EÖTVÖS LORÁND TUDOMÁNYEGYETEM

INFORMATIKAI KAR

Programozási nyelvek és Fordítóprogramok Tanszék

Webalkalmazás fejlesztése Java nyelvben a Spring keretrendszerrel

Témavezető:
Török Márk
tanársegéd(gyakornok)

Szerző:
Katona Áron
programtervező informatikus BSc

Budapest, 2015

Tartalomjegyzék

Bevezetés

Szakdolgozatom témájának megválasztásakor a fő cél az volt, hogy egy olyan alkalmazást készítsek, ami olyan technológiák segítségével van megvalósítva, amiket az iparban is használnak. A választásom ezért egy webalkalmazás fejlesztésére esett és mivel ez mellett rajongok a Forma-1-ért, ezért a kettőt ötvöztem és a webalkalmazás témája egy Forma-1 manager játék lett. A játék bárki számára elérhető, aki rendelkezik internet kapcsolattal. A felhasználóknak először regisztrálniuk kell, majd aktiválni az e-mail-ben kapott aktivációs kód segítségével. Ezután a játékos kap egy kezdőösszeget és vásárolhat az elérhető csapatok és pilóták közül, akik képviselni fogják a következő futamon és ez által a játékos pontot és pénzt szerezhet. A játék párhuzamosan fut a valós Forma-1 bajnoksággal. A pilóták és csapatok ára az alapján változik, hogy a futamon milyen helyezést értek el. A felhasználók saját ligákat hozhatnak létre, hogy a ligában szereplők eredményeit minél gyorsabban láthassák. Minden futam között látható, hogy az előző futamokon a játékos mennyi pontot, pénzt szerzett vagy milyen pilótafelállással szerepelt. A programhoz tartozik egy adminisztrációs felület, ahol a játék adatait lehet beállítani, de ezt csak a megfelelő jogosultsággal rendelkezőek érhetik el.

A programozási nyelv és a szerveroldali keretrendszer kiválasztásakor a legfontosabb szempontok többek között a robosztusság, modularitás, megbízhatóság volt, ezért a Spring Framework mellett döntöttem. A felhasználói felület elkészítéséhez a Bootstrap és jQuery keretrendszereket választottam.

Felhasználói dokumentáció

Adminisztrátorok számára

Az alkalmazás futtatásához szükséges programok szerveroldalon:

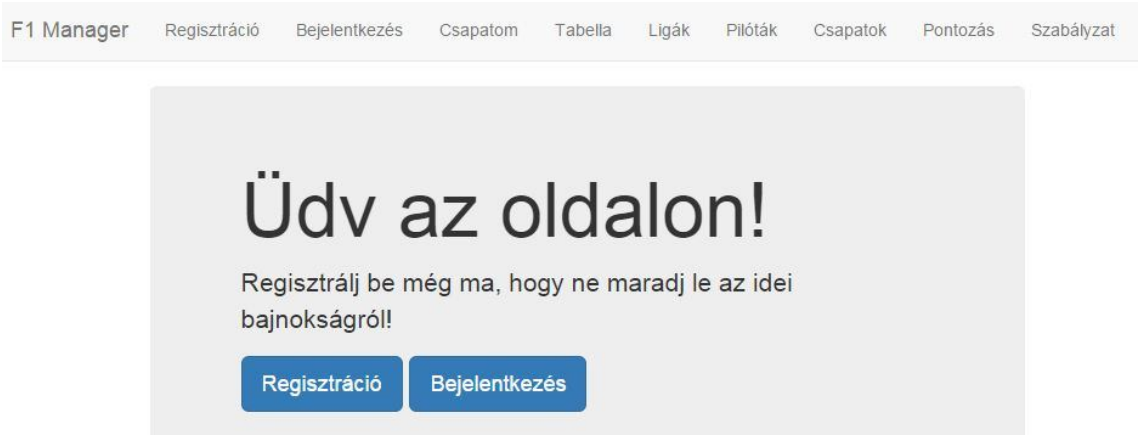
- Java 1.7
- MySQL 5.5
- Tomcat 7 vagy másik alkalmazásszerver, ami képes Java servletek futtatására

Miután az előző szoftverek telepítése megtörtént, létre kell hozni egy „szakdolgozat” nevű adatbázist. Továbbá az alkalmazásból generálni kell egy „war” fájlt a Maven segítségével, amit már telepíteni lehet az alkalmazásszerverre. Az adatbázis sémát a Hibernate telepítés során automatikusan létrehozza és beállítja a megfelelő paramétereket.
Ezek után az alkalmazás használható.

Felhasználók számára

A felhasználóknak rendelkeznie kell egy internethez kapcsolódó eszközzel, mint pl: mobiltelefon, számítógép, tablet, amelyen tetszőleges internetböngésző futtatható. A weboldal a legtöbb böngészőben működik, és optimális méretben jelenik meg, azaz a weboldal alkalmazkodik a kijelző felbontásához.

Az oldal betöltése után a felhasználót a következő kép fogadja:
Bejelentkezés előtt lévő oldal (1. ábra):

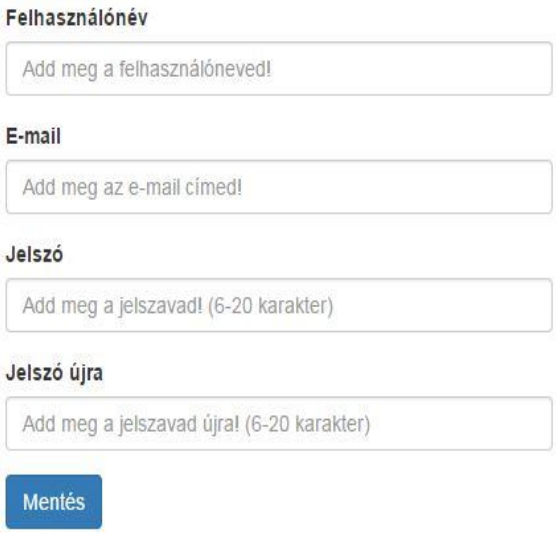


1. ábra

A felhasználói felület

Regisztrációs oldal

Az oldal új látogatóinak lehetősége van az oldalra regisztrálnia, amit egy regisztrációs űrlap segítségével lehet megtenni (2. ábra).
A regisztrálás során szükség van egyedi (még nem regisztrált) felhasználónévre, egyedi e-mail címre és jelszóhoz a bejelentkezéshez.
Ha a felhasználónév vagy e-mail cím már szerepel a rendszerben, akkor a felhasználó azonnali visszajelzést kap erről még gépelés közben. Ha a jelszavak nem egyeznek vagy



2. ábra

túl rövidek, akkor erről is jelzés érkezik.

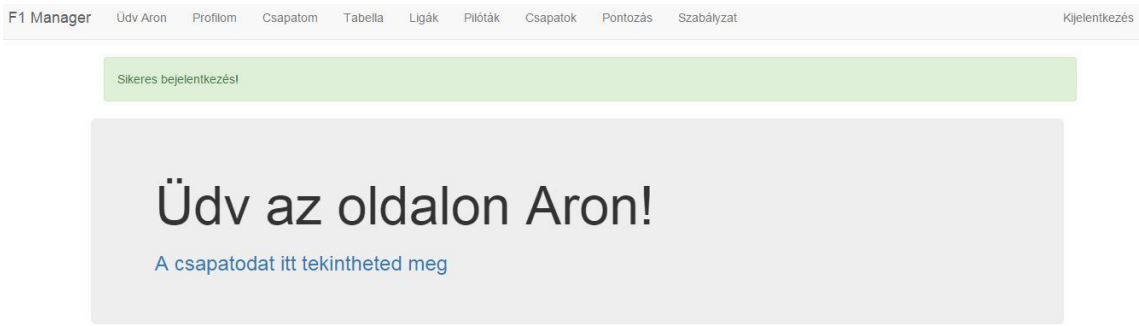
Regisztráció sikeressége után szükség van a megadott e-mail cím megerősítésére. Ezt a megadott e-mail címre küldött hivatkozásra kattintva lehet megtenni. Ha az aktivációs e-mail elveszne, akkor a Bejelentkezés oldalon új igényelhető.

Bejelentkezés oldal

Ezen az oldalon tudnak a felhasználók bejelentkezni a fiókjukba a felhasználónevük és jelszavuk megadásával. Ha a bejelentkezés sikeres volt, de a felhasználó még nem erősítette meg az e-mail címet, akkor ezt itt kell megtennie. Ha elvesztette az aktivációs kódot, itt lehet újat igényelni.

Ha a felhasználó elfelejtette a jelszavát, akkor a már regisztrált e-mail cím megadásával lehet újat igényelni. Ekkor a rendszer elküld egy levelet, ami tartalmazza, azt a hivatkozást, ahol a felhasználó megváltoztathatja a jelszavát.

Az oldal a sikeres bejelentkezés után (3. ábra)



3. ábra

Csapatom oldal

Ennek az oldalnak a tartalmát a felhasználó csak akkor látja, ha már bejelentkezett.
Itt lehet megtekinteni, hogy jelenleg mennyi pénz áll a felhasználó rendelkezésére, a bajnokságban eddig mennyi pontot szerzett és az előző futamokon milyen felállással indult.
Ezen felül itt lehet eladni és vásárolni pilótákat, csapatokat. Összesen két pilótát és három csapatot lehet vásárolni. Nem lehet ugyanazt a pilótát vagy csapatot többször is

megvenni. Ha ezt a felhasználó megpróbálja, akkor az oldal kiírja a felhasználó számára a megfelelő üzenetet. Továbbá a két pilóta nem tartozhat ugyanabba a csapatba.

Tabella oldal

Ezen az oldalon tekinthető meg az összes regisztrált felhasználó az aktuális bajnokságban lévő helyezésükkel és további adatokkal.
Ha az összes felhasználó száma túllépi az egy oldalon megjeleníthető felhasználók számát, akkor több oldalon keresztül lehet megtekinteni az felhasználókat, azaz a táblázat lapozható. A felhasználónév mellett csak kevés adat jelenik meg ilyenkor, de a felhasználónévre kattintva megtekinthető a teljes profil. Lehetőség van felhasználónév alapján szűrni a lista tartalmát. Ilyenkor csak azok a felhasználók fognak szerepelni, akik neve tartalmazza a megadott karaktersorozatot.

Ligák oldal

Bejelentkezés előtt csak az összes ligát lehet megtekinteni és egy liga nevére kattintva jön be az adott ligához tartozó oldal.

Bejelentkezés után már egy külön gombra kattintva lehet elérni azokat a ligákat, amikben a bejelentkezett felhasználó szerepel.

Egy harmadik gomb segítségével a felhasználó létrehozhat saját ligát is. Itt a ligának egyedi nevet kell adni, azaz nem lehet a rendszerben másik ugyanilyen nevű liga, ezen felül egy rövid leírást is kell adni. Amelyik felhasználó létrehozta a ligát az lesz a liga adminisztrátora. Nem léphet ki a ligából és joga van bármelyik felhasználót kirakni abból. A liga minden tagjának lehetősége van meghívni bármelyik felhasználót, aki még nincs benne az adott ligában. Ilyenkor név alapján tud rákeresni a felhasználókra és a talált felhasználókat meg tudja hívni e-mailen keresztül. Ilyenkor a rendszer elküld egy levelet a meghívott játékosnak, aki a levélben található hivatkozással csatlakozhat vagy figyelmen kívül hagyhatja ezt.

Pilóták és Csapatok oldalak

Itt lehet megtekinteni egy táblázatban, hogy az aktuális bajnokságban milyen pilóták és csapatok közül lehet válogatni, illetve itt látszódik, hogy a bajnokságban az adott pilóta vagy csapat mennyi pontot ért el és mennyibe kerül.

Pontozás oldal

Itt tekinthető meg, hogy adott helyezését mennyi pont jár.

Szabályzat oldal

Erre az oldalra kattintva lehet az oldalhoz tartozó részletes szabályzatot elolvasni, hogy az oldalt látogató új tagok is tudják, hogy mi a játék lényege. és adott futamok alapján ki mennyi pontot és pénzt kap.

Kijelentkezés

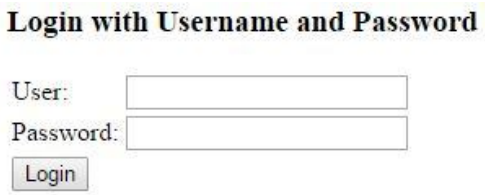
A bejelentkezés után a jobb felső sarokban elérhető kijelentkezés menüpontra kattintva az oldal kijelentkezteti a felhasználót és a kezdőoldalra továbbítja.

Az oldalhoz tartozó adminisztrációs felület

Az oldalhoz tartozó admin felületre csak a megfelelő felhasználónév jelszó párossal rendelkezőek tudnak bejelentkezni.
Egy funkció sem érhető el a bejelentkezés előtt, ha azonban ezt valaki mégis megkíséreli, akkor az oldal továbbít a bejelentkező oldalra.

Bejelentkező felület

Ha a megfelelő felhasználónév és jelszót megadjuk a felületen, akkor az oldal továbbít az admin oldalra, különben hibát jelez és megkéri a felhasználót, hogy próbáljon meg újra bejelentkezni.



4. ábra

Ha a bejelentkezés sikeres volt, akkor a következő felület fogadja az adminisztrátort:

5. ábra

Csapatok menüpont

Az első választási lehetőség az összes csapat listázása. Ilyenkor egy táblázatban látható az összes csapat. A táblázat minden sorában szerepel egy „Szerkeszt” gomb, amire kattintva megnyílik az adott csapathoz tartozó adatokat szerkesztő oldal. Egy csapat esetén négy adatot kell megadni. A csapat nevét, árát, a hozzátartozó kép URL címét és azt, hogy az idei bajnokságban megvásárolható-e. A csapat nevének egyedinek kell lennie. Az árat a bajnokság elején érdemes beállítani, mert minden futam eredményének feltöltése után a rendszer automatikusan növeli vagy csökkenti a megfelelő mennyiséggel. A képhez tartozó mezőt üresen lehet hagyni vagy érvényes URL címet kell megadni, ami egy képhez tartozó elérhetőséget tartalmaz. Egy csapat nem törölhető, helyette beállítható, hogy az aktuális bajnokságban a felhasználók megvásárolhatják-e. Ennek az oka, hogy a rendszer menti a felhasználók régi eredményeit is, és bármikor visszanézhetik, hogy mikor melyik csapattal rendelkeztek attól függetlenül, hogy az aktuális bajnokságban indult a csapat vagy sem.
A második választási lehetőség az új csapat létrehozása. Ebben az esetben egy teljesen új csapatot vesszünk fel a rendszerbe. Ilyenkor a felvett adatokra ugyanazok a szabályok érvényesek, mint módosításkor.

Pilóták menüpont

Itt is meg lehet tekinteni a már meglévő pilótákat vagy újat lehet felvenni. A különbség a pilóták és csapatok között, hogy egy pilótához hozzá kell rendelni egy csapatot is. A csapat, amit szeretnénk beállítani a pilótának az egy listából választható ki.

Pályák menüpont

A „Pályák listázása” menüpont alatt a már megszokott táblázatban láthatjuk a rendszerben lévő pályákat és itt választhatjuk ki, hogy melyiket szeretnénk szerkeszteni.

Új pálya felvétele esetén négy adatot lehet megadni. A pálya neve, aminek egyedinek kell lennie és legfeljebb 100 karakterből állhat. Ország és város nevének kitöltése kötelező. Pályák esetén is lehetőség van kép felvételére, ilyenkor érvényes URL címet kell megadni.

Futamok menüpont

A futamok listázása menüpont alatt a futamok pontos dátuma, helyszíne és státusza látható, ami azt jelzi, hogy a futamhoz már lett eredmény állítva vagy még nem. Továbbá elérhető egy „Eredmény" gomb is, ami csak akkor jelenik meg, ha már lett állítva eredmény az adott futamhoz. A gombra kattintva lehet megtekinteni, hogy az időmérő edzésen és a versenyen ki milyen helyezést ért el a pilóták közül.
Új futam felvétele esetén három adatot kell megadni. Az első adat a futam időpontja, aminek egyedinek kell lennie, mert egy napon nem szoktak több versenyt szervezni. Két futam között 1-4 hét szokott eltelni egy bajnokságban. Továbbá a bajnokság évszámát és a pálya nevét ahol a futam megrendezésre kerül. Ezt a két adatot a már rendszerben lévő bajnokságokból és pályákból válogathatjuk ki.

Bajnokság menüpont

A bajnokságok listázása alatt az összes bajnokságot láthatjuk.
Új bajnokság menüpont alatt pedig újat hozhatunk létre. Egy bajnokság felvételéhez csak az adott évet kell megadni, aminek egyedinek kell lennie. A bajnokságok éve később is szerkeszthető, ha rosszul lett megadva.

Eredmények menüpont

Ezen belül két menüpont érhető el:
Az első esetén egy adott verseny végeredményét állíthatjuk be. Itt kell beállítani, hogy melyik pilóta hol végzett az időmérőn és versenyen. A csapatokat nem kell megadni, mert minden pilótának tartoznia kell egy csapatba és a rendszer tudja, hogy melyik pilóta melyik csapatban van, ezért a megadott pilóták alapján beállítja a csapatokhoz tartozó értékeket is.

Ugyanaz a pilóta nem érhet el két helyezést egy időmérő edzésen vagy versenyen, ha mégis ilyen adatot próbálnánk menteni, akkor a rendszer figyelmeztet erre minket. Az eredményt figyelmesen kell beállítani, mert ez később már nem módosítható. Miután beállítottuk a helyezéseket és a „Mentés” gombra nyomunk akkor a rendszer több adatot is beállít. Először beállítja a helyezések alapján minden csapathoz és pilótához a szerzett pont és pénz mennyiségét. Utána ezen értékek alapján kiosztja a díjakat a felhasználók számára és utána kiszámolja, hogy a futam után melyik felhasználó hányadik a bajnokságban.
Hogy a felhasználók a későbbiekben is megtekinthessék mikor milyen felállással és milyen sikerességgel vettek részt a futamokon, ezért elmentjük ezeket az adatokat is.

A második menüpontban lehet egy táblázatban beállítani, hogy egy pilóta/csapat mennyi pontot szerez egy időmérőn/futamon elért helyezésért. Továbbá beállítható egy „százalék” érték is, ami azt a célt szolgálja, hogy a pilóta/csapat értéke a helyezéshez viszonyítva megfelelő mértékben növekedjen vagy csökkenjen.

Excel menüpont

Ez az oldal arra a célra szolgál, hogy ha egy időben szeretnénk sok csapat tulajdonságát módosítani vagy új csapatokat felvenni, akkor minél gyorsabban és könnyebben lehessen kivitelező. Ehhez egy Excel fájl feltöltése szükséges, aminek megfelelő formátumban kell tartalmaznia a csapatokhoz tartozó adatokat. Az Excel-be ugyanazokat az adatok kell megadni, mint a weboldalon történő szerkesztés esetén. A csapat nevét, értékét, képét,megszerzett pontok számát és azt, hogy az aktuális bajnokságban a csapat indul-e. Ehhez a fájlban hat oszlopot kell felvennünk. Az oszlopok első sorai a következőeknek kell lennie tetszőleges sorrendben: „Id”, „Név”, „Ár”, „Kép”, „Pont” és „Aktív”. Ha bármelyik oszlop nincs megadva vagy hibásan van gépelve, akkor a rendszer nem fogadja el a fájlt és erről tájékoztat minket. Az oldalról letölthető egy üres fájl, ami már tartalmazza ezt a struktúrát kialakítva, hogy a felhasználónak ne magának kelljen létrehoznia. Ha már a rendszerben szereplő csapatot szeretnénk módosítani, akkor az „Id” oszlopban kell megadni a csapathoz tartozó azonosítót, hogy a rendszer tudja, hogy ez már egy meglévő csapat módosítása. Ha egy olyan azonosítót adunk meg, ami nem létezik az adatbázisban, akkor a rendszer erről

szintén figyelmeztet. Új csapat esetén ezt a cellát a sorban üresen kell hagyni. Ha a csapathoz tartozó bármelyik érték nem felel meg a validálási szempontoknak (lásd: Csapatok menüpont részletezése), akkor a rendszer erről tájékoztatást ad. Az „Aktív” oszlopba 0 vagy 1 érték írható. A 0 jelentése, hogy a csapat nem indult az idei bajnokságban, míg az 1 jelentése, hogy igen.

A harmadik opció esetében egy Excel fájl tölthető le, ami tartalmazza a rendszerben lévő összes csapatot a hozzátartozó adatokkal együtt.

Az új csapatok vagy már meglévőek módosítása előtt mindenképpen ajánlott letölteni valamelyik előre készített Excel fájlt.

A kijelentkezés gomb

Erre a gombra kattintva a bejelentkezett adminisztrátor ki tud jelentkezni a felületről és visszatérni az oldalhoz tartozó bejelentkezés oldalra.

Fejlesztői dokumentáció

Az alkalmazás fejlesztéséhez használt technológiák

A szerveroldalhoz a Spring framework-öt használtam. A keretrendszer 2002-ben jelent meg és jelenleg az iparban az egyik legnépszerűbb keretrendszer Java alkalmazások fejlesztéséhez. A Java EE szabvány hiányosságait próbálták vele helyettesíteni, ami sok esetben sikerült is, ezek később a standard szabványra is hatással voltak.

Az alkalmazás mögött lévő adatbázis manipulációhoz a Hibernate-re esett a választásom. Ez egy objektum-relációs (ORM) leképezést megvalósító programkönyvtár. Az ORM technológia nagy előnye, hogy az adatbázist osztályokon keresztül tudjuk kezelni és nincs szükség SQL utasítások írására, ezért a kód eltérő adatbázis-kezelő rendszerekkel is tud működni és rossz esetben is minimális módosításokra van csak szükség.

A szerver futtatásához szükség van egy alkalmazásszerverre is, amiből több fajta is létezik. A szakdolgozatomhoz a Tomcat alkalmazásszervert használtam, ami ingyenes és egyszerűen konfigurálható.

Sok funkcióval rendelkező szoftvereknél elengedhetetlen a tesztelés is. Ebben a JUnit és a Spring Test volt a segítségemre.

Továbbá szükség van arra is, hogy naplózzuk a rendszer hibáit, ami alapján meg tudjuk keresni a hibákat és javítani tudjuk őket, ehhez a log4j-t használtam.

A dinamikus oldalak előállításához szükség van egy template engine-re is, ami arra a célra szolgál, hogy html kódot tudjunk dinamikusan generálni. Ehhez az Apache Velocity-t választottam, aminek a szintaktikája nagyon letisztult és a bővítése is egyszerű.

Adatbázis

Az alkalmazásban folyamatosan adatbázisban lévő adatok manipulációja történik. A következő táblák és a hozzájuk tartozó entitások szerepelnek az alkalmazásban. Az adatbázis 12 táblából áll, ezek a következőek.

User tábla

Ebben a táblában találhatóak a felhasználó legfontosabb adatai

Mező	**Típus**	**Leírás**
id	bigint(20)	A felhasználó azonosítója
activated	bit	A fiók aktiválva lett-e az e-mailben kapott azonosító segítségével
activation_code	varchar(255)	Az e-mailben küldött aktivációs kód
actual_money	bigint(20)	A felhasználó aktuális pénze
actual_point	bigint(20)	A felhasználó aktuális pontjainak száma
change_password_token	varchar(255)	Az elfelejtett jelszóhoz tárolt token
email	varchar(100)	A felhasználó e-mail címe
name	varchar(100)	A felhasználónév
registration_date	date	A regisztráció időpontja
actual_driver1_id	bigint(20)	A felhasználó elsőszámú pilótája
actual_driver2_id	bigint(20)	A felhasználó másodszámú pilótája
actual_team1_id	bigint(20)	A felhasználó elsőszámú csapata
actual_team2_id	bigint(20)	A felhasználó másodszámú csapata

actual_team3_id	bigint(20)	A felhasználó harmadszámú csapata
password	varchar(255)	A felhasználó jelszava titkosítva, hashben tárolva
actual_position	bigint(20)	A felhasználó aktuális pozíciója a bajnokságban

Team tábla

Ebben a táblában találhatóak a csapatok legfontosabb adatai

Mező	**Típus**	**Leírás**
id	bigint(20)	A csapat azonosítója
name	varchar(100)	A csapat neve
picture	longtext	A csapathoz tartozó kép URL címe
point	int(11)	A csapat aktuális pontszáma
price	bigint(20)	A csapat aktuális értéke
active	bit	A csapat részt vesz-e az aktuális bajnokságban

Driver tábla

Itt találhatóak a pilóták legfontosabb adatai.

Mező	**Típus**	**Leírás**
id	bigint(20)	A pilóta azonosítója
name	varchar(100)	A pilóta neve
picture	longtext	A pilótához tartozó kép URL címe
point	int(11)	A pilóta aktuális pontszáma

price	bigint(20)	A pilóta aktuális értéke
active	bit	A pilóta részt vesz-e az aktuális bajnokságban
team_id	bigint(20)	A pilóta csapatának azonosítója

Championship tábla

A bajnokság legfontosabb adatai:

Mező	**Típus**	**Leírás**
id	bigint(20)	A bajnokság azonosítója
year	int(11)	A bajnokság éve
winner_team_id	bigint(20)	A győztes csapat
winner_driver_id	bigint(20)	A győztes pilóta
winner_user_id	bigint(20)	A győztes felhasználó

League tábla

Egy ligához tartozó legfontosabb adatok:

Mező	**Típus**	**Leírás**
id	bigint(20)	A liga azonosítója
name	varchar(100)	A liga neve
description	longtext	A liga részletes leírása
creator_user_id	bigint(20)	A ligát létrehozó felhasználó azonosítója
number_of_users	smallint(6)	A ligában tartózkodó felhasználók száma
date	date	A liga létrehozásának dátuma

User_in_league tábla

Ebben a táblában található azok az információk, hogy egy felhasználónak milyen kapcsolata van egy olyan ligában, amiben szerepel.

Mező	Típus	Leírás
id	bigint(20)	A kapcsolat azonosítója
league_id	bigint(20)	A liga azonosítója
user_id	bigint(20)	A felhasználó azonosítója
role	bit	A ligában lévő felhasználó hatásköre
join_date	date	A felhasználó ligába való belépésének dátuma

Track tábla

Ebben a táblában található egy pályához tartozó információk

Mező	Típus	Leírás
id	bigint(20)	A pálya azonosítója
name	varchar(100)	A pálya neve
country	varchar(100)	A pálya melyik országban található
city	varchar(100)	A pálya melyik városban található
picture	longtext	A pályához tartozó kép URL címe

Race tábla

Egy versenyhez tartozó adatok:

Mező	Típus	Leírás
id	bigint(20)	A futam azonosítója
date	date	A futam dátuma
track_id	bigint(20)	A pálya azonosítója ahol a futam megrendezésre kerül
championship_id	bigint(20)	A bajnokság azonosítója, amelyikben a futam megrendezésre kerül
is_result_set	bit	A futamhoz lett-e állítva már eredmény

Result_point tábla

Ez az a tábla, amiben megtalálhatóak azok az értékek, hogy egy futam után a csapatok és pilóták és ez által a felhasználók mennyi pontot kapnak.

Mező	Típus	Leírás
id	bigint(20)	A pontozás azonosítója
result	int(11)	A helyezés a futamon
driver_race_point	int(11)	A helyezéshez tartozó pontszám a futamon, amit a pilóta kap
driver_qualification_point	int(11)	A helyezéshez tartozó pontszám az időmérő edzésen, amit a pilóta kap
team_race_point	int(11)	A helyezéshez tartozó pontszám a futamon, amit a csapat kap
team_qualification_point	int(11)	A helyezéshez tartozó pontszám az időmérő

		edzésen, amit a csapat kap
rate	int(11)	A helyezéshez tartozó érték ami szerint a csapat/pilóta aktuális értéke csökken/nő

Result_qualification tábla

Az időmérő edzéséhez tartozó eredményeket leíró tábla

Mező	**Típus**	**Leírás**
id	bigint(20)	Az eredmény azonosítója
result	int(11)	Az eredmény az időmérő edzésen
race_id	bigint(20)	A futam azonosítója
driver_id	bigint(20)	A pilóta azonosítója
team_id	bigint(20)	A csapat azonosítója

Result_race tábla

A futamhoz tartozó eredményeket leíró tábla

Mező	**Típus**	**Leírás**
id	bigint(20)	Az eredmény azonosítója
result	int(11)	Az eredmény a versenyen
race_id	bigint(20)	A futam azonosítója
driver_id	bigint(20)	A pilóta azonosítója
team_id	bigint(20)	A csapat azonosítója

User_result_history tábla

Ebben a táblában találhatóak meg azok az információk, hogy egy futamon egy felhasználó milyen felállással és milyen eredményekkel vett részt.

Mező	Típus	Leírás
id	bigint(20)	Az eredmény azonosítója
user_id	bigint(20)	A felhasználó azonosítója
driver1_id	bigint(20)	A felhasználó elsőszámú pilótájának azonosítója
driver2_id	bigint(20)	A felhasználó másodszámú pilótájának azonosítója
team1_id	bigint(20)	A felhasználó elsőszámú csapatának azonosítója
team2_id	bigint(20)	A felhasználó másodszámú csapatának azonosítója
team3_id	bigint(20)	A felhasználó harmadszámú csapatának azonosítója
point	bigint(20)	A futamon szerzett pontok száma
money	bigint(20)	A futamon szerzett pénz
position	bigint(20)	A futam utáni pozíció a bajnokságban

Entitások

A kódban minden táblának van egy megfelelő entitás.
Az entitás egy olyan Java osztály, ami az adatbázis egy tábláját reprezentálja.

Az adatbázis műveleteket a Hibernate Criteria API segítségével implementáltam, ami lehetővé tette számomra, hogy az SQL utasítások helyett objektumorientáltan függvények segítségével vezéreljem az adatbázist.

Az alkalmazásban használtam a DAO (data access object) architektúra mintát, ami arra a célra szolgál, hogy az üzleti logika és az adatbázison végrehajtódó művelet külön rétegbe kerüljenek. Az üzleti logikát leíró függvények a Service rétegben találhatóak meg és ezekből kerül meghívásra a DAO réteg, ami az adatbázis rekordjain végrehajtja a tényleges módosításokat. Ennek többek között nagy előnye, hogy ha a későbbiekben Hibernate helyett másik ORM könyvtárat szeretnék használni, akkor elég csak a DAO rétegben szereplő metódusokat átírnom.

Konfigurációs fájlok

Az alkalmazás elkészítése során használtam a Maven-t, ami szoftverprojektek menedzselésére nyújt segítséget.
A pom.xml fájlban találhatóak a buildelési folyamatokat leíró utasítások és a projekt függőségei.

A naplózáshoz található beállítások a log4j.xml-ben találhatóak. A konfigurációs fájlban megadtam, hogy a naplófájlok létrehozása hol történjen meg és milyen gyakran jöjjenek létre új naplófájlok. Minden nap új naplófájl indul, ezzel jobban átláthatóvá teszi a hibák keresését.

A Java-ban készített webalkalmazásoknak szükségük van egy web.xml fájlra is, ami egyfajta telepítési útmutatóként szolgál. Itt beállítottam a kódolást és megadtam a Spring és Spring Security-hez tartozó xml fájlok útvonalát is.

A Springhez tartozó konfigurációk a servlet-context.xml fájlban találhatóak.
Ebben a fájlban található többek között az adatbázishoz való csatlakozást leíró beállítás is.

A Spring Security-hez tartozó spring-security.xml fájlban lett beállítva, hogy az adminisztrációs felület csak bejelentkezés után érhető el és itt adtam meg azt a felhasználónév jelszó párost, amivel ellehet érni ezt a felületet is.

További properties fájlok

Az application.properties fájlban olyan adatokat adtam meg, amit nem szerettem volna a Java osztályokba égetni, se adatbázisba menteni.

A database.properties fájlban írtam le, hogy az adatbázis, amihez az alkalmazás csatlakozik az hol található és a csatlakozáshoz milyen információk szükségesek. Ezeket az adatokat a servlet-context.xml használja fel.

Továbbá webalkalmazásoknál fontos, hogy a felület több nyelven is elérhető legyen. Ezt úgy tudjuk elérni, ha az alkalmazáshoz tartozó nézetekbe nem direkt írjuk be a szövegeket, hanem kiszervezzük külön fájlokba vagy adatbázisba minden egyes nyelvhez tartozó értékeket. A Spring-hez tartozó xml fájlban beállítottam, hogy jelenleg a különböző nyelvekhez tartozó értékeket melyik properties fájlokban keresse.

A messages_hu.properties fájl tartalmazza a nézetekben található tokenekhez a magyar nyelvű értékeket.
Jelenleg az oldal csak magyarul használható, de új nyelv felvétele a properties fájlok miatt nagyon egyszerűen és gyorsan tehető meg.

Spring MVC

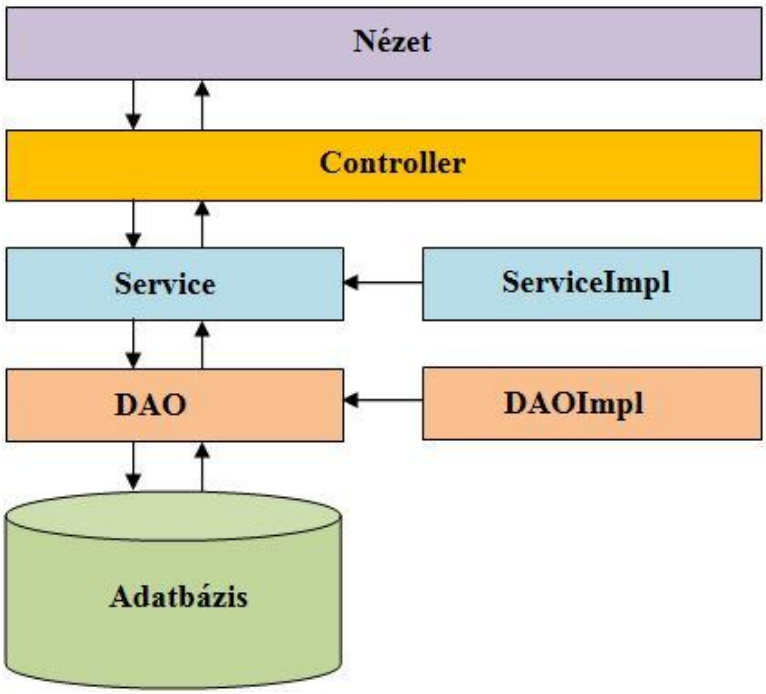
A keretrendszer a Spring MVC modul integrálásával lehetővé teszi az MVC (Model-View-Controller) architektúra minta szerinti fejlesztést. Ennek a lényege, hogy az alkalmazás három fő részre tagolódik:

1: Model: A modellekben reprezentáljuk az adatbázis táblákat és itt implementáljuk az üzleti logikát.

2: View: Nézetekben szerepelnek az alkalmazáshoz tartozó oldalak kinézetei.

3: Controller: A bejövő adatokat a modellek segítségével feldolgozza, majd az új adatokat a nézet segítségével megjeleníti.

Az előző architektúra minták következtében egy akció, amit a felhasználó kiad az oldalon, a következő hívási láncot indítja el a szerveren (6. ábra).



6. ábra

Első lépésben a felhasználó kattint az oldalon. Ekkor a felhasználó kérése első lépésben a Controller rétegbe vezet. Innen kerül meghívásra a Service réteg egy Interface-n keresztül. A Service rétegben végbemegy az üzleti logika, majd ezután meghívásra kerül a DAO réteg szintén egy Interface-n keresztül, ami közvetlen kapcsolatot létesít az adatbázissal, majd a felhasználó látja az oldalon a végeredményt.

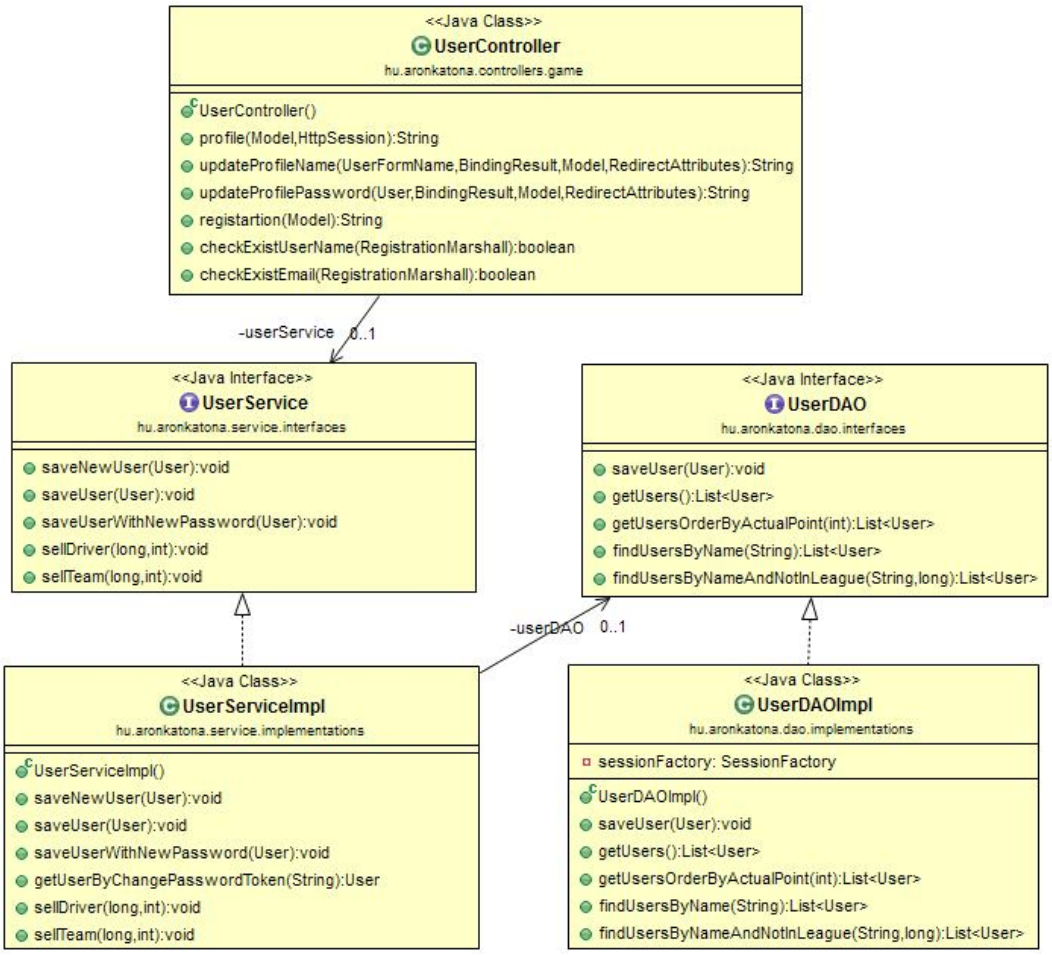
Vezérlők az alkalmazásban

A vezérlőket két külön csomagba helyeztem el. Az egyik csomagban a felhasználó felülethez tartozó vezérlők szerepelnek, míg a másikban az adminisztrátori felületért felelősek. A vezérlők visszatérhetnek nézetekkel vagy JSON adatstruktúrával.
A nézetek szintén két külön csomagban találhatóak. Az egyik csoport a felhasználói, a másik az admin felülethez tartozó nézeteket tartalmazza.

A felhasználói felület vezérlői és a hozzátartozó nézetek

UserController

A UserController osztályban találhatóak meg azok a függvények, amik a felhasználóval kapcsolatos módosításokért felelősek:
Az alábbi diagramon látható a vezérlőhöz tartozó osztályok és függvények egy része.

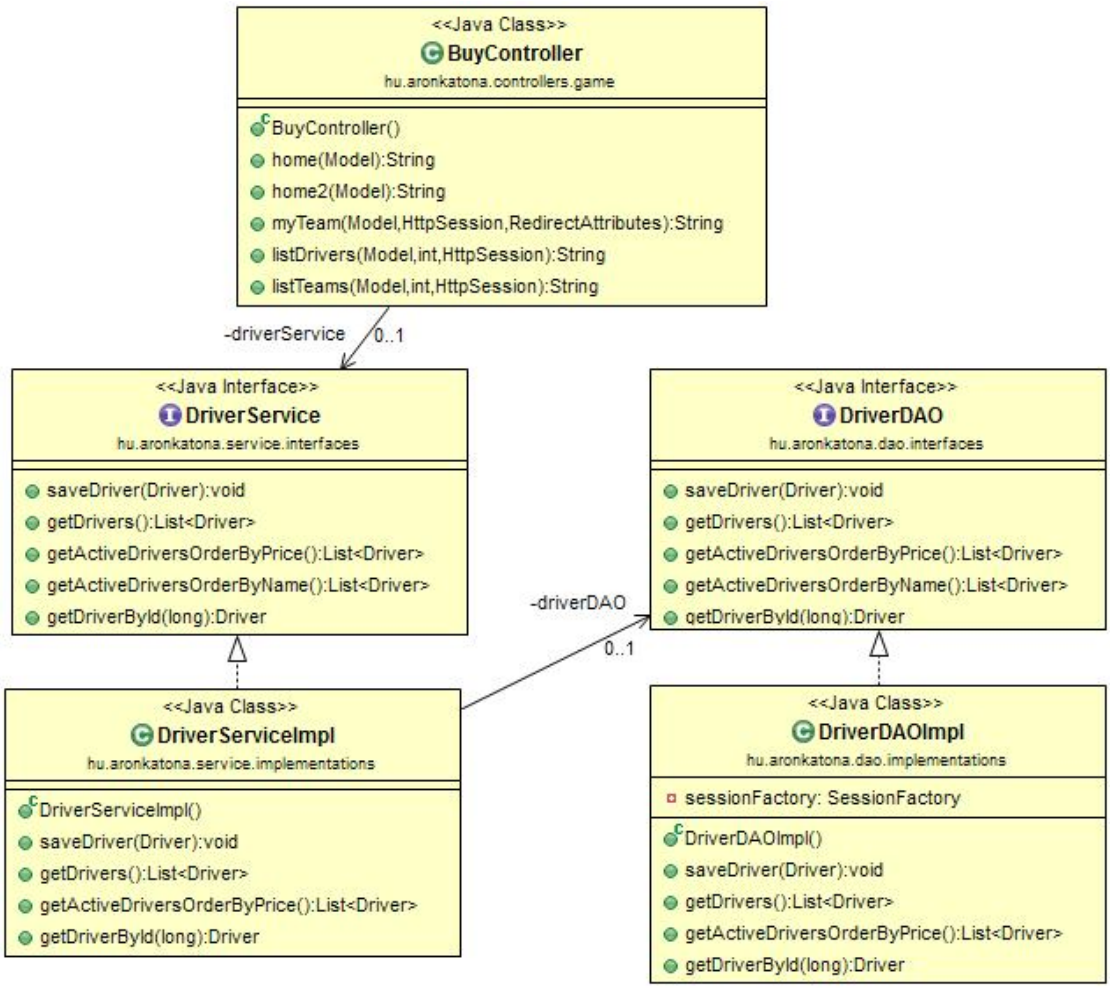


7. ábra

1) **registration**: Visszatér a „registration” nézettel ahol a felhasználó regisztrálni tud az oldalon.
2) **saveUser**: Ha a felhasználó kitöltötte a regisztrációs űrlapot, akkor az adatok ide érkeznek be. Ha minden adat megfelel a validálási feltételeknek, akkor a felhasználót felvesszük az adatbázisba, majd tájékoztatjuk a sikeres regisztrációról.
3) **checkExistUserName**: A regisztráció során, azért hogy a felhasználó minél gyorsabban tájékoztatást kapjon arról, hogy a rendszerben már szerepel-e ilyen nevű felhasználó Ajax hívást alkalmazunk, amikor leüt egy újabb karaktert. A függvény egy igaz vagy hamis értékkel tér vissza.
4) **checkExistEmail**: Az előzőhöz hasonlóan viselkedik annyi különbséggel, hogy a szervertől csak akkor kérdezi meg, hogy létezik-e már az e-mail cím, ha a megadott e-mail cím érvényes.
5) **login**: Ez a metódus dönti el, hogy a megadott felhasználónév és jelszó páros megtalálható-e a rendszerben. Ha megtalálható, de az e-mail cím még nem került megerősítésre, akkor az „activateAccount” nézettel tér vissza és megkéri a felhasználót, hogy aktiválja az e-mail címét. Ha megtalálható és már aktivált, akkor a „home” nézettel tér vissza és elérhetővé tesz a felhasználónak további funkciókat is az oldalon.
6) **logout**: Ez az akció jelentkezteti ki a felhasználót az oldalról és a kezdőoldalra továbbítja.

BuyController

Ebben a vezérlőben találhatóak a felhasználó vásárlási és eladásáért felelős akciók.

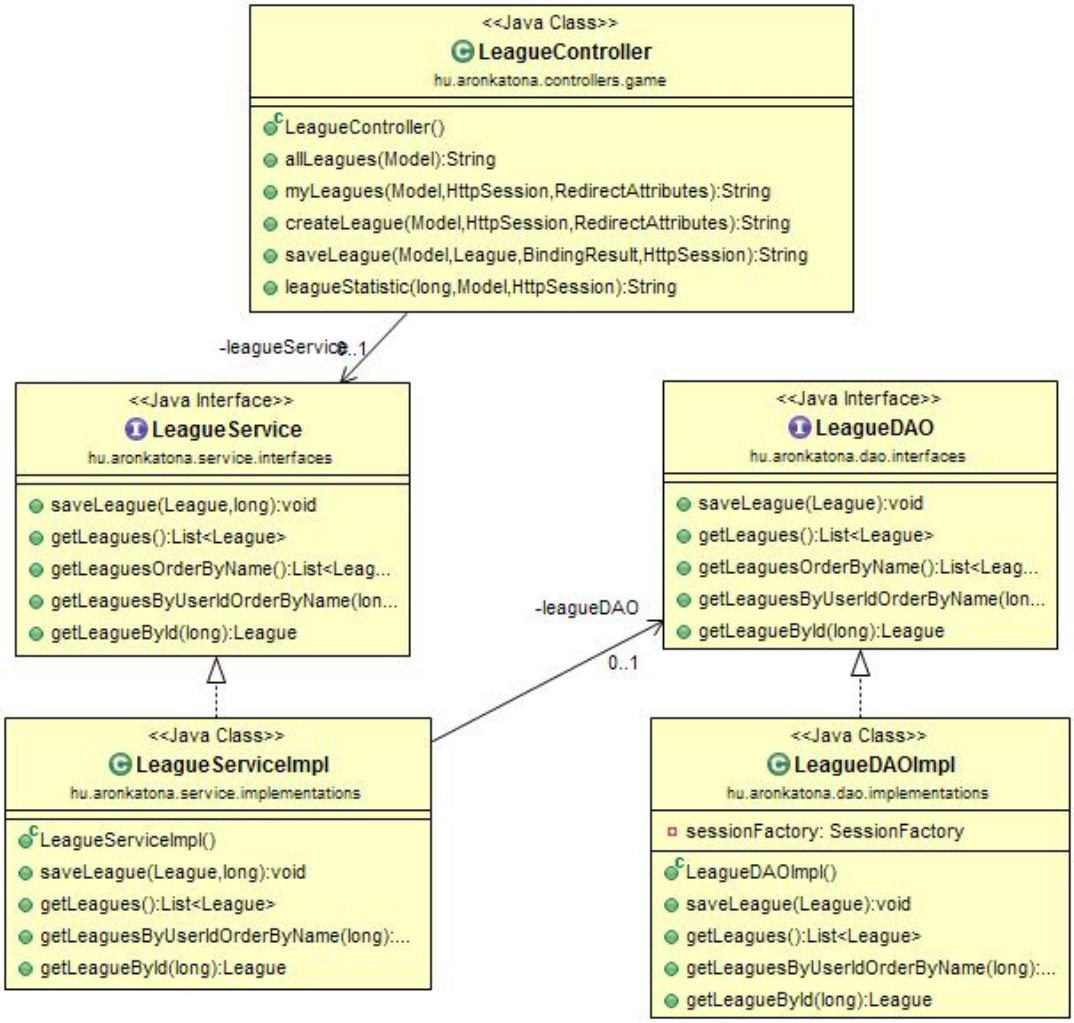


8. ábra

1) **myTeam**: A metódus lekéri a már bejelentkezett felhasználóhoz tartozó adatokat és visszatér a „myTeam” nézettel.
2) **listDrivers**: A vásárlás gombra kattintva ez a metódus fut le és listázza az oldalon az elérhető pilótákat, amik közül a felhasználó válogathat.
3) **listTeams**: Az előzőhez hasonlóan, csak a csapatokkal.
4) **buyDriver**: Ez a függvény felel a pilóta vásárlásért. Ha a felhasználónak minden feltétele adott ahhoz, hogy megvehesse a kiválasztott pilótát, akkor a vásárlás sikeres volt, ellenkező esetben tájékoztatja a felhasználót.
5) **buyTeam**: Az előzőhez hasonlóan, csak csapatra vonatkozóan.
6) **sellDriver**: Ez a függvény adja el a kiválasztott pilótát.
7) **sellTeam**: A felhasználó által kiválasztott csapatot adja el a függvény.

LeagueController

Ebben a controllerben találhatóak a ligával kapcsolatos eljárások.



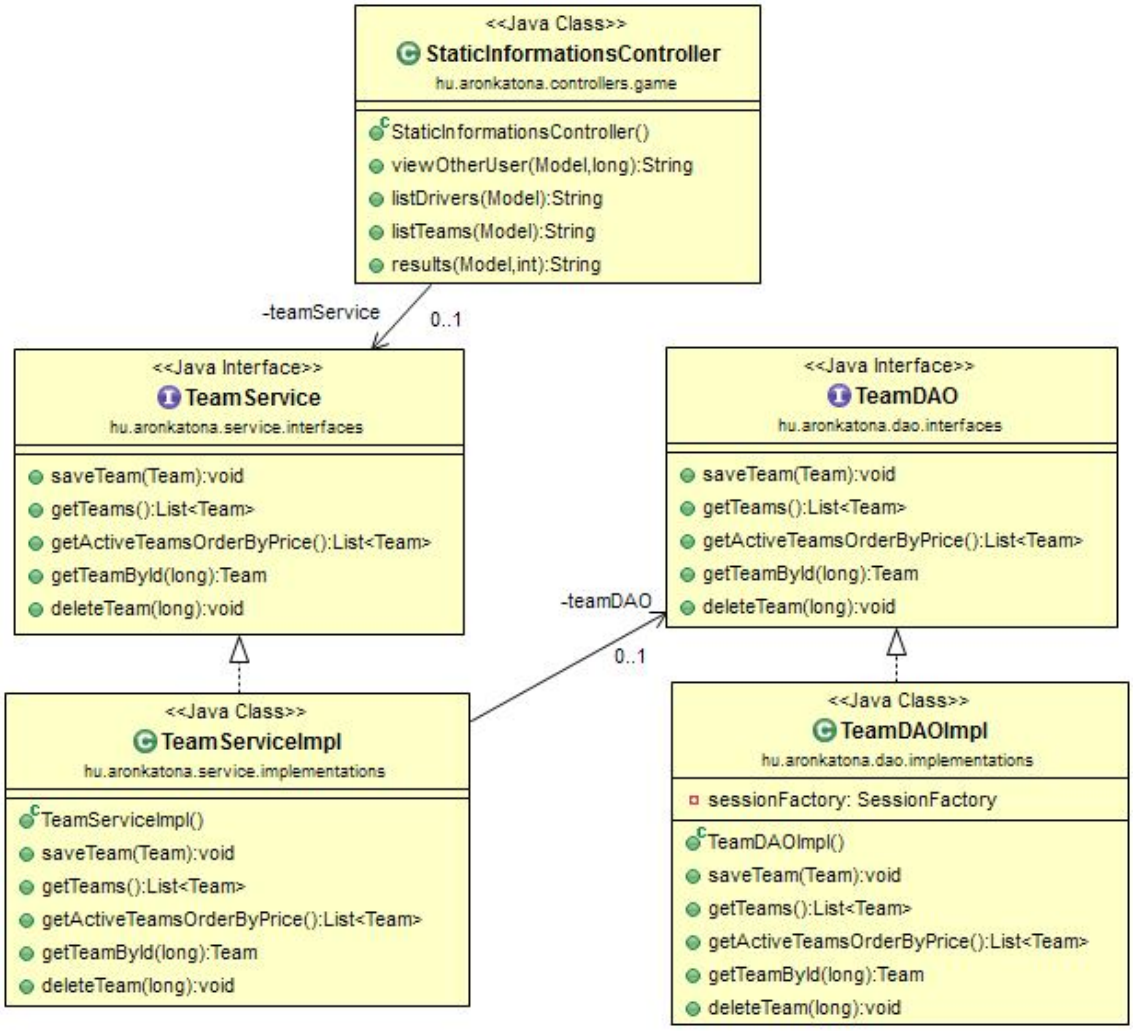
9. ábra

1) **allLeagues**: Lekéri a rendszerben található összes ligát és visszatér a „leagues” nézettel.
2) **myLeagues**: Lekéri a rendszerben található összes ligát, amiben a bejelentkezett felhasználó szerepel.
3) **createLeague**: Visszatér a „createLeague” nézettel, mely tartalmazza a liga felvételéhez szükséges űrlapot, hogy a felhasználó új ligát hozhasson létre.
4) **saveLeague**: A felhasználó által megadott adatokat validálja és ha minden adat megfelelő, akkor a rendszer létrehozza a ligát.

5) **searchByUserNameToInvite**: Ha a paraméterben megadott karaktersorozat megtalálható a rendszerben lévő felhasználók nevében, akkor ezen felhasználók listázásra kerülnek a „users” nézet segítségével.
6) **inviteUserToLeagueWithEmail**: Ez a függvény egy meghívó levelet küld a kiválasztott felhasználónak, hogy csatlakozzon a ligába.
7) **joinToLeague**: A bejelentkezett felhasználó ennek a függvénynek a segítségével tud bejelentkezni egy ligába.

StaticInformationsController

A StaticInformationsController osztályban a statikus oldalakért felelős metódusok szerepelnek.



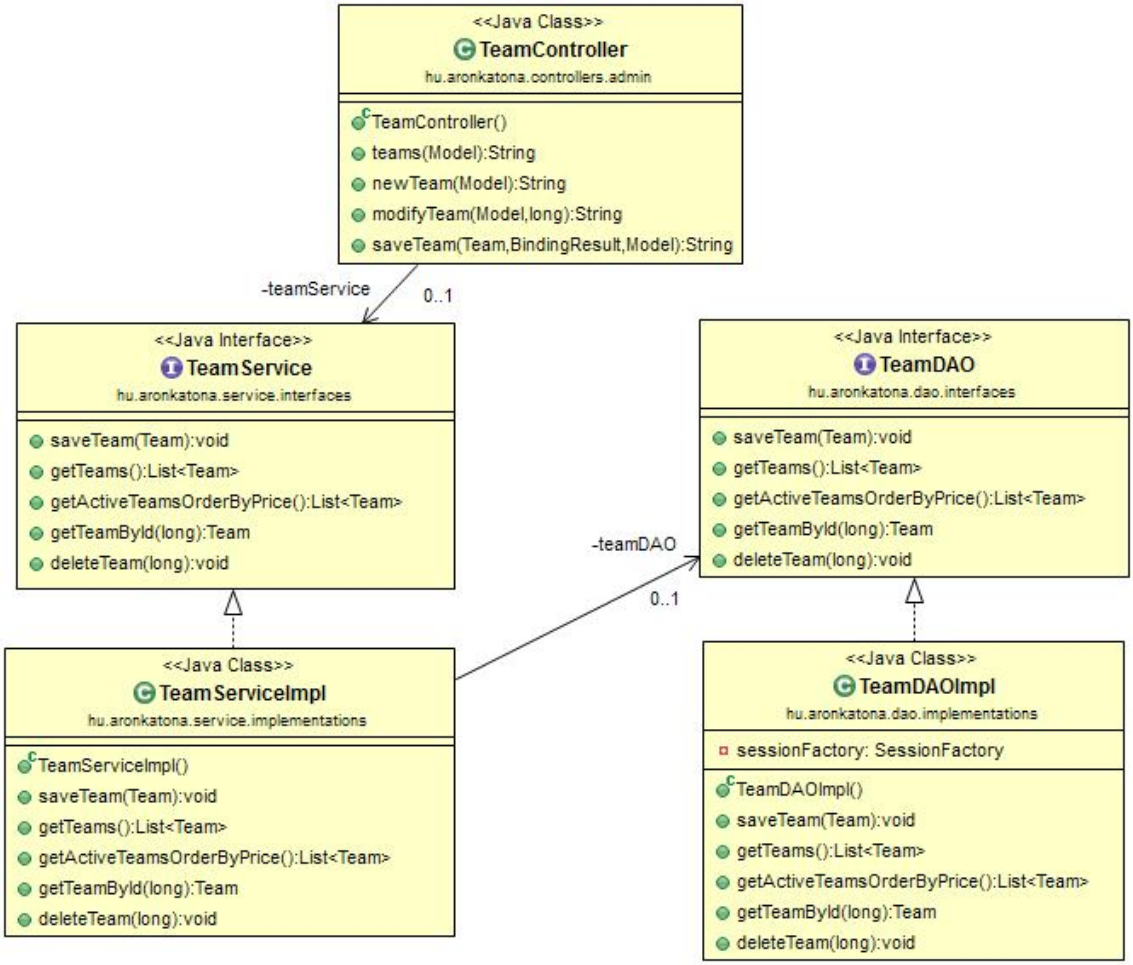
10. ábra

1) **viewOtherUser**: Ennek a segítségével tekinthetjük meg az egyes felhasználókhoz tartozó oldalakat az adatokkal együtt. A függvény paraméterben vár egy azonosítót, ami egy már létező felhasználóhoz tartozik és visszatér a „myTeam” nézettel.
2) **listDrivers**: A bajnokságban az aktív pilótákat listázza és visszatér a „drivers” nézettel.
3) **listTeams**: A bajnokságban az aktív csapatokat listázza és visszatér a „teams” nézettel.
4) **results**: A tabella megjelenítéséért felelős függvény. Az alkalmazáshoz sok felhasználó is tartozhat, ezért nem célszerű rögtön az összeset megmutatni az oldalon, ezért a függvény vár egy paramétert hogy az oldalon szereplő táblázat hányadik oldalát mutassuk meg. Ez a vezérlő a „users” nézettel tér vissza.
5) **searchByUserName**: Ennek segítségével kereshetünk rá felhasználókra. Ha a bejött karaktersorozat megtalálható valamelyik felhasználó nevében, akkor ezeket a felhasználókat megmutatjuk az oldalon a „users” nézet segítségével.
6) **rules**: Ez az akció egy teljesen statikus oldallal tér vissza, amin a játékhoz tartozó szabályok találhatóak.

Az adminisztrátori felülethez tartozó vezérlők és nézetek

TeamController

A csapatok módosításáért felelő metódusokat tartalmazza.

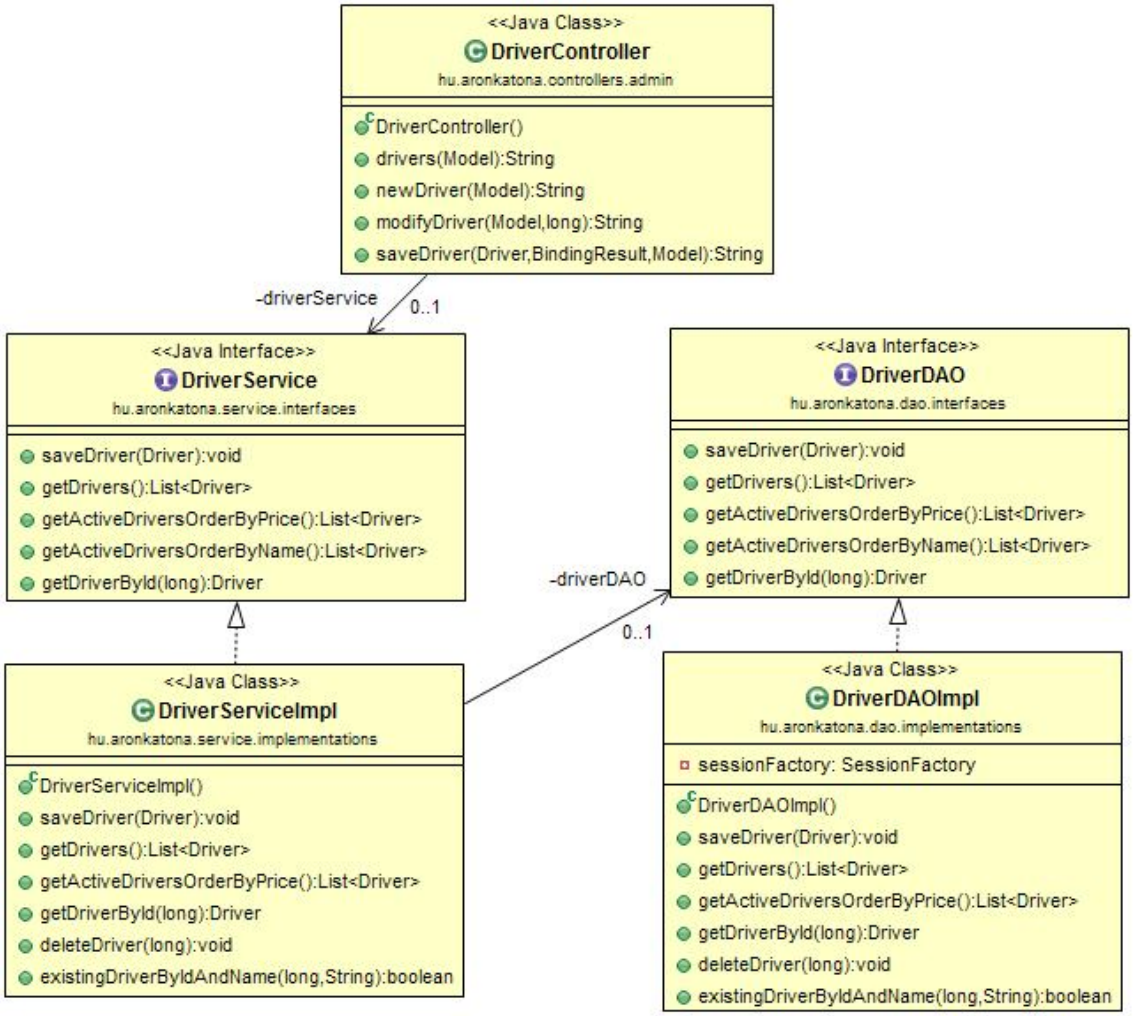


11. ábra

1) **teams**: Az összes csapatot kéri le az adatbázisból és visszatér a „teams” nézettel.
2) **newTeam**: Visszatér egy űrlappal, ahol az új csapatra vonatkozó adatokat adhatjuk meg.
3) **saveTeam**: Az űrlapon megadott adatokat validálja és ha minden megfelelő, akkor felveszi vagy módosítja az űrlapon található csapat adatait az adatbázisban.
4) **modifyTeam**: A paraméterben kapott azonosító alapján kikeresi az adatbázisból a csapatot és visszatér egy űrlappal, ahol a csapathoz tartozó adatokat tudjuk módosítani.

DriverController

Ebben az osztályban találhatóak azok az akciók, amik a pilóták adatait módosítják.

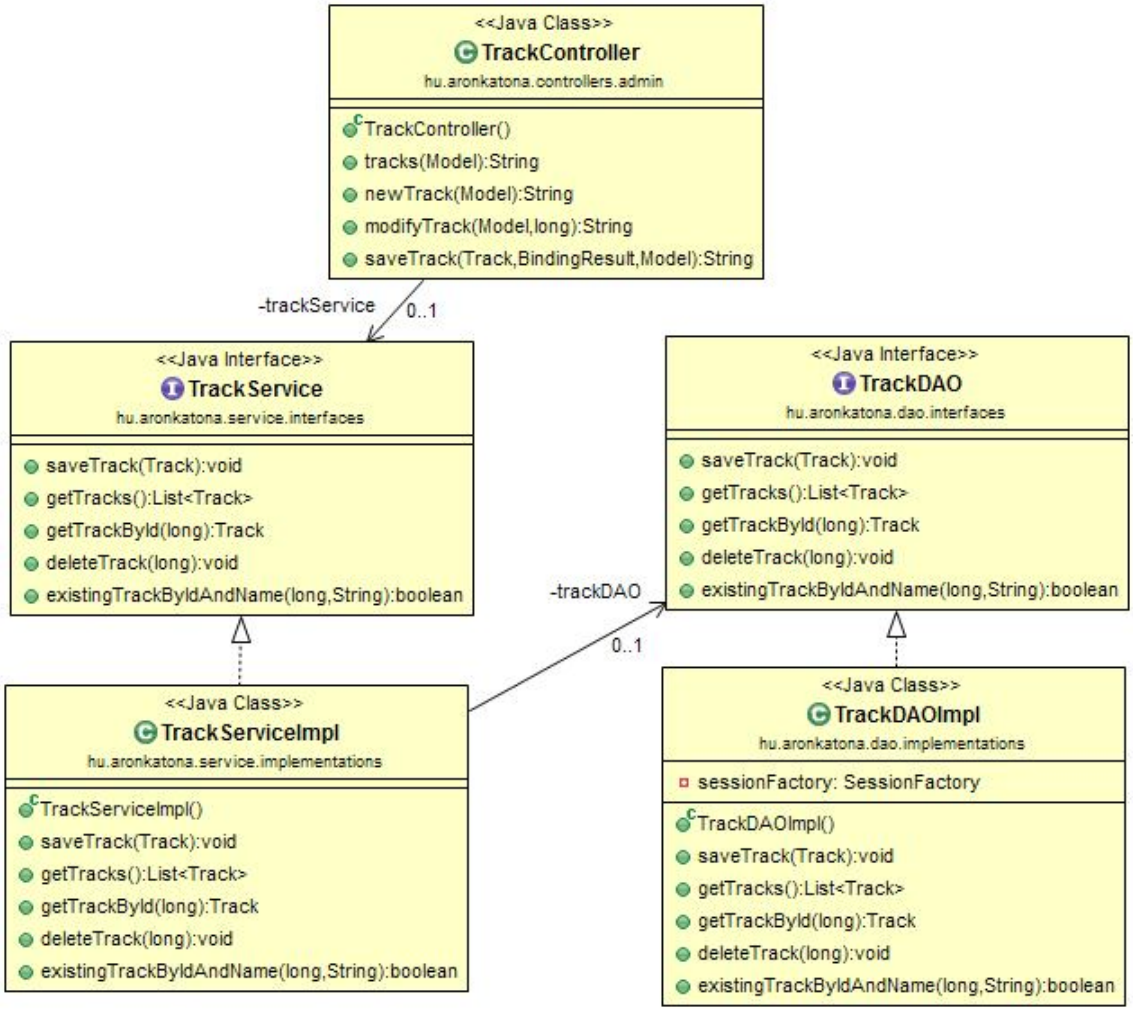


12. ábra

1) **drivers**: Az összes pilótát kéri le az adatbázisból és visszatér a „drivers” nézettel.
2) **newDriver**: Visszatér egy űrlappal, ahol az új pilótára vonatkozó adatokat adhatjuk meg.
3) **saveDriver**: Az űrlapon megadott adatokat validálja és ha minden megfelelő, akkor felveszi vagy módosítja az űrlapon található pilóta adatait az adatbázisban.
4) **modifyDriver**: A paraméterben kapott azonosító alapján kikeresi a rendszerből a pilótát és visszatér egy űrlappal, ahol a pilótához tartozó adatokat tudjuk módosítani.

TrackController

A pályák módosításáért felelő metódusokat tartalmazza.

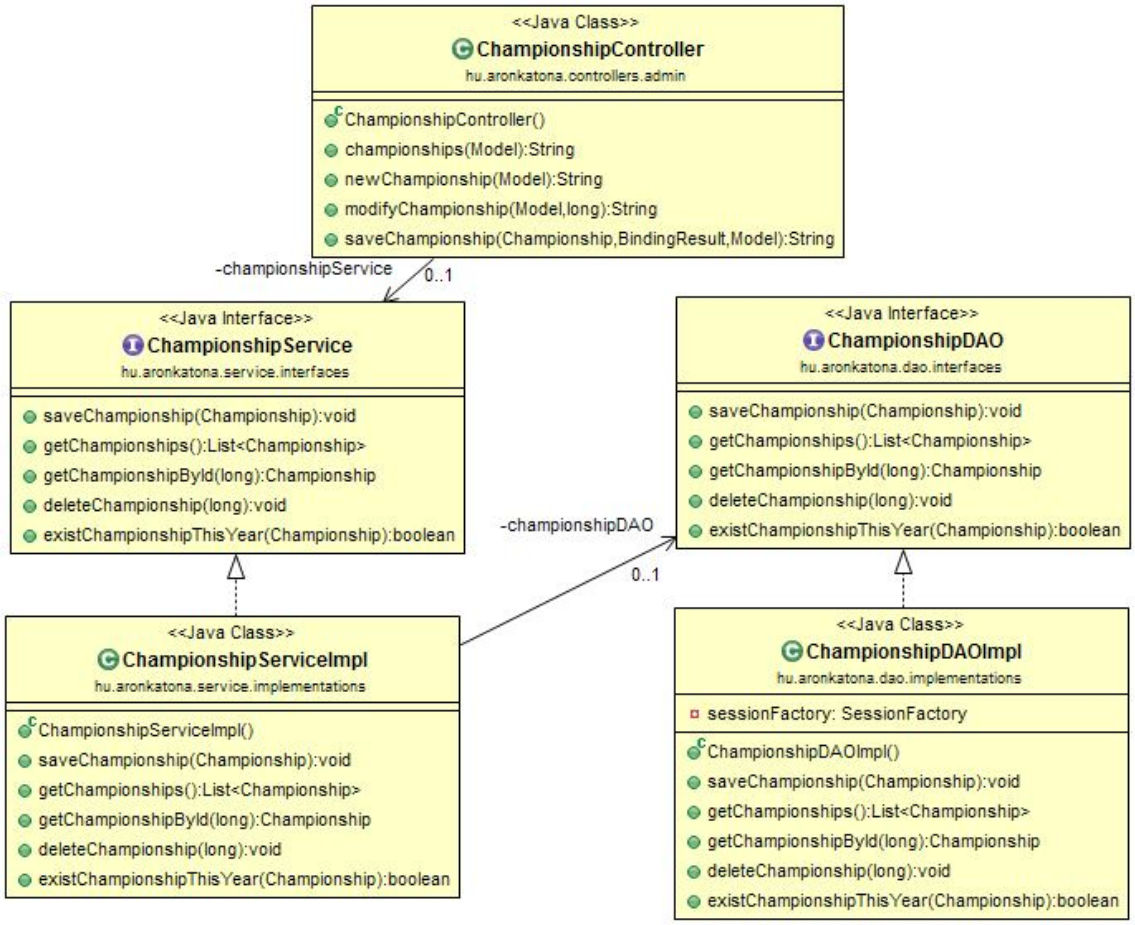


13. ábra

1) **tracks**: Az összes pályát kéri le az adatbázisból és visszatér a „tracks” nézettel.
2) **newTrack**: Visszatér egy űrlappal, ahol az új pályára vonatkozó adatokat adhatjuk meg.
3) **saveTrack**: Az űrlapon megadott adatokat validálja és ha minden megfelelő, akkor felveszi vagy módosítja az űrlapon található pálya adatait az adatbázisban.
4) **modifyTrack**: A paraméterben kapott azonosító alapján kikeresi az adatbázisból a pályát és visszatér egy űrlappal, ahol a pályához tartozó adatokat tudjuk módosítani.

ChampionshipController

A bajnokságok módosításáért felelő metódusokat tartalmazza.

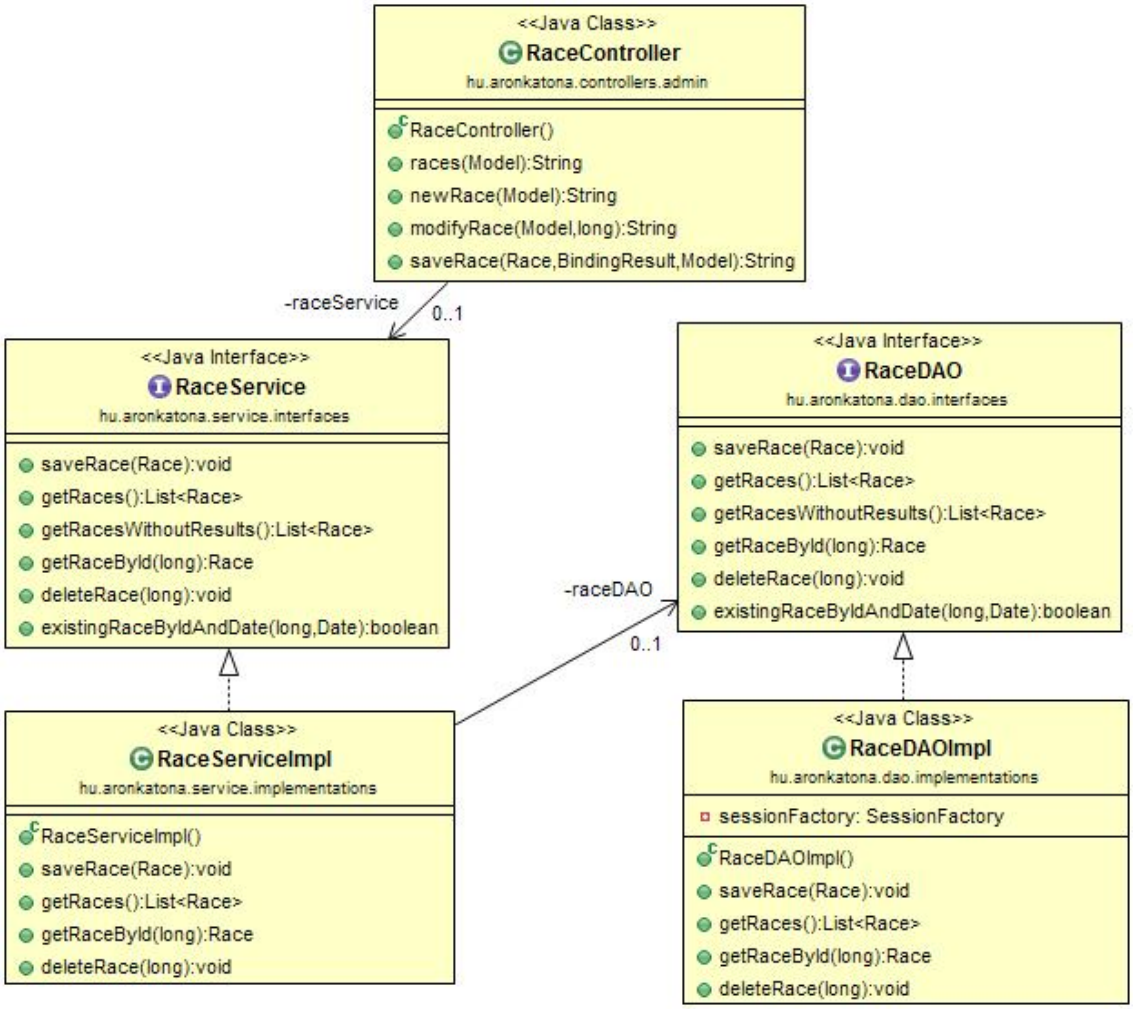


14. ábra

1) **championships**: Az összes bajnokságot kéri le az adatbázisból és visszatér a „championships” nézettel.
2) **newChampionship**: Visszatér egy űrlappal, ahol az új bajnokságra vonatkozó adatokat adhatjuk meg.
3) **saveChampionship**: Az űrlapon megadott adatokat validálja és ha minden megfelelő, akkor felveszi vagy módosítja az űrlapon található bajnokság adatait az adatbázisban.
4) **modifyChampionship**: A paraméterben kapott azonosító alapján kikeresi az adatbázisból a bajnokságot és visszatér egy űrlappal, ahol a pályához tartozó adatokat tudjuk módosítani.

RaceController

A futamok módosításáért felelő metódusokat tartalmazza.

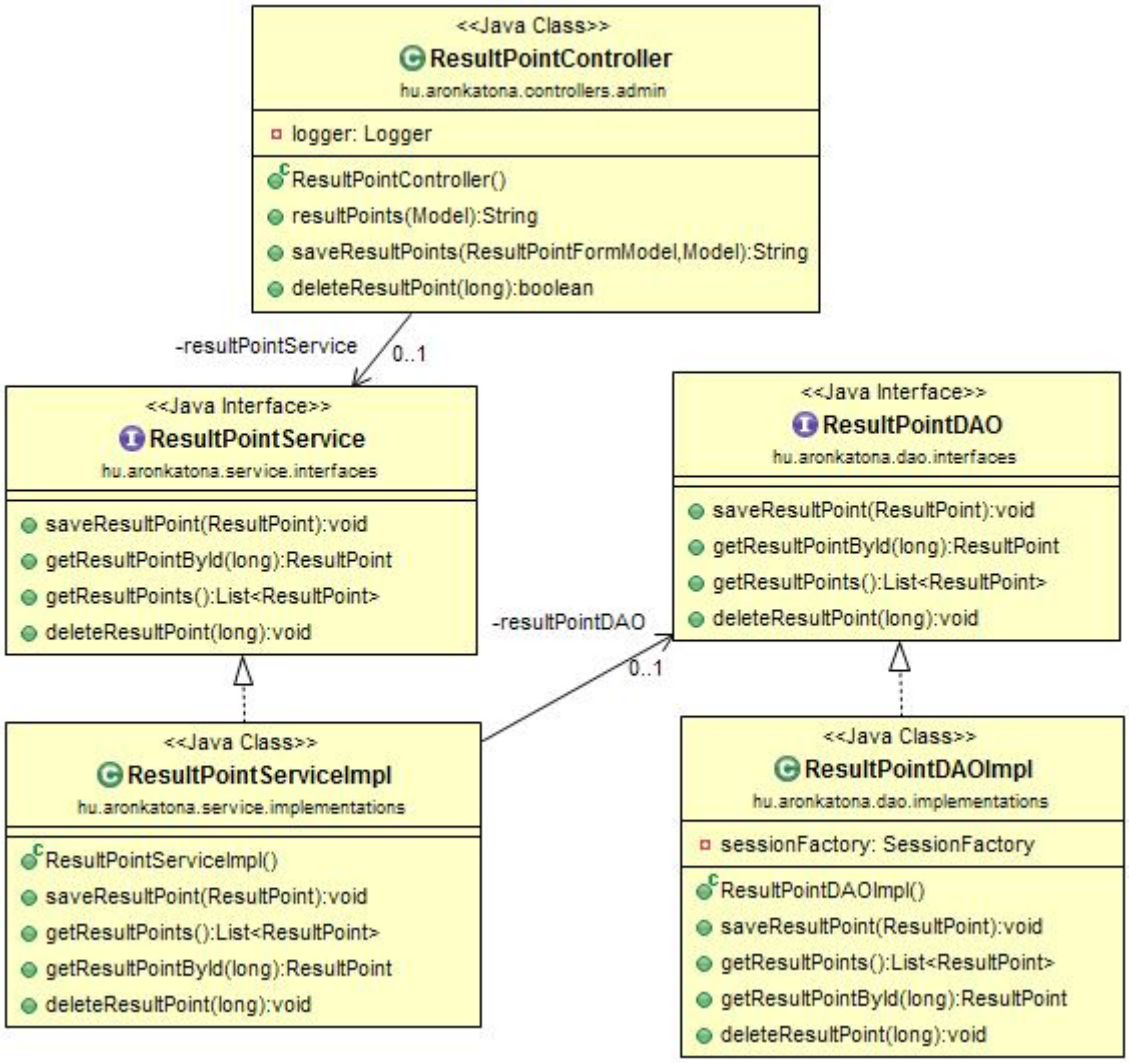


15. ábra

1) **races**: Az összes futamot kéri le az adatbázisból és visszatér a „races” nézettel.
2) **newRace**: Visszatér egy űrlappal, ahol az új futamra vonatkozó adatokat adhatjuk meg.
3) **saveRace**: Az űrlapon megadott adatokat validálja és ha minden megfelelő, akkor felveszi vagy módosítja az űrlapon található futam adatait az adatbázisban.
4) **modifyRace**: A paraméterben kapott azonosító alapján kikeresi az adatbázisból a futamot és visszatér egy űrlappal, ahol a pályához tartozó adatokat tudjuk módosítani.

ResultPointController

A vezérlőben található akciók segítségével tudjuk beállítani, hogy a futamok után milyen mértékű legyen a díjazás.

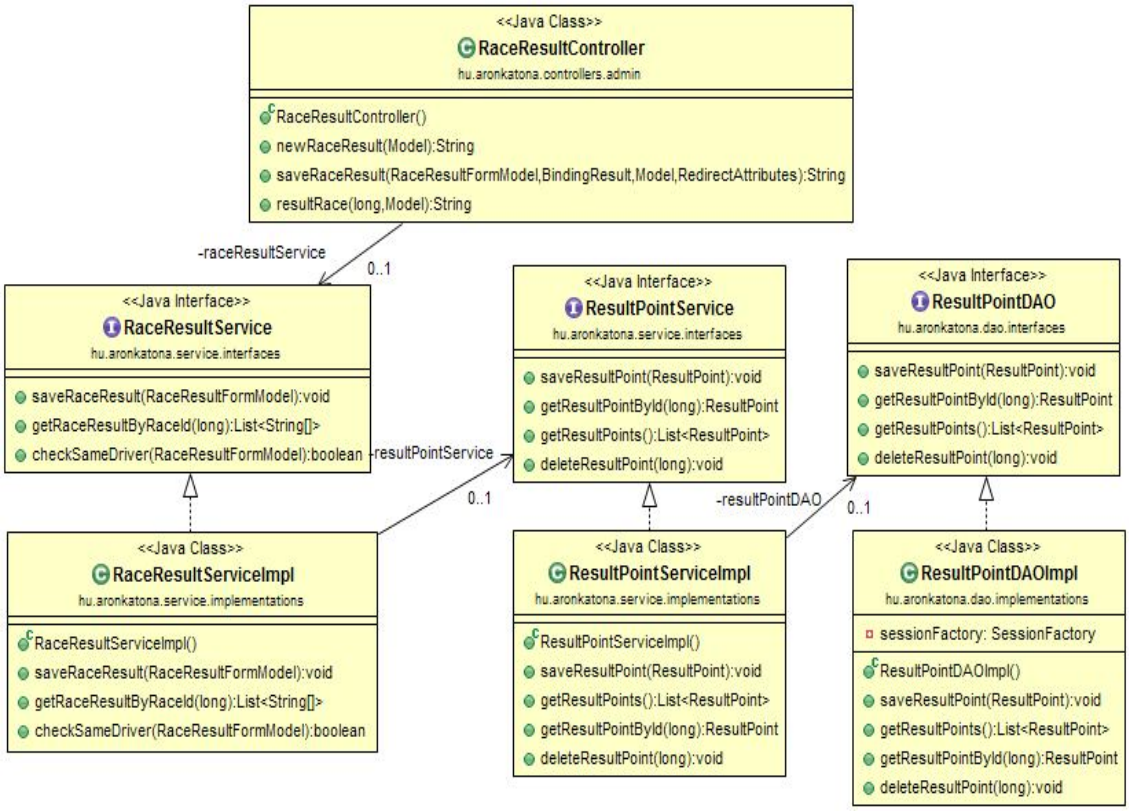


16. ábra

1) **resultPoints**: Visszatér egy táblázattal, ami valójában egy űrlap. Itt lehet módosítani a különböző helyezésekhez tartozó adatokat módosítani vagy újat felvenni.
2) **saveResultPoints**: Az előző táblázatban megadott adatokat validálja és ha minden megfelel, akkor menti a rendszerbe őket.
3) **deleteResultPoint**: Ez egy Ajax hívás, ami segítségével az előző táblázatból tudunk egy sort kitörölni.

RaceResultController

Ennek a vezérlőnek a segítségével lehet felvenni vagy megtekinteni egy futamhoz tartozó eredményt.

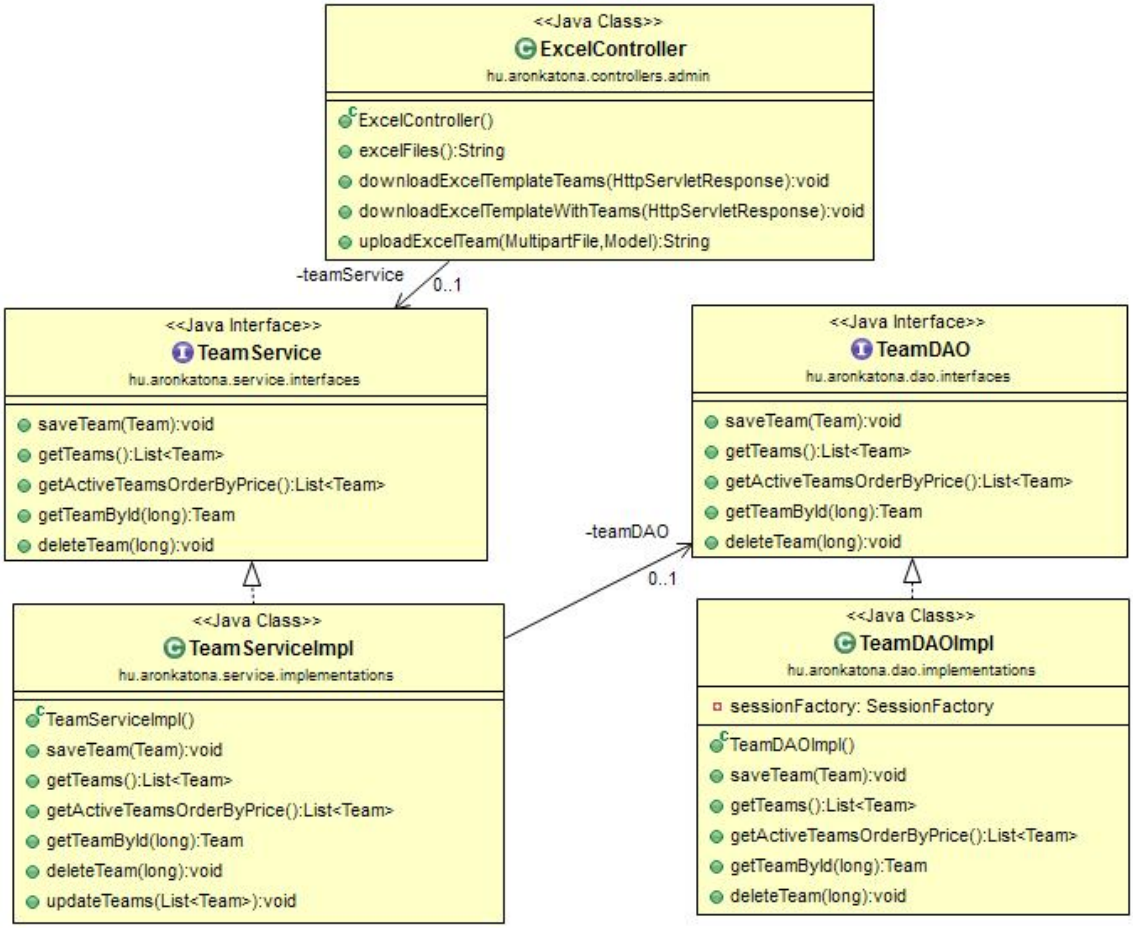


17. ábra

1) **newRaceResult**: Visszatér egy táblázattal, ami tartalmazza az összes bajnokságban szereplő pilótát. Minden helyezéshez csak egy pilóta rendelhető. Csapatokat nem látunk ebben a táblázatban, mert a rendszer tudja, hogy melyik pilóta melyik csapatban szerepel.
2) **saveRaceResult**: Az előző táblázatban beállított eredményt menti el az adatbázisba, ha nem szerepel egy pilóta kétszer ugyanabban az oszlopban. Frissíti a pilóták és csapatok értékeit, illetve kiosztja a felhasználóknak a díjazásokat és elmenti, hogy a futamon milyen felállással indultak.
3) **resultRace**: A paraméterben kapott azonosító alapján keresi ki a futamhoz tartozó végeredményt és visszatér a „raceResult” nézettel.

ExcelController

A controllerben található metódusok vizsgálják be vagy állítják elő a megfelelő excel fájlokat.



18. ábra

1) **excelFiles**: Visszatér az „excel” nézettel, ahol letölthetőek vagy feltölthetőek lesznek a fájlok.
2) **downloadExcelTemplateTeams**: Ez a metódus állítja elő azt az excel fájlt és teszi letölthetővé a felhasználó számára, ahol csak az oszlop nevek vannak kitöltve.
3) **downloadExcelTemplateWithTeams**: Ez az akció állítja elő azt az excel fájlt, amiben szerepel minden csapat az adatbázisból.
4) **uploadExcelTeam**: Ez a metódus vizsgálja be a feltöltött excel fájlt. Ha a fájlban minden megfelel a feltételeknek, akkor frissíti az adatbázisban a csapatok táblát. Ha a fájlban legalább egy adat nem felel meg, akkor erről tájékoztatja a felhasználót és az adatbázison nem hajt végre módosítást.

Tesztelés

Manuális tesztelés

Az oldalon található funkciókat teszteltem le helyes illetve hibás adatokkal. A következő funkciók helyességét vizsgáltam.

Felhasználói felületen

- Regisztrációs felület
- Bejelentkező felület
- Elfelejtett jelszó
- Csapatok és pilóták vásárlása/eladása
- Ligába való csatlakozás/kilépés/meghívás
- Felhasználókra való keresés

Itt a felhasználók szemszögéből teszteltem az elérhető funkciókat, ezért ezek felelnek meg a fekete doboz tesztelésnek.

Regisztrációs felület

A regisztrációs felületen teszteltem a hibás adatokat (üres felhasználónév, helytelen e-mail cím, túl rövid vagy túl hosszú jelszó, a két jelszó nem egyezik). Továbbá azt is ellenőriztem, hogy a regisztrációs felület ne engedjen tovább olyan felhasználónév vagy e-mail címet, ami már létezik a rendszerben. Ha az adatok sikeresek voltak, akkor az e-mail megérkezését is ellenőriztem.

Bejelentkező felület

A bejelentkezés során helytelen adatok megadása esetén a rendszer nem engedhet tovább, míg helyes adatok esetén a felhasználónak bejelentkezett állapotba kell kerülnie.

Elfelejtett jelszó

Itt teszteltem, hogy a megadott e-mail címre tényleg megérkezik-e a link, aminek segítségével jelszót lehet változtatni. Ha a jelszóváltoztatás sikeres volt, akkor ellenőriztem, hogy tényleg életbe lépett-e az új jelszó.

Csapatok és pilóták vásárlása/eladása

Megvizsgáltam azokat az eseteket, amikor a felhasználó olyan egységet akar venni, amivel már rendelkezik, ekkor a rendszer nem engedélyezte a műveletet. Továbbá ha a felhasználónak nincs elég pénze, akkor figyelmeztetést kapott a felhasználó.
Ha olyan pilótát akartam venni, ami abban a csapatban van, mint a már meglévő másik pilóta, a rendszer ebben az esetben is figyelmeztetett.

Ligába való csatlakozás/kilépés/meghívás

Itt megnéztem, hogy minden felhasználó tud-e csatlakozni az általa kiválasztott ligába, majd csatlakozás után van-e lehetőség a liga elhagyására. Továbbá teszteltem azt is, hogy a levelet megkapja-e a ligába meghívott felhasználó és a hivatkozás, amit tartalmaz a levél annak segítségével valóban csatlakozni lehet-e az adott ligába.

Felhasználókra való keresés

Ellenőriztem, hogy valóban csak azok a felhasználók kerülnek-e listázásra, akiknek a felhasználónevükben ténylegesen megtalálható a megadott karaktersorozat.

Admin felületen

- Csapatok szerkesztése
- Pilóták szerkesztése
- Pályák szerkesztése
- Futamok szerkesztése
- Bajnokságok szerkesztése
- Új eredmény felvétele

- Pontok beállítása
- Excel le- és feltöltés

Csapatok/pilóták/pályák/futamok/bajnokságok szerkesztése

A megfelelő egységek felvétele vagy módosítása után megnéztem, hogy a felhasználói felületen is ténylegesen megváltoztak-e az adatok.

Új eredmény felvétele

Ha a versenynek vége lett, akkor beállítottam a verseny végeredményt, majd ellenőriztem, hogy a kiosztott díjazások megfelelnek-e a valóságnak és a felhasználók jó adatokat látnak.

Pontok beállítása

Pontok átállítása után szintén teszteltem az új eredmény beállítása menüpontot és, hogy a rendszer valóban az új pontok alapján számol-e.

Excel le- és feltöltés

Ezen modul tesztelése esetén kipróbáltam, hogy a letöltött excel fájlok tartalma valójában megfelel-e az adatbázisban található adatoknak.
A feltöltés esetén megnéztem, hogy hibás adatok esetén a felhasználó valójában látja-e a hibákra vonatkozó hibaüzeneteket. Ha a feltöltés sikeres volt, akkor ellenőriztem, hogy az adatbázisban lévő rekordok tartalma helyesen módosult-e.

Egység tesztek

Az egységtesztek implementálásához a JUnit és Spring Test keretrendszereket használtam.
Ennek a tesztelési módszernek az volt az előnye, hogy a teszteket csak egyszer kellett megírnom a különböző esetekre. Ha kódban változtattam a különböző logikákon, akkor

az eseteket nem kellett a felhasználói felületen manuális lekattintatni, csak elindítani az egység teszteket, és ha valami hiba történt az implementáció megváltoztatása során, akkor azonnal láttam, hogy hol és milyen hiba fordult elő.

Továbbfejlesztési lehetőségek

Az alkalmazáshoz számos funkciót lehetne még implementálni.

Lehetne készíteni fórum felületet, ahol a felhasználók bármilyen témában tudnak beszélgetni. Továbbá lehetne chat lehetőséget is biztosítani a külön ligákhoz.

A futamokhoz tartozó eredmények beállításához lehetne készíteni olyan modult, ami automatikusan tölti le a Forma-1 hivatalos oldaláról az eredményeket futamok után és rögtön elvégzi a rendszer belüli módosításokat és mentéseket.

Továbbá az alkalmazáshoz lehetne készíteni natív android és iOS alkalmazásokat is.
Azonban ehhez a szerver oldalon szükség lenne kisebb módosításokra a vezérlő rétegen belül. Jelenleg az alkalmazás funkciói közül csak a töredék érhető el Ajax hívások segítségével.

A vezérlő réteg teljes átalakításával lehetne felépíteni egy REST alapú webservice-t, ami napjainkban nagyon elterjedt megoldás. Ennek nagy előnye, hogy a szerver teljes mértékben független a kliensoldaltól, ezért a kliensalkalmazások szinte bármilyen technológiával és szinte bármilyen eszközre elkészíthetőek lennének és mindet ugyanaz a szerver szolgálhatná ki.

Ha a vezérlő réteg átalakítása megtörténne, és a frontendet szeretnénk átírni más technológiára, akkor ehhez remek választás lenne az AngularJS. Ezt a keretrendszert a Google fejleszti, ami közel garancia a remekül tervezett és kivitelezett keretrendszerhez. Az Angular segítségével a frontendet is fel lehet építeni az MVC tervezési minta szerint, ami sokkal áttekinthetőbbé és gyorsabban fejleszthetővé teszi az alkalmazást.
Ha ez az átírás megtörténik, akkor a mobilos alkalmazások már csak egy lépés innen.

Napjainkban egyre nagyobb teret nyernek a hibrid alkalmazások mobilon, amik azt a célt szolgálják, hogy a mobilalkalmazások web technológiák segítségével fejleszthetőek és ezért ezek a megoldások platform függetlenek lehetnek. Azaz nem kell külön megírni a mobilos alkalmazást android,iphone és windows phone eszközökre.

Összegzés

A szakdolgozatom írása során a kiválasztott technológiáknak köszönhetően sok új és hasznos dolgot tanultam, azonban rengeteg olyan területe van még ezeknek a keretrendszereknek, amihez sok időre van szükségem, hogy elsajátítsam.

A program felépítése a keretrendszerek használata miatt rugalmas lett és könnyen továbbfejleszthető. Ezek az eszközök csapatmunkára is lehetőséget adnak, ezért a továbbfejlesztése az alkalmazásnak még gyorsabban folytatódhat.

A témabejelentőben tartalmazott célkitűzéseket sikerült teljesítenem, és egy olyan webalkalmazást készítenem, ami korszerű technológiák segítségével van megvalósítva és mindenki számára elérhető.

Irodalomjegyzék

Spring framework:

http://spring.io/

Elérés dátuma: 2015. május 09.

Hibernate:

http://hibernate.org/

Elérés dátuma: 2015. május 09.

Maven:

https://maven.apache.org/

Elérés dátuma: 2015. május 09.

Apache Velocity

http://velocity.apache.org/

Elérés dátuma: 2015. május 09.

JUnit

http://junit.org/

Elérés dátuma: 2015. május 09.

Tomcat

http://tomcat.apache.org/

Elérés dátuma: 2015. május 09.

MySQL

https://www.mysql.com/

Elérés dátuma: 2015. május 09.

Bootstrap:

http://getbootstrap.com/

Elérés dátuma: 2015. május 09.

jQuery

https://jquery.com/

Elérés dátuma: 2015. május 09.