# CoffeeShop - Programrendszerek fejlesztése gyakorlat - Projekt

***Szvoreny Viktor, DXVVS0***

# Munkanapló

**2022.05.02-05. - NodeJS + MongoDB + Heroku**

A projekt bázisát képező MongoDB adatbázist úgy valósítottam meg, hogy a MongoDB cloudban létrehoztam egy fiókot, majd egy clustert, amiben megtalálható az adatbázisom is, amely két táblából áll: `aruk`, mely a kávékat tartalmazza a tulajdonságaival együtt, valamint a `users`, amely a felhasználók adatait tartalmazza (a jelszót hashelve). Elkészítettem az `arukereso.schema.js` fájlban az adatstruktúrát majd a `routes.js`-ben a POST, GET, PUT, DELETE műveleteket valósítottam meg. Ezután Postmanben teszteltem a megfelelő működést. Az elején sok hibaüzenetet kaptam a POST-nál, majd több kísérlet és olvasás után kiderült, hogy a Body típusánál a raw-text volt kiválasztva, nem pedig a JSON formátum. Ezt megváltoztatva már megfelelő volt a működés Postmanen keresztül. Ezután a felhasználónál hasonlóan jártam el, azonban itt szükség volt lokális stratégiára és autentikációra is, ami a Passport segítségével történik meg. A felhasználók adatbázisában a jelszó a bcryptet használva hashelve tárolódik el. Ez a `user.schema.js`-ben valósul meg a jelszó összehasonlítással együtt.

Miután meggyőződtem róla, hogy a funkciók megfelelően működik, regisztráltam Herokura és elkezdtem a hosztolás megvalósítását. Itt `fatal: Not a git repository (or any of the parent directories): .git` problémába ütköztem, de rájöttem, hogy csak a git műveletek sorrendjén kellett módosítani. Ezután `error: src refspec master does not match any.` jellegű hibába ütköztem, amikor a `git push heroku main` parancsot szerettem volna futtatni. Ebben az esetben a git verziómmal lehetett probléma, mert a `git push heroku master` parancssal már működött a művelet.

**2022.05.06-09. - Angular alkalmazás**

Létrehoztam a first, guard, login komponenseket, valamint a login és aruk serviceket. A loginnál és az aruknál egyaránt összekötöttem a megfelelő műveleteket a korábban megvalósítottakkal. A loginhoz még szükséges volt egy felület is az adatok beviteléhez. Ezután az volt a cél, hogy detektáljam, hogy sikeres volt-e a bejelentkezés, ha igen, akkor a következő oldal a `first` volt, amely az az oldal, ahol a felhasználó láthatja az árukat. Ehhez a korábban megvalósított lekérésekre volt szükségem, mely az `aruk.service.ts`-ben van implementálva. Ennél a részfeladatnál a lekérés eredményének típusával akadtak problémáim, így nem tudtam megfelelően letárolni. Viszont, ha nem tároltam le, csak meghívtam a függvényt, akkor az végtelen ciklust eredményezett. Ezért elkészítettem egy interface-t `Coffee` néven az elvárt struktúrával. Ebből aztán elkészítettem egy típust, így a listázó metódus alapértelmezetten ebbe kasztolta a válaszként kapott JSON-t. Így már sikeres volt a listázás. A kávékat szerettem volna esztétikusabb módon megjeleníteni, melyhez `MatCard`-ot használtam. Elhelyeztem egy `BUY` gombot is, amelyet megnyomva meghívódik a PUT metódus, és a vásárlással eggyel kevesebb lesz az adott kávéból. Az Angularos részt a Firebase segítségével hosztoltam, amely a korábban megvalósított Heroku szervert használja. Ezen kívül a listázáshoz szükséges volt megoldani a CORS problémát, melyet az órán tanult `whiteList`-es megoldással kiviteleztem. A `whiteList` két elemet tartalmaz: a

`localhost:4200`-t a lokális futtatáshoz, valamint a Firebase által hosztolt URL-t. Így megfelelően működik a kommunikáció a backend és frontend között, a műveletek sikeresen megvalósulnak.

# Elérhetőség

Szvoreny Viktor - szvoreny.viktor@inf.u-szeged.hu

Project Link (Firebase): https://coffee-shop-szv.web.app/ (be lehet jelentkezni szaboz userrel)

Heroku Link: https://protected-gorge-95133.herokuapp.com/ (de így magában internal server errort dob a CORS whitelist miatt, viszont a https://coffee-shop-szv.web.app/ -ról elérhető)