



győri szakképzési centrum

Jedlik Ányos  
Gépipari és Informatikai  
Technikum és Kollégium



9021 Győr, Szent István út 7.

+36 (96) 529-480

+36 (96) 529-448

OM: 203037/003

jedlik@jedlik.eu

www.jedlik.eu

## Záródolgozat feladatkiírás

Tanulók neve: Magashegyi Ádám, Vadkerti Sára  
Képzés: nappali munkarend  
Szak: 5 0613 12 03 Szoftverfejlesztő és tesztelő technikus

### A záródolgozat címe:

## Move Your Body

Konzulens: Bognár Pál  
Beadási határidő: 2022. 04. 15.

Győr, 2021. 10. 01

---

**Módos Gábor**  
igazgató

## Konzultációs lap

	A konzultáció		Konzulens aláírása
	ideje	témája	
1.	2022.10.01.	Témaválasztás és specifikáció	
2.	2022.02.01.	Záródolgozat készültségi fokának értékelése	
3.	2022.03.17.	Dokumentáció véglegesítése	

## Tulajdonosi nyilatkozat

Ez a dolgozat a saját munkánk eredménye. Dolgozatunk azon részeit, melyeket más szerzők munkájából vettünk át, egyértelműen megjelöltük.

Ha kiderülne, hogy ez a nyilatkozat valótlan, tudomásul vesszük, hogy a szakmai vizsgabizottság a szakmai vizsgáról kizár és szakmai vizsgát csak új záródolgozat készítése után tehetünk.

Győr, 2022. március 17.

Tanulók aláírásai:

---

Magashegyi Ádám

---

Vadkerti Sára

Jedlik Ányos Gépipari és Informatikai  
Technikum és Kollégium

Move Your Body  
dokumentáció

Készítették:

Magashegyi Ádám

Vadkerti Sára

## Tartalomjegyzék

1. Bevezetés.....	6
1.1 A probléma és a megoldás.....	6
1.2 Az ötletelés.....	6
1.3 A végeredmény.....	7
1.4 A jövő.....	7
1.5 A csapatmunka .....	8
1.6 Publikálás .....	10
2. A program.....	11
2.1 Technikai részletek.....	11
2.2 Az adatbázis felépítése .....	12
2.2.1 Táblák:.....	12
2.3 Az adatbázistól a megjelenítésig .....	14
2.4 Autentikáció, biztonság .....	15
2.4.1 A technológia .....	15
2.4.2 Működése .....	15
2.4.3 Jelszavak.....	16
2.4.4 Jogosultságok .....	16
2.5 API dokumentáció.....	16
2.5.1 LocationController .....	16
2.5.2 UserController .....	17
2.5.3 TagController .....	19
2.5.4 CategoryController.....	20
2.5.5 TrainingController.....	20
2.5.6 TrainingSessionController.....	24
2.5.7 ApplicantController.....	25
2.5.8 TagTrainingController.....	26
2.5.9 AuthController.....	27
2.6 Frontend dokumentáció.....	28
2.6.1 Felépítése.....	28
3. Tesztek .....	29
3.1 Backend tesztek.....	29
3.2 Frontend tesztek .....	29
4. Felhasználói kézikönyv .....	30
4.1 Navigációs menü .....	30
4.1.1 Kezdőlap.....	30
4.1.2 Kategóriák .....	30

4.1.3 Tag-ek.....	30
4.2 Regisztráció .....	31
4.3 Bejelentkezés.....	31
4.4 Profil beállításai.....	32
4.5 Edzések.....	32
4.5.1 Megjelenítés .....	32
4.5.2 Létrehozás .....	33
4.5.3 Szerkesztés .....	33
4.5.4 Bőngészés.....	34
4.6 Alkalmak .....	35
4.6.1 Megjelenítés .....	35
4.6.2 Létrehozás .....	37
4.6.3 Törlés.....	38
4.6.4 Jelentkezés.....	38
4.6.5 Lemondás .....	38
4.7 Fejlesztői futtatási kézikönyv .....	39
4.8 Demo .....	39

# 1. Bevezetés

## 1.1 A probléma és a megoldás

A **MoveYourBody** egy online edzésfoglaló-, és gyűjtő webalkalmazás.

Manapság az edzés az emberek életének fontos tényezője lett. Azonban a különböző edzők mind más-más felületen, módon bonyolítják le edzéseik megszervezését és hirdetését. Ez megnehezíti az edzeni vágyók döntéshozatalát is, hiszen eddig nem volt egy egységes platform, ahol kereshettek preferenciáik szerint.

Így a program három problémát old meg egyszerre:

- megkönnyíti a kliensek számára a keresést, választást és jelentkezést,
- egységes felületet biztosít,
- az edzők számára a hirdetések, a jelentkezők és az időpontok kezelését leegyszerűsíti.

A weboldal bárholnan elérhető internetkapcsolat segítségével akár mobilról, tabletről vagy számítógépről is. A regisztráció, a hirdetés, a böngészés, a jelentkezés mind online felületen keresztül történik, ezért nem igényel semmiféle eszközön letöltést vagy telepítést.

## 1.2 Az ötletelés

Ebben a fejezetben az ötletelés fázisban megfogalmazott terveinket részletezzük.

Az edzők oldaláról biztosítani szeretnénk volna azt, hogy egy könnyen átlátható felületen tudjanak edzéseket létrehozni. Egy-egy edzéshez több alkalmat is hozzárendelhetnek, ami segítségével átláthatják a jelentkezők listáját is. Az edzésekhez különböző kategóriákat és tagokat rendelhetnek, amik a keresést segítik.

A sportolni vágyó kliensek oldalán a fő célunk a hatékony böngészés és a gyors jelentkezés biztosítása volt. Az edzéseket számos szempont alapján szűrhetik, majd pár kattintással feliratkozhatnak a jelentkezők listájára. A jelentkezéseiket később meg is tekinthetik egy külön oldalon. Ezt azért tartottuk fontosnak, mert tudjuk, hogy a mai ember rengeteg teendője között könnyen elfelejti a kisebb részleteket, de így nem is kell megjegyeznie.

Munkánkat jelentősen segítette hivatásos edzők és potenciális felhasználók véleménye. Segítségükkel valós igényekhez szabhattuk a weboldal működését.

## 1.3 A végeredmény

Eredeti terveink mellett még számos funkciót sikerült beépítenünk a programba.

Ilyenek például:

- az e-mailküldés,
- az adminisztrációs felület,
- a képek adatbázisban tárolása,
- a helyszínek kezelése.

A megvalósított tervek mellett viszont látunk fejlesztési lehetőségeket is a jövőre nézve, mint például az online bankkártyás fizetés megoldása.

## 1.4 A jövő

A projekt előzetes tervezésekor számtalan ötlettel álltunk elő. A végleges tervezéskor azonban mérlegelnünk kellett, mi az, ami a megadott határidőn belül kivitelezhető. Annak érdekében, hogy az alkalmazás leadáskor használható legyen, bizonyos funkciók megvalósításába nem kezdtünk bele. Ezeket, a jelenlegi verzióból kimaradt ötleteinket azonban a jövőben mindenképpen szeretnénk véghez vinni. Fontosnak tartjuk megemlíteni ezeket, hiszen megmutatják ambícióinkat és a fejlődés lehetőségeit, irányait a projekt szempontjából.

Az egyik legjelentősebb tervünk az online fizetés lehetősége. A jelentkezők bankkártyával előre kifizethetnék az edzés árát, ezzel csökkentve a készpénzes tranzakciókat, valamint a pénz beszedésével eltöltött időt is. Ezzel viszont az is járna, hogy korlátozni kellene a lemondási lehetőségeket. Gyakorlatban a legtöbb edző, ha egy napon belül szeretné lemondani egy jelentkező az edzést, ennek ellenére elkéri annak az árát, hiszen nem talál ilyen rövid időn belül másik embert a felszabadult helyre. Hasonlóképpen terveznénk ezt mi is megvalósítani: késői lemondás esetén ugyanúgy terhelve lenne a kliens bankkártyája.

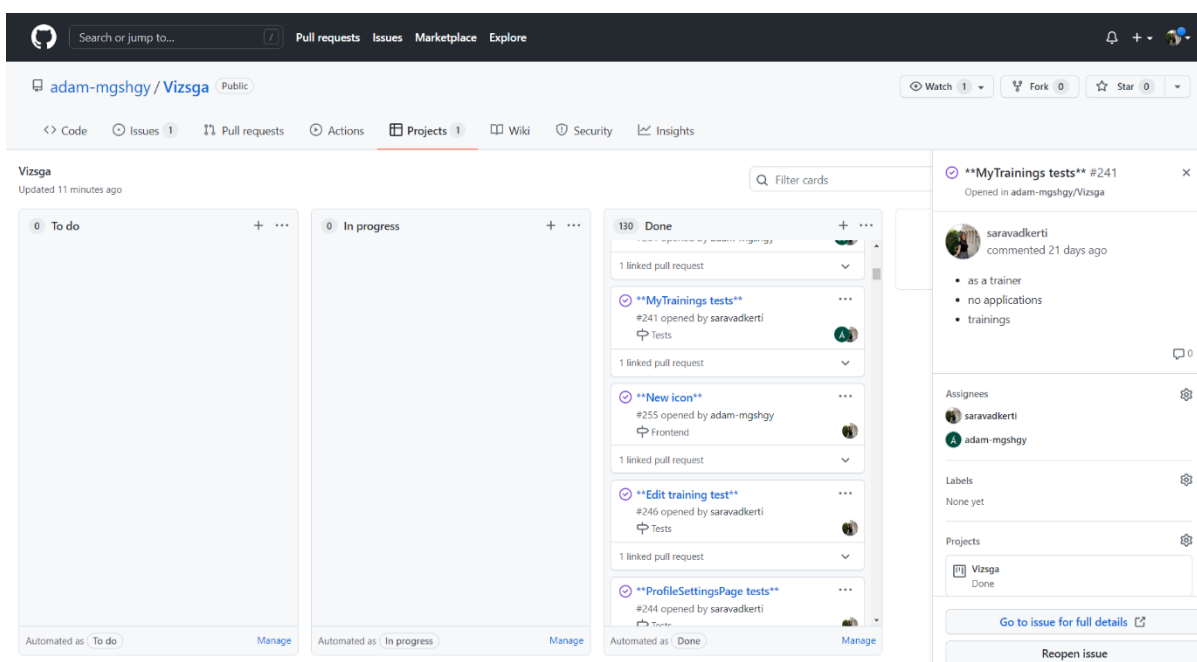
Ezen felül lehetőséget látunk az oldal hirdetési felületként való üzemeltetésében is. Az edzői jogosultság egy fizetős opcióként jelenne meg. Az edzések listázása mögött bonyolultabb üzleti logika lenne, például az előfizetés mérete szerint lennének rendezve.

Bár a weboldal teljes mértékben reszponzív, elképzelhetőnek tartjuk egy esetleges mobilalkalmazás fejlesztését is. Ennek érdekében alakítottuk úgy a backendet, hogy az a frontentdtől független legyen, arra az esetre, ha egy másik típusú alkalmazást is hozzá szeretnénk kapcsolni.

## 1.5 A csapatmunka

A programozás elkezdése előtt fontos volt, hogy megfelelően osszuk fel a feladatokat. Mindketten szerettünk volna teljeskörű ismereteket szerezni a webfejlesztés területén, ezért abban állapodtunk meg, hogy nem a frontend és a backend ketté választásával osztjuk meg a munkát. Helyette mindketten részt vettünk a fullstack webfejlesztésben és a feladatokat tartalmilag osztottuk szét. A csapatmunka megkönnyítése végett és a verziókövetéshez a GitHubot használtuk. Ezen a felületen nyomon követhető, ki, melyik feladatot végezte.

A GitHub repository-nk elérhető a <https://github.com/adam-mgshgy/Vizsga> URL-en.

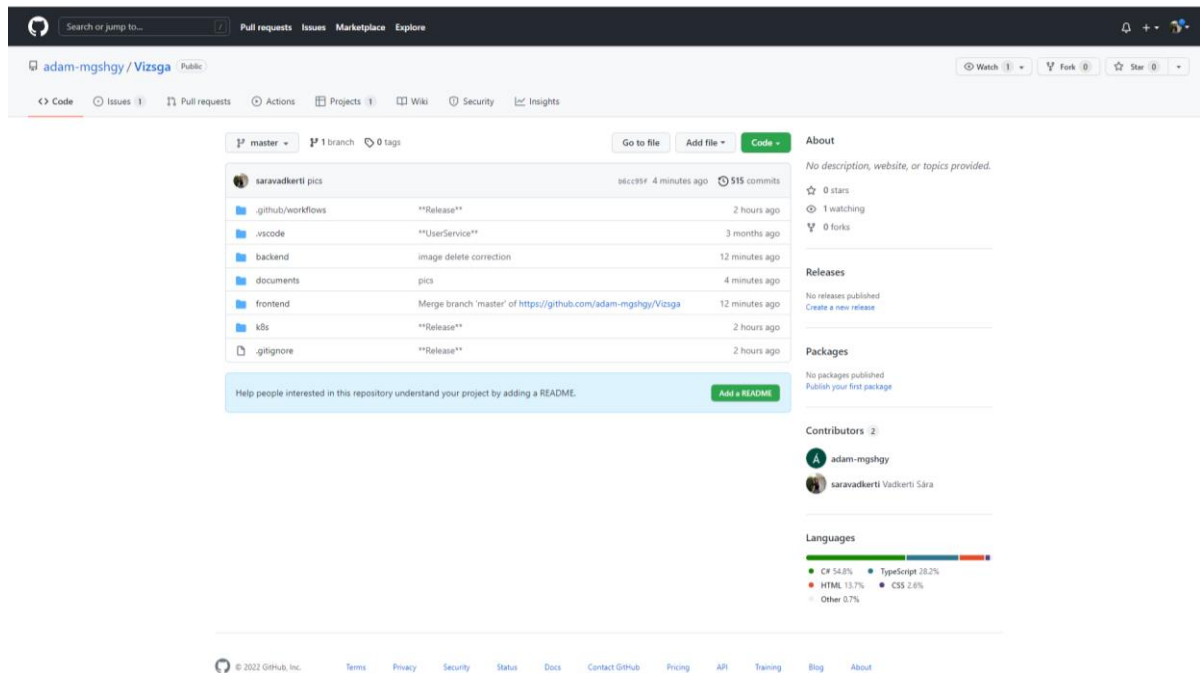


A projekt beadása előtt körülbelül 130 feladatot zártunk le. Ezek láthatóak a *Done* oszlopban. A kép jobb oldalán megtekinthető a feladathoz hozzárendelt leírás, amely leginkább akkor szolgált segítségünkre, amikor egymástól függetlenül, külön programoztunk. Így tudtuk egyértelművé tenni, mit szeretnénk véghezvinni. Emellett, amikor szükség volt rá, például tanítási szünetben, vagy hétvégéken a Google Meet-en folytattunk kommunikációt, így valósult meg ilyenkor a csapatmunka.

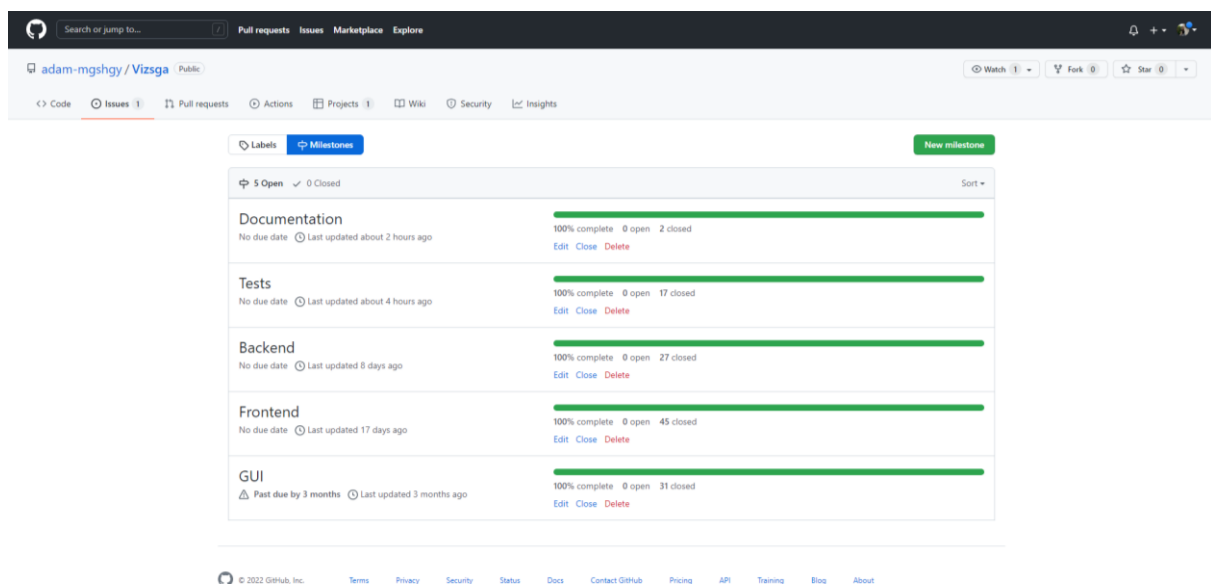
Minden feladat során külön branch-ben dolgoztunk, melyeket később, a helyes működés biztosítása után, Pull request segítségével töltöttünk fel a *Master* ágba. Ezzel elkerültük a már működő programkód hibással való felülírását, valamint így egyszerűen tudtunk egy időben dolgozni a programon.



## Move Your Body dokumentáció



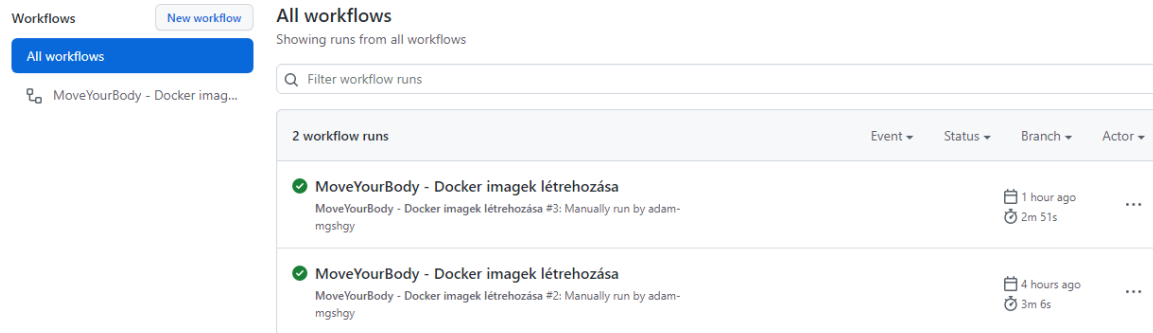
Erről a képről leolvasható, hogy milyen programnyelveket milyen arányban használtunk a fejlesztéskor. Az is látható, hogy a kép készítésekor 515 commitot hajtottunk végre. A mappastruktúra is megjelenik ezen az oldalon.



A feladatok rendszerezéséhez mérföldköveket hoztunk létre. Ezek alapján kategorizáltuk a projekt fejlesztésének lépéseit.

## 1.6 Publikálás

A demo alkalmazáshoz a GitHub Actions funkciójával Docker konténereket készítettünk, amelyek futtatását az iskolai Linux szerver végzi, Kubernetes segítségével. Az image-eket szintén a GitHubon tároljuk, így azok onnan bármikor letölthetők és futtathatók bármely másik, erre alkalmas szerveren, vagy felhőben.



Workflows New workflow

All workflows

Showing runs from all workflows

Filter workflow runs

2 workflow runs	Event	Status	Branch	Actor
<div>✓ MoveYourBody - Docker imagek létrehozása</div> <div>MoveYourBody - Docker imagek létrehozása #3: Manually run by adam-mgshgy</div>		1 hour ago 2m 51s		...
<div>✓ MoveYourBody - Docker imagek létrehozása</div> <div>MoveYourBody - Docker imagek létrehozása #2: Manually run by adam-mgshgy</div>		4 hours ago 3m 6s		...

```
k8s > ! moveyourbody-front-end-deployment.yaml
You, 2 seconds ago | 2 authors (adam-mgshgy and others)
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: moveyourbody-front-end-deployment
5  spec:
6    replicas: 1
7    selector:
8      matchLabels:
9        component: moveyourbody-front-end
10   template:
11     metadata:
12       labels:
13         component: moveyourbody-front-end
14     spec:
15       containers:
16       - name: client
17         image: ghcr.io/adam-mgshgy/moveyourbodycli:1.0.1
18         ports:
19         - containerPort: 80
```

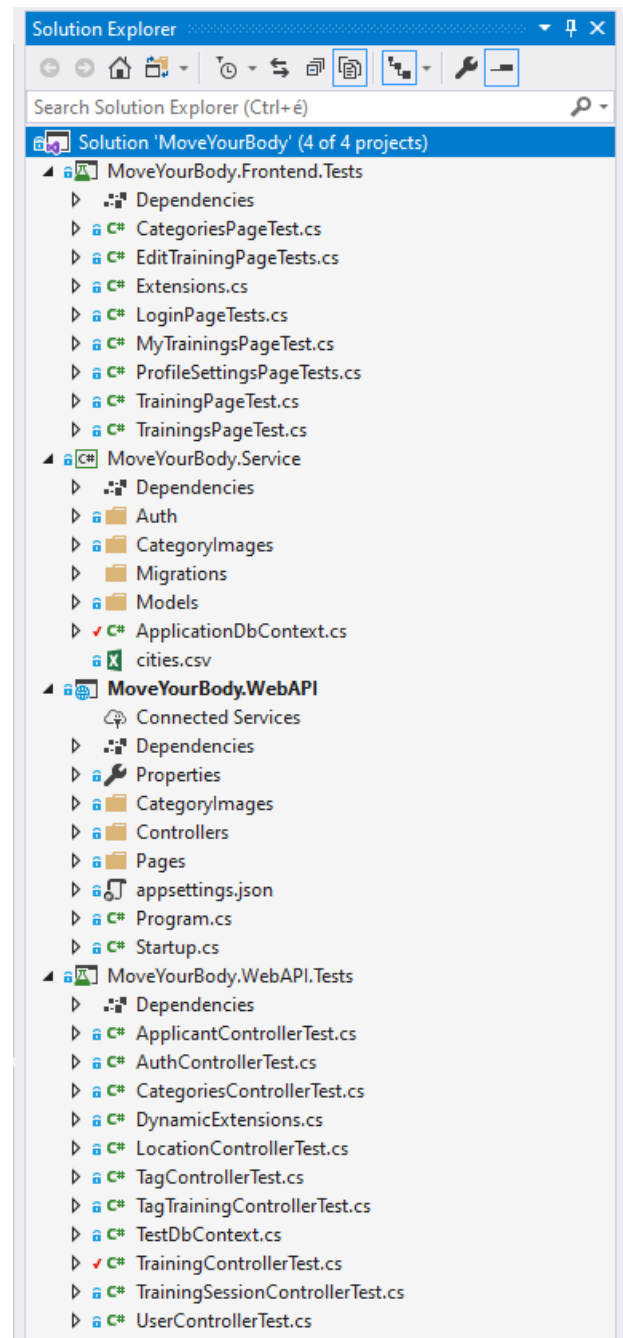
## 2. A program

### 2.1 Technikai részletek

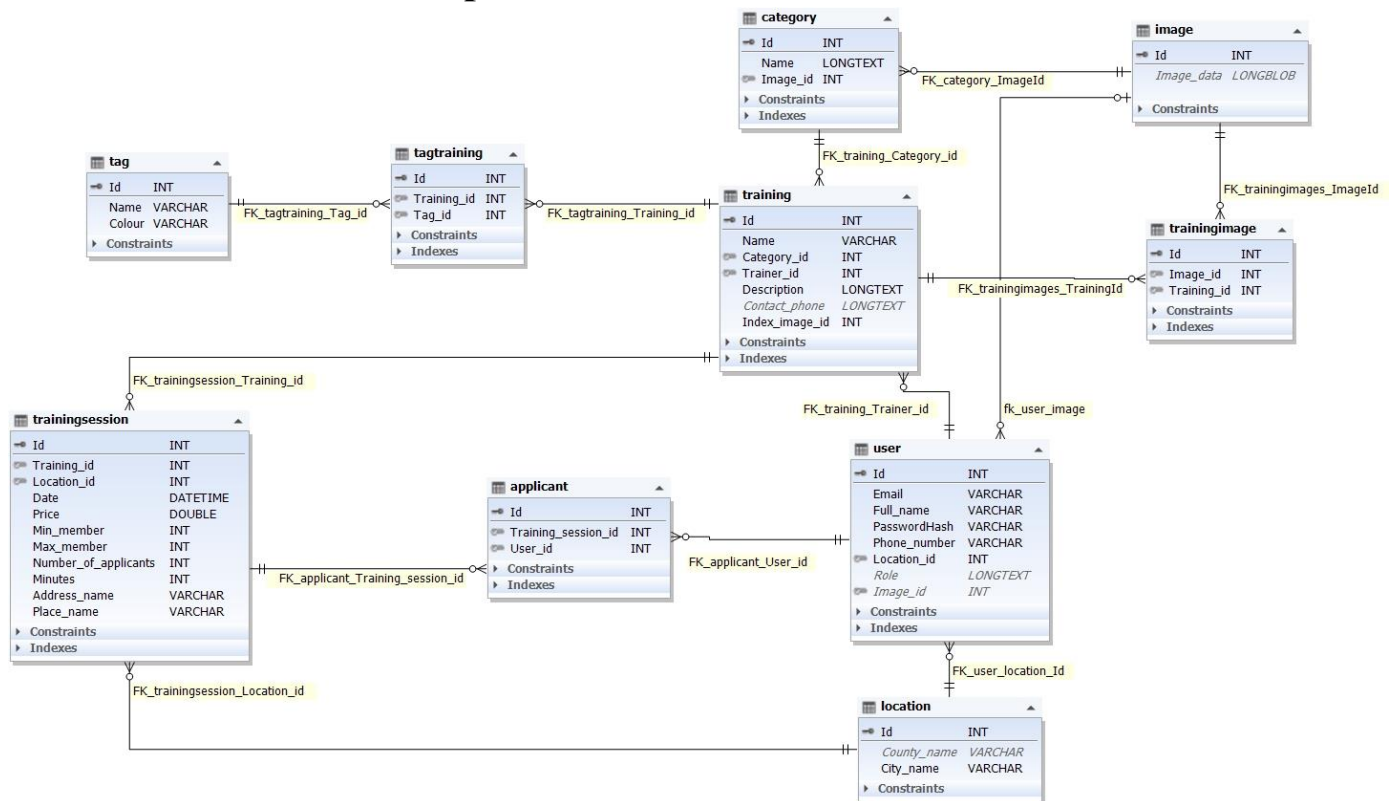
A projekt frontendje Angular keretrendszert használ. Az *AngularJS* egy Google által fejlesztett, nyílt forráskódú JavaScript keretrendszer dinamikus webes alkalmazásokhoz. Segítségével nagyban egyszerűsödik a webes alkalmazások frontend fejlesztése. Használatával a HTML eszköztára kibővül és az alkalmazások komponensei még egyértelműen elkülönülnek. Az Angular adatkapcsolásának, illetve függőség injektálásának köszönhetően, rengeteg felesleges boilerplate kód elhagyható. (ELTE IK, 2022)

A backend fejlesztéséhez az ASP.Net Core 3.1-es verzióját használtuk, mert ez volt a projekt elkezdésekor a legfrissebb olyan verzió, amely hosszú távú támogatásban részesül. A fejlesztés C# nyelven történt.

Az adatbázis és a backend összekötéséhez Entity Framework-öt használtuk. Az adatbázis létrehozása a Model First (Modell Először) elv alapján történt. A modellek megtervezése után az Entity Framework segítségével generáltuk le a MySQL adatbázist.



## 2.2 Az adatbázis felépítése



### 2.2.1 Táblák:

**Location - Hely**

- *Id*: elsődleges azonosító
- *County\_name*: megye neve
- *City\_name*: város neve

Az adatbázis létrehozásakor minden magyarországi város és megye bekerül ebbe a táblába.

## Image - Képek

- *Id*: elsődleges azonosító
- *Image\_data*: a kép base64 formátumban való tárolására szolgál

Az adatbázis létrehozásakor egy JSON fájlból kerülnek betöltésre az induláshoz szükséges képek, mint például a kategóriák képei.

### User - Felhasználó

- *Id*: elsődleges azonosító
- *Email*: felhasználó e-mail-címe, amely a bejelentkezéshez szükséges adatok egyike
- *Full\_name*: a felhasználó teljes neve
- *PasswordHash*: regisztráció során a backenden kerül titkosításra a felhasználó által megadott jelszó
- *Phone\_number*: a felhasználó telefonszáma, ami a kapcsolattartásra szolgálhat edző és jelentkező között
- *Location\_id*: idegenkulcs a Location táblával, a felhasználó által választott város kerül elmentésre
- *Role*: jogosultságok kezelésére szolgáló mező, bővebben a 2.4.4 fejezetben
- *Image\_id*: idegenkulcs az Image táblával, felhasználói profilkép tölthető fel

A regisztráció során kerülnek be az adatok a táblába.

### Tag - Címke

- *Id*: elsődleges azonosító
- *Name*: a tag neve
- *Colour*: a tag háttérszíne

Az adatbázis létrehozásakor néhány előre megadott tag-gel feltöltjük a táblát, melyet később az admin bővíthet. (2.4.4 fejezet)

### Category - Kategória

- *Id*: elsődleges azonosító
- *Name*: kategória neve
- *Image\_id*: idegenkulcs, a kategóriához tartozó, azt jellemző képhez

Az adatbázis létrehozásakor néhány előre megadott kategóriával feltöltjük a táblát, melyet később az admin bővíthet. (2.4.4 fejezet)

### Training - Edzés

- *Id*: elsődleges azonosító
- *Name*: az edzés neve
- *Category\_id*: idegenkulcs, ahhoz a kategóriához, melybe az edzés sorolható
- *Trainer\_id*: idegenkulcs, az edzést tartó felhasználó azonosítója
- *Description*: rövid leírás az edzésről
- *Contact\_phone*: alapértelmezetten üres, ha nem, akkor ez a telefonszám jelenik meg kapcsolattartási lehetőségként
- *Index\_image\_id*: idegenkulcs, az edzéshez kapcsolható több kép közül az elsődleges kép azonosítója

### TrainingSession - Edzés alkalom

- *Id*: elsődleges azonosító
- *Training\_id*: idegenkulcs, az edzés azonosítója, melyhez az alkalmat rendelte az edző
- *Location\_id*: idegenkulcs, alapértelmezetten az edző által regisztrációkor megadott város azonosítója

- *Date*: az edzés dátuma, formátuma: év. hónap. nap. óra:perc(yyyy.MM.dd. hh:mm)
- *Price*: az edzés ára forintban
- *Min\_member*: az edzés indításához szükséges minimum jelentkezők száma
- *Max\_member*: a jelentkezők maximális száma
- *Number\_of\_applicants*: jelentkezők száma az alkalomhoz, nulláról indul, jelentkezéskor vagy lemondáskor automatikusan változik
- *Minutes*: az edzés hossza percben
- *Address\_name*: az edzés címe
- *Place\_name*: az edzés helyszíne

#### **Applicant** - Jelentkező

- *Id*: elsődleges azonosító
- *Training\_session\_id*: idegenkulcs, az edzés alkalom azonosítója, melyre a felhasználó jelentkezett
- *User\_id*: idegenkulcs, a jelentkező felhasználó azonosítója

Az edzésre történő jelentkezéskor kerülnek be az adatok a táblába.

#### **TrainingImage** - Edzés-képek összekötő

- *Id*: elsődleges azonosító
- *Image\_id*: idegenkulcs, a kép azonosítója, mely az edzéshez lett feltöltve
- *Training\_id*: idegenkulcs, az edzés azonosítója melyhez a kép fel lett töltve

Az edzés létrehozása/módosítása során történő képfeltöltéskor kerülnek az adatok a táblába.

#### **TagTraining** - Tag-edzés összekötő

- *Id*: elsődleges azonosító
- *Training\_id*: idegenkulcs, az edzés azonosítója, melyhez a tag hozzá lett rendelve
- *Tag\_id*: idegenkulcs, a tag azonosítója, mely hozzá lett rendelve az edzéshez

Az adatbázis létrehozásakor, illetve az admin felületen történő feltöltéskor kerülnek az adatok a táblába

## **2.3 Az adatbázistól a megjelenítésig**

A backendünk két részből áll: egy WebAPI és egy Service rétegből. Az adatbázis így nem közvetlen a WebAPI-val, hanem a Service rétegen keresztül kommunikál. Utóbbi tartalmazza a modelleket, a hitelesítéshez (JWT) szükséges osztályokat, az induláshoz szükséges képeket tartalmazó fájlt, illetve a dbcontext osztályt, amely vezérli az adatbázis-kapcsolatok kezelését.

A WebAPI réteg tartalmazza:

- az *appsettings* fájlt, melyben megtalálható többek között a *connection string*, illetve más hitelesítéshez fontos adatok.

- a kontrollereket, melyek a HTTP kéréseket dolgozzák fel és küldik vissza a válaszokat
- a Swagger bővítményt, mely segítségével a backend futtatásakor egy helyen tekinthetjük meg az összes kérést logikusan, kontrollerekre bontva (2.5) (Swagger, 2022)

Frontendünk különböző servicek segítségével küldi el a backend felé a HTTP kéréseket, majd a kontrollereken keresztül éri el az adatbázis megfelelő adatait.

## 2.4 Autentikáció, biztonság

### 2.4.1 A technológia

A weboldalon történő bejelentkezéshez a JWT (JSON Web Token) szabványt használjuk, melyet a backend Service rétegében található osztályban definiálunk. Ebben az osztályban határozzuk meg a tokenben lévő adatokat, mint például a lejárat idejét vagy a felhasználó jogait. *A JWT egy olyan szabvány, amely kompakt és önálló módszert határoz meg az információk biztonságos továbbításához az ügyfél és a szerver között, mint JSON objektumot. A kompakt méret megkönnyíti a tokenek átadását URL-en, POST paraméteren vagy egy HTTP fejlécen keresztül. Továbbá, mivel önállóak, tartalmazzák a felhasználóval kapcsolatos összes szükséges információt, így az adatbázist nem kell többször lekérdezni.* (Ilusionity, 2022)

### 2.4.2 Működése

#### Backend oldalon:

A token létrehozása a kontrollerek között található AuthController nevű osztályban kezdődik. A Login függvény meghívásakor ellenőrzés után a JwtService segítségével e-mail és a felhasználó jogosultsága alapján a GenerateSecurityToken függvény hozza létre a hitelesítéshez szükséges tokenet.

#### Frontend oldalon:

Bejelentkezéskor a frontenden található authentication.service-ben található login függvény kerül meghívásra, mely a felhasználó által megadott e-mail és jelszó párosítást küldi tovább a backend felé. A válaszként kapott token, illetve az aktuálisan bejelentkezett felhasználó szükséges adatai a böngésző lokális tárolóján kapnak helyet.

A HTTP kérések során a backenden a kontrollerek határozzák meg, a felhasználónak van-e joga elérni azt. Frontenden az AuthGuard szolgáltatás ellenőrzi az adott felhasználó útvonalakhoz történő hozzáférését.

### 2.4.3 Jelszavak

Sikeres regisztráció után a backendre érkező jelszó titkosítására kerül sor, melyet ebben a formában tárolunk el az adatbázisban. A lekérések során ez a titkosított jelszó semmiféle formában nem kerül visszaküldésre a frontend irányába az esetleges támadások végett. A felhasználó által megadott titkosítatlan jelszó nem kerül tárolásra az adatbázisban.

### 2.4.4 Jogosultságok

Háromféle jogosultság alapján vannak a felhasználók besorolva: User, Trainer, Admin. Regisztrációkor választható, hogy edzőként vagy kliensként szeretné-e használni fiókját, admin joggal csak a rendszergazda rendelkezhet.

**User** típusú felhasználó:

A felhasználó regisztrációt követően böngészhet az edzések között, foglalhat időpontot edzőkhöz, illetve megtekintheti jelentkezéseit. Edzés elmaradása esetén a kliens e-mailben értesül.

**Trainer** típusú felhasználó:

A User típusú felhasználó jogain felül az edző a továbbiakkal is rendelkezik:

- edzőként létrehozhat edzést és azokhoz alkalmakat rögzíthet
- megtekintheti az alkalmakra jelentkezők nevét és telefonszámát. Ez azért szükséges, hogy változás vagy rendkívüli esetben könnyen elérhesse klienseit.

**Admin** típusú felhasználó:

Az Admin típusú felhasználó felvehet újabb kategóriákat, illetve tag-eket egy külön erre a célra létrehozott felületen.

## 2.5 API dokumentáció

### 2.5.1 LocationController

Elérési útvonal: /locations

**List()**

- Kérés típusa: HttpGet
- Paraméter: nincs
- Jogosultság: Trainer, Admin, User
- Feladat: kilistázza az összes várost, megyével együtt
- Válasz: *Location* típusú tömb
- Elérési útvonal: /

Location	
GET	/locations
GET	/locations/counties
GET	/locations/field



**ListCounties()**

- Kérés típusa: `HttpGet`
- Paraméter: nincs
- Jogosultság: `Trainer`, `Admin`, `User`
- Feladat: megyék alapján csoportosítva listáz
- Válasz: megye nevekből álló névtelen típusú tömb
- Elérési útvonal: `/counties`

**ListByField([FromQuery] string field)**

- Kérés típusa: `HttpGet`
- Paraméter: szöveges típusú
- Jogosultság: `Trainer`, `Admin`, `User`
- Feladat: kilistázza azokat a városokat, amelyekre igaz a megadott paraméter. A paraméter lehet azonosító, megye név vagy város név
- Válasz: *Location* típusú tömb
- Elérési útvonal: `/field/field=`

**2.5.2 UserController**

Elérési útvonal: `/user`

**GetById(int id)**

- Kérés típusa: `HttpGet`
- Paraméter: `int` típusú
- Jogosultság: `Trainer`, `Admin`, `User`
- Feladat: azonosító alapján megkeresi és visszaadja az adott felhasználót, ha létezik
- Válasz: *User* típusú osztálypéldány
- Elérési útvonal: `/id`

**SaveImages(string[] base64, [FromQuery] int userId)**

- Kérés típusa: `HttpPut`
- Paraméter: `base64` és `int` típusú
- Jogosultság: `Trainer`, `Admin`, `User`
- Feladat: eltárolja az adott felhasználó profilképét
- Válasz: nincs visszatérési érték
- Elérési útvonal: `/image/userId=`

User	
GET	<code>/user/{id}</code>
PUT	<code>/user/image</code>
GET	<code>/user/image</code>
GET	<code>/user/email</code>
PUT	<code>/user/register</code>
GET	<code>/user/getTrainer</code>
POST	<code>/user/modify</code>
DELETE	<code>/user/image/delete</code>
DELETE	<code>/user</code>

### **GetImageById(int imageId)**

- Kérés típusa: HttpGet
- Paraméter: int típusú
- Jogosultság: Trainer, Admin, User
- Feladat: azonosító alapján megkeresi és visszaadja az adott felhasználó képét
- Válasz: *Image* típusú osztálypéldány
- Elérési útvonal: /image

### **EmailExists([FromQuery] string email)**

- Kérés típusa: HttpGet
- Paraméter: szöveges típusú
- Jogosultság: Trainer, Admin, User
- Feladat: ellenőrzi, hogy a paraméterben kapott e-mail cím már létezik-e az adatbázisban
- Válasz: logikai érték
- Elérési útvonal: /email/email=

### **New(User user)**

- Kérés típusa: HttpPut
- Paraméter: *User* típusú osztálypéldány
- Jogosultság: nincs korlátozva
- Feladat: új felhasználó regisztrálása
- Válasz: *User* típusú osztálypéldány
- Elérési útvonal: /register

### **GetTrainer(int training\_id)**

- Kérés típusa: HttpGet
- Paraméter: int típusú
- Jogosultság: Trainer, Admin, User
- Feladat: a paraméterben kapott edzés azonosítója alapján megkeresi a hozzá tartozó edzőt
- Válasz: *User* típusú osztálypéldány
- Elérési útvonal: /getTrainer/trainingId=

### **Modify(User user)**

- Kérés típusa: HttpPost
- Paraméter: *User* típusú osztálypéldány
- Jogosultság: Trainer, Admin, User
- Feladat: felhasználó adatainak módosítása
- Válasz: *User* típusú osztálypéldány
- Elérési útvonal: /modify

### DeleteImage(int id)

- Kérés típusa: HttpDelete
- Paraméter: int típusú
- Jogosultság: Trainer, Admin, User
- Feladat: azonosító alapján megkeresi és törli a felhasználó képét
- Válasz: nincs visszatérési érték
- Elérési útvonal: /image/delete

### Delete(User user)

- Kérés típusa: HttpDelete
- Paraméter: *User* típusú osztálypéldány
- Jogosultság: Trainer, Admin, User
- Feladat: megkeresi az adott felhasználót és törli a fiókját. Emellett az esetleges jelentkezéseket, edzéseket, alkalmakat is törli
- Válasz: *User* típusú osztálypéldány
- Elérési útvonal: /

## 2.5.3 TagController

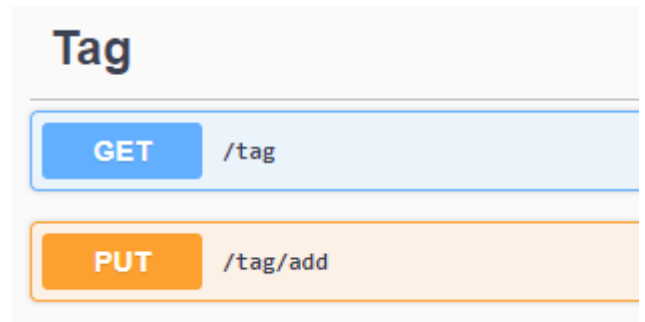
Elérési útvonal: /tag

### ListAll()

- Kérés típusa: HttpGet
- Paraméter: nincs
- Jogosultság: nincs korlátozva
- Feladat: listázza az összes tag-et
- Válasz: *Tag* típusú elemekből álló lista
- Elérési útvonal: /

### Add(Tag tag)

- Kérés típusa: HttpPut
- Paraméter: *Tag* típusú osztálypéldány
- Jogosultság: Admin
- Feladat: új tag mentése az adatbázisba
- Válasz: *Tag* típusú osztálypéldány
- Elérési útvonal: /add

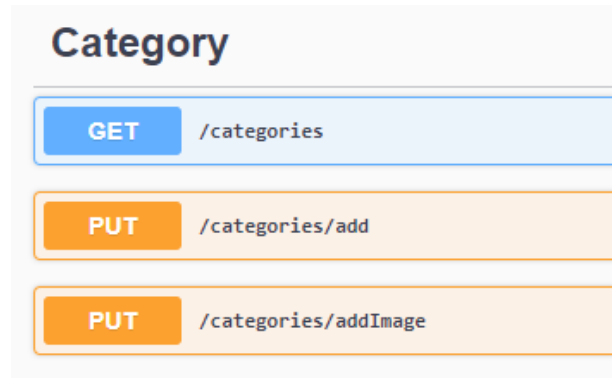


## 2.5.4 CategoryController

Elérési útvonal: /categories

### Get()

- Kérés típusa: HttpGet
- Paraméter: nincs
- Jogosultság: nincs korlátozva
- Feladat: listázza az összes kategóriát a hozzá tartozó képpel együtt
- Válasz: anonim osztály, mely tartalmaz egy *Category* és egy *Image* típusú elemekből álló listát
- Elérési útvonal: /



### Add(Category category)

- Kérés típusa: HttpPut
- Paraméter: *Category* típusú osztálypéldány
- Jogosultság: Admin
- Feladat: új kategória mentése az adatbázisba
- Válasz: *Category* típusú osztálypéldány
- Elérési útvonal: /add

### AddImage(Category category)

- Kérés típusa: HttpPut
- Paraméter: szöveges típusú
- Jogosultság: Admin
- Feladat: új kép mentése az adatbázisba
- Válasz: *Image* típusú osztálypéldány
- Elérési útvonal: /addImage

## 2.5.5 TrainingController

Elérési útvonal: /training

### SaveImages(string[] images, [FromQuery] int trainingId)

- Kérés típusa: HttpPut
- Paraméter: szöveges és int típusú
- Jogosultság: Trainer, Admin
- Feladat: képek feltöltése az adatbázisba a paraméterben kapott azonosító alapján megkeresett edzéshez
- Válasz: nincs visszatérési érték
- Elérési útvonal: /Images

**GetImageById(int id)**

- Kérés típusa: HttpGet
- Paraméter: int típusú
- Jogosultság: Trainer, Admin, User
- Feladat: azonosító alapján megkeresi és visszaadja az adott edzés képeit
- Válasz: anonim osztály, mely egy *TrainingImage* és egy *Image* típusú elemekből álló lista
- Elérési útvonal: /Images/id

**New(Training training)**

- Kérés típusa: HttpPut
- Paraméter: *Training* típusú osztálypéldány
- Jogosultság: Trainer, Admin
- Feladat: új edzés létrehozása
- Válasz: *Training* típusú osztálypéldány
- Elérési útvonal: /

**Modify(Training training)**

- Kérés típusa: HttpPost
- Paraméter: *Training* típusú osztálypéldány
- Jogosultság: Trainer, Admin
- Feladat: edzés részleteinek módosítása
- Válasz: *Training* típusú osztálypéldány
- Elérési útvonal: /modify

**Delete(Training training)**

- Kérés típusa: HttpDelete
- Paraméter: *Training* típusú osztálypéldány
- Jogosultság: Trainer, Admin
- Feladat: edzés törlése
- Válasz: *Training* típusú osztálypéldány
- Elérési útvonal: /

**DeleteImage(int[] id)**

- Kérés típusa: HttpDelete
- Paraméter: int típusú tömb
- Jogosultság: Trainer, Admin
- Feladat: azonosító alapján megkeresi és törli a képeket
- Válasz: nincs visszatérési érték
- Elérési útvonal: /Images/delete

Training	
PUT	/training/Images
GET	/training/Images/{id}
PUT	/training
DELETE	/training
POST	/training/modify
DELETE	/training/Images/delete
GET	/training/{id}
GET	/training/data
GET	/training/TrainerId/{trainerId}
GET	/training/UserId/{userId}
GET	/training/category
GET	/training/tag
GET	/training/all
GET	/training/county
GET	/training/city
GET	/training/name

### **GetById(int id)**

- Kérés típusa: `HttpGet`
- Paraméter: `int` típusú
- Jogosultság: nincs korlátozva
- Feladat: edzés megkeresése azonosító alapján
- Válasz: *Training* típusú osztálypéldány
- Elérési útvonal: `/id`

### **ListDataById([FromQuery] int trainingId)**

- Kérés típusa: `HttpGet`
- Paraméter: `int` típusú
- Jogosultság: nincs korlátozva
- Feladat: edzés részleteinek listázása azonosító alapján
- Válasz: anonim osztály, mely tartalmaz egy *Trainer*, egy *Training* és egy *Location* típusú osztálypéldányt
- Elérési útvonal: `/data/trainingId=`

### **GetByTrainerId(int trainerId)**

- Kérés típusa: `HttpGet`
- Paraméter: `int` típusú
- Jogosultság: nincs korlátozva
- Feladat: edző azonosítója alapján listázza az általa létrehozott edzéseket és azok részleteit
- Válasz: anonim osztály, mely tartalmaz egy *Trainer* típusú osztálypéldányt, egy *Training* és egy *TagTraining* típusú elemekből álló listát
- Elérési útvonal: `/TrainerId/trainerId`

### **GetByUserId(int userId)**

- Kérés típusa: `HttpGet`
- Paraméter: `int` típusú
- Jogosultság: nincs korlátozva
- Feladat: felhasználó azonosítója alapján listázza azokat az edzéseket, melyeknek valamelyik alkalmára jelentkezett és a szükséges részleteket
- Válasz: anonim osztály, mely tartalmaz, *Training*, *TagTraining*, *User*, *TrainingSession* és *Applicant* típusú elemekből álló listákat
- Elérési útvonal: `/UserId/userId`

### **GetByCategory([FromQuery] int id)**

- Kérés típusa: `HttpGet`
- Paraméter: `int` típusú
- Jogosultság: nincs korlátozva

- Feladat: kategória azonosítója alapján listázza az ahhoz tartozó edzéseket és a szükséges adatokat
- Válasz: anonim osztály, mely tartalmaz, *Training*, *TagTraining* és *User* típusú elemekből álló listákat, és egy *Category* típusú osztálypéldányt
- Elérési útvonal: /category/id=

#### **GetByTag([FromQuery] int id)**

- Kérés típusa: HttpGet
- Paraméter: int típusú
- Jogosultság: nincs korlátozva
- Feladat: tag azonosítója alapján listázza az ahhoz tartozó edzéseket és a szükséges adatokat
- Válasz: anonim osztály, mely tartalmaz, *Training*, *TagTraining* és *User* típusú elemekből álló listákat, és egy *Tag* típusú osztálypéldányt
- Elérési útvonal: /tag/id=

#### **GetAll()**

- Kérés típusa: HttpGet
- Paraméter: nincs
- Jogosultság: nincs korlátozva
- Feladat: listázza az összes edzést és a szükséges adatokat
- Válasz: anonim osztály, mely tartalmaz, *Training*, *TagTraining* és *User* típusú elemekből álló listákat
- Elérési útvonal: /all

#### **GetByCounty([FromQuery] string county)**

- Kérés típusa: HttpGet
- Paraméter: szöveges típusú
- Jogosultság: nincs korlátozva
- Feladat: megye alapján listázza az ott tartott edzéseket és a szükséges adatokat
- Válasz: anonim osztály, mely tartalmaz, *Training*, *TagTraining* és *User* típusú elemekből álló listákat
- Elérési útvonal: /county/county=

#### **GetByCity([FromQuery] string city)**

- Kérés típusa: HttpGet
- Paraméter: szöveges típusú
- Jogosultság: nincs korlátozva
- Feladat: város alapján listázza az ott tartott edzéseket és a szükséges adatokat
- Válasz: anonim osztály, mely tartalmaz, *Training*, *TagTraining* és *User* típusú elemekből álló listákat
- Elérési útvonal: /city/city=

**GetByName([FromQuery] string trainingName)**

- Kérés típusa: `HttpGet`
- Paraméter: szöveges típusú
- Jogosultság: nincs korlátozva
- Feladat: névegyezés alapján listázza az edzéseket és a szükséges adatokat
- Válasz: anonim osztály, mely tartalmaz, *Training*, *TagTraining* és *User* típusú elemekből álló listákat
- Elérési útvonal: `/name/trainingName=`

**2.5.6 TrainingSessionController**

Elérési útvonal: `/sessions`

**ListByTraining([FromQuery] int trainingId)**

- Kérés típusa: `HttpGet`
- Paraméter: `int` típusú
- Jogosultság: `Trainer`, `Admin`, `User`
- Feladat: listázza az összes alkalmat, ami a paraméterben kapott azonosítójú edzéshez tartozik
- Válasz: anonim osztály, mely tartalmaz egy *Category*, egy *User*, egy *Training* és egy *Image* típusú osztálypéldányt, valamint egy *TrainingSession* típusú elemekből álló listát
- Elérési útvonal: `/list/trainingId=`

TrainingSession	
GET	<code>/sessions/list</code>
GET	<code>/sessions/applied</code>
GET	<code>/sessions/get</code>
PUT	<code>/sessions/create</code>
POST	<code>/sessions/modify</code>
DELETE	<code>/sessions</code>

**ListAppliedSessions([FromQuery] int trainingId, [FromQuery] int userId)**

- Kérés típusa: `HttpGet`
- Paraméter: `int` típusú
- Jogosultság: `Trainer`, `Admin`, `User`
- Feladat: listázza az összes alkalmat az adott azonosítójú edzéshez, amire a paraméterben kapott azonosítójú felhasználó jelentkezett
- Válasz: anonim osztály, mely tartalmaz egy *Training* típusú osztálypéldányt, valamint egy *TrainingSession* típusú elemekből álló, dátum alapján rendezett listát
- Elérési útvonal: `/applied/trainingId=&userId=`

**GetById([FromQuery] int sessionId)**

- Kérés típusa: `HttpGet`
- Paraméter: `int` típusú
- Jogosultság: nincs korlátozva
- Feladat: listázza az adott azonosítójú alkalmat és részleteit



- Válasz: anonim osztály, mely tartalmaz egy *Training*, egy *Location* és egy *Session* típusú osztálypéldányt
- Elérési útvonal: /get/sessionId=

### CreateSession(TrainingSession session)

- Kérés típusa: HttpPut
- Paraméter: *TrainingSession* típusú osztálypéldány
- Jogosultság: Trainer, Admin
- Feladat: új alkalom létrehozása
- Válasz: *TrainingSession* típusú osztálypéldány
- Elérési útvonal: /create

### Modify(TrainingSession session)

- Kérés típusa: HttpPost
- Paraméter: *TrainingSession* típusú osztálypéldány
- Jogosultság: Trainer, Admin
- Feladat: alkalom módosítása
- Válasz: *TrainingSession* típusú osztálypéldány
- Elérési útvonal: /modify

### Delete(TrainingSession session)

- Kérés típusa: HttpDelete
- Paraméter: *TrainingSession* típusú osztálypéldány
- Jogosultság: Trainer, Admin
- Feladat: edzés törlése
- Válasz: *TrainingSession* típusú osztálypéldány
- Elérési útvonal: /

## 2.5.7 ApplicantController

Elérési útvonal: /applicants

### ListBySessionId([FromQuery] int trainingSessionId)

- Kérés típusa: HttpGet
- Paraméter: int típusú
- Jogosultság: Trainer, Admin, User
- Feladat: azonosító alapján listázza az adott alkalomra jelentkezőket
- Válasz: anonim osztály, mely *Applicant* és *User* típusú elemekből álló listákat tartalmaz
- Elérési útvonal: /list/session/trainingSessionId=

Applicant	
GET	/applicants/list/session
GET	/applicants/list/user
PUT	/applicants/add
DELETE	/applicants
DELETE	/applicants/delete

**ListByUserId([FromQuery] int userId)**

- Kérés típusa: `HttpGet`
- Paraméter: `int` típusú
- Jogosultság: Trainer, Admin, User
- Feladat: azonosító alapján listázza az adott felhasználó jelentkezéseit
- Válasz: *TrainingSession* típusú elemekből álló lista
- Elérési útvonal: `/list/user/userId=`

**AddApplicant(Applicant applicant)**

- Kérés típusa: `HttpPut`
- Paraméter: *Applicant* típusú osztálypéldány
- Jogosultság: Trainer, Admin, User
- Feladat: új jelentkezés rögzítése
- Válasz: *Applicant* típusú osztálypéldány
- Elérési útvonal: `/add`

**Delete([FromQuery] int applicantId)**

- Kérés típusa: `HttpDelete`
- Paraméter: `int` típusú
- Jogosultság: Trainer, Admin, User
- Feladat: jelentkezés törlése
- Válasz: *Applicant* típusú osztálypéldány
- Elérési útvonal: `/applicantId=`

**DeleteByIds([FromQuery] int userId, [FromQuery] int sessionId)**

- Kérés típusa: `HttpDelete`
- Paraméter: `int` típusú
- Jogosultság: Trainer, Admin, User
- Feladat: jelentkezés törlése azonosítók alapján
- Válasz: *Applicant* típusú osztálypéldány
- Elérési útvonal: `/delete/userId=&sessionId=`

**2.5.8 TagTrainingController**

Elérési útvonal: `/tagTraining`

**New(TagTraining tagTraining)**

- Kérés típusa: `HttpPut`
- Paraméter: *TagTraining* típusú osztálypéldány
- Jogosultság: Trainer, Admin

TagTraining	
PUT	/tagTraining
DELETE	/tagTraining
GET	/tagTraining/tag
GET	/tagTraining/training
GET	/tagTraining/refresh

- Feladat: edzés létrehozásakor a kiválasztott tag-ek mentése
- Válasz: *TagTraining* típusú osztálpéldány
- Elérési útvonala: /

#### **ListByTag([FromQuery] int id)**

- Kérés típusa: *HttpGet*
- Paraméter: *int* típusú
- Jogosultság: nincs korlátozva
- Feladat: tag-edzés kapcsolatok listázása tag alapján
- Válasz: *TagTraining* típusú elemekből álló lista
- Elérési útvonala: /tag?id=

#### **ListByTraining([FromQuery] int id)**

- Kérés típusa: *HttpGet*
- Paraméter: *int* típusú
- Jogosultság: nincs korlátozva
- Feladat: tag-edzés kapcsolatok listázása edzés alapján
- Válasz: *TagTraining* típusú elemekből álló lista
- Elérési útvonala: /training?id=

#### **Delete(TagTraining tagTraining)**

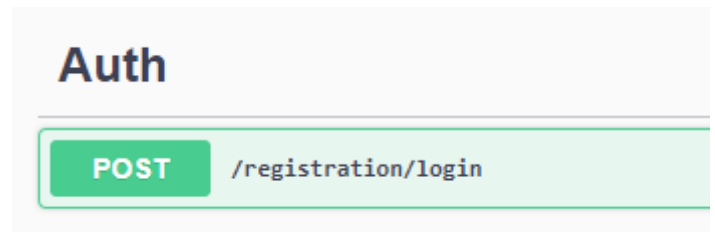
- Kérés típusa: *HttpDelete*
- Paraméter: *TagTraining* típusú osztálpéldány
- Jogosultság: Trainer, Admin
- Feladat: tag-edzés kapcsolat törlése
- Válasz: *TagTraining* típusú osztálpéldány
- Elérési útvonala: /

### **2.5.9 AuthController**

Elérési útvonala: /registration/login

#### **Login(LoginModel login)**

- Kérés típusa: *HttpPost*
- Paraméter: *LoginModel* típusú osztálpéldány
- Jogosultság: nincs korlátozva
- Feladat: bejelentkezés ellenőrzése, token létrehozása
- Válasz: anonim osztály, mely tartalmazza a *string* típusú token és az *int* típusú felhasználói azonosítót
- Elérési útvonala: /



## 2.6 Frontend dokumentáció

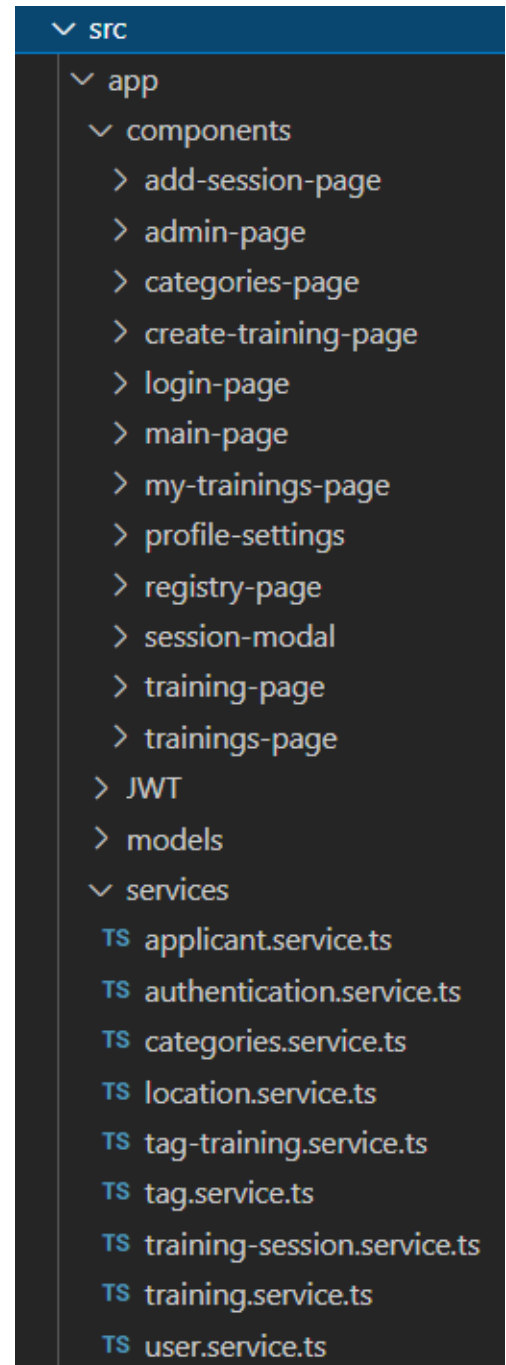
### 2.6.1 Felépítése

Frontendünk struktúrája főként a komponensekből, modellekből, a JWT-hez szükséges fájllokból és a servicekből áll.

A modellek megegyeznek a backend service rétegének modelljeivel. A frontend servicei a backend azonos elnevezésű kontrollereinek RestAPI kéréseit és válaszait kezeli.

A komponensek kezelik a webalkalmazás oldalainak megjelenítését. Elnevezésük beszédes, kifejezi, melyik oldal milyen funkciót biztosít. A komponensek mappái három fájlt tartalmaznak: a html-t, a css-t és a script fájlt. Ezek mappán belül összeköttetésben vannak, valamint komponensek közötti kommunikáció is megvalósul a szükséges helyeken. Ezeket dependency injectionnel érjük el. *A dependency injection lényege, hogy egy objektum más objektumok függőségeit elégíti ki.* (Wikipédia, 2022)

Az oldalakról bővebben a 4. Felhasználói kézikönyv című fejezetben írunk. A mappaszerkezet a képen látható.



## 3. Tesztek

### 3.1 Backend tesztek

A backend teszteket memória adatbázissal végezzük. Ennek lényege, hogy a teszt során végzett módosításokat nem az eredeti adatbázison hajtjuk végre, így annak tartalma nem módosul. A memória adatbázis is feltöltésre kerül mintaadatokkal, esetenként azok segítségével ellenőrizzük a függvények és kérések működését.

Az összes kontroller minden kérésére írtunk Unit tesztet, ezzel biztosítottuk azok rendeltetésszerű működését. Ezeket a kép is mutatja.

▲ ✓ MoveYourBody.WebAPI.Tests (48)	11,5 sec
▲ ✓ MoveYourBody.WebAPI.Tests (48)	11,5 sec
▶ ✓ ApplicantControllerTest (5)	275 ms
▶ ✓ AuthControllerTest (1)	1,3 sec
▶ ✓ CategoriesControllerTest (2)	1,1 sec
▶ ✓ LocationControllerTest (5)	243 ms
▶ ✓ TagControllerTest (2)	59 ms
▶ ✓ TagTrainingControllerTest (5)	227 ms
▶ ✓ TrainingControllerTest (15)	701 ms
▶ ✓ TrainingSessionControllerTest (6)	3,9 sec
▶ ✓ UserControllerTest (7)	3,7 sec

### 3.2 Frontend tesztek

Frontend oldalon a Selenium webes bővítmény segítségével hajtjuk végre a tesztelési fázist. A Selenium a megírt programkód futtatásával automatikusan a böngészőben végrehajtva az utasításokat teszteli a webalkalmazás frontend oldali funkcióit. A tesztelés nem memória adatbázison fut, az eredeti adatokat módosítva kerülnek tesztelésre a webalkalmazás oldalai. A tesztek a frontenden lévő főbb funkciókat ellenőrzik. (Selenium, 2022)

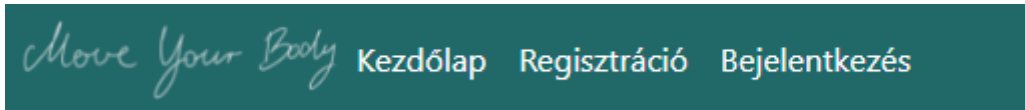
▲ ✓ MoveYourBody.Frontend.Tests (19)	3,4 min
▲ ✓ MoveYourBody.Frontend.Tests (19)	3,4 min
▲ ✓ CategoriesPageTest (3)	13,9 sec
✓ CategoriesDropdown	3,3 sec
✓ CheckUrl	6,6 sec
✓ TextOfCategories	3,9 sec
▶ ✓ EditTrainingPageTests (1)	50 sec
▶ ✓ LoginPageTests (1)	15,3 sec
▲ ✓ MyTrainingsPageTest (3)	1 min
✓ TrainerViewApplications	19 sec
✓ TrainerViewTrainings	19 sec
✓ UserViewApplications	22,4 sec
▲ ✓ ProfileSettingsPageTests (3)	18,8 sec
✓ DataLoadedTest	10,1 sec
✓ PasswordChangeTest	4,5 sec
✓ TrainerChangeTest	4,3 sec
▲ ✓ TrainingPageTest (1)	19,1 sec
✓ ApplyTraining	19,1 sec
▲ ✓ TrainingsPageTest (7)	27,9 sec
✓ FilterByCategory	3,8 sec
✓ FilterByCity	4 sec
✓ FilterByCounty	3,8 sec
✓ FilterBySearch	4,6 sec
✓ FilterByTag	4,2 sec
✓ FilterByTrainer	3,5 sec
✓ TextOfTrainings	3,8 sec

## 4. Felhasználói kézikönyv

### 4.1 Navigációs menü

#### 4.1.1 Kezdőlap

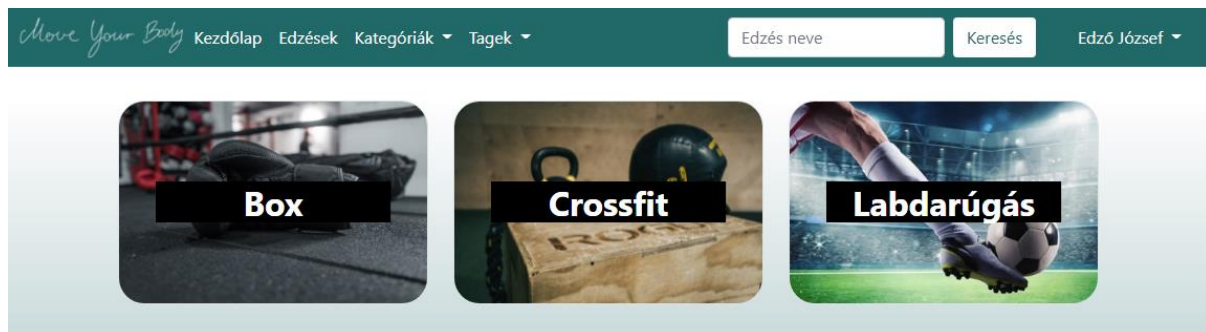
A weboldal megnyitásakor a *Kezdőlap* jelenik meg. Az oldal tartalmaz linkeket a *Regisztrációhoz* és a *Bejelentkezéshez*. A *Kezdőlap* megtekinthető a logóra kattintva is.



1. ábra – Navigációs menü

#### 4.1.2 Kategóriák

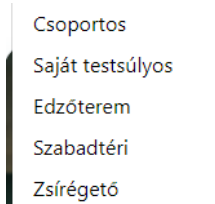
A navigációs menü *Kategóriák* feliratú legördülő menüpontja melletti nyílra kattintva jelenik meg az összes kategória. A felíratra kattintva azok egy külön oldalon tekinthetők meg megfelelő háttérkép kíséretében. A kép közepén található felíratra kattintva az *Edzések* oldal a kategória szerinti szűréssel kerül megjelenítésre.



2. ábra – Kategóriák oldal részlet

#### 4.1.3 Tag-ek

A navigációs menü *Tag-ek* feliratú legördülő menüpontja melletti nyílra kattintva jelenik meg az összes tag. A tag kiválasztásával az *Edzések* (9. ábra) oldal ez alapján történő szűréssel kerül megjelenítésre.



3. ábra – Tag-ek legördülő menüpont részlet

## 4.2 Regisztráció

Az oldal navigációs menüjében található *Regisztráció* gombot választva megjelenik a regisztrációs felület. Itt az *e-mail cím*, *teljes név*, *telefonszám*, *preferált megye*, *város* és *új jelszó* megadásával, valamint az *edzői jogosultság kérésével* hozható létre új **MoveYourBody** fiók. Bármilyen hiba esetén a hibaüzenet az oldal alján jelenik meg. A *Regisztráció* gomb megnyomása után, ha nincs hiba, elkészül az új fiók. Ezután automatikusan a *Bejelentkezés* oldal jelenik meg.

## 4.3 Bejelentkezés

Az oldal navigációs menüjében található *Bejelentkezés* gombra kattintva, vagy sikeres regisztráció után automatikusan megjelenik a bejelentkezési felület. Itt a regisztrációkor megadott *e-mail cím* és *jelszó* páros megadásával beléphet a felhasználói fiókjába. Hiba esetén az oldal tartalma alatt jelenik meg a hibaüzenet. Bejelentkezés után megjelenik a navigációs menü jobb oldalán a felhasználó teljes neve, valamint 30 perc elteltével újra be kell jelentkezni a navigálás folytatásához a felhasználó biztonsága érdekében.

**Bejelentkezés**

E-mail cím

Jelszó

Regisztrálni szeretnék!

Bejelentkezés

5. ábra – Bejelentkezési felület

**Regisztráció**

A mezők kitöltése kötelező!

Teljes név:

E-mail cím:

Az e-mail cím regisztráció után nem módosítható!

Telefonszám:

Kérjük ezt a formátumot használni: +36123456789

Helyszín

Kérjük adja meg a várost és megyét, ahol szívesen edzene!

Megye:

Jelszó:

Jelszó ismét:

☐ Edzőként szeretnék regisztrálni

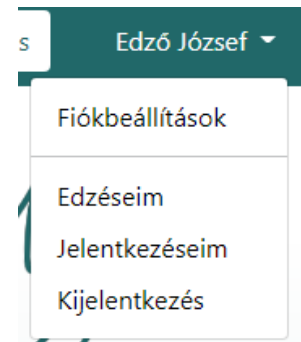
Regisztráció Mégsem

4. ábra – Regisztrációs felület



## 4.4 Profil beállításai

Az oldal navigációs menüjének jobb oldalán található a felhasználó teljes neve, melyre kattintva egy lenyíló lista jelenik meg. A listában található a *Fiókbeállítások* menüpont. Erre kattintva a *Regisztrációhoz* hasonló felületen szerkeszthetők a korábban megadott adatok. Probléma esetén az oldal alján hibaüzenet figyelmeztet. A jelszó módosítása külön gombnyomással valósítható meg. Sikeres módosítás esetén a *Kezdőlapra* irányít át a rendszer.




6. ábra – Felhasználói menü

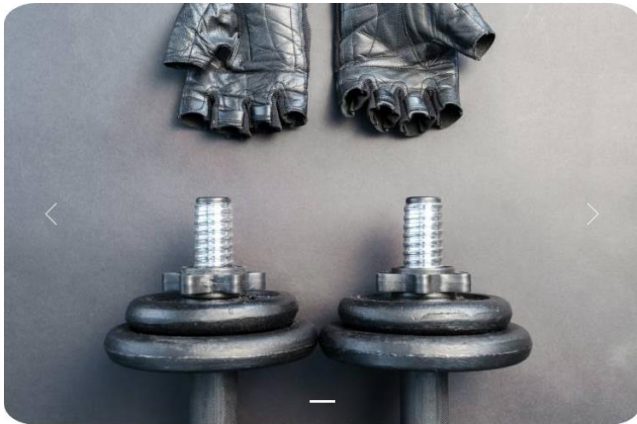
## 4.5 Edzések

### 4.5.1 Megjelenítés

**Box Józsiával**



**Edző József**  
+36701234567



**Box**

Csoportos
Saját testsúlyos
Aerobic

Köredzés

**Az edzés leírása:**

Szeretettel vár Józsi Edző a legnépszerűbb box edzésén! Kezdőket és haladókat is szívesen fogadunk. Nem kell más, csak egy törölköző, víz, és hatalmas lelkesedés!

Város	Cím	Időpont	Ár (Ft)	Hossz (Perc)	Jelentkezők száma	
Csornvás	Virág utca 5. Sportközpont	2022.03.28. 12:30	1500	45	2/10	<div style="border: 1px solid #00796b; padding: 2px 10px; color: #00796b; font-weight: bold;">Jelentkezek!</div>

7. ábra – Edzés oldala

Böngészéskor az edzés nevére kattintva megjelenik annak külön oldala. Itt látható az *edzés neve*, *rövid leírása*, *kategóriája*, a hozzá tartozó *tag-ek* és az *alkalmak*. Az edzéshez feltöltött képeket is itt tekintheti meg a felhasználó. A szervező *edző neve*, *képe* és a *kapcsolattartási telefonszáma* is itt érhető el.



A navigációs menüben a *saját név*-re kattintva a *Jelentkezéseim* opciót választva jelennek meg azok az edzések, melyekre jelentkezett a felhasználó. Jelentkezést követően ezek az edzések azonnal ide kerülnek.

#### 4.5.2 Létrehozás

Edzőként a navigációs menüben a *saját név*-re kattintva az *Edzéseim* opciót választva jelennek meg a felhasználó által létrehozott edzéseket. Az *Új edzés* gombra kattintva érhetjük el az *Edzés létrehozása* oldalt, ahol a megfelelő adatok kitöltésével, opcionálisan képek feltöltésével hozható létre új edzés. A *Másik telefonszámot adok meg* lehetőséggel a saját telefonszám helyett, kliensek általi kapcsolattartás céljára másik telefonszám adható meg. A *mégsem* gombbal azonnal megszakíthatjuk az új edzés felvételének menetét. Sikertelen mentés esetén a hiba, a korábbiakhoz hasonlóan, az oldal alján jelenik meg. Sikeres mentés után a rendszer az *Edzéseim* oldalra átirányít.

8. ábra – Edzés létrehozása

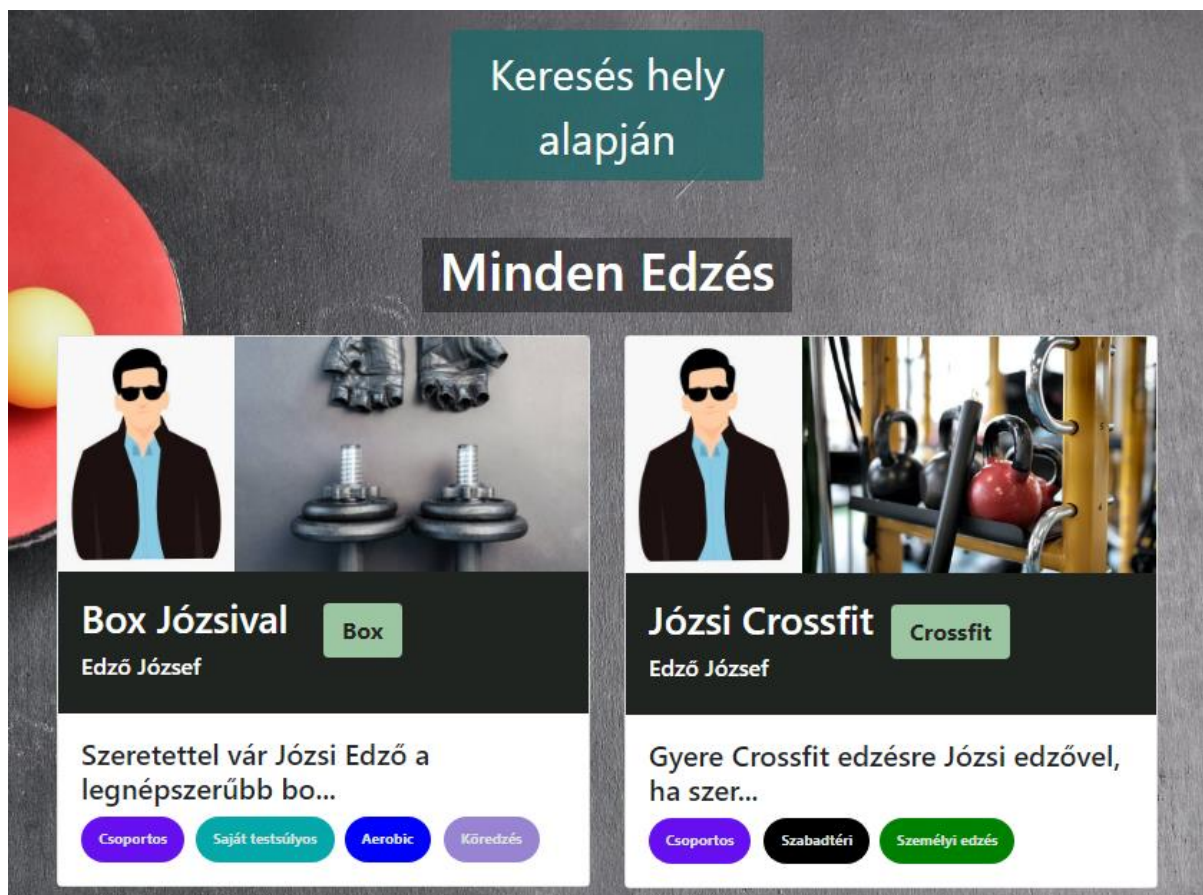
#### 4.5.3 Szerkesztés

Az *Edzéseim* oldalon bármely edzéshez tartozó *Szerkesztés* gombra kattintva a már létrehozott edzéseket szerkeszthetjük külön oldalon, melynek mezői az oldal betöltésekor már az aktuálisan mentett adatokat tartalmazzák. Az *Edzés mentése* gombra kattintva a megváltoztatott adatokat menthetjük.

#### 4.5.4 Böngészés

A navigációs menü *Edzések* feliratú gombját választva böngészhet a felhasználó a meghirdetett edzések között. Ilyenkor az összes jelenleg elérhető edzés kerül megjelenítésre.

A *Keresés hely alapján* gombra kattintva a felhasználó választhat megyét vagy várost, ahol edzést keresni szeretne. Ha az edzésekártyákon található kategóriánévre kattint, akkor az adott kategóriához tartozó edzések jelennek meg. A tag-ekre kattintva szintén meg lehet tekinteni az összes olyan edzést, amely a választott tag-gel rendelkezik. Az edző nevére nyomva az általa létrehozott edzések kerülnek kilistázásra.



9. ábra – Edzések oldal részlet

## 4.6 Alkalmak

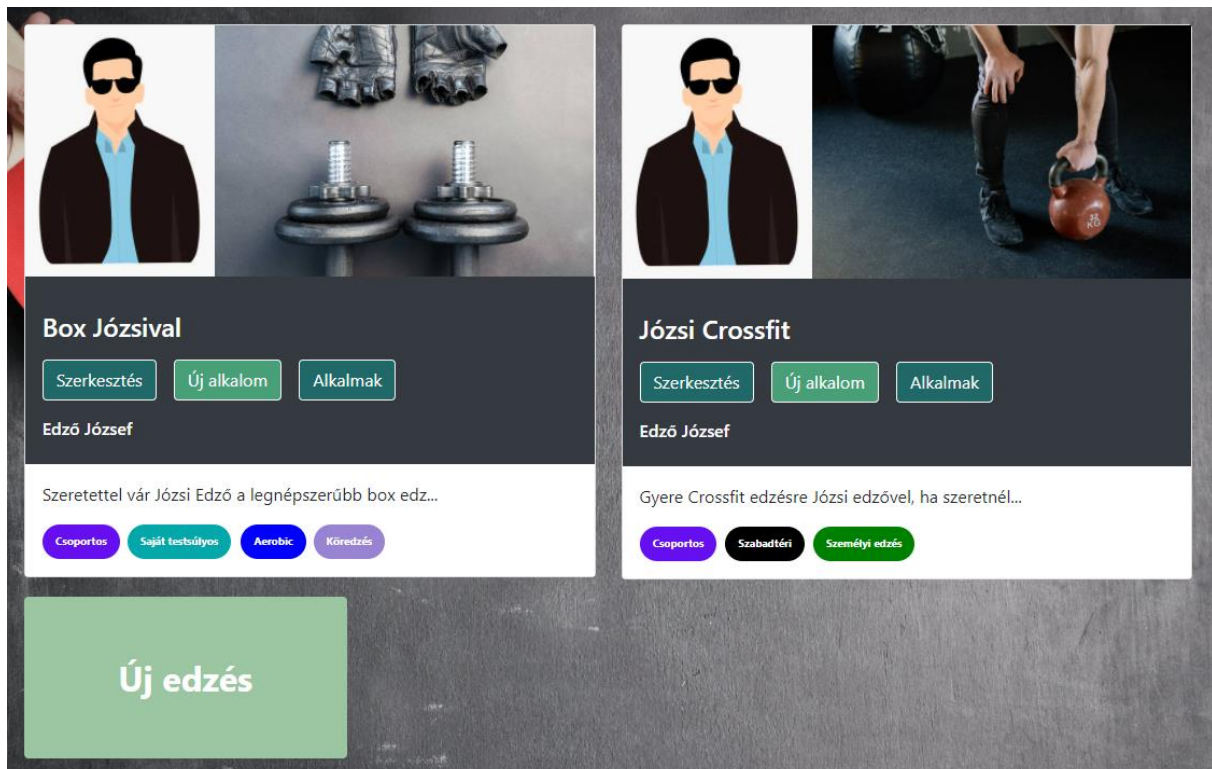
### 4.6.1 Megjelenítés

#### Edzői oldal:

Edzőként az *Edzéseim* fülre kattintva az általa létrehozott edzéseket tekintheti meg. Itt az adott edzés kártyáján található *Alkalmak* gombot megnyomva láthatóak az eddig létrehozott alkalmak, vagy azok hiányában a *Még nincsenek alkalmak létrehozva* felirat. Itt láthatja az edző az időpontokat, amelyeket gombok segítségével duplikálhat vagy törölhet. Az időpontra kattintva megjelennek az eddigi jelentkezők adatai. Felirat jelzi, hogy az alkalomra van-e elég jelentkező, vagy ha az edzés ezen időpontja elérte a maximális létszámot.

Időpont			
2022.02.10. 12:30	Duplikálás	Törölés	Lejárt
2022.03.28. 12:30	Duplikálás	Törölés	
Jelentkezők <span>2/10</span>			
User Dániel +36701234563			
User Gabriella +36701234568			
2022.04.10. 12:30	Duplikálás	Törölés	

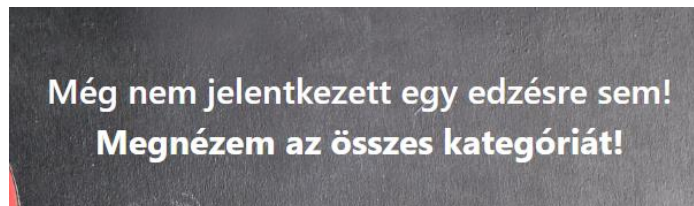
10. ábra – Alkalmak edzői oldalon



11. ábra – Edzéseim oldal edzői oldalon

**Kliens oldal:**

Kliensként és edzői fiókkal is a *Jelentkezéseim* oldalon lehet megtekinteni azokat az edzéseket, melyekre jelentkezett. Itt az alkalom *időpontja*, *helyszíne*, *ára* is megjelenik. Lemondásra is ezen a felületen van lehetőség.



12. ábra – Jelentkezéseim oldal, ha nem jelentkezett a felhasználó egy edzésre sem

Időpont

---

2022.04.20. 12:30 Lemondás ^

Részletek ^

Helyszín	Csurgó, Virág utca 2.
Hossz	45 perc
Ár	1500 Ft

13. ábra – Jelentkezés a kliens oldaláról

**Box Józsiával**

Jelentkezések

Edző József

Szeretettel vár Józsi Edző a legnépszerűbb box edz...

Csoporthoz
Saját testúlyos
Aerobic
Köredzés

14. ábra – Jelentkezéseim kliens oldalról



#### 4.6.2 Létrehozás

Edzőként új alkalom az *Edzéseim* oldalon az edzésekártyán hozható létre. Itt az aktuális *edzés* neve és leírása is fel van tüntetve az edzők munkájának könnyítése érdekében. Az alkalom létrehozásához dátumot kell választani, év, hónap, nap, valamint óra és perc részletességgel. A megye és város automatikusan arra választódik ki, amelyiket az edző regisztrációkor megadott, a folyamat gyorsítása végett, de ez, természetesen, megváltoztatható. A helyszín pontosításához szükséges még megadni a közterület nevét és számát, valamint a létesítmény nevét. Az alkalmon résztvevők *minimális* és *maximális* számát, az *edzés* hosszát *percben* és *árát forintban* kötelezően meg kell adni. A *Mégsem* gombbal elvethető az addig bevitt adat. Ha minden mező helyesen van kitöltve, az *Alkalom mentése* gomb

megnyomása után megtörténik a létrehozás. Hiba esetén az oldal alján jelenik meg a hibaüzenet.

A másik módja az új alkalom létrehozásának az, ha az *Alkalmak* fülön a lista valamelyik eleméhez tartozó *Duplikálás* gombra kattint az edző. Ez akkor praktikus, ha egy, már korábban elmentett alkalomhoz hasonlót szeretne készíteni, ugyanis ilyenkor a választott alkalom adatai automatikusan betöltődnek a beviteli mezőkbe.

13. ábra – Új alkalom hozzáadása az edzéshez

### 4.6.3 Törlés

A lejárt alkalmak halványabban jelennek meg, amelyeket nem töröl automatikusan a rendszer arra az esetre, ha az edző vissza szeretné nézni, ki vett részt a korábbi edzéseken. A lejárt jelentkezéseket sem törli automatikusan a rendszer, hátha a felhasználó soron szeretné követni korábbi részvételeit. Ha azonban nem szeretnék ezeket látni, az alkalom sorában lévő *Törlés* gomb megnyomásával eltávolíthatják azt véglegesen a listából. Abban az esetben, ha már vannak jelentkezők arra az alkalomra, akkor őket a rendszer automatikusan értesíti e-mailben a változásról. (10. ábra)

### 4.6.4 Jelentkezés

A jelentkezés minden típusú felhasználó esetében ugyanúgy valósul meg. Az edzések böngészésekor a kívánt edzés nevére kattint, megjelenik annak oldala. Itt táblázat formájában megtekinthetők az elérhető alkalmak és azok részletei. A *Jelentkezek!* gomb megnyomásával rögzíti a rendszer a jelentkezést. Egynél többször nem lehet jelentkezést leadni ugyanarra az alkalomra.

Város	Cím	Időpont	Ár (Ft)	Hossz (Perc)	Jelentkezők száma	
Csurgó	Virág utca 2. Sportközpont	2022.04.20. 12:30	1500	45	1/10	<a href="#">Jelentkezek!</a>

Sikeres jelentkezés!

14. ábra – Sikeres jelentkezés edzésre

### 4.6.5 Lemondás

A korábbi jelentkezések megtekintésekor lehet a lemondásokat végrehajtani. Itt a lemondani kívánt alkalom sorában található *Lemondás* gomb megnyomása után törlésre kerül a jelentkezés mind a kliens, mind az edző oldalán. Ezután, ha mégis részt szeretne venni az edzésen, újból jelentkeznie kell arra. (13. ábra)

## 4.7 Fejlesztői futtatási kézikönyv

A program fejlesztői változatának futtatásához a következő parancsokat kell kiadni a megfelelő helyen:

Az adatbázis létrehozásához a Visual Studio Package Manager Console-on két utasítást kell lefuttatnunk. Először a Default Project-ként a MoveYourBody.Service kiválasztása után `Add-Migration init -StartupProject MoveYourBody.Service` parancsot adjuk ki. Sikeres migráció után az `Update-Database -StartupProject MoveYourBody.Service`-szel jön létre az adatbázis. A lépések során fontos az adatbázis-szerver futása. Ezután elindítható a backend, mely a localhost:5000-en fut.

Frontend oldalon a Vizsga\frontend\MoveYourBody mappában parancssor segítségével adható ki az `npm install` parancs, mely a package.json alapján letölti a szükséges csomagokat a megfelelő verzióval. A frontend futtatásához az `ng serve` parancs kaidása után a localhost:4200-on érhető el a weblap.

## 4.8 Demo

A webalkalmazás demo-ja a következő URL-en érhető el:

<http://cluster.jedlik.eu/moveyourbody/home>

Készítettünk felhasználókat, amelyekkel kipróbálhatók a különböző funkciók. Ezekhez a bejelentkezési adatok a következők:

Típus	E-mail	Jelszó
Edző	jozsiedzo@email.com	jozsi
Kliens	evi@email.com	evi
Admin	admin@email.com	admin

## Irodalomjegyzék

ELTE IK, P. N. (2022. 03 15). *ELTE IK*. Forrás: Programozási Nyelvek és Fordítóprogramok:  
<http://nyelvek.inf.elte.hu/leirasok/JavaScript/index.php?chapter=27>

*Ilusionity*. (2022. 03 16). Forrás: Mik azok a JSON web tokenek? JWT hiteles bemutató:  
<https://hu.ilusionity.com/1157-what-are-json-web-tokens-jwt-auth-tutorial>

*Selenium*. (2022. 03 16). Forrás: About Selenium: <https://www.selenium.dev/>

*Swagger*. (2022. 03 16). Forrás: <https://swagger.io/>

*Wikipédia*. (2022. 03 15). Forrás: A függőség befecskendezése:  
[https://hu.wikipedia.org/wiki/A\\_f%C3%BCgg%C5%91s%C3%A9g\\_befecskendez%C3%A9se](https://hu.wikipedia.org/wiki/A_f%C3%BCgg%C5%91s%C3%A9g_befecskendez%C3%A9se)