szia hello mindenért kopócsy anyját szidjátok nyugodtan

1. A Backend (Node.js / Express)

A backend felel az adatbázis-műveletekért, az üzleti logikáért és az API végpontok biztosításáért, amelyeket a frontendek használnak.

**Projekt Mappa Struktúrája:**

* CONFIG: Itt található a konfigurációs fájl az adatbázis-kapcsolathoz (db.js). Az adatokat (felhasználónév, jelszó, stb.) a .env fájlból olvassa be, így a szenzitív információk nincsenek a kódban. **Ehhez nem kell hozzányúlnod.**
* CONTROLLERS: Ezek a fájlok tartalmazzák a fő logikát. Amikor egy kérés beérkezik egy API útvonalra, a router az illetékes controller funkciójának adja át a feldolgozást. A logikai egységeket szétválasztottuk:
* userController.js: Felhasználói műveletek (regisztráció, bejelentkezés).
* movieController.js: Filmekkel kapcsolatos műveletek (CRUD).
* screeningController.js: Vetítésekkel kapcsolatos műveletek (CRUD).

Ez az átláthatóságot szolgálja. Ha új funkciót adsz hozzá (pl. Jegyvásárlás), hozz létre neki egy új controller fájlt.

* bookingController.js:

Helyfoglalásokkal kapcsolatos műveletek (foglalás, listázás, törlés). Ez egy új controller, ami a jegyvásárlási funkciókat kezeli.

* LOGS: Alapértelmezetten üres. Ide kerülnek a naplófájlok, ha a rendszer hibát észlel a futás során.
* MIDDLEWARE: A middleware-ek olyan funkciók, amik a controller hívása előtt futnak le. Itt található például az auth.js, ami ellenőrzi a bejövő kérések JWT tokenjét. Ha a hitelesítés nem sikeres, a kérés nem jut el a controllerhez.
* MODELS: Itt definiáljuk az adatbázis táblák sémáit a Sequelize segítségével (pl. User.js, Movies.js, Screenings.js). A controllerek ezeket a modelleket használják az adatbázis-műveletekhez.
* index.js: A modellek központi belépési pontja. Ez a fájl felelős az összes modell betöltéséért és a köztük lévő kapcsolatok (associations) létrehozásáért, elkerülve a körkörös hivatkozásokat.
* ROUTES: [User.js](http://user.js), [Movi](http://movies.js)es.js, [Screenings.js](http://screenings.js)  
   [Booking.js](http://booking.js): Az új modell, amely a felhasználók, vetítések és a lefoglalt helyek közötti kapcsolatot tárolja az adatbázisban.
* ROUTES: Az api.js fájl definiálja az összes API útvonalat (végpontot), és összerendeli őket a megfelelő controller funkciókkal. **Új fájlt itt ne hozz létre.**
* UTILS: Segédfüggvényeket tartalmaz, jelenleg a logolás beállítását (logger.js). **Ehhez sem kell hozzányúlnod.**
* .env: **Fontos!** Ebben a fájlban kell megadnod a saját, helyi környezetedhez tartozó beállításokat (adatbázis elérés, JWT titkos kulcs), mielőtt elindítod a szervert.
* server.js: A backend alkalmazás belépési pontja. Elindítja a szervert, beállítja a middleware-eket (pl. CORS, limiter) és összeköti a /api útvonalat a routes/api.js fájllal.

**Fontosabb Backend Funkciók:**

**Rate Limiter:** Egy IP-címről 15 percen belül maximum 100 kérést engedélyez a túlterheléses támadások kivédésére.

**CORS (Cross-Origin Resource Sharing):** Beállítja, hogy mely forrásokból (domainekből) fogadjon el a szerver kéréseket.

**Globális Hibakezelés:** Egy speciális app.use middleware biztosítja, hogy ha egy kérés közben váratlan hiba történik, a szerver ne omoljon össze, hanem egy standard 500-as (Internal Server Error) hibával válaszoljon.

# 

# 

# **API Útvonalak** (routes/api.js):

* **FELHASZNÁLÓK**
* POST /users/loginCheck: Bejelentkezés. Visszaadja a felhasználó adatait és egy JWT tokent, vagy hibaüzenetet.
* POST /users/register: Új felhasználó regisztrációja.
* GET /users/getUser/:accountId: Adott felhasználó adatainak lekérése (jövőbeli admin funkciókhoz). auth middleware védi.
* **FILMEK**
* GET /movies/movies: Visszaadja az összes filmet.
* GET /movies/movie-by-id/:movieId: Egy film adatainak lekérése ID alapján.
* POST /movies/movies: Új film létrehozása (admin).
* PUT /movies/movies/:movieId: Film frissítése ID alapján (admin).
* DELETE /movies/movies/:movieId: Film és a hozzá tartozó vetítések törlése ID alapján (admin).
* **VETÍTÉSEK**
* GET /screenings/screenings: Visszaadja az összes vetítést a hozzájuk tartozó film címével.
* GET /screenings/movie/:movieId: Egy adott filmhez tartozó összes vetítés lekérése.
* POST /screenings/screenings: Új vetítés létrehozása (admin).
* GET /screenings/date/:date: Vetítések szűrése dátum alapján (YYYY-MM-DD formátumban).
* GET /screenings/details/:id: Egy adott vetítés részletes adatai, beleágyazva a teljes film objektumot.
* **FOGLALÁSOK**
  + POST /bookings: Új helyfoglalás létrehozása egy adott vetítésre. A kérés törzsében screeningId-t és seat azonosítót vár.
  + GET /users/me/bookings: A bejelentkezett felhasználó összes foglalásának lekérése. A válasz tartalmazza a vetítés és a film részletes adatait is.
  + DELETE /bookings/:id: Egy konkrét foglalás törlése az azonosítója alapján. A felhasználó csak a sajátját törölheti, kivéve ha admin.

2. A Web Frontend (JavaScript/HTML/CSS)

Ez a felület a mozi látogatói számára készült. Lehetővé teszi a filmek böngészését, regisztrációt, bejelentkezést és a vetítések megtekintését.

* **index.html**: A projekt egyoldalas (Single Page Application - SPA) jellegű, minden tartalom itt jelenik meg dinamikusan.
* **js/api.js**: Itt találhatóak azok a fetch hívások, amelyek a backend API-val kommunikálnak. Minden funkció (pl. createUser, getMovies) egy-egy API végpontot hív meg.
* **js/main.js**: A frontend logikájának központja. Ez a fájl kezeli a DOM manipulációt, az eseménykezelést (kattintások, űrlapküldés), a modális ablakok megjelenítését és a felhasználói felület frissítését a bejelentkezési állapot alapján. A vetítések “Részletek” gombjának logikája mostantól egy interaktív modált nyit meg, és kezeli a foglalási űrlap elküldését.
* **Keresés és Szűrés:**
  + **Filmek keresése:** Kliensoldalon történik. Az összes film betöltődik a allMoviesCache változóba, és a keresőmezőbe gépelve a JavaScript szűri a már betöltött adatokat. Ez gyors és nem terheli a szervert.
  + **Vetítések szűrése:** A szűrés szerveroldali. A felhasználó kiválaszthat egy filmet vagy egy dátumot, és a frontend egy specifikus API kérést küld a backendnek, ami a szűrt eredménnyel tér vissza.
* **profil.html**: Egy egyszerű oldal, ami a bejelentkezett felhasználó adatait jeleníti meg. Az adatokat a böngésző sessionStorage-éből olvassa ki, így nem igényel külön backend kérést. Kiegészült egy szekcióval, amely listázza a felhasználó aktív foglalásait, és lehetőséget biztosít azok törlésére.
* **js/**[**profil.js**](http://profil.js): A profil oldalhoz tartozó JavaScript logika. Mostantól API hívással lekéri és megjeleníti a felhasználó foglalásait, valamint kezeli a törlés gombok eseményeit.

[Repository](https://github.com/csongorcsongorcsongorcsongor/kurvaanyadkopocsy)

3. A WPF Asztali Kliens

Ez egy C# nyelven írt, Windows asztali alkalmazás, amely elsősorban adminisztrációs feladatokra szolgál. Lehetővé teszi a filmek és vetítések teljes körű kezelését (CRUD).

**Felépítés és Működés:**

A projekt egyetlen ablakból (MainWindow) áll, amely dinamikusan változtatja a tartalmát a felhasználói interakciók és a bejelentkezési állapot alapján.

* MainWindow.xaml: Leírja az ablak vizuális felépítését (panelek, gombok, listák).
* MainWindow.xaml.cs: Tartalmazza a teljes logikát:
  + **Modellek (DTO-k):** A fájl alján találhatók azok az osztályok (User, Movie, Screening stb.), amelyek a szerverről érkező JSON adatok C# objektumokká alakítását szolgálják.
  + **API Kommunikáció**: A HttpClient osztály segítségével kommunikál a backend API-val.
  + **Adatkezelés**: A RefreshAllDataAsync a központi adatfrissítő metódus, amely minden film- vagy vetítés-módosító művelet után lefut, biztosítva, hogy a felület mindig naprakész legyen. Az adatokat listákban (allMoviesCache, allScreeningsCache) tárolja a gyors elérés érdekében.
  + **Eseménykezelők**: A ...\_Click és ...\_SelectionChanged nevű metódusok reagálnak a felhasználói interakciókra (pl. gombnyomás, elem kiválasztása a listából).
  + **UI Logika**: Az UpdateUIVisibility felel azért, hogy a bejelentkezett felhasználó jogosultságai (pl. admin) alapján a megfelelő gombok és panelek jelenjenek meg vagy tűnjenek el.
* BookingWindow.xaml / [BookingWindow.xaml.cs](http://bookingwindow.xaml.cs):  
  Egy új, dedikált ablak, ami a “Vetítés Részletei” gombra kattintva jelenik meg. Tartalmazza a vetítés és a film adatait, valamint egy beviteli mezőt és egy gombot a helyfoglalásához. A foglalási logikát és az API hívást ez az ablak kezeli.

* ProfileWindow.xaml / [ProfileWindow.xaml.cs](http://profilewindow.xaml.cs): Egy új, a “Profilom” gombra megjelenő ablak. Betöltődéskor API hívással lekéri a bejelentkezett felhasználó összes foglalását, és egy listában megjeleníti azokat. Minden foglalás mellet egy “Törlés” gomb is található.

**Főbb Funkciók:**

* **Felhasználókezelés**: Regisztráció és bejelentkezés.
* **Filmek Kezelése (Admin):**
  + Filmek listázása és keresése.
  + Részletes adatok megtekintése.
  + Új film létrehozása.
  + Meglévő film szerkesztése.
  + Film törlése.
* **Vetítések Kezelése (Admin):**
  + Vetítések listázása és szűrése film alapján.
  + Új vetítés hozzáadása egy meglévő filmhez.
  + Vetítések szűrése film és dátum alapján.
  + Vetítések részletes adatainak megtekintése (egy felugró ablakban).
* **Foglalások Kezelése (Felhasználó és Admin):**
  + Vetítések részletes adatainak megtekintése és helyfoglalás:  
    A vetítések listájában a “Részletek” gomb egy új ablakot nyit meg, ahol a bejelentkezett felhasználó helyet foglalhat.
  + Felhasználói profil és foglalások megtekintése:  
    A főablakban egy “Profilom” gomb elérhetővé válik bejelentkezés után, ami egy új ablakot nyit meg.
  + Foglalások listázása és törlése:  
    A profil ablakban a felhasználó listázva látja az összes foglalását, és a “Törlés” gombbal lemondhatja azokat.

[Repository](https://github.com/csongorcsongorcsongorcsongor/kurvaanyadkopocsywpf)

[hét 3](https://github.com/Koxonium/plojekt3het3)

3. hét 3. tanuló + dolgok

**1. Backend (Node.js + Express + Sequelize)**

**Funkciók**

* **Top 5 film** foglalásszám szerint:  
  GET /api/movies/top  
  → Visszaadja az 5 legtöbbet foglalt filmet.
* **Felhasználói statisztika**:  
  GET /api/users/:id/stats  
  → Visszaadja, hány foglalása van a megadott felhasználónak.
* **Vetítés statisztika (foglaltsági arány)**:  
  GET /api/screenings/:id/stats  
  → Visszaadja a foglalt helyek számát, összes ülőhelyet, és a foglaltság százalékát.

**2. WPF alkalmazás (admin és user felület)**

**Admin statisztikák (Top Movies)**

* LoadTopMoviesAsync()  
  → Lekéri és megjeleníti az 5 legnépszerűbb filmet.

**Vetítés statisztika (foglaltság)**

* LoadScreeningStatsAsync()  
  → Vetítés foglalt/összes hely és százalékos arány megjelenítése.

**Felhasználói profil**

* A ProfileWindow betölti a felhasználó foglalásait.
* Foglalások törlése gombbal.
* LoadUserBookingsAsync() + DeleteBookingButton\_Click() metódusok.

**3. Webes frontend (HTML + JS)**

**5 Legtöbbet lefoglalt film**

* a lefoglalások alapján az 5 legtöbbel lefoglalt filmet rangsorolja
* a rangsor frissítéséhet frissíteni kell az oldalt, nem kaptak a fejlesztők elég fizetést h megcsinálják (main.js)

**Felhasználói statisztika (foglalásszám)**

* Lekéri: GET /users/:id/stats (profil.js)
* Megjeleníti a foglalásszámot a profileBookingCount elemben.

**Vetítés statisztika felülete (input + gomb)**

* Egyszerű UI:
  + Beviteli mező vetítés ID-nek.
  + Gomb a statisztika lekéréséhez.
  + Kimeneti mező százalékos értékkel. (stats.js)