FDSS –
FISCAL DEVICE SERVICING SYSTEM

25.04.2014. System Design

FDSS - Fiscal Device Servicing System

SYSTEM DESIGN

Sadržaj

Historijat revizije dokumenta

Datum	Opis verzije	Autor	Komentar
25.04.2014	v 1.0	Tim 6	Prva verzija dokumenta

1. UVOD

1.1. Svrha dokumenta

Dizajn sistema je dokument koji detaljno opisuje softverske komponente i njihove međusobne veze. Ovaj dokument je baziran na Specifikaciji sistemskih zahtjeva (SRS) za Fiscal Device Servicing System (FDSS), te ima za cilj prevesti funkcionalne zahtjeve opisane u SRS-u u konkretne softverske komponente, koje u cjelini, implementiraju i nefunkcionalne zahtjeve.

Dizajn sistema identifikuje, opisuje i prikazuje:

- Kontekst i načine korištenja sistema
- Dizajn sistemske arhitekture
- Ključne objekte u sistemu
- Dizajn modele
- Interfejse objekata

Glavna namjena ovog sistema je dokumentovati kompletan dizajn sistema, koristeći pri tome dijagrame i opise različitih patterna. Sistem će biti prikazan iz niza različitih pogleda: logičkog, procesnog, implementacijskog i razvojnog, a zatim i kroz niz različitih slučajeva upotrebe. Za izradu dijagrama će se koristiti UML 2.0 standard. To je standard koji opisuje standardnu notaciju za crtanje dijagrama.

Na kraju, ovaj dokument, zajedno sa Specifikacijom korisničkog interfejsa (UIS) će se koristiti za naredne faze razvoja FDSS sistema. Na osnovu ovog dokumenta će se izraditi niz prototipa sistema, a zatim izvesti i kompletna implementacija sistema. Specifičnosti sistema prikazane u ovom dokumentu, ali i u UIS-u, će sa jedne strane koristiti naručiocu softvera da percipira šta će tačno sistem raditi, na koji način će to raditi, te da validira u odnosu na svoje zahtjeve (naravno, u mjeri u kojoj je to moguće, budući da mnogi dijelovi ovog dokumenta koriste tehničku terminologiju i opisuju dijelove koji su duboko vezani za implementaciju, a samim tim nerazumljivi naručiocu softvera). Sa druge strane, razvojni tim će koristiti ovaj dokument kao polaznu osnovu za kreiranje prototipova i implementaciju sistema. U tom smislu, namjena ovog dokumenta je da unaprijedi razumijevanje samog sistema, a zatim i detaljno opiše šta će tačno sistem raditi i na koji način će to biti implementirano, što će u konačnici u mnogome olakšati implementaciju.

1.2. Opseg dokumenta

Ovaj dokument će u nastavku:

1. Opisati kontekst i načine korištenja sistema putem **Use case dijagrama**. Use case dijagram cijeli sistema će pokazati najopštiji slučaj upotrebe sistema. Samim tim će pokazati kako identificirani korisnici koriste sistem, te sa kojim drugim sistemima FDSS ima interakcije. Također, u ovom dijelu će se pokazati i određeni specifični slučajevi upotrebe sistema.

2. Prikazati arhitekturu kompletnog sistema. U sklopu toga, ovaj dokument će identificirati ključne softverske komponente sistema, kao i njihove međusobne veze. Za potrebe arhitekturalnog dizajna, opisat će se i **arhitekturalni patern** koji će implementirati FDSS. Da bi se u potpunosti prikazala sistemska arhitektura, ovaj dokument će uključiti sljedeće dijagrame:

a. **Dijagram komponenti** – ovim dijagramom će se prikazati ključne softverske komponente sistema, njihove interfejse i međusobne ovisnosti komponenti.
b. **Dijagram raspoređivanja** – ovim dijagramom će se identificirati hardverske komponente potrebne za implementaciju sistema, te način na koji se softverske komponente raspoređuju na njih

3. Identificirati ključne objekte u sistemu, zajedno sa njihovim interfejsima. U ovom dijelu dokumenta će se opisati svi objekti koji se mogu identificirati u ovoj fazi razvoja sistema, a opisat će se u smislu njihovih atributa i metoda. Pored toga, jasno će biti naznačeni interfejsi tih objekata prema drugim objektima u sistemu. Zatim će se opisati međusobne interakcije ovih objekata, te njihovo logičko grupisanje. Pokazat će se statički i dinamički modeli objekata sistema. Također, razmotrit će se i upotreba **dizajn paterna**. Za potrebe ovog dijela koristit će se sljedeći dijagrami :

    a. **Dijagram klasa** – statički model objekata sistema. Ovim dijagramom će se identificirati sve klase u sistemu, pokazati atributi i metode ovih klasa, te interfejsi tih klasa prema drugim klasama u sistemu, ali i prema izlazima iz sistema. Sljedeći ključan apekat koji će biti prikazan ovim dijagramom su veze između klasa. Ovaj model je statičan budući da pokazuje strukturu sistema, ne uzimajući u obzir dinamičke aspekte izvršavanja.
    b. **Dijagram objekata** – dinamički model koji pokazuje ponašanje konkretnih objekata nastalih kao instance klasa opisanih dijagramom klasa. Ovaj dijagram će pokazati stanje objekata sistema u određenom trenutku izvršavanja sistema – vrijednosti atributa konkretnih objekata i njihove međusobne linkove.
    c. **Dijagrami sekvence** – ovim dijagramima će se pokazati sekvence interakcije između konkretnih objekata u sistemu u određenim slučajevima upotrebe sistema (u određenim trenutcima oslanjajući se na određene scenarije).
    d. **Dijagram paketa** – dijagram koji će prikazati logičko grupisanje klasa i objekata u pakete.
    e. **Entitet-relacija dijagram** – dijagram kojim će se pokazati detaljna arhitektura baze podataka. Ovaj dijagram će identificirati sve entitete u bazi podataka, atribute i pripadajuće tipove atributa svakog od entiteta, te relacije između entiteta.

4. Predstaviti dizajn modele koji će prikazati sistem iz pogleda koji prethodnim dijelovima dokumenta nisu obuhvaćeni. Tu ćemo razmotriti sljedeće dvije vrste dijagrama :

    a. Dijagram aktivnosti – ovim dijagramima će se prikazati kako sistem radi, odnosno koje aktivnosti se izvršavaju u sistemu, te od kojih akcija se sastoje te aktivnosti. Također, ovaj dijagram će pokazati kako pojedini objekti nastaju kao rezultat određenih akcija.
    b. Dijagram stanja – ovaj dijagram će pokazati stanja u kojem se sistem i pojedini objekti mogu nalaziti, a zatim i tranziciju između tih stanja.

Dakle, ovaj dokument će dokumentovati dizajn sistema u najširem smislu.

Ono što se nalazi izvan opsega ovog dokumenta je sljedeće:

- Korisnički interfejs. Iako dokument obuhvata i opisuje dijelove interakcije korisnika sa sistemom, na određenom nivou apstrakcije, te interfejse objekata, opis i prikaz grafičkog korisničkog interfejsa nisu dijelovi ovog dokumenta. Oni su specificirani dokumentom „Specifikacija korisničkog interfejsa FDSS-a".
- Korisnički i sistemsi zahtjevi. Iako se dokument kontinuirano referira na ove zahtjeve, te sve što je dizajnirano je bazirano na zahtjevima naručioca softvera, dokument ne uključuje specifikaciju i opise ovih zahtjeva. Oni se nalaze u dokumentu „Specifikacija sistemskih zahtjeva FDSS-a".
- Implementacija. Ovaj dokument ne razmatra aspekte implementacije niti bilo koje pitanje u vezi sa implementacijom. Međutim, dizajn koji je dokumentovan ovdje će se koristiti kako bi se na osnovu njega i implementirao sistem.

1.3. Opis sistema

Osnovna namjena ovog proizvoda jeste da omogući kompanijama koje se bave održavanjem i servisiranjem fisikalnih uređaja što jednostavnije poslovanje, to jest da na što jednostavniji način vode evidenciju o resursima sa kojim raspolažu, kako ljudskim tako i materijalnim, kao i jednostavniju evidenciju vitalnih podataka. Dokument sadrži sljedeće namjene, funkcionalnosti i okvire sistema :

Bullet board podsistem – Predstavlja naslovnu stranu sa aktuelnim novostima i obavijestima. Prikazuje zadatke koje logovani korisnik treba da izvrši i u kojem roku.

Dispatcher podsistem – Evidentira se tačno vrijeme telefonskog poziva kojim se zahtjeva servis nekog uređaja, kao i osnovni podaci o klijentu. Kreira se novi zahtjev i obavještava odgovorni menadžer preko njegovog bullet board-a. Takođe se vrši dispatch servisera na neki određeni zadatak. Zatvaraju se zadaci koji su izvršeni. Prikazuje se status svih aktivnih zadataka i servisera kojima je dodijeljen taj zadatak.

Podsistem za evidenciju stanja i informacija o uređajima koji su na servisu – Omogućava unos podataka o svim uređajima i korisnicima koji koriste usluge servisiranja, brisanje podataka o svim uređajima i korisnicima koji koriste usluge servisiranja, kao i ažuriranje podataka o svim uređajima i korisnicima koji koriste usluge servisiranja.

Podsistem za pregled svih korisnika i uređaja – Nudi mogućnost prikaza i pretrage svih uređaja i korisnika koji koriste usluge servisiranja, kao i poruka upozorenja(alarmiranje) vezano za uređaje kojima ističe rok servisiranja.

Podsistem za generisanje izvještaja – Omogućava generisanja različitih tipova izvještaja kao :

- Izvještaj o popravci uređaja kojeg kreira serviser
- Izvještaj o uposlenicima kojeg kreira menadžer

Svi izvještaji se mogu kreirati na određeni period: sedmica, mjesec, kvartal, godina...

Administracija sistema – Pod administracijom sistema podrazumijevamo pregled svih trenutnih uposlenika, unos novih uposlenika koji koriste sistem, brisanje uposlenika koji više nemaju dodira sa sistemom, ažuriranje podataka o unesenim zaposlenicima, određivanje privilegija korisnicima kao i mogućnost uređivanja profila izmjene username/password podataka. Takođe je zastupljena i mogućnost mapiranja ranijih podataka iz Excel tabela u bazu podataka.

1.5. Vanjske reference

Dokumenti na koje se referira ovaj dokument, odnosno dokumenti na temelju kojih se izrađuje ovaj dokument su sljedeći:

1. Specifikacija sistemskih zahtjeva (SRS) za Fiscal Device Servicing System (FDSS)
2. Specifikacija korisničkog interfejsa (UIS) za FDSS
3. Prof. dr. Đonko, Dženana i prof. dr. Omanović, Samir. Objektno orijentirana analiza i dizajn primjenom UML notacije. Sarajevo, 2009.

1. DIZAJN SISTEMA

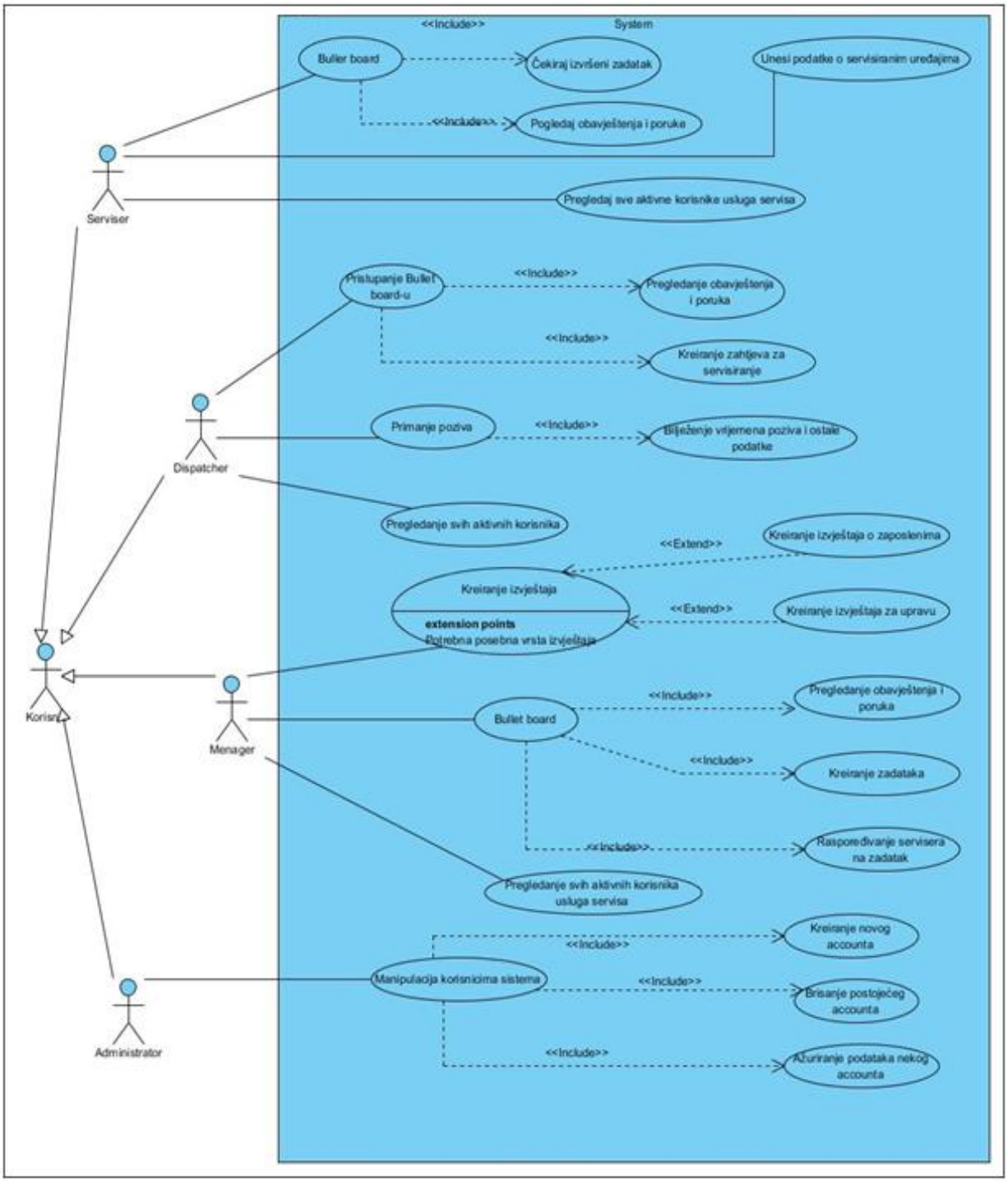
Dizajniranje sistema je vitalna faza u svakom projektu jer tu ustvari po prvi put bilježimo kako će sistem zaista izgledati. U narednim pglavljima ćemo se osvrnuti upravo na dizajn našeg FDSS, a preći ćemo :

- Kontekst i način korištenja sistema
- Arhitekturalni dizajn
- UML modeliranje sistema
- Dizajn baze podataka

Dizajniranje korisničkog interfejsa je zbog opširnosti izostavljeno, i biti će obrađeno u zasebnom dokumentu.

1.1. Kontekst i način korištenja sistema

1.1.1. Use Case dijagram

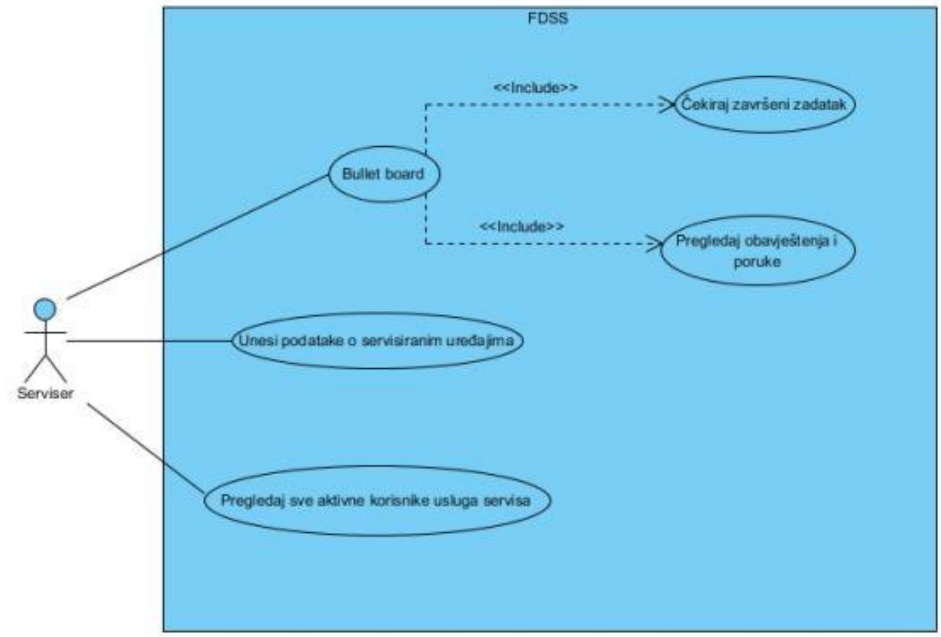


Slika 1 – Use Case dijagram cjelokupnog sistema

Opis : Korisnici sistema mogu da budu različite osobe, koje imaju svoje funkcije. U skladu sa tim korisnici imaju i različite mogućnosti pristupa svim funkcionalnostima sistema. Korisnici sistema mogu da budu: administrator, menadžer, radnik u dispatcheru i serviser.

Ukoliko se korisnik prijavi na sistem kao serviser, on ima mogućnost pregleda svojih zadataka i unos podataka o servisiranim uredjajima. Radnik u dispatcheru može pristupiti bulett boardu i kreirati zahtjev za servisiranje uređaja, te može da prima pozive i bilježi pojedinosti o njima. Menadžer ima nesto više mogućnosti prilikom rukovanja sa sistemom. On može da kreira izvještaje, pristupa bullet boardu, kao i da pregleda aktivnosti svih korisnika sistema. Administrator sistema ima mogućnost da manipuliše korisnicima sistema.

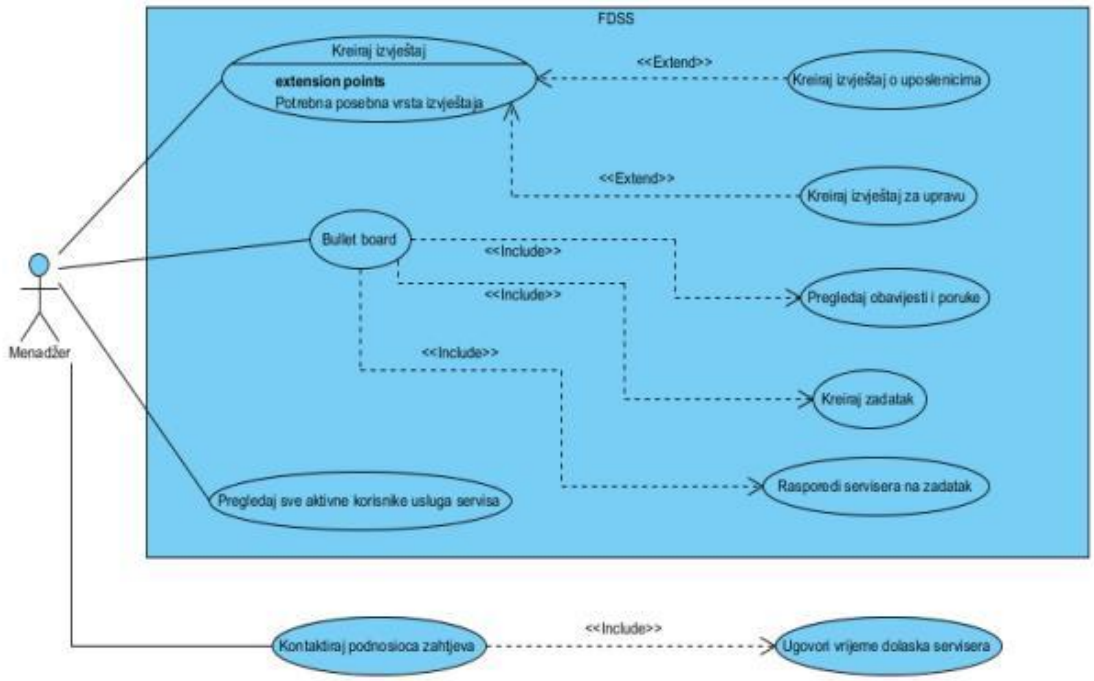
1.1.1.1. Detaljniji pogled – Interakcija servisera sa sistemom



Slika 3 – Use Case dijagram interakcije servisera sa sistemom

Opis : Obični serviser ima pristup najosnovnijim funkcijama sistema. On može da pristupi unosu uređaja koji se servisiraju, pregledu spiskova svih uređaja i korisnika, kao i pristup svom vlastitom bullet boardu, na kojem ima pregled aktivnih zadataka i rokove do kada ih mora završiti, kao i mogućnost označavanja izvršenih zadataka.

1.1.1.2. Detaljniji pogled – Interakcija servisera sa sistemom

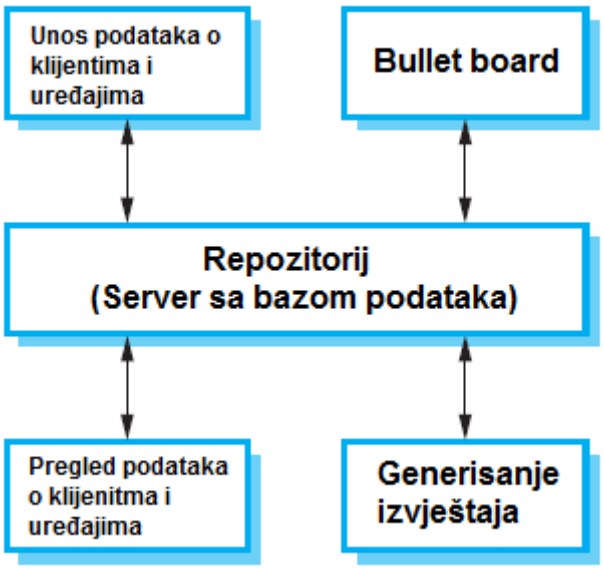


Slika 4 – Use Case dijagram interakcije menadžera sa sistemom

Opis : Korisnik sistema se prilikom logovanja prijavljuje kao menadžer. Menadžer, kao što smo već rekli, može da pristupi najviše funkcija prilikom korištenja sistema. Kao i serviser i radnik u dispatcheru, on može da pristupi vlastitom bullet boardu, što uključuje pregled poruka i obavještenja, kreiranje zadataka i raspored servisera na zadatak. Dodatna mogućnost koju nemaju ostali korisnici je kreiranje izvještaja, u šta spada kreiranje izvještaja o uposlenicima i kreiranje finansijskih izvještaja, i to sve na određeni vremenski period (mjesečno, kvartalno, godišnje). Menadžeri, u skladu sa svojom funkcijom imaju mogućnost pregledanja aktivnosti svih korisnika usluga sistema.

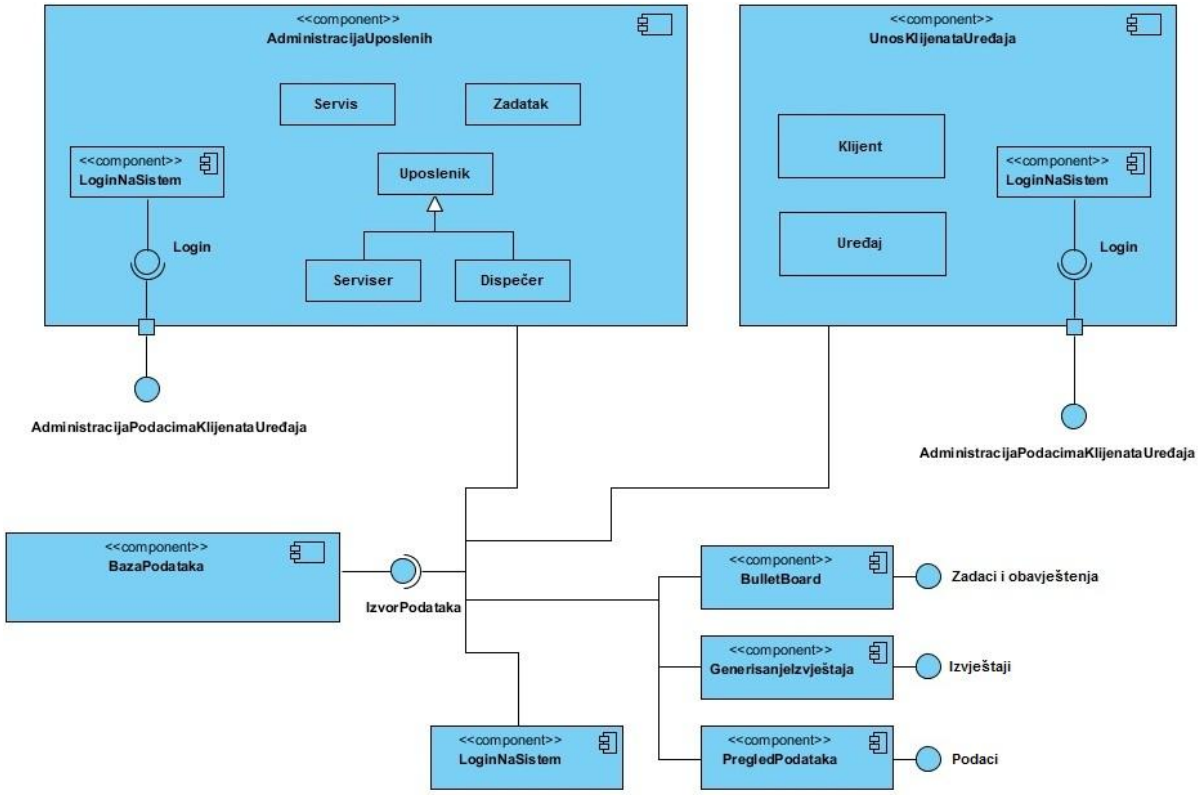
1.2. Arhitekturalni dizajn

Od arhitekturalnih paterna za naš sistem koristili smo repozitorij. Ovaj patern koristi se kada nam je u sistemu neophodna pohrana velike količine podataka na duži vremenski period, što je upravo osnovna namjena našeg sistema. (pohrana podataka o uređajima, klijentima...) To znači da će svi podaci u sistemu biti smješteni u centralni repozitorij koji će biti dostupan svim ostalim komponentama sistema, te one neće komunicirati direktno, već samo kroz repozitorij.



Slika 2 – Nacrt primjene arhitekturalnog patterna

1.2.1. Dijagram komponenti



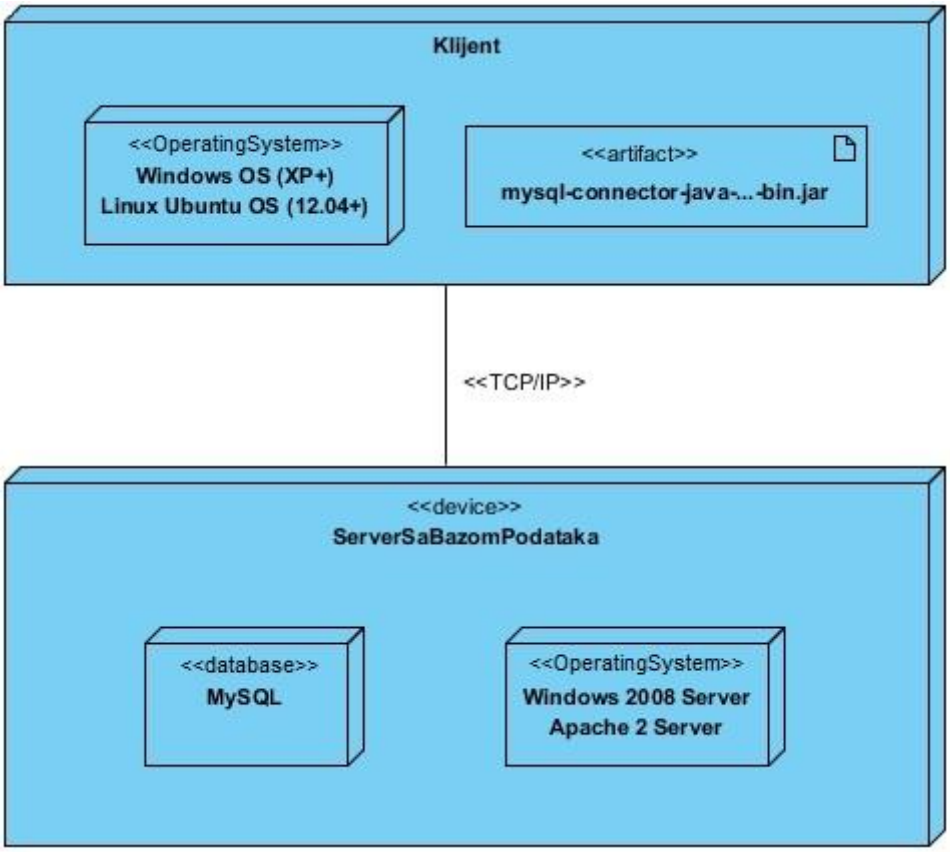
Slika 3– Dijagram komponenti sistema

Opis : Kao što znamo, komponente su dijelovi sistema koje mogu da funkcionišu zasebno i potpuno samostalno, u izolaciji od drugih dijelova sistema. Na ovom dijagramu vidimo koji su to takvi dijelovi FDSS sistema. To su komponenta za unos podataka o klijentima i uređajima, kao i komponenta za administraciju podataka uposlenih. One nude interfejs upravo za pristup administriranju ako je korisnik logovan sa tim privilegijama, a traže interfejs izvora podataka, koji ostvaruju naravno preko baze podataka, koja ovdje figuriše kao zaseban komponenta. Ostale komponente uglavnom služe za prikupljanje, obradu i prikaz podataka iz baze, a to su:

- Bullet board
- Generisanje izvještaja
- Pregled podataka

Jedna zanimljiva komponenta jeste login na system. Login se ostvaruje tako što se unosi username i password, te se na osnovu toga identificira korisnik. Zanimljiv je upravo po tome što figuriše kao vitalna komponenta drugih komponenti jer upravo reguliše pristup podacima.

1.2.2. Dijagram raspoređivanja



Slika 4 – Dijagram raspoređivanja sistema

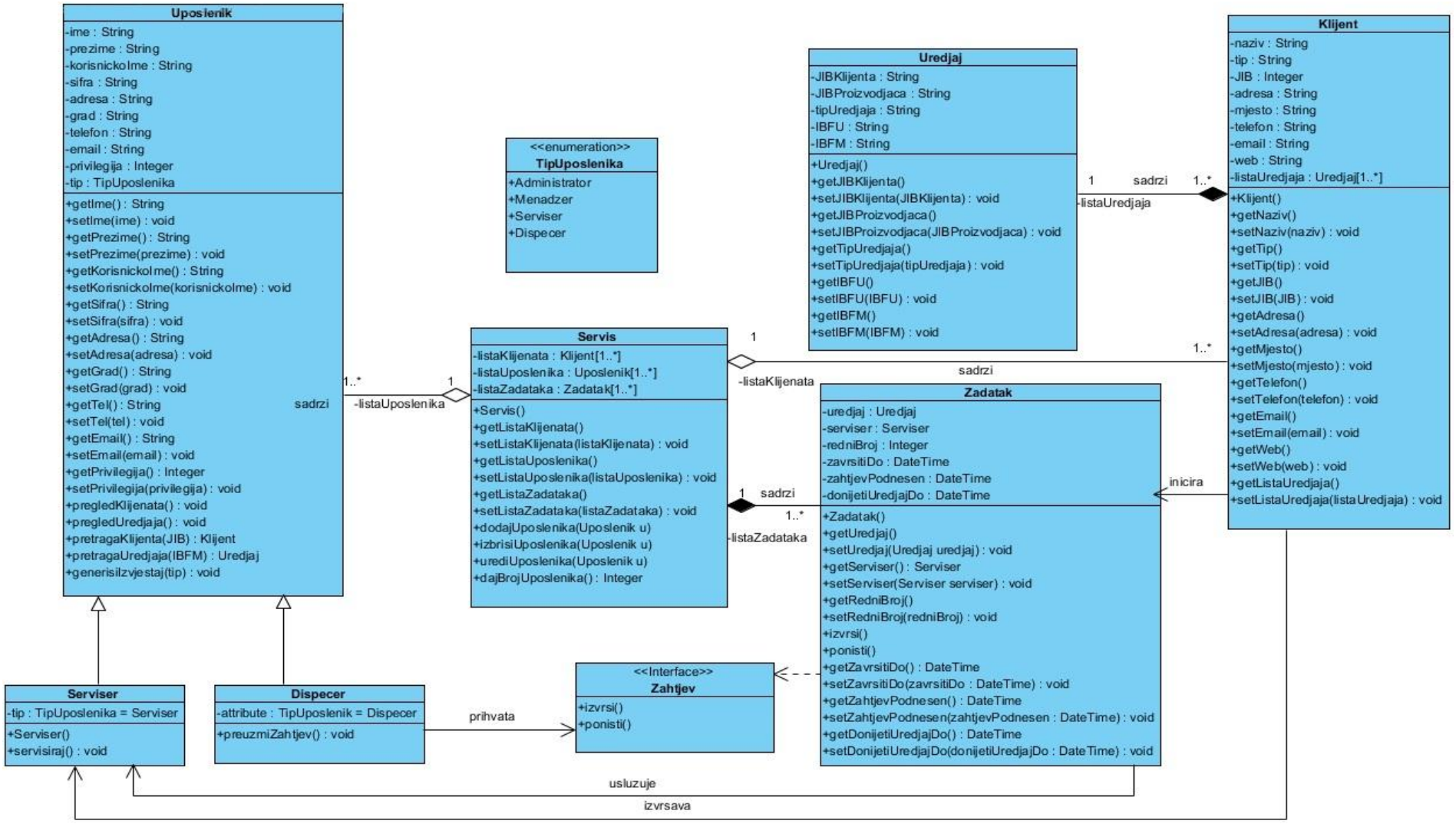
Opis : Ovdje vidimo konkretan način implementacije FDSS sistema. Pošto se sistem uglavnom svodi na komunikaciju aplikacije sa bazom podataka, vidimo da je upravo to definisano na ovom dijagramu. Računari na kojima je instalirana aplikacija moraju imati instalirani operatvni sistema Windows XP(ili više) ili Linux ubuntu 12.04(ili ivše), te takođe moraju sadržavati artefakt koji uopće omogućuje komunikaciju sa bazom podataka. To je naravno .jar file koji se dobije u projektu MySQLConnector/J.

Na serveru takođe treba da bude instaliran neki od serverskih operativnih sistema, a to mogu biti Windows Server 2008 ili Apache 2 server. Kao što je već više puta navedeno koristiti će se open source rješenje za DBMS u obliku MySQL baze podataka.

1.3. UML modeli sistema

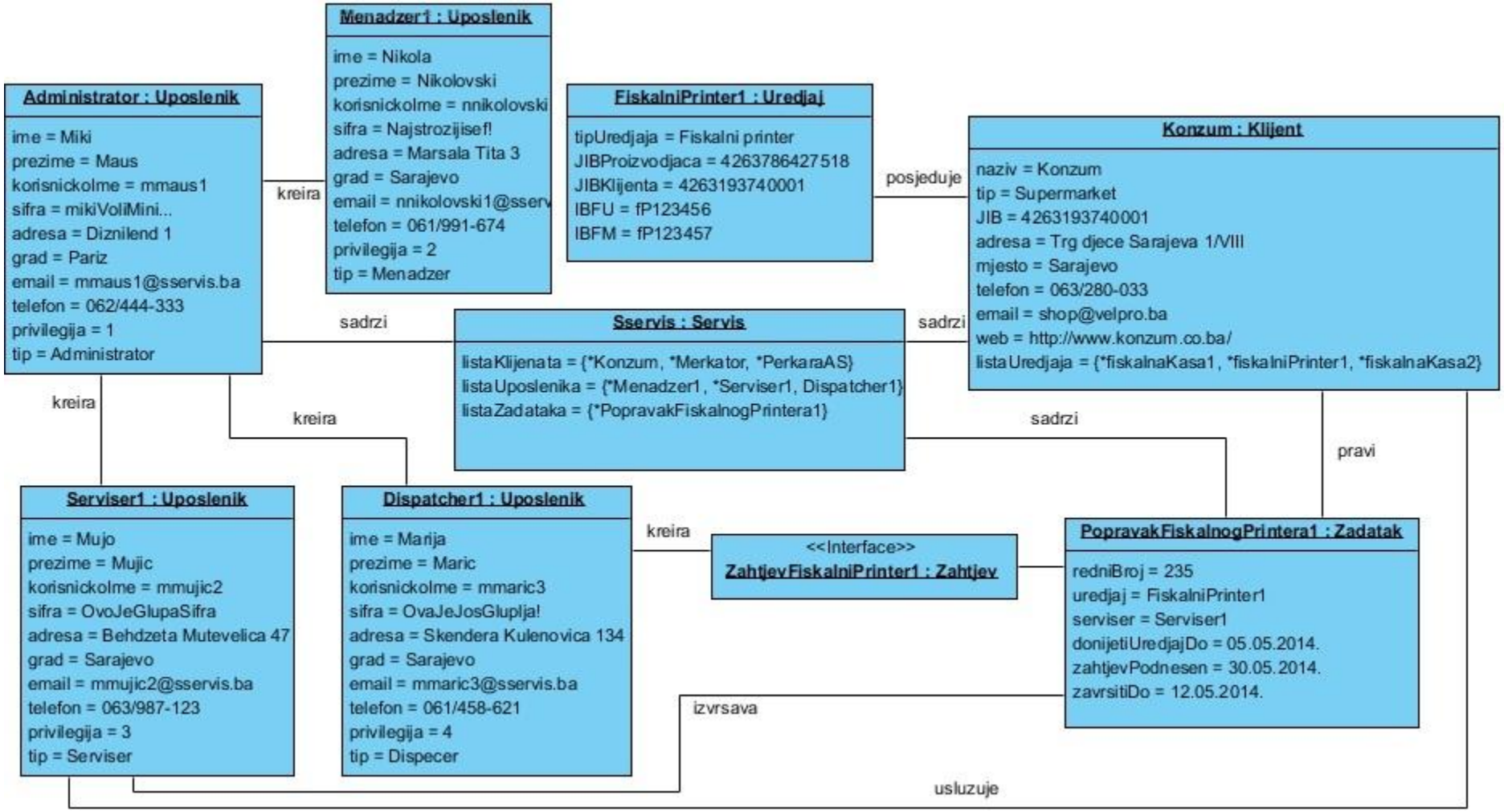
U ovom dijelu ćemo se predstaviti način funkcionisanja FDSS sistema, kao i njegovu hardversku implementaciju koristeći UML notaciju. Prvo ćemo pregledati način na koji korisnici vode interakciju sa sistemom kroz use case dijagrame.

1.3.1. Dijagram klasa



Slika 5 – Dijagram klasa sistema

1.3.2. Dijagram objekata



Slika 6 – Dijagram objekata sistema

Opis : Kako u servisu postoje 3 vrte uposlenika i administrator koji kreira uposlenike i brine za sve vezano za sistem, kreirali smo klasu Uposlenik koja čuva podatke o uposlenicima. Kako ova klasa za atribute ima korisničko ime, šifru i tip, prilikom prijave na sistem, sistem prepozna o kojem tipu uposlenika je riječ i prikaže mu odgovarajući interfejs. Tipovi uposlenika pobrojani su u klasi enumeracije TipUposlenika.

Na ovaj način omogućeno je lagano dodavanje novog uposlenika u sistem.
Dijagram klasa sadrži još Klijenta i Uređaj. Ove klase čuvaju i obrađuju podatke za koje je to predviđeno i zakonom. One su povezane kompozicijom (jaka agregacija) koja povezuje dio s cjelinom s tim da se dio ne može izostaviti od cjeline.

Centralna klasa ovog sistema je kontejnerska klasa Servis u kojoj se čuvaju svi podaci (o klijentima, uposlenicima, zadacima...)u odgovarajućim listama.

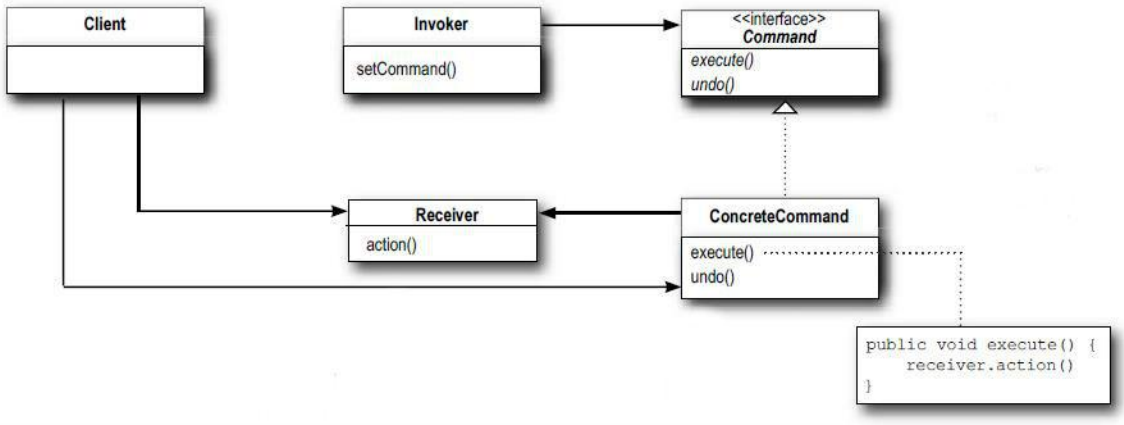
Preostale klase opisane su prilikom opisa dizajn paterna koji smo koristili prilikom kreiranja dijagrama klasa.

1.3.3. Design pattern

Na sistem smo primijenili tzv. „Command pattern".

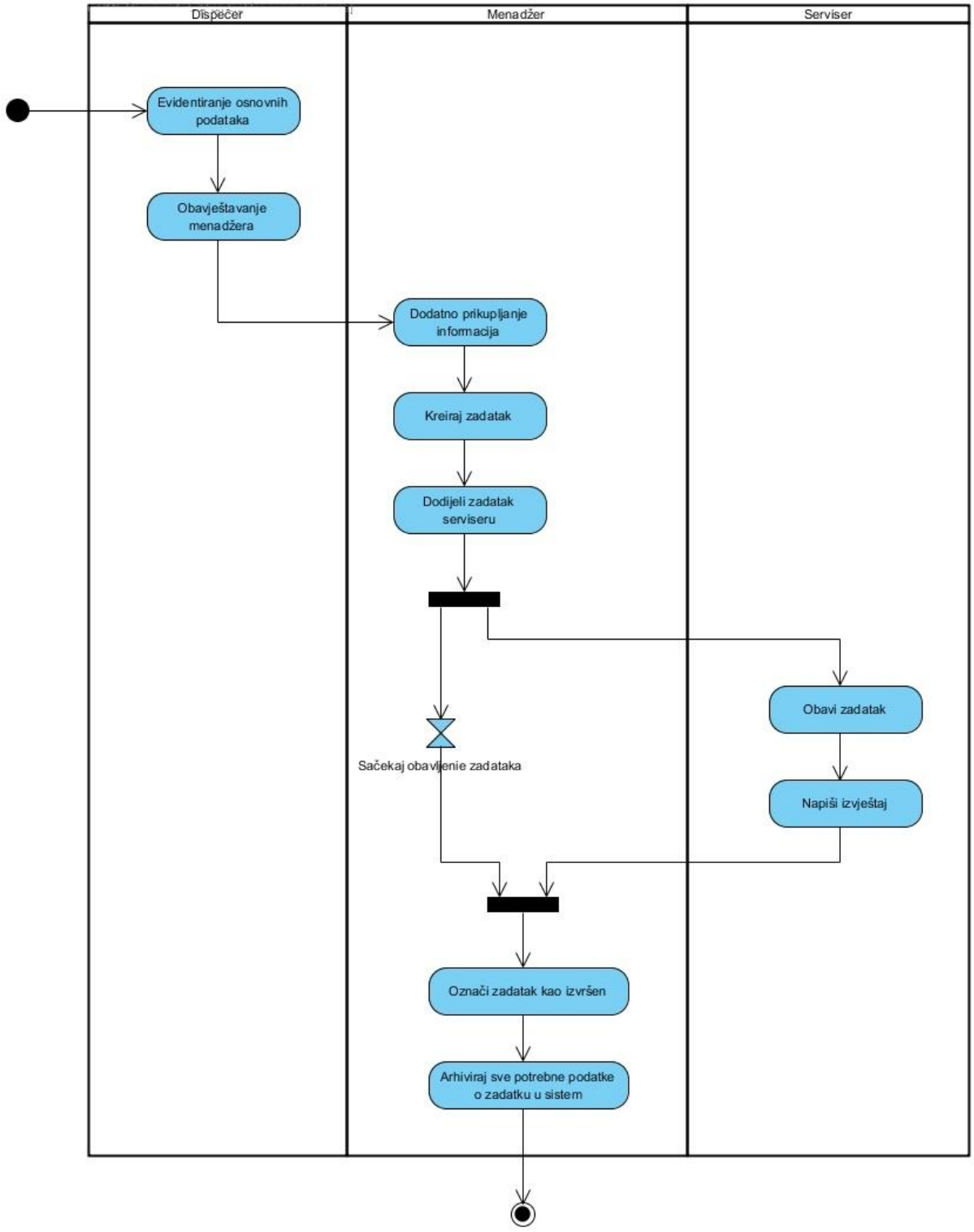
Definicija ovog paterna je : Command pattern enkapsulira zahtjev kao objekat, te mu tako dopušta da parametrizira ostale objekte sa različitim zahtjevima.

Njegovu primjenu na dijagram klasa možemo vidjeti na slici 1. Primijenjeno na naš sistem (sa slike 1) invoker je u našem sistemu dispečer koji kreira komandu, u našem slučaju zahtjev pozivom metode preuzmiZahtjev(). Zahtjev se poziva kroz njegovu metodu izvrsi(). Iz komande je naslijeđena konkretna komanda, u našem slučaju zadatak, koji je poveznica između akcije, servisiranja, i primaoca, servisera. Dakle, **dispečer pravi zahtjev pozivajući metodu izvrsi() a zadatak to ostvaruje pozivajući jednu ili više akcija nad serviserom.**Pozivom metode izvrsi() klase zadatak zapravo pozivamo primaoca, u našem slučaju servisera da izvrši zadatak, tj da servisira uređaj. Klijent je i u našem sistemu klijent. On je odgovoran za iniciranje zadatka i primanje povratne informacije od servisera.



Slika 7 – Command pattern u dijagramu klasa

1.3.4. Activity dijagram

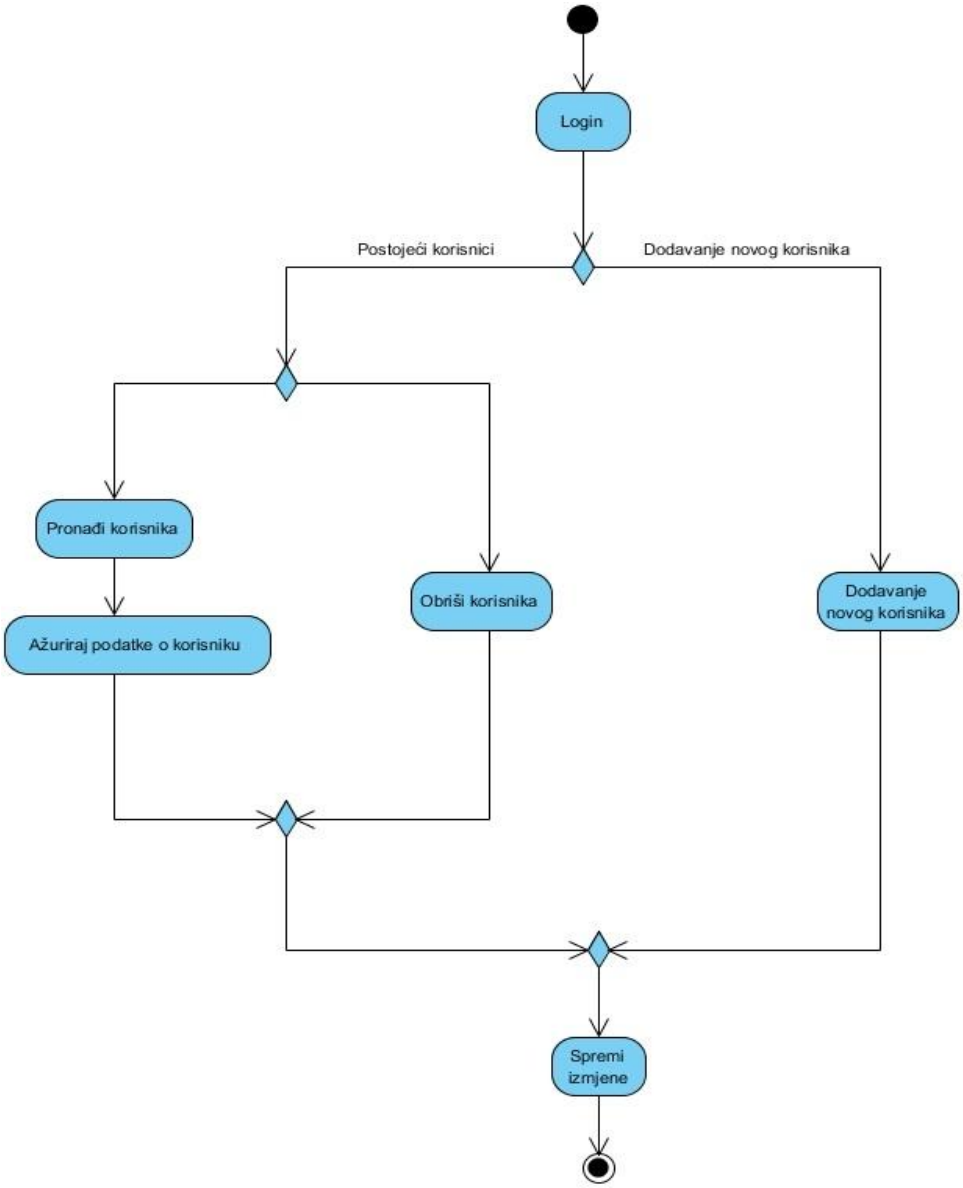


Slika 8 – Activity dijagram odvijanja obrade zadatak servisiranja

Opis : Kada neki klijent nazove sistem i da zahtjev za servisiranje ili popravku nekog uređaja prvo što se uradi jeste evidentiranje njegovih osnovnih podataka, te nekih osnovnih podataka o šifri i tipu uređaja, hitnosti itd. Dispečer na osnovu tih podataka kreira zahtjev i prosljeđuje ga menadžeru. Menadžer taj zahtjev detaljnije pregleda i informiše se detaljnije o kvaru i tipu

uređaja te na kraju kreira zadatak sa svim potrebnim informacijama. Zadatke raspoređuje serviserima koji obavljaju svoje zadatke. Menadže nastavlja da radi svoj posao dok mu neki serviser ne dostavi potvrdu da je završio svoj zadatak i donese izvještaj o kvaru i trajnju i koštanju popravke na papiru. Menadžer evidentira sve potrebne podatke u sistem i označava zadatak kao završen te ga arhivira. Time se završava proces obrade zadataka.

1.3.4.1. Detaljniji pogled – Activity dijagram rada administratora

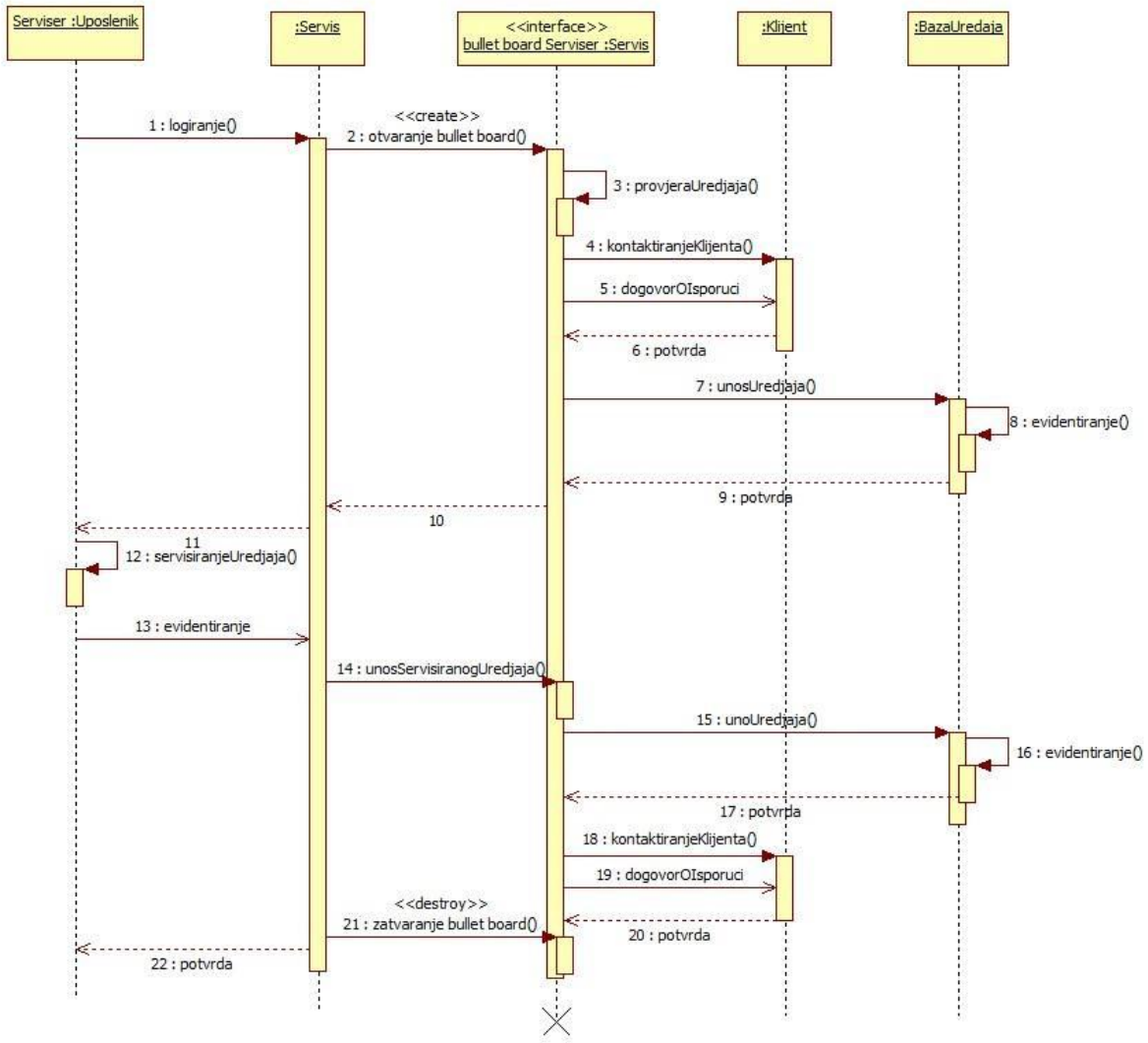


Slika 9 – Command pattern u dijagramu klasa

Opis : Kao što smo već ranije naveli administrator ima jednu od najvažnijih uloga u našem sistemu.Uloga administratora je da se pobrine za pravilno funkcionisanje sistema i svih funkcionalnosti koje sistem ima. Jedna od osnovnih funkcija je rad sa korisničkim računima sto podrazumijeva dodavanje, brisanje, ažuriranje i pregled korisničkih računa. Da bi se uopšte moglo pristupiti upravljanju korisničkih računa osoba se mora ulogovati kao administrator jer samo administrator ima mogućnost izmjene, dodavanja i brisanja korisnika. Nakon uspješnog

logina administrator ima mogućnost odabira dodavanja novog korisničkog računa ili rada sa postojećim korisničkim računima. Ukoliko izabere da doda novog korisnika potrebno je da unese neke lične informacije kao što su ime, prezime, mjesto i godina rođenja, JMBG itd kao i postavljanje username-a i password-a. Nakon unošenja svih potrebnih informacija vrši se spremanje korisničkog računa u bazu podataka našeg sistema i time je dodavanje završeno. Ukoliko ipak odabere mogućnost da radi sa postojećim korisničkim računima nudi mu se mogućnost pregleda postojećih te njihovo mijenjanje ili brisanje.

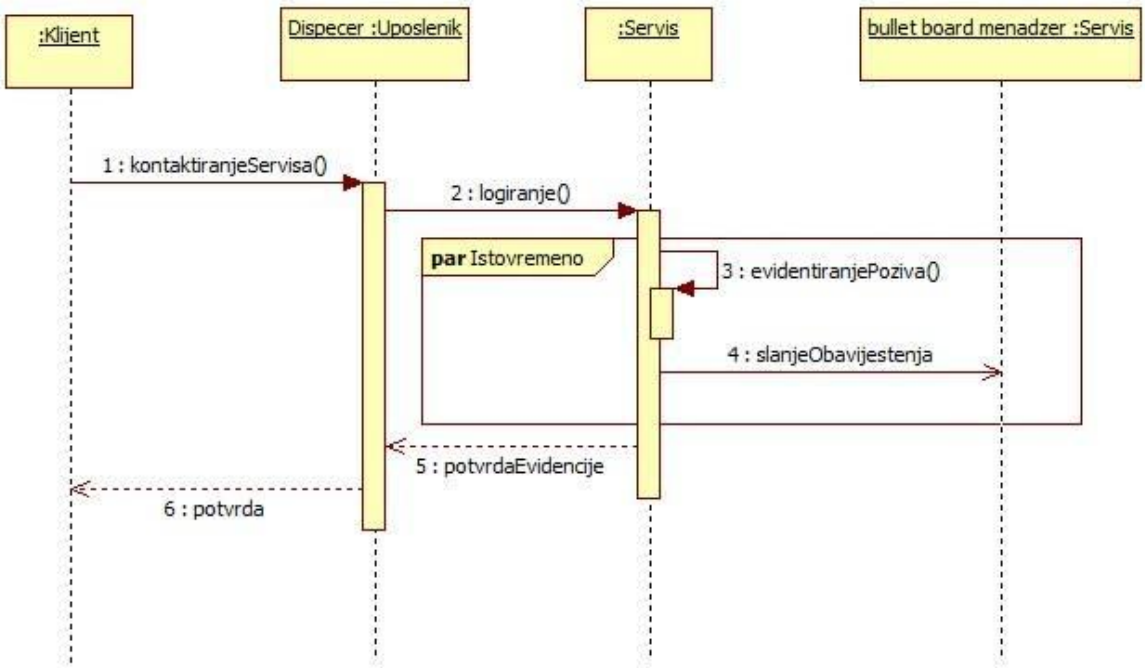
1.3.5. Dijagram sekvenci



Slika 10 – Dijagram sekvenci sistema

Opis : Serviser se logira na sistem sa svojim imenom i lozinkom. Nakon što se logira, sistem otvara korisnički interfejs specifičan za servisera, sa njegovim „bullet board-om“. Na „bullet board-u“ vidi izlistane uređaje po koje treba otići kod klijenta. Nakon toga, serviser kontaktira klijenta i ugovara vrijeme kada može otići po uređaj. Nakon što preuzme uređaj od klijenta, evidentira preuzimanje putem svog „bullet board-a“ u bazu podataka. Nakon toga, serviser servisira uređaj, te nakon što završi posao, evidentira u bazi podataka da je servisiranje završeno. Nakon toga, kontaktira klijenta i ugovara vrijeme isporuke servisiranog uređaja. Nakon što serviser isporuči uređaj klijentu, evidentira tu promjenu u bazu podataka.

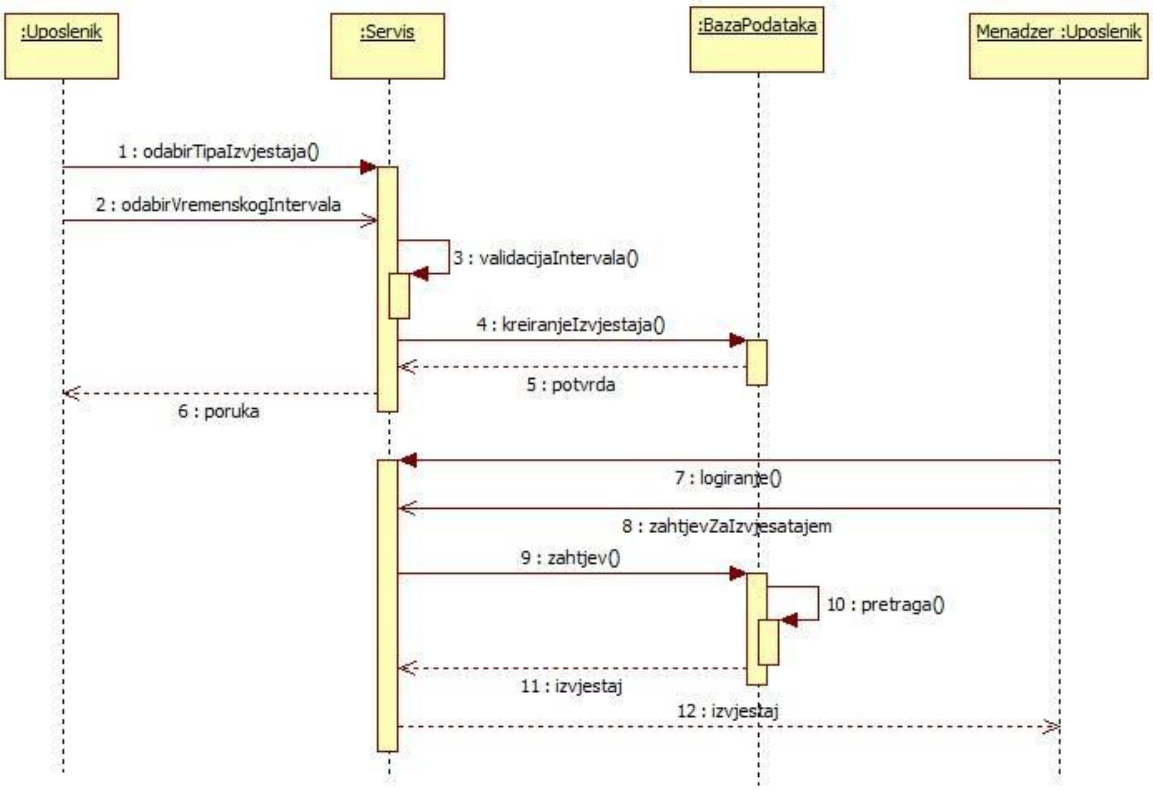
1.3.5.1. Detaljniji pogled – Kontaktiranje servisa



Slika 11 – Detaljniji dijagram sekvenci – Kontaktiranje servisa

Opis : Uposlenik vrši logovanje sa svojim username i password-om da bi pristupio svom bullet board-u .Vrši se provjera da li ima novih uređaja za servisiranje.Nakon što je uređaj evidentiran kontaktira se klijent koji je vlasnik tog uređaja i sa njim se dogovaraju detalji o isporuci istog.Klijent šalje potvrdu i vrši se unos uređaja u bazu podataka gdje se vodi evidencija o uređaju i nakon čega se potvrda šalje iz baze podataka na serviserov bullet board.Serviser zatim pristupa popravci tog uređaja evidentirajući pri tome sve promjene koje je izvršio i kvarove koje je otklonio.Podatke o servisiranom uređaju upisuje u svoj bullet board čime vrši unos uređaja u bazu podataka iz koje stiže potvrda da je uređaj uredno evidentiran.Kada je potvrda o popravci uređaja stigla kontaktira se klijent da bi se dogovorili detalje o preuzeću uređaja.Potvrdom klijenta o preuzimanju zatvara se bullet board i sam serviser dobiva potvrdu o završenom poslu.

1.3.5.2. Detaljniji pogled – Kreiranje izvještaja



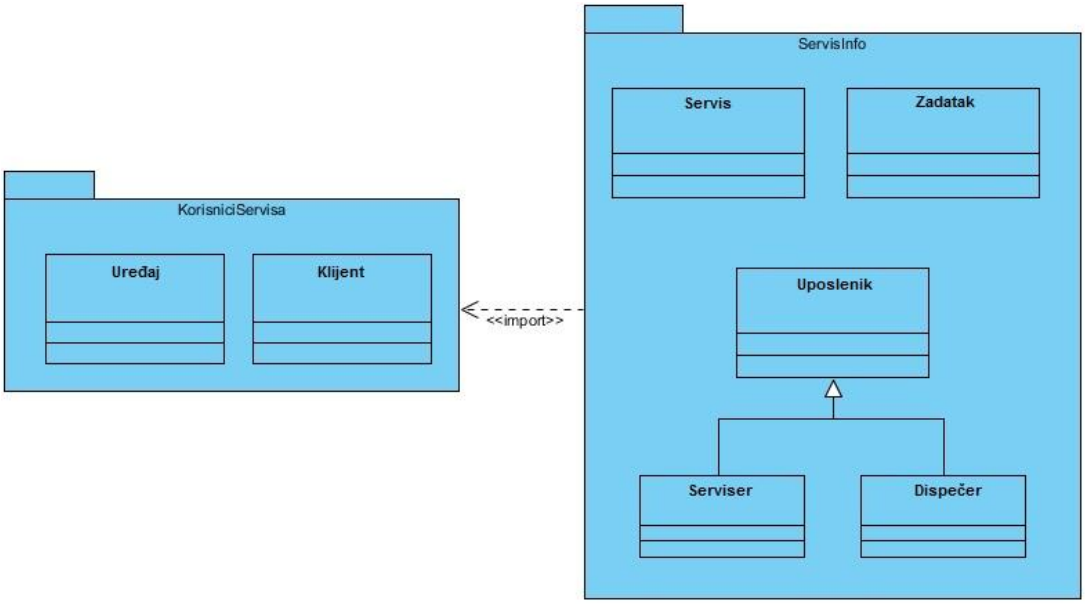
Slika 12 – Detaljniji dijagram sekvenci – Generisanje izvještaja

Opis : Uposlenik odnosno serviser kreira izvještaj i unosi vremenski interval u kojem uređaj treba da bude servisiran.Zatim se vrši validacija odnosno provjera tog vremenskog vremenskog intervala .Ukoliko su ispunjeni uslovi dolazi do kreiranja izvještaja koji se upisuje u bazu podataka našeg sistema.Serviser biva obaviješten o stanju kreiranja njegovog izvještaja tj. da li je izvještaj uspješno kreiran ili ne.

Menadžer da bi uživao privilegije koje odgovaraju njegovom statusu u firmi neophodno je da se uloguje.Nakon uspješnog logina salje servisu zahtjev za određenim izvještajem.Servis njegov upit prosljeđuje u bazu podataka u kojoj se vrši pretraga.

U slučaju uspješne pretrage izvještaj se proslijeđuje servisu nakon čega servis isti taj izviještaj šalje menađeru.

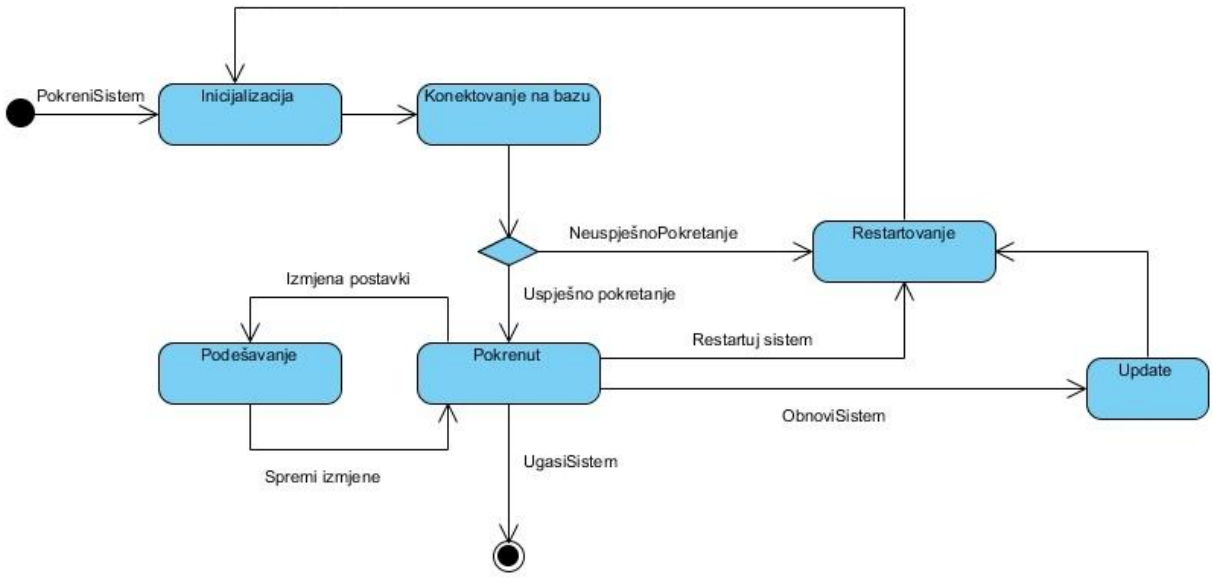
1.3.6. Dijagram paketa



Slika 13 – Dijagram paketa sistema

Opis : Ovaj dijagram nam ustvari predstavlja načina na koji su dijelovi sistema ili tačnije rečeno klase grupisane. Način grupisanja predstavljen na slici programiranjem u Javu ćemo ostvariti svrstavanjem klasa u pojedine namespace-e. Postojat će dva namespace sa nazivim KorisniciServisa i ServisInfo. ServisInfo će uključivati ovaj drugi namespace za rad samog sistema su potrebne informacije o klijentima i uređajima.

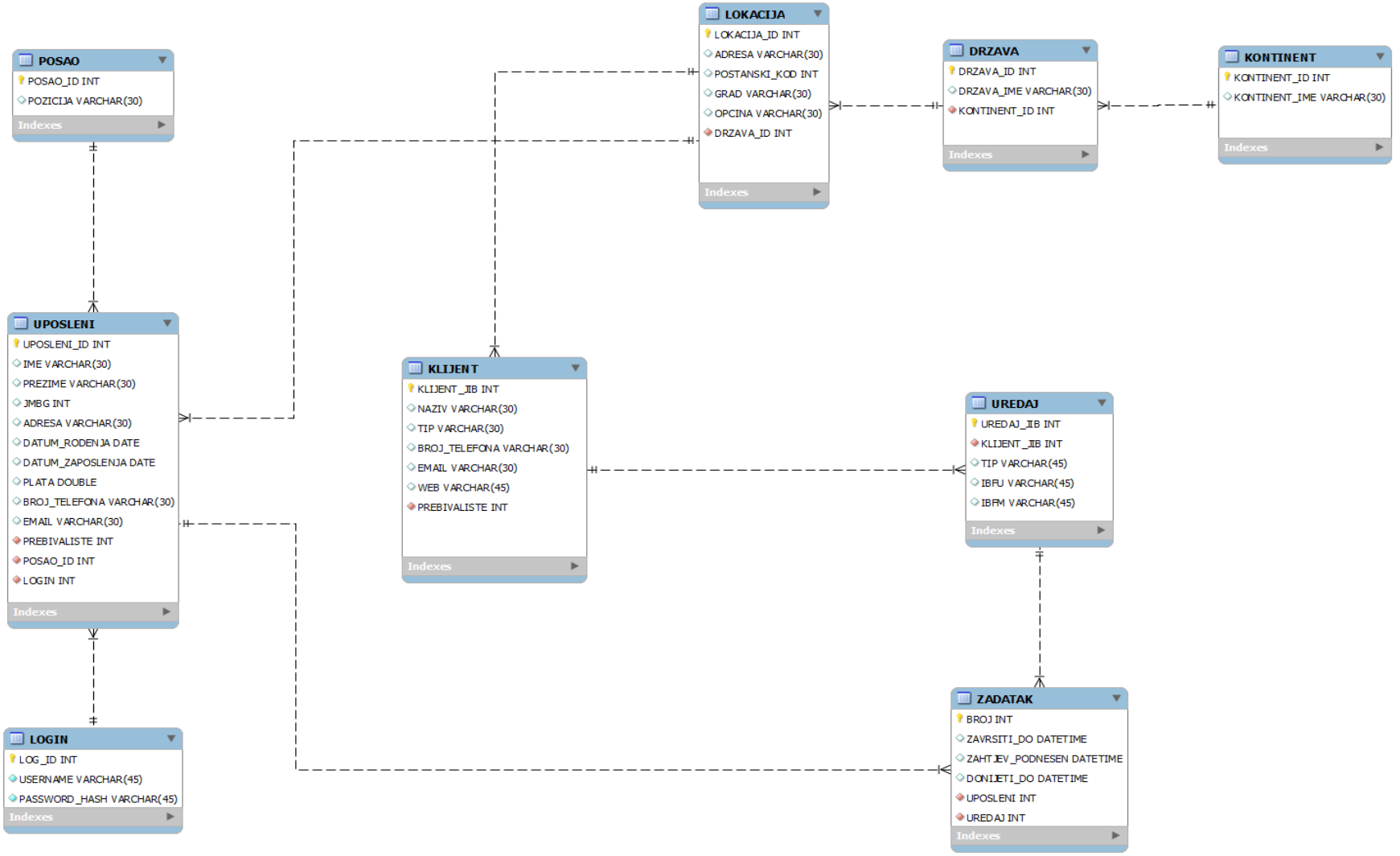
1.3.7. Dijagram stanja



Slika 14 – Dijagram stanja sistema

Opis : Na ovom dijagramu se vide vide stanja u kojima se naš sistem može nalaziti i akcije koje uzrokuju prelazak iz jednog stanja u drugo. Pošto je sistem uglavnom fokusiran na evidenciju podataka, ovaj dijagram predstavlja više ponašanje cjelokupnog sistema, a ne same desktop aplikacije. Kada se sistem pali prvo se mora obaviti incijalizacija tj. kompjuter mora odraditi svoje, nakon tih uvodnih aktivnosti se program pokušava konektovati na bazu podataka, ako te ne uspije sistem se ponovno pokreće i pokušava ponovo. Ako se uspješno konektuje sistem je u radnom stanju i spreman je za izvršavanje akcija. Određena podešavanja se mogu uraditi online i nakon promjena sistem zapamti te postavke. Kada je potrebno obnoviti sistem na noviju verziju tada je neizbježno ponovno pokretanje sistema i njegovo isključivanje iz upotrebe na određeni period vremena koji je specificiran u nefunkcionalnim zahtjevi.

1.4. Dizajn baze podataka



Slika 15 – ER dijagram baze podataka za FDSS