



Kisvárdai SZC Fehérgyarmati Petőfi Sándor Technikum
Szoftverfejlesztő és -tesztelő szakma

ScoreSchool projekt

SZAKDOLGOZAT

Konzulens tanár:

Lakatos Sándor

Berki Balázs

Készítette:

Ármós Szabolcs

Gáspár Marianna Dominika

Sankó Balázs

Tartalomjegyzék

Bevezető	3
Weboldal felépítése	4
Főoldal	4
Keresés	5
Adatkezelés	5
Statisztika	6
Tervezés	7
Photoshop	7
Canva	8
Draw.io	8
Technológiai megvalósítás	8
Adatbázis	8
Backend – Node.js	15
Frontend – Angular	17
Programtervezési minta – MVC	20
Bootstrap	21
Verziókezelés	22
Applikáció	23
Tesztelés	23
Tapasztalatok és csapatmunka	24
Jövőkép	26
Ábra jegyzék	28
Felhasznált irodalom	29

Bevezető

Projektünk ötletének alapja az, hogy mindannyian lelkes futballrajongók vagyunk és Szabolcs és Balázs aktívan űzik a sportot. A hazai és nemzetközi bajnokságokat is rendszeresen követjük, amelyek eredményeit több online weboldalon is megtalálhatjuk. Ezek mind információkat szolgálnak a meccsekről, csapatokról és játékosokról. Innen jött az ötlet egy olyan weboldalhoz, amely kifejezetten az iskolai tornák adatait tárolná.

Iskolánkban, a Petőfi Sándor Technikumban, régóta hagyomány a Falcsik Ferenc Emléktorna, amelyet néhai testnevelés tanárunkról, Falcsik Ferencről neveztek el. A tornára bárki jelentkezhet saját csapattal az iskola tanulói közül, így akár osztályok vagy vegyes csapatok is nevezhetnek. Ez remek lehetőség arra, hogy a diákok összemérjék tudásukat és versenyezzenek különböző díjakért (legjobb kapus, gól király, legjobb játékos). A torna január elején kezdődik és március elejéig tart. Minden szerdán 2-2 csapat játszik egymás ellen és a torna során mindegyik csapat játszik mindegyik csapattal. A meccsek alapján pontokat gyűjthetnek (győzelem: 3, döntetlen: 1, vereség: 0), amelyek alapján a torna végén rangsorba kerülnek. Ha két csapat ugyan annyi pontot szerzett, akkor vagy az dönt, hogy az egymás ellen játszott meccsen ki nyert vagy (döntetlen esetén) az, hogy kinek jobb a gólaránya (kapott gólokból kivonjuk a rúgott gólokat). Viszont az eredmények vezetése eddig papír alapon történt, amelynek több hátránya is van:

1. Az adatok bármikor elveszhetnek, hiszen csak egy lapon szerepelnek.
2. Az eredmények kézi vezetése sok időt igényel.
3. Nehézségeket okozhat a meccsek vezetése és ezekből kimutatásokat készíteni (például: ki lett a gólkirály, körbeverés esetén melyik csapat rendelkezik a legtöbb ponttal, vagy melyiknek jobb a gólaránya).
4. A tornák eredményei csak a lebonyolítást végző tanárnak vagy bírónak vannak meg, így a diákok nem nézhették vissza teljesítményüket.

Weboldalunk nem csak egyszerűsíti, de korszerűsíti a torna szervezését. A weboldal nem csak arra nyújt lehetőséget, hogy saját tornáink adatait mentésük, hanem más iskolák eseményeit megtekinthetjük, így egy olyan rendszer jönne létre, ahol összekapcsolja a különböző iskolák sportéletét. A weblapunk lehetőséget nyújt statisztikai kimutatások készítésére is. Például, hogy melyik játékos rúgta a legtöbb gólt, melyik csapat rendelkezik a

legtöbb ponttal, gólaránnyal vagy hány kapott és rúgott góljuk van. A kimutatások növelhetik a versenyszellemet, hiszen így a diákok folyamatosan nyomon követhetik a fejlődésüket.

Összegezve, weboldalunk nem csak a tornák vezetésére nyújt korszerű megoldást, de a diákokat is jobb teljesítményre ösztönzi. Ezenkívül olyan közöl platformot biztosít, ahol nemcsak saját, de más iskolák sporteseményeit is követhetjük.

Weboldal felépítése

Weboldalunk 1 főoldalból és 3 aloldalból áll, amelyek mind más funkciókkal rendelkeznek. Az aloldalak csak bejelentkezés után érhetők el, és a navigációs sáv segítségével tudunk közöttük egyszerűen váltani.

Főoldal

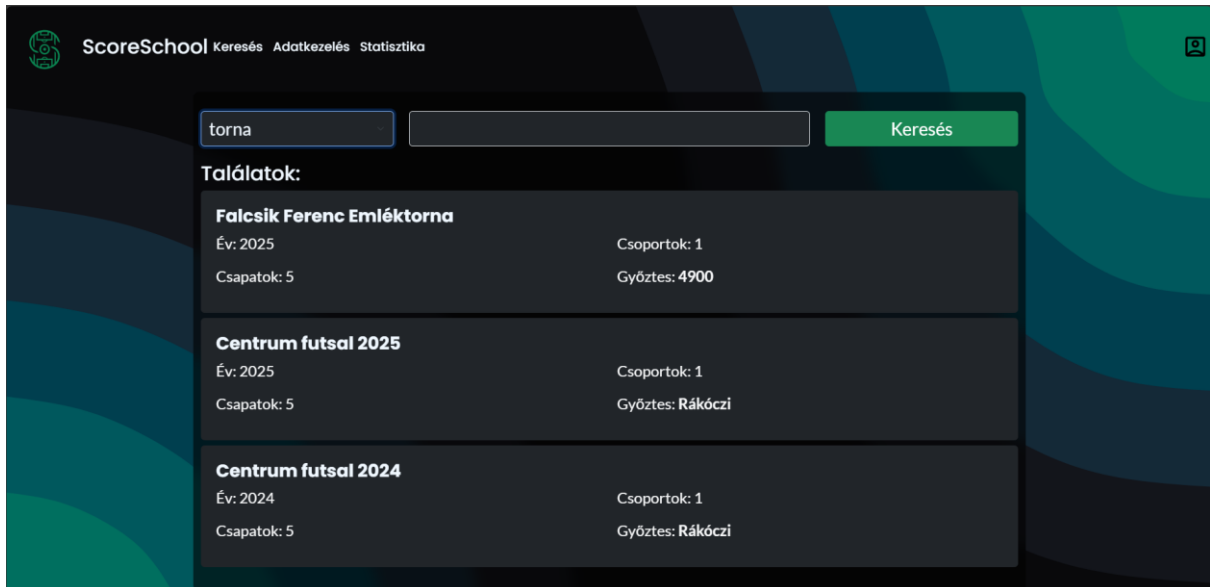
A weboldal megnyitásakor először a főoldal köszönti a felhasználót. A weblap alján található sávon a felhasználó megismerkedhet a legfontosabb funkciókkal. A „Gyere kezdjünk” gombra kattintva lehetősége van bejelentkezni, vagy ha még nem rendelkezik fiókkal, akkor regisztrálhat. A reszponzív dizájnnek köszönhetően oldalunk nem csak számítógépen, hanem tableten és mobilon is elérhető és használható.



1. ábra

Keresés

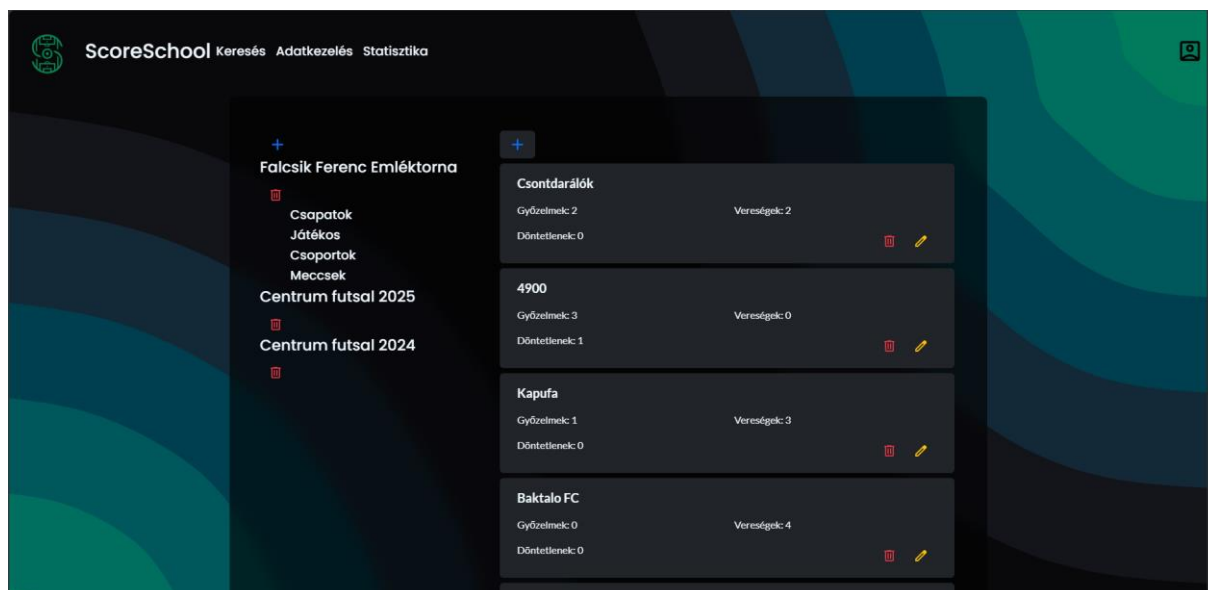
A keresés lap lehetőséget nyújt a felhasználónak arra, hogy az adatbázisban szereplő adatok között böngésszen. A keresés 3 kategória szerint lehetséges: torna, csapat, játékos. A felhasználók egy legördülő listából választhatják ki melyik kategória adataira kíváncsiak és a weblap azonnal frissül e szerint. A keresőmező arra szolgál, hogy egy konkrét adatra keressünk a neve alapján.



2. ábra

Adatkezelés

Bejelentkezés után erre a felületre navigálja át az oldal a felhasználót, ahol az általa feltöltött adatokat találhatja. Itt hozhat létre új tornákat, rögzítheti a meccsek, csapatok, csoportok vagy játékosok eredményeit, vagy a meglévő adatokat módosíthatja és törölheti. A weblap a műveletek után azonnal frissül. Az adatok kezelését egy REST API-n keresztül működő backend oldali szerver biztosítja, amely aszinkron módon egyszerre több kérésre is tud reagálni.



3. ábra

Statistika

Ez az oldal szolgál a statisztikai kimutatások készítésére. A felhasználó az általa feltöltött tornák közül választva jelenítheti meg a megfelelő adatokat, majd kiválaszthatja, hogy pontosan milyen elemzést szeretne készíteni, például, hogy a csapatok pontjait, gólarányait, kapott vagy rúgott góljait szeretné megjeleníteni vagy a játékosok között szeretne egy ranglistát gólok alapján. A weboldal az adatokat egy oszlop diagrammon jeleníti meg, amelyet a felhasználó .svg (skálázható vektorgrafika) kiterjesztésben letölthet, így az minden méretben jól olvasható és éles.

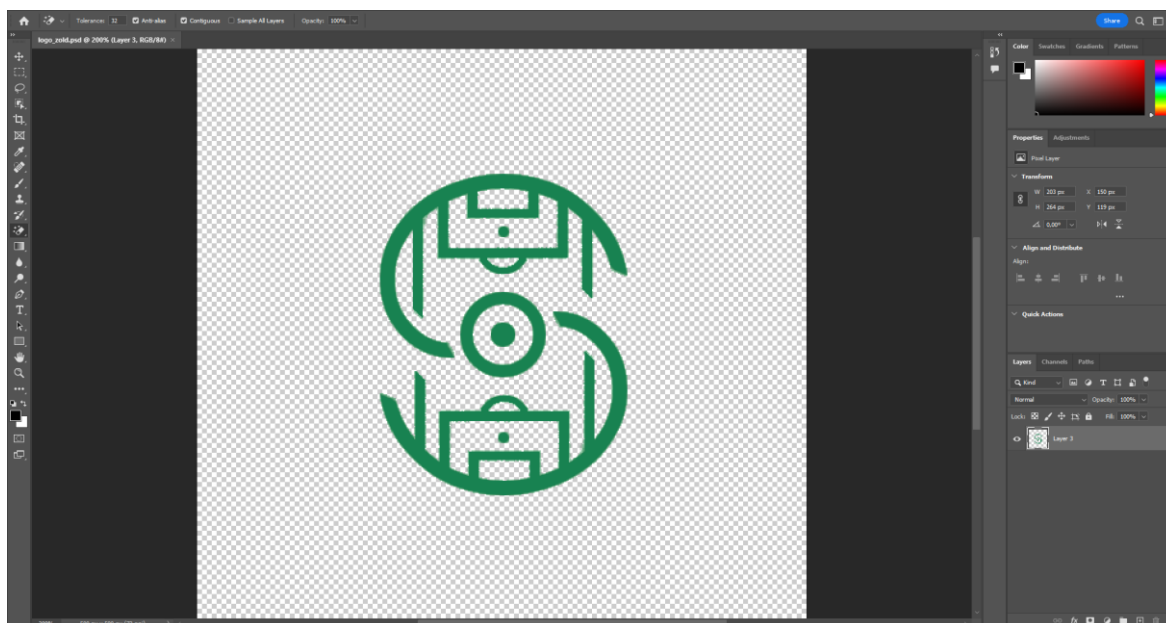


4. ábra

Tervezés

Photoshop

A weboldalunkon található képek szerkesztését Photoshoppal végeztük el. A képek többsége olyan weboldalokról származnak, amelyek szerzői jogdíjmentes képeket kínálnak (például Undraw). Ahhoz, hogy minden eszközön tökéletesek legyenek a grafikák, törekedtünk arra, hogy .svg formátumban használjuk őket. Amelyeknél erre nem volt lehetőség azt Photoshop használatával átméreteztük és eltávolítottuk a háttérüket. Logónk is ebben az alkalmazásban készült, amelyet először digitálisan megrajzoltunk, majd átalakítottuk .svg-vé.



5. ábra

Canva

A weboldal megtervezéséhez a Canva grafika tervezési weboldalt használtuk, amelyen nem csak a különböző elemek elhelyezkedését próbálhattuk ki, hanem különböző színeket és betűtípusokat is. Ennek köszönhetően gyorsan és egyszerűen tervezhettük meg a weboldalunk elemeinek elrendezését.

Draw.io

Az adatbázisunkról külön dokumentációt készítettünk, amelyhez található egy UML (Unified Modeling Language – Általános célú modellező nyelv) és egy ER (Entity-Relationship model – Egyed-kapcsolat modell) diagram is a felépítéséről. Ezek arra szolgálnak, hogy a táblák és a közöttük lévő kapcsolat jól olvasható és könnyen értelmezhető legyen.

Technológiai megvalósítás

Adatbázis

Adatbázisunk MySQL alapú, amelyet a phpMyAdmin oldalán menedzstünk.

A MySQL a legnépszerűbb nyíltforráskódú, relációs adatbázis-kezelő szerver. Az 1990-es évek közepén fejlesztették ki, és azóta is népszerű, hiszen ingyenes, gyors és erős. A relációs

adatbázis annyit jelent, hogy az adathalmazban tárolt adatokat a relációs algebrai alapl műveletekkel szűrjük (unió, metszet, különbség). A MySQL elnevezésben az SQL (Structured Query Language – Strukturált lekérdező nyelv) nyelvre utal, amely más relációs adatbázis-kezelők (Oracle, MariaDB) alapja is.

Adatbázisunk elkészítésekor a figyelembe vettük a normalizáció szabályait, így tudatosan elkerültük a felesleges adat ismétlődést, a null értékeket és minden adattábla saját primary key-el rendelkezik.

Adatbázisunk tervezését 8 fő lépésre osztottuk fel, amelyek arra szükségeseket, hogy lássuk, milyen adatokra lesz szükségünk és hogyan kellene ezeket a legoptimálisabban tárolnunk.

1. Lépés: Probléma meghatározása

- a. Milyen adatokra lesz szükségünk: hány gólt lőttek a játékosok, a csapatok hány meccset nyertek meg, mikor játszották le a meccseket, a profilok milyen email címhez tartoznak, kik nyerték meg a tornákat, a különböző csoportokba melyik csapatnak lett a legtöbb pontja
- b. Milyen különböző entitások lesznek: torna, meccs, csoport, csapat, játékos, profil
- c. Milyen kapcsolat lesz közöttük: egy tornán több csoport is részt vesz, egy csapat több tornán is részt vehet és egy tornán több csapat is játszik, egy csoportba több csapat is van, egy csapat több meccsen is részt vesz, de egy meccsen csak két csapat játszik, egy játékos csak egy csapatban játszhat, de egy csapatban több játékos is van, egy profilhoz több csapat is tartozik és egy profil több tornát is feltölthet

2. Lépés: Entitások azonosítása:

- a. Játékos (azonosító: id)
- b. Csapat (azonosító: id)
- c. Meccs (azonosító: id)
- d. Profil (azonosító: id)
- e. Csoport (azonosító: id)
- f. Torna (azonosító: id)

3. Attribútumok meghatározása:

profil tábla

- id: Egyedi azonosító a profil számára (auto-increment).
- nev: A profilhoz tartozó felhasználónév
- email: A profilhoz tartozó e-mail cím, egyedi és nem lehet üres.

- jelszo: A profilhoz tartozó jelszó, amely nem lehet üres.

csapat tábla

- id: Egyedi azonosító a csapat számára (auto-increment).
- tornaid: A torna id-ja, amelyen a csapat részt vesz.
- profilid: A csapatot feltöltő profil id-ja.
- gyozelmek: A csapat győzelmeinek száma.
- veresegek: A csapat vereségeinek száma.
- dontetlenek: A csapat döntetleneinek száma.
- csapatneve: A csapat neve.

jatekos tábla

- id: Egyedi azonosító a játékos számára (auto-increment).
- csapatid: A csapat id-ja amibe a játékos tartozik.
- golokszama: A játékos által szerzett gólok száma.
- sargalapok: A sárgalapjainak száma.
- piroslapok: A piroslapjainak száma.
- nev: A játékos neve.
- pozicio: A játékos pozíciója.

meccs tábla

- id: Egyedi azonosító a mérkőzés számára (auto-increment).
- tornaid: A torna id-ja, amely során a meccset lejátszották.
- meccstipusa: A meccs típusa (elődöntő, döntő, barátságos)
- csapat1: Az első csapat id-ja.
- csapat2: A második csapat id-ja.
- cs1gol: Az első csapat által szerzett gólok száma.
- cs2gol: A második csapat által szerzett gólok száma.
- datum: A meccs dátuma.

torna tábla

- id: Egyedi azonosító a torna számára (auto-increment).
- profilid: A tornát létrehozó profil id-ja.
- tornaneve: A torna neve.
- ev: A torna éve.
- csoportokszama: A tornán szereplő csoportok száma.
- csapatokszama: A tornán szereplő csapatok száma.
- gyoztescsapat: A torna győztes csapata.

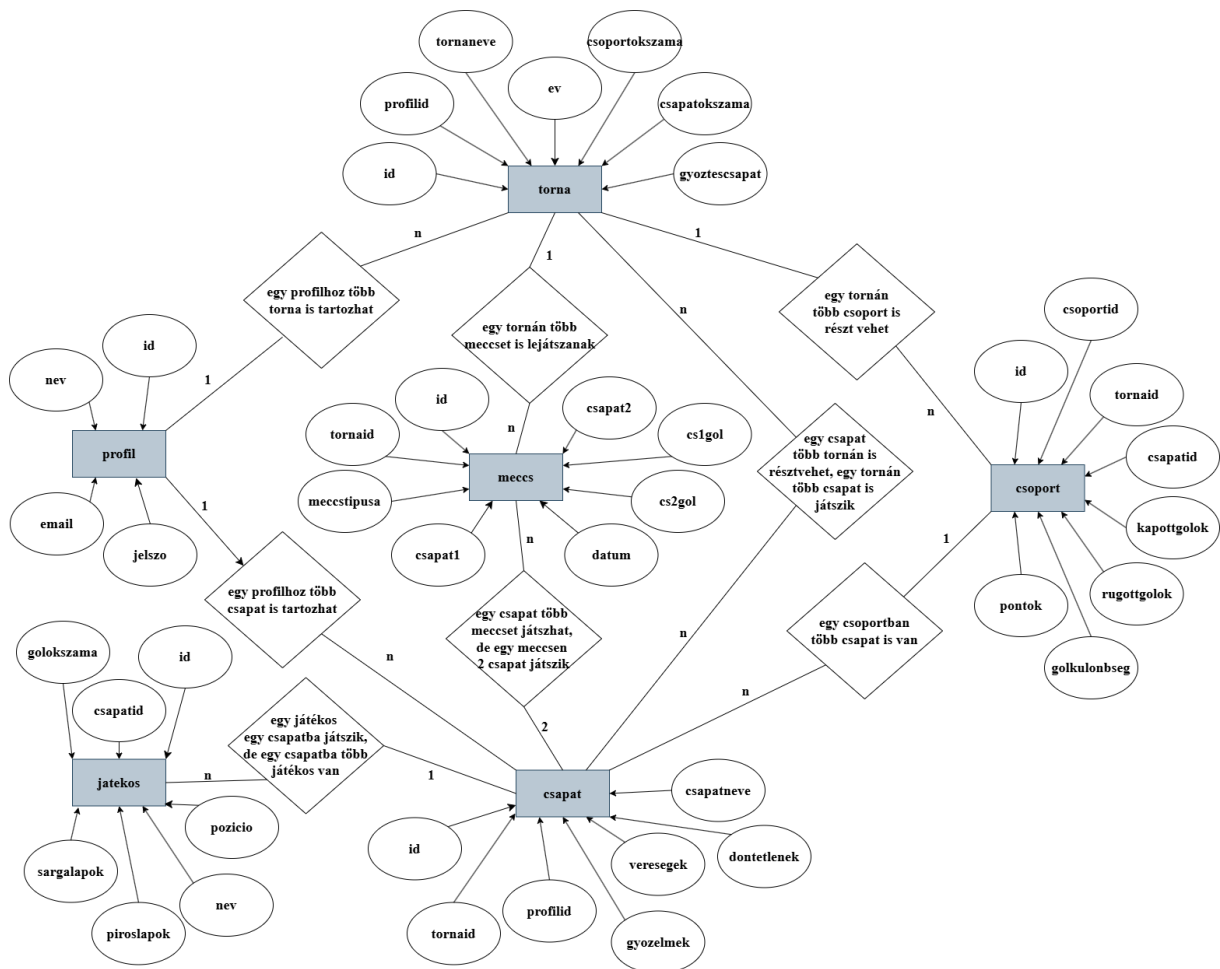
csoport tábla

- id: Egyedi azonosító az egyes adatoknak (auto-increment)
- csoportid: egy azonosító, ami összekapcsolja az egy csoportba tartozó csapatokat
- tornaId: A csoportot tartalmazó torna azonosítója.
- csapatid: A csoportban játszó csapat azonosítója.
- kapottgolak: A csapat által kapott gólok száma a csoportban.
- rugottgolak: A csapat által szerzett gólok száma a csoportban.
- golkulonbseg: A csapat gólkülönbsége a csoportban (szerzett gólok - kapott gólok).
- pontok: A csapat által szerzett pontok száma a csoportban (győzelem = 3, döntetlen = 1).

4. Lépés: Kapcsolatok azonosítása:

- torna és csoport (1:N): egy tornán több csoport is részt vesz
- torna és csapat (N:N): egy csapat több tornán is résztvehet és egy tornán több csapat is játszik
- csoport és csapat (1:N): egy csoportba több csapat is van
- meccs és csapat (N:2): egy csapat több meccsen is részt vehet, de egy meccsen csak két csapat játszik
- jatekos és csapat (N:1): egy játékos csak egy csapatba játszhat, de egy csapatba több játékos is van
- profil és csapat (1:N): egy profilhoz több csapat is tartozik
- profil és torna (1:N): egy profil több tornát is feltölthet

5. Lépés: ER diagram elkészítése:



6. ábra

6. Lépés: Normalizálás: Adatbázisunk készítésekor fokozottan figyeltünk arra, hogy adatmodellünk megfeleljen a normalizáció elveinek. Így tudatosan elkerültük a felesleges adat ismétlődést, minden mező csak egy értéket tartalmaz, nincsenek null (hiányzó értékek) és minden tábla saját primary key-el (elsődleges kulccsal) rendelkezik.
7. Lépés: Adatbázis sémájának elkészítése:

```

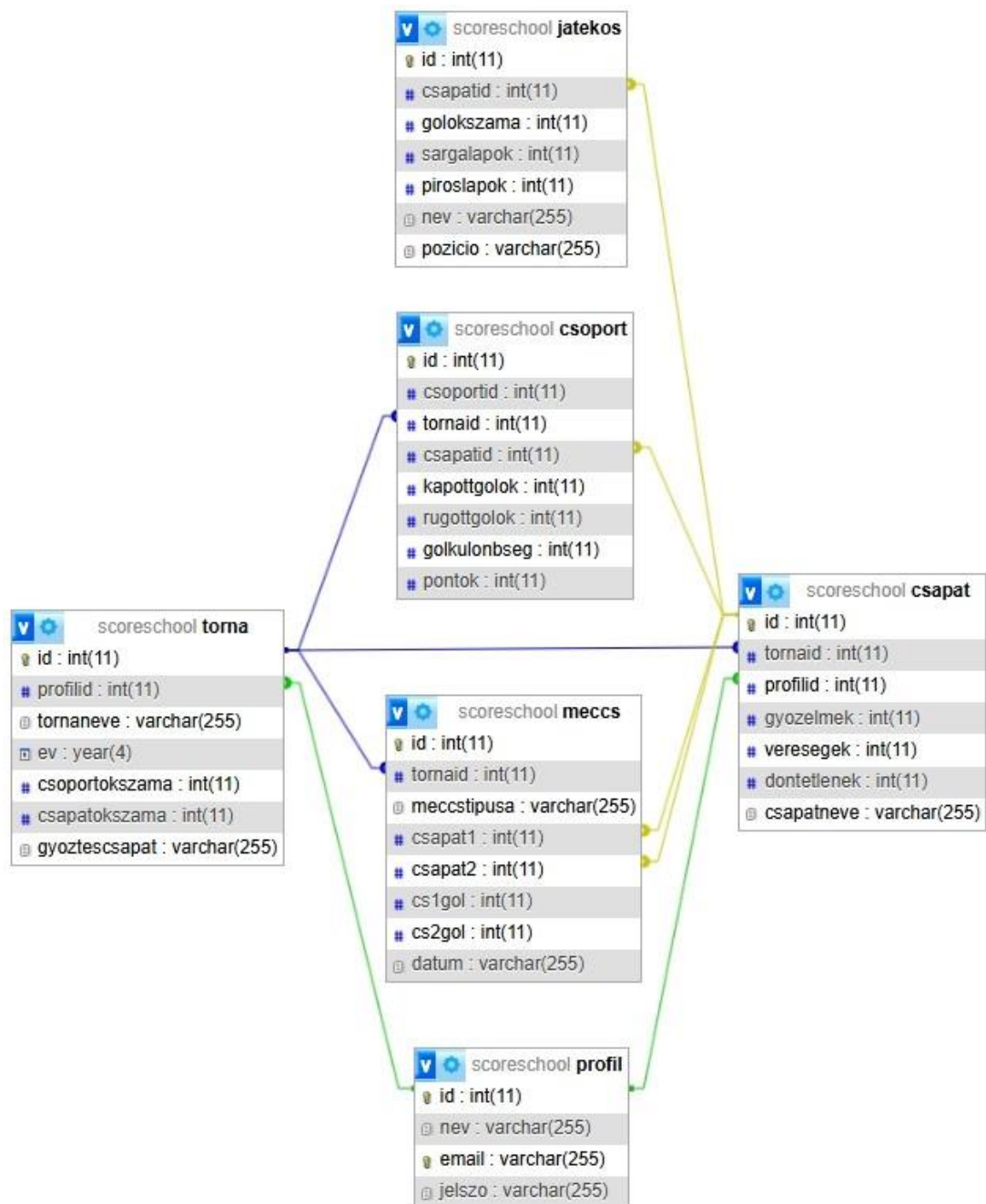
dokumentacio - adatbazis.sql

1 CREATE DATABASE scoreschool
2   CHARACTER SET utf8
3   COLLATE utf8_general_ci;
4
5 USE scoreschool;
6
7 CREATE TABLE profil (
8   id INT AUTO_INCREMENT PRIMARY KEY,
9   nev VARCHAR(255) NOT NULL,
10  email VARCHAR(255) NOT NULL UNIQUE,
11  jelszo VARCHAR(255) NOT NULL
12 );
13
14 CREATE TABLE torna (
15   id INT AUTO_INCREMENT PRIMARY KEY,
16   profilid INT NOT NULL,
17   tornaneve VARCHAR(255),
18   ev YEAR,
19   csoportokszama INT DEFAULT 0,
20   csapatokszama INT DEFAULT 0,
21   gyoztescsapat VARCHAR(255) NULL,
22   FOREIGN KEY (profilid) REFERENCES profil(id) ON DELETE CASCADE
23 );
24
25 CREATE TABLE csapat (
26   id INT AUTO_INCREMENT PRIMARY KEY,
27   tornaid INT NOT NULL,
28   profilid INT NOT NULL,
29   gyozelmek INT DEFAULT 0,
30   veresegek INT DEFAULT 0,
31   dontetlenek INT DEFAULT 0,
32   csapatneve VARCHAR(255),
33   FOREIGN KEY (profilid) REFERENCES profil(id) ON DELETE CASCADE,
34   FOREIGN KEY (tornaid) REFERENCES torna(id) ON DELETE CASCADE
35 );
36
37 CREATE TABLE jatekos (
38   id INT AUTO_INCREMENT PRIMARY KEY,
39   csapatid INT NOT NULL,
40   golokszama INT DEFAULT 0,
41   sangalapok INT DEFAULT 0,
42   piroslapok INT DEFAULT 0,
43   nev VARCHAR(255),
44   pozicio VARCHAR(255),
45   FOREIGN KEY (csapatid) REFERENCES csapat(id) ON DELETE CASCADE
46 );
47
48 CREATE TABLE meccs (
49   id INT AUTO_INCREMENT PRIMARY KEY,
50   tornaid INT NOT NULL,
51   meccstipusa VARCHAR(255),
52   csapat1 INT,
53   csapat2 INT,
54   cs1gol INT DEFAULT 0,
55   cs2gol INT DEFAULT 0,
56   datum VARCHAR(255) NOT NULL,
57   FOREIGN KEY (csapat1) REFERENCES csapat(id) ON DELETE CASCADE,
58   FOREIGN KEY (csapat2) REFERENCES csapat(id) ON DELETE CASCADE,
59   FOREIGN KEY (tornaid) REFERENCES torna(id) ON DELETE CASCADE
60 );
61
62 CREATE TABLE csoport (
63   id INT AUTO_INCREMENT PRIMARY KEY,
64   csoportid INT,
65   tornaid INT NOT NULL,
66   csapatid INT,
67   kapottgolok INT DEFAULT 0,
68   rugottgolok INT DEFAULT 0,
69   golkulonbseg INT DEFAULT 0,
70   pontok INT DEFAULT 0,
71   FOREIGN KEY (tornaid) REFERENCES torna(id) ON DELETE CASCADE,
72   FOREIGN KEY (csapatid) REFERENCES csapat(id) ON DELETE CASCADE
73 );

```

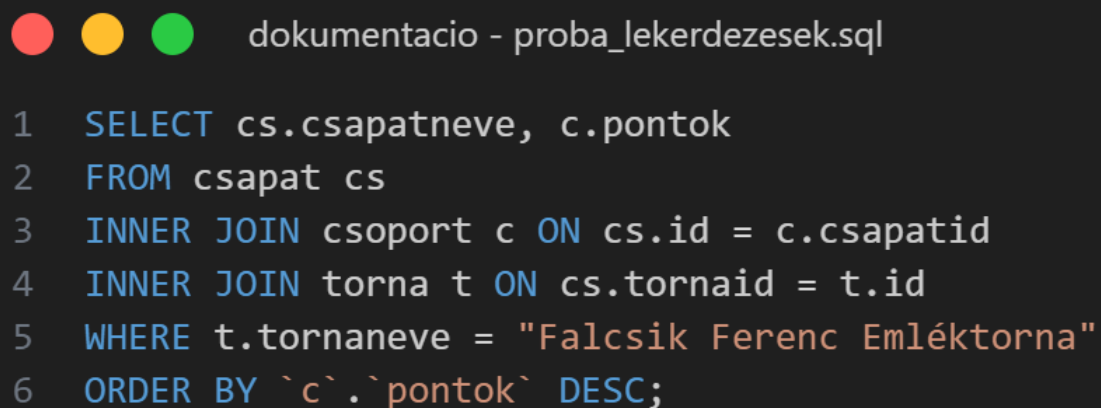
7. ábra

8. Lépés: Létrehozás és tesztelés



8. ábra

Adatbázisunk tesztelésére 10 próba lekérdezést készítettünk, hogy megbizonyosodjunk arról, a táblák közötti kapcsolat tökéletes.



```
1 SELECT cs.csapatneve, c.pontok
2 FROM csapat cs
3 INNER JOIN csoport c ON cs.id = c.csapatid
4 INNER JOIN torna t ON cs.tornaid = t.id
5 WHERE t.tornaneve = "Falcsik Ferenc Emléktorna"
6 ORDER BY `c`.`pontok` DESC;
```

9. ábra

Backend – Node.js

Backend oldali szerverünket a Node.js keretrendszerrel készítettük el, amely egy JavaScript alapú futtatókörnyezet.

A Node.js aszinkron műveleteket alkalmaz, amelyek mivel nem blokkolják a működést, így a szerver egyszerre több kérést is tud kezelni. Emellett a kód esemény vezérelt modellre épül, ami miatt nem csak reagálni tud a kérésekre, de callback-eket (visszahívásokat) tud használni

A keretrendszerhez tartozik az NPM (Node Package Manager), ami a legnagyobb nyíltforráskódú csomagkezelő. Ennek köszönhetően különböző könyvtárakat és modulokat értünk el, amelyek segítségünkre voltak a fejlesztés során. Ilyen volt például az Express, CORS, Body-parser és MySQL2.

- Express: webkeretrendszer Node.js számára, amely segít az API-k fejlesztésében. Megkönnyíti a REST API-k készítését és támogatja a middleware-eket (köztes szoftver – olyan szoftver, ami az operációs rendszer és a rajtafutó alkalmazás között helyezkedik el).
- CORS: Node.js middleware, ami lehetővé teszi a Cross-Origin Resource Sharing-t, így a weboldal hozzáférhet majd a szerver által lekért adatokhoz.
- Body-parser: olyan middleware, amely a HTTP-kérések body-ját képes JSON formátumban feldolgozni.
- MySQL2: MySQL kliensmodul Node.js számára.

```
Projekt - methodusok.js

1 // GET - minden adat lekérése
2 app.get('/:tabla', (req, res) => {
3     const { tabla } = req.params;
4     const lekerdezes = 'SELECT * FROM ?? ';
5     adatb.query(lekerdezes, [tabla], (err, results) => {
6         if (err) {
7             return res.json({error: 'Sikertelen lekérdezés: ', err});
8         }
9         res.json(results);
10    });
11 });
```

10. ábra

A szerver létrehozásakor a legfőbb célunk az volt, hogy egy gyors, megbízható, dinamikus és minden funkciót tökéletesen megvalósító backend oldali szervert hozzunk létre.

A backendünk egy olyan REST API-n keresztül éri el az adatbázist, amelyhez különböző HTTP metódusokat használtunk (GET – lekérés, POST – feltöltés, PUT – módosítás, DELETE – törlés). Az API végpontokat dinamikusan hoztuk létre, hogy egy végpont minden táblát le tudjon kezelni, így a tiszta kód elvének is megfelel a kódunk. Ezt úgy valósítottuk meg, hogy a request (kérés) url-jében egy router parameter-ként (útválasztó paraméterként) kérjük be a tábla nevét, amelyet majd a kérés indításakor a frontend fog meghatározni. A megírt szerver tökéletes volt a mobil applikációnak számára is, így mind a kettő ugyan azt a kódot használja.

Az API végpontok működésének tesztelésére először a ThunderClient bővítményt használtuk. A teszteket kollekcióként mentettük .json kiterjesztésben.

A JavaScript egy objektumorientált szkriptnyelv, amelyet Brendan Eich fejlesztette ki 1996-ban, Mocha néven, majd LiveScript lett. A JavaScript nevet később kapta a Java programozási nyelvről, amely akkoriban nagyon népszerű volt.

A szkriptnyelvek olyan nyelvek, amelyeket nem kell elfordítani, mert a program azonnal tudja értelmezni a feladatot. Így általában automatizálásra vagy weboldalak működtetésére használják.

Frontend – Angular

A projektünk frontendje Angular használatával készült, amely egy TypeScript alapú frontend keretrendszer. 2012-ben adta ki a Google, akkor még JavaScript alapon, viszont 2016-ban TypeScript-re váltottak.

Az Angular modulok köré épül, amik logikailag csoportosítják a különböző komponenseket, szolgáltatásokat és egyéb funkciókat. Minden alkalmazás legalább egy modult, a gyökérmodult tartalmazza. Weboldalunk elkészítéséhez több különböző modult és használtunk: HttpClientModule, CommonModule, FormsModule, ngxChartsModule és BrowserAnimationsModule. Ezeket, hogy megfelelően tudjuk alkalmazni őket majd a későbbiekben, az app.module.ts-ben kellett importálnunk.

A HttpClientModule az Angular egyik alap modulja, amelyet nem kell külön telepíteni Node Package Manager-el. A modul a HTTP-kérések kezelésére alkalmas, a szerverrel való kommunikációt biztosítja.

A CommonModule beépített modul, amely a direktívákat és funkciókat tartalmazza (*ngIf, *ngFor). A lekért adatokat dictionary-kben (szótárakban) tároltuk. Ahhoz, hogy ezekből megtudjuk jeleníteni az adatokat a képernyőn szükség volt egy ciklusra. Erre pedig tökéletes volt számunkra az *ngFor.

A FormsModule a Template-Driven Forms (sablonvezérelt űrlapok) modulja az Angularban. Lehetővé teszi a kétirányú adatbindingot (vagyis a .html és a .ts fájlok között az adatok automatikusan szinkronizálódnak). Ez szükséges volt a módosítás és az új adatok feltöltéséhez, hiszen másképp nem tudtuk volna az információt bekérni a felhasználótól.

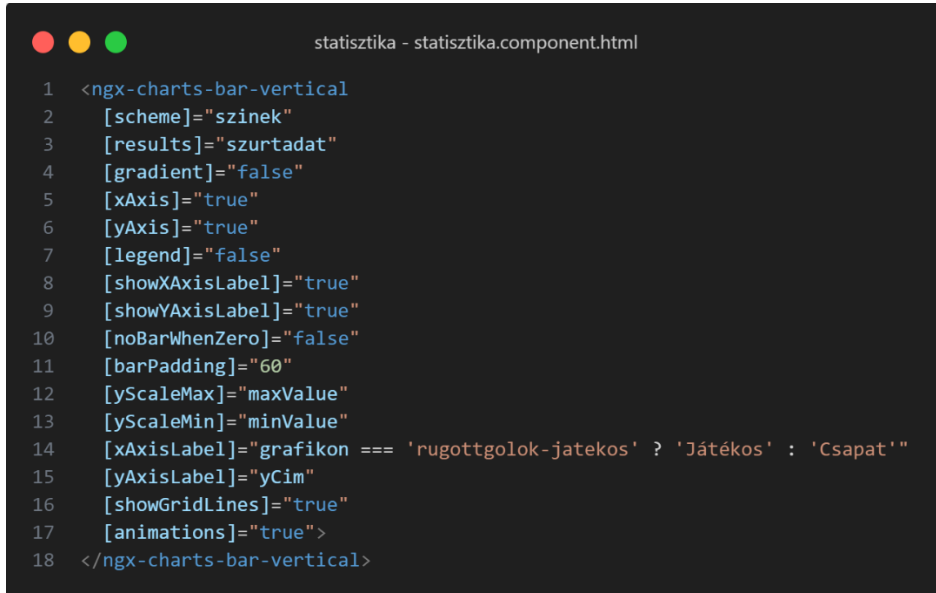
Az Ngx-ChartsModule egy diagramkönyvtár az Angular számára, amely testesztelhető és interaktív diagramtípusok széles választékát kínálja. Az ngx-diagramok támogatott diagramtípusai közé tartozik a vonal, terület, oszlop, vízszintes sáv, kör, fánk, mérőeszköz, hő térkép, kényszerirányított grafikon, buborékdiagram és egyéb változatok.

Jellemzői:

- D3.js alapú – Nagy választék a grafikonok között
- Angular-hoz optimalizált – Angular-kompatibilis, nincs szükséges külső JavaScriptre
- Reszponzív és interaktív – Automatikusan igazodik a képernyőmérethez és interaktív elemeket biztosít (pl. hover effektek, kattintható elemek).
- Támogatja a komplex diagramokat – Hő térképek, fa diagramok

Hátrányok:

- Nehezebb az elsajátítása
- Nem lehet teljesen testre szabni



```
statisztika - statisztika.component.html
1 <ngx-charts-bar-vertical
2   [scheme]="szinek"
3   [results]="szurtadat"
4   [gradient]="false"
5   [xAxis]="true"
6   [yAxis]="true"
7   [legend]="false"
8   [showXAxisLabel]="true"
9   [showYAxisLabel]="true"
10  [noBarWhenZero]="false"
11  [barPadding]="60"
12  [yScaleMax]="maxValue"
13  [yScaleMin]="minValue"
14  [xAxisLabel]="grafikon === 'rugottgoloj-jatekos' ? 'Játékos' : 'Csapat'"
15  [yAxisLabel]="yCim"
16  [showGridLines]="true"
17  [animations]="true">
18 </ngx-charts-bar-vertical>
```

11. ábra

A BrowserAnimationsModule az Angular alapvető modulja, amely nélkül nem lehet a programban animációkat alkalmazni. Weboldalunk és a grafikonok design-ja miatt a használata számunkra elkerülhetetlen volt.

Az Angular komponensekre épül, amik független, újra használható kódmodulok. Különbözően tartalmaznak HTML-t, CSS-t és TypeScriptet is. A HTML a weboldal vázát, a CSS a megjelenését, a TypeScript pedig a viselkedést tartalmazza. Minden oldalunkat külön komponensként hoztuk létre egy fő gyökérkönyvtárban.

A service-k a weboldalunk logikai részének kezelésére szolgálnak. A feladatok, amelyeket elvégeznek egy központi helyen vannak (például az adatlekérés), így ez minden komponens számára egyaránt elérhető. A frontend a service-ken keresztül vannak kapcsolatban a backenddel. Itt kapja meg az adatokat és innen indítja a különböző műveleteket (módosítás, törlés, hozzáadás) a felhasználó. Két különböző service-t készítettünk, az egyik a fiók műveleteket kezeli, a másik pedig a tornák adatait.

```

Projekt - adatok.service.ts

1 import { HttpClient } from '@angular/common/http';
2 import { Injectable } from '@angular/core';
3 import { Observable, forkJoin } from 'rxjs';
4
5 @Injectable({
6   providedIn: 'root'
7 })
8 export class AdatokService {
9   // url és egy tömb, ami a táblák neveit tárolja
10  private apiUrl = 'http://localhost:3000';
11  private tablak = ['profil', 'csapat', 'torna', 'jatekos', 'meccs', 'csoport'];
12
13  constructor(private http: HttpClient) { }
14
15  // a függvény végig megy a táblák tömbön, aztán elindítja a végpont kéréseket egyszerre
16  GETmindentabla(): Observable<any> {
17    let requests = this.tablak.map(tabla =>
18      this.http.get<any[]>(`${this.apiUrl}/${tabla}`)
19    );
20    return forkJoin(requests); // Egyszerre indítja az összes kérést
21  }
22
23  // ez arra van ha esetleg egy konkrét tábla adatát akarjuk lekérni
24  GETegytabla(tabla: string): Observable<any[]> {
25    return this.http.get<any[]>(`${this.apiUrl}/${tabla}`);
26  }
27
28  // hozzáadás
29  add(tabla: string, adatok: any): Observable<any> {
30    return this.http.post(`${this.apiUrl}/${tabla}`, adatok);
31  }
32
33  // törlés
34  delete(tabla: string, id: number): Observable<any> {
35    return this.http.delete(`${this.apiUrl}/${tabla}/${id}`);
36  }
37
38  // módosítás
39  update(tabla: string, id: number, adatok: any): Observable<any> {
40    return this.http.put(`${this.apiUrl}/${tabla}/${id}`, adatok);
41  }
42 }

```

12. ábra

A routing-nak köszönhetően SPA-kat (Single-Page Application – Egy oldalas alkalmazás) hozhatunk létre. A fájlban meg kell adnunk a URL-t, amelyen a felhasználó eléri a weblapot, így váltáskor a weboldal viszont nem vált az oldalak között, csupán a megadott útvonal alapján frissíti a felhasználói felület adatait.

```
Projekt - app-routing.module.ts

1 import { NgModule } from '@angular/core';
2 import { RouterModule, Routes } from '@angular/router';
3
4 import { FooldalComponent } from './components/fooldal/fooldal.component';
5 import { KeresesComponent } from './components/kereses/kereses.component';
6 import { AdatkezelesComponent } from './components/adatkezeles/adatkezeles.component';
7 import { StatisztikaComponent } from './components/statisztika/statisztika.component';
8
9 // útvonalak
10 const routes: Routes = [
11
12   {path: "", component: FooldalComponent},
13   {path: "kereses", component: KeresesComponent},
14   {path: "adatkezeles", component: AdatkezelesComponent},
15   {path: "statisztika", component: StatisztikaComponent}
16
17 ];
18
19 @NgModule({
20   imports: [RouterModule.forRoot(routes)],
21   exports: [RouterModule]
22 })
23 export class AppRoutingModule { }
```

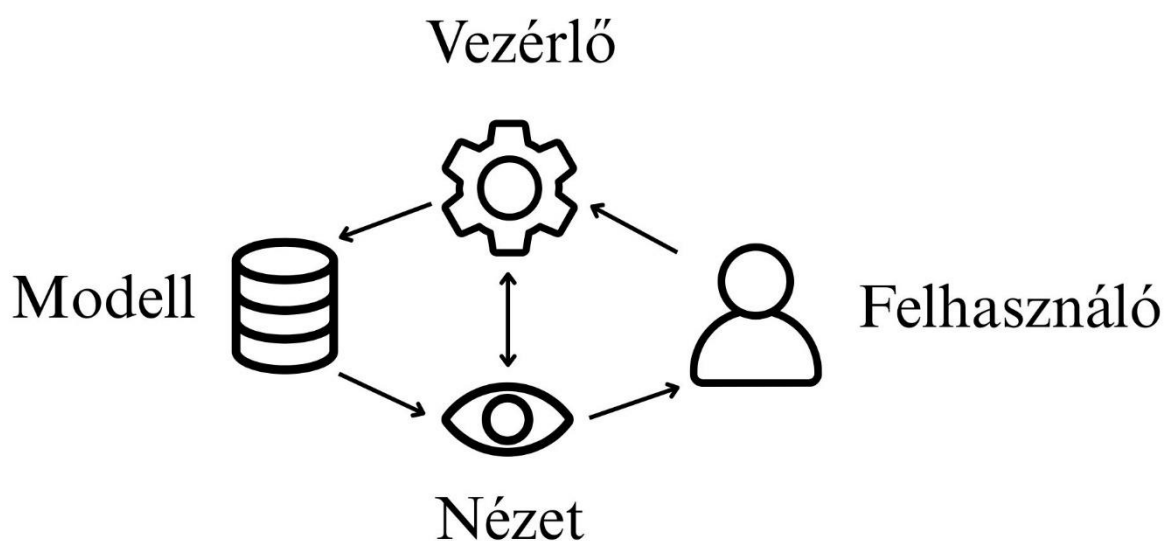
13. ábra

A TypeScriptet a Microsoft fejlesztette ki 2010-ben. Gyakran „JavaScript kiterjesztés” vagy „JavaScript típusrendszerrel” néven emlegetik. A TypeScript egyszerre programozási nyelv, típusellenőrző és fordító.

- Programozási nyelv: a saját nyelvi szintaxisán kívül a JavaScriptet is tartalmazza.
- Típusellenőrzés: képes .js és .ts fájlokat értelmezni és jelez, ha a felépítésük hibás.
- Fordító: lefuttatja a típusellenőrzést és elkészíti a .js formáját a kódnak

Programtervezési minta – MVC

MVC (Model-View-Controller – Modell-Nézet-Vezérlő) programtervezési mintára épül a weboldalunk. Elkülönítjük a modellt (adatok) a nézettől (felhasználói felület), így a nézet változásai nem befolyásolják a modellt és fordítva. Ez annak köszönhető, hogy bevezeti a vezérlőt (backendoldali szerver). Ez feldolgozza felhasználói műveleteket, válaszol rájuk, és megváltoztatja a modellt.



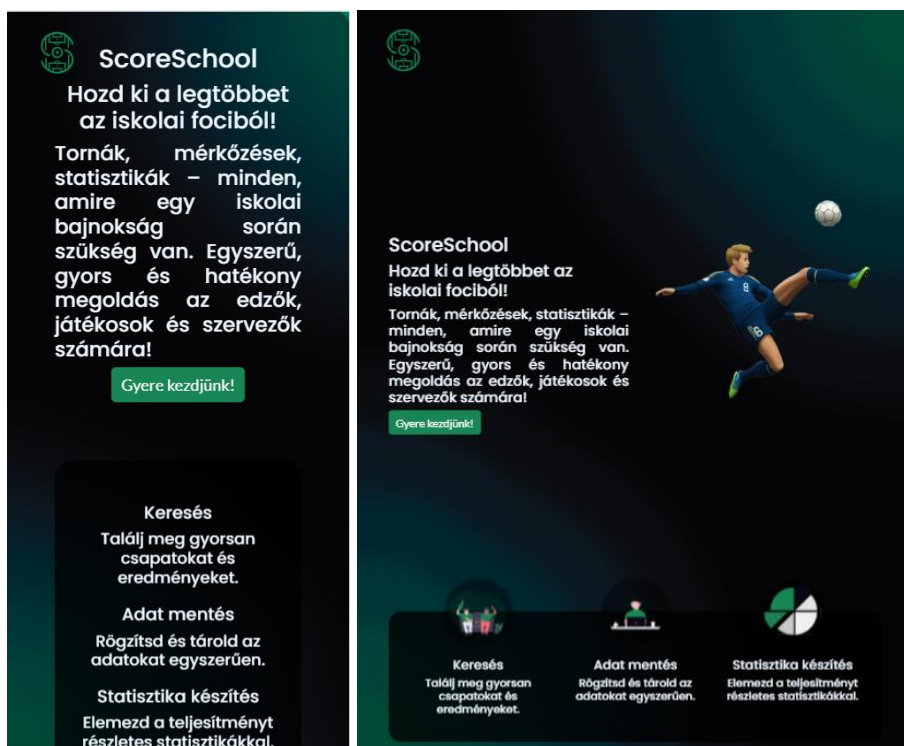
14. ábra

Bootstrap

Az oldal reszponzivitását Bootstrap-el oldottuk meg, ami egy nyílt forráskódú frontend keretrendszer tervezéshez és fejlesztéshez. A rács szerkezete kiváló bármilyen projekt számára, hiszen a szükséges CSS osztályok előre definiálva vannak 6 különböző méretben (xs, sm, md, lg, xl, xxl). A weboldalunk monitorra, tabletre és telefonra lett optimalizálva.



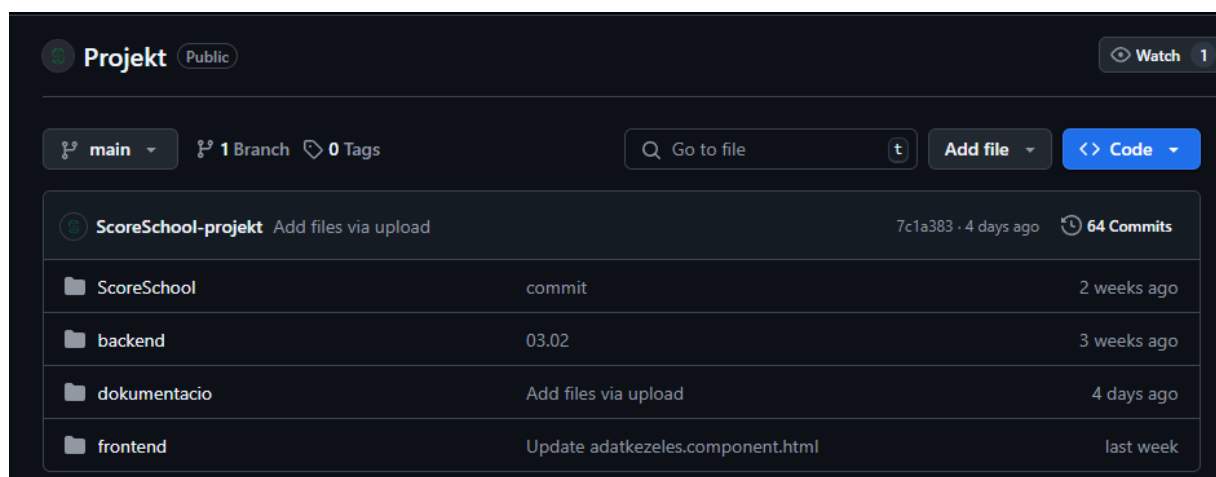
15. ábra



16. ábra

Verziókezelés

Verziókezelésre a GitHubot használtuk. Itt lehetőségünk volt követni a változásokat és egymás munkáját. Projektünket egy közös repository-ban tároltuk, így bármikor hozzáférhettünk és egyszerre dolgozhattunk rajta. A repository tartalmazza a backendet, a frontendet, a mobil applikációt és a dokumentációt is.



Applikáció

Az alkalmazást Expo segítségével készítettük, amely egy nyíltforráskódú platform. JavaScript és React alapú natív mobilalkalmazásokat készíthetünk vele. A cross-platform-nak köszönhetően nem csak Androidon, de iOS-en is tökéletesen fut az appunk.

Az appunk adatbázisa és backendoldali szervere megegyezik a weboldaléval. Ennek a legfőbb oka az, hogy a projekt során ne kelljen két különböző adatbázist kezelni és hogy az adatok valós időbe frissüljenek bármelyik platformon is változtatunk rajta.

Az alkalmazásunk frontendje React Native és az Expo keretrendszer segítségével készült és főleg informatív célt szolgál:

- A főoldal ismerteti a weboldal fő funkcióit.
- Bejelentkezhethetünk meglévő fiókunkba vagy újjal regisztrálhatunk.
- A keresőoldalon a legördülő listából kiválasztva a kategóriát szűrhetjük az adatokat, vagy a kereső mezővel pontosan kereshetünk bárkire vagy bármire.

Tesztelés

A projektünk White box teszteléséhez a Jest-et alkalmaztunk. A Jest egy univerzális tesztkeretrendszer, ami képes bármilyen JavaScript könyvtárhoz vagy keretrendszerhez igazodni. Kiváló backend és frontend tesztelésre is, hiszen képes a weboldal főbb funkcióin kívül az API végpontokat is egyszerűen tudtuk tesztelni a használatával. Támogatja az egység és az aszinkron teszteket is.

Az összes API végpontra külön tesztet készítettünk. A tesztek mindig egy adott tábla adataival futnak le és a visszakapott státuszkódot, válasz üzenetet ellenőrzik. Adat lekérés esetén pedig, hogy nem üres listával tér-e vissza.



```

1  const request = require('supertest');
2  const app = require('../methodusok');
3
4  afterAll(async () => {
5      await app.lezaras();
6  });
7
8  describe('GET - minden adat teszt', () => {
9      it('GET /:tabla - lekérdezés', async () => {
10         const res = await request(app).get('/jatekos');
11         expect(res.status).toBe(200);
12         expect(Array.isArray(res.body)).toBe(true);
13     });
14 });

```

18. ábra

A frontend tesztelésekor leginkább az egyes funkciók tényleges működését akartuk ellenőrizni. A projekt létrehozásakor az automatikus generált az egyes komponensekhez és service-khez külön .spec.ts fájlt, ezeket használtuk fel a tesztek megírásakor. A service-k esetén minden eljárást külön teszteltünk le, hogy megbizonyosodjunk róla, megkapja az adatokat a backend szervertől. A komponensek esetén pedig a célunk az volt, hogy megvizsgáljuk, minden funkció sikeresen végrehajtódik-e, ha az elvárt adatokat kapja és hiányos vagy rossz adat esetén, ne hajtódjon végre.

Tapasztalatok és csapatmunka

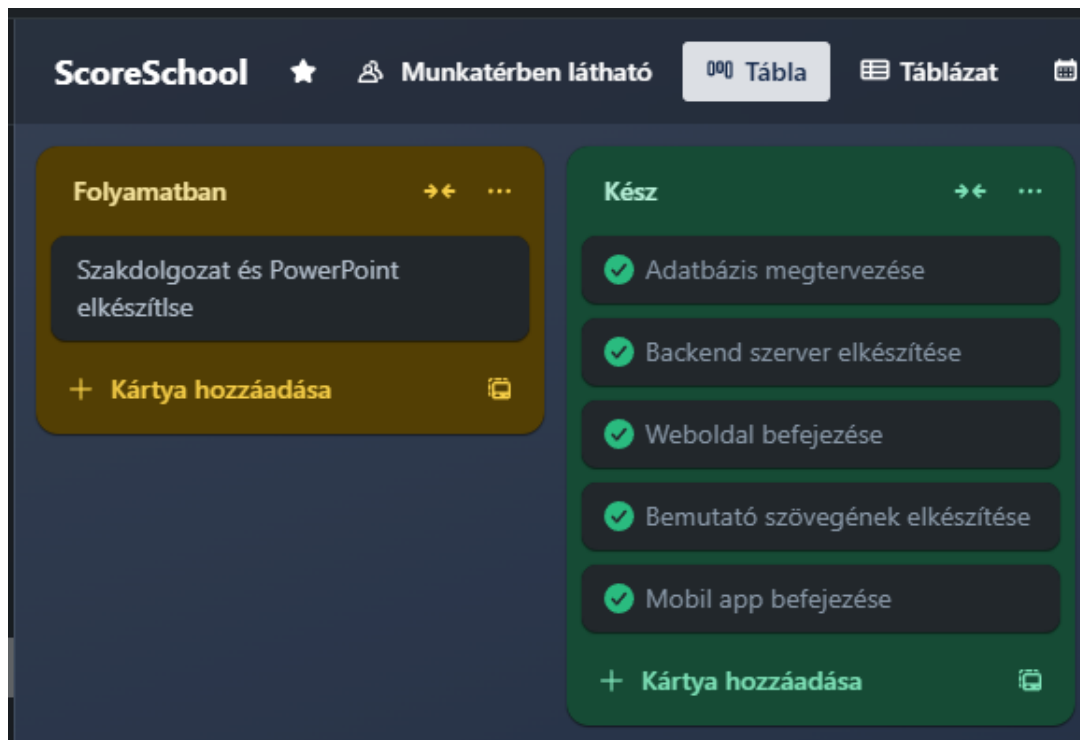
A ScoreSchool fejlesztése során az egyik legnagyobb kihívás számunkra az volt, hogy hogyan hangoljuk össze a munkát hármunk között. A munkamegosztás alapja a képességeink és érdeklődési köreink voltak. A csoportunk tagjai mind másban jók és más részeket tartanak számukra közelebb így nem volt kérdéses ki milyen feladatot vállal. Dominika feladata volt a backend létrehozása, Szabolcs a frontendet készítette el, míg Balázs a mobil applikációt valósította meg. A munka során voltak olyan részek is, amelyeket közösen dolgoztunk ki, ilyen volt például az adatbázis. Mindhárman külön tervet készítettünk a szerintünk legoptimálisabb adatbázisról, majd ezeket egy Daily Scrum során átnéztük és megvitattuk melyik és miért lenne a legjobb a projektünk számára.

A tanórák keretein belül a tanáraink lehetőséget adtak 15 perces Daily Scrum meetingekre. Ilyenkor megbeszélhettük ki hol tart, milyen problémába ütközött vagy éppen milyen új ötlete támadt az előző nap. Ezek a megbeszélések elengedhetetlenek voltak a fejlesztés során, hiszen ilyenkor a tanáraink is ráláttak a projektünk és javaslatokat tettek.

Daily Scrum	
Dátum: 2025.02.19	Megbeszélés témái:
Scrum Master: Lakatos Sándor Jelenlévők: 1. Ármós Szabolcs 2. Gáspár Marianna Dominika 3. Sankó Balázs	Megbeszélés kérdései: 1. Mit csináltál a tegnapi nap folyamán? 2. Voltak nehézségek a feladat során? 3. Mit fogsz ma csinálni?
Csapat tagjai: 1. Ármós Szabolcs 2. Gáspár Marianna Dominika 3. Sankó Balázs	Válaszok: 1. Ármós Szabolcs: a. A tegnap délutánomat weblap designok és templatek keresésével töltöttem, hogy a projektünk frontend-jének fejlesztéséhez legyen valamilyen ihletünk. b. Nem ütköztem nehézségekbe, mert az interneten több weboldal design és template is elérhető. A talált sablonok Pinterestről vannak, főleg a weboldal statisztika részéhez, hogy hogyan nézne ki a legjobban és milyen elrendezések kellenének. c. A mai nap a feladatom az, hogy dokumentációt készítsék arról, hogy mi a különbség az NgxChart és a Primeng-n keresztül elérhető Chart.js között. Ez alapján könnyebben tudunk majd dönteni arról, hogy melyiket lenne érdemesebb használnunk a

19. ábra

A feladatok beadási dátumát és sorrendjét a Trello weboldalán tudtuk nyomon követni. Így mindenki láthatta, hogy haladunk és mit kell még megcsinálni. Ha valamelyikünk elakadt itt jelezni tudtuk és a feladatokat újra oszthattuk, vagy segítettünk egymásnak, hogy mindig befejezzük a határidők előtt. Annak ellenére, hogy a feladatokat felosztottuk, mindenki rálátott a másik haladására és mindig megvitattuk, ha új ötletünk volt. A különböző részeket mindig átbeszéltük, hogy ne csak a saját részünket, de a többiekét is átlássuk.



20. ábra

Jövőkép

Projektünk elsődleges célja, hogy minél több iskolában elterjedjen használata, viszont több ötletünk is van a fejlesztés kapcsán:

- „Kivetítő funkció”: élő eredmény, amelyet a meccs során folyamatosan lehet módosítani és akár kivetítőn vagy egy TV-n ki lehetne vetíteni a nézők számára, hogy követhessék a gólokat.
- Élő közvetítés a meccsekről.
- Lehetőség videók társítására a különböző meccsekhez, mint például a legszebb gól, vagy egy válogatás a legjobb pillanatokról.
- Bejelentkezési lehetőség Gmail vagy Facebook segítségével.
- „Fogadási lehetőség”: a felhasználók az élő meccsek közben megtippelhetik a meccs eredményét, bármilyen tét nélkül. Minden héten a lejátszott meccsek fogadásai alapján lenne egy rangsor a legpontosabb felhasználókból. A játék során „kitűzőket” lehetne nyerni, amelyek a felhasználó neve mellett jelennének meg.

- Lehetőség saját diagram készítésére. A felhasználó választana ki, minden elemét saját tetszése alapján, legyen szó akár a megjeleníteni akart adatról, színekről vagy a diagram típusáról.

Természetesen projektünk megosztása után kíváncsian várjuk majd a felhasználók javaslatait és véleményeit arról, hogy hogyan fejleszthetnénk weboldalunkat.

Ábra jegyzék

1. ábra: Főoldal
2. ábra: Keresés
3. ábra: Adatkezelés
4. ábra: Statisztika
5. ábra: Logónk a Photoshop alkalmazásba
6. ábra: Az adatbázis ER diagramja
7. ábra: Az adatbázis sémája
8. ábra: Az adatbázis UML diagramja
9. ábra: Teszt lekérdezés
10. ábra: Backend oldali szerver GET Api végpontja
11. ábra: ngx-Chart alkalmazása
12. ábra: adatok.service.ts fájl
13. ábra: app-routing.module.ts fájl
14. ábra: Model-View-Controller model
15. ábra: weblap megjelenése LG (monitor) méretben
16. ábra: weblap megjelenése MD és SM (tablet és telefon) méretben
17. ábra: a projekt github repository-jának felépítése
18. ábra: get.test.js fájl
19. ábra: Daily Scrum dokumentációja (2025.02.19)
20. ábra: a projekt feladatainak követése a Trello weboldalán

Felhasznált irodalom

1. GOOGLE, LLC. *Express dokumentáció* Elérhetőség: <https://expressjs.com/>
2. GOOGLE, LLC. *CORS dokumentáció* Elérhetőség: <https://www.npmjs.com/package/cors>
3. GOOGLE, LLC. *Body-parser dokumentáció* Elérhetőség: <https://www.npmjs.com/package/body-parser>
4. GOOGLE, LLC. *MySQL2 dokumentáció* Elérhetőség: <https://www.npmjs.com/package/mysql2>
5. GOOGLE, LLC. *React Native* Elérhetőség: <https://reactnative.dev/>
6. GOOGLE, LLC. *Expo dokumentáció* Elérhetőség: <https://docs.expo.dev/>
7. Robin Nixon *Learning PHP, MySQL & JavaScript, 7th Edition* 2025. Elérhetőség: <https://learning.oreilly.com/library/view/learning-php-mysql/9781098152345/>
8. David Herron *Node.js Web Development* 2020. Elérhetőség: <https://learning.oreilly.com/library/view/node-js-web-development/9781838987572/>
9. Aristeidis Bampakos *Learning Angular – Fifth Edition* 2025. Elérhetőség: <https://learning.oreilly.com/library/view/learning-angular/9781835087480/>
10. Brett Jephson, Lewis Coulson, Ana Carolina Silveira *Practical HTML and CSS – Second Edition* 2024. Elérhetőség: <https://learning.oreilly.com/library/view/practical-html-and/9781835080917/>
11. Anna Skoulikari *Learning Git* 2023. Elérhetőség: <https://learning.oreilly.com/library/view/learning-git/9781098133900/>
12. Josh Goldberg *Learning TypeScript* 2022. Elérhetőség: <https://learning.oreilly.com/library/view/learning-typescript/9781098110321/>