Fakultet za Poslovnu Informatiku, Univerzitet Singidunum, Beograd

Aleksandar Jevremović

Informacioni sistem univerziteta

- Magistarski rad -

Beograd 2007

Univerzitet Singidunum, Fakultet za Poslovnu Informatiku, Beograd

Informacioni sistem univerziteta

Mentor
prof. dr Mladen Veinović

Kandidat
Aleksandar Jevremović, dipl. inž.

Beograd 2007

Sažetak

U ovom radu je opisan proces izrade informacionog sistema univerziteta po modelu poslovnog sistema univerziteta Singidunum. Rad se bavi analizom poslovnog sistema, analizom tehnologija koje omogućavaju realizaciju zahteva, razradom modela informacionog sistema, opisom rešenja realizovanog na osnovu modela i pravcima u kojima se očekuje dalji razvoj rešenja. U delove rada na koje je postavljen akcenat spadaju analiza trenutno aktuelnih softverskih tehnologija (softverskih mrežnih arhitektura i web baziranih tehnologija) i opis korišćenja postignutog rešenja. Kao izrazita orijentacija rešenja (i njegovog budućeg razvoja) se navodi SOA arhitektura čije su osobine i protokoli u radu detaljno obrađeni.

Abstract

This paper describes the development process of the University's Information System based on the major business case. In this document I wrote about analysis of the business information system, it's model development, solution description based on business system model and all possible directions of future development. Major keystones of this paper are focused on analysis of contemporary software technologies (software network architectures and web based technologies) and description of current system achievement. SOA architecture, which is also well described and analyzed in this paper, is presented as preferred orientation.

S A D R Ž A J

1. Uvod....12
2. Analiza poslovnog sistema....13
2.1. Analiza zahteva....14
2.2. Analiza kategorija korisnika IS-a....15
2.3. Analiza poslovnih procesa....17
3. Analiza softverskih arhitektura, tehnologija i gotovih rešenja....20
3.1. Analiza softverskih arhitektura....21
3.1.1. Host-based arhitektura....21
3.1.2. Klijent-Server arhitektura....22
3.1.2.1. Iterativna i konkurentna obrada zahteva....23
3.1.3. Peer-to-Peer arhitektura....25
3.1.4. SOA arhitektura....27
3.1.4.1. Proceduralni i objektno-orijentisani pristup....27
3.1.4.2. Objektno-orijentisani i servisno-orijentisani pristup....31
3.1.4.3. Zaključno o SOA....32
3.2. Analiza tehnologija kompatibilnih sa zahtevima....34
3.2.1. Tehnologije vezane za izgradnju baze podataka....35
3.2.1.1. Sistemi za upravljanje bazama podataka (SUBP)....36
3.2.1.1.1. Oracle....38
3.2.1.1.2. IBM DB2....39
3.2.1.1.3. Microsoft SQL server....40
3.2.1.1.4. MySQL....41
3.2.1.2. Alati za modelovanje baza podataka....43
3.2.2. Tehnologije vezane za izgradnju logičkih jedinica softvera....44
3.2.2.1. Microsoft .Net....45
3.2.2.2. Java....45
3.2.2.3. PHP....45
3.2.3. Tehnologije vezane za izgradnju korisničkog interfejsa....46
3.2.3.1. (X)HTML....47
3.2.3.2. CSS....48
3.2.3.3. JavaScript....49
3.2.3.4. Smarty template engine....50

3.2.4. Tehnologije vezane za izgradnju interfejsa ka drugim IS....51
3.2.4.1. Tehnologije bazirane na XML-u....52
3.2.4.1.1. XML-RPC....53
3.2.4.1.2. SOAP....54
3.2.4.1.3. WDDX....55
3.2.4.2. Ostale tehnologije....56
3.2.4.2.1. JSON (JavaScript Object Notation)....56
3.2.4.2.2. Hessian (web service protocol)....57
3.3. Analiza gotovih rešenja i komponenti....58
3.3.1. Analiza postojećih IS za univerzitete....58
3.3.2. Analiza postojećih univerzalnih IS....59
3.3.3. Analiza gotovih komponenti....59
3.3.3.1. EMS....60
3.3.3.2. EFW.PHP....62
3.3.3.3. EFW.JS....62
4. Model IS-a....63
4.1. Kategorije korisničkih modula....63
4.1.1. Menadžment....64
4.1.2. Studentske službe....64
4.1.3. Profesori....64
4.1.4. Studenti....64
4.1.5. Računovodstvo....64
4.1.6. Administratori IS-a....65
4.2. Model arhitekture IS-a....66
4.3. Model baze podataka....68
4.3.1. Tabele baze podataka....69
4.4. Vremensko/funkcionalni model realizacije i implementacije....78
5. Prikaz realizovanih funkcionalnosti IS-a....80
5.1. Menadžment....81
5.1.1. Fakulteti....82
5.1.1.1. Dodavanje....82
5.1.2. Programi (Smerovi)....83
5.1.2.1. Dodavanje....83
5.1.2.2. Izmena....84

5.1.2.3. Pregled....86
5.1.3. Predmeti....88
5.1.3.1. Dodavanje....88
5.1.3.2. Pregled....89
5.1.4. Instruktori....90
5.1.4.1. Dodavanje....90
5.1.4.2. Pregled....91
5.1.5. Tipovi studija....92
5.1.5.1. Dodavanje....92
5.1.6. Lokacije....93
5.1.6.1. Dodavanje....93
5.1.6.2. Pregled....94
5.2. Studentske službe....95
5.2.1. Studenti....96
5.2.1.1. Dodavanje....96
5.2.1.2. Uklanjanje....97
5.2.1.3. Pregled....98
5.2.2. Ispitni rokovi....99
5.2.2.1. Dodavanje....99
5.2.2.2. Izmena....100
5.2.2.3. Uklanjanje....104
5.2.2.4. Pregled....105
5.2.2.5. Zapisnik....106
5.2.3. Prijave ispita....108
5.2.3.1. Dodavanje....108
5.2.3.2. Overa....112
5.2.3.3. Pregled....114
5.3. Studenti....116
5.3.1. Sesija....116
5.3.1.1. Prijava / Odjava....117
5.3.2. Prijava ispita....118
5.3.2.1. Nova prijava....118
5.3.2.2. Pregled prijava....120
6. Dalji pravci razvoja....122

6.1. Podizanje bezbednosti/pouzdanosti sistema na svim nivoima....123
6.1.1. Razvoj internog podsistema privilegija korisnika unutar kategorija....124
6.1.2. Kontrola pristupa korišćenjem SmartCard tehnologije....124
6.1.3. Kreiranje redundantnih vitalnih komponenti sistema....124
6.2. Realizacija preostalih korisničkih modula....126
6.3. Iterativno uvođenje novih funkcionalnosti....126
6.4. Prevođenje na potpunu SOA arhitekturu....127
6.5. Povezivanje sa drugim sistemima....128
6.5.1. Sistem za automatizovano testiranje studenata....129
6.5.2. Sistem za Distance Learning....130
7. Zaključak....131

Sažetak

U ovom radu je opisan proces izrade informacionog sistema univerziteta po modelu poslovnog sistema univerziteta Singidunum. Rad se bavi analizom poslovnog sistema, analizom tehnologija koje omogućavaju realizaciju zahteva, razradom modela informacionog sistema, opisom rešenja realizovanog na osnovu modela i pravcima u kojima se očekuje dalji razvoj rešenja. U delove rada na koje je postavljen akcenat spadaju analiza trenutno aktuelnih softverskih tehnologija (softverskih mrežnih arhitektura i web baziranih tehnologija) i opis korišćenja postignutog rešenja. Kao izrazita orijentacija rešenja (i njegovog budućeg razvoja) se navodi SOA arhitektura čije su osobine i protokoli u radu detaljno obrađeni.

Abstract

This paper describes the development process of the University's Information System based on the major business case. In this document I wrote about analysis of the business information system, it's model development, solution description based on business system model and all possible directions of future development. Major keystones of this paper are focused on analysis of contemporary software technologies (software network architectures and web based technologies) and description of current system achievement. SOA architecture, which is also well described and analyzed in this paper, is presented as preferred orientation.

Spisak slika

1. Spiralni model etapa razvoja.
2. Kategorije korisnika
3. Slojevita klijent-server arhitektura
4. Iterativna obrada zahteva
5. Konkurentna obrada zahteva
6. Šema decentralizovane P2P mreže
7. Šema centralizovane P2P mreže
8. Šema hibridne P2P mreže
9. DB Designer 4 alat za modelovanje baze podataka
10. Tehnologije orijentisane ka korisničkom interfejsu Web aplikacija
11. Korišćenje JavaScript u okviru Web aplikacija
12. Arhitektura sistema sa Smarty template engine-om
13. Odnos spoljnih IS-a sa centralnim
14. Oblast i način upotrebe XML-RPC tehnologije
15. Arhitektura korišćenja ECMS sistema
16. Model modularnog informacionog sistema
17. Klijent-server arhitektura informacionog sistema
18. Mrežna realizacija klijent-server arhitekture
19. SOA arhitektura informacionog sistema
20. Mrežna realizacija informacionog sistema baziranog na SOA arhitekturi
21. Apstraktan model baze podataka
22. Formular za dodavanje novog fakulteta u IS.
23. Formular za dodavanje novog programa (smera)
24. Odabir studijskog programa za izmenu
25. Primer izmene studijskog programa u modulu menadžmenta
26. Lista svih studijskih programa
27. Primer pregleda studijskog programa
28. Primer dodavanja novog predmeta
29. Pregled postojećih predmeta u informacionom sistemu
30. Obrazac za dodavanje instruktora
31. Lista instruktora po fakultetima

32. Formular za dodavanje novog tipa studija
33. Dodavanje nove lokacije u modulu menadžmenta
34. Lista lokacija sa opcionim detaljnim prikazom
35. Formular za dodavanje (upis) novih studenata
36. Poruka pri odabiru opcije uklanjanja studenata.
37. Formular za postavljanje kriterijuma kod pregleda studenata
38. Lista studenata
39. Formular za dodavanje novog ispitnog roka
40. Formular za odabir ispitnog roka i tipa izmene
41. Formular za izmenu termina i lokacija ispita u ispitnom roku
42. Tabela sa listom odabranih termina i lokacija po ispitu
43. Zahtev za potvrdom objavljivanja ispitnog roka
44. Poruka o uspešnom objavljivanju ispitnog roka.
45. Poruka pri pokušaju uklanjanja ispitnog roka.
46. Lista svih ispitnih rokova po fakultetima
47. Formular za odabir roka, predmeta, akcije i dodatnih parametara
48. Lista studenata koji su prijavili predmet
49. Varijanta za štampu zapisnika
50. Formular za popunjavanje zapisnika
51. Odabir studenta za koga se prijavljuju ispiti
52. Odabir ispitnog roka za koji se prijavljuju ispiti
53. Odabir predmeta (ispita) koji se prijavljuju
54. Poruka o uspešnoj prijavi ispita za studenta.
55. Formular za unos jedinstvenog identifikatora prijave koja se overava
56. Prikaz parametara prijave i zahtev za potvrdom overe
57. Poruka o uspešnoj overi prijave ispita
58. Formular za odabir fakulteta i ispitnog roka za koji se pregledaju prijave ispita
59. Lista studenata koji su prijavili ispite za odabrani ispitni rok
60. Prijava studenata na sistem
61. Osnovni ekran nakon uspešne prijave studenta
62. Odabir ispitnog roka za koji se postavlja prijava
63. Odabir predmeta koji se prijavljuju
64. Zahtev za potvrdom prijave
65. Informacija u uspešnoj prijavi i roku za uplatu.

66. Lista postavljenih prijava ispita
67. Detaljan prikaz odabrane prijave sa popunjenim nalogom za uplatu
68. Slojevi na kojima je potrebno vršiti zaštitu sistema
69. Mrežna realizacija informacionog sistema baziranog na SOA arhitekturi
70. Odnos spoljnih IS-a sa centralnim

Spisak tabela

1. Vremenski plan realizacije po modulima
2. Prošireni vremenski plan realizacije po modulima

1. Uvod

Univerzitet se sa poslovnog aspekta može sagledati kao specifični poslovni sistem visoke kompleksnosti, velikog broja i diverziteta učesnika u poslovnim procesima koji su najčešće čvrsto povezani i zavisni jedni od drugih. Takva organizacija čak i u svojim najranijim fazama zahteva informacioni sistem sposoban da podrži većinu poslovnih procesa, njihovu evoluciju i integraciju kao i da pruži visok nivo pouzdanosti kroz otpornost na korisničke greške (slučajne i pokušaje zloupotrebe) i kvalitet unutrašnjih i spoljnih procedura sistema. Predmet ovog rada je da analizira zahteve koji se stavljaju pred jedan ovakav informacioni sistem, napravi model sistema i istraži savremena tehnološka dostignuća koja omogućavaju njegovu realizaciju. Ovaj rad se bazira i na praktičnom delu u vidu projekta nazvanog „UNIS – University Network Information System“.

U II delu rada je urađena analiza poslovnog sistema Univerziteta tj. dela poslovnog sistema koji je potrebno podržati informacionim sistemom. Analiza poslovnog sistema počinje analizom zahteva koji su postavljeni pred informacioni sistem i nastavlja se kroz analizu poslovnih procesa i kategorija korisnika informacionog sistema.

III deo rada se bavi analizom trenutno aktuelnih softverskih komponenti, arhitektura, tehnologija i gotovih rešenja. Ovaj deo obrađuje proverena ali i najsavremenija rešenja koja pokazuju tendenciju i potencijale da postave buduće standarde razvoja softverskih sistema. U ovom delu se obrađuju i EMS/ECMS platforme koje čine osnovu za većinu softverskih projekata na univerzitetu Singidunum i koje su takođe delo autora ovog rada.

Rezultati analiza dobijeni iz II i III dela rada čine predmet rada IV dela u kome se daje model informacionog sistema. S obzirom na kompleksnost projekta model informacionog sistema se obrađuje u okviru modela arhitekture IS-a, modela baze podataka, modela korisničkih modula kao i plana koji je korišćen u realizaciji.

Praktični deo rada predstavlja fokus V dela u kome se opisuju realizovane funkcionalnosti i način(i) njihovog korišćenja. Ovaj deo se može smatrati posebno korisnim kada je u pitanju upotreba IS-a u praksi jer čini osnovnu dokumentaciju na osnovu koje se korisnici mogu inicijalno upoznati sa sistemom i pravilnim načinom korišćenja.

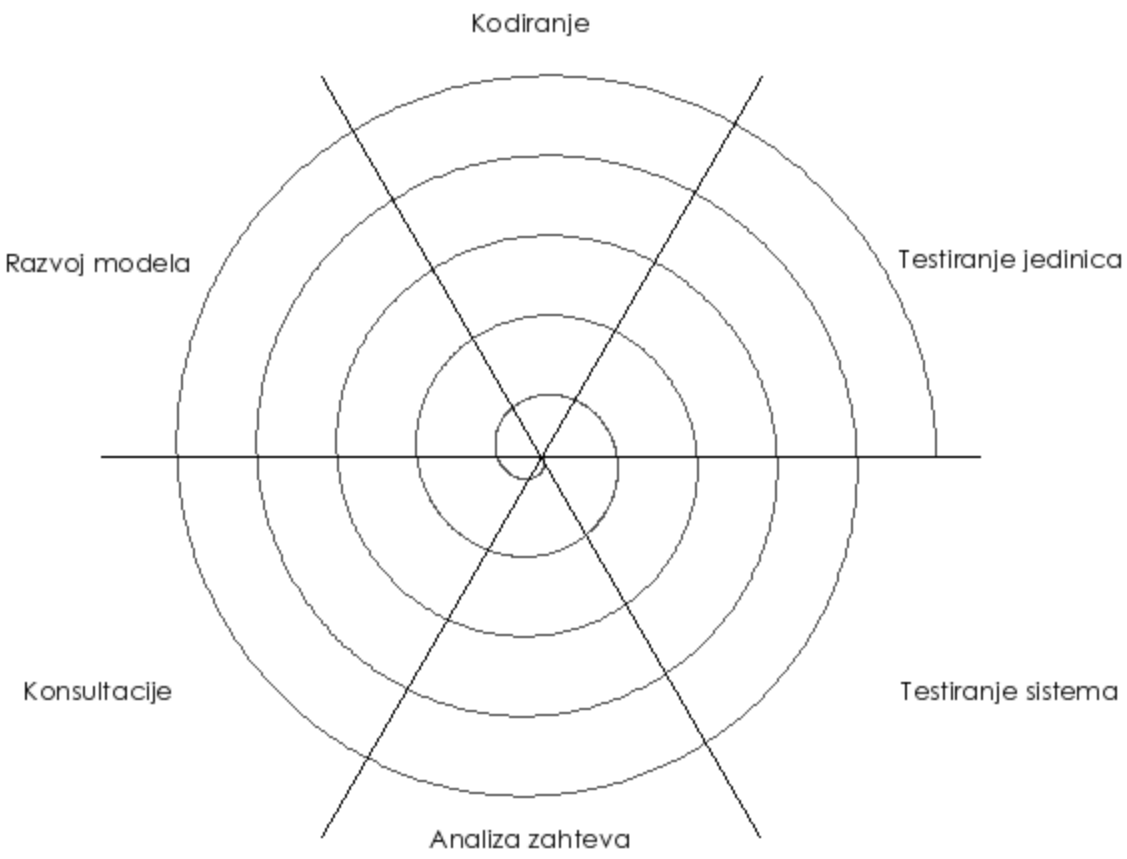
VI deo rada se bavi budućim pravcima razvoja informacionog sistema. Ovaj deo se bavi bezbednosnim slabim tačkama rešenja i načinima njihove optimizacije, predviđenim funkcionalnostima koje nisu realizovane, načinima uvođenja novih funkcionalnosti. Jedan od važnih fokusa VI dela rada jeste i prevođenje na SOA arhitekturu u cilju povezivanja informacionog sistema sa spoljnim informacionim sistemima (kao što su sistem za automatizovano testiranje studenata i sistem za Distance Learning) tj. omogućavanja korišćenja podataka/procedura informacionog sistema tim sistemima.

Zaključni deo rada sadrži rezime teorijskog i praktičnog dela i govori o sledećim ostvarenim rezultatima:

1. funkcionalnom softverskom rešenju - informacionom sistemu univerziteta.
2. dokumentaciji vezanoj za realizaciju rešenja.
3. dokumentaciji vezanoj za dalje pravce razvoja rešenja.
4. dokumentaciji vezanoj za korišćenje softvera.
5. dokumentaciji koja opisuje trenutna dostignuća vezana za tehnologije za izradu softvera.

2. Analiza poslovnog sistema

Za izgradnju informacionog sistema jedne poslovne organizacije potrebno je izvršiti adekvatnu analizu poslovnog sistema i na osnovu rezultata te analize napraviti model koji realizovan putem informacionih tehnologija čini informacioni sistem poslovne organizacije. Iako Univerzitet predstavlja pre svega edukativnu organizaciju pri izgradnji informacionog sistema neophodno je promeniti aspekat i u analizi koristiti atribute koji se odnose na poslovne strukturu, procese i sl.



Slika 1 – Spiralni model etapa razvoja.

U analizu poslovnog sistema spadaju etape *analiza zahteva* i *konsultacije* [9]. U slučajevima kada postoji kompleksnija procedura postavljanja zahteva (veći broj osoba ovlašćenih da daju zahteve, različiti prioriteti itd.) potrebno je obezbediti posebne mehanizme za upravljanje zahtevima klijenata. U slučaju projekta „UNIS“ nije bilo potrebe za takvim mehanizmima usled malog broja inicijalnih zahteva.

Drugu fazu analize poslovnog sistema čine konsultacije sa nadležnim članovima organizacije tj. razvoj korisničkih zahteva. Ovaj deo rada se bavi analizom poslovnog sistema kroz analizu i razvoj korisničkih zahteva.

2.1. Analiza zahteva

Osnovni zahtev koji je otvorio pitanje izgradnje informacionog sistema je bio omogućavanje studentima koji studiraju putem Interneta (Distance Learning) na univerzitetu Singidunum da on-line prijave ispite. Razmatranjem ovog zahteva došlo se do sledećih zaključaka:

1. Potrebno je napraviti bazu podataka o studentima.
2. Potrebno je napraviti bazu podataka o predmetima.
3. Potrebno je napraviti bazu podataka o ispitnim rokovima.
4. Potrebno je napraviti bazu o finansijskim parametrima studenta jer oni mogu biti kriterijum za prijavljivanje ispita.
5. Potrebno je napraviti veze između pomenutih objekata koje ograničavaju studente na prijavljivanje njima dostupnih ispita.
6. Potrebno je napraviti sistem u vidu Web aplikacije koji omogućava prijavljivanje studenata na sistem putem naloga zaštićenog lozinkom.
7. Potrebno je pridružiti jedinstveni identifikator svakoj prijavi u cilju evidentiranja uplate troškova polaganja ispita u računovodstvu.
8. Potrebno je omogućiti štampanje ispitnih spiskova kreiranih u ovom sistemu.

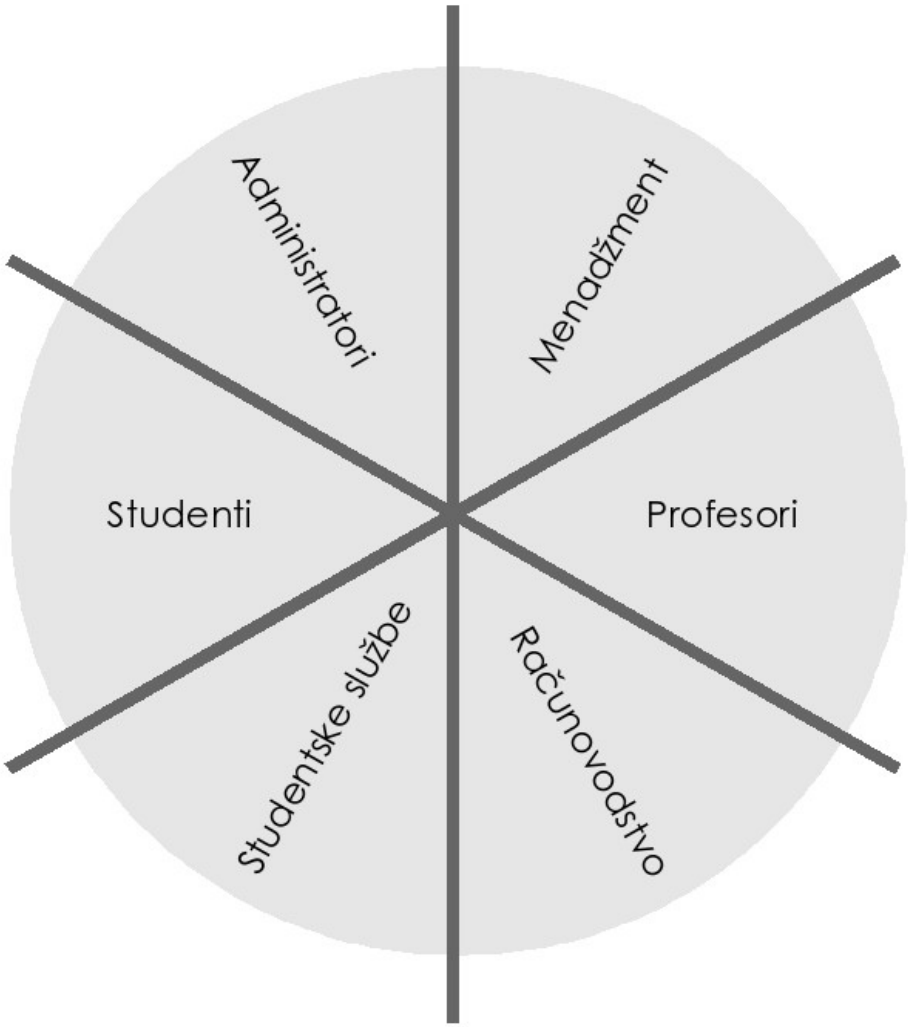
S obzirom na nepostojanje ažurne i funkcionalne centralne baze podataka Univerziteta pogodne da pruži potrebne informacije sistemu za on-line prijavu ispita došlo se do zaključka da se uz manje izmene i proširenja projekat može usmeriti u pravcu glavnog informacionog sistema Univerziteta sa interfejsom za on-line prijavu ispita. Izvučeni zaključak se bazira na tome da je analizom komponenti koje je potrebno napraviti za sistem za on-line prijavu ispita identifikovano preko 90% onih koje je moguće iskoristiti i za brojne druge aktivnosti na Univerzitetu. Takođe, s obzirom na nepostojanje informacionog sistema na Univerzitetu realizacija usko-orijentisanog sistema za on-line prijavu bi u budućnosti dovela do potrebe za sinhronizacijom podataka sa eventualno uvedenim IS-om (što bi moglo da predstavlja problem s obzirom na nedostatak informacija o potencijalnom IS-u) ili bi dovela do neadekvatnog (nenamenskog) korišćenja sistema za on-line prijavu i pridružene baze podataka.

Dolaženjem do usmerenja ka izgradnji kompletnog informacionog sistema otvoreno je novo pitanje tj. pitanje resursa (pre svega vremenskih) potrebnih za realizaciju kompletnog informacionog sistema. Zaključak izvučen na ovu temu je da su resursi potrebni za realizaciju delova IS-a neophodnih za funkcionalnost on-line prijave ispita svega 10-20 procenata veći od resursa potrebnih za realizaciju namenskog sistema za on-line prijavu ispita. Takođe, ovakav način realizacije uslovljava modularan dizan IS-a koji bi omogućio faznu realizaciju komponenti i proširivost.

Kao što je ranije napomenuto, u projektu „UNIS“ nije korišćen poseban sistem za upravljanje korisničkim zahtevima usled jednostavnosti inicijalnih zahteva. Međutim, s obzirom na kompleksnost projekta, izvesna su proširanja u budućnosti koja će zahtevati uvođenje takvog sistema.

2.2. Analiza kategorija korisnika IS-a

Za razvoj uspešnog informacionog sistema, osim adekvatne analize poslovnih procesa i razvoja modela kroz konsultacije, potrebno je izvršiti i kategorizaciju korisnika, pre svega u slučajevima kada IS koristi veliki broj korisnika koji pristupaju različitim grupama podataka i sa različitim ovlašćenjima. U tu svrhu izvršena je kategorizacija korisnika paralelno sa analizom poslovnih procesa tj. putem utvđivanja učesnika u utvrđenim procesima.



Slika 2 – Kategorije korisnika

Ovom analizom izdvojeno je šest grupa (kategorija) korisnika informacionog sistema. U okviru svake od grupa se nalaze korisnici koji učestvuju u istim poslovnim procesima (ili u određenim fazama procesa) i na koje se može primeniti isti skup funkcionalnosti, privilegija i ograničenja. Izdvojene grupe korisnika su:

1. Administratori
2. Menadžment
3. Profesori
4. Računovodstvo

5. Studentske službe
6. Studenti

Uloga korisnika informacionog sistema kategorije „Administratori“ ima najmanji broj dodirnih tačaka sa poslovnim procesima. Ova kategorija je, međutim, zadužena omogućavanje rada ostalim korisnicima sistema putem praćenja stanja sistema, kreiranja rezervnih kopija baze podataka, prilagođavanja informacionog sistema promenama u poslovnom sistemu, otvaranja korisničkih naloga za korisnike ostalih kategorija i sl.

Uloga korisnika informacionog sistema ostalih kategorija biće data u delu koji se bavi analizom poslovnih procesa iz razloga što su identifikovani poslovni procesi pridruženi odgovarajućim kategorijama korisnika.

2.3. Analiza poslovnih procesa

U analizi poslovnih procesa koje informacioni sistem treba da podrži nisu učestvovali svi procesi već samo oni koji obezbeđuju inicijalnu funkcionalnost sistema i oni koji daju dovoljno informacija na osnovu kojih je moguće projektovati bazu podataka sposobnu da podrži buduće izmene. Osnovni zahtev koji je postavljen pred informacioni sistem jeste on-line prijava ispita. U skladu sa tim, osnovni proces koji je analiziran jeste postojeći proces prijave ispita uz izmene neophodne za omogućavanje on-line varijante. Postojeći proces obuhvata tri grupe učesnika - studentske službe, profesore i studente. Koraci koje oni obavljaju jesu:

1. **Studentska služba:**
   Pravi raspored predmeta za polaganje u novom ispitnom roku.
2. **Studentska služba:**
   Objavljuje raspored polaganja ispita u ispitnom roku.
3. **Studentska služba:**
   Kreira prazan spisak studenata koji su prijavili ispit za svaki od predmeta.
4. **Student:**
   Vrši odabir predmeta uz napomenu koji put polaže ispit i popunjava formular.
5. **Student:**
   Množi broj odabranih ispita sa cenom ispita i uplaćuje iznos na žiro-račun fakulteta.
6. **Student:**
   Predaje studentskoj službi formular i overen nalog za uplatu.
7. **Studentska služba:**
   Proverava da li student ima pravo da polaže odabrane ispite na osnovu kriterijuma:
   1. trenutna godina studija studenta
   2. stanje vezano za izmirenje školarine
   3. eventualne zabrane
   4. da li je naveden broj polaganja tačan
   5. da li je uplaćeni iznos adekvatan
8. **Studentska služba:**
   U slučaju da nisu zadovoljeni svi uslovi iz sedmog koraka obaveštava studenta.
9. **Student:**
   Vrši korekciju i vraća se na korak 6.
10. **Studentska služba:**
    Nakon zadovoljenja svih uslova iz sedmog koraka unosi ime, prezime i broj indeksa na ispitne spiskove predmeta koje je student prijavio.
11. **Studentska služba:**
    Na dan svakog od ispita zaključuje ispitne spiskove i na osnovu njih pravi zapisnik o polaganju ispita.
12. **Profesor:**
    Na dan ispita preuzima zapisnik o polaganju na osnovu koga evidentira prisutne studente i koji popunjava ocenama koje su studenti ostvarili na polaganju.

13. **Profesor:**
    Dostavlja studentskoj službi popunjen zapisnik o polaganju ispita.
14. **Studentska služba:**
    Zaključuje zapisnik i stavlja u arhivu.

Ovih 14 hronološki poređanih koraka predstavlja celokupan proces prijave ispita zaključno sa zaključivanjem zapisnika i postavljanjem u arhivu. Jedini iskorak iz ovog toka koraka mogu predstavljati koraci 6-9 u okviru kojih je moguće ponavljanje. Takođe, korak 7 se može smatrati veoma zahtevnim u pogledu vremena, napora i resursa koje studentska služba mora da utroši a ujedno predstavlja i tačku u kojoj se povećava mogućnost greške. Informacioni sistem je projektovan tako da taj deo posla automatski obavlja umesto studentske službe a studenta gotovo trenutno obaveštava o eventualnim neispunjenjima uslova 7.1-7.4. Takođe, treba pomenuti da dodatnu kontrolu finansijskog dela procesa vrši računovodstvo koje na eventualne propuste u sumarnim rezultatima ukazuje studentu na kraju školske godine. Za potrebe on-line prijave ispita opisani proces je izmenjen tako da je ubačen dodatni infrastrukturni učesnik - informacioni sistem - a koraci procesa i njihov redosled izgledaju ovako:

1. **Studentska služba:**
   Pravi raspored predmeta za polaganje u novom ispitnom roku u okviru informacionog sistema.
2. **Studentska služba:**
   Objavljuje raspored polaganja ispita u ispitnom roku u okviru informacionog sistema.
3. **Informacioni sistem:**
   Kreira prazan spisak studenata koji su prijavili ispit za svaki od predmeta.
4. **Student:**
   Pristupa sistemu i odabira akciju prijavljivanja ispita.
5. **Informacioni sistem:**
   Učitava objavljene ispitne rokovi i za svaki predmet u okviru svakog od rokova proverava da li student ima pravo da polaže odabrane ispite na osnovu kriterijuma:
   1. trenutna godina studija studenta
   2. stanje vezano za izmirenje školarine
   3. eventualne zabrane
   4. da li je naveden broj polaganja tačan
6. **Informacioni sistem:**
   Na osnovu odabranih ispita od strane studenta izračunava sumu koju je student dužan da uplati i izveštava studenta o ukupnoj sumi uz prikaz odabranih ispita. Informacioni sistem zahteva potvrdu od studenta.
7. **Student:**
   Potvrđuje prijavu odabranih ispita. (U protivnom se ceo proces prijave poništava.)
8. **Informacioni sistem:**
   Nakon potvrde studenta evidentira novu prijavu kojoj pridružuje jedinstveni identifikacioni broj (prijava se u bazi vodi kao neoverena do trenutka evidentiranja uplate). Studenta pridružuje ispitnim spiskovima svih prijavljenih predmeta. Kreira izgled popunjenog naloga za uplatu sa pomenutim jedinstvenim identifikacionim brojem kao pozivom na broj.

9. **Student:**
   Nakon dobijanja izgleda popunjenog naloga za uplatu vrši uplatu na žiro račun-fakulteta.
10. **Računovodstvo:**
    Evidentira uplatu i o vrednosti poziva na broj (kao jedinstvenom identifikacionom parametru prijave) obaveštava studentsku službu.
11. **Studentska služba:**
    Unosi vrednost dobijenu od računovodstva u informacioni sistem i na taj način overava prijavu.
12. **Studentska služba:**
    Na dan svakog od ispita zaključuje ispitne spiskove i štampa ih putem opcije u informacionom sistemu.
13. **Profesor:**
    Na dan ispita preuzima zapisnik o polaganju na osnovu koga evidentira prisutne studente i koji popunjava ocenama koje su studenti ostvarili na polaganju.
14. **Profesor:**
    Dostavlja studentskoj službi popunjen zapisnik o polaganju ispita.
15. **Studentska služba:**
    Unosi vrednosti zapisnika u informacioni sistem. Zaključuje zapisnik i stavlja u arhivu.

Na ovaj način informacioni sistem, ne samo što omogućava on-line prijavu već nudi i dodatnu bazu svih prijava ispita kao i ostvarenih rezultata (bodova i ocena) studenata što znatno olakšava i automatizuje provere 1-4 7. koraka originalne postavke procesa. Međutim, analizom obe postavke procesa može se zaključiti i da je proces prijave ispita tesno povezan sa mnoštvom drugih informacija u poslovnom sistemu počev od informacija vezanih za:

- fakultete
- predmete
- programe
- profesore
- studenate
- školarine
- ispitne rokove
- lokacije
- itd.

te da je moguće izgraditi veoma neprecizan i neautomatizovan sistem za prijavu ispita bez ovih informacija ili je te informacije potrebno staviti informacionom sistemu na raspolaganje i na taj način postići tačnost i automatizaciju. U skladu sa tim i sa činjenicom da potrebne informacije ne postoje u centralnoj bazi pogodnoj za korišćenje od strane IS (tj. ne postoje u elektronskom obliku) zaključeno je treba napraviti centralnu bazu potrebnih informacija i rad sa tim informacijama treba podržati putem proširenog sistema za on-line prijavu ispita što vodi ka izgradnji informacionog sistema univerziteta.

3. Analiza softverskih arhitektura, tehnologija i gotovih rešenja

U ovom delu rada izvršena je analiza arhitektura i tehnologija vezanih za razvoj softvera putem kojih je moguće realizovati rešenje kao i analiza gotovih rešenja/komponenti dostupnih na tržištu. Nakon analize zahteva i poslovnih procesa po AMP modelu:

1. Analiza
2. Modeliranje
3. Programiranje

potrebno je izvšiti i analizu pomenutih komponenti rešenja i razvoja:

1. softverskih arhitektura
2. tehnologija i
3. gotovih rešenja/komponenti

da bi se omogućio razvoj optimalnog modela koji:

1. je moguće realizovati i
2. koji će biti realizovan putem aktuelnih rešenja

Eventualne greške u ovoj fazi razvoja mogu u potpunosti onemogućiti realizaciju informacionog sistema ili dovesti do rešenja koje (je):

- zastarelo pre same realizacije
- neefikasno po pitanju performansi i/ili tačnosti
- nije moguće proširivati
- nije moguće povezivati sa ostalim savremenim softverskih rešenjima
- previše skupo, inicijalno i u fazi održavanja
- usko vezano za određenog proizvođača rešenja/komponenti
- ne uključuje adekvatne gotove i proverene komponente
- ...

U skladu sa tim ovaj deo rada detaljno opisuje proces analize na čijim je rezultatima obavljeno modeliranje i realizacija informacionog sistema.

3.1. Analiza softverskih arhitektura

Odabir adekvatne softverske arhitekture na kojoj će informacioni sistem biti zasnovan predstavlja važan zadatak iz tok razloga što bi odabir neadekvatne arhitekture narušio model IS-a kroz izmene uslovljene implementacijom. U ovom delu rada je dat pregled popularnih arhitektura:

1. Host-based arhitektura
2. Klijent-server arhitektura
3. Peer-to-peer arhitektura
4. SOA arhitektura

kao i pregled njihovih varijanti, prednosti i mana i sl.

Za realizaciju informacionog sistema odabrane su klijent-server arhitektura sa host-based elementima i SOA arhitektura. Peer-to-peer arhitektura je u ovom stadijumu razvoja odbačena kao izbor ali je njena primena u budućnosti moguća s obzirom na modularan dizajn IS-a.

3.1.1. Host-based arhitektura

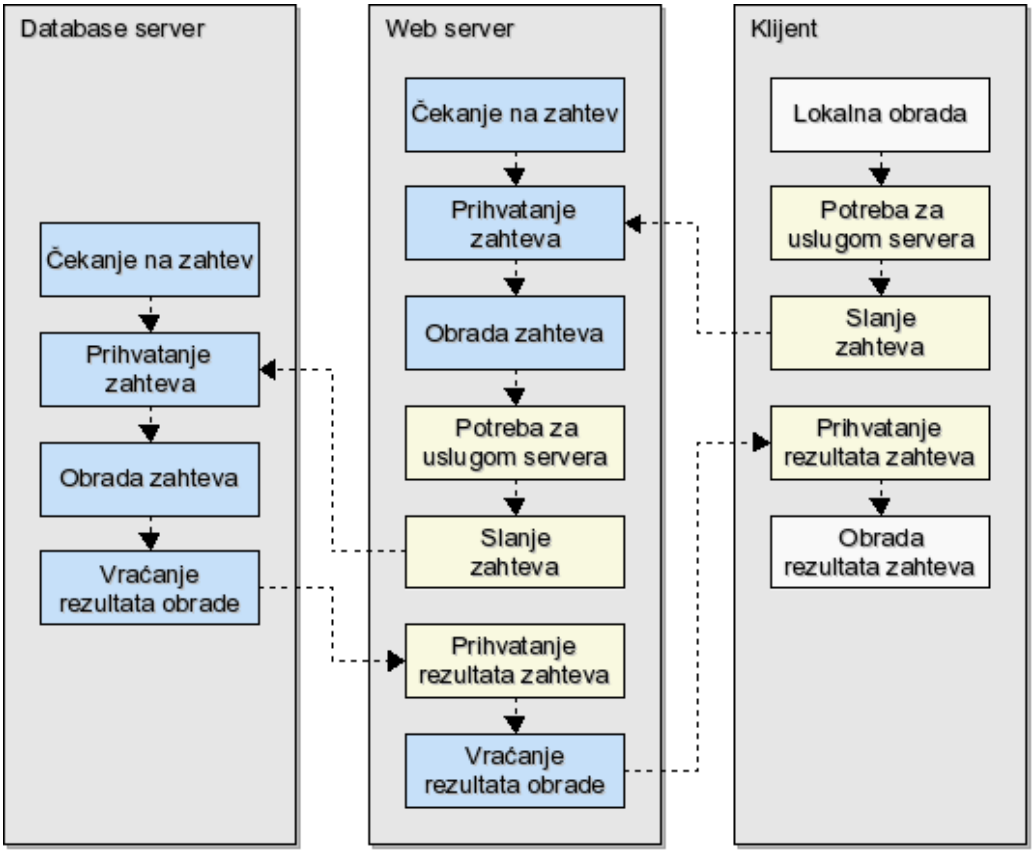
[1] Host-based mreže su zasnovane na jednoj od najstarijih mrežnih arhitektura. Kod mreža ovog tipa u centru mreže se nalazi jak mainframe računar na koga su povezani terminali. Razlika između mainframe i terminal članova je u tome što mainframe računar obavlja sva izračunavanja dok terminali služe samo kao interfejs sa korisnicima. Host-based arhitektura svoju popularnost u ranim fazama razvoja računara i računarskih mreža duguje visokoj ceni procesorske moći koju je bilo neekonomično pridruživati korisničkim računarima, terminalima.

Danas se mreže sa host-based arhitekturom retko sreću u svom izvornom obliku. Razlog zbog koga je host-based arhitektura danas u upotrebi nije velika razlika u ceni između terminala i centralnih servera već je u pitanju smanjenje troškova kroz centralizovanu administraciju.

Host-based arhitektura je više usmerena ka mrežnoj nego ka softverskoj organizaciji. Elementi host-based arhitekture se mogu naći kod i u organizaciji Web-a (uzevši u obzir da se sve operacije i podaci mogu izvršavati i nalaziti na serveru) ali se sa obzirom na postojanje Web klijentskog softvera i njemu pridruženo procesiranje (parsiranje sadržaja, renderovanje prikaza, izvršavnje JavaScript instrukcija) Web bazira na klijent-server arhitekturi.

3.1.2. Klijent-Server arhitektura

[1] Klijent-server arhitektura je jedan od najčešće korišćenih pristupa kod distribuirane obrade podataka. Koreni ove arhitektura se nalaze kod mainframe računara (host-based arhitektura) i njima priključenih terminala. Sličnost sa ovom arhitekturom jeste postojanje jednog člana sposobnog za izvršavanje zadataka koji su van mogućnosti ostalih članova mreže. Postoje, međutim, bitne razlike između ove dve arhitekture. Kod host-based arhitekture terminali nemaju nikakvu mogućnost obrade podataka dok kod klijent-server arhitekture klijenti od servera dobijaju podatke koje zatim koriste u lokalnom procesu obrade. Zatim, mainframe računari predstavljaju autonomne članove mreže koji za proces obrade podataka koriste lokalne resurse. Nasuprot tome, server se u vidu klijenta može obratiti drugim serverima u mreži za određeni resurs ili distribuiranu obradu.



Slika 3 - Slojevita klijent-server arhitektura

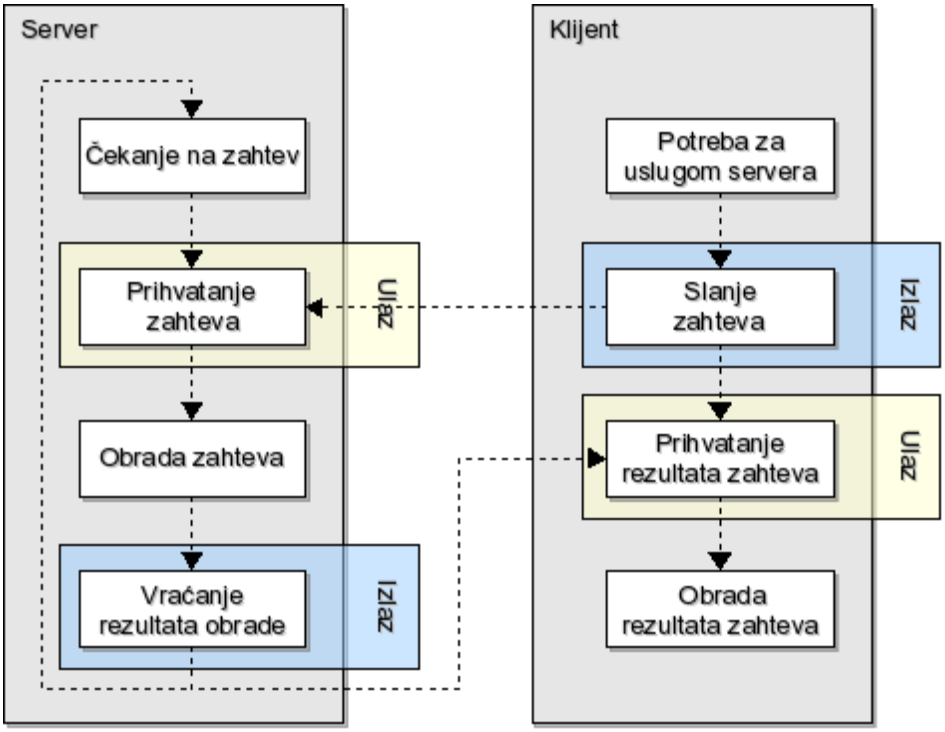
S obzirom na to da je klijent-server arhitektura iskorišćena u vidu Web aplikacija koje može da karakteriše veliki broj korisnika istovremeno (pre svega u modulu za studente) performanse i konkurentni pristup su dva pitanja na koja treba obratiti pažnju pri realizaciji. Različita serverska rešenja imaju različite pristupe za povećanje performansi kroz vertikalnu i/ili horizontalnu skalabilnost. Najčešći primer horizontalne skalabilnosti jesu klasteri i load-balancer rešenja. Na mogućnost vertikalne skalabilnosti (povećanje internih resursa) najčešće utiče model realizacije u vidu iterativne ili konkurentne obrade zahteva.

3.1.2.1. Iterativna i konkurentna obrada zahteva

[1] U zavisnosti od načina obrade zahteva servere možemo podeliti na:

- servere sa iterativnom obradom zahteva
- servere sa konkurentnom obradom zahteva.

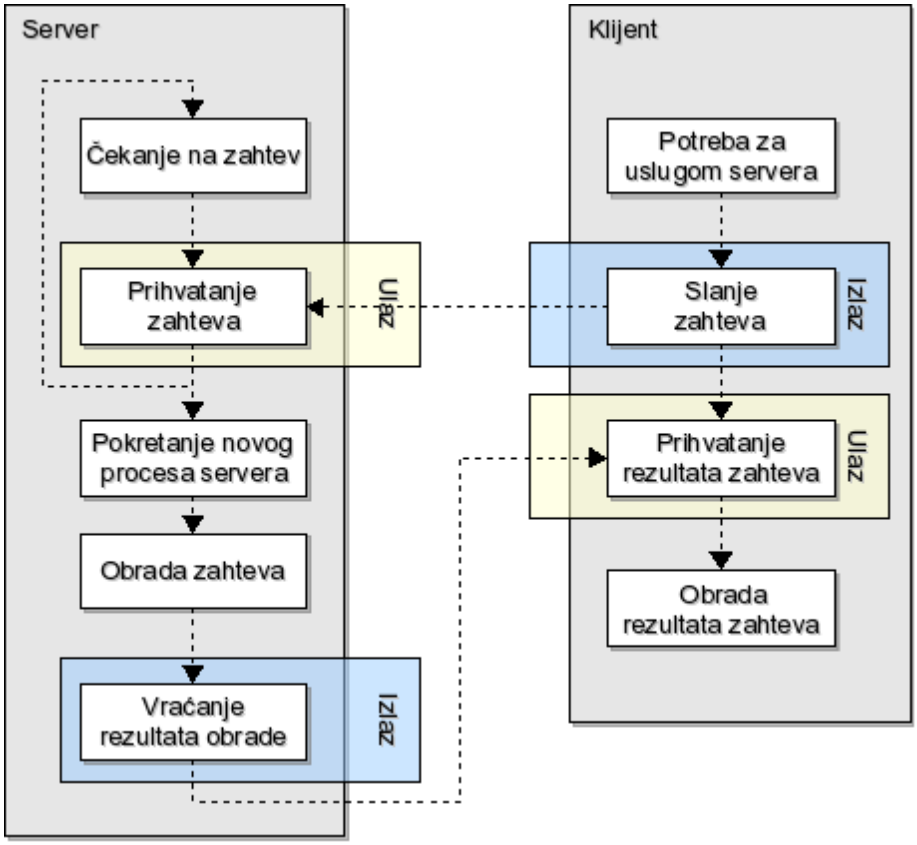
Serveri sa iterativnom obradom zahteva osluškuju na dodeljenom portu čekajući na zahtev klijenta. Nakon prihvatanja zahteva klijenta zahtevi ostalih klijenata se odbacuju ili čekaju u ulaznom baferu sve dok se prihvaćeni zahtev ne obradi i rezultati njegove obrade pošalju natrag klijentu.



Slika 4 - Iterativna obrada zahteva

Mana iterativne obrade zahteva je u tome što takav pristup može znatno uticati na performanse u smislu broja obrađenih zahteva po jedinici vremena. Ukoliko obrada jednog zahteva u toku svog izvršenja zauzme sve resurse servera performanse se mogu smatrati optimalnim. Međutim, ukoliko obrada zahteva oduzme dodatno vreme usled čekanja na resurs koji nije potreban za obradu ostalih zahteva (koji su odbačeni ili čekaju u ulaznom baferu), iterativni pristup pokazuje lošije performanse od konkurentog. Glavna prednost iterativnog pristupa jeste eliminisanje problema konkurentnog pristupa internim resursima servera time što se u jednom trenutku obrađuje samo jedan zahtev tako da ne može doći do višestrukih zahteva za istim resursom.

Konkurentna obrada zahteva je pristup koji nudi bolje performanse od iterativnog pristupa u situacijama u kojim server obrađuje veliki broj zahteva od strane više klijenata. Poboljšanje pefromansi potiče od mogućnosti obrade više zahteva paralelno. Paralelna obrada se postiže pokretanjem novog procesa (ili niti procesa, u zavisnosti od operativnog sistema) za obradu zahteva kod svakog klijentskog zahteva.



Slika 5 - Konkurentna obrada zahteva

Za ovakav pristup je potreban kompleksniji serverski softver koji se sastoji od dispečerskog dela (dela koji je zadužen za prihvatanje zahteva i pokretanje procesa njihove obrade) i dela koji je zadužen za konkretnu obradu zahteva. Konkurentna obrada zahteva može u određenim situacijama pokazati slabije performanse od iterativne obrade usled trošenja procesorskog vremena na pokretanje novih procesa za obradu zahteva. Takođe, softver koji omogućava konkurentnu obradu je kompleksniji jer interno rešava konkurentni pristup sistemskim resursnima. Softver za konkurentnu obradu najčešće unapred pokreće određen broj procesa za obradu zahteva a po potrebi taj broj povećava do konfiguracione vrednosti ili ograničenja sistemskim resursima.

3.1.3. Peer-to-Peer arhitektura

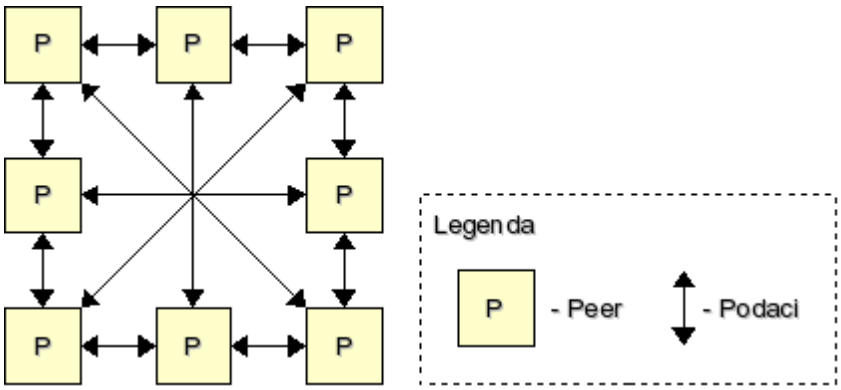
[1] Peer-to-peer (P2P) arhitektura predstavlja vid distrubuiranog računarstva u kome svaki čvor (eng. node) ima dvostruku ulogu. Svaki čvor P2P mreže komunikaciju sa ostalim članovima P2P mreže obavlja putem simetričnog softvera koji se može ponašati i kao klijent (zahtevajući podatke ili usluge od ostalih čvorova) i kao server (odgovarajući na zahteve ostalih čvorova). Na ovaj način P2P arhitektura omogućava veću autonomiju članova mreže. P2P arhitektura se uglavnom primenjuje kod potreba u kojima postoji veća tolerancija greške kod distribuirane odbrade. Glavne primene su:

1. Razmena fajlova
2. Komunikacija
3. Distribuirana obrada ogromne količine podataka
4. Heš tabele
5. Softver za zabavu

Glavni nedostatak P2P arhitekture jeste adresiranje članova mreže. Dok je kod klijent-server mreža potrebno samo da klijenti imaju informaciju o tome koji serveri su dostupni na mreži (i koja je njihova adresa) kod P2P arhitekture je potrebno da svaki član ima informaciju dostupnosti ostalih članova. Iz tog razloga postoji više različitih arhitektura unutar P2P arhitekture:

1. decentralizovana arhitektura
2. centralizovana arhitektura
3. hibridna arhitektura

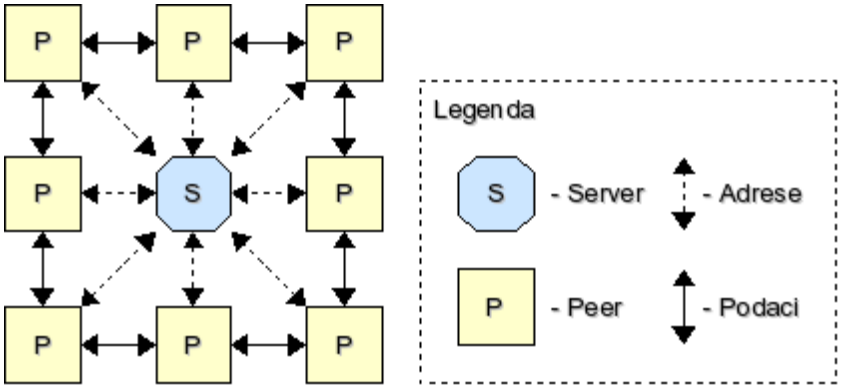
Decentralizovana P2P arhitektura predstavlja arhitekturu najbližu osnovnom P2P modelu. Ona je sačinjena isključivo od peer čvorova koji međusobno komuniciraju direktno.



Slika 6 - Šema decentralizovane P2P mreže

Kod decentralizovane P2P arhitekture ne postoji centralni registar članova već se otkrivanje ostalih članova vrši preko internog protokola (najčešće u vidu broadcast zahteva).

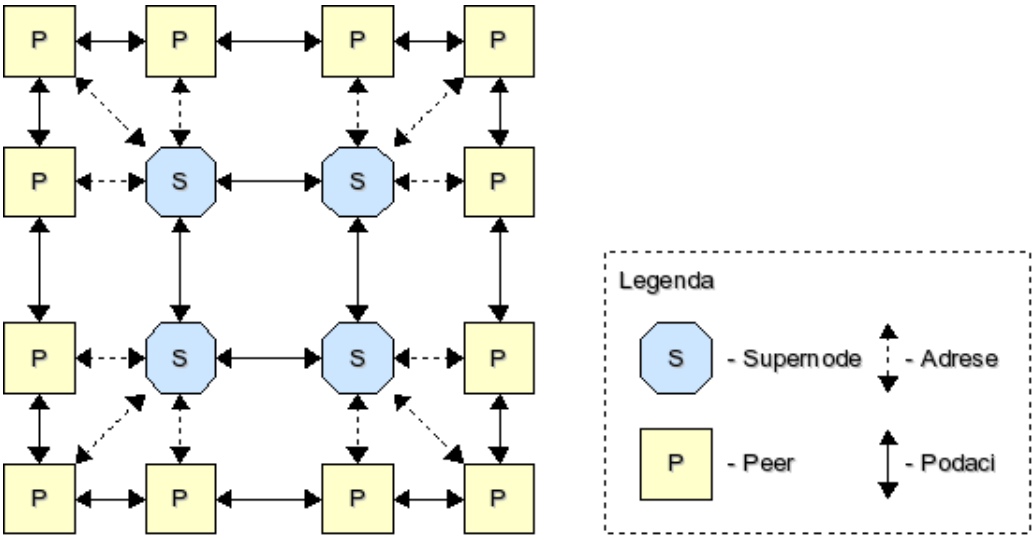
Centralizovana P2P arhitektura predstavlja mešavinu P2P i klijent-server arhitektura.



Slika 7 - Šema centralizovane P2P mreže

Kao i kod decentralizovane P2P arhitekture mrežu čine peer čvorovi koji međusobno komuniciraju direktno sa tom razlikom da postoji centralni server čiji je zadatak evidentiranje peer članova mreže.

Hibridna P2P arhitektura predstavlja varijantu centralizovane P2P arhitekture koja se koristi u slučajevima kada se mreža sastoji od velikog broja peer čvorova i/ili uloga servera podrazumeva i dodatne operacije sem evidentiranja.



Slika 8 - Šema hibridne P2P mreže

Kod hibridne P2P arhitekture ulogu servera preuzima veći broj Supernode čvorova. Ove čvorove najbliži peer čvorovi koriste kao servere dok adresne informacije vezane za peer čvorove supernode čvorovi međusobno razmenjuju.

3.1.4. SOA arhitektura

Servisno-orijentisana arhitektura (eng. Service-Oriented Architecture, SOA) predstavlja iskorak u oblasti razvoja softvera značajan koliko i prelazak sa proceduralnog na objektno-orijentisani način razmišljanja, analize, dizajna i razvoja. Iako deli mnoge karakteristike sa OO konceptima (ponovno iskorišćenje komponenti, ugovaranje interfejsa itd.), SOA uvodi nove potencijale (pre svega kod izgradnje mrežnih informacionih sistema) koji je ističu kao ultimativnu platformu savremenih distribuiranih sistema za obradu podataka. Servisno-orijentisana arhitektura se može smatrati i primenom proverenih OO koncepata u cilju odgovora na savremene potrebe za distribuiranom obradom podataka i komunikacijom između raznorodnih softverskih sistema.

3.1.4.1. Proceduralni i objektno-orijentisani pristup

Jedna od glavnih karakteristika servisno-orijentisane arhitekture je proširivanje skupa funkcionalnosti koje mogu biti iskorišćene u softverskom rešenju. Ovo proširivanje se najbolje može prikazati poređenjem sa objekto-orijentisanim konceptima i njihovim poređenjem sa proceduralnim razvojem softvera. Glavna karakteristika OO "revolucije" je kategorizacija i adresiranje već urađenih funkcionalnosti što programerima omogućava viši nivo apstrakcije u rešavanju problema i mogućnost iskorišenja gotovih komponenti u novom softverskom proizvodu. Uzmimo za primer sledeći proceduralni kod:

```
var stanje;
function dodaj(iznos) {
   global stanje;
   stanje += iznos;
}
function oduzmi(iznos) {
   global stanje;
   stanje -= iznos;
}
function proveri() {
   global stanje;
   return stanje;
}
```

Ovaj kod prikazuje tri korisnički definisane funkcije koje dele zajedničku (globalnu) promenljivu. Pretpostavimo da se kod odnosi na upravljanje tekućim računom korisnika. Osnovni nedostatak prikazanog rešenja (tj. nedostatak proceduralnih programskih jezika) jeste globalno opterećivanje sistema lokalnim funkcionalnostima (ili, bolje rečeno, jednom funkcionalnošću). Konkretno, u prikazanom kodu su rezervisane četiri opšte reči (stanje, dodaj, oduzmi, proveri) i njihova upotreba u kompleksnijem softveru (čiji bi prikazani kod bio samo modul) mogla bi se rezultovati greškom (sistemskom ili logičkom) ukoliko neki drugi modul koristi pomenute reči.

Bolje organizovan proceduralni kod bi mogao da se dobije dodavanjem prefiksa koji bi na neki način adresirao funkcije i promenljive:

```
var TekuciRacun_stanje;
function TekuciRacun_dodaj(iznos) {
```

```
   global TekuciRacun_stanje;
   TekuciRacun_stanje += iznos;
}
function TekuciRacun_oduzmi(iznos) {
   global TekuciRacun_stanje;
   TekuciRacun_stanje -= iznos;
}
function TekuciRacun_proveri() {
   global TekuciRacun_stanje;
   return TekuciRacun_stanje;
}
```

Ovakav pristup olakšava integraciju sa drugim softverskim modulima, štedi "opšte" pojmove i smanjuje mogućnost sistemskih i logičkih grešaka. Ipak, problem koji ostaje kod proceduralnih jezika može se uvideti preko suvišnosti funkcije "TekuciRacun_proveri()". Uzmimo za sledeći primer kod koji kao modul koristi prethodno prikazano rešenje:

```
// Modul 'PodizanjeNovca';
include 'TekuciRacun.module';

function PodizanjeNovca_kes(iznos) {
   trenutno_stanje = TekuciRacun_proveri();
   if (iznos < tekuce_stanje) {
      TekuciRacun_oduzmi(iznos);
      return true;
   }
   else {
      print("Na racunu nema dovoljno novca!");
      return false;
   }
}
```

Ovaj kod ilustruje mogućnost iskorišćenja gotovih proceduralnih rešenja u kompleksnijem softveru. Pogledajmo i varijantu optimizovanu po pitanju performansi:

```
// Modul 'PodizanjeNovca';
include 'TekuciRacun.module';

function PodizanjeNovca_kes(iznos) {
   global TekuciRacun_stanje;
   if (iznos < TekuciRacun_stanje) {
      TekuciRacun_oduzmi(iznos);
      return true;
   }
```

```
   else {
      print("Na racunu nema dovoljno novca!");
      return false;
   }
}
```

Na ovaj način eliminišemo nekoliko procesorskih ciklusa direktno pristupajući promenljivoj umesto preko funkcije. Rezultati su:

1. Softver koji se brže izvršava.
2. Mogućnost grešaka u budućnosti (npr. ukoliko se u funkciju "TekuciRacun_proveri()" vremenom doda još neka instrukcija.
3. Nagoveštaj nemogućnosti kontrole pristupa promenljivama u proceduralnim programskim jezicima.

Analizirajmo svaki od ova tri rezultata:

- Softver koji se brže izvršava.
  Jedan od glavnih atributa proceduralnih jezika jeste optimalnije korišćenje centralnog procesora. Ipak, mogućnosti i cena današnjih procesora najčešće daju prednost rešenjima sa manjom mogućnošću greške (koja zahteva ljudsku intervenciju i dodatne troškove) bez obzira na nekoliko (ili nekoliko desetina) procenata veće vreme potrebno za njihovo izvršavanje.
- Mogućnost grešaka u budućnosti (npr. ukoliko se u funkciju "TekuciRacun_proveri()" vremenom doda još neka instrukcija.
  Zarad prethodnog rezultata stvorena je mogućnost netačnog rezultata rada softvera u budućnosti ukoliko dođe do izmena unutar pomenute funkcije. Za izbegavanje greške potrebno je nakon izmene modula "TekuciRacun" izmeniti sav softver koji koristi taj modul šo najčešće nije moguće ili zahteva značajne resurse. Ipak, autor modula "PodizanjeNovca" se ne može kriviti za mogućnost greške jer je njegova odluka za direktan pristup promenljivoj jedna od opcija interfejsa ka modulu "TekuciRacun".
- Nagoveštaj nemogućnostikontrole pristupa promenljivama u proceduralnim programskim jezicima.
  Jasno je da se u nedostatku boljeg mehanizma (kod proceduralnih jezika) za zaštitu privatnih resursa softverskog modula autor mora osloniti na pažljivost korisnika koji će u svojim rešenjima koristiti taj modul.

Ranije pomenuto dodavanje prefiksa imenima korisnički definisanim članicama softverskog modula, kao i poslednja dva problema (neprecizno definisani interfejs i kontrola pristupa promenljivama) rešavaju objektno-orijentisani jezici. Modul "TekuciRacun" napisan u OO programskom jeziku bi mogao da izgleda ovako:

```
class TekuciRacun {
   protected stanje;
   public function dodaj(iznos) {
      this.stanje += iznos;
   }
```

```
   public function oduzmi(iznos) {
      this.stanje -= iznos;
   }
   public function proveri() {
      return this.stanje;
   }
}
```

a korišćenje klase “TekuciRacun” u modulu “PodizanjeNovca” bi moglo da izgleda ovako:

```
// Modul 'PodizanjeNovca';
include 'TekuciRacun.class';
function PodizanjeNovca_kes(iznos) {
   TekuciRacun = new TekuciRacun();
   if (iznos < TekuciRacun.proveri()) {
      TekuciRacun.oduzmi(iznos);
      return true;
   }
   else {
      print(“Na racunu nema dovoljno novca!”);
      return false;
   }
}
```

Ukoliko bi smo ponovo pokušali da pristupimo promenljivoj “stanje” direktno:

```
// Modul 'PodizanjeNovca';
include 'TekuciRacun.class';
TekuciRacun = new TekuciRacun();
function PodizanjeNovca_kes(iznos) {
   global TekuciRacun;
   if (iznos < TekuciRacun.stanje) {
      TekuciRacun.oduzmi(iznos);
      return true;
   }
   else {
      print(“Na racunu nema dovoljno novca!”);
      return false;
   }
}
```

sistem ne bi dozvolio direktan pristup (jer je osobina “stanje” definisana kao zaštićena) i objavio bi grešku sa porukom tipa “Cannot access protected property”. Ova važna osobina OO jezika se naziva “ugovaranje interfejsa” i može odigrati ogromnu ulogu kod kompleksnih sistema čije su komponente sklone asinhronom razvoju.

Takođe, OO pristup, osim što rešava prethodno identifikovane probleme proceduralnih jezika, nudi i mogućnost jednostavnog rada sa više objekata istog tipa (što u proceduralnim jezicima zahteva dodatno kodiranje uz već opisane probleme). Na primer, ukoliko imamo potrebe za istovremenim radom sa više različitih tekućih računa, veoma lako možemo napraviti više autonomnih instanci klase "TekuciRacun":

```
// Modul 'XYZ';
include 'TekuciRacun.class';
TekuciRacun1 = new TekuciRacun();
TekuciRacun2 = new TekuciRacun();
// Prebacivanje novca sa racuna na racun
TekuciRacun1.dodaj(TekuciRacun2.stanje);
TekuciRacun2.oduzmi(TekuciRacun2.stanje);
...
```

Cilj prethodno navedenih primera je da ukažu na probleme proceduralnih jezika rešene objektno-orijentisanim pristupom čiji su glavni principi primenjeni na servisno-orijentisani pristup.

3.1.4.2. Objektno-orijentisani i servisno-orijentisani pristup

Jedan od glavnih doprinosa objektno-orijentisanog pristupa je nesumnjivo jednostavnost korišćenja gotovih komponenti u razvoju novog rešenja. Međutim, izraz "gotova komponenta" otvara tri nova problema koja izlaze van okvira OO mogućnosti:

1. Potrebno je doći do gotove komponente tj. obezbediti je za korišćenje novom softverskom proizvodu.
2. Postavlja se pitanje da li je gotova komponenta zaista gotova (završena).
3. Postavlja se pitanje da li uslovi okruženja novog softverskog rešenja odgovaraju svim iskorišćenim gotovim komponentama.

Analizirajmo ove probleme:

1. Potrebno je doći do gotove komponente tj. obezbediti je za korišćenje novom softverskom proizvodu.
   Rešenje ovog problema se najčešće ogleda u preuzimanju svih datodeka gotove komponente i njihovom skladištenju na lokalnom fajl-sistemu novog rešenja. Iako ovakav pristup deluje kao potpuno prihvatljivo rešenje, njegovi nedostaci biće razmatrani u naredna dva problema.
2. Postavlja se pitanje da li je gotova komponenta zaista gotova (završena).
   Uzmimo za primer autora A koji radi na softverskom rešenju X i želi da iskoristi gotovu komponentu Y autora B. Inicijalno, autor A može preuzeti poslednju verziju komponente Y i uključiti je u rešenje X. Pretpostavimo, međutim, da autor B naknadno uoči propust u komponenti Y čija ispravka komponentu dovodi u sledeću verziju. Ova izmena zahteva od autora A ponovno preuzimanje komponente Y (tj. njene poslednje verzije) i njenu zamenu u rešenju X (što dovodi i do nove verzije rešenja X). Osim što pomenuti proces može biti zahtevan po pitanju vremena i resursa treba imati u vidu da rešenje X može imati i druge komponente kao i da ono može biti uključeno u rešenje Z u vidu gotove komponente.
3. Postavlja se pitanje da li uslovi okruženja novog softverskog rešenja odgovaraju svim iskorišćenim gotovim komponentama.

Kompleksna rešenja mogu sadržati veći broj gotovih komponenti različitih autora, različitih pretpostavki, rađenih za različite sisteme i različita okruženja. U 1. problemu se navodi najčešći zahtev korišćenja svih gotovih komponenti u istom okruženju. Jasno je da se u takvoj situaciji lako može dogoditi da uslovi za rad nekih komponenti nisu ispunjeni (npr. različiti OS, programski jezik i sl.) a njihovim ispunjenjem se ugrožavaju uslovi neophodni za rad ostalih komponenti. Dodatni problem može biti i ograničenje hardverskih resursa jednog računarskog sistema pa se za rad svih komponenti mora primeniti dodatno rešenje koje dodatno komplikuje sistem (npr. klasteri računara i sl.).

Uzmimo za primer prethodno navedene probleme korišćenja gotovih komponenti i eliminišimo ih izostavljajući realizaciju kroz model višeg nivoa apstrakcije. Atomska vrednost takvog modela bi bila Objekat (bez detalja gde je skladištena definicija njegove klase, koja je verzija u pitanju i sl.) sa svojim Metodama, Svojstvima i Atributima:

ObjekatX.MetodY(Atribut1, Atribut2, Atribut3)

Postoje alati za razvoj softvera koji omogućavaju prevođenje ovakvog modela na gotovo rešenje (npr. UML) ali njima nisu rešeni osnovni prethodno opisani problemi - problemi korišćenja spoljnih, gotovih komponenti. Rešenja ovih problema nudi servisno-orijentisana arhitektura putem istih principa kojima je objektno-orijentisani pristup rešio glavne probleme proceduralnog programiranja - principa povećanja autonomije i apstrakcije.

Servisno-orijentisana arhitektura omogućava korišćenje gotovih komponenti na njihovim matičnim lokacijama uz zamenu interne računarske magistrale računarskom mrežom. Ukoliko u prethodno pomenutom apstraktnom modelu zamenimo reč Objekat rečju Provjader a reč Metod rečju Servis, model bi za svaku od iskorišćenih gotovih komponenti bio dopunjen adresom provajdera koji njene javne metode nudi kao servise. Ovakav pristup nudi sledeće prednosti:

- Pristup rešava tri pomenuta problema vezana za korišćenje gotovih komponenti kod OO pristupa.
- Sve gotove komponente iskorišćene kao servisi funkcionišu u svom matičnom okruženju.
- Servisi mogu imati pristup podacima kojima pristup ne bi bio moguć bez potpune kontrole nad ponašanjem komponente.
- Moguće je model novog softverskog rešenja automatski prevesti u radni oblik.
- Moguće je znatno brža izgradnja novog softverskog rešenja.
- ...

Osnovni nedostaci SOA pristupa su:

- Performanse.
- Zavisnost od komunikacionog kanala (sem ukoliko se svi servisi i rešenje koje ih koristi ne nalaze na istom računaru).
- Zavisnost od provajdera servisa.
- Nedovoljna razvijenost alata za rad sa SOA.

3.1.4.3. Zaključno o SOA

Postoje, naravno, situacije u kojima SOA nije primenljiva ili nudi skromnije rezultate od proceduralnih ili OO rešenja. To su najčešće situacije u kojima se zahtevaju visoke preformanse,

garantovana dostupnost svih komponenti i/ili potpuna kontrola nad svim komponentama sistema.

Mogućnosti i prednosti koje donosi servisno-orijentisana arhitektura su neosporne. Savremeno, dinamično i distribuirano infomaciono okruženje stvara nove potrebe na koje samo servisno-orijentisani pristup može da da odgovore. Trenutno najčešće korišćeni oblik SOA su web servisi (eng. web services) a pojavljuju se sve kvalitetniji alati za razvoj u ovoj i drugim primenama. Sličnosti sa OO pristupom omogućavaju laku migraciju (PHP programski jezik omogućava objavljivanje postojeći OO klasa u vidu web servisa sa samo nekoliko linija koda) i nije nerealno očekivati u bližoj budućnosti da se rešenja bazirana na SOA nađu u oblastima informacionih tehnologija u kojima trenutno nema nikakvog nagoveštaja za mogućnošću distribuirane obrade podataka.

3.2. Analiza tehnologija kompatibilnih sa zahtevima

U skladu sa odabranom osnovnom troslojnom klijent-server arhitekturom potrebno je odabrati adekvatne tehnologije za realizaciju komponenti svakog od slojeva. Troslojna klijent-server arhitektura podrazumeva sledeće slojeve:

1. sloj podataka
2. sloj aplikacije
3. sloj prezentacije

što zahteva sledeće alate/tehnologije:

1. tehnologije vezane za izgradnju baze podataka:
   1.1. sistemi za upravljanje bazama podataka (SUBP)
   1.2. alati za modelovanje baze podataka
2. tehnologije vezane za izgradnju logičkih jedinica softvera
3. tehnologije vezane za izgradnju interfejsa:
   3.1. tehnologije vezane za izgradnju korisničkog interfejsa
   3.2. tehnologije vezane za izgradnju interfejsa ka drugim IS

U narednom delu rada biće razmatrane navedene tehnologije i kriterijumi na osnovu kojih je vršen izbor tehnologija primenjenih u izgradnji informacionog sistema univerziteta.

3.2.1. Tehnologije vezane za izgradnju baze podataka

Dve ključne komponente u izgradnji baze podataka informacionog sistema jesu:

1. sistem za upravljanje bazama podataka (SUBP)
2. alati za izradu modela baze podataka (i prevođenje u SUBP)

Sistemi za upravljanje bazama podataka (eng. DataBase Management System, DBMS) predstavljaju aktivne komponente čije korišćenje traje tokom celog životnog ciklusa informacionog sistema. Ovi sistemi predstavljaju veoma značajnu komponentu sistema jer se sav pristup podacima (čitanje, dodavanje, izmena, uklanjanje) vrši posredstvom njih.

Alati za izradu modela baze podataka predstavljaju opcionu komponentu (imajući u vidu da je izrada modela moguća i putem ručnog unošenja komandi DDL-a) ali je njihovo korišćenje poželjno jer smanjuje mogućnost greške i vreme potrebno za izradu modela. Ova komponenta se koristi u fazi razvoja informacionog sistema i povremeno za potrebe izmene strukture baze podataka.

3.2.1.1. Sistemi za upravljanje bazama podataka (SUBP)

Sistemi za upravljanje bazama podataka (DataBase Management System, DMBS) su kompleksni softverski sistemi čija je osnovna uloga rad sa podacima u pridruženoj bazi podataka. SUBP sistemi su zaduženi za organizaciju, skladištenje i pristup podacima u bazi. U tu svrhu se koriste adekvatni jezici a najčešće SQL (Structured Query Language) koji je podeljen na:

1. DDL (Data Definition Language) – skup komandi namenjen za definisanje organizacije baze pdoataka.
2. DML (Data Manipulation Language) – skup komandi namenjen za pristup podacima (dodavanje, izmenu, pregled i uklanjanje).
3. DCL (Data Control Language) – skup komandi namenjen za kontrolu pristupa podacima (podešavanje korisničkih naloga i privilegija).

Sistemi za upravljanje bazama podataka mogu koristiti različitu organizaciju podataka (objektno-orijentisanu, hijerarhijsku...) ali je u upotrebi najčešći relacioni model koji je iskorišćen i za potrebe informacionog sistema. Pri odabiru sistema za upravljanje bazama podataka korišćeni su sledeći kriterijumi:

- jezik za upite (query language)
  - da li je u pitanju SQL i koja verzija
  - dodatne funkcije jezika (matematičke funkcije i sl.)
- importovanje i eksportovanje podataka
  - podržani formati
  - pravljenje zaštitnih kopija
- tipovi podataka
  - podržani tipovi podataka (celobrojni, decimalni brojevi, stringovi...)
  - podrška za standardnu IEEE reprezentaciju decimalnih brojeva
  - podrška za tipove vezane za datume i vreme
  - podrška za binarne podatke
  - podrška za nizove i strukture
- upravljanje nultim (praznim) vrednostima
- performanse
  - minimalni resursi
  - stepen iskorišćenja resursa
  - poređenje sa drugim SUBP
- skalabilnost i ograničenja
  - ograničenja u broju zapisa po tabeli
  - ograničenja u broju kolona po tabeli
  - ograničenja u bajtovskoj veličini tabele

  - ograničenja u bajtovskoj veličini zapisa
  - ograničenja u bajtovskoj veličini polja
  - ograničenja u broju tabela u bazi podataka
  - ograničenja u broju baza podataka
  - ograničenja vezana za fajl sistem i mogućnost korišćenja sirovih particija
  - mogućnost kreiranja replika baza podataka
  - mogućnost kreiranja klastera
  - ograničenja u količini radne memorije
  - ograničenja u broju procesora
- indeksiranje
  - tipovi indeksiranja
- interfejsi i API
  - podržani tipovi eksternih interfejsa (JDBC, ODBC...)
- bezbednost
  - ugrađeni bezbednostni mehanizmi
  - kontrola pristupa
- upravljanje sistemom
- kompatibilnost
  - sa rešenjima drugih proizvođača
  - sa platformama
- cena

U skaldu sa ovim kriterijumima u uži izbor su ušla neka od najpopularnijih rešenja danas:

1. Oracle
2. IBM DB2
3. Microsoft SQL server
4. MySQL

Za potrebe informacionog sistema je odabran MySQL SUBP imajući u vidu cenu i potrebe u prvoj fazi. S obzirom na to da je pristup podacima u aplikativnom sloju rešen apstraktno (preko brokera) postoji mogućnost zamene SUBP-a u budućnosti ukoliko se sistem ispostavi neadekvatnim ili neadekvatan za novonastalu situaciju.

3.2.1.1.1. Oracle

Kompanija Oracle je jedan od vodećih proizvođača sistema za upravljanje bazama podataka (SUBP/DBMS) kao i ERP (Enterprise Resource Planning), CRM (Consumer Relationship management) i SCM (Supply Chain Management) softvera. Ova kompanija je osnovana 1977. godine, poseduje kancelarije u više od 145 zemalja sveta i zapošljava više od 50.000 ljudi.

Prvu verziju DBMS softvera kompanija Oracle je na tržište izbacija 1978. godine. Trenutno aktuelna varijanta ovog sistema za upravljanje bazama podataka jeste 10g (na tržištu se pojavila 2003. godine). Oracle DBMS se danas nudi u više različitih varijanti (edicija) sa ciljem da privuče različite kategorije korisnika:

- **Standard Edition (SE)**
  Oracle DMBS u ovoj varijanti sadrži osnovne funkcionalanosti a licenciranje se vrši po broju korisnika ili centralnih procesora. Standard Edition (SE) podržava do 4 centralna procesora s tim da ne postoje ograničenja vezana za količinu memorije i veličinu baze podataka kao i da podržava rad na klasteru računara.
- **Enterprise Edition (EE)**
  Oracle DBMS u ovoj varijanti ne poseduje ograničenja vezana za broj centralnih procesora koji će biti korišćeni. Poseduje sve funkcionalnosti kao i Standard Edition (SE) uz dodatak funkcionalnosti koje omogućavaju povećanje performansi i nivoa bezbednosti podataka.
- **Standard Edition One**
  Ova varijanta Oracle DBMS-a se pojavila sa verzijom 10g i može se najbliže porediti sa varijantom Standard Edition (SE) s tim da poseduje dodatna ograničenja vezana za funkcionalnosti i broj centralnih procesora (1-2 centralna procesora).
- **Express Edition (XE)**
  Ova varijanta Oracle DBMS-a se pojavila sa verzijom 10g i njena osobenost je mogućnost slobodnog (i besplatnog) distribuiranja na MS Windows i Linux platformama. Ograničenja ove varijante su korišćenje jednog centralnog procesora, do 1GB radne memorije i najveća veličina baze podataka od 4GB. Osim ovih ograničenja XE varijanta takođe ne podrazumeva nikakav vid tehničke podrške.
- **Personal Edition**
  Ova varijanta Oracle DBMS-a poseduje sve funkcionalnosti EE varijante sa tom razlikom što je namenjenima programerima za razvoj softvera i licenca ne dozvoljava korišćenje u praksi.
- **Lite Edition**
  Lite varijanta Oracle DBMS-a je namenjena za korišćenje na prenosnim računarima. Osnovna ideja jeste skladištenje određene grupe podataka na prenosnom računaru da bi se eliminisala potreba za stalnom vezom sa *back-end* sistemima.

Dugogodišnje prisustvo na tržištu i orijentacija proizvoda ka poslovnim sistemima pozicioniraju kompaniju Oracle i njene proizvode kao ultimativan izbor kada su u pitanju velike baze podataka sa intenzivnim korišćenjem. Međutim, relativno visoka cena (pre svega u poređenju sa besplatnim rešenjima) uslovila je nalaženje alternativnog rešenja uz ostavljanje otvorene mogućnosti kasnijeg prelaska na Oracle DBMS.

3.2.1.1.2. IBM DB2

Kompanija IBM (International Business Machines) je osnovana 1889. godine a njeno sedište je u državi New York u SAD. Ovaj gigant svetskih razmera ima predstavništva u preko 170 zemalja sveta i zapošljava preko 350.000 ljudi. Zaposleni IBM-a su tokom postojanja ove kompanije dobili 3 Nobelove nagrade, četiri Tjuringove nagrade, pet nacionalnih medalja za tehnologiju i pet nacionalnih medalja za nauku.

Osim razvoja softvera za upravljanje bazama podataka kompanija IBM je vodila (i vodi) mnoštvo različitih projekata vezanih za softver (operativni sistemi, razvojna okruženja...) i hardver. Trenutno aktuelni proizvod kompanije IBM vezan za baze podataka jeste IBM DB2.

IMB DB2 se smatra prvim sistemom za upravljanje bazama podataka koji je koristio SQL jezik. Ovaj proizvod se takođe bazira i na teoriji relacionih baza podataka koju je 1970. godine objavio Dr. Edgar F. Kod (takođe zaposlen u IBM-u). Dugi niz godina IBM DB2 sistem za upravljanje bazama podataka je bilo moguće koristiti isključivo na IBM-ovim mainframe računarima. Danas se ovaj softver može koristiti i na platformama kao što su Unix, MS Windows i Linux. Osim podrške za različite platforme IBM DB2 se nudi i u više različitih varijanti (edicija):

- Express Edition
- Workgroup Edition
- Enterprise Edition
- Warehouse Enterprise Edition
- Express-C Edition

Trenutna verzija DB2 sistema je 9 a u trenutku njenog pojavljivanja DB2 Express sistemi su naplaćivani po ceni od 4.874 dolara po procesoru ili 165 dolara po korisniku (uz minimum od 5 korisnika) na ostalim platformamam i 4.450 dolara mesečno na z/OS platformi. Izuzetak od ove cene jeste Express-C varijanta koja se pojavila januara 2006. godine kao odgovor na besplatne varijante Oracle i Microsoft DBMS sistema.

Express-C varijanta podrazumeva korišćenje bez plaćanja uz ograničenje na 2 centralna procesora, do 4GB radne memorije a podržane platforme su MS Windows i Linux. U poređenju sa Express varijantom iz varijante Express-C su izostavljene sledeće funkcionalnosti:

- Replication
- High Availability
- Spatial Extender Client and Samples
- Microsoft Cluster Server support
- Informix Data Source support
- DB2 Web tools
- GSKit
- APPC and Netbios

3.2.1.1.3. Microsoft SQL server

Kompanija Microsoft je osnovana 1975. godine sa sedištem u Albukerkiju u SAD a danas se sedište nalazi u Redmondu, SAD. Kompanija ima predstavništva u 102 zemlje i zapošljava oko 76.000 ljudi. Osim razvoja softvera za upravljanje bazama podataka kompanija se bavi i proizvodnjom operativnih sistema, kancelarijskih i razvojnih alata, poslovnih rešenja, zabavnog softvera i razvojem hardvera.

Microsoft SQL Server je sistem za upravljanje bazama podataka razvijen od strane kompanije Microsoft. Ovaj sistem je razvijen na osnovu Sybase SQL Server rešenja a koristi Tranasact-SQL implementaciju SQL-92 standarda. Trenutna verzija Microsoft SQL Servera je 2005 ili 9.0. Postoji više varijanti (edicija) Microsoft SQL Servera 2005:

- **Enterprise Edition**
  Varijanta koja poseduje pun skup funkcionalnosti i ne postavlja dodatna ograničenja po pitanju podršće za hardver.
- **Developer Edition**
  Ova varijanta sadrži sve funkcionalnosti kao i Enterprise varijanta sa tom razlikom što licenca ograničava upotrebu samo za razvoj i demonstraciju.
- **Standard Edition**
  Podržano centralnih procesora: 4
  Podržano radne memorije: bez ograničenja
  Podržana veličina baze podataka: bez ograničenja
- **Workgroup Edition**
  Podržano centralnih procesora: 2
  Podržano radne memorije: 3GB
- **Express Edition**
  Podržano centralnih procesora: 1
  Podržano radne memorije: 1GB
  Podržana veličina baze podataka: 4GB
- **Compact Edition**
  Ova varijanta Microsoft SQL Servera je namenjena upotrebi na lokalnim računarima.

Takođe, postoje i dodatna rešenja koja su u sprezi sa ovim softverom:

- SQL Server Integration Services
- SQL Server Analysis Services
- SQL Server Reporting Services
- SQL Server Notification Services
- SQL Server Management Studio

Glavna prednost Microsoft SQL Servera jeste laka integracija sa ostalim proizvodima kompanije Microsoft i, obrnuto, mana ovog sistema jeste slaba podrška ostalih proizvođača. Izuzetak od ovog pravila čini funkcionalnost koja omogućava pristup SUBP-u putem SOAP protokola. S obzirom na izbor ostalih tehnologija koje će biti korišćene u izgradnji informacionog sistema, Microsoft SQL Server je izuzet kao mogući izbor za centralnu bazu podataka.

3.2.1.1.4. MySQL

Kompanija MySQL AB je osnovana 1995. a njena sedišta trenutno se nalazre u Upsali (Švedska) i Kaliforniji (SAD). Kompanija ima predstavništva u oko 25 zemalja sveta i zapošljava oko 360 ljudi. Model licenciranja koji koristi ova kompanija za svoje proizvode je „dual licencing“ što ih čini besplatno dostupnim (bez tehničke podrške) i komercijalno dostupnim. Najreprezentativni proizvod kompanije jeste MySQL sistem za upravljanje bazama podataka koji je, prema podacima proizvođača, instaliran na preko 10 miliona računara.

MySQL SUBP je jedan od najpopularnijih sistem na Linux platformi i čini jednu od osnovnih komponenti za razvoj Web aplikacija na ovom sistemu. Izraz LAMP (Linux, Apache, MySQL, PHP) se često koristi kao akronim za ovu razvojnu platformu. MySQL SUBP ne poseduje sve funkcionalnosti kao sistemi tipa Oracle ali dovoljan skup funkcionalnosti i povoljna cena su pozitivno uticali na prihvatanje ovog softvera od strane tržišta. Danas većina programskih jezika poseduje biblioteke za korišćenje MySQL-a a postoji i ODBC interfejs za jezike u koje nije ugrađena podrška. Osim na Linux platformi ovaj sistem ima podršku i za AIX, BSD sisteme, Mac OS X, Novell Netware, OS/2, SGI IRIX, Solaris, SunOS, SCO sisteme i Misrosoftove operativne siteme.

Glavne sistemske karakteristike MySQL-a su podrška za više procesora i višekorisnički rad i izvorni kod napisan u C i C++ programskim jezicima a koristi i yacc i lexer parsere. Licence pod kojima se ovaj sistem može koristiti su:

- GPL (General Public Licence)
- MySQL Enterprise subsctiption

Osim razlike u licenci MySQL se do verzije 5 distribuirao u okviru tri kompilacije (Standard, Max i MySQL Debug) ali se od verzije 5 (u zavisnosti od licence) distribuira kao:

- MySQL Community Server
- MySQL Enterprise Server

Jedine razlike ove dve distrubucije se nalaze u licenci i načinu tehničke podrške a dele zajedničku osnovu koja podržava:

- veći deo ANSI SQL 99 standarda (uz dodatna proširenja)
- skladištenje procedura
- okidače (triggers), kursore (cursors) i izmenljive poglede (views)
- transakcije
- podršku za SSL
- keširanje upita
- ugnježdavanje upita
- replikaciju
- UTF8 podršku
- podršku za ACID (Atomicity, Consistency, Isolation, Durability)

- ...

Takođe, od verzije 5.1 MySQL sistemi [15] podržavaju i:

- particionisanje baza
- online bekap
- fail-safe replikaciju i replikaciju klastera
- ograničenja na nivou kolona
- zakazivanje akcija (eng. event scheduling)
- XML funkcije
- load emulator (podrška za simulaciju upotrebe baze sa izvešajima o performansama)

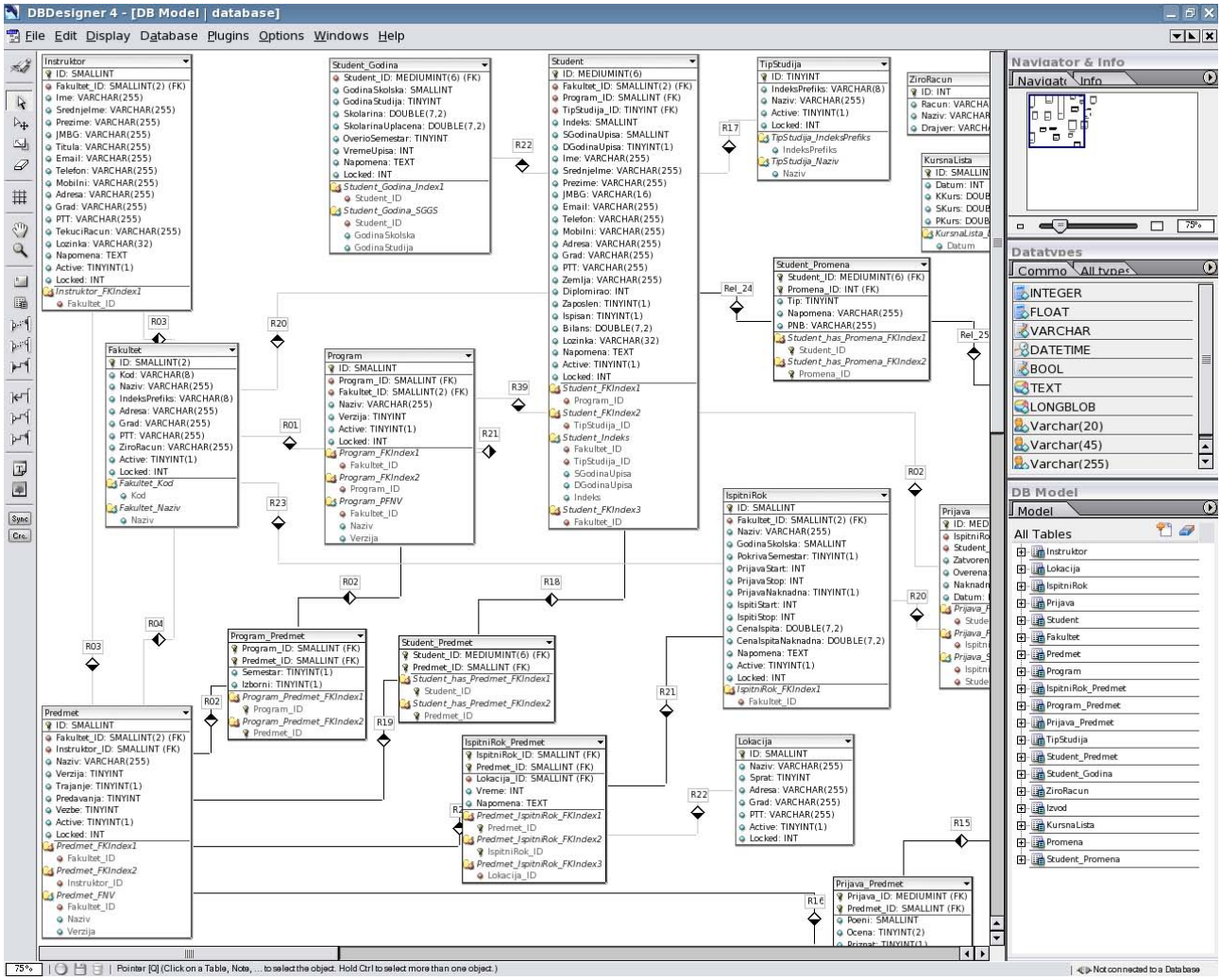
MySQL sistem za upravljanje bazama podataka je primenjen u mnoštvu različitih informacionih sistema od kojih su najpoznatiji:

- Wikipedija (on-line enciklopedija sa više stotina miliona pregleda i par miliona izmena dokumenata u toku dana)
- CNET Networks
- Nokia (klaster za praćenje informacija o korisnicima mobilnih usluga u realnom vremenu)
- Flickr (upravljanje milionima fotografija i korisničkih naloga)
- YouTube
- ...

Neki od glavnih nedostataka MySQL sistema tokom razvoja su bili nedostatak relacionih funkcionalnosti, podrške za ACID i tretiranje NULL vrednosti. Međutim, počev od verzije 5 MySQL predstavlja ozbiljan sistem za upravljanje bazama podataka u kome su rešeni pomenuti nedostaci. Takođe, mogućnost besplatnog korišćenja sofvera u okviru GPL licence je dodatni razlog zašto je ovaj sistem odabran kao primarni sistem za upravljanje bazom podataka informacionog sistema univerziteta.

3.2.1.2. Alati za modelovanje baza podataka

Alati za modelovanje podataka predstavljaju jednu od bitnih komponenti u razvoju kako velikih informacionih sistema tako i svih softverskih proizvoda koji koriste relacione baze podataka putem sistema za upravljanje bazama podataka. Uloga ovih alata jeste da omoguće brz i jednostavan razvoj modela baze podataka koji je kasnije lako razumeti i menjati u skladu sa potrebama. Bitna karakteristika ovih alata jeste podrška za određene DBMS sisteme i njihove specifičnosti.



Slika 9 – DB Designer 4 alat za modelovanje baze podataka

U razvoju modela baze podataka informacionog sistema univerziteta korišćen je alat DB Designer verzije 4. Ovaj alat ne poseduje mnoge funkcionalnosti profesionalnih alata (kakve npr. nudi Sybase) ali omogućava brz i efikasan razvoj modela primenljivog na MySQL DBMS. Alat nudi grafički prikaz i razvoj modela a čuvanje se vrši u XML formatu. Alat takođe nudi eksport modela u vidu SQL naredbi za kreiranje baze kao i direktnu sinhronizaciju sa samim serverom.

3.2.2. Tehnologije vezane za izgradnju logičkih jedinica softvera

Jedna od nosećih komponenti informacionog sistema, pored baze podataka, jeste i skup logičkih jedinica softvera tj. softverskih modula putem kojih se podržava logika poslovnih procesa. Ovi moduli se mogu razvijati na različite načine s tim da su dva osnovna:

1. razvoj putem ručnog kodiranja softvera u odgovarajućem programskom jeziku i
2. razvoj putem razvoja modela u nekom od UML (Unified Modeling Language) alata uz naknadno prevođenje rešenja u neki od programskih jezika.

Razvoj putem UML alata predstavlja brži i efikasniji način čiji je rezultat (pored koda u odabranom programskom jeziku) i pregledan model logike i relacija podržanih poslovnih procesa. Ovakav način razvoja, međutim, nije pogodan kod razvoja softverskih sistema koji se baziraju i na spoljnim *framework* okvirima tj. okvirima koje zadaju komponente nevezane za korišćeni UML alat.

Zbog pomenutog razloga razvoj informacionog sistema nije baziran na korišćenju UML alata već je umesto toga odabrano ručno kodiranje uz korišćeje adekvatnog *framework* sistema koji je svojim gotovim funkcionalnostima nadoknadio nedostatke (pre svega vezane za vreme potrebno za razvoj) takvog načina razvoja. Iako korišćeni *framework* zahteva rad na PHP programskom jeziku (makar kod centralnih delova sistema) analizirane su tri popularne tehnologije:

- Microsof .Net
- Java
- PHP

Prednost JSP-a i PHP naspram ASP tehnologije je platformska nezavisnost. Java programski jezik je poznat po svom revolucionarnom konceptu vrituelne mašine (koja omogućava izvršavanje istog programa na različitim platformama) dok PHP predstavlja interpretirani programski jezik čiji je interpreter dostupan za različite platforme. Nasuprot tome, ASP tehnologija je usko vezana za Microsoft platformu što predstavlja značajno ograničenje.

U razvoju informacionog sistema je korišćen PHP programski jezik zbog svojih sve naprednijih mogućnosti i pomenutog *framework*-a. Dodatna prednost PHP-a jeste i mogućnost direktnog korišćenja gotovih komponenti izrađenih u Java programskom jeziku.

Dodatni kriterijum koji se koristio pri odabiru programskog jezika jeste i kompatibilnost sa odabranom softverskom arhitekturom (klijent-server/SOA). U skladu sa tim kriterijumom odlični programski jezici (npr. C, C++, itd.) za određene primene (npr. sistemsko programiranje) su ocenjeni kao neadekvatni.

3.2.2.1. Microsoft .Net

Active Server Pages (ASP) predstavlja tehnologiju kompanije Microsoft i dostupna je samo u okviru IIS (Internet Information Server) Web servera (takođe proizvod kompanije Microsoft). Poslednje verzije ASP podrške omogućavaju korišćenje .NET tehnologije. Većina ASP stranica je napisana u VBScript programskom jeziku ali je moguće koristiti i bilo koji drugi ActiveScripting jezik.

Jedno od najvažnijih ograničenja vezanih ne samo za .Net već za većinu tehnologija i rešenja koja potiču iz kompanije Majkrosoft jeste vezanost za Majkrosoft platformu (MS Windows operativni sistemi, MS Internet Information Server, MS SQL server...) i nedostupnost izvornog koda.

Zvanični resursi vezani za ASP.Net 2.0 tehnologiju se nalaze na adresi: http://www.asp.net

3.2.2.2. Java

Java Server Pages (JSP) tehnologija se bazira na Javi. Odnosi se na ugnježdavanje java koda u HTML dokumente preko posebnih JSP tagova. Ovaj kod se izvršava na serverskoj strani dinamički neposredno pre isporuke stranice klijentu. U novije vreme razvijeno su čitave biblioteke JSP tagova koje omogućavaju veću produktivnost u izradi Web aplikacija.

JSP predstavlja nastavak servlet tehnologije i njeno je proširenje. Prednost JSP-a je što je to u potpunosti objektno orijentisana tehnologija koja omogućava korišćenje svih funkcionalnosti Java jezika. Za korišćenje JSP-a koriste se aplikacioni serveri od kojih su najpoznatiji Tomcat, Sun Application Server, JBoss itd.

Zvanični resursi vezani za Java tehnologiju se nalaze na adresi: http://java.sun.com

Zvanični resursi vezani za JSP tehnologiju se nalaze na adresi: http://java.sun.com/products/jsp

3.2.2.3. PHP

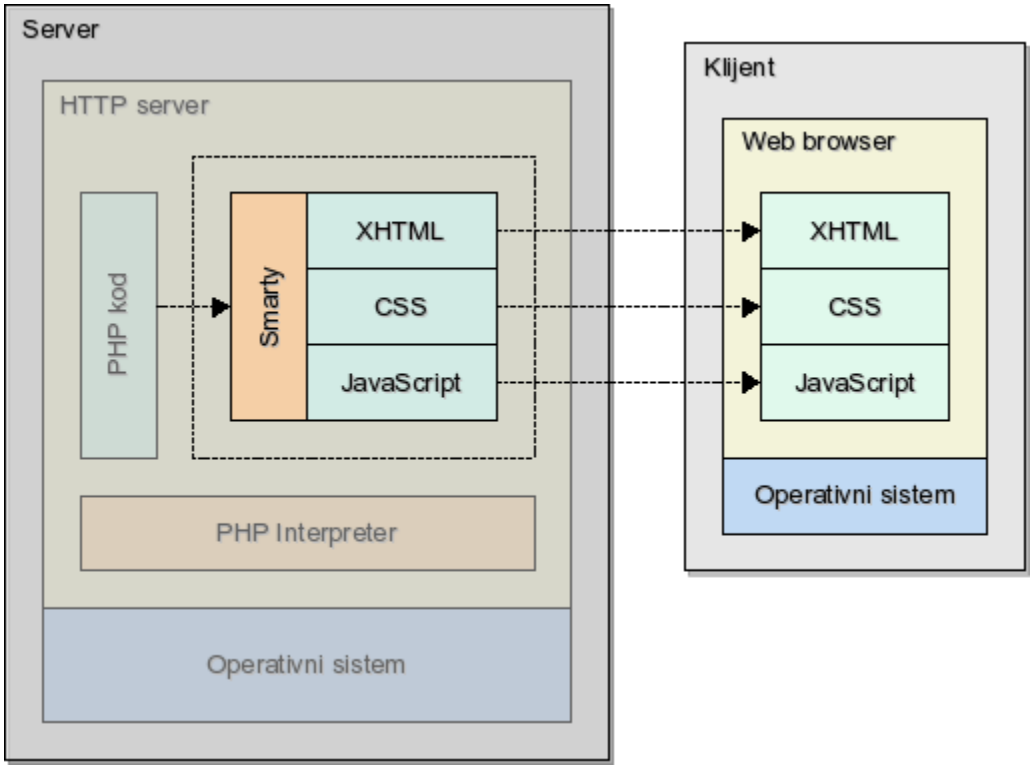
PHP Hypertext Preprocessor (PHP) je programski jezik orijentisan ka Web-u ali se danas, usled svoje popularnosti, može sresti i u grafičkim desktop aplikacijama, Unix skriptovima i sl. Prva verzija ovog jezika se pojavila 1994. godine a trenutno aktuelna verzija 5 datira iz 2005. godine. PHP se može koristiti na različitim platformama ali se najčešće sreće kod Apache Web servera na Linux operativnom sistemu. Korišćenje PHP-a je slobodno s obzirom na to da je njegov izvorni kod javno dostupan a njegova licenca ne podrazumeva plaćanje korišćenja [5].

Zvanični resuri vezani za PHP se nalaze na adresi: http://www.php.net

3.2.3. Tehnologije vezane za izgradnju korisničkog interfejsa

Prezentacioni sloj troslojne klijent-server arhitekture predstavlja interfejs sistema ka korisnicima. Iako programeri najčešće ovaj sloj smatraju najmanje bitnim i fokusiraju se na logiku aplikativnog sloja, prezentacioni sloj može predstavljati ključni faktor kada je u pitanju pravilno korišćenje sistema ili, čak, prihvatanje/odbijanje korišćenja od strane korisnika.

SOA arhitektura omogućava realizaciju korisničkog interfejsa na različitim tehnologijama (u vidu .Net ili Java aplikacija itd.) ali je realizacija korisničkog interfejsa informacionog sistema bazirana na Web klijentima (pre svega Mozilla Firefox) jer njihovo korišćenje omogućava korišćenje ugrađenih funkcionalnosti i protokola a i eliminiše potrebu za dodatnom instalacijom.



Slika 10 – Tehnologije orijentisane ka korisničkom interfejsu Web aplikacija

Osnovne tehnologije koje se koriste za izgradnju korisničkog interfejsa u vidu Web aplikacija jesu:

- XHTML
- CSS
- JavaScript

Takođe, iako predstavlja tehnologiju koja se koristi na serverskoj strani i predstavlja rešenje bazirano na PHP-u, u tehnologije korišćene za izgradnju korisničkog interfeja može se uključiti i:

- Smarty

s obzirom na to da ovo rešenje služi kao most između serverske i klijentske strane na taj način što koristi interne podatke PHP-a i na osnovu njih generiše korisnički interfejs.

Prednosti ranije pomenute SOA arhitekture su dostupne i kod interfejsa u vidu Web aplikacije pre svega u vidu Ajax poziva, SOAP protokola i XML-a.

3.2.3.1. (X)HTML

HTML (Hyper Text Markup Language) je jezik iz porodice jezika za označavanje (Markup Language) [7]. Jezici za označavanje se ponekad svrstavaju u programske jezike što je pogrešno. Uloga jezika za označavanje je da označe delove dokumenta. Na primer:

```
Ovo je <u>jezik za označavanje</u>.
```

predstavlja deo HTML dokumenta i u njemu je označeno da se reči uokvirene <u> i </u> oznakama (eng. tag) prikažu podvučene. Za sam prikaz je zadužen klijent (Web čitač). HTML dokumenti nisu predviđeni da sadrže binarne podatke (mada je moguće zaobilaženje ovog ograničenja) ali mogu imati reference prema binarnim resursima (npr. slici).

Klasičan HTML dokument se sastoji od dve celine: zaglavlja i tela dokumenta. Podaci u zaglavlju su namenjeni Web čitaču i sadrže informacije o naslovu dokumenta, ključnim rečima, datumu isticanja i sl. Telo dokumenta je deo koji je namenjen za prezentovanje (koje može biti prikaz na ekranu ali i zvučni izlaz) korisnicima.

```
<html>
   <head>
      <title>Naslov Dokumenta</title>
   </head>
   <body>
      Ovo je 1. red.<br />
      Ovo je 2. red.<br />
      Ovo je <a href="http://www.singidunum.ac.yu">link</a>.
   </body>
</html>
```

Uvođenje CSS tehnologije je omogućilo prečišćavanje HTML-a tj. izbacivanje oznaka (tagova) i atributa koji se odnose na definisanje prikaza a ne na strukturu i sadržaj. Rezultat ove izmene je XHTML (Extensible HyperText Markup Language) format čija je specifikacija znatno restriktivnija.

Pored XHTML-a, XML (Extensible Markup Language) predstavlja još jedan od Markup jezika koji se koriste u okviru Web servisa (s tim da primena XML-a nije ograničena samo na Web). XML je jezik orijentisan ka strukturi dokumenta i ne podrazumeva bilo kakvu definiciju prikaza. Ova osobina čini XML jezikom čiji su dokumenti pre ulaz računarskog programa nego dokumenti koji će interpretirati Web klijent (Web browser). XML dokumente je moguće prikazati direktno u Web klijentima preko dva tipa transformacija:

1. definisanja prikaza elemenata putem pomenutih CSS definicija
2. prevođenja dokumenata u (X)HTML putem XSLT transformacija

XSLT (Extensible Stylesheet Language Transformations) je jezik baziran na XML-u i koristi se za prevođenje XML podataka u neki drugi format. Uz korišćenje XSLT-a, XML dokumenti se mogu prevesti u HTML format na svakom Web klijentu koji ima XSLT procesor.

Kao podrazumevanog mark-up jezika W3C organizacije, Web prezentacija XHTML-a se nalazi na adresi: http://www.w3.org/MarkUp/

3.2.3.2. CSS

Razvojem HTML-a, a pre svega tehnologija za kreiranje dinamičkih dokumenata, iskristalisala se potreba za razdvajanjem strukture i sadržaja od prezentacije (prikaza). Ova potreba se dodatno povećala uvođenjem tehnologija koje omogućavaju osobama sa oštećenim čulom vida da koriste računare preko audio interfejsa.

Glavno unapređenje se javilo u vidu CSS-a (Cascading Style Sheets), tehnologije koja omogućava nezavisno definisanje prikaza elemenata HTML-a.

```
p {
   margin: 10px;
   border-width: 3px;
   border-style: double;
   border-color: #FF0000;
   padding: 20px;
   background-color: #0000FF;
   text-align: justify;
   text-transform: uppercase;
   font-family: Times New Roman;
   font-size: 11pt;
   font-weight: bold;
}
```

Uvođenje CSS tehnologije je omogućilo prečišćavanje HTML-a tj. izbacivanje oznaka (tagova) i atributa koji se odnose na definisanje prikaza a ne na strukturu i sadržaj. Rezultat ove izmene je XHTML (Extensible HyperText Markup Language) format čija je specifikacija znatno restriktivnija.

Pored XHTML-a, XML (Extensible Markup Language) predstavlja još jedan od Markup jezika koji se koriste u okviru Web servisa (s tim da primena XML-a nije ograničena samo na Web). XML je jezik orijentisan ka strukturi dokumenta i ne podrazumeva bilo kakvu definiciju prikaza. Ova osobina čini XML jezikom čiji su dokumenti pre ulaz računarskog programa nego dokumenti koji će interpretirati Web klijent (Web browser). XML dokumente je moguće prikazati direktno u Web klijentima preko dva tipa transformacija:

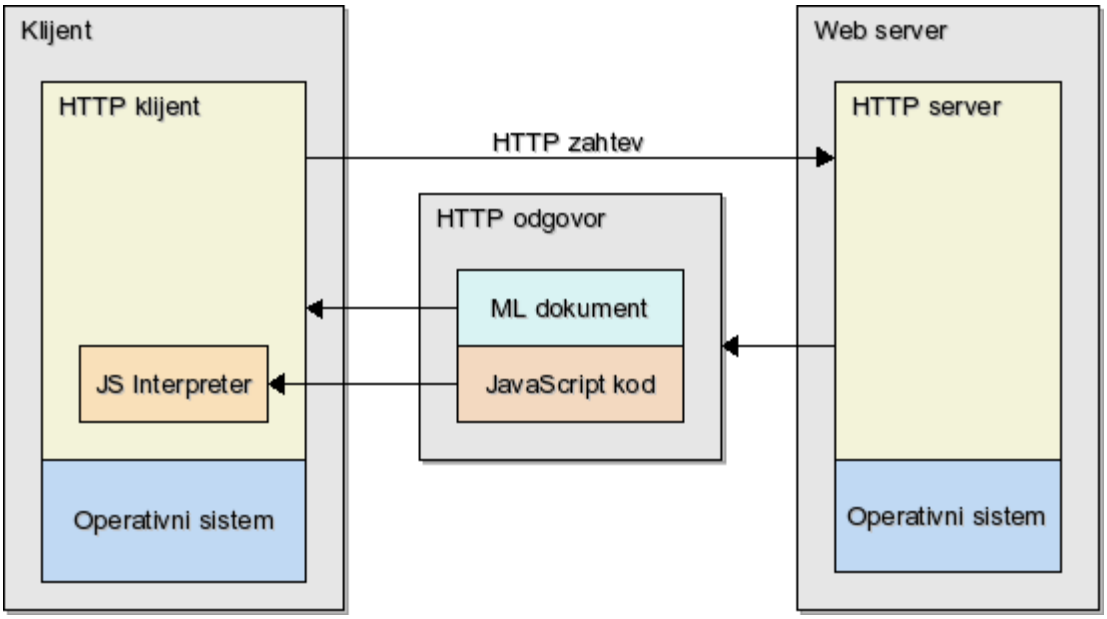
1. definisanja prikaza elemenata putem pomenutih CSS definicija
2. prevođenja dokumenata u (X)HTML putem XSLT transformacija

XSLT (Extensible Stylesheet Language Transformations) je jezik baziran na XML-u i koristi se za prevođenje XML podataka u neki drugi format. Uz korišćenje XSLT-a, XML dokumenti se mogu prevesti u HTML format na svakom Web klijentu koji ima XSLT procesor.

S obzirom na to da je CSS standard W3C organizacije [6], zvanična informacije o ovoj tehnologiji se mogu naći na: http://www.w3.org/Style/CSS

3.2.3.3. JavaScript

JavaScript tehnologija predstavlja značajnu komponentu u razvoju Web aplikacija. Ovaj programski jezik koristi sintaksu sličnu C programskom jeziku i ne poseduje bitne sličnosti osim imena sa Java programskim jezikom [8]. Naziv JavaScript je inicijalno uvela Netscape Communication korporacija (a kasnije i Mozilla fondacija) a u pitanju je implementacija ECMAScript standarda koji je postavila Ecma International organizacija za standardizaciju.



Slika 11 – Korišćenje JavaScript u okviru Web aplikacija

JavaScript programski jezik se može koristiti u različite svrhe (desktop aplikacije, server-side jezik za razvoj Web aplikacij i sl.) ali se najčešće koristi za izgradnju dinamičkog korisničkog interfejsa za Web aplikacije. S obzirom na to da je u pitanju interpreterski jezik, za izvršavanje JavaScript naredbi kod Web aplikacija potrebno je imati interpreter u okviru Web klijenta. Većina popularnih Web klijenata danas (Mozilla Firefox, Internet Explorer, Safari...) poseduje ugrađen interpreter a neki od najpopularnijih JavaScript interpretera su Rhino, Tamarin...

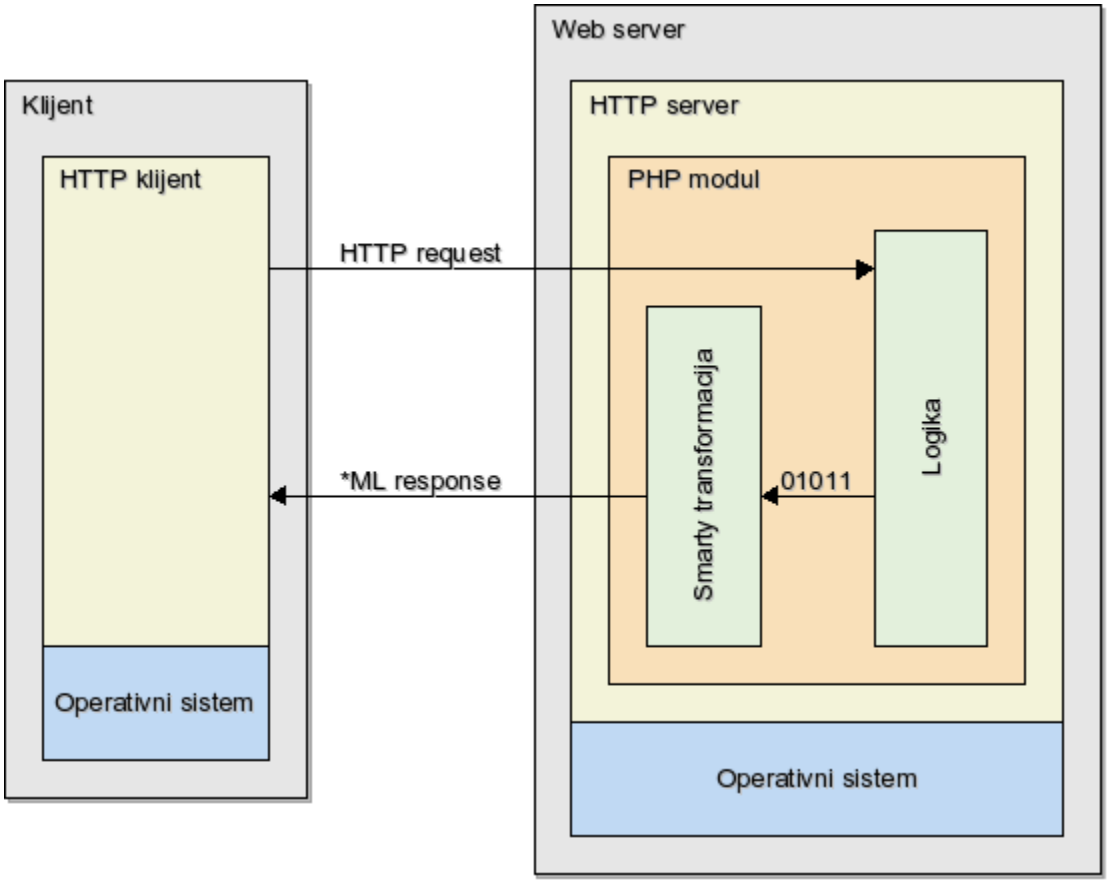
Imajući u vidu postojanje različitih interpretera u okviru različitih Web klijenata ili njihovih verzija kao i usku povezanost JavaScript-a sa DOM (Document Object Model) čija interpretacija takođe može biti različita, postoji mogućnost različite interpretacije istih JavaScript programa na različitim platformama. Takođe, pri razvoju Web aplikacija treba imati u vidu i mogućnost potpunog nepostojanja podrške za JavaScript kao i mogućnost izmene koda na klijentu. Ove probleme treba imati u vidu pri projektovanju Web aplikacija i odabiru funkcionalnosti koje se smeju izmestiti na stranu klijenta. U protivnom, moguće su delimični ili potpuni nedostaci funkcionalnosti baziranih na JavaScript-u kao i bezbednosni propusti.

U razvoju UNIS informacionog sistema JavaScript tehnologija je intenzivno korišćena za razvoj korisničkog interfejsa. U okviru različitih modula IS-a JavaScript je korišćen u skladu sa predviđenim diverzitetom korisničkih Web klijenata. JavaScript tehnologija je u okviru UNIS projekta korišćena u vidu EFW.JS framework-a o kome će kasnije biti više reči.

Imajući u vidu brojne implementacije JavaScript-a, za centralnu Web adresu može se uzeti: http://en.wikipedia.org/wiki/JavaScript

3.2.3.4. Smarty template engine

Smarty sistem šablona predstavlja proširenje PHP programskog jezika (tačnije, rešenje zasnovano na PHP programskom jeziku) izgrađeno sa ciljem da omogući efikasno i jednostavno razdvajanje poslovne logike od prezentacione logike Web aplikacije. Isključivanje prezentacionih elemenata omogućava čist i pregledan dizajn poslovne logike a izraz „prezentaciona logika“ označava da se rezulati poslovne logike mogu dodatno prilagođavati uslovima prezentovanja. Takođe, razdvanjanje ova dva dela omogućava nezavisno menjanje unutar svakog od njih.



Slika 12 - Arhitektura sistema sa Smarty template engine-om

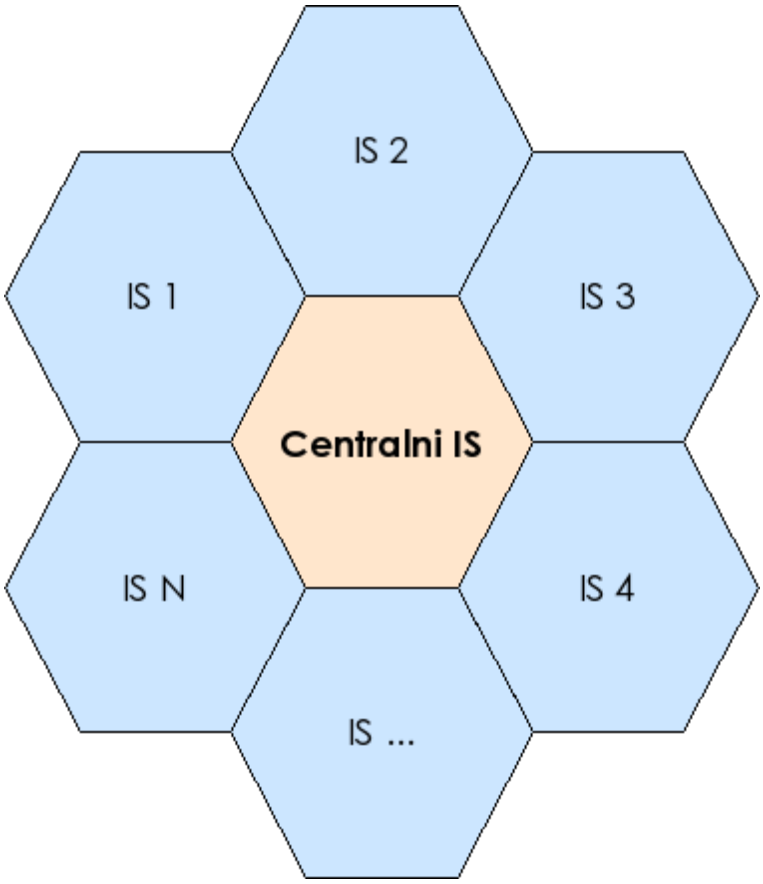
Smarty predstavlja moćan sistem koji ujedno olakšava i skraćuje vreme razvoja Web aplikacija a takođe olakšava i naknadne izmene aplikacije. Neke od glavnih karakteristika Smarty sistema šablona su [4]:

- performanse
- kompajliranje šablona
- skalabilnost po pitanju logike
- kompatibilnost sa Zend Accelerator sistemom za povećavanje performansi
- ...

Zvanična Web adresa Smarty sistema šablona je: http://smarty.php.net

3.2.4. Tehnologije vezane za izgradnju interfejsa ka drugim IS

U delu „dalji pravci razvoja“ je opisana neophodnost povezivanja centralnog informacionog sistema (a pre svega centralne baze podataka) sa ostalim informacionim (softverskim) sistemima. U skladu sa tim potrebno je izvršiti analizu tehnologija koje omogućavaju razmenu podataka između tih sistema kao i distribuiranu obradu podataka.



Slika 13 - Odnos spoljnih IS-a sa centralnim

Ovaj deo rada se bavi analizom tehnologija koje omogućavaju razmenu podataka i putem kojih se može kreirati API (Application Programming Interface) centralnog informacionog sistema za korišćenje putem mreže. Akcenat ove analize je postavljen na tehnologije bazirane na XML (Extensible Markup Language) jeziku jer one (kao i sam XML) nude mogućnost razvoja kompatibilnosti kroz prilagođavanje formata podataka. Osim tehnologija baziranih na XML-u analizirane su i JSON i Hessian tehnologije.

3.2.4.1. Tehnologije bazirane na XML-u

XML predstavlja izuzetno značajnu tehnologiju kod savremenih softverskih sistemima za distribuiranu obradu podataka. XML (Extensible Markup Language) je jezik za označavanje (eng. *markup language*) opšte namene, izgrađen sa osnovnom idejom da omogući razmenu podataka između raznorodnih sistema a putem računarskih mreža. Korišćenje XML-a je licencom odobreno bez nadoknade a kao standard je preporučen od strane W3C konzorcijuma.

Pored osnovne namene - razmene podataka - XML je moguće koristiti i u mnogim drugim primenama kao što je interno skladištenje podataka, definisanje strukture itd. Jedna od glavnih primena XML-a danas jeste za povezivanje različitih informacionih sistema putem računarskih mreža. Kao osnova se mogu koristiti protokoli svih slojeva OSI i TCP/IP modela ali se najčešće koriste protokoli aplikativnog nivoa, prvenstveno HTTP protokol. Osim za prenos podataka putem mreže između različitih softverskih sistema, korišćenjem XML-a se može kompletno obuhvatiti ceo proces distribuirane obrade podataka. Uzmimo za primer sledeću deklaraciju funkcije u C programskom jeziku:

```
int sabiranje(int a, int b) { ... }
```

Iz deklaracije funkcije se vidi da je njen naziv „sabiranje“, prihvata kao argumente dva cela broja i vraća celobrojnu vrednost. Ovu funkciju je jednostavno pozvati iz programa u kome se ona nalazi:

```
c = sabiranje(2, 3);
```

XML omogućava definisanje poziva funkcije u obliku pogodnom za prenos putem mreže (uz, naravno, postojanje adekvatnih interfejsa na obe strane):

```
<naredba>
   <pokrenuti>sabiranje</pokrenuti>
   <argument tip='int'>2</argument>
   <argument tip='int'>3</argument>
</naredba>
```

Dakle, uz postojanje adekvatnog softverskog interfejsa (kojeg, inače, nije teško realizovati) čija bi uloga bila parsiranje XML poruke i pravljenja poziva ka internoj ranije opisanoj C funkciji, sistem (koji vrši poziv, tj. šalje XML poruku) bi mogao da dobije kao rezultat takođe XML poruku:

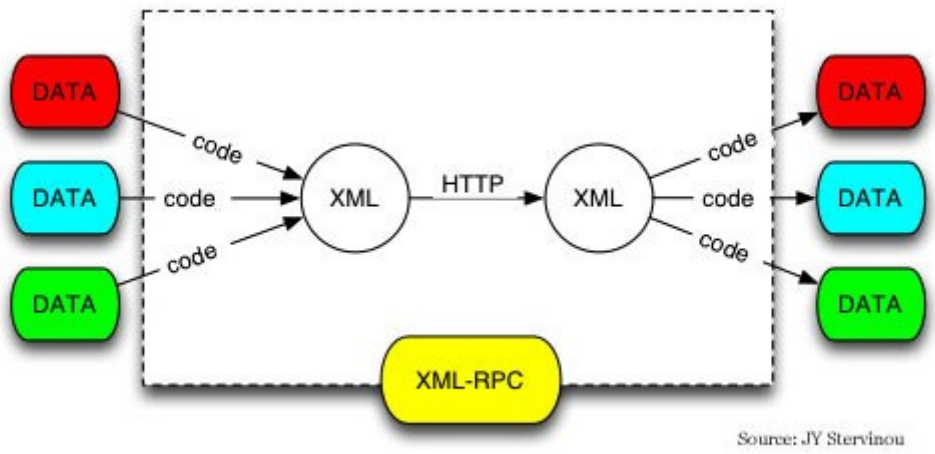
```
<odgovor>
   <rezultat tip='int'>5</rezultat>
</odgovor>
```

Iako jednostavan, ovaj primer jasno pokazuje kako je putem poruka u XML formatu moguće distribuirati pozive funkcija a samim tim i obradu podataka. U ovom radu analizirane su tri tehnologije (protokola) koje omogućavaju povezivanje softverskih sistema putem XML-a:

1. XML-RPC
2. SOAP
3. WDDX

3.2.4.1.1. XML-RPC

XML-RPC protokol omogućava pozivanje procedure na udaljenom računaru (eng. *Remote Procedure Call, RPC*) putem poruka u XML formatu koje se distribuiraju HTTP protokolom [11]. Iako jednostavan, XML-RPC protokol predstavlja izuzetno efikasno rešenje tako da je često u upotrebi i pored naprednijeg SOAP protokola.



Slika 14 - Oblast i način upotrebe XML-RPC tehnologije

XML-RPC omogućava definisanje naredbi a podržava sledeće tipove podataka:

- **array**
  niz vrednosti, ne podržava ključeve
- **base64**
  base64 kodirani binarni podaci
- **boolean**
  logička vrednost (true/false, 0/1)
- **date/time**
  vrednosti vezane za datum i vreme
- **double**
  realni brojevi
- **integer**
  celobrojne vrednosti
- **string**
  niz karaktera u kodnoj stranici XML poruke
- **struct**
  niz vrednosti sa podržanim ključevima
- **nil**
  null vrednost (XML-RPC ekstenzija)

3.2.4.1.2. SOAP

SOAP (Service Oriented Arhitecture Protocol) protokol omogućava razmenu XML poruka putem računarskih mreža (najčešće HTTP protokola) [3]. Ovaj protokol predstavlja osnovni sloj grupe protokola koja se koristi za definisanje, lociranje i korišćenje Web servisa (eng. Web services stack) i kao takav čini osnovu frejmvorka za razmenu poruka kao apstraktnog sloja OSI i TCP/IP modela [2]. Osim što omogućava pozivanje procedura na udaljenim računarima (eng. *Remote Procedure Call, RPC*), SOAP protokol omogućava i druge tipove poruka. SOAP protokol je naslednik XML-RPC protokola i koristi ista pravila za transport ali se razlikuje po načinu formatiranja (struktuiranja) poruka. Primer SOAP zahteva bi izgledao:

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <getStudentData xmlns="http://studenti.unis.local/ws">
      <Student_ID>2704</Student_ID>
    </getStudentData>
  </soap:Body>
</soap:Envelope>
```

Ovim zahtevom se poziva metoda *getStudentData* sa parametrom *Student_ID* 2704. Na ovakav zahtev server *student.unis.local* obavlja internu obradu (npr. pristupa bazi podataka i iščitava parametre sloga sa identifikacionim brojem 4837. Odgovor servera bi izgledao:

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <getStudentDataResponse xmlns="http://studenti.unis.local/ws">
      <getStudentDataResult>
        <ID>2704</ID>
        <Ime>Petar</Ime>
        <Prezime>Petrović</Prezime>
        <BrojIndeksa>III-8/2004</BrojIndeksa>
        <Fakultet_ID>2</Fakultet_ID>
        <Fakultet_Kod>FPI</Fakultet_Kod>
      </getStudentDataResult>
    </getStudentDataResponse>
  </soap:Body>
</soap:Envelope>
```

SOAP protokol predstavlja odličnu osnovu za uvođenje SOA arhitekture u Web aplikacije ali i način da se pristup određenim podacima i procedurama omogući svim hardverskim i softverskim urešajima koji imaju podršku za njega. U skladu sa tim, ovaj protokol je uzet kao izbor u delu „Prevođenje na potpunu SOA arhitekturu“.

3.2.4.1.3. WDDX

WDDX (Wev Distributed Data eXchange) tehnologija omogućava razmenu podataka između raznorodnih softverskih okruženja putem prevođenja osnovnih tipova podataka u XML format [12]. U podržane tipove podataka spadaju brojevi, stringovi, strukture, nizovi, itd. S obzirom da većina programskih jezik koristi iste ili slične tipove podataka, WDDX je podržan od strane više programskih jezika među kojima su i PHP, Ruby, Python, Java, C++, .NET, Flash, Lisp, Haskell... Prevođenjem interne strukture u WDDX paket dobija se paket kojim može da izgleda ovako:

```
<wddxPacket version='1.0'>
 <header comment='UNIS'/>
 <data>
   <struct>
     <var name='ID'>
       <number>2704</number>
     </var>
     <var name='Ime'>
       <string>Petar</string>
     </var>
     <var name='Fakultet'>
       <array length='2'>
         <number>2</number>
         <string>FPI</string>
       </array>
     </var>
   </struct>
 </data>
</wddxPacket>
```

WDDX predstavlja odlično rešenje za razmenu podataka između udaljenik i/ili raznorodnih okruženja. Glavni uticaj da SOAP postane popularnije rešenje jeste pre svega mogućnost pozivanja udaljenih procedura (što nije osnovna funkcionalnost WDDX-a) ali i činjenica da je u prvoj fazi svoj postojanja WDDX bio zatvoren za javnost što je ograničilo širenje usled nemogućnosti programera da pristup izvornom kodu rešenja. Inače, WDDX je prvi put predstavljen 1998. godine (isto kao i XML-RPC) od strane autora Simeona Simeonova (Allaire korporacija).

3.2.4.2. Ostale tehnologije

Jedna od glavnih zamerki vezanih za tehnologije bazirane na XML-u jeste „preopširnost“ njegovih poruka tj. veliki procenat kontrolnih bajtova potrebnih za definisanje poruke i uključenih podataka. Takođe, za korišćenje XML-a je potreban parser koji zahteva dodatne procesorske i memorijske resurse. Ovi nedostaci se, međutim, u današnje vreme (s obzirom na mogućnosti savremenih procesora računara) ne mogu smatrati ozbiljnim s obzirom na dodatne pogodnosti i mogućnosti koje nudi kroz platformsku nezavisnost. Ipak, u analizi dostupnih tehnologija razmatrane su i dve tehnologije za razmenu podataka između raznorodnih sistema:

1. JSON
2. Hessian

koje se ne baziraju na XML-u.

3.2.4.2.1. JSON (JavaScript Object Notation)

JSON format podrazumeva tekstualno predstavljanje objekata i ostalih tipova podataka u cilju razmene podataka između udaljenih i/ili raznorodnih sistema [13]. Osnovni princip na kome se bazira JSON jeste serijalizacija - prevođenje internih struktura podataka u format koji je moguće snimiti na spoljnu memoriju ili preneti putem mreže. Bez obzira na pominjanje JavaScript tehnologije u nazivu JSON-a, ovaj format se može koristiti i u ostalim programskim jezicima: ActionScript, C, C++, C#, ColdFusion, Common Lisp, Delphi/Object Pascal, E, Erlang, Haskell, Java, Lasso, Limbo, Lua, ML, Objective-C, Objective CAML, Perl, PHP, Python, Rebol, Ruby, Smalltalk i Tcl. Primer objekta koji se koristi u UNIS sistemu prevedenog u JSON format bi izgledao ovako:

```
{
    "ID": 2704,
    "Ime": "Petar",
    "Prezime": "Petrović",
    "BrojIndeksa": "III-8/2004",
    "Fakultet": {
        "ID": 2,
        "Kod": "FPI"
    }
}
```

3.2.4.2.2. Hessian (web service protocol)

Osnovna razlika između Hessian protokola i ostalih opisanih rešenja jeste binarni format podataka. Ovakav pristup omogućava korišćenje Hessian protokola bez potrebe za dodatnim proširenjem protokola u cilju prenosa binarnih podataka (npr. fajlova priključenih poruci) [14]. Autor Hessian protokola je kompanija Caucho Technology, Inc. koja je pored definicije protokola razvila i Java i Python implementacije (dostupne pod Apache licencom). Osim pomenutih zvaničnih implementacija postoje i implementacije razvijene od trećih lica za jezike C++, .NET C#, PHP, Ruby i Objective-C).

3.3. Analiza gotovih rešenja i komponenti

Osim analize elementarnih tehnologija za potrebe projekta UNIS izvršena je i analiza gotovih rešenja i komponenti, i to:

1. analiza postojećih informacionih sistema za univerzitete,
2. analiza postojećih univerazalnih informacionih sistema i
3. analiza gotovih komponenti.

Pod pojam gotove komponente mogu se svrstati i proširenja programskih jezika, sami programski jezici, kompajleri, sistemske biblioteke, operativni sistemi, hardver, pa čak i mediji za umrežavanje računara. Međutim, granica između osnovnih tehnologija i gotovih komponenti je postavljena na osnovu nivoa na kome komponenta radi (tj. iznad programskog jezika ili njegovog proširenja).

Korišćenje gotovih rešenja i komponenti otvara i određene bezbednosne rizike (posebno kada su u pitanju rešenja čiji izvorni kod nije dostupan) ali je savremen razvoj softvera bez njihovog korišćenja gotovo nezamisliv. Može se reći i da postoji tendencija kretanja programiranja od ručnog kreiranja koda ka menadžmentu gotovih rešenja i komponenti.

3.3.1. Analiza postojećih IS za univerzitete

Informacioni sistemi namenjeni univerzitetima nisu baš čest softverski proizvod na tržištu. Takođe, većinu rešenja (i stranih i domaćih) karakteriše interni razvoj za potrebe sopstvene organizacije a zatim nastup na tržištu kao mogućnost dodatne zarade. Za potrebe univerziteta Singidunum su u analizu uključena pronađena domaća rešenja, i to:

1. Informacioni sistem fakulteta, proizvod firme Ideasoft
2. Informacioni sistem DOSITEJ, proizvod Elektrotehničkog fakulteta u Beogradu

Međutim, kvalitet i mogućnosti pomenutog softvera nisam u mogućnosti da navodim jer nisam imao priliku da se sa njima detaljnije upoznam a moje znanje o tim sistemima se bazira na informacijama dostupnim na sajtovima proizvođača:

1. http://www.ideasoft.co.yu/projekti/i3s/index.html
2. http://rc.etf.bg.ac.yu/?p=evides

U svakom slučaju, bliskost interne organizacije univerziteta u Srbiji otvara mogućnost da su pomenuta rešenja u velikoj meri (ili čak u potpunosti) primenljiva na univerzitetu Singidunum ali rizik od zavisnosti od spoljne organizacije kada je u pitanju osetljiva komponenta poslovanja kao što je centralni informacioni sistem ipak upućuje na interni razvoj.

3.3.2. Analiza postojećih univerzalnih IS

Univerzalni informacioni sistemi uglavnom ne predstavljaju 100% gotova rešenja već razvojno okruženje visokog nivoa koje nudi biblioteke gotovih komponenti za poslovne sisteme i sintaksu u vidu njihovog načina povezivanja. Takvi sistemi omogućavaju relativno brz razvoj i eventualne izmene rešenja. Glavni nedostaci ovakvih rešenja su, pre svega, visoka cena kao i neefektivnost kada su i pitanju specifični zahtevi i nestandardni problemi. Dodatno, u praksi se pokazuje da je za korišćenje ovakvih rešenja potrebna bliska saradnja sa sertifikovanim tehničkim licima i strogo pridržavanje pravila koje sistem podrazumeva.

U glavne proizvođače ovakvih sistema spadaju Microsoft (proizvod Navision), SAP i Sybase. Od tehničke podrške Navision-a sam dobio informaciju da za glavne konkurente smatraju i sisteme: MAS 500, Visual, Symix, Macola, Made2Manage i MRP9000.

U skladu sa visokom cenom i specifičnostima poslovnog sistema univerziteta korišćenje pomenutih sistema je ostavljeno kao alternativno rešenje u budućnosti.

3.3.3. Analiza gotovih komponenti

Savremeni softverski alati, a pre svega objekto orijentisano programiranje (OOP) omogućava relativno laku integraciju gotovih komponenti kao i razvoj komponenti koje mogu biti primenjene u različitim rešenjima. Na Internetu postoji veliki broj skladišta gotovih komponenti koje se putem mreže mogu (najčešće besplatno) preuzeti i koristiti u sopstvenim rešenjima. Jedno od glavnih skladišta gotovih klasa napisanih u PHP jeziku jeste sajt http://www.phpclasses.org na kome se nalaze

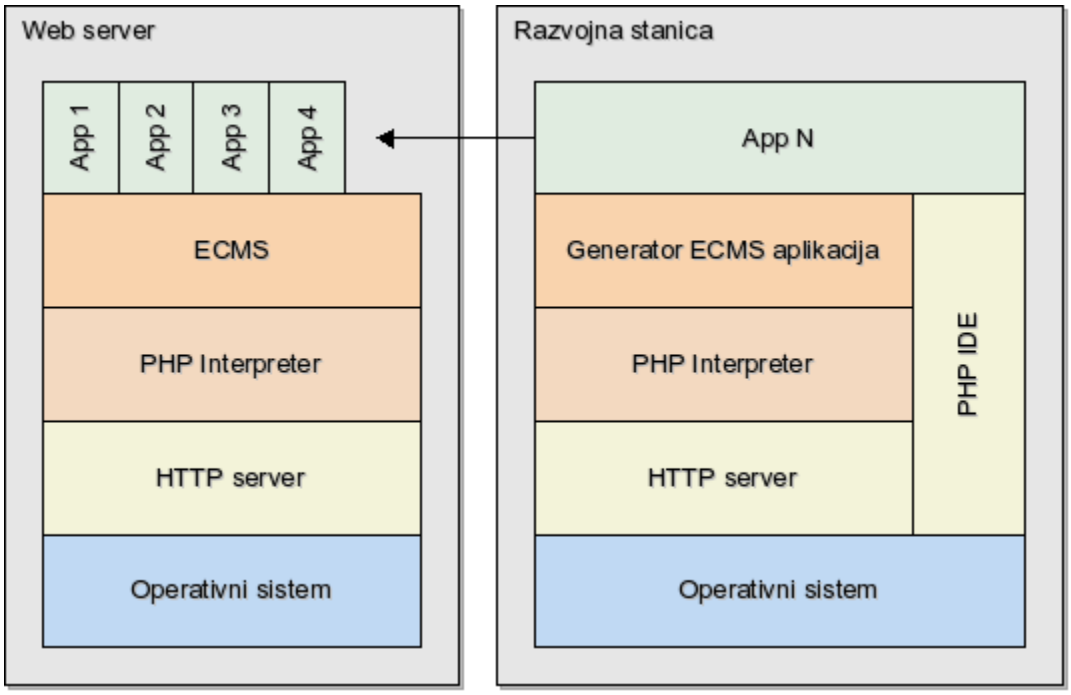
Međutim, usled ranije pomenutih bezbednosnih rizika koje otvara korišćenje gotovih komponenti drugih autora za potrebe razvoja informacionog sistem univerziteta Singidunum u razmatranje su ušle samo komponente koje je u ranijim projektima razvio autor ovog rada:

1. EMS
2. EFW.PHP
3. EFW.JS

Takođe, izbegnuto je i korišćenje velikog broja komponenti u cilju povećanja stabilnosti sistema (veliki broj komponenti, a posebno kada su u pitanju različiti proizvođači, može izazvati nestabilnost celokupnog sistema usled nekompatibilnosti) i očuvanja čistog i jednostavnog dizajna.

3.3.3.1. EMS

EMS (Expert Management System) je sistem za razvoj i platforma za izvršavanje web aplikacija. Razvijen je od strane autora ovog rada a na osnovu ECMS (Expert Content Managemet System) sistema za upravljanje sadržajem. Razvoj ECMS sistema je započet krajem 2002. godine na Višoj Tehničkoj Školi u Kragujevcu (takođe od strane autora ovog rada) u vidu projekta koji je trebalo da obezbedi kvalitetan sistem za upravljanje sadržajem Web prezentacija baziran na PHP programskom jeziku. Nedostatak modularnog dizajna je uslovio kompletan redizajn sistema tako da je krajem 2004. godine ECMS postao potpuno funkcionalan modularan (proširiv) sistem za upravljanje sadržajem. Do sredine 2005. sistem je aktivno testiran i dorađivan tako da je u septembru iste godine pušten u rad u vidu sajta Univerziteta Singidunum a ubrzo nakon toga je upotrebljen i za sajtove Fakulteta za Poslovnu Informatiku, Fakulteta za Finansijski Menadžment i Osiguranje, Fakulteta za Turistički i Hotelijerski Menadžment i Univerziteta Sinergija. Danas se ECMS sistem aktivno koristi za pomenute sajtove kao i više dodatnih internih i eksternih sajtova Univerziteta Singidunum. U proteklom periodu je ECMS uspešno odgovorio na desetine miliona zahteva klijenata i efikasno se odbranio od više DoS napada.



Slika 15 - Arhitektura korišćenja ECMS sistema

Jedna od važnih komponenti ECMS sistema jeste i generator Web aplikacija, alat koji omogućava automatsko generisanje aplikacija kompatibilnih sa ECMS sistemom a na osnovu definicije mape aplikacije. Ova komponenta je u proteklom periodu omogućila brz i efikasan razvoj preko sto Web aplikacija za potrebe Univerziteta Singidunum. Međutim, jedan od glavnih zaključaka izvučenih na osnovu ove komponente, pored toga da takva komponenta drastično smanjuje broj grešaka i vreme potrebno za razvoj, jeste i to da je ECMS sve više postajao platforma za izvršavanje najrazličitijih Web aplikacija a sve manje sistem za upravljanje statičkim sadržajem. U skladu sa tim, krajem 2006. godine je započet redizajn ECMS-a u vidu EMS-a čiji je razvoj usmeren ka platformi koja omogućava izvršavanje zahtevnih Web aplikacija uz efikasnu kontrolu. Jedna od bitnijih razlika jeste i objektno orijentisan dizajn EMS-a naspram proceduralnog ECMS-a.

Osnovni ciljevi (nasleđeni i od ECMS-a) EMS-a su:

- Omogućavanje izvršavanja Web aplikacija.
- Kontrola pristupa Web aplikacijama.
- Eliminisanje razvoja delova Web aplikacija koji se podrazumevaju, mogu se učiniti univerzalnim a predstavljaju čest izvor grešaka.
- Omogućavanje komunikacije sa raznorodnim okruženjem putem različitih formata podataka (XHTML, XML, binarni, PHP serialized...) i protokola.
- Usmeravanje autora Web aplikacija na korišćenje gotovih komponenti sistema.
- Usmeravanje autora Web aplikacija na poštovanje pravila dizajna i razvoja identifikovanih kao "najbolje prakse".
- Omogućavanje korišćenja gotovih komponenti/aplikacija napisanih u drugim programskim jezicima a pre svega u Java-i.
- Omogućavanje skalabilnosti kroz distribuirano izvršavanje aplikacija.

Iako ova lista predstavlja prilično ozbiljne zadatke za jedan softverski proizvod treba pomenuti da je većina navedenog već realizovana u okviru ECMS sistema a u EMS sistemu prolazi samo kroz kontrolu i eventualni redizajn.

3.3.3.2. EFW.PHP

Razvoj ranije opisanog EMS sistema na osnovu ECMS sistema ima za cilj da napravi moćan sistem čistog dizajna i arhitekture. U tom cilju analizom ECMS sistema razdvojene su dve kategorije:

- funkcionalnosti
- komponente

S obzirom na to da su funkcionalnosti postavljene u domen EMS sistema komponente na osnovu kojih će one biti omogućene su postavljene u domen novog projekta - EFW.PHP (Expert FrameWork.PHP). Zadatak EFW.PHP projekta jeste da obezbedi bazu univerzalnih i opšte-primenljivih komponenti (napisanih u PHP programskom jeziku) putem kojih je moguće jednostavno i efikasno izvršiti kompleksne operacije:

- pristup fajl sistemu
- pristup bazama podataka
- pristup elektronskoj pošti
- pristup LDAP serverima
- upravljanje sesijom
- upravljanje korisničkim nalozima
- ...

3.3.3.3. EFW.JS

Kao i EFW.PHP projekat, EFW.JS takođe ima zadatak da obezbedi bazu gotovih komponenti koje će biti korišćene u rešenjima baziranim na EMS platformi sa tom razlikom da komponente EFW.JS-a budu primenljive na sloju korisničkog interfejsa. Komponente EFW.JS-a su napisane u JavaScript programskom jeziku i kompatibilne su sa Mozilla Firefox web brauzerima. Neke od funkcionalnosti koje nudi EFW.JS su:

- upravljanje učitavanjem stranica
- upravljanje formularima i njihovim elementima
- upravljanje vremenom
- ...

Razvoj JS frejmvorka je kompleksan i zahtevan posao, pre svega usled nekompatibilnosti Web brauzera različitih proizvođača tako da će razvoj EFW.JS-a verovatno biti prekinut u bliskoj budućnosti a funkcionalnosti tog domena biti nadomeštene nekim od popularnih projekata kao što su:

- Prototype
- Dojo Toolkit
- Mootools
- Adobe Spry framework
- ...

4. Model IS-a

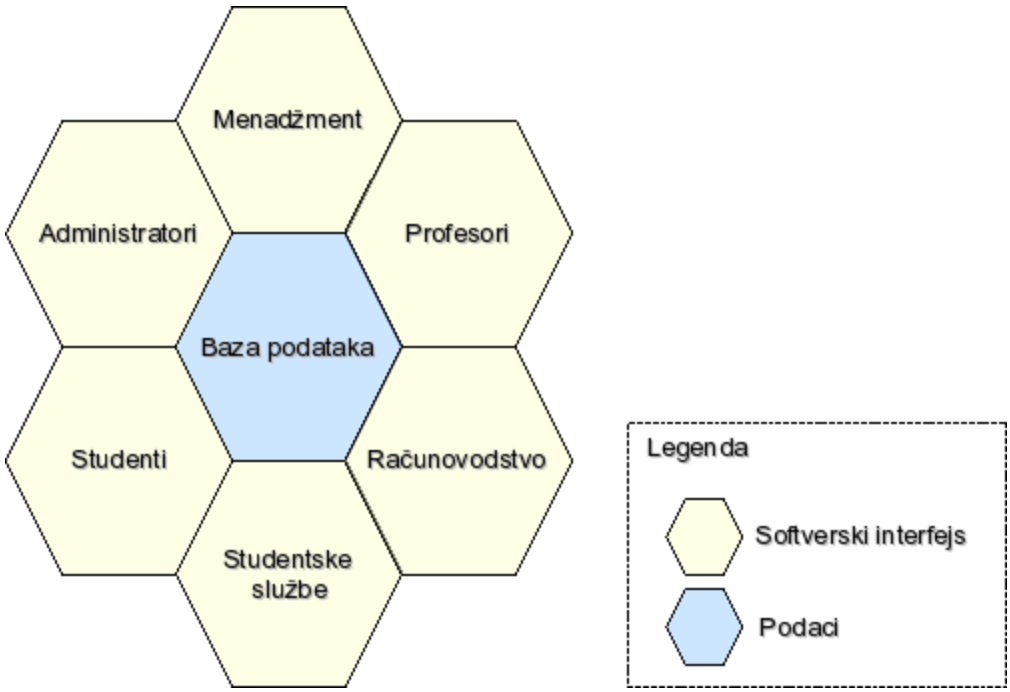
Na osnovu analiza opisanih u 2. i 3. delu rada napravljeno je više modela:

1. model korisničkih modula
2. model arhitekture IS-a
3. model baze podataka

na osnovu kojih je izgrađen informacioni sistem.

4.1. Kategorije korisničkih modula

Analizom poslovnog sistema (poslovnih procesa i kategorija korisnika) identifikovano je 6 kategorija korisnika informacionog sistema, tj. 6 različitih tipova korisnika za koje je potrebno obezbediti različite skupove funkcionalnosti. Imajući u vidu takvu raspodelu funkcionalnosti kao i eventualne propuste u analizi i/ili eventualna dodavanja kategorija vremenom, odabrana je modularna struktura informacionog sistema za organizaciju funkcionalnosti.



Slika 16 – Model modularnog informacionog sistema

Kod modularne ovakve modularne organizacije moduli predstavljaju interfejse korisnika ka bazi podataka. Dodatna pogodnost ovakve organizacije jeste i mogućnost dodele različitih privilegija nad podacima baze podataka na nivou kategorije korisnika tj. određenog modula. Takođe, s obzirom na to da je i interna organizacija modula modularna, to ostavlja mogućnost internog proširivanja modula novim funkcionalnostima u vidu pod-modula. U narednom tekstu je data lista funkcionalnosti koje su pridružene identifikovanim modulima.

4.1.1. Menadžment

U okvir modula namenjenoj korisnicima kategorije „Menadžment“ postavljene su sledeće funkcionalnosti:

- rad sa fakultetnima
- rad sa studijskim programima (smerovima)
- rad sa instruktorima (profesorima i asistentima)
- rad sa tipovima studija (klasično, DLS...)
- rad sa lokacijama

4.1.2. Studentske službe

U okvir modula namenjenoj korisnicima kategorije „Studentske službe“ postavljene su sledeće funkcionalnosti:

- rad sa studentima
- rad sa prijavama ispita studenata, ispitnim rokovima i ostvarenim rezultatima

4.1.3. Profesori

U okvir modula namenjenoj korisnicima kategorije „Profesori“ trenutno nije stavljena ni jedna funkcionalnost s obzirom na to da ovaj modul nije bio neophodan za odgovor na osnovne zahteve. Odložena realizacija ovog modula ostavlja mogućnost dodatne analize i odabira funkcionalnosti koje treba da omogći. Takođe, nakon realizacije ovog modula postoji i mogućnost da se određene funkcionalnosti drugih modula (npr. funkcionalnost popunjavanja ispitnog zapisnika u okviru modula „Studentske službe“) izmeste u njega.

4.1.4. Studenti

U okvir modula namenjenoj korisnicima kategorije „Studenti“ postavljene su sledeće funkcionalnosti:

- rad sa prijavama ispita
- rad sa finansijama

Funkcionalnost „rad sa finansijama“ trenutno nije u potpunosti realizovana s obzirom na to da je uslovljena realizacijom određenih funkcionalnosti u modulu „Računovodstvo“.

4.1.5. Računovodstvo

U okvir modula namenjenoj korisnicima kategorije „Računovodstvo“ postavljene su sledeće funkcionalnosti:

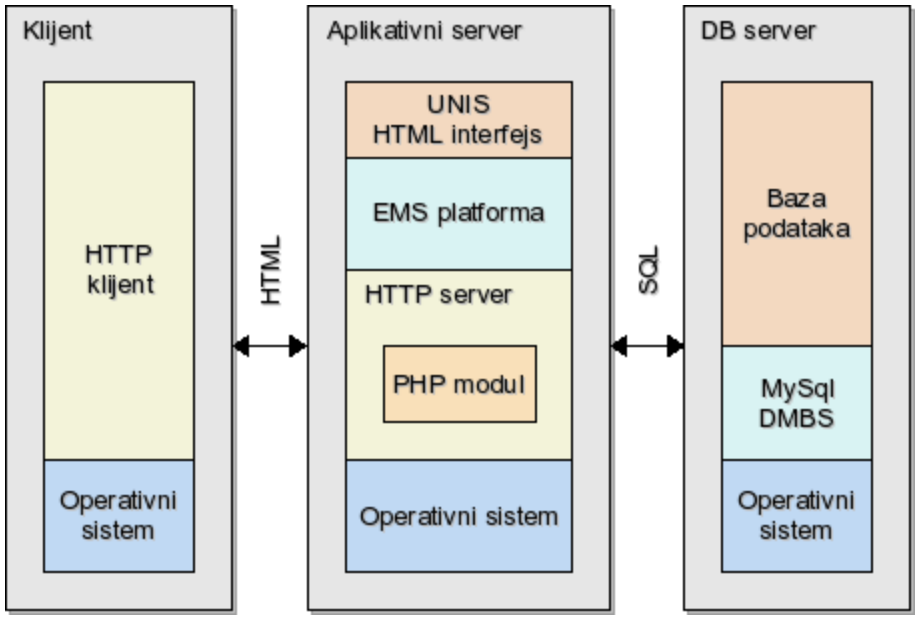
- rad sa žiro računima
- rad sa izvodima
- rad sa evidencijom o kretanju kursa dinara/evra

4.1.6. Administratori IS-a

U okvir modula namenjenoj korisnicima kategorije „Administratori“ trenutno nije stavljena ni jedna funkcionalnost s obzirom na to da ovaj modul nije bio neophodan za odgovor na osnovne zahteve. Odložena realizacija ovog modula ostavlja mogućnost dodatne analize i odabira funkcionalnosti koje treba da omogći. Jedan od glavnih razloga zbog koga je poželjno odložiti realizaciju ovog modula jeste potreba za većom količinom podataka u bazi i većim brojem obavljenih akcija (od strane korisnika ostalih kategorija) da bi se utvrdile eventualne administrativne akcije koje su potrebne za održavanje sistema.

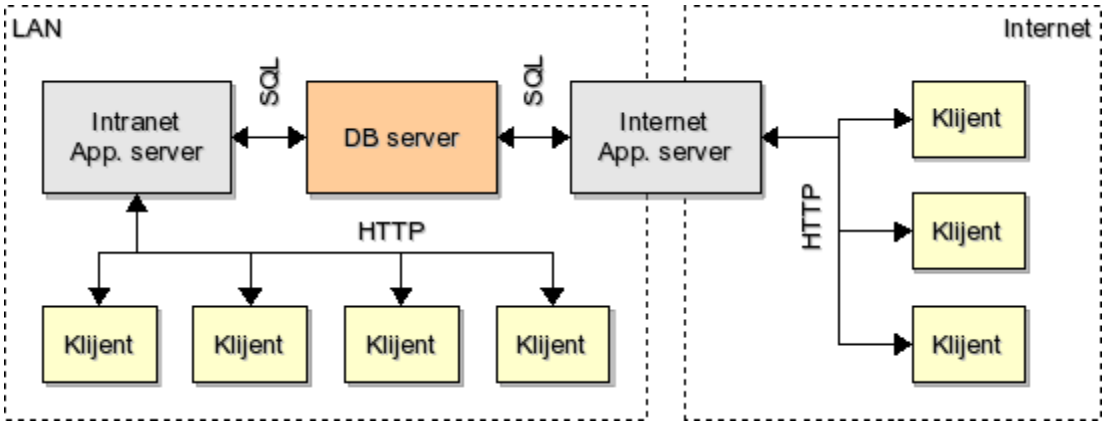
4.2. Model arhitekture IS-a

Na osnovu rezultata dobijenih iz analize opisane u delu 3.1 „Analiza softverskih arhitektura“, kao i ostalih analiza iz 3. dela rada, napravljen je model arhitekture informacionog sistema. Inicijalna odabrana arhitektura (klijent-server) u svojoj troslojnoj varijanti podrazumeva slojeve korisničkog interfejsa, logike i podataka.



Slika 17 - Klijent-server arhitektura informacionog sistema

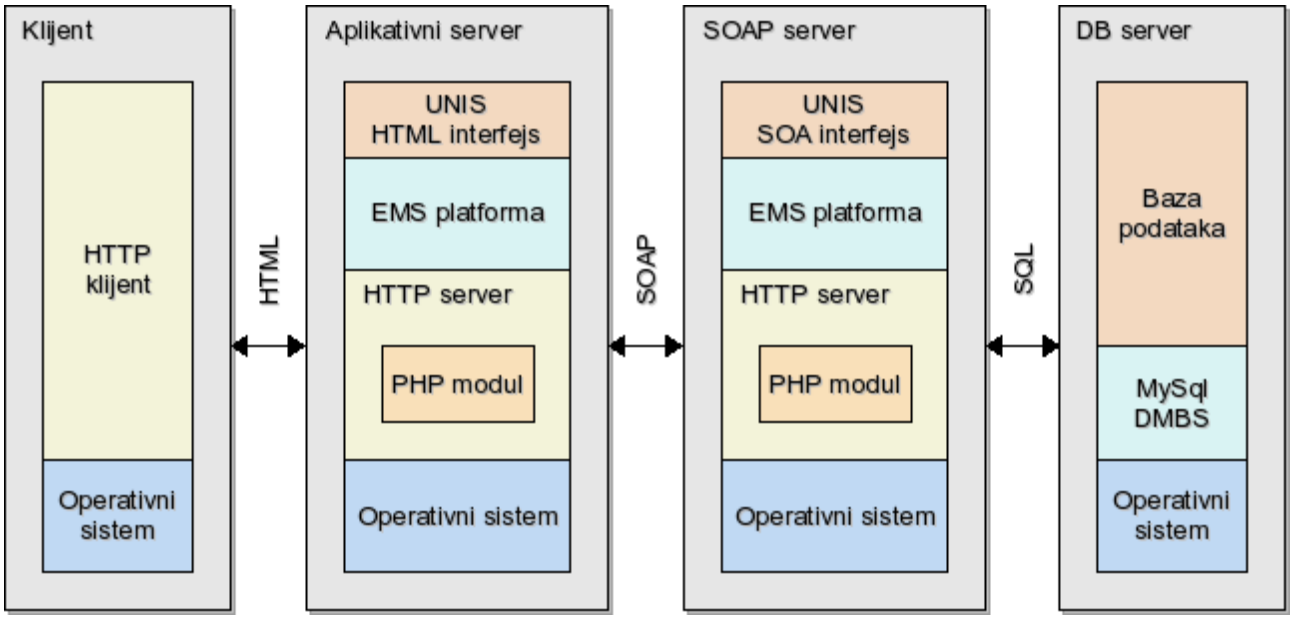
Ovakva arhitektura eliminiše potrebu za razvojem posebnog klijentskog softvera jer omogućava korišćenje gotovih Web klijenata (Internet Explorer, Mozilla Firefox...). Mrežna realizacija ovakve arhitekture podrazumeva instalaciju jednog ili više aplikativnih servera koji bi direktno komunicirali sa serverom baze podataka. U tom slučaju je za pristup sistemu sa Interneta (prijava ispita od strane studenata) potrebno instalirati aplikativni server na granici između lokalne mreže i Inteneta, tj. server koji je dostupan putem Internet mreže a ima pristup serveru baze podataka u lokalnoj mreži.



Slika 18 - Mrežna realizacija klijent-server arhitekture

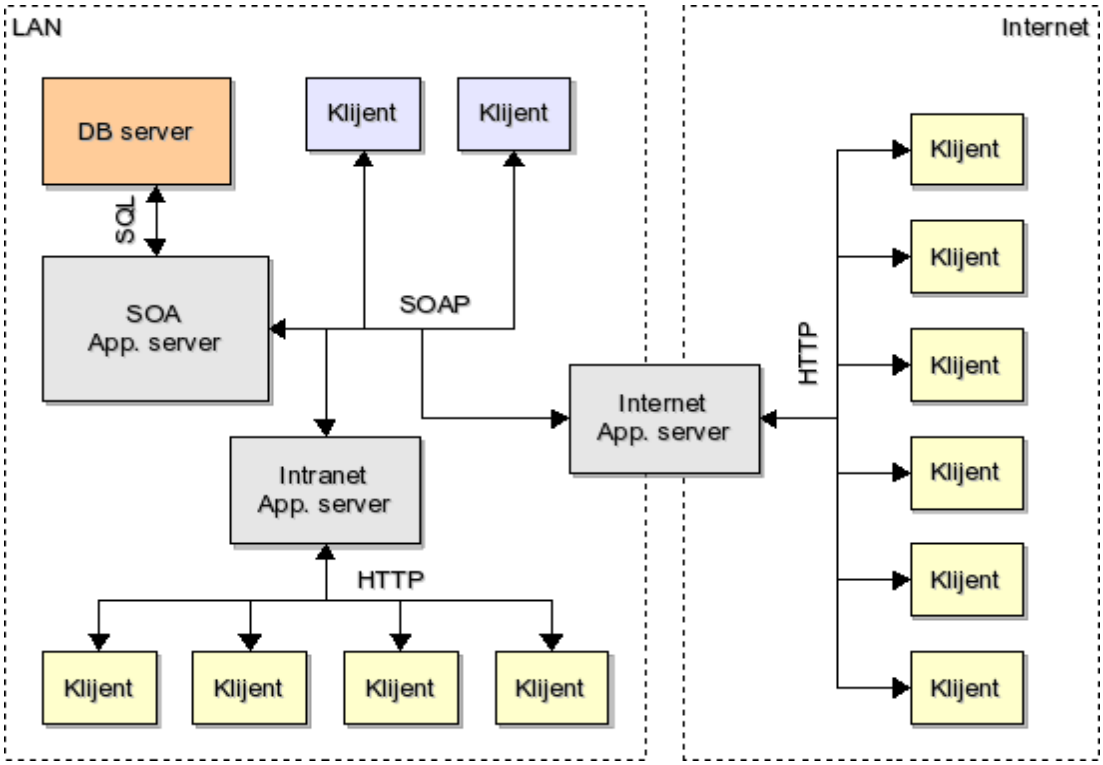
Ovakva arhitektura (klijent-server) i njena mrežna realizacija, iako jednostavna i funkcionalna (a ujedno i laka za razvoj) sa sobom nosi određene bezbednosne rizike. Na primer, ukoliko neko ostvari kontrolu nad aplikativnim serverom vidljivim sa Interneta, on ima mogućnost pristupa

centralnoj bazi podataka sa privilegijama koje imaju korisnički moduli instalirani na tom serveru. U skladu sa tim, za dalji razvoj informacionog sistema je uzeta u obzir SOA arhitektura koja nudi jednu od mogućnosti za dodatnu kontrolu pristupa centralnoj bazi podataka.



Slika 19 - SOA arhitektura informacionog sistema

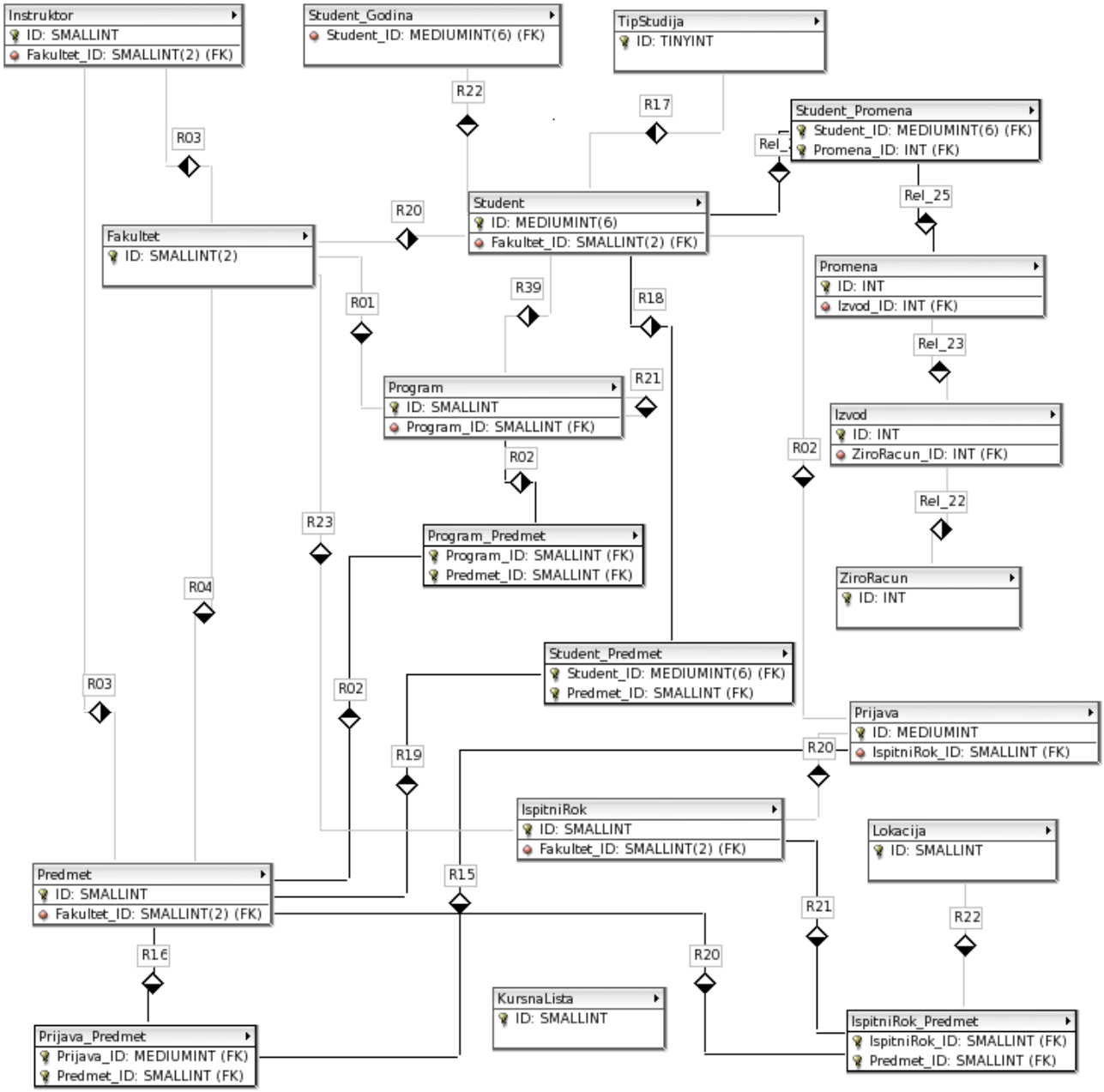
Prevođenje sistema na SOA arhitekturu, iako traži razvoj dodatnih modula za podršku SOAP protokola, nudi mogućnost ograničavanja pristupa centralnoj bazi podataka na posredstvo SOA servera na kome je moguće primeniti dodatne mehanizme zaštite i kontrole pristupa.



Slika 20 - Mrežna realizacija informacionog sistema baziranog na SOA arhitekturi

4.3. Model baze podataka

Baza podataka informacionog sistema projektovana je tako da podrži buduća proširenja informacionog sistema i u pogledu razvoja novih funkcionalnosti putem korisničkih modula i u pogledu prelaska na drugi sistem za upravljanje bazama podataka (SUBP). MySQL kao planirani SUBP nudi prihvatljive performanse, stabilnost i pouzdanost u prvom stadijumu razvoja ali sa obzirom na planirani rast količine podataka i frekvencije njihove upotrebe postoji mogućnost pojavljivanja potrebe za prelaskom na robustnije sisteme (npr. Oracle).



Slika 21 – Apstraktan model baze podataka

Pri modeliranju tabela baze podataka korišćeni su opšti tipovi podataka i izbegnuto je korišćenje specifičnih funkcionalnosti MySql SUBP-a. Npr. zaključavanje zapisa u tabeli je rešeno putem *integer* polja sa *timestamp*-om pristupa i uz proveru na aplikativnom nivou.

4.3.1. Tabele baze podataka

Lista i definicije tabela baze podataka po abecednom redosledu:

Fakultet	
ID	SMALLINT(2) UNSIGNED NOT NULL AUTO_INCREMENT
Kod	VARCHAR(8) NOT NULL
Naziv	VARCHAR(255) NOT NULL
IndeksPrefiks	VARCHAR(8) NULL
Adresa	VARCHAR(255) NOT NULL
Grad	VARCHAR(255) NULL
PTT	VARCHAR(255) NULL
ZiroRacun	VARCHAR(255) NOT NULL
Active	TINYINT(1) UNSIGNED NOT NULL DEFAULT '1'
Locked	INT UNSIGNED NOT NULL DEFAULT '0'
PRIMARY KEY(ID) UNIQUE INDEX Fakultet_Kod(Kod) UNIQUE INDEX Fakultet_Naziv(Naziv)	

Instruktor	
ID	SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT
Fakultet_ID	SMALLINT(2) UNSIGNED NOT NULL
Ime	VARCHAR(255) NOT NULL
SrednjeIme	VARCHAR(255) NULL
Prezime	VARCHAR(255) NOT NULL
JMBG	VARCHAR(255) NULL
Titula	VARCHAR(255) NULL
Email	VARCHAR(255) NULL
Telefon	VARCHAR(255) NULL
Mobilni	VARCHAR(255) NULL
Adresa	VARCHAR(255) NULL
Grad	VARCHAR(255) NULL

PTT	VARCHAR(255) NULL
TekuciRacun	VARCHAR(255) NULL
Lozinka	VARCHAR(32) NULL
Napomena	TEXT NULL
Active	TINYINT(1) UNSIGNED NOT NULL DEFAULT '1'
Locked	INT UNSIGNED NOT NULL DEFAULT '0'
PRIMARY KEY(ID) INDEX Instruktor_FKIndex1(Fakultet_ID)	

IspitniRok	
ID	SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT
Fakultet_ID	UNSIGNED NOT NULL
Naziv	VARCHAR(255) NOT NULL
GodinaSkolska	SMALLINT UNSIGNED NOT NULL
PokrivaSemestar	TINYINT(1) UNSIGNED NOT NULL DEFAULT '1'
PrijavaStart	INT UNSIGNED NOT NULL
PrijavaStop	INT UNSIGNED NOT NULL
PrijavaNaknadna	TINYINT(1) UNSIGNED NOT NULL DEFAULT '0'
IspitiStart	INT UNSIGNED NOT NULL
IspitiStop	INT UNSIGNED NOT NULL
CenaIspita	DOUBLE(7,2) NOT NULL DEFAULT '0'
CenaIspitaNaknadna	DOUBLE(7,2) NULL DEFAULT '0'
Napomena	TEXT NULL
Active	TINYINT(1) UNSIGNED NOT NULL DEFAULT '1'
Locked	INT UNSIGNED NOT NULL DEFAULT '0'
PRIMARY KEY(ID) INDEX IspitniRok_FKIndex1(Fakultet_ID)	

IspitniRok_Predmet	
IspitniRok_ID	SMALLINT UNSIGNED NOT NULL

Predmet_ID	SMALLINT UNSIGNED NOT NULL
Lokacija_ID	SMALLINT UNSIGNED NOT NULL
Vreme	INT UNSIGNED NOT NULL
Napomena	TEXT NULL
PRIMARY KEY(IspitniRok_ID, Predmet_ID) INDEX Predmet_IspitniRok_FKIndex1(Predmet_ID) INDEX Predmet_IspitniRok_FKIndex2(IspitniRok_ID) INDEX Predmet_IspitniRok_FKIndex3(Lokacija_ID)	

Izvod	
ID	INT NOT NULL AUTO_INCREMENT
ZiroRacun_ID	INT NOT NULL
BrojIzvoda	INT NULL
DatumIzvoda	INT NULL
PrethodniSaldo	DOUBLE(7,2) NULL
BrojNalogaPriliva	SMALLINT UNSIGNED NULL
SumaPriliva	DOUBLE(7,2) NULL
BrojNalogaOdliva	SMALLINT UNSIGNED NULL
SumaOdliva	DOUBLE(7,2) NULL
Saldo	DOUBLE(7,2) NULL
PRIMARY KEY(ID) INDEX Izvod_FKIndex1(ZiroRacun_ID)	

KursnaLista	
ID	SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT
Datum	INT NOT NULL
KKurs	DOUBLE(7,2) NULL
SKurs	DOUBLE(7,2) NOT NULL
PKurs	DOUBLE(7,2) NULL
PRIMARY KEY(ID) UNIQUE INDEX KursnaLista_Datum(Datum)	

Lokacija	
ID	SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT
Naziv	VARCHAR(255) NOT NULL
Sprat	TINYINT UNSIGNED NULL DEFAULT '0'
Adresa	VARCHAR(255) NULL
Grad	VARCHAR(255) NULL
PTT	VARCHAR(255) NULL
Active	TINYINT(1) UNSIGNED NOT NULL DEFAULT '1'
Locked	INT UNSIGNED NOT NULL DEFAULT '0'
PRIMARY KEY(ID)	

Predmet	
ID	SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT
Fakultet_ID	SMALLINT(2) UNSIGNED NOT NULL
Instruktor_ID	SMALLINT UNSIGNED NOT NULL
Naziv	VARCHAR(255) NOT NULL
Verzija	TINYINT UNSIGNED NOT NULL DEFAULT '1'
Trajanje	TINYINT(1) UNSIGNED NOT NULL DEFAULT '1'
Predavanja	TINYINT UNSIGNED NOT NULL DEFAULT '1'
Vezbe	TINYINT UNSIGNED NOT NULL DEFAULT '0'
Active	TINYINT(1) UNSIGNED NOT NULL DEFAULT '1'
Locked	INT UNSIGNED NOT NULL DEFAULT '0'
PRIMARY KEY(ID) INDEX Predmet_FKIndex1(Fakultet_ID) INDEX Predmet_FKIndex2(Instruktor_ID) UNIQUE INDEX Predmet_FNV(Fakultet_ID Naziv Verzija)	

Prijava	
ID	MEDIUMINT UNSIGNED NOT NULL AUTO_INCREMENT

IspitniRok_ID	SMALLINT UNSIGNED NOT NULL
Student_ID	MEDIUMINT(6) UNSIGNED NOT NULL
Zatvorena	TINYINT(1) UNSIGNED NOT NULL DEFAULT '0'
Overena	TINYINT(1) UNSIGNED NOT NULL DEFAULT '0'
Naknadna	TINYINT(1) UNSIGNED NOT NULL DEFAULT '0'
Datum	INT UNSIGNED NOT NULL
PRIMARY KEY(ID) INDEX Prijava_FKIndex1(Student_ID) INDEX Prijava_FKIndex2(IspitniRok_ID) UNIQUE INDEX Prijava_Student_IspitniRok(IspitniRok_ID Student_ID)	

Prijava_Predmet	
Prijava_ID	MEDIUMINT UNSIGNED NOT NULL
Predmet_ID	SMALLINT UNSIGNED NOT NULL
Poeni	SMALLINT UNSIGNED NULL
Ocena	TINYINT(2) UNSIGNED NOT NULL DEFAULT '0'
Priznat	TINYINT(1) UNSIGNED NOT NULL DEFAULT '0'
Napomena	TEXT NULL
PRIMARY KEY(Prijava_ID, Predmet_ID) INDEX Prijava_Predmet_FKIndex1(Prijava_ID) INDEX Prijava_Predmet_FKIndex2(Predmet_ID)	

Program	
ID	SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT
Program_ID	SMALLINT UNSIGNED NOT NULL DEFAULT '0'
Fakultet_ID	SMALLINT(2) UNSIGNED NOT NULL
Naziv	VARCHAR(255) NOT NULL
Verzija	TINYINT UNSIGNED NOT NULL DEFAULT '1'
Active	TINYINT(1) UNSIGNED NOT NULL DEFAULT '1'
Locked	INT UNSIGNED NOT NULL DEFAULT '0'
PRIMARY KEY(ID)	

```
INDEX Program_FKIndex1(Fakultet_ID)
INDEX Program_FKIndex2(Program_ID)
UNIQUE INDEX Program_PFNV(Fakultet_ID Naziv Verzija)
```

Program_Predmet	
Program_ID	SMALLINT UNSIGNED NOT NULL
Predmet_ID	SMALLINT UNSIGNED NOT NULL
Semestar	TINYINT(1) UNSIGNED NOT NULL
Izborni	TINYINT(1) UNSIGNED NOT NULL DEFAULT '0'
PRIMARY KEY(Program_ID, Predmet_ID) INDEX Program_Predmet_FKIndex1(Program_ID) INDEX Program_Predmet_FKIndex2(Predmet_ID)	

Promena	
ID	INT NOT NULL AUTO_INCREMENT
Izvod_ID	INT NOT NULL
Tip	VARCHAR(32) NOT NULL
TekuciRacun	VARCHAR(255) NULL
Naziv	VARCHAR(255) NULL
Iznos	DOUBLE(7,2) NULL
Svrha	VARCHAR(255) NULL
PNBZ	VARCHAR(255) NULL
PNBO	VARCHAR(255) NULL
DatumValute	INT NULL
Referenca	VARCHAR(255) NULL
Prioritet	VARCHAR(255) NULL
PRIMARY KEY(ID) INDEX Promena_FKIndex1(Izvod_ID)	

Student	
ID	MEDIUMINT(6) UNSIGNED NOT NULL AUTO_INCREMENT

Fakultet_ID	SMALLINT(2) UNSIGNED NOT NULL
Program_ID	SMALLINT UNSIGNED NOT NULL
TipStudija_ID	TINYINT UNSIGNED NOT NULL
Indeks	SMALLINT UNSIGNED NOT NULL
SGodinaUpisa	SMALLINT UNSIGNED NOT NULL
DGodinaUpisa	TINYINT(1) UNSIGNED NOT NULL DEFAULT '1'
Ime	VARCHAR(255) NOT NULL
SrednjeIme	VARCHAR(255) NOT NULL
Prezime	VARCHAR(255) NOT NULL
JMBG	VARCHAR(16) NOT NULL
Email	VARCHAR(255) NULL
Telefon	VARCHAR(255) NULL
Mobilni	VARCHAR(255) NULL
Adresa	VARCHAR(255) NULL
Grad	VARCHAR(255) NULL
PTT	VARCHAR(255) NULL
Zemlja	VARCHAR(255) NULL
Diplomirao	INT UNSIGNED NOT NULL DEFAULT '0'
Zaposlen	TINYINT(1) UNSIGNED NOT NULL DEFAULT '0'
Ispisan	TINYINT(1) UNSIGNED NOT NULL DEFAULT '0'
Bilans	DOUBLE(7,2) NOT NULL DEFAULT '0'
Lozinka	VARCHAR(32) NULL
Napomena	TEXT NULL
Active	TINYINT(1) UNSIGNED NOT NULL DEFAULT '1'
Locked	INT UNSIGNED NOT NULL DEFAULT '0'
PRIMARY KEY(ID) INDEX Student_FKIndex1(Program_ID) INDEX Student_FKIndex2(TipStudija_ID) UNIQUE INDEX Student_Indeks(Fakultet_ID TipStudija_ID SGodinaUpisa DGodinaUpisa Indeks) INDEX Student_FKIndex3(Fakultet_ID)	

Student_Godina	
Student_ID	MEDIUMINT(6) UNSIGNED NOT NULL
GodinaSkolska	SMALLINT UNSIGNED NOT NULL
GodinaStudija	TINYINT UNSIGNED NOT NULL DEFAULT '1'
Skolarina	DOUBLE(7,2) NOT NULL DEFAULT '0'
SkolarinaUplacena	DOUBLE(7,2) NOT NULL DEFAULT '0'
OverioSemestar	TINYINT UNSIGNED NOT NULL DEFAULT '0'
VremeUpisa	INT UNSIGNED NOT NULL DEFAULT '0'
Napomena	TEXT NULL
Locked	INT UNSIGNED NOT NULL DEFAULT '0'
INDEX Student_Godina_Index1(Student_ID) UNIQUE INDEX Student_Godina_SGGS(Student_ID, GodinaSkolska, GodinaStudija)	

Student_Predmet	
Student_ID	MEDIUMINT(6) UNSIGNED NOT NULL
Predmet_ID	SMALLINT UNSIGNED NOT NULL
PRIMARY KEY(Student_ID Predmet_ID) INDEX Student_has_Predmet_FKIndex1(Student_ID) INDEX Student_has_Predmet_FKIndex2(Predmet_ID)	

Student_Promena	
Student_ID	MEDIUMINT(6) UNSIGNED NOT NULL
Promena_ID	INT NOT NULL
Tip	TINYINT UNSIGNED NOT NULL
Napomena	VARCHAR(255) NULL
PNB	VARCHAR(255) NULL
PRIMARY KEY(Student_ID Promena_ID) INDEX Student_has_Promena_FKIndex1(Student_ID) INDEX Student_has_Promena_FKIndex2(Promena_ID)	

TipStudija	
ID	TINYINT UNSIGNED NOT NULL AUTO_INCREMENT
IndeksPrefiks	VARCHAR(8) NULL
Naziv	VARCHAR(255) NOT NULL
Active	TINYINT(1) UNSIGNED NOT NULL DEFAULT '1'
Locked	INT UNSIGNED NOT NULL DEFAULT '0'
PRIMARY KEY(ID) UNIQUE INDEX TipStudija_IndeksPrefiks(IndeksPrefiks) UNIQUE INDEX TipStudija_Naziv(Naziv)	

ZiroRacun	
ID	INT NOT NULL AUTO_INCREMENT
Racun	VARCHAR(32) NOT NULL
Naziv	VARCHAR(255) NULL
Drajver	VARCHAR(32) NOT NULL
PRIMARY KEY(ID)	

4.4. Vremensko/funkcionalni model realizacije i implementacije

Osim razvoja samog modela informacionog sistema razvijen je i model realizacije i implementacije rešenja. Ono što treba naglasiti jeste da faza implementacije neznatno kasni za realizacijom usled vrlo ograničenog vremena u okviru koga je sistem trebalo omogućiti korisnicima kao i intenziteta testiranja realizovanih jedinica.

Inicijalni period razvoja (oko nedelju dana) je iskorišćen za modelovanje baze podataka. S obzirom na to da ova komponenta informacionog sistema predstavlja preduslov za normalan razvoj ostatka sistema, kao i da naknadne izmene modela baze podataka mogu imati veliki uticaj na ostatak sistema, ovom koraku je posvećena posebna pažnja a konačan model je testiran različitim scenarijima iz prakse. Dodatno, baza je projektovana što univerzalnije imajući u vidu buduća proširenja i namene. Takođe, imajući u vidu moguće izmene inicijalnog modela softverske jedinice za pristup bazi (brokeri) su projektovani tako da se automatski prilagođavaju naknadnim izmenama.

Nakon razvoja modela baze podataka izvršen je presek skupa planiranih funkcionalnosti i na taj način dobijen podskup funkcionalnosti koji omogućava odgovor na osnovni korisnički zahtev - elektronsku prijavu ispita studenata koji studiraju putem Interneta (Distance Learning).

Nakon identifikovanja ovih funkcionalnosti iz pomenutog podskupa i njihovog pridruživanja adekvatnoj kategoriji korisnika (tj. korisničkom modulu), izvršena je analiza zavisnosti čiji je rezltat iskorišćen kao plan realizacije:

F/T	I	II	III	IV	V	VI	VII	VIII	IX
Menadžment									
S. Služba									
Studenti									
Profesori									
Adm.									
Računovo.									

Tabela 1 - Vremenski plan realizacije po modulima

- **I nedelja**
  - Menadžment
    - Fakulteti
      - Dodavanje
    - Programi (Smerovi)
      - Dodavanje
      - Izmena
      - Pregled
    - Predmeti
      - Dodavanje
      - Izmena
- **II Nedelja**
  - Menadžment
    - Instruktori
      - Dodavanje
      - Pregled
    - Tipovi studija
      - Dodavanje
    - Lokacije
      - Dodavanje
      - Pregled
- **III Nedelja**
  - Studentske službe
    - Studenti
      - Dodavanje
      - Uklanjanje
      - Pregled
- **IV Nedelja**
  - Studentske službe
    - Ispitni rokovi
      - Dodavanje
      - Uklanjanje
      - Pregled
- **V Nedelja**
  - Studentske službe
    - Ispitni rokovi
      - Izmena
- **VI Nedelja**
  - Studenti
    - Sesija
      - Prijava/Odjava
- **VII Nedelja**
  - Studenti
    - Prijava ispita
      - Nova prijava
- **VIII Nedelja**
  - Studentske službe
    - Prijave
      - Dodavanje
      - Overa
      - Pregled
    - Ispitni rokovi
      - Zapisnik
- **IX Nedelja**
  - Studenti
    - Prijava ispita
      - Pregled prijava

Tabela 2 - Prošireni vremenski plan realizacije po modulima

5. Prikaz realizovanih funkcionalnosti IS-a

U ovom delu rada su predstavljene realizovane funkcionalnosti informacionog sistema kao i način njihovog korišćenja. Ovaj deo rada se takođe može koristiti kao osnovna dokumentacija za korisnike IS-a. Dokumentaciju vremenom treba proširivati u skladu sa daljim razvojem informacionog sistema tj. realizacijom dodatnih funkcionalnosti ili izmenom/uklanjanjem postojećih. Moduli unutar kojih su realizovane određene funkcionalnosti obuhvataju:

- Menadžment
- Studentske službe
- Studenti

Sve funkcionalnosti su organizovane u okviru modula i njima pripadajućih kategorija:

- **Menadžment**
  - Fakulteti
    - Dodavanje
  - Programi (Smerovi)
    - Dodavanje
    - Izmena
    - Pregled
  - Predmeti
    - Dodavanje
    - Izmena
  - Instruktori
    - Dodavanje
    - Pregled
  - Tipovi studija
    - Dodavanje
  - Lokacije
    - Dodavanje
    - Pregled
- **Studentske službe**
  - Studenti
    - Dodavanje
    - Uklanjanje
    - Pregled
  - Ispitni rokovi
    - Dodavanje
    - Izmena
    - Uklanjanje
    - Pregled
    - Zapisnik
  - Prijave
    - Dodavanje
    - Overa
    - Pregled
- **Studenti**
  - Sesija
    - Prijava / Odjava
  - Prijava ispita
    - Nova prijava
    - Pregled prijava

5.1. Menadžment

Unutar softverskog interfejsa (modula) namenjenog menadžmentu realizovane su sledeće funkcionalnosti:

- Fakulteti:
  - Dodavanje
- Programi (Smerovi):
  - Dodavanje
  - Izmena
  - Pregled
- Predmeti:
  - Dodavanje
  - Izmena
- Instruktori:
  - Dodavanje
  - Pregled
- Tipovi studija:
  - Dodavanje
- Lokacije:
  - Dodavanje
  - Pregled

5.1.1. Fakulteti

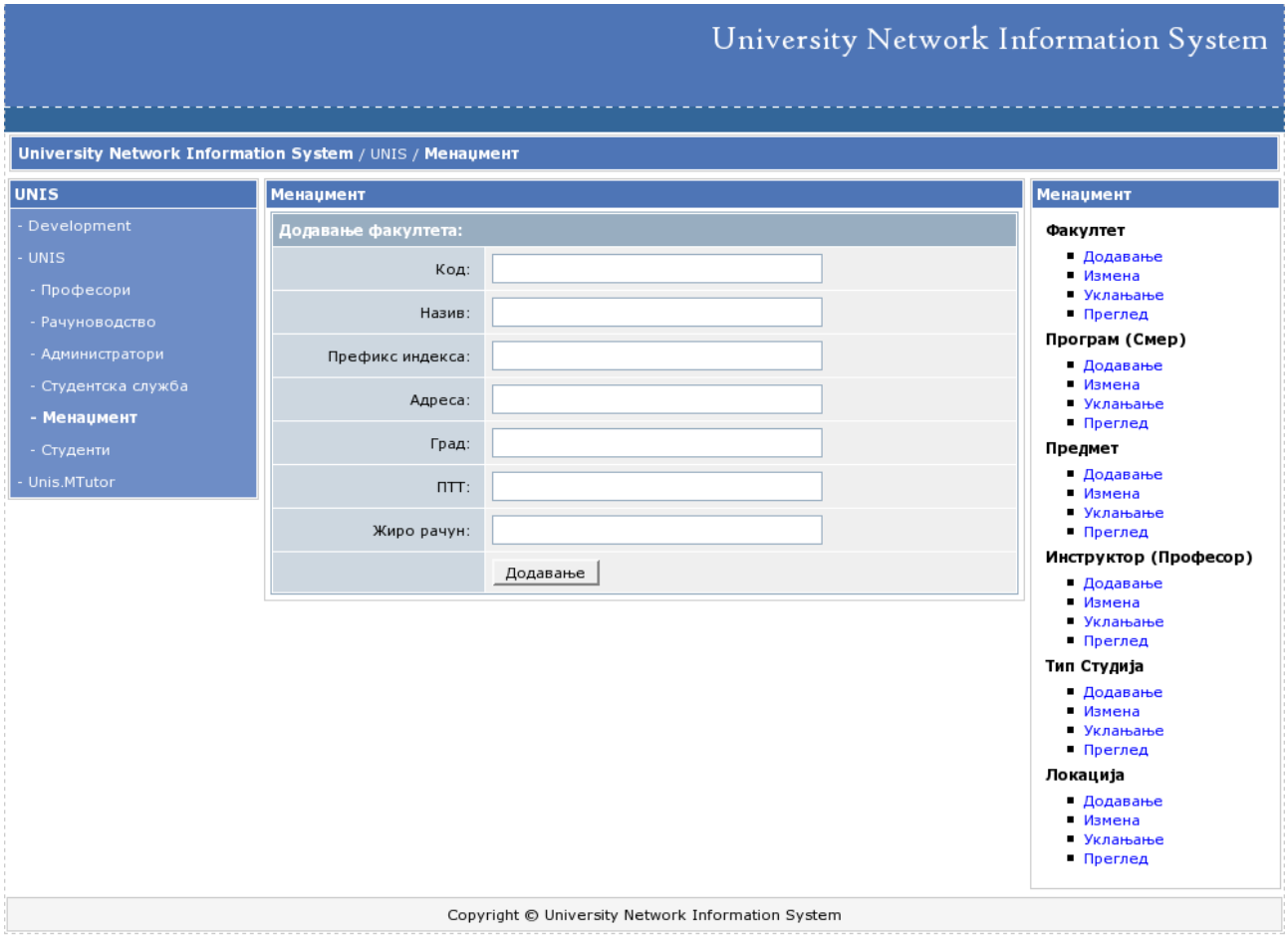
Unutar informacionog sistema fakulteti čine jedinicu za adresiranje:

1. programa (smerova)
2. predmeta
3. instruktora (profesora i asistenata)
4. studenata

U okviru ove grupe funkcionalnosti modula za kategoriju korisnika „Menadžment“ trenutno je realizovana samo funkcionalnost dodavanja novih fakulteta.

5.1.1.1. Dodavanje

Dodavanje novog fakulteta se vrši odabirom opcije „Fakultet/Dodavanje“ unutar modula „Menadžment“. Odabirom ove opcije sistem daje formular u koji je potrebno uneti informacije vezane za fakultet koji se dodaje.



Slika 22 – Formular za dodavanje novog fakulteta u IS.

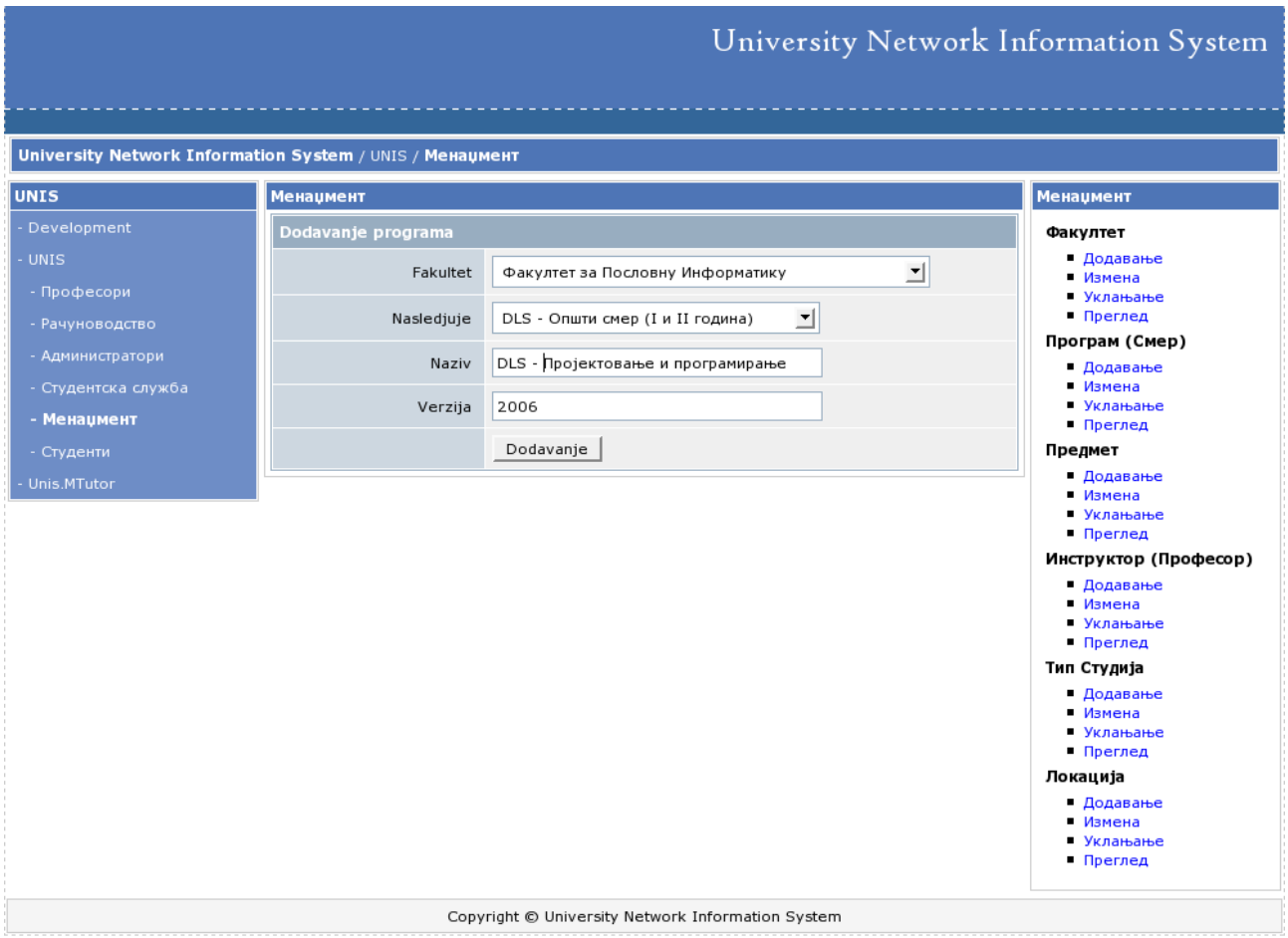
Nakon unošenja parametara fakulteta njegovo dodavanje se inicira aktiviranjem dugmeta „Dodavanje“ nakon čega sistem izbacuje poruku o uspešnom dodavanju ili eventualnoh grešci.

5.1.2. Programi (Smerovi)

Studijski programi (smerovi) u informacionom sistemu čine veznu informaciju između predmeta i studenata. Upis studenata zahteva odabir studijskog programa na kome oni studiraju i na taj način se studentima pridružuje predmeti koje mogu da prijavljuju u ispitnim rokovima.

5.1.2.1. Dodavanje

Dodavanje novog programa (smera) se vrši odabirom opcije „Program (Smer)/Dodavanje“ unutar modula „Menadžment“. Odabirom ove opcije sistem daje formular u koji je potrebno uneti informacije vezane za program koji se dodaje.



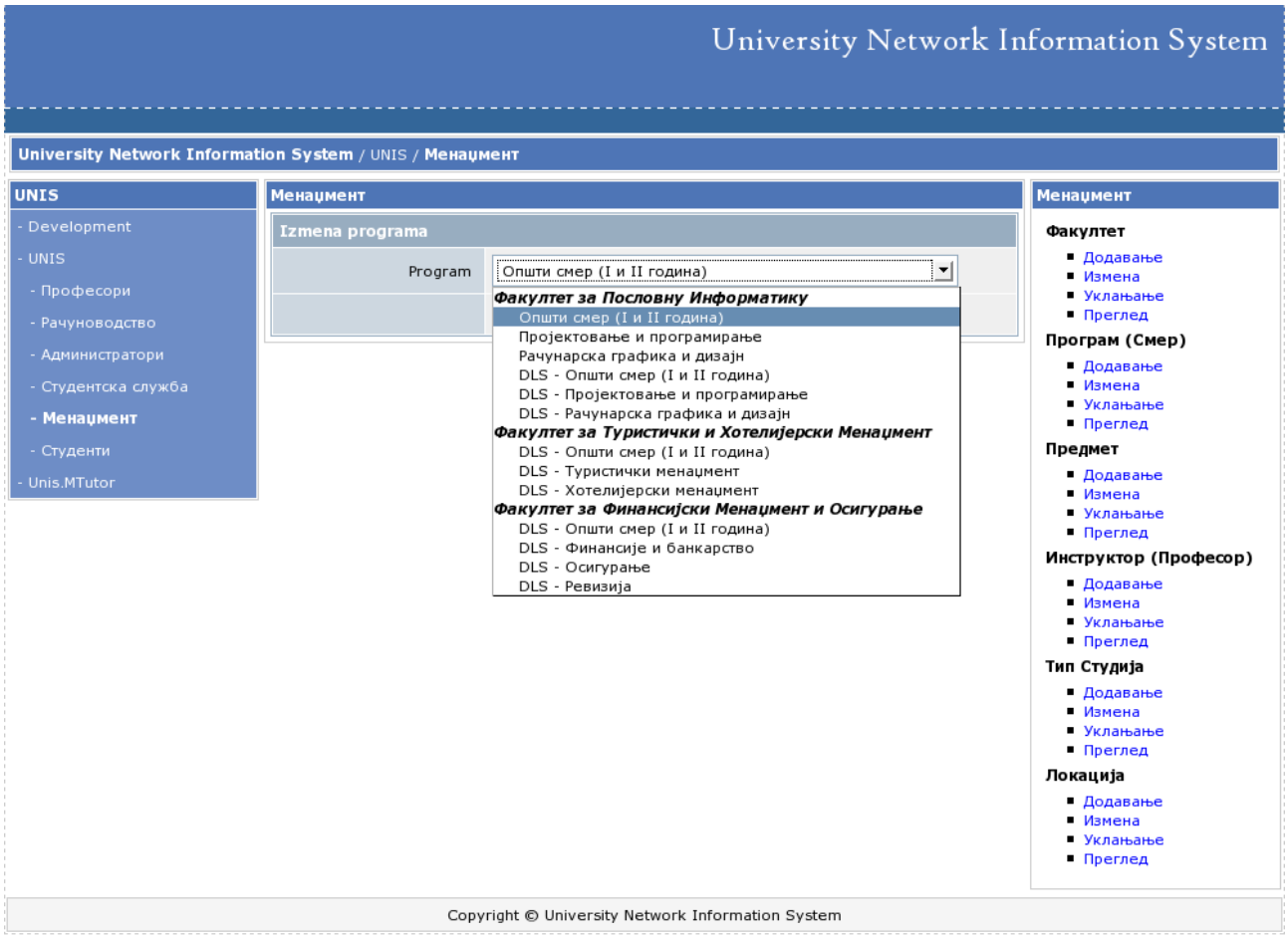
Slika 23 – Formular za dodavanje novog programa (smera)

Jedna od ugrađenih funkcionalnosti koje treba pomenuti jeste i mogućnost „nasleđivanja“ smerova koja novom smeru (npr. 5-8. semestra) omogućava da od već postojećeg smera (npr. 1-4. semestra) nasledi pridružene predmete. Trenutno je omogućeno nasleđivanje najviše jednog smera što odgovara realnoj situaciji na fakultetima univerziteta Singidunum. Verzija programa omogućava unos više smerova istog naziva tokom godina.

Nakon unošenja parametara programa njegovo dodavanje se inicira aktiviranjem dugmeta „Dodavanje“ nakon čega sistem izbacuje poruku o uspešnom dodavanju ili eventualnoh grešci.

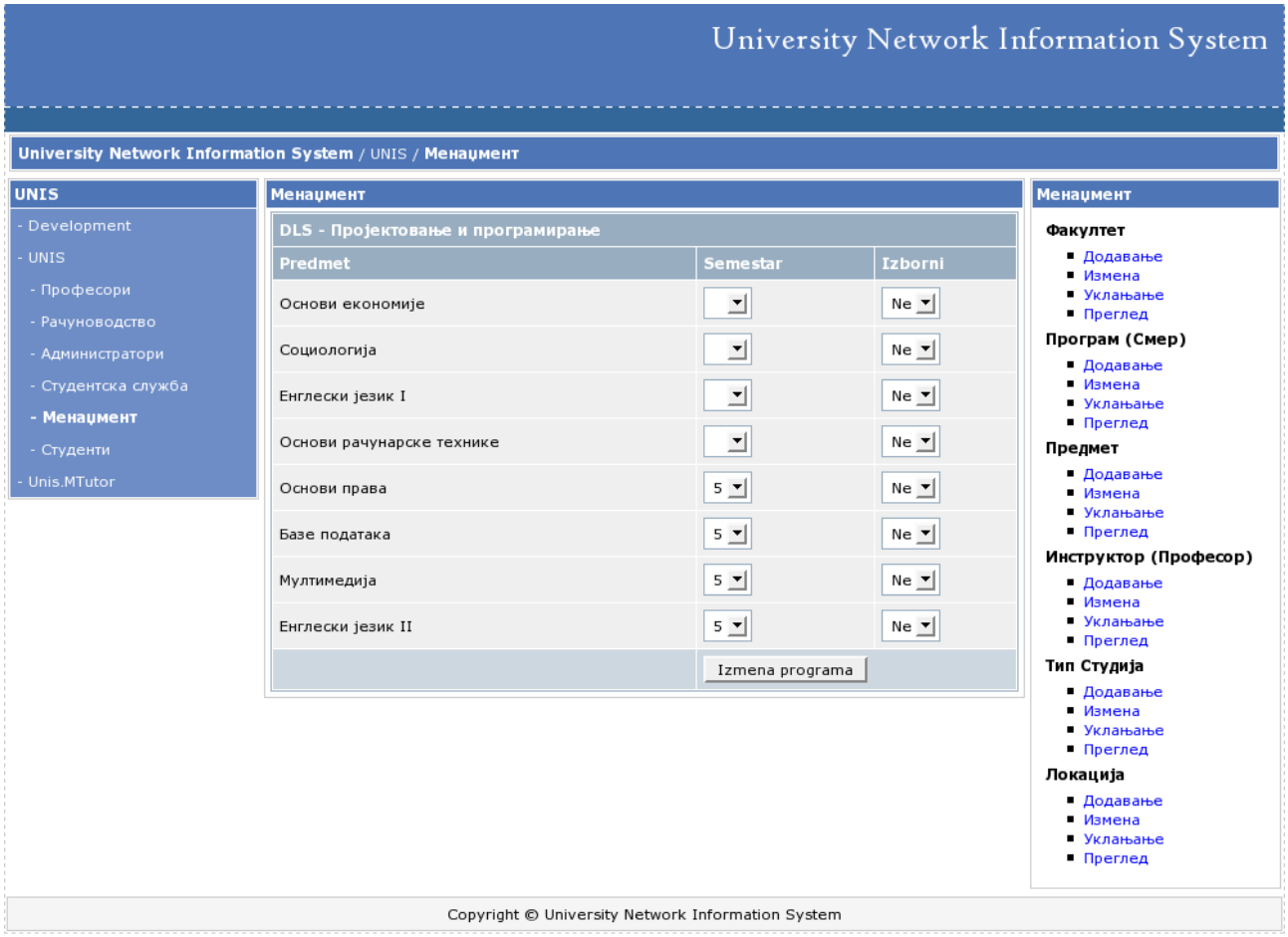
5.1.2.2. Izmena

Izmena studijskog programa se vrši odabirom opcije „Program (Smer)/Izmena“ unutar modula „Menadžment“. Odabirom ove opcije sistem daje formular u kome je potrebno odabrati program koji se će biti izmenjen. U formularu su ponuđeni svi aktivni programi (tj. smerovi nad kojima nije izvršena akcija brisanja). Programi su grupisani po fakultetima kojima pripadaju.



Slika 24 – Odabir studijskog programa za izmenu

Nakon odabira programa koji treba izmeniti aktiviranjem dugmeta „Dalje“ se pristupa formularu za izmeni programa. Formular za izmenu programa sadrži listu svih predmeta na pripadajućem fakultetu. Svaki od predmeta je moguće uključiti u program putem odabira semestra u kome će predmet biti dostupan. Predmeti kod kojih nije odabran semestar nisu pridruženi programu. Dodatna opcija za svaki od predmeta jeste i to da li je predmet izborni. Ova informacija se trenutno ne koristi u ostalim delovima informacionog sistema ali je vrlo verovatno da će biti potrebna nakon dodavanja dodatnih funkcionalnosti u IS.

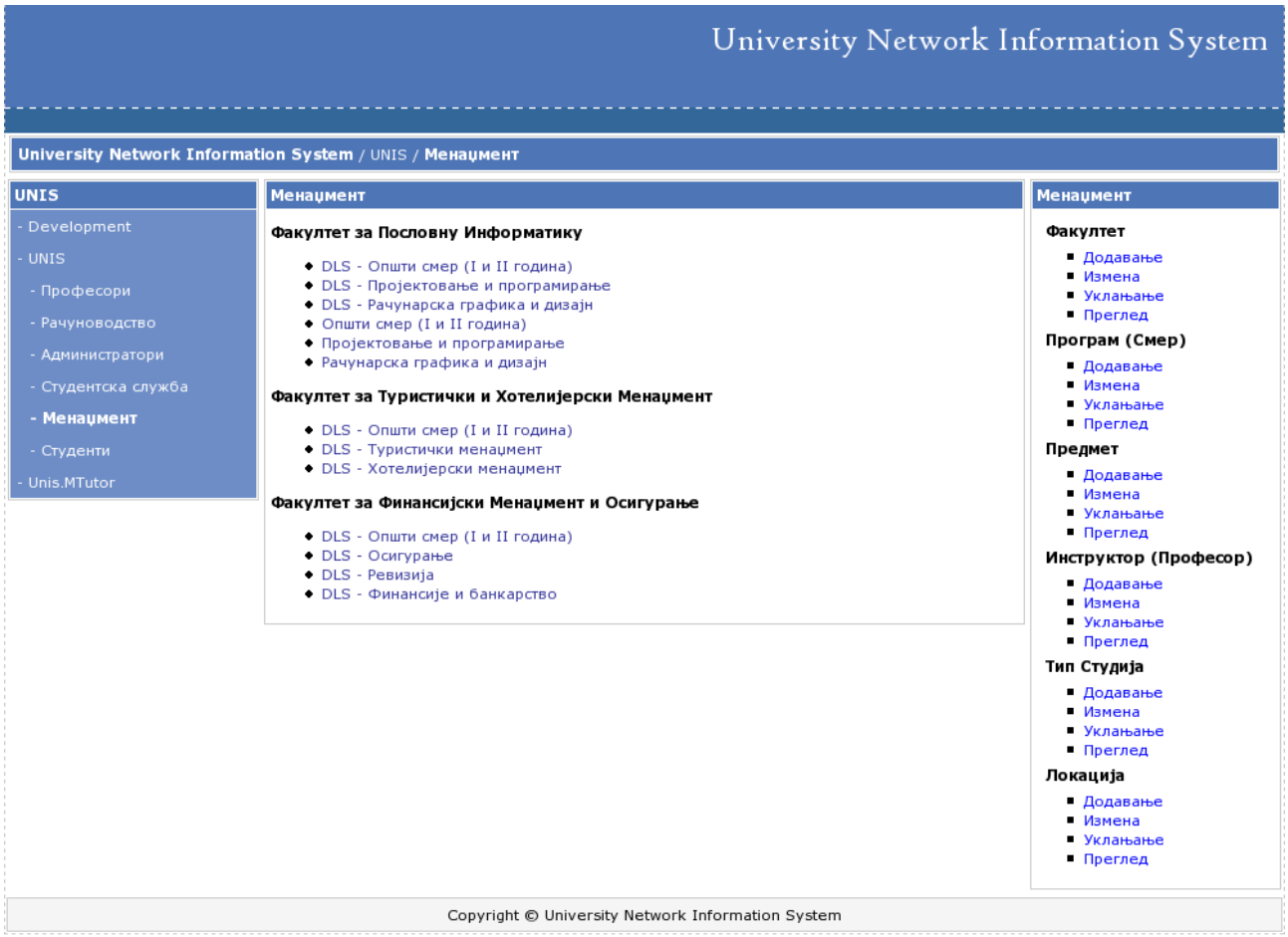


Slika 25 – Primer izmene studijskog programa u modulu menadžmenta

Nakon odabira predmeta koji će biti pridruženi programu izmena se primenjuje aktiviranjem dugmeta „Izmena programa“ nakon čega sistem izbacuje poruku o uspešnoj izmeni ili eventualnoh grešci.

5.1.2.3. Pregled

Pregled studijskih programa se vrši odabirom opcije „Program (Smer)/Pregled“ unutar modula „Menadžment“. Odabirom ove opcije sistem daje listu svih studijskih programa po fakultetima.



Slika 26 – Lista svih studijskih programa

Osim zbirnog pregleda smerova po fakultetima moguće je videti i detalje svakog od smerova. Nakon odabira jednog od smerova sistem prikazuje listu predmeta pridruženih smeru. Dodatno, sistem daje tabelarni prikaz semestara po predmetima.

University Network Information System

University Network Information System / UNIS / Менаџмент

UNIS
- Development
- UNIS
  - Професори
  - Рачуноводство
  - Администратори
  - Студентска служба
  - **Менаџмент**
  - Студенти
- Unis.MTutor

Менаџмент

DLS - Пројектовање и програмирање

Рб.	Предмет/Семестар	I	II	III	IV	V	VI	VII	VIII
1	Основи економије								
2	Социологија								
3	Енглески језик I								
4	Основи рачунарске технике								
5	Основи права								
6	Базе података								
7	Мултимедија								
8	Енглески језик II								

Менаџмент

Факултет
- Додавање
- Измена
- Уклањање
- Преглед

Програм (Смер)
- Додавање
- Измена
- Уклањање
- Преглед

Предмет
- Додавање
- Измена
- Уклањање
- Преглед

Инструктор (Професор)
- Додавање
- Измена
- Уклањање
- Преглед

Тип Студија
- Додавање
- Измена
- Уклањање
- Преглед

Локација
- Додавање
- Измена
- Уклањање
- Преглед

Copyright © University Network Information System

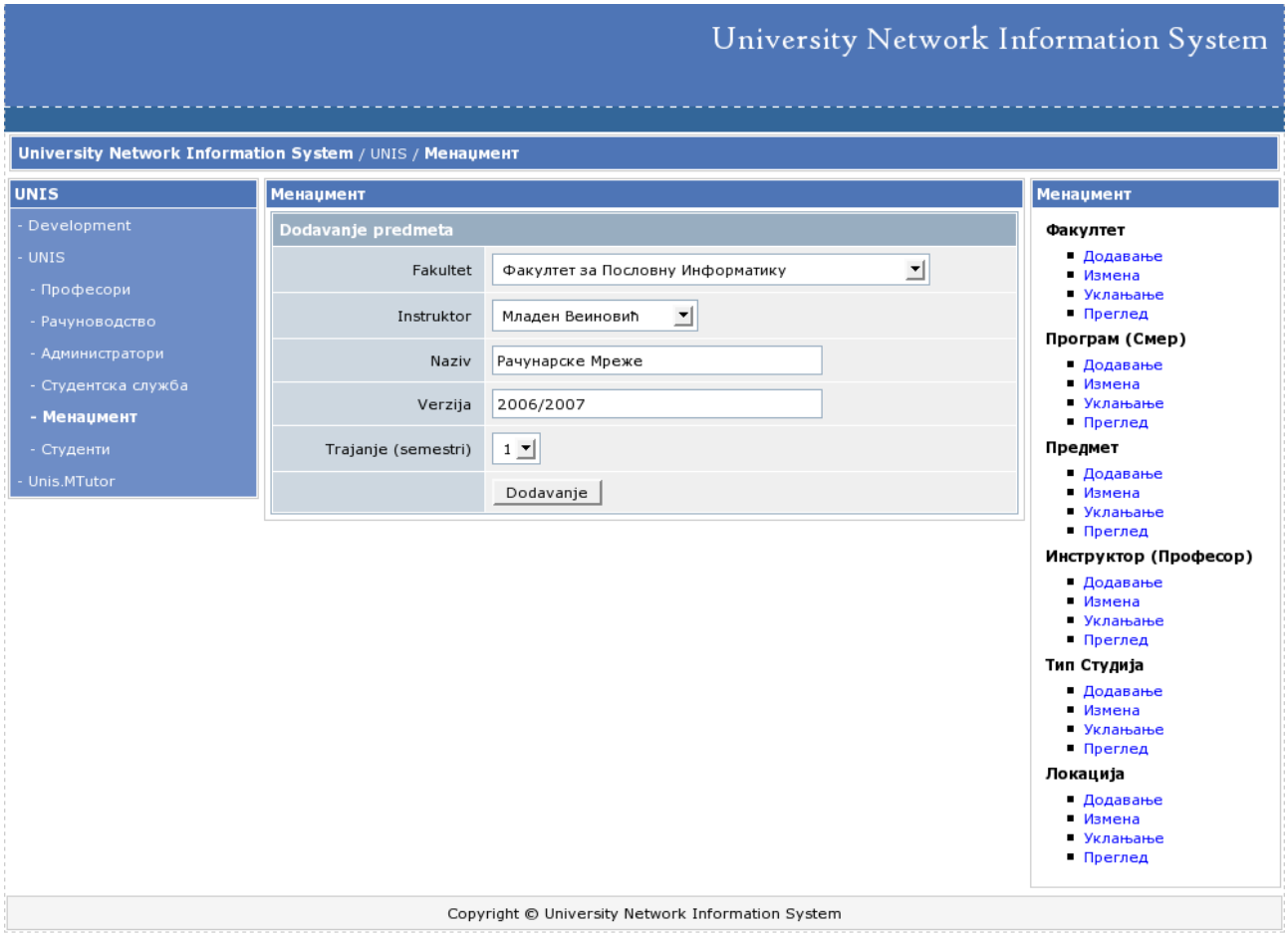
Slika 27 – Primer pregleda studijskog programa

5.1.3. Predmeti

Informacije o predmetima u informacionom sistemu se trenutno koriste u vezi sa ispitima u ispitnim rokovima. Realizovane funkcionalnosti vezane za predmete jesu dodavanje i pregled.

5.1.3.1. Dodavanje

Dodavanje novih predmeta u informacionom sistemu se vrši odabirom odabirom opcije „Predmet/Dodavanje“ unutar modula „Menadžment“. Odabirom ove opcije sistem daje formular u koji je potrebno uneti informacije vezane za predmet koji se dodaje.

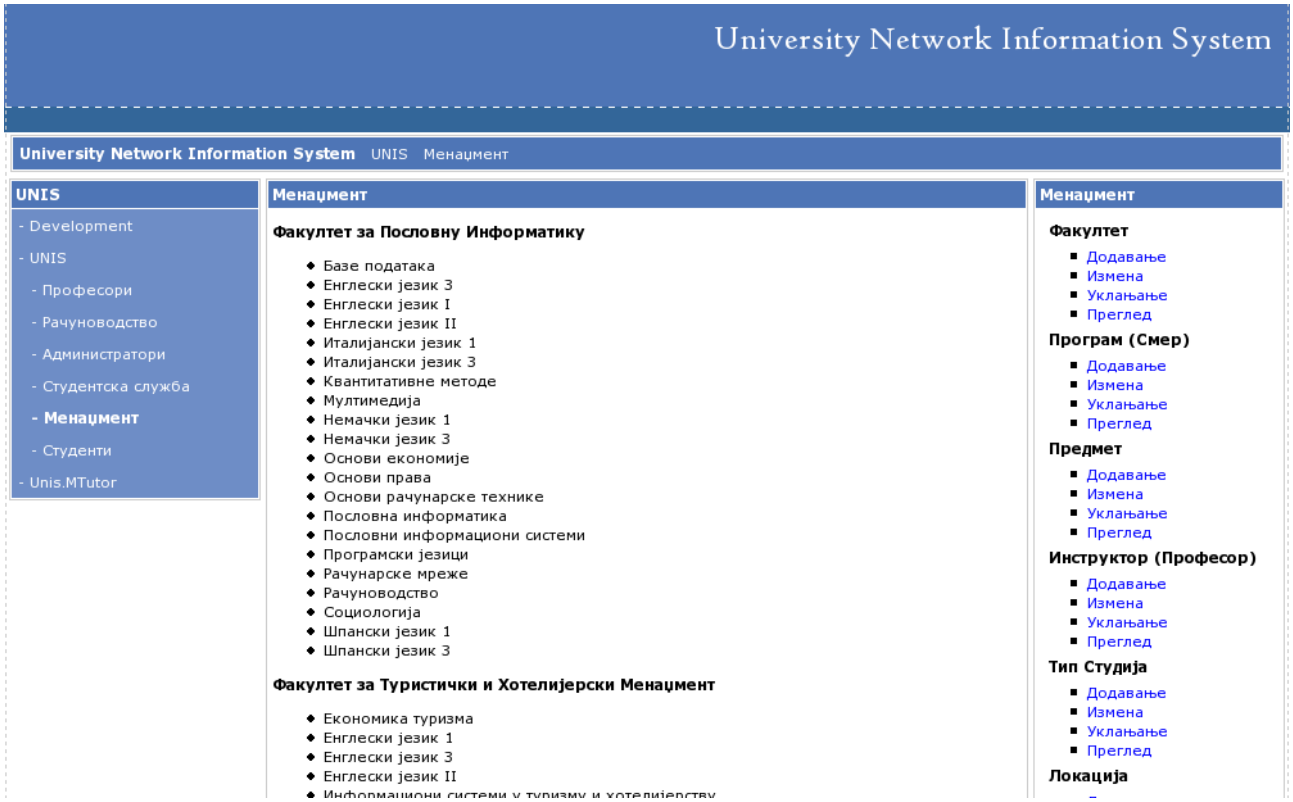


Slika 28 – Primer dodavanja novog predmeta

Predmet je neohpodno pridružiti određenom fakultetu i profesoru a moguć je i unos predmeta sa istim nazivom ali različitim verzijama. Za predmet je, takođe, potrebno definisati i broj semestara koji je potreban za njegovo održavanje.

5.1.3.2. Pregled

Pregled postojećih predmeta u informacionom sistemu se vrši odabirom odabirom opcije „Predmet/Pregled“ unutar modula „Menadžment“. Odabirom ove opcije sistem daje listu svih definisanih predmeta grupisanih po fakultetima kojima pripadaju.



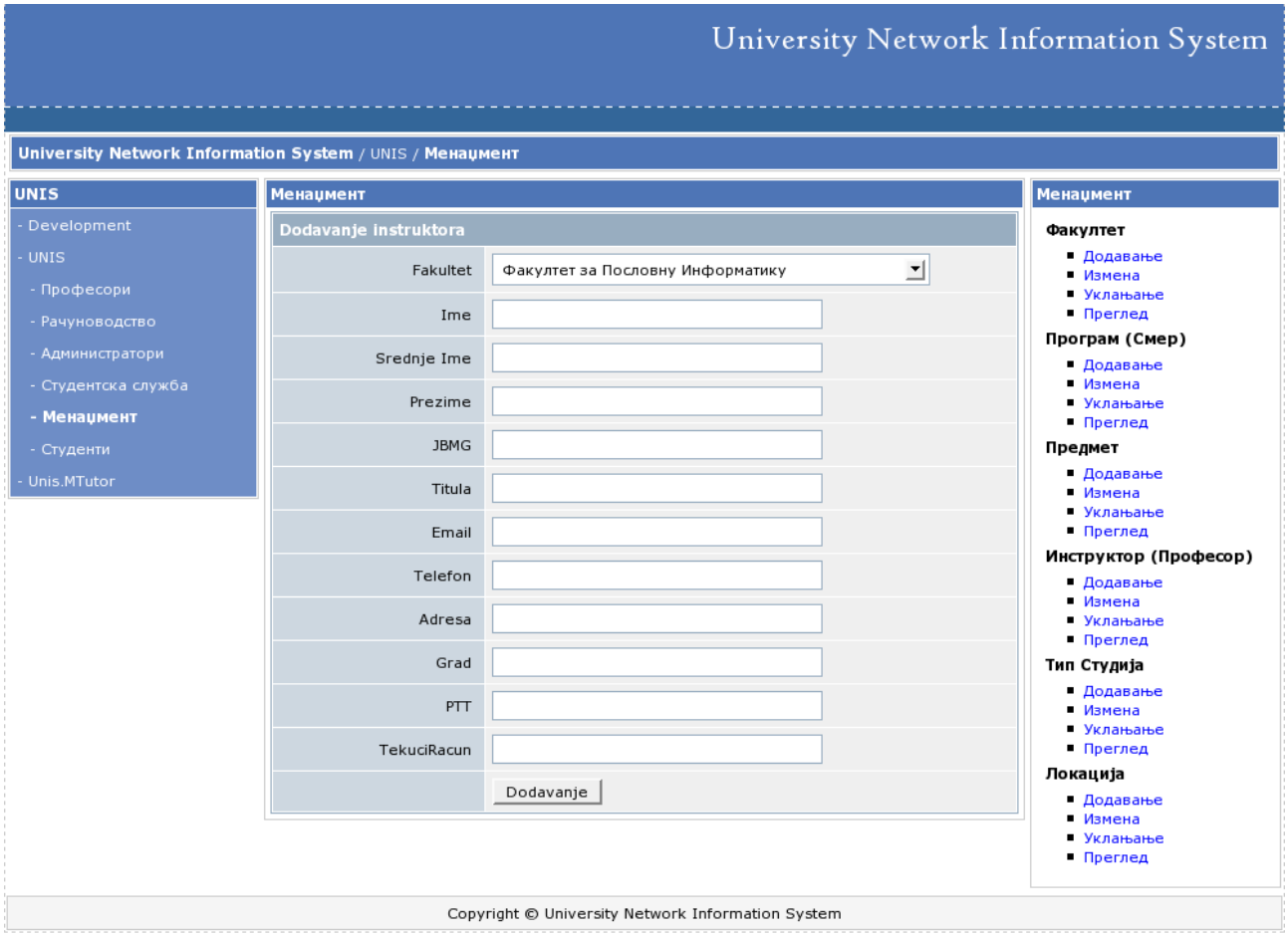
Slika 29 - Pregled postojećih predmeta u informacionom sistemu

5.1.4. Instruktori

Pod instruktorima se u informacionom sistemu podrazumevaju sva lica vezana za izvođenje nastave (profesori i asistenti). Trenutno nema funkcionalnosti unutar informacionih sistema koje koriste razliku statusa (titule) instruktora. Nad podacima tipa „instruktor“ su realizovane funkcionalnosti dodavanja i pregleda.

5.1.4.1. Dodavanje

Dodavanje novih instruktora u informacionom sistemu se vrši odabirom odabirom opcije „Instruktor/Dodavanje“ unutar modula „Menadžment“. Odabirom ove opcije sistem daje formular u koji je potrebno uneti informacije vezane za instruktora koji se dodaje.

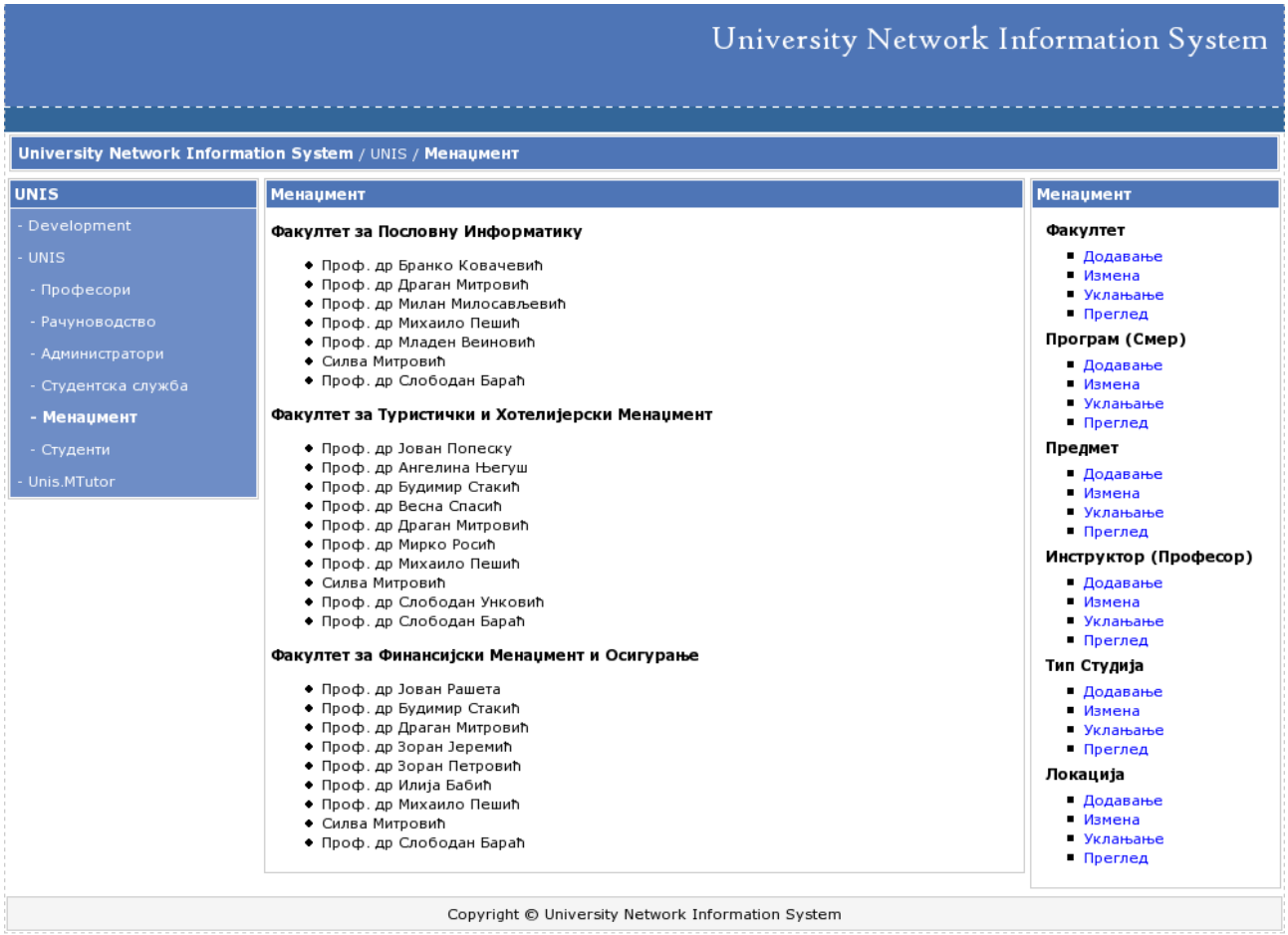


Slika 30 - Obrazac za dodavanje instruktora

Za instruktora koji se dodaje je potrebno odabrati fakultet kome pripada i uneti odgovarajuće parametre. Trenutno se u IS koriste samo parametri „Ime“ i „Prezime“ a ostali parametri su predviđeni za buduću upotrebu (npr. parametar „TekuciRacun“ za potrebe modula „Računovodstvo“).

5.1.4.2. Pregled

Pregled postojećih instruktora u informacionom sistemu se vrši odabirom odabirom opcije „Instruktor/Pregled“ unutar modula „Menadžment“. Odabirom ove opcije sistem daje listu svih definisanih instruktora grupisanih po fakultetima kojima pripadaju.



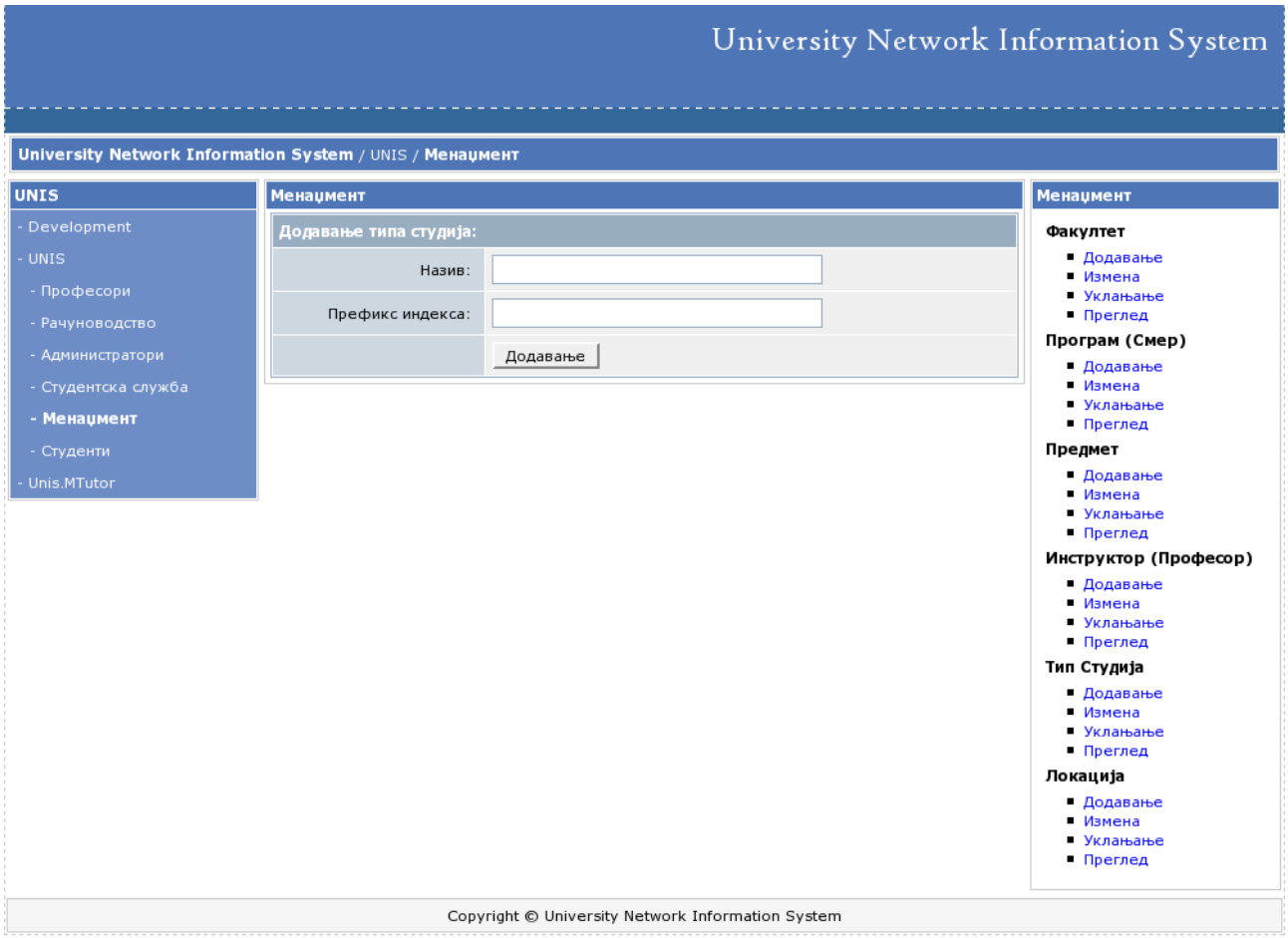
Slika 31 – Lista instruktora po fakultetima

5.1.5. Tipovi studija

Tipovi studija u okviru informacionog sistema služe kao atribut studenata koji označava na koji način studenti studiraju (klasično studiranje, DLS...). Odabrani tip studija studentskom indeksu dodaje odgovarajući prefiks.

5.1.5.1. Dodavanje

Dodavanje novih tipova studiranja u informacionom sistemu se vrši odabirom odabirom opcije „Tip studija/Dodavanje“ unutar modula „Menadžment“. Odabirom ove opcije sistem daje formular u koji je potrebno uneti informacije vezane za tip studija koji se dodaje.



Slika 32 - Formular za dodavanje novog tipa studija

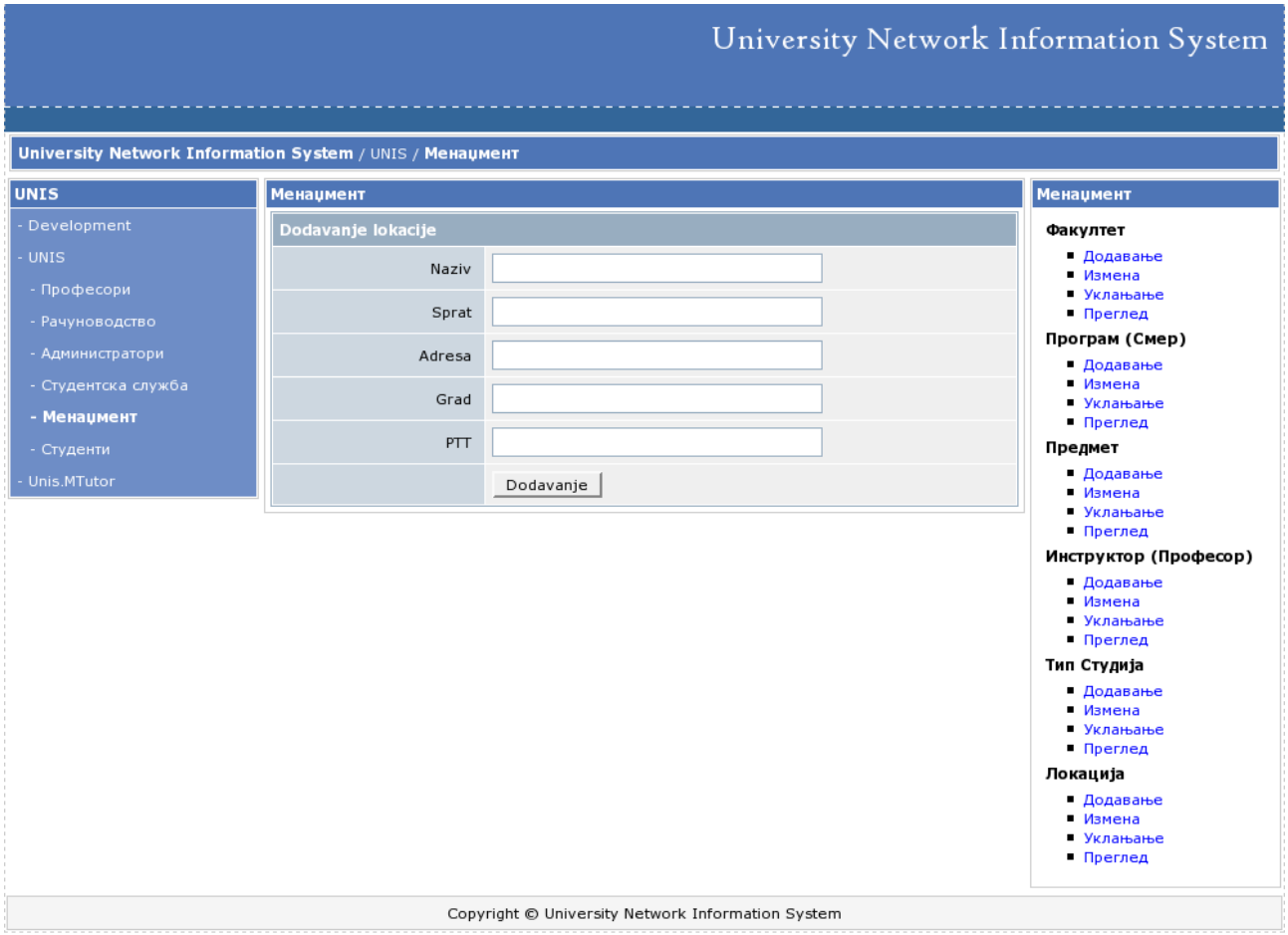
Naziv studija se odnosi na simboličko ime tipa koji se unosi. Prefiks indeksa je parametar koji se koristi u indeksu studenata koji na taj način studiraju. Prefiksi indeksa svih tipova studija moraju biti različiti a prazna vrednost se odnosi na klasični način studiranja.

5.1.6. Lokacije

Informacije o lokacijama se unutar informacionog sistema koriste za određivanje prostorije u kojoj će se održati određeni ispit. U skladu sa dodavanjem novih funkcionalnosti očekuju se dodatne primene (npr. određivanje prostorije u kojoj će se držati predavanja/vežbe u rasporedu predavanja i sl.).

5.1.6.1. Dodavanje

Dodavanje novih lokacija u informacionom sistemu se vrši odabirom odabirom opcije „Lokacija/Dodavanje“ unutar modula „Menadžment“. Odabirom ove opcije sistem daje formular u koji je potrebno uneti informacije vezane za lokaciju koji se dodaje.

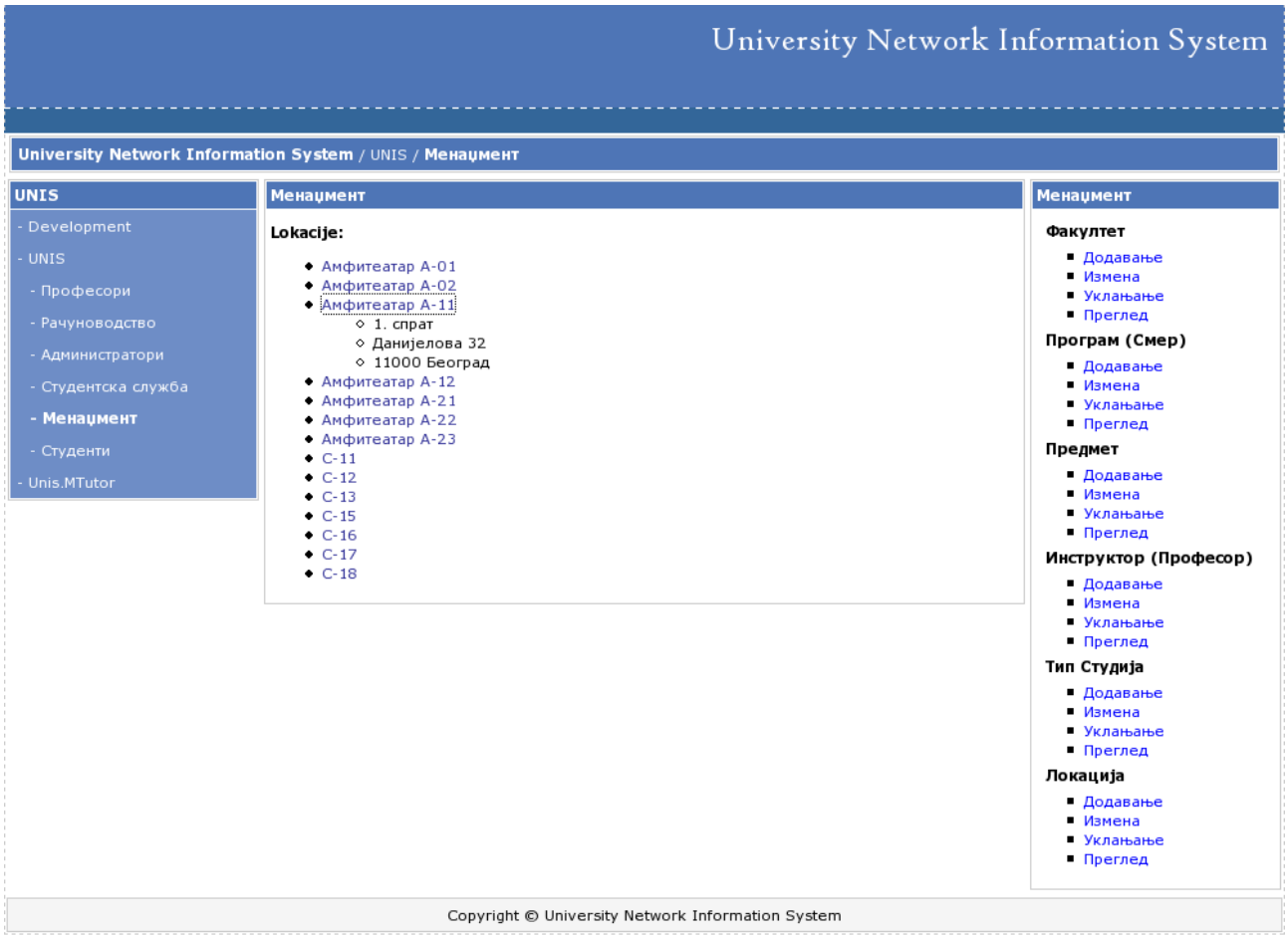


Slika 33 – Dodavanje nove lokacije u modulu menadžmenta

Svi parametri nove lokacije (osim naziva) su opcioni i koriste se isključivo za vizuelnu reprezentaciju.

5.1.6.2. Pregled

Pregled postojećih lokacija u informacionom sistemu se vrši odabirom odabirom opcije „Lokacija/Pregled“ unutar modula „Menadžment“. Odabirom ove opcije sistem daje listu svih definisanih lokacija (s obzirom na to da su lokacije adresirane na nivou univerziteta, svi fakulteti univerziteta mogu koristiti sve lokacije). Odabirom određene lokacije lista se proširuje njenim parametrima.



Slika 34 – Lista lokacija sa opcionim detaljnim prikazom

5.2. Studentske službe

Unutar softverskog interfejsa (modula) namenjenog studentskim službama realizovane su sledeće funkcionalnosti:

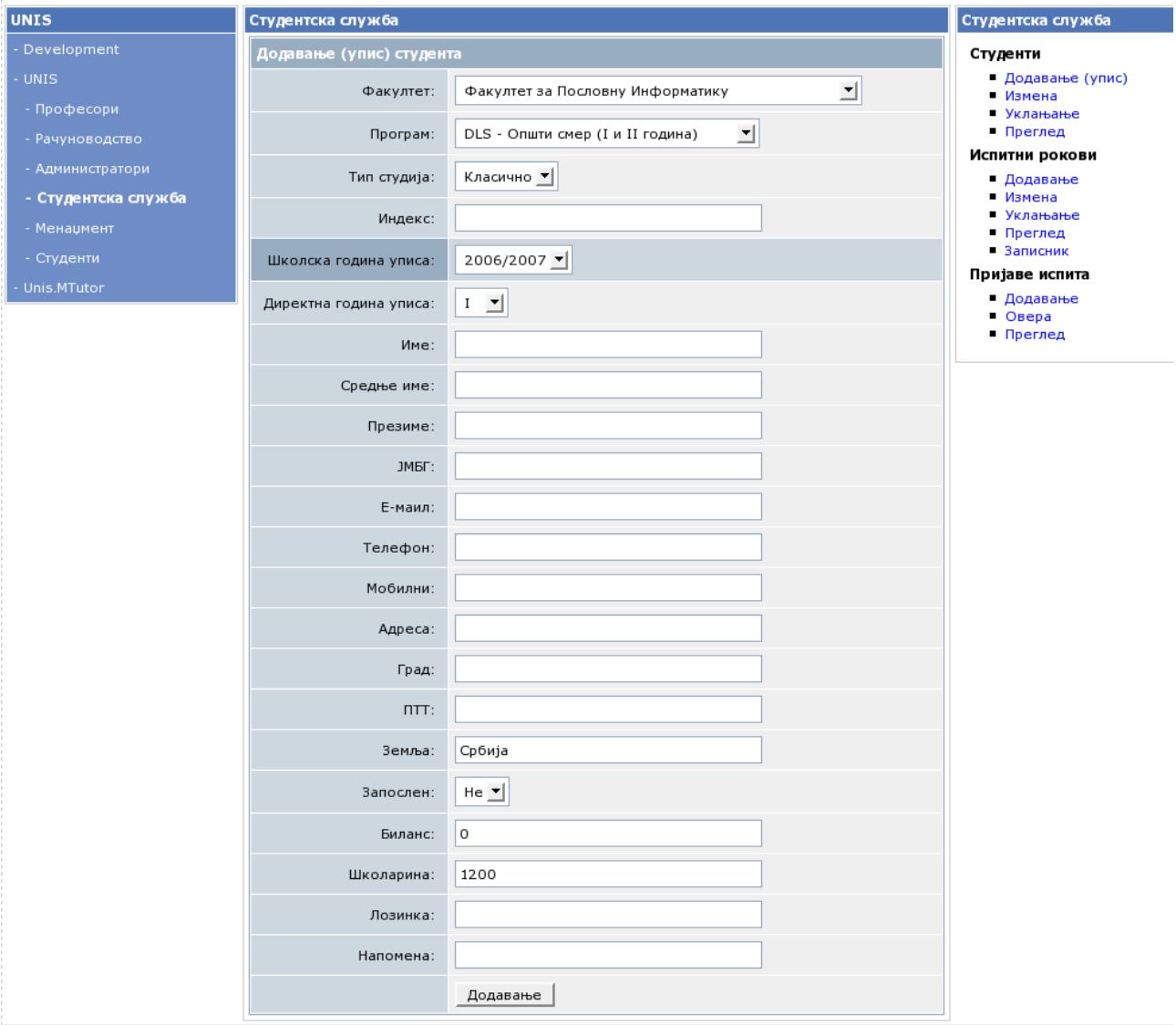
- Studenti
  - Dodavanje
  - Uklanjanje
  - Pregled
- Ispitni rokovi
  - Dodavanje
  - Izmena
  - Uklanjanje
  - Pregled
  - Zapisnik
- Prijave
  - Dodavanje
  - Overa
  - Pregled

5.2.1. Studenti

Trenutno podržane operacije nad podacima vezanim za studente univerziteta su dodavanje uklanjanje i pregled. Operacija izmene trenutno nije realizovana a treba pomenuti da uklanjanje studenata iz baze nije moguće već je moguća samo deaktivacija studentskog naloga.

5.2.1.1. Dodavanje

Dodavanje novih studenata u informacionom sistemu se vrši odabirom odabirom opcije „Studenti/Dodavanje“ unutar modula „Studentske službe“. Odabirom ove opcije sistem daje formular u koji je potrebno uneti informacije vezane za novog studenta.



Slika 35 – Formular za dodavanje (upis) novih studenata

Polje formulara koje je specifično jeste „Indeks“ koje se odnosi na redni broj indeksa studenta. U ovo polje je moguće uneti željenu vrednost (koja mora biti jedinstvena za taj fakultet, godinu upisa, direktnu godinu upisa i tip studiranja). U slučaju praznog polja sistem uzima prvu sledeću vrednost od najveće unete za taj fakultet, godinu upisa, direktnu godinu upisa i tip studiranja.

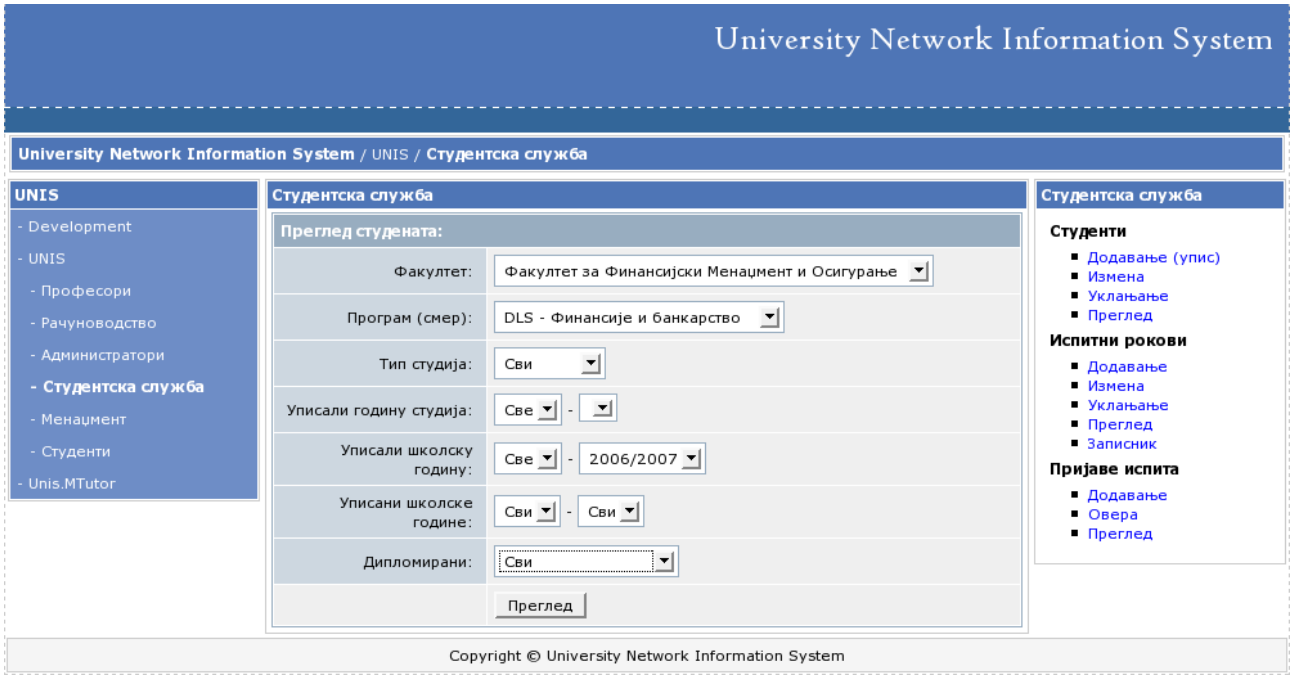
5.2.1.2. Uklanjanje

Kao što je ranije objašnjeno, uklanjanje zapisa o studentim iz baze nije moguće. Razlog ovome je mogućnost deaktivacije studentskog naloga i sprečavanje slučajnog uklanjanja zapisa. S obzirom na to da je zapis u tabeli „Student“ ne prelazi 1KB, ne postoji mogućnost preopterećivanja baze deaktiviranim nalozima.

Slika 35 – Poruka pri odabiru opcije uklanjanja studenata.

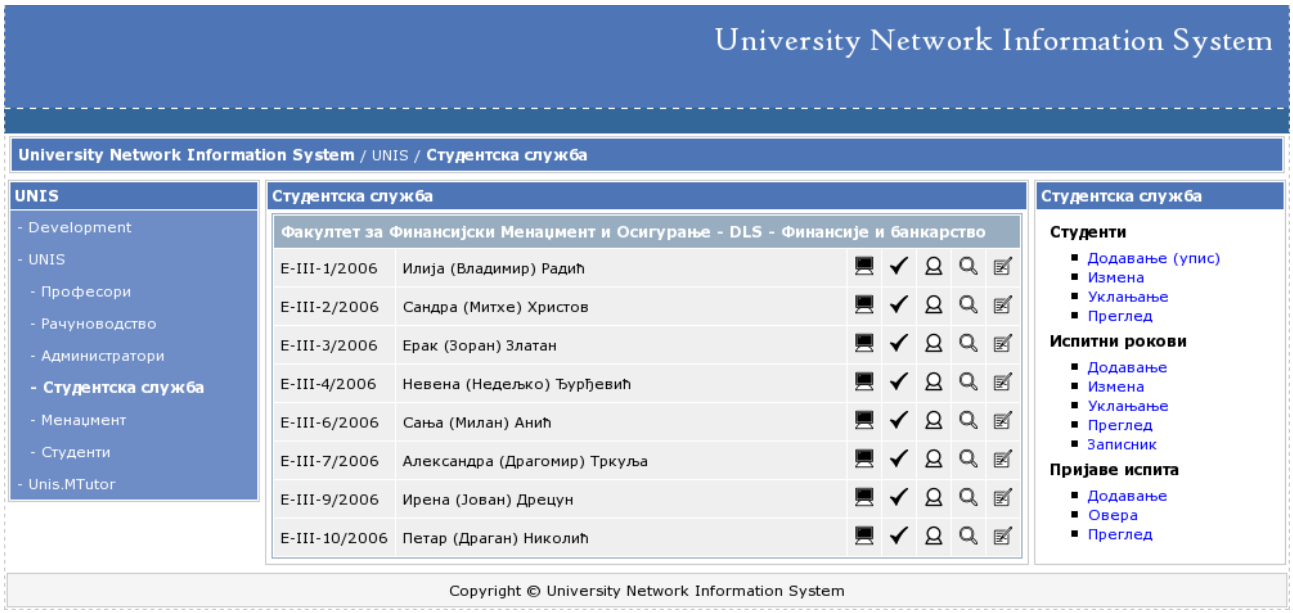
5.2.1.3. Pregled

Pregled studenata u informacionom sistemu se vrši odabirom odabirom opcije „Studenti/Pregled“ unutar modula „Studentske službe“. Odabirom ove opcije sistem daje formular u koji je potrebno uneti kriterijume vezane za prikaz studenata.



Slika 37 – Formular za postavljanje kriterijuma kod pregleda studenata

Nakon odabira kriterijuma sistem prikazuje studente koji ih ispunjavaju.



Slika 38 – Lista studenata

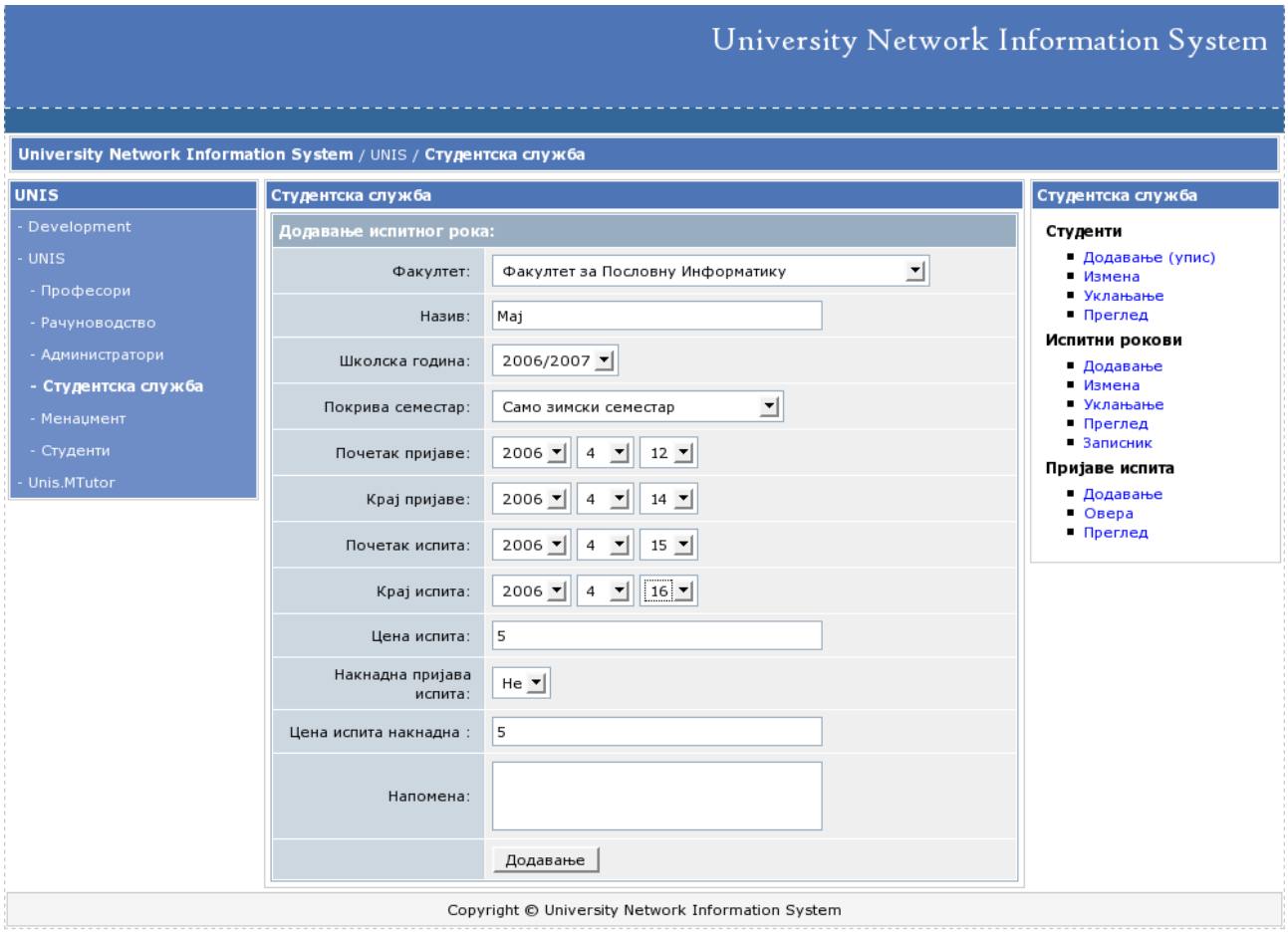
Lista studenata pored broja indeksa i imena i prezimena sadrži i vizuelnu informacijo o tipu studiranja, upisu godine, statusu diplomiranja, link ka detaljnim parametrima i njihovoj izmeni.

5.2.2. Ispitni rokovi

Ispitni rokovi u informacionom sistemu daju mogućnost studentima da izvrše prijavu ispita za predmete koji su definisani u ispitnom roku. Jedna od najvažnijih operacija vezanih za ispitne rokove jeste njihova izmena koja daje dodatna ograničenja (o kojima će kasnije biti reči).

5.2.2.1. Dodavanje

Dodavanje ispitnih rokova u informacionom sistemu se vrši odabirom odabirom opcije „Ispitni rokovi/Pregled“ unutar modula „Studentske službe“. Odabirom ove opcije sistem daje formular u koji je potrebno uneti parametre novog ispitnog roka.

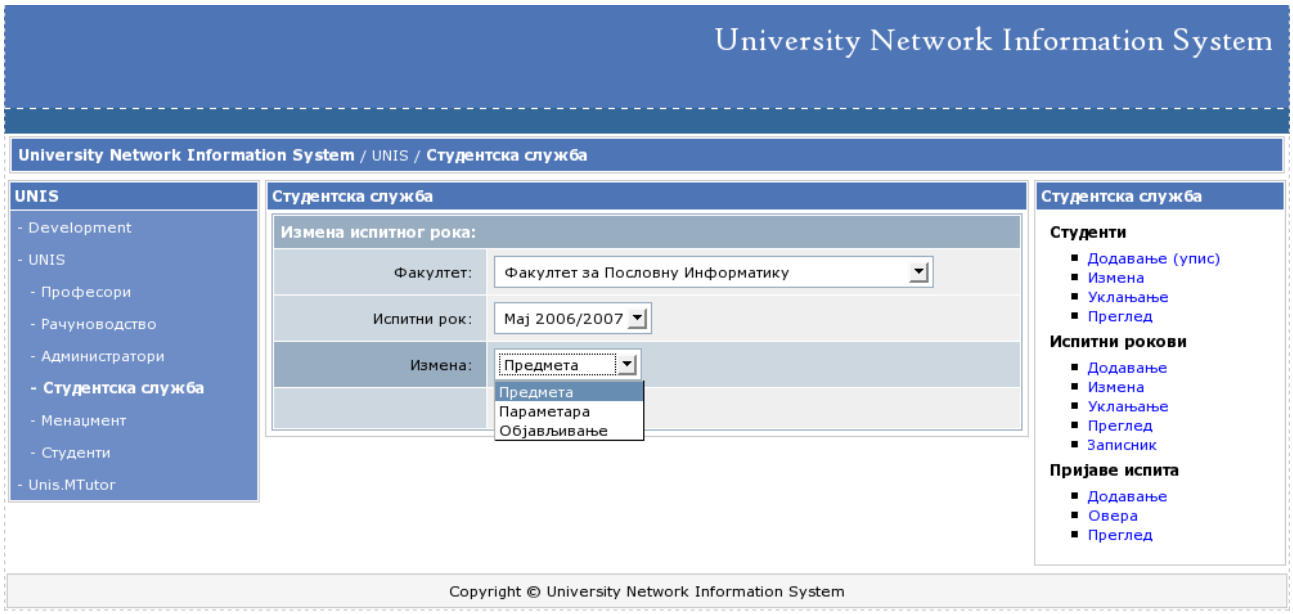


Slika 39 – Formular za dodavanje novog ispitnog roka

Parametar fakultet se odnosi na fakultet kome ispitni rok pripada a bitniji parametri svakog roka jesu: pokriva semestar (odnosi se na to da li je moguće prijaviti ispite zimskog ili i letnjeg semestra), period prijave (period u kome je studentima omogućeno prijavljivanje ispita), period ispita (period unutar kog je moguće zakazivanje ispita) i naknadna prijava (omogućava prijavljivanje ispita nakon isteka osnovnog perioda a pre termina održavanja ispita).

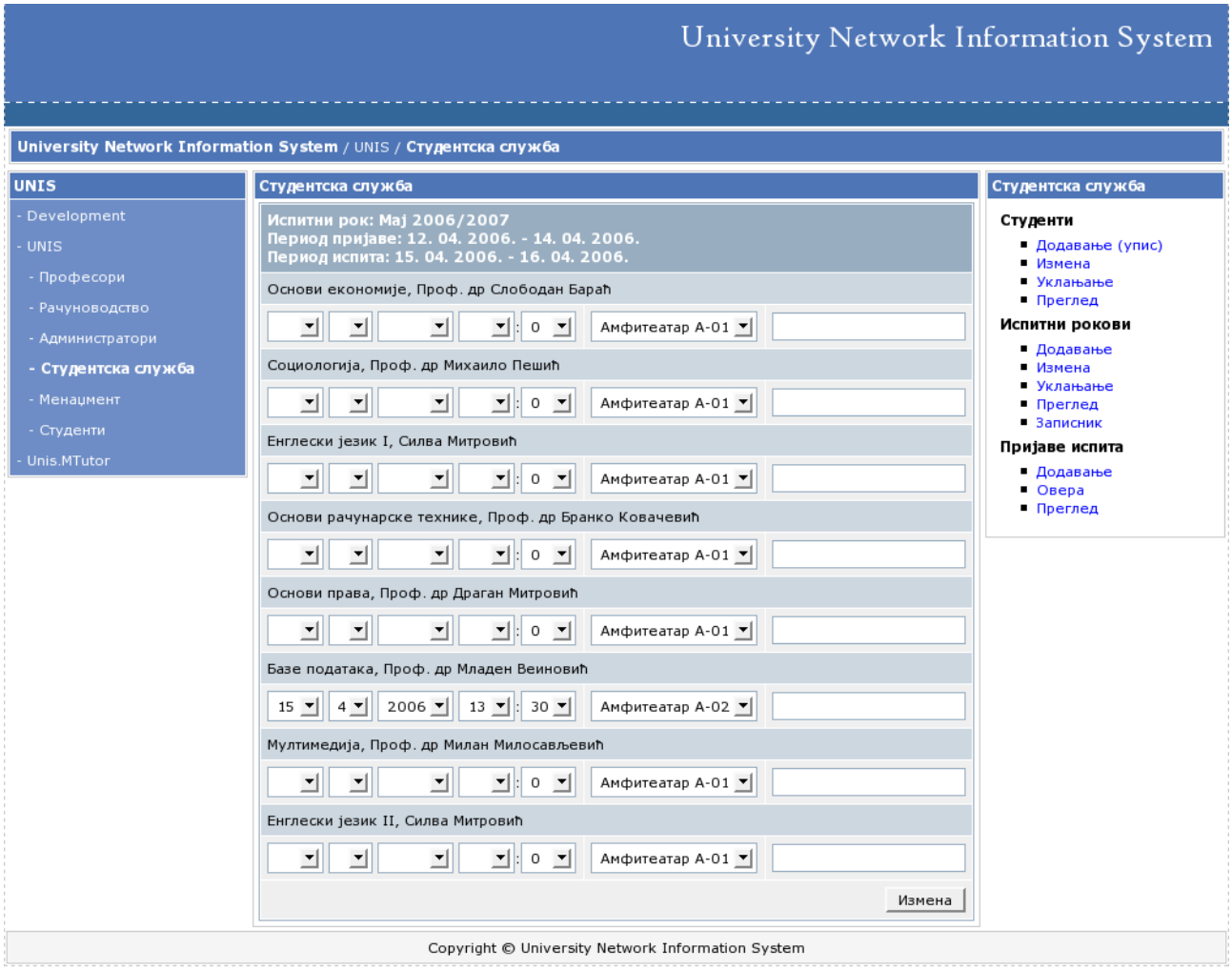
5.2.2.2. Izmena

Nakon dodavanja ispitnog roka potrebno ga je izmeniti tj. definisati ostale parametre (termine ispita iz predmeta) a nakon toga ga objaviti (omogućiti studentima da izvrše prijavu). Izmena ispitnog roka se vrši odabirom odabirom opcije „Ispitni rokovi/Izmena“ unutar modula „Studentske službe“. Nakon odabira ove opcije sistem nudi formular za odabir ispitnog roka nad kojim se vrši izmena.



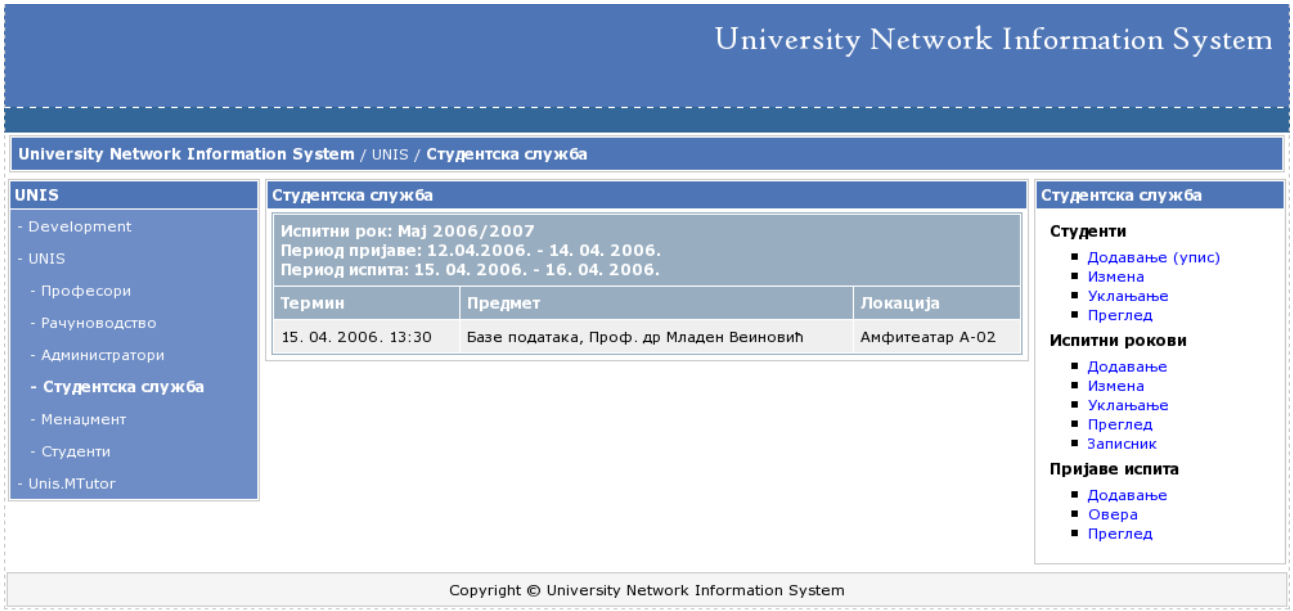
Slika 40 – Formular za odabir ispitnog roka i tipa izmene

Sa formulara se, osim odabira ispitnog roka preko fakulteta, vrši i odabir tipa izmene. Izmena ispitnog roka po pitanju predmeta se odnosi na određivanje termina ispita za definisane predmete fakulteta.



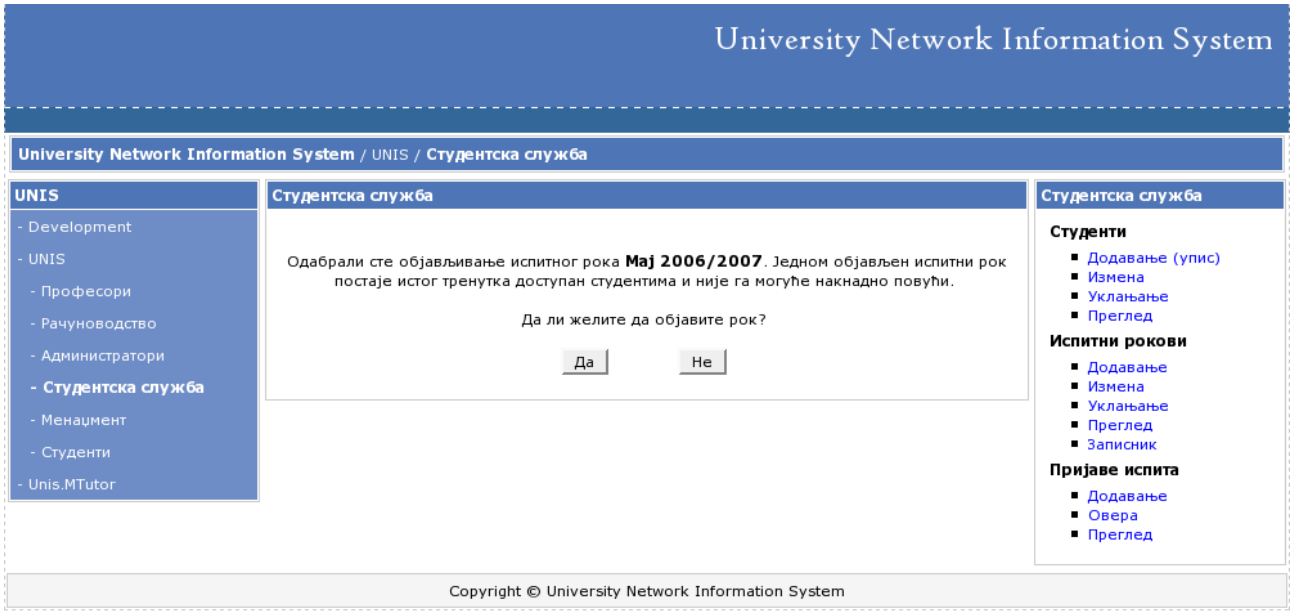
Slika 41 – Formular za izmenu termina i lokacija ispita u ispitnom roku

Odabirom izmene ispitnog roka po pitanju predmeta sistem nudi formular u kome je za svaki od predmeta moguće definisati termin i lokaciju održavanja ispita. Izmenu predmeta (ispita) je moguće vršiti neograničen broj puta ali pre objavljivanja roka. Nakon objavljivanja roka izmenu rasporeda ispita nije moguće vršiti usled mogućnosti da su određeni studenti već izvršili prijavu i da o izmeni neće biti obavešteni. Nakon odabira termina i lokacija za željene predmete i primenom izmene aktiviranje dugmeta „Dalje“, sistem prikazuje novi raspored ispita za odabrani ispitni rok.



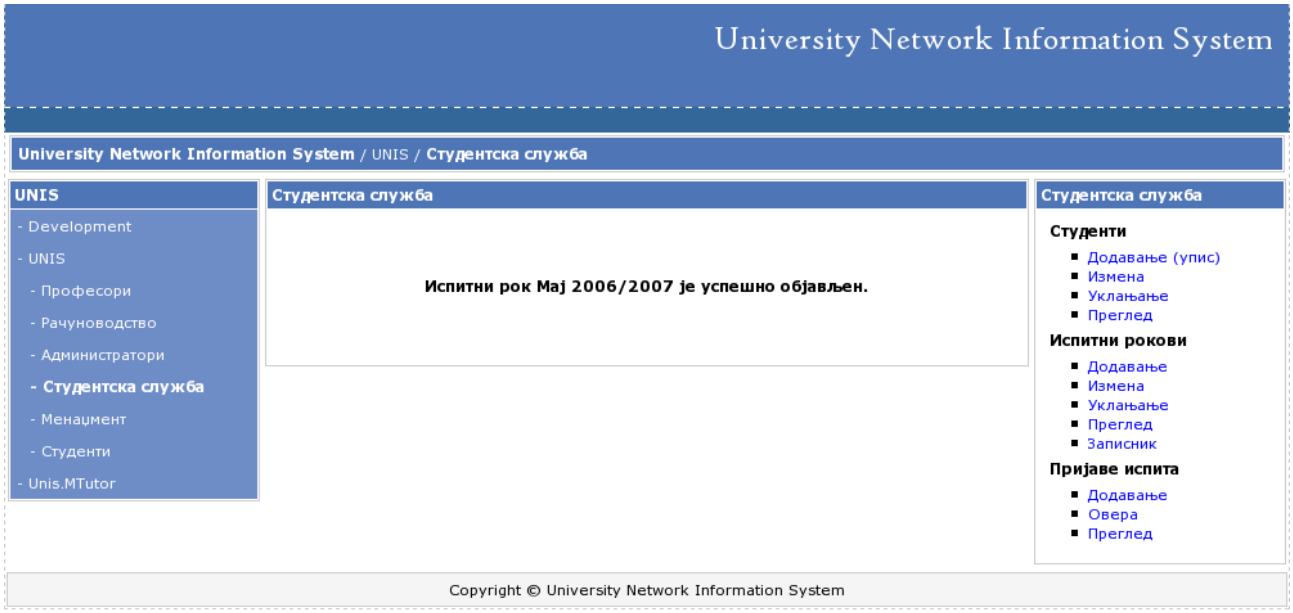
Slika 42 – Tabela sa listom odabranih termina i lokacija po ispitu

Osim mogućnosti za izmenu predmeta ispitnog roka, sledeća moguća izmena jeste i „objavljivanje“ ispitnog roka. Samo definisanje ispitnog roka i rasporeda ispita u okviru njega studentima neće omogućiti prijavu ispita sve dok se ispitni rok ne objavi. Odabirom izmene tipa „objavljivanje“ sistem obaveštava operatera o nemogućnosti naknadne izmene bilo kojih parametara i traži potvrdu objavljivanja.



Slika 43 – Zahtev za potvrdom objavljivanja ispitnog roka

Nakon dobijanja potvrde od strane operatera sistem ispitni rok označava objavljenim (postaje dostupan studentima odmah tj. u prvoj sekundi perioda prijave) i o tome daje odgovarajuću poruku:



Slika 44 – Poruka o uspešnom objavljivanju ispitnog roka.

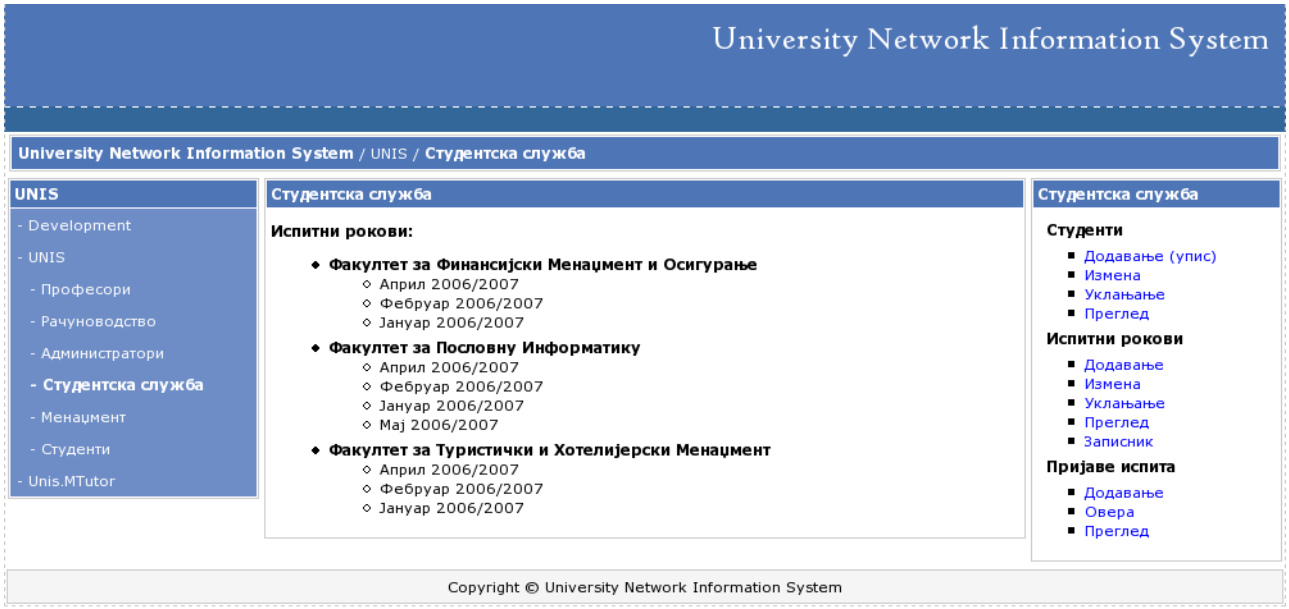
5.2.2.3. Uklanjanje

Uklanjanje ispitnih rokova trenutno nije omogućeno u informacionom sistemu usled opasnosti da greškom budu uklonjeni ispitni rokovi u kojima su studenti položili određene predmete. U skladu sa tim, pri odabiru opcije „Ispitni rokovi/Uklanjanje“ sistem izbacuje poruku o nemogućnosti obavljanja akcije uklanjanja.

Slika 45 – Poruka pri pokušaju uklanjanja ispitnog roka.

5.2.2.4. Pregled

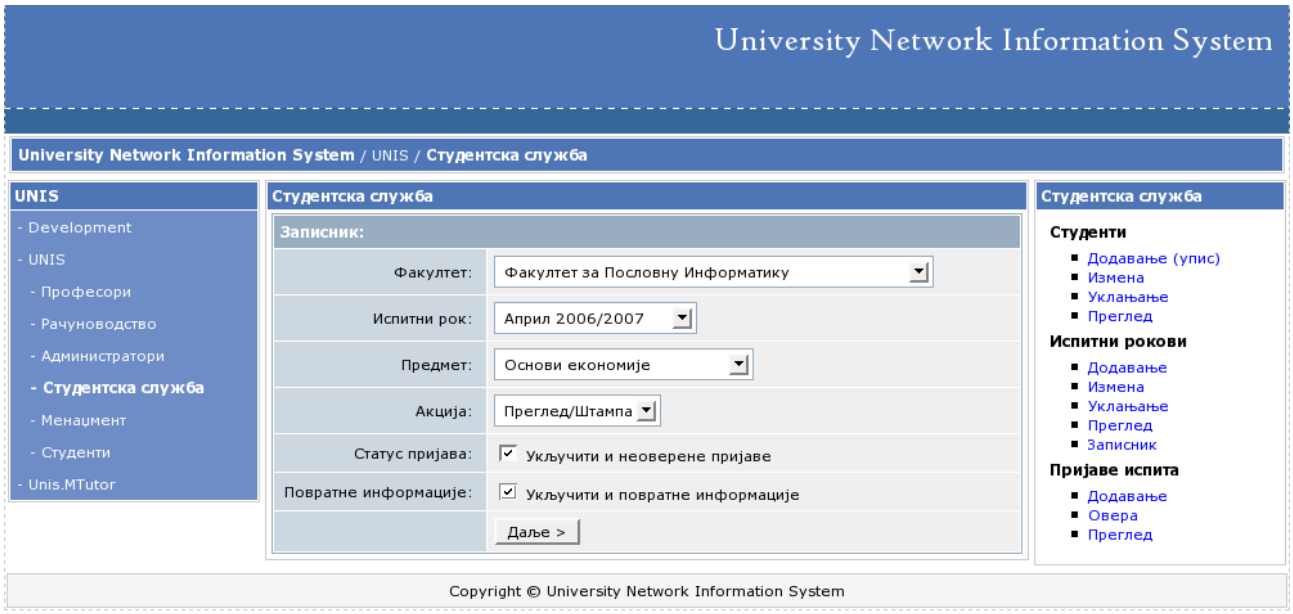
Pregled ispitnih rokova se vrši odabirom opcije „Ispitni rokovi/Pregled“ u okviru modula studentskih službi. Ova opcija je delimično završena i trenutno prikazuje samo listu ispitnih rokova po fakultetima.



Slika 46 - Lista svih ispitnih rokova po fakultetima

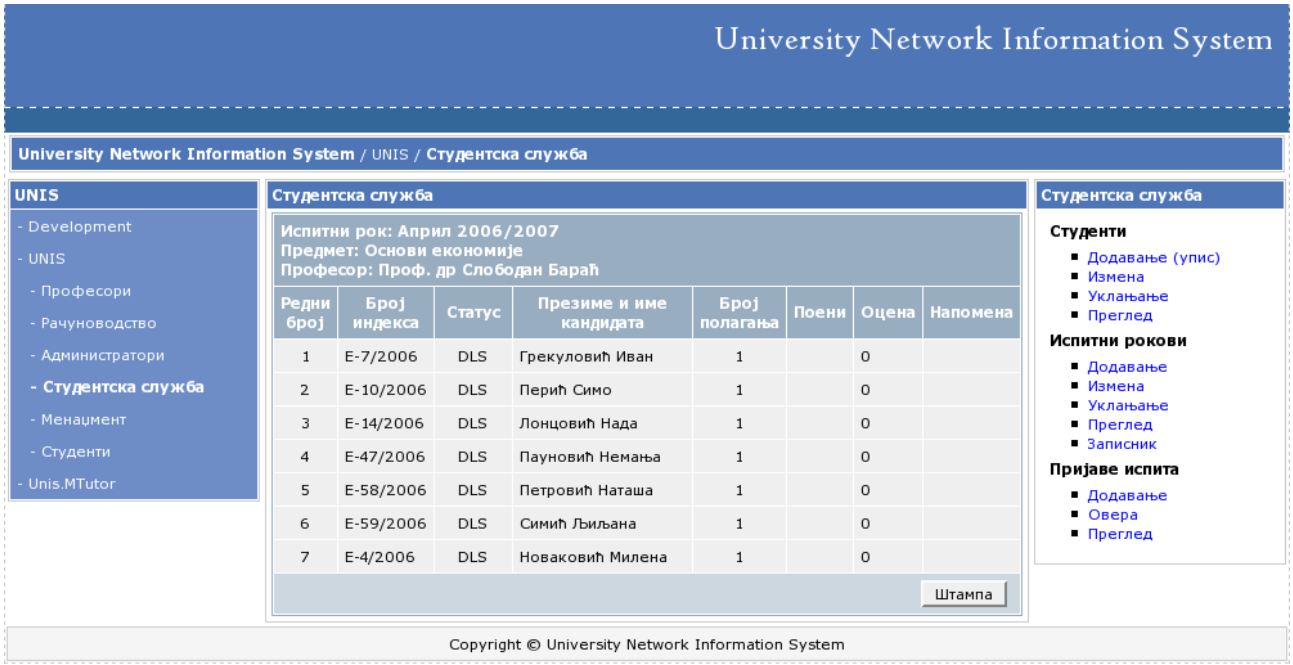
5.2.2.5. Zapisnik

Opcija „Zapisnik“ unutar akcione grupe „Ispitni rokovi“ u okviru modula studentskih službi nudi mogućnost pregleda, štampe i popunjavanja zapisnika o ispitima ispitnih rokova. Odabirom ove opcije potrebno je odabrati ispitni rok, predmet, tip akcije kao i to da li u zapisnik treba uključiti i neoverene prijave i povratne informacije (unešene rezultate, ukoliko ih ima).



Slika 47 – Formular za odabir roka, predmeta, akcije i dodatnih parametara

Odabir opcije „Pregled/Štampa“ prikazuje listu studenata koji su prijavili polaganje ispita iz odabranog predmeta.



Slika 48 – Lista studenata koji su prijavili predmet

Na dnu liste se nalazi dugme „Štampa“ čijom se aktivacijom u novom prozoru dobija varijanta zapisnika pogodna za štampu:

Факултет за пословну информатику

ЗАПИСНИК О ПОЛАГАЊУ ИСПИТА

Факултет за Пословну Информатику, Београд							
Предмет: Основи економије				**Шифра предмета:**			
Шифра наставника:			**Име и презиме наставника:** Проф. др Слободан Бараћ	**Група:** 1		**Испитни рок:** Април 2006/2007	
Датум: 13. 04. 2007.			**Време:** 11:30	**Сала:** Амфитеатар А-11			
Редни број	**Број индекса**	**Статус**	**Презиме и име кандидата**	**Број полагања**	**Поени**	**Оцена**	**Напомена**
1	Е-7/2006	DLS	Грекуловић Иван	1		0	
2	Е-10/2006	DLS	Перић Симо	1		0	
3	Е-14/2006	DLS	Лонцовић Нада	1		0	
4	Е-47/2006	DLS	Пауновић Немања	1		0	
5	Е-58/2006	DLS	Петровић Наташа	1		0	
6	Е-59/2006	DLS	Симић Љиљана	1		0	
7	Е-4/2006	DLS	Новаковић Милена	1		0	
Датум:				**Потпис испитивача:**			

Slika 49 – Varijanta za štampu zapisnika

Odabir opcije „Popunjavanje“ na osnovnom formularu za zapisnike nudi formular sa studentima koji su prijavili ispit uz mogućnost unosa ostvarenih rezultata (broja osvojenih poena, ocene i eventualne napomene). Podrazumevana ocena je 0 što označava da student nije polagao ispit.

University Network Information System

University Network Information System / UNIS / Студентска служба

UNIS
- Development
- UNIS
  - Професори
  - Рачуноводство
  - Администратори
  - Студентска служба
  - Менаџмент
  - Студенти
- Unis.MTutor

Студентска служба

Испитни рок: Април 2006/2007
Предмет: Основи економије
Професор: Слободан Бараћ

Редни број	Број индекса	Статус	Презиме и име кандидата	Број полагања	Поени	Оцена	Напомена
1	E-47/2006	DLS	Пауновић Немања	1		0	
2	E-4/2006	DLS	Новаковић Милена	1		0	

Унос

Студентска служба

Студенти
- Додавање (упис)
- Измена
- Уклањање
- Преглед

Испитни рокови
- Додавање
- Измена
- Уклањање
- Преглед
- Записник

Пријаве испита
- Додавање
- Овера
- Преглед

Copyright © University Network Information System

Slika 50 – Formular za popunjavanje zapisnika

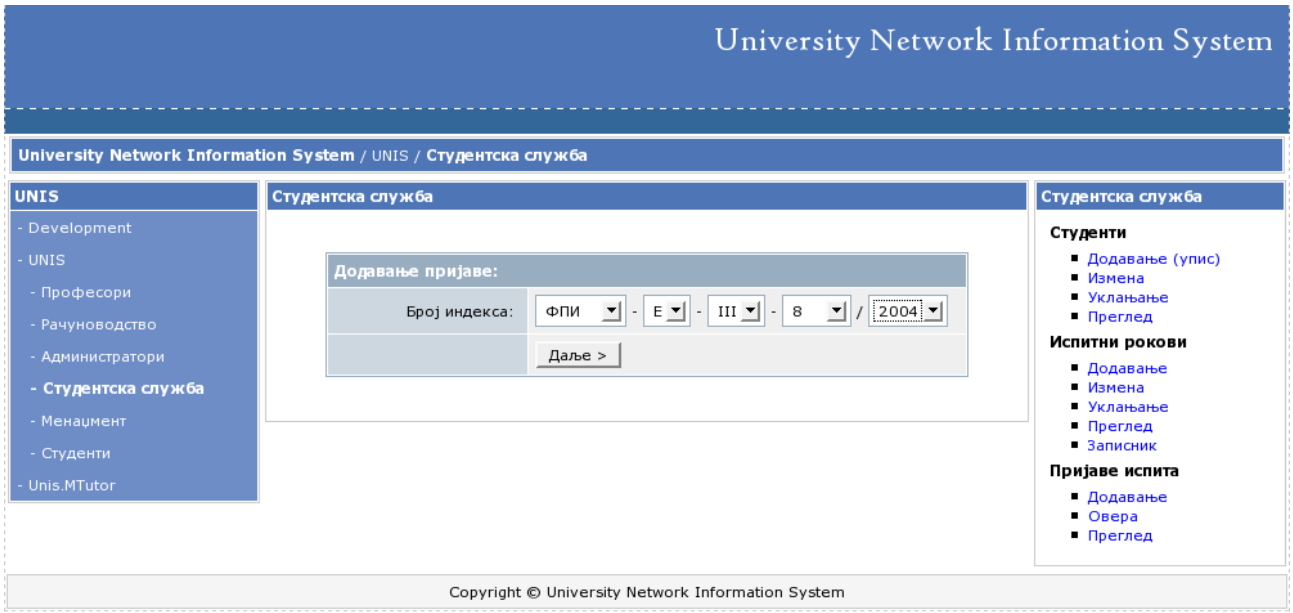
Nakon unosa poena, ocena i napomena aktivacijom dugmeta „Unos“ sistem unosi parametre u bazu podataka.

5.2.3. Prijave ispita

Sekcija „Prijave ispita“ u okviru modula informacionog sistema za studentske službe sadrži opcije putem kojih se manipuliše prijavama ispita studenata.

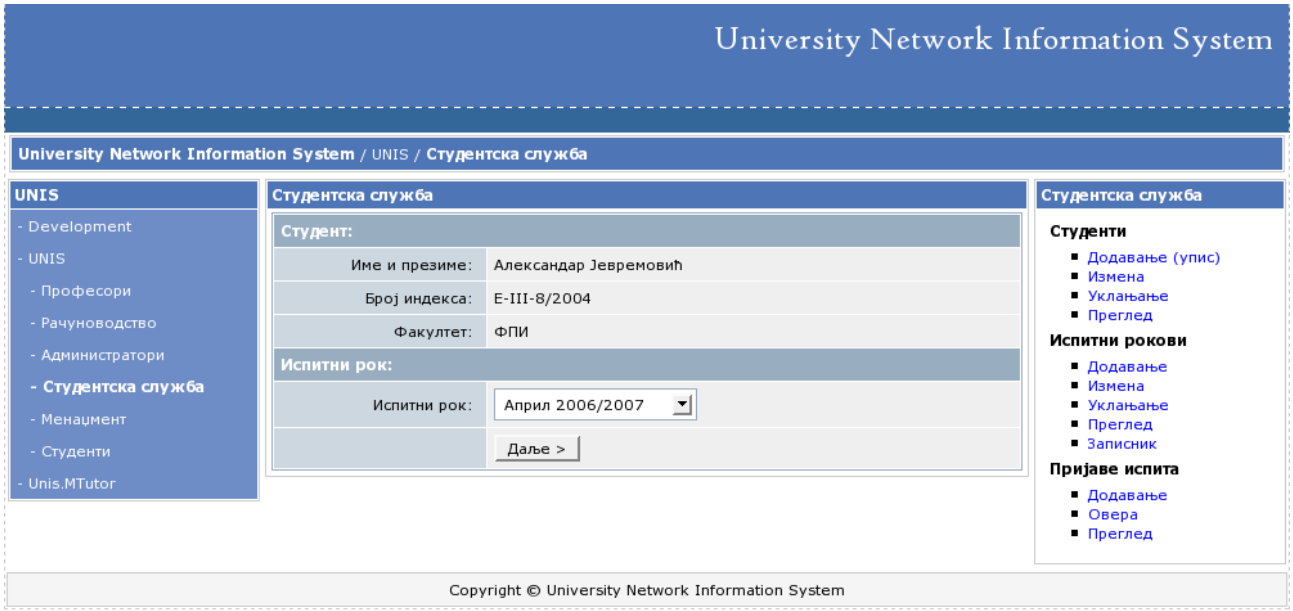
5.2.3.1. Dodavanje

Iako studenti imaju mogućnost da u okviru svog modula prijave ispit, mogućnost dodavanja prijave ispita je uključena i u modul studentske službe. Odabirom opcije „Prijave ispita/Dodavajne“ sistem prikazuje formular u okviru koga je potrebno uneti broj indeksa studenta za koga se vrši prijava ispita.



Slika 51 – Odabir studenta za koga se prijavljuju ispiti

Nakon unosa broja indeksa studenta, sistem pronalazi studenta u bazi i u narednom formularu prikazuje njegovo ime i prezime pored polja u kome je potrebno odabrati ispitni rok za koji se vrši prijava ispita. U slučaju da u bazi podataka ne postoji student sa odabranim brojem indeksa, sistem prikazuje adekvatnu poruku.



Slika 52 – Odabir ispitnog roka za koji se prijavljuju ispiti

Odabirom ispitnog roka sistem prikazuje novi formular koji uključuje parametre studenta, parametre ispitnog roka kao i listu predmeta (ispita) sa mogućnošću odabira.

University Network Information System

University Network Information System / UNIS / Студентска служба

UNIS
- Development
- UNIS
  - Професори
  - Рачуноводство
  - Администратори
  - Студентска служба
  - Менаџмент
  - Студенти
- Unis.MTutor

Студентска служба

Студент:

Име и презиме:	Александар Јевремовић
Број индекса:	E-III-8/2004
Факултет:	ФПИ

Испитни рок:

Испитни рок:	Април 2006/2007
Период пријаве:	29. 03. 2007. - 12. 04. 2007.
Период испита:	11. 04. 2007. - 26. 04. 2007.

Предмети:

- [ ] **Основи економије**, Проф. др Слободан Бараћ
  13.04.2007. 11:30 , Амфитеатар А-11
- [ ] **Социологија**, Проф. др Михаило Пешић
  18.04.2007. 11:00 , С-17
- [x] **Основи рачунарске технике**, Проф. др Бранко Ковачевић
  19.04.2007. 19:00 , Амфитеатар А-23
- [ ] **Основи права**, Проф. др Драган Митровић
  16.04.2007. 12:00 , Амфитеатар А-12
- [x] **Базе података**, Проф. др Младен Веиновић
  13.04.2007. 17:30 , С-11
- [ ] **Мултимедија**, Проф. др Милан Милосављевић
  20.04.2007. 19:00 , С-11

Пријава испита

Студентска служба

Студенти
- Додавање (упис)
- Измена
- Уклањање
- Преглед

Испитни рокови
- Додавање
- Измена
- Уклањање
- Преглед
- Записник

Пријаве испита
- Додавање
- Овера
- Преглед

Copyright © University Network Information System

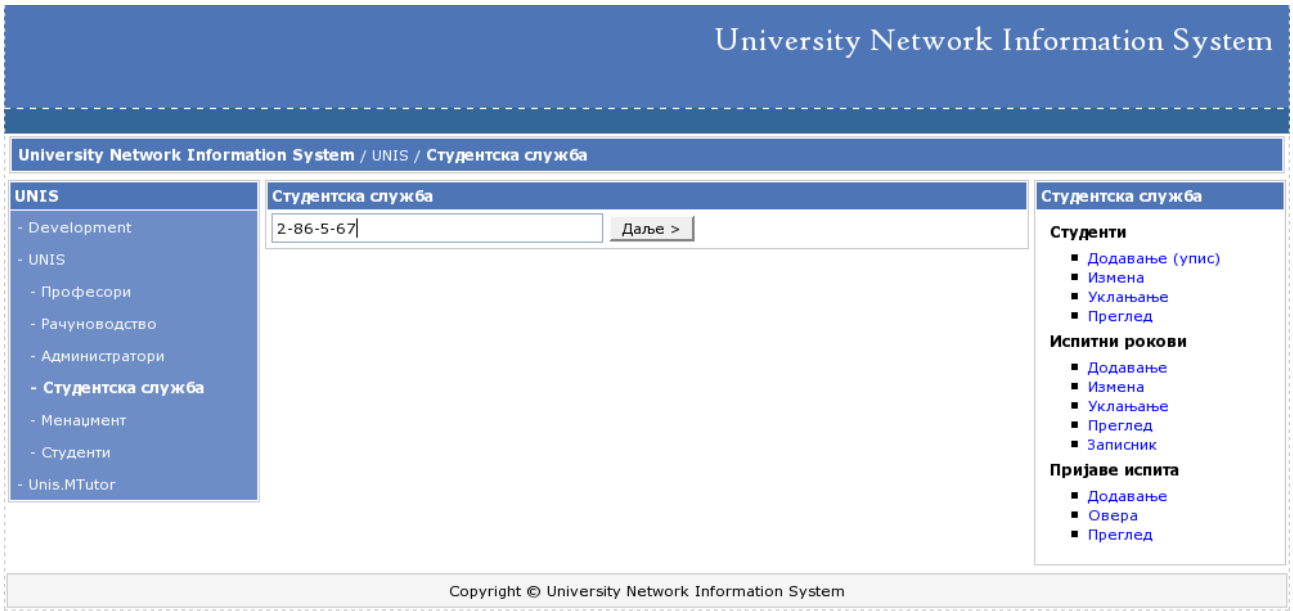
Slika 53 – Odabir predmeta (ispita) koji se prijavljuju

Odabirom željenih ispita i aktiviranjem dugmeta „Prijava ispita“ sistem upisuje parametre prijave u bazu podataka i izbacuje odgovarajuću poruku.

Slika 54 - Poruka o uspešnoj prijavi ispita za studenta.

5.2.3.2. Overa

Overa prijava ispita je operacija kojom se potvrđuje da je student ispunio finansijske obaveze nakon postavljanja prijave. S obzirom na to da se svakoj prijavi ispita pridružuje jedinstveni identifikacioni kod (koji se kod naloga za uplatu koristi u polju „poziv na broj“) overa prijava se vrši na osnovu tog parametra. (Inicijalna ideja je bila da se overa ispita vrši automatski nakon prijema izvoda sa žiro računa u modul računovodstva ali se u praksi pokazalo da postoji preveliki broj grešaka za automatizaciju). Odabirom ove opcije sistem nudi formular u koji je potrebno uneti jedinstveni identifikacioni kod prijave.



Slika 55 – Formular za unos jedinstvenog identifikatora prijave koja se overava

Nakon unosa koda prijave i potvrđivanjem unosa aktiviranjem dugmeta „Dalje“, sistem u bazi pronalazi prijavu kojoj odgovara uneti kod. (U slučaju da u bazi podataka ne postoji prijava sa tim identifikacionim kodom, sistem izbacuje odgovarajuću poruku). Na sledećem formularu sistem prikazuje sve parametre o studenti, ispitnom roku i prijavljenim ispitima na koje se odnosi prijava kao i iznos koji je bilo potrebno uplatiti. Sistem traži potvrdu overavanja prijave.

University Network Information System

University Network Information System / UNIS / Студентска служба

UNIS
- Development
- UNIS
  - Професори
  - Рачуноводство
  - Администратори
  - Студентска служба
  - Менаџмент
  - Студенти
- Unis.MTutor

Студентска служба

Испитни рок:	
Назив:	Мај 2006/2007
Период пријаве:	12. 04. 2006. - 13. 04. 2007.
Период испита:	15. 04. 2006. - 16. 04. 2006.
Цена испита:	5.00€
Пријава:	
Позив на број:	2-86-5-67:E-III-8/2004
Датум:	13. 04. 2007. 14:33
Затворена:	Да
Оверена:	Не
Накнадна:	Не
Укупна цена:	5€
Студент:	
Бр. индекса:	E-III-8/2004
Име и презиме:	Александар (Данило) Јевремовић
Адреса:	Безимена ББ 11000 Београд Србија
Е-маил:	ajevremovic@singidunum.ac.yu
Телефон:	381 64 132 7119
Предмети:	
Базе података	0
Акција:	

Назад Овера

Студентска служба

Студенти
- Додавање (упис)
- Измена
- Уклањање
- Преглед

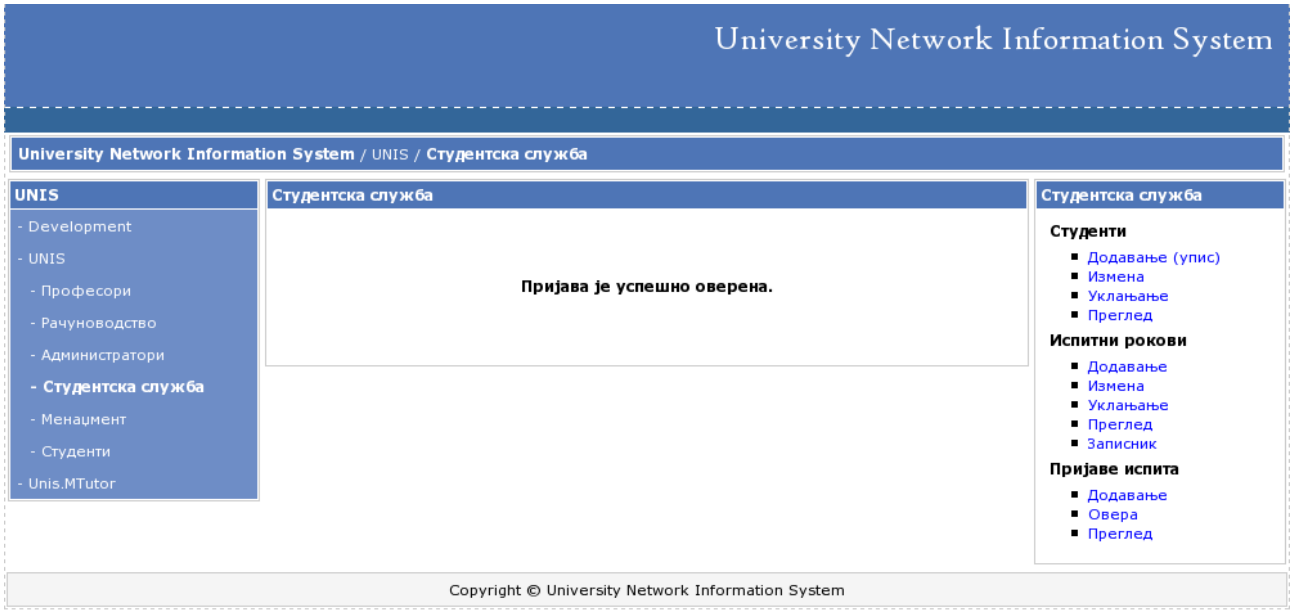
Испитни рокови
- Додавање
- Измена
- Уклањање
- Преглед
- Записник

Пријаве испита
- Додавање
- Овера
- Преглед

Copyright © University Network Information System

Slika 56 - Prikaz parametara prijave i zahtev za potvrdom overe

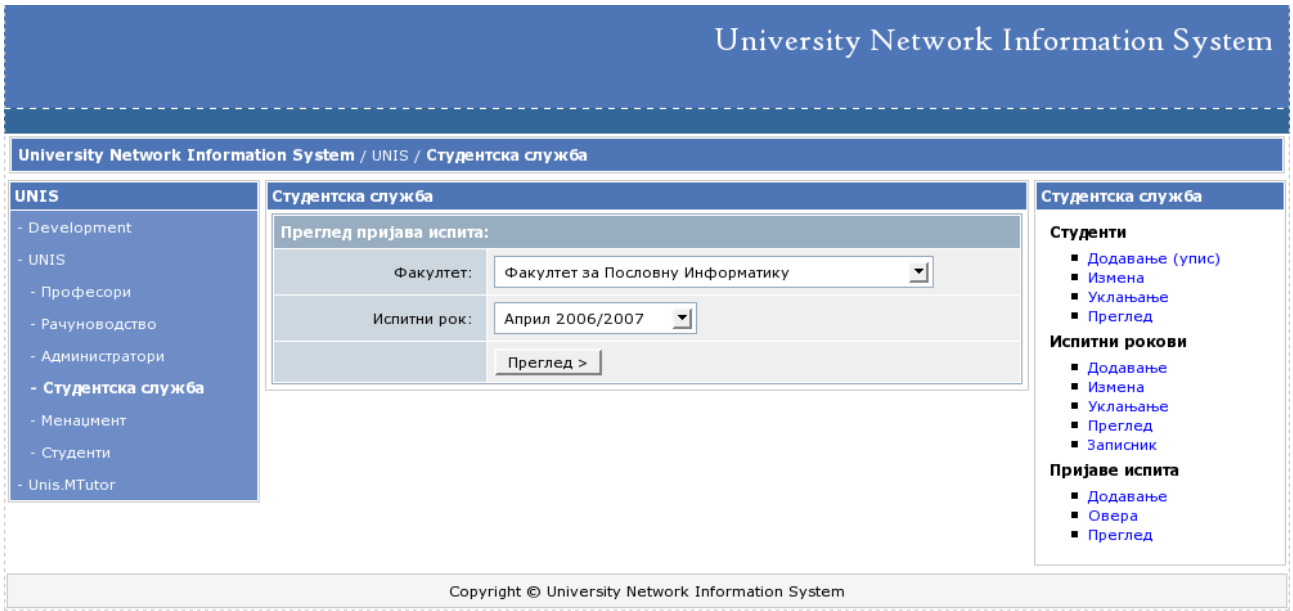
Potvrđivanjem overe prijave ispita aktivacijom dugmeta „Overa“ sistem u bazi podataka označava prijavu kao overenu i ispisuje adekvatnu poruku.



Slika 57 – Poruka o uspešnoj overi prijave ispita

5.2.3.3. Pregled

Opcija „Pregled“ u sekciji „Prijave ispita“ omogućava pregled svih prijava koje su postavljene na odabranom fakultetu u odabranom ispitnom roku:

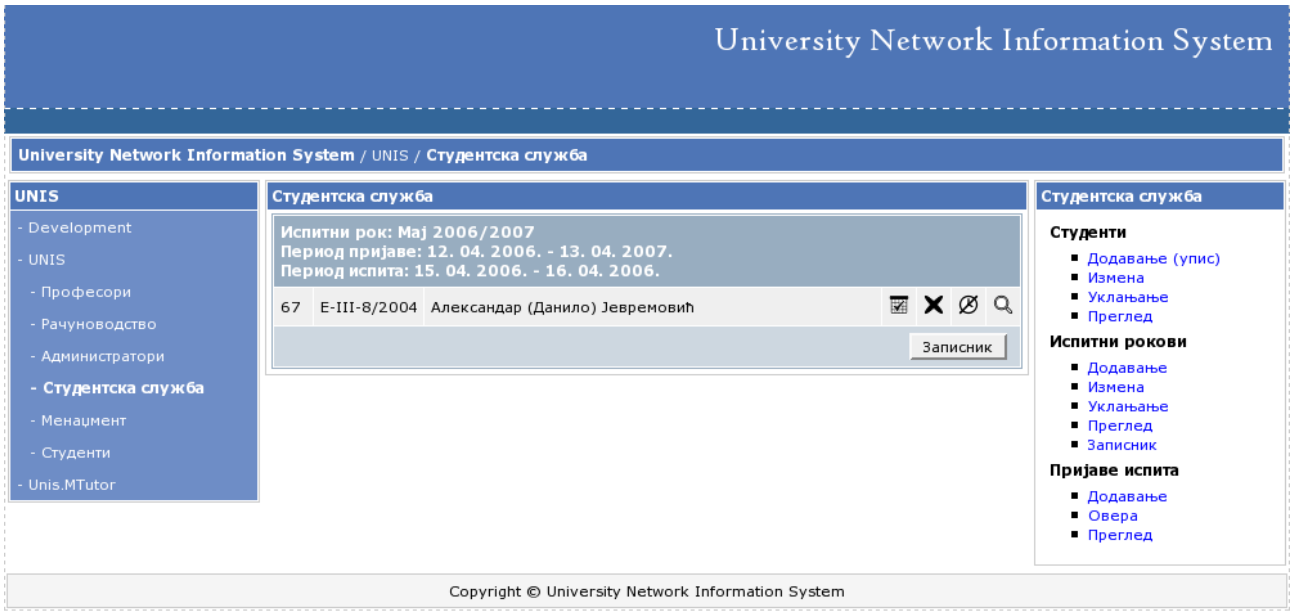


Slika 58 – Formular za odabir fakulteta i ispitnog roka za koji se pregledaju prijave ispita

Nakon odabira ispitnog roka, sistem prikazuje listu svih postavljenih prijava u formatu:

- broj indeksa studenta
- ime i prezime
- da li je prijava zatvorena (rezervisano za buduća proširenja)
- da li je prijava overena

- da li je prijava naknadna
- link ka detaljnim prikazom prijave



Slika 59 - Lista studenata koji su prijavili ispite za odabrani ispitni rok

5.3. Studenti

Unutar softverskog interfejsa (modula) namenjenog studentima realizovane su sledeće funkcionalnosti:

- Sesija
  - Prijava / Odjava
- Prijava ispita
  - Nova prijava
  - Pregled prijava

Modul je projektovan tako da podrži i rad sa finansijskim parametrima studenta kao i uputstvo o načinu korišćenja modula ali te funkcionalnosti trenutno nisu realizovane.

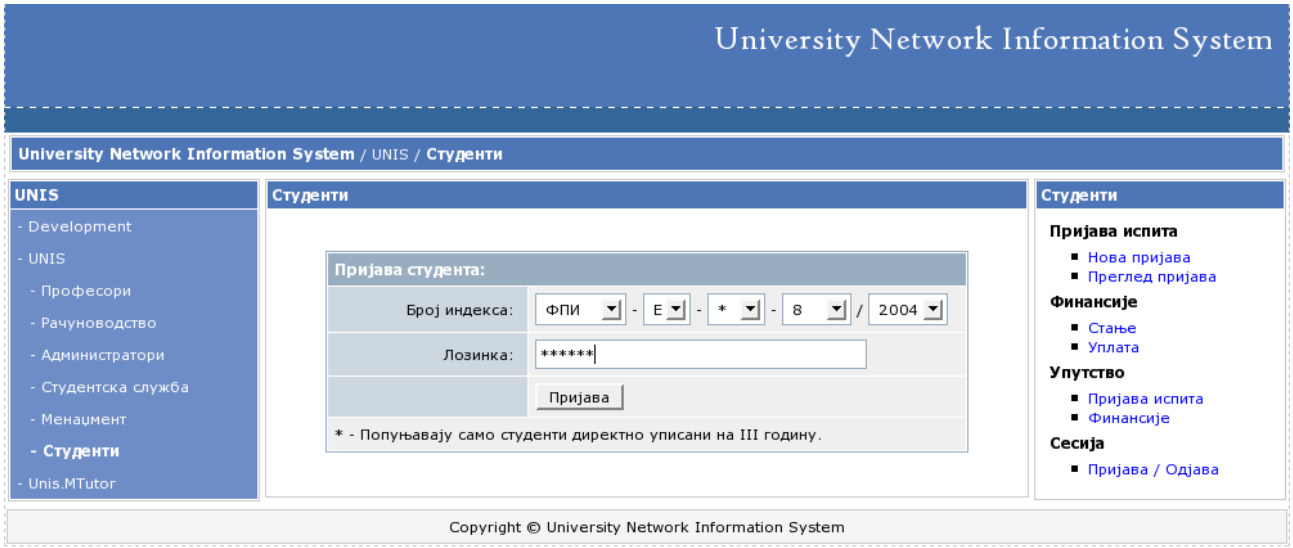
5.3.1. Sesija

Sekcija „Sesija“ u okviru studentskog modula je namenjena da podrži prijavljivanje na sistem i odjavljivanje sa sistema. Za pristup bilo kojoj funkcionalnosti sistema neophodno je da se student prvo prijavi na sistem dostavljanjem broja indeksa i adekvatne lozinke. Osim provere da li je dostavljena odgovarajuća lozinka vrši se i provera da li je studentski nalog aktivan (tj. da li nije isključen u bazi podataka) kao i da li je student upisan na neku godinu studija u trenutku u kome se vrši prijavljivanje. Pristupne lozinke se inicijalno automatski generišu tajnim algoritmom i moguće ih je promeniti u modulu za studentske službe. Pregled lozinki studenata nije moguć s obzirom na to da se one u bazi čuvaju kao rezultat hash funkcija. U planiranom proširenju studentskog modula predviđeno je dodavanje mogućnosti da studenti sami menjaju lozinke kao i da se lozinke za pristup informacionom sistemu koriste i za pristup računarima na Univerzitetu.

Odjavljivanje sa sistema je korak koji studenti obavljaju nakon završetka rada. Prijavljivanje na sistem čini pristup ostalim funkcionalnostima moguć sve do odjavljivanja sa sistema. Odjavljivanje sa sistema se vrši automatski pri zatvaranju svih prozora Web klijenta studenta ili nakon isteka vremena definisanog u sistemu (60 minuta). Nakon odjavljivanja studenti dobijaju ponovni pristup formularu za prijavljivanje.

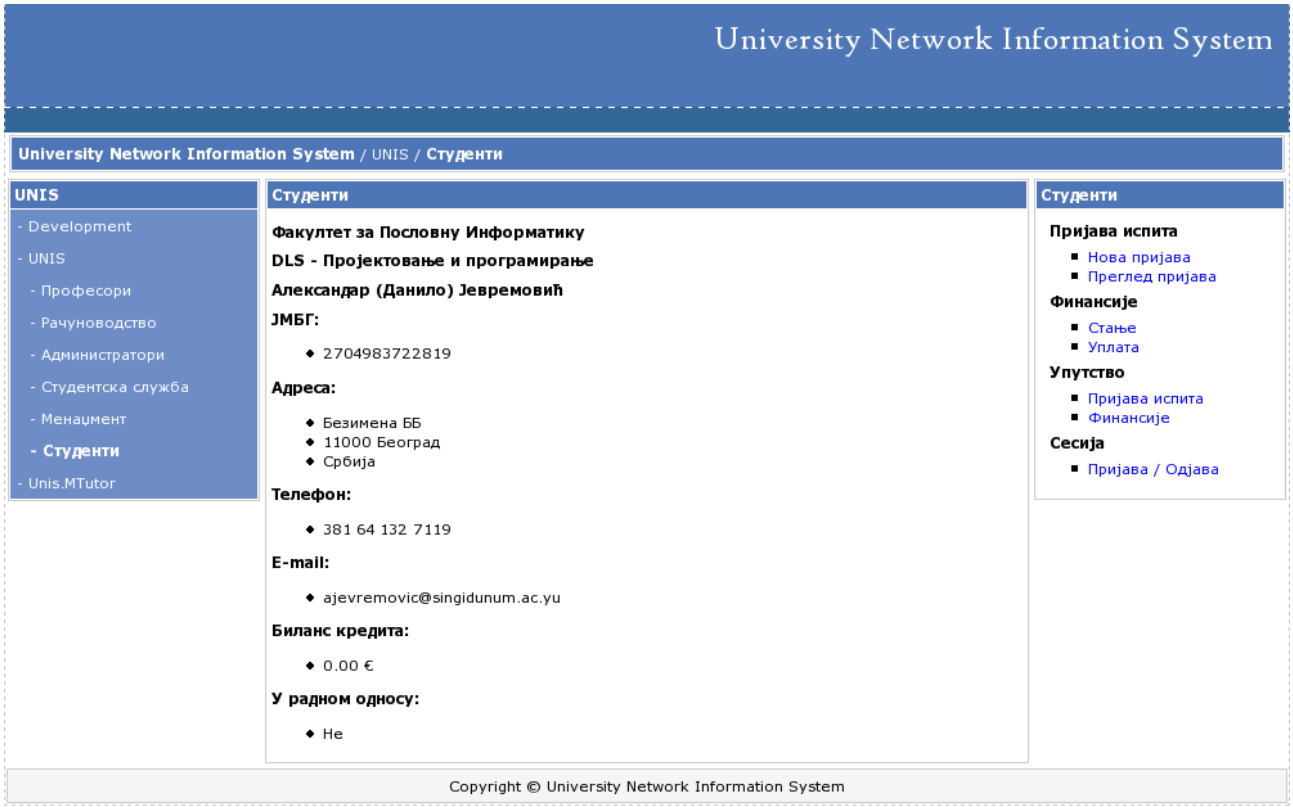
5.3.1.1. Prijava / Odjava

Prijavljivanje na sistem se vrši putem formulara u koji studenti unose broj indeksa i odgovarajuću lozinku. Lozinke inicijalno određuje studentska služba a planirana je mogućnost izmene od strane studenata.



Slika 60 – Prijava studenata na sistem

Nakon uspešne prijave na sistem student pristupa osnovnoj stranici modula na kojoj su ispisani parametri studenta pridruženi nalogu.



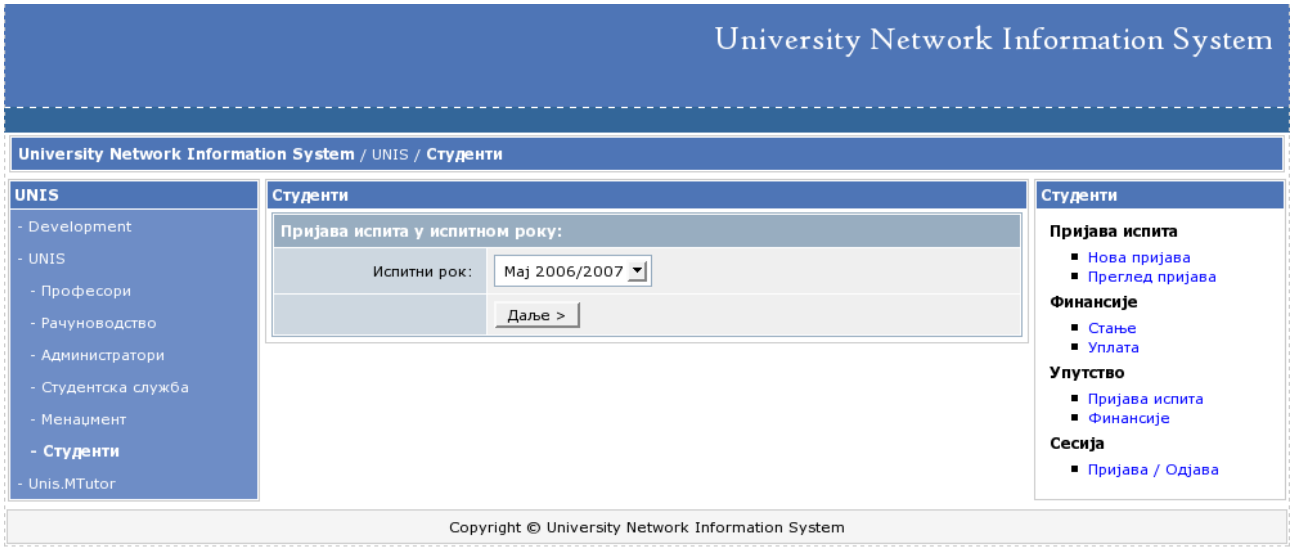
Slika 61 – Osnovni ekran nakon uspešne prijave studenta

5.3.2. Prijava ispita

U sekciji studentskog modula vezanoj za prijavu ispita studenti mogu obaviti on-line prijavu ispita u određenom ispitnom roku (koji je prethodno kreiran i objavljen u modulu studentske službe). Ova sekcija takođe omogućava studentima da vide parametre popunjenog naloga za uplatu kao i da prate stanje i parametre prethodnih prijava. Stanje prijave može biti „neoverena“ i „overena“. Neoverenom se smatraju prijave koje su postavljene ali za koje u računovodstvu nije evidentirana uplata troškova. Nakon evidentiranja uplate troškova u modulu studentske službe se vrši overa prijave.

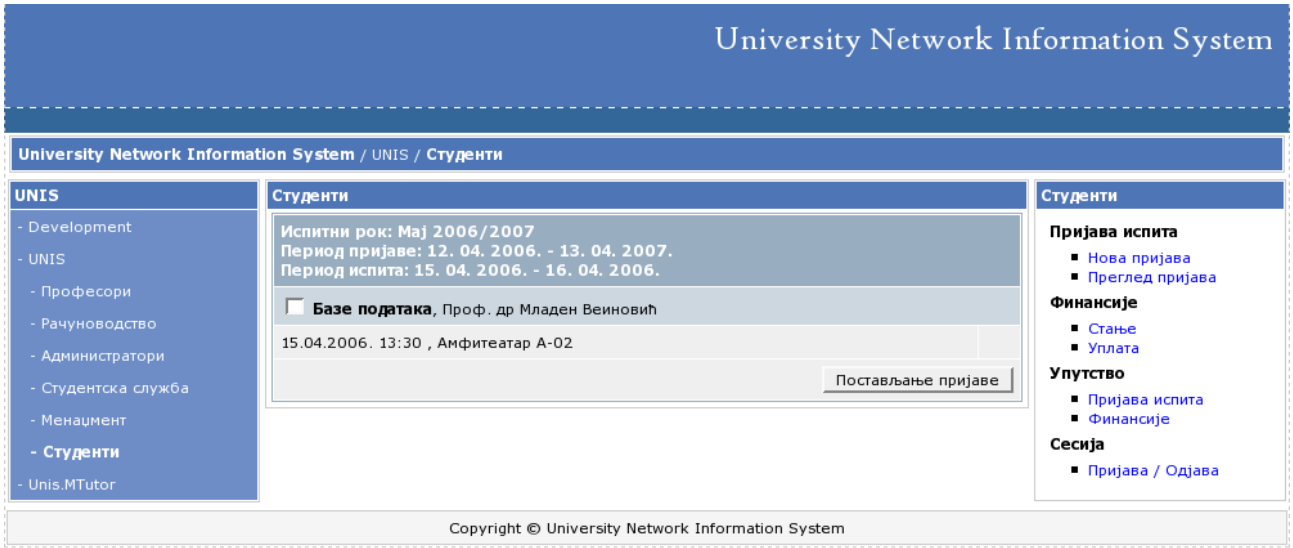
5.3.2.1. Nova prijava

Odabirom opcije „Nova prijava“ u studetskom modulu na ekranu se pojavljuje formular za odabir ispitnog roka za koji se prijavljuju ispiti:



Slika 62 – Odabir ispitnog roka za koji se postavlja prijava

U slučaju da trenutno nema ispitnih rokova za koje je moguća prijava ispita umesto formulara će se prikazati odgovarajuća poruka. Sledeći korak je odabir ispita koji se prijavljuju:

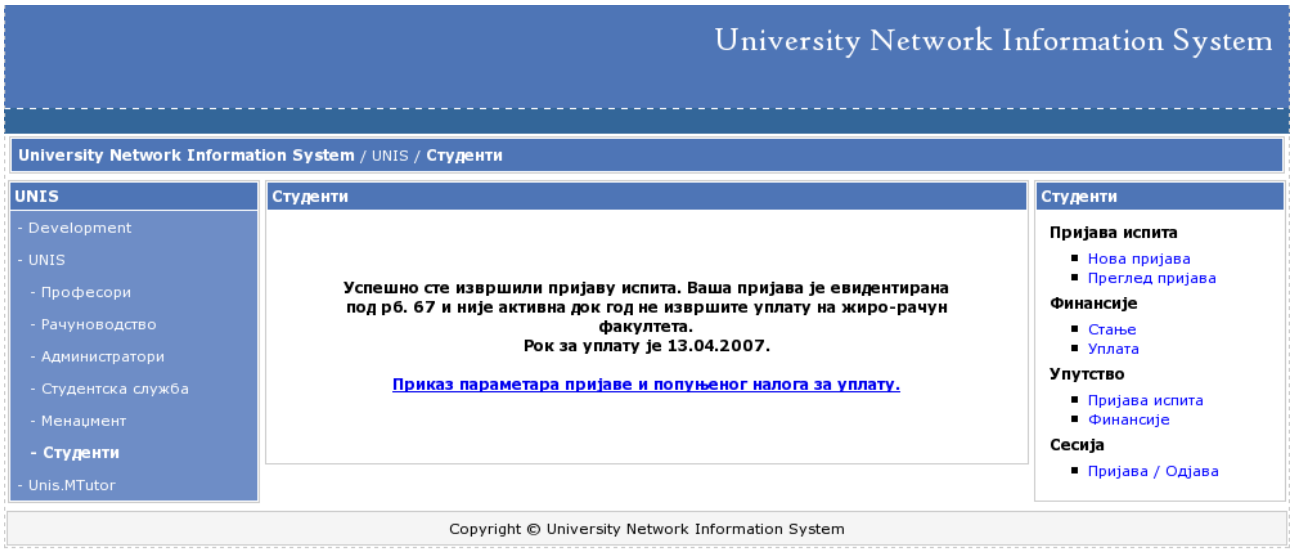


Slika 63 – Odabir predmeta koji se prijavljuju

Markiranjem predmeta (ispita) koji se prijavljuju i prelaskom na sledeći korak dobija se pregled odabranih ispita, sumarna iznos troškova polaganja (cena ispita x broj ispita) i od studenta se zahteva potvrda prijave.

Slika 64 – Zahtev za potvrdom prijave

U slučaju da je student napravio grešku pri odabiru predmeta koje želi da prijavi odbijanjem potvrde (opcija „Ne“) se vraća na prvi korak. U protivnom prelazi na sledeći korak koji se sastoji od informacije o uspešnoj prijavi, rednom broju prijave i uputstvima i rokovima za dalje aktivnosti.



Slika 65 – Informacija u uspešnoj prijavi i roku za uplatu.

Završni korak se sastoji i od linka ka prikazu popunjenog naloga za uplatu o čemu će biti više reči u delu „Pregled prijava“ s obzirom na to da se prikaz naloga za uplatu generiše u toj sekciji programa.

5.3.2.2. Pregled prijava

Akcija „Pregled prijava“ u studentskom modulu sastoji se od dva koraka:

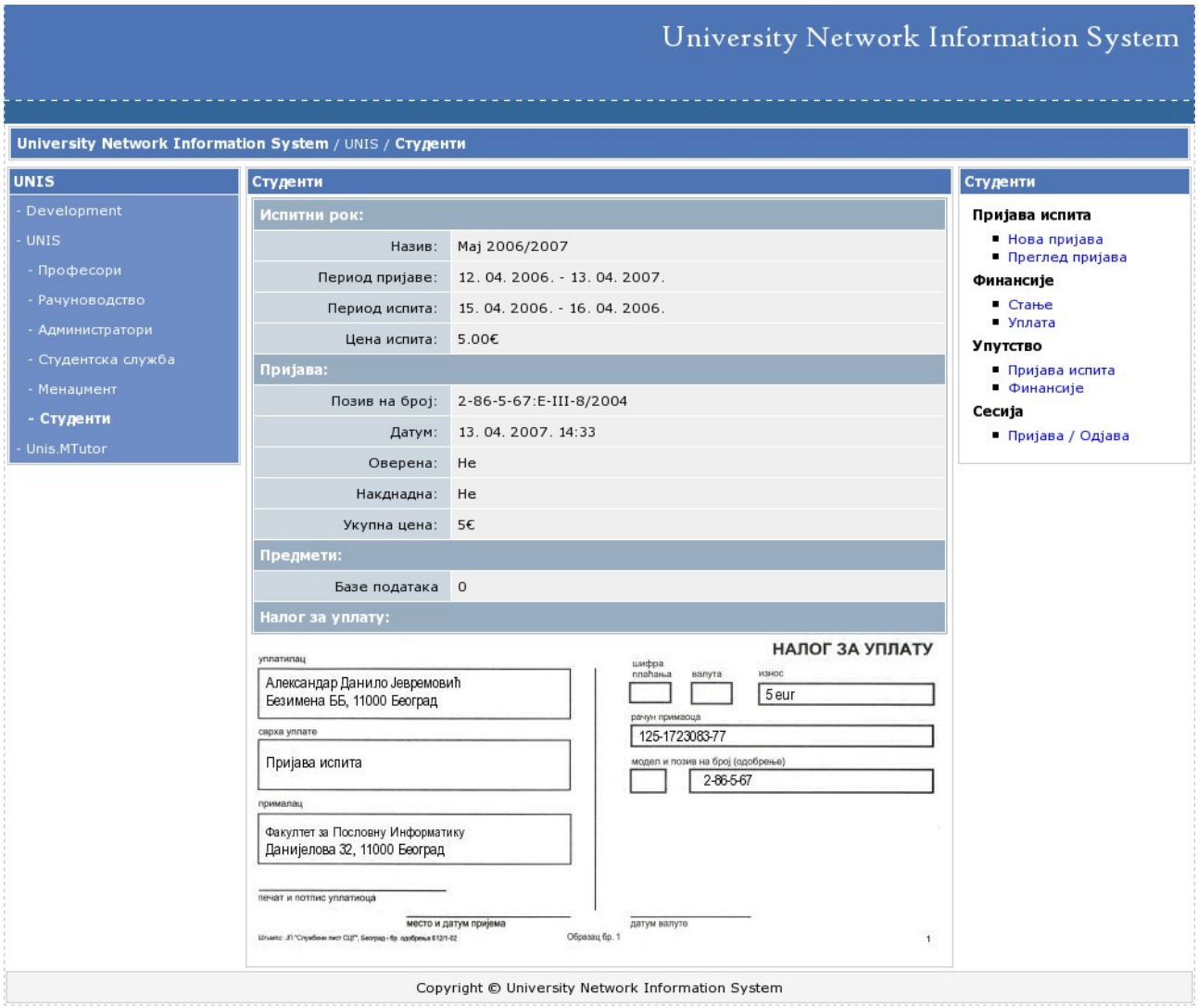
1. lista svih prijava ispita
2. detaljni prikaz parametara jedne prijave

Inicijalni korak jeste prikaz liste svih dosadašnjih prijava studenata sa pridruženom informacijom o statusu (overena/neoverena) u vidu boje prikaza:

Slika 66 – Lista postavljenih prijava ispita

Neoverene prijave su označene sivom a overene prijave crnom bojom. Informacija o statusu prijave je značajna za studente zbog toga što overavanje prijave znači da je evidentirana uplata a samo overene prijave ulaze u konačan zapisnik za polaganje ispita.

Odabirom neke od prijava sa liste (bez obzira da li je u pitanju overena ili neoverena prijava) dobija se prikaz detaljnih parametara prijave i izgled popunjenog naloga za uplatu:



Slika 67 – Detaljan prikaz odabrane prijave sa popunjenim nalogom za uplatu

Detaljni parametri prijave sadrže informacije o ispitnom roku na koji se odnosi prijava (naziv, period prijave, period ispita, cena ispita), informacije o samoj prijavi (poziv na broj, datum postavljanja, status, ukupni troškovi...), informacije o prijavljenim predmetima (naziv i ocena) i prikaz popunjenog naloga za uplatu.

6. Dalji pravci razvoja

Realizacijom osnovnih opisanih funkcionalnosti informacionog sistema univerziteta razvoj se ne može smatrati završišenim, ne samo iz razloga što određene funkcionalnosti i korisnički moduli nisu realizovani, već i zbog toga što je sistem potrebno povezati sa ostalim softverskim sistemima koji se koriste na univerzitetu, stalno prilagođavati novim potrebama poslovnog sistema, održavati korak i koristiti prednosti novih tehnologija i rešenja, itd. U skladu sa tim, definisano je 5 osnovnih pravaca u kojima se predviđa dalji razvoj sistema:

1. Podizanje bezbednosti/pouzdanosti sistema na svim nivoima
2. Realizacija preostalih korisničkih modula
3. Iterativno uvođenje novih funkcionalnosti
4. Prevođenje na potpunu SOA arhitekturu
5. Povezivanje sa drugim sistemima

Prioritetni koraci daljeg razvoja jesu „Podizanje bezbednosti/pouzdanosti sistema na svim nivoima“ i „Realizacija preostalih korisničkih modula“ što ukazuje na potrebu da se u sistem ugrade dodatni kontrolni i zaštitni mehanizmi pre masovnije upotrebe. Dodatne funkcionalnosti bi u sistem bile uvođene u iteracijama tj. sistem bi se proširivao i na područja poslovnog sistema koja nisu obuhvaćena inicijalnim modelom a praćene bi bile i izmene u poslovnom sistemu univerziteta.

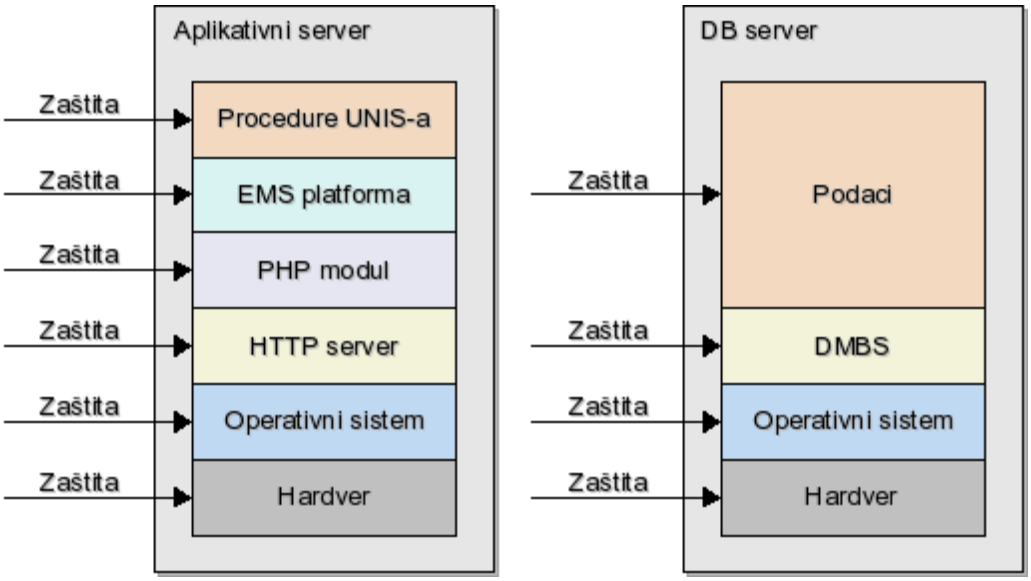
S obzirom na činjenicu da životni vek savremenih informacionih tehnologija postaje sve kraći, jedan od zadataka u daljem razvoju bi bio i obezbeđivanje kompatibilnosti sa savremenim tehnologijama u cilju blagovremene pripreme na eventualne tehnološke migracije. Kao najperspektivnije rešenje se ističe SOA arhitektura koja ima velike izglede da konceptualno preživi duži vremenski period i omoući bezbolnu eventualnu tehnološku migraciju sistema. SOA arhitektura dodatno nudi značajne pogodnosti u pravljenju interfejsa ka ostalim softverskim sistemima sa kojima bi centralni informacioni sistem bio povezan.

6.1. Podizanje bezbednosti/pouzdanosti sistema na svim nivoima

Ulaskom informacionog sistema u upotrebu pred njega se postavlja jedan novi i posebno važan zahtev - da omogući nesmetan rad korisnika kroz konstantnu dostupnost i pružanje tačnih i ažurnih informacija. Iako jednostavno formulisan ovaj zahtev otvara nova pitanja čiji se odgovori mogu postaviti u okviru kategorija:

1. bezbednost (zaštita)
2. pouzdanost (dostupnost)

Kada je u pitanju bezbednost (zaštita) sistema ona se pre svega odnosi na zaštitu baze podataka s obzirom na to da se aplikativni podaci mogu jednostavno povratiti u slučaju oštećenja. Sa druge strane, uspešan napad na aplikativni deo sistema koji bi napadaču omogućio izmenu procedura za rad sa bazom podataka, imao bi isti efekat, što ukazuje i na potrebu za zaštitom svih komponenti sistema koje mogu dovesti do oštećenja i neovlašćenog pristupa podacima.



Slika 68 - Slojevi na kojima je potrebno vršiti zaštitu sistema

Za ostvarivanje adekvatnog nivoa bezbednosti (zaštite) sistema potrebno je ograničiti kontrolu pristupa svim slojevima sistema s obzirom na to da se pomoću neovlašćenog pristupa nižim slojevima mogu vršiti izmene u višim slojevima. To znači da je zaštitu (kontrolu pristupa) potrebno vršiti već na hardverskom nivou (fizička oštećenja, krađa...) a zatim i na svim višim nivoima. Na najvišem nivou (procedure UNIS-a) za kontrolu pristupa su identifikovana dva osnovna načina:

1. razvoj internog podsistema privilegija korisnika unutar kategorija i
2. kontrola pristupa korišćenjem SmartCard tehnologije

Pitanje pouzdanosti (dostupnosti) sistema je takođe veoma značajno jer problemi koji se na tom nivou mogu javiti imaju mogućnosti, od toga da uskrate korisnicima normalan rad do toga da delimično ili u potpunosti unište centralnu bazu podataka. U tom cilju je za podizanje pouzdanosti sistema odabran princip:

3. kreiranje redundantnih vitalnih komponenti sistema

6.1.1. Razvoj internog podsistema privilegija korisnika unutar kategorija

Inicijalni zahtev koji je postavljen pred informacioni sistem je bio omogućavanje studentima koji studiraju putem DLS-a da putem Interneta prijave ispite. U takvoj situaciji je za kontrolu prijava i štampanje zapisnika zaduženo DLS služba univerziteta koja bi se mogla porediti sa studentskom službom studenata svih fakulteta univerziteta koji studiraju putem DLS sistema. Takva postavka je eliminisala potrebu za internim ograničavanjem pristupa operatera studentskim prijavama ostalih fakulteta unutar modula za korisnike kategorije „Studentske službe“. Slična situacija se može primeniti i na module korisničkih kategorija „Menadžment“, „Profesori“, „Računovodstvo“ i (delimično) „Administratori“.

Intenzivnijom upotrebom sistema javlja se potreba da u njoj učestvuje veći broj ljudi što je signal da je potrebno preciznije određivanje njihovih privilegija i da je dodeljivanje privilegija na osnovu korisničkog modula informacionog sistema neadekvatno rešenje. S obzirom na to da su operacije unutar modula organizovane u akcione grupe i njima pripadajuće akcije, interni podsistem privilegija se može realizovati njihovim jednostavnim pridruživanjem korisničkim nalozima. Dodatno, elemente unutar baze podataka (fakultet, programe, predmete...) je takođe moguće pridružiti nalozima i na taj način ograničiti korisnike na rad sa podacima za koje su nadležni.

6.1.2. Kontrola pristupa korišćenjem SmartCard tehnologije

Osim kreiranja opisanih korisničkih naloga (i njima pridruženih aktivnosti) potrebno je uvesti dodatnu kontrolu njihovog korišćenja nekim od sistema koji omogućavaju precizniju proveru identiteta korisnika. Jedan od osnovnih sistema za ovakvu proveru jesu korisničke lozinke koje se, međutim, u današnje vreme pokazuju kao sve neefikasniji sistem provere identiteta (npr. korisnici zbog lakšeg pamćenja najčešće koriste istu lozinku na svim sistemima na kojima imaju naloge). Osetljivost informacija kojima informacioni sistem nudi pristup zahteva dodatnu kontrolu kakvu je danas najoptimalnije dobiti putem SmartCard tehnologije.

SmartCard tehnologija omogućava skladištenje dodatnih parametara na mini-računarima (Smart kartice). Ti parametri (digitalni potpisi, dodatne lozinke...) se mogu koristiti kao daleko pouzdanije rešenje od jednostavnih kratkih nizova karaktera (lozinki). Detaljniji opis SmartCard tehnologije i načina njene upotrebe u informacionom sistemu izlazi van okvira ovog rada ali je ta tehnologija identifikovana kao jedan od daljih pravaca razvoja informacionog sistema.

6.1.3. Kreiranje redundantnih vitalnih komponenti sistema

Ranije pomenuto pitanje pozdanosti (dostupnosti) sistema zavisi od mnoštva faktora i na njega je moguće dati odgovore u vidu većeg broja odgovora (npr. korišćenje visoko-kvalitetnih hardverskih komponenti). Međutim, jedan od osnovnih pristupa jeste uvođenje redundantnih vitalnih komponenti sistema tako da on može nastaviti sa radom u slučaju delimičnog otkazivanja nosećih komponenti. Tri osnovne komponente od kojih zavisi dostupnost informacionog sistema jesu komponente servera baze podataka, komponente aplikativnih servera i komponente komunikacionih kanala. Otkazivanje komponenti aplikativnih servera i komunikacionih kanala može onemogućiti korišćenje IS-a u određenom periodu dok otkazivanje komponenti baze podataka može izazvati trajni gubitak podataka. U skladu sa tim, na serveru baze podataka je instaliran RAID sistem višestrukih kopija podataka i ugrađene su procedure za kreiranje zaštitnih kopija. Dodatna mera zaštite od otkazivanja bi bila i instalacija replikacionih servera. Računarska mreža univerziteta

Singidunum već poseduje redundane komunikacione kanale dok je instalacija redundantnih aplikacionih servera odložena zbog niskog rizika po pitanju dostupnosti i visokog rizika po pitanju bezbednosti.

6.2. Realizacija preostalih korisničkih modula

U trenutku pisanja ovog rada na listi korisničkih modula informacionog sistema u okviru kojih su realizovane predviđene funkcionalnosti spadaju:

- Menadžment
- Studentske službe
- Studenti

što znači da ostale inicijalno predviđene korisničke module:

- Profesori
- Računovodstvo
- Administratori

treba naknadno realizovati. S obzirom na inicijalne korisničke zahteve, kompleksnost i tesnu povezanost svih modula unutar sistema, redosled realizacije je bio uslovljen pomenutim faktorima koji će imati uticaj i u daljem razvoju sistema. Dodatno, pomenuti nerealizovani moduli su izuzetno kompleksni i zahtevaju obimne konsultacije sa adekvatnim učesnicima u poslovnom sistemu, analizu poslovnih procesa, razvoj modela (uz dodatan kriterijum kompatibilnosti sa realizovanim modulima), testiranje na odvojenoj bazi podataka i strogo kontrolisano uvođenje u primenu.

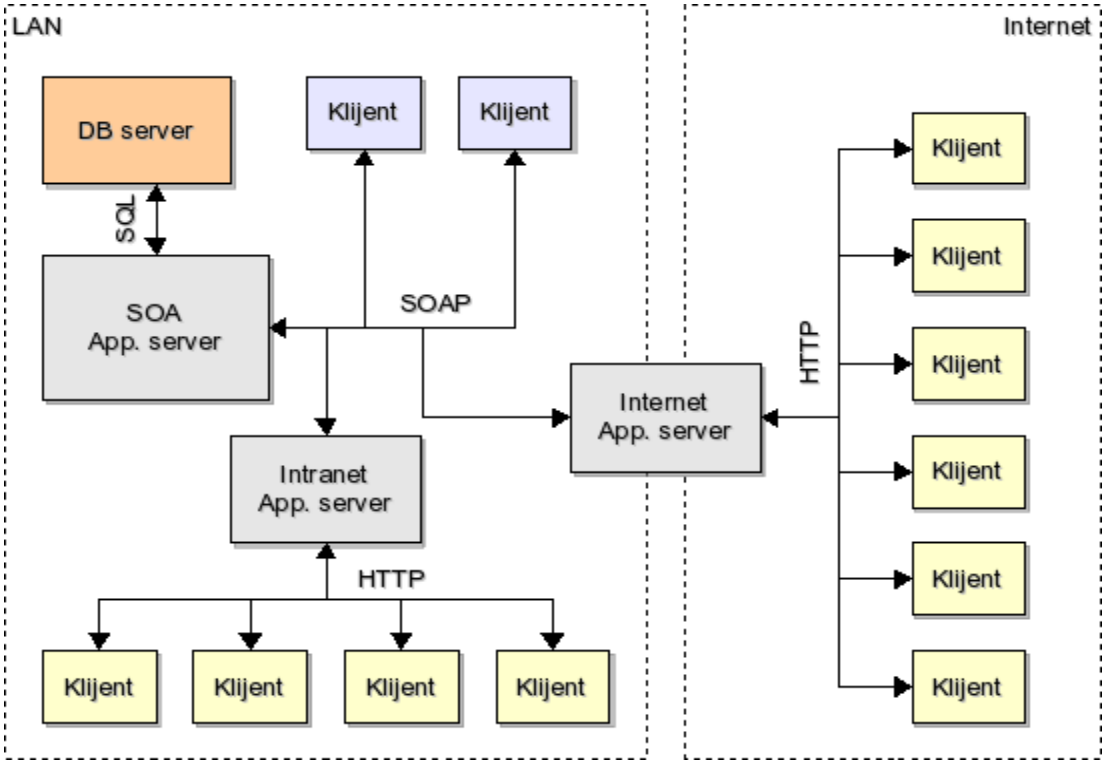
6.3. Iterativno uvođenje novih funkcionalnosti

Kao što je navedeno u delu 6.5 „Povezivanje sa drugim sistemima“ proširivanje skupa funkcionalnosti se može bazirati na dodavanju novih funkcionalnosti direktno u IS ili na kreiranju autonomnih informacionih sistema koji imaju mogućnost da koriste podatke iz baze podataka centralnog IS-a. Prednost dodavanja novih funkcionalnosti direktno u IS se pre svega ogleda u tome što omogućava korišćenje internih, gotovih komponenti sistema i integraciju u postojeći korisnički interfejs. Međutim, ozbiljna opasnost koja stoji iza ovakvog pristupa jeste gubitak elegantnog i jednostavnog dizajna što se može rezultovati glomaznim sistemom sa visokom cenom izmene nekog dela usled mnoštva bočnih efekata.

U skladu sa navedenim, osnovi predlog za dalji razvoj informacionog sistema po pitanju uvođenja novih funkcionalnosti jeste direktno uvođenje u informacioni sistem samo onih funkcionalnosti koje su neodvojivo vezane za već podržane poslovne procese. Ostale funkcionalnosti (tj. funkcionalnosti koje se ne odnose na elementarne poslovne procese) bi bilo najbolje realizovati u vidu zasebnog informacionog sistema (ili podsistema, u zavisnosti od obima i strukture poslovnih procesa koje treba podržati) uz mogućnost korišćenja određenih delova centralne baze podataka.

6.4. Prevođenje na potpunu SOA arhitekturu

U delu 3.1 „Analiza softverskih arhitektura“ kao i u delu 4.2 „Model arhitekture IS-a“ dati su osnovni razlozi tendencije prelaska na punu SOA arhitekturu u okviru budućeg razvoja sistema.

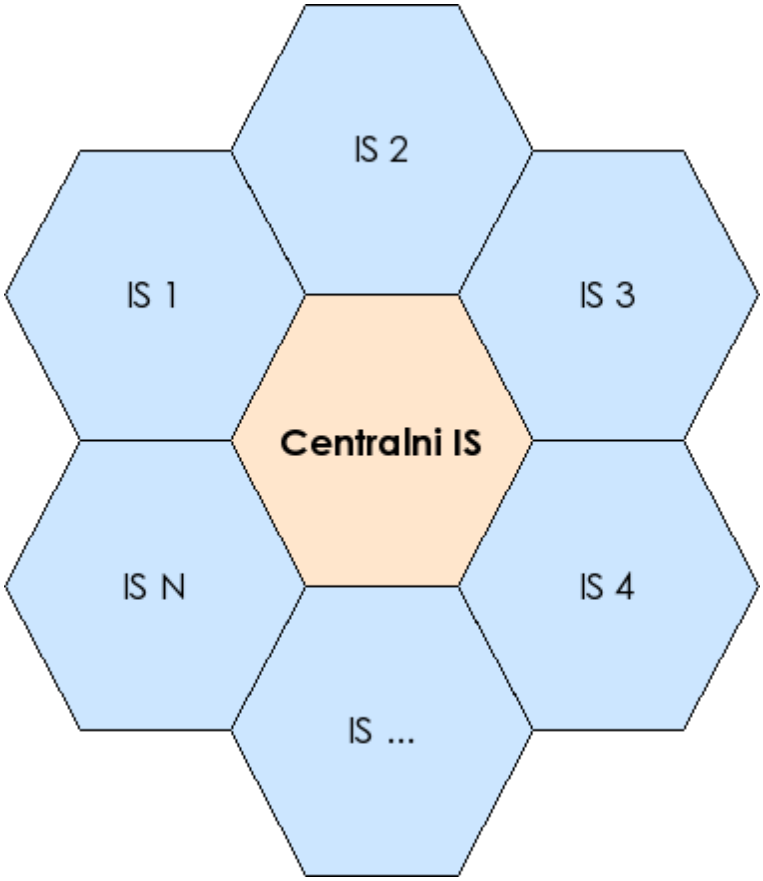


Slika 69 - Mrežna realizacija informacionog sistema baziranog na SOA arhitekturi

Jedan od glavnih razloga za uvođenje SOA interfejsa ka bazi podataka jeste mogućnost potpune kontrole pristupa podacima (pristup bazi je moguć samo putem procedura na SOA interfejs serveru). Osim ove pogodnosti, SOA arhitektura omogućava i potpunu platformsku nezavisnost što omogućava uvođenje novih tehnologija (pre svega tehnologija vezanih za izgradnju korisničkog interfejsa) u izgradnju IS-a. Dodatno, uvođenje novih tehnologija može biti uslovljeno budućom potrebom za izmenom korišćenih tehnologija a SOA arhitektura u tom slučaju omogućava postepenu migraciju [10]. Još jedna bitna prednost SOA arhitekture jeste lako povezivanje sa ostalim softverskim sistemima što je gotovo izvestan scenario kada je u pitanju kompleksna organizacija kakva je univerzitet.

6.5. Povezivanje sa drugim sistemima

Proširivanje skupa funkcionalnosti se može bazirati na dodavanju novih funkcionalnosti direktno u IS ili na kreiranju autonomnih informacionih sistema koji imaju mogućnost da koriste podatke iz baze podataka centralnog IS-a. Prednost dodavanja novih funkcionalnosti direktno u IS se pre svega ogleda u tome što omogućava korišćenje internih, gotovih komponenti sistema i integraciju u postojeći korisnički interfejs. Međutim, ozbiljna opasnost koja stoji iza ovakvog pristupa jeste gubitak elegantnog i jednostavnog dizajna što se može rezultovati glomaznim sistemom sa visokom cenom izmene nekog dela usled mnoštva bočnih efekata.



Slika 70 - Odnos spoljnih IS-a sa centralnim

Kreiranje autonomnih informacionih sistema predstavlja drugačiji pristup koji zahteva razvijene interfejse ka centralnom informacionom sistemu ali ne opterećuje dodatno internu definiciju centralnog informacionog sistema. Ranije opisana SOA arhitektura predstavlja adekvatnu osnovu za ovakav pristup s tim da je on moguć i na druge načine (npr. direktnim povezivanjem sa SUBP centralne baze podataka). Prednost SOA arhitekture je pre svega u tome što omogućava platformsku nezavisnost kao i kontrolu pristupa podacima sopstvenim softverskim interfejsom.

6.5.1. Sistem za automatizovano testiranje studenata

Podaci o aktivnim studentskim nalozima sa pridruženim predmetima i prijavljenim ispitima u ispitnom roku predstavljaju bitnu informaciju u softverskom sistemu za automatizovano testiranje studenata a dostupni su u opisanoj konfiguraciji centralnog informacionog sistema. Sistem za automatizovano testiranje studenata nije samo jedna od mogućih opcija već predstavlja realizovan sistem, izgrađen paralelno sa izradom ovog rada.

Sistem za automatizovano testiranje studenata nosi naziv „UNIS.MTutor“ i projektovan je tako da može da radi nezavisno od centralnog UNIS sistema (koristi sopstvenu bazu podataka) s tim da u takvoj konfiguraciji predmeti (kursevi) i nalozi studenata moraju biti uneti i lokalnu bazu kao i kategorisani u vidu studentskih grupa. Interna baza Unis.MTutor sistema je je izgrađena tako da je format podataka kompatibilan sa formatom podataka u bazi samog UNIS sistema što omogućava laku integraciju u budućnosti.

Dodatna pogodnost koja proizilazi iz integracije ova dva sistema jeste i mogućnost automatskog kreiranja grupa studenata koje imaju pravo da pirstupe određenom testu putem UNIS.MTutor sistema a u skladu sa informacijama o prijavljenim ispitima u bazi podataka UNIS sistema. Dodatno, zajedničko korišćenje informacija o predmetima može omogućiti i automatsko kreiranje testova u skladu sa terminom ispita u ispitnom roku.

Za razvoj UNIS.MTutor sistema korišćena je ista, ranije opisana platforma kao i za UNIS sistem - EMS - uz dodatak intenzivne upotrebe Java tehnologije putem PHP-Java bridža. Unis.MTutor sistem je takođe izrgrađen modularno i sastoji se od sledećih modula:

- **Administrator**
  Modul namenjen za administraciju, kreiranje novih predmeta i studentskih i tutorskih naloga.
- **Tutor (nastavnik)**
  Modul namenjen nastavnicima, omogućava struktuiranje predmeta, unos pitanja, definisanje provera (testova), saradnika...
- **Student**
  Studentski modul koji omogućava pristup testovima i testiranje. Za potrebe ovog modula napravljena je klijentska desktop aplikacija sa funkcionalnostima sličnim standardnim Web brauzerima.
- **Dispatcher**
  Sistemski modul koji omogućava skaliranje sistema tj. kreiranje klastera servera koji opslužuju studente pri testiranju.

UNIS.MTutor sistem je ušao u prvu fazu testiranja i korišćenja početkom marta 2007. godine i od tada pa do pisanja ovog teksta je putem njega testirano više stotina različitih studenata u više provera i putem oko 50.000 otvorenih pitanja i 250.000 odgovora. Više informacija o UNIS.MTutor sistemu se trenutno može dobiti samo od strane autora, mr. Gorana Šimića i Aleksandra Jevremovića, saradnika Fakulteta za Poslovnu Informatiku.

6.5.2. Sistem za Distance Learning

Postojanje centralne baze podataka koja sadrži informacije o studentskim nalozima, programima, predmetima i sl. predstavlja osnovu za izgradnju sistema za studiranje putem Interneta (Distance Learning). Dodatno, baza podataka ranije opisanog UNIS.MTutor sistema nudi dodatno struktuiranje predmeta putem podele na oblasti i lekcije. Postojanje ovakve baze podataka i opisanih funkcionalnosti UNIS i UNIS.MTutor sistema nudi solidnu osnovu za izgradnju dodatnog modula koji bi ponudio mogućnost studentima da putem Interneta prate predavanja, proveravaju svoj napredak kroz testove i zadatke, komuniciraju međusobno i sa profesorima, učestvuju u radnim grupama i sl.

7. Zaključak

U ovom radu je obrađen proces izgradnje informacionog sistema za potrebe univerziteta Singidunum. Rad se sastoji od dva dela - funkcionalnog softverskog rešenja i dokumentacije vezane za analizu potreba i tehnologija, izradu modela rešenja, proces realizacije i za dalje pravce razvoja sistema.

Praktični deo ovog rada (tj. varijanta koja je obezbeđivala osnovne funkcionalnosti) je stavljen u upotrebu krajem 2006. godine i od tada pa do zaključivanja ovog rada je putem njega 58 studenata univerziteta Singidunum (koji studiraju putem DLS sistema) postavilo 106 prijava ispita u okviru kojih je prijavljeno 257 ispita u 12 ispitnih rokova na 3 fakulteta (Fakultet za Finansijski Menadžment i Osiguranje, Fakultet za Poslovnu Informatiku i Fakultet za Turistički i Hotelijerski Menadžment). Ovaj podatak govori i to da je putem sistema ostvaren promet od 1285 evra (u dinarskoj protivvrednosti) koji je uredno evidentiran u službi računovodstva univerziteta. Iako ove cifre ne deluju značajno naspram preko 5.000 studenata univerziteta, takav uzorak je bio sasvim dovoljan da se potvrdi stabilnost i efikasnost sistema.

Glavni problem koji je rešen izgradnjom informacionog sistema jeste prijavljivanje ispita za studente koji ne žive u Beogradu (ili čak ni u Srbiji) i nemaju mogućnost da prijavljuju ispite u zgradi fakulteta. Međutim, osim za rešavanje tog problema, sistem je projektovan tako da podrži i ostale aktivnosti poslovnog sistema univerziteta (od upisa studenata pa do provere položenih ispita i dodele statusa „diplomirao“ studentu) tako da se može primeniti i na ostale studente (npr. studente koji studiraju putem klasičnog načina pohađanja nastave) što bi znatno smanjilo opterećenje studentskih službi ali i pružilo tačnije informacije u kraćem vremenskom periodu.

Modularni dizajn sistema omogućava fazni razvoj bez ugrožavanja ostvarenih funkcionalnosti a opisana tendencija prelaska na punu SOA arhitekturu omogućava povezivanje centralnog informacionog sistema sa ostalim softverskim sistemima koji će biti uvođeni u upotrebu. Jedan od sistema sa kojim je centralni informacioni sistem već povezan jeste i opisani Unis.MTutor sistem za automatizovano testiranje studenata. Dodatno, sistem nudi osnovu za izgradnju sopstvenog sistema za DLS, bilo u vidu zasebnog sistema koji koristi centralnu bazu podataka, bilo u vidu modula informacionog sistema.

Osim praktične vrednosti rada, njegov teorijski deo nudi preciznu dokumentaciju o njegovoj realizaciji i načinu korišćenja pa se može koristiti i kao osnovni priručnik za izmenu i upotrebu. Dodatno, u radu su opisane sve korišćene tehnologije kao i tehnologije koje nisu ušle u izbor za upotrebu tako da rad sadrži i presek trenutno aktuelnih informacionih tehnologija (pre svega onih namenjenih za izgradnju Web aplikacija). Zaključno, VI deo rada ukazuje na oblasti informacionog sistema u kojima se predviđa potreba za pojačavanjem u bliskoj budućnosti (pre svega bezbednosni aspekti usled povećanja broja korisnika) kao i moguća proširenja (na nivou arhitekture i dodavanja konkretnih funkcionalnosti).

Literatura

[1] Uvod u računarske mreže / Mladen Veinović, Aleksandar Jevremović. - Beograd : Univerzitet Singidunum, Fakultet za poslovnu informatiku, 2007.

[2] TCP/IP Illustrated, Volume 1: The Protocols, Addison-Wesley, 1994, ISBN 0-201-63346-9.

[3] SOAP Version 1.2 Specification, W3C, http://www.w3.org/TR/soap

[4] Smarty template engine dokumentacija, http://smarty.php.net/manual/en/

[5] PHP dokumentacija, http://www.php.net/docs.php

[6] CSS dokumentacija, W3C, http://www.w3.org/Style/CSS/

[7] (X)HTML dokumentacija, W3C, http://www.w3.org/MarkUp/

[8] The Philosophy of JavaScript, anonimus, http://20bits.com/2007/03/08/the-philosophy-of-javascript/

[9] A Spiral Model of Software Development and Enhancement, Barry W. Boehm, TRW Defence System Group, 2002.

[10] SOA Reference Model TC, OASIS, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=soa-rm

[11] XML-RPC dokumentacija, http://www.xmlrpc.com/

[12] WDDX dokumentacija, http://www.openwddx.org/

[13] JSON dokumentacija, http://www.json.org/

[14] Hessian dokumentacija, Caucho Technology, http://www.caucho.com/resin-3.1/index.xtp

[15] MySQL 5.1 Reference Manual, http://dev.mysql.com/doc/refman/5.1/en/index.html