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 u 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 liječniku. 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 liječnika, 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.

Liječnik 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 liječniku pripisan. Odabirom pacijenta s popisa iz čekaonice liječnik 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 zamjenjujući 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 nekog razreda
* 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

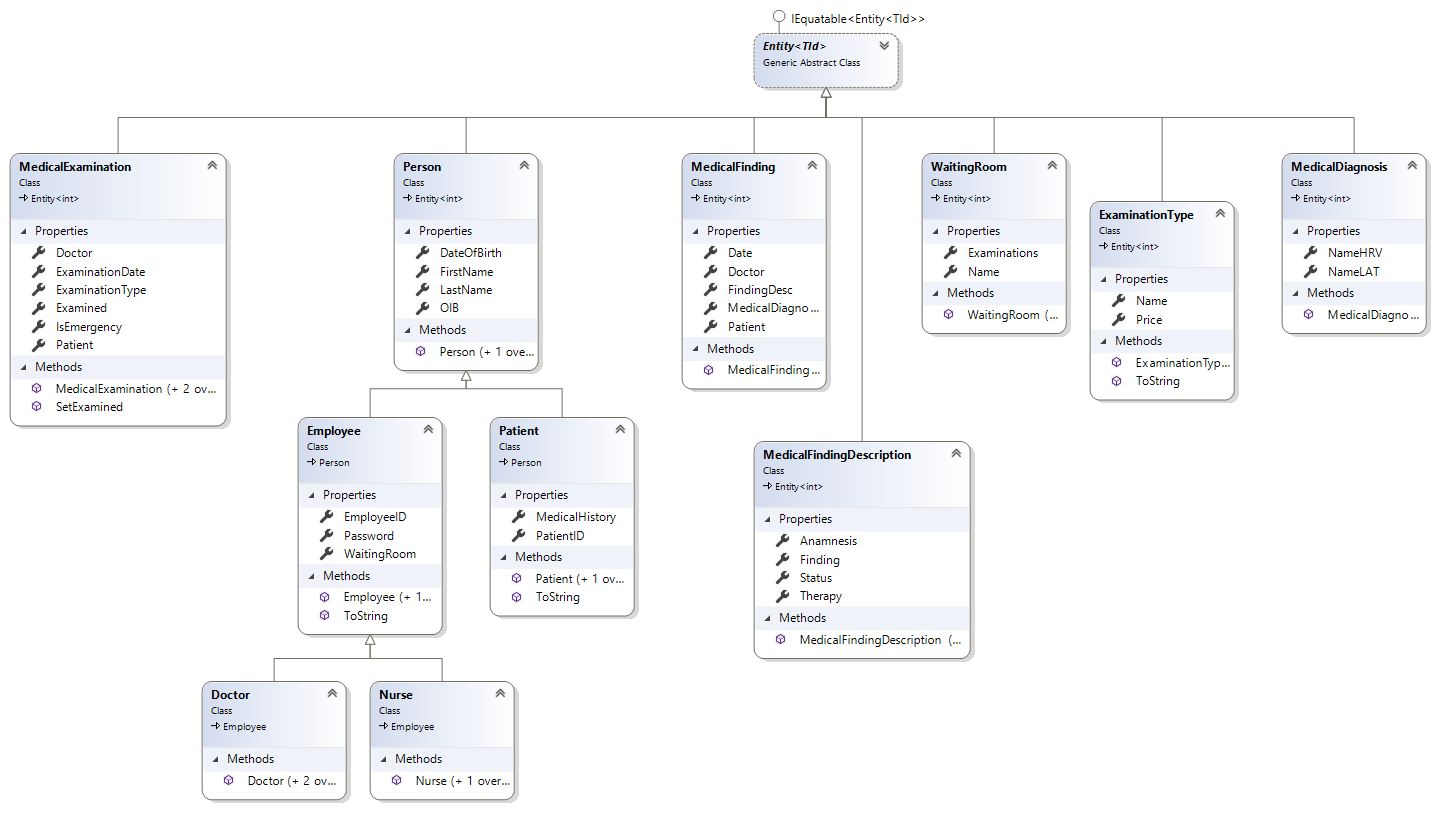
Osim njih, kreirani su specifični repozitoriji za svaki model koji omogućuju dodatne operacije.

* 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



Razred *Employee* predstavlja medicinsko osoblje, liječnike i medicinske sestre, koristi se prilikom prijave u sustav za utvrđivanje pristupa informacijama, a služi i kao spremnik osnovnih informacija o osoblju. Razred *Patient* predstavlja osobu koja koristi medicinsku ustanovu, te služi kao spremnik osnovnih informacija ali i medicinske povijesti, odnosno sadrži referencu sa pohranjenim prijašnjim pregledima. *MedicalExamination* 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.

Razred *MedicalFinding* služi za pohranjivanje informacija u medicinskom nalazu. 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 nagnalo 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.

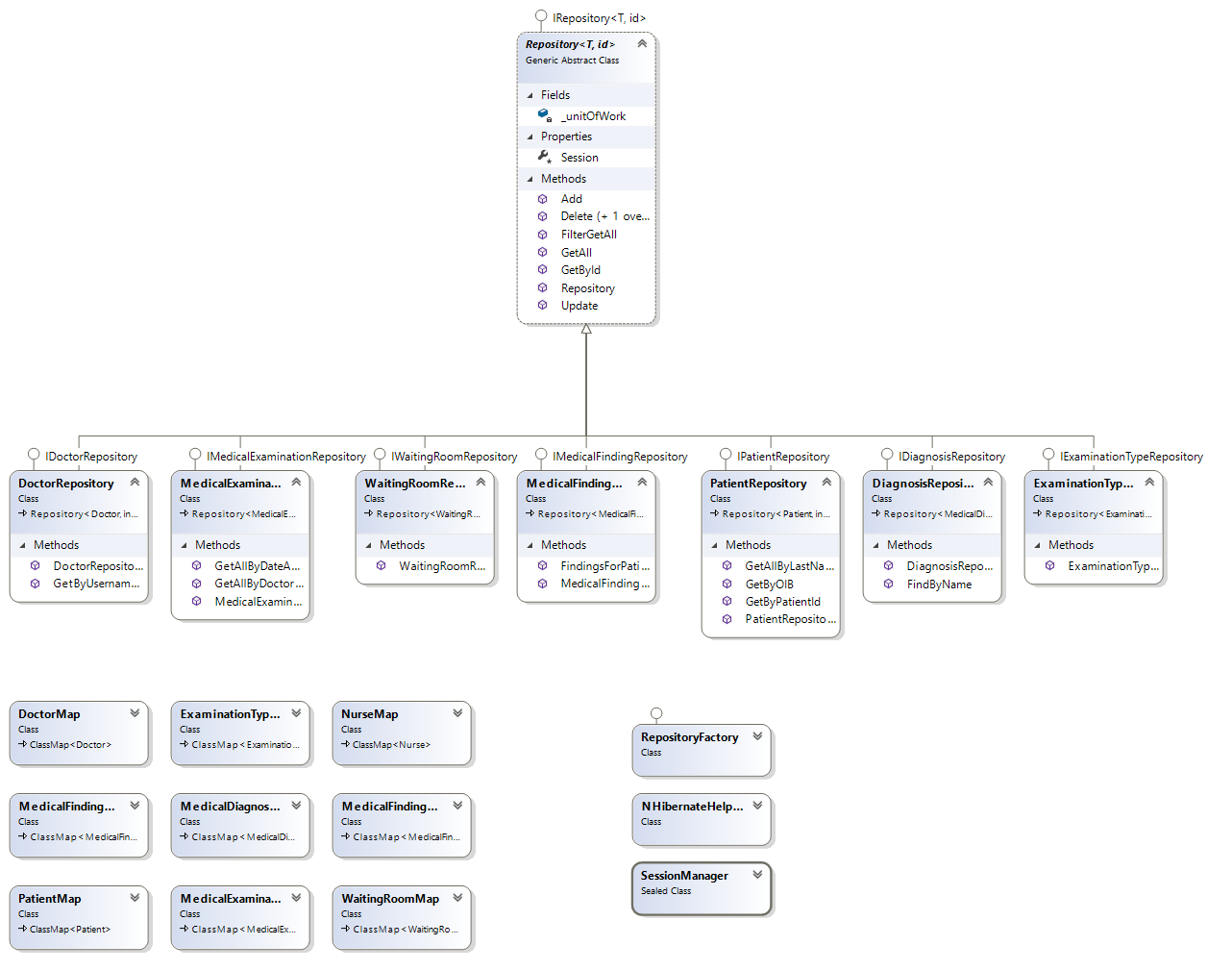
*ExaminationType* opisuje tipove pregleda te sadrži podatke o njihovim cijenama. Ovo se odnosi na hitan pregled, pregled u svrhu kontrole i slično. Pretežito se koristi za računanje statistike (koliko je sredstava liječnik osigurao), te kako bi pacijenti s naznakom “HITNO” mogli biti pregledani što prije.

Modeli u objektu su uglavnom Entity tipa, budući da je bitno razlikovati preglede, zaposlenike i pacijente. Value objekt je Statistics jer služi samo za modeliranje odnosno pohranu i združivanje atributa koji ga opisuju.

# Opis implementacije perzistencije

Ovaj 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 (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 ne zahtjevaju dodatna mijenjanja prepostalih repozitorija. Također, svaki repozitorij služi za dohvaćanje objekata isključivo jednog razreda kako bi se osiguralo načelo jedinstvene odgovornosti.

Razred *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 razreda *PatientMap* koji nasljeđuje *ClassMap*. Pojedina svojstva su mapirana naredbom Map, a povijest bolesti pacijenta naredbom *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*. Rezultat ovakve organizacije su bolje performance, odnosno izbjegavanje naredbe join pri dohvaćanju adrese pacijenta.

#### EmployeeMap, DoctorMap, NurseMap

Za mapiranje hijerarhije nasljeđivanja se koristi *table per class hierarchy.* Razredi *Employee*, *Doctor* i *Nurse* su mapirani 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 razredi izvedene iz baznog razreda 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 razredu *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”.

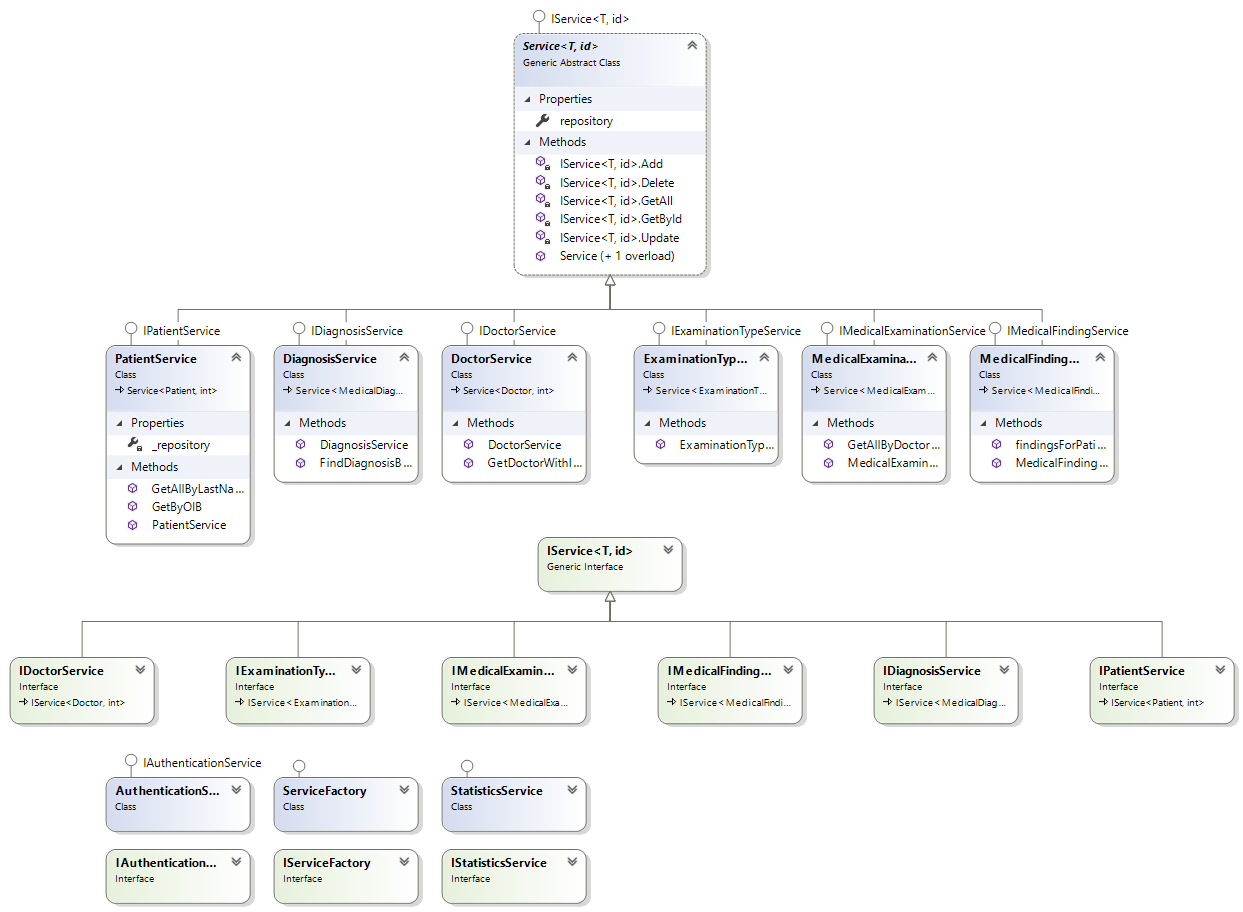
#### MedicalFindingMap

Pri mapiranju *MedicalFinding* korištene su Map funkcije za datum, anamnezu, opis nalaza, status i terapiju. Budući da se dijagnoza može samo odabrati iz već postojećeg skupa koji sadrži njeno hrvatsko i latinsko ime, korištena je funkcija References uz dodatke Not.LazyLoad. NHibernate ima standard koji podupire LazyLoading odnosno učitavanje objekata povezanih referencom tek kad oni zatrebaju. Budući da je medicinska dijagnoza prijeko potrebna za pregled nalaza i medicinske povijesti, ovdje se Lazy Load ne koristi. Također, reference su dodane na svojstva Pacijent i Doktor .

#### MedicalExaminationMap

Pri mapiranju medicinskog pregleda koristila se Map funkcija za atribute koji označavaju datum i zastavicu *Examined* koja identificira je li pregled izvršen. Reference se koriste pri mapiranju čekaonice, doktora, pacijenta i tipa pregleda.

## Sloj poslovne logike



Slika: Business Layer

Sloj poslovne logike, po uzoru na repozitorije, sadrži parametrizirani razred *Service* kojeg nasljeđuju svi ostali. Taj se razred bavi implementacijom već navedenih, osnovnih 6 metoda. Uz to, svaki od razreda sadrži dodatne metode koje bi mogle zatrebati posebno za taj objekt.

Primjerice, *DoctorService* sadrži dodatne metode poput *GetByEmployeeId* – omogućuje dohvaćanje zaposlenika temeljem njegovog atributa koji predstavlja korisničko ime. Ova se metoda često koristi budući da je autentifikacija napravljena na način da u autentifikacijski kolačić sprema korisničko ime. Korisničko ime je tako dostupno prilikom svakog zahtjeva, a djelatnik kojeg to ime karakterizira dohvaća se navedenom funkcijom. Funkcija *GetDoctorWithIdAndPassword* služi prilikom prijave za provjeru identiteta.

*MedicalExaminationService* dodatno definira i implementira *GetAllByDoctorAndNonExamined* i *GetAllByWaitingRoom* funkcije. *GetAllByDoctorAndNonExamined* kao parameter prima id liječnika te pronalazi sve preglede koji još nisu obavljeni a pripadaju tom liječniku, *GetAllByWaitingRoom* dohvaća sve preglede za odgovarajuću čekaonicu.

*MedicalFindingService* proširuje osnovni skup metoda s metodom *FindingsForPatient*. Ova metoda kao argument prima pacijenta te dohvaća njegovu medicinsku povijest odnosno sve njegove medicinske nalaze.

Navedeni servisi metode najčešće delegiraju repozitorijima koji zatim izvode odgovarajuće upite. Cilj toga je da se već pri upitu prema bazi podataka jasno (koliko je moguće) specificiraju svojstva koja objekti moraju imati, posebice jer se radi o najčešće pozivanim metodama. Primjerice, *GetAllByDoctorAndNonExamined* u *MedicalExanimation* servisu vraća preglede koji zastavicu *Examined* imaju postavljenu na *False*, kako bi se smanjilo opterećenje na sustav i prenošenje nepotrebne količine podataka (npr. dohvaćanje svih pregleda i izdvajanje onih koji imaju zastavicu postavljenu na *False*) te je ta logika izdvojena u upit prema bazi.

*StatisticsService* računa statistiku metodom *CalculateStatistics*. Pomoću repozitorija dohvaća sve izvedene preglede navedenog liječnika na odgovarajući datum te temeljem te liste računa statistiku. Statistika obuhvaća listu predmeta, ukupnu svotu zarađenih sredstava u kunama te broj pacijenata koji su pregledani taj dan.

Svaki od servisa napravljen je kao implementacija sučelja. Sučelja, kao i navedeni razredi, nasljeđuju ono glavno parametrizirano Service sučelje.

Dodatno, *AuthenticationService* sadrži logiku vezanu za autentifikaciju koja se nalazi u metodi *Authenticate gdje pozivima u odgovarajuće repozitorije provjerava postoji li zaposlenik s unesenim korisničkim podatcima te ga posprema kako bi se mogao dohvatiti iz bilo kojeg servisa*.

# Opis izgrađene desktop aplikacije

Za realizaciju desktop aplikacije je korištena MVC arhitektura. Za prikaz view-ova su korišteni Windows Forms-i, a svakom formom upravlja odgovarajući kontroler. Modeli korišteni u sloju domene su prethodno opisani u poglavlju 3. Korisnik preko formi unosi naredbe, koje kontroler preusmjeruje u sloj poslovne logike gdje se obavljaju odgovarajući postupci te nakon izvršavanja kontroler o tome obavještava korisnika pomoću formi.

## Kontroleri

Aplikacija se sastoji od nekoliko upravljača koji povezuju GUI sučelje s aplikacijskom logikom.

### MainFormController

Upravlja prijavom i odjavom zaposlenika u sustav. Sadrži funkcije *CheckAuthentication,* *Logout* i *ShowWaitingRoom.* *CheckAuthentication* poziva servis koji služi za provjeru autorizacije zaposlenika, *Logout* odjavljuje zaposlenika, a *ShowWaitingRoom* stvara formu za prikaz čekaonice te odgovarajuću kontroler koji njome upravlja.

### WaitingRoomController

Upravlja formom za prikaz čekaonice te služi za navigaciju u ostale dijelove aplikacije. Osnovne funkcije koje sadrži su *AddExamination, Examine, ShowStatistics, ShowHistory, GetUpdatedExaminations. AddExamination* stvara formu za unos novog pregleda te upravljač koji njome upravlja. *Examine* se poziva nakon odabira pregleda iz čekaonice te stvara formu za obavljanje pregleda. *ShowStatistics* stvara formu za prikaz statistike doktora. *ShowHistory* stvara formu za prikaz povijesti bolesti odabranog pacijenta. *GetUpdatedExaminations* ažurira listu pregleda u čekaonici.

### MedicalExaminationController

Upravlja formom za upis novog pregleda. Sadrži metode *AddNewMedicalExamination, ShowSelectPatient* i *UpdateInfo. AddNewMedicalExamination* korištenjem odgovarajućih servisa stvara novi pregled te ga dodaje u čekaonicu. *ShowSelectPatient* otvara formu za pretragu pacijenata, a *UpdateInfo* ažurira podatke na formi ovisno o odabranom pacijentu.

### PatientController

PatientController upravlja formom za odabir pacijenta te sadrži metode *SearchPatients* i *UpdateAddMedicalExamination* koje služe za pretragu po prezimenu te odabir pacijenta.

### MedicalFindingFormController

MedicalFindingFormController upravlja formom za obavljanje pregleda te sadrži metode *AddMedicalFinding* i *SaveFinding* koje služe za pospremanje nalaza nakon što je pregled proveden. Uz njih sadrži i metodu *ChooseDiagnosis* koja otvara formu za odabir završne dijagnoze.

### MedicalDiagnosisChoiceController

Upravlja formom za odabir završne dijagnoze te sadrži metode za pretragu dijagnoza po imenu.

### StatisticsController

Upravlja formom koja prikazuje statistiku liječnika.

## Prijava u sustav

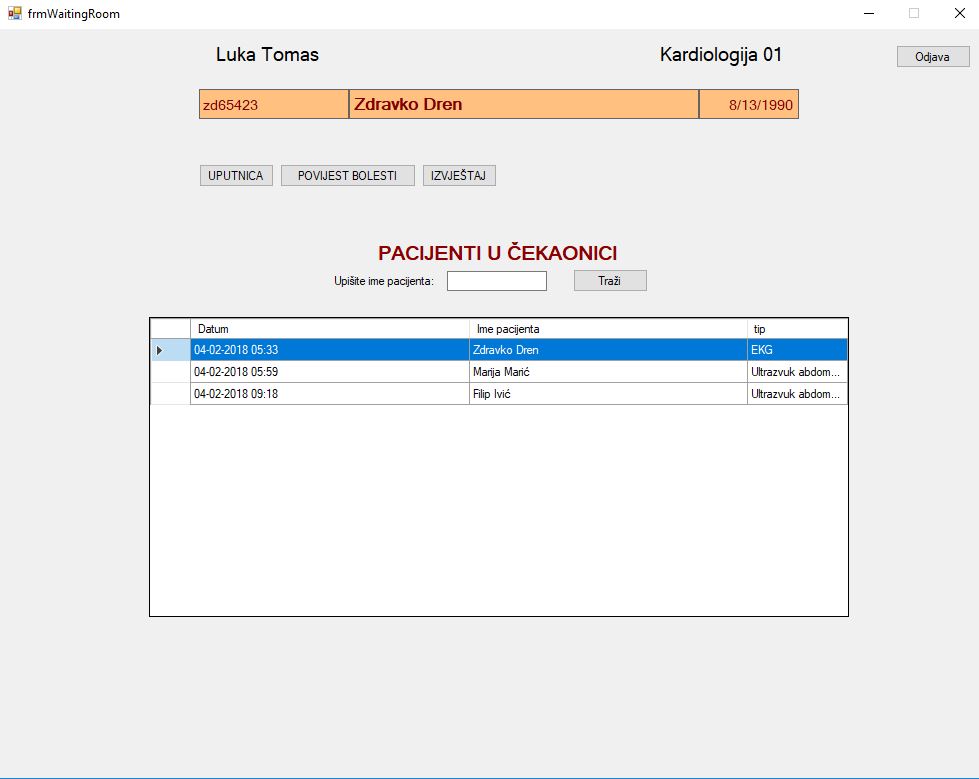
Prilikom pokretanja aplikacije prikazuje se početni zaslon preko kojeg se zaposlenik prijavljuje u sustav. Za izgled je zadužena Windows Forms klasa frmMainWindow, a s modelom domene je povezana s MainFormController-om. Nakon pritiska na gumb ''Prijavi me'' poziva se naredba *CheckAuthentication* unutar kontrolera koja poziva odgovarajuću naredbe iz sloja poslovne logike te ovisno o rezultatu upita nastavlja. Ukoliko je prijava neuspješna, odnosno zaposlenik s upisanim korisničkim podatcima ne postoji, zaposlenik je obavješten o neuspješnoj prijavi odgovarajućom porukom. Ako je prijava uspješna poziva se naredba *ShowWaitingRoom* unutar koje se stvara nova forma za prikaz čekaonice(pogledati 5.2) te odgovarajući kontroler koji njome upravlja.



Slika: Prijava u sustav

## Čekaonica

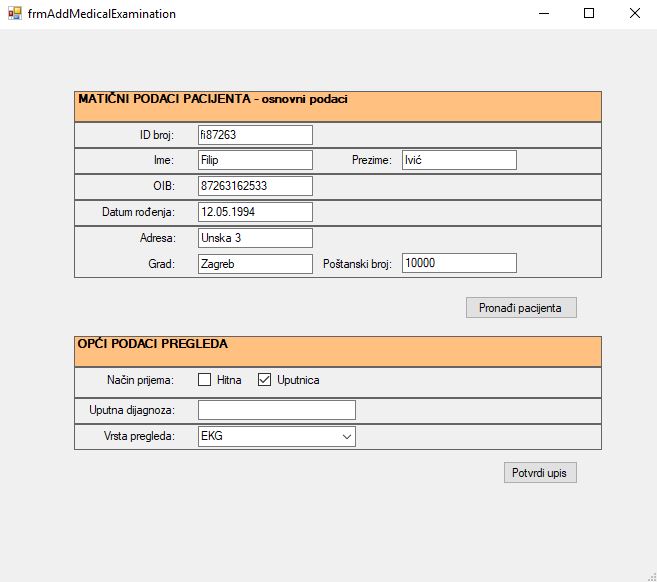
Nakon uspješne prijave prikazuje se forma frmWaitingRoom prikazana na slici. Sastoji se od prikaza liste upisanih pregleda koji čekaju svoj red, te gumbova kojima se navigira u ostale dijelove aplikacije. Korisničkim akcijama upravlja WaitingRoomController. Pritiskom na pregled unutar liste poziva se funkcija *Examine* u kontroleru koja stvara formu za obavljanje pregleda. Klikom na gumb ''UPUTNICA'' otvara se forma za upis novog pregleda, klikom na gumb ''POVIJEST BOLESTI'' otvara se forma za pregled prethodnih nalaza odabranog pacijenta, klikom na ''IZVJEŠTAJ'' prikazuje se forma sa statistikom liječnika prijavljenog u sustav.



Slika: Čekaonica

## Upis novog pregleda

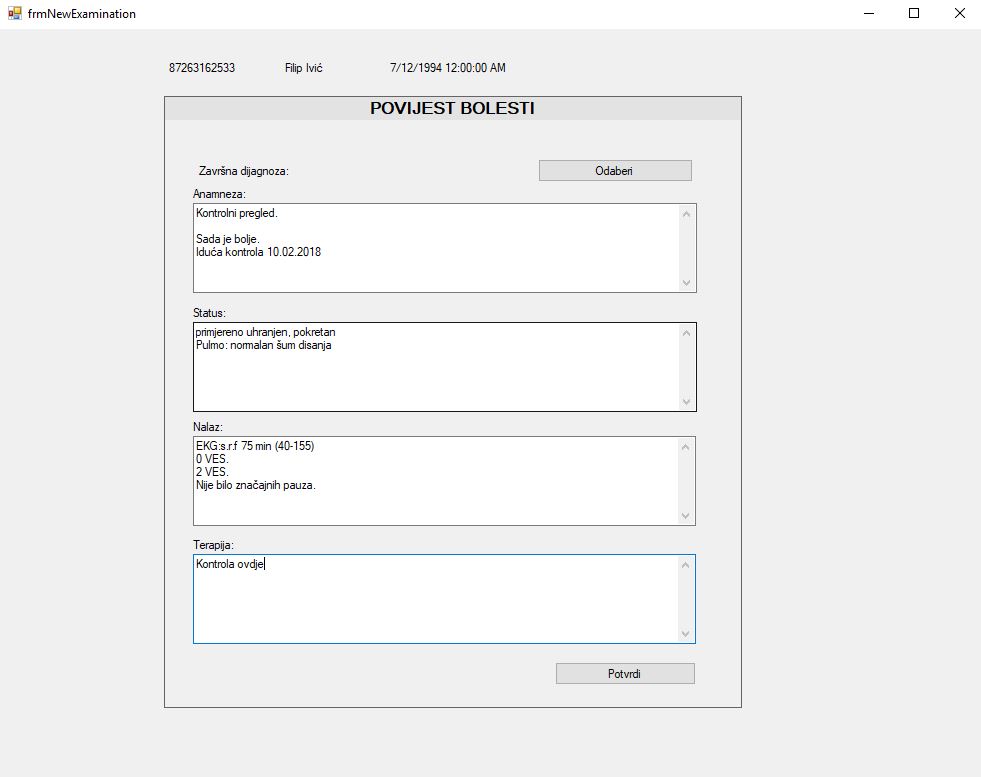
Nakon odabira unosa pregleda u čekaonici otvara se forma prikazana na slici ispod. Korisnik unosi podatke o pacijentu te tip pregleda i način prijema. Ukoliko je pacijent već bio na pregledu njegovi podatci se ne moraju iznova unositi već se klikom na gumb ''Pronađi pacijenta'' otvara forma u kojoj se pacijenti pretražuju po prezimenu, te korisnik može odabrati pacijenta čime se njegovi podatci u formi automatski ispunjavaju. Ako uneseni podatci nisu ispravni korisnik je o tome obavješten odgovarajućom porukom. Nakon potvrde unosa pregleda on se dodaje u čekaonicu te se forma zatvara. Ažuriranje liste pregleda u čekaonici je ostvareno obrascem Observer. View frmWaitingRoom je observer kojeg repozitorij obavještava o dodavanju novog pregleda. frmWaitingRoom zatražuje od kontrolera koji njime upravlja da dohvati novu listu pregleda.



Slika: Unos novog pregleda

## Nalaz

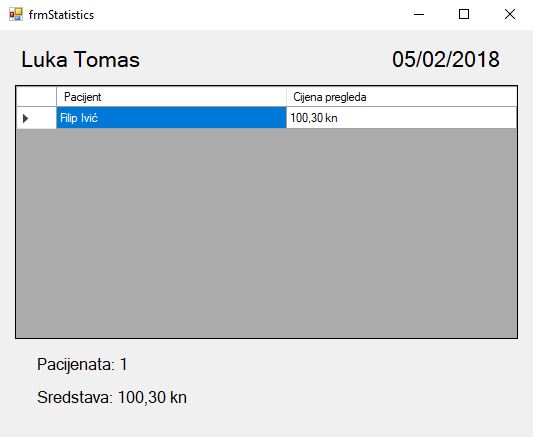
Nakon odabira pacijenta iz čekaonice otvara se forma za obavljanje pregleda i pisanje nalaza. Ovu formu može otvoriti samo liječnik, a medicinska sestra ne može. Liječnik upisuje podatke o pregledu te po završetku odabire završnu dijagnozu iz liste pohranjenih dijagnoza. Nakon što je ispunio nalaz svoj pregled potvrđuje klikom na gumb ''Potvrdi'' te se nalaz pohranjuje u bazu podataka, a pregled se označava kao pregledan i briše iz čekaonice uporabom Observer obrasca.



Slika: Nalaz

## Statistika

Prikazuje statistiku liječnika prijavljenog u sustav.



Slika: Statistika

# Opis izgrađene web aplikacije

U web aplikaciji korištene su reference na logiku poslovnog sloja kako bi omogućili učinkovitu komunikaciju s bazom podataka, također, korišteni su jednaki modeli (opisani u poglavlju 3.).

## Upravljači

Web aplikacija sastoji se od nekoliko upravljača ključnih za njeno pravilno funkcioniranje.

To su:

* BaseController

Ovaj je upravljač korišten u svrhu postavljanja *UnitOfWork* radnog okvira.

* LoginController

*LoginController* obavlja logiku vezanu za prijavu korisnika. Sastoji se od *Login* i *Logout* funkcije.

*Login* funkcija provjerava upisano korisničko ime i lozinku, te ovisno o tome stvara autentifikacijski kolačić (*engl. authentication cookie*) koji sadrži korisničko ime koje se koristi u daljnjim u akcijama i operacijama.

Funkcija Logout uništava autentifikaciju kartu (*engl. authentication ticket*) iz autentifikacijskog kolačića. Autentifikacijska karta je u svom najosnovnijem obliku vrijednost autentifikacijskog kolačića koji identificira korisnika.

* MedicalExaminationController

Ovaj je upravljač jedan od najopsežnijih u sustavu te obavlja većinu operacija koje sustav zahtjeva. Sastoji se od nekoliko akcija: *Index, Details, Create i Statistics*.

*Index* služi za prikazivanje liste najavljenih medicinskih pregleda. Preglede dohvaća pomoću odgovarajućeg servisa, a prikazuje pacijente liječnka koji je ulogiran u sustav. Time se postiže da liječnik ne može vidjeti čekaonice (popise zakazanih pregleda) drugih liječnika, kao ni obavljati preglede u njihovo ime.

*Create* se bavi zakazivanjem odnosno stvaranjem pregleda. Liječnik može ugovarati preglede samo sebi, ne i drugim liječnicima. Pregled se zakazuje odabirom datuma, pacijenta te tipa pregleda (hitan, redovan, sistematski). Liječnik ne upisuje ime i prezime pacijenta već njegov OIB kojeg sustav zatim provjerava te ukoliko je validan zapisuje pregled.

*Statistics* se bavi računanjem statistike za odgovarajućeg liječnika na specificiran datum. Liječnik ne može vidjeti statistike drugih liječnika.

* MedicalFindingsController

Ovaj se upravljač sastoji od nekoliko operacija: *Index, Details i Create.*

*Index* se bavi prikazivanjem liste medicinskih za određenog pacijenta (pacijent za kojeg mu je u argumentu prosljeđen OIB). Medicinska povijest sadrži sve nalaze – neovisno od liječnika koji su ih kreirali.

*Details* prikazuje detaljnije medicinske nalaze omogućujući liječniku da provjeri sve opaske o pacijentu zapisane ranije od strane različitih liječnka.

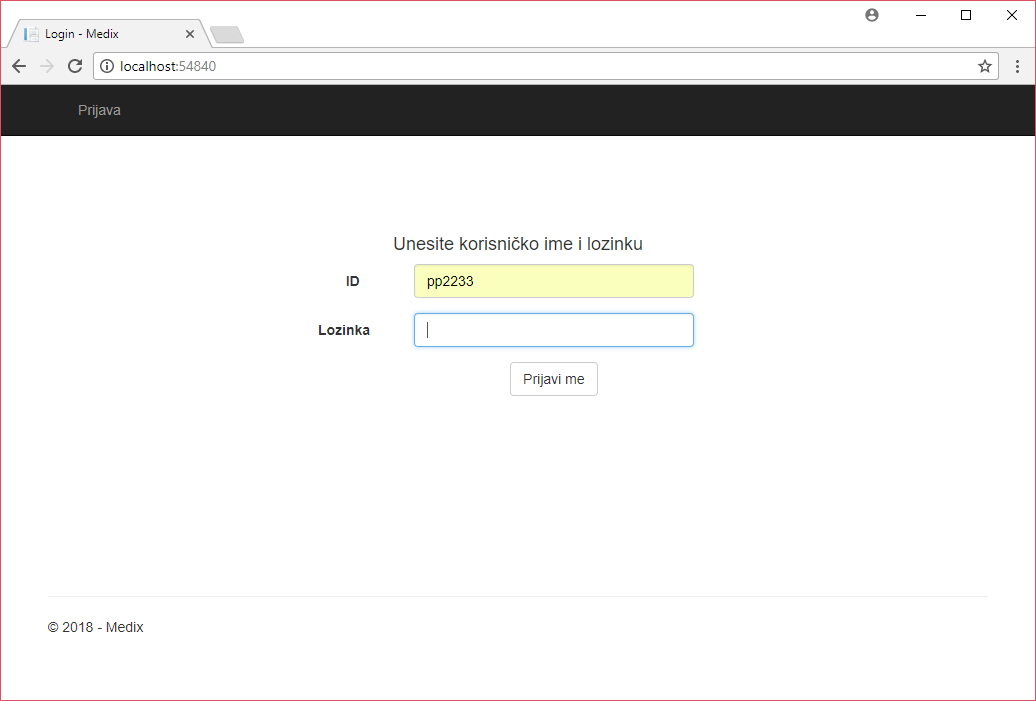
*Create –* najbitniji dio ovog upravljača, stvara nov medicinski nalaz. Liječnik upisuje terapiju, status, anamnezu i zapažanje – nalaz, te odabire dijagnozu iz liste postojećih dijagnoza. Stvaranjem zapisa upravljač označava pacijenta pregledanim te se on više ne pokazuje u čekaonici, odnosno prikazuje se u statistici.

Niti jedan od navedenih upravljača ne sadrži *Delete* i *Update* akcije. Razlog tome je što se medicinski nalazi u sustavu striktno ne smiju brisati i mijenjati. U slučaju promjene mogu se unjeti novi.

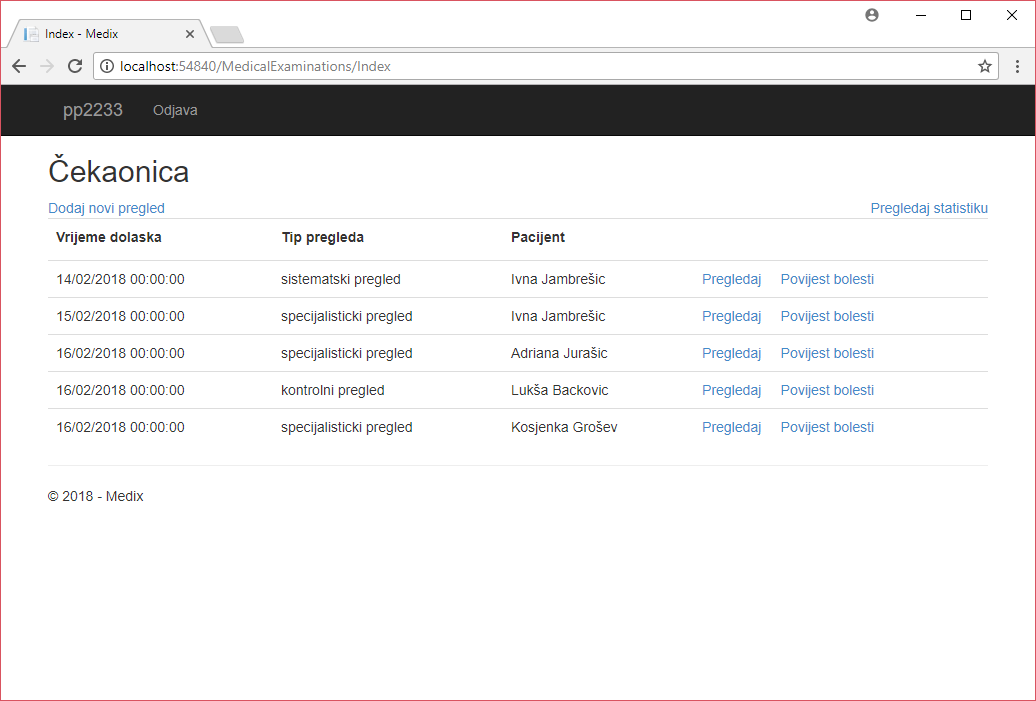
Svaki upravljač manipulira podatcima pomoću servisa, te obrađene podatke prenosi idućem sloju – sloju pogleda.

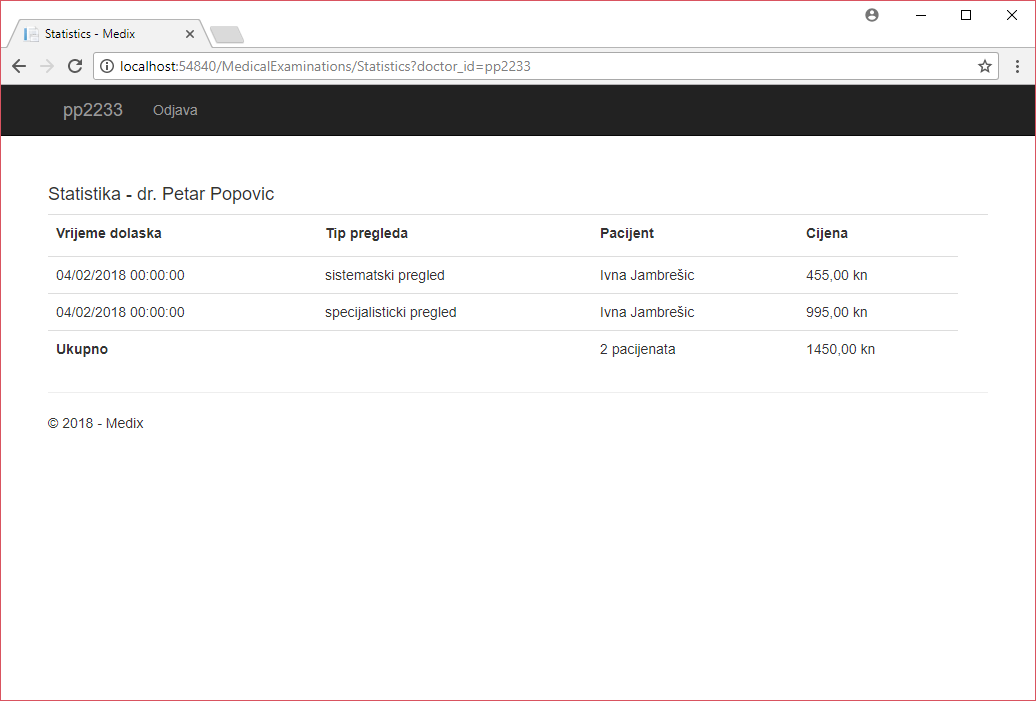
## Pogledi

### LoginView - Prijava u sustav



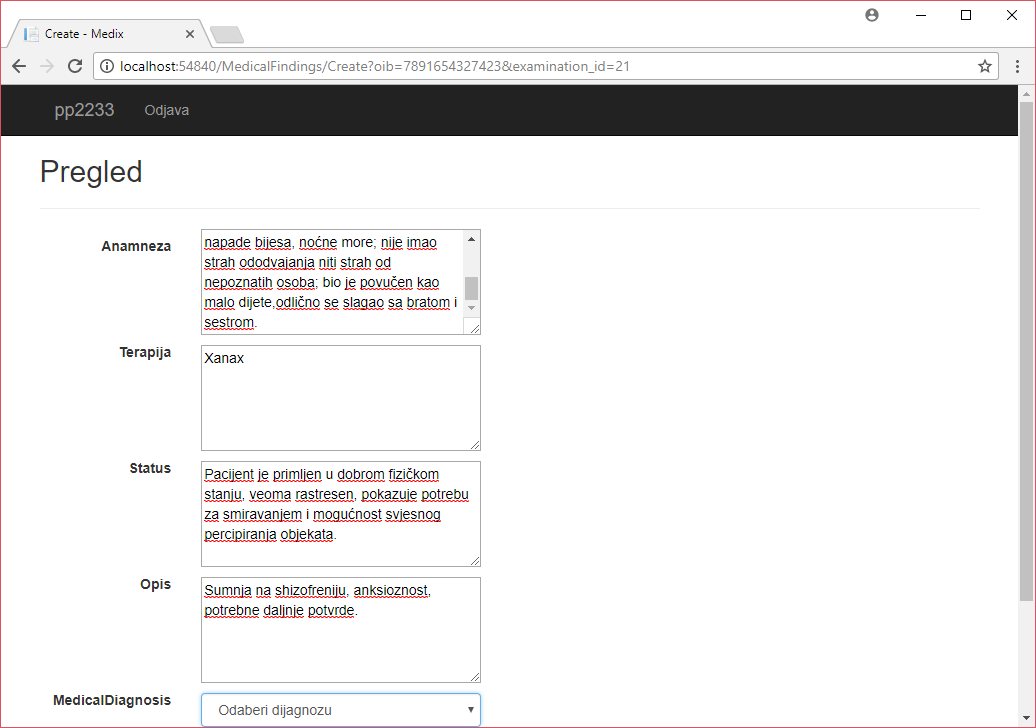
### MedicalExaminationView - čekaonica (popis zakazanih pregleda)

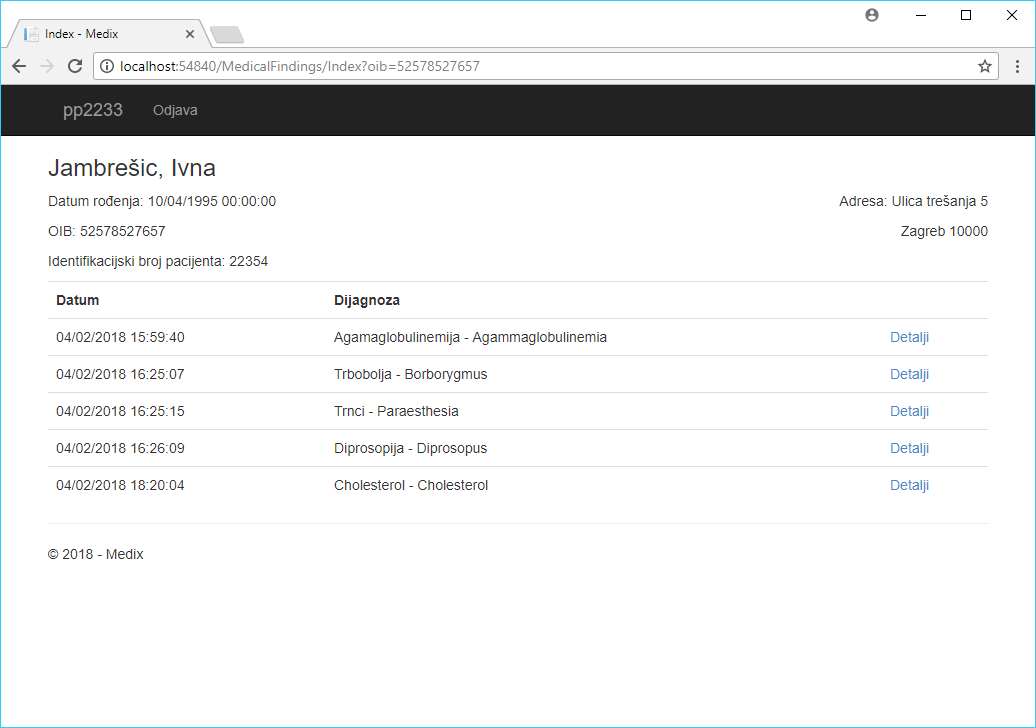


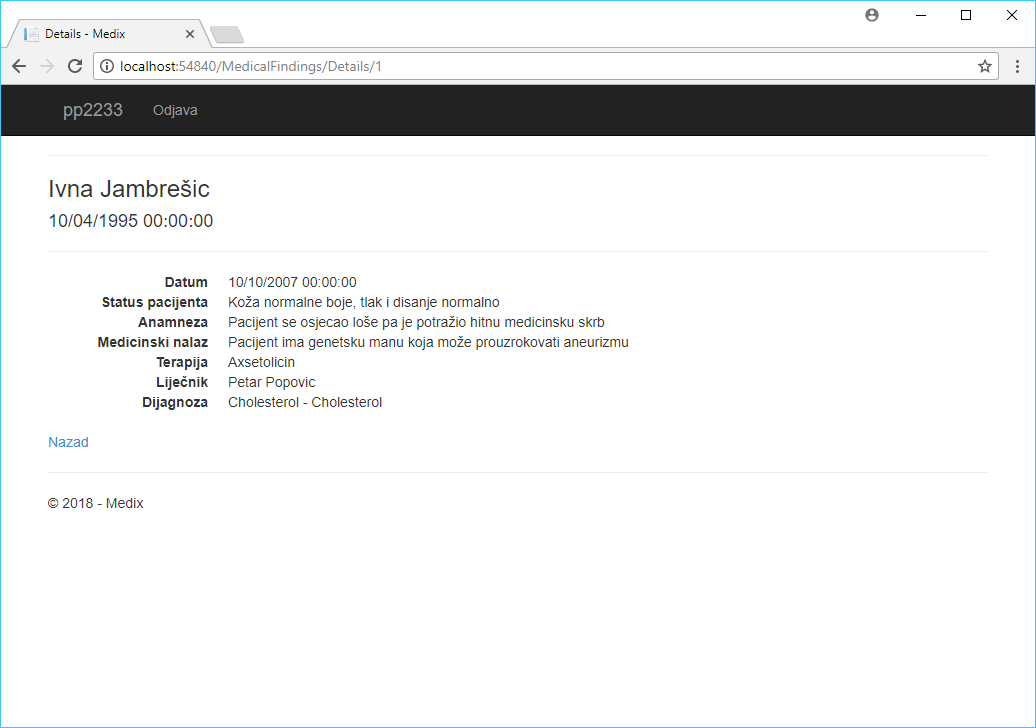
Ovo je centralni pogled nakon prijave u sustav. Prikazuje samo zakazane preglede liječnika koji je u sustav prijavljen. Klikom na “Dodaj novi pregled” otvara se novi pogled za zakazivanje pregleda. Klikom na “Pogledaj statistiku” liječnik može vidjeti statistiku svojeg rada za taj dan. Svakog pacijenta može pregledati klikom na “Pregledaj”. Pregled pacijenta podrazumjeva stvaranje novog medicinskog nalaza. “Povijest bolesti” prikazuje listu dosadašnjih medicinskih nalaza. 

Statistika prikazuje popis obavljenih pregleda (i pregledanih pacijenata) te sredstva osigurana tim radom.

### MedicalFindingsView



Klikom na potvrdi se nalaz pohranjuje. Osim toga liječnik može pregledati i medicinsku povijest pacijenta: 



Svi pogledi dijele neke zajedničke dijelove (kao što se na prethodnim slikama može uočiti). To su zaglavlje koje sadrži identifikacijsko ime liječnika (klikom na njega se liječnik vraća u čekaonicu) i gumb za odjavu iz sustava.