Két heti beszámoló + dokumentáció

i.sz. 2k19. Július 12.

Kovács Máté
ELTE Szakmai gyakorlat

Két hét részletesebb felbontása:

Első hét (2019.07.01-05)

Az első héten a SAP Hana szerver megismerésére szántuk csoportosan.Sajnos az SAP gyorsan fejlődő rendszere miatt a kapott tutorialok már elavultak voltak, de gyors korrigáció után már a megfelelő tutorialokat tudtuk böngészni mindannyian.
A tutorialok magába foglalták az SAP fájlrendszerének használatát, azonkívül betekintést nyertünk a különböző projectek összekötésének how-to-jába.

Az SAP keretrendszer megismerés a hét végére átcsapott fakultatívvá, tekintve mindenki elkezdett "szakosodni" az általa választott feladatkörben.Én a Java programozási nyelv megismerését tűztem ki célul, több primitív konzol- és GUI alkalmazást is készítettem, melyekkel sikerült elsajátítani a nyelv alapvető szintaktikáját.

A hét végén megkaptuk az első feladatunkat, mely lentebb van részletesen leírva.
A feladat megoldása eleinte minden személynél eltért, mivel az adatbázis tisztításához mindenki máshogy állt hozzá.
Külön Problémát okozott az SAP szerverek túlterheltsége/leállása, emiatt a pénteki nap nagyrészt eredménytelenül telt el.

Második hét(2019.07.08-12)

Hétfő

A második héten már tudtuk használni az ELTE által futatott házi SAP hana webszervert, Hétfő délután már teljes jogosultságunk volt hozzá, ezzel újból nekiállhattunk tisztítani a kapott adatbázist(lentebb részletezve).

Kedd (Szerda, csütörtök, péntek nyaraláson vettem részt, hétvégén hoztam be a lemaradást)

Reggel Béleczki András segítségével a teljesen elérhető odata-ban dolgoztunk.
Déltől kezdve felbátorodva azon az ötleten, miszerint a kapott adatokból a cégeket el lehetne helyezni egy beágyazott map api-n, nekiálltam a feladat megoldásának Java környezetben.A feladatot tovább nehezítette a tény, hogy :
-Google maps API nem ingyenes, ezáltal későbbi javaslat után Bing maps API-t kellett használnunk
- a Bing maps API jóval fejletlenebb és kevésbé felhasználóbarát Java környezetben
-Java környezetben ilyenfajta problémát előttem még nagyon kevés ember oldott meg, melyet általában nem dokumentált az interneten, ezáltal "ismeretlen vizeken hajóztam".

Hosszú keresés után végre találtam egy oldalt, ahol a probléma megoldása hasonló volt ahhoz amire nekem szükségem van.(Sajnos az oldalt a mai napig nem sikerült újra megtalálni.)

A neten talált kód alapján sikerült létrehozni egy olyan programot, mely a bing api-t használva, megfelelő mennyiségű helyadat(házszám, utca, város, ország) birtokában egy bing szerveren tárolt xml/json file-t ad vissza.Az itt felmerülő probléma az volt, hogy a visszakapott xml minden karaktere a két koordinátán kívül felesleges, 'trash' volt, melytől a megszabadulás újabb problémákat vetett fel.Eleinte próbálkoztunk az xml/json file parse-olásával, ezáltal könnyen kinyerhetővé és hivatkozhatóvá váltak volna a koordináták. Következő probléma egy, előre megírt parser beimportálása volt a projektbe, mely jelentős ideig (egy-másfél nap) eltartott, mivel az online talált parserek mind 'homebrew', third party által készített parserek voltak, amik használata és megértése kisebb-nagyobb nehézségeket vetett fel(végre megértettem ezáltal a helyes és kiterjedt dokumentáció lényegét). Ellenben hamar rájöttem, hogy a bing szerver oldali xml file szintaxisa eltér a parser által várt szintaxistól, így kénytelen voltam saját parsert írni, mely működése meglepően egyszerű: Egy egyszerű keresés tételt (http://bzsr.web.elte.hu/progmod2/konyv.pdf, 143.o. 12.2.5 ös bekezdés) módosítottam úgy, hogy a .json file-on vesszőnként splitelve, stringeket ellenőrizve haladjon végig, és így találja meg a megfelelő adattagot.Mivel a keresés nem megy végig a fileon, emiatt a saját "parser"-em gyorsabb, mint bárki más által írt parser, feladatspecifikussága miatt.Sajnos a program működése még így is elfogadhatatlanul lassú, főleg a webes lekérdezés lassúsága miatt, mely jelentősen függ a hálózat sebességétől, a szerveroldal terheltségétől, stb., és ezáltal más programba beágyazva, 1000+ lekérdezésnél már észrevehető várakozási idő.

Ezt a problémát figyelembe véve döntöttem a Java nyelv időleges félretételén, és kezdtem el az Angular környezet és JS nyelv felületes tanulmányozását, mellyel töltöttem el a maradék időmet a hétvégéből.

Feladat megoldásának áttekintése

2019, es ELTE-Soft nyári szakmai gyakorlatának első két hete.

Az első két hét feladata ismeretterjesztő jellegű volt, hiszen mindannyian teljesen új területre sodródtunk, egy adatbázis rendszert kellett megvalósítanunk az SAP-HANA adatbáziskezelőben, majd az adatbázis odata formátumban elérve, egy front-end oldali diagram megvalósítása volt a cél.

Célok

1. egy adatbázis rendszert létrehozni SAP-HANA adatbáziskezelőben

2. odata formátumban elérve, egy front-end oldali diagram megvalósítása

Tervezet

1. Adatbázis
    a. Adott csv fájl feltöltése a SAP Web IDE-be , majd adattisztítás és normalizálás, amennyire csak lehetséges
    b. Ezután kalkulációs nézet létrehozása
    c. Majd Odata-ban exportálni.
2. Front-end
    a. Odata elérése
    b. Adatok vizualizációja

Megvalósítás menete

Adatbázis

Egy eléggé rossz csv fájlt kaptunk kiindulópontnak, hiszen tele volt sorduplikátumokkal, összeférhetetlen adatokkal, valamint helytelenül kitöltött sor is található volt benne. Ezek kiszűrése kisebb fejfájást okozott de előbb utóbb mindenkinek sikerült megoldást találni a problémára. Én személy szerint próbálkoztam azzal, hogy töröltem azokat a sorokat amelyekben volt üres adatmező, vagy hibás típusú adat volt valahol, emellett töröltem a duplikátumokat egy sima distinct lekérdezéssel.

A csv fájl feltöltése igen egyszerű volt, manuálisan annyit kellet megtenni, hogy a projekthez, hozzáadunk egy database module-t majd a .hdbcds kiterjesztésű fájlba kézzel beleírni az adattábla nevét és sorainak attribútumait: pl

```
entity regio_kod {
    regio_kod : Integer;
    regio     : String(600);
};
```

Ezután lehetett is a táblába importálni a csv-t.

Miután ez megvolt, kezdődhetett a tábla darabokra vágása, külön táblába szedtem mindent aminek volt, kód attribútuma (így tűnt logikusnak).

És jöhet a calc view: miután megvolt a tábla értelmesen elvégzett szétdarabolása eme tutoriál segítségével a Kalkulációs nézet megteremtése pofonegyszerű volt (csak kicsit macerás a sok húzogatás)
https://developers.sap.com/tutorials/xsa-graphical-cube-view.html

És az odata : a projekthez hozzá kellet adni egy új modult, : ‘new nodejs module’

És engedélyezni kellett az xsjs supportot benne létrehozáskor.Majd az újonnan készült moduleban létrehozni egy .dxodata kiterjesztésű fájlt és abba a következő kódrészlet megoldotta az exportot:

```
service{
"Calc_view" as "DATA" keys generate local "ID"  ;
"db.data" as "ALL" keys generate local "ID" ;
"TARSASAGOKORIGIN" as "TARS" keys generate local "ID";
}
```

Front end

Angular oldal készítését vállaltam el, hiszen ajánlották, hogy jó lenne ha valaki foglalkozna angularral.

Legfontosabb kérdés, az volt, hogy hogyan érem el az eddig készített Odata exportom. Erre egyszerű megoldást találtunk : http.get(odata url) segítségével szuper egyszerűen megoldható. (ELTEkintettünk attól a problémától, hogy csak 1+e3 sort tudunk egyszerre lekérdezni, most ez is jó volt nekünk.)

Ehhez ez a tutorial oldal tökéletes magyarázatot nyújtott:
https://angular.io/guide/http

Ez tökéletes is volt csak, létezik egy **same-origin policy** ami eléggé nagy fejfájást okozott, de proxyval ki lehetett kerülni.

Ezután már meg is voltak az adatok amiket js- oldalon könnyű formázni és kimutatni, kerestünk is egy lib et ami pont ezzel foglalkozik, így találtam rá a canvasjs-re mellyel gyönyörűbbnél gyönyörűbb ábrákat lehetett készíteni, így kész is lett a kitűzött feladat: