ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Alba Regia Műszaki Kar

Geoinformatikai Intézet

TUDOMÁNYOS DIÁKKÖRI DOLGOZAT

ORSZÁGOS MÉHÉSZETI TÉRINFORMATIKAI TÁMOGATÓ RENDSZER FEJLESZTÉSE

Szerző: **Böröcz Balázs**

Földmérő mérnök BSc. szak, IV. évf.

Konzulens: **Dr. habil. Molnár Gábor Péter**

egyetemi docens

Székesfehérvár, 2022.

Tartalomjegyzék

Tartalmi kivonat 3

1. Bevezetés, probléma bemutatása 4

1.1. Előszó 4

1.2. Elvárások a rendszerrel szemben 5

2. Adatok, adatstruktúra 6

2.1. Adatok forrása a fejlesztéshez 6

2.2. Adastruktúra 7

2.3. Adatok az éles rendszerben 9

3. A rendszer: fejlesztése, működése 11

3.1. A webtérkép 11

3.1.1. Az OpenLayers felépítése 11

3.1.2. A rendszerben alkalmazott térképi interakció 14

3.1.3. A geolokáció és Marker elhelyezése 17

3.2. Térkép szerepe a rendszerben 19

3.3. A Backendről röviden 21

3.4. Az alkalmazás 21

3.4.1. A dizájn 21

3.4.2. Applikáció fejlesztés 25

4. Összegzés 26

Irodalomjegyzék 27

Köszönetnyilvánítás 29

Tartalmi kivonat

A TDK dolgozatomban bemutatom az általam megtervezett térinformatikai rendszer fejlesztését. A rendszer a méhészeti adminisztrációk digitalizálását szeretné elősegíteni, továbbá, hogy a méhész számára megkönnyítse vándorhely bejelentés folyamatát. A rendszer képes lesz okostelefon segítségével GPS koordinátákat mérni, és a korábban személyre szabott állományt helyadatokkal ellátva, egy szerveren tárolni.

A program teszteléséhez jelenleg fiktív adatokat használok, mert idén már nem lesz alkalom valós vándoradatok regisztrálására, de a jövőben, méhészek bevonásával, szeretném élesben kipróbálni és tovább fejleszteni a rendszert.

A rendszer rendelkezni fog digitális webtérképpel, amely információval látja el a méhészeket, ezáltal megkönnyítve a vándorlás folyamatát, valamint a méhlegelők túlterhelésének csökkentéséhez is hozzájárul. A térkép JavaScript© környezetben kerül fejlesztésre, amelyhez egy térképspecifikus nyílt forráskódú könyvtárat alkalmazok, az OpenLayers©-t. Az adatok PostgreSQL© adatbázisban tárolódnak, egy téradat orientált kiegészítő, a PostGIS© segítségével, amely a térinformatikai alapú lekérdezést biztosítani fogja.

A dolgozat bemutatja továbbá a letisztult felhasználói felületet, valamint a Backend és Frontend kommunikációját.

1. Bevezetés, probléma bemutatása

1.1. Előszó

Ez Európai méztermelés egyik legjelentősebb képviselője a magyar méhészet [1]. Hazánkban jelentős része a méhészetnek vándorméhészettel foglalkozik, amely azt jelenti, hogy a méztermelők a virágzást követve vándorolnak állományaikkal virágzásról virágzásra. Ezzel együtt természetesen adminisztrációs teher is nehezedik a termelők vállára, ugyanis minden egyes vándorhely bejelentési, és kijelentési kötelezettséget is hordoz maga után [2]. Ezek a bejelentések plusz időráfordítást igényelnek a méhészek számára, ugyanis a legtöbb helyen személyesen kell megtenni ezt a bejelentést egy bejelentő lap segítségével. Néhol ez egyszerűbben is megoldható email segítségével, azonban ott sem egyszerű a bejelentés, mivel ez elsősorban helyrajzi szám feltüntetésével együtt lehetséges, ami egyfelől nehézséget jelent, mivel ki kell keresni a helyrajzi számot, másfelől pedig pontatlan is, ugyanis egy helyrajzi szám akár nagyobb területet is lefedhet, ezáltal adott helyzetben akár nem megfelelő helyzetképet kapunk arról, hogy pontosan hol helyezkednek el a méhészeti állományok.

Jelentős nehézség a méhészek számára továbbá, az ideális vándorhelyek keresése. Több szempontot is figyelembe kell venni egy vándorhely kiválasztásánál, ilyen a megközelíthetőség, a megfelelő minőségű virágzás, a méhlegelő túlterheltsége, telephelytől való távolsága, terepviszonyok valamint, hogy mennyire biztonságos a környezet. Ezen adatok egy részéről a méhészek ismeretekkel is rendelkeznek, viszont némely adatra, mint például a túlterheltségre csak következtethetni lehet, azáltal, hogy a többi méhész, korábban miként helyezett el foglaló táblákat, azonban ez nem mindig tükrözi a valóságot.

A méztermelést erősen veszélyeztetik a méhek között előforduló járványos betegségek, amely az állományok pusztulásával járnak. A megbetegedések figyelésével és a szükséges intézkedések megtételével a méhegészségügy, a méhegészségügyi felelősökön keresztül foglalkozik, akik munkájuk során olyan kihívásokkal és problémákkal szembesülnek, mint például a nem bejelentett hobbi méhészetek, melyek nagyon sok esetben, a nem megfelelő, vagy a kezelés teljes hiánya miatt, potenciális fertőzési gócpontokat alakítanak ki. A méhek kötelező bejelentését jogszabály írja elő, melynek egyik legfontosabb célja, hogy a hatósági állatorvost, valamint a méhegészségügyi felelőst tájékoztassa a közelben elhelyezett méhállományokról.

A felmerülő problémákkal gyerekkorom óta sikerült közelebbről megismerkednem mivel szüleim családi vállalkozás szintjén foglalkoznak méhészettel. Az általam kitalált, és megtervezett térinformatikai rendszer, egy web alapú, egyszerű felhasználói felülettel rendelkező adatbázis, amelyhez az adatokat, maguk a méhészek fogják szolgálni. A dolgozatban bemutatásra kerül a rendszer fejlesztése, azonban az egyes informatikai részek fejlesztésénél (a kellő mélységű informatikai háttértudás hiányában) segítségemre volt Gábor Dániel webfejlesztő informatikus.

1.2. Elvárások a rendszerrel szemben

A rendszernek rendelkeznie kell olyan térképpel, amely képes lekérdezés továbbítására az adatbázis felé, oly módon, hogy az egyszerű felhasználónak (méhészek) csak olyan adatokat szolgáljon melyhez jogosultsága van hozzá férni. Ez a gyakorlatban azt jelenti, hogy a méhész, a térképen látja a saját méhállományainak a helyzetét, azonban a többi méhészet pontos helyzetét nem lesz képes látni, viszont a térképről, kattintás útján, statisztikai információkat szerezhet.

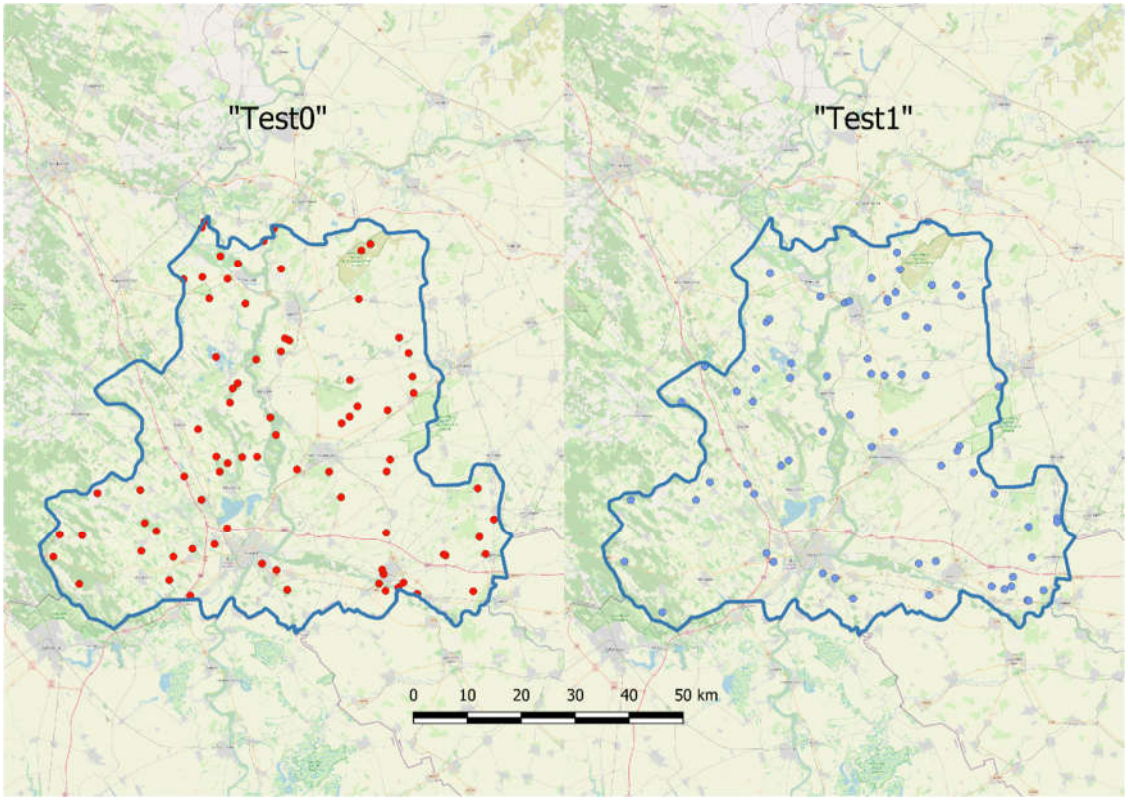
Fontos részt képez továbbá, hogy a méhész képes legyen a rendszerhez pár gombnyomással bejelenteni a méhállományait (egyelőre csak szimulációs szinten), ezáltal a bürokratikus terhei nagy mértékben csökkennének, valamint a szakmaközi szervezet, az Országos Magyar Méhészeti Egyesület (továbbiakban OMME) , sokkal átláthatóbb képet kapna a hazai méhészet helyzetéről, állapotáról. Fontos továbbá megjegyezni az állategészségügyi helyzet javulásának lehetőségeit, melyek a rendszer egyik legfontosabb alapkövének számítanának, a javulás pedig a rendszer sokkal átláthatóbb információ biztosításának lenne köszönhető, amellyel hatékonyabban lehetne védekezni a méhészeteket veszélyeztető fertőzésekkel szemben.

2. Adatok, adatstruktúra

2.1. Adatok forrása a fejlesztéshez

Az adatok jelenleg, amellyel elkészült a projekt és a tesztelés is zajlott, fiktív, kitalált adatok, melyek létrehozásánál szempont volt, hogy a könnyebb tesztelhetőség érdekében Csongrád-Csanád megyében helyezkedjenek el. Rendelkeznie kellett továbbá térbeli geometriával (a mi esetünkben pont), és különböző eltárolt attribútumokkal, ilyen például a méhcsalád szám, a bejelentés időpontja, valamint egy ID (azonosító szám) amellyel össze lehetett kapcsolni az összetartozó állományokat.

A pontokat QGIS-ben generáltattam le, és hozzá rendeltem az attribútumokat, melyeket a PostGIS nyújtotta lehetőségekkel [3] feltöltöttem PostgreSQL adatbázisba. Két csoportban helyeztem el a pontokat melyeknek a „Test0” és „Test1” nevet adtam. (*2-1 ábra*). A „Test0” elnevezésű csoportban 80 db pont helyezkedett el, és mind ugyanazzal a kiinduló dátummal volt ellátva (2022.03.01.). Ezt a dátumot tekintettem a ’0’ időpontnak. Ezután került létrehozásra a „Test1” elevezésű csoport amelyben már csak 70 db pont került elhelyezésre és mindegyikük ’ID’-je megtalálható volt a „Test0”-ban, de a dátumuk mind különböző volt.



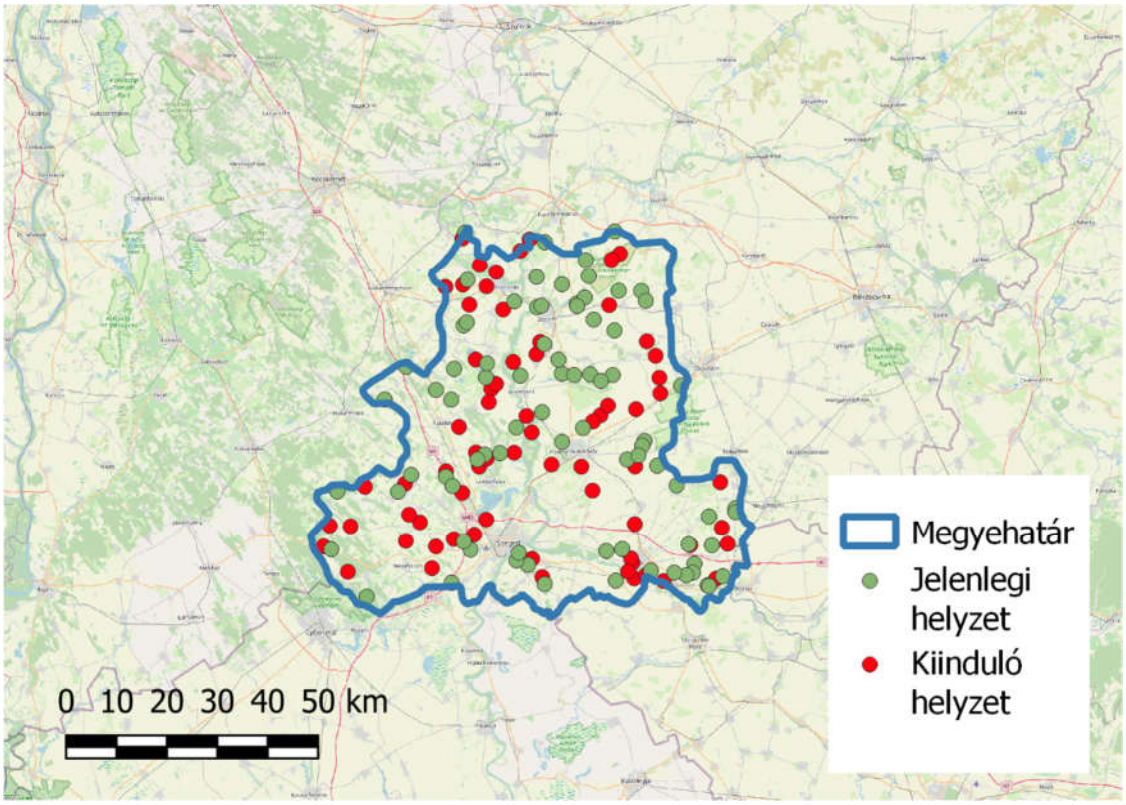
2-1. ábra: a "Test0"csoport összehasonlítása a "Test1" csoporttal

A „Test1” csoport szimulálja, hogy a „Test0”-ban elhelyezkedő pontok, 10 pont kivételével, helyzetet változtattak. A következő lépésként a két csoportot unióban egyesítettem, és ez a csoport kapta az „un” elnevezést, majd PostgreSQL-ben írtam hozzá, egy olyan kódot (*2-2. ábra*), amellyel leválogatta, mindegyik ’ID’-hez tartozó legfrissebb helyzetet, az így készült csoport a „latest” elnevezést kapta az adatbázisban (*2-3. ábra*).

Query Query History

```
1  SELECT Distinct on ("b_id") id, b_id, geom, date
2  into table latest
3  from un
4  order by "b_id","date" DESC nulls LAST, "b_id"
5
```

2-2. ábra: PostgreSQL kód a legfrissebb adatok leválogatására



2-3. ábra: A kiinduló helyzet az úgynevezett "un" csoport, valamint a "latest" csoport a legfrissebb helyzet ábrázolására

2.2. Adastruktúra

Az adatbázis fejlesztése közben három táblában helyeztük (*2-4. ábra*) el az adatainkat, melyek kapcsolatban állnak egymással. Az első tábla a „locations” (tehát helyszínek) nevet

kapta. Ez a tábla az egyik központi eleme a rendszernek, ugyanis minden egyes betáplált adat (bejelentett méhállomány) ebben a táblában kap helyet. A tábla a következő elemeket tartalmazza:

- ’id’ (egyedi azonosító): minden egyes újabb sor kap egy új egyedi azonosítót, annak érdekében, hogy az adatok keveredését megelőzzük.
- ’geom’ (geometria): speciális adatforma, melyet a PostGIS kiegészítő tesz lehetővé, tárolja a geometriai leírást (a mi esetünkben pont), valamint a térbeli helyzethez tartozó adatokat is. Külön érdekessége, hogy a kód amiben tárol vetületi rendszer független [3].
- ’start_date’ (kezdési időpont): a rendszerbe bejelentés dátuma
- ’bee_stock_id’ (méhállomány azonosító): a bejelentett méhállomány egyedi azonosítója, lehetővé teszi, a ’bee_stocks’ nevezetű táblával való összekapcsolást
- ’end_date’ (vég dátum): a rendszer automatikusan generálja, amikor ugyanazzal a ’bee_stock_id’-val rendelkező állomány új helyre bejelentkezik. Az újonnan kialakult ’end_date’ meg fog egyezni az előző helyzet ’start_date’ adatával.
- ’status’ (státusz): tartalmazza az adott helyadat, jelen pillanatban lévő státuszát. Amennyiben a méhész, véletlen, vagy rossz bejelentést tett, azt vissza tudja vonni. Ebben az esetben a ’status’ értéke ’CANCELLED’ (visszavont) értéket kap. Ezen felül felveheti a ’VALID’ és a ’RESERVED’ értéket is, előbbi arra vonatkozik, ha valóban helyeztek el ott méhállományt, utóbbi arra ha csak helyfoglalást hajtottak végre.

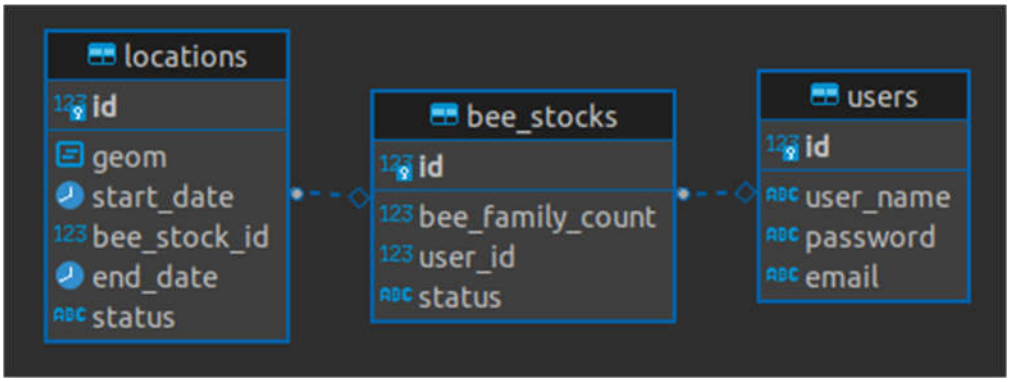
A következő tábla a ’bee_stocks’ (méh állományok). A tábla csak akkor változik, ha új értéket kap (tehát új állomány kerül regisztrálásra). A tábla a következő adatokat tartalmazza:

- ’id’ (egyedi azonosító): minden egyes újabb sor kap egy új egyedi azonosítót, annak érdekében, hogy az adatok keveredését megelőzzük, továbbá erre az azonosítóra hivatkozik a ’locations’ elnevezésű tábla ’bee_stock_id’ elnevezésű mezője.
- ’bee_family_count’ (méhcsalád szám): az adat tárolja, hogy adott állomány mekkora méhcsalád számmal rendelkezik.
- ’user_id’ (felhasználó azonosító): meghivatkozza a ’users’ elnevezésű tábla azon sorát, amelyben ez az egyedi azonosító található.

- ’status’ (státusz): tartalmazza az adott állomány státuszát. Ez az adat két féle értéket vehet fel, lehet ’VALID’, ha az állomány jelenleg is létezik, valamint lehet ’TERMINATED’, ha az állományt megszüntették.

A harmadik tábla a ’users’ (felhasználók) nevet kapta. Ez a tábla tartalmazza a felhasználóhoz párosított adatokat, melyek az alábbiak:

- ’id’ (egyedi azonosító): ennek az adatnak a segítségével kapcsolódik a ’users’ tábla, a ’bee_stocks’ táblához. Minden felhasználó rendelkezik egy saját egyedi azonosítóval.
- ’user_name’ (felhasználónév): a felhasználó belépéséhez, illetve rendszerhez használt felhasználóneve.
- ’password’ (jelszó): a felhasználó belépéséhez szükséges jelszó.
- ’email’: a felhasználó elérhetősége ahol az értesítést kapja



2-4. ábra: Adatstruktúra

2.3. Adatok az éles rendszerben

Éles adatok a pilot üzem során kerülnek majd először a rendszerbe, a 2023-as év során. A pilot üzemhez már előzetesen volt egyeztetés több méhésszel, valamint méhészeti szaktanácsadóval, továbbá előadást tartok decemberben a Gyulai mézesnapokon, ami méhészeti szakmai napok is egyben, így reményeim szerint sikerül elegendő méhészt toborozni a pilot projekthez.

Mint korábban már említésre került, a rendszerhez az éles adatokat, maguk a méhészek fogják szolgáltatni. Ez úgy fog megvalósulni, hogy közben az adatbázis tábláit az alábbi módon töltik fel adatokkal:

1. Regisztráció során a ’users’ táblába kerülnek az általuk betáplált adatok.

2. Regisztrálják az állományokat méhcsaládszámmal ellátva, ezáltal hozzájárulnak ’bee_stocks’ tábla bővüléséhez.
3. A bejelentéskor csak kiválasztják azt állományt amelyiknek az adatait rögzíteni szeretnék, majd mérnek egy GPS-es helyadatot, megnézik a térképen, és amennyiben pontatlannak ítélik meg, manuálisan a térképre kattintva javíthatnak, majd a kívánt adatokat regisztrálják a rendszer felé. Ezáltal a ’locations’ elnevezésű tábla is bővítésre kerül.

3. A rendszer: fejlesztése, működése

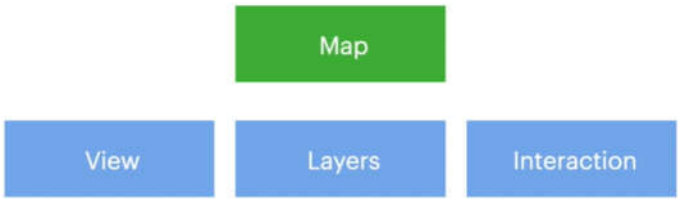
3.1. A webtérkép

A webtérkép fejlesztése HTML, a CSS, és a JavaScript nyelveken történt. A fejlesztést Visual Studio© környezetben végeztem, egy Live Server elnevezésű kiegészítővel kibővítve, amely lehetővé tette, hogy amit leírok kódként, azonnal megjelenjen a weben, így gyakorlatilag élőben lehetett szerkeszteni. A térkép elkészítéséhez két térkép specifikus JavaScript könyvtárral is megpróbálkoztam. Először a Leaflet© könyvtárat használtam, mely fejlesztési szempontokból sokkal könnyebb használatú, azonban az elérhető interakciói korlátoltabbak [4], így végül az OpenLayers© [5] mellett döntöttem. Mind két könyvtár teljesen nyílt forráskódú és felhasználású (open source).

A webtérkép készítésénél, komolyabb HTML és CSS kódra nem volt szükségem, így azokat csak a teljesség kedvéért mutatom be. A HTML kód készítésekor gyakorlatilag az alap kódba csak annyiban kellett belenyúlnom, hogy a fejlesztési környezethez használt könyvtárakat meghívtam, ilyen például az OpenLayers CSS és JavaScript könyvtára [5], előbbi a térkép megjelenítéséhez, valamint a különböző térképi csempék beállításához kellet, utóbbi pedig a térkép fejlesztéséhez játszottak kulcsfontosságú szerepet. Ezen kívül meghívásra került még egy ”Font Awesome” elnevetésű könyvtár [6] amely egy ikon könyvtár, applikáció fejlesztéshez, valamint a saját CSS és JavaScript kódom hivatkozásai.

3.1.1. Az OpenLayers felépítése

A webtérkép három fő elemből ál: ’view’ (nézet), ’layers’ (rétegek), ’interactions’ (interakciók) [5] [7] (*3-1. ábra*).



3-1. ábra: A webtérkép összetétele

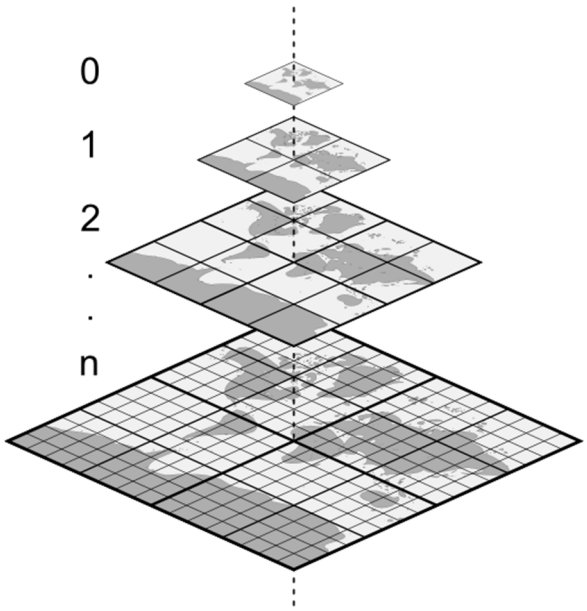
A ’view’ beállításainál három dolog kerül beállításra, a ’center’(az induláskori térkép közepe), a ’zoom level’, valamint a ’projection’ (vetület). A center beállításánál először ’[0,0]’ koordináta párt alkottam, majd miután készen voltam a térkép kódjával és meg tudtam jeleníteni egy weblapon, kikerestem azt a nézetablakot ahogy meg szerettem volna jeleníteni, majd a ’consol’ ablak segítségével lekértem a center koordinátáit, valamint a zoom levelt. A vetület

kiválasztása során több szempontot kellett figyelembe vennem. Az EPSG adatbázis kód [8] beállításával lehet változtatni, ami a European Petroleum Survey Group rövidítését jelenti, és minden vetület rendelkezik egyel (például a Google Maps által használt EPSG szám a Web Mercator vetületet alkalmazza melynek száma 3857, de ha lekérünk koordinátát, akkor a kapott koordináta WGS84 koordinátaként jelenik meg tizedesfokban, melynek száma 4326). Az OpenLayers alapértelmezésben, csak a WGS84, vagy a Web Mercator vetülettel rendelkezik [5] [9], azonban ha az online elérhető proj4 JavaSript könyvtárat is meghívjuk [9], akkor további EPSG számokat is lehet használni [8]. Ezt legkönnyebben az ”epsg.io” webhely segítségével tehetjük, ahol minden proj4 transzformációs paraméter elérhető és egyszerűen kimásolható. Annak érdekében, hogy ne kelljen minden EPSG számot fejből tudni, már a script legelején megnevesítettem őket az alábbi módon:

```
WebM = 'EPSG:3857'
WGS84 = 'EPSG:4326'
EOV = 'EPSG:23700'
```

3-2. ábra: A vetületi azonosítók kiírása állandókba

Következő elem a térképen a ’layers’ (rétegek). A rétegek többféle típusba sorolhatóak, melyek közül fontos kiemelni a ’Tile layer’ (tükörfordításban csempe réteg) szerepét. Ez a réteg fogja az alaptérképünk, az úgynevezett ’Base map’ szerepét biztosítani, a ”csempés” rendszer pedig a térinformatika egyik legfontosabb eszköze annak érdekében, hogy a különböző zoom level előállításához kisebb számítási kapacitásra legyen szüksége a rendszernek [7] [10](*3-3. ábra*).



3-3. ábra: Az úgynevezett csempék viszonya egymáshoz a különböző zoom levelen [10]

A kész rendszernél szeretnénk majd, hogy a felhasználó több térkép közül is választhasson, azonban a fejlesztések jelenlegi állapotában csak az Open Street Map © alaptérképet használtuk. Az OpenLayers könyvtárban számtalan alaptérkép elérhető [11], de ezen felül is lehet további alaptérképeket hozzáadni a fejleszteni kívánt térképhez.

A térképünkön további rétegként került elhelyezésre egy vektor réteg, mely a ’markerLayer’ nevet kapta. A réteg induláskor üres, azonban az interakció során módosul.

A webtérkép legfontosabb eleme, az interakció. Az interakciók amiket végre tudunk hajtani egy térképpel teszi igazán különössé a webtérképet, nélkülük gyakorlatilag semmivel sem tud többet, mint egy papír alapú térkép. Az OpenLayers külön kód megírása nélkül biztosítja a felhasználó számára a ’Zoom In’, ’Zoom Out’ interakciót (*3-4. ábra*), továbbá a térképen kurzorral (vagy telefonon az újunkkal) tudjuk mozgatni a térképet [5] [9]. A fejlesztés során szempont volt, hogy a térkép méhészek számára készül, így fontos, mivel hogy ők nem informatikusok, ezáltal a lehető legegyszerűbb térképet kell elkészíteni, lehetőleg kevés interakcióval.

3-4. ábra: OpenLayers segítségével készített térkép, bal felső sarokban látható alap Zoom In, Zoom Out interakciókkal [9]

3.1.2. A rendszerben alkalmazott térképi interakció

A térképi interakciókat egyetlen, a térképen elhelyezett gomb segítségével kívántuk elérni, melynek megnyomásával a rendszer lekérdezi a felhasználó helyzetét, valamint lehetőséget biztosít számára, hogy saját, úgynevezett ’marker’-t helyezzen el. A gombot úgy kellett elhelyezni, hogy az ne legyen zavaró a felhasználó számára, és a lehető legkevesebbet takarja ki a térképből. Praktikus volt továbbá, hogy folytatólagosan helyezkedjen el a Zoom gombokkal, valamint, hogy ugyanazt a stílust kövesse. Az OpenLayers cookbook-ban [9] találtam rá útmutatót, hogy ezt hogyan lehet a legegyszerűbben kivitelezni. Következő szempont volt, hogy a gomb ikonja egyértelmű legyen a felhasználónak, ezért a ”Font Awesome” weboldalán [6] megkeresetem a számomra legideálisabb ikont, melynek csak kimásoltam a hivatkozását, és a HTML kódhoz korábban hozzáadott könyvtárnak köszönhetően egyszerűen hozzá tudtam adni a kódomhoz (*3-5., 3-6. ábra*).

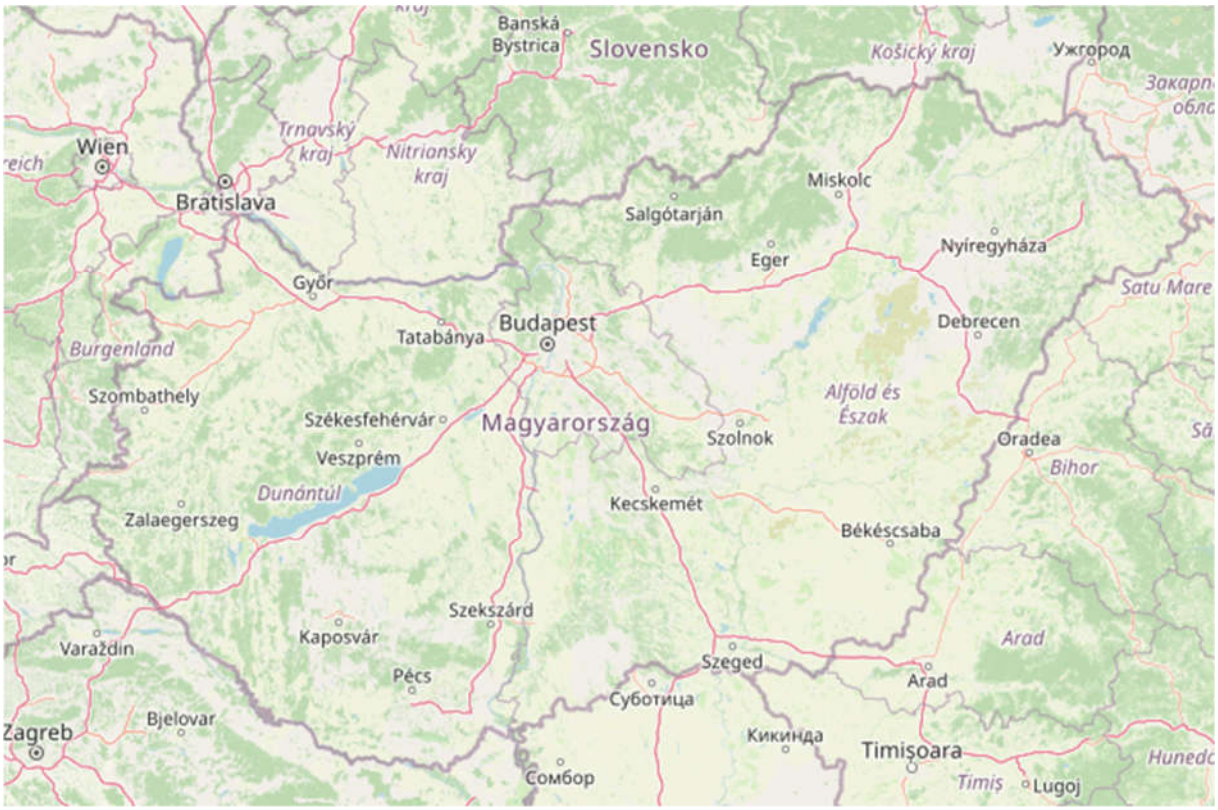
```
button.innerHTML = '<i class="fa-solid fa-map-location-dot"></i>';
```

3-5. ábra: A JavaScript-ben található Font Awesome hivatkozás, mely az ikont biztosítja [6].

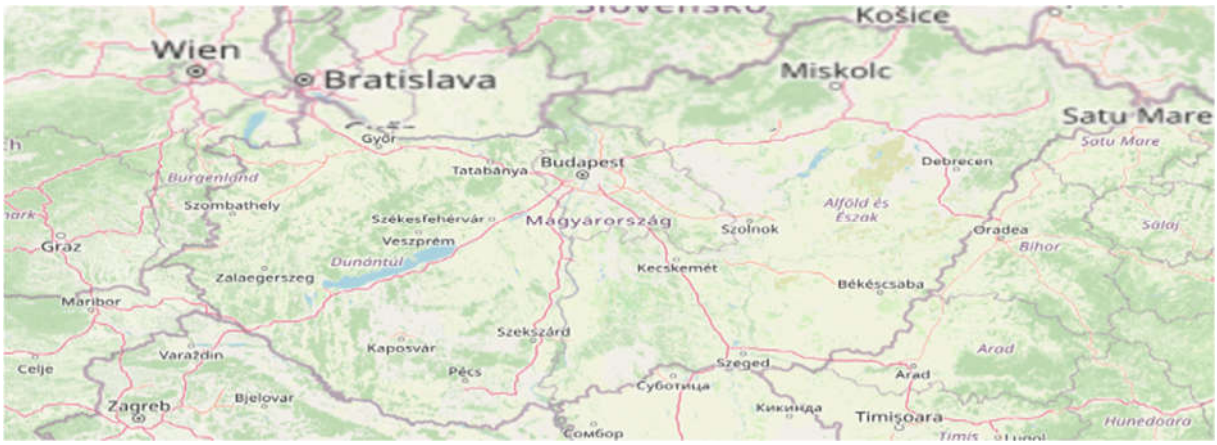
3-6. ábra: Az új gomb elhelyezése a térképen, valamint a kezdőképernyőn megjelenő ikonja [9].

A következő lépés a gomb működésének beállításai. Először is létrehoztam egy ’Mpoint’ (Marker point/ marker pont) elnevezésű globális változót a script legelején, majd hozzá adtam a térképhez egy eventet, ami azt tette lehetővé, hogy minden egyes kattintás után az ’Mpoint’ koordinátája vegye fel a kattintás helyét, valamint, hogy a consol ablakban ezt az adatot írja ki. Az első próba után tudatosult bennem, hogy a vetületekkel probléma lesz, ugyanis a PostGIS-ben megírt távolságalapú lekérdezéshez alapértelmezésben WGS84 vetületi koordináták kellenek. Az OpenLayers alapértelmezetten Web Mercatort használ, azonban könnyen ki lehet cserélni WGS84 rendszerre, de abban az esetben a térképünk megnyújtottan jelenik meg, ami egy torz térképet eredményez a felhasználó számára, és ezáltal ronthatja a felhasználói élményt [7] (*3-7., 3-8. ábra*). Megoldásként két opció jöhet szóba, vagy PostGIS-ben transzformáljuk a koordinátákat, vagy a webtérképen [3]. Az utóbbi könnyebben kivitelezhetőnek tűnt, ezért én azt választottam, mivel az OpenLayers rendelkezik erre már előre megírt függvénnyel [9]. A megoldást két lépesben kell megoldanunk, mivel a koordinátára egy marker-t is szeretnénk elhelyezni, így a későbbiekben is szükségük van az ’Mpoint’ nevezetű változóra, ezért még egy globális változót hoztam létre, amely a ’PMpoint’ (Projection Marker point/ Vetület Marker pont) elnevezést kapta. A ’PMpoint’ változó szintén üres az

induláskor, azonban kattintáskor az alábbi függvény lefut: `PMpoint = ol.proj.transform(evt.coordinate,WebM,WGS84)` A függvény működése a következő: az 'ol.proj.transform' (OpenLayers projection transformation / OpenLayers vetületi transzformáció) függvény után zárójelben a transzformálni kívánt koordináta kerül, majd a kiinduló vetület, és végül a cél vetület vesszővel elválasztva.



3-7. ábra: Magyarország Web Mercator vetületben.



3-8. ábra: Magyarország WGS84 rendszerben. A torzulást az okozza, hogy egységnyi szög, hosszúságban kisebb távolságnak felel meg, mint szélességi szög esetén. A térkép tetején különböző zoom levelhez tartozó csempék megjelenése is megfigyelhető

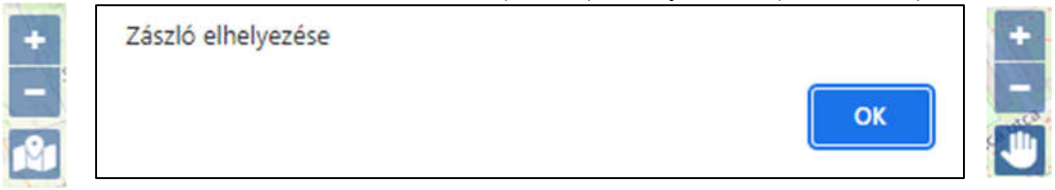
Egy ilyen gombbal szemben a felhasználónak jogos elvárása lehet, hogy az interakciót ki-be lehessen kapcsolni, ugyanazzal a gombbal, valamint, hogy ezt egyértelműen jelezze a

felhasználó felé. Ezt egy if-else megoldással oldottam meg (*3-9. ábra*), valamint egy úgynevezett ’flag’-gel. A ’flag’ egy olyan változó amely vagy igaz (true), vagy hamis (false) értéket vett fel, a jelen kódban a ’buttonIsClick’ (a gomb meg van nyomva) elnevezést kapta.

```
var StartStop = function() {
  if (buttonIsClick){
    map.removeEventListener('singleclick')
    buttonIsClick = false
    button.innerHTML = '<i class="fa-solid fa-map-location-dot"></i>';
  }
  else {
  map.on('singleclick', function(evt){
    Mpoint = evt.coordinate;
    PMpoint = ol.proj.transform(evt.coordinate,WebM,WGS84)
    console.log(PMpoint)
      iconFeature.setGeometry(new ol.geom.Point(Mpoint))
  })
  buttonIsClick = true
  geolocation.setTracking(true)
  button.innerHTML = '<i class="fa-solid fa-hand"></i>';
alert('Zászló elhelyezése')
}
};
```

3-9. ábra: Az if-else megoldás a programban

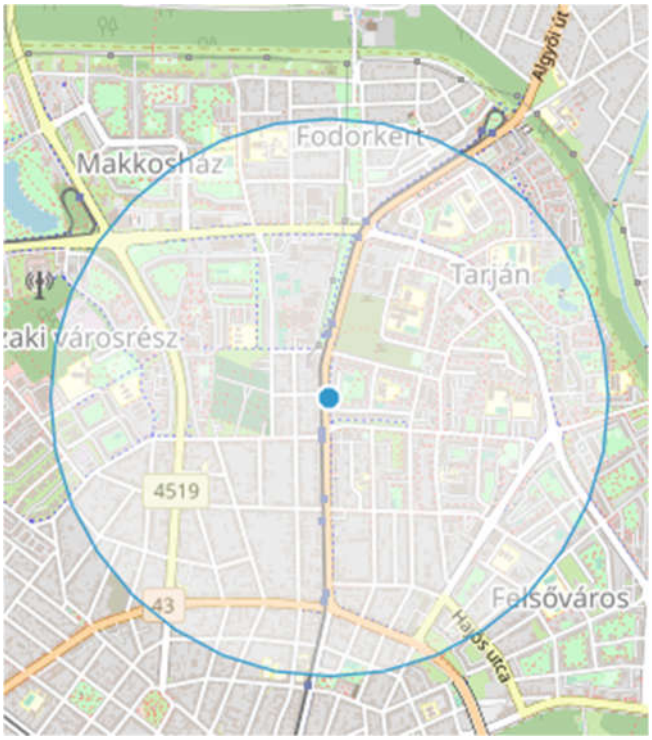
A 3-9 ábrán azt látjuk, hogy a gomb megnyomásával elindul a ’StartStop’ nevezetű funkció, mellyel végig fut egy ’if-else’ kódblokk. Először megnézi, hogy a ’flag’ igaz-e (induláskor alapértelmezésben hamis). Amennyiben igaz, a „térképen való kattintás” eventet visszavonja, a ’flag’-et hamisra állítja, és a gomb ikonját visszaállítja kiinduló állapotára. Amennyiben viszont hamis, a „térképen való kattintás” eventet elérhetővé teszi, és minden egyes kattintás vetületi koordinátája eltárolódik az ’Mpoint’ változóban és felülírja az előzőt. A ’PMpoint’ esetében kicsit bonyolultabb, mert ott először átváltja az a kattintás koordinátáit a cél vetületben, és utána tárolja el a változóban, mindig felülírva az előzőt. Minden kattintás után, a ’consol’ kiírja a ’PMpoint értékét, de erre igazából csak a fejlesztés során történő visszajelzés miatt volt szükség. Az ikon elhelyezése, egy marker elhelyezést takar az Mpoint koordinátáin. Ezután a ’flag’-et igazzá teszi, elindítja a geolokációt, és megváltoztatja a gomb ikonját egy tenyérre. Az új ikonra ezért van szükség, hogy a felhasználó tudja, hogyha rákattint akkor az interakció, melyet a gomb hozott létre leáll. Végül pedig feldob egy figyelmeztetést, hogy a felhasználó elérhetővé tette a marker (zászló) elhelyezését (*3-10. ábra*).



3-10. ábra: Bal oldalon a kikapcsolt állapotban lévő gomb, középen a bekapcsolásakor megjelenő üzenet, jobb oldalon pedig a bekapcsolt állapotban lévő gomb látható, melyre, ha rákattintunk újra kikapcsolttá teszi.

3.1.3. A geolokáció és Marker elhelyezése

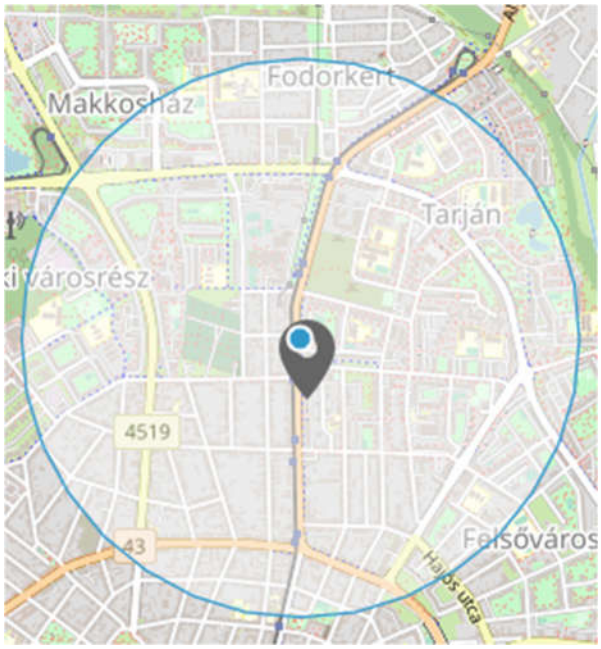
Az előző alfejezetben már említésre került geolokáció, amely egy általános információs koncepció, mely fizikai tárgyak vagy emberek földrajzi helyzetének meghatározását jelenti, leggyakrabban az eszköz GPS- vagy IP-címét használva [12]. Ez a gyakorlatban számunkra azt fogja jelenteni, hogy amikor a térképen elhelyezett gombot megnyomjuk, a telefonunk engedélyt fog kérni tőlünk, hogy felhasználja a helyadatainkat. Amikor ezt lehetővé tesszük a rendszer számára, meg fog jelenni egy kék pont a térképen, mely jelzi számunkra a saját helyzetünket. Mint tudjuk, a telefonos helyzetmeghatározás nem a legpontosabb, viszont a méhészeti állományok bejelentésére, megfelelő lehet. Azonban annak érdekében, hogy a felhasználó nagyjából tudja, hogy helyzete mennyire megbízható, a rendszer kirajzolja számára a geolokáció középhibáját (*3-11. ábra*).



3-11. ábra: A geolokáció és középhibája, jelen esetben ez 920m , azonban figyelembe kell venni, hogy ezt Ip cím alapján állapította meg mely pontatlan. Telefon és GPS mérés esetében természetesen ez pontosabb lesz.

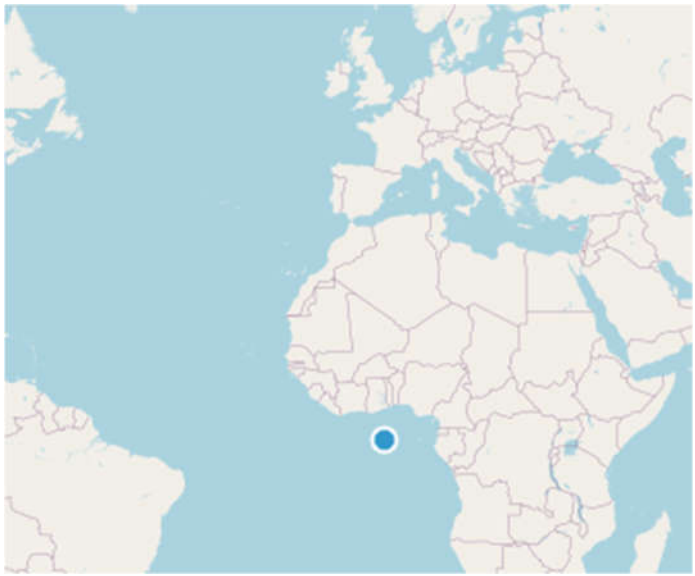
Abban az esetben, ha a középhiba miatt pontosítani szeretne a felhasználó, térképre való kattintással ezt megteheti. A kattintás által fekete Marker fogja jelezni a pontos helyzetet, amit a felhasználó ítél meg (*3-13. ábra*). A kirajzolt középhiba ebben is segíteni fogja a felhasználót, mivel 66%-os valószínűséggel a valós helyzet bele fog esni a számolt középhiba tartományba [12] (*3-12. ábra*).

3-12. ábra: Az Ip címből adódó rossz helyzetmeghatározás, és a valós helyzet



3-13. ábra: A valós helyzet beleesik a középhiba tartományába

A geolokációnál szintén fontos megemlíteni, hogy a vetületi beállításokra oda kell figyelni, mivel alapértelmezésben az OpenLayers-ben megírt függvényünk WGS84 koordinátát hoz létre, és ha ezt meghívjuk a térképünkre hibás helyzet fog kirajzolódni a felhasználó számára [9].



3-14. ábra: Példa a rossz vetület alkalmazására. A felhasználó WGS84 rendszerbeli koordinátái, elhelyezve a Web Mercator rendszerben készült térképünkön.

Mivel azonban nekünk a kimentett koordináta WGS84 vetületi rendszerben kell, így létrehoztam egy újabb globális változót, amely a ’poscoordinates’ (pozíció koordináták) nevet

kapta, és szintén elvégeztem rá a vetületi transzformációt. Abban az esetben, ha a felhasználó Marker-t helyez el, a rendszer azt fogja továbbítani az adatbázis felé.

3.2. Térkép szerepe a rendszerben

A térkép központi eleme a rendszernek, ugyanis amellett, hogy a bejelentést támogatja a méhészeknek, döntéshozás támogatási szerepe is van. A bejelentés megvalósítása egyszerű, hiszen a felhasználó kiválasztja melyik állományát szeretné bejelenteni, majd mér egy koordinátát és amennyiben úgy ítéli meg, manuálisan javíthatja a térképen a pozíciót, majd elmenti, és a rendszer eltárolja 'VALID' adatként a 'locations' táblában (*2-4. ábra*). Ennél valamivel komplexebb a tervezés támogató rendszer.

A tervezés támogatása alatt a méhész vándortelephely kiválasztás elősegítésének folyamatát értjük. A méhész, amikor megnyitja a térképet a rendszeren belül, térbeli lekérdezéseket tehet bárhonnan, akár saját helyzetéről is. Ebben az esetben a lekérdezni kívánt pont koordinátái bekerül a rendszerbe, és egy PostGIS-ben elkészített lekérdezésben behelyettesítésre kerül [3] (*3-15. ábra*).

```
1 usage
@Query(value = "SELECT * FROM locations WHERE ST_DWithin(geom, ST_MakePoint(:latitude, :longitude)\\:\\:geography, :distance)", nativeQuery = true)
List<LocationEntity> findAllWithinPointAndDistance(Double latitude, Double longitude, Long distance);
```

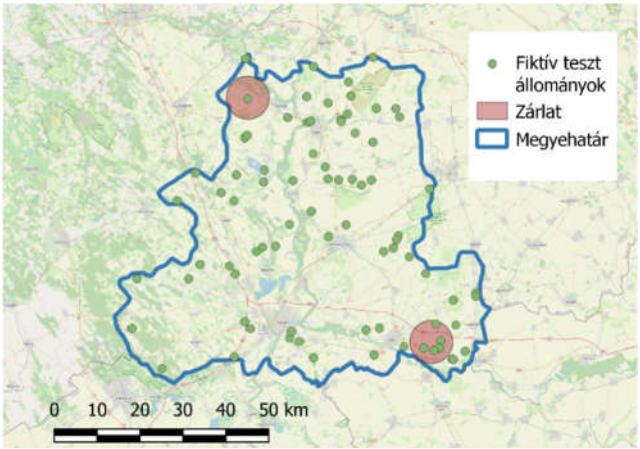
3-15. ábra: A PostGIS-ben elkészített lekérdezés, már adatbázisba beépítve.

Ez gyakorlatilag egy kereső kört fog számunkra eredményezni, melynek sugarát a méhész nem fogja tudni állítani. Ennek a szerepe az adatvédelemben keresendő. Több méhésszel történő egyeztetés után arra a következtetésre jutottam, hogy bár a méhészek statisztikához szeretnének hozzáférni, azt nem szeretnék, hogy a többi méhész a pontos helyzetüket tudja. A lekérdezés után a méhész egy olyan statisztikát fog visszakapni, amelyből látszik, hogy hány méhcsalád van jelenleg is a terület 5 km-es sugarában bejelentve. Azért esett 5 km-es keresősugárra a választás, mert azzal kellő távolságot biztosítunk ahhoz, hogy a méhész biztonságban tudhatja állományának helyzetét. Valamint ha betegség miatt méhészeti zárlat van a területen, akkor az is 5 km-es sugarú körre terjed ki a jelenlegi szabályozás szerint [2]. Mivel minden méhész nagyjából egy időpontban vándorol (ez adódik az adott méhlegelő virágzásából), ha csak azt venné figyelembe a rendszer, hogy mekkora méhállomány van bejelentve az adott területen, nem megfelelő képet festene a felhasználó számára, így a gyakorlatban is működő foglalási rendszert kell a lehető legjobban digitalizálnunk.

Gyakorlatban a foglalási rendszer úgy működik, hogy a méhész még a virágzás megkezdődése előtt ellátogat a kiszemelt méhlegelőre, majd a növényzet állapotából

megpróbálja megbecsülni, hogy milyen minőségű mézelés várható a területen. Ha megfelelőnek ítéli meg, akkor a kiszemelt helyszínre, ahová az állományát telepíteni kívánja elhelyez egy foglaló táblát, mely tartalmazza a méhész nevét, telefon számát és a méhcsaládok számát. Ezután felveszi a kapcsolatot a terület tulajdonosával, és engedélyt kér a letelepedésre. Azonban ha a vándorlás előtt a méhész azt tapasztalja, hogy nagyon sok foglalás van a helyszínen, érdemes elgondolkodnia a terület túlterheltségén. Fordult már elő olyan helyzet a gyakorlatban, hogy egy szépnek megítélt virágzás közel annyi termést se hozott, mint egy sokkal gyérebb virágzás, mivel a szebbnek megítélt méhlegelő túl volt terhelve. A táblás foglalás egyik veszélye azonban, hogy egyes méhészek, sokkal több foglaló táblát elhelyeznek, mint amennyire vándorolni szándékoznak, ezáltal a többi méhészben fals képet festenek a tervezéshez. Ennek a problémának a megakadályozására a rendszerben korlátozzuk a foglaló helyek számát. Mivel a tapasztalatok alapján nagyjából 50 méhcsalád/vándorállomány tükrözi a valóságot, így a vándorállomány minden megkezdett 50 méhcsaládja után a rendszer biztosít a méhész számára egy további foglaló helyet. Alapértelmezésben a rendszer minden felhasználónak biztosít 1 foglaló helyet. Ez a gyakorlatban azt jelenti, hogy például 40 vándor méhcsalád esetén 1+1 foglaló hely lesz biztosítva felhasználónak, 80 család esetén 1+2 foglaló hely lesz biztosítva.

Amikor a méhész vándorlást tervez, továbbra is ki kell majd látogatnia a méhlegelőre, azonban könnyebben tud döntést hozni, ha a lekérdezés után látja, hogy hány méhcsalád van jelenleg is a területen, valamint, hogy hány méhcsaládda terveznek a területre vándorolni. Helyszínen el tudja dönteni, hogy nagyjából milyen virágzás várható a növények állapota alapján, azonban a digitális térkép nyújtotta lehetőségek nagyban hozzájárulnak majd a döntéshozáshoz, továbbá az aktuális térképen mindig feltüntetésre kerülnek majd az éppen aktuális méhegészségügyi zárlatok is (*3-16. ábra*).



3-16. ábra: Méhegészségügyi zárlat a térképen piros körrel ábrázolva. Ezekre a területekre a vándorlás nem lehetséges, továbbá aki a zárlaton belül van nem hagyhatja el a területet.

3.3. A Backendről röviden

A backend Java alapú, mely számunkra azért volt előnyös, mert a forráskódból csak bájtkódot állít elő, nem pedig közvetlenül futtatható gépi kódot. A bájtkódot a 'Java Virtual Machine' (Jáva virtuális gép, röviden JVM) segítségével fordítják át egyéb kódra, tehát a Java-ban megírt program bármilyen eszközön futtatható, amire létezik JVM [13].

A rendszer Spring Boot keretrendszert használ, amely a Spring-keretrendszer kiterjesztéseként elkészült modul [13] [14]. A Spring nyílt forráskódú projekt, amely 2003-ban indult a Java fejlesztés bonyodalmainak kiküszöbölésére, megkönnyítve a Java alkalmazások fejlesztését, ezáltal időt spórolva a fejlesztőknek. Thymeleaf (egy különálló 'template engine' (szabad fordításban sablon motor) mely egy kiegészítő pluginnak köszönhetően használható) segítségével dinamikusan generáljuk a html-t, meglévő statikus oldalvázra [15].

A kód strukturális elrendezéséhez 'Controller-Service-Repository' mintát [16] követtem, amely segítségével könnyen átlátható és fejleszthető kódot hoztam létre, melyhez a későbbi fejlesztések során könnyedén hozzá tudok nyúlni és további fejlesztéseket tudok benne eszközölni. A minta 3 fő részből áll, a 'Controller' amivel összekapcsolom a 'REST Interface'-en keresztül a klienst és a biznisz logikát, melynek segítségével a rendszer 'JSON' fájlt képes adni és fogadni. A 'Service' rétegben megvalósítom a biznisz logikát valamint a 'Repository' réteg segítségével eltárolom az adatokat az adatbázisban. A biznisz logika a program azon része, amely a valós üzleti szabályokat kódolja, melyek meghatározzák, hogyan lehet adatokat létrehozni, tárolni és megváltoztatni [17].

A 'JSON' (JavaScript Object Notation) egy kis erőforrás igényű adatátviteli forma, mely tartalmazza a szükséges adatot a kommunikációhoz, JavaScript nyelvből alakult ki, de programozási nyelv független [18].

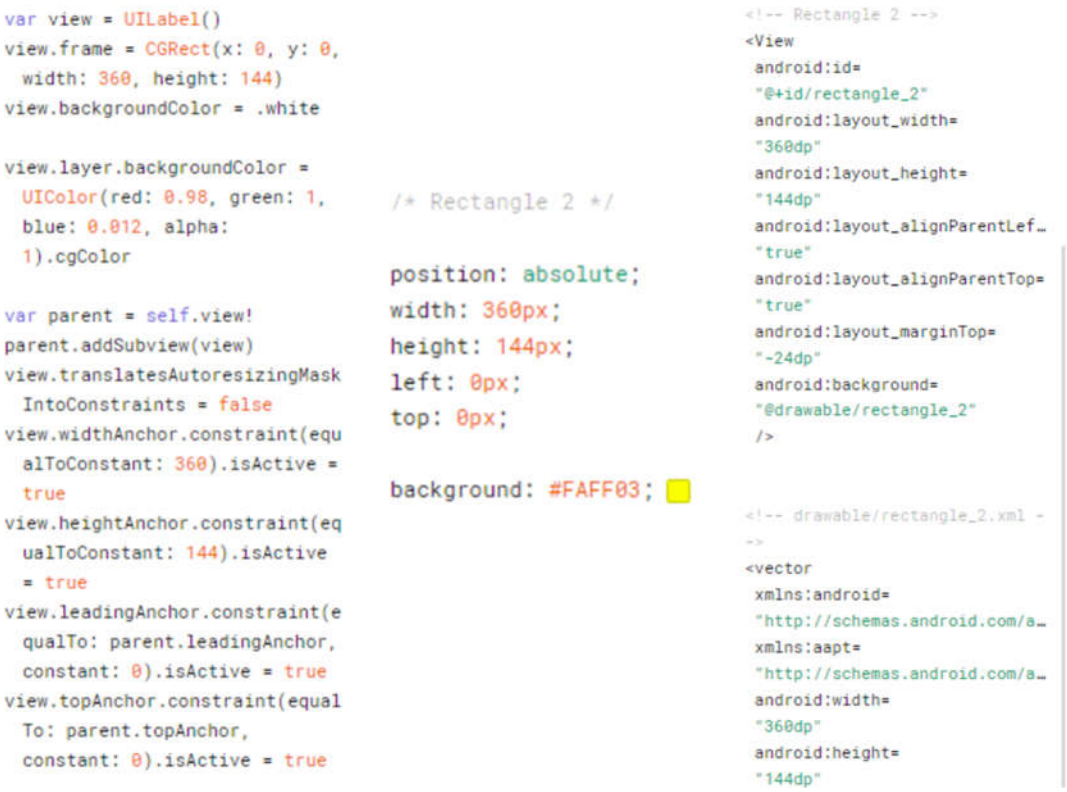
3.4. Az alkalmazás

Ebben az alfejezetben bemutatásra kerül az alkalmazás működése és dizájnja. Jelenleg web alkalmazás formájában készült el, azonban a későbbiekre való tekintettel Android és IOS operációs rendszerre is szeretnénk alkalmazást fejleszteni, ezáltal megkönnyítve a felhasználók munkáját, valamint emelve a felhasználói élményt az alkalmazás használata közben.

3.4.1. A dizájn

Az alkalmazás megtervezésénél elsődleges szempont volt egy olyan dizájn elkészítése, amely a felhasználók számára könnyen értelmezhető, letisztult, ugyanakkor küllemre szép és

ezáltal a felhasználó is szívesen veszi igénybe a rendszer szolgáltatásait. A tervezéshez egy nyílt forráskódú, *Figma* elnevezésű felülettervező rendszert [19] használtam, melynek előnye, az egyszerű dizájn tervezés mellett a prototípus tervezés, valamint az elkészített elemeket kód formában ki lehet másolni, Web (CSS), Android, és IOS rendszer fejlesztésekhez egyaránt (*3-17. ábra*). Mivel jelenleg weboldalt fejlesztettünk így egyelőre csak a CSS kódokra volt szükségünk, későbbiekben azonban jobban ki tudjuk használni a rendszer nyújtotta lehetőségeket.



3-17. ábra: Példa a Figma által generált kódokra, ugyanarra az elemre. Baloldalon IOS, középen CSS, jobb oldalon Android kódból egy részlet látható.

A prototípustervezés azt jelenti, hogy a fejlesztő már a dizájnnál megtervezheti, hogy melyik gomb hova vezesse át a későbbi felhasználót (*3-21. ábra*), valamint, animációkat is beletervezhet, melyeket egy próba módban (*3-18. ábra*) lefuttathat. Az alkalmazás fejlesztéséhez azonban ezeket a fejlesztőnek önállóan kell lefejlesztenie, mivel a Figma rendszeréből ezeket nem lehet kimenteni.

A felhasználónak először a bejelentkező felület fog megjelenni, melyen saját felhasználó nevével, illetve jelszavával bejelentkezhet. Amennyiben nem emlékszik a jelszóra kérhet emlékeztetőt, vagy amennyiben nincs felhasználó profilja regisztrálhat (*3-18 ábra*). Bejelentkezést követően a főmenü felülete jelenik meg, ahol 4 tevékenység közül választhat a felhasználó: ’Állományok kezelése’, ’Állomány bejelentés’, ’Vándorhely foglalás’ és ’Térkép’.

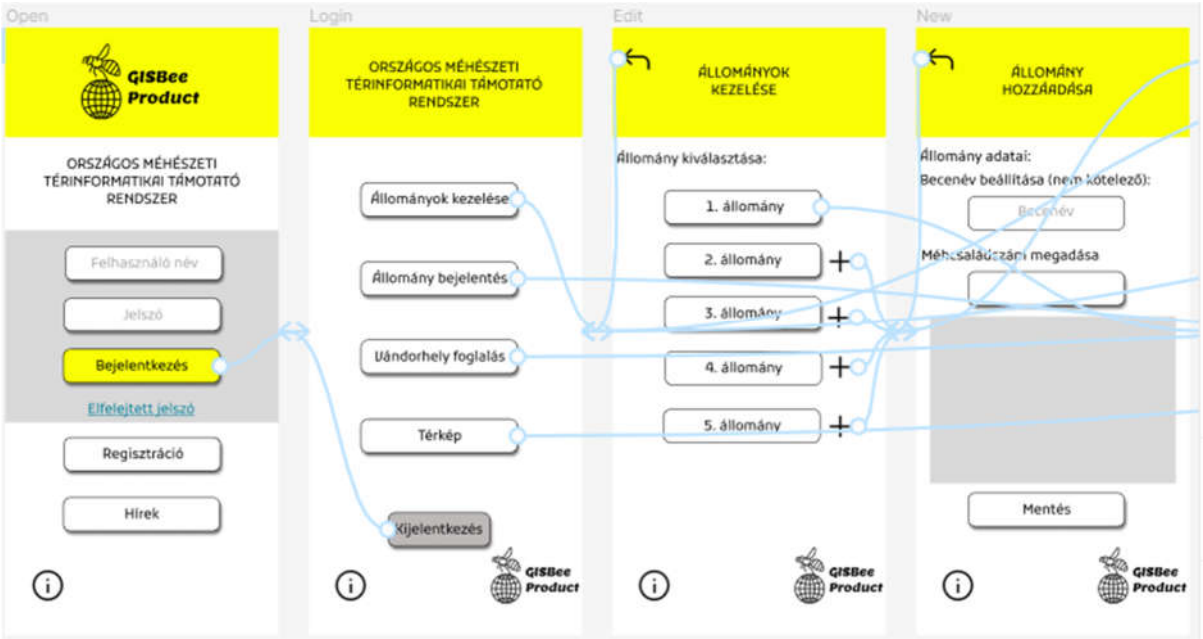
3-18. ábra: A Figma prototípus üzemmódban, az applikáció bejelentkezési felületén

Az ’Állományok kezelése’ menüpont megnyomásával a felhasználó szerkesztheti a méhállományokat, valamint hozzáadhat újabb állományt is. A szerkesztésnél beállíthat az állomány számára becenevet, ha számára úgy könnyebb az azonosítás. Alapértelmezésben sorszámot kapnak, valamint módosíthatja a méhcsalád számát is. Új állomány hozzáadásánál ugyanúgy hozzáadhat becenevet ha szeretne, viszont ebben az esetben a méhcsalád szám megadása kötelező elem.

A főmenü következő menüpontja az ’Állomány bejelentés’. Ennél a menüpontnál a felhasználó kiválaszthatja a már létező állományok közül, hogy melyik állományt szeretné bejelenteni, ezután pedig a webtérkép jelenik meg számára, melynek segítségével a felhasználó a korábban említett (*3.2. alfejezet*) módszer segítségével bejelentheti állományát a rendszer felé. Ehhez a folyamathoz hasonló módon működik a ’Vándorhely foglalás’ menüpont, azzal a különbséggel, hogy az állományok nem ’VALID’-ként jelennek meg az adatbázisban, hanem ’RESERVED’ értéket kapnak a ’locations’ táblán belül (*2.2. alfejezet*), továbbá a vándorhely foglalásnál, már lehetősége van a felhasználónak terület lekérdezést is véghez vinnie. Végezetül utolsó menüpont a főmenüben a ’Térkép’, mely megnyomásával egyből a webtérkép jelenik meg, és ezen a felületen terület lekérdezést tudunk végrehajtani.

3-19. ábra: Részlet a Figma tervezői felületéből, az alkalmazás első 4 elkészült ablakáról.

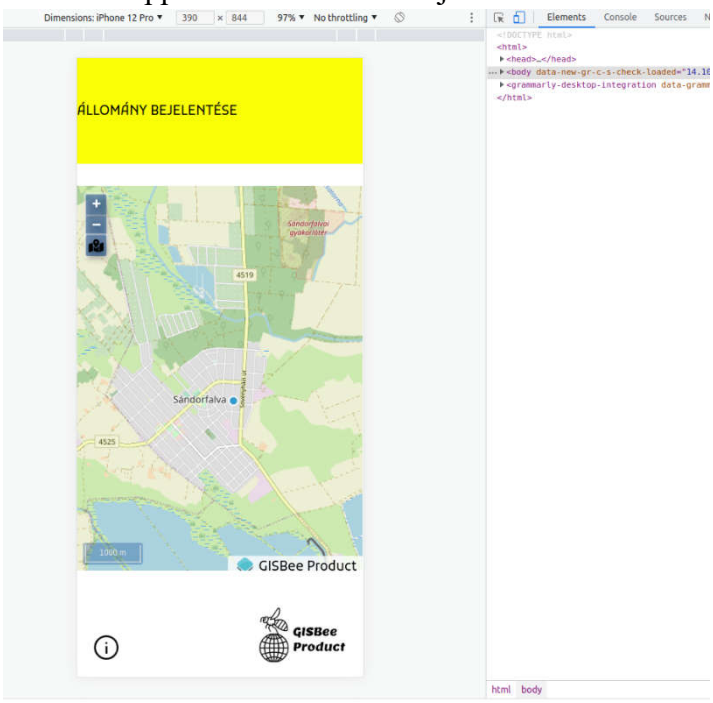
A dizájn megtervezésekor 9 darab ablak került megtervezésre, melyeknél mindig törekedtem a lehető legegyszerűbb megjelenésre. A színek kiválasztásánál olyan színek kombinációját igyekeztem használni, mely véleményem szerint könnyen összeköthető a méhészettel, valamint a rendszerben nincs idegen hatása. A domináns szín választásánál a sárgára került a választás. A rendszerhez továbbá kitaláltunk egy brandet is, mely '*GISBee*' elnevezést kapta (GIS a földrajzi információs rendszerre, a Bee pedig a méhekre vonatkozik), valamint hozzá illő logót is (*3-20. ábra*).



3-20. ábra: Az első négy felület, a Figma prototípus fejlesztő felületében. A világoskék nyilak jelzik, hogy melyik gomb lenyomása, hova navigálja a felhasználót.

3.4.2. Applikáció fejlesztés

A kész dizájn kimentése a Figma rendszeréből könnyedén megtehető, amely után a következő lépés a gombok paranccsal való ellátása volt melyet JavaScript segítségével oldottunk meg. Az applikáció ugyanazt a kódbázist használja mint a weboldal, azonban később Androidra és IOS-re is külön applikációt tervezünk fejleszteni.



3-21. ábra: Részlet az aplikáció fejlesztésből, a benne elhelyezett térképpel.

3-22. ábra: A termék brandjének logója.

4. Összegzés

Bár a méhészet kisebb társadalmi figyelmet kap, mint az agrárterület legtöbb résztvevője, fontos részét képzi, akár a virágok beporzását vesszük figelembe, akár az európai méztermelésből kivett részét. A méhészek támogatására ezért olyan rendszert kell kialakítani, mellyel lecsökkenthetjük az adminisztrációs terhet. Mivel a méhészek többsége vándorméhészettel foglalkozik és ezáltal hatalmas területeket vándorolnak be az ország területén, így praktikus elképzelés lehet számukra egy térinformatikai rendszer üzemeltetése, mely vizuális elemmel (térképpel) van ellátva. Ezáltal sokkal könnyebben és számukra is átláthatóbban tudják a bejelentést intézni, valamint a vándorlást segítő adatoknak köszönhetően a vándortelephelyek kiválasztása is nagyban egyszerűsödik.

A rendszer fejlesztése során többször is kikértük jövőbeli felhasználó (méhész) tanácsát. Véleményünk szerint a rendszer használata könnyen elsajátítható, informatikában kevésbé jártas szakemberek számára is. Sajnos az idő rövidsége miatt nem volt időnk minden egyes rész kidolgozására, viszont a rendszer egyik fő elemét képző térkép az elvárásoknak megfelelően produkál. A térképi megjelenítés fejlesztése közben hasznomra volt a korábban megszerzett térinformatikai tudásom és szemléletem, mivel annak ellenére, hogy bár az asztali térinformatikai programokkal ellentétben, most teljes egészében kódsorokból kellet dolgozni, egy térkép felépítésének folyamata ugyanazt a logikai vonalat követte, mint a grafikus térképszerkesztő programok esetében.

A jövőben megvalósuló fejlesztések közé tartozik, a térképen megjelenő, felhasználó által bejelentett állományok (egyelőre csak az adatbázisban jelenik meg, melyet egy asztali térinformatikai program segítségével tudunk vizuálissá tenni), valamint a méhegészségügyi és OMME felület elkészítése. A méhegészségügyi felület úgy épül fel terveink szerint, hogy minden méhegészségügyi felelős területe egy-egy poligont képezne, amelyre, ha felkerül egy új vándortelephely, az adott méhállomány korábbi egészségügyi adataihoz a terület méhegészségügyi felelőse online hozzáférne, ezáltal már előre tudná, mire számíthat. A méhegészségügyi felelős szerkesztheti továbbá a rendszerben jövőben megvalósuló online egészségügyi kiskönyvet, mellyel minden állomány rendelkezne. Az OMME felhasználó szintén egy kibővített jogokkal rendelkező felhasználó lenne. Jogosultságai közé tartozik, hogy az egész ország területén lévő méhészeti állományokat képes lesz látni, időben elemezni a helyzetüket, valamint országos statisztikákat tud lekérdezni.

Irodalomjegyzék

[1] „European Commission, Agriculture and rural development,” European Union (Európai Unió), [Online]. Available: https://agriculture.ec.europa.eu/farming/animal-products/honey_en. [Hozzáférés dátuma: 07 11 2022].

[2] „70/2003. (VI. 27.) FVM rendelet,” Magyar Közlöny Lap- és Könyvkiadó Kft., 30 12 2017. [Online]. Available: https://njt.hu/jogszabaly/2003-70-20-82. [Hozzáférés dátuma: 07 11 2022].

[3] „PostGIS,” PostgreSQL, [Online]. Available: https://postgis.net/. [Hozzáférés dátuma: 07 11 2022].

[4] V. AGAFONKIN, „Leafletjs,” Leaflet, [Online]. Available: https://leafletjs.com/. [Hozzáférés dátuma: 07 11 2022].

[5] „OpenLayers,” OpenLayers, [Online]. Available: https://openlayers.org/. [Hozzáférés dátuma: 07 11 2022].

[6] „Font Awesome,” [Online]. Available: https://fontawesome.com/. [Hozzáférés dátuma: 07 11 2022].

[7] M. MONOMIER, HOW TO LIE WITH MAPS 3rd Edition, Chicago: The University of Chicago Press, 2018.

[8] „EPSG,” IOGP, [Online]. Available: https://epsg.org/home.html. [Hozzáférés dátuma: 07 11 2022].

[9] „OpenLayers v4.6.5 Examples,” OpenLayers, [Online]. Available: https://openlayers.org/en/v4.6.5/examples/. [Hozzáférés dátuma: 07 11 2022].

[10] F. VÉGSŐ, Térinformatika 4. Raszteres adatszerkezet, Székesfehérvár: Nyugat-magyarországi Egyetem Geoinformatikai Kar, 2010.

[11] „OpenLayers v4.6.5 API,” OpenLayers, [Online]. Available: https://openlayers.org/en/v4.6.5/apidoc/. [Hozzáférés dátuma: 07 11 2022].

[12] „GEOLOCATION,” Geolocation.com, [Online]. Available: https://www.geolocation.com/. [Hozzáférés dátuma: 07 11 2022].

[13] „Azure Microsoft,” Microsoft, [Online]. Available: https://azure.microsoft.com/hu-hu/resources/cloud-computing-dictionary/what-is-java-spring-boot/. [Hozzáférés dátuma: 07 11 2022].

[14] „Spring Boot,” [Online]. Available: https://spring.io/projects/spring-boot. [Hozzáférés dátuma: 07 11 2022].

[15] W. DEBLAUWE, Taming Thymeleaf: Practical Guide to Building a Webapplication with Spring Boot and Thymeleaf, Lulu.com, 2021.

[16] T. Collings, „Controller-Service-Repository,” 10 08 2021. [Online]. Available: https://tom-collings.medium.com/controller-service-repository-16e29a4684e5. [Hozzáférés dátuma: 07 11 2022].

[17] J. FRANKENFIELD, „Business Logic,” 28 08 2020. [Online]. Available: https://www.investopedia.com/terms/b/businesslogic.asp. [Hozzáférés dátuma: 07 11 2022].

[18] „Introducing JSON,” 12 1999. [Online]. Available: https://www.json.org/json-en.html. [Hozzáférés dátuma: 07 11 2022].

[19] „Figma,” Figma, [Online]. Available: https://www.figma.com. [Hozzáférés dátuma: 07 11 2022].

Köszönetnyilvánítás

A Kulturális és Innovációs Minisztérium ÚNKP-22-1 kódszámú Új Nemzeti Kiválóság Programjának a Nemzeti Kutatási, Fejlesztési és Innovációs Alapból finanszírozott szakmai támogatásával készült.

Szeretnék köszönetet mondani Gábor Dániel, gyerekkori barátomnak, aki a legtöbb segítségnyújtást eszközölte a projekt megvalósulása érdekében. Köszönöm továbbá Dr. Molnár Gábor konzulensemnek a jótanácsokat és tippeket, valamint édesapámnak, Böröcz Istvánnak, és édesanyámnak, Böröczné Simon Margitnak, akik sokéves méhészeti szaktudásukkal és tapasztalataikkal nyújtottak számomra támogatást.

Köszönet illeti továbbá Lászlóffy Zsoltot, az OMME Csongrád-Csanád megyei szaktanácsadóját meglátásaiért, valamint Dr. Pődör Andrea Tanárnőt a térinformatikai tudásom megalapozásáért és a könyvajánlásért, továbbá a Valéta Bálon megejtett eszmecseréért. Hálás vagyok Dr. Földváry Lóránt Tanár Úrnak is, aki annak idején felkeltette bennem a programozás iránti érdeklődést, melyet azóta már számtalan alkalommal kamatoztattam a munkáim során. Végül, de nem utolsó sorban szeretném megköszönni páromnak, legkedvesebb patikusomnak, Albert Annának, akinek ismét végig kellet hallgatnia megannyiszor problémáimat és nehézségeimet, de ennek ellenére végig támogatott és mellettem állt.