

Tartalomjegyzék

1.	Felhasználói dokumentáció	2
1.1	A program általános specifikációja	2
1.2	Rendszerkövetelmény	4
1.2.1	Hardver követelmény	4
1.2.2	Szoftverkövetelmény	4
1.3	A program telepítése	5
1.4	A program használatának részletes leírása	8
2.	Fejlesztői dokumentáció	13
2.1	Témaválasztás indoklása	13
2.2	Az alkalmazott fejlesztői eszközök	14
2.3	Adatmodell leírása.....	15
2.4	Részletes feladat-specifikáció, algoritmusok.....	18
2.5	Tesztelési dokumentáció	22
2.6	Tovább fejlesztési lehetőségek	24
2.7	Irodalomjegyzék, forrásmegjelölés	25

1. Felhasználói dokumentáció

1.1 A program általános specifikációja

A program edzőterem összefoglaló weboldal, ami Laravel keretrendszerben dolgozik. Ahogy a web dinamikus fejlődését figyelembe vesszük, szinte elkerülhetetlen az MVC alapismeretek jelentősége, mivel a legtöbb keretrendszernek ez az alapja köztük a Laravelnek is. Az MVC alapja a megfelelő logikai szétbontás szempontjából hasznos, mivel elkülöníti a logikai, megjelenítési, illetve vezérlő részeket a programban. Ezt a PHP keretrendszert többnyire webes szerveroldali műveletek végrehajtásához készítették. A Laravel előnye, az egyszerű, kifejező szintaxist használ, az ismétlődő hosszú kódsorokat leegyszerűsíti, illetve RESTful routing (útvonalkezelés) segítségével meghatározhatjuk az URL címhez beírt elemek hova irányítsanak, valamint get, post, put stb. kérést hajt-e végre a címen keresztül a program.

A kinézet létrehozásához Materialize css stílus-al került kialakításra, amely egy front-end keretrendszer. Napjainkban elterjedt ezeknek a használata különböző felületek megvalósításához. A Materialize egy előre megírt, multifunkcionálisan alkalmazható eszközkészletet kínál, aminek a segítségével gyorsabban, átláthatóbban és hatékonyabban dolgozhatunk.

Adatbázis tárolás céljára az MS (Microsoft) SQL szerver áll rendelkezésünkre. Az adatbáziskezelők piacán jelentős részesedéssel bír a MS SQL, részben technológiai megoldásai, részben más népszerű Microsoft rendszerekkel való kompatibilitása miatt. Nagyobb biztonságot képes nyújtani emellett, mint az ingyenesen használható MYSQL ingyenes verziói, azonban költségesebb egy cégnek. A program írása során figyelni kell arra, hogy egy cég milyen SQL rendszert használ, mivel a nyelv egyes szintaktikákban különbözhet. A Laravel migrációs állományok segítségével azonban könnyedén képesek vagyunk felvinni MS SQL adatbázisba is a teljes adatbázis, valamint MYSQL -re vagy SQL Lite-ra is. A program igazodik az adatbázis típusához, amit megadtunk neki, és így kezeli is a lekérdezéseket, feltöltéseket, módosításokat, törléseket.

Az edzőterem program lényege, hogy képesek vagyunk egy vagy több céget, különböző címekkel ellátva felvinni. Feltételezzük, hogy csak azok a cégek viszik fel adataikat és minimum egy telephelyüket, ahol különböző termekben órát lehet tartani. A cég felvételénél meg kell adni a bankszámlaszámot, adószámot, regisztrációs számot, telefonszámot, a cég

típusát, nyelvet, annak címét és minimum egy termét. Ezt a cégfelvétel után határozhatjuk meg mennyi termet vihet fel. Ezt a felvételt csak az admin hajthatja végre. Ezen kívül az admin határozza meg kinek, milyen, jogosultsági köre lehet, valamint felhasználókat tud kezelni. Jogosultságokat képes létrehozni, valamint szerkeszteni, törölni. Ezen kívül a jogosultságokhoz tartozó szabályokat is képes létrehozni, amivel le tudja kezelni ki hova férhet hozzá. Ez hasznos lehet az esetleges tovább fejlesztés esetében is, ha egy új jogkörrel akarjuk bővíteni a programot.

Miután az admin felvitte a céget a program használója, aki órát szeretne tartani egy cégnél, beregisztáció után az admin kiosztja neki a szükséges jogokat, amivel hozzáférhet az edzőkre vonatkozó menüpontokhoz. Az edző bejelentkezés után, amikor megkapta már a szükséges hozzáféréseket a menüpontokhoz, létrehozhat órát valamelyik felvitt cégnél, és azon belül egy megjelölt teremben. A program lekezeli az esetleges ütközéseket, ha abban az időpontban, abban a teremben már valamelyik másik edző már jelezte, hogy órát tartana. Miután létrehozta az órát, az edző képes ezen szerkeszteni, törölni, valamint eltárolni kik azok, akik jelentkeztek az órára és részt is vettek rajta vagy esetleg nem jelentek meg. Az edző, illetve az admin ezen kívül képes más edzők óráira feliratkozni.

Ezután ha a felhasználó bejelentkezik, és belelép az „Aktuális óráink” menüpontra, azonban még nincs felvitt óra, amire fel tudna iratkozni, akkor a program jelzi, hogy „Nincs felvitt óra jelenleg”. Ha az edző felvitt már legalább egy órát, akkor a program megjeleníti az edző elérhetőségét, az óra időpontját, a címet és a termet ahol az óra lesz. Ezek az adatok mellett megjelenik egy „Feliratkozás” gomb, amivel jelzi, hogy részt venne az órán. Azonban ha már egy órára jelentkezett, akkor a gomb eltűnik és jelzi a felhasználónak, hogy oda már feliratkozott.

1.2 Rendszerkövetelmény

1.2.1 Hardver követelmény

PHP esetén, a párhuzamos oldalak lekérésekor, oldalanként külön folyamatok indulnak.

- Minden folyamat saját memória területtel rendelkezik
- Minden folyamat saját adatbázis kapcsolattal rendelkezik
- Az oldalak nincsenek egy közös alkalmazásba szervezve, különálló programrészek

A Program Képes egy gyengébb szervergépen.

Minimális követelmény:

RAM: 512 MB

Processzor: 1,2 Ghz Intel

1.2.2 Szoftverkövetelmény

Operációs rendszer:

BSD vagy Windows alapú szerver oldali rendszer, vagy Szerver oldali tárolós NAS rendszer

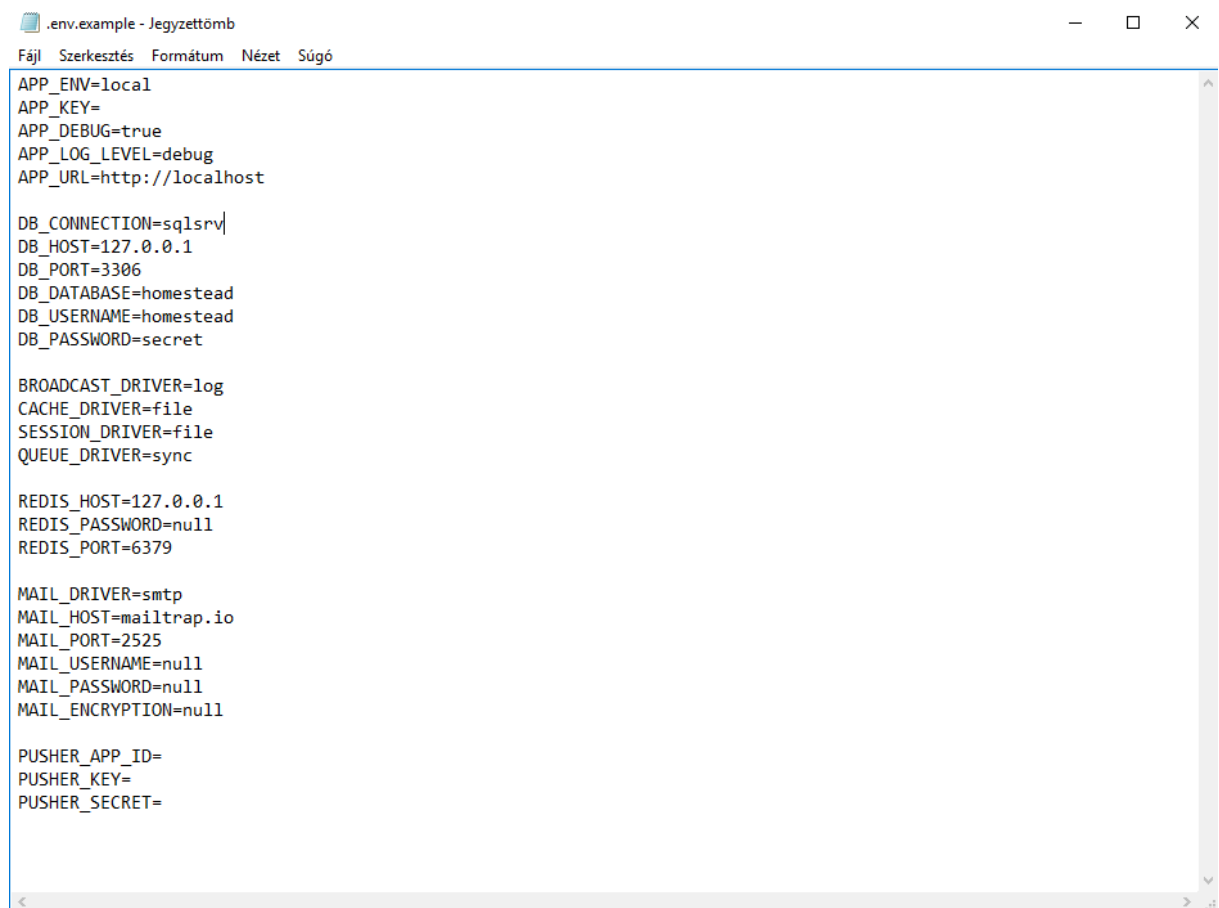
- Composer -> Laravel futtatásához szükséges program
- PHP >= 5.6.4-es verzió
- Apache
- MS SQL SERVER 2012

A Laravel web alkalmazás különlegessége, hogy beállíthatjuk, milyen adatbázisra szeretnénk kapcsolódni, és ezt ő felismeri és azt képes kezelni. Ezzel csatlakoztathatjuk MySQL –hez vagy SQL Lite –hoz is akár.

1.3 A program telepítése

Telepítés XAMPP local szerverre, Windows operációs rendszerre:

- Telepítsük a composer-t
 - Telepítésnél a telepítő mindent beállít, hozzá igazít megfelelően a működéséhez.
- Létrehozunk SQL szerver-ben a az adatbázist, amiben tárolni akarjuk a táblákat.
- A mappában lévő .env.example fájlt megnyitjuk jegyzettömbben és kitöljük az adatbázis csatlakozáshoz szükséges részt.
 1. **DB_CONNECTION**-hoz az SQL szerver fajtáját (MYSQL, MS SQL, SQL Lite)
 2. **DB_HOST**-hoz megadjuk az adatbázishoz csatlakozáshoz szükséges IP címet
 3. **DB_PORT**-hoz megadjuk amelyik porton csatlakozunk az adatbázishoz
 4. **DB_DATABASE**-hez beírjuk az adatbázis nevét, amit használni akarunk
 5. **DB_USERNAME**-hez az SQL szerver felhasználó nevét, amivel hozzáfér az adatbázishoz
 6. **DB_PASSWORD**-hoz az SQL szerver felhasználó jelszavát



```
.env.example - Jegyzettömb
Fájl Szerkesztés Formátum Nézet Súgó

APP_ENV=local
APP_KEY=
APP_DEBUG=true
APP_LOG_LEVEL=debug
APP_URL=http://localhost

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=homestead
DB_USERNAME=homestead
DB_PASSWORD=secret

BROADCAST_DRIVER=log
CACHE_DRIVER=file
SESSION_DRIVER=file
QUEUE_DRIVER=sync

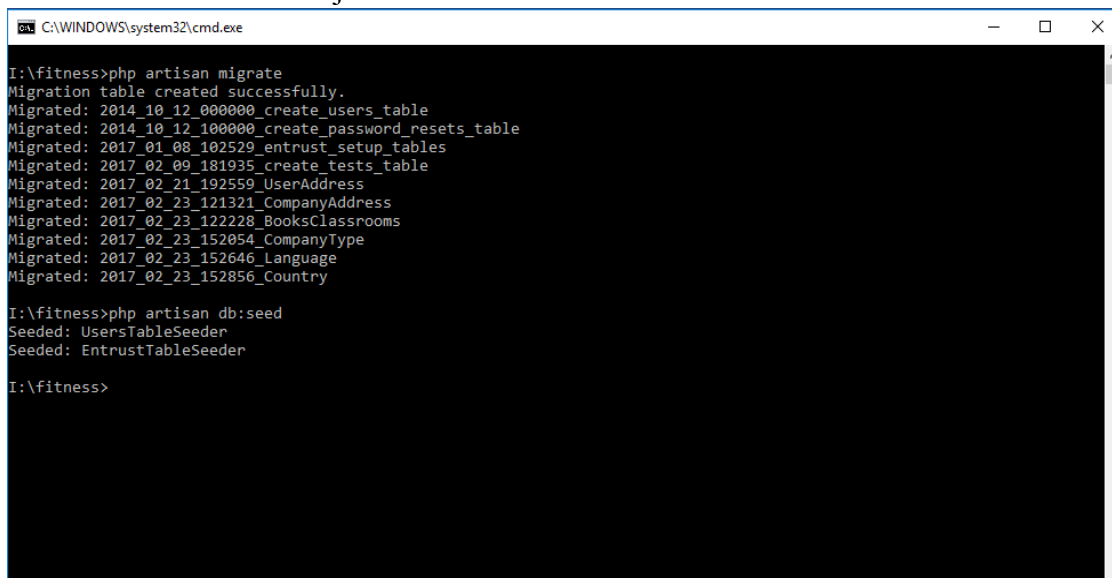
REDIS_HOST=127.0.0.1
REDIS_PASSWORD=null
REDIS_PORT=6379

MAIL_DRIVER=smtp
MAIL_HOST=mailtrap.io
MAIL_PORT=2525
MAIL_USERNAME=null
MAIL_PASSWORD=null
MAIL_ENCRYPTION=null

PUSHER_APP_ID=
PUSHER_KEY=
PUSHER_SECRET=
```

- Mentjük .env formátumban
- A weboldal mappájából Shift+Jobb egérgomb majd ott kiválasztjuk a Parancsablak nyitása menüpontot
- Parancssorba begépeljük a következő parancsot majd lefuttatjuk

- Először a php artisan migrate parancsot -> ezzel létrejönnek az adatbázisban a táblák
- Ezután a php artisan db:seed parancsot-> ezzel létrejönnek az alapvető jogosultságok, valamint 3 jogkör szerinti felhasználó:
 - Admin:
 - email: admin@admin.com
 - jelszó: admin
 - Edző:
 - email: editor@editor.com
 - jelszó: editor
 - Felhasználó:
 - email: user@user.com
 - jelszó: user



```
C:\WINDOWS\system32\cmd.exe

I:\fitness>php artisan migrate
Migration table created successfully.
Migrated: 2014_10_12_000000_create_users_table
Migrated: 2014_10_12_100000_create_password_resets_table
Migrated: 2017_01_08_102529_entrust_setup_tables
Migrated: 2017_02_09_181935_create_tests_table
Migrated: 2017_02_21_192559_UserAddress
Migrated: 2017_02_23_121321_CompanyAddress
Migrated: 2017_02_23_122228_BooksClassrooms
Migrated: 2017_02_23_152054_CompanyType
Migrated: 2017_02_23_152646_Language
Migrated: 2017_02_23_152856_Country

I:\fitness>php artisan db:seed
Seeded: UsersTableSeeder
Seeded: EntrustTableSeeder

I:\fitness>
```

- A XAMPP mappáján belül kikeressük az **apache\conf\extra** mappát, ahol megnyitjuk jegyzetömbbe megnyitjuk a httpd-vhosts.conf állományt és begépeljük a legaljára a következő parancsokat:



```

##DocumentRoot "C:/xampp/htdocs/dummy-host2.example.com"
##ServerName dummy-host2.example.com
##ErrorLog "logs/dummy-host2.example.com-error.log"
##CustomLog "logs/dummy-host2.example.com-access.log" common
##</VirtualHost>

<VirtualHost *:80>
DocumentRoot "C:/xampp/htdocs/fitness/public"
ServerName edzoterem.dev
ServerAlias www.edzoterem.dev
<Directory "C:/xampp/htdocs/fitness/public">
Order allow,deny
Allow from all
</Directory>
</VirtualHost>

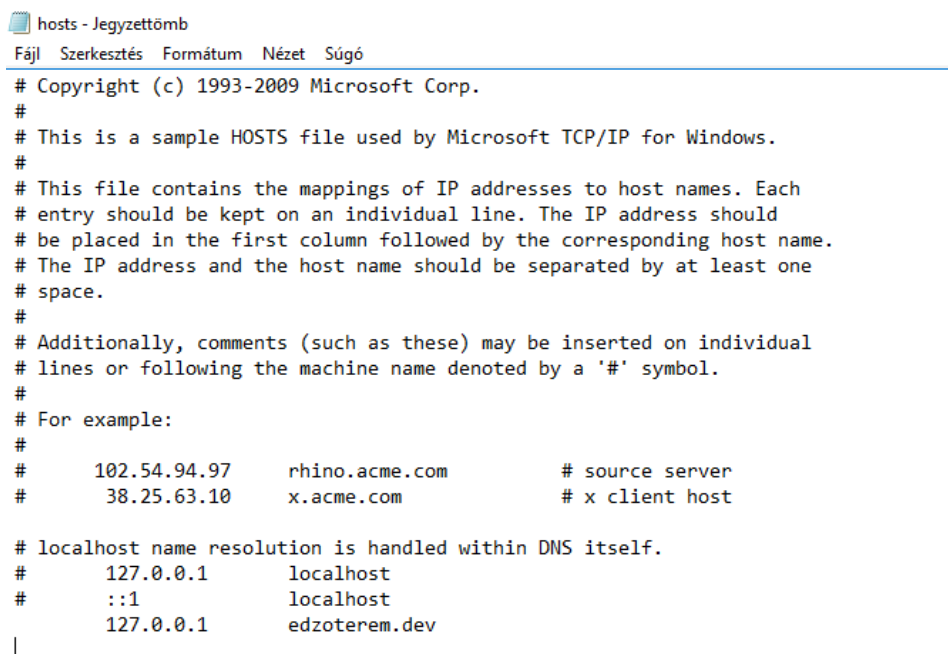
<VirtualHost *:80>

```

- A C:\Windows\System32\drivers~etc mappában lévő host fájlt adminisztrátor módban megnyitjuk jegyzetömbben és beleírjuk a Xampp Virtual Host elérési címét ->

Servername

A következőképpen valósítjuk ezt meg:



```

# Copyright (c) 1993-2009 Microsoft Corp.
#
# This is a sample HOSTS file used by Microsoft TCP/IP for Windows.
#
# This file contains the mappings of IP addresses to host names. Each
# entry should be kept on an individual line. The IP address should
# be placed in the first column followed by the corresponding host name.
# The IP address and the host name should be separated by at least one
# space.
#
# Additionally, comments (such as these) may be inserted on individual
# lines or following the machine name denoted by a '#' symbol.
#
# For example:
#
#       102.54.94.97       rhino.acme.com   # source server
#       38.25.63.10      x.acme.com        # x client host

# localhost name resolution is handled within DNS itself.
#       127.0.0.1         localhost
#       ::1               localhost
#       127.0.0.1         edzoterem.dev

```

- Ezután a XAMPP -ot újraindítjuk és a program eléréséhez a böngészőbe beírjuk az URL részhez a hozzá tartozó .dev megnevezést (pl.:edzoterem.dev)

1.4 A program használatának részletes leírása

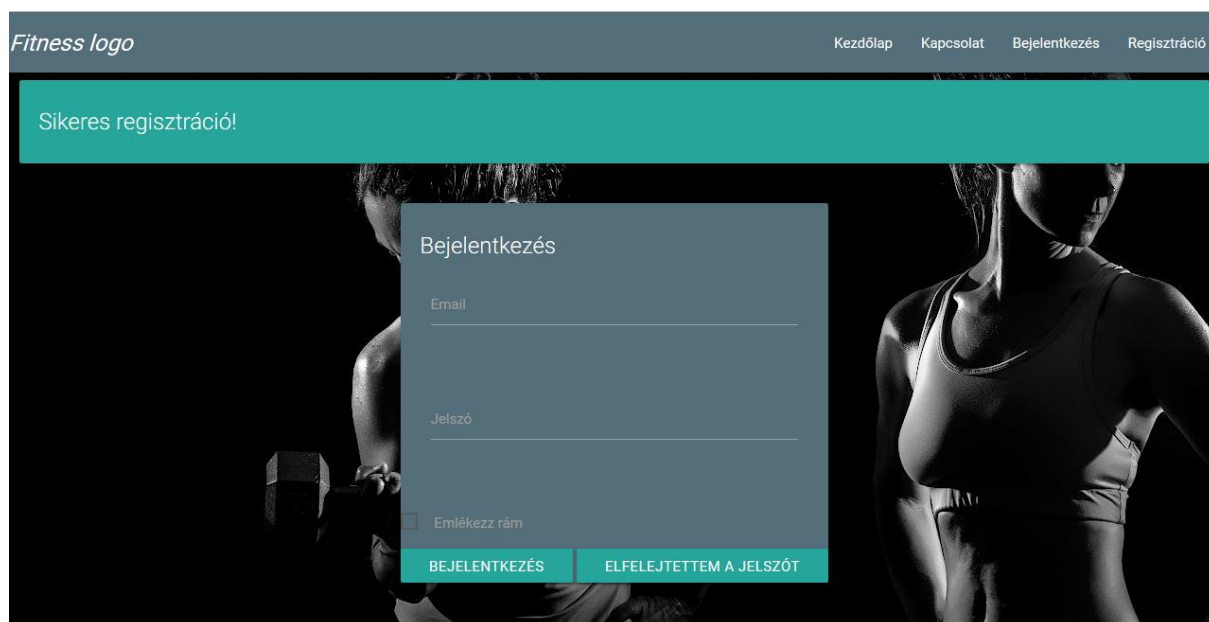
A program telepítése után a következő képernyő fogad minket:

A program felső sora tartalmazza a menüt, ahol először a „Regisztráció” gombbal tud regisztrálni, valamint a „Bejelentkezéssel” belépni






Először ha regisztrálunk az oldalra odakell figyelni, hogy minden mező megfelelően legyen kitöltve. A regisztrációs mezőben.

Ezek alapján tájékozódhat minden felületen, bármilyen kitöltésnél, hogy milyen mezőt töltött ki rosszul. Amennyiben a regisztráció sikeresen lezajlott, a program átirányít a bejelentkezésre, és üzenettel jelzi a regisztráció sikerességét.



A program 3 fajta jogosultságot különít el így alapértelmezettként a belevan építve van egy admin, edző, illetve user felhasználó. Náluk a menüpontjai dinamikusan változik a hozzá tartozó jogosultság alapján

Menüpontok különböző jogosultságoknál:

 admin@admin.com	 editor@editor.com	 user@user.com
Jogok és szabályok	Termek	Aktuális órák
Jogosultságok	Az ön órái	Adatlap
Szabályok	Aktuális órák	Logout
Felhasználók kezelése	Adatlap	
Cégek	Logout	
Aktuális órák		
Adatlap		
Logout		

Admin:

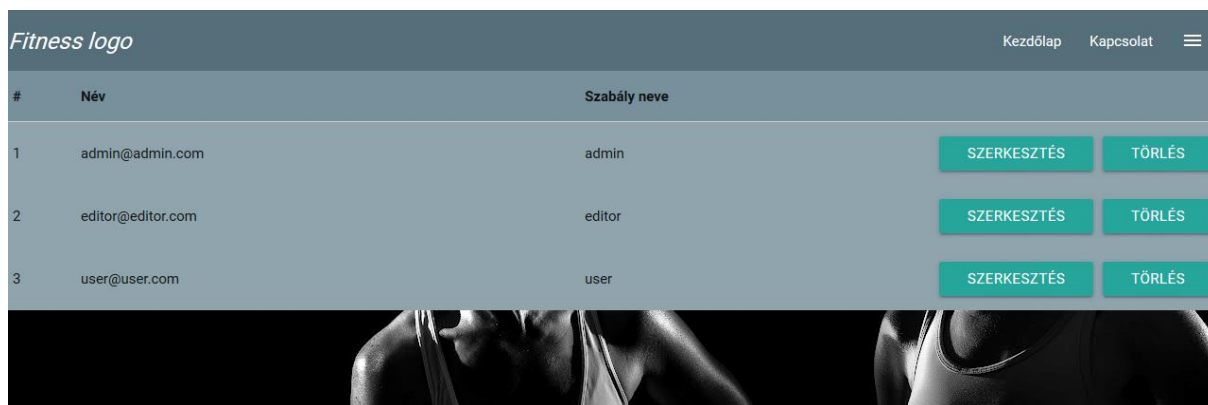
Jogok és szabályok: Az admin itt állíthatja be egy táblázatba foglalt jogok és szabályok összesítéssel a jogokat és ahhoz tartozó szabályokat.

Jogosultságok: Itt tudja módosítani, szerkeszteni, törölni a felhasználók jogait.

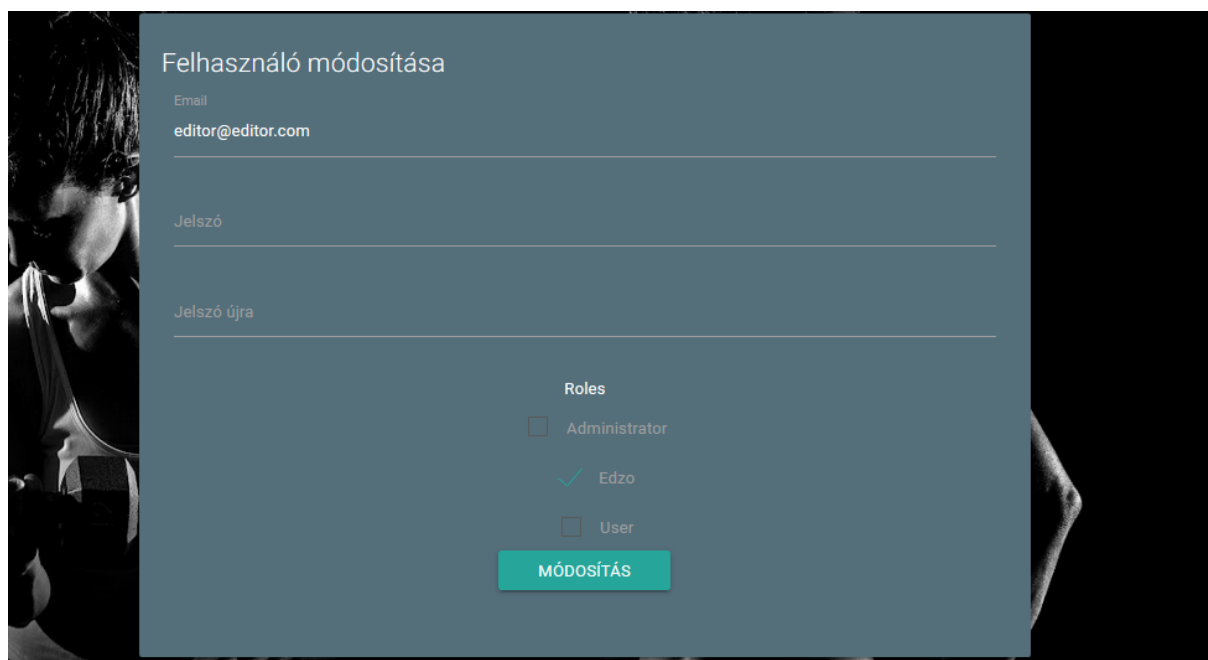
Szabályok: Ebben a menüpontban tudja beállítani, hogy a felhasználó melyik menüpontokhoz férhessen hozzá.

Ezekon a felsorolt menüpontok csak tovább fejlesztés esetén érdemes változtatni.

Felhasználók kezelése: Felhasználók módosítása, törlése, hozzá tartozó jogok kiosztására szolgáló menüpont. Használata során beállíthatunk regisztrált felhasználóknak jogosultságokat.



#	Név	Szabály neve		
1	admin@admin.com	admin	SZERKESZTÉS	TÖRLÉS
2	editor@editor.com	editor	SZERKESZTÉS	TÖRLÉS
3	user@user.com	user	SZERKESZTÉS	TÖRLÉS



Felhasználó módosítása

Email
editor@editor.com

Jelszó

Jelszó újra

Roles

☐ Administrator

☒ Edzo

☐ User

MÓDOSÍTÁS

Cégek: A menüpontra kattintva megjelenik egy táblázatban az összes eddig felvitt cég, annak szükséges adatai, valamint a cégekhez „Szerkesztés”, „Törlés”, „Adatok” gomb. Az adatok gombra kattintva megjelenik a kiválasztott cég minden terme, amit szerkeszthetünk, hozzáadhatunk, törölhetünk. Ezen kívül a menüpontban vihetünk fel új céget majd, ha

The screenshot shows a web application interface for adding a new room. At the top, there's a header with 'Fitness logo', 'Kezdőlap', 'Kapcsolat', and a menu icon. Below the header, a modal form titled 'Új terem felvitele' is displayed. The form has two input fields: 'Teremnév' (Room name) and 'Szabad hely' (Free space) with a unit selector icon. A green 'FELVITEL' (Add) button is at the bottom of the form. Below the form, there's a table with columns: 'Cégnév', 'Regisztrációs szám', 'Bankszámlaszám', 'Adószám', 'Address id', 'Város', 'Cím', and 'Zip'. The table contains one row of data for 'Minta' company. Below the table, there's another table with columns: 'Teremnév', 'Férőhely', 'SZERKESZTÉS', and 'TÖRLÉS'. It contains two rows of data for rooms 1 and 2.

Cégnév	Regisztrációs szám	Bankszámlaszám	Adószám	Address id	Város	Cím	Zip
Minta	32234112	82737489323	747383298	29	Budapest	Kossuth utca 32	1811

Teremnév	Férőhely	SZERKESZTÉS	TÖRLÉS
1	10	SZERKESZTÉS	TÖRLÉS
2	14	SZERKESZTÉS	TÖRLÉS

sikeresen hajtottuk végre akkor ezt üzenettel jelzi.

Edző:

Termek: Ebben a menüpontban találja meg az Edző a felvitt cégek termeit, illetve azok adatait. A termék alatt szereplő „Óra felvitele” gombbal.

The screenshot shows the 'Felvitt termek' (Added products) section of the web application. The header is the same as the previous screenshot. Below the header, there's a section titled 'Felvitt termek'. It contains two product cards for the 'Minta' company. Each card displays the company name, address, and room information. A green 'ÓRA FELVITEL' (Add hour) button is at the bottom of each card.

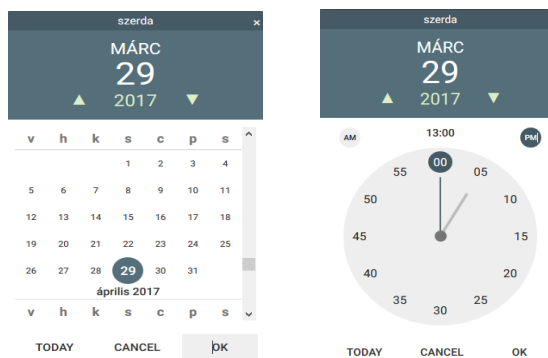
Cég: Minta
Város: Budapest
Cím: Kossuth utca 32
Teremnév: 1
Férőhely: 10

ÓRA FELVITEL

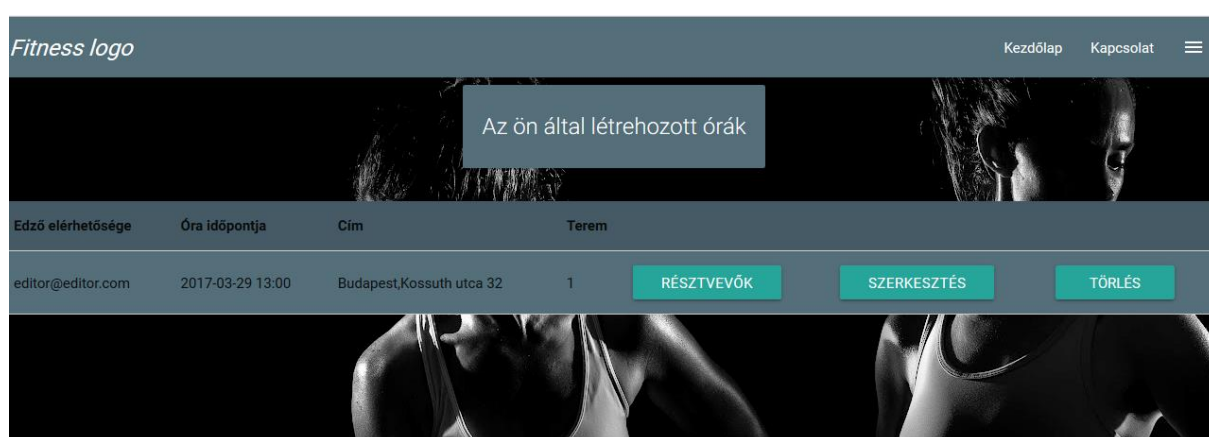
Cég: Minta
Város: Budapest
Cím: Kossuth utca 32
Teremnév: 2
Férőhely: 14

ÓRA FELVITEL

Ezután Dátum és idő megadásával hozhat létre órát abban a teremben. Először kiválasztja az évet és a napot, majd átdob, egy órára ahol az időt lehet beállítani.

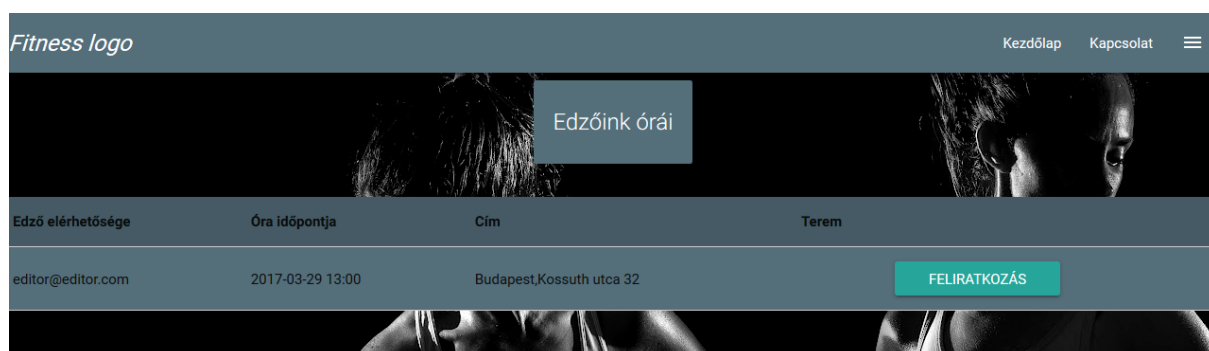


Az ön órái: Itt a bejelentkezett edző által felvitt órák jelennek meg, amit szerkeszthet, törölhet, illetve résztvevőket ellenőrizhet. A „Résztvevők ellenőrzése” gombra kattintva kilistázza a program az órára jelentkezőket, és ott bejelölheti, hogy megjelent-e vagy sem.



Felhasználó:

Aktuális óráink: Minden felhasználó jognál megjelenik, mert minden felhasználó jelentkezhet órára. Edzőknél ilyenkor a saját órái nem kerülnek ki listázásra.



Logout: Kijelentkezés a programból

2. Fejlesztői dokumentáció

2.1 Témaválasztás indoklása

A szakdolgozatom témája egy edzőterem weboldala. Azért ezt a témát választottam, mert a téma messze áll tőlem és a megvalósításához átkell látni egy edzőterem lényegét, ahol csoportos óra keretein belül meghirdetett órára jelentkezik, mielőtt azon részt vesz. A témában érdekes problémának tűnt az óra kezelési rendszer, valamint feltételezés kéne, ha a weboldalon több cég is szeretné, ha az egyik telephelyükre, ahol órát tartanának meghirdetni és edző által órákat tartani.

A létrehozásában egy kis erősségként egy keretrendszert választottam, mert hasonló MVC elvekkel működő rendszerrel dolgozik és mert a dinamikusan fejlődő webes világban lassan elkerülhetetlen egy-egy keretrendszer ismerete a webalkalmazás fejlesztés területén. A témában a jogosultság kezelése különösen érdekesnek tűnt, valamint az összetettebb adatbázis megépítése.

A kinézeti megvalósítás is különösen érdekesnek hangzott, hogy mi lehet az a kinézet amire egy edzőterem weboldalán a felhasználó szívesen műveleteket végezne, valamint egy olyan kinézet megvalósítását akartam kialakítani, amivel egy bizonyos szinten az oldal látogatója szívesen részt venne a cégek által felkínált órákon.

A témának az adatbázisának megvalósításakor többféle koncepciót és logikai problémát láttam, amiket többszempontról is átkellett gondolni, hogy a működésben ne akadályozza a felhasználót, ne legyenek ütközések, ne legyen redundáns az adatbázis.

2.2 Az alkalmazott fejlesztői eszközök

A program elkészítéséhez PHP szerver oldali script nyelvet használtam ezen belül is Laravel keretrendszerben dolgoztam ki a program felépítését.

Sublime Text:

Ennek a programnak a segítségével dolgoztam a PHP kódomon. Az IDE különlegessége, hogy több programozási nyelvet is felismer. Egyaránt kezeli a HTML, Javascript, PHP nyelveket, amelyeket a szakdolgozat kidolgozása során használtam. Tartalmaz beépített konzolt, amiben különböző parancssori parancsokat tudunk lefuttatni, ami a szakdolgozatomban hasznosnak bizonyult, mivel a Laravel beépített parancsait, amit lefuttatunk például Model vagy Controller létrehozásánál az alapértelmezett függvényeivel együtt létre tudjuk hozni egy rövid parancsal.

Előnye: Kis erőforrás igénye van és több kiegészítővel is felszerelhető, több fajtaprogramozási nyelvet is használható.

Hátránya: Szintaktikai hibákat nem észleli, a függvények közötti keresést megnehezíti azzal, hogy kilistázza az összes osztályt, amikben megtalálható a függvény.

Star UML:

Diagramkészítő program. Adatbázis tervezés során, valamint osztály alapú programtervezés során használjuk, amivel vizuálisan építhetjük fel a programunkat. Az elkészült munkákat különböző formátumba exportálhatjuk, illetve menthetjük későbbi munkához.

Előnye: Kisebb gépigényen is tökéletesen működik, egyszerű benne a diagram létrehozása, ingyenes program.

MS SQL SERVER 2012:

A Microsoft Hivatalos SQL SERVER programja, amit ingyenesen letölthető a Microsoft honlapjáról. Egy jól védhető adatbázis kezelő rendszer, amelyet használhatunk távoli szerver adatbázis csatlakoztatására, valamint otthoni virtuális környezetet is készíthetünk benne az adatbázis tesztelésére. Szemléltetésre az adatbázis táblákat létrehozza összezsolt állapotban ezzel könnyítve az adatbázis átláthatóságát. Létrehozhatunk adatbázist, táblákat valamint szerkesztéseket, feltöltéseket, törléseket tudunk és egyéb Query parancsokat végrehajtani, amit a programozás során felhasználtam.

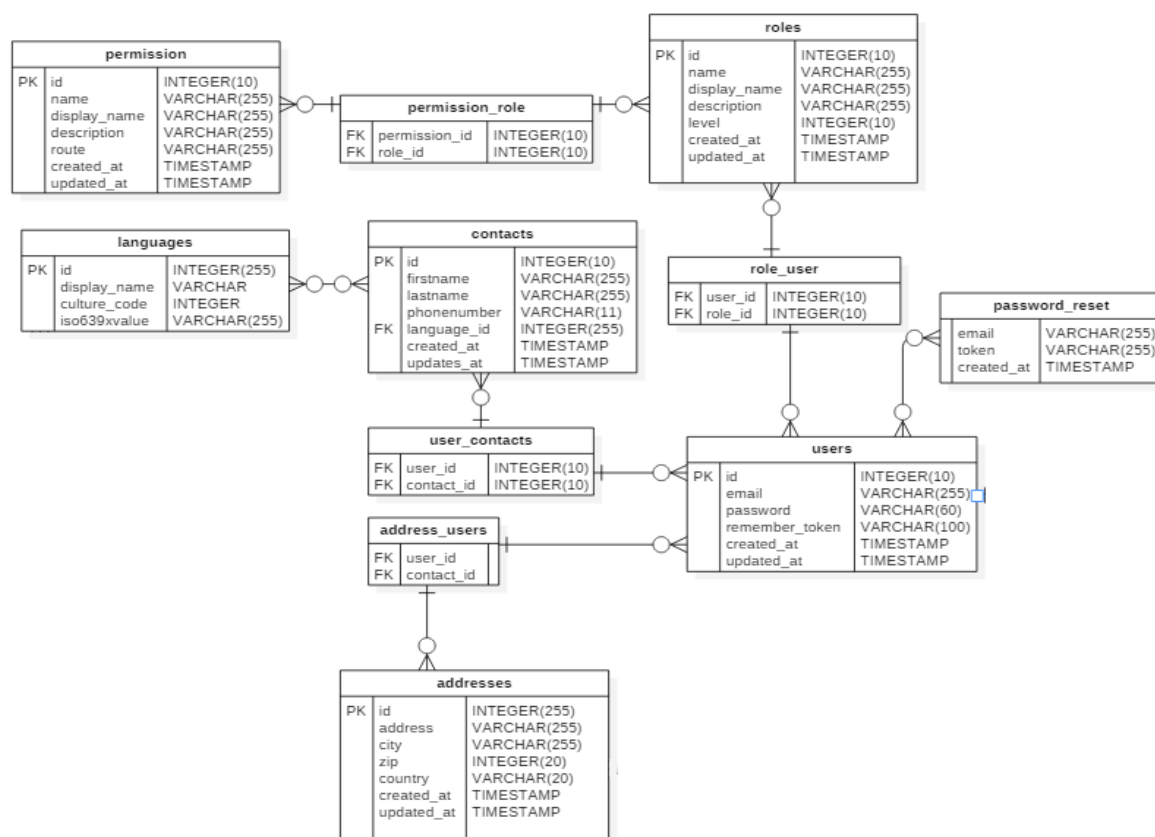
Előnye: Jól védhető, átlátható rendszerrel rendelkezik.

Hátrány: Nagy erőforrásigény

2.3 Adatmodell leírása

Adatbázis kialakításánál először a jogosultság kezelés és a felhasználó adatok tárolása, illetve kialakítása volt az elsődleges, mert, ha egy felhasználó sem létezik, az adatbázisban nem történhet változás.

A felhasználókhoz kötött jogosultságok szabályozzák, hogy melyik felhasználó hol törölhet, módosíthat, új elemet vihet fel az adatbázisba. Az admin képes a felhasználók kezelésére, jogaik kiosztására, új jelszó megadására, cégek, termék felvitelére, törlésére, szerkesztésére. Az edzo képes termekbe órákat felvinni, szerkeszteni a dátumot, törölni a saját óráját, eltárolni a megjelent embereket. A felhasználó tud órára jelentkezni. Tárolásnál ügyelni kell, hogy a felhasználóknak milyen adatait szükséges eltárolni amivel beazonosíthatjuk őket. Felhasználók és a hozzá tartozó kapcsolások:

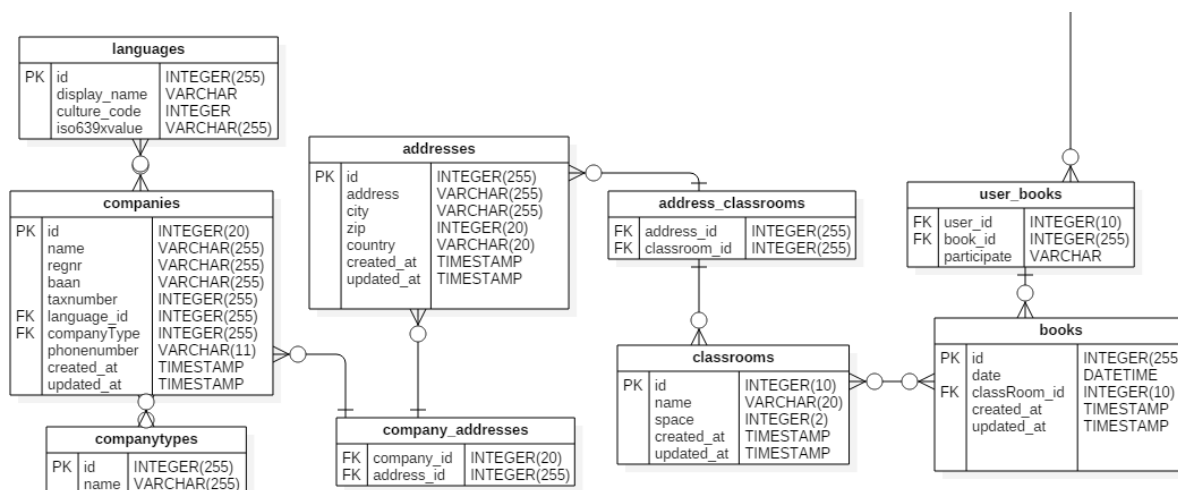


Létre kell hozni egy felhasználók táblát, ami a diagramon a „users” tábla, amiben bejelentkezési adatok kerülnek eltárolásra. Ezt hozzá kell kapcsolni a jogosultságokhoz, vagyis „role” táblához, amiből kiderül mit hajthat végre a felhasználó. Ezt egy kapcsolótáblán keresztül kapcsoljuk, mert ugyanazt a jogot több felhasználó is megkaphatja, ezért ezt „id” alapján kapcsoljuk egymáshoz. A „users” emellett összekell kapcsolni egy felhasználó adatok

táblához, ami a „contacts” tábla. Itt el kell tárolni a felhasználó szükséges adatait nevét, telefonszámát, nyelvét. A nyelvet a nyelv vagyis „languages” táblából veszi ki, amiben előre eltárolt nyelvek közül választhat a felhasználó. A felhasználóknál ezenkívül meg kell határozni a lakhelyét. Ezért a „users” táblát hozzá kell kapcsolni az „addresses” táblához, amiben id alapján különíti el a program hozzá tartozó címet. Az „addresses” táblában szét kell tagolni a lakcímhez szükséges mezőket, ezért külön fel kell venni a címet, irányítószámot, várost, országot.

A jogosultság meghatározást, amivel a program dolgozik, a „permission”, „roles” és a hozzá tartozó kapcsolótáblákkal vannak meghatározva. Ezt csak az admin módosíthatja, törölheti a programon belül. Ezek a táblák azonban nem igényelnek beavatkozást, csak tovább fejlesztés esetén. Az admin ezenkívül cégek adatait képes kezelni.

A Cég , terem, foglalások és azokhoz kapcsolódó táblák:

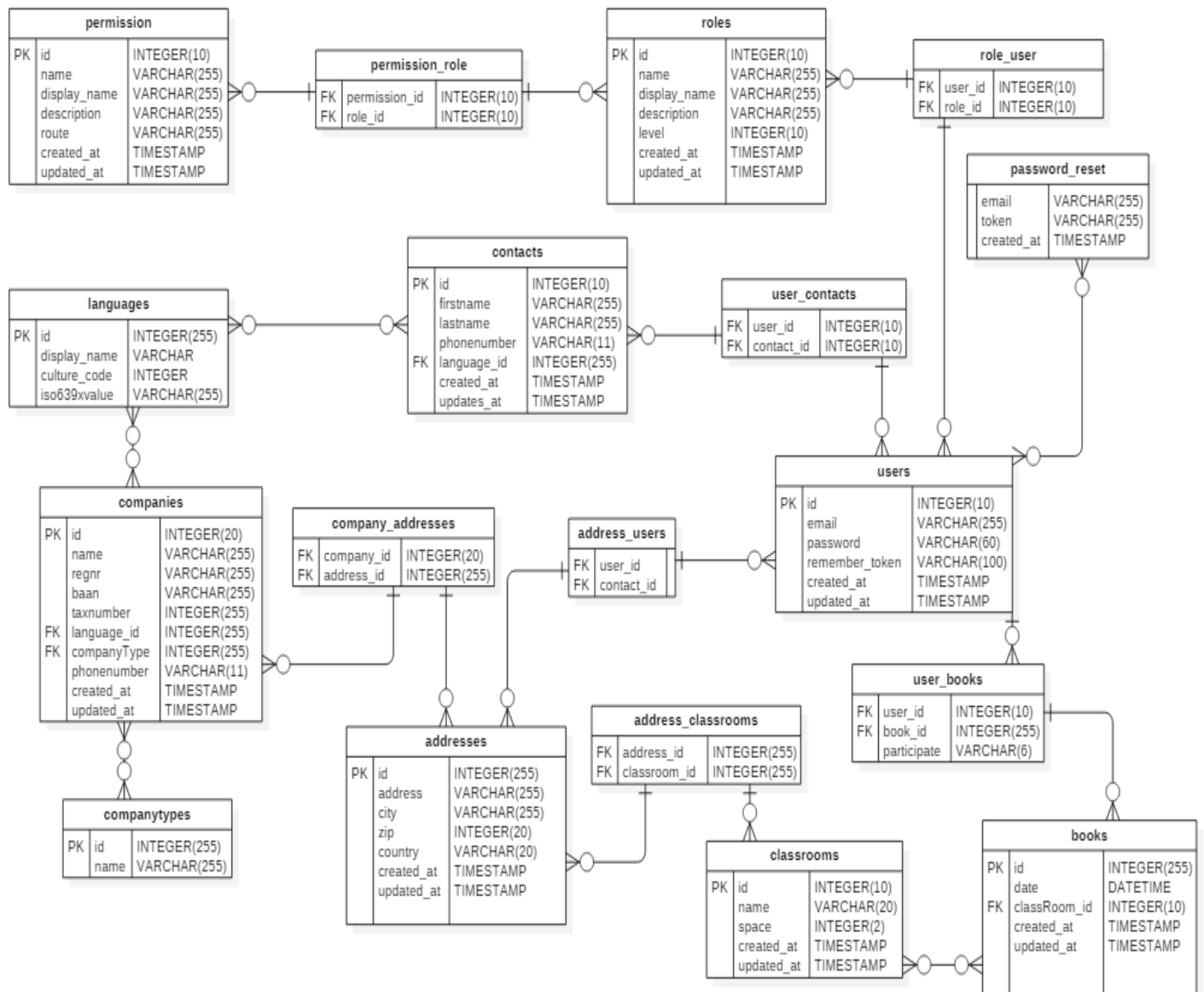


A „companies” táblában tárolja a céghez szükséges adatok. Ebben, a táblában tárolja a cég nevét, regisztrációs számát, adószámát, bankszámlaszámát, alapértelmezett nyelvét és cég típusát. A cég típusát hasonlóképp, mint a nyelvet egy előre felvitt „companyType” táblából veszi ki a felhasználó, ezzel meghatározva a cég típusát. A cég felvitelhez ezenkívül kapcsolódnia kell az „addresses” táblához, mert egy telephely minimum van ahol az órát lehet tartani és ahol órát lehet tartani ott teremnek is kell lennie, ezért 1 terem fel kell vinni cég felvitelnél minimum. Ezt a cég címéhez kapcsolt „classrooms” táblában tároljuk, ahol meghatározzuk a terem nevét, férőhelyét.

Ha terem, és cég létrehozása megvalósult, akkor a teremhez kapcsolódó „books” táblában tárolhat az edző órákat. Itt tárolja az adatbázis az időpontot és a terem id-jét. Miután létrejött az óra a felhasználó képes az órára jelentkezni, ami a „user_books” táblában tárol az

adattábas. Itt a felhasználó és a foglalás id-jét tárolja az adattábas, amit a „users” és „books” táblából emel ki. Itt tárolja továbbá a megjelenést is

A teljes adattábas ER diagramja:



2.4 Részletes feladat-specifikáció, algoritmusok

Bejelentkezési rendszer kiépítéséhez a laravel beépített bejelentkezés végrehajtó modulját használtam. A regisztrálás során a egy felhasználó, ha elküldi az adatokat, át a ContactController kapja meg elsőként a kérést. A programrész a következőképp néz ki:

```
public function store(Request $request)
{
    $validation = $this->validationReg($request->all());
    if ($validation->fails())
        return redirect('/regist')
            ->withErrors($validation->errors()->all())
            ->withInput();

    $user = $this->createUser($request->only('email','password'));
    $contact = $this->createContact($request->only('firstname','lastname','language_id','
        phonenumber'));
    $address = $this->createAddress($request->only('address','city','zip','country'));
    UserContact::create([
        'user_id'          => $user->id,
        'contact_id'       => $contact->id,
    ]);
    AddressUser::create([
        'address_id'       => $address->id,
        'user_id'          => $user->id,
    ]);

    $request->session()->reflash();
    $request->session()->flash('success', 'Sikeres regisztráció!');
    return redirect()->route('login');
}

public function createUser($dat){
    return $contact = User::create($dat);
}

public function createContact($dat){
    return $contact = Contact::create($dat);
}

public function createAddress($dat){
    return $contact = Address::create($dat);
}
```

A függvény kap egy request értéket, amelyet utána vizsgál azzal, hogy átadja a teljes \$request->all() értéket a validationReg függvénynek, amelyben végrehajtódik a validációk. A Laravel ezt a megadott \$rule-k alapján levizsgálja, majd a hozzá tartozó \$message-eket a Validator osztály make függvényének adja át, ami a Laravel beépített osztályában található. Ez vizsgálja, hogy minden \$rule teljesült-e amennyiben nem akkor visszatér a \$message-el amit nyelvi fájlból olvas ki. Ezzel elkerülhető az esetleges kódolási hibák. Ha a regisztráció során valamilyen hiba lép fel a rossz kitöltés miatt akkor visszairányít a hibaüzenetekkel a regisztrációs felületre. Ha a validáció sikeresen végrehajtott, akkor a user táblába feltöltendő adatokat, a \$user változóba tesszük, ahol ezután lefut a createUser függvény, amiben a \$request->only('email','password') vagyis a kapott \$requestből csak azoknak a mezőknek az értékét küldje tovább a createUser függvénybe, amiket kijelöltünk benne. A

createUser függvény továbbítja a hozzá tartozó adatokat a User modelbe, ahol create-el létrehozza az adatbázisban. Ha ez sikeresen végrehajtott, akkor a kapott id-jüket beletöltjük a kapcsolótáblákba. Ezt a UserContact::create függvényben hajtjuk végre, ahol egy tömbbe helyezzük a mezők neveit, és az előzőleg létrehozott id-ket belerendeljük.

A create függvényeknél a \$contact megteveszthető lehet, azonban funkciója nincs, továbbfejlesztés szempontjából nem hátrány, ha megvan adva egy változóba, amivel további műveletek hajthatók végre a fejlesztés során.

Ha végrehajtott az összes feltöltés, akkor egy session() függvény-t írunk, ami a session-öket üríti, ezután behelyezzük a success –be a „Sikeres regisztráció” feliratot, ami a sikeres végrehajtás üzenete. Ezt a view-nál, vagyis a kinézetnek adjuk tovább. A hozzá tartozó view-hoz a redirect()->route('login') parancson keresztül érünk el. Ekkor az URL irányító központban a a web.php a Route::auth(); függvényen keresztül old fel, és irányít.

```
@section('content')
    @if ( Session::has('success') )
    <div class="vlayout-wrapper">
        <div class="row">
            <div class="col s12">
                <div class="card" style="background-color: #26A69A">
                    <div class="card-content white-text">
                        <span class="card-title">{{ Session::get('success')}}</span>
                        <p></p>
                    </div>
                </div>
            </div>
        </div>
    </div>
    @endif
```

Sikeres felvitel esetén vizsgáljuk, hogy a Session::has('success') értéke tartalmaz-e valamilyen értéket. Amennyiben igen akkor Session::get('success') függvénnyel kiírjuk a felhasználó tájékoztatására.

```
class User extends Model implements AuthenticatableContract, CanResetPasswordContract {

    use Authenticatable, CanResetPassword, EntrustUserTrait;

    protected $table = 'users';
    protected $fillable = ['email', 'password'];
```

A User class-ban a \$table jelöli a tábla nevét, amit a Laravel felold, hogy a művelet melyik táblában hajtódik végre. A \$fillable jelöli a tábla oszlopok neveit, amik valamilyen művelet végzés szemponttal rendelkeznek.

```
public function userContact()
{
    return $this->hasMany(Contact::class, 'user_id', 'contact_id');
}

public function userAddress()
{
    return $this->hasMany(Addresses::class, 'user_id', 'address_id');
}

public function booksto()
{
    return $this->hasMany(Book::class, 'user_id', 'book_id');
}
```

A User modelben függvények segítségével a Laravel képes felépíteni egy táblához, más táblákkal való kapcsolatát. Ennél a résznél a belongsToMany függvénnyel megadjuk a másik táblához lévő elérési utat, vagyis a hozzá tartozó sorokat kapcsolótáblán keresztül ellenőrzi a program.

Miután a Regisztráció végrehajtódott, bejelentkezhetünk a létrehozott felhasználónevünkkel. Ha adminnal jelentkezünk be azt a web.php-ban a Route-oknál jelölve van, kinek, milyen címekhez van joga. Ezzel is korlátozva a hozzáféréseket. A menünek a view-jánál ugyanígy ábrázolva van melyik felhasználó jogosult a menüpontok megnyitására.

Admin bejelentkezés esetén képesek vagyunk a jogosultság kezelésre ami a Permission illetve a Role modellel és Controllerével vannak társítva. Fejlesztés szempontjából érdemes csak a permission táblának módosításában változtatni a programon belül, de a Controllerben létrehozott módosítás törléssel ezt egy felületen meg lehet valósítani. Ezen kívül a Role felvitelnél illetve módosításnál, törlésnél a felvitel hasonló képen zajlik, mint egy regisztráció.

Az edzőterem weboldal másik legfontosabb része amikor az admin Egy cég felvitel és a hozzá tartozó adatokat tud felvinni. Ezt a Company Controllerben hajtja végre a program ahol megtörténik terem és a cím felvitel, módosítása is. A Company Controllerben a store függvény határozza meg, hogy egy cég hogyan jön létre, az edit függvény a megjelenítésre szolgál, amit a modellek alapján keresi ki az aktuális parancsot. Az all() függvényparancs lekérdezi az összes kapcsolatot amit kikell olvasnia az adatbázisból, de ez a parancs csak akkor lép teljes használatba, amikor hivatkozunk rá vagy a view-ban, vagy a controllerben. A másik függvény amit a program több helyen is alkalmaz az a find(), amiben a kapcsolatokon keresztül egy másik adatot kérjünk le. Alkalmaz még a program ezen kívül a where('id',\$id) függvényt ami egy táblás lekérdezés, és id szerint azonosítja be az adott sort egy táblában. Ez

SQL -ben egy WHERE utasítással egyenlő. Ezt használja program például szerkesztésnél is és a törlésnél is, aminek a kialakítása a programban összetettnek bizonyult, mivel törölni kell az órára jelentkezőt ha létezik, a hozzá kapcsolódó foglalást, az ahoz tartozó osztályt, az ahoz kapcsolt címet és az ahoz kapcsolt céget.

```
public function destroy($id)
{
    $company = Company::where('id',$id)->get();
    # lekérdezés a companyhoz tartozó címet
    $comp = Company::find($id)->sites()->get();
    foreach ($comp as $com) {
        # lekérdezés a címhez tartozó osztályt
        $address = Address::find($com->id)->classroom()->get();
        foreach ($address as $add) {
            # foglalás kikeresése ahol egyezik a teremszám a teremmel
            $books = Book::where('classroom_id', $add->id)->get();

            if ($books != null) {
                foreach ($books as $book) {
                    # kikeresi hogy van e foglalás az felvitt foglalások között és törlés detach-al
                    $userbook = Book::find($book->id)->userto()->detach();
                }
            }

            #törlés ahol a classroom_id egyezik
            $books = Book::where('classroom_id', $add->id)->delete();
        }
        # terem törlés
        $address = Address::find($com->id)->classroom()->delete();
        # terem kapcsoló törlés
        $address = Address::find($com->id)->classroom()->detach();
    }
    # cím törlés
    Company::find($id)->sites()->delete();
    # cím kapcsoló törlés
    Company::find($id)->sites()->detach();
    # cég törlés
    Company::where('id',$id)->delete();
    $request->session()->reflash();
    $request->session()->flash('success', 'Sikeres törlés!');
    return redirect()->route('company');
}
```

A törlés során a kiválasztott cég id-jét visszük tovább és vizsgáljuk le 7 táblán, hogy melyik táblák kapcsolódnak hozzá. Ezután a delete() függvény parancsal töröljük, és a detach() függvénnyel pedig a hozzá tartozó kapcsolótábla adatait töröljük. A másik legfontosabb dolog a foglalások kezelése. A programban a CompanyController kezeli. Itt a megjelenítés a legfontosabb, hogy hol, mikor, milyen órát, kik tartanak. ez a következőképp hajtódik végre a programban:

```
public function userbook()
{
    $books = Book::all();
    $userbooks = UserBook::all();

    return view('books.userbook')->with('books',$books)
        ->with('userbooks',$userbooks);
}

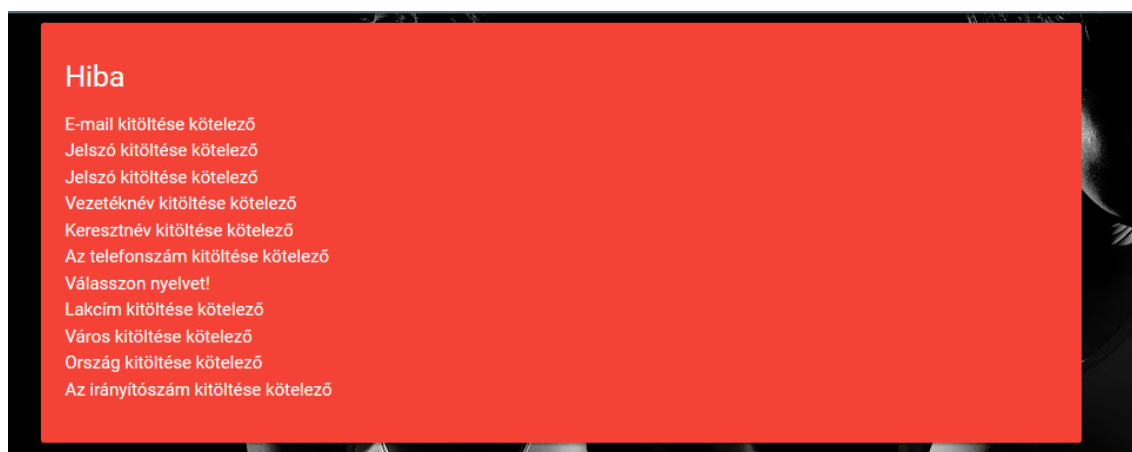
public function coachlesson()
{
    $books = Classroom::all();
    return view('books.coachlesson')->with('books',$books);
}
```

A `userbook()` függvény azokat az órákat küldi tovább a felhasználó felé, amely lekérdezi a modellen keresztül az összes foglalat és a hozzá tartozó kapcsolatokat, amelyeket továbbad a `with()` függvényen keresztül a megjelölt `view()` felületre. valamint továbbadja az összes

UserBook model függvényeit, amiből kiolvassa ugyanazokat az oszlopokat és a view ban vizsgáljuk. Azért van szükségét függvényre, mert a user által órára jelentkezést külön kezeljük, és az edző által felvitt, jelentkezésre szolgáló órákra

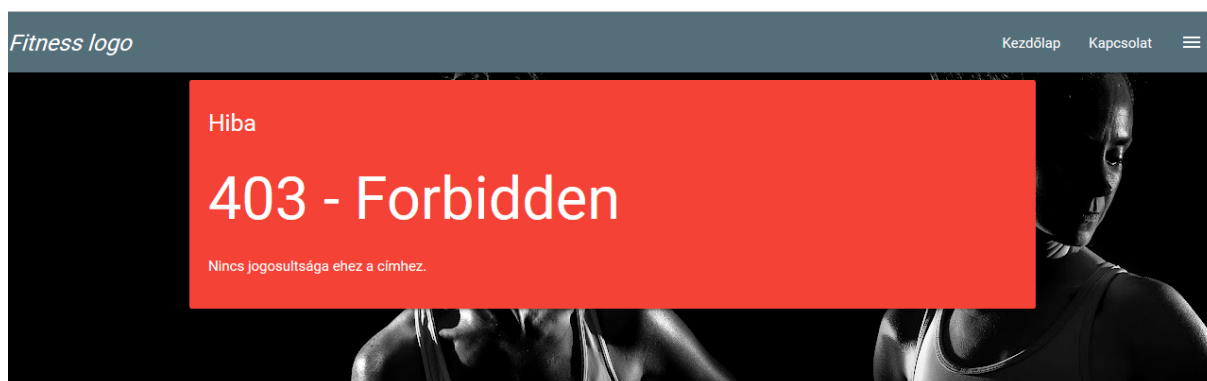
2.5 Tesztelési dokumentáció

Tesztelés Során különböző validációs kezelések, valamint hibaüzenetek megjelenítésének vizsgáltam le, ezzel biztosítottam az esetleges hibára futásokat. Ha a felhasználó valamilyen mezőt rosszul tölt ki például a regisztrációnál, akkor ez az üzenet fogadja:



Ebben persze szerepelnek a különböző szám, email felviteli validációk is hogy a felhasználó ne legyen képes abba a mezőbe felvinni olyan adatot, amit az adatbázis nem tud kezelni. Ezzel gátolva a felhasználót a hibás adat felviteltől.

Ha az oldal megtekintéséhez nem jogosult a felhasználó, akkor a következő üzenet fogadja:



Ezzel gátolva például, ha nem admin menüpontjainak valamilyen elérési útját, ne tudjon hozzáférni azokhoz. Ezt az adatbázisban kezeli le a jogosultságkezelő. Lehet szabályok szerinti irányításokat kiosztani, valamint jogok alapján kiosztani, hogy ki, mihez férhet hozzá.

Tesztelés során a felmerülő hibákat a Laravel beépített `dd()`; függvényével lehet ellenőrizni. Ez hasonló a `var_dump()`; parancsához. A `dd()`; függvénynél a függvény paraméterének megadjuk az ellenőrizni kívánt értéket és a lefutásával kimutatja annak összes értékét, adatbázis esetén minden kapcsolatot a hozzá tartozó táblák között és annak mezőneveit. Request kérésnél a felviteli összes mezőknek az értékeit, amit megadtunk neki ellenőrzésére, milyen értékkel futtatja tovább a program.

Tesztelés során elsőként felmerülő problémák:

- Feltöltések nem megfelelő működése:

A felhasználó táblák közötti összekötések, és a modellekben lévő táblanevek nem megfelelő meghatározása esetén a program hibába futott feltöltés során, amiben az adatbázis egyik mezője „null” értéket kapott. Ennek a hibának az okát elsőként a formon létrehozott mezőnevek, és a Controllerbe átadott mezőnevek egyezésével vizsgáltam, majd a metódusát, hogy megfelelő POST kérést küld-e tovább. Ha a hiba nem jelent meg itt, akkor a `web.php`-ban, ahol a forgalom irányítás zajlik, megvizsgálom, hogy a post kérés melyik Controlleren és annak melyik függvényén keresztül dolgozódik fel.

- Megjelenítési probléma:

A megjelenítés során van h a modellekben több táblás lekérdezések hajtódnak végre, ha lefuttatjuk a függvényt. A view-nál is lehetőséget ad a keretrendszer az ilyen lekérdezések létrehozására, azonban, ha a kapcsolások, tábla, vagy mező nevek rosszul szerepelnek akkor a program hibára fut. ilyenkor végig kell ellenőrizni a lekérdezés útját. Elsőként a Controllerben lekérdezem a Model teljes tartalmát (pl: `User::all()`;), ezt egy változóba teszem és átadom a hozzá tartozó kinézetnek, ahol `foreach`-el lekérdezem. Az érték visszaadásokat ellenőrzöm egytáblás lekérdezés esetében, valamint többtáblásnál függvények helyességét, a modellek tartalmát, illetve az átadott értékek helyességét.

2.6 Tovább fejlesztési lehetőségek

A weboldalon tovább fejlesztési lehetőség lehetne, a felhasználók szerkeszthetősége. Erre időhiány miatt nem sikerült befejezni a megvalósítását, azonban a Controllernél megvan neki a megfelelő függvény a végrehajtáshoz.

Másik tovább fejleszthető lehetőség lehet, a felhasználó tájékoztatása arról, hogy az edző a megjelentek ellenőrzése során feljegyezte-e hogy az órán részt vett-e vagy megjelent, mégis hiányzóként van feltüntetve. Ebből tovább lehet fejleszteni azt is, hogy a megjelent órák alapján a különböző cégekhez ármeghatározás alapján, összegezi az órán résztvevő személynek mennyit kellene fizetnie, és melyik cégnek kellene az órákat kifizetnie. Ebből kilehetne szűrni, ha túl sok órán nem jelent meg figyelmeztetést küldeni, hogy csak akkor jelentkezzen az órára, ha azon részt vesz, mert a terem létszámának megfelelő személy vehet részt az órán, és ne foglalja a helyet. Ezt tovább lehetne fejleszteni egy időszakos tiltással is például, ha túlsok órán nem vett részt, de feliratkozott, akkor tiltja ki annak az edzőnek az órájáról megadott ideig.

Tovább fejlesztési lehetőség még ezen a szálon, ha nem tud részt venni és jelezni kívánná, akkor törölhesse a jelentkezését az óráról, ezzel elkerülve a foglalási problémákat.

Az edző szempontjából fejlesztési lehetőség lenne, ha időközben egy óra elmaradás esetén tudná tájékoztatni a felhasználót üzenet keretében tájékoztatni az egyes óra elmaradásokról. Ezt tájékoztatás jellegűen a bejelentkezés után látná a felhasználó és ez alapján az elmaradó órája mellett megjelenne egy „Elmarad” felirat.

Fejlesztési lehetőségként, egy komolyabb PayPal online fizetési mód beépítése is lehetséges lenne, amivel egy szálát kiállítva a felhasználó online tudná kifizetni azokat az órákat, amikre jelentkezett. Ezt validációval lehetne korlátozni, hogy csak akkor jelentkezzen órákra, ha az előzőket kifizette már, és meg is jelent rajtuk.

A regisztráció meg könnyítése érdekében egy „Bejelentkezés Facebook” gombal vagy másfajta fiókkal könnyedén lekérdezhetőek, és elmenthetőek lennének a szükséges adatok és csak azokat a plusz mezőket kellene mellé kitölteni, amiket a másik fiókjából nem tudott a program eltárolni.

2.7 Irodalomjegyzék, forrásmegjelölés

Könyv:

Laravel keretrendszer hivatalos dokumentációja: Laravel 5 Documentation.pdf

Elérési hely: <https://leanpub.com/laravel-5>

Felhasznált tartalmak:

Hiba megoldás segítségére szolgáló oldalak:

<https://stackoverflow.com/>

<https://prog.hu/>

Felhasznált eszközök és dokumentációi:

Laravel hivatalos oldala: <https://laravel.com/>

Entrust Laravel Modul: <https://github.com/Zizaco/entrust>

Date Time Picker: <https://github.com/ripjar/material-datetime-picker>

Materialize Css hivatalos oldala: <http://materializecss.com/>

Informálódás keretrendszerekről:

<http://webmestertanfolyam.hu/webmester-blog/2015-legjobb-php-keretrendszerei>