2. beadandó

Gasparin Zsombor, RVDSN3

Feladatleírás

Készítsünk kliens-szerver rendszert, amellyel egy magánkórház bejelentkezési naptárát, valamint a betegek kórlapjának kezelését tudjuk elősegíteni.

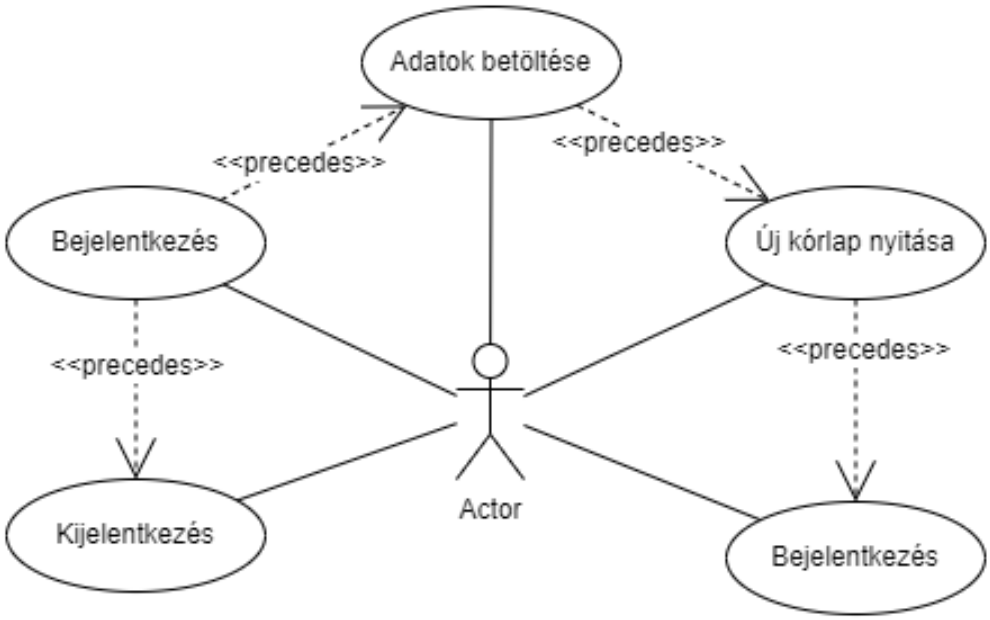
Az orvosok az előjegyzéseket az asztali grafikus felületen keresztül adminisztrálják.

- 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).

Elemzés

- Az webalkalmazáshoz **ASP.NET WebAPI** keretrendszert használunk
- A webszolgáltatást **MVC** architektúrában írjuk meg
- Az asztali adminisztrációs alkalmazást **WPF** grafikus keretrendszer segítségével valósítjuk meg, **MVVM** architektúrában
- Az adatok tárolását **Entity Framework Core** keretrendszer biztosítja
- A felhasználók jogosultságáért az **Entity Framework Core**-al együttműködő Identity könyvtárat használjuk
- Az alkalmazás három ablakból áll:
  - A bejelentkező ablak, ami a felhasználónevet és a jelszót kéri
  - A főablakot, ahol listázva vannak az orvoshoz tartozó foglalások
  - A kórlap ablak, ahol ki lehet tölteni a kórlapot

Felhasználói esetek diagramm



Rendszer tervezés

Az alkalmazás 4 projektből épül fel:

- A webszolgáltatásból: **Hospital.WebAPI**
- Az asztali alkalmazásból: **Hospital.Desktop**
- Az adatbáziskezelésért felelős projektből: **Hospital.Persistence**
- A kommunikációs osztályokat tartalmazó projektből: **Hospital.Data**

A webszolgáltatás

A webszolgáltatást **ASP.NET WebAPI** keretrendszerrel írjuk. A modell tartalmazza a szolgáltatásokat (Repository pattern): **DoctorService, MedicalRecordService, PatientService, ReservationService**. A controllerek felelősek a requestek fogadásáért és a megfelelő válasz visszaküldéséért: **DoctorsController, MedicalRecordsController, ReservationsController**.

DoctorsController

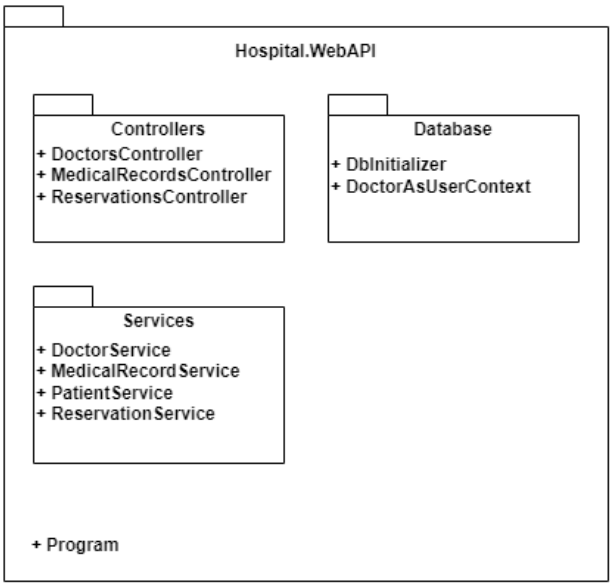
A kontroller 2 akciót valósít meg. A **Login()** POST metódus bejelentkezteti az orvost, ha megfelelő a felhasználó név és jelszó. Ilyenkor visszaküldi az orvos id-jét. Ha sikertelen a bejelentkezés akkor Unauthorized kódot küld vissza. A **Logout()** GET metódus kijelentkezteti az orvost.

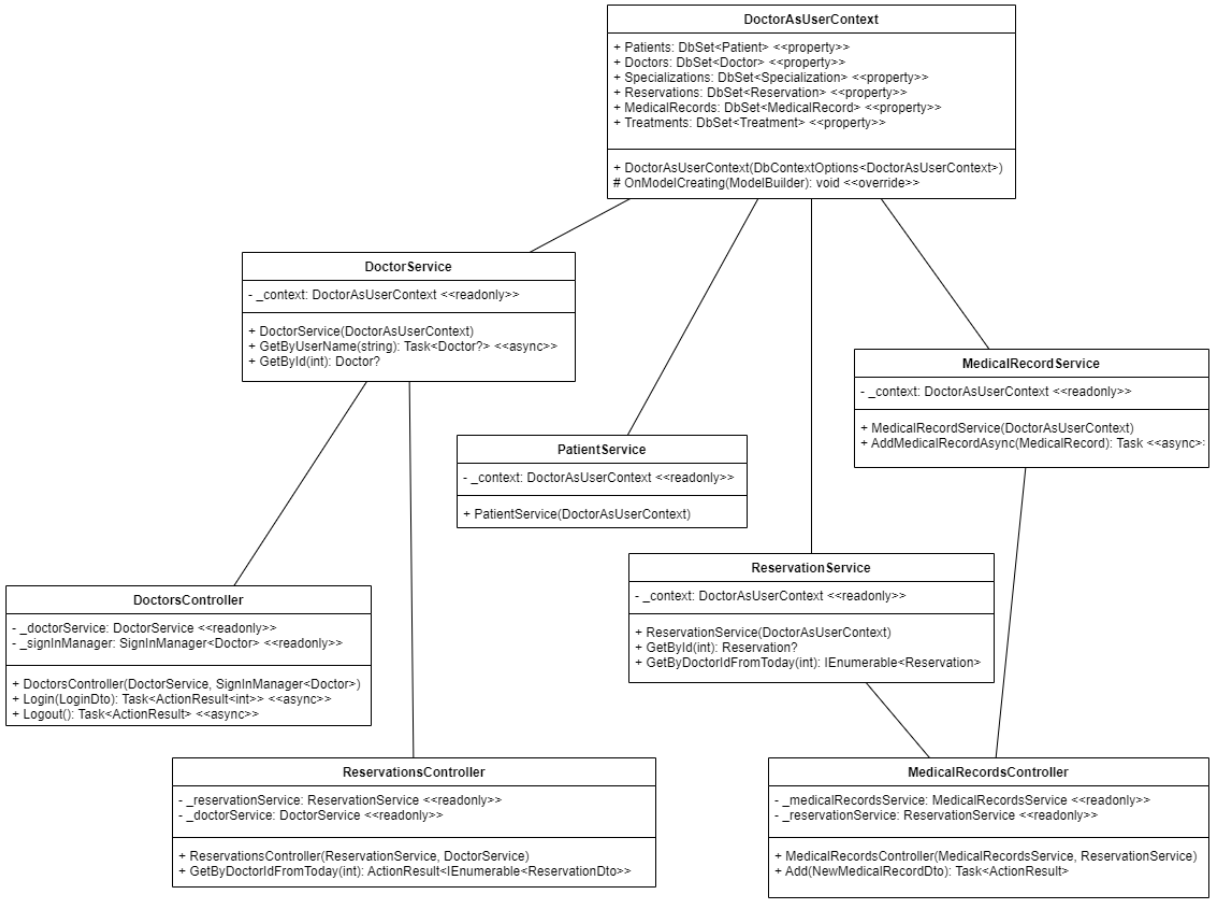
MedicalRecordsController

Ez a kontroller felelős az új kórlap felviteléért. Az **Add()** POST metódus hozzáad egy új kórlapot az adatbázishoz. Ezt a metódust csak bejelentkezve lehet elérni.

ReservationsController

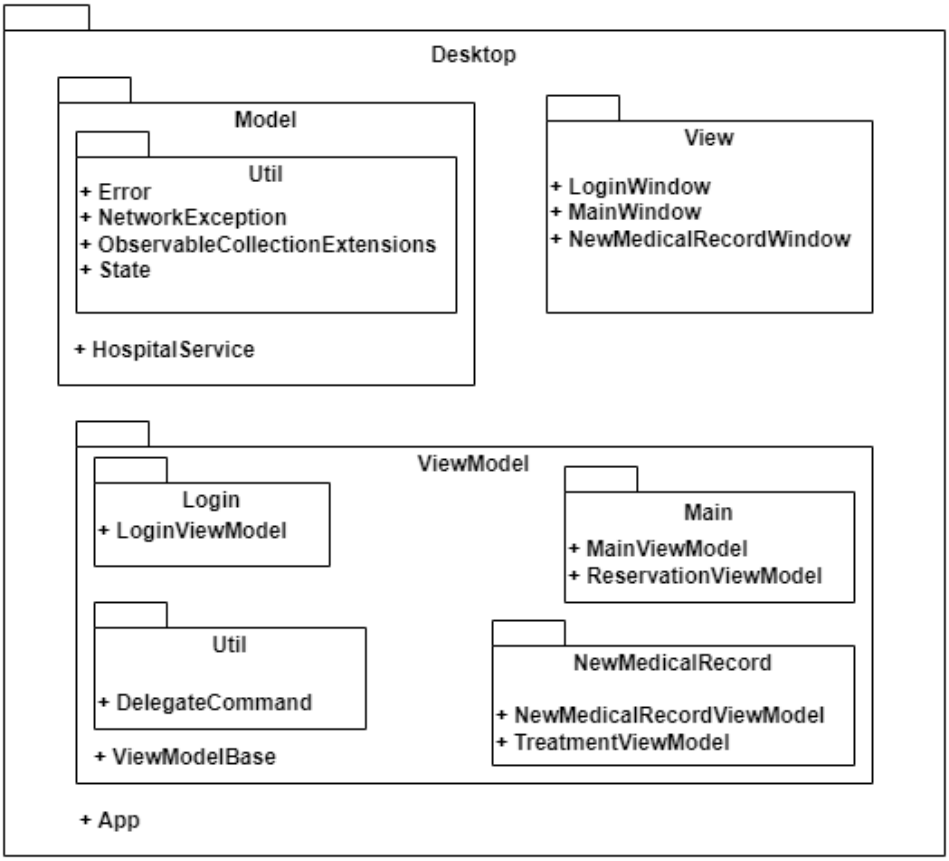
Ez a kontroller felelős a megfelelő foglalások lekérdezéséért. **GetByDoctorIdFromToday()**, GET típusú metódusa a doktor id-je alapján visszaadja azokat a jövőben foglalásokat, amik az orvoshoz tartoznak, illetve nem tartozik még hozzájuk kórlap.


Hospital.WebAPI
Controllers
+ DoctorsController
+ MedicalRecordsController
+ ReservationsController
Database
+ DbInitializer
+ DoctorAsUserContext
Services
+ DoctorService
+ MedicalRecordService
+ PatientService
+ ReservationService
+ Program


DoctorAsUserContext
+ Patients: DbSet<Patient> <<property>>
+ Doctors: DbSet<Doctor> <<property>>
+ Specializations: DbSet<Specialization> <<property>>
+ Reservations: DbSet<Reservation> <<property>>
+ MedicalRecords: DbSet<MedicalRecord> <<property>>
+ Treatments: DbSet<Treatment> <<property>>
+ DoctorAsUserContext(DbContextOptions<DoctorAsUserContext>)
# OnModelCreating(ModelBuilder): void <<override>>
DoctorService
- _context: DoctorAsUserContext <<readonly>>
+ DoctorService(DoctorAsUserContext)
+ GetByUserName(string): Task<Doctor?> <<async>>
+ GetById(int): Doctor?
MedicalRecordService
- _context: DoctorAsUserContext <<readonly>>
+ MedicalRecordService(DoctorAsUserContext)
+ AddMedicalRecordAsync(MedicalRecord): Task <<async>>
PatientService
- _context: DoctorAsUserContext <<readonly>>
+ PatientService(DoctorAsUserContext)
ReservationService
- _context: DoctorAsUserContext <<readonly>>
+ ReservationService(DoctorAsUserContext)
+ GetById(int): Reservation?
+ GetByDoctorIdFromToday(int): IEnumerable<Reservation>
DoctorsController
- _doctorService: DoctorService <<readonly>>
- _signInManager: SignInManager<Doctor> <<readonly>>
+ DoctorsController(DoctorService, SignInManager<Doctor>)
+ Login(LoginDto): Task<ActionResult<int>> <<async>>
+ Logout(): Task<ActionResult> <<async>>
ReservationsController
- _reservationService: ReservationService <<readonly>>
- _doctorService: DoctorService <<readonly>>
+ ReservationsController(ReservationService, DoctorService)
+ GetByDoctorIdFromToday(int): ActionResult<IEnumerable<ReservationDto>>
MedicalRecordsController
- _medicalRecordsService: MedicalRecordsService <<readonly>>
- _reservationService: ReservationService <<readonly>>
+ MedicalRecordsController(MedicalRecordsService, ReservationService)
+ Add(NewMedicalRecordDto): Task<ActionResult>

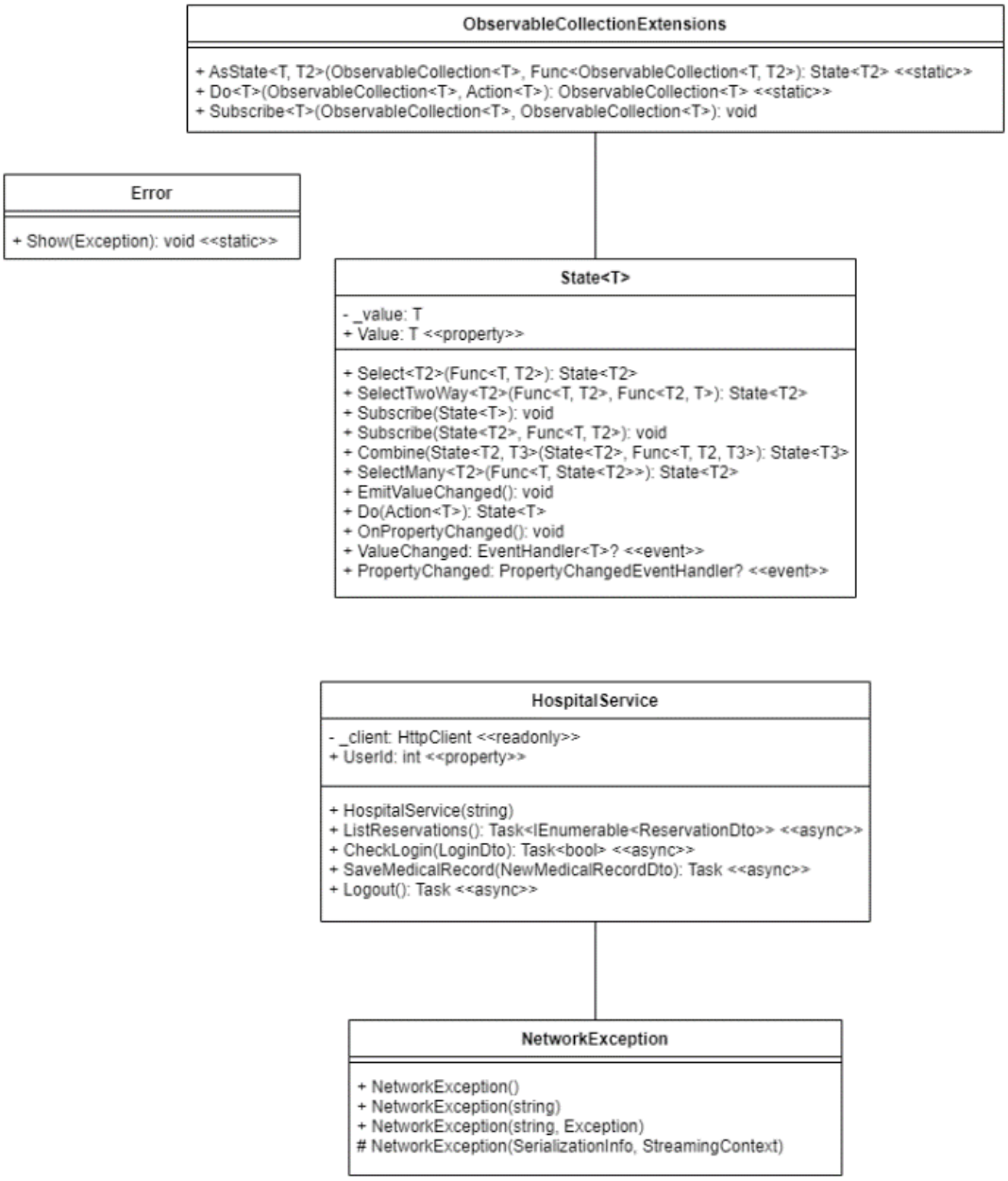
Asztali alkalmazás

Az asztali alkalmazást **WPF** keretrendszerben, **MVVM** architektúrában épült. Több csomagból épül fel.



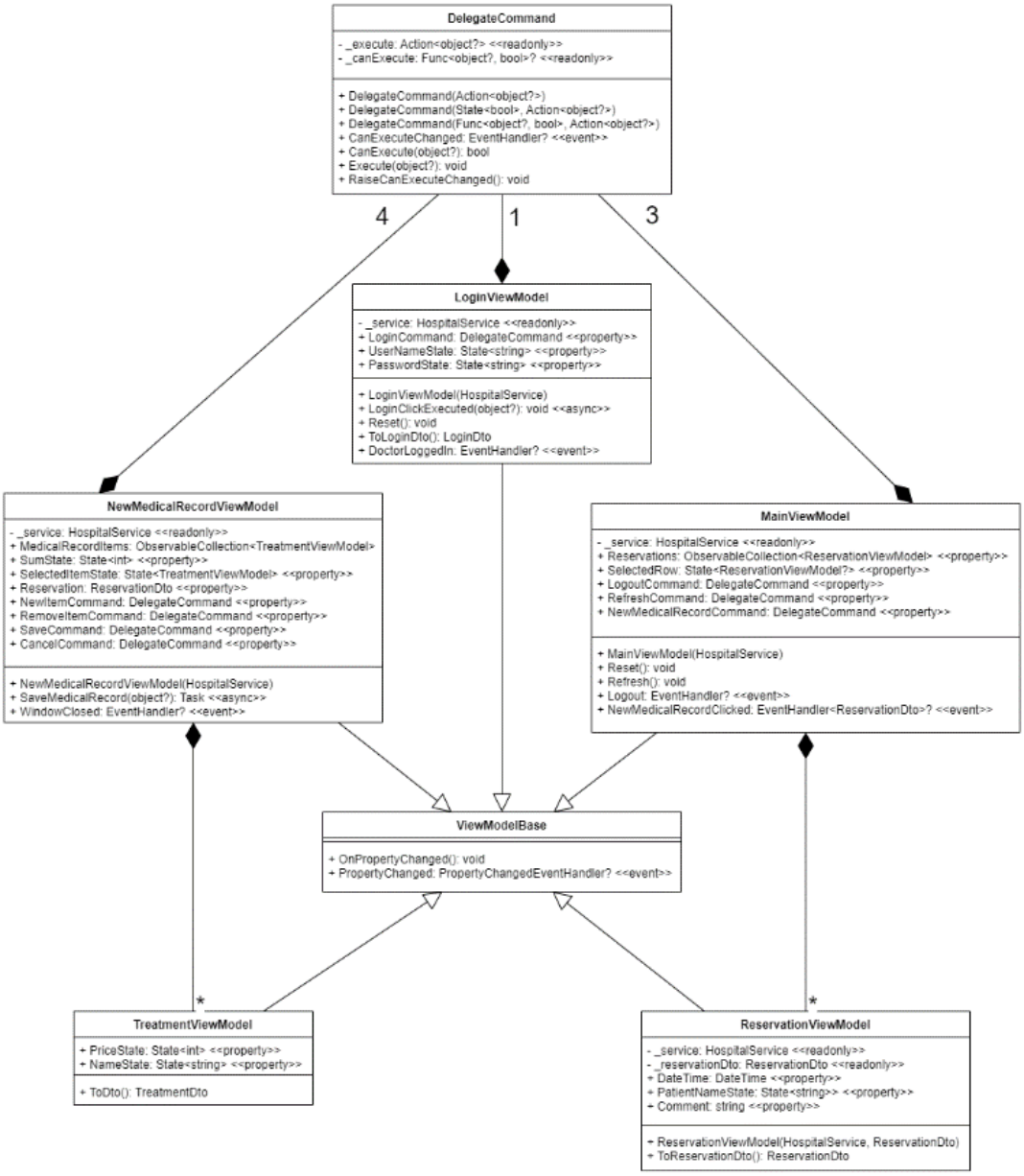
Modell

A modellben található a **HospitalService** osztály, aminek a webszerverrel való kommunikálás a fő feladata. Figyeli a szerver válaszait és **NetworkException**-t dob, ha sikertelen a kommunikáció, vagy a kérés. A **State** osztály **Megfigyelő mintát** (Observer pattern) implementál, segítségével deklaratív stílusban lehet leírni két adat között a kapcsolatot. Az **ObservableCollectionExtensions** osztály bővítménymetódusokat tartalmaz az ObservableCollection típushoz, ezekkel együtt tud működni a State osztállyal.



Nézetmodell

A nézetmodell felelős a modell adatait megjeleníthető formára alakításáért. A **ViewModelBase** osztály implementálja a INotifyPropertyChanged interfészt, hogy a nézet figyelni tudja az adatok változását, illetve módosítani tudja őket. A **DelegateCommand** osztály az ICommand interfészt implementálva alkalmas parancsok implementálása lambda függvények segítségével, illetve fel tudja használni a State osztályt, a parancs futtathatósága egy adattól függ.



Perzisztencia

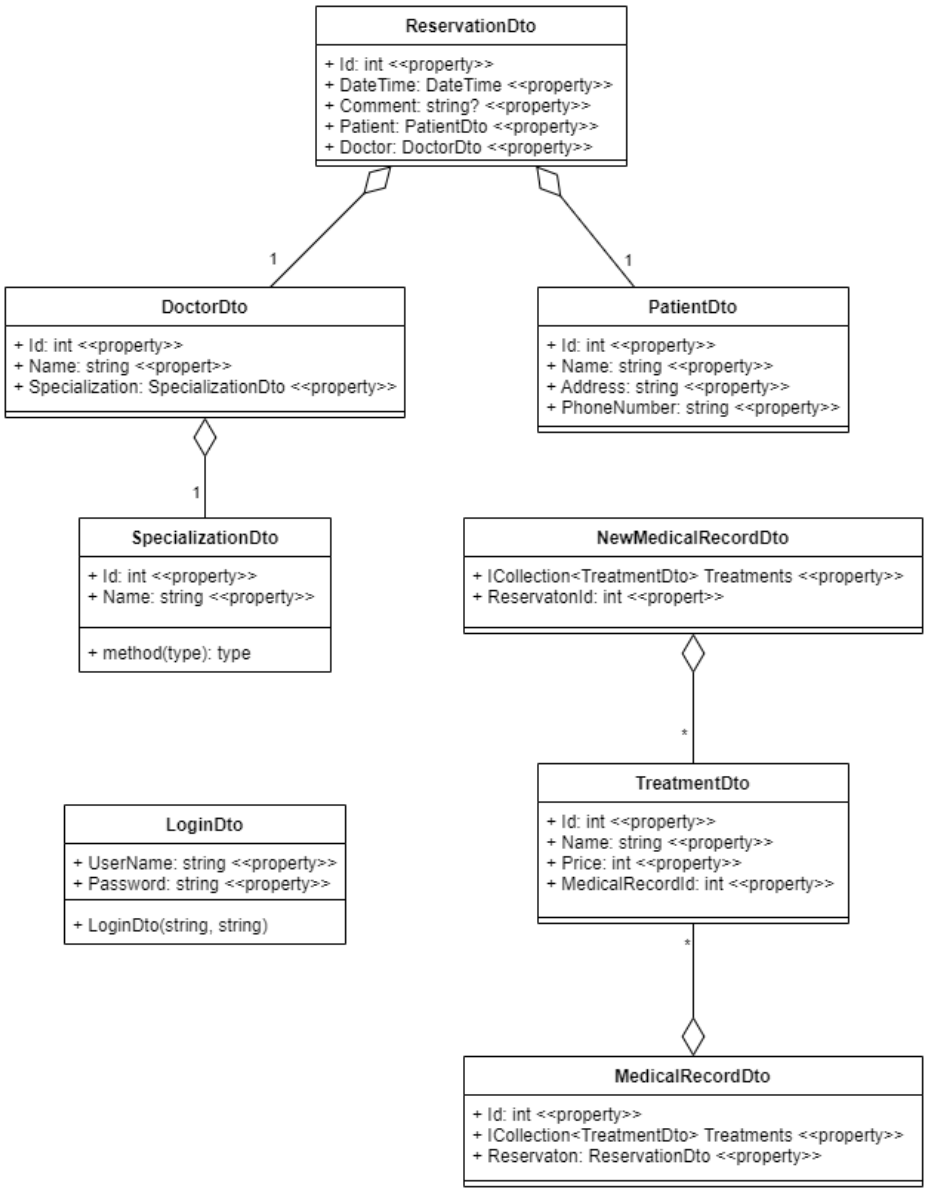
Az adatok tárolására **Entity Framework Core**-t használunk. A felhasználók kezelését az Identity NuGet csomag segítségével implementáljuk.

A használt adatbázis séma:

Az Entity Framework Core magától is létrehoz táblákat, ezek nem szerepelnek az ábrán.

Kommunikációs osztályok

A webszolgáltatás és az asztali alkalmazás közti kommunikáció **DTO**-k használatával végezzük. Nincsenek bennük körkörös referenciák, így alkalmasak a kommunikációra.



Tesztelés

A webszolgáltatás akciómetódusait **egységtesztek**kel teszteltük. A teszek a Hospital.WebAPI.Tests projektben találhatóak. A teszteléshez **xUnit** keretrendszert használtunk. Az orvosok be- és kijelentkezését teljes mértékben a beépített SignInManager osztály végzi, így az nem került tesztelésre.

MedicalRecordsController tesztelése:

- **AddMedicalRecordTest()**: leteszteli, hogy hozzá lehet-e adni egy helyes kórlapot az adatbázishoz és a metódus a megfelelő válasszal tér-e vissza
- **AddMedicalRecordWrongReservationIdTest()**: leteszteli, hogy ha rossz foglalási azonosítót adunk meg, akkor **NotFound** választ kapunk-e
- **AddMedicalRecordNoTreatmentTest()**: leteszteli, hogy ha a kórlapban nincs egy kezelés sem, akkor az nem valid, és **UnprocessableEntity** választ kapunk-e vissza

ReservationsController tesztelése:

- **ReservationsGetByDoctorIdFromToday()**: leteszteli, hogy létező doktor azonosítóval megfelelő választ ad-e vissza a metódus
- **ReservationsGetByDoctorIdFromTodayWrongDoctorId()**: leteszteli, hogy nem létező doctorId esetén **NotFound** választ ad-e vissza a metódus
- **ReservationsGetByDoctorIdFromTodayOwnReservations()**: leteszteli, hogy minden orvos csak a saját foglalásait kapja-e meg
- **ReservationsGetByDoctorIdFromTodayOnlyFuture()**: leteszteli, hogy csak jövőbeli foglalásokat adjon vissza az akció
- **ReservationsGetByDoctorIdFromTodayOnlyWithoutMedicalRecords()**: leteszteli, hogy csak olyan foglalásokat adjon vissza a metódus, amihez még nincs kórlap.