Jedlik Ányos Gépipari és Informatikai Technikum és Kollégium

Move Your Body

dokumentáció

Készítették:

Magashegyi Ádám

Vadkerti Sára

Tartalomjegyzék

[1.1 A probléma és a megoldás 4](#_Toc98246137)

[1.2 Az ötletelés 4](#_Toc98246138)

[1.3 A végeredmény 4](#_Toc98246139)

[1.4 A jövő 5](#_Toc98246140)

[1.5 A csapatmunka 5](#_Toc98246141)

[2. A program 6](#_Toc98246142)

[2.1 Technikai részletek 6](#_Toc98246143)

[2.2 Az adatbázis felépítése 6](#_Toc98246144)

[2.2.1 Táblák: 7](#_Toc98246145)

[2.3 Az adatbázistól a megjelenítésig 9](#_Toc98246146)

[2.4 Autentikáció, biztonság 10](#_Toc98246147)

[2.4.1 A technológia 10](#_Toc98246148)

[2.4.2 Működése 10](#_Toc98246149)

[2.4.3 Jelszavak 10](#_Toc98246150)

[2.4.4 Jogosultságok 10](#_Toc98246151)

[2.5 API dokumentáció 11](#_Toc98246152)

[2.5.1 LocationController 11](#_Toc98246153)

[2.5.2 UserController 12](#_Toc98246154)

[2.5.3 TagController 14](#_Toc98246155)

[2.5.4 CategoryController 14](#_Toc98246156)

[2.5.5 TrainingController 15](#_Toc98246157)

[2.5.6 TrainingSessionController 19](#_Toc98246158)

[2.5.7 ApplicantController 20](#_Toc98246159)

[2.5.8 TagTrainingController 22](#_Toc98246160)

[2.5.9 AuthController 23](#_Toc98246161)

[2.6 Frontend dokumentáció 23](#_Toc98246162)

[2.6.1 Felépítése 24](#_Toc98246163)

[3. Tesztek 24](#_Toc98246164)

[3.1 Backend tesztek 24](#_Toc98246165)

[3.2 Frontend tesztek 24](#_Toc98246166)

[4. Felhasználói kézikönyv 25](#_Toc98246167)

[4.1 Navigációs menü 25](#_Toc98246168)

[4.1.1 Kezdőlap 25](#_Toc98246169)

[4.1.2 Kategóriák 25](#_Toc98246170)

[4.1.3 Tagek 25](#_Toc98246171)

[4.2 Regisztráció 25](#_Toc98246172)

[4.3 Bejelentkezés 26](#_Toc98246173)

[4.4 Profil beállításai 26](#_Toc98246174)

[4.5 Edzések 26](#_Toc98246175)

[4.5.1 Megjelenítés 26](#_Toc98246176)

[4.5.2 Létrehozás 26](#_Toc98246177)

[4.5.3 Szerkesztés 27](#_Toc98246178)

[4.5.4 Böngészés 27](#_Toc98246179)

[4.6 Alkalmak 27](#_Toc98246180)

[4.6.1 Megjelenítés 27](#_Toc98246181)

[4.6.2 Létrehozás 27](#_Toc98246182)

[4.6.3 Törlés 28](#_Toc98246183)

[4.6.4 Jelentkezés 28](#_Toc98246184)

[4.6.5 Lemondás 28](#_Toc98246185)

**1.** **Bevezetés**

## 1.1 A probléma és a megoldás

A Move Your Body 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 bárhonnan 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 szerettük 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 tageket 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 frontendtől 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.

# 2. A program

## A képen szöveg látható Automatikusan generált leírás2.1 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 taggel 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

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 lejárati idejét vagy a felhasználó jogait.

### 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 tokent.

**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 helyzet esetén könnyen elérhesse klienseit.

**Admin** típusú felhasználó:

Az Admin típusú felhasználó felvehet újabb kategóriákat, illetve tageket 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: /

**ListCounties()**

* Kérés típusa: HttpGet
* Paraméter: nincs
* Jogosultság: Trainer, Admin, User
* Feladat: megyék alapján csoportosítva listáz
* Válasz: megyenevekből 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, megyenév vagy városnév
* 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=

**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: /

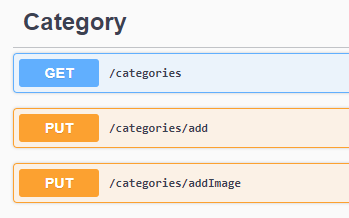
### 2.5.3 TagController

Elérési útvonal: /tag

**ListAll()**

* Kérés típusa: HttpGet
* Paraméter: nincs
* Jogosultság: nincs korlátozva
* Feladat: listázza az összes taget
* 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

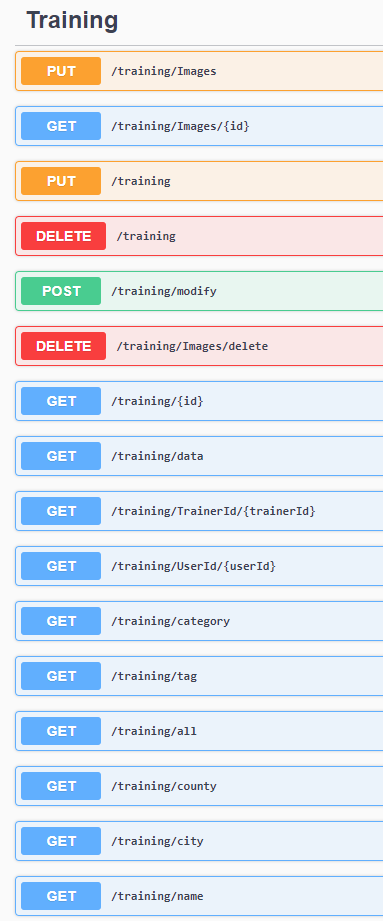
**AddImage(Category category)**

* 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 tartalmaz egy *TrainingImage* és egy *Image* típusú elemekből álló listát
* 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

**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=

**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 tartalmazegy *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 tartalmazegy *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=

**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(Tagraining tagTraining)**

* Kérés típusa: HttpPut
* Paraméter: *TagTraining* típusú osztálypéldány
* Jogosultság: Trainer, Admin
* Feladat: edzés létrehozásakor a kiválasztott tagek mentése
* Válasz: *TagTraining* típusú osztálypéldány
* Elérési útvonal: /

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

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

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

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

**Delete(TagTraining tagTraining)**

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

### 2.5.9 AuthController

Elérési útvonal: /registration/login

**Login(LoginModel login)**

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

## 2.6 Frontend dokumentáció

### 2.6.1 Felépítése

Frontendünk struktúrája főként a komponensekből, modellekből, a JWT-hez szükséges fájlokból 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

## A képen szöveg látható Automatikusan generált leírás3.1 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.

## A képen szöveg látható Automatikusan generált leírás3.2 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 funkcióit ellenőrzi.

# 4. Felhasználói kézikönyv

## 4.1 Navigációs menü

### 4.1.1 Kezdőlap

A weboldal megnyitásakor a *Kezdőlap* jelenik meg. Az oldal tartalmaz linkeket a *Regisztrációhoz*, a *Bejelentkezéshez* és a *Kategóriákhoz* is. Ez bejelentkezés nélkül is megtekinthető.

### 4.1.2 Kategóriák

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

### 4.1.3 Tagek

A navigációs menü *Tagek* feliratú legördülő menüpontja melletti nyílra kattintva jelenik meg az összes tag. A tag kiválasztásával az *Edzések* oldal ez alapján történő szűréssel kerül megjelenítésre.

## 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 automatikusan kijelentkezik a rendszer a felhasználó biztonsága érdekében.

## 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.

## 4.5 Edzések

### 4.5.1 Megjelenítés

Böngészéskor az edzés nevére kattintva megjelenik annak külön oldala. Itt látható az *edzés neve, rövid leírása, kategóriája*, a hozzá tartozó *tagek* és az *alkalmak* (4.7.4). 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.

### 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 edzéskártyákon található kategóranévre kattint, akkor az adott kategóriához tartozó edzések jelennek meg. A tagekre kattintva szintén meg lehet tekinteni az összes olyan edzést, amely a választott taggel rendelkezik. Az edző nevére nyomva az általa létrehozott edzések kerülnek kilistázásra.

## 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.

**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.

### 4.6.2 Létrehozás

Edzőként új alkalom az *Edzéseim* oldalon az edzéskártyán 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.

### 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.

### 4.6.4 Jelentkezés

A jelentkezés minden típusú felhasználó esetében ugyanúgy valósul meg. Az edzések böngészésekor a kívánt edzés nevére kattint, megjelenik annak oldala (4.6.1). 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.

### 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.

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

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