Webes Alkalmazások Fejlesztése

Dokumentáció a 2. beadandó feladathoz

Huszár Ádám

X4YWL8

1. Kitűzött feladat:

Az orvosok az előjegyzéseket az asztali grafikus felületen keresztül adminisztrálják.

• Az orvos bejelentkezhet (azonosító és jelszó megadásával) a programba. Sikeres bejelentkezést követően

látja a rá vonatkozó foglalásokat (az aktuális dátumtól), illetve kijelentkezhet.

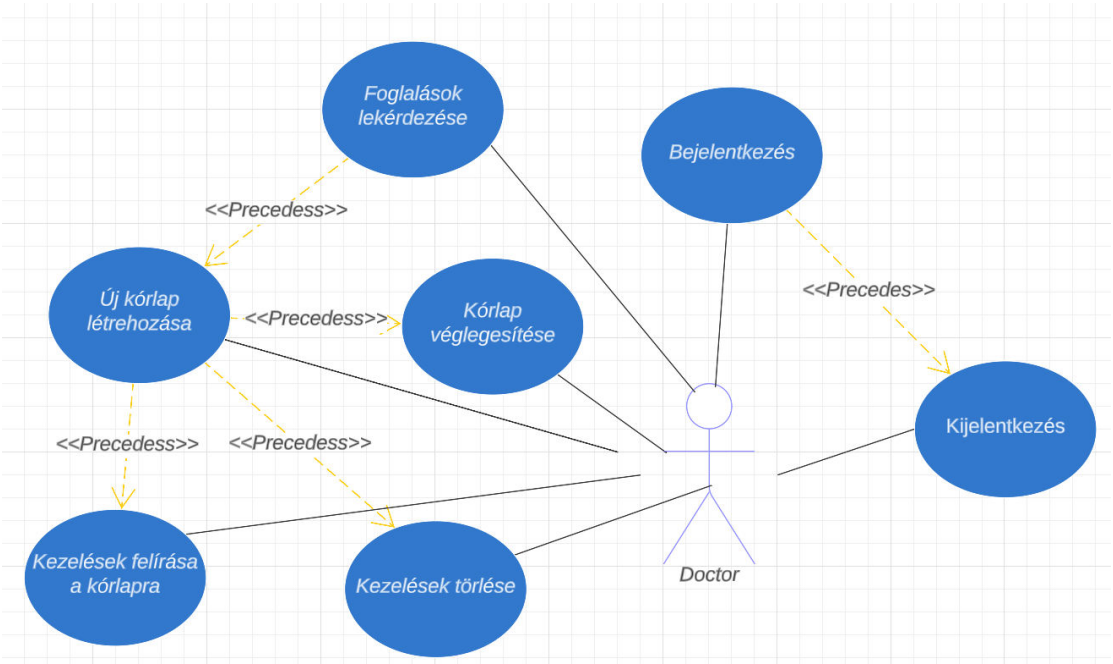
• Bejelentkezést követően listázódnak (a kórlappal még nem rendelkező) foglalások (időpont, foglaló neve, kategória). Egy foglalást kiválasztva új kórlap nyitható (a páciens adatai és az időpont automatikusan áttöltődnek).

• A kórlapra tetszőleges számú tétel (pl. kezelési költség, különféle gyógyszerköltség) vihető fel, szabad szöveges bevitellel és az összeg megadásával. Ezeket törölni is lehet a felvitel után. A kórlapon látható a végösszeg, amely az egyes tételek hozzáadásával/törlésével változik. Végül az orvos véglegesítheti a kórlapot (ehhez a program kérjen megerősítést).

2. Elemzés:

Az alkalmázoshoz létrehozunk egy új WebAPI projektet, ami a WebApi-s Controller végpontokat tartalmazza majd az asztali alkalmazáshoz, ami HTTP klienssel éri el a WebAPI szolgáltatásait**.** A webAPI controllerei kérik le a foglalásokat illetve módosítják a kórlapokat illetve a hozzájuk tartozó kezeléseket. Az adatbázis inicializálása átkerült a WebAPI-s projekt Program.cs forrásfájlba így, a csak Webes felület elérése megköveleli a WebApi futtatását. Adatküldésre JSON szerializált DTO (Data Transfer objekteket használunk a foglalásokra, a kórlapokra illetve a kezelésekre is.

3. Felhasználói esetek diagrammja



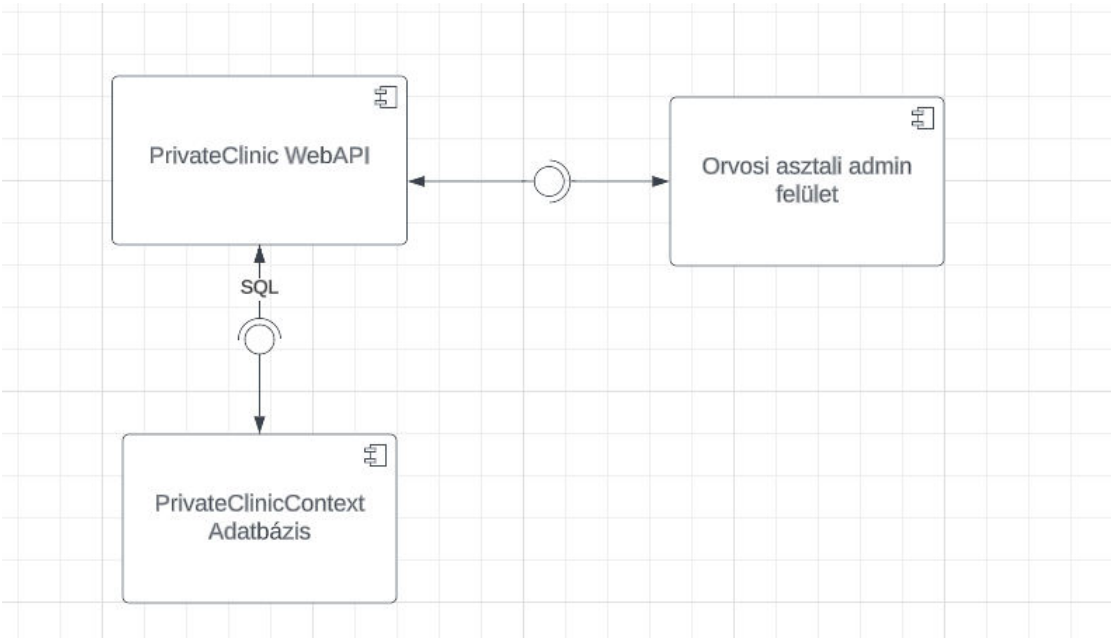
4. Tervezés:

4.1 Programszerkezet

A WebAPI-t WebAPI projekttel valósítjuk meg az asztali alkalmazás pedig egy WPF projekt lesz. A WebAPI 4 féle kontrollert tartalmaz: AccountController, BookingsController, MedicalRecordsController, TreatmentsController. Az asztali alkalmazás tartalma egy Service osztályt ami HTTP Klienssel kommunikál az API-val. Modell, ViewModell és View rétegekkel MVVM architektúra alapján felépítve. Az aztali alkalmazásnak van egy főabalaka (MainWindowl) amit csak a bejelentkezési ablakkal való sikeres bejelentkezéssel érjük el (LoginWindow) valamint a kórlap és kezelés szerkesztő ablak (TreatmentEditorWindow)

EntityFrameWorkCore segítségével tároljuk az adatokat az adatbázisban, amit külön Perzisztencia projektbe van szervezünk ki és mind az eddig megvalósított webes felület mind a WebAPI eléri. A WebAPI Program.cs forrásfájlja inicializálja az adatbázist így előkövetelménye a webes alkalmazás helyes működésének is.

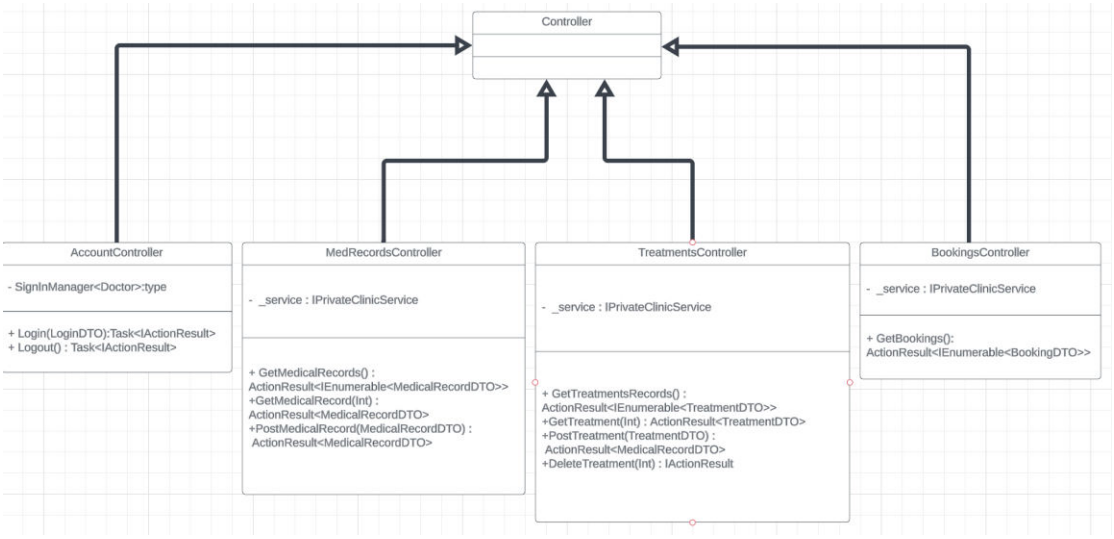
4.2 Az alkalmazás komponensdiagrammja



5. WebAPI

5.1 A vezérlők osztálydiagrammja

Mind a négy vezérlő egy közös ősosztályból, a controllerbase-ből származik le.

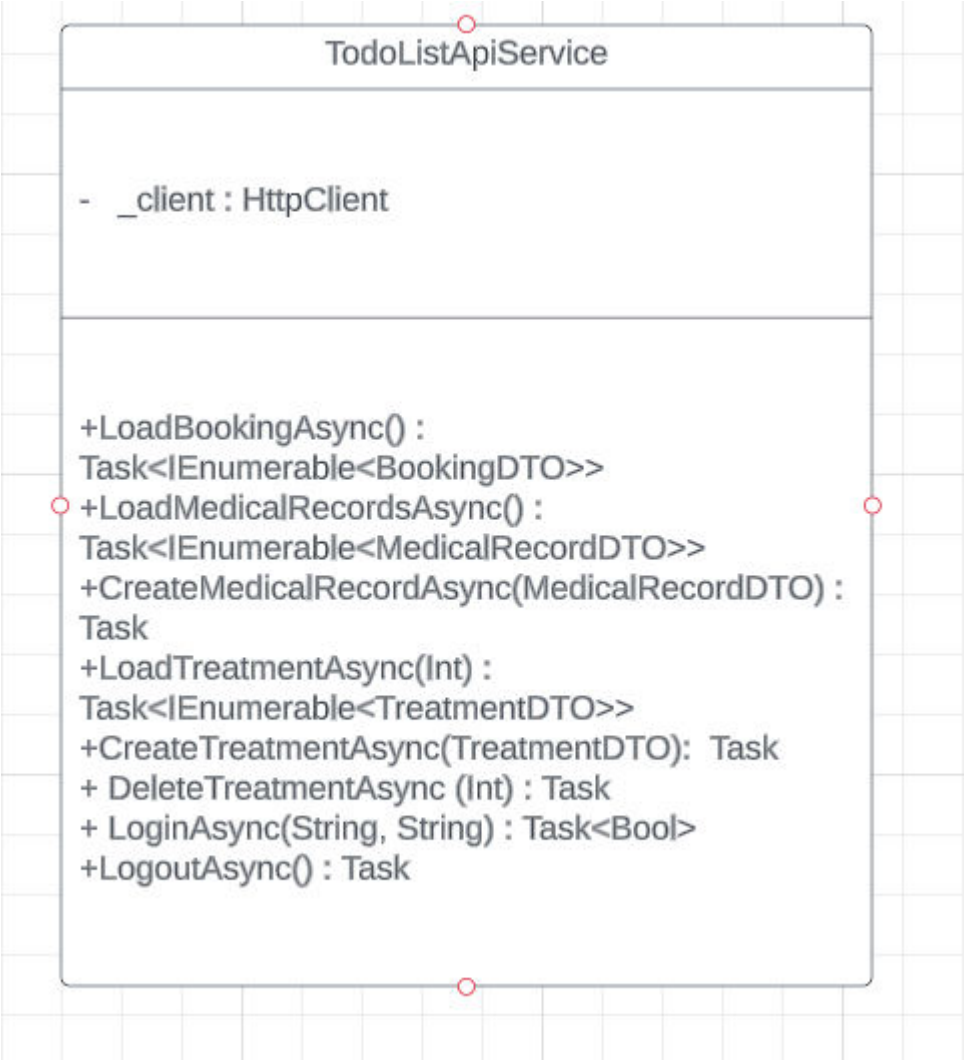


6. WPF Alkalmazás

A WPF alkalmazás MVVM architektúrában készült.

6.1 Model:

Egy klienst tart karban ami az API-val létesít kapcsolatot a következő osztályarchitektúrával:



6.2. Nézetmodell

A nézetmodell megvalósításához felhasználunk egy általános utasítás (DelegateCommand), valamint egy ős változásjelző (ViewModelBase) osztályt. A nézetmodell fő feladatait a MainViewModel osztály látja el, ami bejelentkezés után érhető el (LoginViewModel). A kijelölés változtatási funkciók miatt létrehoztuk a BookingViewModel egy MedicalRecordViewModel és egy TreatmentViewModel osztályokat.

6.3 Nézet

A nézetet a WPF alkalmazás biztosítja. 3 Window alkotja a nézetet:

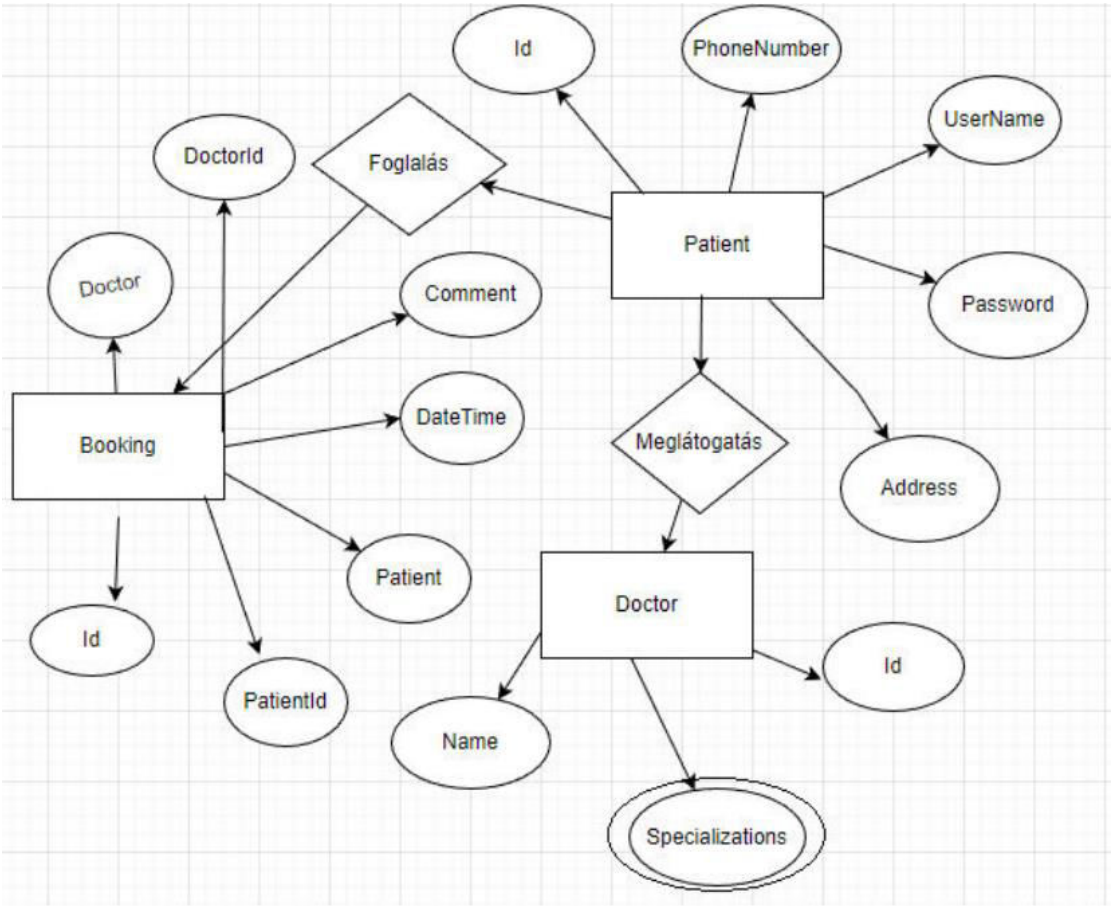
- LoginWindow: Bejelentkezésért felel

- TreatmentEditorWindow: A kórlap és a kezelésekért fele

- MainWindow: A fő ablak ahol listázódnak a foglalások

7. Adatbázis:

Az adatbázis nem változott nagyban az előző részfeladathoz képest. Az orvosok és a páciensek is az ApplicationUser osztályból származnak, ami az IdentityUser-ből. A és a Treatment és a MedicalRecord osztályok között sok az egyhez kapcsolat van. A MedicalRecord osztály majdnem megegyezk a foglalási osztállyal. A páciens és doktor és időpont között létesít kapcsolatot kiegészítve a kezelésekkel. A DTO (Data Transfer Objekt) elrejtik az adatbázis ábrázolását lényeges adatokra kitérve és az egymásra utaló végtelen ciklusokat is (MedicalRecord és Treatmentek kezelésénél)



8. Tesztek:

Az alkalmazásban a WebAPI-hoz egységteszteket készítettünk Xtest környezet segítsével.

A következő tesztesetek kerültek megvalósításra:

GetMedicalRecordsTest(): Megfelelő mennyiségű MedicalRecord lekérdezése

GetMedicalRecordByIdTest(Int32 id): Adott MedicalRecord lekérdezése

GetInvalidMedicalRecordTest(): Helytelen azonosítójú MedicalRecord kezelése

PutMedicalRecordTest(Int32 id): Adott MedicalRecord változtatása

PostMedicalRecordTest(): MedicalRecord objektum létrehozása