**SAPIENTIA ERDÉLYI MAGYAR TUDOMÁNYEGYETEM**

**CSÍKSZEREDAI KAR**

**GAZDASÁGI INFORMATIKA SZAK**

**DIPLOMADOLGOZAT**

**ESEMÉNY LÉTREHOZÓ WEBes applikáció  
(Lecrevent)**

**Végzős hallgató:**

**Vántsa Alpár**

**Témavezető:**

**Dr. Pál László, egyetemi docens**

**2023**

**Román kivonat**

**Creator de evenimente aplicație web**

Fusce a metus eu diam varius congue nec nec sapien. Vestibulum orci tortor, sollicitudin ac euismod non, placerat ac augue. Nunc convallis accumsan justo. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas.

Donec malesuada vehicula lectus, viverra sodales ipsum gravida nec. Integer gravida nisi ut magna mollis molestie. Nullam pharetra accumsan sagittis. Proin tristique rhoncus orci, eget vulputate nisi sollicitudin et. Quisque lacus augue, mollis non mollis et, ullamcorper in purus. Morbi et sem orci. Praesent accumsan odio in ante ullamcorper id pellentesque mauris rhoncus. Duis vitae neque dolor.

Duis sed purus at eros bibendum cursus nec a nulla. Donec turpis quam, ultricies id pretium sit amet, gravida eget leo.

Fusce a metus eu diam varius congue nec nec sapien. Vestibulum orci tortor, sollicitudin ac euismod non, placerat ac augue. Nunc convallis accumsan justo. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas.

Donec malesuada vehicula lectus, viverra sodales ipsum gravida nec. Integer gravida nisi ut magna mollis molestie. Nullam pharetra accumsan sagittis. Proin tristique rhoncus orci, eget vulputate nisi sollicitudin et. Quisque lacus augue, mollis non mollis et, ullamcorper in purus. Morbi et sem orci. Praesent accumsan odio in ante ullamcorper id pellentesque mauris rhoncus. Duis vitae neque dolor.

Duis sed purus at eros bibendum cursus nec a nulla. Donec turpis quam, ultricies id pretium sit amet, gravida eget leo.

Fusce a metus eu diam varius congue nec nec sapien. Vestibulum orci tortor, sollicitudin ac euismod non, placerat ac augue. Nunc convallis accumsan justo. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas.

Donec malesuada vehicula lectus, viverra sodales ipsum gravida nec. Integer gravida nisi ut magna mollis molestie. Nullam pharetra accumsan sagittis. Proin tristique rhoncus orci, eget vulputate nisi sollicitudin et. Quisque lacus augue, mollis non mollis et, ullamcorper in purus.

**Angol kivonat**

**Event creator web application**

Fusce a metus eu diam varius congue nec nec sapien. Vestibulum orci tortor, sollicitudin ac euismod non, placerat ac augue. Nunc convallis accumsan justo. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas.

Donec malesuada vehicula lectus, viverra sodales ipsum gravida nec. Integer gravida nisi ut magna mollis molestie. Nullam pharetra accumsan sagittis. Proin tristique rhoncus orci, eget vulputate nisi sollicitudin et. Quisque lacus augue, mollis non mollis et, ullamcorper in purus. Morbi et sem orci. Praesent accumsan odio in ante ullamcorper id pellentesque mauris rhoncus. Duis vitae neque dolor.

Duis sed purus at eros bibendum cursus nec a nulla. Donec turpis quam, ultricies id pretium sit amet, gravida eget leo.

**Tartalomjegyzék**

**[1.](#_Toc534978338)****[Bevezető](#_Toc534978338)** [5](#_Toc534978338)

**[2.](#_Toc534978339)****[Használt technológiák](#_Toc534978339)** [6](#_Toc534978339)

**[2.1. ASP NET Core](#_Toc534978340)** [6](#_Toc534978340)

**[2.2.](#_Toc534978341)****[Javascript - Vue.js - Vuetify](#_Toc534978341)** [6](#_Toc534978341)

**[3.](#_Toc534978342)****[Alkalmazás bemutatása](#_Toc534978342)** [7](#_Toc534978342)

**[3.1.](#_Toc534978343)****[Programozási környezet](#_Toc534978343)** [7](#_Toc534978343)

**[3.2.](#_Toc534978344)****[Adatbázis](#_Toc534978344)** [7](#_Toc534978344)

**[3.3.](#_Toc534978345)****[Az alkalmazás fontosabb funkciói](#_Toc534978345)** [8](#_Toc534978345)

**[3.3.1.](#_Toc534978346)****[Login](#_Toc534978346)** [8](#_Toc534978346)

**[3.3.2.](#_Toc534978347)****[Regisztráció](#_Toc534978347)** [9](#_Toc534978347)

**[4.](#_Toc534978348)****[Következtetések](#_Toc534978348)** [9](#_Toc534978348)

**[Irodalomjegyzék](#_Toc534978349)** [10](#_Toc534978349)

# **Bevezető**

Székelyföld sokszínű és változatos, ez nem csak az emberekre, a környezetre és a hétköznapi életre mondható el, hanem a sportra is. Az itt élő emberek igencsak változatosak, a földrajzi helyzetnek köszönhetően, sokféle sportolási lehetőség adott. A sportnak kitűnő összekovácsoló ereje van, összehozza az embereket, új barátságok szövődnek, emellett egészségen és fitten tarthatjuk magukat. Az online tér egy kítűnő hely, ahol az emberek megismerkedhetnek, összekapcsolódhatnak, az alkalmazásom segítségével az online térben lehetőséget kapnak a felsználók egymás megismerésére, majd ezt követően a való életben találkozhatnak és érezhetik jól magukat.

Projektem legfőbb célja, hogy összehozza az embereket a különböző események által és akár életre szóló barátságokat, élményeket szerezzenek a webes applikáció segítségével. Noha ott van a Facebook, Instagram és még számtalan applikáció amely által a felhasználók értesülhetnek különböző eseményekről, de minden ilyen és az elöbb említett applikációból hiányzik valami, vagy éppenséggel átláthatatlan a felület. Éppen ezért készítettem egy olyan applikációt, amely kizárólag csak az eseményekre összpontosít, emellett pedig egyszerű és átlátható felhasználóbarát felülettel rendelkezik amellyel rengeteg idő megspórolható. Más szempontból mindenkivel előfordult már, hogy a barátai valami miatt éppen nem tudtak részt venni az aznapi focizáson, vagy egyéb közös programon. Az alkalmazás az ilyen esetekre is remek megoldás lehet, hiszen egy esemény létrehozásával máris lehetőséget adunk valakinek, hogy csatlakozzon és ezáltal pótolja a hiányzó helyet.  
 A felhasználóknak egy egyszerű regisztrációt követően lehetőségük van életkor illetve távolság szerint szűrniük a különböző eseményeket, ezáltal a felhasználót nem fogják “zavarni” az olyan események amelyekre amúgy sem lenne lehetősége eljutni, vagy esetlegesen nem az általa preferált korosztály vesz részt az eseményen. Az esemény létrehozásakor a felhasználónak lehetősége van beállítani a szabad helyek számát, ezáltal elkerülhetőek a különböző kellemetlenségek a jövőben.

A dolgozatom készítése során rengeteg új dolgot sikerült megtanulnom, emellett a tanult dolgokat is sikerült jobban elmélyítenem. Ide sorolnám a C# ASP NET Core-t [1] amely által a backendet hoztam létre, illetve a Vue.js [2] keretrendszert amely jelentősen megkönnyítette a felhasználói felület elkészítését. Ezen modern technológiák segítségével egy modern és gyors weboldalt sikerült létrehoznom, amely minden korosztálynak megfelelő segítséget nyújthat új emberek megismerésében és egy jobb, egészségesebb élet kialakításában.

# **Használt technológiák**

A weboldal létrehozásához különböző technológiákat használtam. A front-end, amely az applikáció azon része amellyel a felhasználó közvetlenül találkozik. Ennek alapját a Vue.js Javascript [2] keretrendszer alkotja, melyhez társítottam a Vuetify-t [3] a Vue.js legismertebb felületi keretrendszeré. A felhasználói felület tervezéséhez a Figma tervező programját használtam. A weboldal back-end része, ami egyben a rendszer müködtetője is, ASP NET Core [1] rendszerben született meg. Emellett még használtam, de nem utolsó sorban XAMPP-t [4] amely segítségével kapcsolat jött létre az API illetve az adatbázis között, illetve a Github-nak [5] köszönhetően folyamatos verziókövetést használtam a fejlesztés során, ezáltal a későbbi hibákat egyszerűbben tudtam kijavítani.

## **ASP NET Core**

A weboldalam fejlesztésében az ASP NET Core Web API 7 [1] modelljét használtam, ami akkor még a legújabb verziónak számított, jelenleg a 8-as verzió mondható a legújabbnak. Ez egy keretrendszer az ASP NET Core-hoz [1], amely lehetővé tette a kommunikációt mint a klins, mint a szerver oldallal HTTP protokollt használva. Rengeteg beépített funkciója van, mint pl. a hitelesítő, de session-öket illetve automatikus adatbáziskapcsolódást is lehetővé teszi.

Az ASP NET Core MVC [6] keretrendszer segítségével az alkalmazás ezen részén is átláthatóbb, könnyebben érthető kód jön létre. Az MVC a Model-View-Controller (modell - nézet - vezérlő ) komponensekből áll. A modellben tárolódnak és kezelődnek az adatok, amelyek lehetnek file-okban tárolt adatok, vagy adatbázisból származó adatok. A nézetnek az a szerepe, hogy megjelenítse a különböző adatokat, illetve biztosítja az alkalmazás felhasználói felületét amellyel a felhasználó közvetlenül kapcsolatba kerül, itt jelennek meg az adatok és itt lép interakcióba a felhasználó a weboldallal. A vezérlő összekapcsolja a nézetet és a modellt. Legfőbb feladata a válaszok visszaküldése és a kérések kezelése.

Ahhoz, hogy kapcsolatot jöjjön létre a C# és MySql között, a C# Entity Framework [7] keretrendszert használtam. A keretrendszernek köszönhetően zökkenőmentesen kapcsolatot tudtam teremteni a MySql és az .Net applikációm között. Előnye az, hogy a kódban végzett változtatások automatikus változást hoznak az adatbázisban is egyaránt. További előnynek mondható, hogy a verziókövetésre van lehetőség, ezáltal könnyen lekövethető az adatbázisban történt változások.

A felhasználó adatai megörzése érdekében Json Web Tokeneket (JWT) [8] használtam. Ezeknek a tokeneknek a felhasználók azonosításában van szerepük, ezáltal a rendszer megjegyzi egy a már előre bejelentkezett felhasználó adatait. Egy token 3 részből tevődik össze, ezen típusok a következők: fejléc (header) amely információt tárol a token típusáról (typ) és az algoritmusról (alg), adatok (payload) itt azon adatok szerepelnek, amelyeket előre meghatározva a backend-en beállítottam pl. felhasználónév, email cím, életkor, stb.. Az utolsó típusa az aláírás (signature) az algoritmus alkalmazásával keletkezik egy kód, amelyet tárol és ennek a segítségével ellenőri a token valódiságát.

## **Javascript - Vue.js - Vuetify**

A Javascript egy objektumorientált gyengén típusú programozási nyelv, amely napjaink egyik elengedhetetlen kelléke a webfejlesztések során. Általa a fejlesztők dinamikus és interaktív weboldalakat hozhatnak létre, emellett különböző funkciókat, adatkezelést vagy akár animációkat is hozzáaadhatnak a weboldalhoz, amelyek megkérdőjelezhetetlenül növelik a felhasználói élményt. A Javascript kódja futtatható a böngészőben, illetve NodeJs [9] segítségével akár a saját operációs rendszerünken is futtathatóak a scriptek.

A Vue.js [2] a JavaScript egyik legismertebb keretrendszere. Legfőbb feladata, hogy megkönnyítse a fejlesztő dolgát, mivel a natív JavaScript-ben leprogramozni különféle feladatokat jelentősen több időt vesznek igénybe. A fejlesztés során bármely változás a kódban, valós idejű változásként jelenik meg a fejlesztő képernyőjén, amely igencsak megkönnyíti a fejlesztést, ez annak köszönhető, hogy a Vue.js [2] virtuális DOM-ot használ. A Route-oknak köszönhetően több oldalas weboldalakat is lehetőségünk van létrehozni, a komponensek segítségével pedig olyan elemeket hozhatunk létre, amelyek előreláthatólag tudunk, hogy többszőr is felfogunk használni (pl. header, footer).

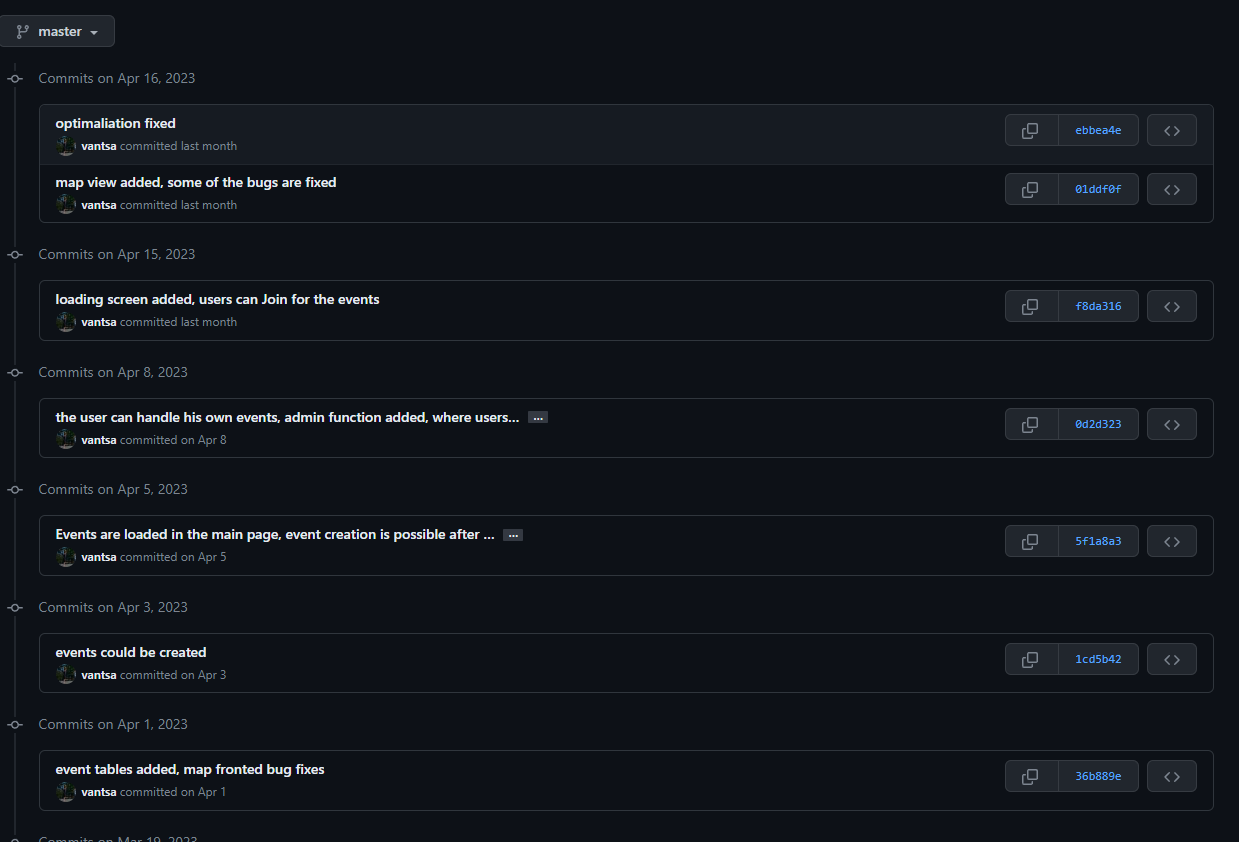
A Vuetify [3] egy komponenskönyvtár amelyet a Vue.js-hez használunk. Lehetővé teszi a fejlesztő számára, hogy korszerű, gyors weboldalt készítsen rövid idő alatt. A Vuetify [3] előre elkészített komponenseket tartalmaz (gombok, kártyák, űrlapok, stb.). Ezen komponensek szabadon felhasználhatóak és elérhetőek, illetve ami a legfontosabb a fejlesztők saját igényeik szerint személyre szabhatják azokat. Ezek az előre legyártott komponensek rendelkeznek különféle attribútomokkal, amelyek segítségével könnyebben és gyorsabban személyre szabható az adott komponens.

## **MySql/Xampp**

A MySql [10] egy adatbáziskezelő rendszer amely az egyik legelterjedtebb a piacon, a webes alkalmazások egyik alapvető pillére. Legfontosabb feladata az adatok tárolása, emellett különböző müveleteket végezhetőek mint például adatok törlése, frissítése, lekérdezése, beszúrása.   
 Az Xampp [4] egy nyílt forrású webes szerveralkalmazás, amely biztosítja a MqSql megfelelő működését a fejlesztő saját rendszerén. Meghatározó elemei az Apache webszerver, MariaDB, és a PHP nyelv futtatható rendszere. Annak ellenére, hogy az Xampp [4] legfőképpen PHP nyelvben megírt kódokra van kifejlesztve, az ASP .NET Correal való összekapcsolás során semmilyen problémába nem ütköztem.

## **Github**

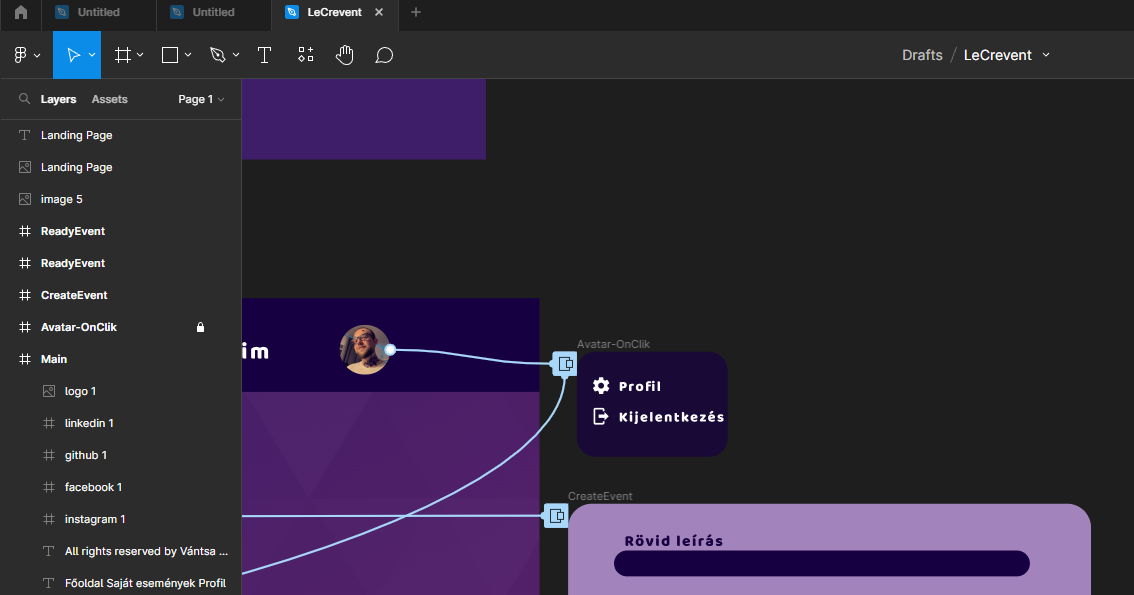
A Github [5] egy olyan felhő alapú szolgáltatás amely lehetőve teszi a fejlesztők számára a saját kódjaik tárolását illetve kezelését, emellett lehetővé teszi, hogy több ember egyszerre dolgozzon ugyanazon a projekten egyidőben, anélkül, hogy problémába ütköznének. Legjelentősebb funkciója a verziókövetés, amely lehetővé teszi a fejlesztők számára a kód felügyeletét illetve a változások követését. A Github [5] lehetőséget ad arra, hogy a meglévő kódot duplikáljuk egy külön ágra, majd a kívánt fejlesztés végrehajtásával, a főszálra visszavezetni a kódot, amely tartalmaz minden szükséges dolgot. Fontos megjegyezni, az adatok felhőben tárolódnak így bármilyen esetleges számítógép meghibásodás során nem érhet kódveszteség. A melléket ábrán (2.1 Ábra) látható az általam használt verziókövetes.



2.1. Ábra. Github verziókövetés

## **Figma**

A Figma [11] egy felhő alapú grafikus felületek tervezésére alkalmas platform. Legfőbb célja, a webes és mobilapplikációk megtervezésének leegyszerűsítése. Kezelőfelülete egyszerű, használata pár óra gyakorlás után teljesen elegendő egy kezdő webdesigner számára. Fontosnak tartottam egy tervezői eszköz használatát az applikációm megtervezéséhez, mivel ezáltal a késöbbiekben már egy kész elképzelés alapján tudtam a weboldalamat fejleszteni, ezáltal elkerültem a késöbbiekben felmerülő bizonytalanságokat az alkalmazásom felhasználói felületével kapcsolatban.

  
2.2. Ábra. Tervezés a Figma rendszerében

## **Node Package Manager (NPM)**

A Node Package Manager [12] egy könyvtárrendszer, amely lehetővé teszi különböző JavaScript csomagok telepítését és használatát a saját kódunkban ezáltal gyorsabb és egyszerübb is a fejlesztés.

Az applikációm fejlesztése során a következő csomagokat használtam:

* vue-axios [13] : api hivások kezelése, kapcsolatteremtés frontend-backend között
* vue-cors [14]: erőforrás kezelése, személyreszabott fejléc engedélyezése
* Eslint [15]: jelzést ad a potenciális hibákról (formázás, stílus, logika)
* jwt-decode [16]: a JWT-ben tárolt adatok dekódolására szolgál
* vue-leaflet [17]: egyszerű és korszerű térképkezelő, amely lehetővé tette a térkép megjelenítését
* vue-leaflet-markercluster: lehetővé tette pontok megjelőlését a térképen
* vue-router [18]: útvonalkezelésére szolgál
* vue-the-mask [19]: form-ok kitöltésénél formázást és korlátozásokat ellenőriz
* Vuetify [3]: Vue.js alapú előre legyártott komponensek könyvtára

# **Alkalmazás bemutatása**

Egy esemény szervező alkalmazásnak elengedhetlen része, hogy adatbázisban tárolódjanak a különféle adatok, ezért is volt szükség az említett technológiák használatára. Egy ilyen típusú alkalmazásnak úgy gondolom kötelezően tartalmaznia kell regisztrációhoz kötött azonosítást. A következőkben az alkalmazás részletes bemutatása következik, ahol külön kitérek a fejlesztés során felmerült nehézségekre, érdekességekre, illetve olyan dolgokra amelyek nélkül nem jöhetett volna létre az alkalmazás.

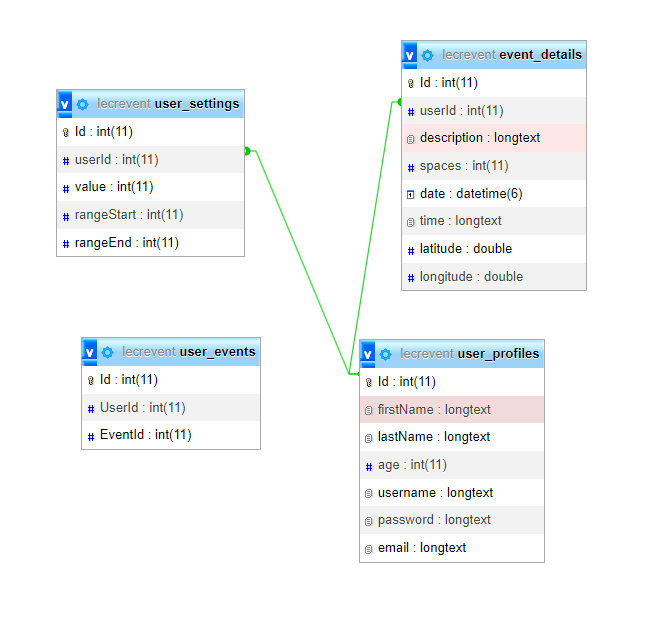
## **Programozási környezet**

Az alkalmazás fejlesztése kezdetlegesen csak és kizárólag a Visual Studio Code fejlesztői környezetében történt. Számomra az első pillanattól kezdődően egyértelmű volt, hogy ebben a kódszerkesztőben fogok dolgozni. Az egyik legfontosabb szempont számomra az az átláthatósága az adott környezetnek, amely a fent említett Visual Studio Code-nak az egyik legprofibban kivitelezett rendszere van, emellett számtalan bővítmény telepíthető amelyek megkönnyítették a fejlesztés során a dolgomat. Ilyen bővítmények például a szintaxisok kiemelése, automatikus kódkiegészítés, hibakeresés. Tovább a Git támogatásnak köszönhetően könnyen és gyorsan feltölthető a kód Github felhőjébe.

Mint említettem kezdetlegesen csak a Visual Studio Code környezetét használtam, de az idő múlásával, a tesztelések számának növekedésével kénytelen voltam a backend fejlesztését (ASP .NET Core) átvinnem a Microsoft által fejlesztett Microsoft Visual Studióba amely eredetileg a .NET Core kódszerkesztője. Az elöbbi kódszerkesztő csupán annyiban különbözik a Visual Studio Code-tól, hogy egy teljes körű fejlesztő környezet, amelyben mobil, windows, web és felhőalapú alkalmazások fejlesztése lehetséges.

## **Adatbázis**

A 3.1 Ábrán látható az alkalmazásom adatbázisa, amely 4 táblából áll. Ezekben a táblákban tárolódnak a különböző adatok, mint például a felhasználók adatai, és beállításai illetve az események adatai. Az alkalmazás fejlesztése során különböző triggereket is használtam, amelyeket a táblák bemutatása után részletesebben kibontok.



3.1. Ábra. Az alkalmazás adatbázis sémája

A ‘user\_profiles’ tábla tartalmazza a felhasználó adatait amelyet a regisztráció során adott meg. Az Id mező automatikusan növekvő (Auto Increment) ezáltal nem áll fent a lehetősége az ismétlődésnek. A felhasználónév és az email egyedi, ez biztosítja azt, hogy egy felhasználó csak egy email címről tudnak regisztálni az alkalmazásra. A jelszó titkosítva van, ezáltal a felhasználón kivül senki más nem tudhatja a jelszavát. A felhasználó életkora egy int típusú adat amelyet a felhasználó születési dátumként ad meg, majd ezt követően int tipussá alakítva mentődik el az adatbázisban.

|  |  |  |
| --- | --- | --- |
| **Attríbutum neve** | **Típus** | **Attríbutum szerepe** |
| id | int, primary, unique | Általános ID |
| firstName | varchar | Felhasználó vezetékneve |
| lastName | varchar | Felhasználó keresztneve |
| age | int | Felhasználó életkora |
| username | varchar, unique | Felhasználónév |
| password | varchar | Felhasználó jelszava(hash) |
| email | varchar, unique | Felhasználó email címe |

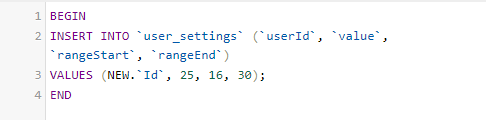
3.1. Táblázat user\_profiles tábla

A ‘user\_settings’ tábla tartalmazza a felhasználó beállításait. Ebben a táblában levő értékeket a felhasználó bejelentkezés után tudja frissíteni. A userId segítségével tudhatjuk, hogy melyik felhasználó beállításai fognak változni, vagy éppen melyik felhasználó beállítását látjuk. A ‘value’ érték meghatározza azt a kilométeres körzetet, amelyben a felhasználó szeretné látni a közelendő eseményeket. A ‘rangeStart’ illetve ‘rangeEnd’ attríbutumok egy olyan intervallum alső és felső határai amelyek által a felhasználónak lehetősége van beállítani azt, hogy mely az a korosztály akinek szeretné az eseményeit látni. Ezen adatok a felhasználó számára személyre szabhatóak.

|  |  |  |
| --- | --- | --- |
| **Attríbutum neve** | **Típus** | **Attríbutum szerepe** |
| id | int, primary | Általános ID |
| userId | int, foreign key | Felhasználó ID-je |
| value | int | Maximális km körzet |
| rangeStart | int | Minimális életkor |
| rangeEnd | int | Maximális életkor |

3.2. Táblázat user\_settings tábla

Regisztrációt követően egy trigger segítségével beállítja a default(alap) értékeket az adott felhasználónak.



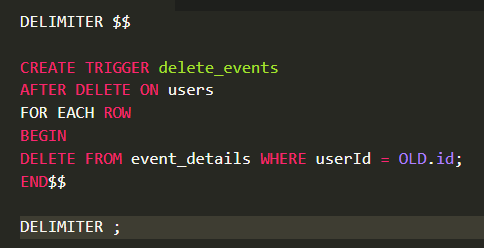
3.2. Ábra. Regisztrációt követő INSERT trigger

Az ‘event\_details’ tábla tartalmazza az eseményeket és az azokhoz tartozó adatokat. A ‘userId’ által tudjuk melyik felhasználó hozta létre az adott eseményt. A ‘description’ szolgál az esemény pontosabb ismertetésére. A ‘spaces’ arra szolgál, hogy megmutatja hány férőhely van az adott eseményre, ez az adat dinamikusan változik amikor egy felhasználó csatlakozik az eseményhez, vagy éppen visszalép az adott eseményről. A ‘date’ és a ‘time’ attríbutumok szolgálnak az esemény pontos idejének meghatározására. A ‘latitude’ és ‘longitude’ adatok a térkép által meghatározott koordinátákat tárolják ,ezek által a Leaflet könnyedén tudja menteni illetve lekérni az esemény helyszínének koordinátáit.

|  |  |  |
| --- | --- | --- |
| **Attríbutum neve** | **Típus** | **Attríbutum szerepe** |
| Id | int, primary | Általános ID |
| userId | int, foreign | Felhasználó ID-je |
| description | longtext | Esemény címe |
| spaces | int | Elérhető helyek száma |
| date | datetime | dátum |
| time | longtext | Pontos idő (óra:perc) |
| latitude | double | szélesség |
| longitude | double | hosszúság |

* 1. Táblázat event\_details tábla

Egy felhasználó esetleg törlése során, minden felhasználó tartozó adat törlődik, beleértve az összes eseményt amelyet az adott felhasználó hozott létre. Ennek a kivitelezéséhez ‘DELETE’ triggert használtam amely a mellékelt (3.3. Ábra) ábrán is látható.



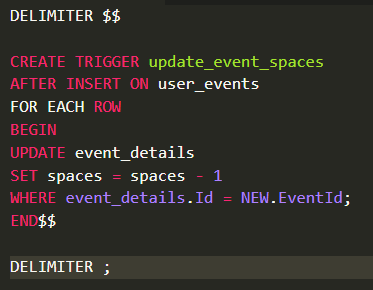
3.3. Ábra. Felhasználó törlését követő DELETE trigger

A ‘user\_events’ tábla tartalmazza mindazokat a felhasználokat és eseményeket amelyekhez csatalkozott már felhasználó. Ennek a táblának köszönhetően lehet tudni, hogy hány elérhető hely van egy adott eseményhez, vagy éppen azt, hogy az adott felhasználó csatlakozott-e már az adott eseményhez

|  |  |  |
| --- | --- | --- |
| **Attríbutum neve** | **Típus** | **Attríbutum szerepe** |
| Id | int, primary | Általános ID |
| UserId | int | Felhasználó ID-je |
| EventId | int | Esemény ID-je |

* 1. Táblázat user\_events tábla

Ha a ‘user\_events’ táblába egy új adat kerül beszúrásra akkor az ‘event\_details’ táblában levő ‘spaces’ mező frissül. Egy új adat beszúrása azt jelenti, hogy egy felhasználó csatlakozott egy eseményhez, így az adott esemény azonosító alapján a ‘spaces’ mező értéke csökkeni fog eggyel, természetesen törlés esetén ez az érték növekszik eggyel. A spaces mező értéke soha nem lehet negatív szám. Ezeknek a müveleteknek a megvalósításához triggereket használtam, a mellékelt (3.4 Ábra) egy ilyen trigger kódját mutatja be.



3.3. Ábra. Beszúrás után ‘spaces’ érték változása

A triggerek segítségével még színesebbé és gyorsabbá tudtam tenni úgy az alkalmazásom mint a hozzá tartozó adatbázist. Az ASP .NET Core-ban használt Entity Framework [7] keretrendszernek köszönhetően a folyamatos kódváltozások backenden dinamikusan megjelentek az adatbázisban is egyaránt.

## **Az alkalmazás fontosabb funkciói**

Az alkalmazás fontosabb funkcióit fogom bemutatni a következő részben. Elsősorban a regisztrációt, bejelentkezést, majd az események létrehozását illetve azok kezelését mutatom be. Ezt követően a felhasználó beállításai, illetve az adminfelület kerül bemutatásra.

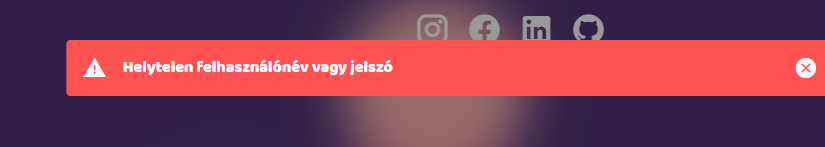
### **Login**

A bejelentkezés a főoldalról lehetséges. Bejelentkezés nélkül a felhasználó nem kap engedélyt, hogy megtekintse a létrehozott eseményeket. A form-nak lazy validation van beállítva, ami azt jelenti, hogy az adatok csak azután kerülnek ellenőrzésre miután a felhasználó megnyomta a ‘Bejelentkezés’ gombot. A real time validation annyiban tér el az elöbbitől, hogy az minden gombleütés után választ küld a szerver fele. A felhasználónév helyére a felhasználó megkell adja a regisztrációkor megadott felhasználónevet. Majd ezt követően a jelszavát amelyet ugyancsak a regisztrációkor adott meg. A jelszó beírása titkosítva van ‘\*’ jelek használatával, viszont a láthatóság ikonra kattintva ez könnyedén változtatható, így a felhasználó bármikor ellenőrízheti a beírt jelszó helyességét, mielött megpróbálna bejelentkezni.



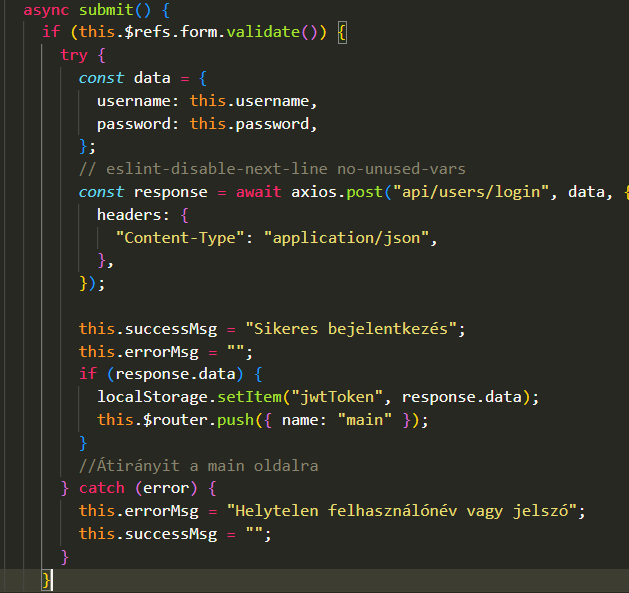
3.4.Ábra. Login ablak

Sikertelen bejelentkezés esetén a felhasználó hibaüzenettel találkozik amelyet a mellékelt ábra is mutat (3.5. Ábra) amely a kijelző jobb felső sarkában jelenik meg.



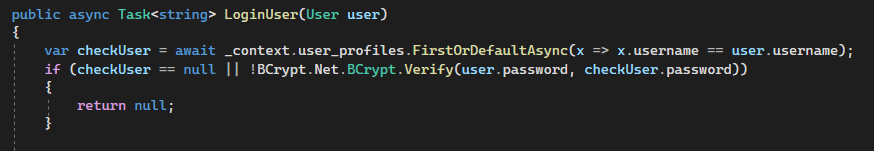
3.5.Ábra. Login hibaüzenet

A bejelentkezés gomb megnyomásával a ‘submit’ metódus hívodik meg, amelyben eltárolom a felhasználó által beírt adatok, majd axios segítségével elküldöm a backend számára. Sikeres bejelentkezés esetén átirányul a főoldalra, ellenkező esetben hibaüzenet jelenik meg.



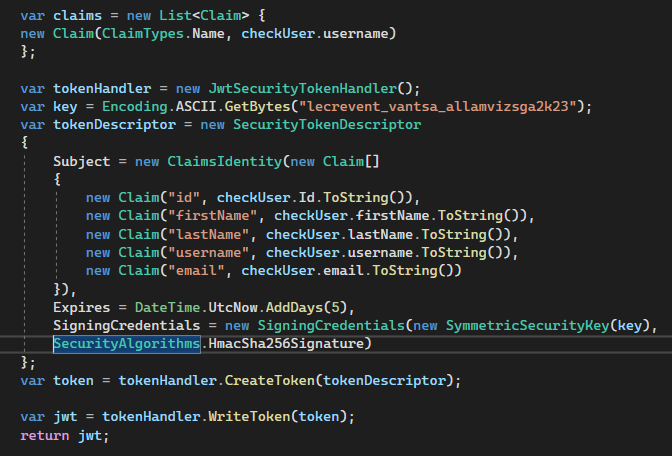
3.6.Ábra. Felhasználó adatok küldése a backend számára

Amint a táblák bemutatása során is említettem, a jelszavak teljesen védve vannak, így a bejelentkezés során elküldött jelszavat enkódolnia kell.



3.7.Ábra. Felhasználónév és jelszó ellenőrzése

Ha létezik a felhasználónév és a jelszavak is megegyeznek, szükséges egy token létrehozása. Ahogyan a technológiáknál is említettem a Json Web Token-t használtam, Személyre szabottan azt lehet menteni a tokenben amire éppen szükség van. Ebben az esetben én a tokenbe mentettem a felhasználó id-ját, vezetéknevét, keresztnevét, felhasználónevét illetve email címjét.

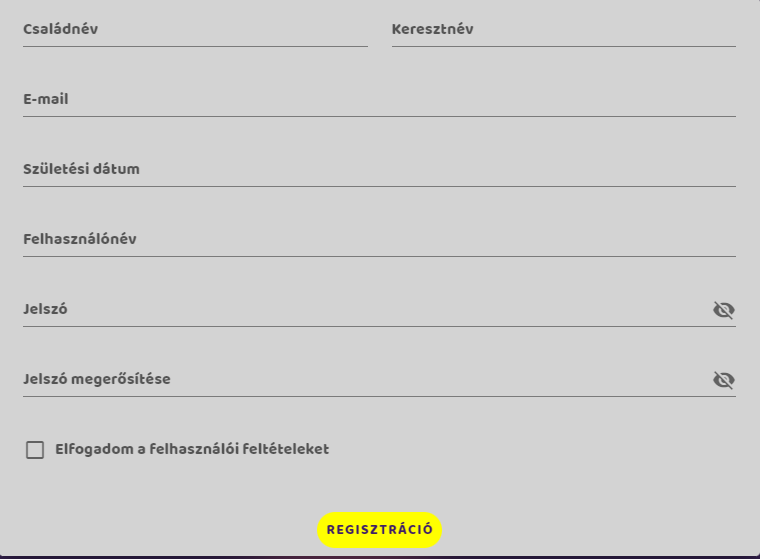


3.7.Ábra. JWT token generálása

Az ‘Expires’ értéknek megadhatjuk, hogy meddig szeretnénk, hogy érvényes legyen az adott token. A ‘SigningCredentials’ mezőnek két értéket kell megadni, elsőként egy személyre szabott jelszavat, majd az algoritmus nevét amellyel szeretnénk kódolni a tokent. Ezt követően a token-t visszaküldjük a böngészőnek, amelyet a megadott ‘Expires’ értéknek megfelelő ideig el is ment a böngésző.

### **Regisztráció**

A weboldal bármely funkciójának használatához szükséges a regisztráció. A regisztráció által a felhasználó tud létrehozni egy saját azonosítot, amellyel a késöbbiekben betud jelentkezni, így lehetősége adódik eseményeket létrehozni, vagy akár más felhasználó által létrehozott események csatlakozásához. A regisztráció elindítása a főoldaról lehetséges amelyet a ‘Regisztráció’ gomb megnyomásával érhetünk el. Ezt követően egy dialog típusú ablak fogadja majd a felhasználót (3.8 Ábra), ahol a felhasználónak kikell töltenie a regisztrációs ívet a megfelelő módon.

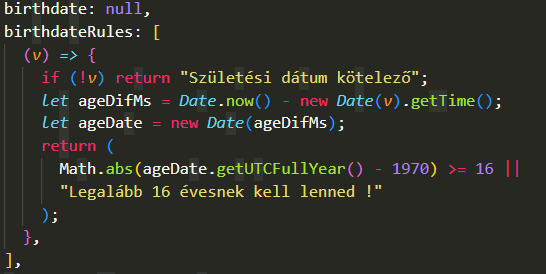


3.8.Ábra. Regisztrációs ablak

A regisztráció során a felhasználó különböző kritériumoknak kell megfelelnie. Első sorban minden egyes mezőt kikell töltenie. Minden mezőre formai ellenőrzés van alkalmazva. Az email kötelezően tartalmaznia kell egy ‘@’ szimbólumot, emellett nem szerepelhet a kitöltött mező sem első sem utolsó karaktereként. A születési dátum esetén egy dialog ablakból tudja kiválasztani a felhasználó a születési dátumát. Ez a dialog ablak egy vuetify által előre legenerrált komponens (3.9. Ábra). Továbbá a felhasználó kötelezően 16. életévét betöltött állampolgár kell legyen, amelyet a következőképp ellenőriztem. (3.10. Ábra). Első sorban leellenőrzöm, hogy az adott mező üres-e. Ezt követően a jelenlegi dátum segítségével leellenőrzöm a felhasználó életkorát. A Math.abs függvény segítségével a dátum abszolút értékét határozom meg kivonva belőle az 1970-et, amely a számítógépes rendszererek időszámlálásának indulását jelzi.



3.9. Ábra Vuetify naptár komponens



3.10. Ábra Életkor ellenőrzés

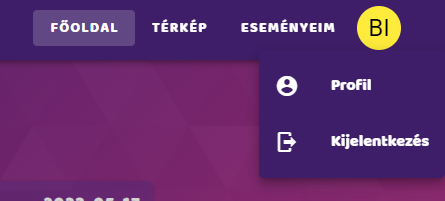
Ezt követően egy egyedi felhasználónév megadására van szükség, amely a weboldalon fog megjelenni mint azonsító, ez alapján fogják tudni egymást beazonosítani a felhasználók. Ha az adott felhasználónév már létezik, a regisztárciót megkell ismételni, természetesen egy másik felhasználónévvel. A jelszó legalább 8 karakter hosszúságú kell legyen, illetve a jelszó megerősítésénél a beírt jelszó megkell hogy egyezzen az előzőleg megadott jelszóval. A felhasználói feltételek elfogadása a ‘checkbox’ mező bepipálásával lehetséges. Az ‘Elfogadom a felhasználási feltételeket’ szövegre kattintva egy új oldalon megjelnik egy általános felhasználási feltétel, amelyet a TermsFeed oldal segítségével generáltam le. Ennek az oldalnak a megnyitásával a felhasználási feltételek automatikusan elfogadottnak vevődik. A felhasználói feltételek visszautasítása, sikertelen regisztrációhoz vezet. Ezt követően a dialógusban szereplő ‘Regisztráció’ gomb megnyomásával a felhasználó befejezi a regisztrációját. Ezt követően a felhasználó értesítést kap a regisztráció sikerességét illetően. Ha a regisztráció sikeresen bezárult egy zöld ablak, ellenkező esetben egy piros ablak fog megjelenni a kijelző jobb felső sarkában (3.10. Ábra).



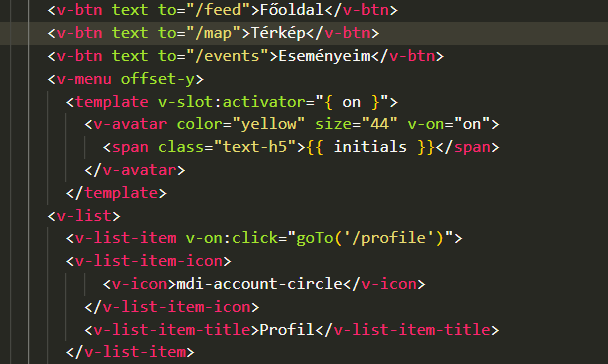
3.10. Ábra Sikeres regisztráció értesítés

### **Menü rendszer**

A menüsávban három opció látható. A főoldal, ahol jelenleg a felhasználó tartózkodik, egy ‘Térkép’, ahol minden esemény látható amely a weboldalon volt létrehozva, egy ‘Eseményeim’ fül, amelyre kattintva a felhasználó a saját eseményeit tudja létrehozni. (A menüben az adott pont, ahol a felhasználó tartózkodik a weboldalon, világosabb hátérrel van megjelenítve. Emellet a felhasználó a jobb felső sarokban látható ikonra kattintva egy menüt hozhat elő, ahol lehetősége adódik szerkeszteni a saját profilját, illetve kijelentkezni az alkalmazásból (3.11. Ábra). A lenyíló menü a Vuetify könyvtárából származik, amelyet az oldal igényei szerint személyre szabtam. A következő ábrán látható egy kódrészlet a menü felépítéséről (3.12. Ábra).



3.11. Ábra Menü rendszer



3.12. Ábra Kódrészlet a menü felépítéséről

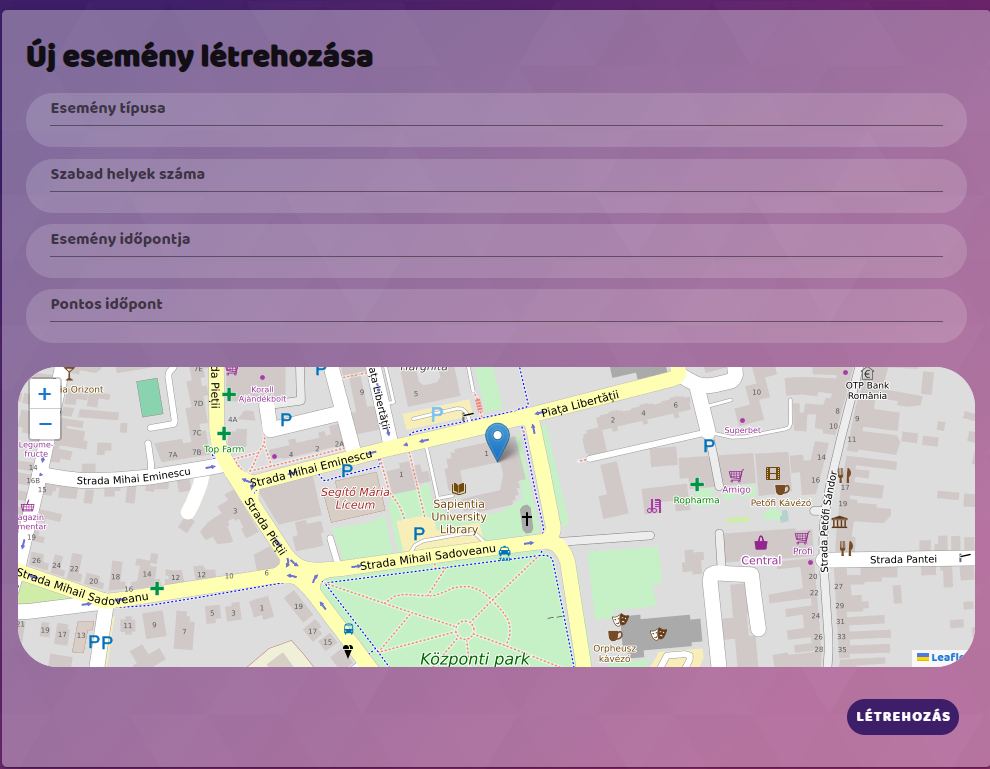
### **Új esemény létrehozása**

Új esemény létrehozása a főoldalról lehetséges. A gomb közvetlenül a létrehozott események fölött helyezkedik el (3.13 Ábra).



3.13. Ábra Esemény létrehozó gomb

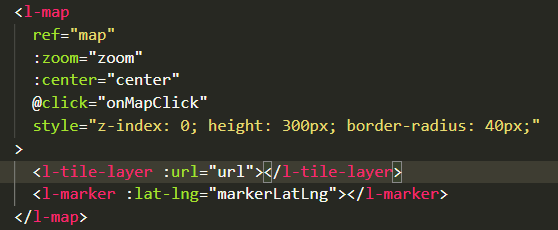
A gomb megnyomásával egy lenyíló ablak jelenik meg, ahol a felhasználó létrehozhatja az eseményt a szükséges adatok kitöltésével (3.14. Ábra).



3.14. Ábra Esemény létrehozása ablak

Az ‘Esemény típusa’ fülhöz a felhasználó egy rövid leírást kell megadjon a létrehozni kívánt eseményről, amely legalább 4 karakterhosszúságú kell legyen. A ‘Szabad helyek száma’ fülhöz, az elérhető helyek számát kell megadnia, pontosabban azt, hogy hány szeméyt szeretne hogy csatlakozzon az eseményhez. Az ‘Esemény időpontja’ fülnél az esemény dátumát kell megadni (év, hónap, nap). Amely ugyancsak a Vuetify dialógus komponensét használja. Majd ezt követően a ‘Pontos időpont’ fülhöz kell megadnia a pontos dátumot (óra:perc). Az előző mezőt illetően fontos a megfelelő formázás, viszont egy mint egy segédlet (hint) megjelenik amint a felhasználó elkezd gépelnia az adott mezőben.  
 Ezt követően a Leaflet által biztosított térképen a felhasználó bekell jelölje azt a pontot, amely az esemény helyszíne. A térkép használata a Vue keretrendszerében roppant egyszerű használni pontosan olyan, mint egy Vuetify által generált komponens használata. A komponensnek különböző attribútumai vannak. Az ‘url’, itt kell megadni azt a kinézetet amelyet szeretnénk használni a térképnek. A ‘zoom’ itt adhatjuk meg a térkép közelítését az adott pontra.

Illetve ‘center’, ahol beállíthatjuk az alapértelmezett pontot amelyre a térkép igazítva lesz (3.15. Ábra). Az ‘l-tile-layer’ tartalmazza azt az url-t amelyet egy ingyenes, vagy fizetős API-ból érhetünk el. Az ‘l-marker’ azt a pontot jelöli amelyet a felhasználó megjelölt, ezt két értéket tartalmaz. Hosszúságot és szélességet, amelyet az adatbázisban lementek, a késöbbiekben ez alapján tudja betölteni a létrehozott eseményeket az adatbázisból. Emellett különböző stílusszabványokat is meghadhatunk a térképnek, mint pl. a z-index, magasság vagy éppen a térkép sarkainak a lekerekítése, de ezek mellet teljesen személyre szabható a térkép kinézete.

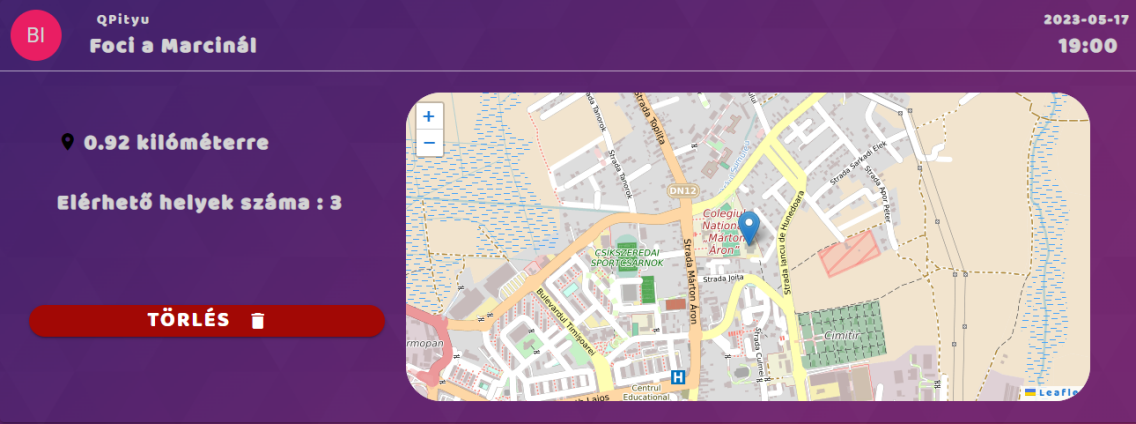


3.15. Ábra Térkép felépítése

Az adatok sikeres kitöltését követően a felhasználó a ‘Létrehozás’ gomb megnyomásával létrehozhatja az eseményt, amelynek sikerességéről egy felugró ablak jelez (3.10. Ábra).

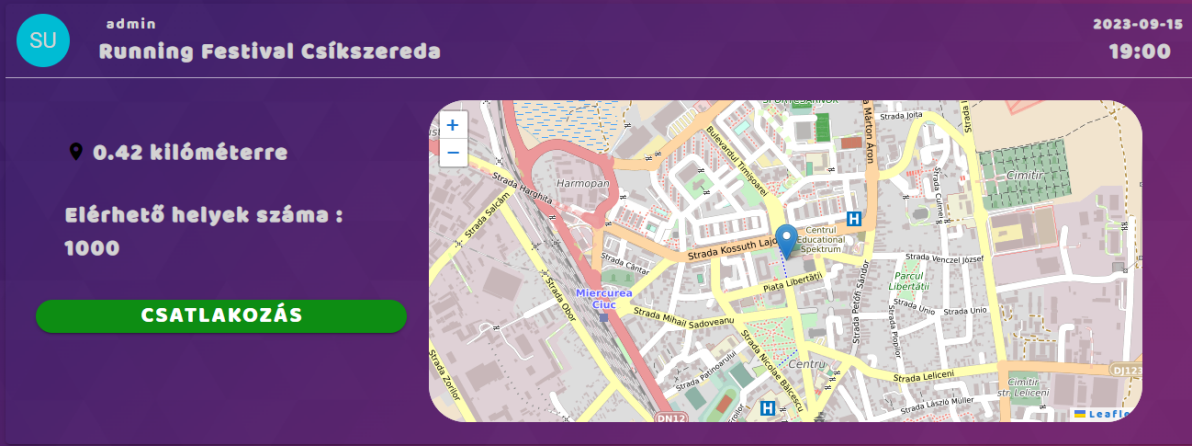
### **Események megtekintése**

Sikeres bejelentkezést követően a felhasználó a főoldalra irányul át, itt láthatóak az eddig létrehozott események, amelyek időrendi sorrendben vannak rendezve. Azok az események, amelyek a jelenlegi dátumhoz a legközelebb vannak, azok jelennek meg elsőként (3.16. Ábra). Azon események ‘kártyáján’ , amelyeket az éppen bejelentkezett felhasználó hozott létre egy piros ‘Törlés’ gomb helyezkedik el, a térkép bal oldalán. Itt a felhasználó törölni tudja az eseményt (3.16. Ábra). Biztonsági okokból egy megerősitő válasz szükséges a végleges törléshez amely egy felugró dialógus ablakban jelenik meg.

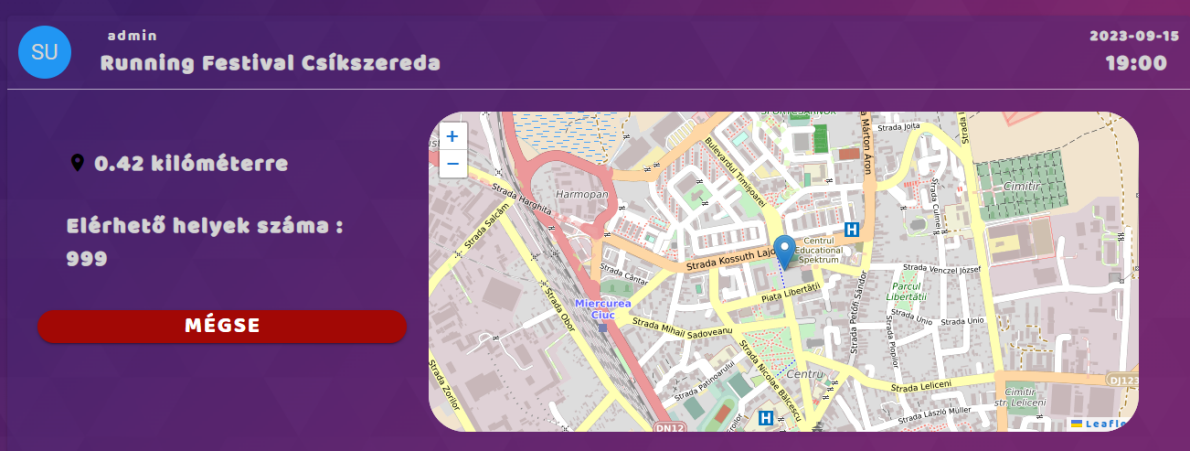


3.16. Ábra Saját eseménykártya

Azokhoz az eseményekhez, amelyeket más felhasználó hozott létre a zöld ‘Csatlakozás’ gombbal lehetséges (3.17. Ábra). Ekkor az elérhető helyek száma csökken és megjelenik egy piros ‘Mégse’ gomb, amellyel a felhasználó bármikor visszaléphet az adott eseményről (3.18. Ábra).

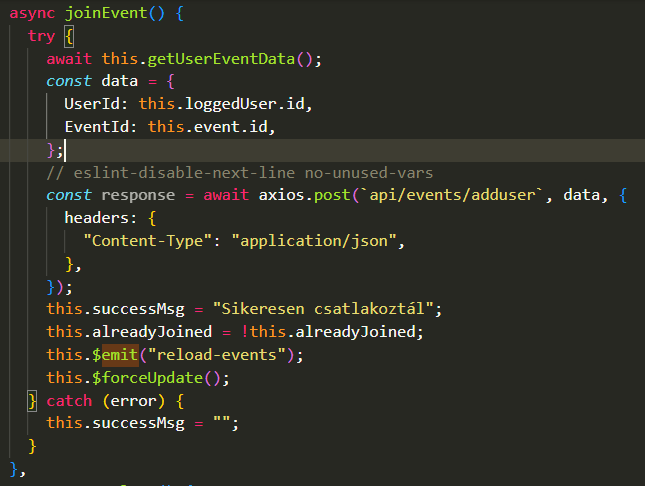


3.17. Ábra Eseménykártya



3.18. Ábra Eseménytől való visszalépés

Az eseményekhez csatlakozott felhasználók egy külön táblában vannak tárolva a ‘user\_events’ táblában. Ez a tábla egy összekötő táblaként funkcionál. Az adatok akkor kerülnek beszúrásra ebben a táblában amikor egy felhasználó csatlakozik egy eseményhez. Illetve a felhasználó visszalépésekor törlődik a táblából a megfelelő sor. A mellékelt ábrán (3.19. Ábra) látható egy ‘post’ kérés küldés amelyben a jelenleg bejelentkezett felhasználó, illetve az adott esemény azonosítója küldődik el.



3.19. Ábra Esemény és user azonosító küldése

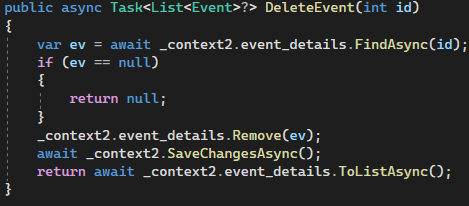
Annak érdekében, hogy az adatok automatikusan frissüljenek a Vue keretrendszer által biztosított metódust a ‘$emit’ -et használtam, amely a komponenes attríbútumaként minden adatmódosulás esetén meghívja a komponenst, ezáltal a felhasználónak nem szükséges frissítenie az oldalt.

Ha egy eseményt törlünk, ugyanez az ‘$emit’ metódus futtatásával van arra lehetőség, hogy a törölt esemény azonnal elérhetetlenné váljon a felhasználók számára (3.20. Ábra).



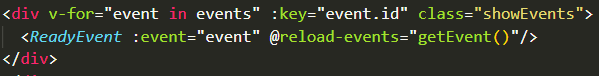
3.20. Ábra Emit funkció használata

Az ‘axios.delete’ használatával, az esemény azonosítóját elküldve a backend számára fog végrehajtódni a törlés, a backend az azonosító által fogja tudni, hogy melyik eseményt kell kitörölni az adatbázisból (3.21. Ábra). Első sorban leellenőrzi, hogy létezik-e az adott azonosítóju esemény, ezt követően törli azt, és a SaveChangesAsnc() menti a változásokat az adatbázisban.



3.21. Ábra Esemény törlése backenden

Annak eléréséhez, hogy minden esemény betöltődjön a főoldalon úgynevezett ‘props’ -okat használtam. A ‘props’ funkció lehetővé teszi az adatok átadását leszármazott komponensek számára. A szülő komponensben definiáljuk a ‘props’ -okat (3.22 Ábra) amelyeket ID alapján tudunk azonosítani. Ezt követően a gyerek komponensben a szükséges adatokat felhasználjuk. Amelyet az én esetemben az event.(kívánt adat) metódussal lehetséges (3.23. Ábra).



3.22. Ábra Szülő komponens



3.23. Ábra Gyerek komponens

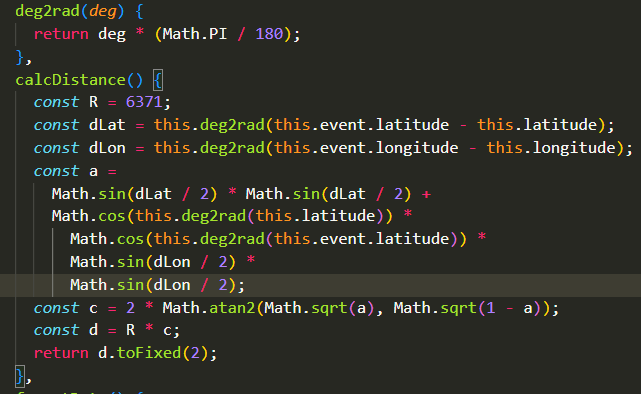
Természetesen az oldalon csak azok az események töltődnek be, amelyek megfelelnek a felhasználó által beállított kritériumoknak. Ezek a kritériumok lehetnek életkorra vonatkozóak, ha egy felhasználó 18 és 30 év közötti felhasználók eseményeit szeretné látni, akkor az oldal kiszűri azokat az eseményeket amelyek megfelelnek az adott kritériumnak (3.24. Ábra).

Továbbá a felhasználónak lehetősége van távolság szerinti szűrésre, itt a felhasználó a kívánt kilóméteres körzeten belüli események alapján tud szűrni. Ezáltal nem áll fent annak a veszélye, hogy olyan eseményeket találjon, amelyekre fizikailag nem áll lehetősége eljutni.

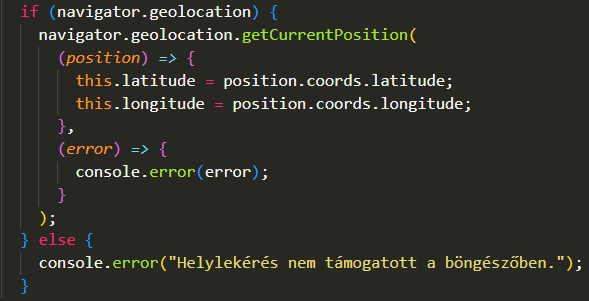


3.24. Ábra Életkor szerinti szűrés

A távolság kiszámítására, különböző matematikai képleteket alkalmaztam. Ezen képletek egy már más felhasználó által kidolgozott képletek. Ezek nélkül nem sikerült volna megoldanom a távolságszámítást (3.25. Ábra). A számításokhoz szükség volt az adott esemény hosszúság és szélességkoordinátáira, illetve a bejelentkezett felhasználó jelenlegi poziciójára amelyet a következő kód segítségével sikerült lekérnem (3.26. Ábra).

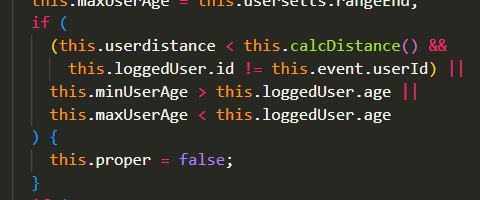


3.25. Ábra Távolság számítása



3.26. Ábra Felhasználó helyadatainak elérése

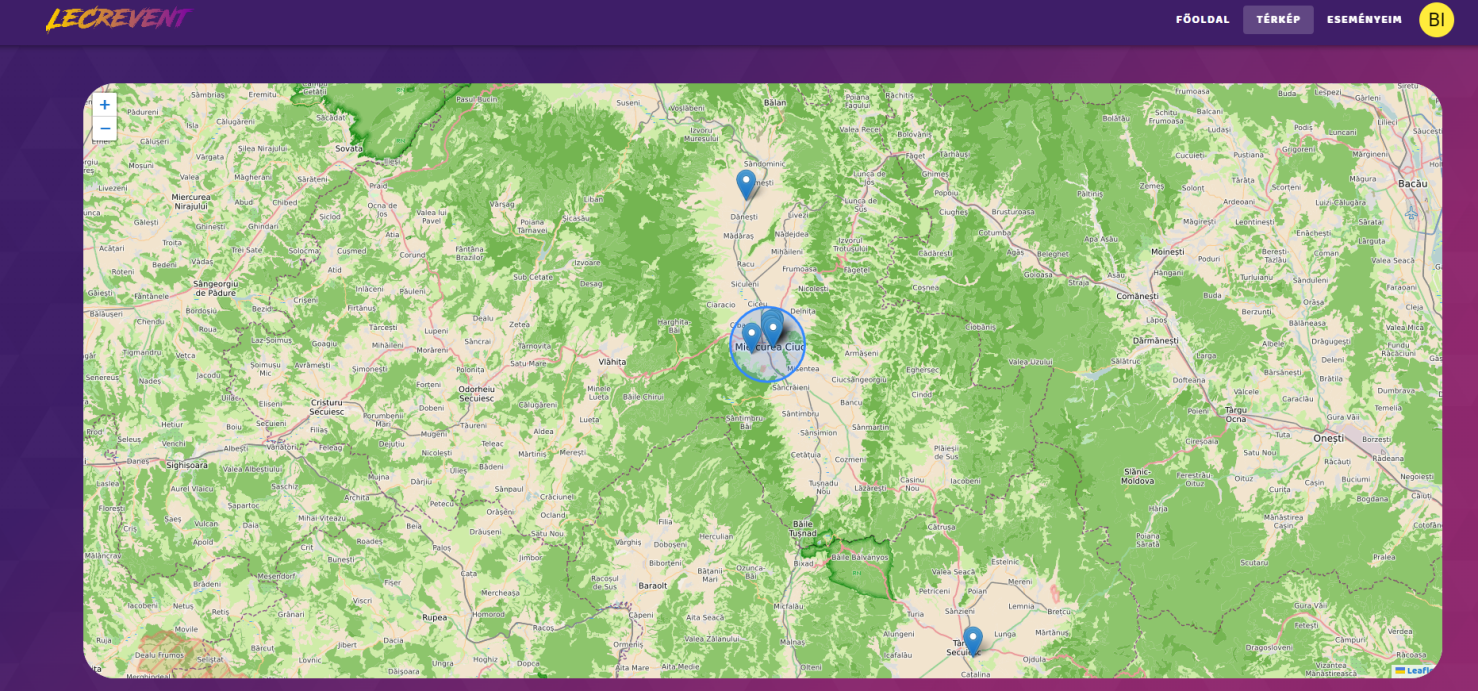
Ahhoz, hogy a felhasználó az oldal minden funkcióját teljes mértékben kitudja használni, engedélyeznie kell a helyadatok elérését. A felhasználó a beállításokat bármikor módosíthatja a ‘Profil’ fül alatt, így akár növelheti, vagy éppen csökkentheti a kívánt távolságot. Ha a kívánt távolságon belül van az adott esemény a felhasználó számára akkor betöltődik, ellenkező esetben nem lesz látható/elérhető (3.26. Ábra). Emellett fontos megjegyezni, azokat az eseményeket amelyet az éppen bejelentkezve levő felhasználó készített, a beállított távolság korlátozástól függetlenül betőltödik, így a felhasználó számára, az bármikor törölhető. Ha a feltételek teljesülnek ‘proper’ érték hamissá tételével érhető el az, hogy a felhasználó számára ne legyen látható az esemény.



3.26. Ábra Távolság ellenörzése

### **Teljes térkép**

A ‘Térkép’ menüpontra kattintva a felhasználó számára láthatóvá válik minden esemény amely a weboldal adatbázisában szerepel, a felhasználó beállításaitól függetlenül (3.27. Ábra).



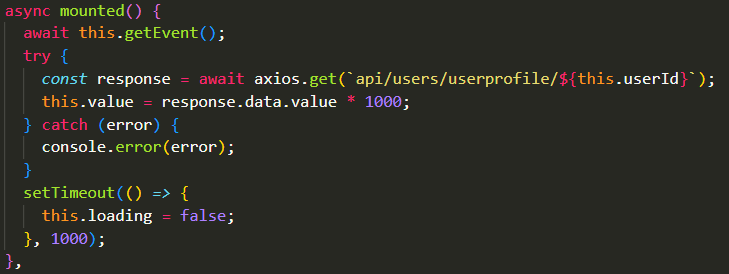
3.27. Ábra Nagy méretü térkép

Egy kék kör jelzi azt a kilóméteres körzetet amelyet a felhasználó beállított, ennek a körnek a középpontja a felhasználó jelenlegi földrajzi helyével azonos. Ezt a funkciót azért tartom fontosnak, mivel ezáltal a felhasználó kényelmi érzete növekedik. Olyan eseményekről szerez tudomást amely nem szerepel a zónájában, viszont nagyon szívesen részt venne rajta, így tud csatlakozni majd a kívánt eseményhez a megfelelő beállítások frissítésével. Egy adott eseménypontra kattintva a felhasználó részleteket tudhat meg az eseményről, amelyet a mellékelt ábra is mutat (3.28. Ábra). Ez azon eseményeknél játszik fontosabb szerepet, amelyek nincsenek benne a felhasználó zónájában.



3.28. Ábra Információ az eseményről

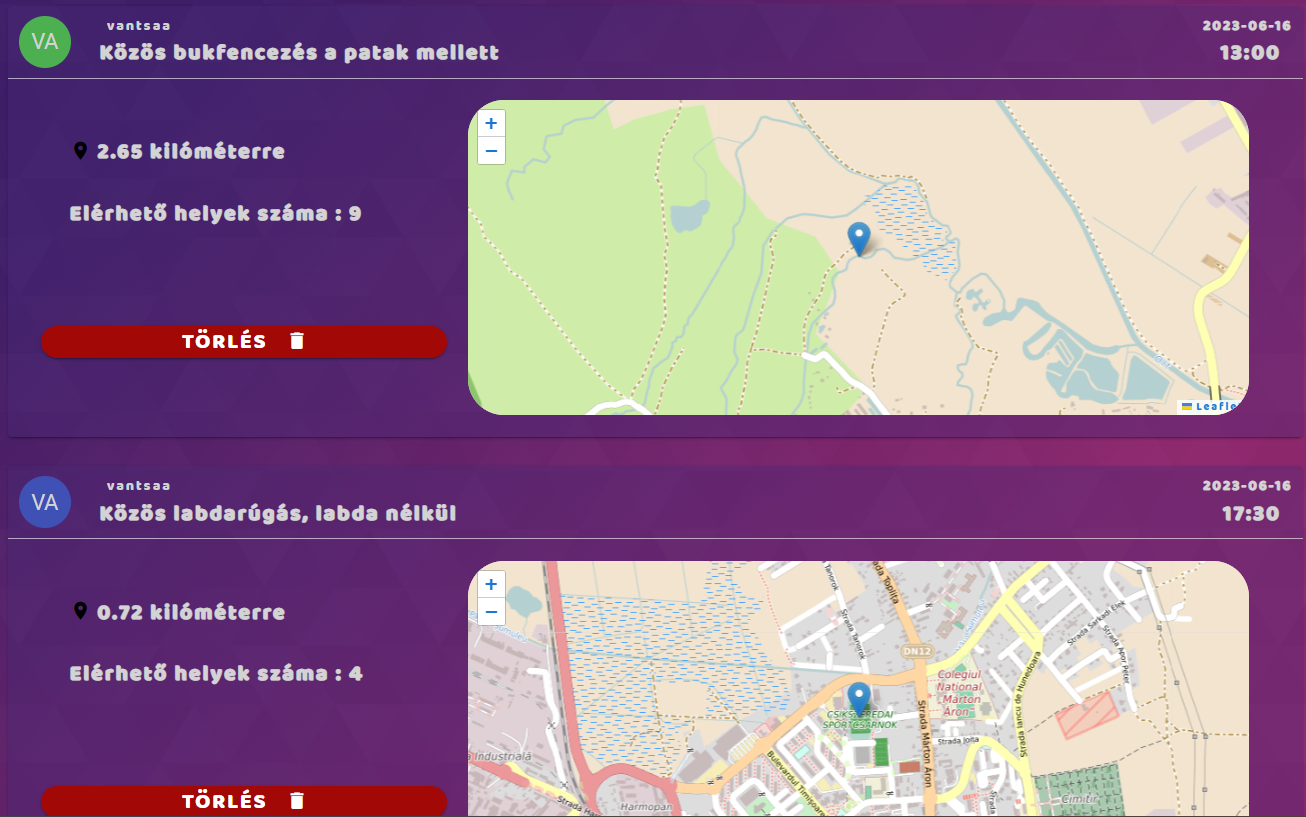
A felugró ablak a következő információkat tartalmazza: az esemény leírását, az esemény dátumát, és pontos időpontját. A felugró ablak egy ‘l-popup’ típusú komponens. A kék zónát amely a felhasználó zónáját jelképezi egy ‘l-circle’ komponens amely két attríbutummal rendelkezik, a felhasználó jelenlegi poziciójával amelyet a (3.26. Ábra) -án látható kód segítségével érhető el. Illetve tartalmazza a kör nagyságát. Ezt az adatbázisból lekért adatot megkell szorozni 1000-el mivel, a kör sugara méterben van meghatározva (3.29 Ábra).



3.29. Ábra Kör sugarának beállítása

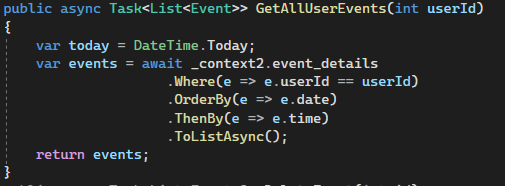
### **Saját események fül**

A felhasználónak az ‘Eseményeim’ fül alatt lehetősége van megtekinteie minden olyan eseményt amelyet ő hozott létre (3.30. Ábra). Ezzel időt spórolhat meg, és nem kell más felhasználó eseményei között keresgélnie, ha valamilyen részletet szeretne leellenőrzini a saját eseményével kapcsolatban, vagy esetlegesen törölni szeretné azt. A ‘Törlés’ gomb megnyomásával egy dialógus ablak jelenik meg amellyel a felhasználó eldöntheti biztos törölni szeretné-e az adott eseményt. Az eseményeken megjelenő távolság azt mutatja, hogy jelenleg milyen távol van az adott felhasználótól az esemény helyszíne, amely a (3.25. Ábra) -án is látható.



3.30. Ábra Saját események

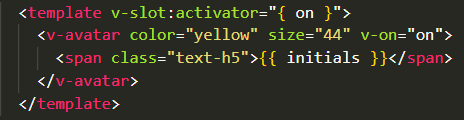
A saját események listázása úgy lehetséges, hogy egy olyan kérést küldök a backend számára, amely paraméterében szerepel a bejelentkezett felhasználó azonosítója, amely által kilistáz minden olyan eseményt amely az adott felhasználó nevéhez kötehető (3.31. Ábra)

.

3.30. Ábra Saját események listázása

### **Profil beállítása**

Ha a felhasználó a jobb felső sarokban található avatárra kattint, egy lenyíló menü fogadja, ahol a ‘Profil’ fül kiválasztásával lehetősége van módosítani a beállításait, vagy akár megnéznie a számlálót, amely azt mutatja, hogy eddig hány eseményt hozott létre az oldalon. Az avatár egy előre elkészített komponens, amelyet a Vuetify biztosít (3.31. Ábra). Az avatar tartalmazza a felhasználó vezetéknevének és keresztnevének első karakterét. A ‘Profil’ oldalon a Vuetify könyvtár által biztosított ‘v-card’ komponens ad keretet (3.32. Ábra) a benne megjelenő tartalomnak.



3.31. Ábra Saját események listázása



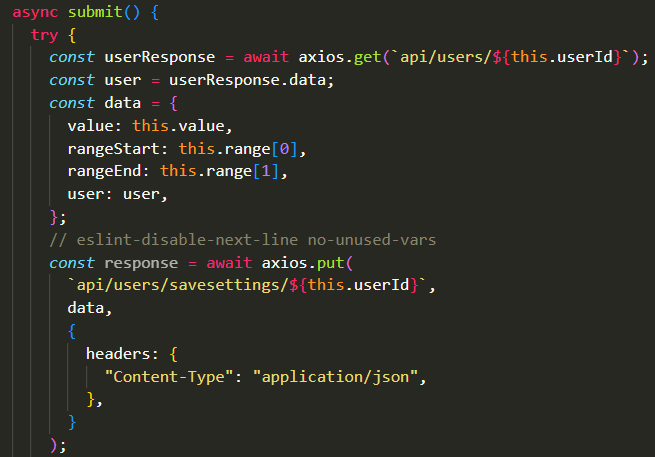
3.32. Ábra Profil kártya

A megjelenő két csúszka segítségével a felhasználó módosíthatja az adatait (3.33. Ábra). Az adatok módosítását követően a felhasználó véglegesítheti a kívánt változtatásokat. A ‘Mentés’ gomb lenyomását követően egy axios kérés történik (update), amely a cserélni kívánt adatok mentését eredményezi (3.34. Ábra). A művelet sikerességéről a már megszokott jobb felső sarokban megjelenő státuszüzenet fogja értesíteni a felhasználót.



3.33. Ábra Beállítások módosítása

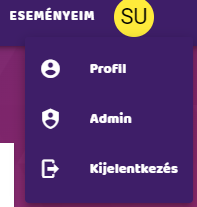
Az imént említéett axios kérés egy put utasítás, amely paramétereként tartalmazza a felhasználó azonosítóját, ezáltal tudja a rendszer, hogy pontosan melyik felhasználónak voltak frissítve a beállításai. Az itt elküldött szerepel a maximális távolság értéke, illetve az a korcsoport alsó és felső határa.

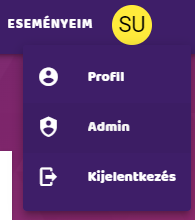
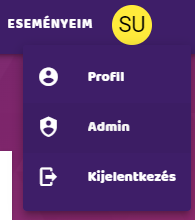


3.34. Ábra Put kérés küldése

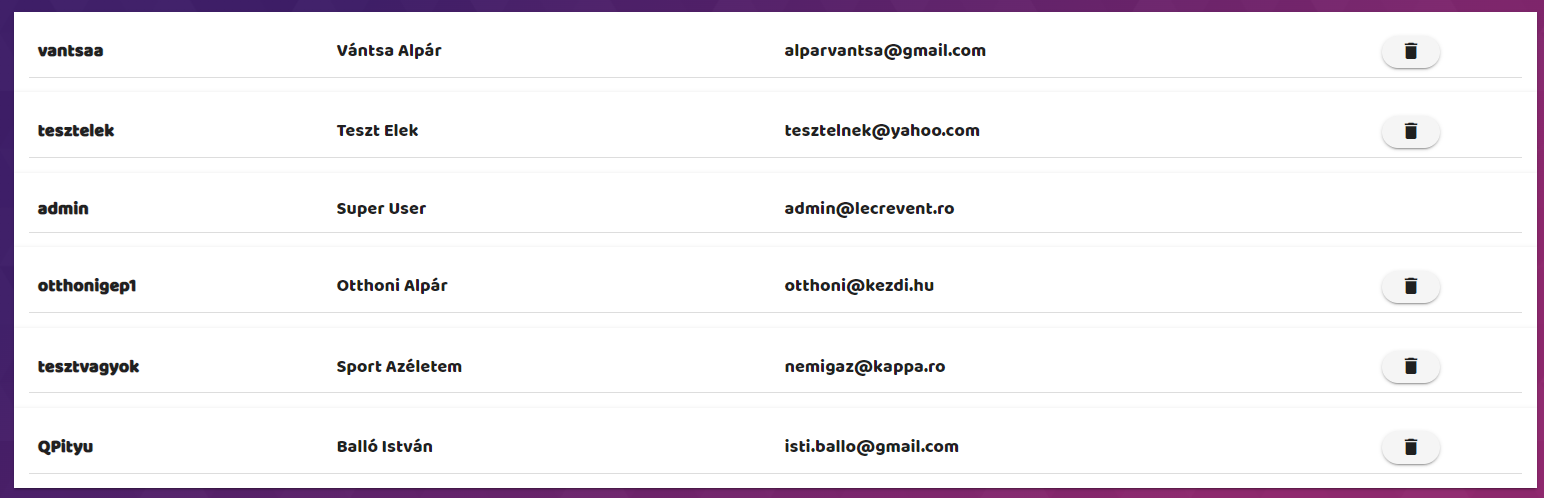
### **Admin felület**

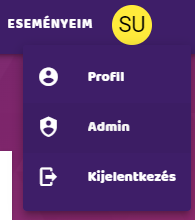
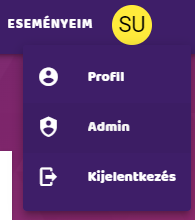
Az alkalmazás rendelkezik egy minimalista adminfelülettel. Az admin számára elérhető minden funkció. Lehetősége van törölni bármely eseményt, bármely beállítás nélkül képes látni minden egyes eseményt és törölni azokat. Emellett az avatar ikonra kattintva (3.35. Ábra) lehetősge van megtekinteni az összes regisztrált felhasználót, illetve ezen felhasználók adatait. Ezen a fülön tudja törölni a felhasználókat, amelyek valamilyen szabálysértést követően nem használhatják a továbbiakban a weboldalt.



3.35. Ábra Admin felület elérése

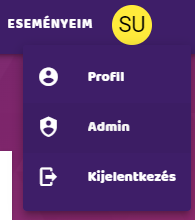
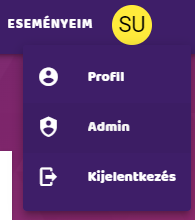
Egy felhasználó törlése, maga után vonja azt, hogy minden olyan esemény is automatikusan törlődik, amelyet az adott felhasználó hozott létre, függetlenül attól, hogy az eseményhez már volt-e előzőleg csatlakozva más felhasználó. Ezáltal az admin csak jól átgondolt és indokolt esetben törölhet egy felhasználót. Az admin felület egyszerűnek mondható, viszont úgy gondolom rendelkezik minden olyan funkcióval, amellyel egy admin felületnek rendelkeznie kell (3.36. Ábra). A alkalmazást illetően, onnan fogja tudni, hogy éppen admin van bejelentkezve, vagy sem, hogy a böngészőben elmentett JWT token segítségével lekéri a felhasználó adatait, ha ezek megegyeznek egy admin jogosultsággal rendelkező felhasználóéval, akkor máris elérhetővé/láthatóvá válnak ezen funkciók az alkalmazásban bejelentkezve levő fél részére.



3.36. Ábra Admin felület

A következő kódot használva, kerül törlésre egy felhasználó, amely a felhasználók melleti törlés (kuka ikon) megnyomásával lehetséges (3.37. Ábra). Az itt törölt felhasználó által létrehozott események trigger segítségével törlődnek, amint a felhasználó eltávolításra kerül az adott táblából (user\_profiles) azt követően aktíválódik a trigger, amely töröl minden olyan sort a az event\_details táblából amelynek a felhasználó azonosítója megegyezik a törölt felhasználó azonosítójával.



3.37. Ábra Felhasználó törlése

# **Következtetések**

Fusce a metus eu diam varius congue nec nec sapien. Vestibulum orci tortor, sollicitudin ac euismod non, placerat ac augue. Nunc convallis accumsan justo. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas.

Donec malesuada vehicula lectus, viverra sodales ipsum gravida nec. Integer gravida nisi ut magna mollis molestie. Nullam pharetra accumsan sagittis. Proin tristique rhoncus orci, eget vulputate nisi sollicitudin et. Quisque lacus augue, mollis non mollis et, ullamcorper in purus. Morbi et sem orci.

Praesent accumsan odio in ante ullamcorper id pellentesque mauris rhoncus. Duis vitae neque dolor. Duis sed purus at eros bibendum cursus nec a nulla. Donec turpis quam, ultricies id pretium sit amet, gravida eget leo.

# **Irodalomjegyzék**

1. Az ASP.NET Core dokumentációjahttps://learn.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-7.0
2. Ekler Péter, [Fehér Marcell](https://www.libri.hu/szerzok/feher_marcell.html), [Forstner Bertalan](https://www.libri.hu/szerzok/forstner_bertalan.html), [Kelényi Imre](https://www.libri.hu/szerzok/kelenyi_imre.html). (2012): *Android-alapú szoftverfejlesztés*, Szak Kiadó Kft., Bicske.
3. Ferrucci, F., Salza, P., Sarro, F. (2018): *[Using Hadoop MapReduce for Parallel Genetic Algorithms: A Comparison of the Global, Grid and Island Models](https://www.mitpressjournals.org/doi/full/10.1162/evco_a_00213),* Evolutionary Computation, Vol. 26, pp. 535–567.
4. Nixon, R. (2018): *Learning PHP, MySQL & JavaScript: With jQuery, CSS & HTML5*, [O'Reilly](http://csikszereda.qulto.ro/ro/results/-/results/publisher/O/solr?p_auth=Fm0hnmWe) Media, USA.
5. Sklar, D. and Trachtenberg, A. (2014): *PHP cookbook*, [O'Reilly](http://csikszereda.qulto.ro/ro/results/-/results/publisher/O/solr?p_auth=rgBPwg8X) Media, USA.
6. Stauffer, M. (2017): *Laravel Up and Running*, O’Reilly Media, USA.
7. Ullman, J.D., <http://infolab.stanford.edu/~ullman/focs/ch07.pdf>, 2019.03.15.
8. https://kinsta.com/knowledgebase/what-is-github/