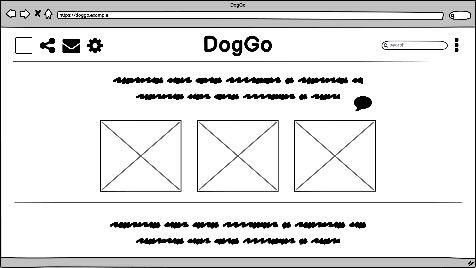
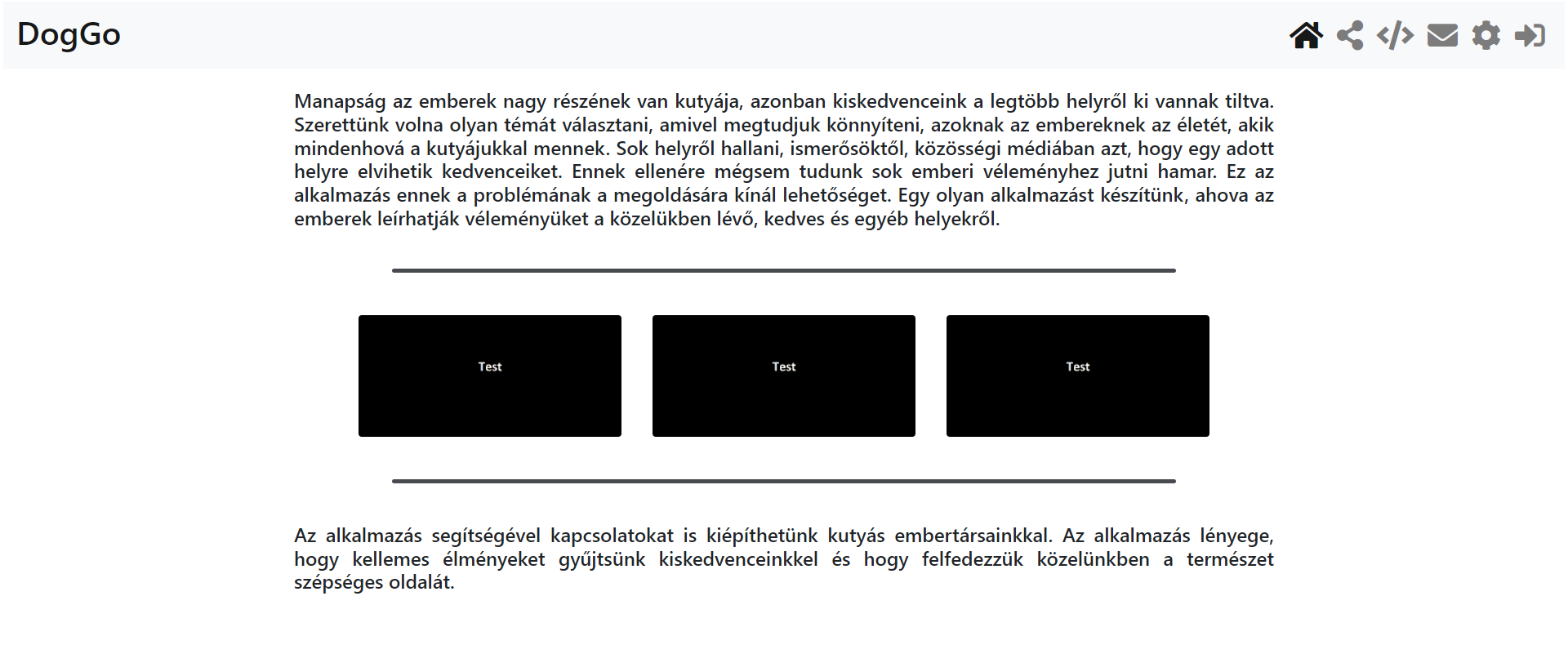
A projekt menete

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.

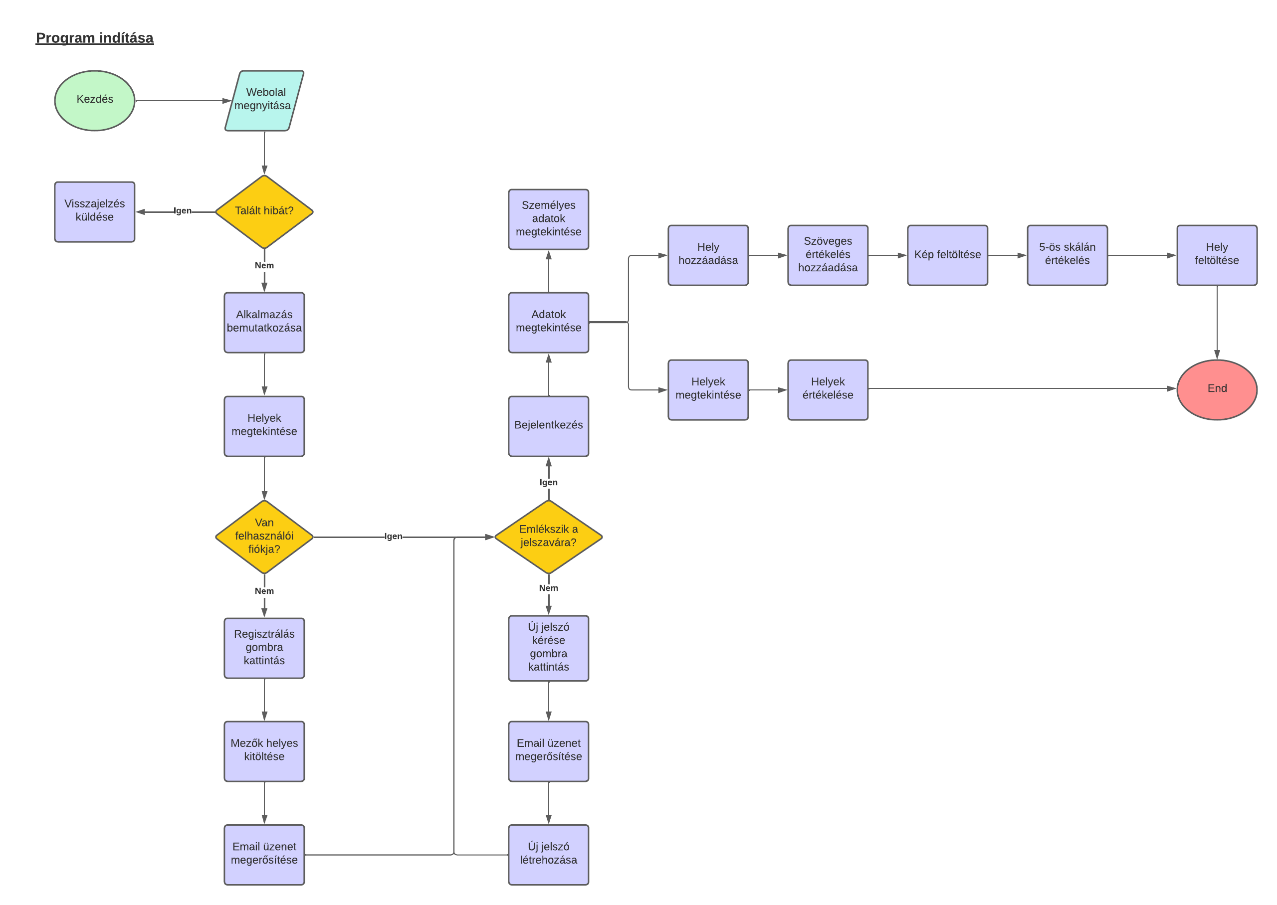


A kezdetleges terv elkészítése után egy sima weboldalt hoztam létre, *HTML* és *CSS* segítségével. Látni szerettem volna, hogy hogy is fog kinézni az alkalmazás.

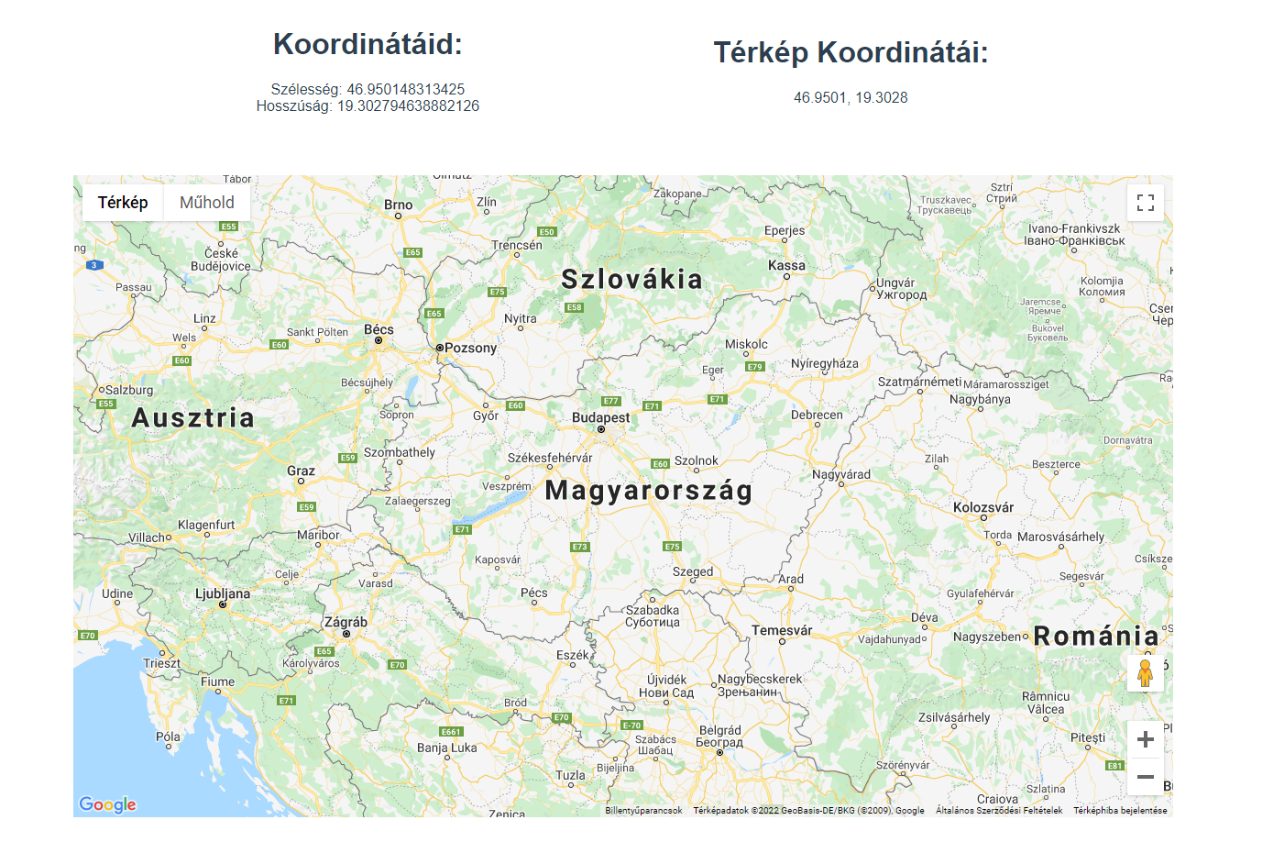


A weboldal tetejére egy navigációs sávot beszéltünk meg. Először magamtól próbáltam meg létrehozni egy reszponzív navigációs sávot. Nem lett olyan az eredmény, mint amilyet elképzeltem. Végül *Bootstrap* segítségével csináltam meg a navigációs sávot, azért is, mert később a Vue keretrendszerhez használhatom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

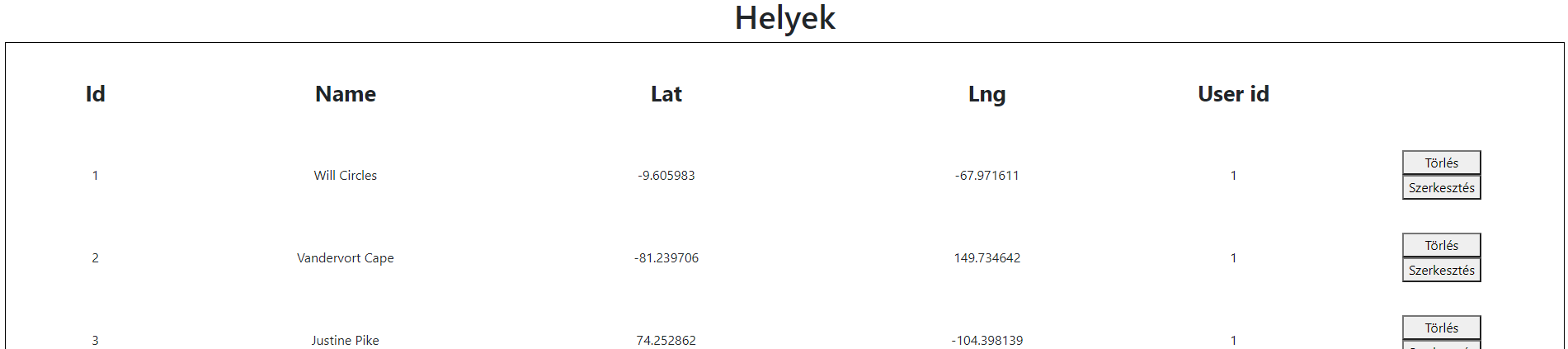
Még mielőtt a keretrendszer segítségével is elkezdtem volna a weboldal elkészítését, az oldal folyamatábráját is meg akartam csinálni.



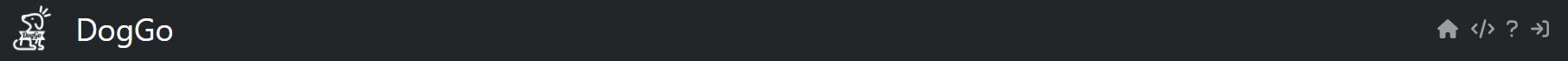
Végül nem a navigációs sáv elkészítésével kezdtem a projektet, hanem a térkép elkészítésével. A *Youtube*-on találtam egy videót, amiben bemutatják a *Google Maps* egyszerű alkalmazását egy weboldalon. A videó végére meg tudtam jeleníteni az aktuális koordinátáimat és a térkép aktuális koordinátáit.



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.



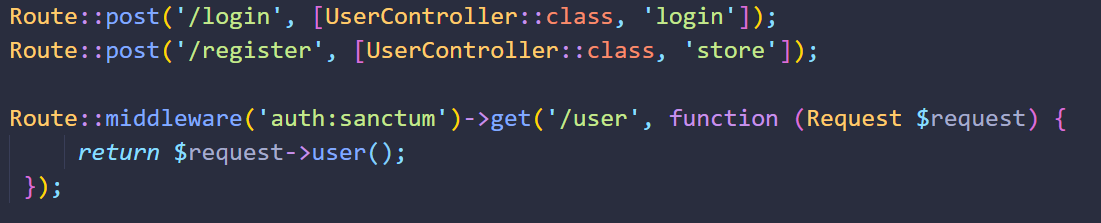
Számomra jó visszaigazolás volt, hogy az app tud a *backend*-del kommunikálni. A kommunikáció sikeres megvalósítása után a bejelentkezést, illetve regisztrálást szerettem volna megvalósítani. A kezdetleges navigációs sáv elkészítése után megpróbáltam a keretrendszer használatával is elkészíteni. Egyelőre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



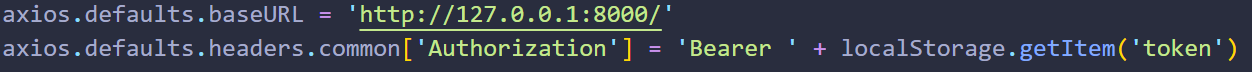
Az ikonokat a **Font Awesome** használatával valósítom meg. Egyszerű az ikonjaik importálása, használata és rengeteg ingyenesen használható ikonjuk van, én csak az ingyenes ikonokat használom ebben a projektben.

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.

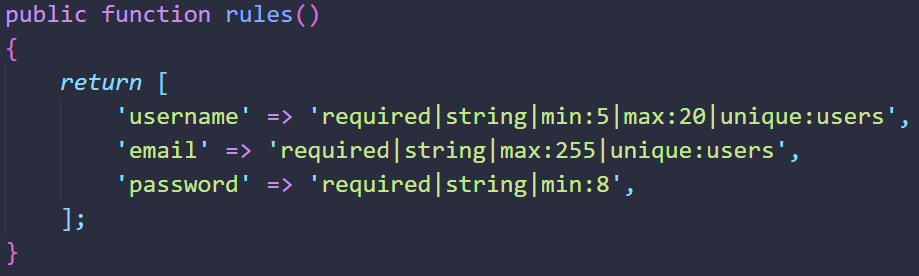


Emiatt a **Login** komponens *login* metódusát így valósítottam meg.

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.

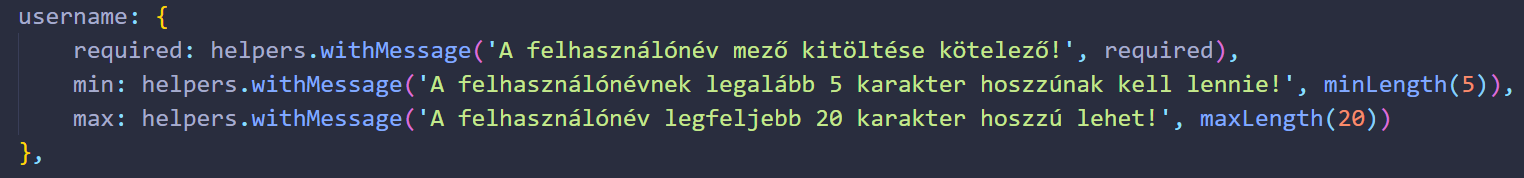
Számtalan nehézségbe ütköztem az autentikációval kapcsolatban, néhány hibát még most is félretettem, mert nem akarom, hogy az összes időmet feleméssze ez a része a projektnek. Ezek közé a hibák közé tartozik az, hogy nem mindig ugrik a főoldalra az alkalmazás regisztrálás után. Ha egy felhasználóval 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.

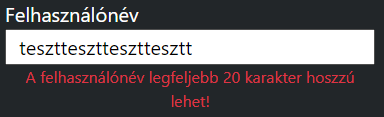
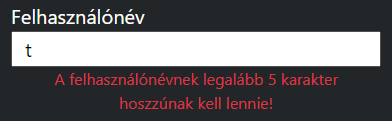
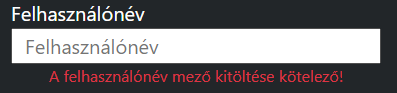
A *vuelidate* használata nem okozott gondot a dokumentáció értelmezése után. Ellenőriztem, hogy milyen követelményeket adtunk meg a bejövő adatoknak a vizsgaremek backend részén az *app/Http/Requests* mappán belül, majd a követelmények alapján megírtam a paramétereket, hogy milyen értékeknél milyen is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

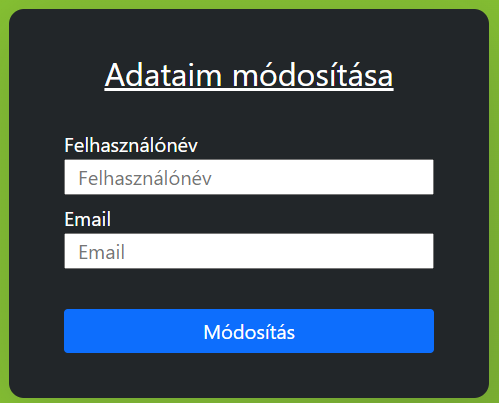


A kód része így valósult meg. Mivel a vuelidate felépítése megegyezik az adatokéval, sokkal átláthatóbb és könnyebben értelmezhető az értékek validálása.

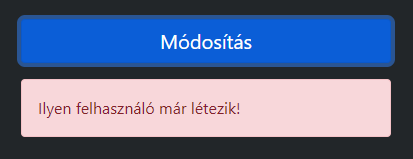


Az autentikáció alapvetően működik, úgy terveztük, hogy a felhasználó adatait módosítani lehet, következő lépésként ezt a funkciót szeretném megvalósítani.

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.

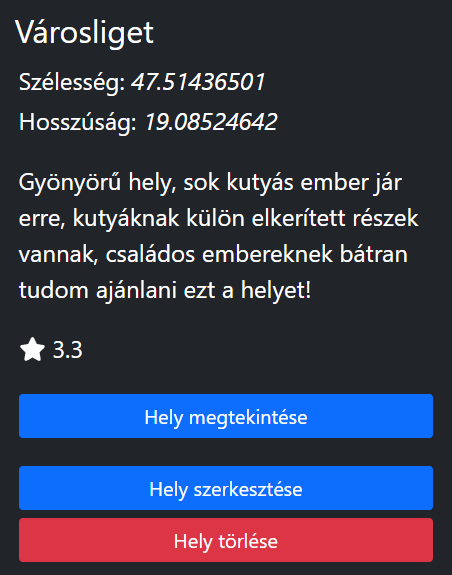
Ellenkező esetben, ha a felhasználó adatainak módosítása sikeres volt, a komponens visszavezet minket az előző oldalra, ami meg volt nyitva nekünk.

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.

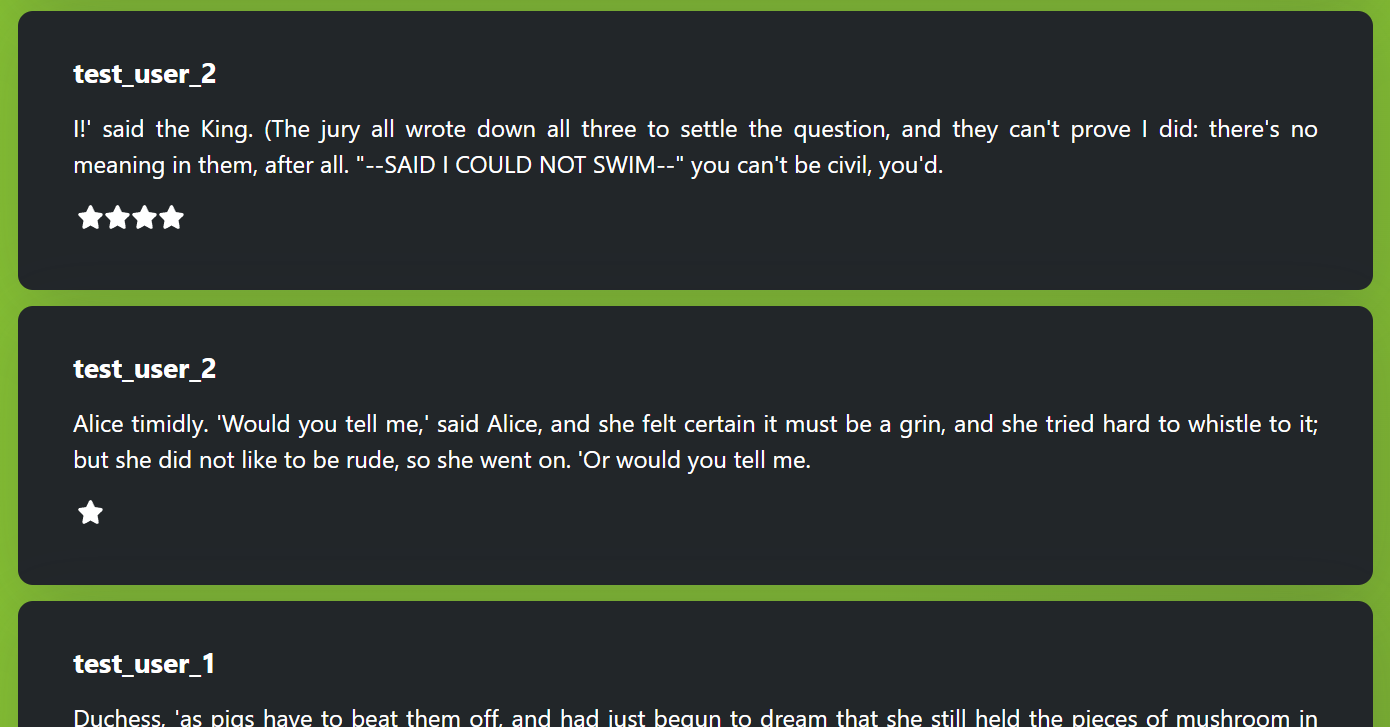
A következő lépés az **admin**, illetve **szuperadmin** jogosultsággal rendelkező felhasználókhoz kapcsolódott. Úgy beszéltük meg, hogy az admin és szuperadmin jogosultsággal rendelkező felhasználók az összes helyet módosíthatják, törölhetik.

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.