Budapesti Műszaki Szakképzési Centrum

Petrik Lajos Két Tanítási Nyelvű Technikum

SZOFTVERFEJLESZTŐ ÉS TESZTELŐ TECHNIKUS SZAKMA

DogGo

**Készítette:** Kilián Marcell András, Korcsmáros Kristóf György, Takács Balázs Levente

Budapest, 2021.

Tartalomjegyzék

1. [DogGo 1](#_Toc95905892)
2. [Tartalomjegyzék 2](#_Toc95905893)
3. [Plágium nyilatkozat 3](#_Toc95905894)
4. [Bevezetés 4](#_Toc95905895)
   1. [Téma 5](#_Toc95905896)
   2. [Témaválasztás indoklása 6](#_Toc95905897)
5. [Fejlesztői dokumentáció 7](#_Toc95905898)
   1. [Funkciók 7](#_Toc95905899)
   2. [Backend 8](#_Toc95905900)
      1. [Adatbázis terv 8](#_Toc95905901)
      2. [Adattáblák (doggodb) 9](#_Toc95905902)
      3. [Backend telepítésének lépései 11](#_Toc95905903)
      4. [API végpontok 12](#_Toc95905904)
   3. [Használati eset diagram 24](#_Toc95905905)
6. [Ábrajegyzék 25](#_Toc95905906)

Plágium nyilatkozat

Alulírott Kilián Marcell András, Korcsmáros Kristóf György, Takács Balázs Levente kijelentjük, hogy ez a vizsgaremek saját tudásunk, önálló munkánk terméke.

A vizsgaremek közös részeit pontosan jelöltük.

Budapest, 2021.

....................................

Kilián Marcell   
András

....................................

Korcsmáros Kristóf György

....................................

Takács Balázs   
Levente

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.

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.

Témaválasztás indoklása

Manapság a háztartások nagy részében óriási 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.

Egyik megoldás lenne, hogy mi felsorolunk híres, számunkra izgalmas, jó helyeket, de ez egyéni vélemény és nem feltétlenül tetszene az emberek nagy részének. Lehet, hogy sok embernek tettszenének azok a helyek, amiket mi mutatnánk, de minél több ember mutat számára kutyasétáltatásra alkalmas helyeket, annál több ember fog számára megfelelő helyet találni, ahova ő is elmenne kiskedvencét sétáltatni.

Emiatt jutottunk arra a megoldásra, hogy ha ezeknek az embereknek különböző helyeket kínálnánk, hogy hova is mehetnének kiskedvenceikkel sétálni, motiválnánk őket, hogy mozduljanak ki. Ha sikerülne kialakítani közösségeket, akkor egymást is ösztönöznék egy tartalmas sétára.

A gondolat a fejlesztői csapat összes tagjának tetszett, a lelkesedést látva, ezt a témát rögtön megválasztottuk.

Fejlesztői dokumentáció

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 | ❌ | ❌ | ✔ |

Backend

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.



. ábra: Adatbázis terv

Adattáblák (doggodb)

users tábla

|  |  |  |  |
| --- | --- | --- | --- |
| **Oszlopnév** | Típus | Hossz | Tartalma |
| **id** | BIGINT, PK, AI | 20 | Elsődleges kulcs |
| **username** | VARCHAR | 255 | Felhasználó felhasználóneve |
| **email** | VARCHAR | 255 | Felhasználó E-mail címe |
| **password** | VARCHAR | 255 | Felhasználó jelszava |
| **permission** | TINYINT | 4 | Milyen jogosultsággal rendelkezik a felhasználó:  0 – default  1 – tiltva  2 – admin  3 – super admin |
| **remember\_token** | VARCHAT | 100 |  |
| **created\_at** | TIMESTAMP |  | Létrehozás dátuma |
| **updated\_at** | TIMESTAMP |  | Módosítás dátuma |

locations tábla

|  |  |  |  |
| --- | --- | --- | --- |
| **Oszlopnév** | Típus | Hossz | Tartalom |
| **id** | BIGINT, PK, AI | 20 | Eldősleges kulcs |
| **name** | VARCHAR | 255 | Hely neve |
| **lat** | DOUBLE |  | Hely koordinátája (szélesség) |
| **lng** | DOUBLE |  | Hely koordinátája (hosszúság) |
| **user\_id** | BIGINT, FK |  | Hivatkozás a users táblára |
| **created\_at** | TIMESTAMP |  | Létrehozás dátuma |
| **updated\_at** | TIMESTAMP |  | Módosítás dátuma |

ratings tábla

|  |  |  |  |
| --- | --- | --- | --- |
| **Oszlopnév** | Típus | Hossz | Tartalom |
| **id** | BIGINT, PK, AI | 20 | Elsődleges kulcs |
| **stars** | INT | 11 | 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 |
| **created\_at** | TIMESTAMP |  | Létrehozás dátuma |
| **updated\_at** | TIMESTAMP |  | Módosítás dátuma |

feedback tábla

|  |  |  |  |
| --- | --- | --- | --- |
| **Oszlopnév** | Típus | Hossz | Tartalom |
| **id** | BIGINT, PK, AI | 20 | Elsődleges kulcs |
| **comment** | VARCHAR | 255 | Visszajelzés szövege |
| **created\_at** | TIMESTAMP |  | Létrehozás dátuma |
| **updated\_at** | TIMESTAMP |  | Módosítás 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>

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."

}

**feedback végpont csoport**

**GET api/feedbacks**

Visszaadja a visszajelzések adatait.

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

[

{

"id": 1,

"comment": "Sed unde tempora vero nam fugiat facere."

},

{

"id": 2,

"comment": "Dolorum numquam asperiores quod dolore ipsa similique eos quia."

}

]

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

Visszaadja az adott ID-val rendelkező visszajelzés adatait.

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

{

"id": 5,

"comment": "Omnis harum cumque nesciunt sed ex facilis. Non non ut nisi."

}

**POST api/feedbacks**

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

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

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

Bemenet:

{

"comment": "test"

}

Eredmény:

{

"comment": "test",

"id": 16

}

**PUT api/feedbacks/{id}**

Módosítja az adott ID-val rendelkező visszajelzés adatait.

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

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

Bemenet:

{

"comment": "test"

}

Eredmény:

{

"id": 5,

"comment": "test"

}

**DELETE api/feedbacks/{id}**

Kitörli az adatbázisból az adott ID-val rendelkező visszajelzést.

Például: DELETE <http://localhost:8000/api/feedbacks/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.

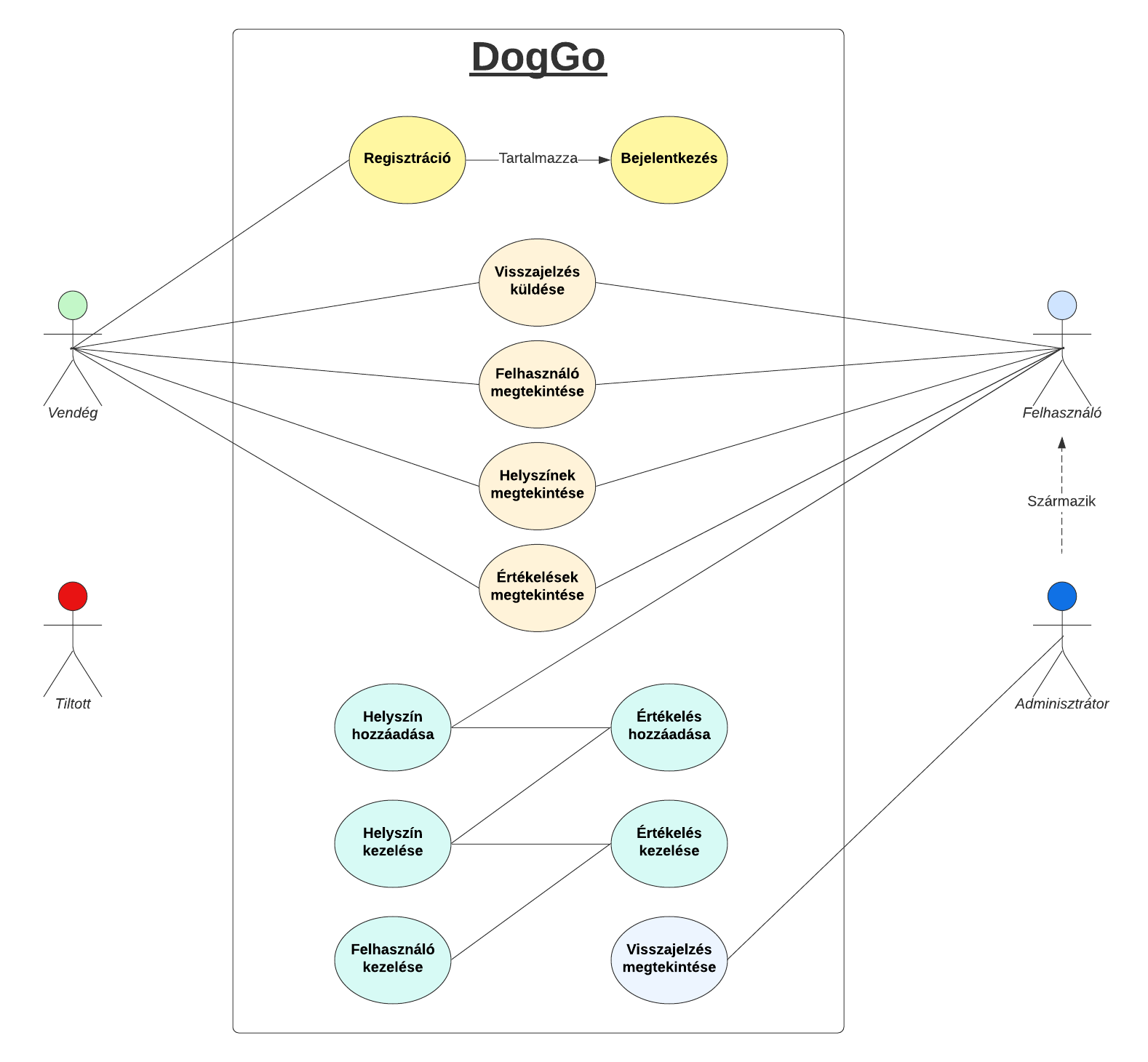
Például: GET <http://localhost:8000/api/feedbacks/9999> (nem létező id)

Status: 404 Not Found

{

"message": "A megadott azonosítóval nem található visszajelzés."

}

Használati eset diagram

2. ábra: Használati eset diagram

Mobil

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.

Text

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

Text, letter

Description automatically generatedA felhasználó meg tudja nézni a saját adatlapját, valamint szerkeszteni is képes az adatait.

Alkalmazott fejlesztői eszközök

Programozási nyelv:

* Java

Fejlesztői környezet:

* Android Studio

Külső könyvtárak:

* Gson
* Google Maps Platform

Adatbázis-kezelő rendszer:

* MySQL

Szerverek:

* Xampp
* Apache
* MySQL

Egyéb programok:

* Microsoft Word
* Microsoft PowerPoint
* MySQL Workbench
* Balsamiq
* Microsoft Paint

Response osztály

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

**responseCode**

*Tulajdonságok:* privát, int

*Leírás:* A válasznak a státuszkódját tárolja el.

**content**

*Tulajdonságok:* privát, String

*Leírás:* A válasznak a szöveges tartalmát tárolja el.

**Az osztály adataihoz getter és setter 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 a következőek:

**get**

*Visszatérési érték:* Response

*Paraméterek:* (String) url

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

**getBearer**

*Visszatérési érték:* Response

*Paraméterek:* (String) url, (String) token

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

**post**

*Visszatérési érték:* Response

*Paraméterek:* (String) url, (String) data

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

**put**

*Visszatérési érték:* Response

*Paraméterek:* (String) url, (String) data

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

**delete**

*Visszatérési érték:* Response

*Paraméterek:* (String) url

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

**addRequestBody**

*Visszatérési érték:* nincs

*Paraméterek:* (HttpURLConnection) conn, (String) data

*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.

Továbbfejlesztési lehetőségek

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

Eddig megvalósitottuk 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 gazdig 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 felhasználő 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 képeikat 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.

Ábrajegyzék

[1. ábra: Adatbázis terv 8](#_Toc95905845)

[2. ábra: Használati eset diagram 24](file:///C:\ma\DogGo\Dokumentacio\DogGo-dokumentacio.docx#_Toc95905846)