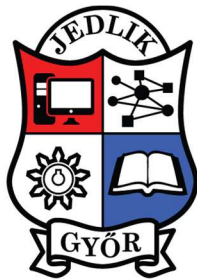




győri szakképzési centrum

Jedlik Ányos
Gépipari és Informatikai
Technikum és Kollégium



9021 Győr, Szent István út 7.

+36 (96) 529-480

+36 (96) 529-448

OM: 203037/003

jedlik@jedlik.eu

www.jedlik.eu

Záródolgozat feladatkiírás

Tanulók neve: Tankovits Bence, Szebik Levente Róbert
Képzés: nappali munkarend
Szak: 5 0613 12 03 Szoftverfejlesztő és tesztelő technikus

A záródolgozat címe:

FootTour

Konzulens: Bognár Pál
Beadási határidő: 2022. 04. 29.

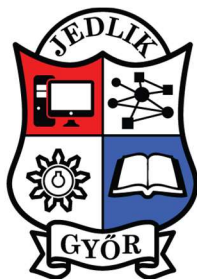
Győr, 2021. 10. 01

Módos Gábor
igazgató



győri szakképzési centrum

Jedlik Ányos
Gépipari és Informatikai
Technikum és Kollégium



9021 Győr, Szent István út 7.

+36 (96) 529-480

+36 (96) 529-448

OM: 203037/003

jedlik@jedlik.eu

www.jedlik.eu

Konzultációs lap

	A konzultáció		Konzulens aláírása
	ideje	témája	
1.	2022.02.15.	Témaválasztás és specifikáció	
2.	2022.03.14.	Záródolgozat készültségi fokának értékelése	
3.	2022.04.17.	Dokumentáció véglegesítése	

Tulajdonosi nyilatkozat

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

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

Győr, 2022. április 28.

Tanulók aláírásai:

Tankovits Bence

Szebik Levente Róbert

Jedlik Ányos Gépipari és Informatikai Technikum és Kollégium

FootTour dokumentáció

Készítették:

Tankovits Bence

Szebik Levente Róbert

Tartalom

1	Bevezetés.....	6
1.1	A probléma és megoldás.....	6
1.2	Tervezés.....	6
1.3	Továbbfejlesztés	7
1.4	Feladat megosztás.....	8
2	A program	10
2.1	Felhasznált technológiák	10
2.2	Az adatbázis.....	10
2.2.1	Tervezés	10
2.2.2	Relációs adatmodell	11
2.2.3	Táblák.....	11
2.2.4	Csatlakozás az adatbázishoz.....	15
2.3	Backend felépítése.....	15
2.4	API dokumentáció	16
2.4.1	DrawController.....	16
2.4.2	EventController	16
2.4.3	GroupController	16
2.4.4	MatchController	17
2.4.5	PlayerController	18
2.4.6	TeamController	19
2.4.7	TeamstoGroupsController.....	19
2.4.8	TournamentController.....	20
2.4.9	UserController.....	22
2.5	Frontend.....	23
2.5.1	Felépítése.....	23
2.5.2	Reszponzív webalkalmazás.....	24

2.6	Autentikáció, biztonság	25
2.6.1	JWT	25
2.6.2	Autentikáció menete.....	25
2.6.3	Jelszavak.....	26
2.6.4	Felhasználó típusok	26
2.7	Tiszta kód	27
3	Tesztek	27
4	Felhasználói kézikönyv	27
4.1	Navigációs sáv	27
4.1.1	Elérhető tornák	28
4.1.2	Tornáim	29
4.2	Regisztráció	30
4.3	Bejelentkezés	30
4.4	Tornák.....	31
4.4.1	Létrehozás	31
4.4.2	Részletes leírás 1	32
4.4.3	Részletes leírás 2	32
4.4.4	Sorsolás	33
4.4.5	Csapat regisztráció tornára	33
4.4.6	Menetrend.....	34
4.4.7	Díjak.....	34
4.5	Mérkőzések.....	35
4.5.1	Játékvezetői jegyzőkönyv	35
4.5.2	Jegyzőkönyv	36
4.6	Fejlesztői futtatási kézikönyv	37

1 Bevezetés

1.1 A probléma és megoldás

A FootTour egy online kispályás labdarugó tornák szervezését, lebonyolítását és azokra történő jelentkezést segítő webalkalmazás. Az ilyen eseményeket nagy érdeklődés övezi, azonban ezek nem minden esetben jutnak el az érdeklődőkhöz. A tornákat általában a szervezők saját közösségi média profiljukon osztják meg. Szerettünk volna alkotni egy egységes platformot ezeknek a tornáknak a hirdetéséhez.

A program négy problémára nyújt megoldást egyszerre:

- megkönnyíti a szervezők számára a tornák hirdetését, torna lebonyolításának menetét
- a labdarugók számára egyszerű felületet biztosít a számukra megfelelő torna megtalálásához és az arra való jelentkezéshez
- adminisztrációs felület az eredmények rögzítéséhez
- egységes gyűjtőfelületet biztosít

A weboldal bárki számára elérhető, aki rendelkezik internetkapcsolattal és egy olyan eszközzel, amely képes hálózati kapcsolat létesítésére. A FootTour minden képernyőfelbontásra optimalizált így egyaránt használható mobil eszközön, tableten, laptopon, valamint asztali számítógépen. Az egész alkalmazás böngészőn belül fut, így nem igényel semmiféle eszközön letöltést és telepítést.

1.2 Tervezés

Ebben a pontban a tervezés fázis menetét és az abban megfogalmazott ötleteket fejtjük ki.

A fejlesztésnek ez a része körülbelül kettő hetet ölelt fel. Ez idő alatt megfogalmaztuk az elsődleges céljainkat, amelyeket mindenképpen szerettünk volna megvalósítani. Majd ezek után azokat a funkciókat, amik a program működésének szempontjából nem létfontosságúak azonban szerettük volna megvalósítani. Ezeknek az ötleteknek a rögzítéséhez a Trello által biztosított platformot használtuk.

Fontos volt, hogy több szemszögből is megközelítsük a problémát. Mind a játékosok, szervezők és játékvezetők érdekeit szemelőt tartottuk. Minden felhasználó számára egy letisztult, könnyen kezelhető felületet akartunk létrehozni.

A jelentkezni vágyó labdarúgók számára az elsődleges célunk a hatékony böngészés és a gyors jelentkezés biztosítása volt. Valamint a tornákon az eredmények követésének lehetősége, mivel ezek általában csak papíron, a helyszín egy adott pontján tekinthetők meg és legtöbbször nem is naprakészek. A tornaszervezők válláról szeretnénk volna a menetrend kialakításának terhet levenni, ezt a program elvégzi helyettük. A játékezőknek pedig egy könnyen kezelhető platformot a mérkőzések jegyzőkönyvének megírásához.

1.3 Továbbfejlesztés

A tervezési fázisban rengeteg ötletünk támadt, amiket aztán skáláztunk fontosság szerint. A megadott határidő miatt azonban nem mindegyiket sikerült megvalósítanunk, mivel az elsődleges célunk az volt, hogy az alkalmazás leadáskor használható legyen. Azonban ezeket az ötleteket, amik a jelenlegi kiadott verzióból kimaradtak, a jövőben mindenképpen szeretnénk megvalósítani. Ezeket azért tartjuk fontosnak megemlíteni mivel ez is jelzi, hogy ez a program számunkra már túlmutat az iskolai beadandó keretein.

Talán a legfontosabb ilyen el nem készült funkció az online fizetés lehetősége. Ennek a programba történő beimplementálását azért vetettük el, mert ez veszélyeztette volna a határidőre történő leadást. Azzal, hogy a jelentkezők előre, bankkártyás fizetéssel kifizetnék, a torna díját csökkenne a készpénzes tranzakciók száma. Ez a szervezők számára egy újabb könnyedség lenne, mivel így nem kellene a pénzük után szaladni, mint ahogyan most azt sok esetben azt teszik. Ez azt is eredményezné, hogy a jelentkezés lemondását csak egy előre meghatározott ideig lehetne tehermentesen megtenni, hiszen a szervezőknek egy nappal a torna előtt nehéz lenne egy másik csapatot találni.

Ezen felül lehetőséget látunk az előfizetési csomagok bevezetésére is. Például a szervezői jogosultság előfizetéshez lenne kötve és azon belül a hirdethető tornák száma a csomag méretétől függene. Egy bonyolultabb üzleti logika része lehetne, hogy a tornára történő befizetéskor egy bizonyos százalékot felszámolna az oldal a jelentkezés összegéből.

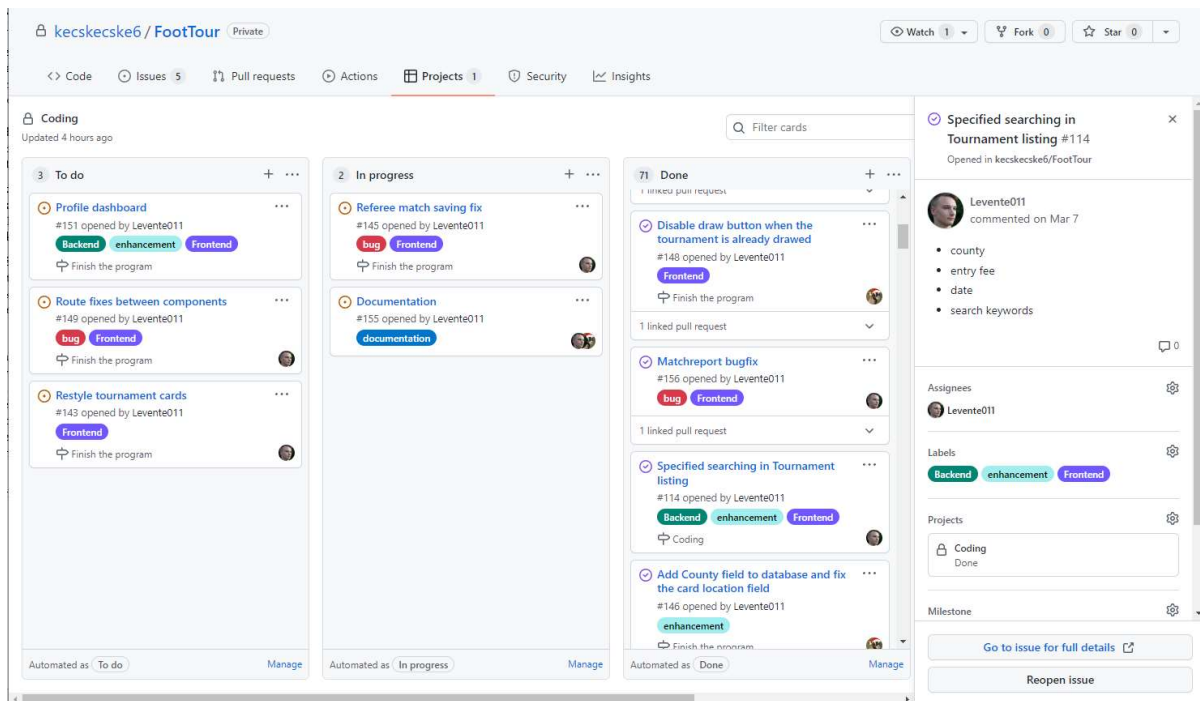
A jövőbeni terveink között szerepel, hogy a programot más sportágakra is kitudjuk terjeszteni és a labdarúgáson kívül más sportág tornáit is kezelni tudjuk.

Mint ahogyan a bevezető részben olvasható volt az oldal teljes mértékben reszponzív, ennek ellenére lehetségesnek tartjuk a jövőben egy mobilalkalmazás fejlesztését. Ezért a backend szervert úgy alakítottuk ki, hogy az a frontentdtől független legyen arra az esetre, ha egy másik típusú alkalmazást szeretnénk hozzá kapcsolni.

1.4 Feladat megosztás

Már a tervezés fázisa előtt megegyeztünk abban, hogy mindketten szeretnénk a fejlesztés során minden folyamatban részt venni, ezáltal minél nagyobb tudást szerezni a webfejlesztés területén. Ezért sem a backend és frontend kettéválasztásának mentén osztottuk fel a feladatokat, hanem mindketten részt vettünk a fullstack fejlesztésben. A feladatokat inkább funkciók szerint osztottuk fel. A csapatmunka megkönnyítése végett és a verziókövetéshez a GitHubot használtuk. Ezen a felületen a Trellohoz összegyűjtött nagyobb feladatokat, több kisebb részfeladatra osztottuk issue-k formájában, amelyeknek határidőket is szabtuk milestone-ok segítségével.

A GitHub repository-nk elérhető a <https://github.com/kecskecske6/FootTour> URL-en.

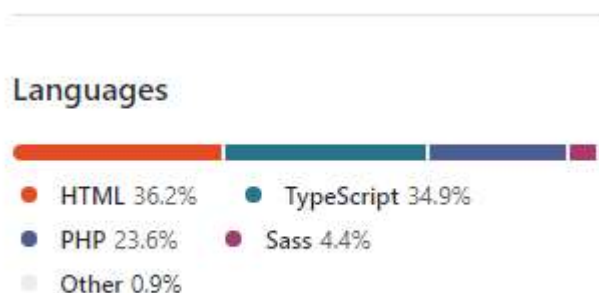


Az issue-kat három lehetséges kategóriába soroltuk. A még el nem kezdeteket a „*To do*” oszlopba helyeztük. Amiken éppen dolgoztunk azokat az „*In progress*” míg a befejezettek a „*Done*” oszlopba kerültek. Ahogyan a képről az is olvasható, a kép készítésekor 71 darab feladatot zártunk le. Mindegyikhez tartozik egy részletes leírás, ahogy az a kép jobboldalán is látható. Valamint különböző labelek is, mint például: Frontend, Backend, New feature stb. Ezekre főképpen azért volt szükség, hogy a könnyebben átlássuk, hogy a másik éppen min dolgozik.

A kommunikáció főképpen személyesen történt az iskolában, itt napi szinten egyeztettünk az előrehaladásunkról, a felmerült problémákról. Tanítási szünetekben, vagy esetlegesen hétvégéig, Discord-on beszélgettünk. Azért ezt a szoftvert választottuk mivel

mindketten jártasak vagyunk a használatában és rendelkezik képernyőmegosztás funkcióval, amely nagyon hasznos tudott lenni az ilyen megbeszélések során.

Minden issue-nál külön branch-ben dolgoztunk, melyeket Pull requestek segítségével töltöttünk fel a Main ágba. Természetesen ezek a branchek csak akkor kerültek feltöltésre, amikor teljes mértékben megbizonyosodtunk a hibátlan működésükről. Ezzel megakadályozva, a működő programkód hibással való felülírását a Main ágon belül. Valamint így hatékonyan tudtunk egyidőben, akár ugyanazon a funkción dolgozni.



A GitHub statisztikája alapján ezeket a programozási nyelveket használtuk a képen látható arányban.

The screenshot shows a GitHub pull request interface. At the top, it says 'kecskecske6 Merge pull request #161 from kecskecske6/kecskecske6/issue150' with a commit hash 'c5d6477' and '5 hours ago', and '322 commits'. Below this is a table listing files and their commit history.

File	Commit Message	Time Ago
Documents	API	3 months ago
backend	Awards done	5 hours ago
frontend	Awards done	5 hours ago
.gitignore	2nd yeet	last month
README.md	Yeet	2 months ago

Látható, hogy a kép készítésekor 322 commit-al rendelkezünk. A képen a mappastruktúra is megjelenik.

2 A program

2.1 Felhasznált technológiák

A programunk frontendje Angular keretrendszert használ. *Az AngularJS egy Google által fejlesztett, nyílt forráskódú JavaScript keretrendszer dinamikus webes alkalmazásokhoz. Segítségével nagyban egyszerűsödik a webes alkalmazások frontend fejlesztése. Használatával a HTML eszköztára kibővül és az alkalmazások komponensei még egyértelműen elkülönülnek. Az Angular adatkapcsolásának, illetve függőség injektálásának köszönhetően, rengeteg felesleges boilerplate kód elhagyható.* (ELTE IK, 2022) Az Angular 12-es verziószámát választottuk, mivel a szoftver fejlesztésének kezdetekor ez volt a legfrissebb kiadott verzió.

A backend fejlesztéséhez a natív PHP-t használtunk. A tanév elején különösebben kimagasló tudással egy programozási nyelvvel kapcsolatban sem rendelkeztünk. Mivel akkor ez tűnt számunkra a legszimpatikusabbnak ezért erre a nyelvre esett a választásunk. Fontos volt számunkra, hogy egy olyan nyelvet válasszunk, amely nagy online közösséggel rendelkezik, hiszen a kódolás során felmerülő hibákra így könnyebben találhatunk megoldást.

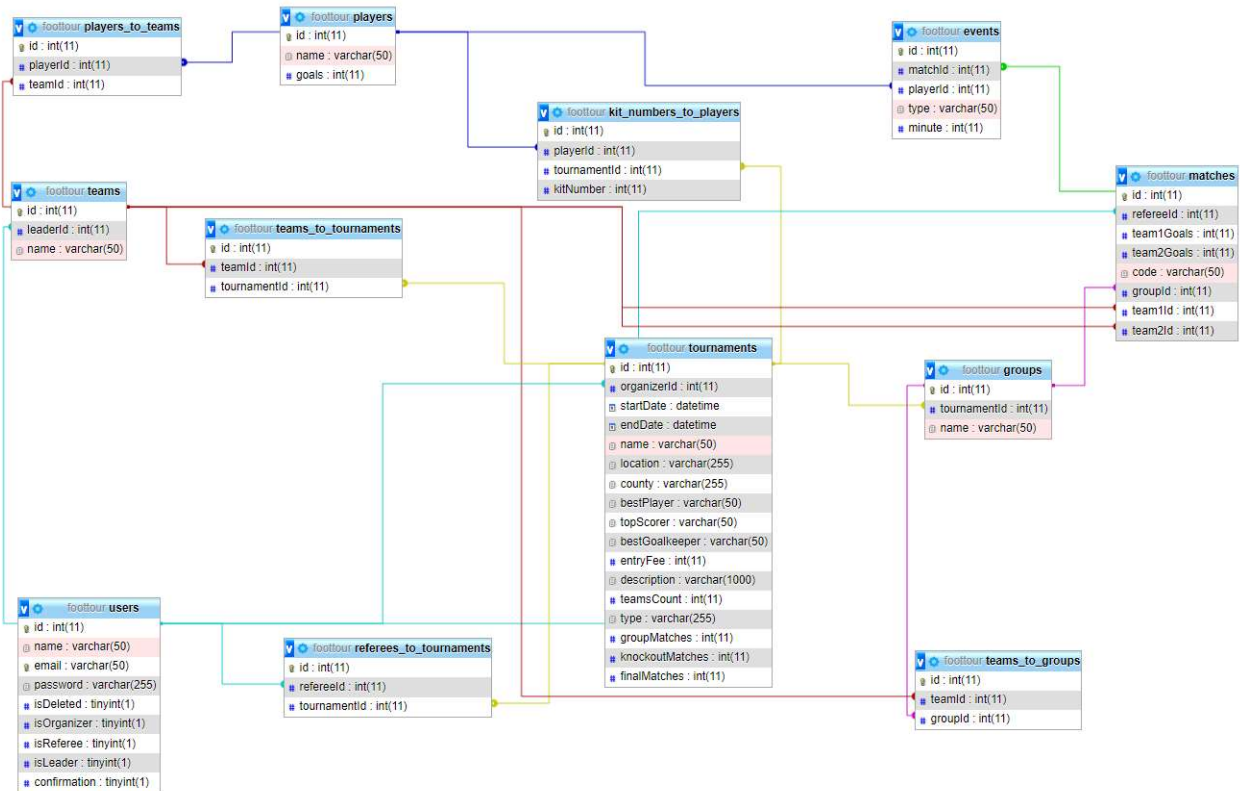
Az adatbázis választásnál egy relációs adatbázis kezelő rendszerre esett a választásunk, mégpedig a MySQL-re. A döntés kézenfekvő volt, mivel korábbi tanulmányainkból kifolyólag mindketten rendelkeztünk előzetes tudással.

2.2 Az adatbázis

2.2.1 Tervezés

Az adatbázis tervezésére nagy hangsúlyt fektettünk. Első körben készítettünk egy vázlatos tervet arról, hogy milyen táblákra lesz szükségünk, majd ezek után arról is, hogy ezek milyen mezőkkel rendelkezzenek. Arról is döntöttünk, hogy egyes mezők felvehetnek-e *null* értéket vagy kötelezően kitöltendőeknek kell lenniük, alapértelmezett értékkel rendelkezzenek-e. Ez a korai tervezési fázis után elkészítettük a weblapok vizuális tervét. Ezeken láttuk, hogy milyen beviteli mezőkkel, milyen adatokat szeretnénk bekérni a felhasználótól. Ez azért volt fontos mivel így összetudtuk hasonlítani, hogy az általunk létrehozott adatbázis terv alkalmas-e az általunk bekért adatok tárolására. Amennyiben azt tapasztaltuk, hogy nem képes pótolni a hiányt további mezők hozzáadásával. Ennek a korai tervezési fázisnak köszönhetően a későbbiekben jelentős módosításokat már nem kellett eszközölnünk az adatbázisunkon.

2.2.2 Relációs adatmodell



2.2.3 Táblák

Users – Felhasználók

A felhasználók adatait tároló tábla.

- *Id*: Elsődleges azonosító.
- *name*: A felhasználó neve, bejelentkezéshez szükséges adat. Legfeljebb 50 karakter hosszúságú lehet.
- *email*: A felhasználó e-mail-címe, bejelentkezéshez szükséges adat, legfeljebb 50 karakter hosszúságú lehet.
- *password*: A felhasználó jelszavát, titkosítva tároló mező.
- *isDeleted*: A felhasználó fiókjának esetleges törlését jelzi.
- *isOrganizer*: A felhasználó tornaszervező jogosultságát tároló mező.
- *isReferee*: A felhasználó játékvezetői jogosultságát tároló mező.
- *isLeader*: A felhasználó csapatvezetői jogosultságát tároló mező.
- *confirmation*: A regisztráció utáni fiók megerősítés állapotát tároló mező.

Teams – Csapatok

A csapatok adatait tároló tábla.

- *Id*: Elsődleges azonosító.
- *leaderId*: Idegenkulcs, a csapatvezető felhasználói fiókjának azonosítója.
- *name*: A csapat neve, legfeljebb 50 karakter hosszúságú lehet.

Players – Játékosok

A játékosok adatait tároló tábla. Mivel a csapatok játékosait nem kötelezzük felhasználói fiók regisztrálására, így minden tornán újra, új rekordként rögzítésre kerülnek a játékosok.

- *Id*: Elsődleges azonosító.
- *Name*: A játékos nevét tároló mező, legfeljebb 50 karakter hosszúságú lehet.
- *goals*: A játékos szerzett góljait tároló mező.

Tournaments – Tornák

A megrendezett tornák adatait tároló tábla, az adatok a torna létrehozását követően kerülnek rögzítésre. Az egyéni díjakat a torna lezárulását követően a szervező egy külön felületen tudja kiosztani a tornán részt vett játékosok között.

- *Id*: Elsődleges azonosító.
- *organizerId*: Idegenkulcs, a szervező felhasználói fiókjának azonosítója.
- *startDate*: A torna kezdetének időpontja.
- *endDate*: A torna végének időpontja.
- *name*: A torna neve, legfeljebb 50 karakter hosszúságú lehet.
- *location*: A torna helyszíne.
- *county*: A torna megyéje, a szűkített keresésnél tudnak a csapatvezetők megyék szerint szűrni tornákra.
- *bestPlayer*: Legjobb játékos, a torna végén megállapítható különdíj.
- *topScorer*: A torna gólkirálya.
- *bestGoalkeeper*: A legjobb kapus, a torna végén megállapítható különdíj.
- *entryFee*: A torna nevezési díja.
- *description*: A torna részletes leírása, amelyben a szervező minden információt megoszthat az eseményről.
- *teamsCount*: A torna férőhely száma.

- *type*: A torna lebonyolítási rendszerének típusa. A torna kezdődhet akár csoportmeccsekkel, majd folytatódhat az egyenes kiesés szakasszal vagy kezdődhet egyből az egyenes kieséses meccsekkel.
- *groupMatches*: A csoportmeccsek rendszere, oda-vissza vágós rendszerben zajlanak-e, vagy minden csapat csak egyszer játszik a másikkal.
- *knockoutMatches*: Az egyenes kieséses meccsek rendszere, oda-vissza vágós rendszerben zajlanak-e, vagy minden csapat csak egyszer játszik a másikkal.
- *finalMatches*: A döntő meccseinek száma.

Groups – Csoportok

A tornák sorsolása során keletkező csoportokat tároljuk ebben a táblában.

- *Id*: Elsődleges azonosító.
- *tournamentId*: Idegenkulcs, a csoporthoz tartozó torna azonosítója.
- *name*: A csoport neve, a torna menetrendjének generálását segíti elő.

Matches – Meccsek

A tornák meccseit, az azokon résztvevő csapatokat és azoknak az eredményeit tároljuk ebben a táblában.

- *Id*: Elsődleges azonosító.
- *refereeId*: Idegenkulcs, a mérkőzéshez hozzárendelt játékvezető azonosítója.
- *team1Goals*: Az elsőszámú csapat góljainak száma.
- *team2Goals*: A másodiksorszámú csapat góljainak száma.
- *code*: A mérkőzés kódja, a torna menetrendjének generálását elősegítő mező, jelzi, hogy a torna melyik fázisába tartozik a mérkőzés.
- *groupId*: Idegenkulcs, a meccshez tartozó csoport azonosítója.
- *team1Id*: Idegenkulcs, a meccshez tartozó elsőszámú csapat azonosítója.
- *team2Id*: Idegenkulcs, a meccshez tartozó másodiksorszámú csapat azonosítója.

Events – Események

A meccseken történő események (pl. gól, sárgalap, piroslap) tárolására szolgáló tábla.

- *Id*: Elsődleges azonosító.
- *matchId*: Idegenkulcs, az eseményhez tartozó mérkőzés azonosítója.

- *playerId*: Idegenkulcs, az eseményhez tartozó játékos azonosítója.
- *type*: Az esemény típusa, lehet akár gól, sárgalap, piroslap.
- *minute*: Az esemény történésének idejét tároló mező.

Kit_number_to_players – Mezszámok játékosokhoz rendelése

A mezszámok és a játékosok közti egy-a-többhöz kapcsolatot megvalósító tábla. A játékosok által választott mezszám csak arra a tornára érvényes amelyikre azt választották, a különböző tornákon, különböző mezszámokban is pályára léphetnek. Az adatok a csapatvezető sikeres jelentkezése után kerülnek rögzítésre.

- *Id*: Elsődleges azonosító.
- *playerId*: Idegenkulcs, a mezszámhoz tartozó játékos azonosítója (egy mezszámhoz több játékos is tartozhat).
- *tournamentId*: Idegenkulcs, azt tárolja, hogy a játékoshoz rendelt mezszám melyik tornára érvényes.
- *kitNumber*: Mezszám.

Players_to_teams – Játékosok a csapatukhoz rendelése

Az adatok a csapatvezető sikeres jelentkezése után kerülnek rögzítésre.

- *Id*: Elsődleges azonosító.
- *playerId*: Idegenkulcs, a csapathoz rendelt játékos azonosítója.
- *teamId*: Idegenkulcs, a játékoshoz rendelt csapat azonosítója.

Teams_to_tournaments – Csapatok tornákhoz rendelése

Az adatok a csapatvezető sikeres jelentkezése után kerülnek rögzítésre.

- *Id*: Elsődleges azonosító.
- *teamId*: Idegenkulcs, a tornához rendelt csapat azonosítója.
- *tournamentId*: Idegenkulcs, a csapathoz rendelt torna azonosítója.

Teams_to_groups – Csapatok csoportokhoz rendelése

Az adatok a torna sorsolása után kerülnek rögzítésre.

- *Id*: Elsődleges azonosító.
- *teamId*: Idegenkulcs, a csoporthoz rendelt csapat azonosítója.

- *groupId*: Idegenkulcs, a csapathoz rendelt csoport azonosítója.

Referees_to_tournaments – Játékvezetők tornákhoz rendelése

Az adatok a torna sikeres létrehozása után kerülnek rögzítésre.

- *Id*: Elsődleges azonosító.
- *refereeId*: Idegenkulcs, a tornához rendelt játékvezető azonosítója.
- *tournamentId*: Idegenkulcs, a játékvezetőhöz rendelt torna azonosítója.

2.2.4 Csatlakozás az adatbázishoz

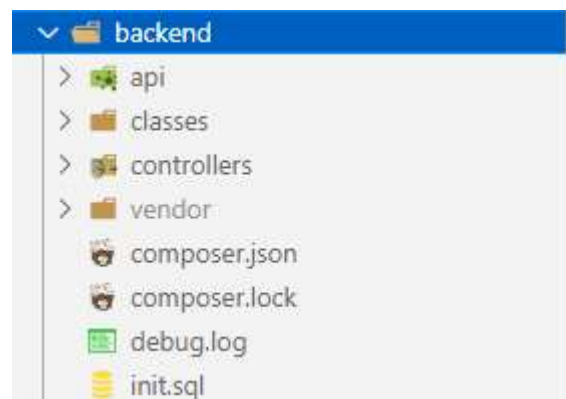
A backendünk adatbázishoz történő csatlakozása egy, erre a célra létrehozott osztályon keresztül történik. A csatlakozáshoz szükséges adatokat egy konfigurációs fájlban tároljuk, esetünkben a *config.ini*-ben. Az adatbázis osztály konstruktorában ezeknek az információknak a segítségével megalkotjuk a kapcsolati sztringünket az alább látható módon.

```
function __construct()
{
    $config = parse_ini_file('config.ini');
    $this->conn = new mysqli($config['db_host'], $config['db_user'], $config['db_pass'], $config['db_name']);
    mysqli_set_charset($this->conn, 'utf8');
    if ($this->conn->connect_error) die("Connection failed: " . $this->conn->connect_error);
}
```

Az osztály továbbá tartalmaz egy függvényt is, amely visszaadja a PHP backendünk és a MySQL adatbázisunk közötti kapcsolatot. Ezt a függvényt az *API* végpontokat tartalmazó fájlokban megtudjuk hívni és a kapott kapcsolatot paraméter formájában továbbítjuk a controllereknek.

2.3 Backend felépítése

A backendünk alapvetően három részre osztható. Ahogyan a képen is látható van egy *api* mappánk, amelyben megtalálhatóak az API végpontokként funkcionáló fájlok. Ezekben a fájlokban történik az autentikáció is (2.5) valamint a megfelelő controller meghívása. A képen az alatta következő *classes* mappa a modell osztályokat tartalmazza. Ezeknek az osztályoknak a mezőnevei megegyeznek az adatbázis tábláinak mezőneveivel. Ezzel jelentősen megkönnyítve a további munkát. Az utolsó ilyen mappa a *controllers*. Minden modellhez tartozik egy controller.

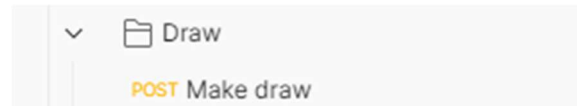


A képen továbbá látható egy *vendor* mappa, valamint a *composer*hez köthető fájlok. A projektben kettő függőséget használunk. Az egyik a firebase által kiadott *PHP-JWT* könyvtár, bővebben a 2.5.1 fejezetben. A másik pedig egy *faker*, mellyel tesztadatokat generáltunk a tesztelés céljából. Továbbá egy *init.sql* elnevezésű fájl, amely az adatbázist létrehozó scriptet tartalmazza.

2.4 API dokumentáció

2.4.1 DrawController

Elérési útvonat: /api/draw

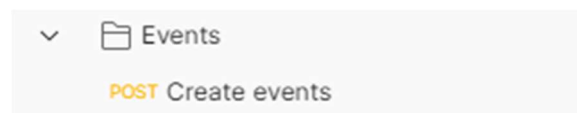


makeDraw(Tournament tournament, Team[] teams, User[] referees)

- Kérés típusa: HttpPost
- Paraméter: *Tournament* típusú osztálypéldány, *Team* típusú tömb és *User* típusú tömb
- Jogosultság: Organizer
- Feladat: Kisorsolja a csapatok párosításait, vagy a csoportokat, és a mérkőzésekhez hozzárendel játékvezetőket
- Válasz: *Match* típusú tömb
- Elérési útvonat: /create.php

2.4.2 EventController

Elérési útvonat: /api/events



createEvents(Event[] events)

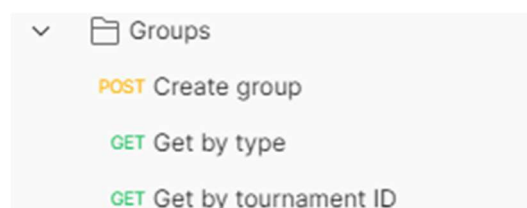
- Kérés típusa: HttpPost
- Paraméter: *Event* típusú tömb
- Jogosultság: Referee
- Feladat: Elmenti a mérkőzéseken történt eseményeket
- Válasz: {"message": "Sikeres"}
- Elérési útvonat: /save.php

2.4.3 GroupController

Elérési útvonat: /api/groups

createGroup(Group group)

- Kérés típusa: HttpPost
- Paraméter: *Group* típusú osztálypéldány



- Jogosultság: Organizer
- Feladat: Létrehoz csoportokat vagy párosításokat
- Válasz: *Group* típusú osztálypéldány
- Elérési útvonala: /create.php

getByType([FromQuery] number tournamentId, [FromQuery] string type)

- Kérés típusa: HttpGet
- Paraméter: szám és szöveges típusú
- Jogosultság: Organizer
- Feladat: Kilistázza az adott torna adott típusú csoportjait vagy párosításait
- Válasz: *Group* típusú tömb
- Elérési útvonala: /list.php?tournamentId=&type=

getByTournamentId([FromQuery] number tournamentId)

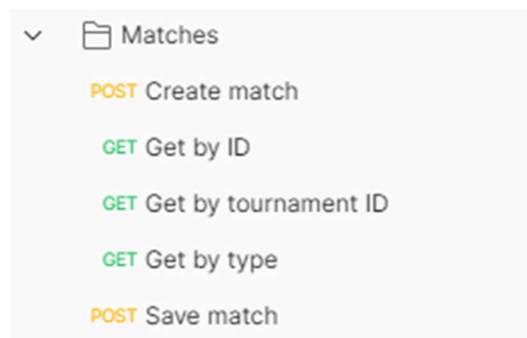
- Kérés típusa: HttpGet
- Paraméter: szám típusú
- Jogosultság: Organizer
- Feladat: Kilistázza az adott torna összes csoportját és párosítását
- Válasz: *Group* típusú tömb
- Elérési útvonala: /list.php?tournamentId=

2.4.4 MatchController

Elérési útvonala: /api/matches

createMatch(Match match)

- Kérés típusa: HttpPost
- Paraméter: *Match* típusú osztálypéldány
- Jogosultság: Organizer, Referee
- Feladat: Létrehoz egy mérkőzést
- Válasz: *Match* típusú osztálypéldány
- Elérési útvonala: /create.php



getById([FromQuery] number matchId)

- Kérés típusa: HttpGet
- Paraméter: szám típusú

- Jogosultság: nincs korlátozva
- Feladat: Kilistázza az adott azonosítóval rendelkező mérkőzést, és az ahhoz tartozó játékosokat és eseményeket
- Válasz: *Match* típusú osztálypéldány, kettő *Player* típusú tömb és *Event* típusú tömb
- Elérési útvonal: /getById.php?matchId=

getByTournamentId([FromQuery] number tournamentId)

- Kérés típusa: HttpGet
- Paraméter: szám típusú
- Jogosultság: nincs korlátozva
- Feladat: Kilistázza egy adott torna összes mérkőzését
- Válasz: *Match* típusú tömb
- Elérési útvonal: /getByTournamentId.php?tournamentId=

getByType([FromQuery] number tournamentId, [FromQuery] string type)

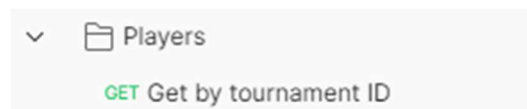
- Kérés típusa: HttpGet
- Paraméter: szám és szöveges típusú
- Jogosultság: nincs korlátozva
- Feladat: Kilistázza egy adott torna összes adott típusú mérkőzését
- Válasz: *Match* típusú tömb
- Elérési útvonal: /getByType.php?tournamentId=&type=

saveMatch(Match match)

- Kérés típusa: HttpPost
- Paraméter: *Match* típusú osztálypéldány
- Jogosultság: Referee
- Feladat: Elmenti egy mérkőzés változásait
- Válasz: { "message": "Sikeres" }
- Elérési útvonal: /save.php

2.4.5 PlayerController

Elérési útvonal: /api/players



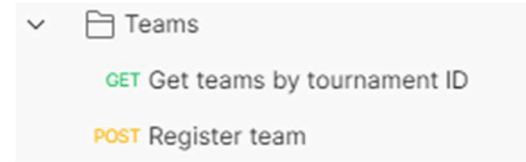
getByTournamentId([FromQuery] number tournamentId)

- Kérés típusa: HttpGet

- Paraméter: szám típusú
- Jogosultság: nincs korlátozva
- Feladat: Kilistázza egy adott tornán játszó összes játékost
- Válasz: *Player* típusú tömb
- Elérési útvonal: /list.php?tournamentId=

2.4.6 TeamController

Elérési útvonal: /api/teams



getTeamsByTournamentId([FromQuery] number tournamentId)

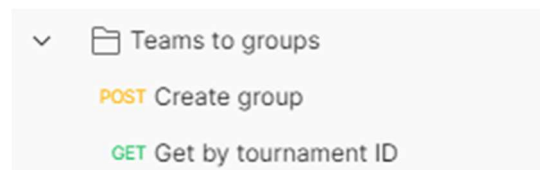
- Kérés típusa: HttpGet
- Paraméter: szám típusú
- Jogosultság: Organizer
- Feladat: Kilistázza egy adott torna összes csapatát
- Válasz: *Team* típusú tömb
- Elérési útvonal: /list.php?tournamentId=

registerTeam(Team team, Player[] players)

- Kérés típusa: HttpPost
- Paraméter: *Team* típusú osztálypéldány és *Player* típusú tömb
- Jogosultság: Leader
- Feladat: Egy tornára regisztrál egy csapatot és játékosait
- Válasz: { "message": "Sikeres regisztráció!" }
- Elérési útvonal: /registerToTournament.php

2.4.7 TeamstoGroupsController

Elérési útvonal: /api/teams_to_groups



createGroup(TeamstoGroups teamstoGroups)

- Kérés típusa: HttpPost
- Paraméter: *TeamstoGroups* típusú osztálypéldány
- Jogosultság: Organizer
- Feladat: Csapatokat hozzárendel egy csoporthoz vagy párosításhoz
- Válasz: *TeamstoGroups* típusú osztálypéldány
- Elérési útvonal: /create.php

getByTournamentId([FromQuery] number tournamentId)

- Kérés típusa: HttpGet
- Paraméter: szám típusú
- Jogosultság: nincs korlátozva
- Feladat: Kilistázza egy adott tornán a csapatok csoport- vagy párosításhozzárendeléseit
- Válasz: *TeamstoGroups* típusú tömb
- Elérési útvonal: /list.php?tournamentId=

2.4.8 TournamentController

Elérési útvonal: /api/tournaments

createTournament(Tournament tournament)

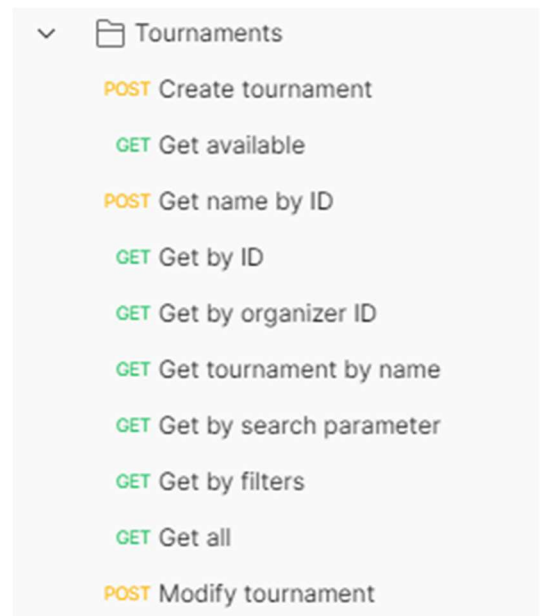
- Kérés típusa: HttpPost
- Paraméter: *Tournament* típusú osztálypéldány
- Jogosultság: Organizer
- Feladat: Létrehoz egy tornát
- Válasz: *Tournament* típusú osztálypéldány
- Elérési útvonal: /create.php

getAvailable()

- Kérés típusa: HttpGet
- Paraméter: nincs
- Jogosultság: nincs korlátozva
- Feladat: Kilistázza az elérhető tornákat
- Válasz: *Tournament* típusú tömb
- Elérési útvonal: /getAvailable.php

getNameById(number id)

- Kérés típusa: HttpPost
- Paraméter: szám típusú
- Jogosultság: nincs korlátozva
- Feladat: Visszaadja egy adott torna nevét és azonosítóját
- Válasz: a torna neve
- Elérési útvonal: /getNameById.php



getById([FromQuery] number id)

- Kérés típusa: HttpGet
- Paraméter: szám típusú
- Jogosultság: nincs korlátozva
- Feladat: Visszaadja adott azonosítójú torna összes adatát
- Válasz: *Tournament* típusú osztálpéldány
- Elérési útvonal: /list.php?id=

getByOrganizerId([FromQuery] number userId)

- Kérés típusa: HttpGet
- Paraméter: szám típusú
- Jogosultság: Organizer
- Feladat: Kilistázza az összes adott szervező által szervezett tornát
- Válasz: *Tournament* típusú tömb
- Elérési útvonal: /list.php?userId=

getTournamentByName([FromQuery] string name)

- Kérés típusa: HttpGet
- Paraméter: szöveges típusú
- Jogosultság: nincs korlátozva
- Feladat: Visszaadja adott nevű torna összes adatát
- Válasz: *Tournament* típusú osztálpéldány
- Elérési útvonal: /list.php?name=

getBySearchParameter([FromQuery] string parameter)

- Kérés típusa: HttpGet
- Paraméter: szöveges típusú
- Jogosultság: nincs korlátozva
- Feladat: Kilistázza az összes tornát, melynek valamelyik attribútuma tartalmazza a keresett szöveget
- Válasz: *Tournament* típusú tömb
- Elérési útvonal: /list.php?parameter=

getByFilters([FromQuery] string county, number min, number max, string pickedDates)

- Kérés típusa: HttpGet
- Paraméter: szöveges és szám típusúak
- Jogosultság: nincs korlátozva
- Feladat: Kilistázza az összes tornát, melyet az adott megyében rendeznek, az adott minimum és maximum nevezési díjjal rendelkezik, és az adott dátumokkor zajlik
- Válasz: *Tournament* típusú tömb
- Elérési útvonal: /list.php?county=&min=&max=&pickedDates=

getAll()

- Kérés típusa: HttpGet
- Paraméter: nincs
- Jogosultság: nincs korlátozva
- Feladat: Kilistázza az összes tornát
- Válasz: *Tournament* típusú tömb
- Elérési útvonal: /list.php

modifyTournament(Tournament tournament)

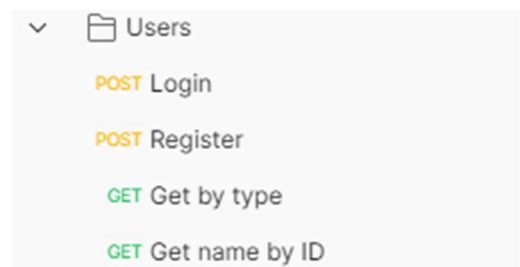
- Kérés típusa: HttpPost
- Paraméter: *Tournament* típusú osztálypéldány
- Jogosultság: Organizer
- Feladat: Módosít egy tornát
- Válasz: *Tournament* típusú osztálypéldány
- Elérési útvonal: /modify.php

2.4.9 UserController

Elérési útvonal: /api/users

login(User user)

- Kérés típusa: HttpPost
- Paraméter: *User* típusú osztálypéldány
- Jogosultság: Leader, Organizer, Referee
- Feladat: Belépteti az adott felhasználót
- Válasz: Egy objektum, mely tartalmazza a hozzáférési tokent
- Elérési útvonal: /list.php



register(User user)

- Kérés típusa: HttpPost
- Paraméter: *User* típusú osztálypéldány
- Jogosultság: nincs korlátozva
- Feladat: Regisztrál egy felhasználót
- Válasz: *User* típusú osztálypéldány
- Elérési útvonat: /create.php

getByType([FromQuery] string type)

- Kérés típusa: HttpGet
- Paraméter: szöveges típusú
- Jogosultság: Organizer
- Feladat: Kilistázza az összes, adott jogosultsággal rendelkező felhasználót
- Válasz: *User* típusú tömb
- Elérési útvonat: /list.php?type=

getNameById([FromQuery] number id)

- Kérés típusa: HttpGet
- Paraméter: szám típusú
- Jogosultság: nincs korlátozva
- Feladat: Visszaadja adott azonosítójú felhasználó nevét
- Válasz: szöveges típusú
- Elérési útvonat: /list.php?id=

2.5 Frontend

2.5.1 Felépítése

A frontendünk struktúrája főként a komponensekből, modellekből, interfészekből és a servicekből áll.

A modellek és interfészek megegyeznek a backend modelljeivel. A services mappában található servicek főképpen a backenddel történő kommunikációért felelnek. Emellett található bennük néhány olyan függvény, amelyet több komponensben is meghívunk. Ilyen például az

`auth.service.ts`-ben található `getToken`, amely a bejelentkezett felhasználó JWT (2.5.1) tokenjét adja vissza.

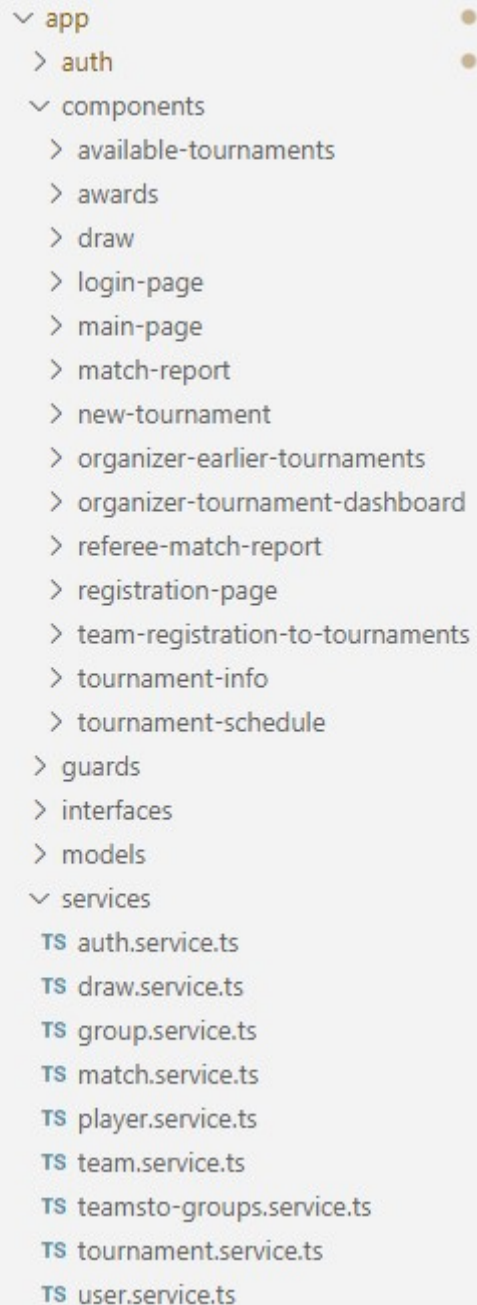
A komponensek kezelik a webalkalmazás oldalainak megjelenítését. Ahogyan a képen is látható elnevezésük beszédes, egyértelműen elárulja, hogy az adott komponens milyen funkciót biztosít. A komponensek mappái három fájl tartalmazznak: a `html`-t a `sass`-t és a `script` fájlt. Ezek a mappán belül összeköttetésben vannak. Abban az esetben, ha a komponensek között kommunikációra van szükségünk, akkor ezt függőséginjektálással (dependency injection) érjük el.

2.5.2 Reszponzív webalkalmazás

Nagyon fontos volt számunkra, hogy az oldal rezponzív legyen, tehát képernyőmérettől függetlenül minden eszközön használható legyen a program. Például biztosak vagyunk benne, hogy nem minden tornán biztosítanak a játékvezetők számára laptopot, hanem a játékvezetők a saját okostelefonjukon írnák meg

a mérkőzések jegyzőkönyvét vagy valaki éppen hazafele tart a munkából egy kollégájával és az utazás alatt a telefonján hétvégi programként jelentkeznek egy a környékén lévő tornára.

A rezponzivitás megvalósításához a Bootstrap keretrendszer 5-ös verzióját használtuk. *A Bootstrap egy nyílt forráskódú keretrendszer (framework), mely HTML, CSS, JavaScript technológiákat használ. Alapvetően arra jó, hogy nagyon könnyedén, és minimális energia befektetéssel tudjon valaki jól kinéző, bármilyen képernyőméreten szépen megjelenő weboldalakat készíteni.* (Gremmedia, 2021)



```

  app
  ├── auth
  ├── components
  │   ├── available-tournaments
  │   ├── awards
  │   ├── draw
  │   ├── login-page
  │   ├── main-page
  │   ├── match-report
  │   ├── new-tournament
  │   ├── organizer-earlier-tournaments
  │   ├── organizer-tournament-dashboard
  │   ├── referee-match-report
  │   ├── registration-page
  │   ├── team-registration-to-tournaments
  │   ├── tournament-info
  │   └── tournament-schedule
  ├── guards
  ├── interfaces
  ├── models
  ├── services
  │   ├── auth.service.ts
  │   ├── draw.service.ts
  │   ├── group.service.ts
  │   ├── match.service.ts
  │   ├── player.service.ts
  │   ├── team.service.ts
  │   ├── teamsto-groups.service.ts
  │   ├── tournament.service.ts
  │   └── user.service.ts
  └── views
  
```


2.6 Autentikáció, biztonság

2.6.1 JWT

A backend felé érkező kérések hitelesítéséhez a JWT (JSON Web Token) szabványt használjuk. *A JWT egy olyan szabvány, amely kompakt és önálló módszert határoz meg az információk biztonságos továbbításához az ügyfél és a szerver között, mint JSON objektumot. A kompakt méret megkönnyíti a tokenek átadását URL-en, POST paraméteren vagy egy HTTP fejlécen keresztül. Továbbá, mivel önállóak, tartalmazzák a felhasználóval kapcsolatos összes szükséges információt, így az adatbázist nem kell többször lekérdezni.* (Ilusionity, 2022) A tokent a felhasználó sikeres bejelentkezésekor generáljuk le. Ez tartalmazza a token lejáratát, a felhasználó fontosabb adatait (természetesen a jelszavát nem) valamint a jogosultságát.

2.6.2 Autentikáció menete

Backend oldalon:

Ahogy már említettük a tokeneket a felhasználó sikeres bejelentkezésénél generáljuk le. Ennek a megvalósításához szükség van egy titkos jelszóra, valamint egy titkosítási algoritmusra. Ezeknek az adatoknak a hiányában a generált token nem visszafejthető és a bennük található adatok nem kerülnek nyilvánosságra.

A beérkező kérés esetén, a kérés fejlécének tartalmaznia kell a tokent. Ezt a kontrollerek között található *auth.php* kezeli. A *getAuthorizationHeader* metódus kiolvassa a fejlécből a tokent, majd azt egy másik függvény ellenőrzi. Amennyiben az ellenőrzés sikertelen a backend egy 401-es hibakóddal ellátott válaszüzenetet küld és a felhasználó kijelentkeztetésre kerül.

Frontend oldalon:

Bejelentkezéskor az *auth.service.ts*-ben található *loginForm* függvény kerül meghívásra, amely továbbítja a backend felé a felhasználó bejelentkezési adatait. A válaszként kapott tokent, illetve a bejelentkezett felhasználó szükséges adatait a böngésző lokális tárolójában tároljuk. Innen a kijelentkezés során törlésre kerülnek.

Ahogy a backend részben említettük, a backend felé irányuló *HTTP* kérések fejlécének tartalmazniuk kell a tokent. Ezt a *token.interceptor.ts* konstruktorában valósítjuk meg. Itt minden kérés fejlécéhez hozzacsatolásra kerül a token.

Egyes elérési útvonalak csak bizonyos jogosultságú felhasználók számára érhetőek el. Az ezekhez eljuttató menüpontokat az Angular direktívái segítségével csak az arra

jogosultaknak jelenítjük meg. Ez önmagában azonban még nem lenne elég, mivel ezek az oldalak továbbra is elérhetőek lennének a kereső böngészőszájába begépelte pontos elérési út által. Ezt megelőzve minden olyan elérési útnak megadtuk, hogy milyen típusú felhasználó érheti azt el. Amikor valaki egy ilyen elérési útvonalat akar aktiválni, az *auth.guard.ts*-ben található `canActivate` függvény ellenőrzi, hogy a felhasználó rendelkezik-e a szükséges jogosultsággal.

```
const routes: Routes = [
  { path: '', component: MainPageComponent },
  { path: 'login', component: LoginPageComponent },
  { path: 'register', component: RegistrationPageComponent },
  { path: 'availabletournaments', component: AvailableTournamentsComponent },
  { path: 'mytournaments', canActivate:[AuthGuard], data:{role: "any"}, component: OrganizerEarlierTournamentsComponent },
  { path: 'tournament/:tournamentinfo', component: TournamentInfoComponent },
  { path: 'matchreport/:id', component: MatchReportComponent },
  { path: 'mytournaments/:id', canActivate:[AuthGuard], data:{role: "any"}, component: OrganizerTournamentDashboardComponent },
  { path: 'referee/:id', canActivate:[AuthGuard], data:{role: "referee"}, component: RefereeMatchReportComponent },
  { path: 'schedule/:id', component: TournamentScheduleComponent },
  { path: 'draw/:id', canActivate:[AuthGuard], data:{role: "organizer"}, component: DrawComponent },
  { path: 'teamregistration/:id', canActivate:[AuthGuard], data:{role: "leader"}, component: TeamRegistrationToTournamentsComponent },
  { path: 'newtournament', canActivate: [AuthGuard], data:{role: "organizer"}, component: NewTournamentComponent },
  { path: 'awards/:id', canActivate: [AuthGuard], data:{role: "organizer"}, component: AwardsComponent }
];
```

2.6.3 Jelszavak

A felhasználó jelszavát regisztráció után titkosított formában tároljuk el az adatbázisban. A regisztrációkor, bejelentkezéskor begépelte titkosítatlan jelszó nem kerül tárolásra. A backendről a jelszavak semmilyen formában sem kerülnek visszaküldésre a frontend irányába, ezzel is növelve a biztonságot.

2.6.4 Felhasználó típusok

A programban jelenleg három féle jogosultság alapján soroljuk be a felhasználókat: Csapatvezető, Szervező, Játékvezető, Nézelődő. A nézelődő kategóriájába esik minden olyan felhasználó, aki felhasználói fiókkal nem rendelkezik és például csak egy mérkőzés jegyzőkönyvét szeretné megtekinteni.

Csapatvezető:

A tornák között tud böngészni, azokra be tudja regisztrálni a saját csapatát annak a nevének és játékosainak megadásával.

Szervező:

Tornákat tud létrehozni különböző lebonyolítási sémák alapján. Ezeknek a tornáknak el tudja készíteni a lebonyolítását csupán egy gombnyomással. A tornák végén kitudja osztani az ilyenkor szokásos egyéni díjakat.

Játékvezető:

A szervezők tudják hozzárendelni tornákhoz a felhasználói azonosítójuk alapján. Mérkőzések jegyzőkönyvét tudja megírni. Rögzíteni tudja a gólszerzőket, sárgalapokat, piroslapokat.

2.7 Tiszta kód

A kódolás során törekedtünk arra, hogy a tiszta kód elveinek megfeleljünk. A változóknak, osztályoknak, függvényeknek beszédes neveket adtunk. A kód jól olvasható, tagolt, behúzásokat használ.

3 Tesztek

Frontend oldalon a Selenium webes bővítményt használtuk tesztelés céljából. A Selenium az általunk megadott utasításokat megismételve teszteli a webalkalmazás frontend oldali funkcióit.

Emellett Angular tesztek is írtunk, melyek az egyes szervizek függvényeit tesztelik le. A képen látható, hogy szinte minden szervizhez társul egy tesztfájl is.

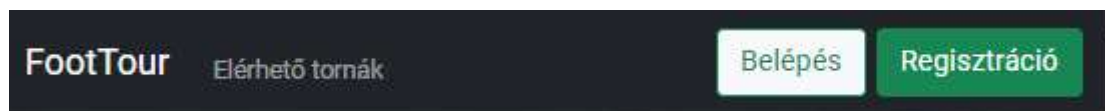
- ✓ Browse
- ✓ Create tournament
- ✓ Draw
- ✓ Login
- ✓ Match report
- ✓ Register team
- ✓ Registration



4 Felhasználói kézikönyv

4.1 Navigációs sáv

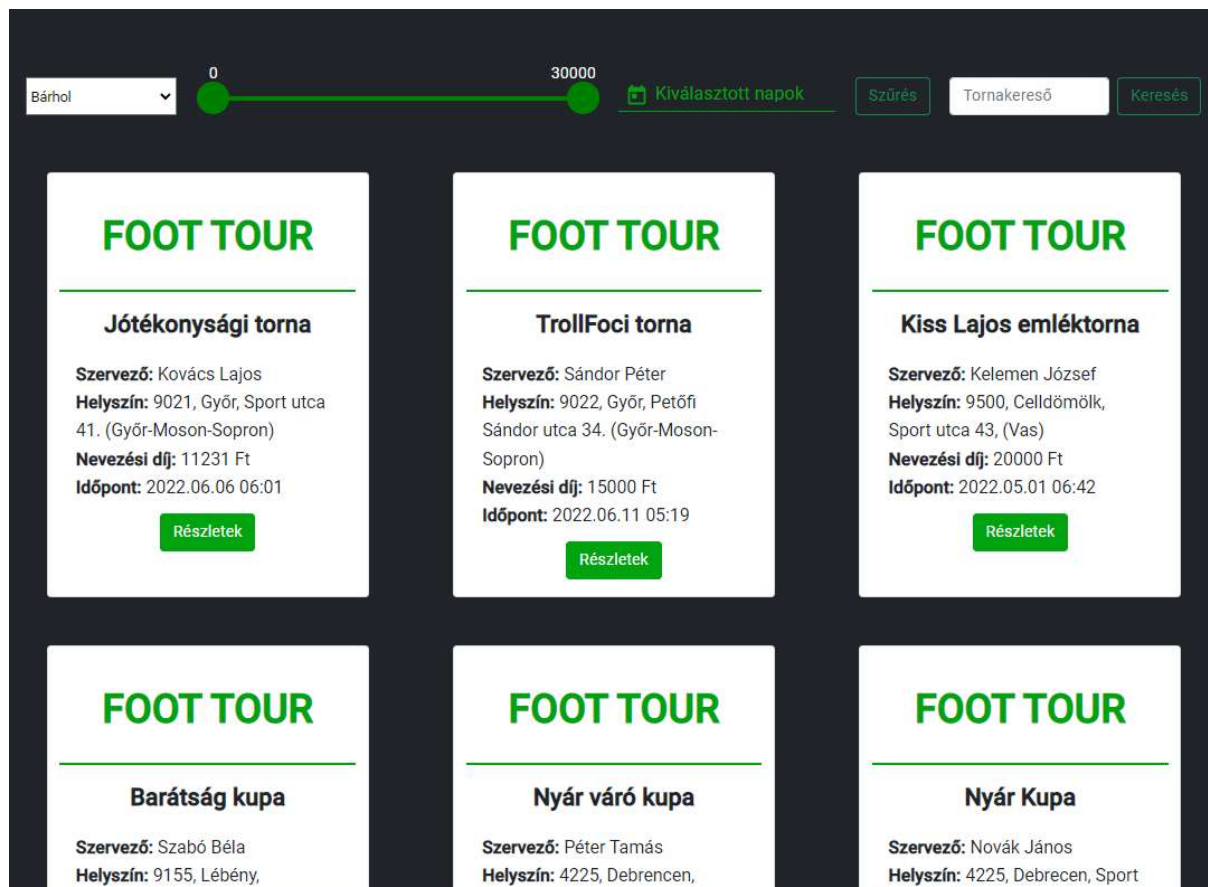
Az oldalt megnyitva a felhasználót a kezdőoldal és annak tetején a navigációs menü fogadja. Ezen a menünk különböző lehetőségek jelennek meg attól függően, hogy a felhasználó be van-e jelentkezve.



1.ábra – Navigációs sáv

4.1.1 Elérhető tornák

Ez az oldal bárki számára elérhető, nem kell hozzá felhasználói fiókkal rendelkeznie. A menüpontra kattintva megnyílik előttünk egy oldal, ahol az összes jövőbeni torna közül tud böngészni az ember. A szűrési opciókkal vagy a tornakeresővel pedig mindenki megtalálhatja a számára legmegfelelőbb labdarugó tornát.



2.ábra – Elérhető tornák

Az érdeklődők a következő szűrési feltételeket alkalmazhatják az oldalon akár önmagukban, akár ezeket kombinálva:

Megye alapján:

Lehetőség van megyékre szűkíteni a megjelenített tornákat, ehhez nem kell mást tennie a felhasználónak, mint a lenyíló listából kiválasztani Magyarország 19 megyéjéből a számára a legszimpatikusabbat kiválasztani.

Nevezési díj alapján:

Az érdeklődő a két csúszka segítségével betudja állítani a minimális és maximális nevezési díjat.

Napok alapján:

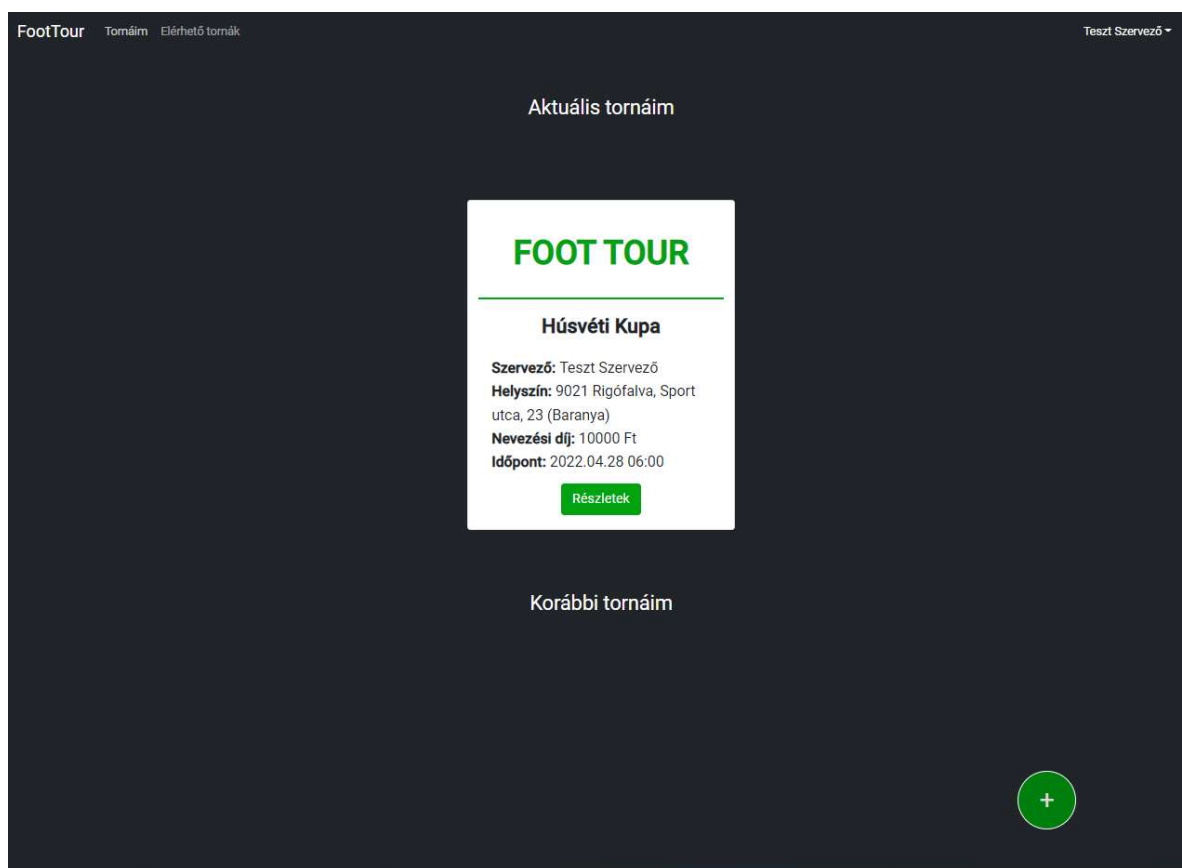
Lehetősége van a felhasználónak napokat is kiválasztani. Megtudja határozni, hogy melyik napokon megszervezett tornákat akar látni. Egy időben maximum 3 dátumot tud kiválasztani a naptárból, ha ennél többet választana ki azt a program a kiválasztott napok alatti zöld csík piros színre váltásával jelzi.

Tornakereső:

Az érdeklődők rátudnak keresni bármely kifejezésre a tornakereső segítségével. Ez azokat a tornákat listázza, amelyeknek kártyái bármilyen formában tartalmazzák a megadott kifejezést.

4.1.2 Tornáim

Ez az oldal csak azok számára elérhető, akik rendelkeznek felhasználói fiókkal. Azonban annak a típusa már nem számít, mivel ez az oldal mindenki számára azokat a tornákat listázza, amelyben korábban érintett volt vagy jelenleg is az, esetleg a jövőben lesz. Az ábrán látható pluszjel csak a szervezők számára jelenik meg. Erre kattintva megjelenik az új torna létrehozása felület (4.2.2).



3.ábra – Tornáim

4.2 Regisztráció

A felhasználói fiókkal nem rendelkező emberek számára a navigációs sávban megjelenik a regisztráció gomb. Erre kattintva az ábrán látható felület nyílik meg számunkra. Itt a teljes nevünk, e-mail címünk, és jelszavunk megadásával regisztrálhatunk felhasználói fiókot. Egy reCAPTCHA hitelesítést is végre kell hajtunk, amely igazolja, hogy emberek vagyunk. Az ábra alján látható opciók közül kiválaszthatjuk, hogy milyen típusú felhasználói fiókot kívánunk regisztrálni (2.5.4). Sikeres regisztráció esetén az alkalmazás átnavigál a bejelentkezés oldalára, ahol már be is tudunk jelentkezni a frissen regisztrált felhasználói fiókunkkal.

4.3 Bejelentkezés

A felhasználói fiókkal nem rendelkező emberek számára a navigációs sávban megjelenik a belépés gomb. Erre kattintva a felhasználó a bejelentkezés felületén találja magát. Sikeres regisztráció után automatikusan megjelenik a felület. Itt a regisztrációkor megadott e-mail és jelszó páros megadásával jelentkezhet be a felhasználó. A bejelentkezés után a bejelentkezés és regisztráció gombok helyén megjelenik a felhasználó neve, amelyre kattintva egy legördülő lista elemeként a kijelentkezés gomb érhető el. A felhasználónak a felhasználói fiókja 1 napig marad beléptetve, 1 nap elteltével lejár a JWT token (2.5.2). Ennek kényelmi és biztonsági okai is vannak.

3.ábra – Regisztráció

4.ábra -
Bejelentkezés

4.4 Tornák

4.4.1 Létrehozás

Ha egy szervező rányom a 3. ábrán látható plusz nyílra akkor megnyílik számára a torna létrehozása felület, amelyet sikeresen kitöltve már létre is hozta a saját tornáját, amely egyből elérhetővé válik az elérhető tornák oldalon. Itt adható meg a lebonyolítási rendszer típusa is. A program jelenleg 8, 16, 32 csapatos labdarugó tornák lebonyolítására képes. A lebonyolítási típusok közül kettő áll a felhasználó rendelkezésére, a csoportkörös küzdelmek majd egyenes kieséses szakasz vagy a csoportkört kihagyva egyből egyenes kieséses meccsekkel is kezdhető a torna.

5.ábra – Torna létrehozása

4.4.2 Részletes leírás 1

A felhasználók tornáim (3. ábra) elérhetik a torna adatait. Itt láthatja az eddig nevezett csapatokat. Az esemény helyszínét, nevezési díját, kezdési időpontját. Valamint két gombot, a menetrend feliratú gomb a torna sorsolásához vezet. A másik gomb csak a szervező számára elérhető abban az esetben, ha a torna beosztása még nem került kisorsolásra.

4.4.3 Részletes leírás 2

Az elérhető tonák oldalon a részletek gombra nyomva (2. ábra) a következő felületet láthatja meg a felhasználó (7. ábra). Amennyiben a felhasználó csapatvezetői jogosultsággal rendelkezik és a tornán még van szabad férőhely, akkor a jelentkezem gomb is megjelenik.

Húsvéti Kupa

Nevezett csapatok: 8/8

1
2
3
4
5
6
7
8

Helyszín: 9021 Rigófalva, Sport utca, 23 (Baranya)

Nevezési díj: 10000Ft

Kezdési időpont: 2022.04.28 06:00

[Menetrend](#) [Sorsolás](#)

6.ábra – Torna kezelő felület

Teszt Torna

Szervező:

Helyszín: 9155, Lébény, Sport utca 53 (Győr-Moson-Sopron)

Nevezési díj: 15000Ft

Csapatok száma: 1/8

Kezdési időpont: 2022.05.17 08:00

Leírás

Egy torna tesztelés céljából.

[Jelentkezem](#)

7.ábra – Torna részletes leírása

4.4.4 Sorsolás

A szervező a kieséses szakasz meccsek számának és a döntő meccsek számának megadásával meghatározhatja a lebonyolítási rendszer menetét. Valamint a tornához játékvezetőket rendelhet a játékvezetők felhasználói fiókjának azonosítójának megadásával.

4.4.5 Csapatrejistráció tornára

A csapatvezető a 7. ábrán látható jelentkezés gomb megnyomásával elkezdheti a regisztrációs folyamatot. A program egy csapatvezetőnek ugyanarra a tornára csak egy csapatot engedélyez benevezni. Első körben a vezetőnek meg kell adnia az, hogy a csapata milyen néven szeretne szerepelni a tornán (9. ábra). Ennek a sikeres kitöltésével és a véglegesítés gomb lenyomásával jelenik meg a 10. ábrán látható oldal, amelyen a csapat játékosait és azoknak a mezszámát kell rögzítenie a csapatvezetőnek. Ha ezeket az adatokat esetlegesen hibásan adja meg akkor sem kell megijednie, hiszen egy játékos adatai szerkeszthetőek, valamint törölhetőek a regisztráció befejezéséig.

8.ábra – Sorsolás

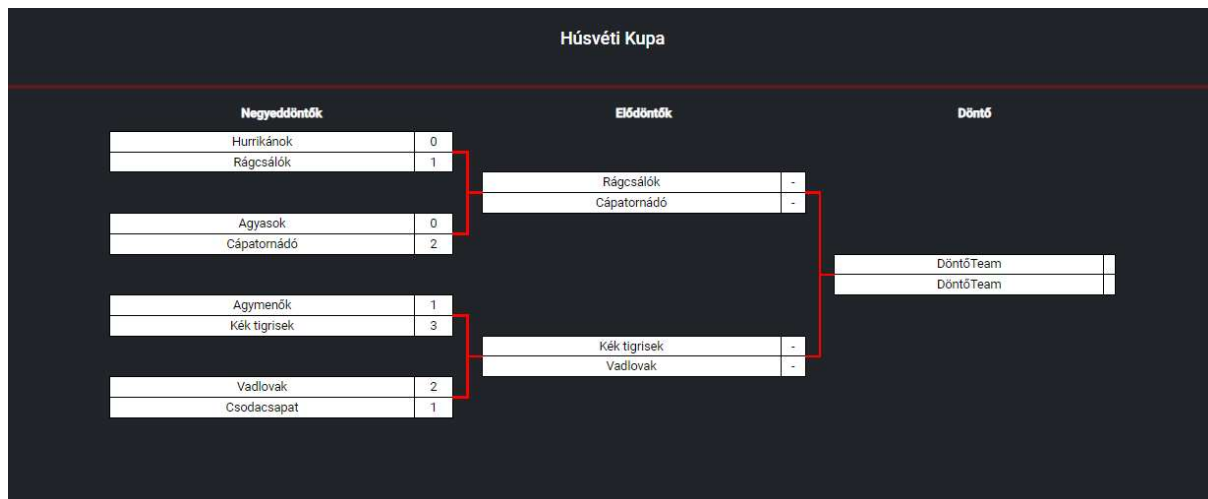
9.ábra – Csapatnév megadás

Mezszám	Név	
0	Játékos neve	✓
1	Kovács János	✎ 🗑
10	Ágfalvi Botond	✎ 🗑
13	Szabó Márk	✎ 🗑

10.ábra – Játékosok megadása

4.4.6 Menetrend

Miután a szervező rányom a 6. ábrán látható sorsolás gombra és kitölti a 8. ábrán látható felületet már készen is lesz egy tornának a beosztása. A 11. ábrán egy nyolc fős torna egyenes kieséses típusú lebonyolítása látható.



11.ábra – Menetrend

Ha itt valaki rákattint az eredményekre akkor annak a mézskörésnek a jegyzőkönyve jelenik meg (14. ábra). Abban az esetben pedig, ha egy játékvezető kattint a még nem rögzített mérkőzésre akkor annak a szerkeszthető jegyzőkönyve jelenik meg számára.

4.4.7 Díjak

A tornák végén szokás kiosztani különböző egyéni díjakat. Ilyenek például a torna játékosa, a legjobb kapus és a gólkirály.

The screenshot shows a form titled 'Díjak kiosztása' (Award Distribution). It contains three dropdown menus for selecting the award recipients:

- Legjobb játékos (Best Player)
- Legtöbb gól (Most Goals)
- Legjobb kapus (Best Goalkeeper)

Below the dropdowns is a blue button labeled 'Véglegesítés' (Finalize).

12.ábra – Díjak kiosztása





4.5 Mérkőzések

4.5.1 Játékvezetői jegyzőkönyv


A játékvezetők külön szerkesztői felülettel rendelkeznek a mérkőzések jegyzőkönyvének rögzítéséhez. Ez a felület mindösszesen annyiban különbözik a bárki számára elérhető jegyzőkönyv oldalától, hogy vannak gombok melyekkel gólokat, sárgalapokat, esetlegesen piros lapokat tudnak rögzíteni.

Rágcsálók
2 - 1
Cápatornádó

Rágcsálók

Mezszám	Név		Gól	Sárga	Piros
1	Orsós László		+	+	+
5	Pap Dominik	 10' 	+	+	+
10	Antal Marcell		+	+	+
18	Miksa Benedek	 21' 	+	+	+
33	Márton Csanád	 14' 	+	+	+

Cápatornádó

Mezszám	Név		Gól	Sárga	Piros
1	Sípos Viktor		+	+	+
3	Apród Sándor	 7' 	+	+	+
10	Hegedüs Kevin		+	+	+
16	Dudás Krisztofer	 3' 	+	+	+
26	Balla Béla		+	+	+

Játékvezető:
Húsvéti Kupa

Rögzítés

13.ábra – Játékvezető jegyzőkönyv

4.5.2 Jegyzőkönyv

Mint ahogyan az előző fejezetben (4.5.1) említettük ez a felület annyiban különbözik az 13. ábrán láthatótól, hogy itt nincs lehetőség szerkesztésre, valamint bárki számára elérhető.

Rágcsálók	2 - 1	Cápatornádó
Rágcsálók		
1	Orsós László	
5	Pap Dominik	 10'
10	Antal Marcell	
18	Miksa Benedek	 21'
33	Márton Csanád	 14'
Cápatornádó		
1	Sípos Viktor	
3	Apród Sándor	 7'
10	Hegedűs Kevin	
16	Dudás Krisztofer	 3'
26	Balla Béla	
Játékvezető: Húsvéti Kupa		

14.ábra – Jegyzőkönyv

4.6 Fejlesztői futtatási kézikönyv

A program fejlesztői verziójának futtatásához a következő lépéseket kell végrehajtanunk:

A futtatáshoz rendelkezniünk kell az XAMPP nevű programmal. Ezt a programot elindítva, a felkínált lehetőségek közül indítsuk el az Apache és MySQL modulokat. Következő lépésként a backend mappában található init.sql adatbázis létrehozó scriptet. Egy parancssort megnyitva, amelynek az aktuális könyvtára a backend mappa legyen, ki kell adni a composer update parancsot, majd a `php -S localhost:8000` parancsot. Ezzel a backend szerverünk fut a 8000-res porton. Ezután egy külön parancssorban nyissuk meg a frontend mappát majd adjuk ki az `npm i` parancsot, ezzel a package.json-ben található függőségeket feltelepítjük. Majd ugyan itt adjuk ki az `ng serve` utasítást. Ezzel a frontendünk is elérhetővé válik a 4200-as porton.

Irodalomjegyzék

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

Gremmedia (2021.08.09). Forrás: Mi az a Bootstrap 4? Hpgyan érdemes használnunk? Mit érdemes tudni róla? <https://gremmedia.hu/mi-az-a-bootstrap-4-hogyan-hasznaljuk#>

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