

DogGo – Budapest kutyás térképe

BUDAPESTI MŰSZAKI SZAKKÉPZÉSI CENTRUM  
PETRIK LAJOS KÉT TANÍTÁSI NYELVŰ TECHNIKUM

SZOFTVERFEJLESZTŐ ÉS TESZTELŐ TECHNIKUS SZAKMA

A szakképesítés azonosító száma: 5-0613-12-03

# DogGo

Budapest kutyás térképe

**Készítette:** Takács Balázs Levente

**Csapattagok:** Kilián Marcell András, Korcsmáros Kristóf György

**Konzulens:** Budaházi Bence

Budapest, 2021.04.19.

## Tartalomjegyzék

1 DogGo.....	1
2 Tartalomjegyzék .....	2
3 Plágium nyilatkozat .....	3
4 Bevezetés .....	4
4.1 Téma.....	5
4.2 Témaválasztás indoklása .....	6
5 Fejlesztői dokumentáció .....	7
5.1 Funkciók.....	7
5.2 Használati eset diagram.....	8
5.3 Backend.....	9
5.3.1 Adatbázis terv .....	9
5.3.2 Adattáblák (doggodb).....	10
5.3.3 Backend telepítésének lépései .....	12
5.3.4 API végpontok.....	13
5.4 DogGo Admin Interface – Asztali alkalmazás.....	25
5.4.1 Alkalmazott fejlesztői eszközök.....	25
5.4.2 Specifikáció .....	26
5.4.3 Felületterv.....	28
5.4.4 Implementáció .....	33
5.4.5 Továbbfejlesztési lehetőségek .....	41
6 Tesztelési dokumentáció.....	42
7 Felhasználói dokumentáció.....	45
7.1 Hardver és szoftver igény.....	45
7.2 Telepítés .....	45
8 Összegzés.....	52
8.1 Project Summary .....	52
9 Felhasznált források.....	53
10 Ábrajegyzék.....	54

## Plágium nyilatkozat

Alulírott Takács Balázs Levente felelősségem tudatában kijelentem, hogy a zárodolgozat saját szellemi tevékenységem eredménye, az abban foglaltak más személyek jogszabályban rögzített jogait nem sértik.

A zárodolgozat egy részét Kilián Marcell Andrással és Korcsmáros Kristóf Györggyel közösen készítettük el. Ezeket a részeket pontosan jelöltük.

Budapest, 2021.04.19.

.....  
Kilián Marcell  
András

.....  
Korcsmáros Kristóf  
György

.....  
Takács Balázs  
Levente

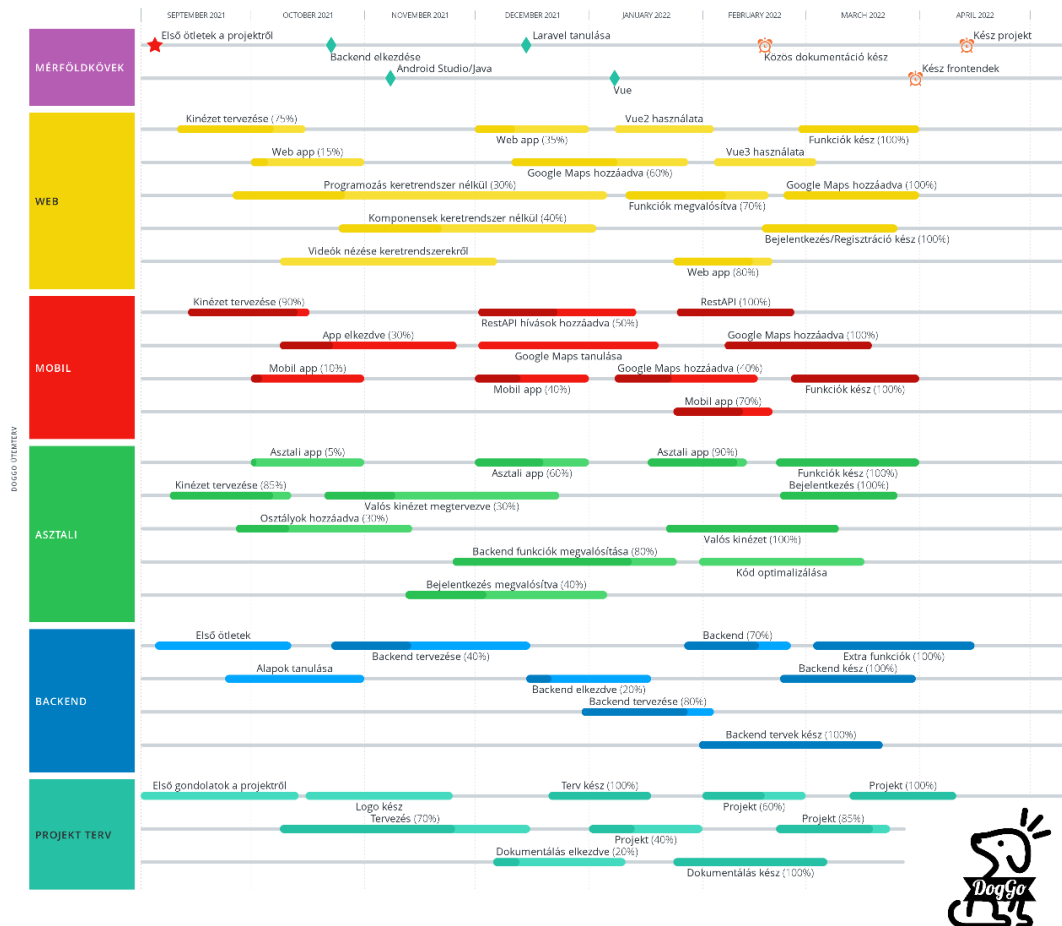
## 4 Bevezetés

Különböző forrásokból hallani, ismerősöktől, közösségi médiából azt, hogy egy adott helyre elvihetik kedvenceiket. Ennek ellenére mégsem tudunk sok véleményt meghallgatni, olvasni az adott helyekről. Ez az alkalmazás ennek a problémának a megoldására kínál lehetőséget.

Egy olyan alkalmazást készítünk, ahova az emberek leírhatják véleményüket a közelükben lévő, számukra jelentéssel bíró helyekről. Ha más ember is hallaná a pozitív tapasztalatokat az adott helyről, tudná, hogy milyen kutyás emberek járnak milyen helyekre, lehet, hogy ő is elmenne és kipróbálni a mások által megjelölt helyeket.

A másik fő célunk ezzel az alkalmazással az, hogy kutyás csoportokat, társaságokat hozzunk létre. A helyek értékelésével, leírásával az emberek kapcsolatba léphetnek egymással, motiválhatják egymást, hogy kimozduljanak otthonról kiskedvenceikkel.

Ezek a tulajdonságok, amelyek egyedivé teszik ezt a programot. Nincs másik program egyelőre, ami akár csak egy délutáni kutyasétáltatást összehozna más emberekkel.



1. ábra DogGo ütemterv

## 4.1 Téma

Téma a kutyasétáltatás. Mivel a választott témának egyedinek kell lennie, ezért olyan problémákat kerestünk a mindennapi alkalmazásokban, amikre egy alternatív program sincs. A kutyákat manapság nem lehet akárhová vinni, külön engedéllyel, bizonyos időközönként vagy egyáltalán nem lehet számos helyre menni velük. Emellett a közösségi médiában is sokszor látni olyat, amikor az emberek segítséget kérnek, hogy mégis hová vihetik a kutyáikat. Ezért lenne jó, egy olyan alkalmazás, ahol minden kutyasétáltatással kapcsolatos információ megtalálható egy adott helyhez, parkhoz.

Mit tud a szoftver? Egy térképen láthatunk megjelölt helyeket, amiket emberek tudnak az alkalmazáshoz rendelni, regisztrálás, bejelentkezés után. Az alkalmazást lehet látogatóként is használni, ebben az esetben, csak megnézhetjük, hogy milyen helyeket, milyen értékelésekkel jelöltek meg az emberek, alkalmas kutyasétáltatásra, azonban nem jelölhetnek meg helyeket.

A helymegjelöléshez tartoznak a képek, egy 5-ös skálán értékelések és a kommentek. Ha egy helyet alkalmasnak tartunk kutyasétáltatásra, akkor megjelölhetjük azt a pozíciót a térképen, hozzárendelhetünk képeket, megoszthatjuk, hogy mennyire tetszett az a hely egy 5-ös skálán és szöveges értékelést (kommentet) csatolhatunk a megjelölt pozícióhoz.

Rosszakaró emberek mindig is léteztek, mindenhol megjelennek. Az asztali alkalmazás azért jött létre, hogy adminisztrátor jogosultsággal szűrni lehessen ezeket a felhasználókat. Ha valaki rengeteg rossz, elfogadhatatlan helyet jelölne meg, mint például egy autópálya közepe, ahol nem feltétlenül biztonságos házikedvencünket sétáltatni, az adminisztrátor jogosultsággal tiltani tudjuk a felhasználókat.

Abban az esetben, ha egy felhasználó hibát észlel, továbbítani tudja a fejlesztőknek. A visszajelzés anonim, nem kell regisztrálni, bejelentkezni ennek a funkciónak a használatához. Az alkalmazás későbbi fejlesztése miatt adjuk hozzá ezt a funkciót az alkalmazáshoz.

## 4.2 Témaválasztás indoklása

Manapság a háztartások nagy részében hatalmas szerepet játszik a háziállat, azonban kiskedvenceink a legtöbb helyről ki vannak tiltva. Szerettünk volna olyan témát választani, amivel megtudjuk könnyíteni, azoknak az embereknek az életét, akik minden-  
hová a kutyájukkal mennének. Gondolkoztunk, hogy hogyan is lehetne ezt a problémát egy alkalmazás segítségével orvosolni, így kezdődött el a DogGo.

Számos esetleges megoldás jutott az eszünkbe, az egyik ilyen megoldás az lenne, hogy mi felsorolunk elterjedt, számunkra izgalmas, jó, kellemes helyeket, jelezve a felhasználóknak, hogy hova mehetnek a kiskedvencükkel sétálni. Azonban ezek a helyek három ember által kedvelt helyek lennének, attól, hogy egy helyet kedvel három ember, nem feltétlenül jelenti azt, hogy az emberek többsége is kedvelné. Ezért ezt a megoldást elvetettük. Rájöttünk, hogy minél több ember mutat számára kutyasétáltatásra alkalmas helyet, annál több ember fog, számára megfelelő helyet találni a sétáltatásra, ahova ő is elmenne kiskedvencét sétáltatni.

Emiatt jutottunk arra a megoldásra, hogy a felhasználók véleményét kell megjelenítenünk egy felületen, mivel minél több ember osztja meg a véleményét, annál több ember talál számára alkalmas helyet az alkalmazás segítségével. Minél nagyobb a választék a helyek közül annál több felhasználó használhatja az alkalmazást. Mivel a helyeket felhasználók adják hozzá, ők is valószínűleg megjelennek a számukra kellemes helyeken. Ezáltal motiválhatjuk a felhasználókat a kimozdulásra. Nagyobb eséllyel megy el sétálni egy ember, ha tudja, hogy beszélgethet, találkozhat másokkal. Ezeken felül, ha sikerülne kialakítani, az alkalmazás segítségével egy állatbarát közösséget, akkor egymást is ösztönöznék egy tartalmas sétára, találkozásra.

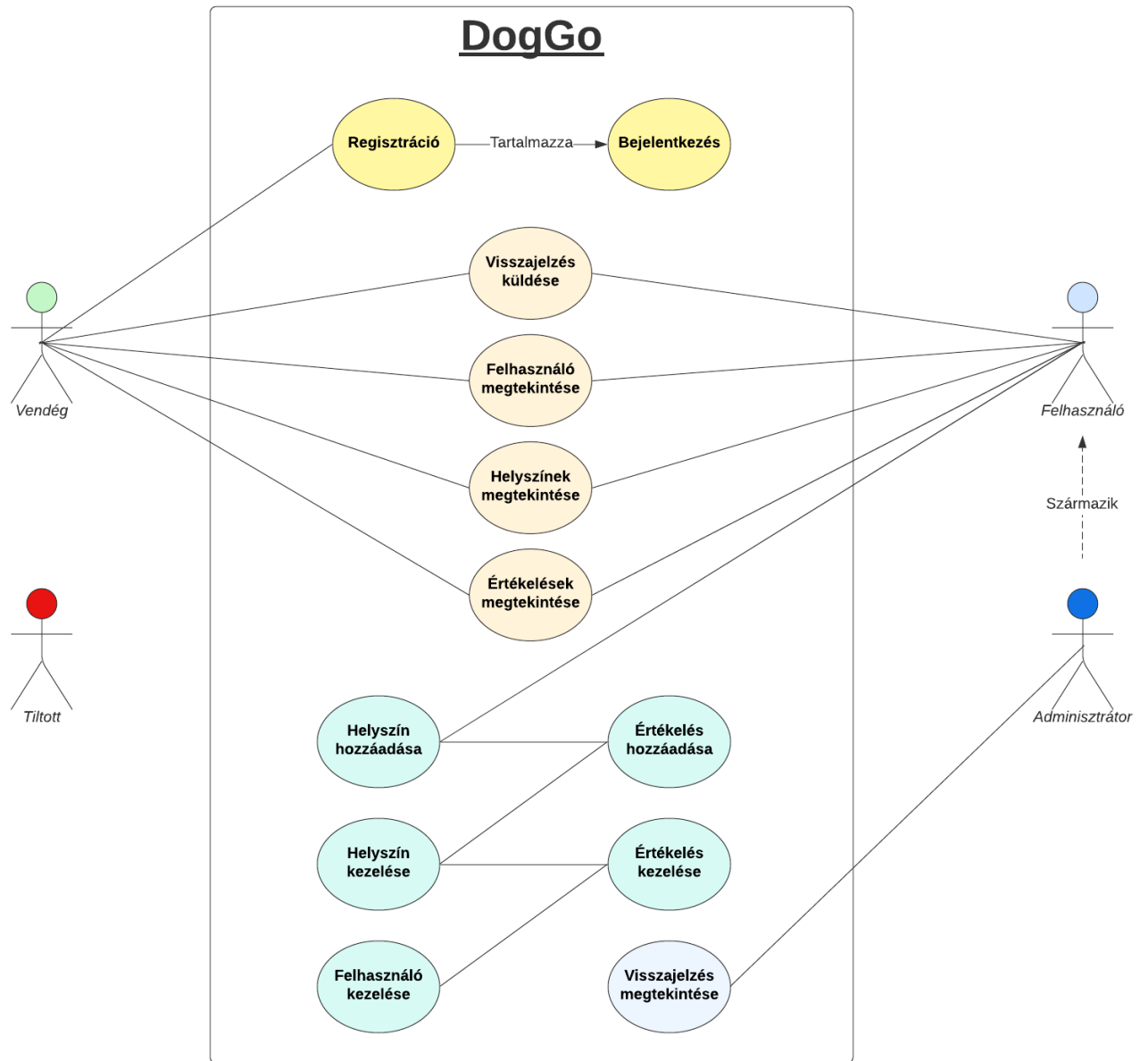
Ez az ötlet a fejlesztői csoport minden tagjának tetszett, a lelkesedést és egyetértést látva a projekt témája meg is született.

## 5 Fejlesztői dokumentáció

### 5.1 Funkciók

	Látogató	Regisztrált felhasználó	Adminisztrátor
Regisztráció	✓	✗	✗
Bejelentkezés	✗	✓	✓
Visszajelzés küldése	✓	✓	✓
Hely megtekintése	✓	✓	✓
Értékelések megtekintése	✓	✓	✓
Hely hozzáadása	✗	✓	✓
Értékelés hozzáadása	✗	✓	✓
Saját hely kezelése	✗	✓	✓
Saját értékelés kezelése	✗	✓	✓
Összes hely kezelése	✗	✗	✓
Összes értékelés kezelése	✗	✗	✓
Felhasználók kezelése	✗	✗	✓
Visszajelzések kezelése	✗	✗	✓

## 5.2 Használati eset diagram



2. ábra: Használati eset diagram

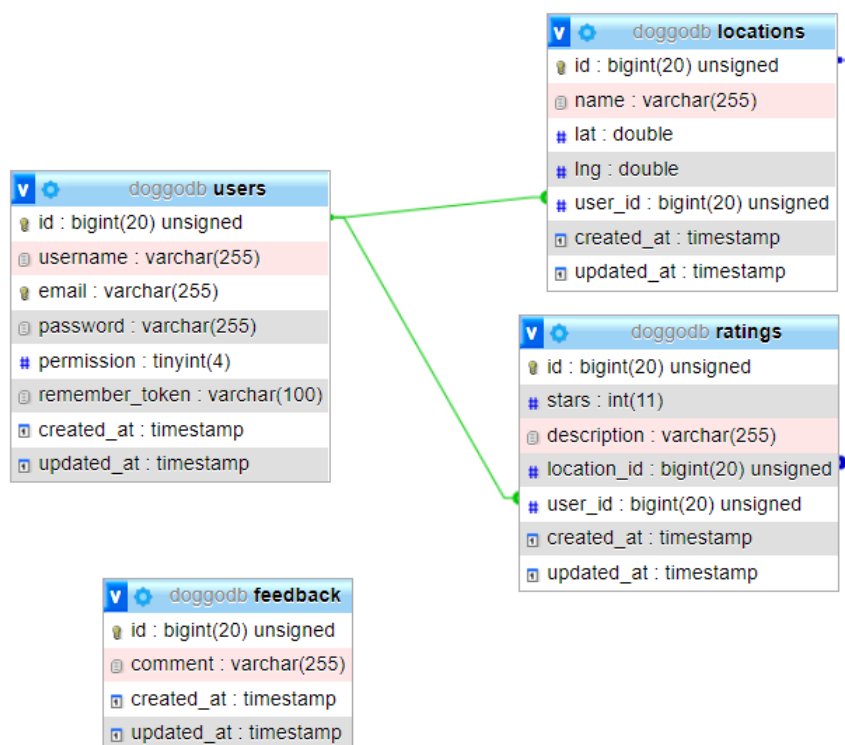


## 5.3 Backend

### 5.3.1 Adatbázis terv

Az adatbázis az egyik legfontosabb rész a projekt működéséhez. Itt tároljuk a **felhasználók**, **helyek**, **értékelések** adatait és a **vi sszajelzéseket**, amikkel fejleszteni, javítani tudjuk az alkalmazásunkat.

A projekt backend részét közösen csináljuk. Megbeszélünk egy időpontot és online folytatjuk a munkát. A táblák létrehozásával kezdtük a backend megvalósítását. Az adatbázis terv miatt egyszerű volt a táblák létrehozása. **Laravel** keretrendszert használtunk az elképzelt adatbázis megvalósításához. Néhány utasítással könnyen lehet a táblákat feltölteni *teszt adatokkal*, így egy sokkal *átláthatóbb* kezdetleges végeredményt látunk munkánk során.



3. ábra: Adatbázis terv

### 5.3.2 Adattáblák (doggodb)

users tábla

Oszlopnév	Típus	Hossz	Tartalma
<b>id</b>	BIGINT, PK, AI	20	Elsődleges kulcs
<b>username</b>	VARCHAR	5-20	Felhasználó felhasználóneve
<b>email</b>	VARCHAR	255	Felhasználó E-mail címe
<b>password</b>	VARCHAR	8-	Felhasználó jelszava
<b>permission</b>	TINYINT	4	Milyen jogosultsággal rendelkezik a felhasználó: 0 – default 1 – tiltva 2 – admin 3 – super admin
<b>created_at</b>	TIMESTAMP		Létrehozás dátuma

locations tábla

Oszlopnév	Típus	Hossz	Tartalom
<b>id</b>	BIGINT, PK, AI	20	Eldősleges kulcs
<b>name</b>	VARCHAR	5-40	Hely neve
<b>description</b>	VARCHAR	255	Hely leírása
<b>lat</b>	DOUBLE		Hely koordinátája (szélesség)
<b>lng</b>	DOUBLE		Hely koordinátája (hosszúság)
<b>user_id</b>	BIGINT, FK		Hivatkozás a users táblára

ratings tábla

Oszlopnév	Típus	Hossz	Tartalom
<b>id</b>	BIGINT, PK, AI	20	Elsődleges kulcs
<b>stars</b>	INT	1-5	Számos értékelés
<b>description</b>	VARCHAR	255	Szöveges értékelés
<b>location_id</b>	BIGINT, FK	20	Hivatkozás a locations táblára
<b>user_id</b>	BIGINT, FK	20	Hivatkozás a users táblára

feedback tábla

Oszlopnév	Típus	Hossz	Tartalom
<b>id</b>	BIGINT, PK, AI	20	Elsődleges kulcs
<b>comment</b>	VARCHAR	255	Visszajelzés szövege
<b>created_at</b>	TIMESTAMP		Létrehozás dátuma

### 5.3.3 *Backend telepítésének lépései*

Ahhoz, hogy elérjük az adatbázist, szükségünk lesz egy adatbázis kiszolgálóra. Ehhez legegyszerűbben használható a **XAMPP** telepítő csomag, melyet az alábbi linken tudunk letölteni:

<https://www.apachefriends.org/hu/download.html>

Telepítés után indítsuk el a **MySQL** kiszolgálót, ezután nyissuk meg a **phpmyadmin**-t és hozzunk létre egy adatbázist „doggodb” néven *utf8mb4\_hungarian\_ci* karakterkódo-lással.

A backend projekt megnyitása után, készítsünk egy másolatot a *.env.example* fájlról és nevezzük át *.env*-re. A fájlban írjuk át megfelelőre az adatbázis kapcsolat adatait.

Hajtsuk végre a következő utasításokat a konzolban:

- composer install
- php artisan key:generate --ansi
- php artisan migrate
- php db:seed
- Indítsuk el a fejlesztői szerveret:
- php artisan serve

Teszteljük **Thunder Client** vagy **Postman** segítségével, hogy az alábbi URL megfelelő **JSON** adatot ad-e vissza:

<http://127.0.0.1:8000/api/users>

### 5.3.4 *API végpontok*

A program **backend** része úgy lett megvalósítva, hogy az adatot **JSON** formátumban adja és várja. Az adatbázis **factory**-k és **seeder**-ek segítségével, **tesztadatokkal** töltöttük fel.

Az megadott adatoknak meg kell felelnie az adatbázisban megadott feltételeknek (*adatbázis feltételek: 9-10. oldal*), a **JSON** példákban emiatt nincs *feltüntetve*, hogy mik a határértékek.

#### **users végpont csoport**

##### **GET /api/users**

Visszaadja a felhasználók adatait.

Például: GET <http://localhost:8000/api/users>

```
[
  {
    "id": 1,
    "username": "Dr. Quinten VonRueden",
    "email": "hlind@example.com",
    "permission": 1
  },
  {
    "id": 2,
    "username": "Rhea Schowalter",
    "email": "russel.hugh@example.org",
    "permission": 1
  }
]
```

##### **GET /api/users/{id}**

Visszaadja az adott ID-val rendelkező felhasználó adatait.

Például: GET <http://localhost:8000/api/users/5>

```
{
  "id": 5,
  "username": "Destinee Tillman",
  "email": "hagenes.irving@example.org",
  "permission": 1
}
```

## POST /api/users

Létrehoz egy új felhasználót a megadott adatokkal. Az ID-n kívül minden adat megadása kötelező. A jelszót **titkosítva** tároljuk el az adatbázisban, **bcrypt** titkosítást használunk, ez a laravel keretrendszer alapértelmezett titkosítási módszere.

Egy titkosított jelszó:

*\$2y\$10\$pGp/1o8W8qedKt5ChlRuX.pdl3WbwNAuKcicl41St0fIFX2aSGvC*

Sikeres hozzáadás esetén a létrehozott felhasználó adatai beleértve a generált ID-t visszaadja.

Például: POST <http://localhost:8000/api/users>

Bemenet:

```
{
  "username": "test",
  "email": "test@example.net",
  "password": "test",
  "permission": 1
}
```

Eredmény:

```
{
  "username": "test",
  "email": "test@example.net",
  "permission": 1,
  "id": 16
}
```

### **PUT /api/users/{id}**

Módosítja az adott ID-val rendelkező felhasználó adatait. Csak a módosítani kívánt adatokat kell megadni. Hogyha csak a felhasználónevet szeretnénk módosítani elég azt megadni:

Sikeres módosítás után visszaadja a módosított felhasználó adatait. Az ID nem módosítható.

Például: PUT <http://127.0.0.1:8000/api/users/5>

Bemenet:

```
{  
  "username": "test"  
}
```

Eredmény:

```
{  
  "id": 5,  
  "username": "test",  
  "email": "hagenes.irving@example.org",  
  "permission": 1  
}
```

### **DELETE /api/users/{id}**

Kitörli az adatbázisból az adott ID-val rendelkező felhasználót.

Például: DELETE <http://localhost:8000/api/users/1>

Eredmény:

Status: 204 No Content

### **Hibakezelés**

Hibás végpont esetén, vagy, ha az adatok nem felelnek meg a követelményeknek, a backend jelzi ezt.

A hibának megfelelő HTTP státuszkódot adja vissza (400-zal kezdődő), és a visszakapott JSON "message" tulajdonsága tartalmazza a hiba okát.

Például: GET <http://localhost:8000/api/users/9999> (nem létező id)

Status: 404 Not Found

```
{  
  "message": "A megadott azonosítóval nem található felhasználó"  
}
```

## **locations végpont csoport**

### **GET /api/locations**

Visszaadja a helyszínek adatait.

Például: GET <http://localhost:8000/api/locations>

```
[
  {
    "id": 1,
    "name": "Yasmine Oval",
    "lat": 44.240438,
    "lng": 84.739212,
    "user_id": 1
  },
  {
    "id": 2,
    "name": "Nicolette Trace",
    "lat": 14.781673,
    "lng": -24.293254,
    "user_id": 1
  }
]
```

### **GET /api/locations/{id}**

Visszaadja az adott ID-val rendelkező helyszín adatait.

Például: GET <http://localhost:8000/api/locations/5>

```
{
  "id": 5,
  "name": "Elmore Turnpike",
  "lat": -17.084211,
  "lng": -171.558352,
  "user_id": 1
}
```



## **POST /api/locations**

Létrehoz egy új helyszínt a megadott adatokkal. Az ID-n kívül minden adat megadása kötelező.

Sikeres hozzáadás esetén a létrehozott helyszín adatai beleértve a generált ID-t visszaadja.

Például: POST <http://127.0.0.1:8000/api/locations>

Bemenet:

```
{
  "name": "test",
  "lat": 44.240438,
  "lng": 84.739212,
  "user_id": 1
}
```

Eredmény:

```
{
  "id": 5,
  "name": "Elmore Turnpike",
  "lat": -17.084211,
  "lng": -171.558352,
  "user_id": 1
}
```

## **PUT /api/locations/{id}**

Módosítja az adott ID-val rendelkező helyszín adatait. Csak a módosítani kívánt adatokat kell megadni. Hogyha csak a nevet szeretnénk módosítani elég azt megadni:

Sikeres módosítás után visszaadja a módosított helyszín adatait. Az ID nem módosítható.

Például: PUT <http://127.0.0.1:8000/api/locations/5>

Bemenet:

```
{
  "name": "test"
}
```

Eredmény:

```
{
  "id": 5,
  "name": "test",
  "lat": -17.084211,
  "lng": -171.558352,
  "user_id": 1
}
```

### **DELETE /api/locations/{id}**

Kitörli az adatbázisból az adott ID-val rendelkező helyszínt.

Például: DELETE <http://localhost:8000/api/locations/1>

Eredmény:

Status: 204 No Content

### **Hibakezelés**

Hibás végpont esetén, vagy, ha az adatok nem felelnek meg a követelményeknek, a backend jelzi ezt.

A hibának megfelelő HTTP státuszкодot adja vissza (400-zal kezdődő), és a visszakapott JSON "message" tulajdonsága tartalmazza a hiba okát.

Például: GET <http://localhost:8000/api/locations/9999> (nem létező id)

Status: 404 Not Found

```
{  
  "message": "A megadott azonosítóval nem található helyszín."  
}
```

## **ratings végpont csoport**

### **GET /api/ratings**

Visszaadja az értékelések adatait.

Például: GET <http://localhost:8000/api/ratings>

```
[
  {
    "id": 1,
    "stars": 1,
    "description": "Quia similique corporis ratione placeat sed sequi.",
    "location_id": 1,
    "user_id": 1
  },
  {
    "id": 2,
    "stars": 3,
    "description": "Id quo facere tempore iste aliquid dolor.",
    "location_id": 1,
    "user_id": 1
  }
]
```

### **GET /api/ratings/{id}**

Visszaadja az adott ID-val rendelkező értékelés adatait.

Például: GET <http://localhost:8000/api/ratings/5>

```
{
  "id": 5,
  "stars": 3,
  "description": "Id perspiciatis consequatur dignissimos tempora.",
  "location_id": 1,
  "user_id": 1
}
```

## **POST /api/ratings**

Létrehoz egy új értékelést a megadott adatokkal. Az ID-n kívül minden adat megadása kötelező.

Sikeres hozzáadás esetén a létrehozott értékelés adatai beleértve a generált ID-t visszaadja.

Például: POST <http://127.0.0.1:8000/api/ratings>

Bemenet:

```
{
  "stars": 3,
  "description": "test",
  "location_id": 1,
  "user_id": 1
}
```

Eredmény:

```
{
  "stars": 3,
  "description": "test",
  "location_id": 1,
  "user_id": 1,
  "id": 16
}
```

## **PUT /api/ratings/{id}**

Módosítja az adott ID-val rendelkező értékelés adatait. Csak a módosítani kívánt adatokat kell megadni. Hogyha csak a szöveges értékelést szeretnénk módosítani elég azt megadni:

Sikeres módosítás után visszaadja a módosított értékelés adatait. Az ID nem módosítható.

Például: PUT <http://127.0.0.1:8000/api/ratings/5>

Bemenet:

```
{
  "description": "test"
}
```

Eredmény:

```
{
  "id": 5,
  "stars": 3,
  "description": "test",
  "location_id": 1,
  "user_id": 1
}
```

### **DELETE /api/ratings/{id}**

Kitörli az adatbázisból az adott ID-val rendelkező értékelést.

Például: DELETE <http://localhost:8000/api/ratings/1>

Eredmény:

Status: 204 No Content

### **Hibakezelés**

Hibás végpont esetén, vagy, ha az adatok nem felelnek meg a követelményeknek, a backend jelzi ezt.

A hibának megfelelő HTTP státuszkódot adja vissza (400-zal kezdődő), és a visszakapott JSON "message" tulajdonsága tartalmazza a hiba okát.

Például: GET <http://localhost:8000/api/ratings/9999> (nem létező id)

Status: 404 Not Found

```
{  
  "message": "A megadott azonosítóval nem található értékelés."  
}
```

### **GET /api/best\_rating**

Visszaadja a legjobb értékeléssel rendelkező helyszín átlag értékelését és nevét.

Például: GET [http://localhost:8000/api/best\\_rating](http://localhost:8000/api/best_rating)

```
{
  "name": "Sedrick Land",
  "atlag": "5.0"
}
```

### **GET /api/worst\_rating**

Visszaadja a legrosszabb értékeléssel rendelkező helyszín átlag értékelését és nevét.

Például: GET [http://localhost:8000/api/worst\\_rating](http://localhost:8000/api/worst_rating)

```
{
  "name": "Mossie Common",
  "atlag": "1.7"
}
```

### **GET /api/rating\_by\_user/{id}**

Visszaadja az adott ID-val rendelkező felhasználó értékeléseit.

Például: GET [http://localhost:8000/api/rating\\_by\\_user/1](http://localhost:8000/api/rating_by_user/1)

```
[
  {
    "id": 1,
    "stars": 2,
    "description": null,
    "location_id": 12,
    "user_id": 1
  },
  {
    "id": 2,
    "stars": 1,
    "description": "Doloremque quo iure assumenda sint quae blanditiis qui.",
    "location_id": 10,
    "user_id": 1
  }
]
```

### **GET /api/locations\_allowed**

Visszaadja az engedélyezett helyszínek adatait.

Például: GET [http://127.0.0.1:8000/api/locations\\_allowed](http://127.0.0.1:8000/api/locations_allowed)

```
[
  {
    "id": 2,
    "name": "Chadd Courts",
    "description": "As she said to Alice; and Alice was not much.",
    "lat": 5.633529,
    "lng": -161.676676,
    "allowed": true,
    "user_id": 1
  }
]
```

### **GET /api/locations\_not\_allowed**

Visszaadja a még nem engedélyezett helyszínek adatait.

Például: GET [http://127.0.0.1:8000/api/locations\\_not\\_allowed](http://127.0.0.1:8000/api/locations_not_allowed)

```
[
  {
    "id": 1,
    "name": "Lockman Plains",
    "description": "Alice. 'And where HAVE my shoulders got to.",
    "lat": 68.222227,
    "lng": -67.375361,
    "allowed": false,
    "user_id": 1
  }
]
```

### **GET /api/feedbacks\_read**

Visszaadja a már olvasott visszajelzések adatait.

Például: GET [http://127.0.0.1:8000/api/feedbacks\\_read](http://127.0.0.1:8000/api/feedbacks_read)

```
[
  {
    "id": 1,
    "comment": "Aperiam explicabo consectetur qui qui ea enim.",
    "read": true,
    "created_at": "2022-03-13T15:33:01.000000Z"
  }
]
```

### **GET /api/feedbacks\_not\_read**

Visszaadja a még nem olvasott visszajelzések adatait.

Például: GET [http://127.0.0.1:8000/api/feedbacks\\_not\\_read](http://127.0.0.1:8000/api/feedbacks_not_read)

```
[
  {
    "id": 3,
    "comment": "Molestiae molestiae totam dicta voluptate exercitationem.",
    "read": false,
    "created_at": "2022-03-13T15:33:01.000000Z"
  }
]
```

### **GET /api/read\_feedback\_count**

Visszaadja az olvasott visszajelzések számát.

Például: GET [http://127.0.0.1:8000/api/read\\_feedback\\_count](http://127.0.0.1:8000/api/read_feedback_count)

Eredmény: 7

### **GET /api/new\_feedback\_count**

Visszaadja az újonnan küldött visszajelzések számát.

Például: GET [http://127.0.0.1:8000/api/new\\_feedback\\_count](http://127.0.0.1:8000/api/new_feedback_count)

Eredmény: 8

### **POST /api/login**

Visszaadja a felhasználó **token**-jét, amivel megkaphatjuk a felhasználó adatait.

Például: POST <http://127.0.0.1:8000/api/login>

Eredmény: 1ljXgAZiSVq9o6gyFw6uDsH3niXSLFT5n9VGtU6Hk1



## 5.4 DogGo Admin Interface – Asztali alkalmazás

A program a web és mobil alkalmazásoktól abban tér el, hogy nem jelenít meg térképet és csak bizonyos felhasználók számára érhető el.

Az interneten való kommunikációban nagyon sok ember megfordul. Az egyik ember véleménye sértő vagy bántó lehet egy másik ember számára ezért nagyon fontos a hozzászólások moderálása. Az adminok látják, hogy valaki obszcén, trágár megjegyzéseket tesz, törölheti azokat vagy akár a felhasználót is kitilthatja. Esetleg valaki véletlenül vagy direkt rossz helyet jelölne meg, erre is megoldást nyújt a felület, mivel egy adminnak vizsgálat után, engedélyeznie kell a jelölést. Rossz, nem odaillő nevek esetén is van lehetősége az átírára.

Minden programnak vannak hibái, ezért egy visszajelzés funkciót is beletettünk, így a felhasználók anonim üzeneteket küldhetnek, hibákról, bővítési lehetőségekről, trágár tartalmakról, felhasználókról vagy helytelen megjelölésekről.

Az engedélyezésre váró helyszínek, olvasatlan visszajelzések számáról, az alkalmazás irányítópult menüpontja alatt vehetnek tudomást.

### 5.4.1 *Alkalmazott fejlesztői eszközök*

Grafikus tervet a Balsamiq nevű szoftverben készítettem el. Pár kattintással meg lehet csinálni egy mobil app, weboldal, vagy asztali alkalmazás drótvázát.

A fejlesztést C# nyelven, WPF keretrendszerrel kezdtem el, de a kezdeti nehézségek miatt, mint például táblázatba betölteni az adatokat vagy új ablakot nyitni az adott sorára kattintva. Ezért inkább JavaFX-ben kezdtem újra. Ami problémák C#-ban felmerültek az ablak tervezése során, az a Java által támogatott Scene Builder segítségével egyszerűen megoldható volt. A forrásprogram szerkesztésére IntelliJ-t használtam. Az ebbe integrált Scene Builder nem annyira használható, nem beépített Scene Builder is tartalmaz hibákat. Néha nem lehet átállítani a betűtípust, méretet, de ezeken kívül mást nem tapasztaltam. Json formátumú fájlok konvertálásához a Google Gson könyvtárát használtam. Az alkalmazáson használt stílusokat külön CSS állomány tartalmazza, így a későbbi változtatásokat egyszerűen végre lehet hajtani.

### 5.4.2 *Specifikáció*

Ebben a részben ismertetem a rendszer főbb funkcióit, feldolgozandó adatok körét, az azokra vonatkozó megszorításokat.

A program minden funkciójához bejelentkezés szükséges.

#### **Bejelentkezés**

A bejelentkezéshez egy regisztrált, admin jogosultsággal rendelkező felhasználó szükséges. A belépéshez meg kell adnunk a felhasználónevet és a jelszót. Ha nem megfelelő felhasználónevet, jelszót adunk meg vagy nem rendelkezünk admin jogosultságokkal, esetleg üresen maradt az egyik mező az alkalmazás hibaüzenettel jelzi a felhasználónak a problémát.

#### **Felhasználók kezelése**

A tábla kívánt sorára duplát kattintva jelenik meg az ablak, ahol a felhasználót tudjuk tiltani/feloldani vagy hozzászólásait törölni. Admin jogot csak a superadmin adhat, illetve vehet el. Jogosultsággal rendelkező felhasználót nem lehet kitiltani.

#### **Helyszínek kezelése**

Dupla kattintásra az adott sor tartalma egy új ablakban jelenik meg, ahol engedélyezhetjük a helyszín megjelenését, módosíthatjuk a nevét, leírását. Nevének kötelező 5 karaktert megadni, leírás nem kötelező. Mentés gombra kattintva menthetjük a változtatásokat, egyébként nem történik meg a módosítás. Ezek mellett lehetséges a jelölés törlése, ha nincs elem kijelölve, akkor a gombra kattintani nem lehet, új helyszínt is lehet hozzáadni. Leíráson kívül mindent kötelező megadni. Legördülő listán kiválasztott megkötések segítségével szűrhetjük a listát (összes, engedélyezett, engedélyezésre vár).

## **Visszajelzések**

Dupla kattintásra az adott sor tartalma egy új ablakban jelenik meg, ahol a teljes szöveget láthatjuk és beállíthatjuk, hogy már olvastuk.

Itt is lehetőségünk van törlésre, illetve szűrésre olvasás alapján.

## **Kijelentkezés**

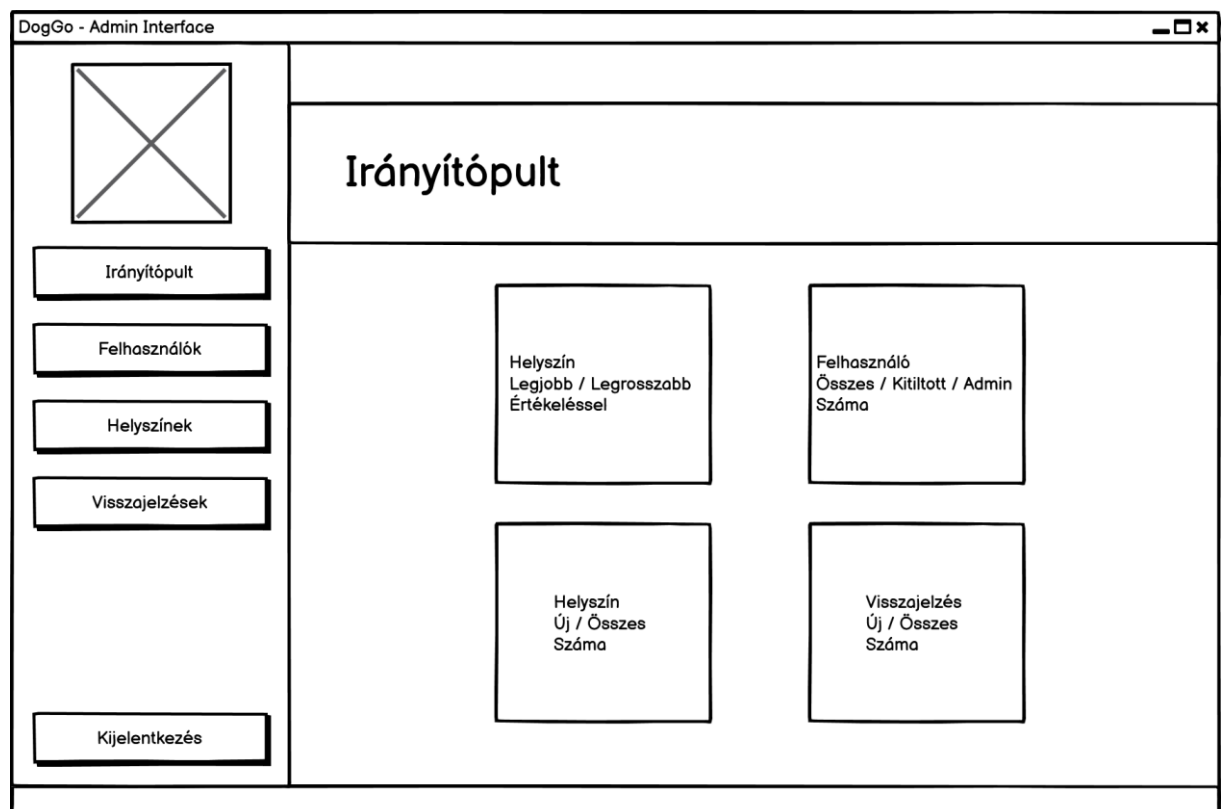
Kijelentkezés gombra kattintva tudunk kijelentkezni a programból. Bezárás után ugyan úgy kijelentkezünk, ezzel biztosítva azt, hogy illetéktelen ember nem használhatja.

### 5.4.3 *Felületterv*

Az alkalmazást próbáltam felhasználóbaráttá tenni, a kinézetnél egyszerű és letisztult dizájnrá törekedtem. A JavaFx lehetővé teszi a CSS fájlok használatát, aminek köszönhetően nagyobb körben tudjuk alakítani a kinézetet egyszerűen.

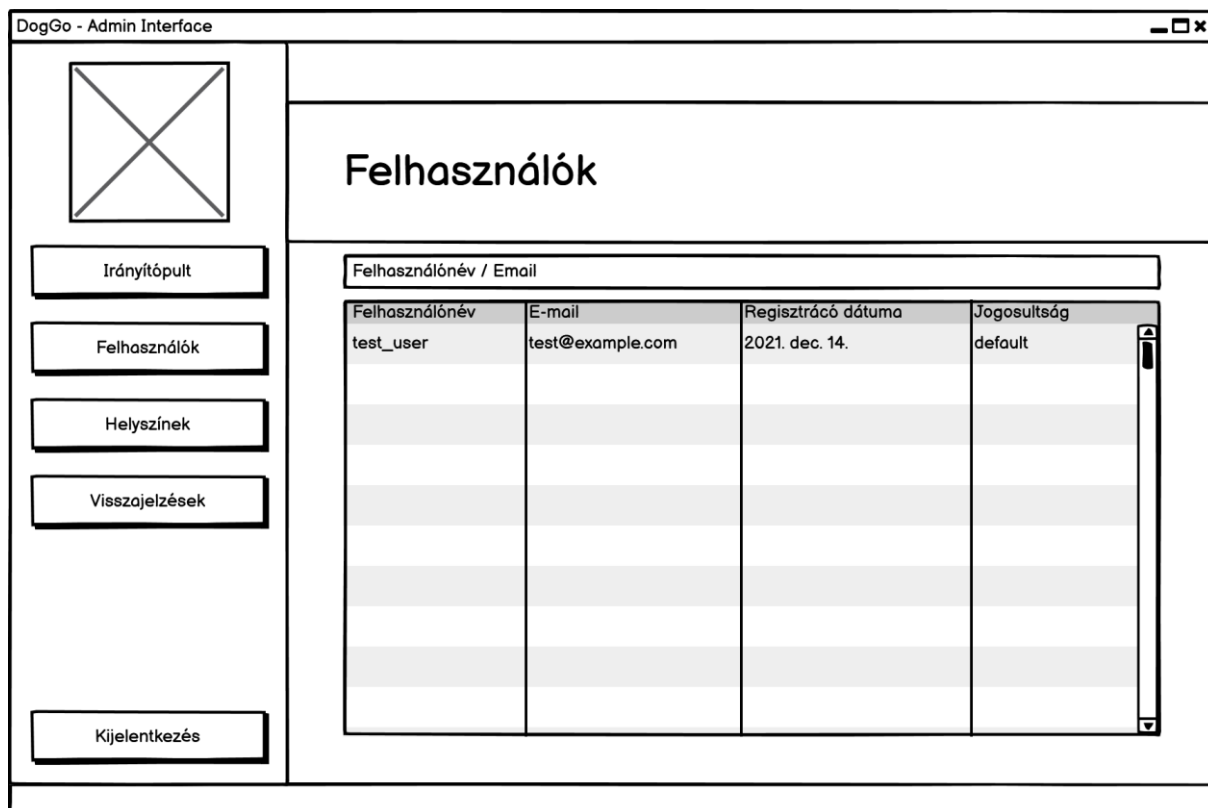
Egy fő ablak van, amelyet egy BorderPane-nel oldottam meg, ezzel könnyedén lehet változtatni a középső rész tartalmát és minden menüpont alatt a bal oldali rész, és a címsor nem változik.

Irányítópult menüpont alatt TabPane segítségével válogathatunk, hogy éppen mit szeretnénk látni, legjobb vagy legrosszabb értékeléssel rendelkező vagy engedélyezésre váró helyszínek számát, olvasatlan visszajelzések darabszámát.



4. ábra: Felületterv - Irányítópult

A többi jelenet nagyon hasonló megjelenést kapott. Az adatok egy TableView-ban jelennek meg, ami lehetőséget ad az oszlopok rendezésére. Egy beviteli mező is található a táblázat felett, amivel kereshetünk az adatok között.



5. ábra: Felületterv - Felhasználók

Felhasználók kezelésére új ablak nyílik meg, amin található egy TableView, TextField, felhasználó adatai és kettő gomb, plusz egy a superadminnak.

Felhasználó

test\_user

Hozzászólás törlése

Tiltás / Feloldás

Admin

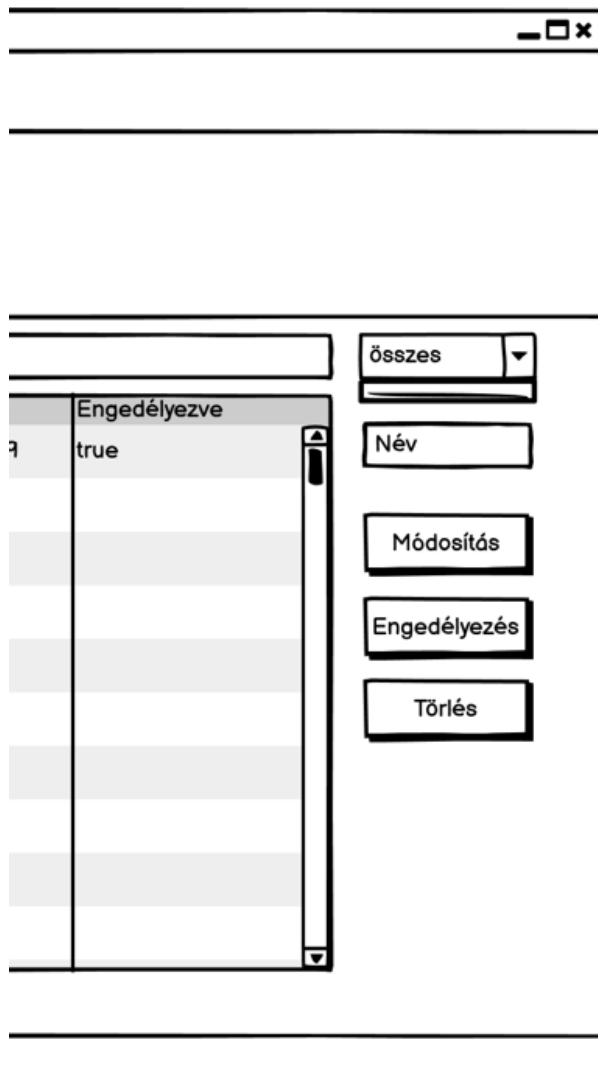
Hozzászólás

Hozzászólás

Hozzászólás	Csillagok
test_comment	5

6. ábra: Felületterv - Felhasználók kezelése

A többi nézet csak minimálisan tér el a Felhasználók nézetétől. A táblázat mellé bekerült egy legördülő lista, beviteli mező és három gomb. A legördülő listával szűrhetjük a lista tartalmát, a gombokkal műveleteket végezhetünk az adatokon.



7. ábra: Felületterv - Helyszínek

Fejlesztés során, a táblázatban megjelenő adatok nem minden esetben jelentek meg szépen, ezért szebbnek láttam, és egyszerűbben kezelhetőnek láttam, hogy egy külön ablakban jelenítsem meg a módosítható tartalmakat és a műveleteket végrehajtó gombokat.

The diagram shows a window titled "Helyszín". It contains a text input field labeled "Név", a larger text area labeled "Leírás", and three buttons: "Engedélyezés", "Mentés", and "Vissza".

8. ábra: Felületterv - Helyszínek kezelése (új)



#### 5.4.4 Implementáció

Az alkalmazást objektumorientáltan készítettem el, az adatokat API hívásokkal kapja a backendtől az adatbázisból. A következő fejezetben a fontosabb osztályokat, kód-sorokat fogom bemutatni.

Felhasználó	
Adattag	Leírás
- id : int	Azonosító.
- username : String	Felhasználónév.
- email : String	Email cím.
- created_at : Date	Regisztráció dátuma.
- permission : int	Jogosultság.
Metódus	Leírás
Tartalmazza az adattagokhoz tartozó getter és setter metódusokat.	
+ getFormattedDate() : String	A created_at dátumát alakítja "yyyy MMM d." formátumúra.
+ getFormattedPermission() : String	A permission-t alakítja szöveges formátumra.

Helyszín	
Adattag	Leírás
- id : int	Azonosító.
- name : String	Név.
- description : String	Leírás
- lat : double	Szélesség.
- lng : double	Hosszúság.
- allowed : boolean	Engedélyezve van-e.
Metódus	Leírás
Tartalmazza az adattagokhoz tartozó getter és setter metódusokat.	
+ getFormattedStatus() : String	Az allowed adattag értékét alakítja szöveges formátumra.

Controller	
Adattag	Leírás
- dialogPane : DialogPane	DialogPane objektum
- stage : Stage	Stage objektum.
Metódus	Leírás
+ getStage()	Visszaadja a stage-et.
- newWindow(String fxml, String cím, int szélesség, int magasság)	Visszaad egy új ablakot, paraméterként vár egy fxml fájlt, a címet, az ablak szélességét és magasságát. Ha hiba van az fxml fájlal, akkor IOException-t dob.
- confirmation(String üzenet)	Visszatér egy logikai értékkel, ami a felugró ablakkal eredménye. Az ablakban a megadott üzenet jelenik meg.
- error(Exception exception)	A hiba tartalmával jelenik meg a felugró ablak, 500 milliszekundum késleltetéssel.
- alert(String üzenet)	Az átadott üzenetet (siker, sikertelen, figyelmeztetés) jeleníti meg felugró ablakban.
- dragWindow(Stage stage, MouseEvent event, double x, double y)	Az UNDECORATED ablak mozgatását teszi lehetővé a kijelző X, Y koordinátáinak használatával.

FelhasznokController extends Controller	
Adattag	Leírás
- userList : ObservableList<Felhasznalo>	Felhasznalo objektumokat tároló lista.
Metódus	Leírás
+ initialize()	Megadja a TableView oszlopainak, hogy a Felhasznalo osztály, melyik GET függvényeit hívják meg a feltöltéshez. Meghívja a userListUpload() és search() metódusokat.
- userListUpload()	Háttérben futó szálon hívja meg az FelhasznaloApi getUsers() függvényét, és az attól kapott adatokkal tölti fel a userList-et. Adatok lekérése közben hiba történik, IOException-t dob.
- search()	Keresés mezőbe, karakterek leütése után egy predikatet állít be a filterdListre, ami logikai értékkel tér vissza. Ezután a filteredList lesz a sortedList tartalma, és a táblázatot feltölti ebből a listából.
+ FXML onUserDoubleClick(MouseEvent mouseEvent)	A táblázat bármelyik elemére duplán kattintva, a Controller newWindow() metódusát hívja meg és a felhasználok-reszletes-view.fxml fájlt használja. Az ablakon megjelenő adatokat a Felhasznalo-ReszletesController setReszletes() metódusa adja át. Az ablak bezárása után a TableView frissül és a módosított adatok jelennek meg.

FelhasznalokReszletesController extends Controller	
Adattag	Leírás
- reszletes : Felhasznalo	Felhasználó objektumot tárol.
- ratingList : ObservableList<Ertekeles>	Ertekeles objektumokat tárol.
- stage : Stage	Stage objektum.
- x : double	Ablak X koordinátáját tárolja.
- y : double	Ablak Y koordinátáját tárolja.
Metódus	Leírás
+ getReszletes()	Visszaad egy Felhasznalo objektumot.
+ setReszletes(Felhasznalo reszletes)	Beállítja a Felhasznalo objektum adatait. Meghaívja az writeData() metódust.
- writeData()	Meghívja a loadUserData().eljárást. Megadja a TableView oszlopainak, hogy az Ertekeles osztály, melyik GET függvényeit hívják meg a feltöltéshez. Beállítja a gombok láthatóságát a bejelentkezett admin és a használhatóságukat a kezelt felhasználó jogosultságai alapján.
- loadUserData()	Beállítja az ablakon a felhasználó adatait egy Label-be. A jogosultságot átformázza szövegre.
- ratingListUpload()	Háttérben futó szálon hívja meg az ErtekelesApi getUserRatings() függvényét, és az attól kapott adatokkal tölti fel a listát és a TableView-t. Adatok lekérése közben hiba történik IOException-t dob.
- ban()	Beállítja a felhasználó jogosultságát 1-re (kitiltott), FelhasznaloApi updateUserPermission() metódusát használva, majd meghívja a loadUserData() eljárást. A tiltás sikerességéről üzenetet küld.

- unban()	Beállítja a felhasználó jogosultságát 0-ra (default), FelhasznaloApi updateUserPermission() metódusát használva, majd meghívja a loadUserData() eljárást. A feloldás sikerességéről üzenetet küld.
- admin()	Beállítja a felhasználó jogosultságát 2-re (admin), FelhasznaloApi updateUserPermission() metódusát használva, majd meghívja a loadUserData() eljárást. A jogosultság módosításának sikerességéről üzenetet küld.
- unadmin()	Beállítja a felhasználó jogosultságát 0-ra (default), FelhasznaloApi updateUserPermission() metódusát használva, majd meghívja a loadUserData() eljárást. A jogosultság módosításának sikerességéről üzenetet küld.
- search()	Keresés mezőbe, karakterek leütése után egy predikatet állít be a filterdListre, ami logikai értékkel tér vissza. Ezután a filteredList lesz a sortedList tartalma, és a táblázatot feltölti ebből a listából.
+ FXML onDescriptionDeleteClick(ActionEvent actionEvent)	A TableView kiválasztott elemére meghívja az ErtekelesApi deleteDescription() függvényét, ami üresre állítja az értékelés leírását. A listát és a TableView tartalmát újra tölti. Sikeres törlés esetén a gomb disable tulajdonságát igazra állítja, a TextArea-ból kitörli a hozzászólást. A művelet eredményéről üzenetet küld.

+ FXML onDescriptionClick(MouseEvent mouseEvent)	A TableView kiválasztott elemére kattintva, a teljes hozzászólás egy TextArea-ban jelenik meg, a törlés gomb disable tulajdonságát hamisra állítja.
+ FXML onUserBanClick(ActionEvent actionEvent)	Felhasználó jogosultsága alapján állítja át a gomb szövegét és hívja meg a ban() vagy unban() eljárásokat.
+ FXML onCloseClick(Event event)	Bezárja az ablakot.
+ FXML onMinimizeClick(Event event)	Kisméretűre állítja az ablakot.
+ FXML onBorderPaneTopDragged(MouseEvent mouseEvent)	Lehetővé teszi az ablak mozgatását, meghívja a dragWindow() metódust.
+ FXML onBorderPaneTopPressed(MouseEvent mouseEvent)	Megszerzi az ablak X és Y elhelyezkedését.

HelyszinekController extends Controller	
Adattag	Leírás
- locationList : ObservableList<Helyszin>	Helyszin objektumokat tároló lista.
Metódus	Leírás
+ initialize()	Megadja a TableView oszlopainak, hogy a Helyszin osztály, melyik GET függvényeit hívják meg a feltöltéshez. Emellett meghívja a locationListUpload(), search() és filter() metódusokat.
- search()	Keresés mezőbe, karakterek leütése után egy predikatet állít be a filteredListre, ami logikai értékkel tér vissza. Ezután a filteredList lesz a sortedList tartalma, és a táblázatot feltölti ebből a listából.

- locationListUpload()	Háttérben futó szálon hívja meg az HelyszinApi getLocation() függvényét, és az attól kapott adatokkal tölti fel a locationList-et. Adatok lekérése közben hiba történik IOException-t dob.
- filter()	ChoiceBox-ből kiválasztott elem alapján a HelyszinApi megfelelő GET metódusát (getNewLocations(), getAllowedLocations()) vagy a locationListUpload()-ot hívja meg, és tölti fel a listát az attól kapott adatokkal.
+ FXML onLocationDeleteClick(ActionEvent actionEvent)	A TableView kiválasztott elemére meghívja a HelyszinApi deleteLocation() függvényét, ami törli az elemet az adatbázisból. Sikeres törlés esetén a gomb disable tulajdonságát igazra állítja. A művelet eredményéről üzenetet küld.
+ FXML onLocationDoubleClick(MouseEvent mouseEvent)	A táblázat bármelyik elemére duplán kattintva, a Controller newWindow() metódusát hívja meg és a helyszin-muveletek-view.fxml fájlt használja. Az ablakon megjelenő adatokat a HelyszineMuveletekController setReszletes() metódusa adja át. Az ablak bezárása után a TableView frissül és a módosított adatok jelennek meg.

HelyszinMuveletekController extends Controller	
Adattag	Leírás
- reszletes : Helyszin	Helyszin objektumot tárol.
- stage : Stage	Stage-et tartalmazó objektum.
- allowed : boolean	Logikai érték, hogy engedélyezve van-e a helyszín.
- saved : boolean	Logikai érték, hogy mentve vannak-e a változtatások, alap értéke true.
- x : double	Ablak X koordinátáját tárolja.
- y : double	Ablak Y koordinátáját tárolja.
Metódus	Leírás
+ getReszletes()	Visszaad egy Helyszin objektumot.
+ setReszletes(Helyszin reszletes)	Beállítja a Helyszin objektum adatait. Meghaívja az writeData() metódust.
- writeData()	Beállítja a TextField-be a kiválasztott helyszin nevét az objektum getName() függvényének meghívásával és a TextArea értékét a getDescription() függvény hívásával.  Az engedélyezve logikai változó értékét az isAllowed() metódus adja meg és a gomb disable tulajdonságát állítja be.
- save()	Beolvassa az input mezőkbe megadott adatokat, ellenőrzi, ha hiba van, azt jelzi. Helyes adatok esetén a HelyszinApi updateLocation() metódusát hívja meg és a változtatások bekerülnek az adatbázisba. Saved értékét igazra állítja.
- close()	Ha nem történt változtatás bezárja az ablakot, ellenkező esetben, ha nincsenek elmentve a változtatások, felugró ablakban kérdezi meg, hogy biztosan kilépünk



	mentés nélkül, vagy szeretnénk-e menteni. A mentéshez a save() eljárást hívja meg.
+ FXML onSaveClick(ActionEvent actionEvent)	Meghívja a save() eljárást.
+ FXML onBackClick(ActionEvent actionEvent)	Meghívja a close() eljárást.
+ FXML onAllowClick(ActionEvent actionEvent)	Igazra állítja az engedelyezve értékét és a gomb disable tulajdonsága igaz, a mentve pedig hamis értéket kap.
+ FXML onCloseClick(ActionEvent actionEvent)	Meghívja a close() eljárást.
+ FXML onMinimizeClick(Event event)	Kisméretűre állítja az ablakot.
+ FXML onBorderPaneTopDragged(MouseEvent mouseEvent)	Lehetővé teszi az ablak mozgatását, meghívja a dragWindow() metódust.
+ FXML onBorderPaneTopPressed(MouseEvent mouseEvent)	Megszerzi az ablak X és Y elhelyezkedését.

#### 5.4.5 Továbbfejlesztési lehetőségek

Tervezett funkcióként indult, de idő hiányában nem sikerült megvalósítani, hogy a helyszínekhez lehessen képeket csatolni, és ezeket a képeket az asztali alkalmazásban megjelenítve, kezelni. Sok felhasználó számára hasznos lenne ez a funkció, hiszen vizuális képet kapnak az adott környezetről.

Továbbfejlesztési lehetőség lehet, hogy kibővítjük az adatbázist, és eltároljuk, hogy melyik admin milyen változtatásokat hajtott végre, felhasználók tiltása, helyszínek módosítása, törlése. Ezzel nyomon követhetjük, személy szerint az adminok munkáját. Biztonságosabbá tenné az alkalmazást egy olyan funkció, ami kiléptetné a felhasználót adott idő utána, hogyha nem használja a felületet.

## 6 Tesztelési dokumentáció

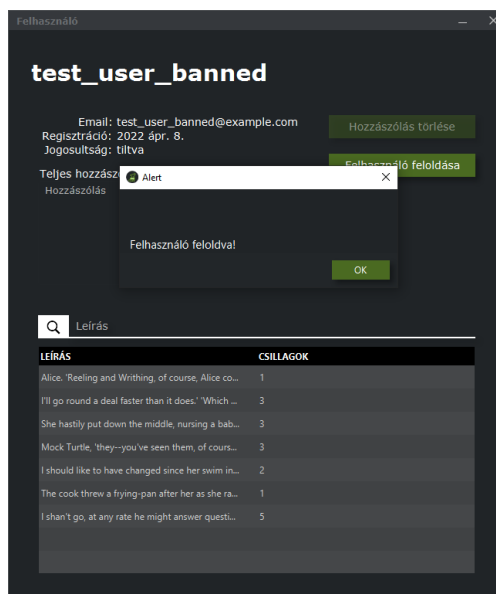
Tesztelés szükséges, ahhoz, hogy az alkalmazásunkban talált hibákat üzembe helyezés előtt kijavítsuk és arról, hogy minden, a specifikációban megfogalmazottaknak megfelelően működjön. A tesztelést 17-es verziójú Java JDK-val, Windows 10 Pro operációs rendszeren hajtottam végre.

Teszteléshez használt regisztrált felhasználók:

Jogosultság	superadmin	admin	default
Felhasználónév	superadmin	admin	test_user_1
Jelszó	admin	admin	test

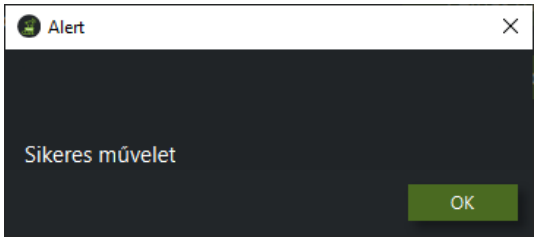
### Felhasználó kezelése

Művelet	Bemenet	Elvárt eredmény	Végeredmény
Admin jogosultsággal kitiltott felhasználó feloldása.	-	Felugró ablakban: „Sikeres feloldás!”	Felugró ablakban: „Sikeres feloldás!”

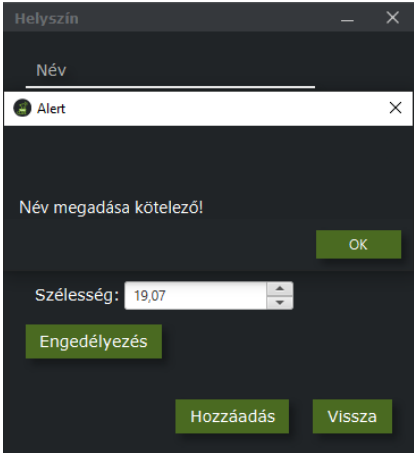


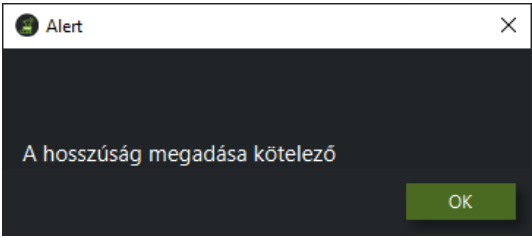
9. ábra: Felhasználó feloldása

Admin jogosultsággal felhasználó tiltása.	-	Felugró ablakban: „Sikeres tiltás!”	Felugró ablakban: „Sikeres tiltás!”
---	---	--	--

Superadmin jogosultsággal admin jog elvétele.	-	Felugró ablakban: „Sikeres művelet!”	Felugró ablakban: „Sikeres művelet!”
 <p>10. ábra: Sikeres admin jogosultság elvétele (felugró ablak)</p>			

#### Helyszín hozzáadása

Művelet	Bemenet	Elvárt eredmény	Végeredmény
Üresen hagyott név mező.	Név: - Leírás: - Hosszúság: 47,5 Szélesség: 19,07	Felugró ablakban: „Név megadása kötelező!”	Felugró ablakban: „Név megadása kötelező!”
 <p>11. ábra: Hibás helyszín hozzáadás (hiányos név)</p>			
Üresen hagyott hosszúság mező.	Név: Városliget Leírás: - Hosszúság: - Szélesség: 19,08	Felugró ablak: „Hosszúság megadása kötelező!”	Felugró ablak: „Hosszúság megadása kötelező!”

 <p>12. ábra: Hibás helyszín hozzáadás (hiányos koordináták)</p>			
Sikeres hozzáadás.	<p>Név: Városliget</p> <p>Leírás: -</p> <p>Hosszúság: 47,51</p> <p>Szélesség: 19,08</p>	Felugró ablak: „Sikeres hozzáadás!”	Felugró ablak: „Sikeres hozzáadás!”

## 7 Felhasználói dokumentáció

### 7.1 Hardver és szoftver igény

#### Minimum rendszerkövetelmény:

Processzor: Intel Core I5-750

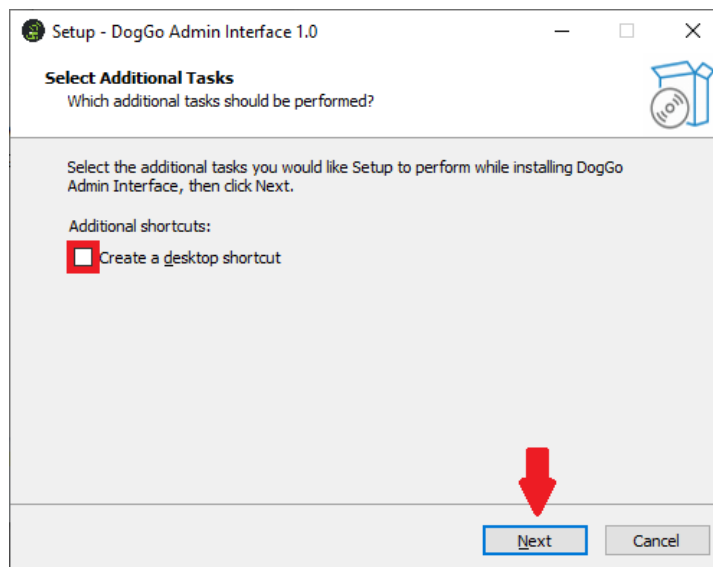
RAM: 4 GB

Operációs rendszer: Windows 10 64-bit

### 7.2 Telepítés

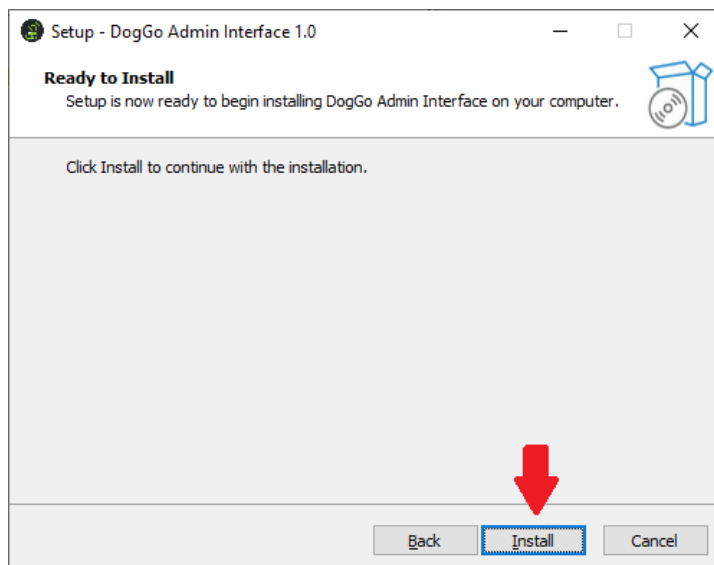
A telepítéshez dupla kattintással indítsuk el a doggo\_setup.exe fájlt. Válasszuk ki a megfelelő nyelvet (Angol / Spanyol), majd kattintsunk az „OK” gombra.

A pirossal jelzett jelölőnégyzet bepipálásával készíthetünk asztali ikont, amennyiben ezt szeretnénk. Tovább lépéshez kattintsunk a „Next” gombra.



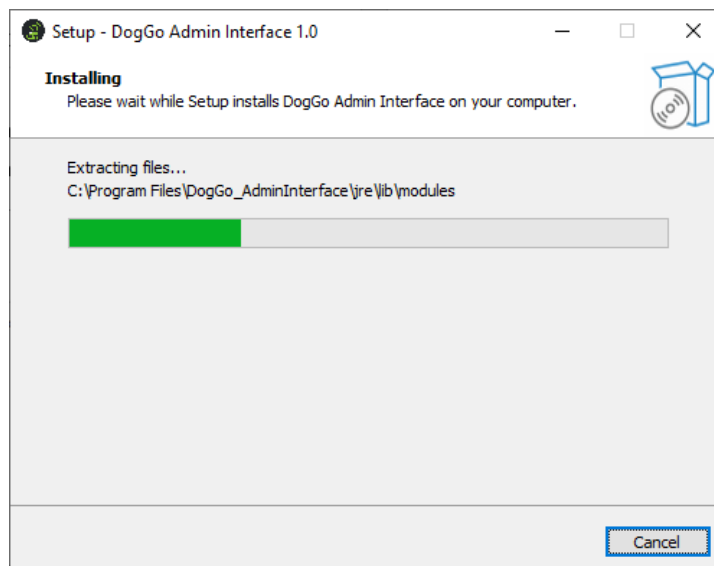
13. ábra: Asztali ikon telepítés

Következő ablakon már csak az „Install” gombra kell rákattintanunk, hogy elinduljon a telepítés, de még lehetőségünk van visszamenni az előző ablakra, hogyha mégsem szeretnénk asztali ikont vagy nem pipáltuk be.



14. ábra: Telepítés elindítása

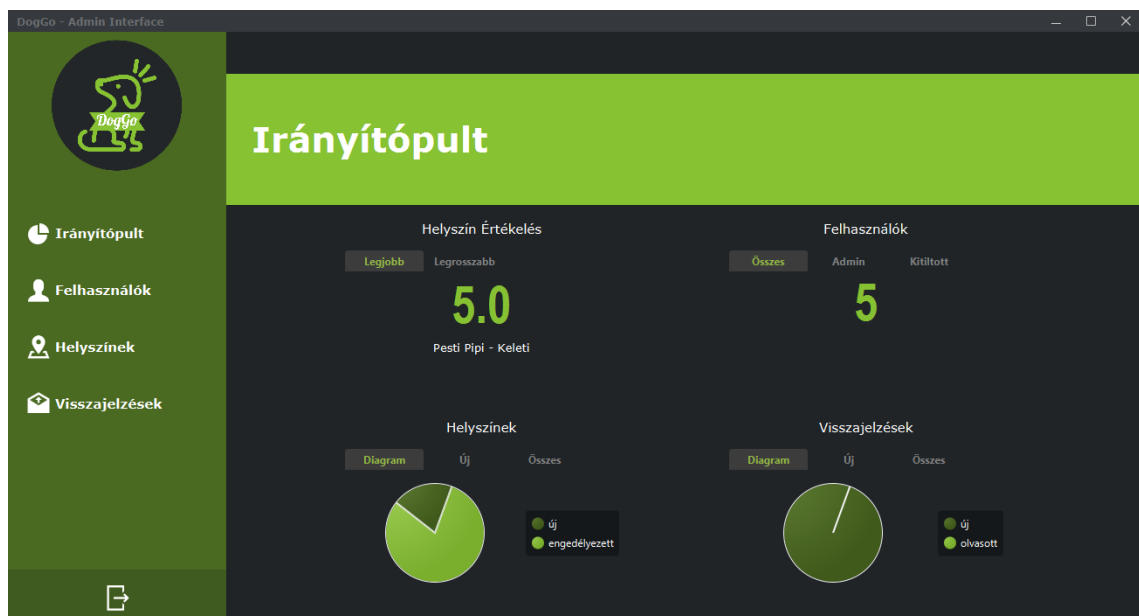
A végén már csak meg kell várnunk, amíg beget ér a telepítés. Eközben utolsó még megszakíthatjuk a telepítést a „Cancel” gombra kattintva.



15. ábra: Telepítés folyamata

Telepítés befejezése után indítsuk el az alkalmazást, melynek DogGo Admin Interface a neve. Először egy bejelentkezési felületet fogunk látni, ahol a felhasználónevet, jelszót kell megadni és admin szintű jogosultsággal kell rendelkezünk.

Helyes adatok beírása után, kattintsunk a bejelentkezés gombra. Sikeres bejelentkezés után a fő ablak „Irányítópult” menüpontján találhatjuk magunkat. Az ablak bal oldalán található a menüsáv, amivel navigálhatunk a különböző jelenetek között. A legelső gombra kattintva léphetünk ki.



16. ábra: Irányítópult

Ezen a felületen láthatjuk, hogy melyik helyszínekhez tartozik a legjobb és legrosszabb értékelés. Mennyi regisztrált felhasználó van és abból hányan adminok, és ezen kívül hány kitiltott van. Az összes, új helyszínek és visszajelzések darabszámát is mutatja.

## Felhasználók

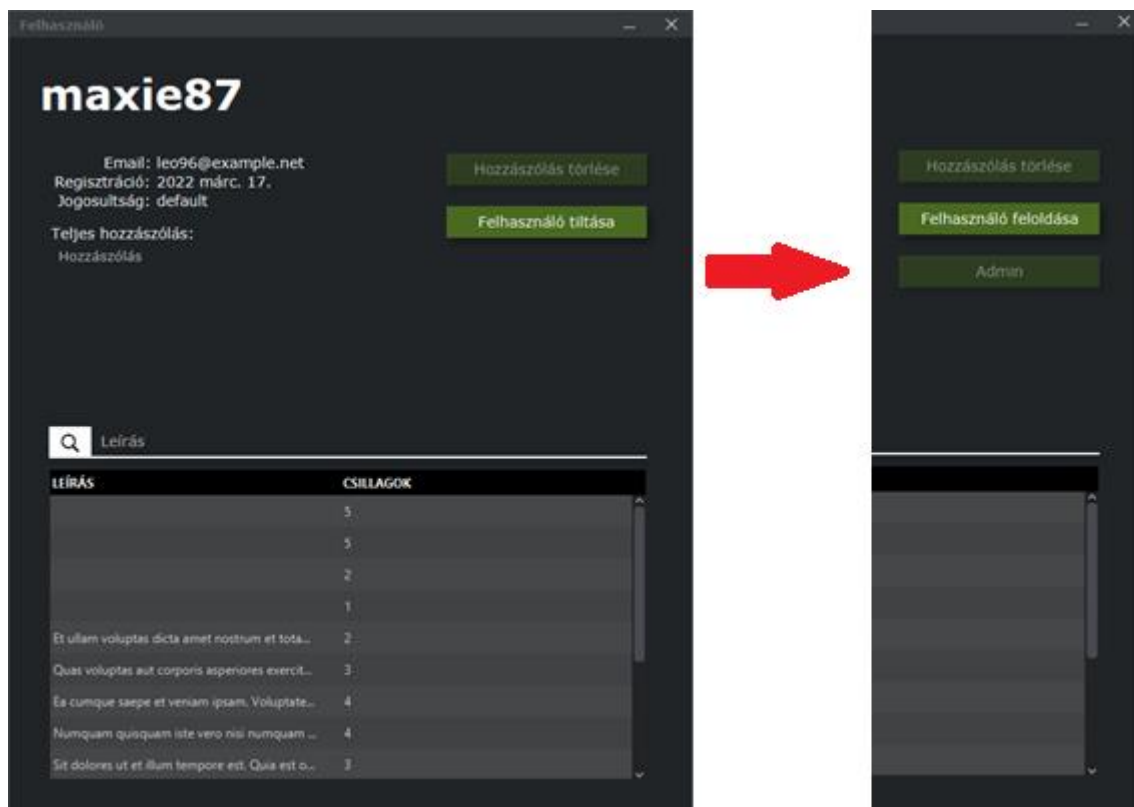


17. ábra: Felhasználók

A keresővel a felhasználónévre vagy emailre kereshetünk. Minden lenyomott billentyű után szűri a listát és semmi más gomb lenyomása nem szükséges hozzá.

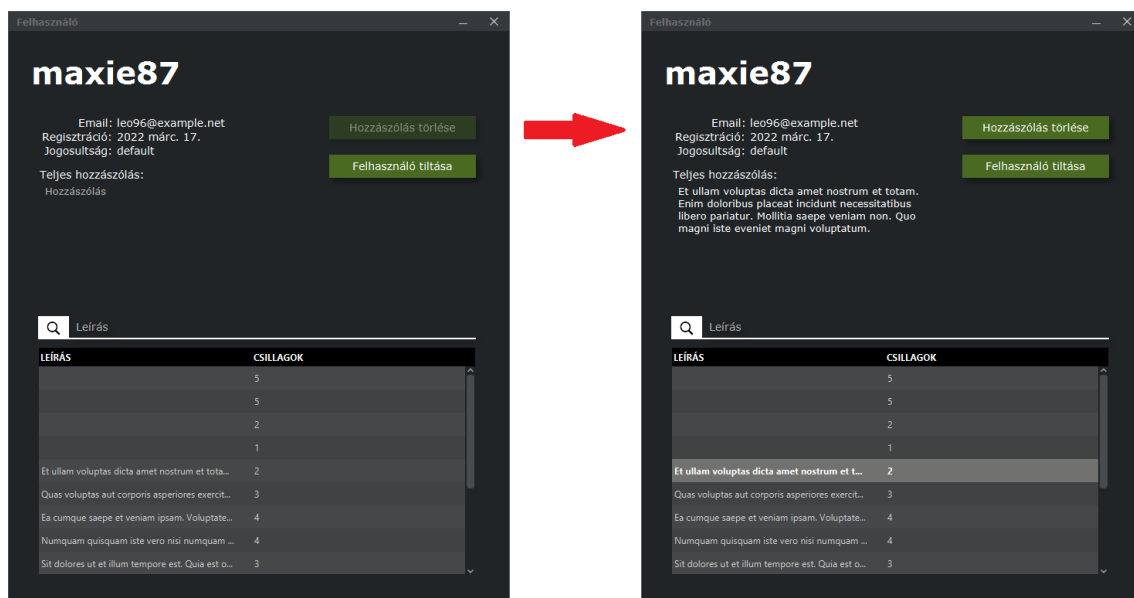
A kezelni kívánt felhasználóra duplán kattintva egy új ablak ugrik fel, ahol admin jogosultsággal kettő gombot találhatunk, és a felhasználó adatait, valamint az értékeléseit. SuperAdminként jelenik meg még egy gomb, amivel admin jogot adhatunk másoknak. Kitiltott felhasználónak nem adhatunk jogot, ilyenkor nem tudunk a gombra kattintani és ellenkező esetben, admint nem lehet kitiltani. Egyszerű adminoknak más adminoknál nem jelenik meg a tiltás gomb, SuperAdminoknál egyik sem jelenik meg.





18. ábra: Felhasználók kezelése (admin - superadmin)

A Hozzászólás törlése gomb csak akkor használható, ha ki van jelölve egy elem a táblázatban és annak van leírása. Ilyenkor a gombon jól látható, hogy rá lehet kattintani.



19. ábra: Hozzászólás kijelölése

Mielőtt törölnénk a hozzászólást egy megerősítő ablakban kell elfogadnunk a művelet végrehajtását. Az OK gombra kattintva törli a leírást.

## Helyszínek

A helyszínek kezeléséhez, a menüsávon kattintsunk a Helyszínek menüpontra.

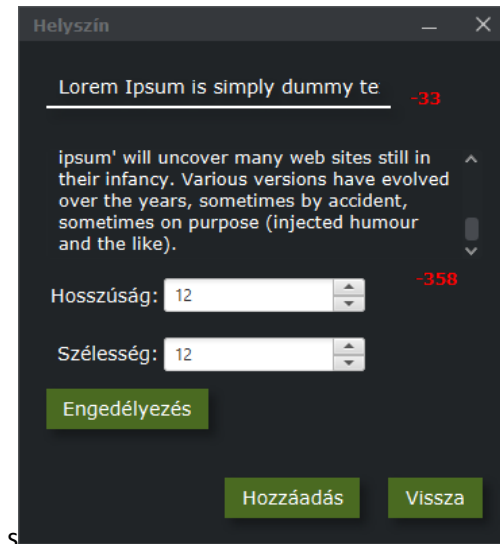
**Helyszínek**

Q Név / Leírás összes Törlés +

NÉV	LEÍRÁS	SZÉLESSÉG	HOSSZSÁG	ÁLLAPOT
McDonald's - Keleti	Gyors kiszolgálás, sok ember számára is elég nagy a hely, kiskedvence...	47.5008947	19.0819858	új
Burger King - Keleti	Kedves pultosok, nem a leghangosabb hely, kellemes az atmoszférája...	47.5008272	19.0817936	új
KFC Budapest - Keleti	Húzták a szájukat, de beengedtek minket kutyával. Az állatokat nem s...	47.5014546	19.0831529	engedélyezve
Pesti Pipi - Keleti	Kellemes volt itteni időtöltésem, külön tában adtak a kutyáinknak viz...	47.5012856	19.0827999	engedélyezve
Kutya Egy Hely kávézó/bár/k...	Nagyon kedves volt minenk, mindent érthetően le tudunk beszélni, ...	47.5060349	19.07362	engedélyezve
Városliget	Gyönyörű hely, sok kutyás ember jár erre, kutyáknak külön elkerített r...	47.51436501	19.08524642	engedélyezve
Normafa	A város zaja egyáltalán nem hallatszódik, olyan mintha nem is Budape...	47.506642529216	18.961865246037	engedélyezve
Margit-sziget	Kellemes élmény volt körbesétálni a szigetet. Több helyen is megálitu...	47.526773338103	19.046413139006	engedélyezve
Naplás-tó	Nagyon nyugodt hely, egy délutáni sétára tökéletes hely. A közelben ...	47.509116993639	19.247203915734	engedélyezve
Népliget	Kellemes csalogást okozott ez a hely, a kicsit hangos háttérzaj ellenér...	47.481421467218	19.109473615177	engedélyezve

20. ábra: Helyszínek

Ezen a felületen a kereső mezőn kívül egy legördülő listát és egy törlés gombot találunk. Legördülő listával szűrhetjük a táblázat elemeit állapotuk szerint. Csak az engedélyezettek, újak, vagy az összes megjelenik. Emellett található a törlés gomb, amire az elem kiválasztása után nyomhatunk, ezt előtte meg kell erősítenünk a végrehajtás érdekében. A törlés eredményéről visszajelzést küld. A „+” gombra kattintva érhetjük el a hozzáadási felületet. A leírás mellett minden kötelező kitölteni. Az engedélyezés gombbal egyből láthatóvá tehetjük. A név minimum 5, maximum 40 karakter lehet, a leírás legfeljebb 255 hosszúságú lehet, ezt a beviteli mezők közelében jelzi. A koordinátákat vesszővel elválasztva kell megadni.



21. ábra: Hibás helyszín módosítás

A kiválasztott helyszínen a műveleteket ugyancsak dupla kattintás után érjük el. A nevet és leírást tudjuk módosítani. Karakterszámra vonatkozó szabály ugyan az, mint a hozzáadásnál és ugyan úgy jelzi. Az ablak nagyon hasonló a hozzáadás ablakára, a hosszúság és szélesség beviteléhez szükséges kettő mező nem található meg.

Ha a helyszín már engedélyezve van, akkor nem lehet a gombra kattintani. Minden változtatást el kell mentenünk, ha nem kattintottunk rá a mentés gombra, akkor az ablak bezárása előtt rákérdez, hogy biztosan nem szeretnénk végrehajtani a módosításokat.

## Visszajelzések

A visszajelzések kezeléséhez, a menüsávon kattintsunk a Visszajelzések menüpontra. Megjelenése és a funkciói is megegyeznek a helyszínek kezelésével. Csak abban tér el, hogy a kezelési ablakon, egyedül azt tudjuk beállítani, hogy olvastuk már az üzenetet, és megjelenik a beküldés dátuma is.

## 8 Összegzés

Véleményem szerintem a vizsgaremek elkészítése során megtanultam nagyon sok új dolgot. Az elején problémák adóttak az időbeosztással, de ez hamar megoldódott és meghatározott fix időpontokban tudtam haladni. Legnagyobb fejlődést a hibák megoldásában értem el, mind vékony és vastag kliens alkalmazásokban. Megtanultam, hogyha igényes munkát szeretnénk végezni, akkor érdemes napokig keresni a megoldást.

Hasznos tapasztalatot szereztem, arról, hogy mégis milyen lesz éles helyzetben, csapatban, projekteken dolgozni. A csapatmunka felőlem nem a leggördülékenyebben indult be, de amint jobban megismertem csapattársaimat, a különböző kollaboratív szoftvereknek köszönhetően gyorsan, szervezeten tudtunk haladni.

### 8.1 Project Summary

Our project topic is about dog walking. The choosen topic must be unique, so we looked for an everyday problem which hasn't got a solution yet.

Lot of people have dogs, and only a few places allows dogs inside. In the social media, people ask if they can take their dog to a leisure park. We figured out an application, that can solve this problem.

The application shows us a map which has markers on it with places who allows dogs. People can mark his favorite locations by allow the GPS on his/her phone and write a description. Others can rate, and write a comment.

My part, in this project to create a desktop admin interface for moderators. On the Dashboard we can see the best and worst rated locations, users count, new / allowed locations count and new/ read feedbacks count.

Admins can delete comments, ban users who shares obscene content or mark fake locations. When someone add a new place, before it is shown on the map an admin have to allow it, after checked it it is real and the name or description does not have nasty words. Users can send feedbacks too if something is wrong with the application or other users.

I have got lot of experience about what would I do at work, and got good skills with working in a group. Sometimes it was hard to understand myself too the other teammates, but at the end I communicated easily with them.

## 9 Felhasznált források

Dizájn ötletek (Utolsó megnyitás 2021.12.12.)

<https://www.youtube.com/channel/UCJLK4lzsBtGtxr9aj-6Ok8Q>

Keresés megvalósítása (Utolsó megnyitás 2022.02.13.)

<https://www.youtube.com/watch?v=2M0L6w3tMOY>

Java SE 17 & JDK Dokumentáció (Utolsó megnyitás 2022.03.27.)

<https://docs.oracle.com/en/java/javase/17/docs/api/index.html>

JavaFX 17 Dokumentáció (Utolsó megnyitás 2022.03.27.)

<https://openjfx.io/javadoc/17/index.html>

Laravel Dokumentáció (Utolsó megnyitás 2022.03.26.)

<https://laravel.com/docs/9.x/installation>

Stackoverflow (Utolsó megnyitás 2022.04.13.)

<https://stackoverflow.com/>

JavaFX TableView CSS (Utolsó megnyitás 2022.01.15.)

<https://edencoding.com/style-tableview-javafx/>

Ikonok (Utolsó megnyitás 2022.03.12)

<https://icons8.com/icons>

<https://www.flaticon.com/>

## 10 Ábrajegyzék

1. ábra DogGo ütemterv.....	4
2. ábra: Használati eset diagram .....	8
3. ábra: Adatbázis terv .....	9
4. ábra: Felületterv - Irányítópult.....	28
5. ábra: Felületterv - Felhasználók.....	29
6. ábra: Felületterv - Felhasználók kezelése .....	30
7. ábra: Felületterv - Helyszínek.....	31
8. ábra: Felületterv - Helyszínek kezelése (új) .....	32
9. ábra: Felhasználó feloldása .....	42
10. ábra: Sikeres admin jogosultság elvétele (felugró ablak) .....	43
11. ábra: Hibás helyszín hozzáadás (hiányos név) .....	43
12. ábra: Hibás helyszín hozzáadás (hiányos koordináták) .....	44
13. ábra: Asztali ikon telepítés.....	45
14. ábra: Telepítés elindítása .....	46
15. ábra: Telepítés folyamata.....	46
16. ábra: Irányítópult.....	47
17. ábra: Felhasználók .....	48
18. ábra: Felhasználók kezelése (admin - superadmin) .....	49
19. ábra: Hozzászólás kijelölése .....	49
20. ábra: Helyszínek .....	50
21. ábra: Hibás helyszín módosítás.....	51