Dokumentáció – Könyvtár Kölcsönző

Készítette: Márton Zoltán _ B44T65

Feladat:

Készítsük el egy könyvtár online kölcsönzői és nyilvántartó rendszerét, amellyel a látogatók kölcsönzését,

előjegyzését, valamint a könyvtárosok adminisztratív munkáját tudjuk támogatni.

2) Részfeladat

A könyvtárosok az asztali grafikus felületen keresztül adminisztrálhatják a könyveket és a kölcsönzéseket.

A könyvtáros bejelentkezhet (felhasználónév és jelszó megadásával) a programba, illetve kijelentkezhet; a további funkcionalitások csak bejelentkezett állapotban elérhetőek.

Az alkalmazás listázza a könyveket, valamint a hozzájuk tartozó köteteket. Lehetőség van új könyv,illetve kötet rögzítésére. A könyvtáros selejtezhet egy kötetet, de csak akkor, ha nincsen aktuálisan kikölcsönözve.

Az esedékes jövőbeni előjegyzések törlésre kerülnek és az adott kötet továbbá nem lesz kölcsönözhető.Az alkalmazás listázza az aktív kölcsönzéseket és a jövőben esedékes előjegyzéseket.

A könyvtáros egy aktív kölcsönzést inaktívvá tehet (visszavitték a könyvet), valamint egy inaktív előjegyzést aktív kölcsönzésnek jelölhet (elvitték a könyvet). Egy kölcsönzés státuszának változtatása nincs a tervezett kezdő és befejező naphoz kötve (gondolva pl. a késedelmes visszavitelre), azonban egy kötetnek egyszerre legfeljebb egy aktív kölcsönzése lehet.

- könyvek (cím, szerző, kiadás éve, ISBN szám, borítókép);
- kötetek (könyv, könyvtári azonosító);
- kölcsönzések (kötet, látogató, kezdő nap, befejező nap, aktív-e)
- látogatók (név, cím, telefonszám, azonosító, jelszó)
- könyvtárosok (név, azonosító, jelszó).

NuGet Package-ek:

Microsoft.AspNetCore.Identity.EntityFrameworkCore v 3.1.14
Microsoft.EntityFrameworkCore v 3.1.14
Microsoft.EntityFrameworkCore.Proxies v 3.1.14
Microsoft.EntityFrameworkCore.Sqlite v 3.1.14
Microsoft.EntityFrameworkCore.SqlServer v 3.1.14
Microsoft.EntityFrameworkCore.Tools v 3.1.14
Microsoft.Extensions.Configuration.Json v 3.1.14
Microsoft.VisualStudio.Web.CodeGeneration.Design v 3.1.5

Elemzés

A Feladatot MVVM architektrában valósítjuk meg, és egy BookstoreAPI nevű API-al biztosítjuk a kapcsolatot a Bookstore.Persistence(Class library) projektben található adatbázissal.

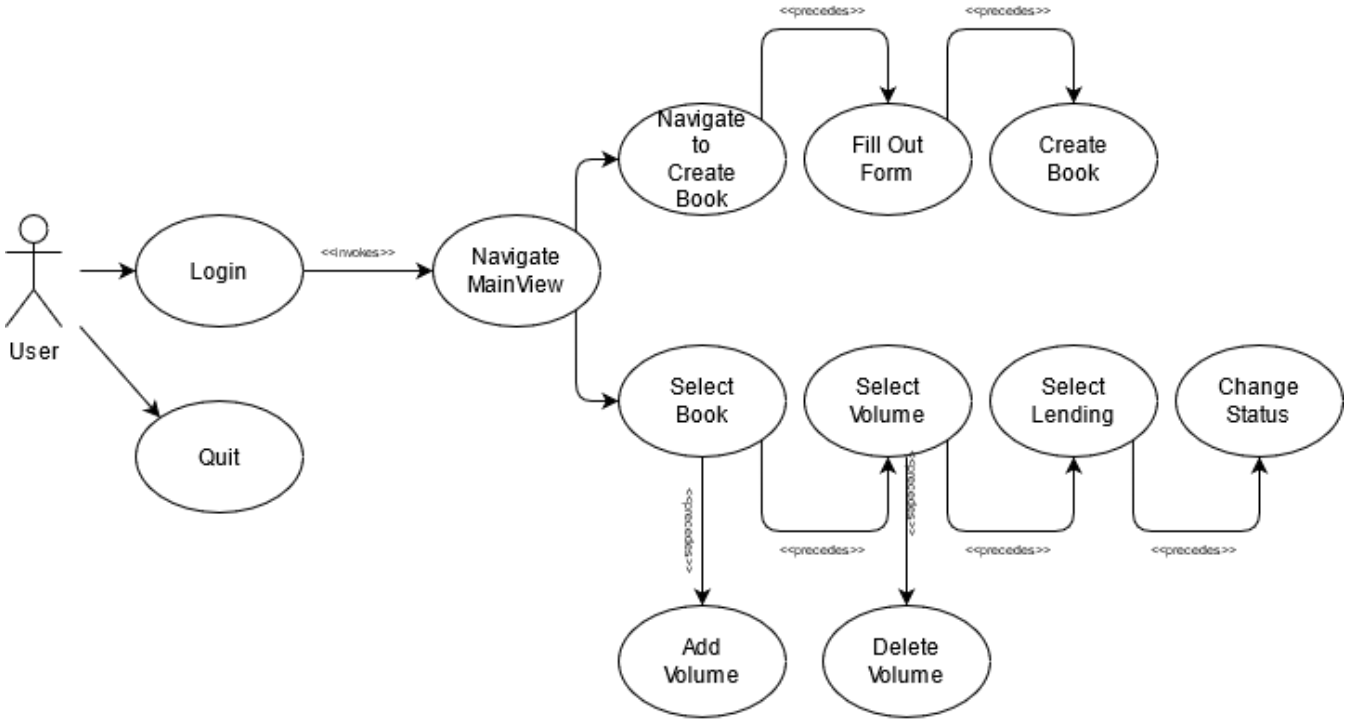
Az adatbázissal az API-on keresztül elérhető Controllerekkel fogunk kommunikációt létrehozni. A Http- Post, -Get, - Put és -Delete kérésekkel. Ennek lebonyolítására, szolgál a Model/BookstoreAPIService amit példányosítunk minden ViewModel-ben.

Miden Viewmodel a ViewModelBase-ből származik le így könnyen megvalósíthatjuk az egyes tulajdonságok változásait a kinézetben.

Ebben a projektben táblaelemek kapcsolatáshoz szükséges DataTransferObject-eket a Persistencián belül definiáljuk.

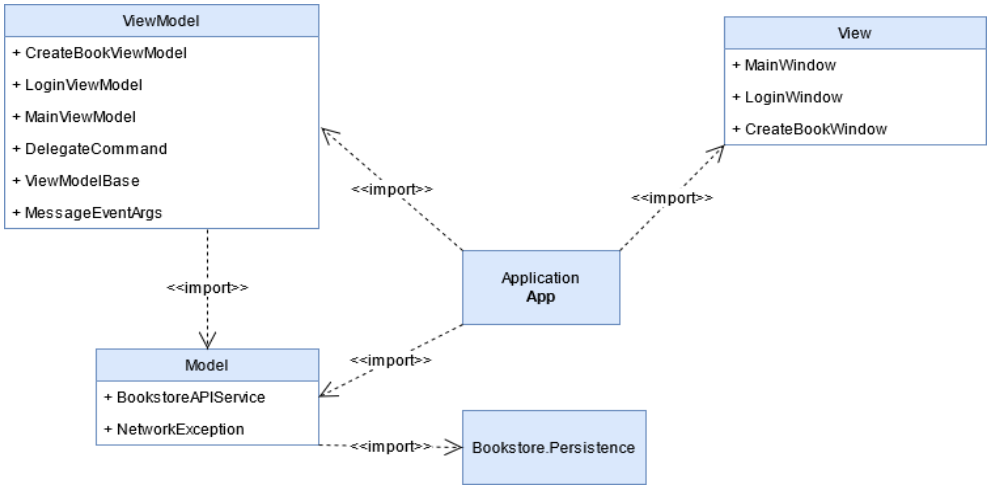
Az asztali kliens 3 ablakból, áll. Először egy Login ablakot jelenítünk meg, mivel a többi funkciót csak egy "Librarian" IdentityRole-al rendelkező user érheti el. Belépés után elérhető a főablak, ahol az egyszerűbb törléseket/módosításokat végezzük. Egy lenyíló ablakkal elérhető a 3. Nézet ami egy új könyv hozzáadásához szükséges form-ot jelenít meg a felhasználónak.

Felhasználói Esetek



Rendszer szerkezete

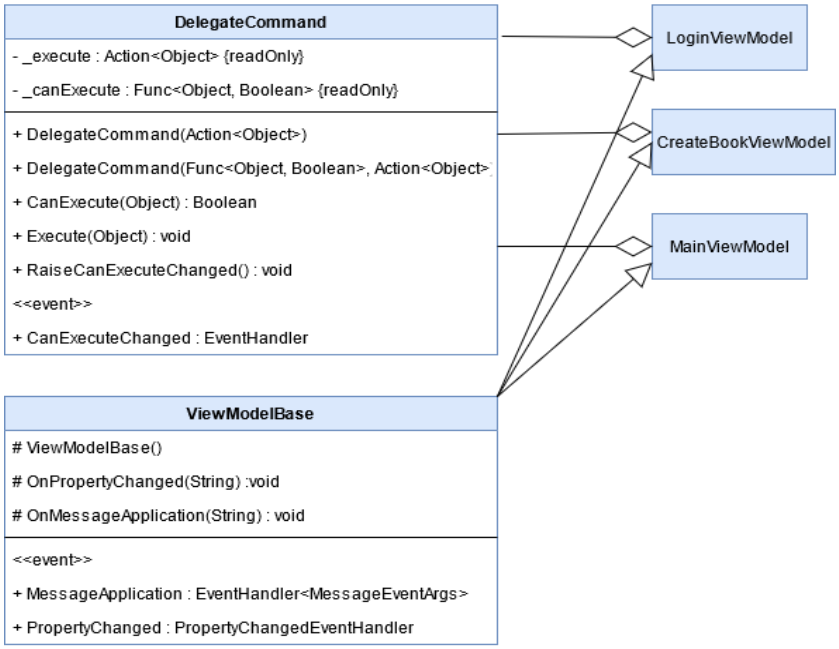
MVVM architektúrát valóstunk meg. Az "App" példányosítja a modellt, nézetmodelleket, nézeteket, kezeli ezek eseményeit, és a közöttük történő kommunikációt.



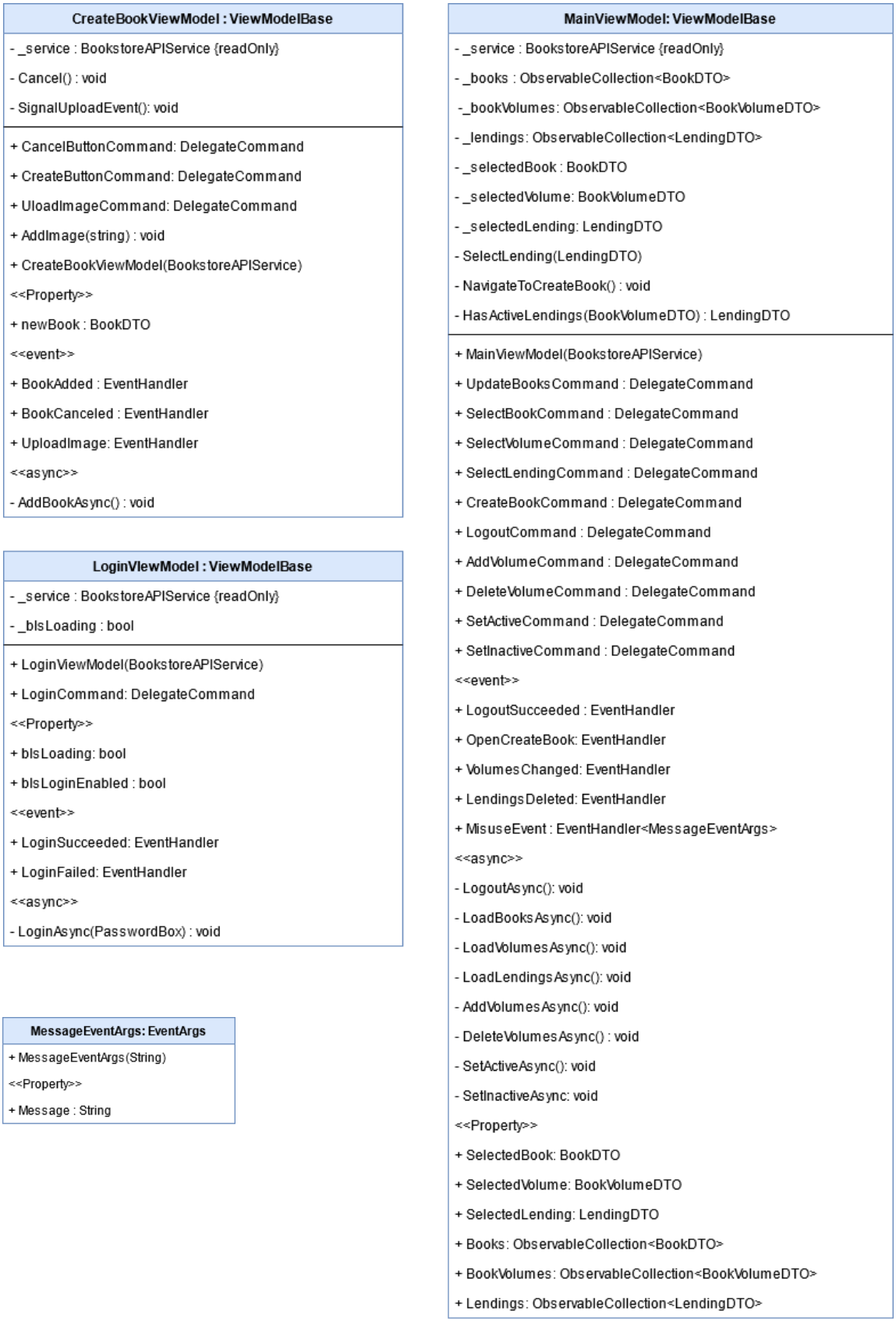
Perszisztencia:

A perszisztencia a program adatkezelésével foglalkozik, itt egy "*code-first*" megvalósítása egy adatbázisnak EntityFramework használatával. Az adatbázissal a BookstoreAPIService osztályban példányosított HttpClient osztályon keresztül Http kérésekkel komminkálunk.

ViewModel:

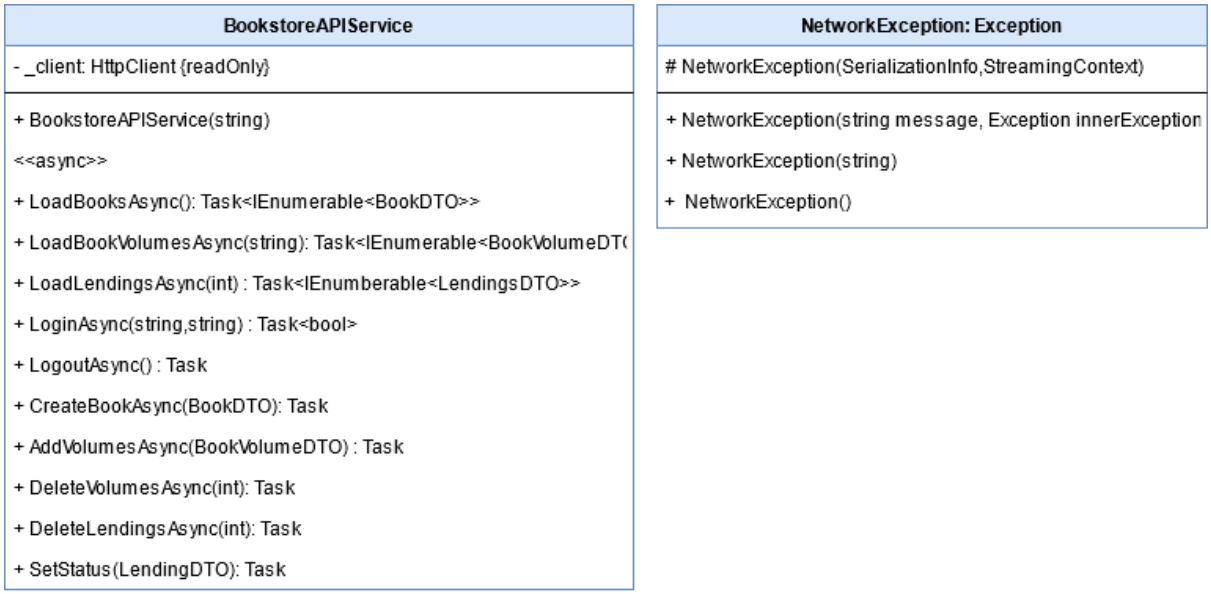


A ViewModel felel egy-egy nézet és a Modell közötti logika és kommunikációért. Ezek megvalósításához használunk egy általános utasítás osztályt valamint egy ősosztályt a nézetek tulajdonságainak megváltoztatásának felismerésére.



Model:

A Model-ben a már korábban említett adatbázis kommunikáció valósul meg Http kérésekkel, továbbá használunk egy saját Exception-t.

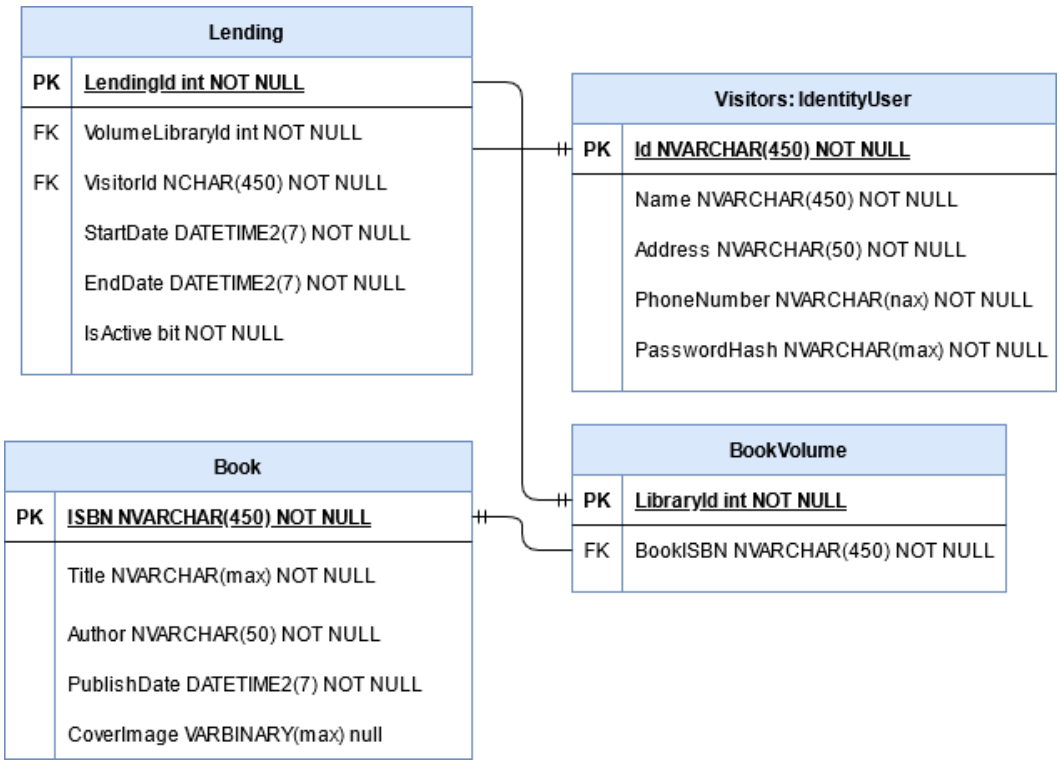


Környezet:

A környezet példányosítja és köti össze a többi réteget, valamint a nézetmodellek eseményeit kezeli le.

Adatbázis felépítése

A felépítés változatlan. Logikailag a Visitors egy BaseUser osztályból származik, aminek kevesebb Property-je van mint egy visitor-nek és ez felel meg a “Könyvtáros”-nak



Tesztelés:

A tesztelést a Bookstore.Test nevű project hajtja végre XUnit felhasználásával. A tesztek során az adatbázis elérést teszteljük, valamint, hogy megfelelően működnek-e a funkciók.

GetBooksTest – A BooksController listalekérését teszteli

GetBookTest – A BooksController egyéni könyvlekérését teszteli

InvalidGetBookTest - Hibatesztelés

TestPostBook – A BooksController új könyv “Post”-olását teszteli

TestGetBookVolumes – A BookVolumesController lekérését teszteli

TestPostBookVolume – A BookVolumesController új kötet “Post”-olását teszteli

TestDeleteBookVolume – A BookVolumesController kötet törtlés funkcióját teszteli

DeleteGetLendings - A LendingsController Lekérés, valamint törlési funkcióját teszteli

BookstoreAPI

BooksConroller:

- *GetBooks:* Paraméter nélkül visszatér egy könyv Data Transfer Object-ekből álló IEnumerable típussal

- *GetBook:* Egy string ID paraméterrel visszaadja a kapott kulcshoz tartozó könyv Data Transfer Objektumát, vagy egy 404 NotFound státuszkódot.
- *PostBook:* Egy könyv Data Trasnfer Object-et kap paraméterül és ezt megkísérli létrehozni az adatbázisban, amennyiben nem létezik. Visszatér egy 500-as internal server error státuszkóddal, vagy egy 201-es Created státuszkóddal attól függően, hogy a művelet sikeres volt-e.

BookVolumesController:

- *GetBookVolumes:* Egy könyv kulcsát paraméterül kapva visszatér egy kötet Data Transfer Object-ekből álló IEnumerable típussal
- *PostBookVolume:* Megkísérel létrehozni, egy kötetet a paraméterül kapott kulcshoz tartozó könyvhöz. Sikeres létrehozás esetén, visszatér egy "Ok" 200-as státuszkóddal, amennyiben nem sikerült, egy 500-as internal error-al tér vissza.
- *DeleteBookVolume:* Megkísérel törölni egy paraméterül kapott kulcshoz tartozó kötetet. Amennyiben sikerül egy 200-as Ok státuszkóddal tér vissza, amennyiben sikertelen volt a törlés egy 500-as internal server error státuszkóddal tér vissza.

LendingsController:

- *GetLendings:* Egy kulcshoz tartozó kötet jegyzéseinek DTO-jából álló IEnumberable típussal tér vissza, ha nincsenek ilyenek akkor egy NotFound 404-es státuszkóddal tér vissza.
- *DeleteLendings:* Megkísérli egy előjegyzés törlését. Ha sikeresen töröl akkor egy 200-as Ok státuszkóddal tér vissza, ellenben ha sikertelen a művelet akkor egy 500-as Internal Server Error státuszkóddal tér vissza.
- *PutLending:* Megkísérli megváltozatatni egy már létező kötet jegyzés property-jeit, arra olyanra mint amilyet paraméterül kap. Sikeres művelet esetén a visszatérési érték 200-as OK státuszkód, sikeretlen művelet esetén viszont egy 500-as Internal Server Error.

LibrarianController:

- *Login:* Paraméterül egy LoginDTO-t kap. Az ebben lévő Username-hez tartozó felhasználót, ha "Librarian" Role-ba tartozik belépteti, ellenben 401-es Unauthorized státuszkóddal tér vissza.
- *Logout:* Kiléptei a jelenlegi felhasználót, majd egy OK státuszkóddal visszatér.