



UNIVERZITET SINGIDUNUM
DEPARTMAN ZA POSLEDIPLOMSKE STUDIJE
-STUDIJSKI PROGRAM-

Projektovanje informacionog sistema zdravstvene ustanove

- Master rad -

Mentor:
Prof dr Mladen Veinović

Student:
Miša Blagojević
Br. indeksa: 410074/2013.

Projektovanje informacionog sistema zdravstvene ustanove

Sažetak: U ovom radu razmatra se projektovanje i tehnička primena projektne dokumentacije. Na početku su prikazane metode projektovanja i modelovanja informacionog sistema. Zatim su prikazane osnove C# programskog jezika, ASP .NET MVC platforme, kao i funkcionalnosti informacionog sistema. U radu su prikazane instalacije neophodnih delova sistema na način na koji se obezbeđuje pravilno funkcionisanje sistema kao celine, sa korišćenjem sertifikata radi obezbeđivanja povećane sigurnosti veze. Prikazano je pravilno korišćenje sistema iz uloge korisnika sistema, kao i demonstracija načina komunikacije između delova sistema kroz karakteristične zahteve.

Ključne reči: Projektovanje informacionog sistema, Tehnička primena informacionog sistema, C#, ASP .NET MVC, CA, Pravilni rad u sistemu

Designing information systems for healthcare facilities

Abstract: In this paper is discussed designing and the technical implementation of the project documentation. At the beginning are presented methods of design and modeling of information systems. Then is shown familiarization with the basics of C # programming language, ASP. NET MVC platform, as well as learning about the functioning of the information system. The paper presents the installation of the necessary parts of the system in a way that secures proper functioning of the system as a whole with the use of certificates for increased security of connections. It is shown the proper use of the system for the aspects of the user roles, as well as a demonstration of the ways of communication between the system components through distinctive requirements

Keywords: Designing information systems, The technical implementation of the information system, C#, ASP .NET MVC, CA, Proper opration in the system

SADRŽAJ

1. METODOLOGIJA ISTRAŽIVAČKOG PROJEKTA	6
1.1. Uvodne napomene i obrazloženje rada	6
1.2. Predmet istraživanja	6
1.3. Ciljevi i zadaci istraživanja	6
1.4. Istraživačke hipoteze	6
1.5. Metodi istraživanja i tok istraživačkog procesa	7
2. SISTEMSKO STRUKTURNA ANALIZA	8
2.1. Verbalni opis sistema	9
2.2. Dijagram konteksta	10
2.3. Dijagram dekompozicije prvog nivoa	12
2.4. Dijagram dekompozicije drugog nivoa	12
2.5. Dijagram dekompozicije trećeg nivoa	12
3. SLUČAJEVI KORIŠĆENJA	13
3.1. Slučajevi korišćenja za ulogu „Klijent“	14
3.2. Slučajevi korišćenja za ulogu „Lekar“	16
3.3. Slučajevi korišćenja za ulogu „Medicinska sestra“	19
3.4. Slučajevi korišćenja za ulogu „Administracija“	22
3.5. Slučajevi korišćenja za ulogu „Dobavljač“	26
4. DIJAGRAMI SEKVENCI	28
4.1. Dijagrami sekvenci za ulogu „Klijent“	29
4.2. Dijagrami sekvenci za ulogu „Lekar“	31
4.3. Dijagrami sekvenci za ulogu „Medicinska sestra“	31
4.4. Dijagrami sekvenci za ulogu „Administracija“	32
4.5. Dijagrami sekvenci za ulogu „Dobavljač“	33
5. DIJAGRAMI KOLABORACIJE	34
5.1. Dijagrami kolaboracije za ulogu „Klijent“	35
5.2. Dijagrami kolaboracije za ulogu „Lekar“	36
5.3. Dijagrami kolaboracije za ulogu „Medicinska sestra“	36
5.4. Dijagrami kolaboracije za ulogu „Administracija“	36
5.5. Dijagrami kolaboracije za ulogu „Dobavljač“	37
6. DIJAGRAMI AKTIVNOSTI	38
6.1. Dijagram aktivnosti za ulogu „Klijent“	39
6.2. Dijagrami aktivnosti za ulogu „Lekar“	40
6.3. Dijagram aktivnosti za ulogu „Medicinska sestra“	41
6.4. Dijagrami aktivnosti za ulogu „Administracija“	42
6.5. Dijagrami aktivnosti za ulogu „Dobavljač“	43
7. DIJAGRAMI STANJA	44
7.1. Dijagrami stanja za ulogu „Klijent“	45
7.2. Dijagrami stanja za ulogu „Lekar“	46
7.3. Dijagrami stanja za ulogu „Medicinska sestra“	47
7.4. Dijagrami stanja za ulogu „Administracija“	48
7.5. Dijagrami stanja za ulogu „Dobavljač“	49
8. DIJAGRAM KLASA	50
9. DIJAGRAM RASPOREDA	51
10. MODEL OBJEKTI I VEZE	53

10.1.	DIJAGRAM MODEL OBJEKTI I VEZE BAZE PODATAKA ZA ČUVANJE RADNIH PODATAKA	54
10.2.	DIJAGRAM MODEL OBJEKTI I VEZE BAZE PODATAKA ZA ČUVANJE BEZBEDNOSNIH PODATAKA.....	55
11.	RELACIONI MODEL.....	56
11.1.	REALCIONI MODEL – PRVA NORMALNA FORMA	57
12.	REČNIK PODATAKA	70
12.1.	REČNIK PODATAKA BAZE ZA RAD	70
13.	C#	71
14.	ASP .NET MVC.....	73
14.1.	MVC PROJEKTNI ŠABLON	73
15.	TEHNOLOŠKI SASTAV INFORMACIONOG SISTEMA.....	74
15.1.	SERVER BAZE PODATAKA	74
15.2.	SERTIFIKACIONO TELO	76
15.3.	API SERVER MODULI	76
15.4.	ASP SERVER MODULI.....	77
15.5.	MODULI.....	78
16.	INSTALACIJA	88
16.1.	SERTIFIKACIONO TELO (CA)	88
16.2.	STVARANJE ULOGE NA SERVERU BAZE PODATAKA.....	88
16.3.	INSTALACIJA INTERNET INFORMATION SERVICES (IIS).....	88
16.4.	INSTALACIJA ASP I API SERVERA	88
16.5.	SLANJE ZAHTEVA, IZDAVANJE I UVOZ SERTIFIKATA	89
17.	NAČIN RADA U INFORMACIONOM SISTEMU	89
18.	KARATERISTIČNE PUTANJE NAREDBI U INFORMACIONOM SISTEMU.....	90
18.1.	PODNOŠENJE ZAHTEVA ZA ULOGU „PACIJENT“	90
18.2.	PRIJAVA NA SISTEM	102
18.3.	UNOS DIJAGNOZE.....	108
18.4.	IZMENA DIJAGNOZE	111
18.5.	PRIKAZ LISTE DIJAGNOZA	114
19.	ZAKLJUČAK	118
	LITERATURA.....	120
	SPISAK SLIKA.....	121
	SPISAK TABELA	121

Uvod

U ovom radu biće reći o prikazu projektne dokumentacija kao i o njenom tehničkom izvršenju. Na početku se obrađuje analiza korisničkih zahteva i njihov prikaz kroz sistemsko strukturnu analizu i ostale dijagrame. Dijagrami pored prikaza korisničkih zahteva prikazuju sastav određenih komponenti. Neophodnih za normalno functionisanje sistema kao celine. Pre konkretnih upustava za tehničko izvršavanje projektne dokumentacije, biće reći o C# programskom jeziku, ASP .NET MVC platformi, načinima komunikacije između delova informacionog sistema kao i njihovog zasebnog objašnjavanja. Pored upustva za tehničko izvršavanje projektne dokumentacije biće reći i o pravilnom korišćenju sistema i demonstrativnom prikazu načina komunikacije između delova informacionog sistema

Rad je podeljen na više celina. Svaka celina poželjno je da se shvati radi lakšeg razumevanja sledećih celina. Prve celine se bave analiziranjem i projektovanjem korisničkih zahteva pomoću sistemsko strukturne analize, slučajeva korišćenja i ostalih, prisutnih dijagrama. Posle toga prva sledeća celina se bavi objašnjavanjem C# programskog jezika u kratkim crtama. Celina koja sledi, takođe u kratkim crtama objašnjava način funkcionisanja ASP .NET MVC platforme kao i smisao MVC projektnog šablona. Vršiti se upoznavanjem sa načinom komunikacije između delova informacionog sistema. Prikazuju se neophodne instalacije delova sistema sa ciljem ispravnog funkcionisanja informacionog sistema u celini. Pretposlednja celina objašnjava pravilan način korišćenja sistema. Poslednja celina ima zadatak da prikaže način komunikacije između delova sistema kroz karakteristične zadatke.

1. Metodologija istraživačkog projekta

1.1. Uvodne napomene i obrazloženje rada

Master rad je imao svrhu formiranja odgovarajuće dokumentacije potrebne za projektovanje informacionog sistema, kao i prikaz njene tehničke implementacije.

Projektni zadatak je izrađen korišćenjem teorijskih i praktičnih znanja iz domena baza podataka i informacionih sistema.

Projektna dokumentacija informacionog sistema, sistem ne ograničava na određenu tehnologiju ili tehnološki proces, već definiše okvire u kojima se obezbeđuje što potpunije ispunjavanje korisničkih zahteva.

Tehnološka implementacija nije ograničila sistem i ne otežava njegovu nadogradnju.

1.2. Predmet istraživanja

Predmet istraživanja je bilo projektovanje i implementacija informacionog sistema u zdravstvenoj instituciji. Projektovanje informacionog sistema je moralo da bude takvo da u dokumentaciji budu prisutni skoro svi bitni aspekti već postojećeg sistema iz doba pre interneta. Pored obezbećivanja saobraznosti sa prethodnim sistemima, informacioni sistem morao je da pruži i modularnost. Modularnost u ovom smislu znači lakoću ubacivanja novih, i izbacivanja prevaziđenih modula poslovanja u informacionom sistemu, bez gašenja celog sistema već samo modula na kome se vrši intervencija. Implementacija projektne dokumentacije je trebalo da bude bazirano na tehnologijama koje su srodne i koje ne otežavaju međusobnu komunikaciju, ali omogućavaju zadovoljavajući stepen zaštite podataka.

1.3. Ciljevi i zadaci istraživanja

Informacioni sistem treba da obezbedi što lakši rad u njemu i za klijente kao i za osoblje. Cilj tog zahteva sastoji se usmeravanju pažnje pacijenata i osoblja na probleme pacijenata, a ne na rad u samom informacionom sistemu.

Neki od zadataka pri projektovanju sistema su pored lakoće rada u njemu bili i brzina rada. U vremenu globalizacije jedan od bitnijih zadataka je omogućavanje nekim od korisnika, kao što je pacijent, pristup informacionom sistemu sa udaljenih računara i obezbeđivanja prihvatljivog nivoa bezbednosti.

1.4. Istraživačke hipoteze

Prvobitno težište u projektovanju i implementaciji informacionog sistema je bilo u sistemu kome bi se komunikacija obavljala u okviru jedne organizacije. Organizacija može da se nalazi na jednoj ili više lokacija.

Tokom istraživanja dolazi se do zaključka da organizacijama je značajno da se omogući udaljen pristup određenim tipovima korisnika. Pored toga jedan od zahteva je da funkcionisanje informacionog sistema bude brzo i lako.

Problem sa time je sadržan u bezbednosti koja je morala da bude unapređena korišćenjem šifrovanja odrađenih sadržaja i korišćenjem dodatnih uređaja kao što je smart kartica, kao i obezbeđivanjem sigurnih veza.

Brzina i lakoća u korišćenju informacionog sistema zavisi od tehnoloških rešenja koja se primenjuju u korišćenju informacionog sistema. Pored tehnoloških rešenja brzina i lakoća rada u sistemu zavise i od logičkog rasporeda procesa koji su potrebni da bi se odrađeni zahtev korisnika obavio. Pored navedenih faktora takođe je bitna i usklađenost sa prethodnim sistemom koji je postojao u organizaciji ili se primenjuje u funkcionisanju drugih organizacija u istoj tržišnoj grani.

Pri projektovanju i implementaciji informacionog sistema akcenat je bio na rasporedu procesa koji bi obezbeđivao najveću moguću brzinu obavljanja zahteva sa održavanjem definisanog nivoa bezbednosti i stabilnosti sistema i podataka u njemu.

Uzeto je u obzir funkcionisanje informacionih sistema u drugim organizacijama, kao prethodnim verzijama takvih sistema. Kao model uzet je informacioni sistem u zdravstvenoj ustanovi koju finansira država.

1.5. Metodi istraživanja i tok istraživačkog procesa

Metodologija istraživanja sastojala se iz tumačenja većeg broja dokumenata, kao što su zdravstveni kartoni, nalozi i slično. Pored analize dokumenata, vršeni su razgovori sa zaposlenima u vezi iskustva sa radom u postojećim sistemima, kao i sa načinom njihovog rada u prethodnim verzijama sistema.

Pri projektovanju i implementaciji informacionog sistema korišćena su iskustva i saveti [prof. dr. Mladena Veinovića](#).

Korišćene su knjige u kojima se bavi obrada materije projektovanja i implementacije informacionog sistema. Takođe su korišćene knjige koje obrađuju materiju projektovanja baza podataka.

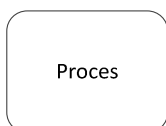
2. Sistemsko strukturna analiza

Sistemsko strukturna analiza predstavlja metodologiju za specifikaciju informacionog sistema. Sistemsko strukturna analiza ili skraćeno SSA ima ulogu da predstavi specifikaciju korisnika za određenim informacionim sistemom na tačan i potpun način. Razlog je u tome jer je ona početna tačka u projektovanju informacionog sistema. Sistemsko strukturna analiza sastoji se iz dva dela. Jedan deo je verbalni opis sistema, dok su drugi deo dijagrami tokova podataka ili skraćeno DTP.

DTP opisuju protok informacija u sistemu. Sistemsko strukturna analiza, posmatranjem kroz njen drugi deo, posmatra sistem kao skup procesa koji imaju svoje ulazne tokove. Uz pomoć ulaznih tokova procesi generišu odgovarajuće izlaze. Tvorci ulaznih tokova mogu da budu interfejsi, ali takođe izvori mogu biti i drugi procesi kao i skladišta podataka.

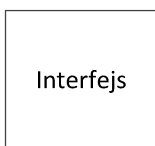
Korišćena legenda označavanja u SSA, uz pomoć alata Microsoft Visual Studio 2010 je:

- Proces:



- Proces služi za obradu tokova podataka i pružanje izlaznog toka kao rezultat procesa. Mogu da se tretiraju i kao tvorci tokova podataka

- Interfejs:



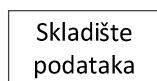
- Interfejsi služe kao izvori odnosno tvorci tokova podataka. Predstavljaju entitete sa kojima informacioni sistem sarađuje.

- Tok podataka:



- Predstavlja posrednik koji uvek nosi podatke u sebi u obradu u procese, iz procesa u skladišta podataka, iz skladišta podataka u procese ili iz procesa do interfejsa.

- Skladište:



- Može da se posmatra kao entitet koji u sebi ima akumulirane tokove podataka. Kao i procesi i interfejsi, i skladišta takođe mogu da budu izvorišta tokova podataka.

Većina informacionih sistema su kompleksne strukture sa više slojeva. Sa ciljem održavanja jasnosti funkcionisanja takvog jednog složenog entiteta, neophodno je da se on dekomponuje.

Početni dijagram od koga se počinje dekompozicija funkcionisanja jednog informacionog sistema je dijagram konteksta (Slika 1.).

Daljom dekompozicijom dijagrama konteksta se dobijaju dijagrami dekompozicije n nivoa. Slovo n je broj koji predstavlja nivo dekompozicije do kog se došlo u razlaganju funkcionisanja jednog informacionog sistema.

Dekompozicija treba da se obavlja sve dok svi procesi ne postanu primitivni, tj. da ne mogu dalje da se dekomponuju jer već predstavljaju jednostavne(atomske) operacije.

2.1. Verbalni opis sistema

Preduzeće se bavi pružanjem zdravstvenih usluga zainteresovanim korisnicima. Prva aktivnost koja mora da bude izvršena je otvaranje korisničkog naloga, bez obzira na ulogu (zaposleni ili klijent). Klijent je u obavezi da po otvaranju korisničkog naloga izabere traženog doktora.

Kada se pojavi potreba kod klijenta, klijent zakazuje, uz pomoć medicinske sestre pregled.

Klijent je u obavezi da naznači koji je tip pregleda, tj. da li je preventiva(sistematski pregled) ili kurativa. U slučaju da je kurativa u pitanju, klijent je u obavezi da pruži anamnezu svog problema. Ukoliko doktor koji je izabran nije na raspolaganju u trenutku dolaska klijenta, klijent može da bira da bude pregledan kod drugog doktora ili može da zakaže kod svog izabranog doktora. Medicinska sestra po prikupljanju dovoljno podataka od klijenta, nalog klijenta postavlja u listu čekanja za pregled.

Kada dođe na red klijentov nalog, ukoliko je slučaj da je kurativa, klijent biva pregledan od lekara koji izvodi nalaz pregleda i donosi dijagnozu. U zavisnosti od dijagnoze, donosi se odgovarajuća terapija, propisuju injekcije, izdaju nalozi za pomagala ili za druge ustanove radi daljih ispitivanja i lečenja. U slučaju da je u pitanju preventiva, lekar vrši odgovarajući pregled i zapisuje svoja zapažanja u karton klijenta. Ukoliko su zapažanja pri pregledu normalna ne izdaju se nalozi za daljim ispitivanjima, a ako to nije slučaj, onda se pacijent svrstava u listu za kurativu i vrše se dalji pregledi.

Po okončanju pregleda, i eventualnog izdavanja terapije i naloga, vrši se obračun pruženih usluga. Klijent može usluge da plati u više rada ili odjednom. Ukoliko postoje neizmirena zaduženja koja nisu namirena u predviđenom vremenskom roku, na njihove iznose teče kamata. Ukoliko klijent ne izmiri svoja zaduženja u određenom vremenskom roku, njegov nalog se suspenduje i on ne može više da se leči u datom preduzeću sve dok se zaduženja ne izmire.

Klijentov nalog može da bude suspendovan i zbog nepoštovanja pravila ponašanja u preduzeću.

Zaposleni mogu da vrše sledeće uloge: Lekar, Medicinska sestra i Administracija.

Najvažnije uloge zaposlenih su:

- Medicinska sestra je zadužena za unos klijenta na liste čekanja, obračun pruženih usluga, uvođenje u sistem uplata od klijenta i slično.
- Lekar je zadužen za dijagnostikovanje problema kod pacijenta shodno prethodno datoj anamnezi i sveukupnom stanju pacijenta. Ovlašćen je da odobrava bolovanja klijentima.
- Administracija je zadužena za kontrolu zaposlenih, obračun njihovih primanja, rešavanje zahteva zaposlenih za godišnjim odmorima. Administracija poseduje ovlašćenja da zapošljava nove zaposlene, suspenduje naloge postojećih zaposlenih kao i da vrši prekide radnih odnosa.

2.2. Dijagram konteksta

Slika 1 Dijagram konteksta

IS Zdravstvo je predviđen da radi sa klijentima koji mogu da budu samo fizička lica, zaposlenima. koji shodno ovlašćenjima mogu da budu lekari, medicinske sestre ili da su zaposleni u administraciji i dobavljačima koji mogu da budu samo pravna lica. Primećuje se da na dijagramu postoje pet objekata, a to su klijent, lekar, medicinska sestra, administracija i

dobavljač. Svi zahtevi i akcije su regulisani tokovima podataka, koji su pokazatelji kako objekti komuniciraju sa sistemom.

2.3. Dijagram dekompozicije prvog nivoa

Dijagrami prvog nivoa su nastali dekompozicijom kontekstnog dijagrama (slika 1). Iz razloga velikog broja tokova podataka, prvi nivo se sastoji od više pod dijagrama koji detaljnije i na lakši način vizuelno prikazuju učesnike, odgovarajuće tokove podataka, procese kao i skladišta podataka. Detalji dijagrama dekompozicije prvog nivoa nalaze se u prilogu dijagrami dekompozicije prvog nivoa.

2.4. Dijagram dekompozicije drugog nivoa

Dijagram drugog nivoa je nastao dekompozicijom dijagrama dekompozicije prvog nivoa. Iz razloga velikog broja tokova podataka, drugi nivo, kao i prvi nivo pre njega, podeljen je na više pod dijagrama koji detaljnije i na lakši način vizuelno prikazuju učesnike, odgovarajuće tokove podataka, procese i skladište podataka. Detalji dijagrama dekompozicije drugog nivoa nalaze se u prilogu dijagrami dekompozicije drugog nivoa.

2.5. Dijagram dekompozicije trećeg nivoa

Dijagram trećeg nivoa je nastao dekompozicijom dijagrama dekompozicije drugog nivoa. Iz razloga velikog broja tokova podataka, treći nivo, kao i nivoi pre njega, podeljen je na više pod dijagrama koji detaljnije i na lakši način vizuelno prikazuju učesnike, odgovarajuće tokove podataka, procese i skladište podataka. Detalji dijagrama dekompozicije trećeg nivoa nalaze se u prilogu dijagrami dekompozicije trećeg nivoa

3. Slučajevi korišćenja

Dijagram slučajeva korišćenja predstavlja skup slučajeva korišćenja i aktera. Dijagram slučajeva korišćenja predstavlja grafički prikaz skupa svih slučajeva korišćenja u sistemu ili delu sistema. Obično se koristi da prikaže neku funkcionalnost sistema i ponašanje korisnika u manifestaciji te funkcionalnosti. Dijagram prikazuje ponašanje sistema ili podsistema.

Uloga dijagrama slučaja korišćenja je da omogući korisniku da razume šta sistem radi kao i da proveri funkcionisanje sistema.

Pojedinačni slučajevi korišćenja predstavljaju opis skupa postupaka, uključujući i alternativne koje sistem obavlja da bi proizveo određeni rezultat za korisnika sistema. Bitno je da se shvati da slučaj korišćenja određuje šta sistem radi ali ne i kako će sistem to sprovesti u delo.

U slučajevima korišćenja postoje osnovni i alternativni scenariji po kojima se odvijaju tokovi događaja. Osnovni scenariji se odvijaju u slučajevima kada ne dolazi do grešaka u radu sistema. Kada do grešaka dođe aktiviraju se alternativni scenariji koji nastavljaju da izvedu na dalje slučaj korišćenja.

Legenda koja je korišćena u projektovanju slučajeva korišćenja, nastala je pomoću alata Visio 2010 kompanije Microsoft

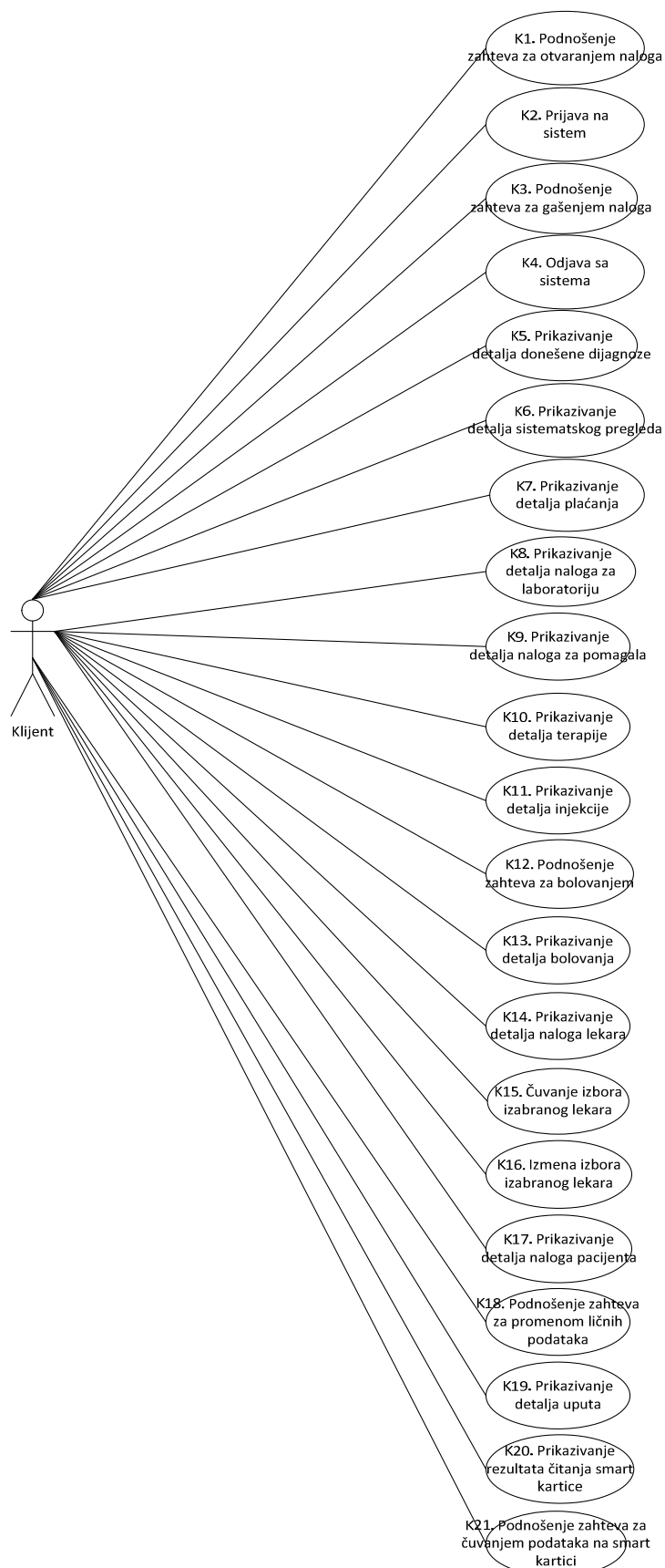


- Učesnik ili akter:
 - o Predstavlja entitet koji koristi odgovarajuće slučajeve korišćenja koji su sa njim povezani



- Slučaj korišćenja:
 - o Predstavlja opis skupa postupaka koje sistem izvršava sa ciljem proizvodjenja izvensnog rezultata
- Komunikacija: _____
 - o Predstavlja se punom linijom i može da bude inicirana od učesnika ali i od slučaja korišćenja. Relacija komunikacije se obavlja između učesnika i slučajeva korišćenja.

3.1. Slučajevi korišćenja za ulogu „Klijent“



Slika 2 Dijagram slučajeva korišćenja za ulogu "Klijent"

Iz navedenog dijagrama (slika 15) za ulogu “Klijent” primećeni su sledeći slučajevi korišćenja:

- K1. Podnošenje zahteva za otvaranjem naloga
- K2. Prijava na sistem
- K3. Podnošenje zahteva za gašenjem naloga
- K4. Odjava sa sistema
- K5. Prikazivanje detalja donešene dijagnoze
- K6. Prikazivanje detalja sistematskog pregleda
- K7. Prikazivanje detalja plaćanja
- K8. Prikazivanje detalja naloga za laboratoriju
- K9. Prikazivanje detalja naloga za pomagala
- K10. Prikazivanje detalja terapije
- K11. Prikazivanje detalja injekcije
- K12. Podnošenje zahteva za bolovanjem
- K13. Prikazivanje detalja bolovanja
- K14. Prikazivanje detalja naloga lekara
- K15. Čuvanje izbora izabranog lekara
- K16. Izmena izbora izabranog lekara
- K17. Prikazivanje detalja naloga pacijenta
- K18. Podnošenje zahteva za promenom ličnih podataka
- K19. Prikazivanje detalja uputa
- K20. Prikazivanje rezultata čitanja smart kartice
- K21. Podnošenje zahteva za čuvanjem podataka na smart kartici

K.1. Podnošenje zahteva za otvaranjem naloga

Akter: Klijent.

Opis: Podnošenje zahteva za otvaranje korisničkog naloga.

Preduslovi: Prikazan je forma u prozoru sa poljima potrebnim za podnošenje zahteva za kreiranje naloga klijenta.

Osnovni scenario:

1. Korisnik popunjava predviđena polja.
2. Sistem proverava validnost unetih podataka.
3. Korisnik traži od sistema da se zahtev za kreiranje naloga sačuva u bazi.
4. Sistem se povezuje sa bazom.
5. Sistem čuva unete podatke u bazi.
6. Sistem prikazuje poruku o uspešnosti obavljanja datog zadatka.

Alternativni scenario:

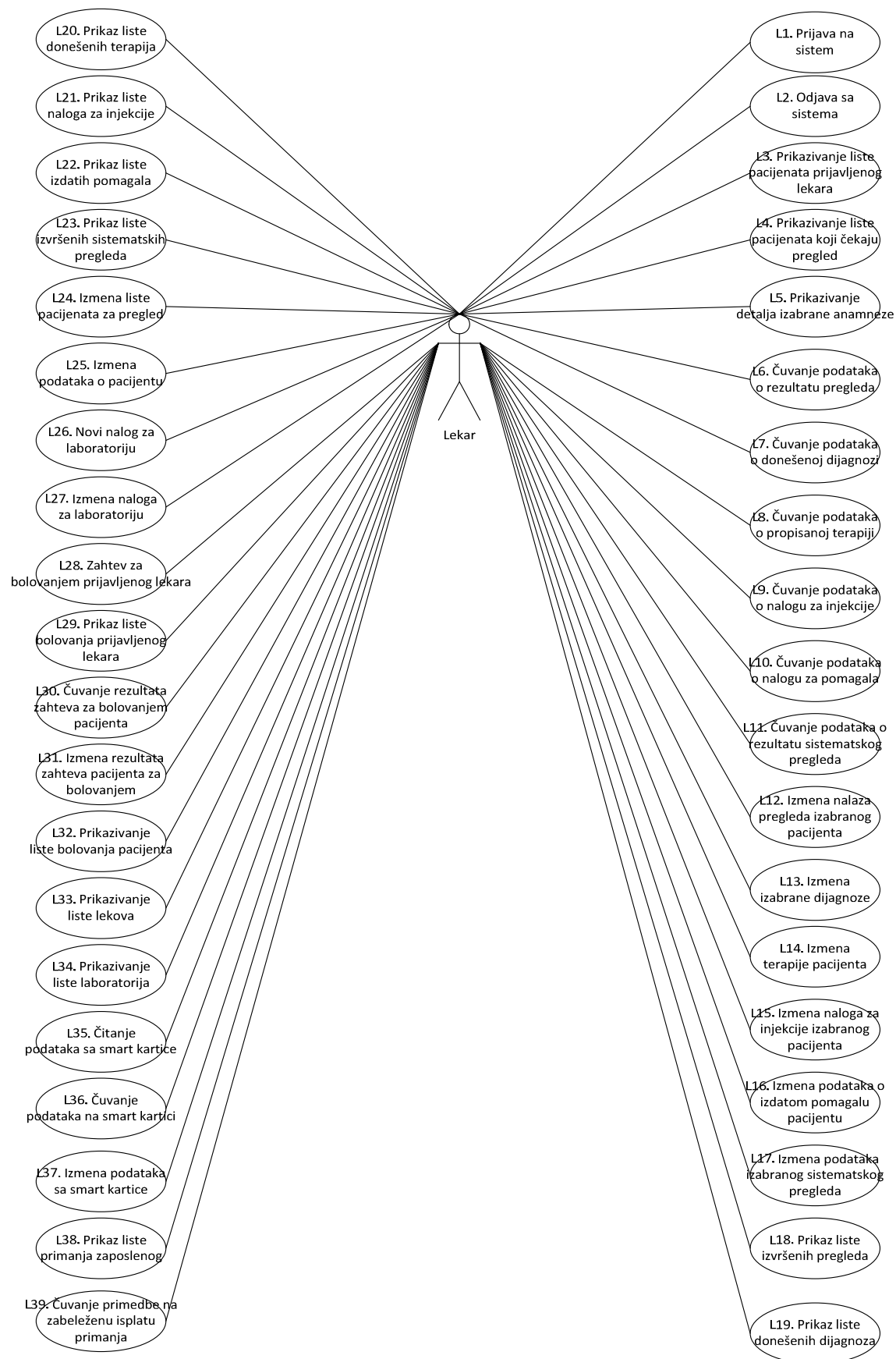
- 2.1. Sistem ne može da proveriti validnost unetog podatka.
- 2.2. Sistem ispisuje odgovarajuću poruku.
- 2.3. Prekida se dalje izvršavanje scenarija.

- 4.1. Sistem ne može da se poveže sa bazom.
- 4.2. Sistem ispisuje odgovarajuću poruku.
- 4.3. Prekida se dalje izvršavanje scenarija.

- 5.1. Sistem ne može da upiše podatke u bazu.
- 5.2. Sistem ispisuje odgovarajuću poruku.
- 5.3. Prekida se dalje izvršavanje scenarija.

Dalje obrade slučajeva korišćenja prikazane su u prilogu slučajevi koriscenja za ulogu klijent.

3.2. Slučajevi korišćenja za ulogu „Lekar“



Slika 3 Dijagram slučajeva korišćenja za ulogu "Lekar"

Iz navedenog dijagrama (slika 16) za ulogu “Lekar” primećeni su sledeći slučajevi korišćenja:

- L1. Prijava na sistem
- L2. Odjava sa sistema
- L3. Prikazivanje liste pacijenata prijavljenog lekara
- L4. Prikazivanje liste pacijenata koji čekaju pregled
- L5. Prikazivanje detalja izabrane anamneze
- L6. Čuvanje podataka o rezultatu pregleda
- L7. Čuvanje podataka o donešenoj dijagnozi
- L8. Čuvanje podataka o propisanoj terapiji
- L9. Čuvanje podataka o nalogu za injekcije
- L10. Čuvanje podataka o nalogu za pomagala
- L11. Čuvanje podataka o rezultatu sistematskog pregleda
- L12. Izmena nalaza pregleda izabranog pacijenta
- L13. Izmena izabrane dijagnoze
- L14. Izmena terapije pacijenta
- L15. Izmena naloga za injekcije izabranog pacijenta
- L16. Izmena podataka o izdatom pomagalu pacijentu
- L17. Izmena podataka izabranog sistematskog pregleda
- L18. Prikaz liste izvršenih pregleda
- L19. Prikaz liste donešenih dijagnoza
- L20. Prikaz liste donešenih terapija
- L21. Prikaz liste naloga za injekcije
- L22. Prikaz liste izdatih pomagala
- L23. Prikaz liste izvršenih sistematskih pregleda
- L24. Izmena liste pacijenata za pregled
- L25. Izmena podataka o pacijentu
- L26. Novi nalog za laboratoriju
- L27. Izmena naloga za laboratoriju
- L28. Zahtev za bolovanjem prijavljenog lekara
- L29. Prikaz liste bolovanja prijavljenog lekara
- L30. Čuvanje rezultata zahteva za bolovanjem pacijenta
- L31. Izmena rezultata zahteva pacijenta za bolovanjem
- L32. Prikazivanje liste bolovanja pacijenta
- L33. Prikazivanje liste lekova
- L34. Prikazivanje liste laboratorija
- L35. Čitanje podataka sa smart kartice
- L36. Čuvanje podataka na smart kartici
- L37. Izmena podataka sa smart kartice
- L38. Prikaz liste primanja zaposlenog
- L39. Čuvanje primedbe na zabeleženu isplatu primanja

L1. Prijava na sistem

Akter: Lekar.

Opis: Prijavljivanje na informacijski sistem.

Preduslovi: Prikazana je forma u prozoru za prijavu na sistem.

Osnovni scenario:

1. Korisnik popunjava predviđena polja.
2. Korisnik traži od sistema da se izvrši prijava na sistem.
3. Sistem se povezuje sa bazom.
4. Sistem proverava unete podatke sa podacima u bazi.
5. Sistem prikazuje poruku o uspešnosti obavljanja datog zadatka.
6. Prikazuje se glavni prozor.

Alternativni scenario:

- 3.1. Sistem ne može da se poveže sa bazom.
- 3.2. Sistem ispisuje odgovarajuću poruku.
- 3.3. Prekida se dalje izvršavanje scenarija.

- 4.1.a Sistem ne može da proveriti unete podatke u bazi.
- 4.2.a Sistem ispisuje odgovarajuću poruku.
- 4.3.a Prekida se dalje izvršavanje scenarija.

- 4.1.b Sistem ne može da pronađe odgovarajuće podatke u bazi sa onim koje je korisnik uneo.
- 4.2.b Sistem ispisuje odgovarajuću poruku.
- 4.3.b Prekida se dalje izvršavanje scenarija.

- 6.1 Sistem ne može da prikaže glavni prozor.
- 6.2 Sistem ispisuje odgovarajuću poruku.

Dalje obrade slučajeva korišćenja prikazane su u prilogu slučajevi korišćenja za ulogu lekar.

3.3. Slučajevi korišćenja za ulogu „Medicinska sestra“



Slika 4 Dijagram slučajeva korišćenja za ulogu "Medicinska sestra"

Iz navedenog dijagrama (slika 17) za ulogu "Medicinska sestra" primećeni su sledeći slučajevi korišćenja:

- MS1. Prijava na sistem
- MS2. Odjava sa sistema
- MS3. Nova anamneza
- MS4. Čuvanje podataka o datoj injekciji
- MS5. Izmena anamneze
- MS6. Izmena podataka o datoj injekciji
- MS7. Prikaz liste pacijenata
- MS8. Dodavanje naloga pacijenta u listu čekanja za pregled
- MS9. Unos podataka o novom pacijentu
- MS10. Izmena liste pacijenata za pregled
- MS11. Izmena podataka o pacijentu
- MS12. Unos zaduženja pacijenta
- MS13. Izmena zaduženja pacijenta
- MS14. Prikaz liste zaduženja pacijenta
- MS15. Unos uplate pacijenta
- MS16. Izmena uplate pacijenta
- MS17. Prikaz liste uplata pacijenta
- MS18. Zahtev za bolovanjem prijavljene medicinske sestre
- MS19. Prikaz liste bolovanja trenutno prijavljene medicinske sestre
- MS20. Unos novog uputa pacijenta
- MS21. Izmena uputa pacijenta
- MS22. Prikaz liste uputa pacijenta
- MS23. Prikaz liste primanja zaposlenog
- MS24. Čuvanje primedbe na zabeleženu isplatu primanja

MS1. Prijava na sistem

Akter: Medicinska sestra.

Opis: Prijavljivanje na informacioni sistem.

Preduslovi: Prikazana je forma u prozoru za prijavu na sistem.

Osnovni scenario:

1. Korisnik popunjava predviđena polja.
2. Korisnik traži od sistema da se izvrši prijava na sistem.
3. Sistem se povezuje sa bazom.
4. Sistem proverava unete podatke sa podacima u bazi.
5. Sistem prikazuje poruku o uspešnosti obavljanja datog zadatka.
6. Prikazuje se glavni prozor.

Alternativni scenario:

- 3.1. Sistem ne može da se poveže sa bazom.
- 3.2. Sistem ispisuje odgovarajuću poruku.
- 3.3. Prekida se dalje izvršavanje scenarija.

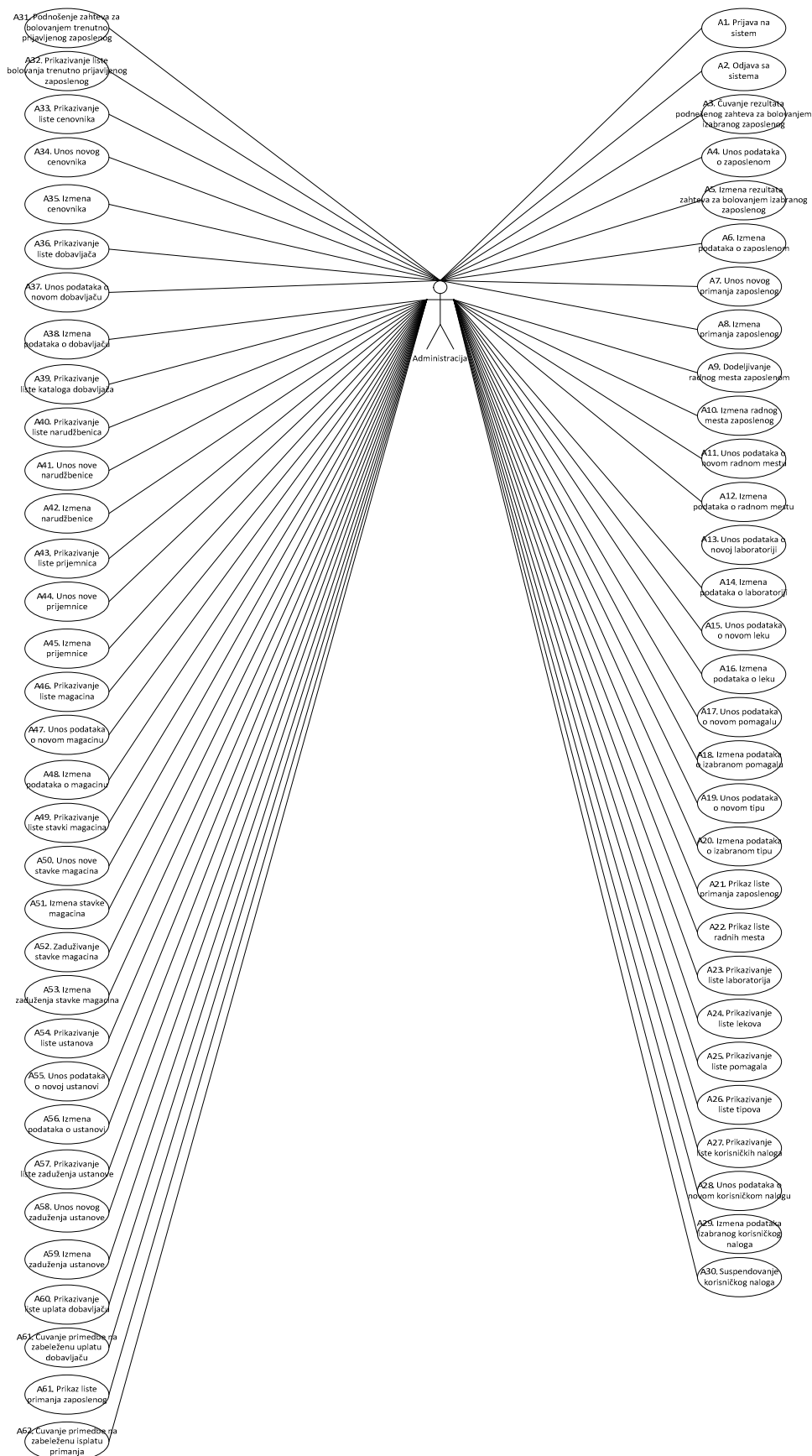
- 4.1.a Sistem ne može da proveriti unete podatke u bazi.
- 4.2.a Sistem ispisuje odgovarajuću poruku.
- 4.3.a Prekida se dalje izvršavanje scenarija.

- 4.1.b Sistem ne može da pronađe odgovarajuće podatke u bazi sa onim koje je korisnik uneo.
- 4.2.b Sistem ispisuje odgovarajuću poruku.
- 4.3.b Prekida se dalje izvršavanje scenarija.

- 6.1 Sistem ne može da prikaže glavni prozor.
- 6.2 Sistem ispisuje odgovarajuću poruku.

Dalje obrade slučajeva korišćenja prikazane su u prilogu slučajevi koriscenja za ulogu medicinska sestra.

3.4. Slučajevi korišćenja za ulogu „Administracija“



Slika 5 Dijagram slučajeva korišćenja za ulogu "Administracija"

Iz navedenog dijagrama (slika 18) za ulogu "Administracija" primećeni su sledeći slučajevi korišćenja:

- A1. Prijava na sistem
- A2. Odjava sa sistema
- A3. Čuvanje rezultata podnešenog zahteva za bolovanjem izabranog zaposlenog
- A4. Unos podataka o zaposlenom
- A5. Izmena rezultata zahteva za bolovanjem izabranog zaposlenog
- A6. Izmena podataka o zaposlenom
- A7. Unos novog primanja zaposlenog
- A8. Izmena primanja zaposlenog
- A9. Dodeljivanje radnog mesta zaposlenom
- A10. Izmena radnog mesta zaposlenog
- A11. Unos podataka o novom radnom mestu
- A12. Izmena podataka o radnom mestu
- A13. Unos podataka o novoj laboratoriji
- A14. Izmena podataka o laboratoriji
- A15. Unos podataka o novom leku
- A16. Izmena podataka o leku
- A17. Unos podataka o novom pomagalu
- A18. Izmena podataka o izabranom pomagalu
- A19. Unos podataka o novom tipu
- A20. Izmena podataka o izabranom tipu
- A21. Prikaz liste primanja zaposlenog
- A22. Prikaz liste radnih mesta
- A23. Prikazivanje liste laboratorija
- A24. Prikazivanje liste lekova
- A25. Prikazivanje liste pomagala
- A26. Prikazivanje liste tipova
- A27. Prikazivanje liste korisničkih naloga
- A28. Unos podataka o novom korisničkom nalogu
- A29. Izmena podataka izabranog korisničkog naloga
- A30. Suspendovanje korisničkog naloga
- A31. Podnošenje zahteva za bolovanjem trenutno prijavljenog zaposlenog
- A32. Prikazivanje liste bolovanja trenutno prijavljenog zaposlenog
- A33. Prikazivanje liste cenovnika
- A34. Unos novog cenovnika
- A35. Izmena cenovnika
- A36. Prikazivanje liste dobavljača
- A37. Unos podataka o novom dobavljaču
- A38. Izmena podataka o dobavljaču
- A39. Prikazivanje liste kataloga dobavljača
- A40. Prikazivanje liste narudžbenica
- A41. Unos nove narudžbenice
- A42. Izmena narudžbenice
- A43. Prikazivanje liste prijemnica
- A44. Unos nove prijemnice

- A45. Izmena prijemnice
- A46. Prikazivanje liste magacina
- A47. Unos podataka o novom magacinu
- A48. Izmena podataka o magacinu
- A49. Prikazivanje liste stavki magacina
- A50. Unos nove stavke magacina
- A51. Izmena stavke magacina
- A52. Zaduživanje stavke magacina
- A53. Izmena zaduženja stavke magacina
- A54. Prikazivanje liste ustanova
- A55. Unos podataka o novoj ustanovi
- A56. Izmena podataka o ustanovi
- A57. Prikazivanje liste zaduženja ustanove
- A58. Unos novog zaduženja ustanove
- A59. Izmena zaduženja ustanove
- A60. Prikazivanje liste uplata dobavljaču
- A61. Čuvanje primedbe na zabeleženu uplatu dobavljaču
- A62. Prikaz liste primanja zaposlenog
- A63. Čuvanje primedbe na zabeleženu isplatu primanja

A1. Prijava na sistem

Akter: Administracija.

Opis: Prijavljivanje na informacijski sistem.

Preduslovi: Prikazana je forma u prozoru za prijavu na sistem.

Osnovni scenario:

1. Korisnik popunjava predviđena polja.
2. Korisnik traži od sistema da se izvrši prijava na sistem.
3. Sistem se povezuje sa bazom.
4. Sistem proverava unete podatke sa podacima u bazi.
5. Sistem prikazuje poruku o uspešnosti obavljanja datog zadatka.
6. Prikazuje se glavni prozor.

Alternativni scenario:

- 3.1 Sistem ne može da se poveže sa bazom.
- 3.2 Sistem ispisuje odgovarajuću poruku.
- 3.3 Prekida se dalje izvršavanje scenarija.

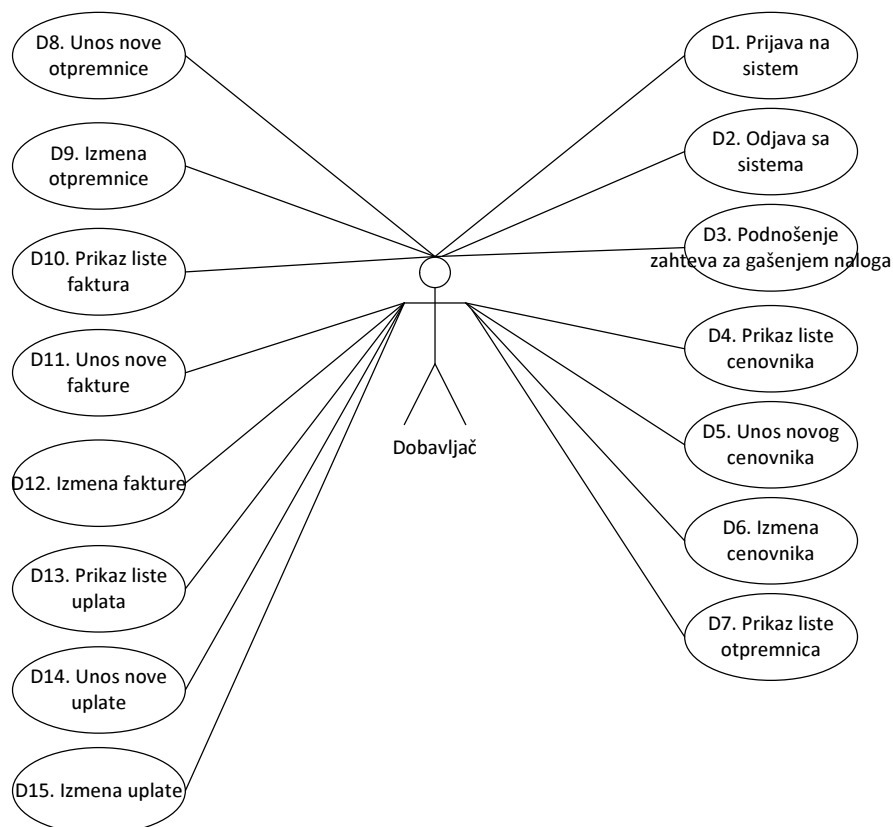
- 4.1.a Sistem ne može da proveriti unete podatke u bazi.
- 4.2.a Sistem ispisuje odgovarajuću poruku.
- 4.3.a Prekida se dalje izvršavanje scenarija.

- 4.1.b Sistem ne može da pronađe odgovarajuće podatke u bazi sa onim koje je korisnik uneo.
- 4.2.b Sistem ispisuje odgovarajuću poruku.
- 4.3.b Prekida se dalje izvršavanje scenarija.

- 6.1 Sistem ne može da prikaže glavni prozor.
- 6.2 Sistem ispisuje odgovarajuću poruku.

Dalje obrade slučajeva korišćenja prikazane su u prilogu slučajevi korišćenja za ulogu administracija.

3.5. Slučajevi korišćenja za ulogu „Dobavljač“



Slika 6 Dijagram slučajeva korišćenja za ulogu "Dobavljač"

Iz navedenog dijagrama (slika 19) za ulogu "Dobavljač" primećeni su sledeći slučajevi korišćenja:

- D1. Prijava na sistem
- D2. Odjava sa sistema
- D3. Podnošenje zahteva za gašenjem naloga
- D4. Prikaz liste cenovnika
- D5. Unos novog cenovnika
- D6. Izmena cenovnika
- D7. Prikaz liste otpremnica
- D8. Unos nove otpremnice
- D9. Izmena otpremnice
- D10. Prikaz liste faktura
- D11. Unos nove fakture
- D12. Izmena fakture
- D13. Prikaz liste uplata
- D14. Unos nove uplate
- D15. Izmena uplate

D1. Prijava na sistem

Akter: Dobavljač.

Opis: Prijavljivanje na informacioni sistem.

Preduslovi: Prikazana je forma u prozoru za prijavu na sistem.

Osnovni scenario:

1. Korisnik popunjava predviđena polja.
2. Korisnik traži od sistema da se izvrši prijava na sistem.
3. Sistem se povezuje sa bazom.
4. Sistem proverava unete podatke sa podacima u bazi.
5. Sistem prikazuje poruku o uspešnosti obavljanja datog zadatka.
6. Prikazuje se glavni prozor.

Alternativni scenario:

- 3.1. Sistem ne može da se poveže sa bazom.
- 3.2. Sistem ispisuje odgovarajuću poruku.
- 3.3. Prekida se dalje izvršavanje scenarija.

- 4.1.a Sistem ne može da proveri unete podatke u bazi.
- 4.2.a Sistem ispisuje odgovarajuću poruku.
- 4.3.a Prekida se dalje izvršavanje scenarija.

- 4.1.b Sistem ne može da pronade odgovarajuće podatke u bazi sa onim koje je korisnik uneo.
- 4.2.b Sistem ispisuje odgovarajuću poruku.
- 4.3.b Prekida se dalje izvršavanje scenarija.

- 6.1 Sistem ne može da prikaže glavni prozor.
- 6.2 Sistem ispisuje odgovarajuću poruku.

Dalje obrade slučajeva korišćenja prikazane su u prilogu slucajevi koriscenja za ulogu dobavljac.

4. Dijagrami sekvenci

Koristi se za modelovanje dinamičkih aspekata sistema. Dijagram sekvenci predstavlja jednu od relizacija slučajeva korišćenja.

Dijagrami sekvenci opisuju saradnju objekata u obavljanju neke aktivnosti vodeći računa o vremenskoj komponenti. Obično prikazuju scenario okvira slučaja korišćenja koji obuhvata izvestan broj objekata i poruka koje objekti međusobno razmenjuju.

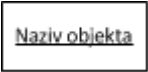
Ima ulogu prikazivanja redosleda događaja, gde spoljni, odnosno ulazni, događaj izaziva učesnik i operacija koje su odziv na događaj koji se desio u sistemu. Dijagrami sekvenci poseduju dve dimenzije. Prva dimenzija je vreme koja se pokazuje po vertikalnoj osi, odnosno dimenziji, a druga je kolekcija objekata koja se prikazuje po horizontalnoj osi, odnosno dimenziji.

Akter sa objektima, kao i objekti između sebe razmenjuju poruke koje mogu biti sinhronne, asinhronne i rekurzivne.

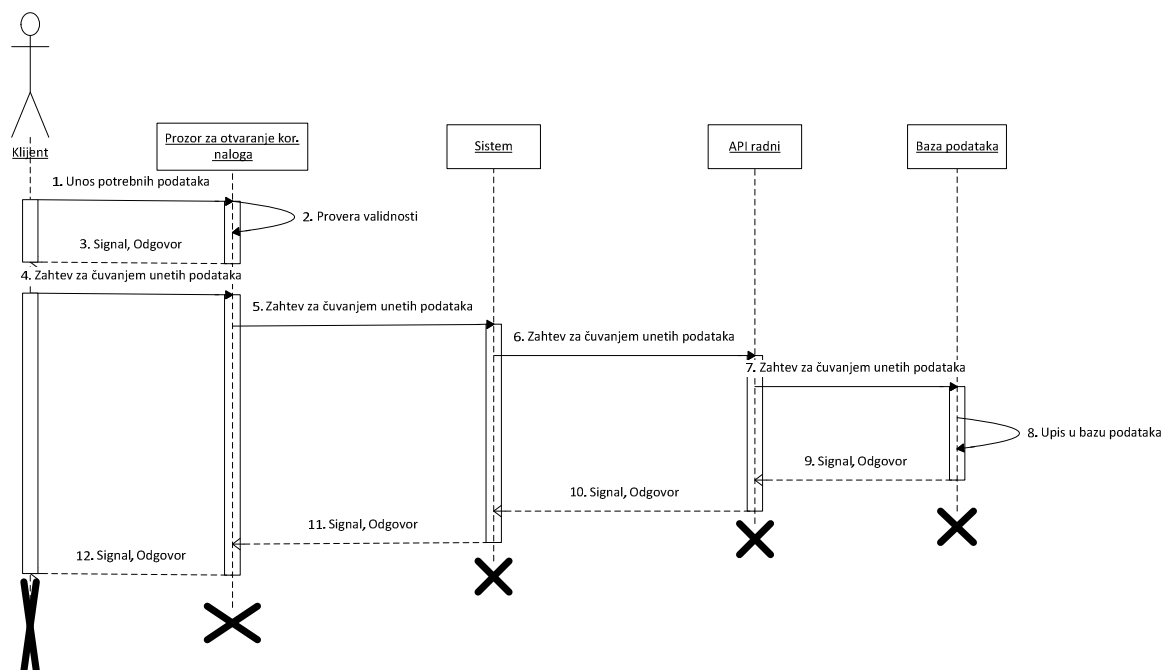
Pored poruka postoji i aktivnost kao pojam koji definise koliko su objekti i akter aktivni u određenom vremenskom intervalu.

Svaki objekat i akter ima svoju liniju života. Kada se završi linija života smatra se da taj akter ili objekat nije više aktivan i ne može mu se u daljem vremenskom intervalu pristupiti.

Legenda, koja je korišćena u projektovanju dijagrama sekvenci, nastala je pomoću alata Visio 2010 kompanije Microsoft.

- Naziv ili simbol objekta: 
 - o Definiše sam objekat
- Linija života: 
 - o Definiše životni vek objekta.
- Poruka:
 - o Sinhrona ili povratna: 
 - Ima ulogu da u komunikaciji između objekata, komunikacija ne bude jednostrana već da objekat vraća odgovor pozivaocu.
 - o Asinhrona ili ne povratna: 
 - Ona ima ulogu da samo šalje odredbe ali nema mogućnost da sluša i odziv drugog objekta na poslatu poruku.
 - o Rekurzivna: 
 - Ima ulogu sa objekat sam sebi šalje poruku, tj. da u isto vreme bude i pošiljalac i primalac.
- Aktivnost: 
 - o Nalazi se na liniji života objekta i definiše kada je i koliko objekat aktivan.

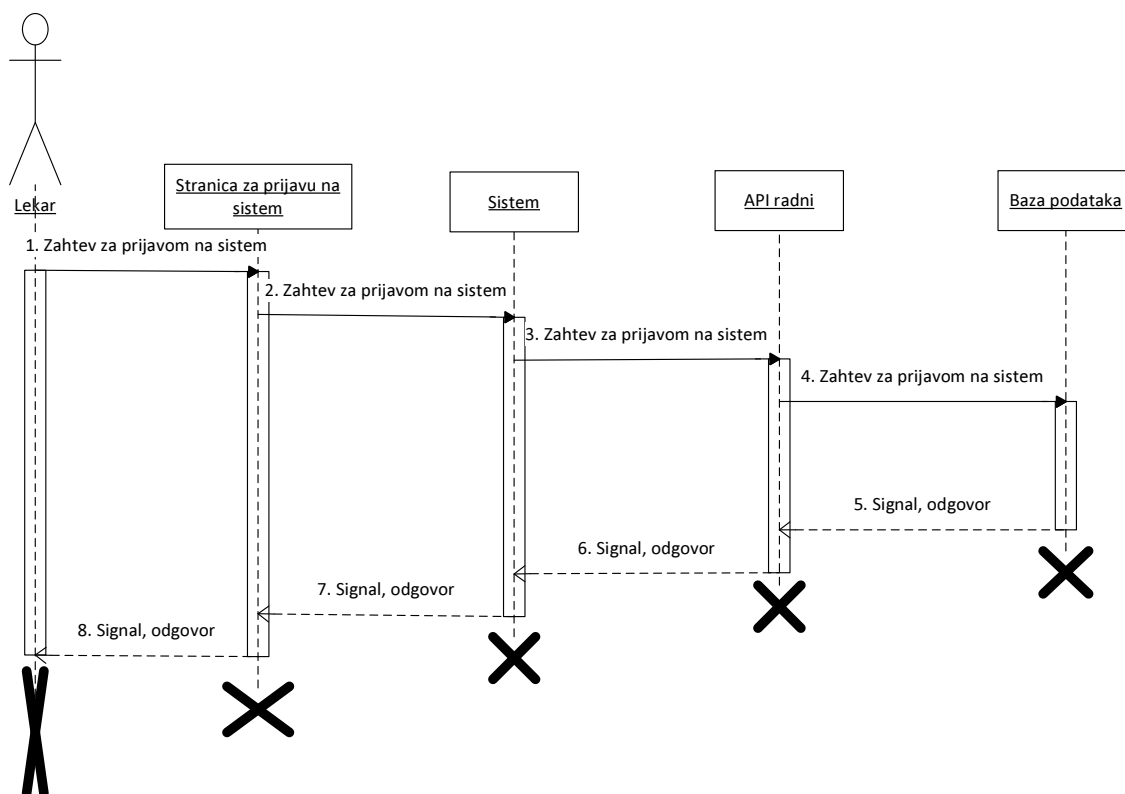
4.1. Dijagrami sekvenci za ulogu „Klijent“



Slika 7. Dijagram sekvenci uloga „Klijent“, Podnošenje zahteva za otvaranjem naloga

Dalji dijagrami sekvenci za ulogu klijent prikazani su u prilogu dijagrami sekvenci za ulogu klijent.

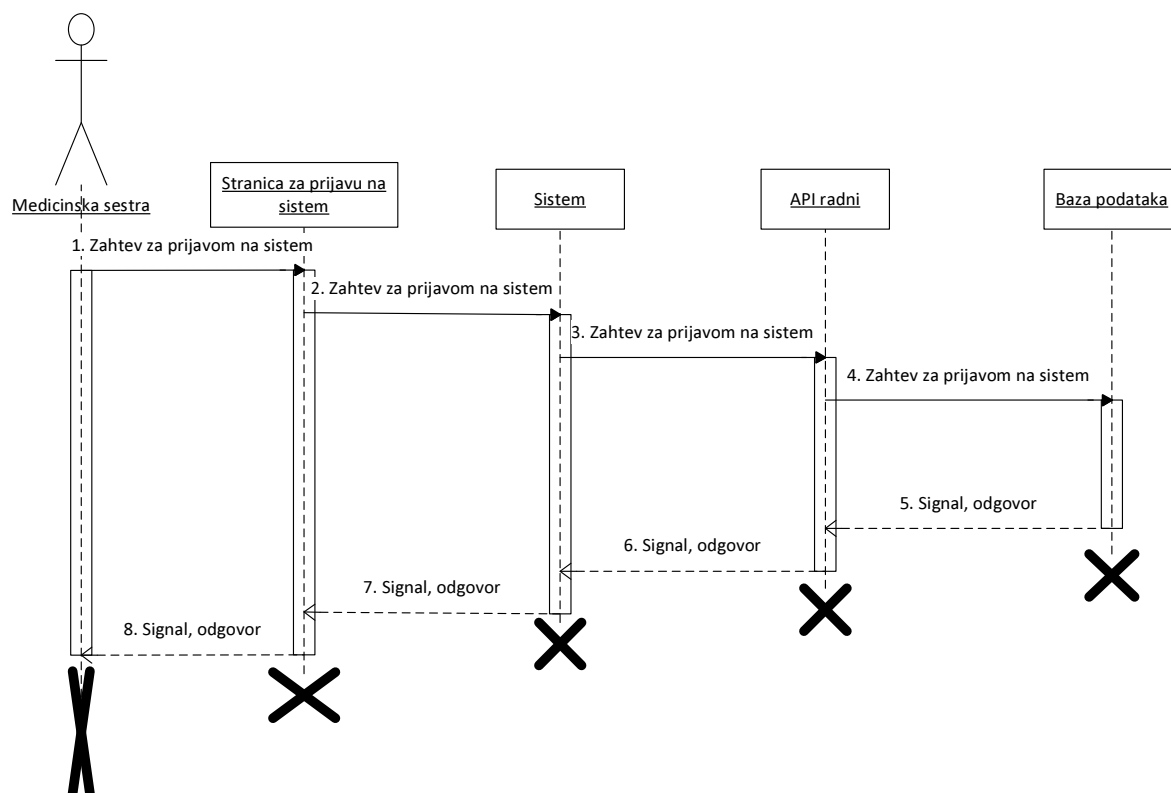
4.2. Dijagrami sekvenci za ulogu „Lekar“



Slika 8. Dijagram sekvenci uloga „Lekar“, Prijava na sistem

Dalji dijagrami sekvenci za ulogu lekar prikazani su u prilogu dijagrami sekvenci za ulogu lekar.

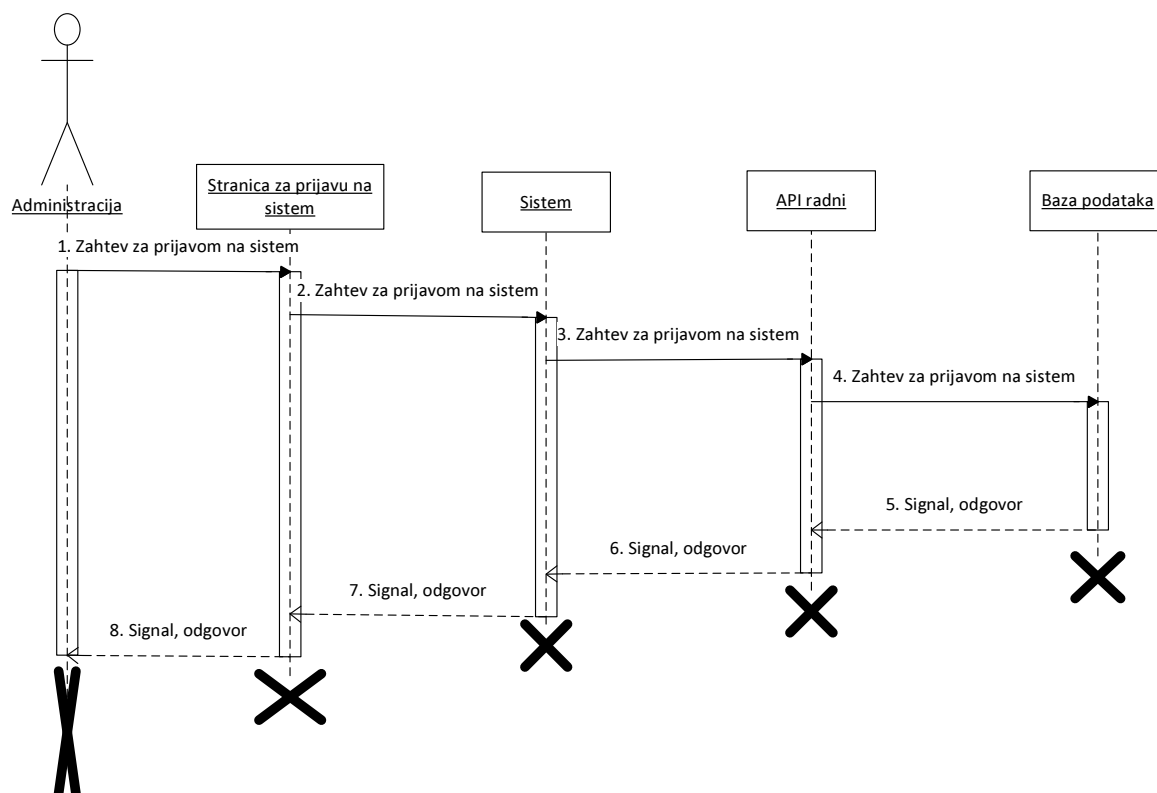
4.3. Dijagrami sekvenci za ulogu „Medicinska sestra“



Slika 9. Dijagram sekvenci uloga „Medicinska sestra“, Prijava na sistem

Dalji dijagrami sekvenci za ulogu medicinska sestra prikazani su u prilogu dijagrami sekvenci za ulogu medicinska sestra.

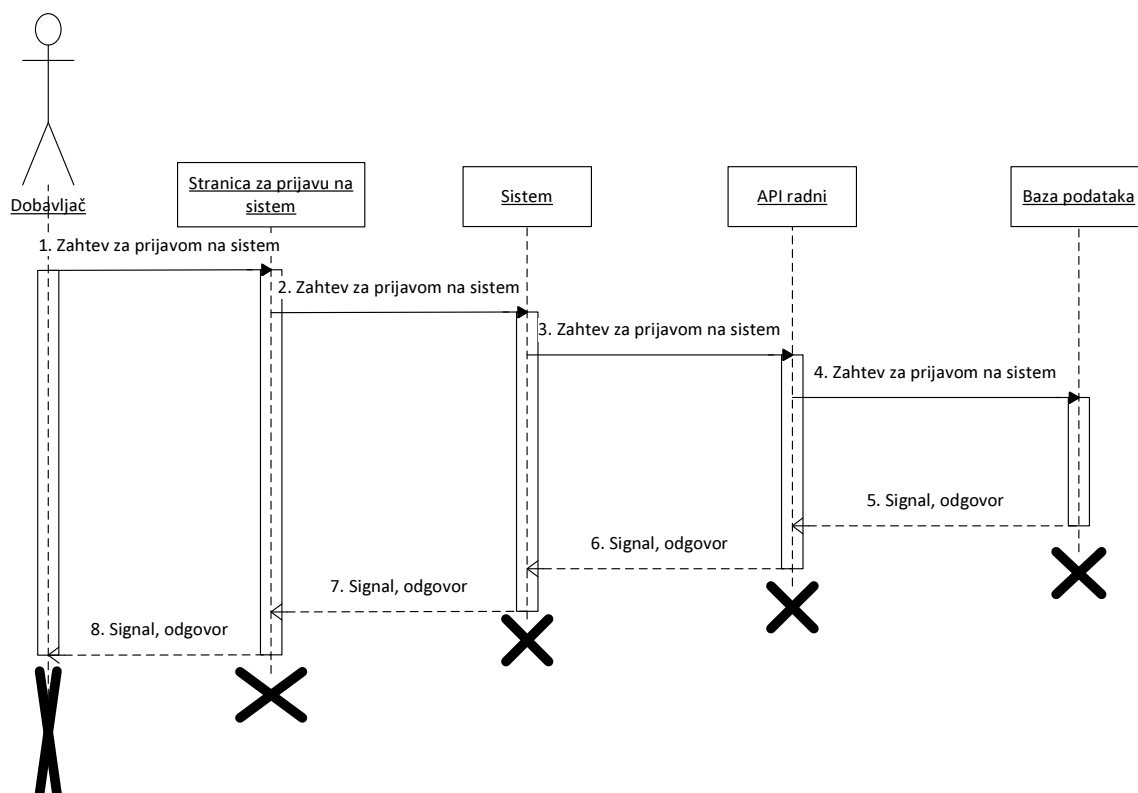
4.4. Dijagrami sekvenci za ulogu „Administracija“



Slika 10. Dijagram sekvenci uloga „Administracija“, Prijava na sistem

Dalji dijagrami sekvenci za ulogu administracija prikazani su u prilogu dijagrami sekvenci za ulogu administracija.

4.5. Dijagrami sekvenci za ulogu „Dobavljač“



Slika 11 Dijagram sekvenci uloga „Dobavljač“, Prijava na sistem

Dalji dijagrami sekvenci za ulogu dobavljač prikazani su u prilogu dijagrami sekvenci za ulogu dobavljač.

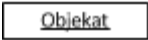
5. Dijagrami kolaboracije

Opisuje interakciju kroz poruke između objekata, i u tome ima sličnosti sa dijagramima sekvenci. Treba da se ima u obziru da, kod dijagrama kolaboracije, žiža nije postavljena na vreme, već na relacije između objekata.

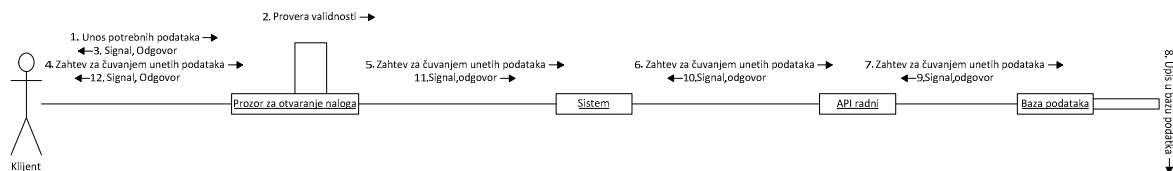
Dijagrami kolaboracije služe za prikaz složenih interakcija između objekata kao i njihove međusobne povezanosti.

Sastoje se od objekata, linkova i poruka.

Legenda, koja je korišćena u projektovanju dijagrama kolaboracije, nastala je pomoću alata Visio 2010 kompanije Microsoft.

- Objekat: 
 - o Entitet koji ostvaruje komunikaciju sa drugim objektima
- Link: 
 - o Veza koja omogućava komunikaciju između objekata.
- Poruka: 
 - o Opis komunikacije između objekata. Ima svoj smer.

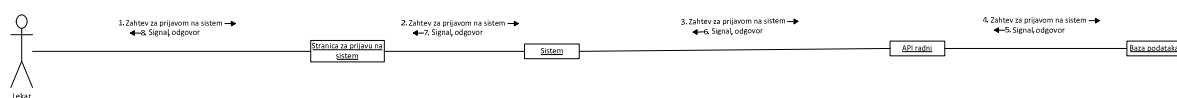
5.1. Dijagrami kolaboracije za ulogu „Klijent“



Slika 12. Dijagram kolaboracije uloga „Klijent“, Podnošenje zahteva za otvaranjem naloga

Dalji dijagrami kolaboracije za ulogu klijent prikazani su u prilogu dijagrami sekvenci za ulogu klijent.

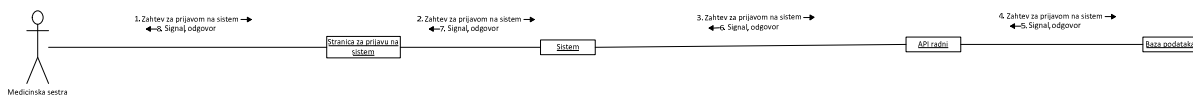
5.2. Dijagrami kolaboracije za ulogu „Lekar“



Slika 13. Dijagram kolaboracije uloga „Lekar“, Prijava na sistem

Dalji dijagrami kolaboracije za ulogu lekar prikazani su u prilogu dijagrami sekvenci za ulogu lekar.

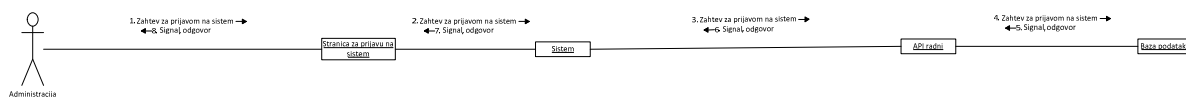
5.3. Dijagrami kolaboracije za ulogu „Medicinska sestra“



Slika 14. Dijagram kolaboracije uloga „Medicinska sestra“, Prijava na sistem

Dalji dijagrami kolaboracije za ulogu medicinska sestra prikazani su u prilogu dijagrami sekvenci za ulogu medicinska sestra.

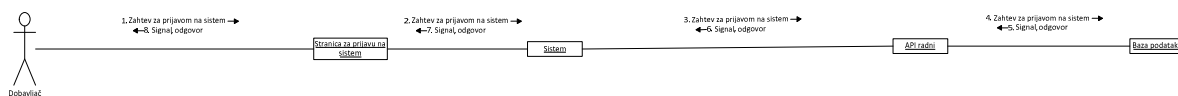
5.4. Dijagrami kolaboracije za ulogu „Administracija“



Slika 15. Dijagram kolaboracije uloga „Administracija“, Prijava na sistem

Dalji dijagrami kolaboracije za ulogu administracija prikazani su u prilogu dijagrami sekvenci za ulogu administracija.

5.5. Dijagrami kolaboracije za ulogu „Dobavljač“



Slika 16 Dijagram kolaboracije uloga „Dobavljač“, Prijava na sistem

Dalji dijagrami kolaboracije za ulogu dobavljač prikazani su u prilogu dijagrami sekvenci za ulogu dobavljač.

6. Dijagrami aktivnosti

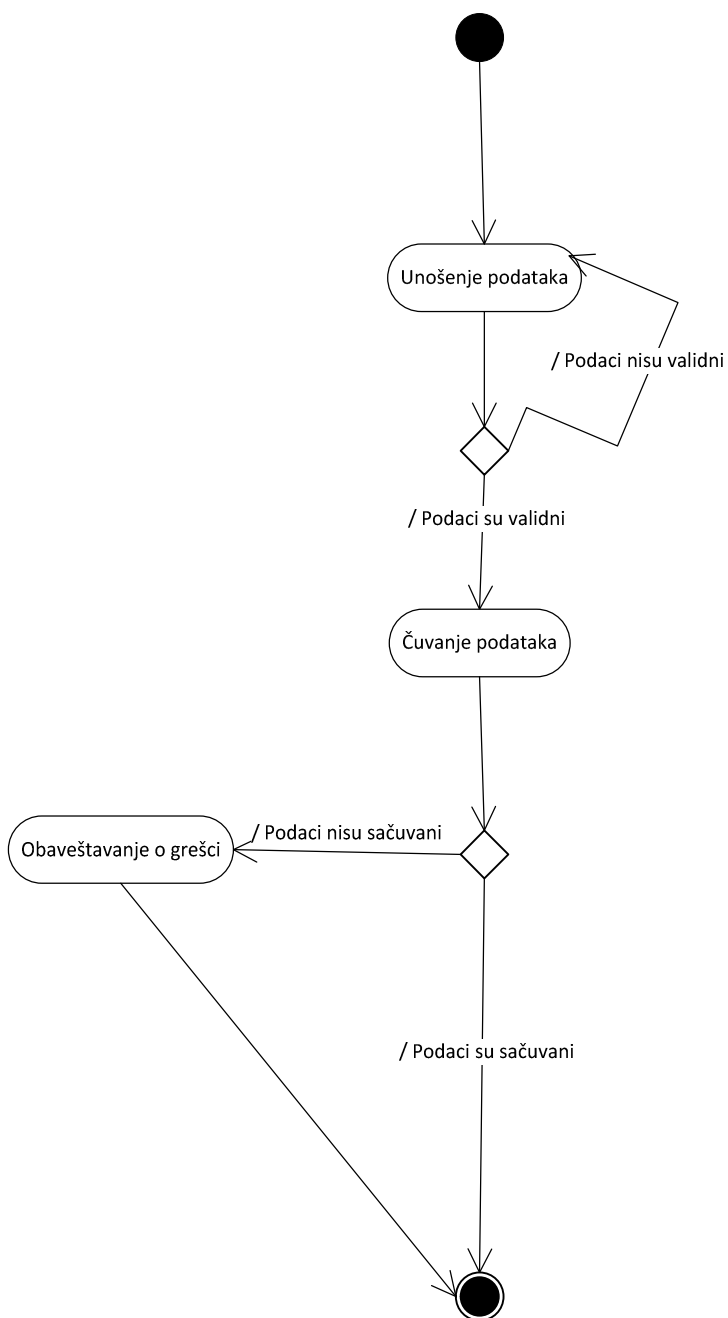
Dijagrami aktivnosti predstavljaju generalizovan , pojednostavljen prikaz određenog procesa. Dijagrami aktivnosti prikazuju tok posla koji treba obaviti kroz opisivanje njegovih logičkih procedura i postupaka.

Svaki dijagram aktivnosti počinje sa početnom tačkom i završava se sa završnom tačkom. Između te dve tačke postoje aktivnosti, tačke odlučivanja i grananja. Namenjeni su modeliranju dinamičkih aspekata ponašanja sistema

Legenda, koja je korišćena u projektovanju dijagrama aktivnosti, nastala je pomoću alata Visio 2010 kompanije Microsoft.

- Početno stanje ili početni čvor: 
 - o To je tačka od koje počinje odvijanje procesa.
- Akcija: 
 - o Predstavlja korak u aktivnosti koji se dalje obično ne dekomponuje.
- Nit: 
 - o Predstavlja putanju od jednog do drugog čvora preko akcija. Mogu da se granaju na više konkurentnih tj. paralelnih niti sinhronizacionim tačkama. Takođe u sinhronizacionim tačkama mogu i da se udružuju.
- Sinhronizaciona tačka: 
 - o Predstavlja tačku grananja niti na više konkurentnih niti ili njihovog udruživanja.
- Grananje: 
 - o Određuje alternativne putanje kojima će se ići u zavisnosti od ispunjenosti uslova. Više grana može ulaziti u grananje kao i više grana može iz njega da izlazi.
- Krajnje stanje ili krajnji čvor: 
 - o Određuje kraj aktivnosti

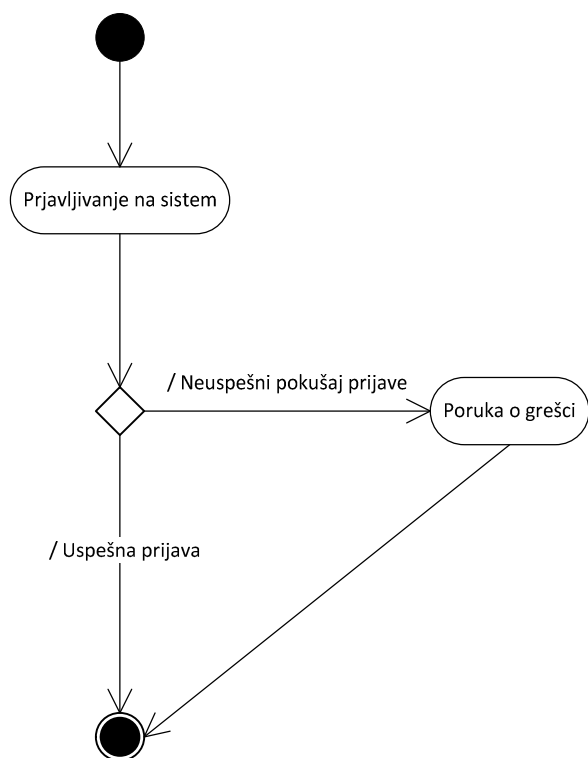
6.1. Dijagram aktivnosti za ulogu „Klijent“



Slika 17. Dijagram aktivnosti uloga „Klijent“, Podnošenje zahteva za otvaranjem naloga

Dalji dijagrami aktivnosti za ulogu klijent prikazani su u prilogu dijagrami sekvenci za ulogu klijent.

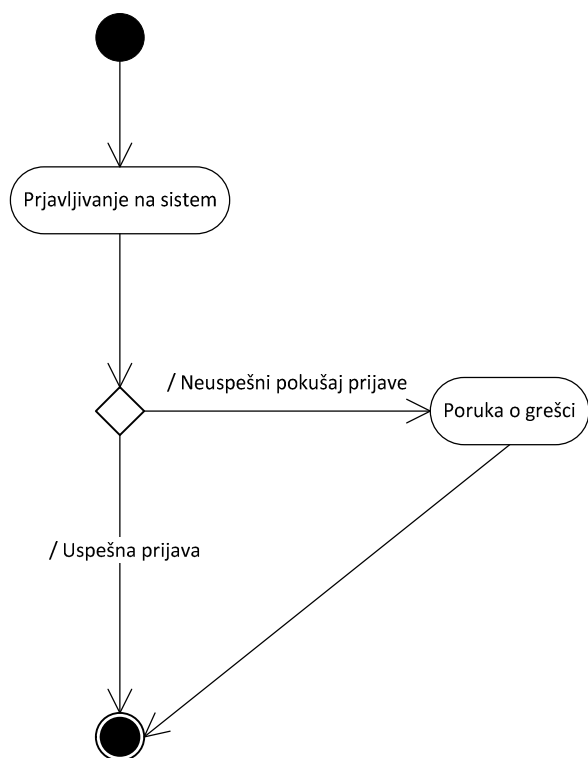
6.2. Dijagrami aktivnosti za ulogu „Lekar“



Slika 18. Dijagram aktivnosti uloga „Lekar“, Prijava na sistem

Dalji dijagrami aktivnosti za ulogu lekar prikazani su u prilogu dijagrami sekvenci za ulogu lekar.

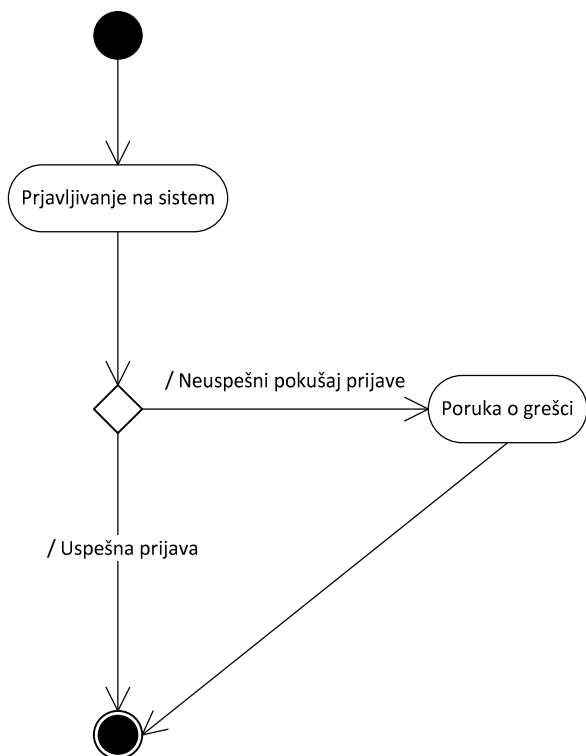
6.3. Dijagram aktivnosti za ulogu „Medicinska sestra“



Slika 19. Dijagram aktivnosti uloga „Medicinska sestra“, Prijava na sistem

Dalji dijagrami aktivnosti za ulogu medicinska sestra prikazani su u prilogu dijagrami sekvenci za ulogu medicinska sestra.

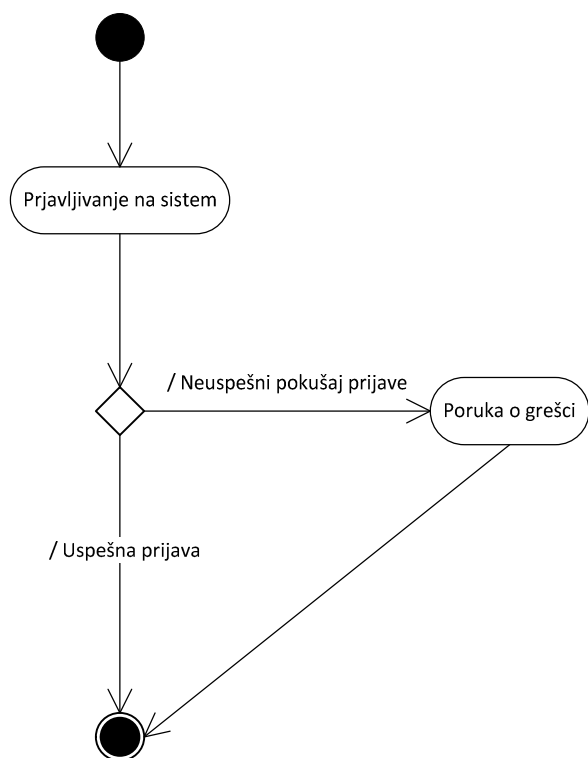
6.4. Dijagrami aktivnosti za ulogu „Administracija“



Slika 20. Dijagram aktivnosti uloga „Administracija“, Prijava na sistem

Dalji dijagrami aktivnosti za ulogu administracija prikazani su u prilogu dijagrami sekvenci za ulogu administracija.

6.5. Dijagrami aktivnosti za ulogu „Dobavljač“



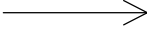
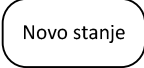
Slika 21 Dijagram aktivnosti uloga „Dobavljač“, Prijava na sistem

Dalji dijagrami aktivnosti za ulogu dobavljač prikazani su u prilogu dijagrami sekvenci za ulogu dobavljač.

7. Dijagrami stanja

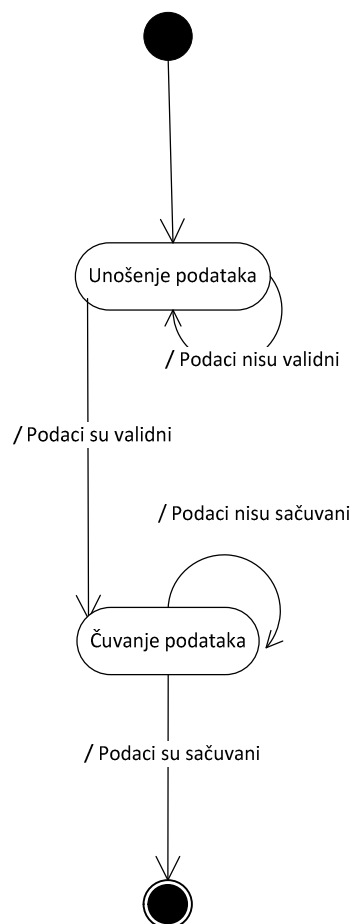
Bitna pretpostavka za dijagram stanja je da sistem koji se opisuje pomoću takvog dijagrama je sastavljen od konačnog broja elemenata. Oni se koriste za opisivanje ponašanja sistema. Obično se pomoću njih modeluju objekti koji reaguju. Između stanja u dijagramu stanja se nalaze tranzicije koje su putanje koje određuju da će objekat iz jednog stanja, izvršiti neke operacije tj. akcije i preći će u drugo stanje. Uslov za prelazak iz jednog stanja u drugo je upravo događaj koji se dešava u tranziciji tj akcije.

Legenda, koja je korišćena u projektovanju dijagrama stanja, nastala je pomoću alata Visio 2010 kompanije Microsoft.

- Početno stanje: 
 - o To je stanje u kome je objekat stvoren. Objekat će menjati to svoje početno stanje.
- Tranzicija: 
 - o Skup aktivnosti sa uslovima koje moraju biti obavljene i uslovi ispunjeni da bi objekat prešao iz jednog stanja u drugo.
- Novo stanje: 
 - o Stanje u koje objekat dolazi kada obavi sve aktivnosti iz prethodne tranzicije kao i uslove.
- Završno stanje: 

Zadnje stanje u kome se dati objekat nalazi shodno gledištu.

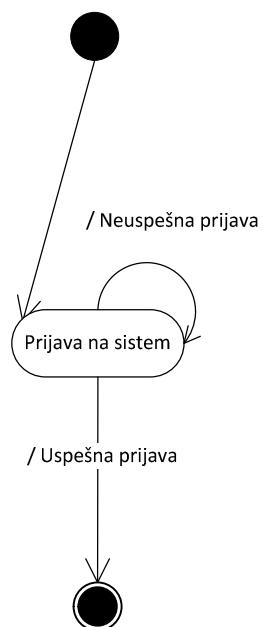
7.1. Dijagrami stanja za ulogu „Klijent“



Slika 22. Dijagram stanja uloga „Klijent“, Podnošenje zahteva za otvaranjem naloga

Dalji dijagrami stanja za ulogu klijent prikazani su u prilogu dijagrami sekvenci za ulogu klijent.

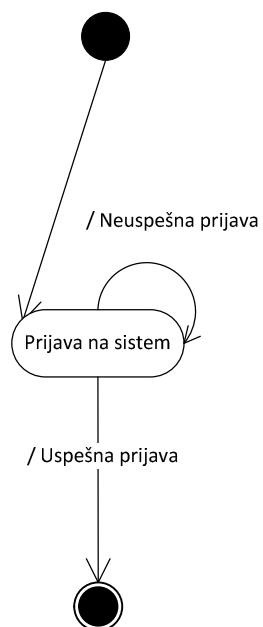
7.2. Dijagrami stanja za ulogu „Lekar“



Slika 23. Dijagram stanja uloga „Lekar“, Prijava na sistem

Dalji dijagrami stanja za ulogu lekar prikazani su u prilogu dijagrami sekvenci za ulogu lekar.

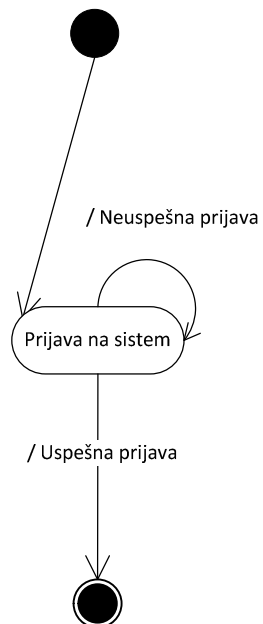
7.3. Dijagrami stanja za ulogu „Medicinska sestra“



Slika 24. Dijagram stanja uloga „Medicinska sestra“, Prijava na sistem

Dalji dijagrami stanja za ulogu medicinska sestra prikazani su u prilogu dijagrami sekvenci za ulogu medicinska sestra.

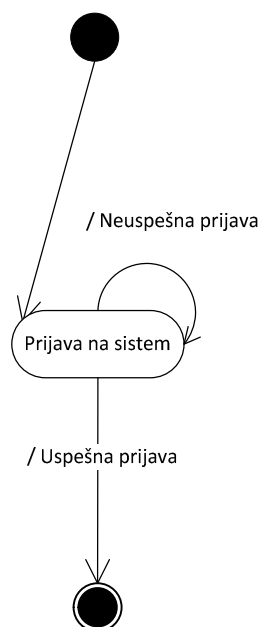
7.4. Dijagrami stanja za ulogu „Administracija“



Slika 25. Dijagram stanja uloga „Administracija“, Prijava na sistem

Dalji dijagrami stanja za ulogu administracija prikazani su u prilogu dijagrami sekvenci za ulogu administracija.

7.5. Dijagrami stanja za ulogu „Dobavljač“



Slika 26 Dijagram aktivnosti uloga „Dobavljač“, Prijava na sistem

Dalji dijagrami stanja za ulogu dobavljač prikazani su u prilogu dijagrami sekvenci za ulogu dobavljač.

8. Dijagram klasa

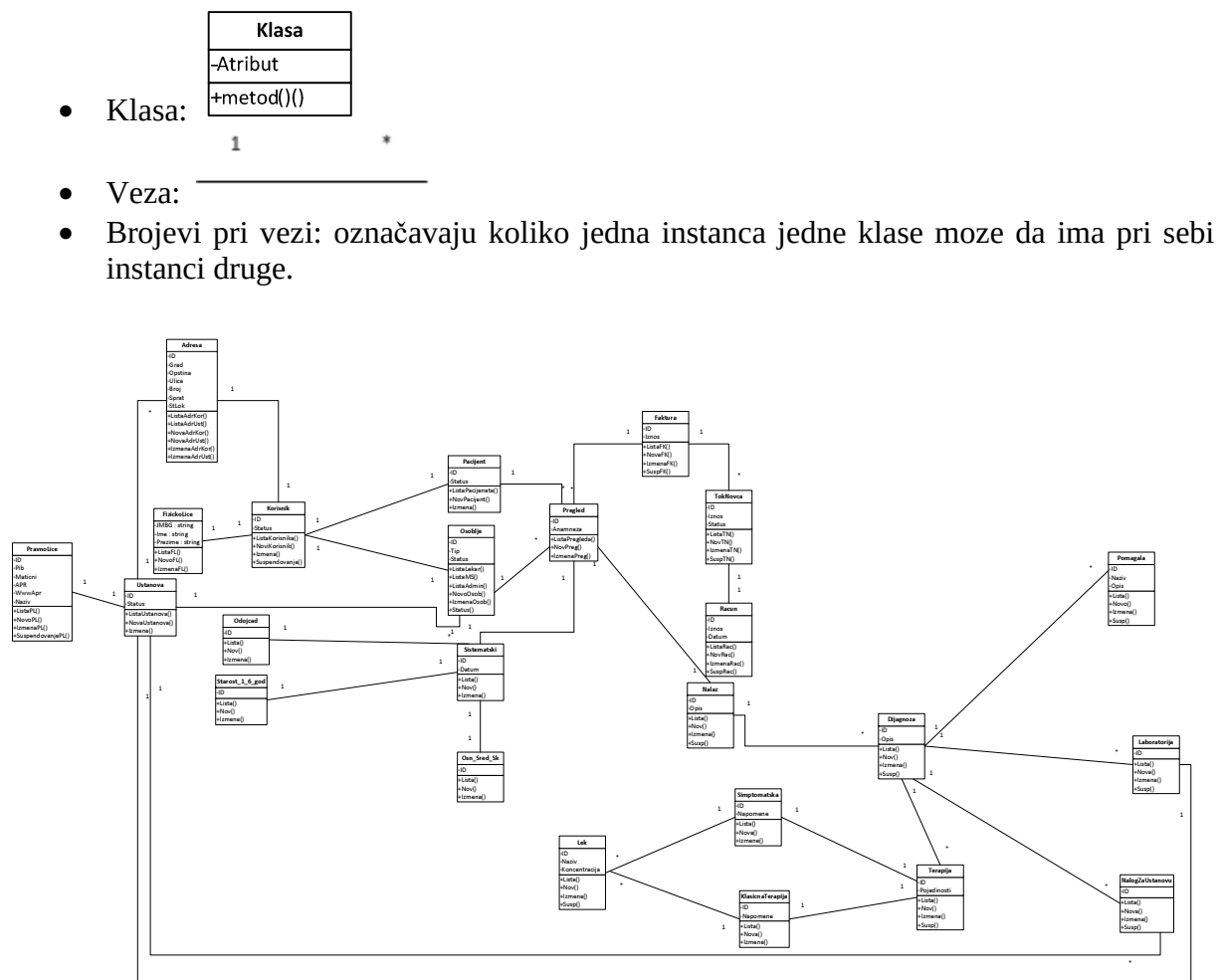
Dijagram klasa je dijagram koji sadrži klase, ali može da sadrži i interfejse i pakete, kao i njihove međusobne odnose, koji sve skupa daju logički pogled na informacioni sistem.

Dijagrami klasa opisuju entitete u sistemu, različite vrste statičkih veza koje postoje između entiteta, kao i ograničenja u načinu njihovog povezivanja.

Svrha dijagrama klasa je da na pojednostavljen način prikaže klase koje se koriste u jednom programu ili i informacionom sistema. Razlog je što jedan informacioni sistem sadrži u sebi jedan ili više programa, aplikacija pored ostalih komponenti.

Svaka klasa u sebi ima određene atribute kao i metode koje rade određene operacije sa atributima.

Legenda, koja je korišćena u projektovanju dijagrama klasa, nastala je pomoću alata Visio 2010 kompanije Microsoft. Predstavljen je klasni dijagram koji, radi preglednosti, sadrži samo najvažnije klase.



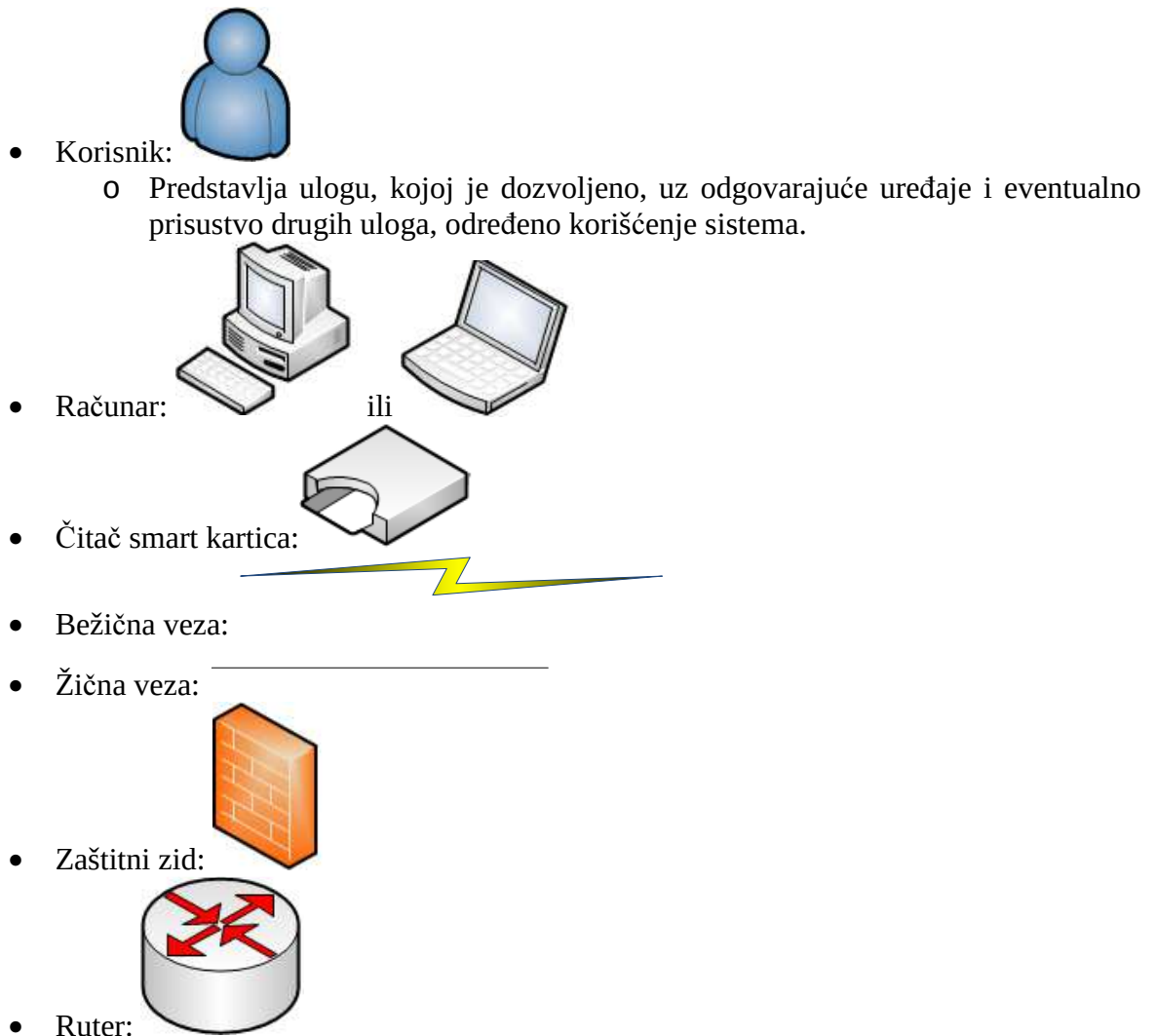
Slika 27. Dijagram osnovnih klasa informacionog sistema

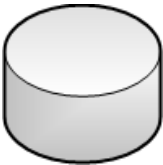
9. Dijagram rasporeda

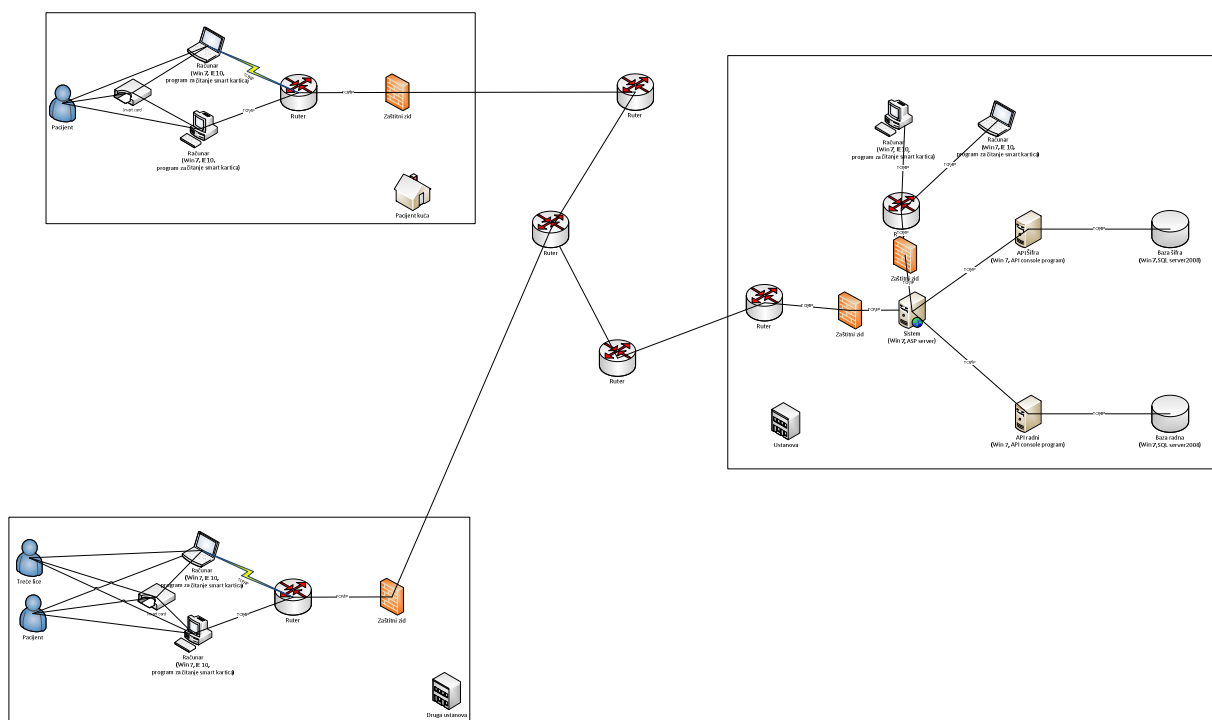
Dijagrami rasporeda opisuju fizičku strukturu sistema, prikazujući njegovu hardversku i softversku arhitekturu.

Dijagrami rasporeda sadrži grafičke reprezentacije uređaja koji se koriste u komunikaciji. Njegova uloga je prikazivanje logičkog rasporeda uređaja koji služe za funkcionisanje informacionog sistema. U njemu, pored uređaja, mogu da budu prikazani i učesnici kao što su pacijenti, lekari, medicinska sestra, administracija.

Legenda, koja je korišćena u projektovanju dijagrama rasporeda, nastala je pomoću alata Visio 2010 kompanije Microsoft. Predstavljen je klasni dijagram koji, radi preglednosti, sadrži samo najvažnije klase.



- Baza podataka: 
- API Server: 
 - o Server koji je zadužen za komunikaciju između web servera i baze podataka.
- Web server: 
 - o Server koji je zadužen za komunikaciju između korisnika i API servera.
- Objekat:  ili 
 - o Objekti označavaju lokacije sa kojih je moguće da se komunicira sa web serverom



Slika 28. Dijagram rasporeda informacionog sistema

10. Model objekti i veze

Model objekti i veze, ili sklaćeno MOV, olakšava projektovanje baza podataka. Njegova svrha je da omogući konceptualni prikaz baze podataka. Ne zavise od izbora DBMS-a (Data Base Management System) tj. SUBP(Sistem za Upravljanje Bazom Podataka). Model se sastoji od entiteta, atributa i veza.

Glavna komponenta MOV pristupa je koncept entiteta (objekata i veza) – opšti pojam za nešto što postoji i može se jednoznačno prepoznati¹.

Entitet može da bude bilo koji objekat ili stavka od interesa. Entiteti kao objekti mogu biti i fizičke ali i više apstraktne stavke(radnik, proizvod, i slično).

Atribut je podređen entitetu, on predstavlja neku činjenicu o entitetu. Vrednosti koje oni nose mogu biti proste i složene, pa samim tim i oni mogu biti prosti i više vrednosni.

Veze služe da se ostvari povezanost između entiteta. Veze određuju koliko jedan objekat učestvuje u interakciji sa drugim. Postoje tri tipa povezanosti između objekata, odnosno učestvovanja objekata u njihovoj vezi. Tipovi povezanosti mogu biti jedan na jedan (1:1), jedan na više (1:N), više na više(N:M).

Postoji gornja i donja granica učestvovanja entiteta u vezi. Ona se označava sa dva broja odvojena zarezom (prim. 1,2) ili brojem i slovom, ovdvojenim zarezom(1,M).

Legenda, koja je korišćena u projektovanju informacionog sistema, nastala je pomoću alata Visio 2010 kompanije Microsoft.

Sistem je projektovan da koristi dva baze podataka. U jednoj bazi podataka biće smešteni radni podaci. Druga baza biće zadužena za čuvanje podataka vezanih za omogućavanje pravilnog šifrovanja i dešifrovanja radnih podataka koji su sačuvani na Smart kartici.

Zbog postojanja dve baze, biće prikazana i dva dijagrama modeli objekti i veze koje date baze određuju i opisuju.

- Entitet

- o Jak: 

Jak entitet može da postoji bez postojanja bilo kog drugog entiteta.

- o Slab: 

Slab entitet ne može da postoji bez postojanja drugih entiteta.

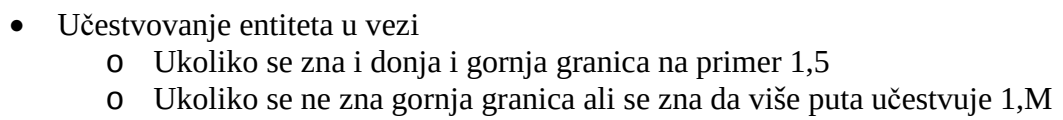
- Atribut

- o Prost: 

- o Složen: 

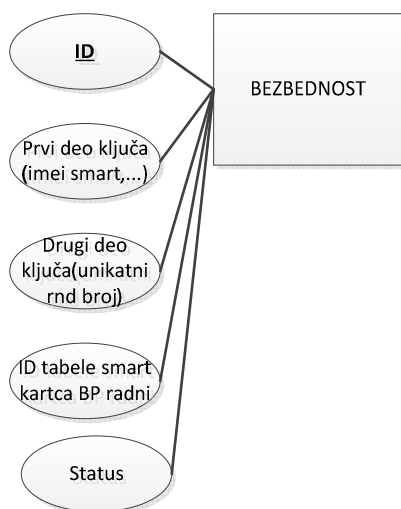
- Veza

¹ M.Veinović, G.Šimić, A.Jevremović, I.Franc, Baze podataka,. Univerzitet Singidunum, 2013, strana 60



54

10.2. Dijagram model objekti i veze baze podataka za čuvanje bezbednosnih podataka



Slika 30. Dijagram model objekti i veze baze podataka za čuvanje bezbednosnih podataka

11. Relacioni model

Sam po sebi relacioni model predstavlja jednu jednostavnu i prihvatljivu strukturu korisniku, jer on vizuelno predstavlja skup tabela.

Pri stvaranju relacionog modela treba da se obrati pažnja na pravila preslikavanja, kao i na prevođenje tabela relacionog modela u odgovarajuće normalne forme.

Osnovna pravila preslikavanja su:

- Veza 1:1 – primarni ključ jedne tabele ide u drugu tabelu kao spoljni ključ.
- Vaza 1:N – primarni ključ sa strane 1 ide u tabelu sa strane N kao spoljni ključ.
- Veza N:M – Stvara se nova tabela koja kao delove svog primarnog ključa ima primarni ključ od svake od tabela učesnica u vezi.
- Ukoliko je povezan slab i jak objekat - primarni ključ jakog objekta prelazi da bude spoljni ključ u slabom objektu
- Više vrednosni atributi – stvara se nova tabela koja ima primarni ključ kao i tabela vlasnik viševrednosnog atributa, i ima obično polje koje je predstavlja prostu vrednost tog, nekadašnjeg složenog atributa.

Prikazani su modeli za svaku normalnu formu, zaključno sa četvrtom normalnom formom.

Proces normalizacije je zasnovan na hijerarhiji normalnih formi. Svaka normalna forma zasniva se na prethodnoj i definiše rešenje problema koje prethodnik nije pokrio.

Normalne forme daju određene uslove po kojima se ispituje da li određena šema podataka ispunjava zahteve da se redundansa smanji ukoliko ne može da se eliminiše i da baza podataka može jednostavno da se koristi i menja. Smanjivanjem redundanse poboljšavaju se i vremenski interval u čekanju dobavljanja podataka iz baze podataka. Preterana primena normalizacije sastoji se u povećanju dužine vremenskog intervala koji je potreban da se prikupe podaci iz baze podataka, zato što postoji veliki broj tabela. Pri normalizaciji tabela pored smanjivanja redundansi, treba da se imaju u vidu i vremenske performanse samog sistema. Normalizovanje do treće normalne forme, po iskustvu mnogih, pokazalo je dobar odnos redundanse i vremenskog intervala.

Baza se smatra normalizovanom ukoliko poštuje pravila iz prve tri normalne forme.

Za potrebe projektovanja sistema korišćen je relacioni model koji prikazuje normalizovane tabele u trećoj normalnoj formi. Neke od tabela ispunjavaju uslove i za Boyce Codd-ovu normalnu formu. Neke od tabela (dešavanje i bezbednost) neće biti normalizovane.

Ukratko o uslovima.

- Prva normalna forma : svi atributi moraju da su atomski.
- Druga normalna forma: ako je u prvoj normalnoj formi i ako svi njeni neključni atributi potpuno i funkcionalno zavise od primarnog ključa

- Treća normalna forma: ako je u drugoj normalnoj formi i ako svi njeni neključni atributi netranzitivno funkcionalno zavise od primarnog ključa.
- Četvrta normalna forma: ako je u trećoj normalnoj formi i ne sadrži zavisnosti višestruke vrednosti

Zbog ograničenog obima stranica rada, u radu je prikazana prva normalna forma, dok prikaz ostalih normalnih formi je prikazan u prilogu relacioni model.

11.1. Relacioni model – prva normalna forma

11.1.1. Relacioni model za radnu bazu podataka

Jaki objekti

Korisnik

<u>ID</u>	IDFL	Uloga
-----------	------	-------

Kontakt podatak

<u>ID</u>	IDKor
-----------	-------

Telefon

<u>ID</u>	IDKontPod
-----------	-----------

Sifra lekara

<u>ID</u>	Sifra
-----------	-------

Tip medicinskog osoblja

<u>ID</u>	Vrednost
-----------	----------

Osoblje

<u>ID</u>	IDTipMedOsob	IDSifLekara	IDKor
-----------	--------------	-------------	-------

Fizicko lice

<u>ID</u>	IDRadM	JMBG	Ime	Prezime
-----------	--------	------	-----	---------

Sifra radno mesto

<u>ID</u>	Vrednost
-----------	----------

Radno mesto

<u>ID</u>	IDSifRM
-----------	---------

Tip uputa

<u>ID</u>	Vrednost
-----------	----------

Uput

<u>ID</u>	IDTipUpu ta	IDKorUpu ce n	IDKorUpu cu je	Opis	Datum
-----------	----------------	---------------------	----------------------	------	-------

Protokol

<u>ID</u>	Broj
-----------	------

Izvestaj

<u>ID</u>	IDProtok ol	IDUput	DatPrijema	Nalaz	Misljenje	DatIzvestaja
-----------	----------------	--------	------------	-------	-----------	--------------

Pravno lice

<u>ID</u>	IDKor	IDSifDel	PIB	Maticni	APR	RegBroj	WwwAPR	Naziv
-----------	-------	----------	-----	---------	-----	---------	--------	-------

Sifra delatnosti

<u>ID</u>	Sifra	Naziv
-----------	-------	-------

Pacijent

<u>ID</u>	IDKor	IDPol	IDRodZivZaj	IDBracnoSt	LBO
-----------	-------	-------	-------------	------------	-----

Pol

<u>ID</u>	Vrednost
-----------	----------

Roditelji zive zajedno

<u>ID</u>	Vrednost
-----------	----------

Bracno stanje

<u>ID</u>	Vrednost
-----------	----------

Podaci o porodjaju i detetu

<u>ID</u>	IDPac	Tezina	Duzina
-----------	-------	--------	--------

Tip vakcine seruma

<u>ID</u>	Vrednost
-----------	----------

Vakcina serum

<u>ID</u>	IDTipVakc	SerBr	Napomene
-----------	-----------	-------	----------

Tip bolesti

<u>ID</u>	Vrednost
-----------	----------

Tip pregleda

<u>ID</u>	Vrednost
-----------	----------

Pregled

<u>ID</u>	IDPac	IDTipPregledanja	Datum
-----------	-------	------------------	-------

Kurativa

<u>ID</u>	IDPregleda	Vrednost
-----------	------------	----------

Sifrarnik anamneza

<u>ID</u>	Vrednost
-----------	----------

Anamneza

<u>ID</u>	IDKur	IDSifAnam	Opis nalaza
-----------	-------	-----------	-------------

Nalaz pregleda

<u>ID</u>	IDAnam	Vrednost
-----------	--------	----------

Sifrarnik dijagnoze

<u>ID</u>	Sifra	NazivSis	LatNazDijag	SrpNazDijag
-----------	-------	----------	-------------	-------------

Dijagnoza

<u>ID</u>	IDNalPre g	IDSifDijag	Napomena	Status
-----------	---------------	------------	----------	--------

Sifrarnik pomagala

<u>ID</u>	Vrednost
-----------	----------

Pomagala

<u>ID</u>	IDDijag	IDSifPomag
-----------	---------	------------

Siptomatska terapija

<u>ID</u>	IDDijag	Naziv
-----------	---------	-------

Sifrarnik lekova

<u>ID</u>	IDJedMereP ak	IDJedMereK onc	Sifra	Naziv	Tip	Pakovanje	Koncentraci ja
-----------	------------------	-------------------	-------	-------	-----	-----------	-------------------

Jedinica mere

<u>ID</u>	Vrednost
-----------	----------

Terapija

<u>ID</u>	IDDijag	Datum
-----------	---------	-------

Nacin davanja injekcije

<u>ID</u>	Vrednost
-----------	----------

Injekcija

<u>ID</u>	IDDijag	IDNacDavInjekcije
-----------	---------	-------------------

Faktura

<u>ID</u>	IDPreg	IDKor	Datlzd	RokVaz	JedMereRV
-----------	--------	-------	--------	--------	-----------

Cenovnik

<u>ID</u>	Naziv	Datum
-----------	-------	-------

Tok Novca

<u>ID</u>	IDFakt	IDRadMesta	Datum	Iznos	Status
-----------	--------	------------	-------	-------	--------

Jedinica mere starosne grupe

<u>ID</u>	Vrednost
-----------	----------

Starosna grupa

<u>ID</u>	IDJMStGr	Vrednost
-----------	----------	----------

Sistematski

<u>ID</u>	IDPreg	IDStGr
-----------	--------	--------

Tip tezine

<u>ID</u>	Vrednost
-----------	----------

Tip cula

<u>ID</u>	Vrednost
-----------	----------

Tip organa

<u>ID</u>	Vrednost
-----------	----------

Sistematski odojceta

<u>ID</u>	IDSistematskog
-----------	----------------

Tip razvoja

<u>ID</u>	Vrednost
-----------	----------

Tip ishrane

<u>ID</u>	Vrednost
-----------	----------

Tip anomalije

<u>ID</u>	Vrednost
-----------	----------

Sistematski adolescenata

<u>ID</u>	IDSistematskog
-----------	----------------

Tip deformacije

<u>ID</u>	Vrednost
-----------	----------

Tip oboljenja

<u>ID</u>	Vrednost
-----------	----------

Tip ustanove

<u>ID</u>	Vrednost
-----------	----------

Tip uhranjenosti

<u>ID</u>	Vrednost
-----------	----------

Osiguranje

<u>ID</u>	RegBr	OsnovOsigt
-----------	-------	------------

Tip kategorije zdravstvene
zastite

<u>ID</u>	Vrednost
-----------	----------

Sistematski 1_6 godina

<u>ID</u>	<u>IDSistAdolesc</u>
-----------	----------------------

Sistematski osn i sred skole

<u>ID</u>	<u>IDSistAdolesc</u>	<u>IDTipUstanove</u>
-----------	----------------------	----------------------

Desavanja

<u>ID</u>	IDKor	NazivTabele	BrojReda	Opis	Napomene
-----------	-------	-------------	----------	------	----------

Zaduzivanje

<u>ID</u>	Datum
-----------	-------

Magacin

<u>ID</u>	IDAD	Naziv
-----------	------	-------

Prijemnica

<u>ID</u>	Datum
-----------	-------

Otpremnica

<u>ID</u>	Datum
-----------	-------

Cena

<u>ID</u>	<u>IDJedMer</u> <i>e</i>	Vrednost
-----------	-----------------------------	----------

Minimalna kolicina

<u>ID</u>	<u>IDJedMer</u> <i>e</i>	Vrednost
-----------	-----------------------------	----------

Dobavljac

<u>ID</u>	IDPL
-----------	------

Katalog dobavljač

<u>ID</u>	<i>IDDob</i>	Naziv	Datum
-----------	--------------	-------	-------

Narudžbenica

<u>ID</u>	<i>IDOsoblje</i>	Datum
-----------	------------------	-------

Stavka kataloga dobavljača

<u>ID</u>	<i>IDKatDob</i>	<i>IDMinKol</i>	<i>IDCena</i>	Datum
-----------	-----------------	-----------------	---------------	-------

Stavka narudžbenice

<u>ID</u>	<i>IDNar</i>	<i>IDStKatDob</i>	Kolicina
-----------	--------------	-------------------	----------

Stavka otpremnice

<u>ID</u>	<i>IDOtpr</i>	<i>IDStNar</i>	Kolicina
-----------	---------------	----------------	----------

Stavka prijemnice

<u>ID</u>	<i>IDPrij</i>	<i>IDStOtpr</i>	Kolicina
-----------	---------------	-----------------	----------

Stavka magacina

<u>ID</u>	<i>IDMag</i>	<i>IDJedMer</i>	<i>IDSifrarnik</i>	<i>IDStPrij</i>	Kolicina
-----------	--------------	-----------------	--------------------	-----------------	----------

Slabi objekti

Pristupni podaci

<u>ID</u>	<u><i>IDKor</i></u>	KorIme	Lozinka	Status
-----------	---------------------	--------	---------	--------

Adresa

<u>ID</u>	Grad	Opština	Ulica	Broj	Sprat	StLok
-----------	------	---------	-------	------	-------	-------

Bolovanje

<u>ID</u>	<u><i>IDKor</i></u>	<u><i>IDOsoblje</i></u>	<u><i>IDDijag</i></u>	DatPoc	DatZav	Status
-----------	---------------------	-------------------------	-----------------------	--------	--------	--------

Datum rođenja

<u>ID</u>	Datum
-----------	-------

Datum pregleda za izveštaj

<u>ID</u>	Datum
-----------	-------

Ovlašćena lica

<u>ID</u>	<u><i>IDPL</i></u>	<u><i>IDFL</i></u>
-----------	--------------------	--------------------

Ustanova

<u>ID</u>	<u>IDPL</u>	<u>IDUput</u>	<u>IDRadMes</u>
-----------	-------------	---------------	-----------------

Racun

<u>ID</u>	<u>IDKor</u>	<u>IDTokNovca</u>	Datum
-----------	--------------	-------------------	-------

Profesionalna obolenja

<u>ID</u>	Naziv	DatumEvidentiranja
-----------	-------	--------------------

Prebolele bolesi

<u>ID</u>	<u>IDPac</u>	<u>IDTipBolesti</u>
-----------	--------------	---------------------

Zdravstvena knjizica

<u>ID</u>	<u>IDPac</u>	BrojKnj	RegBroj	PIB
-----------	--------------	---------	---------	-----

Porodiliste

<u>ID</u>	<u>IDPodPorDet</u>	Vrednost
-----------	--------------------	----------

Porodjaj

<u>ID</u>	<u>IDPodPorDet</u>	Vrednost
-----------	--------------------	----------

Kogenitalne anomalije

<u>ID</u>	<u>IDPodPorDet</u>	Vrednost
-----------	--------------------	----------

Status deteta

<u>ID</u>	<u>IDPodPorDet</u>	Opcija	Vrednost
-----------	--------------------	--------	----------

Fiksni telefon

<u>ID</u>	<u>IDTel</u>	Vrednost
-----------	--------------	----------

Mobilni telefon

<u>ID</u>	<u>IDTel</u>	Vrednost
-----------	--------------	----------

Elektronska posta

<u>ID</u>	<u>IDKontPod</u>	Vrednost
-----------	------------------	----------

Oboljenja od veceg socio
med znacaja

<u>ID</u>	<u>IDPac</u>	Vrednost
-----------	--------------	----------

Poremecaji psihofiz statusa

<u>ID</u>	<u>IDPac</u>	Vrednost
-----------	--------------	----------

Kategorija zdravstvene
zastite

<u>ID</u>	<u>IDPac</u>	<u>IDTipKatZdrZas</u>	PrviPodatak	Promena
-----------	--------------	-----------------------	-------------	---------

Laboratorija

<u>ID</u>	<u>IDPL</u>
-----------	-------------

Davanje injekcija

<u>ID</u>	<u>IDInjekcija</u>	<u>IDOsoblje</u>	Datum
-----------	--------------------	------------------	-------

Stavka fakture

<u>ID</u>	<u>IDFakt</u>	<u>IDStCen</u>	Kolicina
-----------	---------------	----------------	----------

Stavka cenovnika

<u>ID</u>	<u>IDCenovnik</u>	<u>IDCena</u>	Kolicina
-----------	-------------------	---------------	----------

Tezina

<u>ID</u>	<u>IDSist</u>	<u>IDTipTezine</u>	Vrednost
-----------	---------------	--------------------	----------

Visina

<u>ID</u>	<u>IDSist</u>	Vrednost
-----------	---------------	----------

Pluca

<u>ID</u>	<u>IDSist</u>	Vrednost
-----------	---------------	----------

Srce

<u>ID</u>	<u>IDSist</u>	Vrednost
-----------	---------------	----------

Cula

<u>ID</u>	<u>IDSist</u>	<u>IDTipCula</u>	Vrednost
-----------	---------------	------------------	----------

Psihicki razvoj

<u>ID</u>	<u>IDSist</u>	Vrednost
-----------	---------------	----------

Alergijska bolest

<u>ID</u>	<u>IDSist</u>	<u>IDTipOrgana</u>	Vrednost
-----------	---------------	--------------------	----------

Eritrociti

<u>ID</u>	<u>IDSist</u>	Vrednost
-----------	---------------	----------

Hemoglobin

<u>ID</u>	<u>IDSist</u>	Vrednost
-----------	---------------	----------

Druge patoloske promene

<u>ID</u>	<u>IDSist</u>	Vrednost
-----------	---------------	----------

Primedba

<u>ID</u>	<u>IDSist</u>	Vrednost
-----------	---------------	----------

Prematorijus_Nedonosenost

<u>ID</u>	<u>IDSistOdojceta</u>	Vrednost
-----------	-----------------------	----------

Razvoj statickih funkcija

<u>ID</u>	<u>IDSistOdojceta</u>	<u>IDTipRazvoja</u>	Vrednost
-----------	-----------------------	---------------------	----------

Rahitis

<u>ID</u>	<u>IDSistOdojceta</u>	Vrednost
-----------	-----------------------	----------

Jetra_Slezina

<u>ID</u>	<u>IDSistOdojceta</u>	Vrednost
-----------	-----------------------	----------

Ishrana

<u>ID</u>	<u>IDSistOdojceta</u>	<u>IDTipIshrane</u>	Vrednost
-----------	-----------------------	---------------------	----------

Koza

<u>ID</u>	<u>IDSistOdojceta</u>	<u>IDSist_1_6god</u>	Vrednost
-----------	-----------------------	----------------------	----------

Kogenitalne anomalije
sistematski

<u>ID</u>	<u>IDSistOdojceta</u>	<u>IDSist_1_6god</u>	<u>IDTipAnomalije</u>	Vrednost
-----------	-----------------------	----------------------	-----------------------	----------

Deformacija kostura

<u>ID</u>	<u>IDSistAdolesc</u>	<u>IDTipDeformacije</u>	Vrednost
-----------	----------------------	-------------------------	----------

Govorna mana

<u>ID</u>	<u>IDSistAdolesc</u>	Vrednost
-----------	----------------------	----------

Crevni paraziti

<u>ID</u>	<u>IDSistAdolesc</u>	Vrednost
-----------	----------------------	----------

Zub

<u>ID</u>	<u>IDSistAdolesc</u>	<u>IDTipObolenja</u>	Vrednost
-----------	----------------------	----------------------	----------

Tonzila

<u>ID</u>	<u>IDSistAdolesc</u>	Vrednost
-----------	----------------------	----------

Limfna zleza

<u>ID</u>	<u>IDSist_1_6god</u>	Vrednost
-----------	----------------------	----------

Kategorisano

<u>ID</u>	<u>IDSist_1_6god</u>	Vrednost
-----------	----------------------	----------

Cistoca

<u>ID</u>	<u>IDSist_OS_Sr</u>	Vrednost
-----------	---------------------	----------

Vasljivost

<u>ID</u>	<u>IDSist_OS_Sr</u>	Vrednost
-----------	---------------------	----------

Telesno drzanje

<u>ID</u>	<u>IDSist_OS_Sr</u>	Vrednost
-----------	---------------------	----------

Telesna uhranjenost

<u>ID</u>	<u>IDSist_OS_Sr</u>	<u>IDTipUhranjenosti</u>	Vrednost
-----------	---------------------	--------------------------	----------

Telesna razvijenost

<u>ID</u>	<u>IDSist_OS_Sr</u>	Vrednost
-----------	---------------------	----------

Usta

<u>ID</u>	<u>IDSist_OS_Sr</u>	Vrednost
-----------	---------------------	----------

Zdrelo

<u>ID</u>	<u>IDSist_OS_Sr</u>	Vrednost
-----------	---------------------	----------

Stitna zleza

<u>ID</u>	<u>IDSist_OS_Sr</u>	Vrednost
-----------	---------------------	----------

Pritisak

<u>ID</u>	<u>IDSist_OS_Sr</u>	Vrednost
-----------	---------------------	----------

Pubertet

<u>ID</u>	<u>IDSist_OS_Sr</u>	Vrednost
-----------	---------------------	----------

Prva menstruacija

<u>ID</u>	<u>IDSist_OS_Sr</u>	Vrednost
-----------	---------------------	----------

Albumen u urinu

<u>ID</u>	<u>IDSist_OS_Sr</u>	Vrednost
-----------	---------------------	----------

Sposobnost za fiz vasp

<u>ID</u>	<u>IDSist_OS_Sr</u>	Vrednost
-----------	---------------------	----------

Uspeh u skoli

<u>ID</u>	<u>IDSist_OS_Sr</u>	Vrednost
-----------	---------------------	----------

Smart kartica

<u>ID</u>	<u>IDPac</u>	SerBr	Datlzd	Status
-----------	--------------	-------	--------	--------

Stavka zaduzivanja

<u>ID</u>	<u>IDZad</u>	<u>IDUstanova</u>	<u>IDStMag</u>	<u>IDOsob</u>	Kolicina
-----------	--------------	-------------------	----------------	---------------	----------

M:N Veze

StarateljSrodnik

<u>IDFLPAC</u>	<u>IDFLSTR</u>	Zanimanje	ZdravStanje
----------------	----------------	-----------	-------------

Korisnik Kontakt podatak

<u>IDKor</u>	<u>IDKontPod</u>
--------------	------------------

Fizicko lice Osiguranje

<u>IDFL</u>	<u>IDOsig</u>
-------------	---------------

Korisnik Adresa

<u>IDKor</u>	<u>IDAdresa</u>
--------------	-----------------

Fizicko lice Datum rođenja

<u>IDFL</u>	<u>IDDatRodj</u>
-------------	------------------

Izvestaj Datum pregleda za izv

<u>IDIzv</u>	<u>IDDatIzv</u>
--------------	-----------------

Pravno lice adresa

<u>IDPL</u>	<u>IDAdr</u>
-------------	--------------

Prethodno lečenje pacijenta

<u>IDPac</u>	<u>IDUstanova</u>	PocDat	KrDat
--------------	-------------------	--------	-------

Pacijent Vakcina

<u>IDPac</u>	<u>IDVakc</u>
--------------	---------------

Pacijent bira lekara

<u>IDPac</u>	<u>IDOsoblje</u>	Datum	Status
--------------	------------------	-------	--------

Lekar leci pacijenta

<u>IDOsoblje</u>	<u>IDPac</u>
------------------	--------------

Injekcija StMagacin

<u>IDInjekcija</u>	<u>IDStMagacin</u> <u>a</u>	Kolicina	Interval
--------------------	--------------------------------	----------	----------

Terapija StMagacin

<u>IDTerapija</u>	<u>IDStMagacina</u>	Hronicno	Kolicina	NacinKonzumacije	BrojKonzUJedMere	JedMere
-------------------	---------------------	----------	----------	------------------	------------------	---------

Dijagnoza Laboratorija

<u>IDDijag</u>	<u>IDLab</u>	Opis
----------------	--------------	------

Slika 31. Prikaz sastava relacionog modela radne baze podataka - prva normalna forma

11.1.2. Relacioni model baze podataka za bezbednost

Bezbednost

<u>ID</u>	PrDeoKlj	DrDeoKlj	IDTabSmart	Status
-----------	----------	----------	------------	--------

Slika 32. Prikaz sastava relacionog modela baze podataka za bezbednost – prva normalna forma

12. Rečnik podataka

Rečnik podataka predstavlja skup podataka o podacima koji se koriste u analiziranom sistemu.

Rečnik podataka predstavlja opis strukture, a takođe i formata podataka u bazi podataka informacionog sistema.

Sastoji se od polja, koja su primitivna i kao takva dalje ne mogu da dekomponuju čuvajući prvobitni njihov smisao.

Polja u sebi sadrže vrednosti. Tip vrednosti koje će polje moći da čuva određuje se domenom. Domeni su ograničenja koja po tipu ograničavaju vrednost koje polje može da sadrži, dok, ograničenja sadrže detaljnija ograničenja koja definišu i granične vrednosti koje neko polje sme da sadrži.

Zbog potreba informacionog sistema, koristiće se dve baze podataka.

1. Baza za rad.
2. Baza za šifre.

Prvo će biti prikazan rečnik podataka baze za rad a potom baze za šifre.

Zbog ograničenog obima stranica rada, u radu je prikazana tabela Korisnik baze podataka ISZdravstvo, dok prikaz ostalih tabela iz pomenute baze podataka, kao i iz baze podataka ISZdravstvoSifra je prikazan u prilogu rečnik podataka.

12.1. Rečnik podataka baze za rad

Naziv podatka	Drugi naziv podatka	Domen	Ograničenja	Napomene
ID	ID	Int	Not null	Primarni ključ
ID tabele Fizicko lice	IDFL	Int	Not null	Spoljašnji ključ
Uloga	Uloga	varchar	Not null	
Status	Uloga	varchar	Not null	

Tabela 1. Korisnik

13.C#

Radi lakšeg razumevanja, u kratkim crtama, biće predstavljen programski jezik C#. Preporučuje se pre čitanja rada, da čitaoci budu upoznati sa pravilima pisanja u C# programskom jeziku kao i sa objektno orientisanim načinom programiranja.

C# je relativno mlad programski jezik koji je najavljivan od strane kompanije Microsoft 2000-te godine, ali je prikazan široj javnosti 2002. godine. Programski jezik je objektno orientisan. Objektna orijentacija jezika mu je omogućena korišćenjem klasa koje mogu da se shvate kao šabloni po kojima mogu da se prave skupovi koji mogu da čuvaju podatke koji su svojstveni skupu kao celini. Ukoliko se želi, da se koristi u programu objekat, koji će biti reprezentacija konkretnog fizičkog lica, potrebno je da se napravi klasa koja će biti korišćena prilikom formiranja objekta fizičkog lica u kome mogu da se čuvaju odgovarajući podaci kao što su, ime, prezime, jmbg i slično.

Prava pristupa mogu da se definišu za bilo koji deo u programskom jeziku C#. Ona obezbeđuju primenjivanje pravila polimorfizma koji je jedan od elemenata objektno orijentisanog načina programiranja.

Najčešće korišćena pravila pristupa su:

- **Public** – predstavlja pravilo pristupa koje omogućava da tako deklarisanom članu može da bude pristupano i iz drugih klasa i modula a ne samo iz klase u kojoj je formiran.
- **Private** – predstavlja pravilo pristupa koje omogućava da tako deklarisanom članu može da bude pristupano samo iz klase u kojoj je formiran.
- **Protected** – predstavlja pravilo koje omogućava tako deklarisanom članu da bude pristupano iz klase u kojoj je formiran, ali i iz klasa koje nasleđuju datu klasu.
- **Internal** – predstavlja pravo pristupa koje omogućava pristup samo klasama koje se nalaze u istom modulu kao i promenljiva.

Promenljive predstavljaju čuvar odgovarajućih vrednosti u memoriji računara. Te vrednosti shodno potrebama mogu da se menjaju. Shodno tipu promenljive mogu da budu proste, tj. promenljive koje ne zahtevaju korišćenje objekata radi njihovog formiranja i objektna koje to zahtevaju.

Radi obezbeđivanja objektno orijentisanog pojma enkapsulacije preporučuje se korišćenje svojstava. Svojstva su delovi koda koji liče jako na metode ali imaju ulogu postavljanja i pribavljanja vrednosti promenljivih od strane drugih objekata ili promenljivih. Preporučljivo korišćenje svojstava je ukoliko postoji potreba za pristupom ili menjanjem promenljive od strane druge promenljive koja se ne nalazi u istoj klasi ili objekta koji može da potiče i iz iste klase.

Ukoliko postoji potreba za primenom neke logičke celine, kao što je izračunavanje vrednosti određenog broja promenljivih i prosleđivanje te vrednosti, onda treba da se koriste metode. Metode, kao i većina celina u C# jeziku, moraju da imaju svoja prava pristupa. Pored prava pristupa metodi zahtevaju da se odredi njihov povratni tip. Povratni tip može da bude bilo koji tip promenljive. Metode omogućavaju da im se proslede parametri ili mogu da koriste, raspoložive za njih, parametre u klasi u kojoj su napravljeni.

Petlje treba da postoje u delovima koda u kojima postoji potreba za ponavljanjem odgovarajućeg zadatka, kao na primer, dobavljanja imena korisnika iz baze podataka. Postoje više vrsti petlji, koje su predodređene njihovom namenom. Namena nije odlučujući faktor u izboru petlje, jer sve petlje mogu da podmiru iste zahteve, ali shodno nameni neke petlje su brže u nekim zahtevima dok su u drugim sporije od ostalih petlji.

Petlje mogu da budu:

- **For** – njena namena je prvenstveno dobavljanje podataka iz skupa podataka sa konačnim brojem članova. Njena prednost je lakoća pristupa potrebnom članu u skupu.

- Foreach – namenjena je lakom pristupu članovima skupa koji nije određen konačnim brojem članova. Otežan joj je pristup članu na konkretnoj poziciji u skupu, primer, vrednost člana na poziciji broj 3 u skupu.
- While – petlja namenjena da koristi resurse računara sve dok se određeni uslov ne ispuni ili se ne iskoriste svi resursi iz računara. Kod while kao i njene modifikovane verzije do-while postoji opasnost postavljanja beskonačne petlje koja kao svoju posledicu ima pad dela ili celog informacionog sistema ili programa u kome se koristi. Razlika između while i do-while petlje je u poziciji uslova u odnosu na sadržaj koji se izvršava sve dok nije uslog ispunjen.

Blok predstavlja skup naredbi. Blok obično počinje i završava se sa velikim zagradama. Blokovi su važni delovi C# programskog jezika koji obezbeđuju polje važnosti promenljivih. Promenljive koje su deklarisanе u bloku nisu vidljive van njega. Deklarisanje promenljivih znači da im je definisan tip i ime. Incijalizacija se odnosi da je promenljivoj pridružena i vrednost.

14.ASP .NET MVC

Za ulogu sprovođenja u delo projektovanog informacionog sistema zdravstvene ustanove izabran je ASP .NET MVC platforma. Razlog takve odluke zasniva se na njegovim mogućnostima radi obezbeđivanja što lakše modularnosti i sigurnosti informacionog sistema. Programski jezik na kojem se zasniva programiranje je C#.

Logika koja je primenjivana kod programiranja zasniva se na MVC (Model-View-Controller) projektnom šablonu. Postoje dva tipa najčešćih programa koji se mogu da programiraju na ASP .NET MVC platformi.

- ASP .NET API se sastoji od API kontrolera, mada može da poseduje i obične asp kontrolere, kao i poglede. Njegova glavna uloga je obezbeđivanje komunikacije između ASP .NET MVC aplikacije i baze podataka. Kontroleri, slobodnije shvaćeno su klase koje poseduju metode koji imaju povratni odgovarajući tip tako da mogu da se formiraju odgovori na poslate zahteve, ukoliko je reč o API server kontrolerima. U daljem tekstu za ASP .NET API biće korišćen i sinonim API server.
- ASP .NET MVC aplikacija koja predstavlja primarnu komunikaciju između API servera i korisnikovih naredbi. ASP .NET MVC aplikacija u sebi pored kontrolera i pogleda može da sadrži i API kontrolere ali onda se time narušava smisao međusobne komunikacije između API servera i ASP .NET MVC aplikacije ukoliko se API server koristi. Kontroleri na strani ASP servera imaju ulogu da aktiviraju poglede, prosleđuju pogledima odgovarajuće tokove podataka, kao i da preuzimaju od pogleda odgovarajuće podatke. U daljem tekstu za ASP .NET MVC aplikaciju biće korišćen sinonim ASP server.

14.1. MVC projektni šablon

MVC projektni šablon se koristi od kasnih 1970-tih. Nastao je kao posledica pokušaja da se organizuju rani programi sa grafikom. Uopšteno govoreći, kada se primenjuje MVC projektni šablon, informacioni sistem treba da se sastoji od minimalno 3 celine. Celine u sebi mogu da imaju podskupove. Takođe celine mogu da dele određene podskupove međusobno.

MVC je skraćenica od Model-View-Controller, odnosno Model-Pogled-Kontroler.

- Model – predstavlja skup klasa sa svojim promenljivima, metodima, i javnim svojstvima. Model omogućava čuvanje vrednosti odgovarajućih podataka pomoću objekata do pogleda od kontrolera i od kontrolera do pogleda.
- Pogled – zadužen je za prikazivanje vrednosti koje su dobijene od kontrolera pomoću objekta iz modela u odgovarajućim poljima i prosleđivanje unetih, od korisnika vrednosti ka kontroleru uz pomoć objekta iz modela.
- Kontroler – predstavlja skup klasa koji u principu poseduju samo metode. Mogu da poseduju i promenljive i svojstva ali onda time narušavaju MVC koncept. Uloga kontrolera je obrada podataka, kao što su određena izračunavanja ali i čuvanje podataka u odgovarajućim skladištima podataka kao i dobavljanje podataka iz skladišta.

Primena MVC šablona na ASP .NET MVC platformi zasniva se na promeni logike. Promena se zasniva u povratnoj informaciji od strane korisnika koja ne dolazi do pogleda već do kontrolera.

15. Tehnološki sastav informacionog sistema

Informacioni sistem „Zdravstvo“ gledano po elementima, podeljen je na tri celine. Tri celine su:

- Server baze podataka
- API server
- ASP server

Na serveru baze podataka čuva se baza podataka, sa pripadajućim procedurama i ulogama. API server i ASP server se sastoje od više modula. Neke od modula oni međusobno dele. Moduli su:

- DatabaseLayer
- Model
- SifrovanjeDesifrovanje
- Transfer
- XMLserializer

API server koristi DatabaseLayer, Model, SifrovanjeDesifrovanje i XMLserializer.

ASP server koristi Model, SifrovanjeDesifrovanje, Transfer i XMLserializer.

15.1. Server baze podataka

Server baze podataka je Microsoft SQL server 2008. Server se nalazi na računaru koji ima instaliran Windows server 2008 R2 datacenter u cilju povećane stabilnosti sistema. Na serveru baze podataka omogućen je SQL login način sa ssl vezom obezbeđenom pomoću sertifikata. Skladište podataka se sa sastoji od dve baze podataka. Jedna baza podataka ISZdravstvo služi za čuvanje podataka o pacijentima, lečenjima, finansijama i zaposlenima. Druga baza ISZdravstvoSifra služi za pružanje ključeva i njihovo ažuriranje za podatke iz baze podataka ISZdravstvo koji to zahtevaju. Napravljena je uloga pomoću koje i njene lozike se omogućuje pristup bazama podataka. U obe baze za skladištenje, izmenu i prikaz podataka koriste se uskladištene procedure. Većinska logika u domenu računanja je prenešena na module dok su procedure zadužene samo za prosta računanja.

15.1.1. Upiti

Podela upita može da se izvrši po više kriterijuma. Najčešći je po njihovoj svrsi. Po svrsi upiti mogu da budu:

- Insert - služe za unos podataka u novi red u odgovarajućoj tabeli. Opšti prikaz insert naredbe je: insert into imeTabele(imeKolona,...) values(dopremljeneVrednosti,...); Pojam imeKolona je opcioni i ukoliko on nije prisutan onda se podrazumevano počinje unos od prve kolone. Preporučljivo je korišćenje pojma imeKolona jer neke kolone poseduju polja koja prilikom unosa automatski od sql servera dobijaju brojčanu vrednost.
- Update - služe za ažuriranje podataka u traženom redu, u odgovarajućoj tabeli. Za izvršavanje update naredbe poželjno, je pored podataka koj žele da se promene, dostavi i jedinstveni identifikator koji određuje u kom redu ili redovima će podaci biti promenjeni. Ukoliko se ne dostavi identifikator, podrazumevano je da će ažuriranje biti obavljeno na svim redovima tražene tabele. Opšti prikaz update naredbe je: update imeTabele set imeKolone = dopremljenaVrednost,... where imeTabele.imeKolone = dopremljeniUslov.

- Delete služe za brisanje podataka iz traženog reda odgovarajuće tabele u bazi podataka. Kao i za update naredbe poželjno je da se dostavi jedinstveni identifikator. Opšti prikaz delete upita je: delete from imeTabele where imeTabele.imeKolone = dopremljeniUslov.
- Select služe za dostavljanje podataka iz baze podataka podnosiocu zahteva za njima. Opšti prikaz select upita: select imeKolona,... from imeTabele,... where imeTabele.imeKolone = dopremljeni uslov

1.1.1.1 Filteri

Filteri se koriste radi postavljanja uslova po kome će biti filtriran sadržaj shodno upitu u kome se nalaže. Najčešći filter koji se u ovom radu koristi je filter where.

Filter where ima ulogu omogućiti izmenu podataka ili njihovo dopremanje podnosiocu zahteva, sve dok uslov koji je postavljen u filteru ispunjen. Filteri ne mogu da se primenjuju za insert upit. Insert upit podrazumeva da se podaci unose u prvi slobodni red u traženoj tabeli.

15.1.2. Procedure

Procedure u SQL bazi podataka se odnose na skupove naredbi koje se izvršavaju kao skup, izvršavajući se jedna za drugom. Procedure mogu, kao povratnu informaciju, da pružaju pozivaocu: brojanu, slovnu vrednost ili povratni skup podataka u slučaju select upita. Opšti prikaz procedure je:

```
Create procedure imeProcedure
(
    @imeParametra tipParametra,
    ...
)
As
Declare @imeParametra2 tipPodatka2
...
Begin
    Naredba odnosno upit
End
```

Parametar @imeParametra je opcioni. Ukoliko on nije pružen, onda nije moguće da se postavi uslov po spoljnim podacima za select upite, niti su mogući zasebni insert, update upiti. Insert i update upiti su mogući u tom slučaju jedino ako im prethodi select upit koji doprema interno podatke iz drugih tabela. Parametri mogu da budu ulazni i izlazni. Ulazni parametri se koriste za dopremanje podataka u proceduru. Izlazni služe za dopremanje odgovora podnosiocu zahteva. Izlazni parametri nisu neophodni za select upite jer select upit sam po sebi vraća skup. Skup može da sadrži podatke ili da bude prazan, i to zavisi od uslova koji je postavljan u where segmentu select upita.

Parametar @imeParametra2 je opcion. Takav parametar se koristi interno u proceduri u kojoj je definisan da bi čuvao određene podatke kao što je broj u primarnom ključu ukoliko je prost primarni ključ. Često se koristi u izvršavanju insert naredbe. Ukoliko je slučaj sa insert naredbom, onda mora da se koristi sistemska promenljiva @@IDENTITY koja pruža podatak u o broju koji je postavljen kao primarni ključ. Ukoliko je prost primarni ključ on može da se postavi da prilikom unosa novog zapisa postavlja vrednost koja je za 1 veća od prethodne, ukoliko u uslovu za njegovo postavljanje nije drugačije naznačeno. Parametar

@imeParametra2 se obično koristi radi prosleđivanja podatka između naredbi u datoj proceduri.

Ukoliko postoje višestruke naredbe u proceduri, poželjno je da se svaka naredba, bez obzira da li je insert, update, delete ili select postavi u svoj begin end deo, u sklopu begin end dela same procedure u kojoj se nalazi. Razlog za takav postupak je obezbeđivanje da ne počinje sledeća naredba dok prethodna nije završena. U slučaju da ne postoje pod celine, odnosno pod blokovi begin end, postoji mogućnost neizvršavanja procedure zbog javljanja greški ukoliko jedna naredba nije završena ali je njen rezultat neophodan za izvršavanje sledeće naredbe koja je upravo počela sa izvršavanjem.

15.2. Sertifikaciono telo

Sertifikaciono telo iako nije deo informacionog sistema, ono obezbeđuje sigurnost u radu sa korisnicima i radu između celina informacionog sistema. Sertifikaciono telo služi radi izdavanja sertifikata zbog ostvarivanja HTTPS veze između potrebnih entiteta. Uloga sertifikacionog tela je da entitet kome je izdat sertifikat je ustvari ono što on tvrdi da jeste. Uobičajeno je da sertifikaciona tela sarađuju sa na primer finansijskim institucijama, radi dobavljanja podataka o podnosiocu zahteva za sertifikat (broj kreditne kartice i slično). U slučaju da sertifikaciono telo nije telo od poverenja kao što može da bude sertifikaciono telo koje se koristi u intranet mreži, potrebno je da se njegovi sertifikati instaliraju i Trusted server certificates odeljak. Sertifikaciono telo je za demonstrativne potrebe informacionog sistema instalirano na računaru kome se nalazi Windows server 2008 R2 datacenter.

15.3. API server moduli

DatabaseLayer modul koristi se radi komunikacije sa bazama podataka. Model se koristi radi prijema podataka sa ASP servera kao i slanje odgovora. Takođe Model se koristi u modulu DatabaseLayer radi prosleđivanja i prijema podataka. SifrovanjeDesifrovanje modul koristi se radi dešifrovanja dobijenih podataka i šifrovanje podataka koji se šalju. XMLserializer je modul koji je zadužen za serializovanje objekata u XML string, kao i njihovo deserijalizovanje.

API server od ASP servera dobija XML string koji mora da deserijalizuje u objekat koji šalje u modul DatabaseLayer radi dalje obrade. Ukoliko je potrebno, ne šalje se ceo objekat već samo njegovi elementi, shodno potrebama i mogućnostima.

Ukoliko u deserijalizovanom objektu postoji šifriran tekst, taj tekst se dešifruje pomoću već dopremljenog ključa. U cilju smanjenja zauzetosti resursa API kontrolera API servera, sifrovani podaci se ponekad ne dešifruju u metodu već u modulu DatabaseLayer.

Metode koje obrađuju zahteve pristigle od ASP servera, pozivaju DatabaseLayer modul i vraćaju odgovor pozivaocu, mogu da se podele u dva oblika.

- Metode koje ne obrađuju primljeni objekat i odgovor šalju objekat sa traženim podacima, npr. Lista svih korisnika.
- Metode koje obrađuju poslat, od pozivaoca objekat radi obrade njegovih podataka ili prosleđivanja u celini objekta DatabaseLayer modulu i slanje odgovora o uspešnosti slanja

Opšti oblik metode koje ne obrađuju primljeni objekat je:

```
[HttpPost]
public HttpResponseMessage imeMetode()
{
```

```

        List<klasaModela> zahtevLista = new List<klasaModela>();
        zahtevLista
XMLserializer.Serialize.Deserializovanje<List<Cenovnik>>(preradjeno, zahtevLista);
        string odgovor = " ";
        odgovor = new KlasaDatabaseLayerModula().metodaKlase(zahtevLista[0]);

        HttpResponseMessage hrm = new HttpResponseMessage();
        hrm.Content = new StringContent(odgovor);
        return hrm;
    }

```

Sve metode rade sa post zahtevima radi ne otkrivanja u konekcionom stringu osetljivih podataka. Pravi se lista koje će biti vraćena u odgovoru, koja se posle obrađuje na ASP serveru.

Metode koje obrađuju objekat poslat od pozivaoca:

```

[HttpPost]
public HttpResponseMessage imeMetode()
{
    HttpContent httpObjekat = Request.Content;
    Task<String> s = httpObjekat.ReadAsStringAsync();
    String objekat = s.Result;
    String preradjeno = "";
    preradjeno = objekat;
    preradjeno = XMLserializer.IspravkaXMLString.IspravkaXML('[', '<',
objekat);
    preradjeno = XMLserializer.IspravkaXMLString.IspravkaXML(']', '>',
preradjeno);

    List<klasaModela> zahtevLista = new List<klasaModela>();
    zahtevLista
XMLserializer.Serialize.Deserializovanje<List<Cenovnik>>(preradjeno, zahtevLista);
    string odgovor = " ";
    odgovor = new KlasaDatabaseLayerModula().metodaKlase(zahtevLista[0]);

    HttpResponseMessage hrm = new HttpResponseMessage();
    hrm.Content = new StringContent(odgovor);
    return hrm;
}

```

Prilikom prijema objekata, pre deserijalizacije, obavezno je da se sporni karakteri koji nisu mogli u izvornom obliku da se pošalju, da se vrate u izvorno stanje, da bi deserijalizacija XML stringa bila uspešna.

15.4. ASP server moduli

SifrovanjeDesifrovanje model služi da bi šifrovao ili dešifrovao objekte shodno pribavljenom ključu radi prosleđivanja takvih podataka na dalju obradu. XMLserializer modul ima ulogu da serijalizuje objekte koji se šalju ka API serveru i da po potrebi deserijalizuje objekte koji se od API servera vraćaju. Serializovanje i deserijalizovanje objekata se vrši pomoću modula Model u kome se nalaze klase sa kojima se objekti formiraju. Slanje objekata ka API serveru vrši se pomoću Transfer modula. U Transfer modulu su definisani pomoćni objekti koji su neophodni za slanje XML stringa ka API serveru, kao i adresa, i port API servera.

Svi unosi u informacionom sistemu prolaze kroz dve vrste provera. Prva je provera na klijentskoj strani za polja pored kojih je zvezdica (*). Radi povećane sigurnosti radi se i serverska provera unetih podataka, za sva potrebna polja.

Opšti oblik segmenta koja se koristi u informacionom sistemu, za slanje zahteva i objekata u njima ka API serveru, ukoliko se prosleđuje objekat radi unosa novog ili izmene postojećeg zapisa je:

```

HttpResponseMessage odgovor =
Transfer.Slanje.slanjePOST<klasaModela>(objekatKlaseModela,
@"ImeKontrolera/ImeMetoda");
if (odgovor.IsSuccessStatusCode)
{
    Task<String> s = odgovor.Content.ReadAsStringAsync();
    String rezultat = s.Result;
    @ViewBag.odgovor = rezultat;
}

```

Obavezno je navođenje kontrolera kao i metoda u njemu koji će biti aktiviran kada zahtev stigne do API servera. Ukoliko je odgovor pozitivan (odgovor.IsSuccessStatusCode), odgovor se obrađuje i prosleđuje preko objekta @ViewBag pogledu gde se odgovor prikazuje.

Opšti oblik segmenta ukoliko je prikaz postojećeg skupa zapisa je:

```

List<klasaModela> listaOdgovor = new List< klasaModela >();
HttpResponseMessage odgovor = Transfer.Slanje.slanjePOST< klasaModela
>(objekatKlaseModela, @"ImeKontrolera/ImeMetoda");
if (odgovor.IsSuccessStatusCode)
{
    Task<String> s = odgovor.Content.ReadAsStringAsync();
    String rezultat = s.Result;
    String preradjeno = "";
    preradjeno = XMLserializer.IspravkaXMLString.IspravkaXML('[', '<',
rezultat);
    preradjeno = XMLserializer.IspravkaXMLString.IspravkaXML(']', '>',
preradjeno);

    listaOdgovor =
XMLserializer.Serialize.Deserializovanje<List<klasaModela>>(preradjeno, listaOdgovor);
}
return lista;

```

Ukoliko je uspešno slanje i prijem zahteva, objekat koji se očekuje kao odgovor mora prvo da se deserijalizuje. Pre deserijalizacije sporni karakteri koji su bili promenjeni moraju da se vrate u prvobitni oblik. Posle uspešne deserijalizacije objekat se dodaje listi. Potom se lista prosleđuje na dalju obradu.

15.5. Moduli

Moduli se primenjuju radi smanjenja ponovnog kucanja istih funkcionalnosti i obezbeđivanja saobraznosti ponašanja API servera i ASP servera u ovom slucaju. Moduli su pravljani sa ciljem obezbeđivanja lakoće funkcionisanja i njihovog korišćenja uz obezbeđivanje konzistentnosti podataka i sigurnosti.

15.5.1. DatabaseLayer (DAL)

Modul DatabaseLayer koristi kao pomoćne module Model i SifrovanjeDesifrovanje. DatabaseLayer ili skraćeno DAL ima ulogu prenosa podataka do baze podataka i dopremanje podataka iz nje. Svoj zadatak izvršava pomoću metoda koji u sebi sadrže naredbe iz C# jezika iz modula System.Data i System.Data.SqlClient.

Opšti zapis stringa za uspostavljanje veze sa bazom podataka ISZdravstvo ili ISZdravstvoSifra je:

```
internal static string VEZA = "Data Source=ImeRačunara;Initial Catalog=NazivBazePodataka;User ID=KorisničkoIme;Password=Lozinka;Encrypt=True";
```

Neophodno je da se koristi ime računara a ne njegova IP adresa, ukoliko se koristi šifrovana veza pomoću sertifikata. U suprotnom može da se ostvari veza sa serverom baze podataka ali ne može da se koristi sertifikat. Deo konekcionog stringa Encrypt=true, određuje da se koristi sertifikat prilikom komunikacije sa serverom baze podataka. Ukoliko se ne želi da se koristi sertifikat deo konekcionog stringa Encrypt=true treba da se izostavi ili promeni na Encrypt=false. Konekcioni string po pravima pristupa postavljen je na internal jer ne postoji razlog da se omogući pristup njemu van modula koji je zadužen za komunikaciju sa bazom podataka.

U slučajevima korišćenja koji zahtevaju unos novog zapisa u bazu podataka ili izmenu postojećih podataka koriste se transakcije ili TransactionScope objekti radi obezbeđivanja transakcija kada se kontaktiraju više baza podataka, i rezultat unosa u jednu bazu podataka je neophodan ulazni podatak kod unosa u drugu bazu podataka.

U slučajevima koji zahtevaju dopremanje podataka iz baze podataka ne koriste se procedure, radi smanjenja zauzetosti resursa.

U svim slučajevima naredbe se izvršavaju u try catch blokovima, koji imaju ulogu, da ukoliko se pojavi izuzetak u try bloku, naredbe u try bloku prekidaju se sa izvršavanjem i počinje izvršavanje catch bloka da bi se izuzetak obradio i sprečio neplanirani prekid rada sistema.

1.1.1.2 Unos novog ili promena postojećeg zapisa u bazi podataka

Najčešći oblik metode za unos novog ili izmene postojećeg zapisa je:

```
public string noviZapis(klasaModela objekatKlase)
{
    string odgovor = "";
    using (SqlConnection konekcija = new SqlConnection(Baza.VEZA))
    {
        SqlCommand komanda = new SqlCommand();

        try
        {
            konekcija.Open();
            SqlTransaction trasaction = konekcija.BeginTransaction();

            komanda.Connection = konekcija;
            komanda.CommandText = "ImeProcedureUBaziPodataka";
            komanda.CommandType = CommandType.StoredProcedure;
            komanda.Transaction = trasaction;

            komanda.Parameters.Add(new SqlParameter("@imeParametra11",
objekatKlase. promenljiva11));
            komanda.Parameters.Add(new SqlParameter("@imeParametra12",
objekatKlase. promenljiva12));
            komanda.ExecuteNonQuery();
        }
    }
}
```

```

        trasaction.Commit();
        odgovor = "Подаци су сачувани.";
    }
    catch (Exception ex)
    {
        odgovor = "Настала је грешка: " + ex.Message;
    }
}
return odgovor;
}

```

Svaka metoda koja se koristi za komunikaciju sa bazom podataka vraća odgovor pozivaocu, bez obzira da li je odgovor skup, podataka, broj, ili rečenica. U metodi za unos podataka formira se objekat SqlConnection koji je zadužen za stvaranje konekcije sa bazom podataka na serveru baze podataka, shodno konekcionom stringu koji se sadrži u promenljivoj VEZA. SqlConnection objekat se koristi u using bloku, koji omogućava automatsko prekidanje veze sa bazom podataka kada se dođe do kraja using bloka sa izvršavanjem naredbi. Objekat SqlConnection se dodeljuje objektu SqlCommand u promenljivoj Connection, jer je SqlCommand objekat zadužen za obavljanje prosleđivanja parametara u bazu podataka i obezbeđivanja transakcije.

U try bloku obavezno je otvaranje veze sa bazom podataka. Posle uspešnog otvaranja veze stvara se objekat Transaction pozivom metode iz SqlCommand objekta BeginTransaction(). Posle povezivanja sa objektom zaduženim za transakcije unosi se naziv procedure u bazi podataka koja treba da se pozove i kojoj trebaju potrebni parametri da se proslede. Ukoliko se koriste procedure obavezno je da se postavi tip komande na CommandType.StoredProcedure, iz razloga da bi se znalo koji tip u bazi podataka da se traži.

Posle definisanja koja veza sa bazom želi da se koristi, koja transakcija, i procedura treba da budu uključene, prosleđuju se parametri potrebni za proceduru. Prosleđivanje parametara se vrši preko bezimenih objekata SqlParameter, koji pored imena parametra u proceduri kojoj se prosleđuje vrednost, zahtevaju da im se prosledi i vrednost koja će da bude isporučena naznačenom parametru. Preporučuje se korišćenje SqlParameter objekta pri prosleđivanju potrebnih vrednosti zbog zaštite od sql inject napada.

Posle dodavanja potrebnih parametara koji se prosleđuju proceduri izvršava se poziv procedure i prosleđivanje parametara shodno izabranoj vezi pomoću metode ExecuteNonQuery(). ExecuteNonQuery metoda se koristi kod unosa novog zapisa kao i kod promene već postojećeg zapisa.

U slučaju neuspeha pokreće se catch blok koji vraća poruku sa opisom greške.

1.1.1.3 Prikaz skupa podataka pomoću select naredbe

U slučajevima koji traže prikaz skupa podataka poziva se procedura koja u sebi sadrži select upit.

Najčešći oblik metode za prikaz skupa podataka iz baze podataka je:

```

public List<klasaModela> listaVrednosti(int parametar)
{
    List<klasaModela> lista = new List<klasaModela>();
    using (SqlConnection konekcija = new SqlConnection(Baza.VEZA))
    {
        try
        {

```

```

        SqlCommand komanda = DBConn.setProc(konekcija, "
ImeProcedureUBaziPodatka ");
        komanda.Parameters.Add(new SqlParameter("@imeParametra11",
objekatKlase. promenljiva11));
        komanda.Parameters.Add(new SqlParameter("@imeParametra12",
objekatKlase. promenljiva12));

        SqlDataReader citac = null;

        konekcija.Open();
        citac = komanda.ExecuteReader();

        while (citac.Read())
        {
            lista.Add(new klasaModela
(Convert.ToInt32(citac["ID"].ToString()), citac["Vrednost1"].ToString(),
Convert.ToDateTime(citac["Vrednost2"].ToString()),
citac["Vresnot3"].ToString()));
        }
        catch (Exception ex)
        {
            lista.Add(new Cenovnik(-89, ex.Message + " " + ex.ToString(), new
DateTime(), " "));
        }
    }
    return lista;
}

```

Svaka metoda koja pruža skup podataka, vraća listu podataka, bez obzira da li su to podaci o detalju izabranog korisničkog naloga ili lista uplata izabranog naloga, u cilju uniformnosti poziva metoda koje pružaju skupove vrednosti i povećane lakoće proširivanja modula koji koriste DatabaseLayer. Kao i za metode koje omogućavaju unos novih zapisa ili promenu postojećih koriste se objekti SqlParameter radi prosleđivanja vrednosti proceduri.

Obavezno je formiranje objekta klase SqlDataReader koji ima ulogu dostavljanja skupa podataka iz baze podataka. Izvršavanje unete procedure izvršava se komandom ExecuteReader(). ExecuteReader ima ulogu da vrati skup podataka iz baze podataka. Iz razloga što se vraća skup podataka objekat tipa klase SqlDataReader je tipa kolekcije.

Kolekcijama može da se pristupi na više načina i pomoću više različitih tipova pelji. Najčešće se koristi while petlja, moguće je i sa foreach petljom ali posle foreach petlje obavezno je kastovanje dobijenih objekata.

U telu while petlje, izalnoj listi se dodaje bezimeni novoformirani objekat tipa navedenog u tipu liste. Treba da se obrati pažnja na konstruktor objekta, naročito na nazive pod navodnicima, jer su to nazivi izlaznih parametara koji se nalaze u povratnom skupu iz baze podataka.

U slučaju neuspeha pokreće se catch blok koji vraća objekat sa promenljivom koja u sebi sadrži opis greške.

1.1.1.4 Zaštita od SQL inject

Predstavlja zaštitu od prosleđivanja zlonamernog koda u bazu podataka. Zlonamerni kod u ovom slučaju odnosi se na kod koji nije u skladu sa procedurom u kojoj se poziva. Obezbeđena je razumna sigurnost korišćenjem objekta SqlParameter kod prosleđivanja, jer

vrednost koja se prosleđuje parametru, mora da bude istog tipa kao parametar i ta vrednost se prosleđuje u polje gde se parametar procedure čuva u tabeli u bazi podataka. Tako, na primer, ukoliko je tabela Fizičko lice, koja ima kolonu Ime, koja je tipa nvarchar, ukoliko se pokuša da se prosledi vrednost za ime (`komanda.Parameters.Add(new SqlParameter("@Ime", objekatKlase.promenljiva11));`), i `promenljiva11` poseduje vrednost ; `DROP TABLE Korisnici`; takva vrednost će da bude sačuvana u polju Ime ukoliko polje to dozvoljava, time biće sprečeno uklanjanje tabele Korisnici.

15.5.2. Model

Modul Model predstavlja skup klasa koje se koriste radi formiranja objekata sa ulogom prenosa podataka između celina informacionog sistema, u samim delovima informacionog sistema, u modulima, kao i između modula.

Svaka klasa poseduje promenljive, javna svojstva, parametarske i besparametarske konstruktore.

1.1.1.5 Promenljive

Klasa, iz razloga čuvanja podataka u objektima formiranim po njenom šablonu, je obavezna da poseduje promenljive. Tip promenljive i ostale pojedinosti, zavise od namene klase a samim tim i objekta.

Opšti tip promenljive je:

```
tipPromenljive nazivPromenljive;
```

Promenljive u informacionom sistemu su ili proste promenljive ili objekti klasa iz modela ili liste takvih objekata.

1.1.1.6 Javna svojstva

Javna svojstva po svom obliku i karakteristikama jako su slična metodima ali su predodređena isključivo za pristup promenljivima. U informacionom sistemu svojstva se koriste isključivo prilikom prosleđivanja parametara promenljivima van klase u kojoj su promenljive definisane ili dobavljanja vrednosti koja se već nalazi u promenljivoj. Ne preporučuje se njihovo korišćenje u klasi u kojoj su definisani jer zauzimaju više resursa nego sam promenljive.

Opšti tip javnog svojstva je:

```
public tipPromenljive NazivSvojstva
{
    get { return nazivPromenljive; }
    set { nazivPromenljive = value; }
}
```

Get deo javnog svojstva dobavlja već potojeću vrednost u promenljivoj.

Set deo javnog svojstva postavlja novu vrednost u promenljivu.

1.1.1.7 Konstruktori

Konstruktori se koriste radi postavljanja vrednosti, odnosno inicijalizovanja promenljivih. U informacionom sistemu koriste se parametarski konstruktori kao i bezparametarski.

Parametarski konstruktori da bi inicijalizovali promenljive traže da im se proslede vrednosti, dok bezparametarski to ne zahtevaju. Bezparametarski konstruktori se koriste u informacionom sistemu zbog serijalizacije i deserijalizacije. Parametarski konstruktori se koriste radi inicijalizacije promenljivih i pravljenja svrsishodnih objekata.

Opšti oblik parametarskog konstruktora:

```
public imeKlase(tipPromenljive1 naziv1, tipPromenljive2 naziv2, ...)
{
    this.promenljiva1 = naziv1;
    this.promenljiva2 = naziv2;
    ...
}
```

15.5.3. Šifrovanje i Dešifrovanje

Modul koji je zadužen za šifrovanje i dešifrovanje, kao i generisanje ključeva. Sastoji se iz tri klase:

- Desifrovanje
- Šifrovanje
- GeneratorReči

Klase zadužene za šifrovanje i dešifrovanje koriste za te potrebe TripleDESCryptoServiceProvider modul koji je napravljene od strane Microsoft firme. Podržava 192 bitne ključeve sa kojima se u informacionom sistemu šifruju odnosno dešifruju potrebni podaci.

Izgled klase Desifrovanje:

```
public class Desifrovanje
{
    public static string DesifrovanjeString(string sifrovanText, bool hashing,
    string kljuc)
    {
        byte[] keyArray;
        byte[] toEncryptArray = Convert.FromBase64String(sifrovanText);

        string key = kljuc;

        if (hashing)
        {
            MD5CryptoServiceProvider hashmd5 = new MD5CryptoServiceProvider();
            keyArray = hashmd5.ComputeHash(UTF8Encoding.UTF8.GetBytes(key));
            hashmd5.Clear();
        }
        else
            keyArray = UTF8Encoding.UTF8.GetBytes(key);

        TripleDESCryptoServiceProvider tdes = new
        TripleDESCryptoServiceProvider();
        tdes.Key = keyArray;
        tdes.Mode = CipherMode.ECB;
        tdes.Padding = PaddingMode.PKCS7;

        ICryptoTransform cTransform = tdes.CreateDecryptor();
        byte[] resultArray = cTransform.TransformFinalBlock(toEncryptArray, 0,
        toEncryptArray.Length);

        tdes.Clear();
        return UTF8Encoding.UTF8.GetString(resultArray);
    }
}
```

```

    }
}

```

Prosleđeni ključ se pretvara u niz bajtova kao i tekst koji se želi da se dešifruje.

Izgled klase Sifrovanje:

```

public class Sifrovanje
{
    public static string SifrovanjeUString(string tekst, bool hashing, string
kljuc)
    {
        byte[] keyArray;
        byte[] toEncryptArray = UTF8Encoding.UTF8.GetBytes(tekst);

        string key = kljuc;

        if (hashing)
        {
            MD5CryptoServiceProvider hashmd5 = new MD5CryptoServiceProvider();
            keyArray = hashmd5.ComputeHash(UTF8Encoding.UTF8.GetBytes(key));
            hashmd5.Clear();
        }
        else
            keyArray = UTF8Encoding.UTF8.GetBytes(key);

        TripleDESCryptoServiceProvider tdes = new
TripleDESCryptoServiceProvider();
        tdes.Key = keyArray;
        tdes.Mode = CipherMode.ECB;
        tdes.Padding = PaddingMode.PKCS7;

        ICryptoTransform cTransform = tdes.CreateEncryptor();
        byte[] resultArray = cTransform.TransformFinalBlock(toEncryptArray, 0,
toEncryptArray.Length);
        tdes.Clear();
        return Convert.ToBase64String(resultArray, 0, resultArray.Length);
    }
}

```

Prosleđeni ključ se pretvara u niz bajtova kao i tekst koji se želi da se šifruje.

Izgled klase GeneratorReci:

```

public static class GeneratorReci
{
    public static string generator(int size)
    {
        String preradjeno = "";
        String izlaz = "";
        while (preradjeno.Length < size)
        {
            Guid g = Guid.NewGuid();
            String prelaz = g.ToString();
            foreach (char c in prelaz.ToCharArray())
            {
                if (!c.Equals('-'))
                {
                    preradjeno += c;
                }
            }
        }
        for (int i = 0; i < size; i++)

```

```

        {
            izlaz += preradjeno[i];
        }
        return izlaz;
    }
}

```

Služi za pravljenje ključeva koji se koriste u metodama klase Sifrovanje i Desifrovanje. Koristi objekat klase GUID radi generisanja novog broja. GUID ili Globally Unique Identifier, po tvrdnjama Microsoft kompanije obezbeđuje veću verovatnoću ka jedinstvenim zapisima. Pored toga u informacionom sistemu posle generisanja ključeva pozivane su metode koje proveravaju unikatnost ključeva, tj. da li se poklapaju sa već generisanim i sačuvanim u bazi ključevima.

15.5.4. Transfer

Modul Transfer koristi se radi obezbeđivanja što lakšeg prenosa podataka od ASP servera ka API serveru i nazad, kao i radi smanjenja broja linija koda, zbog sprečavanja ponavljanja istog bloka koda za slanje kroz informacioni sistem.

Transfer modul se sastoji od jedne klase Slanje koja je statička, kao i metoda slanjePost koji je takođe statički. Samim time smanjuje se zauzimanje resursa jer ne traži se da se formiraju objekti radi samog čina slanja podataka.

Izgled klase Slanje i primadajućeg metoda je:

```

public static class Slanje
{
    public static HttpResponseMessage slanjePOST<T>(T objektatZaSlanje, String
ciljaniKontroler)
    {
        List<T> listaKorisnik = new List<T>();
        listaKorisnik.Add(objektatZaSlanje);
        String slanje = XMLserializer.Serialize.serializovanje(listaKorisnik);
        String preradjeno = "";
        preradjeno = XMLserializer.IspravkaXMLString.IspravkaXML('<', '['',
slanje);
        preradjeno = XMLserializer.IspravkaXMLString.IspravkaXML('>', ']',
preradjeno);
        HttpClient client = new HttpClient();
        HttpContent htc = new StringContent(preradjeno);
        htc.Headers.ContentType = new
System.Net.Http.Headers.MediaTypeHeaderValue("application/text");
        HttpResponseMessage odgovor =
client.PostAsync("https://localhost:8088/api/"+ciljaniKontroler, htc).Result;
        return odgovor;
    }
}

```

Metoda slanjePost je generička metoda. Generička metoda se odnosi na metode kojima se definiše tip parametara prilikom njihovih poziva. U metodi slanjePost parametarZaSlanje može da bude objekat bilo koje klase, jer ukoliko pored oznake slanjePOST<T> ne stoji <T:nekaKlasa> nije ograničen tip objekta samo na klase koje nasleđuju nekaKlasa klasu. Podaci se prenose u telu poruke pomoću post metode, iz više razloga je to poželjno a jedan od njih je i da se je vidi u adresi podaci koji se prenose. Sam sadržaj poruke odnosno objekat koji se prenosi, prvo se serijalizuje pomoću modula XMLserializer, potom se u takvom nizu zamene sporni znakovi < i > sa znakovima [i] koji se posle menjaju u prvobitne znakove na strain API server. Slanje se zahvaljujući prethodno izdatim i instaliranim sertifikatima obavlja preko https veze tj šifrovano pomoću sertifikata.

15.5.5. XMLserializer

XMLserializer je modul koji je zadužen za serializovanje i deserializovanje objekata u XML format, kao i zamene spornih karaktera u stringu.

Sastoji se od tri klase:

- Serialize
- ZamenaZnakova
- IspravkaXMLString

1.1.1.8 Serialize

Serialize klasa služi za serializovanje objekata u XML string pomoću metode Serializovanje i deserializovanje objekata pomoću klase Deserializovanje.

Klasa izgleda:

```
public class Serialize
{
    public static string serializovanje(Object objekat)
    {
        StringWriter s = new StringWriter();
        System.Xml.Serialization.XmlSerializer x = new
System.Xml.Serialization.XmlSerializer(objekat.GetType());

        x.Serialize(s, objekat);
        String rezultat = s.ToString();
        s.Close();
        return rezultat;
    }

    public static T Deserializovanje<T>(String s, T objekat)
    {
        System.Xml.Serialization.XmlSerializer x = new
System.Xml.Serialization.XmlSerializer(objekat.GetType());
        StringReader citac = new StringReader(s);
        T rezultat = (T)x.Deserialize(citac);
        return rezultat;
    }
}
```

U procesu serializacije pronalazi se tip potosleđenog objekta koji se želi da se serializuje. Serializacija se čuva u objektu StringWriter. Zbog potrebe da sadržaj poruke bude XML string, od objekta StringWriter se formira string i string se vraća pozivaocu metode.

U procesu deserializacije pored XML stringa, neophodno je da se prosledi i objekat tipa klase koja je bila šablon za objekat koji je bio serializovan. Od stringa se formira objekat StringReader koji se pomoću klase XmlSerializer deserializuje i kasteruje u objekat tipa klase iz koje je potekao.

1.1.1.9 ZamenaZnakova

Klasa ZamenaZnakova ima ulogu formiranja objekata koji kasnijim korišćenjem u klasi IspravkaXMLString omogućava lakšu zamenu spornih znakova.

Pored promenljivih stariZnak i noviZnak, postoje i pripadajuća njihova javna svojstva StariZnak i NoviZnak kao i parametarski konstruktor koji se koristi za inicijalizaciju promenljivih.

Klasa izgleda:

```
public class ZamenaZnakova
{
    char stariZnak, noviZnak;

    public char StariZnak
    {
        get { return stariZnak; }
        set { stariZnak = value; }
    }

    public char NoviZnak
    {
        get { return noviZnak; }
        set { noviZnak = value; }
    }

    public ZamenaZnakova(char stariZnak1, char noviZnak1)
    {
        this.stariZnak = stariZnak1;
        this.noviZnak = noviZnak1;
    }
}
```

1.1.1.10 IspravkaXMLString

Klasa IspravkaXMLString u sebi poseduje metodu IspravkaXML koja pomoću foreach petlje koja prolazi kroz prosleđeni tekst i if uslova određuje ispunjenost uslova kada treba da se stari znak zameni sa novim znakom. Na kraju metoda vraća ispravljeni tekst pozivaocu metode.

Klasa izgleda:

```
public static class IspravkaXMLString
{
    public static string IspravkaXML(char stariZnak, char noviZnak, String
recenica)
    {
        String preradjeno = "";
        foreach (char c in recenica.ToCharArray())
        {
            if (c == stariZnak)
            {
                preradjeno += noviZnak;
            }
            if (c != stariZnak)
            {
                preradjeno += c;
            }
        }
        return preradjeno;
    }
}
```

16. Instalacija

U segmentu instalacija biće prikazana instalacija sertifikacionog tela, instalacije uloga na serveru baze podataka, isporuka IIS serverima API servera i ASP severa, podnošenje zahteva za sertifikatom, izdavanje sertifikata kao i njegova instalacija.

16.1. Sertifikaciono telo (CA)

Sertifikaciono telo za demonstrativne potrebe informacionog sistema instalirano je na računaru koji pokreće Windows server 2008 R2 Datacenter. Posle instalacije CA na izabranom računaru, dati računar ne može da menja svoje ime niti grupu kojoj pripada.

Zbog ograničenog broja strana ovog rada, koran po korak upustva za instalaciju su prikazana u prilogu instalacija CA.

16.2. Stvaranje uloge na serveru baze podataka

Korišćenje sertifikata sa serverom baze podataka Microsoft SQL server 2008, nalaže stvaranje uloge pomoću koje bi se vršila prijava na server baze podataka.

Zbog ograničenog broja strana ovog rada, koran po korak upustva za instalaciju su prikazana u prilogu stvaranje uloga na serveru baze podataka.

16.3. Instalacija Internet Information Services (IIS)

Radi lakšeg slanja zahteva za sertifikatom, kao i za objavljivanje ASP i API servera, treba da se izvrši instalacija IIS.

Zbog ograničenog broja strana ovog rada, koran po korak upustva za instalaciju su prikazana u prilogu instalacija IIS.

16.4. Instalacija ASP i API servera

Zbog ograničenog broja strana ovog rada, koran po korak upustva za instalaciju su prikazana u prilogu instalacija ASP i API servera.

16.5. Slanje zahteva, izdavanje i uvoz sertifikata

Prikazano je slanje zahteva za server delove informacionog sistema, izdavanje sertifikata od lokalnog CA tela i uvoz sertifikata za server delove informacionog sistema kao i instalacija potrebnih setrifikata na klijent računaru.

Zbog ograničenog broja strana ovog rada, koran po korak upustva za instalaciju su prikazana u prilogu slanje zahteva, izdavanje i uvoz sertifikata.

17. Način rada u informacionom sistemu

Za rad u sistemu postoje 4 tipa uloga koje korisnici mogu da poseduju. U jednom trenutku mogu da poseduju samo jedan tip uloge.

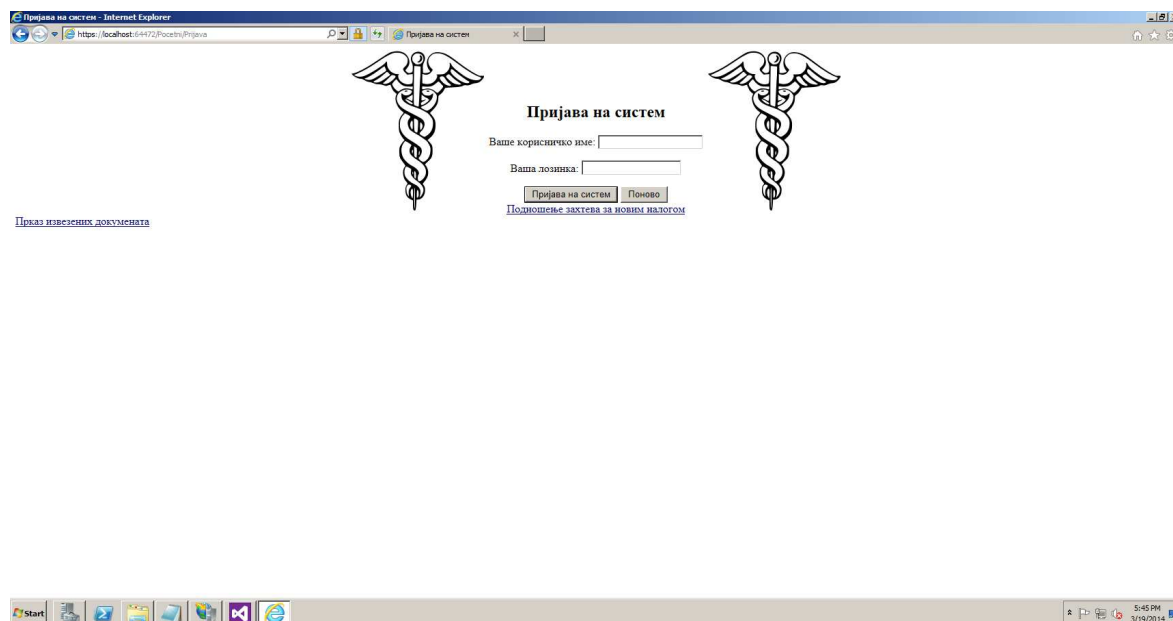
Tipovi uloga su:

- Pacijent,
- Lekar,
- Medicinska sestra,
- Administracija.

Zajedničko za sve stranice koje poseduju mogućnost unosa, odnosno izmene podataka da važi pravilo da su sva polja sa zvezdicom (*) obavezna.

Zajedničko za sve uloge je stranica za prijavu (Slika 745). Na stranici za prijavu postoje polja za unos korisničkog imena i lozinke. U slučaju da korisničko ime, ili lozinka se ne nalaze u tabeli u bazi podataka u istom zapisu korisniku se pokazuje poruka o grešci. Takođe poruke o grešci se prikazuku u slučajevima ukoliko nije unešeno korisničko ime, odnosno lozinka. Pored polja za unos korisničkog ime i lozinke postoje dugme „Пријава на систем“ za podnošenje zahteva za prijavom na sistem, linkovi koji vode ka stranicama za podnošenjem zahteva za otvaranjem korisničkog naloga, odnosno link ka pregledu izveženih dijagnoza. Dugme „Поново“ briše prethodno unete vrednosti .

Zbog ograničenog obima rada, način pravilnog korišćenja informacionog sistema prikazan je u prilogu nacin rada u informacionom sistemu.



Slika 33 Stranica za prijavu na sistem

18. Karateristične putanje naredbi u informacionom sistemu

Radi lakšeg shvatanja načina funkcionisanja informacionog sistema zdravstvene institucije, u daljem tekstu biće prikazane karakteristične putanje naredbi za određene zahteve.

18.1. Podnošenje zahteva za ulogu „Pacijent“

Pritiskom na dugme za podnošenje zahteva poziva se metoda PodnosenjeZahteva [HttpPost]

```
public ActionResult PodnosenjeZahteva(Korisnik kor)
```

```

        {
            if (kor != null && kor.KorIme != null && kor.Ložinka != null && kor.Fl !=
null && kor.Fl.JMBG != null && kor.Fl.Ime != null && kor.Fl.Prezime != null &&
                kor.Fl.Adr != null && kor.Fl.Adr.Grad != null && kor.Fl.Adr.Opstina !=
null && kor.Fl.Adr.Ulica != null && kor.Fl.Adr.Broj != null && kor.Fl.Adr.Sprat !=
null
                && kor.Fl.Adr.StLok != null)
            {
                KorisnikController kc = new KorisnikController();
                if (kc.proveraJMBG(kor) == true)
                {
                    @ViewBag.odgovor = "ЈМБГ број је заузет";
                    return View("PodnosenjeZahteva");
                }
                else if (kc.proveraKorIme(kor) == true)
                {
                    @ViewBag.odgovor = "Лозинка је заузета";
                    return View("PodnosenjeZahteva");
                }
                else if (new DatumKontrola().proveraDatuma(kor.Fl.DatRodj) == true)
                {
                    @ViewBag.odgovor = "Неисправан датум";
                    return View("PodnosenjeZahteva");
                }
                else if (kc.proveraJMBG(kor) == false && kc.proveraKorIme(kor) ==
false)
                {
                    //Sifrovanje
                    Kljuc k = new Kljuc(0,
SifrovanjeDesifrovanje.GeneratorReci.generator(24),
SifrovanjeDesifrovanje.GeneratorReci.generator(24));
                    while (true)
                    {
                        if (new KljuceviController().provera(k.JmbgKljuc) == false)
                        {
                            break;
                        }
                        else
                        {
                            k.JmbgKljuc =
SifrovanjeDesifrovanje.GeneratorReci.generator(24);
                        }
                    }
                    while (true)
                    {
                        if (new KljuceviController().provera(k.LožinkaKljuc) == false)
                        {
                            break;
                        }
                        else
                        {
                            k.LožinkaKljuc =
SifrovanjeDesifrovanje.GeneratorReci.generator(24);
                        }
                    }
                    kor.K = k;
                    kor.Fl.JMBG =
SifrovanjeDesifrovanje.Sifrovanje.SifrovanjeUString(kor.Fl.JMBG, false,
kor.K.JmbgKljuc);

```

```

        kor.Lozinka
SifrovanjeDesifrovanje.Sifrovanje.SifrovanjeUString(kor.Lozinka,
kor.K.LozinkaKljuc);
        =
        false,

        HttpResponseMessage
Transfer.Slanje.slanjePOST<Korisnik>(kor, @"Korisnik/podnosenjeZahteva");
        odgovor
        =
        if (odgovor.IsSuccessStatusCode)
        {
            Task<String> s = odgovor.Content.ReadAsStringAsync();
            String rezultat = s.Result;
            @ViewBag.odgovor = rezultat;
            return View("Odgovor");
        }
        else
        {
            @ViewBag.odgovor = "Није могуће да се изврши повезивање са
сучељем(интерфејсом) за комуникацију са базом података.\nПодаци нису сачувани.";
            return View("Odgovor");
        }
    }
    else
    {
        @ViewBag.odgovor = "Поља са звездицом(*) су обавезна";
        return View("PodnosenjeZahteva");
    }
}
else
{
    @ViewBag.odgovor = "Поља са звездицом(*) су обавезна";
    return View("PodnosenjeZahteva");
}
}
}

```

Pre slanja vrši se provera na strani ASP servera validnosti podataka koji su prosleđeni od klijenta. Postoji uvek verovatnoća da se modifikuje stranica za slanje koja je poslata klijentu od strane klijenta, ovim se to onemogućuje. Ukoliko su validni podaci vrši se provera jedinstvenosti u odnosu na podatke u bazi podataka. Podaci kao JMBG broj ili korisničko ime moraju da budu jedinstveni zapisi.

Jedinstvenost JMBG i korisničkog imena proverava se pomoću metoda:

```

public bool proveraJMBG(Korisnik kor)
{
    bool provera = false;
    foreach (Korisnik k in listaKorisnika1())
    {
        if (k.Fl.JMBG.Trim().Equals(kor.Fl.JMBG.Trim()))
        {
            provera = true;
        }
    }
}

```

```

    }
    return provera;
}

public bool proveraKorIme(Korisnik kor)
{
    bool provera = false;
    foreach (Korisnik k in listaKorisnika1())
    {
        if (k.KorIme.Trim().Equals(kor.KorIme.Trim()))
        {
            provera = true;
        }
    }
    return provera;
}

public List<Korisnik> listaKorisnika1()
{
    Korisnik kor = new Korisnik();
    List<Korisnik> listaOdgovor = new List<Korisnik>();
    HttpResponseMessage odgovor = Transfer.Slanje.slanjePOST<Korisnik>(kor,
@"Korisnik/listaKorisnika1");
    if (odgovor.IsSuccessStatusCode)
    {
        Task<String> s = odgovor.Content.ReadAsStringAsync();
        String rezultat = s.Result;
        String preradjeno = "";
        preradjeno = XMLserializer.IspravkaXMLString.IspravkaXML('[', '<',
rezultat);
        preradjeno = XMLserializer.IspravkaXMLString.IspravkaXML(']', '>',
preradjeno);

        listaOdgovor =
XMLserializer.Serialize.Deserializovanje<List<Korisnik>>(preradjeno, listaOdgovor);
        if (listaOdgovor.Count == 0)
        {
            TempData["odgovor"] = MessageBoxPoruka.Poruka("Не постоји тренутно
расположива листа корисника.");
        }
        else if (listaOdgovor.Count > 0)
        {
            foreach (Korisnik korOdgovor in listaOdgovor)
            {
                korOdgovor.Ložinka =
SifrovanjeDesifrovanje.Desifrovanje.DesifrovanjeString(korOdgovor.Ložinka, false,
korOdgovor.K.LožinkaKljuc);
                korOdgovor.Fl.JMBG =
SifrovanjeDesifrovanje.Desifrovanje.DesifrovanjeString(korOdgovor.Fl.JMBG, false,
korOdgovor.K.JmbgKljuc);
            }
        }
    }
    return listaOdgovor;
}

```

Podaci se dobavljaju od API servera pomoću metode listaKorisnika1. Podaci kao što su korisničko ime i JMBG radi povećanja sigurnosti su šifrovani, samim tim oni moraju prvo da budu dešifrovani pozivanjem metode DesifrovanjeString i prosleđivanjem ključeva koji su dobijeni od API servera.

Posle dešifrovanja podaci u obliku objekta Korisnik se prosleđuju metodama proveraJMBG i proveraKorIme. Ukoliko su jedinstveni zapisi metode će vratiti false vrednost. Ukoliko nisu vratiće vrednost true.

Metoda koja prihvata zahtev na API serveru za listu korisnika izgleda:

```
[HttpPost]
public HttpResponseMessage listaKorisnika1()
{
    HttpContent httpObjekat = Request.Content;
    Task<String> s = httpObjekat.ReadAsStringAsync();
    String objekat = s.Result;
    String preradjeno = "";
    preradjeno = objekat;
    preradjeno = XMLserializer.IspravkaXMLString.IspravkaXML('[', '<',
objekat);
    preradjeno = XMLserializer.IspravkaXMLString.IspravkaXML(']', '>',
preradjeno);

    List<Korisnik> zahtevLista = new List<Korisnik>();
    zahtevLista = XMLserializer.Serialize.Deserializovanje<List<Korisnik>>(preradjeno, zahtevLista);
    zahtevLista.Clear();
    zahtevLista = new KorisnikDAL().listaKorisnika1SaKljucovima();
    String slanje = XMLserializer.Serialize.serializovanje(zahtevLista);
    preradjeno = "";
    preradjeno = XMLserializer.IspravkaXMLString.IspravkaXML('<', '[',
slanje);
    preradjeno = XMLserializer.IspravkaXMLString.IspravkaXML('>', ']',
preradjeno);

    HttpResponseMessage hrm = new HttpResponseMessage();
    hrm.Content = new StringContent(preradjeno);
    return hrm;
}
```

Kada se zahtev prihvati stvara se lista koja će da prihvati odgovor od metoda iz DatabaseLayer modula po imenu listaKorisnikaSaKljucovima.

Metoda listaKorisnikaSaKljucovima izgleda:

```
public List<Korisnik> listaKorisnika1SaKljucovima()
{
    List<Korisnik> listaK = listaKorisnika1();
    foreach (Korisnik kor in listaK)
    {
        List<Kljuc> listaKljuc = listaKljucevaPoIDKor(kor.Id);
        kor.K = listaKljuc[0];
    }
    return listaK;
}
```

Metoda se sastoji od pozive dve metode, od koji jedna pribavlja listu korisnika a druga listu ključeva.

Metoda listaKorisnika1 u DataBaseLayer modulu uzgleda:

```
public List<Korisnik> listaKorisnika1()
{
    List<Korisnik> lista = new List<Korisnik>();
    using (SqlConnection konekcija = new SqlConnection(Baza.VEZA))
    {
        try
        {
```

```

        SqlCommand komanda = DBConn.setProc(konekcija, "ListaKorisnika1");
        SqlDataReader citac = null;

        konekcija.Open();
        citac = komanda.ExecuteReader();

        while (citac.Read())
        {
            Adresa          adr          =          new
            Adresa(Convert.ToInt32(citac["IDAdresa"].ToString()),          citac["Grad"].ToString(),
            citac["Opstina"].ToString(),          citac["Ulica"].ToString(),          citac["Broj"].ToString(),
            Convert.ToInt32(citac["Sprat"].ToString()), citac["StLok"].ToString());
            FizickoLice      fl          =          new
            FizickoLice(Convert.ToInt32(citac["IDFizLice"].ToString()), citac["JMBG"].ToString(),
            citac["Ime"].ToString(),          citac["Prezime"].ToString(),
            Convert.ToDateTime(citac["Datum"].ToString()), adr);
            lista.Add(new
            Korisnik(Convert.ToInt32(citac["IDKorisnik"].ToString()),          citac["KorIme"].ToString(),
            citac["Lozinka"].ToString(),          citac["Uloga"].ToString(),          citac["Status"].ToString(),
            fl));
        }
    }
    catch (Exception ex)
    {
        lista.Add(new Korisnik(89, ex.Message + " " + ex.ToString(), "",
        "", ""));
    }
}
return lista;
}

```

Metoda poziva proceduru u bazi podataka ListaKorisnika1 koja vraća kao odgovor skup korisnika.

Procedura ListaKorisnika1 u bazi podataka izgleda:

```

USE [ISZdravstvo]
GO
/***** Object:      StoredProcedure [dbo].[ListaKorisnika1]      Script Date:
03/20/2014 13:22:08 *****/
SET ANSI_NULLS ON
GO

```

```

SET QUOTED_IDENTIFIER ON
GO
ALTER procedure [dbo].[ListaKorisnika1]
as
begin
    select [Fizicko lice].*, [Korisnik.*], [Pristupni podaci].*, Adresa.* ,
    [Datum rodjenja].*,
    [Fizicko lice].ID IDFizLice, [Korisnik.ID IDKorisnik, [Pristupni
    podaci].ID IDPristupniPodaci,
    [Datum rodjenja].ID IDDatumRodjenja , Adresa.ID IDAdresa
    from [Fizicko lice], [Korisnik], [Pristupni podaci], [Datum
    rodjenja], Adresa, [Fizicko lice Korisnik],
    [Korisnik Adresa], [Korisnik Pristupni podaci], [Fizicko lice Datum
    rodjenja]
    where [Fizicko lice Korisnik].IDFL=[Fizicko lice].ID
    and [Fizicko lice Korisnik].IDKor=[Korisnik.ID
    and [Korisnik Adresa].IDKor=[Korisnik.ID
    and [Korisnik Adresa].IDAdresa=Adresa.ID
    and [Korisnik Pristupni podaci].IDKor=[Korisnik.ID
    and [Korisnik Pristupni podaci].IDPrisPod=[Pristupni podaci].ID
    and [Fizicko lice Datum rodjenja].IDFL=[Fizicko lice].ID
    and [Fizicko lice Datum rodjenja].IDDatRodj=[Datum rodjenja].ID
end

```

Metoda listaKljucevaPoIDKor u DatabaseLayer modelu izgleda:

```

public List<Kljuc> listaKljucevaPoIDKor(int idKor)
{
    List<Kljuc> lista = new List<Kljuc>();
    using (SqlConnection konekcija = new SqlConnection(Baza.VEZASIFRA))
    {
        try
        {
            SqlCommand komanda = DBConn.setProc(konekcija,
            "ListaKljucevaPoIDKorisnika");
            komanda.Parameters.Add(new SqlParameter("@IDKor", idKor));
            SqlDataReader citac = null;

            konekcija.Open();
            citac = komanda.ExecuteReader();

            while (citac.Read())
            {
                lista.Add(new
                Kljuc(Convert.ToInt32(citac["IDKor"].ToString()), citac["JMBGKljuc"].ToString(),
                citac["LozinkaKljuc"].ToString()));
            }
        }
        catch (Exception ex)
        {
            lista.Add(new Kljuc(-89, ex.Message + " " + ex.ToString(), ""));
        }
    }
    return lista; }

```

Metoda poziva proceduru u bazi podataka ISZdravstvoSifra sa nazivom ListaKljucevaPoIDKorisnika, radi dobavljanja ključeva za šifrovana polja shodno korisničkom broju, koja izgleda:

```

USE [ISZdravstvo_Sifra]
GO
/***** Object:      StoredProcedure      [dbo].[ListaKljucevaPoIDKorisnika]
Script Date: 03/20/2014 13:26:46 *****/

```

```

SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER procedure [dbo].[ListaKljucevaPoIDKorisnika]
(
    @IDKor int
)
as
begin
    select * from Sifra where Sifra.IDKor=@IDKor;
end

```

Posle provere jedinstvenosti zapisa JMBG broja kao i korisničkog imena, jmbg vrednost kao i korisničko ime se šifruju jedinstvenim ključevima. Jrdinstvenost ključeva se obezbeđuje pomoću poziva metoda provera koji pomoću foreach petlje proverava da li je prosleđeni ključ jedinstven u odnosu na pojedinačne vrednosti iz skupa već sačuvanih ključeva. Ukoliko je jedinstven metod vraća vrednost falsem u suprotnom vraća vrednost false.

```

public bool provera(string broj)
{
    bool prov = false;
    foreach (Kljuc s in ListaSvihKljuceva())
    {
        if (s.JmbgKljuc.Trim().Equals(broj))
        {
            prov = true;
        }
    }
    return prov;
}

```

Skup ključeva metoda provera dobija pozivom metode ListaSvihKljuceva.

```

public List<Kljuc> ListaSvihKljuceva()
{
    Kljuc k = new Kljuc();
    List<Kljuc> listaOdgovor = new List<Kljuc>();
    HttpResponseMessage odgovor = Transfer.Slanje.slanjePOST<Kljuc>(k,
@"Kljucevi/ListaSvihKljuceva");
    if (odgovor.IsSuccessStatusCode)
    {
        Task<String> s = odgovor.Content.ReadAsStringAsync();
        String rezultat = s.Result;
        String preradjeno = "";
        preradjeno = XMLserializer.IspravkaXMLString.IspravkaXML('[', '<',
rezultat);
        preradjeno = XMLserializer.IspravkaXMLString.IspravkaXML(']', '>',
preradjeno);

        listaOdgovor
XMLserializer.Serialize.Deserializovanje<List<Kljuc>>(preradjeno, listaOdgovor);
    }
    return listaOdgovor;
}

```

Metoda ListaSvihKljuceva skup ključeva dobija pozivajući metodu na API serveru ListaSvihKljuceva koji izgleda:

```

[HttpPost]
public HttpResponseMessage ListaSvihKljuceva()
{
    HttpContent httpObjekat = Request.Content;
    Task<String> s = httpObjekat.ReadAsStringAsync();
}

```

```

        String objekat = s.Result;
        String preradjeno = "";
        preradjeno = objekat;
        preradjeno = XMLserializer.IspravkaXMLString.IspravkaXML('[', '<',
objekat);
        preradjeno = XMLserializer.IspravkaXMLString.IspravkaXML(']', '>',
preradjeno);

        List<Kljuc> zahtevLista = new List<Kljuc>();
        zahtevLista =
XMLserializer.Serialize.Deserializovanje<List<Kljuc>>(preradjeno, zahtevLista);
        zahtevLista.Clear();
        zahtevLista = new KljuceviDAL().listaSvihKljuceva();
        String slanje = XMLserializer.Serialize.serializovanje(zahtevLista);
        preradjeno = "";
        preradjeno = XMLserializer.IspravkaXMLString.IspravkaXML('<', '[',
slanje);
        preradjeno = XMLserializer.IspravkaXMLString.IspravkaXML('>', ']',
preradjeno);

        HttpResponseMessage hrm = new HttpResponseMessage();
        hrm.Content = new StringContent(preradjeno);
        return hrm;
    }

```

Metoda ListaSvihKljuciva sa API servera poziva metodu listaSvihKljuceva iz DatabaseLayer modula koja izgleda:

```

public List<Kljuc> listaSvihKljuceva()
{
    List<Kljuc> lista = new List<Kljuc>();
    using (SqlConnection konekcija = new SqlConnection(Baza.VEZA))
    {
        try

```

```

        {
            SqlCommand komanda = DBConn.setProc(konekcija, "ListaKljuceva");
            SqlDataReader citac = null;

            konekcija.Open();
            citac = komanda.ExecuteReader();

            while (citac.Read())
            {
                Kljuc k1 = new Kljuc() { JmbgKljuc = citac["Broj"].ToString()
};
                Kljuc k2 = new Kljuc() { JmbgKljuc =
citac["izvozKljuc"].ToString() };
                Kljuc k3 = new Kljuc() { JmbgKljuc =
citac["MaticniSifra"].ToString() };
                Kljuc k4 = new Kljuc() { JmbgKljuc =
citac["APRSifra"].ToString() };
                Kljuc k5 = new Kljuc() { JmbgKljuc =
citac["PIBSifra"].ToString() };
                Kljuc k6 = new Kljuc() { JmbgKljuc =
citac["RegBrSifra"].ToString() };
                Kljuc k7 = new Kljuc() { JmbgKljuc =
citac["JMBGKljuc"].ToString() };
                Kljuc k8 = new Kljuc() { JmbgKljuc =
citac["LozinkaKljuc"].ToString() };

                lista.Add(k1);
                lista.Add(k2);
                lista.Add(k3);
                lista.Add(k4);
                lista.Add(k5);
                lista.Add(k6);
                lista.Add(k7);
                lista.Add(k8);
            }
        }
        catch (Exception ex)
        {
            lista.Add(new Kljuc() { JmbgKljuc = ex.Message + " " +
ex.ToString() });
        }
    }
    return lista;
}

```

Metoda listaSvihKljuceva u DatabaseLayer modulu poziva proceduru ListaKljuceva iz baze podataka ISZdravstvoSifra koja izgleda:

```
USE [ISZdravstvo_Sifra]
```

```
GO
```

```
/***** Object:      StoredProcedure [dbo].[ListaKljuceva]
```

```
Script Date:
```

```
03/20/2014 13:40:05 *****/
```

```
SET ANSI_NULLS ON
```

```
GO
```

```
SET QUOTED_IDENTIFIER ON
```

```
GO
```

```

ALTER procedure [dbo].[ListaKljuceva]
as
begin
    select
        izvozKljuc, PLSifra.MaticniSifra, PLSifra.APRSifra,
        PLSifra.PIBSifra, PLSifra.RegBrSifra,
        Sifra.JMBGKljuc, Sifra.LozinkaKljuc from Izvoz, PLSifra, Sifra
End

```

Posle provere jedinstvenosti generisanih ključeva, vrši se šifrovanje JMBG vrednosti kao i vrednosti korisničkog imena. Kada se završi šifrovanje šifrovane vrednosti se dodeljuju odgovarajućim promenljivima kao i ključevi. Vrši se poziv podnosenjeZahteva metode na API server strani koji izgleda:

```

[HttpPost]
public HttpResponseMessage podnosenjeZahteva()
{
    HttpContent httpObjekat = Request.Content;
    Task<String> s = httpObjekat.ReadAsStringAsync();
    String objekat = s.Result;
    String preradjeno = "";
    preradjeno = objekat;
    preradjeno = XMLserializer.IspravkaXMLString.IspravkaXML('[', '<',
objekat);
    preradjeno = XMLserializer.IspravkaXMLString.IspravkaXML(']', '>',
preradjeno);

    List<Korisnik> zahtevLista = new List<Korisnik>();
    zahtevLista
XMLserializer.Serialize.Deserializovanje<List<Korisnik>>(preradjeno, zahtevLista);
    string odgovor = " ";
    odgovor= new KorisnikDAL().podnosenjeZahteva(zahtevLista[0]);
    HttpResponseMessage hrm = new HttpResponseMessage();
    hrm.Content = new StringContent(odgovor);
    return hrm;
}

```

Metoda podnosenjeZahteva vrši zamenu spornih znakova. Posle zamene znakova vrši se deserializacija i prosleđivanje objekata metodi podnosenjeZahteva u DatabaseLayer modulu koja izgleda:

```

public String podnosenjeZahteva(Korisnik kor)
{
    String odgovor = "";
    int idKor = -1;
    using (TransactionScope ts = new TransactionScope())
    {
        using (SqlConnection konekcija = new SqlConnection(Baza.VEZA))
        {
            SqlCommand komanda = new SqlCommand();

```

```

        try
        {
            konekcija.Open();

            komanda.Connection = konekcija;
            komanda.CommandText = "PodnosenjeZahtevaFL";
            komanda.CommandType = CommandType.StoredProcedure;

            komanda.Parameters.Add(new SqlParameter("@KorIme",
kor.KorIme));
            komanda.Parameters.Add(new SqlParameter("@Lozinka",
kor.Lozinka));
            komanda.Parameters.Add(new SqlParameter("@JMBG",
kor.Fl.JMBG));
            komanda.Parameters.Add(new SqlParameter("@Ime", kor.Fl.Ime));
            komanda.Parameters.Add(new SqlParameter("@Prezime",
kor.Fl.Prezime));
            komanda.Parameters.Add(new SqlParameter("@Datum",
kor.Fl.DatRodj.ToShortDateString()));
            komanda.Parameters.Add(new SqlParameter("@Grad",
kor.Fl.Adr.Grad));
            komanda.Parameters.Add(new SqlParameter("@Opstina",
kor.Fl.Adr.Opstina));
            komanda.Parameters.Add(new SqlParameter("@Ulica",
kor.Fl.Adr.Ulica));
            komanda.Parameters.Add(new SqlParameter("@Broj",
kor.Fl.Adr.Broj));
            komanda.Parameters.Add(new SqlParameter("@Sprat",
kor.Fl.Adr.Sprat));
            komanda.Parameters.Add(new SqlParameter("@StLok",
kor.Fl.Adr.StLok));

            SqlParameter Sid = new SqlParameter("@IDKor", SqlDbType.Int);
            Sid.Direction = ParameterDirection.Output;
            komanda.Parameters.Add(Sid);
            komanda.ExecuteNonQuery();

            idKor = Convert.ToInt32(Sid.Value);

            if (idKor == -1)
            {
                ts.Dispose();
            }

        }
        catch (Exception ex)
        {
            ts.Dispose();
            return ex.Message;
        }

    }
    if (idKor != -1)
    {
        using (SqlConnection konekcija = new
SqlConnection(Baza.VEZASIFRA))

```

```

        {
            SqlCommand komanda = new SqlCommand();

            try
            {
                konekcija.Open();

                komanda.Connection = konekcija;
                komanda.CommandText = "NoviKorisnikIKljucevi";
                komanda.CommandType = CommandType.StoredProcedure;

                komanda.Parameters.Add(new SqlParameter("@IDKor", idKor));
                komanda.Parameters.Add(new SqlParameter("@JMBGKljuc",
kor.K.JmbgKljuc));
                komanda.Parameters.Add(new SqlParameter("@LozinkaKljuc",
kor.K.LozinkaKljuc));

                komanda.ExecuteNonQuery();

                ts.Complete();
                odgovor = "Захтев је успешно сачуван у бази података";
            }
            catch (Exception ex)
            {
                ts.Dispose();
                return ex.Message;
            }
        }
    }
    return odgovor;
}

```

Metoda podnosenjeZahteva iz DatabaseLayer modula koristi TransactionScope objekat koji omogućava izvršavanje transakcija između više baza podataka, sa osiguranjem da ako u jednoj bazi podataka ne uspe naredba, TransactionScope prekida dalje izvršavanje i sve što je do tada izvršeno se poništava. Postoji velika sličnost sa Transaction objektom, samo što TransactionScope obavlja sličnu ulogu nad više baza podataka. Prvo se pokušava upis podataka o zahtevu. Ukoliko se nije javio izuzetak, nastavlja se sa upisom pripadajućih ključeva.

Posle poslatog zahteva metod pozivaoc iz ASP servera prosledjuje odgovor pogledu Odgovor.

18.2. Prijava na sistem

Prijava na sistem posle unosa u pogledu potrebnih podataka počinje pozivom metoda Prijava koji kao parametar prihvata objekat tipa klase Korisnik iz Model modula. Metod prijava izgleda:

```

[HttpPost]
public ActionResult Prijava(Korisnik kor)
{

```

```

        if (kor.KorIme == null || kor.LoZinka == null)
        {
            TempData["odgovor"] = MessageBoxPoruka.Poruka("Унесите потребне
податке");
            return View();
        }
        else if (kor.KorIme.Length > 0 && kor.LoZinka.Length > 0)
        {
            HttpResponseMessage odgovor =
Transfer.Slanje.slanjePOST<Korisnik>(kor, @"Korisnik/listaKorisnikaPoLoginPodacima");

            if (odgovor.IsSuccessStatusCode)
            {
                Task<String> s = odgovor.Content.ReadAsStringAsync();
                String rezultat = s.Result;
                String preradjeno = "";
                preradjeno = XMLserializer.IspravkaXMLString.IspravkaXML('[', '<',
rezultat);
                preradjeno = XMLserializer.IspravkaXMLString.IspravkaXML(']', '>',
preradjeno);
                List<Korisnik> listaOdgovor = new List<Korisnik>();
                listaOdgovor =
XMLserializer.Serialize.Deserializovanje<List<Korisnik>>(preradjeno, listaOdgovor);
                if (listaOdgovor.Count == 0)
                {
                    TempData["odgovor"] = MessageBoxPoruka.Poruka("Унели сте
нетачне податке");
                    return View();
                }
                else if (listaOdgovor.Count > 0)
                {
                    listaOdgovor[0].LoZinka =
SifrovanjeDesifrovanje.Desifrovanje.DesifrovanjeString(listaOdgovor[0].LoZinka, false,
listaOdgovor[0].K.LoZinkaKljuc);
                    listaOdgovor[0].Fl.JMBG =
SifrovanjeDesifrovanje.Desifrovanje.DesifrovanjeString(listaOdgovor[0].Fl.JMBG, false,
listaOdgovor[0].K.JmbgKljuc);
                    TempData["listaUstanova"] = new
UstanovaController().listaSvihUstanova();
                    TempData["listaSvihKorisnikaSemPrijavljenog"] = new
KorisnikController().listaKorisnikaBezPrijavljenog(listaOdgovor[0]);
                    TempData["ListaZaposlenih"] = new
ZaposleniController().listaZaposlenihBezPrijavljenog();
                    TempData["ListaSvihMagacina"] = new
MagacinController().listaMagacina();
                    TempData["ListaSvijJedinicaMera"] = new
JedinicaMereController().listaJedinicaMera();
                    TempData["ListaSifrarnikDijagnoze"] = new
SifrarnikDijagnozeController().listaSvihSifrarnika();
                    TempData["ListaNacinaDavanjaInjekcija"] = new
NacinDavanjaInjekcijeController().listaNacina();
                    TempData["ListaTipUputa"] = new
TipUputaController().listaSvihTipovaUputa();
                    TempData["ListaCenovnika"] = new
CenovnikController().listaSvihCenovnika();
                    if (listaOdgovor[0].Uloga.Equals("Медицинска сестра") ||
listaOdgovor[0].Uloga.Equals("Лекар"))
                    {
                        List<Korisnik> listaSvih = new
KorisnikController().listaKorisnikaBezPrijavljenog(listaOdgovor[0]);
                        List<Korisnik> listaPac = new List<Korisnik>();
                        foreach (Korisnik k in listaSvih)

```

```

        {
            if (k.Uloga.Equals("Пацијент"))
            {
                listaPac.Add(k);
            }
        }
        TempData["ListaPacijenata"] = listaPac;
    }
    if (listaOdgovor[0].Uloga.Equals("Пацијент"))
    {
        Pacijent p = new
KorisnikController().listaIDPacZaIDKor(listaOdgovor[0].Id)[0];
        TempData["ListaPregleda"] = new
PregledController().listaPregledaKurativaZaIDPac(p.Id);
    }
    return View("Maticna", listaOdgovor[0]);
}
else
{
    return View();
}
}
else
{
    ViewBag.odgovor = MessageBoxPoruka.Poruka("Није могуће да се
изврши повезивање са сучељем(интерфејсом) за комуникацију са базом података.");
    return View("Odgovor");
}
}
else
{
    return View("Maticna", kor);
}
}

```

Metod prvo proverava da li su vrednosti unete za korisničko ime i lozinku, ukoliko jesu unete poziva se metod iz API servera radi dobijanja korisnika koji poseduje uneto korisničko ime i lozinku. Proverava se zatim njegova uloga u shodno ulozi dobavljaju se vrednosti privremenim poljima. API server metod koji je pozvan nosi naziv listaKorisnikaPoLoginPodacima.

Metod listaKorisnikaPoLoginPodacima izgleda:

```

[HttpPost]
public HttpResponseMessage listaKorisnikaPoLoginPodacima()
{
    HttpContent httpObjekat = Request.Content;
    Task<String> s = httpObjekat.ReadAsStringAsync();
    String objekat = s.Result;
    String preradjeno = "";
}

```

```

        preradjeno = objekat;
        preradjeno = XMLserializer.IspravkaXMLString.IspravkaXML('[', '<',
objekat);
        preradjeno = XMLserializer.IspravkaXMLString.IspravkaXML(']', '>',
preradjeno);

        List<Korisnik> zahtevLista = new List<Korisnik>();
        zahtevLista =
XMLserializer.Serialize.Deserializovanje<List<Korisnik>>(preradjeno, zahtevLista);
        string korIme = zahtevLista[0].KorIme;
        string lozinka = zahtevLista[0].Lozinka;
        zahtevLista.Clear();
        zahtevLista = new
KorisnikDAL().listaKorisnikaPoSifriILOzinki(korIme, lozinka);
        String slanje = XMLserializer.Serialize.serializovanje(zahtevLista);
        preradjeno = "";
        preradjeno = XMLserializer.IspravkaXMLString.IspravkaXML('<', '[',
slanje);
        preradjeno = XMLserializer.IspravkaXMLString.IspravkaXML('>', ']',
preradjeno);

        HttpResponseMessage hrm = new HttpResponseMessage();
        hrm.Content = new StringContent(preradjeno);
        return hrm;
    }

```

U metodu na API server strani komunikacije vrši se dobavljanje iz objekta vrednosti za korisničko ime i lozinku, kao i njihovo prosleđivanje metodi DatabaseLayer modula listaKorisnikaPoSifriILOzinki koja izgleda:

```

public List<Korisnik> listaKorisnikaPoSifriILOzinki(string korIme, string lozinka)
{
    List<Korisnik> kor1 = listaKorisnikaPoKorImenu(korIme);
    List<Kljuc> listaKljuc = listaKljucevaPoIDKor(kor1[0].Id);
    String lozinkaProvera =
SifrovanjeDesifrovanje.Desifrovanje.DesifrovanjeString(kor1[0].Lozinka, false,
listaKljuc[0].LozinkaKljuc);

    if (lozinka ==lozinkaProvera)
    {
        kor1[0].K = listaKljuc[0];
    }

    return kor1;
}

```

Metoda poziva druge metode koje dostavljaju listu korisnika, kao i listu pripadajućih ključeva.

Metoda za dobavljanje liste korisnika izgleda:

```

public List<Korisnik> listaKorisnikaPoKorImenu(string korIme)
{
    List<Korisnik> lista = new List<Korisnik>();
    using (SqlConnection konekcija = new SqlConnection(Baza.VEZA))
    {
        try

```

```

        {
            SqlCommand komanda = DBConn.setProc(konekcija,
"ListaKorisnikaPoKorImenu");
            komanda.Parameters.Add(new SqlParameter("@KorIme", korIme));
            SqlDataReader citac = null;

            konekcija.Open();
            citac = komanda.ExecuteReader();

            while (citac.Read())
            {
                Adresa adr = new
                Adresa(Convert.ToInt32(citac["IDAdresa"].ToString()), citac["Grad"].ToString(),
                citac["Opstina"].ToString(), citac["Ulica"].ToString(), citac["Broj"].ToString(),
                Convert.ToInt32(citac["Sprat"].ToString()), citac["StLok"].ToString());
                FizickoLice fl = new
                FizickoLice(Convert.ToInt32(citac["IDFizLice"].ToString()), citac["JMBG"].ToString(),
                citac["Ime"].ToString(), citac["Prezime"].ToString(),
                Convert.ToDateTime(citac["Datum"].ToString()), adr);
                lista.Add(new
                Korisnik(Convert.ToInt32(citac["IDKorisnik"].ToString()), citac["KorIme"].ToString(),
                citac["Lozinka"].ToString(), citac["Uloga"].ToString(), citac["Status"].ToString(),
                fl));
            }
        }
        catch (Exception ex)
        {
            lista.Add(new Korisnik(89, ex.Message + " " + ex.ToString(), "",
            "", ""));
        }
    }
    return lista;
}

```

Svoje podatke metoda dobija pozivom sql procedure iz baze podataka ISZdravstvo koja izgleda:

```

USE [ISZdravstvo]
GO
/***** Object:  StoredProcedure [dbo].[ListaKorisnikaPoKorImenu]      Script
Date: 03/20/2014 14:11:00 *****/
SET ANSI_NULLS ON
GO

```

```

SET QUOTED_IDENTIFIER ON
GO
ALTER procedure [dbo].[ListaKorisnikaPoKorImenu]
(
    @KorIme nvarchar(50)
)
as
begin
    select [Fizicko lice].*, Korisnik.*, [Pristupni podaci].*, Adresa.* ,
    [Datum rodjenja].*,
    [Fizicko lice].ID IDFizLice, Korisnik.ID IDKorisnik, [Pristupni
podaci].ID IDPristupniPodaci,
    [Datum rodjenja].ID IDDatumRodjenja , Adresa.ID IDAdresa
    from [Fizicko lice], Korisnik, [Pristupni podaci], [Datum
rodjenja], Adresa, [Fizicko lice Korisnik],
    [Korisnik Adresa], [Korisnik Pristupni podaci], [Fizicko lice Datum
rodjenja]
    where [Fizicko lice Korisnik].IDFL=[Fizicko lice].ID
    and [Fizicko lice Korisnik].IDKor=Korisnik.ID
    and [Korisnik Adresa].IDKor=Korisnik.ID
    and [Korisnik Adresa].IDAdresa=Adresa.ID
    and [Korisnik Pristupni podaci].IDKor=Korisnik.ID
    and [Korisnik Pristupni podaci].IDPrisPod=[Pristupni podaci].ID
    and [Fizicko lice Datum rodjenja].IDFL=[Fizicko lice].ID
    and [Fizicko lice Datum rodjenja].IDDatRodj=[Datum rodjenja].ID
    and Korisnik.Status='АКТИВНО'
    and [Pristupni podaci].KorIme=@KorIme
end

```

```

Metoda za dobavljanje ključeva izgleda
public List<Kljuc> listaKljucevaPoIDKor(int idKor)
{
    List<Kljuc> lista = new List<Kljuc>();
    using (SqlConnection konekcija = new SqlConnection(Baza.VEZASIFRA))
    {
        try
        {

```

```

        SqlCommand komanda = DBConn.setProc(konekcija,
"ListaKljucevaPoIDKorisnika");
        komanda.Parameters.Add(new SqlParameter("@IDKor", idKor));
        SqlDataReader citac = null;

        konekcija.Open();
        citac = komanda.ExecuteReader();

        while (citac.Read())
        {
            lista.Add(new
Kljuc(Convert.ToInt32(citac["IDKor"].ToString()), citac["JMBGKljuc"].ToString(),
            citac["LozinkaKljuc"].ToString()));
        }
    }
    catch (Exception ex)
    {
        lista.Add(new Kljuc(-89, ex.Message + " " + ex.ToString(), ""));
    }
}
return lista;
}

```

Svoje podatke metoda dobija pozivom sql procedure iz baze podataka ISZdravstvoSifra koja izgleda:

```

USE [ISZdravstvo_Sifra]
GO
/***** Object:  StoredProcedure [dbo].[IzmenaKljucevaZaIDKor]      Script
Date: 03/20/2014 14:12:04 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER procedure [dbo].[IzmenaKljucevaZaIDKor]
(
    @IDKor int,
    @JMBGKljuc nvarchar(4000),
    @LozinkaKljuc nvarchar(4000)
)
as
begin
    update Sifra set JMBGKljuc=@JMBGKljuc, LozinkaKljuc=@LozinkaKljuc
where Sifra.IDKor=@IDKor;
end

```

Ukoliko je dobijen uspešni odgovor od strane API servera vrši se zamena spornih znakova u XML stringu i njegova deserijalizacija. Podaci za JMBG i korisničko ime su šifrovani i pre prikazivanja korisniku moraju da se dešifruju. Uništava se vrednost lozinke radi sprečavanje njenog prosleđivanja pozivaozu, odnosno podnosiocu zahteva za prijavu na sistem. Ukoliko je dešifrovanje obavljeno, prikazuje se početna stranica shodno ulozi koja je korisniku dodeljena.

18.3. Unos dijagnoze

Unos dijagnoze posle unosa potrebnih podataka nastavlja se pozivom metoda DijagnozaCuvanje koji izgleda:

```

public ActionResult DijagnozaCuvanje(Dijagnoza dijag, int idAnam, int idSifra)
{

```

```

        if (dijag.Nalaz != null && dijag.Napomena != null && idSifra != null)
        {
            dijag.Nalaz.IdAnamneze = idAnam;
            dijag.SifDijag = new SifrarnikDijagnoze() { Id = idSifra };
            HttpResponseMessage odgovor =
Transfer.Slanje.SlanjePOST<Dijagnoza>(dijag, @"Dijagnoza/NovaDijagnoza");
            if (odgovor.IsSuccessStatusCode)
            {
                Task<String> s = odgovor.Content.ReadAsStringAsync();
                String rezultat = s.Result;
                @ViewBag.odgovor = rezultat;
            }
            return View("Odgovor");
        }
        else
        {
            @ViewBag.odgovor = "Поља са звездицом(*) су обавезна";
            return PregledLekar(idAnam);
        }
    }

```

Pre pozivanja kontrolera na API serveru, vrši se provera unetosti potrebnih podataka. Ukoliko su uneti potrebni podaci, vrši se poziv metoda NovaDijagnoza u kontroleru koji izgleda:

```

[HttpPost]
public HttpResponseMessage NovaDijagnoza()
{
    HttpContent httpObjekat = Request.Content;
    Task<String> s = httpObjekat.ReadAsStringAsync();
    String objekat = s.Result;
    String preradjeno = "";
    preradjeno = objekat;
    preradjeno = XMLserializer.IspravkaXMLString.IspravkaXML('[', '<',
objekat);
    preradjeno = XMLserializer.IspravkaXMLString.IspravkaXML(']', '>',
preradjeno);

    List<Dijagnoza> zahtevLista = new List<Dijagnoza>();
    zahtevLista =
XMLserializer.Serialize.Deserializovanje<List<Dijagnoza>>(preradjeno, zahtevLista);
    string odgovor = " ";
    odgovor = new DijagnozaDAL().novaDijagnoza(zahtevLista[0]);

    HttpResponseMessage hrm = new HttpResponseMessage();
    hrm.Content = new StringContent(odgovor);
    return hrm;
}

```

Metod vrši zamenu spornih znakova u XML stringu, deserijalizaciju XML stringa i prosleđivanje objekta novaDijagnoza metodu u DatabaseLayer modulu koji izgleda:

```

public string novaDijagnoza(Dijagnoza dijag)
{
    string odgovor = "";
    using (SqlConnection konekcija = new SqlConnection(Baza.VEZA))
    {
        SqlCommand komanda = new SqlCommand();
    }
}

```

```

try
{
    konekcija.Open();
    SqlTransaction trasaction = konekcija.BeginTransaction();

    komanda.Connection = konekcija;
    komanda.CommandText = "NovaDijagnoza";
    komanda.CommandType = CommandType.StoredProcedure;
    komanda.Transaction = trasaction;

    komanda.Parameters.Add(new SqlParameter("@Vrednost",
dijag.Nalaz.Vrednost));

    komanda.Parameters.Add(new SqlParameter("@Napomena",
dijag.Napomena));

    komanda.Parameters.Add(new SqlParameter("@IDSifDijag",
dijag.SifDijag.Id));
    komanda.Parameters.Add(new SqlParameter("@IDAnam",
dijag.Nalaz.IdAnamneze));

    komanda.ExecuteNonQuery();

    trasaction.Commit();
    odgovor = "Подаци су сачувани.";
}
catch (Exception ex)
{
    odgovor = "Настала је грешка: " + ex.Message;
}
}
return odgovor;
}

```

Metoda novaDijagnoza iz DatabaseLayer modula preuzima objekat, dodaje parametre preko SqlParameter vrednostima iz preuzetog objekta. Vršiti se pozivanje procedure NovaDijagnoza koja izgleda:

```

USE [ISZdravstvo]
GO
/***** Object:      StoredProcedure [dbo].[NovaDijagnoza]      Script Date:
03/20/2014 14:28:39 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

```

```

ALTER procedure [dbo].[NovaDijagnoza]
(
--Nalaz
@Vrednost nvarchar(50),

--Dijagnoza
@Napomena nvarchar(150),

@IDSifDijag int,
@IDAnam int
)
as
declare @IDNalaz int
declare @IDDijag int
begin
    begin
        insert into [Nalaz pregleda](Vrednost, Status)
values(@Vrednost, 'АКТИВНО');
        set @IDNalaz=@@IDENTITY;
    end
    begin
        insert into
Dijagnoza(Napomena, Status)values(@Napomena, 'АКТИВНО');
        set @IDDijag=@@IDENTITY
    end
    begin
        insert into [Nalaz pregleda
Anamneza](IDAnam, IDNalaz)values(@IDAnam, @IDNalaz);
    end
    begin
        insert into [Dijagnoza nalaz](IDDijag, IDNalaz)
values(@IDDijag, @IDNalaz);
    end
    begin
        insert into [Dijagnoza Sifrarnik dijagnoza](IDDijag, IDSifDijag)
values (@IDDijag, @IDSifDijag);
    end
end

```

Ukoliko je uspешan unos korisnik o tome biva obavešten pomoću prozora Odgovor.

18.4. Izmena dijagnoze

Posle izmene potrebnih podataka vrši se poziv metoda u kontroleru izmenaDijagnoze koji izgleda:

```

public ActionResult IzmenaCuvanje(Dijagnoza dijag, int idDijag, int idNalaz, int idSifra)
{
    if (dijag.Nalaz != null && dijag.Napomena != null && idDijag != null && idNalaz != null && idSifra != null)
    {
        dijag.Id = idDijag;
    }
}

```

```

        dijak.Nalaz.Id = idNalaz;
        dijak.SifDijag = new SifrarnikDijagnoze() { Id = idSifra };
        HttpResponseMessage odgovor
Transfer.Slanje.SlanjePOST<Dijagnoza>(dijag, @"Dijagnoza/IzmenaDijagnoze");
        if (odgovor.IsSuccessStatusCode)
        {
            Task<String> s = odgovor.Content.ReadAsStringAsync();
            String rezultat = s.Result;
            @ViewBag.odgovor = rezultat;
        }
        return Detaljnije(idDijag);
    }
    else
    {
        @ViewBag.odgovor = "Поља са звездицом(*) су обавезна";
        return Izmena(idDijag);
    }
}

```

Ukoliko su podaci uneti vrši se poziv IzmenaDijagnoze metoda u kontroleru sa strane API servera koji izgleda:

```

[HttpPost]
public HttpResponseMessage IzmenaDijagnoze()
{
    HttpContent httpObjekat = Request.Content;
    Task<String> s = httpObjekat.ReadAsStringAsync();
    String objekat = s.Result;
    String preradjeno = "";
    preradjeno = objekat;
    preradjeno = XMLserializer.IspravkaXMLString.IspravkaXML('[', '<',
objekat);
    preradjeno = XMLserializer.IspravkaXMLString.IspravkaXML(']', '>',
preradjeno);

    List<Dijagnoza> zahtevLista = new List<Dijagnoza>();
    zahtevLista
XMLserializer.Serialize.Deserializovanje<List<Dijagnoza>>(preradjeno, zahtevLista);
    string odgovor = " ";
    odgovor = new DijagnozaDAL().izmenaDijagnoze(zahtevLista[0]);

    HttpResponseMessage hrm = new HttpResponseMessage();
    hrm.Content = new StringContent(odgovor);
    return hrm;
}

```

Vrši se zamena spornih karaktera u XML stringu, njegova deserijalizacija i prosleđivanje novonastalog objekta izmenaDijagnoze metodu u DatabaseLayer modulu.

izmenaDijagnoze metod u DatabaseLayer modulu izgleda:

```

public string izmenaDijagnoze(Dijagnoza dijak)
{
    string odgovor = "";
    using (SqlConnection konekcija = new SqlConnection(Baza.VEZA))
    {
        SqlCommand komanda = new SqlCommand();

        try
        {
            konekcija.Open();

```

```

        SqlTransaction trasaction = konekcija.BeginTransaction();

        komanda.Connection = konekcija;
        komanda.CommandText = "IzmenaDijagnoza";
        komanda.CommandType = CommandType.StoredProcedure;
        komanda.Transaction = trasaction;

        komanda.Parameters.Add(new SqlParameter("@IDNalaz",
dijag.Nalaz.Id));
        komanda.Parameters.Add(new SqlParameter("@Vrednost",
dijag.Nalaz.Vrednost));

        komanda.Parameters.Add(new SqlParameter("@IDDijag", dijag.Id));
        komanda.Parameters.Add(new SqlParameter("@Napomena",
dijag.Napomena));

        komanda.Parameters.Add(new SqlParameter("@IDSifDijag",
dijag.SifDijag.Id));

        komanda.ExecuteNonQuery();

        trasaction.Commit();
        odgovor = "Подаци су сачувани.";
    }
    catch (Exception ex)
    {
        odgovor = "Настала је грешка: " + ex.Message;
    }
}
return odgovor;
}

```

Metoda izmenaDijagnoze iz DatabaseLayer modula preuzima objekat, dodaje parametre preko SqlParameter vrednostima iz preuzetog objekta. Vrš se pozivanje procedure IzmenaDijagnoza koja izgleda:

```

USE [ISZdravstvo]
GO
/***** Object:  StoredProcedure [dbo].[IzmenaDijagnoza]      Script Date:
03/20/2014 14:39:01 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

```

```

ALTER procedure [dbo].[IzmenaDijagnoza]
(
--Nalaz
@IDNalaz int,
@Vrednost nvarchar(50),

--Dijagnoza
@IDDijag int,
@Napomena nvarchar(150),

@IDSifDijag int
)
as
begin
    begin
        update [Nalaz pregleda] set Vrednost=@Vrednost
        where [Nalaz pregleda].ID=@IDNalaz;
    end
    begin
        update      Dijagnoza      set      Napomena=@Napomena      where
Dijagnoza.ID=@IDDijag;
    end
    begin
        update      [Dijagnoza      Sifrarnik      dijagnoza]      set
IDSifDijag=@IDSifDijag
        where [Dijagnoza Sifrarnik dijagnoza].IDDijag=@IDDijag;
    end
end

```

18.5. Prikaz liste dijagnoza

Prikazi listi se obično u informacionom sistemu obavljaju preko metoda koje vraćaju liste objekata. Liste objekata se potom dodeljuju privremenim prenosiocima kao što je TempData radi prikazivanja u pogledima ili se one sam koriste u drugim metodima. Dobavljanje liste dijagnoza se obavlja pozivom metoda čija je jedina razlika po kom kriterijumu se dijagnoze dobavljaju. Za potrebe demonstracije, biće prikazan ListeDijagnozaPoIDPacijenta, odnosno dijagnoze koje su donešene za izabranog pacijenta. Metoda ListeDijagnozaZaIDPacijenta izgleda:

```

public List<Dijagnoza> ListaDijagnozaPoIDPacijenta(int idPacijenta)
{
    Dijagnoza d = new Dijagnoza();
    d.Id = idPacijenta;
    List<Dijagnoza> listaOdgovor = new List<Dijagnoza>();
    HttpResponseMessage odgovor = Transfer.Slanje.slanjePOST<Dijagnoza>(d,
@"Dijagnoza/ListaDijagnozaPoIDPacijenta");
    if (odgovor.IsSuccessStatusCode)
    {
        Task<String> s = odgovor.Content.ReadAsStringAsync();
        String rezultat = s.Result;
        String preradjeno = "";
        preradjeno = XMLserializer.IspravkaXMLString.IspravkaXML('[', '<',
rezultat);
        preradjeno = XMLserializer.IspravkaXMLString.IspravkaXML(']', '>',
preradjeno);

        listaOdgovor =
XMLserializer.Serialize.Deserializovanje<List<Dijagnoza>>(preradjeno, listaOdgovor);
        if (listaOdgovor.Count == 0)
        {
            TempData["odgovor"] = MessageBoxPoruka.Poruka("Не постоје подаци о
магацинима.");
        }
        else if (listaOdgovor.Count > 0)
        {
        }
    }
    return listaOdgovor;
}

```

Posle formiranja objekta koji u sebi nosi broj pacijenta, vrši se njegovo slanje ka metodu u kontroleru koji izgleda:

```

[HttpPost]
public HttpResponseMessage ListaDijagnozaPoIDPacijenta()
{
    HttpContent httpObjekat = Request.Content;
    Task<String> s = httpObjekat.ReadAsStringAsync();
    String objekat = s.Result;
    String preradjeno = "";
    preradjeno = objekat;
}

```

```

objekat);
preradjeno = XMLserializer.IspravkaXMLString.IspravkaXML('[', '<',
preradjeno);
preradjeno = XMLserializer.IspravkaXMLString.IspravkaXML(']', '>',
preradjeno);

List<Dijagnoza> zahtevLista = new List<Dijagnoza>();
zahtevLista =
XMLserializer.Serialize.Deserializovanje<List<Dijagnoza>>(preradjeno, zahtevLista);
int idDijagnoza = zahtevLista[0].Id;
zahtevLista.Clear();
zahtevLista = new DijagnozaDAL().listaDijagnozaPoIDPacijenta(idDijagnoza);
String slanje = XMLserializer.Serialize.serializovanje(zahtevLista);
preradjeno = "";
preradjeno = XMLserializer.IspravkaXMLString.IspravkaXML('<', '[' ,
slanje);
preradjeno = XMLserializer.IspravkaXMLString.IspravkaXML('>', ']',
preradjeno);

HttpResponseMessage hrm = new HttpResponseMessage();
hrm.Content = new StringContent(preradjeno);
return hrm;
}

```

Po prijemu zahteva metod ListaDijagnozaPoIDPacijenta na strani API servera, vrši zamenu spornih znakova u XML stringu, njegovu deserializaciju i prosleđivanje parametra koji nosi vrednost broja pacijenta lokalnoj promenljivoj. Lokalna promenljiva koja poseduje broj pacijenta se prosleđuje metodi listaDijagnozaPoIDPacijenta iz DatabaseLayer modula koja izgleda:

```

public List<Dijagnoza> listaDijagnozaPoIDPacijenta(int idPacijenta)
{
    List<Dijagnoza> lista = new List<Dijagnoza>();
    using (SqlConnection konekcija = new SqlConnection(Baza.VEZA))
    {

```

```

        try
        {
            SqlCommand komanda = DBConn.setProc(konekcija,
"ListaDijagnozaPoIDPacijenta");
            komanda.Parameters.Add(new SqlParameter("@IDPacijent",
idPacijent));
            SqlDataReader citac = null;

            konekcija.Open();
            citac = komanda.ExecuteReader();

            while (citac.Read())
            {
                NalazPregleda np = new
NalazPregleda(Convert.ToInt32(citac["IDNalaz"].ToString()),
citac["Vrednost"].ToString(),
citac["StatusNalaz"].ToString());
                lista.Add(new
Dijagnoza(Convert.ToInt32(citac["IDDijagnoza"].ToString()), new SifrarnikDijagnoze() {
Id = Convert.ToInt32(citac["IDSifDijag"].ToString()) }, np,
citac["Napomena"].ToString(), citac["StatusDijagnoza"].ToString()));
            }
        }
        catch (Exception ex)
        {
            lista.Add(new Dijagnoza(-89, new SifrarnikDijagnoze(), new
NalazPregleda(), ex.Message + " " + ex.ToString(), " "));
        }
    }
    return lista;
}

```

Metoda listaDijagnozaPoIDPacijenta iz DatabaseLayer modula preuzima broj pacijenta, dodaje parametre preko SqlParameter vrednostima iz preuzetog objekta. Vršiti se pozivanje procedure ListaDijagnozaPoIDPacijenta koja izgleda:

```

USE [ISZdravstvo]
GO
/***** Object:      StoredProcedure      [dbo].[ListaDijagnozaPoIDPacijenta]
Script Date: 03/20/2014 14:53:54 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER procedure [dbo].[ListaDijagnozaPoIDPacijenta]

```

```

(
@IDPacijent int
)
as
begin
    select  [Nalaz pregleda].ID IDNalaz, [Nalaz pregleda].Status
StatusNalaz,
[Nalaz pregleda].* , Dijagnoza.ID IDDijagnoza, Dijagnoza.*,
Dijagnoza.Status StatusDijagnoza, [Dijagnoza Sifrarnik
dijagnoza].IDSifDijag
from [Nalaz pregleda],Dijagnoza,[Dijagnoza Sifrarnik dijagnoza],
[Dijagnoza nalaz],
[Nalaz pregleda Anamneza],Anamenza,Kurativa,Pregled
where [Nalaz pregleda].ID=[Dijagnoza nalaz].IDNalPreg
and [Dijagnoza nalaz].IDDijag=Dijagnoza.ID
and Dijagnoza.ID=[Dijagnoza Sifrarnik dijagnoza].IDDijag
and [Nalaz pregleda].ID=[Nalaz pregleda Anamneza].IDNalaz
and [Nalaz pregleda Anamneza].IDAnam=Anamenza.ID
and Anamenza.IDKur=Kurativa.ID
and Kurativa.IDPregleda=Pregled.ID
and Pregled.IDPac=@IDPacijent
end

```

Ukoliko postoje dijagnoze one će biti dodate listi i biti prikazane u odgovarajućem prozoru ili obrađene u drugim metodama.

19. Zaključak

U ovom radu prikazano je projektovanje i implementacija informacionog sistema organizacije koja se bavi pružanjem i naplatom zdravstvenih usluga. U radu je prikazana metodologija po kojoj je istraživanje obavljeno u kojoj je sadržan predmet istraživanja, ciljevi, zadaci istraživanja, istraživačke hipoteze, metode istraživanja kao i tok istraživačkog procesa. Takođe su objašnjene tehnologije koje se koriste pri implementaciji rešenja, prikazani su načini instalacije potrebnih komponenti radi pravilnog funkcionisanja sistema, prikazani su pravilni načini korišćenja sistema, kao i karakteristični delovi koda.

U poglavlju sistemsko strukturna analiza vrši se razlaganje korisničkih zahteva na tokove podataka i procese koji se dalje razlažu na korake koji su potrebni za njihovo izvršavanje.

Prikazani su koraci kroz vremensku osu i po rasporedu, sve sa ciljem što bolje optimizacije i kompetentosti sistema za sigurnim, potpunim i brzim funkcionisanjem sistema.

Omogućeno je da sistem menja svoje delove, tj. module, bez velikog uticaja na sam sistem i bez potrebe gašenja sistema dok se njegova prilagođavanja izvršavaju.

Mogućnosti prilagođavanja i proširivanja sistema odnose se pored procesa u samom sistemu i na mogućnosti koje se korisnicima stavljaju na raspolaganje.

Pojmovi bezbednosti, pouzdanosti i brzine informacionog sistema su uvek podložni promenama. To je uzeto u obzir pri projektovanju sistema, i ostavljena je mogućnost njihovog prilagođavanja potrebama organizacije.

Prilikom ugradnje informacionog sistema, sistem je u mogućnosti da prihvata i obrađuje zahteve sa udaljenih računara koji nisu u sklopu organizacije, korišćenjem https veze i digitalnih sertifikata.

Programerski deo sistema za obradu zahteva od korisnika bazira se na ASP . NET MVC 4.0 tehnologiji. na serverskom delu komunikacije i .ASP .NET API na drugom delu komunikacionog kanala u komunikaciji sa bazom podataka. Uloga ASP .NET API dela sastoji se u dobavljanju i upisivanju podataka u bazu podataka na zahtev ASP .NET MVC servera.

Programski jezik koji se koristi je C#.

Trebalo bi da se nastoji da svi serveri, za programe i za baze podataka, se ne iznajmljuju već čuvaju unutar organizacije. Razlog je povećana kontrola izaštita poslovnih podataka zadržavstvene organizacije, i samim tim smanjenje rizika od pravnih tužbi zbog nedozvoljenog pristupa zdravstvenim podacima.

Sastav računara koji bi imali ulogu ASP servera, preporučljivo je da imaju višeprocesorske mogućnosti, sa mrežnom karticom veće propusne moći.

Sastav računara koji bi imali ulogu servera baza podataka, preporučljivo je da imaju višeprocesorske mogućnosti, sa brzim odzivima medijuma na kojima se skladište podaci.

Preporučljivo je da na računarima koji imaju ulogu servera bude instaliran najmanje Windows 7 ili Windows server 2008, ili njihove novije verzije. Razlog insistiranja na Windows operativnim sistemima je u ASP i SQL serverima i njihovoj međusobnom, relativno mirnom, saživotu sa pomenutim operativnim sistemima.

Serveri za programe i serveri za baze podataka poželjno je da budu povezani sa gigabitnim kablovima, zbog izdržavanja potencijalnog povećanja zahteva između servera.

Serveri za programe i serveri za baze podataka prostorno trebalo bi da su udaljeni, odnosno da nisu u istoj zgradi. Shodno udaljenosti, adekvatni kablovi trebalo bi da se nabave.

Preporučuje se da se da prioritet komunikaciji sa serverima koja potiče iz organizacije u odnosu na komunikaciju koja potiče iz drzghih izvora. Razlog su prava koja imaju korisnici koji pristupaju izvan organizacije su po važnosti manja od potencijalnih hitnih zahteva za podacima u njoj.

Odredjena programska zaštita treba da postoji, zaštitni zidovi, programi protiv zlonamernih programa (anti visrus, anti spam, anti malware), kao i fizičko-tehnička zaštita, na lokacijama gde se nalaze serveri.

Literatura

- [1] Veinović, M., Jevremović, A., Šimić, G., **Baze podataka**, prvo izdanje, Univerzitet Singidunum, Beograd, 2013.
- [2] Marsic, I., **Software engineering**, Rutgers, 2012.
- [3] Tomašević, V., **Razvoj aplikativnog softvera**, Univerzitet Singidunum, 2012.
- [4] Sommerville, I., **Software engineering**, deveto izdanje, Addison-Wesley, 2011.
- [5] Pressman, R., **Software engineering a practitioner's approach**, sedmo izdanje, Mc Graw Hill, 2010.
- [6] Riordan, R., **Projektovanje baza podataka**, Mikroknjiga Beograd, 2006.
- [7] Grupa autora, **Softversko inženjerstvo teorija i praksa prevod trećeg izdanja**, Računarski fakultet, 2006.
- [8] Bell, D., **Software engineering for students a programming approach**, četvrto izdanje, Addison-Wesley, 2005.
- [9] Grupa autora, **Uml za projektovanje baza podataka**, CET Beograd, 2002.
- [10] Todman, C., **Projektovanje skladišta podataka**, CET Beograd, 2001.
- [11] Freeman, A., **Pro ASP .NET MVC 4**, četvrto izdanje, Apress, 2012.
- [12] Liberty, J., **Programiranje na jeziku C#**, prevod četvrtog izdanja, Mikroknjiga, Beograd, 2007.
- [13] Lee, M., Bieker, G., **SQL Server 2008**, prevod prvog izdanja, Kompjuter biblioteka, Čačak, 2008.
- [14] Ronaldo, J., **Beginning ASP. NET MVC 4**, prvo izdanje, Apress, 2012.
- [15] Microsoft virtual academy, Developing ASP.NET MVC 4 Web Applications Jump Start, <http://www.microsoftvirtualacademy.com/training-courses/developing-asp-net-mvc-4-web-applications-jump-start>, 18.01.2014
- [16] The official forum for Microsoft ASP .NET, Getting started with web API (C#), <http://www.asp.net/web-api/overview/getting-started-with-aspnet-web-api/tutorial-your-first-web-api>, 20.01.2014.
- [17] Technet, Create a login sql server 2008, <http://technet.microsoft.com/en-us/library/aa337562.aspx>, 4.02.2014.
- [18] Codeproject, A Beginner's Tutorial for Understanding Transactions and TransactionScope in ADO.NET, <http://www.codeproject.com/Articles/522039/A-Beginners-Tutorial-for-Understanding-Transaction>, 10.02.2014.
- [19] Stackoverflow, Html.BeginForm and adding properties, <http://stackoverflow.com/questions/216600/html-beginform-and-adding-properties>, 15.02.2014.
- [20] The official forum for Microsoft ASP .NET, MVC, <http://forums.asp.net/1146.aspx>, 27.02.2014.
- [21] Stackoverflow, JQuery populate drop down list in mvc, <http://stackoverflow.com/questions/17110202/jquery-populate-drop-down-list-in-mvc>, 5.03.2014

Spisak slika

Slika 1 Dijagram konteksta.....	11
Slika 2 Dijagram slučajeva korišćenja za ulogu "Klijent"	14
Slika 3 Dijagram slučajeva korišćenja za ulogu "Lekar"	17
Slika 4 Dijagram slučajeva korišćenja za ulogu "Medicinska sestra"	20
Slika 5 Dijagram slučajeva korišćenja za ulogu "Administracija"	23
Slika 6 Dijagram slučajeva korišćenja za ulogu "Dobavljač"	27
Slika 7. Dijagram sekvenci uloga „Klijent“, Podnošenje zahteva za otvaranjem naloga	30
Slika 8. Dijagram sekvenci uloga „Lekar“, Prijava na sistem.....	31
Slika 9. Dijagram sekvenci uloga „Medicinska sestra“, Prijava na sistem	32
Slika 10. Dijagram sekvenci uloga „Administracija“, Prijava na sistem	33
Slika 11 Dijagram sekvenci uloga „Dobavljač“, Prijava na sistem.....	34
Slika 12. Dijagram kolaboracije uloga „Klijent“, Podnošenje zahteva za otvaranjem naloga.....	36
Slika 13. Dijagram kolaboracije uloga „Lekar“, Prijava na sistem	36
Slika 14. Dijagram kolaboracije uloga „Medicinska sestra“, Prijava na sistem.....	36
Slika 15. Dijagram kolaboracije uloga „Administracija“, Prijava na sistem.....	37
Slika 16 Dijagram kolaboracije uloga „Dobavljač“, Prijava na sistem.....	37
Slika 17. Dijagram aktivnosti uloga „Klijent“, Podnošenje zahteva za otvaranjem naloga.....	39
Slika 18. Dijagram aktivnosti uloga „Lekar“, Prijava na sistem	40
Slika 19. Dijagram aktivnosti uloga „Medicinska sestra“, Prijava na sistem.....	41
Slika 20. Dijagram aktivnosti uloga „Administracija“, Prijava na sistem.....	42
Slika 21 Dijagram aktivnosti uloga „Dobavljač“, Prijava na sistem.....	43
Slika 22. Dijagram stanja uloga „Klijent“, Podnošenje zahteva za otvaranjem naloga	45
Slika 23. Dijagram stanja uloga „Lekar“, Prijava na sistem	46
Slika 24. Dijagram stanja uloga „Medicinska sestra“, Prijava na sistem	47
Slika 25. Dijagram stanja uloga „Administracija“, Prijava na sistem	48
Slika 26 Dijagram aktivnosti uloga „Dobavljač“, Prijava na sistem.....	49
Slika 27. Dijagram osnovnih klasa informacionog sistema	50
Slika 28. Dijagram rasporeda informacionog sistema.....	52
Slika 29. Dijagram model objekti i veze baze podataka za čuvanje radnih podataka.....	54
Slika 30. Dijagram model objekti i veze baze podataka za čuvanje bezbednosnih podataka ..	55
Slika 31. Prikaz sastava relacionog modela radne baze podataka - prva normalna forma.....	69
Slika 32. Prikaz sastava relacionog modela baze podataka za bezbednost – prva normalna forma.....	69
Slika 33 Stranica za prijavu na sistem.....	90

Spisak tabela

Tabela 1. Korisnik	70
--------------------------	----