

**SAPIENTIA ERDÉLYI MAGYAR TUDOMÁNYEGYETEM
MAROSVÁSÁRHELYI KAR,
INFORMATIKA SZAK**



**SAPIENTIA
ERDÉLYI MAGYAR
TUDOMÁNYEGYETEM**

Kosárlabda bajnokságok és statisztikai felmérések adatainak
létrehozása, importálása .Net Mauiba

DIPLOMADOLGOZAT

Témavezetők:

Drd. Csaholczi Szabolcs,
Egyetemi tanársegéd
Dr. Iclánzan David,
Egyetemi Docens

Végzős hallgató:

Major Zsolt

2023

**UNIVERSITATEA SAPIENTIA DIN CLUJ-NAPOCA
FACULTATEA DE ȘTIINȚE TEHNICE ȘI UMANISTE,
SPECIALIZAREA INFORMATICĂ**



**UNIVERSITATEA
SAPIENTIA**

Crearea, importarea campionatelor de basketball și a datelor statistice de sondaj în .Net Maui

LUCRARE DE DIPLOMĂ

Coordonatori științifici:
Drd. Csaholczi Szabolcs,
Asistent universitar
Dr. Iclănanz David,
Conferențiar universitar

Absolvent:
Major Zsolt

2023

**SAPIENTIA HUNGARIAN UNIVERSITY OF
TRANSYLVANIA**
FACULTY OF TECHNICAL AND HUMAN SCIENCES
COMPUTER SCIENCE SPECIALIZATION



SAPIENTIA
HUNGARIAN UNIVERSITY
OF TRANSYLVANIA

Create and import basketball league and statistical survey data
into .Net Maui

BACHELOR THESIS

Scientific advisors:
Drd. Csaholczi Szabolcs,
Assistant professor
Dr. Iclănanz David,
Associate professor

Student:
Major Zsolt

2023

LUCRARE DE DIPLOMĂ

Îndrumător: Csaholczi Szabolcs Coordonator științific: dr. Iclanțan David	Candidat: Major Zsolt Anul absolvirii: 2023
<p>a) Tema lucrării de licență: Crearea, importarea campionatelor de basketball și a datelor statistice de sondaj în .Net Maui</p> <p>b) Problemele principale tratate:</p> <ul style="list-style-type: none">- Studiu bibliografic privind limbajele C# și .Net Maui- Studierea bazei de date MySql pentru a stoca datele- Echipele din NBA care utilizează API-ul de interogare, studiindu-l în consecință- Documentarea adecvată a stadiilor de proiectare a aplicațiilor <p>c) Desene obligatorii:</p> <ul style="list-style-type: none">- Schema bloc a sistemului- Diagrame de proiectare pentru aplicația software realizată. <p>d) Softuri obligatorii:</p> <ul style="list-style-type: none">- Aplicația este bazată pe tehnologia .Net Maui și C#- Echipele din NBA care utilizează API de interogare.- Aplicația este o aplicație .NET MAUI bazată pe limbajul C#, al cărei scop principal este de a afișa și evalua statistic anumite echipe și jucători de baschet, de a le compara și evalua față de alte echipe și jucători, precum și o opțiune de a vă crea propria ligă de baschet cu echipe și jucători creați de dvs. sau cu una existentă.- Datele de interfață sunt stocate într-o bază de date MySql. <p>e) Bibliografia recomandată:</p> <p>[1] https://learn.microsoft.com/en-us/dotnet/maui/</p> <p>[2] https://learn.microsoft.com/en-us/dotnet/architecture/maui/mvvm</p> <p>[3] https://bleacherreport.com/articles/1813902-advanced-nba-stats-for-dummies-how-to-understand-the-new-hoops-math</p> <p>f) Termene obligatorii de consultații:</p> <p>g) Locul și durata practicii: Universitatea „Sapientia” din Cluj-Napoca, Facultatea de Științe Tehnice și Umaniste din Târgu Mureș, (sala 327 B) Primit tema la data de: 01.10.2022 Termen de predare: 30.06.2023</p>	
Semnătura Director Departament	Semnătura coordonatorului
Semnătura responsabilului programului de studiu	Semnătura candidatului

Declarație

Subsemnatul/a MAJOR ZSOLT, absolvent(ă) al/a specializării
INFORMATICA, promoția 2023 cunoscând prevederile Legii Educației Naționale 1/2011 și a Codului de etică și deontologie profesională a Universității Sapientia cu privire la furt intelectual declar pe propria răspundere că prezenta lucrare de licență/proiect de diplomă/disertație se bazează pe activitatea personală, cercetarea/proiectarea este efectuată de mine, informațiile și datele preluate din literatura de specialitate sunt citate în mod corespunzător.

Localitatea, CORUNCA
Data: 15.06.2023.

Absolvent

Semnătura... 

Kivonat

A sportok és kompetitív küzdelmek terében nagyon gyakori az az eset, mikor azon személyek, akik a háttérből követnek egy mérkőzést bele élik magukat a versengés világában, olyannyira, hogy erőteljesen szurkolnak kedvelt feleikért vagy akár fogadnának is bizonyos versenyzőkbe. Az ember kompetitív kíváncsisága sok helyen felterülhet, kezdve egy élő mérkőzésen számos más rajongóval való őrjöngéstől, egészen egy minden nap beszélgetésig egy bárban haverjainkkal arról, hogy ki fog győzni a mai napi meccsen. Én ebből az ötletből kiindulva egy olyan számítógépes alkalmazást készítettem, ami két fontos lehetőséget kínál a felhasználó számára a kosárlabda bajnokságok terén felmerült dilémák megoldására: egy csapat és játékos teljesítményeit importáló és rögzítő rendszer, amely segítségével nem csak egy sportoló személyes statisztikáiról szerezhetünk komplex és fejlett kimutatásokat, hanem egy másik féllel szemben való felmérést is rendez, valamint egy bajnokság készítő program, ahol haverok által vezetett csapatokat hozhatunk létre és velük mérkőzéseket és azokról rangsorolásokat tarthatunk karban.

Ennek a rendszernek a kivitelezéséhez egy C# nyelven alapuló .Net MAUI programot fejlesztettem ki, amelyiket MySQL nyelven alapuló phpMyAdmin adatbázis-rendszer alkalmaz csapatok, játékosok és statisztikák lementésére és alkalmazására, valamint egy API segítségével lehet lekérni valódi NBA játékosok és csapatok szezonjainak statisztikáit, amelyeket programunk metrikai felmérések kialakításához fog alkalmazni.

Alkalmazásom három jellegzetes főmenüpontra osztható fel. Az első a ligakészítő, ahol a felhasználó sajátlag létrehozott klubokat alkalmazhat különböző mérkőzésekre, azon belül a csapatok számára létrehozott játékosok számára statisztikákat és végső eredményeket rendelhetnek hozzá, valamint a csapatokat és játékosokat különböző kritériumok alapján rangsorolhatja. A második a csapat és játékos készítő, ahol létrehozhatnak feleket, testre szabhatják őket, valamint valódi csapatokat, játékosokat és ezek szezonbeli teljesítményeket is lekérhetik meg lementhetik az adatbázisba. A harmadik, egyben kulcsfontosságú, maga a feleket összehasonlító és felmérő rendszer, ahol az importált csapatokat és játékosokat a lekért teljesítmények segítségével különböző komplex metrikák és grafikonok által tudják ezeket összemérettetni.

Kulcsszavak: kosárlabda, sajátos bajnokság, mérkőzések karbantartása, összemérettetések, statisztikák, metrikák, kompetitív kíváncsiság, adatbányászat

Rezumat

În domeniul sporturilor și al competițiilor devine tot mai obișnuit, ca cei care vizionează un concurs să devină atât de investiți, încât să se piardă în lumea competițiilor în aşa fel, încât să-și susține cu toată voința echipa preferată sau chiar și să pună pariuri pe niște competitori. Curiositatea competitivă a omului poate ieși la iveală în multe ocazii, începând de la prezența și susținerea unei competiții live, până la o simplă discuție cu prietenii într-un bar despre cine va câștiga meciul zilei. Pornind din această ipoteză, am dezvoltat o aplicație de calculator, care va oferi două funcții cheie utilizatorilor pentru rezolvarea acestor dileme în domeniul campionatelor de baschet: un sistem de importare și înregistrare a performanțelor echipelor și jucătorilor, prin care nu doar că putem obține statistici complexe și avansate despre un sportiv, ci putem și organiza o evaluare față de un alt competitor, precum și un program pentru crearea unor campionate, unde putem forma echipe conduse de prieteni și putem menține meciuri și clasamente cu acestea.

Pentru implementarea acestui sistem, am dezvoltat o aplicație .Net MAUI bazată pe limbajul C#, care utilizează sistemul de bază de date phpMyAdmin bazat pe MySQL pentru a salva și aplica informații despre echipe, jucători și statistici, iar prin intermediul unei API-uri, putem obține statistici ale jucătorilor și echipelor reale din NBA, pe care le vom utiliza pentru a dezvolta măsurări metrice complexe.

Aplicația mea poate fi împărțită în trei funcții principale caracteristice. Primul este creare a ligilor, unde utilizatorul poate adăuga cluburi create de el în diferite meciuri, poate atribui statistici jucătorilor creați pentru echipe și rezultate finale, iar poate clasifica echipele și jucătorii în funcție de diferite criterii. Cel de-al doilea este susținerea echipelor și jucătorilor, unde pot fi create echipe, pot fi personalizate, iar pot fi obținute și salvate echipele și jucătorii reali, precum și performanțele lor sezoniere într-o bază de date. Cel de-al treilea, și în același timp cel mai crucial, este sistemul de comparare și evaluare a echipelor, unde echipele și jucătorii importați pot fi evaluati și comparati utilizând diferite metrice și grafice complexe.

Cuvinte cheie: baschet, campionat personalizat, susținerea meciurilor, comparații sportive, statistici, metrici, curiozitate competitivă, minerit de date

Abstract

In the realm of competitive sports, it is not uncommon for individuals who follow a contest from the sidelines to immerse themselves in the world of competition to such a degree, that they passionately cheer for a favorite side or even place bets on certain competitors. The competitive curiosity of humans can manifest in various ways, ranging from the frenzy of cheering alongside numerous other fans at a live match to everyday conversations with friends at a bar on who will win today's big game. Building upon this idea, I have developed a computer application that offers two important possibilities for users to solve dilemmas that arise in the realm of basketball championships: a system for importing and recording team and player performances, which enables us to create complex and advanced insights not only into an athlete's personal statistics, but also to compare them with those of another party, and a championship creation program where we can form teams led by others, engage in matches with them, and maintain rankings based on those matches.

To implement this system, I have developed a .Net MAUI program based on the C# language. It utilizes a MySQL-based phpMyAdmin database system to store and retrieve data for teams, players, and their statistics. Additionally, through an API, there is a possibility to fetch statistics of real NBA players and teams for different seasons, which will be used to create multi-layered metric measurements.

My application can be divided into three distinctive menu sections. The first one is the league creator, where users can employ personally created clubs in various matches. Within this section, they can assign statistics and final results to players associated with the teams, and also rank the teams and players based on different criterias. The second section is the team and player creator, where users can create teams, customize them, and also retrieve and store the seasonal performances of real teams and players in the database. The third section, which is also a crucial one, is the comparation and evaluation section, where imported teams and players can be evaluated and compared using different convoluted metrics and graphs based on the imported information based on their performance.

Key words: basketball, personalized championship, match maintenance, comparisons, statistics, metrics, competitive curiosity, datamining

Tartalomjegyzék

Ábrák jegyzéke	11
Táblázatok jegyzéke	12
1. Bevezető	13
2. Alapötletek és Célok	14
2.1. Alapvető Tervek	14
2.2. Betartandó Szabályok	15
2.3. Más Szoftverből Nyert Inspirációk	15
3. Nyelv, eszközök és felépítés	18
3.1. .NET MAUI	18
3.2. C#	19
3.3. XAML	19
3.4. Visual Studio 2022	20
3.5. phpMyAdmin és MySQL	20
3.6. XAMPP	21
3.7. API	21
4. Kód felépítése, szerkezete	23
4.1. MVVM és a Felhasználói Felület Bevezetése	23
4.2. Szükséges Könyvtárak	24
4.3. Osztályok (Classes)	24
4.4. MVVM összekötése a felülettel	25
4.5. Egyéb jellegzetes elemek	26
5. Alkalmazás Bemutatása	27
6. Funkciók bemutatása	28
6.1. Első menü: Ligakészítés	29
6.2. Második menü: Felek Létrehozása és Importálása	30
6.2.1. Alap Felület	30
6.2.2. Csapatok Felülete	31
6.2.3. Játékosok Importálása	32
6.2.4. Statisztikák Importálása	33
6.3. Harmadik menü: Felmérések és Metrikák	34

6.3.1. Felület és Használat Leírása	35
6.3.2. Alapvető Statisztikák Bemutatása	35
6.3.3. Komplex Statisztikák Bemutatása	36
7. Felmérések Tesztelése és Kimutatása	41
8. További Fejleszthetőségek	44
9. Konklúzió	46
Irodalomjegyzék	48

Ábrák jegyzéke

2.1.	Basketball Reference Kyrie Irving statisztikáit megjelenítő oldala	16
2.2.	Challenge Place weboldalán létrehozott sajátos bajnokság felülete	17
6.1.	Az alkalmazás Use Case diagramja	28
6.2.	Az alkalmazás fő oldala, ami megjelenik indításkor	29
6.3.	A ligakészítőn belül egy liga főoldala, ahonnan meccseket és rangsorolásokat érhetünk el	30
6.4.	Létrehozott csapatot bemutató felület	32
6.5.	Létrehozott csapatot bemutató felületen a játékosok listája	32
6.6.	Importált csapat felülete	34
6.7.	Importált játékos felülete	35
6.8.	Felmérések és Metrikák felülete	36
6.9.	Alapvető statisztikákról kimutatott képek játékosoknál és csapatoknál . .	37
7.1.	LeBron James és Luka Doncic hasznossági aránya van a grafikonon ábrázolva	43
7.2.	LeBron James és Luka Doncic valódi lövési aránya van a grafikonon ábrázolva	43

Táblázatok jegyzéke

6.1. Harden és Iguodala 2012-2013-as szezonbeli statisztikái Basketball Reference-n	38
6.2. Hollinger 2022-23 NBA rangsorolása PER szerint	40
7.1. LeBron James és Luka Doncic programunk által kimutatott statisztikái a 2019-es szezonban	42

1. fejezet

Bevezető

A modern ember egy megszokottan kompetitív és küzdő lelkű lény, akinek sokszor érzelmei és azok általi rajongó cselekedetei befolyásolják mások sikerét. Ez a tény a legjobban a sport és a küzdelem világában érezhető át. Sok világszerte rajongó buzdítja saját élsportolót és csapatait a történelmi szintű győzelmekhez, és sokszor az ellenfeleket és annak rajongóit a földig szidnák. Végtelen számú sportágakat ūznek napjainkban, egyeseket internacionális szinten számtalan rajongó által követve, másokat pedig csak digitálisan játszodják egy videójátékban vagy 4 barát valakinél otthon. A sport történelme még időszámításunk előttiig is visszamegy, valamint napjainkban egyre több a szórakozási kereslet sportesemények követésére, szurkolására és akár egyes sportolókra és csapatokra való fogadásokra.

Az emberek vérében van a kíváncsiság, így sokan szeretnek bizonyos statisztikákat és összehasonlításokat végezni ezen sportolók és csapatok iránt. Ilyenek általában az olyan csevegések a haverokkal, amikor bizonyos sportokat beszéltek meg, pontosabban, hogy az aznap esti játszmát melyik csapat fogja megnyerni. Ezen beszélgetésekkel szemben egy sokkal pontosabban és mértékletesebben leíró statisztikákkal is össze lehet hasonlítani csapatainkat. Különböző statisztikákkal és kimutatásokkal lehetséges egyes csapatok és játékosok teljesítményeit valamilyen szinten megjósolni, akár osztályozni és elemezni magára vagy más játékos ugyanazon osztálybeli statisztikáinak függvényében. Ezeket a módszereket általában sportközvetítő csatornák, dezignált weboldalak és alkalmazások, valamint sportfogadók alkalmazzák ahhoz, hogy meg tudják határozni mely játékosok és csapatok vannak az élen jelen pillanatban, valamint, hogy kikre érdemes bízni és fogadni ahhoz, hogy nyerjünk is ezen fogadásunkkal.

2. fejezet

Alapötletek és Célok

2.1. Alapvető Tervek

Amint említettem az előbbiekben, egy olyan alkalmazás létrehozására törekszünk, mely egyénileg létrehozott vagy általunk keresett csapatokat vagy játékosokat statisztikailag felmérhetünk és kiértékelhetünk bizonyos metrikák és kimutatások segítségével. Ehhez a programunk megfelelő módon 2 jelentékes területen kell végezzen pompás munkát ezen célok eléréséhez.

Az egyik a létrehozott felekkel való dolgozás. Ezeket a klubokat vagy sportolókat az alkalmazásunk képes kell legyen, hogy megadott specifikációk és kiszabott adatok felhasználásával létre tudja hozni, ezeket megadott bajnokságokhoz tudjuk hozzárendelni, más létrehozott csapatokkal együtt való új mérkőzések megszervezése, valamint ezen játszmák testreszabásának, lévén a játékosok és a két csapat teljesítményeinek lejegyzésére. Az így megszerzett információk segítségével fog kelleni tudnia a programnak bajnokságon belüli statisztikákat rendszereznie, tabellákba rendeznie különböző megjelenítési és rendezési szabályok alapján, amiket maga a felhasználó fog tudni állítgatni különböző kimutatások elérése érdekében.

A másik terület, ahol alkalmazásunk nélkülözhetetlenül sikeresen kell működjön az az általunk kiválasztott játékosok és csapatok statisztikai elemzése. Ahhoz, hogy elérhesük a mi kiszabott felünk adatait, ahhoz egy megadott ablak vagy oldal segítségével a program el fogja kérni felhasználónktól a kívánt sportolót vagy csapatot másuktól elkülönítő adatait, ez legyen akár egy csapatnak a neve vagy egy játékos nevében szereplő szó, és ennek alapján a programunk egy megadott listát vagy objektumot fog visszatéríteni a program user-jének. Ez annak érdekében lényeges, hogy a felhasználónknak tudjunk adni egy valamilyen szintű megbízhatósági szintet az alkalmazásunk felé, hisz ennek az elemnek legfőbb szerepe az lenne, hogy a user meg tudjon bizonyosodni a játékos vagy csapat lekérendő adatainak helyességéről. Így nem lesz meglepve a felhasználó, mikor az adatok helytelenek a valódi interneten utánánézett adataikkal szemben. Ez abban is jó, hogy elkerüljük a helytelen statisztikák és kimutatások létrehozását. Miután leellenőriztük az adatbázisba bevivendő adatainkat, a program képes kell legyen azokat az adatokat becsületesen és helyesen lementenie, egy olyan formátumban, amely kompatibilis lesz bármilyen más adat- vagy kódrészettel az alkalmazás különböző funkcióinak során. Az adatok könnyű és hatékony elérése érdekében importáljuk egy megkapott API-ból és azt feltöljük az adatbázisunkba, hogy aztán onnan relatív gyorsan lekérhesse az alkalmazás.

A lekért importált adatokat ezek után meg szeretnénk jeleníteni és azok segítségével összehasonlítási algoritmusokat végezni, az azokból megszerzett eredményeket pedig ki-mutatni különböző informatív, hasznos és átlátható statisztikák segítségével. Ehhez egy harmadik menüpontot fog kelleni létrehozunk, ahol a játékosok és csapatok összehason-lítása, összemérettetése történik, és ahol ezen hasonlításokból nyert információt megjele-níthessük. Az oldalon 2 lenyitható lista fog szerepelni, ahonnan a felhasználó kiválasztott kritériumok alapján fog tudni választani különböző szűrési opciókat, amik szerint ki tud választani 2 ugyanolyan típusú felet. A program, ami után a user megszabta melyik két felet szeretné összehasonlítani, lekéri ezeknek összes vitális adatait és statisztikáit, és azok függvényében hoz létre különböző összehasonlítási metrikákat, kimutatásokat, amik alapján a felhasználó látni fogja melyik fél erősebb a másikhoz képest. Itt az első op-ció az lesz, hogy a felhasználó csapatokat vagy játékosokat szeretne összehasonlítani. Az importált játékosokat vagy csapatokat megszabott szezon szerint lehet majd összemérni, minden két félnek választ a user egy megadott évet, ami importálva van az adatbázisban, így a megadott évben játszott mérkőzésekéről szerzett teljesítményi statisztikákkal fog az alkalmazásunk dolgozni. Játékosok esetében lesznek csapat szerinti összehasonlítások is, amik alapján a program nem csak a 2 sportoló személyes teljesítményét teszi össze, hanem azok egész csapatainak teljesítménye is fel lesz mérve.

2.2. Betartandó Szabályok

A MyLeague alapvető nélkülözhetetlen funkcióin kívül lényeges számontartani az alkalmazás készítésének univerzális szabályait és követelményeit. Ezek közé tartoznak legfőképpen a felhasználó igényeinek betartása, avagy a felhasználó által kívánt funkció-kat a programunk legyen képes teljesíteni, valamint ezen opciókból kinyert eredmények feltétlenül teljesítsék a user által megszabott elvárásokat. Továbbá biztosítani kell az alkalmazásunk teljesítményének sikerét, a funkcionalitást és a megbízható, gyors válasz-időket kínálja számunkra program, hisz, ha programunkban gyakoriak a lassú, sok időt elvező funkciók vagy maga a program gyakran fagy le vagy hibákkal rendelkezik, akkor az általában szignifikánsan csökkenti a felhasználói élményt. Alkalmazásunkban fontos szerepet fog játszani a skálázhatóság, hisz ez egy nagy adatokat importáló és feldolgo-zó szoftver, amely növekvő adatmennyiség esetében is megfelelően kell működjön. Végül két további alapvető szabálynak is meg kell feleljen a programunk, az lévén a jogosítási szabályok betartása, avagy, hogy az alkalmazásunk ne tartalmazzon adatokat vagy jogokat megsértő elemeket, funkciókat, valamint alkalmazkodni kell különböző platformokon való optimalizáláshoz, hisz mivel ez egy .NET MAUI alkalmazás, képesek leszünk több felületre is kifejleszteni alkalmazásunkat, legyen az számítógépen vagy telefonon.

2.3. Más Szoftverből Nyert Inspirációk

A legfontosabb szoftverek és oldalak, ahonnan nyertem ki az alkalmazásomat megszabó ötleteimet azok a Basketball Reference weboldal és a Challenge Place nevezetű telefonos és webes alkalmazás. Ennek a két szoftvernek a megszabó funkcióit szeretném implementálni applikációmiba, valamint azok körül kifejleszteni alapvető terveimet.

A Basketball Reference egy bárki számára elérhető weboldal, ahol egy megadott játékoszt vagy csapatot megkereshetünk és különböző fejlett statisztikákat jeleníthetünk meg mindenféle időket és körülményeket figyelembe véve. Ennek a sportágnak legnagyobb rajongói is ezt az oldalt használják bizonyos statisztikák ellenőrzésére, valamint egyes mérkőzéseken részt vevő felek kiértékelésére. Alkalmazásom erről a weboldalról leginkább az adatok statisztikai felmérésének és megjelenítésének jellegét venné át, avagy nálunk is az importált játékosokról megadott meccsekben nyert teljesítményeket szeretném felmérni statisztikailag, jellegzetes metrikákat és grafikus megjelenítések segítségével. Az adatok egyszerű megjelenítése és statisztikai felmérése mellett az én szoftverem nem csak egyszeri játékossal vagy csapattal fog dolgozni, hanem egyszerre két fél fog összemérettetni statisztikai szempontból. Így alkalmazásunk nem csak a személyes teljesítményt fogja felmérni, hanem a másik általunk kiválasztott fél teljesítményével szemben is különböző metrikák lesznek megjelenítve, így összehasonlítás által kinyert adatokat is meg fogunk szerezni, ami által meghatározhatjuk melyik fél a jobb.

The screenshot shows the Basketball Reference homepage with a search bar at the top. Below it is a navigation menu with links to Players, Teams, Seasons, Leaders, Scores, WNBA, Draft, Stathead, Newsletter, and Full Site Menu. The main content area features a large image of Kyrie Irving and his name. Below the image are his stats: 60 games, 27.1 PTS, 5.1 TRB, 5.5 AST, 49.4 FG%, 37.9 FG3%, 90.5 FT%, 57.2 eFG%, 22.4 PER, 7.4 WS. To the right are several achievement boxes: 8x All Star, 2016 NBA Champ, 3x All-NBA, 2011-12 All-Rookie, 2011-12 ROY, 2013-14 AS MVP. Below these are four small jerseys with numbers 2, 11, 11, and 2. A sidebar on the left from 'HOOPS RUMORS' contains a link to a news item about Max Strus.

2.1. ábra. Basketball Reference Kyrie Irving statisztikáit megjelenítő oldala

A Challenge Place az egy alap bajnokságkészítő alkalmazás, ahol létrehozott csapatokat szabhatunk testre és sok különböző mérkőzésekbe sorolhatjuk őket. Ezen alkalmazásnak alapvető szerepe a felhasználó által megszabott és megalkotott ligákban az általa megalapított csapatokat mérkőzésekbe sorolása, így egy sajátlagos megalkotott bajnokságot tarthat karban. A mi szoftverünkben szereplő ligákészítő menüpont is ehhez az alkalmazáshoz hasonló funkciókat fog tartalmazni, pontosabban a csapat, játékos és bajnokság készítésének lehetősége, ligákon belüli mérkőzések létrehozása, valamint a mérkőzésekben szereplő csapatok teljesítményét felmérő statisztikák rangsorolása. A mi alkalmazásunk még azt fogja bevezetni a Challenge Place alkalmazás funkcióihoz, hogy több, a jelenlegi ligát átvélező adatot fog megjeleníteni. Pontosabban, a mérkőzéseket megjelenítő ablakunk több hangsúlyt fog fektetni a sportolók egyéni teljesítményére, vagyis minden csapat él-játékosának bajnokságon belüli statisztikáit is meg fogja jeleníteni az oldal. A játékosok személyes teljesítményére tett hangsúlyt fokozza a liga statisztikai rangsorolásai is, mivel nem csak a csapatok rekordjait és performenciáját fogjuk megjeleníteni, hanem a

játékosok teljesítményi adatait is. Képesek leszünk rangsorolni sportolóinkat mindenféle adatkategóriát figyelembe véve, így nem csak a legjobban teljesítő csapatokat, hanem a legjobban teljesítő játékosokat is képesek leszünk figyelemben tartani.

Ezen szabályokat, ötleteket és célokat betartva képesek leszünk egy olyan szoftver létrehozására, amely egy felhasználóbarát felületen adatokat és statisztikákat fogunk tudni lekérni és felhasználni, amint programunk okosan kitervezett funkciói által gyors és megbízható módon metrikákat és kimutatásokat fog létrehozni, amik alapján felhasználóink meg fogják tudni határozni melyik játékos vagy csapat a jobb, valamint, hogy melyikre érdemesebb fogadni.

The screenshot displays the Challenge Place website interface for the VNBA Regular Season. At the top, there's a dark header with the logo 'CHALLENGE.PLACE'. Below it is a blurred background image of a basketball court. The main content area has two tabs: 'Standings' and 'Schedule'. The 'Standings' tab is active, showing the Light Division table. The 'Schedule' tab shows the games for Sunday, January 15th. The table below shows the current standings:

		Standings	W	L	P
1		Atlanta Hawks	49	9	20
2		Minnesota Timberwolves	48	10	19
3		Oklahoma City Thunder	45	13	16
4		Washington Wizards	39	19	10
5		Cleveland Cavaliers	39	19	10
6		Los Angeles Lakers	29	29	0
7		Chicago Bulls	28	30	-1
8		Detroit Pistons	28	30	-1
9		Boston Celtics	24	34	-5
10		Miami Heat	21	37	-8

The 'Schedule' tab shows the following games:

- Sat 1/14/2023 • 3:30 PM GMT+3 FedExForum: MEM 91 — PHI 93
- Sat 1/14/2023 • 3:30 PM GMT+3 Toyota Center: HOU 132 — IND 117
- Sun 1/15/2023 • 5:00 PM GMT+3 AT&T Center: SAS 110 — BKN 97
- Sun 1/15/2023 • 3:30 PM GMT+3 Spectrum Center: CHA 100 — NYK 97

2.2. ábra. Challenge Place weboldalán létrehozott sajátos bajnokság felülete

3. fejezet

Nyelv, eszközök és felépítés

A fentiekben megszabott feladataink sikeres teljesítéséhez több különböző eszközt fog kelleni felhasználnunk. Ezen eszközök közé tartoznak legfontosabban maga az adatbázis, amiben tárolni fogjuk a leimportált adatokat, valamint egy megbízható programozási felület, amiben megírhatjuk az alkalmazásunkat.

3.1. .NET MAUI

A .NET (dotNET) a Microsoft által kitervezett, széles körben használt fejlesztői platform és keretrendszer, mely olyan lehetőségekkel járul a szoftverek fejlesztőihez, amin keresztül különböző alkalmazásokat programozhatunk meg, valamint számos szolgáltatások fejlesztését és futtatását teszi kivitelezhetővé különböző operációs rendszereken és eszközökön keresztül, legyen az Windows, macOS, Linux, vagy akár Android és iOS mobil eszközök is. Ez a platform létrehozásának korszaka óta moduláris, kiterjeszthető architektúrára épült, ami további testreszabási lehetőségeket kínál a programozók számára, így akaratukhoz és ötleteikhez modellezve képesek fejleszteni és bővíteni alkalmazásaiat. A keretrendszer rendelkezik az összes szükséges alapvető könyvtárral és komponenssel, így már a kezdettől rendelkezünk gyakran alkalmazott funkciókkal és adatelérési módszerekkel, amik hálózati kommunikációt és más hasonló feladatokat is képesek elvégezni. A .NET alkalmazások futási környezetét ennek a központi eleme, a CLR (Common Language Runtime) biztosítja, ami legfőbb szerepet tölt be alkalmazásaink lefutásában, valamint memóriakezeléssel, futás alatti optimalizációval és hatékonyságot és biztonságot biztosító eljárásokkal is foglalkozik.

Mi a .NET MAUI verzióját fogjuk használni [Law23], ami leginkább a többplatformos alkalmazások fejlesztésére alkalmazzák. Ez lehetővé teszi a szoftverfejlesztők számára, hogy egységes kódüzemben alapuló programokat alkothassanak és futtassanak különböző környezetekben. A .NET MAUI az előző generációs Xamarin.Forms keretrendszer továbbfejlesztése, a .NET Core és a Mono platformokat használja a futtatáshoz, valamint a platformspecifikus funkcionalitások biztosításához. Míg a hagyományos .NET alkalmazások fejlesztéséhez gyakran külön projektet kell létrehozni ahhoz, hogy bizonyos meghatározott platformokra is optimizálhassuk az alkalmazásunkat, .NET MAUI esetében a “Single Project” modell van alkalmazva. Ez azt biztosítja, hogy az alkalmazás kódja és forrása egységesen kezelhetővé válik egy projekt keretén belül, így egyszerűbbé válik annak karbantartása és fejlesztése, valamint csökken a duplikált kód mennyisége is. A

MAUI verziója a .NET platformnak különböző új vezérlőket és sablonokat használ fel, amelyek fejlesztik az alkalmazás teljesítményét, valamint tovább segítik a megszabott funkcióinknak a sebességét és a felhasználói élményt. Amint láthatjuk, a .NET folyamatosan új fejlődésekben és újításokon megy keresztül, így jött létre a MAUI verzió is, ami bizonyos újabb eszközökkel képes most már több platformon is egyszerre kompatibilis alkalmazások létrehozására.

3.2. C#

A .NET lehetővé teszi a fejlesztőknek a választást különböző programozási nyelvek között, például a C#, a Visual Basic .NET és a C# nyelvek. Ez rugalmasságot nyújt a fejlesztők számára, akik így kiválaszthatják a számukra legmegfelelőbb nyelvet a projektük igényeihez. Mi a projektünkhez a C# nyelvet választottuk, ami a Microsoft által kiépített modern és objektumorientált programozási nyelv, aminek elsődlegesen a legfőbb szerepe egyszerűsíteni Windowson az alkalmazásfejlesztést. A C# a .NET keretrendszer alapnyelveként szolgál, valamint ez a nyelv egy keresztpalatformos nyelvként széles körben elterjedt [Mil19]. Ezek mellett lehetővé teszi, hogy más operációs rendszereken is képesek legyen lefutni ebben a nyelvben megírt alkalmazásaink. Ez a nyelv könnyen tanulható és alkalmazható egyszerű és olvasható szyntaxa miatt, egyúttal támogat számos hasznos fejlesztőbarát jellemzőket, ezek lévén az erős típusosság, szolid memóriakezelés, automatikus szemétgyűjtés, valamint sok fejlett nyelvi funkciókat is képes feldolgozni, mint delegáltakat, eseményeket, lambda kifejezéseket vagy akár a LINQ-t (Language Integrated Query) is. Ezen hasznos eszközök segítségével a programozók hatékonyan és okosan megtervezhetik és kivitelezhetik az alkalmazás logikáját. A C# nyelvet alkalmazva, az alkalmazások fejlesztéséhez a programozók számos eszközt és keretrendszert használhatnak, mint például az ASP.NET, a Windows Forms, a WPF, az Xamarin és a Blazor. A Microsoft folyamatosan fejleszti a C# nyelvet, így a fejlesztők mindenkor a legújabb nyelvi jellemzőket és funkciókat használhatják ki a fejlesztés során. Ez a nyelv erőteljes, rugalmas és sokoldalú, amely lehetővé teszi a fejlesztők számára, hogy egy kiegészített és extenzív funkció-gyűjteménnyel rendelkezve a legjobb minőségű alkalmazásokat hozhassák létre.

3.3. XAML

Az alkalmazásunk felhasználói felületének létrehozásához az XAML nyelvet fogjuk felhasználni, ami egy deklaratív jelölésnyelv. Lehetővé teszi, hogy vizuálisan kifejezzük applikációnk felhasználói felületének struktúráját, elrendezését és megjelenését, valamint, hogy ezeket a C# kódunkhoz köthessük további megjelenítési ötletek és célok kivitelezésére, mint például egyes, a rákötött objektumorientált programozási nyelvünkbeli kiragadott objektumjainknak XAML elemekhez való hozzárendelése. Az XAML rendkívül hasznos eszköznek bizonyul a C# keretrendszeren belül, hisz lehetővé teszi a felhasználói felület kódolását elválasztva az üzleti logikától [CH14]. Ez a gyakorlat nagymértékben javítja a fejlesztés hatékonyságát és karbantarthatóságát, mivel lehetővé teszi a tervezőknek és programozóknak, hogy párhuzamosan dolgozzanak az alkalmazásokon, anélkül hogy átfedésbe kerülnének egymással. Rendkívül rugalmas és számos lehetőséget kínál a felhasználói felületek testreszabására és stílusozására. Az XAML segítségével könnyedén

létrehozhatunk különböző vezérlőket, elrendezéseket, animációkat és adatadatkötéseket. Emellett támogatja az újrafelhasználhatóságot és a komponensalapú fejlesztést is, így lehetővé teszi a fejlesztők számára, hogy egyszerűen létrehozhassanak és karbantarthassanak összetett felületeket. Az XAML és a C# közötti szoros integráció lehetővé teszi a fejlesztők számára, hogy könnyen hozzájussanak az XAML-ben definiált elemekhez és tulajdonságokhoz a C# kódon belül. Így könnyen kezelhetjük az alkalmazás állapotát, annak eseményeit és az adatokat a C# kódunkból. Tehát ennek a két nyelvnek a kombinációja lehetővé teszi a fejlesztők számára, hogy gazdag és lenyűgöző felhasználói felületeket hozzanak létre modern alkalmazásokhoz.

3.4. Visual Studio 2022

A .NET ökoszisztémája gazdag és fejlett fejlesztői eszközöket tartalmaz, mint például a Visual Studio, amely egy integrált fejlesztői környezet (IDE), ennek pontosabban a 2022-es verzióját fogjuk használni. A Visual Studio egy erőteljes és teljes körű fejlesztői környezet, amely kifejezetten az alkalmazások fejlesztésére és karbantartására szolgál, valamint rendkívül népszerű a fejlesztők között, számos kényelmi funkciója miatt, eszközt és funkciókészletet kínál, amelyek megkönnyítik és gyorsítják az alkalmazásfejlesztést [[Str22](#)]. A szoftver lehetőséget nyújt a kód szerkesztésére, hibakeresésre, verziókezelésre, tesztelésre, tervezésre és kiépítésre, valamint számos egyéb fejlesztői munkafolyamat támogatására. A Visual Studio széles körű integrációval rendelkezik, ezek között legkiemelkedőbb példák lévén más Microsoft termékek és szolgáltatások, mint például az Azure felhőalapú platform, a Microsoft SQL Server adatbázis-kezelő rendszer, a Git verziókezelő rendszer, valamint az Azure DevOps projektmenedzsment eszközök. Ez lehetővé teszi a fejlesztők számára, hogy zökkenőmentesen dolgozzanak a fejlesztési folyamat minden szakaszában, valamint egyszerűen kezeljék és nyomon követhessék projektjeiket, amin dolgoznak. Az aktív fejlesztői közösséggel, a folyamatos frissítésekkel és az átfogó dokumentációval további előnyöket nyújtanak a Visual Studio használóinak. Összességében a Visual Studio egy megbízható és teljes körű fejlesztői környezet, amely lehetővé teszi a fejlesztők számára, hogy hatékonyan dolgozzanak az alkalmazások fejlesztésén és karbantartásán. A könnyen kezelhető felület, a széleskörű eszközök és a fejlett funkciók használatával a programozóknak lehetőség járul ahhoz, hogy a lehető legtöbbet hozzák ki a projektjeikből.

3.5. phpMyAdmin és MySql

Ahhoz, hogy adatainkat tárolhassuk és lekérhessük a programunkon belüli funkciókhoz való felhasználásukhoz, egy olyan adatbázis-kezelő rendszert kell használnunk, amelyik képes gyorsan és sikeresen operálnia bármilyen helyesen megadott lekérésünkre, valamint az adatainkat is biztonságosan és jó állapotban tárolja. Ehhez a phpMyAdmin webes alapú adatbázis-kezelő eszközt választottam, amelyet kifejezetten a MySQL adatbázis kezelésére fejlesztettek ki [[Kof05](#)]. Ez egy ingyenes és nyílt forráskódú szoftver, amely lehetővé teszi a felhasználók számára, hogy könnyedén kezeljék és adminisztrálják az adatbázisukat egy intuitív felhasználói felületen keresztül. Népszerű az adatbázis-kezelésben jártas fejlesztők és rendszergazdák körében, mivel egyszerű és hatékony eszköz az adatbázisok és táblák létrehozásához, adatok importálásához és exportálásához, lekérdezések

futtatásához, felhasználók kezeléséhez és számos más feladathoz. Egy böngészőn keresztül érhetjük el és letelepíthetjük a szerverünkre vagy távoli kiszolgálóra. A telepítés és konfiguráció általában egyszerű, és a phpMyAdmin elérhetővé válik a webcím segítségével. Számos hasznos funkciót és eszközt nyújt, amelyek megkönnyítik az adatbázisok adminisztrációját, például hatékony megoldásokat kínál a táblák és mezők szerkesztésére, az adatok keresésére és szűrésére, a lekérdezések és műveletek futtatására, a visszaállítási és biztonsági mentési feladatok végrehajtására, a jogosultságok kezelésére, valamint a teljesítmény és teljesítményfigyelési eszközök használatára.

A phpMyAdmin a MySQL adatbázis rendszert támogatja, amely a legnépszerűbb nyílt forráskódú relációs adatbázis-kezelő rendszer, széles körben használják az alkalmazásfejlesztésben és a webes alkalmazások háttértárolójaként [Wan18]. A MySQL adatbázis-kezelő rendszer nyílt forráskódú, vagyis bárki szabadon használhatja, módosíthatja és fejlesztheti a szoftvert. Kiemelkedő teljesítményt és megbízhatóságot nyújt, amely lehetővé teszi nagy adatforgalmú és üzleti kritikus alkalmazások hatékony működését. A rendszer optimalizált adatbázis-motorral rendelkezik, amely gyors lekérdezéseket és adatmanipulációt, valamint a többfelhasználós környezetek hatékony kezelését is lehetővé teszi. Támogatja a szabványos SQL nyelvet (Structured Query Language), ami egyszerű és hatékony eszköz az adatok tárolására, manipulálására és lekérdezésére. A sima SQL-lel szemben a MySQL bővítményeket és tárolt eljárásokat is támogat, amelyek lehetővé teszik a fejlesztők számára, hogy összetettebb adatmanipulációs és adatfeldolgozási feladatokat hajtsanak végre. A MySQL rendkívül skálázható, így alkalmazkodni tud az adatbázis méretének és a felhasználói igények növekedéséhez. A rendszer támogatja a több szerveren futó adatbázisokat és a replikációt is, amely lehetővé teszi az adatok másolását és szinkronizálását különböző szerverek között. Ez javítja a rendelkezésre állást és a teljesítményt, valamint lehetővé teszi a terheléselosztást és a redundanciát. Ez a rendszer biztonságos és rugalmas adatbázis-kezelést tesz lehetővé, és támogatja a felhasználók jogosultságainak finom beállítását, valamint a biztonsági mentéseket és helyreállítást.

3.6. XAMPP

Ahhoz, hogy alkalmazásunkon belül használhassunk MySQL nyelven alapuló adatbázisokat, ahhoz egy XAMPP nevezetű nyílt forráskódú szoftvercsomagot kell használnunk, ami lehetővé teszi egy teljes fejlesztői környezet létrehozását a számítógépen [Har15]. Ennek célja, hogy egyszerűvé tegye a szerver oldali fejlesztést és tesztelést, különösen webes alkalmazások esetén. Az Apache webszervert használja, amely lehetővé teszi a helyi weboldalak és alkalmazások kiszolgálását a számítógépről. Az XAMPP része a MySQL adatbázis-kezelő rendszer is, amely lehetővé teszi az adatbázisok létrehozását, módosítását és kezelését, így valós környezetben lehetőség nyílik a webalkalmazások és adatbázisok fejlesztésére, anélkül, hogy egy ténylegesen nyilvános webszervert vagy adatbázist kellene használni.

3.7. API

Azon csapatok és játékosok esetében, amelyeket importálni szeretnénk, azokhoz egy megadott forrásból lekérjük a megszabott kérendő adatokat és beiktatjuk a programunk-

ba. Azon belül ezeket az adatokat feldolgozzuk és beletesszük az adatbázisunkba, ahonnan majd különböző feladatok esetében lekérjük és különböző műveleteket végzünk velük. Ennek elérésére egy olyan forrásból kell megszereznünk a beiktatandó információt, ahol nem csak az adatok helyességére és pontosságára tesszük a hangsúlyt, hanem annak lekérési módszerének formájára is. Ez azt jelenti, hogy amennyire tudjuk próbáljuk mellőzni az olyan forrásokat, ahonnan az adatok egy túlzsúfolt és komplikált szerkezetben lehet lekérni. Az adatokat JSON objektumok formájában lehet lekérni, amiken belül különböző adatalemekre van osztva [Bas15]. Ezek az elemek lehetnek egyéni adatok mint nevek vagy számok, vagy egy összetettebb, komplexebb esetben lehetnek tömbök, amiken belül van egy vagy több egyéni információ. A lekérendő adatok zsúfoltsága így úgy magyarázható, hogy a lekért objektum ne tartalmazzon túl sok értékterrel adatot, ezek elérési neve ne legyen túl komplikált vagy érthetetlen, vagy tömbök esetében ne legyen túl sok rengeteg elemmel ellátott elem-tömb. Innen következtetve, egy olyan adatforrást kell szerezni, ami a meghatározott kategórián belül minél relevánsabb és lényegretörőbb objektumot térítsen vissza, amiből könnyedén elérhetjük a számunkra fontos információt, ami egyben igaz és pontos adatokat képviselnek.

A forrás, amiből szerezni fogjuk az alkalmazásunknak fontos játékos vagy csapat importálási funkciójához kulcsfontosságú adatokat, egy NBA API lesz a RapidAPI hivatalos weboldaláról. Az Application Programming Interface (röviden API) lehetővé teszi a szoftverek és alkalmazások közötti kommunikációt és adatcsere folyamatát [AJ18]. Ez egy interfész, amely meghatározza, hogy az alkalmazások hogyan kommunikálhatnak egymással, és milyen módon érhetnek el és manipulálhatnak adatokat. A mi API-nk esetében ennek segítségével fogunk tudni egy meghatározott link segítségével lekérni JSON objektumok formájában különböző NBA ligán alapuló adatokat és elemeket. Ezeket a linkekkel olyan formátumban is megadhatjuk programunknak, hogy bizonyos kulcsszavakkal is elláthatjuk ahoz, hogy kritériumokat szabhamunk meg az API-nak abban, hogy mennyit és melyik játékosokat vagy csapatokat kérjük le. Annak érdekében, hogy az alkalmazásunkat ne lássuk el rengeteg objektummal és adattal egyszerre, az API-ból való lekérést úgy szabták meg a szolgáltatás létrehozói, hogy meg lehet szabni a linken belül, hogy "oldalanként" mennyi objektumot térítsen vissza egyszerre. Így meg lehet adni, hogy például egy megadott játékos 82 lejátszott mérkőzéséből egy ciklus alatt csak 20-at kérjünk le, így meg lehet szabni a cikluson belül, hogy a lekérési linknél a paraméterek közé meghatározzuk, hogy a limit az 20 és hogy az oldalszám nőjön minden ciklusban, így 20 elemként betudjuk iktatni programunkba az összes 82 mérkőzést. Az oldalszám és a lekérési határon kívül léteznek az alapvető keresési kritériumok, amik alapján kereshetünk bizonyos értékeket. A linken belül megszabhatjuk, hogy milyen szó vagy kulcs alapján szűrje meg a lekért játékosokat vagy csapatokat, hogy melyik évszámbeli mérkőzéseket térítse vissza és más hasonló megszabható feltételeket. Ezen kritériumok segítségével könnyíthetjük munkánkat az importálandó adatok megszerzésével kapcsolatban.

A fentiekben felsorolt eszközök, platformok és programozási nyelvek segítségével képesek leszünk létrehozni alkalmazásunk összes kiszabott funkciót és eljárásait. Ezen elemek megfelelő és helyes használatával kivitelezhetjük megszabott céljainkat és terveinket, valamint segítségükkel karban is tarthatjuk alkalmazásunk adatait és működését becsületes módon.

4. fejezet

Kód felépítése, szerkezete

Ahhoz, hogy a programunk kifejlesztése sikeres legyen, valamint, hogy minden egyes része az alkalmazásunk létrehozásának semmilyen probléma nélkül menjen végbe, nem lesz elég a szükséges eszközök megszerzése és a programozási nyelvek elsajátítása. Az applikáció-készítés egyik nélkülözhetetlen szakasza az alkalmazás létrehozásának tervének meghatározása és betartása szoftverfejlesztés során. Így megszabhatunk egy alapot, egy menetrendet, ami alapján haladunk programunk létrehozása során. Egy ilyen menetrend nélkül az alkalmazás készítésének menete nagyon összezárt lehet, így könnyen elveszíthetjük a kreatív munkavégzési fonalat, ami a munka haladásának és minőségének hátráltatásához vezethet.

4.1. MVVM és a Felhasználói Felület Bevezetése

Mint minden hasonló létrehozott szoftvernél, előbb egy valamennyire megfelelő felhasználói felületet kell létrehoznunk ahhoz, hogy meg tudjunk jeleníteni bármilyen tesztelendő vagy funkciót képviselő adatot. Ehhez a felülethez aztán megfelelő elvégzendő kódot kell megírnunk, amivel kitöljük a megszabott felületi elemeket, mint szövegek, listák és más hasonló. Itt a felületet, amint fentebb említettem, XAML kódban fogjuk megírni, míg az elvégzendő kódot C#-ban.

Az alkalmazásunk pontos és sikerdús teljesítményéhez egy jól kidolgozott architektúrát kell alkalmazzunk. Ehhez egy MVVM (Model-View-ViewModel) tervezési mintán fogjuk alapozni szoftverünket, aminek lényege az alkalmazások elkülönítése az adatoktól és a felhasználói felülettől, valamint az üzleti logika és a felhasználói felület közötti hatékony kommunikáció biztosítása [Wei16]. Amint a neve is sugallja, az MVVM három fontos részből áll.

A modell, ami az alkalmazás adatainak reprezentációja, felelős az adatok tárolásáért, kezeléséért és manipulálásáért. Ez lehet adatbázisból származó adat, hálózatról letöltött információ vagy bármilyen más adatforrás, valamint ez a komponens független a többi két komponenstől.

A nézet a felhasználói felületet reprezentálja, amellyel a felhasználók interakcióba léphetnek. Ez egy grafikus felület, ami felelős az adatok megjelenítéséért és a felhasználói események kezeléséért. Ez a rész is függetlenül működik a többi 2 résztől.

A nézetmodell közvetítőként működik a modell és a nézet között. Kapcsolatot tart a két komponens között és felelős az adatok átalakításáért és kiszolgálásáért a nézet

számára. Tartalmazza az üzleti logikát, az adatkezelést és a felhasználói interakciók feldolgozását. A nézetmodell lehetővé teszi, hogy a nézetek könnyen kommunikáljanak a modellel, anélkül, hogy közvetlenül függősége lenne az adatoktól vagy az üzleti logikától.

Ennek az architektúrának előnyei közé tartozik az erős elkülönítés az adatok, a nézet és a nézetmodell között, ami tesztelhetőséget és karbantarthatóságot eredményez. Továbbá támogatja a könnyű felhasználói felületi változtatásokat, mivel a nézet és a nézetmodell közötti kommunikáció eseményalapú, és a nézetek automatikusan frissülnek, ha a modell változik. Innen láthatjuk, hogy az MVVM lehetővé teszi a kódmezők újra felhasználását és a fejlesztési folyamat hatékonyúságának növelését.

4.2. Szükséges Könyvtárak

A kódunkon belül, hogy elérhessünk bizonyos adatbázis-kezelő vagy architekturális funkciókat és lehetőségeket, ahhoz különböző könyvtárakat kell letöltenünk a már létező könyvtárak mellé [BD13]. Az alkalmazásomhoz több specifikus könyvtárat hozzátelepítettem, mindegyik saját céllal és feladattal rendelkezve programomon belül. Ezek közé tartozik a CommunityToolkit.Mvvm [Mic] csomag, ami a .NET MVVM strukturális alapvető funkcióit tartalmazza, a MySQL.EntityFrameworkCore [MyS] ami egy MySQL szerver-alapú adatbázis elérhetőségét és működését biztosítja, a Newtonsoft.JSON [New] ami JSON alapú objektumokkal való dolgozást engedélyezi, a Syncfusion könyvtárai ki-mutatásokat és metrikákat megjelenítő gráfok létrehozása érdekében [Syn], és még sok más.

4.3. Osztályok (Classes)

Ezek mellé bevezethetjük a specifikusan megszabott osztály-típusokat is, amik segítségével létrehozhatunk sajátos objektumokat, amiket felhasználhatunk a kódunkban. Ez továbbá könnyítheti dolgunkat, hisz ezek külön elem-típusoknak vannak meghatározva, így egy sajátosan megszabott osztályon belül számos különböző típusú változót lementhetünk, valamint ezekre az osztálybeli elemekre simán és röviden rá lehet hivatkozni amikor fel akarjuk használni a kódon belül. A programunkban az osztályok az adatbázisunkból lekért objektumokat fogják képviselni, mint például egy játékos statisztikáinak összességét. Így elérhetjük azt, hogy ne úgy mentse le a sportoló adatait, hogy külön kódot átvevő változókat definiálunk és változtatunk, hanem egy létrehozott "Statisztika" osztályon belül lementjük ezeket az adatokat, így csak egy változóval dolgozunk. Ez azt az előnyt adja nekünk, hogy egy rendezettabb program fenntartását teszi nekünk lehetővé, a kódunk sokkal átláthatóbbá és megérthetőbbé válik. Külön megszabott classesek által még jobban is el fogjuk tudni különíteni típusainkat, amiket megszabunk az alkalmazásunk kódján belül. Így különbséget fogunk tudni tenni az importált és létrehozott játékosok, csapatok és statisztikák között, hisz attól lehet, hogy minden típusú félnek léteznek megegyező elemei, mint pontok, gólpásszok (assists), lepattanók száma (rebounds), viszont létrehozásuk vagy alkalmazásuk szempontjából létezhetnek eltérő elemek is az osztályokon belül, ami által nem lennék képesek az egyik létrehozott játékost lementeni egy importált játékosnak szánt osztályba, meg fordítva. Ilyen a lekért sportolóknál a szezonbeli statisztikák, hisz az importált játékosoknál a különböző átlagolt teljesítmé-

nyeket szezonokra különítjük, míg a létrehozott játékosoknál pedig létrehozott ligákra, így létezik az importált feleknél egy “season” elem a táblán belül, ahol lementjük, hogy melyik év statisztikáiról van szó jelenleg. Ennek függvényében létezhetnek teljesen meggyező osztályok is, amelyeknek különböző nevük és szerepük van, ilyenek a létrehozott és importált felek információi, így programunkon belül az általunk létrehozott játékosunk ugyanannyira autentikusnak számít, mint a már létező élsportolók, akiket lekértünk az alkalmazásunkba. Kódon belül is segíthet ez az osztályok közötti különbségteképzés, pontosabban típusellenőrzések nél. Így programunkban leelenőrizhetjük, hogy a felhasználó jelenleg egy létrehozott vagy beimportált játékost vagy csapatot választott, ami alapján fogunk tudni külön helyzettől függően megírni kódunkban a választás következményét.

4.4. MVVM összekötése a felülettel

Az MVVM struktúra használata, amint említettem a fentiekben, lehetővé teszi számunkra, hogy a programkódunkon belül szabjunk meg bizonyos változókat, amiket könnyedén összeköthetünk az XAML felhasználói felületet felépítő kódunkon belüli elemekkel. Ezeket a ViewModel-t meghatározó kódunkon belül “ObservableProperty”-knek nevezzük. Az ObservableProperty sorral ellátott változóinkat ugyanúgy változtathatjuk a program lefutásának folyamán, valamint ugyanolyan típusúak is lehetnek, mint bármilyen hétköznapi változó. A legfőbb meghatározó különbség az ObservableProperty változók és a minden nap változók között ezek összeköthetősége a felhasználói felülettel, hisz ezen elemek segítségével tudjuk meghatározni az applikációnk milyen változó információt fog megjeleníteni a felhasználó számára. Ezeket a változókat egy úgynevezett “Binding” eljárással tudjuk összekötni. Bármilyen meghatározható tulajdonságánál a megadott felületi elemek, legyen az sima szöveg, egy gomb, egy lista, vagy bármi hasonló, ahelyett, hogy megadnánk a tulajdonságot megszabó értéket egy string formátumban, a “Binding” segítségével összekapcsolhatjuk egy, a ViewModeken belül megszabott ObservableProperty változával. Ez ahhoz vezet, hogy minden egyes alkalommal mikor megváltozik a változónk értéke, a megadott felhasználói felületet megszabó elem tulajdonsága is megváltozik, pontosabban a változó értékére. A változó és felhasználói felületi elem ezen módszerrel való összekapcsolásával elérhetünk különböző előnyöket, mint egy szöveg folytonos, reaktív változtatása, vagy akár az elem stíluszkai tulajdonságainak átállítása, legyen az a szöveg színe, mérete, láthatósága és más. “Binding” segítségével egy teljes listát is kitölthetünk különböző elemekkel, amit szintén ObservableProperty elemekkel való összekapcsolással érünk el. Elemei esetében, ahelyett, hogy egy “List” vagy “Array” típusú listát szabnánk meg, egy “ObservableCollection” elemsorba mentjük el az elemeinket, amit majd a meghatározott “ListView” felületi elemet képviselő listával kapcsoljuk össze. Abban az esetben, ha az “ObservableCollection” gyűjteményünkbe megszabott osztályokat mentünk le, a “ListView” elemein belül, legyenek azok bármilyen elemek, amik rendelkeznek testreszabható szöveggel, megadhatjuk, hogy a megadott osztályú objektumunknak melyik elemének szövegét állítsa be a megadott elem szövegének. Így egy meghatározott osztály összes elemét képesek leszünk felsorolni a “ListView” listánkon belül. Végül, amit mi még alkalmazni fogunk, “Binding” használatával fogjuk összekötni bizonyos metódusainkat elemeinkkel, amik rendelkeznek végbevhető eseményekkel ellátott funkciókkal, ami nálunk leginkább gombok nyomására fog alapozódni. Azon függvényeket, amiket összekapcsolnánk XAML elemeinkkel, azokat “RelayCommand”-ként

határozzuk meg. Ez a metódus hasonlít a sima változók összekapcsolásával, az elemknél a megszabott tulajdonság a Command lesz, ahol megadjuk a bindolandó metódus nevét, ellátva a “Command” szóval. Így egy “NextPage” nevezetű függvénynél úgy lesz megadva a megnyomandó gombnak a tulajdondonságainál, hogy ‘Command=”{ Binding NextPageCommand } ”’, így a megszabott gomb megnyomásával a program meg fogja hívni a NextPage metódust.

4.5. Egyéb jellegzetes elemek

A ViewModel-t meghatározó elemeken és funkciókon kívül jellegzetes kiemelünk más kulcsfontosságú részeket is a kódunkon belül. Az egyik a MySql és phpMyAdminnal kapcsolatot képző elemek. Ezek közül kiemelem a MySqlConnection-t, ami által létrehozhatjuk a kapcsolatot a program és a phpMyAdmin felület között, a MySqlCommand, amiben megszabjuk egy string segítségével az elvégzendő lekérést a felületen belül, valamint a MySqlReader amibe le fogjuk tudni kérni a felületből lekért adatokat. Ezekben kívül jellegzetes megadni a MySqlConnection paramétereit, pontosabban a lokális adatbázis eléréséhez szükséges adatokat, ezek lévén a szerver, a felhasználói kulcs, a jelszó és az elérő adatbázis. Egy másik jellegzetes része az API-ból lekérendő adatokhoz szükséges változók, amik által elérhetjük az API funkciót és lekérhetünk belőle használandó adatokat. Ehhez szükségünk lesz egy WebClient változóhoz, aminek megadva a megfelelő linket letölthetjük az API-ból kellő adatainkat és egy string változóba mentjük, aztán ezt egy JObject elemre alakítva azt lementjük egy dinamikus változóba, ahonnan majd felhasználhatjuk az objektumból az adatokat külön elem-címeket megadva.

5. fejezet

Alkalmazás Bemutatása

Sok olyan alkalmazás létezik, ahol megnézhetünk bizonyos statisztikákat és kimutatásokat egy adott sportágban, valamint olyan alkalmazások, ahol fogadhatunk is bizonyos játékosokra és sportklubbakra, csapatokra, viszont inkább legfőként focival vagy ritkábban tenisszel kapcsolatosak csak. Valamint egy személy simán utánanézhet ezen statisztikáknak, attól függetlenül, hogy milyen évről vagy csapatokról van szó, viszont azt nem tudja helyben összehasonlítani fejlett statisztikák szempontjából egy másik bármilyen megadott csapattal vagy játékossal.

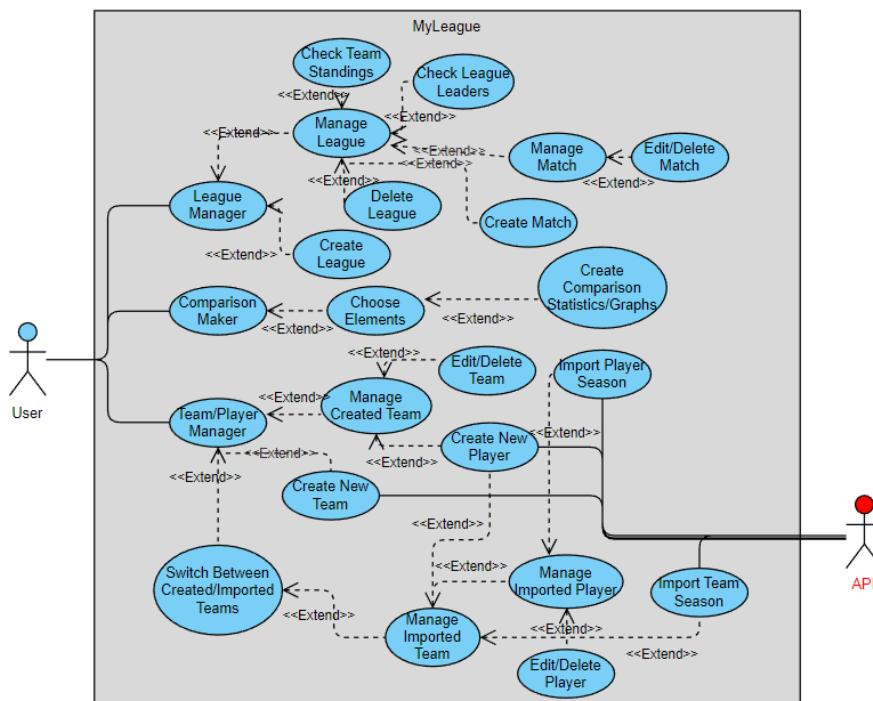
Az általam készített alkalmazás egy C# nyelven alapuló .NET MAUI alkalmazás, amelynek fő lényege bizonyos kosárlabda csapatok és játékosok statisztikai kimutatása és értékelése, más csapatokkal és játékosokkal szemben való összehasonlítás és kiértékelés, valamint egy opció, hogy saját kosárlabda ligát készíthessünk általunk készített csapatokkal és játékosokkal. Ezen ligákon belül bármilyen mérkőzéseket megszervezhetünk két általunk kiválasztott csapat között, ahol szintén bizonyos statisztikákat szabhatunk meg a jelenlegi mérkőzással kapcsolatban, valamint alkalmazásunk a bajnokságunkban szereplő klubok és atléták statisztikáit is kimutathatjuk, valamint más létrehozott kosárlabda csapathoz össze tudjuk hasonlítani és értékelni ezeket. Alkalmazásom ezek mellett csapatokkal kapcsolatos információkban és azok prezentációjában is gazdagabb, mint más bajnokság készítő weboldal vagy alkalmazás, hisz általunk létrehozott csapatainkat és játékosainkat testreszabhatjuk és modifikálhatjuk, valamint számos érdekes adatokkal és tudnivalókkal elláthatjuk őket ahhoz, hogy minél jobban hasonlítsanak más valódi csapatokhoz, ami egy autentikusabb és különlegesebb felhasználói élményt garantál.

A kigenerált statisztikai adatok által számos előretörő információkat nyerhetünk ki, amikkel minél pontosabban és becsületesebben tudjuk kiértékelni saját vagy akár kiválasztott létező csapatainkat és annak játékosait. Ezek között a legfőbb cél más csapatokkal és játékosokkal való összemérettetés, életbevágó és nélkülözhetetlen adatok általi összehasonlítás, amivel bizonyos értékek kiszámítására vezetnek, amik tökéletesen leírják csapatunk vagy játékosunk teljesítményét és sikerét, pontosabban fejlett adatokkal egy pontos értékelést tudunk adni nekik, amivel szintén más statisztikai kimutatásokat és ötleteket vihetünk végbe, mint csapatok kifejtett leírása, más bizonyos csapatokkal való összemérések, vagy akár aránylagos kimutatások.

6. fejezet

Funkciók bemutatása

Alkalmazásunk fő megszabott céljait figyelembe véve három elkülönítendő részre lehet osztani a program alapvető funkcióit. Ez a háromrészes szerkezet már a főoldalon észrevehető, amint a felhasználó elindítja az alkalmazást, három gomb közül választhat. Ez a három gomb szabja meg a három kulcsfontosságú részét a szoftverünknek: a liga-készítő aspektusa a programnak, ahol a már létrehozott csapatainkat mérkőzésekbe tehetjük és egy szezon szimulálhatunk sajátosan meghatározott statisztikákkal és meccskimenetelekkel; az összemérgettetéseket végbevivő menünk, ahol importált csapatokat és játékosokat összehasonlíthatunk különböző komplex statisztika segítségével; és a feleket létrehozó és átalakító részleg, ahol csapatokat és azon belül játékosokat hozhatunk létre és testreszabhatunk, valamint itt importálhatunk különböző klubokat, sportolókat és évszám szerinti teljesítményeket.



6.1. ábra. Az alkalmazás Use Case diagramja

A fenti Use Case diagram segítségével át lehet látni, hogy a felhasználó milyen funkciókkal rendelkezik az alkalmazás elindításakor, valamint hogy azon opciók milyen más lehetőségekkel bírnak. Az ábrát a következő fejezetekben fogom részletesen bemutatni.



6.2. ábra. Az alkalmazás fő oldala, ami megjelenik indításkor

6.1. Első menü: Ligakészítés

A főmenü bal oldali gombjának megnyomásával a felhasználó elér a ligakészítő oldalra, ahol létrehozhat egy új ligát, valamint a már létrehozott ligák közül kiválaszthat egyet. Amint kiválaszt egy bajnokságot, az alkalmazás megnyitja annak főoldalát, ahol már létrehozott mérkőzések eredményeit fogja megkapni, valamint különböző más funkciókat is képes elérni.

Ezen funkciók közé tartoznak a mérkőzések létrehozása, ahol meg kell adjon 2 küzdő felet és egy dátumot, mikor fog lezajlani a megadott meccs, különböző ligabelüli kímutasok és rangsorolások megtekintése, valamint az opció a liga törlésére. Ezen az oldalon a felhasználó minden kulcsfontosságú opciót el tud érni a liga karbantartásához, ami a mérkőzések elérése, azok módosítása és megszervezése, és a csapatok és játékosok teljesítményének felmérése. Ha a user megnyomja az egyik mérkőzést, elér egy oldalra, ahol megjelennek a mérkőzés felei, a végső eredmény, a dátum és a helyiség, ahol megszervezésre került a meccs, valamint a két csapat legjobban teljesítő játékosainak három alapvető statisztikája a ligán belül (pontok, assistok és reboundok). Itt két opció lesz elérhető, egy a mérkőzés statisztikáinak megszabására, ahol a felhasználó megadhatja a végeredményt és a játékosok egyéni teljesítményeit, valamint a meccs törlésének lehetősége.

Amint említettem a Challenge Place alkalmazásnál a fentiekben, nem csak egy csapatok nyeréseinek vagy összpontszámának függvényében létrehozott rangsorolásokat tekinthet meg a user, hanem a csapatokban szereplő játékosok egyéni teljesítményeit felmérő listákat is megtekinthet. A csapatokat felmérő rangsorolás tartalmazza a klubok nyeréseit és veszítéseit, valamint az összesen általuk és ellenfelük által megszerzett pontok számát. Ez a lista a csapatokat rekord alapján helyezi sorrendbe, amit a nyerések és veszítések aránya adja meg. Abban az esetben, ha két vagy több csapatnak ugyanaz a rekordja, az

a csapat szerzi az előnyt a többi fölött amelyiknek a legnagyobb az összesen megszerzett pontok száma, utána meg az a csapat, amelyik a legkevesebb pontot engedte megszereznie az ellenfeleinek. Játékosok esetében, a ranglista tartalmazza az összes alapvető kosárlabda statisztikát, ami nélkülözhetetlen egy fél hatékonyiságának felmérésséhez. Ezek a statisztikák a megszerzett pontok, reboundok, assistok, stealek, blockok, valamint a dobások pontosságát felmérő statisztikák, amikből a három pontos dobások százalékos arányát jelenítjük meg. A felhasználó, a statisztika nevére kattintva, tetszőlegesen rendezheti el a bajnokságban szereplő sportolókat. Bármilyen megjelenített statisztika alapján sorrendbe tehetjük játékosainkat, beleértve a nevük alfabetikus sorrendjén alapuló rendezést is, ha a user a “Player” feliratra kattint.

A fentiekben megszabott funkciók és oldalak által létre tudunk hozni egy sima, de valamilyen szinten extenzív bajnokság létrehozó és karbantartó menüt, ahol nem csak meccseket hozhatunk létre, hanem azokat felmérhetjük és elemzésekre használható ranglistákat alkothatunk belőlük.

The screenshot shows the NBA app's League Leaders screen. At the top, there are three tabs: NEW MATCH, STATISTICS, and LEAGUE LEADERS. The STATISTICS tab is selected. Below the tabs, there is a table of NBA team statistics:

Team	Score	Opponent	Date
Toronto Raptors	98	95	Cleveland Cavaliers 3/9/2023
Houston Rockets	143	138	Philadelphia 76ers 3/9/2023
Charlotte Hornets	121	109	Milwaukee Bucks 3/23/2023
New York Knicks	0	0	Los Angeles Lakers 7/7/2023
New Orleans Pelicans	0	0	Detroit Pistons 7/11/2023

At the bottom left is a red "DELETE LEAGUE" button, and at the bottom right is an info icon (i inside a circle).

6.3. ábra. A ligakészítőn belül egy liga főoldala, ahonnan meccseket és rangsorolásokat érhetünk el

6.2. Második menü: Felek Létrehozása és Importálása

6.2.1. Alap Felület

Az alkalmazás második jellegzetes menüpontját a jobb oldali gomb megnyomásával érhetjük el, ahol a szoftver alapvető funkcióihoz nélkülözhetetlen felek, azaz csapatok és játékosok létrehozásának lehetőségét kínálja. Ennél a menüpontnál a felhasználó választhat, hogy milyen módszerrel alakítja meg a programon belül a felhasználó csapatokat, valamint azokat testre szabhatja és azokon belül játékosokat is megszabhat a megadott csapathoz.

Ennél a második menünél a usernek két választása fog adódni a csapatok létrehozásához, amit kiválaszthat a “Create Team” gombbal. Ennek lenyomásával az applikáció megnyit egy oldalt, ahol megadhatunk a csapatot képviselő információt, mint név, város, stadion, ahol játszani fognak és egy képet, ami a klub logója lesz. Abban az esetben,

ha a felhasználó nem szeretne létrehozni egy sajátos csapatot, hanem inkább importolna egyet a programhoz kötött kosárlabda API-ból, akkor a megadott listából kiválaszthat egy létező csapatot. Ilyenkor a felhasználónak nem kell kitöltenie semmilyen adatot a csapatok létrehozásához használt felületi elemekbe, hanem csak egy csapatot szelektál a megadott listából. Ha a felhasználó valóban csak egy klubot választott a megadott menüben, akkor a program kiad egy ellenőrzési ablakot az importálandó csapat adatairól. Miután a user leellenőrizte a csapat információinak helyességét, egy gombnyomás erejével importálhatja a lokális adatbázisba, amit majd bármikor elérhet az alkalmazás bizonyos menüpontjaiban.

Amiután a program belehelyezte a kívánt csapatot az adatbázisba, a felhasználó azt látni fogja a második menüpont főoldalán, ahol a számára elérhető csapatok lesznek felsorolva. Elsődlegesen a szoftver a saját kezüleg létrehozott klubokat mutatja meg számunkra, de a csapat létrehozásának gombja mellett gombot megnyomja (Pontosabban a “Show Sampled Team” gomb), akkor a programunk átvált a felhasználó által, API-ból lekért csapatok felsorolására.

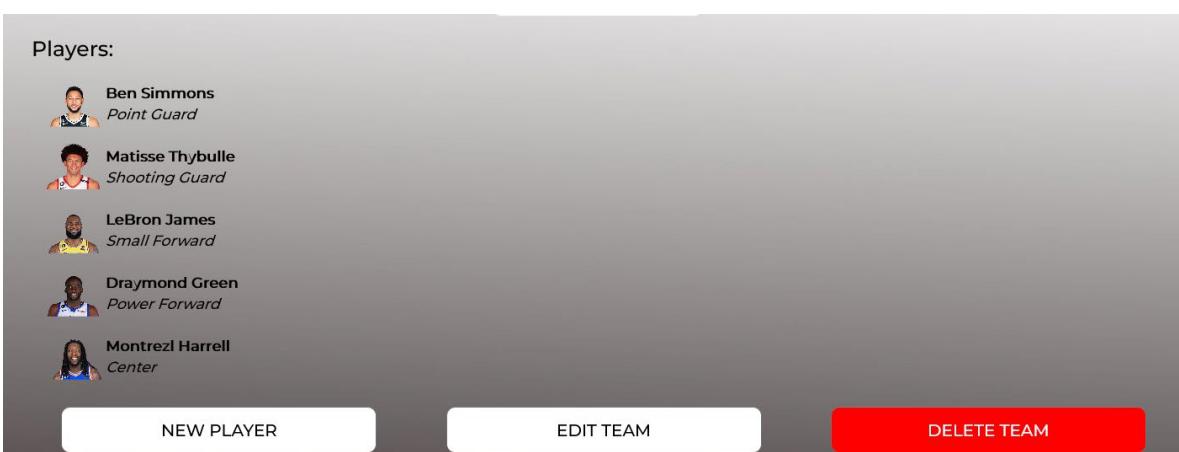
6.2.2. Csapatok Felülete

A két gombunk mellett a user ráklikkelhet egy megadott csapatra is a listából, ami által a program elvisz egy új oldalra, ahol a kiválasztott csapatot mutatja be különböző adatok és statisztikák segítségével, annak a csapatnak a játékosait elérhetővé teszi különböző funkciók elvégzéséhez, valamint ott fogunk tudni állítani bizonyos testreszabási opciókat létrehozott klubjainknál. Ez a felület szignifikánsan különbözik a létrehozott és az importált csapatok között, hisz különböző adatszettel rendelkeznek egymáshoz viszonyítva.

A létrehozott csapatoknál a felhasználó számára elérhető lesz a csapat városa és állama, a stadionja, amit megnyithat a térképen egy gomb segítségével, valamint a csapat játékosai is fel vannak listázva. Az importált csapatoknál megjelennek a csapat neve és városa, valamint annak importált szezonjai részletezve és a lekért sportolók, akik jelenleg a megadott csapatnak játszanak. A létrehozott és a lekért csapatok közötti különbség leginkább az alkalmazásban betöltött szerepükönél érződik, mivel a létrehozott csapatoknál esztétikai jellegzetességeket és adatokat adhatunk meg, hogy azok prezentációja minél kifejlettebb legyen a ligákon belül, míg az importált csapatoknál a megadott API-ból is leginkább csak statisztikailag jellegzetes információt adhatunk meg, mivel azoknak legfőbb lényege teljesítményt és hatékonyságot felmérő metrikák és kimutatások létrehozása és elemzése. Mindkét csapattípusnál két választható opció fog megjelenni, ami hasonlóképpen működik minden esetben: a játékosok létrehozása és a csapat törlése az adatbázisból.



6.4. ábra. Létrehozott csapatot bemutató felület



6.5. ábra. Létrehozott csapatot bemutató felületen a játékosok lista

6.2.3. Játékosok Importálása

Míg a klub törlésének opciója csak egy ablakban felugrott kérdés által vihető végbe, a sportolók létrehozásának folyamata hasonlóképpen működik, mint a csapatok létrehozásának folyamata. Játékosok létrehozása esetében is a felhasználó megadhat különböző kulcsfontosságú információt a létrehozandó kosárlabdázónkról, az lévén a keresztnév és családnév, a pozíció, amiben leginkább játszik a játékosunk, valamint egy kép a megadott játékosról. Ami különbözik a csapatok importálási módszerétől az maga a lekérést meg-hívó folyamat, hisz itt lesz egy külön gomb, amit, ha a user megnyom, akkor lekérhet az API-ból egy megadott játékost, ahelyett, hogy létrehozna egyet. A gomb megnyomásával a program kidob egy ablakot, ahol a felhasználó megadhat egy szót, ami szerint kereszen az alkalmazásunk játékost, akit majd beimportálunk a rendszerünkbe. Ez a módszer leginkább azért különbözik a csapatok importálási módszerétől, mert míg a klubok esetében legtöbb esetben csak az NBA 30 hivatalos csapata közül választhatunk egyet, a játékosok esetében több mint 500 különböző sportoló vesz részt minden évben az ameri-

kai kosárlabda ligában. Csak tavaly számoltak 540 jelenleg aktív játékost, aki részt vett legalább egy NBA mérkőzésen [Hoo]. Ezen ok miatt vezettem be inkább egy olyan lekérési módszert, ahol a user megadhat egy keresési paramétert, ami által limitálhatjuk az egyszerre lekért játékosokat tartalmazó lista nagyságát, ahonnan majd a felhasználó ki-választhatja az importálandó sportolót. A felugró ablakban a user megad egy szót, amit a program felhasznál ahhoz, hogy lekérje az összes olyan játékost, akinek a keresztnévben vagy a családnevében szerepel a megadott szó. Ilyenkor a szoftver elvisz egy új oldalra, ahol felsorolja az összes sportolót, aki megfelelt a keresési kritériumnak. Itt a felhasználó kiválasztja a kívánt játékost, hasonlóan a csapatok importálásánál a program kiad egy ablakot az összes lekért adatról, amit lekért a sportolóról, és az adatok ellenőrzése után a user egy gombnyomással belehelyezheti a játékost az adatbázisba. A program ekkor visszakeríti a sampleelt csapatokat megjelenítő oldalt, ahonnan, ha ráklikkelünk a csapatra, ahol játszik az előbbiekbén importált játékosunk, akkor a játékosok listájában meg fog jelenni az importált atlétánk.

6.2.4. Statisztikák Importálása

A játékosok importálási listáját létrehozó funkciónkhoz hasonlóan, a programunkban megoldott statisztika-lekérési módszer is egy keresési ablakon keresztül működik. Ezt egy harmadik gomb által érhetjük el az importált játékosokat képviselő oldalon, ami az “Import Season” nevezetű elemünk lesz. Ezen gomb lenyomásával a program kiad egy ablakot, ahol kéri a felhasználót, hogy adjon meg egy szezonot, amelyiket importálni szeretné lekért csapata számára. A megadott szezon egy évszámot fog képviselni, avagy azt az évet, amikor kezdődött a kívánt szezon. Erre egy példa az, ha a user megadja a programnak, hogy “2018”, akkor a szoftver a 2018-2019-es szezon fogja lekérni, mivel minden évben a szezon összel kezdődik és tavasz végén ér véget. Abban az esetben, ha a felhasználó egy invalid évszámot ad meg, akkor a program nem fogja elvégezni a lekérést és meg fogja kérni, hogy egy megfelelő évszámot adjon meg.

A szezon képviselő statisztikákat hasonló módon a program egy kis-ablak formájában megjeleníti a felhasználó számára, hogy tehessen az adatok helyességéről mielőtt bevezetné a rendszerbe. Az API-ból lekérhető mérkőzések összes lehetséges teljesítményeit lekéri a programunk, ezek között lévén a pontok, reboundok, assistok, próbált és sikeres rendes, szabad vagy hárompontos lövések száma, és még sok más érdekfeszítően jellegzetes adatok a játékos teljesítményéről a jelenlegi meccsről. A mi adatbázisunk két formában fogja lementeni ezeket az adatokat. Az egyiket a “sampled_player_stats” nevezetű táblában, ahol az összes előbbiekbén említett statisztikát lementünk egy mérkőzésbe, így egy játékos személyes produktivitását hozzáköthetjük egy megadott mérkőzéshez, ami egyúttal tartalmazza a meccs kulcsát és a szezon amikor játszódott a mérkőzés, amit a sportoló kulcsával és egyéni teljesítményeivel fog összekapcsolni az adatbázis. A másik forma az a szezon képviselő statisztikák a “seasonal_stats” táblában, ahol az épp lekért szezonbeli mérkőzéseit egy bizonyos játékosnak a program átjárja, meccsek száma szerint átlagot számol az összes statisztikából, és egy szezon alá menti el. Így a két formát úgy különítjük el, hogy az egyik egy megadott mérkőzést és azalatt lezajlott teljesítményeket képvisel, míg a másik sok mérkőzésből álló szezon képvisel. Így könnyíthetünk munkánkon, hisz nem kell minden egyes alkalommal újraszámolni egy megadott szezonban

lezajlott meccsekben elért statisztikák átlagát, hanem csak simán rá tudunk hivatkozni egy megadott szezon és játékos-kulcs által.

Csapatok esetében is megegyezik a statisztikákat lekérő tervünk, abban, hogy a lehető összes információt lekérünk a mérkőzésekről, a lekért meccseket belehelyezzük egy dezignált táblába, aztán meg a program számol átlagot és más alapvető adatot a csapat szezonjáról és egy szezonokat képviselő táblába helyezi. Az API-ban is két különböző adattípus szerint tudjuk megkülönböztetni a játékos és a csapat teljesítményét egy meccsen belül: a játékosokhoz a “Stats” objektumokat kérjük le, míg a csapatokhoz a “Games” objektumokat. Ez a két objektum abban tér el, hogy a “Games” objektum a két csapatot szabja meg, az általuk elért végső eredményt, és a mérkőzés alapvető adatait, mint dátum vagy szezon, míg a “Stats” objektum tartalmazza a lekért játékos személyes statisztikáit is, nem csak a mérkőzés végeredményét. Csapatoknál is egy megadott évszám által kérünk le bizonyos szezonokat, a lekért adatok meccsenként meg a két csapat kulcsa, a két csapat végeredménye és a mérkőzést megszabó említett adatok. Amint említettem, ezekről is készít a program átlagokat, és egy megadott szezon alá menti le jelenlegi csapat-kulccsal együtt. Az ide lementett adatok a szerzett pontok átlaga meccsenként, az ellenfelük által szerzett pontok átlaga, valamint nyerések és a vereségek száma a megadott szezonban, ami által rekordot tudunk számolni a klubnak.

Ezekkel a módszerekkel lekért adatokat a programunk kronológiai sorrendben fogja megjeleníteni a felhasználó számára a megadott csapat vagy játékost megjelenítő oldalon a “Seasons” címke alatt listában. Ezeket az adatokat fogja felhasználni a programunk statisztikai teljesítmények metrikai felméréséhez a harmadik menüpontunkban.

The screenshot shows the Atlanta Hawks team stats page. At the top, it displays the team name "Atlanta Hawks". Below that, there's a section titled "Seasonal Stats:" with a table showing performance data for three seasons: 2017, 2018, and 2019. The table includes columns for Year, Wins, Losses, PPG, and APPG. Underneath the table, there's a "Players:" section listing two players: John Collins (NA) and Trae Young (Guard). At the bottom of the page are four buttons: "NEW PLAYER" (white), "EDIT TEAM" (white), "IMPORT SEASON" (white), and "DELETE TEAM" (red).

Year	Wins	Losses	PPG	APPG
2017	24	58	103.4	108.8
2018	29	53	113.3	119.4
2019	20	47	111.8	119.7

Players:

John Collins
NA

Trae Young
Guard

NEW PLAYER EDIT TEAM IMPORT SEASON DELETE TEAM

6.6. ábra. Importált csapat felülete

6.3. Harmadik menü: Felmérések és Metrikák

A harmadik, egyben utolsó fő menüpontja az alkalmazásunknak a csapatokat és játékosokat összemérettető oldal, ahol az előbbiekben importált adatok segítségével komplex

Trae Young							
Seasons:							
Year	Points	Rebounds	Assists	Steals	Blocks	FG Made	Threes Made
2019	28.68	4.11	9.03	1.05	0.13	8.81	3.31
2018	18.89	3.67	7.96	0.87	0.18	6.4	1.9

IMPORT SEASON
EDIT PLAYER
DELETE PLAYER

6.7. ábra. Importált játékos felülete

metrikákat és kimutatásokat hozunk létre ahhoz, hogy meg tudjuk határozni a két kiválasztott fél közül, hogy melyik a jobb.

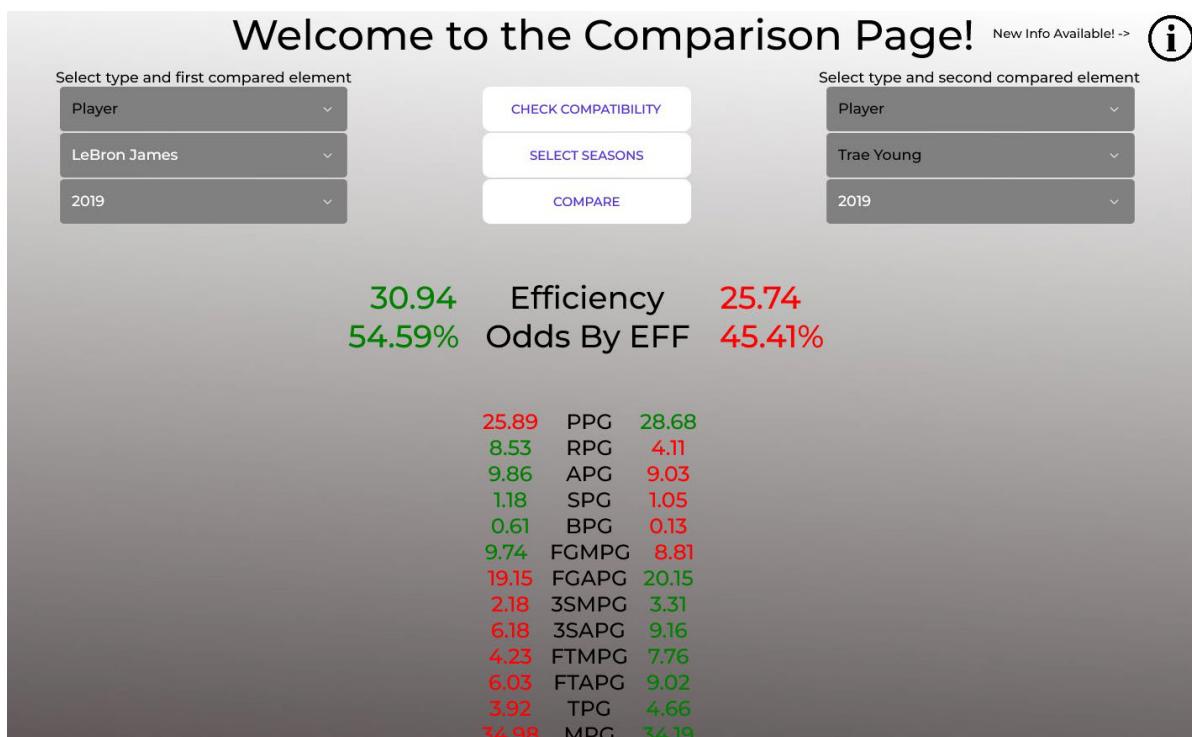
6.3.1. Felület és Használat Leírása

Az oldalon a felhasználó kiválasztásos módszerrel fogja meghatározni a két összehasonlítandó felet, avagy kiválaszt bizonyos kritériumokat listák segítségével, amivel megadja melyik feleket és melyik szezonokat fogja megjeleníteni. Először a user kiválasztja, hogy játékosokat vagy csapatokat szeretne összehasonlítani, azután meg választása szerint egy meghatározott listából kiválasztja a két sportolót vagy klubbot, utána meg a kívánt szezonjukat, ahonnan szeretnék lekérni az adatokat. Abban az esetben, ha a felhasználó egy csapatot és egy játékost választott ki, akkor a program kiadja, hogy invalid adatokat adott meg és nem fogja a két felet összehasonlítani, valamint ha bármelyiknek is hiányoznának statisztikai adatok, a usernek megadódik a lehetőség hogy egyből az importáló oldalhoz kerüljön. Ha a user megfelelő kritériumokat választott meg a két félhez, akkor a program sikeresen el fogja végezni az összehasonlításukat. Lekéri a csapatok vagy játékosok kért szezonbeli teljesítményüket és megjeleníti azokat, valamint fejlett statisztikákat és metrikákat számol belőlük. Játékosokat vonatkozóan a program a megadott szezonbeli csapataikat is összehasonlítja lekért szezonos statisztikák szerint. Ha a két játékos kiválasztott szezonja megegyezik, akkor megkapjuk csapatjaik egymással játszott mérkőzéseit is, ahol meg lesznek jelenítve a 2 fél végeredménye, valamint a dátum amikor lezajlott a megadott meccs.

6.3.2. Alapvető Statisztikák Bemutatása

Miután a programunk feldolgozta a két fél adatait és statisztikáit, azokat megjeleníti és fejlett statisztikákat számol belőlük. Ezeket a komplex kimutatásokat használhatjuk feleink egymáshoz viszonyított hatékonyságuk felmérésehez, pontosabban meghatározzatjuk összemérettetésükön nyert sikerekből, hogy a két sportoló közül melyik inkább a favorizált.

A legalapvetőbb statisztikák, amiket a szoftverünk itt megjelenít azok a lekért nyers teljesítményekből nyert átlagok, amiket a megadott szezonnal együtt mentettünk le az adatbázisba. A lekért adatok alapvető kosárlabda teljesítmények, amelyek a statisztikák alapját képzik, amik a mi alkalmazásunkban átlagolva vannak mérkőzésekre. Ezek a statisztikák a szerzett pontok átlaga (PPG/Points Per Game), lepattanók, avagy sikertelen



6.8. ábra. Felmérések és Metrikák felülete

dobás által a palánkról lepattant labda megszerzések számának átlaga (RPG/Rebounds Per Game), gólpasszok átlaga (APG/Assists Per Game), labdaszerzések, avagy ellenfelektől megszerzett labdák átlaga (SPG/Steals Per Game), blokkok, avagy kivédett dobások átlaga (BPG/Blocks Per Game), az összes lövési próbálkozás, valamint sikeres próbálkozás átlaga (FGMPG & FGAPG/Field Goals Made & Field Goals Attempted Per Game), a hárompontos próbálkozások és sikeres próbálkozások átlaga (3SMPG & 3SAPG/Threes Made & Threes Attempted Per Game), és más hasonló statisztika.

Ezen alap statisztikákhöz rangsoroljuk a csapat szintjén lévő adatokat is, ahol csak a szezonbeli átlagokat kérjük le a két játékos csapatairól. Kimutatásunkban megjelenítjük a két csapat nevét, a győzelmek és vereségek számát (Wins & Losses), a nyerési átlagukat százalékban (Win Percentage), valamint az importálások alatt számolt szerzett pontok átlagát és az ellenfél által szerzett pontok átlagát (Points Per Game & Away Points Per Game). Megegyező évszám esetében a “Teams’ Matchup” alatt meg fognak jelenni a megadott szezonban a két csapat egymással vívott mérkőzések végeredményét és lejátszási dátumát.

6.3.3. Komplex Statisztikák Bemutatása

A fentiekben felsorolt adatok segítségével képesek leszünk több fejlettebb metrikákat is létrehozni. Ezen újonnan létrehozott információk által nem csak egy komplexebb és mutatósbabb kimutatást fogunk alkotni, hanem a felhasználó számára is olyan adatokat adunk meg, amelyek segítségével tisztábban és pontosabban meg fogja tudni határozni, hogy a két sportoló közül valóban melyik a jobb.

25.89	PPG	28.68	Lakers	Team	Hawks
8.53	RPG	4.11	68	Wins	20
9.86	APG	9.03	24	Losses	47
1.18	SPG	1.05	73%	Win %	29%
0.61	BPG	0.13	113.3	PPG	111.8
2.18	FGMPG	3.31	107.3	APPG	119.7
6.18	FGAPG	9.16	46.31	Pythagorean	33.28
2.18	3SMPG	3.31			Wins
6.18	3SAPG	9.16			
Teams' Matchups					
122	11/17/2019	101			
101	12/15/2019	96			

6.9. ábra. Alapvető statisztikákról kimutatott képek játékosoknál és csapatoknál

A kimutatott alapvető teljesítmények mellett más alap adatokat is lekértünk az adatbázisunkba ahhoz, hogy ki tudjuk számolni a megszabott komplex metrikákat. Ezen adatokhoz tartoznak az összes szabad-dobások száma és a sikeres szabad-dobások száma (ft_made & ft_attempted), és az eladott labdák száma (turnovers), avagy, hogy hányszor vesztette el a játékos a labdát az ellenfeléhez. Hasonlóan a többi statisztikával, ezeket is az összes személyes teljesítményt képviselő mérkőzéshez rögzítettük, valamint ezekből is átlagot számoltunk és hozzárendeltük a szezonos statisztikához.

Így az összes lekért adatot felhasználva komplexebb adatokat is képesek leszünk meg-határozni és megjeleníteni, amik egy pontosabb képet fognak adni feleink mérettetéséből nyert eredményről. Ezen fejlett adatok kiszámításához relatív összetett képleteket fogunk alkalmazni, amiket a Bleacher Reports [Rep13], az NBAStuffer [NBA] és a Basketball Reference [Refa] fejlett statisztikáinak szójegyzékéből szereztem.

A komplexebb adatok esetében azokat vettet figyelembe, amiket az API-ból összes lehetőleg lekérhető statisztikák segítségével meg lehet határozni, így nem leszünk képesek minden lehetséges fejlett statisztikát bevezetni a kimutatásunkba.

A fontosabb komplex adataink közé tartozik a valódi dobási arány (True Shooting Percentage), ami által egy pontosabb lövési arányt határozhatunk meg játékosainknak. Legtöbb esetben dobások pontosságára a Field Goal Percentage-t, avagy az összes dobások arányát szokták figyelembe venni, viszont az egyedüli nélkülözhetetlen gondja ennek használatában az az a tény, hogy persze a FG% figyelembe veszi a játszmában tett összes dobást, ami a kosárlabda pályán zajló játék alatt történik, legyen az sima kétpontos dobás vagy akár egy hárompontos dobás, viszont ez a statisztika nem veszi figyelembe a szabad-dobások arányát. Ehhez egy szembetűnő példa lenne Andre Iguodala és James Harden NBA játékosok esete. A 2012-es NBA szezonban Iguodala 45.1%-os field-goal aránnyal rendelkezett, míg Harden csak 43.8%-al. Szabad szemmel figyelve így bárki meghatározhatja, hogy akkor Iguodala jobban dobott, mint Harden ebben a szezonban. Ennek ellenére viszont Harden 25.9 pontot átlagolt, míg Iguodala csak 13 pontot. Ha megfigyeljük a többi dobási statisztikát is, akkor láthatjuk, hogy Harden 36.8% hárompontos aránnyal és 85.1% szabad-dobás aránnyal rendelkezett, míg Iggy csak 31.7% hárompontos és egy nagyon alacsony 57.4% szabad-dobás aránnyal. Így a

kiszámolt valódi dobási aránya a két játékosnak 59.8% és 52% volt, ami azt mutatja meg, hogy James Harden valóban sokkal pontosabb volt dobásaiban, mint Andre Iguodala [Refb][Refc]. A valódi lővési aránynak a kiszámítási képlete a következő: $0.5^*(\text{összesen szerzett pontok}) / [(\text{összes dobások száma}) + 0.44^*(\text{összes szabad-dobások száma})]$.

Név	FG Percentage	FT Rate	3s Shooting %	Points/Game	TS%
Harden	.438	.851	.368	25.9	.598
Iguodala	.451	.574	.317	13	.520

6.1. táblázat. Harden és Iguodala 2012-2013-as szezonbeli statisztikai Basketball Reference-n

Egy másik komplex adat az a Free Throw Rate, ami megszabja a szabad dobások sikérének arányát, egy elégé szembetűnő és könnyen megérthető statisztika, amit szintén figyelemben kell tartani. Amint múltak az évek, az NBA játékosok egyre gyakrabban lőttek merészebb dobásokat, ez azzal magyarázható, hogy az átlag lővési távolság a ligában egyre jobban kiterjedt a hárompontos vonal irányába, hisz olyan élsztárok mint Steph Curry vagy Damian Lillard olyan távoli dobásokat kezdenek minél gyakrabban eltalálni, amiket több évtizeddel ezelőtt álmodni sem mertek. Ezen változás behozott egy új standardot a mai napi kosárlabdában, ami más trükkök és cselek szülemlényével járt. Az egyik ilyen trükk a szabad dobások kierötetése a játékvezetőből, hisz olyan játékosok, mint Trae Young vagy a fentebb említett Harden is néhány évvel ezelőtt rendesen kitervezett technikával közelítették meg védőjüket ahhoz, hogy majdnem garantáltan foul, avagy személyi hibát követelt a védőtől, így két szabad dobásból szerezhettek 2 extra pontot. Ennek függvényében következtethetünk arra, hogy a szabad dobások ugyanolyan fontosak lehetnek, mint a hárompontos dobások a mai napi NBA-ben. A szabad dobások aránya simán csak a sikeres és az összes szabad dobás arányából nyerhető ki.

Csapatok esetében is bevezettünk két jellegzetesebb fejlettebb statisztikát, ami által a felhasználó könnyebben megkaphatja a két kiválasztott klub közül, hogy melyik lenne a nyertes, ha a másikkal játszana egy mérkőzést. Az egyik az a minden kosárlabda és akár bármilyen csapatossport rajongó számára ismert nyerési arány, amit százalékban határoznak meg. Ehhez a statisztikához csak a megadott csapat győzelmeinek és vereségeinek száma szükséges, a százalék meg a nyerések aránya a veszteségekhez adja meg. Ezen statisztikát alkalmazzák legtöbb esetben a csapatok sorrendbe tevésében, hisz azon csapatok vezetik saját divíziójukat, amelyeknek jobb a rekordja a megadott szezonban, a rekordot meg legjobban a nyerési arány tudja megadni. A másik fejlett statisztika, ami bevezettünk az applikációnkba az a kevésbé ismert pitagoraszi győzelmek száma (Pythagorean Wins). Ezt a statisztikát sokkal ritkábban alkalmazzák, mint a nyerések arányát, viszont szerintem ez az adat sokkal pontosabban meg tudja határozni, hogy 2 csapat közül valóban melyik győzne. A pitagoraszi győzelmek száma nem csak azt veszi figyelembe, hogy hányszor győzött vagy vesztett a megadott kosárlabda válogatott, hanem megszerzett pontok mennyiségét is beleszámítja a metrikába. Így több adatréteget beintegrálhatunk a metrikáinkba, ami a kimutatás hitelességét segíti elő, avagy egy sokkal hihetőbb és megbízhatóbb adatot figyelhetünk meg, ami a csapatok valódi sikerét illeti. Ezek mellett ez a

metrika nem csak a saját csapat teljesítményét, hanem az ellenfél csapat teljesítményét is alkalmazza, így egy valamennyire összekötött metrikával dolgozhatunk, amivel minden két csapat produktivitását használja a két fél sikerének eldöntéséhez, nem csak sajátos statisztikák által. A pitagoraszi győzelmek számának a képlete: (első csapat mérkőzéseinek száma) * (első csapat szerzett pontjainak száma / (mindkét csapat összesen szerzett pontjainak száma)). Az általunk bevezetett két csapat-szintű fejlett statisztika által így egy tisztább képet kínálhatunk a user számára ahhoz, hogy meghatározhassa a két általa megszabott csapat közül melyik lenne a legnagyobb eséllyel a győztes.

Az utolsó, egyben programunknak legfontosabb komplex statisztikája, ami leginkább meg tudja határozni egy játékos hasznát és sikerét a kosárlabda pályán, az a hasznossági metrika (Efficiency). Ez a metrika egyenesen és lényegretörően elmondja számunkra, hogy a megadott játékos valóban mennyi hasznot vagy sikert hoz saját csapatának, mikor nekik játszik. Sok rajongó számára ez a metrika is kulcsfontosságú, mivel majdnem az összes alapvető kosárlabda-teljesítményt figyelembe veszi, így nagyjából minden lehetséges területet analizálni tud statisztikailag. A legtöbb esetben a hasznossági arányt inkább a támadást leíró, "offense" statisztikák befolyásolják, avagy szerzett pontok, gólpásszok, reboundok meg hasonlók, viszont tartalmaz védekező, "defense" statisztikákat is, mint a stealek és a blokkok. Úgy, mint a valódi lövési aránynál, ez a metrika tud különbséget tenni egyes alap statisztikák által kimutatott furább eredmények között, legyen az bizonyos játékosok nagyon magas átlagolt pontszáma, attól függetlenül, hogy minden más metrikában nem a legjobb. Amint a neve is mutatja, ez az arány nem azt mutatja számunkra, hogy melyik játékos jobb megadott területeken a mérkőzések alatt, hanem hogy melyiket érdemesebb inkább engedni játszani a csapatnak, avagy melyik játékosból nyerne ki több hasznot a csapat. Sok edző is ezt a metrikát figyeli ahhoz, hogy meghatározza melyik játékosnak adjon több percet a mérkőzéseken másokkal szemben. Ezt az arányt alkalmazva kaphatunk olyan játékosokat is, akik nem a legkitűnőbbek a ligában, vagy akár saját csapatukban sem, viszont kitűnően teljesített ahhoz mérve, hogy mennyit játszott vagy próbálkozott meccsek során. Ilyen volt Clint Capela is a tavaly, az Atlanta Hawks fő "nagyembere", aki tényleg sokkal kevesebbet játszott, mint a Hawks sztárjátékosa Trae Young, viszont lehet látni magas hasznossági arányából, hogy az időt, amit a kosárlabda pályán töltötte, jobban kihasználta, mivel PER, avagy hasznossági arány szerint jobban végezte a 2023-as szezont (22.25 és 22.15). Innen lehet látni, hogy a játékosokat nem csak alapvető teljesítmények alapján kéne ítélnünk, hanem a próbálkozások ideje és száma alapján is. Ennek a statisztikának függvényében is számoltam egy százalékos kimutatást a 2 sportoló hasznossági arányából, ami alapján a felhasználó meghatározhatja alkalmazásunkon belül, hogy melyik játékos inkább a kedveltebb statisztikailag [ESP]. A hasznossági arányt a következőképpen lehet kiszámolni: (pontok száma + lepattanók + gólpásszok + stealek + blokkok) - ((összes dobási próbálkozások száma - összes sikeres próbálkozások száma) + (összes szabad dobások száma - összes sikeres szabad dobások száma) + eladott labdák száma).

A fenti táblázatban látható a 21 és 25 közötti rangsorral rendelkező játékosokat, ahol a GP a lejátszott meccsek számát, az MPG az mérkőzések átlagban lejátszott percek számát, a TS% a valódi dobási arány, APG a meccsenkénti gólpásszok aránya, a TO a meccsenkénti eladott labdák aránya, valamint a PER, ami a játékos hasznossági aránya.

Ezen alapvető és fejlett statisztikák segítségével fog tudni a felhasználó játékosokat és csapatokat felmérni metrikai szempontból. Kimutatásunkban megjelenítünk több faj-

Rang	Név	GP	MPG	TS%	APG	TO	PER
21	Lauri Markannen	66	34.4	.641	7.8	8.1	22.32
22	Clint Capela	65	26.6	.656	8.2	7.7	22.25
23	Devin Booker	53	34.6	.601	17.6	8.7	22.21
24	Trae Young	73	34.8	.573	27.3	11.1	22.15
25	De'Aaron Fox	73	33.4	.599	20.8	8.4	21.98

6.2. táblázat. Hollinger 2022-23 NBA rangsorolása PER szerint

ta adatot is, valamint a metrikák formája is különbözik, legyenek azok színekkel ellátott alapvető szövegbeli kimutatások, listák vagy grafikonok. A játékosoknál alkalmazott fejlett statisztikákat implementáltuk a csapatoknál is, olyan módon, hogy a megadott csapat szezonbeli játékosainak statisztikáit egybe számoltuk, és átlagot számoltunk belőlük, így a csapat összes importált játékosainak statisztikáiból is fogunk tudni metrikákat létrehozni csapatok szintjén. A kimutatásban szereplő gráfok is fejlett statisztikákat mutatnak ki, pontosabban az előbbiekbén említett valódi lövési arányt és a most megszabott hasznossági arányt is. A gráf pontosabban mérkőzéseként felvázolja minden játékos hasznossági arányát, ahonnan lefigyelhető, hogy mennyire változott meg a két sportoló teljesítménye a szezon során.

7. fejezet

Felmérések Tesztelése és Kimutatása

Megszabott és véghezvitt metrika számoló oldalaink és funkcióink sikeres létrehozása által létre tudtunk hozni egy olyan platformot felhasználóink számára, ahol bármilyen megszabott játékost vagy csapatot összehasonlíthat egy másikkal. Ez a funkció működik bármilyen, az API-ban részt vevő NBA félre, játékosok esetén csapat vagy teljesítmény szintjétől függetlenül. A továbbiakban össze fogunk hasonlítani 2 játékost az alkalmazásunk segítségével. Az összehasonlításban két hasonló szintű játékost fogunk felnérni, a Lakers és Mavericks sztárjátékosait, azaz LeBron James és Luka Doncic.

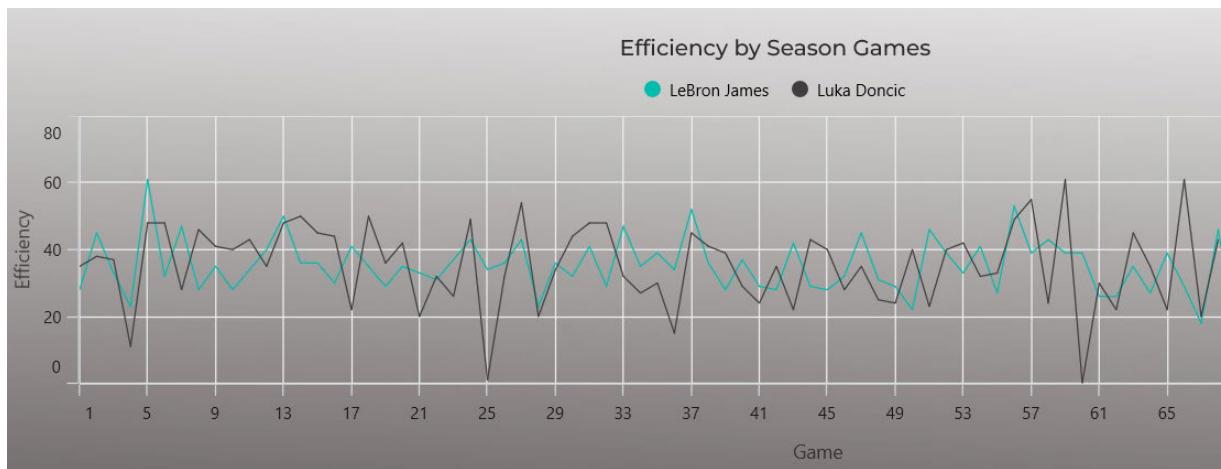
Első sorban importálni fogjuk csapatainkat, ahol játszanak a kért játékosaink, valamint a játékosokat is le fogjuk kérni az API-ból. Amiután belehelyeztük adatbázisunkba a felek adatait, importáljuk azon megadott adatait egy megadott szezonra. Ehhez a ki-mutatáshoz a 2019-es szezont választottam, ami azt jelenti, hogy ha szeretném, hogy az alkalmazás a játékosok csapatait is felületesen felnérje csapat-szintű statisztikák segítségével, akkor nem csak a játékosok, hanem a csapatok 2019-es szezonjait is importálnom kéne. Amiután befejeztük az importokat, akkor a felnérési oldalra térhetünk, ahol kiválasztjuk feleinket és szezonjainkat, amiket fel szeretnénk mérni.

A kimutatás során következtethetünk, hogy LeBron James és Luka Doncic valóban hasonló szinten játszott a 2019-es szezonban. Ezt a tényt nem csak az alapvető statisztikák, hanem a fejlett adatok is kimutatják, mivel az alábbi táblázatban is látható, hogy a hasznossági arány, a metrika, ami szerint elhatározzuk programunkban, hogy melyik játékos győzne a másikkal szemben, többé-kevésbé megegyezik.

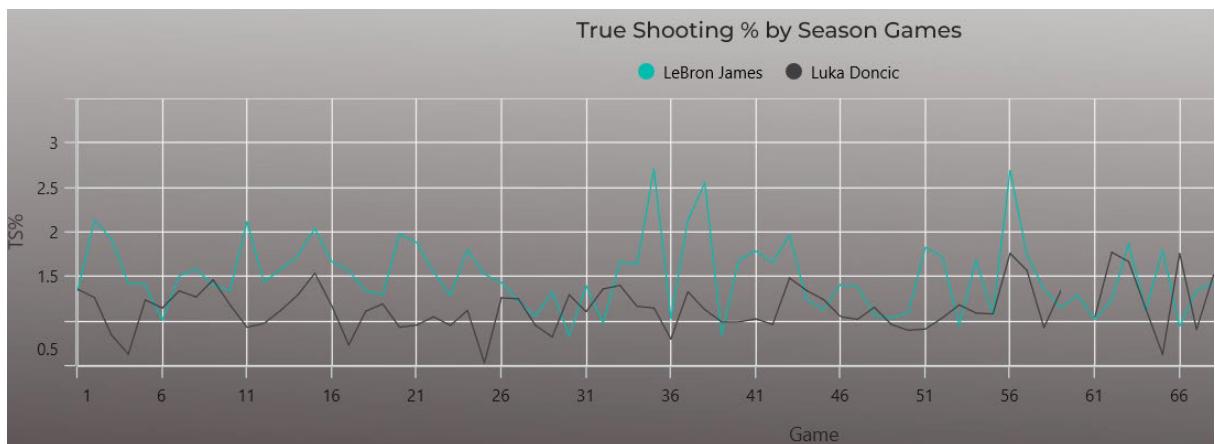
Statisztika Neve	Lebron James	Luka Doncic
Hasznossági arány	30.94	30.37
Hasznossági arány százaléka	.504	.495
Pontok átlaga	25.89	28.6
Reboundok átlaga	8.53	9.29
Assists átlaga	8.53	9.29
Stealek átlaga	8.53	9.29
Blokkok átlaga	8.53	9.29
Összes sikeres dobások száma	9.74	9.49
Összes dobások száma	19.15	20.34
Sikeres hárompontos dobások száma	2.18	2.75
Összes hárompontos dobások száma	6.18	8.6
Valódi dobási arány	.59	.59
Szabad dobások aránya	0.31	0.45

7.1. táblázat. LeBron James és Luka Doncic programunk által kimutatott statisztikái a 2019-es szezonban

Ezek a kimutatások a gráfunkban is alá vannak támasztva, ahol a szezon alatt LeBron és Luka egymást váltogatták hasznossági arányban. Ennek függvényében viszont meglát-szik LeBron tapasztalata és megfontoltsága az ifjonccal szemben, hisz a valódi lövések aránya inkább a Lakers játékosát kedvezi. Ebből azt vonhatjuk le, hogy az meglehet, hogy minden játékos hasonlóan kiválóan teljesít, viszont, ha a lövések pontosságát figyeljük, akkor James jobban teljesít. Ez ahhoz vezethet inkább, hogy a 2 játékos közül melyiket választanánk ahhoz, hogy egy utolsó kiegyenlítő vagy mérkőzést nyerő dobást próbáljon a csapat számára.



7.1. ábra. LeBron James és Luka Doncic hasznossági aránya van a grafikonon ábrázolva



7.2. ábra. LeBron James és Luka Doncic valódi lövési aránya van a grafikonon ábrázolva

Ezen kimutatások szerint akkor arra következtethetünk, hogy LeBron James és Luka Doncic hasonló szinten produkált, viszont a pontosabb dobások által LeBron hasznossági aránya egy kicsivel jobb lett, mint Lukának, ezért a programunk kimutatta a felhasználó számára, hogy 50.46% és 49.54%-os arányban mégis LeBron a kedveltebb.

8. fejezet

További Fejleszthetőségek

Alkalmazásunk készítésének folyamata alatt sok általunk kiszabott célokat és terveket végrehajtottunk sikeresen, ezek közül a legalapvetőbb feladatainkat, amik az alkalmazásunk integritását teljes mértékben megadják. Ez leginkább azt jelképezi, hogy minden funkciót, ami a programunknak kulcsfontosságú és nélkülözhetetlen egy bajnokság készítő és importáló alkalmazás számára, gondok nélkül elvégzi a felhasználó számára. Szembetűnő sikerünk mellett viszont arra is kell gondolnunk, hogy bármilyen szoftver tervezéséről és létrehozásáról van szó, mindig létezik valamilyen formában legalább egy funkció vagy fejlesztés, amit hozzá lehet rendelni a programunkhoz. Ennek a ténynek függvényében természetes, hogy a mi alkalmazásunk is tartalmazhat több opciót vagy szolgáltatást, mint amiket beiktattunk.

Számomra a legfontosabb és legszembeíróbb fejlesztés, amit hozzá tudnék adni az alkalmazásomhoz, az több és fejlettebb komplex statisztikák. Léteznek olyan kifejlesztett összetett adatok, amik leírják egy sportoló vagy csapat teljesítményét, amelyek olyan faktorokat is beleszámít kimutatási képletében, amik vagy nem elérhető mindenki számára, vagy egyszerűen nem követhető számon hagyományos adatokat szerző és karbantartó módszerekkel. Az egyik ilyen statisztika maga a támadási és védekezési arány (offensive & defensive rating), ami által megtudhatjuk pontosabban, hogy melyik játékosnak a támadási vagy a védekezési képességei a jobbak. A probléma számunkra ennek a statisztikának a kiszámításához az a csapatszintű statisztikák és más specifikus adatok alkalmazása, amikkel az általunk használt API nem rendelkezik, ezért képtelenek vagyunk meghatározni ezen teljesítményt játékosaink számára. Egy másik statisztika lenne a plusz/minusz (Adjusted Plus/Minus), ami egy eléggé elterjedt teljesítmény-kimutatási adat, amely pontosabban meghatározza, hogy egy adott játékos mennyire tudja segíteni csapatát mikor épp jelenleg játszik. Ez nagyon hasonlít az általunk kiszámolt hasznossági arányhoz, az lévén a legfőbb különbség, hogy míg a hasznossági arány a teljes mérkőzés alatt szerzett teljesítmények segítségével számolja ki egy sportoló hasznát, a plusz/minusz a jelenlegi helyzetet veszi figyelembe, így ez a statisztika percenként változik annak függvényében, hogy a játékos mennyire befolyásolja a csapatok pontszerzését. Ennek az adatnak kiszámításához az a legnagyobb gondunk, hogy nagyon belejátszik minden játékos percenkénti befolyása a pályán, vagyis figyelembe kell venni azt is, hogy egy mérkőzés 48 perce alatt pont melyik minutumokban vesz részt az adott játékos, egy adat, amit nagyon nehéz megszerezni egy bármilyen általunk meghatározott mérkőzésről.

Adatbányászati és metrikák létrehozási funkcióinkat figyelmen kívül hagyva szerintem nélkülözhetetlen a ligakészítő részleg további fejlesztése. Persze gond nélkül létre tudunk hozni egy bajnokságot sajátos csapataink segítségével, ahol mérkőzésekbe tehetjük őket és játékosait, statisztikákkal tudjuk ellátni őket és ezen adatok által rangsorolhatjuk is őket, viszont léteznek még egyes NBA ligájából szerezhető opciók és lehetőségek, amik jól fognának ehhez a rendszerünkhez. A csapatokat beoszthatnánk külön konferenciákba, létre hozhatnánk egy teljesen eltérő, kiütéses kupát, ami a bajnokságunkon belül lenne és azt követné, az NBA Playoffs-hoz hasonlóan, valamint azon belül csapatok rekordja szerint automatikusan beoszthatjuk őket ebbe a kupába, valamint még számos más vizuális és praktikai funkciót is bevezethetnénk.

A fentiekben megszabott változtatások nélkül is kiviteleztük terveinket és elértük akart eredményeinket, hogy egy pontos és megalapozott alkalmazást hozzunk létre, ahol egy user könnyedén tud adatokat importálni és feldolgozni egy megadott kosárlabda játékos vagy csapat teljesítményeiről, valamint sajátos bajnokságokat is létre tud hozni és karbantartani. Ezen változtatások bevezetése által viszont egy olyan új szintre emelhetnénk szoftverünket, ami által egy sokkal tetszetősebb és komplexebb alkalmazássá válik, ami mérkőzések létrehozásának vagy felmérésének funkciói által olyan lehetőségeket kínálunk felhasználóink számára, amelyekre nem is gondoltak volna, hogy hasznos lenne szervezéseik vagy kutatásaink során.

9. fejezet

Konklúzió

Importálási és adatbányászati kalandjaink során sok fajta statisztikai jellegeket és megközelítéseket találtunk. Fontos kiemelnünk, hogy amint láthattuk nem elég a nyers adatokat nézni és azok alapján következtetni arra, hogy melyik mérkőzést ki fogja megnyerni vagy elveszíteni, hanem azok alapján következtetnünk kell más megközelítési lehetőségekre is. A sportokban elért teljesítményeink rengeteg faktorra bonthatóak fel, amik minden beleszámíthatnak bizonyos tereken és vagy akár célok elérésében.

Ennek függvényében segített számunkra ez az alkalmazás készítése, hisz kaphattunk, elemezhettünk és még bizonyos szinten el is sajátíthattunk bizonyos metrikai és statisztikai kimutatási módszereket, amik több dimenziókat adnak az adatelemzés világában. Innen következtethetünk arra, hogy a sportrajongók összessége két jellegzetes részre osztható: azon személyek, akik minden metrikát saját kezűleg megkeresnek, összegyűjtenek és felnérnek mérkőzések kimenetelének megjósolásának érdekében, és a minden nap sport-élvező, aki nem fogja tudni ezt a sok statisztikát megkeresni vagy akár megérteni. Ennek érdekében sikerült egy olyan alkalmazást létrehozni, ami minden nap megfelel. A statisztikai kockafejek számára megalkotottunk egy felületet, ahol egy helyen megtalálhatnak lényegretörő, és egyben nélkülözhetetlen statisztikákat, amiket felhasználhatnak játékosok felméréséhez. A minden nap rajongó számára pedig egy olyan programot hoztunk létre, ahol kíváncsiságból utána nézhet két csapatnak vagy játékosnak az alkalmazásba beépített kisegítő gombok segítségével, és gyorsan, de pontosan megkaphatja, hogy valóban melyik nyerne egy mérkőzésben, anélkül, hogy bármilyen komplikált statisztikai képletek vagy tudás birtokában legyen.

Befejezésképpen arra a tényre térünk, hogy egy ilyen adatokat feldolgozó alkalmazás, ami egy sajátlag létrehozott ligát is képes fenntartani, a sport-rajongók gondolatmenetét előggé pontosan ábrázolja. A liga készítő alkalmazásunk által képesek leszünk kisded álmainkat létrehozni vagy egy haveri körben játszani egy olyan bajnokságot, ami teljes kezűleg testreszabható a mi vágyaink alapján. A metrikákat létrehozó funkcióink pedig egy olyan megközelítést adnak a sport világának szociális területére, ami által bárki egy gyors és megbízható módszerrel rájöhet egy válaszra haveri körében felhozott dilémáikra, abban, hogy egy megadott évben ez a játékos valóban legyőzte volna azt a játékost mai napjainkban.

Az alábbi linken található a projektem GitHub repositoryja: <https://github.com/majamanmajor3/MyLeague.git>

Irodalomjegyzék

- [AJ18] D. Ashby and C. T. Jensen. *APIs For Dummies, Chapter 4 - Implementing Your API Program*. John Wiley & Sons, Inc., 3rd ibm limited edition edition, 2018.
- [Bas15] L. Bassett. *Introduction to JavaScript Object Notation: A To-the-Point Guide to JSON, Chapters 1 & 2*. O'Reilly Media, 2015.
- [BD13] M. Balliauw and X. Decoster. *Pro NuGet, Consuming and Managing Packages in a Solution*. Apress, Berkeley, CA, 2013.
- [CH14] M. Chand and S. Hobbs. *Programming XAML Beginners Guide*. C# Corner, 2014.
- [ESP] ESPN. Hollinger's nba player stats. URL: <https://insider.espn.com/nba/hollinger/statistics>.
- [Har15] B. Harwani. *Foundations of Joomla, Installing XAMPP and Joomla*. Apress, Berkeley, CA, 2015.
- [Hoo] Hoopsbeast. How many players are in the nba? URL: www.hoopsbeast.com/total-nba-players/.
- [Kof05] M. Kofler. *The Definitive Guide to MySQL 5, phpMyAdmin*. Apress, Berkeley, CA, 2005.
- [Law23] S. Lawrence. *The Fundamentals of .NET MAUI*. Apress, Berkeley, CA, 2023.
- [Mic] Microsoft. .net, community toolkits. URL: <https://learn.microsoft.com/en-us/dotnet/communitytoolkit/mvvm/>.
- [Mil19] R. Miles. *C# Programming Yellow Book*. 2019.
- [MyS] MySQL. Entity framework. URL: <https://dev.mysql.com/doc/connector-net/en/connector-net-entityframework-core.html>.
- [NBA] NBAStuffer. True shooting percentage (ts%) explained. URL: <https://www.nbastuffer.com/analytics101>true-shooting-percentage/>.
- [New] Newtonsoft. Json. URL: <https://www.newtonsoft.com/json>.
- [Refa] Basketball Reference. Glossary. URL: <https://www.basketball-reference.com/about/glossary.html>.

- [Refb] Basketball Reference. Harden stats. URL: <https://www.basketball-reference.com/players/h/hardeja01.html>.
- [Refc] Basketball Reference. Iguodala stats. URL: <https://www.basketball-reference.com/players/i/iguodan01.html>.
- [Rep13] Bleacher Report. Advanced nba stats for dummies: How to understand the new hoops math. 2013. URL: <https://bleacherreport.com/articles/1813902-advanced-nba-stats-for-dummies-how-to-understand-the-new-hoops-math>.
- [Str22] D. Strauss. *Getting Started with Visual Studio 2022, Working with Visual Studio 2022*. Apress, Berkeley, CA, 2022.
- [Syn] Syncfusion. How to create a line chart. URL: <https://support.syncfusion.com/kb/article/11273/how-to-create-a-line-chart-in-net-maui>.
- [Wan18] K.C. Wang. *Systems Programming in Unix/Linux, MySQL Database System*. Springer, Cham, 2018.
- [Wei16] A. Weil. *Learn WPF MVVM - XAML, C# and the MVVM pattern, MVVM Pattern For WPF*. Lulu.com, 2016.