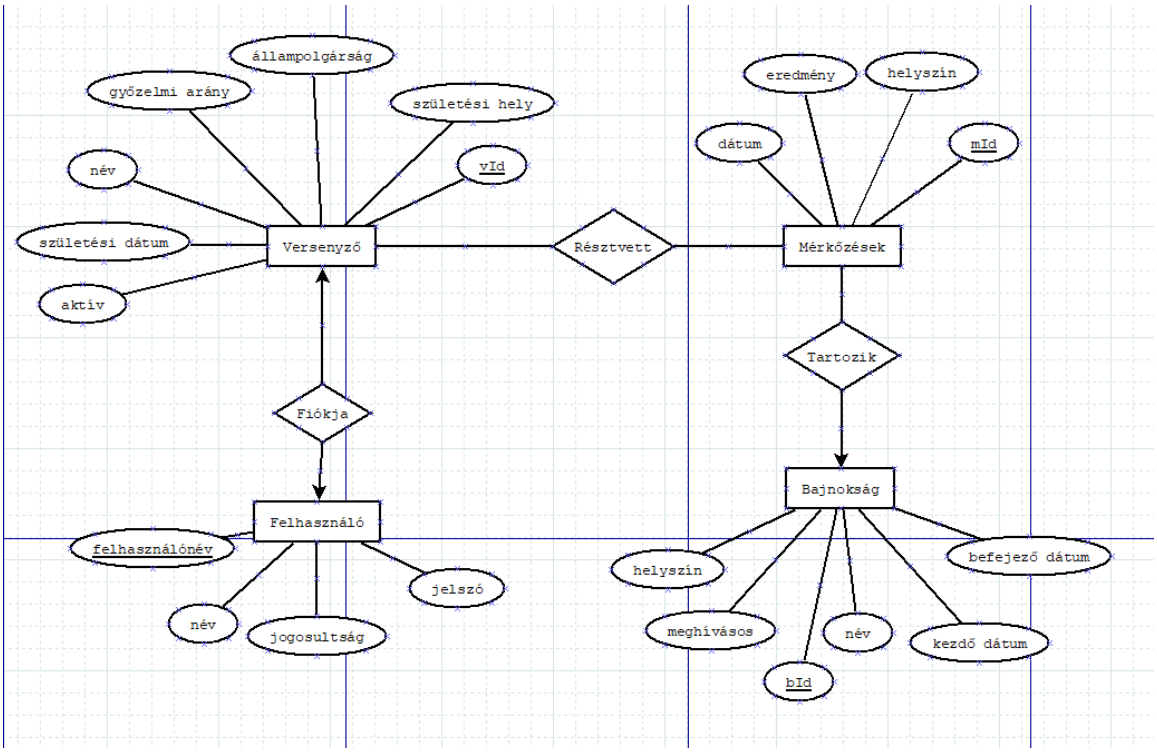
Dokumentáció

E-K diagram:



(Ha a kép nem látható mellékeltem egy png filet.)

Egy felhasználó lehet versenyző is, de nem kötelező. Fordítva is igaz, hogy nem minden versenyző rendelkezik felhasználói. Egy mérkőzésen több versenyző is részt vehet és egy versenyző több mérkőzésen vehet részt. Egy bajnokság több mérkőzésből áll, de egy mérkőzés (legfeljebb) egy bajnoksághoz tartozik.

Relációs sémákká képezés:

Felhasználó(<u>felhasználónév</u>, jelszó, név, jogosultság)

Versenyző(vId, név, születési dátum, születési hely, állampolgárság, győzelmi arány, aktív)

Bajnokság(bId, név, kezdő dátum, befejező dátum, helyszín, meghívásos)

Mérkőzések(mId, eredmény, dátum, helyszín)

Résztvétel(*vId*, *mId*)

Fiókja(*vId*, *felhasználónév*)

Tartozik(*mId*, *bId*)

(Megjegyzés: aláhúzva a kulcsokat, dőlt betűvel a külső kulcsokat jelöltem)

Normalizálás:

Minden sémát úgy alkottam meg, hogy az már 3NF-ben legyen, magyarázat:

1NF: mert minden adat atomi, azaz nincs összetett vagy többértékű attribútum

2NF: az elsődleges attribútumoktól mindenhol egyértelműen függenek a másodlagosak, mivel ahol csak egyelemű a kulcs, ott ez adott, ahol pedig többelemű a kulcs ott nincs másodlagos attribútum

3NF: nincs tranzitív függősék, mivel mindenhol, ahol vannak másodlagos attribútumok ott a kulcstól függenek

Táblatervek:

Mező - Típus - Megjegyzés

Felhasználó tábla

felhasznalonev - VARCHAR(50) - Felhasználónév, egyedi kulcs

jelszo - VARCHAR(100) - Jelszó

nev - VARCHAR(100) - Felhasználó neve

jogosultsag - VARCHAR(50) - Felhasználói jogosultság

Versenyző tábla

vId - INT - Versenyző ID, egyedi kulcs (auto increment)

nev - VARCHAR(100) - Versenyző neve

szuletesi_datum - DATE - Születési dátum

szuletesi_hely - VARCHAR(100) - Születési hely

allampolgarsag - VARCHAR(50) - Állampolgárság

gyozelmi_arany - DECIMAL(5,2) - Győzelmi arány (pl. 0.75)

aktiv - BOOLEAN - Aktív státusz (igaz/hamis)

Bajnokság tábla

bId - INT - Bajnokság ID, egyedi kulcs (auto increment)

nev - VARCHAR(100) - Bajnokság neve

kezdo_datum - DATE - Kezdő dátum

befejezo_datum - DATE - Befejező dátum

helyszin - VARCHAR(100) - Helyszín

meghivasos - BOOLEAN - Meghívásos jelleg (igaz/hamis)

Mérkőzések tábla

mId - INT - Mérkőzés ID, egyedi kulcs (auto increment)

eredmeny - VARCHAR(100) - Eredmény (Itt a győztes versenyzőt mentem)

datum - DATE - Mérkőzés dátuma

helyszin - VARCHAR(100) - Mérkőzés helyszíne

Résztvétel tábla

vId - INT - Versenyző ID (hivatkozás a Versenyző táblára)

mId - INT - Mérkőzés ID (hivatkozás a Mérkőzések táblára)

- - Kombinált elsődleges kulcs (vId, mId)

Fiókja tábla

vId - INT - Versenyző ID (hivatkozás a Versenyző táblára)

felhasznalonev - VARCHAR(50) - Felhasználónév (hivatkozás a Felhasználó táblára)

- - Kombinált elsődleges kulcs (vId, felhasznalonev)

Tartozik tábla

mId - INT - Mérkőzés ID (hivatkozás a Mérkőzések táblára)

bId - INT - Bajnokság ID (hivatkozás a Bajnokság táblára)

- - Kombinált elsődleges kulcs (mId, bId)

SQL KÓDBAN:

```
-- Felhasználó tábla
CREATE TABLE Felhasznalo (
   felhasznalonev VARCHAR(50) PRIMARY KEY,    -- Felhasználónév, egyedi kulcs
   jelszo VARCHAR(100) NOT NULL,              -- Jelszó, nem lehet NULL
   nev VARCHAR(100) NOT NULL,                 -- Név, nem lehet NULL
   jogosultsag VARCHAR(50) NOT NULL           -- Jogosultság, nem lehet NULL
);


-- Versenyző tábla
```

```
CREATE TABLE Versenyzo (
   vId INT PRIMARY KEY AUTO_INCREMENT,       -- Versenyző ID, auto increment
   nev VARCHAR(100) NOT NULL,                -- Név, nem lehet NULL
   szuletesi_datum DATE NOT NULL,  -- születési dátum
   szuletesi_hely VARCHAR(100) NOT NULL,     -- Születési hely, nem lehet NULL
   allampolgarsag VARCHAR(50) NOT NULL,      -- Állampolgárság, nem lehet NULL
   gyozelmi_arany DECIMAL(5,2) CHECK (gyozelmi_arany >= 0 AND gyozelmi_arany <= 1),
 -- Győzelmi arány 0 és 1 között
   aktiv BOOLEAN NOT NULL DEFAULT TRUE       -- Aktív státusz, alapértelmezett érték: TRUE
);

-- Bajnokság tábla
CREATE TABLE Bajnoksag (
   bId INT PRIMARY KEY AUTO_INCREMENT,       -- Bajnokság ID, auto increment
   nev VARCHAR(100) NOT NULL,                -- Bajnokság neve, nem lehet NULL
   kezdo_datum DATE NOT NULL,                -- Kezdő dátum, nem lehet NULL
   befejezo_datum DATE NOT NULL CHECK (befejezo_datum >= kezdo_datum),
-- Befejező dátum nem lehet korábbi, mint a kezdő dátum
   helyszin VARCHAR(100) NOT NULL,           -- Helyszín, nem lehet NULL
   meghivasos BOOLEAN NOT NULL               -- Meghívásos jelleg, nem lehet NULL
);

-- Mérkőzések tábla
CREATE TABLE Merkozesek (
   mId INT PRIMARY KEY AUTO_INCREMENT,       -- Mérkőzés ID, auto increment
```

```
    eredmeny VARCHAR(100) CHECK (eredmeny != ''), -- Eredmény nem lehet üres
    datum DATE NOT NULL,                    -- Dátum, nem lehet NULL
    helyszin VARCHAR(100) NOT NULL          -- Helyszín, nem lehet NULL
);

-- Résztvétel tábla (Kapcsoló tábla a versenyzők és mérkőzések között)
CREATE TABLE Resztvetel (
    vId INT,                            -- Versenyző ID
    mId INT,                             -- Mérkőzés ID
    PRIMARY KEY (vId, mId),                 -- Kombinált elsődleges kulcs
    FOREIGN KEY (vId) REFERENCES Versenyzo(vId) ON DELETE CASCADE,
-- Hivatkozás a Versenyző táblára, törléskor cascade
    FOREIGN KEY (mId) REFERENCES Merkozesek(mId) ON DELETE CASCADE
-- Hivatkozás a Mérkőzések táblára, törléskor cascade
);

-- Fiókja tábla (Kapcsoló tábla a felhasználók és versenyzők között)
CREATE TABLE Fiokja (
    vId INT,                            -- Versenyző ID
    felhasznalonev VARCHAR(50),             -- Felhasználónév
    PRIMARY KEY (vId, felhasznalonev),        -- Kombinált elsődleges kulcs (vId, felhasznalonev)
    FOREIGN KEY (vId) REFERENCES Versenyzo(vId) ON DELETE CASCADE,
 -- Hivatkozás a Versenyző táblára, törléskor cascade
    FOREIGN KEY (felhasznalonev) REFERENCES Felhasznalo(felhasznalonev) ON DELETE CASCADE  --
Hivatkozás a Felhasználó táblára, törléskor cascade
);
```

```
-- Tartozik tábla (Kapcsoló tábla a mérkőzések és bajnokságok között)
CREATE TABLE Tartozik (
   mId INT,                          -- Mérkőzés ID
   bId INT,                          -- Bajnokság ID
   PRIMARY KEY (mId, bId),           -- Kombinált elsődleges kulcs
   FOREIGN KEY (mId) REFERENCES Merkozesek(mId) ON DELETE CASCADE,
-- Hivatkozás a Mérkőzések táblára, törléskor cascade
   FOREIGN KEY (bId) REFERENCES Bajnoksag(bId) ON DELETE CASCADE
 -- Hivatkozás a Bajnokság táblára, törléskor cascade
);
```

Programfunkciók:

1. Program Funkciók

Az alkalmazás alapvetően az alábbi funkciókkal rendelkezik:

1.1. Regisztráció és Bejelentkezés

Felhasználó regisztrációja: Az új felhasználók regisztrálhatnak a rendszerbe egy felhasználónév és jelszó megadásával. A regisztráció során meg kell adni a felhasználó nevét jogosultsága alapból néző.

Felhasználó bejelentkezése: A regisztrált felhasználók bejelentkezhetnek a rendszerbe a felhasználónevük és jelszavuk megadásával. A sikeres bejelentkezés után a felhasználó hozzáférhet az alkalmazás egyes funkcióihoz.

1.2. Versenyzők Kezelése

Nézőknek elérhető funkciók:

Versenyzők listázása: Az alkalmazás képes megjeleníteni az összes versenyzőt, és keresni közöttük például név alapján.

Adminnak elérhető funkciók:

Új versenyző felvétele: Az alkalmazás lehetőséget biztosít új versenyzők felvételére, akik tartalmazhatják a versenyző nevét, születési dátumát, születési helyét, állampolgárságát, győzelmi arányát és aktív státuszát.

Versenyzők listázása: Az alkalmazás képes megjeleníteni az összes versenyzőt, és keresni közöttük például név alapján.

Versenyző módosítása: Lehetőség van meglévő versenyzők adatainak módosítására (pl. állampolgárság, győzelmi arány).

Versenyző törlése: Az alkalmazás lehetőséget biztosít egy versenyző törlésére a rendszerből.

1.3. Bajnokságok Kezelése

Nézőknek elérhető funkciók:

Bajnokságok listázása: A bajnokságok listája megjeleníthető, és a felhasználók kereshetnek közöttük különböző szempontok szerint.

Adminfunkciók:

Új bajnokság felvétele: Az adminisztrátor új bajnokságokat vehet fel, amelyek tartalmazhatják a bajnokság nevét, kezdő és befejező dátumát, helyszínét, valamint hogy meghívásos-e.

Bajnokságok listázása: A bajnokságok listája megjeleníthető, és a felhasználók kereshetnek közöttük különböző szempontok szerint.

Bajnokság módosítása: Lehetőség van a bajnokságok adatainak módosítására (pl. helyszín, dátumok).

Bajnokság törlése: A rendszer lehetőséget biztosít a bajnokságok törlésére.

1.4. Mérkőzések Kezelése

Mérkőzések listázása: A felhasználók megtekinthetik az összes mérkőzést, és kereshetnek közöttük különböző szempontok szerint (pl. dátum, eredmény).

2. Felhasznált Megvalósítás

A rendszer megvalósításához az alábbi technológiákat és eszközöket alkalmazom:

2.1. Adatbázis

Szerver: Az alkalmazás adatainak tárolására MySQL relációs adatbázis-kezelőt használok.

MySQL egy népszerű, megbízható és skálázható adatbázis-kezelő, amelyet széles körben alkalmaznak webalkalmazásokhoz.

Az adatbázis szerkezetét az előzőekben bemutatott táblák szerint terveztem meg, és az adatbázis műveleteket (CRUD műveletek) kézzel kell megírni SQL-ben a backend alkalmazásban.

2.2. Alkalmazás Fejlesztési Nyelvek és Keretrendszerek

Backend: A backend fejlesztéséhez Java Spring Boot keretrendszert használok.

Spring Boot egy Java alapú, nyílt forráskódú keretrendszer, amely segít az alkalmazás gyors fejlesztésében és telepítésében.

Az alkalmazás logikája, mint a felhasználók regisztrációja, bejelentkezése, versenyzők, mérkőzések és bajnokságok kezelése, a Spring Boot segítségével valósul meg.

Az adatbázis műveletekhez JDBC (Java Database Connectivity) kapcsolatot használok, ahol az SQL parancsokat kézzel írom meg a táblák közötti műveletekhez (pl. beszúrás, frissítés, törlés).

Frontend: A frontend fejlesztéséhez Thymeleaf-et használok.

Thymeleaf egy modern, Java alapú template motor, amely lehetővé teszi a dinamikus HTML oldalak generálását, és szoros integrációt biztosít a Spring Boot alkalmazásokkal.

A frontend HTML sablonok a felhasználói felületet biztosítják, és lehetővé teszik a dinamikus adatmegjelenítést, mint például:

A versenyzők, mérkőzések és bajnokságok listázása.

A regisztrációs és bejelentkezési űrlapok megjelenítése.

A felhasználói interakciók kezelése és űrlapok feldolgozása.

2.3. Authentikáció és Jogosultságok Kezelése

Authentikáció: Az autentikációs folyamat során session alapú autentikációt alkalmazok.

A felhasználók bejelentkezését követően egy session ID kerül generálásra, amely biztosítja, hogy a felhasználó azonosítva legyen a rendszer további műveletei során. A session ID-t a backend a frontenddel való kommunikáció során cookie-ban tárolja.

Jogosultságok kezelése: Az autentikációs folyamat során a felhasználó jogosultságait is kezeljük:

Adminisztrátorok: Az adminisztrátorok hozzáférhetnek minden funkcióhoz, például új felhasználók regisztrálásához, bajnokságok és versenyzők kezeléséhez.

Nézők: A nézők csak a nyilvános adatokat érhetik el, mint például mérkőzések és versenyzők listája.

2.4. Adatbázis Műveletek

Az adatbázis műveletekhez JDBC-t használok:

A SELECT, INSERT, UPDATE, és DELETE SQL parancsokat alkalmazok az adatbázis kezelésére.

Az adatbázis kapcsolatot JDBC-n keresztül kezelem.

A Spring Boot alkalmazásban egyéni DAO (Data Access Object) réteget hozok létre az adatbázis műveletek kezelésére, amely biztosítja, hogy az adatbázis-interakciók tisztán és biztonságosan történjenek. (Ezek a Repositoryk)

Példa SQL műveletek:

Felhasználó regisztrációja (INSERT)

```
INSERT INTO Felhasznalo (felhasznalonev, jelszo, nev, jogosultsag)
VALUES (?, ?, ?, ?);3
```

Felhasználó bejelentkezése (SELECT)

```
SELECT * FROM Felhasznalo WHERE felhasznalonev = ? AND jelszo = ?;
```

Versenyző rögzítése (INSERT)

```
INSERT INTO Versenyzo (nev, szul_datum, szul_hely, allampolgarsag, gyozelmi_arany, aktiv)
VALUES (?, ?, ?, ?, ?, ?);
```

Versenyző adatainak módosítása (UPDATE)

UPDATE Versenyzo SET gyozelmi_arany = ?, aktiv = ? WHERE vId = ?;

2.5. Webszolgáltatások és API-k

A backend és a frontend közötti kommunikációt REST API-k biztosítják. Az alkalmazás lehetőséget biztosít arra, hogy a frontend HTTP kéréseket küldjön a backendhez, például:

POST /login - Felhasználó bejelentkezése.

POST /register - Felhasználó regisztrációja.

Összegzés

A rendszer a következő technológiákat és megoldásokat használja:

Szerver: MySQL adatbázis

Backend: Java Spring Boot keretrendszer, JDBC adatbázis műveletek

Frontend: Thymeleaf templát motor, HTML, CSS

Authentikáció és Jogosultságok Kezelése: Session alapú autentikáció

Az alkalmazás képes a felhasználók regisztrációjára, bejelentkezésére, a versenyzők, mérkőzések és bajnokságok kezelésére

Összetett lekérdezések:

Az alábbiakban összetett SQL lekérdezést mutatok be, amelyek a rendszer működését támogatják. Minden lekérdezés tartalmazza az SQL kódot, a lekérdezés célját és azt, hogy melyik fájlban található a megvalósítása.

1. Lekérdezés: Versenyzők listázása mérkőzések alapján

Leírás: Ez a lekérdezés lekéri a versenyzőket, akik részt vettek az adott mérkőzésen. A lekérdezés a versenyzők minden adatát listázza.

SQL Kód:

SELECT *

```
FROM merkozesek m

JOIN resztvetel r ON m.mId = r.mId

JOIN versenyzo v ON r.vId = v.vId

WHERE m.mId = ?;
```

Magyarázat:

- A lekérdezés először lekéri a Merkozesek és Resztvetel és Versenyzo táblákat a versenyzők által játszott mérkőzésekhez.
- A WHERE feltétel meghatározza, hogy melyik mérkőzéseket.

Fájl: MerkozesekRepository.java