**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?

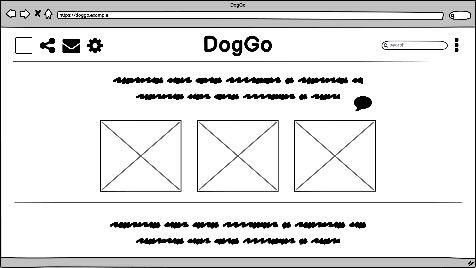
A **Vue** Javascript keretrendszer olyan *feature*-öket tartalmaz, nyújt számomra, amiket már használtam, ismerek. Ezek az előnyök például a többoldalas appok könnyű építése, a *routing*, *komponensek*, adatok *validálása*, számtalan hasznos csomag (*package*), *third party plugin* tartozik a keretrendszerhez (*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.

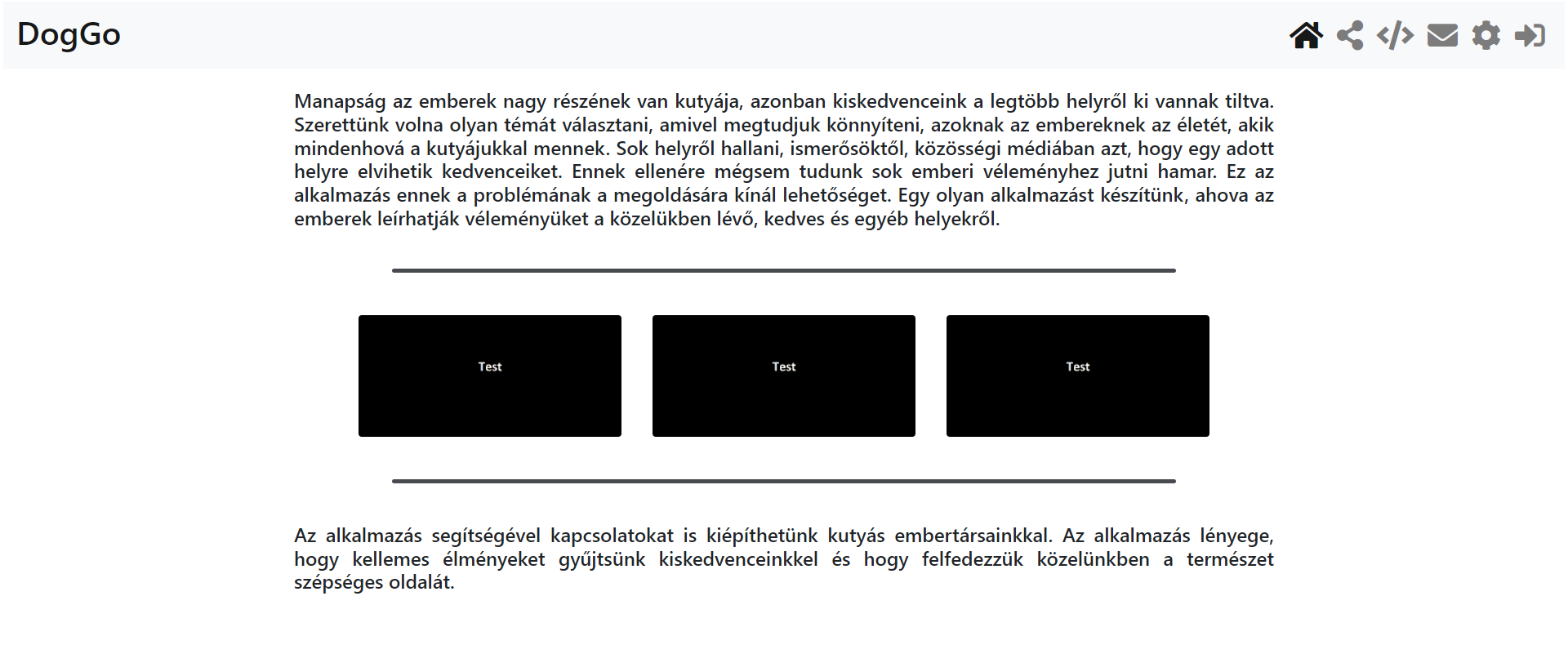
Tudtam, hogy ha elakadok akkor kérhetek segítséget, mert a környezetemben lévő emberek is használták/használják ezt a keretrendszert. Ezeknek a tényezőknek az összessége miatt választottam ezt a keretrendszert, ennek a vizsgaremeknek az elkészítésére

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.

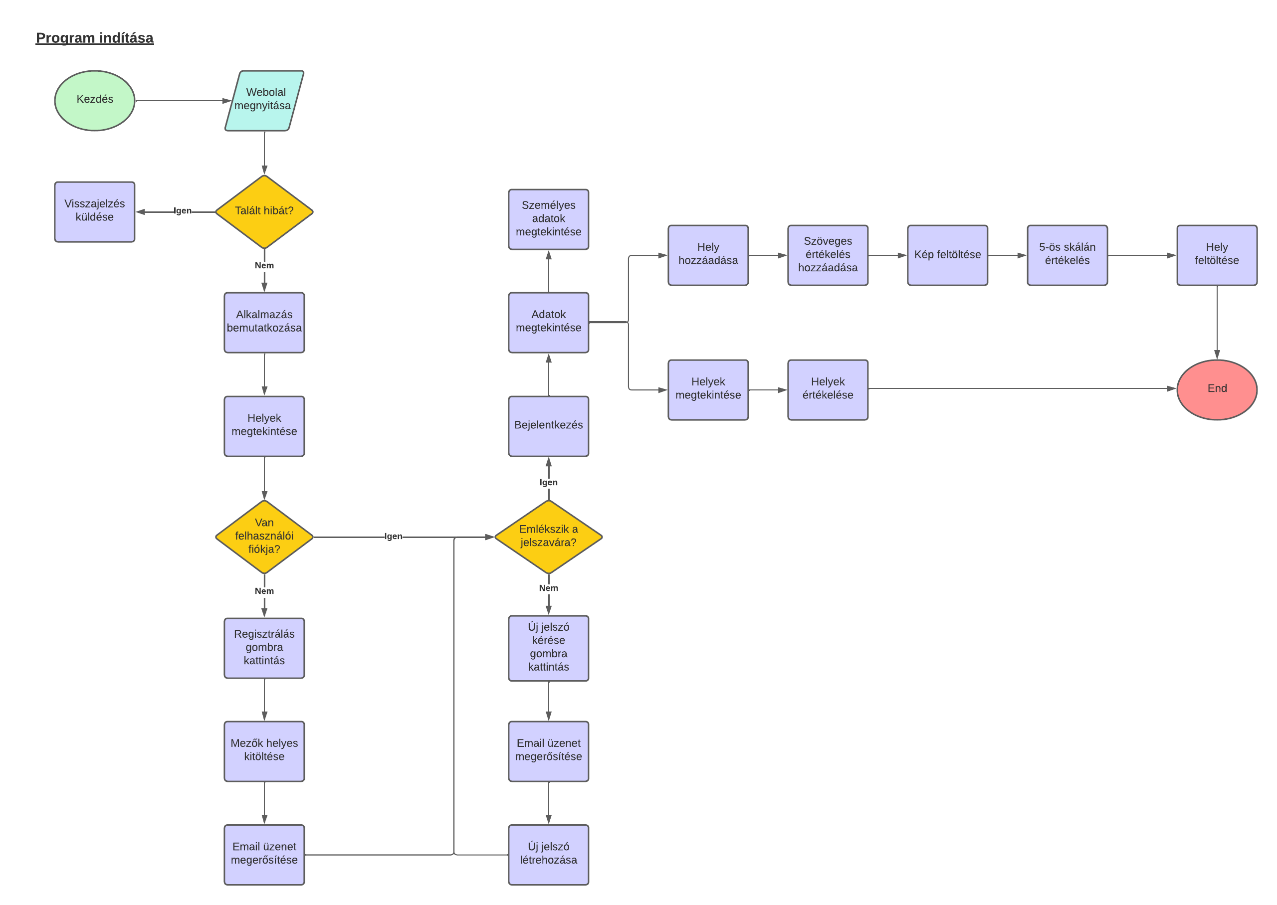


A kezdetleges terv elkészítése után egy sima weboldalt hoztam létre, *HTML* és *CSS* segítségével. Látni szerettem volna, hogy hogy is fog kinézni az alkalmazás.

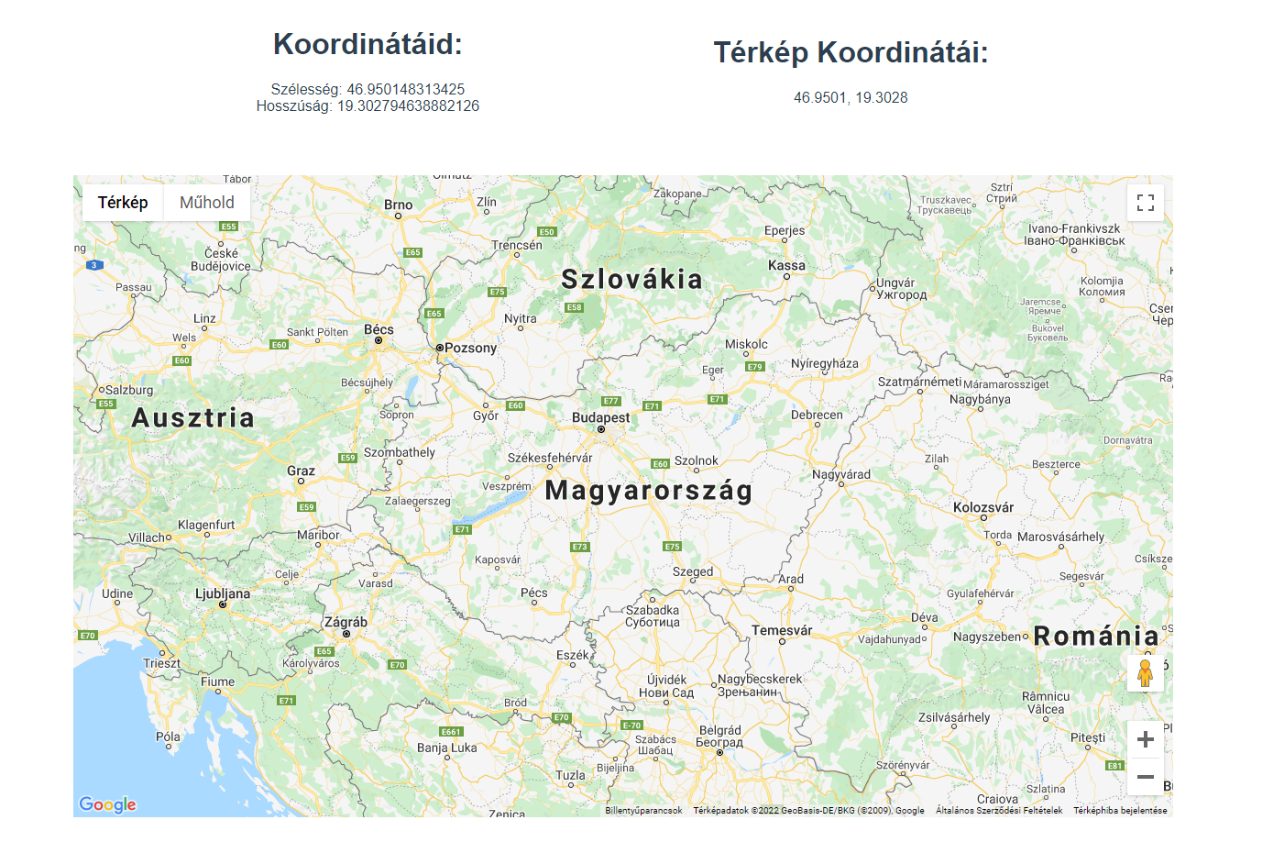


A weboldal tetejére egy navigációs sávot beszéltünk meg. Először magamtól próbáltam meg létrehozni egy reszponzív navigációs sávot. Nem lett olyan az eredmény, mint amilyet elképzeltem. Végül *Bootstrap* segítségével csináltam meg a navigációs sávot, azért is, mert később a Vue keretrendszerhez használhatom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

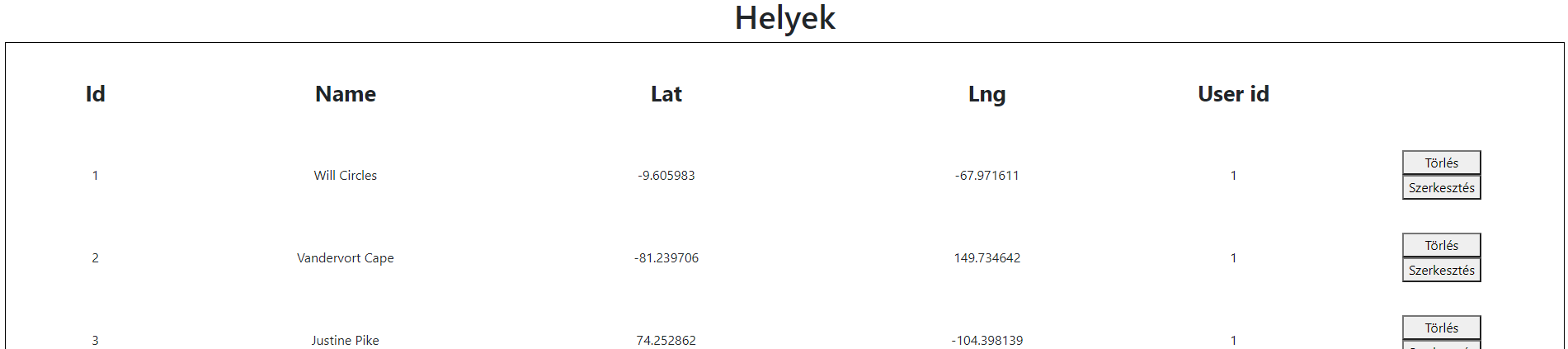
Még mielőtt a keretrendszer segítségével is elkezdtem volna a weboldal elkészítését, az oldal folyamatábráját is meg akartam csinálni.



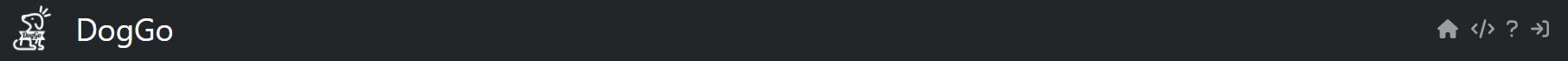
Végül nem a navigációs sáv elkészítésével kezdtem a projektet, hanem a térkép elkészítésével. A *Youtube*-on találtam egy videót, amiben bemutatják a *Google Maps* egyszerű alkalmazását egy weboldalon. A videó végére meg tudtam jeleníteni az aktuális koordinátáimat és a térkép aktuális koordinátáit.



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.



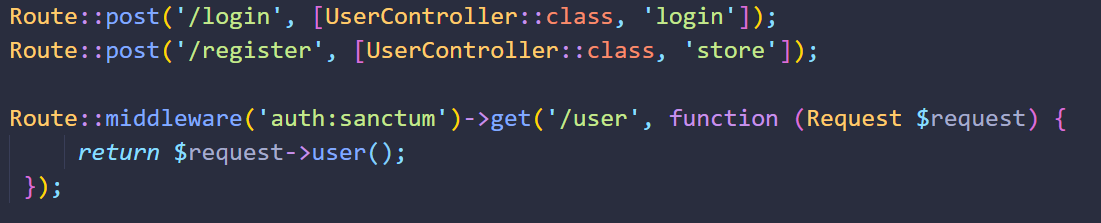
Számomra jó visszaigazolás volt, hogy az app tud a *backend*-del kommunikálni. A kommunikáció sikeres megvalósítása után a bejelentkezést, illetve regisztrálást szerettem volna megvalósítani. A kezdetleges navigációs sáv elkészítése után megpróbáltam a keretrendszer használatával is elkészíteni. Egyelőre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



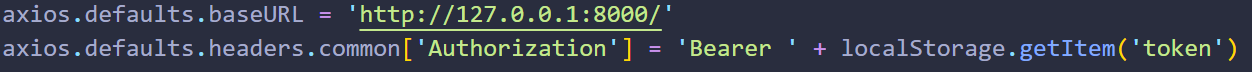
Az ikonokat a **Font Awesome** használatával valósítom meg. Egyszerű az ikonjaik importálása, használata és rengeteg ingyenesen használható ikonjuk van, én csak az ingyenes ikonokat használom ebben a projektben.

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.

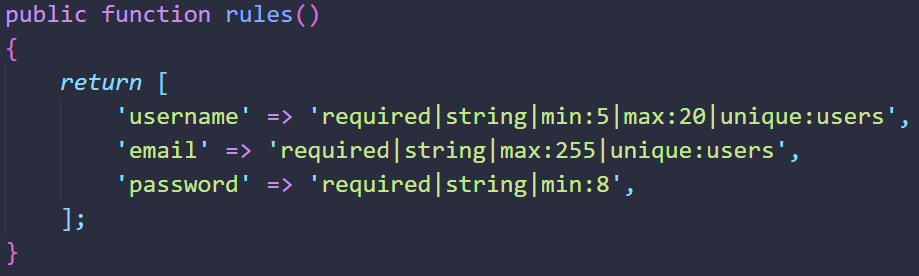


Emiatt a **Login** komponens *login* metódusát így valósítottam meg.

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.

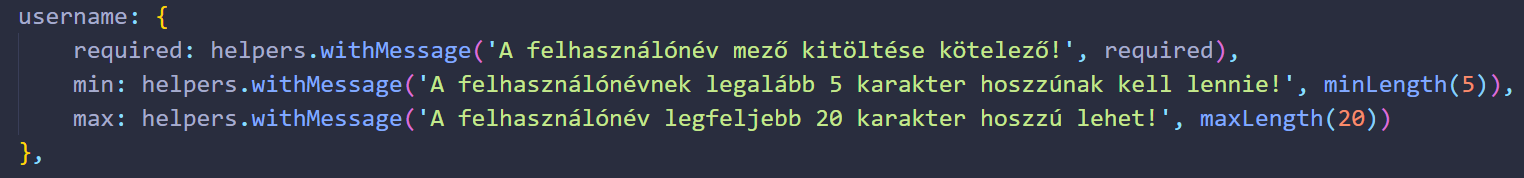
Számtalan nehézségbe ütköztem az autentikációval kapcsolatban, néhány hibát még most is félretettem, mert nem akarom, hogy az összes időmet feleméssze ez a része a projektnek. Ezek közé a hibák közé tartozik az, hogy nem mindig ugrik a főoldalra az alkalmazás regisztrálás után. Ha egy felhasználóval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

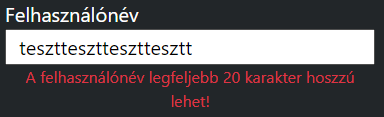
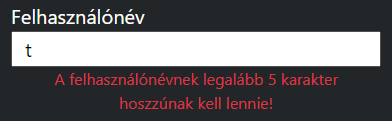
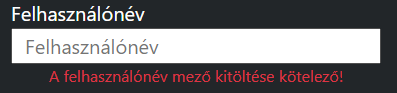
A *vuelidate* használata nem okozott gondot a dokumentáció értelmezése után. Ellenőriztem, hogy milyen követelményeket adtunk meg a bejövő adatoknak a vizsgaremek backend részén az *app/Http/Requests* mappán belül, majd a követelmények alapján megírtam a paramétereket, hogy milyen értékeknél milyen is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

Egy felhasználó felhasználónevének a következő paramétereket állítottuk be:

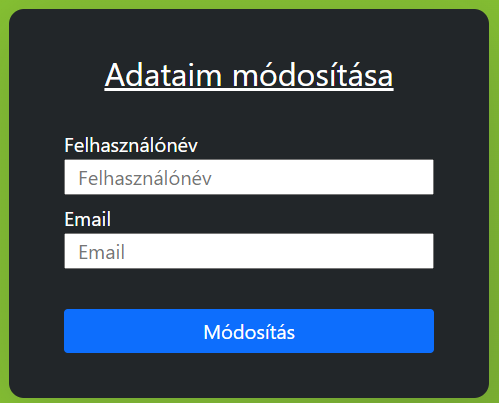
* A felhasználónév mező **kitöltése kötelező**
* A felhasználónévnek legalább **5** karakter hosszúnak kell lennie
* A felhasználónév legfeljebb **20** karakter hosszú lehet



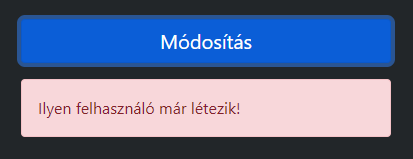
A kód része így valósult meg. Mivel a vuelidate felépítése megegyezik az adatokéval, sokkal átláthatóbb és könnyebben értelmezhető az értékek validálása.



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 **emailcímét** 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.

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.

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.