|  |  |
| --- | --- |
|  | **EÖTVÖS LORÁND TUDOMÁNYEGYETEM**  **INFORMATIKAI KAR**  [**Programozáselmélet és Szoftvertechnológia Tanszék**](http://pszt.inf.elte.hu/) |

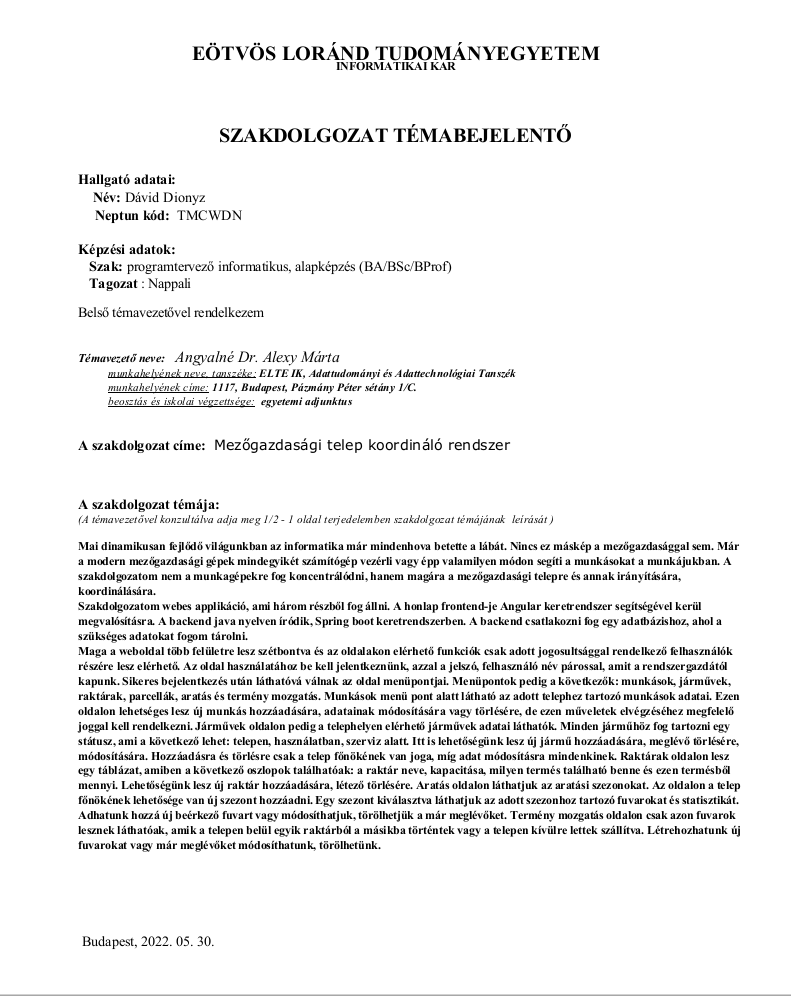
**Mezőgazdasági telep koordináló rendszer**

**Témavezetők:** **Szerző:**

Angyalné Dr. Alexy Mária Dávid Dionyz

egyetemi adjunktus programtervező informatikus BSc

Budapest, 2022



[1 Bevezetés 4](#_Toc626750755)

[2 Felhasználói dokumentáció 5](#_Toc1900666328)

[2.1 Rendszer követelmény 6](#_Toc439359423)

[2.2 Alkalmazás használata 6](#_Toc615727084)

[2.2.1 Főoldal 6](#_Toc1694383265)

[2.2.2 Menü 6](#_Toc1375592130)

[2.2.3 Oldalak 7](#_Toc1984884150)

[2.2.3.1 Mezőgazdasági területek 8](#_Toc2013289191)

[2.2.3.1 Raktárak 9](#_Toc756800256)

[2.2.3.1 Munkások 11](#_Toc1060894462)

[2.2.3.1 Járművek 13](#_Toc1744664899)

[2.2.3.1 Aratási napló 15](#_Toc762907220)

[2.2.3.1 Termény mozgatás 17](#_Toc1198494627)

[3 Fejlesztői dokumentáció 18](#_Toc1379860999)

[3.1 Frontend 19](#_Toc1733932983)

[3.1.1 Tervezés 19](#_Toc872457515)

[3.1.2 Megvalósítás 22](#_Toc1920282134)

[3.1.2.1 Weboldal 23](#_Toc748918048)

[3.1.1.2 Angular material 24](#_Toc195834723)

[3.1.3 Tesztelés 25](#_Toc2104682978)

[3.1.3.1 Bejelentkezés 26](#_Toc204545214)

[3.1.3.2 Főoldal 26](#_Toc220851068)

[3.1.3.3 Földek 27](#_Toc1846484753)

[3.1.3.4 Raktárak 29](#_Toc1273090490)

[3.1.3.5 Munkások 30](#_Toc1569154339)

[3.1.3.6 Járművek 31](#_Toc1413302702)

[3.1.3.7 Aratási napló 32](#_Toc287969090)

[3.1.3.8 Termény mozgatás 35](#_Toc680388292)

[3.2.1 Tervezés 37](#_Toc432707578)

[3.2.2 Megvalósítás 37](#_Toc1981049767)

[3.2.2.1 Kontrollerek 37](#_Toc517826415)

[3.2.2.1 Szervizek 42](#_Toc415167345)

[3.3 Adatbázis 50](#_Toc2042660628)

[4 Összegzés 52](#_Toc470293285)

[5 Irodalomjegyzék 53](#_Toc667097531)

# 1 Bevezetés

A mai dinamikusan fejlődő világunkban az informatika már mindenhova betette a lábát. Nincs ez másképp a mezőgazdaságban sem. A modern mezőgazdasági gépek mindegyikét számítógép vezérli vagy épp valamilyen módon segíti a munkásokat a munkájukban. A szakdolgozatom nem a munkagépekre fog koncentrálni, hanem magára a mezőgazdasági telepre és annak irányítására, koordinálására.

A dolgozat célja egy olyan felület megalkotása, amely alkalmas a farmhoz tartozó raktárak, járművek, mezőgazdasági területek és az ott dolgozó alkalmazottak menedzselésére.

Ezen felül lehetőséget fog biztosítani az oldal aratási naplók vezetésére, melyben a különböző szállítmányok áttekinthetőek.

Természetesen mivel egy farmon gyakran megesik, hogy mozgatják az ott tárolt terményeket így az alkalmazás erre is fog kínálni megoldást.

# 2 Felhasználói dokumentáció

## 2.1 Rendszer követelmény

Az alkalmazás három részből épül el, így az egyes komponenseknek eltérő rendszerigénye van.

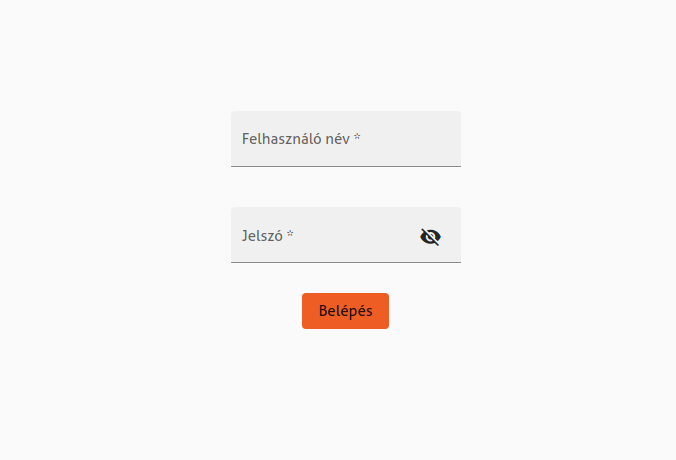
A frontend-nek azaz a weboldalon látható felhasználói felületnek a követelménye, hogy a felhasználó számítógépe kapcsolódjon az internethez és telepítve legyen rajta egy böngésző az alábbi verziók közül:

* Google Chrome - 96.0.4664.110 vagy újabb verzió
* Firefox - 95.0 vagy újabb verzió
* Edge - 96.0.1054.62 vagy újabb verzió
* Safari - 15.0 vagy újabb verzió

## 2.2 Alkalmazás használata

### 2.2.1 Főoldal

Az alkalmazás használatához be kell jelentkezni (1. ábra). Bejelentkezéshez a rendszergazdától kapott email és jelszó páros használható. Sikeres bejelentkezés után a “üdvözlő” oldalra navigál az alkalmazás.

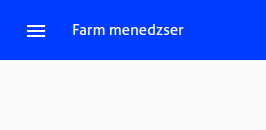


1. ábra: Bejelentkezés

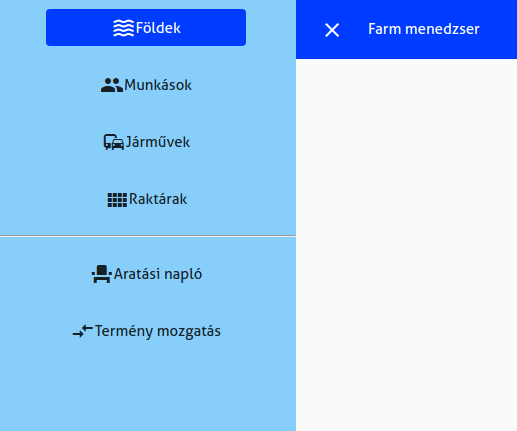
### 2.2.2 Menü

A menü a bal felső sarokban található gomb segítésével nyitható meg (2. ábra). A gomb megnyomásával a képernyő bal oldaláról becsúszik a menü (3. ábra), mely segítségével navigálhatunk az oldalon. A következő menüpontok találhatóak:

* Földek
* Raktárak
* Munkások
* Járművek
* Aratási napló
* Terménymozgatás



2. ábra: Menü helye

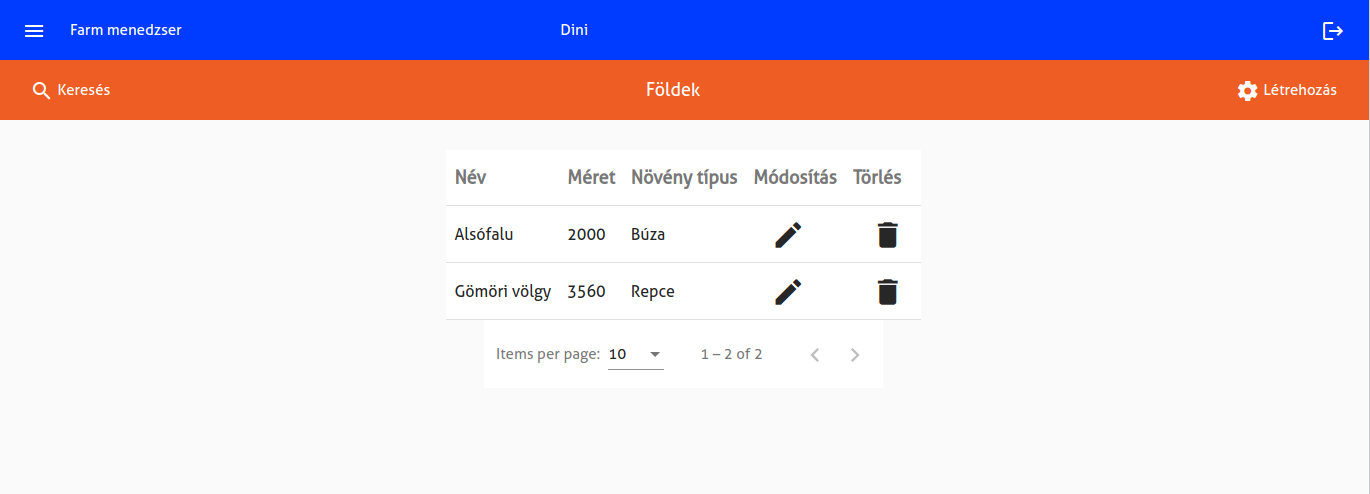


3. ábra: Menü

### 2.2.3 Oldalak

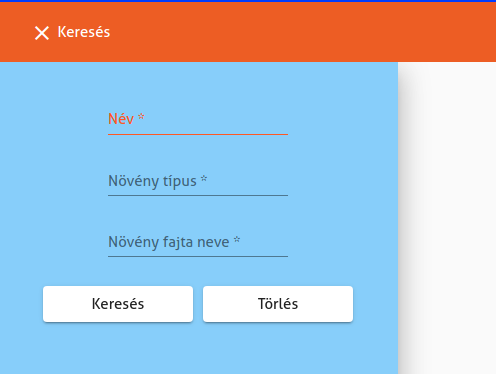
#### 2.2.3.1 Mezőgazdasági területek

A mezőgazdasági területek oldalon láthatjuk a farm-hoz tartozó szántóföldeket egy táblázatban (4. ábra). A táblázatban mindegy egyes terület neve, mérete és a jelenleg termesztett növény fajtáját láthatjuk. A táblázat utolsó két oszlopa pedig lehetőséget biztosít a sorban szereplő terület adatainak módosítására vagy törlésére. Az adatok módosításához és törléséhez megfelelő jogosultság szükséges.



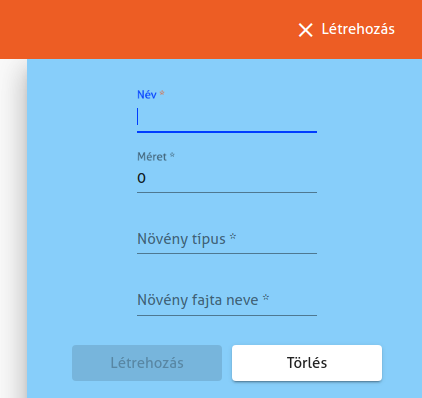
4. ábra: Mezőgazdasági területek oldal

Lehetőség van a táblázatban történő keresésre is a Menü alatt elérhető Keresés opcióra kattintva. (5. ábra). A terület neve, illetve a termesztett növény típusa és fajtája alapján lehet keresni az adatok között.



5. ábra: Földek keresés

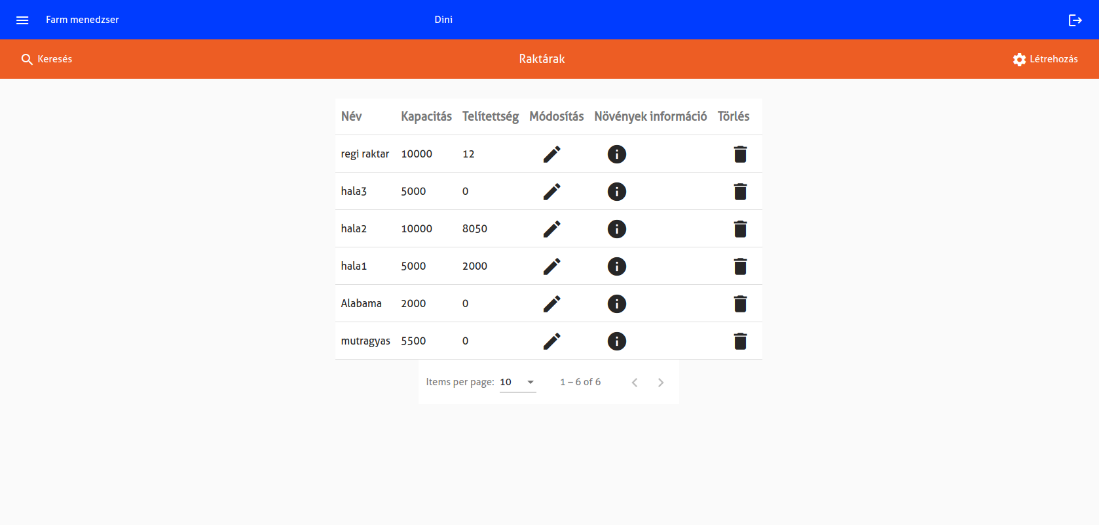
Ahhoz, hogy új földeket adjunk a rendszerbe megfelelő jogkörrel kell rendelkeznünk az oldalon. Létrehozás opciót az eddig megoldásokkal hasonló módon hozhatjuk elő. Létrehozás elhozásához tartozó gombot a képernyő jobb oldalán találhatjuk (6. ábra). Létrehozás során meg kell adnunk az új onnan létrehozandó föld nevét, méretét, és hogy esetlegesen milyen növény fajta van ültetve benne.



6. ábra: Földek létrehozása

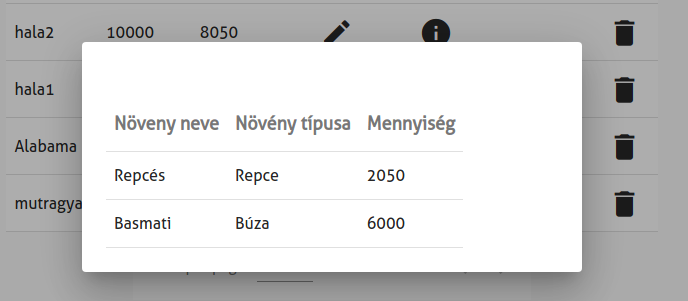
#### 2.2.3.1 Raktárak

A raktárak oldalon a farmhoz tartozó raktárak láthatóak egy táblázatban (7. ábra), ahol mindegy egyes raktár neve, kapacitása, telítettsége leolvasható. A táblázat utolsó két oszlopa pedig lehetőséget biztosít a sorban szereplő raktár adatainak módosítására vagy törlésére. Az adatok módosításához adminisztrátor jogkör szükséges.



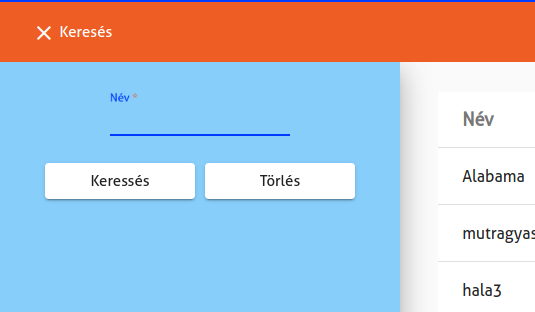
7. ábra: Raktárak oldal

A Növények információ gombra kattintva ellenőrizhető, hogy milyen terményből mennyi található az adott raktárban (8. ábra).



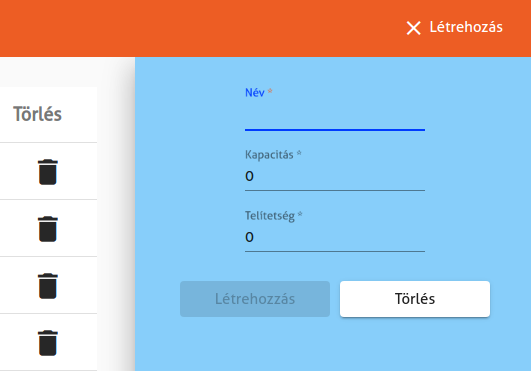
8. ábra: Növények információ

Hasonlóan a mezőgazdasági területek oldalhoz itt is lehetőség van keresésre a táblázatban, illetve megfelelő jogkörrel új raktár is létrehozható. Az oldalon a keresés a raktárak nevére korlátozódik (9. ábra).



9. ábra: Raktár keresés

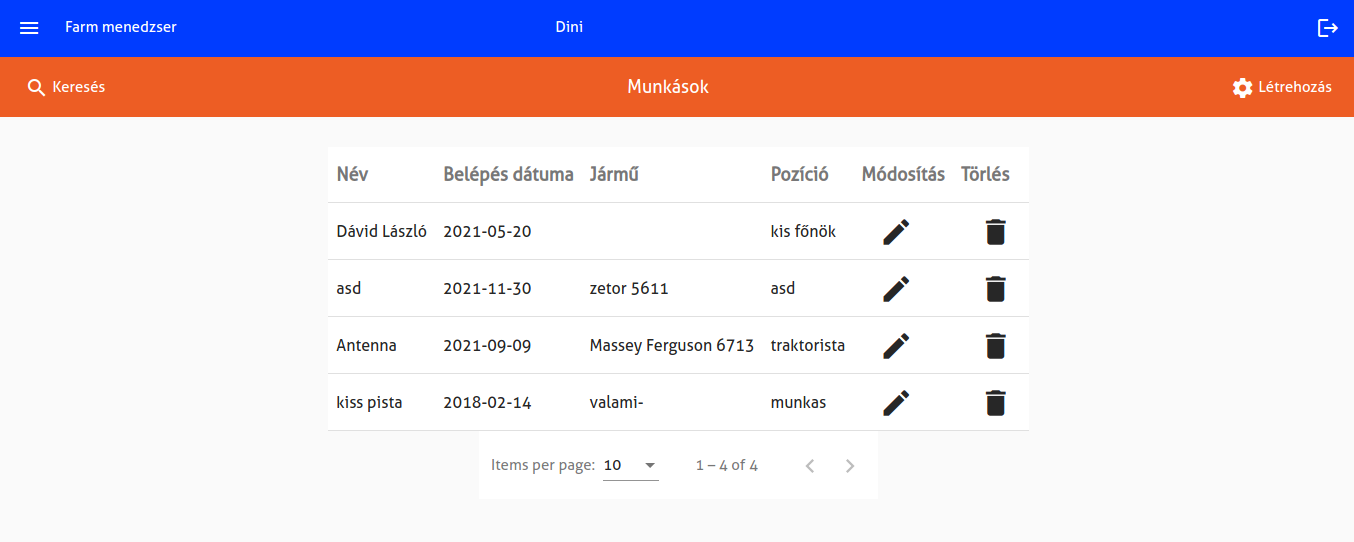
Új raktár létrehozásához szükséges megadni annak nevét, kapacitását és telítettségét (10. ábra).



10. ábra: Raktár létrehozása

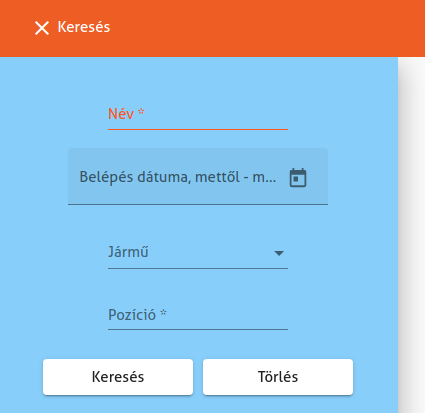
#### 2.2.3.1 Munkások

A munkások oldalon a farmon dolgozó alkalmazottak adatai láthatóak egy táblázatban (11. ábra), ahol szerepel az alkalmazott neve, belépésének dátuma, melyik jármű tartozik hozzá és hogy milyen pozícióban dolgozik a farmon. A táblázat utolsó két oszlopa pedig lehetőséget biztosít a sorban szereplő munkás adatainak módosítására vagy törlésére, melyhez adminisztrátori jogkör szükséges.



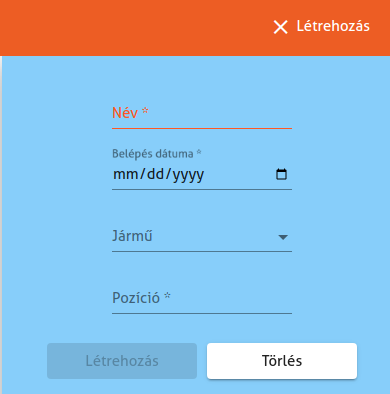
11. ábra: Munkások oldal

Természetesen ezen az oldalon is van lehetőség keresésre a táblázatban az alkalmazott neve, belépés dátuma, jármű és pozíció alapján. Megfelelő jogkör esetén új alkalmazott felvételére is van lehetőség. (12. ábra).



12. ábra: Munkás keresés

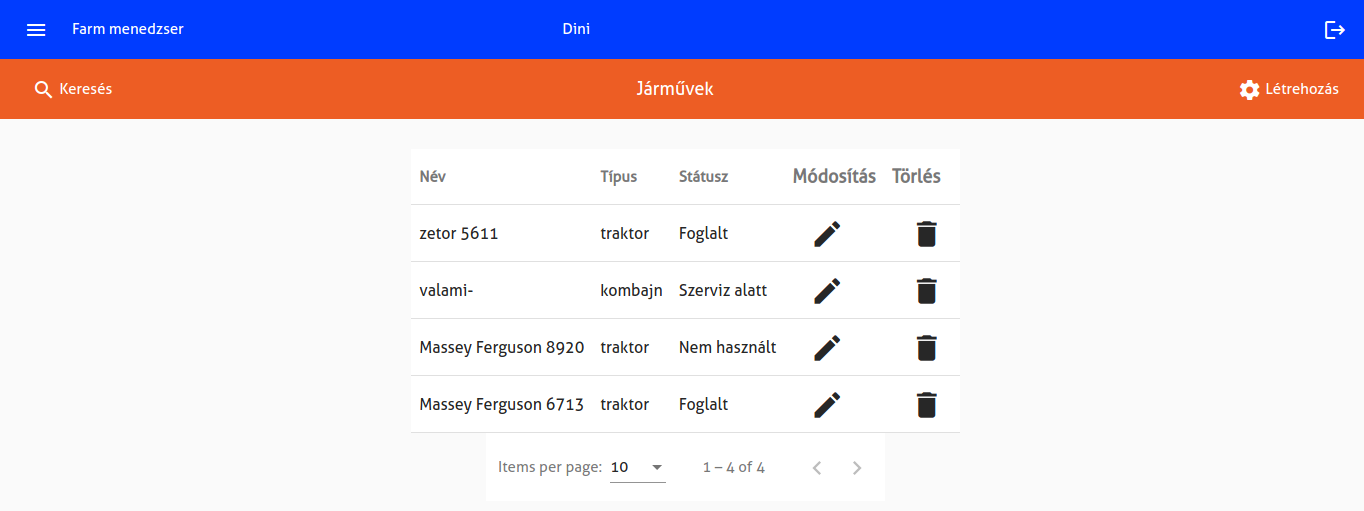
Új alkalmazott hozzáadása esetén szükséges megadni a munkás nevét, belépési dátumát a céghez, a járművet (opcionálisan amennyiben tartozik már hozzá) és a pozícióját a farmon (13. ábra).



13. ábra: Alkalmazott hozzáadása

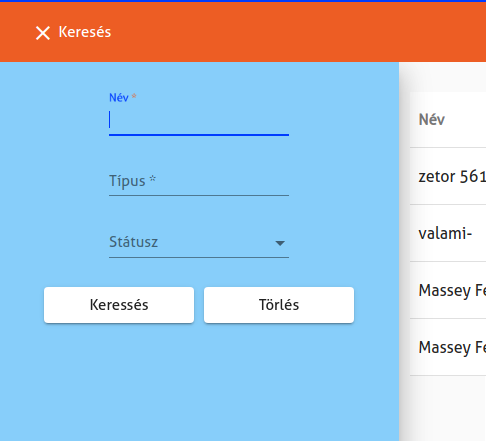
#### 2.2.3.1 Járművek

A járművek oldalon a farmhoz tartozó járművek adatai láthatóak egy táblázatban (14. ábra). A táblázatban szerepel a jármű neve, típusa és a státusza. A táblázat utolsó két oszlopa pedig lehetőséget biztosít a sorban szereplő jármű adatainak módosítására vagy törlésére. Az adatokat módosításához vagy törléséhez megfelelő jogkör szükséges.



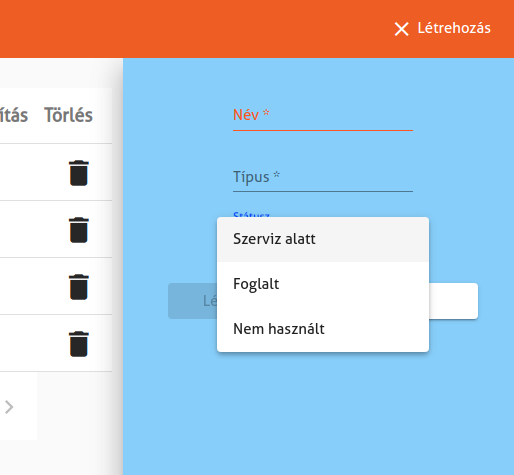
14. ábra: Járművek oldal

A jármű neve, típusa és státusza alapján lehetőség van keresésre. (15. ábra).



15. ábra: Jármű keresés

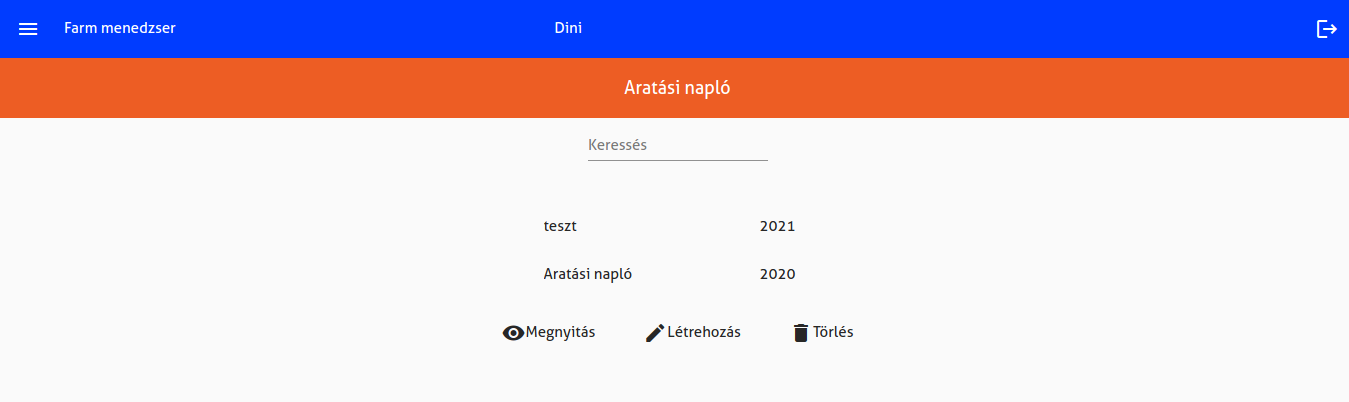
Megfelelő jogkör esetén pedig új jármű hozzáadására is van lehetőség a táblázathoz, ahol szükséges megadni a jármű nevét, típusát és az aktuális státuszát (foglalt, szerviz alatt, nem használt) (16. ábra).



16. ábra: Jármű létrehozása

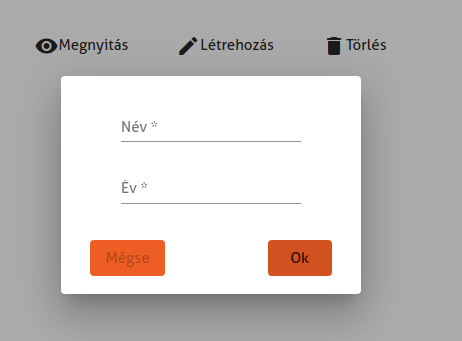
#### 2.2.3.1 Aratási napló

Az aratási napló oldalon egy lista nyílik meg, melyben a korábban létrehozott aratási naplók láthatóak (17. ábra). A lista felett egy keresőmező található, alatta pedig a megnyitás, létrehozás, törlés gombok.



17. ábra: Aratási naplók

A létrehozás gomb segítségével új aratási napló létrehozására van lehetőség, megnyomása után egy felugró ablakban adhatóak meg a szükséges adatok. (18. ábra). A lista valamelyik elemére kattintva egy elsődleges kiválasztás történik, ami után megnyitható vagy törölhető a kiválasztott aratási napló. Új naplót létrehozni vagy törölni csak az adminisztrátor joggal rendelkező felhasználók tudnak.



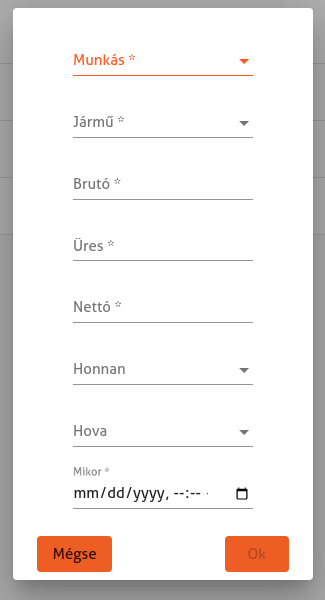
18. ábra: Aratási napló létrehozása

Egy aratási napló megnyitása után megjelenik egy táblázat, melyben a hozzá tartozó fuvaradatok láthatóak (19. ábra). Visszanézhető, hogy melyik alkalmazott szállította a fuvart, melyik járművel honnan hozta és hová vitte. Látható továbbá a jármű bruttó tömege (amikor beérkezik a a telepre terménnyel megrakodva) és az üres tömege (amikor lerakta a szállítmányt). Így a bruttó és az üres tömeg különbségéből kiszámítható a nettó tömeg.



19. ábra: Szállítmányok táblázat

A táblázat feletti keresőmezőben az alkalmazott, a jármű, a honnan és hova oszlopok tartalmára lehet keresni. Az új fuvar hozzáadása gomb megnyomása után egy felugró ablakban adhatjuk meg az új fuvar adatait (20. ábra).



20. ábra: Fuvar létrehozása

#### 2.2.3.1 Termény mozgatás

A termény mozgatás menüpont kiválasztása után megnyíló táblázatban láthatóak a farmra beérkezett, illetve egyik raktárból egy másikba átszállított tételek. (21. ábra).



21. ábra: Termények mozgatása

# 3 Fejlesztői dokumentáció

Az alkalmazás három komponensből épül fel: frontend, backend és adatbázis. A frontend “Angular” keretrendszerben [[1]](#Bookmark6) íródott, a backend pedig Java nyelven “Spring boot” keretrendszerben [[3]](#Bookmark4). Az adatbázis “PostgreSQL” azaz egy relációs adatbázis-kezelő rendszer.

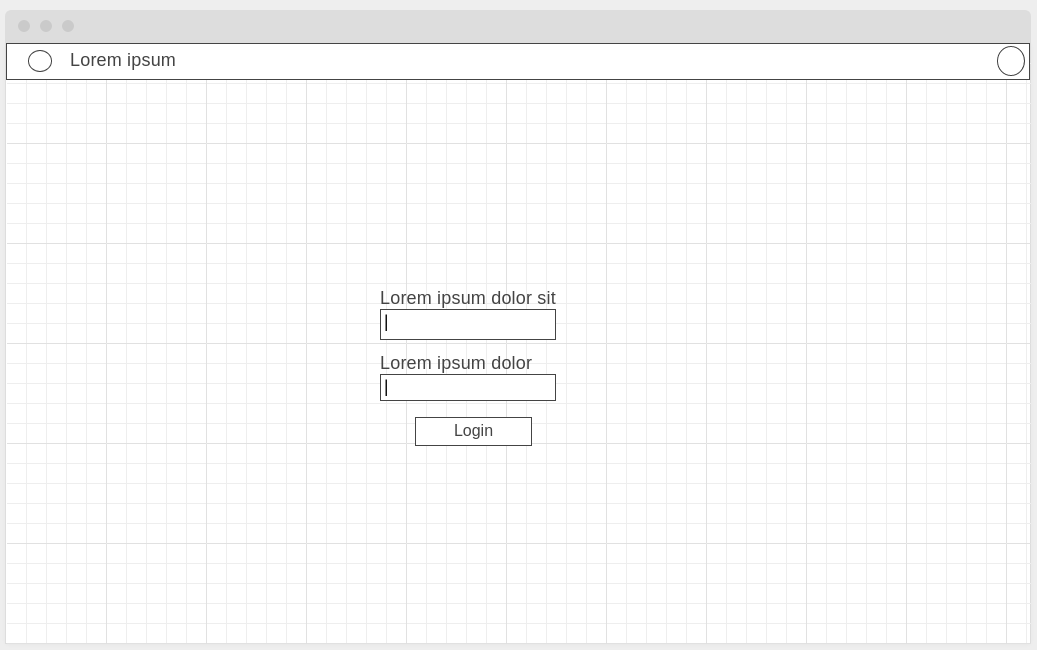
A fejlesztői dokumentációban ebben a sorendben fogom őket végig venni tervezés, megvalósítás és tesztelés szempontjából.

## 3.1 Frontend

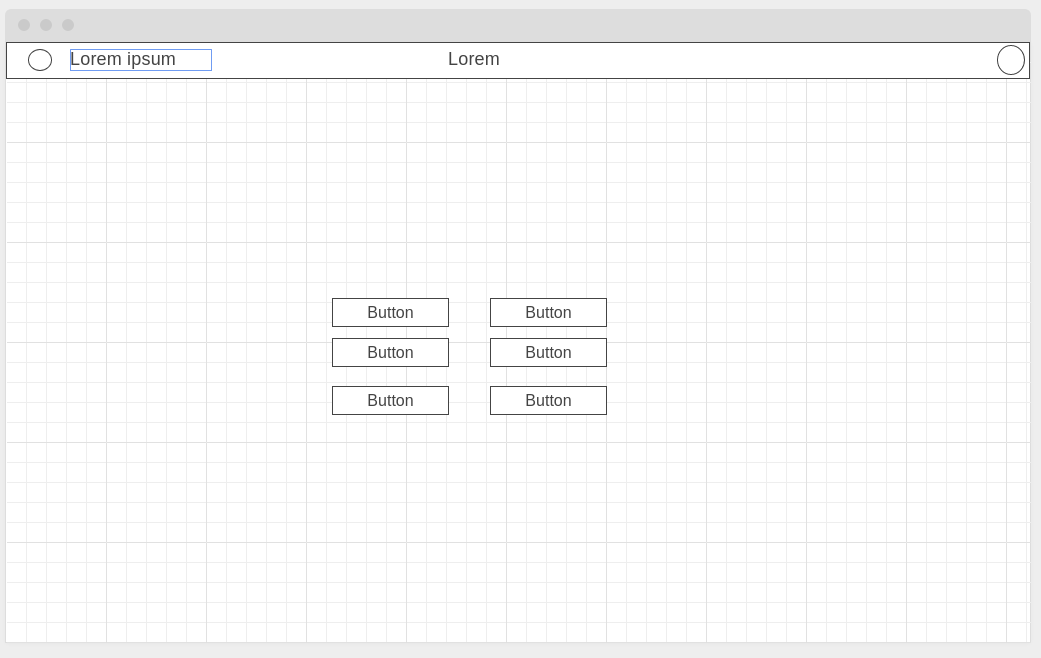
### 3.1.1 Tervezés

A frontend-nek azaz továbbiakban a webes felhasználói felületnek a következő oldalakat kell tartalmaznia, hogy megfeleljen a követelményeknek:

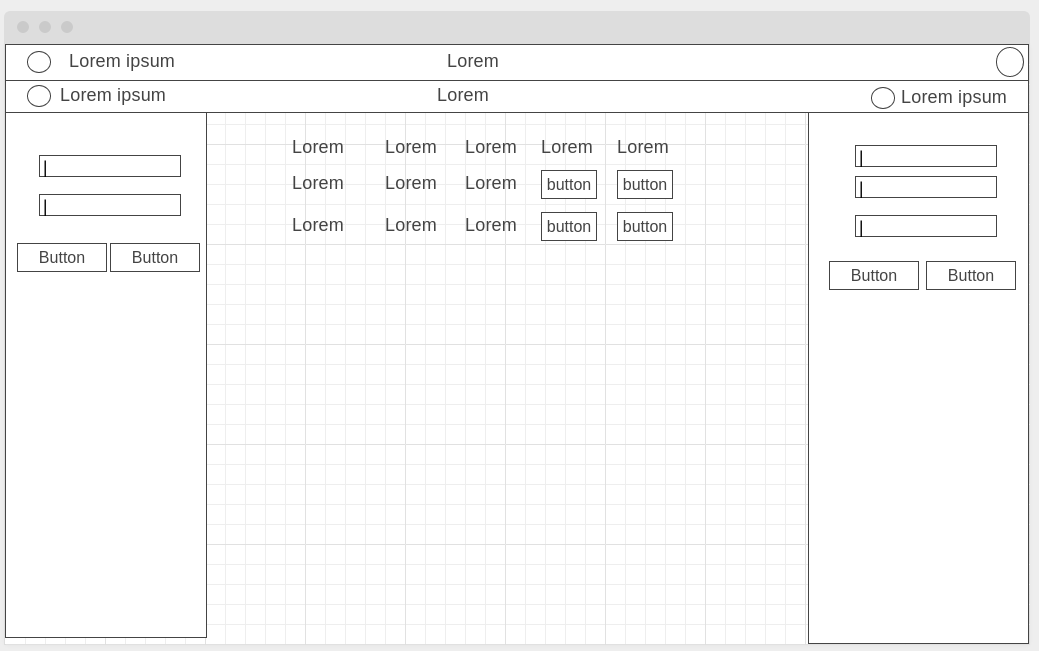
* Bejelentkezés
* Főoldal
* Földek oldal
* Raktár oldal
* Járművek oldal
* Munkások oldal
* Aratási napló oldal
* Terménymozgatás oldal



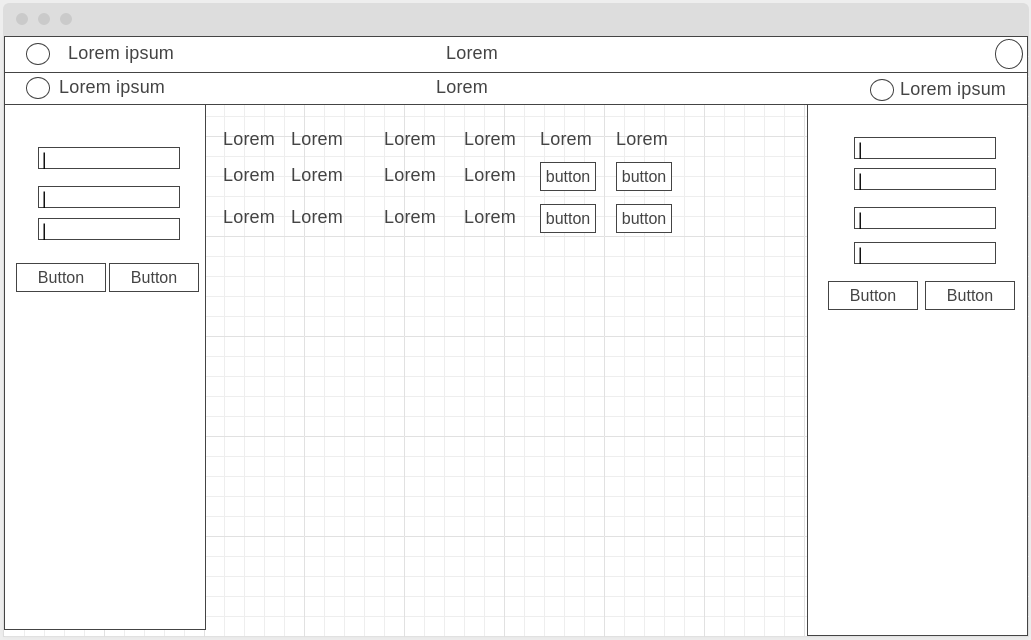
Bejelentkezés



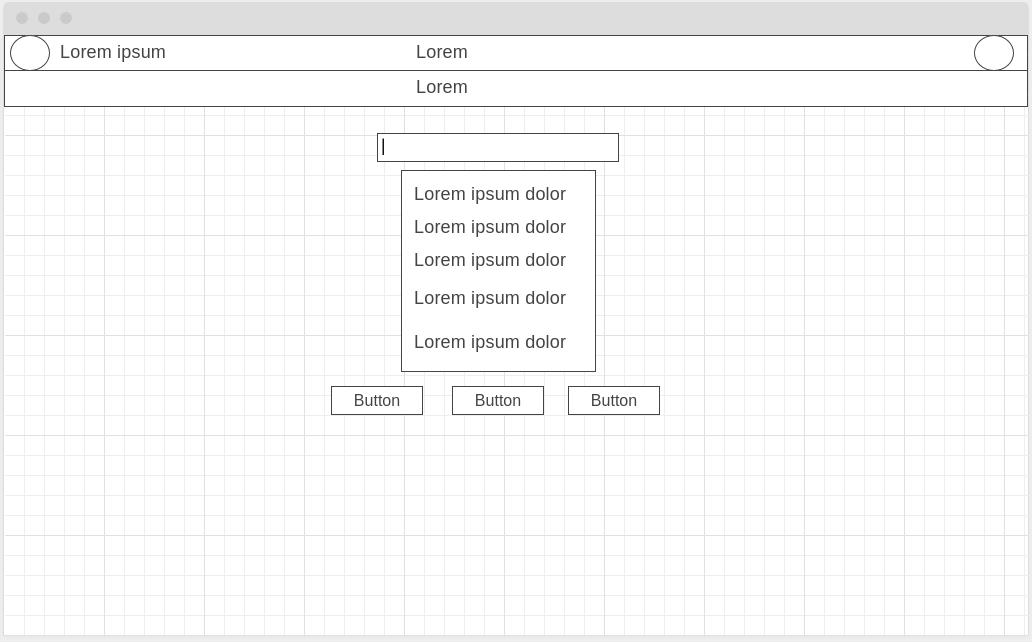
Főoldal



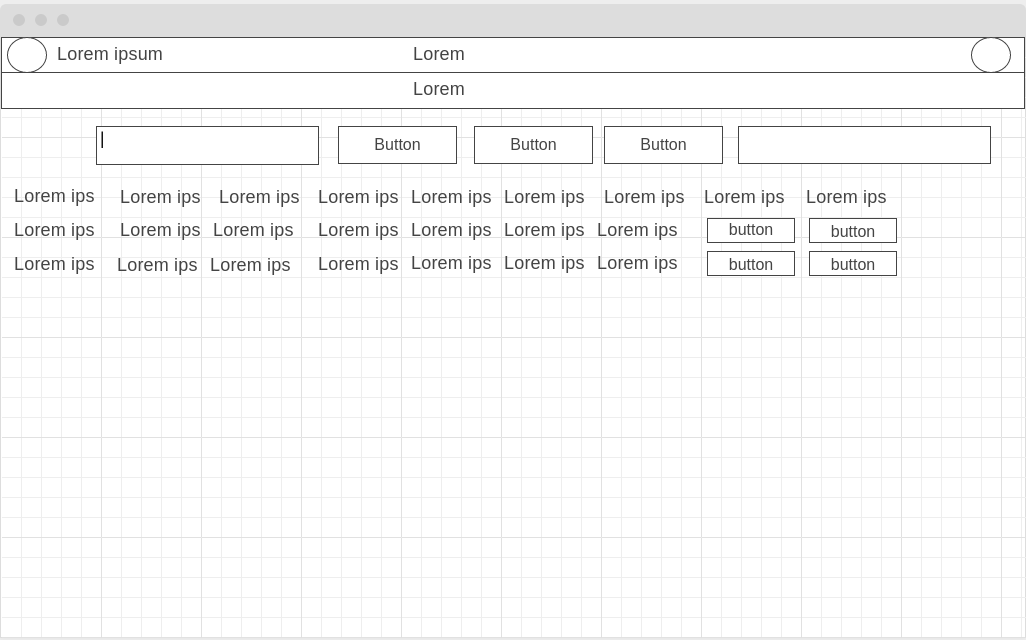
Raktárak, földek, járművek (egyforma a három)



Munkások



Aratási napló



Szállítmányok, aratási napló és terménymozgatás

3.1.2 Megvalósítás

#### 3.1.2.1 Weboldal

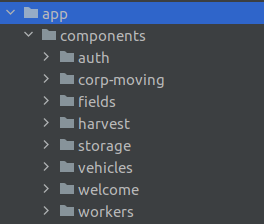
A weboldal megvalósításakor figyelembe vettem az elvárt oldalakat és ezek szerint bontottam le a weboldal forráskódját.

Az Angular-ben egy komponens három részből tevődik össze:

* HTML
* SCSS
* TS

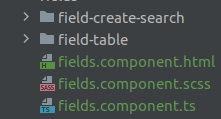
A “html” felelős az oldal alapvázáért, az “scss” adja meg a kinézetet, a “ts” fájlban pedig egy egyszerű háttér logikát írhatunk meg az oldalakhoz.

A különböző oldalakat megvalósító komponensek a “app/components” mappában találhatóak, ahol az oldalak nevei szerinti további almappákra vannak bontva.



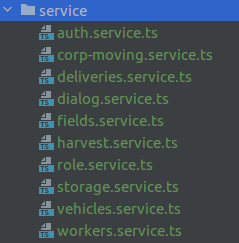
Komponensek mapparendszere

A földek, raktárak, járművek és munkások lebontása azonos rendszer szerint történt. Mindegyik oldalhoz tartozik egy főkomponens. Ez a főkomponens fogja adni az oldal alapvázát és felhasználni az oldalhoz készült további két komponenst.



Lebontási rendszer

Az “app/service” mappában található service-k fontos szerepet játszanak a weboldal és a backend kommunikációjában. Látható, hogy követik az oldalak neve szerinti lebontást.

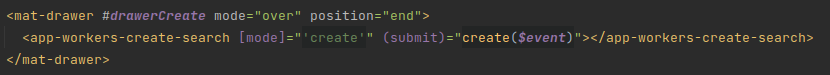


Service-ek

Ezekben a service-ekben találhatóak meg a backend-hez való hívások, ami alapján megkapja a weboldal az adatokat a megjelenítéshez, valamint innen kerülnek elküldésre a backendnek a módosítani kívánt adatok is.

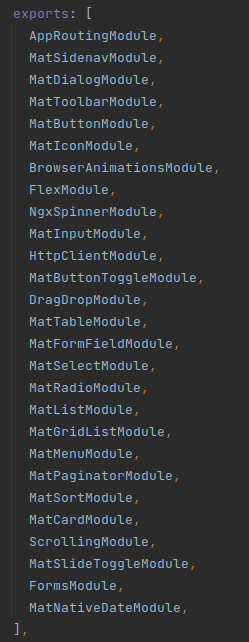
#### 3.1.1.2 Angular material

A weboldal megvalósításakor nagyon sok helyen felhasználtam az Angular material [[2]](#Bookmark5) által nyújtott előre elkészített komponenseket. Ezeket a komponenseket úgy ismerhetjük fel, hogy a “html” kódban a tag elején “mat” szó található.



Mat-drawer használata

A weboldalban használt Angular material komponensek listájáta az "app” mappában a “material.modul.ts” fájlban találhatjuk meg.



Material modulok listája

## 3.1.3 Tesztelés

A weboldal teszteléséhez manuális tesztelési technikákat alkalmaztam. Specifikáció alapú technikák közül használati eset teszteléssel ellenőriztem a felhasználói leírásban szereplő funkciókat. A 3.1.3.1- 3.1.3.8 fejezetekben részletezem a teszteseteket. A számokat tartalmazó beviteli mezők teszteléséhez az ekvivalencia particionálás és a határérték elemzés módszerét választottam.

Tapasztalat alapú technikák közüla hibasejtés és a felderítő tesztelés módszerét választottam.

#### 3.1.3.1 Bejelentkezés

|  |  |  |
| --- | --- | --- |
| Bejelentkezés 1. eset | GIVEN: | Bejelentkezési felület. |
| WHEN: | Helytelen adatokat adunk meg. |
| THEN: | Sikertelen bejelentkezés. |
| Bejelentkezés 2. eset | GIVEN: | Bejelentkezési felület. |
| WHEN: | Helyes adatokat adunk meg. |
| THEN: | Sikeres bejelentkezés. |

#### 3.1.3.2 Főoldal

|  |  |  |
| --- | --- | --- |
| Főoldal navigáció 1. eset | GIVEN: | Adott a főoldal. |
| WHEN: | Rákattintunk a raktárak gombra. |
| THEN: | Átváltunk a raktárak oldalra. |
| Főoldal navigáció 2. eset | GIVEN: | Adott a főoldal. |
| WHEN: | Rákattintunk a földek gombra. |
| THEN: | Átváltunk a földek oldalra. |
| Főoldal navigáció 3. eset | GIVEN: | Adott a főoldal. |
| WHEN: | Rákattintunk a munkások gombra. |
| THEN: | Átváltunk a munkások oldalra. |
| Főoldal navigáció 4. eset | GIVEN: | Adott a főoldal. |
| WHEN: | Rákattintunk a járművek gombra. |
| THEN: | Átváltunk a járművek oldalra. |
| Főoldal navigáció 5. eset | GIVEN: | Adott a főoldal. |
| WHEN: | Rákattintunk az aratási napló gombra. |
| THEN: | Átváltunk az aratási napló oldalra. |
| Főoldal navigáció 6. eset | GIVEN: | Adott a főoldal. |
| WHEN: | Rákattintunk a termény mozgatás gombra. |
| THEN: | Átváltunk a termény mozgatás oldalra. |

#### 3.1.3.3 Földek

|  |  |  |
| --- | --- | --- |
| Létrehozás 1. eset | GIVEN: | Adott a létrehozás ablak. |
| WHEN: | Nem teljesen töltjük ki az adott mezőket. |
| THEN: | Létrehozás nem lehetséges. |
| Létrehozás 2. eset | GIVEN: | Adott a létrehozás ablak. |
| WHEN: | Teljesen kitöltjük az adott mezőket. |
| THEN: | Létrejön az új raktár. |
| Frissítés | GIVEN: | Adott a földek táblázat. |
| WHEN: | Frissítjük a táblázat tartalmát. |
| THEN: | Megjelennek az aktuális földek a táblázatban. |
| Törlés | GIVEN: | Adott a földek táblázat. |
| WHEN: | Törlünk egy földet a táblázatból. |
| THEN: | A táblázatból eltűnik a törölt föld. |
| Keresés 1. eset | GIVEN: | Adott a földek táblázat. |
| WHEN: | Létező adatot írunk be. |
| THEN: | A táblázatban megjelennek a helyes adatok. |

|  |  |  |
| --- | --- | --- |
| Keresés 2. eset | GIVEN: | Adott a földek táblázat. |
| WHEN: | Nem létező adatot írunk be. |
| THEN: | A táblázat üres lesz. |

#### 3.1.3.4 Raktárak

|  |  |  |
| --- | --- | --- |
| Létrehozás 1. eset | GIVEN: | Adott a létrehozás ablak. |
| WHEN: | Nem teljesen töltjük ki az adott mezőket. |
| THEN: | Létrehozás nem lehetséges. |
| Létrehozás 2. eset | GIVEN: | Adott a létrehozás ablak. |
| WHEN: | Teljesen kitöltjük az adott mezőket. |
| THEN: | Létrejön az új raktár. |
| Frissítés | GIVEN: | Adott a raktárak táblázat. |
| WHEN: | Frissítjük a táblázat tartalmát. |
| THEN: | Megjelennek az aktuális raktárak a táblázatban. |
| Törlés | GIVEN: | Adott a raktárak táblázat. |
| WHEN: | Törlünk egy földet a táblázatból. |
| THEN: | A táblázatból eltűnik a törölt raktár. |

|  |  |  |
| --- | --- | --- |
| Keresés 1. eset | GIVEN: | Adott a raktárak táblázat. |
| WHEN: | Létező adatot írunk be. |
| THEN: | A táblázatban megjelennek a helyes adatok. |
| Keresés 2. eset | GIVEN: | Adott a raktárak táblázat. |
| WHEN: | Nem létező adatot írunk be. |
| THEN: | A táblázat üres lesz. |

#### 3.1.3.5 Munkások

|  |  |  |
| --- | --- | --- |
| Létrehozás 1. eset | GIVEN: | Adott a létrehozás ablak. |
| WHEN: | Nem teljesen töltjük ki az adott mezőket. |
| THEN: | Létrehozás nem lehetséges. |
| Létrehozás 2. eset | GIVEN: | Adott a létrehozás ablak. |
| WHEN: | Teljesen kitöltjük az adott mezőket. |
| THEN: | Létrejön az új munkás. |
| Frissítés | GIVEN: | Adott a munkások táblázat. |
| WHEN: | Frissítjük a táblázat tartalmát. |
| THEN: | Megjelennek az aktuális munkások a táblázatban. |

|  |  |  |
| --- | --- | --- |
| Törlés | GIVEN: | Adott a munkások táblázat. |
| WHEN: | Törlünk egy munkást a táblázatból. |
| THEN: | A táblázatból eltűnik a törölt munkás. |
| Keresés 1. eset | GIVEN: | Adott a munkások táblázat. |
| WHEN: | Létező adatot írunk be. |
| THEN: | A táblázatban megjelennek a helyes adatok. |
| Keresés 2. eset | GIVEN: | Adott a munkások táblázat. |
| WHEN: | Nem létező adatot írunk be. |
| THEN: | A táblázat üres lesz. |

#### 3.1.3.6 Járművek

|  |  |  |
| --- | --- | --- |
| Létrehozás 1. eset | GIVEN: | Adott a létrehozás ablak. |
| WHEN: | Nem teljesen töltjük ki az adott mezőket. |
| THEN: | Létrehozás nem lehetséges. |
| Létrehozás 2. eset | GIVEN: | Adott a létrehozás ablak. |
| WHEN: | Teljesen kitöltjük az adott mezőket. |
| THEN: | Létrejön az új jármű. |

|  |  |  |
| --- | --- | --- |
| Frissítés | GIVEN: | Adott a járművek táblázat. |
| WHEN: | Frissítjük a táblázat tartalmát. |
| THEN: | Megjelennek az aktuális járművek a táblázatban. |
| Törlés | GIVEN: | Adott a járművek táblázat. |
| WHEN: | Törlünk egy járművet a táblázatból. |
| THEN: | A táblázatból eltűnik a törölt jármű. |
| Keresés 1. eset | GIVEN: | Adott a járművek táblázat. |
| WHEN: | Létező adatot írunk be. |
| THEN: | A táblázatban megjelennek a helyes adatok. |
| Keresés 2. eset | GIVEN: | Adott a járművek táblázat. |
| WHEN: | Nem létező adatot írunk be. |
| THEN: | A táblázat üres lesz. |

#### 3.1.3.7 Aratási napló

|  |  |  |
| --- | --- | --- |
| Kiválasztás | GIVEN: | Adott az aratási naplók listája. |
| WHEN: | Rákatinttuk az egyik lista elemre. |
| THEN: | Az adott lista elem halvány szürke hátteret kap. |
| Törlés | GIVEN: | Adott egy kiválasztott lista elem. |
| WHEN: | Megnyomjuk a törlés gombot. |
| THEN: | A kiválasztott elem törlődik. |
| Megnyitás | GIVEN: | Adott egy kiválasztott lista elem. |
| WHEN: | Megnyomjuk a megnyitás gombot. |
| THEN: | Megjelenik egy táblázat, amiben azon aratási naplóhoz tartozó szállítmányok vannak. |
| Keresés | GIVEN: | Adott az aratási naplók listája. |
| WHEN: | Léteztő adatokat írunk be a keresési mezőbe. |
| THEN: | A listában megjelenek a kereset elemek |

|  |  |  |
| --- | --- | --- |
| Keresés | GIVEN: | Adott az aratási naplók listája. |
| WHEN: | Nem léteztő adatokat írunk be a keresési mezőbe. |
| THEN: | A listában öresen jelenik meg. |

|  |  |  |
| --- | --- | --- |
| Aratási naplóban szállítmány létrehozás 1. eset | GIVEN: | Adott az aratási naplóhoz tartozó szállítmányok táblázata. |
| WHEN: | Megnyomjuk az új fuvar hozzáadása gombot. |
| THEN: | Megjelenik egy felugró ablak. |
| Aratási naplóban szállítmány létrehozás 2. eset | GIVEN: | Adott a létrehozás 1.esete után megnyílt ablak |
| WHEN: | Kitoltjuk az adatokat majd megnyomjuk az oké gombot. |
| THEN: | Létrejön egy új fuvar. |

|  |  |  |
| --- | --- | --- |
| Keresés | GIVEN: | Adott az aratási naplóhoz tartozó szállítmányok táblázata. |
| WHEN: | Kitöltjük a keresési mezőt |
| THEN: | Megjelenek a táblázatban a kereset adatok. |
| Törlés | GIVEN: | Adott az aratási naplóhoz tartozó szállítmányok táblázata. |
| WHEN: | Megnyomjuk a törlés gombot |
| THEN: | Az adott szállítmány törlődik. |

#### 3.1.3.8 Termény mozgatás

|  |  |  |
| --- | --- | --- |
| Termény mozgatás szállítmány létrehozás 1. eset | GIVEN: | Adott a termény mozgatáshoz tartozó szállítmányok táblázata. |
| WHEN: | Megnyomjuk az új fuvar hozzáadása gombot. |
| THEN: | Megjelenik egy felugró ablak. |
| Termény mozgatás szállítmány létrehozás 2. eset | GIVEN: | Adott a létrehozás 1.esete után megnyílt ablak |
| WHEN: | Kitoltjuk az adatokat majd megnyomjuk az oké gombot. |
| THEN: | Létrejön egy új fuvar. |

|  |  |  |
| --- | --- | --- |
| Keresés | GIVEN: | Adott a termény mozgatáshoz tartozó szállítmányok táblázata. |
| WHEN: | Kitöltjük a keresési mezőt |
| THEN: | Megjelenek a táblázatban a kereset adatok. |
| Törlés | GIVEN: | Adott az termény mozgatáshoz tartozó szállítmányok táblázata. |
| WHEN: | Megnyomjuk a törlés gombot. |
| THEN: | Az adott szállítmány törlődik. |
| Frissítés 1.eset | GIVEN: | Adott a termény mozgatáshoz tartozó szállítmányok táblázata. |
| WHEN: | Megnyomjuk a frissítés gombot. |
| THEN: | Megjelenik egy ablak. |
| Frissítés 2.eset | GIVEN: | Adott a frissítés 1.esete után megnyílt ablak. |
| WHEN: | Módosítjuk az adatokat, majd megnyomjuk az ok gombot. |
| THEN: | Az adatok frissülnek. |

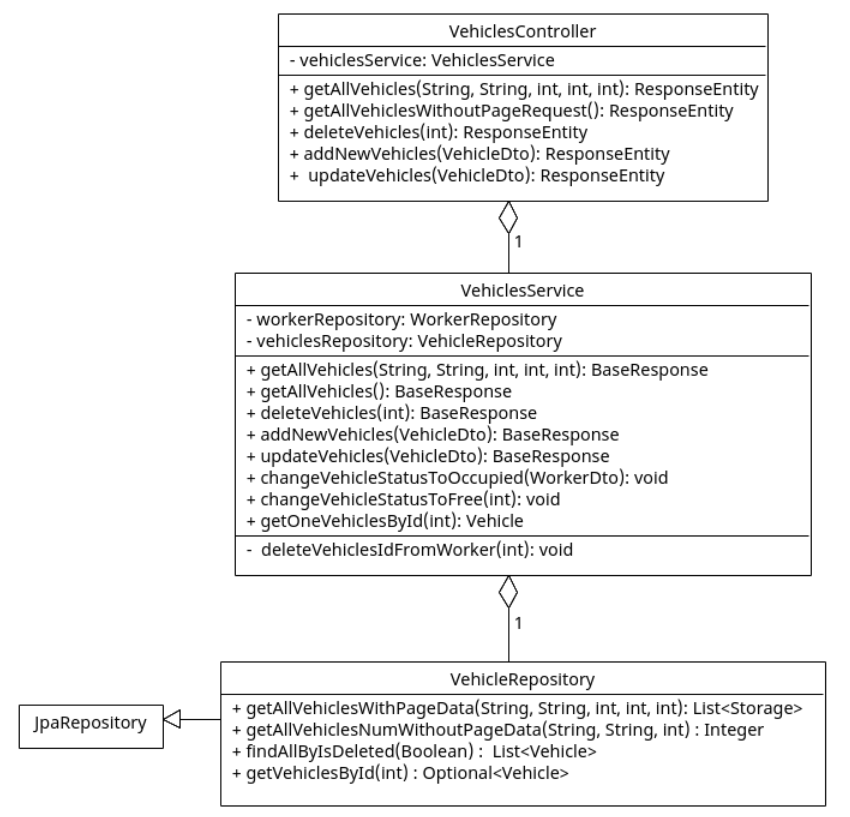
3.2 Backend

A backend nyelvének, illetve keretrendszerének kiválasztása egy webes alkalmazás fejlesztése során az egyik meghatározó pont. A fejlesztők általában a jól megszokott, eddigi projekteken megfelelt eszközöket próbálják használni az új technológiákkal szemben. Számtalan variáció készíthető el, de az adott fejlesztés számára legmegfelelőbb, legrugalmasabb megoldást kell választani.

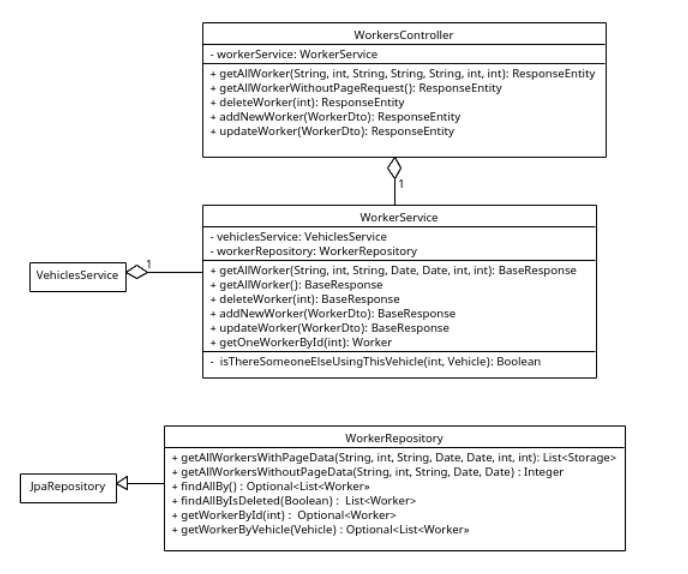
Választásom azért esett a Java programozási nyelvre, mivel szakmai gyakorlatom során és később a munkahelyemen is ezt használtam, így ez kézenfekvő megoldás volt.

### 3.2.1 Tervezés

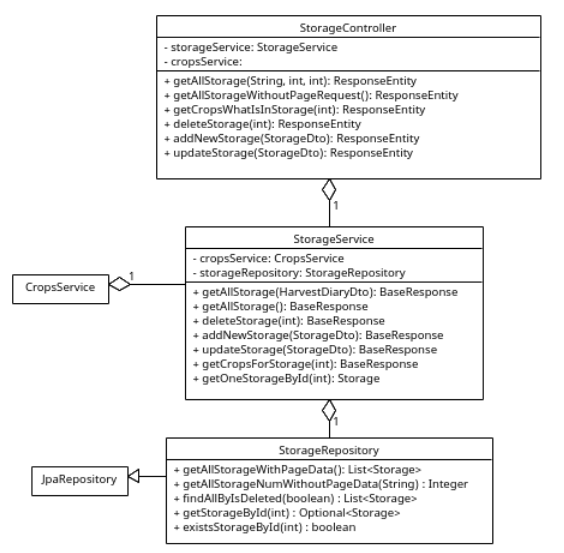
Ahogyan már említésre került a weboldal tervezésénél is az oldalak szerinti lebontás, így továbbra is magam előtt tartottam ezt a szemléletet és a backend tervezésénél is ehhez ragaszkodtam.



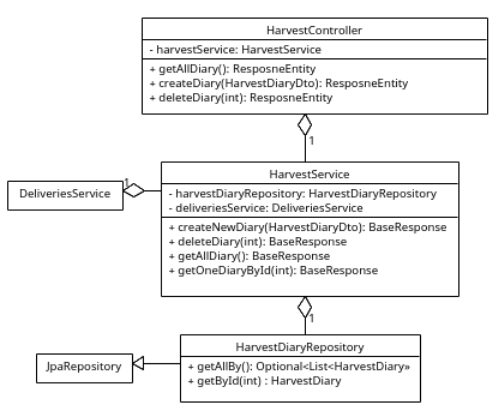
Vehciles controller, service és repository tervezet



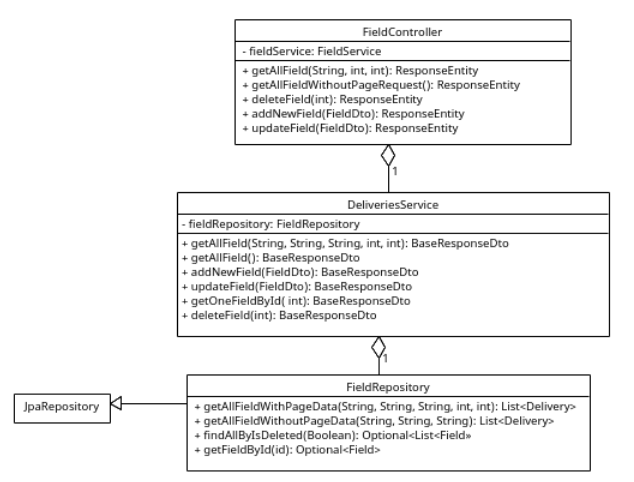
Workers controller, service és repository tervezet



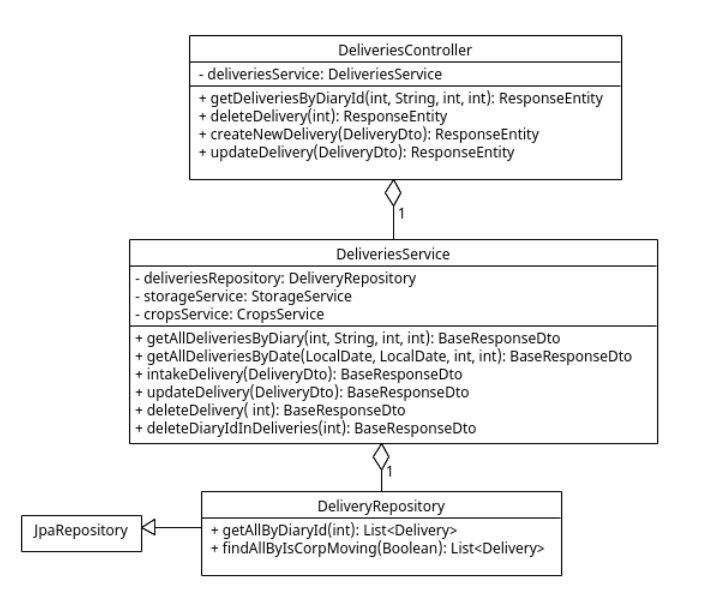
Storages controller, service és repository tervezet



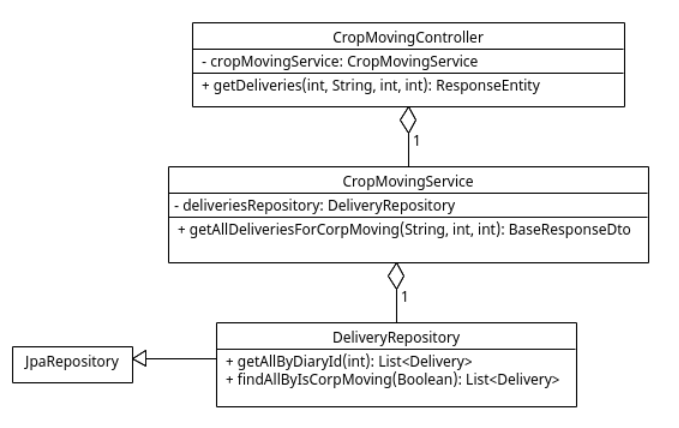
Harvest controller, service és repository tervezet



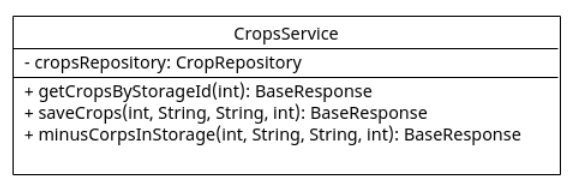
Field controller és repository tervezet



Deliveries controller, service és repository tervezet



Crop moving controller, service és repository tervezet



Crop service tervezet

### 3.2.2 Megvalósítás

#### 3.2.2.1 Kontrollerek

A backend egyik legfontosabb része a kontrollerek, mivel ezek tartalmazzák azokat a végpontokat, amiket a weboldal meg tud hívni. A kontrollerek nem végeznek üzleti logikát.

**StorageController**

A “StorageController” felelős a raktárakhoz kapcsolódó adatok visszaadásáért, módosításáért.

**Property:**

* StorageService : Típusa “StorageService” erről az osztályról később esik szó, röviden ez az osztály végzi a raktárakhoz tartozó üzleti logikát.

**Metódusok:**

* **getAllStorage** : Három paramétert vár el: ‘name’, ‘page’, ‘pageSize’. Ezek a paraméterek az adatokban való kereséshez és az oldalon való lapozáshoz szükségesek. Meghívja a “storageService”-nek a “getAllStorage” metódusát és a kapott paramétereket továbbadja.
* **getAllStorageWithoutPageRequest**: Nincs paramétere. Meghívja a “storageService”-nek a “getAllStorage” metódusát.
* **getCropsWhatIsInStorage** : Egy paramétert vár el: ‘storageId’. Meghívja a “storageService”-nek a “getCropsByStorageId” metódusát és a kapott paraméterét tovább adja.
* **deleteStorage**: Egy paramétert vár el: ‘storageId’. Meghívja a “storageService”-nek a “deleteStorage” metódusát és a kapott paraméterét továbbadja.
* **addNewStorage**: Egy paramétert vár el: ‘storage’ . Meghívja a “storageService”-nek a “addNewStorage” metódusát és a kapott paraméterét továbbadja.
* **updateStorage**: Egy paramétert vár el: ‘storage’. Meghívja a “storageService”-nek a “addNewStorage” metódusát és a kapott paraméterét továbbadja.

**WorkersController**

A “WorkersController” felelős a munkásokhoz kapcsolódó adatok vissza adásáért, módosításáért.

**Property:**

* workerService : Típusa “WorkerService” erről az osztályról később esik szó, röviden ez az osztály végzi a raktárakhoz tartozó üzleti logikát.

**Metódusok:**

* **getAllWorker**: Hét paramétert vár el: ‘name’, ‘vehicle’, ‘position’, ‘createdAtFrom’, ‘createdAtTo’, ‘page’, ‘pageSize’. Ezek a paraméterek az adatokban való kereséshez és az oldalon való lapozáshoz szükségesek. Meghívja a “workerService”-nek a “getAllWroker” metódusát és a kapott paramétereket tovább adja.
* **getAllWorkerWithoutPageRequest**: Nincs paramétere. Meghívja a “workerService”-nek a “getAllWroker” metódusát.
* **deleteWorker**: Egy paramétert vár el: workerId. Meghívja a “workerService”-nek a “deleteWorker” metódusát és a kapott paraméterét tovább adja.
* **addNewWorker**: Egy paramétert vár el: ‘worker’. Meghívja a “workerService”-nek a “addNew Worker” metódusát és a kapott paraméterét tovább adja.
* **updateWorker**: Egy paramétert vár el: ‘worker’. Meghívja a “workerService”-nek a “addNew Worker” metódusát és a kapott paraméterét tovább adja.

**VehiclesController**

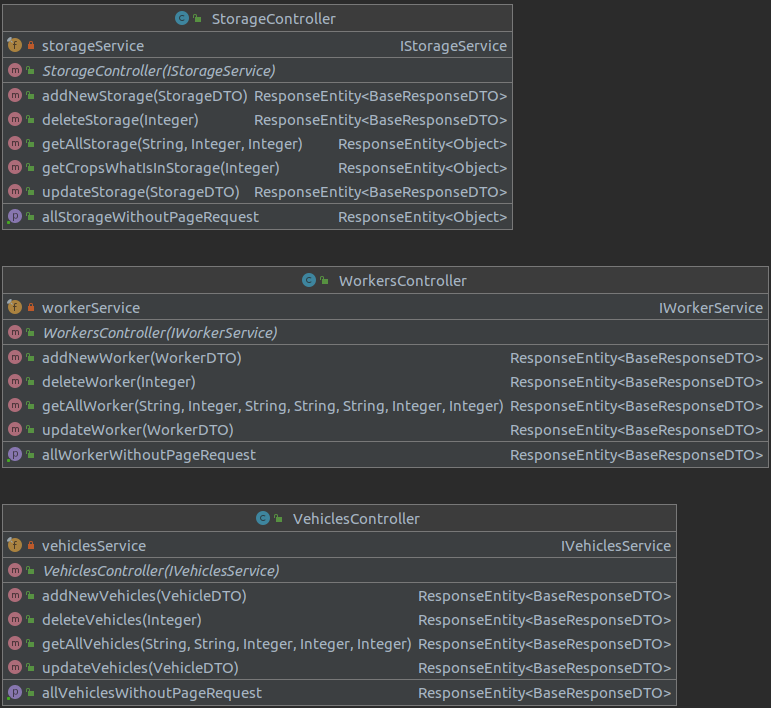
A “VehiclesController” felelős a járművekhez kapcsolódó adatok vissza adásáért, módosításáért.

**Property:**

* **vehiclesService** : Típusa “VehiclesService” erről az osztályról később esik szó, röviden ez az osztály végzi a raktárakhoz tartozó biznisz logikát.

**Metódusok:**

* **getAllVehicles** : Öt paramétert vár el: ‘name’, ‘type’, ’status’, ‘page’, ‘pageSize’. Ezek a paraméterek az adatokban való kereséshez és az oldalon való lapozáshoz szükségesek. Meghívja a “vehiclesService”-nek a “getAllVehicles” metódusát és a kapott paramétereket tovább adja.
* **getAllVehicles WithoutPageRequest**: Nincs paramétere. Meghívja a “vehiclesService”-nek a “getAllVehicles” metódusát.
* **deleteVehicles**: Egy paramétert vár el: ‘vehiclesId’. Meghívja a “vehiclesService”-nek a “deleteVehicles” metódusát és a kapott paraméterét tovább adja.
* **addNewVehicles**: Egy paramétert vár el: ‘vehicle’. Meghívja a “vehiclesService”-nek a “addNewVehicles” metódusát és a kapott paraméterét tovább adja.
* **updateVehicles**: Egy paramétert vár el: ‘vehicle’. Meghívja a “vehiclesService”-nek a “addNewVehicles” metódusát és a kapott paraméterét tovább adja.



Storage, Workers, Vehicles kontrollerek

**FieldController**

A “FieldController” felelős a földekhez kapcsolódó adatok vissza adásáért, módosításáért.

**Property:**

* **fieldService**: Típusa “FieldService” erről az osztályról később esik szó, röviden ez az osztály végzi a raktárakhoz tartozó biznisz logikát.

**Metódusok:**

* **getAllVehicles**: Hat paramétert vár el: ‘name’, ‘corpType’, ’corpName’, ‘page’, ‘pageSize’. Ezek a paraméterek az adatokban való kereséshez és az oldalon való lapozáshoz szükségesek. Meghívja a “fieldService”-nek a “getAllVehicles” metódusát és a kapott paramétereket tovább adja.
* **getAllVehicles WithoutPageRequest**: Nincs paramétere. Meghívja a “fieldService”-nek a “getAllField” metódusát.
* **deleteVehicles**: Egy paramétert vár el: ‘vehiclesId’. Meghívja a “fieldService”-nek a “deleteField” metódusát és a kapott paraméterét tovább adja.
* **addNewVehicles**: Egy paramétert vár el: ‘vehicle’. Meghívja a “fieldService”-nek a “addNewField” metódusát és a kapott paraméterét tovább adja.
* **updateVehicles**: Egy paramétert vár el: ‘vehicle’. Meghívja a “fieldService”-nek a “addNewField” metódusát és a kapott paraméterét tovább adja.

**DeliveriesController**

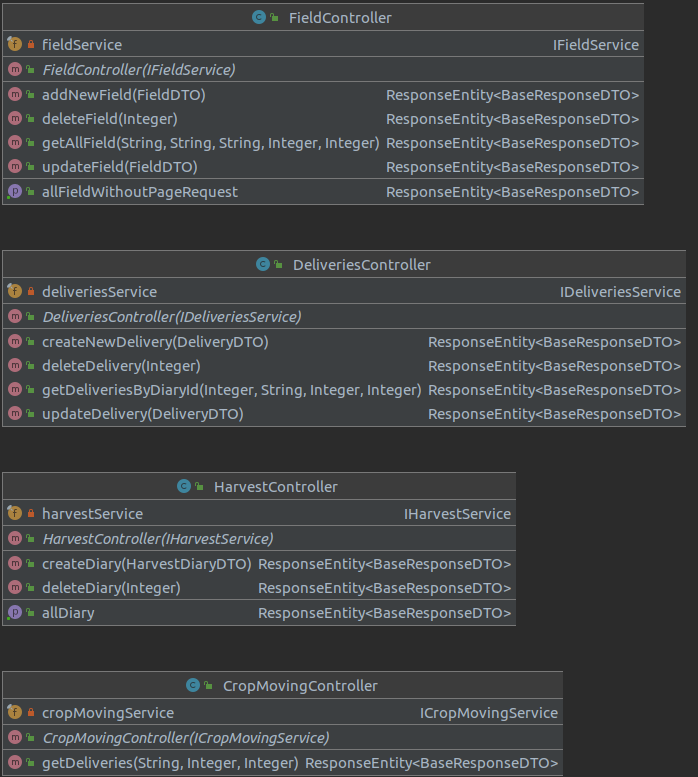
A “DeliveriesController” felelős a szállítmányokhoz kapcsolódó adatok vissza adásáért, módosításáért

**Property:**

* **deliveriesService**: Típusa “DeliveriesService” erről az osztályról később esik szó, röviden ez az osztály végzi a raktárakhoz tartozó biznisz logikát.

**Metódusok:**

* **getDeliveriesByDiaryId**: Három paramétert vár el: ‘search’, ‘page’, ‘pageSize’. Ezeken felül van egy útvonal változója is ‘diaryId’. Ezek a paraméterek az adatokban való kereséshez és az oldalon való lapozáshoz szükségesek. Meghívja a “deliveriesService”-nek a “getAllVehicles” metódusát és a kapott paramétereket tovább adja.
* **deleteVehicles**: Egyparamétert vár el: ‘deliveryId’. Meghívja a “deliveriesService”-nek a “deleteDelivery” metódusát és a kapott paraméterét tovább adja.
* **addNewDelivery**: Egyparamétert vár el: ‘deliveriesDTO’. Meghívja a “deliveriesService”-nek a “intakeDelivery” metódusát és a kapott paraméterét tovább adja.
* **updateDelivery**: Egyparamétert vár el: ‘deliveriesDTO’. Meghívja a “deliveriesService”-nek a “addNewDelivery” metódusát és a kapott paraméterét tovább adja.



Field, Deliveries, Harvest, CropMoving kontrollerek

**HarvestController**

A “HarvestController” felelős az aratási naplókhoz kapcsolódó adatok vissza adásáért, módosításáért

**Property:**

* harvestService: Típusa “HarvestService” erről az osztályról később esik szó, röviden ez az osztály végzi a raktárakhoz tartozó üzleti logikát.

**Metódusok:**

* **getAllDiary**: Meghívja a “harvestService”-nek a “deleteDelivery” metódusát és a kapott paraméterét tovább adja.
* **createDiary**: Egy paramétert vár el: ‘diaryDTO’. Meghívja a “harvestService”-nek a “deleteDiary” metódusát és a kapott paraméterét tovább adja.
* **deleteDiary**: Egy paramétert vár el: ‘diaryId’. Meghívja a “harvestService”-nek a “deleteDiary” metódusát és a kapott paraméterét tovább adja.

**CropMovingController**

A “CropMovingController” felelős a termény mozgatásban részt vevő szállítmányok adatinak vissza adásáért, módosításáért

**Property:**

* cropMovingService: Típusa “CropMovingService” erről az osztályról később esik szó, röviden ez az osztály végzi a raktárakhoz tartozó biznisz logikát.

**Metódusok:**

* **getDeliveries**: Három paramétert vár el: ‘search’, ‘page’, ‘pageSize’. Ezek a paraméterek az adatokban való kereséshez és az oldalon való lapozáshoz szükségesek. Meghívja a “CropMovingService”-nek a “getAllDeliveriesForCorpMoving” metódusát és a kapott paramétereket tovább adja.

#### 3.2.2.1 Szervizek

**StorageService**

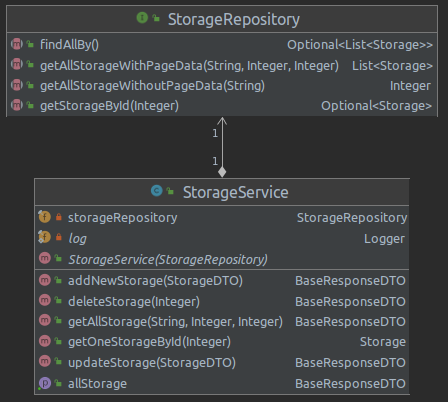
Feladata, hogy elvégezze a raktárakhoz tartozó üzleti logikát.

**Property:**

* **storageRepositry:** Típusa StorageRepository. Ezen property segítségével végezhetünk el az adatbázison hívásokat.
* **Log:** Ezen property segítségével tudunk az alkalmazás futása sorrén loggolni.

**Methódusok:**

* **getAllStorage:** Három paramétert vár el: ‘name’, ‘page’, ‘pageSize’. Elkéri az adatbázisban található raktárakat, amiket a ‘name’ paraméter szerint szűr.
* **getAllStorage:** Nulla paramétert vár el, ez az előző metódusnak egy olyan verziója, ami nem szűri le az adatokat csak visszaadja az összes elemet az adatbázisból, amit talál.
* **getOneStorageById:** Egy paramétere van ‘id’. Megkeresi az adatbázisban azt a raktárat, aminek az id-je a paraméterrel megegyezik és visszaadja.
* **deleteStorage:** Egy paramétere van ‘id’. Megkeresi az adatbázisban azt a raktárat, aminek az id-je a paraméterrel megegyezik és törli.
* **updateStorage**: Paraméterben elvár egy raktárosztályt, ami szerint a létező raktár adatait módosítani tudja az adatbázisban.
* **CreateStorage:** A paraméterben megkapott raktárat létrehozza az adatbázisban.



StorageService

**WrokerService**

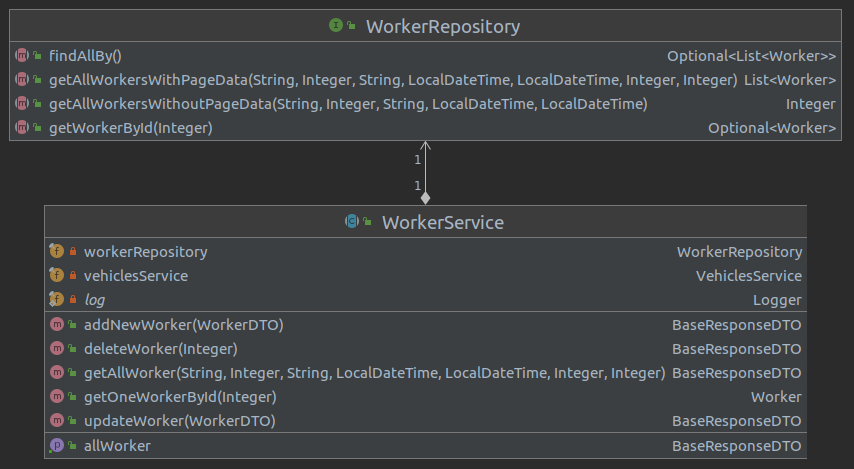
Feladata, hogy elvégezze a munkásokhoz tartozó biznisz logikát.

**Property:**

* **workerRepositry:** Típusa WorkerRepository. Ezen property segítségével végezhetünk el az adatbázison hívásokat.
* **vehcileService:** Feladata a járművekhez tartozó üzleti logikák elvégzése.
* **log:** Ezen property segítségével tudunk az alkalmazás futása sorrén loggolni.

**Methódusok:**

* **getAllWorker:** Hét paramétert vár el: ‘name’, ‘page’, ‘pageSize’. Elkéri az adatbázisban található raktárakat, amiket a ‘name’, ‘vehicle’, ‘position’, ‘createdAtFrom’, ‘createdAtTo’ paraméterek szerint szűr.
* **getAllWorker:** Nulla paramétert vár el, ez az előző metódusnak egy olyan verziója, ami nem szűri le az adatokat csak visszaadja az összes elemet az adatbázisból, amit talál.
* **getOneWorkerById:** Egy paramétere van az ‘id’. Megkeresi az adatbázisban azt a munkást, aminek az id-je a paraméterrel megegyezik és visszaadja.
* **deleteWorker:** Egy paramétere van az ‘id’. Megkeresi az adatbázisban azt a munkást, aminek az id-je a paraméterrel megegyezik és törli.
* **addNewWorker:** Paraméterben elvár egy munkásosztályt, ami szerint a létező munkás adatait módosítani tudja az adatbázisban.
* **updateWorker:** A paraméterben megkapott munkást létrehozza az adatbázisban.



WorkerService

**VehiclesService**

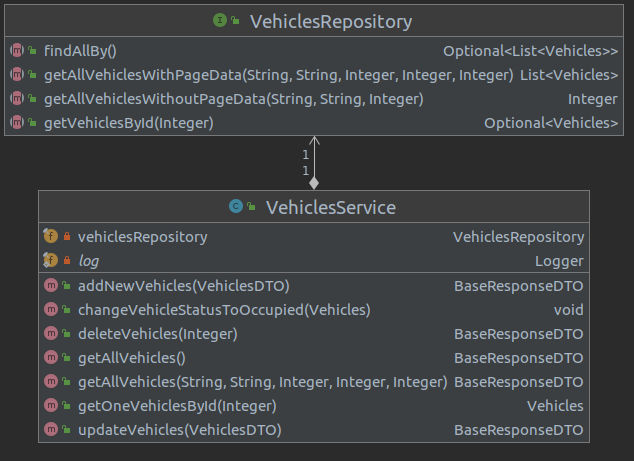
Feladata, hogy elvégezze a járművekhez tartozó üzleti logikát.

**Property:**

* **vehiclesRepositry:** Típusa VehicleRepository. Ezen property segítségével végezhetünk el az adatbázison hívásokat.
* **log:** az alkalmazás futása során a naplózásért felelős.

**Methódusok:**

* **getAllVehicles:** Öt paramétert vár el: ‘name’, ‘type’, ’status’, ‘page’, ‘pageSize’. Elkéri az adatbázisban található raktárakat, amiket a ‘name’, ‘type’, ’status’ paraméterek szerint szűr.
* **getAllVehicles:** Nulla paramétert vár el, ez az előző metódusnak egy olyan verziója, ami nem szűri le az adatokat csak visszaadja az összes elemet az adatbázisból, amit talál.
* **getOneVehiclesById:** Egy paramétere van az ‘id’. Megkeresi az adatbázisban azt a járművet, aminek az id-je a paraméterrel megegyezik és visszaadja.
* **deleteVehicles:** Egy paramétere van az ‘id’. Megkeresi az adatbázisban azt a munkást, aminek az id-je a paraméterrel megegyezik és törli.
* **addNewVehicles:** Paraméterben elvár egy jármű osztályt, ami szerint a létező munkás adatait módosítani tudja az adatbázisban.
* **updateVehicles:** A paraméterben megkapott járművet létrehozza az adatbázisban.



VehiclesService

**FieldService**

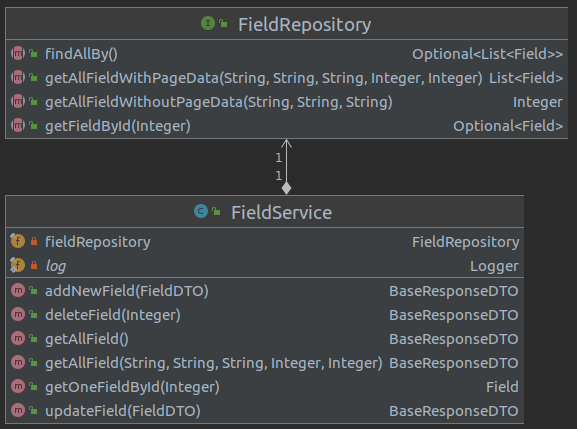
Feladata, hogy elvégezze a földekhez tartozó üzleti logikát.

**Property:**

* **fieldRepositry:** Típusa FieldRepository. Ezen property segítségével végezhetünk el az adatbázison hívásokat.
* **log:** Ezen property segítségével tudunk az alkalmazás futása során logolni.

**Methódusok:**

* **getAllField:** Hat paramétert vár el: ‘name’, ‘corpType’, ’corpName’, ‘page’, ‘pageSize’. Elkéri az adatbázisban található raktárakat, amiket a ‘name’, ‘type’, ’status’ paraméterek szerint szűr.
* **getAllField:** Nulla paramétert vár el, ez az előző metódusnak egy olyan verziója, ami nem szűri le az adatokat csak visszaadja az összes elemet az adatbázisból, amit talál.
* **getOneFieldById:** Egy paramétere van az ‘id’. Megkeresi az adatbázisban azt a földet, aminek az id-je a paraméterrel megegyezik és visszaadja.
* **deleteField:** Egy paramétere van az ‘id’. Megkeresi az adatbázisban azt a földet, aminek az id-je a paraméterrel megegyezik és törli.
* **addNewField:** Paraméterben elvár egy jármű osztályt, ami szerint a létező föld adatait módosítani tudja az adatbázisban.
* **updateField:** A paraméterben megkapott földet létrehozza az adatbázisban.



FieldService

**CropMovingService**

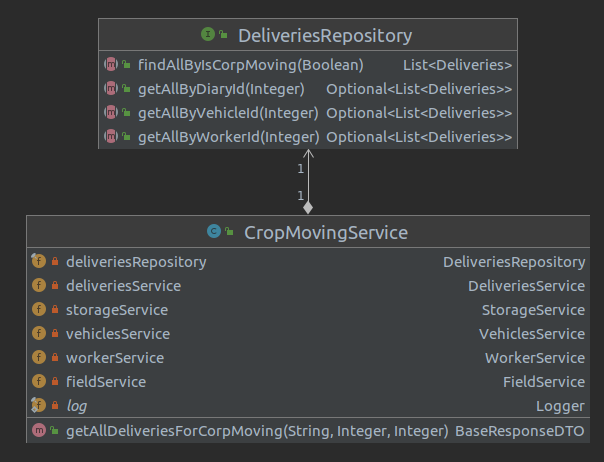
Feladata, hogy elvégezze a termény mozgatásban részt vevő szállítmányokhoz tartozó biznisz logikát.

**Property:**

* **DeliveriesRepositry:** Típusa DeliveriesRepository. Ezen property segítségével végezhetünk el az adatbázison hívásokat.
* **log:** Ezen property segítségével tudunk az alkalmazás futása során logolni.

**Methódusok:**

* **getAllDeliveriesForCorpMoving:** Háromparamétert vár el: ’search’, ‘page’, ‘pageSize’. Elkéri az adatbázisban található raktárakat, amiket a ‘name’ paraméter szerint szűr.



CropMovingService

**DeliveriesService**

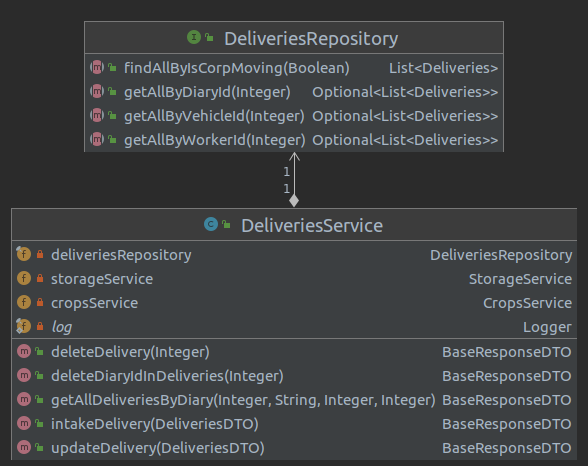
Feladata, hogy elvégezze a szállítmányokhoz tartozó üzleti logikát.

**Property:**

* **deliveriesRepositry:** Típusa DeliveriesRepository. Ezen property segítségével végezhetünk el az adatbázison hívásokat.
* **storageService:** Feladata, hogy elvégezze a raktárakhoz tartozó üzleti logikát.
* **cropsService:** Feladata, hogy elvégezze a növényekhez tartozó üzleti logikát.
* **log:** Ezen property segítségével tudunk az alkalmazás futása során logolni.

**Methódusok:**

* **getAllDeliveriesByDiary**: Négy paramétert vár el: ‘diaryId’, ‘search’, ‘page’, ‘pageSize’. Kikéri azon szállítmanyok adatait az adatbázisból, amelyikek telejsítik a keresési feltételt, amit a ‘search’ paraméterben kapunk és a ‘diaryId’ paraméterben megadott aratási naplóhoz tartoznak.
* **intakeDelivery**: Paraméterben egy szállítmány adatait kapja meg, amit lement az adatbázisaban. Ezen felül a “storageService” segítségével az a raktár adatokat is frissíti.
* **updateDelivery**: Paraméterben egy szállítmány adatait kapja meg, ami szerint frissíti a már létező szállítmányt. Ezen felül a “storageService” segítségével az a raktár adatokat is frissíti.
* **deleteDelivery**: Paraméterben egy szállítmány id-át kapja meg, amit töröl az adatbázisból. Ezen felül a “storageService” segítségével az a raktár adatokat is frissíti.
* **deleteDiaryIdInDeliveries**: Egy paramétere van a ‘diaryId’. Feladata, hogy a paraméterben kapott aratási naplóhoz tartozó szállítmányoknak a referenciáját erre az adot naplóra null-ra módosítsa.



DeliveriesService

**CropsService**

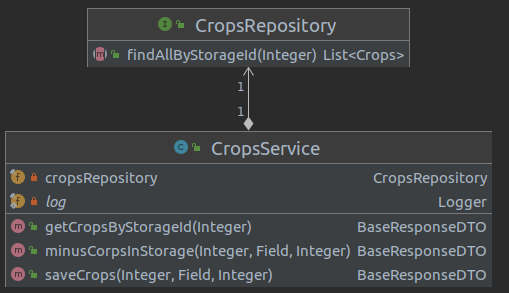
Feladata, hogy elvégezze a növényekhez tartozó üzleti logikát.

**Property:**

* **deliveriesRepositry:** Típusa DeliveriesRepository. Ezen property segítségével végezhetünk el az adatbázison hívásokat.
* **log:** Ezen property segítségével tudunk az alkalmazás futása során logolni.

**Methódusok:**

* **getCropsByStorageId**: Egy paramétert vár el ez pedig a ‘storageId’. Feladata a metódusnak, hogy ezen raktárban található növényeket visszaadja.
* **saveCrops**: Négy paramétere van ‘storageId’, ‘field’, ‘amount’. Metódus lemenmt egy új növény mennyiséget a raktárhoz, ha még nincs, ilyen növény a raktárban egyébként akkor már a létező adatokat módosítja.
* **minusCorpsInStorage**: Négy paramétere van ‘storageId’, ‘field’, ‘amount’. A metódus feladata, hogy megkeresse a paraméterben kapott növényt a raktárban majd a mennyiségéből vonja ki azt az értéket, amit a paraméterben kapott. Ha az értéke nulla vagy az alá esik, akkor törölje az adatbázisból.

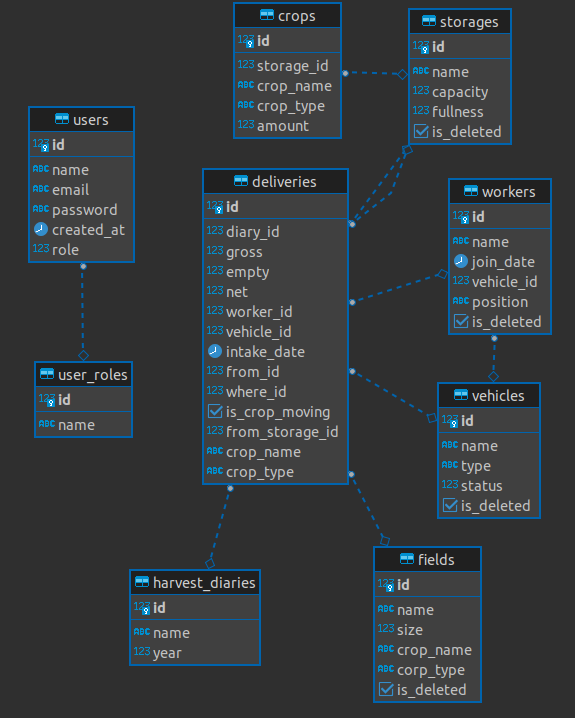


CropsService

## 3.3 Adatbázis

Az adatbázisom PostgreSQL [[4]](#Bookmark3) alapú adatbázis. Ez az adatbázis összesen kilenc darab táblát tartalmaz. Név szerint a következőket:

1. workers tábla: a dolgozók adatait tartalmazza (id, név, csatlakozás ideje, jármű id, pozíció, törölve van-e)
2. vehicles tábla: a járművek adatait tartalmazza (id, név, típus, státusz, törölt-e)
3. crops tábla: a növények adatait tartalmazza (id, raktár id, növény neve, növény típusa, mennyiség)
4. storage tábla: a raktárak adatait tartalmazza (id, név, kapacitás, telítettség)
5. field tábla: a földek adatait tartalmazza (id, név, méret, növény típusa, növény neve)
6. harvest\_diary tábla: az aratási naplók adatait tartalmazza (id, név, év)
7. users tábla: a felhasználók adatait tartalmazza (id, név, email, jelszó, betöltött tisztség, létrehozás dátuma)
8. user\_roles tábla: a felhasználók tisztségeit tartalmazza (id, név)
9. deliveries tábla: a szállítmányok adatait tartalmazó tábla (id, napló id, a szállítmány súlya bruttóban, az üres súlya, a szállítmány nettó súlya, munkás id, jármű id, beszállítási dátum, honnan szállítják (id), hová szállítják (id), növényt szállítanak-e, küldő raktár id)



Adatbázis schema

## 3.4 Szoftver futtatás

Az szoftver megírása során alkalmaztam egy projekt menedzsert név szerint a Maven-t [[6]](#Bookmark1). Szerencsére kompatibilis Angular alapú projektekkel is, így ez rengeteget tud segíteni projekt kitelepítésében és futatásában.

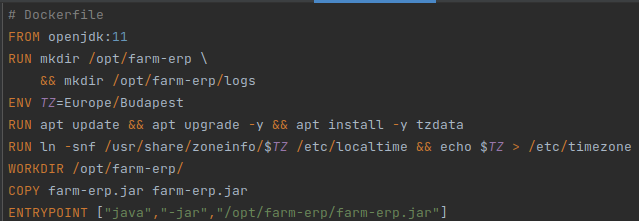
Maven egy olyan projekt menedzser rendszer, amit Java-ban írtak és ebből kifolyólag ebben a nyelvben íródott projektekhez a legoptimálisabb. Segítségünkre van a projekt futtatásában, felépítésében és függőségeinek kezelésében.

Maven segítségével egyszerűen ‘build’-elhetjük a projektet a ‘mvn clean install’ parancs segítségével, aminek következtében meg kapjuk az alkalmazást tartalmazó ‘farm-erp.jar’ fájlt.

Az alkalmazás kitelepítésének egyik legegyszerűbb formája lehet, ha egy olyan szervere rakjuk ki ahol található Docker [[7]](#Bookmark2).

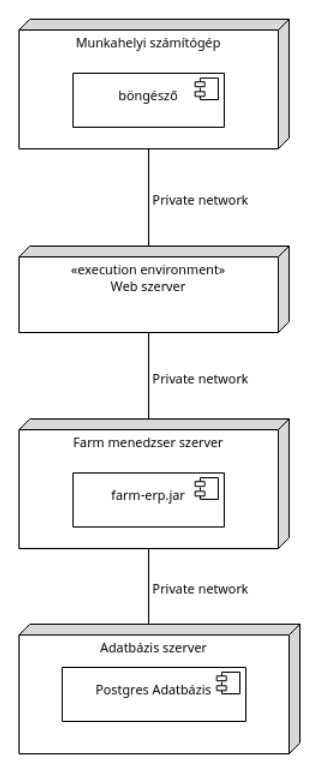
Docker segítségével, úgy nevezett konténereket építhetünk fel, amik egy külön álló rendszert képeznek a szerveren. Manapság egyes alkalmazások kitelepítésének egyik legelterjedtebb formája a Dockerizáció, mivel nagyon könnyedén pár parancs segítségével építhetünk fel egy konténert, ami alkalmas az alkalmazásunk futtatására.

Egy konténer felépítéséhez szükségünk van egy ‘image’-re, amihez szükségünk van egy ‘Dockerfile’-ra. Ebben megadhatjuk az alapbeállításokat. Ezen fájl segítségével másolhatjuk be a szükséges futtatandó fájlokat és állíthatunk be egy indító parancsot, ami elindítja az alkalmazást.



Az alkalmazáshoz tartozó Dockerfile

Ahozz, hogy a Dockerfile-ból felépítsük az image-ünket a ‘docker build .’ parancsra van szükségünk. Ezekután a ‘docker run --name farm-erp -p 80:8080 –d farm-erp’ parancs használatával az elkészült image-ből konténert építünk.



Kitelepítési diagramm

# 4 Összegzés

A szakdolgozatom elkészítése során a legnagyobb nehézségek közé sorolnám magát a 3 különböző rész megtervezését, illetve megvalósítását. Ezekhez segítségül tudtam hívni az egyetemi webes alapjaimat, valamint a java-s egyetemi tapasztalataimat, kiegészítve a munkában elsajátított tapasztalatokkal. Emellett a weboldal egységes design-jának megtervezése is kihívást jelentett, de ebben segített az Angular Material.

Az oldalt több szempontból is tovább lehet még fejleszteni. Például meg lehetne valósítani azt, hogy egy földön több fajta növény is lehessen, vagy a kapacitás ellenőrzését a raktáraknál, illetve ezeken kívül még a szállítmányoknál jelezni, hogy pontosan milyen növényt is szállítanak.

# 5 Irodalomjegyzék

[1] - Angular keretrendszer. [Online] [Hivatkozva 2021.12.10] [angular.io/guide/forms-overview](https://angular.io/guide/forms-overview)

[2] - Angular material keretrendszer. [Online] [Hivatkozva 2021.12.10] [material.angular.io](https://material.angular.io/)

[3] - Spring boot keretrendszer. [Online] [Hivatkozva 2021.12.10] [spring.io/projects/spring-boot](https://spring.io/)

[4] - PostgreSQL adatbázis. [Online] [Hivatkozva 2021.12.10] [postgresql.org](https://www.postgresql.org/)

[5] - Adatbázisrendszerek - Alapvetés - Második, átdolgozott kiadás. [Könyv] [Kiadás 2009] [Szerző Ullman, Jeffrey D. - Widom, Jennifer]

[6] - Maven projekt menedzser [Online] [Hivatkozva 2022.11.14] [maven.apache.org](https://maven.apache.org/)

[7] - Docker [Online] [Hivatkozva 2022.11.14] [docker.com](https://www.docker.com/)