Eötvös Loránd Tudományegyetem

Informatikai Kar

Média- és Oktatásinformatikai Tanszék

Céges termék nyilvántartó alkalmazás

**Témavezető:**

Menyhárt László Gábor

Adjunktus

**Szerző:**

Katona János Dávid

Programtervező Informatikus BSc.

Budapest, 2020

Tartalom

[1. Bevezetés 5](#_Toc27324492)

[2. Felhasználó dokumentáció 7](#_Toc27324493)

[2.1. Célközönség 7](#_Toc27324494)

[2.2. Minimális rendszerkövetelmények 7](#_Toc27324495)

[2.3. Bejelentkezés előtti állapot 7](#_Toc27324496)

[2.4. Hitelesített felhasználó 9](#_Toc27324497)

[2.4.1. Bolt hozzáadása 10](#_Toc27324498)

[2.4.2. Bolt szerkesztése 10](#_Toc27324499)

[2.4.3. Új termék hozzáadása 11](#_Toc27324500)

[2.4.4. Cég termékei 12](#_Toc27324501)

[2.4.5. Bolt termékei 14](#_Toc27324502)

[2.4.6. Eladások 15](#_Toc27324503)

[2.4.7. Cég felhasználói 16](#_Toc27324504)

[2.4.8. Regisztrációskód generálás 17](#_Toc27324505)

[2.4.9. Kijelentkezés 17](#_Toc27324506)

[3. Üzemeltetési dokumentáció 18](#_Toc27324507)

[4. Fejlesztői dokumentáció 19](#_Toc27324508)

[4.1. Tervezés 19](#_Toc27324509)

[4.1.1. Funkcionális követelmények 19](#_Toc27324510)

[4.1.2. Használati esetek 19](#_Toc27324511)

[4.1.3. Nem funkcionális követelmények és architektúra terv 19](#_Toc27324512)

[4.1.4. Adatbázis terv 20](#_Toc27324513)

[4.2. Implementáció 22](#_Toc27324514)

[4.2.1. Fejlesztői környezet 22](#_Toc27324515)

[4.2.2. Az alkalmazás konfigurálása 22](#_Toc27324516)

[4.2.3. Alkalmazás telepítése és futtatása 23](#_Toc27324517)

[4.2.4. Backend Architektúrája 24](#_Toc27324518)

[4.2.5. API-k 25](#_Toc27324519)

[4.2.6. A Frontend Architektúrája 27](#_Toc27324520)

[4.2.6.1. Frontend modulok 28](#_Toc27324521)

[4.3. Fejlesztést segítő eszközök 28](#_Toc27324522)

[4.4. Tesztek 29](#_Toc27324523)

[4.4.1. Egységtesztek 29](#_Toc27324524)

[4.4.2. Manuális tesztek 29](#_Toc27324525)

[4.5. További fejlesztési lehetőségek 35](#_Toc27324526)

# Bevezetés

A modern felgyorsuló világban, ahol szinte minden információt meg lehet találni néhány percen belül, elengedhetetlen, hogy ezeket az adatokat a lehető legrendszerezettebb módon tároljuk. Fontos az is, hogy ezeket helytakarékosan, ugyanazt az információmorzsát minél kevesebbszer raktározzuk el úgy, hogy az ne menjen a működés rovására. A vonalkódok nyilvántartásában ma is káosz van, erről sok dolog tehet… A szabályozatlanság, a hanyagság, az, hogy a technológia nem volt alkalmas arra, hogy ilyen adatokat tároljanak nagy mennyiségben abban az időben, amikor bejött a köztudatba. Hozzátett ehhez a cégek érdektelensége, hogy ezeket az adatokat eltárolják egy közös nagy rendszerben.

A szakdolgozatom célja az, hogy egy ilyen rendszer alapjait letegye, mely egyben vonzóvá teszi az értékesítők számára a használatát. Ennek legkönnyebben terjeszthető módja egy többrétegű webes alkalmazás, amely a következő elemekből áll:

* Egy frontendből, ami REST API hívásokon keresztül kommunikál a backenddel JSON objektumok által, ezt Angularral valósítottam meg.
* Egy backendből, ami JPA repository segítségével keresi, és menti el az adatokat.
* Illetve egy H2 adatbázisból, ami minden indulásnál újra feltölti magát az előre felvitt adatokkal, így a fejlesztés folyamán keletkező rekordokat nem kell minden alkalommal kitakarítani.

Az alkalmazásban három felhasználó típust különböztetek meg. Az egyik az ADMIN, aki felveszi a céget az alkalmazásba, később egy esetleges névváltoztatás után őt kell megkérni, hogy tegye meg a szükséges módosításokat. A cég felvétel után egy regisztrációs kódot küld neki, amivel el tudja kezdeni a cég a működést. A másik a COMPANY\_ADMIN, aki a fel tud venni boltokat, tárgyakat a céghez, a felvett boltokhoz hozzárendelheti a céghez rendelt tárgyakat kiegészítő adatokkal, meg tudja nézni az eladásokat az elmúlt időszakra, és ő is tud generálni regisztrációs kódokat, de ezeket csak a saját cégéhez. Természetesen az eladásokon és a cég nevén kívül minden céggel kapcsolatos információt módosítani is tud. A harmadik felhasználó a SELLER, aki egy csökkentett jogú COMPANY\_ADMIN, mert bár a céghez tud adni termékeket, és módosítani is tudja, de nem tud regisztrációs kódokat generálni, nem láthatja az eladásokat, és csak a saját boltjainak a termékeit tudja módosítani, más boltokét nem, ezt is csak akkor, ha egy COMPANY\_ADMIN már beállította, hogy melyik bolthoz tartozik.

A megoldás még kezdetleges, de ennek az is oka, hogy minden cégnek más igényei lehetnek egy ilyen alkalmazással kapcsolatban, így egy általam elképzelt a funkcionalitásokat kiemelő elvárásrendszer mentén van a feladat implementálva. Ilyen egyszerűsítés például, hogy nem a pénztárgépre van jelenleg kötve az eladás, hanem csak egyszerűen a boltok termékei lista melletti gombra megtörténik a tranzakció.

# Felhasználó dokumentáció

Ebben a fejezetben ismertetem az elérni kívánt felhasználóbázist, a minimális rendszerkövetelményeket és a program működését felhasználói szempontból.

## Célközönség

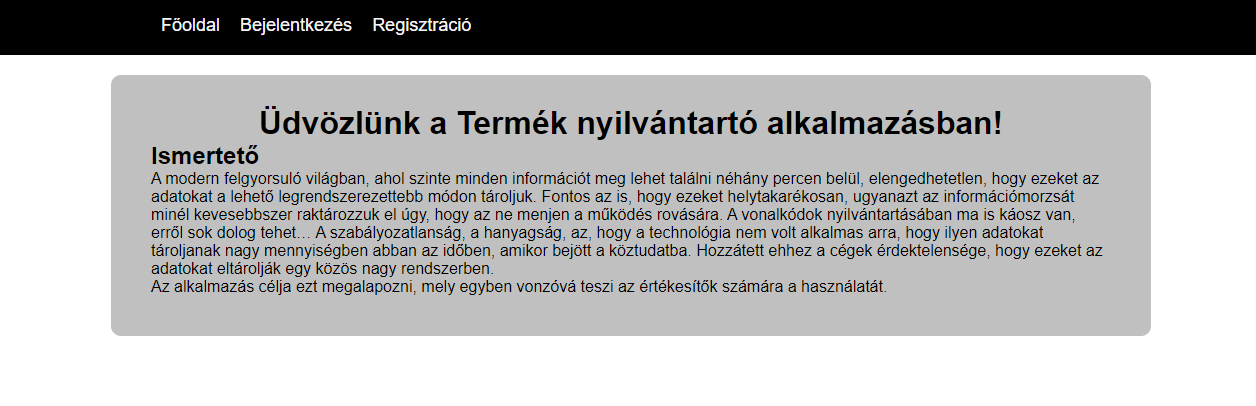
Az alkalmazás azért jött létre, hogy egy nagy termékadatbázis legyen belőle, így minden értékesítéssel foglalkozó cégnek hasznos lehet, hogy beregisztrálják magukat ide, azért, hogy az eladási statisztikájukat, leltározási feladatukat egy helyen tudják végezni. Később az alkalmazást tovább fejlesztve a felhasználóbázis bővülhet a vevők bevonásával. Ebben a központi adatbázisban egy termék vonalkódjára rákeresve megnézhetik, hogy kik árulják a keresett terméket, milyen áron, és hol.

## Minimális rendszerkövetelmények

Az alkalmazás bármely népszerű böngészőből elérhető, kisebb megjelenési különbségekkel. Speciális konfigurációt, telepítést ez a program nem igényel felhasználói oldalról. Az alkalmazás stabil hálózati kapcsolattal, minimum 1440x900-as felbontással ajánlott.

## Bejelentkezés előtti állapot

Nem regisztrált és hitelesített felhasználónak az alkalmazás a főoldalt (1. ábra), a bejelentkezést, és a regisztráció oldalt jeleníti meg.



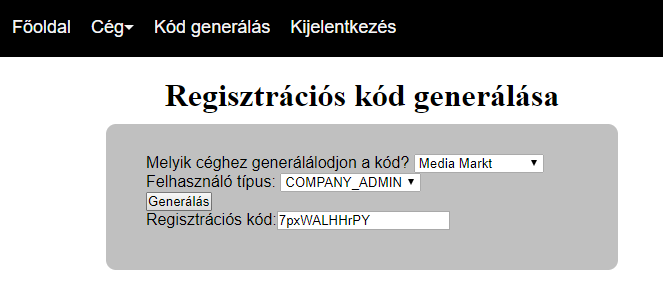
1.ábra Az alkalmazás főoldala

Amennyiben az oldal látogatójának még nincs felhasználója, úgy egy már regisztrált tagtól kell kérnie egy regisztrációs kódot. Attól függően, hogy egy új céget szeretne beregisztrálni, vagy már létezik a cég, ahova egy alfelhasználót szeretne igényelni más emberrel kell kapcsolatba lépnie egy ilyen kódhoz való hozzájutáshoz.

Ha új céget szeretne a látogató bejegyezni, egy ADMIN jogosultságú felhasználót kell megkérnie, a megfelelő papírokkal alátámasztva, hogy ő a cégnek egy törvényes képviselője, aki ezt az alkalmazást használni szeretné. Ilyenkor az ADMIN létrehoz egy céget (2. ábra), majd generál egy regisztrációs kódot (3. ábra), amivel egy COMPANY\_ADMIN jogosultságú felhasználót lehet beregisztrálni. Ezt a megbeszélt formában eljuttatja az új ügyfelének.



2.ábra Cég hozzáadása



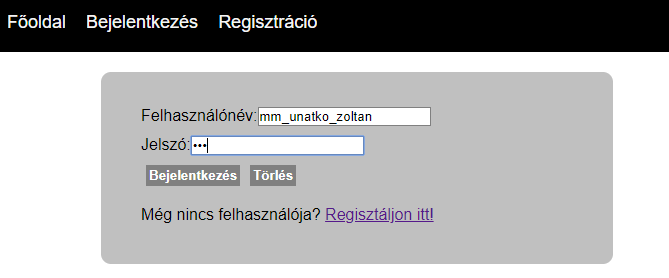
3.ábra Regisztrációs kód generálása

Az regisztrációs űrlapon (4. ábra) egy egyedi felhasználónév, e-mail cím, jelszó, annak megerősítése és a regisztrációs kód megadása utána a Regisztráció gomb megnyomásával ellenőrzi a szerver az adatokat, és ha minden megfelelő, akkor elmenti az adatbázisba az új felhasználót, és a főoldalra navigál át. Nem megfelelő esetben figyelmeztetésként kap visszajelzést a felhasználó, a problémával kapcsolatban.



4.ábra Regisztráció

Sikeres regisztráció után nincs mire várni, rögtön be is lehet lépni a bejelentkezés oldalon (5. ábra) a regisztrációnál megadott adatokkal.



5.ábra Bejelentkezés

## Hitelesített felhasználó

Hitelesített felhasználó 3 féle lehet:

* Az ADMIN, akinek a két főbb funkcióját megmutattuk az előző pontban. Ilyen jogosultsághoz csak másik ADMIN tud létrehozni regisztrációs kódot.
* A COMPANY\_ADMIN, aki most lett létrehozva, és lentebb lesz bemutatva.
* Illetve a SELLER, aki egy COMPANY\_ADMIN kevesebb jogosultsággal, a kettő közti különbség lentebb kerül szemléltetésre.

### Bolt hozzáadása

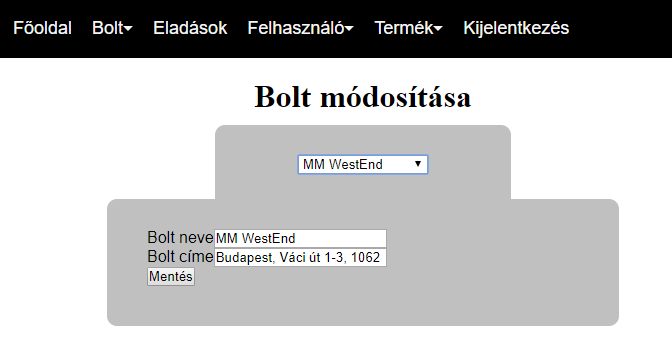
A COMPANY\_ADMIN jogosultságú felhasználókkal lehetőség van bolt felvételére (6. ábra), itt egy olyan nevet érdemes megadni, amiről később fel lehet ismerni az üzletet. Ez azért fontos, mert nincs megkötés arra, hogy minden bolt neve egyedi legyen, tehát minden helyet ugyanúgy nevez a felhasználó, akkor hamar felhasználási problémába fog ütközni. Címet is érdemes itt megadni, így majd a vásárlók meg tudják találni az üzletet. A mentés gomb megnyomásával hozzáadódik a bejelentkezett felhasználó cégéhez a megadott bolt. Ilyet nem tud a SELLER jogosultsággal rendelkező felhasználó.



6.ábra Bolt hozzáadása

### Bolt szerkesztése

A bolt esetleges költözése, vagy csak névváltoztatása esetére van egy bolt szerkesztése oldal (7. ábra). Itt a legördülő listában kiválasztott üzletet tudja szerkeszteni. Miután kiválasztotta az általa szerkeszteni kívánt boltot, a két szöveges beviteli mező kitöltődik az aktuálisan eltárolt értékekkel. Ez abban az esetben hasznos, ha csak valami elírást akar javítani rajta a felhasználó, így az könnyebben megtehető. A Mentés gombra kattintva eltárolódik az adatbázisban a módosított bolt. Ilyet nem tud a SELLER jogosultsággal rendelkező felhasználó.



7.ábra Bolt szerkesztése

### Új termék hozzáadása

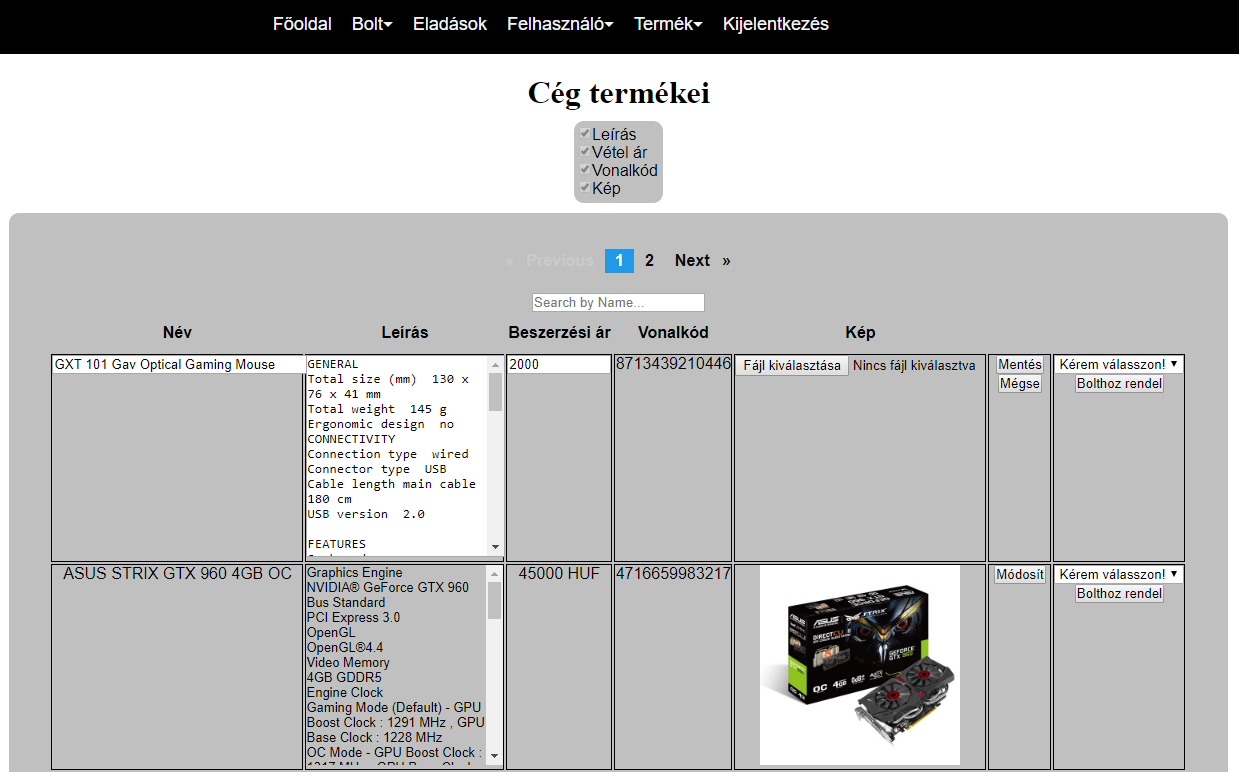
Az eladni kívánt termékeket először a céghez kell hozzáadni (8. ábra), itt olyan adatokat kell megadni, ami a boltoktól függetlenül érvényes a termékre. Ilyen adatok a termék neve, leírása, beszerzési ára, vonalkódja, és egy kép a termékről (jelenleg csak png-t fogad el az alkalmazás). Ez utóbbi opcionális, de a többi fontos, hogy meg legyen jól adva, minél részletesebben, van a leírás megfogalmazva annál több információt fog a vásárló látni a termékről, ez segíti őt a vásárlás melletti döntésben. A vonalkódot fontos, hogy pontosan adja meg a felhasználó, ugyanis ennek módosítására később nincs alkalom, ha elrontja, akkor az üzemeltetést kell megkérnie, hogy távolítsa el ezt az adatbázisból. A mentés gombra nyomva bekerülnek az adatok az adatbázisba, ha még nem volt ilyen vonalkódú terméke a cégnek. Ebben az esetben figyelmeztetésként látható, hogy már létezik ilyen terméke a cégnek. Ilyet tud a SELLER jogosultságú felhasználó.



8.ábra Új termék hozzáadása

### Cég termékei

Miután hozzáadta a céghez a termékeket, ki tudja listázni őket a felhasználó (9. ábra). A fenti jelölőnégyzetek segítségével kiválasztható, hogy mely adatokat mutassa a termékről. Lehet köztük keresni, de jelenleg csak név alapján. Itt a módosításra gombra kattintva tudja szerkeszteni az adott sor elemeit. A felhasználó végzett a szerkesztéssel, akkor a mentésre kattintás esetén a módosított adatok el lesznek tárolva az adatbázisban. Ha nincs megadva új kép, akkor a régi marad, így egy kép törlésére nincs lehetőség ilyen módon. Ezt a SELLER jogosultságú felhasználó is meg tudja tenni.



9.ábra Cég termékei

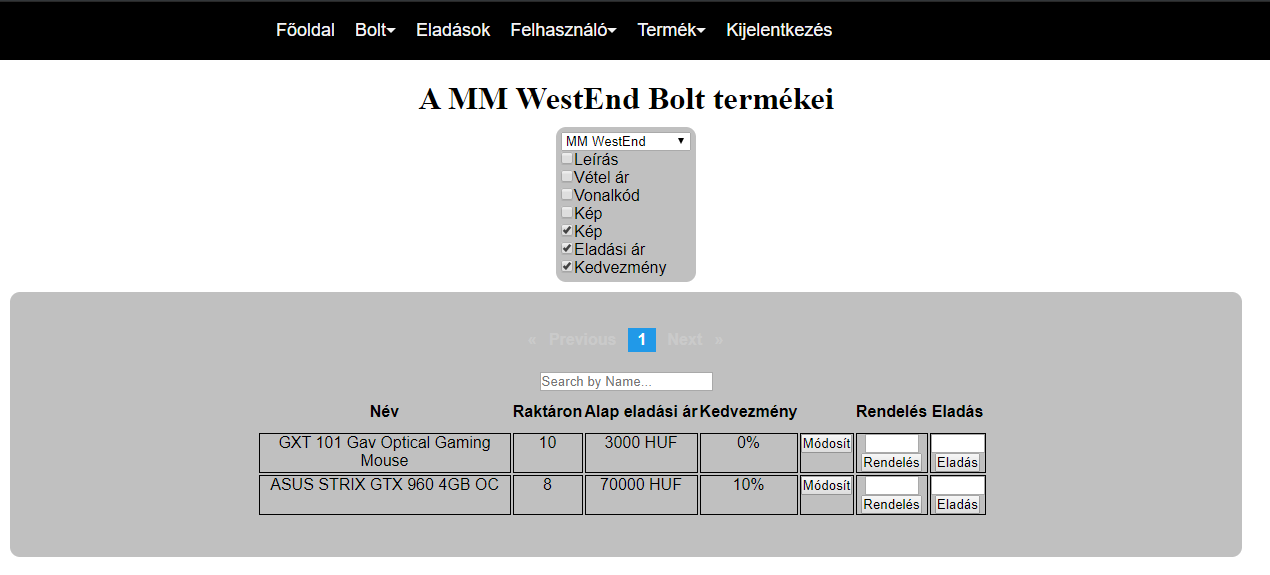
Az utolsó oszlopban az aktuális sorban megjelenő terméket lehet a cég boltjaihoz rendelni. Miután kiválasztotta a felhasználó a legördülő választási listából a boltot, amihez hozzá szeretné rendelni, az alatta lévő bolthoz rendel gombra kattintva, átnavigálja egy űrlaphoz (10. ábra), amin a termék bolthoz kapcsolódó adatait lehet megadni. Ha a felhasználó nem találja a választási opciók között a boltot, amihez hozzá szeretné rendelni a terméket, akkor ez azért van, mert már korábban ez megtörtént, így a duplikátumok szűrve vannak. Az űrlapon a következő adatokat kell megadni: alap eladási ár, raktáron lévő mennyiség és kedvezmény (ez utóbbit százalékban). A Mentés gomb megnyomása után az adatok eltárolódnak az adatbázisban, és a bolt termékei között a hozzáadott termék is megtalálható lesz. Ezt a SELLER jogosultságú felhasználó is meg tudja tenni, annyi szigorítással, hogy ő csak a saját boltjához tudja rendelni a terméket, így neki nem lesz választási listája, csak egy Boltomhoz rendelem gomb.



10.ábra Termék hozzáadása bolthoz

### Bolt termékei

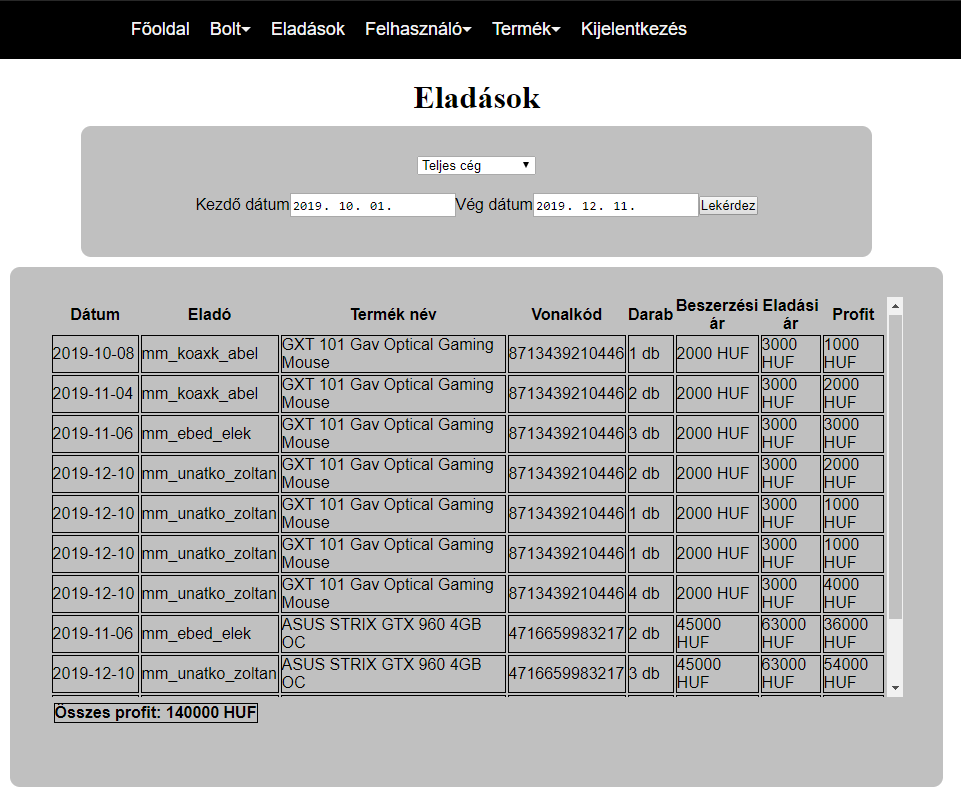
Miután van bolthoz rendelt termék, ezeket is ki lehet listázni (11. ábra). Felül egy legördülő választási listából lehet kiválasztani, hogy melyik bolt termékeit mutassa a program. Lentebb azonos keresőmező, táblázat, és hozzátartozó jelölőnégyzetek láthatóak, mint a Cég termékei oldalon, annyi különbséggel, hogy itt meg vannak jelenítve azok az adatok is, amiket a bolthoz rendelés során lettek megadva. Ez utóbbiakat lehet módosítani is, de amit a Cég termékei lapon voltak módosíthatóak, azokat itt már csak megtekinteni lehet. Az utolsó két oszlopban található a Rendelés, és az Eladás. A rendelés, és a raktáron lévő mennyiség módosítása között jelenleg nincs különbség, később viszont hasznos lehet, egyéb funkciók hozzátársítása mellett. Az eladásnál viszont elmentődik a felhasználó, aki eladta, hogy mennyi volt az aktuális ár a kedvezmény levonása után, és hogy mennyi darab lett eladva. Ezt a SELLER jogosultságú felhasználó annyi különbséggel használhatja, hogy neki csak a saját boltjának a termékei jelennek meg, nincsenek választási opciói.



11.ábra bolt termékei

### Eladások

A cégnek fontos látnia, milyen bevételeik vannak az eladásból, így erre is van egy oldal (12. ábra). Kiválasztható felül a választási listából az, hogy melyik boltra akarjuk az eladásokat vizsgálni, de akár a teljes céget is vizsgálhatjuk a felhasználó által lentebb megadott időszakban. A Lekérdez gomb megnyomása után megjelenik a megadott időszakban lezajlott összes eladása a cégnek egy táblázatban. Ebben a táblázatban feltüntetett adatok a következőek: dátum, eladó felhasználóneve, termék neve, vonalkódja, eladási darabszám, beszerzési ár, eladási ár és a profit. A táblázat alján pedig a teljes profit összege látható az adott időszakra. Erre egy SELLER jogosultságú felhasználónak nincs lehetősége.



12.ábra Eladások

### Cég felhasználói

Ezen az oldalon látható, hogy milyen felhasználók tartoznak a céghez. A SELLER jogosultságúaknak itt lehet beállítani, hogy melyik bolthoz tartozzanak egy legördülő választási lista segítségével. A mentés gomb lenyomásával bekerül az adatbázisba így, ha a felhasználónak eddig még nem volt boltja, akkor ezután már képes a saját boltjához rendelni cégtermékeket, listázni, rendelni és eladni a boltja termékeit.



13.ábra Cég felhasználói

### Regisztrációskód generálás

Az ADMIN felhasználóhoz hasonlóan a COMPANY\_ADMIN-nak is lehetősége van regisztrációs kódot generálnia (3. ábra). A különbség annyi, hogy nem kell megadnia a céget, amihez generálja a kódot, csak a felhasználói jogosultságot, amivel a felhasználót be akarja regisztráltatni. Ez két féle lehet: COMPANY\_ADMIN és SELLER. A generált kódot ugyanúgy, megbeszélés alapján eljuttatja a kijelölt személynek. Erre a SELLER jogosultságú felhasználónak nincs joga.

### Kijelentkezés

A kijelentkezés gombra kattintva újra a főoldalon leszünk, már nem beléptetett felhasználóként. Ezt a műveletet a SELLER jogosultságú felhasználó is el tudja végezni.

# Üzemeltetési dokumentáció

A fejlesztőnek kötelessége ezt a fejezetet frissítenie minden kiadott verzió után.

Ami elengedhetetlen az üzemeltetéshez, az a program futtatásához szükséges szoftverek. Ezek jelenleg a JAVA 1.8, Maven 3.5.3, NodeJs 10.16.0. JAVA-hoz a JAVA\_HOME környezeti változó szükséges, a Mavenhöz pedig a MAVEN\_HOME.

A frontendhez szükséges NodeJs telepítő fájlja általában elvégzi helyettünk a környezeti változó beaállítását, de ha mégsem, akkor itt csak a PATH-hez kell hozzáadni a C:/Program Files/nodejs/ útvonalat. Az npm-et szintén beállítja a NodeJs telepítője, ha mégsem, akkor ennek is kell a PATH-be a mappa elérési útvonala. Nekem ez az útvonal a C:/Users/%username%/AppData/Roaming/npm volt. Ezután adjuk ki az *npm install -g angular-cli* parancsot. Jó esetben beállítja az PATH-t az angular-hoz, de ha nem, akkor a  
C:/Users/%username%/AppData/Roaming/npm/node\_modules/@angular/cli/bin útvonalat kell hozzáírni.

A backend telepítése az *mvn package* paranccsal indítható el az alkalmazás főkönyvtárából. Ez létrehoz egy JAR fájlt a target mappában. Ezt *java -jar %filename%* paranccsal indítható el a target mappából. Indítás után a backend figyelni fog az application.properties fájlban megadott porton keresztül.

A frontend telepítése az src/client/thesis mappában az *ng build* paranccsal indítható el. Ez létrehoz egy dist mappát, ebben lesz minden fájl, ami szükséges az alkalmazás webszerveren való üzemeltetéséhez.

A port-forwarding és tűzfal beállításokat a rendszergazda szakértelmére bízzuk.

# Fejlesztői dokumentáció

Ebben a fejezetben ismertetem a fejlesztésbe bekapcsolódni kívánó embereknek, milyen információkra lehet szükségük ahhoz, hogy minél hamarabb érdemben hozzá tudjanak tenni a projekthez.

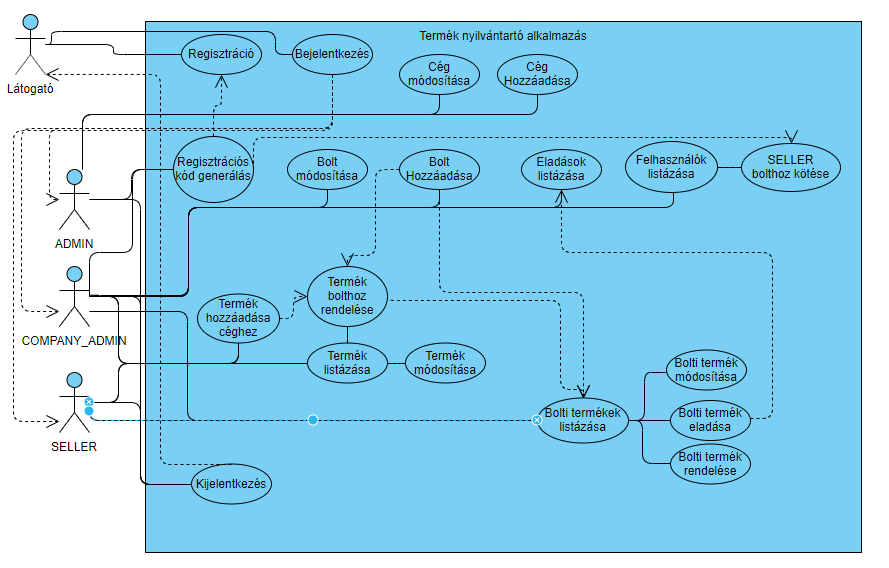
## Tervezés

### Funkcionális követelmények

A dolgozatban egy webes alkalmazást, egy katalogizáló szoftvert valósítok meg. Az alkalmazás képes a felhasználók kezelésére, a hitelesített felhasználó jogosultságától függően tud műveleteket végrehajtani.

### Használati esetek

A használati eseteket az alábbi képen illusztrálom. (14. ábra)



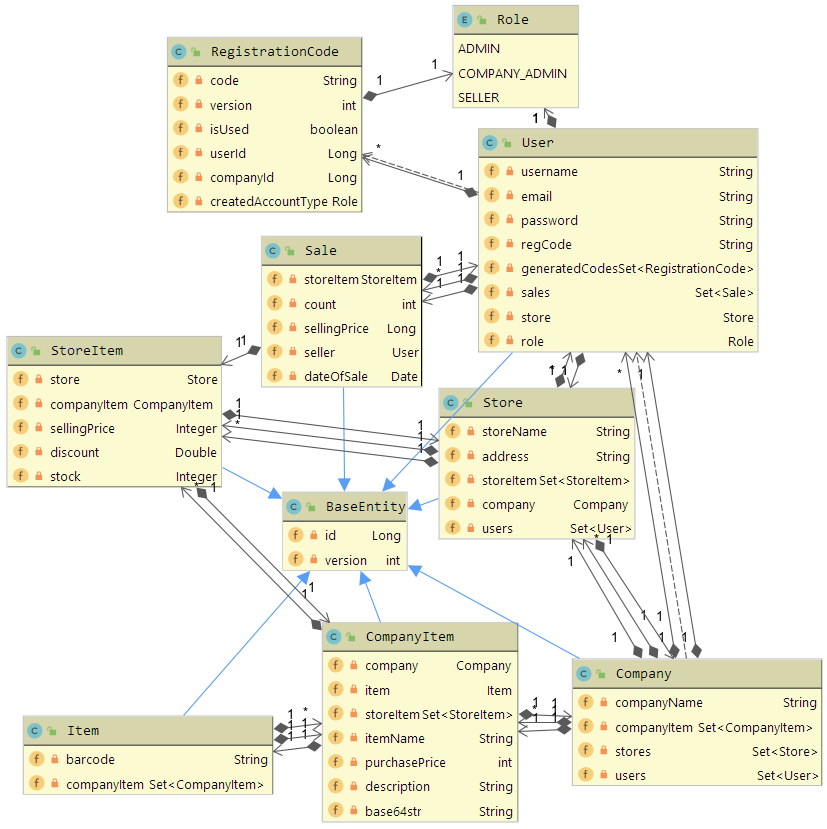
14.ábra Használati esetek

### Nem funkcionális követelmények és architektúra terv

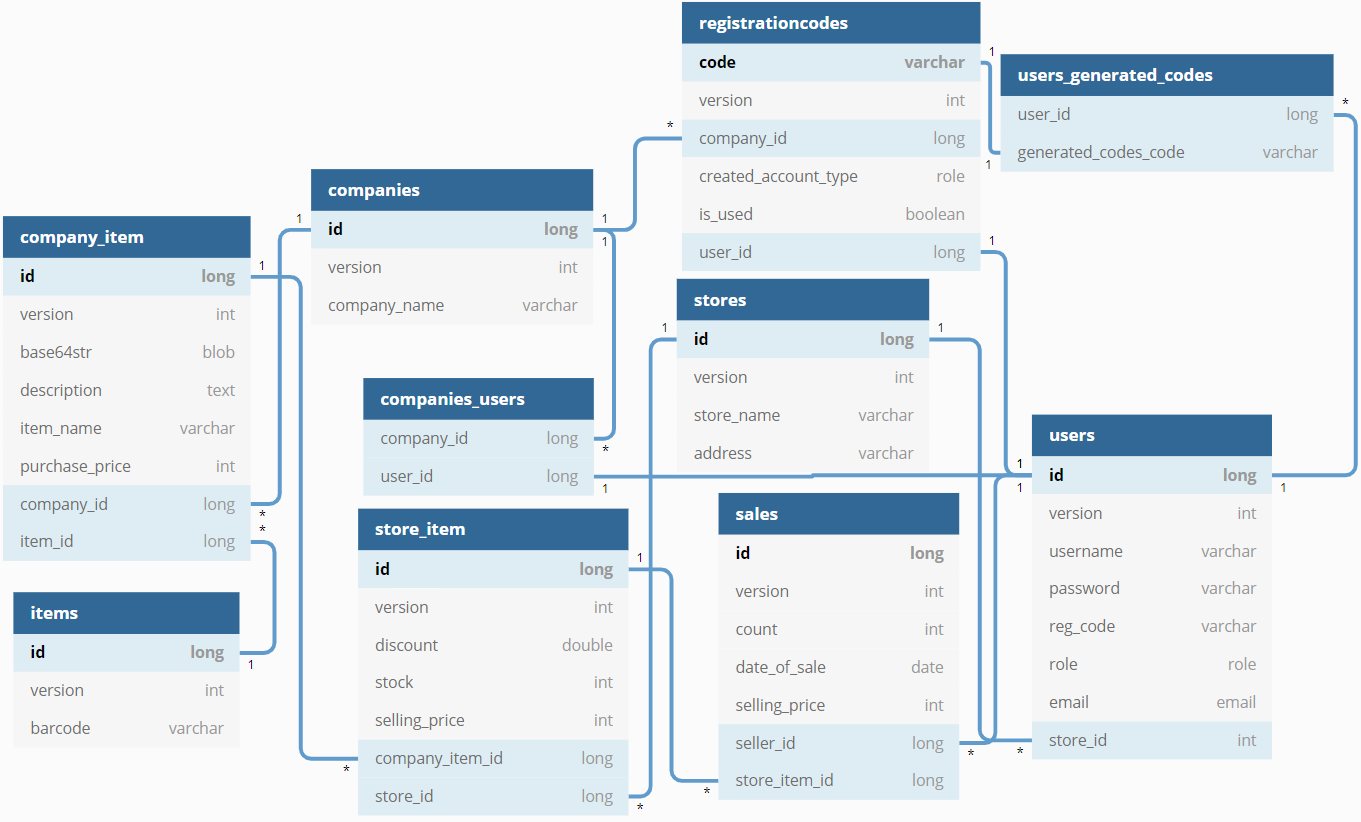
Az alkalmazás legyen platformfüggetlen, továbbfejleszthető, és részekre bontott. Ezért két külön futó egységre, egy megjelenítésért felelős, és egy alkalmazás szerverre bontottam. Így a megjelenítés teljesen különálló, könnyen cserélhető. Akár mobil alkalmazást is lehet hozzá készíteni, a könnyebb vonalkód, fénykép bevitelhez. A kommunikáció a két egység között REST szabvány szerint működik, ez könnyíti az alkalmazás kiegészítését újabb használati esetekkel.

### Adatbázis terv

Egy katalogizáló szoftvernek a legfontosabb része az adatbázis, melyben az adatokat tárolja. Ahhoz, hogy ezeket tárolhassuk, tudnunk kell milyen mezői lesznek a tábláknak (16. ábra), illetve, hogy milyen osztályokkal férhetünk hozzá az adatstruktúrához (15. ábra).



15.ábra Entitások osztálydiagramja



16.ábra Az adatbázis táblák és kapcsolataik

## Implementáció

### Fejlesztői környezet

A szerver futtatásához szükséges szoftverek:

* JAVA 1.8
* Apache Maven 3.5.3

A kliens futtatásához szükséges szoftverek:

* NodeJs 10.16.0
* npm 6.13.0

Továbbiakban még szükség van egy böngészőre is, amin keresztül látható a módosítások, amit a fejlesztő eszközölt.

A backenden a gettereket, settereket, és egyéb gyakran használt függvényeket a Lombok nyílt forráskódú könyvtár rakja bele a fordítás előtt a kódba. Ezeket a fordítóprogram dönti el, hogy kellenek-e vagy sem az osztályok, vagy osztályszintű adattagok előtt szereplő annotációk alapján. Az általam használt IDE-hez fel kellett telepíteni a Lombok plugint, hogy helyesen lássa, ezeknek a függvényeknek a létezését.

### Az alkalmazás konfigurálása

Az alkalmazás két komponensből áll. A Spring keretrendszerben írt backendből és az Angularban írt megjelenítő szerverből. A backend indításakor le fog futni egy csomó sql utasítás, ami feltölti a H2 adatbázis a benne megadott adatokkal. Ezeket az sql parancsokat a src/main/resources mappában található data.sql fájlban érhető el, és módosítható, ha a fejlesztőnek más adatok kellenek. Fontos megjegyezni, hogy amíg ezt a H2 adatbázist használja az alkalmazás, addig minden egyes alkalommal, amikor újraindítjuk a backendet, akkor az adatok, amik az adatbázisban módosultak az előző indítás óta, azok elvesznek.

A backend konfigurálását az src/main/resources/application.propertities fájlban található. Itt beállítható a szerver portja (**server.port**) illetve az adatbázis konfigurációja, annak elérhetősége és karakterkódolása is.

A frontendnél az src/client/thesis/src mappában az environment.ts-ben kell megadni a szerver url-jét, és portját, amire ez csatlakozni fog. Az alkalmazás ezen része alapból a 4200-as porton fut, melyet csak akkor módosítsunk az indító scripthez hozzáírható –port %port\_száma% flaggel, ha a backendben módosítjuk a CrossOrigin-t.

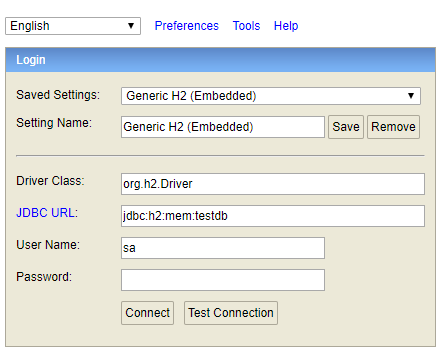
### Alkalmazás telepítése és futtatása

Miután konfiguráltuk az alkalmazást érdemes ellenőrizni, hogy a környezeti változók is meg vannak adva. A JAVA-hoz a JAVA\_HOME környezeti változó szükséges, a Mavenhöz pedig a MAVEN\_HOME. A NodeJs telepítő fájlja általában elvégzi helyettünk ezt a munkát, de ha mégsem, akkor itt csak a PATH-hez kell hozzáadni a C:/Program Files/nodejs/ útvonalat. Az npm-et szintén beállítja a NodeJs telepítője, ha mégsem, akkor ennek is kell a PATH-be a mappa elérési útvonala. Nekem ez az útvonal a C:/users/%username%/AppData/Roaming/npm volt.

A környezetei változók beállítása után az alkalmazás fő mappájában adjuk ki az *mvn clean build* parancsot a szerverhez kapcsolódó függelékek letöltéséhez, és telepítéséhez. Ezeket a Maven a pom.xml fájlban találja. Telepítés után lefuttatja a szerverhez kapcsolódó egységteszteket is, így láthatjuk, ha valami nem stimmel az alkalmazással.

A kliens telepítéséhez menjünk bele a src/client/thesis mappába, itt adjuk ki az *npm install* parancsot. Ez hasonló módon a mavenhöz letölti és telepíti a klienshez szükséges függelékeket. A Node Package Manager ezeket a package.json-ből olvassa ki, és a node\_modules mappába telepíti a szükséges komponenseket.

A szerver az *mvn spring-boot:run* paranccsal indul el, amit az alkalmazás fő könyvtárában kell kiadni. Az adatbázist, ha nem módosítottuk a konfigurációt, akkor a http://localhost:3000/h2 linken érhető el a böngészőből. A 17. ábrán látható, hogy néz ki a bejelentkező felület, és mivel léphetünk be.



17.ábra H2 adatbázis konzol

A frontend az *ng serve* utasítással kezdi meg a működését, amit a a src/client/thesis mappából kell kiadni. A megjelenítési felület a http://localhost:4200-on tesztelhető.

### Backend Architektúrája

A Spring keretrendszer, amit használ az alkalmazás egy népszerű nyílt forráskódú keretrendszer a JAVA-hoz. Moduláris felépítésű, ami különböző architektúrájú alkalmazások fejlesztéséhez nyújt támogatást. Számos modulja van ezek közül, amit ez a program is használ: Spring boot, Spring Web, Spring Security és a Spring Data JPA. A Spring Boot egy alkalmazás szerver tárolót biztosít, így nem kell a programot külön telepíteni, hanem egy Tomcat szerverre fogja felépíteni induláskor magát az alkalmazás.

A backend MVC vagyis Mode-View-Controller minta megvalósítást próbál követni. A model réteg a backend szerver központi eleme, ide tartoznak az adattároló -kereső eljárások, és az entitások, amikbe kerül az adatbázisból kinyert record(ok). A tárolás és kereséshez ez az alkalmazás JPA repository-kat használ, ami kicsit megkönnyíti a fejlesztő életét, ha könnyű adatbázis műveletekre használja, de nagyon nehéz optimalizálni a lekérdezéseket. A lekérdezés végeredménye Data Access Objectekben tárolódnak a program futása során, ezek a model mappában találhatóak. A lekérdezésekhez használt függvényeket a repository mappában találjuk DAO-k szerint elkülönítve. Ezeket a repositorykat a szervizekből hívódnak meg, ami már a controller réteghez kapcsolódik. A szervízeket pedig API hívások segítségével éri el a View réteg, amit a controller osztályok fognak el, és adják tovább a megfelelő szerviz függvénynek. Vannak esetek, amikor nem elég a DAO, mert több, vagy épp kevesebb adattaggal kommunikál a frontend a backend felé, így az ilyen esetekre létre kell hozni DTO-kat, vagyis Data Trasfer Objecteket. Ezeket a communication mappában találjuk.

### API-k

Négy Controller osztályba össze lettek sűrítve azok a kommunikációs útvonalak, amik kívülről elérhetőek http kommunikáción keresztül. A viszonylag kevés lehetőséget sem használja ki az alkalmazás, ugyanis, csupán GET és POST metódusokon keresztül el lehet érni az összes API-t. Mielőtt elérné a controllereket a hívás, a spring security megnézi, hogy a kérést küldő személynek van-e joga annak az API-nak a használatára. Ezt a config mappán belüli SecurityConfig osztály szabályozza.

Minden szervízben van egy ErrorResponse ami, ha nem null, akkor a Controller osztály ami meghívta a szervízt tudja, hogy hiba keletkezett a futás során. Ilyenkor ezt fogja elküldeni a válaszban, majd visszaállítja nullra, hogy legközelebb is működjön a hibakezelés.

A Company Controllerben ezek a következők:

* /api/company (GET) Visszaadja az összes company nevét, és ID-jét az adatbázisból.
* /api/company/mycompany (GET) Visszaadja annak a felhasználónak cégét, minden hozzátartozó adattal, amelyik be van lépve.
* /api/company/addcompany (POST) Ez egy cég nevet vár, és visszaadja a kreált company nevét, és ID-jét, ha sikeresen bekerült az adatbázisba.
* /api/company/editcompany (POST) Ez egy company ID-t és az új cégnevet várja, visszaküldi az új cég nevét és a company ID-t, ha sikeresen bekerült az adatbázisba.
* /api/company/addstore (POST) Ez egy company ID-t és egy boltnevet vár, visszaküldi a boltot, ami létrejött, ha sikeresen bekerült az adatbázisba.
* /api/company/editstore (POST) Ez egy store ID-t és egy boltnevet vár, visszaküldi a módosult boltot, ha az sikeresen bekerült az adatbázisba.

Az Item Controller APIjai:

* /api/item/additemtocompany (POST) Ez egy vonalkódot, egy terméknevet, egy beszerzési árat, egy leírást és egy képet vár base64-gyel kódolva. Visszaad egy teljes CompanyItem-et.
* /api/item/additemtostore (POST) Ez egy companyItem ID-t vár, egy store ID-t, egy leltáron lévő mennyiséget, egy alap eladási árat és egy kedvezményt. Visszaad egy teljes StoreItem-et.
* /api/item/editcompanyitem (POST) Ez egy companyItem ID-t egy terméknevet, egy beszerzési árat, egy leírást és egy képet vár base64-gyel kódolva. Visszaadja a módosított CompanyItem-et.
* /api/item/editstoreitem (POST) Ez egy storeItem ID-t, egy leltáron lévő mennyiséget, egy alap eladási árat és egy kedvezményt vár. Visszaadja a módosított storeItem-et
* /api/item/orderitem (POST) Ez egy storeItem ID-t és egy darabszámot vár. Visszaadja a módosított storeItem-et

A Sale Controller APIjai:

* /api/sales/getsales (POST) Ez két dátumot vár, egy tól-ig időtartamot. Visszaadja a bejelentkezett felhasználó cégének az eladásait a megadott időintervallumon.
* /api/sales/getsalesbystore (POST) Ez két dátumot vár, egy tól-ig időtartamot, és egy store ID-t. Visszaadja a store-hoz tartozó eladásokat a megadott időintervallumon.
* /api/sales/addsale (POST) Ez egy mennyiséget, és egy storeItem ID-t vár. Visszaadja a módosított storeItem-et.

A User Controller APIjai:

* /api/user/login (POST) Egy felhasználónevet, és egy jelszavat vár. Visszaadja a felhasználónévhez tartozó user entitást.
* /api/user/register (POST) Egy felhasználónevet, egy jelszavat, egy e-mailt és egy regisztrációs kódot vár. Visszaadja, a felhasználót, ha sikeres a regisztráció.
* /api/user/getregistrationcode (POST) Egy role-t, és egy company ID-t vár. Visszaad egy regisztrációs kódot.
* /api/user/addusertostore (POST) Egy user ID-t és egy store ID-t vár. Visszaadja a user entitást, ha sikeresen módosultak az adatok az adatbázisban.

### A Frontend Architektúrája

Az Angular egy nyílt forráskódú, platform független kliens-oldali web alkalmazás keretrendszer. Az utóbbi verziói pedig a TypeScript használatát is lehetővé teszi, amit tömören típusos JavaScriptként lehet jellemezni. A TypeScript Compilere nem meglepő módon JavaScriptre fordít, hiszen a web kliens oldali programozását nem lehet elképzelni a HTML, CSS és JavaScript mesterhármasa nélkül. Nem véletlen, hogy az angular-cli-jal generált komponens fájljai HTML, CSS és TypeScript fájlokból állnak. A data binding, vagyis adat kötések segítségével a dinamikus oldal generálás még egyszerűbb, mint a natúr JavaScripttel. Ez nagyban segíti a fejlesztési munkát.

A frontenden is megtalálhatóak az MVC nyomai. A modellek, vagyis a típusok, amiket kapunk a backendtől megtalálhatóak a model mappában. A kommunikációt service mappában található szervizek képzik, ám itt találhatóak security-ért felelős guard fájlok, amik megakadályozzák, hogy bizonyos oldalnak hiába tudod a linkjét, át fog róla irányítani a főoldalra. A maradék pedig a nézetért felelős. Ezeket component fájlokként találjuk meg, minden ilyen komponenshez 4 típusú fájl tartozik. 1 HTML, amiben a nevéből következő módon a szokásos html szintaxisával építhetjük fel, hogy mit lássunk a komponensen. Egy CSS fájl, ami csak az azonos komponenshez tartozó html fájlra vonatkozik, ez könnyíti a változatos stílusok alkalmazását. Illetve két TypeScript fájlból, amiből az egyik a teszteléshez van. A másikban lehet megírni, hogy a komponenssel mi történjen inicializáláskor. Lehet benne definiálni függvényeket, amiket hasonló módon a javascript függvényekkel meg lehet hívni a html fájlból. És lehet benne változókat létrehozni, bizonyos események után módosítani, amiket a html a data binding segítségével meg tud jeleníteni.

### Frontend modulok

Fontos megemlíteni az Angular által beépített modulokat, amiket a package.json fájlban lehet bővíteni, a modul nevével, és verziószámával. Ezek után mindig fontos, hogy újra kiadjuk az *npm install* parancsot. Ezek a főbb modulok, amik fel lettek használva a frontend fejlesztése közben:

* angular/core A komponensek felépítéséért, az eseménykezelésért, és az esetleges újrarenderelésért felelős.
* angular/forms A formok könnyebb használatáért, validációjáért és data-binding használatáért felelős.
* angular/router Ezzel navigálunk komponensek között, és ennek köszönhetően tűnik úgy, hogy más oldalon vagyunk a link szerint, pedig az egész alkalmazás ugyanazon az oldalon érhető el, csak más komponenst jelenítünk meg.
* angular/http Ennek segítségével tudunk http szerint kommunikálni a backenddel.
* ngx-pagination A táblázatok lapozhatóságáért felelős modul.

## Fejlesztést segítő eszközök

Kezdjük a legfontosabbal: Internet. Egy viszonylag kis csomagú mobilnettel rengeteg hasznos írásos anyagot tudsz találni, ami a problémádat részben, vagy egészben megoldja. Ha jól tud keresni az ember, akkor kevés olyan hibaüzenet van, amit nem tud egy fejlesztő viszonylag rövid idő alatt dekódolni, és megtalálni a hiba forrását.

Papír és ceruza. Egy ötletet nem szabad hagyni elszállni, addig kell leírni, amíg el nem felejti az ember. Érdemes a feladatokat is beosztani, hogy mikor mit csináljon a fejlesztő ugyanis, ha minden alkalommal, amikor leül az ember foglalkozni az alkalmazással, azon kezd el gondolkodni, hogy mit csináljon, rengeteg időt veszít. Egy egyszemélyes projekten elég egy papírra felírni a teendőket, majd ezeknek adni egy sorszámot. Lehet, hogy a munka közbe rájön az ember, hogy valamit még meg kellett volna csinálni már korábban, de az ezen való bosszankodás, és gondolkodás még mindig kevesebb időt vesz igénybe, ha elterveztük a folyamatot. Az időnél pedig nincs fontosabb.

A fejlesztő legjobb barátja a szövegszerkesztő, vagy IDE, amiben a műveit készíti. Nekem ebben a projektben a Visual Studio Code és az IntelliJ volt ez. Mindkettőben szimpatikus, hogy van beépített terminál, ami segíti a fejlesztőt, hogyha csak egy monitorja van. Az IntelliJ-ben hasznos a gyakran használt kódok generálása az osztályokhoz, azaz a getter setter constructor és társai.

A verziókezelő hasznos, ha több különböző ember dolgozik a projekten, de akkor is jó, ha nem tudsz mindig a saját gépeden dolgozni. Ha van idő és internet kapcsolat, akkor bármelyik publikus gépen tudsz haladni, vagy egyetemi előadások között a hallgatóknak fenntartott gépterem is jól használható erre a célra.

Végezetül a Postman nevű alkalmazás segített még sokat, amikor a backendet írtam, és még nem volt kliens oldal. Ezzel tudtam könnyen tesztelni a végpontokat. Megadható benne a JSON objektum, a fejléc, így létrehozható vele az API híváshoz szükséges környezet. A küldés után pedig látjuk azt is, hogyan reagált az elküldött objektumra a backend.

## Tesztek

Az automatikus teszteléshez JUnit keretrendszer van használva.

### Egységtesztek

A backend repositoryjait teszteltem egységtesztekkel, ugyanis az adatbázis műveletek a legfontosabbak, hogy jól működjenek. Lényegi logika a service-ekben nincs, ott csak annak az ellenőrzése folyik, hogy van-e joga a felhasználónak az adott műveletet végrehajtani, vagy nincs.

### Manuális tesztek

Az adatbázis műveletek egységtesztekkel lettek lefedve. A frontendhez nem készültek egységtesztek, ezeknek a főbb funkcióit manuálisan teszteltem, melyek részleteit itt ismertetem.

* Bejelentkezés
  + Előfeltétel: már van regisztrált felhasználó, a bejelentkezés oldalon van, kitöltötte a felhasználónév-jelszó párost és a Bejelentkezés gombra kattint.
  + Hibás kitöltés esetén a felhasználó hibaüzenetet kap, és a rendszer nem engedi belépni.
  + Helyes adatok esetén a program beengedi, a fejléc módosul a felhasználó jogai szerint.
* Cég hozzáadása
  + Előfeltétel: ADMIN jogosultságú a felhasználó, be van jelentkezve, és a cég hozzáadása oldalon van. Kitöltötte a mezőt a cég nevével, és a Mentés gombra kattint.
  + Ha már létezik ilyen cégnév a felhasználó kap róla egy hibaüzenetet, és nem kerül be az adatbázisba az adat.
  + Ellenkező esetben kap egy Sikeresen hozzáadva üzenetet az oldalon, a cég bekerül az adatbázisba.
* Cég szerkesztése
  + Előfeltétel: ADMIN jogosultságú felhasználó, be van jelentkezve, és a cég módosítása oldalon van. Kiválasztotta a módosítani kívánt céget a legördülő menüből, átírta a nevet, és rákattint a Mentés gombra.
  + Ha már létezik ilyen cégnév a felhasználó kap róla egy hibaüzenetet, és nem módosul az adat.
  + Ellenkező esetben kap egy Sikeres módosítás üzenetet az oldalon, a cég neve módosul az adatbázisban.
* Kód generálás
  + Előfeltétel: ADMIN, vagy COMPANY\_ADMIN jogosultságú a felhasználó, be van jelentkezve, és a kód generálás oldalon van. ADMIN esetében ki kell választani a céget is, amelyikhez generálni fogja a kódot. Mindkét esetben választani kell egy felhasználói típust, majd rá kell kattintani a Generálás gombra.
  + Ha ADMIN generál ADMIN-nak kódot, akkor a cég választó lista eltűnik.
  + Sikeres generálás esetén megjelenik a regisztrációs kód. A mezőbe kattintva, rögön a vágólapra kerül.
* Kijelentkezés
  + Előfeltétel: A felhasználó be van jelentkezve és rákattint a kijelentkezés gombra.
  + Az alkalmazás frontend része visszatér a bejelentkezés előtti állapotra.
* Regisztráció
  + A felhasználó nincs bejelentkezve, a regisztrációs oldalon van. Kitöltötte az űrlapot és rányomott a Regisztráció gombra.
  + Hiba esetén a hiba okáról kap üzenetet a felhasználó. Több hiba esetén viszont csak az első megtalált hibát fogja a szerver mindig jelezni.
  + Sikeres regisztráció esetén az alkalmazás átnavigál a kezdőoldalra, a regisztrált felhasználó bekerül az adatbázisba.
* Bolt hozzáadása
  + Előfeltétel: COMPANY\_ADMIN jogosultságú a felhasználó, be van jelentkezve, és a bolt hozzáadása oldalon van. Kitöltötte a mezőt a bolt nevével, és címével, majd a Mentés gombra kattint.
  + Hiba esetén az alkalmazás jelzi a hiba okát. Az adatbázisba nem kerül be az adat.
  + Ellenkező esetben kap egy Sikeres hozzáadás üzenetet az oldalon, a bolt bekerül az adatbázisba, és hozzá van rendelve a céghez.
* Bolt szerkesztése
  + Előfeltétel: COMPANY\_ADMIN jogosultságú felhasználó, be van jelentkezve, és a bolt módosítása oldalon van. Kiválasztotta a módosítani kívánt céget a legördülő menüből, átírta a nevet, és/vagy címet, majd rákattint a Mentés gombra.
  + Hiba esetén az alkalmazás jelzi a hiba okát. Az adatbázisban nem módosul az adat.
  + Ellenkező esetben kap egy Sikeres módosítás üzenetet az oldalon, a bolt adatai módosulnak az adatbázisban.
* Eladások listázása
  + Előfeltétel: COMPANY\_ADMIN jogosultságú a felhasználó, be van jelentkezve, és az eladások oldalon van. Kiválasztotta, hogy mely egységre kívánja végrehajtani a lekérdezést, illetve, hogy mely időszakra, majd rákattint a Lekérdez gombra.
  + Hiba esetén az alkalmazás jelzi a hiba okát.
  + Sikeres lekérdezés esetén, látni fogjuk a kijelölt egységhez tartozó eladásokat a megadott időszakon, a táblázat alján pedig a listázott eladások összes profitját tekinthetjük meg.
* Felhasználó bolthoz rendelése
  + Előfeltétel: COMPANY\_ADMIN jogosultságú a felhasználó be van jelentkezve és a felhasználók oldalon van. Kiválasztott 1 SELLER típusú felhasználónak egy másik boltot, mint ami eredetileg be volt állítva, amikor az oldalra navigált. Rányomott a mentés gombra. Megj.: a visszaállítás gomb csak a mentés gomb megnyomása előtt tudja kifejteni a hatását.
  + Hiba esetén az alkalmazás jelzi a hiba okát, a módosítás nem került be az adatbázisba.
  + Sikeres mentés esetén az adatbázisban beállított érték módosult.
* Termék hozzáadása a céghez
  + Előfeltétel: COMPANY\_ADMIN, vagy SELLER jogosultságú a felhasználó, be van jelentkezve, és a Termék hozzáadása a céghez oldalon van. Ha képet akar hozzáadni, jelenleg az csak PNG formátumban teheti meg. A kitöltés befejeztével rákattint a Mentés gombra.
  + Hiba esetén annak okáról figyelmeztetésben kap értesítést a felhasználó. Az adatbázisba nem kerül be a termék.
  + Sikeres Mentés esetén megjelenik egy Sikeresen hozzáadva felirat az oldalon. Az adatbázisba bekerül a termék.
* Cég termékének módosítása
  + Előfeltétel: COMPANY\_ADMIN, vagy SELLER jogosultságú a felhasználó, be van jelentkezve, és a Cég termékei oldalon van. Van már céghez hozzáadott termék. Rákattint a módosítani kívánt termék sorában a Módosít gombra, a megjelenő inputmezőkben módosítja az adatokat az általa kívántra, majd megnyomja a Mentés gombot. Megj.: a képet törölni nem lehet azzal, hogy kép feltöltés nélkül próbáljuk módosítani a terméket.
  + Hiba esetén az alkalmazás figyelmeztetésben írja meg a hiba okát, az adatbázisban a termék nem módosul.
  + Sikeres mentés esetén a termék módosul az adatbázisban, az inputmezők eltűnnek a mentett termék sorából, és az új értékek jelennek meg benne.
* Termék bolthoz kötése (1)
  + Előfeltétel: COMPANY\_ADMIN, vagy SELLER jogosultságú a felhasználó. Ha SELLER, akkor a felhasználót előtte már bolthoz kellett rendelni. A felhasználó be van jelentkezve, és a cég termékei oldalon van. Van olyan cég termék, ami még nincs bolthoz rendelve. Ekkor a táblázat utolsó oszlopában megjelenik a még nem bolthoz rendelt terméknek egy gomb és COMPANY\_ADMIN esetében egy legördülő választási lista, melyben azok a bolt(ok) szerepelnek, amelyhez a terméket még nem adták hozzá. Kiválaszt a boltok közül egyet a felhasználó, és az alatta lévő gombra kattint.
  + Az alkalmazás átnavigál egy űrlaphoz, ami felett megjelenik a bolthoz adni kívánt termékről néhány adat.
* Termék bolthoz kötése (2)
  + Előfeltétel: a Termék bolthoz kötése (1) eset által a felhasználó a Termék hozzáadása a bolthoz oldalon van. A felhasználó kitölti az űrlapot, majd a Mentés gombra kattint.
  + Hiba esetén a felhasználót az alkalmazás figyelmeztetésben tájékoztatja a hiba okáról. A termék nem lesz a bolthoz kötve az adatbázisban.
  + Sikeres mentés esetén az oldalon megjelenik a Sikeresen hozzáadva felirat.
* Bolt termékének módosítása
  + Előfeltétel: COMPANY\_ADMIN, vagy olyan SELLER jogosultságú a felhasználó, akihez rendeltek már boltot. A felhasználó be van jelentkezve, a Bolt termékei oldalon van, COMPANY\_ADMIN esetében választott egy boltot a legördülő listában. A bolthoz már van termék kötve. A módosítani kívánt termék sorában a felhasználó rákattint a Módosít gombra, a megjelenő inputmezőket kitölti az új adatokkal, majd a Mentés gombra kattint.
  + Hiba esetén az alkalmazás figyelmeztetésben írja meg a hiba okát, az adatbázisban a termék nem módosul.
  + Sikeres mentés esetén a termék módosul az adatbázisban, az inputmezők eltűnnek a mentett termék sorából, és az új értékek jelennek meg benne, ha a kereső mező üres. Ellenkező esetben módosítsuk a kereső mezőt, így megjelennek a helyes értékek a táblázatban.
* Bolti termék rendelése
  + Előfeltétel: COMPANY\_ADMIN, vagy olyan SELLER jogosultságú a felhasználó, akihez rendeltek már boltot. A felhasználó be van jelentkezve, a Bolt termékei oldalon van, COMPANY\_ADMIN esetében választott egy boltot a legördülő listában. A bolthoz már van termék kötve. A rendelni kívánt termék sorában kitölti a Rendelés oszlopot, majd a Rendelés gombra kattint.
  + Hiba esetén az alkalmazás figyelmeztetésben írja meg a hiba okát, az adatbázisban a termék nem módosul.
  + Sikeres rendelés esetén, a Raktáron oszlop módosul, a rendelt darabszámmal, ha a kereső mező üres. Ellenkező esetben módosítsuk a kereső mezőt, és a Raktáron oszlop módosulni fog a rendelt darabszámmal. Az adatbázisban a termék módosul.
* Bolti termék eladása
  + Előfeltétel: COMPANY\_ADMIN, vagy olyan SELLER jogosultságú a felhasználó, akihez rendeltek már boltot. A felhasználó be van jelentkezve, a Bolt termékei oldalon van, COMPANY\_ADMIN esetében választott egy boltot a legördülő listában. A bolthoz már van termék kötve. Az eladni kívánt termék sorában kitölti az Eladás oszlopot, majd az Eladás gombra kattint.
  + Hiba esetén az alkalmazás figyelmeztetésben írja meg a hiba okát, az adatbázisban a termék nem módosul, az eladás nem kerül be az adatbázisba.
  + Sikeres eladás esetén, a Raktáron oszlop módosul, az eladott darabszámmal, ha a kereső mező üres. Ellenkező esetben módosítsuk a kereső mezőt, és a Raktáron lévő oszlop módosulni fog az eladott darabszámmal.

## További fejlesztési lehetőségek

Az alkalmazást rengeteg irányba lehet továbbfejleszteni. Az elsődleges szempont az volt, hogy az adatokat eltároljuk, így egy egységes nagy adatbázist létre lehet vele hozni, de mit érnek az adatok, ha nem osztjuk meg az emberekkel? Így a legfontosabb lehetőség, amit érdemes kihasználni, hogy a nem regisztrált felhasználók számára láthatóak legyenek az adatok, miszerint mely terméket hol lehet megvenni, és mennyiért.

A képek tárolási módját is van hova fejleszteni. Legjobb megoldás az lenne, ha lenne egy külön szerver, ahova ezeket a felhasználók feltöltik a képeket, amiket meg szeretnének jeleníteni a termékről, és ennek a linkjét mentjük bele az adatbázisba. Ezzel nagy mértékben csökkenne az adatforgalom.

A cégek számára egy nagyobb feldolgozót érdemes létrehozni, hogy ne egyesével kelljen hozzáadni a termékeket, hanem megfelelően kiimportálva a saját adatbázisukból például CSV formátumban, rögtön feltölthessék az eladni kívánt portékájukat.

További segítség lehet a termékfelvételhez egy mobil alkalmazást létrehozni, így nem kell amiatt aggódni, hogy elírja a vonalkódot. Létre lehetne hozni egy listát a felhasználó számára, amiben gyűjti a beolvasott vonalkódokat, esetleg társít hozzá egy képet, és amikor oda ül a gép elé, sokkal gördülékenyebben tud menni az adatfelvitel.

A nem céges felhasználó rétegnek is lehet csinálni egy mobilverziót, amibe bele lehetne rakni egy térképet, amin szerepelnek a boltok, így ki tudja választani, hogy melyik eladó helyiség van hozzá közel, ami árulja a keresett terméket a számára megfelelő áron. Vonalkód leolvasót is lehet bele tenni így, ha egy boltban meglát egy terméket, ami kell a felhasználónak, de túlárazottnak találja, akkor könnyedén rá tud keresni, hogy hol találja meg olcsóbban.

Céges eladási versenyeket is lehet rendezni ezzel az alkalmazással, ugyanis az alkalmazás tárolja, hogy ki adja el a terméket. Így lehet írni olyan lekérdezést, hogy adott időszakra ki volt a legjobb eladó a cégben vagy boltban.