Név: Barnák Péter
Email: barnak.peter1@gmail.com
5. csoport

Dokumentáció
(Mozi)

Feladat:

Készítsünk egy mozi üzemelteto rendszert, amely alkalmas az eloadások, illetve jegyvásárlások kezelésére.

1. részfeladat: a webes felületen keresztül a nézok tekinthetik meg a mozimusort, valamint rendelhetnek
jegyeket.

- A fooldalon megjelenik a napi program, azaz mely filmeket mikor vetítik a moziban, valamint kiemelve az öt legfrissebb (legutoljára felvitt) film plakátja.
- A filmet kiválasztva megjelenik annak részletes leírása (rendezo, foszereplok, hossz, szinopszis), plakátja, továbbá az összes eloadás idopontja.
- Az idopontot kiválasztva lehetoség nyílik helyfoglalásra az adott eloadásra. Ekkor a felhasználónak meg kell adnia a lefoglalandó ülések helyzetét (sor, illetve oszlop) egy, a mozitermet sematikusan ábrázoló grafikus felületen. Egyszerre legfeljebb 6 jegy foglalható, és természetesen csak a szabad helyek foglalhatóak (amelyek nem foglaltak, vagy eladottak). A felhasználónak ezen felül meg kell adnia teljes nevét, valamint telefonszámát, ezzel véglegesíti a foglalást.

2. részfeladat: az asztali grafikus felületet az alkalmazottak használják a mozipénztárakban az eloadások meghirdetésére, illetve jegyek kiadására.

- Az alkalmazott bejelentkezhet (felhasználónév és jelszó megadásával) a programba, illetve kijelentkezhet.
- Új film felvitelekor ki kell tölteni a film adatait (cím, rendezo, foszereplok, hossz, szinopszis), valamintfeltölthetünk egy képet plakátként.
- Új eloadás meghirdetéséhez a felhasználónak ki kell választania a termet, valamint a filmet, és az idopont megadásával hirdetheti meg az eloadást. A meghirdetéskor ügyelni kell arra, hogy az eloadás ne ütközzön más eloadásokkal az adott teremben (figyelembe véve a kezdés idopontját, illetve a filmhosszát), illetve két eloadás között legalább 15 percnek kell eltelnie a takarítás végett.
- A jegyvásárláshoz ki kell választani a filmet és az eloadást. Ezt követoen listázódnak a helyek (sor, oszlop, státusz). A szabad, illetve foglalt helyek eladhatóak, illetve a foglalt helyeket kiválasztva meglehet tekinteni a foglaló adatait (név, telefonszám).

Elemzés:

A két feladatot két nagyobb projectben valósítjuk meg, melyekhez kisebb segéd projecteket készítünk a kommunikációhoz.

Az első ilyen project a webes applikáció amit ASP.NET keretrendszerben MVC arhitektúrában valósítjuk meg C# nyelven. Ezen kívül a webes projecthez javacsript és css nyelvű programkódok is felhasználásra kerültek, a jobb élmény érdekében.

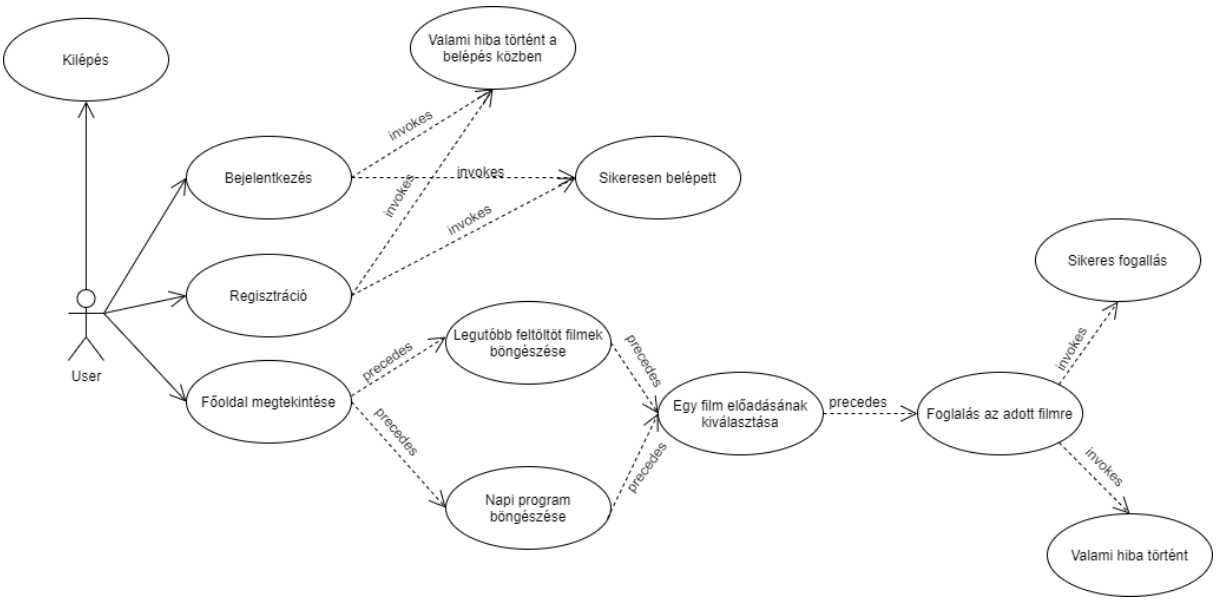
A második project egy asztali alkalmazás, amely szintén C# nyelven készült, MVVM arhitektúrában.

A két alkalmazáshoz készült egy SqlServer adatbázis is, amelyben a feladat végrehajtásához szükséges adatokat tároljuk. Az adatbázist code first módon definiáltuk a Persistence segédprojectben. Ezen projectben tárultuk még el az adatbázis entitásokat konvertáló DTO classainkat is, amelyek az egyes lekérdezések JSON serializációjában segítik a munkát.

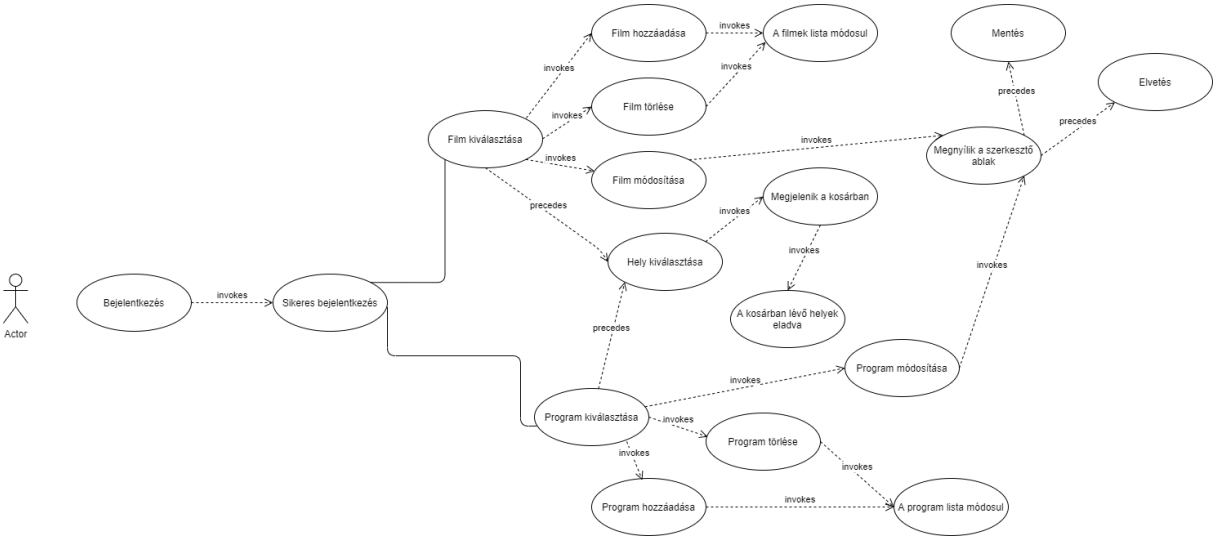
Név: Barnák Péter
Email: barnak.peter1@gmail.com
5. csoport

Felhasználói esetek

Webes alkalmazás:



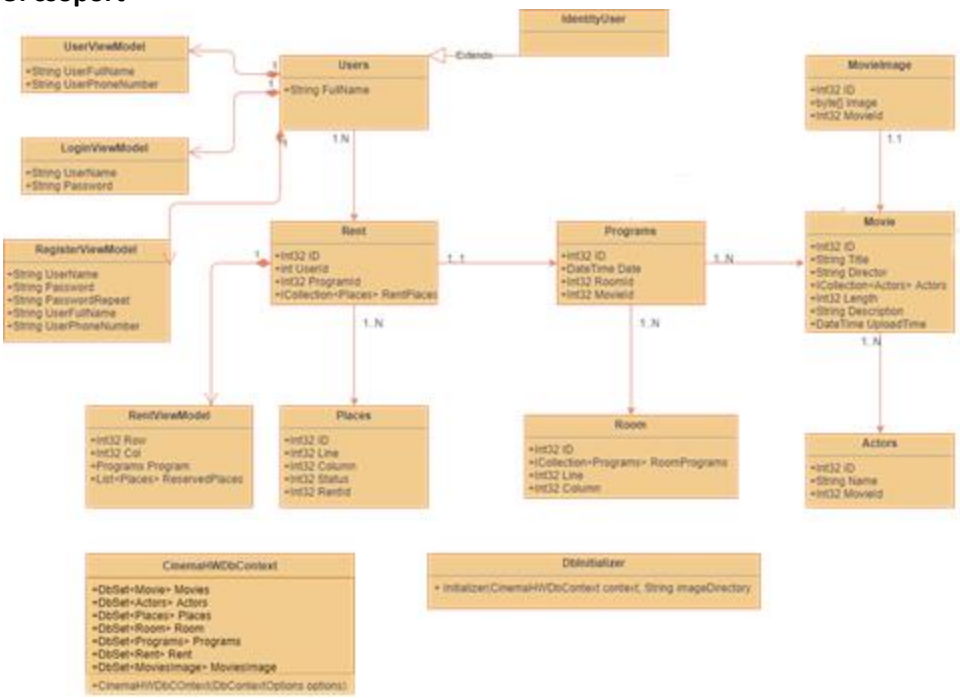
Asztali alkalmazás:



Tervezés

A teljes project 4 részre osztható. Ebből kettő a már említett asztali és webes alkalmazás. Ezen kívül szerepet kap még WebAPI és a már említett Persistence is. A WebAPI feladata, hogy kommunikációt engedjen meg az asztali alkalmazásunk és az adatbázis között. Ez a kommunikáció HTTP kérések formájában történik, ahol is az asztali alkalmazás elküld a megadott webcímre egy „speciális URL”-t amelyet a WebAPI megfelelő controllere fogad. Ezután a controller metódusok segítségével lekérdezéseket hajt végre az adatbázison, majd a lekérdezés eredményét visszaküldi az asztali alkalmazásnak. Az asztali alkalmazás felé a WebAPI Data Transfer Objectekkel kommunikál, ezek lehetővé teszik, hogy az adatbázisból kiolvasott adatokat egy JSON fájlá konvertálva továbbíthassuk.
Ez felül még a projectünk rendelkezik egy 5. egységgel is, ez a WebAPI.Tests. Ennek feladata a WebAPI kontrollereinek metódusait egy teszt adatbázison végrehajtani, ezzel biztosítva azt, hogy a lekérdezéseink helyesek.

Név: Barnák Péter
Email: barnak.peter1@gmail.com
5. csoport



Persistence

Ebben a rétegben definiáljuk az adatbázist leíró entitásmodelleket, illetve létesítünk kapcsolatot az adatbázissal. Az adatbázis inicializációjára Migration-öket használunk. Az adatbázis séma a következő:

- **Actors**(Id, Name, MovieId): Egy film színészeit tartalmazó tábla.
- **Movie**(Id, Title, Director, Actors(List<>), Length, Description,UploadTime): Egy film tulajdonságit leíró tábla, amely tartalmazza az adott filmhez tartozó színészeket is, amelyeket az EntityFrameworkCore rendel hozzá a Movie táblához MovieId alapján.
- **Program**(Id, Date, RoomId, MovieId): Az egyes műsorokat leíró tábla, tartalmazza, hogy az adott film (MovieId) mely teremben (RoomId) és mikor fog menni (Date).
- **Room**(Id, Line, Column, RoomPrograms(List<>)): Egy termet reprezentáló tábla, amely tartalmazza egy terem méreteit illetve a hozzá kapcsolódó programokat is.
- **Rent**(Id, UserId, ProgramId, RentPlaces(List<>)): Egy foglalást reprezentáló osztály, amely tartalmazza a foglaló felhasználó azonosítóját (UserId), illetve a foglalás részleteit(melyik programra foglalt, illetve, mely helyeket foglalta le)
- **Place**(Id, Line, Column, Status, RentId): Egy foglalt helyet reprezentál, amely helyzet tartozik egy foglalás azonosító is, hogy nyomon lehessen követni, mégis melyik programra szólt a foglalás. Tartalmaz egy Status mezőt is amelynek szerepe majd csak a 2. beadandóban lesz igazán.
- **Users**(FullName): Az egyes felhasználókat tartalmazó tábla, amely önmagában csak a felhasználó teljes nevét tartalmazza, viszont ez a tábla megvalósítja az IdentityUser osztályát az ASP.NET keretrendszerből, amely egy felhasználókat kezelő adattábla.
- **MoviesImage**(Id, MovieId, Image): Az egyes filmekhez tartozó plakát byte kódját tároljuk el itt

Ezen adatbázisokon kívül itt találhatóak még a JSON serializációhoz szükséges Data Transfer Objectek is. Ezen DTO-k nagyban eltérhetnek az adatbázis felépítésétől, viszont ezen feladat esetében ez nem így történt, a DTO-ink egy az egyben az adatbázis táblákat írják le, így ennek

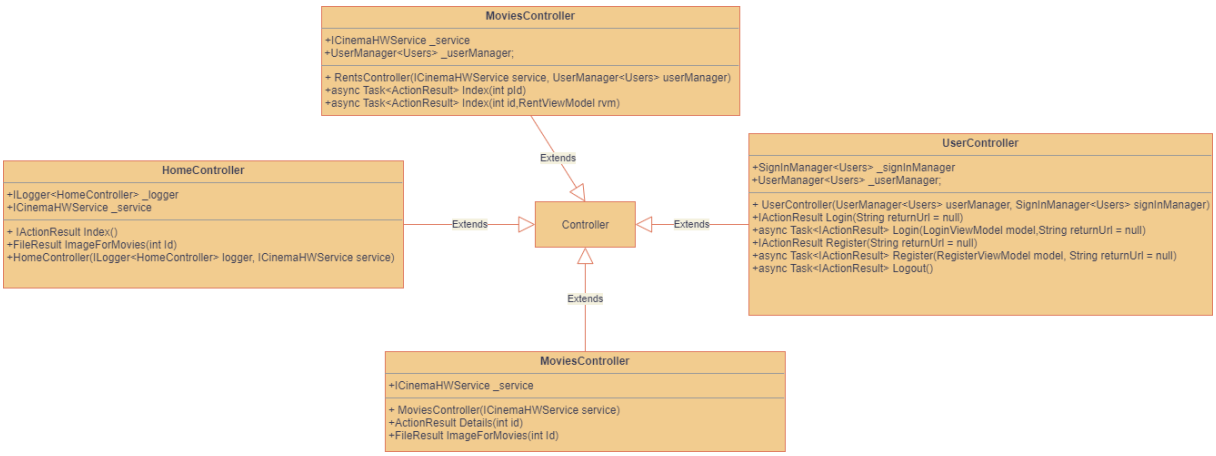
Név: Barnák Péter
Email: barnak.peter1@gmail.com
5. csoport
a felépítését már bemutattok. Minden DTO-ban definiálva van egy explicit művelet az adatbázis objektumok átalakításhoz és fordítva.

Web (webes alkalmazás)

Az alkalmazást MVC architektúrában valósítom meg. A megvalósításhoz szükségem lesz egy View, egy Model, egy Controller és egy Service rétegre. Az egyes rétegek külön-külön feladatokat látnak el, viszont egymással kommunikálnak. A View réteg a megjelenítésért felelős. A Controller réteg hajtja végre az egyes felhasználói kéréseket, tehát ő az a réteg amely kapcsolatot tart a felhasználó és a modell között. A Controller rétegben kerülnek felhasználásra a Persistence Service rétegben előre definiált adatbázis lekérdezések. Az adatbázis lekérdezéseket LINQ lekérdezések formájába fogjuk implementálni. A web az indítás után a Startup.cs állománya alapján inicializálni fogja a Persistenceben szereplő Migration-ök alapján az adatbázist.

Model

A webes alkalmazás Model rétegének hatalmas része a Persistencben található. Ezáltal itt csak két osztályunk maradt amelyek a weben történő adatmegjelenítésben vesznek részt.



Controller:

A Persistence Service rétegben definiált adatbázis lekérdezések segítségével a Controller réteg megjeleníti a kívánt adatokat a View-n.

- HomeController:
  - Index: Megjelenik az 5 legújabban feltöltött film és plakátja. Illetve az aznapi program.
  - ImageForMovies: A képek megjelenítésért felelős.
- MoviesController:
  - Details: Az adott film adatait jeleníti meg.
  - ImageForMovies: A képek megjelenítésért felelős.
- RentsController:
  - Index:
    - Get: A kiválasztott programhoz tartozó felületet jeleníti meg, azaz betölt egy NxM-es mozitermet ábrázoló „gombrácsot”

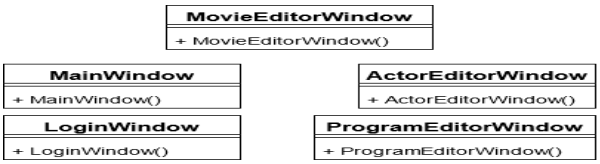
Név: Barnák Péter
Email: barnak.peter1@gmail.com
5. csoport

        - Post: A kiválasztott székeket lefoglalja a megadott felhasználó adataival.
- UserController:
    - Logout: Kijelentkezteti a felhasználót
    - Login:
        - Get:Megjeleníti a Bejelentkezéshez szükséges mezőket.
        - Post: Bejelentkezteti a felhasználót, ha létezik felhasználó a megadott adatokkal
    - Register
        - Get: Megjeleníti a Regisztrációhoz szükséges mezőket.
        - Post: Regisztrálja a felhasználót, ha a felhasználó jó adatokat adott meg.

View:

A Controllerek által küldött információk megjelenítését szolgálja, ezáltal minden controller rendelkezik legalább egy hozzárendelt nézettel. A nézet az ASP.NET által használt cshtml állományokból áll, amelyek lehetővé teszik „dinamikus” html oldalak elkészítését, azaz egyszerre tudunk egy frontend websiteot készíteni, és egyszerre tudunk az adott oldalon c# kódot írni. Ezáltal tudjuk megjeleníteni a Controllerek által modelként továbbított értékeket a View-n.

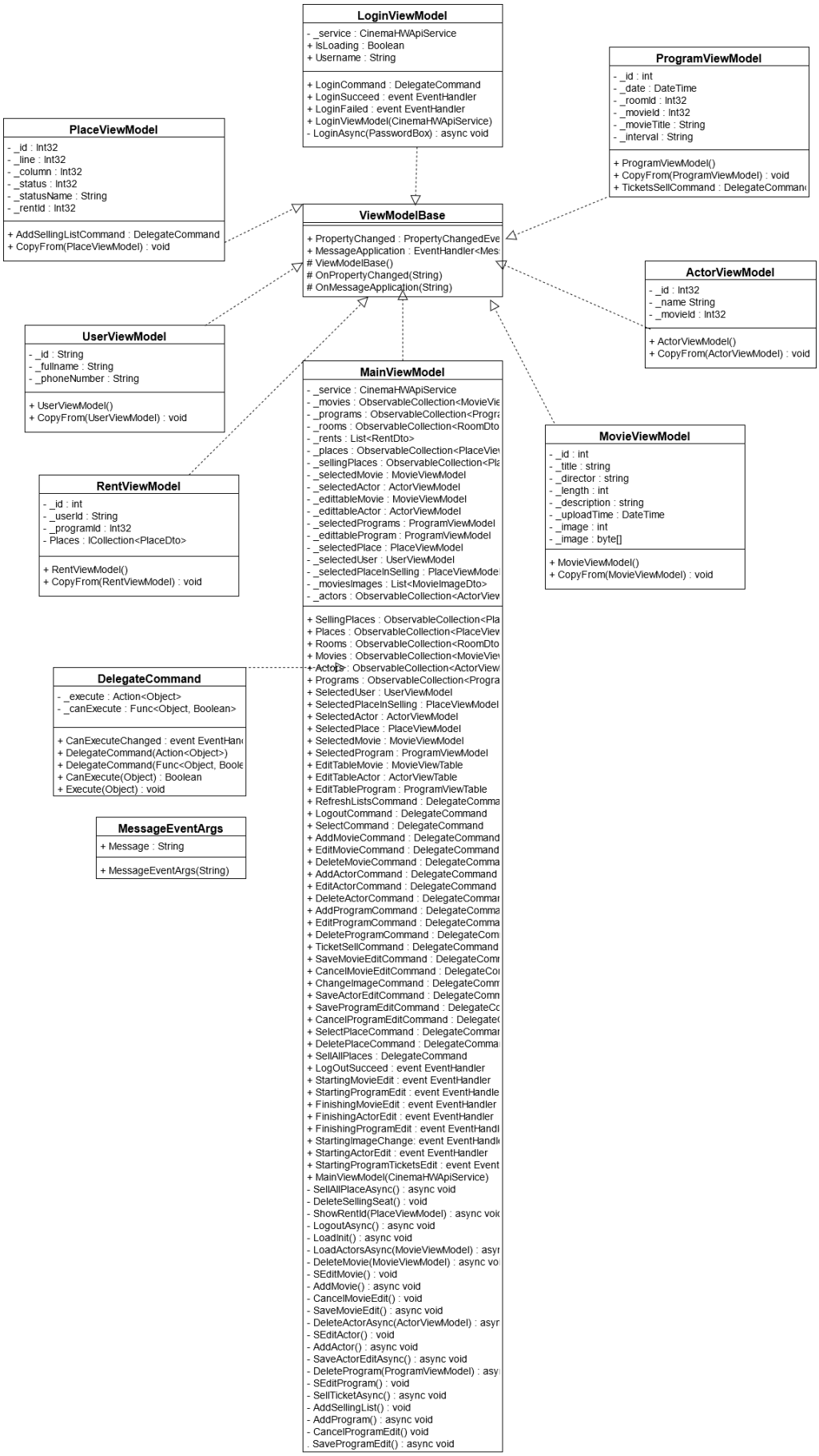
Asztali alkalmazás:



View:

Az asztali alkalmazás nézete 5 különböző formból tevődik össze. Az alkalmazás megnyitása után megjelenő LoginWindowból, a sikeres bejelentkezés utána feltűnő és lényegében a fő programhoz tartozó MainWindowból, illetve a MainWindowból elérhető segédablakokból (ActorEditorWindow, MovieEditorWindow, ProgramEditorWindow). Ezen segédablakok segítségével tudjuk beállítani az egyes filmek adatait, az egyes szereplők nevét, illetve az új programjainkat tudjuk személyre szabni.

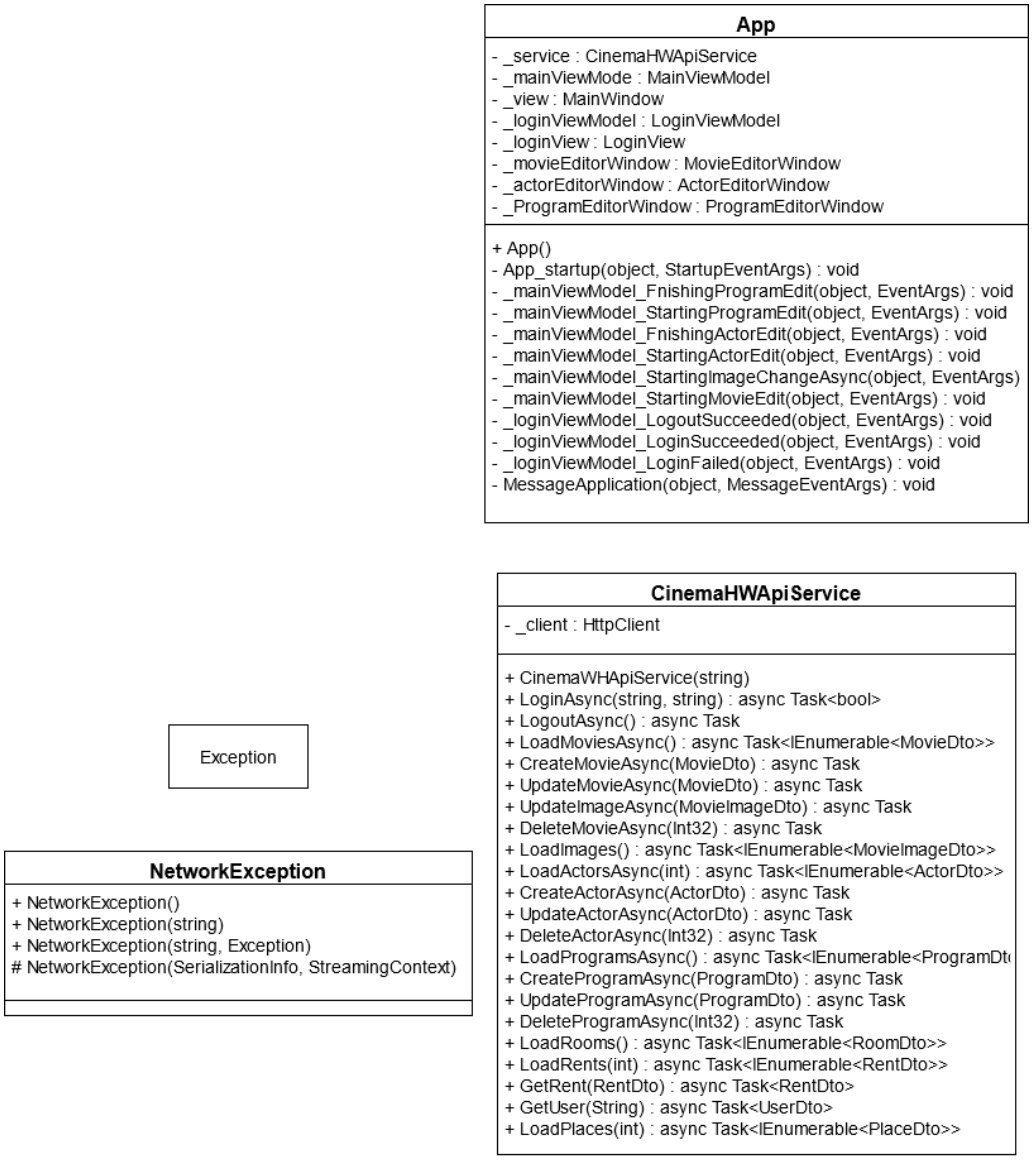
Név: Barnák Péter
Email: barnak.peter1@gmail.com
5. csoport



Név: Barnák Péter
Email: barnak.peter1@gmail.com
5. csoport

ViewModel:

Az asztali alkalmazás esetében a viewmodellek feladata az egyes ablakokon megjelenő adatok beállítása. Ezen projecthez 8 viewmodell tartozik, melyek egytől egyig szerepet játszanak a program működésében. A fő ezek közül a MainViewModel, amely nem csak a MainWindowról tartalmaz információt, hanem az összes többi megjelenő ablaknál is szerepet játszik. Ez kommunikál a Model réteggel és kéri el tőle, a WebAPI által az adatbázisból megszerzett adatokat is.



Model:

A fő HTTP kéréseket tartalmazza, amelyekkel a WebAPI-nk adatot tud nekünk szolgáltatni, fel tud tölteni egy új rekordot az adatbázisba, módosítani tud már meglévő rekordokat illetve törölni is tud azok közül. Ezen kívül még rendelkezik a Model egy általunk definiált Exceptionnel is, amellyel hálózati hibát tudunk jelezni a felhasználó felé.

App:

Név: Barnák Péter
Email: barnak.peter1@gmail.com
5. csoport
Ez az a része az alkalmazásunknak amely a MainViewModeltől kapott eventek segítségével inicializálja az egyes nézeteket, illetve esetleges hibaüzeneteket is közöl a felhasználóval.

Alkalmazás felületének leírása:

Főablak:

Minden lényeges vezérlő ebben az ablakban található. A menüsorban található egy kijelentkező és egy frissítő gomb, amivel a filmek és programok listája frissíthető. Ez alatt található a filmeket felsoroló lista, amelyek megjelenítik a létező filmek címét. Ezek a filmet szerkeszthetőek, fel lehet venni újat, és törölni is lehet közülük. A következő listában listázódnak ki a programok, mégpedig megjelenítve a program idejét a filmet és, hogy melyik teremben lesz az előadás. A módosítási tulajdonságaik megegyeznek a filmekével. A programok végén található Sell Tickets gomb megnyomására a bal alsó listában megjelennek az adott programhoz tartozó terem aktuális helyei. Az szabad vagy foglalt helyek kiválasztásakor azok átkerülnek a jobbra lévő listboxba, ezzel jelezve a felhasználónak, hogy ők „kosárba” vannak rakva. Az alsó Sell Places gombbal lehet eladni a helyeket. Ha egy foglalt helyet választunk ki, akkor róla kiíródik, hogy mely felhasználó foglalta le az adott helyet.

Film szerkesztése:

A film szerkesztésének megnyitásával lehetőségünk nyílik szerkeszteni a fim adatait. Beállíthatjuk a címét a hosszát, a hozzá tartozó képet stb. Illetve alul található egy lista az adott filmhez tartozó színészek neveivel. Ez a lista is szerkeszthető ugyan úgy mint a filmek listája.

Színészek szerkesztése:

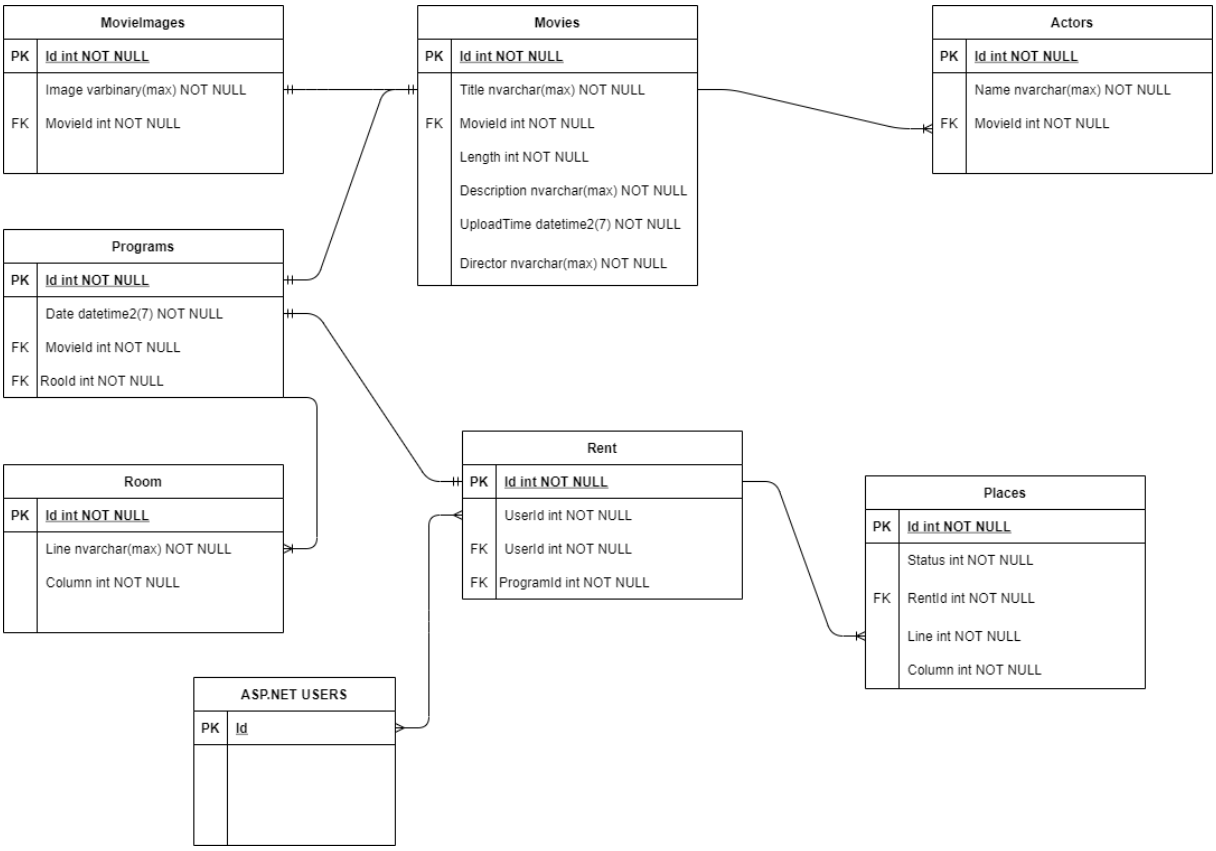
Ha egy színész nevét szeretnénk szerkeszteni, egy kis pici ablak ugrik fel a képernyőre egyetlen inputmezővel, ahol megadhatjuk az adott színész pontos nevét.

Programok szerkesztése:

A programok szerkesztése során megnyíló ablakban két legördülő lista és egy datetime pickerrel találkozhatunk. Ezek segítségével tudjuk megadni, hogy mely filmet melyik terembe és mikor szeretnénk leadni

Név: Barnák Péter
Email: barnak.peter1@gmail.com
5. csoport

***Adatbázis*:**



Tesztelés:

- GetActorByIdTest
- GetActrosTest
- GetInvalidActorTest
- GetInvalidMoveiTest
- GetInvalidPlaceTest
- GetInvalidProgramTest
- GetInvalidRentTest
- GetMoveisByIdTest
- GetMoviesTest
- GetProgramByIdTest
- GetProgramsTest
- GetRentByIdTest
- GetRentsTest
- PostActorTest
- PostMovieTest
- PostProgramTest