**DogGo – Web**

Kilián Marcell András

**Bevezetés**

Mért választottam a weboldal elkészítését?

A webfejlesztés régóta érdekel, barátaim, ismerőseim megmutatták nekem, hogy mit is sikerült összehozniuk különböző keretrendszerek segítségével. Érdekelt, hogy hogyan kezdődik el egy ilyen projekt megvalósítása, mik a nehézségek egy ilyen oldal elkészítése során, hogyan zajlik a megvalósítás menete, milyen tervekhez igazodnak a tervezők.

Pár hónapja még azt sem tudtam mi az a keretrendszer, annyit tudtam róla, hogy hasznos egy webolal elkészítésében. Elképesztő élmény volt nézni ismerősöket, barátokat, ahogy csinálják a programjukat egy keretrendszer segítségével, kellemes élmény volt számomra, hogy nem értettem a kódjukat, mert még csak ennyire sem láttam, hogy mennyire széleskörű is a programozás. Emiatt szerettem volna egyszer én is érteni egy hasonló programot, amit keretrendszer segítségével hoztam létre.

Nekem ezelőtt nem volt alkalmam, ötletem és nagyon időm sem egy hasonló, nagyobb projekt elkészítéséhez. Közös megegyezés alapján én készítem el végül az alkalmazásunk webes részét. Szerencsémre most kipróbálhatom magam, hogy mennyire is tetszik ez az ága a programozásnak.

Terv/Weboldal kinézete

A weboldal tetején egy navigációs sávot képzeltünk el a többiekkel, a sáv bal oldalán a logónk, jobb oldalán sorba az egyéb opciók, például bejelentkezés, forráskód megtekintése, weboldal megosztása. Egy egyszerű sáv, viszont megkönnyebbíti a felhasználók számára a navigálást az alkalmazáson belül.

A főoldal törzse az alkalmazás bemutatkozásával, leírásával kezdődik. Az alkalmazás leírása után képeket terveztünk beilleszteni, hogy milyen is az alkalmazás, milyen közösségek jöhetnek létre az alkalmazás használatával, azonban átgondoltuk és eldöntöttük, annak érdekében, hogy a *vendég* felhasználók is részesülhessenek valamennyi élményben, amit az alkalmazás nyújt, az adatbázisban szereplő helyeket jelenítjük meg a leírás után.

A helyek megjelenítéséhez a *Google Maps* térképét akarjuk használni, ehhez létre kell hozni egy *Google Maps API kulcsot* a developers.google.com oldalon. Azért a Google térképét választottuk, mert Javascript segítségével könnyen lekérdezhető az aktuális helyünk, könnyű megjeleníteni különböző helyeket rajta és egyszerűen formázhatjuk azokat.

Az oldal aljára egy láblécet terveztünk beilleszteni, ahol megjelenítenénk az elérhetőségeinket, egyéb velünk kapcsolatos információkat, az alkalmazás logóját.

Megvalósítás

A tanév elején nagyon elveszettnek éreztem magam. Azt sem tudtam megmondani, hogy milyen eszközöket szeretnék használni a projekt megvalósításához. Igaz, elkezdtem videókat nézni, legtöbbet a *YouTube*-on, ahol emberek felépítenek egy teljes alkalmazást, lemásolnak alkalmazásokat, megcsinálják ugyan azt az alkalmazást, csak más-más keretrendszerekkel. Ezek után még inkább elveszettnek éreztem magam. Nem értettem, hogy miket és hogyan csinálnak azok az emberek, csak a végeredményt láttam, úgy éreztem, hogy hiába szánok időt ennek a megértésére, nem érek el vele semmit.

Az idei tanév alatt, számomra az első nagyobb lépés, amit a vizsgaremek elkészítéséhez tettünk, az a *Laravel* keretrendszer tanulásának elkezdése volt. Ezt a keretrendszert használjuk a projekt **backend** részének megvalósításához. Ahogy elkezdtük csinálni a projektünket, kicsit jobban megértettem, hogy mi is az a keretrendszer, hogy miért olyan hasznos, miért és hogyan is épül fel.

Újra elkezdtem videókat nézni különböző alkalmazások klónozásáról. A legtöbb fejlesztő a **React** Javascript keretrendszert használta ezekben a videókban. Mi is egy Javascript keretrendszert tanultunk eddig idén, a **Vue** Javascript keretrendszert.

Miért a Vue?

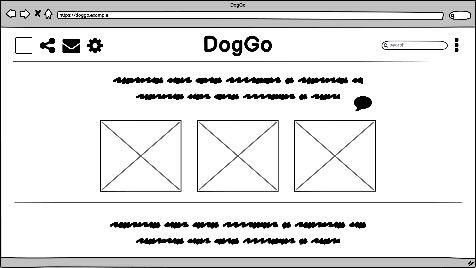
A **Vue** Javascript keretrendszer olyan *feature*-öket tartalmaz, nyújt számomra, amiket már használtam, ismerek. Ezek az előnyök például a többoldalas appok könnyű építése, a *routing*, *komponensek*, adatok *validálása*, számtalan hasznos csomag (*package*), *third party plugin* tartozik a keretrendszerhez (*Vuex, Vuelidate, Bootstrap Vue*). Ezek felett a *Laravel* keretrendszerrel könnyen lehet használni, szinkronizálva van a két keretrendszer.

Ezen felül ez az első frontend keretrendszer, amit idén tanítottak/tanítanak nekünk. Az első órán nem értettem semmit, nem tudtam, hogy mi is az a *kompones*, miért hasznosak, hogyan épül fel egy Vue alkalmazás, milyen parancsokat kell beírni a program tesztelésére, megjelenítésére. Ahogy az idő telik egyre több és több hasznos funkciót fedezek fel a keretrendszerben. Több ismerősöm is mondta, hogy egy könnyen értelmezhető keretrendszer a Vue, tanulásnak jó, ahhoz, hogy jobban megértsem a keretrendszerek felépítését, jó gyakorlás.

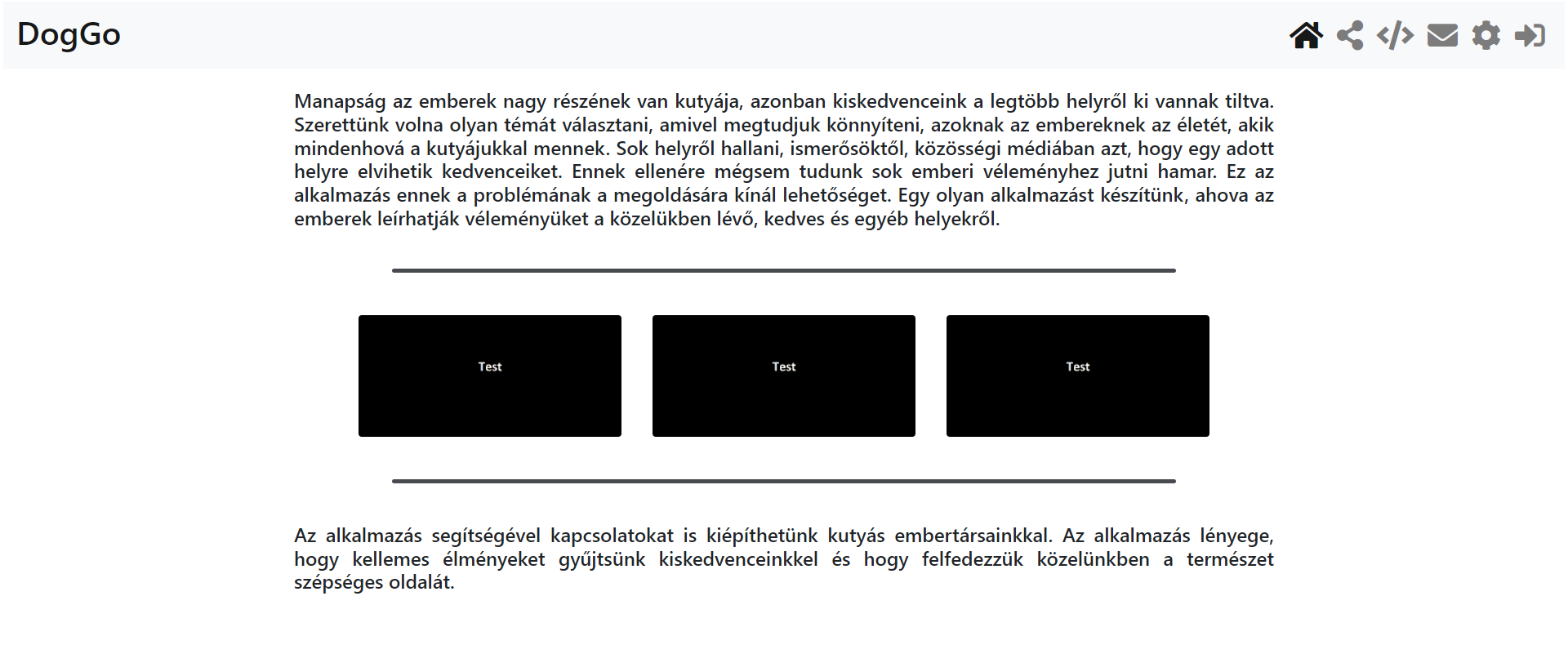
Tudtam, hogy ha elakadok akkor kérhetek segítséget, mert a környezetemben lévő emberek is használták/használják ezt a keretrendszert. Ezeknek a tényezőknek az összessége miatt választottam ezt a keretrendszert, ennek a vizsgaremeknek az elkészítésére

A projekt menete

A weboldal kinézetének tervét először megbeszéltük a csapattal, hogy nekik mi az elvárásuk a weboldallal kapcsolatban, milyen legyen a főoldal eloszlása, mutattam egy hasonló felépítésű weboldalt, majd miután megegyeztünk elkezdhettem a tervezést. A *Balsamiq* program segítségével készítettem el a weboldal kinézetének tervét. A regisztráció, bejelentkezés és a főoldal oldalait akartam eleinte megtervezni.

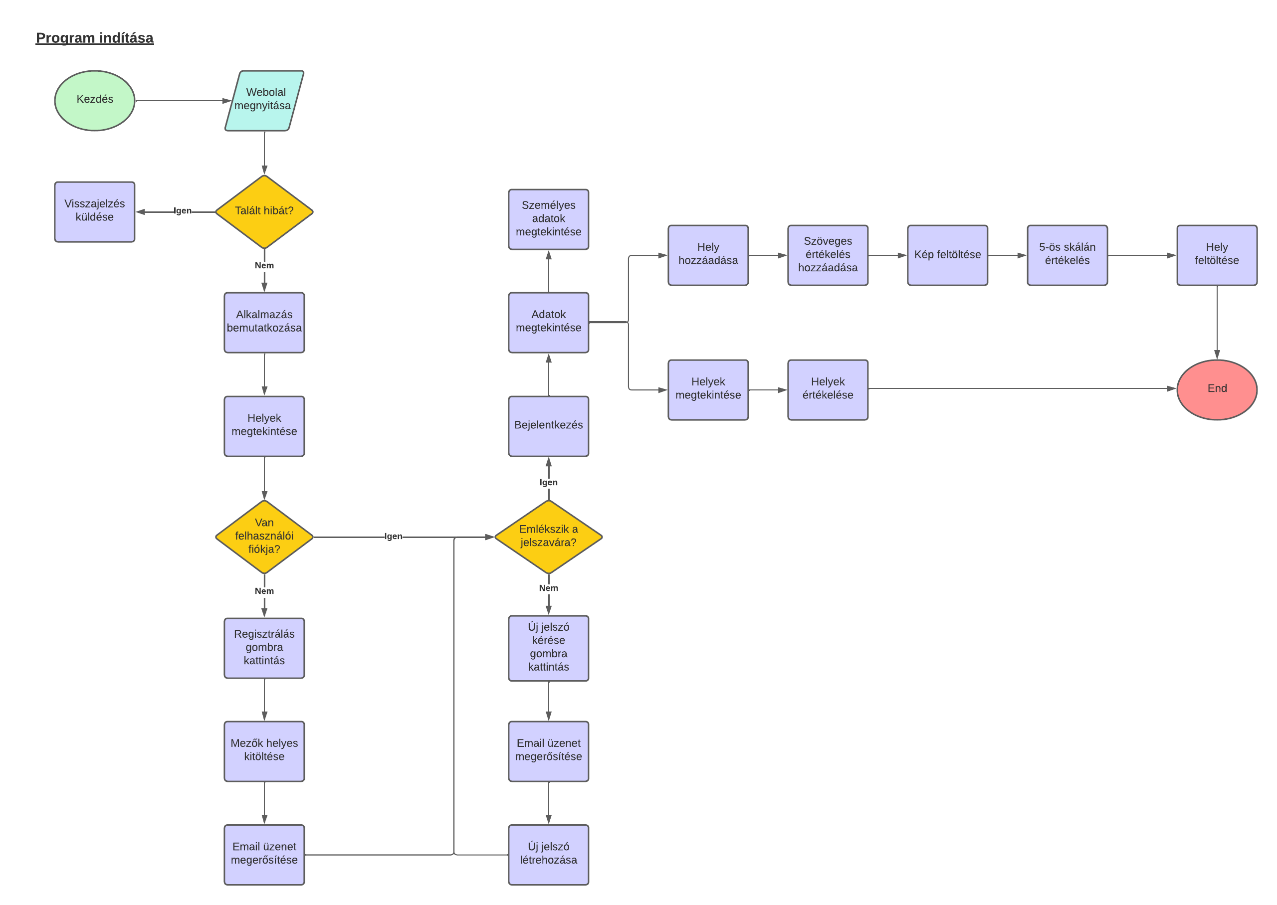


A kezdetleges terv elkészítése után egy sima weboldalt hoztam létre, *HTML* és *CSS* segítségével. Látni szerettem volna, hogy hogy is fog kinézni az alkalmazás.

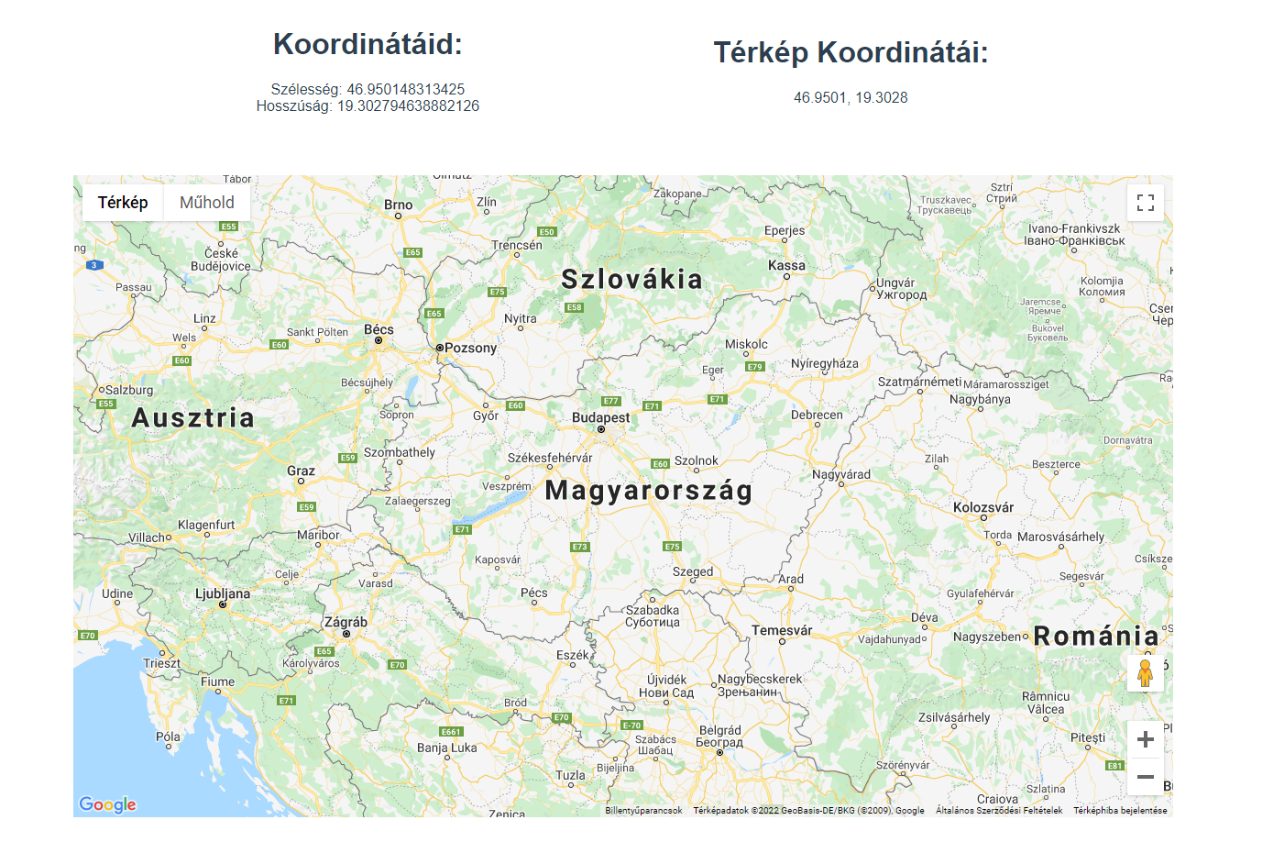


A weboldal tetejére egy navigációs sávot beszéltünk meg. Először magamtól próbáltam meg létrehozni egy reszponzív navigációs sávot. Nem lett olyan az eredmény, mint amilyet elképzeltem. Végül *Bootstrap* segítségével csináltam meg a navigációs sávot, azért is, mert később a Vue keretrendszerhez használhatom a *bootstrap*-et is és csak át kell másolnom a kódot.

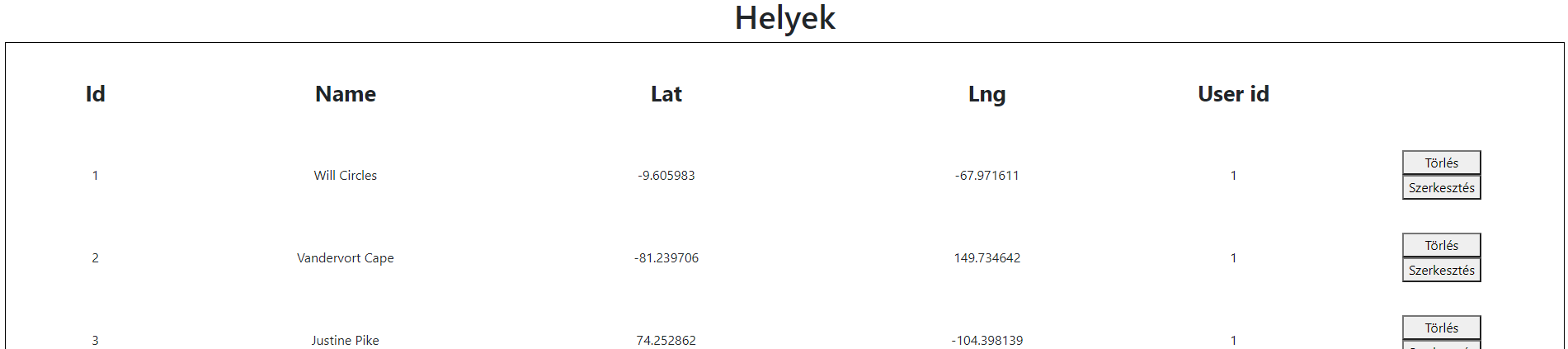
Még mielőtt a keretrendszer segítségével is elkezdtem volna a weboldal elkészítését, az oldal folyamatábráját is meg akartam csinálni.



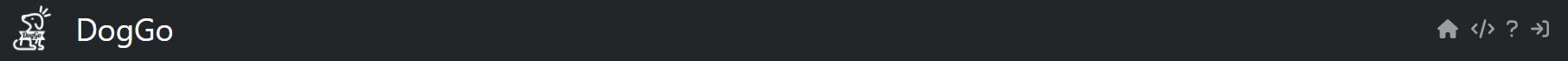
Végül nem a navigációs sáv elkészítésével kezdtem a projektet, hanem a térkép elkészítésével. A *Youtube*-on találtam egy videót, amiben bemutatják a *Google Maps* egyszerű alkalmazását egy weboldalon. A videó végére meg tudtam jeleníteni az aktuális koordinátáimat és a térkép aktuális koordinátáit.



Elég sok idő telt el a térkép megvalósításával és nem haladtam a weboldal elkészítésével, ennek a problémának az orvoslására egy kis időre a térképet egy Lista *komponens*sel helyettesítem. Ez a komponens megjeleníti az adatbázisban lévő helyeket és azoknak adatait. A helyeket tudjuk módosítani, törölni és új helyet is hozzá tudunk adni az adatbázishoz.



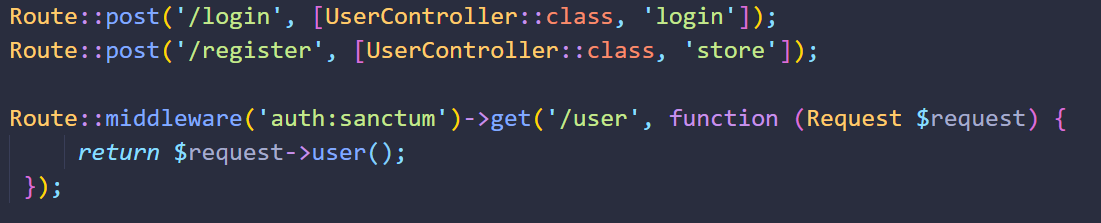
Számomra jó visszaigazolás volt, hogy az app tud a *backend*-del kommunikálni. A kommunikáció sikeres megvalósítása után a bejelentkezést, illetve regisztrálást szerettem volna megvalósítani. A kezdetleges navigációs sáv elkészítése után megpróbáltam a keretrendszer használatával is elkészíteni. Egyelőre így néz ki a navigációs sáv.



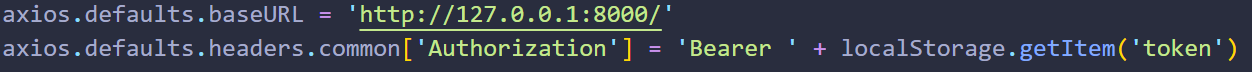
Az ikonokat a **Font Awesome** használatával valósítom meg. Egyszerű az ikonjaik importálása, használata és rengeteg ingyenesen használható ikonjuk van, én csak az ingyenes ikonokat használom ebben a projektben.

Amikor bejelentkezünk akkor a felhasználó nevét akarom megjeleníteni a bejelentkezés gomb helyett, amire kattintva egy lenyíló menü jelenik meg, ahol egy kijelentkezés és egy adatok módosítása gomb jelenik meg. Az adatok módosítása gombra kattintva az alkalmazás átirányít egy másik komponensre, ahol a felhasználó adatait módosíthatjuk.

Ennek a tervnek a megvalósításához el kell készítenem egy működő regisztrálás és bejelentkezés komponenst. A regisztráláshoz a */register* végpontot kell használnom, a bejelentkezéshez a */login* végpontot. A */login* végpont a felhasználó *token*-jét, ”kulcsát” adja vissza helyes bejelentkezés esetén. Ezt a kulcsot a */user* végpontnak adom át, ami visszaadja majd a felhasználó adatait. Ezzel a módszerrel a felhasználó adatai titkosítva vannak.



A regisztrálás egy sima *post* típusú metódus olyan, mint a hely hozzáadása. A bejelentkezésnél a */user* végpont megvalósítása jelentett nehézséget. Létrehoztam egy **axios.js** nevű fájlt, amiben eltároltam az ismétlődő *axios* kódokat.

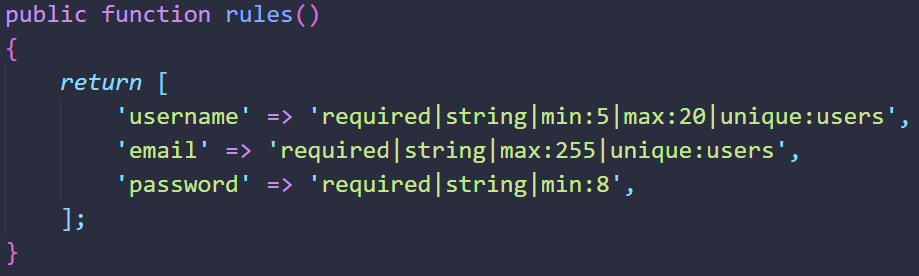


Emiatt a **Login** komponens *login* metódusát így valósítottam meg.

A Vue ügyféloldali tárolójában eltároltam a *token*-t, hogy az oldal újratöltésénél automatikusan bejelentkezhessen majd az alkalmazás. A bejelentkezés gombra kattintva a főoldalra navigál minket a program. Mivel számos weboldalon a regisztráció után rögtön megvalósítja a bejelentkezést is, én is úgy döntöttem, hogy alkalmazom ezt a funkciót az alkalmazásban. Amint látszódik a kódrészben, **Vuex**-et használok a felhasználó adatainak eltárolására. A felhasználó adatai így elvesznek, ha frissítem az oldalt.

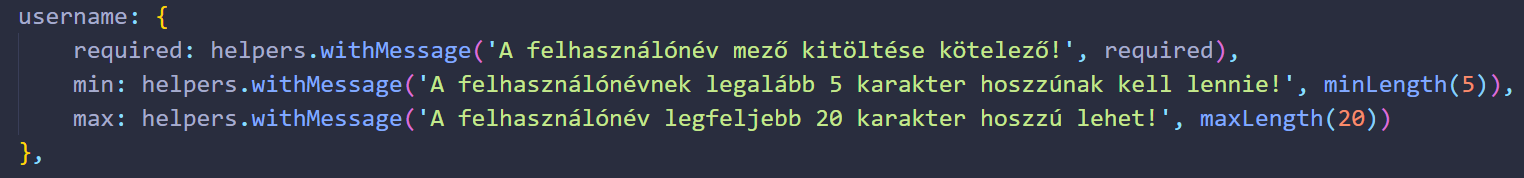
Számtalan nehézségbe ütköztem az autentikációval kapcsolatban, néhány hibát még most is félretettem, mert nem akarom, hogy az összes időmet feleméssze ez a része a projektnek. Ezek közé a hibák közé tartozik az, hogy nem mindig ugrik a főoldalra az alkalmazás regisztrálás után. Ha egy felhasználóval bejelentkeztem, kijelentkezek, majd egy másik fiókkal lépek be, az alkalmazás az előző felhasználónevet írja ki a navigációs sávnál. Mivel összességében sikerült az autentikáció megvalósítása a frontend részén a projektnek, a felhasználó adatai titkosítva vannak és a bejelentkezés is megfelelően működik a token használatával, későbbre halasztottam ezeknek a hibáknak a javítását.

Miután végeztem az alapvető autentikáció elkészítésével, a bemeneti értékek validálásával foglalkoztam. **Frontent validációt** akartam a kezdetektől használni, mert nem kell a szerverrel kommunikálni az értékek érvényesítéséhez, nem csak akkor kapok választ, hogy megfelelőek-e az adatok, ha *meghívom* a végpontot, tehát ezzel a problémával nem terhelném a szervert. Szerencsémre a Vue keretrendszer erre a validálásra is nyújt megoldást. A **vuelidate** a keretrendszer egyik kiegészítő eleme, ennek a segítségével valósítottam meg az adatok ellenőrzését.

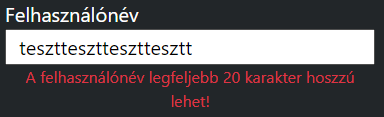
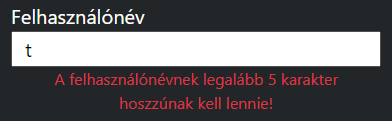
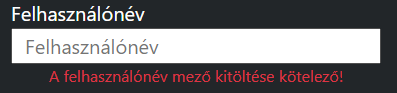
A *vuelidate* használata nem okozott gondot a dokumentáció értelmezése után. Ellenőriztem, hogy milyen követelményeket adtunk meg a bejövő adatoknak a vizsgaremek backend részén az *app/Http/Requests* mappán belül, majd a követelmények alapján megírtam a paramétereket, hogy milyen értékeknél milyen is legyen a hibaüzenet az alkalmazás frontent részén.

Egy felhasználó felhasználónevének a következő paramétereket állítottuk be:

* A felhasználónév mező **kitöltése kötelező**
* A felhasználónévnek legalább **5** karakter hosszúnak kell lennie
* A felhasználónév legfeljebb **20** karakter hosszú lehet

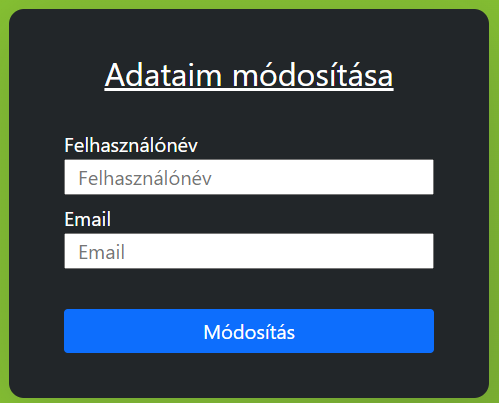


A kód része így valósult meg. Mivel a vuelidate felépítése megegyezik az adatokéval, sokkal átláthatóbb és könnyebben értelmezhető az értékek validálása.

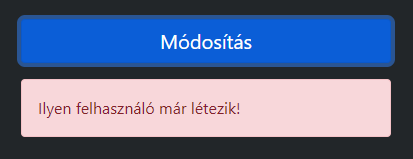


Az autentikáció alapvetően működik, úgy terveztük, hogy a felhasználó adatait módosítani lehet, következő lépésként ezt a funkciót szeretném megvalósítani.

Létrehoztam egy *UserData* nevű komponenst, itt valósítom meg ezt a funkciót. Mivel a jelszót email üzenet segítségével módosíthatjuk ezen a komponensen belül csak a **felhasználónevet** és a felhasználó **email címét** módosíthatjuk, ezért kell nekem kettő bemeneti érték. A komponens alapvető formázásával kezdtem. A két bemeneti értékhez egy gombot adtam hozzá, amivel a kérésünket elküldjük a szervernek.

Sikertelen módosítás esetén visszajelzést kapunk az oldalon. Az adatok sikeres módosítása esetén a felhasználót visszanavigálja az alkalmazás a főoldalra, a navigációs sáv jobb oldalán pedig az új felhasználónevet láthatjuk. Az értékek validálása után, ezek az értékek megegyeznek a felhasználó létrehozásáéval (*vuelidate*-tel), a megfelelő végpont meghívása következett. Mivel meglévő adatot módosítunk, ez egy *PUT* metódus lesz.

Eleinte az volt a hiba ennél a metódusnál, hogy nem volt a jelszó titkosítva, ha egy felhasználó adatát módosítottuk. A projekt backend részén megoldottuk ezt a problémát, utána minden sikeresen működött.

Ha olyan adatokat próbálnánk megadni, amik már léteznek egy másik felhasználónál, egy hibaüzenetet jelenít meg a *Módosít* gomb alatt a komponens.

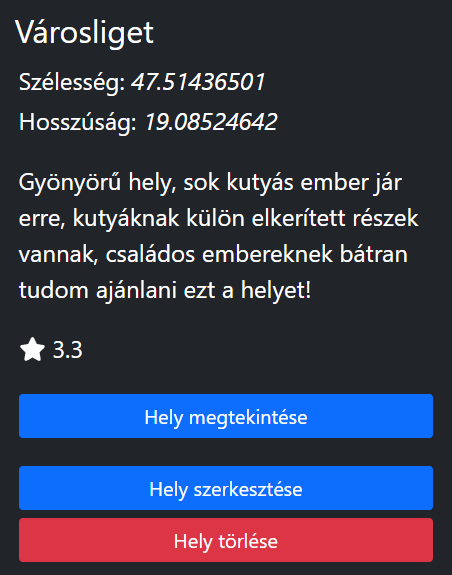
Ellenkező esetben, ha a felhasználó adatainak módosítása sikeres volt, a komponens visszavezet minket az előző oldalra, ami meg volt nyitva nekünk.

A következő lépés számomra a **helyek** és a helyekhez tartozó **értékelések** megvalósítása. A helyek megjelenítését **Google Maps** segítségével terveztük megvalósítani. Nem sikerült megvalósítanom rövid időn belül, nem akartam, hogy minden időm menjen el ezzel is, úgy döntöttünk, hogy ez egyelőre egy továbbfejlesztési lehetőség lesz és most csak azt valósítom meg, hogy a helyeket listázni lehessen, helyeket hozzáadni, módosítani és törölni. A helyekhez tartozó értékeléseket megtekinteni, hozzáadni értékeléseket, módosítani és törölni.

A csapattársaimmal azt beszéltük meg, hogy a főoldalon, az alkalmazás bevezetése után jelenítem meg a helyeket. A végpont elkészítése után a helyek listázása egyszerű volt. Létrehoztunk egy végpontot, ami megjeleníti a legsikeresebb, legjobban értékelt helyet. Ezt a helyet a többi hely fölé helyeztem.

A *”Hely megtekintése”* gombra kattintva, az alkalmazás a hely értékeléseihez navigál minket. A projekt ezen része nem jelentett nehézséget, minden funkció, amit meg akartam valósítani hamar működött megfelelően.

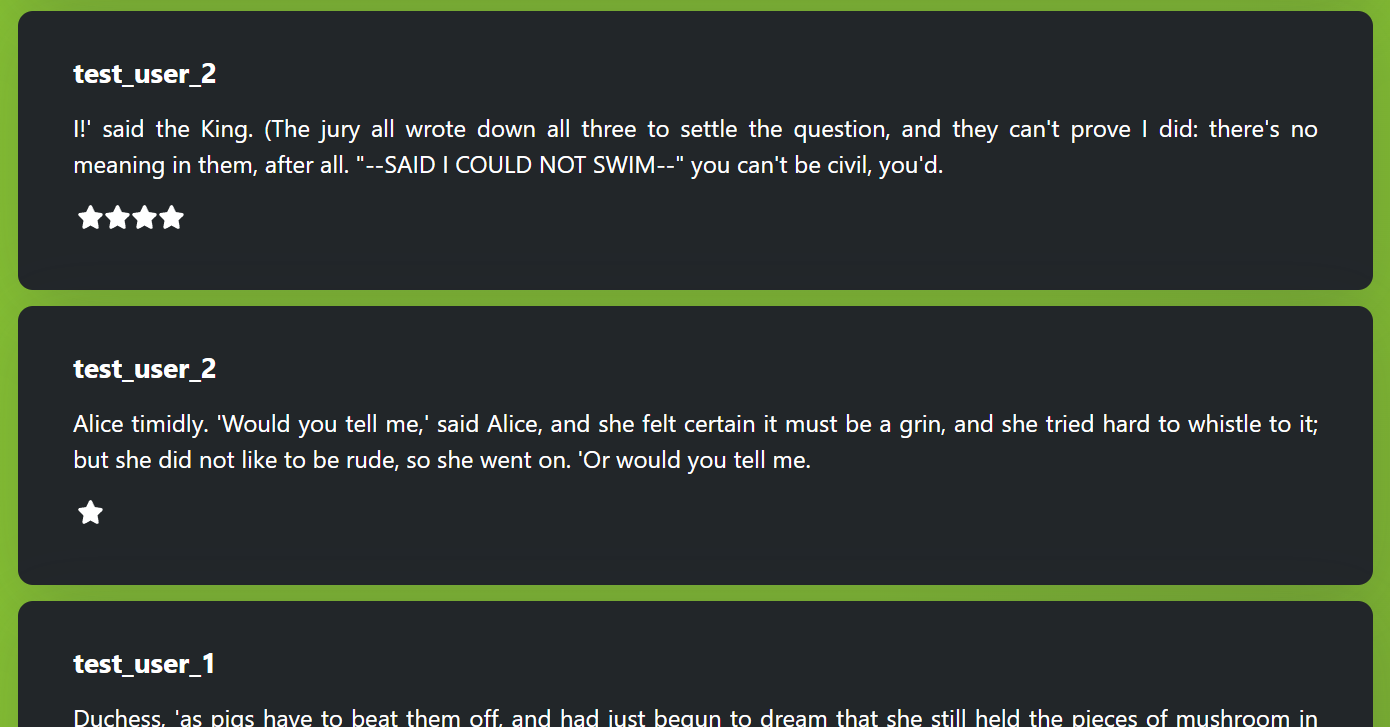
A következő lépés az **admin**, illetve **szuperadmin** jogosultsággal rendelkező felhasználókhoz kapcsolódott. Úgy beszéltük meg, hogy az admin és szuperadmin jogosultsággal rendelkező felhasználók az összes helyet módosíthatják, törölhetik.

Az alkalmazásnak ez a funkciója is hamar megvalósult. Egyelőre kezdetleges a megvalósítása, de megfelelően működik.

A hely szerkesztése esetén, a *”Hely szerkesztése”* gombra kattintása után, a beviteli mezők a főoldal alján jelennek meg, ott lehet a hely adatait módosítani.

A hely törlése egyszerűbb, a *”Hely törlése”* gombra kattintva, eltávolíthatjuk az adatbázisból a helyet.

A következő rész a projektben az értékelések listázása volt. Ennek a funkciónak a helyes megvalósítása jelentett eleinte gondot. A helyek értékeléseit el tudtam kérni a szervertől, minden értékeléshez tartozik egy *”location\_id”*. Az értékeléseket a helyekkel a projekt frontend részén kapcsoltam össze az alkalmazásban. Ez a fajta megvalósítás egy nagyon lassú betöltést jelentett és nagyon megterhelő is a szerver számára.

A megoldás erre a problémára az, hogy csináltam még egy végpontot, aminek a meghívásakor megkapom az értékelésekhez tartozó helyek nevét és a hozzá tartozó felhasználó nevét is, annak a felhasználónak a nevét, aki az értékelést írta. Ezzel a megoldással egy kéréssel megkapom az adott helyhez tartozó értékeléseket és az értékelésekhez tartozó **felhasználó nevét** is meg tudtam jeleníteni.

Minden értékelés tetején található a felhasználó (aki írta az adott értékelést), az értékelés leírása, és a skálás értékelés, csillag ikonokkal formázva, ez mutatja, hogy az ötös skálán hanyas értékelést adott a felhasználó a helynek.

Így, hogy kész lett a projekt fő funkciója, ami miatt egyedi ez a program, elkezdtem a projekt készítése közben kialakult hibákat javítani. A számomra legkellemetlenebb hibával kezdtem, ez az autentikációs hiba volt.

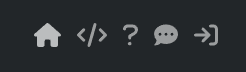
Ezt a hibát úgy kezeltem, hogy a **Login** komponensen belül csak a felhasználó *token*-jét tároltam el, a bejelentkezést eltávolítottam ebből a komponensből. A token eltárolása után frissítettem az oldalt. Az oldal frissítésekor a **Header** komponens ellenőrzi, hogy a token létezik-e. Mivel bejelentkezésnél a token értéke nem lehet üres és csak érvényes értéket vehet fel, elég csak ez a vizsgálat, emiatt, ha a token létezik, tehát az érték nem üres, a bejelentkezés metódust meghívja. Abban az esetben, ha bejelentkezett felhasználóként használjuk a weboldalt, a **Login** és **Register** komponensek nem jeleníthetők meg, a weboldal visszanavigál minket a főoldalra.

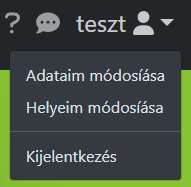
Komponensek

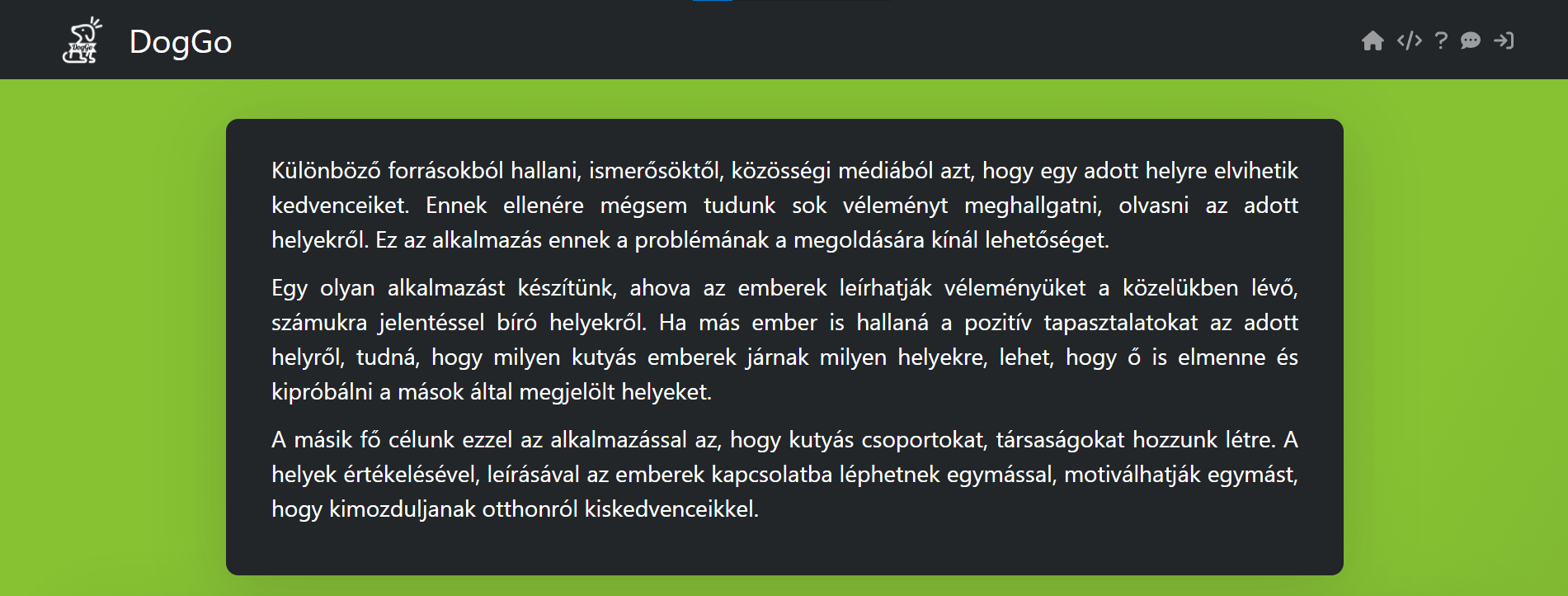
A komponenseket a *components* mappán belül hoztam létre. Ezen a mappán belül pedig négy almappában tároltam el a különböző komponenseket.

Az *authentication* almappán belül találhatóak a Login és Register komponensek. A **Login** komponens használatával tudunk bejelentkezni az alkalmazásba, új felhasználó létrehozása nélkül. Mint már megemlítettem, ennél a komponensnél a token használata jelentett nehézséget számomra, mert nem láttam át eleinte, hogy miért és hogyan lehet a token alkalmazásával egy felhasználó adatait megkapni és aztán eltárolni. Ennél a komponensnél egy metódus található, a **login** metódus. Ez az eljárás a bejelentkezést valósítja meg, ha a szervertől hibaüzenet érkezik, akkor a hibaüzenetet megjeleníti a komponens. A **Register** komponensnél is, hasonlóan, egy metódus található, a **register** metódus. A regisztráción kívül a bejelentkezést is megvalósítja ez a metódus. Abban az esetben, ha a szerver válaszként hibát küld vissza, az oldalon hibaüzenet jelenik meg.

Felhasználói dokumentáció

Az alkalmazás megnyitásával a weboldal **főoldalát** láthatjuk. Az applikáció minden oldalán található egy *navigációs sáv*, a könnyebb navigálás érdekében.

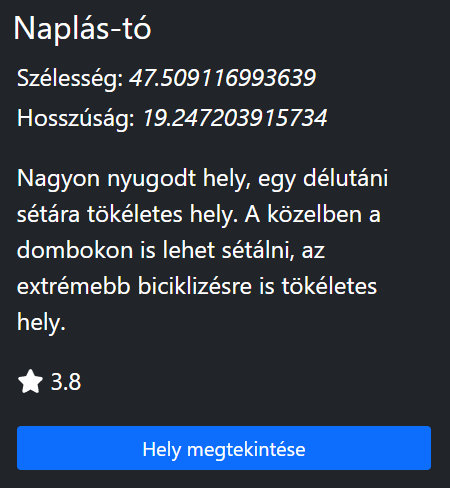
Az alkalmazás tetején a navigációs sávot láthatjuk. A különböző ikonokra kattintva navigálhatjuk magunkat az oldalak között. A navigációk sorba a **főoldalt**, a **forráskód** megtekintését, **információt** a weboldalról, a **hiba jelentést** és a **regisztrációt**, illetve **bejelentkezést** jelentik. Abban az esetben, ha bejelentkeztünk, a regisztráció ikon helyett a *saját felhasználónevünk* jelenik meg, egy legördülő menüpontként, ahol a menüpontok sorba a felhasználó **adatainak** módosítását, a felhasználó **helyeinek** módosítását és a **kijelentkezést** jelentik.

A főoldal tetején az alkalmazás bemutatkozása található. Itt megérthetik a felhasználók, hogy miért is jött létre ez az alkalmazás, mi az alkalmazás alapvető célja.

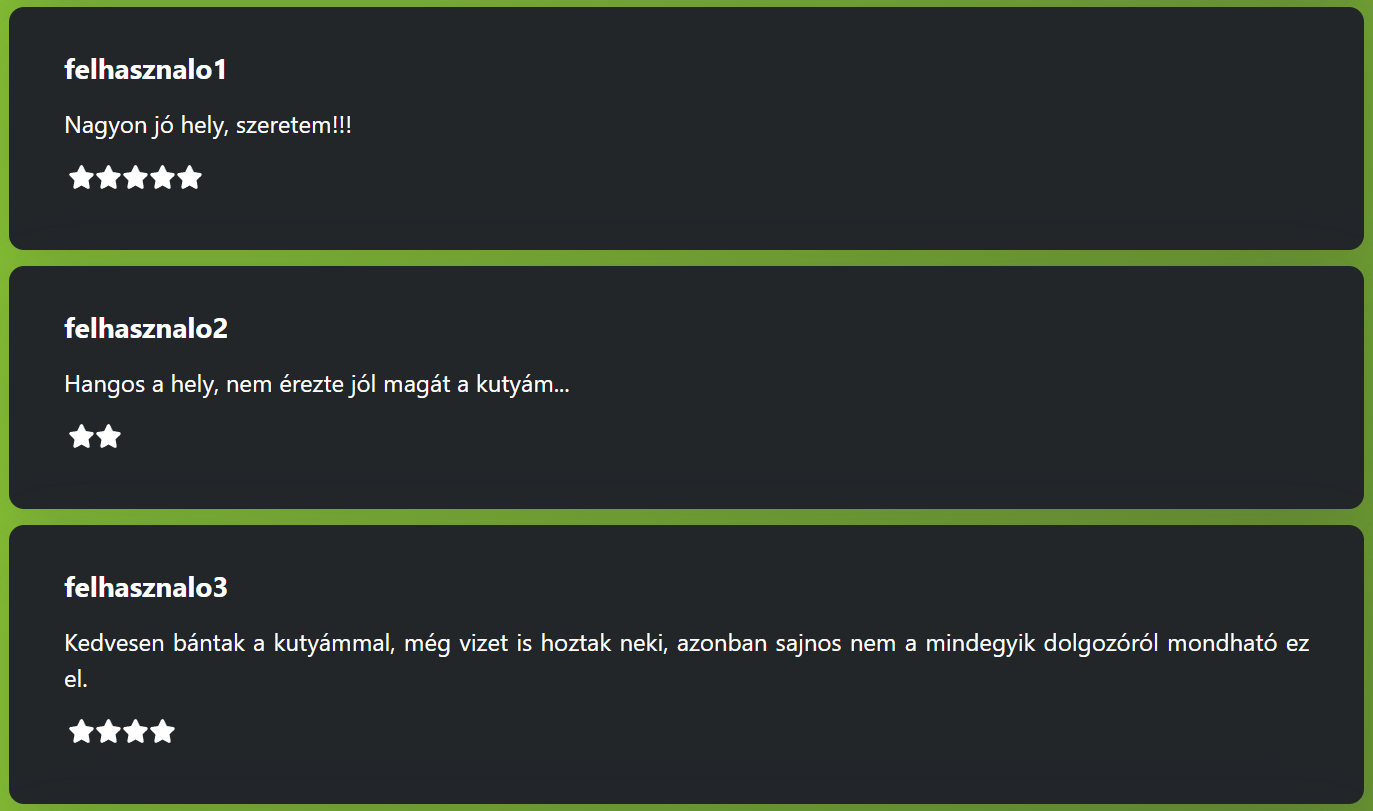
A bemutatkozás után az alkalmazás **helyei** találhatóak. A helyek tetején található a *legjobb értékeléseket* kapott hely. Ezt a helyet ajánlották leginkább a felhasználók, alatta található a többi hely.

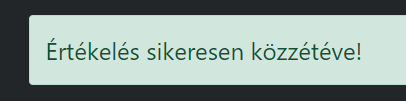
Mindegyik hely rendelkezik egy szélességgel és hosszúsággal, ez jelzi a helyek koordinátáit.

**Értékelések**

Abban az esetben, ha a hely kapott legalább egy **értékelést**, a hely átlagos értékelését láthatjuk a hely tulajdonságai között (a hely leírása alatt). A helyek tulajdonságai sorba a hely **neve**, a hely **szélessége**, **hosszúsága**, a hely **leírása**, a hely **átlagos értékelés** és az **értékelések** megtekintése, amelyet a *”Hely megtekintése”* gombra kattintva tekinthetünk meg. Abban az esetben ha meg szeretnénk nézni egy hely értékeléseit, a gombra kattintva egy új oldalra navigál minket az alkalmazás.

A helyek *értékeléseinek megtekintéséhez* nem kell bejelentkezni, *vendég felhasználóként* is megtekinthetjük az értékeléseket. Ellenkező esetben, a helyeket csak akkor lehet értékelni, ha bejelentkezett felhasználóként használjuk az alkalmazást.

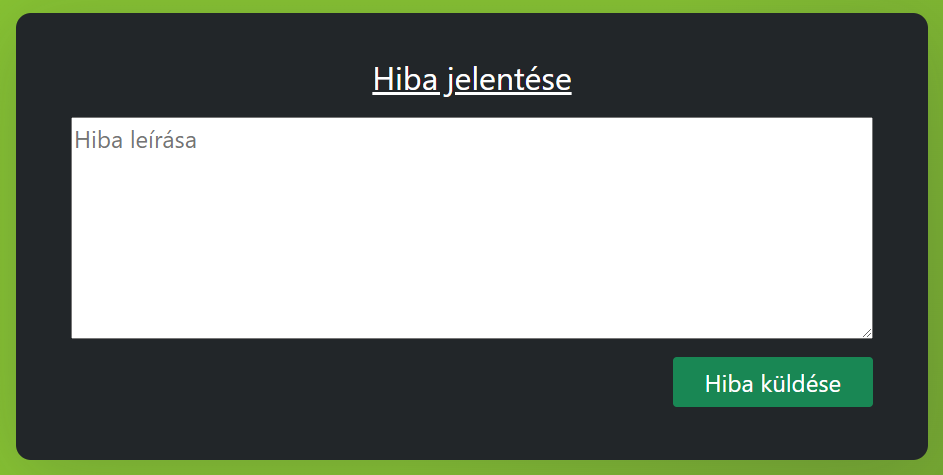


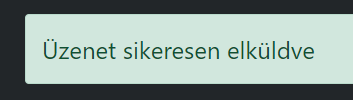
Egy értékelés tulajdonságai sorra a következők, a **felhasználónév**, azt jelenti, hogy melyik felhasználó értékelését láthatjuk, a **szöveges értékelés** és a **skálás értékelés**. A szöveges értékelés legfeljebb 255 karakter lehet, a skálás értékelés pedig egy egytől ötig tartó skálás értékelés.

Az értékelés *sikeres*, illetve *sikertelen* közzétételéről visszajelzést kapunk. Abban az esetben, ha nincs megfelelően kitöltve az értékelésünk, értesítést kapunk, csak úgy, mint minden beviteli mezőnél az alkalmazásban.

**Hiba jelentése**

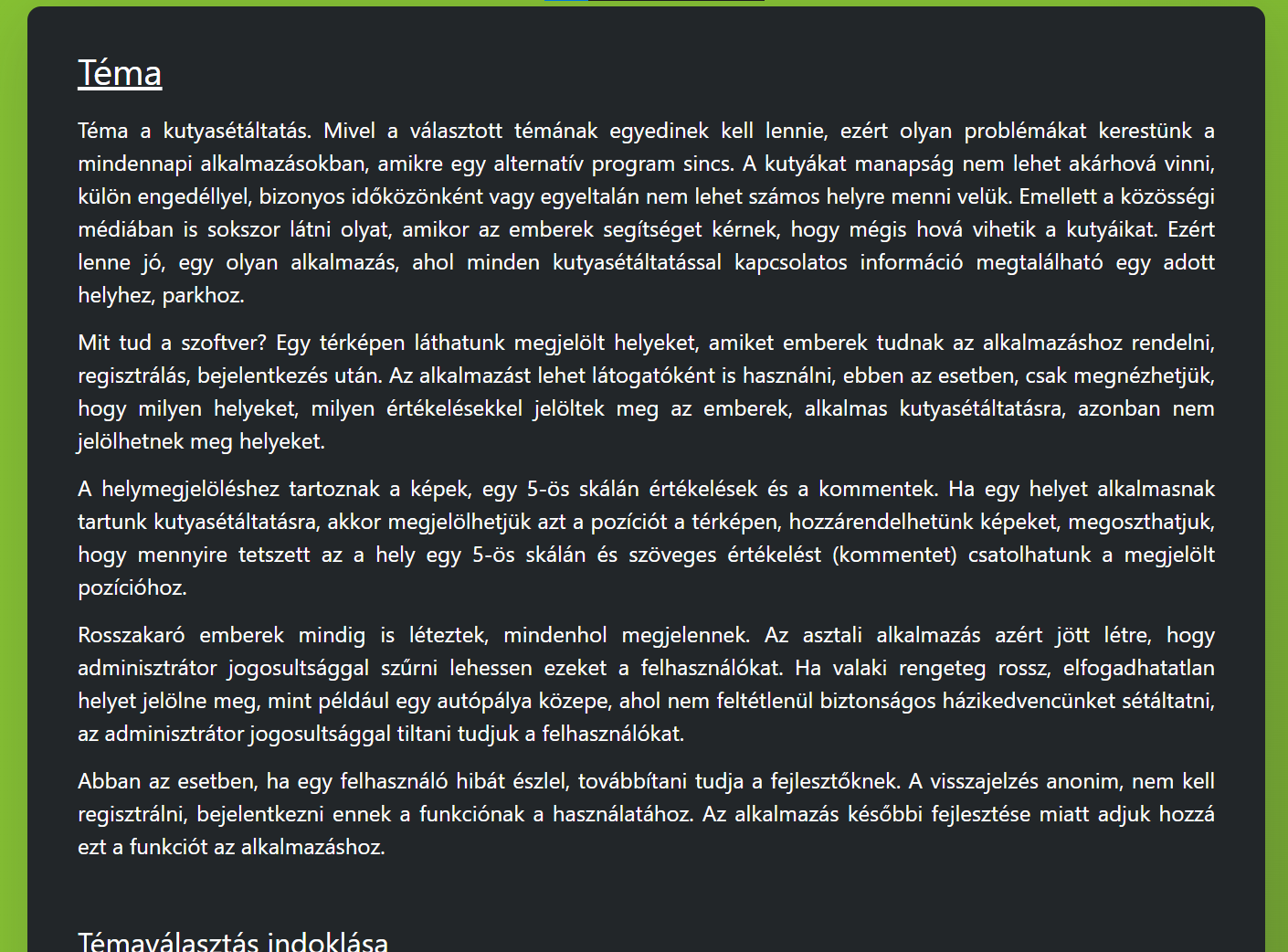
Abban az esetben ha valamilyen hibát észlel a felhasználó, lehetősége van ezt a hibát továbbítani a fejlesztőknek a *hibajelentés* fülön.



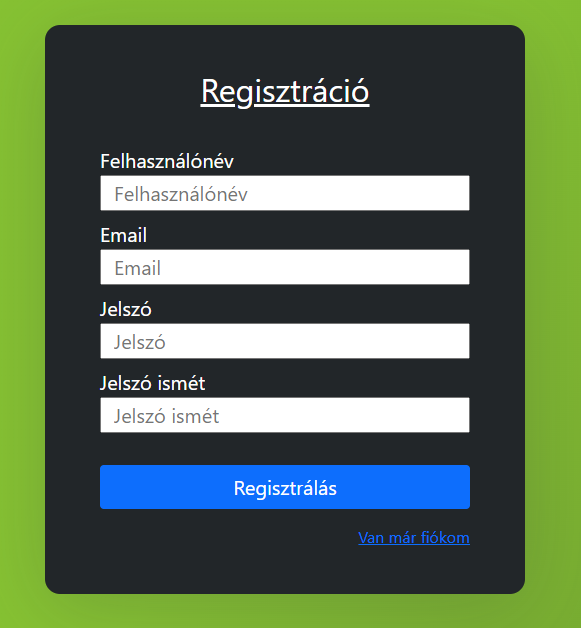
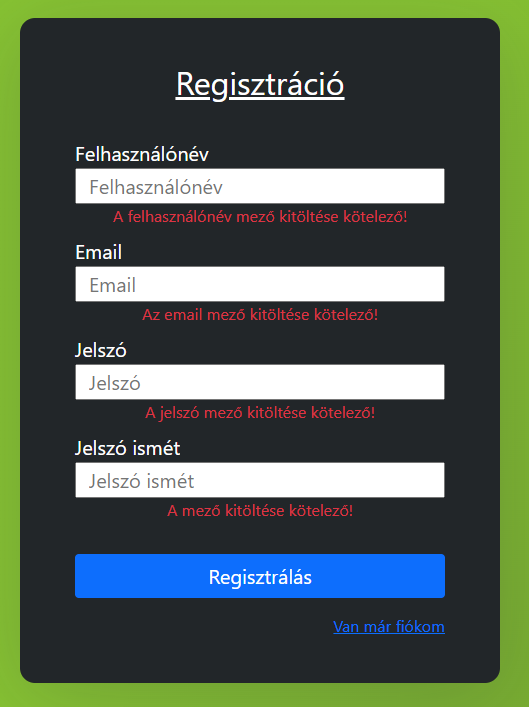
A mező helyes kitöltése után, a hiba elküldéséről szintén visszajelzést kapunk.

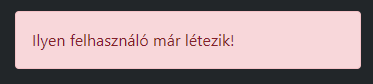
A hiba jelentéséhez nem kötelező a bejelentkezés.

**Információ**

Az *információ* oldalon az alkalmazás részletesebb ismertetését láthatjuk, ezen az oldalon nincs semmilyen funkció a részletek elolvasásán kívül.

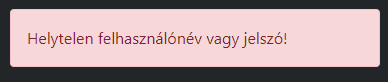
**Regisztráció**

A *”regisztrálás”* gombra kattintva, a regisztrációs oldalra kerül a felhasználó. Négy mező kitöltése *kötelező* a regisztráláshoz. Helyes beviteli értékek esetén, a regisztrálás gombra kattintva a regisztráció sikerességéről visszajelzést kapunk.

Sikeres regisztrálás esetén automatikusan bejelentkezik az alkalmazás, emiatt látjuk felhasználónevünket a fejléc jobb oldalán, sikeres regisztráció után. Ellenkező esetben, például, ha olyan felhasználót próbálunk létrehozni, amelyik már létezik, a következő hibaüzenet jelenik meg a regisztráció oldalán.

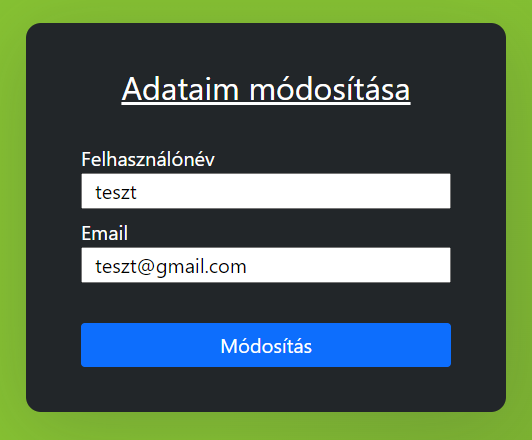
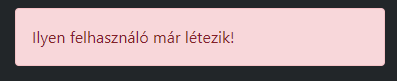
**Bejelentkezés**

Abban az esetben, ha van már felhasználói fiókja és abba szeretne a felhasználó bejelentkezni, a *”Van már fiókom”* gombra kattintva tud bejelentkezni a felhasználó.

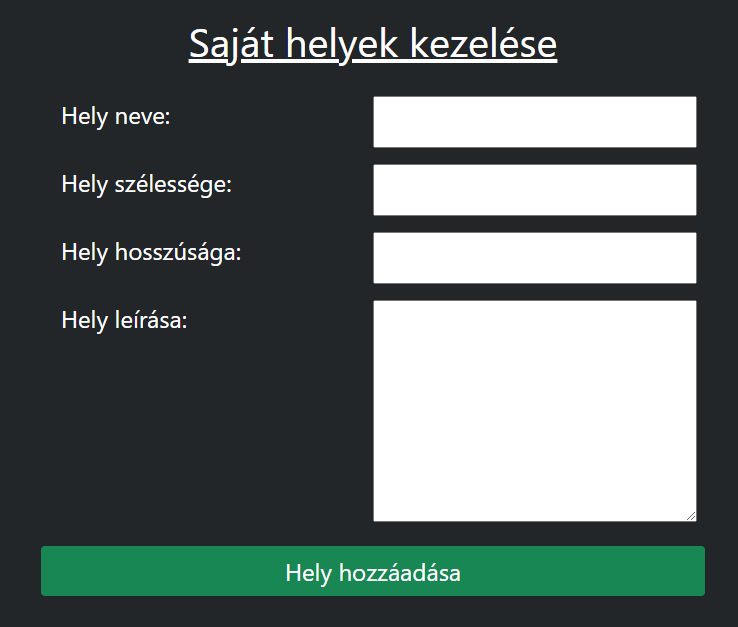
A mezők helyes kitöltése után a *”Bejelentkezés”* gombra kattintva bejelentkezünk, ha helyesen adtuk meg az adatainkat, ebben az esetben a navigációs sáv jobb oldalán a felhasználónevünket láthatjuk és értékeléseket és helyeket is hozzá tudunk adni az alkalmazáshoz.

Ha a bejelentkezés sikertelen, hibaüzenet jelenik meg. A beírt adatok ilyenkor nem vesznek el, könnyen lehet a beviteli mezőket módosítani.

**Felhasználó adatainak módosítása**

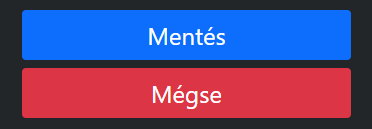
Abban az esetben ha módosítani kívánja adatait, a navigációs sávon a felhasználónevére kattintva, a legördülő menü *”Adataim módosítása”* menüpontjára kattintva a weboldal átnavigál minket az *adataim módosítása* oldalra. A **felhasználónév**, illetve az **email cím** módosítására van lehetőségünk. Az oldal betöltésekor a beviteli mezők értékei automatikusan betöltődnek. A mezők megfelelő kitöltése után a *”Módosítás”* gombra kattintva tudjuk módosítani az adatainkat. A módosítás sikeréről visszajelzést kapunk. Az adataink sikeres módosítása után a főoldalra navigál minket a program, a navigációs sávon pedig az **új felhasználónevünket** láthatjuk.

Abban az esetben, ha sikertelen volt adataink módosítása, hibaüzenet jelenik meg az adataim módosítása oldalon.

**Saját helyek**

Ha magunk akarunk egy helyet hozzáadni az alkalmazáshoz, a *”Helyeim módosítása”* menüpontra kattintva ezt megtehetjük.

Az alkalmazás a **Saját helyek kezelése** oldalra navigál minket, itt találhatjuk azokat a helyeket, amiket mi adtunk az alkalmazáshoz. Ha mi is hozzá szeretnénk adni egy helyet a programhoz, a mezők megfelelő kitöltése után a *”Hely hozzáadása”* gombra kattintva ezt megvalósíthatjuk.

A hely publikálása után megváltoztathatjuk, illetve törölhetjük az általunk hozzáadott helyek adatait. A *”Szerkesztés”* gombra kattintva módosíthatjuk a kiválasztott helyünk adatait, ahol hozzáadtuk a helyünket, azokba a mezőkben betöltődnek a módosításra váró hely adatai. A hely adatainak javítása után, a *”Mentés”* gombra kattintva frissíthetjük helyünk adatait, ha meggondoltuk magunkat és mégsem akaruk a hely adatait módosítani a *”Mégse”* gombra kattintva kiléphetünk a hely módosításából. A *”Törlés”* gombra kattintva a kiválasztott helyünket **törölhetjük**.