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: Korcsmáros Kristóf György

Csapattagok: Kilián Marcell András, Takács Balázs Levente

Konzulens: Czinkóczi Tamás

Budapest, 2021.04.19.

Tartalomjegyzék

Tartalomjegyzék 2
Plágium nyilatkozat 3
1 Bevezetés 4
1.1 Téma 5
1.2 Témaválasztás indoklása 6
2 Fejlesztői dokumentáció 7
2.1 Funkciók 7
2.2 Használati eset diagram 8
2.3 Backend 9
2.3.1 Adatbázis terv 9
2.3.2 Adattáblák (doggodb) 10
Backend telepítésének lépései 12
2.3.3 API végpontok 13
2.4 Mobil 25
2.4.1 Elképzelés 25
2.4.2 Alkalmazott fejlesztői eszközök 26
2.4.3 Sima osztályok 27
2.4.4 Activity osztályok 32
2.4.5 Tesztelési dokumentáció 38
2.4.6 Továbbfejlesztési lehetőségek 40
3 Felhasználói dokumentáció 41
3.1 A program általános specifikációja 41
3.2 Rendszerkövetelmények 41
3.3 A program telepítése 41
3.4 A program használatának részletes leírása 41
4 Összegzés 44
4.1 Summary 44
5 Források 45
6 Ábrajegyzék 46

Plágium nyilatkozat

Alulírott Korcsmáros Kristóf György felelősségem tudatában kijelentem, hogy a záródolgozat saját szellemi tevékenységem eredménye, az abban foglaltak más személyek jogszabályban rögzített jogait nem sértik.

A záródolgozat egy részét Kilián Marcell Andrással és Takács Balázs Leventével közösen készítettük el. Ezeket a részeket pontosan jelöltük.

Budapest, 2021.

.....	.....	.....
Kilián Marcell András	Korcsmáros Kristóf György	Takács Balázs Levente

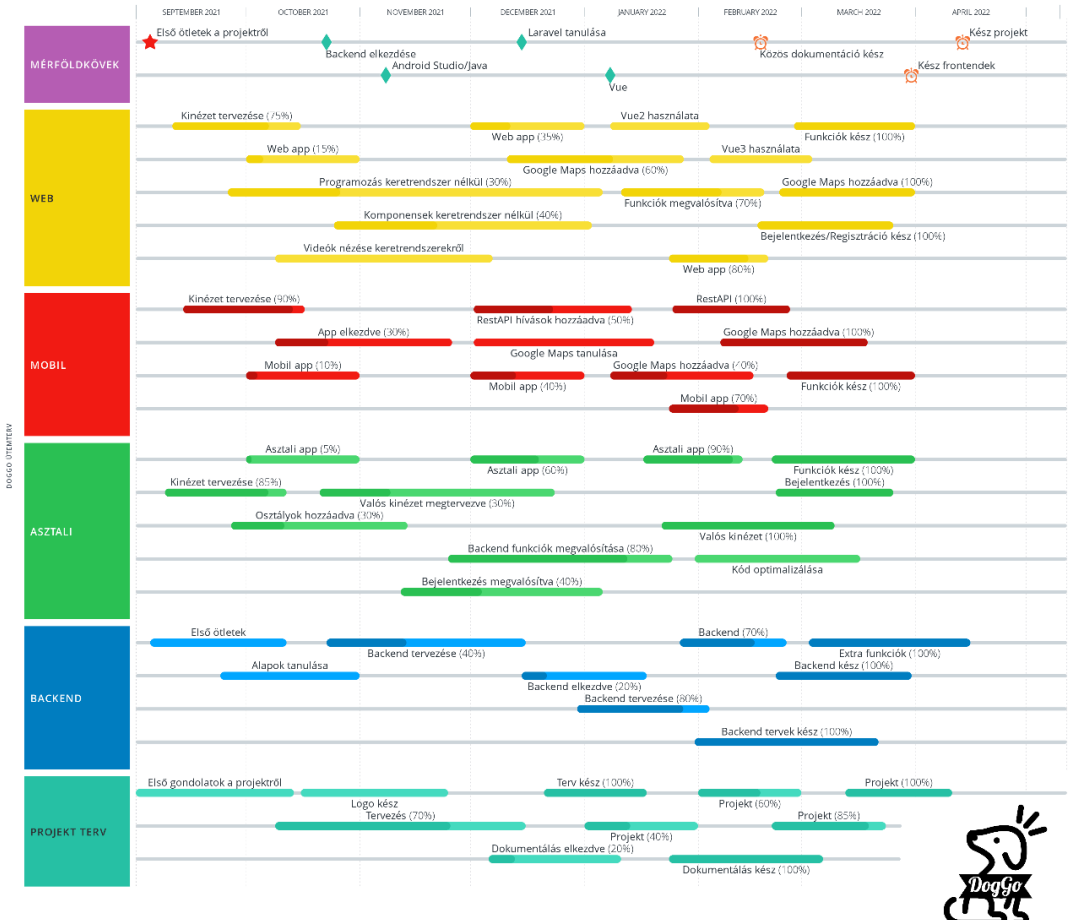
1 Bevezetés

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

1.1 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 egyeltalán 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.

1.2 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 mindenhová 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.

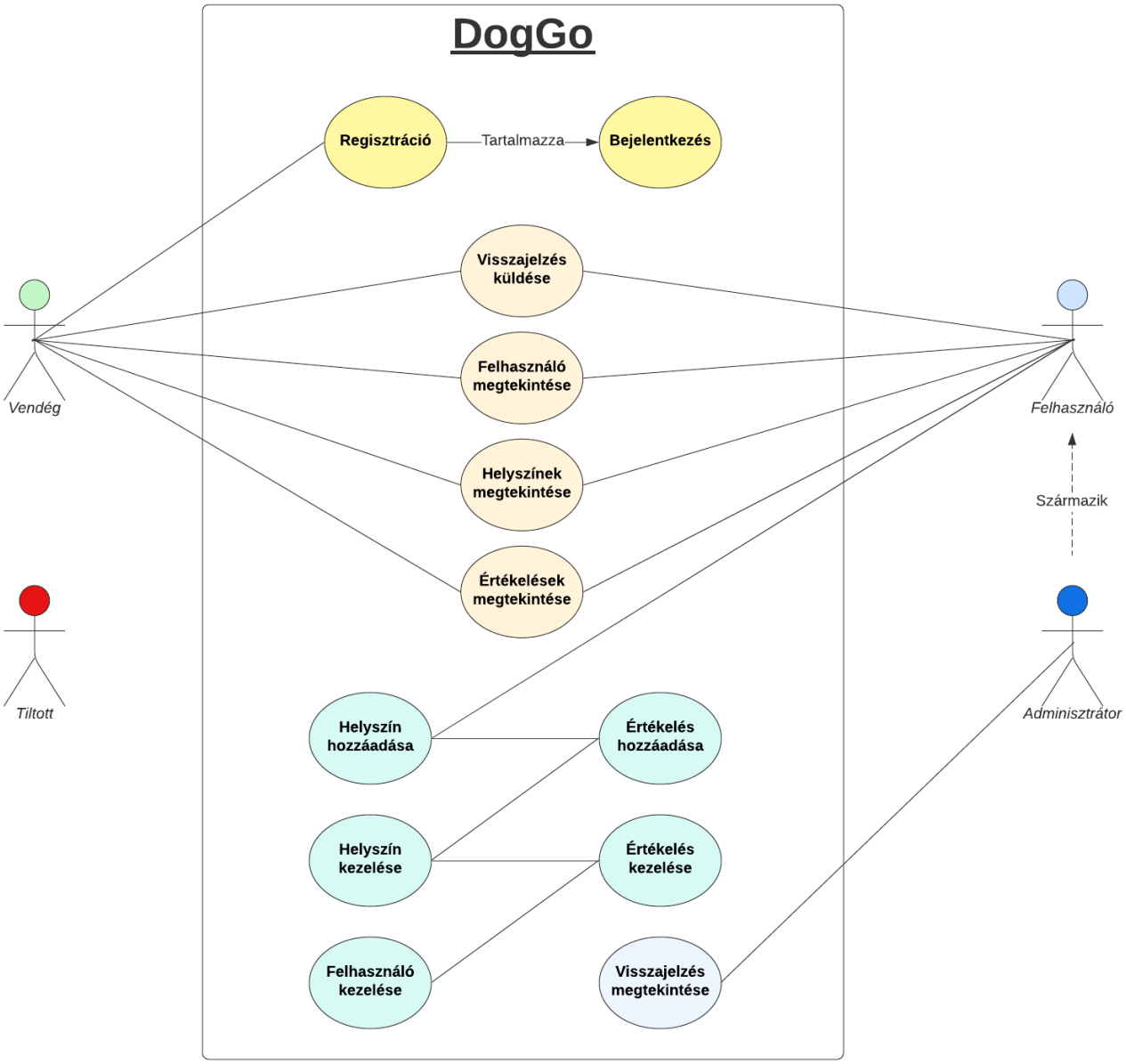
Ez az ötlet a fejlesztői csoport minden tagjának tetszett, a lelkesedést és egyetértést látva a projekt témája meg is született.

2 Fejlesztői dokumentáció

2.1 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	✘	✘	✔

2.2 Használati eset diagram



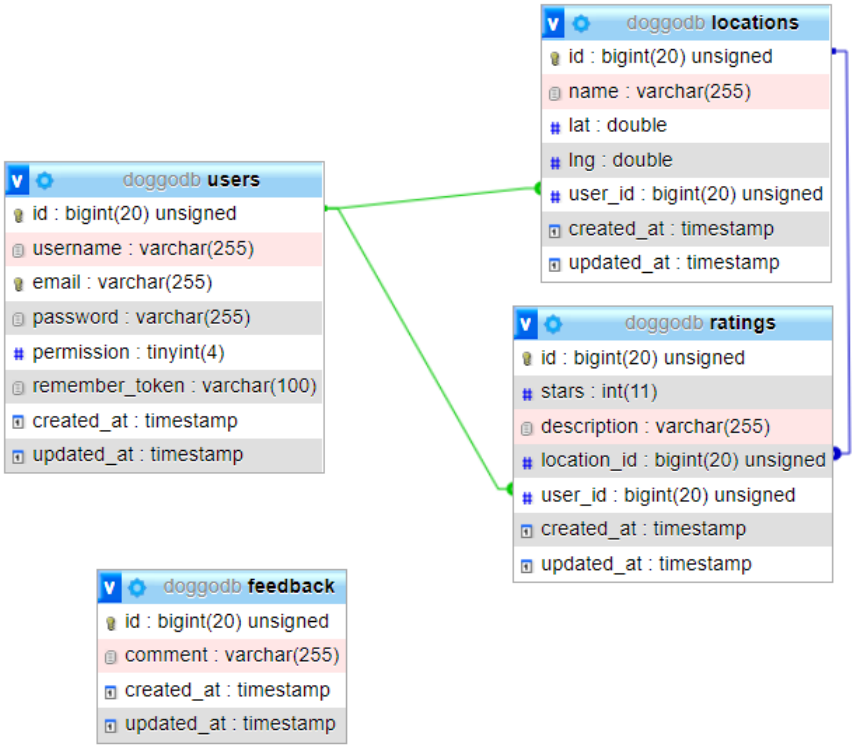
2. ábra: Használati eset diagram

2.3 Backend

2.3.1 *Adatbázis terv*

Az adatbázis az egyik legfontosabb rész a projekt működéséhez. Itt tároljuk a **felhasználók**, **helyek**, **értékelések** adatait és a **visszajelzések**et, 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

2.3.2 *Adattáblák (doggodb)*

users tábla

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

locations tábla

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

ratings tábla

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

feedback tábla

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

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* karakterkódolással.

A backend projekt megnyitása után, készítsünk egy másolatot a *.env.exmaple* fájlról és nevezzük át *.env*-re. A fájlban írjük 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 szervert:
- 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

2.3.3 *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.pdl3WbwNAuKciclb41St0fIFX2aSGvC

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 státuszkódot 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


```
{
   "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


```
{
   "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


```
[
   {
      "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


```
[
   {
      "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


```
[
   {
      "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


```
[
   {
      "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


```
[
   {
      "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

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

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: 1|jXgAZiSVq9o6gyFw6uDsH3niXSLFT5n9VGtU6Hk1

2.4 Mobil

2.4.1 *Elképzelés*

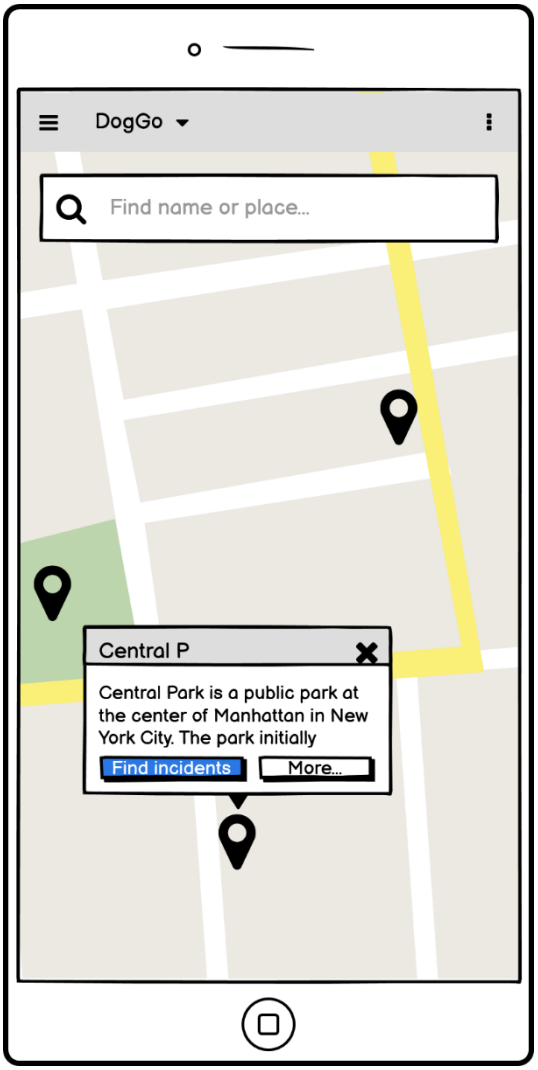
A DogGo mobil applikáció az átlagos felhasználónak készül, könnyen kezelhetőre és jó felhasználói élményre tervezve.

Az applikációval a felhasználó egy Google Maps térképen helyeket kereshet amelyet más felhasználók adtak hozzá, új hely hozzáadása regisztrációhoz és bejelentkézeshez kötött.

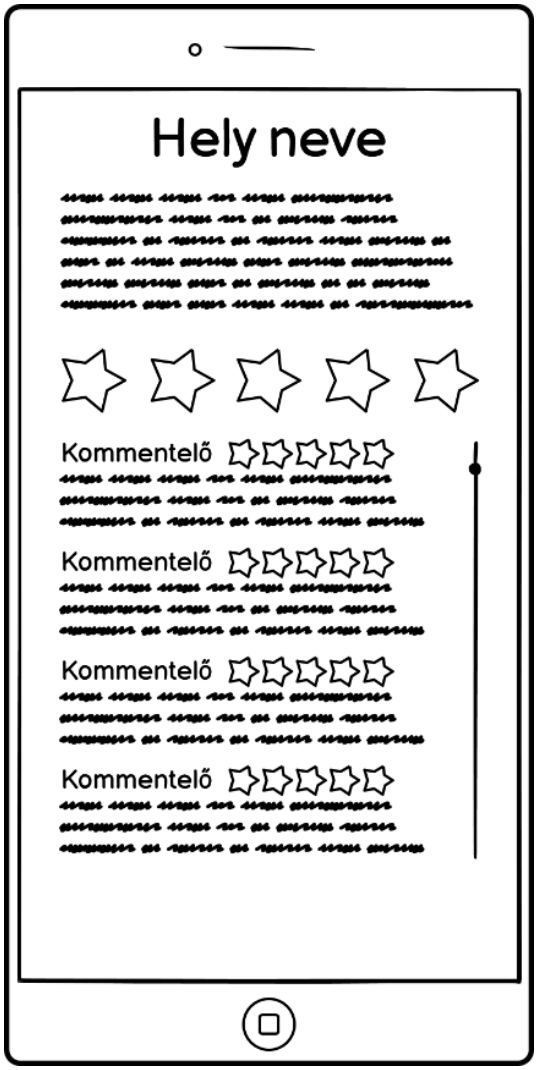
Egy helyre kattintva a térképen előugrik a kiválasztott hely adatlapja, amely kilistázza a hely adatait, valamint a hozzá tartozó értékeléseket. Értékelést hozzáadni bejelentkezett felhasználó tud.

Ha valami problémát észlel a felhasználó az alkalmazás használata során, névtelen visszajelzést is tud küldeni, amit az asztali alkalmazásban az adminisztrátor tud megtekinteni.

A felhasználó meg tudja nézni a saját adatlapját, valamint szerkeszteni is képes az adatait.



4. ábra: WireFrame térkép



5. ábra: WireFrame hely adatlapja

2.4.2 *Alkalmazott fejlesztői eszközök*

Programozási nyelv:

- Java

Fejlesztői környezet:

- Android Studio *2020.3.1 Patch 4*

Emulátor:

- Pixel 4 API 30

Külső könyvtárak:

- Gson 2.9.0
- Google Maps Platform *18.0.2*

Adatbázis-kezelő rendszer:

- MySQL

Szerverek:

- Xampp *v3.3.0*
    - Apache
    - MySQL

Egyéb programok:

- Microsoft Word
- Microsoft PowerPoint
- MySQL Workbench
- Balsamiq
- Microsoft Paint

2.4.3 *Sima osztályok*

Response osztály

A Response osztály a REST API kérések válaszának tárolására jött létre. Az osztály adatszerkezete:

- responseCode

Tulajdonságok: private, int

Leírás: A válasz státuszkódja.

- content

Tulajdonságok: private, String

Leírás: A válasz szöveges tartalma.

Az osztály adataihoz getter metódusokkal férhetünk hozzá.

RequestHandler osztály

A RequestHandler osztály felelős a REST API kérések küldésére, feldolgozására. Az osztálynak csak statikus metódusai vannak, ezért nem kell példányosítani az osztályt. Az osztály metódusai:

- get

Paraméterek: (String) url

Visszatérési érték: Response

Leírás: Egy GET kérést küld a paraméternek megadott url-re.

- getBearer

Paraméterek: (String) url, (String) token

Visszatérési érték: Response

Leírás: Egy GET kérést küld a paraméternek megadott autentikációhoz kötött url-re, ami a token alapján azonosít.

- post

Paraméterek: (String) url, (String) data

Visszatérési érték: Response

Leírás: Egy POST kérést küld a paraméternek megadott url-re, a szöveges tartalmát JSON formátumban adjuk meg.

- put

Paraméterek: (String) url, (String) data

Visszatérési érték: Response

Leírás: Egy PUT kérést küld a paraméternek megadott url-re, a szöveges tartalmát JSON formátumban adjuk meg.

- delete

Paraméterek: (String) url

Visszatérési érték: Response

Leírás: Egy DELETE kérést küld a paraméternek megadott url-re.

- addRequestBody

Paraméterek: (HttpURLConnection) conn, (String) data

Visszatérési érték: nincs

Leírás: A paraméterként megadott kérésnek a testébe beleteszi a szöveges tartalmat JSON formátumban.

- setupConnection

Visszatérési érték: HttpURLConnection

Paraméterek: (String) url

Leírás: A paraméterként megadott url-re létrehozza a kérést.

- getResponse

Visszatérési érték: Response

Paraméterek: (HttpURLConnection) conn

Leírás: A paraméterként megadott kérést feldolgozza, a válaszba visszaadja a státuszkódot, amennyiben a kérés sikertelen volt eltárolja a hiba szövegét, ha sikeres volt eltárolja a kérés szöveges tartalmát.

Token osztály

A Token osztály a felhasználó bejelentkezése után megkapott token tárolására jött létre. Az osztály adatszerkezete:

- token

Tulajdonságok: private, String

Leírás: A token.

Az osztály adataihoz getter metódusokkal férhetünk hozzá.

User osztály

A User osztály a felhasználók tárolására jött létre. Az osztály adatszerkezete:

- id

Tulajdonságok: private, int

Leírás: A felhasználó azonosítója.

- username

Tulajdonságok: private, String

Leírás: A felhasználó felhasználóneve.

- email

Tulajdonságok: private, String

Leírás: A felhasználó email fiókja.

- permission

Tulajdonságok: private, int

Leírás: A felhasználó osztálya.

- remember_token

Tulajdonságok: private, String

Leírás: Jelenleg nincs funkciója.

Az osztály adataihoz getter metódusokkal férhetünk hozzá.

Location osztály

A Location osztály a helyek tárolására jött létre. Az osztály adatszerkezete:

- id

Tulajdonságok: private, int

Leírás: A hely azonosítója.

- name

Tulajdonságok: private, String

Leírás: A hely neve.

- description

Tulajdonságok: private, String

Leírás: A hely leírása.

- lat

Tulajdonságok: private, double

Leírás: A hely szélességi köre.

- lng

Tulajdonságok: private, double

Leírás: A hely hosszúsági köre.

- allowed

Tulajdonságok: private, boolean

Leírás: Amikor egy új helyet felvesz egy felhasználó, az adminisztrátornak engedélyeznie kell az adott helyet, ha engedélyezte látható lesz a térképen.

- user_id

Tulajdonságok: private, int

Leírás: A helyhez tartozó felhasználó azonosítója.

Az osztály adataihoz getter metódusokkal férhetünk hozzá.

LocationRating osztály

A LocationRating osztály a helyek értékeléseinek tárolására jött létre. Az osztály adatszerkezete:

- id

Tulajdonságok: private, int

Leírás: Az értékelés azonosítója.

- stars

Tulajdonságok: private, int

Leírás: Az értékelés 1-5 skálán való értékelése.

- description

Tulajdonságok: private, String

Leírás: Az értékelés leírása.

- location_id

Tulajdonságok: private, int

Leírás: Az értékeléshez tartozó hely azonosítója.

- user_id

Tulajdonságok: private, int

Leírás: Az értékeléshez tartozó felhasználó azonosítója.

- username

Tulajdonságok: private, String

Leírás: Az értékeléshez tartozó felhasználó felhasználóneve.

Az osztály adataihoz getter metódusokkal férhetünk hozzá.

ErrorMessage osztály

Az ErrorMessage osztály a sikertelen API hívások üzeneteinek tárolására jött létre. Az osztály adatszerkezete:

- message

Tulajdonságok: private, String

Leírás: A sikertelen API hívás üzenete.

Az osztály adataihoz getter metódusokkal férhetünk hozzá.

2.4.4 *Activity osztályok*

MapsActivity

A MapsActivity az applikáció belépőpontja. Az activity adatszerkezete:

- GoogleMap *mMap*

Leírás: A Google Map-hez tartozó eseményeket kezeli.

- SharedPreferences *sharedPreferences*

Leírás: A bejelentkezett felhasználóhoz tartozó tokent tárolja el.

- DrawerLayout *drawerLayoutMaps*

Leírás: A MapsActivity-n elhelyezett drawer menüt kezeli.

- NavigationView *navigationViewMaps*

Leírás: A MapsActivity-n elhelyezett drawer menüben az egyes elemeket kezeli. Publikus és statikus, hogy más activity-ből is el lehessen érni annak érdekében hogy ki- és bejelentkezésnél helyes elemek legyenek a menüben.

- ActionBarDrawerToggle *actionBarDrawerToggle*

Leírás: A MapsActivity-n elhelyezett drawer menünek a hamburger gombját kezeli.

- boolean *newLocation*

Leírás: Alaphelyzetben hamis az értéke, amikor a felhasználó új helyet akar elhelyezni igazra vált és a térképen ki lehet jelölni egy pontot ahova az új hely kerülni fog.

- String *URL*

Leírás: A REST API backend szerverének URL-jét tárolja el. Publikus és statikus, hogy más activity-ből is el lehessen érni annak érdekében hogy bárhonnan lehessen REST API hívást kezdeményezni.

- List<Location> *locationList*

Leírás: A szervertől kapott helyeket tárolja el.

- private class RequestTaskGetLocations

Leírás: Letölti a backend szerverről az összes helyet és a térképen elhelyezi.

RegisterActivity

A RegisterActivity-ben lehet regisztrálni új felhasználót. Ezt az activity-t a MapsActivity drawer menüjéből érheti el egy vendég aki nincs bejelentkezve. Az activity adatszerkezete:

- EditText *editTextUsername, editTextEmail, editTextPassword*

Leírás: A regisztráláshoz szükséges adatok megadását teszik lehetővé.

- String *username, email, password*

Leírás: A regisztráláshoz szükséges adatokat tárolják el.

- Button *buttonRegister*

Leírás: Regisztrál egy új felhasználót a megadott adatokkal.

- Button *buttonCancel*

Leírás: Bezárja az activity-t.

- private class RequestTaskRegister

Leírás: Feltölti a backend szerverre az regisztrált felhasználót. Bezárja az activity-t.

LoginActivity

A LoginActivity-ben lehet bejelentkezni egy felhasználói fiókba. Ezt az activity-t a MapsActivity drawer menüjéből érheti el egy vendég aki nincs bejelentkezve. Az activity adatszerkezete:

- EditText *editTextUsername, editTextPassword*

Leírás: A bejelentkezéshez szükséges adatok megadását teszik lehetővé.

- String *username, password*

Leírás: A bejelentkezéshez szükséges adatokat tárolják el.

- Button *buttonLogin*

Leírás: Bejelentkezik egy felhasználói fiókba a megadott adatokkal.

- Button *buttonCancel*

Leírás: Bezárja az activity-t.

- SharedPreferences *sharedPreferences*

Leírás: A bejelentkezett felhasználóhoz tartozó tokent tárolja el.

- private class RequestTaskLogin

Leírás: Elküldi a backend szerverre az adatokat. Amennyiben a megadott adatok helyesek, visszakap egy tokent a szervertől. Az adott tokennel le lehet kérni a jelenleg bejelentkezett felhasználó adatait. Bezárja az activity-t.

FeedbackActivity

A FeedbackActivity-ben lehet jelezni az adminisztrátornak ha a felhasználó találna hibákat. Ezt az activity-t a MapsActivity drawer menüjéből érheti el a vendég és a bejelentkezett felhasználó is. Az activity adatszerkezete:

- EditText *editTextFeedback*

Leírás: A visszajelzéshez szükséges adatok megadását teszik lehetővé.

- String *comment*

Leírás: A visszajelzéshez szükséges adatokat tárolják el.

- Button *buttonFeedbackSend*

Leírás: Elküldi a visszajelzést.

- Button *buttonFeedbackCancel*

Leírás: Bezárja az activity-t.

- private class RequestTaskFeedback

Leírás: Elküldi a backend szerverre a visszajelzést a megadott adatokkal. Bezárja az activity-t.

LocationActivity

A LocationActivity-ben lehet megnézni egy adott hely adatlapját. Ezt az activity-t a MapsActivity egyik térkép jelzőjére kattintva érheti el a vendég és a bejelentkezett felhasználó is. Az activity adatszerkezete:

- TextView *textViewLocationName, textViewLocationDescription*

Leírás: Kiírják a hely nevét és leírását.

- Int *id*

Leírás: A helyhez tartozó azonosító.

- ListView *listViewRatings*

Leírás: Kilistázza a helyhez tartozó értékeléseket.

- List<LocationRating> *ratingList*

Leírás: A helyhez tartozó értékeléseket tárolja el.

- SharedPreferences *sharedPreferences*

Leírás: A bejelentkezett felhasználóhoz tartozó tokent tárolja el.

- Button *buttonVissza*

Leírás: Bezárja az activity-t.

- Button *buttonRating*

Leírás: Átvisz a RatingActivity-re, ahol értékelést lehet hozzáadni a helyhez.

- private class RatingAdapter

Leírás: A helyhez tartozó értékeléseket kezeli, a ListView-hoz csatolandó.

- private class RequestTaskGetRatings

Leírás: A helyhez tartozó értékeléseket letölti a backend szerverről, majd a RatingAdapter-rel beletölti a ListView-ba.

RatingActivity

A LoginActivity-ben lehet bejelentkezni egy felhasználói fiókba. Ezt az activity-t a LocationActivity-ből érheti el egy bejelentkezett felhasználó. Az activity adatszerkezete:

- EditText *editTextRating*

Leírás: Az értékeléshez szükséges adatok megadását teszik lehetővé.

- String *description, stars, location_id*
- Int *user_id*

Leírás: Az értékeléshez szükséges adatokat tárolják el.

- Spinner *spinnerStars*

Leírás: Egy 1-5 skálán lehet értékelni a helyet.

- Button *buttonRatingSend*

Leírás: Hozzáad egy értékelést a helyhez a megadott adatokkal.

- Button *buttonRatingCancel*

Leírás: Bezárja az activity-t.

- SharedPreferences *sharedPreferences*

Leírás: A bejelentkezett felhasználóhoz tartozó tokent tárolja el.

- private class RequestTaskGetUser

Leírás: A tokent átadva visszakéri a szervertől a bejelentkezett felhasználó adatait.

- private class RequestTaskRating

Leírás: Elküldi a backend szerverre az értékelést a megadott adatokkal. Bezárja az activity-t.

NewLocationActivity

A NewLocationActivity-ben lehet egy új helyet felvenni. Ezt az activity-t a MapsActivity drawer menüjéből érheti el a bejelentkezett felhasználó. Az activity adatszerkezete:

- EditText *editTextNewLocationName, editTextNewLocationDescription*

Leírás: Az helyhez szükséges adatok megadását teszik lehetővé.

- String *name, description, lat, lng*
- Int *id*

Leírás: Az értékeléshez szükséges adatokat tárolják el.

- Button *buttonNewLocationSend*

Leírás: Hozzáad egy új helyet a megadott adatokkal.

- Button *buttonRatingCancel*

Leírás: Bezárja az activity-t.

- SharedPreferences *sharedPreferences*

Leírás: A bejelentkezett felhasználóhoz tartozó tokent tárolja el.

- private class RequestTaskGetUser

Leírás: A tokent átadva visszakéri a szervertől a bejelentkezett felhasználó adatait.

- private class RequestTaskNewLocation

Leírás: Elküldi a backend szerverre a helyet a megadott adatokkal. Bezárja az activity-t.

ProfileActivity

A ProfileActivity-ben lehet a bejelentkezett felhasználó adatait megnézni és módosítani. Ezt az activity-t a MapsActivity drawer menüjéből érheti el a bejelentkezett felhasználó. Az activity adatszerkezete:

- EditText *editTextEditUsername, editTextEditEmail*

Leírás: Az felhasználó adatainak módosítását teszik lehetővé.

- String *username, email3*
- User *user*

Leírás: Az felhasználó adatainak módosításához szükséges adatokat tárolják el.

- boolean *editing*

Leírás: Alaphelyzetben hamis, ez a változó határozza meg hogy jelenleg módosíthatóak a felhasználó adatai vagy sem.

- Button *buttonProfileEdit*

Leírás: Első kattintásra eltünteti a 2 TextView-t és megjeleníti a 2 EditText-et, a buttonProfileCancel gomb megváltozik, az editing igazra vált. Második kattintásra a megadott adatokkal módosítja a felhasználó adatait.

- Button *buttonProfileCancel*

Leírás: Ha az editing hamis, bezárja az activity-t. Ha igaz, visszaállítja az eredeti állapotra az activity-t.

- SharedPreferences *sharedPreferences*

Leírás: A bejelentkezett felhasználóhoz tartozó tokent tárolja el.

- private class RequestTaskGetUser

Leírás: A tokent átadva visszakéri a szervertől a bejelentkezett felhasználó adatait.

- private class RequestTaskEditUser

Leírás: Módosítja a backend szerveren a felhasználó adatait a megadott adatokkal. Bezárja az activity-t.

2.4.5 *Tesztelési dokumentáció*

Regisztráció

Művelet	Bemenet	Elvárt eredmény	Végeredmény
Üres mezők	felhasználónév: - email: - jelszó: -	A felhasználónév minimum 5, maximum 20 karakterből állhat!	A felhasználónév minimum 5, maximum 20 karakterből állhat!
Röviden kitöltött mezők	felhasználónév: test email: test jelszó: test	A felhasználónév minimum 5, maximum 20 karakterből állhat!	A felhasználónév minimum 5, maximum 20 karakterből állhat!
Foglalt felhasználónévvel kitöltött mező	felhasználónév: admin email: test@test.test jelszó: testtest	The username has already been taken.	The username has already been taken.
Helyes adatokkal kitöltött mezők	felhasználónév: test_1 email: test@test.test jelszó: 12345678	Sikeres regisztráció	Sikeres regisztráció

Bejelentkezés

Művelet	Bemenet	Elvárt eredmény	Végeredmény
Üres mezők	felhasználónév: - jelszó: -	A felhasználónév minimum 5, maximum 20 karakterből állhat!	A felhasználónév minimum 5, maximum 20 karakterből állhat!
Röviden kitöltött mezők	felhasználónév: test jelszó: test	A felhasználónév minimum 5, maximum 20 karakterből állhat!	A felhasználónév minimum 5, maximum 20 karakterből állhat!
Helytelen jelszóval kitöltött mező	felhasználónév: test_1 jelszó: testtest	Helytelen felhasználónév vagy jelszó.	Helytelen felhasználónév vagy jelszó.
Helyes adatokkal kitöltött mezők	felhasználónév: test_1 jelszó: 12345678	Sikeres bejelentkezés	Sikeres bejelentkezés

Visszajelzés

Művelet	Bemenet	Elvárt eredmény	Végeredmény
Üres mező	komment: -	A visszajelzés minimum 1, maximum 255 karakterből állhat!	A visszajelzés minimum 1, maximum 255 karakterből állhat!
Helyesen kitöltött mező	komment: Teszt visszajelzés	Sikeres visszajelzés	Sikeres visszajelzés

Felhasználó módosítása

Művelet	Bemenet	Elvárt eredmény	Végeredmény
Üres mezők	felhasználónév: - email: -	A felhasználónév minimum 5, maximum 20 karakterből állhat!	A felhasználónév minimum 5, maximum 20 karakterből állhat!
Röviden kitöltött mezők	felhasználónév: test email: test	A felhasználónév minimum 5, maximum 20 karakterből állhat!	A felhasználónév minimum 5, maximum 20 karakterből állhat!
Foglalt felhasználónévvel kitöltött mező	felhasználónév: admin email: test@test.test	The username has already been taken.	The username has already been taken.
Helyes adatokkal kitöltött mezők	felhasználónév: test_2 jelszó: 12345678	Sikeres módosítás	Sikeres módosítás

2.4.6 *Továbbfejlesztési lehetőségek*

A DogGo alkalmazás arra a célra lett megalkotva, hogy a gazdik több helyre tudják elvinni kiskedvenceiket, lehetőséget adva a szocializálására más állatokkal vagy gazdikkal.

Eddig megvalósítottuk az alkalmazás azon részét, amely lehetőséget a gazdinak kiválasztani a legjobb helyet, ahol eltöltheti idejét a kisállatukkal. Mi azonban egy közösséget is szeretnénk építeni ezen felül, hogy a gazdik kommunikálhassanak egymással, megtervezzék közös kirándulásaikat, különféle eseményeket hozhassanak létre.

Ezen szempontok alapján fogok bemutatni pár továbbfejlesztési lehetőséget, ötletet:

Események

Az események lehetőséget adnak gazdiknak, hogy megismerkedjenek új emberekkel, új kapcsolatokat alakítsanak ki más gazdikkal egy adott időpontban.

Az menüben egy események nevű menüpontot lehetne, ahol eseményeket lehet hozzáadni vagy keresni. Az eseményre a felhasználó be tud jelentkezni, ez lehetőséget ad arra, hogy lássuk hány ember lesz jelen az adott eseményen.

Barátok

A barát rendszer létrehozása lehetőséget ad gazdiknak, hogy kommunikálhassanak más gazdikkal és nyomon követhessük az általunk megismert új ismerősöket, például milyen eseményekre vannak ők bejelentkezve.

A menüben létre lehetne hozni egy barátok menüpontot, amely kilistázza a felhasználó hozzáadott barátait. Hozzáadni új barátot például egy saját kóddal lehetne, amit a felhasználó oszt meg az új ismerőseivel. A barátok egy chat ablakban kommunikálhatnának egymással.

Képfeltöltés

Képfeltöltéssel a felhasználók megoszthatnák kepéiket az adott hely vagy esemény oldalán, hogy mások is láthassák milyen élményekben volt részük.

Minden esemény és hely adatlapján lehetne készíteni egy képnézegető felületet, ahova fel tudjuk tölteni a képeinket. A képeket egy külső fájlszerverre lehetne feltölteni, amire az adatbázisban csak egy linkkel hivatkozunk.

3 Felhasználói dokumentáció

3.1 A program általános specifikációja

A DogGo mobil alkalmazás az átlagos felhasználónak lett kifejlesztve. A Google Map által nyújtott térképen helyeket lehet nézni vagy megjelölni, ahova kisállatainkat elvihetjük.

Az applikációban be tudunk regisztrálni és be tudunk lépni a saját fiókunkba, de lehet az alkalmazást regisztráció nélkül használni vendégként. Ennek annyi a hátránya, hogy a vendég nem tud helyeket megjelölni és a már létező helyeket értékelni.

A felhasználók megtekinthetik saját adatlapjukat, és lehetőségük van megváltoztatni adataikat. Ha valami hibát találnak a felhasználók, azt név nélkül tudják jelezni az adminisztrátornak a visszajelzés gombbal.

3.2 Rendszerkövetelmények

- Ajánlott operációs rendszer: Windows 10
- Android Studio: 2020.3.1 Patch 4 vagy újabb
- Emulátor: Minimum 24 API szint
- Ajánlott RAM: 8 GB vagy több

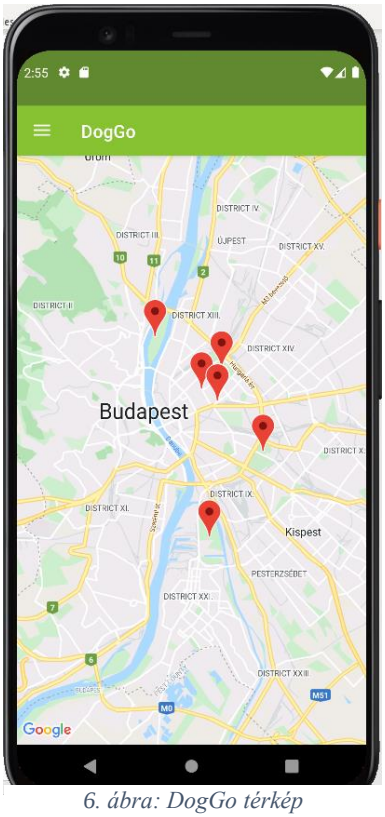
3.3 A program telepítése

A GitHub-ról letöltött forráskódot nyissuk meg az Android Stúdióval. Indítsunk el egy emulátort, ami futtatja majd az applikációt. Hogyha ez kész, futtassuk az applikációt.

Figyelem: Az applikáció csak akkor működik megfelelően, ha a helyi számítógépen fut mellette a backend szerver.

3.4 A program használatának részletes leírása

Amikor elindítjuk az alkalmazást, egy Google térképet látunk magunk előtt. Ezen a térképen meg vannak jelölve helyek, ahova elvihetjük kisállatainkat kirándulni, esetleg találkozni más kisállatokkal. Ezenkívül az applikáció tetején látunk egy gombot, amit, ha megnyomunk, előjön egy menü.



6. ábra: DogGo térkép

Regisztráció

Ha a menüben rákattintunk a regisztráció gombra, átkerülünk a regisztrációs ablakba. Itt három adatot kell megadnunk, hogy új fiókot hozhassunk létre: Felhasználónév, Email, Jelszó. Vannak szabályok, amiket be kell tartani:

- Felhasználónév: Minimum 5, maximum 20 karakterből állhat, és egyedinek kell lennie.
- Email: Minimum 5, maximum 255 karakterből állhat, és egyedinek kell lennie.
- Jelszó: Minimum 8 karakterből kell állnia.

Miután a mezőkbe beleírtuk az adatainkat, kattintsunk rá a regisztráció gombra. Ha helyesen adtuk meg az adatokat, visszakerülünk a térképre a sikeres regisztráció után. Ha hibásan adtuk meg az adatainkat, a program kiír egy hibaüzenetet. Ha mégse szeretnénk regisztrálni, kattintsunk a mégse gombra, ez visszavisz a térképre.

Bejelentkezés

Ha a menüben rákattintunk a bejelentkezés gombra, átkerülünk a bejelentkezés ablakba. Itt már csak kettő adatot kell megadni: Felhasználónév, Jelszó. A szabályok ugyanazok mint a regisztrációnál.

Miután a mezőket kitöltöttük az adatokkal, kattintsunk rá a bejelentkezés gombra. Ha helyesen adtuk meg az adatokat, visszakerülünk a térképre, de már bejelentkezve. Ha hibásan adtuk meg az adatainkat, a program kiír egy hibaüzenetet. Ha mégse szeretnénk bejelentkezni, kattintsunk a mégse gombra, ez visszavisz a térképre.

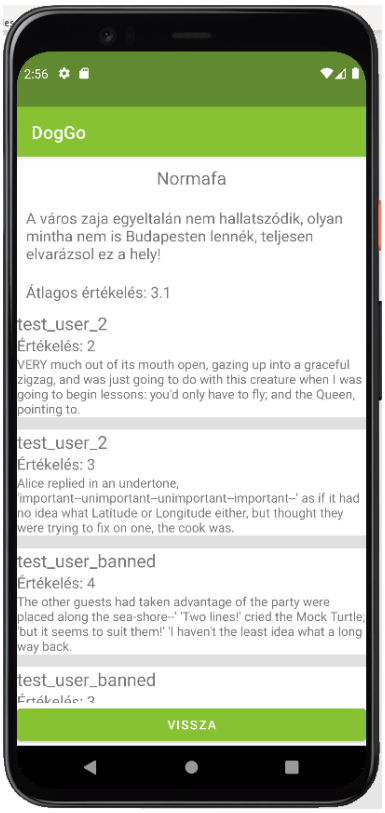
Hely megtekintése

Ha a térképen rákattintunk egy helyre, átkerülünk a hely adatlapjára. Itt láthatjuk az kiválasztott hely adatait, és a helyhez tartozó értékeléseket. A vissza gombra kattintva visszakerülünk a térképre.

Hely értékelése

Ha egy hely adatlapján vagyunk és be vagyunk jelentkezve, értékelést tudunk hozzáadni a helyhez. Ehhez kattintsunk rá az értékelés gombra. Átkerülünk az értékelés ablakba. Itt kettő adatot adhatunk meg: komment és egy 1-5 skálán való értékelés. Kommentet nem feltétlenül kell megadni, maximum 255 karakter hosszú lehet.

Ha kész vagyunk, kattintsunk az értékelés gombra. Ha mégse szeretnénk értékelni, kattintsunk a mégse gombra, hogy visszakerüljünk a hely adatlapjára.



7. ábra: DogGo hely adatlapja

Visszajelzés

Ha hibát találnánk az applikáció futása közben, ezt jelezni tudjuk az adminisztrátornak.

Ha a menüben rákattintunk a visszajelzés gombra, átkerülünk a visszajelzés ablakba. Itt megadhatunk egy kommentet, amiben leírhatjuk az észrevételünket. A komment maximum 255 karakterből állhat.

Miután a mezőt kitöltöttük, kattintsunk rá a visszajelzés küldése gombra. Ha helyesen adtuk meg az adatot, visszakerülünk a térképre sikeres visszajelzés után. Ha hibásan adtuk meg az adatot, a program kiír egy hibaüzenetet. Ha mégse szeretnénk visszajelzést küldeni, kattintsunk a mégse gombra, ez visszavisz a térképre.

Adatlapom

Ha a menüben rákattintunk az adatlapom menüpontra, átkerülünk a bejelentkezett felhasználó adatlapjára. Itt láthatjuk a felhasználó felhasználónevét és email fiókját. Ezeket módosítani is tudjuk hogyha a módosít gombra kattintunk, miután átírtuk az adatokat kattintsunk a mentésre. Ha mégse szeretnénk módosítani, kattintsunk a mégse gombra. A vissza gombra kattintva visszakerülünk a térképre.

8. ábra: DogGo felhasználó adatlapja

Hely hozzáadása

Ha a menüben rákattintunk a hely hozzáadása menüpontra, utána ki kell választanunk egy helyet a térképen. Miután rákattintottunk a kívánt helyre a térképen, átkerülünk a hely hozzáadása ablakba. Itt két adatot tudunk megadni: hely neve és hely leírása. A hely neve minimum 5, maximum 40 karakterből állhat. A leírás opcionális, maximum 255 karakterből állhat.

Miután megadtuk az adatokat, kattintsunk a hely hozzáadása gombra. Ha jól adtuk meg az adatokat, akkor sikeres volt a hozzáadás. Ezt az új helyet nem fogjuk a térképen egyből látni, mivel az adminisztrátornak előbb engedélyeznie kell. Ha mégse szeretnénk helyet hozzáadni, kattintsunk a mégse gombra, ezzel visszakerülünk a térképre.

4 Összegzés

Az év elején, amikor eldöntöttük hogy ezt az alkalmazást fogjuk elkészíteni, még nem tudtam mennyire nehéz is lesz. Azt se tudtam hogyan kezdjem el a projektet, de szerencsére nem sokkal később megtanultuk hogyan kell a backendet létrehozni. Azóta már megtanultam hogyan kell egy REST API alkalmazást csinálni. Nem volt könnyű, sokat tanultam ahhoz, hogy sikerüljön. Sajnos idő hiányában számos funkció nem került bele az alkalmazásba, amit megbeszéltünk a csapattársaimmal.

A csapattársaimmal nagyon szorgalmasok voltunk, megtanultam hogyan kell csapatban dolgozni. Élvezetes volt sokszor összeülni és megbeszélni a projektet, hogy ki hogy halad, milyen funkciók kellenek, stb.

Annak is nagyon örülök, hogy megtanulhattam számos keretrendszert és programozási nyelvet, amivel előrébb jutottam a programozás területén. Továbbra is tanulni fogok, hogy minél jobb programokat tudjak fejleszteni.

4.1 Summary

At the start of the school year, me and my friends banded together to create a really cool app, that would allow pet owners to socialize with each other more easily. At the time we didn’t know how difficult it would be to develop such a program.

Our first obstacle was that we didn’t even know how to start. Weeks later, we were finally able to start developing the backend of our application. We planned out the details of the database and the user experience. Once we built up the backend, we went our separate ways to develop the frontend.

I was nervous, because I had never developed a real mobile app before, so I had to research a ton about it. After a while, I started developing the app for real, but I had to start over numerous times thanks to some annoying bugs in the code. Although there were a lot of problems, I finally completed the app.

While I was developing the app, unfortunately, I forgot about the documentation, so I had to write it in the last days before the deadline. There were some features that I couldn’t implement in the app, so I still need to learn a lot.

Overall, I am really glad that I completed this year because I learned a lot about programming languages, frameworks, and most importantly, teamwork. It was really enjoyable working with my friends on this project, and I’ll most likely never forget it.

5 Források

- https://laravel.com/docs/9.x

  Utolsó megnyitás: 2022.04.19

- https://stackoverflow.com/

  Utolsó megnyitás: 2022.04.19

- https://developers.google.com/maps/documentation/android-sdk/start

  Utolsó megnyitás: 2022.04.19

- https://www.geeksforgeeks.org/

  Utolsó megnyitás: 2022.04.19

- https://balsamiq.com/

  Utolsó megnyitás: 2022.04.19

- https://developer.android.com/

  Utolsó megnyitás: 2022.04.19

- https://www.flaticon.com/

  Utolsó megnyitás: 2022.04.20

6 Ábrajegyzék

1. ábra: DogGo ütemterv 4
2. ábra: Használati eset diagram 8
3. ábra: Adatbázis terv 9
4. ábra: WireFrame térkép 25
5. ábra: WireFrame hely adatlapja 25
6. ábra: DogGo térkép 41
7. ábra: DogGo hely adatlapja 42
8. ábra: DogGo felhasználó adatlapja 43