Projektmunka Dokumentációja

Mozi helyfoglaló webalkalmazás

Premozi

Premontrei Szakgimnázium és Technikum

Keszthely

2025

Szoftverfejlesztő és -tesztelő szak

5-0613-12-03

Készítette:

Bató Bence, Magyarósi Bálint

Tartalom

[1. Bevezetés 4](#_Toc195302803)

[2. Fontos programok 5](#_Toc195302804)

[3. Technológiai felépítés 5](#_Toc195302805)

[4. NuGet csomagok 6](#_Toc195302806)

[Biztonság és Hitelesítés 6](#_Toc195302807)

[Adatbázis és Entity Framework 6](#_Toc195302808)

[API és JSON Kezelés 6](#_Toc195302809)

[Tesztelés 6](#_Toc195302810)

[Egyéb Segédeszközök 7](#_Toc195302811)

[5. Adatbázis szerkezete 7](#_Toc195302812)

[Adatbázis modell 8](#_Toc195302813)

[Film tábla 9](#_Toc195302814)

[Terem tábla 9](#_Toc195302816)

[Vetítés tábla 9](#_Toc195302817)

[Foglalt székek 10](#_Toc195302818)

[VetítésSzékek tábla 10](#_Toc195302819)

[Httplogs 10](#_Toc195302820)

[Székek tábla 11](#_Toc195302821)

[FoglalásAdatok tábla 11](#_Toc195302822)

[Users tábla (felhasználók) 12](#_Toc195302824)

[Images tábla (képek) 12](#_Toc195302825)

[jegytipus tábla (jegyárak 12](#_Toc195302826)

[Email2FACodes tábla (kétlépcsős azonosítás kódok) 13](#_Toc195302827)

[\_\_EFMigrationsHistory tábla (migrációs előzmények) 13](#_Toc195302829)

[Összefoglalás 13](#_Toc195302830)

[Kapcsolatok 13](#_Toc195302831)

[6. REST API funkciók 14](#_Toc195302832)

[Felhasználókezelés (AuthController) 15](#_Toc195302833)

[Foglalások kezelése (FoglalasController) 16](#_Toc195302834)

[Termek kezelése (TeremController) 16](#_Toc195302835)

[Vetítések kezelése (VetitesController) 17](#_Toc195302836)

[Képek kezelése (ImageController) 17](#_Toc195302836)

[7. Weboldal működése 18](#_Toc195302837)

[8. NUnit teszt 24](#_Toc195302838)

[AuthController tesztek 24](#_Toc195302839)

[Regisztráció tesztelése (Register\_ReturnsOk\_WhenSuccessful): 24](#_Toc195302840)

[Email megerősítés tesztelése (ConfirmEmail\_ReturnsOk\_WhenTokenIsValid): 24](#_Toc195302841)

[Felhasználói adatok lekérése (GetUser\_ReturnsUser\_WhenAuthenticated): 24](#_Toc195302842)

[Bejelentkezés állapotának ellenőrzése 24](#_Toc195302843)

[Felhasználó törlésének jogosultságvizsgálata 24](#_Toc195302844)

[FilmController tesztek 25](#_Toc195302845)

[Filmlekérdezési tesztek 25](#_Toc195302846)

[Filmkezelési műveletek 26](#_Toc195302847)

[FoglalasController tesztek 27](#_Toc195302848)

[Jegytípus kezelés tesztjei 27](#_Toc195302849)

[Foglalás lekérdezési tesztek 27](#_Toc195302850)

[CRUD műveletek tesztjei 28](#_Toc195302851)

[TeremController tesztek 29](#_Toc195302852)

[VetitesController tesztek 29](#_Toc195302853)

[Módosítási műveletek tesztjei 29](#_Toc195302854)

[Törlési műveletek tesztjei 30](#_Toc195302855)

[9. Fejlesztési javaslatok 30](#_Toc195302856)

[10. Összegzés/Reflexió 31](#_Toc195302857)

[11. Források 32](#_Toc195302858)

# Bevezetés

A szakmai vizsgaprojektünk egy mozijegy-foglaló rendszer, amelynek megvalósítását azért választottuk, mert csapatunk egy korábbi iskolai projekt során már dolgozott mozi témájú alkalmazáson, így ezt az ismeretet tovább tudtuk fejleszteni. A projekt célja egy olyan online rendszer létrehozása, amely lehetővé teszi a felhasználók számára, hogy kényelmesen és gyorsan lefoglalják helyeiket a kívánt vetítésekre.

A rendszer fejlesztése során kiemelt figyelmet fordítottunk a hatékony csapatmunkára és a feladatok megfelelő elosztására. Minden csapattag a saját erősségei és tapasztalatai alapján kapott szerepet a projektben, így biztosítva a munka gördülékenységét és a lehető legjobb eredmény elérését. A kommunikáció és az együttműködés kulcsfontosságú volt, rendszeresen tartottunk megbeszéléseket, ahol megosztottuk az előrehaladásunkat, megbeszéltük a felmerülő problémákat, és közösen kerestünk megoldásokat.

A jegyfoglaló rendszer egy webalapú alkalmazásként működik, amely egy MySQL adatbázisra épül. A felhasználók regisztrálhatnak, bejelentkezhetnek, és a rendszer segítségével kiválaszthatják a kívánt vetítést, majd lefoglalhatják helyeiket. Az adatok kezelése és a foglalási folyamat biztosítása egy REST API segítségével történik, amely biztosítja az adatok biztonságos és gyors elérését.

Dokumentációnk részletesen bemutatja a rendszer felépítését, működését és technológiai hátterét. Ismertetjük az alkalmazott fejlesztési eszközöket, az adatbázis szerkezetét, valamint a rendszer főbb funkcióit.

Összességében a projekt nem csak technikai kihívást jelentett számunkra, hanem lehetőséget adott arra is, hogy fejlesszük együttműködési készségeinket, megtanuljuk a hatékony munkamegosztás fontosságát, és valós fejlesztői környezetben szerezzünk tapasztalatot. Bízunk benne, hogy a mozijegy-foglaló rendszerünk egy jól működő, felhasználóbarát megoldást kínál a felhasználók számára.

# Fontos programok

* Visual Studio 2022

A fő fejlesztői környezet, ahol írtuk, debugoltuk és futtattuk a programot, C# és ASP.NET keretrendszer használatával.

* XAMPP

A program adatbázisa egy lokális szerveren van csak jelenleg, ezt a XAMPP-al és a PhpMyAdmin-nal kezeljük (Apache, MySQL, PHP).

* Discord

Kommunikációs platformként szolgált, ahol gyorsan és egyszűen megbeszéltük a feladatokat, megosztottuk a frissítéseket és tartottuk a kapcsolatot.

* GitHub

A verziókövetéshez és a csapatmunka koordinálásához használtuk, ahol a kódot tároltuk, megosztottuk és együtt dolgoztunk rajta bárhonnan.

* ProtonVPN

A ProtonVPN segítségével tudtunk elérni bármilyen segítséget az interneten, ami az iskolai hálózaton néha le volt tiltva. (pl.: Stackoverflow, Discord, W3Schools)

# Technológiai felépítés

* **Backend:** ASP.NET Core 8.0
* **Frontend:** React + Vite
* **Adatbázis:** MySQL
* **Adatbázis kommunikáció:** REST API

# NuGet csomagok

## Biztonság és Hitelesítés

BCrypt.Net-Next (4.0.3): Jelszavak biztonságos hash-elésére és tárolására (pl. regisztráció, bejelentkezés).

Microsoft.AspNetCore.Authentication.JwtBearer (8.0.2): JWT (JSON Web Token) alapú hitelesítés implementálásához API-okban.

Microsoft.IdentityModel.JsonWebTokens (8.6.1): JWT tokenek generálásához és ellenőrzéséhez.

## Adatbázis és Entity Framework

Microsoft.EntityFrameworkCore (8.0.2): ORM keretrendszer adatbázis-kezeléshez .NET-ben.

Pomelo.EntityFrameworkCore.MySql (8.0.2): MySQL támogatás Entity Framework Core-hoz.

MySql.Data (8.0.20) & MySql.Data.EntityFramework (8.0.20): MySQL adatbázis-kapcsolat létrehozásához és kezeléséhez.

Microsoft.EntityFrameworkCore.Design (8.0.2) & Tools (8.0.2): Adatbázis migrációk és scaffolding (pl. dotnet ef parancsok).

## API és JSON Kezelés

Microsoft.AspNetCore.Mvc.NewtonsoftJson (8.0.2): JSON serializálás/deserializálás testreszabásához (pl. dátumformátumok, ciklikus referenciák).

Swashbuckle.AspNetCore (6.6.2): Swagger/OpenAPI dokumentáció generálásához .

## Tesztelés

NUnit (4.3.2) & NUnit3TestAdapter (5.0.0): Unit tesztek írásához és futtatásához.

xunit (2.9.3) & xunit.runner.visualstudio (3.0.2): Alternatív tesztkeretrendszer CI/CD folyamatokhoz.

Moq (4.20.72): Mock objektumok létrehozásához unit tesztekben.

Microsoft.NET.Test.Sdk (17.13.0): Tesztelési keretrendszer támogatás.

## Egyéb Segédeszközök

Microsoft.AspNetCore.SpaProxy (8.0.2): SPA (pl. React, Vite) fejlesztéshez proxy szerverrel.

Microsoft.VisualStudio.Web.CodeGeneration (8.0.2): Automatikus kódgenerálás (pl. scaffolding).

SendGrid (9.29.3): E-mail küldés (pl. regisztrációs megerősítések, jelszó-visszaállítás).

SixLabors.ImageSharp (3.1.7): Képek feldolgozására (pl. átméretezés, formátumkonverzió).

System.Linq.Async (6.0.1): Aszinkron LINQ műveletek (pl. IAsyncEnumerable kezelés).

# Adatbázis szerkezete

Az adatbázis a következő táblákból áll:

* **users**: Felhasználók adatait tárolja.
* **film**: A mozi filmjeinek adatait tartalmazza.
* **vetites**: A vetítések adatait tartalmazza.
* **terem**: A mozitermek adatait tartalmazza.
* **email2facodes**: Emailes kétfaktoros hitelesítés ideiglenes kódjai.
* **foglalasadatok**: Felhasználók által létrehozott foglalások.
* **foglaltszekek**: Foglalt székek adatai (milyen jeggyel, melyik vetítésen).
* **httplogs**: API kérések naplózása (hibakereséshez, nyilvántartáshoz).
* **images**: Filmplakátok és egyéb képek tárolása.
* **jegytipus**: Jegytípusok (pl. diák, felnőtt) és áraik.
* **szekek**: A termekben lévő székek elhelyezkedése és állapota (pl. elérhető-e).
* **vetitesszekek**: Vetítésenként, mely székek foglaltak vagy szabadok.
* **\_\_efmigrationshistory:** EntityFrameworkCore automatikusan generált log táblája

## Adatbázis modell

A képen szöveg, szám, szoftver, Betűtípus látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

Az adatbázisunk a mozi weboldalának jól működő hátteret biztosít, lehetővé teszi a zavartalan működést.

## Film tábla

## 

A moziban vetített filmek adatait tartalmazza (pl. cím, műfaj, leírás).

## Terem tábla



A mozi termeit rögzíti (név, elhelyezkedés).

## Vetítés tábla



A mozi vetítéseinek időpontjait, helyszíneit, műsorát rögzíti.

## Foglalt székek



Rögzíti, hogy ki melyik széket foglalta le és milyen típusú jeggyel. Kompozit kulcsát a hozzá tartozó vetítés szék kompozit kulcsa és a foglalás id-je teszi ki.

## VetítésSzékek tábla



Nyomon követi, hogy mely székek foglaltak vagy szabadok egy adott vetítésen. A felhasználó a szabad székek közül választhat (1-es kódú állapot – a 0-ás kód az elérhetetlen, a 2-es kód a már lefoglalt székeket jelöli). Kompozit kulcsát a hozzá tartozó vetítés id-je és a szék koordinátái teszik ki.

## Httplogs



A rendszer működését naplózza (hibakereséshez, nyilvántartáshoz). A felhasználó számára láthatatlan, de segít a problémák gyors és hatékony megoldásában.

## Székek tábla



A termek székeinek elrendezését tárolja. Ez adja a mintát a vetítések kiosztásának generálásához. Kompozit kulcsát a hozzá tartozó terem id-je és a szék koordinátái teszik ki.

## FoglalásAdatok tábla

## 

A felhasználók által létrehozott foglalásokat tárolja.

## Users tábla (felhasználók)



A felhasználók adatait tárolja. A felhasználó ezzel a fiókkal tud bejelentkezni, jegyet foglalni, és személyre szabott élményt kap.

## Images tábla (képek)



A filmekhez tartozó plakátok és egyéb képek adatait tárolja. A képek a szerveren vannak tárolva, amik futáskor a backenden, statikus úton keresztül érthetőek el.

jegytipus tábla (jegyárak)

A képen szöveg, Betűtípus, képernyőkép látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

Meghatározza a jegyek típusait és árait (pl. diák, felnőtt). A felhasználó ennek alapján választhat kedvezményes vagy normál jegyet.

## Email2FACodes tábla (kétlépcsős azonosítás kódok)

## 

Biztonsági kódokat tárol Email-es kétlépcsős azonosításhoz. A felhasználó ezzel erősítheti meg bejelentkezését.

## \_\_EFMigrationsHistory tábla (migrációs előzmények)



Az EntityFrameworkCore automatikusan generált log tábljája.

## Összefoglalás

A felhasználó a **users** táblán keresztül regisztrál, a **film** és **vetites** táblák alapján kiválaszt egy filmet és időpontot, majd a **szekek** és a **foglaltszekek** segítségével lefoglalja a helyét. A **jegytipus** meghatározza az árát, a **foglalasadatok** pedig tárolja a rendelését. Mindez a mozijegy-vásárlás zökkenőmentes élményét biztosítja.

## Kapcsolatok

**Felhasználó tábla kapcsolatai**

**users ↔ email2facodes**  
Kapcsolat típusa: 1:N (egy felhasználónak több kétlépcsős azonosító kódja lehet)

**users ↔ foglalasadatok**  
Kapcsolat típusa: 1:N (egy felhasználó több foglalást hozhat létre)

**Film tábla kapcsolatai**

**images ↔ film**  
Kapcsolat típusa: 1:N (egy képhez több film is tartozhat, de egy filmnek csak egy képe van)

**film ↔ vetites**  
Kapcsolat típusa: 1:N (egy filmhez több vetítés tartozhat)

**Vetítés tábla kapcsolatai**

**terem ↔ vetites**  
Kapcsolat típusa: N:1 (több vetítés történhet egy teremben)

**vetites ↔ vetitesszekek**  
Kapcsolat típusa: 1:N (egy vetítéshez több szék tartozhat)

**Foglalás táblák kapcsolatai**

**foglalasadatok ↔ foglaltszekek**  
Kapcsolat típusa: 1:N (egy foglalás több széket tartalmazhat)

**foglaltszekek ↔ vetitesszekek**  
Kapcsolat típusa: 1:1 (egy foglalt szék egy vetítés székre hivatkozik)

**foglaltszekek ↔ jegytipus**  
Kapcsolat típusa: N:1 (a foglalt székeknek van jegytípusa)

**Terem tábla kapcsolatai**

**terem ↔ szekek**  
Kapcsolat típusa: 1:N (egy teremben több szék van)

**terem ↔ vetites**  
Kapcsolat típusa: 1:N (egy teremben több vetítés történhet)

# REST API funkciók

A rendszer REST API-t használ az adatok kezelésére. Az alábbi fő végpontok érhetőek el:

## Felhasználókezelés (AuthController)

POST /api/auth/login – Bejelentkezés

POST /api/auth/register – Regisztráció

GET /api/auth/confirm-email – Email cím megerősítése

POST /api/auth/verify-email-2fa – 2FA kód ellenőrzése

POST /api/auth/resend-2fa-code – 2FA kód újraküldése

POST /api/auth/enable-email-2fa – 2FA engedélyezése

POST /api/auth/disable-email-2fa – 2FA letiltása

GET /api/auth/getUser – Felhasználó adatainak lekérése (csak saját adatok)

GET /api/auth/getUserAdmin/{id} – Felhasználó adatainak lekérése (admin, bárki adatai)

GET /api/auth/get – Összes felhasználó lekérése (admin)

POST /api/auth/queryUsers – Felhasználók szűrése (admin, végül nem használt)

PATCH /api/auth/editUser – Felhasználó adatainak módosítása

PATCH /api/auth/editUserAdmin – Felhasználó adatainak módosítása(admin, végül nem használt)

PATCH /api/auth/editPassword – Jelszó módosítása

GET /api/auth/checkIfLoggedIn – Bejelentkezési állapot ellenőrzése

DELETE /api/auth/logout – Kijelentkezés

POST /api/auth/checkIfAdmin – Admin jogosultság ellenőrzése

DELETE /api/auth/deleteUser/{id} – Felhasználó törlése

PATCH /api/auth/change-password – Jelszó megváltoztatása

PATCH /api/auth/force-password-change/{id} – Jelszóváltoztatás kényszerítése (admin)

PATCH /api/auth/change-status/{id} – Felhasználó állapotának módosítása (admin)

POST /api/auth/request-password-reset/{userId} – Jelszó-visszaállítási kérés (admin)

GET /api/auth/verify-password-reset – Jelszó-visszaállítási token ellenőrzése

POST /api/auth/complete-password-reset – Jelszó visszaállítása

**Filmek kezelése (FilmController)**

POST /api/film/query – Filmek szűrése (végül nem használt)

GET /api/film/get – Összes film lekérése

GET /api/film/get/{id} – Film lekérése ID alapján

POST /api/film/add – Új film hozzáadása (admin)

PATCH /api/film/edit – Film módosítása (admin)

DELETE /api/film/delete/{id} – Film törlése (admin)

## Foglalások kezelése (FoglalasController)

GET /api/foglalas/getJegyTipus – Jegytípusok lekérése

GET /api/foglalas/get – Összes foglalás lekérése (admin)

GET /api/foglalas/getByVetites/{vid} – Foglalások lekérése vetítés alapján

GET /api/foglalas/getByUser/{uid} – Foglalások lekérése felhasználó alapján

POST /api/foglalas/add – Új foglalás létrehozása

PATCH /api/foglalas/edit – Foglalás módosítása (végül nem használt)

DELETE /api/foglalas/delete/{id} – Foglalás törlése

## Termek kezelése (TeremController)

GET /api/terem/get – Összes terem lekérése

GET /api/terem/get/{id} – Terem lekérése ID alapján

POST /api/terem/add – Új terem hozzáadása (admin)

PATCH /api/terem/edit – Terem módosítása (admin)

DELETE /api/terem/delete/{id} – Terem törlése (admin)

## Vetítések kezelése (VetitesController)

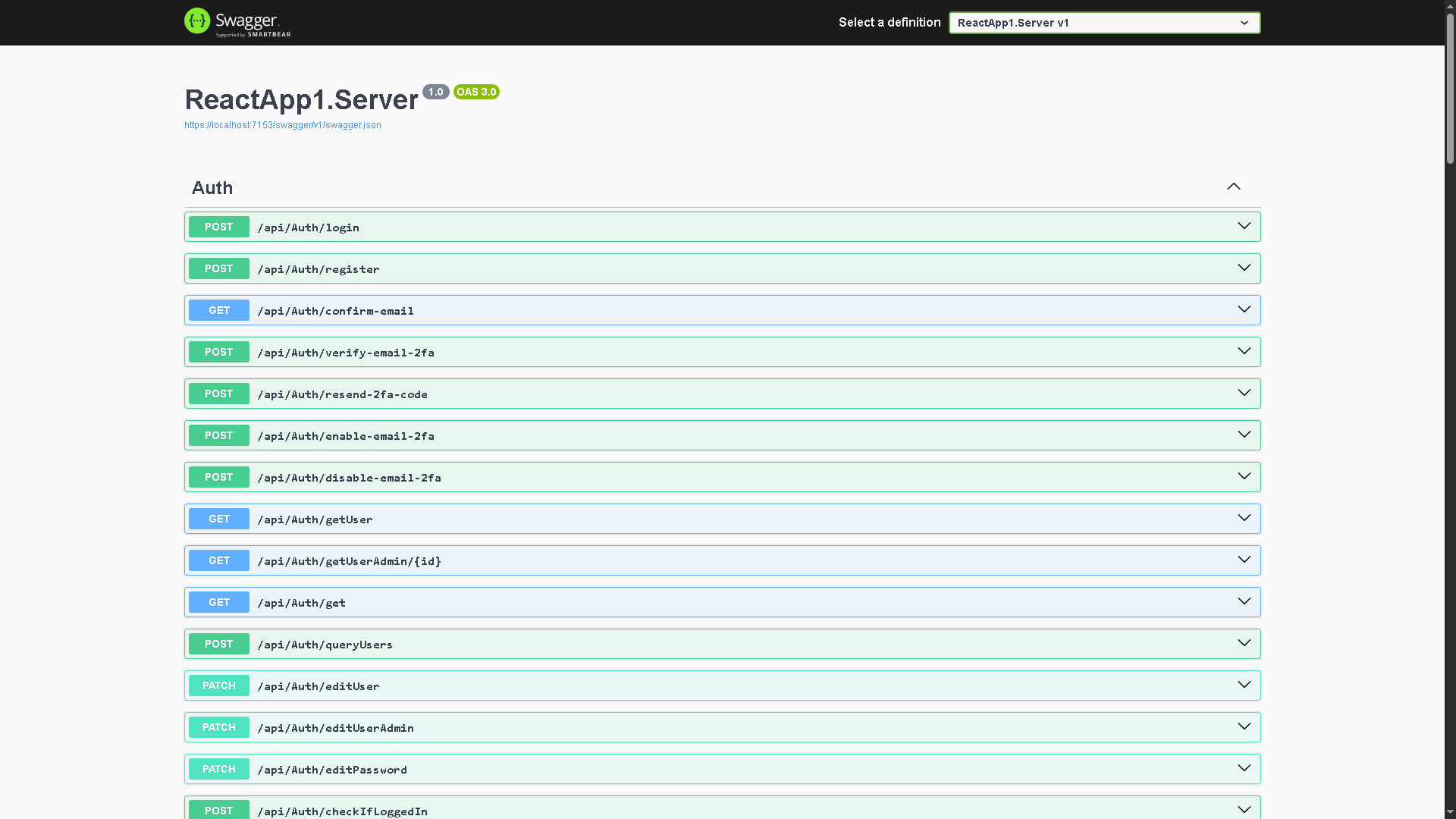
GET /api/vetites/get – Összes vetítés lekérése

GET /api/vetites/get/{id} – Vetítés lekérése ID alapján

POST /api/vetites/add – Új vetítés hozzáadása (admin)

PATCH /api/vetites/edit – Vetítés módosítása (admin)

DELETE /api/vetites/delete/{id} – Vetítés törlése (admin)

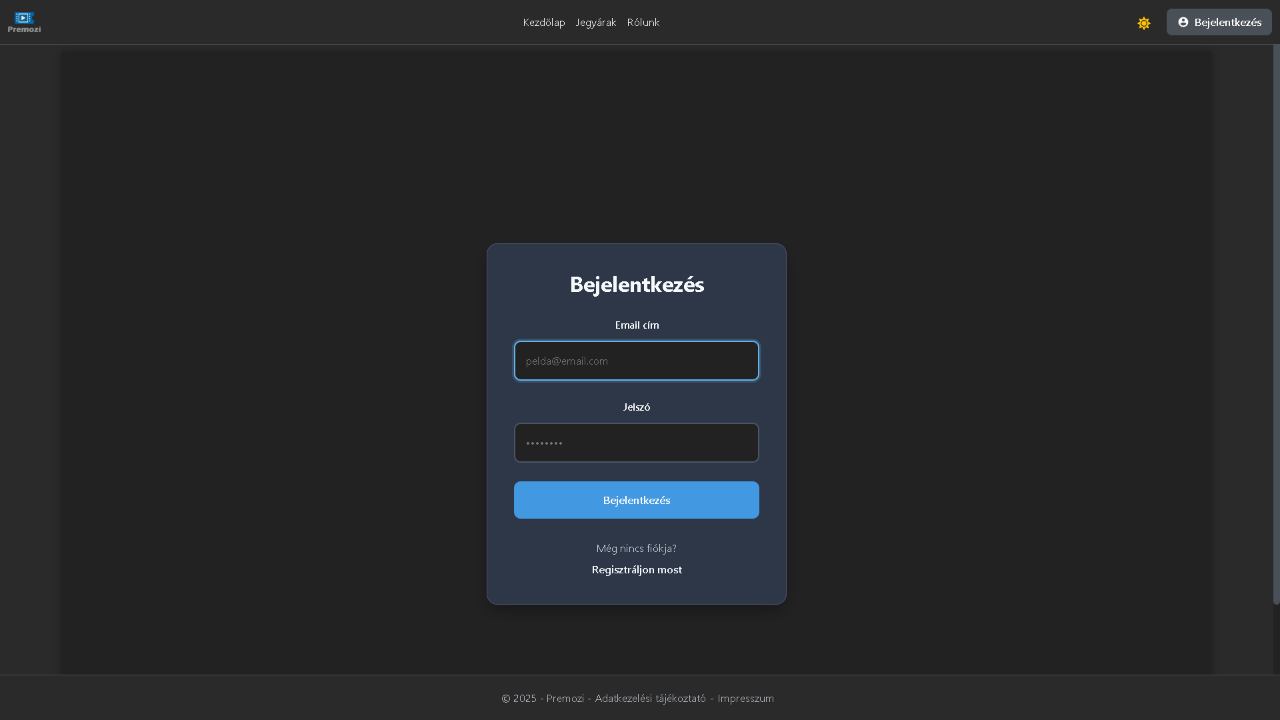
**Képek kezelése (ImageController)** GET /api/images/get – Minden kép adatainak lekérése (admin)  
 POST /api/images/upload – Képfeltöltés (csak az api számára hozzáférhető végpont)  
  
(Képernyőkép a Swagger-ről, amivel teszteltük az API-t)

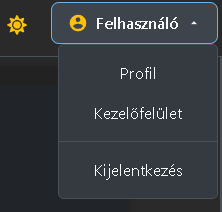
# Weboldal működése A weboldal egy reszponzív felhasználói felületet biztosít, amely az alábbi fő oldalakat tartalmazza:

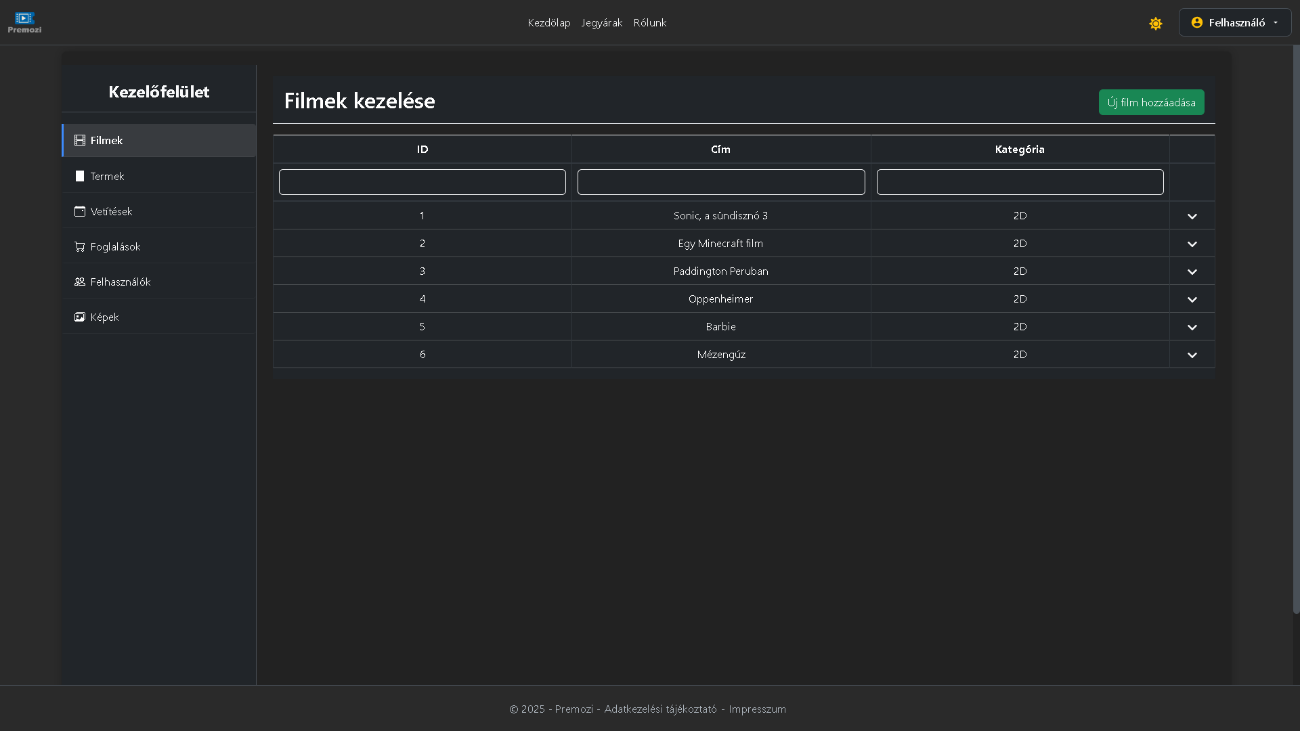
* Bejelentkezés és regisztráció

Elérés út: <https://localhost:60769/account/login>

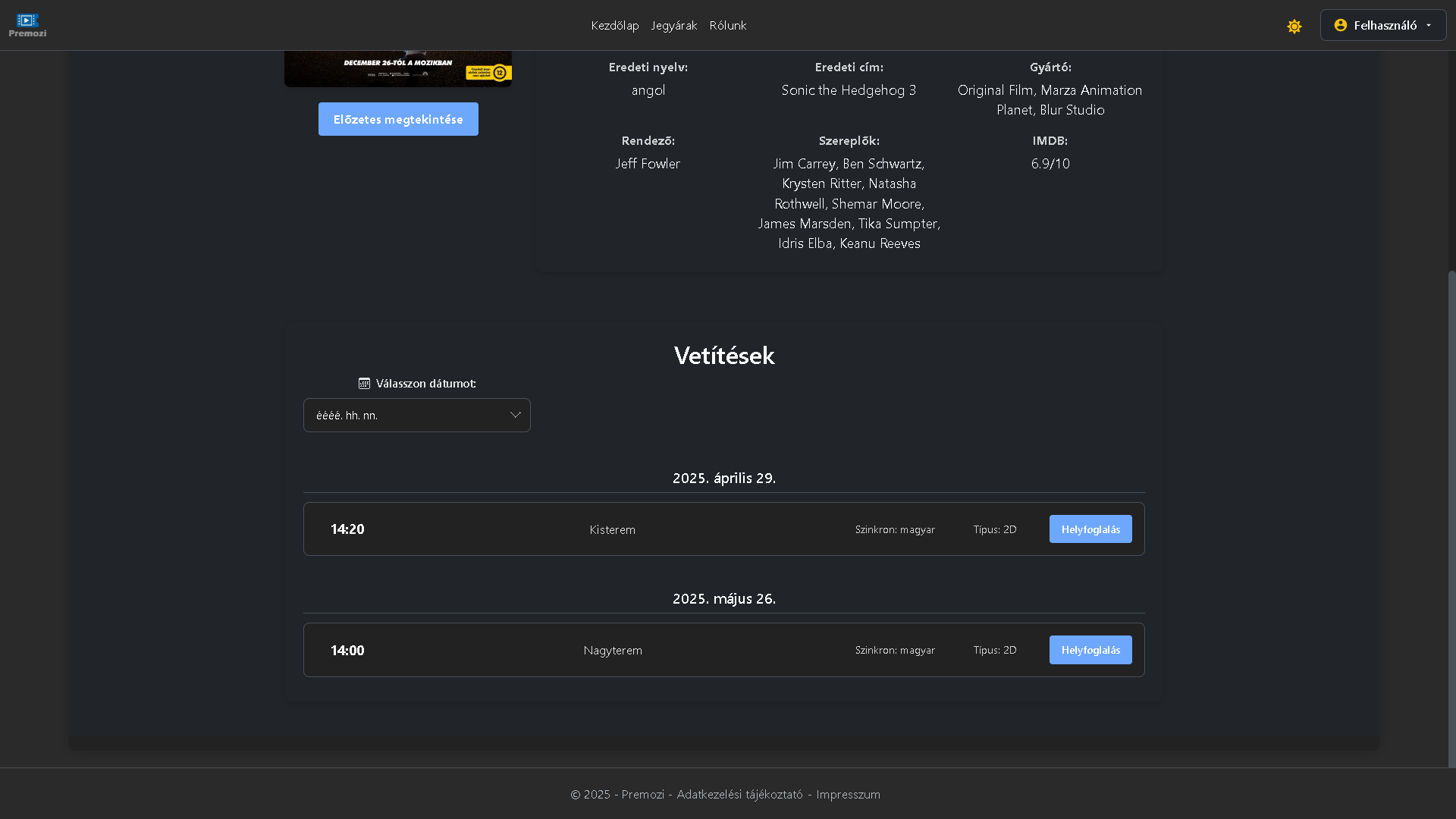
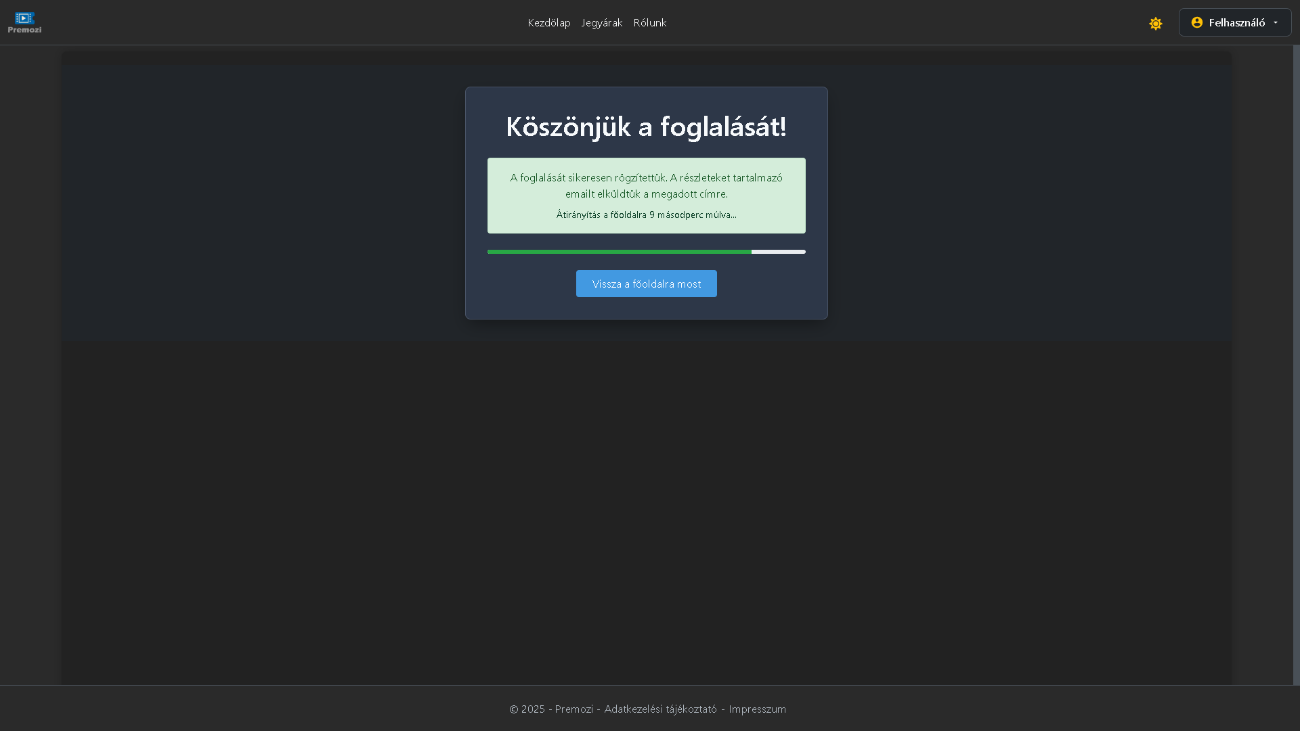
A felhasználó be tud jelentkezni, egy letisztult, bejelentkezési felületen.

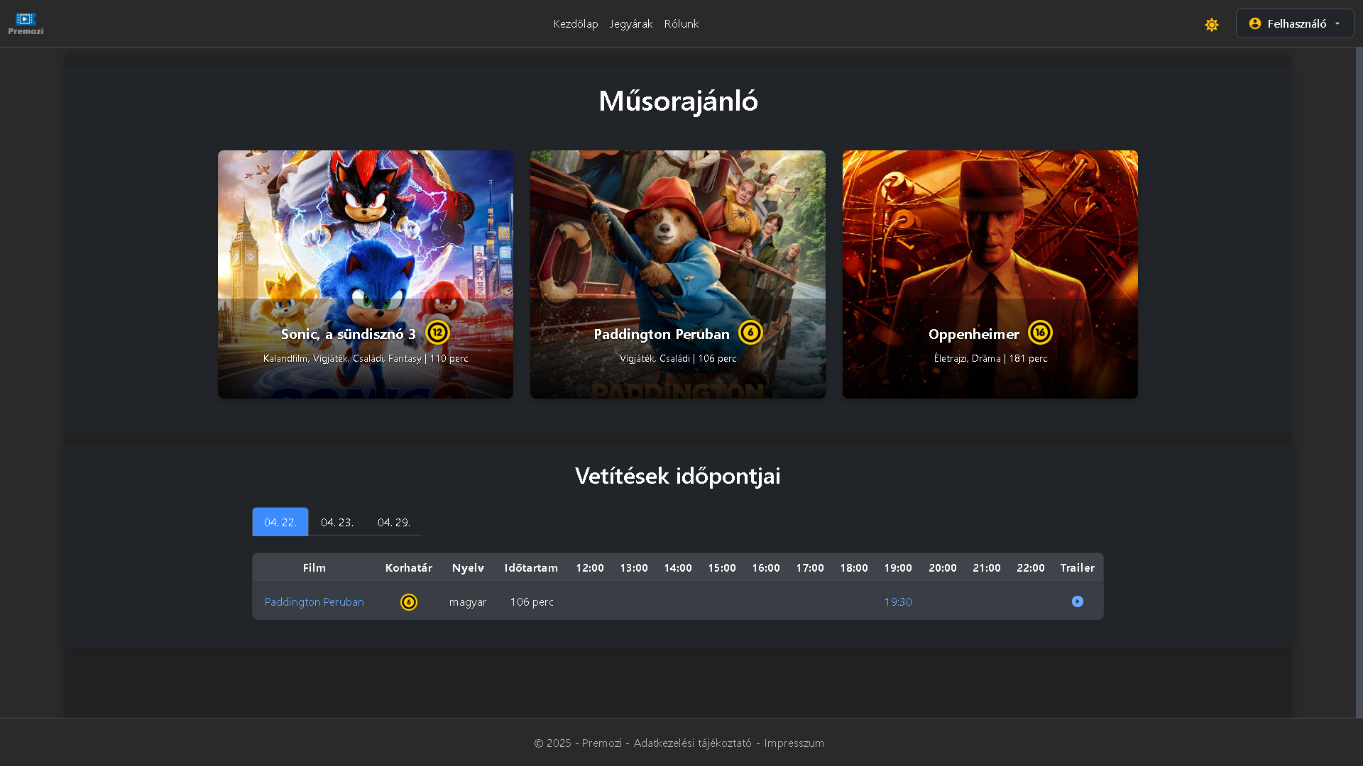
Ha nincs fiókja, az alsó „Nincs fiókja? Regisztráljon most!” -ra kattintva, átírányítódik a regisztrációs felületre, ahol létrehozhatja fiókját, amit utána egy emailben kapott linkkel erősíthet meg.

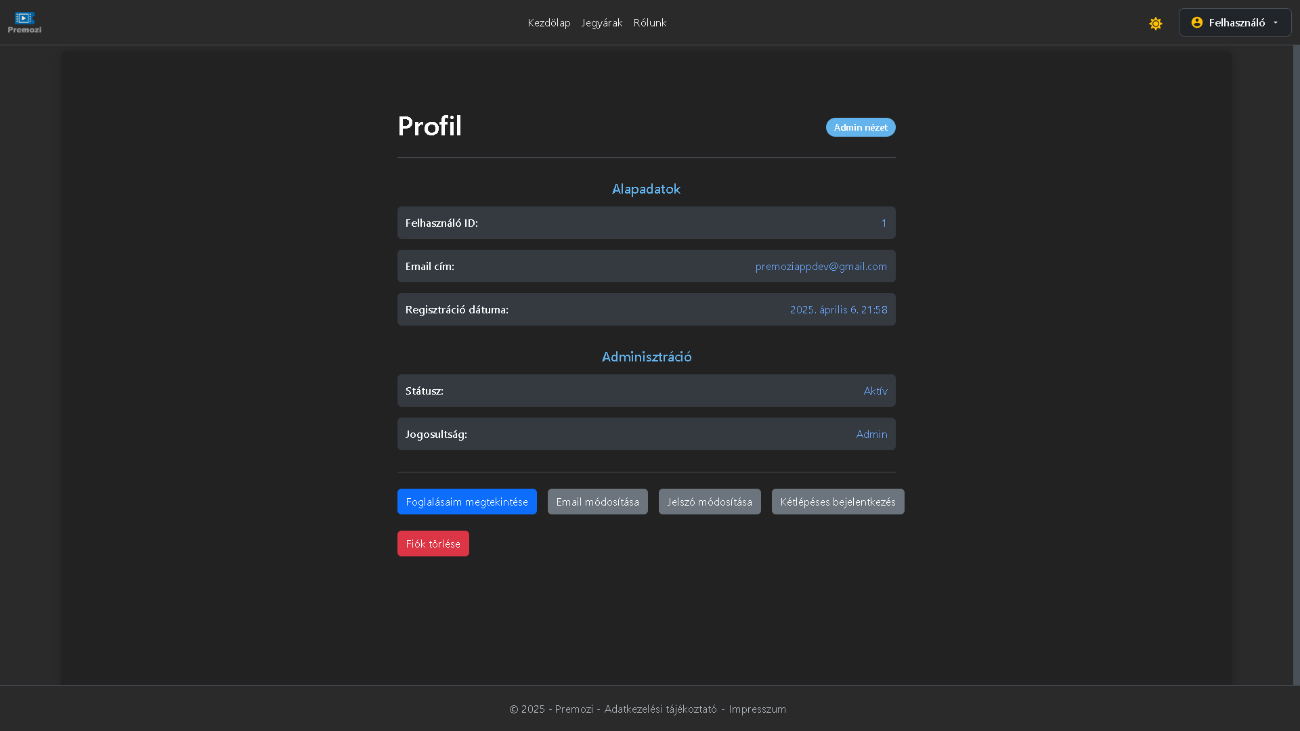
* Ha admin fiókkal jelenkezünk be, az admin fiókkal a következők tehetőek:
  + Felhasználók kezelése/módosítása
  + adatok kezelése/módosítása/törlése

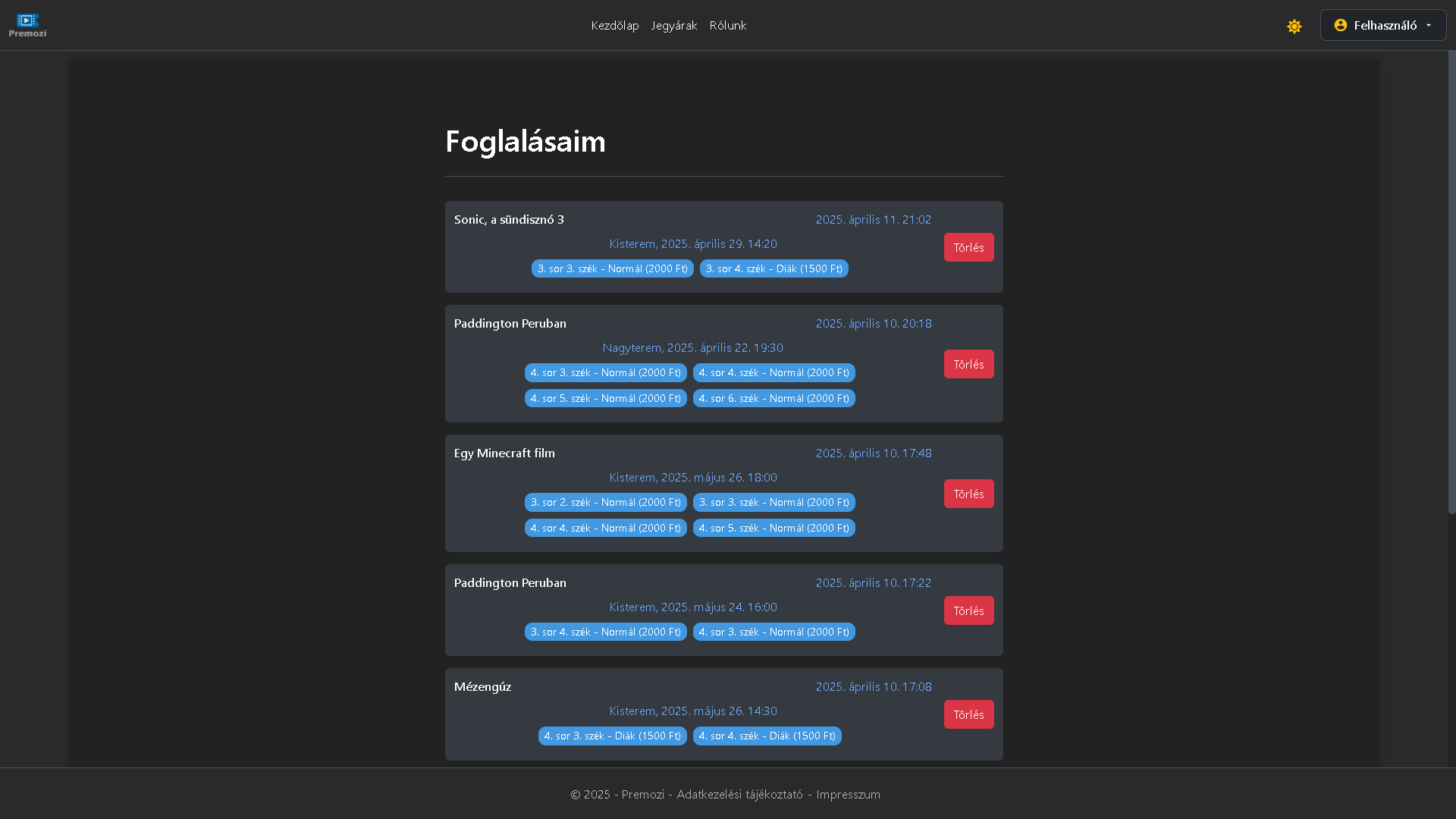
  
Ezeket a funkciókat egy csak az adminisztrátorok számára elérhető kezelőfelületen érhetőek el. A felhasználókat törölni nem tudja az adminisztrátor, de felfüggeszteni igen. A felhasználó maga tudja törölni fiókját.

* Helyfoglalás
  + A helyfoglaláshoz választani kell egy filmet a listán, majd rányomni a helyfoglalás gombra. Ez után megjelenik a helyfoglalás felület, ha be van jelentkezve – ha nincs, a bejelentkező felületre irányít át.   
    A helyfoglaláshoz választani kell egy több széket, szükség esetén beállítani a jegy típusát, majd rákattintani a foglalás gombra. Ezután egy sikeres foglalást jelző oldalra irányít át az oldal, ami 10 másodperc után visszairányít a kezdőlapra. Emellett a sikeres foglalásról emailt is kap a felhasználó.

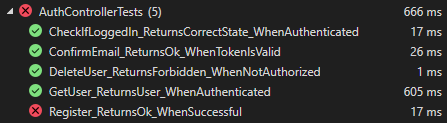
* Vetítés időpontok és filmek listázása
  + A Főoldalunkon megjelenik az összes 30 napon belüli vetés
* Filmmel kapcsolatos adatok megjelenítése
  + Ha kiválasztunk egy filmet, akkor megjelenítődnek a filmmel kapcsolatos adatok
* Helyfoglalás adott vetítésre
  + Ha kiválasztottunk egy filmet, akkor a film adatok alatt megjelennek a vetítés időpontok, ahol, ha kiválasztunk egy időpontot, tudunk helyet foglalni.

* Foglalások megjelenítése
  + A foglalt helyeinket megtudjuk nézni a profilunkon belül a „Foglalásaim megtekintése” gombnál

  
  
  
  
A weboldal az adatokat a REST API-on keresztül kéri le és tölti fel az adatbázisba.

# NUnit tesztek

## AuthController tesztek



## Regisztráció tesztelése (Register\_ReturnsOk\_WhenSuccessful):

* Ellenőrzi, hogy a regisztráció sikeresen lefut-e és megfelelő választ ad-e vissza
* Mockolt szolgáltatásokkal szimulálja a regisztrációs folyamatot
* Ez hibával tér vissza, de valójában jó, mert létezik ilyen fiók.

## Email megerősítés tesztelése (ConfirmEmail\_ReturnsOk\_WhenTokenIsValid):

* Ellenőrzi, hogy érvényes token esetén sikeresen megerősíti-e az email címet
* A token dekódolását és feldolgozását teszteli

## Felhasználói adatok lekérése (GetUser\_ReturnsUser\_WhenAuthenticated):

* Hitelesített felhasználó esetén ellenőrzi a felhasználói adatok helyes visszaadását
* Claim-ek alapján történő azonosítást tesztel

Bejelentkezés állapotának ellenőrzése (CheckIfLoggedIn\_ReturnsCorrectState\_WhenAuthenticated):

* Ellenőrzi, hogy a rendszer helyesen detektálja-e a bejelentkezés állapotát
* A hitelesítési állapot változásait figyeli

Felhasználó törlésének jogosultságvizsgálata (DeleteUser\_ReturnsForbidden\_WhenNotAuthorized):

* Ellenőrzi, hogy nem admin felhasználó ne tudjon más felhasználót törölni
* Jogosultságkezelést tesztel

## FilmController tesztek

A képen szöveg, képernyőkép, Betűtípus, szoftver látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

## Filmlekérdezési tesztek

**QueryFilm\_ReturnsOkResult\_WithValidRequest**

* Ellenőrzi, hogy a QueryFilm metódus helyesen visszaad-e egy filmlistát érvényes szűrőparaméterek esetén.
* Biztosítja, hogy a szűrt lekérdezések helyesen működjenek.

**QueryFilm\_ReturnsBadRequest\_WhenServiceReturnsNull**

* Ellenőrzi a hibakezelést, ha a szolgáltatás null értéket ad vissza.
* Biztosítja, hogy a rendszer ne omoljon össze hibás adatok esetén.

**GetFilm\_WithoutId\_ReturnsOkResult**

* Ellenőrzi, hogy a GetFilm metódus helyesen visszaad-e egy filmlistát, ha nincs ID megadva.
* Biztosítja, hogy az összes film lekérdezése helyesen működjön.

**GetFilm\_WithId\_ReturnsOkResult\_WhenFilmExists**

* Ellenőrzi, hogy a GetFilm metódus helyesen visszaad-e egy filmet, ha az ID létezik.
* Biztosítja, hogy az egyedi filmlekérdezés helyesen működjön.

**GetFilm\_WithId\_ReturnsBadRequest\_WhenFilmDoesNotExist**

* Ellenőrzi a hibakezelést, ha a film nem létezik.
* Biztosítja, hogy a rendszer ne omoljon össze, ha a film nem található.

## Filmkezelési műveletek

**AddFilm\_ReturnsOkResult\_WhenSuccessful**

* Ellenőrzi, hogy a AddFilm metódus helyesen visszaad-e egy sikeres üzenetet, ha a film hozzáadása sikerült.
* Biztosítja, hogy a film hozzáadása helyesen működjön.

**AddFilm\_ReturnsBadRequest\_WhenUnsuccessful**

* Ellenőrzi a hibakezelést, ha a film hozzáadása sikertelen.
* Biztosítja, hogy a rendszer helyesen kezelje a hibákat.

**EditFilm\_ReturnsOkResult\_WhenSuccessful**

* Ellenőrzi, hogy a EditFilm metódus helyesen visszaad-e egy sikeres üzenetet, ha a film módosítása sikerült.
* Biztosítja, hogy a film módosítása helyesen működjön.

**EditFilm\_ReturnsBadRequest\_WhenUnsuccessful**

* Ellenőrzi a hibakezelést, ha a film módosítása sikertelen.
* Biztosítja, hogy a rendszer helyesen kezelje a hibákat.

**DeleteFilm\_ReturnsOkResult\_WhenSuccessful**

* Ellenőrzi, hogy a DeleteFilm metódus helyesen visszaad-e egy sikeres üzenetet, ha a film törlése sikerült.
* Biztosítja, hogy a film törlése helyesen működjön.

**DeleteFilm\_ReturnsBadRequest\_WhenUnsuccessful**

* Ellenőrzi a hibakezelést, ha a film törlése sikertelen.
* Biztosítja, hogy a rendszer helyesen kezelje a hibákat.

## FoglalasController tesztek

A képen szöveg, képernyőkép, Betűtípus, szoftver látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

## Jegytípus kezelés tesztjei

**GetJegyTipus\_ReturnsListOfJegyTipus**

* Jegytípusok listájának lekérdezésének tesztelése

## Foglalás lekérdezési tesztek

**GetFoglalas\_AdminRole\_ReturnsListOfFoglalas**

* Admin jogosultságú foglaláslistázás tesztelése
* Ellenőrzi, hogy megfelelően kapjuk-e vissza a foglalásokat

**GetFoglalasByVetites\_ReturnsListOfFoglalas\_WhenExists**

* Vetítéshez tartozó foglalások lekérdezése
* Ellenőrzi, hogy nem üres lista érkezik-e vissza

**GetFoglalasByVetites\_ReturnsBadRequest\_WhenNotFound**

* Nem létező vetítéshez tartozó foglalások kezelése
* Ellenőrzi a BadRequestObjectResult visszakapását

**GetFoglalasByUser\_ReturnsListOfFoglalas\_WhenExists**

* Felhasználóhoz tartozó foglalások lekérdezése
* Ellenőrzi a felhasználó foglalási listát

**GetFoglalasByUser\_ReturnsBadRequest\_WhenNotFound**

* Nem létező felhasználói foglalások kezelése
* Ellenőrzi a BadRequestObjectResult-t

## CRUD műveletek tesztjei

**AddFoglalas\_ReturnsOkResult\_WhenSuccessful**

* Foglalás létrehozásának tesztelése
* Ellenőrzi az OkObjectResult-ot a sikerüzenettel

**AddFoglalas\_ReturnsBadRequest\_WhenUnsuccessful**

* **Cél**: Sikertelen foglalás készítés kezelése
* Ellenőrzi a BadRequestObjectResult-ot

**EditFoglalas\_ReturnsOkResult\_WhenSuccessful**

* Foglalás módosításának tesztelése
* Ellenőrzi a sikerüzenetet a válaszban

**EditFoglalas\_ReturnsBadRequest\_WhenUnsuccessful**

* Sikertelen módosítás kezelése
* Ellenőrzi a BadRequestObjectResult-ot

**DeleteFoglalas\_ReturnsOkResult\_WhenSuccessful**

* Foglalás törlésének tesztelése
* Ellenőrzi a sikerüzenetet

**DeleteFoglalas\_ReturnsBadRequest\_WhenUnsuccessful**

* Sikertelen törlés kezelése
* Ellenőrzi a BadRequestObjectResult-ot

## TeremController tesztek

A képen szöveg, képernyőkép, Betűtípus, sor látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

**DeleteTerem\_ReturnsOk\_WhenSuccessful**

* Sikeres terem törlésének tesztelése
* Ellenőrzi, a visszatérési érték típusát
* Ellenőrzi, hogy az eredmény OkObjectResult típusú
* Összehasonlítja, hogy a visszaadott üzenet megegyezik-e a mockolt sikerüzenettel

**DeleteTerem\_ReturnsBadRequest\_WhenUnsuccessful**

* Sikertelen terem törlésének tesztelése
* Ellenőrzi, hogy a visszatérési érték típusa ActionResult<ErrorModel>
* Ellenőrzi, hogy az eredmény BadRequestObjectResult típusú

## VetitesController tesztek

A képen szöveg, képernyőkép, Betűtípus látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

## Módosítási műveletek tesztjei

**EditVetites\_ReturnsOk\_WhenSuccessful**

* Sikeres vetítésmódosítás tesztelése
* Ellenőrzi a visszatérési érték típusát (ActionResult<ErrorModel>)
* Ellenőrzi, hogy az eredmény OkObjectResult típusú
* Ellenőrzi a visszaadott hibaüzenet tartalmát

**EditVetites\_ReturnsBadRequest\_WhenUnsuccessful**

* Sikertelen vetítésmódosítás tesztelése
* Ellenőrzi a visszatérési érték típusát
* Ellenőrzi, hogy az eredmény BadRequestObjectResult típusú
* Ellenőrzi a hibaüzenet tartalmát

## Törlési műveletek tesztjei

**DeleteVetites\_ReturnsOk\_WhenSuccessful**

* **Cél**: Sikeres vetítéstörlés tesztelése
* **Előkészítés**:
  + Mockolja a deleteVetites metódust, hogy "Sikeres törlés" üzenettel térjen vissza vetítés ID=1 esetén
* **Végrehajtás**:
  + Meghívja a controller DeleteVetites metódusát vetítés ID=1 paraméterrel
* **Ellenőrzés**:
  + Ellenőrzi a visszatérési érték típusát
  + Ellenőrzi, hogy az eredmény OkObjectResult típusú
  + Ellenőrzi a sikerüzenet tartalmát

**DeleteVetites\_ReturnsBadRequest\_WhenUnsuccessful**

* Sikertelen vetítéstörlés tesztelése
* Ellenőrzi a visszatérési érték típusát
* Ellenőrzi, hogy az eredmény BadRequestObjectResult típusú
* Ellenőrzi a hibaüzenet tartalmát

# Fejlesztési javaslatok

* A backend és a lekérdezések optimalizálása és biztonságosabbá tétele
* Web szerverre való feltöltés és futtatás (folyamatban, ha sikeres, akkor a <https://premozi.site> oldalon lesz elérhető a projekt)
* Fizetési lehetőség
* Frontend optimalizálása, a design javítása
* Adatbázis optimalizálása
* Docker konfigurálása és az onnan való futtatás

# Összegzés/Reflexió

A mozijegy-foglaló webalkalmazás fejlesztése során számos kihívással és tanulsággal gazdagodtunk. A projekt nem csupán technikai ismeretek bővítését jelentette számunkra, hanem lehetőséget adott a csapatmunka és a problémamegoldó készségeink fejlesztésére is. Az alkalmazás elkészítése során sikerült egy olyan rendszert létrehoznunk, amely lehetővé teszi a felhasználók számára a mozijegyek gyors és kényelmes foglalását, miközben biztonságos és hatékony adatkezelést biztosít.

A projekt során kiemelt figyelmet fordítottunk az adatbázis tervezésére és a REST API implementálására. Az Entity Framework és a MySQL kombinációja kiválóan bizonyított az adatok kezelésében, míg a JWT token-alapú hitelesítés lehetővé tette a biztonságos felhasználókezelést. A frontend React és Vite segítségével készült, ami reszponzív és felhasználóbarát felületet eredményezett. A tesztelés során az NUnit keretrendszert használtuk, ami segített a kód megbízhatóságának és stabilitásának biztosításában.

A csapatmunka kulcsfontosságú volt a projekt sikerében. Minden csapattag a saját erősségei alapján vállalt feladatokat, és a rendszeres megbeszélések lehetővé tették a problémák gyors megoldását. A kommunikációhoz a Discord-ot, a verziókövetéshez pedig a GitHub-ot használtuk, ami nagyban hozzájárult a munka hatékonyságához. A ProtonVPN segítségével pedig akadálytalanul tudtunk elérni külső forrásokat, mint például a StackOverflow-t vagy a W3Schools-t.

A projekt során felmerülő nehézségek, például az adatbázis normalizálása vagy a kétlépcsős hitelesítés implementálása, lehetőséget adtak a kreatív problémamegoldásra és a tanulásra. Ezek a tapasztalatok értékesebbé tették számunkra a fejlesztési folyamatot.

Összességében elmondható, hogy a projekt sikeresen teljesítette a kitűzött célokat, és egy jól működő, felhasználóbarát alkalmazást eredményezett. A jövőben további fejlesztésekkel, például egy adminisztrációs felülettel vagy értesítési rendszerrel, még jobbá tehető a rendszer. A projekt nem csupán technikai ismereteket adott, hanem megerősítette bennünket abban, hogy a csapatmunka és a kitartás kulcsfontosságú a sikerhez.

# Források, segítségek

* ChatGPT. Deepseek, Cody
* W3Schools
* Frostbyte
* Stackoverflow
* React dokumentáció (19.1)
* Youtube videók
* Egyéb, a programkódban, kommentben jelzett források