ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Alba Regia Műszaki Kar

Geoinformatikai Intézet

SZAKDOLGOZAT

Országos Méhészeti Térinformatikai Támogató Rendszer Fejlesztése

OE-AMK

2022

Hallgató neve: Böröcz Balázs

Hallgató törzskönyvi száma: T000551/FI12904/A-G

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Óbudai Egyetem
Alba Regia Műszaki Kar
Geoinformatikai Intézet

SZAKDOLGOZAT FELADATLAP

Hallgató neve: Böröcz Balázs
Szakdolgozat száma: SZD22011414062785
Törzskönyvi száma: T000551/FI12904/A-G **Neptun kódja:** [redacted]

Szak: földmérő és földrendező mérnök
Specializáció: Geoinformatika specializáció

A dolgozat címe: Országos Méhászeti Térinformatikai Támogató Rendszer fejlesztése
A dolgozat címe angolul: Developing a GIS Based Application for Bee-Keepers
A feladat részletezése:
A jelölt fejlesszen egy olyan térinformatikai rendszert, amely a magyarországi méhészeket támogatja. A rendszer tartalmazzon egy digitális országos méhészeti téradatbázist, mellyel könnyebben kiszűrhetőek lehetnek a nem bejelentett méhészetek, valamint a fertőzési csomópontok, továbbá segíti a méhészeteket olyan információval ellátni, amely megkönnyítené a vándorlástervezési folyamatot. A rendszer tartalmazzon egy telefonos alkalmazást (vagy annak leírását), amely a méhészek számára lehetővé teszi saját adataik egyszerű lejelentését és ezzel az adatbázis feltöltését.
A rendszer tegye lehetővé, hogy egy online platformon keresztül a méhegészségügyi felelősök bejegyzéseket tehessenek, ezzel megkönnyítve a munkájukat és átláthatóbbá tenni az országos méhegészségügyi viszonyokat.
A rendszer biztosítson hozzáférést az Országos Magyar Méhészeti Egyesület (továbbiakban OMME) számára.

Intézményi konzulens neve: Dr. habil Molnár Gábor Péter

A kiadott téma elévülési határideje: 2024. 12. 15.
Beadási határidő: 2022. 12. 15.

A szakdolgozat: Nem titkos.

Kiadva: Székesfehérvár, 2022. 10. 12. P.H

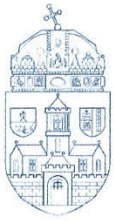
Alba Regia Műszaki Kar ÓBUDAI EGYETEM Geoinformatikai Intézet 1.

.....
Intézetigazgató

A dolgozatot beadásra alkalmasnak találom: 2022. 12. 15.

.....
belső konzulens

.....
külső konzulens



ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Alba Regia Műszaki Kar
Geoinformatikai/Mérnöki/
Természettudományi és Szoftvertechnológiai Intézet

HALLGATÓI NYILATKOZAT

Alulírott hallgató kijelentem, hogy a szakdolgozat/diplomamunka saját munkám eredménye, a felhasznált szakirodalmat és eszközöket azonosíthatóan közöltem. Az elkészült szakdolgozat/diplomamunkában található eredményeket az egyetem és a feladatot kiíró intézmény saját céljára térítés nélkül felhasználhatja, a titkosításra vonatkozó esetleges megkötések mellett.

Kelt: Székesfehérvár 2022.12.11.

.....

hallgató aláírása

Tartalomjegyzék

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.

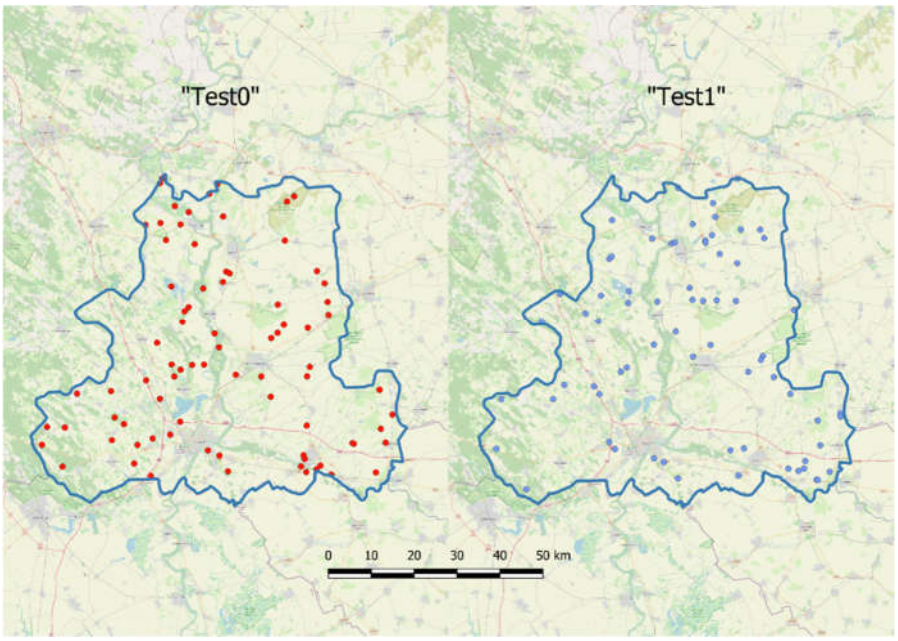
Fontos részt képez továbbá, hogy a méhész képes legyen a rendszerhez pár gombnyomással bejelenteni a méhállományait (egyelőre csak szimulációs szinten), ezáltal a bürokratikus terhei nagy mértékben csökkennének, valamint a szakmaközi szervezet, az Országos Magyar Méhészeti Egyesület (továbbiakban OMME) , 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] eltöltöttem 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.



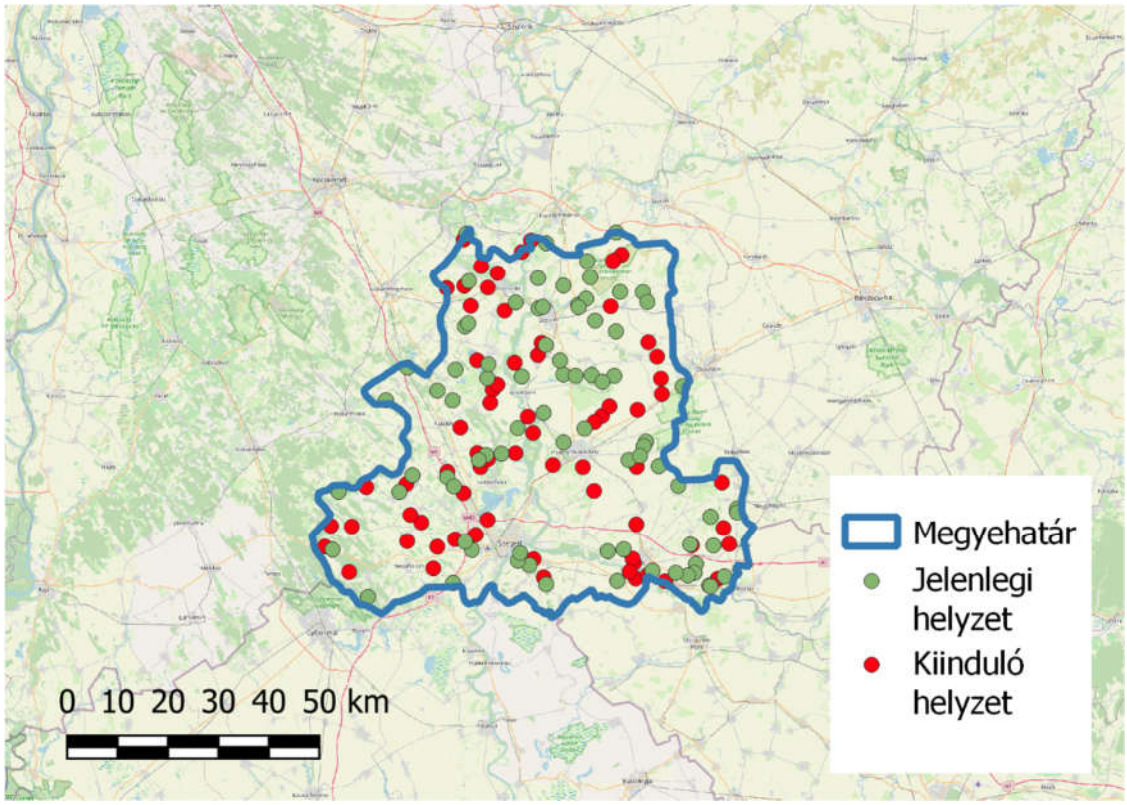
2-1. ábra: a "Test0"csoport összehasonlítása a "Test1" csoporttal

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:

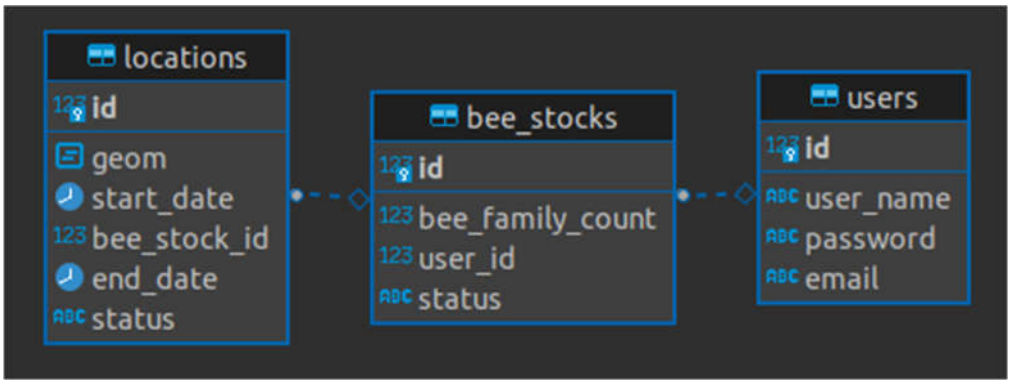
- ’id’ (egyedi azonosító): minden egyes újabb sor kap egy új egyedi azonosítót, annak érdekében, hogy az adatok keveredését megelőzzük, továbbá erre az

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

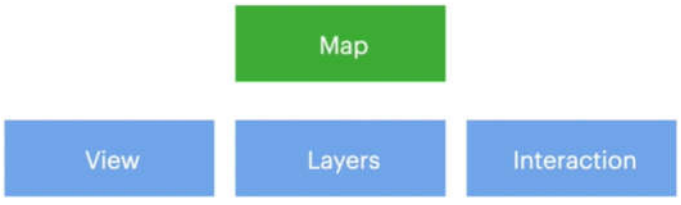
3.1. A webtérkép

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*).



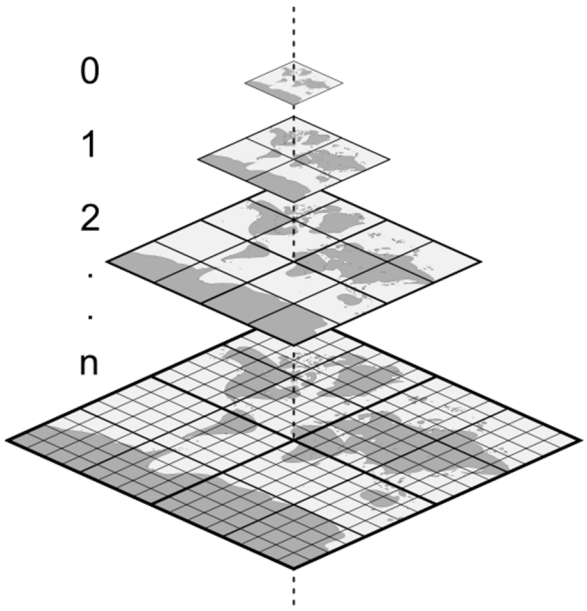
3-1. ábra: A webtérkép összetétele

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 térképen való megjelenítéshez Web Map Service (WMS) réteget alkalmaztam. A WMS egy szolgáltatás, melyet weben történő térképek megjelenítésére alkalmazunk.

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].

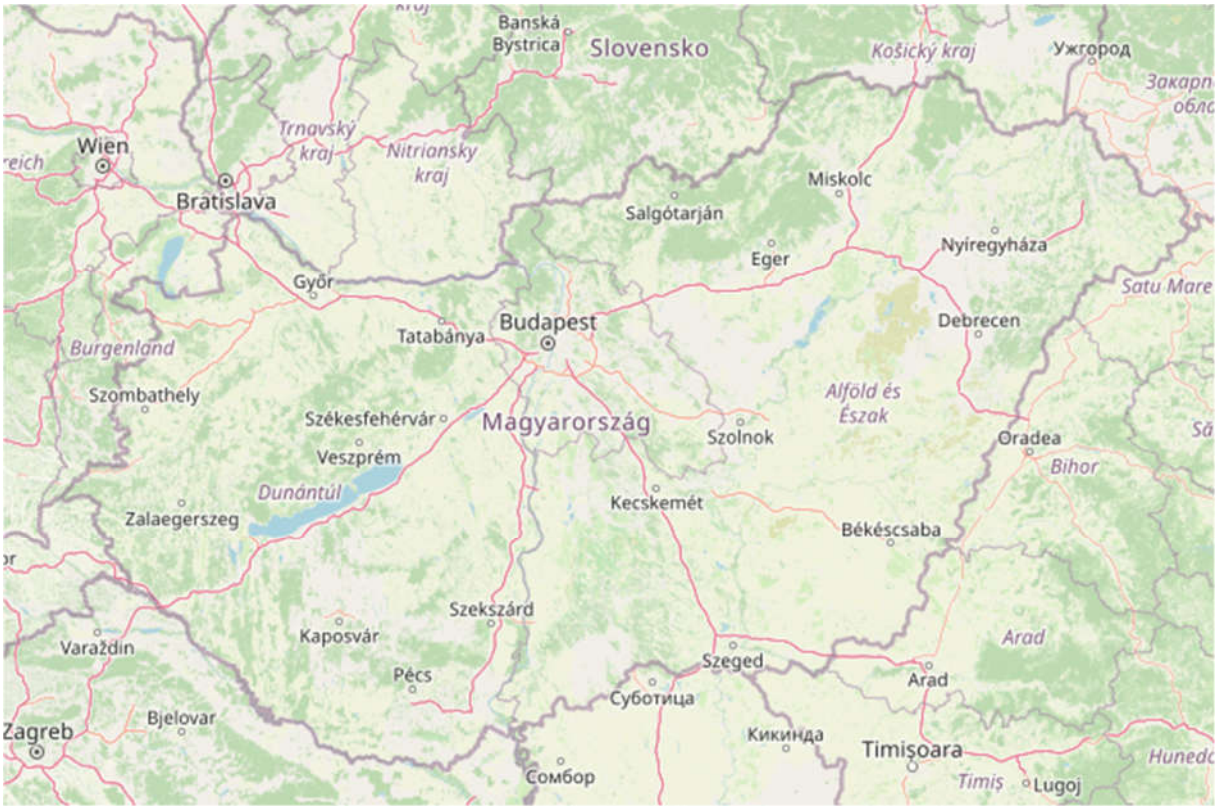
3-6. ábra: Az új gomb elhelyezése a térképen, valamint a kezdőképernyőn megjelenő ikonja [9].

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:

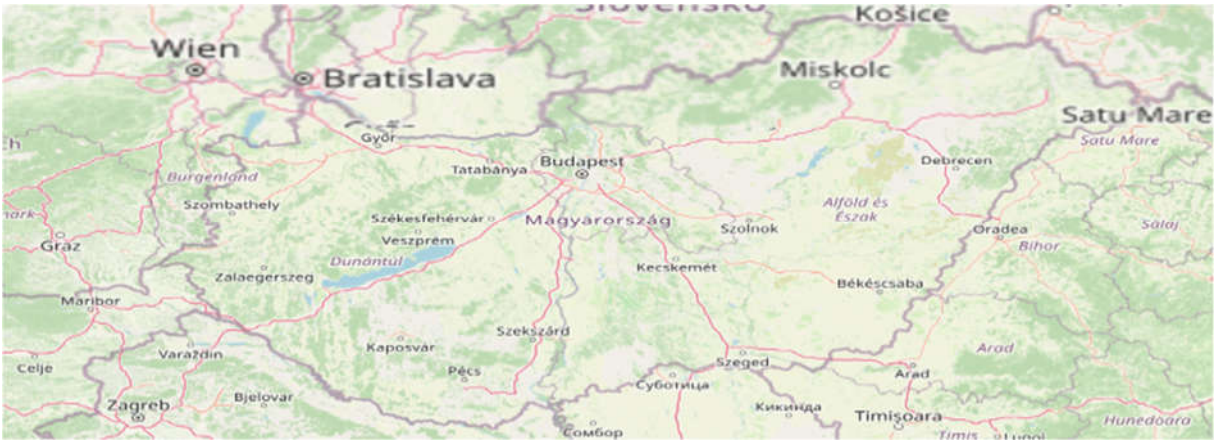
```
PMpoint = ol.proj.transform(evt.coordinate,WebM,WGS84)
```

3-7. ábra: Vetületi rendszer transzformációjának végrehajtása OpenLayers-ben

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.



3-8. ábra: Magyarország Web Mercator vetületben.



3-9. 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')
}
};
```

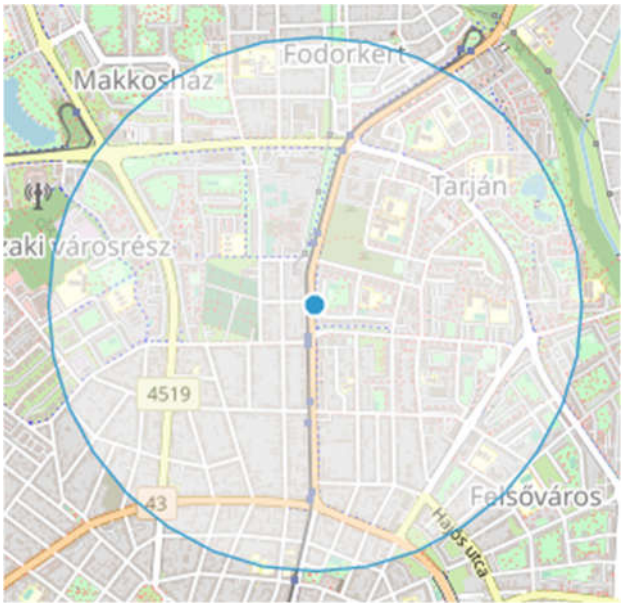
3-10. ábra: Az if-else megoldás a programban

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*).

3-11. 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

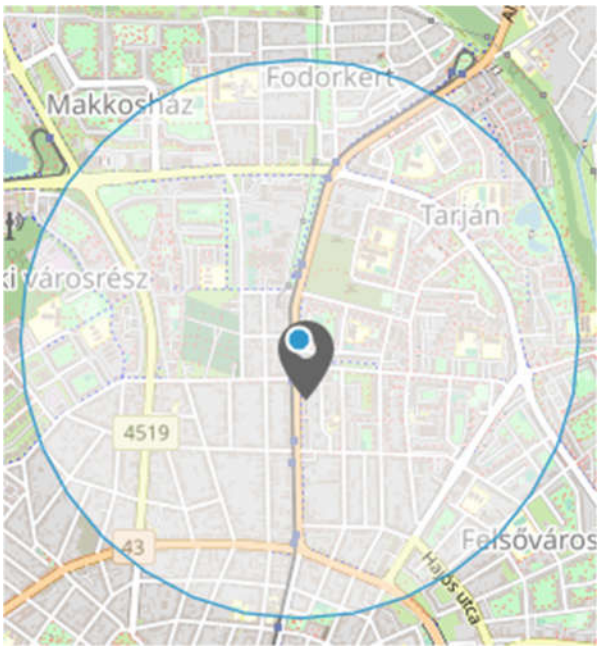
Az előző alfejezetben már említésre került geolokáció, amely egy általános információs koncepció, mely fizikai tárgyak vagy emberek földrajzi helyzetének meghatározását jelenti, leggyakrabban az eszköz GPS- vagy IP-címét használva [12]. Ez a gyakorlatban számunkra azt fogja jelenteni, hogy amikor a térképen 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*).



3-12. 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*).

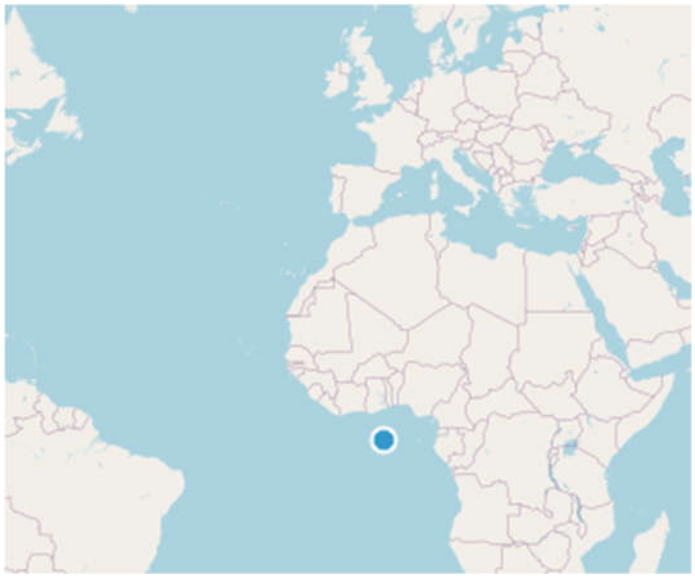
3-13. ábra: Az Ip címből adódó rossz helyzetmeghatározás, és a valós helyzet



3-14. 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].

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.



3-15. ábra: Példa a rossz vetület alkalmazására. A felhasználó WGS84 rendszerbeli koordinátái, elhelyezve a Web Mercator rendszerben készült térképünkön.

3.2. Web Map Service (WMS)

3.2.1. GeoServer

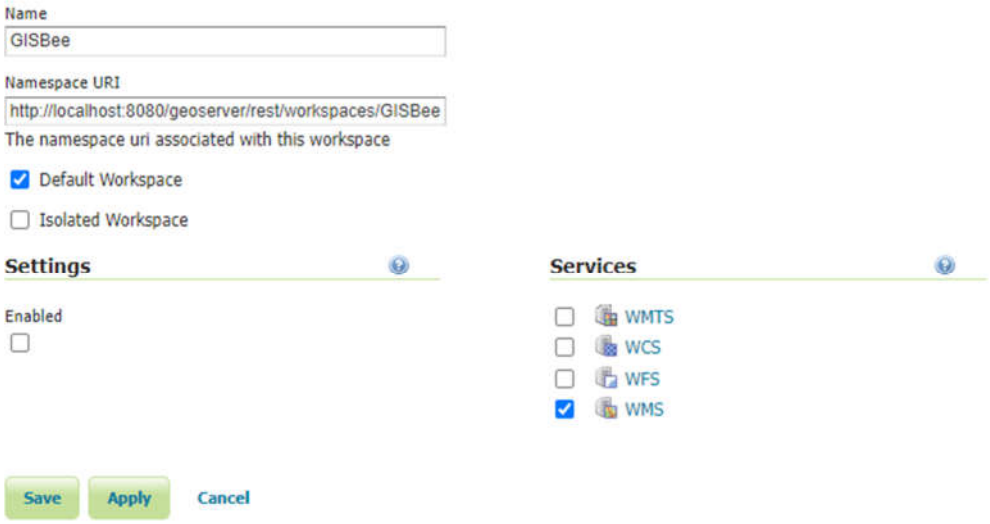
A rendszer fejlesztéséhez szükségem volt egy „térképszerverre” amellyel a későbbiekben a WMS szolgáltatást tudok nyújtani, mellyel az adatokat meg tudom jeleníteni a térképen. A WMS egy, az OGC (Open Geospatial Consortium) [13] által fejlesztett szabvány, amely a georeferált térképek interneten keresztül történő kiszolgálását hajtja végre. Több térképszerver közül is választhattam, ugyanis ezek többsége nyílt forráskódú (ilyen például a QGIS server, vagy MapServer, de fizetős szerverre is van példa, melyek közül a legismertebb az ArcGIS Server), választásom a GeoServer-re esett [14], amely az Open Source Geospatial Foundation (OSGeo) [15] egyik projektje (többek között az OpenLayers és a PostGIS-el együtt).

3-16. ábra GeoServer logója

A GeoServer telepítése egyszerű, nem igényel komolyabb informatikai tudást. Először a 2.20-as verziót használtam, de későbbiekben szükségem volt egy újabb verzióra, ezért frissítettem 2.21.2-es verzióra. Telepítés után 4 db ikon jelenik meg a Windows Start menüben, egy súgó, egy szerver indító és leállító, valamint a web portál gyors elérési ikonja (*3-17. ábra*).

3-17. ábra: A GeoServer Windows Start menüjéből elérhető ikonok

Miután elindítottuk a GeoServer-t és megnyitottuk a web portált megjelenik előttünk a felhasználói felület. Első lépésként be kell jelentkeznünk, majd ezt követően elérhetővé válik számunkra a bal oldalon található menü. A projekt szempontjából számunkra a legfontosabb a második fül alatt található menüpontok, sorrendben „Layer Preview” (előnézeteket tudunk lekérni a szerverben található rétegekről), „Workspaces” (munkaterületek), „Stores” (az adatforrást tudjuk itt beállítani), „Layers” (rétegek), „Layer Groups” (réteg csoportok) és végül „Styles” (stílusok). Kezdjük a Workspace létrehozásával, amelynél egyszerű dolgunk van, csak egy nevet és egy URI nevet kell neki adnunk, amelyek megegyezhetnek, a mi esetünkben mind a kettő a GISBee nevet kapta. Ezután beállíthatjuk a munkaterülettel kívánt szolgáltatás típusát mely lehet ’Web Map Tile Service’ (WMTS), ’Web Coverage Service’ (WCS), ’Web Feature Service’ (WFS), valamint számunkra a legfontosabb ’Web Map Service’ (WMS). Amennyiben üresen hagyjuk mindegyik mezőt, úgy mindegyik szolgáltatás elérhető a munkaterületről. Utolsó lépésként beállítottam, hogy a munkaterület legyen a „Default Workspace” (alapértelmezett munkaterület) (*3-18. ábra*).

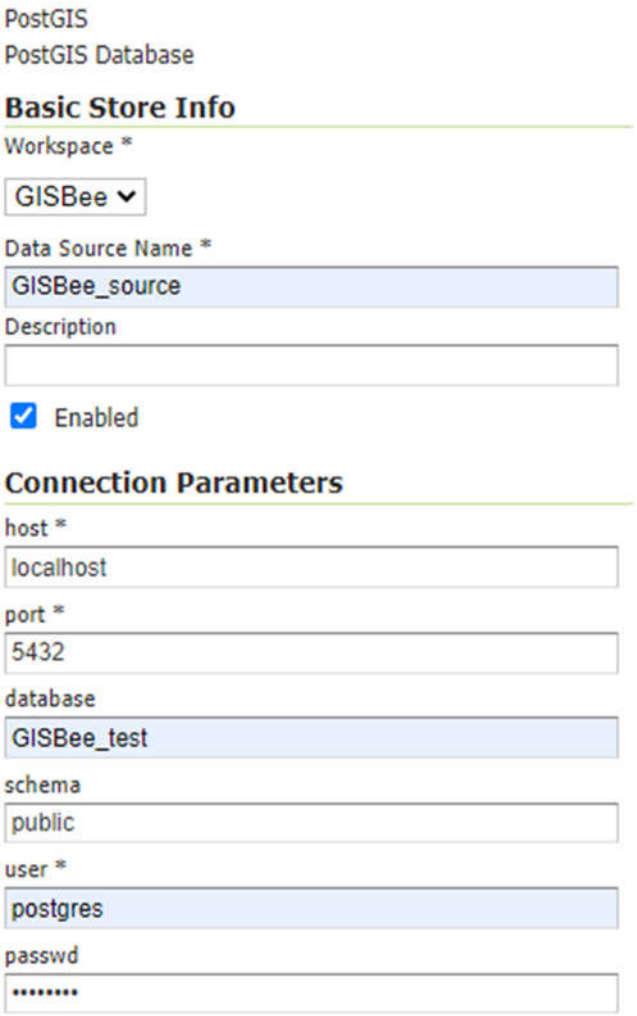


3-18. ábra: Munkaterület (Workspace) beállítása

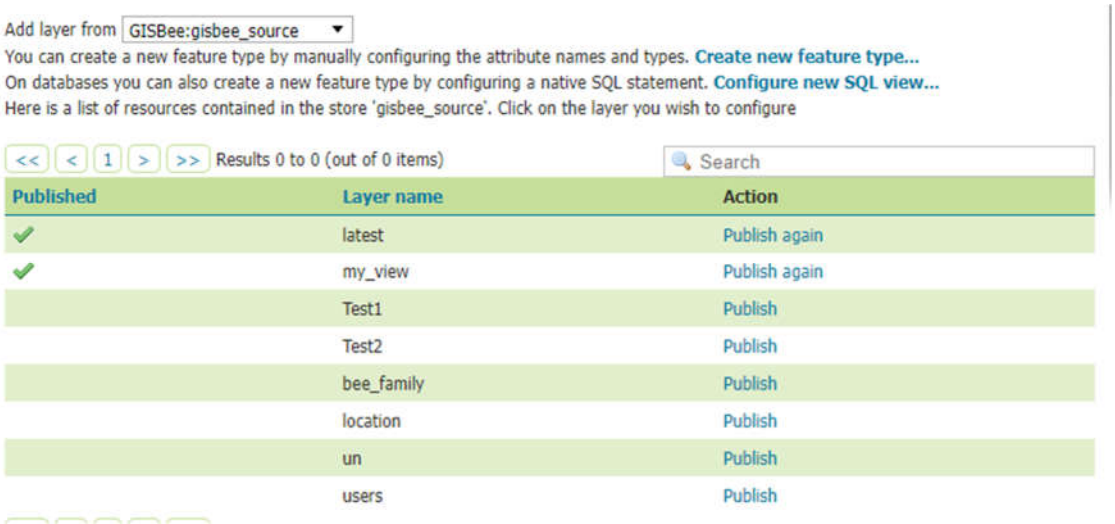
A „Stores" létrehozása már kissé bonyolultabb feladat. Miután kiválasztjuk az „új tárhely hozzáadása" funkciót (Add new Store) ki kell választanunk, hogy milyen fajta adatot kívánunk hozzáadni. Lehetőségünk van vektor, raszter és WMS, vagy WMTS adat hozzáadására. Az én esetemben a vektor adatokon belül a PostGIS adatbázist kell kiválasztanom adatforrásnak. A megjelenő felületen ki kell választani, hogy melyik munkaterületben kívánjuk elhelyezni az adatforrást, majd el kell nevezni és ezután be kell állítani a csatlakozási paramétereket. A csatlakozási paramétereknél beállítjuk a 'host'-ot, a PostGIS 'port'-ját, kiválasztjuk, hogy a PostGIS-en belül melyik adatbázisra csatlakozzon, beállítjuk a sémát, majd a korábban létrehozott PostGIS felhasználó (amely admin joggal rendelkezik az PostGIS-ben tárolt adatbázisunk felett) felhasználónevével és jelszójával hozzáadjuk az adatbázist. Az egyéb paraméterek beállítása nem szükséges (*3-19. ábra*). A feladat végrehajtása közben természetesen a PostGIS adatbázisnak folyamatosan futnia kell a háttérben. A későbbiekben, ha a PostGIS a WMS szolgáltatás közben nem fut, akkor a GeoServer mindig a legutolsó állapotot továbbítja a felhasználó számára, így, ha az adatunk nem dinamikusan változó adat, a PostGIS-re nincs szükségünk a háttérben, azonban a GeoServer futása nélkül az OpenLayers nem képes adatokat megjeleníteni a térképen (mivel nem jön létre WMS).

A munkaterület és az adatforrás hozzáadása után a rétegek hozzáadása következik (a rendszer fejlesztéséhez egyelőre csak egy rétegről van szó). Egy újabb réteg hozzáadásához ki kell választani az „Add a new layer" parancsot, majd a megjelenő

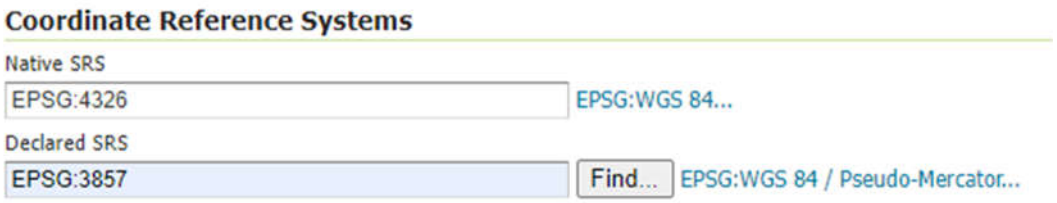
felületen ki kell választanunk az adatforrást. Ezután megjelennek az adatforrásban található rétegek, melyek lehetnek a 'table', vagy 'view' formában is az adatbázisunkban. A megjelenő rétegek közül kiválasztjuk a publikálni kívánt réteget és rákattintunk a „Publish" feliratra (*3-20. ábra*). Az új réteg publikálása előtt be kell állítanunk a térbeli referenciarendszert (SRS) amit EPSG-vel tudunk megadni (csak a publikálandó adat SRS értékét lehet változtatni, a forrás adat SRS értékét nem) (*3-21. ábra*). Végül be kell állítani a „vetületi határokat". A vetületi határok beállítása lehetővé teszi, hogy igény esetén a publikálni kívánt térbeli adatainknak csak egy részét publikáljuk, amennyiben azonban az egészet publikálni kívánjuk érdemes a „Compute from SRS bounds" opciót választani, így az összes adat be fog kerülni a rétegünkbe (*3-22. ábra*). A beállítások után megnyomjuk a mentés gombot a rétegünk megjelenik a listában.



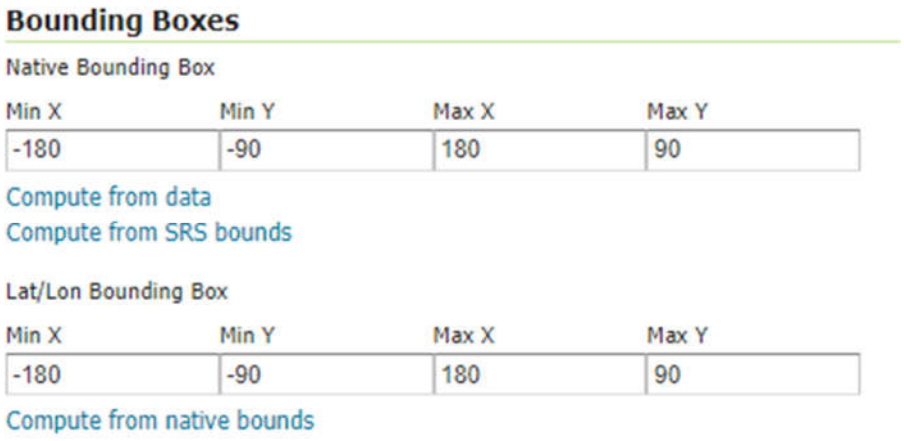
3-19. ábra: PostGIS adabázis hozzáadása a GeoServer rendszeréhez



3-20. ábra: Új réteg publikálása a PostGIS adatforrásunkból



3-21. ábra: Forrás és publikálandó réteg térbeli referenciarendszerének beállítása GeoServer-ben.



3-22. ábra: Vetületről megjeleníteni kívánt terület határainak beállítása a GeoServer felületén

3.2.2. Cross-layer filtering

A rendszer szempontjából fontos kritérium, hogy a méhészek, csak a saját adataikat láthassák, ezáltal a rendszert valamivel biztonságosabbá tudjuk tenni. Erre a problémára jelent megoldást a GeoServer Cross-layer filtering kiegészítője [16]. A

GeoServer alapértelmezett helyzetben is képes egyszerűbb lekérdezések végrehajtására [14], azonban az összetettem lekérdezésekhez már elengedhetetlen a kiegészítő telepítése. A kiegészítő telepítésénél, a letöltés előtt fontos megbizonyosodni arról, hogy a telepíteni kívánt kiegészítő verzió száma teljesen megegyezik a GeoServer általunk használt verziószámával, egyéb esetben nem fog működni (esetemben a 2.21.2 verziószámmal ellátott kliens kiegészítőjére volt szükségem). A letöltött fájlt (gs-querylayer-2.21.2.jar) elhelyeztem a GeoServer ”WEB-INF/lib” könyvtárába és a GeoServer újraindítását követően a kiegészítő már működött. Hogy ellenőrizzem, valóban sikeres volt a kiegészítő telepítése, a GeoServer ”Service Capabilities” menüpontjából megnyitottam a ’WFS 1.1.0’ elnevezésű XML fájlt. Sikeres telepítés esetén az XML fájlban a funkcióknál a ’Query’ mellet meg kellet jelennie a ’QueryCollection’ és a ’QuerySingle’ funkciónak is (*3-23. ábra*).

```
<ogc:FunctionName nArgs="2">pow</ogc:FunctionName>
<ogc:FunctionName nArgs="1">property</ogc:FunctionName>
<ogc:FunctionName nArgs="1">PropertyExists</ogc:FunctionName>
<ogc:FunctionName nArgs="-2">Quantile</ogc:FunctionName>
<ogc:FunctionName nArgs="-1">Query</ogc:FunctionName>
<ogc:FunctionName nArgs="-1">queryCollection</ogc:FunctionName>
<ogc:FunctionName nArgs="-1">querySingle</ogc:FunctionName>
<ogc:FunctionName nArgs="0">random</ogc:FunctionName>
<ogc:FunctionName nArgs="-1">RangeLookup</ogc:FunctionName>
<ogc:FunctionName nArgs="-1">RasterAsPointCollection</ogc:FunctionName>
<ogc:FunctionName nArgs="-2">RasterZonalStatistics</ogc:FunctionName>
<ogc:FunctionName nArgs="-6">RasterZonalStatistics2</ogc:FunctionName>
<ogc:FunctionName nArgs="5">Recode</ogc:FunctionName>
```

3-23. ábra: Részlet a GeoServer funkcióinak listájából. A 'Cross-layer filtering' kiegészítő sikeres telepítése után megjelent a 'queryCollection' és a 'querySingle' funkció is, a 'Query' funkció mellett.

A kiegészítő telepítésével az új funkciókon túl egyéb stílus elemek is telepítésre kerülnek, melyek a megjelenítést kívánják elősegíteni. Az új stílusok között fellelhetőek raszter, vonalas, felület és pontszerű adatok ábrázolására szolgáló stílus elemek is.

Amikor adatot akarunk lekérdezni a WMS szolgáltatásból CQL (Common Query Language) filtert kell alkalmazni. A filterezés történhet a GeoServer web portálján is, de a dinamikus filterezés miatt célszerű a feladatot az OpenLayers felületén végrehajtani. Fontos megemlíteni azonban, hogy a létrehozott WMS szolgáltatásban a CQL filter megjelenik [16] és ezáltal egy hozzá értő személy, egy URL dekódoló segítségével vissza tudja keresni (*3-24. ábra*).

```
http://localhost:8080/geoserver/GISBee/wms?
SERVICE=WMS&VERSION=1.3.0&REQUEST=GetMap&FORMAT=image/png&TRANSPARENT=true&LAYERS=GISBee:latest&TILED=
true&CQL_FILTER=b_id=10&WIDTH=256&HEIGHT=256&CRS=EPSG:3857&STYLES=&BBOX=2191602.4749925733,5792092.255
337516,2269873.991956594,5870363.772301536
```

3-24. ábra: dekódolt WMS URL, a CQL filternél kiemelve.

A probléma véleményem szerint nem okoz komolyabb gondot, mivel, sok adat van az adatbázisban, ezért bár meglehet, hogy a felhasználó olyan adatokat bírjon lekérdezni, amelyhez nincs jogosultsága, de csak véletlenszerűen tud egyéb adatok ID értéke alapján lekérdezni (a felhasználók a saját adataik ID értékét sem ismerik). Egyéb ok, amely miatt a kockázat véleményem szerint elenyésző, hogy a méhészek nem informatikusok, ezért kis esélyt látok rá, hogy a webhely hálózati vizsgálatához fognának. Természetesen a problémára a GeoServer rendelkezik megoldásokkal, jelenleg a legígéretesebbnek a „Parameters Extractor module” tűnik, mely úgyszintén egy telepíthető modul [17], de a szakdolgozat elkészültéig sajnos nem volt lehetőségem kipróbálni.

3.2.3. Megjelenítés a térképen

A térképen való megjelenítéshez a GeoServer-ről a korábban említett WMS szolgáltatást kell továbbítanunk az OpenLayers felé. Az OpenLayers kódsorában létre kellet hoznom egy forrást. A forrást ’TileWMS’ forrásként hoztam létre, amely azt eredményezi, hogy a térképen megjelenített adatok a térképi ’Zoom Level’-hez viszonyítva változtatják a méretüket, tehát a gyakorlatban, a felhasználó mindig ugyanakkora méretben látja az objektum jelkulcsát.

```
89  // GEOSERVER Layer
90  var geoserverLayerSource = new ol.source.TileWMS({
91    url:'http://localhost:8080/geoserver/GISBee/wms',
92    params: {
93      'LAYERS':'GISBee:latest',
94      'TILED': true,
95      'CQL_FILTER':Filter
96      },
97    serverType: 'geoserver',
98  })
99
100 var geoserverLayer = new ol.layer.Tile({
101   source: geoserverLayerSource
102 })
```

3-25. ábra: A GeoServer által szolgáltatott WMS hozzáadása rétegként az OpenLayers webtérképükhöz.

A 3-25. ábrát kifejtve jobban megérthetjük, hogy működik a WMS szolgáltatás az OpenLayers térképekben. Az forrásnak szüksége van egy URL forrásra (a teszteléshez használt URL a saját számítógépemen futó webszerver, az úgynevezett „localhost”). Az URL forrásban megjelenik a munkaterület megnevezése mellet a szolgáltatás típusa a WMS is. Az URL-t legkönnyebben a GeoServer webportáljáról tudjuk kimásolni, ha a munkaterület valamelyik rétegéről készítünk egy előnézetet. Az előnézet ilyenkor mindig egy új lapon jelenik meg, és egész egyszerűen az URL elejét kimásoljuk és beillesztjük kódunkba. A következő elemként a paramétereket kell beállítani. A paramétereknél két darab paraméter beállítása kötelező, a ’Layers’, mely a réteget (vagy igény szerint több réteget) biztosítja, valamint a ’TILED’ beállítása, mely lehet igaz (true), vagy hamis (false). A ’Layers’ paraméter megadásánál meg kell adnunk a munkaterület nevét valamint a megosztani kívánt réteg nevét is az alábbi formátumban: ’GISBee:latest’, melynél a ’GISBee’ a munkaterület nevére utal, a ’latest’ pedig a rétegre. A ’TILED’ beállítás igazra állításával, érhetjük el, hogy a megjelenített jelkulcsok a ’Zoom Level’ változásával egyidejűleg változzon. Egyéb, nem kötelező paraméter beállítások közé tartozik CQL filter beállítása. A CQL filter beállításánál közvetlenül megadhatjuk a lekérdezés értékét, vagy megadhatunk változót is. Az általam készített projektnél a változó megadása jelenti a megoldást, ugyan is a későbbiekben ezzel kívánjuk elérni, hogy a felhasználó bejelentkezésekor a térképén a saját méhállományait láthassa csak. Ez a gyakorlatban azt jelenti, hogy a GeoServer által szolgáltatott WMS egy réteget tartalmaz és nem kell minden egyes felhasználónak újabb réteget létrehozni a megjelenítés érdekében, hanem a felhasználó profiljával filterezni tudjuk a számára megjelenő adatokat. A WMS forrás utolsó elemeként meg kell adunk, hogy milyen térképszerver típust alkalmazunk, amely esetemben ’geoserver’ értéket kapott (*3-25. ábra*) [11].

A forrás beállítása után egy réteget hoztam létre. Mivel a forrás TileWMS, Tile réteget kell létrehoznunk, majd hozzá kell kapcsolnunk a forrást (*3-25. ábra*). Miután ezzel is elkészültem, a rétegek listájához hozzáadtam a GeoServer rétegemet és a WMS sikeresen megjelent a térképemen (*3-26. ábra*).

```
113    // Layer Arrey
114    var layerArray = [OSM,geoserverLayer,markerLayer]
```

3-26. ábra: A GeoServer rétegének hozzáadása a rétegek listájához

3.3. A 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] (*3-27. 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);
```

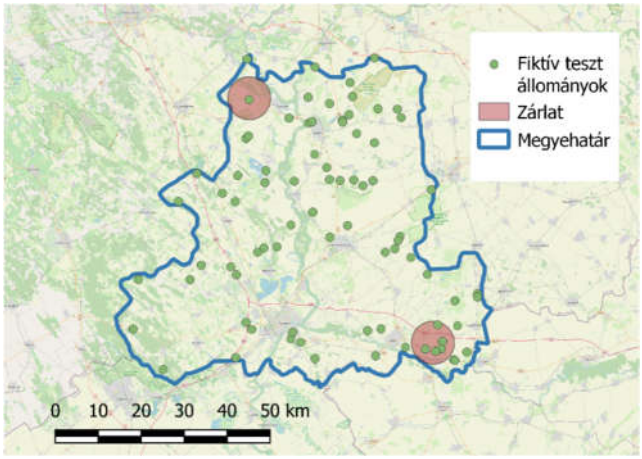
3-27. ábra: A PostGIS-ben elkészített lekérdezés, már adatbázisba beépítve.

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 foglalható 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.

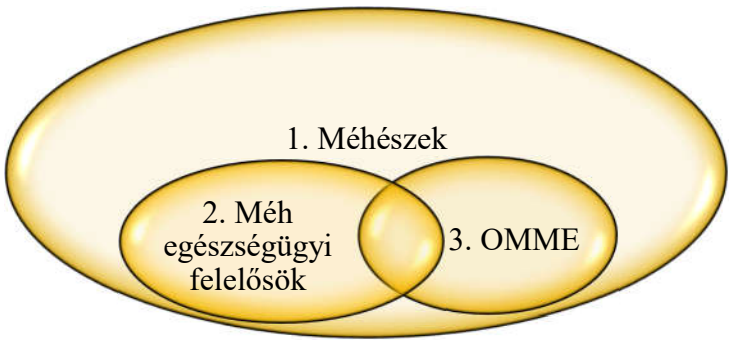
Amikor a méhész vándorlást tervez, továbbra is ki kell majd látogatnia a méhlegelőre, azonban könnyebben tud döntést hozni, ha a lekérdezés után látja, hogy hány méhcsalád van jelenleg is a területen, valamint, hogy hány méhcsaláddal terveznek a területre vándorolni. Helyszínen el tudja dönteni, hogy 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 (*3-28. ábra*).



3-28. 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.

3.4. A GIS további fejlesztési lehetőségei

A kész rendszer az elképzelés szerint 3 különböző felhasználót különböztet meg. Elsődleges csoport a méhészek csoportja, akik bejelentéseket tudnak kezelni, valamint térképi lekérdezéseket tudnak végrehajtani, az állományuk kezelése mellet. A második csoportnak a méhegészségügyi felelősöket tudjuk tekinteni, az ő feladatuk, hogy az állományok egészségügyi adatait tudják szerkeszteni. A harmadik csoport az OMME (Országos Magyar Méhészeti Egyesület) [18]. Az OMME felhasználók sokkal több adatok tudnak lekérdezni a rendszertől, mint az egyéb felhasználók. A következő alfejezetekben részletezésre kerül mindegyik csoport, valamint elképzelt megoldásaik.



3-29. ábra: A felhasználói szintek felépülése.

3.4.1. Méhészek (1. csoport)

A rendszer minden felhasználója méhész, így ezekkel a jogokkal minden felhasználó rendelkezik. A jelenleg elkészült rendszerben a méhészek képesek bejelentéseket tenni, valamint szerkeszteni a meglévő állományaikat, és adott területekről lekérdezéseket végezni. Jelenlegi szintjén a lekérdezés azt jelenti, hogy a felhasználó kattintással, vagy geolokációval mért koordináta segítségével, 5 km sugarú körben információkat biztosít a méhész számára arról, hogy hány méhcsalád tartózkodik jelenleg is a területen, továbbá, hogy hány méhcsalád rendelkezik ott foglalóhellyel. Ez egy bizonyos szinten már elősegíti a vándorlástervezés folyamatát, azonban a későbbiekben a várható virágzással és a lekérdezett méhcsaládok egészségügyi statisztikáival szeretnék hozzájárulni a folyamatokhoz.

A várható virágzáshoz egy összetett GIS adatbázisra van szükség, mely tartalmazza az ország bizonyos területein meghonosodott mézelő növénykultúrákat. A Szegedi Tudomány Egyetem Geoinformatikai, Természet- és Környezetföldrajzi Tanszéke által készített „Inváziós Növényfajok Országos Térinformatikai Adatbázisa” elnevezésű adatbázisban megtalálható összes növénykultúra jelentős mézelő növény, listába szedve [19]:

- Fehér akác (Robinia pseudoacacia)
- Gyalog akác (Amorpha fruticosa)
- Aranyvessző (gyakran szolidágó néven ismerik) (Solidago spp.)
- Keskenylevelű ezüstfa (Elaeagnus angustifolia)
- Mirigyes bálványfa, vagy bálványfa (bár a keserű ernyedt ízű méze miatt sok méhész kerüli) (Ailanthus altissima)
- Selyemkóró (ismert még selyemfű, tejelőkóró, vaddohány és papagájvirág néven is) (Asclepias syriaca)

Érdekességként megemlíteném, hogy a weboldal, melyen az adatbázis található, a természetvédelem legnagyobb kihívásaként hivatkozik a növényekre, és mindegyikről negatívumokat sorol fel, azonban a méhészeti és egyéb hasznukról nem ejt szót.

A méhlegelőkről információkat saját rendszerrel is gyűjthetünk, ha a távérzékelés eszköztárát próbáljuk meg kihasználni. Véleményem szerint a legcélravezetőbb az adott

növények virágzásának időpontjában történő műholdas földmegfigyelés lehet, természetesen ezzel a technológiával, csak a következő évre lehet előre jelzéseket tenni.

Az egészségügyi statisztikák, mellyel a méhészek döntési folyamatát kívánom elősegíteni, a méhegészségügyi felelős bejegyzésein fog alapulni (*3.4.2. alfejezet*).

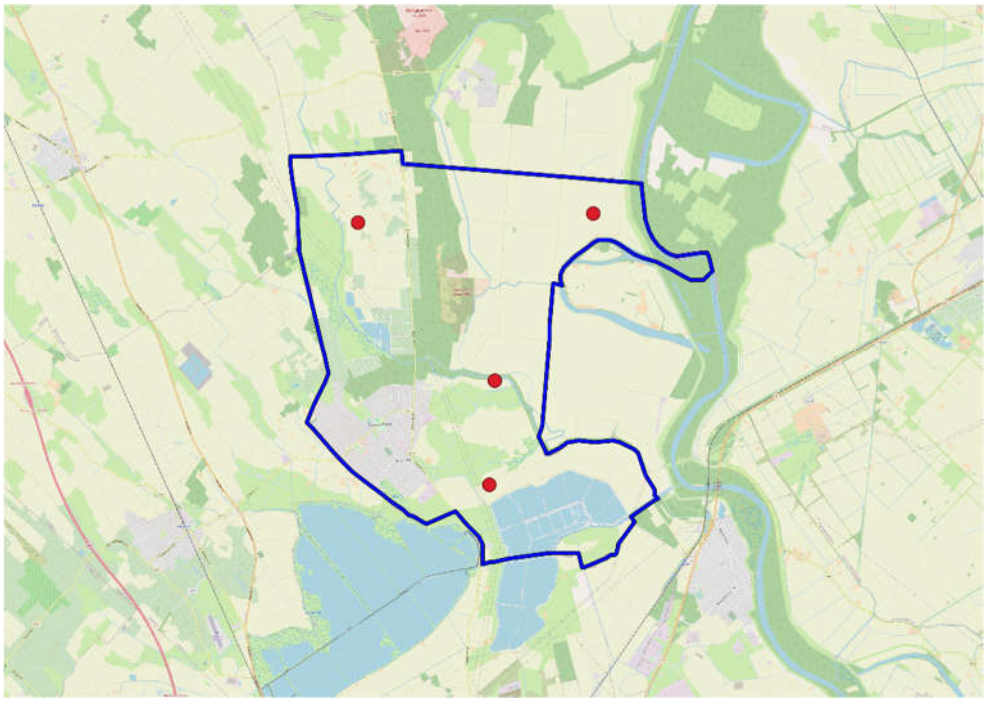
3.4.2. Méhegészségügyi felelősök (2. csoport)

A rendszer egyik legfontosabb eleme, az állategészségügyi hozzájárulás, mellyel a jövőben kívánok részletesebben foglalkozni. A méhek állategészségügyi kezelése sok mindenben eltér, mint a legtöbb állattartás esetén, mivel a méheket nem lehet bezárni. Ez a gyakorlatban azt jelenti, hogy a fertőzések is könnyedén tudnak terjedni a méhcsaládok között. Bizonyos fertőzés esetén (mézelőméhek nyúlós költésrothadása) a fertőzött állomány körüli 5 km-es terület zárlati zónává minősül. A méhegészségügyi felelős olyan méhész, aki szakmai tapasztalata alapján kellő tudással rendelkezik, valamint a munkakörére vonatkozó jogszabálynak megfelelően képes eljárni [2]. Minden településhez tartozik egy, vagy több méhegészségügyi felelős (méhcsaládszámtól függően), illetve egy felelőshöz több település is tartozhat, ha azokon a településeken kevesebb a méhcsaládok száma. A méhegészségügyi felelős területét poligonokkal kívánom megoldani, amelyhez a települések külterületi határait kívánom egyetlen rétegben elhelyezni (*3-30. ábra*) és a korábban említett (*3.2.2. alfejezet*) filterezéssel, a felelősökhöz rendelni. Minden méhészet rendelkezik méhegészségügyi kiskönyvvel, melyet elsősorban a méhegészségügyi felelős vezet, a hatósági állatorvos közreműködésével.

Sokszor a méhészek számára nehézséget okoz eldönteni, hogy esetlegesen ők beleesnek-e a zárlati zónába. A zárlati zóna azt jelenti, hogy a területre sem bevándorolni, sem pedig a területet elhagyni a méhállományoknak nem lehet, a fertőzött állományt pedig el kell pusztítani és minden méhészeti eszközt hőkezelni kell, a fa eszközöket és lépeket pedig el kell égetni. Annak érdekében, hogy a méhész könnyedén megbizonyosodjon róla, hogy állománya nem tartozik egyetlen zárlat alá sem, az alkalmazásban található térkép minden újabb zárlatot ábrázolni fog kör segítségével. A gyakorlati életben sajnos előfordul olyan eset is, amikor némely méhész megpróbál kibújni a zárlati kötelezettség alól, oly módon, hogy hírt kap a zárlatról, még a zárlat

hivatalossá tétele előtt és állományaival elvándorol. A rendszerben erre az esetre is próbáltam megoldást kidolgozni. A hivatalosan bejelentett zárlat időpontját megelőző két hétben a területen tartózkodó minden állomány 'potenciális veszélyforrás' kategóriába fog kerülni, ami azzal jár, hogy az új bejelentkezésnél illetékes egészségügyi felelőst a rendszer értesíteni fogja, ezáltal a felelős ellenőrizheti az állományt, ezáltal elősegítheti a fertőzés terjedésének visszaszorítását.

Másik fontos elem, szintén a döntéstámogató szerep. Ha a méhészek már a vándorlást megelőzően tisztában vannak azzal, hogy melyek azok a területek, ahol fertőzés van, véletlenül sem fognak zárlat alá vonulni, és ezáltal zárlat alá kerülni.



3-30. ábra: Péda a méhegészségügyi felelős területére, és a benne található állományokra (az ábrán fiktív állományok találhatóak).

3.4.3. OMME felhasználók

Az Országos Magyar Méhészeti Egyesület (OMME) a méhészek szakmaközi szervezete. A magyar méhészetben történő szerepvállalása elengedhetetlen, a pályázatok, hazai és nemzetközi konferenciák szervezése, szakmai tanácsadás, érdekképviselet és valamennyi OMME által ellátott feladat miatt. Elenyésző kivétellel minden méhész OMME tag.

A megtervezett alkalmazás rendelkezni fog OMME felhasználói felülettel, mely jelen esetben az OMME vezetőinek lesz megtervezve. A felület elsődlegesen azt szolgálná, hogy a méhésztársadalom vezetői, országosan képesek legyenek időben és térben elemezni a magyar méhészetet. Terveim szerint, ez úgy valósul meg, hogy a számukra szolgáltatott WMS nem rendelkezik filterezéssel, ezáltal az ország miden pontján képesek lesznek látni az összes méhállományt, valamint keresővel rá is tudnának keresni egy-egy állományra, és az állományról adatokat tudnak lekérdezni. Az elemzéseket segíteni, hogy időpont szerint is lehetőségük lenne kiválasztani, hogy mely dátum az, amely az elemzéseik szempontjából őket leginkább érdekelné. A rendszer továbbá elősegíteni, hogy a nem bejelentett, hobbi méhészeteket könnyebben ki lehessen szűrni, ezáltal a potenciális fertőzési gócpontok könnyebben visszaszoríthatóak lehetnek.

Természetesen az összes felhasználó felületét és jogosultságait úgy kell kialakítani, hogy a GDPR (általános adatvédelmi rendelet) és más jogszabályoknak megfeleljen [20].

3.5. 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 [21].

A rendszer Spring Boot keretrendszert használ, amely a Spring-keretrendszer kiterjesztéseként elkészült modul [21] [22]. 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 [23].

A kód strukturális elrendezéséhez ’Controller-Service-Repository’ mintát [24] 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 [25].

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 [26].

3.6. 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.

3.6.1. 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 [27] 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 (*3-31. ábra*). Mivel jelenleg weboldalt fejlesztettünk így egyelőre csak a CSS kódokra volt szükségünk, későbbiekben azonban jobban ki tudjuk használni a rendszer nyújtotta lehetőségeket.

A prototípustervezés azt jelenti, hogy a fejlesztő már a dizájnnál megtervezheti, hogy melyik gomb hova vezesse át a későbbi felhasználót (*3-34. ábra*), valamint, animációkat is beletervezhet, melyeket egy próba módban (*3-32. ábra*) lefuttathat. Az alkalmazás fejlesztéséhez azonban ezeket a fejlesztőnek önállóan kell lefejlesztenie, mivel a Figma rendszeréből ezeket nem lehet kimenteni.

```
var view = UILabel()
view.frame = CGRect(x: 0, y: 0,
  width: 360, height: 144)
view.backgroundColor = .white

view.layer.backgroundColor =
  UIColor(red: 0.98, green: 1,
  blue: 0.012, alpha:
  1).cgColor

var parent = self.view!
parent.addSubview(view)
view.translatesAutoresizingMask
  IntoConstraints = false
view.widthAnchor.constraint(equ
  alToConstant: 360).isActive =
  true
view.heightAnchor.constraint(eq
  ualToConstant: 144).isActive
  = true
view.leadingAnchor.constraint(e
  qualTo: parent.leadingAnchor,
  constant: 0).isActive = true
view.topAnchor.constraint(equal
  To: parent.topAnchor,
  constant: 0).isActive = true
```

```
/* Rectangle 2 */

position: absolute;
width: 360px;
height: 144px;
left: 0px;
top: 0px;

background: #FAFF03;
```

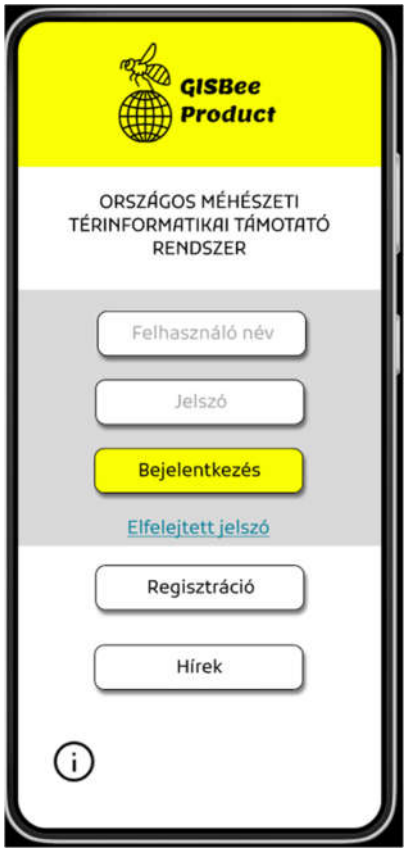
```
<!-- Rectangle 2 -->
<View
 android:id=
 "@+id/rectangle_2"
 android:layout_width=
 "360dp"
 android:layout_height=
 "144dp"
 android:layout_alignParentLef…
 "true"
 android:layout_alignParentTop=
 "true"
 android:layout_marginTop=
 "-24dp"
 android:background=
 "@drawable/rectangle_2"
 />


<!-- drawable/rectangle_2.xml -
->
<vector
 xmlns:android=
 "http://schemas.android.com/a…
 xmlns:aapt=
 "http://schemas.android.com/a…
 android:width=
 "360dp"
 android:height=
 "144dp"
```

3-31. ábra: Példa a Figma által generált kódokra, ugyanarra az elemre. Baloldalon IOS, középen CSS, jobb oldalon Android kódból egy részlet látható.

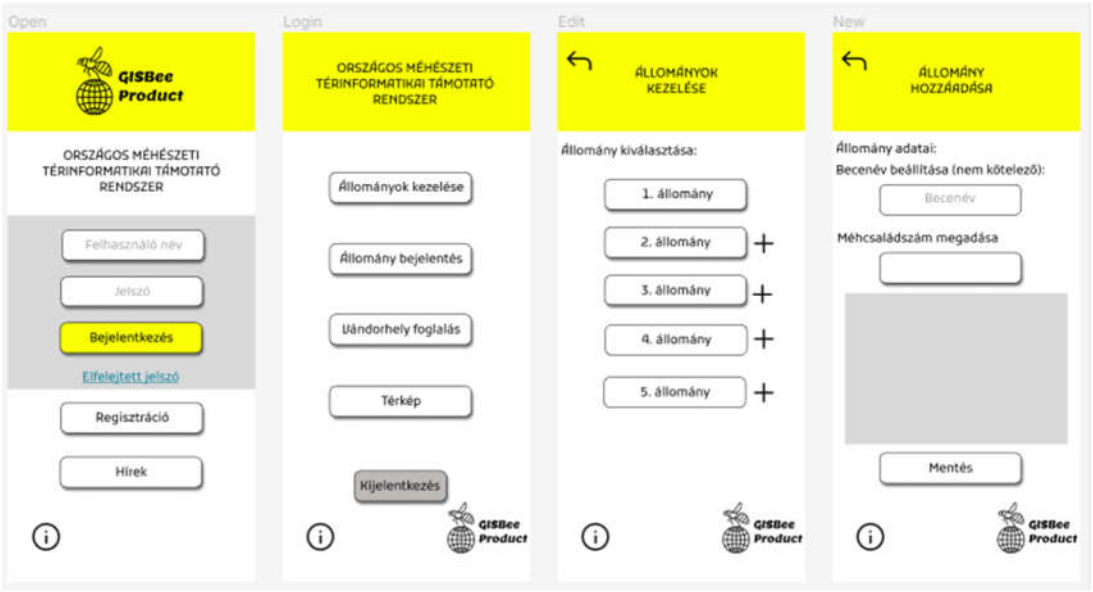
A felhasználónak először a bejelentkező felület fog megjelenni, melyen saját felhasználó nevével, illetve jelszavával bejelentkezhet. Amennyiben nem emlékszik a jelszóra kérhet emlékeztetőt, vagy amennyiben nincs felhasználó profilja regisztrálhat (*3-32. ábra*). Bejelentkezést követően a főmenü felülete jelenik meg, ahol 4 tevékenység közül választhat a felhasználó: ’Állományok kezelése’, ’Állomány bejelentés’, ’Vándorhely foglalás’ és ’Térkép’.

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.



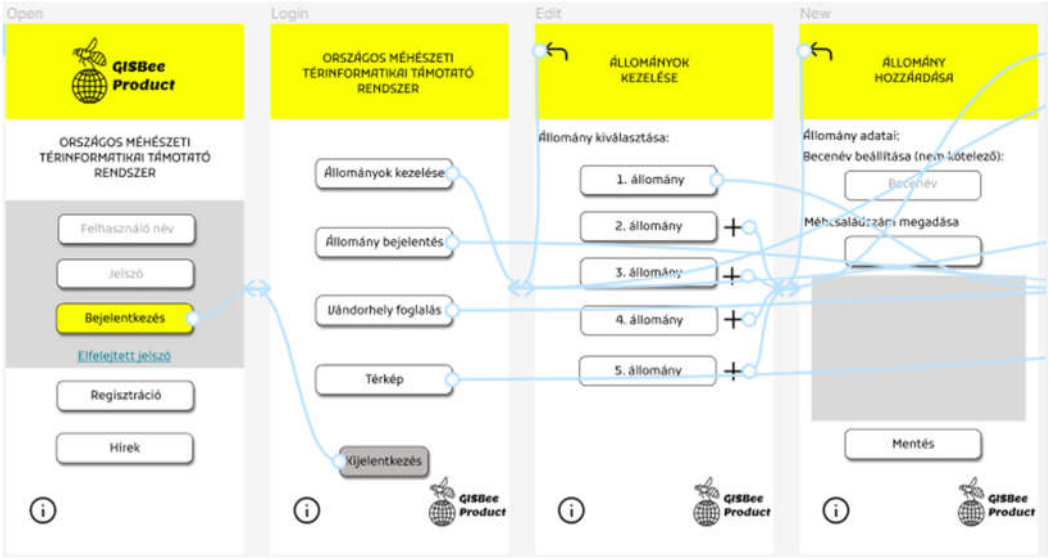
3-32. ábra: A Figma prototípus üzemmódban, az applikáció bejelentkezési felületén

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 (*3.3. 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.



3-33. ábra: Részlet a Figma tervezői felületéből, az alkalmazás első 4 elkészült ablakáról.

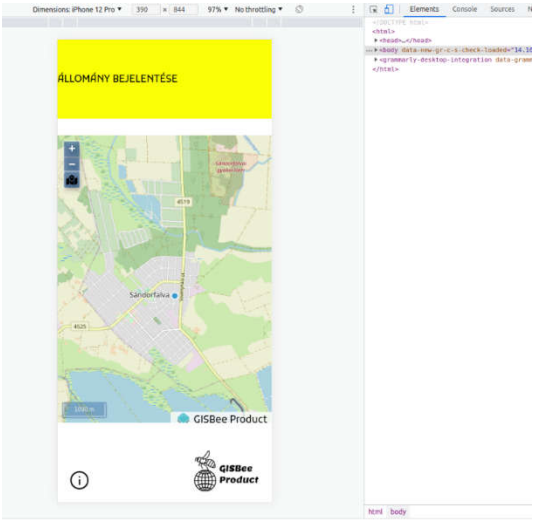
A dizájn megtervezésekor 9 darab ablak került megtervezésre, melyeknél mindig törekedtem a lehető legegyszerűbb megjelenésre. A színek kiválasztásánál olyan színek kombinációját igyekeztem használni, mely véleményem szerint könnyen összeköthető a méhészettel, valamint a rendszerben nincs idegen hatása. 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 (*3-36. ábra*).



3-34. ábra: Az első négy felület, a Figma prototípus fejlesztő felületében. A világoskék nyilak jelzik, hogy melyik gomb lenyomása, hova navigálja a felhasználót.

3.6.2. 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.



3-35. ábra: Részlet az aplikáció fejlesztésből, a benne elhelyezett térképpel.

GISBee Product

3-36. 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 figyelembe, 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 kikértem jövőbeli felhasználó (méhész) tanácsát. Véleményem szerint a rendszer használata könnyen elsajátítható, informatikában kevésbé jártas szakemberek számára is. Sajnos az idő rövidsége miatt nem volt időm 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 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 dolgozat elkészítésénél négy térinformatikai programot alkalmaztam. Az adatoknál PostGIS adatbáziskezelést alkalmaztam, mely a PostgreSQL téradat kiegészítőjével ellátott rendszer. A térkép fejlesztése OpenLayers webtérkép fejlesztő környezetben készült el, és a térképen megjelenő adatok általam készített WMS szolgáltatásból valósult meg. A WMS szolgáltatáshoz egy újabb térinformatikai rendszert, a GeoServer-t alkalmaztam. A tesztadatok QGIS környezetben készültek.

A jövőben megvalósuló fejlesztések közé tartozik a méhegészségügyi és OMME felület elkészítése, valamint a hozzájuk tartozó adatok térképen történő megjelenítése. 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 képes lesz lekérdezni.

5. Summary

Although beekeeping receives less attention than most other participants in the agricultural sector, it is an important part of it, whether it is the pollination of flowers or the part taken out of European honey production. A system of support for beekeepers must therefore be developed to reduce the administrative burden. Since most beekeepers are migratory beekeepers and thus cover large areas of the country, a useful idea would be to operate a geographic information system with a visual element (map). This will make it much easier and more transparent for them to manage the registration process, and will also greatly simplify the selection of migratory sites thanks to the data that will help them to migrate.

During the development of the system, I have repeatedly sought the advice of future users (beekeepers). In my opinion, the system is easy to learn to use, even for less IT-skilled professionals. Unfortunately, due to time limitations, I did not have time to work on all the details, but the map, which is a key element of the system, performs as expected. During the development of the map visualisation, I benefited from my previously acquired knowledge and approach to geospatial information technology, because although, unlike desktop geospatial programs, I had to work entirely from lines of code, the process of building a map followed the same logic as in the case of graphical map editors.

Four geospatial programs were used for this project. For the data I used PostGIS database management, a system with the spatial data add-on PostgreSQL. The map was developed in the OpenLayers web map development library and the data displayed on the map was implemented from a Web Map Service (WMS) that I created. For the WMS I used a another geospatial information system, GeoServer. The test data were produced in the QGIS system.

Future developments include the creation of a bee health and National Hungarian Beekeepers' Association (OMME) interface and the display of the corresponding data on a map. The bee health interface is planned to be built in such a way that each bee health officer's area would form a polygon. When a new migratory site is added, to the polygon, the bee health officer in the area would have online access to the historical health data of

the colony, so that he or she would know what to expect in advance. The bee health officer could also edit the online health booklet that will be implemented in the future, which would be available to all colonies. The OMME user would also be a user with extended rights. His/her rights would include the ability to see the beekeeping colonies in the whole country, to analyse their status in a timely manner and to query national statistics.

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] „OGC,” OGC, [Online]. Available: https://www.ogc.org/. [Hozzáférés dátuma: 11 12 2022].

[14] „GeoServer,” OSGeo, [Online]. Available: https://geoserver.org/. [Hozzáférés dátuma: 11 12 2022].

[15] „OSGeo,” OSGeo, [Online]. Available: https://www.osgeo.org/. [Hozzáférés dátuma: 11 12 2022].

[16] „GeoServer Cross-layer filtering,” OSGeo, [Online]. Available: https://docs.geoserver.org/stable/en/user/extensions/querylayer/index.html. [Hozzáférés dátuma: 11 12 2022].

[17] „GeoServer Parameters Extractor module,” [Online]. Available: https://docs.geoserver.org/stable/en/user/extensions/params-extractor/usage.html. [Hozzáférés dátuma: 11 12 2022].

[18] „OMME,” OMME, [Online]. Available: http://www.omme.hu/. [Hozzáférés dátuma: 11 12 2022].

[19] „Inváziós Növényfajok Országos Térinformatikai Adatbázisa,” SZTE Geoinformatikai, Természet- és Környezetföldrajzi Tanszék, [Online]. Available: http://www.geo.u-szeged.hu/invasive/. [Hozzáférés dátuma: 11 12 2022].

[20] *AZ EURÓPAI PARLAMENT ÉS A TANÁCS 2016. április 27-i (EU) 2016/679 RENDELETE [GDPR],* 2016.

[21] „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].

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

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

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

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

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

[27] „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.