ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

NEUMANN JÁNOS
INFORMATIKAI KAR

SZAKDOLGOZAT

OE-NIK
2021

Horti Kristóf
T/006225/FI12904/N

Óbudai Egyetem
Neumann János Informatikai Kar
Kiberfizikai Rendszerek Intézet

SZAKDOLGOZAT FELADATLAP

Hallgató neve: **Horti Kristóf**
Törzskönyvi száma: T/006225/FI12904/N
Neptun kódja: [redacted]

A dolgozat címe:

BI rendszer kialakítása adattárház segítségével, az NHL-ből származó adatok elemzése és vizualizációja
BI system development with the help of a data warehouse from NHL data

Intézményi konzulens: Sarkadi Katalin
Külső konzulens:

Beadási határidő: 2021. december 15.

A záróvizsga tárgyai: Számítógép architektúrák
Big Data és üzleti intelligencia specializáció

A feladat

A szakdolgozat célja, az NHL-el (National Hockey League) kapcsolatos adatok adattárházban történő feldolgozása. A feladat az elérhető adatforrásokból az adatok átvitelének megtervezése és megvalósítása ETL réteg segítségével adattárházba. Feladat még továbbá az adatok betöltése egy megfelelő BI rendszerbe, melynek segítségével megvalósítható lesz az adatok vizualizációja. A hallgató feladata továbbá egy webalkalmazás fejlesztése.

A dolgozatnak tartalmaznia kell:

- az adatforrás réteg meghatározását, illetve a rendszerek és a kívánt adatok/táblák kiválasztását,
- ETL réteg részletes megtervezését és megalkotását,
- adattárház tervezését és megvalósítását,
- BI alkalmazás/portál megtervezését és megalkotását,
- adat vizualizáció megvalósítását,
- webalkalmazás megtervezését, megalkotását.

Ph.

ÓBUDAI EGYETEM Neumann János Informatikai Kar 1. Budapest

FC VR

.....
Dr. Fleiner Rita
intézetigazgató

A szakdolgozat elévülésének határideje: **2023. december 15.**
(OE TVSz 55.§ szerint)

A dolgozatot beadásra alkalmasnak tartom:

.....
külső konzulens

.....
intézményi konzulens

HALLGATÓI NYILATKOZAT

Alulírott hallgató kijelentem, hogy a szakdolgozat / diplomamunka saját munkám eredménye, a felhasznált szakirodalmat és eszközöket azonosíthatóan közöltem. Az elkészült szakdolgozatomban / diplomamunkámban található eredményeket az egyetem és a feladatot kiíró intézmény saját céljára térítés nélkül felhasználhatja.

Budapest, 2021.12.05

.....

hallgató aláírása

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Neumann János Informatikai Kar

KONZULTÁCIÓS NAPLÓ

Hallgató neve:	Neptun kód:	Tagozat:
Horti Kristóf.....	[redacted].....	**Nappali**.....

Telefon:	Levelezési cím (pl: lakcím):
[redacted].....	[redacted].....

Szakdolgozat / Diplomamunka[1] címe magyarul:

BI rendszer kialakítása adattárház segítségével, az NHL-ből származó adatok elemzése és vizualizációja

Szakdolgozat / Diplomamunka[2] címe angolul:

BI system development with the help of a data warehouse from NHL data

Intézményi konzulens: **Sarkadi Katalin**

Külső konzulens: -

Kérjük, hogy az adatokat nyomtatott nagybetükkel írja!

Alk.	Dátum	Tartalom	Aláírás
1.	2021.02.25	Téma egyeztetése	
2.	2021.03.17	Webalkalmazás kialakítása	
3.	2021.04.26	Beszámoló készítése	
4.	2021.05.10	Beszámoló javítása, vizualizációs eszközök	

A Konzultációs naplót összesen 4 alkalommal, az egyes konzultációk alkalmával kell láttamoztatni bármelyik konzulenssel.

A hallgató a Szakdolgozat I. / Szakdolgozat II. (BSc) vagy Diplomamunka 1 / Diplomamunka 2 / Diplomamunka 3 / Diplomamunka 4[3] tantárgy követelményét teljesítette, beszámolóra / védésre[4] bocsátható.

Budapest, 2021. 05. 14

intézményi konzulens

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Neumann János Informatikai Kar

KONZULTÁCIÓS NAPLÓ

Hallgató neve: Neptun kód: Tagozat:

Horti Kristóf [redacted] **Nappali**

Telefon: Levelezési cím (pl: lakcím):

[redacted] [redacted]

Szakdolgozat / Diplomamunka[1] címe magyarul:

BI rendszer kialakítása adattárház segítségével, az NHL-ből származó adatok elemzése és vizualizációja

Szakdolgozat / Diplomamunka[2] címe angolul:

BI system development with the help of a data warehouse from NHL data

Intézményi konzulens: Külső konzulens: -

Sarkadi Katalin

Kérjük, hogy az adatokat nyomtatott nagybetűkkel írja!

Alk.	Dátum	Tartalom	Aláírás
1.	2021.09.20	Webalkalmazás felépítése	
2.	2021.11.05	Szakdolgozat adminisztrációs kérdései	
3.	2021.11.24	Fejlesztés eredménye, dolgozat felépítése	
4.	2021.12.05	Szakdolgozat javítása, beadással kapcsolatos kérdések	

A Konzultációs naplót összesen 4 alkalommal, az egyes konzultációk alkalmával kell láttamoztatni bármelyik konzulenssel.

A hallgató a Szakdolgozat I. / Szakdolgozat II. (BSc) vagy Diplomamunka 1 / Diplomamunka 2 / Diplomamunka 3 / Diplomamunka 4[3] tantárgy követelményét teljesítette, beszámolóra / védésre[4] bocsátható.

Budapest, 2021. 12. 04.

intézményi konzulens

Tartalom

1. Bevezető

Világunk nagyon gyorsan fejlődik, új technológiák, új ötletek jelennek meg nap mint nap. Ebben a rohamos fejlődésben nagy szerepe van az informatikának. A hétköznapi életünk szerves részévé váltak az okostelefonok, laptopok, számítógépek, okosórák stb. Ennek a fejlődésnek köszönhető, hogy ma már könnyedén tudunk a telefonunk segítségével bevásárolni, banki átutalásokat bonyolítani. A modernizáció velejárója, hogy egyik pillanatról a másikra rengeteg adat képződik, amit valahogy el is kell tárolni, és fel kell dolgozni. Azonban nem elég, ha ezeket az adatokat valahogy el tudjuk tárolni, a cél az, hogy hasznos információkat tudjunk belőlük kinyerni, és olyan következtetéseket levonni, amelyek elősegítik döntéseinket.

A legelterjedtebb és leghatásosabb módszer jelenleg az Adattárház technológia. Az adattárház alapvetően egy olyan adatbázis, ahol az üzleti elvárásoknak megfelelően optimalizálva, elemzésre alkalmas formában találhatóak meg az adatok. Az üzleti szempontból hasznos tudás kinyerésére azonban szükség van egyéb eszközökre is, ezeket a rendszereket Üzleti Intelligencia (BI) rendszereknek nevezzük.

A szakdolgozat célja az adattárház és üzleti intelligencia, illetve különböző adatelemzési módszerek bemutatása elméletben és gyakorlatban egyaránt. Egy, az NHL (National Hockey League) adatait tartalmazó adatbázis adattárházba töltése és az ehhez kapcsolódó feladatok elvégzése, valamint ezekre az adatokra épülő vizualizációk és elemzések elkészítése. Végül egy webalkalmazás tervezése és fejlesztése, amin a felhasználó megtekintheti az előzőleg elkészített elemzéseket és különböző szűrési feltételek megadása mellett tudja megtekinteni a kívánt adatokat.

A téma választásában közrejátszott az is, hogy a technológiák fejlődésének ellenére a többi sporthoz képest a jégkorong statisztikák még mindig gyerekcipőben járnak, a jelenleg elérhető legjobb modellek is „csak” 62%-os pontossággal tudják megjósolni, hogy melyik csapat fog nyerni az adott mérkőzésen, többnyire a játékosok egyéni tehetségének különbségei és az úgy nevezett „puck luck” miatt, ami azokat az eseményeket takarja, amelyeket nem lehet előre látni, és nem függenek a csapat stratégiájától, például a jég minősége, korong furcsa szögben való pattanása, és minden egyéb, ami véletlenszerűen befolyásolja a pontszerzést. Érdekelt, hogy a rendelkezésre álló adatokból sikerül-e kinyerni bármilyen hasznos információt, esetleg következtetéseket levonni, vagy olyan szabályszerűséget megfogalmazni, ami befolyásolhatja a döntéseket a jövőben.

2. Adattárházak és üzleti intelligencia

2.1 Az üzleti intelligencia fogalma, története

Az üzleti intelligencia fogalmának meghatározása évekig Howard Dresner nevéhez fűződött, aki 1989-ben olyan módszerek összességének nevezte, melyek tényalapú rendszerek (MIS – Vezetői információs rendszer, OLAP – Online Analytical Processing, DM – Adatpiac, DSS – Döntéstámogató rendszer) segítségével javítják a döntéshozás folyamatát. Később, 2009-ben Claudia Imhoff javaslatára visszadatálták az üzleti intelligencia első definícióját Hans Peter Luhn 1958-ban írt cikkére, mely „A Business Intelligence System" címmel az IBM Journal októberi számában jelent meg. Luhn elképzelése egy olyan világ volt, ahol a különböző szervezetek, vállalatok, egyének egyre több adatot fognak termelni, és ezeket automatikusan működő, intelligens rendszerek fogják feldolgozni és információkat kinyerni belőlük, hogy segítsék az emberek, szervezetek tevékenységeit. Az 1958-as és 1989-es írások ellenére az üzleti intelligencia csak az 1990-es évek végén épült be az informatikai cégek szókincsébe. [5][7][8][10]

Az 1990-es évekre olyan hatalmas mennyiségű adat halmozódott fel a nagy cégeknél, amit már nem lehetett többé hatékonyan felhasználni átfogó stratégiai döntések meghozatalához a hagyományos adatbázis-kezelés keretein belül. A fő probléma az volt, hogy a tárolók sok évre visszamenőleg tartalmaztak adatokat, és a sok részlet nagyon megnehezítette a jövőre vonatkozó tervek létrehozását, valamint a TPS (Toyota termelési rendszer) rendszerekből származó adatok inkonzisztensek, eltérő formátumúak voltak, emiatt nehéz volt összegzett információt létrehozni. [3][2][5][7]

2.3 Diszparát adatok problémája

A vállalatok érdekeinek kiszolgálásához az adatoknak olyanoknak kell lenniük, hogy viszonylag könnyen, jó minőségű információt lehessen kinyerni belőlük. A nagyvállalatok régóta használnak adatbázisokat az adataik eltárolásához, azonban még mindig sok cég küzd az úgynevezett diszparát adatok problémájával, ami hátráltatja őket a döntéshozásban. Diszparát adatokról beszélünk, ha az adatállományban az adatok formátuma változó, jelentésük nem egyértelmű, pontosságuk nem ismert, időszerűségük nem megfelelő, az adatok különböző helyeken, különböző adatbázisokban, különböző struktúrával léteznek és ezeket nem lehet egymással összekapcsolni, redundáns vagy inkonzisztens. Diszparát adatforrásra példa a 2.1 ábra, ugyanis a Név mezőről nem egyértelműen eldönthető, hogy milyen sorrendben tartalmazza a vezeték- és keresztneveket, valamint az SZI mezőről sejteni lehet, hogy a születési időt tartalmazza, viszont nem lehet tudni, hogy milyen formátumban, vagyis melyik rész mit jelöl.

Név	SZI
Péter József	10/08/06
John Bill	05/11/04

2.1 ábra: Diszparát adatok [5]

A relációs adatbázisok elméletben lehetővé teszik, hogy integrált, egységes és redundanciamentes adatokat tároljunk benne, azonban tervezési hibák, az adatbázis szerkezetének változása és emberi hibák (üresen hagyott mezők, elírások), okozhatják ennek hiányát. Az adatbázisokat utólag tisztítani, kijavítani nagy munka, nagyobb, mintha kezdettől fogva egységesen írtuk volna bele az adatokat, azonban megoldható. A fő lépései a következők:

1. adatok azonosítása;
2. adatok megértése – metaadatok készítése;
3. adatok tisztítása
    - formátumok egységesítése,
    - hiányzó adatok kitöltése,
    - redundanciák megszüntetése;
4. adatok között a hiányzó kapcsolatok kiépítése;
5. adatok elemzése, rendezése;
6. adatok telepítése.

A diszparát adatok elsősorban nem a mindennapi munkát nehezítik meg, hiszen általában csak egy kisebb részterület adataival foglalkoznak a dolgozók, és feltehetőleg ismerik is ezeket az adatokat. Akkor okoz gondot, mikor az adatokat magasabb szintű döntéshozáshoz akarják felhasználni. [5]

2.4 Operatív működés és döntéshozás

Felhasználási terület szerint két része lehet osztani az adatbázisokat: operatív adatbázis és információs adatbázis.

Operatív adatbázisoknak hívjuk azokat az adatbázisokat, melyeket a napi teendők ellátására használunk. A mai korszerű rendszerek már azonnal feldolgozzák a tranzakciókat, ezért azt is mondhatjuk, hogy az operatív adatbázisokat online tranzakció-feldolgozásra használják (OLTP - Online Transaction Processing). A vállalatok raktárkészlet-nyilvántartása és a repülőgépes helyfoglaló rendszerek is ilyen adatbázisokat használnak.

Az operatív adatbázisok állandóan változnak, ennek köszönhetően mindig az aktuális adatokat tartalmazzák. Manapság egy jól felépített adatbázisban megtalálhatóak korábbi adatok az esetleges visszaállítás miatt, viszont a féléves, vagy akár több éves adatokat nem lehet elérni, csak esetleges külön eljárás keretében (Pl.: több évre visszamenőleg számlatörténet lekérése). Ez a mindennapi munkához általában elég is, azonban azoknak a döntéshozóknak, akik több éves adatokból kinyert információk alapján szeretnének dolgozni, ezek az adatbázisok nem megfelelőek. Erre megoldás az információs adatbázis, ahol azokat az adatokat tároljuk, amik a későbbi döntéshozáshoz szükségesek lehetnek. A 2.2-es ábra jól szemlélteti az operatív- és információs adatbázisok közötti lényeges különbségeket. [5]

Jellemzők	Operatív adatbázis	Információs adatbázis
Felhasználók	alkalmazottak	menedzserek
Időszerűség	mindig az aktuális állapotot mutatja	idősorokra van szükség
Adatelérés gyakorisága	folyamatos	alkalomszerű
Adatgyűjtés	egy alkalmazástól	különböző forrásokból
Adatok formátuma	„nyers”, feldolgozatlan adatok	feldolgozott (összegzett, szűrt adatok)
Adatok elérésének módja	több felhasználó egyszerre	általában egy felhasználó
Adatok módosíthatók?	igen, folyamatosan	nem, csak a saját másolat
Adatelérés rugalmassága	nem rugalmas, előre meghatározott mód	nagyon rugalmas
Egy műveletnél használt rekordszám	kb. 10 rekord	100-as,1000-es nagyságrend
Teljesítmény	gyors válaszidő	lehet lassabb
Kívánt adatok	jól definiált, előre tudott	bizonytalan, alkalomtól függő, felfedezés jellegű

2.2 ábra: Operatív- és információs adatbázisok [5]

2.5 Az adattárház koncepciója

Az információs adatbázisok jelenlegi legelterjedtebb és legfejlettebb megvalósítása a Data Warehouse (DW), magyar elnevezés szerint az adattárház. Az adattárház definiálásával az idők során többen is megpróbálkoztak, azonban két általánosan elfogadott meghatározás maradt fent. Az egyik Ralph Kimball, a másik Bill Inmon nevéhez köthető.

Kimball definíciója így hangzik:

„The conglomeration of an organizations’s data warehouse staging and presentation areas, where operational data is specifically structured for query and analysis performance, and ease-of-use.” [6]

Ez a megközelítés azt mondja, hogy az adattárház egy adott szervezet adatgyűjtő és -szolgáltató része, ahol az adatokat újra strukturálják annak érdekében, hogy riportokat lehessen készíteni belőlük, és könnyedén elemezni lehessen őket. Ez a definíció nem határozza meg, hogy az adattárháznak konkrétan csak a döntéshozásban van szerepe.

Inmon definíciója pedig így hangzik:

„A data warehouse is a subject oriented, integrated, nonvolatile, and time variant collection of data in support of management's decision." [4]

Inmon egy olyan adatbázisnak képzelte el az adattárházat, ami témaorientált, integrált, időfüggő, nem felejtő adattár, és a menedzseri döntéshozatalt hivatott támogatni, vagyis egy olyan speciális adatbázis, ahol a vállalati adatok kifejezetten elemzési és riport-készítési célra optimalizálva vannak eltárolva.

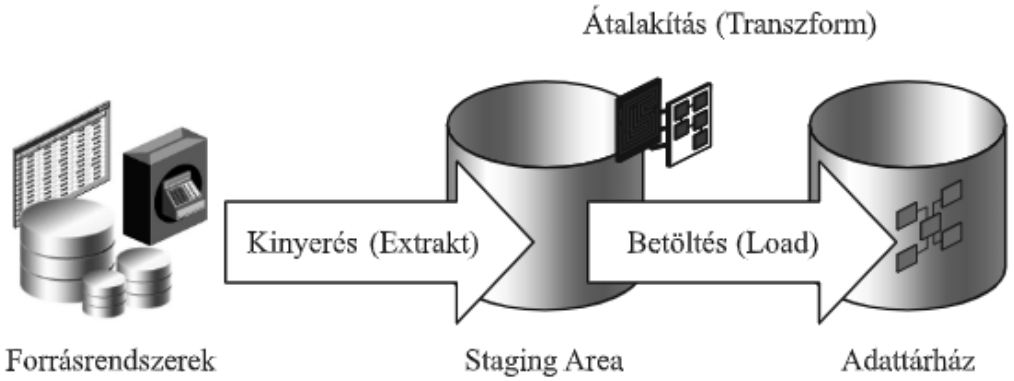
Az Inmon-féle definícióban szereplő fogalmak részletesebb ismertetése:

- Subject oriented (témaorientált): a cél az, hogy az adatok témák szerint csoportosítva, a döntéshez szükséges módon legyenek tárolva.
- Integrated (integrált): az adatok tisztított, egységes kitöltési formátummal rendelkezve, konzisztens elnevezésekkel kerülnek tárolásra.
- Nonvolatile (nem felejtő): az adatok a keletkezési idejüknek megfelelő időbélyeggel kerülnek tárolásra, ezeket az időbélyegeket később már nem változtatják meg, ha valami változás történik, akkor új időbélyeggel felveszik az adatokat, és a korábbi adat lejárati időbélyegét beállítják az aktuális dátumra. Ezért mondják, hogy az adattárház nem felejtő, mert historikus adatokat is tartalmaz.
- Time variant (időfüggő): tekintve, hogy az elemzések általában historikus adatokat és azok időbélyegjeit használják, ezért az adattárházban a forrásrendszerben történő változásokat nyomon követve időpontok és időintervallumok szerint következik be az adatok tárolása.

Az adattárház struktúrában alapvetően két részre lehet osztani az adatokat: dimenzióadatok és tényadatok. A dimenziótáblák általában lassan, vagy egyáltalán nem változó adatokat tartalmaznak, mint például az idő, vagy ügyfélinformációk, termék adatok. A dimenziótábláknak van saját kulcsuk, ami összeköti őket a ténytáblával. A ténytáblák olyan tény adatokat tartalmaznak, amelyek segítségével elkészíthetőek a grafikonok, kimutatások, egyes dimenziók mentén, a dimenzió kulcsának segítségével. [1][11]

2.6 ETL folyamat

Az úgynevezett ETL (Extract, Transform, Load, azaz Adatkinyerés, Átalakítás, Betöltés) folyamat felel a forrásrendszerek eltérő formátumú adatainak kinyeréséért, integrálásáért, átalakításáért, megtisztításáért és az adattárházba való betöltéséért. Ezen folyamatok a legköltségesebbek és ezek veszik igénybe a legtöbb időt az adattárház kiépítése során.



2.3 ábra: ETL folyamat [1]

2.6.1 Az ETL folyamat alapfeladatai

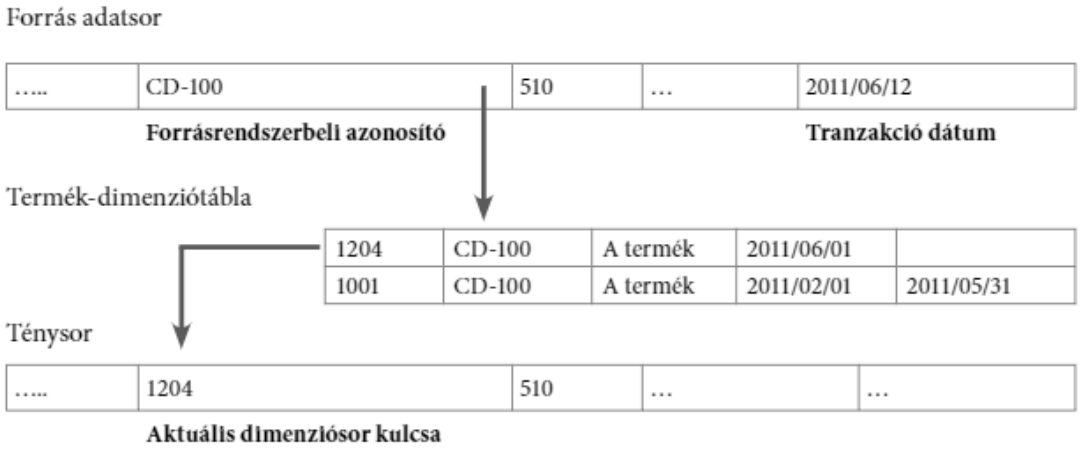
Extract

Amint az a 2.3-as ábrán is látható, az adatkinyerés folyamat azért felel, hogy a forrásrendszerekből a kívánt adatokat a staging területre „hozza”.

Transform

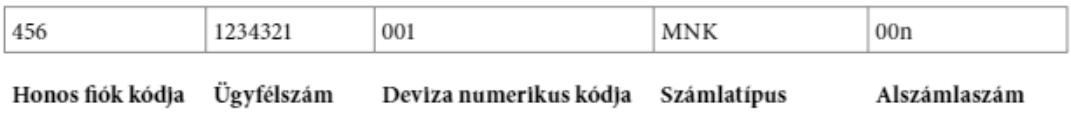
A transzformáció egy összefoglaló elnevezés, számos különböző feladatot értünk alatta. A cél az, hogy a folyamat végére megtisztított, integrált, történetiséget tartalmazó adattárház-struktúra jöjjön létre.

Tipikus transzformációs lépés például, hogy a sorok új, adattárházbeli azonosítót kapnak, annak érdekében, hogy ne keveredjenek az esetlegesen ugyanolyan kulccsal rendelkező, különböző forrásból érkező adatokkal.

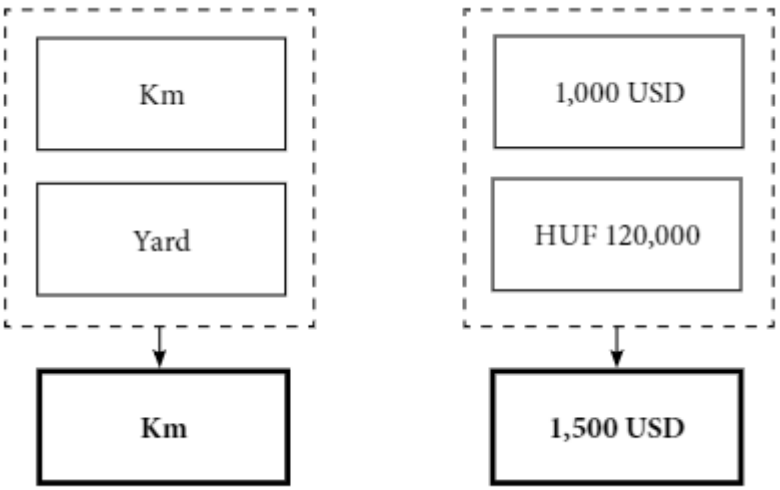


2.4 ábra: Új azonosító hozzárendelése [1]

Tipikus transzformációs lépés még az összetett adatok szétválasztása és a szabványos mértékegységek kialakítása is. Ezekre a 2.5 és 2.6-ös ábrák mutatnak be egy-egy példát.



2.5 ábra: Összetett adatok szétválasztása [1]



2.6 ábra: Szabványos mértékegységek kialakítása

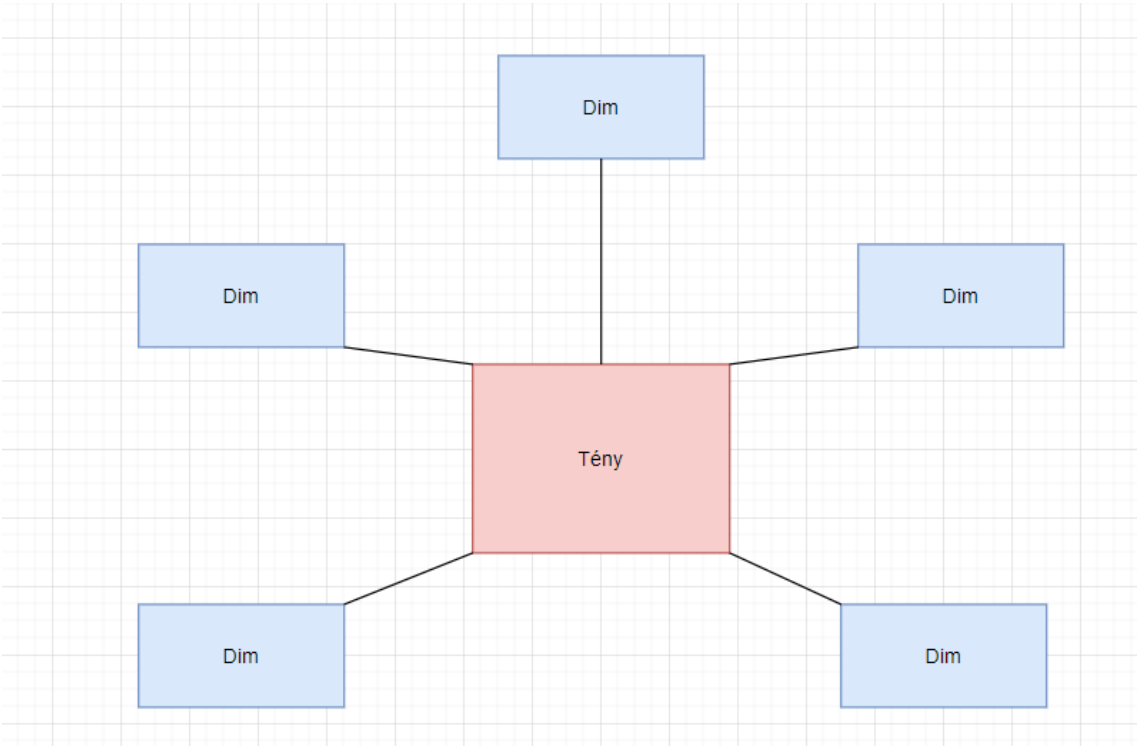
Az adatok betöltése az adattárházba az ETL folyamat utolsó lépése. [1]

2.7 Adattárház séma megvalósításai

A ténytáblák és dimenziótáblák egymáshoz való kapcsolódása és viszonya alapján három elterjedt sémát különböztethetünk meg: csillagséma, hópehelyséma, galaxis séma.

Csillagséma

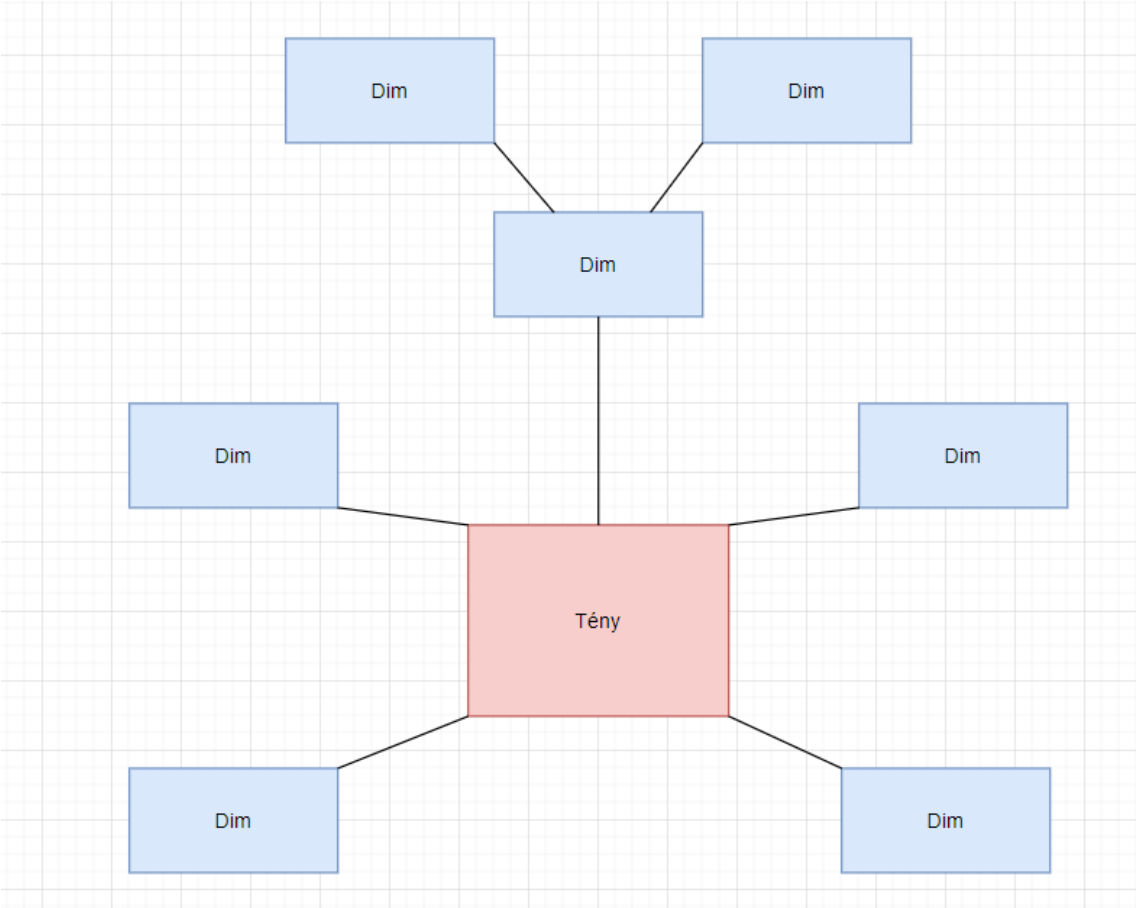
Általánosságban a csillagséma a legelterjedtebb kialakítás, hiszen ez egy alacsony komplexitású adatmodell és a megvalósítása sem túl bonyolult. Az összes dimenziótábla egyetlen központi ténytáblához kapcsolódik, az elsődleges kulcsuk segítségével.



2.7 ábra: Csillagséma

Hópehelyséma

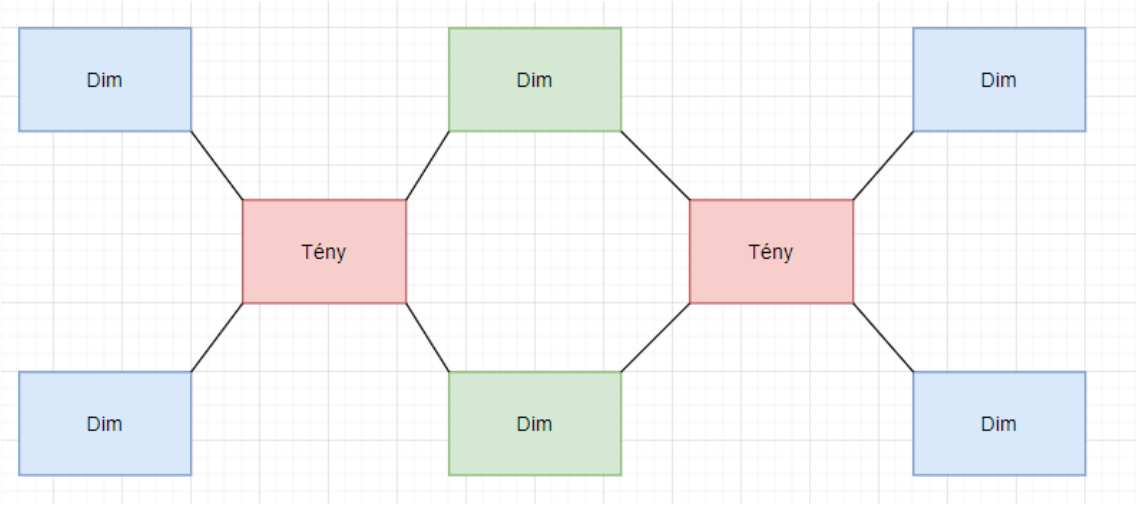
Hópehelyséma esetén a dimenziótáblákat további dimenziókra lehet bontani. Ez helytakarékosság szempontjából előnyös, mivel bizonyos adatok nem redundáns módon kerülnek eltárolásra, azonban teljesítmény szempontjából hátrányos, hiszen a lekérdezéseknél, riportok készítésénél több táblát kell összekapcsolni.



2.8 ábra: Hópehely séma

Galaxis séma

A galaxis séma lényegében a csillagséma kibővítése. Ebben az esetben a dimenziótáblák nem csak egy ténytáblához kapcsolódnak. A ténytáblák között megoldott az átjárhatóság, annak köszönhetően, hogy ugyanazokat a dimenziókat használják.



2.9 ábra: Galaxis séma

3. Webalkalmazás

A webalkalmazás egy dinamikus, általában adatbázisra épülő, asztali alkalmazásokhoz hasonló funkcionalitással, szerver- és kliensoldali logikával rendelkező alkalmazás, ami általában egy konkrét témához kapcsolható. Webalkalmazás például a Google Docs. [12]

3.1 Webalkalmazások kategóriái

A webalkalmazásokat komplexitásuk és időigényességük alapján több csoportba lehet osztani.

Dokumentum központú webalkalmazás

Ezeket a webalkalmazások előzményeiként is nevezik, statikus HTML dokumentumok, általában manuális frissítéssel. Előnyük ugyan, hogy gyorsak, egyszerűek és megbízhatóak, azonban a gyakori frissítés megnöveli a karbantartás költségeit és nem automatizálható. [12]

Interaktív és tranzakcionális webalkalmazás

Az interaktív webalkalmazásokat azért nevezzük így, mert interaktív elemeket tartalmaznak, pl.: űrlapok, gombok. Ilyen oldalak jellemzően a modern híroldalak, ahol a dinamikus megjelenés a felhasználói beviteltől függ. A tranzakcionális webalkalmazásoknál általában már jellemző az adatbázis használata, valamint a tartalom frissítése a CRUD (Create – Read – Update - Delete, Létrehozás - Olvasás - Módosítás - Törlés) műveletek segítségével. [12]

Munkafolyamat-alapú alkalmazások

Általában üzleti folyamatokon alapuló alkalmazások, felépítését, struktúráját az üzleti logika határozza meg. Ilyenek például a szolgáltatásorientált (SOA) alkalmazások. [12]

Mindenhonnan elérhető (Ubiquitous) alkalmazások

Ezek magas komplexitással rendelkező webalkalmazások, amelyek olyan testreszabott szolgáltatásokat nyújtanak, amiket bármilyen eszközről el lehet érni. Nagy szerepet kap az emberek és számítógépek közötti „együttműködés” (Human-Computer Interaction, röviden HCI). Ez egy jelenleg is fejlődő terület, ilyen alkalmazás pl.: Evernote. [12]

4. Fejlesztői eszközök

Adatbáziskezelő rendszer tekintetében lehetőség a Microsoft SQL Server, valamint az Oracle is. Lényeges különbség, hogy az Oracle szinte bármilyen platformon használható, míg az MSSQL egészen a 2017-es verzióig, csak Windowson használható, a 2017-es és az utána lévő verziók Linuxon is elérhetőek. Mindkét eszköz alkalmas nagy adatmennyiség tárolására, és mindegyiknek van ingyenesen elérhető része. Az Oracle-ben van beépített lekérdezés optimalizálás is. Nem lesz szükségem lekérdezés optimalizálásra, és Windows operációs rendszeren fogok dolgozni. Az előbb említett információk, valamint annak függvényében, hogy az Oracle kicsit komplexebb, és nehezebb kezelni, az MSSQL mellett döntöttem.

Az ETL folyamatok megvalósítására a Microsoft – SQL Server Integration Services (SSIS) és a Talend között gondolkodtam. Az SSIS alkalmas a folyamatok párhuzamos futtatására, valamint rengeteg eszközt biztosít ezen folyamatok elvégzéséhez. Viszonylag egyszerűen használható, flexibilis. Az SSIS integrálva van a Microsoft Visual Studioba, ezáltal könnyen elérhető számomra, valamint egyszerűen szinkronizálni lehet Gittel, ha szükséges. A Talendben is megtalálható minden szükséges eszköz az ETL elvégzéséhez, és szinte minden adatbáziskezelő rendszerrel összekapcsolható. Jó választássá teszi az is, hogy kifejezetten könnyen lehet vele XML és JSON formátumú adatokat kezelni. Azonban nehézkes a szinkronizálása Gittel, valamint összeomlásra hajlamos, ha több millió sornyi adattal próbálunk dolgozni. Az SSIS mellett szól még az is, hogy mivel ugyanúgy Microsoft termék, mint a választott adatbáziskezelő rendszer, így biztosan nem lesz probléma az összekapcsolhatósággal. Ezeket figyelembe véve, az SSIS mellett döntöttem.

Az adatok vizualizációjához számos eszköz elérhető, azonban a végére kettő maradt, amik közül választanom kellett. A Microsoft PowerBI és az ChartJS. A PowerBI előnye, hogy kifejezetten egyszerű használni, nem komplex. Az asztali verziója ingyenes, és a Microsoft jó dokumentációt biztosít hozzá, ha bármi probléma merülne fel használat közben. A ChartJS is ingyenes, és rengeteg eszköz elérhető benne az igényes és hasznos vizualizációk készítéséhez. Mindkét eszköz alkalmas nagy adatmennyiség feldolgozására. A fő oka annak, hogy az ChartJS-re esett a választásom, hogy a PowerBI-nak fizetős a publikációja, vagyis nem tudnám a beépíteni a webalkalmazásba a vizualizációkat, míg a ChartJS-el ezt könnyedén, ingyen megtehetem JavaScript kód formájában.

A webalkalmazás fejlesztésére az ASP.NET Core keretrendszert választottam. Rengeteg beépített funkciót tartalmaz, ami megkönnyíti a fejlesztést. Támogatja a C#, HTML és JavaScript kódokat. Lehetőség volt még a PHP is, jó választás lehet annak tekintetében, hogy viszonylag könnyen tanulható és teljesen ingyenes. Nagyon népszerű, ennek köszönhetően rengeteg megoldást lehet találni az interneten, ha valamivel elakad az ember. Nagyon sok egyéni beállítási lehetőség van benne. Mindkettő gyors és megbízható, azonban annak fényében, hogy nagy adatmennyiséggel fogok dolgozni,

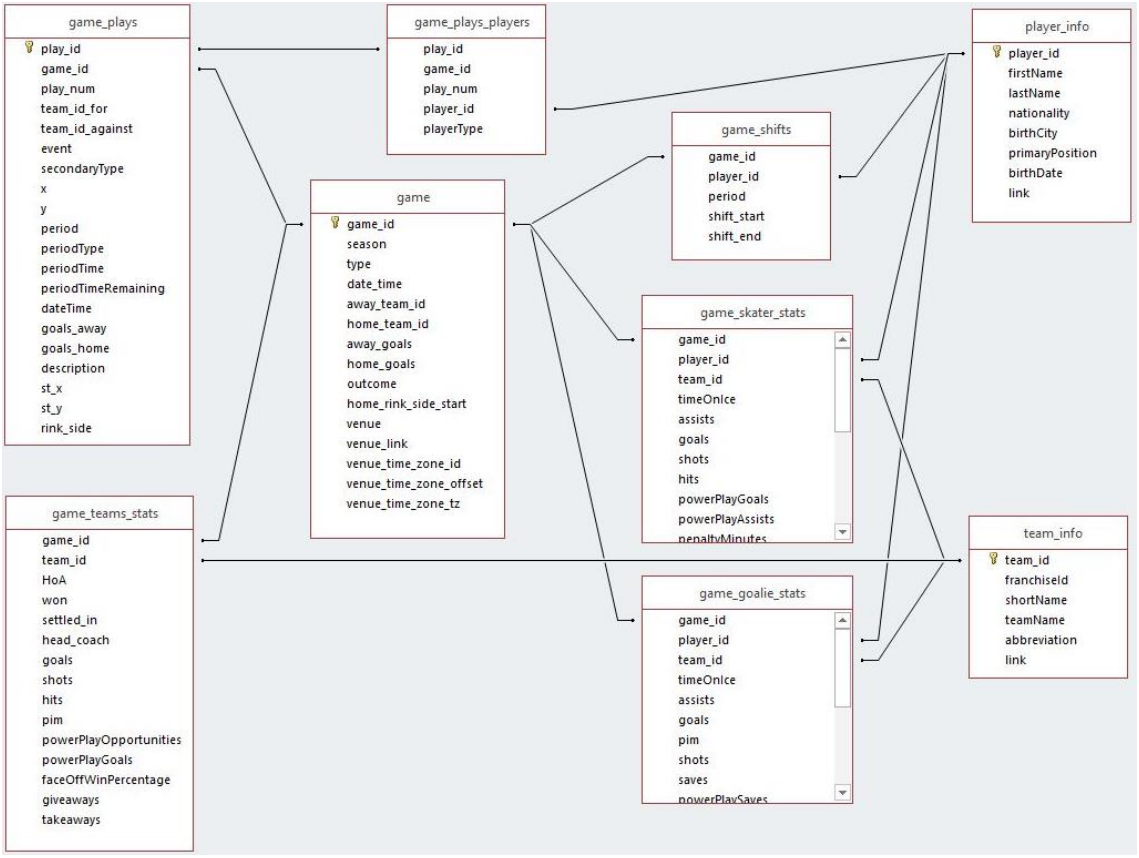
biztosabb választás az ASP.NET. A fentebb választott vizualizációs eszköz pedig támogatja az ASP.NET-be ágyazott JavaScript kódot a vizualizációk beépítésére.

5. Specifikáció

Az adattárházba töltendő adathalmaz az NHL (National Hockey League) hivatalos mérési adatait tartalmazza. Ezek az adattáblák körülbelül húsz évre visszamenőleg az összes NHL által lebonyolított jégkorong mérkőzéssel kapcsolatosan tartalmaznak adatokat. A cél az, hogy az adatok adattárházba történő töltése után vizualizációkat készítsek, valamint egy ezen adatokra épülő webalkalmazást hozzak létre.

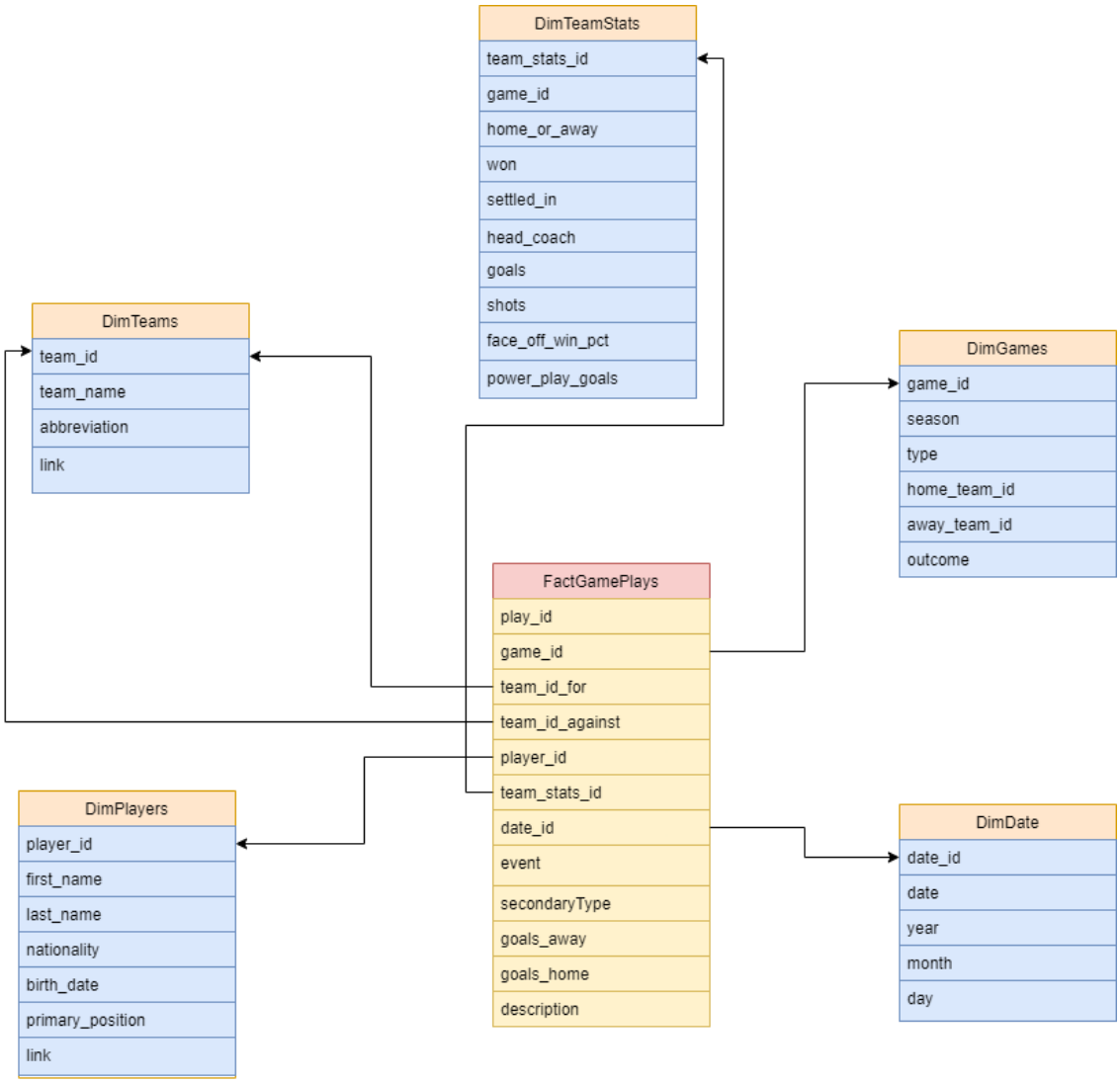
5.1 Táblák

Az alap táblastruktúra a következőképpen néz ki:



5.1 ábra: Táblastruktúra

Főként a *game_plays*, azaz az összes mérkőzés összes jelentősebb eseményét tartalmazó táblával szeretnék foglalkozni, és emellé szükségem van az ehhez kapcsolódó (játékosok, csapatok, csapatok statisztikái) táblákra. Ezeket figyelembevéve, valamint egy külön dátum dimenziót hozzáadva, létrejött a csillagséma.



5.2 ábra: Csillagséma táblái

5.2 A tervezett csillagséma táblái

FactGamePlays: Az összes mérkőzés összes eseményét, valamint az esemény bekövetkeztekor fennálló adatokat tartalmazza. Ez a ténytábla.

DimPlayers: Játékosokat és azok adatait tartalmazó dimenziótábla.

DimTeams: Csapatokat és azok adatait tartalmazó dimenziótábla.

DimTeamStats: Minden csapat minden mérkőzéshez kapcsolódó statisztikáit tartalmazó dimenziótábla.

DimGames: Minden mérkőzés alap adatait tartalmazó dimenziótábla.

DimDate: Dátum dimenziótábla.

5.3 A webalkalmazás tervezete

A cél egy olyan webalkalmazás megvalósítása, ahol a felhasználó kedvére szűrhet a mérkőzések között, megnézheti annak adatait, eseményeit, valamint az ebben az időszakban az adatbázisban megtalálható csapatokra és játékosokra is rákereshet. Bizonyos szűrési esetekben pedig az előre elkészített vizualizációk jelennek majd meg, a szimpla adatok mellé. Az adatok és az elemzések eredményeinek ábrákon, diagramokon való megjelenítése nagyban segíti a felhasználót abban, hogy egyrészt megértse azokat, másrészt esetleges következtetéseket vonjon le a múltra nézve, vagy akár döntéseket hozzon a jövőre nézve. Például, ha valaki a kedvenc csapatára keres rá, meg tudja nézni az utóbbi évek eredményeit, össze tudja hasonlítani azokat.

A főoldalról lehetőség legyen elnavigálni különböző menüpontok segítségével, valamint minden esetben visszatérni oda. A csapatok legyenek kilistázva, és az összesített számuk is legyen feltüntetve. A játékosok között lehessen szűrni különböző feltételek megadása mellett, például kereszt- és vezetéknév, valamint pozíció szerint. A mérkőzések között szintén szűrési feltételek megadása mellett lehessen keresni, például csapatnév és szezon szerint. Legyen feltűntetve az adott keresési feltétel melletti találatok száma, valamint a talált mérkőzésekhez tartozó adatok. Az eseményeket kétféle módon lehessen listázni: elsődleges és másodlagos események. Mindegyik eseményhez legyen feltűntetve az előfordulásainak száma.

A vizualizációkat szintén a főmenüből lehessen elérni.

A tervezett vizualizációk: Játékosok nemzetiség szerinti eloszlása, összesített gólok száma idényenként, hazai és idegenben lőtt gólok csapatokra bontva, emberelőnyben szerzett gólok aránya, játékosok súlya és pozíciója közötti összefüggés, valamint az elsődleges események előfordulása.

5.4 Tervezett ütemezés

Először az adatok adattárházba töltését kell elvégezni, az ETL folyamatok segítségével. Az ETL folyamatok során fog megtörténni az adatok egységes formátumra hozása, ha esetleg lennének eltérések. Elírások, üres mezők felderítése, kezelése.

Az ETL folyamatokkal párhuzamosan már elkezdődhet a webalkalmazás tervezése, fejlesztése, a lényeg, hogy mikor arra kerül a sor, legyenek elérhetőek az előző lépés által létrehozott adatok.

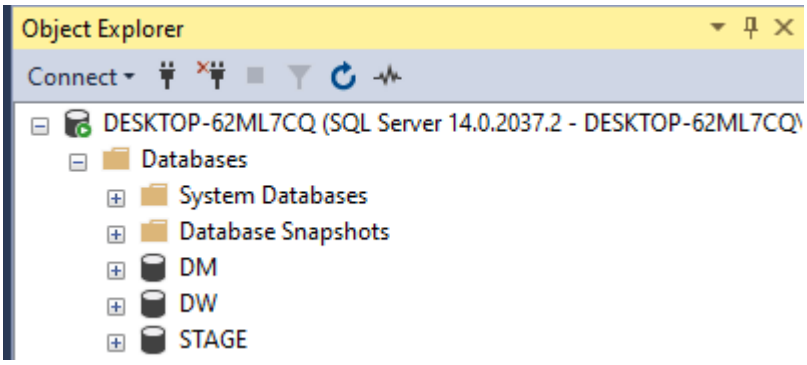
A webalkalmazás fejlesztéséhez tartozik a vizualizációk megjelenítésére szolgáló JavaScript kód elkészítése is.

Miután elkészült a webalkalmazás, szükség lesz annak tesztelésére is, hogy minden funkció a kívánt módon működik-e. Az esetleges hibák kijavítása és a finomhangolás marad a végére.

6. ETL folyamatok

6.1 Előkészületek

Az adatok sikeres adattárházba történő töltése érdekében először szükséges volt létrehozni a megfelelő adatbázisokat az adatbáziskezelő alkalmazásban. Amint a 6.1-es ábrán látható, három adatbázist hoztam létre.



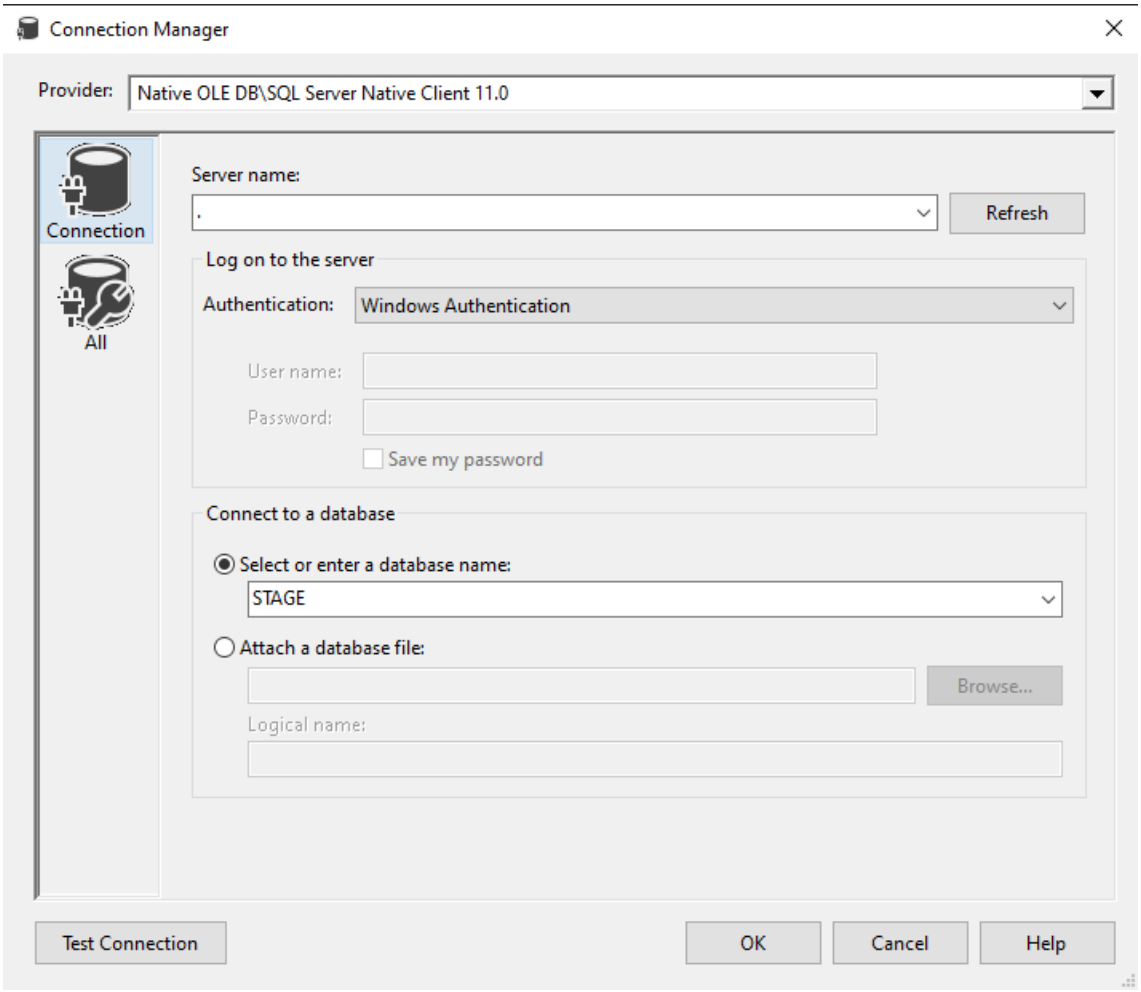
6.1 ábra: Adatbázisok

Az előkészületekhez tartozott még továbbá a forrásadatok vizsgálata is. Itt történt egy új oszlop beszúrása az egyik dimenziótáblába, valamint a duplikátumok eltávolítása az összes táblából.

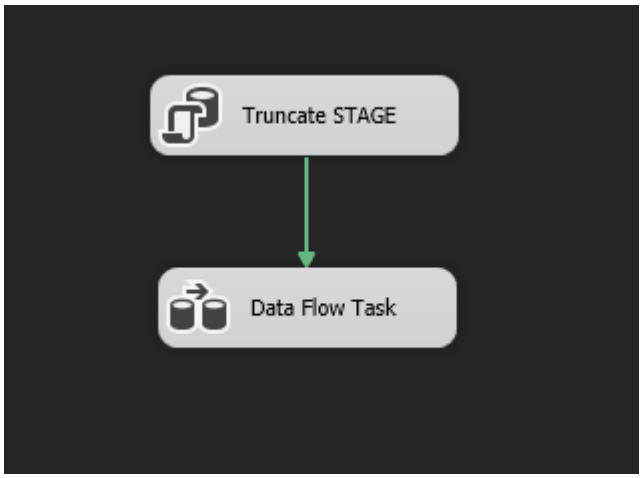
6.2 Extract

Itt történt az adatok kiolvasása a .csv kiterjesztésű forrásfájlokból és a STAGE rétegbe töltésük. Ennek érdekében először szükséges volt kialakítani a kapcsolatot az adatbázissal, ennek a beállítása a 6.2-es ábrán látható.

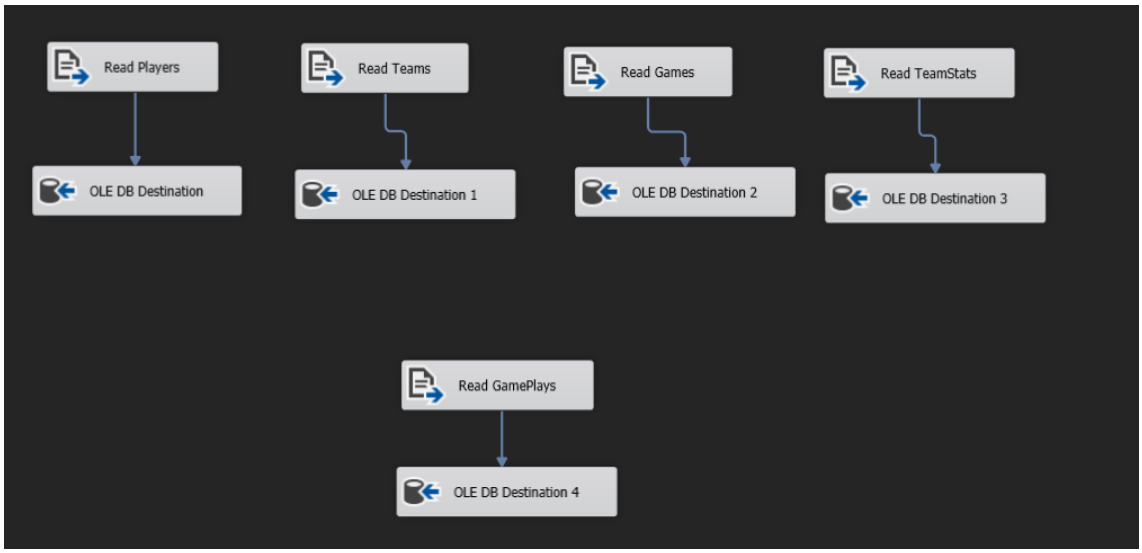
Ezután egy SQL Task segítségével a forrásadatok betöltése előtt kiürítünk minden táblát az adatbázisból. Ez azt a célt szolgálja, hogy ha esetleg hibásan futna le a program és emiatt több alkalommal próbálkoznánk az adatok áttöltésével, ne sokszorozódjanak a sorok az adatbázisban. A következő objektum egy Data Flow Task, ebben valósult meg az adatok beolvasása, táblánként egy-egy Flat File Source objektum segítségével. A Control Flow a 6.3-as ábrán, a Data Flow pedig a 6.4-es ábrán látható.



6.2 ábra: Kapcsolat az adatbázissal



6.3 ábra: Control Flow-Extract réteg

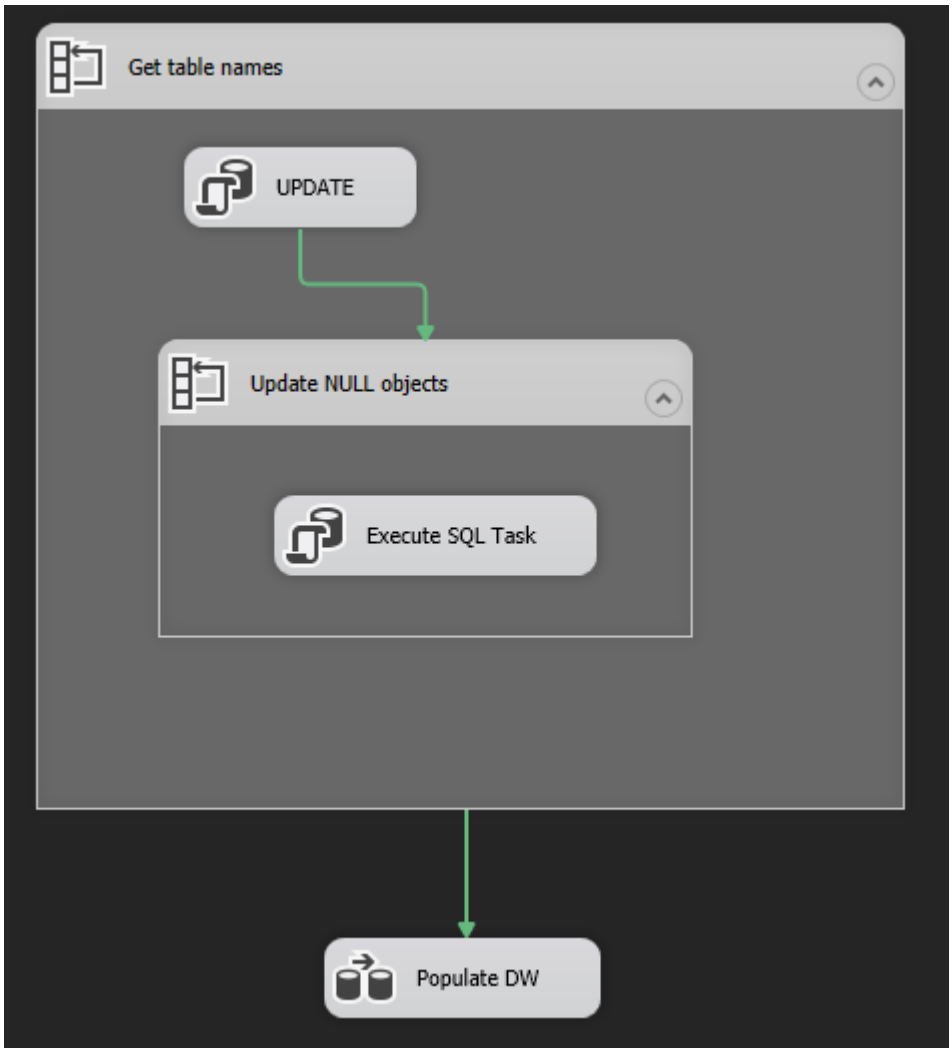


6.4 ábra: Data Flow-Extract réteg

6.3 Transform

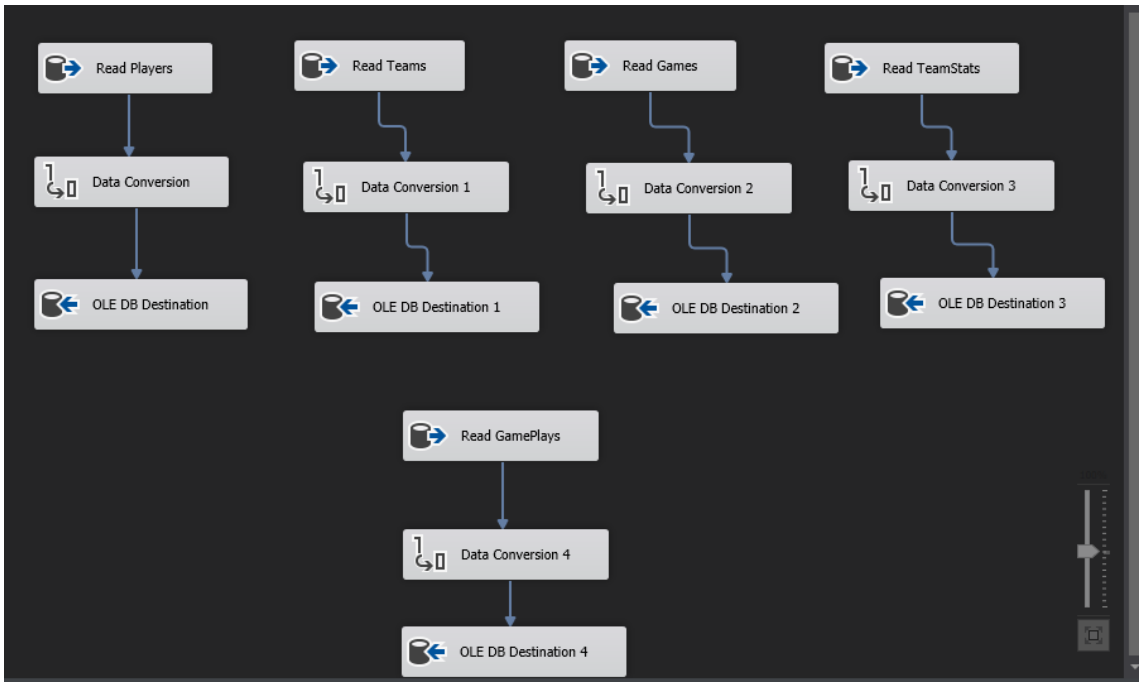
A következő lépésben a DW adatbázist töltöttem fel. Ehhez először a STAGE rétegből a „NULL” értékeket valódi NULL értékekké kellett konvertálni. Ez a folyamat a 6.5. ábrán látható. A külső ciklus meghatározza a táblák neveit, amelyeken végig kell iterálni, a belső ciklus pedig a tábla sorain fog végig lépkedni és frissíteni azokat.

Az ezután következő objektum pedig egy Data Flow, amiben az adatbázis táblák feltöltése történt meg.



6.5 ábra: Control Flow-Transform réteg

Ebben a Data Flow-ban az adatok beolvasásra kerültek a STAGE rétegből, és egy Data Conversion objektumon keresztül a DW adatbázis rétegbe töltődtek. Az adatok konverziójára azért volt szükség, mert eddig a pontig minden oszlop VARCHAR (256) típusban volt tárolva, itt azonban már beállításra kerültek a megfelelő típusok és hosszúságok. Ezek az objektumok a 6.6-os ábrán láthatóak.



6.6 ábra: Data Flow-Transform réteg

6.4 Load

A harmadik rétegben OLE DB Source segítségével beolvasásra kerültek az adatok a DW adatbázis rétegből, ezután Lookup objektumok segítségével meghatároztuk a dimenziótáblák Surrogate Key értékeit. A ténytáblában lévő kulcsokhoz be kellett állítani a szükséges dimenziótábla oszlopokat. Ezzel fel is töltődött az adattárház.

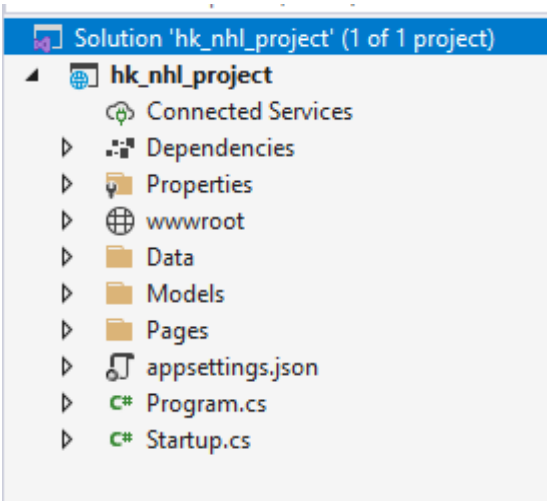
7. Webalkalmazás

7.1 Fejlesztő környezet

A webalkalmazást a Microsoft Visual Studio 2019 használatával, egy ASP.NET Core Razor Pages projekt keretei között valósítottam meg. Tekintve, hogy már rendelkeztem az adatbázissal, a fejlesztés során a Database First megközelítést használtam, vagyis az adatbázis alapján hoztam létre a modelleket.

7.2 Mappastruktúra

A projekt mappastruktúrája a 7.1-es ábrán látható módon épült fel.



7.1 ábra: A projekt mappastruktúrája

Logikai szempontból alapvetően három részre osztható a projekt: Data, Models, Pages. A Data mappa tartalmazza az adatbázis kapcsolathoz szükséges interfészt és osztályt, valamint annak metódusait. Ezek segítségével lehet lekérdezni az adatokat az adatbázisból. Az adatbázis kapcsolathoz szükséges Connection String-et a 7.2-es kódrészlet tartalmazza.

```
"ConnectionStrings": {
"NHL": "Data Source=.;Initial
Catalog=DM;Integrated Security=True"}
```

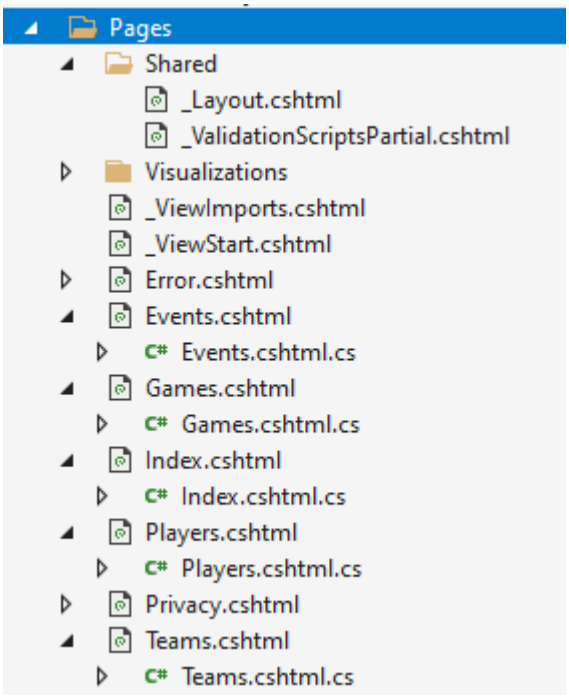
7.2 kódrészlet: Connection String

A Models mappában találhatóak az adatbázis táblák megfelelői osztályok formájában. Ezeket az osztályokat, valamint a *DbContext* osztályt a 7.3-es kódrészlet segítségével generáltam.

```
"Scaffold-DbContext „Data Source=.;Initial
Catalog=DM;Integrated Security=True”
Microsoft.EntityFrameworkCore.SqlServer -OutputDir
Models"
```

7.3 kódrészlet: DbContext és Model osztályok generálása

A Pages mappában találhatóak az „oldalak” „.cshtml” kiterjesztéssel és a hozzájuk szorosan hozzájuk tartozó „.cshtml.cs” kiterjesztésű PageModel osztályok. Az előbb említett objektumok a 7.4-es ábrán láthatóak.



7.4 ábra: Pages mappa tartalma

7.3 Egy oldal felépítése

Mint azt már fentebb is említettem, egy oldal alapvetően egy Page objektumból és egy hozzá tartozó PageModel osztályból áll. Ebben az osztályban lehet testre szabni az oldal OnGet és OnPost metódusait, vagyis azt, hogy mi történjen az oldal első betöltésekor és például egy „Keresés” gomb lenyomásakor. Magában a Page

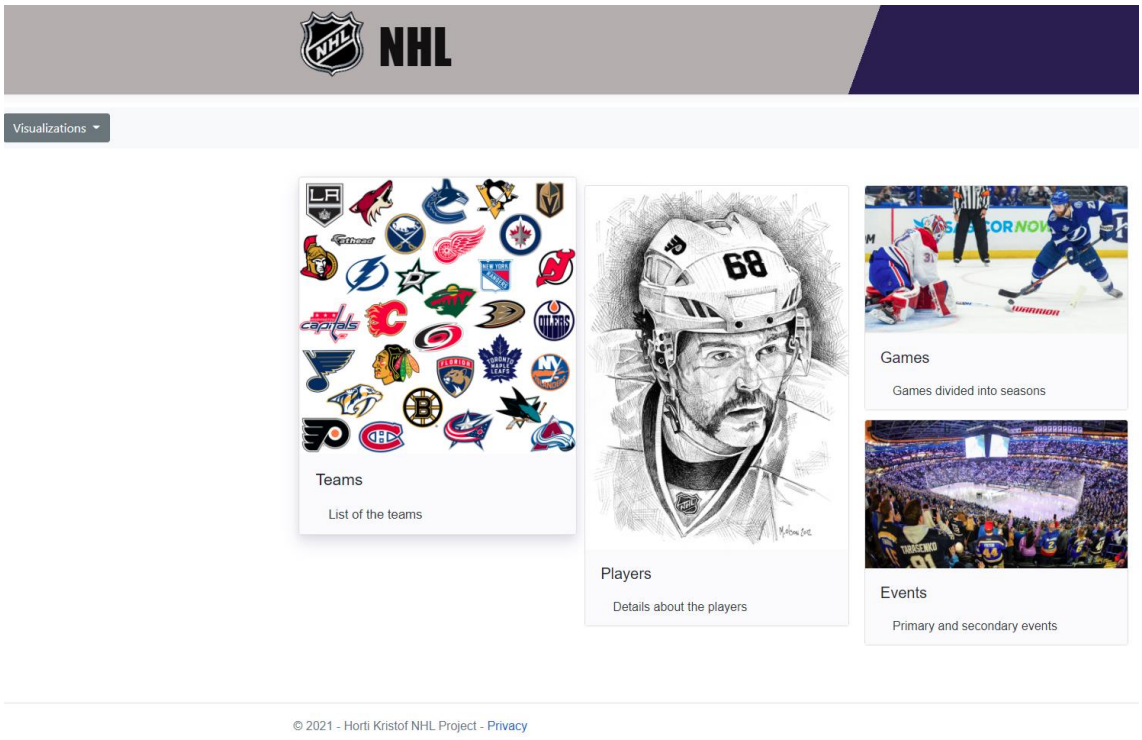
objektumban elsősorban HTML kóddal lehet dolgozni, azonban a „@” beírása után lehetőség van közvetlenül C# kóddal dolgozni. Az oldalhoz tartozó Model osztályban található mezőket az „asp-for” paranccsal lehet elérni, ez a 7.5 kódrészletben látható egy bemutatott példán, ami a játékosok kereséséhez kapcsolódik.

```
<form method="post" class="card p-3">
    <div class="row">
        <div asp-validation-summary="All"></div>
    </div>
    <div class="form-group mb-0 align-middle">
        <label asp-for="SearchedPlayer.FirstName">First
Name</label>
        <input type="text" asp-
for="SearchedPlayer.FirstName" class="mr-5">
        <label asp-for="SearchedPlayer.LastName">Last
Name</label>
        <input type="text" asp-
for="SearchedPlayer.LastName" class="mr-5">
        <label asp-
for="SearchedPlayer.PrimaryPosition">Primary
Position</label>
        <select asp-for="SearchedPlayer.PrimaryPosition"
asp-items="Html.GetEnumSelectList<Positions>()" class="mr-
5"></select>
        <button class="btn btn-primary">Find</button>
    </div>
    <div class="row">
        <label style="color:darkgreen">
@Model.SelectedPlayers.Count() players found. </label>
    </div>
</form>
```

7.5 kódrészlet: PageModel Property elérése/használata

8. A webalkalmazás bemutatása

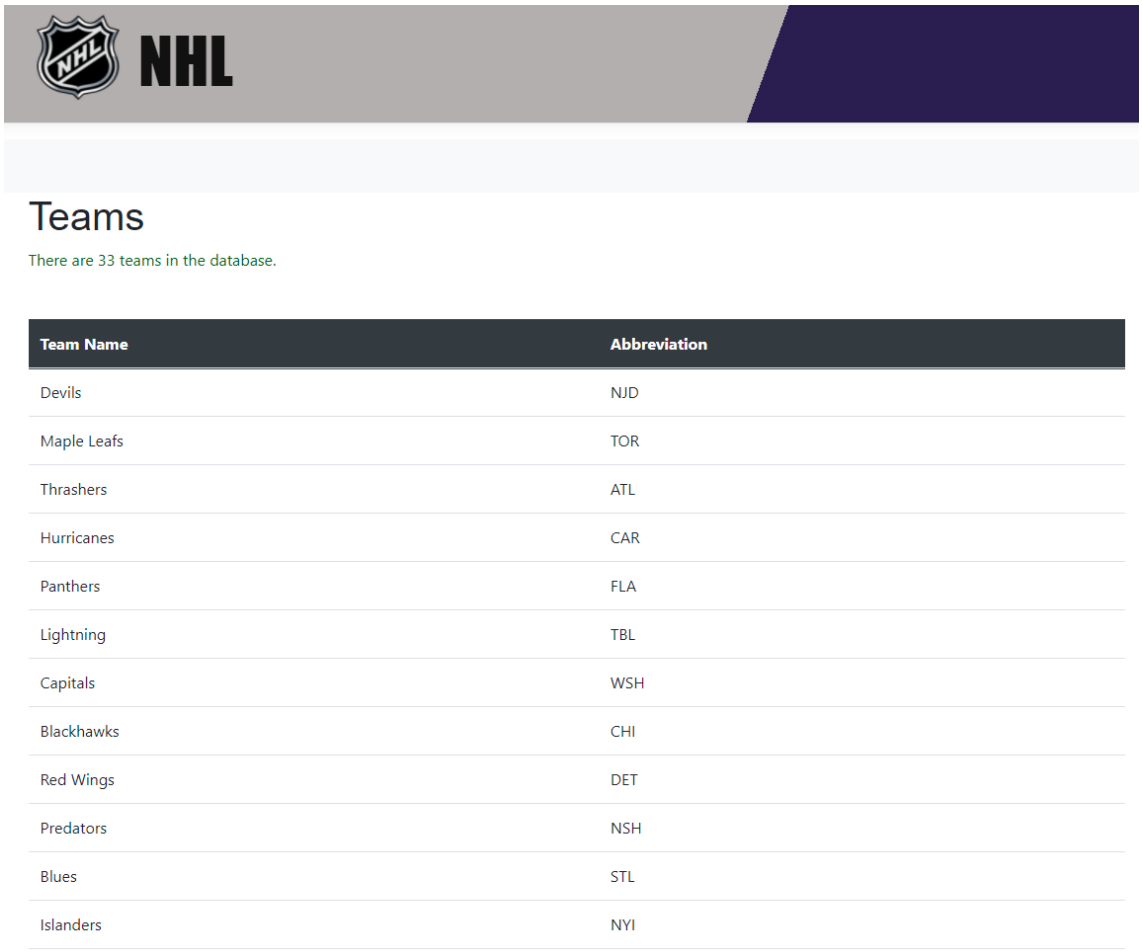
Az alkalmazás indítását követően a főoldal fogadja a felhasználót. Itt az úgy nevezett „navigáló kártyák” segítségével lehet elérni az különböző oldalakat. Ezt az oldalt a 8.1-es ábra mutatja be. Az kép fenti részén található sáv a webalkalmazás egészére érvényes, azaz minden oldalon látszódik. Az „NHL” felirat és a logó össze van kapcsolva a főoldallal, ami azt jelenti, hogy bárhol járunk az oldalon, rájuk kattintva mindig vissza lehet jutni a főoldalra.



8.1 ábra: Főoldal

8.1 Csapatok

A „Teams” kártyára kattintva megjelennek az adatbázisban tárolt csapatok. Amint a 8.2-es ábrán is látható, jelenleg 33 csapatról találhatóak információk az oldalon. A csapatok neve mellett a három betűs rövidítésük is megtalálható, amelyeket általában a mérkőzések során használnak, például az eredményjelző táblán.

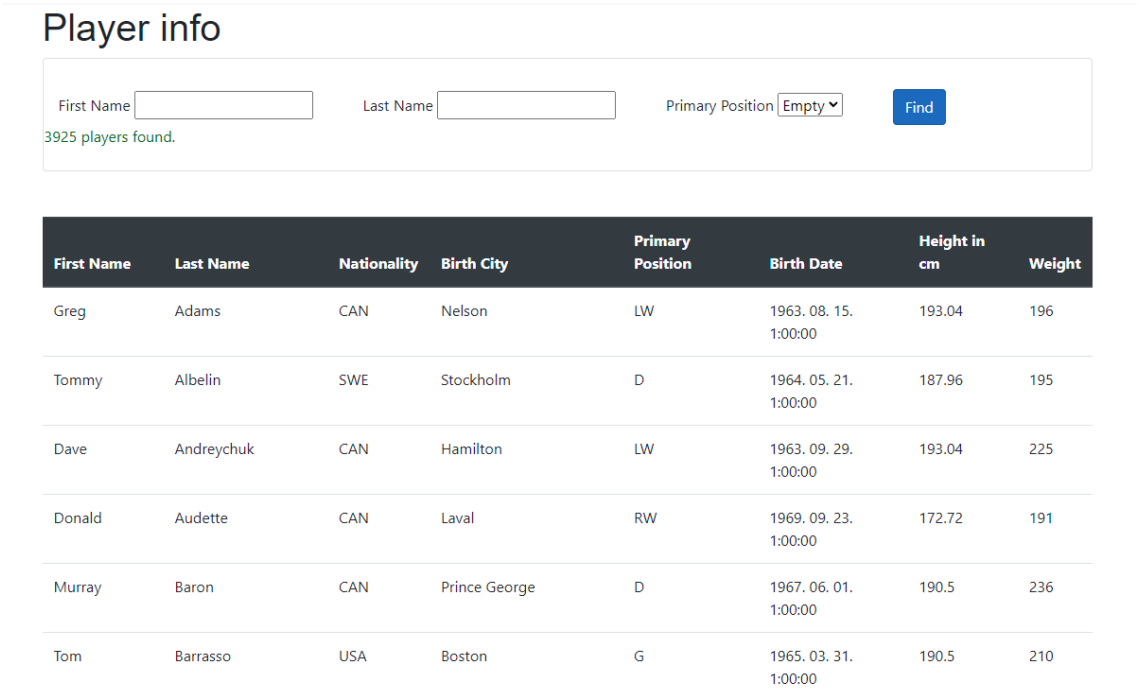


8.2 ábra: Adatbázisban tárolt csapatok

8.2 Játékosok

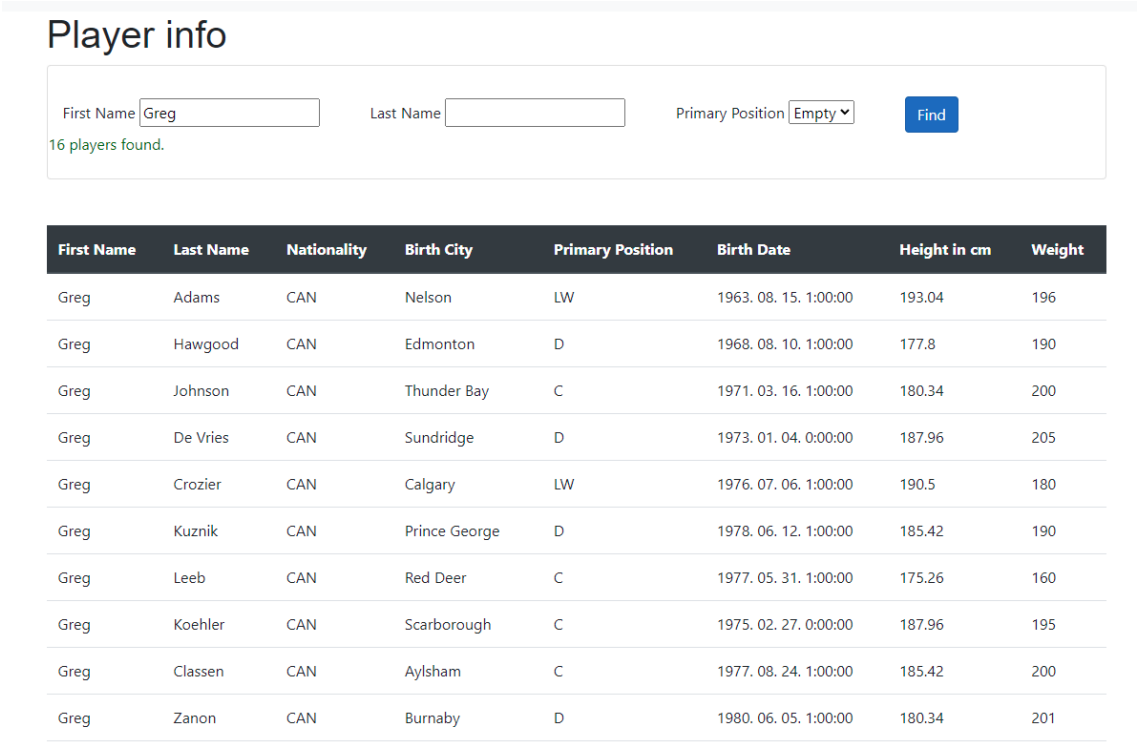
A játékosokat a „Players” feliratú kártyára kattintva lehet elérni. Alapértelmezetten az adatbázisban található összes játékost kilistázza a program, így, ha valakinek nincsen konkrét elképzelése, csak a játékosokat szeretné böngészni, az megteheti.

Viszont azt figyelembe véve, hogy a felhasználók valószínűleg konkrét játékost/játékosokat keresnek, beépítettem egy egyszerűbb keresőablakot a programba. Ez a felület a 8.3 ábrán figyelhető meg.



8.3 ábra: Keresési lehetőségek

Amint a fenti ábrán is látható, lehetőség van kereszt -és vezetéknév, valamint elsődleges pozíció szerint keresni. A keresési feltételek összevonhatóak, tehát ha pontosan tudjuk egy játékos nevét, akkor rögtön meg fogja találni a program, azonban abban az esetben, ha csak a keresztnevére emlékszünk, akkor le fogja szűkíteni a találatokat az adott keresztnévre. Erre látható egy példa a 8.4-es ábrán.



8.4 ábra: Részleges keresés vezetéknév alapján

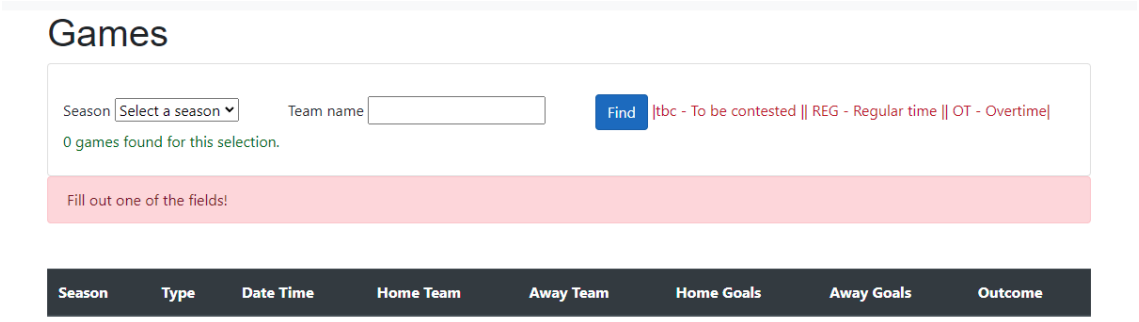
A fenti keresés esetében csak a játékos keresztnevét adtam meg, és ennek köszönhetően a több ezer nevet leszűkítette 16-ra.

Ugyanígy lehet szűkíteni a találatokat a többi feltétel segítségével is, például, ha csak a kapusokat szeretnénk listázni, akkor az elsődleges pozíciók legördülő menüjéből a „G” -t, azaz a „Goaltender” -t kell választani.

8.3 Mérkőzések

A jégkorong mérkőzéseket és a hozzájuk kapcsolódó információkat a „Games” kártyára kattintva érjük el.

Mivel itt jóval több adatsor található, mint a játékosok esetében, ezért értelmetlen lenne kilistázni az összes játékot, így a keresési felület is ehhez lett igazítva: amennyiben nem adunk meg keresési feltételt, egy piros hibaüzenet jelzi, hogy szűkítsük a keresést. Ez a 8.5-ös ábrán látható.



8.5 ábra: Hibaüzenet keresési feltételek nélkül

A mérkőzéseket alapvetően szezonokra lebontva vizsgálhatjuk meg, a 2000-2001-es idénytől kezdve egészen a 2019-2020-asig. Ha a felhasználó nem kíváncsi az adott szezon összes összecsapására, tovább pontosíthatja a szűrést például a kedvenc csapata nevével, így az adott időszakban csak a csapata által játszott mérkőzések kerülnek megjelenítésre. Erre egy példát a 8.6-os ábrán lehet megtekinteni.

Games

Season 2001-2002 Team name Stars Find |tbc - To be contested || REG - Regular time || OT - Overtime|

82 games found for this selection.

Season	Type	Date Time	Home Team	Away Team	Home Goals	Away Goals	Outcome
20012002	R	2001. 10. 06. 2:30:00	Stars	Predators	4	1	home win REG
20012002	R	2001. 10. 07. 19:30:00	Hurricanes	Stars	3	0	home win REG
20012002	R	2001. 10. 10. 2:30:00	Stars	Kings	2	1	home win REG
20012002	R	2001. 10. 12. 2:30:00	Stars	Canucks	1	4	away win REG
20012002	R	2001. 10. 14. 2:00:00	Stars	Flames	3	4	away win OT
20012002	R	2001. 10. 18. 2:00:00	Blues	Stars	2	2	tbc win OT
20012002	R	2001. 10. 19. 2:30:00	Stars	Coyotes	3	1	home win REG
20012002	R	2001. 10. 21. 2:00:00	Stars	Blackhawks	2	2	tbc win OT
20012002	R	2001. 10. 25. 2:00:00	Penguins	Stars	3	2	home win REG
20012002	R	2001. 10. 27. 1:30:00	Red Wings	Stars	3	5	away win REG
20012002	R	2001. 10. 28. 19:00:00	Islanders	Stars	3	2	home win OT

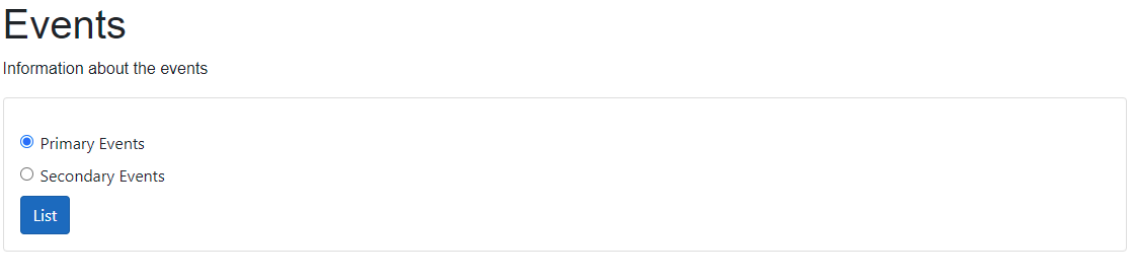
8.6 ábra: Csapat és szezon szerinti keresés

Ahogyan az az előző két ábrán is megfigyelhető, a keresési sávba beépítettem egy jelmagyarázatot, amely a mérkőzések végkimenetelének megértésében segítheti a felhasználót. Egy adott évben általában minden csapat 82 játékban vett részt.

8.4 Események

A mérkőzéseken történt eseményeket a főoldalról az „Events” kártyára navigálva lehet elérni.

Itt sincs alapértelmezett listázás, tekintve, hogy közel 5 millió adatsor található ebben a táblában. Két lehetőség közül lehet választani, az 5.7-es ábrán látható módon: elsődleges és másodlagos események.

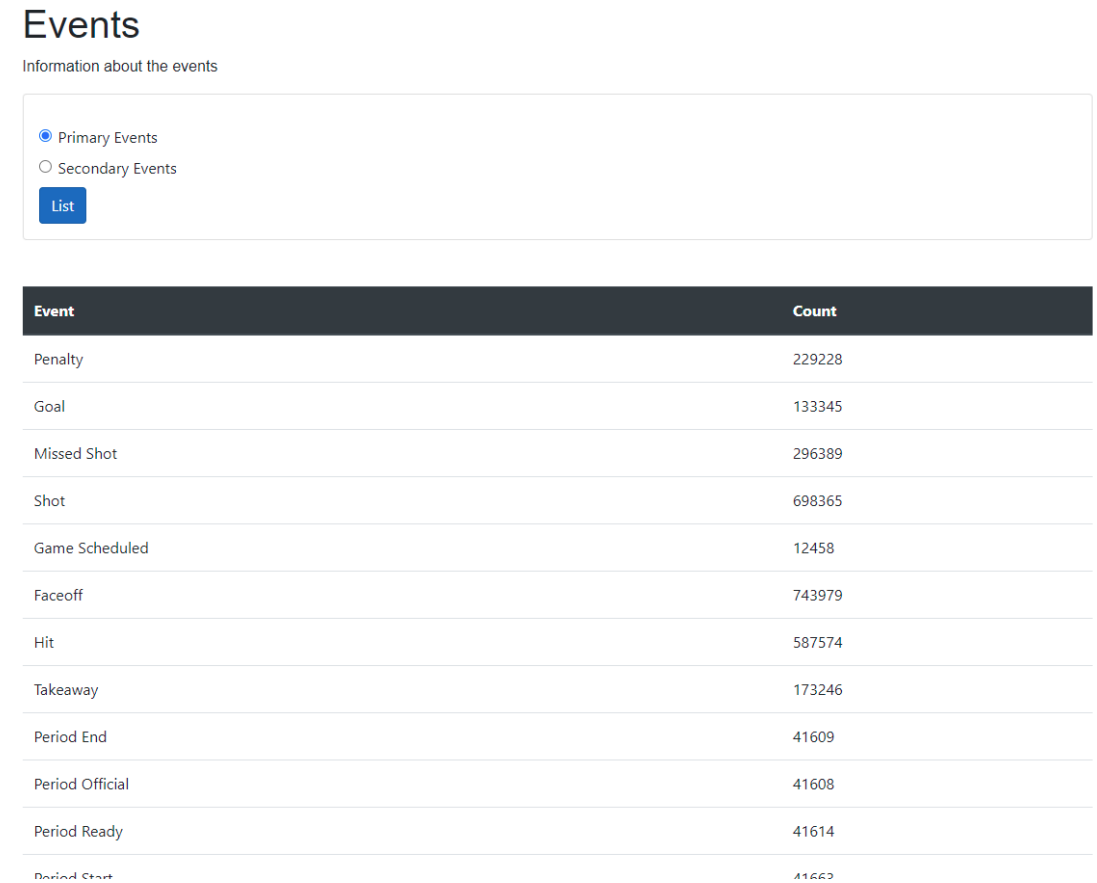


8.7 ábra: Események szűrése

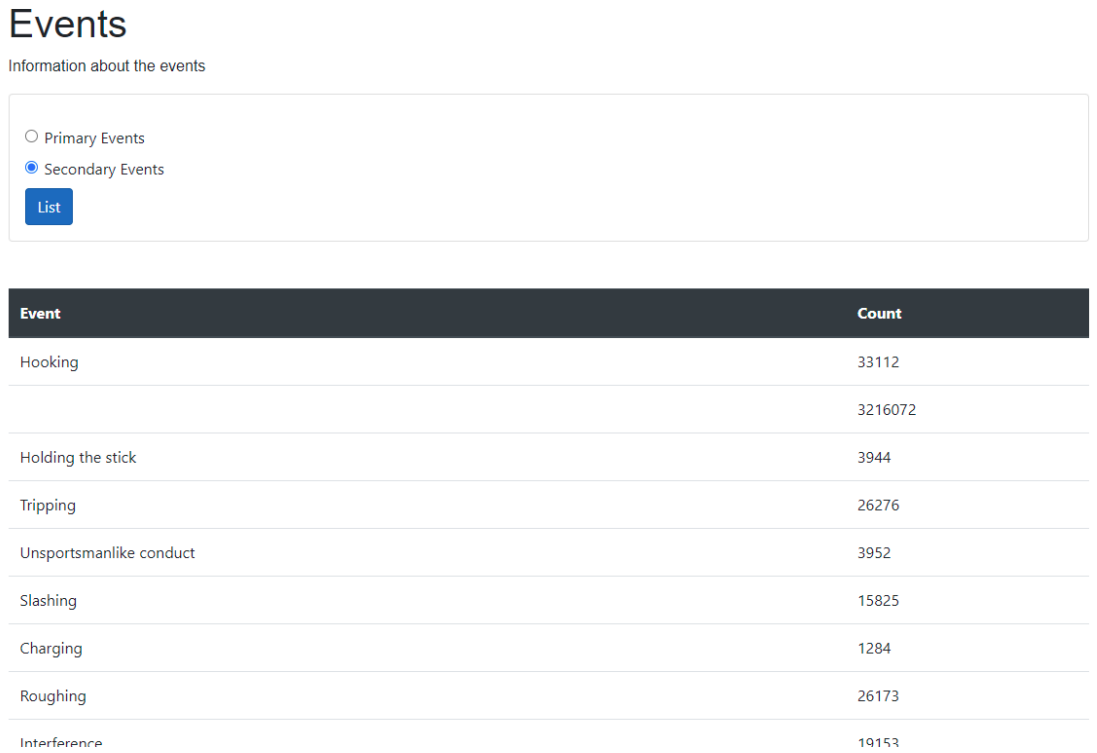
Az elsődleges események a főbb mozzanatai egy játéknak. Megtalálható minden említésre méltó történés: ütközés, lövés, kihagyott lövés, lövés blokkolása stb.

A másodlagos események az elsődlegesek kiegészítései, azonban ezek az esetek többségében nem adottak. Ilyen például, ha a „mérkőzés megállítása” esemény mellé megadják, hogy azért történt, mert túl sok játékos tartózkodott a jégen, vagy ha egy büntetés mellé megadják az okát, például könyöklés.

Az elsődleges és másodlagos események csoportosítva lettek, és mindegyik mellett megjelenik az előfordulásainak száma. Ezekre egy-egy példát a 8.8 és 8.9-es ábrán lehet látni.



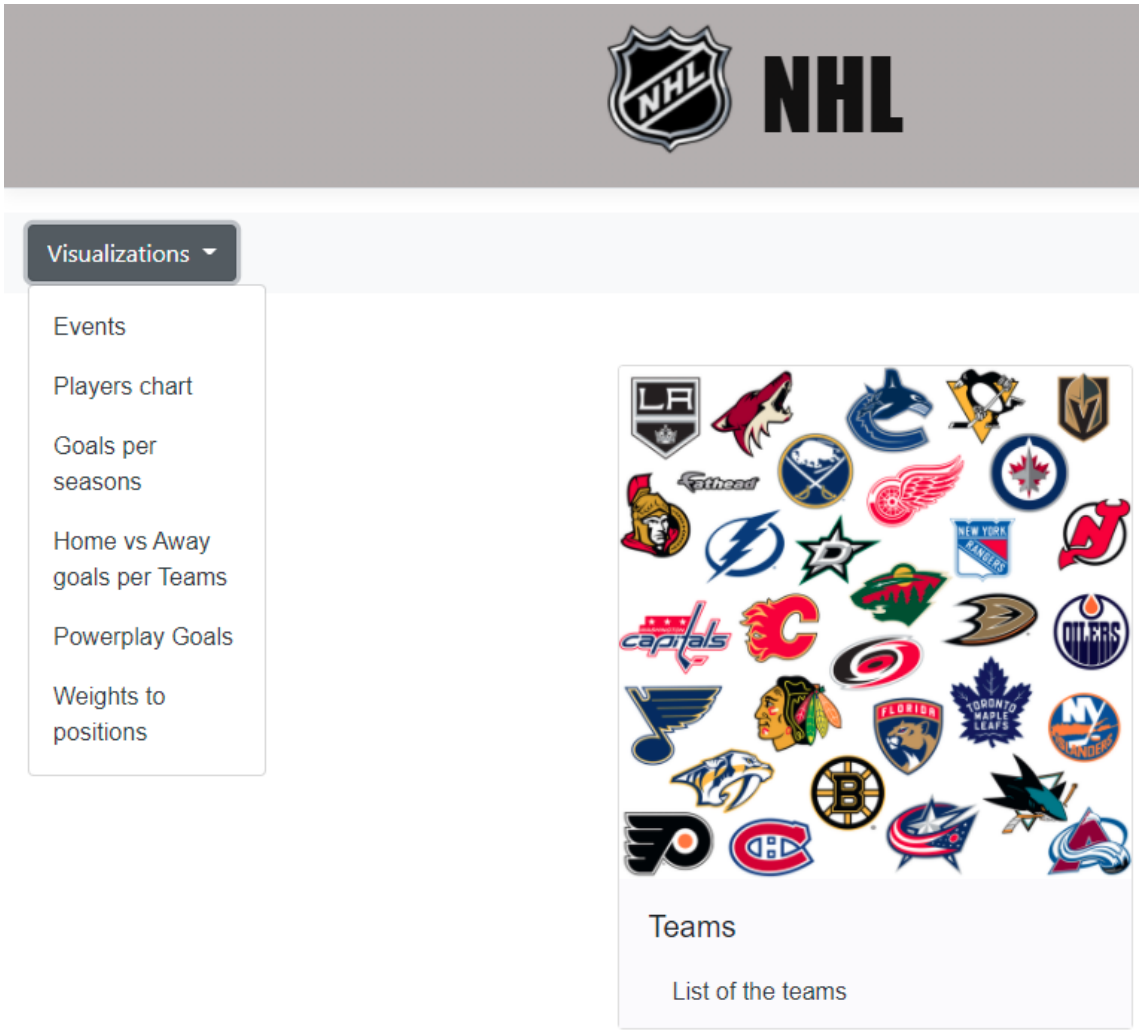
8.8 ábra: Elsődleges események



8.9 ábra: Másodlagos események

9. Vizualizációk és elemzésük

A vizualizációkat a főoldalról lehet elérni egy legördülő menü segítségével (9.1-es ábra). A grafikonokat ChartJS használatával készítettem, az oldalba ágyazott JavaScript kód segítségével.



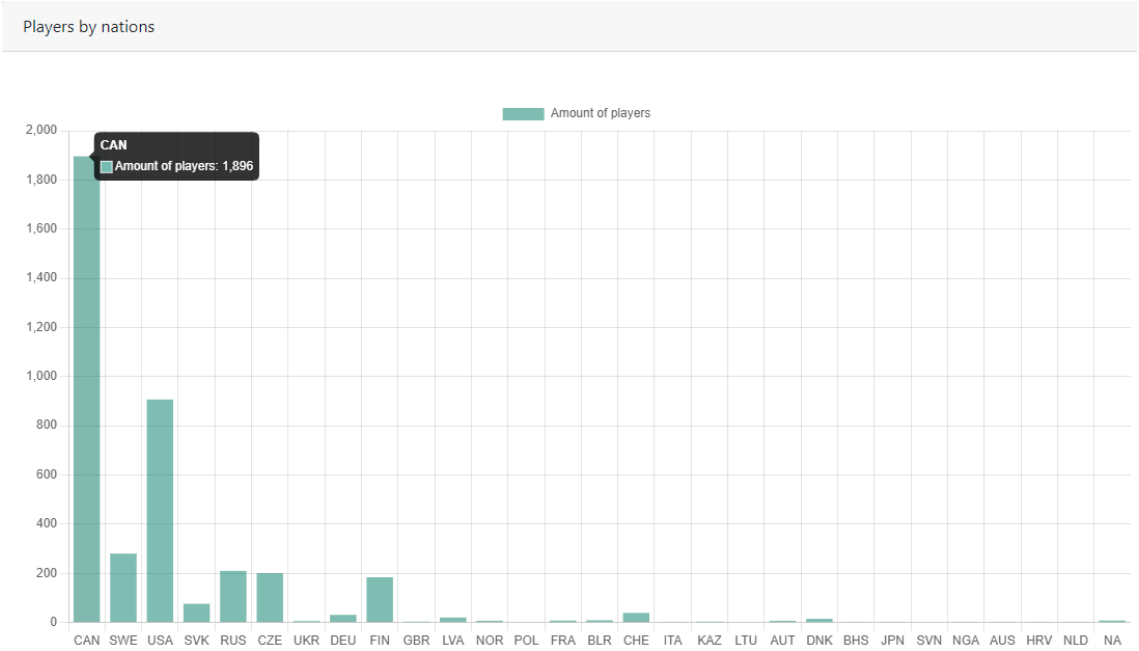
9.1 ábra: Vizualizációk elérése

9.1 Játékosok eloszlása nemzetiség szerint

Az NHL-ben a Föld minden tájáról származó játékosok játszanak, azonban ahogyan az a 9.2-es ábrán is megfigyelhető, körülbelül 6-7 ország adja a játékosok 95%-át. Ezek az országok: Kanada, Svédország, Amerikai Egyesült Államok, Szlovákia, Oroszország, Csehország, Finnország.

A fent említett helyek közül Kanada és az Amerikai Egyesült Államok külön kiemelkedik, hiszen csak Kanada egyedül lefedi a játékosok ~50%-át (1896 db játékos), az USA pedig 907 játékosnak szerepel származási helyként az adatbázisban.

Nyolc játékosnál nem szerepel információként a születési ország, a legkevesebb játékost pedig a Bahama-szigetek, Nigéria és Japán adták, fejenként egyet.



9.2 ábra: Játékosok nemzetiség szerint

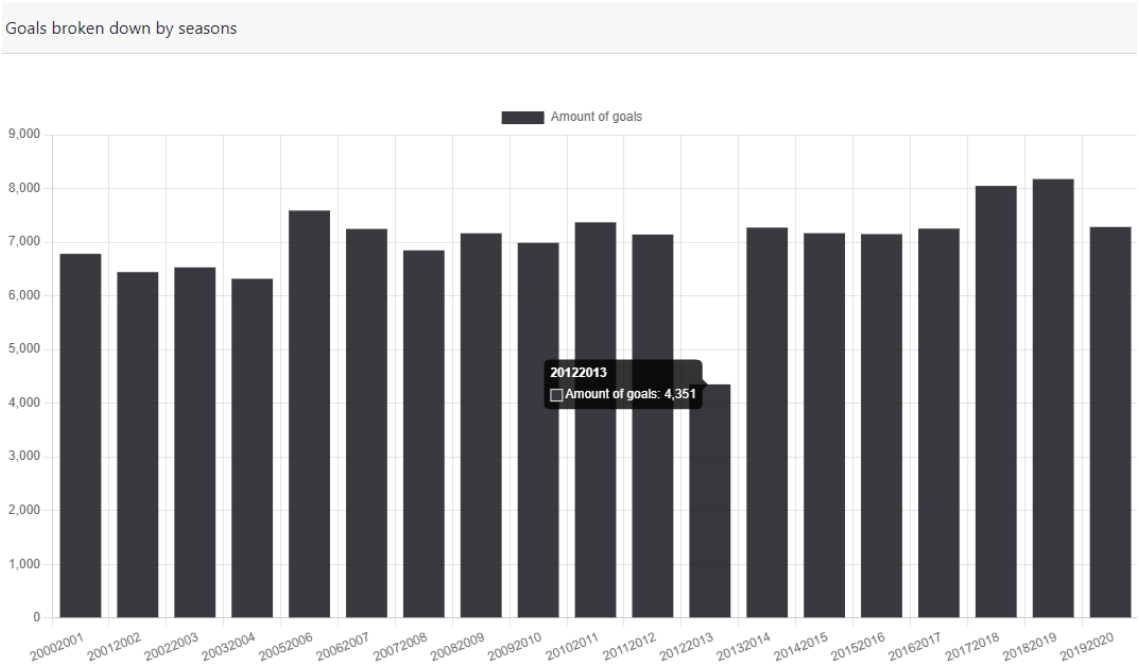
9.2 Gólok száma szezonokra bontva

A következő vizualizáció a lőtt gólok számára vonatkozik, idényekre lebontva. Ahogy azt a 9.3-as ábra is mutatja, nagyjából kiegyensúlyozottak az eredmények.

Egyedül a 2012-2013-as szezon marad el a többitől, azonban ennek meg van az oka: a játékok között erre az időszakra rákeresve látható, hogy összesen csupán 806 mérkőzést játszottak a csapatok, az átlagos 1250-1350 helyett. Ennek az oka, hogy az adott időszakban október helyett januárban indult az idény, emiatt minden csapat 48 mérkőzést játszott (82 helyett).

A fentieket és a többi év eredményeit figyelembe véve kiszámolható, hogy ha teljes szezont játszottak volna a csapatok, közel azonos eredmény született volna.

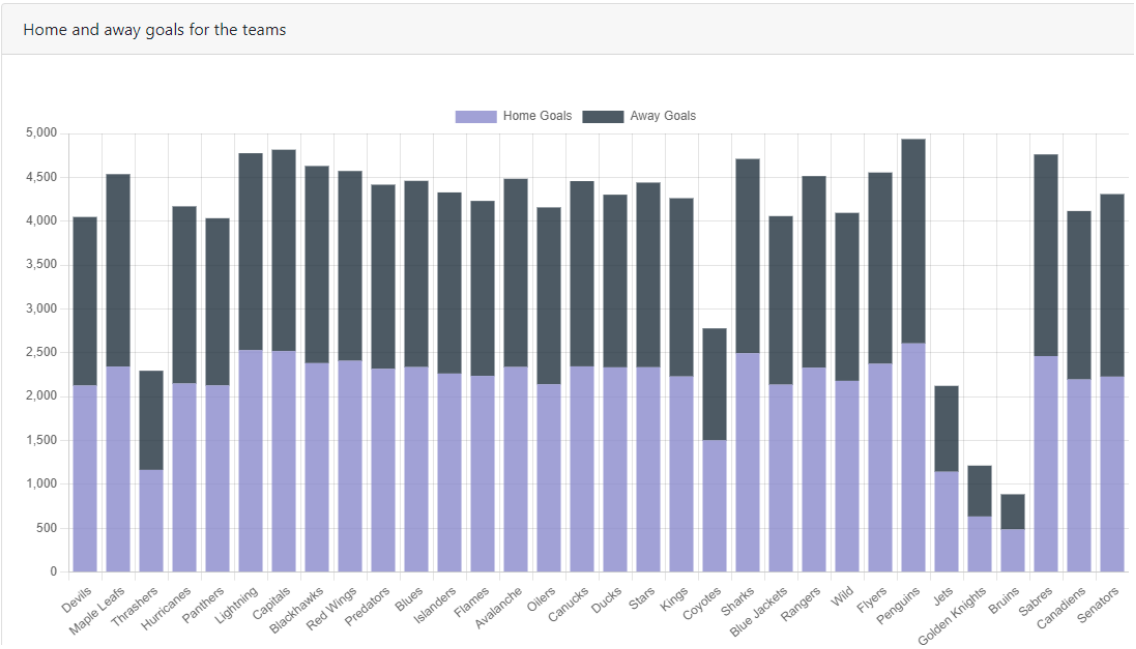
A legeredményesebb a 2018-2019-es év volt, 8175 lőtt góllal.



9.3 ábra: Gólok idényekre bontva

9.3 Csapatok hazai és vendég góljai

Általános elképzelés, hogy a csapatok gyengébben teljesítenek idegenben, mint haza pályán. A 9.4-es ábrán látható halmozott oszlop diagramon megfigyelhető a gólok száma csapatokra lebontva. Leolvasható az összes gól, valamint jól látható a hazai és idegenben szerzett pontok közötti arány is.



9.4 ábra: Hazai és vendég gólok száma

Valóban elmondható, hogy kivétel nélkül az összes csapat kevesebb gólt lőtt vendég pályán, mint hazai környezetben. A két érték közötti különbség a legtöbb csapat esetében ~200. A legkisebb különbség az Atlanta Thrashers csapatánál figyelhető meg, csupán 35-tel szereztek kevesebb gólt idegenben játszott mérkőzések során, mint otthon.

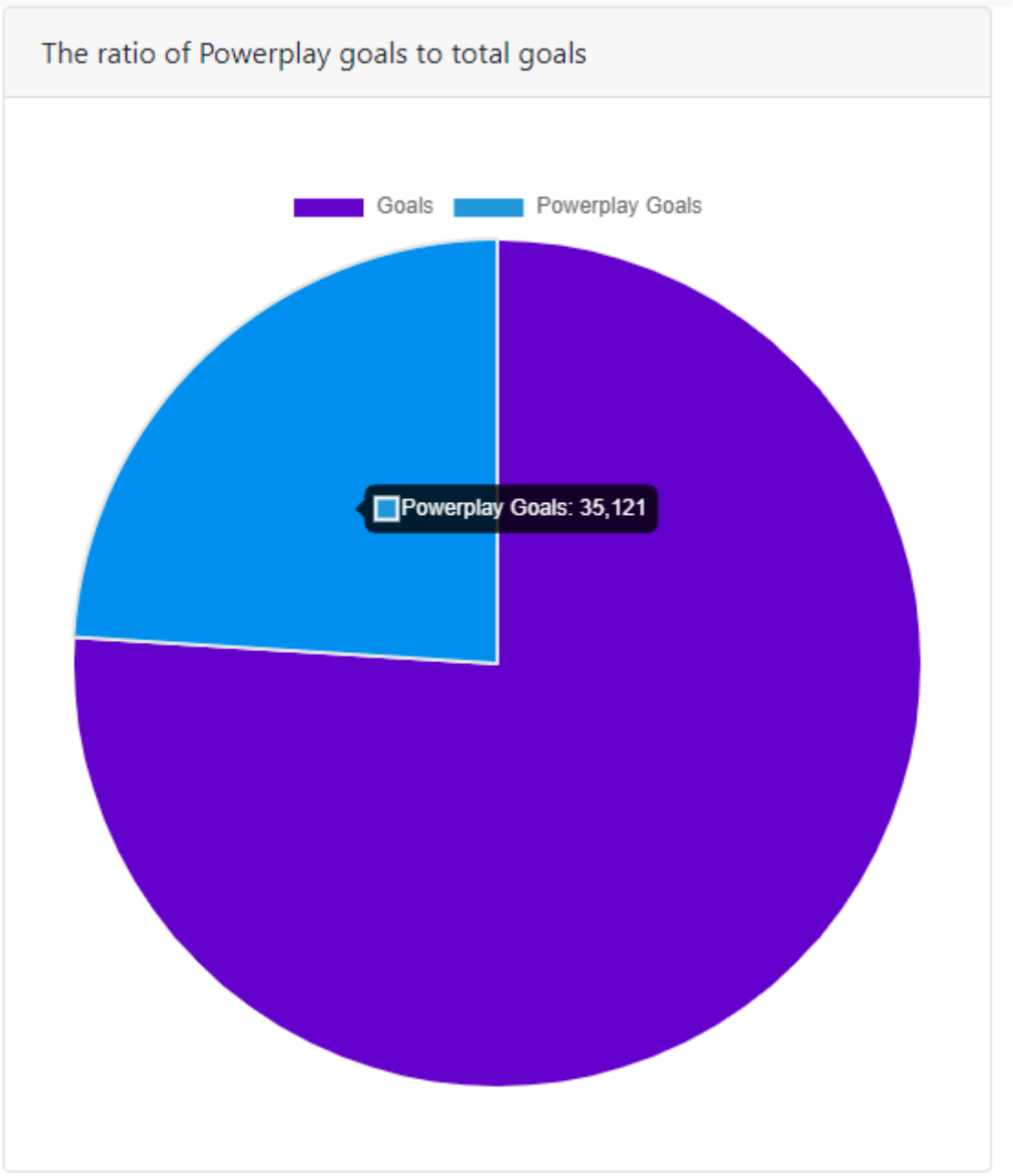
A legkevesebb gólt a Boston Bruins csapata szerezte: összesítve 887-et.

A legtöbb gólt pedig a Pittsburgh Penguins csapata lőtte, összesítve 4936-ot.

9.4 Gólok emberelőnyben

Abban az esetben, ha az egyik csapatból szabálytalanság miatt kiállítanak egy vagy több játékost legalább két percre, és ez idő alatt az emberelőnyben lévő csapat gólt szerez, azt emberelőnyös gólnak, vagy angolul „powerplay goal'-nak nevezzük. A többi sportággal szemben a jégkorongban nem ritkaság, ha egy játékost leküldenek a jégről néhány percre. Tekintve, hogy itt 6 játékos tartózkodik egyszerre a jégen egy csapatból (öt mezőnyjátékos + egy kapus), sokkal nagyobb jelentősége van egy kiállított embernek, mint például a labdarúgásban. Ezt bizonyítja a 9.5-ös ábra is: megfigyelhető, hogy az adatbázisban szereplő gólok közel negyedét emberelőnyben szerezték a csapatok.

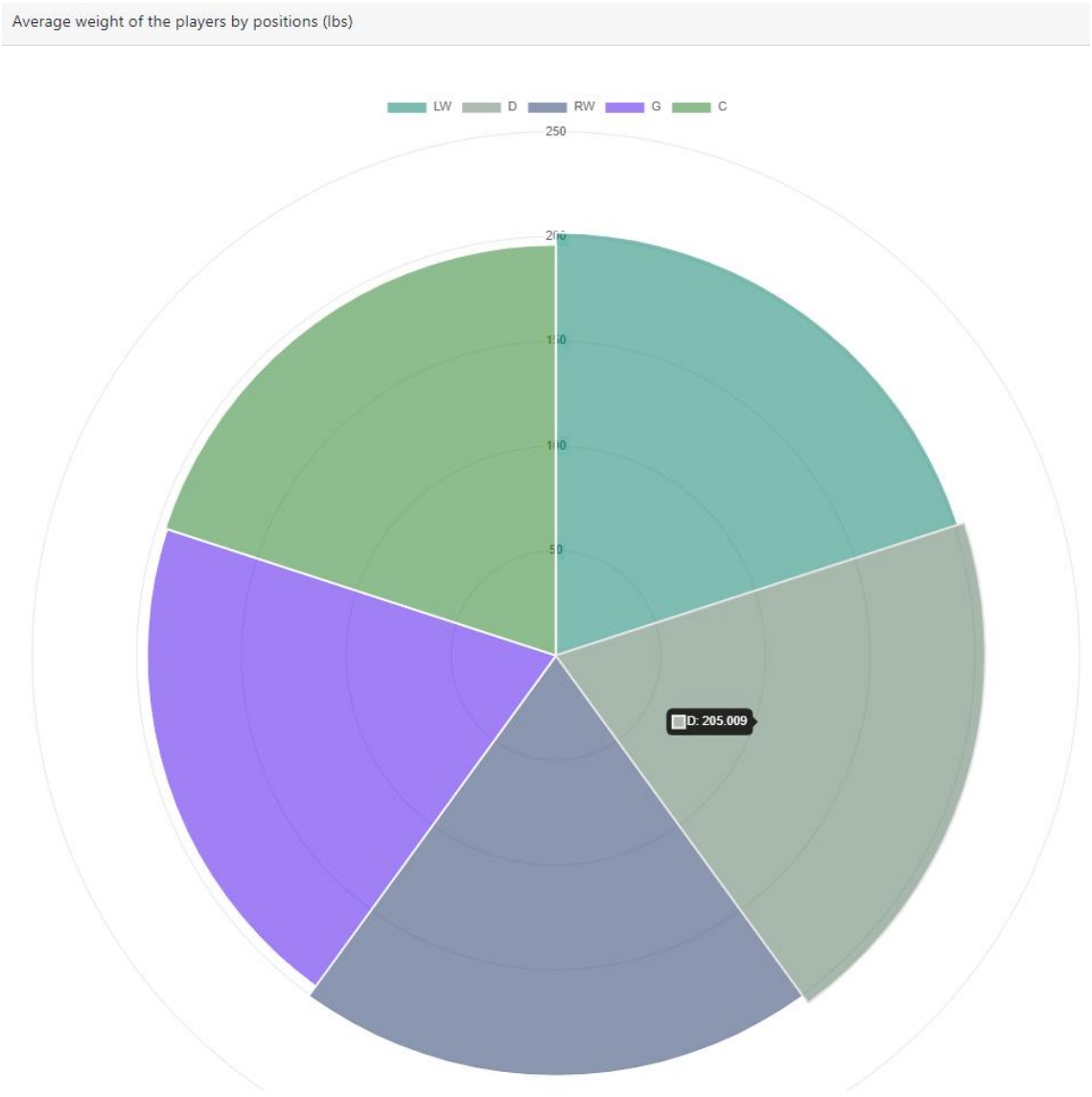
Szám szerint 35 121 és 111 180.



9.5 ábra: Emberelőnyben szerzett gólok aránya

9.5 Játékosok tömege és pozíciójuk közötti összefüggés

A rendelkezésre álló adatok alapján megvizsgáltam, hogy van-e bármilyen összefüggés a játékosok súlya és az elsődleges pozíciójuk között. Az ehhez kapcsolódó diagram a 9.6-os ábrán látható.



9.6 ábra: Pozíciók és a hozzájuk tartozó átlagsúly

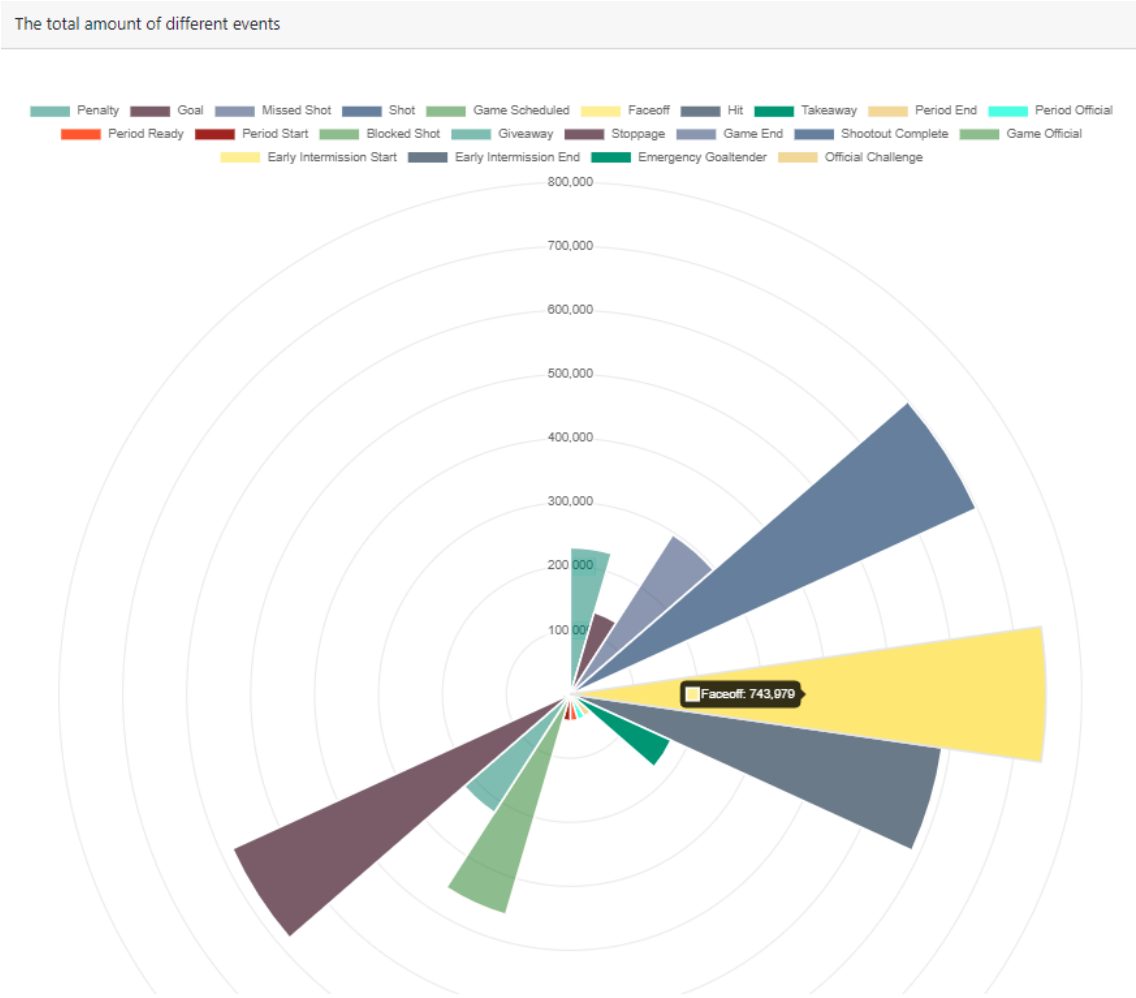
Mind az öt adatbázisban szereplő pozícióhoz kiszámoltam az ott játszó játékosok átlagsúlyát. Amint az a fenti ábráról is leolvasható, az adatok elég kiegyenlítettek, minimális eltérés van közöttük. A legnagyobb különbség a kapusok és a védőjátékosok között figyelhető meg, körülbelül 10 font, azaz 4.5 kilogramm.

Ezen adatok alapján kijelenthető, hogy nincs lényeges összefüggés a játékosok súlya és játszott pozíciója között, általánosságban elmondható, hogy egy játékos tömege 195 és 205 font, azaz 88 és 93 kilogramm között van.

9.6 Események előfordulása

Az utolsó vizualizáció a már fentebb is említett események előfordulására irányul. Míg a listázás esetében könnyen el lehet veszni a sorok között, vizuális formában könnyebb értelmezni és elemezni az adatokat.

Az elsődleges események előfordulását vizsgáltam, az ábra a 9.7-es ábrán látható.



9.7 ábra: Események gyakorisága

Megállapítható, hogy alapvetően egy mérkőzésen 4 esemény a leggyakoribb: „face-off”, lövés, ütközés és a játék megállítása. Ezek közül az első helyen a „face-off”, vagy magyarul „küzdelem/bedobás” áll. Ez az esemény azt takarja, amikor két játékos egymással szemben áll, és a játékvezető bedobja közéjük a jégkorongot az úgynevezett „bedobási pont” -ra és mindegyik játékos megpróbálja megszerezni. 743 979 db van belőle nyilvántartva az adatbázisban, ehhez 23 735 lejátszott mérkőzés tartozik.

Elmondható tehát, hogy egy mérkőzésen átlagosan több mint harmincszor dobják be a labdát.

A második leggyakoribb esemény a lövés, 698 365 előfordulással. A harmadik és negyedik helyen néhány ezres különbséggel az ütközések és a játék játékvezető általi megállítása állnak.

Összességében jól látszik, hogy egy jégkorong mérkőzést leginkább befolyásoló tényezők nagyon sok dologtól függenek, emiatt kifejezetten nagy a véletlen faktor a többi sporthoz képest. Nagyon nehéz előre megjósolni ezeknek az eseményeknek a bekövetkezését, azt, hogy a pálya mely részén következnek be (egy pályán például kilenc bedobási pont található) és azt, hogy milyen hatással lesznek a játék további részére.

10. Tesztelés

Az oldalt manuálisan teszteltem, emberi erőforrásokkal. A működést két böngészőben vizsgáltam meg: Brave és Google Chrome.

Az oldal minden funkciója az elvártaknak megfelelően működött, mindkét böngészőben zökkenőmentesen tudtam szemügyre venni a kívánt adatokat. A teszteléshez tartozott az oldal reszponzívitásának vizsgálata is, két különböző méretű és felbontású kijelzőt használtam, valamint az ablakok méretét is variáltam. Az egyik kijelző egy BENQ GW2255 21,5 col-os monitor, a másik pedig egy 29 col-os LG UltraWide monitor volt.

Az oldal minden esetben alkalmazkodott a környezethez, az ábrák, képek mérete arányosan változott.

Egy probléma merült fel a tesztelés során, a mérkőzések eseményeivel kapcsolatban. Tekintve, hogy viszonylag nagy tábla tartozott hozzá (több mint 4 millió sor), a ráépülő listák és a vizualizáció sajnos kicsit lassan töltött be. A felhasználó élményt nagyban rontja, ha akár 1-2 percet is várni kell egy oldal betöltésére, emiatt ezt orvosolni kellett. A megoldás az lett, hogy az oldal betöltésekor történő számítás és csoportosítás helyett a szükséges lekérdezéseket egy nézetbe töltöttem, és az oldal azzal dolgozott a továbbiakban.

11. Fejlesztés összefoglalása, további fejlesztési lehetőségek

Az előző fejezetekben bemutattam egy adattárház, és egy hozzá tartozó webalkalmazás fejlesztését. A követelmények és funkciók meghatározása után egy olyan oldalt sikerült megvalósítani, amely értékes információkkal szolgál a sport kedvelőinek, vagy akár bizonyos elemzésekhez járulhat hozzá. Véleményem szerint az oldal erőssége abban rejlik, hogy nem csak a legfrissebb adatokhoz lehet hozzáférni, hanem visszamenőleg sok év anyaga található meg gyorsan, egy helyen, egyszerű és letisztult módon.

A fejlesztés jelenleg csak az adatok és vizualizációk megjelenítésére, valamint az adatok közötti – szűrési feltételek megadása mellett – keresésre korlátozódott. Ennek a célnak teljes mértékben megfelel, azonban mint minden rendszert, ezt is lehet tovább fejleszteni.

Egy fejlesztési irány lehet például a vizualizációk interaktívvá tétele. Ez alatt azt értem, hogy a felhasználó ne csak az előre elkészített diagramokat lássa, hanem lehetősége legyen kedve szerint – nyilván bizonyos határokon belül – kiválasztani a megjeleníteni kívánt adatokat. Valamint amennyiben szükség van rá, az általa összeállított adatokat más diagramtípus segítségével tudja megvizsgálni, értelmezni.

12. Összefoglalás

A szakdolgozatomban sikeresen elkészítettem egy adattárházat, és egy ráépülő webalkalmazást, amely a National Hockey League utóbbi kb. 20 év mérkőzéseinek és a hozzájuk tartozó adatoknak megértését és elemzését segíti. Sikerült egy olyan alkalmazást fejlesztenem, amely segítségével lehetőség nyílik évekre visszamenően megvizsgálni a mérkőzések, csapatok eredményit különféle helyzetekben.

A webalkalmazás elkészítése során lehetőségem volt megtapasztalni egy fejlesztés lépéseit a tervezéstől egészen a megvalósításig. Annak érdekében, hogy egy stabil és naprakész alkalmazás jöjjön létre, munkám során törekedtem a legmodernebb eszközök használatára. A fejlesztéshez szükséges ismereteket főként az interneten elérhető hivatalos dokumentációkból, valamint oktató videókból tanultam meg. Az a tudás és tapasztalat, amelyet a szakdolgozat készítése közben szereztem, egész biztosan a jövőben is hasznomra lesz.

Summary

In my thesis, I've successfully created a data warehouse, and a web application based on it, that helps to understand and analyze the data of the National Hockey League from the last 20 years or so. I've managed to develop an application which provides the opportunity to examine the details of the games and teams in different scenarios throughout the years.

During the creation of the application I've had the opportunity to experience a program development from the planning phase to the implementation. In order to create a stabile and up-to-date software, I tried to use tools that are as modern as possible. I've collected the knowledge that was required to develop the application mainly from official product documentation and tutorial videos from the Internet. The knowledge and experience that I've gathered during my thesis will certainly benefit me in the future.

Irodalomjegyzék

[1] Bánné Varga, G.: Az adattárház-készítés technológiája. *Typotex Kiadó*, 2012

[2] Brackett, M. H.: The Data Warehouse Challenge. *Wiley*, 1996

[3] Fajszi, B., Krauth, P.: Üzleti tudás az adatok mélyén. *Budapesti Műszaki és Gazdaságtudományi Egyetem*, 2004

[4] Inmon, W. H.: Building the Data Warehouse. *Wiley*, 2005

[5] Kaucsukné Bruckner, L., Kiss T.: Bevezetés az üzleti informatikába. *Akadémiai Kiadó Zrt.*, 2009

[6] Kimball, R., Ross, M.: The Data Warehouse Toolkit – Second Edition. *John Wiley & Sons, Inc.*, 2002

[7] Kővári, A.: Üzleti intelligencia (Business Intelligence, BI) fogalma,

(https://www.biprojekt.hu/Uzleti-intelligencia-Business-Intelligence-BI.htm), utoljára megtekintve: 2021.05.08

[8] Luhn, H. P.: A Business Intelligence System. *IBM Journal*, Vol. 2, 1958, pp. 314

[9] Orfali-Harkey-Edwards: The Essential Client/Server Survival guide. *Wiley*, 1999

[10] Power, D. J.: A Brief History of Decision Support Systems

(http://dssresources.com/history/dsshistory.html), utoljára megtekintve: 2021.04.29

[11] Sidló, Cs.: Összefoglaló az adattárházak témaköréről,

(http://scs.web.elte.hu/Work/DW/adattarhazak.htm), utoljára megtekintve: 2021.04.29

[12] Tarcsi, Á., Hortváth, Gy.: Webes alkalmazások tervezése, fejlesztésének menete,

(http://ade.web.elte.hu/02_Webes_alkalmazasok_tervezese.pdf), utoljára megtekintve: 2021.05.09

[13] Felhasznált adatok forrása: https://www.kaggle.com/martinellis/nhl-game-data?select=game_plays.csv