GRUPA 2

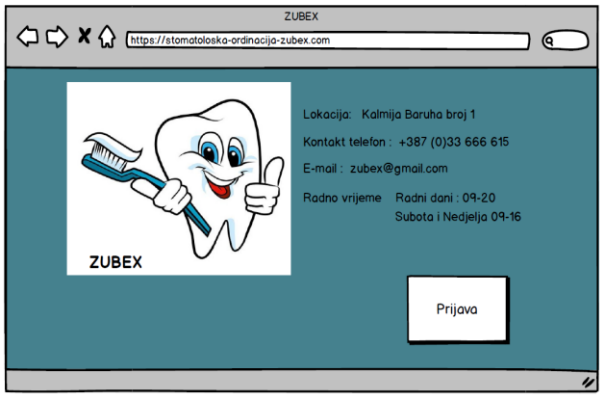
Tema: Stomatološka ordinacija

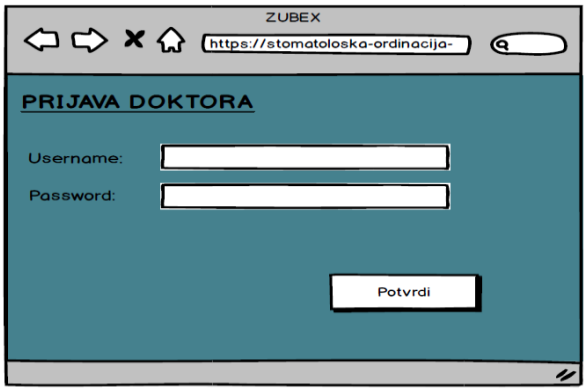
Članovi tima:

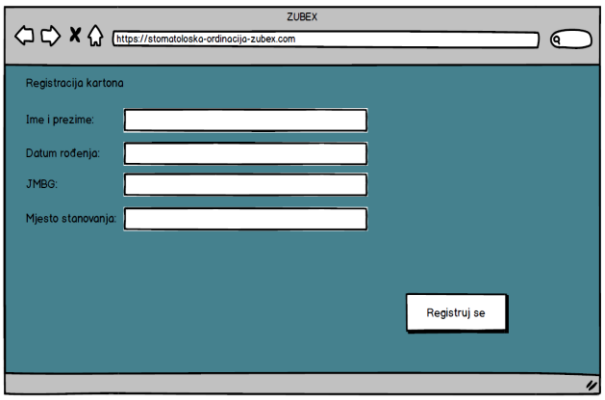
- Ema Hondo
- Tarik Osmanagić
- Sumeja Selmanović

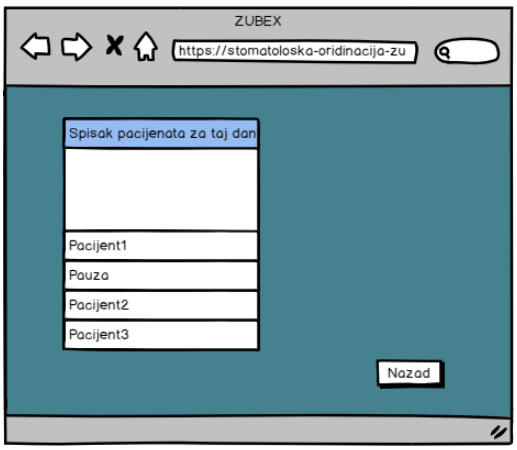
UI PROTORIP

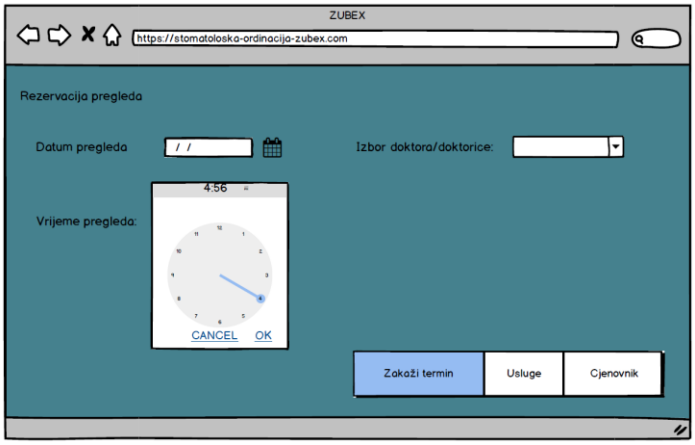
U par slika ćemo prikazati kako bi otprilike neke stvari na našoj aplikacija trebale izgledati, ostatak se nalazi u folderu UIPrototip na našem git profilu.

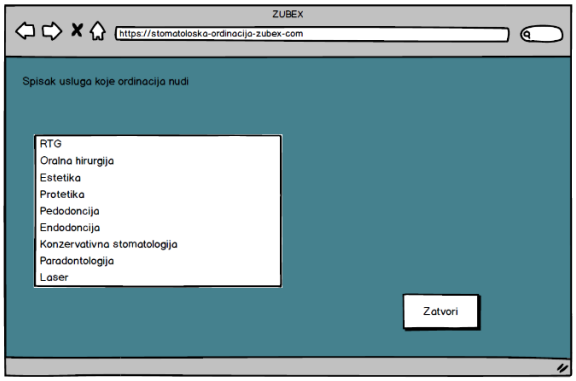


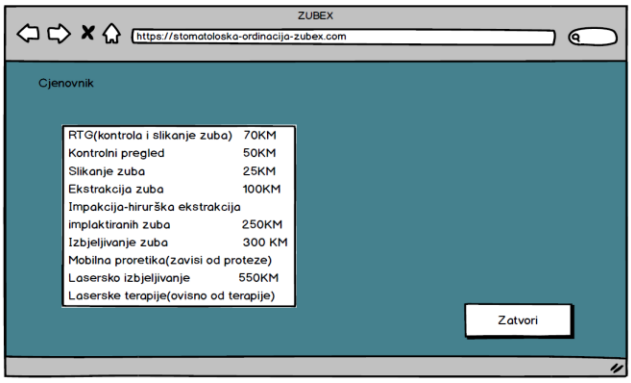




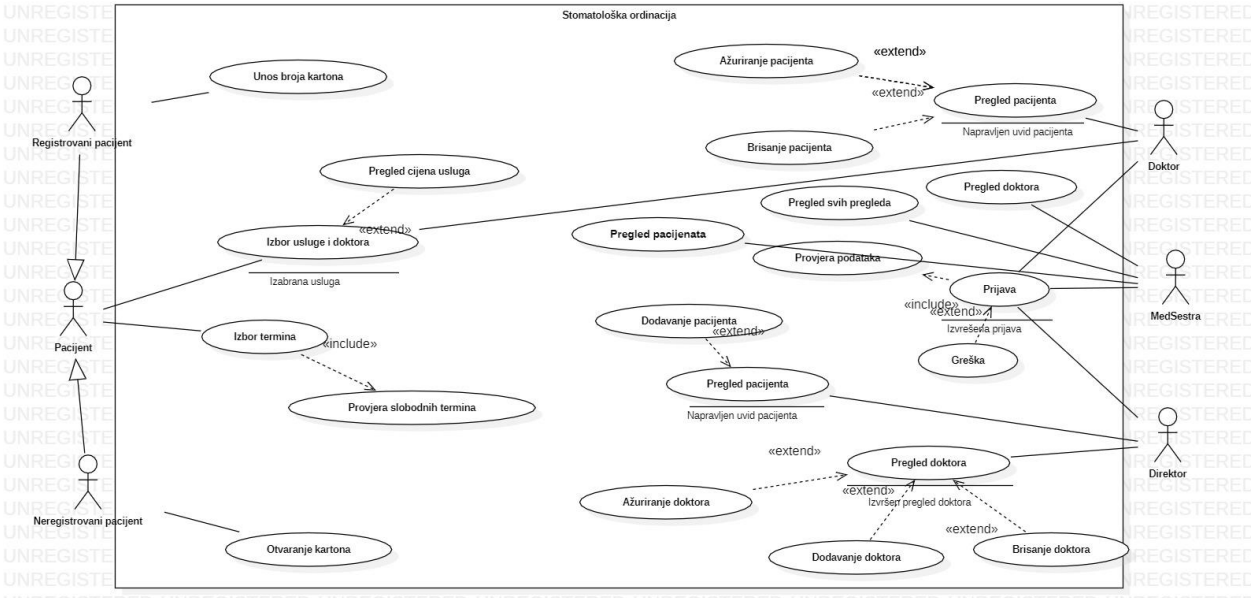








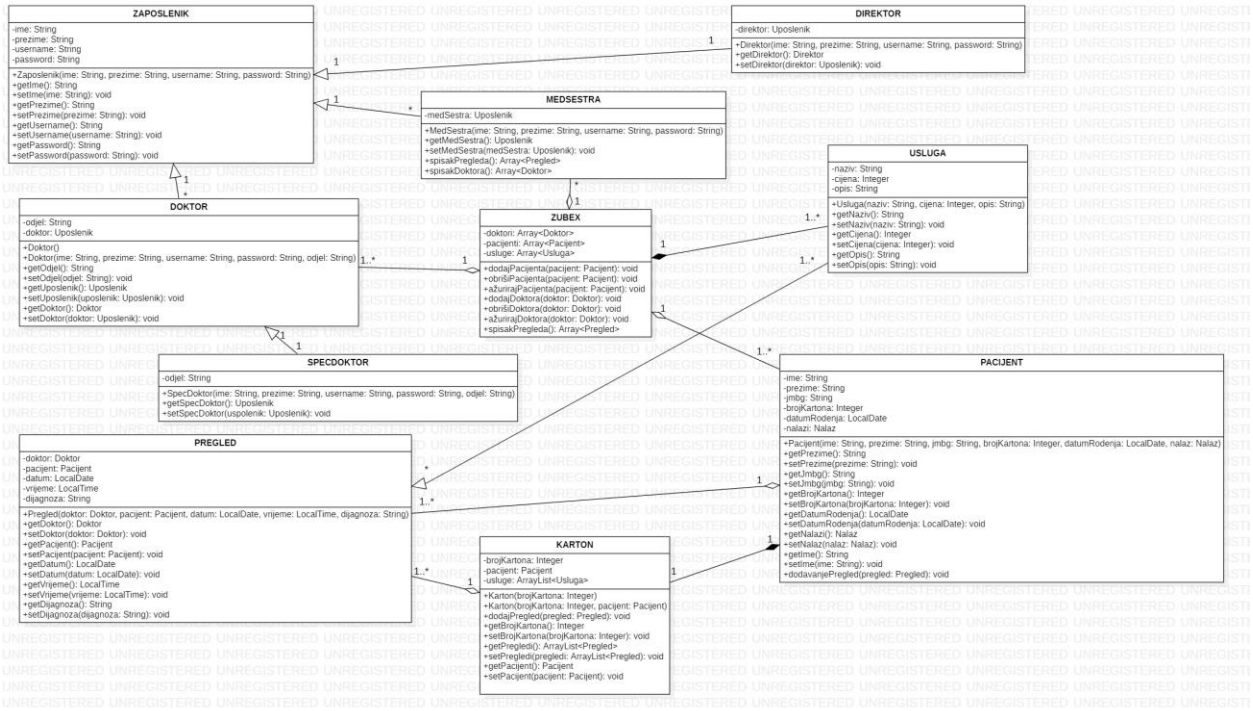
USECASE DIJAGRAM



Slučajevi upotrebe predstavljaju tehniku modeliranja koja se koristi za opisivanje šta bi to novi sistem trebao da radi ili šta postojeći sistem radi, sa spekta interakcije između sistema i korisnika.Akteri su Pacijent(Registrovani i Neregistrovani), Medicinska Sestra, Doktor i Direktor.Prvi akter je pacijent čija ja funkcionalnost izbor termina koji su slobodni.Nakon što pacijent izabere termin, vrši se funkcionalnost biranja doktora i usluge, što također može uključivati i pregld cijena usluga koje naša ordinacija Zubex pruža.Naš pacijent je rastavljen na dvije veće podkategorije koje se razlikuju u prvoj uslovnoj funkcionalnosti, a to je uslov posjedovanja kartona.Prvi tip pacijenta, registrovani pacijent, ima funkcionlanost unosa broja svog kartona, dok drugi tip, neregistrovani pacijent vrši funkcionalnost otvaranja svog kartona, prije bilo kakve interakcije sa našim sisestemom.Drugi akteri su doktor, direktor i medicinska sestra koji prvo vrše prijavu i provjerava se da li su uneseni podaci tačni, ukoliko se pojavljuje upozorenja da ulazni podaci nisu odgovarajućeg formata vrši se ponovna prijava.Nakon uspjesne prijave doktor ima mogućnost mogućnost pregleda svih pacijenata i nad njima može vršiti ažuriranja i brisanja.Nakon uspješe prijave medicinske sestre ona ima prisup svim pregledima i pacijentima kao i spisku svih doktora.Nakon što se uspješne prijave direktora on ima mogućnost pregleda svih pacijenata i doktora, a nad doktorima može vršiti ažuriranje, brisanje i dodavanje novih doktora.

CLASS DIJAGRAM

Dijagram klasa opisuje tipove objekata u sistemu i raličite vrste statičkih veza koje postoje među njima.Ovi dijagrami također pokazuju svojstva i operacije klasa, kao i razne načine povezivanja objekata.

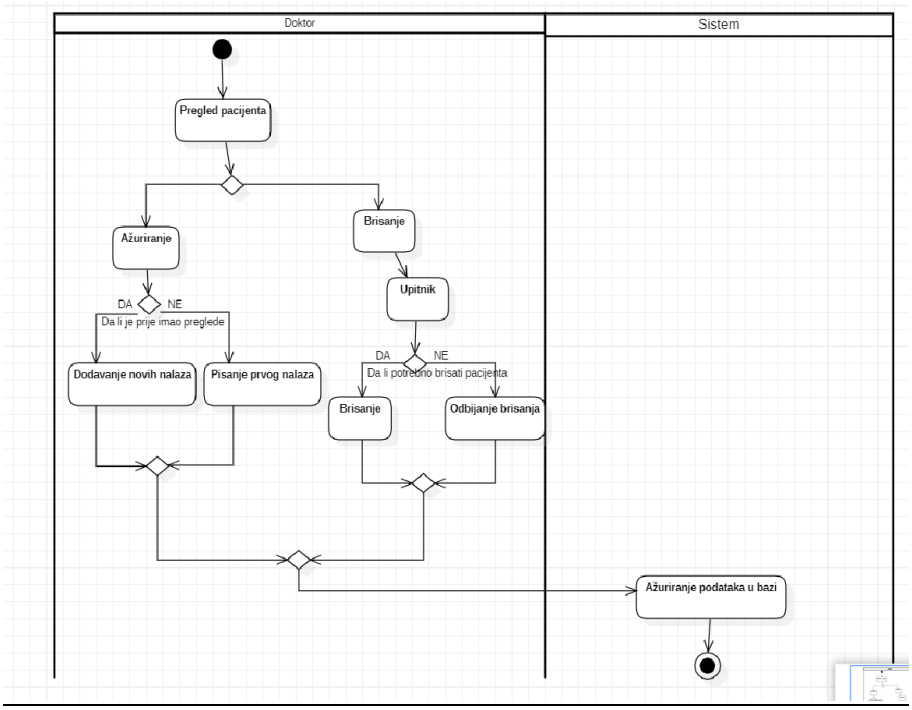


Svaka oridinacija ima svoje zaposlenike, tako da klasu Zaposlenik nasljeđuje više klasa u našem dijagramu. Doktor, MedSestra i Direktor su generalizacije klase Zaposlenik. Za svaki slučaj smo ovo

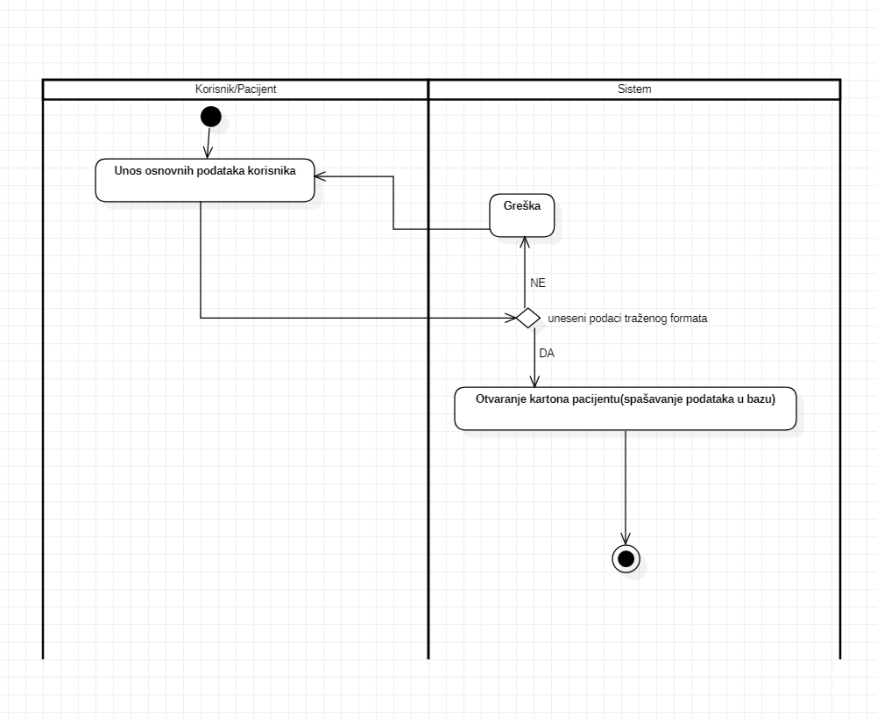
uradili na ovaj način, ako u budućnosti bude bilo potrebe za dodavanjem nekog uposlenika različitog od postojećih. SpecDoktor je generalizacija od klase Doktor, zato što postoji više vrsta specijalista u stomatologiji, kao npr. Specijalista dječije stomatologije, obiteljska stomatologija, oralna hirurgijja, ortodoncija itd. Klasa Direktor služi kao neka vrsta admina koja nadgleda sistem i ima mogućnost upravljanja samom oridinacijom. Okosnicu našeg dijagrama klasa čine klase Zubex i Pacijent. Zubex u našem dijagramu predstavlja oridinaciju. Klasa Zubex je povezana sa više klasa, spojena je agregatnom vezom sa klasama Dokror, MedSestra, Direktor i Pacijent. Agregacija kao vrsta asocijacije se koristi da se indicira da klasa pored vlastitih atributa, može uključivati, u zavisnosti od kardinalnosti, određen broj instanci drugih klasa. Zubex se sastoji od jednog ili više doktora, od većeg broja medicinskih sestara, od jednog ili više pacijenata. Ove klase koriste i druge klase, pa nisu isključivo vezane za klasu Zubex. Možda je najbolji primjer iz predavanja između tima i osobe. Broj osoba se može smanjivati ili povećavati ali će cjelina tim i dalje postojati. U našem slučaju broj doktora, direktora, medicinskih sestara i pacijenata se može smanjivati ili povećavati, ali će cjelina oridinacija postojati. Vezom kompozicije Zubex je spojena sa klasom Usluga, jer u asocijaciji kompozicije, cjelina strogo posjeduje svoje dijelove, ako se cjelina kopira ili briše, njeni dijelovi se kopiraju ili brišu sa njima što nije slučaj kod agregacije. Možemo reći bez klase Zubex nema ni Usluge, odnosno dok postoji klasa Zubex, postojat će i usluge. U usluge oridinacije spadaju npr. popravljanje zuba, slikanje, kontrola itd. Zubex čuva listu usluga, nešto kao cjenovnik, ali također i historiju izvršenih usluga. Klasa Pacijent je možda i najveća klasa našeg dijagrama. U klasi Pacijent se nalaze osnovne informacije o pacijentu, njegovom broju kartona, broju zakazanih pregleda itd. Ona je povezana agregacijom sa klasom Pregled. Pacijent može imati jedan ili više zakazanih pregleda. U klasi Pregled se nalaze informacije o pregledu pacijenta. Tu se nalazi datum i vrijeme pregleda, ime i prezime doktora, koji će pregledati pacijenta, u pregledu možemo vidjeti koja od usluga oridinacije je korištena. Može se provjeriti dijagnoza, vrsta liječenja, može se napisati nova dijagnoza, itd. Klasa Pacijet je povezana i sa klasom Karton, vezom kompozicije, jer karton pacijenta je usko vezan za njega i brisanjem pacijenta, briše se i njegov karton. Klase Pregled i Karton su povezane agregacijom. U klasi Karton se nalaze podaci o pregledima pacijenta, ako pacijent prvi put dolazi na pregled, otvara mu se novi karton, sa posebnim indentifikacionim brojem. U kartonu se nalazi lista svih usluga oridinacije. Bitno je napomenuti da se svaki dijagram klasa može nadograditi i poboljšati u slučaju dodatnih zahtijeva klijenta.

DIJAGRAM AKTIVNOSTI

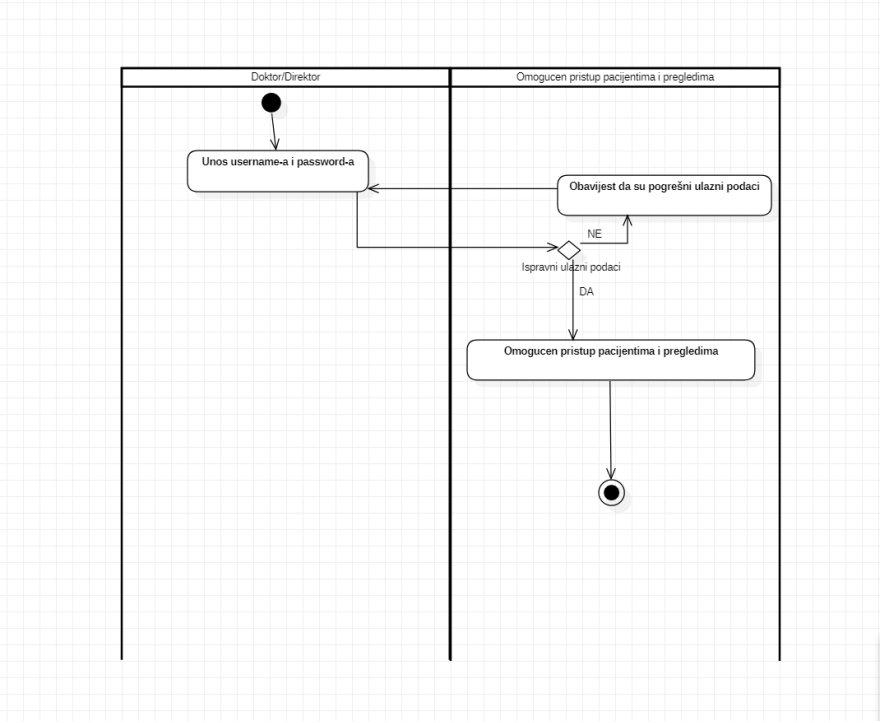
Pomoću dijagrama aktivnosti dozvoljeno nam je da specificiramo kako naš sistem radi, on je posebno dobar za opisivanje poslovnih procesa i poslovnih tokova. U nastavku se nalazi pet dijagrama aktivnosti i njih ćemo posebno opisati.



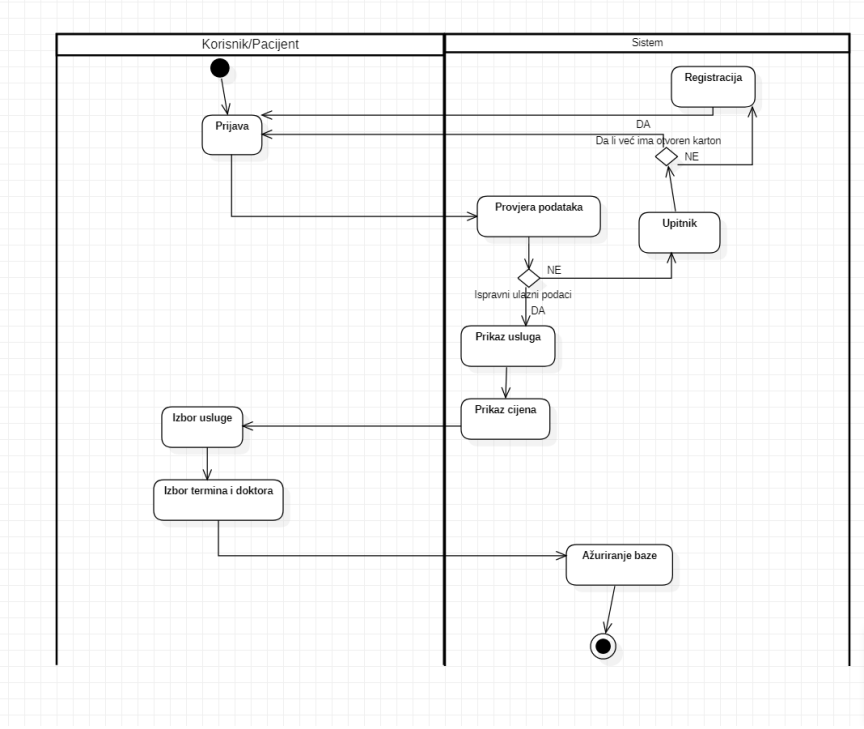
Ovim dijagramom je opisano ažuriranje i brisanje pacijenta od strane doktora.Doktor nakon što uradi pregled ima dvije opcije može ažurirati pacijenta ili ga obisati.Ukoliko mu je potrebno ažuriranje postoje dvije moguće situacije a to je ako je pacijent već prije nekad radio preglede onda se vrši dodavanje nalaza,a ukoliko nema ranijih pregleda njemu se dodaje prvi nalaz i time se ažuriraju podaci u bazi.Ukoliko je ipak potrebno da doktor obriše pacijenta dobit će prvo upitink da pitanjem da li je siguran da ga želi obrisati ako je odgovor pozitivan pacijent se briše u suprotnom se odbija brisanje paijenta i ponovo se vrši ažuriranje baze.



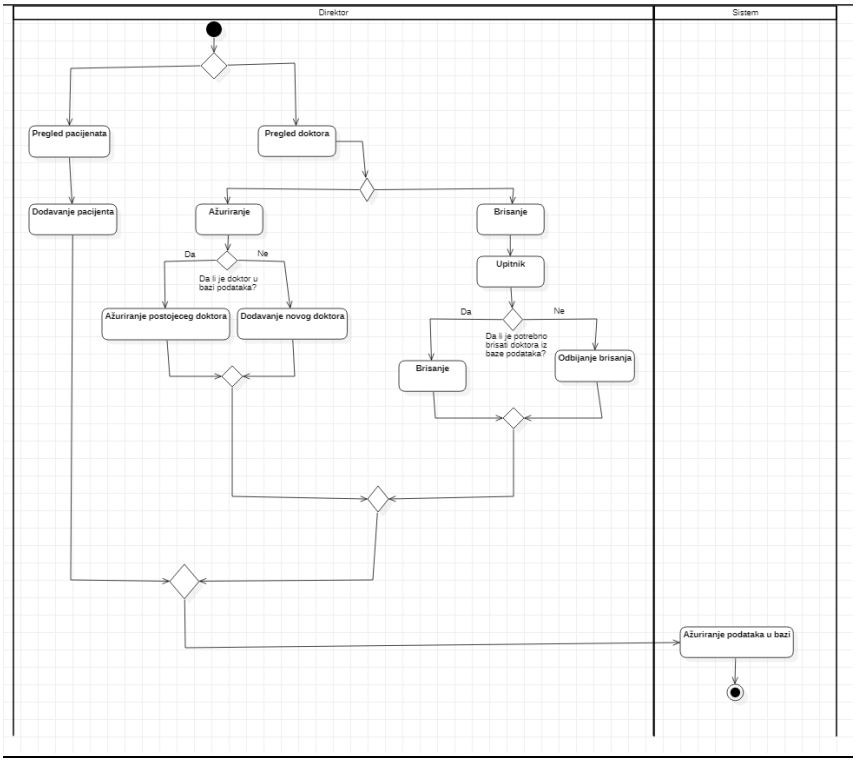
Ovim dijagramom je opisano otvaranje kartona korisniku/pacijentu.Prvo se unose osnovni podaci koje aplikacija zahtijeva zatim ukoliko su oni ispravnog formata vrši se otvaranje kartona i spašavanje podataka u bazu, a ukoliko unesni podaci nisu bili traženog formata javlja grešku i traži ponovni unos osnovnih podataka.



Ovim dijagramom je opisana prijava doktora i direktora, nakon što unesu svoj username i password vrši se provjera podataka ukoliko oni nisu tačnog formata pojavljuje se obavijest da su pogrešni ulazni podaci i ponovo se traži unos usernama i passworda, a ukoliko su tačnog formata omogućava se pristup svim pacijentima i pregledima.

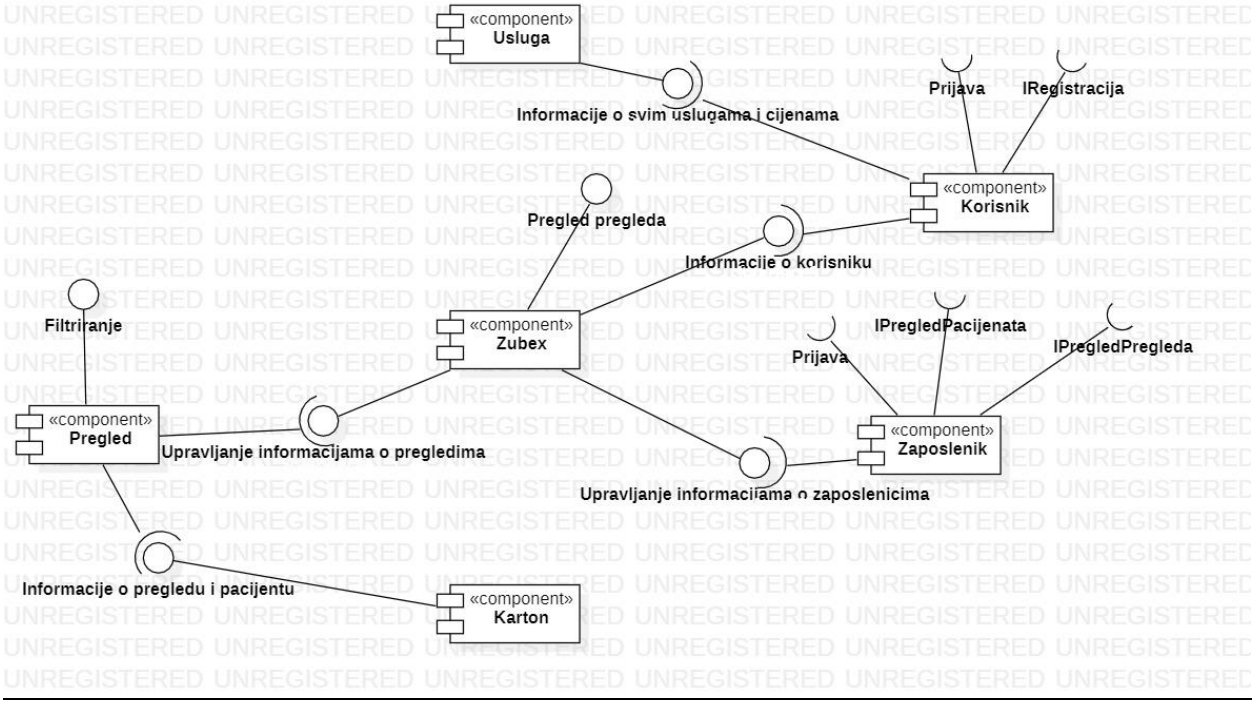


Ovim dijagramom je prikazan odnos sistema i korisnika/pacijenta.Nakon što se pacijent prijavi vrši se provjera podataka ukoliko su oni pogrešnog formata pojavljuje se upitik gdje se korinik pita da li ima već otvoren karton, ako je odgovor da ona se ponovo vraća na početak gdje treba unijeti tačne podatke za prijavu a ukoliko je odgovor negativan na upitik onda se vrši registracija tj. otvaranje kartona i zatim ponovo prijava.Na kraju kad se ulazni podaci tačni korisnik dobija prikaz usluga i cijena tih usluga i zatim on bira koja usluga/usluge su mu potrebne, zakazuje termin i doktora i na kraju se ažurira baza sa novim podacima.



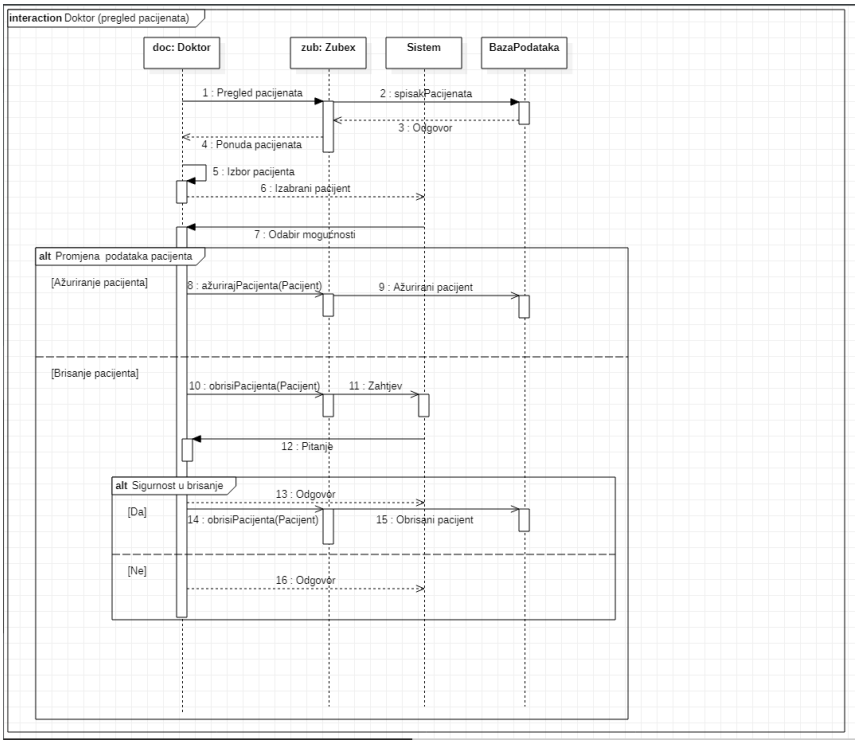
Ovaj dijagram opisuje ažuriranje pacijenta i doktora od strane diretkora.S obzirom da je direktor glavni i ima pristup svemu on može imati pregleda svih doktora kao i svih pacijenta.Što se tiče pacijenata može ih dodati i odmah nakon dodavanja se vrši ažuriranje baze.Doktore direktor može birsati i ažurirati.Ukoliko ih ažurira potrebno je znati da li je doktor već u bazi ako nije novi doktor se dodaje, a ko jeste ažurira se postojeći.A ukoliko je ipak potrebno obrisati doktora pojavljuje se upitik sa potanjem da li potrebno obrisati doktora iz baze podataka ako nije odbija se brisanja a ko je potrebno brisati doktora vrši se brisanje i ažuriranje baze podataka.

DIJAGRAM KOMPONENTE

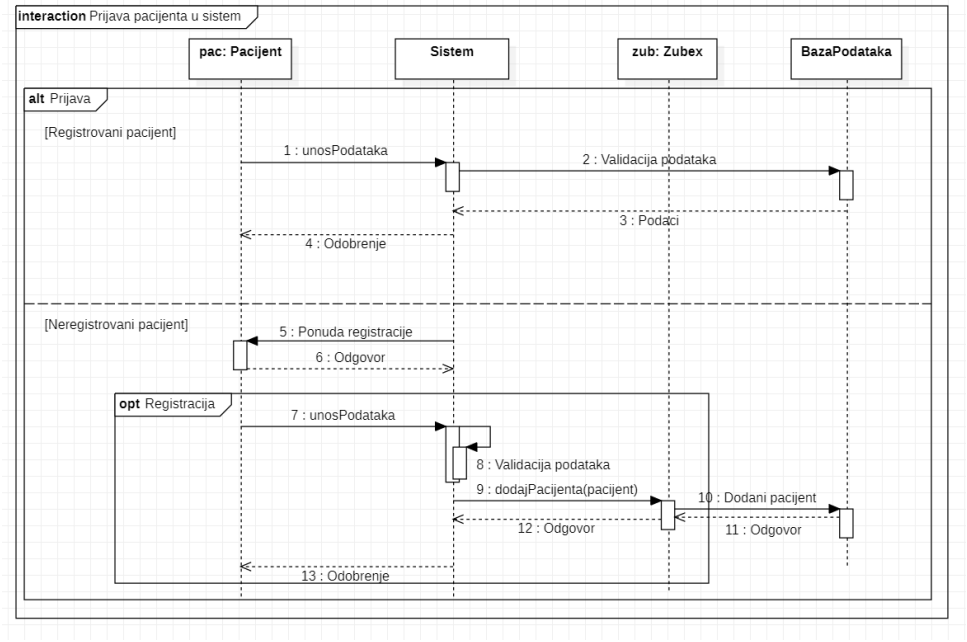


UML dijagram komponenti modelira komponente sistema i na taj način formira dio razvojnog pogleda.Komponente koje su bitne za naš sistem su Zubex, Usluga, Korisnik, Zaposlenik, Karton i Pregled. Komponenta Kornik zahtijeva interfejse Prijava, Registracija i Informcije o svim usugama i cijenama, dok komponenta Usluga upravo nudi zahtijevani interfejs Korisnika tj. Informacije o svim uslugama i cijenama, također komponenta Korisnik zahtijeva informacije o korisniku koje nudi Komponenza Zubex.Komponenta Zaposlenik zahtijeva interfejse Prijavu, PregledPregleda(svi pregledi), PregledPacijenata(svi pacijenti) i Upravljanje informacijama o zaposlenicima, a upravo taj interfejs nudi komponenta Zubex.Komponenta Zubex nudi informacije o korisnicima, Pregled pregleda(svi pregledi), upravljanje informacijama o svim zaposlenicima i upravljanje informacijama o pregledima a upravo taj interfejs zahtijeva komponenta Pregled.Komponenta Pregled nudi filtriranje pregleda i pored upravljanja informacijama o pregledima zahtijeva i informacije o pregledu i pacijentu a to je interfejs koji nudi komponenta Karton.

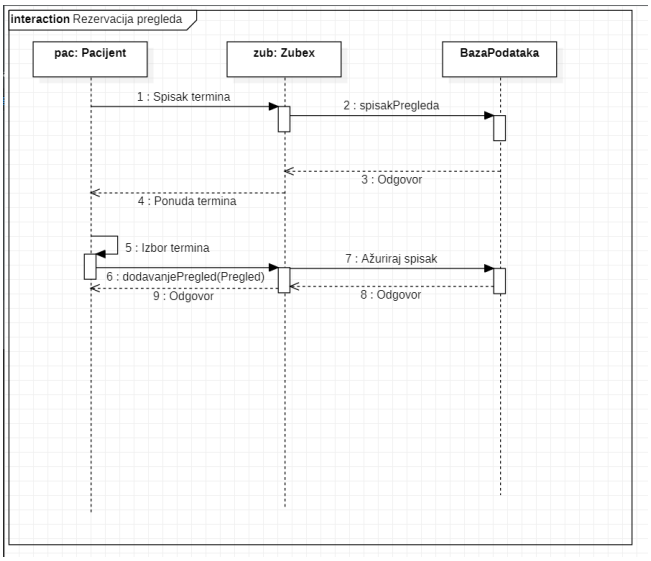
DIJAGRAM SEKVENCE



Doktor želi da vidi pregled svih pacijenata u sistemu. U alt fragmentu Promjena podataka pacijenta sistem mu nudi mogućnost ažuriranja pacijenta ili njegovog brisanja. Ako korisnik odluči da ažurira pacijenta to će biti pribilježeno u bazi podataka, ako se odluči za brisanje pacijenta, sistem će ga pitati u sigurnost njegove odluke i u zavisnosti od odgovora, ažurirat će se baza podataka ili će se korisnik vratiti na ranije dvije mogućnosti ažuriranja pacijenta ili njegovog brisanja.

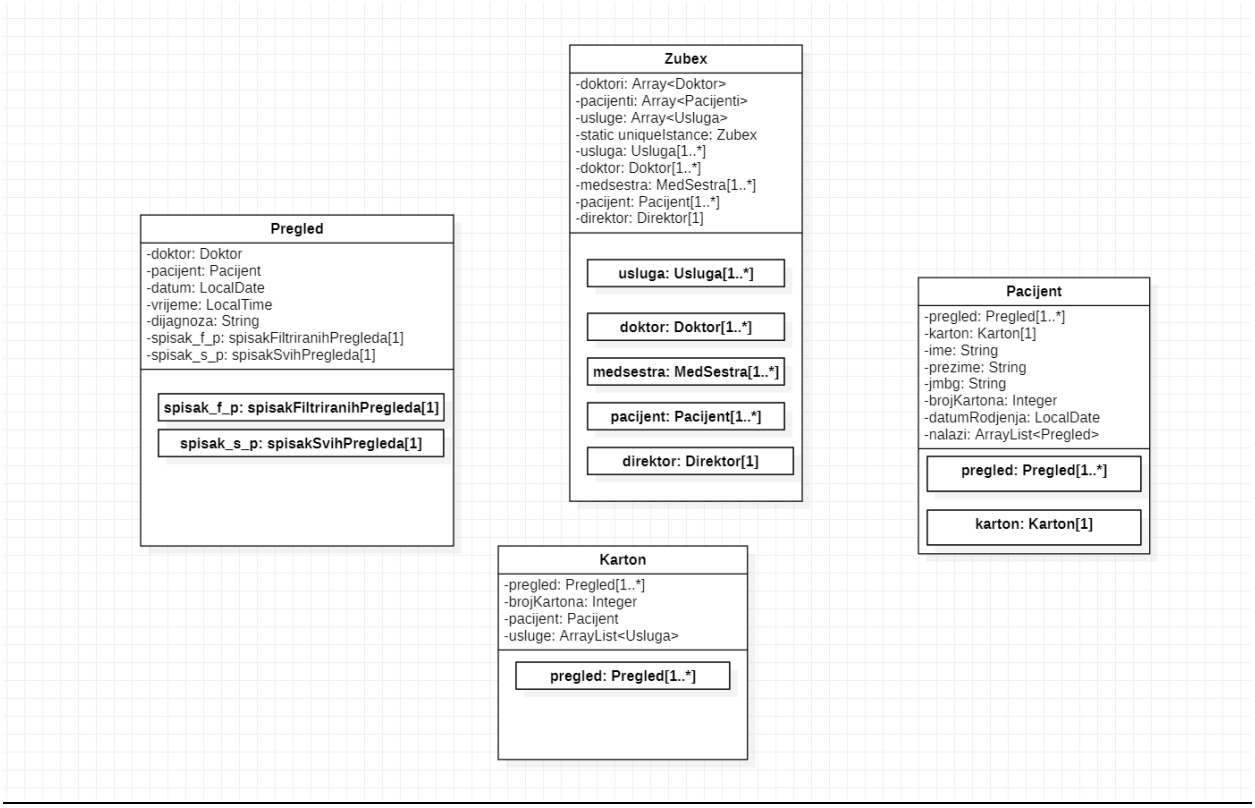


Grupi UML dijagrama interakcije pripadaju dijagrami sekvence. Najčešći oblik dijagrama interakcije koji se koristi u praksi jeste dijagram sekvenci (eng. sequence diagram). Ovaj sekvencijalni dijagram prikazuje interakciju između sistema i potencijalnog korisnika sistema, što je u našem slučaju pacijent. Na dijagramu se koristi operator alt, koji je operator grananja, koji je podijeljen na dva dijela. Korisniku, u zavisnosti od njegovog stanja u sistemu, odnosno da li korisnik već registrovan ili nije, nude mu se dvije mogućnosti. Ako je korisnik registrovan, nudi mu se prijava na sistem. Ukoliko nije registrovan, ponuđena mu je registracija. U nastavku dijagrama se koristi operator opt što znači da je ovo opcionalni fragment. Korisnik unosi potrebne podatke i registruje se u sistem.



Na ovom sekvencijalnom dijagramu korisnik želi da rezerviše pregled, od ponuđenih termina bira jedan i njegova rezervacija biva zabilježena.

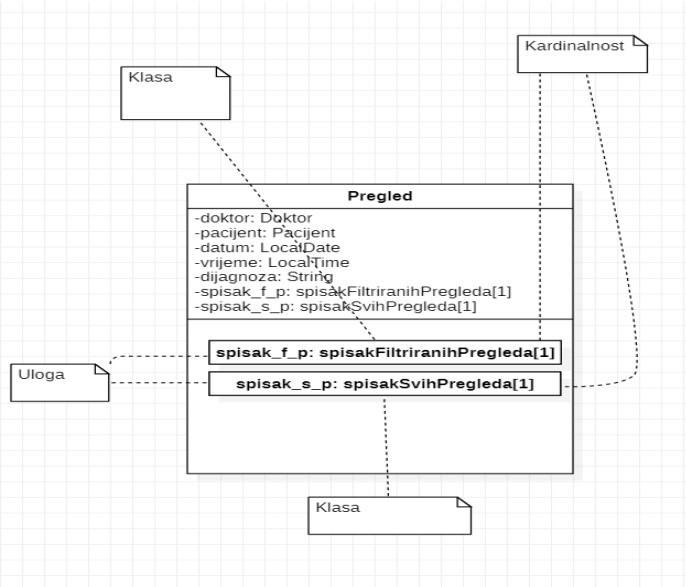
DIJAGRAM SLOŽENE STRUKTURE



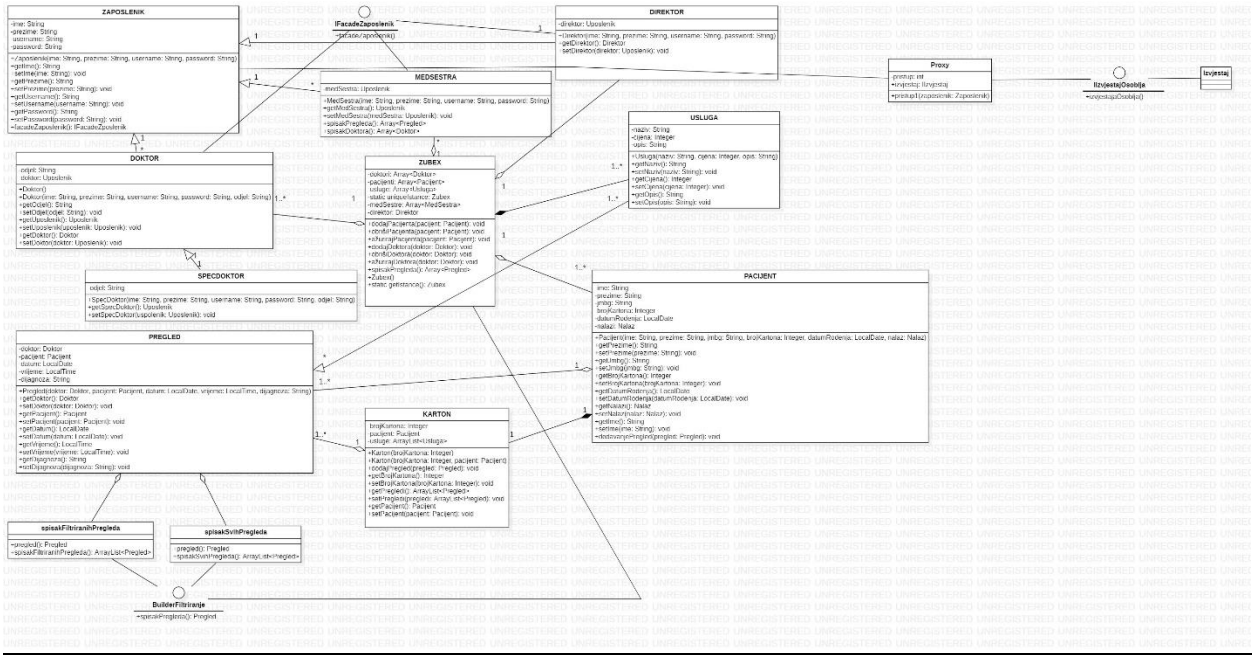
Dijagram složene strukture pomaže da se pregledno prikaže složena struktura i veza između klasifikatora visokog nivoa kao što su slučajevi upotrebe, klase i komponente. Ovaj tip UML dijagrama pokazuju i kako objekti rade zajednički da bi se postigla određena funkcionalnost sistema. Dijagram složene strukture pomaže u formiraju sveobuhvatnog logičkog pogleda na sistem.

Dijagram klase sadrži veze asocijacije i kompozicije. Dijagram složene strukture nudi alternativni način pokazivanja ovih relacija tako da se unutar klase mogu prikazati i njeni sastavni dijelovi. Dijagram složene strukture ustvari predstavlja internu strukturu klase. Interna struktura klase se prikazuju tako da se sve klase sa kojima je ta klasa u vezi kompozicije navedu unutar klase.

Klasa Zubex je povezana agregatnom vezom sa klasama Doktor, MedSestram Direktor i pacijent i vezom kompozicije sa klasom Usluga, tako da su te klase i njihovi atributi navedeni unutar strukture klase Zubex. Klasa Pacijent je povezana agregatnom vezom sa klasom Pregled i vezom kompozicije sa klasom Karton, tako da su te klase i njihovi atributi navedeni unutar strukture klase Pacijent. Klasa Karton je povezana agregatnom vezom sa klasom Pregled, tako da su ona i njeni atributi navedeni unutar strukture klase Karton. Klasa Pregled je povezana sa klasama spisakFiltriranihPregleda i spisakSvihPregleda agregatnom vezom, te su te klase i njihovi atributi navedeni unutar strukture klase Pregled. Kardinalnosti klasa i njihovih atributa su navedene unutar uglatih zagrada.



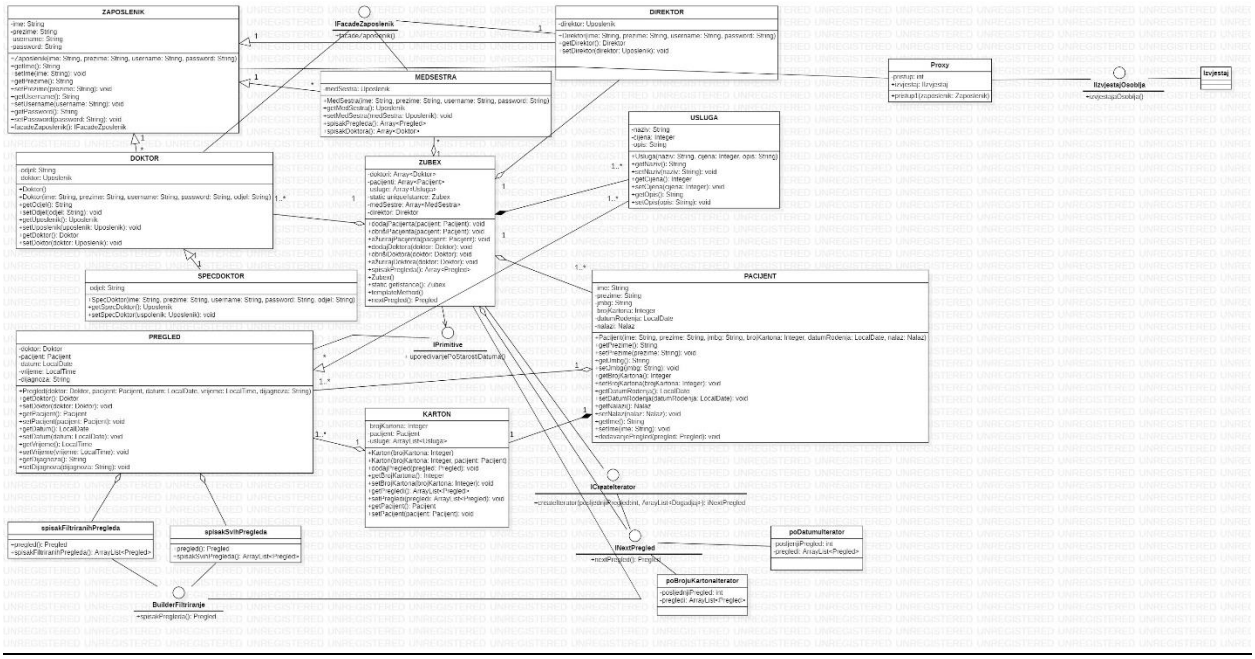
DIJAGRAM SA STRUKTURNIM I KREACIJSKIM PATERNIMA



Od kreacijskih paterna smo iskoristili Singleton i Builder, a od strukturnih Facade i Proxy patern. Zašto smo baš izabrali ove paterne navest ćemo sada.S obzirom da je uloga Singleton paterna da osigura da se klasa može instancirati samo jednom i da osigura globalni pristup kreiranoj instanci klase, iskoristili smo ga u našoj glavnoj klasi Zubex(jer nam je potrebno samo jedno instanciranje ove klase) gdje se dodao privatni static konstruktor, static read.only objekat koji će se instancirati korištenjem privatnog konstruktora i metoda pomoću koje pristupamo privatnom objektu te ga instancira ako objekat ne postoji.Uloga Builder je odvajanje specifikacije kompleksnih objekata o njihove stvarne konstrukcije.Ovaj patern je iskoristen u projektu kako bi se omogućilo dobijanja spiska svih pregleda i spisak pregleda koji će biti filtriran po nekom kriteriju koji ćemo mi zadati.Potrebno je napraviti interfejs Builderfiltriranje

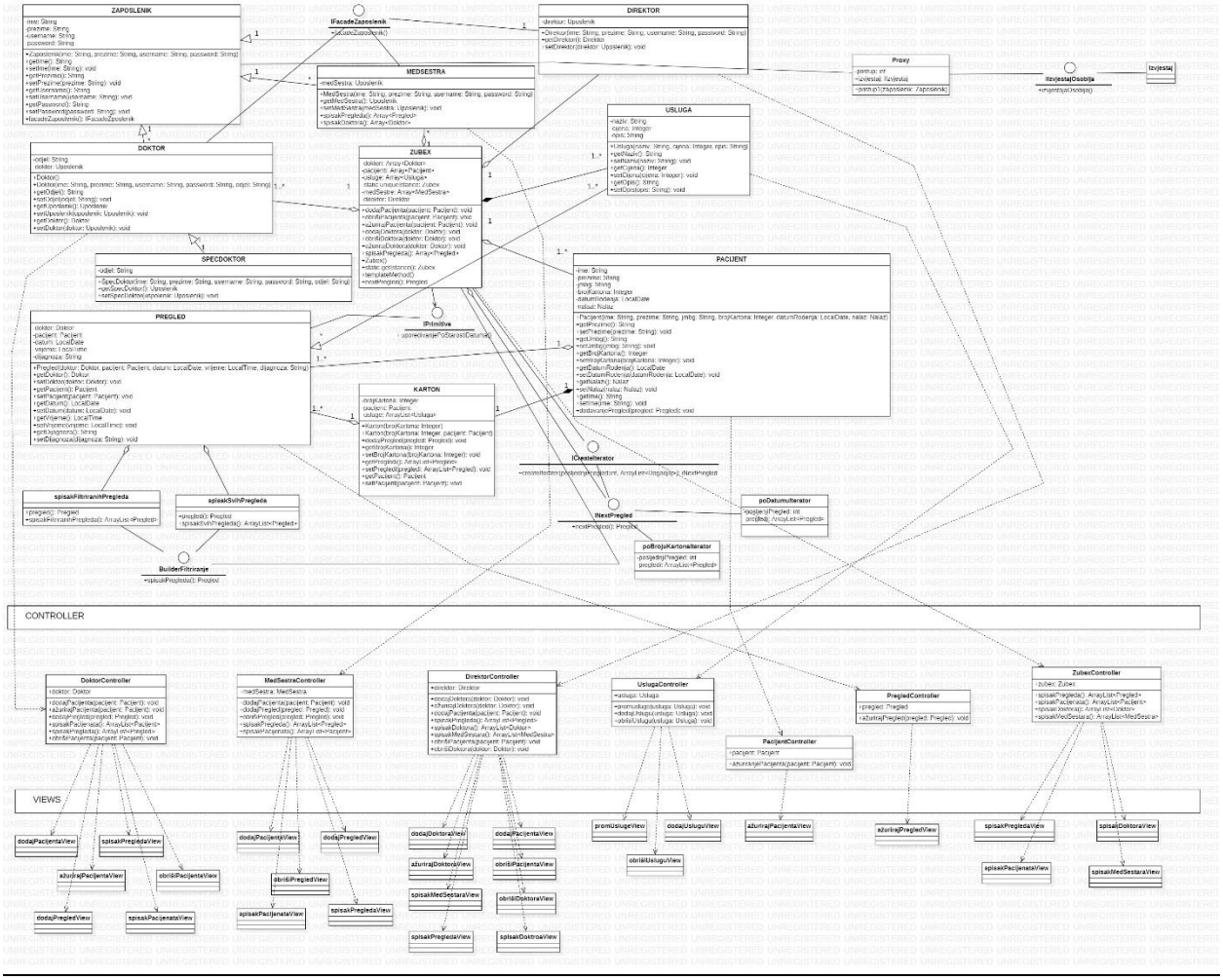
dvije klase koje se spajaju sa pregledom i na kraju to se sve spaja sa našom glavnom klasom Zubex. Facade patern se koristi kad je potrebno osigurati više pogleda visokog nivoa na podsistem tj. implemetacija sistema je skrivena od korisnika.Ovaj patern se veže za klasu Zaposlenik(apstraktna klasa), jer se unutar njega nalaze različte klase kao atributi , i pozivom različitih metoda iz zubex klase pozivaju se različite metode iz odgovarajućih klasa i time je urađen kompleksniji proces nego što je prikazano. Proxy patern se koristi za omogućavanje pristupa i kontrolu stvarnom objektima.U našem projektu ovaj patern smo iskoritili za omogućavanja pristupa pregleda osoblju, koji je omogućen samo direktoru.Dakle, potrebno je dodati proxy klasu u kojoj se dodaje metoda pomoću koje ćemo pristup postavljat na 1 ukoliko direktor pokušava pristupiti pregledu osoblja, ili 0 ukoliko je bilo koja druga osoba koja pokušava pristupiti pregledu osoblja.

DIJAGRAM SA PATERNIMA PONAŠANJA



Od paterna ponašanja u našem projektu su uvedeni Templatemethod i Iterator. TemplateMethod nam omogućava izdvajanje određenih koraka algoritma u odvojene podklase.Ovim paternom smo omogućili sortiranje pregleda, sortiranje će se vršiti po starosti datuma pregleda.U glavnu klasu Zubex se postavlja klasa Algorithm i tu se dodaje operacija TemplateMethod(), i u ovoj klasi će se vršiti sortiranje po našem zadatom kriteriju.Također je potrebno dodati novi interfejs Iprimitive koji sadrži metodu uporedivanjePoStarstiDatuma() i ovaj interfejs treba implementirati klasa Pregled(jer se to sortira). Iterator omogućava sekvencijalni pristup elemntima kolekcije bez poznavanja kako je kolekcija struktuirana.Ovaj patern je iskorišten za mogućnost pretraživanja,sortiranja pregleda/doktora/pacijenta.Zubex je klasa Client jer se tu nalazi spisak svih pacijenata, doktora i preglda, potrebno je dodati interfejs iduciPregled koji treba naslijediti sve ostale iteratore i još na kraju porebno je dodati IcreateIterator koji se povezuje sa samom klasom Zubex kako bi se omogućio pristup kolekciji.

DIJAGRAM SA MVC



MVC spada u softver dizajn paterne i u našem sistemu se nalazi 7 kontrolerskih klasa a to su: Doktor, MedSestra, Direktor, Usluga, Pregled, Pacijent i Zubex.Ti kontroleri predstavljaju povezujući most između prikazanih view klasa sa klasama modela sistema.Kao primjer korištenja MVC paterna pokazat ćemo na primjeru kada doktor treba brisati pacijenta.Kada se doktor uloguje na aplikaciju i želi obrisati nekog od pacijenata, on će komunicirati sa sistemom preko view ObrisiPAcijentaView, gdje se dobija jedan upitnik sa pitanje „Da li ste sigurni da želite obrisati pacijenta?“, nakon što doktor potvrdi brisanje on završava svoj rad sa aplikacijom.Ova akcija se dalje veze za metodu ObrisiPacijenta u klasi DoktorController koja dalje komunicira sa modelom i klasom Doktor.Na jako sličan način funkcionišu i ostali navedeni viewovi.

SOLID PRINCIPI

SOLID je akronim za 5 principa dobrog dizajna:

- **S**ingle Responsibility Principle – Princip pojedinačne odgovornosti
  Mislimo da je ovaj princip prilično zadovoljen, jer smo pokušali da sve koncepte odvojimo u njihove vlastite klase tako da svaka klasa ima jedan i samo jedan razlog za promjenu.

- **O**pen Closed Principle – Otvoreno zatvoren princip
  Ovaj princip nije narušen, jer mislimo da entiteti softvera bi trebali biti otvoreni za nadogradnju, ali zatvoreni za modifikacije.
- **L**iskov Substitution Principle – Liskov princip zamjene
  Mislimo da ovaj princip nije narušen, jer nijedan podtip ne može biti zamijenjen nekim osnovnim tipom.
- **I**nterface Segregation Principle – Princip izoliranja interfejsa
  Misllimo da ovaj princip nije narušen, jer korisnik nema veliki izbor metoda na raspolaganju i one su prilično dozirane njegovim potrebama. Sistem ne posjeduje “debele” klase. Tako da nije potrebno korisnika štiti od metoda koje im ne trebaju i od poznavanja implementacije objekta kojeg koristi.
- **D**ependency Inversion Principle – Princip inverzije ovisnosti
  Mislimo da ovaj princip nije narušen, jer sistem ne ovisi od konkretnih klasa. Prilikom nasljeđivanja osnovne klase su apstraktne.