/angular/) után. Ezeket a tudásokat nem sajátítottam el, inkább csak olvasgattam róluk.
    - Legutolsósorban a *REST Api* és az *ODATA* elmélete (https://www.odata.org/getting-started/understand-odata-in-6-steps/), iránt érdeklődtem

Mini-Projekt megkezdése

- Második hét elejére végre elkészült a saját szerverünk, így meg is kezdődött a saját mini-projektünk elkészítése.
- Először is létrehoztam egy új *MTA applikációt* a *SAP Web IDE Workspace*-ben, ami egy webes felületet jelenít meg, míg perzisztencia rétegét adatbázis támogatja.

Adatbázis létrehozása

- Projekten belül létrehoztam egy *SAP HANA Database Module*-t, ami alapján csatlakozni tudunk adatbázishoz, illetve ezen belül adatbázishoz szükséges kellékeket (kontextus, tábla, nézet) tudunk létrehozni. Ezt build-elve automatikusan létrehozott, illetve csatlakozott a saját adatbázisunkhoz (ha esetleg nem automatikusan, akkor *Database Explorer*-ben manuálisan is megtehetjük).
- Az előbb készített mappában ezután létrehoztam egy *HDB CDS Artifact*-ot, ami egy adatbázis kontextus, itt tudjuk definiálni a szükséges fogalmakat (típusokat) a táblákhoz, illetve létre tudunk hozni táblákat, azoknak attribútumjait, illetve megszorításokat tenni rá. Jelen esetben létrehoztam az általunk megkapott tábla sémáját, aztán magába a táblába beimportáltam az adatokat.
- Ezután megkezdődött a normalizálás és a tisztítás: a normalizált táblák sémáját szintén létrehoztam, a fő táblából feltöltöttem ezeket ügyelve arra, hogy kiszűrjem a duplikátumokat, illetve a hibás adatokat javítsam.

Kalkulációs nézet létrehozása

- Először is utánanéztem a kalkulációs nézetnek, pontosan mit is jelent ez, illetve milyen fajtái vannak, illetve mik is az előnyeik (https://www.guru99.com/sap-hana-calculation-view.html).
- Majd megkezdtem a létrehozását, először is a *HDB CFS Artifact*-ban létrehoztam egy *SQL Access Only* kalkulációs nézetet, join típusként projekciót használtam, ugyanis szimplán az adatokat akartam elérni. Ha bármilyen aggregációt használtam volna, akkor természetesen *CUBE* típusút, illetve a aggregáció join-t használtam volna, mivel úgy hatékonyabb.
- Ezután hozzáadtam még egy join-t, amiben megadtam hogy a normalizált táblákat hogyan kösse össze egymással, illetve a join output-jához

hozzáadtam minden táblát, ugyanis minden adatra kíváncsiak vagyunk. Majd végül ezt rákötöttem a projekciós join-ra, és ott szintén minden adatot hozzáadtam output-nak.

- Ezután egy build volt szükséges a sikeres kalkulációs nézet elkészüléséhez.

Összekapcsolás *ODATA*-val

- A cél az volt, hogy egy *Node.js* modult futtatva url-en keresztül külsőleg le lehessen kérdezni adatokat *ODATA* segítségével.
- Emiatt létrehoztam egy új *Node.js* modult az *MTA* applikációmban és a következőket módosítottam:
    - lib mappában létrehoztam egy xsodata kiterjesztésű fájlt, amiben megadtam, hogy melyik táblámat miként tudjam elérni az url-ben *(service{ <elérni_kívánt_táblanév> as <url-ben_hivatkozandó_fájlnév>; }).*
    - *server.js* fájlban a "*redirectUrl*"-t az előbb készített xsodata fájl nevét megadtam.
    - *mta.yaml* fájlban az odata modulban megadtuk hogy szüksége van a *HDI container*-re (*requieres : - <hdi_container_module_név>*)
- Ezek után futtatva sikeresen elérjük a táblát.

Adatok megjelenítése *Angular*-ral

- Első hét alatt szereztem minimális *Angular* ismereteket, ezt bővítettem tovább mielőtt nekikezdtem volna a megjelenítésnek tutorial-ok alapján (https://www.npmjs.com/package/angular-odata-es5).
- Először is a projektnek hozzá kell tudnia csatlakozni az előbb elkészített back-end szerverünkhöz, ezt *proxy* segítségével sikerült megoldani egy tutorial alapján (https://medium.com/bb-tutorials-and-thoughts/angular-how-to-proxy-to-backend-server-6fb37ef0d025).
- Sikeres csatlakozás után további *Angular* ismeretek szereztem, egy tutorial (https://www.codementor.io/antonselin/consume-odata-service-restful-angular-2-uj1i6szsw) segítségével el tudtam készíteni a saját projektet, aminek az eredménye az adatok diagramos ábrázolása, szűrhető feltétellel (aminek az értékei szintén az adatbázisból vannak).