**SZAKDOLGOZAT**

**Csanádi Péter**

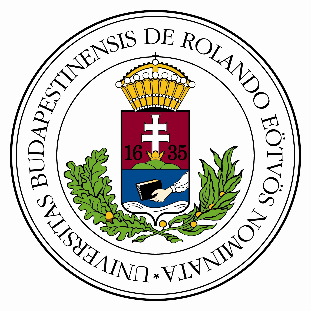
**2020**

**MTLSZ Játékvezetői Bizottság nyilvántartás**

**Bende Imre Csanádi Péter**

**tanársegéd programtervező informatikus BSc**

**Budapest, 2020**

 **EÖTVÖS LORÁND TUDOMÁNYEGYETEM**

**INFORMATIKAI KAR**

**SZAKDOLGOZAT-TÉMA BEJELENTŐ**

Név: **Csanádi Péter** Neptun kód: **YJRB29**

Tagozat: **esti** Szak: **programtervező informatikus BSc**

Témavezető neve: **Bende Imre**

munkahelyének neve és címe: **ELTE IK Média- és Oktatásinformatikai Tanszék**

**1117 Budapest, Pázmány Péter sétány 1/C.**

beosztása és iskolai végzettsége: **tanársegéd, tanári MA**

A dolgozat címe: **MTLSZ Játékvezetői Bizottság nyilvántartás**

A dolgozat témája: A Magyar Tollaslabda Szövetség által szervezett versenyek játékvezetői kérelmeinek és a játékvezetők jelentkezéseinek nyilvántartása.

A weboldal célja, hogy az MTLSZ Játékvezető Bizottsága vezetőjének feladatait megkönnyítse azzal, hogy a versenynaptárban szereplő versenyek és az azokra kért játékvezetők listázása mellett a potenciális játékvezetők a jelentkezéseiket a Bizottság vezetőjének manuális közreműködése nélkül megtehessék.

A verseny előtt pár nappal a Bizottság elnöke kiválaszthatja, hogy a versenyre jelentkezett játékvezetők közül kiket választ ki véglegesen és kik a tartalékok. A kiválasztott játékvezetők e-mail értesítést kapnak, mely leírja a verseny főbb adatait és a szokásos információkat.

Tervezett funkciók: versenyek felvitele, listázása, módosítása, törlése, versenyre jelentkezett játékvezetők listázása, játékvezető jelentkezés, jelentkezés törlése, jelentkezések listázása, kiválasztás a jelentkezett játékvezetők közül, tájékoztató e-mail kiküldése, elnök (Bizottság elnöke) által felhasználók (játékvezetők) jogosultságainak kezelése.

Tervezett technológiák: szerver oldal: natív PHP, adatbázis: MySQL, kliens, natív JS (a dátumválasztáshoz Jquery), design: Bootstrap.

A témavezetést vállalom:

....................................................

(a témavezető aláírása)

Kérem a szakdolgozat témájának jóváhagyását.

Budapest, 20................

....................................................

(a hallgató aláírása)

A szakdolgozat témát az Informatikai Kar jóváhagyta.

Budapest, 20................

....................................................

(a témát engedélyező tanszék vezetője)

Tartalomjegyzék

[1. Bevezetés 1](#_Toc26976950)

[2. Felhasználói dokumentáció 2](#_Toc26976951)

[2.1 A feladat 2](#_Toc26976952)

[2.2 Kezdőoldal 2](#_Toc26976953)

[2.3 Menü 2](#_Toc26976954)

[2.3.1. Normál felhasználó 2](#_Toc26976955)

[2.3.2. Adminisztrátor felhasználó 4](#_Toc26976956)

[3. Fejlesztői dokumentáció 10](#_Toc26976957)

[3.1 A telepítés menete 10](#_Toc26976958)

[3.1.1. platform 10](#_Toc26976959)

[3.1.2. Szoftverek 10](#_Toc26976960)

[3.1.3. A telepítés lépései 10](#_Toc26976961)

[3.2 Keretrendszer 11](#_Toc26976962)

[3.3 Adatbázis 12](#_Toc26976963)

[3.1.4. Migrációk 13](#_Toc26976964)

[3.1.5. A táblák feltöltése tesztadatokkal 13](#_Toc26976965)

[3.4 Admin middleware 13](#_Toc26976966)

[3.5 HTTP útvonalak 14](#_Toc26976967)

[3.6 Modellek 15](#_Toc26976968)

[3.6.1. Eloquent 15](#_Toc26976969)

[3.6.2. Játékvezetői és döntnöki szint (’UmpireLevel’, ’RefereeLevel’) 17](#_Toc26976970)

[3.6.3. Csarnok (’Venue’) 17](#_Toc26976971)

[3.6.4. Felhasználó (’User’) 18](#_Toc26976972)

[3.6.5. Verseny (’Tournament’) 18](#_Toc26976973)

[3.6.6. Jelentkezések (’UmpireApplication’, ’RefereeApplication’) 19](#_Toc26976974)

[3.7 Vezérlők 19](#_Toc26976975)

[3.7.1. Általános beállítások 19](#_Toc26976976)

[3.7.2. UserController 20](#_Toc26976977)

[3.7.3. VenueController 21](#_Toc26976978)

[3.7.4. TournamentController 22](#_Toc26976979)

[3.7.5. ApplicationController 24](#_Toc26976980)

[3.7.6. EmailController 25](#_Toc26976981)

[3.8 Nézetek 25](#_Toc26976982)

[3.8.1. Általános információ 25](#_Toc26976983)

[3.8.2. Blade sablonok 27](#_Toc26976984)

[3.8.3. Felhasználó nézetek (user mappa) 28](#_Toc26976985)

[3.8.4. Csarnok nézetek (venue mappa) 30](#_Toc26976986)

[3.8.5. Verseny nézetek (tournaments mappa) 31](#_Toc26976987)

[3.8.6. Email nézet (emails mappa) 33](#_Toc26976988)

[3.8.7. Hiba nézetek (errors mappa) 33](#_Toc26976989)

[4. Tesztelés 35](#_Toc26976990)

[4.1 Új felhasználó létrehozása 35](#_Toc26976991)

[4.2 Felhasználó törlése és visszaállítása 35](#_Toc26976992)

[4.3 Csarnok szerkesztése 35](#_Toc26976993)

[4.4 Verseny szerkesztése 36](#_Toc26976994)

[4.5 Jelentkezés versenyre 36](#_Toc26976995)

[4.6 Jelentkezések kezelése 37](#_Toc26976996)

[4.7 Értesítő email kiküldése 37](#_Toc26976997)

[5. Továbbfejlesztési lehetőségek 38](#_Toc26976998)

[6. Összegzés 39](#_Toc26976999)

# Bevezetés

Szakdolgozatom témája a Magyar Tollalabda Szövetség Játékvezetői és Versenybírói Bizottságának online nyilvántartása. A weboldal célja, hogy a bizottság tájékoztassa a versenybírókat a versenynaptárról, online felületet biztosítson a versenyekre való jelentkezéshez, ezen jelentkezéseket nyilvántarsa és a kiválasztott versenybírókat erről a tényről tájékoztassa.

Az 1969. óta önállóan létező Magyar Tollaslabda Szövetség Játékvezetői és Versenybírói bizottsága feladatai között szerepel a játékvezetők és versenybírók versenyekre való kijelölése és értesítése, valamint minősítése.

Az elmúlt években ennek megvalósítására a bizottság elnöke egy Google Drive-ban tárolt táblázatot használt, melynek soraiban szerepeltek a versenynaptár azon elemei, melyekre a szervező döntnököt és/vagy játékvezetőt igényelt. A versenybírók – ha jelentkezni kívántak egy versenyre – beírhatták a nevüket a táblázat megfelelő oszlopába. (Szerencsés esetben az addigi bejegyzéseket nem kitörölve.) Átmeneti megoldásként a még jelenleg is létező és használatos táblázatot csak a bizottság elnöke szerkesztheti, a többieknek megtekintési joguk van. Jelentkezni az elnöknek küldött email útján lehetséges, aki a feldolgozás során frissíti a táblázatot. Ezt a gyakorlatot kívánja leváltani az az alkalmazás, melynek jelen szakdolgozat kívánja letenni az alapjait.

Szakdolgozat a Laravel keretrendszerrel készül, amelyet azért választottam, mert egyrészt az authentikációra szinte kész megoldást szállít, másrészt olyan eszközöket és technológiákat használ és szolgáltat, amik helyes használat esetén leveszik a fejlesztő válláról az alacsony szintű programozás terhét, hogy teljes mértékben a megoldandó feladatra koncentrálhasson és logikailag rendszerezni tudja azt.

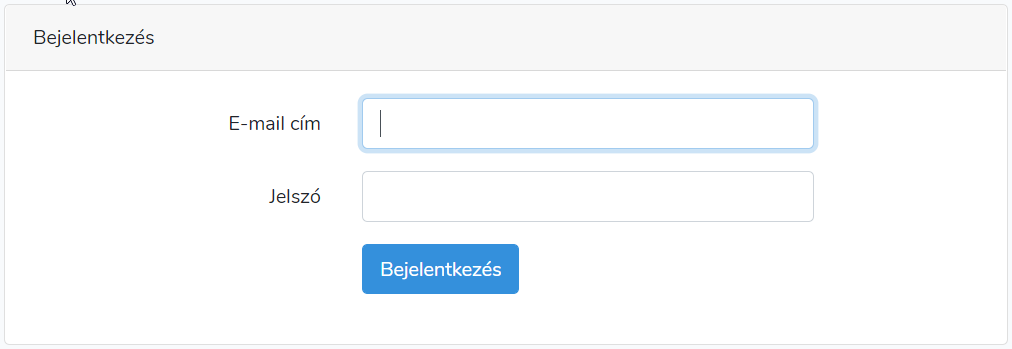
# Felhasználói dokumentáció

## A feladat

A weboldal célja, hogy a versenybírók (döntnökök és játékvezetők) jelentkezéseinek nyilvántartását kezelje, online elérést biztosítson minden felhasználónak, aki authentikáció után saját maga tekintheti meg a versenynaptárat és azt, hogy mely versenyekre és milyen szerepre jelentkezett.

## Kezdőoldal

Mivel a versenynaptár megtekinthető a <http://badminton.hu/versenyek> címen, nincs olyan része a weboldalnak, melyet az authentikáció előtt érdemes megjeleníteni. Ezen megfontolásból a kezdőoldalon csak bejelentkezésre van lehetőség.



1. ábra - Bejelentkezés

## Menü

Bejelentkezés után minden oldal tetején a menü jelenik meg, ami az oldalon való navigációt segíti.

### Normál felhasználó

Normál felhasználóként az alábbi kettő menüpont áll rendelkezésre:



2. ábra - Menü (normál felhasználó)

#### Versenynaptár

A sikeres authentikációt követően a versenynaptár jelenik meg. A képernyőn az aktuális versenyszezon versenyei látszanak (dátum, elnevezés, helyszín, igényelt játékvezetők száma, eddig jelentkezett játékvezetők száma). Ha a felhasználó jogosult döntnökként közreműködni versenyeken, akkor egy további oszlop jelenik meg ’Döntnök’ fejléccel, mely oszlop segítségével a versenyekre jelentkezni tud. (Amennyiben jelentkezett, ezt visszavonni tudja.) Hasonlóan ha a felhasználó rendelkezik játékvezetői minősítéssel, a jelentkezéseit a ’Játékvezető’ fejlécű oszlopban kezelheti. A jelentkezés (és leiratkozás) csak jövőbeli versenyekkel kapcsolatban lehetséges.



3. ábra - Versenynaptár

A jelenlegitől eltérő versenyszezon versenyeinek megjelenítésére a táblázat felett látható navigációs gombokkal van lehetőség.



4. ábra - Jelentkezéseim

#### Jelentkezéseim

Ezen az oldalon azok a jövőbeli versenyek jelennek meg, ahova a felhasználó akár döntnöki akár játékvezetői szerepre jelentkezett. Az elrendezés a versenynaptáréhoz hasonló, itt azonban jelentkezésre nem, csak a már megtett, de még nem jóváhagyott jelentkezések visszavonására van lehetőség.

#### Jelentkezés versenyre

A versenynaptárban lévő versenyekre a felhasználó akkor tud jelentkezni, ha rendelkezik az adott szerepkörben minősítéssel.

A ’Döntnök’ és ’Játékvezető’ oszlopokban lévő ikonok jelentése:

* jelentkezett (még nem jóváhagyott):

a felhasználó már jelentkezett a versenyre, de a bizottság még nem választotta ki a verseny közreműködőit (a jelentkezés még visszavonható a kisebb ikonra kattintva)

* jelentkezett (jóváhagyott):

a felhasználó jelentkezett a versenyre és a bizottság kiválasztotta közreműködőnek

* jelentkezett (elutasított):

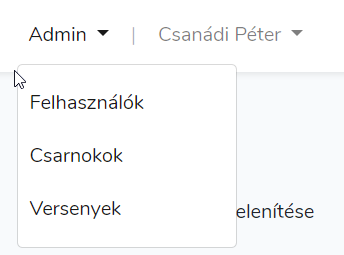
a felhasználó jelentkezett a versenyre, de a bizottság őt nem választotta ki közreműködőnek

* nem jelentkezett:

a felhasználó még nem jelentkezett a versenyre (jelentkezhet a kisebb ikonra kattintva)

### Adminisztrátor felhasználó

Az adminisztrátor jogosultsággal rendelkező felhasználók további menüpontokat érhetnek el, melyek a fenti menü jobb oldalán az Admin menüpont lenyitásával érhetők el.



5. ábra – Menü (adminisztrátor felhasználó)

#### Felhasználók

Ezen az oldalon listázhatjuk az adatbázisban szereplő felhasználókat. Megjelenített adatok: név, email cím, játékvezetői (Jv) szint, döntnöki (D) szint, adminisztrátor-e a felhasználó. Minden sorban találunk ’Szerkesztés’ és ’Törlés’ (törölt felhasználó esetén ’Visszaállítás’) gombokat.



6. ábra - Felhasználók

##### Felhasználó törlése és törölt felhasználó visszaállítása

A felhasználó sorában található törlés gomb segítségével törölhetjük. A törölt felhasználók megjelenítése a táblázat felett található jelölőnégyzet segítségével lehetséges. Ha egy felhasználó már törölt, a törlés gomb helyett a visszaállítás gomb jelenik meg, amivel a törlést visszavonhatjuk.



**7**. ábra - Felhasználó szerkesztése

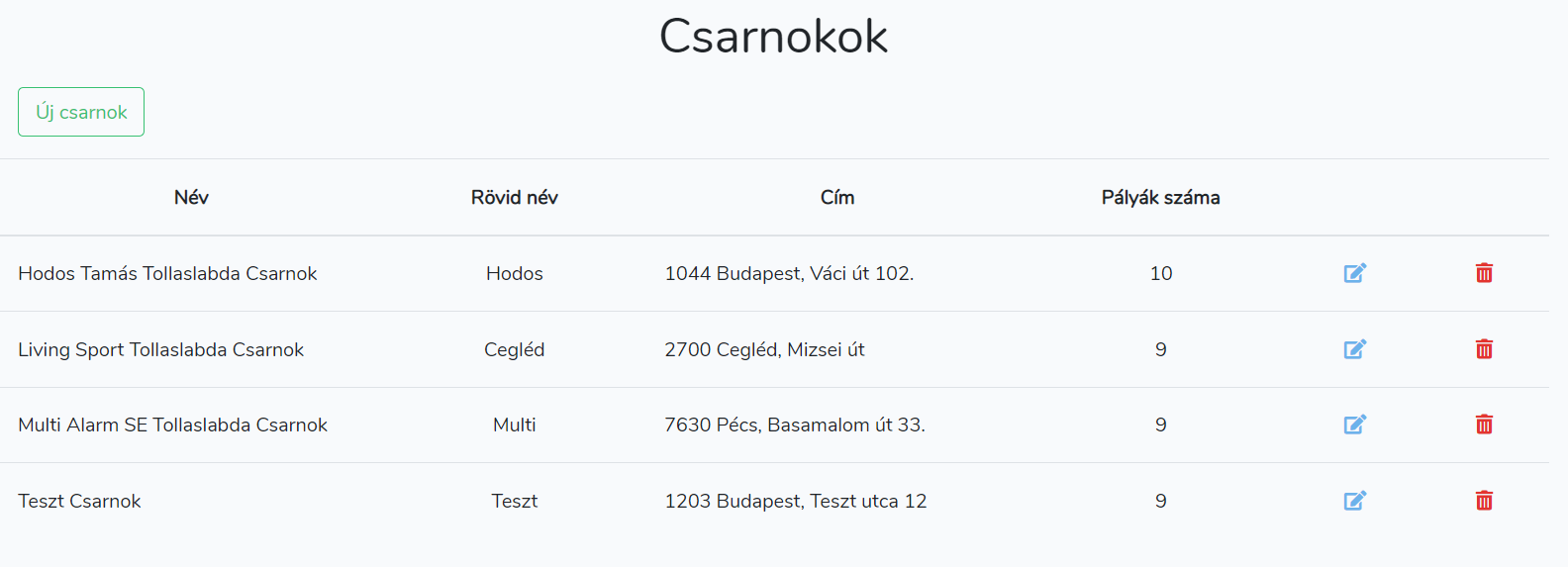
##### Felhasználó létrehozása

Új felhasználót az ’Új felhasználó’ gombra kattintva és a megfelelő adatokat megadva tudunk létrehozni. Kezdeti jelszót kötelező megadni (vagy a mező melletti gombra kattintva generálni).

##### Felhasználó szerkesztése

Ezen az oldalon egy meglévő felhasználó adatait módosíthatjuk. A beviteli mezők a felhasználó jelenlegi adataival vannak eredetileg kitöltve. A ’jelszó’ mezőt üresen hagyva a jelenlegi jelszó érvényben marad, a kitöltés ugyanúgy lehetséges, mint új felhasználó létrehozása esetén. A módosítások elvégzése után az oldal alján található gombok közül választhatunk:

* **visszaállítás**: A beviteli mezők eredeti (a módosítások előtti) értékekre való visszaállítása. (A ’jelszó’ mező üres lesz)
* **mégse**: Visszatérés az összes felhasználót listázó oldalra a módosítások mentése nélkül.
* **mentés**: Az adatok mentése és visszatérés az összes felhasználót listázó oldalra. Az oldal tetején (a menü alatt) látható üzenet tájékoztat, hogy a mentés sikeres volt-e.



8. ábra - Csarnokok

#### Csarnokok

Ezen az oldalon listázhatjuk az adatbázisban szereplő csarnokokat. Megjelenített adatok: név, rövid név, cím, pályák száma. Minden sorban találunk ’Szerkeszté’ és ’Törlés’ (törölt csarnok esetén ’Visszaállítás’ gombokat. Előbbi a szerkesztés oldalra irányít, az utóbbira való kattintás után ugyanezen az oldalon maradunk. Minden sorban találunk ’Szerkesztés’ és ’Törlés’ (törölt csarnok esetén ’Visszaállítás’) gombokat.

##### Csarnok törlése és törölt csarnok visszaállítása

A csarnok sorában található törlés gomb segítségével törölhetjük. A törölt csarnokok megjelenítése a táblázat felett található jelölőnégyzet segítségével lehetséges. Ha egy csarnok már törölt, a törlés gomb helyett a visszaállítás gomb jelenik meg, amivel a törlést visszavonhatjuk.

##### Csarnok felvitele

Új csarnokot ’Új csarnok’ gombra kattintva és a megfelelő adatokat megadva tudunk létrehozni.



9. ábra - Csarnok szerkesztése

##### Csarnok szerkesztése

Ezen az oldalon egy meglévő csarnok adatait módosíthatjuk. A beviteli mezők a csarnok jelenlegi adataival vannak kitöltve. A módosítások elvégzése után az oldal alján található gombok közül választhatunk.

#### Versenyek

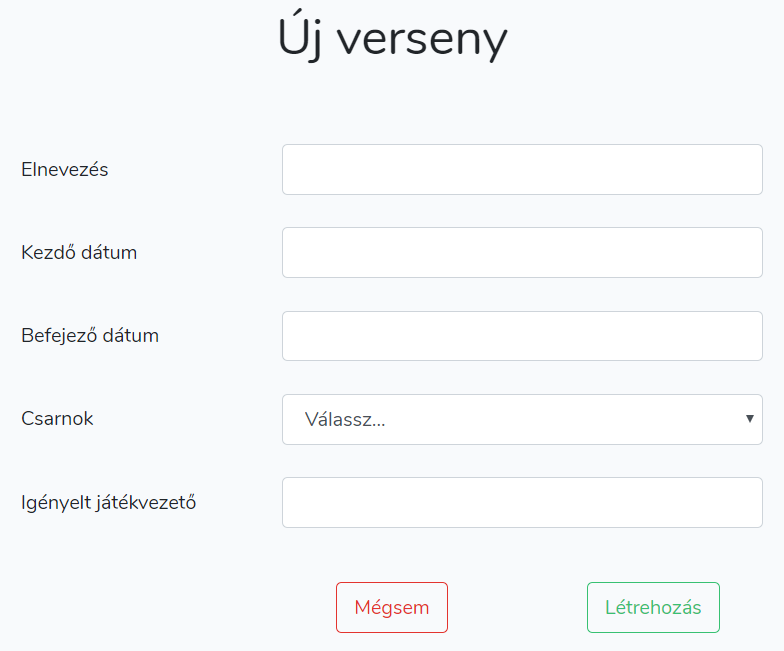
Ezen az oldalon listázhatjuk az adatbázisban található versenyeket. Megjelenített adatok: Név, kezdő dátum, befejező dátum, csarnok, igényelt játékvezetők száma. A ’csarnok’ oszlopban annak a csarnoknak a rövid neve szerepel, ahol a verseny megrendezésre fog kerülni. Ha az egeret egy csarnok rövid neve fölé visszük, megjelenik a csarnok teljes neve és címe.

##### Verseny törlése és törölt verseny visszaállítása

A verseny sorában található törlés gomb segítségével törölhetjük. A törölt versenyek megjelenítése a táblázat felett található jelölőnégyzet segítségével lehetséges. Ha egy verseny már törölt, a törlés gomb helyett a visszaállítás gomb jelenik meg, amivel a törlést visszavonhatjuk.

##### Verseny felvitele

Új csarnokot ’Új csarnok’ gombra kattintva és a megfelelő adatokat megadva tudunk létrehozni.



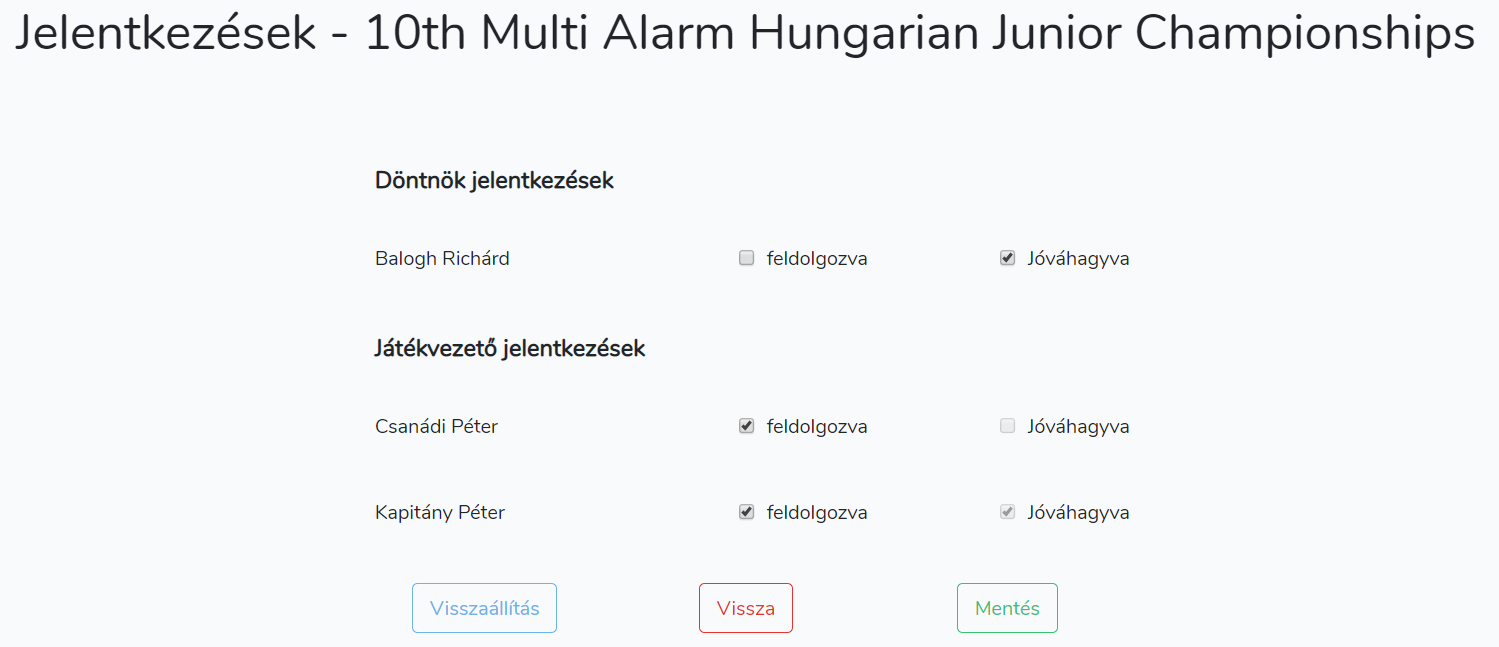
10. ábra - Verseny felvitele

##### Verseny szerkesztése

Ezen az oldalon egy meglévő verseny adatait módosíthatjuk. A beviteli mezők a verseny jelenlegi adataival vannak eredetileg kitöltve. A módosítások elvégzése után az oldal alján található gombok közül választhatunk, melyek működése a felhasználók szerekesztése oldalon szereplő gombokéval megegyezik.

##### Jelentkezések kezelése

Ezen az oldalon felsorolásra kerül azok a felhasználók, akik döntnök vagy játékvezető szerepkörre jelentkeztek a versenyre. Eldönthetjük minden jelentkezésről, hogy jóváhagyjuk vagy sem, ezután pedig azt, hogy a jelentkezést feldolgozottnak jelezzük. Ha egy jelentést feldolgoztunk, nem tudjuk a jóváhagyás állapotát változtatni. (Az értesítő emailt kiküldésekor azok a felhasználók kapják meg, akiknek a jelentkezése feldolgozott és jóváhagyott.)



11. ábra - Jelentkezések kezelése

##### Értesítő email kiküldése

Azoknak a felhasználóknak, akik a versenyre jelentkeztek és a jelentkezésük feldolgozott és jóváhagyott státuszú, értesítő levelet küldhetünk. A levél általános tájékoztatást tartalmaz.

# Fejlesztői dokumentáció

## A telepítés menete

### platform

A fejlesztés Windows 10 operációs rendszeren történt.

### Szoftverek

* Xampp (Cross platform Apache, MariaDB, PHP,Perl)

<https://www.apachefiends.org/index.html>

* Composer

<https://getcomposer.org/download/>

* node.js

<https://nodejs.org/en/>

### A telepítés lépései

A fenti szoftverek telepítése után:

> git clone <https://github.com/pcsanadi/szakdolgozat.git>

A forráskód letöltése a githubról.

> cd <projekt elérési út>

> composer global require laravel/installer

> composer install

A Laravel keretrendszer letöltése és telepítése

> npm install

> npm run dev

A node.js package manager telepítése és futtatása

A <xampp\_mappa>\apache\conf\httpd.conf fájlban:



DocumentRoot: Ebben a mappában keresi a webkiszolgáló a kezdőoldalt (*index.php*)

A http://localhost/phpmyadmin felületen új felhasználó (minden jogosultsággal) és hozzá azonos nevű adatbázis létrehozása

> cp .env.example .env

A *.env.example* minta konfigurációs fájl lemásolása (alapértelmezetten ez a fájl kimarad a verziókezelésből, hiszen minden fejlesztő környezetében más értékekre lehet szükség).

> php artisan key:generate

Kulcs generálása (beállítja a *.env* fájlban az APP\_KEY értékét).

A.env fájlban a következő sorokat kell még átírni:



APP\_NAME: az alkalmazás neve, ez jelenik meg a menüsor bal oldalán isAPP\_DEBUG: hiba esetén szűkszavúbb hibaüzenet jelenik meg a böngészőben

APP\_URL: az alkalmazás a böngészőben ezen az URI-n érhető el

MAIL\_DRIVER: helyi fejlesztés során nem küld ki a weboldal emaileket (a kiküldendő levelek a *storage/logs/laravel-<dátum>.log* fájlban találhatók).

> php artisan migrate:fresh –seed

Minden adatbázistábla eldobása, minden migrációs fájl végrehajtása, a táblák kezdeti adatokkal való feltöltése.

## Keretrendszer

A szakdolgozat a Laravel keretrendszer felhasználásával készült. Készítői hisznek benne, hogy a fejlesztésnek élvezhető és kreatív élménynek kell lennie és ez a keretrendszer leveszi a teher egy részét a fejlesztő válláról azzal, hogy megkönnyíti a legtöbb web projektben előforduló általános tevékenységeket.

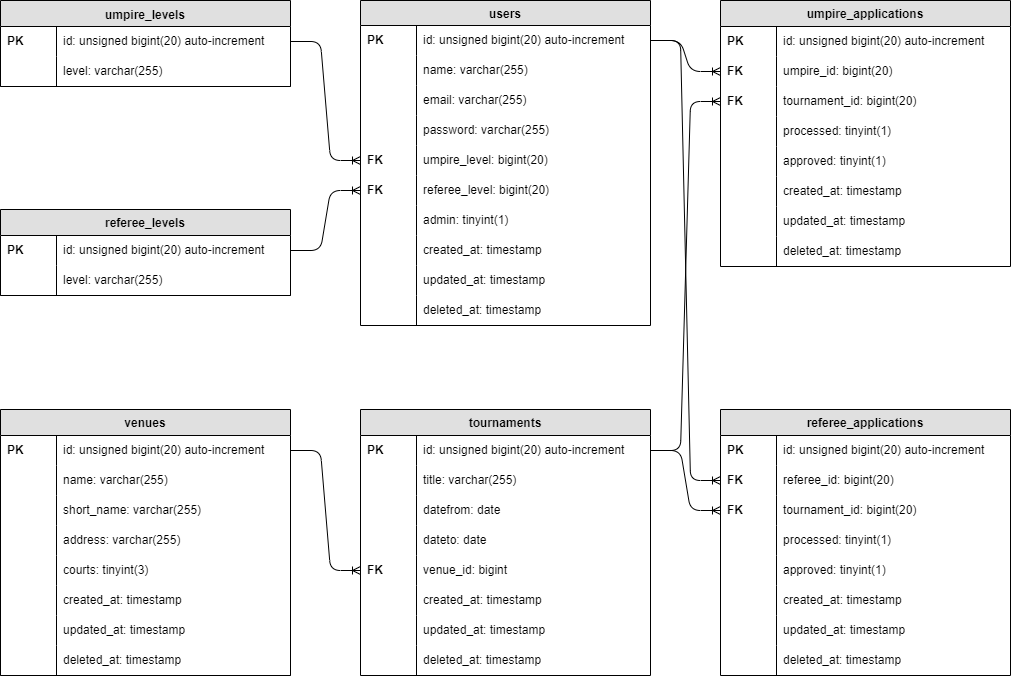
Minden beérkező kérés belépési pontja a *public/index.php* file, ami nem tartalmaz sok kódot, a feladata mindössze annyi, hogy elindítsa a keretrendszer többi részének betöltését. Szakdolgozatomban a console (api) requestek automatikusan elutasításra kerülnek, a HTTP kéréseket pedig a HTTP Kernel dolgozza fel, mely az *Illuminate\Foundation\Http\Kernel* osztály leszármazottja. Tartalmazza bootstrapperek egy tömbjét, melyek olyan elemekért felelősek, mint pl. a hibakezelés és a naplózás konfigurációja, a futtatási környezet feltérképezése és egyéb feladatok, amiket még a kérés feldolgozása előtt meg kell tenni. Ezt a részt nem módosítottam, legfeljebb a konfigurációs beállításokat tartalmazó *config* mappa fájljain keresztül.

A HTTP kernel definiál ú.n. middleware-eket is, amiken minden kérésnek még a feldolgozás előtt át kell haladnia. Az alapértelmezetten rendelkezésre álló (és az autentikáció hozzáadása után létrejött) middleware-ek mellett egy middleware-t definiáltam (*RedirectIfNotAdmin*), melynek célja, hogy az adminisztrátor jogosultsággal nem rendelkező felhasználók ne érjenek el nekik nem engedélyezett részeket.

Egyéb felhasznált technológiák:

* Bootstrap: nyílt forráskódú eszközkészlet, HTML, CSS és JavaScript használatával történő fejlesztéshez, reszponzív rácsvonal rendszer, könnyen kiegészíthető előkészített komponensekkel
* jQuery: gyors, kicsi, feature-gazdag JavaScript könyvtár

## Adatbázis

****

**12**. ábra - Adatbázis

Az alkalmazás felépítéséből adódóan a felhasználókat (potenciális döntnököket és játékvezetőket) tartalmazó *’users’* és a versenyeket tartalmazó *’tournaments’* tábláknak központi szerepe van.

Az *’umpire\_levels*’és *’referee\_levels*’ táblák kivételével – melyek várhatóan sosem fognak változni – minden tábla rendelkezik a következő mezőkkel: ’*created\_at’, ’updated\_at’*, valamint ’*deleted\_at’*. Ezeket a mezőket a Laravel keretrendszer hozza létre és kezeli. Előbbi kettő a változások követésére, utóbbi a – később visszavonható – törlés kezelésére szolgál.

Minden tábla elsődleges kulcsa (PK) az ’*id’,* ezek az egy-több kapcsolatokban külső kulcsként (FK) is előfordulnak

### Migrációk

A Laravel az adatbázis tábláit ú.n. migrációk segítségével hoza létre (és módosítja). A migrációk a *database/migrations* mappában találhatók. Minden migráció tartalmaz egy *up()* és egy *down()* metódust. Előbbi segítségével létrehozhatunk és módosíthatunk táblákat, utóbbi pedig ezen műveletek megfordítására való. A migrációk sorrendjét a fájlnevekben szereplő időbélyegek jelzik a keretrendszer számára. Szakdolgozatomban csak táblákat létrehozó migrációk vannak (és a táblák eldobására vonatkozó parancsok), az adatbázis felépítésére alkalmasak, egyfajta verziókövetésre nem.

### A táblák feltöltése tesztadatokkal

A *database/seeds* mappában található osztályok a létrehozott adatbázistáblák kezdeti és/vagy tesztadatokkal való feltöltésére szolgálnak. Minden táblához tartozik seed osztály elsősorban tesztelési céllal, bár ha a *’users’* táblában egy adminisztrátor jogosultsággal rendelkező felhasználó sem lenne, akkor nem tudnánk használni az alkalmazást.

A migrációk és seedek futtatása az alábbi paranccsal történik:



A parancs hatására az összes eddigi adatbázistábla eldobásra kerül (*down()* metódus), majd újra létrejönnek és a *--seed* kapcsoló hatására a kezdeti adatokkal feltöltődnek.

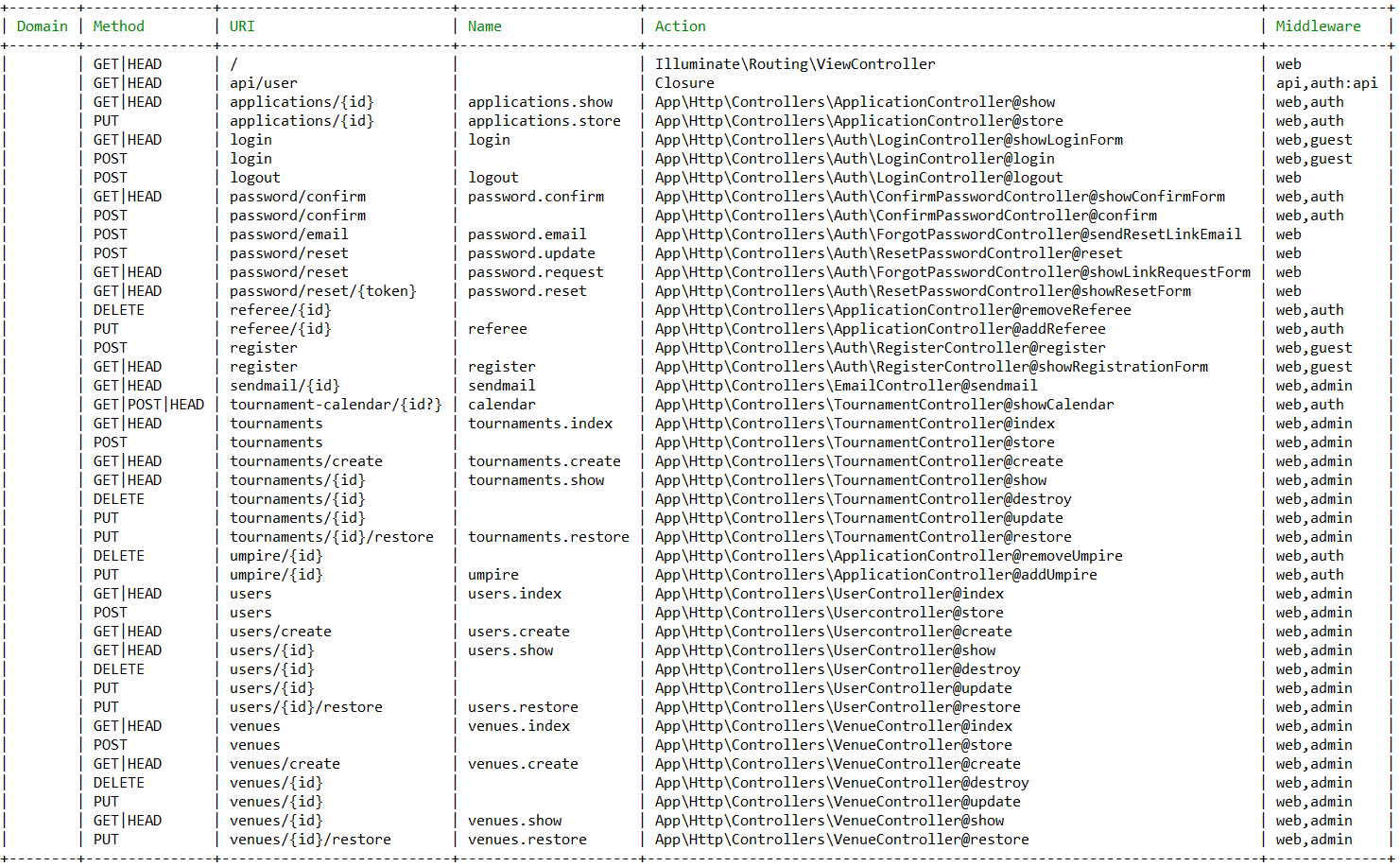
## Admin middleware

A keretrendszer által eleve biztosított (valamint az autentikáció hozzáadásával bekerült) middleware-ek kiegészítéseként egy újabb middleware-re is szükség van (*app/Http/Middleware/RedirectIfNotAdmin.php*). Ugyan sorrendben az *Authenticate* után áll (ld. *app/Http/Kernel.php:$middlewarepriority*), az útvonalak definiálásakor mindkettő nem került beállításra, így ez a middleware is ellenőrzi, hogy létezik-e felhasználó (vagyis bejelentkezett-e). Ezután foglalkozik azzal, hogy adminisztrátor jogosultsággal rendelkezik-e. Ha ezen feltételek egyike nem teljesül, átirányít a ’*versenynaptár*’ útvonalra. Különben a kérést továbbítja (a következő middleware-nek).



## HTTP útvonalak

A programban a következő útvonalak vannak definiálva:



13. ábra - Útvonalak

* Method: az elfogadott HTTP kérés típusok
* URI: az elfogadott elérési út
* Name: az útvonal neve
  + a {} közötti paraméterek kötelezőek (kivéve, ha ? szerepel mögöttük)
  + a keretrendszer *route()* segédfüggvénye a név és az esetleges paraméterek alapján visszaadja az útvonal elérési útját
* Action: a kezelés módja, amikor az útvonalon kérés érkezik be
  + egy kivétellel minden beérkező kérést egy vezérlő metódus dolgoz fel
* Middleware: azok a middleware-ek, amiken a kérés átmegy, mielőtt feldolgozásra kerülne

Az útvonalak definiálása és tulajdonságaik beállítása a HTTP kérések kapcsán a *routes/web.php* fájlban történik. A *route* osztály statikus metódusai segítségével adhatjuk meg, hogy milyen útvonalról, milyen HTTP kérést hogyan kezeljen a keretrendszer. A kezelés történhet egy vezérlő egy metódusával vagy egyszerűen egy nézetre való átirányítással. Ha akarunk, nevet is adhatunk az útvonalnak, mely lehetővé teszi, hogy a *route()* segítségével a hozzá tartozó elérési utat egy nézetbe könnyen beilleszthessük.

Példa:

**

Ha PUT HTTP kérés érkezik az *’/umpire/{id}’* útvonalon (az {id} kötelező paraméter), akkor az ApplicationController *addUmpire()* függvénye dolgozza fel. Az útvonal neve *’umpire’*.

## Modellek

### Eloquent

A Laravel keretrendszerhez tartozó Eloquent ORM (object-relational mapping) az AcitveRecord design pattern egy megvalósítása az adatbázissal való munkához. Minden adatbázistáblához tartozik egy modell, amin keresztül a táblán műveleteket végezhetünk.

#### Általános beállítások

Az alábbi változók szinte az összes modellnél beállításra kerültek, az Eloquent ezek segítségével határozza meg a modell és az adatbázistábla közti kapcsolat jellegét és a kezelés módját.

* *$table*: a modellhez tartozó adatbázistábla neve
* *$primaryKey*: az adatbázistábla elsődleges kulcsa
* *$incrementing*: az elsődleges kulcs automatikusan növekszik-e
* *$timestamps*: szükség van-e a ’created\_at’ és ’updated\_at’ mezőkre, melyek nyilvántartják, hogy egy rekord mikor jött létre és mikor változott meg utoljára
* *$casts*: azon mezők listája, amelyeket beolvasáskor valamilyen beépített PHP típusra kell kényszeríteni
* *$with*: azon egyéb modellek listája, melyeket beolvasáskor mindig be kell tölteni, mert a visszaadott változón kresztül azokat is el akarjuk érni
* *$dates*: a dátum típusú mezők listája

#### Soft delete

A rekordok valós letörlése mellett az Eloquent része a ’soft delete’ is. Ebben az esetben a modell nem kerül törlésre az adatbázisból, ehelye.tt a *’deleted\_at’* attribútuma kerül beállításra és a táblában tárolásra. Ha a modellnek nem null a *’deleted\_at’* értéke, akkor a modell ’soft deleted’.

A ’soft delete’ használatához a modellben a *’use SoftDeletes’* direktívára van szükség. Szakdolgozatomban a játékvezetői és döntnöki szinteket reprezentáló modelleken kívül mindenhol ezzel a lehetőséggel él a kiszolgáló. Ez teszi lehetővé, hogy a modellek az esetleges törlés után visszaállíthatók legyenek.

Amikor az Eloquent segítségével ki akarjuk listázni egy modell összes rekordját, az *all()* függvényt használjuk. Ezeknél a modelleknél viszont ez a függvény csak a nem törölt rekordokat adja vissza. Ha tényleg minden rekordra szükségünk van, a *withTrashed()* hívására is szükség van. (Az *onlyTrashed()* pedig csak a törölt elemeket listázza.)

#### Kapcsolatok

Az adatbázistáblák közötti kapcsolatok leírásához és használatához az Eloquent kényelmes eszközt kínál. Szakdolgozatomban a táblák között egy-több kapcsolatok vannak, melyek a következő módon definiálhatók:

app/UmpireApplication.php:



app/User.php:



Ezen a módon az *UmpireApplication* modellben elérhető a jelentkezéshez tartozó felhasználó egyszerűen a *’user’* attribútumon keresztül. (És elérhető a felhasználó minden attribútuma is, pl. *$application->user->name*.) A *’User’* modellben pedig elérhető lesz a *’umpireApplications’* attribútum, amin például foreach ciklussal végigiterálhatunk.

A kialakított egy-több kapcsolatok a következők:

|  |  |  |
| --- | --- | --- |
| **egy** | **több** | **kulcs** |
| *UmpireLevel* | *User* | *umpire\_level* |
| *RefereeLevel* | *User* | *referee\_level* |
| *Venue* | *Tournament* | *venue\_id* |
| *User* | *UmpireApplication* | *umpire\_id* |
| *User* | *RefereeApplication* | *referee\_id* |
| *Tournament* | *UmpireApplication* | *tournament\_id* |
| *Tournament* | *RefereeApplication* | *tournament\_id* |

A modellekben szereplő *$with* változó elemei azt jelzik, hogy az adott modell adatbázisból való betöltésekor a kapcsolatban álló modellt is töltse be az Eloquent.

Akár többszörös kapcsolat betöltési igényét is jelezhetjük.

(Pl. az *’UmpireApplication’* modell ben a *’tournament.venue’* hatására elérhető ez is: *$application->tournament->venue->address*.)

### Játékvezetői és döntnöki szint (*’UmpireLevel*’, *’RefereeLevel*’)

A lehetséges játékvezetői és döntnöki szinteket reprezentáló modellek. A hozzájuk tartozó adatbázistáblák: *’umpire\_levels’* és *’referee\_levels’*.

Attribútumaik:

* azonosító (*id*)
* szint (*level*)

Az attribútumok és az adatbázistáblában nekik megfeleltethető mezők nevei megegyeznek.

Ezek a modelleki konstansnak tekinthetők, nem várható, hogy a lehetséges játékvezetői és döntnöki szintek a közeljöbőben változnak, ezért nem is tartozik a modellekhez controller. Szerepük mindössze az, hogy a felhasználók játékvezetői és döntnöki szintjei ne szövegként legyenek ’beleégetve’ az adatbázis rekordjaiba. Ha mégis szükséges változtatni rajtuk, az a web api-n keresztül nem, csak közbetlen adatbáziskapcsolattal lehetséges.

### Csarnok (*’Venue’*)

A tollaslabda versenyek magyarországi lebonyolítására alkalmas csarnokokat reprezentáló modell. A hozzá tartozó adatbázistábla: *’venues*’.

Attribútumai:

* azonosító (*id*)
* név (*name*)
* rövid név (*short\_name*)
* cím (*address*)
* pályák száma (*courts*)
* versenyek (*tournaments* – kapcsolat által)

Az attribútumok és az adatbázistáblában nekik megfeleltethető mezők nevei megegyeznek.

### Felhasználó (*’User’*)

A felhasználókat reprezentáló modell. A hozzá tartozó adatbázistábla: *’users’*.

Attribútumai:

* azonosító (*id*)
* név (*name*)
* email cím (*email*)
* jelszó (*password*)
* játékvezetői szint (*umpireLevel* – *umpire\_level*)
* döntnöki szint (*refereeLevel* – *referee\_level*)
* adminisztrátor-e (*admin*)
* játékvezetői jelentkezések (*umpireApplications* – kapcsolat által)
* döntnöki jelentkezések (*refereeApplications* – kapcsolat által)

Az attribútumok és az adatbázistáblában nekik megfeleltethető mezők nevei megegyeznek.

### Verseny (*’Tournament’*)

A magyarországi tollaslabda versenyeket reprezentáló modell. A hozzá tartozó adatbázistábla: *’tournaments’*.

Attribútumai:

* azonosító (*id*)
* megnevezés (*title*)
* kezdő dátum (*datefrom*)
* befejező dátum (*dateto*)
* csarnok (*venue* – *venue\_id)*
* igényelt játékvezetők száma (*requested\_umpires*)
* játékvezetői jelentkezések (*umpireApplications* – kapcsolat által)
* döntnöki jelentkezések (*refereeApplications* – kapcsolat által)

### Jelentkezések (*’UmpireApplication’*, *’RefereeApplication’*)

A felhasználók jelentkezéseit reprezentáló modellek. A hozzájuk tartozó adatbázistáblák: *’umpire\_applications’* és *’referee\_applications’*.

Attribútumaik:

* azonosító (*id*)
* játékvezető (*umpire* – kapcsolat által) és döntnök (*referee* – kapcsolat által)
* verseny (*tournament* – kapcsolat által)
* feldolgozva (*processed*)
* jóváhagyva (*approved*)

Amikor a program email értesítést küld egy versenyre kiválasztott döntnök(ök)nek és játékvezetőknek, azokat a felhasználókat választja ki, melyeknek a jelentkezése jóváhagyott és feldolgozott státuszú.

## Vezérlők

A vezérlők az eseményeket, felhasználói műveleteket dolgozzák fel és válaszolnak rájuk, az *app/Http/Controllers* mappában találhatók. A HTTP útvonalak definíciója során minden létrehozott útvonalon beérkezett kérést az egyik vezérlő megfelelő metódusa dolgozza fel.

### Általános beállítások

A konstruktorban (*\_\_construct()*) van lehetőség megadni, hogy a vezérlő metódusai esetlegesen melyik middleware közbeiktatása után dolgozzák fel a beérkezett kérést. A *$this->middleware()* segítségével adhatjuk meg általánosan, melyet később az *only()* és *except()* hívásokkal specializálhatunk. Azok a metódusok, amelyek csak adminisztrátor jogosultságú felhasználók számára elérhetőek, az *’admin’* middleware-t használják, a többi pedig az *’auth’*-ot.

A vezérlők döntik el azt is, hogy a beérkező kérésnek értelmes-e és a felhasználó, akinek a nevében a kérés érkezett, jogosult-e a kért parancs kezdeményezésére. Ha ezen feltételbármelyike nem áll fenn vagy valamilyen más hiba történt (pl. nem sikerült egy modellt az adatbázisból kiolvasni), akkor az *abort()*, *abort\_if()* és *abort\_unless()* függvényhívásokkal a vezérlő kivételt vált ki, melyben beállítja a hibakódot és a hibaüzenetet. Ezeket a kivételeket a keretrendszer elkapja és ha van a HTTP válaszkódnak megfelelő nézet, akkor a hibát azzal jeleníti meg.

A vezérlők metódusai a kérés feldolgozása során szükséges lépések elvégzése után vagy egy nézettel tér vissza (amihez mellékeli a szükséges változókat), vagy átirányít egy másik útvonalra (szintén esetenként addicionális változókkal).

Pl.:



### UserController

Elsősorban a felhasználó (*’User’*) modellel kapcsolatban beérkező kérések feldolgozásáért felel.

Metódusai:

* *index()*: az összes felhasználó (beleértve a töröltek) listázása
* *show()*: megjeleníti a paraméterként megadott azonosítójú felhasználó adatit
* *update()*: a paraméterként megadott azonosítójú felhasználó adatait frissíti a kéréshez csatolt értékekre
* *restore()*: a paraméterként megadott azonsítójú felhasználót visszaállítja
* *destroy()*: a paraméterként megadott azonosítójú felhasználót törli (soft delete)
* *create()*: megjeleníti az új felhasználó létrehozásához használatos nézetet
* *store()*: a kéréshez csatolt értékek segítségével új felhasználót hoz létre
* *validator()*: meghatározza a feltételeket, melyeket a kérésekhez csatolt értékeknek teljesíteniük kell

A vezérlő minden beérkezett kérést csak akkor dolgoz fel, ha az *’admin’* middleware átengedte.

Az összes felhasználó listázásánál a törölt elemeket is listázza (*withTrased()*), mert a nézetben lehetőség van azok megjelenítésére is (visszaállítás céljából). Frissítés és törlés esetén csak a nem törölt felhasználók között keres. A visszaállítás természetesen csak a törölt felhasználók között próbálja a megfelelőt megtalálni.

Ahol a megjelenítéshez szükséges, ott kilistázza és mellékeli a lehetséges döntnöki és játékvezetői szinteket.

A felhasználókat az emailcímük azonosítja, ezért már az adatbázis sem enged új felhasználót létrehozni olyan címmel, ami már szerepel egy rekordban (akár töröltben).

Ha a kérés végrehajtása sikertelen, átirányít a megfelelő útvonalra vagy nézetre és mellékel egy hibaüzenetet (*with(’error’,...)*). Sikeres kezelés esetén erről a tényről ad át üzenetet, ahol erre lehetőség van (*with(’message’,...*).

A jelszavak kezelése a következőképpen történik: a jelszavakat nem sima szövegként tárolja a program az adatbázisban, hanem a keretrendszer által biztosított *Hash* osztály *make()* metódusa által kódolva. Természetesen ebből a kódból nem visszafejthető az eredeti jelszó, csak az ellenőrizhető, hogy a megadott jelszóból kialakuló kód azonos-e a tárolttal. Felhasználó frissítése esetén nem kötelező új jelszót megadni, ebben az eseten az eddig tárolt jelszó érvényben marad.



A validator által ellenőrzött feltételek:

* név: kötelező
* email: kötelező, email formátum, (létrehozás esetén) még nincs ilyen az adatbázisban
* jelszó: (létrehozás esetén) kötelező
* döntnöki és játékvezetői szint: kötelező

### VenueController

Elsősorban a csarnok (*’Venue’*) modellel kapcsolatban beérkező kérések feldolgozásáért felel.

Metódusai:

* *index()*: az összes csarnok (beleértve a töröltek) listázása
* *show()*: megjeleníti a paraméterként megadott azonosítójú csarnok adatait
* *update()*: a paraméterként megadott azonosítójú csarnok adatait frissíti a kéréshez csatolt értékekre
* *restore()*: a paraméterként megadott azonosítójú csarnokot visszaállítja
* *destroy()*: a paraméterként megadott azonosítójú csarnokot törli (soft delete)
* *create()*: megjeleníti az új csarnok létrehozásához használatos nézetet
* *store()*: a kéréshez csatolt értékek segítségével új csarnokot hoz létre
* *validator()*: meghatározza a feltételeket, melyeket a kérésekhez csatolt értékeknek teljesíteniük kell

A vezérlő minden beérkezett kérést csak akkor dolgoz fel, ha az *’admin’* middleware átengedte.

Az összes csarnok listázásánál a törölt elemeket is listázza (*withTrased()*), mert a nézetben lehetőség van azok megjelenítésére is (visszaállítás céljából). Frissítés és törlés esetén csak a nem törölt csarnokok között keres. A visszaállítás természetesen csak a törölt csarnokok között próbálja a megfelelőt megtalálni.

A csarnokokat a név, rövid név, cím közül bármelyik mező egyértelműen azonosítja, ezért már az adatbázis sem enged új csarnokot azonos adatokkal rendelkező már létező csarnok esetén.

Ha a kérés végrehajtása sikertelen, átirányít a megfelelő útvonalra vagy nézetre és mellékel egy hibaüzenetet (*with(’error’,...)*). Sikeres kezelés esetén erről a tényről ad át üzenetet, ahol erre lehetőség van (*with(’message’,...*)).

A validator által ellenőrzött feltételek:

* név: kötelező, (létrehozás esetén) egyedi
* rövid név: kötelező, (létrehozás esetén) egyedi
* cím: kötelező, (létrehozás esetén) egyedi
* pályák száma: egész szám típusú

### TournamentController

Elsősorban a verseny (*’Tournament’*) modellel kapcsolatban beérkező kérések feldolgozásáért felel.

Metódusai:

* *index()*: az összes verseny (beleértve a töröltek) listázása
* *show()*: megjeleníti a paraméterként megadott azonosítójú verseny adatait
* *update()*: a paraméterként megadott azonosítójú verseny adatait frissíti a kéréshez csatolt értékekre
* *restore()*: a paraméterként megadott azonosítójú versenyt visszaállítja
* *destroy()*: a paraméterként megadott azonosítójú versenyt törli (soft delete)
* *create()*: megjeleníti az új verseny létrehozásához használatos nézetet
* *store()*: a kéréshez csatolt értékek segítségével új versenyt hoz létre
* *validator()*: meghatározza a feltételeket, melyeket a kérésekhez csatolt értékeknek teljesíteniük kell
* *intervalDate()*: a paraméterként megadott kezdő és befejező dátumot alakítja át intervallum formátumúvá úgy, hogy csak azt az adatot jeleníti meg mindkettőből amelyben már különböznek
* *showCalendar()*: a versenynaptár megjelenítése (arra a versenyszezonra vonatkozóan, amit a kérés *’season’* mezője tartalmaz, vagy az aktuálisra, ha nincs ilyen mező)
  + paraméter nélkül: az összes verseny
  + paraméterrel: azok a versenyek, melyekre a megadott paraméterrel rendelkező azonosítójú felhasználó döntnöki vagy játékvezetői szerepben jelentkezett

A vezérlő minden beérkezett kérést – a *showCalendar()* metódust kivéve – csak akkor dolgoz fel, ha az *’admin’* middleware átengedte. A *showCalendar()* metódushoz az *’auth’* middleware szükséges.



Az összes verseny listázásánál a törölt elemeket is listázza (*withTrased()*), mert a nézetben lehetőség van azok megjelenítésére is (visszaállítás céljából). Frissítés és törlés esetén csak a nem törölt versenyek között keres. A visszaállítás természetesen csak a törölt versenyek között próbálja a megfelelőt megtalálni.

Ahol a megjelenítéshez szükséges, ott kilistázza és mellékeli a csarnokok listáját.

A versenyek listázásánál csak azokat a versenyeket adja vissza, amelyek kezdő dátuma a mai napnál nem régebbi.

Az egyetlen metódus, ahol nem egy vagy több modellt ad vissza, a *showCalendar()*. Ez a függvény az, amit a nem adminisztrátor jogosultságú felhasználók leginkább használnak, mert ezen keresztül lehet (felhasználóra szűrve vagy általánosan) a versenynaptárt megjeleníteni. A kért versenyszezon meghatározása után (minden szezon szeptember 1. és augusztus 31. között tart) a szezon összes versenyét feldolgozza, felhasználóra szűrt versenynaptár esetén kigyűjtve hozzájuk a döntnöki és játékvezetői jelentkezéseket. A feldolgozás során új adatszerkezetet állít elő, amely tartalmazza a megjelenítéshez szükséges mezőket. Elegendő lenne itt is modelleket átadni a nézetnek, de igyekeztem a nézetekben a lehetséges legkevesebb logikát elhelyezni és csak a megjelenítésre koncentrálni.

A *showCalendar()* a *$tournament* változót a következő lekérdezéssel tölti fel:



Ha a kérés végrehajtása sikertelen, átirányít a megfelelő útvonalra vagy nézetre és mellékel egy hibaüzenetet (*with(’error’,...)*). Sikeres kezelés esetén erről a tényről ad át üzenetet, ahol erre lehetőség van (*with(’message’,...*).

A validator által ellenőrzött feltételek:

* megnevezés: kötelező
* kezdő dátum: kötelező, dátum formátum, nem későbbi a befejező dátumnál
* befejező dátum: kötelező, dátum formátum, nem korábbi a kezdő dátumnál
* csarnok: kötelező
* igényelt játékvezetők: egész szám típusú

### ApplicationController

A döntnöki és játékvezetői jelentkezés (*UmprieApplication* és *RefereeApplication*) modellekkel kapcsolatban beérkező kérések feldolgozásáért felel.

Metódusai:

* *addUmpire()* és *addReferee()*: új jelentkezés
* *removeUmpire()* és *removeReferee()*: meglévő jelentkezés törlése
* *show()*: a paraméterként megadott azonosítójú versenyre való döntnöki és játékvezetői jelentkezések listázása
* *store()*: a paraméterként megadott azonosítójú versenyekre való döntnöki és játékvezetői jelentkezések feldolgozott és jóváhagyott státuszainak frissítése
* *add()* és *remove()*: a kétféle (döntnöki és játékvezetői) művelet olyan kevés eltéréssel renelkezik, hogy a megvalósításuk közös függvényben történik, a *’type’* paraméter segítségével, melynek lehetséges értékei: *’umpire’* vagy *’referee’*.

A vezérlő minden beérkezett kérést csak akkor dolgoz fel, ha az *’auth’* middleware átengedte.

Ha a kérés végrehajtása sikertelen, átirányít a megfelelő útvonalra vagy nézetre és mellékel egy hibaüzenetet (*with(’error’,...)*). Mivel a jelentkezések hozzáadása vagy törlése esetén is az előző nézet jelenik meg újra, sikeres végrehajtás esetén nincs üzenet.

### EmailController

Az értesítő email kiküldéséért felel.

Metódusa:

* *sendmail()*: a paraméterként megadott azonosítójú versenyre kiküldi a tájékoztató emailt azoknak a felhasználóknak, akiknek a versenyre feldolgozott és jóváhagyott státuszú döntnöki vagy játékvezetői jelentkezésük van

A vezérlő minden beérkezett kérést csak akkor dolgoz fel, ha az *’admin’* middleware átengedte.

Az egyetlen *sendmail()* metódus kigyűjti a szükséges modelleket az adatbázisból és átadja a keretrendszer által biztosított *Mail::send()* metódusnak. Az email sablonja egy nézetben HTML formátumban található meg, amelynek neve a metódus első paramétere.

## Nézetek

### Általános információ

#### Űrlap metódusok módosítása

A HTML űrlapok metódusa csak GET vagy POST lehet. Emiatt amikor PUT, PATCH vagy DELETE útvonalakra szeretnénk a kérést irányítani, be kell állítanunk egy rejtett változót, melynek neve ’\_method’:



A keretrendszer segédfüggvényt is biztosít erre a célra, így a fenti sor helyettesíthető:



#### Cross-site request forgery

Azért, hogy bejelentkezett felhasználó nevében kívülről ne lehessen akár rosszindulatú kérést a szerverhez eljuttatni, a keretrendszer minden munkamenethez generál egy tokent, ami azonosítja. Ezt minden űrlapba szükséges mellékelni, mert enélkül a kérés mindig sikertelen lesz.

Természetesen ehhez is biztosít segédfüggvényt a keretrendszer:



A segédfüggvény egy rejtett szövegbeviteli mezőt helyez az űrlapba, amely tartalmazza a munkamenethez tartozó tokent.

#### Validáció

Bár az űrlapoknál a weboldal használja a HTML biztosította lehetőségeket (’autocomplete’, ’required’, ’numeric’, stb.), a vezérlők minden esetben amellett, hogy minden szükséges mező értéke átadásra került-e, ellenőrzik azt is, hogy az értékek teljesítik-e a szükséges előfeltételeket. Erre a célra a keretrendszer a *’Validator’* osztályt biztosítja, amelynek a vezérlő átadja a kérés adatait. Ha valamelyik mező nem teljesíti az összes feltételt, visszairányít arra az oldalra, ahonnan a kérés érkezett, és beállítja az *’error’* változót. Az azonos nevű segéddel pedig alkalmazható a Bootstrap érvénytelen mezőkre létrehozott osztálya. A feltételt nem teljesítő értéket az *’old’* segéddel érhetjük el, a hibaüzenet pedig a *$message* változóban lesz.

Például új felhasználó létrehozása esetén az email cím beviteli mező a következő:



#### Tájékoztató üzenetek

A vezérlők azon műveletek után, melyek a listázó nézetre irányítanak, a *with()* metódus segítségével a munkamenethez hozzáadnak egy elemet, melynek neve sikeres művelet esetén *’message’*, sikertelen esetén pedig *’error’*. Ha bármelyik elem létezik, akkor már a *’layouts*.*app’* nézet megjeleníti az üzenetet, így a menüsor alatt biztosan ez lesz az oldal tetején. A hibaüzenetek a Bootstrap *’alert-danger’* osztályának segítségével piros, az egyéb üzenetek pedig az *’alert-success’* osztály hatására zöld háttérrel jelennek meg.

Például a *TournamentContoller.php store()* metódusának utolsó utasítása:



Az *’error’* felhasználása a *layouts/app.blade.php* fájlban:



#### Lokalizáció

Annak érdekében, hogy a weboldal akár más országokban is felhasználható legyen hasonló célra, az email sablon kivételével minden a felhasználó számára megjelenő üzenet angol nyelven van. A keretrendszer biztosítja a lehetőséget, hogy a helyi igényeknek megfelelő üzeneteket jeleníthessünk meg. A *config/app.php* file tartalmazza a *’”locale” => ”hu”’* beállítást. Ez alapján a keretrendszer, a *’\_\_(<szöveg>)’* kifejezés kiértékelése közben ellenőrzi, hogy a *resources/lang/hu.json* fájl tartalmazza-e kulcsként a ’<szöveg>’ karakterláncot és ha igen, akkor a neki megfelelő értékkel helyettesíti.

A validáció során keletkező hibaüzenetek fordítása a *resource/lang/hu/validation.php* fájlban található.

### Blade sablonok

A Blade egy PHP sablon-motor, amit a Laravel keretrendszer biztosít. Előnye a modularitása mellett az, hogy van benne lehetőség egyszerű PHP kódot is írni a sablonokba.

Minden Blade nézet fájl kiterjesztése *’.blade.php’*, a *resources/views* mappában vannak.

Az értesítő email sablonja kivételével minden nézet ’szülője’ a *’layouts/app’* nézet. Ez tartalmazza a HTML oldal head szekcióját és a body alapjait. Itt van az oldal tetején található menü leírása (ami csak akkor jelenik meg, ha bejelentkezett felhasználóként böngészünk, amit az *@auth* segéd ellenőriz).

A ’gyerek’ nézetek mindegyike az *@extends(layouts.app)* direktívával kezdődik. Ezután definiálják a *’content’* nevű szekciót a *@yield()* direktívával. Ebből kifolyólag a nézetek nem tartalmazzák egy teljes HTML oldal kódját, csak az *app.blade.php* ’*content’* szekcióját.

A felső menüsor csak bejelentkezett felhasználó számára jelenik meg, mert a weboldal nem tartalmaz olyan részt, ami egy vendég felhasználó számára megjeleníthető lenne.

Bejelentkezett felhasználó esetén a menüsor bal oldalán a ’Versenynaptár’ (átirányít a *’calendar’* útvonalra) és a ’Jelentkezéseim’ (átirányít a *’calendar/{id}’* útvonalra) pontokból választhat. Ha adminisztrátor jogosultsággal rendelkezik, akkor a menüsor jobb oldalán az ’Admin’ lenyíló menüből is választhat, a ’Felhasználók’, ’Csarnokok’, ’Versenyek’ menüpontok közül.

A nézetek elrendezésének alapja a Bootstrap rácsvonal elrendezése (grid layout). Ennek segítségével definiáltam saját CSS osztályokat, hogy a különböző, de hasonló nézetek egységes külsővel rendelkezzenek. Ezeket a *resources/sass/app.scss* file tartalmazza, melyből az *npm* generálja a *css/app.css* fájlt, amit az *’layouts.app’* nézet header szekciójában beilleszt.

A keretrendszer kényelmesen biztosítja a szokásos programkonstrukciókat, többek között az *@if-@elseif-@else-@endif* és a *@foreach-@endforeach* segédekkel.

A nézetnek átadott változók kiértékelésének módja: {{ $változónév }}.

### Felhasználó nézetek (*user* mappa)

#### Listázás (*list.blade.php*)

Ez a nézet a *’layouts.app’* nézetet egészíti ki, a *’content’* szekció definiálásával.

A nézet tetején a bal oldalon az *’Új felhasználó’* gomb található, mely átirányít a *’users.create’* útvonalra. Ugyanebben a sorban a jobb oldalon a ’*Töröltek megjeleníése’* jelölőnégyzet helyezkedik el, melynek állapota törlés és visszaállítás esetén nem változik.



Bejelölt állapot esetén a törölt felhasználók is látszanak, bejelöletlen esetben is tartalmazza őket az oldal, de nem jelennek meg (*class=”d-none”*).

Ezután a felhasználókat listázó táblázat fejléce következik, melynek oszlopai: *’Név’, ’Email cím (borítékkal jelölve)’, ’Jv (játékvezetői) szint’, ’D (döntnöki) szint’, ’Admin’*.

A nézet paraméterként megkapja a vezérlőtől az összes felhasználót tartalmazó kollekciót (beleértve a törölteket is), melyen végigiterálva azokat soronként jeleníti meg. Ha felhasználó adminisztrátor jogosultsággal rendelkezik, akkor az *’Admin’* oszlopban egy pipa jelenik meg, egyébként az az oszlop üres. A fejléc nélküli oszlopok közül az elsőben szerkesztés ikon jelenik meg (ami átirányít a *’users.show’* útvonalra), de csak akkor, ha a felhasználó nem törölt. Az utolsó oszlop tartalma is attól függ, hogy a felhasználó törölt-e. Ha igen, akkor visszállíáts ikon (átirányít a *’users.restore’* útvonalra), ha nem, akkor törlés ikon (átirányít a *’users.delete’* útvonalra) jelenik meg. Ezeket a műveleteket egy-egy rejtett űrlap segítségével valósítjuk meg, melynek a *submit()* metódusát az ikonra való kattintás aktiválja. (Megjegyzés: azért, hogy a felhasználó a saját fiókját ne kísérelje meg törölni, abban a sorban ez az oszlop mindig üres marad.)

#### Szerkesztés (*edit.blade.php*)

Ez a nézet a *’layouts.app’* nézetet egészíti ki, a *’content’* szekció definiálásával.

A nézet kettős célt szolgál: új felhasználó létrehozása vagy létező felhasználó szerkesztése. A kettő között a nézet az *isset($user)* feltétel ellenőrzésével dönt.

A felépítés: cím, űrlap (táblázatos elrendezésben), gombok. A cím a fent említett feltételtől függően *’Új felhasználó’* vagy *’Felhasználó szerkesztése’*. Az űrlap létrehozás esetén a *’/users’* útvonalra irányít (POST kéréssel), szerkesztéskor pedig a *’*/*users/{id}’* útvonalra (PUT kéréssel). A beviteli mezők értékei vagy üresek (létrehozás) vagy tartalmazzák a szerkesztett felhasználó jelenleg mentett adatait (szerkesztés). Ez alól kivétel a jelszó, mely mezőt ha üresen hagyunk, a jelenlegi jelszó változatlan marad. Ha mégis meg kívánjuk változtatni, lehetőség van tetszőleges jelszót megadni, vagy a mező melletti gombra kattintva egy 8 karakter hosszú alfanumerikus véletlen generált karakterláncot választani. A játékvezetői és döntnöki szintek legördölő választógombok, melyek lehetséges értékei a vezérlőtől kapott *$umpire\_levels* és *$referee\_levels* kollekciók tagjai.

A nézet alján csak szerkesztés esetén jelenik meg a *’Visszaállítás’* gomb, mely az űrlap minden mezőjét az eredeti értékére változtatja. Mindkét esetben megjelenik a *’Mégsem’* (mentés nélkül a *’users.index’* útvonalra irányít) és a *’Mentés’* vagy *’Létrehozás’* gomb (elküldi az űrlapot).

### Csarnok nézetek (*venue* mappa)

#### Listázás (*list.blade.php*)

Ez a nézet a *’layout.app*’ nézetet egészíti ki, a *’content’* szekció definiálásával.

A nézet tetején a bal oldalon az ’Új csarnok’ gomb található, mely átirányít a *’venues.create’* útvonalra. Ugyanebben a sorban a jobb oldalon a ’Töröltek megjelenítése’ jelölőnégyzet helyezkedik el. Bejelölt állapot esetén a törölt csarnokok is látszanak, bejelöletlen esetben is tartalmazza őket az oldal, de nem jelennek meg (class=”d-none”). A jelölőnégyzet csak akkor látható, ha szerepel az adatbázisban legalább egy csarnok.

Ezután a csarnookat listázó táblázat fejléce következik, melynek oszlopai: ’Név’, ’Rövid név’, ’Cím’, ’Pályák száma’.

A nézet paraméterként megkapja a vezérlőtől az összes csarnokot tartalmazó kollekciót (beleértve a törölteket is), melyen végigiterálva azokat soronként jeleníti meg. A fejléc nélküli ooszlopok közül az elsőben szerkesztés ikon található (ami átirányít a *’venues.show’* útvonalra), de csak akkor, ha a csarnok nem törölt. Az utolsó oszlop tartalma is attól függ, hogy a csarnok törölt-e. Ha igen, akkor visszaállítás ikon (átirányít a *’venues.restore’* útvonalra), ha nem, akkor törlés ikon (átirányít a *’users.delete’* útvonalra) jelenik meg. Ezeket a műveleteket egy-egy rejtett űrlap segítségével valósítjuk meg, melynek a *submit()* metódusát az ikonra való kattintás aktiválja.

#### Szerkesztés (*edit.blade.php*)

Ez a nézet a *’layouts.app’* nézetet egészíti ki a *’content’* szekció definiálásával.

A nézet kettős célt szolgál: új csarnok létrehozása vagy létező csarnok szerkesztése. A kettő között a nézet az *isset($venue)* feltétel ellenőrzésével dönt.

A felépítés: cím, űrlap (táblázatos elrendezésben), gombok. A cím a fent említett feltételtől függően ’Új csarnok’ vagy ’Csarnok létrehozása’. Az űrlap létrehozás esetén a *’/venues’* útvonalra irányít (POST kéréssel), szerkesztéskor pedig a *’/venues/{$id}’* útvonalra (PUT) kéréssel. A beviteli mezők értékei vagy üresek (létrehozás) vagy tartalmazzák a szerkesztett csarnok jelenleg mentett adatait (szerkesztés).

A nézet alján csak szerkesztés esetén jelenik meg a ’Visszaállítás’ gomb, mely az űrlap minden mezőjét az eredeti értékére változtatja. Mindkét esetben megjelenik a ’Mégsem’ (mentés nélkül a *’venues.index’* útvonalra irányít) és a ’Mentés’ vagy ’Létrehozás’ gomb (elküldi az űrlapot).

### Verseny nézetek (*tournaments* mappa)

#### Listázás (*list.blade.php*)

Ez a nézet a *’layouts.app’* nézetet egészí ki, a *’content’* szekció definiálásával.

A következő sorban a bal oldalon az ’Új verseny’ gomb található, mely átirányít a tournaments.create útvonalra. Ugyanebben a sorban a jobb oldalon a ’Töröltek megjelenítése’ jelölőnégyzet helyezkedik el. Bejelölt állapot esetén a törölt versenyek is látszanak, bejelöletlen esetben is tartalmazza őket az oldal, de nem jelennek meg (class=”d-none”). A jelölőnégyzet csak akkor látható, ha szerepel az adatbázisban legalább egy verseny.

Ezután a versenyeket listázó táblázat fejléce következik, melynek oszlopai: ’Név’, ’Kezdő dátum’, ’Befejező dátum’, ’Csarnok’, ’Igényelt játékvezető’.

A nézet paraméterként megkapja a vezérlőtől az összes versenyt (beleértve a törölteket is) úgy, hogy minden verseny tartalmazza a csarnokot, ahol a verseny lesz (*with()*). A kollekción végigiterálva a versenyeket soronként jeleníti meg. (A ’Csarnok’ oszlopban a csarnok rövid neve látszik.) Az ’Igényelt játékvezető’ oszlopban amellett, hogy a verseny kiírója hány fő játékvezetőt kért a versenyre, szerepel két ikon, az első az applications.show útvonalra irányít (jelentkezések szerkesztése), a második pedig a sendmail útvonalra (tájékoztató email kiküldése). A fejléc nélküli oszlopok közül az elsőben a szerkesztés ikon található (ami átirányít a *’tournaments.show’* útvonalra), de csak akkor, ha a verseny nem törölt. Az utolsó oszlop tartalma is attól függ, hogy a verseny törölt-e. Ha igen, akkor visszaállítás ikon (átirányít a *’tournaments.restore’* útvonalra), ha nem, akkor törlés ikon (átirányít a *’tournaments.delete’* útvonalra) jelenik meg. Ezeket a műveleteket egy-egy rejtett űrlap segítségével valósítjuk meg, melynek a *submit()* metódusát az ikonra való kattintás aktiválja.

#### Szerkesztés (*edit.blade.php*)

Ez a nézet a *’layouts.app’* nézetet egészíti ki a *’content’* szekció definiálásával.

A nézet kettős célt szolgál: új verseny létrehozása vagy létező verseny szerkesztése. A kettő között a nézet az *isset($tournament)* feltétel ellenőrzésével dönt.

A felépítés: cím, űrlap (táblázatos elrendezésben), gombok. A cím a fent említett feltételtől függően ’Új verseny’ vagy ’Verseny szerkesztése’. Az űrlap létrehozás esetén a *’/tournaments’* útvonalra irányít (POST kéréssel), szerkesztéskor pedig a *’/tournaments/{id}’* útvonalra (PUT kéréssel). A beviteli mezők értékei vagy üresek (létrehozás) vagy tartalmazzák a szerkesztett verseny jelenleg mentett adatait (szerkesztés). A csarnok legördülő választógomb, melynek lehetséges értékei a vezérlőtől kapott *$venues* kollekció tagjai.

#### Jelentkezések kezelése (*applications.blade.php*)

Ez a nézet a *’layouts.app’* nézetet egészíti ki a *’content’* szekció definiálásával.

A nézet célja, hogy egy versenyre beérkezett jelentkezések közül az adminisztrátor felhasználó kiválaszthassa, hogy melyeket hagyja jóvá és ezek közül mely döntései véglegesek.

Legfölül az alábbi felirat látható: ’Jelentkezések - <verseny neve>’. Ha a szöveg fölé visszük az egeret, megjelenik a kedző és a befejező dátum, a helyszínül szolgáló csarnok rövid neve, valamint az igényelt játékvezetők száma.

Ezután két szekció következik, melyek ugyanannak az űrlapnak a részei (táblázatos elrendezésben): ’Döntnök jelentkezések’ és ’Játékvezető jelentkezések’. Ha valamelyik szekcióban egy elem sincs, azt a címével egy sorban üzenet jelzi. A beérkezett jelentkezések soronként jelennek meg, egy sorban a felhasználó neve és két jelölőnégyzet kerül, ’feldolgozva’ és ’jóváhagyva’.

A nézet alján három gomb jelenik meg: ’Visszaállítás’ (az oldalon elvégzett változtatások visszavonása), Vissza’ (átirányít a *’tournaments.index’* útvonalra) és ’Mentés’ (elküldi az űrlapot az *’applications.store’* útvonalra).

#### Versenynaptár (*calendar.blade.php*)

Ez a nézet a *’layouts.app’* nézetet egészíti ki a *’content’* szekció definiálásával.

A nézet kettős célt szolgál a versenynaptár egy szezonjának megjelenítésére: az összes verseny megjelenítése vagy csak azon versenyeké, amelyekre a felhasználó jelentkezett. A kettő közül a vezérlő által átadott *$filtered* logikai változó értéke alapján választ.

A megjelenő cím ’Versenynaptár’ (összes nézet) vagy ’Jelentkezéseim’ (szűrt nézet).

A következő sorban annak a vesenyszezonnak a kezdőéve látható, amelynek a versenyeit a nézet megjeleníti. Az ’előző’és ’következő’ gombokkal változtathatjuk a megjelenített versenyszezont.

Ha a szezonhoz az adatbázisban nem tartozik verseny (összes nézet) vagy a szezonban egy versenyre sem jelentkezett a felhasználó (szűrt nézet) a ’Nincs megjeleníthető verseny’ felirat jelenik meg.

Ellenkező esetben a versenyeket táblázatos formában jelenenítjük meg. A táblázat fejléce a következőket tartalmazza: ’Dátum’, ’Elnevezés’, ’Helyszín’, ’Igényelt játékvezető”, ’Játékvezető jelentkezések’, ’Döntnök’, ’Játékvezető’. Az utolsó két elem akkor jelenik meg, ha a felhasználónak van döntnöki vagy játékvezetői minősítése.

A nézet a vezérlőtől megkapja az összes versenyt, amelyeken végigiterálva azokat soronként jeleníti meg.

A ’Helyszín’ oszlopban a csarnok rövid neve látható, ha az egeret fölévisszük, megjelenik a név és a cím is. A ’Játékvezető jelentkezések’ oszlopban azon felhasználók száma kerül, akik játékvezetőként jelentkeztek a versenyre. Adminisztrátor felhasználóknak ha a kurzor a szám fölött van, megjelenik a jelentkezettek névsora.

A ’Döntnök’ oszlop csak akkor nem üres, ha a felhasználó rendelkezik döntnöki minősítéssel. Ekkor a következő négy eset lehetséges:

* nem jelentkezett még a versenyre: nagy piros X és mellette egy kicsi zöld pipa

(ekkor a pipára kattintva tud jelentkezni a versenyre, az átirányítás a *’referee’* útvonalra egy rejtett űrlap segítségével történik

* jelentkezett a versenyre: nagy zöld pipa és mellette egy kicsi piros X

(ekkor az X-re kattintva vissza tudja vonni a jelentkezését, az átirányítás a *’referee’* útvonalra (DELETE kéréssel) egy rejtett űrlap segítségével történik)

* jelentkezett a versenyre és a jelentkezése feldolgozott és elfogadott státuszú: nagy zöld pipa
* jelentkezett a versenyre, a jelentkezése feldolgozott és elutasított státuszú: nagy piros X

Megjegyzés: szűrt nézetben jelentkezni nincs , csak a feldolgozott státuszú jelentkezések megtekintésére és a még nem feldolgozott jelentkezések visszavonására van lehetőség.

A ’Játékvezető’ oszlop analóg módon működik.

### Email nézet (*emails* mappa)

#### Értesítés (*notify.blade.php*)

A nézet a versenyre kiválasztott és jóváhagyott döntnökök és játékvezetők számára kiküldendő email váza.

### Hiba nézetek (*errors* mappa)

Ha valamelyik *abort()* függvény hatására a kiszolgáló kivételt dob, a keretrendszer elkapja és a HTTP válaszkód alapján választja ki, melyik nézetet jelenítse meg (ha az a nézet létezik).

A vezérlők elsősorban 500-as válaszkóddal térnek vissza hiba esetén, ezért ezt és a leggyakrabban szükséges 404-es nézetet módosítottam az alapértelmezettől eltérőre. Mivel ezek a nézetek kivételkezelés útján érhetők el, a segédfüggvények nem használhatók, ezért a menüsor nem jelenik meg, a hozzáadott ’Versenynaptár’ hivatkozással tud a felhasználó a normál használathoz visszatérni.

# Tesztelés

Manuális tesztesetek:

## Új felhasználó létrehozása

* felhasználók listázása: kattintsunk az ’Új felhasználó’ gombra
* töltsük ki a beviteli mezőket a következő adatokkal:
  + Név: Cserni Mária
  + Email cím: [cserni.maria@gmail.com](mailto:cserni.maria@gmail.com)
  + Jelszó:
    - kattintsunk a mező melletti ikonra
      * véletlen generált jelszó jelenik meg a mezőben
  + Játékvezető szint: országos
  + Döntnöki szint: nincs
* kattintsunk a ’Létrehozás’ gombra
  + a ’felhasználók listázása’ nézet’ jelenik meg
  + a nézet tetején zöld háttérrel üzenet a létrehozás sikerességéről
  + az adatbázis ’users’ táblájában új sor jött létre a megadott adatokkal

## Felhasználó törlése és visszaállítása

* felhasználók listázása: kattintsunk egy felhasználó sorában a ’Törlés’ ikonra
  + a felhasználó sora eltűnik
  + az adatbázisban a megfelelő rekord ’deleted\_at’ mezője nem NULL
* jelöljük be a ’Töröltek megjelenítése’ jelölőnégyzetet
  + a törölt felhasználó sora megjelenik, nincs ’Szerkesztés’ gomb, ’Törlés’ helyett ’Visszaállítás’ gomb van
* kattintsunk a ’Visszaállítás’ gombra
  + a jelölőnégyzet bejelölt maradt
  + az előbb törölt felhasználó sorában megjelenik a ’Szerkesztés’ gomb
  + az adatbázisban a megfelelő rekord ’deleted\_at’ mezője NULL

## Csarnok szerkesztése

* csarnokok listázása oldal: kattintsunk az egyik sorban a ’Szerkesztés’ gombra
  + megjelenik a ’Csarnok szerkesztése’ nézet
* a pályák száma ne legyen szám
* kattintsunk a ’Mentés’ gombra
  + az oldalon maradunk, üzenet a pályák száma kapcsán
* kattintsunk a ’Visszaállítás’ gombra
  + a csarnok eredeti adatai kerülnek a mezőkbe
* változtassunk legalább egy mezőn
* kattintsunk a ’Mégsem’ gombra
  + megjelenik a ’Csarnokok listázása’ nézet, a csarnok adatai nem változtak
* nyissuk meg újra a ’Szerkesztés’ nézetet
* változtassunk a mezőkön és kattintsunk a ’Mentés’ gombra
  + megjelenik a ’Csarnokok listázása’ nézet
  + az nézet tetején zöld háttérrel üzenet a frissítés sikerességéről
  + az adatbázisban a csarnok adatai frissültek (az időbélyegek is)

## Verseny szerkesztése

* versenyek listázása oldal: kattintsunk a ’Szerkesztés’ gombra
  + megjelenik a ’Verseny szerkesztése’ nézet
* állítsuk a kezdő dátumot a befejező dátumnál későbbire
* változtassunk más mezőket is
* kattintsunk a ’Mentés’ gombra
  + az oldalon maradunk, a dátumoknál hibaüzenet
  + minden más mezőben a már megváltoztatott érték van
* kattintsunk a ’Mégsem’ gombra
  + megjelenik a ’Versenyek listázása’ nézet
  + a verseny adatai nem változtak

## Jelentkezés versenyre

* versenynaptár oldal: jelentkezzünk egy olyan versenyre, ahova eddig nem
  + az ikon megváltozik
  + az adatbázisban az ’umpire\_applications’ táblában új sor jön létre
  + a ’Jelentkezések kezelése’ nézetben látszik a jelentkezés (üres jelölőnégyzetekkel)
* jelentkezzünk még egy versenyre
* töröljük a jelentkezést
  + az ikon az újra az eredeti
  + az adatbázisban új sor van, a ’deleted\_at’ mező nem NULL
  + a ’Jelentkezések kezelése’ nézetben a jelentkezés nem látszik

## Jelentkezések kezelése

* versenynaptár nézet: jelentkezzünk egy versenyre
* versenyek listázása: kattintsunk a ’Jelentkezések kezelése’ ikonra egy olyan versenynél, ahova jelentkeztünk
  + látjuk a jelentkezést üres jelölőnégyzetekkel
* állítsuk át a jelentkezések jelölőnégyzeteit
  + ha a ’feldolgozott’ bejelölt, akkor a ’jóváhagyott’ inaktív
* kattintsunk a ’Mentés’ gombra
  + megjelenik a ’versenyek listázása’ nézet
  + a versenynaptárban változott a jelentkezés állapota
  + ha ’feldolgozott’, akkor már nem lehet változtatni rajta

## Értesítő email kiküldése

* versenyek listázása ol dal: kattintsunk az ’Információs email kiküldése’ ikonra
  + a nézet tetején zöld háttérrel üzenet a sikeres küldésről
  + a storage/logs mappában a mai log file-ban ott vannak az emailek
  + olyan felhasználó kapott csak emailt, akinek a jelentkezése feldolgozott és elfogadott

# Továbbfejlesztési lehetőségek

* nemzetközi versenyek és jelentkezések kezelése
* a játékvezetőknek szükséges dokumentumsablonok letöltési lehetősége
* szabályzatok, ajánlások elérhetősége
* a versenyekről beérkezett döntnöki jelentések tárolása
* munkanyilvántartás (record of work)
* jelszó megváltoztatása adminisztrátor közreműködése nélkül
* játékvezetői és döntnöki jelentkezés adminisztrátor általi hozzáadása
* fórum a szakmai konzultációra
* időszakos hírlevél küldése

# Összegzés

A Magyar Tollaslabda Szövetség Játékvezetői és Versenybírói Bizottságának feladata összetett és sokrétű. A legnehezebb feladat a versenybírókkal való kapcsolattartás mellett a versenyidőszak (kb. szeptembertől májusig) során szinte hetente előforduló ’A’ kategóriás versenyekre a megfelelő létszámú és minősítésű döntnök(ök) és játékvezetők kiválasztása. Ehhez elengedhetetlen, hogy a lehetséges résztvevők időben értesüljenek azokról a versenyekről, melyeken szerepet vállalhatnak. Ugyanakkor a Bizottság elnökének jelentős adminisztrációs terhet okoz, hogy a jelentkezéseket kezelje és nyilvántartsa, hiszen ha egy megosztott táblázatot használnak, akkor érdemes az írási jogot nem mindenkinek megadni, a véletlen hibák – főleg törlések – elkerülése végett.

Szakdolgozatom annak az önálló weboldalnak az alapjait valósítja meg, ami ezt az terhet legalább részben leveszi az elnök válláról. A versenynaptár és a jelentkezések kezelésével a versenybírók szinte beavatkozás nélkül tudják karbantartani a saját időbeosztásukkal egyeztetett szakmai munkájuk tervét.

Azzal, hogy bármilyen információhoz csak bejelentkezés után fér hozzá a felhasználó biztosítja, hogy illetéktelenek nem férnek hozzá rájuk nem tartozó információhoz (pl. melyik versenyre pontosan hányan és kik jelentkeztek).

Ez a program jó kiindulópont egy olyan weboldalhoz, ami felhőszolgáltatóktól függetlenül online felületet biztosít a Bizottság és a versenybírók kommunikációjához és információcseréjéhez.

(Megjegyzés: a készítés idején a szerző a Bizottság elnöke, első kézből látja a program előnyeit és felhasználhatóságát; a Bizottság legutóbbi ülésén a bemutatott működés pozitív fogadtatásra talált.)