**SZEGEDI SZAKKÉPZÉSI CENTRUM**

**VASVÁRI PÁL GAZDASÁGI ÉS INFORMATIKAI TECHNIKUM**

**SZOFTVERFEJLESZTŐ– ÉS TESZTELŐ**

**SZAKKÉPZÉS**

**„VIZSGAREMEK”**

**EventExplorer**

**Készítette:**

**Juhász Adél**

**Rácz Nikoletta**

**SZEGED**

**2022.**

**BEVEZETÉS**

A vizsgaremek témájának alapjául az az elgondolás vezérelte a szoftverfejlesztés menetét, hogy egy olyan rendszert készítsünk el, amely a felhasználók számára nyújt kényelmes szolgáltatást a számtalan fellelhető rendezvények felkutatása és rá való jelentkezése céljából.

Ehhez az alábbi komponensek jelentik az alapot egy modern szoftverrendszerhez:

* ***Adatbázis***: adatok perzisztens tárolása
* **REST API (backend):** egységes, szabványos alkalmazás, amely képes kiszolgálni bármennyi klienst
* **Asztali alkalmazás (front-end):** Konfigurációs és adminisztrációs felület a rendszer karbantartói és üzemeltetői számára – pl. felhasználókezelés
* **Webalkalmazás (front-end):** Számítógépen, internetes böngésző kliensek számára elérhető, felhasználói oldali alkalmazás – pl. felhasználó rendezvényeket tud kilistázni, jelentkezni rá, stb.
* **Mobilalkalmazás (front-end):** Mobil eszközök kliensei számára elérhető, felhasználói oldali alkalmazás – lényegében ugyanazzal a funkcióval bír, mint a webalkalmazás, de a modern technika világában kényelmes, hordozható mobileszközön is elérhető gazdag felhasználói környezet

Ezen rendszer megvalósításának folyamatát, illetve fejlesztői és felhasználói kézikönyvét tartalmazza ez a dokumentáció.

**ADATBÁZIS**

**Fejlesztő: Juhász Adél**

**Hardverigény:**

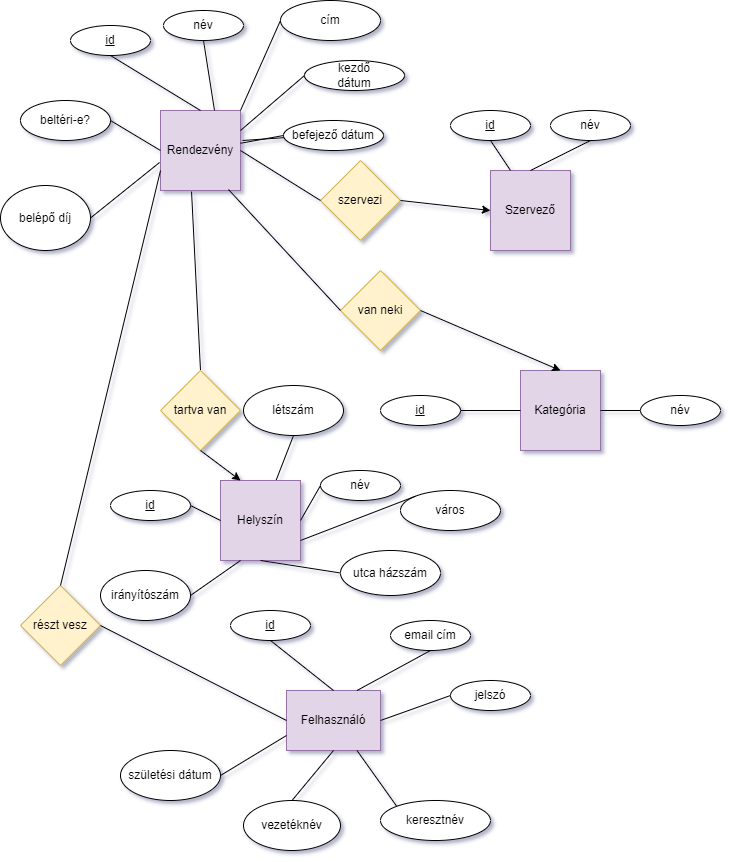
* Operációs rendszer: A backend által felépített séma platform független, akár Windows vagy Linux alapú operációs rendszereken is kiépíthető az adatbázis séma az adatokkal
* Szoftverigény: MySQL adatbázis kezelő legújabb verzió
* Processzor igény: Intel/AMD
* Memóriaigény: 4 GB

Az EventExplorer alapját az adatbázis adja. A korábbiakban említettük, hogy perzisztensen, azaz tartósan itt tárolódik minden elengedhetetlen adat az alkalmazás működtetéséhez, amely az alkalmazások számára nélkülözhetetlenek (pl. felhasználó fiókjához tartozó e-mail cím és jelszó páros, stb.), valamint ezt a komponenst használja a REST API backend, hogy az egységes felületet biztosítson a kliens-oldali alkalmazások számára adatelérés céljából. Az adatbázist közvetlenül ez a backend alkalmazás használja közvetlenül, bizonyos adatait manipulálhatja is, de magát az adatbázis sémát is ő építi fel MySQL technológiát alkalmazva.

Az adatbázis a REST API projekten belül készült el *MySql.EntityFrameworkCore* könyvtár segítségével, migrációs szkriptekkel. Az adatok manipulációját kifejtő backend végpontokat a REST API-t taglaló fejezetek részletezi. A backend projektben C# osztályok reprezentálják az adatbázisban szereplő táblákat, illetve azon tulajdonságaik (adattagjaik, property-jeik) pedig az oszlopokat, amelyekből MySQL specifikus migrációs szkriptet készít C# nyelven az EntityFrameworkCore MySQL specifikus keretrendszere – továbbá migrációs szkript fog teszt adatokat is elhelyezni a fizikai adatbázisban. Ezeket szintén a backend projekt forrásai között lehet megtalálni. A migrációs szkriptek végrehajtása során a backend projektben megadott adatbázis kapcsolatleíró karakterlánca (connection string) segítségével a kijelölt számítógépre egy parancs segítségével felépíthető az adatbázis séma a teszt adatokkal.

Előzetesen a hagyományos egyed-kapcsolat diagram modellező technológia segítségével is megterveztük az adatbázist: az adatbázis táblák kivétel nélkül magyar nyelvű karakterkódolást illetve harmadik normálformáig normalizált táblákat, a relációs adatbázisra leginkább jellemző egy-több kapcsolatokkal, minél kevesebb redundancia megvalósítása idegenkulcs megkötésekkel az anomáliák és inkonzisztens adatok elkerülése céljából.

Ahogyan a mellékelt egyed-kapcsolat diagram is reprezentálja, a legtöbb tábla kapcsolatban áll egymással. Az ***events*** tábla a központi tábla, amely tárol mindenfajta információt a megrendezésre kerülő eseményről, ez több-több kapcsolatban áll a felhasználók személyes adatait tároló táblával – egy felhasználó több rendezvényre is jelentkezhet, egy rendezvényre több különböző felhasználó is jelentkezhet. Egy-több kapcsolat áll fent az eseményeket tároló táblával, úgy, mint azt egy szervező szervezheti, de egy szervező több rendezvényt is meghirdethet; egy rendezvény egy helyen van tartva egyszerre, de különböző időben több rendezvény is tartható egy adott helyen; illetve a kategóriákat tároló táblával is egy-több kapcsolat a mérvadó: egy rendezvénynek egy kategóriája lehet, de egy kategóriába több különböző rendezvény is besorolható.

****

**REST API (BACKEND)**

**Fejlesztő: Juhász Adél**

**Hardverigény:**

* Operációs rendszer: A backend a .NET5 keretrendszer miatt platform független, akár Windows vagy Linux alapú operációs rendszereken is kiépíthető az adatbázis séma az adatokkal
* Szoftverigény: .NET5 SDK és Runtime
* Processzor igény: Intel/AMD
* Memóriaigény: 4 GB

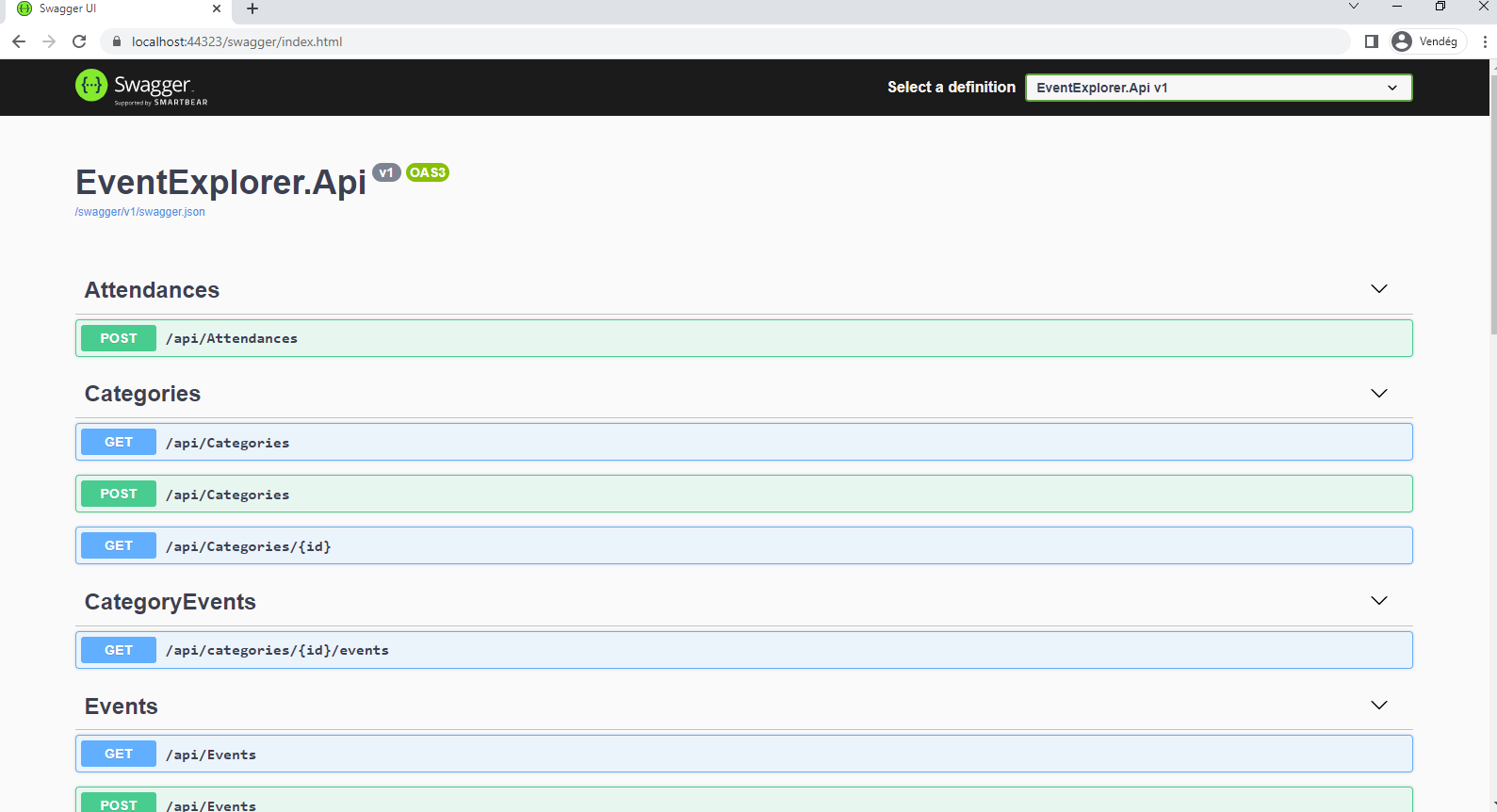
Az EventExplorer backendje, amely közvetlen kapcsolatot teremt a rendszer és az adatbázis között. Technológiáját tekintve C# programozási nyelv 7-es verziójában készült, .NET5 runtime alatt, illetve ***Kestrel*** development szerveren közvetlenül futtatható alkalmazás (a platformfüggetlenség miatt még csak a Windows platform függő ***IIS Express*** development szerver telepítése sem szükséges).

Az alkalmazás architektúráját tekintve hagyományos többrétegű, MVC szerinti felbontásban készült, illetve a REST alapú architektúrát valósítja meg (http igék használata, pl. POST, GET http metódusok megfelelő http státuszkód visszaküldésével a kliens számára). A legalsóbb réteg, a Model réteg tárolja az alkalmazásban szereplő adatmodellek definícióját (ezek által készül el a fizikai adatbázisséma is teszt adatokkal EntityFrameworkCore ORM-könyvtár által biztosított migrációs szkriptek segítségével). A felette elhelyezkedő Persistence réteg implementációja az adatkezelési rétegnek: ez a réteg tartalmazza a modelleket MySQL szabvány szerint leíró konfigurációs osztályokat (pl. objektum property-k nevének közvetlen megadása MySQL specifikus nevezéktan szerint – kisbetűs oszlopnevek). Erre azért van szükség, mivel az EntityFrameworkCore convention-over-configuration elven működik, azaz a nagybetűs C# property-k alapján az ORM nem kisbetűs oszlopneveket készítene a fizikai adatbázisban; valamint lehetőség van elnevezni a táblákat, pl. többes szám alak használata. Az EntityFrameworkCore specifikus ApplicationDbContext osztályt is ez a réteg tartalmazza, amely a tényleges, fizikai kapcsolatot valósítja meg a backend és az adatbázis között EntityFrameworkCore specifikus C# típusokkal, pl. egy DbSet típus egy tábla absztrakciója a .NET platform szerint. Tovább a Repository programtervezési minta megvalósítása is itt helyezkedik el, amely az ApplicationDbContext és azon funkcionalitásának elrejtéséért felel (ha legközelebb az ORM cseréjére lenne szükség valamely más könyvtár által EntityFrameworkCore helyett, akkor a többi réteg nem fogja érzékelni a cserét, mivel csak a repository-kban lévő adatkezelő funkcionalitást kell csak újra implementálni ezekben a komponensekben az új választott könyvtár logikája szerint).

A Service réteg tárolja az alkalmazásban előforduló valamennyi üzleti logikát, megkötést modell osztályok szerint csoportosítva; valamint a Controller osztályok az alkalmazás végpontjait definiálja a modell osztályok szerint csoportosítva – pl. felhasználó adatait lekérdező illetve újakat létrehozó végpontok ezért a felelősek a UsersControllerben. Közvetlen ezekkel a végpontokkal létesíthető kapcsolat http protokoll segítségével bármely http klienssel (pl. Postman, Insomnia, webalkalmazás vagy kényelmi funkciók eléréért a Swagger elnevezésű OpenAPI specifikációt implementáló eszköz által nyújtott kényelmes, grafikus felület a Swagger UI komponense, Ez az eszköz a .NET specifikus ApiController annotációkból generálja le a teljes backend végpontok specifikációit. Ez a komponens egy sima böngészőben megnyitható alkalmazás, amely a lokális számítógépen fut a projekt elindítása után.

Nagyon fontos részét képezi még az alkalmazásban helyet kapó resource funkcionalitást ellátó C# osztályok, amelyeket a controllerek válasz formájában küldenek vissza vagy fogadnak egy GET vagy POST folyamat elindítása során JSON formára szerializálva a front-end kliensnek vagy klienstől – mivel az adatmodelleket nem küldheti, és nem fogadhatja a backend határát jelző controller réteg. Ennek az az egyszerű oka, hogy POST küldésnél nem kell egyedi azonosítókat megadni, ha egy új modell-t (pl. user-t készítünk), mivel a soron egyedi azonosítót nem tudhatja a felhasználó, mivel nincs rálátása az adatbázisra: ennélfogva egy azonosító mező nélküli request resource-t vár el a megfelelő controller megfelelő végpontja. Szintén nem küldhető vissza a teljes adatmodell a GET kérésre a végponton, mivel több helyen az egy-több kapcsolat miatt listákban vannak tárolva összekapcsolódó objektumok, amelyet nem lehet végtelen ciklus nélkül JSON formába parse-olni, illetve nem is minden mezőt kell egy kérésre visszaadni (pl. user esetén jelszót nem küldünk a kliensnek). Ezeken a helyeken response resource objektumok lesznek a segítségünkre. A request és a resource-okra történő map-elést az AutoMapper nevű könyvtár konvenció szerint automatikusan végzi egy mapping profile konfigurációs komponens alapján.

Mivel a backenddel az EventExplorer rendszer felhasználója nem létesít közvetlen, csak közvetett kapcsolatot (webalkalmazás vagy mobil alkalmazás által), így ehhez a fejezethez nem tartozik felhasználói dokumentáció.



**KÉSŐBBI FEJLESZTÉSEK A REST API-HOZ:**

* Regisztráció és bejelentkezés végpontok tényleges megvalósítása
* API továbbfejlesztése .NET6 runtime-hoz (itt már sokkal kevesebb boilerplate code kell egy API megírásához)
* Eseményről való leiratkozás megvalósítása felhasználó szemszögéből (DELETE végpontok)
* Hónapok, városok, egyéb dátumok szerinti szűrés megvalósítása
* Pagination megvalósítása a repository-kban
* Felhasználói adatmódosítás (PUT végpontok)

**VUE.JS ALAPÚ WEBALKALMAZÁS**

**Fejlesztő: Juhász Adél**

**FEJLESZTŐI DOKUMENTÁCIÓ:**

**Hardverigény:**

* Operációs rendszer: A front-end a Node.js platform miatt platform független
* Szoftverigény: Node.js 16.15.0 (long-term support) vagy Node legutóbbi verziója szerinti Node Package Managerrel
* Processzor igény: Intel/AMD
* Memóriaigény: 4 GB

A webalkalmazás az EventExplorer-t használó felhasználók számára nyújtott applikáció, amely a rendezvénykeresés és jelentkezés kényelmét szolgálja. Az alkalmazás Vue.js JavaScript keretrendszer segítségével és előnyeivel készült fejlesztésre az ECMAScript 6 szabvány szerinti „modern” JavaScript programozási és HTML leíró nyelveken. A stílus gerincét leginkább a Bootstrap 4 CSS komponens könyvtár adja, saját CSS kód nem került implementálásra. Alapvetően az SPA, azaz single-page application vezérelte a webalkalmazás kialakítását, tehát mindig egy oldalon fut az egész alkalmazás, csak a komponensek cserélődnek, ezáltal javítva az azonnali HTML renderelést a felhasználó számára (nincsenek lassan betöltődő oldalak). Alapvetően két komponens jelenik meg különböző útvonalakon: a fő oldal valamint a saját eseményeket kilistázó oldal, amelyekre a felhasználó jelentkezett.

Mivel nem készült a regisztráció és a belépés funkciók, így felhasználót váltani a config mappában lévő user.config.json fájlban lehet, ahol az adott felhasználó azonosítóját kell beírni. A backend felé ezután a felhasználói azonosítóval történik minden interakció: rendezvényre jelentkezés vagy a jelentkezett rendezvények listázása. A gyökér könyvtárban lévő main.js amely az egész Vue applikációt inicializálja, az index.js tartalmazza a fő oldal és az események komponensek URL útvonalát, az App.vue pedig a navigációs sávot tartalmazza, valamint ezen single page-re történik a másik két fő komponens kirendelése, amely megvalósítja az SPA modellt.

A ***util*** mappában egyéb, utility függvényeket tartalmazó JavaScript modulok, függvények találhatók, úgy, mint dátum formázó util függvény, illetve egy olyan metódus, amely ellenőrzi, hogy a konfigurációs fájl alapján van-e felhasználó bejelentkezve az alkalmazásba vagy sem. Ennek függvényében rendereli a navigációs sávot az App.vue: ha van felhasználó, akkor a felhasználók számára elérhető menüket listázza, illetve a bejelentkezett felhasználó nevét is jelzi. Ha nincs bejelentkezett felhasználó, akkor nem jelenít meg olyan funkciókat a vendég felhasználónak, amelyeket csak regisztrált felhasználók alkalmazhatnak (pl. rendezvényre jelentkezés komponens útvonala).

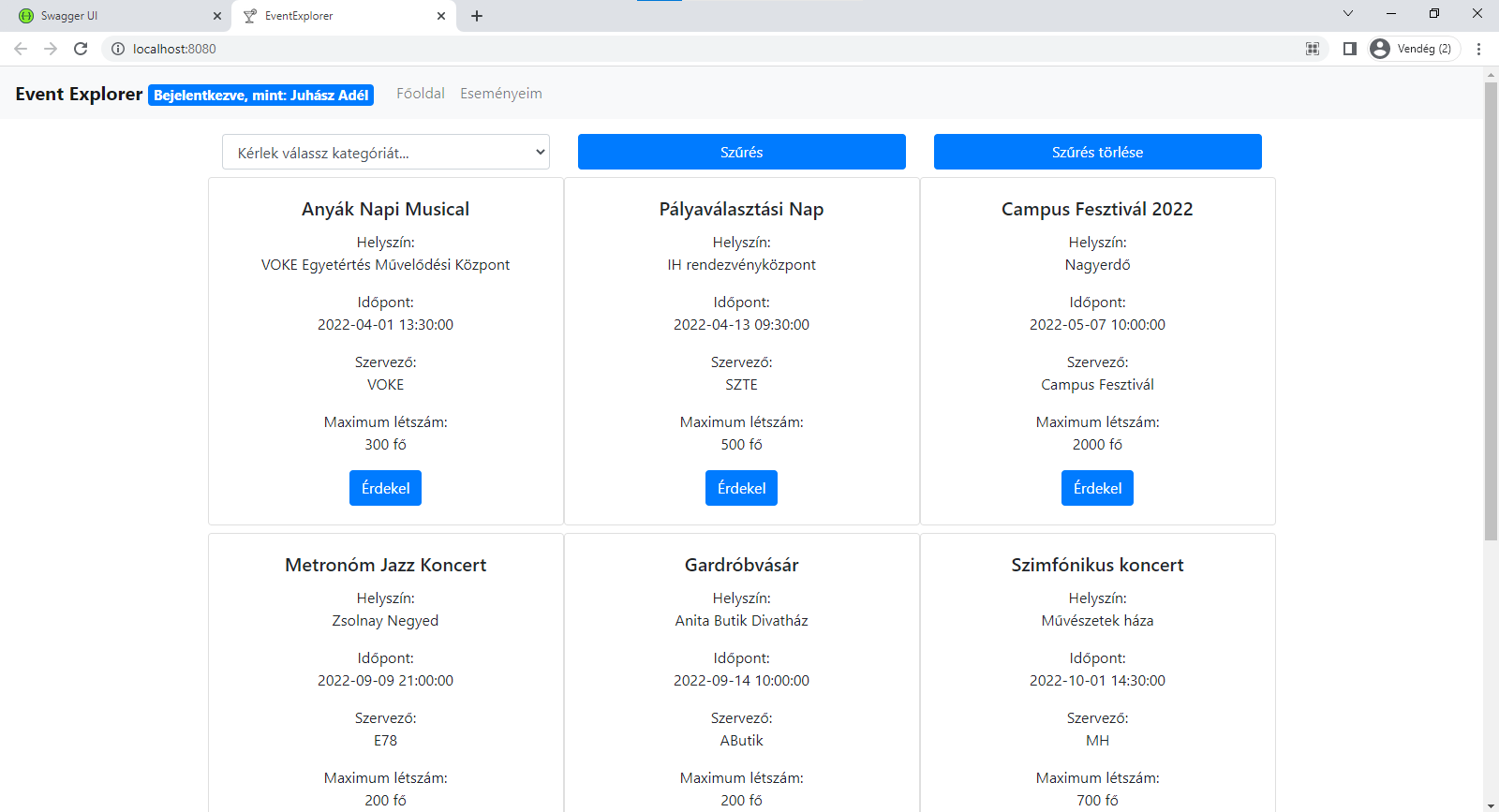
A ***services*** mappa tartalmaz az adatbázisban tárolt adatmodellek szerint felosztott függvényeket, amely ***axios*** könyvtár segítségével http kéréséket végez el a backend végpontjai felé (pl. jelentkezés eseményre), illetve adat lekérdezés után a backend-től visszakapott JSON objektumokat ad vissza a hívó grafikus komponensek számára, amelyeket kirenderelnek HTML-be ágyazva.

**FELHASZNÁLÓI DOKUMENTÁCIÓ:**

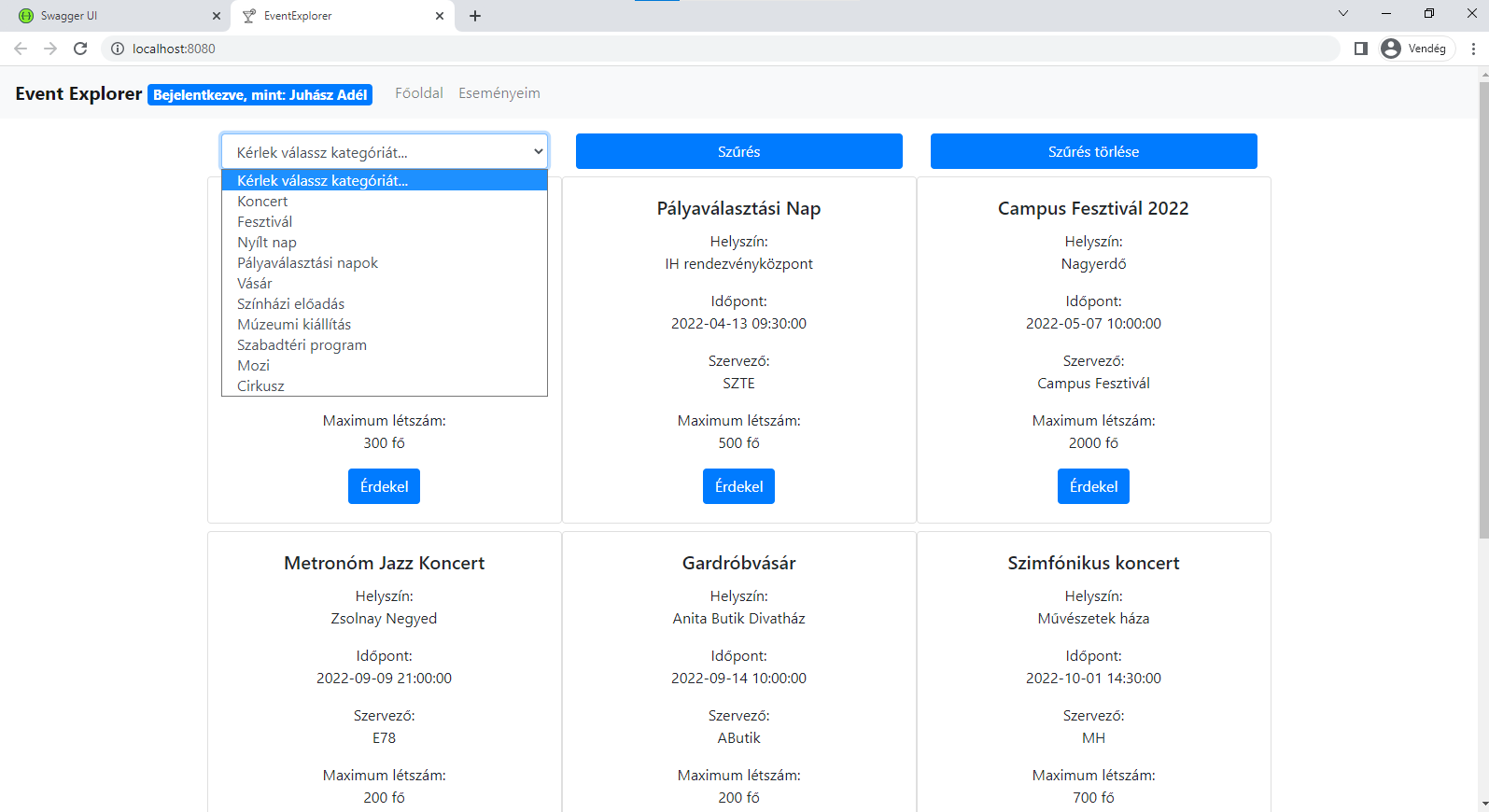
**Hardverigény:**

* Operációs rendszer: A front-end webes applikáció lévén platform független, számítógép vagy mobileszköz böngészőjében elérhető
* Szoftverigény: Szabadon választott webböngésző
* Processzor igény: Intel/AMD
* Memóriaigény: 2 GB

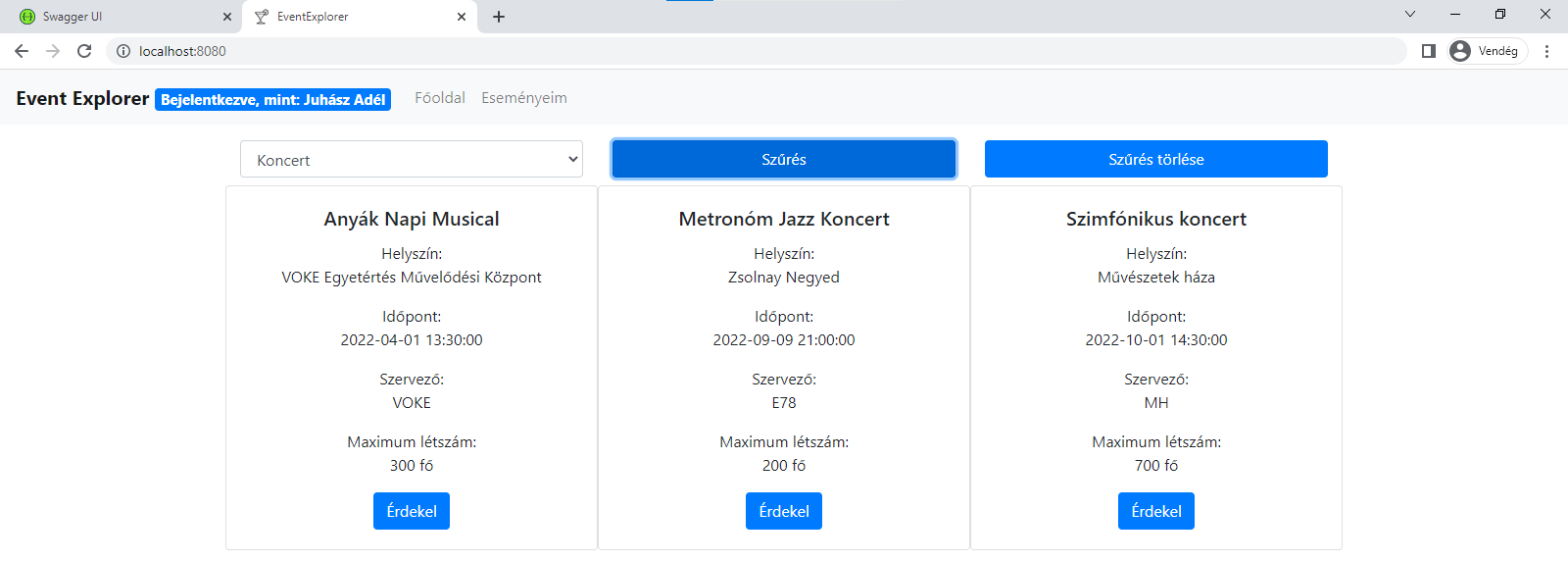
A bejelentkezés után a felhasználó a főoldalt láthatja, ahol minden féle kategóriájú esemény közül válogathat. A navigációs sávon láthatja a saját nevét, amelyet regisztrációkor megadott. Itt láthatóak az események adatai, pl. név, helyszín, az esemény szervezőjének neve, időtartam valamint a maximum létszám. Amennyiben egy esemény után érdeklődik, az „Érdekel” gombra kattintva jelezheti részvételét.



Lehetőség van arra is, hogy kategória szerint válogasson eseményt, amit a kategória szűrővel tehet meg.



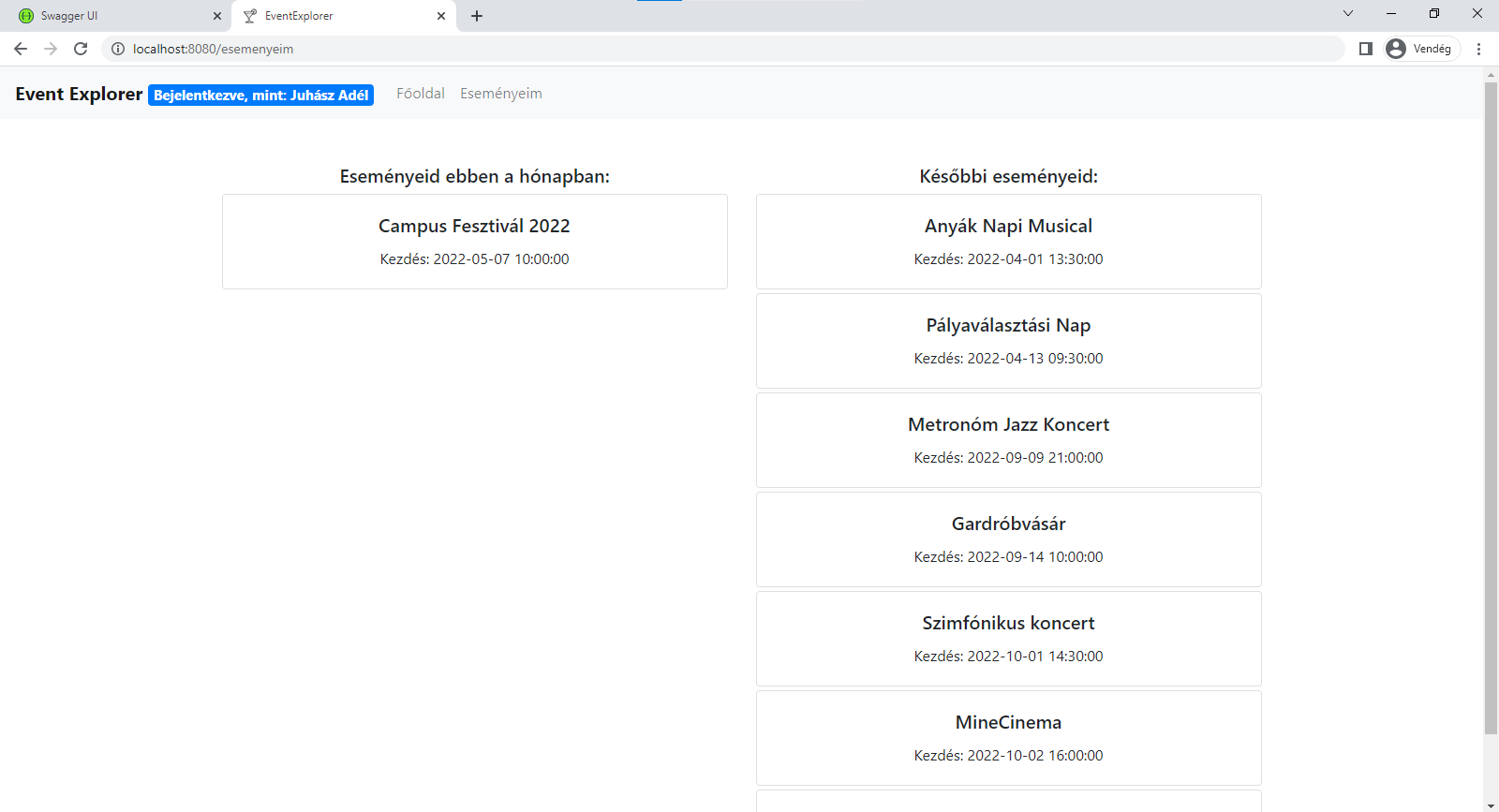
Amennyiben koncert kategória kerül kijelölésre, úgy a koncertek jelennek meg a „Szűrés” gombra kattintva. Amennyiben ismét láthatóvá szeretné tenni az összes meghirdetett eseményt, úgy a „Szűrés törlése” gombra kattintva tehető meg.



Amennyiben olyan eseményen kattintja be az „Érdekel” gombot, amelyen már részt vesz, úgy hibaüzenet jelenik meg az esemény kártyában:



A navigációs sávon az „Eseményeim” menüpontra kattintva érthetők el az adott események, amelyeken részvételét fejezte ki. Az oldal két szekcióban mutatva jeleníti meg az Ön eseményeit: a közelgő események hasábban az adott hónapban megrendezésre kerülő események vannak kilistázva, a második hasábban pedig a nem az adott hónapban megrendezésre kerülő, későbbi események láthatók.



**KÉSŐBBI FEJLESZTÉSEK A VUE.JS WEBALKALMAZÁSHOZ:**

* Regisztráció és bejelentkezés funkciók tényleges megvalósítása
* Más, modernebb stíluslap használata Bootstrap helyett
* Kisebb és újra felhasználhatóbb Vue komponensek implementálása
* Oldalak még kisebb komponensekbe történő szétszedése
* Eseményről való leiratkozás megvalósítása
* Hónapok, városok, egyéb dátumok szerinti szűrés
* Pagination (oldalakra tördelés, ne egy teljes oldalon jelenjen meg minden esemény, hanem oldalszámokkal léptetve)