OBJEKTNO OBLIKOVANJE

SEMINAR – MEDIX

Tomislav Ante Brnčić, Ivana Prebeg

# Uvod

Medix je aplikacija koja služi za asistiranje medicinskom osoblju u obavljanju njihova posla. Olakšava zadatke vezane uz papirologiju, logistiku i organizaciju koje medicinsko osoblje svakodnevno obavlja ostavljajući prostora da se fokus prebaci na ono bitno – zdravlje pacijenata.

Aplikacija prati rad medicinske ustanove, pod medicinskom ustanovom se podrazumijeva zgrada kojoj se obavljaju medicinske aktivnosti u više odjela (npr. kardiologija, neurologija), u kojima sudjeluje više liječnika i sestara, odnosno pacijenata. Rad u aplikaciji započinje prijavom u sustav odnosno osiguravanjem da informacijama može pristupiti samo autorizirano osoblje. Nakon uspješne prijave započinje rad. Pacijenti dolaze sestrama sa svojim uputnicama, koje zatim sestre unose u sustav te ih prosljeđuju odgovarajućem doktoru. Uputnica sadrži informacije o pacijentu te ostale informacije potrebne za njegov pregled (je li pacijent osiguran, koliko je pregled hitan, koliko pregled košta itd.). Unosom uputnice sustav pronalazi pacijenta te ga stavlja na popis pacijenata koji čekaju pregled kod nekog doktora, odnosno na nekom odjelu. Uloga sestre je u tom slučaju izvršena, odnosno ona ponavlja istu aktivnost tijekom svog radnog vremena nakon čega se odjavljuje iz sustava.

Doktor se na početku svojeg radnog vremena prijavljuje u sustav koristeći korisničko ime i lozinku danu od sustava. Prijava omogućava pregled podataka o svim pacijentima kao i pacijentima koji čekaju pregled na odjelu koji je doktoru pripisan. Odabirom pacijenta s popisa iz čekaonice doktor započinje pregled. Sustav mu prikazuje sve potrebne informacije za obavljanje pregleda. Liječnik treba obaviti pregled i zatim popuniti odgovarajuće obrasce u sustavu koji će drugim liječnicima pružiti informacije bitne za daljnje preglede, odnosno terapiju. Sustav osigurava pregled povijesti bolesti, općih informacija o pacijentu, te prijašnje izdanih nalaza. Liječnik osim toga ispunjava anamnezu, status te završnu dijagnozu. Sustav omogućuje odabir dijagnoze iz medicinskih imena te sparivanje s njihovim latinskim nazivom, sve u svrhu bolje razumljivosti. Unosom i potvrdom unesenih podataka je pregled završen, ažurirani pacijentovi podatci su u sustavu te se automatski miče s liste pregleda u čekaonici. Na pregled dolazi novi pacijent. Odabir vrši liječnik svojom voljom, no pacijenti su sortirani u čekaonici ovisno o hitnosti odnosno vremenu dolaska kako bi se osiguralo prioritiziranje hitnih pregleda. Na samom kraju liječnik može vidjeti statistiku obavljenih pregleda u određenom razdoblju. Statistika pokazuje dijagnozu, pacijente, ali i broj zarađenih sredstava – odnosno sumu cijene svih obavljenih pregleda u ovisnosti o osiguranju pacijenta. Time završava rad liječnika u sustavu, rad završava odjavom.

Sustav sadrži i informacije bitne za rad osoblja, njihov poseban ID u ustanovi te lozinku kojom se prijavljuju u sustav i obavljaju operacije koje su im dostupne na odgovarajućoj razini prava.

Sustav Medix rađen je po uzoru na postojeći sustav koji se koristi u medicinskim ustanovama, pokušavajući ga optimizirati kako bi bio bolje prilagođen liječnicima. U tu svrhu je izbačeno pisanje kodova za odgovarajuće djelatnosti zamijenjujući ih padajućim izbornicima s jasno definiranim detaljima, te općenito pristupačnije i razumljivije korisničko sučelje.

# Zahtjevi

## Use Cases

|  |  |
| --- | --- |
| Naziv | UC1 – Prijava korisnika u sustav |
| Opis | Pri početku rada u sustavu, korisnik se prijavljuje kako bi vidio osjetljiv sadržaj (osobne podatke pacijenta) nužan za obavljanje pregleda. |
| Glavni akter | Medicinska sestra / liječnik |
| Sporedni akteri |  |
| Preduvjet | Korisnik je prethodno unesen u sustav |
| Okidač | Korisnik pokušava pristupiti sustavu uz pomoć izbornika za pristup |
| Glavni uspješni scenario | 1. Korisnik unosi podatke (korisničko ime, lozinku) potrebne za prijavu. 2. Sustav provjerava jesu li podaci koje je korisnik unio autentični. 3. Uneseni podaci su autentični - korisniku se omogućuje pristup sustavu. |
| Ekstenzije | 1. Uneseni podaci nisu autentični.    1. Sustav obavještava korisnika o neispravnosti unesenih podataka.    2. Korisnik ponavlja postupak započevši od koraka 1 glavnog scenarija. |
| Uvjeti koji trebaju biti zadovoljeni | Korisnik je prijavljen. |

|  |  |
| --- | --- |
| Naziv | UC2 – Unos novog pregleda |
| Opis | Medicinska sestra unosi podatke bitne za pregled pacijenta. |
| Glavni akter | Korisnik - medicinska sestra / liječnik |
| Sporedni akteri | Pacijent |
| Preduvjet | Korisnik je prijavljen u sustav |
| Okidač | Korisnik odabire opciju unosa novog pregleda na formi. |
| Glavni uspješni scenario | 1. Korisnik ispunjava formu pružajući informacije o pregledu    1. Korisnik unosi podatke o pacijentu ako je pacijent prvi put na pregledu    2. Korisnik unosi podatke o pregledu 2. Sustav provjerava ispravnost unesenih podataka 3. Sustav pohranjuje informacije i raspoređuje pacijenta u čekaonicu 4. Pacijent se nalazi na listi čekanja za pregled |
| Ekstenzije | 1. Pacijent je već bio na pregledu    1. Korisnik pretražuje i odabire pacijenta (UC3)    2. Sustav ispunjuje podatke o pacijentu u formi 2. Pružene informacije nisu ispravne    1. Sustav obavještava korisnika o neispravnosti unesenih podataka.    2. Korisnik ponavlja postupak započevši od koraka 1 glavnog scenarija. 3. Čekaonica odjela kojeg pacijent treba ne postoji u ustanovi |
| Uvjeti koji trebaju biti zadovoljeni | Pregled je unesen u sustav i nalazi se u listi pregleda čekaonice. |

|  |  |
| --- | --- |
| Naziv | UC3 – Pretraga pacijenata |
| Opis | Korisnik pretražuje listu pacijenata unosom prezimena pacijenta. |
| Glavni akter | Korisnik - medicinska sestra / liječnik |
| Sporedni akteri | Pacijent |
| Preduvjet | Korisnik je prijavljen u sustav |
| Okidač | Korisnik odabire opciju pretrage pacijenata iz odgovarajuće forme |
| Glavni uspješni scenario | 1. Korisnik unosi prezime traženog pacijenta u odgovarajuću formu za unos 2. Korisnik odabire opciju pretrage 3. Sustav pretražuje bazu podataka i vraća listu pacijenata koji zadovoljavaju uvjet 4. Korisnik odabire pacijenta iz liste |
| Ekstenzije | 1. Pacijent ne postoji u bazi podataka    1. Sustav obavještava korisnika da traženi pacijent ne postoji    2. Korisnik odustaje od pretrage |
| Uvjeti koji trebaju biti zadovoljeni |  |

|  |  |
| --- | --- |
| Naziv | UC4 – Odabir pregleda |
| Opis | Korisnik odabire pregled iz liste pregleda u čekaonici |
| Glavni akter | Korisnik – liječnik |
| Sporedni akteri | Pacijent |
| Preduvjet | Korisnik je prijavljen u sustav |
| Okidač |  |
| Glavni uspješni scenario | 1. Korisnik odabire pacijenta iz liste pacijenata koji se nalaze u čekaonici 2. Sustav prikazuje formu za pregled |
| Ekstenzije | 1. Drugi pacijent je u trenutku odabir na pregledu    1. Sustav obavještava korisnika |
| Uvjeti koji trebaju biti zadovoljeni |  |

|  |  |
| --- | --- |
| Naziv | UC5 – Obavljanje pregleda |
| Opis | Korisnik ispunjava podatke relevantne za daljnje liječenje pacijenta |
| Glavni akter | Korisnik - liječnik |
| Sporedni akteri | Pacijent |
| Preduvjet | Korisnik je prijavljen u sustav;  Pacijent je unesen u sustav;  Pregled (uputnica) je unesen u sustav |
| Okidač | Korisnik je odabrao pacijenta s popisa čekaonice |
| Glavni uspješni scenario | 1. Sustav otvara formu za unos podataka vezanih uz pregled 2. Liječnik pregledava podatke o pacijentu koji uključuju opće informacije i povijest bolesti 3. Liječnik ispunjava podatke vezane za pregled (nalaz, anamnezu, terapiju) 4. Liječnik odabire dijagnozu 5. Liječnik potvrđuje unos podataka, zatvara pregled |
| Ekstenzije |  |
| Uvjeti koji trebaju biti zadovoljeni | Nalaz je pohranjen u bazu podataka. |

|  |  |
| --- | --- |
| Naziv | UC6 – Pregled statistike |
| Opis | Pregled statistike liječnika, analiza pregleda u određenom vremenskom razdoblju |
| Glavni akter | Korisnik - iječnik |
| Sporedni akteri |  |
| Preduvjet | Korisnik je prijavljen u sustav |
| Glavni uspješni scenario | 1. Korisnik odabire vremenski period tijekom kojeg želi vidjeti statistiku svog rada 2. Sustav izračunava te prikazuje statističke podatke poput dijagnoze i ukupne cijene naplaćenih pregleda |
| Ekstenzije |  |
| Uvjeti koji trebaju biti zadovoljeni |  |

|  |  |
| --- | --- |
| Naziv | UC7 – Odjava |
| Opis | Odjava iz sustava |
| Glavni akter | Korisnik - medicinska sestra / liječnik |
| Sporedni akteri |  |
| Preduvjet | Korisnik je prijavljen u sustav |
| Okidač | Korisnik odabire opciju odjave iz sustava u glavnom izborniku |
| Glavni uspješni scenario | 1. Sustav odjavljuje korisnika |
| Ekstenzije |  |
| Uvjeti koji trebaju biti zadovoljeni | Korisnik je odjavljen. |

## API specifikacija

Aplikacija bi trebala organizirana u nekoliko slojeva. Upravljači viših slojeva komuniciraju sa servisom koji služi kao poveznica za niže slojeve. Servise sa bazom povezuje sloj koji sadrži repozitorije.

Baza bi trebala sadržavati tablice sa podatcima koji će se koristiti u radu aplikacije. Repozitoriji služe kako bi centralizirali dohvaćanje podataka odnosno komunikaciju s bazom te osigurali primjenu određenih pravila i zadovoljavanja uvjeta. Također, na taj smo način učinili kod mnogo čitljivijim i lakšim za razumjeti, što pomaže u slučaju da ga je potrebno mijenjati ili nadograđivati.

Na sloj repozitorija veže se sloj servisa koji služe za obavljanje operacija čiji će se rezultat kasnije prikazivati na prezentacijskom sloju desktop i web aplikacije.

### Repository Layer

Ovaj sloj bi trebao sadržavati repozitorije oblikovane na temelju modela za svaki od objekata potrebnih za rad aplikacije. Trebao bi sadržavati repozitorije za upravljanje podatcima vezanih uz medicinsko osoblje, pacijente, izvršene preglede, dijagnoze i terapije. Budući da većina tih podataka zahtjeva gotovo jednake metode za dohvat iz baze, napravljeno je sučelje koje sadrži set osnovnih operacija koje svaki repozitorij treba imati. To su:

* GetAll – funkcija koja iz baze dohvaća sve objekte neke klase
* GetById – funkcija koja dohvaća objekt koji sadrži prosljeđen id
* Add – funkcija koja dodaje objekt u bazu
* Update – funkcija koja mijenja objekt u bazi
* Delete – funkcija koja briše objekt iz baze

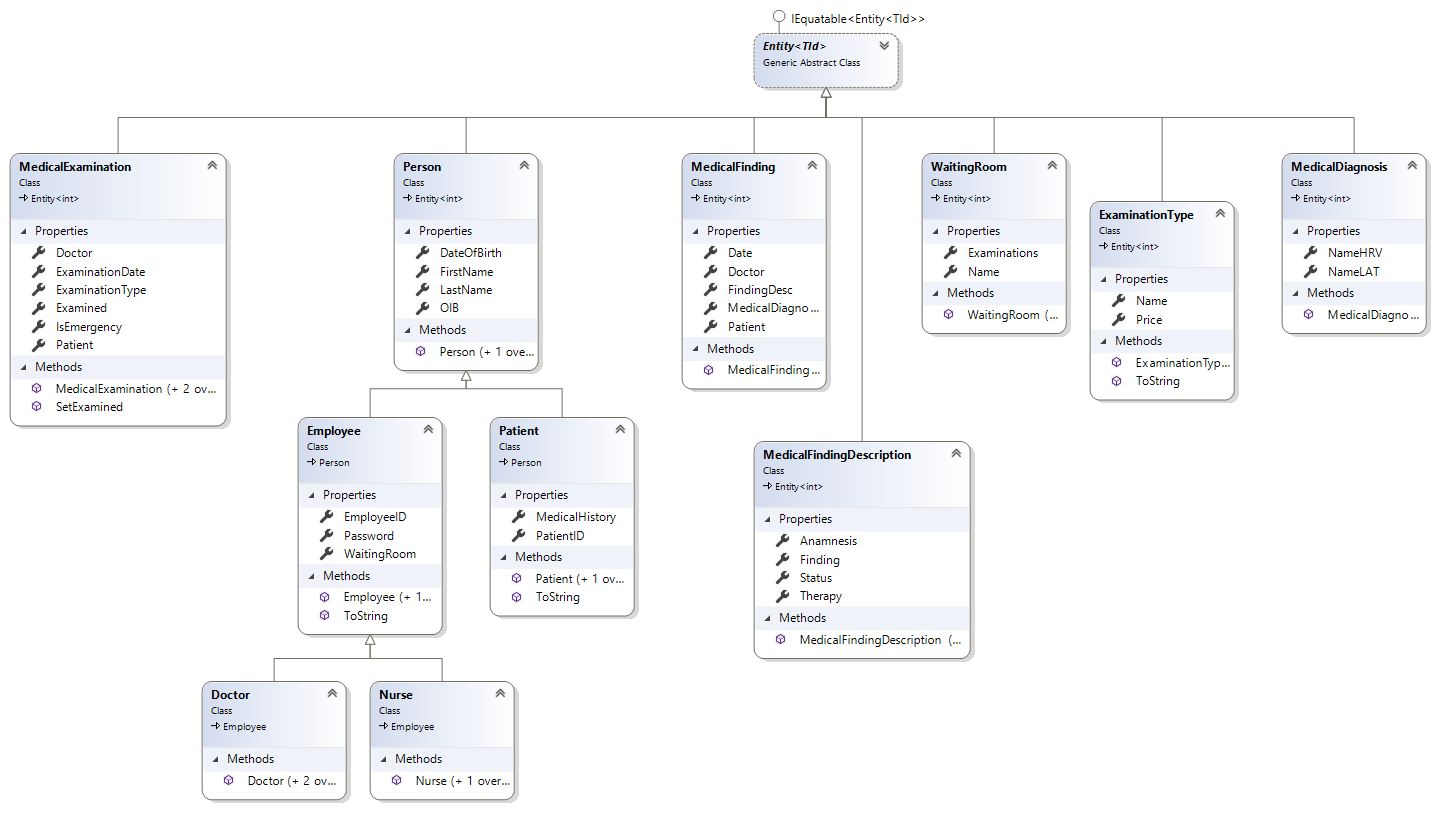
Osim njih, kreirani su specifični repozitoriji za svaki model koji omogućuju dodatne operacije.

* Liječnik : GetDoctorByIdAndPassword – funkcija namijenjena za olakšavanje prijave u sustav
* Pacijent : GetAllByLastName – funkcija koja kao parametar prihvaća prezime pacijenta te iz baze dohvaća sve pacijente s navedenim prezimenom
* Pregled : GetAllUnExaminedForDoctor – funkcija namijenjena punjenju čekaonice, odnosno dohvaćanja svih pregleda zakazanih kod nekog liječnika koji nisu obavljeni
* Pregled : GetAllExaminedToday – funkcija koja vraća listu svih pregleda jednog liječnika obavljenih u specificiranom danu koja služi kao pomoć za izračunavanje statistike

### Service Layer

Sloj servisa oblikovan je po uzoru na repozitorije, odnosno prosljeđuje već navedene funkcije upravljačima desktop i web aplikacije. Osim toga, servis sloj bi trebao imati dodatne razrede koji bi nosili logiku za odrađivanje operacija nevezanih usko uz bilo koju klasu, operacije poput prijave u sustav, pretraživanja podataka, te računanja statistike obavljenog posla za svakog liječnika.

# Opis modela



Klasa Employee predstavlja medicinsko osoblje, koristi se prilikom prijave u sustav za utvrđivanje pristupa informacija, a služi i kao spremnik osnovnih informacija o osoblju. Klasa Pacijent predstavlja osobu koja koristi medicinsku ustanovu, te služi kao spremnik osnovnih informacija ali i medicinske povijesti, odnosno sadrži referencu sa pohranjenim prijašnjim pregledima. MedicalExmination služi za opisivanje pregleda. Metoda SetExamined označava pregled izvršenim, izvršeni pregledi koriste se kasnije za izračunavanje statisike, dok se neizvršeni prikazuju u čekaonici kako bi liječnik znao koga još nije pregledao.

Klasa MedicalFinding služi za pohranjivanje informacija u medicinskom nalazu. FindingDescription i MedicalDiagnosis odvojeni su u različite klase kako bi se u budućnosti mogli stvoriti drugačiji opisi (koji bi npr sadržavali i RTG snimke ili neke druge podatke) te kako bi se liječnicima omogućio lakši odabir dijagnoze (povezivanje hrvatskih i latinskih naziva).

FindingDescription sadrži polja Anamnezu, Status, Terapiju i Nalaz. Anamneza opisuje događaje koje prethode sadašnjem stanju pacijenta (npr. kako se osjećao posljednjih dana ili što ga je nagnao da dođe na pregled, Status opisuje stanje pacijentna – ima li temperaturu, je li pri svijesti, diše li i slično. Terapija se koristi za propisivanje terapije, a Nalaz sadrži opis liječničkih opaski.

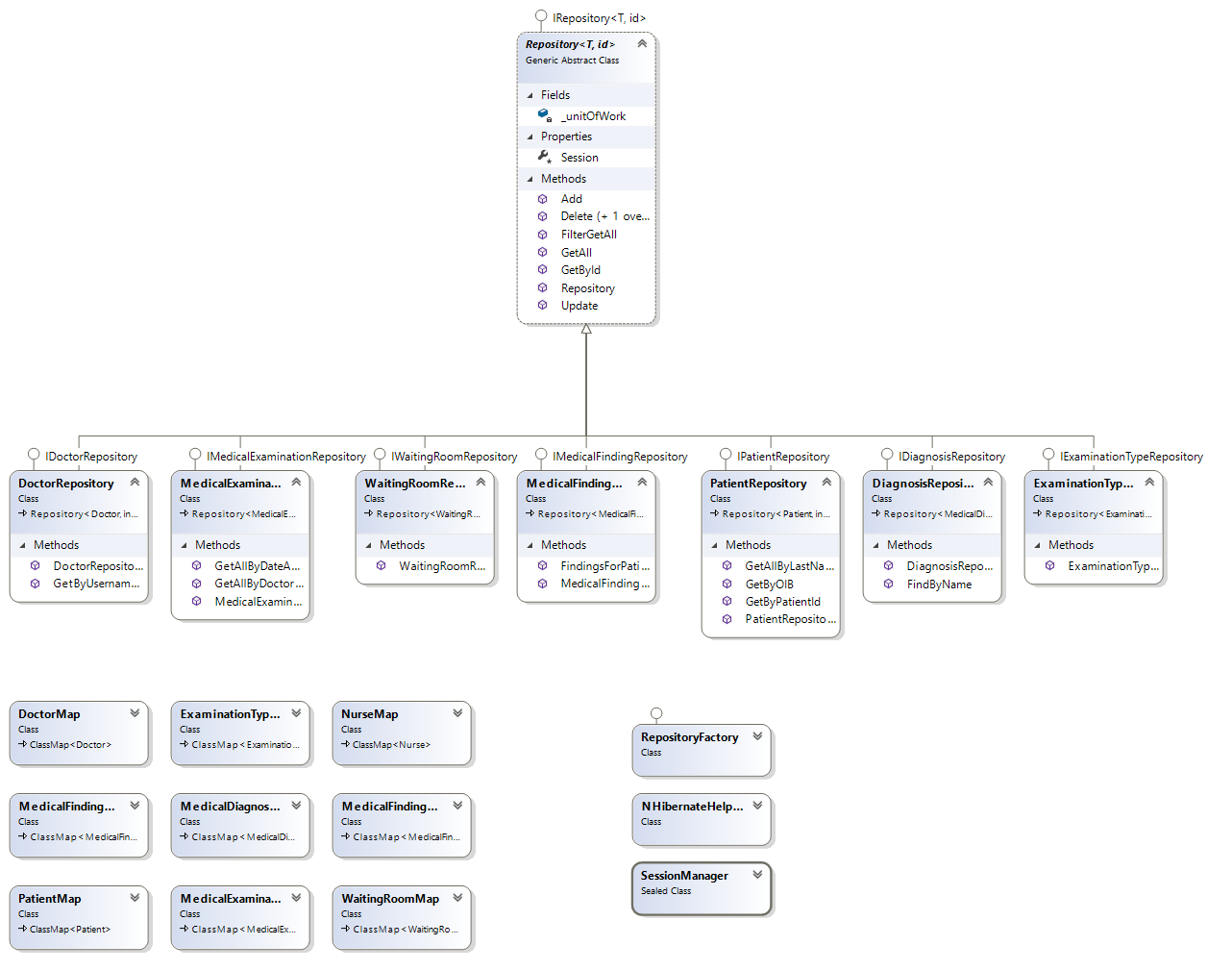
ExaminationType opisuje tipove pregleda te sadrži podatke o njihovim cijenama. Ovo se odnosi na hitni pregled, pregled u svrhu kontrole i slično. Pretežito se koristi za računanje statistike (koliko je sredstava liječnik osigurao), te kako bi pacijenti s naznakom “HITNO” mogli biti pregledani što prije.

osnova je class diagram vašeg objektnog modela, koji po potrebi možete dodatno proširiti kratkim tekstualnim opisom pojedinih klasa (što su im odgovornosti, relevantno ponašanje i sve ono što se iz statičkog dijagrama klasa ne vidi). Naznačiti gdje u modelu imate entitete, value objekte i servise, a gdje su vam agregati, „tvornice“ i repozitoriji.

# Opis implementacije perzistencije

Centralni dio aplikacije služi za njenu komunikaciju s bazom. Sadrži tri sloja. Baza komunicira sa DataAccess slojem (sloj pristupa podatcima) te navedene podatke kroz repozitorije prosljeđuje BussinesLayer sloju (sloj poslovne logike odnosno servisa). Sloj servisa obavlja sve bitne manipulacije podatcima i složenije izračune ostavljajući prezentacijskoj logici samo one jednostavnije. Za opisivanje objekata svi se slojevi koriste Model klasama (opisano u poglavlju 3.)

## Sloj pristupa podatcima



Slika: DataAccess layer

Svi repozitoriji organizirani su kako bi komunicirali pomoću sučelja. Glavno sučelje je IRepository koje zahtjeva osnovnih 6 metoda opisanih u specifikaciji zahtjeva (vidi poglavlje 2.2.1). Klasa koja implementira to sučelje naziva se Repository, parametrizirana je po tipu T i id, T služi kao klasa za opis objekata koji će se tim repozitorijem dohvaćati, te id koji predstavlja ključ objekata spremljenih u bazi. Repository sadrži parametrizirane implementacije svih 6 metoda. Komunikacija se odvija pomoću objekta Session koji sadrži reference na UnitOfWork. Svaki od ostalih repozitorija nasljeđuje osnovnu klasu Repository te implementira vlastito sučelje koje opisuje dodatne metode koje treba podržavati. Npr. MedicalExaminationRepository dodatno podržava metodu GetAllByDoctorUnexamined koja dohvaća sve preglede za određenog liječnika koji još nisu izvršeni. Ovakvom se organizacijom smanjuje duplikacija koda (osnovnih 6 metoda) te se omogućuje nadogradnja bez promjene, odnosno promjene su lokalizirane i zahtjevaju dodatna mjenjanja prepostalih repozitorija. Također, svaki repozitorij služi za dohvaćanje objekata isključivo jedne klase kako bi se osiguralo načelo jedinstvene odgovornosti.

Klasa RepositoryFactory služi kao tvornica navedenih repozitorija.

### Mapiranja

Za mapiranja objekata s tablicama u SQL bazi koristili smo NHibernate, odnosno Fluent radni okvir.

#### PatientMap

Pacijent je mapiran pomoću klase PatientMap koja nasljeđuje ClassMap. Pojedina svojstva su mapirana naredbom Map, a povijest bolesti pacijenta s naredbo HasMany jer je povijest bolesti pacijenta predstavljena listom starih nalaza. Kod pacijenata je također poznata adresa stanovanja koja je realizirana kao objekt Address sa svojstvima ime ulice, grad i poštanski broj. S obzirom da adresa ne postoji u bazi kao zasebna tablica, a postoji objekt Address ona je realizirana kao komponenta. Članske varijable objekta Address su zapisane unutar retka pacijenta u bazi podataka. U mapiranju je to realizirano uporabom komponenti. Mapiranje AddressMap nasljeđuje ComponentMap i nema Id. Unutar PatientMap mapiranje adrese se obavlja korištenjem ključne riječi Component.

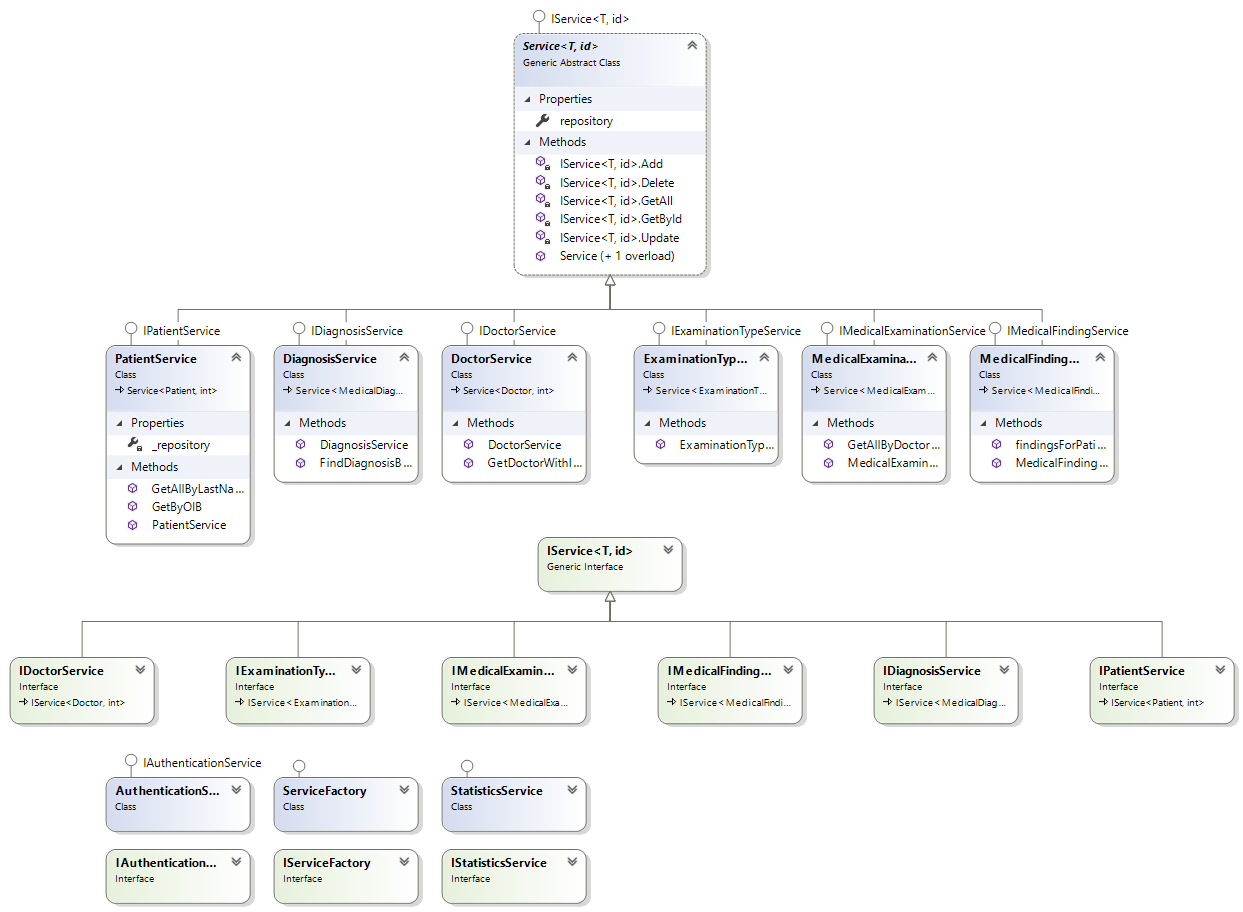
#### EmployeeMap, DoctorMap, NurseMap

Za mapiranje hijerarhije nasljeđivanja se koristi *table per class hierarchy.* Klase Employee, Doctor i Nurse su mapirane u jednu tablicu Employee koja sadrži sva svojstva te diskriminatorski stupac koji služi za određivanje tipa izvedene klase. *Table per class* hijerarhija je odabrana jer klase izvedene iz bazne klase nemaju dodatna svojstva nego služe za određivanje razine dozvola u aplikaciji. Ovom strategijom se omogućuje izvođenje polimorfnih upita koji su potrebni prilikom prijave u sustav. U klasi EmployeeMap su definirana mapiranja za zajednička svojstva ovih klasa (Id, FirstName, LastName, OIB, DateOfBirth, Username, Password, WaitingRoom), te definicija diskriminatorskog stupca. DoctorMap i NurseMap imaju definiranu vrijednost diskriminatorskog stupca, za liječnika “dr”, a za medicinsku sestru “nurse”.

#### EmployeeMap

Dodaj opis mapiranja

## Sloj poslovne logike



Slika: Business Layer

Dodaj opis za bussines layer

kako ste implementirali i organizirali repozitorije, opis mapiranja za ključne klase u domeni, kako ste organizirali transakcije i Unit of Work

# Opis izgrađene desktop aplikacije

uz prikaz 3-5 screenshotova formi koji realiziraju glavne use caseove potrebno je opisati kako ste realizirali MVP (MVVM) pattern i povezali svoje GUI sučelje s modelom domene

# Opis izgrađene web aplikacije

slično kao i za desktop aplikaciju, i ovdje je bitno opisati organizaciju modela, viewova i kontrolera u vašoj web aplikaciji i kako ste sve to povezali u cjelinu