UNIVERZITET U NOVOM SADU
TEHNIČKI FAKULTET "MIHAJLO PUPIN"
ZRENJANIN

Mr Ljubica Kazi
Mr Zoltan Kazi
Prof. dr Biljana Radulović

INFORMACIONI SISTEMI 1 i INFORMACIONI SISTEMI 2

- praktikum za vežbe -

Zrenjanin, 2013.

Autori:
Mr Ljubica Kazi
Mr Zoltan Kazi
Prof. dr Biljana Radulović

Recenzenti:
Prof. dr Dragica Radosav
Prof. dr Ivana Berković
Prof. dr Branko Markoski

Izdavač:
Tehnički fakultet «Mihajlo Pupin», Zrenjanin

Za izdavača:
prof. dr Milan Pavlović, dekan

Nastavno-naučno veće Tehničkog fakulteta «Mihajlo Pupin» u Zrenjaninu donelo je odluku 11.12.2013. godine da se rukopis ove knjige može koristiti kao udžbenik Fakulteta.

Tehnička obrada:
Mr Ljubica Kazi

Dizajn korice:
Mr Ljubica Kazi

Tiraž:
200 komada

Elektronski udžbenik na CD-u
Biblioteka „Udžbenici", broj 179, školska 2013/2014. godina

ISBN (elektronski): 978-86-7672-215-0

CIP – Каталогизација у публикацији
Библиотека Матице српске, Нови Сад

004(075.8)(076)

КАЗИ, Љубица
Informacioni sistemi 1 i Informacioni sistemi 2 [Elektronski izvor] / Ljubica Kazi, Zoltan Kazi, Biljana Radulović. - Zrenjanin : Tehnički fakultet "Mihajlo Pupin", 2013. - 1 elektronski optički disk (CD-ROM) : tekst ; 12 cm. - (Biblioteka Udžbenici ; br. 179)

Nasl. sa naslovnog ekrana. Tiraž 200. – Bibliografija.

ISBN 978-86-7672-215-0
1. Кази, Золтан 2. Радуловић, Биљана
а) Информациони системи – Задаци
COBISS.SR-ID 282456583

PREDGOVOR

Ovaj praktikum za vežbe je prvenstveno namenjen slušaocima kurseva na osnovnim studijama »Informacioni sistemi 1« i »Informacioni sistemi 2«. Predstavlja osnovni materijal za pripremu praktičnog dela ispita (kolokvijuma i seminarskih radova) iz navedenih nastavnih predmeta. S obzirom da proces razvoja predstavlja nedeljivu celinu (implementacija se nadovezuje na prethodno projektovanje), pripremljen je objedinjen praktikum.

U okviru navedenog materijala dat je opis metoda, heuristika (navedene u tekstu kao napomene) i alata u projektovanju i implementaciji informacionih sistema. Primerima i rešenim zadacima su ilustrovani rezultati pojedinih faza razvoja informacionog sistema. Za izradu zadataka korišćeni su alati: CASE alat Power Designer verzije 15, DBMS alati MS Access 2000, MS SQL Server 2008 R2 i Oracle 11g Express, kao i programsko razvojno okruženje Visual Studio .NET 2010.

Navedeni materijal je nastao kao rezultat višegodišnjeg rada autora u nastavi pomenutih i srodnih kurseva. Predstavlja objedinjen materijal koji je u parcijalnoj elektronskoj formi u toku višegodišnjeg nastavnog rada bio dostupan studentima sa fakultetskog file-servera i web sajta fakulteta.

Odeljak koji se odnosi na tehnološki nezavisne aktivnosti procesa razvoja pripada sadržaju vežbi na predmetu »Informacioni sistemi 1«, dok se odeljak koji se odnosi na tehnološki zavisne aktivnosti, odnosno implementaciju, odnosi na sadržaj vežbi nastavnog predmeta »Informacioni sistemi 2«.

Zahvalnost za kritički pregled rukopisa dugujemo recenzentima.

U Zrenjaninu, 2013.

Autori

SADRŽAJ

1. **RAZVOJ INFORMACIONIH SISTEMA** 1
1.1. Definicija i funkcije informacionih sistema 1
1.2. Problemi i principi razvoja informacionih sistema 3
1.3. Arhitektura informacionog sistema 4
1.4. Vrste informacionih sistema 7
1.5. Proces razvoja informacionog sistema 9
1.5.1. Elementi upravljanja projektima 10
1.5.2. Faze i aktivnosti procesa razvoja informacionog sistema 12
1.6. Zakonska regulativa u oblasti informacionih sistema 16

2. MODEL PRAKTIČNE NASTAVE U OBLASTI RAZVOJA INFORMACIONIH SISTEMA 20

3. ANALIZA ORGANIZACIONOG SISTEMA I PLANIRANJE PROJEKTA 25
3.1. Tehnike upoznavanja sa organizacionim sistemom 25
3.2. Upoznavanje sa semantikom poslovanja 25
3.3. Snimak stanja postojećeg informacionog sistema i planova razvoja 26
3.4. Specifikacija zahteva korisnika o potrebnim karakteristikama novog sistema 26
3.5. Definisanje elemenata projekta 28
3.6. REŠENI ZADACI 29

4. MODELOVANJE POSLOVNIH PROCESA 41
4.1. Model toka poslovnih procesa 41
4.2. Dijagram toka podataka 44
4.3. Rečnik podataka 49
4.4. REŠENI ZADACI 51

5. DIZAJN FUNKCIONALNOG ASPEKTA PREDLOGA REŠENJA 58
5.1. Preslikavanje primitivnih poslovnih procesa u softverske funkcije 58
5.2. Dizajn softverskog rešenja uz definisanje prioriteta razvoja 59
5.3. Dijagram slučajeva korišćenja 63
5.4. Specifikacija slučaja korišćenja 68
5.5. REŠENI ZADACI 71

6. KREIRANJE MODELA PODATAKA 81
6.1. KREIRANJE KONCEPTUALNOG MODELA PODATAKA 81
6.1.1. Pristupi i metode za kreiranje konceptualniog modela podataka 81
6.1.2. Kreiranje konceptualnog modela u CASE alatu 83
6.1.3. Kreiranje podmodela konceptualnog modela podataka 90
6.1.4. Kriterijumi vrednovanja konceptualnog modela podataka 91
6.2. KREIRANJE FIZIČKOG MODELA PODATAKA 92
6.2.1. Podešavanje alternativnog ključa - unique indeksa 94
6.3. GENERISANJE BAZE PODATAKA 97
6.3.1. Kreiranje SQL skripta za generisanje baze podataka 98
6.3.2. Direktno generisanje baze podataka iz CASE alata 101
6.4. KREIRANJE OBJEKTNOG MODELA PODATAKA 106
6.4.1. Kreiranje dijagrama klasa direktnim modelovanjem 106
6.4.2. Kreiranje dijagrama klasa automatskom transformacijom 106
6.4.3. Generisanje programskog koda klasa 110
6.6. REŠENI ZADACI 116

7. RAD SA BAZOM PODATAKA 131
7.1. STRUKTURA I TIPOVI PODATAKA U RAZLIČITIM DBMS 131
7.2. MS SQL SERVER BAZA PODATAKA 133
7.2.1. Radno okruženje 133
7.2.2. Kreiranje baze podataka 134
7.2.2.1. Kreiranje tabela i relacija baze podataka vizualnim alatima DBMS-a 134
7.2.2.2. Kreiranje elemenata baze podataka SQL naredbama 140
7.2.3. Izdvajanje fajla baze podataka i korišćenje sa druge lokacije 148
7.2.4. Rad sa podacima 150
7.2.4.1. Ažuriranje podataka 150
7.2.4.2. Izdvajanje i pretraga podataka 152
7.2.4.3. PRIMERI select upita 155
7.3. ORACLE BAZA PODATAKA 161
7.3.1. Radno okruženje 161
7.3.2. Kreiranje baze podataka 162
7.3.2.1. Kreiranje baze podataka iz vizuelnog radnog okruženja 162
7.3.2.2. Kreiranje baze podataka pomoću SQL naredbi 168
7.3.3. Rad sa podacima 169

8. KONEKCIJA NA BAZU PODATAKA 171
8.1. Konekcija iz korisničkog interfejsa direktnim povezivanjem sa bazom podataka 171
8.2. Konekcija programskim kodom pomoću gotovih klasa 172

9. KREIRANJE BIBLIOTEKE KLASA SLOJA PODATAKA 174
9.1. Generisanje klasa korišćenjem Entity Framework-a 174
9.2. Kreiranje sopstvenih opštih tehnoloških klasa 179
9.2.1. Klase za rad sa TXT, XML i XLS fajlovima 181
9.2.2. Klase za rad sa MS Access bazom podataka 183
9.2.3. Klase za rad sa MS SQL Server bazom podataka 186
9.2.4. Klase za rad sa ORACLE bazom podataka 190
9.2.5. Klase za rad sa transakcijama 191
9.2.5.1. Korišćenje gotovih klasa za rad sa transakcijama 192
9.2.5.2. Korišćenje gotovih klasa za rad sa distribuiranim transakcijama 192
9.2.5.3. Kreiranje sopstvene klase za rad sa transakcijama 194
9.3. Biblioteka klasa podataka 195

10. IMPLEMENTACIJA WINDOWS APLIKACIJE u Visual Studio.NET 201
10.1. Radno okruženje 201
10.2. Fajlovi u strukturi aplikacije 204
10.3. Globalna podešavanja i otvaranje konekcije 205
10.4. Pokretanje aplikacije 207
10.5. Prijava na sistem 208
10.6. Glavni meni 210
10.7. Administracija korisnika sistema 212
10.8. Rad sa šifarnikom 214
10.9. Osnovni ekran za rad 216
10.9.1. Osnovna podešavanja i stanja forme 216
10.9.2. Ažuriranje podataka 220
10.9.3. Navigacija i pozicioniranje 227
10.9.4. Filtriranje podataka 230

10.9.5. Rad sa XML i XLS fajlovima 232
10.9.6. Štampa 232
10.9.6.1. Kreiranje izveštaja koristeći Print document 233
10.9.6.2. Rad sa rdlc izveštajima 237
10.10. Rad sa grafikonima 239
10.11. Forma Master–detail 242

11. IMPLEMENTACIJA ASPX WEB APLIKACIJE u Visual Studio.NET 246
11.1. Radno okruženje i kreiranje ASPX aplikacije 246
11.2. Konekcija na bazu podataka putem klasa sloja podataka 248
11.3. Kreiranje forme i povezivanje sa menijem 250
11.4. Tabelarni prikaz podataka sa filtriranjem 251
11.5. Unos podataka na formi sa combo box-om 253
11.6. Rad sa sesijama 255

12. LITERATURA 256

1 . RAZVOJ INFORMACIONIH SISTEMA

1.1. Definicija i funkcije informacionih sistema

»SISTEM predstavlja skup povezanih elemenata koji predstavljaju jasno izdvojenu celinu sa određenim ponašanjem, gde je funkcionisanje svakog elementa podređeno ostvarenju zajedničkih ciljeva. Ključne karakteristike sistema su: skup elemenata, postoje relacije između elemenata i interakcije sa okruženjem, uređenost elemenata u strukturi, uređenost, organizovanost sa jasnom ulogom elemenata u doprinosu ciljevima celine, atributi kao karakteristike elemenata, strukture, funkcionisanja i sistema kao celine, karakteristike sistema kao celine (statičke i dinamičke), izdvojena celina od okruženja, funkcionisanje (pravila, funkcija, zakon), ponašanje – niz stanja, procesi, ulaz/izlaz, cilj.« [1] Prema teoriji sistema, „sistem je uređeni skup elemenata sa određenom interakcijom, ciljem i uzajamnim uticajem posmatranog sistema i okoline" [2] .

Pod ORGANIZACIONIM SISTEMOM se prema [2] smatra ljudska tvorevina – ljudsko udruživanje radi ostvarivanja zajedničkog cilja, svesno delovanje čoveka kojim se usklađuju svi činioci (ljudski i materijalni – sredstva za rad, predmeti rada) u svrhu postizanja najvećeg proizvodnog učinka s najmanjim utroškom ljudskog rada i materijala. Organizaciju sačinjavaju dva osnovna podsistema: socijalni i tehnički. [3]

Među teorijskim pristupima proučavanju organizacije [3] klasične teorije organizacije zasnovane su na mehanicističkom i determinisičkom pristupu, usredsređene prvenstveno na strukturu organizacije, a manje na njenu dinamiku i razvoj), dok savremeni pristupi razmatraju organizaciju SA STANOVIŠTA TEORIJE SISTEMA – ORGANIZACIONI SISTEM je otvoren dinamički društveno-tehnički sistem sa sledećim podsistemima potrebnim za njen opstanak i razvoj: podsistem proizvodnje, podsistem podržavanja (obezbeđuje resurse), podsistem održavanja (održava strukturnu celovitost), podsistem prilagođavanja (odgovoran za prilagođavanje na promene, npr. odeljenje za istraživanje i razvoj), podsistem upravljanja (koordinira, kontroliše i usmerava ostale podsisteme).

Prema [4], POSLOVNI SISTEM predstavlja bilo koji ekonomski subjekt. Svaki poslovni sistem se sastoji iz tri podsistema:

1. Izvršni podsistem
2. Upravljaki podsistem
3. Informacioni podsistem.

Opšti funkcionalni model svakog organizacionog sistema može se predstaviti na sledeći način:

<table>
<tr><td colspan="2">UPRAVLJAČKI PODSISTEM</td><td rowspan="4">INFORMACIONI SISTEM</td></tr>
<tr><td>IZVRŠNI PODSISTEM</td><td></td></tr>
<tr><td>OSNOVNE delatnosti</td><td>POMOĆNE delatnosti</td></tr>
<tr><td>Podsistem ostvarivanja osnovne svrhe postojanja i namene sistema</td><td>Podsistem obezbeđivanja i obrade resursa</td></tr>
</table>

Slika 2. Opšti funkcionalni model organizacionog sistema

Delatnosti izvršnog podsistema odnose se na pomoćne delatnosti obezbeđivanja i obrade resursa za rad osnovne delatnosti, čije se aktivnosti odvijaju radi ostvarivanja svrhe postojanja i namene sistema. Uloga informacionog sistema je da obezbedi informacionu podršku za rad svih drugih podsistema – izvršnog i upravljačkog.

Da bismo definisali pojam informacionog sistema, potrebno je definisati pojam INFORMACIJE. Informacija nastaje u procesu kroz niz aktivnosti, a bazira se na podacima. Navedeni proces može se opisati kroz sledeće korake:

1. Postojanje realnog događaja ili pojave
2. Uočavanje karakteristika realnog događaja ili pojave
3. Evidentiranje karakteristika realnog događaja ili pojave.Opis karakteristika nekog događaja ili pojave opisan nekim jezikom predstavlja PODATAK. [3]
4. Obrada podataka – izračunavanja, formatiranja, oblikovanje radi prezentovanja čoveku
5. Tumačenje (razumevanje) i vrednovanje (dodeljivanje nekog značaja radi eventualne osnove za aktivnosti) od strane čoveka predstavlja proces kreiranja INFORMACIJE. Informacija predstavlja protumačen i vrednovan skup podataka od strane čoveka [1] koji kazuje nešto ranije nepoznato primaocu, koristi se u primaočevom odlučivanju i redukuje neizvesnost. »Reč informacija je izvedena iz latinske reči informatio, što znači pojam, poruka, skup spoznaja, predstava. Informacija je značenje koje čovek pripisuje podacima u skladu sa opštim dogovorima.» [3]

Informacioni sistem predstavlja podsistem organizacionog sistema i smatra se jednom od infrastrukturnih komponenti organizacije. »Polazeći od sistemskog pristupa, INFORMACIONI SISTEM možemo definisati kao sređeni skup metoda, procesa i operacija za prikupljanje, čuvanje, obradu, prenošenje i distribuciju podataka u okviru jedne organizacije, uključujući i opremu koja se u te svrhe koristi i ljude koji se tim aktivnostima bave.» [5] Informacioni sistem je sistem u kojem su elementi međusobno i sistem kao celina sa okruženjem povezani informacionim tokovima. [1]

Osnovne funkcije informacionih sistema su [4]:
1. Akvizicija podataka – registrovanje podataka
2. Skladištenje podataka – beleženje na medijima, arhiviranje
3. Obrada podataka – izračunavanja, transformacije (u različite oblike i formate)
4. Prezentovanje podataka – analitički prikaz, tabelarni prikaz, statistički prikaz, grafički prikaz, štampa
5. Razmena podataka – između računara u mreži, između različitih aplikacija (interoperabilnost)

„POSLOVNI INFORMACIONI SISTEM povezuje upravljačke i izvršne funkcije tako što formirane informacije na osnovu poslovnih podataka stavlja na raspolaganje upravljačkom podsistemu. Te zadatke poslovni informacioni sistem ostvaruje prikupljanjem, obrađivanjem i sređivanjem podataka u cilju dobijanja poslovnih informacija i njihovim stavljanjem na raspolaganje upravljačkom podsistemu. Poslovni informacioni sistem ima četiri osnovne funkcije:
1. obezbeđivanje informacija o prošlim poslovnim zbivanjima
2. obezbeđivanje informacija o budućim poslovnim zbivanjima
3. obezbeđivanje kontrolnih informacija
4. obezbeđivanje analitičkih informacija.» [4]

Poslovni informacioni sistem pre svega služi za podršku odlučivanju na svim nivoima – operativnom, taktičkom i strateškom. Neke od teorija upravljanja [3] definišu sledeće funkcije upravljanja: predviđanje, organizovanje, usmeravanje i kontrolisanje, odnosno upravljačke uloge: 1 – interpersonalne, 2 – informacione (primanje i analiza podataka u organizaciji i sa okruženjem), 3- odlučivanje (inovator, inicijator promena, rešavanje problema, raspoređivanje resursa, pregovaranje), 4 – planske (određivanje strategije, struktuiranje organizacije, planiranje), 6 – monitoring (kontrolisanje, nadgledanje) 7- koordinisanje, 8 – marketing.

Literatura

[1] Radulović B, Kazi Lj, Kazi Z: *Informacioni sistemi – odabrana poglavlja*, Tehnicki fakultet Mihajlo Pupin Zrenjanin, 2011.
[2] Ristić D: *Proizvodno poslovni sistemi – osnovi organizacije rada*, Edicija YU Experti, IBN centar, Beograd, 1991.
[3] Balaban N, Ristić Ž, Đurković J: *Principi informatike*, Savremena administracija, 1996.
[4] Stankić R: *Poslovna informatika*, Ekonomski fakultet Univerziteta u Beogradu, 2005.
[5] Jauković M: *Uvod u informacione sisteme*, Tehnička knjiga, Beograd, 1992.

1.2. Problemi i principi razvoja informacionog sistema

Osnovni problemi koji se javljaju u razvoju informacionog sistema odnose se na:

1. Problem SLOŽENOSTI POSTOJEĆIH SISTEMA (sektori, organizacione jedinice, službe i sl.). Rešava se na dva načina: PODELOM SISTEMA NA PODSISTEME (PODCELINE) i PODELOM RAZVOJA INFORMACIONOG SISTEMA NA FAZE u postupku projektovanja.

> >>>>>> DEKOMPOZICIJA : AKTIVNOSTI/PODACI matrica - PODSISTEMI
>
> >>>>>> FAZE: IDEJNI PROJEKAT (cilj, obuhvat, ograničenja, zahtevi, studija opravdanosti i izvodljivosti), GLAVNI PROJEKAT (sistemska analiza, dizajn), DETALJNI PROJEKAT, IMPLEMENTACIJA, TESTIRANJE, ODRŽAVANJE

2. Problem KOMUNIKACIJE SA KORISNIKOM (poznaje problematiku poslovnog i organizacionog sistema). Korisnik definiše svoje potrebe i zahteve, a projektanti definišu način na koji će se potrebe korisnika zadovoljiti. Ovaj problem rešava se korišćenjem zajedničkog, JEDNOSTAVNOG JEZIKA SIMBOLA i grafičkih oblika koji su dovoljno jednostavni i razumljivi korisniku. Takođe, PROTOTIPSKI PRISTUP omogućuje dobijanje upotrebljivog rešenja nad kojim korisnik može detaljnije izraziti svoje zahteve povodom funkcionalnih karakteristika softvera.

> >>>>>> MODELOVANJE , dijagrami, rečnik termina, rečnik podataka
>
> >>>>>> CASE ALATI I RAD (rapid application development) ALATI
>
> >>>>>> AGILNE METODE, PROTOTIPSKI PRISTUP …

3. Problem OGRANIČENJA (novac, vreme). Rešava se tako što se za informacioni sistem utvrdi logička struktura celog sistema (model) a implementacija (uvođenje i primena) zatim sledi deo po deo (definišu se PODSISTEMI ZA KOJE SE ODREĐUJU PRIORITETI IMPLEMENTACIJE).
4. Problem UPRAVLJANJA RAZVOJEM. U projektovanje informacionog sistema, kao deo projektantskog tima je uključeno najviše rukovodstvo organizacije, koje mora da da podršku za uvođenje novog informacionog sistema. Problem se javlja u slučaju kada je projekat urađen, projektantski tim je krenuo u realizaciju pojedinačnih zadataka a rukovodstvo organizacije se promeni ili se u toku vremena promeni organizaciona struktura organizacije. Time se menjaju početni uslovi koji utiču na završetak posla. Problem se rešava postojanjem pisanog traga o zahtevima korisnika – SPECIFIKACIJA SISTEMA ili SPECIFIKACIJA ZAHTEVA KORISNIKA (Dokument kojim korisnik izražava svoje zahteve). Međutim kako korisnik ne poznaje nove mogućnosti informacionog sistema, specifikacija ne mora biti nepromenljiva i konačna jer korisnik ne mora da zna šta želi!

Osnovni principi u razvoju informacionih sistema:

1. VREME – potrebno je posmatrati promene u realnom sistemu u vremenu i omogućiti stabilnost informacionog sistema – INFORMACIONI SISTEM TREBA DA ŽIVI U VREMENU I ADAPTIRA SE U SKLADU SA PROMENAMA ORGANIZACIONOG SISTEMA KOJI JE POD STALNIM UTICAJEM OKRUŽENJA IZLOŽEN STALNIM PROMENAMA. Ovo se postiže

ADAPTIBILNOŠĆU, tj. parametrizovanjem aplikacija, uopštavanjem realnih objekata, šifarnicima u bazi podataka.

2. KORISNIK – cilj projektovanja i implementacije IS-a su ZADOVOLJENJE SPECIFIKACIJE ZAHTEVA KORISNIKA koji zna šta želi a ne zna da to realizuje. Zadovoljenje ovog principa se postiže kroz odgovarajući pristup u projektovanju, kroz UČEŠĆE RUKOVODSTVA I ZAPOSLENIH u projektovanju i kroz obuku korisnika za rad u novom sistemu.
3. PROJEKTOVANJE ODOZGO A IMPLEMENTACIJA SA DNA – SISTEM SE SAGLEDA KAO CELINA, podeli se na podsisteme, projektuje i zatim se IMPLEMENTIRA JEDAN PO JEDAN PODSISTEM sa uvažavanjem činjenice da podsistem nije izolovan, već pripada celini.
4. EKONOMIČNOST – obezbediti IS-u da zahteva MINIMUM UTROŠKA VREMENA I NOVCA u fazi PROJEKTOVANJA, IMPLEMENTACIJE I KORIŠĆENJA. Ovaj princip podrazumeva USER FRIENDLY KORISNIČKI INTERFEJS, čime se povećava produktivnost rada korisnika sistema, ali i mnoge tehnike brzog razvoja softvera (PROTOTIPSKI PRISTUP, OBJEKTNO ORJENTISANA VIŠESLOJNA ARHITEKTURA itd.).
5. STANDARDI – međunarodni sistem standarda ISO 9000 - 3 Razvoj softvera u praksi, ISO/IEC 12207 Faze i aktivnosti životnog ciklusa RAZVOJA SOFTVERA, Objektno orijentisani dizajn softvera i ISO/IEC 9126, Information Technology – Software Product Evaluation – Quality Characteristics and Guideline for their Use. Ovaj princip podrazumeva i definisanje STANDARDA I PROPISA U KONKRETNOJ OBLASTI PRIMENE SISTEMA. Softveri IS-a trebaju da imaju podršku za standarde, treba da poseduju STANDARDIZOVANE MASKE, EKRANE, IZVEŠTAJE i sl.

1.3. Arhitektura informacionog sistema

»Arhitektura se definiše kao osnovna organizacija sistema sadržana u njegovim komponentama, njihova međusobna veza i veza sa okruženjem i principi koji vladaju razvojem i evolucijom sistema.« ([1])

»Informacioni sistemi rastu i razvijaju se u različitosti tehnologije, domena primene, broja korisnika, geografskoj distribuiranosti, broja funkcionalnih jedinica, komponenti i veza između njih. Mnoštvo arhitekturalnih šablona (patterna) pojavili su se u nastojanju uvođenja reda. Dizajn arhitekture ima ključnu ulogu u razvoju informacionog sistema. To je prvi korak kojim se specifikacije zahteva pretvaraju u radni sistem koji se sastoji od hardvera i softvera.« [2]

»Arhitektura informacionog sistema je deo šireg skupa arhitektura i modela koje su relevantne za organizaciju. Na različitim nivoima arhitekture, možemo razlikovati:

- Arhitekturu organizacije/preduzeća (Enterprise Architecture)
- Arhitekturu informacionog sistema (ISA – Information System Architecture)
- Softversku arhitekturu (SWA – Software Architecture).« [3]

Arhitektura organizacije-preduzeća je grupa modela koji su definisani za dobijanje koherentne i razumljive slike o organizaciji-preduzeću. [5] Ovi modeli definišu različite perspektive i poglede sa kojih se kompanija razmatra, fokusirajući se na neke aspekte i ignorišući druge, u cilju smanjenja kompleksnosti [6]. «Model kompanije može da sadrži nekoliko dijagrama aktivnosti, dijagrama procesa, dijagrama organizacije, dijagrama protoka informacija i dijagrama ponašanja u organizaciji. Arhitektura organizacije se smatra širim konceptom od arhitekture ISA, koja uključuje poslovne strategije i procese izvan modela informacionog sistema koji ih podržavaju.» [3] Obično na nivou arhitekture preduzeća, IS se smatra jednostavnim resursom koji se koristi u poslovanju (pored ljudskih resursa, opreme, materijala itd) [7, 8]. Osnovna oblast istraživanja softverske arhitekture odnosi se na način interne izgradnje programa ili komponenti aplikacije [4]. Na ovom nivou važni aspekti se odnose na objekte i klase potrebne za implementaciju softvera. 1980-tih godina, softverska arhitektura i ISA su smatrane sinonimima. Tek se u poslednjih desetak godina pojavila potreba za manipulaciju konceptima koji prevazilaze opis kako je sistem interno izgrađen.

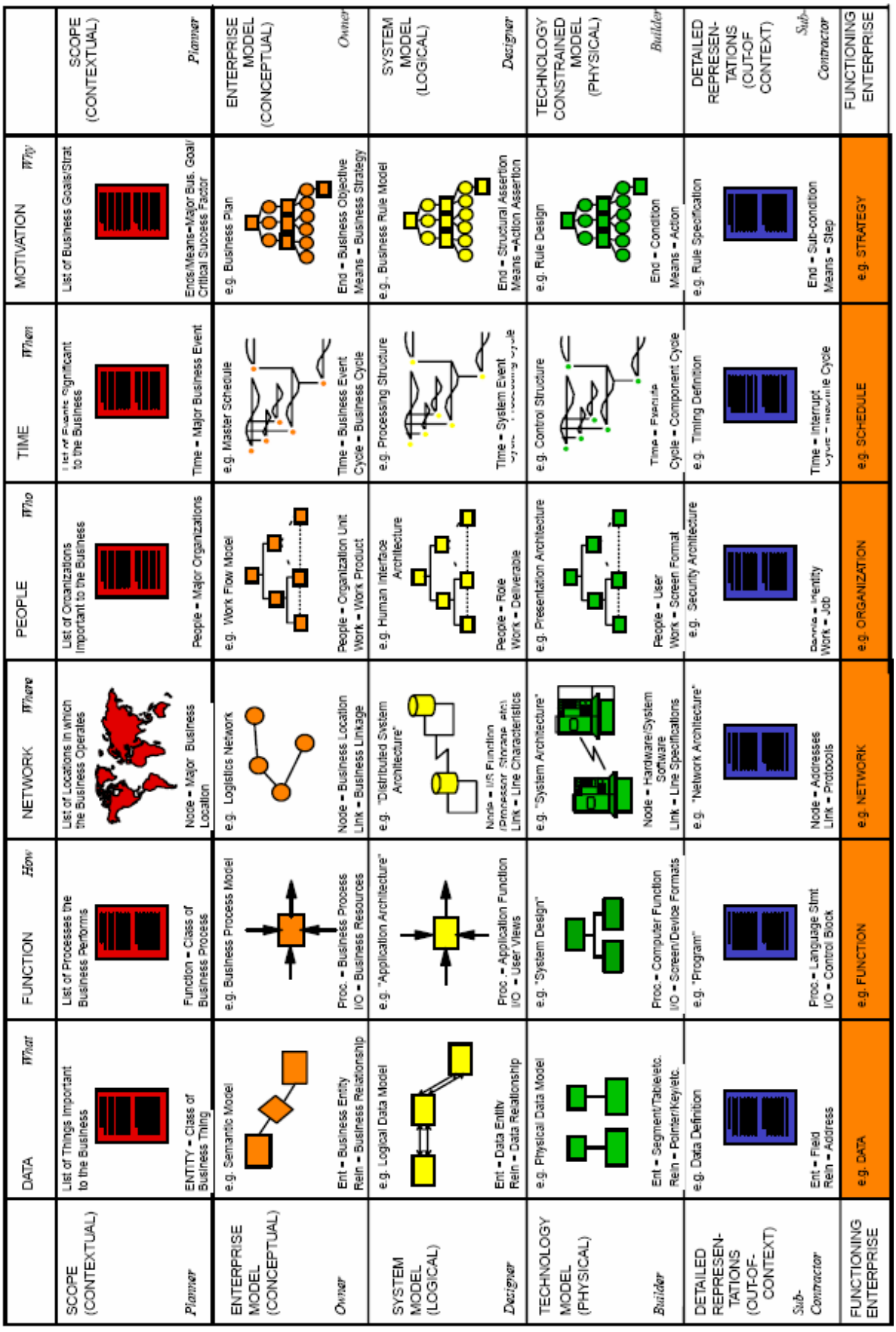
Prema [9], informacioni sistem čine osnovne komponente:

1. TEHNIČKA OPREMA (Hardware) – računari, štampači, skeneri, uređaji za rezervno napajanje, uređaji za rezervne kopije podataka, mrežna oprema
2. PROGRAMSKA PODRŠKA (Software) – operativni sistemi, aplikativni softver kancelarijskog poslovanja, antivirusni programi, uslužni programi, poslovno-aplikativni softver (program, baza podataka)
3. ORGANIZACIONA PODRŠKA (Orgware) – zakoni i pravilnici u oblasti poslovanja, zakoni i pravilnici u oblasti informacionih sistema, organizacija rada, pravila rada, principi rada. „Organizacionu podršku čine organizacione metode i postupci za usklađivanje svih delova informacionog sistema u organizacionu celinu" [9].
4. KADROVSKA PODRŠKA (Lifeware) – zaposleni u poslovnom delu organizacije, kao i zaposleni u delu razvoja i održavanja informacionog sistema sa svojim znanjima, iskustvom, zahtevima, navikama, interesima, potrebama itd. „Kadrovsku podršku čine kadrovi koji neposredno rade na poslovima koji omogućavaju da informacioni sistem ostvari svoju funkciju, ali i krajnji korisnici informacionog sistema." [9]

Prema [3], arhitekturu informacionog sistema čine podsistemi, odnosno arhitekture:

1. INFORMACIONA ARHITEKTURA - arhitektura podataka
2. APLIKACIONA ARHITEKTURA – funkcionalna arhitektura
3. TEHNOLOŠKA ARHITEKTURA – infrastrukturna i platformska arhitektura.

Zachman-ov frejmwork [10] je prvi odvojio koncept SWA i ISA, odnosno pokazao da SWA nije jedini koji čini ISA. Zachmann je u okviru Instituta za unapređenje frejmworka dao detaljni prikaz arhitekture preduzeća (organizacije) i odnos prema arhitekturi informacionog sistema (Slika 3).



Slika 3. Zachmannov framework za arhitekturu informacionog sistema [10]

Literatura

[1] ANSI/IEEE Std 1471-2000, *Recomended Practice for architectural description of Sofware-Intensive systems,* 2000.
[2] Lockemann P.: Information system Architectures: From Art to Science, Conference BTW2003 (Business, Technologie and Web) Proceeedings 2003., http://doeseno.informatik.uni-leipzig.de/proceedings/paper/keynote_lockeman.pdf
[3] Vasconcelos A, Soursa P, Tribolet J: *Information system Architecture Metrics: an Enterprise Engineering Evaluation Approach*, The Electronic Journal of Information Systems Evaluation, Vol 10, Issue 1, pp 91-122, ISSN 1566-6379, www.ejise.com
[4] Carnegie: *How do You Define Software Architecture?*, Software Engineering Institute, Carnegie Mellon University, December 2000, http://www.sei.cmu.edu/architecture/definitions.html
[5] Tissot, Florence, and Wes Crump, *An Integrated Enterprise Modeling Environment*, P. Bernus, K. Mertins, G. Schmidt (Eds.), Handbook on Architectures of Information Systems, Springer, pp.59-79, ISBN 3-540-64453-9, 1998.
[6] Vernadat, François, *Enterprise Modeling and Integration*, London, Chapman and Hall, 1996.
[7] Eriksson, Hans-Erik, and Magnus Penker: *Business Modeling with UML: Business Patterns at Work*, John Wiley and Sons, ISBN 0-471-29551-5, 2000.
[8] Marshall, Chris: *Enterprise Modeling with UML: Designing Successful Software through Business Analysis*, Addison Wesley Longman, ISBN 0 201-43313-3, 2000
[9] Stankić R: Poslovna informatika, Ekonomski fakultet Univerziteta u Beogradu, 2005.
[10] Zachman, John, *A Framework for Information System Architecture*, IBM system journal Vol.26 Nº 3, 1987, p.276 – 292.

1.4. Vrste informacionih sistema

Jauković [1] daje klasifikaciju informacionih sistema:

1. Prema vrsti pruženih usluga: sistemi za računarske usluge opšte namene, sistemi za čuvanje i pretraživanje podataka, sistemi za komutaciju poruka, sistemi za upravljanje fizičkim procesima, sistemi za kontrolu i upozorenja, sistemi za obradu transakcija,
2. Prema oblasti primene - u okviru poslovnih sistema razvijaju se tzv. poslovni informacioni sistemi («Business information systems»).
3. Prema stepenu automatizacije – istorijski razvoj generacija informacionih sistema:
   - NEAUTOMATIZOVANI INFORMACIONI SISTEMI – koriste mehanografska sredstva obrade, nosioci podataka su dokumenti, obrada nije jedinstvena i ponekad se ne obavlja uvek na isti način, podaci nisu formatizovani i strogo strukturirani, zadaci koji se rešavaju često nisu do kraja definisani, značajno su zastupljeni usmeni informacioni tokovi, koji su promenljivi i nepostojani, mnoge odluke se donose na osnovu usmenih informacija i iskustva ili na osnovu nepotpunih i često zastarelih podataka, subjektivni činioci mogu znatno da utiču na tok obrade podataka i dobijanja rezultata, dolazi do dupliranja rada na obradi podataka, naročiro kod izrade izveštaja, obrada je spora, prave se greške itd.
   - SISTEMI AUTOMATSKE OBRADE PODATAKA – korišćenje računara za prikupljanje i registrovanje podataka, klasificiranje, uređivanje i ažuriranje podataka, razna računanja i sumiranja, štampanje izveštaja. Odnose se na poslove koji su u potpunosti poznati i dobro opisani, čija je obrada regulisana zakonima i propisima organizacije. Sistemi AOP sastoje se od posebnih aplikacija koje obuhvataju pojedine funkcije ili organizacione celine (ili njihove delove). Prednosti: sistematizacija prikupljanja i čuvanja podataka, uvodi se opšta nomenklatura i jedinstven sistem obeležavanja, ubrzavanje prikupljanje i obrada podataka, povećava kvalitet informacija, sistematizuju izveštaji i način izveštavanja na pojedinim organizacionim i upravljačkim nivoima. Nedostaci: zanemaren sistemski pristup, svodi se na projektovanje programa i organizaciju podataka (cilj nije obrada podataka, već povećanje efikasnosti odlučivanja na osnovu dobrih i blagovremenih informacija), zasebne aplikacije – redundansa podataka.

- UPRAVLJAČKI INFORMACIONI SISTEMI («Management information system») – težište na informacijama i njihovom korišćenju za donošenje odluka, daju informacije brzo i u obliku koji je prilagođen potrebama onih koji odlučuju, daje izveštaje za donošenje odluka kod struktuiranih problema odlučivanja (srednji nivo odlučivanja u organizaciji – problemi za koje se može unapred tačno utvrditi konačno rešenje, koji su podaci potrebni da se ono dobije, kao i algoritam njegovog rešavanja), predstavljaju međusobno povezane podsisteme koji čine jedinstvenu celinu (podsistemi se odnose na vitalne funkcije realnog sistema, predstavljaju logički i tehnološki zaokružene celine koje se kod implementacije mogu nezavisno tretirati).
- IZVRŠNI INFORMACIONI SISTEMI – daju analitičke informacije o tekućem stanju organizacije i projekciji u budućnosti, a namenjene su najvišem nivou upravljanja.
- SISTEMI ZA PODRŠKU ODLUČIVANJU – daju podršku odlučivanju kod nestrukturiranih i slabo strukturiranih problema, zavise od eksternih podataka (koji mogu biti i nepouzdani), rešavanje problema se zasniva na znanju, pružaju podršku svim nivoima odlučivanja, ali podržavaju i vertikalne informacione tokove i integraciju informacija koje se koriste na različitim organizacionim i upravljačkim nivoima (posebno značajni za više nivoe), olakšavaju sintezu informacija iz pojedinih podsistema za strateško odlučivanje, mogućnost uočavanjauzajamnih relacija između različitih funkcija u raznim organizacionim celinama i razumevanje njegovog uticaja na organizaciju. Sastoji se od baze podataka, modela odlučivanja i specijalnog softvera koji ih integriše i omogućava korisnicima korišćenje ovih modela.
- EKSPERTNI SISTEMI – oponašaju rad eksperata, razvijeni su primenom tehnika veštaćke inteligencije. Opisuju znanja i pravila odlučivanja eksperata. Problemi: kako pribaviti znanje eksperata, kako to znanje predstaviti na računaru, kao na osnovu pribavljenog znanja izdvojiti rešenja konkretnih praktičnih problema. Sastoje se od: baza znanja, mehanizam zaključivanja, specijalizovana baza podataka (baza činjenica), mehanizam objašnjenja i korisnički interfejs.

Prema [2], istorijski razvoj informacionih sistema može biti podeljen na četiri perioda, na osnovu vrste informacionih sistema u tom periodu. To su:

- informacioni sistemi za obradu podataka (DPS – data processing systems)
- upravljački informacioni sistemi (MIS – management information systems)
- informacioni sistemi za podršku odlučivanju (DSS- decision support systems)
- ekspertni sistemi (ES – expert systems).

«Davis i Olson upravljački informacioni sistem (MIS – management information system) uopšteno definišu kao integrisani sistem obezbeđivanja informacija za podržavanje operacija upravljanja i funkcija odlučivanja u organizaciji, koji se koristi hardverom i softverom računara, manuelnim procedurama, modelima za planiranje, kontrolisanje i odlučivanje, kao i bazom podataka.» [3]

«Prema nivou odlučivanja kojem služe, informacioni sistemi mogu se podeliti na:

1. Informacione sisteme za operativne odluke
2. Informacione sisteme za taktičke odluke
3. Informacione sisteme za strateške odluke.» [2]

Prema [2], sa aspekta implementacije, informacioni sistemi mogu biti podeljeni prema:

1. povezanosti – neintegrisane informacione sisteme (sastoje se iz skupa više nepovezanih informacionih sistema), integrisane informacione sisteme (čine ga više međusobno povezanih podsistema).
2. centralizaciji:
   - centralizovani informacioni sistemi – «prednost centralizovano izgrađenih informacionih sistema ogleda se u jedinstvenoj bazi podataka koja u potpunosti eliminiše dupliranje podataka i omogućaa efikasniju obradu podataka u realnom

vremenu. Međutim, izgradnja, implementacija i održavanje centralizovanog informacionog sistema veoma je složen i dug proces. Takođe, sve izmene su složene zbog uzajamne zavisnosti svih komponenti informacionog sistema.
- delimično distribuirani informacioni sistemi – «Izgrađuje se sa distribuiranim procesima obrade podataka i distribuiranim formiranjem informacija za operativne odluke po pojedinim delovima procesa rada, zajedničkom bazom srodnih podataka, podela na podsisteme se vrši po homogenim funkcionalnim celinama».
- distribuirani informacioni sistemi – potpuno distribuirani informacioni sistem je organizovan kao mreža računara u svim informacionim podsistemima koji se potpuno i neposredno povezuju. Prihvatljiviji su sa ekonomskog aspekta, omogućavaju postepeni razvoj, održavanje je jednostavnije, veća je prilagodljivost na promene.

Literatura

[1] Jauković M: *Uvod u informacione sisteme*, Tehnička knjiga, Beograd, 1992.
[2] Stankić R: *Poslovna informatika*, Ekonomski fakultet Univerziteta u Beogradu, 2005.
[3] Balaban N, Ristić Ž, Đurković J: *Principi informatike*, Savremena administracija, 1996.

1.5. Proces razvoja informacionog sistema

Prema [1], informacija predstavlja strateški resurs organizacije. Upravljanje uticajem informacionih tehnologija na kvalitet ostvarivanja poslovnih ciljeva organizacionog sistema, čime se postiže optimizacija troškova i povećanje efikasnosti poslovnih informacionih sistema, podržano je različitim metodologijama, među kojima je i savremena CoBIT metodologija [2].

S obzirom na značaj informacionog sistema, prema Stankiću [3], razvoj informacionog sistema treba da se odvija u okviru dugoročnog plana razvoja organizacionog sistema kojem pripada. U okviru tog dugoročnog plana razvoja treba da postoji dugoročni plan razvoja informacionog sistema, koji predstavlja okvir unutar kog se može planirati, projektovati i realizovati svaki poseban projekat unapređenja nekog podsistema informacionog sistema. Prema Stankiću, osnovni plan razvoja informacionog sistema treba da sadrži sledeće komponente, koje dalji projekti treba da podrže konkretnim tehnološkim rešenjima:
- model poslovnih procesa
- model poslovnih podataka
- opis radnih procedura
- tok informacija
- struktura organizacije (organizaciona struktura).

Proces razvoja informacionog sistema je kontinuiran, ali se najčešće odvija u okviru definisanih projekata unapređenja. U odgovarajućim projektima jasno su definisani ciljevi i obuhvat aktivnosti i očekivani rezultati (koji se mogu odnositi na pojedine komponente ili ceo informacioni sistem). U zavisnosti od obuhvata posla, razvoj novog informacionog sistema može biti dugotrajan proces, često praćen rizicima neuspeha projekta. Zato je potrebno poznavati i primenjivati neku od metodologija razvoja informacionog sistema, ali u okviru primene metodologije upravljanja projektima.

Literatura

[1] Balaban N, Ristić Ž, Đurković J: *Principi informatike*, Savremena administracija, 1996.
[2] *Cobit 5 metodologija*, http://www.isaca.org/cobit/pages/default.aspx
[3] Stankić R: *Poslovna informatika*, Ekonomski fakultet Univerziteta u Beogradu, 2005.

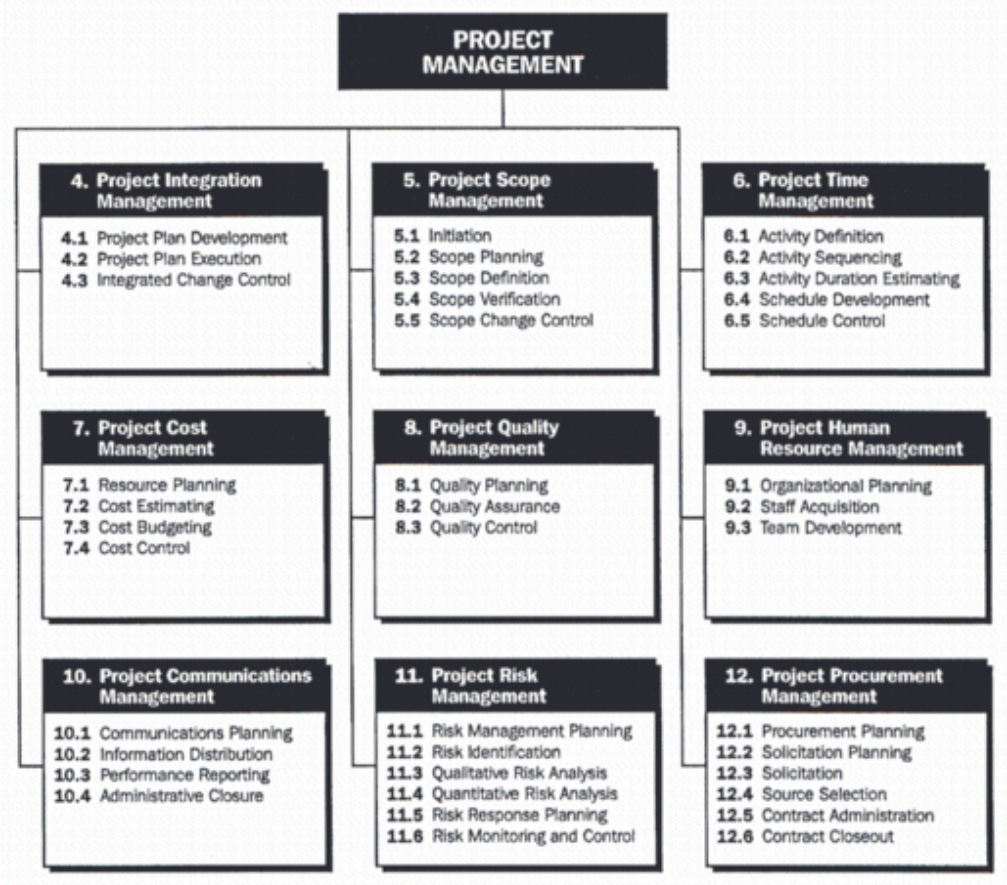
1.5.1. Elementi upravljanja projektima

«Upravljanje projektom se posmatra kao usmeravanje i koordinacija ljudskih i materijalnih resursa da bi se projekat realizovao u planiranom vremenu, sa planiranim kvalitetom i planiranim troškovima.» *[1]* PMBOK [2] definiše životni ciklus projekta kroz sledeće faze [3]:

- Inicijalizacija projekta
- Planiranje projekta
- Izvršavanje projekta
- Monitoring i kontrola projekta
- Zatvaranje projekta.

«Na osnovu analize teorije i prakse project managementa, mogu se izdvojiti i definisati sledeće funkcionalne oblasti project management akoje danas čine njegovu teorijsku bazu i realne mogućnosti za praktičnu primenu: Upravljanje obimom projekta, Upravljanje vremenom, Upravljanje troškovima, Upravljanje kvalitetom, Upravljanje ugovaranjem, Upravljanje nabavkom, Upravljanje ljudskim resursima, Upravljanje komunikacijama, Upravljanje konfliktima, Upravljanje promenama u projektu, Upravljanje rizikom. « [1]

Sistematskim procesima u devet osnovnih funkcionalnih oblasti project managementa, odnosno oblasti znanja vrši se proces upravljanja projektom: Upravljanje integrisanjem projekta, Upravljanje obimom projekta, Upravljanje troškovima, Upravljanje vremenom,Upravljanje kvalitetom, Upravljanje ljudskim resursima, Upravljanje komunikacijama, Upravljanje ugovaranjem,Upravljanje rizikom. [2]



Slika 4. Pregled oblasti znanja i procesa projektnog menadžmenta (prema [2])

„Nakon upoznavanja sa osnovnim elementima upravljanja projektom pomoću PMBOK koja pruža širok opis znanja (pojmovi, tehnike, postupci) u ovoj oblasti, preporučuje se korišćenje

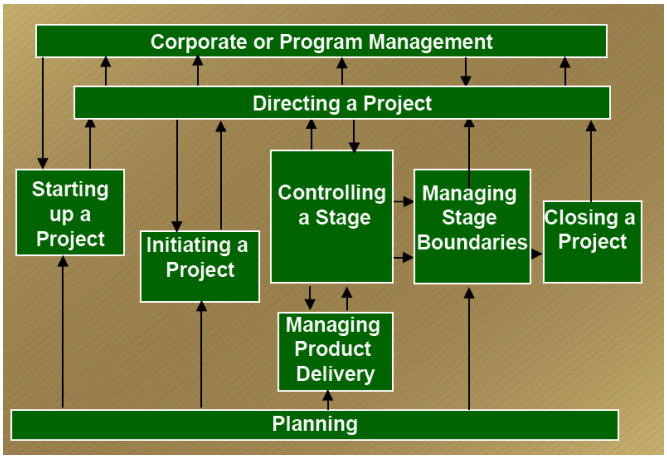
PRINCE2 metodologije koja obezbeđuje detaljnija uputstva o načinu izvršavanja konkretnih aktivnosti u okviru životnog ciklusa projekta.» [4] Prema komparacijama [5] metodologija PMBOK i PRINCE2, date su sledece karakteristike:

- „PMBOK – sveobuhvatna, široko deskriptivna, a preskriptivna (daje konkretna uputstva) samo na visokom nivou posmatranja, orjentisan ja korisničke zahteve, nosilac vlasništva nad projektom su sponzori i zainteresovane strane (stakeholders), USA/međunarodni standard
- PRINCE2 – fokusira se na ključne oblasti rizika, ne tvrdi da je kompletna, visok nivo preskriptivnosti (daje mnogo konkretnih uputstava), posebno u vezi strukture procesa, adaptabilna na bilo koju veličinu projekta, orjentisan na poslovni problem, jasno definisano vlasništvo nad projektom i usmerenje od strane starijeg (senior) menadžmenta, UK standard.” [4]

PRINCE2 (Projects in Controllled Environments, Verzija 2), razvijena je inicijalno 1975. godine, a od 1989. godine razvija se od strane vladine organizacije Velike Britanije (OGC – office of Government Commerce). Počev od 1996. godine nosi oznaku verzije 2 i dostupno je javnom sektoru. Poslednja verzija PRINCE2 metodologije je iz 2009. godine. PRINCE2 se sastoji od osam procesa, osam komponenti i tri tehnike. Osnovne komponente su [6]: Poslovni problem (business case), Organizacija, Planovi,Kontrola, Upravljanje rizikom, Kvalitet u projektnom okruženju, Upravljanje konfiguracijom,Kontrola promena. Osnovni procesi PRINCE2 su [1]:

Tabela 1. Spisak procesa u okviru PRINCE2 metodologije

	USMERAVANJE PROJEKTA		
STARTOVANJE PROJEKTA	INICIRANJE PROJEKTA	UPRAVLJANJE GRANICAMA FAZE	ZATVARANJE PROJEKTA
	PLANIRANJE	KONTROLISANJE FAZE	
		UPRAVLJANJE ISPORUKOM REZULTATA	



Slika 5. Model procesa upravljanja projektima prema PRINCE 2 metodologiji ([5])

Tri tehnike koje se koriste u PRINCE2 [3]:

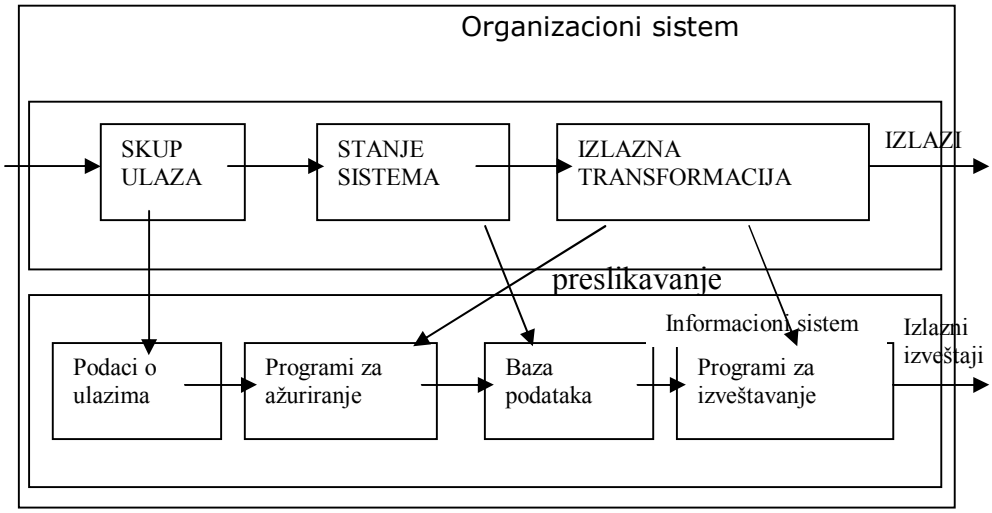
1. Proizvod bazirano planiranje (opis konačnog proizvoda, Struktura proizvoda (PBS – Product Breakdown structure), opis pojedinačnih proizvoda, Dijagram toka proizvoda)
2. Kontrola promena – praćenje realizacije rešavanja problema kroz sve faze
3. Kontrola kvaliteta – procedura za utvrđivanje da li je proizvod u skladu sa zahtevima.

Literatura

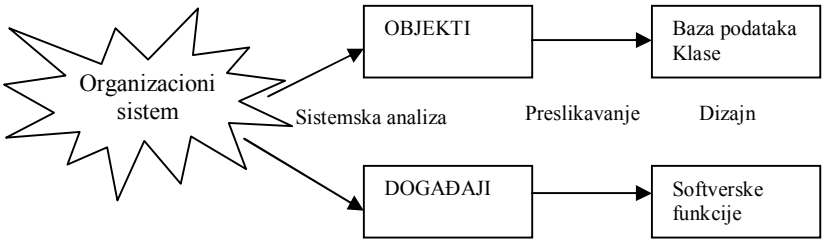
[1] Jovanović P: *Upravljanje projektom*, Fakultet organizacionih nauka, Beograd, 2006.
[2] Project Management Institute*:Project Management Body of Knowledge*, Upper Darby, 1987.
[3] Yeong A: *The marriage proposal of PRINCE2 and PMBOK*, 2007.
[4] Kazi Lj: *Razvoj višeslojnog web baziranog informacionog sistema za podršku upravljanju projektima*, Master rad, Tehnički fakultet «Mihajlo Pupin» Zrenjanin, 2009.
[5] Siegelaub J. M: *How PRINCE2 can complement PMBOK and your PMP*, PMI/Westchester chapter, 2004.
[6] CCTA: *Managing Successful Projects with PRINCE2*, Key Skills Limited, 1999.

1.5.2. Faze i aktivnosti procesa razvoja informacionog sistema

Prema [1], informacioni sistem se razvija kao model kojim se preslikavaju objekti i događaji organizacionog sistema (Slika 6) u elemente aplikativnog softvera (Slika 7), koji se izvršava na odgovarajućoj računarsko-komunikacionoj opremi.



Slika 6. Preslikavanje elemenata org. sistema u inf. sistem (prema [3])



Slika 7. Preslikavanje objekata i događaja u elemente aplikativnog softvera

Prema [2], s obzirom da je razvoj informacionog sistema blisko povezan sa organizacionim sistemom, organizacione strukture i planovi treba da sadrže elemente modela kojima se jasno izdvajaju poslovni procesi i podaci. Razvijene su različiti metodološki pristupi modelovanju organizacionih sistema u svrhu dekompozicije i pripreme za preslikavanje u odgovarajuće elemente informacionih sistema, kao što je BSP („Business System Planing") metoda firme IBM, gde se pristupom s vrha ka dnu organizacioni sistem deli na funkcionalne celine (grupisanjem sličnih poslovnih procesa) - podsisteme, a informacije u formi matrice dodeljuju odgovarajućim podsistemima.

Prema Stanojeviću [3], razlikuju se tri najzastupljenije koncepcije pristupa razvoju informacionog sistema:

- Koncepcija PARCIJALNOG PRISTUPA po kojoj se novim informacionim sistemima u pojedinim etapama obuhvataju određeni poslovi, ili grupe poslova, bez prethodne identifikacije celokupnog organizacionog sistema i bez razrade integralnog modela novoj informacionog sistema,
- Koncepcija KOMPLEKSNOG SISTEMSKOG PRISTUPA, koja obuhvata identifikaciju celokupnog organizacionog sistema, razradu integralnog modela novoj informacionog sistema i realizacije novog informacionog sistema kao integralne celine,
- Koncepcija SISTEMSKO-MODULARNOG PRISTUPA po kojima se novim informacionim sistemom, u pojedinim etapama razvoja, obuhvataju modeli, tj. informacioni podsistemi. Sukcesivnim projektovanjem i razvojem informacionih podsistema formira se i gradi novi informacioni sistem.

Kao praktično najprihvatljivija, Stanojević izdvaja koncepciju sistemsko-modularnog pristupa. Najvažnije faze u ovom pristupu su:

- Snimanje, analiza i identifikacija struktura organizacionog sistema za čije potrebe se razvija novi informacioni sistem
- Identifikuju se grupe procesa koji su po svojoj funkcionalnoj nameni neposredno povezani i predstavljaju funkcionalno-logičke celine (najčešće odgovaraju, ali i ne moraju odgovarati organizacionim podsistemima)
- Formiraju se informacioni podsistemi u okviru svakog organizacionog podsistema i identifikuju se svi skupovi informacija koji se u njima javljaju
- vrši se kvantitativna analiza strukture organizacionih sistema i određuju mere uticaja koje procesi vrše na funkcionisanje organizacionih sistema
- određuje se redosled razvoja informacionih podsistema i to tako da priorite imaju podsistemi koji pripadaju organizacionim podsistemima većeg uticaja
- formiraju se strukture skupova informacija informacionih podsistema i vrši provera opravdanosti proizvodnje pojedinih informacija.

Prema Stanojeviću, u skladu sa usvojenom sistemsko-modularnom koncepcijom, aktivnosti u projektovanju, razvoju i funkcionisanju informacionog sistema se mogu podeliti u deset faza: Izrada studije opravdanosti razvoja novog informacionog sistema, Izrada programa razvoja i definisanja projektnih zadataka; Snimanje informacionih podsistema, modeliranje i analiza; Izrada idejnog projekta informacionih podsistema obuhvaćenih projektnim zadatkom; Izbor sistema obrade; Izrada detaljnog projekta informacionih podsistema; Programiranje; Pripreme za uvođenje projektovanih sistema; Instaliranje sistema, testiranje i probni rad; Funkcionisanje projektovanih informacionih podsistema.

Prema Stankiću [4], proces razvoja informacionog sistema obuhvata 4 osnovne faze: projektovanje, implementacija, uvođenje i funkcionisanje. Razlikujemo linearni i evolutivni pristup. Kod složenih sistema sistem se deli na podsisteme koji uklapaju u celinu. Linearnim pristupom faze razvoja slede jedna za drugom tek nakon sto se prethodna faza u potpunosti zavrsi, gde kod slozenih sistema faza projektovanja moze biti izuzetno dugotrajna. Ipak kod slozenih sistema se i dalje najcesce koristi linearni pristup. Evolutivni pristup zasniva se na realizaciji prototipa podrske podsistemima i unapredjenju u ciklusima.
Kod linearnog pristupa, osnovna je podela na faze: izrada koncepta (prevodjenje ideje u koncept resenja), definisanje projekta (detaljniji opis nacina izvodjenja), izvodjenje projekta (implementacija, testiranje i odrzavanje).
Prema Stankicu, projekat informacionog sistema se najcesce deli na niz projekata pojedinih informacionih podsistema ili projektnih segmenata. Projektovanje obuhvata analizu i oblikovanje sistema.
Analiza sistema obuhvata utvrdjivanje i analizu cinjenicnog stanja, dok oblikovanje podrazumeva defiinisanje programskih obrazaca i izvestaja, baze podataka, izbor opreme, pisanje organizacionih uputstava u vezi primene sistema, specifikaciju programa. Implementacija sistema podrazumeva – osposobljavanje osoblja i testiranje sistema, kao i proveru funkcionisanja sistema.
Prema Stankicu proces razvoja novog informacionog sistema odvija se kroz sledeće glavne faze:

1. «Istraživanje spremnosti poslovnog sistema da prihvati savremeni informacioni sistem podržan savremenom informacionom tehnologijom» [4], odnosno STUDIJA IZVODLJIVOSTI [3]. Odnosi se na analizu opšte, organizacione, kadrovske i finansijske spremnosti organizacionog sistema kao preduslova za izgradnju i funkcionisanje informacionog sistema. Ukoliko su ovi preduslovi ispunjeni, pristupa se projektovanju informacionog sistema.
2. «Izrada idejnog projekta» koja ima za cilj omogućavanje odluka o izgradnji integralnog informacionog sistema, pre svega o troškovima i organizaciji novog informacionog sistema. Osnovni elementi IDEJNOG PROJEKTA su: opis opšteg pristupa izgradnji integralnog informacionog sistema, opis organizacije konkretnog poslovnog sistema (sadrži organizacionu shemu, kratak opis delatnosti svih organizacionih delova, analizu sistema donošenja najznačajnijih odluka, analizu sistema komunikacija, opis veza sistema sa okruženjem), analiza postojećeg informacionog sistema, koncepcija novog informacionog sistema, definisanje podsistema novog informacionog sistema, određivanje nivoa informisanosti, upravljanje izgradnjom i funkcionisanjem informacionog sistema, tehničke i organizacione pretpostavke za izgradnju informacionog sistema, položaj poslovnog informacionog sistema u odnosu na informacioni sistem nekog višeg nivoa, materijalni uslovi za realizaciju izgradnje informacionog sistema, kadrovska osnovna, ekonomska opravdanost izgradnje informacionog sistema s definisanjem prioriteta u izgradnji pojedinih podsistema, dinamika projektovanja i izgradnje informacionih podsistema.
3. «Izrada glavnog izvršnog projekta» ima za cilj da omogući realizaciju, odnosno uvođenje novog informacionog sistema kao sastavnog dela poslovnog sistema. Aktivnosti koje obuhvata izrada izvršnog projekta odnose se na: detaljno definisanje informacionih podsistema, detljno snimanje postojećeg stanja, izrada nove organizacije i kreiranje organizacionih uputstava, kreiranje strukture datoteka-baze podataka, kreiranje strukture ulaznih obrazaca i izlazne dokumentacije, izrada programskih zadataka.

U cilju uvođenja standarda u ovoj oblasti, međunarodne organizacije kao što su ISO (International Standards Organization) i druge stručne organizacije (npr. IEEE) definišu standardne procedure i dokumentaciju u razvoju informacionih sistema i softvera. Na nivou Republike Srbije je za potrebe standardizacije u oblasti razvoja informacionih sistema organa uprave definisano uputstvo [5] kojim se proces razvoja informacionog sistema odvija tako što se «aktivnosti na razvoju informacionog sistema organizuju u okviru projektnih celina: preliminarna studija o razvoju informacionog sistema, izrada idejnog, glavnog i izvođačkog projekta, uvođenje projektovanih rešenja. Pri projektovanju vrlo složenih informacionih sistema, za ceo sistem se priprema samo preliminarna studija o njegovom razvoju, a ostale aktivnosti na projektovanju IS sprovode se za svaki sistem posebno» [5].

1. ***Preliminarna studija o razvoju informacionog sistema -*** služi za odlučivanje o razvoju i planiranje razvoja informacionog sistema sadrži: projektni zahtev za izradu informacionog sistema sa precizno definisanim ciljevima, zadacima i obuhvatom informacionog sistema; snimak i analizu postojećeg stanja; spisak korisnika informacionog sistema; dekompoziciju informacionog sistema na podsisteme; prikaz rešenja osnovnih funkcija informacionog sistema (podsistema) na nivou idejne skice sa mogućim alternativama; rezultate istraživanja mogućih efekata informacionog sistema; specifikaciju zahteva informacionog sistema (finansijski, kadrovski, tehnički, organizacioni i dr.); plan izrade projekta informacionog sistema i uvođenja projektovanih rešenja.

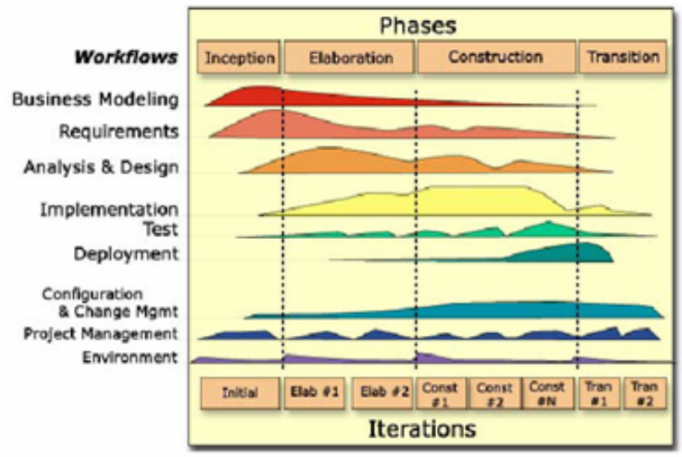
2. ***Idejni projekat informacionog sistema -***vrši se logičko modeliranje sistema u skladu sa zahtevima iz preliminarne studije o razvoju informacionog sistema. Idejni projekat sadrži: dijagram toka podataka celokupnog sistema (skladišta podataka i tokova podataka, procese i funkcije koji transformišu te podatke); logički model podataka (objekti - veze) tj. logičku organizaciju podataka; rečnik podataka (simbolički naziv, opis, tip podataka, skup mogućih vrednosti podataka); procenu resursa i drugih preduslova za razvoj i rad informacionog sistema (tip računarskog sistema i druge opreme i softvera, broj i profil kadrova, prostorije, organizacione promene, propisi i dr.); procenu efekata informacionog sistema (ekonomski, organizacioni, kvalitativni i dr.); operativni plan realizacije informacionog sistema kojim se

razrađuju aktivnosti na izradi glavnog projekta informacionog sistema i izvođačkog projekta i uvođenja projektovanih rešenja.

3. ***Glavni projekat informacionog sistema -*** vrši se konverzija logičkog modela u fizički tj. razrada i prilagođavanje za konkretnu realizaciju svih elemenata idejnog projekta. Glavni projekat sadrži: podelu logičkog modela procesa i podataka na jedinične postupke tj. delove koji se izvršavaju kao celina (automatizovano ili manuelno); razradu detalja svakog jedinačnog postupka da bi se dobili svi potrebni elementi za uvođenje u sistem: deo dijagrama toka podataka koji se koristi, skupovi podataka kojima se pristupa, izgledi ekrana i izlaznih pregleda, logika postupka iskazana strukturnim jezikom. Za jedinične postupke se daje: lista postupaka, tekstualni opis, ulazni podaci i logika postupka opisana na odgovarajući način; fizički model podataka; programsku dokumentaciju: izvorne verzije programa, izgledi ekrana i izlaznih izveštaja; rečnik podataka;radna uputstva (za korisnike i za održavanje baze podataka); postupak prelaska na novi sistem: opis postupka prikupljanja podataka radi kreiranja osnove na novi sistem, programska dokumentacija za preliminarnu obradu podataka i kreiranje baze podataka, plan i način prelaska na novi sistem; detaljnu specifikaciju potrebne računarske i druge informacione opreme i softvera.

4. ***Izvođački projekat i uvođenje projektovanih rešenja-*** U okviru izvođačkog projekta i uvođenja projektovanih rešenja vrši se testiranje, uvođenje i uhodavanje sistema do njegovog puštanja u operativni rad. U okviru ove projektne celine vrši se: utvrđivanje svih aktivnosti i operativnog plana realizacije informacionog sistema; određivanje test podataka sa ciljem da se obuhvate svi slučajevi koji mogu da se pojave u praksi; testiranje programa, delova sistema i celokupnog sistema (za redovne situacije i transakcije u predviđenim terminima i promenljive vrednosti određenih parametara, u slučaju nastavka obrade prilikom nenormalnog prekida rada i regeneracije sistema usled pada sistema, proveru softverske zaštite podataka i funkcionisanja sistema), kreiranje datoteka, odnosno baza podataka (prikupljanje stvarnih podataka i njihova kontrola, preliminarna obrada podataka, kreiranje datoteka odnosno baza podataka, provera programa za ažuriranje sadržaja datoteka odnosno baza podataka); probna obrada sistema sa stvarnim podacima (testiranje kompletnog programskog sistema, provera programske dokumentacije, provera izrađenih uputstava, ispravka uputstava, ispravka uočenih nepravilnosti i odgovarajuća korekcija dokumentacije, analiza probne obrade); finalizacija projektne dokumentacije (usklađivanje dokumentacije sa isprobanim i testiranim rešenjima); obuka kadrova; prelazak na novi sistem (realizacija projektovanog načina uvođenja sistema, vođenje evidencije o svim fazama uvođenja sistema, praćenje kadrova koji će učestvovati u radu sistema, analiza uvođenja informacionog sistema); evaluacija novog sistema (osim analize relevantnih faktora probne obrade i uvođenja sistema vrši se i analiza vremena reagovanja sistema, broja i obima izmena, vremena rada sistema sa posebnim osvrtom na efikasnost sistema, uslove i troškove za poboljšanje sistema); primopredaja projekta; praćenje rada informacionog sistema (pružanje pomoći od strane projektanta pri izvođenju složenih obrada, otklanjanje nejasnoća, nepreciznosti i eventualnih grešaka u programima i dokumentaciji, dovođenje sistema u ispravno stanje).

Jezgro razvoja i uvođenja novog informacionog sistema predstavlja razvoj aplikativnog softvera. Osnovni proces razvoja softvera [6] predstavljen je danas već standardnim objedinjenim procesom razvoja softvera, koji je uvela firma IBM i firma Rational («Rational Unified Process») predstavljen na slici 8.



Slika 8. Faze osnovnog objedinjenog procesa razvoja softvera [6]

Literatura

[1] Lazarević B,Jovanović V, Vučković M: *Projektovanje informacionih sistema I deo*, Naučna knjiga, Beograd, 1988.
[2] Balaban N, Ristić Ž, Đurković J: *Principi informatike*, Savremena administracija, 1996.
[3] Stanojević M: *Osnovi projektovanja informacionih sistema*, Naučna knjiga, Beograd, 1986.
[4] Stankić R: *Poslovna informatika*, Ekonomski fakultet Univerziteta u Beogradu, 2005.
[5] Uputstvo za izradu i usvajanje projekata informacionih sistema organa uprave, Službeni glasnik SRS, broj 49/89
[6] Balduino R: *Basic Unified Process*, IBM, 2007.

1.6. Zakonska regulativa u oblasti informacionih sistema

U okviru organizacionog aspekta razvoja informacionog sistema, izuzetno važan aspekt predstavlja zakonska regulativa, koja mora biti uzeta u obzir u procesu razvoja informacionog sistema. Neki od najvažnih zakona u ovoj oblasti su.

1. Zakon o informacionom sistemu Republike Srbije
2. Zakon o potvrđivanju konvencije o zaštiti lica u odnosu na automatsku obradu ličnih podataka
3. Zakoni o zaštiti podataka o ličnosti
4. Zakon o elektronskom potpisu

Zakon o informacionom sistemu Republike Srbije ("Sl. glasnik RS", br. 12/96)
U nastavku je dat izvod iz navedenog zakona.
„Ovim zakonom uređuju se prava i obaveze državnih organa i organizacija, organa teritorijalne autonomije i organa lokalne samouprave kad obavljaju poverene poslove državne uprave, kao i preduzeća, ustanova i drugih organizacija kad obavljaju poverena upravna ovlašćenja. Evidenciju propisanu zakonom, organ i organizacija vodi u skladu sa tim zakonom i koristeći sredstva za elektronsku obradu podataka. Informacioni sistem čine informacioni podsistemi. Podaci čije je vođenje predviđeno zakonom iz određene oblasti čine sadržaj baze podataka te oblasti. Određeni podaci o građanima, pravnim i drugim licima koja obavljaju delatnost i podaci o prostornim jedinicama, kao i šifarnici i klasifikacije neophodni za njihovo korišćenje, čine sadržaj zajedničke baze podataka. Organ i organizacija povezuju se preko računsko-telekomunikacione mreže i u okviru informacionog sistema obezbeđuju evidentiranje podataka na mestu njihovog nastanka, tačnost, kvalitet, zaštitu podataka pri obradi i prenosu, dostupnost podataka pod jednakim uslovima ovlašćenim korisnicima, primenu jedinstvenih standarda i razmenu podataka i dokumenata. Dokument u informacionom sistemu sačinjen u elektronskom ili drugom obliku ima istu pravnu snagu. Organ i organizacija redovno održava podatke iz svoje nadležnosti, a naročito blagovremeno, potpuno i tačno unosi podatke u zajedničku bazu podataka. Organ i organizacija vodi rečnik podataka informacionog podsistema. Organ državne

uprave nadležan za informacioni sistem vodi rečnik podataka informacionog sistema i obezbeđuje jedinstveno projektno rešenje za vođenje rečnika podataka informacionog podsistema. Rečnik podataka, u smislu ovog zakona, je programsko rešenje kojim se opisuje sadržaj baza podataka, registra i evidencija i uslovi i mogućnosti za njihovo korišćenje. Organ i organizacija preduzima mere obezbeđivanja i zaštite informacionog podsistema u svim fazama razvoja i funkcionisanja, u redovnim i vanrednim prilikama. Mere iz stava 1 ovog člana odnose se na: podatke i sve nosioce podataka, programsku podršku, računarsku opremu, računarske mreže, prostorije u kojima su smeštene oprema i instalacije i učesnike u informacionom sistemu. Vlada Republike Srbije propisaće sadržaj i način sprovođenja mera obezbeđenja zaštite imovine i sistema i zaštite podataka od zloupotrebe. Vlada Republike Srbije donosi program razvoja informacionog sistema koji obuhvata naročito: opis zadataka, rokove i nosioce njihovog izvršenja, neophodne kadrovske, tehničke i druge uslove i potrebna finansijska sredstva."

Zakon o potvrđivanju konvencije o zaštiti lica u odnosu na automatsku obradu ličnih podataka ("Sl. list SRJ - Međunarodni ugovori", br. 1/92 i "Sl. list SCG - Međunarodni ugovori", br. 11/2005 - dr. zakon)

U okviru ovog zakona „Potvrđuje se Konvencija o zaštiti lica u odnosu na automatsku obradu ličnih podataka, usvojena 28. januara 1991. godine u Strazburu u originalu na engleskom i francuskom jeziku." U nastavku su dati izvodi iz navedenog zakona. «Cilj ove konvencije je da garantuje svakom fizičkom licu, bez obzira na njegovu nacionalnu pripadnost ili na mesto stanovanja, poštovanje njegovih osnovnih prava i sloboda, a naročito njegovog prava na privatnost kada je reč o automatskoj obradi njegovih ličnih podataka "Automatska obrada" označava sledeće operacije koje se delimično ili u potpunosti obavljaju automatski: unošenje podataka, obrada tih podataka pomoću logičkih i/ili aritmetičkih operacija, unošenje izmena, njihovo brisanje, pronalaženje ili difuzija. Konvencija ce se primenjivati i u slučaju datoteka sa ličnim podacima koji nisu automatski obrađeni Lični podaci koji se automatski obrađuju: (a) lojalno se i zakonito dobijaju i obrađuju, (b) uneti su za tačno utvrđene i legitimne svrhe i ne koriste se nenamenski, (c) adekvatni su, relevantni i odgovarajućeg obima u odnosu na svrhe za koje su i uneti, (d) brižljivo su složeni i, kada je potrebno, blagovremeni, (d) pohranjeni su u obliku koji omogućuje identifikaciju zainteresovanih lica u periodu koji ne premašuje onaj neophodan period za svrhe za koje su i pohranjeni. Svako lice mora biti u mogućnosti: (a) da bude obavešteno o postojanju automatizovane zbirke sa ličnim podacima, njenoj osnovnoj svrsi, kao i o identitetu i stalnoj adresi ili pak adresi sedišta firme u kojoj je zaposlen rukovalac zbirke, (b) da dobija u razumnim intervalima i bez preteranih rokova ili troškova potvrdu o postojanju ili ne, u automatizovanoj zbirci ličnih podataka koji se na njega odnose, kao i da mu se ti podaci dostave u razumljivom obliku«

Zakon o zaštiti podataka o ličnosti ("Sl. list SRJ", br. 24/98 i 26/98 - ispr. i "Sl. list SCG", br. 1/2003 - Ustavna povelja)

U nastavku je dat izvod iz navedenog zakona.

„Zaštitom ličnih podataka, u smislu ovog zakona, bez obzira na oblik u kome su iskazani, smatraju se postupci i mere kojima se sprečava nezakonito prikupljanje, obrada, čuvanje, korišćenje i razmena ličnih podataka, kao i njihovo iznošenje iz zemlje. Lični podaci mogu se prikupljati, obrađivati i koristiti samo za svrhe utvrđene zakonom, a u druge svrhe - na osnovu pismene saglasnosti građanina. lični podaci su informacije koje su sadržane u zbirkama tih podataka, a koji se odnose na privatnost, integritet ličnosti, lični i porodični život i druga lična prava koja su u vezi sa identifikovanim licem. obrada ličnih podataka je skup ručnih, poluautomatizovanih ili automatizovanih aktivnosti s podacima radi prikupljanja, memorisanja, pregrupisavanja, izmene, razmene, prenosa, pretraživanja, širenja, zabrane korišćenja i brisanja podataka. rukovalac zbirkom ličnih podataka je državni organ ili organizacija, organ jedinice lokalne samouprave, pravno lice i preduzetnik koji su zakonom ili pismenom saglasnošću građanina ovlašćeni da prikupljaju, obrađuju, čuvaju i prenose lične podatke i da uspostavljaju, održavaju i koriste zbirku ličnih podataka. Zbirka ličnih podataka je uređen skup ličnih podataka (evidencija, registar, datoteka i sl.), bez obzira na oblik iskazivanja i sredstva za njihovo čuvanje; katalog zbirki ličnih podataka je pregled zbirki ličnih podataka, sa opisom njihove sadržine i namene, kao i načina obezbeđivanja i dostupnosti; Rukovalac zbirkom ličnih podataka uspostavlja i vodi katalog zbirki ličnih podataka. Katalog je javan i dostupan je na uvid

svakom građaninu. Pismenu saglasnost o uspostavljanju zbirke ličnih podataka građanin daje rukovaocu zbirkom ličnih podataka radi obrade i čuvanja tih podataka.U ime lica koja su lišena poslovne sposobnosti pismenu saglasnost iz stava 1. ovog člana daje staratelj, a za maloletna lica tu saglasnost daju roditelji ili usvojioci, odnosno staratelji. Zbirka ličnih podataka, deo zbirke ili pojedini podaci iz nje mogu se koristiti za naučne, obrazovne ili slične svrhe u obliku koji ne omogućava identifikaciju građanina. Zbirka ličnih podataka, deo zbirke ili pojedini podaci, u smislu stava 2. ovog člana, mogu se dati u obliku koji omogućava identifikaciju građanina samo uz njegovu pismenu saglasnost. Građanin ima pravo da sazna: 1) u kojim zbirkama ličnih podataka se nalaze podaci koji se odnose na njega; 2) koji se podaci o njemu obrađuju, ko ih obrađuje, u koje svrhe i po kom osnovu; 3) ko su korisnici ličnih podataka koji se na njega odnose i po kom osnovu. Građanin može da zahteva od rukovaoca zbirkom ličnih podataka: 1) obaveštenje o postojanju zbirke ličnih podataka i pismeni dokaz (potvrdu) o ličnim podacima koji se u zbirci vode o njemu; 2) uvid u podatke koji se na njega odnose; 3) ispravku netačnih podataka koji se na njega odnose; 4) brisanje podataka koji se na njega odnose ako njihova obrada nije u skladu sa zakonom, odnosno ugovorom; 5) zabranu korišćenja netačnih, neažurnih i nepotpunih podataka koji se na njega odnose; 6) zabranu korišćenja podataka iz zbirki podataka i sl. ako se ne koriste u skladu sa zakonom, odnosno ugovorom.Lični podaci moraju biti tačni, ažurni i zasnovani na verodostojnim izvorima, a s obzirom na namenu za koju se prikupljaju - i potpuni. Lični podaci se prikupljaju na način kojim se ne vređa dostojanstvo čoveka. Lični podaci o građanima mogu se prikupljati od drugih građana ili preuzimati iz postojećih zbirki ličnih podataka u slučajevima predviđenim zakonom ili na osnovu pismene saglasnosti građanina. Rukovalac zbirkom ličnih podataka odgovoran je za kvalitet tih podataka. Od trenutka uzimanja ličnih podataka od građanina, rukovalac zbirkom ličnih podataka odgovoran je za tačnost, ažurnost i potpunost podataka, kao i za označavanje podataka čije je korišćenje zabranjeno. Lični podaci o rasnom poreklu, nacionalnoj pripadnosti, religioznim i drugim uverenjima, političkim i sindikalnim opredeljenjima i seksualnom životu mogu se prikupljati, obrađivati i davati na korišćenje samo uz pismenu saglasnost građanina. Lični podaci o zdravstvenom stanju i osuđivanosti građanina mogu se prikupljati, čuvati i davati na korišćenje samo u skladu sa zakonom."

Zakon o elektronskom potpisu (Sluzbeni glasnik Republike Srbije br. 135/04)
U nastavku su dati izvodi iz navedenog zakona :
« Ovim zakonom uređuje se upotreba elektronskog potpisa u pravnim poslovima i drugim pravnim radnjama, poslovanju, kao i prava, obaveze i odgovornosti u vezi sa elektronskim sertifikatima, ako posebnim zakonima nije drugačije određeno. Odredbe ovog zakona primenjuju se na opštenje organa, opštenje organa i stranaka, dostavljanje i izradu odluke organa u elektronskom obliku u upravnom, sudskom i drugom postupku pred državnim organom. 1)"Elektronski dokument" - dokument u elektronskom obliku koji se koristi u pravnim poslovima i drugim pravnim radnjama, kao i u upravnom, sudskom i drugom postupku pred državnim organom; 2)"Elektronski potpis" - skup podataka u elektronskom obliku koji su pridruženi ili su logički povezani sa elektronskim dokumentom i koji služe za identifikaciju potpisnika; 3)"Kvalifikovani elektronski potpis" - elektronski potpis kojim se pouzdano garantuje identitet potpisnika, integritet elektronskih dokumenata, i onemogućava naknadno poricanje odgovornosti za njihov sadržaj, i koji ispunjava uslove utvrđene ovim zakonom; 4)"Potpisnik" - lice koje poseduje sredstva za elektronsko potpisivanje i vrši elektronsko potpisivanje u svoje ime ili u ime pravnog ili fizičkog lica; 5)"Podaci za formiranje elektronskog potpisa" - jedinstveni podaci, kao što su kodovi ili privatni kriptografski ključevi, koje potpisnik koristi za izradu elektronskog potpisa; 6)"Sredstva za formiranje elektronskog potpisa" - odgovarajuća tehnička sredstva (softver i hardver) koja se koriste za formiranje elektronskog potpisa, uz korišćenje podataka za formiranje elektronskog potpisa; 7)"Sredstva za formiranje kvalifikovanog elektronskog potpisa" - sredstva za formiranje elektronskog potpisa koja ispunjavaju uslove utvrđene ovim zakonom; 8)"Podaci za proveru elektronskog potpisa" - podaci, kao što su kodovi ili javni kriptografski ključevi, koji se koriste za proveru i overu elektronskog potpisa; 9)"Sredstva za proveru elektronskog potpisa" - odgovarajuća tehnička sredstva (softver i hardver) koja služe za proveru elektronskog potpisa, uz korišćenje podataka za proveru elektronskog potpisa; 10)"Sredstva za proveru kvalifikovanog elektronskog potpisa" - sredstva za proveru elektronskog potpisa koja ispunjavaju uslove utvrđene ovim zakonom;

11)"Elektronski sertifikat" - elektronski dokument kojim se potvrđuje veza između podataka za proveru elektronskog potpisa i identiteta potpisnika; 12)"Kvalifikovani elektronski sertifikat" - elektronski sertifikat koji je izdat od strane sertifikacionog tela za izdavanje kvalifikovanih elektronskih sertifikata i sadrži podatke predviđene ovim zakonom; 13)"Korisnik" - pravno lice, preduzetnik, državni organ, organ teritorijalne autonomije, organ lokalne samouprave ili fizičko lice kome se izdaje elektronski sertifikat; 14)"Sertifikaciono telo" - pravno lice koje izdaje elektronske sertifikate u skladu sa odredbama ovog zakona. Elektronskom dokumentu se ne može osporiti punovažnost ili dokazna snaga samo zbog toga što je u elektronskom obliku.» Ovaj stav se ne primenjuje na: pravne poslove kojima se vrši prenos prava svojine na nepokretnosti ili kojima se ustanovljavaju druga stvarna prava na nepokretnostima; izjave stranaka i drugih učesnika u postupku za raspravljanje zaostavštine, formu zaveštanja, ugovore o ustupanju i raspodeli imovine za života, ugovore o doživotnom izdržavanju i sporazume u vezi sa nasleđivanjem, kao i druge ugovore iz oblasti naslednog prava; ugovore o uređivanju imovinskih odnosa između bračnih drugova;ugovore o raspolaganju imovinom lica kojima je oduzeta poslovna sposobnost; ugovore o poklonu; druge pravne poslove ili radnje, za koje je posebnim zakonom ili na osnovu zakona donetih propisa, izričito određena upotreba svojeručnog potpisa u dokumentima na papiru ili overa svojeručnog potpisa. Ako je zakonom ili drugim propisom predviđeno da određeni dokument treba čuvati, to se može učiniti i u elektronskom obliku, pod uslovom da je elektronski dokument: dostupan i da je na raspolaganju za kasniju upotrebu;sačuvan u obliku u kome je formiran ili primljen; sačuvan na način koji omogućava identifikaciju vremena i mesta nastanka ili prijema, i lica koje ga je formiralo; formiran primenom tehnologije i postupaka koji omogućavaju da se na pouzdan način može utvrditi bilo kakva izmena u elektronskom dokumentu. Kvalifikovani elektronski potpis, mora da zadovolji sledeće uslove: isključivo je povezan sa potpisnikom; nedvosmisleno identifikuje potpisnika; nastaje korišćenjem sredstava kojima potpisnik može samostalno da upravlja i koja su isključivo pod nadzorom potpisnika; direktno je povezan sa podacima na koje se odnosi, i to na način koji nedvosmisleno omogućava uvid u bilo koju izmenu izvornih podataka; formiran je sredstvima za formiranje kvalifikovanog elektronskog potpisa; proverava se na osnovu kvalifikovanog elektronskog sertifikata potpisnika. Sredstva za formiranje kvalifikovanog elektronskog potpisa su sredstva koja moraju da obezbede: da se podaci za formiranje kvalifikovanog elektronskog potpisa mogu pojaviti samo jednom i da je obezbeđena njihova poverljivost; da se iz podataka za proveru kvalifikovanog elektronskog potpisa, ne mogu u razumno vreme i trenutno dostupnim sredstvima, dobiti podaci za formiranje kvalifikovanog elektronskog potpisa; da kvalifikovani elektronski potpis bude zaštićen od falsifikovanja, upotrebom trenutno dostupne tehnologije; da podaci za formiranje kvalifikovanog elektronskog potpisa budu pouzdano zaštićeni od neovlašćenog korišćenja. Sredstva za formiranje kvalifikovanog elektronskog potpisa, prilikom formiranja potpisa, ne smeju promeniti podatke koji se potpisuju ili onemogućiti potpisniku uvid u te podatke pre procesa formiranja kvalifikovanog elektronskog potpisa. Sredstva za proveru kvalifikovanog elektronskog potpisa su sredstva koja obezbeđuju: pouzdano utvrđivanje da podaci korišćeni za proveru elektronskog potpisa odgovaraju podacima prikazanim licu koje vrši proveru; pouzdano verifikovanje potpisa i korektno prikazivanje rezultata provere; omogućavanje pouzdanog uvida u sadržaj potpisanih podataka; pouzdano verifikovanje autentičnosti i validnosti elektronskog sertifikata potpisnika u trenutku provere elektronskog potpisa; korektno prikazivanje identiteta potpisnika; da se bilo koje izmene u potpisanim podacima pouzdano otkriju. Kvalifikovani elektronski potpis u odnosu na podatke u elektronskom obliku ima isto pravno dejstvo i dokaznu snagu kao i svojeručni potpis, odnosno svojeručni potpis i pečat, u odnosu na podatke u papirnom obliku. Elektronski sertifikat, u smislu ovog zakona, je elektronska potvrda kojom se potvrđuje veza između podataka za proveru elektronskog potpisa i identiteta potpisnika. Elektronske sertifikate izdaje sertifikaciono telo. Kvalifikovani elektronski sertifikat može se izdati svakom licu na njegov zahtev, na osnovu nesumnjivo utvrđenog identiteta i ostalih podataka o licu koje je podnelo zahtev.«

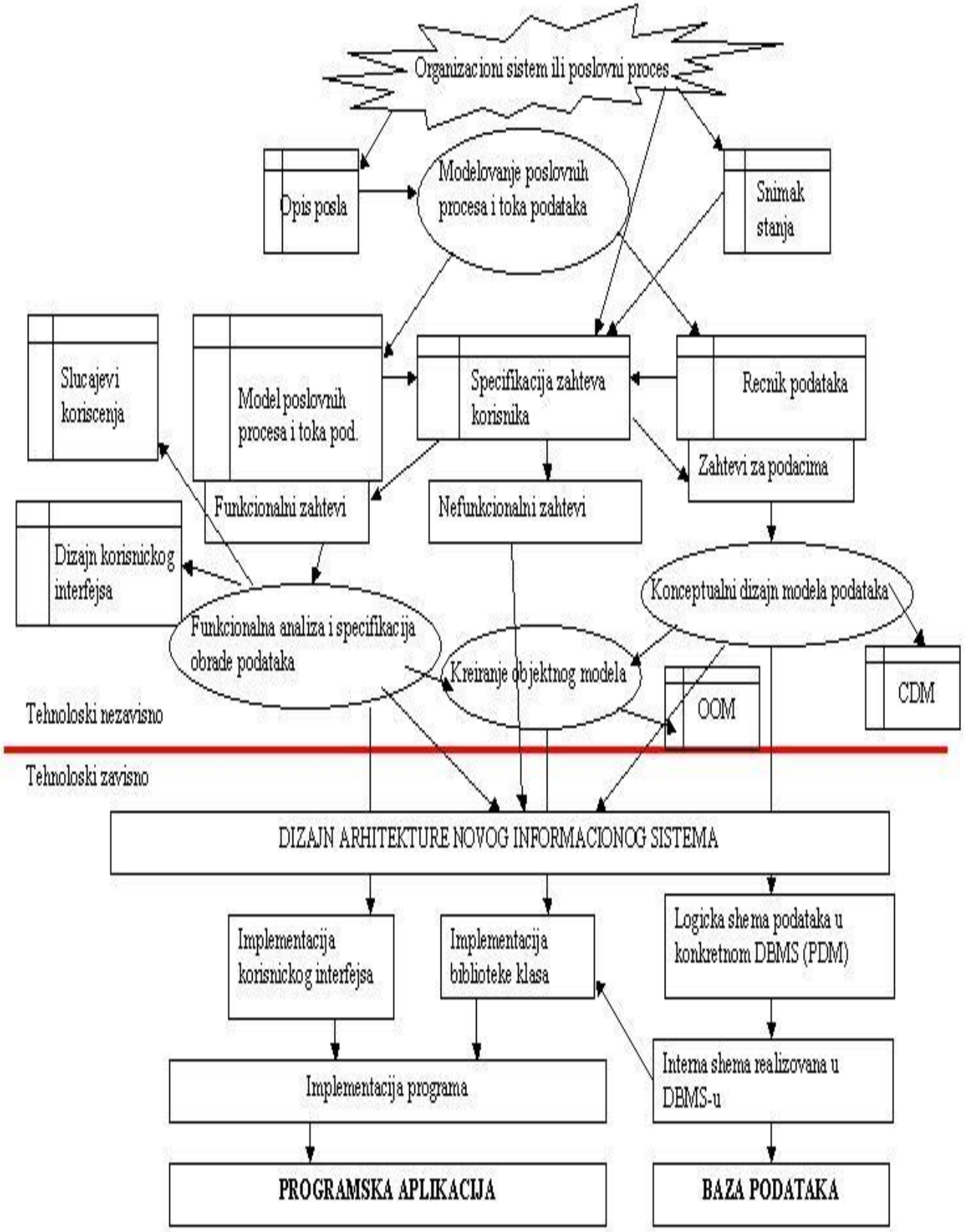
2.MODEL PRAKTIČNE NASTAVE U OBLASTI RAZVOJA INFORMACIONOG SISTEMA

Teorijski kontekst praktične nastave predstavljaju faze razvoja informacionog sistema i prateći rezultati (prikazani na tabeli 1.)

Tabela 1. Faze razvoja i rezultati razvoja informacionog sistema

FAZE RAZVOJA INFORMACIONOG SISTEMA	**REZULTAT**
1. Upoznavanje sa organizacionim sistemom	Opis posla, zakonska regulativa i pravilnici Poslovni rečnik Model poslovnih procesa i toka podataka Rečnik podataka Snimak stanja postojećeg informacionog sistema
2. Specifikacija zahteva korisnika	Preslikavanje poslovnih procesa u softverske funkcije, ponuda rešenja Specifikacija zahteva korisnika: definicija profila korisnika i njihovih pojedinačnih zahteva za funkcijama Definisati prioritete razvoja (odvojiti funkcije koje direktno pokrivaju poslovne procese „need to have" i dodatne „nice to have" opcije) i rizike
3. Dizajn sistema	Modeli podataka – konceptualni (ER), fizički (relacioni), objektni UML modeli Dizajn korisničkog interfejsa Dizajn izveštaja
4. Konstrukcija prototipa	Implementacija baze podataka Programski kod
5. Zahtevi za izmenama od korisnika – prilagođavanje potrebama i zahtevima korisnika	Evidencija zahteva za izmenama Evidencija izvršenih izmena
6. Kreiranje sistema	Implementacija baze podataka Programski kod
7. Testiranje	Test – primeri, izveštaj o testiranju
8. Dokumentovanje	Dokumentacija za korisnika (korisničko uputstvo) i održavanje sistema (tehničko uputstvo)
9. Isporuka	Instalacija opreme, Postavljanje baze podataka, Instalacija softvera Dokumentacija Obuka
10. Održavanje	Evidencija o promenama u održavanju

Koncept, odnosno model procesa u okviru praktične nastave iz oblasti projektovanja i implementacije informacionih sistema prikazan je na slici 1.



Slika 1. Model procesa nastave u oblasti informacionih sistema na Tehničkom fakultetu "Mihajlo Pupin" u Zrenjaninu (([1], proširenje u odnosu na [2])

U okviru praktične nastave u delu koji se odnosi na tehnološki nezavisne procese (sistemska analiza i dizajn), studenti učestvuju u praktičnoj realizaciji procesa upoznavanja sa organizacionim sistemom kroz opis posla i njegovu analizu, snimak stanja, specifikaciju zahteva korisnika, modelovanje poslovnih procesa, preslikavanje poslovnih procesa u softverske funkcije, modelovanje podataka – konceptualno, fizicko i objektno, izrada podmodela za svaku softversku funkciju.

U okviru praktične nastave u delu implementacije studenti se upoznaju sa razvojem višeslojnih "client-server" aplikacija u LAN (Local Area Network" okruženju pristupom bazi podataka na lokalnoj računarskoj mreži, kao i sa razvojem WEB aplikacije. U okviru praktične nastave koristi se Visual studio NET sa DBMS (DataBase Management System) SQL Server i ORACLE.

Tabela 2. Pregled arhitektura i tipova aplikacija

Arhitektura	**Desktop**	**Client-server (LAN)**	**Client-server (WEB)**
Dvoslojna	1. NETFramework (Exe) + DBMS (baza podataka)	1. Exe + DBMS (na mrezi)	1. Web browser Web server (ASPX) DBMS
Troslojna	2. NETFramework (Exe + dll) + DBMS (baza podataka)	2. Exe + dll + DBMS (na mrezi)	2. Web browser Web server (ASPX + dll) DBMS
Višeslojna	3. NETFramework (Exe + dll1 + dll2 ...) +DBMS (baza podataka)	3. Exe + dll1 + dll2 ... +DBMS (na mrezi)	2. Web browser Web server (ASPX + dll1 + dll2 + dll3) DBMS

Koncepti višeslojne arhitekture prezentuju se studentima u kontekstu podrške upravljanju promenama. Viseslojna arhitektura zasniva se na korišćenju biblioteka klasa kojima se realizuju pojedinačni slojevi. (Tabela 3.)

Tabela 3. Elementi višeslojne arhitekture

PREZENTACIONI SLOJ	KORISNICKI INTERFEJS
	Klase korisničkog interfejsa (u skladu sa strukturom formi i industrijske klase)
SLOJ POSLOVNE LOGIKE	Klase radnih tokova
	Klase poslovne logike (slične strukturi dokumenata, entitetima i pojmovima poslovnog sveta, poslovna pravila i radni tokovi) sa podrškom za implementaciju poslovnih pravila
SLOJ PODATAKA	Klase podataka (slične tabelama iz baze podataka)
	Tehnološke univerzalne klase za pristup bazi podataka - sopstvene i industrijske
	DBMS

Osnovni elementi funkcionalnosti softvera informacionih sistema prikazani su u Tabeli 4.

Tabela 4. Osnovne funkcije softvera u oblasti informacionih sistema

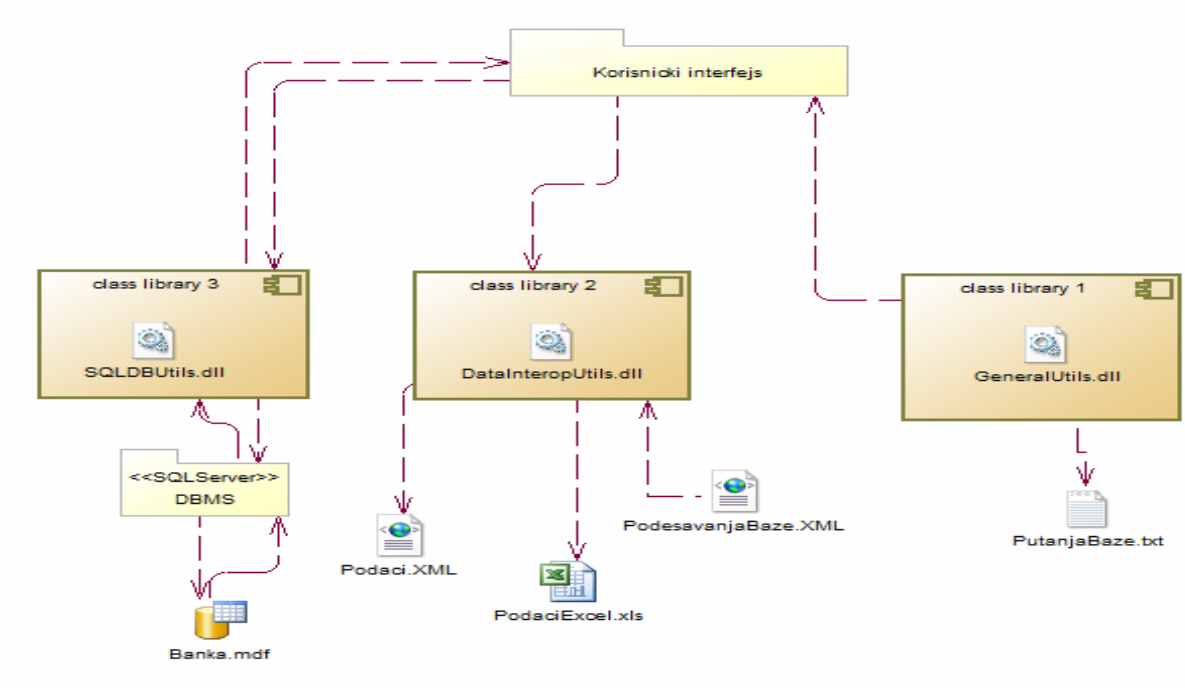
AŽURIRANJE PODATAKA	Ažuriranje - unos, brisanje, izmena
	Rad sa raznim tipovima podataka: string, celobrojni, realan, datumski (formati, konverzije tipova)
	Transakcije
INTEROPERABILNOST (RAZMENA) PODATAKA	Kreiranje i brisanje XML fajlova
	Rad sa txt datotekama
	Rad sa xls (MS Excel) fajlovima
PRETRAGA PODATAKA	Navigacija - prvi, prethodni,sledeći, poslednji
	Tabelarni prikaz, Pozicioniranje na zapis
	Filtriranje prema različitim kriterijumima, prikaz svih podataka
PREZENTOVANJE PODATAKA	Rad sa izveštajima (reportima)
	Rad sa grafikonima

Osnovni tipovi ekranskih formi u odnosu na složenost koji se ilustruju i/ili realizuju na časovima vezbi su predstavljeni na tabeli 5.

Tabela 5. Tipovi ekranskih formi u softveru informacionih sistema

Jednostavna forma	1 tabela
Osnovna forma	2 tabele - 1 glavna, 2. Pomocna (puni combo)
Master – detail forma	više tabela, forma 1 (zaglavlje) :M (stavke)

Opšta struktura aplikacija koja se implementira u praktičnom delu nastave sastoji se od sledecih komponenti:



Iz korisnickog interfejsa (Windows ili Web aplikacija) povezujemo se sa bibliotekama klasa za rad sa SQL Server bazom podataka, za rad sa XML i XLS fajlovima, kao i tekstualnim datotekama. Putanja baze (ako je u pitanju MS Access baza podataka) čuva se u posebnom fajlu PutanjaBaze.txt, a u XML fajlu PodesavanjaBaze.XML ako je u pitanju SQL Server baza podataka. Korisnicki interfejs posredstvom odgovarajućih klasa čita podešavanja baze i šalje sve potrebne podatke odgovarajućoj biblioteci za rad sa bazom podataka (SQLDBUtils za SQLServer ili AccessDBUtils za rad sa MSAccess bazom podataka), koja na osnovu toga pristupa bazi podataka i otvara konekciju. Sva dalja obraćanja bazi podataka dešavaju se iz korisnickog interfejsa sa nazivima tabela i SQL upitima, koji se prosledjuju odgovarajućoj klasi, a zatim se ona obraća bazi podataka radi izvršavanja upita i realizacije svih potrebnih operacija koje su zadate iz korisničkog interfejsa posredstvom te klase.

Literatura

[1] Kazi Lj, *Integracija metoda za konceptualno modelovanje podataka u nastavi informacionih sistema*, Pupin Almanah, Tehnicki fakultet Mihajlo Pupin Zrenjanin, 2012.
[2] Elmasri R., & Navathe S.B. (2007), *Fundamentals of Database Systems*, 5th edition, Pearson International Edition

3. ANALIZA ORGANIZACIONOG SISTEMA I PLANIRANJE PROJEKTA

3.1. Tehnike upoznavanja sa organizacionim sistemom

Organizacioni sistem upoznajemo primenom tehnika:

- Analize zakonske regulative i pravilnika iz navedene oblasti
- Analize internih propisa organizacije, ako postoji i dokumentaciju ISO9000 (procedure, radna uputstva). Posebno analiziramo organizacionu shemu i pravilnik o sistematizaciji radnih mesta, gde su opisane aktivnosti i odgovornosti različitih učesnika u poslovnim procesima organizacije.
- Analize dokumentacije, odnosno obrazaca dokumenata
- Intervjua sa konkretnim pitanjima koja se odnose na funkcionisanje organizacije, probleme, potrebe i stanje postojećeg informacionog sistema
- Upitnika koji se dostavlja zaposlenima u osnovnoj, pomoćnoj i upravljačkoj delatnosti, a odnosi se na funkcionisanje organizacije, probleme, potrebe i stanje postojećeg informacionog sistema
- Direktnim praćenjem načina rada u okviru organizacije.

Na ovaj način poznajemo se sa načinom poslovanja sistema, vršimo snimak stanja i specifikaciju zahteva korisnika, što nam daje osnovu za definisanje elemenata projekta uvođenja novog informacionog sistema.

3.2. Upoznavanje sa semantikom poslovanja

Obuhvat obrade poslovne analize može biti organizacija (npr. Fabrika), organizaciona jedinica (Računovodstvo) ili jasno definisan poslovni proces (Obrada troškova službenog puta). Za treću navedenu vrstu dat je primer opisa posla koji se koristi u realizaciji zadataka.

OBRADA TROŠKOVA SLUŽBENOG PUTA

Direktor organizacije odlučuje o potrebi za službenim putom radi realizovanja nekog poslovnog zadatka. Odluku dokumentuje u obliku naloga za službeni put. Sekretarica direktora unosi podatke, a direktor potpisuje nalog. Nalog se dostavlja zaposlenom koji je određen za put, ali pre toga se dostavlja računovodstvu na obradu. Ako ima više zaposlenih, svak dobija svoj nalog. Zaposleni je putem naloga i usmenog razgovora sa upravom ili nekog pratećeg dokumenta opisa delatnosti na službenom putu obavešten o svrsi, aktivnostima i očekivanim rezultatima na službenom putu. Zaposleni odlučuje o načinu prevoza. Ako se opredeli za putničko vozilo, može da bira između službenog i privatnog vozila i ako je predviđeno pravilnikom, dobija od računovodstva bonove za gorivo. Računovodstvo na osnovu planiranog putovanja i želje zaposlenog (koju evidentira na putnom nalogu), zaprima od NIS-a određen broj bonova u dinarskoj protivvrednosti. Ukoliko nisu predviđeni ili zaprimljeni bonovi, zaposleni tankuje gorivo na pumpi, plaća vlastitim novcem i po povratku donosi gotovinski i fiskalni račun radi refundacije novca. Ukoliko su pravilnikom predviđene dnevnice za službeni put, radnih računovodstva formira nalog za isplatu dnevnica, direktor odobrava, a zatim zaposleni dobija gotovinu. Za određenu kilometražu predviđene su odgovarajuće količine goriva koje se mogu refundirati. Ostatak novca ukoliko je više potrošeno se ne refundira. Zajedno sa nalogom za put dostavljaju se svi gotovinski računi računovodstvu koji se prosleđuju upravi radi odobravanja. Nakon odobravanja, računovodstvo podiže iz banke odgovarajući iznos novca i obaveštava zaposlenog o odobravanju i mogućnosti isplate gotovine. Zaposleni dolazi i prima gotovinu, a potpisuje isplatnu listu da je novac primio.

ZADATAK: Na osnovu analize opisa posla i druge dokumentacije odrediti:

- Rečnik poslovnih termina sa objašnjenjima
- Svrha postojanja sistema (osnovne, upravljačke i pomoćne delatnosti - razgraničiti)
- Osnovni objekti obrade u sistemu

- Organizaciona struktura sistema (organizacione jedinice organizacije, pripadnost drugim većim organizacionim celinama, povezanost sa okruženjem)
- Radna mesta i uloge (spisak i opis radnih mesta, opis posla radnih mesta, opis odgovornosti i ovlašćenja)
- Tok aktivnosti obrade osnovnih objekata obrade (životni ciklus obrade)
- Dokumentacija koja se koristi ili je rezultat rada aktivnosti (formulari, knjige evidencija itd.)
- Okruženje sistema i komunikacija sa sistemom
- Način arhiviranja podataka u sistemu
- Zakonsku regulativu i pravila poslovanja u posmatranom organizacionom sistemu

3.3. Snimak stanja postojećeg informacionog sistema i planova razvoja

ZADATAK: Za dati sistem odrediti:

1. POSTOJEĆE STANJE INFORMACIONOG SISTEMA

   Prikazati postojeće stanje informacionog sistema, odnosno navesti karakteristike postojećih elemenata:
   - SOFTWARE – program i baza podataka, operativni sistem, komunikacioni softver itd.
   - HARDWARE – računarska oprema: računari, štampači, sistem za backup, računarska mrežna i komunikaciona oprema
   - ORGWARE – organizacija rada izražena kroz opis postupaka rada informacionog podsistema organizacionog sistema – znači kako se postupa sa podacima u svim fazama njegove obrade od prikupljanja do arhiviranja, u zajedničkom delovanju sa osnovnim aktivnostima organizacionog sistema u ostvarivanju svoje svrhe postojanja
   - LIFEWARE – ljudi i njihova podela radnih uloga u organizacionom sistemu i u informacionom sistemu, zaduženja, odgovornosti i ovlašćenja

   Opisati postojeće postupke: prikupljanja, obrade, prenosa, prezentovanja i arhiviranja podataka, sa tehnološkog i organizacionog aspekta.

2. SNIMAK PLANOVA RAZVOJA SISTEMA

   Evidentirati da li postoje dugoročni planovi razvoja informacionog sistema organizacije ili šireg organizacionog konteksta gde ova organizacija pripada. Evidentirati da li postoje započeti projekti celovitog ili parcijalnog pristupa informacionom sistemu organizacije ili je u toku razvoj pojedinačnih aplikacija ili uvođenje aplikacije. Evidentirati zakonske obaveze povodom upotrebe aplikacija u regionu-državnom nivou i pratiti razvoj planova i aplikacija koje omogućavaju integraciju sa drugim organizacijama regiona ili sektora delatnosti.

3.4.Specifikacija zahteva korisnika o potrebnim karakteristikama novog sistema

ZADATAK: U okviru datog sistema odrediti:

1. Probleme postojećeg načina rada

   Potrebno je uočiti probleme postojećeg informacionog sistema i kako to utiče na uspešnost funkcionisanja i izvršavanja osnovnih delatnosti realnog sistema. Koje greške, neusaglašenosti, neažurnosti, neefikasnosti se mogu javiti ili postoje u poslovanju zahvaljujući postojećem informacionom sistemu.
2. Evidentiranje eksplicitno izraženih zahteva i potreba

   Potrebe su eksplicitno izraženi zahtevi za funkcijama softvera i njihovom načinu izvršavanja a vezani za konkretnu problematiku posmatranog informacionog sistema. Među tim zahtevima, mogu se naći opšti zahtevi za osobinama korisničkog interfejsa, koji mogu da važe za bilo koji drugi softver. Ove opšte zahteve vezane za korisnički interfejs nije potrebno navoditi, s obzirom da se podrazumevaju.

   Primeri:

Potrebe:
- Mogućnost razduživanja pojedinačnih slika sa zajedničkog reversa.
- Glavna knjiga (finansijsko knjigovodstvo) – Otvaranje i zatvaranje poslovne godine.
- Mogućnost štampanja izveštaja REVERS u Galeriji sa ili bez kolone »Vrednost slike«.

Opšte funkcije softvera:
- Validacija podataka prilikom unosa.
- Štampanje različitih izveštaja.
- »User Friendly« korisnički intefejs.
- Pretrage podataka i sortiranje po jednom ili više kriterijuma.

Eksplicitno izraženi zahtevi za funkcijama softvera i njihovom načinu izvršavanja a vezani za konkretnu problematiku posmatranog informacionog sistema izražavaju se dokumentom SPECIFIKACIJA ZAHTEVA KORISNIKA (»client requirements document«) koji najčešće ima sledeću strukturu:
- svrha softvera
- definicije pojmova i skraćenice
- poslovni ciljevi (Bussiness objectives) - šta se želi postići u poslovanju - ušteda vremena, pokrivanje kojih poslovnih aktivnosti i na koji način
- Opis rada sistema (overal description) - hronološki opis kako će raditi ceo sistem sa uključenim novim softversko/hardverskim rešenjima
- Detaljni opis rada sa softverskim sistemom - koji se ekran kada otvara i koja opcija koristi za izvršavanje poslova. Šta treba od funkcija da sadrži koji ekran. Opšte osobine (softverskog) sistema - features i detaljnije odredjenje - requirements
- poslovna pravila (rules) koja treba automatski da se izvrsavaju - ogranicenja.
- potrebni izvestaji (reports) - ekranski i papirni, do nivoa elementarnog podatka
- potrebni upiti
- Buduce osobine softvera - nece biti ukljucene u ovoj verziji, a mogu da se kasnije dodaju kao unapredjenje, novi modul i sl. (statistika, razmena podataka i sl.)
- Zahtevi za osobinama korisničkog interfejsa
- Bezbedonosni zahtevi - nacin zastite podataka od neovlascenog prikazivanja, menjanja, kopiranja, pristupanja, backup podataka / replikacija itd.

3. Evidentiranje ciljeva i interesa rukovodstva i zaposlenih
   U toku upoznavanja sa realnim organizacionim sistemom, jasno izraženi ili skriveni ciljevi razvoja novog informacionog sistema dolaze do izražaja. Jedan od kriterijuma uspeha celog projekta u smislu subjektivnog zadovoljstva rezultatima je zadovoljavanje ciljeva i interesa rukovodstva i zaposlenih. Da bi se izbegla subjektivnost u proceni kvaliteta softvera, potrebno je u toku potpisivanja ugovora o izradi informacionog sistema dobiti i formalnu saglasnost sa dokumentom SPECIFIKACIJA ZAHTEVA KORISNIKA koja sadrži objektivno nabrojane elemente i karakteristike budućeg rešenja. Sve dodatne karakteristike potrebno je zahtevati u aneksu ugovora i sl. Naravno, u smislu zadovoljenja očekivanja svih učesnika iz organizacije, potrebno je poštovati prioritet hijerarhije radnih mesta – od rukovodstva ka nižim hijerarhijskim nivoima radnih mesta...
4. Evidentiranje objektivno utvrđenih potreba
   Na osnovu posmatranja i poznavanja organizacionog sistema, jasno se mogu utvrditi potrebe organizacionog sistema za softverom određenih karakteristika, koje ne zavise od pojedinačnih zahteva zaposlenih.
5. Evidentiranje odluka i informacione podrške odlukama
   Koje se odluke donose u sistemu, na osnovu čega i kojih podataka. Jedan od osnovnih ciljeva informaciono-upravljačkih sistema je obezbeđivanje informacione osnove za donošenje odluka. Softverska realizacija informacione podrške odlučivanju najčešće je u obliku parametarskih upita ili dijagrama i grafikona.
6. Evidentiranje potrebnih automatizama u softverskom rešenju
   Navesti šta sve treba program da radi automatski.
   Primer:

- Automatsko zatvaranje poslovne godine, izrada bilansa stanja, bilansa uspeha, zaključnog lista i otvaranje nove poslovne godine sa prenosom svih konta koja imaju saldo (Finansijsko knjigovodstvo).
- Automatsko smeštanje slike na smeštajno mesto nakon što je vraćena od strane komitenta koji se razdužio.

7. Evidentiranje zakonski izraženih potreba i ograničenja
8. Odnos prema organizaciji rada

   Cilj svakog razvoja informacionog sistema je unapređenje poslovanja, što često dovodi do potrebe i za reorganizacijom postojećeg načina rada organizacionog sistema.

   Reorganizacijom postojećeg informacionog sitema javljaju se sledeće organizacione implikacije:

   - Potreba za novim radnim mestima za elektronsku obradu podataka ukoliko ista ne postoje,
   - Dopuna opisa posla postojećih radnih mesta,
   - Sistematizacija, usaglašavanje, pa čak i izrada novih poslovnih pravila i dokumentacije,
   - Obaveza/mogućnost evidentiranja podataka koja do tada nije postojala,
   - Obaveza/mogućnost za izvršavanje određenih aktivnosti u realnom sistemu.

   Programi treba da budu dovoljno fleksibilni da ne primoravaju korisnika na izvršavanje određenih aktivnosti u realnom sistemu osim ukoliko to nije eksplicitni zahtev. Korisnik nejčešće ne želi da uvođenje novog IS-a utiče na bilo kakve organizacione promene, kao ni na navike u dosadašnjem radu. Ipak, projektanti novog IS-a treba da nastoje da unaprede postojeći IS, koji obuhvata pored softverske podrške i unapređenje postojeće organizacije i metdologije rada.
9. Utvrđivanje grupa zahteva i prioriteta u razvoju

3.5. Definisanje elemenata projekta

ZADATAK: Za predlog novog informacionog sistema potrebno je:

- Definisanje ciljeva projekta

  Postojeći informacioni sistem ne zadovoljava poslovne potrebe. Potrebno je navesti postojeće probleme i na koji način će biti realizovano rešavanje ovih problema. Ciljevi projekta predstavljaju zadovoljavanje potreba i zahteva korisnika, kao i rešavanje problema u postojećem načinu rada.
- Definisanje obuhvata posla – celine, podsistema i prioriteti razvoja

  Potrebno je prikazati celinu informacionog sistema, podsisteme i njihovu povezanost, odnos prema organizacionim jedinicama. Navesti redosled implementacije rešenja. Savetuje se da se najpre uvode moduli koji se odnose na osnovnu delatnost, zatim pomoćnu, a na kraju, upravljačku delatnost, a u okviru osnovne delatnosti, redosled uvođenja treba da prati redosled životnog cilusa osnovnog objekta obrade. Naravno, prioritete određuju i korisnici, prema svojim potrebama.
- Definisanje rezultata projekta – opis i karakteristike kvaliteta

  Jasno definisati šta su rezultati projekta – programska rešenja sa određenim funkcionalnim karakteristikama (dijagram slučaja korišćenja) koja su razmeštena po određenim lokacijama u org. Jedinicama (razmeštaj se može prikazati dijagramom razmeštaja). Ova programska rešenja omogućavaju podršku određenim poslovima organizacije, odnosno radnim mestima, što takođe treba navesti.
- Definisanje ograničenja i pretpostavki razvoja

  Prilikom razvoja softvera potrebno je jasno definisati ograničenja koja se javljaju u organizacionom sistemu ili razvojnom okruženju kako bi se unapred definisale mogućnosti dobijenog rešenja. Takođe, potrebno je opisati polazne pretpostavke

sa kojima se započinje proces razvoja softvera, a odnose se na organizacioni sistem, pravila rada, razvojno okruženje, opremu i sl.

- Studija opravdanosti (procena troškova i dobiti)

  Da li je opravdano pokrenuti razvoj novog informacionog sistema? Šta time dobijamo, a koji su troškovi? Da li će troškovi koji pre svega se odnose na novac, biti nadoknađeni dobiti, odnosno onim što se dobija kada novi sistem bude počeo da se koristi? Odnos dobit-troškovi se pre svega odnosi na novac – koji iznos novca se štedi novim sistemom, odnosno koji su troškovi korišćenja novog sistema u poređenju sa starim sistemom? Naravno, postoje i drugi pokazatelji koristi, osim novca – brzina rada i obrade objekata obrade, preciznost i odsustvo kvalitativnih grešaka u radu, uvođenje reda... Mnoge kvalitativne karakteristike rada koje se ne mogu izmeriti novcem takođe predstavljaju elemente opravdanosti pokretanja projekta, međutim, finansijski aspekt je uvek najvažniji. Potrebno je pokazati koji su objektivni troškovi korišćenja novog i starog sistema i prikazati da postoji ušteda novca koja će s vremenom isplatiti troškove projekta.
- Plan razvoja po izvršiocima, fazama aktivnosti, podsistemima i troškovima

  Prikazati vremenski sled aktivnosti razvoja informacionog sistema, planiran početak i kraj razvoja i uvođenja svakog od podsistema, kao i kontrolne tačke u toku razvoja, zatim odrediti izvršioce pojedinih aktivnosti, rezultate pojedinih aktivnosti i troškove pojedinih aktivnosti. Na ovaj način se dobija pregled celine rada na projektu, rezultata i vremenskih odrednica isporuke pojedinih rezultata.
- Studija izvodljivosti

  Da li je u okviru postojećih ograničenja i planiranih aktivnosti-izvršioca-troškova moguće realno izvesti projekat do kraja uspešno... Potrebno je proceniti preciznije na osnovu obuhvata posla (broj ekrana, broj izveštaja, broj upita nad bazom itd.) da li je sa postojećim kadrom i u okviru datih vremenskih ograničenja moguće realno završiti projekat u planiranim terminima... Tu su naravno i drugi aspekti – postojeća oprema, alati i znanje koje je na raspolaganju za implementaciju itd. Jedna od jednostavnih preglednih tehnika analize izvodljivosti i pripreme za analizu rizika je SWOT analiza – Strength, Weakness, Opportunity, Threat ... matrica u okviru koje se evidentiraju pojedini aspekti projekta i pregledno prikazuju, kako bi se pripremila osnova analize izvodljivosti projekta.
- Određivanje kriterijuma kvaliteta i zadovoljstva korisnika

  Potrebno je definisati šta predstavlja očekivanja korisnika u smislu kvalitativnih karakteristika softvera, koje će biti osnov za evaluaciju realizovanog rezultata. Najčešće karakteristike obuhvataju sveobuhvatnost, adaptibilnost, performanse, bezbednost, pouzdanost itd. Bitno je istaći redosled karakteristika po prioritetu, posebno kada je u pitanju sveobuhvatnost u odnosu na vreme izrade.
- Definisanje rizika projekta i tehnika upravljanja rizikom

  Svaki projekat ima svoje rizike. Rizici se odnose najčešće na ljudski faktor, raspoloživo vreme, opremu i novac koji ograničavaju rad i dovode do potencijalnog problema i rizika od neuspeha ili parcijalnog uspeha projekta: prekoračenja vremena izrade i potrebnog kvaliteta i funkcionalnih karakteristika softverskog rešenja. Da bi se sprečilo da se ovi problemi dogode, potrebno je planski razmotriti šta je potrebno uraditi da ne dođe do ovih problema i ukoliko dođe, kako bi se ovi problemi rešili.

3.6. Rešeni zadaci

IDEJNI PROJEKAT NOVOG INFORMACIONOG SISTEMA
ORGANIZACIONI SISTEM: Biblioteka fakulteta

Napomena:
Navedeni primer skice idejnog projekta novog informacionog sistema fakultetske biblioteke urađen je kao probni primer fakultetske biblioteke (ne odnosi se ni na jednu realnu biblioteku).

1. UPOZNAVANJE SA ORGANIZACIONIM SISTEMOM

1.1. Primenjene tehnike i postupci

Prikupljena je dokumentacija, odnosno obrasci, knjige evidencija i ostali dokumenti.
Izvršen je intervju sa bibliotekarom i razmotreno postojeće stanje, problemi i predlozi rešenja.Sagledano je stanje evidencije i računarske opreme na licu mesta dana 1.10.2008.

1.2. Analiza sistema

- Svrha postojanja sistema (osnovne, upravljačke i pomoćne delatnosti - razgraničiti)

  SVRHA POSTOJANJA SISTEMA: izdavanje bibliotečkih jedinica na korišćenje korisnicima (članovima) biblioteke

  OSNOVNA DELATNOST – izdavanje i evidentiranje zaduživanja i vraćanja bibliotečkih jedinica, obrada korisnika biblioteke – učlanjivanje, ispis, praćenje zaduživanja

  POMOĆNA DELATNOST – nabavka, prijem i arhiviranje bibliotečkih jedinica

  UPRAVLJAČKA DELATNOST – odlučivanje o nabavci bibliotečkih jedinica, odlučivanje o rashodovanju bibliotečkih jedinica, pripremanje odluke povodom upisa semestra za studentsku službu (iz evidencije zaduženja korisnika)
- Osnovni objekti obrade u sistemu

  BIBLIOTEČKA JEDINICA, KORISNIK BIBLIOTEKE
- Organizaciona struktura sistema (organizacione jedinice organizacije, pripadnost drugim većim organizacionim celinama, povezanost sa okruženjem)

  ORG JEDINICE: BIBLIOTEKAR JE JEDINI ZAPOSLENI, NEMA DRUGIH ORG. JEDINICA

  PRIPADNOST: FAKULTET, UNIVERZITETSKE BIBLIOTEKE

  POVEZANOST SA OKRUŽENJEM:

  1. INTERNO: DEKANAT, KADROVSKA SLUŽBA, STUDENTSKA SLUŽBA, SKRIPTARNICA, RAČUNOVODSTVO
  2. EKSTERNO: DOBAVLJAČI, MATICA SRPSKA, IZDAVAČI, UNIVERZITETSKA BIBLIOTEKA
- Radna mesta i uloge (spisak i opis radnih mesta, opis posla radnih mesta, opis odgovornosti i ovlašćenja)

  RADNO MESTO: BIBLIOTEKAR

  OPIS RADNOG MESTA: DATO JE U PRAVILNIKU O SISTEMATIZACIJI RADNIH MESTA: obrada nabavke bibliotečke građe, obrada zaduživanja i razduživanja izdavanja bibliotečkih jedinica korisnicima
- Tok aktivnosti obrade osnovnih objekata obrade (životni ciklus obrade)

  OBJEKAT OBRADE: BIBLIOTEČKA JEDINICA

  Bibliotečka jedinica može biti: završni rad (diplomski, specijalistički, magistarski, doktorat), knjiga, časopis, CD, udžbenici fakulteta.

  1. Završni rad stiže u biblioteku iz studentske službe nakon predaje od strane studenta. Vrši se evidentiranje u knjizi završnih radova, dobija karton i inventarni broj. Odlaže se na odgovarajuću policu.
  2. Knjiga se nabavlja periodično, u skladu sa zahtevima katedri ili nastavnika. Katedra ili nastavnik dostavlja spisak knjiga i dobavljača-izdavača. Bibliotekar vrši proveru podataka radi formiranja narudžbenice. Narudžbenicu potpisuje dekan. Narudžbenica se šalje faxom, e-mailom ili putem on-line forme na sajtu dobavljača. Nakon toga stiže predračun, koji dekan mora da overi, odnosno odobri. Predračun stiže obično faksom ili poštom, dakle obavezno u papirnoj formi. Predračun se dostavlja računovodstvu. Računovodstvo formira virman i podnosi dekanu na potpis, a kurir odnosi u banku da se izvrši uplata. Nakon što je uplata evidentirana, dobavljač pakuje i šalje knjige. Po prispeću knjiga, dekanat zavodi prijem paketa, prosleđuje

knjige biblioteci. Svaka knjiga dobija inventarni broj i odlaže se na odgovarajuću policu.

3. Časopis se nabavlja tako što nastavno osoblje ili katedra dostavljaju zahtev za nabavkom časopisa u određenoj školskoj ili kalendarskoj godini. Bibliotekar vrši proveru podataka radi formiranja narudžbenice. Narudžbenicu potpisuje dekan. Narudžbenica se šalje faxom, e-mailom ili putem on-line forme na sajtu dobavljača. Nakon toga stiže predračun, koji dekan mora da overi, odnosno odobri. Predračun stiže obično faksom ili poštom, dakle obavezno u papirnoj formi. Predračun se dostavlja računovodstvu. Računovodstvo formira virman i podnosi dekanu na potpis, a kurir odnosi u banku da se izvrši uplata. Nakon što je uplata evidentirana, časopis stiže poštom periodično u skladu sa periodom izlaska časopisa. Po prispeću časopisa dekanat zavodi prijem pošte, prosleđuje časopis biblioteci. Svaki časopis dobija inventarni broj i odlaže se na odgovarajuću policu.
4. CD izdanja dobijaju se uz završni rad, uz časopis ili knjigu, tako da dobijaju isti inventarni broj kao i osnovna bibliotečka jedinica, sa posebno dodatom oznakom da je u pitanju prateći CD. Posebna CD izdanja se nabavljaju isto kao i knjige.
5. Udžbenici fakulteta nastaju tako što nastavno osoblje pripremi udžbenik u elektronskoj formi i na pausima. Pausi se dostavljaju dekanatu, koji šalje biblioteci Matice srpske radi zavođenja u evidenciju i dobijanja ISBN brojeva. Nakon toga rukopis u pausima se šalje u štampariju. Štamparija šalje predračun, a fakultet plaća troškove štampanja udžbenika. Kada knjiga stigne iz štamparije prijem evidentira dekanat. Određen broj primeraka dobija besplatno autor, jedan broj primeraka odlazi u biblioteku, a ostatak ostaje u skriptarnici radi prodaje studentima. Po prijemu udžbenika u biblioteci, udžbenik dobija karton, inventarni broj i smešta se na odgovarajuće police.

OBJEKAT OBRADE: KORISNIK

Korisnici biblioteke su: zaposleni fakulteta ili studenti.

1. Zaposleni fakulteta nakon što budu primljeni na posao, treba da se jave bibliotekaru kako bi im se otvorio karton člana. Zaposleni imaju pravo da koriste knjige iz bibliotečkog fonda u skladu sa pravilnikom o radu biblioteke, što znači da vreme koje im je na raspolaganju povodom iznajmljivanja knjige je duže nego studentima. Nakon prestanka radnog odnosa, zaposleni mora da vrati sve knjige i tada prestaje da bude korisnik biblioteke.
2. Studenti nakon upisa dobijaju indeks. Sa indeksom student dolazi u biblioteku i javlja se bibliotekaru kako bi mu se otvorio karton člana. Studenti imaju pravo da koriste knjige iz bibliotečkog fonda u skladu sa pravilnikom o radu biblioteke. Ukoliko student ne vrati knjigu na vreme, ne dobija (trebalo bi, ali trenutno ne) opomenu, ali sve knjige mora vratiti prilikom overe semestra i diplomiranja. Na kraju svakog semestra, bibliotekar formira spisak studenata koji nisu razdužili sve knjige i taj spisak dostavlja studentskoj službi. Student prestaje da bude član biblioteke nakon diplomiranja.

OPŠTI POSTUPAK ZADUŽIVANJA I RAZDUŽIVANJA ZA SVE ČLANOVE:

Korisnik na osnovu evidencija o fondu (u papirnom obliku) ili uvida u stanje na policama, bira određene bibliotečke jedinice. Potpisuje se na kartonima tih jedinica, sa datumom uzimanja i ti kartoni se ulažu u karton člana. Po povratku bibliotečke jedinice, upisuje se datum vraćanja i karton se vraća u bibliotečku jedinicu i ona se odlaže na police. Ukoliko je bibliotečka jedinica koja ga interesuje zauzeta, korisnik može da rezerviše, odnosno da bude upisan u listu čekanja, pri čemu treba periodično da proverava stanje izdavanja lično (ne obaveštava se telefonom, e-mailom i

sl.). Ukoliko korisnik vrati oštećenu bibliotečku jedinicu, potrebno je u dogovoru sa dekanatom da nadoknadi štetu.

- Dokumentacija koja se koristi ili je rezultat rada aktivnosti (formulari, knjige evidencija itd.)
  - obrasci – karton knjige, karton člana
  - sagledana struktura evidencije u knjigama – spisak diplomskih radova, spisak CD izdanja, spisak časopisa
  - dodatni dokumenti – katalog dobavljača, narudžbenica, račun
  - zakoni i pravilnici – zakoni, Status Fakulteta, Pravilnici fakulteta
- Okruženje sistema i komunikacija sa sistemom

  POVEZANOST SA OKRUŽENJEM:

  INTERNO:

  a. DEKANAT – dokumenti u nabavci (narudzbenica, prijem)
  b. KADROVSKA SLUŽBA – podaci o zaposlenom
  c. STUDENTSKA SLUŽBA – podaci o studentima, zaduženja knjiga
  d. SKRIPTARNICA – nova izdanja udžbenika
  e. RAČUNOVODSTVO – uplate povodom nabavke knjiga

  EKSTERNO:

  f. DOBAVLJAČI – narudžbenice, katalozi ponude naslova
  g. MATICA SRPSKA – evidencije kod izdavanja udžbenika, katalozi
  h. IZDAVAČI – katalozi ponude naslova
  i. UNIVERZITETSKA BIBLIOTEKA – podaci o bibliotečkom fondu
- Način arhiviranja podataka u sistemu

  PAPIRNA: KARTON ČLANA, KNJIGE EVIDENCIJA i sl. Tačnije, to su PODACI O ZADUŽIVANJU I RAZDUŽIVANJU .

  ELEKTRONSKA – podaci o bibliotečkom fondu čuvaju se u bazi podataka programa BISIS, autor: Dušan Surla i saradnici
- Zakonska regulativa (zakoni i propisi po kojima se obavljaju poslovi i u kojima su definisani poslovi, aktivnosti, organizacione strukture, dokumentacija, njena struktura i postupci u korišćenju dokumentacije)
  - ZAKON O RADU BIBLIOTEKE
  - OPŠTI DRŽAVNI AKTI O BIBLOTEKARSTVU
- Lokalna pravila poslovanja (odnose na opis načina izvršavanja aktivnosti koji važe na nivou date organizacije, a nisu precizirana zakonima i drugim opštim propisima za sve organizacije tog tipa. Definisane su npr. statutom te organizacije, raznim pravilnicima ili dokumentima sistema kvaliteta ISO 9000, 14000 itd. Nekad lokalna pravila poslovanja nisu strogo formalno dokumentovana, a ipak su neophodna. Svaka organizacija ima određenu fleksibilnost u okviru opštih propisa i zakona i tu definiše svoja pravila i parametre poslovanja)
  - PRAVILNIK O RADU BIBLIOTEKE
  - PRAVILNIK O SISTEMATIZACIJI RADNIH MESTA

2. SNIMAK STANJA POSTOJEĆEG INFORMACIONOG SISTEMA

2.1. SOFTWARE

Operativni sistem: Windows XP

Ostalo: MS Office XP, BISIS (program za evidenciju bibliotečkog fonda, baza ORACLE 7)

Antivirusni programi

Internet Explorer

2.2. HARDWARE

PC računar, štampač, mrežni switch

2.3. ORGWARE

PRAVILNIK O RADU BIBLIOTEKE

Postupci: prikupljanja, obrade, prenosa, prezentovanja i arhiviranja podataka, sa tehnološkog i organizacionog aspekta.

1. prikupljanje podataka: USMENO, PAPIRNO, ELEKTRONSKI (INTERNET)
2. obrada: PAPIRNO, ELEKTRONSKI

3. prenos: PAPIRNO, ELEKTRONSKI
4. prezentovanje: PAPIRNO
5. arhiviranje: PAPIRNO, ELEKTRONSKI

2.4. LIFEWARE

BIBLIOTEKAR – sa položenim državnim stručnim ispitom za rad biblitekara, sa osnovnim poznavanjem rada na računaru

2.5.PERSPEKTIVE RAZVOJA SISTEMA

Fakultet nema dokument dugoročnog plana razvoja informacionog sistema fakulteta kao celine, ni za pokrivanje rada biblioteke. U toku su pregovori fakulteta i softverske firme u saradnji sa Univerzitetom radi preuzimanja-kupovine gotovog softvera koji pokriva rad celog fakulteta, pa samim tim i biblioteke. U toku je korišćenje softvera za rad biblioteke (koji koriste sve biblioteke Univerziteta) koji se stalno unapređuje, međutim ovaj softver vodi evidenciju fonda, bez evidencije zaduživanja i razduživanja bibliotečkih jedinica.

3. SPECIFIKACIJA ZAHTEVA KORISNIKA ZA NOVIM SISTEMOM

3.1. Problemi postojećeg načina rada postojećeg informacionog sistema

1. Korisnik nema uvid u fond biblioteke u elektronskoj formi (on-line), sa mogućnošću pretrage, već mora lično da dolazi
2. Bibliotekar i korisnik nema uvid u zaduženje korisnika osim direktnog pristupa papirnoj formi kartona člana – spora izrada izveštaja o zaduženju radi overe semestra, a izrada izveštaja o zaduženju pojedinca u svakom trenutku se ne radi opšte...
3. Ne šalju se opomene za vraćanje knjiga, a moguće je ukoliko bi bila elektronska evidencija zaduženja i prateći softver koji bi vodio računa o terminima izdavanja i vraćanja i slao elektronske opomene....
4. Zaposleni nemaju sistematičan uvid u kataloge novih izdanja od dobavljača – potrebna je on-line veza sa dobavljačima i katalozima
5. Nastavno osoblje nema uvid o stanju prodaje udžbenika, kako bi znali da pripremaju nova izdanja udžbenika
6. Razmena podataka o kadrovima sa kadrovskom službom i studentima sa studentskom službom i usalašenost baza podataka radi evidencije korisnika biblioteke – do sad je papirna evidencija, odnosno direktno indeksom studenta....
7. Nije omogućen uvid u stanje vraćanja knjige, kada je knjiga rezervisana, odnosno obaveštavanje o tome da je knjiga na raspolaganju korisnicima koji su rezervisali.

3.2. Eksplicitno izraženi zahtevi i potrebe

1. Omogućiti evidenciju izdavanja i vraćanja bibliotečkih jedinica u bazi podataka
2. Povezati bazu podataka o fondu (BISIS) sa bazom podataka izdavanja
3. Povezati bazu podataka studentske službe sa bazom podataka izdavanja bibliotečkih jedinica
4. Povezati bazu podataka kadrova iz kadrovske službe sa bazom podataka o izdavanju bibliotečkih jedinica
5. Omogućiti da korisnik putem web aplikacije u okviru fakultetskog web portala može da pregleda i pretražuje, filtrira podatke o fondu bibliotečke građe
6. Personalizovan pristup web aplikaciji biblioteke, radi uvida u stanje zaduženja knjiga za svakog korisnika, kako bi korisnik znao koje je knjige zadužio i treba da vrati.
7. Izrada izveštaja o zaduženosti pojedinog korisnika
8. Izrada izveštaja o spisku korisnika koji nisu vratili knjige do određenog datuma – za zaposlene posebno, za studente posebno
9. Prikaz upozorenja na web formi opomene za vraćanje, slanje e-maila za opomene, odnosno slanje SMS-poruke sa opomenom
10. Rezervisanje knjige putem web forme, odnosno uvid u stanje vraćanja rezervisane knjige i njene raspoloživosti
11. Povezanost sa bazom podataka skriptarnice radi uvida u prodaju udžbenika
12. Povezanost sa bazom podataka dobavljača-izdavača radi uvida u katalog ponude knjiga

3.3. Evidentiranje ciljeva i interesa rukovodstva i zaposlenih – U SKLADU SA ODGOVORNOSTIMA:

- INTERES I CILJEVI BIBLIOTEKARA: korisnici uredno vraćaju knjige i u ispravnom stanju, bibliotečki fond zadovoljava potrebe korisnika – po naslovima i broju primeraka
- INTERES I CILJEVI KORISNIKA: bibliotečke jedinice na raspolaganju, naslovi u skladu sa savremenim zahtevima nauke i tehnologije i u skladu sa zahtevima nastavnih sadržaja i zahteva.
- INTERES I CILJEVI DEKANATA: zadovoljni korisnici biblioteke, troškovi nabavke bibliotečke građe u okviru godišnjeg budžeta za biblioteku, bibliotečka građa vraćena i u ispravnom stanju

3.4. Evidentiranje objektivno utvrđenih potreba
- UVID U STANJE BIBLIOTEČKOG FONDA – NASLOVI, ZAUZETOST, REZERVISANOST
- UVID U STANJE KORIŠĆENJA – IZVEŠTAJ O IZDAVANJU, STATISTIKA PO KORISNICIMA I NASLOVIMA
- UVID U ZADOVOLJSTVO KORISNIKA- UPITNIK, STATISTIKA, PREDLOZI ZA NABAVKU
- UVID U TROŠKOVE NABAVKE - STATISTIKA

3.5. Evidentiranje odluka i informacione podrške odlukama
- BIBLIOTEKAR: naslovi za naručivanje-budžet- prioriteti, opomene, prioritet kod rezervisanja
- KORISNIK: naslovi za iznajmljivanje, vraćanje, rezervisanje
- DEKANAT: budžet za nabavku knjiga

3.6. Evidentiranje potrebnih automatizama u softverskom rešenju
1. Provera datuma vraćanja i generisanje opomene
2. Datum vraćanja >= datum izdavanja
3. godišnji budžet za nabavku > iznos na narudžbenici
4. preuzimanje podataka o korisnicima iz kadrovske i studentske službe
5. prilikom vraćanja provera da li je knjiga rezervisana, pa ako jeste, slanje obaveštenja korisniku koji je prvi na listi za rezervisanje
6. vraćanje knjiga automatski osvežava prikaz na webu povodom raspoloživosti naslova i zaduženja korisnika
7. kreiranje izveštaja o zaduženim korisnicima određenog datuma
8. broj knjiga ne može biti veći od 3 odnosno 5
9. Korisnik ne može da zadužuje nove naslove, ako prethodne nije vratio

4. Evidentiranje zakonski izraženih potreba i ograničenja

Na osnovu pravilnika o radu biblioteke fakulteta:
1. Student ne može iznajmiti više od 3 knjige istovremeno
2. Zaposleni ne može iznajmiti više od 5 knjiga istovremeno
3. Student ne može zadržati knjigu duže od 2 meseca
4. Zaposleni ne može zadržati knjigu duže od 6 meseci
5. Opomena se šalje 15 dana nakon isteka roka vraćanja

3.7. Odnos prema organizaciji rada

Uvođenje softverske podrške za deo aktivnosti koji je bio pokriven usmeno ili papirno unapređuje kvalitet rada u smislu brzine i preciznosti, efikasnosti. Postojeća organizacija rada u smislu redosleda procedura i odgovornosti-ovlašćenja u ovom slučaju neće biti izmenjena.

Izmene u načinu organizacije rada bi nastale ukoliko bi se obezbedila softverska podrška koja to omogućuje:
- Ukoliko bi se izvršila integracija sa kadrovskom službom i studentskom službom, tada proces učlanjivanja ne bi morao da postoji, dakle ne mora korisnik lično da se pojavi u biblioteci, već bi se evidentiranjem studenta ili zaposlenog automatski izvršilo i otvaranje kartona korisnika biblioteke.
- Ukoliko se obezbedi softverski modul za rezervisanje naslova, tada se ni ovaj deo ne bi morao izvršavati lično od strane korisnika, već putem web forme.

3.8. Utvrđivanje grupa zahteva i prioriteta u razvoju

Moduli su grupisani prema funkcionalnim i tehnoloskim karakteristikama, kao i prioritetima. Redosled 1-9 jeste i redosled implementacije pojedinih delova-modula po prioritetu, tako da broj 1 nosi modul sa najvecim prioritetom razvoja.

OBAVEZNI DEO («NEED TO HAVE»)

1. DEO (desktop aplikacija)– EVIDENCIJA KORISNIKA, ZADUŽIVANJA I RAZDUŽIVANJA, REZERVACIJE I OPOMENE, STATISTIKA KORIŠĆENJA NASLOVA, IZVEŠTAJI O ZADUŽIVANJU
2. DEO (modul veza) – POVEZANOST SA BAZOM PODATAKA FONDA BIBLIOTEKE PROGRAMA BISIS
3. DEO (web aplikacija) – UVID U FOND, ZADUŽIVANJA, OPOMENE I REZERVACIJE

OPCIONI DEO («NICE TO HAVE»)

4. DEO (SMS, e-MAIL) – slanje obaveštenja korisnicima putem SMS i e-mail poruka
5. DEO (modul veza) – POVEZANOST SA BAZOM PODATAKA STUDENATA
6. DEO (modul veza) – POVEZANOST SA BAZOM PODATAKA ZAPOSLENIH
7. DEO (web aplikacija) - UPITNIK O ZADOVOLJSTVU uslugama biblioteke, PREDLOZI POBOLJŠANJA RADA I NABAVKE NOVIH NASLOVA

NEPOTREBNI DEO («NEED NOT TO HAVE»)

8. DEO (modul) – POVEZANOST SA SKRIPTARNICOM – podaci o prodaji udžbenika i statistika stanja, radi uvida zaposlenima
9. DEO (modul) – POVEZANOST SA BAZOM PODATAKA IZDAVAČA-DOBAVLJAČA radi sistematičnog prikaza KATALOGA

3.9. SPECIFIKACIJA ZAHTEVA KORISNIKA (client requirements document)

- SVRHA SOFTVERA
  - Evidentiranje korisnika, zaduživanja, razduživanja, rezervacija i opomena korisnicima biblioteke, uz prateće izveštaje i statistiku
  - Pretraga fonda od strane korisnika, kao i personalizovani pregled zaduženja, stanja rezervisanih naslova i opomena
- DEFINICIJE POJMOVA I SKRAĆENICE
  - Bibliotečka jedinica – knjiga, završni rad, CD, udžbenik
  - Korisnik – zaposleni fakulteta ili student
- POSLOVNI CILJEVI (BUSSINESS OBJECTIVES)
  - Pokrivanje poslovnih aktivnosti - obrade korisnika, zaduženja i razduživanja, uz prateći prikaz fonda biblioteke
  - Poslovni ciljevi – bolja informisanost korisnika i veće zadovoljstvo uslugama od strane korisnika, brža obrada korisnika i zaduženja od strane bibliotekara
- TEHNOLOGIJA PO MODULIMA
  - Desktop aplikacija - Evidentiranje korisnika, zaduživanja, razduživanja, rezervacija i opomena korisnicima biblioteke, uz prateće izveštaje i statistiku
  - Web aplikacija - Pretraga fonda od strane korisnika, kao i personalizovani pregled zaduženja, stanja rezervisanih naslova i opomena
  - Baza podataka:
    - Fond – koristi se baza podataka programa BISIS prilikom evidentiranja zaduživanja i prilikom prikaza fonda na web-u
    - KorisniciZaduzenja – posebna baza podataka novog programa, koja se puni iz desktop aplikacije, a čita iz web aplikacije radi prikaza
- POSLOVNA PRAVILA (RULES) KOJA TREBA AUTOMATSKI DA SE IZVRSAVAJU - OGRANICENJA.
  - Ograničenja:
  - Datum vraćanja ne može biti u periodu duži od X odnosno Y meseci
  - Datum vraćanja >= datum izdavanja
  - prilikom vraćanja provera da li je knjiga rezervisana, pa ako jeste, slanje obaveštenja korisniku koji je prvi na listi za rezervisanje
  - broj knjiga ne može biti veći od Xodnosno Y
  - Korisnik ne može da zadužuje nove naslove, ako prethodne nije vratio
- POTREBNI IZVESTAJI (REPORTS) - EKRANSKI I PAPIRNI, DO NIVOA ELEMENTARNOG PODATKA

  EKRANSKI:
  - Spisak korisnika
  - Spisak bibliotečkih jedinica
  - Spisak slobodnih naslova
  - Spisak rezervisanih naslova

        - Spisak redosleda rezervacije za naslov:_____
        - Spisak naslova autora: _____
        - Spisak naslova sa nazivom dela: _____
        - Spisak korisnika za opomenu na dan _____

    PAPIRNI:
        - Spisak studenata koji su zaduženi do datuma: _____
        - Spisak nastavnika koji su zaduženi do datuma: _____
        - Spisak zaduženih bibliotečkih jedinica za korisnika: _____
        - Statistika zaduživanja po bibliotečkim jedinicama
        - Opomena za korisnika: _____
- POTREBNI UPITI
    - prikaz zaduzenih korisnika
    - prikaz slobodnih korisnika
    - prikaz zaduzenih biblioteckih jedinica
    - prikaz slobodnih biblioteckih jedinica
    - prikaz svih korisnika
    - prikaz korisnika sa prezimenom:_____
    - prikaz naslova sa nazivom dela: _____
    - prikaz naslova autora: _____
    - prikaz rezervisanih naslova
    - prikaz redosleda rezervacija za naslov: _____
    - prikaz zaduzenih biblioteckih jedinica za korisnika: _____
    - prikaz zaduzivanja bibliotečkih jedinica grupisano po naslovima (statistika)
- ZAHTEVI ZA OSOBINAMA KORISNIČKOG INTERFEJSA
    - DESKTOP APLIKACIJA – svetla boja pozadine, uobičajen izgled windows aplikacija (meni sa sličicama i padajući meni), za svaku funkciju 3 ekranske kartice (ažuriranje, pretraga, štampa)
    - WEB APLIKACIJA – deo web portala fakulteta, svetla boja pozadine, meni levo, desno radni deo za svaku opciju, javni deo i personalizovani deo za korisnike – prijava korisničko ime i šifra
- BEZBEDONOSNI ZAHTEVI

  ZAŠTITA OD NEOVLAŠĆENOG PRISTUPA:
    - DESKTOP APLIKACIJA – koristi samo bibliotekar, pristupa putem korisničkog imena i šifre, (oba su kriptovana u bazi podataka). Prilikom učlanjivanja korisnik dobija korisničko ime i šifru za pristup web portalu.
    - WEB APLIKACIJA – pristupaju korisnici putem korisničkog imena i šifre koji dobijaju prilikom učlanjivanja u biblioteku. Kasnije izmene šifre su moguće u svakom trenutku.

  BACKUP: Vrši se backup obe baze podataka. To radi automatski softver svako veće u 22 časa...

  REPLIKACIJA: Baza podataka koja se koristi u biblioteci nalazi se na računaru u biblioteci. Baza podataka koja je vidljiva na webu nalazi se na web serveru u računskom centru fakulteta. Svako veče sa početkom u 24 časa vrši se prebacivanje podataka iz jedne u drugu bazu podataka. Tačnije, 2 baze podataka se »presipaju« u druge 2 baze podataka. Naravno, ne presipa se sve, već samo novi podaci koji su uneti tog dana.
- OPIS RADA SISTEMA (OVERAL DESCRIPTION) - HRONOLOŠKI OPIS KAKO ĆE RADITI CEO SISTEM SA UKLJUČENIM NOVIM SOFTVERSKO/HARDVERSKIM REŠENJIMA:
    - Desktop aplikacija:

      Korisnik dolazi u biblioteku da se učlani (ovo je zadržano, dok se ne uradi modul automatskog preuzimanja studenata i zaposlenih, kada ova aktivnost neće biti potrebna). Bibliotekar se prijavljuje na desktop aplikaciju. Otvara formu za dodavanje novih korisnika. Ako je student, uzima indeks i iz indeksa unosi podatke, a ako je zaposleni, uzima ličnu kartu. U toku dodavanja novog korisnika, unosi i korisničko ime i šifru za pristup web aplikaciji i saopštava korisniku. Korisnik na licu mesta pristupa računaru bibliotekara ili nekom dodatnom računaru u biblioteci i pretražuje naslove po nazivu dela ili autoru, na spisku

raspoloživih naslova. Korisnik bira naslov. Bibliotekar otvara formu za zaduživanje korisnika i unosi podatke o zaduživanju. Softver proverava broj naslova i da li je korisnik prethodno vratio ranije zadužene naslove. Ako nije, softver zabranjuje izdavanje knjiga. Ako je knjiga na raspolaganju i ako korisnik ima pravo da uzme novi naslov, bibliotekar donosi potrebnu bibliotečku jedinicu, vadi karton i upisuje datum izdavanja, a korisnik se potpisuje (zadržan je i papirni deo i dalje). Bibliotekar unosi podatke o zaduživanju.

Ako željena knjiga nije na spisku raspoloživih, korisnik može da rezerviše knjigu. Bibliotekar unosi podatke o rezervaciji knjige za korisnika na ekranskoj formi za rezervisanje. Time se korisnik stavlja na listu čekanja za dati naslov.

Kada korisnik vraća bibliotečku jedinicu, donosi je, a bibliotekar unosi podatke o razduživanju na ekranu za razduživanje.Biblotekar vadi karton knjige iz kartona korisnika i upisuje datum vraćanja.

Softver svakog dana automatski proverava za sve korisnike i sva zaduženja da li su vraćena i ako nisu, da li je prošao rok vraćanja. Ako jeste, generiše poruku o opomeni.

Bibliotekar za željeni datum, periodično ili po potrebi, može da odštampa spisak zaduženih

studenata, što je potrebno radi uvida u stanje zaduženosti kod overe odnosno upisa semestra.

Bibliotekar po potrebi može da odštampa i statistiku upotrebe pojedinih naslova kako bi se donela odluka o nabavci novih primeraka istih naslova, rashodovanju dotrajalih primeraka i slično.

- Web aplikacija :

  Pristupom web portalu fakulteta, prikazuje se link ka posebnom portalu za rad biblioteke. Pristupom toj opciji, učitava se meni web aplikacije. Korisnik može da bira pretragu fonda, odnosno slobodnih i rezervisanih naslova ili uvid u stanje sopstvenih zaduženja i rezervacija.

  Prva opcija nudi stavke: pretraga naslova po nazivu-autoru, prikaz slobodnih naslova, prikaz rezervisanih naslova.

  Druga opcija otvara dijalog za prijavu korisnika. Nakon prijave sa korisničkim imenom i šifrom, web aplikacija nudi stavke za prikaz korisnikovih: zaduženja, rezervacija i opomena.

- DETALJNI OPIS RADA SA SOFTVERSKIM SISTEMOM - KOJI SE EKRAN KADA OTVARA I KOJA OPCIJA KORISTI ZA IZVRŠAVANJE POSLOVA. ŠTA TREBA OD FUNKCIJA DA SADRŽI KOJI EKRAN. OPŠTE OSOBINE (SOFTVERSKOG) SISTEMA - FEATURES I DETALJNIJE ODREDJENJE – REQUIREMENTS – potrebno je:
  - *NACRTATI IZGLED DIZAJNA SVIH EKRANA SA OPCIJAMA KOJE POSTOJE NA EKRANIMA*
  - *NACRTATI DIJAGRAM AKTIVNOSTI KOJIM SE ILUSTRUJE TOK IZVRŠAVANJA PROGRAMA I KORIŠĆENJA POJEDINIH OPCIJA*
- BUDUCE OSOBINE SOFTVERA - NECE BITI UKLJUCENE U OVOJ VERZIJI, A MOGU DA SE KASNIJE DODAJU KAO UNAPREDJENJE, NOVI MODUL I SL. (STATISTIKA, RAZMENA PODATAKA I SL.)
  - DEO (web aplikacija) – unos rezervacija knjiga od strane korisnika
  - DEO (SMS, e-MAIL) – slanje obaveštenja korisnicima putem SMS i e-mail poruka
  - DEO (modul veza) – POVEZANOST SA BAZOM PODATAKA STUDENATA
  - DEO (modul veza) – POVEZANOST SA BAZOM PODATAKA ZAPOSLENIH
  - DEO (web aplikacija) - UPITNIK O ZADOVOLJSTVU uslugama biblioteke, PREDLOZI POBOLJŠANJA RADA I NABAVKE NOVIH NASLOVA

4. ELEMENTI UPRAVLJANJA PROJEKTOM

- definisanje ciljeva projekta
  - Pokrivanje poslovnih aktivnosti - obrade korisnika, zaduženja i razduživanja, uz prateći prikaz fonda biblioteke
  - Poslovni ciljevi – bolja informisanost korisnika i veće zadovoljstvo uslugama od strane korisnika, brža obrada korisnika i zaduženja od strane bibliotekara
  - Ciljevi projekta – razvoj desktop i web aplikacije koji omogućavaju uvid u fond i obradu i uvid zaduženja korisnika

- definisanje obuhvata posla – celine, podsistema i prioriteti razvoja
  - postoje 3 celine, koje bi se razvile ovim redosledom prioriteta:
    1. desktop aplikacija
    2. modul za integraciju sa bazom podataka programa BISIS
    3. web aplikacija
- definisanje rezultata projekta – opis i karakteristike kvaliteta
  1. RAČUNAR U BIBLIOTECI – BISIS, DESKTOP PROGRAM (dot net), BAZA KORISNICI-ZADUŽENJA (MS SQL baza)
  2. RAČUNAR KORISNIKA (INTERNET) – INTERNET EXPLORER
  3. WEB SERVER FAKULTETA – WEB PORTAL FAKULTETA, WEB APLIKACIJA ZA RAD BIBLIOTEKE (ASPX), REPLIKACIJA BAZE FONDA I KORISNIKA-ZADUŽENJA
- definisanje ograničenja i pretpostavki razvoja

  TEHNOLOŠKE:
  - PRETPOSTAVKE: uklapanje sa ORACLE bazom BISIS, uklapanje sa WEB portalom fakulteta (ASPX, LINUX...)
  - OGRANIČENJA: -

  ORGANIZACIONE I LIFEWARE:
  - PRETPOSTAVKE: programer ima pravo da pristupa bazi podataka programa BISIS
  - OGRANIČENJA: angažovanje eksternog programera koji poznaje tehnološku osnovu, s obzirom da sistem administrator, kao ni asistenti na fakultetu nisu zaduženi, nemaju vremena za ovaj deo (programiranje), a ni dovoljno znanja-iskustva za rad sa ORACLE bazom podataka.
- studija opravdanosti (procena troškova i dobiti)
  - Troškovi: plaćanje programera za izradu softvera
  - Dobiti: nisu finansijske prirode (osim manje uštede novca zbog ozbiljnije evidencije vraćanja knjiga, šime se sprečavaju zloupotrebe-ne vraćanja knjiga... ali to je svakako u malom procentu). Znači, nema značajnog povraćaja novca, već je dobit isključivo kvalitativna: POVEĆANJE STEPENA INFORMISANOSTI I ZADOVOLJSTVA KORISNIKA BIBLIOTEKE, A SAMIM TIM I POVEĆANJE ZNANJA STUDENATA KROZ BOLJU EDUKACIJU....
- plan razvoja po izvršiocima, fazama aktivnosti, podsistemima i troškovima

  AKTIVNOSTI
  1. SISTEMSKA ANALIZA
  2. DIZAJN SOFTVERA I BAZE PODATAKA
  3. PROGRAMIRANJE MODULA ZA KORIŠĆENJE PODATAKA IZ ORACLE BAZE BISIS-A
  4. PROGRAMIRANJE DESKTOP APLIKACIJE
  5. PROGRAMIRANJE WEB APLIKACIJE
  6. TESTIRANJE MODULA
  7. INTEGRACIJA MODULA
  8. TESTIRANJE INTEGRISANOG SOFTVERSKOG SISTEMA
  9. POSTAVLJANJE APLIKACIJE U REALNOM OKRUŽENJU
  10. TESTIRANJE RADA U REALNOM OKRUŽENJU
  11. DOKUMENTOVANJE – KORISNICKO UPUTSTVO I UPUTSTVO ZA ODRŽAVANJE
  12. OBUKA KORISNIKA

TRAJANJE PROJEKTA: 8 MESECI
PLANIRANI POČETAK: nakon potpisivanja ugovora i prihvatanja idejnog projekta
KONTROLNE TAČKE: nakon završene svake faze, overa uspešnosti faze od strane korisnika
IZVRŠIOCI AKTIVNOSTI:

FAZA	UČESNIK (iz programerske firme)
1,2	– Marija E.
6,8,10	– Ivana B.
11,12	– Vesna C.
3,4	– Bojan A.
5,7	– Igor F.

REZULTATI i TROŠKOVI:

FAZA	REZULTAT	TROŠKOVI (VREME)
SISTEMSKA ANALIZA	MODEL POSLOVNOG DOMENA, DIJAGRAM TOKA PODATAKA, MODEL PODATAKA	1 NEDELJA
DIZAJN SOFTVERA I BAZE PODATAKA	BAZA PODATAKA UML DIJAGRAMI – SPECIFIKACIJA DIZAJNA SOFTVERA	1 NEDELJA
PROGRAMIRANJE MODULA ZA KORIŠĆENJE PODATAKA IZ ORACLE BAZE BISIS-A	MODUL ZA PRISTUP I ČITANJE PODATAKA IZ ORACLE BAZE, TJ. DLL (KLASA)	1 NEDELJA
PROGRAMIRANJE DESKTOP APLIKACIJE	DESKTOP PROGRAM SA IZVEŠTAJIMA I STATISTIKOM (EXE, RPT)	2 MESECA
PROGRAMIRANJE WEB APLIKACIJE	WEB APLIKACIJA (ASPX)	2 MESECA
TESTIRANJE MODULA	IZVEŠTAJ O TESTIRANJU	1 NEDELJA
INTEGRACIJA MODULA	IZVEŠTAJ O INTEGRACIJI – PROJEKTNA TEHNIČKA DOKUMENTACIJA	1 NEDELJA
TESTIRANJE INTEGRISANOG SOFTVERSKOG SISTEMA	IZVEŠTAJ O TESTIRANJU	1 NEDELJA
POSTAVLJANJE APLIKACIJE U REALNOM OKRUŽENJU	IZVEŠTAJ O TESTIRANJU	1 NEDELJA
TESTIRANJE RADA U REALNOM OKRUŽENJU	IZVEŠTAJ O TESTIRANJU	1 NEDELJA
DOKUMENTOVANJE – KORISNICKO UPUTSTVO I UPUTSTVO ZA ODRŽAVANJE	IZVEŠTAJ O TESTIRANJU	1 NEDELJA
OBUKA KORISNIKA	IZVEŠTAJ O TESTIRANJU	1 NEDELJA

UKUPNO MINIMALNO TRAJANJE PROJEKTA: APPROX. 6 MESECI
(JOŠ 2 MESECA DODATO RADI OMOGUĆAVANJA EVENTUALNIH ZASTOJA DA SE OTKLONE)
UKUPNO: 8 MESECI

- studija izvodljivosti (SWOT analiza i druge)

  Angažovanje eksternog programera odnosno programerske firme dovodi projekat u pitanje, s obzirom da fakultet ne može objektivno utvrditi sposobnost firme da izvrši planirane aktivnosti u datom roku, niti da li firma raspolaže dovoljnim brojem ljudi, računara, i potrebnim znanjem da se ovaj posao izvrši. SWOT analiza je namenjena samoj softverskoj firmi – da ona proceni da li može da izvede posao do kraja, dakle služi za preispitivanje sopstvenih mogućnosti.
  ILUSTROVAĆEMO ANALIZU SOFTVERSKE FIRME KOJA TREBA DA URADI POSAO:
  SWOT analiza – Strength, Weakness, Opportunity, Threat
  Strength – dovoljan broj ljudi i računara, poznavanje rada sa ORACLE bazom, poznavanje izrade web aplikacija
  Weakness – nepoznavanje programa BISIS, nepoznavanje strukture web portala fakulteta
  Opportunity – razvoj opštih modula za rad sa oracle bazom, iskustvo sa radom sa programom BISIS, reference za firmu i radnike

Threat – slaba saradnja sa sistem administratorom (česti izostanci sa radnog mesta, drugi angažmani u toku radnog vremena), nejasno ko je odgovoran od strane fakulteta za komunikaciju i podršku realizaciji projekta
ZAKLJUČAK: SOFTVERSKA FIRMA PRIHVATA POSAO, JER SMATRA DA JE IZVODLJIV, SAMO AKO SE ODREDE 2 OSOBE KOJE SU ZADUŽENE ZA POTPUNU SARADNJU SA PROGRAMERIMA, A KOJE SU TEHNIČKI KOMPETENTNE (sistem administrator i web administrator)

- određivanje kriterijuma kvaliteta i zadovoljstva korisnika
    1. ispunjavanje potrebnih karakteristika softvera (funkcija, ekrana, izveštaja, upita...)
    2. završavanje projekta u predviđenom terminu
    3. korektno funkcionisanje softvera, bez grešaka
    4. brz rad softvera, bez zastoja
    5. dokumentacija-uputstva o korišćenju
    6. podrška kasnijem održavanju softvera - biti na raspolaganju za buduće izmene

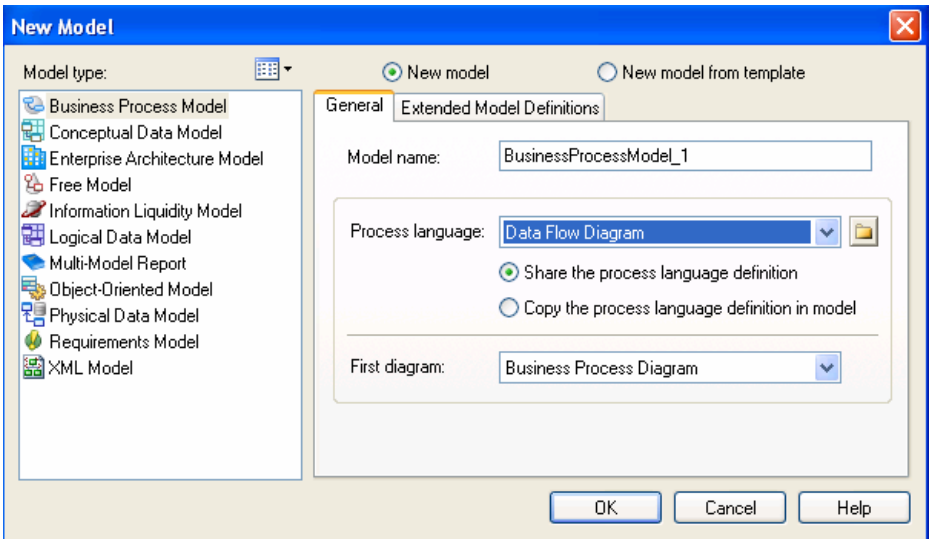
    Ovi zahtevi treba da su uključeni u tekst ugovora, kao i da je predviđeno koji korisnici softvera ili zaposleni koji su zaduženi za projekat treba da verifikuju uspešnost projekta svojim potpisom.
- definisanje rizika projekta i tehnika upravljanja rizikom

    POTENCIJALNI PROBLEMI-RIZICI ovog projekta:
    1. izmene programa BISIS i strukture baze podataka koja ga prati u toku izrade softvera koji se na ovu bazu podataka nadovezuje
    2. Odsustvo saradnje osoblja sa fakulteta povodom omogućavanja pristupa i integracije web serveru
    3. Odsustvo osoblja softverske firme ili angažovanje na drugim projektima i samim time nedovoljna posvećenost ove firme završetku projekta

    REŠAVANJE PROBLEMA KOJI DOVODE PROJEKAT U RIZIK:
    1. adaptibilno softversko rešenje koje bi se brzo prilagodilo svakoj strukturi baze podataka
    2. uključivanje opisa odgovornosti osoblja fakulteta u projektu u tekst ugovora, dokumentacije kojom se evidentiraju aktivnosti saradnje i predviđeno izuzimanje odgovorsnosti softverske firme za uspeh projekta, ako saradnja ne bude adekvatna
    3. Interno ugovaranje posla zaposlenih u softverskoj kući i dobijanje novca po završetku projekta, angažovanje vezano za projekat, a ne generalno zapošljavanje programera....

4.MODELOVANJE POSLOVNIH PROCESA

U okviru CASE alata Power Designer 15, nakon izbora opcije New Model dobijamo dijalog prozor za izbor tipa modela (Slika 3.2.)



Slika 3.2. Dijalog prozor za izbor tipa modela

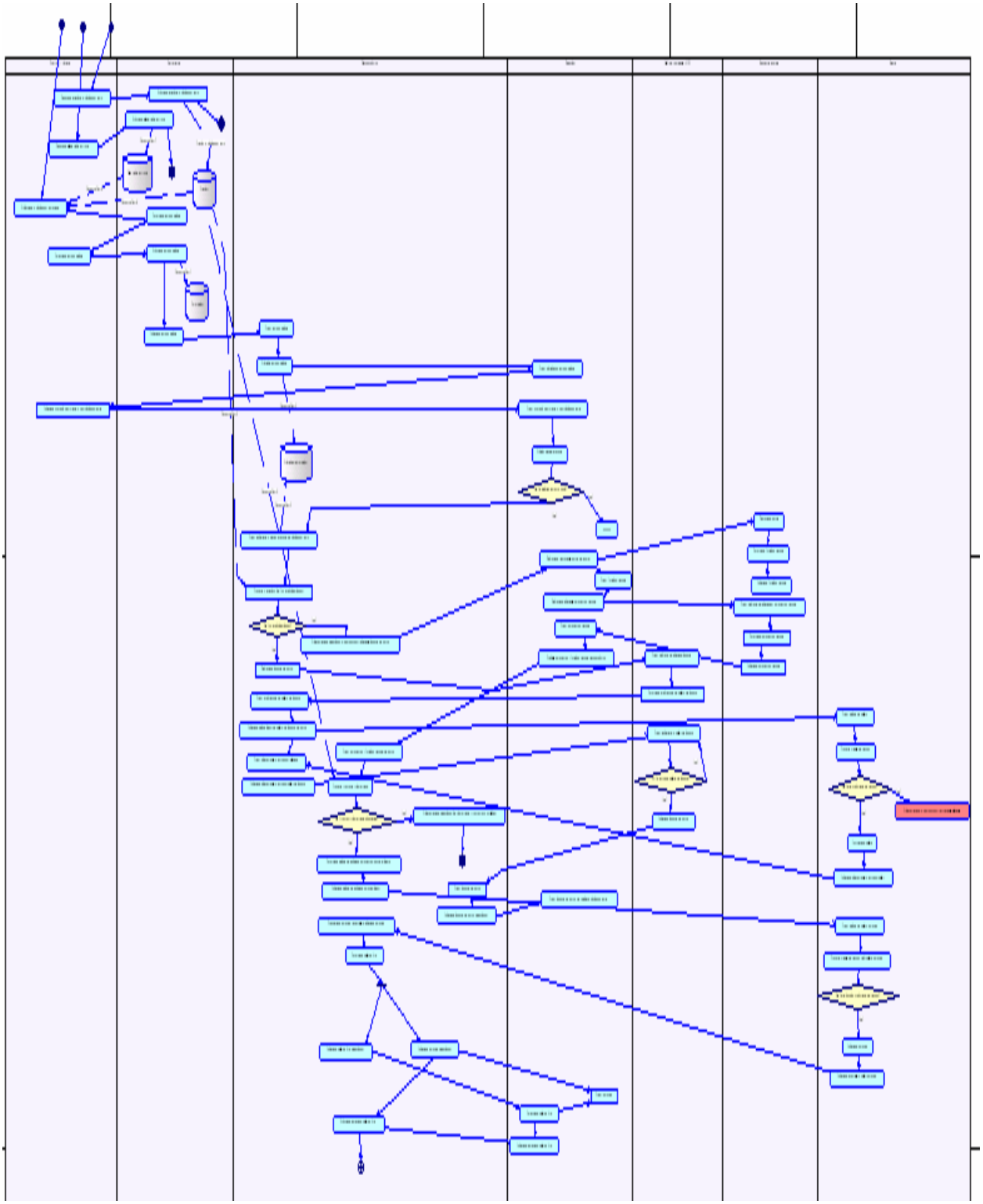
Za izradu modela poslovnih procesa koristimo „Business Process model" grupu modela sa leve strane dijalog prozora, kao first dijagram koristimo „Business Process Diagram", a kao process language:

1. Za dijagram toka poslovnih procesa (slično „algoritamskom" crtanju toka poslovnih procesa) koristimo "Analysis"
2. Za dijagram toka podataka (hijerarhijsko predstavljanje poslovnih procesa, postepenim uvođenjem detalja) koristimo „Data Flow dijagram")

4.1. Modelovanje toka poslovnih procesa

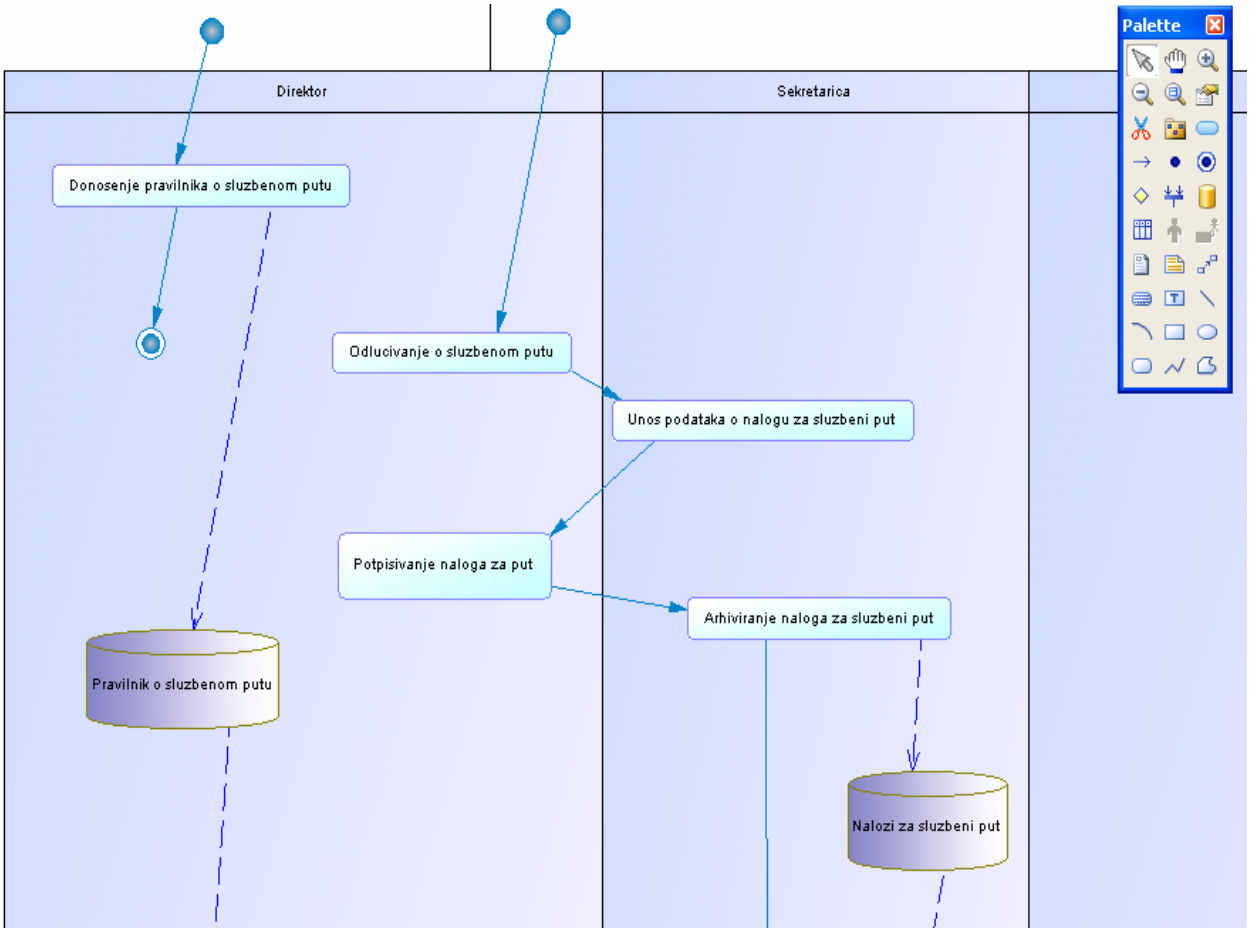
ZADATAK: Na osnovu datog opisa posla izraditi u CASE alatu dijagram toka poslovnih procesa, tj. Business process model.

REŠENJE: Na Slici 3.2.1. predstavljen je deo urađanog modela. Za svako radno mesto određena je po jedna «plivačka linija», odnosno «organizaciona jedinica», a u okviru nje se postavljaju odgovarajuće aktivnosti (ovali) pri čemu vodimo računa o načinu izražavanja (najčešće koristimo glagolske imenice – npr. definisanje pravilnika, unos podataka o putnom nalogu) i preciznosti – potpunosti izražavanja. Posebno predstavljamo skladišta podataka (simbol valjka), tokove procesa – puna linija sa strelicom, tokove podataka – isprekidana linija sa strelicom, sinhronizacije i uslovna grananja (za svako uslovno grananje mora biti barem 2 jasno definisane alternative – sa vrednostima kao za pitanja – da/ne ili sa visestrukim konkretnim vrednostima). Crtanje dijagrama toka procesa slično je crtanju algoritama, s tim da može biti više paralelnih tokova poslovnih procesa; znači više simbola za početak i završetak.

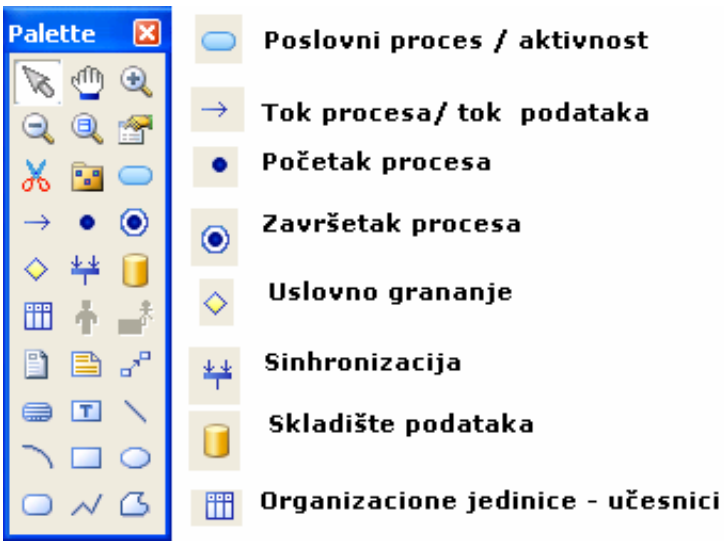


Slika 3.2.1.1. Primer urađenog modela toka procesa

Napomena: U tekstu nisu dati svi detalji kako bi se dijagram kompletno mogao završiti (nedostaju neka od grana uslovnih grananja – obeleženo „crvenim ovalom"), ali se zato u interakciji sa krajnjim korisnikom na osnovu uradjenog dijagrama i naznaka o potrebama za dodatnim informacijama vrši dopuna teksta i korigovanje-kompletiranje dijagrama. Takođe, tekst nije nedvosmislen, precizan i hronološki uređen. Kreiranje dijagrama na osnovu teksta dovodi do reda i preciznosti, potrebnih za dalje faze razvoja.



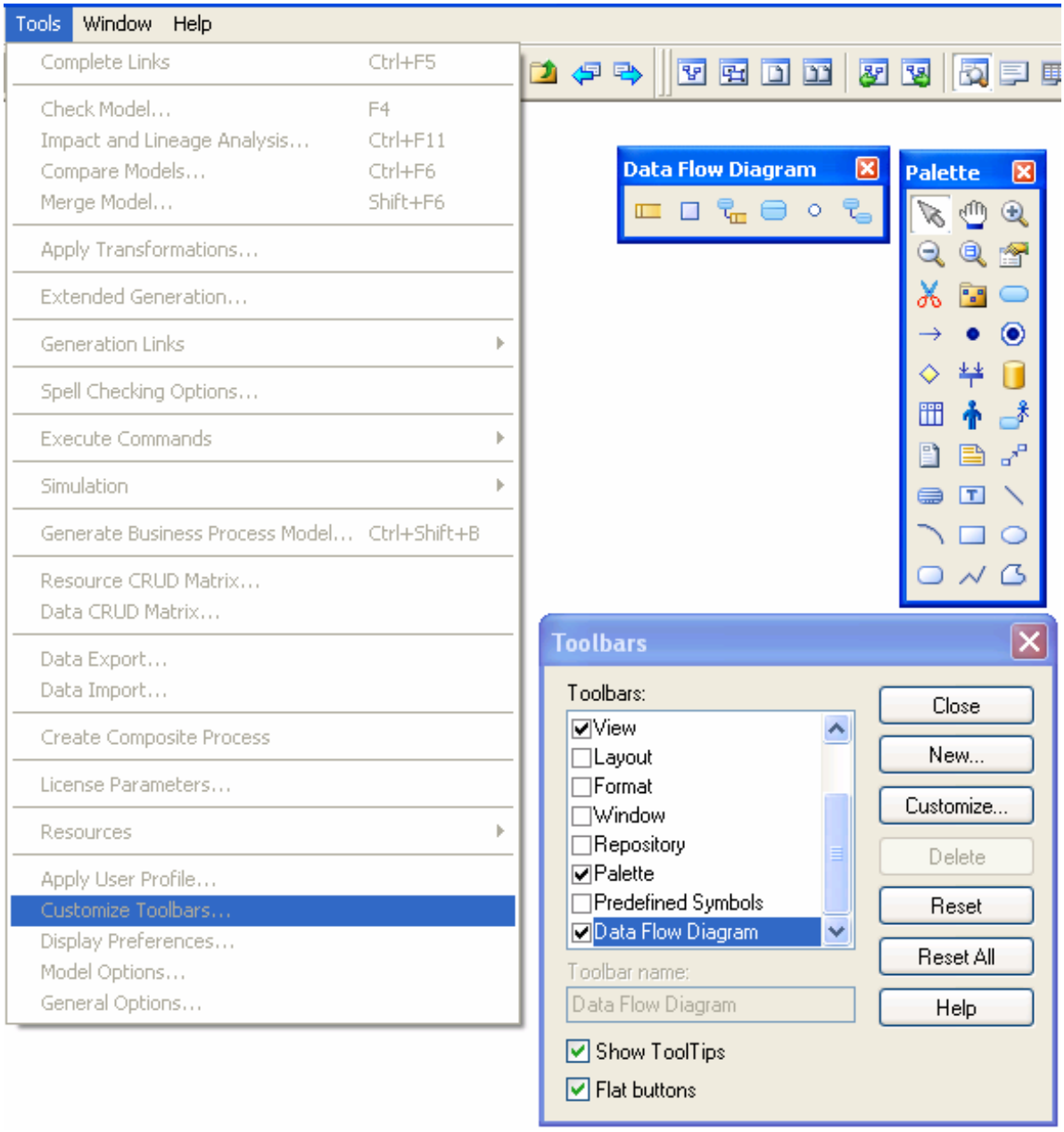
Slika 3.2.1.2. Detaljniji prikaz dijagrama iz CASE alata



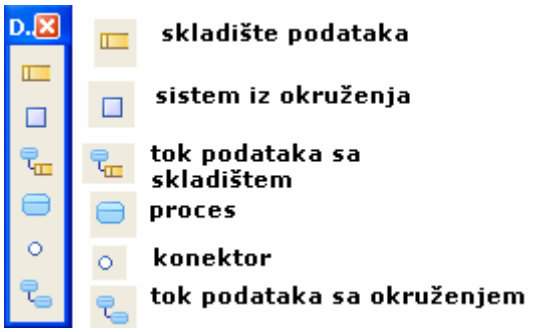
Slika 3.2.1.3. Prikaz palete za rad sa elementima dijagrama i objašnjenje pojedinih alata

4.2. Dijagram toka podataka

Dijagram toka podataka nastaje kao rezultat primene metoda Strukturne sistem analize, čija je suština u hijerarhijskoj uređenosti procesa u formi stabla i postepenim uvođenjem detalja za pojedine segmente poslovnih procesa. Prednost ove metode u odnosu na prethodnu metodu je u fokusiranju na posmatrani sistem (dok se u prethodnoj metodi predstavljaju svi učesnici iz opisa posla, nezavisno od fokusa. Iako algoritamski način crtanja dijagrama toka poslovniih procesa ima prednost u kompletnoj preglednosti, ipak za složenije sisteme ta preglednost se značajno gubi. Uvođenjem podele na podprocese i postepenim uvođenjem detalja dobija se na organizovanosti modela i boljoj preglednosti.



Slika 4.2.1. Prikazivanje palete za rad sa Data flow dijagramom u Power Designer 15



Slika 4.2.2. Paleta za rad sa data flow dijagramom

NAPOMENE:

- *DEKOMPONOVATI ZNAČI PODELITI BAREM NA 2 PODPROCESA.*

0. *NIVO DEKOMPOZICIJE – samo jedan proces: naziv sistema (npr. Umetnicka radionica) ili naziv problema, tj. Jasno definisane grupe procesa (obrada troskova sluzbenog puta), sistemi iz okruzenja i tokovi podataka*
1. *NIVO DEKOMPOZICIJE – osnovne grupe procesa, interni tokovi i skladišta podataka, sistemi iz okruženja i eksterni tokovi podataka. PRAVILO BALANSA (skup tokova podataka koji je ušao u neki sistem ili proces ili skup podataka koji izlazi iz datog sistema ili procesa ne sme biti narušen na dijagramima dekompozicije tako da ima više ili manje tokova nego na prethodnom dijagramu. Ovo pravilo se odnosi na eksterne tokove, dok internih tokova moze biti više - proizvoljno). KOMUNIKACIJA IZMEDJU PROCESA ISKLJUČIVO POSREDSTVOM SKLADIŠTA, NE DIREKTNO.*
   Osnovne vrste podela na 1. nivou dekompozicije:
   1. Osnovne, Pomocne, Upravljačke
   2. Pripremne aktivnosti, Izvršne aktivnosti, Završne aktivnosti
2. *NIVO DEKOMPOZICIJE – dekompozicija pojedinačnog procesa na njegove podprocese, uz prikaz podprocesa, internih skladišta podataka i internih tokova, uz prikaz i eksternih tokova i sistema iz okruženja koji se odnose na taj proces*
3. *... DALJE SVE DO NIVOA PRIMITIVNIH PROCESA, TJ. PROCESA KOJI SE DALJE NE DELE. Kako odrediti primitivni proces: Ako njegovom dekompozicijom dobijemo procese ili aktivnosti koje ne mogu nezavisno da se pokreću, ne mogu biti samostalni i da između njih ne moze postojati vremenski prekid, tada je proces koji oni čine primitivni proces. Npr. Evidentiranje ispitne prijave je primitivni proces je se ne može se podeliti na evidentiranje broja indeksa, evidentiranje predmeta itd. – ne mogu se nezavisno pokretati. Ne moraju svi procesi biti dekomponovani do istog nivoa.*

- *Stablo procesa se može formirati koristeći heurističke pristupe:*
  1. *Kod većih sistema, na 1. nivou podela na osnovnu, pomoćnu i upravljačku delatnost, a u okviru njih podela na jasno definisane celine u skladu sa hronologijom poslovnih procesa.*
  2. *Kod manjih grupa ili ako posmatramo samo osnovnu delatnost, podela već na prvom nivou na pripremnu, izvršnu i završnu aktivnost, a unutar njih podela na jasno definisane celine u skladu sa hronologijom poslovnih procesa.*
  3. *Jasno definisane grupe poslovnih procesa koje se odvijaju prema hronologiji.*
- *Oznake stavki stabla procesa moraju biti brojčane i označavati se redom prema hronologiji izvršavanja procesa.*
- *Potrebno je nazivati stavke stabla kao aktivnosti, odnosno koristiti glagolske imenice.*

ZADATAK: Za navedeni proces obrade troškova službenog puta uraditi stablo procesa.

REŠENJE:
Jedno rešenje stabla procesa dato je u nastavku:

OBRADA TROŠKOVA SLUZBENOG PUTA

1. PRIJEM PODATAKA O PUTOVANJU
   - 1.1. Prijem pravilnika o službenom putu
   - 1.2. Prijem putnog naloga (od uprave)
   - 1.3. Prijem podataka za obradu putovanja (od zaposlenog)
   - 1.4. Priprema putnog naloga za dalju obradu (objedinjavanje podataka od uprave i zaposlenog)

2. OBRADA BONOVA
   - 2.1. Obrada preuzimanja bonova
     - 2.1.1. Zahtevanje bonova za gorivo
     - 2.1.2. Uplata za bonove
       - 2.1.2.1.Formiranje naloga (banci) za uplatu za bonove
       - 2.1.2.2.Prijem izveštaja o izvršenoj uplati (od banke)
       - 2.1.2.3.Izdavanje izveštaja o izvrsenoj uplati (naftnoj industriji)
     - 2.1.3. Preuzimanje bonova (od naftne industrije)
   - 2.2. Izdavanje bonova (zaposlenom)

3. OBRADA RAČUNA
   - 3.1. Prijem računa
   - 3.2. Obezbeđivanje novca
     - 3.2.1. Formiranje naloga za podizanje gotovine sa računa
     - 3.2.2. Preuzimanje gotovine
   - 3.3. Refundiranje (zaposlenom)
     - 3.3.1. Formiranje isplatne liste
     - 3.3.2. Izdavanje gotovine

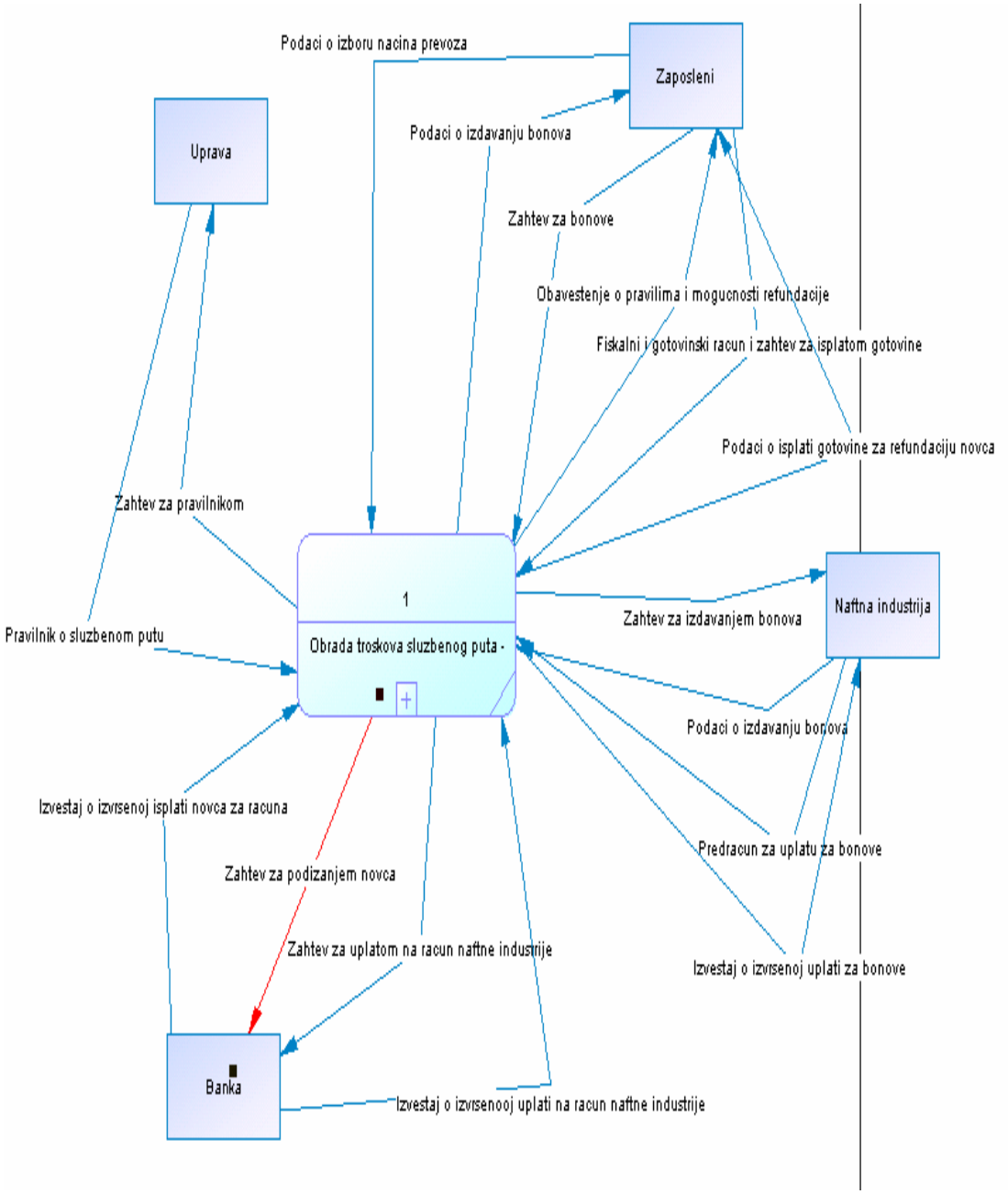
NAPOMENE ZA CRTANJE DIJAGRAMA:

1. *Tokovi podataka moraju imati nazive kao dokumenti ili paketi podataka. Ne sme biti naziva kao aktivnosti ili kao materija.*
2. *Ne sme biti nepovezanih delova dijagrama.*
3. *Procesi razmenjuju podatke preko skladišta – ne sme biti direktne povezanosti, ali i ne sme biti situacija da procesi uopšte ne komuniciraju – da funkcionišu u potpunosti nezavisno od ostalih procesa.*

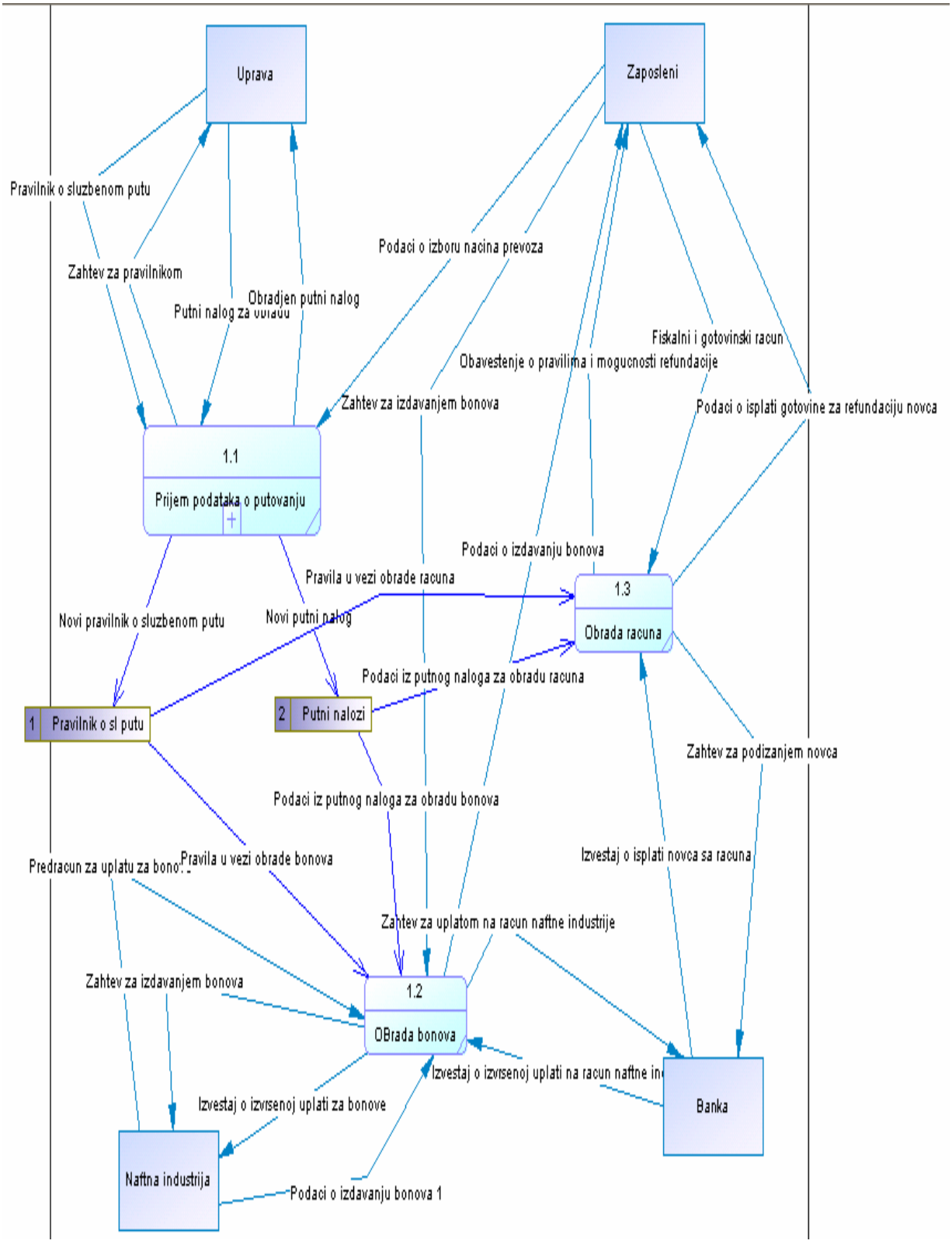
ZADATAK: U skladu sa navedenim stablom procesa uraditi dijagrame toka podataka.

REŠENJE:

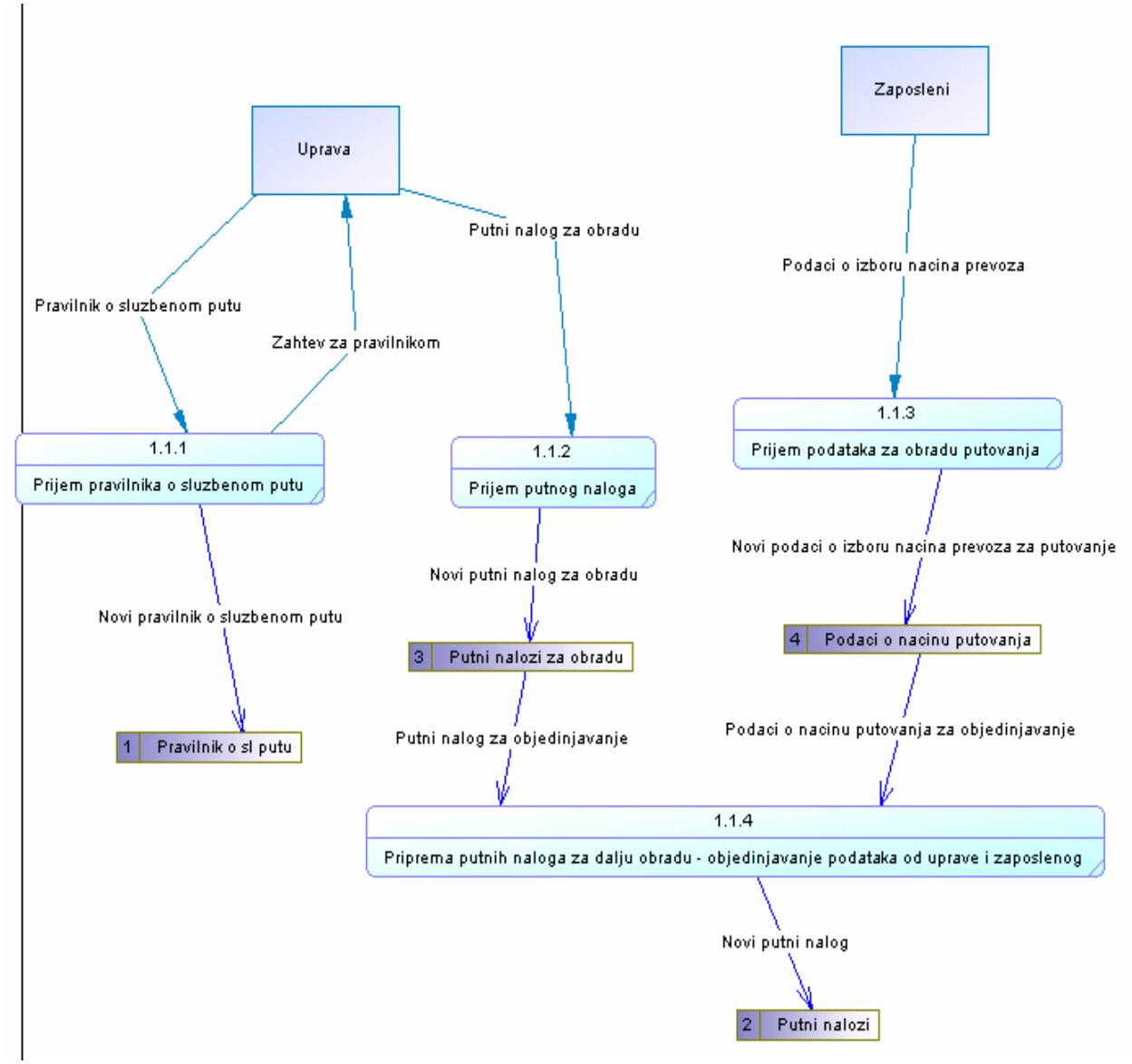
Dijagram toka podataka 0. nivoa dekompozicije:



Dijagram toka podataka 1. nivoa dekompozicije:



Dijagram toka podataka 2. nivoa dekompozicije:



4.3. Rečnik podataka

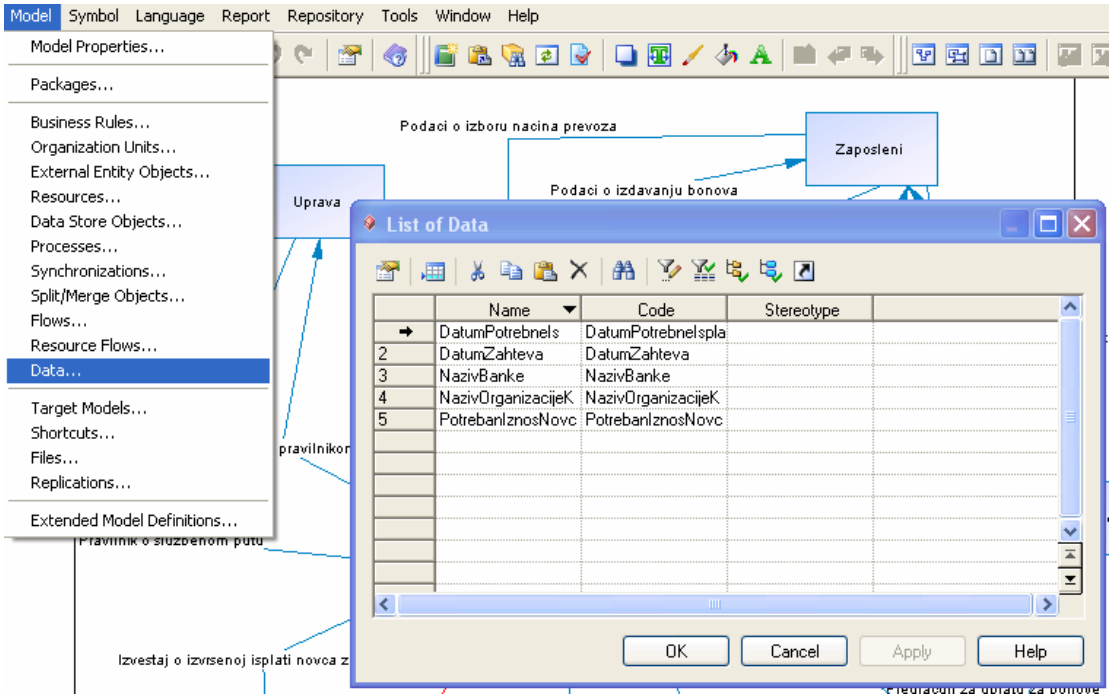
Rečnik podataka predstavlja skup svih elementarnih podataka na nivou celog dijagrama toka podataka, kao i skup sintaksnih specifikacija strukture tokova i skladišta podataka. Popunjavanjem strukture svih tokova podataka i skladišta podataka u okviru celog modela, dobijamo popunjen rečnik podataka. Na taj način definisane su informacione potrebe navedenog sistema. Na osnovu rečnika podataka dalje se razvijaju odgovarajući modeli podataka koji na uređen način treba da podrže ažuriranje i korišćenje podataka iz baze podataka i programske aplikacije.

Napomene:

1. *Nastojimo da nazivi elementarnih podataka budu u jednini i da nemaju nazive kao strukture, vec kao atomarne vrednosti (npr. ne Mesto, vec NazivMesta).*

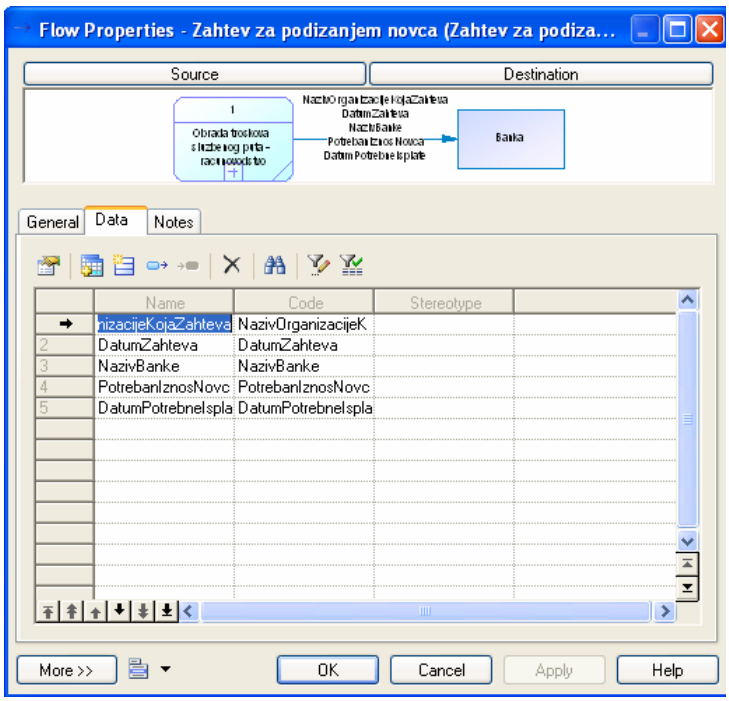
2. *Elementarne podatke opisujemo tipom podatka ili domenom. Pored standardnih tipova podataka(string, integer, realni broj, datumska vrednost) postoje i uže definisani skupovi vrednosti koji se nazivaju domeni i definisani su nad standardnim tipovima, sa opisom ograničenja ili nabrajanjem pojedinačnih vrednosti koje su dozvoljene.*

Popunjavanje elementarnim podacima u CASE alatu Power Designer 15 započinje opcijom Model – Data, gde se dobija spisak svih elementarnih podataka modela.

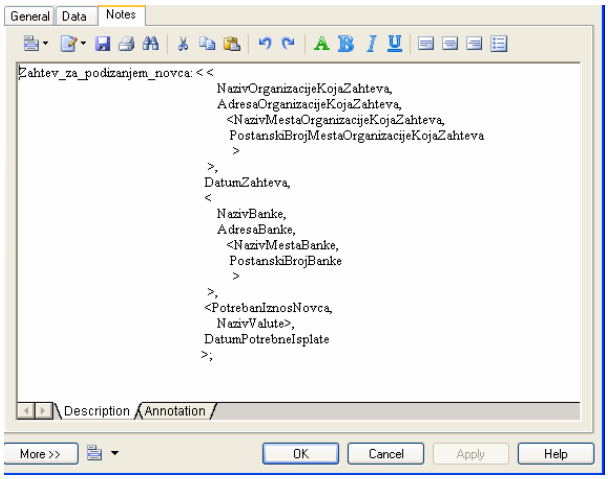


Slika 3.2.3.

Nakon popunjavanja elementarnim podacima koji se odnose na neki konkretan tok podataka (na slici sa DTP 0. nivoa obeležen crvenom bojom) u ovom odeljku, pokrećemo opciju Properties nad navedenim tokom podataka i karticu Data, kao na sledecoj slici.



U okviru ovog odeljka dodajemo odgovarajuće elementarne podatke, a u okviru kartice Notes definisemo strukturu, koristeci specijalne sintaksne simbole:
<> - bliža povezanost elementarnih podataka, svaki se popunjava i to jednom
{} – ponavljanje podataka iste strukture, najčešće se obeležava sa {< >}
[] – ako ima jedan element, opciono se popunjava, ako ima vise, međusobno se isključuju
// - kombinacije, unija podataka



4.4. REŠENI ZADACI

ZADATAK
Na osnovu datog opisa posla za osnovnu delatnost:
1. Uraditi stablo dekompozicije procesa.
2. Nacrtati dijagram toka podataka 0 i 1 nivoa dekompozicije, odnosno jedan dijagram 2. nivoa za jedan odabrani proces sa 1. nivoa dekompozicije.
3. Prikazati elementarne podatke, tipove podataka i sintaksni prikaz strukture datog dokumenta.

1. PRIMER - Mala turistička agencija

OPIS POSLA
Da bi turistička agencija mogla da radi, mora imati licencu za rad. Za licencu se prijavljuje ministarstvu turizma i podnosi zahtev. Nakon dobijanja licence, turistička agencija može da radi. S obzirom da je ovde reč o maloj turističkoj agenciji (dalje MTA), ona sklapa ugovor o saradnji sa drugim većim turističkim agencijama (dalje VTA), koje imaju svoje ponude i direktne kontakte sa turističkim destinacijama (hotelima i slično), tako da je ona njihov predstavnik. Od tih drugih turističkih agencija dobija katalog ponude na sajmu turizma, koji se organizuje jednom godišnje, obično u februaru. Kada građanin dođe sa zahtevom za organizovanje putovanja, od MTA dobija kataloge ponuda. Nakon razgledanja kataloga, građanin se odlučuje za neku destinaciju. Građanin ostavlja lične podatke, podatke drugih osoba koje sa njim putuju i podatke o željenoj destinaciji i trajanju boravka, kao i načinu transporta. MTA kontaktira VTA tako što šalje dokument "najava puta" sa svim prethodno uzetim podacima, kako bi zahtevala predračun za uplatu. Nakon prijema predračuna, MTA kontaktira građanina da ga obavesti o prispeću predračuna. Građanin usmeno dobija podatke potrebne za uplatu i uplaćuje novac na žiro račun MTA. Nakon uplate, donosi MTA priznanicu o uplati. MTA šalje podatke o izvršenoj uplati (na osnovu priznanice o uplati) VTA i zahteva od VTA da im ona pošalje vaučer. Nakon prijema vaučera, MTA obaveštava građanina o prispeću vaučera.Najkasnije 7 dana pre puta građanin od turističke agencije dobija vaučer, koji predstavlja dokaz da je uplatio troškove boravka izvršena i da može da ide na put.

DOKUMENT ZA ANALIZU:

Šalje:
TURISTIČKA AGENCIJA "SUNCE"
Djure Salaja 33
22 300 Novi Banovci
datum: 21.4.2011.

Upućeno:
MEGA TRAVEL
Karadjordjeva bb
11000 BEOGRAD

NAJAVA PUTA

Destinacija: HOTEL "PLAVI HORIZONT", Radovići, Crna Gora

Vreme: 21.7.2011. - 31.7.2011.

VRSTA USLUGE (podvuci izabrano) : b (nocenje), bb (nocenje sa doruckom), **pp (polupansion)**, p (pansion)

SPISAK GOSTIJU:
1. Jovan Petrovic, JMBG 1111111111111, Datum rođenja: 14.3.1970. odrasla osoba
2. Marija Petrovic, JMBG 2222222222222, Datum rođenja: 18.8.1974. odrasla osoba
3. Jovana Petrovic, JMBG 3333333333333, Datum rođenja: 23.3.2009. dete do 2 godine - gratis
4, Ivana Petrovic, JMBG 4444444444444, Datum rođenja: 14.4.2003. dete

Odgovorno lice
_____

NAPOMENA: U nastavku su data studentska rešenja zadatka sa kolokvijuma, sa komentarima grešaka.

PRVO REŠENJE :

1. ZADATAK - Stablo dekompozicije procesa

0.Osnovna delatnost – svrha postojanja: Organizovanje boravka putnika na željenoj destinaciji
1. Pripremna faza
   1.1. Izdavanje kataloga potencijalnom klijentu (ili bolje: Informisanje klijenta o ponudama)
   1.2. Prijem prijave za izbor željene destinacije od klijenta
   1.3. Slanje dokumenta „najava puta" ili Najavljivanje puta
   1.4. Zahtevanje predračuna
   1.5. Prijem predračuna

2. Izvršna faza
   2.1.Obavestavanje klijenta o uplati
   2.2. Prijem uplate od strane klijenta
   2.3.Slanje podataka o uplati VTA i zahtev za vaučerom
   2.4.Prijem vaučera

3. Završna faza
3.1.Obaveštavanje klijenta o prispeću vaučera
3.2. Izdavanje vaučera klijentu

3. ZADATAK – analiza strukture dokumenta (toka podataka)

Naziv	Tip / DOMEN	Napomena
Naziv_turisticke_agencije	String100	
Ulica_turisticke_agencije	String100	
Ptt_broj_turisticke_agencije	String6	
Naziv_Mesta_turisticke_agencije	String50	
Datum_slanja_dokumenta	Date	
Naziv_firme_primaoca	String100	
Ulica_firme_primaoca	String 100	
Ptt_firme_primaoca	String 6	
Naziv_Mesta_firme_primaoca	String50	
Destinacija_putovanja	String100	
Vreme_putovanja	Date	Treba da su 2 elementarna podatka: datum pocetka i datum zavrsetka
Vrsta usluge	Vrste_usluga	
Ime_gosta	String30	
Prezime_gosta	String30	
JMBG_gosta	String13	
Datum_rodjenja_gosta	Date	
Uzrast_gosta	TipUzrasta	

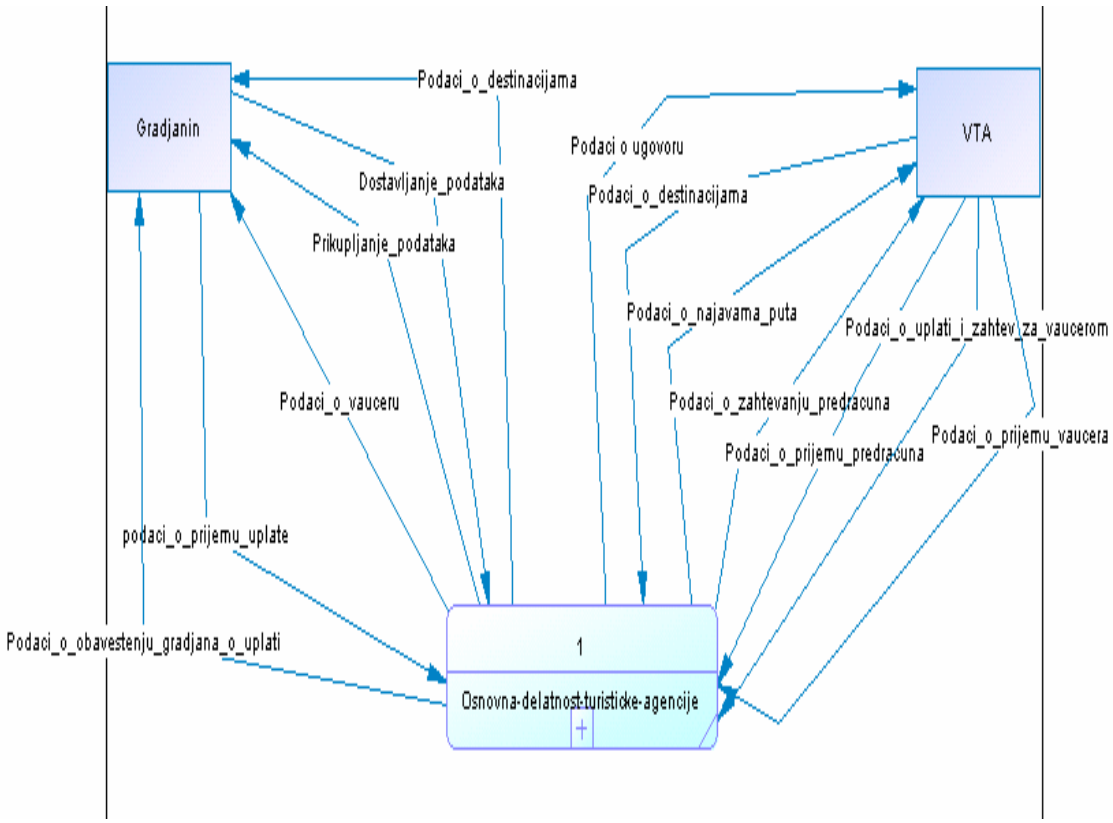
DOMENI:

Vrste_usluga :String100(“b (noćenje)”,” bb (noćenje sa doručkom)”,''pp (polupansion)'', ''p (pansion)'')
TipUzrasta: String20(‘’(odrasla_osoba)’’,”(dete_do_2_godine)”,”(dete)”)

SINTAKSNI PRIKAZ STRUKTURE DOKUMENTA:

```
Najava_puta: <          <Naziv_turisticke_agencije,
                          <Ulica_turisticke_agencije,
                           Ptt_turisticke_agencije,
                           Naziv_Mesta_turisticke_agencije,
                          >
                         >
                         <Naziv_firme_primaoca,
                          < Ulica_firme_primaoca,
                            Ptt_firme_primaoca,
                            Naziv_Mesta_firme_primaoca,
                          >
                         >
                          Destinacija_putovanja,
                          <Datum_pocetka_boravka,
                           DatumZavrsetkaboravka>
                          Vrste_usluga,
                           {<Ime_gosta,
                            Prezime_gosta,
                            JMBG_gosta,
                            Datum_rodjenja_gosta,
                            Uzrast_gosta
                           >}
                          >
                    >;
```

1. ZADATAK - dijagram toka podataka 0.nivoa



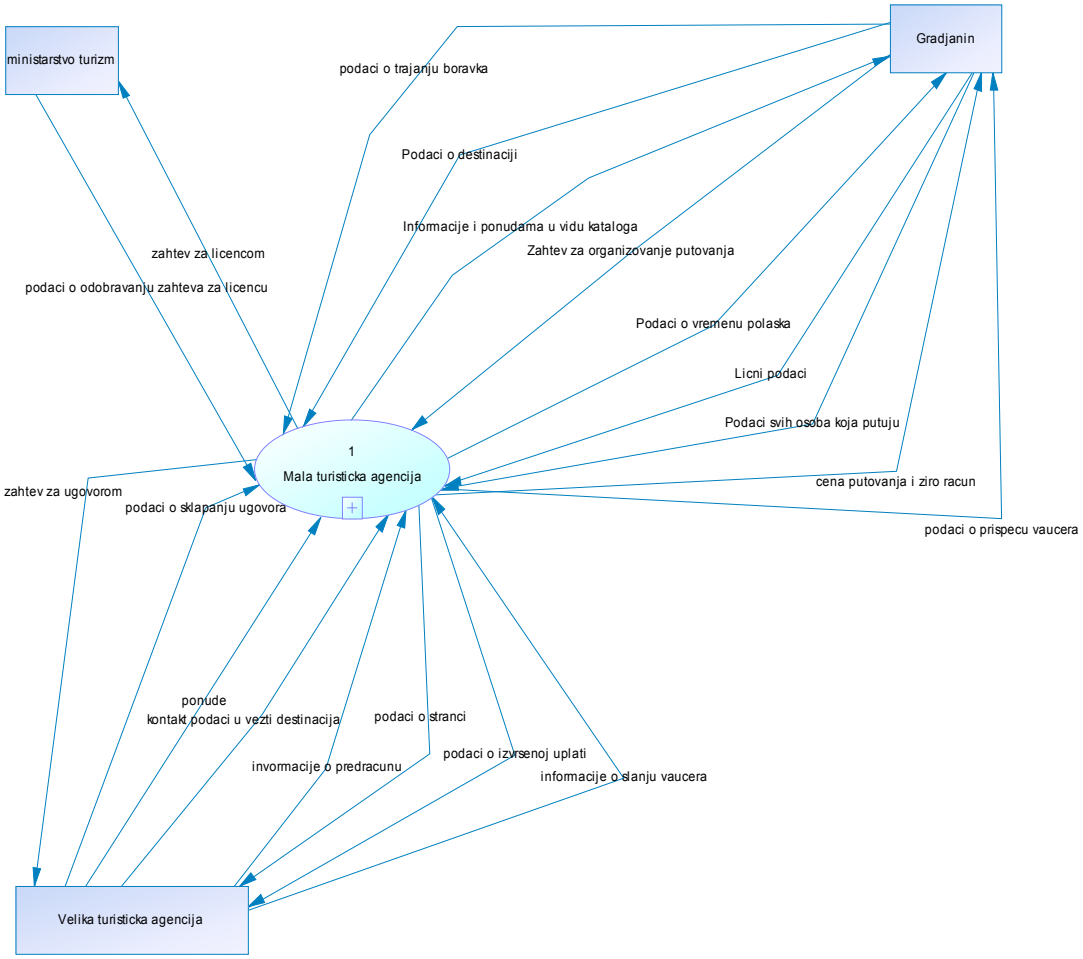
DRUGO REŠENJE :

1. zadatak – stablo procesa

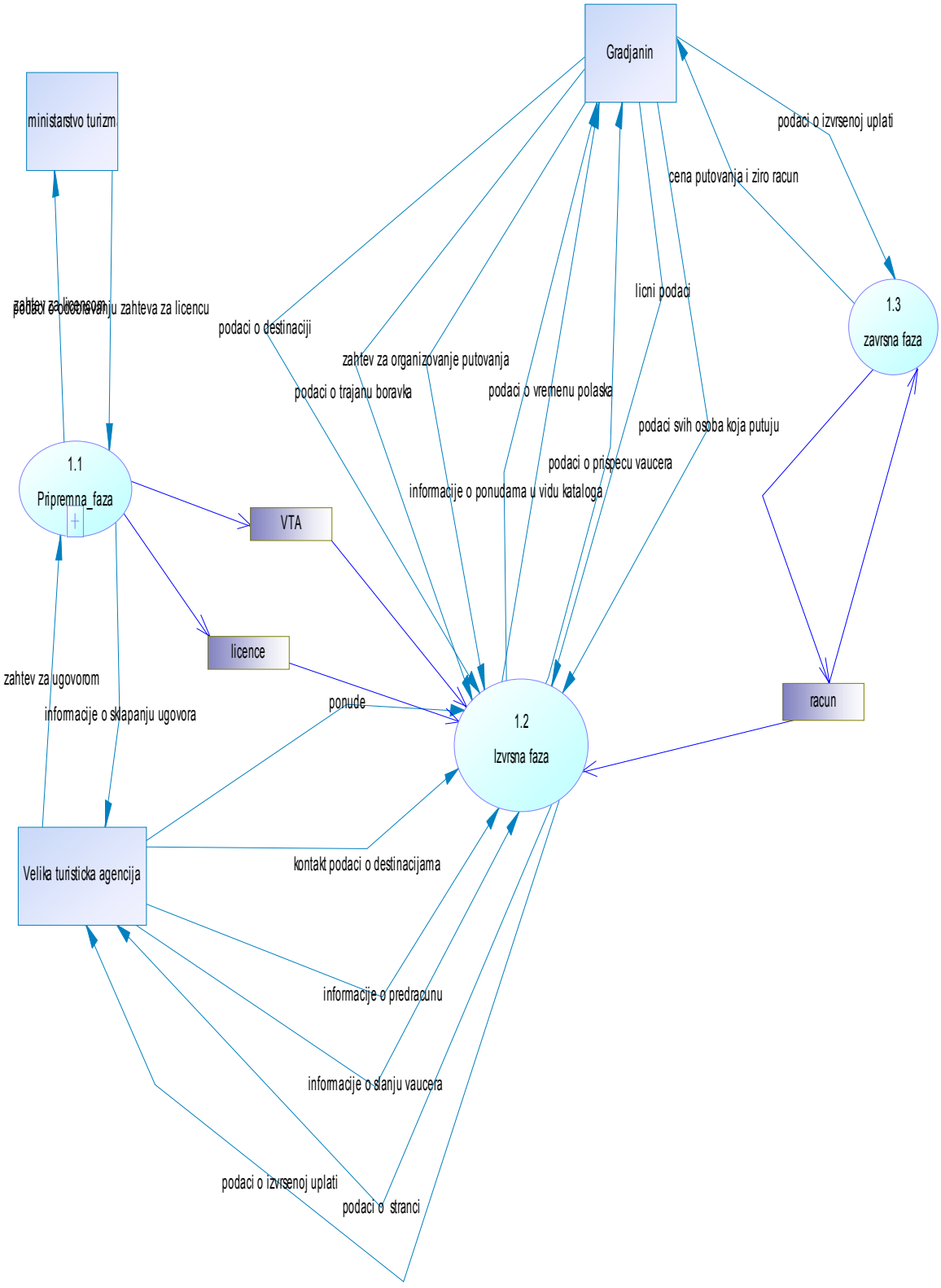
0. Mala turistička agencija
1. Pripremna faza <---- pripremna faza u ovom zadatku pripada pomoćnim delatnostima
   - 1.1 Nabavljanje licence
     - 1.1.1. Slanje zahteva za licencom
     - 1.1.2. Objavljivanje informacija o licenciranju
   - 1.2 Sklapanje ugovora sa drugim agencijama
     - 1.2.1 Slanje zahteva za ugovorom
     - 1.2.2 Podpisivanje ugovora
2. Izvršna faza – organizovanje putovanja
   - 2.1. Prijem gradjana
     - 2.1.1 Izdavanaje kataloga
     - 2.1.2. Uzimanje podataka osoba koja putuju
     - 2.1.3. Uzimanje podataka o destinaciji
   - 2.2. Kontaktiranje VTA
     - 2.2.1. Slanje dokumenta o najavi puta
     - 2.2.1. Prijem predracuna
3. Završna vaza
   - 3.1 Naplata
     - 3.1.1. kontaktianje stranke o predracunu
     - 3.2.2 Slanje priznanice o uplati
     - 3.1.2 Slanje zahteva za vaučer
   - 3.2. Izdavanje vaučera

2. zadatak – dijagram toka podataka

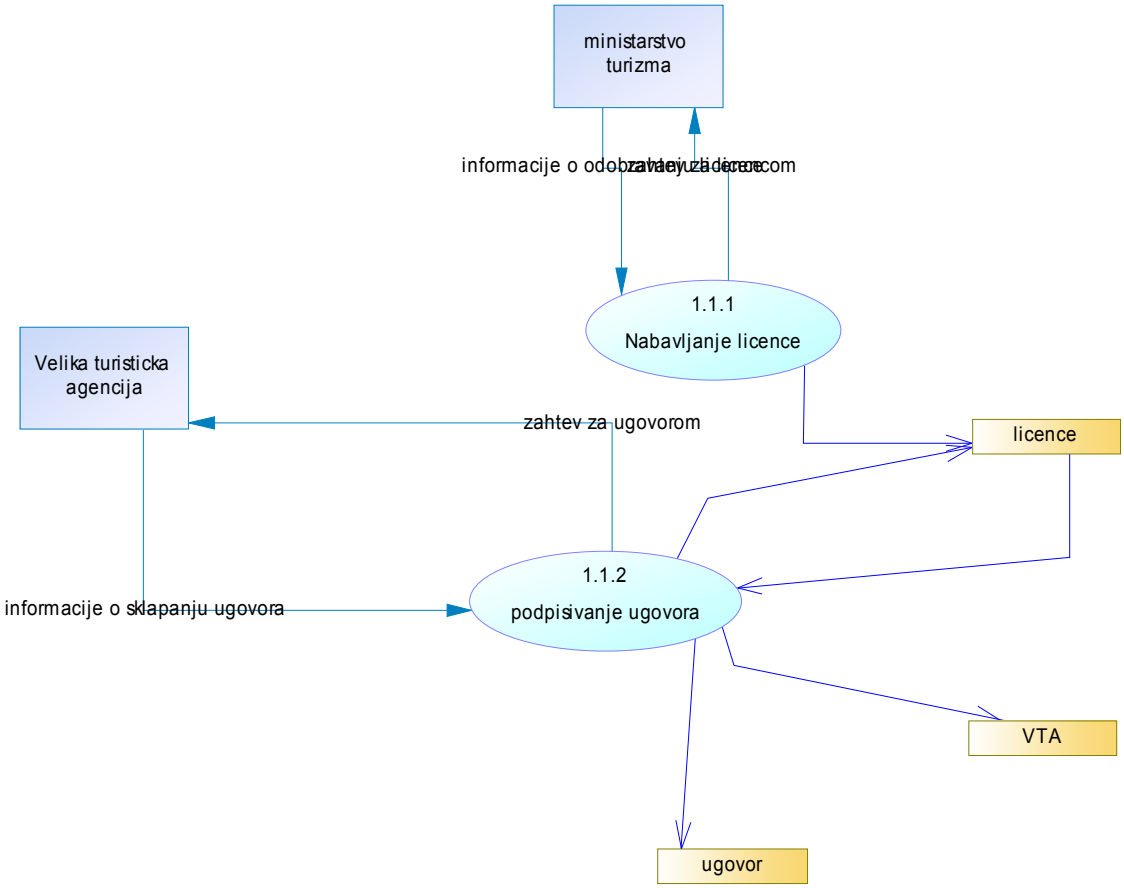
DTP 0. nivoa



DTP 1. nivoa


Gradjanin
ministarstvo turizm
podaci o izvrsenoj uplati
cena putovanja i ziro racun
licni podaci
1.3
zavrsna faza
podaci o destinaciji
zahtev za organizovanje putovanja
podaci o trajanu boravka
podaci o vremenu polaska
podaci svih osoba koja putuju
podaci o prispecu vaucera
informacije o ponudama u vidu kataloga
1.1
Pripremna_faza
VTA
licence
zahtev za ugovorom
informacije o sklapanju ugovora
ponude
1.2
Izvrsna faza
racun
Velika turisticka agencija
kontakt podaci o destinacijama
informacije o predracunu
informacije o slanju vaucera
podaci o izvrsenoj uplati
podaci o strranci

DTP 2. nivoa



3. zadatak – analiza dokumenta

SPISAK ELEMENTARNIH PODATAKA

Naziv elementarnog podatka	**Tip podatka / Domen**
ImeAgencije	String[40]
MestoAgencije	String[40]
AdresaAgencije	String[40]
PttMestaAgencije	Int
TelefonAgencije	String[20]
DatumNajavePuta	Date
ImeVelikeAgencije	String[40]
MestoVelikeAgencije	String[40]
AdresaVelikeAgencije	String[40]
PttMestaVelikeAgencije	Int
TelefonVelikeAgencije	String[20]
NazivDokumenta <----- nije elementarni podatak	String[40]
NazivHotelaDestinacije	String[30]
MestoDestinacije	String[40]
DrzavaDestinacije	String[20]
TerminDestinacije <---- treba odvojiti u 2 datuma	Date
VrstaUsluge	Usluga
ImeIprezimeGosta <--- treba odvojiti u 2 el. podatka	String[40]
JmbgGosta	String[13]
DatumRodjenjaGosta	Date
UzrastGosta	Uzrast

DOMENI:

Usluga : String 100 [nocenje, nocenje sa doruckom, polupansion,pansion]
Uzrat: String 40 [odrasla osoba,dete do 2 godine, dete]

SINTAKSNI PRIKAZ STRUKTURE DOKUMENTA:

```
Dokument o najavi puta:
        <<Ime agencije,
        <Mesto agencije,
        Adresa agencije
        Ptt mesta agencije>
        Telefon agencije>
        Datum najave puta,
        <Ime velike agencije,
        <Mesto velike agencije,
        Adresa velike agencije,
        Ptt mesta velike agencije>,
        Telefon velike agencije>,
        <Naziv hotela destinacije,
        Mesto destinacije,
        Drzava destinacije>,
        <DatumPocetkaBoravka,
        DatumZavrsetkaBoravka>,
        Vrsta usluge,
                {<Ime gosta,
                Prezime gosta,
                Jmbg gosta,
                Datum rodjenja gosta,
                Uzrast gosta>}
        >
```

5.DIZAJN FUNKCIONALNOG ASPEKTA PREDLOGA REŠENJA

5.1. Preslikavanje primitivnih poslovnih procesa u softverske funkcije

Informacione tehnologije treba da podrže poslovanje u svim funkcionalnim aspektima. Potrebno je razmotriti mogućnosti razvoja odgovarajuće IT podrške svakom poslovnom procesu. Osnovna ideja vezana za dizajn funkcionalnog aspekta budućeg rešenja se sastoji u preslikavanju primitivnih procesa iz stabla procesa u odgovarajuće softverske funkcije. [1]

Metoda preslikavanja sastoji se u formiranju tabele gde su u prvoj koloni primitivni procesi, u drugoj nazivi radnih mesta organizacije koja izvršava navedeni primitivni proces, dok se u nastavku tabele nalaze nazivi softverskih funkcija koje «realizuju» odgovarajući poslovni proces, kao i dodatne softverske funkcije i softverske funkcije koje predstavljaju preduslov za uspešno izvršavanje osnovnih softverskih funkcija. U navedenoj tabeli definišu se i profili korisnika (actor-i) odgovarajućeg slučaja korišćenja, odnosno softverske funkcije.

Najčešće softverske funkcije u oblasti informacionih sistema:

- automatsko izračunavanje podataka
- ažuriranje (unos, brisanje, izmena) podataka
- prikaz podataka (tabelarni, analiticki - pojedinačni)
- filtriranje podataka
- štampanje izvestaja
- eksport podataka (snimanje podataka u druge formate)
- statistička obrada podataka

NAPOMENE:

- *Najčešće profili korisnika imaju iste nazive kao i nazivi radnih mesta za primitivne procese*
- *Izuzetno je važna preciznost izražavanja, jer u ovom procesu popunjavanja tabele zapravo vršimo specifikaciju zadataka za programere, pa mora biti jasno šta konkretno i na koji način treba biti realizovano. Primer: nije dobro nazvana softverska funkcija – štampanje izveštaja ili filtriranje podataka ... bitno je navesti kog izveštaja, filtriranje kojih podataka prema kom kriterijumu*
- *Obratiti pažnju da se ne mešaju poslovni procesi sa softverskim, tj. da se u kolonama tabele koje se odnose na softverske funkcije upisuju zaista samo nazivi softverskih funkcija, a ne nazivi poslovnih procesa*
- *Voditi računa o jednini i množini u nazivu. Najčešće unos podataka se vrši za jednu pojavu, pa se koristi jednina (primer: Unos podataka o uplati, a nije dobro: Unos podataka o uplatama), dok se kod tabelarnog prikaza i štampe nekih vrsta izveštaja može koristiti množina (npr. Tabelarni prikaz uplata, Štampanje spiska uplata)*
- *Ukoliko stablo procesa nije dobro urađeno (proces koji je postavljen kao primitivan ipak nije i može se dalje dekomponovati, stablo nije potpuno). Moguće je u ovoj fazi korigovati stablo procesa i u skladu sa tom korekcijom uraditi i tabelu preslikavanja.*
- *Izuzetno je bitno utvrditi fokus poslovnog procesa. Naime, ukoliko je poslovni proces tipa prijema podataka, onda bismo kao odgovarajuću softversku funkciju koja dati poslovni proces pokriva imali tip softverske funkcije - „unos podataka". Ako je fokus poslovnog procesa na prezentovanju podataka, imali bismo softverske funkcije tipa „tabelarni prikaz sa filtriranjem", stampanje,a ako bi fokus bio na dostavljanju podataka, tada bismo imali eksportovanje podataka, slanje e-maila i slično.*
- *Veoma je bitno pravilno odrediti preduslov za izvršavanje softverske funkcije. Ukoliko je u pitanju unos podataka, najčešće je preduslov neki drugi unos (recimo šifarnika koji su potrebni), a ako je osnovna softverska funkcija tipa tabelarnog prikaza, preduslov je naravno unos podataka.*

Literatura

[1] Kazi Lj., Radulović B., Radosav D., Bhatt M., Grmuša N., Štiklica N.: *Business Process Model and Elements Of Software Design: The Mapping Approach*, Konferencija «Applied Internet and Information Technologies» AIIT 2012, Zrenjanin

5.2. Dizajn softverskog rešenja uz definisanje prioriteta razvoja

Preslikavanje primitivnih procesa u softverske funkcije može se zasnivati na smernicama heuristika. Ipak, samo preslikavanje nije jednoznačno, jer može biti više predloga rešenja za isti primitivni proces. Takođe, iste softverske funkcije koje su predlog mogu biti realizovane na različitim tehnološkim platformama.

U pripremanju dokumentacije ponude softverske firme nekom organizacionom sistemu (klijentu), u procesu definisanja IT podrške poslovnim procesima mogu se navesti sva moguća raspoloživa i ostvariva rešenja. U tom procesu dizajn kao kreativna aktivnost dolazi do izražaja, ali i procena ograničenja (vreme, kadrovi – njihovo znanje, opterećenje itd) koja utiču na konačno formiranje ponude koja bi se predstavila klijentu. Dakle, spisak svih realno ostvarivih rešenja (sa postojećom razvojnom tehnologijom i znanjem) se klasifikuje po prioritetima ili čak filtrira (neke opcije se odbacuju) s obzirom na ograničenja u procesu razvoja. Tek tada dobijamo konačan oblik prve ponude za realizaciju softvera. Konačan oblik ponude dobija se tek nakon što je prva ponuda predstavljena klijentu i usaglašena sa zahtevima klijenta.

Definisanje prioriteta u razvoju softverskih funkcija odnosi se na razvrstavanje svih predloga ideja za podršku odgovarajućem poslovnom procesu u najmanje 2 kategorije:

1. Pristup – prvi prioritet imaju tzv. «need to have» softverske funkcije tj. one koje su nužne, bez kojih se ne može podržati smisao poslovnog procesa, dok drugi prioritet imaju tzv. «nice to have» softverske funkcije – dodatne funkcije koje omogućavaju prijatan rad sa softverom, ali koje nisu nužne, bez njih poslovni sistem može da funkcioniše
2. Pristup – prvi prioritet imaju softverske funkcije koje direktno podržavaju navedeni poslovni proces, a drugi prioritet softverske funkcije koje nemaju nikakve direktne veze sa datim poslovnim procesom, ali dodatno proširuju mogućnosti.

NAPOMENA:

1. *Najčešće ako su softverske funkcije prvog prioriteta tipa «unos podataka», onda su softverske funkcije 2. prioriteta tipa «tabelarni prikaz sa filtriranjem», «štampa» itd.*
2. *Ne mora svaka softverska funkcija 1. prioriteta da ima odgovarajuće softverske funkcije 2. prioriteta (npr. ako je softverska funkcija 1. prioriteta tipa «tabelarni prikaz»)*

Konačno, izgled tabele koja se koristi za preslikavanje primitivnih procesa u softverske funkcije:

1	2	3	4	5	6	7	8	9
PRIM. PROC.	**RADNO MESTO**	**SOFT. F. 1. PRIOR.**	**ACTOR**	**SOFT. F. 2. PRIOR.**	**ACTOR**	**SOFT. F. PREDUSLOV**	**ACTOR**	**TIP SOFTV.**

Objašnjenje:

1- primitivni proces
2- radno mesto koje izvršava primitivni proces
3- softverska funkcija 1. prioriteta
4- actor (profil korisnika) koji ima prava pristupa i izvršavanja softverske funkcije 1. prioriteta
5- softverska funkcija 2. prioriteta

6- actor (profil korisnika) koji ima prava pristupa i izvrsavanja softverske funkcije 2. prioriteta
7- softverska funkcija koja je preduslov (mora biti izvršena) za softversku funkciju 1. prioriteta
8- actor (profil korisnika) koji ima prava pristupa i izvršavanja softverske funkcije preduslova
9- Tip softvera za softverske funkcije prikazane levo

ZADATAK
Na osnovu datog opisa posla i stabla procesa popuniti tabelu kojom se vrši preslikavanje poslovnih procesa u softverske funkcije. Odrediti preduslove i prioritete razvoja.

PRIMER:

Organizacioni sistem: STUDENTSKI DOM

OPIS POSLA
Na pocetku svake godine, studenti koji ne stanuju u mestu u kome je fakultet se prijavljuju za studentski dom. Preuzimaju iz studentske službe uverenje o statusu studenta i prosek ocena. Kompletiranu dokumentaciju (prijava sa ličnim podacima i podacima stanovanja i uverenje o status studenta sa prosekom ocena) podnose sekretarijatu studentskog doma. Na osnovu primljenih prijava, sekretarijat doma formira rang liste. U obzir se uzimaju parametri kao sto su: udaljenost mesta stanovanja od mesta fakulteta, godina studija, prosek ocena. Nakon formiranja rang liste, postavljaju se na vratima studentskog doma i na vratima fakulteta. Studenti koji su dobili mogućnost da stanuju u domu prvog dana dolaze da se prijave (dostavljaju ličnu kartu ili indeks radi identifikacije), a zatim dobijaju ključeve sobe. Za vreme boravka u domu dužni su da se pridržavaju kućnog reda i imaju na raspolaganju sobu i prateće prostorije za učenje i rekreaciju. Novu posteljinu zadužuju periodično (jednom nedeljno). Ishrana u studentskom domu je rešena tako sto studenti dobijaju (kupuju) bonove (na početku svakog meseca) ukoliko žele da se hrane u studentskoj menzi koja se nalazi u okviru studentskog doma. Na kraju školske godine u periodu razduživanja studenti su duzni da prijave izlazak iz doma domaru. On dolazi da proveri stanje sobe. Nakon toga student može da vrati ključ i da se razduži.

STABLO PROCESA

STUDENTSKI DOM – OBEZBEĐIVANJE SMEŠTAJA STUDENTIMA
1. PRIPREMNA FAZA – PRIMANJE STUDENATA U DOM
   - 1.1. Raspisivanje konkursa o prijemu studenata u studentski dom
   - 1.2. Preuzimanje dokumentacije od studenata
   - 1.3. Formiranje rang lista prijavljenih studenata
   - 1.4. Objavljivanje rang lista prijavljenih studenata
   - 1.5. Useljavanje primljenih studenata
     - 1.5.1. Identifikacija primljenih studenata
     - 1.5.2. Raspoređivanje primljenih studenata po sobama
     - 1.5.3. Uručivanje ključeva sobe primljenim studentima
     - 1.5.4. Uručivanje kućnog reda
2. IZVRSNA FAZA – ORGANIZOVANJE BORAVKA STUDENATA U STUDENTSKOM DOMU
   - 2.1. Vođenje računa o sprovođenju kućnog reda
   - 2.2. Održavanje soba i pratećih prostorija
   - 2.3. Obezbeđivanje čiste posteljine
   - 2.4. Uručivanje bonova za studentku menzu studentima
3. ZAVRSNA FAZA – OMOGUCAVANJE RAZDUZIVANJA STUDENTIMA
   - 3.1. Primanje prijava o iseljavanju stanara doma (studenata)
   - 3.2. Proveravanje stanja soba studenata koji se iseljavaju

3.3. Preuzimanje ključa soba od studenata koji se iseljavaju

REŠENJE:

Prikazana su rešenja za samo 2 primitivna procesa. Možemo videti da se za jedan primitivni proces može dizajnirati i više softverskih funkcija.

1. rešenje - prvi pristup podeli prioriteta – need to have / nice to have

U okviru ovog rešenja razvoj web aplikacije generalno smo smatrali dodatnom funkcionalnošću koja se moze razvijati u nekoj narednoj iteraciji («nice to have») pa smo postavili sve funkcije web aplikacije u softverske funkcije 2. prioriteta.

Primitivni poslovni proces	**Radna uloga (radno mesto)**	**Softverska funkcija * 1. po prioritetu**	**Actor**	**Softverska funkcija * 2. po prioritetu**	**Actor**	**Softverska funkcija * Preduslov za uspesno izvrsavanje**	**Actor**	**Tip softvera (c/s LAN, WEB)**
Raspisivanje konkursa o prijemu studenata u studentski dom	Sekretar studentskog doma	Unos podataka o konkursu Stampanje konkursa	Sekretar	Tabelarni prikaz unetih konkursa Filtriranje konkursa prema godini Analitički prikaz konkursa	Sekretar	Unos podataka o studentskom domu Unos podataka o fakultetima	Sekretar	c/s LAN
				Analitički (pojedinačni) prikaz aktuelnog konkursa	Neimenovani korisnik			web
Preuzimanje dokumentacije od studenata	Sekretar	Unos podataka o prijavi studenta za boravak u domu	Sekretar	Tabelarni prikaz svih unetih prijava Filtriranje spiska prijava prema prezimenu Štampanje spiska svih prijavljenih studenata				c/s LAN
				Unos podataka o prijavi studenta za boravak u domu	Student			WEB

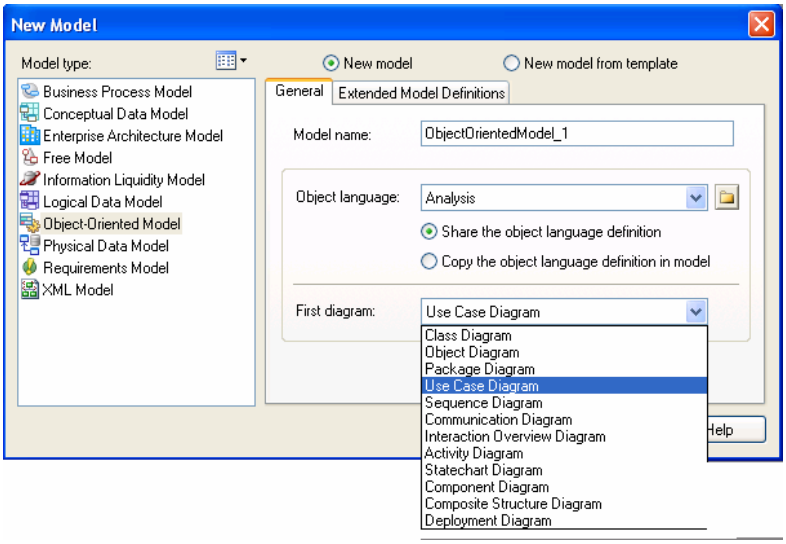
2. rešenje – drugi pristup podeli prioriteta – direktno rešavanje fokusa ili nema nikakve veze sa fokusom poslovnog procesa

U okviru ovog pristupa sve softverske funkcije koje direktno rešavaju fokus poslovnog procesa postavljene su u kolonu 1. po prioritetu, bez obzira da li su web aplikacije. U softverske funkcije 2. prioriteta postavljamo dodatne opcije koje su korisne, ali ne rešavaju direktno fokus poslovnog procesa.

Primitivni poslovni proces	**Radna uloga**	**Softverska funkcija * 1. po prioritetu**	**Actor**	**Softverska funkcija * 2. po prioritetu**	**Actor**	**Softverska funkcija * Preduslov za uspesno izvrsavanje**	**Actor**	**Tip softvera (c/s LAN, WEB)**
Raspisivanje konkursa o prijemu studenata u studentski dom	Sekretar studentskog doma	Unos podataka o konkursu Stampanje konkursa Tabelarni prikaz unetih konkursa Filtriranje konkursa prema godini Analiticki prikaz konkursa	Sekretar			Unos podataka o studentskom domu Unos podataka o fakultetima	Sekretar	c/s LAN
		Analiticki (pojedinacni) prikaz aktuelnog konkursa			Neimenovani korisnik			web
Preuzimanje dokumentacije od studenata	Sekretar	Unos podataka o prijavi studenta za boravak u domu	Sekretar	Tabelarni prikaz svih unetih prijava Filtriranje spiska prijava prema prezimenu Stampanje spiska svih prijavljenih studenata				c/s LAN
		Unos podataka o prijavi studenta za boravak u domu	Student					WEB

5.3. Dijagram slučaja korišćenja

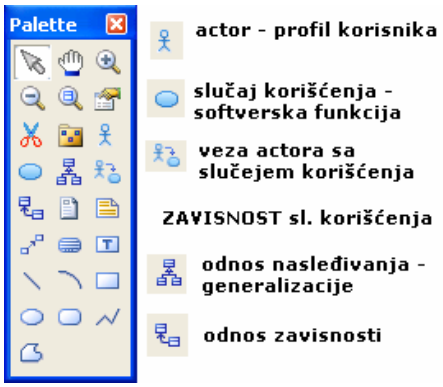
U okviru kreiranja predloga dizajna novog rešenja, koriste se dijagrami UML-a (Unified Modelling Language). U okviru CASE alata Power Designer 15, podrška UML dijagramima data je u okviru tzv. «object oriented modela» koji se biraju kao tip modela. Na slici 5.3. vidimo spisak svih vrsta modela podržanih u kategoriji «object-oriented model», koji su zapravo modeli UML-a verzije 2.0.



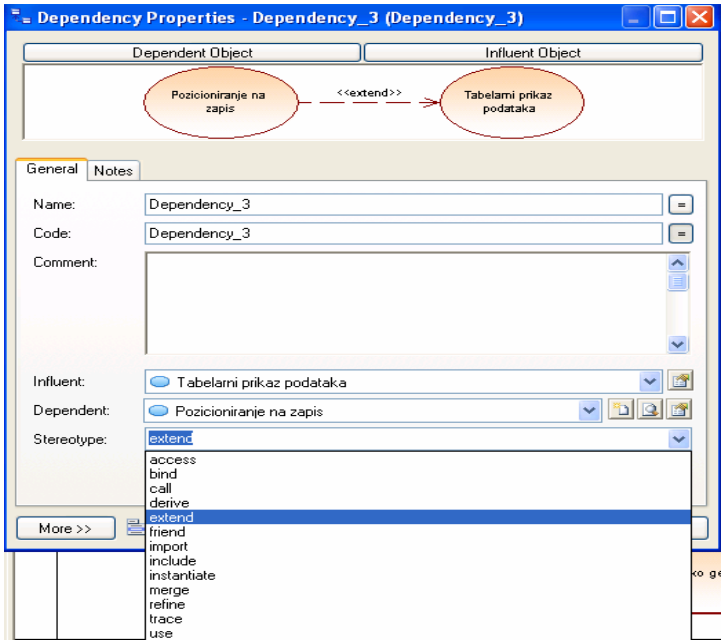
Slika 5.3.1. Dijagrami UML 2.0 u Power Designeru 15

Najvažniji UML modeli - dijagrami su:

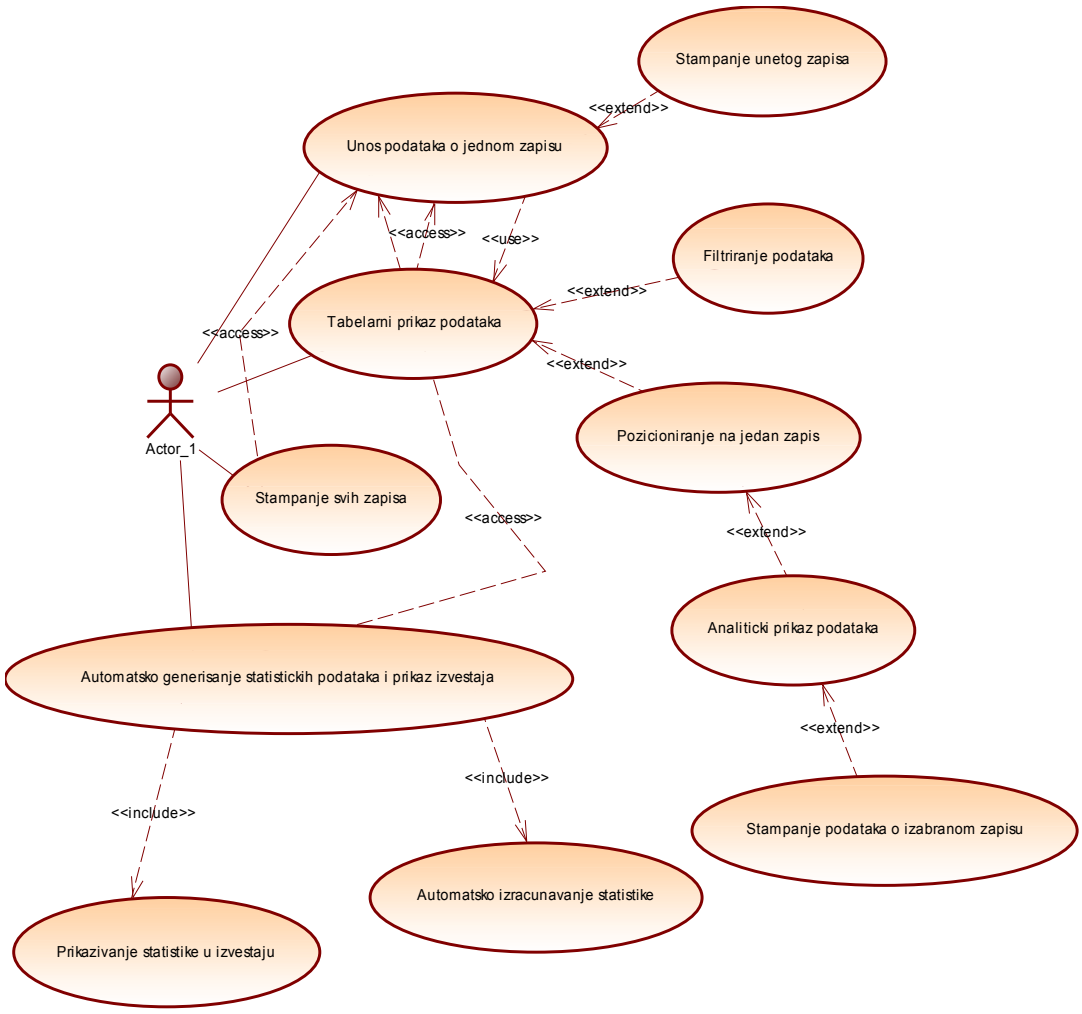
1. USE CASE diagram – prikazuje slučajeve korišćenja sistema (softverske funkcije)
2. Class diagram – prikazuje klase u realizaciji programa
3. Sequence diagram – prikazuje razmenu poruka klasa u implementaciji jednog slučaja korišćenja
4. Activity diagram – prikazuje algoritam izvršavanja (tok aktivnosti) slučaja korišenja ili metode neke klase
5. Statechart diagram – prikazuje moguća stanja objekta neke klase i način prelaska iz jednog u drugo stanje
6. Ccomponent diagram – prikazuje komponente i fajlove u strukturi izvorne i izvršne verzije nekog softverskog rešenja
7. Deployment diagram – prikazuje raspored komponenti softvera po računarima u mreži



Slika 5.3.2. Paleta sa alatima za rad sa dijagramom slučaja korišćenja



Slika 5.3.3. Podešavanje stereotipa – tipa zavisnosti slučaja korišćenja



Slika 5.3.4. Najčešći oblik zavisnosti slučaja korišćenja

Prilikom crtanja dijagrama slučaja korišćenja, najčešće ovalima predstavljamo softverske funkcije. Postavljamo najpre actor-e i povezujemo iz direktnim vezama sa slučajevima korišćenja za koje je predviđeno (npr. u tabeli preslikavanja definisano) da mogu da pristupaju. Neki slučajevi korišćenja se ne mogu direktno pokrenuti sa glavnog menija aplikacije. Nakon toga postavljamo veze zavisnosti između slučaja korišćenja. Najčešće vrste veza između slučajeva korišćenja su prikazane na slici 5.3.4.

Vrste veza koje se najčešće javljaju na dijagramima slučajeva korišćenja u oblasti informacionih sistema (po redosledu najčešćeg pojavljivanja):

1. access – pristup podacima
2. extend – proširivanje dodatnim funkcijama koje se mogu, ali ne moraju koristiti
3. use – odnos ravnopravnih slučajeva korišćenja, koji se oba mogu direktno pokrenuti sa menija od strane korisnika (actora), jedan slučaj korišćenja uslužno može pokrenuti u toku svog izvršavanja drugi slučaj korišćenja
4. include – jasno definisana softverska funkcija koja se nikad ne može samostalno i direktno pokrenuti, već predstavlja sastavni deo neke druge softverske funkcije, bez koje ova ne može da izvrši svoju funkcionalnost.

NAPOMENE:

- *Obratiti pažnju na smer strelice kod odnosa zavisnosti slučajeva korišćenja – strelica je okrenuta u smeru «čitanja rečenice», npr. SLUČAJ 2 proširuje SLUČAJ 1, strelica ide od slučaja 2 ka slučaju 1, ili SLUČAJ 1 sadrži slučaj 2 . strelica ide od slučaja 1 ka slučaju 2.*
- *Savetuje se radi preglednosti da se slučajevi koriscenja postavljaju odozgo nadole hronološkim redosledom izvršavanja, prateći sled izvršavanja odgovarajućih poslovnih procesa. Takođe, na vrh treba staviti slučajeve koriscenja koji se odnose na šifarnike, a zatim slučajeve koriscenja koji preslikavaju poslovne procese.*
- *Povezati sa actorima "jake" slučajeve korišćenja (one koji mogu direktno da se pokreću od strane korisnika)*
- *Postaviti zavisne ("slabe") - slučajevi korišćenja koji se ne mogu direktno pokrenuti od strane actora, vec preko nekog drugog slučaja korišćenja*
- *Postaviti veze između slučajeva korišćenja:*

  ODNOS SA "SLABIM"

  1. include - funkcionalnost bez koje ne moze da se posao završi, NIKADA se ne može direktno pokrenuti od strane korisnika sa menija

  2. extend - proširuje dodatnom funkcionalnošću osnovnu (nadovezuje, nastavlja), bez ove se može ("nice to have"), omogućena tek kada se osnovna funkcionalnost završi

  ODNOS 2 "JAKA"

  1. use - obe mogu da se pokrenu od strane korisnika, pokreće se uslužno da izvrši svoju aktivnost koja je nama potrebna u toku izvrsavanja drugog slučaja korišćenja

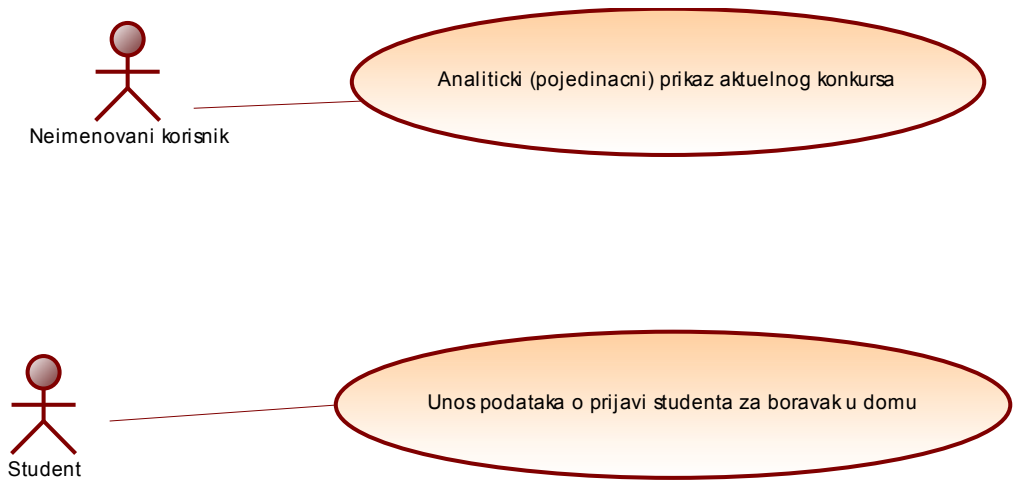
  2. access - obe mogu nezavisno da se pokrenu od strane korisnika, koriste se samo podaci kao rezultat rada
- *Može biti više veza međuzavisnosti između slučajeva korišćenja (najčešće 1 ili 2).*
- *Može se crtati više dijagrama slučajeva korišćenja, odnosno podeliti ih prema:*
  - *tipu aplikacije (npr. posebno sa WEB, c/s LAN, itd)*
  - *actor-ima (za svaki profil korisnika poseban dijagram)*

ZADATAK
Na osnovu spisaka softverskih funkcija iz tabele preslikavanja primitivnih procesa u softverske funkcije uraditi USE CASE dijagram.

REŠENJE:

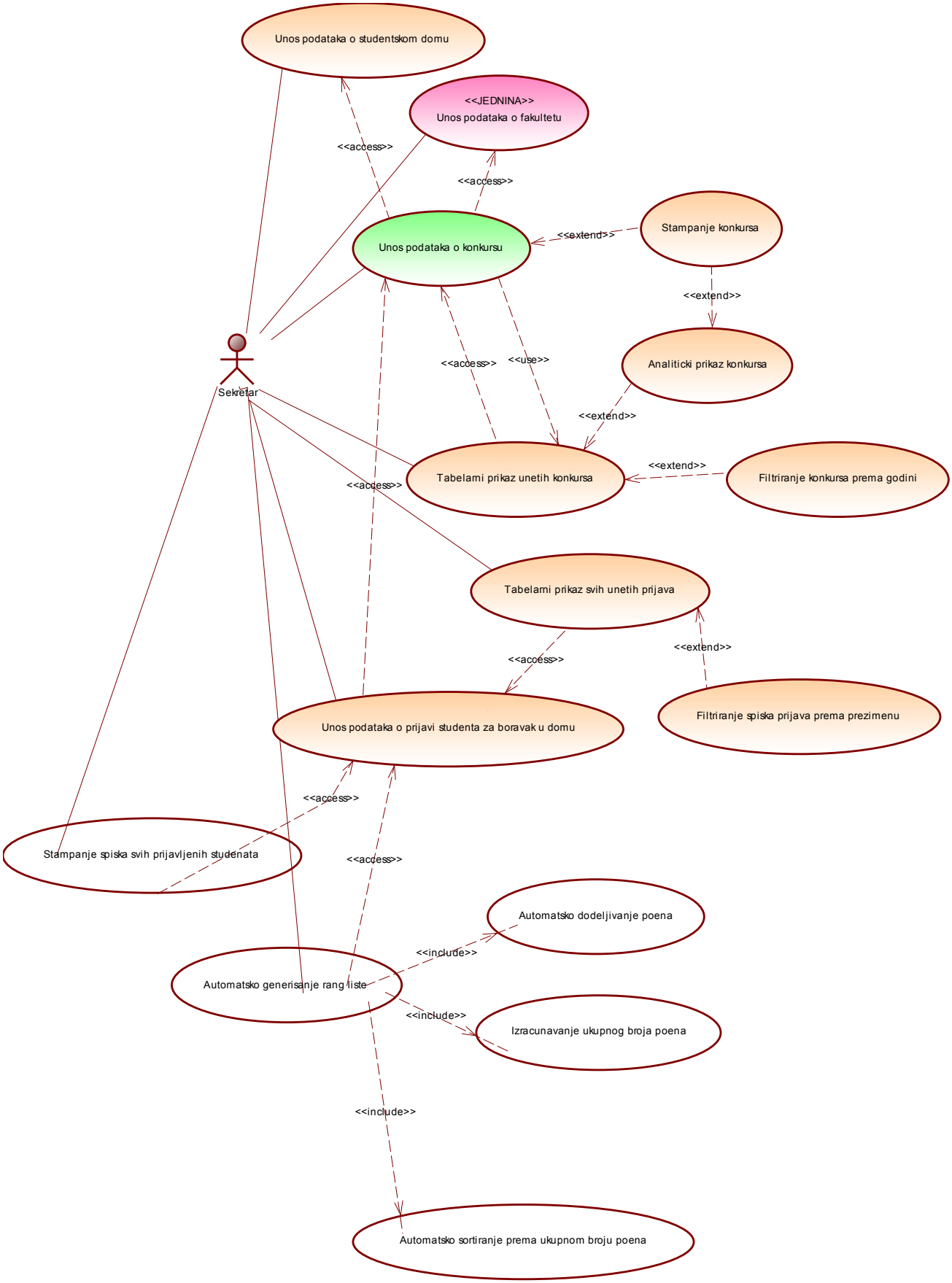
U nastavku je dato rešenje dijagrama slučaja korišćenja za primer studentskog doma koji je obrađen prethodnom tabelom (dakle nisu obrađeni svi poslovni procesi, pa ni skup slučajeva korišćenja nije potpun). U navedenom rešenju posebno su kreirana 2 dijagrama slučaja korišćenja – jedan za web i jedan za c/s LAN aplikaciju, koji samo ilustruju način crtanja dijagrama (nisu potpuna rešenja).

WEB aplikacija



Slika 5.3.5. Slučajevi korišćenja za web aplikaciju

Client server aplikacija u LAN okruženju data je na slici 5.3.6., gde su zastupljene sve najčešće vrste zavisnosti slučaja korišćenja (include, extend, use, access). Posebnom bojom i stereotipom («JEDNINA») je obeležen jedan slučaj korišćenja «Unos podataka o fakultetu» da bi se naglasilo da slučajevi korišćenja koji se odnose na unos podataka najčešće treba da se nazivaju u jednini. Drugi slučaj korišćenja je posebno obeležen (zelenom bojom) da bismo naznačili da je za njega urađena specifikacija, tj. detaljni tekstualni opis osobina slučaja korišćenja.



Slika 5.3.6. Slučajevi korišćenja za c/s LAN aplikaciju

5.4. Specifikacija slučaja korišćenja

Specifikacija slučaja korišćenja sastoji se o detaljnom opisu:

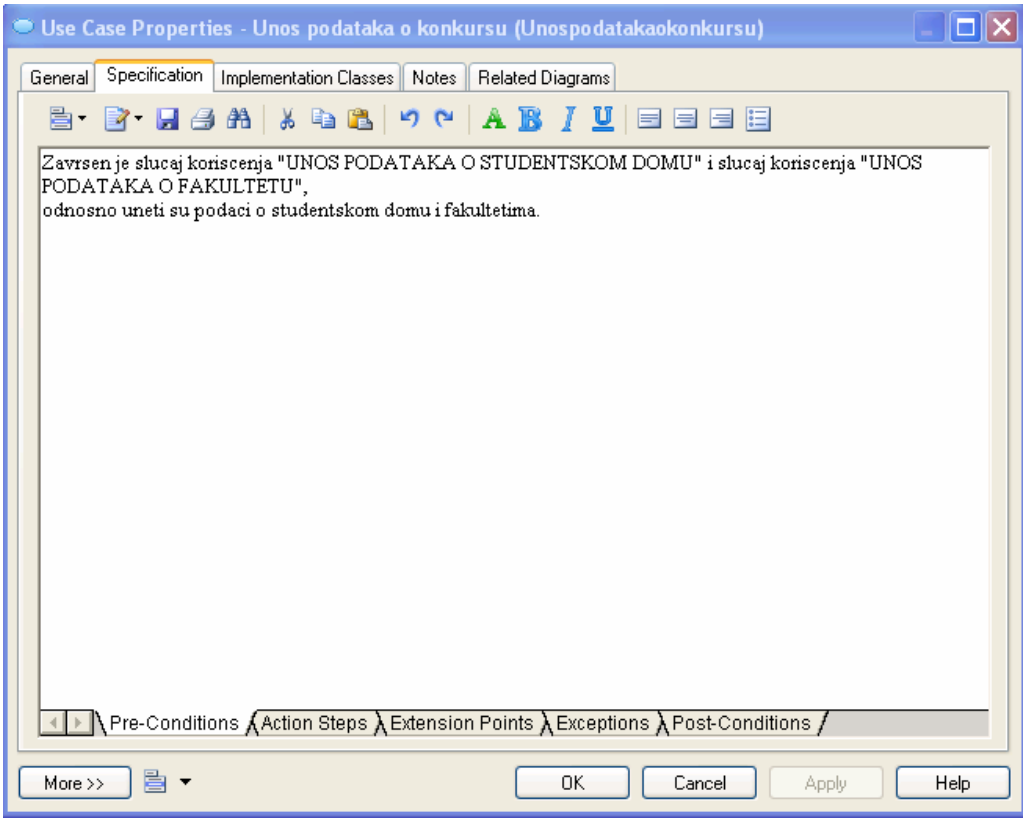
- Preduslova za uspešno izvršavanje (Preconditions) – koji slučajevi korišćenja moraju biti završeni, odnosno koji podaci na raspolaganju, tj. koje stanje programa mora biti zadovoljeno da bi mogao da se pokrene navedeni slučaj korišćenja
- Koraka akcije (Action steps) - Pseudo kod toka interakcije korisnika i sistema, koji sadrži uobičajene strukture sekvence, selekcije i iteracije, uz korišćenje službenih reči (AKO-ONDA-INAČE ili PONAVLJAJ DOK...) i linkove ka tačkama proširenja i obrade grešaka
- Tačke proširenja (Extension points) – Objašnjenje linkova za tačke proširenja koje opisuju pokretanje drugih slučajeva korišćenja (odnos «extend», «use», «include»)
- Obrada grešaka (Exceptions) – objašnjenje linkova za greške, odnosno načina kako sistem reaguje na greške. Odnosi se na unapred predviđene rizične situacije i mogućnost neispravnog rada programa, odnosno opisuju se načini planiranog reagovanja na rizične situacije. Potencijalne situacije se odnose na ulazne podatke (kada sistem koristi druge podatke iz drugih slučajeva korišćenja, npr. «access» odnos)
- Rezultate rada (Postconditions) – stanje u kom se program nalazi nakon završetka rada, odnosno rezultati rada

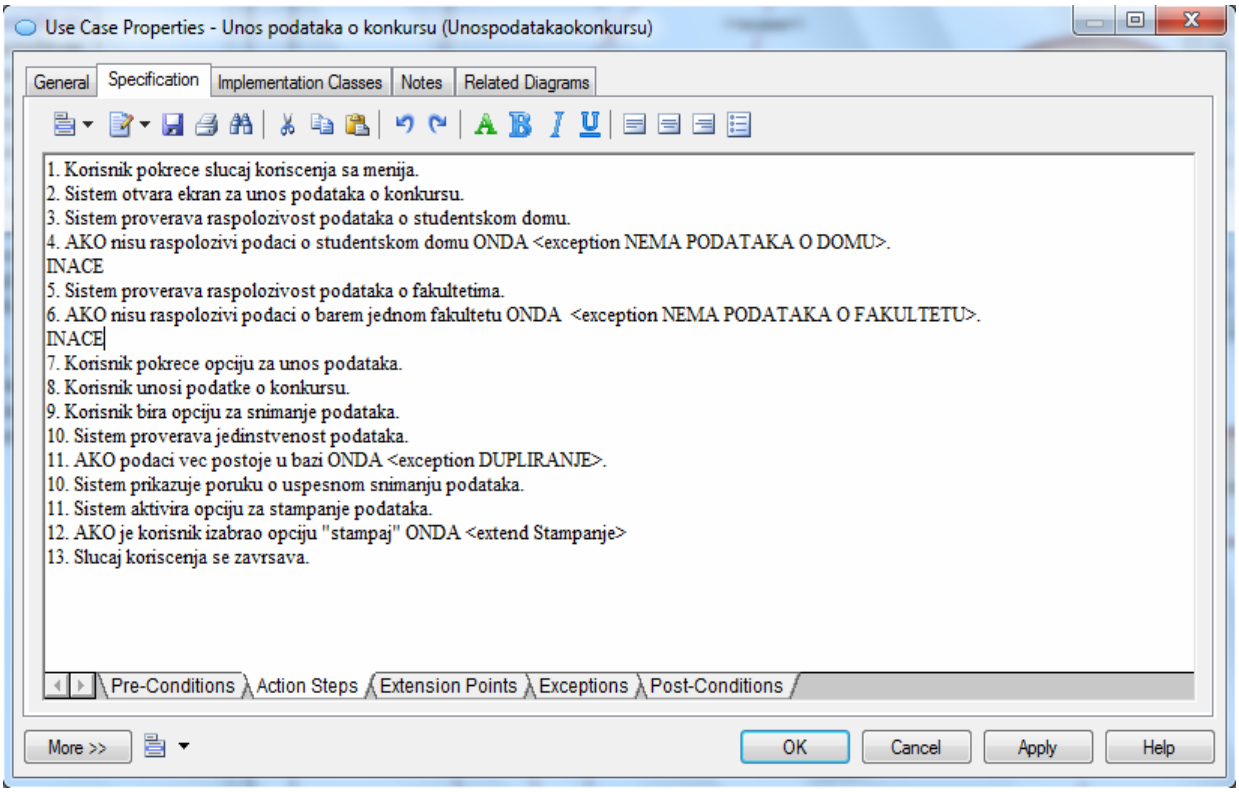
NAPOMENE:
1. U EXTENSION POINTS kartici specifijacije može biti naveden i use, include, a ne samo extend vrsta odnosa
2. DIJAGRAM mora biti usklađen sa specifikacijom - zato u toku izrade „action steps" pogledacemo sa kojim je sve slučajevima korišćenja posmatrani slučaj korišćenja povezan i sve te veze (osim „access" veze ako datom slučaju korišćenja pristupa) predstaviti u action steps i u extension point dati objašnjenja.

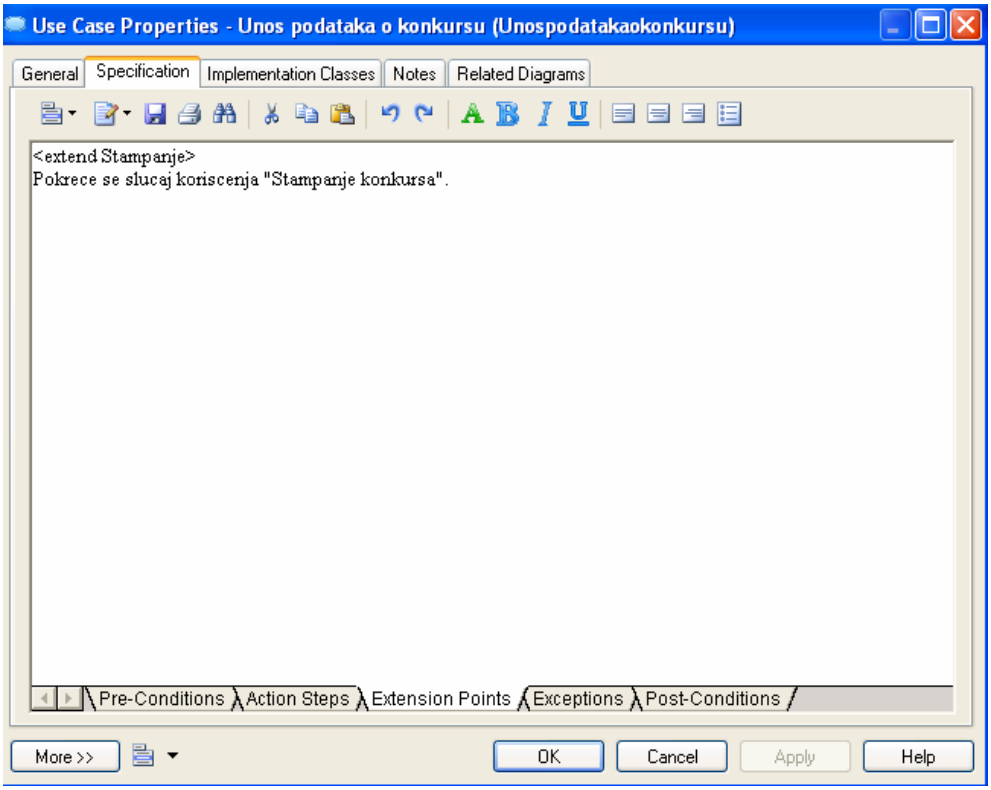
ZADATAK: Za dati slucaj koriscenja napisati tekstualnu specifikaciju kojom se opisuje tok izvrsavanja slucaja koriscenja.

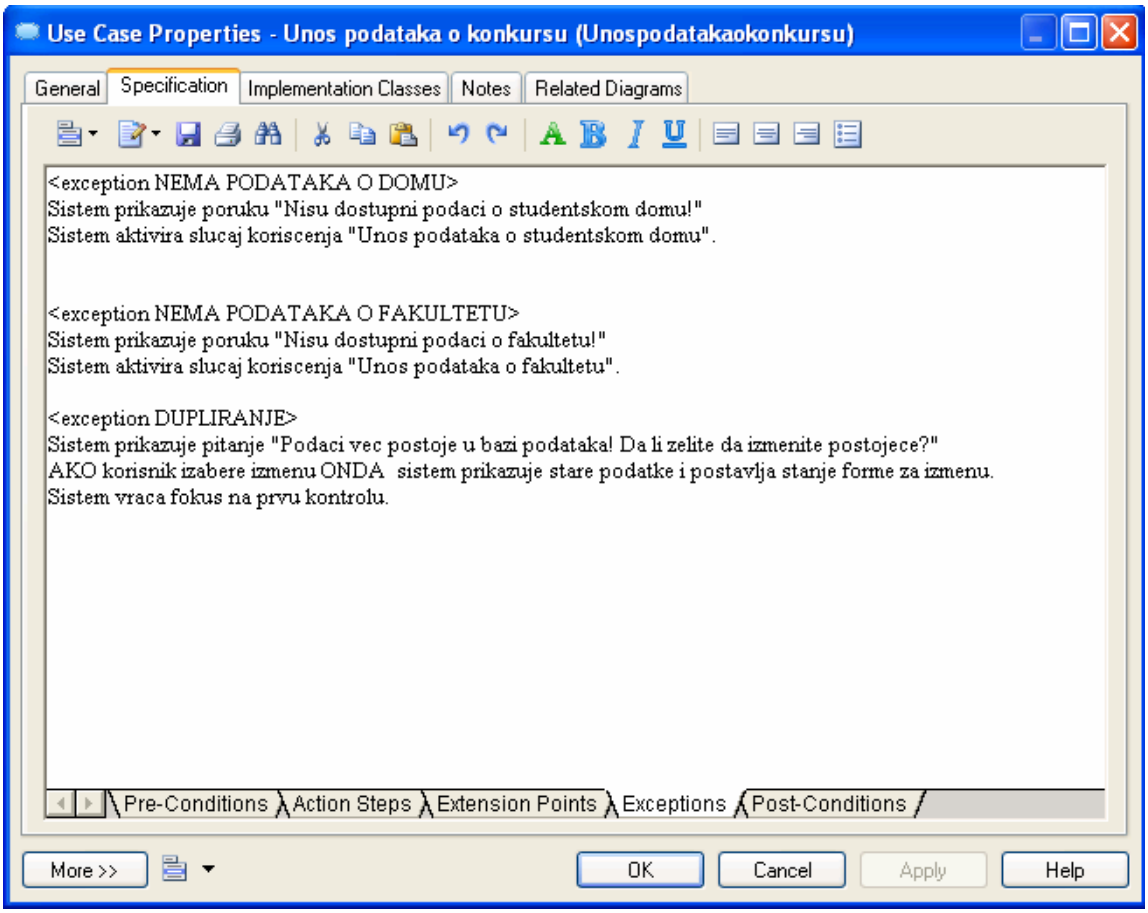
REŠENJE:

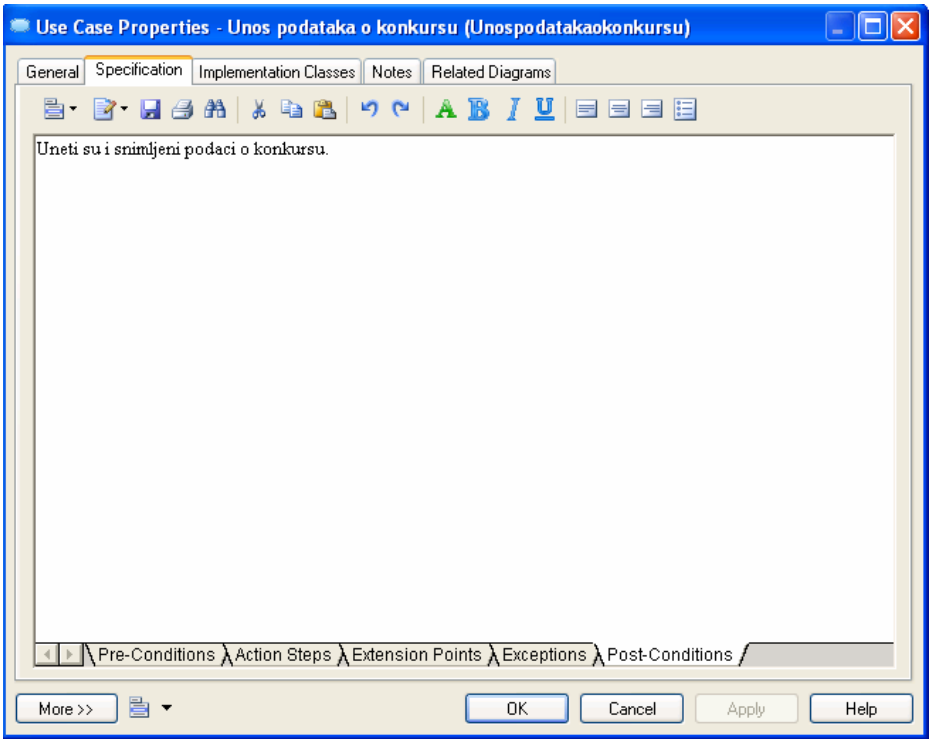
Za navedeni primer, slučaj korišćenja koji je izabran je «Unos podataka o konkursu».


Use Case Properties - Unos podataka o konkursu (Unospodatakaokonkursu)
General
Specification
Implementation Classes
Notes
Related Diagrams
Zavrsen je slucaj koriscenja "UNOS PODATAKA O STUDENTSKOM DOMU" i slucaj koriscenja "UNOS PODATAKA O FAKULTETU",
odnosno uneti su podaci o studentskom domu i fakultetima.
Pre-Conditions
Action Steps
Extension Points
Exceptions
Post-Conditions
More >>
OK
Cancel
Apply
Help


Use Case Properties - Unos podataka o konkursu (Unospodatakaokonkursu)
General
Specification
Implementation Classes
Notes
Related Diagrams
1. Korisnik pokrece slucaj koriscenja sa menija.
2. Sistem otvara ekran za unos podataka o konkursu.
3. Sistem proverava raspolozivost podataka o studentskom domu.
4. AKO nisu raspolozivi podaci o studentskom domu ONDA <exception NEMA PODATAKA O DOMU>.
INACE
5. Sistem proverava raspolozivost podataka o fakultetima.
6. AKO nisu raspolozivi podaci o barem jednom fakultetu ONDA <exception NEMA PODATAKA O FAKULTETU>.
INACE
7. Korisnik pokrece opciju za unos podataka.
8. Korisnik unosi podatke o konkursu.
9. Korisnik bira opciju za snimanje podataka.
10. Sistem proverava jedinstvenost podataka.
11. AKO podaci vec postoje u bazi ONDA <exception DUPLIRANJE>.
10. Sistem prikazuje poruku o uspesnom snimanju podataka.
11. Sistem aktivira opciju za stampanje podataka.
12. AKO je korisnik izabrao opciju "stampaj" ONDA <extend Stampanje>
13. Slucaj koriscenja se zavrsava.
Pre-Conditions
Action Steps
Extension Points
Exceptions
Post-Conditions
More >>
OK
Cancel
Apply
Help


Use Case Properties - Unos podataka o konkursu (Unospodatakaokonkursu)
General
Specification
Implementation Classes
Notes
Related Diagrams
<extend Stampanje>
Pokrece se slucaj koriscenja "Stampanje konkursa".
Pre-Conditions
Action Steps
Extension Points
Exceptions
Post-Conditions
More >>
OK
Cancel
Apply
Help


Use Case Properties - Unos podataka o konkursu (Unospodatakaokonkursu)
General
Specification
Implementation Classes
Notes
Related Diagrams
<exception NEMA PODATAKA O DOMU>
Sistem prikazuje poruku "Nisu dostupni podaci o studentskom domu!"
Sistem aktivira slucaj koriscenja "Unos podataka o studentskom domu".
<exception NEMA PODATAKA O FAKULTETU>
Sistem prikazuje poruku "Nisu dostupni podaci o fakultetu!"
Sistem aktivira slucaj koriscenja "Unos podataka o fakultetu".
<exception DUPLIRANJE>
Sistem prikazuje pitanje "Podaci vec postoje u bazi podataka! Da li zelite da izmenite postojece?"
AKO korisnik izabere izmenu ONDA sistem prikazuje stare podatke i postavlja stanje forme za izmenu.
Sistem vraca fokus na prvu kontrolu.
Pre-Conditions
Action Steps
Extension Points
Exceptions
Post-Conditions
More >>
OK
Cancel
Apply
Help



5.5. REŠENI ZADACI

ZADATAK:

1. Na osnovu datog opisa posla i stabla procesa realizovati tabelarni prikaz primitivnih procesa i njihovo preslikavanje u odgovarajuće softverske funkcije
2. Softverske funkcije predstaviti dijagramom slučajeva korišćenja
3. Odabrani slucaj koriscenja predstaviti specifikacijom

SISTEM: Mala turistička agencija

OPIS POSLA

Da bi turistička agencija mogla da radi, mora imati licencu za rad. Za licencu se prijavljuje ministarstvu turizma i podnosi zahtev. Nakon dobijanja licence, turistička agencija može da radi. S obzirom da je ovde reč o maloj turističkoj agenciji (dalje MTA), ona sklapa ugovor o saradnji sa drugim većim turističkim agencijama (dalje VTA), koje imaju svoje ponude i direktne kontakte sa turističkim destinacijama (hotelima i slično), tako da je ona njihov predstavnik. Od tih drugih turističkih agencija dobija katalog ponude na sajmu turizma, koji se organizuje jednom godišnje, obično u februaru. Kada građanin dođe sa zahtevom za organizovanje putovanja, od MTA dobija kataloge ponuda. Nakon razgledanja kataloga, građanin se odlučuje za neku destinaciju. Građanin ostavlja lične podatke, podatke drugih osoba koje sa njim putuju i podatke o željenoj destinaciji i trajanju boravka, kao i načinu transporta. MTA kontaktira VTA tako što šalje dokument "najava puta" sa svim prethodno uzetim podacima, kako bi zahtevala predračun za uplatu. Nakon prijema predračuna, MTA kontaktira građanina da ga obavesti o prispeću predračuna. Građanin usmeno dobija podatke potrebne za uplatu i uplaćuje novac na žiro račun MTA. Nakon uplate, donosi MTA priznanicu o uplati. MTA šalje podatke o izvršenoj uplati (na osnovu priznanice o uplati) VTA i zahteva od VTA da im ona pošalje vaučer. Nakon prijema vaučera, MTA obaveštava građanina o prispeću vaučera.Najkasnije 7 dana pre puta građanin od turističke agencije dobija vaučer, koji predstavlja dokaz da je uplatio troškove boravka izvršena i da može da ide na put.

STABLO PROCESA

0.Osnovna delatnost – svrha postojanja: Organizovanje boravka putnika na željenoj destinaciji

1. Pripremna faza
   1.1. Izdavanje kataloga potencijalnom klijentu (ili bolje: Informisanje klijenta o ponudama)
   1.2. Prijem prijave za izbor željene destinacije od klijenta
   1.3. Slanje dokumenta „najava puta” ili Najavljivanje puta
   1.4. Zahtevanje predracuna
   1.5. Prijem predracuna

2. Izvršna faza
   2.1.Obavestavanje klijenta o uplati
   2.2. Prijem uplate od strane klijenta
   2.3.Slanje podataka o uplati VTA i zahtev za vaučerom
   2.4.Prijem vaucera

3. Zavrsna faza
3.1.Obaveštavanje klijenta o prispeću vaučera
3.2. Izdavanje vaučera klijentu

REŠENJE:

1. ZADATAK

Rešenje prvog zadatka (samo za 3 odabrana primitivna procesa) dato je u sledećoj tabeli. Kao kriterijum za selekciju sofverskih funkcija po prioritetima, odabrali smo drugi pristup – pokrivanje fokusa poslovnog procesa.

Primitivni poslovni proces	Radna uloga	Softverska funkcija * 1. po prioritetu	Actor	Softverska funkcija * 2. po prioritetu	Actor	Softverska funkcija * Preduslov za uspesno izvrsavanje	Actor	Tip softvera (c/s LAN, WEB)
Izdavanje katalog a klijentu	radnik na šalteru agencije	n/a (lično)	Klijent (građanin)			Dizajn kataloga Štampanje kataloga	Radnik u štampariji	Softver u stampariji
		Tabelarni prikaz katalog a Tabelarni prikaz ponuda iz kataloa Filtriranje katalog a prema godini Filtriranje ponude iz katalog a prema drzavi, mestu, godini ponude	c-s LAN: radnik na šalteru agencije, klijent u agenciji WEB: Neimenovani korisnik			Unos podataka o drzavi Unos podataka o mestu Unos podataka o destinaciji Unos podataka o ponudi za destinaciju Unos podataka o katalogu	Sekretar – komercijalista u agenciji	c-s LAN WEB
		Prikaz PDF sa katalogom na sajtu	WEB: Neimenovani korisnik			Kreiranje PDF sa katalogom za sajt Postavljanje PDF sa katalogom na sajt	Sekretar – komercijalista u agenciji	WEB
		Slanje PDF sa katalogom e-mailom	Sekretar – komercijalista u agenciji					Internet (e-mail servis)

NAPOMENA:

- *Samo za ovaj jedan primitivni proces vidimo da postoji više varijanti rešenja. Bitno je naglasiti da ne moramo uvek imati softversku funkciju za prvi prioritet. Kada je u pitanju štampanje kataloga, katalog se daje lično u papirnom obliku i nema potrebe za softverskom podrškom (pišemo n/a).*
- *Takođe, povodom preduslova možemo videti da je direktni preduslov „Unos podataka o katalogu", koji se zapravo sastoji od „Unosa podataka o ponudama u katalogu". Njihovi*

preduslovi su takođe tu napisani, a to je „Unos podataka o destinaciji", kojoj mora da prethodi „unos podataka o mestu" i „unos podataka o državi".

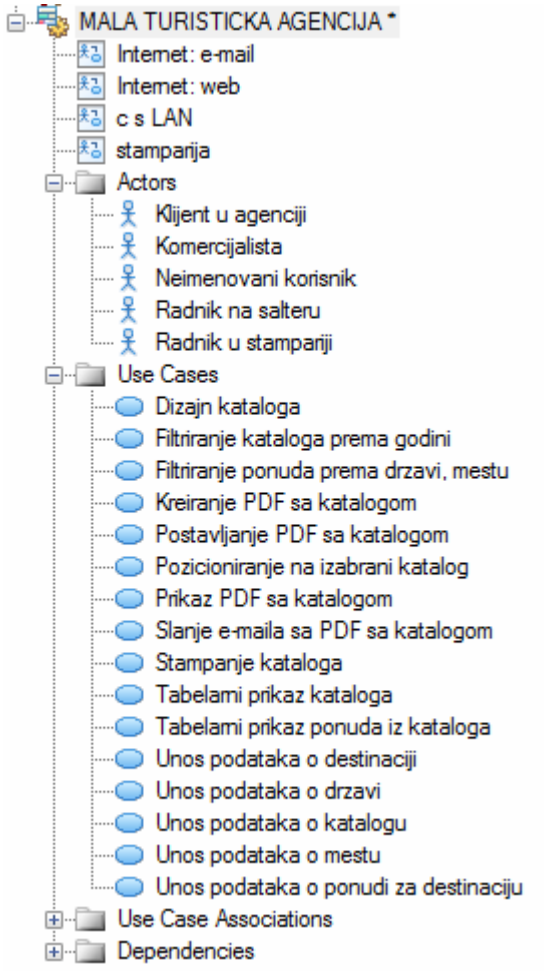
Primitivni poslovni proces	**Radna uloga**	**Softverska funkcija * 1. po prioritetu**	**Actor**	**Softverska funkcija * 2. po prioritetu**	**Actor**	**Softverska funkcija * Preduslov za uspesno izvrsavanje**	**Actor**	**Tip softvera (c/s LAN, WEB)**
Prijem prijave za izbor željene destinacije od klijenta	radnik u agenciji	Unos podataka o prijavi građanina i izboru željene ponude	radnik na šalteru agencije	Tabelarni prikaz unetih prijava Filtriranje prijava prema prezimenu Pozicioniranje na željenu prijavu Štampanje izabrane prijave Štampanje unete prijave	radnik na šalteru agencije	Povodom unosa podataka o građaninu: Unos podataka o državi Unos podataka o mestu Povodom unosa podataka o željenoj ponudi: Unos podataka o ponudi	Sekretar – komercijalista u agenciji	c/s LAN
		Unos podataka o klijentu (registracija korisnika)	Neimenovani korisnik (ovim unosom postaje klijent)			Povodom unosa podataka o građaninu: Unos podataka o državi Unos podataka o mestu	Sekretar – komercijalista u agenciji	web
		Unos podataka o izboru ponude	Klijent			Povodom unosa podataka o željenoj ponudi: Unos podataka o ponudi	Sekretar – komercijalista u agenciji	web
Slanje dokumenta „najava puta"	radnik u agenciji	Slanje e-maila sa PDF „najava puta"	Sekretar – komercijalista u agenciji	Tabelarni prikaz svih poslatih e-mailova Filtriranje e-mailova prema datumu, prezimenu klijenta ili ciljnoj e-mail adresi		Unos podataka o prijavi građanina i izboru željene ponude Automatsko kreiranje PDF "najava puta"	Sekretar – komercijalista u agenciji	c-s LAN, Internet (e-mail servis)

Napomena:

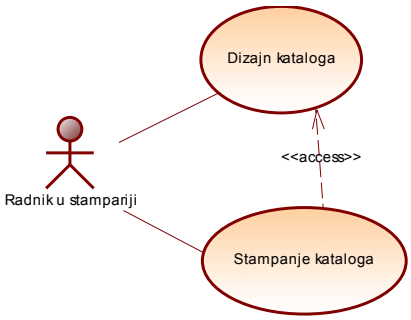
- *Za svaku softversku funkciju bitno je precizno definisati naziv, ali i detaljno definisati sve preduslove da bi se softverska funkcija iz prvog prioriteta mogla izvršiti, jer moramo specificirati programski zadatak programeru.*
- *Takođe, bitno je uzeti u obzir i krajnjeg korisnika i ponuditi mu dovoljnu funkcionalnost softvera za normalan rad.*

2. Zadatak - USE CASE dijagrami

U nastavku ce biti prikazani samo use case dijagrami koji se odnose na prvu tabelu iz 1. Zadatka.

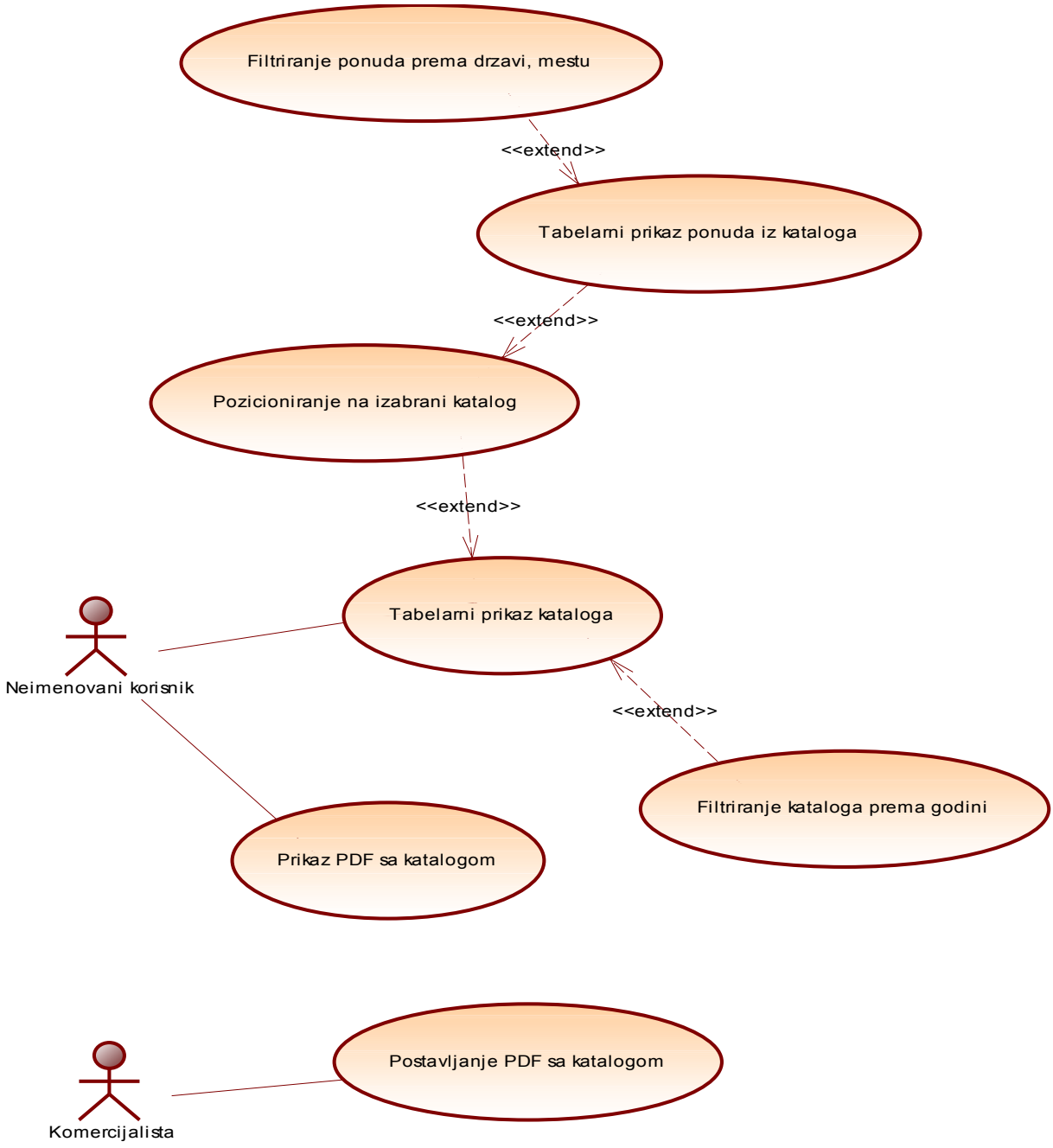


SOFTVER U STAMPARIJI

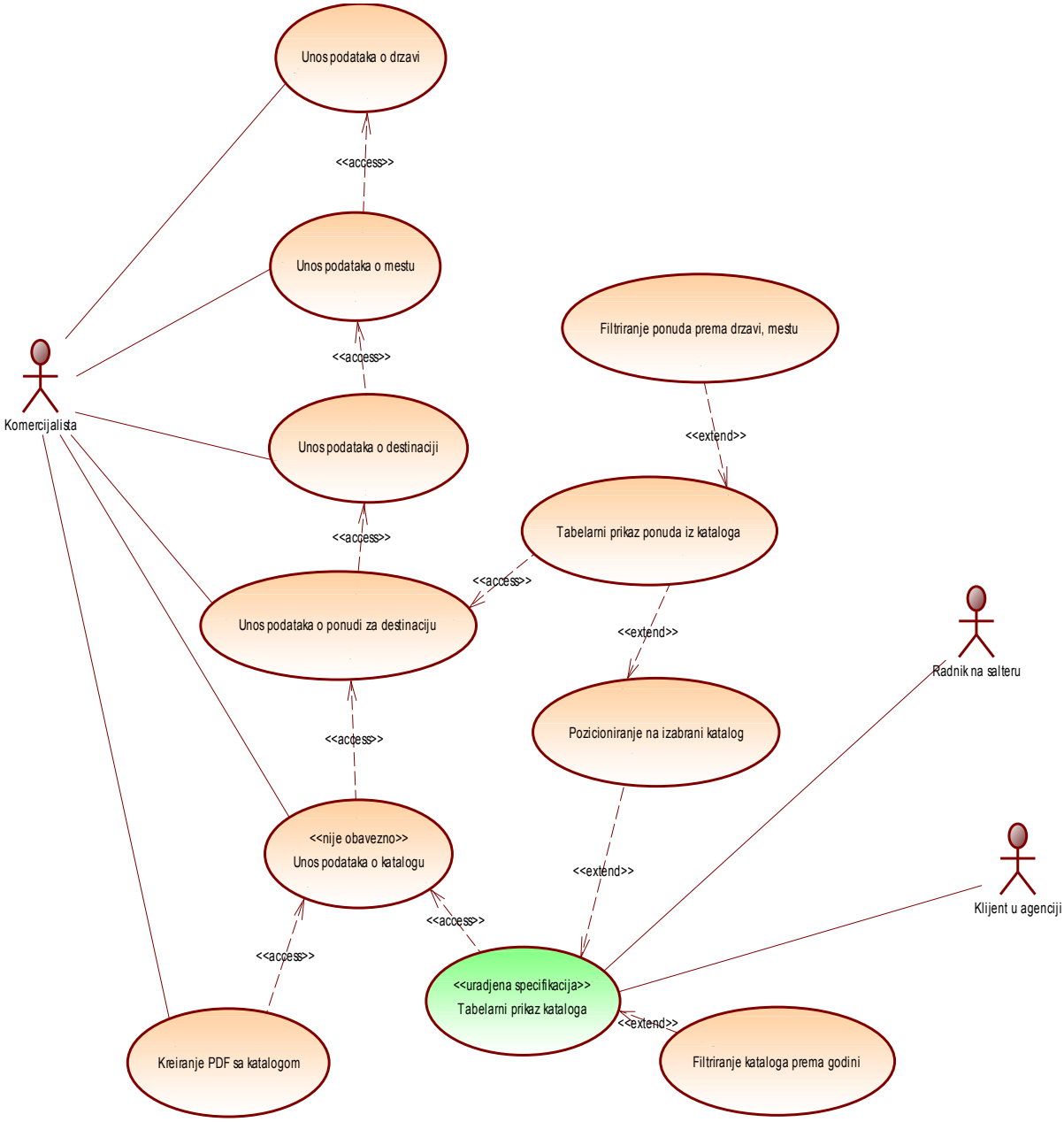


INTERNET : E-MAIL

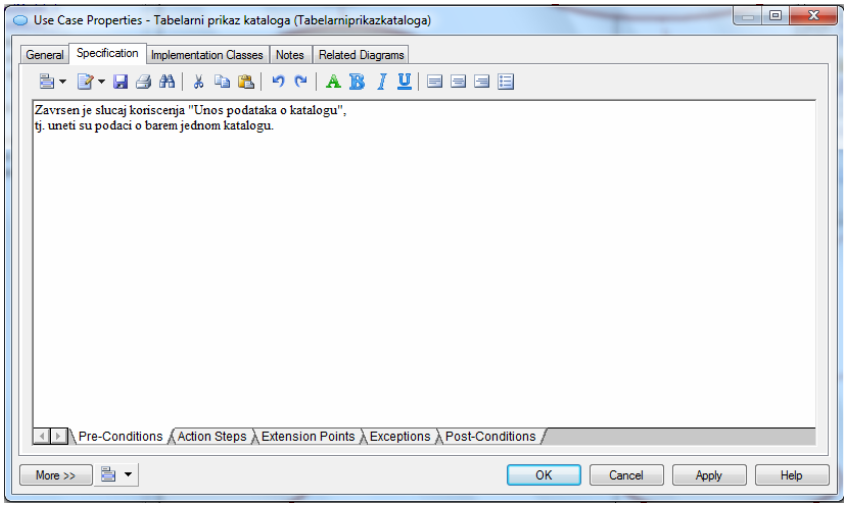
INTERNET : web

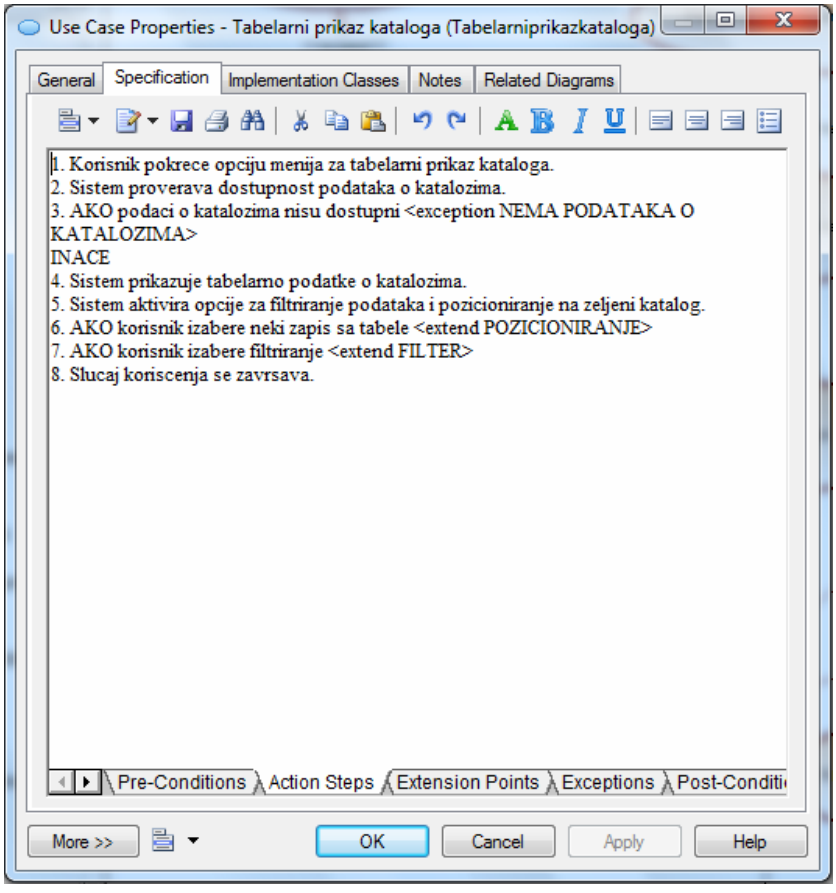


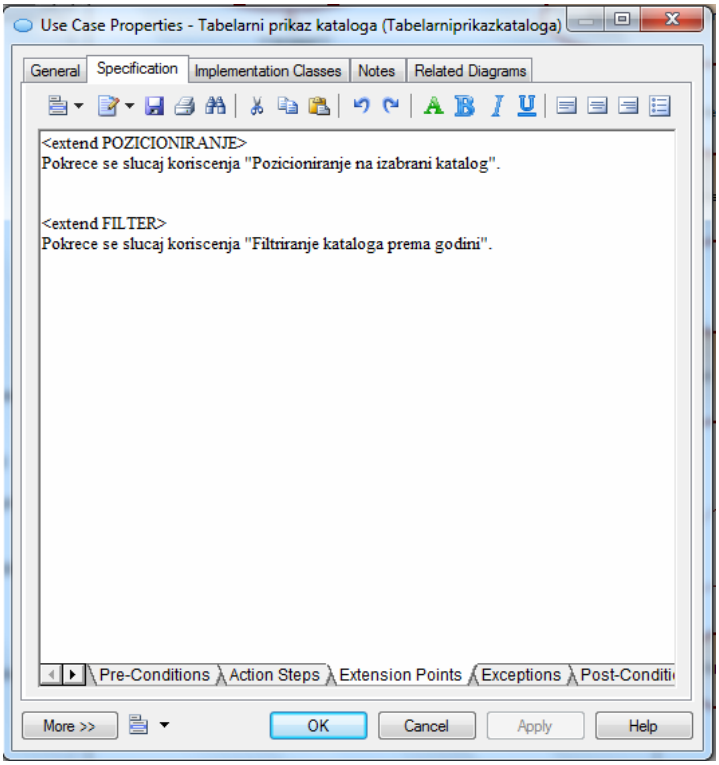
CLIENT SERVER APLIKACIJA U LAN OKRUZENJU:

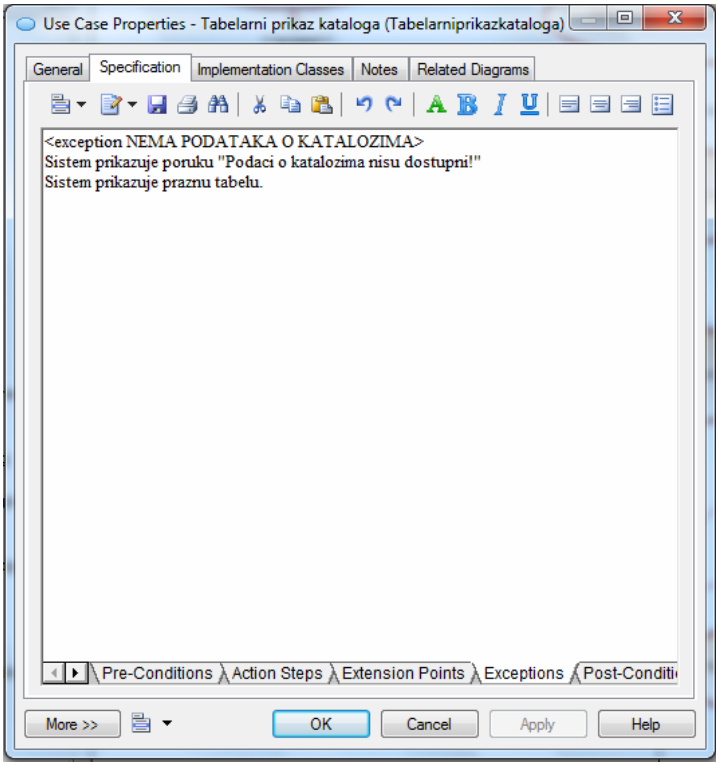


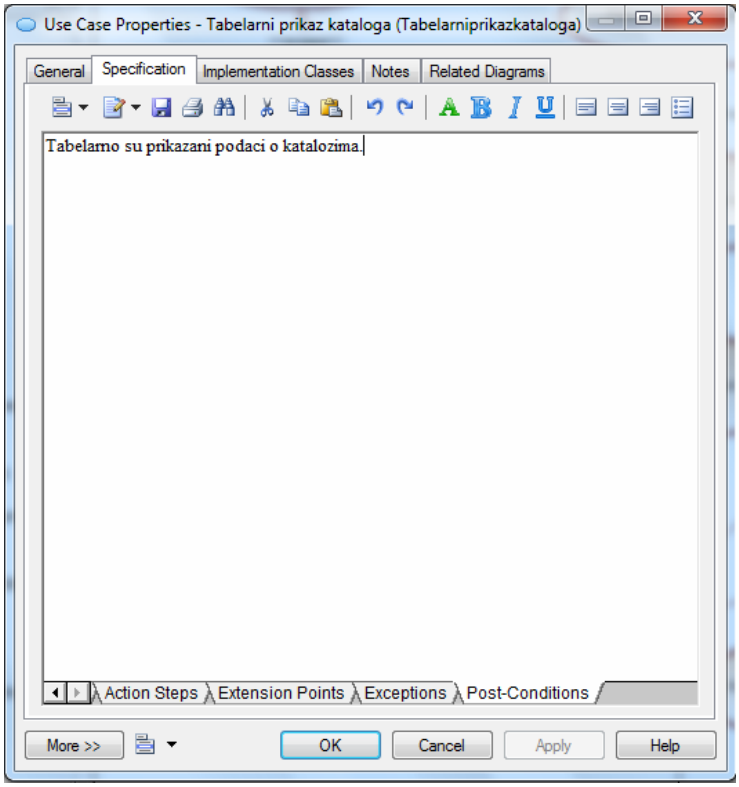
3. Zadatak – specifikacija izabranog slucaja koriscenja






Use Case Properties - Tabelarni prikaz kataloga (Tabelarniprikazkataloga)
General
Specification
Implementation Classes
Notes
Related Diagrams
<extend POZICIONIRANJE>
Pokrece se slucaj koriscenja "Pozicioniranje na izabrani katalog".
<extend FILTER>
Pokrece se slucaj koriscenja "Filtriranje kataloga prema godini".
Pre-Conditions
Action Steps
Extension Points
Exceptions
Post-Conditi
More >>
OK
Cancel
Apply
Help


Use Case Properties - Tabelarni prikaz kataloga (Tabelarniprikazkataloga)
General
Specification
Implementation Classes
Notes
Related Diagrams
<exception NEMA PODATAKA O KATALOZIMA>
Sistem prikazuje poruku "Podaci o katalozima nisu dostupni!"
Sistem prikazuje praznu tabelu.
Pre-Conditions
Action Steps
Extension Points
Exceptions
Post-Conditi
More >>
OK
Cancel
Apply
Help


Use Case Properties - Tabelarni prikaz kataloga (Tabelarniprikazkataloga)
General
Specification
Implementation Classes
Notes
Related Diagrams
Tabelarno su prikazani podaci o katalozima.
Action Steps
Extension Points
Exceptions
Post-Conditions
More >>
OK
Cancel
Apply
Help

6. KREIRANJE MODELA PODATAKA

6.1. KREIRANJE KONCEPTUALNOG MODELA PODATAKA

6.1.1. Pristupi i metode za kreiranje konceptualnog modela podataka

Postoji nekoliko osnovnih pristupa konceptualnom modelovanju:

- PRISTUP: Direktno (sveobuhvatno) modelovanje – sve specifikacije poslovnih procesa [5] i svi zahtevi korisnika koji se odnose na sve profile korisnika i sve potrebe za pogledima u bazi podataka [4] se prikupljaju i udružuju u jedinstvenu listu zahteva na osnovu kojih se kreira zajednički kompleksni model podataka.
  - METODA: Korišćenje dizajn obrazaca – paterna *[5]* - Paterni su opšta rešenja klase problema.
  - METODA: Gramatička analiza teksta opisa posla [3] – imenice su kandidati za entitete, glagoli za poveznike
  - METODA: Normalizacija skladišta podataka [7]
  - METODA: Kreiranje entiteta dodeljivanjem atributa iz rečnika podataka entitetima [7]
- PRISTUP: Sekvencijalno (parcijalno) modelovanje [4, 5] – kreiranje podmodela i njihova integracija.
  - METODA: Gramatička analiza teksta specifikacije slučaja korišćenja, slično kao i kod gramatičke analize teksta opisa posla - imenice su kandidati za entitete, glagoli za poveznike
  - METODA: Analiza XML schema [5] – Svaka XML shema se transformiše u podmodel konceptualnog modela
  - METODA: Parcijalno modelovanje eksternih šema (pogleda)[4] i njihova integracija database.
  - Kreiranje podmodela [5] za svaki primitivni proces i njihova integracija
- PRISTUP: Iterativno poboljšanje konceptualnih modela [1] kroz niz iteracija, dok se ne dobije najpogodniji dizajn
- PRISTUP: Konceptualno modelovanje u svakoj od faza razvoja informacionog sistema [2]

U okviru pristupa konceptualnom modelovanju u svakoj fazi razvoja informacionog sistema, naredna tabela prikazuje neke faze razvoja i odgovarajuće materijale i metode konceptualnog modelovanja koja bi se u odgovarajućoj fazi mogla primeniti [7]:

FAZA RAZVOJA INFORMACIONOG SISTEMA	**REZULTAT FAZE I ULAZ U PROCES KONCEPTUALNOG DIZAJNA**	**METODA KONCEPTUALNOG MODELOVANJA**
Specifikacija zahteva korisnika	Tekst specifikacije zahteva korisnika	Gramatička analiza teksta
Upoznavanje sa poslovnim procesima	Tekstualni opis poslovnog procesa	Gramatička analiza teksta
Modelovanje poslovnih procesa	Primitivni proces	Kreiranje podmodela za svaki primitivni proces
	Tok podataka	Normalizacija toka podataka
	Skladište podataka	Normalizacija skladišta
	Atributi	Dodeljivanje atributa entitetima
Dizajn sistema	Actor-i (profili korisnika)	Podmodel za svakog actora (pogled) i integracija
	Slucajevi koriscenja	Podmodel za svaki slucaj koriscenja i integracija

Nijedan od navedenih metoda ne daje kompletan i korektan model. Samo primenom integracije metoda možemo dobiti dovoljno kompletan i korektan model, ali ako bismo morali da biramo između metoda, najpogodnija bi bila dekompozicija ili normalizacija skladišta podataka [7].

U nastavi vežbi na predmetima koji se bave razvojem informacionih sistema primenjuje se već duži niz godina pristup integracije navedenih metoda, kroz osnovni pristup postepenog razvoja modela kroz više iteracija. Naredna tabela prikazuje integrativni pristup (proširen dodatnom proverom potpunosti u odnosu na slučajeve korišćenja, u odnosu na [7]) primeni više metoda u kreiranju zajedničkog kompletnog i korektnog konceptualnog modela.

FAZA KONCEPTUALNOG MODELOVANJA	MATERIJAL	METODA
1. radna verzija	Atributi iz rečnika podataka iz modela procesa (dijagrama toka podataka)	Dodeljivanje atributa entitetima i kreiranje entiteta na osnovu atributa
2. radna verzija	Tekst opisa posla	Gramatička analiza teksta
3. radna verzija	Skladišta podataka sa dijagrama toka podataka	1) Normalizacija 2) Analiza sintaksne strukture
4. radna verzija integracija	1,2,3 radna verzija	INTEGRACIJA
5. radna verzija	4. radna verzija	Provera i korekcija grešaka, normalizacija
FINALNA VERZIJA	5. radna verzija	Provera potpunosti – -kreiranje podmodela za svaki primitivni proces -Kreiranje podmodela za svaki slucaj koriscenja

NAPOMENA:
U okviru pristupa da primena pojedinačnih metoda ne daje kompletna rešenja, već je potrebna primena različitih metoda i njihova integracija, moguće je navedeni pristup primenjivati na dva načina: 1) kreirati zasebne modele primenom pojedinačnih metoda ili 2) redom primenjivati jednu za drugom metodu, čime bi se postepeno zajednički model dograđivao i dopunjavao. U nastavnom procesu se drugi način pokazao kao efikasniji.

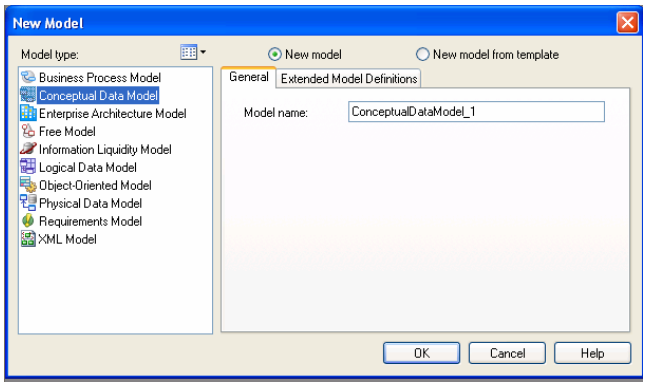
Nakon kreiranja konceptualnog modela, u CASE alatu može biti direktno transformisan u relacioni model. Ipak, u tom procesu ne mogu biti automatski uočene i otklonjene greške normalizacije [6], pa je normalizacija dobijenog relacionog modela neophodna. Normalizacija relacija (odnosno tabela u relacionoj bazi podataka) *[1] [5]* predstavlja proces kreiranja relacionog modela podataka, gde svaka relacija (tabela) treba da bude u odgovarajućoj normalnoj formi (barem u 3 normalnoj formi), čime se izbegavaju anomalije ažuriranja. Konačno, nakon normalizacije potrebno je postići operativnu efikasnost modela u fizičkoj realizaciji, tako da se vrše dodatna podešavanja [1]. Ta podešavanja uključuju dizajn fizičke baze podataka – indeksiranje, denormalizacija, horizontalno i vertikalno particionisanje itd. radi unapređenja performansi baze podataka, tj. ubrzavanja rada sql upita i slično.

LITERATURA

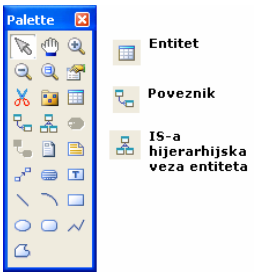
[1] R. Elmasri, S.B Navathe: *Fundamentals of Database Systems*, 5th edition, Pearson International Edition, 2007
[2] E. J. Naiburg, R.A. Maksimchuk: UML for Database Design, Addison-Wesley, 2001
[3] P. Mogin, I. Lukovic: *Principles of databases*, University of Novi Sad, School for technical sciences, Novi Sad, Serbia, 1996.
[4] P. Mogin, I. Lukovic, M. Govedarica: *Principles of database projecting*, University of Novi Sad, School for
technical sciences, Novi Sad, Serbia, 2000.
[5] B. Lazarevic, Z. Marjanovic Z., N.Anicic, S. Babarogic: *Databases*, University of Belgrade, School for organizational sciences, Belgrade, Serbia, 2003.
[6] D.B. Bock: *Entity-Relationship Modelling and Normalization Errors*, Journal of Database Management, 1997.
[7] Kazi Lj, Kazi Z, Radulović B, Letić D, Bhatt M: Applying Integration Of Conceptual Data Modelling Methods Within Information System Development: A Case Study, Metalurgia International, ISSN 1582 - 2214 vol. XVII no. 6 (2012) pp. 67-74

6.1.2. Kreiranje konceptualnog modela u CASE alatu

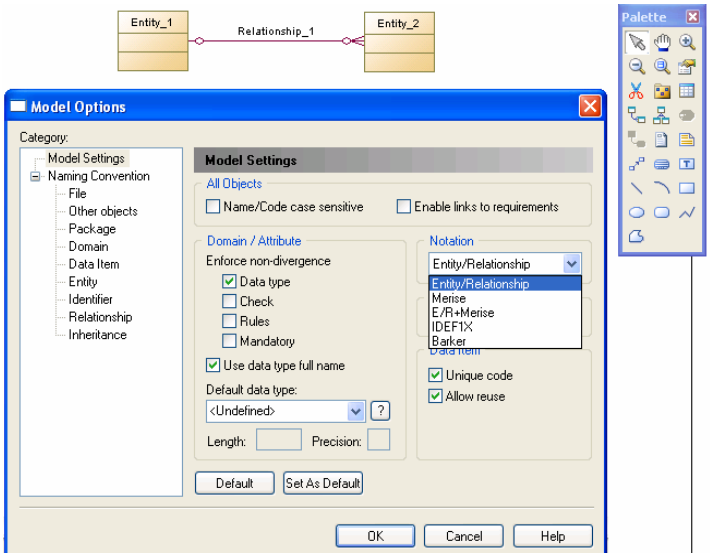
U CASE alatu Power Designer 15, pokretanjem opcije „New Model" dobijamo dijalog prozor za kreiranje novog modela biramo "Conceptual data model".



Nakon pokretanja, dobijamo paletu sa alatima:



U okviru alata možemo izabrati podvrstu konceptualnog modela pomoću opcije Tools – Model options.



NAPOMENE U VEZI NAZIVA ENTITETA:

1. Entiteti treba da imaju nazive u jednini.

2.Nastojimo da entiteti imaju nazive kao suštinski objekti i događaji u sistemu, a ne dokumenti. Razlozi za opredeljivanje za stabilne elemente sistema u struktuiranju buduće baze podataka:

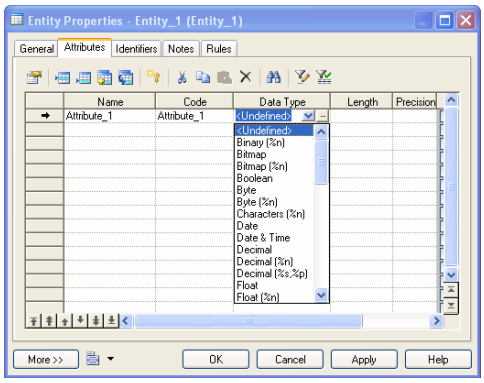
- *Dokumenti menjaju strukturu i naziv, pa bi promene strukture baze podataka bile učestalije.*
- *Ako bismo imali orjentaciju na dokumente, isti elementarni podatak mogao bi da se nađe u više entiteta. Osnovni princip je minimizacija redundanse – jedan elementarni podatak treba čuvati samo u jednom entitetu, a povezivati relacijom sa svim ostalim entitetima gde je potreban.*

Zato podrška dokumentima ostavljamo da bude u srednjem poslovnom sloju klasa višeslojne arhitekture.

3. Za isti poslovni domen može biti više jednako korektnih modela podataka. Mogu se razlikovati po nivou apstrakcije (uopštavanja). Nastojimo da što više uopštavamo modele, izdvajajući šifarnike, s ciljem da se postigne podrška olakšanom kasnijem održavanju,

odnosno budućim izmenama i dopunama u bazi podataka. Ipak, u uopštavanju ne smemo previše uopštiti model, jer se time mogu „izgubiti" važni specifični elementarni podaci.
4. Jedan od najvažnijih principa je da se podaci beleže samo jednom – minimizacija redundanse. Nastojimo da obezbedimo da se podaci beleže u trenutku i na mestu gde su nastali. Ipak, iz razloga bezbednosti i optimizacije rada, uvodi se kontrolisana redundansa, tj. ponavljanje podataka, odnosno istoimeni podaci se mogu naći u različitim tabelama baze podataka. Primer: Tabele „Kandidat za upis na fakultet" i „Student" imaju iste podatke, kao sto su Prezime, Ime, Adresa... ali ipak ćemo ove podatke čuvati u 2 različite tabele.

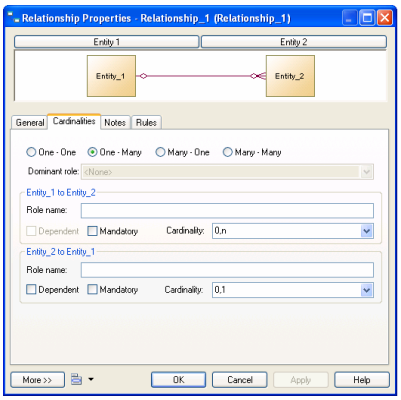
U okviru entiteta, najvažnije podešavanje se odnosi na atribute – naziv, tip podatka, dužinu:



NAPOMENA u vezi PREUZIMANJA ELEMENTARNIH PODATAKA IZ BPM modela:
S obzirom da je u prethodnoj fazi modeliranja poslovnih procesa popunjen rečnik podataka, tj. imamo kompletnu specifikaciju informacionih potreba sistema kroz skup potrebnih elementarnih podataka, postoji mogućnost u CASE alatu da se preuzmu ti elementarni podaci za korišćenje u konceptualnom modelu. U CASE alatu Power Designer u okviru istog Workspace potrebno je najpre otvoriti Business Process model sa rečnikom podataka, a zatim kreirati prazan konceptualni model u istom Workspace. Kada smo pozicionirani u okviru Business process modela, biramo opciju Tools, Data Export i zatim kroz odgovarajuće dijalog prozore preuzimamo podatke u konceptualni model.

NAPOMENA U VEZI NAZIVA ATRIBUTA:
Nastojimo da atribute nazivamo kao atomarne vrednosti (jednina u nazivu, naziv kao prost elementarni podatak, a ne struktura).

Što se tiče relacija, najvažnija podešavanja se odnose na kardinalitet (globalni tip veze i donje granice preslikavanja – Mandatory) i identifikacionu zavisnost (Dependent).



Svaka relacija se sastoji od 2 preslikavanja, odnosno 2 uloge:
1. Odnos prvog entiteta prema drugom
2. Odnos drugog entiteta prema prvom

Podešavanja kardinaliteta se rade u 2 koraka:
1. Podešavanje generalnog kardinaliteta (gornje granice preslikavanja): 1-1, 1-M, M-1, M-M
2. Podešavanje donje granice kardinaliteta – osobina Mandatory (kada je podešena, donja granica je 1)

Što se tiče relacija, odnosno odnosa između entiteta, razlikujemo:
1. IS-a hijerarhiju (poseban tip poveznika sa palete)

2. ostale tipove veza – rešava se podešavanjem osobina zavisnosti u Properties:
    a. Jak – slab – Entitet koji je slab tip je Dependent prema drugome entitetu
    b. Jak – Jak – nema dependent odnosa
    c. Veze sa gerundom (objekat-veza) – entitet je dependent barem prema druga 2 entiteta sa kojima ostvaruje relacije

NAPOMENA:

1. Kod određivanja kardinaliteta posebno treba obratiti pažnju na:

1.1.1. Odnos M:M – najčešće relacija sa ovim kardinalitetom ima važna obeležja i može već u konceptualnom modelu da se predstavi entitetom koji će imati ta važna obeležja

1.1.2. Odnos 1:1 – postavljamo pitanje da li ima opravdanja da eniteti koji su u odnosu 1:1 budu 2 odvojena entiteta, ili da budu integrisani u jedan entitet

2. Kod određivanja kardinaliteta postavljamo 2 pitanja:

2.1. BROJNI ODNOSI realan svet i baza podataka

2.1.1. koja je donja i gornja granica kardinaliteta za preslikavanje prvog ka drugome entitetu u realnom svetu – kada bismo određivali brojne odnose realnih predmeta i pojava. Najčešće su kardinaliteti realnog sveta suviše restriktivni. PRIMER: u realnom svetu odnos mesta i zemlje je takav da ne može postojati zemlja bez barem jednog mesta (pa je donja granica 1 u kardinalitetu odnosa zemlja --- (1,M) --- mesto).

2.1.2. Zbog prethodno opisanog ograničenja, razmatramo drugi korak – razmatranje donje granice u kontekstu baze podataka . PRIMER: da li je neophodno da u bazi podataka bude obavezno da prilikom unosa podataka u tabelu zemlja moramo obavezno uneti barem jedno mesto ... u ovom slučaju ne moramo, znači, donja granica bi mogla biti 0. Ipak, u primeru RAČUN-STAVKE RAČUNA ili generalno MASTER-DETAIL najčešće mora postojati barem jedna stavka, dakle i donja granica bi bila 1.

- *Konačni kardinalitet ostaje definisan u kontekstu baze podataka.*

2.2. BROJNI ODNOSI trenutak ili promene u vremenu

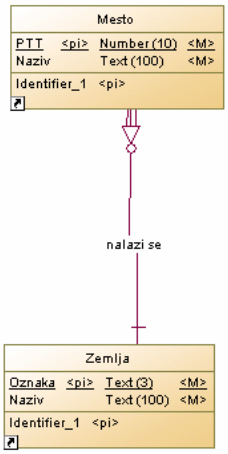
2.2.1. Brojne odnose možemo utvrditi u kontekstu jednog trenutka, kada se, ukoliko bi došlo do promena, stari podaci brišu i zamenjuju novim. PRIMER: Zaposleni radi na jednom radnom mestu. Dakle, Radno mesto prema Zaposlenom je 1:M tip relacije.

2.2.2. Brojne odnose možemo posmatrati i u vremenu, kada postoji potreba da se beleži istorija promena. PRIMER: U dužem vremenu posmatrano, Zaposleni može da radi i na više radnih mesta. Svaka promenu radnih mesta se beleži i čuva u bazi podataka. Dakle, Zaposleni prema Radnom mestu i Radno mesto prema Zaposlenom je u odnosu M:M. Tu se odmah pitamo da li postoje važni atributi i utvrđujemo da je važno beležiti datum dodeljivanja određenog radnog mesta.

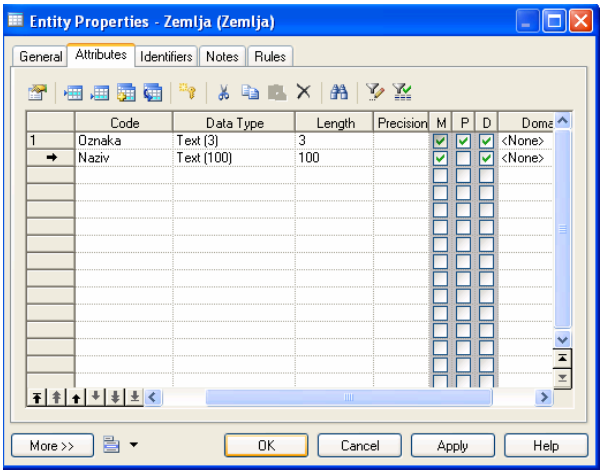
- Konačni kardinalitet ostaje u skladu sa potrebama – ako se radi o sistemu u kojem je važno beležiti istorijat promena, odrediće se kardinalitet u skladu sa drugim pristupom, a ako je to sistem gde nije važno pratiti sve promene, dovoljno je odrediti kardinalitet u skladu sa „jednim trenutkom". Ipak, generalno ostaje napomena da je važno imati stav da je potrebno čuvati istorijat promena podataka, a ne dizajnirati takvu bazu podataka da će se stari podaci gubiti usled promena i uvođenja novih podataka.

PRIMERI:

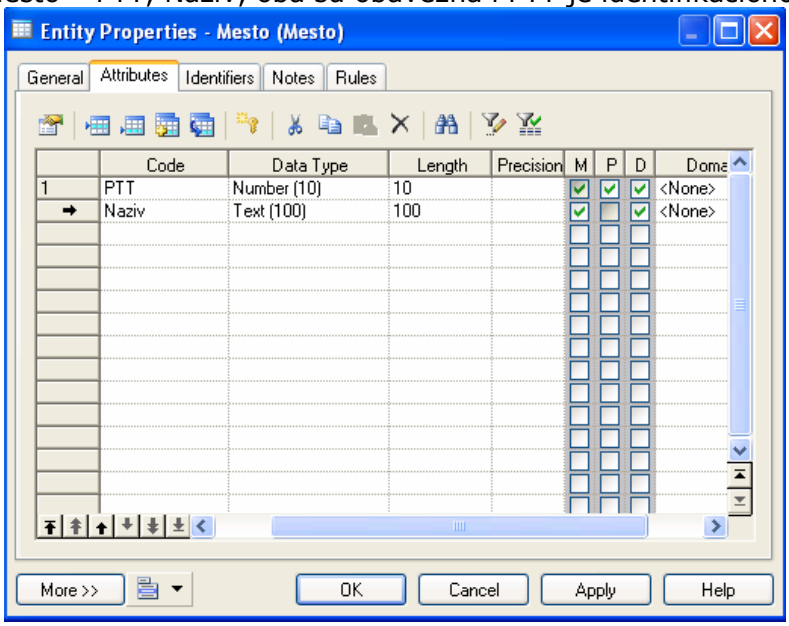
1. PRIMER - primer odnosa Zemlja – Mesto (JAK – SLAB)



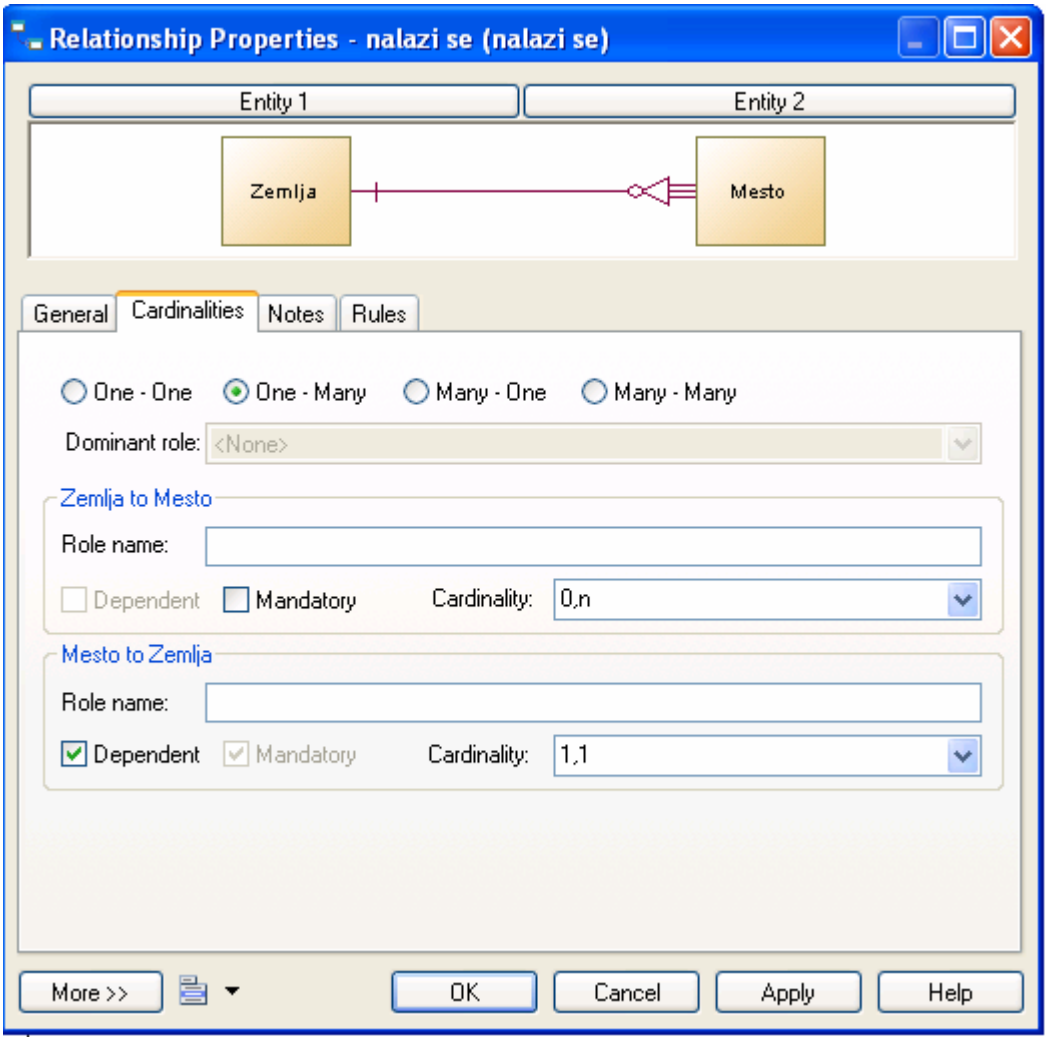
1. Atributi tabele Zemlja – oznaka, naziv, gde su oba atributa obavezna za unos (not null, odnosno Mandatory – oznaka M), a oznaka je primarni kljuc – oznaka P



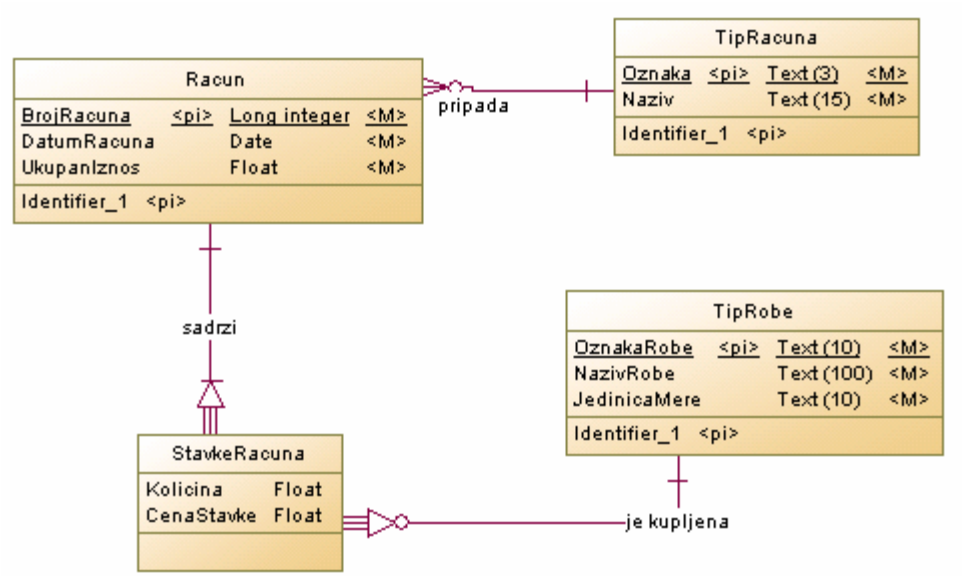
2. Atributi tabele Mesto – PTT, Naziv, oba su obavezna i PTT je identifikaciono obeležje



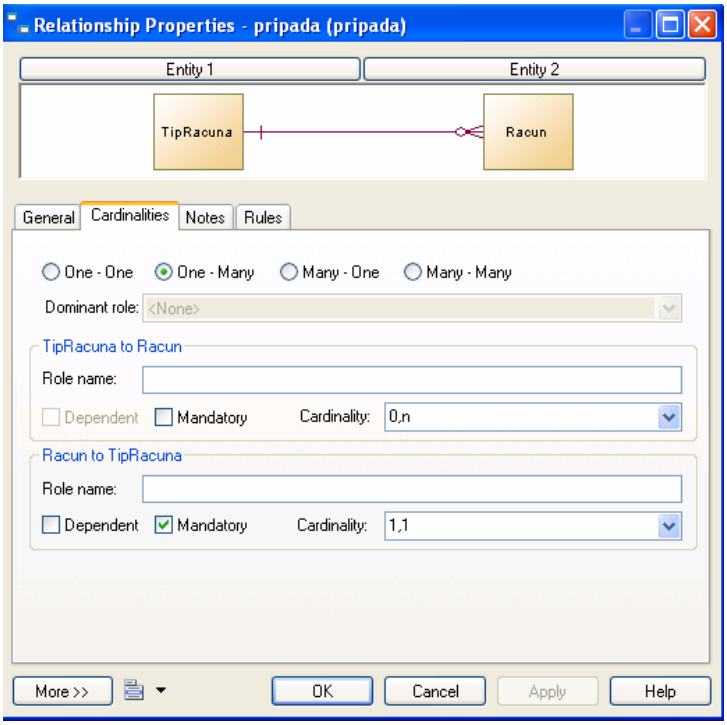
3.Podešavanje relacije – u ovom slučaju mesto je slab objekat u odnosu na zemlju, identifikacono je zavisan (Dependent)



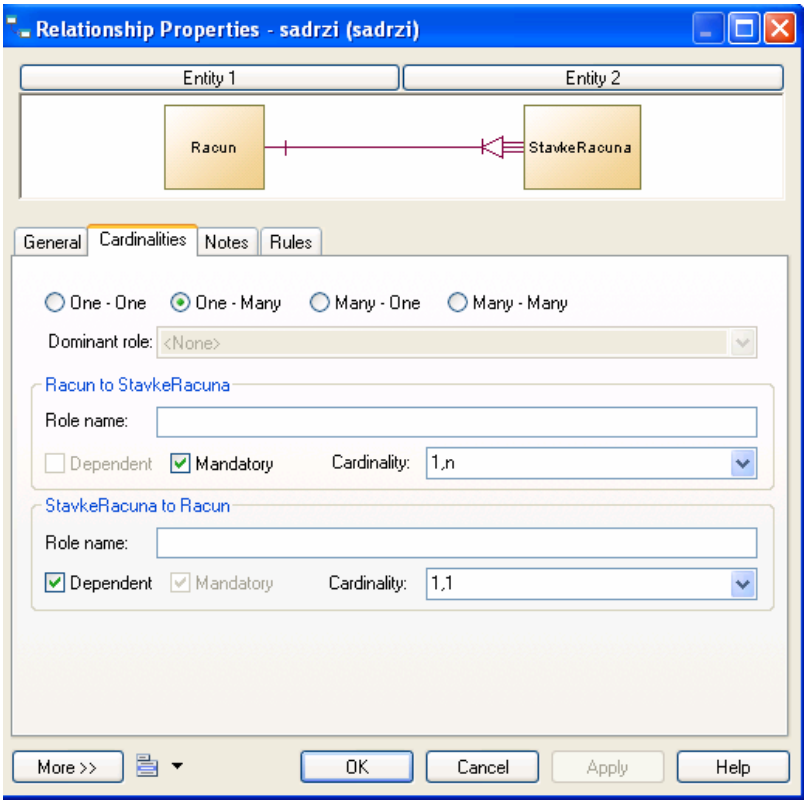
2. primer - Racun – tip racuna – stavke racuna – roba (odnos 2 jaka i gerund)



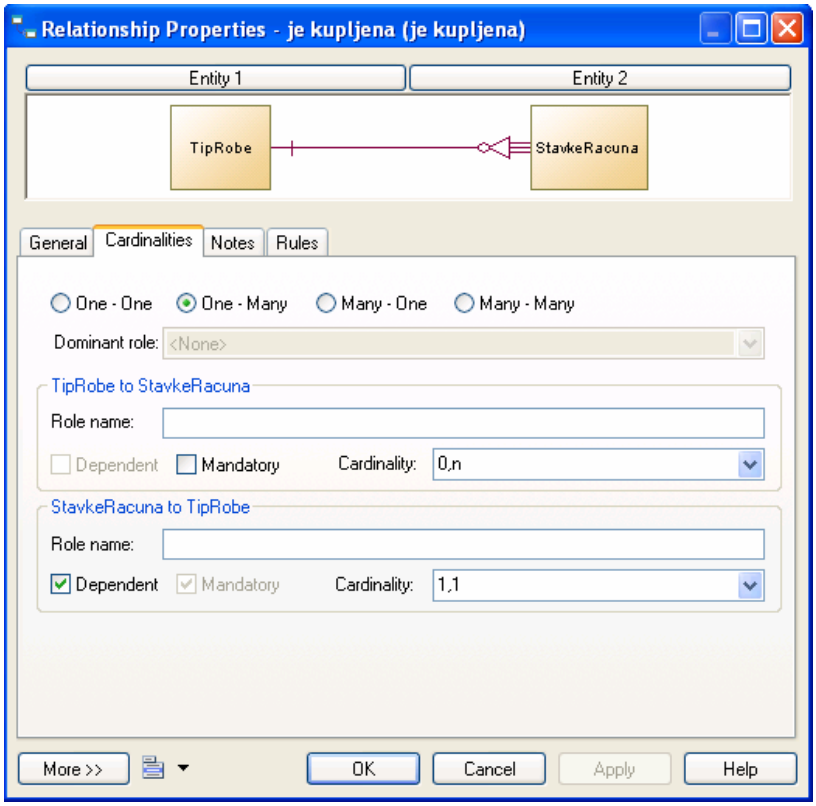
Odnos 2 jaka entiteta – nije podešen Dependent:



Za gerund Stavke računa – identifikaciona zavisnost od 2 druga entiteta – od Racuna:

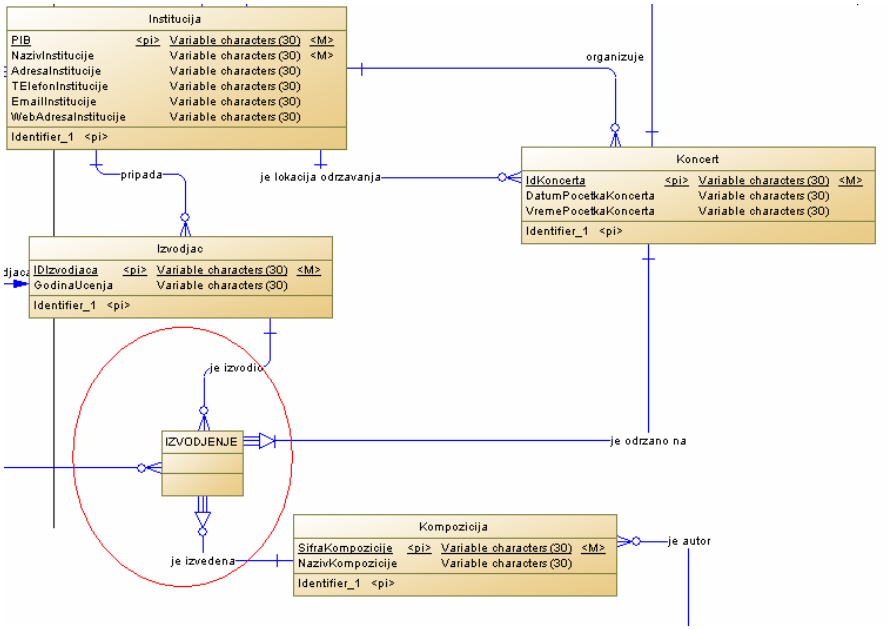


kao i od Tipa Robe:



3. primer – IZVOĐENJE kompozicije na koncertu

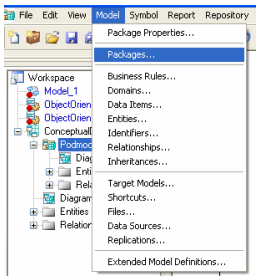
NAPOMENA:
Kod određivanja GERUNDA, odnosno objekta –veze, najčešće se javlja situacija da sam poveznik treba da ostvaruje relacije sa entitetima. Tada se bira opcija „change to entity" i automatski se prevodi u entitet. U takvoj situaciji postavlja se pitanje da li takav entitet ima važnih atributa koje treba postaviti. Takođe, nastojimo da prethodni glagol sada transformišemo u glagolsku imenicu. Primer: Učenik izvodi Kompoziciju na koncertu. Dakle, poveznik izvodi postaje IZVOĐENJE između učenika, kompozicije i koncerta kao entiteta. Takođe, početna identifikaciona zavisnost koja bi se javila pri automatskom prevođenju poveznika u entitet se može promeniti. U navedenom primeru, IZVOĐENJE je identifikaciono zavisno od koncerta i kompozicije, a ne, kao bi bilo inicijalnom transformacijom, od učenika i kompozicije.



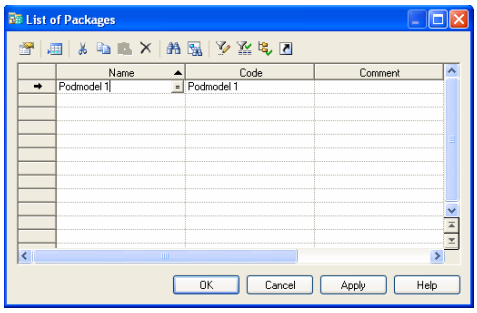
6.1.3. Kreiranje podmodela konceptualnog modela podataka

U okviru provere kompletnosti konceptualnog modela, za svaki slučaj korišćenja ili svaki primitivni proces može da se uradi po jedan podmodel. Ako je moguće sve primitivne procese i sve slučajeve korišćenja „pokriti" odgovarajućim entitetima, to znači i da je konceptualni model potpun.

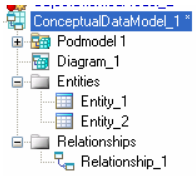
U okviru CASE alata Power Designer podmodeli konceptualnog modela mogu se kreirati izborom opcije Model - Packages.



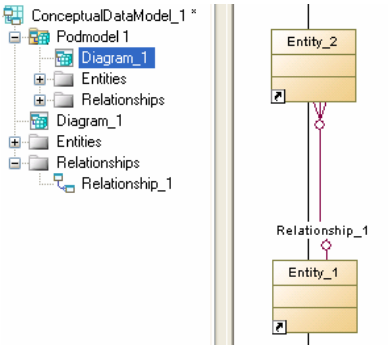
Dobijamo listu „Paketa":



odnosno na „Browseru":



Prevlačenjem entiteta, odnosno relacija na nivou celog modela na dijagram podmodela, dobijamo odgovarajuće entitete u datom podmodelu.



6.1.4. Kriterijumi vrednovanja konceptualnog modela podataka

Van Belle-ov [1] okvirni pristup evaluaciji modela daje osnovne kriterijume:

Aspekt	**Tip**	**Kriterijum evaluacije modela**
Sintaksni	Kvantitativni	Kompleksnost
	Kvalitativni	Korektnost
Semantički	Kvalitativni	Usklađenost sa problemskim domenom
	Kvantitativni Kvalitativni	Kompletnost
Pragmatički	Kvalitativni	Ponovna iskoristivost (Reusability)
	Kvalitativni	Fleksibilnost
	Kvalitativni	Proširivost (Scalability)

Karakteristike konceptualnog modela koje se vrednuju u ocenjivanju radova studenata na Tehničkom fakultetu „Mihajlo Pupin" u Zrenjaninu, predstavljene su sledećom tabelom:

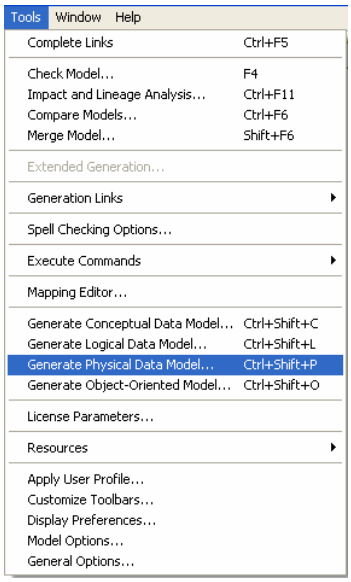
GRUPA ELEMENATA	**GRUPA OSOBINA**	**Potrebna osobina - METRIKA**
CELINA MODELA	Semantika	Semantički u skladu sa problemom
	Usklađenost sa datim materijalom	Pokrivenost opisa posla
		Pokrivenost importovanih elementarnih podataka
		Pokrivenost skladišta podataka sa DTP
	Kompletnost	Kompletnost skupa entiteta
		Kompletnost skupa atributa
		Kompletnost skupa poveznika
	Jezička nedvosmislenost	Nepostojanje sinonima u nazivima entiteta i atributa
		Nepostojanje homonima u nazivima entiteta i atributa
ENTITET	Naziv	Imenica ili glagolska imenica
		Izražava sustinske objekte i dogadjaje
		Ne izražava nazive dokumenata
		Jednina u nazivu
	Entitet i atributi	Ima identifikaciono obeležje ili je identifikaciono zavisan od drugih entiteta
		Dodeljen atribut odgovarajućem entitetu
		(2NF) Ne postoje brojni odnosi atributa 1:M u entitetu
		(3NF) Ne postoje tranzitivne zavisnosti atributa u entitetu
ATRIBUT	Naziv	(1NF) Jednina u nazivu
		(1NF) Nema naziv kao struktura
	Atributi i tipovi /domeni podataka	Odgovarajući tip podatka
		Atribut nema za tip podatka domen (uzi skup vrednosti nego sto je standardni tip podatka), vec je izvucen kao sifarnik
	Karakteristike	Atributi se sa istim nazivom ne ponavljaju (nema redundanse)
		Nema izračunljivih atributa, vec samo analitickih vrednosti medju atributima na osnovu kojih se vrše izracunavanja
POVEZNIK	Naziv	Poveznik ima naziv
		Naziv kao glagol
	Nacin povezivanja entiteta	Povezani direktno entiteti koji treba da budu povezani
		Entiteti su povezani hronoloskim sledom međuzavisnih događaja
		Povezani su entiteti u gerund odnosu, ako je potrebno
		Ako poveznik ima važne atribute, pretvoren je u entitet
	Podesavanja osobina	Pravilno podešen kardinalitet
		Pravilno podešena strana na vezi 1:M
		Pravilno određeno da li ima opravdanja za vezu 1:1 ili je jedan entitet
		Pravilno određeno u vezi 1:1 ko je dominant
		Pravilno određeno u vezi M:M da li ima važnih atributa ili ostaje M:M
		Pravilno podesena identifikaciona zavisnost
		Pravilno podešena is-a hijerarhija (gde je nadredjeni, bar jedan podređeni, migrira samo ID)

Literatura

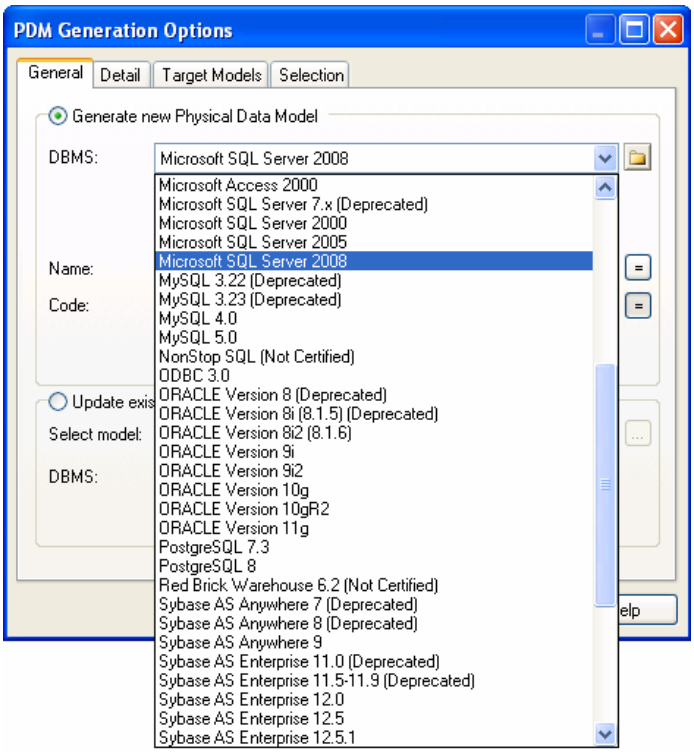
[1] J-P Van Belle: *A Framework for the Evaluation of Business Models and its Empirical Validation*, The Electronic Journal Information Systems Evaluation 2006, Volume 9 Issue 1, pp 31-44, www.ejise.com

6.2. KREIRANJE FIZIČKOG MODELA PODATAKA

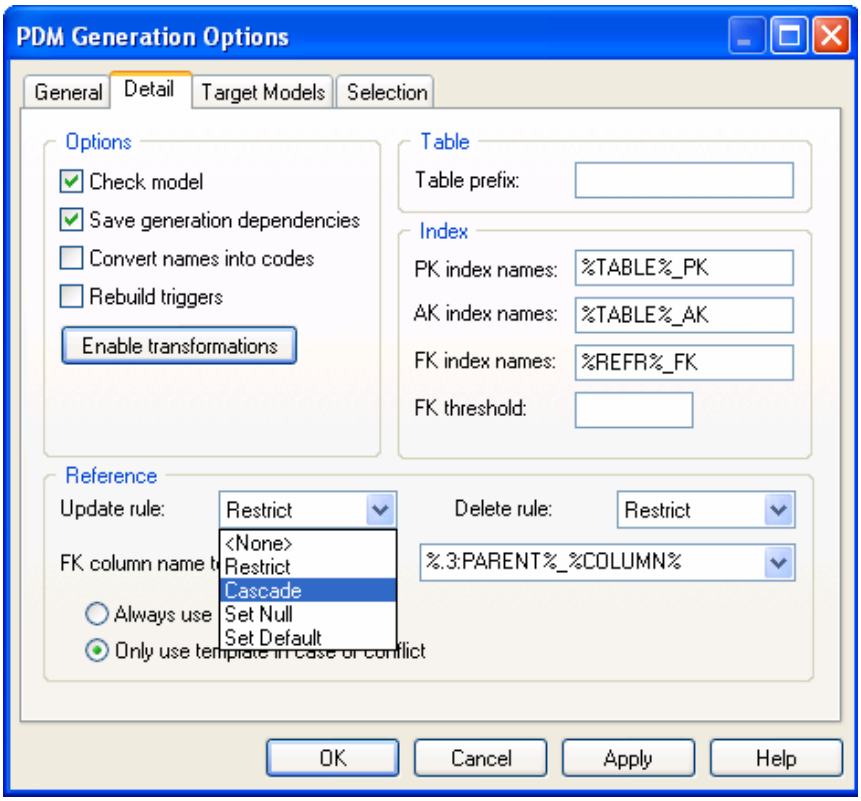
U okviru CASE alata Power Designer 15, izborom opcije Tools omogućena je automatska transformacija konceptualnog modela u fizički model (odnosno prevođenje ER dijagrama u relacioni).



Nakon toga dobijamo dijalog prozor za izbor ciljnog DBMS-a:

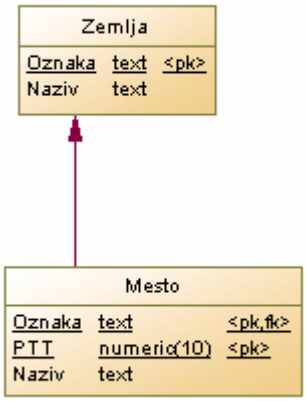


Podešavamo osobine referencijalnog integriteta, biramo za Update rule CASCADE (automatska izmena vrednosti stranog ključa ukoliko se vrednost primarnog ključa promeni), a za Delete rule RESTRICT (zabrana izvršavanja brisanja ukoliko postoji bar jedan zapis na strani M). Ovo su najčešća podešavanja.



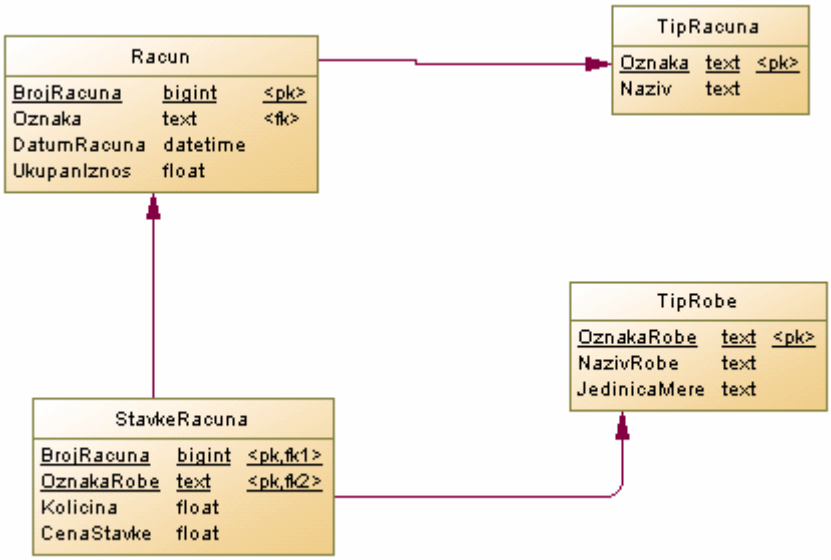
1. PRIMER

Za prethodni primer kreiranog konceptualnog modela Zemlja – Mesto, nakon kreiranja fizičkog modela primarni ključ iz tabele zemlja migrirao je u tabelu mesto:



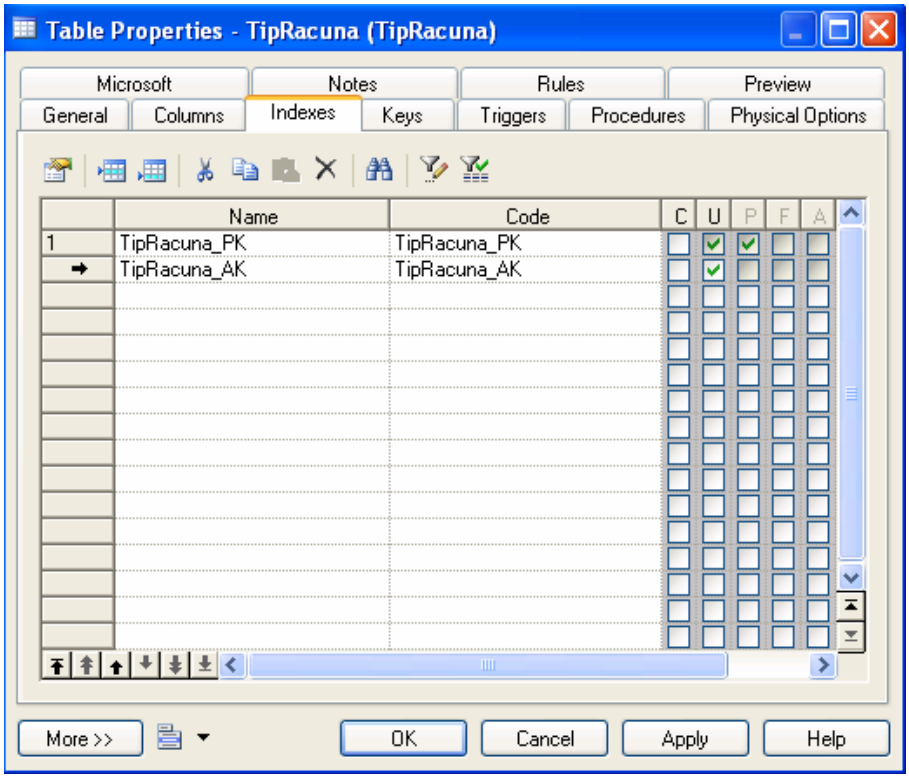
2. PRIMER

Za prethodni primer kreiranog konceptualnog modela za evidenciju računa, nakon kreiranja fizičkog modela dobijamo:

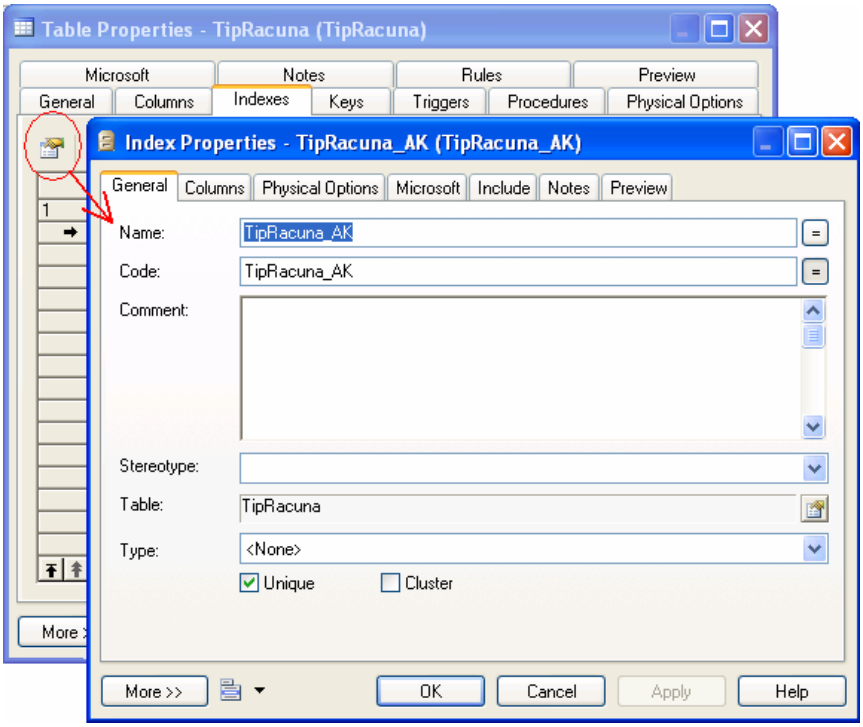


6.2.1. Podešavanje alternativnog ključa – unique indeksa

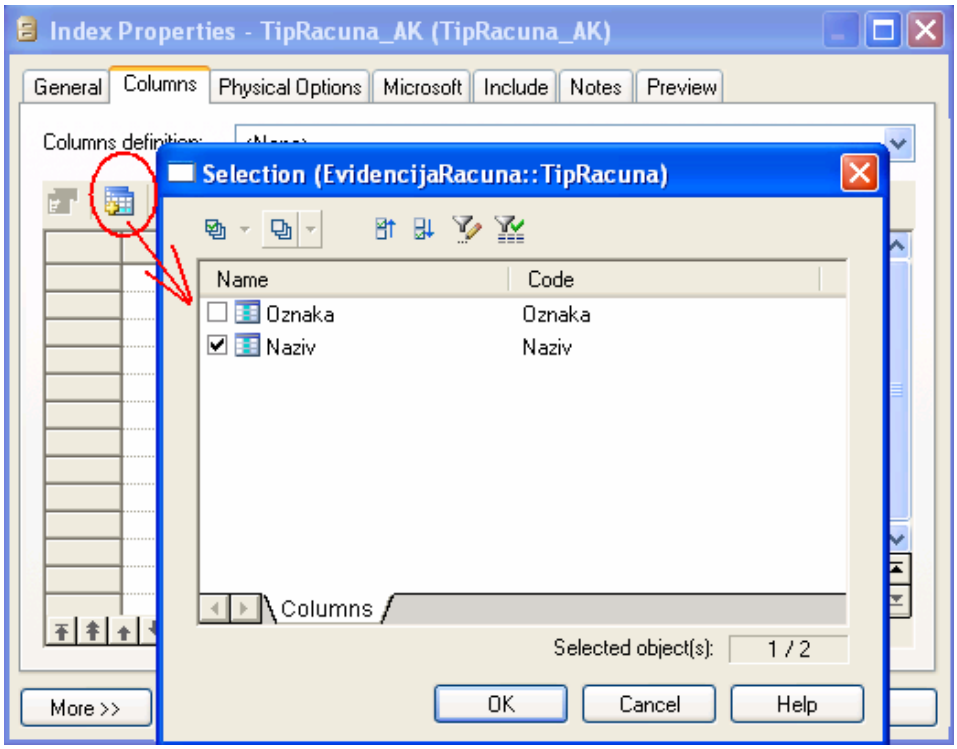
Za prethodni primer, za tabelu TipRačuna podešavamo alternativni ključ, odnosno UNIQUE indeks kojim se sprečava dupliranje podataka. Primarni ključ sprečava ponavljanje vrednosti primarnog ključa, ali ostala neključna polja se mogu ponavljati. Iz tog razloga nad neključnim poljima (minimalnim skupom) se definiše unique indeks (nije primary – podešena je samo opcija U).



Izborom opcije Properties, otvaramo dijalog prozor za definisanje strukture indeksa.



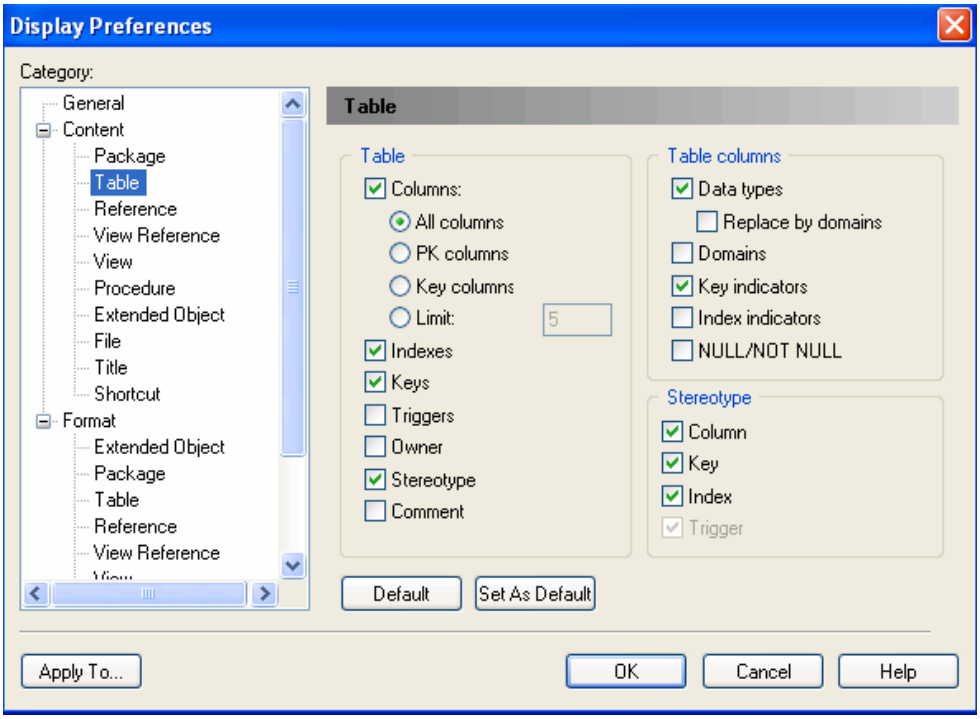
U okviru kartice Columns, izborom opcije za dodavanje polja biramo Naziv kao neključno polje.



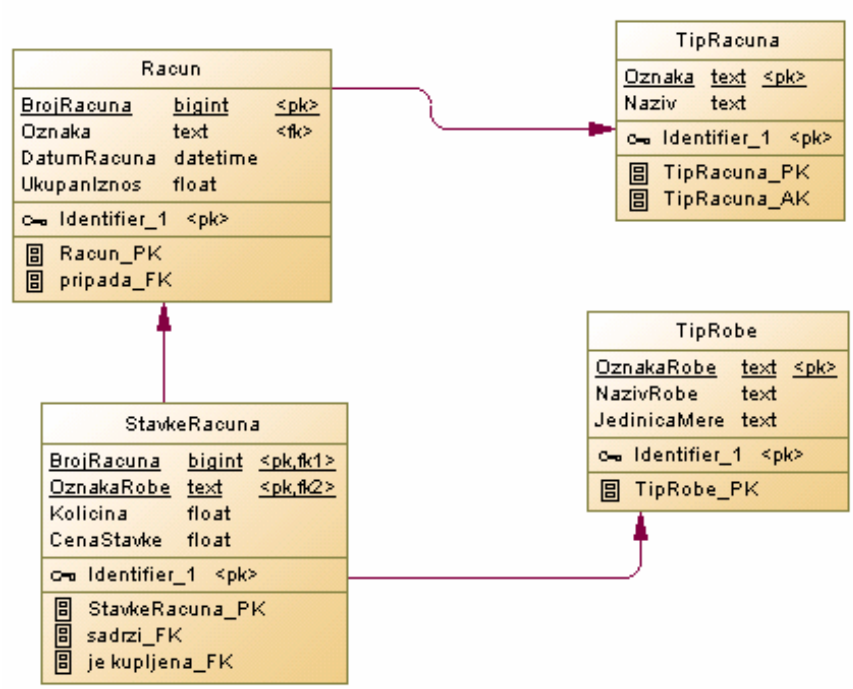
NAPOMENA:

Nastojimo da u svakoj tabeli gde je potrebno definišemo minimalan broj indeksa,a svaki indeks da ima minimalan, ali dovoljan broj polja.

Podešavamo Tools - Display Preferences u CASE alatu, kako bismo uključili na prikazu podatke.

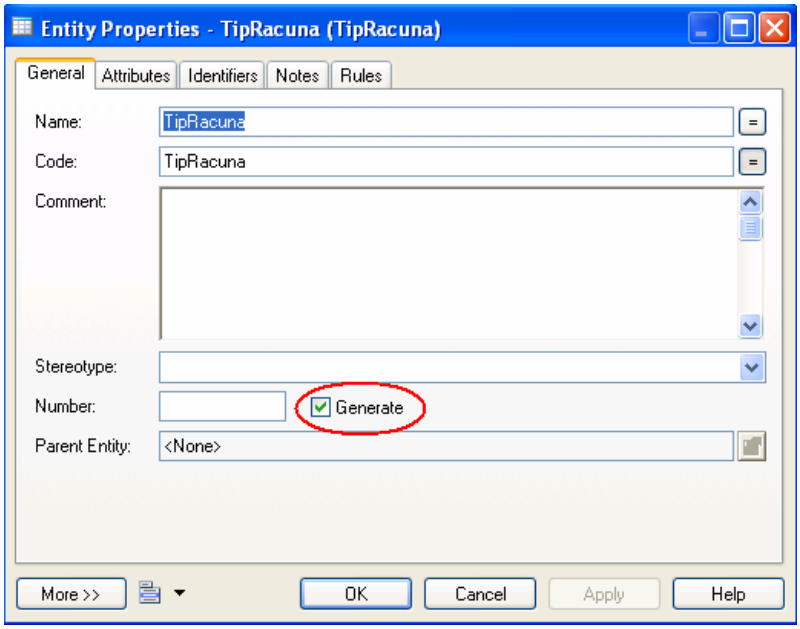


Da bismo na kraju dobili prikaz svih željenih elemenata. Za tabelu Tip računa postavili smo alternativni ključ – TipRačuna_AK. Za tabelu TipRobe trebalo bi takođe uraditi alternativni ključ, ali to nismo uradili u ovom primeru.

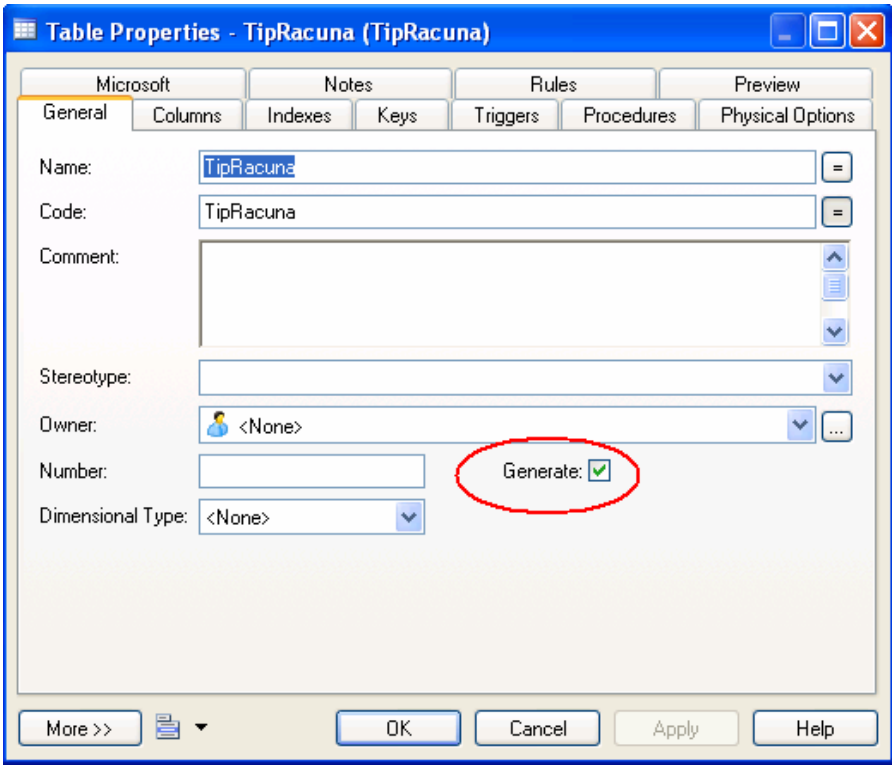


6.3. GENERISANJE BAZE PODATAKA

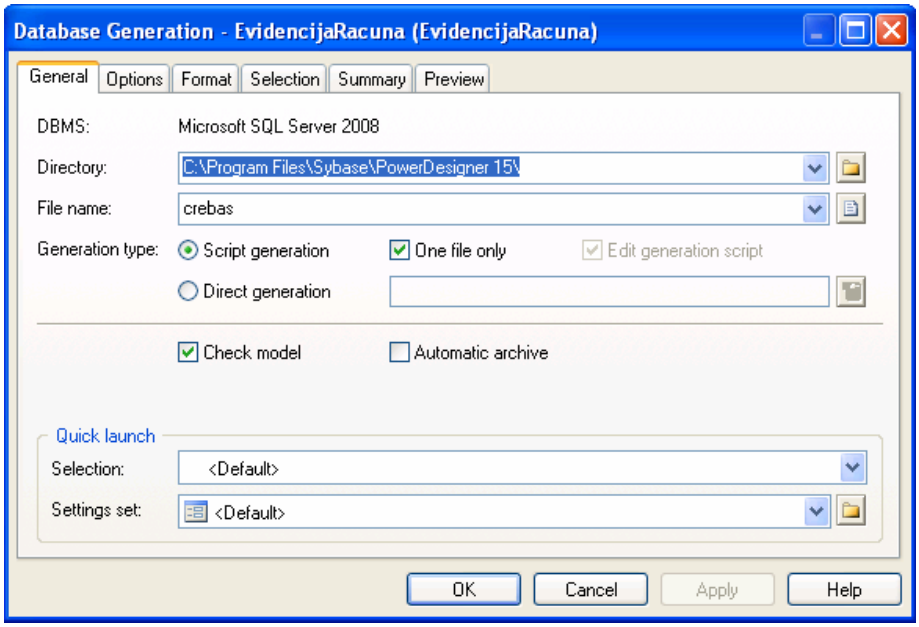
Da bi neki entitet iz konceptualnog modela bio kreiran u tabelu fizičkog modela, mora biti podešena opcija „Generate" za dati entitet.



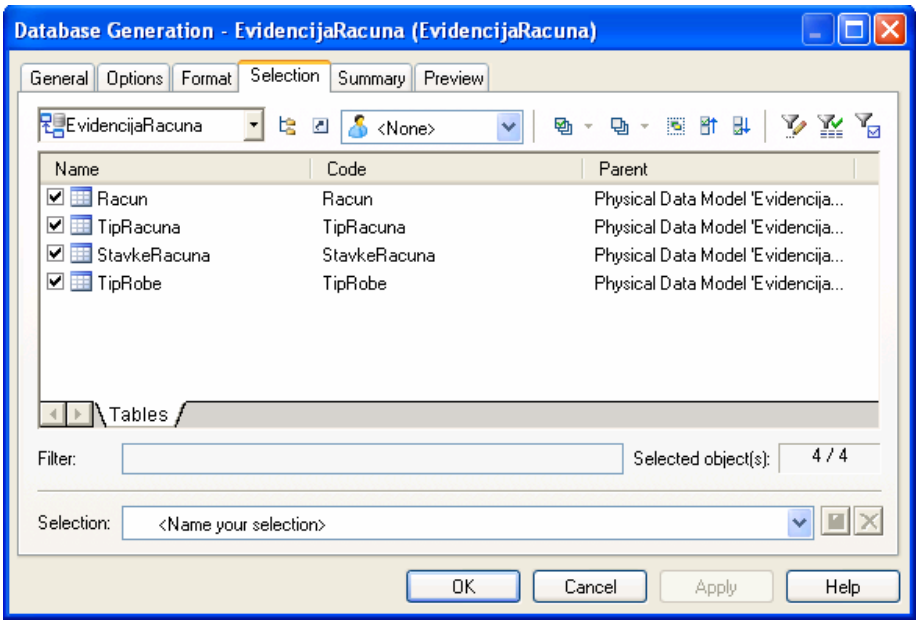
Takođe, u fizičkom modelu, za tabelu takođe mora biti podešena osobina „Generate".



Da bismo generisali bazu podataka iz CASE alata, koristimo opciju glavnog menija Database – podstavka Generate Database, koja se aktivira kada je fizički model aktuelan. Dobijamo dijalog prozor za izbor načina generisanja baze podataka.



Na kartici „Selection" proveravamo da li su sve tabele uključene.

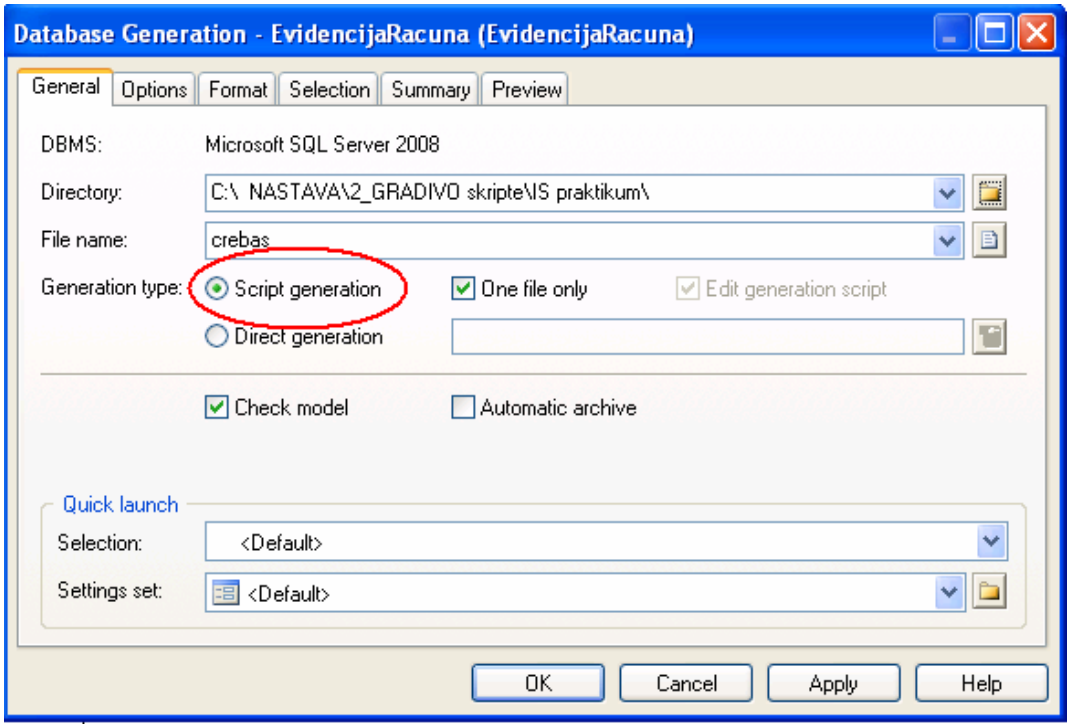


Baza podataka može biti generisana na 2 načina:

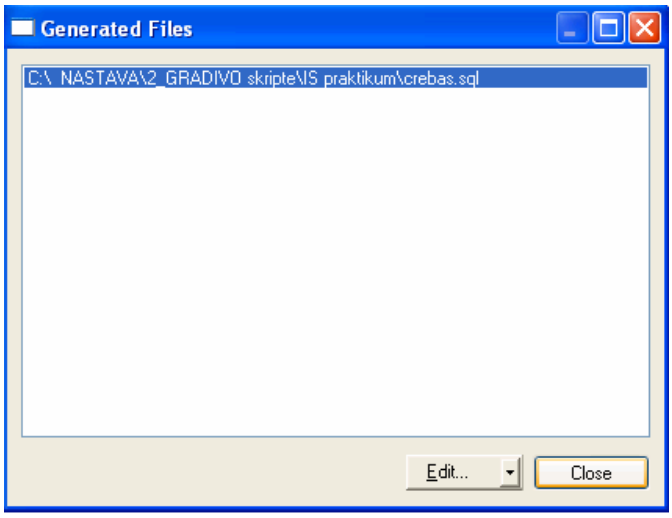
1. Kreiranje skripta (tekstualna datoteka sa SQL naredbama)
2. Direktno generisanje (najpre se generiše skript, a zatim se direktnom konekcijom na bazu podataka realizuje kreiranje tabela i relacija)

6.3.1. Kreiranje SQL skripta za generisanje baze podataka

Biramo prvu opciju – kreiranje SQL skripta:



Dobijamo izveštaj o generisanom skriptu:



Sadržaj kreiranog SQL skripta:

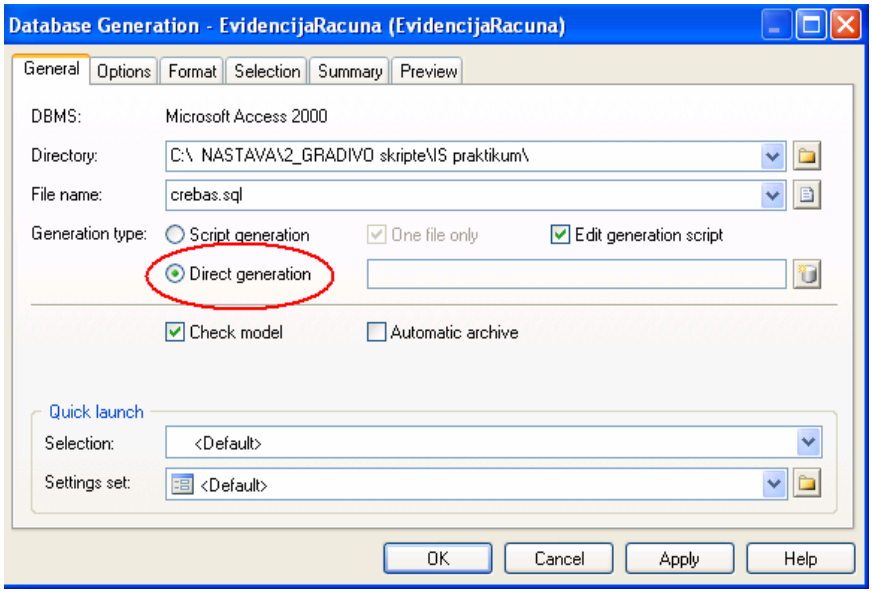
```
/*==============================================================*/
/* DBMS name:      Microsoft SQL Server 2008                    */
/* Created on:     1.12.2013 12:23:57                           */
/*==============================================================*/


if exists (select 1
          from  sysindexes
         where  id    = object_id('Racun')
          and   name  = 'pripada_FK'
          and   indid > 0
          and   indid < 255)
   drop index Racun.pripada_FK
go

if exists (select 1
          from  sysobjects
```

```
       where  id = object_id('Racun')
        and   type = 'U')
  drop table Racun
go

if exists (select 1
        from  sysindexes
       where  id    = object_id('StavkeRacuna')
        and   name  = 'je kupljena_FK'
        and   indid > 0
        and   indid < 255)
  drop index StavkeRacuna."je kupljena_FK"
go

if exists (select 1
        from  sysindexes
       where  id    = object_id('StavkeRacuna')
        and   name  = 'sadrzi_FK'
        and   indid > 0
        and   indid < 255)
  drop index StavkeRacuna.sadrzi_FK
go

if exists (select 1
        from  sysobjects
       where  id = object_id('StavkeRacuna')
        and   type = 'U')
  drop table StavkeRacuna
go

if exists (select 1
        from  sysindexes
       where  id    = object_id('TipRacuna')
        and   name  = 'TipRacuna_AK'
        and   indid > 0
        and   indid < 255)
  drop index TipRacuna.TipRacuna_AK
go

if exists (select 1
        from  sysobjects
       where  id = object_id('TipRacuna')
        and   type = 'U')
  drop table TipRacuna
go

if exists (select 1
        from  sysobjects
       where  id = object_id('TipRobe')
        and   type = 'U')
  drop table TipRobe
go

/*==============================================================*/
/* Table: Racun                                 */
/*==============================================================*/
create table Racun (
  BrojRacuna         bigint            not null,
  Oznaka             text              not null,
  DatumRacuna        datetime          not null,
  UkupanIznos        float             not null,
  constraint PK_RACUN primary key nonclustered (BrojRacuna)
)
go

/*==============================================================*/
/* Index: pripada_FK                            */
/*==============================================================*/
create index pripada_FK on Racun (
Oznaka ASC
)
```

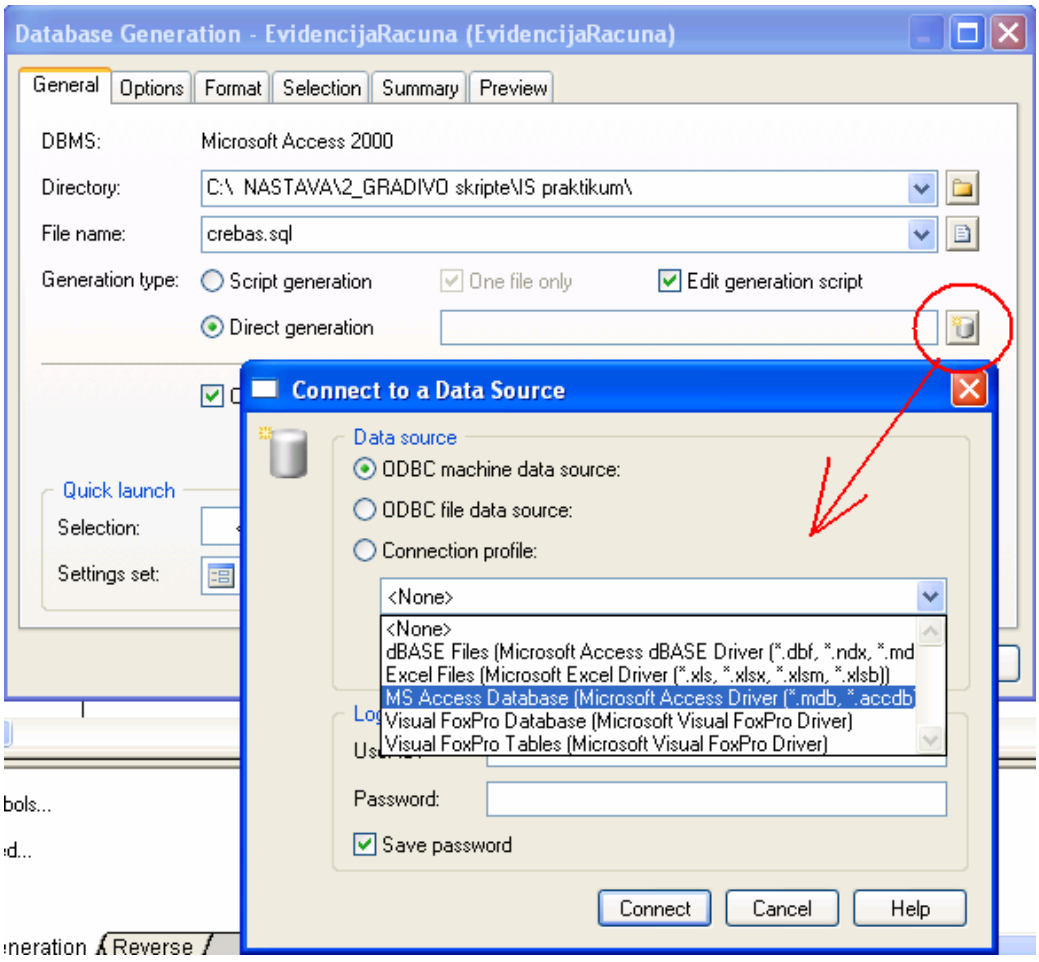
```
go

/*==============================================================*/
/* Table: StavkeRacuna                                          */
/*==============================================================*/
create table StavkeRacuna (
   BrojRacuna          bigint              not null,
   OznakaRobe          text                not null,
   Kolicina            float               null,
   CenaStavke          float               null,
   constraint PK_STAVKERACUNA primary key (BrojRacuna, OznakaRobe)
)
go

/*==============================================================*/
/* Index: sadrzi_FK                                             */
/*==============================================================*/
create index sadrzi_FK on StavkeRacuna (
BrojRacuna ASC
)
go

/*==============================================================*/
/* Index: "je kupljena_FK"                                      */
/*==============================================================*/
create index "je kupljena_FK" on StavkeRacuna (
OznakaRobe ASC
)
go

/*==============================================================*/
/* Table: TipRacuna                                             */
/*==============================================================*/
create table TipRacuna (
   Oznaka              text                not null,
   Naziv               text                not null,
   constraint PK_TIPRACUNA primary key nonclustered (Oznaka)
)
go

/*==============================================================*/
/* Index: TipRacuna_AK                                          */
/*==============================================================*/
create unique index TipRacuna_AK on TipRacuna (
Naziv ASC
)
go

/*==============================================================*/
/* Table: TipRobe                                               */
/*==============================================================*/
create table TipRobe (
   OznakaRobe          text                not null,
   NazivRobe           text                not null,
   JedinicaMere        text                not null,
   constraint PK_TIPROBE primary key nonclustered (OznakaRobe)
)
go
```

6.3.2. Direktno generisanje baze podataka iz CASE alata

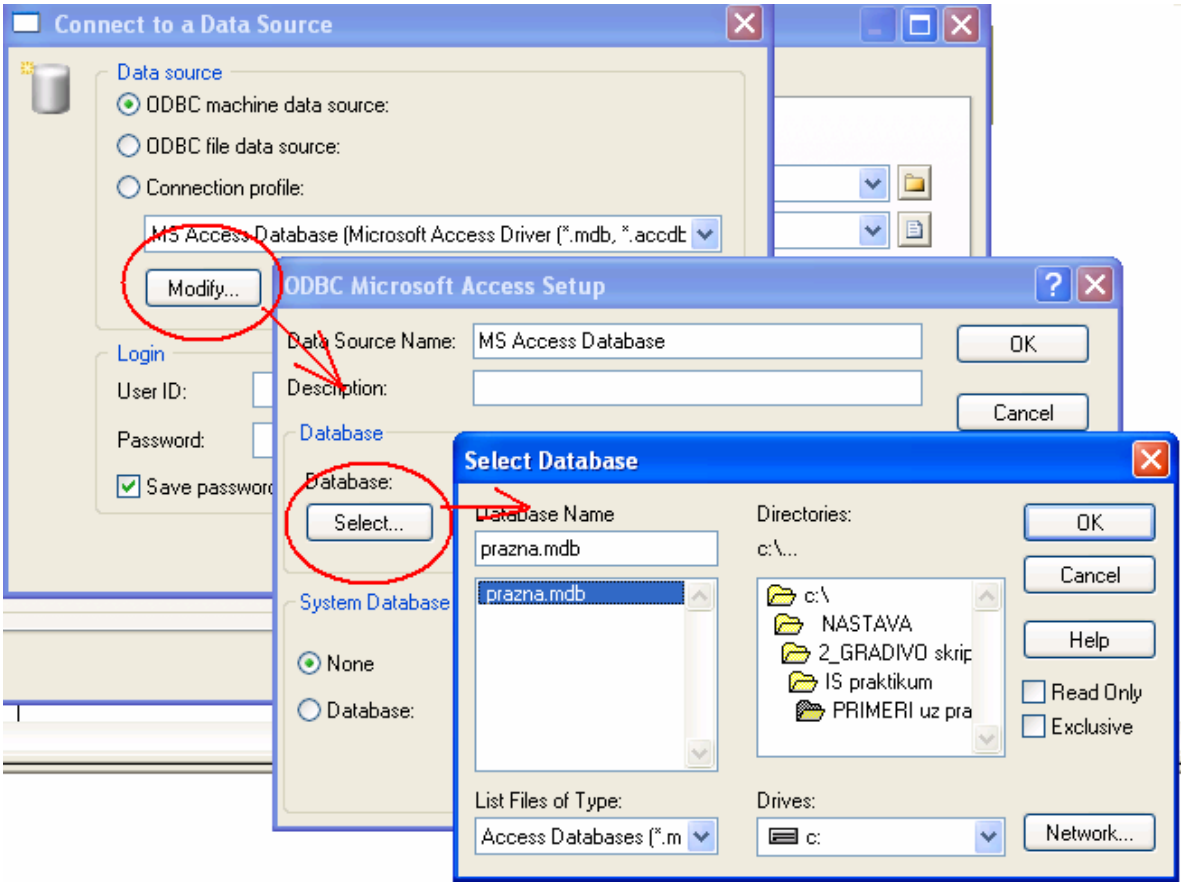
Biramo opciju za direktno generisanje:



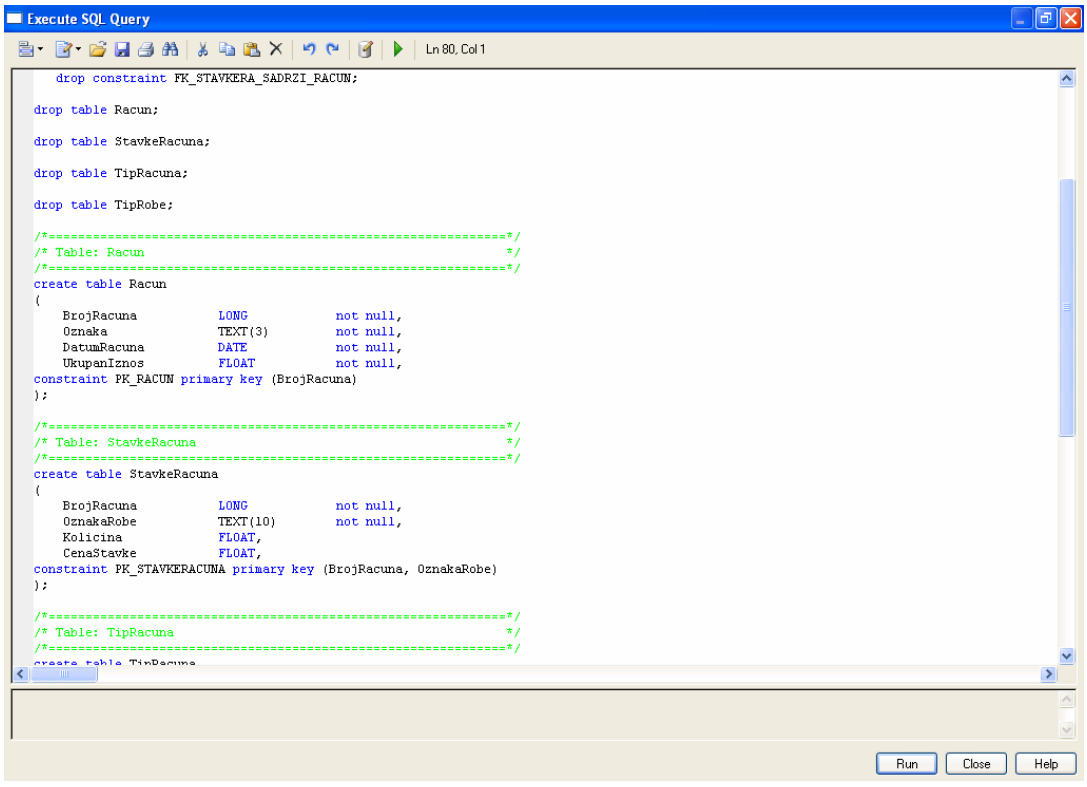
Nakon izbora ikonice pored opcije za direktno generisanje dobijamo dijalog prozor, gde biramo ODBC (Open DataBase Connectivity) machine data source.



Biramo opciju Modify ... , Select ... Biramo fajl prazne baze podataka.



Dobijamo SQL skript:



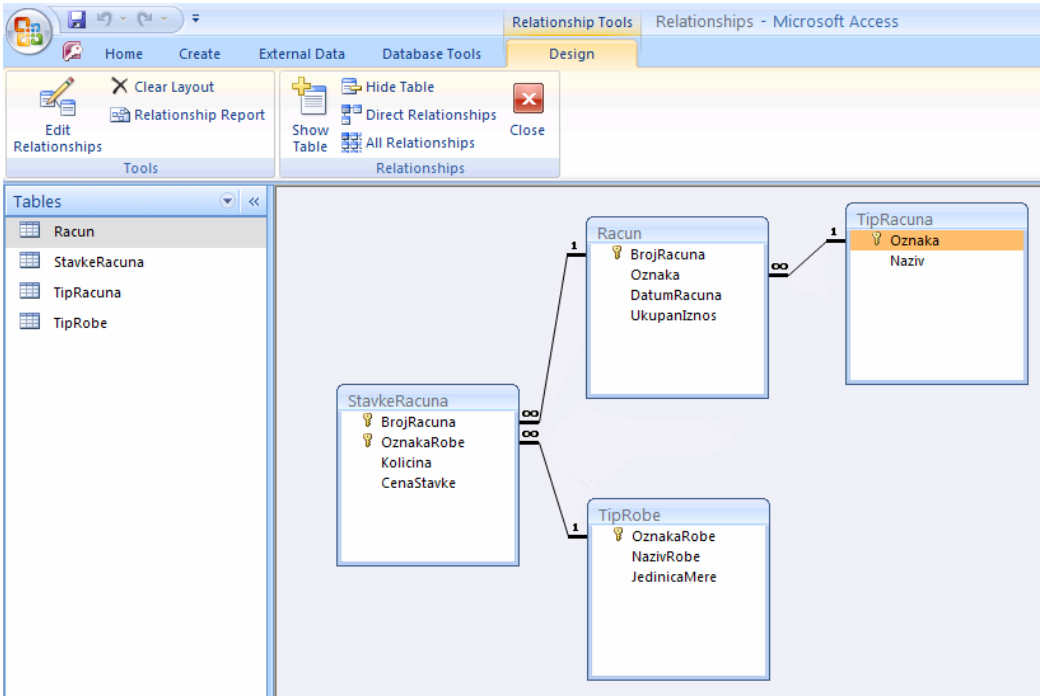
==========/

```
/* DBMS name:      Microsoft Access 2000                   */
/* Created on:     1.12.2013 12:50:43                      */
/*============================================================*/


alter table Racun
  drop constraint FK_RACUN_PRIPADA_TIPRACUN;

alter table StavkeRacuna
  drop constraint "FK_STAVKERA_JE KUPLJE_TIPROBE";

alter table StavkeRacuna
  drop constraint FK_STAVKERA_SADRZI_RACUN;

drop table Racun;

drop table StavkeRacuna;

drop table TipRacuna;

drop table TipRobe;

/*============================================================*/
/* Table: Racun                                               */
/*============================================================*/
create table Racun
(
   BrojRacuna        LONG         not null,
   Oznaka            TEXT(3)      not null,
   DatumRacuna       DATE         not null,
   UkupanIznos       FLOAT        not null,
constraint PK_RACUN primary key (BrojRacuna)
);

/*============================================================*/
/* Table: StavkeRacuna                                        */
/*============================================================*/
create table StavkeRacuna
(
   BrojRacuna        LONG         not null,
   OznakaRobe        TEXT(10)     not null,
   Kolicina          FLOAT,
   CenaStavke        FLOAT,
constraint PK_STAVKERACUNA primary key (BrojRacuna, OznakaRobe)
);

/*============================================================*/
/* Table: TipRacuna                                           */
/*============================================================*/
create table TipRacuna
(
   Oznaka            TEXT(3)      not null,
   Naziv             TEXT(15)     not null,
constraint PK_TIPRACUNA primary key (Oznaka)
);

/*============================================================*/
/* Table: TipRobe                                             */
/*============================================================*/
create table TipRobe
(
   OznakaRobe        TEXT(10)     not null,
   NazivRobe         TEXT(100)    not null,
   JedinicaMere      TEXT(10)     not null,
constraint PK_TIPROBE primary key (OznakaRobe)
);

alter table Racun
  add constraint FK_RACUN_PRIPADA_TIPRACUN foreign key (Oznaka)
    references TipRacuna (Oznaka);
```

```
alter table StavkeRacuna
  add constraint "FK_STAVKERA_JE KUPLJE_TIPROBE" foreign key (OznakaRobe)
    references TipRobe (OznakaRobe);

alter table StavkeRacuna
  add constraint FK_STAVKERA_SADRZI_RACUN foreign key (BrojRacuna)
    references Racun (BrojRacuna);
```

Pokrećemo opciju RUN i dobijamo:

Dobijamo bazu podataka sa tabelama i relacijama:

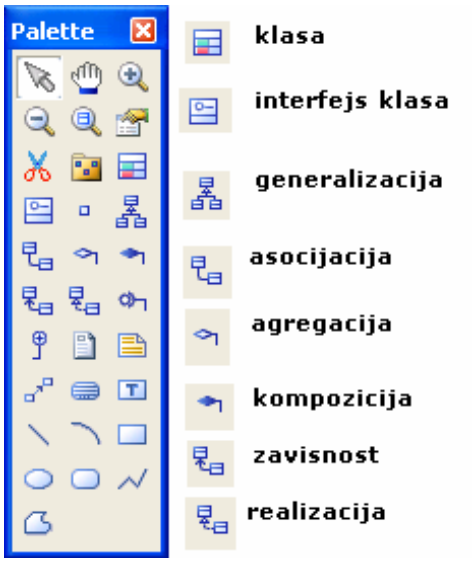


6.4. KREIRANJE OBJEKTNOG MODELA PODATAKA

6.4.1. Kreiranje dijagrama klasa direktnim modelovanjem

Suštinu objektnog modela čini dijagram klasa koji je sličan po strukturi konceptualnom, odnosno fizičkom modelu podataka (klase odgovaraju entitetima odnosno tabelama, a relacije između klasa poveznicima između entiteta, odnosno relacijama između tabela).

Kreiranje objektnog modela, odnosno dijagrama klasa može se vršiti u CASE alatu Power Designer 15, direktnim modelovanjem koristeći paletu alata.



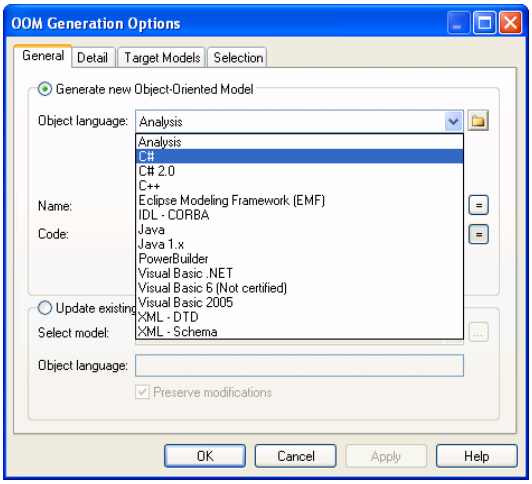
U okviru palete alata za rad sa objektnim modelom, tj. dijagramom klasa razlikujemo najčešće vrste elemenata:
1. Klasu (sadrži atribute i metode) i interfejs klasu (javni predstavnik više drugih klasa koje realizuju pojedine funkcije interfejs klase)
2. Odnos klasa - generalizacija - odnos nasleđivanja
3. Asocijacija – odnos klasa gde atribut jedne klase predstavlja objekat druge klase
4. Agregacija – atribut jedne klase je uređena lista objekata druge klase, objekat te druge klase može biti uključen u uređene liste više objekata prve klase
5. Kompozicija - atribut jedne klase je uređena lista objekata druge klase, objekat te druge klase može biti uključen u uređenu listu samo jednog objekta prve klase

6.4.2. Kreiranje dijagrama klasa automatskom transformacijom

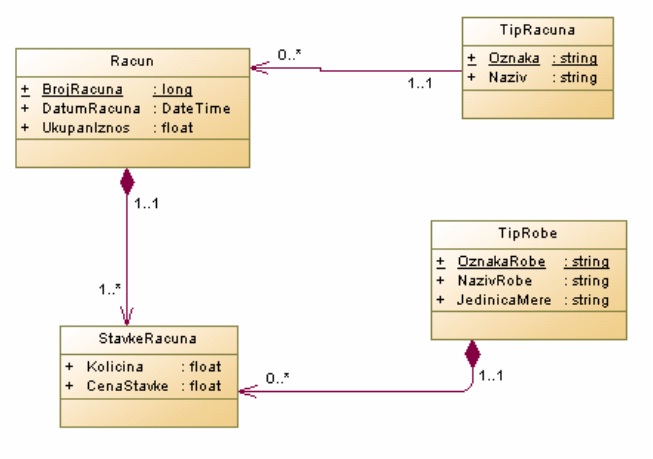
Kreiranje objektnog modela može se izvršiti i automatskim generisanjem u CASE alatu na osnovu:

1. Konceptualnog modela podataka
2. Fizičkog modela podataka

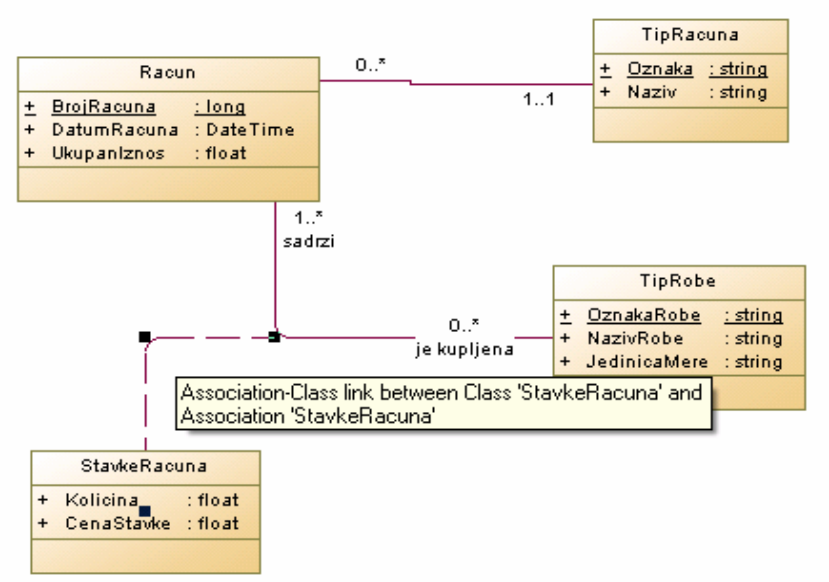
U CASE alatu Power Designer 15, kada je aktivan odgovarajući model, biramo opciju sa menija Tools – Generate object-oriented model i dobijamo dijalog prozor gde možemo birati ciljni jezik, u kojem kasnije može biti generisan programski kod klasa. Biramo C# jezik.



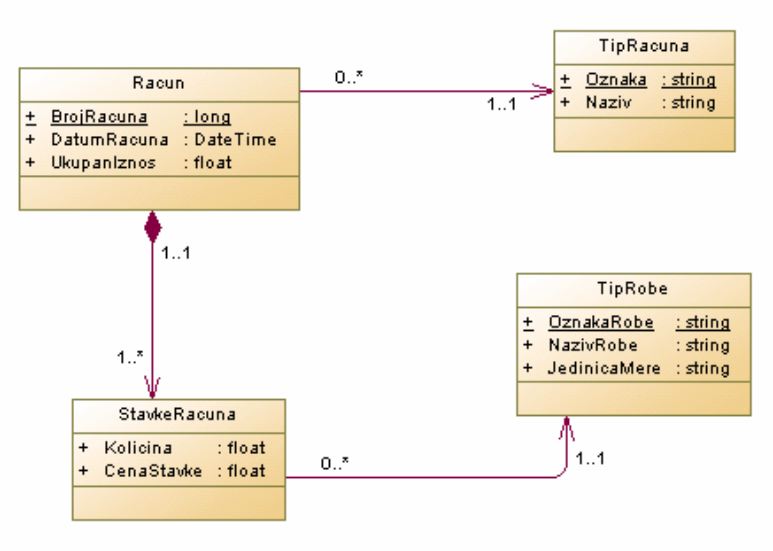
Na osnovu konceptualnog modela dobili smo:



Na osnovu fizičkog modela podataka, generisani objektni model je:

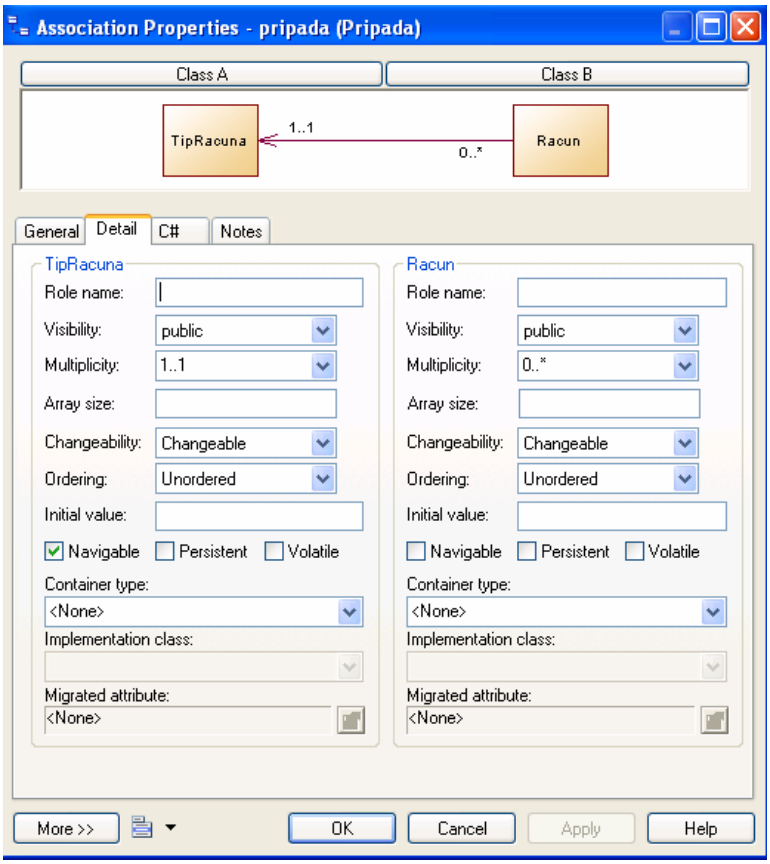


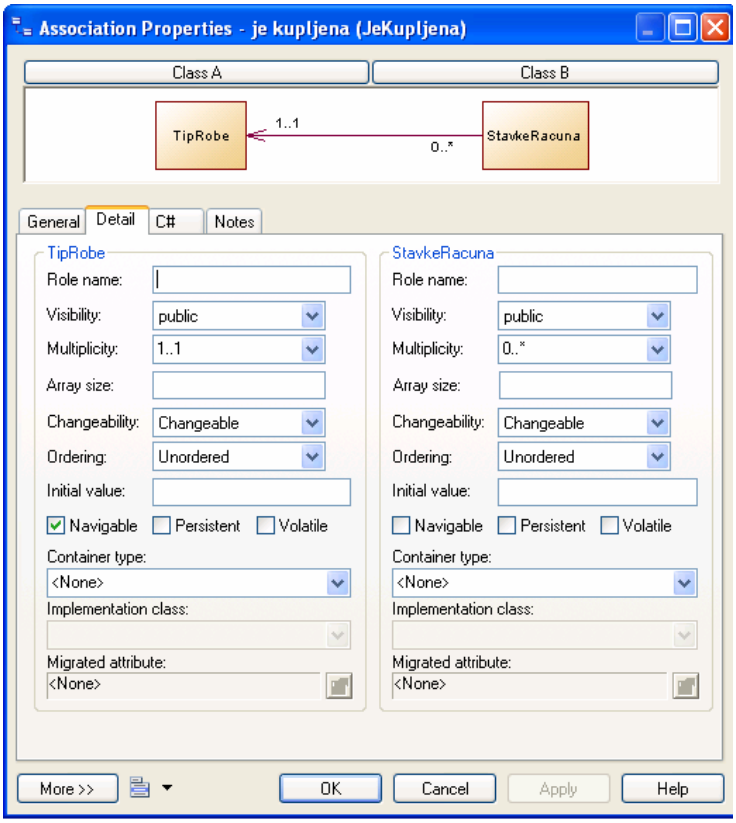
Konačno, kreirani dijagram klasa (model kreiran na osnovu konceptualnog modela podataka je približniji finalnom rešenju, pa njega dalje korigujemo) korigujemo i dobijamo odgovarajuće rešenje dijagrama klasa.



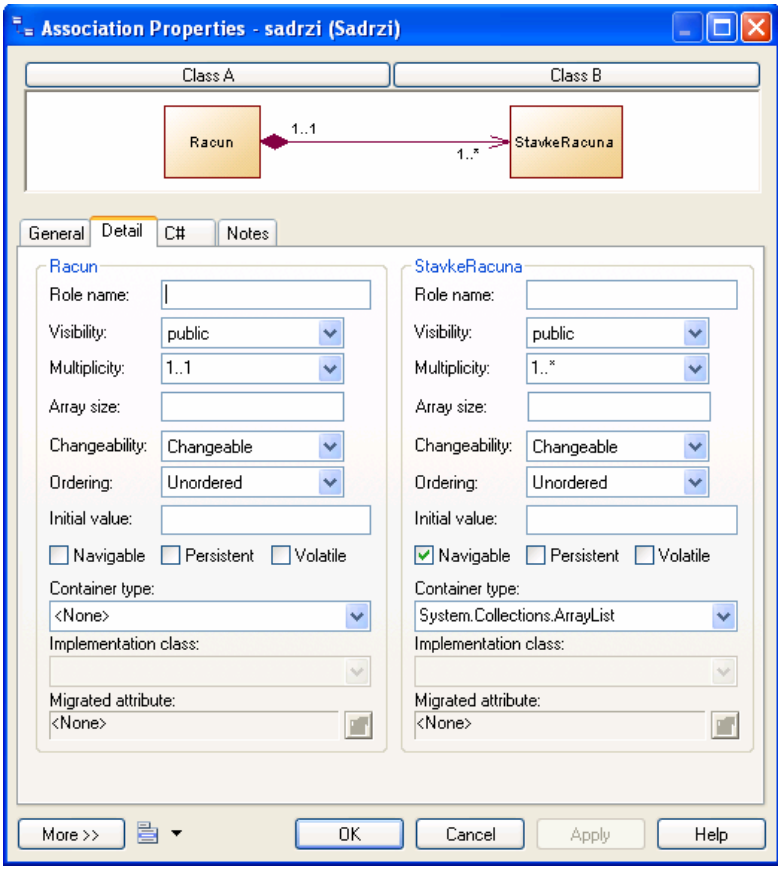
Korekcije koje smo izvršili odnose se na podešavanja relacija između klasa:

- osobina „Navigable" – određuje objekat koje klase će prilikom generisanja programskog koda klasa da migrira kao atribut druge klase, na dijagramu određuje smer strelice.
- osobina „Container type" – određuje da li se prilikom migriranja formira uređena lista objekata, ili migrira kao pojedinačni objekat klase.



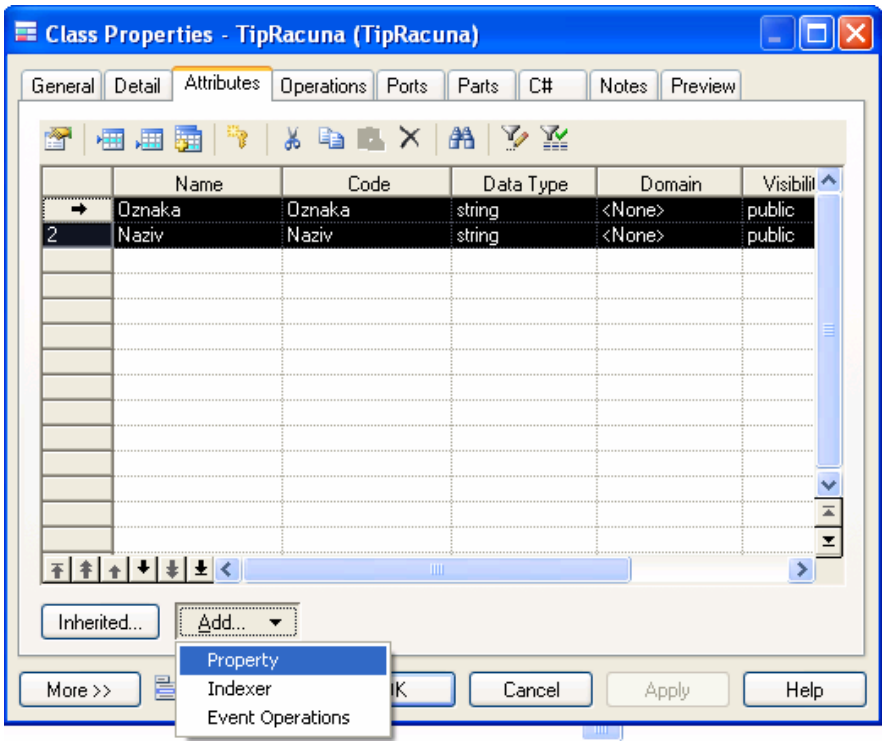


Relacija koja je automatski kreirana iz CASE alata tipa kompozicije je ima odgovarajuće osobine.

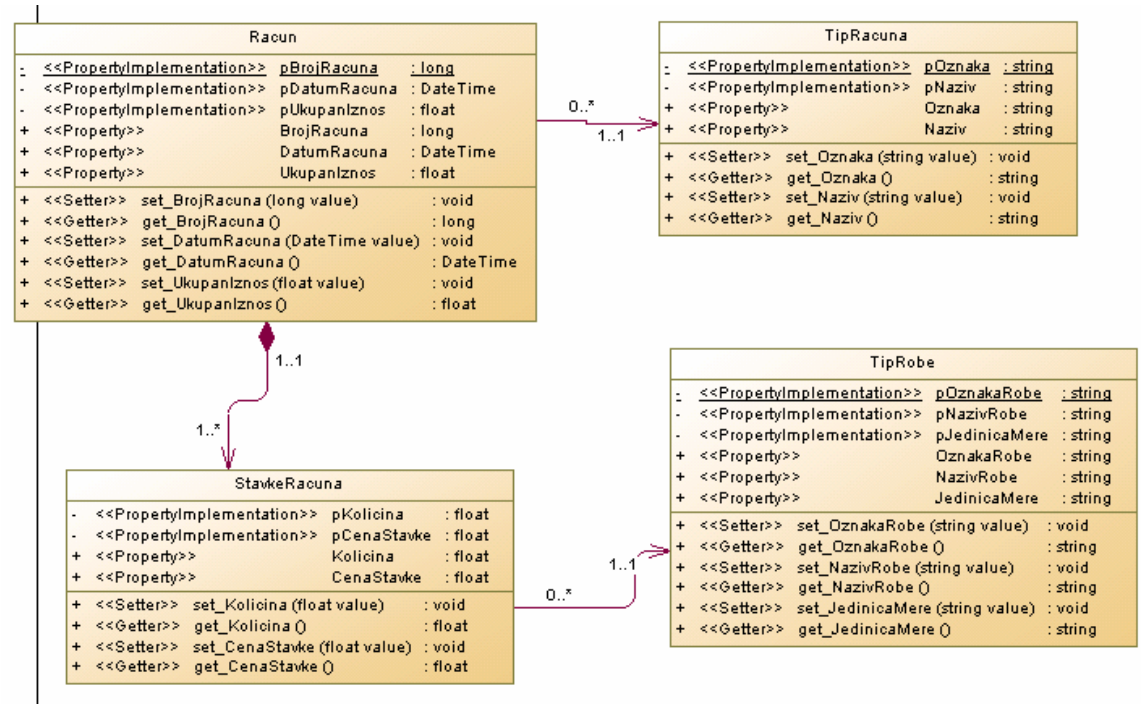


6.4.3. Generisanje programskog koda klasa

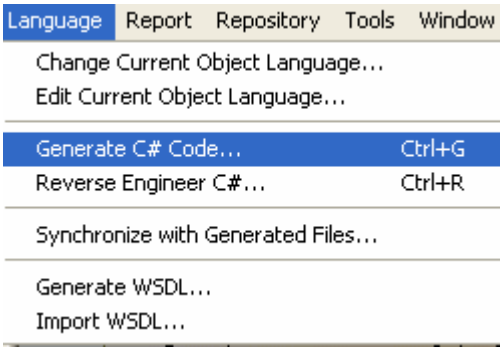
Na osnovu dijagrama klasa možemo generisati programski kod klasa. Pre toga treba da dodamo svakoj klasi odgovarajuće privatne atribute i set-get metode.



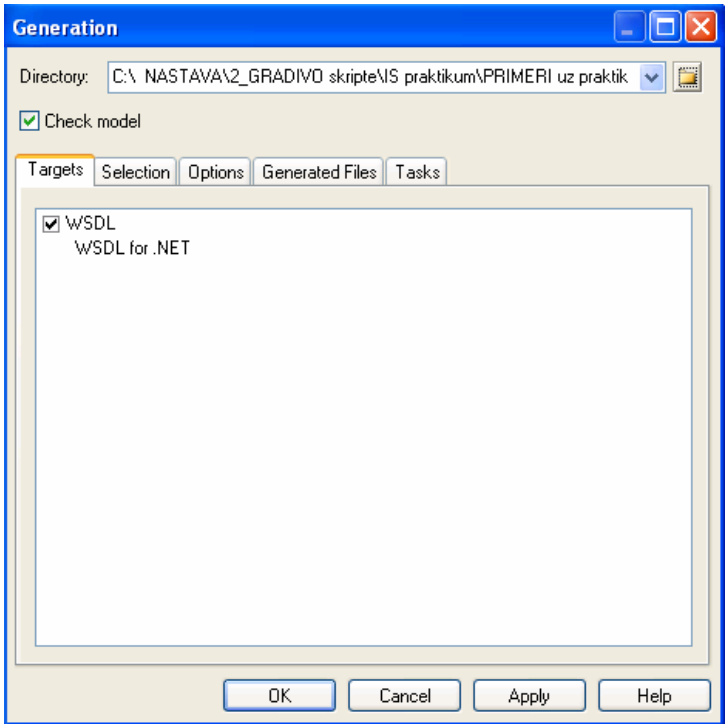
Dobijamo dijagram klasa entiteta sa metodama set-get.



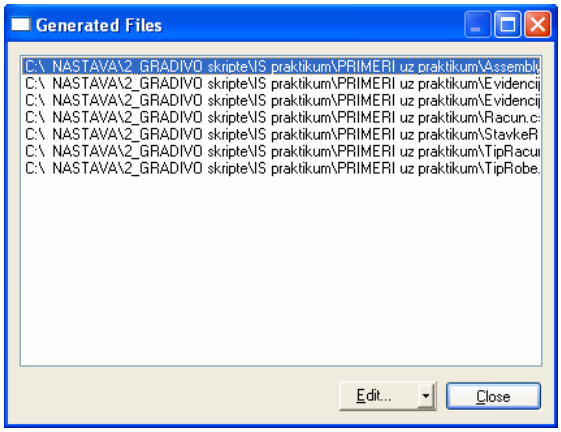
Generisanje programskog koda klasa:



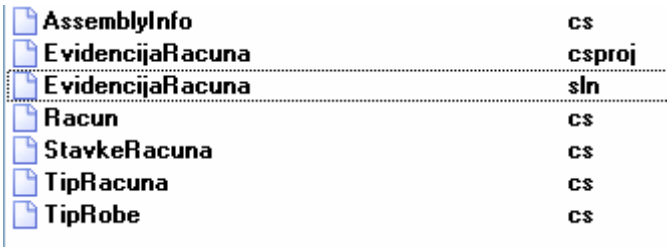
Biramo putanju za snimanje programskog koda:



Dobijamo izveštaj o generisanom programskom kodu klasa:



Dobijamo kreiran projekat tipa biblioteke klasa (class library) sa fajlovima:




```
/*************************************************************************
 * Module:  TipRacuna.cs
 * Author:  ljubica
 * Purpose: Definition of the Class TipRacuna
 *************************************************************************/

using System;

public class TipRacuna
{
  private string pOznaka;
  private string pNaziv;

  public string Oznaka
  {
    get
    {
      return pOznaka;
    }
    set
    {
      if (this.pOznaka != value)
        this.pOznaka = value;
    }
  }

  public string Naziv
  {
    get
    {
      return pNaziv;
    }
    set
    {
      if (this.pNaziv != value)
        this.pNaziv = value;
    }
  }

}



/*************************************************************************
 * Module:  TipRobe.cs
 * Author:  ljubica
 * Purpose: Definition of the Class TipRobe
 *************************************************************************/

using System;

public class TipRobe
{
  private string pOznakaRobe;
  private string pNazivRobe;
  private string pJedinicaMere;

  public string OznakaRobe
```

```
  {
    get
    {
      return pOznakaRobe;
    }
    set
    {
      if (this.pOznakaRobe != value)
        this.pOznakaRobe = value;
    }
  }

  public string NazivRobe
  {
    get
    {
      return pNazivRobe;
    }
    set
    {
      if (this.pNazivRobe != value)
        this.pNazivRobe = value;
    }
  }

  public string JedinicaMere
  {
    get
    {
      return pJedinicaMere;
    }
    set
    {
      if (this.pJedinicaMere != value)
        this.pJedinicaMere = value;
    }
  }

}

/*********************************************************************
 * Module:  StavkeRacuna.cs
 * Author:  ljubica
 * Purpose: Definition of the Class StavkeRacuna
 *********************************************************************/

using System;

public class StavkeRacuna
{
  public TipRobe tipRobe;

  private float pKolicina;
  private float pCenaStavke;

  public float Kolicina
  {
    get
    {
      return pKolicina;
    }
    set
    {
      if (this.pKolicina != value)
        this.pKolicina = value;
    }
  }

  public float CenaStavke
  {
    get
```

```
    {
      return pCenaStavke;
    }
    set
    {
      if (this.pCenaStavke != value)
        this.pCenaStavke = value;
    }
  }

}

/*********************************************************************
 * Module:  Racun.cs
 * Author:  ljubica
 * Purpose: Definition of the Class Racun
 *********************************************************************/

using System;

public class Racun
{
  public System.Collections.ArrayList stavkeRacuna;

  /// <pdGenerated>default getter</pdGenerated>
  public System.Collections.ArrayList GetStavkeRacuna()
  {
    if (stavkeRacuna == null)
      stavkeRacuna = new System.Collections.ArrayList();
    return stavkeRacuna;
  }

  /// <pdGenerated>default setter</pdGenerated>
  public void SetStavkeRacuna(System.Collections.ArrayList newStavkeRacuna)
  {
    RemoveAllStavkeRacuna();
    foreach (StavkeRacuna oStavkeRacuna in newStavkeRacuna)
      AddStavkeRacuna(oStavkeRacuna);
  }

  /// <pdGenerated>default Add</pdGenerated>
  public void AddStavkeRacuna(StavkeRacuna newStavkeRacuna)
  {
    if (newStavkeRacuna == null)
      return;
    if (this.stavkeRacuna == null)
      this.stavkeRacuna = new System.Collections.ArrayList();
    if (!this.stavkeRacuna.Contains(newStavkeRacuna))
      this.stavkeRacuna.Add(newStavkeRacuna);
  }

  /// <pdGenerated>default Remove</pdGenerated>
  public void RemoveStavkeRacuna(StavkeRacuna oldStavkeRacuna)
  {
    if (oldStavkeRacuna == null)
      return;
    if (this.stavkeRacuna != null)
      if (this.stavkeRacuna.Contains(oldStavkeRacuna))
        this.stavkeRacuna.Remove(oldStavkeRacuna);
  }

  /// <pdGenerated>default removeAll</pdGenerated>
  public void RemoveAllStavkeRacuna()
  {
    if (stavkeRacuna != null)
      stavkeRacuna.Clear();
  }
  public TipRacuna tipRacuna;

  private long pBrojRacuna;
  private DateTime pDatumRacuna;
```

```
  private float pUkupanIznos;

  public long BrojRacuna
  {
    get
    {
      return pBrojRacuna;
    }
    set
    {
      if (this.pBrojRacuna != value)
        this.pBrojRacuna = value;
    }
  }

  public DateTime DatumRacuna
  {
    get
    {
      return pDatumRacuna;
    }
    set
    {
      if (this.pDatumRacuna != value)
        this.pDatumRacuna = value;
    }
  }

  public float UkupanIznos
  {
    get
    {
      return pUkupanIznos;
    }
    set
    {
      if (this.pUkupanIznos != value)
        this.pUkupanIznos = value;
    }
  }

}
```

6.6. REŠENI ZADACI

Zadatak:
1. Na osnovu datog materijala (opis posla, dokument i sintaksna analiza dokumenta, dijagram toka podataka i recnik podataka, dijagram slucajeva koriscenja) izraditi konceptualni model podataka:
a) Na osnovu gramatičke analize opisa posla.
b) Na osnovu dodeljivanja atributa iz rečnika podataka entitetima.
c) Na osnovu sintaksne analize dokumenta.
d) Na osnovu sintaksne analize skladišta podataka.
e) Na osnovu normalizacije skladišta podataka.

2. Izvrsiti:
a) integraciju resenja iz 1. Zadatka,
b) proveru i korekciju gresaka,
c) proveru potpunosti modela u odnosu na:
c1) primitivne procese
c2) slucajeve koriscenja.

3. Kreirani konceptualni model transformisati u fizicki model.
4. Generisati bazu podataka.
5. Generisati i korigovati dijagram klasa.
6. Generisati programski kod klasa.

Radi ilustracije načina izrade zadataka, izabran je jednostavan sistem taksi službe i to samo osnovna delatnost, a u okviru nje – rad operatera. Naravno, osnovna delatnost se ne može vrsiti bez rezultata rada i pomocne delatnosti koja se odnosi na nabavku i održavanje resursa – opreme, kadrova i slično, ali smo zbog potrebe za jednostavnošću i fokusiranjem na primenu metoda, minimizovali obuhvat posmatranja.

MATERIJAL ZA IZRADU ZADATKA:

Posmatrani sistem: taksi sluzba – osnovna delatnost – rad operatera
Za navedeni sistem dat je opis posla, analiza strukture dokumenta, stablo procesa, dijagrami toka podataka i tabela preslikavanja primitivnih procesa u softverske funkcije. Struktura skladišta podataka identično je strukturi dokumenta i spisak elementarnih podataka sa sintaksnom analizom predstavlja istovremeno i rečnik podataka.

1. Opis posla za osnovnu delatnost

Građanin (klijent, putnik) telefonskim pozivom ili SMS porukom naručuje prevoz – govori adresu na koju treba da dođe taksi. Operater salje putem radio veze poziv svim aktivnim vozacima da se jave (licitiraju) za datu voznju. Vozač koji se prvi javi i da najpovoljnije vreme će dobiti voznju. Operater saopštava klijentu očekivano vreme dolaska taksi vozila. Ako to klijentu odgovara, vožnja je dogovorena. Operater beleži kom vozaču je dodeljena koja vožnja (svaki vozač/vozilo je jedinstveno identifikovan ID brojem). Ako se nijedno vozilo ne javi, vožnja se ne belezi u dnevniku. Vozač stiže na željenu adresu u predviđenom vremenu. Putnik određuje destinaciju. Vozač uključuje taksimetar. Po dolasku na destinaciju vozač sa taksimetra očitava troškove vožnje. Putnik dobija račun i vrši uplatu za uslugu prevoza. Ukoliko vozač ne stigne na vreme, a klijent poziva ponovo i žali se na čekanje, vozaču se za svaku takvu vožnju oduzima izvestan iznos honorara (ukoliko je zakasnio iz neopravdanih razloga).

2. Dokument za analizu

TAKSI SLUZBA »MAXI-TAXI«
Bačvanska 33
NekoMesto

DNEVNIK RADA
Datum: 13.3.2013.

Vozač	Vreme	TelefonPoziva	PočetnaStanica	Realizovano	Naplaćeno (din)	Razlog nerealizovane voznje
1446	10.30	021/111212	4. Juli 48	DA	240,00	
350	10.35	022/111232	Masarikova 18	DA	350,00	
3456	10.40	023/111242	Knićaninova 25	NE	0	klijent nije došao
451	11.15	024/111252	Voje Tankosića 68	NE	0	vozač je zakasnio
462	11.20	025/111262	Braće Barnić 48	NE	0	kvar na vozilu

Operater
Masarik Juliska

2.1. ANALIZA DOKUMENTA:

2.1.1. Elementarni podaci

NAZIV ELEMENTARNOG PODATKA	TIP PODATKA	DOMEN
NazivTaksiSluzbe	String 50	
AdresaTaksiSluzbe	String 70	
NazivMestaTaksiSluzbe		Mesta
DatumDnevnikaRada	Date	
IDBrojVozaca		IDBrojeviVozaca
VremePozivaKlijenta	Time	
TelefonPozivaKlijenta	String 40	
AdresaPocetneStanice	String 70	
NaplaceniIznos	Real	
RazlogNerealizovaneVoznje		RazloziNerealizovaneVoznje
PrezimeOperatera	String 50	
ImeOperatera	String 30	

2.1.2. DOMENI:
Mesta: String 100 (Nazivi mesta u Srbiji – sifarnik)
IDBrojeviVozaca: String 15 (»1446«, »350«, »3456«, »451«, »462«)
RazloziNerealizovaneVoznje: String 30 (»klijent nije dosao«, »vozac je zakasnio«, »kvar na vozilu«)

2.1.3. SINTAKSNI PRIKAZ STRUKTURE DOKUMENTA:

```
DNEVNIK_RADA:< <NazivTaksiSluzbe,
                AdresaTaksiSluzbe,
                NazivMestaTaksiSluzbe>,
                DatumDnevnikaRada,
                        {<IDBrojVozaca,
                           VremePozivaKlijenta,
                           TelefonPozivaKlijenta,
                           AdresaPocetneStanice,
                           [NaplacenIznos,
                           RazlogNerealizovaneVoznje]
                         >},
                 <PrezimeOperatera,
                  ImeOperatera>
               >;
```

3. Dijagram toka podataka

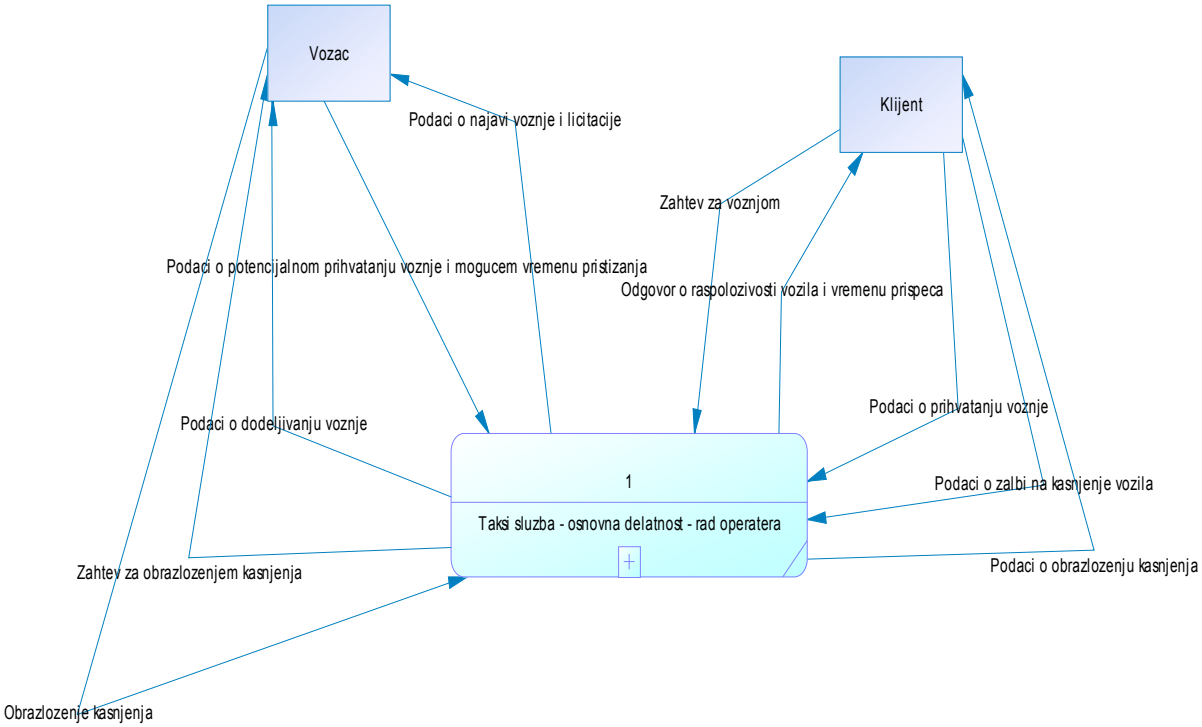
3.1. Stablo procesa

0. Taksi sluzba – osnovna delatnost – rad operatera
1. Priprema voznje
    a. Prijem poziva od klijenta
    b. Obaveštavanje o licitaciji za voznju
    c. Prijem prijave za voznju
    d. Obaveštavanje klijenta o mogućnosti prispeća vozila
    e. Prijem potvrde od klijenta za vožnju
    f. Obaveštavanje vozača o potvrdi za voznju
2. Obrada neizvršene vožnje
    a. Prijem žalbe klijenta o neizvršenoj voznji
    b. Slanje zahteva za obrazloženjem neizvršene vožnje vozaču
    c. Slanje obrazloženja za neizvršenu voznju klijentu
3. Evidentiranje izvršene voznje

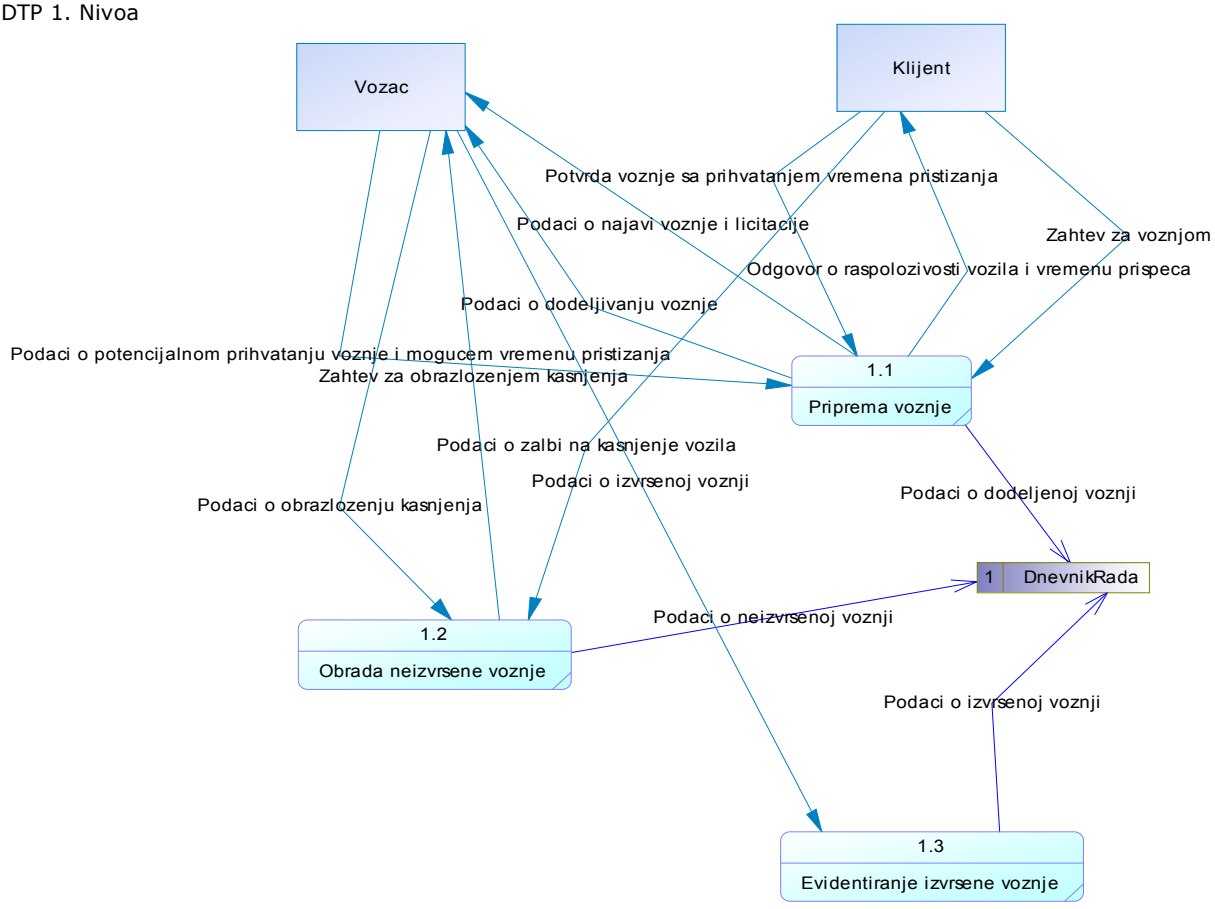
3.2. Dijagrami toka podataka

U ovom primeru su dati samo DTP 0. Nivoa i 1. Nivoa.

DTP 0. Nivoa



DTP 1. Nivoa



4. Dijagram slučajeva koriscenja

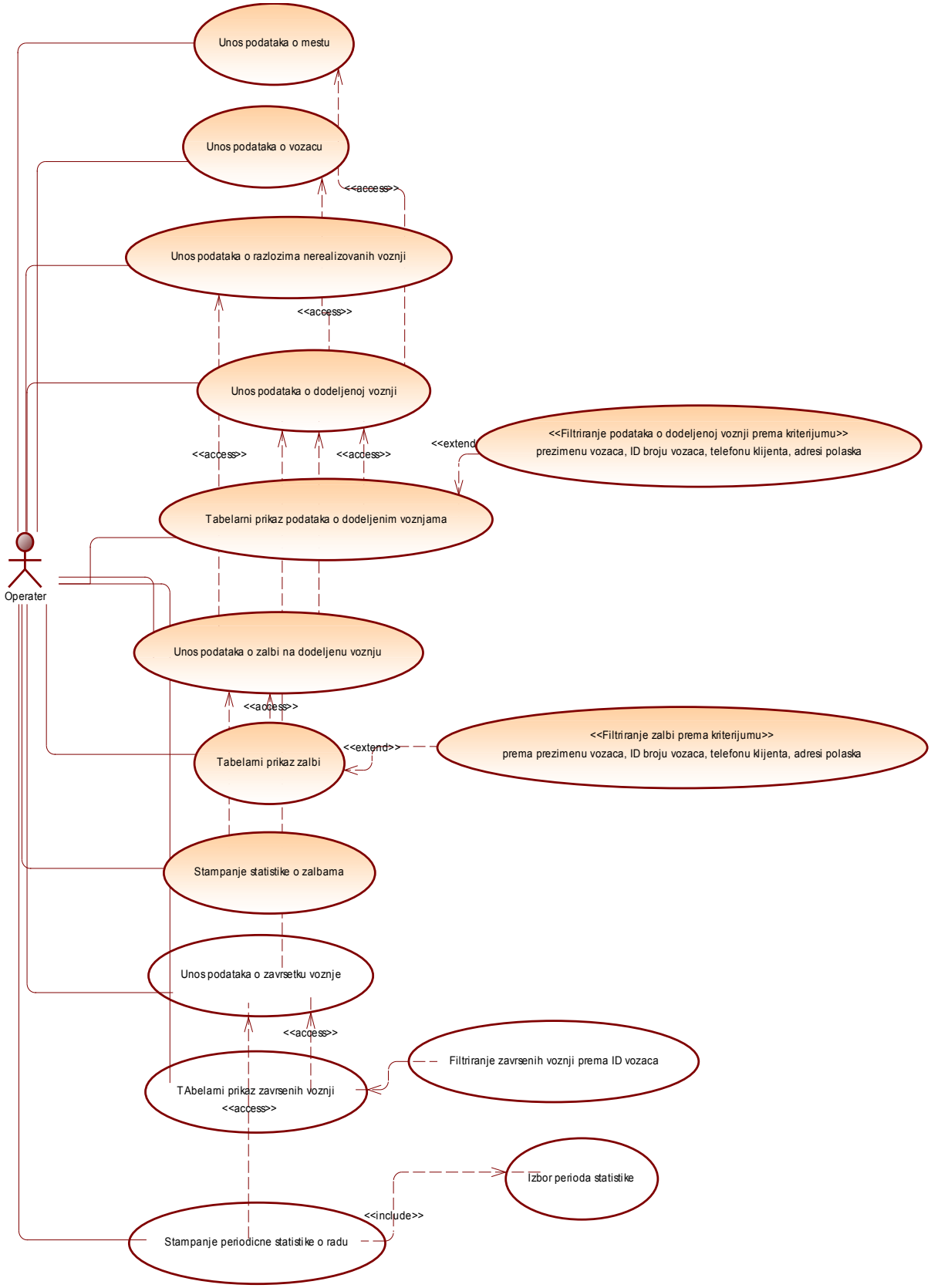
4.1. Preslikavanje primitivnih poslovnih procesa u softverske funkcije

PRIMITIVNI PROCES	**NIJE SOFTVERSKI PODRŽANO**	**SOFTVERSKA FUNKCIJA 1. PRIORITETA**
Prijem poziva od klijenta	telefonski	
Obavestavanje o licitaciji za voznju	radio stanica	
Prijem prijave za voznju	radio stanica	
Obavestavanje klijenta o mogucnosti prispeca vozila	telefon	
Prijem potvrde od klijenta za voznju		Unos podataka o dodeljenoj voznji
Obavestavanje vozaca o potvrdi za voznju		Unos podataka o dodeljenoj voznji
Prijem zalbe klijenta o neizvrsenoj voznji		Unos podataka o zalbi na dodeljenu voznju
Slanje zahteva za obrazlozenjem neizvrsene voznje vozacu	radio stanica	
Slanje obrazlozenja za neizvrsenu voznju klijentu	telefon	Unos podataka o zalbi na dodeljenu voznju
Evidentiranje izvrsene voznje		Unos podataka o zavrsetku voznje

Sve poslovne i softverske funkcije izvrsava operater.

PRIMITIVNI POSLOVNI PROCES	**SOFTVERSKA FUNKCIJA 1. prioritet**	**SOFTVERSKA FUNKCIJA 2. PRIORITET**	**SOFTVERSKA FUNKCIJA preduslov**
Prijem poziva od klijenta	Unos podataka o dodeljenoj voznji	Tabelarni prikaz dodeljenih voznji	Unos podataka o vozacu
Obavestavanje o licitaciji za voznju		Filtriranje dodeljenih voznji prema prezimenu vozaca, Id broju vozaca, telefonu klijenta, adresi polaska	Unos podataka o mestu
Prijem prijave za voznju			
Obavestavanje klijenta o mogucnosti prispeca vozila			
Prijem potvrde od klijenta za voznju			
Obavestavanje vozaca o potvrdi za voznju			
Prijem zalbe klijenta o neizvrsenoj voznji	Unos podataka o zalbi na dodeljenu voznju	Tabelarni prikaz zalbi	Unos podataka o razlozima nerealizovanih voznji
Slanje zahteva za obrazlozenjem neizvrsene voznje vozacu		Filtriranje zalbi po prezimenu vozaca, Id broju vozaca, telefonu klijenta, adresi polaska	Unos podataka o dodeljenoj voznji
Slanje obrazlozenja za neizvrsenu voznju klijentu		Stampanje statistike o zalbama	
Evidentiranje izvrsene voznje	Unos podataka o zavrsetku voznje	Tabelarni prikaz zavrsenih voznji	Unos podataka o dodeljenoj voznji
		Filtriranje zavrsenih voznji prema ID broju vozaca	
		Stampanje periodicne statistike o radu (ukljucuje: Izbor perioda statistike)	

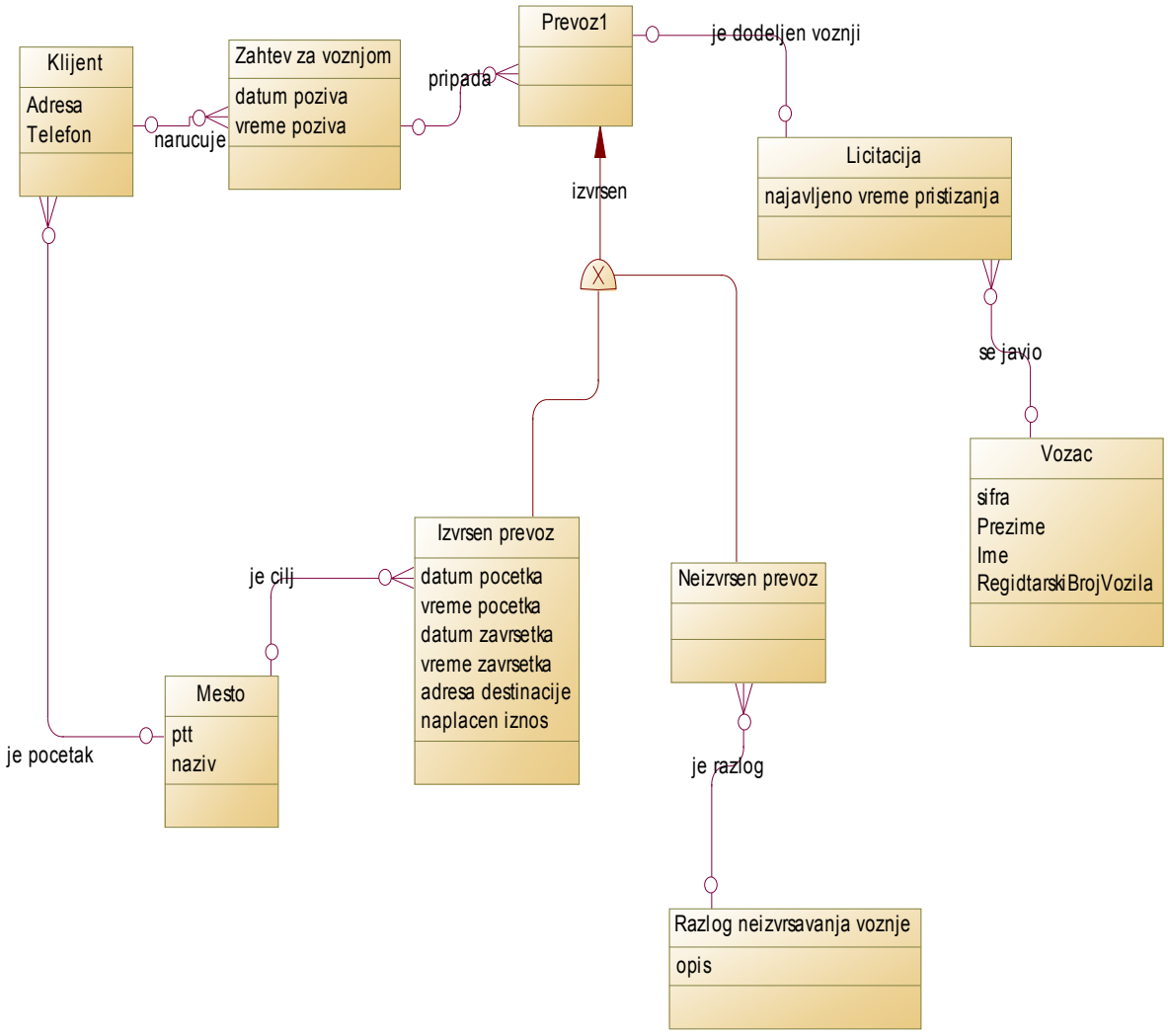
4.2. Dijagram slučajeva korišćenja



REŠENJA ZADATAKA

1. a) PRISTUP – GRAMATIČKA ANALIZA TEKSTA OPISA POSLA

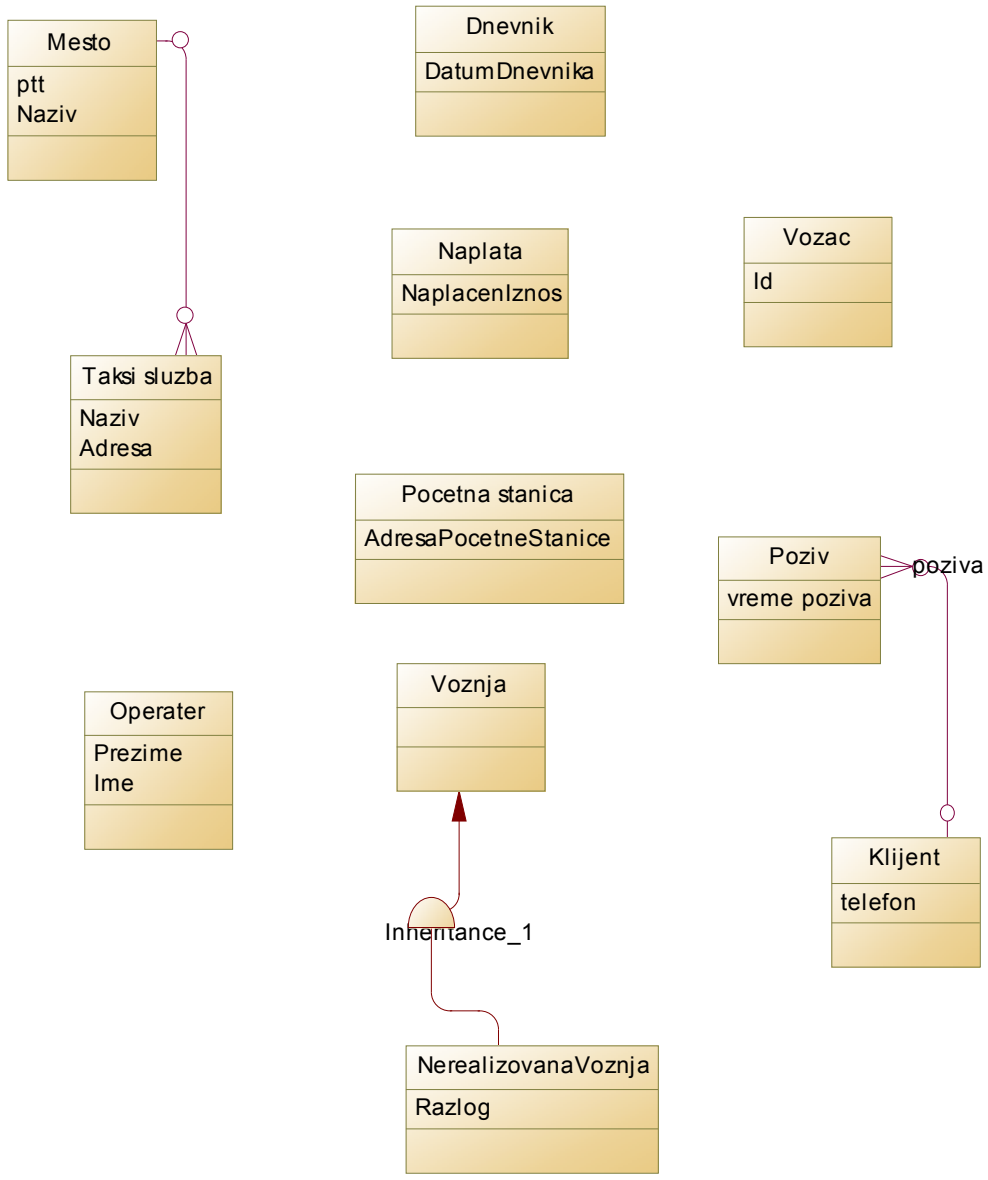
Metoda:
Imenice – kandidati za entitete, Glagoli – kandidati za poveznike. Ako poveznik ima vazne atribute, postaje entitet.



1.b) DODELJIVANJE ATRIBUTA IZ REČNIKA PODATAKA ENTITETIMA

Za primenu metode koristicemo spisak elementarnih podataka iz prethodne analize dokumenta.

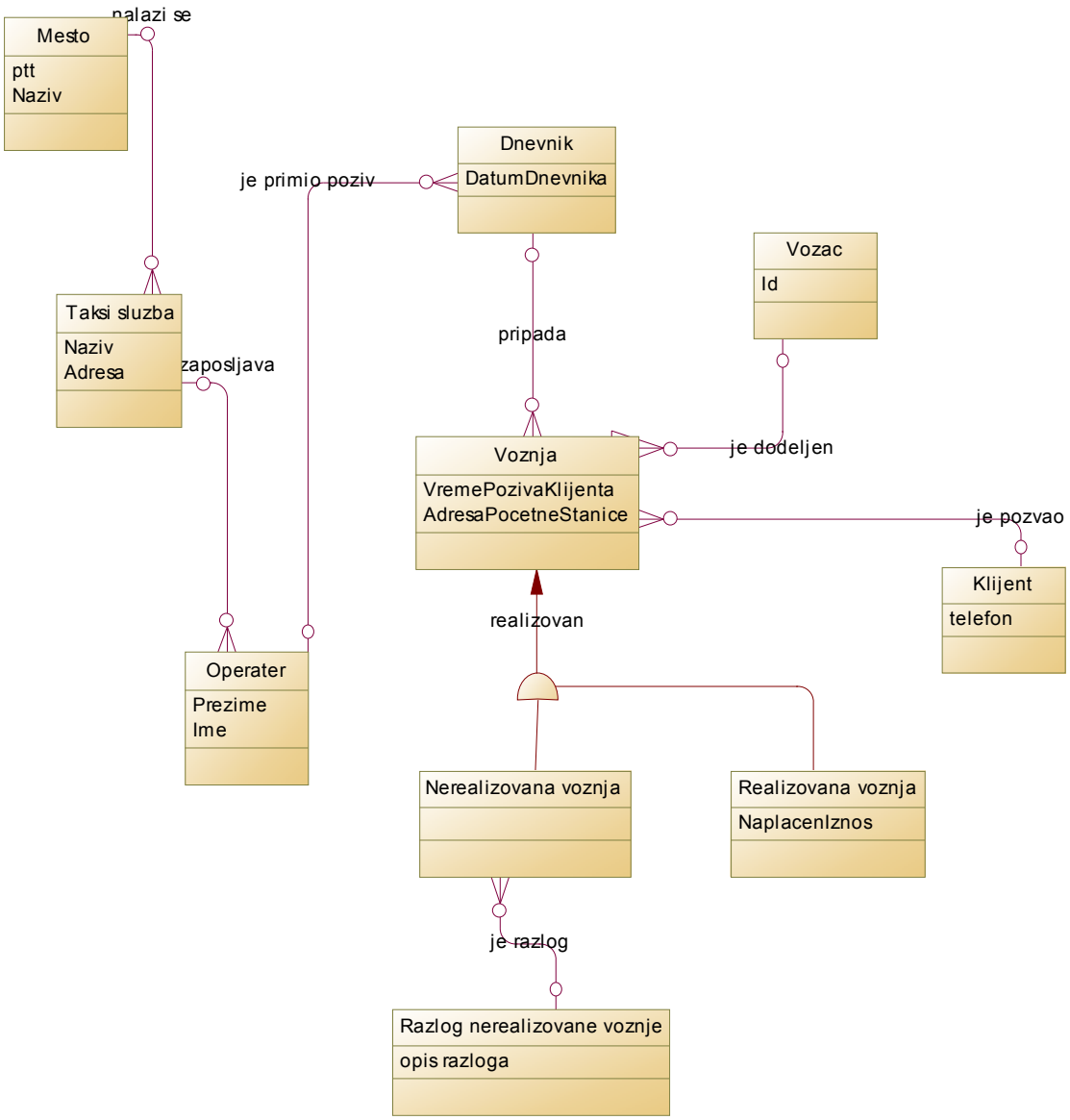
Metoda: Na osnovu naziva atributa odredjujemo naziv entiteta kojem pripada.



1. c) d) SINTAKSNA ANALIZA DOKUMENTA ili SKLADIŠTA PODATAKA

U nastavku je dat prikaz primene metode sintaksne analize dokumenta, a slican postupak bi se primenio i za sintaksnu analizu skladista podataka.

Metoda: Atributi koji su blisko povezani (udruženi simbolima <>) nalaze se u zajedničkom entitetu, atributi koji su u odnosu 1:M (odvojeni {}) se odvajaju u poseban entitet, atributi koji se međusobno isključuju odvajaju se u is-a hijerarhiju.



1 e) NORMALIZACIJA SKLADIŠTA PODATAKA

Metoda: Skladište podataka smatramo nenormalizovanom tabelom relacionog modela i primenjujemo svođenje na:

1. normalnu formu (atributi treba da imaju atomarne vrednosti – nema množine u nazivu ili naziva kao strukture),
2. normalnu formu (funkcionalna zavisnost od identifikacionih obelezja:
   f(kljucno identifikaciono obelezje)= jedna jedinstveno odredjena vrednost nekljucnog obelezja),
2. normalnu formu (nema tranzitivnih funkcionalnih zavisnosti, nema izracunljivih polja, polja sa domenima – novi entiteti).

NAPOMENA:
Na ovaj nacin direktno kreiramo konceptualni model (prvenstveno uvodimo identifikaciona obelezja, izdvajamo u posebne entitete atribute koji su identifikaciono zavisni od identifikacionog obelezja i povezujemo entitete poveznicima) koji treba da zadovolji osobine 1,2,3 normalne forme relacionog modela.

1. normalna forma

NAZIV ELEMENTARNOG PODATKA	JEDNINA u nazivu	STRUKTURA u nazivu	Napomena	1. NF
NazivTaksiSluzbe	da	ne		da
AdresaTaksiSluzbe	da	ne		da
NazivMestaTaksiSluzbe	da	ne	bilo bi da pise MestoTaksiSlužbe	da
DatumDnevnikaRada	da	ne	datumski tip je slozen od dana, meseca i godine, ali postoji standardni tip podatka Date	da
IDBrojVozaca	da	ne		da
VremePozivaKlijenta	da	ne	vreme se sastoji od sata, minuta i sekundi, ali postoji tip podatka Time	da
TelefonPozivaKlijenta	da	ne	telefon je struktura od pozivnog broja i samog broja, ali se moze smatrati jednim podatkom	da
AdresaPocetneStanice	da	ne	adresa se sastoji od ulice i broja, ali se moze smatrati jednim podatkom	da
NaplaceniIznos	da	ne		da
RazlogNerealizovaneVoznje	da	ne		da
PrezimeOperatera	da	ne		da
ImeOperatera	da	ne		da

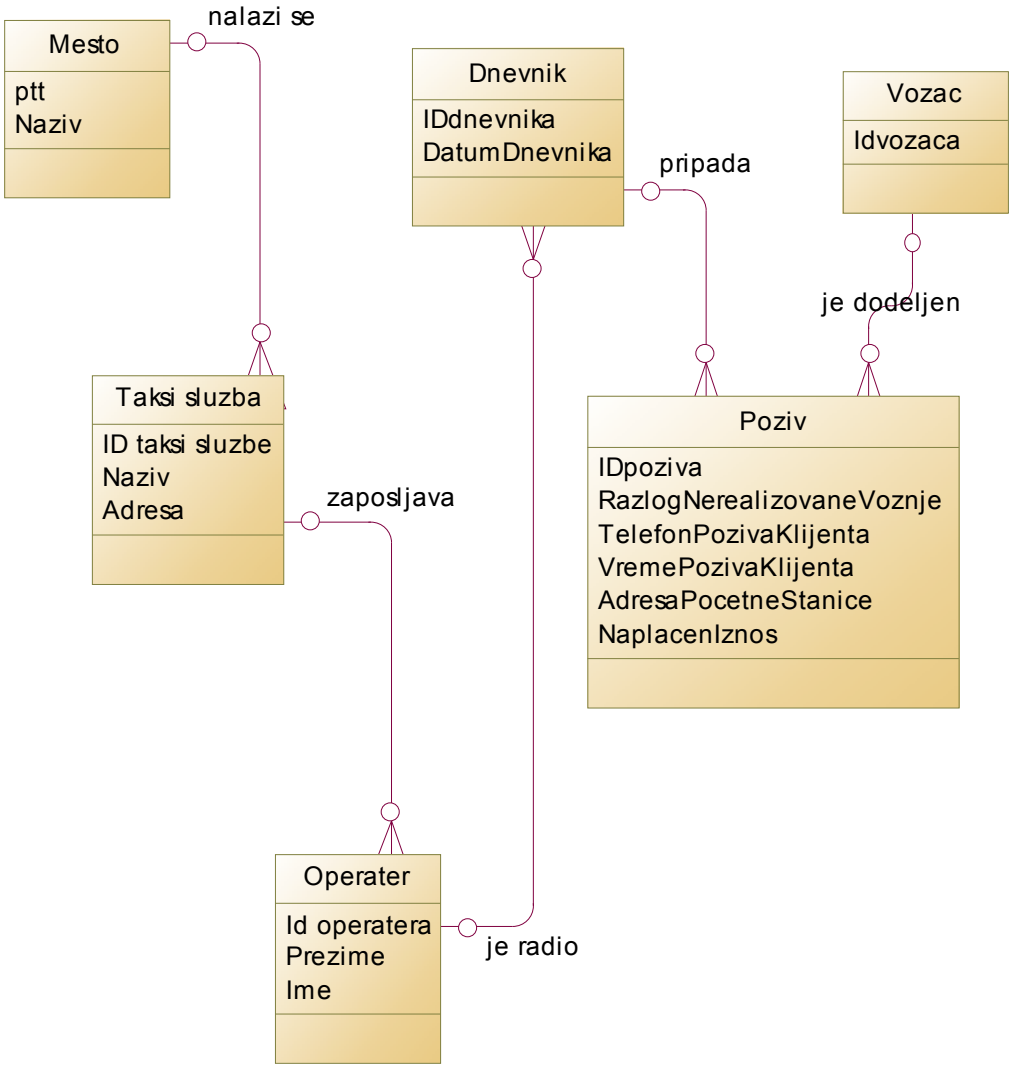
2. normalna forma

NAZIV ELEMENTARNOG PODATKA	FUNKCIONALNO (identifikaciono) ZAVISI OD
NazivTaksiSluzbe	ID taksi sluzbe
AdresaTaksiSluzbe	ID taksi sluzbe
NazivMestaTaksiSluzbe	PTT mesta ptt mesta taksi sluzbe zavisi od ID taksi sluzbe
DatumDnevnikaRada	Id dnevnika rada
IDBrojVozaca	IDBrojVozaca
VremePozivaKlijenta	ID stavke dnevnika rada ili ID Poziva, Id poziva zavisi od ID dnevnika rada
TelefonPozivaKlijenta	ID stavke dnevnika rada ili ID poziva
AdresaPocetneStanice	ID poziva Id poziva zavisi od ID dnevnika rada
NaplaceniIznos	ID stavke dnevnika rada Id poziva zavisi od ID dnevnika rada
RazlogNerealizovaneVoznje	ID stavke dnevnika rada Id poziva zavisi od ID dnevnika rada
PrezimeOperatera	Id operatera Id operatera zavisi od ID taksi sluzbe
ImeOperatera	Id operatera

3. normalna forma

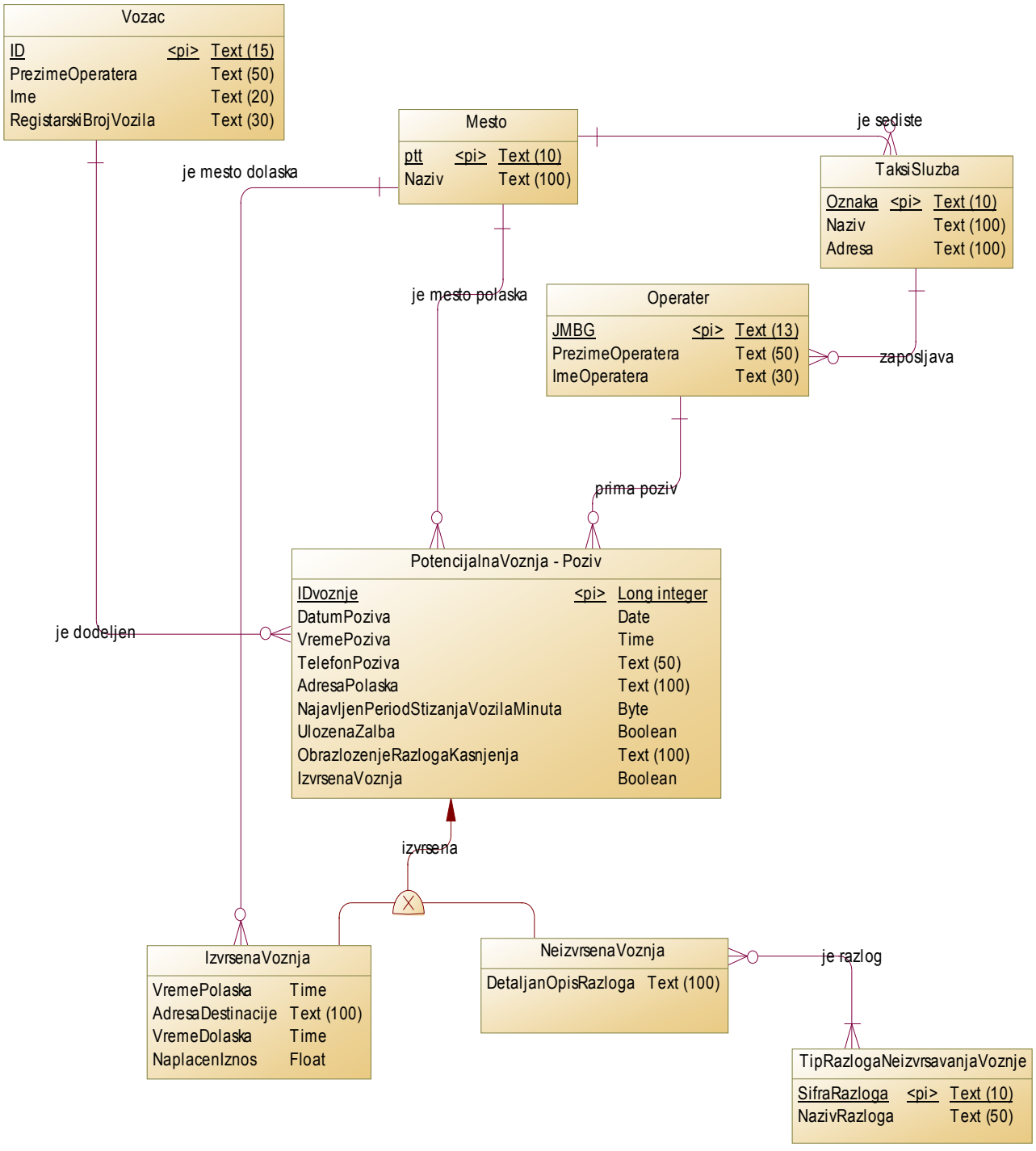
NAZIV ELEMENTARNOG PODATKA	FUNKCIONALNO (identifikaciono) ZAVISI OD	TRANZITIVNO ZAVISI OD
NazivTaksiSluzbe	ID taksi sluzbe	
AdresaTaksiSluzbe	ID taksi sluzbe	
NazivMestaTaksiSluzbe	PTT mesta	ID taksi sluzbe
DatumDnevnikaRada	Id dnevnika rada	Id operatera
IDBrojVozaca	IDBrojVozaca	ID Poziva
VremePozivaKlijenta	ID Poziva	ID dnevnika rada
TelefonPozivaKlijenta	ID Poziva	ID dnevnika rada
AdresaPocetneStanice	ID Poziva	ID dnevnika rada
NaplaceniIznos	ID Poziva	ID dnevnika rada
RazlogNerealizovaneVoznje	ID Poziva	ID dnevnika rada
PrezimeOperatera	Id operatera	ID taksi sluzbe
ImeOperatera	Id operatera	ID taksi sluzbe

Konceptualni model nakon svodjenja na 1,2,3 normalne forme:



2.INTEGRACIJA MODELA, PROVERA KOREKTNOSTI I POTPUNOSTI

2 a) Integracija modela



NAPOMENA:

Šifarnici su: Mesto, Taksi služba, Vozač, Operater, TipRazlogaNerealizovanjaVožnje

2.b) PROVERA KOREKTNOSTI MODELA

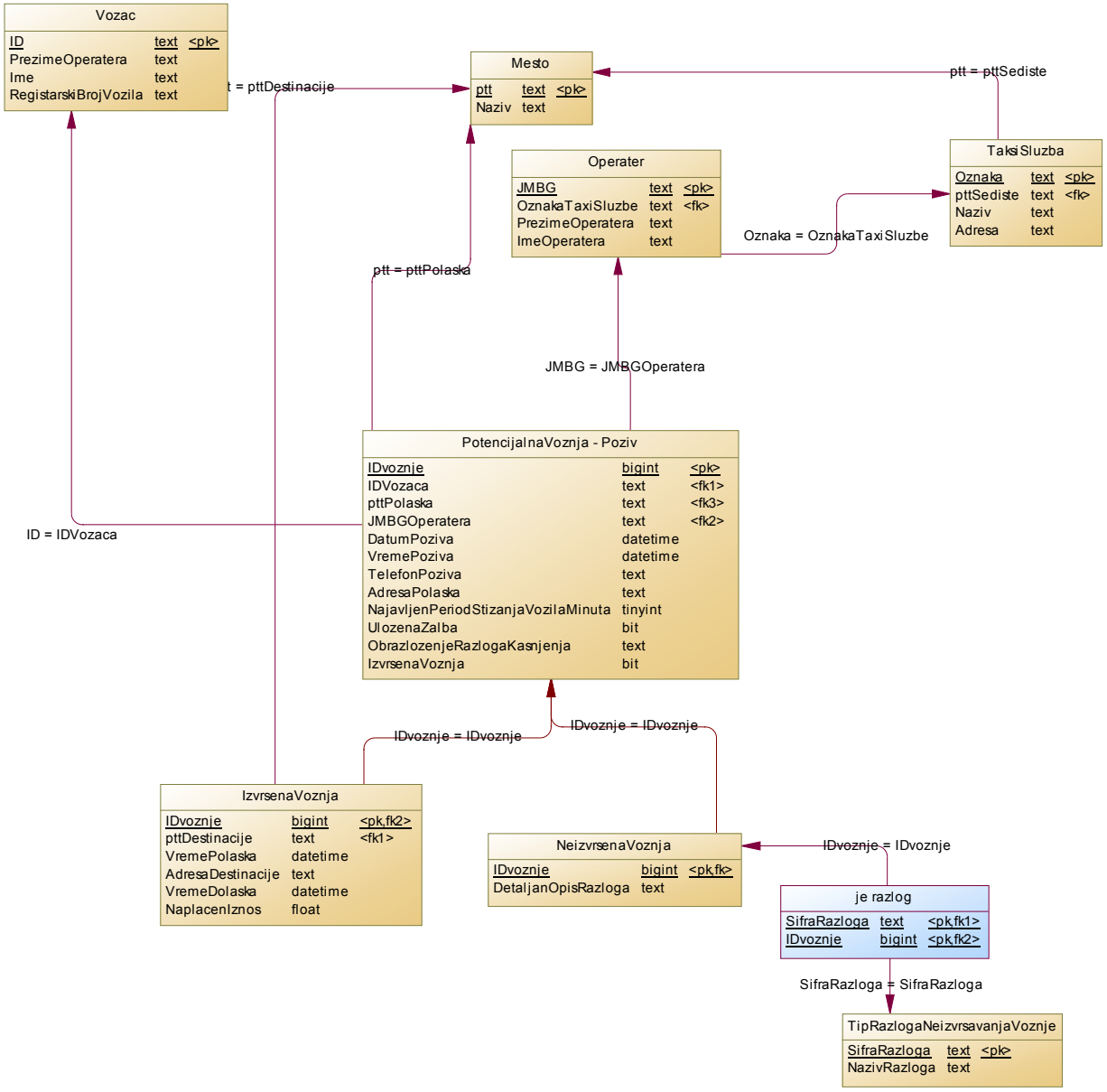
GRUPA ELEMENATA	GRUPA OSOBINA	Potrebna osobina - METRIKA	PROVERA
CELINA MODELA	semantika	Semanticki u skladu sa problemom	DA
	uskladjenost sa datim materijalom	Pokrivenost opisa posla	DA
		Pokrivenost importovanih elementarnih podataka	DA
		Pokrivenost skladista podataka sa DTP	DA
	kompletnost	Kompletnost skupa entiteta	da
		Kompletnost skupa atributa	da
		Kompletnost skupa poveznika	da
	jezicka nedvosmislenost	Nepostojanje sinonima u nazivima entiteta i atributa	DA
		Nepostojanje homonima u nazivima entiteta i atributa	DA
ENTITET	naziv	Imenica ili glagolska imenica	DA
		Izrazava sustinske objekte i dogadjaje	DA
		Ne izrazava nazive dokumenata	DA
		Jednina u nazivu	DA
	Entitet i atributi	Ima identifikaciono obelezje ili je identifikaciono zavisan od drugih entiteta	DA
		Dodeljen atribut odgovarajućem entitetu	DA
		(2NF) Ne postoje brojni odnosi atributa 1:M u entitetu	DA
		(3NF) Ne postoje tranzitivne zavisnosti atributa u entitetu	DA
ATRIBUT	Naziv	(1NF) Jednina u nazivu	DA
		(1NF) Nema naziv kao struktura	DA
	Atributi i tipovi /domeni podataka	Odgovarajuci tip podatka	DA
		Atribut nema za tip podatka domen (uzi skup vrednosti nego sto je standardni tip podatka), vec je izvucen kao sifarnik	DA – izdvojeni su sifarnici
	Karakteristike	Atributi se sa istim nazivom ne ponavljaju (nema redundanse)	DA
		Nema izracunljivih atributa, vec samo analitickih vrednosti medju atributima na osnovu kojih se vrse izracunavanja	DA
POVEZNIK	Naziv	Poveznik ima naziv	DA
		Naziv kao glagol	DA
	nacin povezivanja entiteta	Povezani direktno entiteti koji treba da budu povezani	da
		Entiteti su povezani hronoloskim sledom medjuzavisnih dogadjaja	DA
		Povezani su entiteti u gerund odnosu, ako je potrebno	nema gerunda
		Ako poveznik ima vazne atribute, pretvoren je u entitet	da
	podesavanja osobina	Pravilno podesen kardinalitet	da
		Pravilno podesena strana na vezi 1:M	da
		Pravilno odredjeno da li ima opravdanja za vezu 1:1 ili je jedan entitet	nema odnosa 1:1
		Pravilno odredjeno u vezi 1:1 ko je dominant	nema odnosa 1:1
		Pravilno odredjeno u vezi M:M da li ima vaznih atributa ili ostaje M:M	ostaje veza M:M
		Pravilno podesena identifikaciona zavisnost	da
		Pravilno podesena is-a hijerarhija (gde je nadredjeni, bar jedan podredjeni, migrira samo ID)	da

3. c) PROVERA POTPUNOSTI MODELA U ODNOSU NA PRIMITIVNE PROCESE I SOFTVERSKE FUNKCIJE

PRIMITIVNI PROCES	SOFTVERSKA FUNKCIJA	ENTITETI PODMODELA
POMOCNA DELATNOST	Unos podataka o taksi službi	Taksi sluzba
	Unos podataka o operateru	Operater
	Unos podataka o vozacu	Vozac
	Unos podataka o mestu	Mesto
	Unos podataka o tipovima razloga nerealizovanih voznji	TipRazlogaNerealizovaneVoznje
OSNOVNA DELATNOST		
Prijem poziva od klijenta	NIJE PODRZANO SOFTVERSKI	
Obavestavanje o licitaciji za voznju	NIJE PODRZANO SOFTVERSKI	
Prijem prijave za voznju	NIJE PODRZANO SOFTVERSKI	
Obavestavanje klijenta o mogucnosti prispeca vozila	NIJE PODRZANO SOFTVERSKI	
Prijem potvrde od klijenta za voznju	Unos podataka o dodeljenoj voznji Tabelarni prikaz dodeljenih voznji Filtriranje dodeljenih voznji prema prezimenu vozaca, Id broju vozaca, telefonu klijenta, adresi polaska	Taksi sluzba, Operater, Vozac, PotencijalnaVoznja - Poziv
Obavestavanje vozaca o potvrdi za voznju	Unos podataka o dodeljenoj voznji	
Prijem zalbe klijenta o neizvrsenoj voznji	Unos podataka o zalbi na dodeljenu voznju	PotencijalnaVoznja – Poziv
Slanje zahteva za obrazlozenjem neizvrsene voznje vozacu	Unos podataka o zalbi na dodeljenu voznju	PotencijalnaVoznja – Poziv
Slanje obrazlozenja za neizvrsenu voznju klijentu	Tabelarni prikaz zalbi Filtriranje zalbi po prezimenu vozaca, Id broju vozaca, telefonu klijenta, adresi polaska	PotencijalnaVoznja – Poziv
Evidentiranje izvrsene voznje	Unos podataka o zavrsetku voznje Tabelarni prikaz zavrsenih voznji Filtriranje zavrsenih voznji prema ID broju vozaca Unos podataka o razlozima nerealizovanih voznji	IzvrsenaVoznja NeizvrsenaVoznja
UPRAVLJACKA DELATNOST	Stampanje periodicne statistike o radu (ukljucuje: Izbor perioda statistike)	PotencijalnaVoznja, IzvrsenaVoznja, NeizvrsenaVoznja
	Stampanje statistike o zalbama	PotencijalnaVoznja – Poziv

Zaključak:
Model je korektan i potpun u skladu sa primitivnim procesima i slučajevima korišćenja.

3.GENERISAN FIZIČKI (relacioni) MODEL

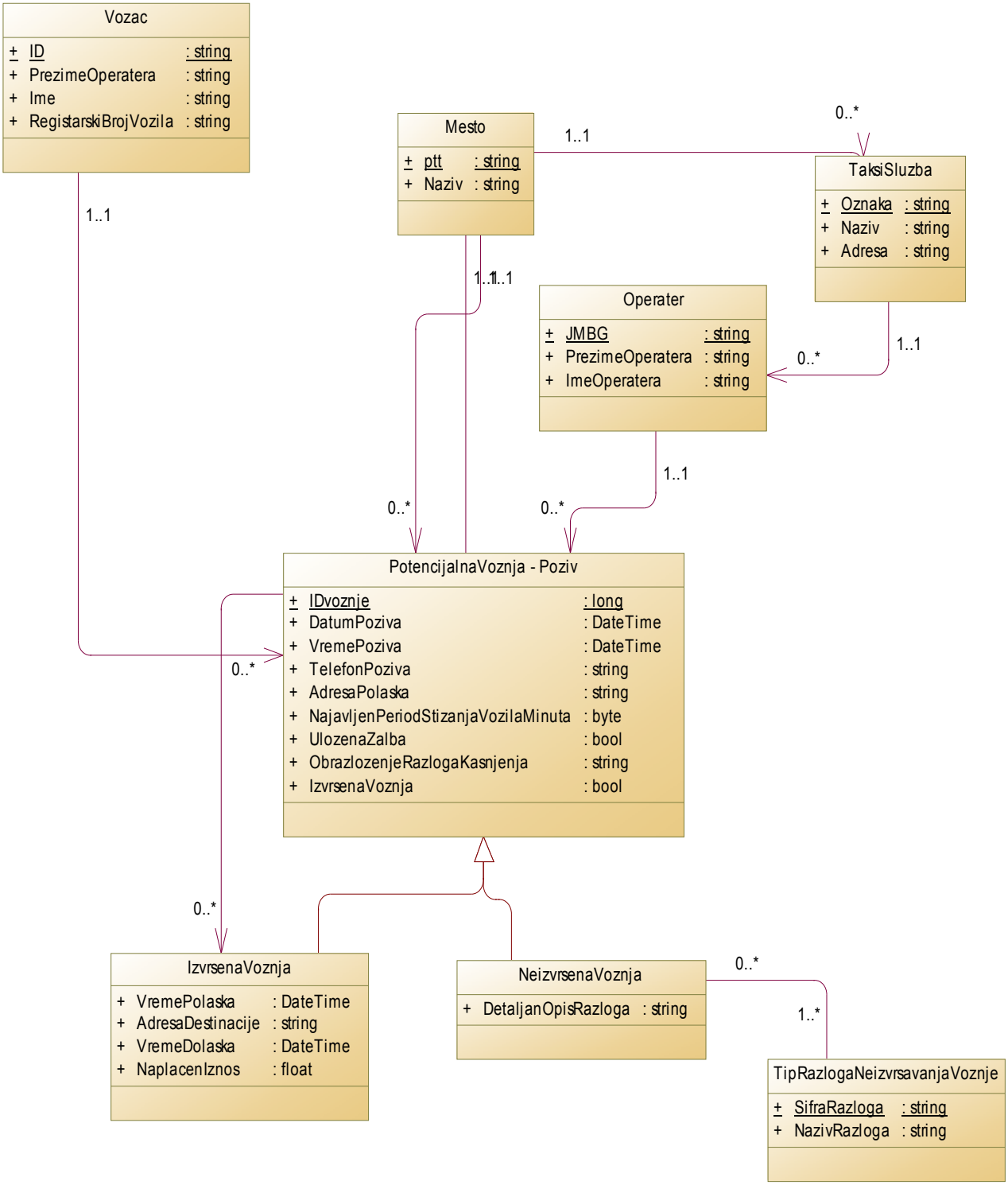


Možemo izvršiti:

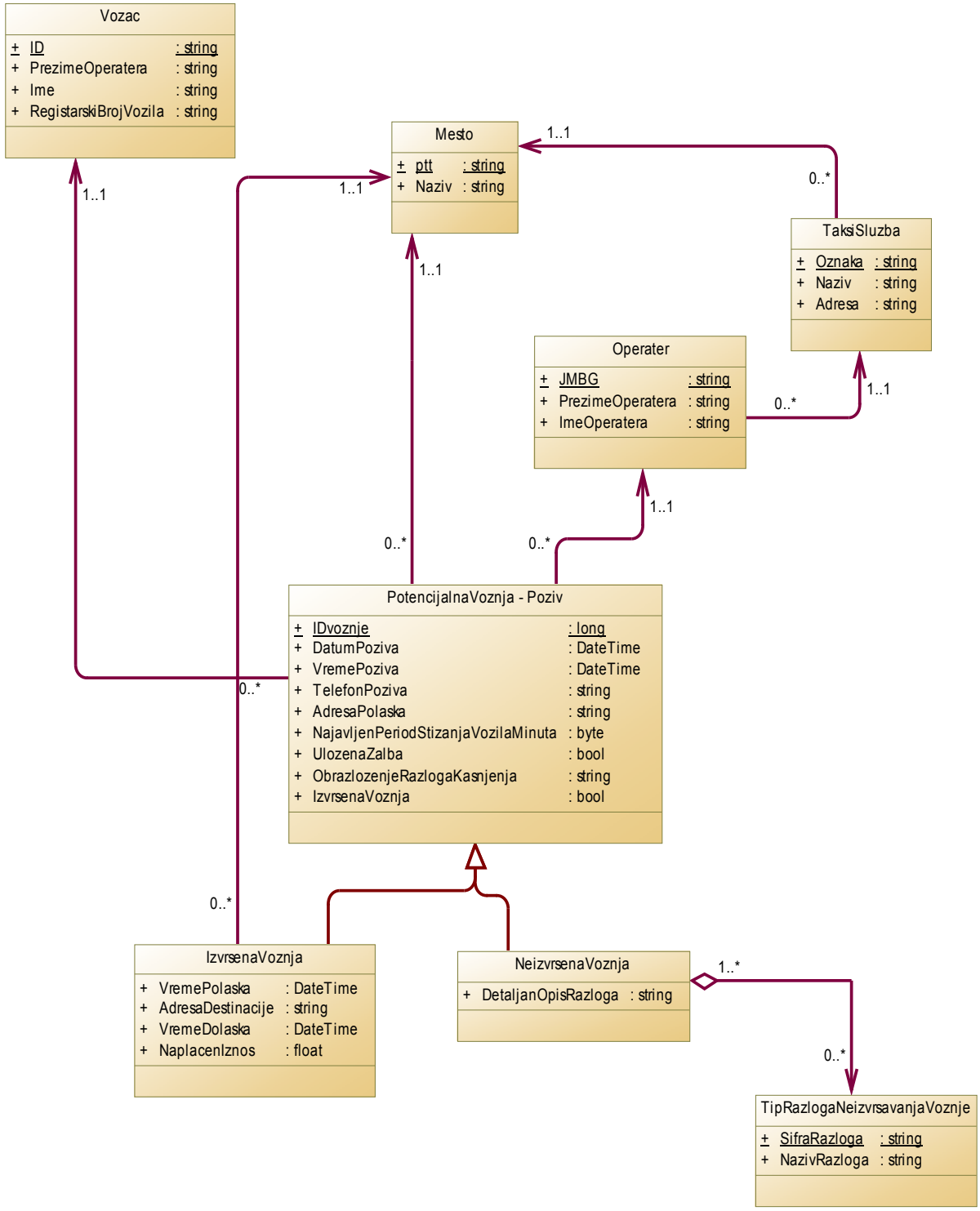
1. Izmenu naziva polja koje predstavlja strani ključ u nekoj tabeli. Npr. oznaka iz tabele Taksi služba kad je migrirala u tabelu Operater „pojavila" se tamo kao „oznaka". Bilo bi razumljivije da dobije drugačije ime i zato menjamo naziv polja u „OznakaTaksiSlužbe".
2. Dodavanje alternativnih ključeva, tj. unique indeksa nad:
   - Nazivom mesta – za naziv mesta ne smemo staviti indeks, jer postoje čak i u jednoj zemlji više mesta sa istim nazivom
   - Nazivom taxi službe – za ovaj primer ima smisla postaviti indeks

5. DIJAGRAM KLASA

5 a) GENERISAN DIJAGRAM KLASA NA OSNOVU KONCEPTUALNOG DIJAGRAMA



5 b) KORIGOVAN DIJAGRAM KLASA



Korekcije su:
1. Usmerenje strelice kod veze tipa asocijacije je bilo pogrešno (osobina „Navigable”.
2. TipRazlogaNeizvrsavanjaVoznje je umesto asocijacije postavljen da bude u odnosu agregacije prema klasi NeizvrsenaVoznja.

7. RAD SA BAZOM PODATAKA

Rad sa bazom podataka u ovom odeljku se odnosi na operacije sa bazom podataka u okviru rada u sistemu za upravljanje bazom podataka (DataBase Management System - DBMS):

1. Kreiranje baze podataka
   - Koristeći DBMS kao vizualni alat za kreiranje
   - Koristeći SQL komande ili skup SQL komandi (skript) u okviru DBMS-a
   - Transformacijama iz drugih tipova baza podataka - koristeci ODBC standard
   - Eksportovanjem ili Importovanjem podataka iz drugih formata (XML, XLS...)
2. Ažuriranje i izdvajanje (pretragu) podataka iz baze podataka – korišćenje SQL upita

Baza podataka može biti kreirana i iz CASE alata, što je ranije opisano (videti odeljak 6.3).

7.1. STRUKTURA I TIPOVI PODATAKA U RAZLIČITIM DBMS

U nastavku je dat uporedni pregled struktura baza podataka (Tabela 1) kao i tipova podataka (Tabela 2) za MS Access 2000, MS SQL Server 2008 R2 i Oracle 11g Express.

Tabela 1. STRUKTURA BAZE PODATAKA u različitim DBMS

MS ACCESS	MS SQL SERVER	ORACLE
Tabele- Tables: • Columns • Keys • Indexes	Tables: • Columns • Keys • Constraints • Triggers • Indexes • Statistics	**Table:** • Columns • Data • Indexes • Constraints • Grants • Statistics • UI defaults • Triggers • SQL
Relacije - Relationships	Database diagrams	**Table** -Model -Dependencies
Forme- Forms		
Izvestaji – Reports		
SQL upiti – Query		
	Views	Views, Materialized Views
	Synonyms	Synonyms
Makroi - Macro	Programmability: • Stored procedures • Functions • Database triggers • Assemblies • Types • Rules • Defaults	Types Procedures Functions Triggers
	Security	
		Indexes
		Sequences
		Packages
		Database links

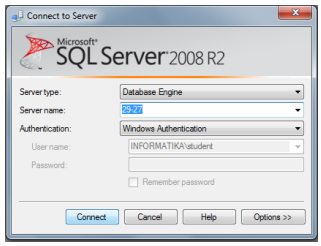
Tabela 2. TIPOVI PODATAKA u različitim DBMS

Tip podatka	**MS ACCESS**	**MS SQL SERVER**	**ORACLE**
String fiksne duzine		Char(n), nChar(n)	char(n), nchar(n)
String	Text	VarChar(n), nVarChar (n)	
Veliki tekst	Memo	nText, text	
Boolean	Yes/No	Bit	
Numericki:		Numeric (n,m)	Number(n,m)
Bit		Bit	
Bajt	Byte		
		tinyint	
		Small int	
Integer	Integer	int	
Long integer	Long integer	Bigint	Long
Realan broj manji	Single	real	
Realan broj veci	Double	Float	Float
Specificni:	Replication ID		
	Decimal	Decimal	
			Binary float
			Binary double
Datum i vreme		Small datetime	
	DateTime	DateTime	Date
		Small money	
		timestamp	TimeStamp
			Time Stamp with Time Zone
			Time Stamp with local time zone
			Interval Year to Month
			Interval day to second
Novac	Currency	Money	
Automatski brojac	Autonumber	Uniqueidentifier	
			RowID
			URowID
Razno:	OLE object		
	Hyperlink		
		Binary(n), VarBinary(n)	
		Image	
		SQL variant	
		XML	
			RAW
			Long Raw
			CLOB, NCLOB (Character Large object, National character large object)
			BLOB (binary large object)
			BFILE

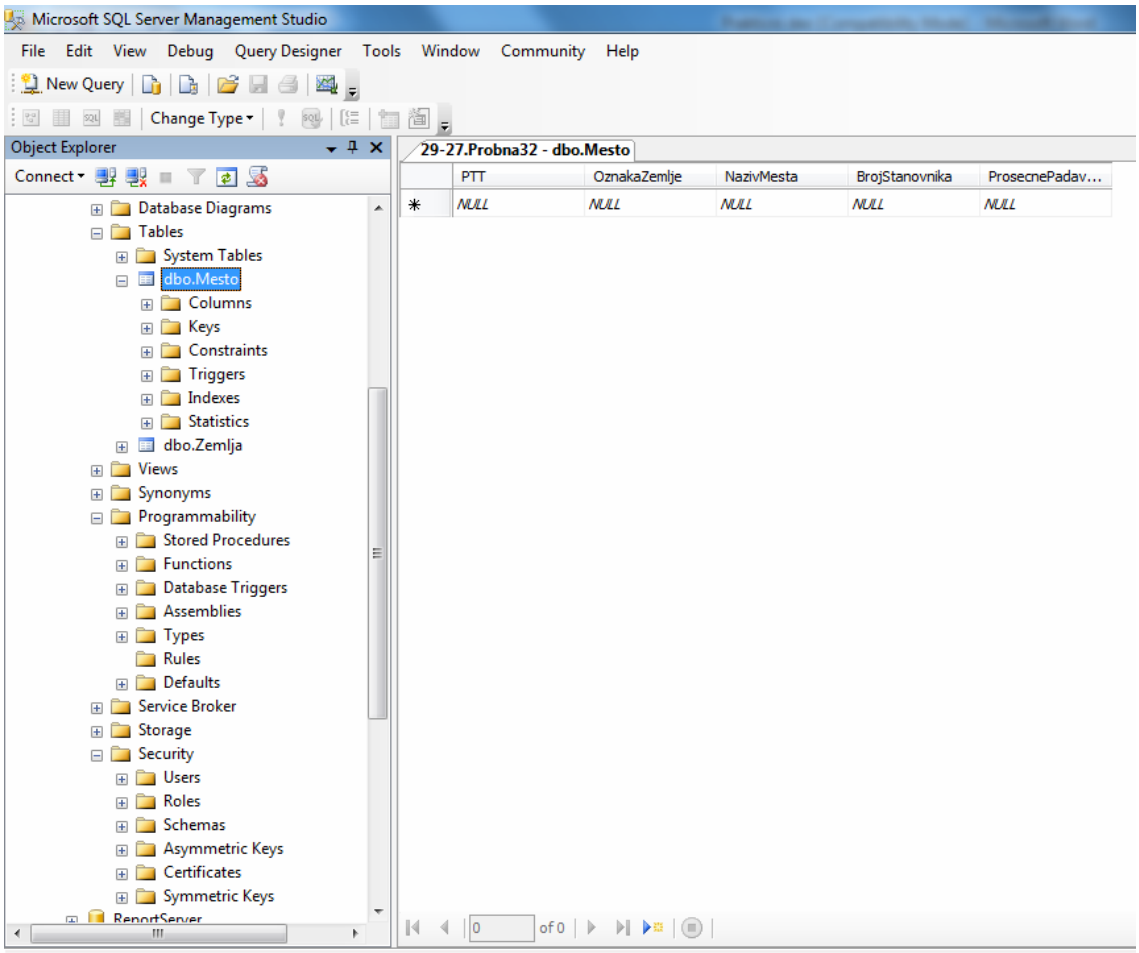
7.2. MS SQL SERVER BAZA PODATAKA

7.2.1. Radno okruženje

Nakon pokretanja i autentifikacije:



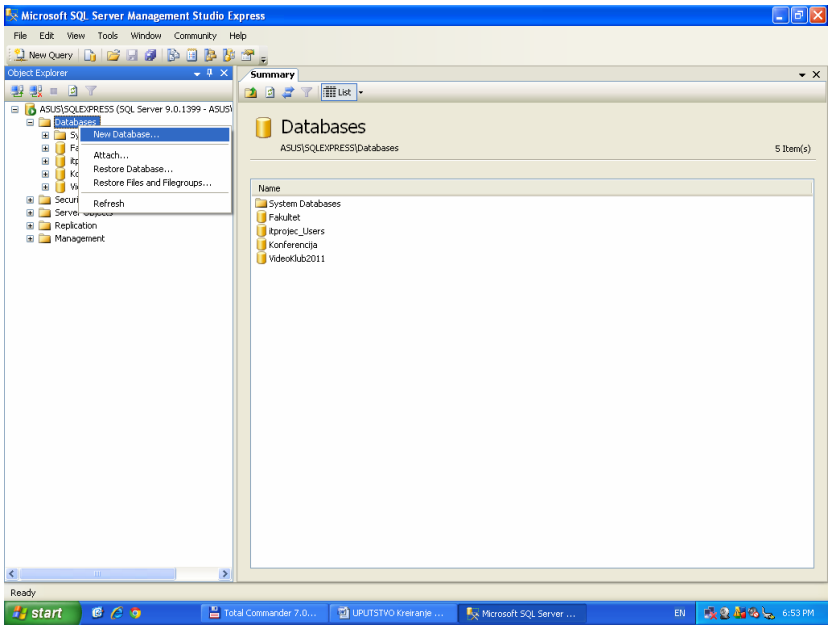
Dobijamo prostor za rad, gde se u levom delu nalazi spisak baza podataka i mogućnosti pristupa i rada sa odgovarajućim strukturama:



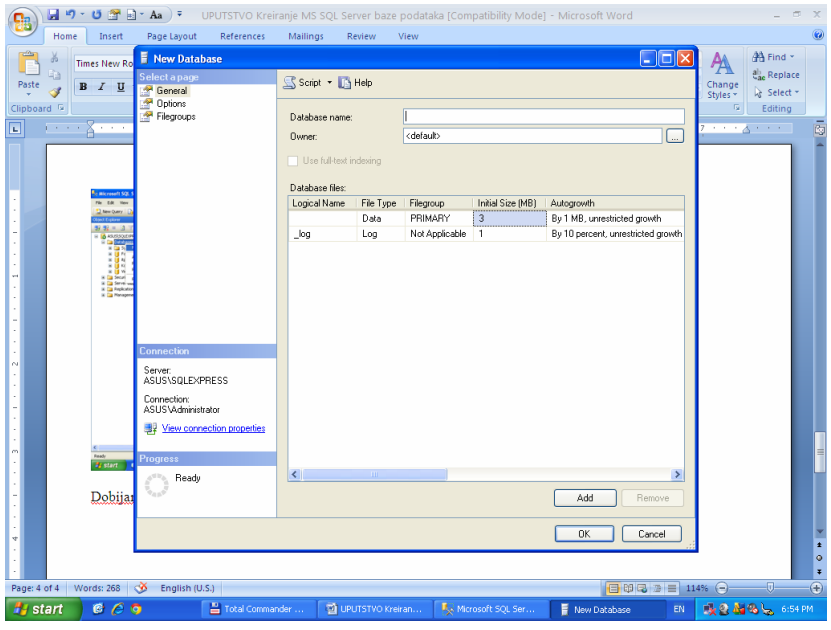
7.2.2. Kreiranje baze podataka

7.2.2.1. Kreiranje tabela i relacija baze podataka vizualnim alatima DBMS-a

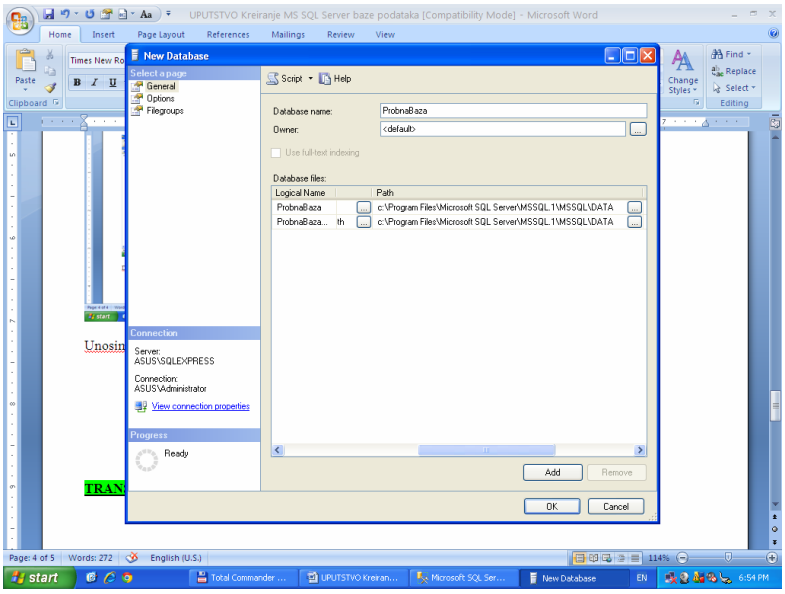
Pokrećemo kreiranje baze podataka – Databases, desni taster: New Database.



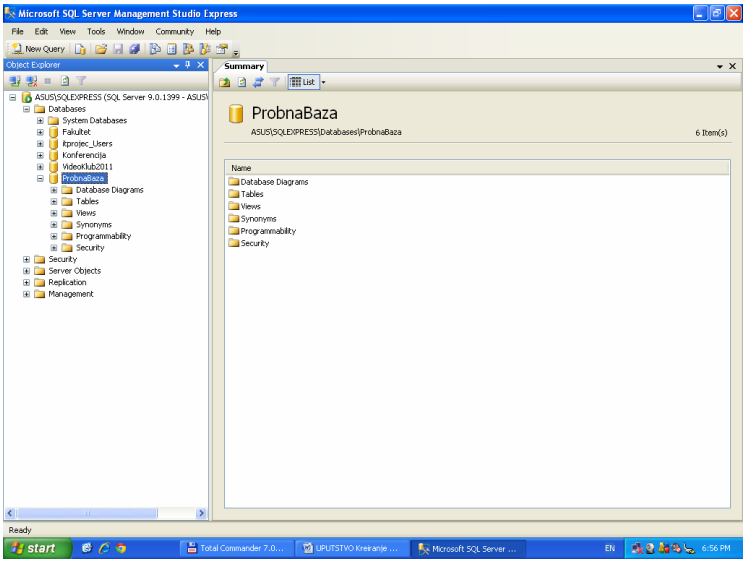
Dobijamo:



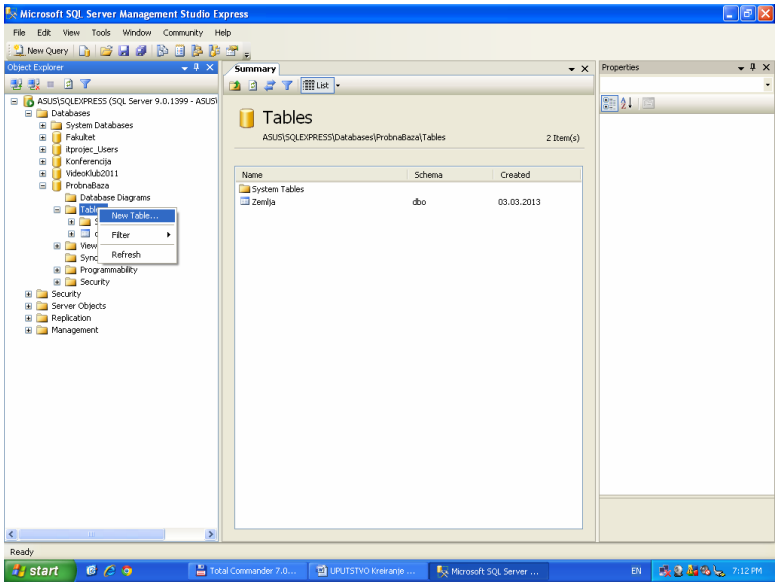
Unosimo naziv baze podataka i proveravamo putanju gde je fajl baze snimljen:



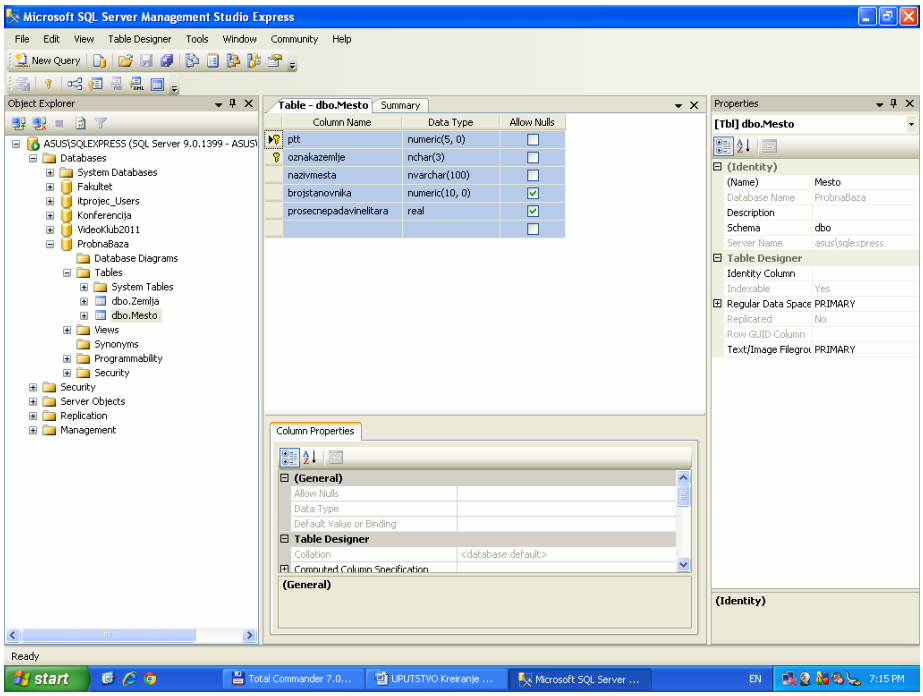
Biramo opciju OK. Dobijamo:



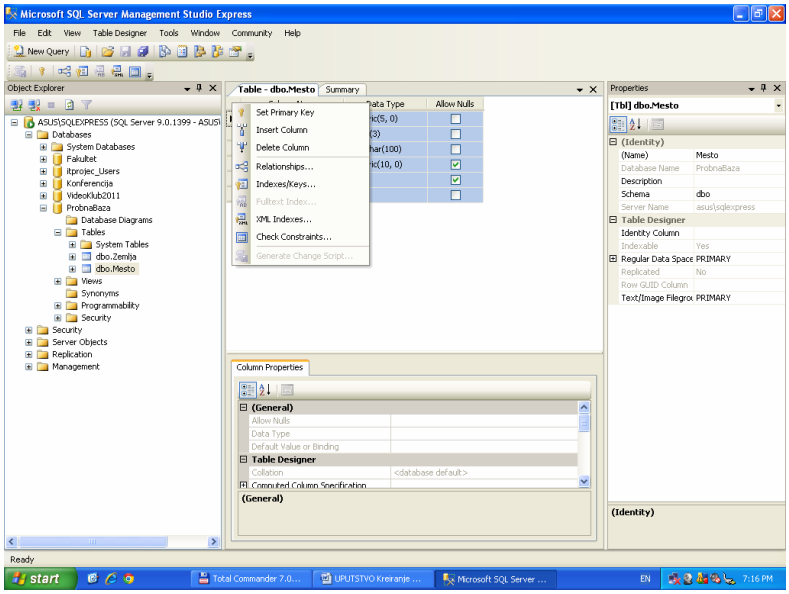
Dodajemo novu tabelu u bazu podataka:



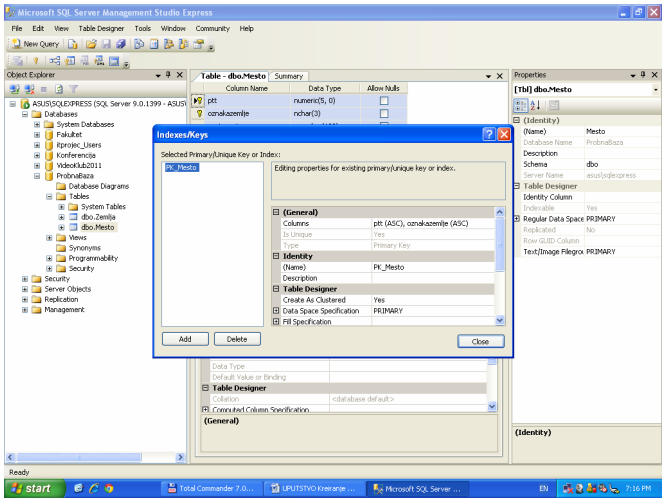
Unosimo polja, podešavamo tipove polja, null/not null osobine i primarni ključ.



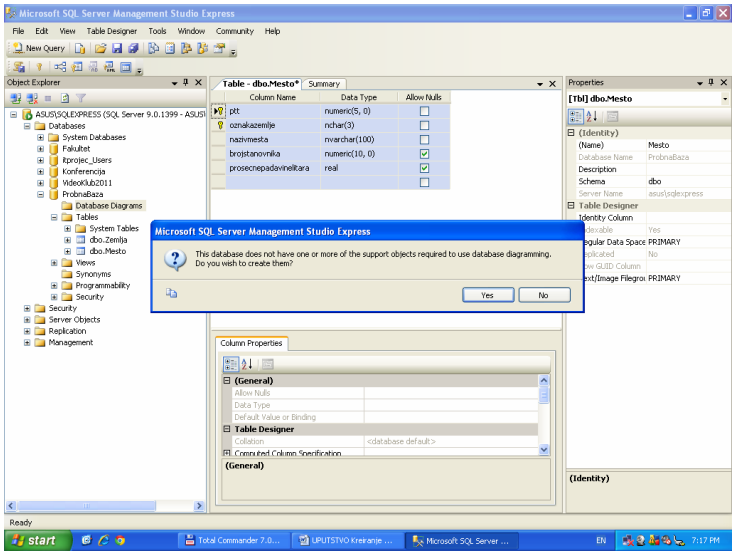
Proveravamo osobine tabele:



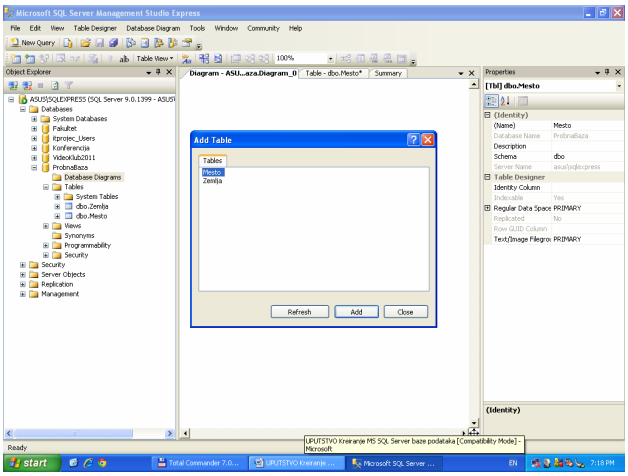
Proveravamo indekse:



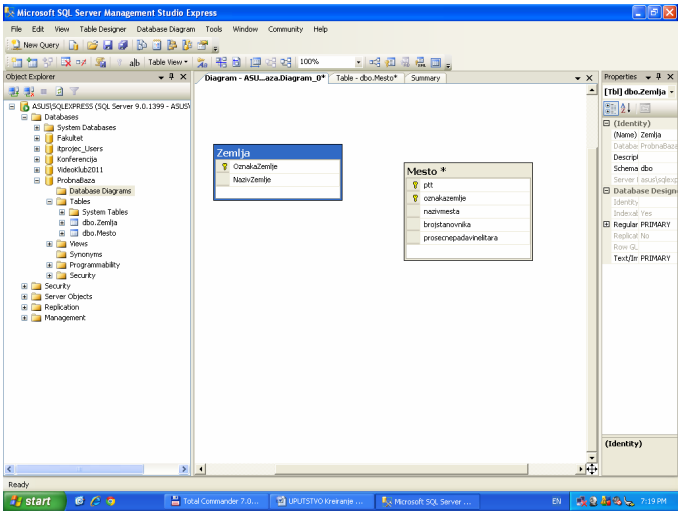
Dodajemo relacije na Database diagrams:



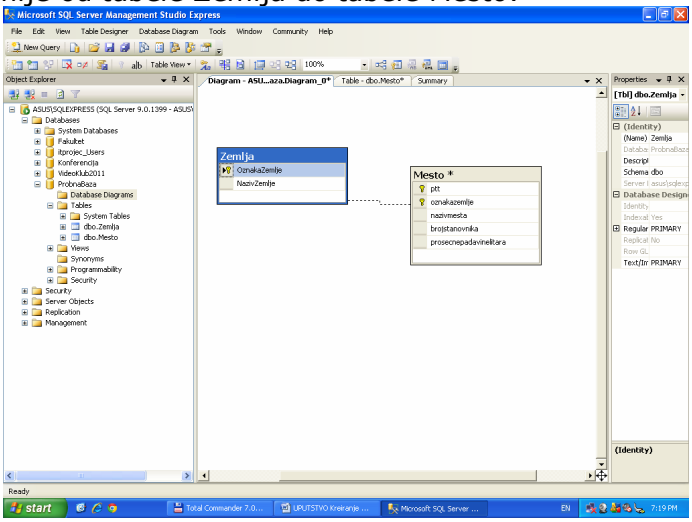
Dodajemo tabele:



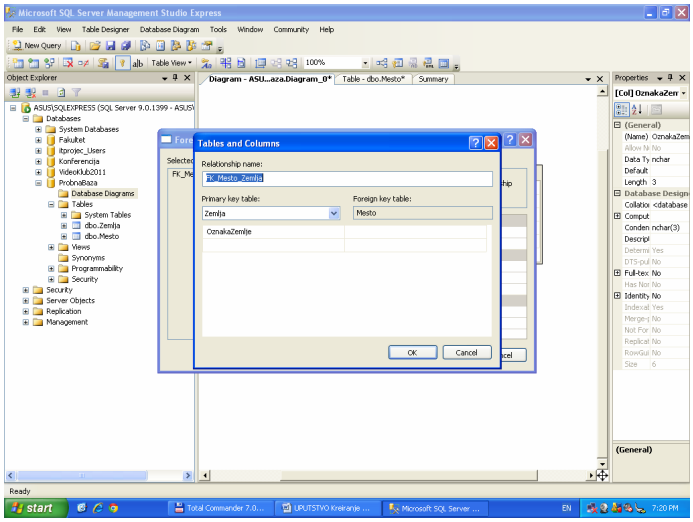
Preduslov za kreiranje relacije - obe tabele moraju imati primarni ključ:



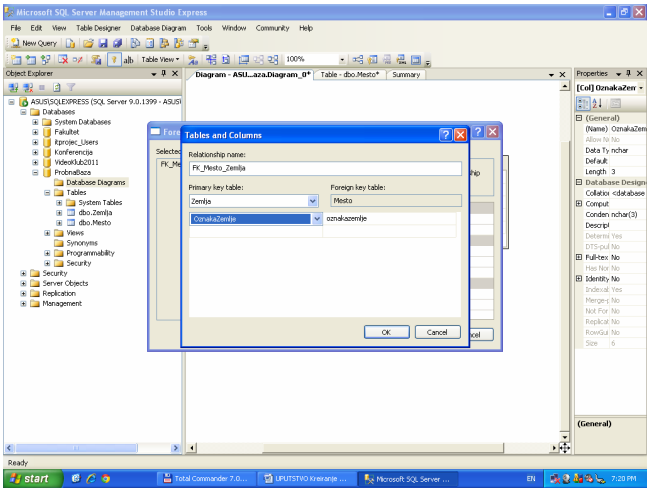
Prevlacimo Oznaku zemlje od tabele Zemlja do tabele Mesto:



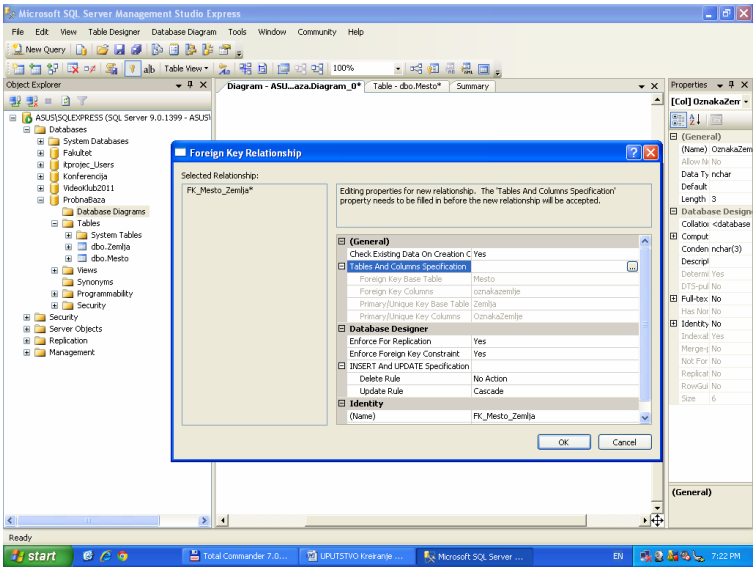
Otvara se dijalog prozor za podešavanje osobina relacije.



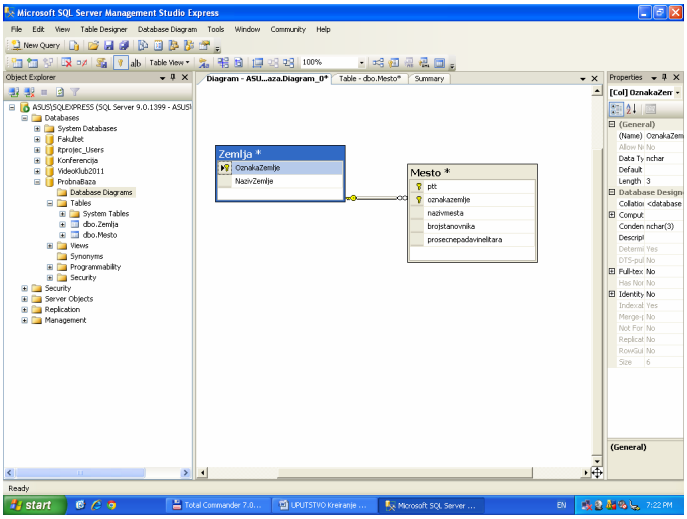
Podešavamo:



Podesavamo osobine relacije (referencijalni integritet):



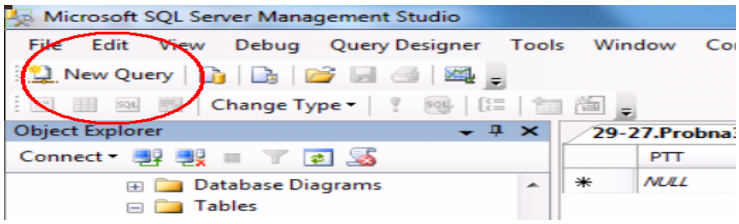
Dobijamo:



7.2.2.2. Kreiranje elemenata baze podataka SQL naredbama

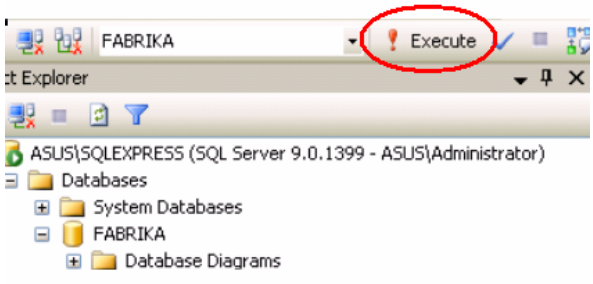
U ovom odeljku biće prikazano kreiranje baze podataka i osnovnih komponenti baze podataka: kreiranje baze podataka, tabela, primarni kljuc, alternativni (semanticki) kljuc, strani kljuc (ostvarivanje relacije), dodavanje ogranicenja na vrednosti polja, dodavanje indeksa, dodavanje trigera, pogleda, procedura.

Navedene komponente biće kreirane potrebom SQL naredbi. Da bismo pisali SQL naredbe koristimo opciju „New Query" sa palete alata.



SQL naredbe se mogu pisati i izvršavati:

1. Pojedinačno, unošenjem naredbe i pokretanjem izvršavanja posebnim tasterom „Execute"



2. Može se izvršavati niz SQL naradbi koje su zajedno napisane ili preuzete iz fajla (kao SQL skript). Tada iza svake naredbe mora pisati ključna reč GO.

KREIRANJE, BRISANJE I AKTIVIRANJE BAZE PODATAKA

Podešavamo u combo boxu da je naziv aktivne baze podataka „master".

Unosimo i izvršavamo SQL naredbu za kreiranje nove baze podataka*:* ***CREATE DATABASE*** *nazivbaze*

primer:

```
CREATE DATABASE fabrika
```

Ako bismo želeli da obrišemo bazu podataka „fabrika”, opet mora biti master baza aktivna da bismo izvrsili:
DROP DATABASE nazivbaze

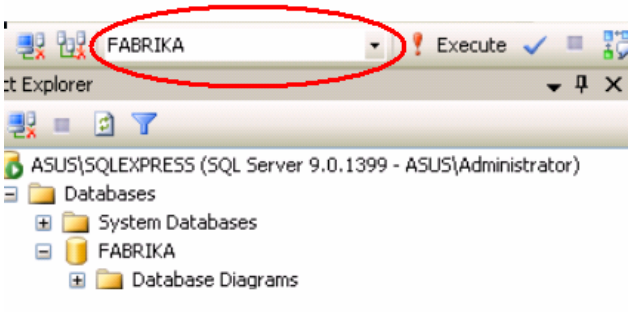
primer:

```
DROP DATABASE fabrika
```

Da bismo postavili da je AKTIVNA NOVA BAZA PODATAKA, Unosimo i izvršavamo SQL naredbu

USE nazivbaze

(ili u combo boxu promenimo da je aktivna baza: „fabrika”)



KREIRANJE TABELA

CREATE TABLE
(
naziv_polja tip podatka null/not null
)

primer:

```
CREATE TABLE proizvod(
sifra_proizvoda nvarchar(10) not null,
naziv_proizvoda nvarchar(100) not null,
sifra_jedinice_mere int null,
kolicina_po_pakovanju int null,
datum_pocetka_proizvodnje datetime null
)
```

RAD SA POLJIMA

Ako želimo da PROMENIMO TIP PODATKA za neko polje, npr. za količinu po pakovanju sa integer na real:

```
ALTER TABLE proizvod
ALTER COLUMN kolicina_po_pakovanju real
```

Ako želimo da DODAMO NOVO POLJE, npr. kolicina_na_lageru integer not null

```
ALTER TABLE proizvod
ADD kolicina_na_lageru int not null
```

Ako želimo da OBRISEMO POLJE, npr. datum_pocetka_proizvodnje.

```
ALTER TABLE proizvod
DROP COLUMN datum_pocetka_proizvodnje
```

2. primer: Kreiramo tabelu Jedinica mere.

```
CREATE TABLE JedinicaMere(
sifra int not null,
naziv nvarchar(40) not null
)
```

Dodajemo novo polje oznaka u tabeli Jedinica mere.

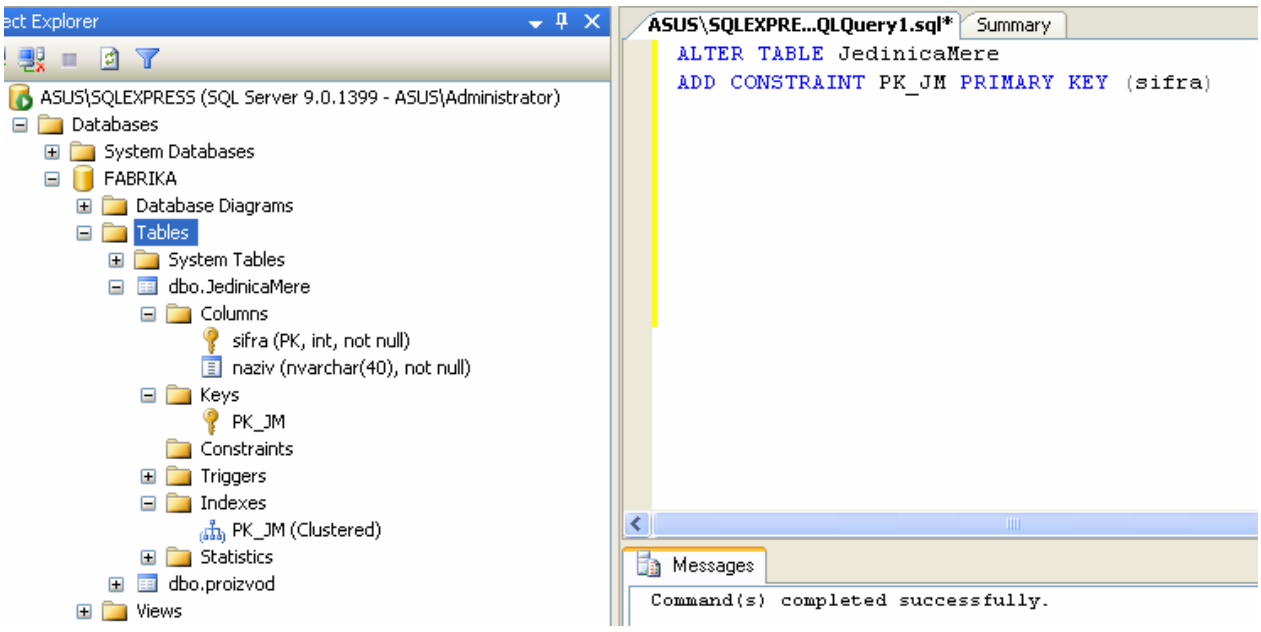
```
ALTER TABLE JedinicaMere
ADD oznaka nvarchar(10) not null
```

Napomena:
U SQL SERVERU više reči koje su odvojene razmakom navodimo u uglastim zagradama.

Kreiranje PRIMARNOG KLJUČA

```
ALTER TABLE JedinicaMere
ADD CONSTRAINT PK_JM PRIMARY KEY (sifra)
```

Nakon izvrsavanja ove naredbe imamo dodat ključ, obeleženo polje kao ključ i dodat indeks kojim se realizuje primarni ključ:



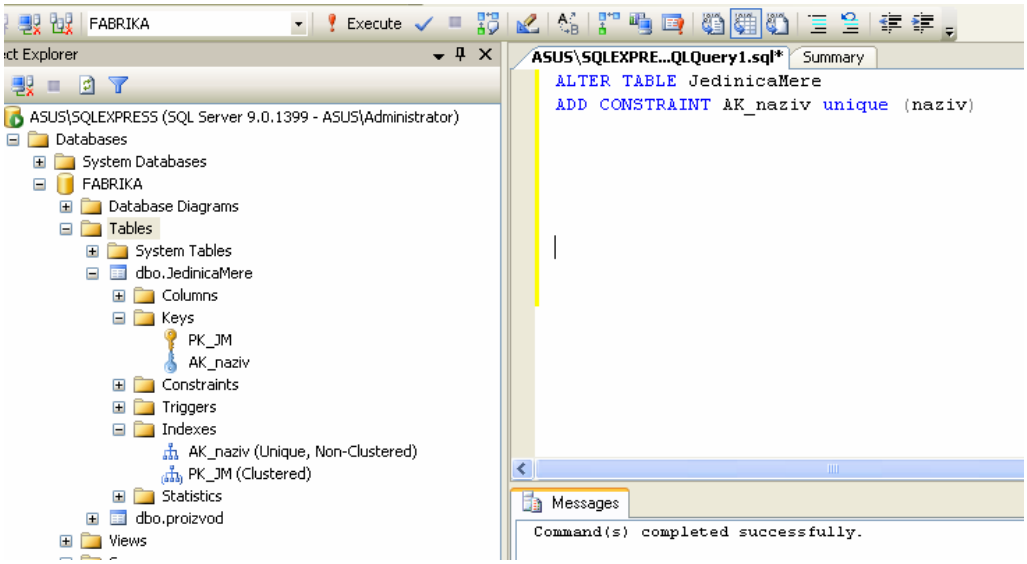
I za drugu tabelu:

```
ALTER TABLE proizvod
ADD CONSTRAINT PK_Proizvod PRIMARY KEY (sifra_proizvoda)
```

Mozemo dodati i UNIQUE INDEKS, ODNOSNO ALTERNATIVNI KLJUČ NAD NEKLJUČNIM POLJEM ili poljima koji treba da očuva jedistvenost semantike.

```
ALTER TABLE JedinicaMere
ADD CONSTRAINT AK_naziv unique (naziv)
```

nakon čega dobijamo:



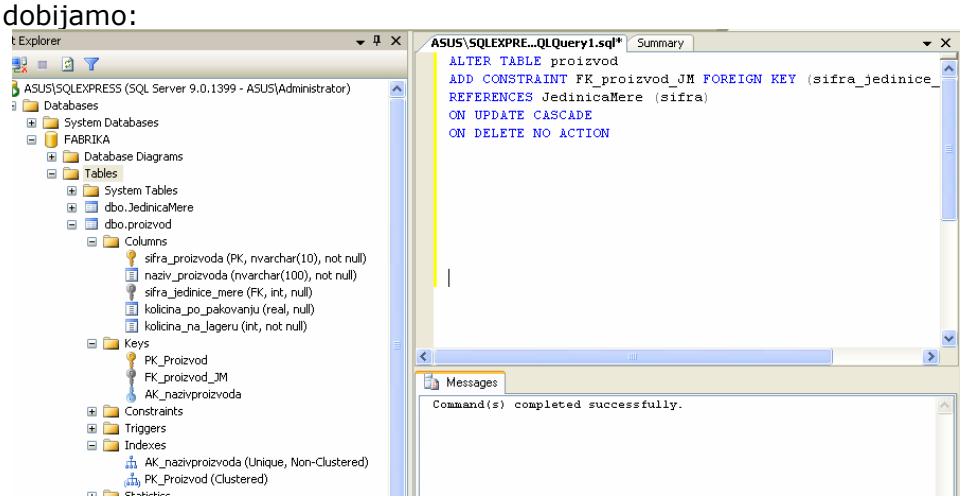
i za drugu tabelu:

```
ALTER TABLE proizvod
ADD CONSTRAINT AK_nazivproizvoda unique (naziv_proizvoda)
```

Uspostavljamo RELACIJU IZMEDJU 2 TABELE - postavljamo strani ključ u tabeli proizvod koji zavisi od tabele jedinica mere (čije je to polje primarni ključ):

```
ALTER TABLE proizvod
ADD CONSTRAINT FK_proizvod_JM FOREIGN KEY (sifra_jedinice_mere)
REFERENCES JedinicaMere (sifra)
ON UPDATE CASCADE
ON DELETE NO ACTION
```

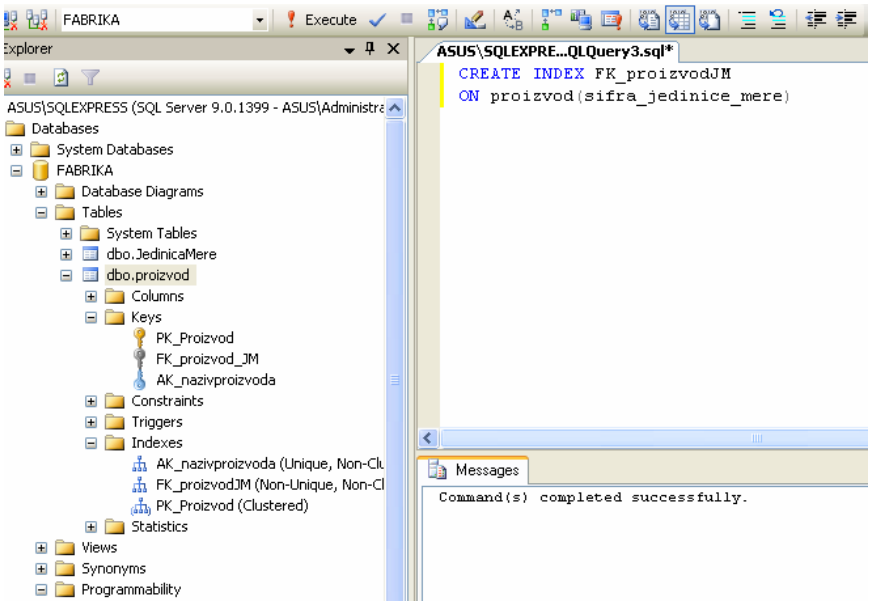
Nakon cega dobijamo:



Vidimo da ne postoji definisan indeks nad poljem sifra jedinice mere. Možemo ga dodati:

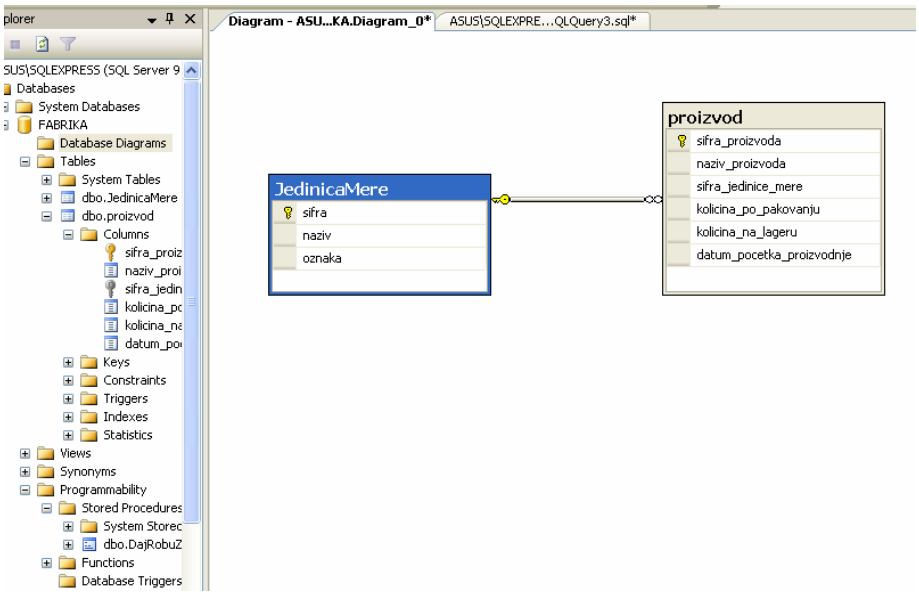
```
CREATE INDEX FK_proizvodJM
ON proizvod(sifra_jedinice_mere)
```

Nakon čega dobijamo:



DIJAGRAM

Dodajemo dijagram i sve tabele, ali posto su ranije korišćenjem SQL naredbi pravilno podešeni primarni i strani ključ, automatski dobijamo i vizualni prikaz relacija.



OČUVANJE PRAVILA SEMANTIČKOG INTEGRITETA PODATAKA – OGRANIČENJA I TRIGERI

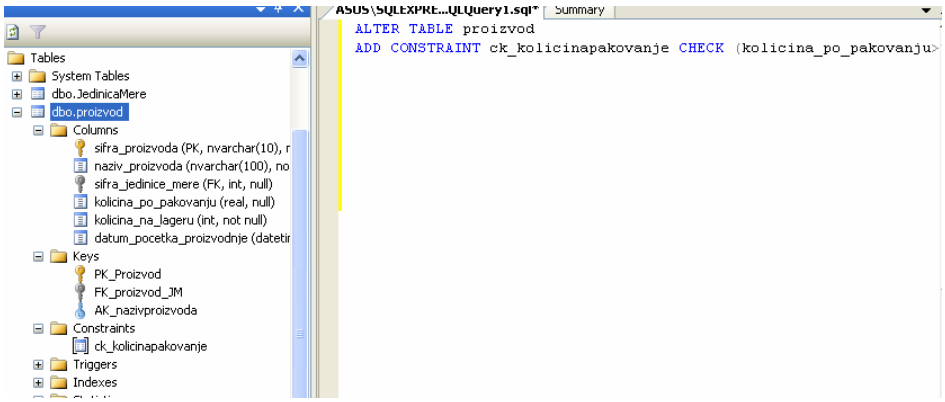
Dodajemo i OGRANICENJA na vrednosti polja.

NAPOMENA: Pravila na koja se odnose ova ograničenja koja se ugrađuju u bazu podataka su jednostavna i nezavisna od promene poslovnih pravila. Zavisna pravila provere ispravnosti podataka ne bismo trebali čuvati u bazi podataka.

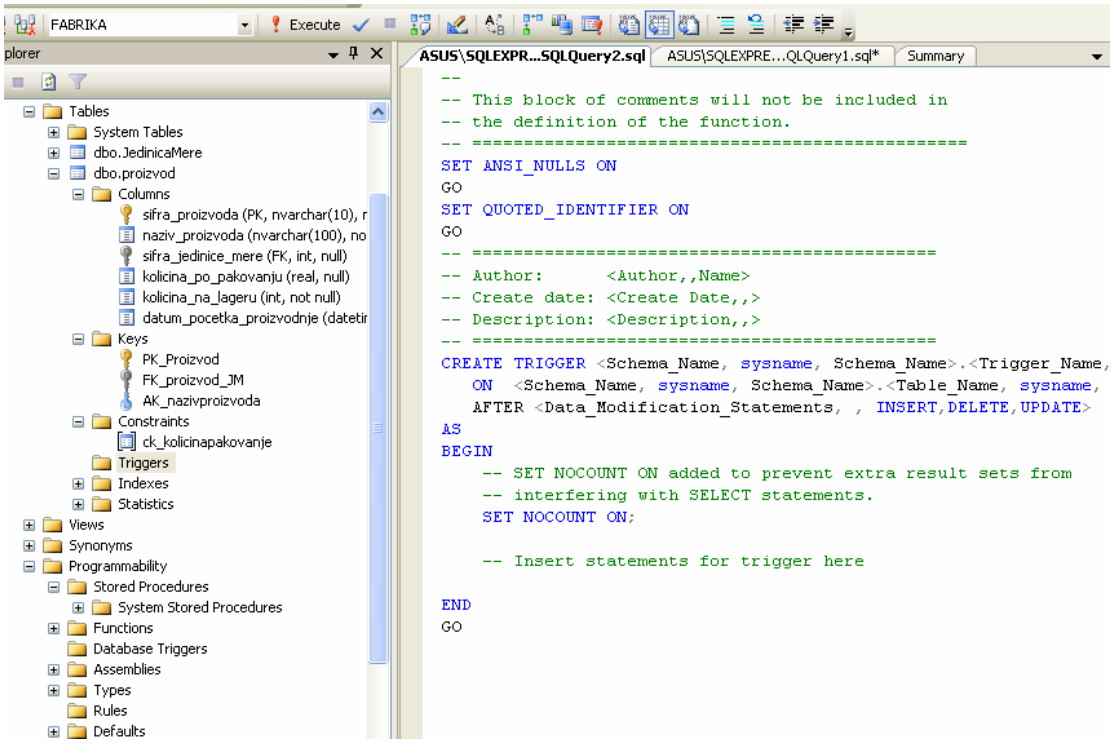
primer: Dodajemo ograničenje koje sprečava da se unese količina po pakovanju =0, već uvek mora biti>0.

```
ALTER TABLE proizvod
ADD CONSTRAINT ck_kolicinapakovanje CHECK (kolicina_po_pakovanju>0)
```

Nakon cega dobijamo:



Dodajemo TRIGER - u object exploreru desni taster add trigger i dobijamo okvirnu strukturu trigera:



primer: Kreiramo triger koji zabranjuje unos proizvoda ako je kolicina na lageru >0 i datum proizvodnje nije < danasnjeg datuma.

```
CREATE TRIGGER ProveraLageraIDatumaProizvodnje
  ON  proizvod
  for INSERT
AS
if exists (select *
```

```
                    from inserted
                    where kolicina_na_lageru>0 and not datum_pocetka_proizvodnje<getdate())

BEGIN
        -- SET NOCOUNT ON added to prevent extra result sets from
        -- interfering with SELECT statements.
        SET NOCOUNT ON;

        rollback transaction
        print 'Proizvod nije evidentiran. Ako je kolicina na lageru >0, datum pocetka proizvodnje mora prethoditi
danasnjem datumu!'
END
```

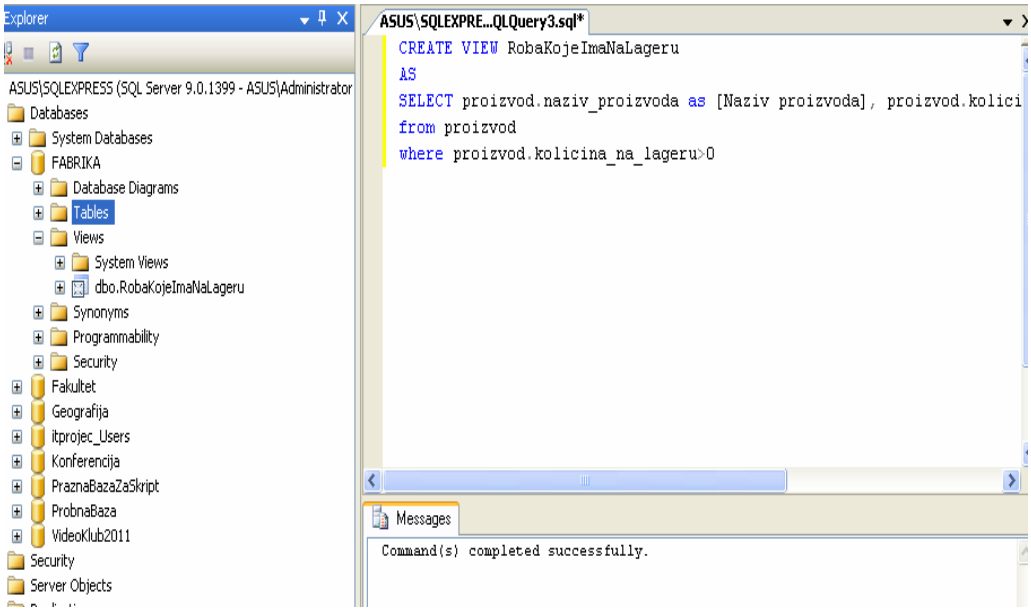
POGLEDI I PROCEDURE

Kreiramo POGLED.

primer: Izdvojiti robu koje ima na lageru.

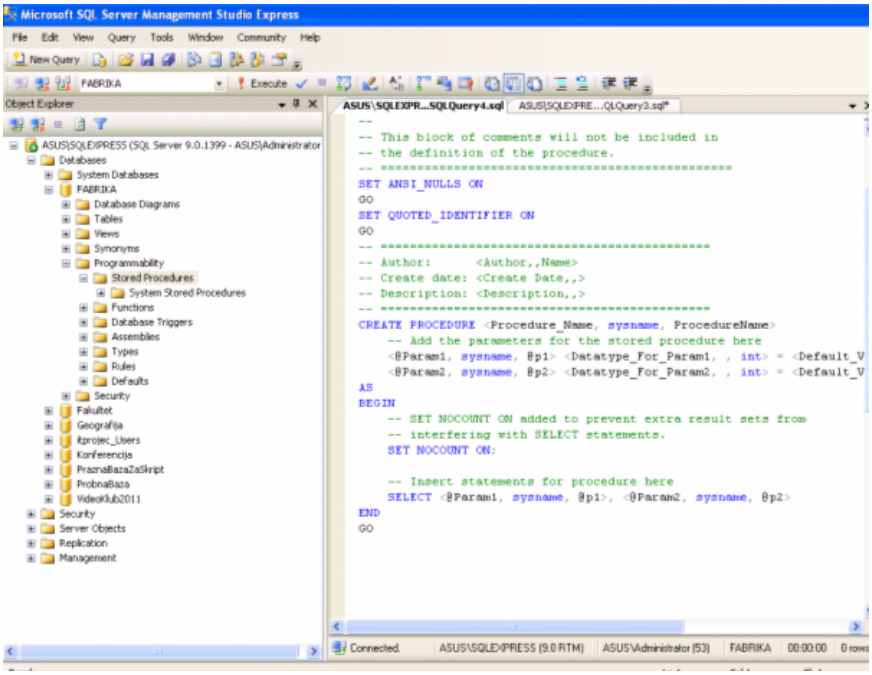
```
CREATE VIEW RobaKojeImaNaLageru
AS
SELECT proizvod.naziv_proizvoda as [Naziv proizvoda], proizvod.kolicina_na_lageru as [Kolicina]
from proizvod
where proizvod.kolicina_na_lageru>0
```

Nakon cega dobijamo:



Kreiramo PROCEDURU.

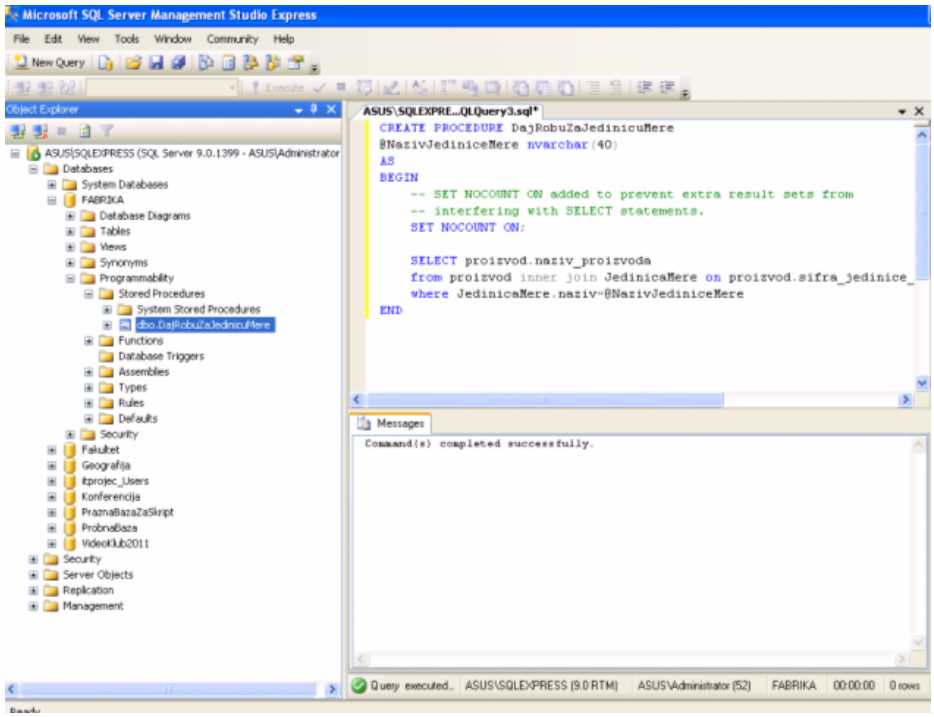
Biramo sa object explorera Programmability, Stored Procedures, New... i dobijamo okvir za pisanje procedure:



Pisemo:

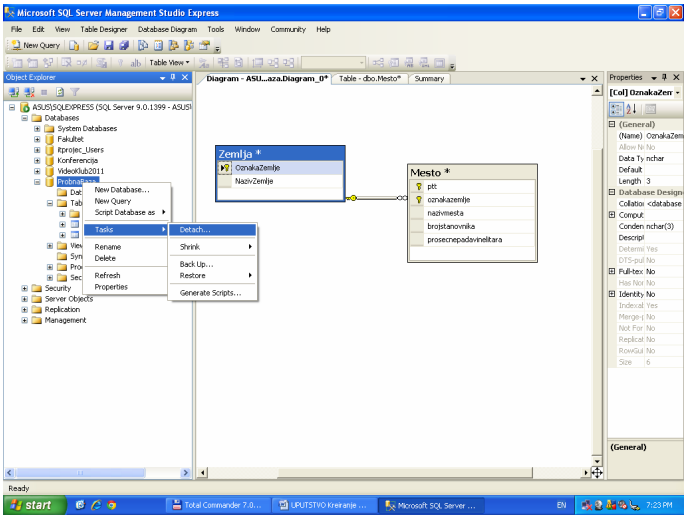
```
CREATE PROCEDURE DajRobuZaJedinicuMere
@NazivJediniceMere nvarchar(40)
AS
BEGIN
	-- SET NOCOUNT ON added to prevent extra result sets from
	-- interfering with SELECT statements.
	SET NOCOUNT ON;

   SELECT proizvod.naziv_proizvoda
   from proizvod inner join JedinicaMere on proizvod.sifra_jedinice_mere=JedinicaMere.sifra
   where JedinicaMere.naziv=@NazivJediniceMere
END
```

Nakon cega dobijamo:

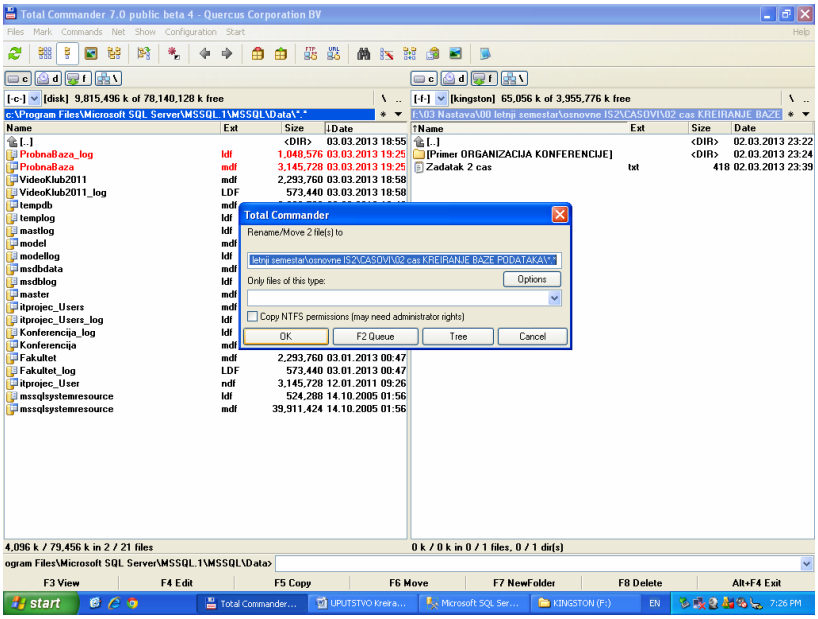


7.2.3. Izdvajanje fajla baze podataka i korišćenje sa druge lokacije

Dobijenu bazu podataka mozemo da "raskačimo" od konekcije ka DBMS-u:

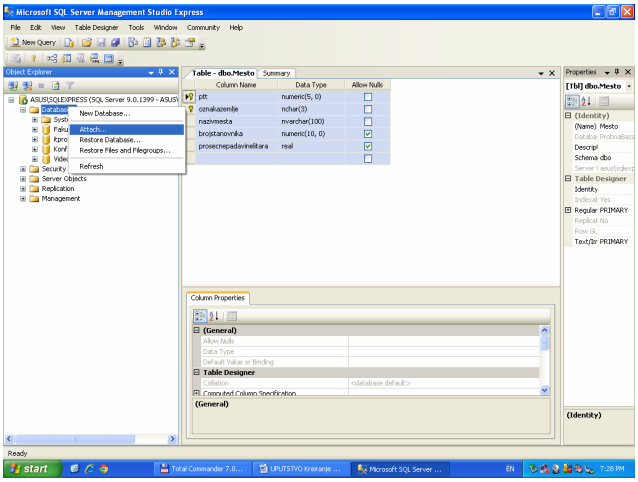


Da bismo mogli da je prebacimo na drugu lokaciju.

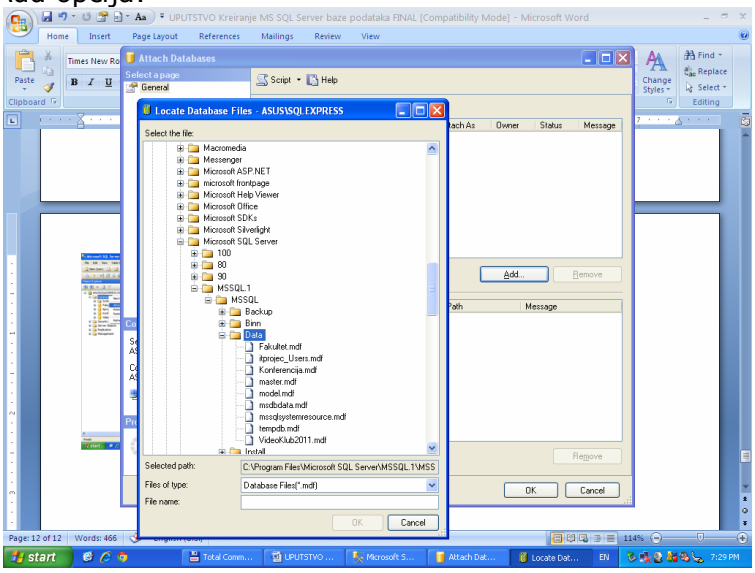


A sada možemo da koristimo bazu podataka sa druge lokacije.

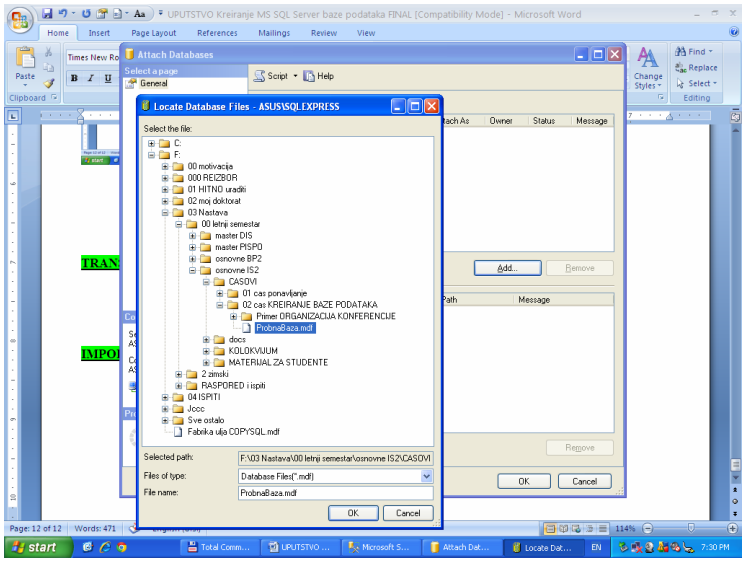
Da bismo ponovo mogli da koristimo bazu podataka sa druge lokacije, moramo da prijavimo ("zakačimo" – Attach) bazu podataka:



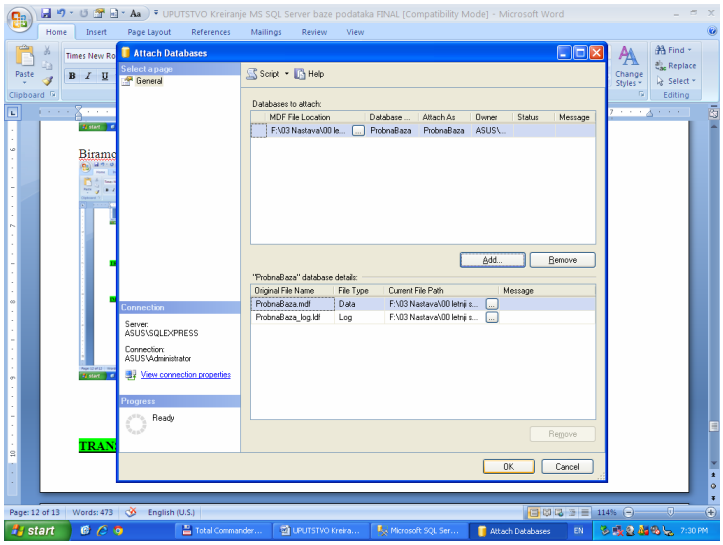
Tražimo fajl baze – Add opcija:



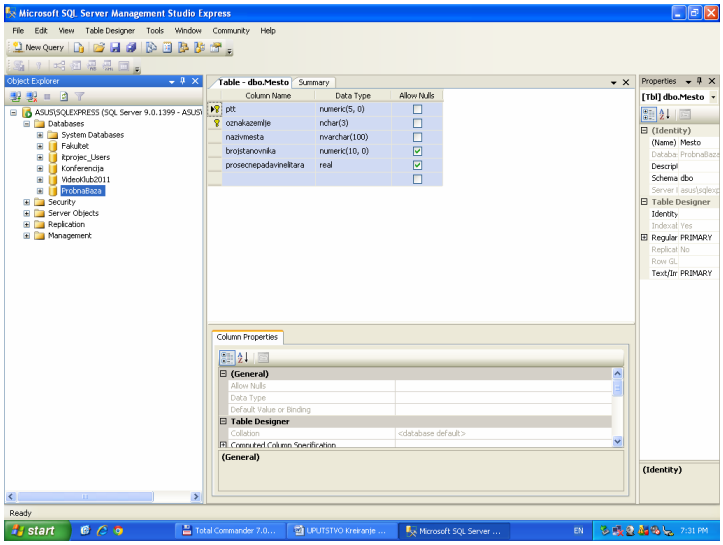
Biramo fajl:



Biramo opciju OK:



Da bismo dobili:



7.2.4. Rad sa podacima

7.2.4.1. Ažuriranje podataka

UNOS PODATAKA

primer:

```
INSERT INTO JedinicaMere (sifra, naziv, oznaka) VALUES (1, 'kilogram', 'kg')
```

Bez navodjenja polja SQL upit funkcioniše (*NAPOMENA: posebno značajno kod složenih tabela od više polja*), samo moramo voditi računa da redosled vrednosti odgovara redosledu polja u dizajnu tabele:

```
INSERT INTO proizvod
VALUES ('P1', 'pasteta', 1, 0.3, 5, '1/3/2013')
```

Napomena: Posebno je važno povesti računa o formatiranju vrednosti - numeričke vrednosti se navode bez apostrofa, realan broj sa tačkom, string i datum sa apostrofima.Datum se navodi u američkom stilu sa kosim crtama.

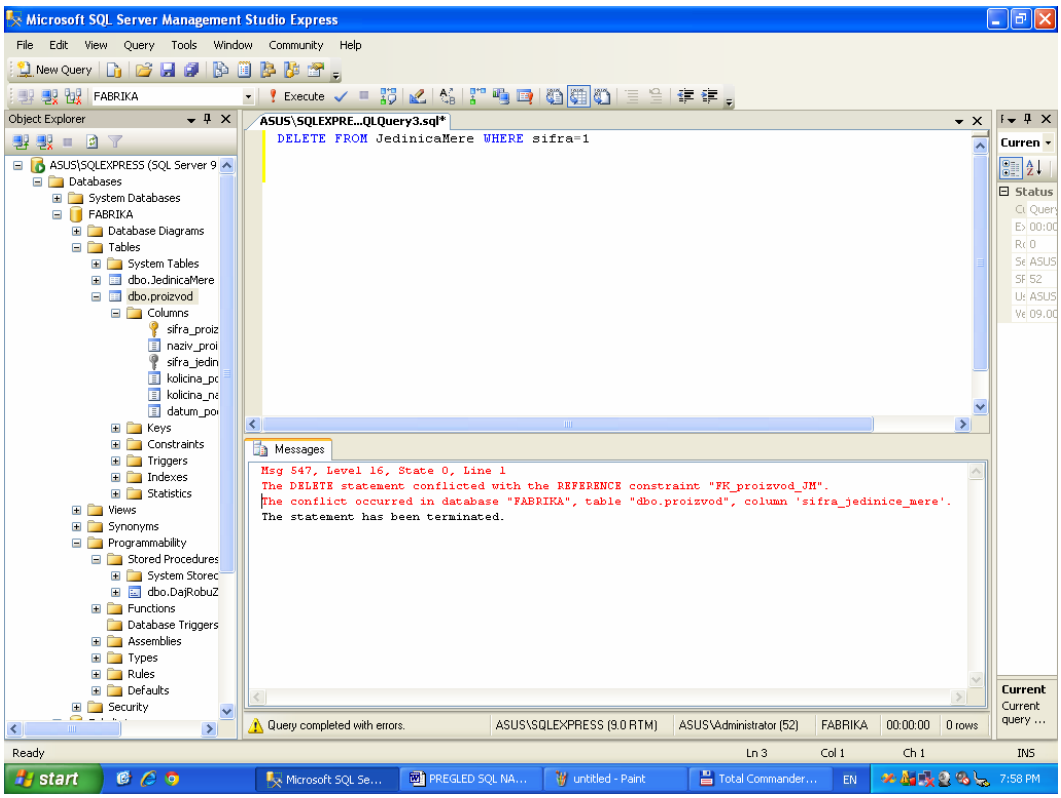
IZMENA PODATAKA

```
UPDATE proizvod
SET kolicina_po_pakovanju=0.2 WHERE sifra_proizvoda='P1'
```

BRISANJE PODATAKA

```
DELETE FROM JedinicaMere WHERE sifra=1
```

primer: Pokusacemo obrisati zapis iz tabele Jedinica mere, za koji postoje vezani podaci u tabeli proizvod i testirati pravila očuvanja referencijalnog integriteta. Dobijamo poruku o grešci.



Očigledno, nije moguće obrisati zapis u tabeli na strani 1, a da se pri tome ne naruši referenijalni integritet. Mehanizam ocuvanja referencijalnog integriteta je definisano kao NO ACTION sto znaci da je nemoguće obrisati zapis na strani 1 ako postoji bar jedan vezani zapis na strani više.

7.2.4.2. Izdvajanje i pretraga podataka

Može se izvršavati korišćenjem:

- Select naredbe
- Korišćenjem pogleda koji sadrži select naredbu
- Pozivom procedure koja sadrzi select naredbu

Očigledno, suština izdvajanja i pretrage podataka je u korišćenju SQL naredbi tipa SELECT.

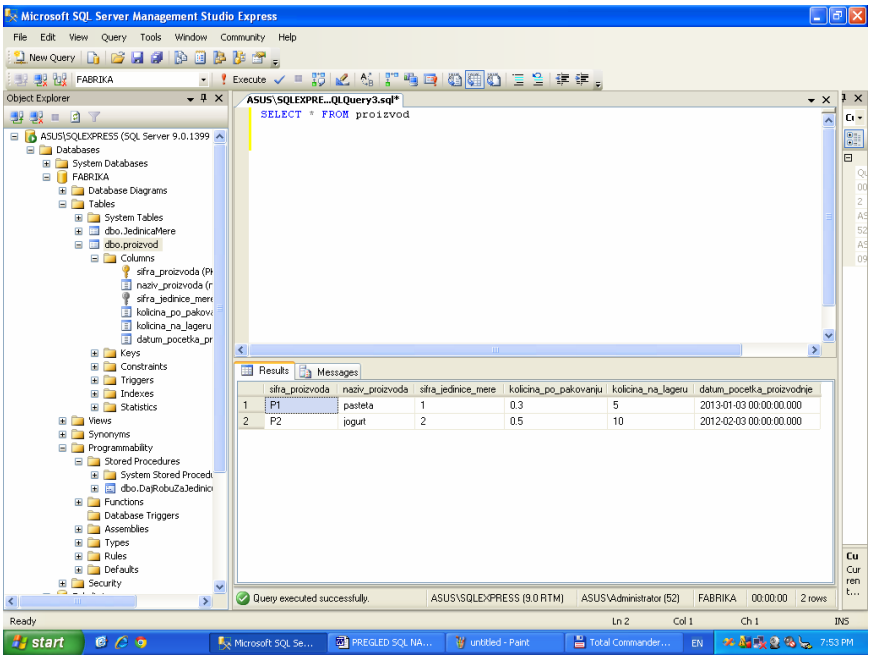
OPŠTI OBLIK SELECT NAREDBE

```
SELECT * ili tabela.POLJE, AGREGATNA FUNKCIJA (tabela.POLJE)
FROM TABELA (inner/left/right) JOIN TABELA
WHERE kriterijum filtera podataka (polje znak vrednost and/or polje znak vrednost)
GROUP BY polja koja prethode agregatnoj funkciji
HAVING agregatna funkcija znak vrednost
ORDER BY polje asc/desc, polje asc/desc
```

Koriscenje SELECT NAREDBE

primer:
SELECT * FROM PROIZVOD

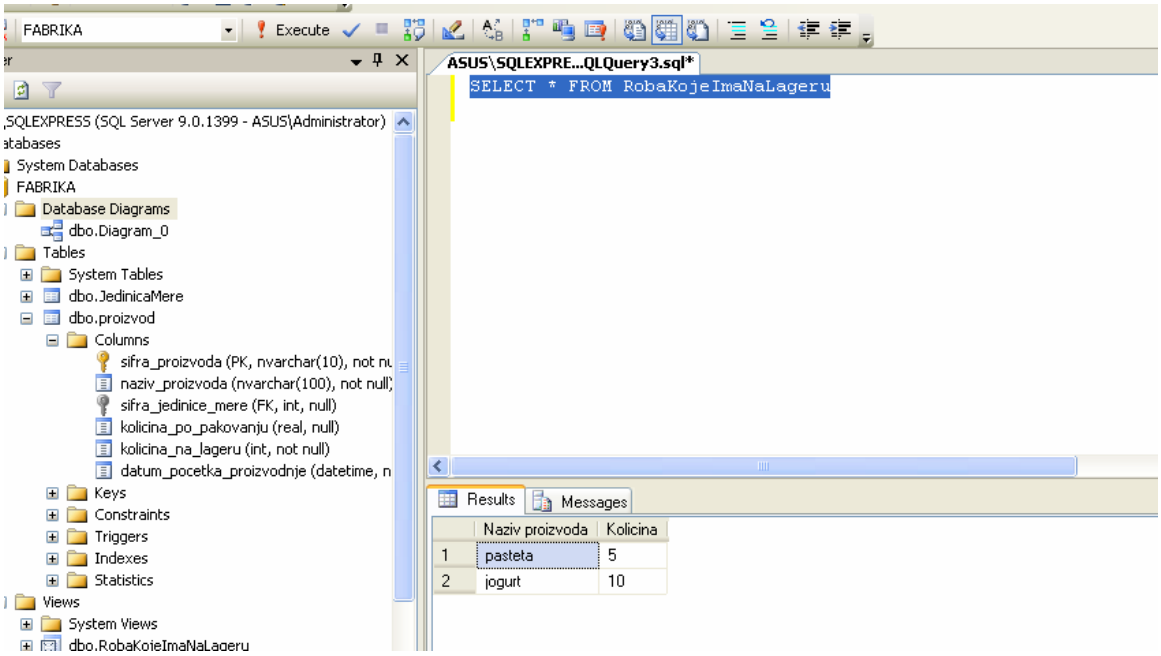
nakon čega dobijamo:



Korišćenje POGLEDA kao naziva tabele.

SQL SERVER:
SELECT * FROM RobaKojeImaNaLageru

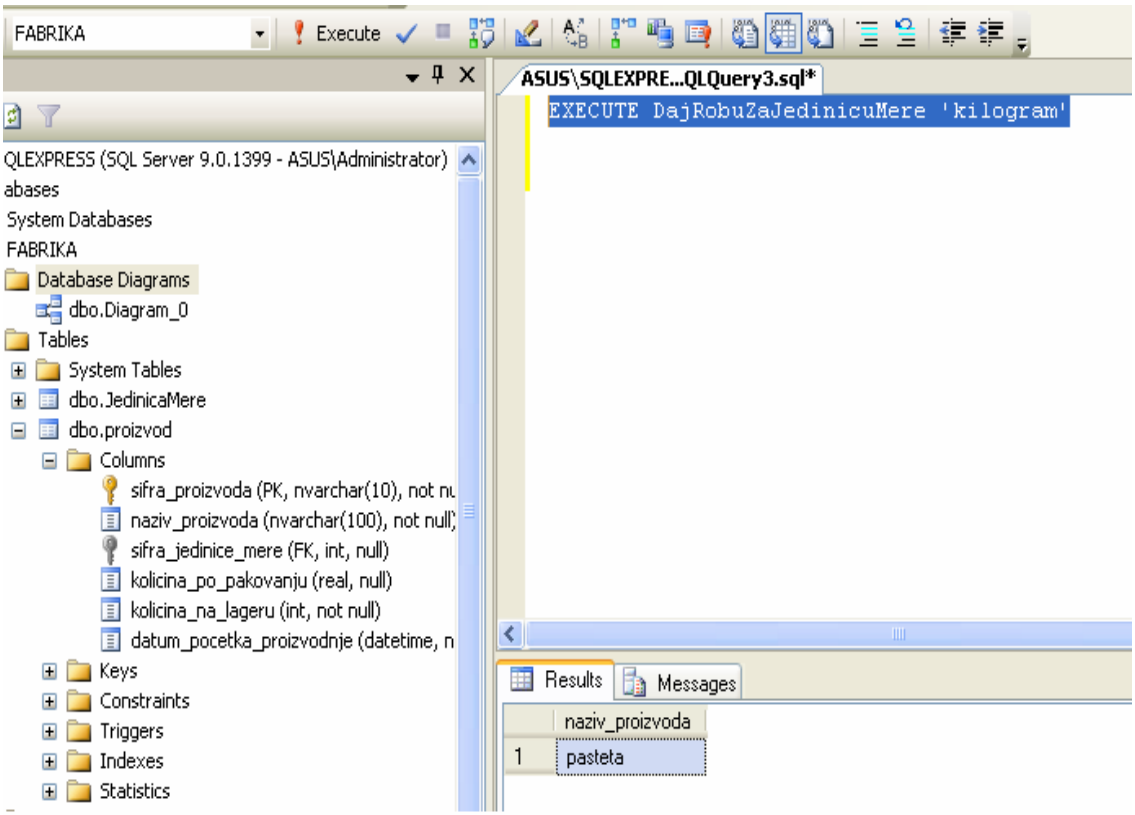
Nakon čega dobijamo:



KORIŠĆENJE PROCEDURE

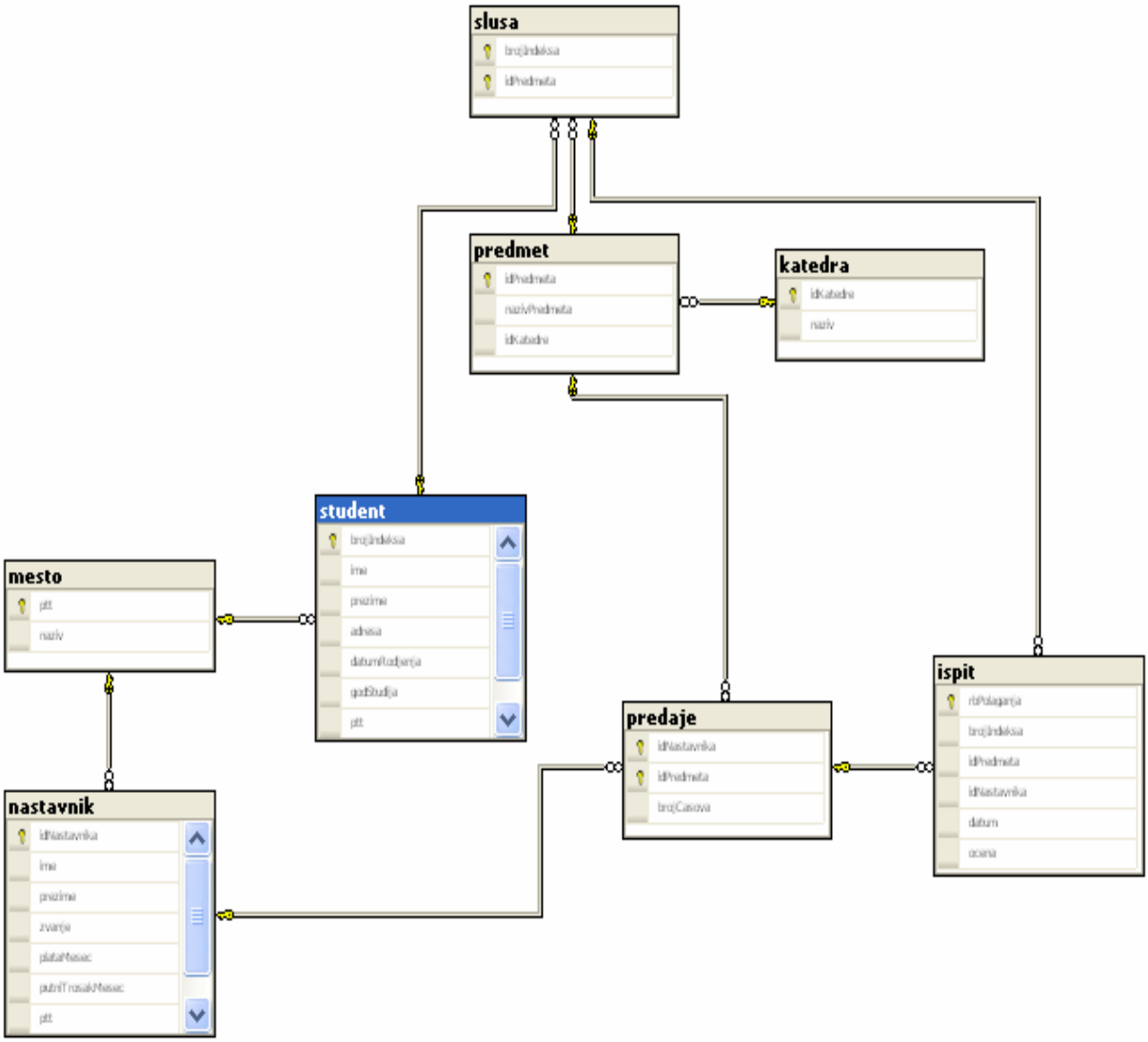
```
EXECUTE DajRobuZaJedinicuMere 'kilogram'
```

Nakon čega dobijamo:



7.2.4.3. PRIMERI SELECT UPITA

U nastavku je dat primer baze podataka "Fakultet" i primeri korišćenja SQL upita tipa SELECT.



RAD SAPOLJIMA, NAZIVIMA TABELA, SPOJ TABELA, WHERE FILTER, ORDER BY

```
select nastavnik.prezime, nastavnik.ime, mesto.naziv, predmet.nazivPredmeta
from ((nastavnik inner join mesto on nastavnik.ptt = mesto.ptt) inner join predaje on
nastavnik.idNastavnika =predaje.idNastavnika) inner join predmet on predmet.idPredmeta =
predaje.idPredmeta
where mesto.naziv='Zrenjanin'
order by predmet.nazivPredmeta desc
```

KORIŠĆENJE AGREGATNIH FUNKCIJA, GROUP BY I HAVING

```
select mesto.naziv, COUNT(Nastavnik.idNastavnika)
from nastavnik inner join mesto on nastavnik.ptt = mesto.ptt
group by mesto.naziv
having COUNT(Nastavnik.idNastavnika) >1
order by mesto.naziv asc
```

RAZLIKA WHERE I HAVING

```
SELECT     student.brojIndeksa, student.ime, student.prezime, AVG(ispit.ocena) AS prosek,
COUNT(ispit.idPredmeta) AS [broj polozenih ispita]
FROM         ispit INNER JOIN
                      student ON ispit.brojIndeksa = student.brojIndeksa
WHERE     (ispit.ocena > 5)
GROUP BY student.brojIndeksa, student.ime, student.prezime
HAVING      (AVG(ispit.ocena) > 9)
ORDER BY prosek DESC
```

KORIŠĆENJE ZAMENE ZA NAZIVE TABELA I POLJA, GROUP BY I AGREGATNA FUNKCIJA

```
select n.zvanje, COUNT(idNastavnika) as [Ukupan broj nastavnika]
from nastavnik n
group by n.zvanje
```

KORIŠĆENJE DRUGIH VRSTA SPOJEVA

```
SELECT NAZIVPREDMETA, COUNT(ISPIT.idPredmeta) AS BRSTUDENATA
FROM PREDMET LEFT JOIN ISPIT
ON ISPIT.idPredmeta = PREDMET.idPredmeta
GROUP BY NAZIVPREDMETA
```

KORIŠĆENJE *

```
select * from nastavnik

select nastavnik.*, mesto.naziv from
nastavnik inner join mesto on nastavnik.ptt=mesto.ptt
```

KORIŠĆENJE % i _

% menja vise (neodredjen broj) simbola

```
select nastavnik.*, mesto.naziv from
nastavnik inner join mesto on nastavnik.ptt=mesto.ptt
where mesto.naziv like '%ni%'
```

_ menja jedan simbol

```
select nastavnik.*, mesto.naziv from
nastavnik inner join mesto on nastavnik.ptt=mesto.ptt
where mesto.naziv like '_ovi%'
```

KORIŠĆENJE BETWEEN

```
select nastavnik.*, mesto.naziv from
nastavnik inner join mesto on nastavnik.ptt=mesto.ptt
WHERE plataMesec between 15000 and 25000
```

KORIŠĆENJE TOP(N)

```
select TOP (3) nastavnik.*, mesto.naziv from
nastavnik inner join mesto on nastavnik.ptt=mesto.ptt
order by nastavnik.plataMesec desc
```

KORIŠĆENJE NULL I ISNULL

```
select nastavnik.*, mesto.naziv from
nastavnik inner join mesto on nastavnik.ptt=mesto.ptt
WHERE putniTrosakMesec is null


select nastavnik.*, mesto.naziv from
nastavnik inner join mesto on nastavnik.ptt=mesto.ptt
WHERE putniTrosakMesec is not null

select nastavnik.prezime, nastavnik.ime, plataMesec, putniTrosakMesec, plataMesec +
ISNULL(putniTrosakMesec,0) as [mesecni prihod]
from nastavnik
```

KORIŠĆENJE CASE

```
select
nastavnik.prezime, nastavnik.ime, plataMesec, putniTrosakMesec,
CASE WHEN putniTrosakMesec is null then 0 else putniTrosakMesec END as [TRANSFORMISANI
putni trosak]
from nastavnik

select
nastavnik.prezime, nastavnik.ime, plataMesec, putniTrosakMesec,
plataMesec + CASE WHEN putniTrosakMesec is null then 0 else putniTrosakMesec END as
[TRANSFORMISANI mesecni prihod]
from nastavnik
```

LOGICKI IZRAZI I OPERATORI (AND, OR,<, >=, !=)

```
SELECT STUDENT.BROJINDEKSA, STUDENT.PREZIME, STUDENT.IME
FROM STUDENT
WHERE STUDENT.IME = 'Goran' or STUDENT.IME ='Jovan'

SELECT STUDENT.BROJINDEKSA, STUDENT.PREZIME, STUDENT.IME
FROM STUDENT
WHERE STUDENT.IME = 'Goran' and_STUDENT.PREZIME ='Cuk'

SELECT NASTAVNIK.*
FROM NASTAVNIK
WHERE PLATAMESEC > 20000

SELECT NASTAVNIK.*
FROM NASTAVNIK
WHERE PLATAMESEC<=19000

SELECT NASTAVNIK.*
FROM NASTAVNIK
WHERE ZVANJE !='docent'
```

KORIŠĆENJE UNION

```
SELECT STUDENT.BROJINDEKSA, STUDENT.PREZIME, IME,
                NAZIVPREDMETA
FROM STUDENT, SLUSA, ISPIT, PREDMET
WHERE PREDMET.IDPREDMETA=SLUSA.IDPREDMETA
```

```
    AND SLUSA.BROJINDEKSA=STUDENT.BROJINDEKSA
    AND (SLUSA.BROJINDEKSA=ISPIT.BROJINDEKSA
    AND SLUSA.IDPREDMETA=ISPIT.IDPREDMETA)
AND ime='Predrag' and prezime ='Pajovic'
and ocena=5
UNION
SELECT STUDENT.BROJINDEKSA, STUDENT.PREZIME, IME,
                NAZIVPREDMETA
FROM STUDENT, SLUSA, PREDMET
WHERE PREDMET.IDPREDMETA=SLUSA.IDPREDMETA
    AND SLUSA.BROJINDEKSA=STUDENT.BROJINDEKSA
AND ime='Predrag' and prezime ='Pajovic'
and SLUSA.IDPREDMETA NOT IN (SELECT IDPREDMETA FROM ISPIT)
```

KORIŠĆENJE IN, not IN

```
SELECT student.brojIndeksa, student.ime, student.prezime, mesto.naziv
FROM mesto INNER JOIN student ON mesto.ptt = student.ptt
WHERE student.brojIndeksa not IN (SELECT brojIndeksa
                                                    FROM ISPIT
                                                    WHERE OCENA>5)
and (mesto.naziv='Zrenjanin' or mesto.naziv='Novi Sad')
and godstudija=1
```

PODUPITI

```
select *
from nastavnik
where platamesec > (select platamesec
                from nastavnik
                where prezime='Malbaski'
                and ime='Dusan')
```

KORIŠĆENJE KORELISANIH PODUPITA

```
select ime, prezime, zvanje, naziv, platamesec
from nastavnik n1 inner join mesto
on n1.ptt=mesto.ptt
where platamesec > (select avg(platamesec)
                                    from nastavnik n2
                                    where n2.ptt=n1.ptt)
```

KORIŠĆENJE AGREGATNIH FUNKCIJA

```
SELECT    MAX(plataMesec) as najveca,
                MIN(plataMesec) as najmanja,
                SUM(plataMesec) as suma,
                count(idnastavnika) as nastavnika,
                avg(plataMesec) as prosek
FROM       nastavnik
```

RAD SA DATUMSKIM FUNKCIJAMA

– izdvaja dan, mesec, godinu iz tekuceg datuma:

```
SELECT day(getdate()), month(getdate()), year(getdate())
```

- izdvaja mesec i transformise ga u rec na engleskom jeziku koja odgovara tom mesecu

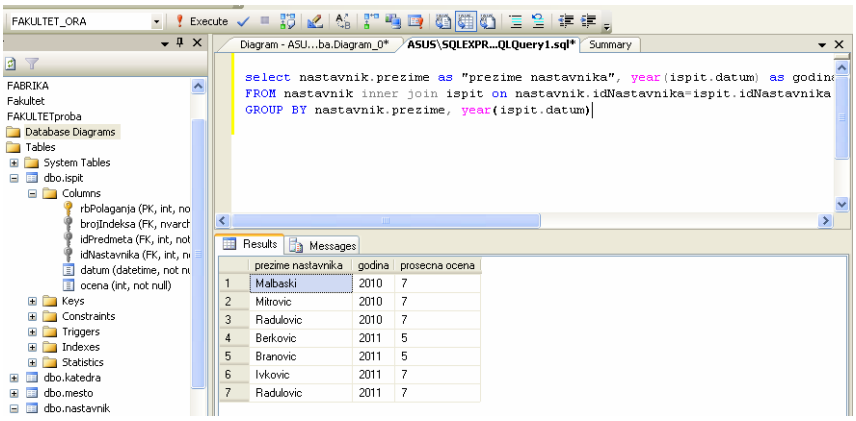
```
DATENAME(MM,DatumPocetka)
```

TYPE CAST
– transformacija tipa podatka funkcijom CAST ili CONVERT

```
select Naziv as [CONFERENCE], CAST(Day(DatumPocetka) as nvarchar(2)) + ' th ' +
CONVERT(VARCHAR(20), DATENAME(MM,DatumPocetka)) + ' ' + CAST(Year(DatumPocetka) as
nvarchar(4)) as [DATE] from Konferencija
```

RAD SA DATUMSKIM FUNKCIJAMA, AGREGATNIM FUNKCIJAMA I NUMERICKIM IZRAZIMA

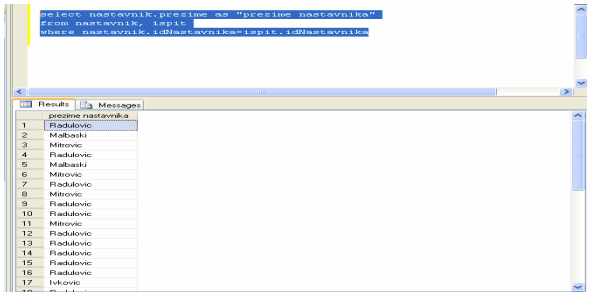
```
SELECT nastavnik.prezime as "prezime nastavnika", year(ispit.datum) as godina,
SUM(ocena)/count(ispit.rbPolaganja) as "prosecna ocena"
FROM nastavnik inner join ispit on nastavnik.idNastavnika=ispit.idNastavnika
GROUP BY nastavnik.prezime, year(ispit.datum)
```



KORIŠĆENJE DISTINCT I SPOJ TABELA KORISTEĆI WHERE

- bez distinct -ponavljanje vrednosti

```
select nastavnik.prezime as "prezime nastavnika"
from nastavnik, ispit
where nastavnik.idNastavnika=ispit.idNastavnika
```



- **sa distinct:**

```
select DISTINCT nastavnik.prezime as "prezime nastavnika"
from nastavnik, ispit
where nastavnik.idNastavnika=ispit.idNastavnika
```

PODUPITI NAD ISTOM TABELOM

```
SELECT distinct s.prezime, s.ime, s.brojIndeksa
from student s
WHERE s.godStudija =  (select MAX(st.godStudija)
from student st)
```

KORIŠĆENJE **EXISTS**

1. studenti koji su polagali ispite

```
SELECT distinct student.prezime, student.ime, student.brojIndeksa
from (student inner join slusa on student.brojIndeksa=slusa.brojIndeksa) inner join ispit on
slusa.brojIndeksa=ispit.brojIndeksa and slusa.idPredmeta=ispit.idPredmeta
```

2. studenti koji su polagali ispite kao I broj izlazenja na ispite

```
SELECT distinct student.prezime, student.ime, student.brojIndeksa, count (rbPolaganja) as
[Broj polaganja ispita]
from (student inner join slusa on student.brojIndeksa=slusa.brojIndeksa) inner join ispit on
slusa.brojIndeksa=ispit.brojIndeksa and slusa.idPredmeta=ispit.idPredmeta
GROUP BY student.prezime, student.ime, student.brojIndeksa
```

3. svi studenti sa brojem izlazenja na ispite

```
SELECT distinct student.prezime, student.ime, student.brojIndeksa, count (rbPolaganja)
from (student left join slusa on student.brojIndeksa=slusa.brojIndeksa) left join ispit on
slusa.brojIndeksa=ispit.brojIndeksa and slusa.idPredmeta=ispit.idPredmeta
GROUP BY student.prezime, student.ime, student.brojIndeksa
```

KORELISANI UPIT SA EXISTS

```
SELECT distinct student.prezime, student.ime, student.brojIndeksa
from student
WHERE EXISTS ( select * from ispit inner join slusa on
            slusa.brojIndeksa=ispit.brojIndeksa and slusa.idPredmeta=ispit.idPredmeta
            where student.brojIndeksa=slusa.brojIndeksa)
```

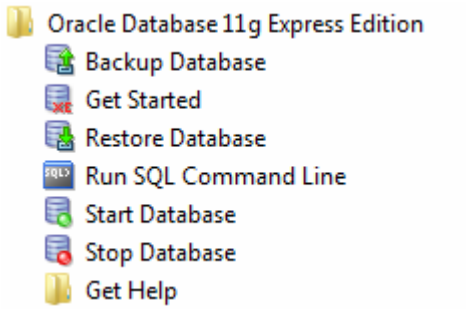
KORIŠĆENJE **ANY**

```
SELECT distinct student.prezime, student.ime, student.brojIndeksa
from student
WHERE student.brojIndeksa = ANY (select slusa.brojIndeksa from slusa inner join predmet on
slusa.idPredmeta=Predmet.idPredmeta where predmet.nazivPredmeta='Informacioni sistemi')
```

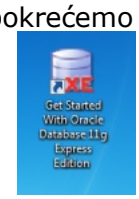
7.3. ORACLE BAZA PODATAKA

7.3.1. Radno okruženje

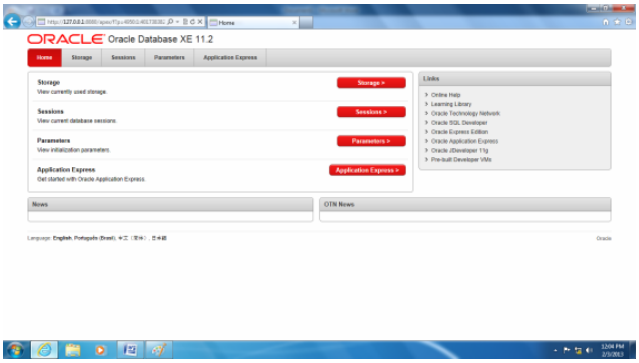
U okviru ovog materijala i praktične nastave, koristi se Oracle Database 11g Express Edition, čija je besplatna instalacija dostupna sa ORACLE web stranice.



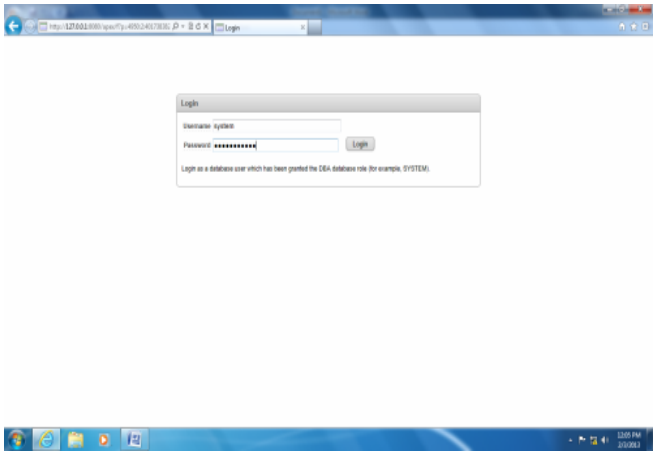
Sa radne površine (Windows desktop-a) pokrećemo ORACLE radno okruženje koristeći ikonicu:



Pokreće se web aplikacija pomoću koje ćemo kreirati bazu podataka i aplikaciju:



Kliknemo na opciju Storage i dobijamo dijalog za prijavljivanje korisnika. Koristimo:

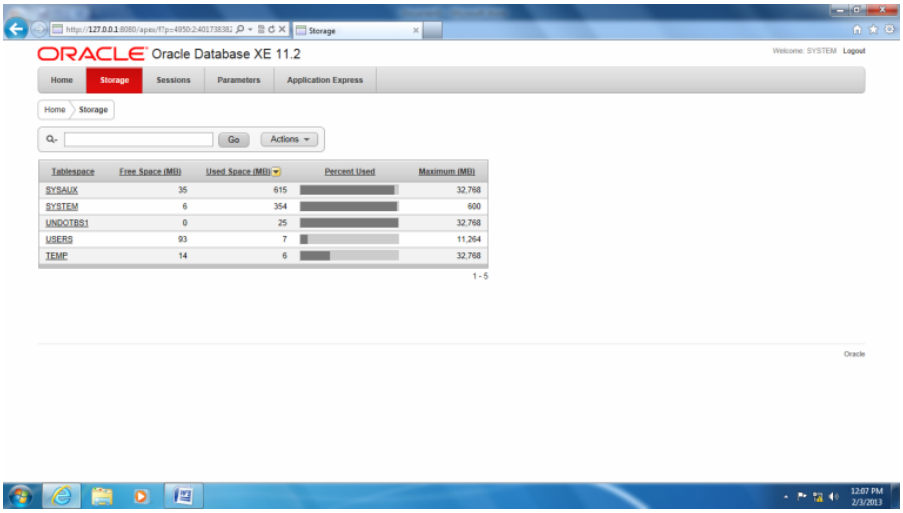


Username: system
Password: oracleadmin

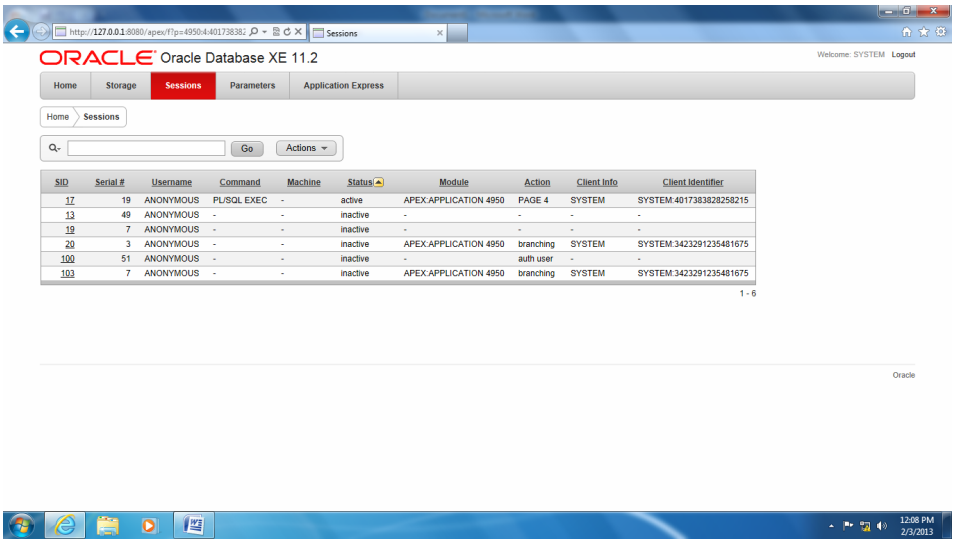
7.3.2. Kreiranje baze podataka

7.3.2.1. Kreiranje baze podataka iz vizuelnog radnog okruženja

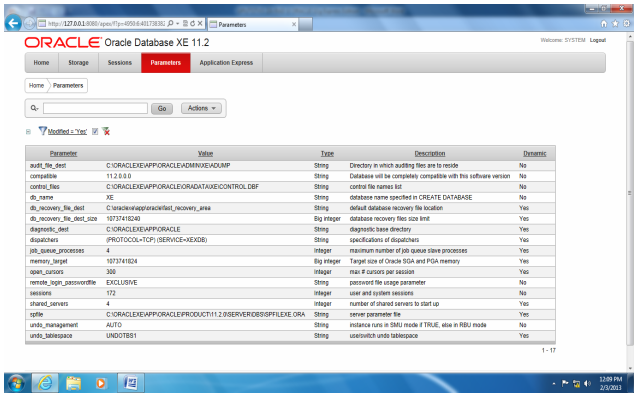
Nakon prijavljivanja možemo pokrenuti bilo koju od opcija: Storage, Sessions, Parameters, Application Express. Opcija Storage prikazuje trenutno aktivne baze podataka i s obzirom da radimo sa Express verzijom, ne možemo dodavati novu bazu podataka.



Opcija Sessions prikazuje trenutno prijavljene sesije nad bazom podataka i korisnike:

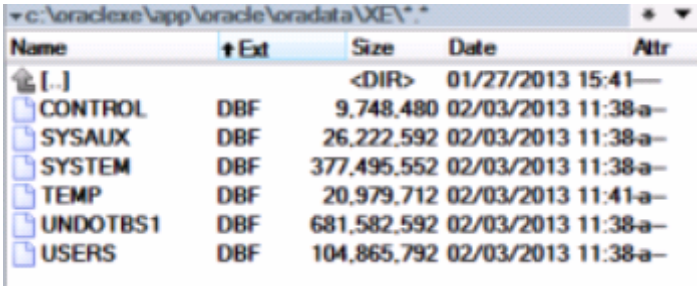


Opcija Parameters nam daje opis osnovnih foldera i parametre za rad sistema:



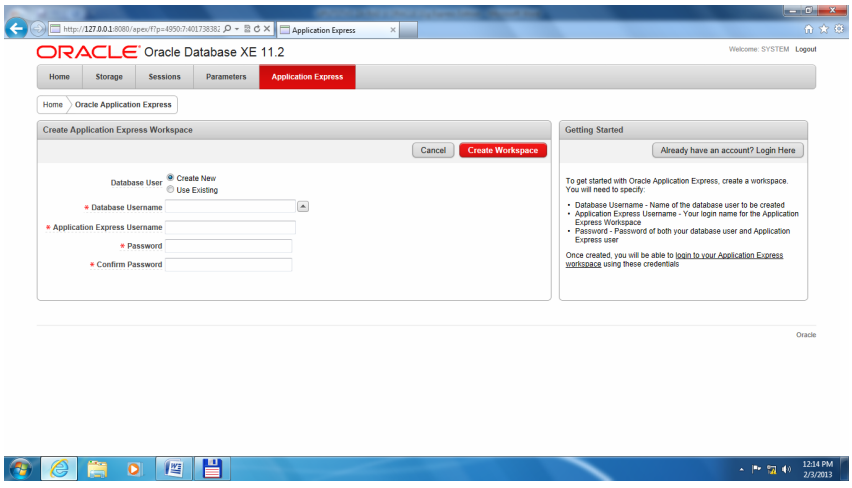
Očigledno, centralna baza podataka se zove XE i podaci koji čine bazu XE se nalaze na putanji:

C:\oraclexe\app\oracle\oradata\XE

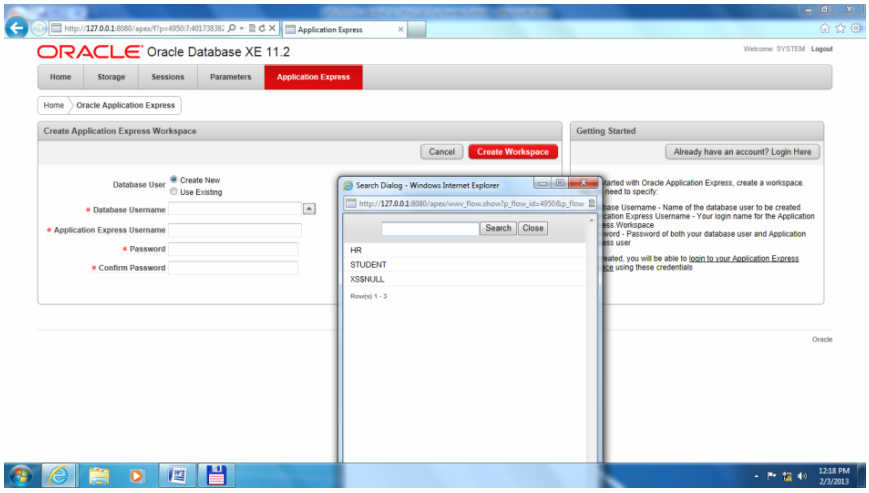


RAD U APPLICATION EXPRESS

Pokrecemo opciju Application Express.

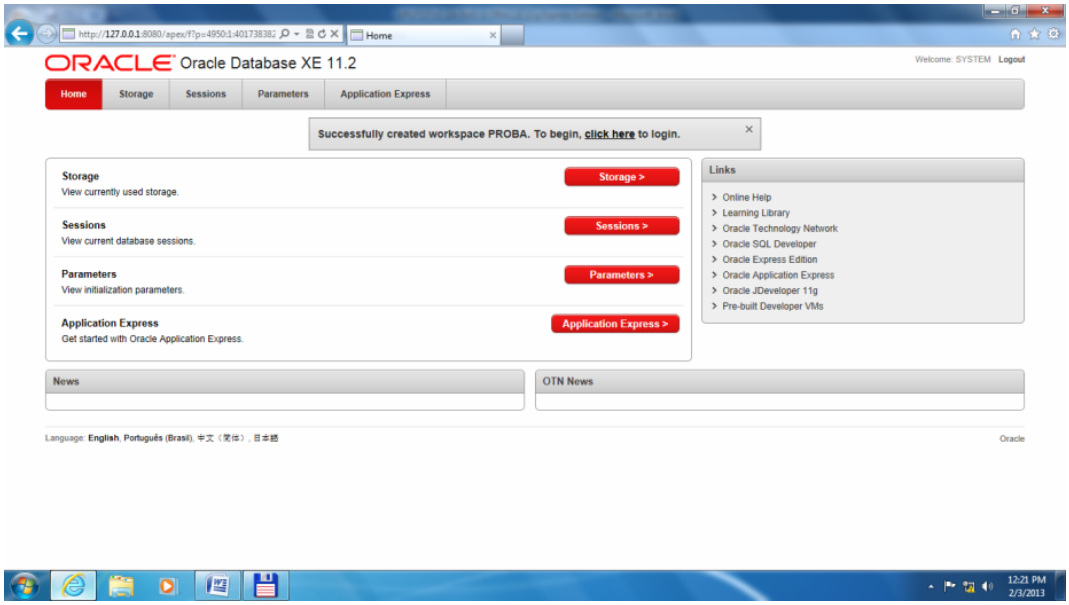


Kreiramo novi Workspace. U okviru toga, moramo kreirati novog korisnika baze podataka, kao i korisnika Application express. Lista već ranije definisanih korisnika se može videti koristeći taster sa trouglom pored text boxa za database username.

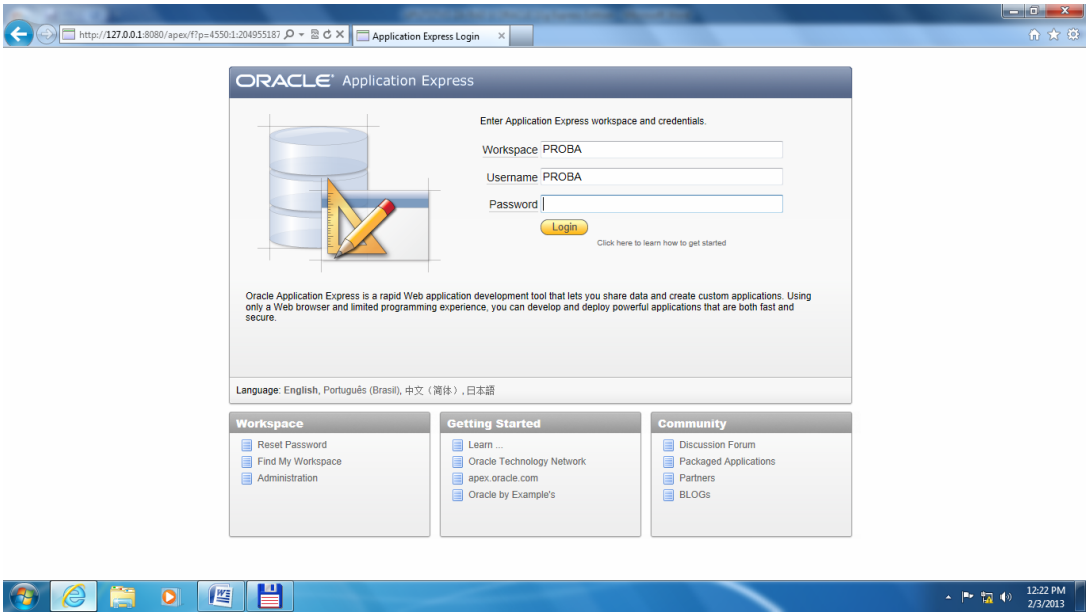


Unosimo podatke o korisniku: proba, proba, sifra, sifra i biramo opciju Create Workspace. (ili test,test, proba, proba)

Dobijamo:

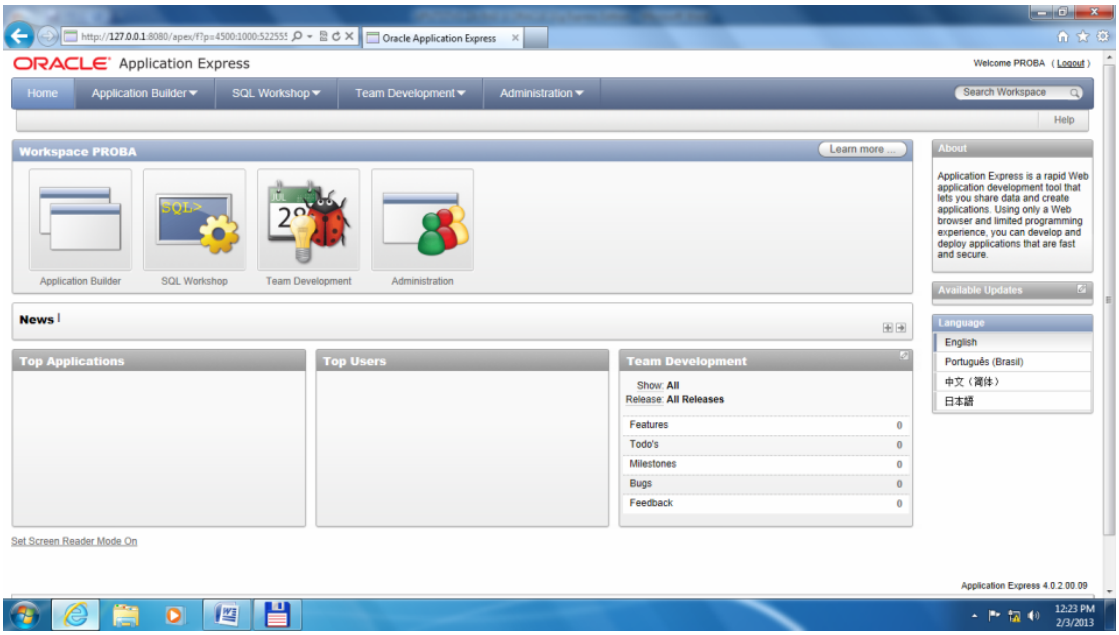


Da bismo nastavili sa radom, kliknemo na “click here” to begin opciju.
Dobijamo dijalog prozor za prijavljivanje novog korisnika:

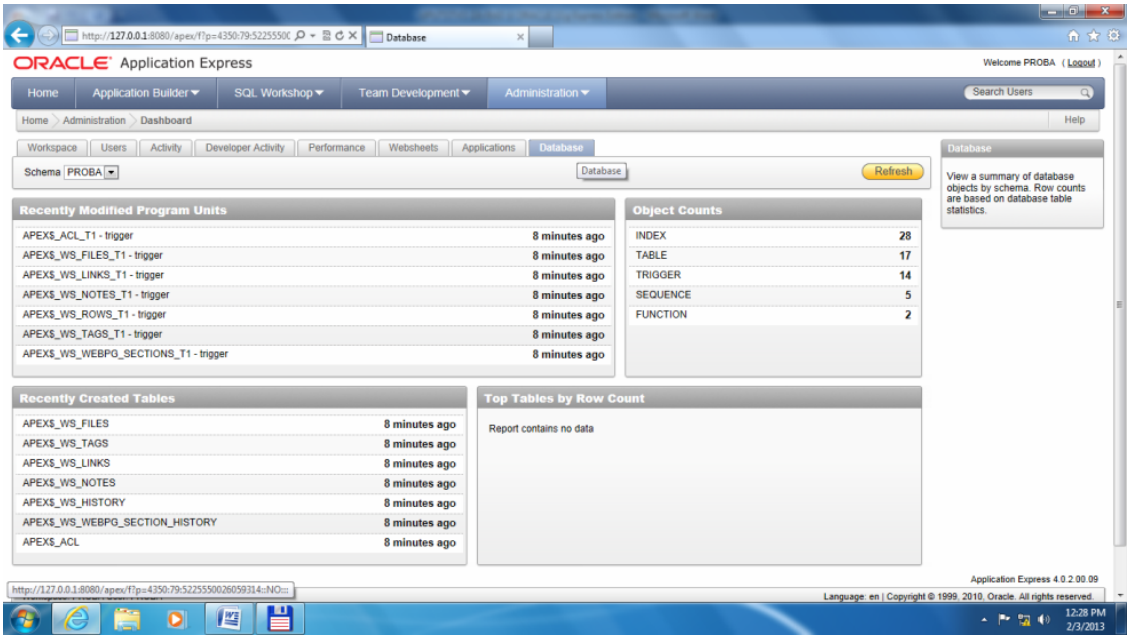


Unosimo šifru:

i dobijamo:

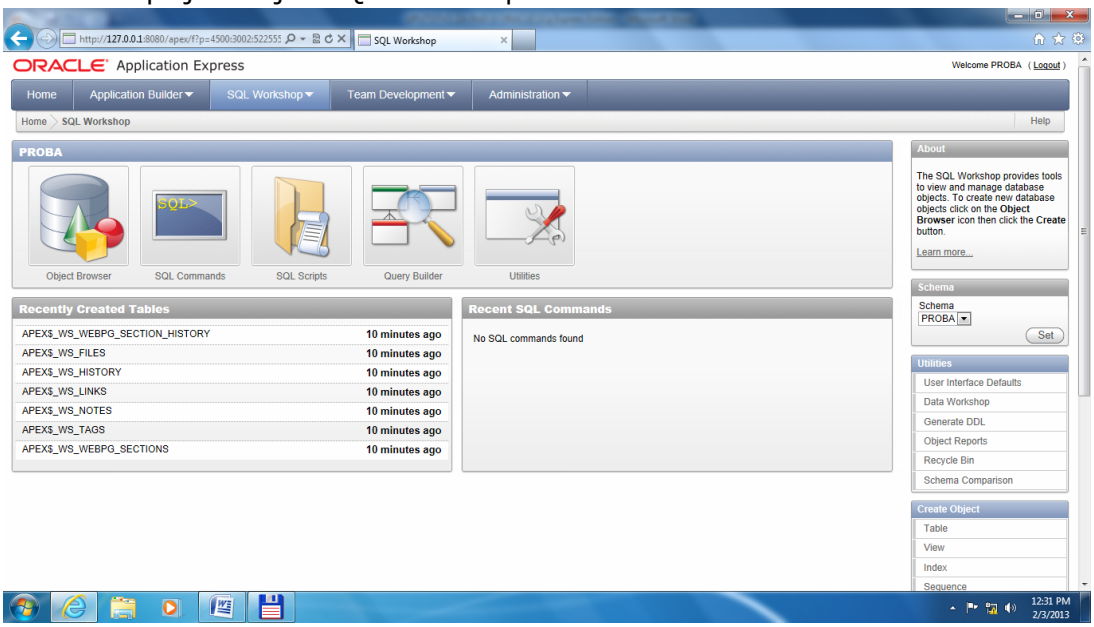


Možemo pogledati odeljak Administration – Dashboards – Database, gde se mogu pratiti promene nad bazom podataka:

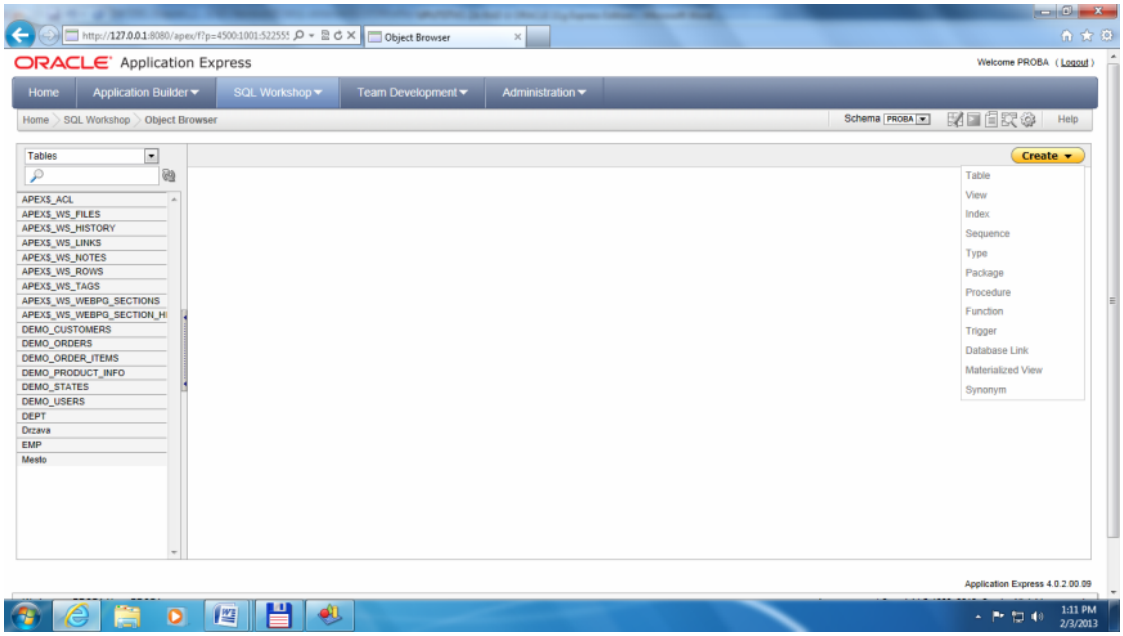


KREIRANJE TABELE U BAZI PODATAKA

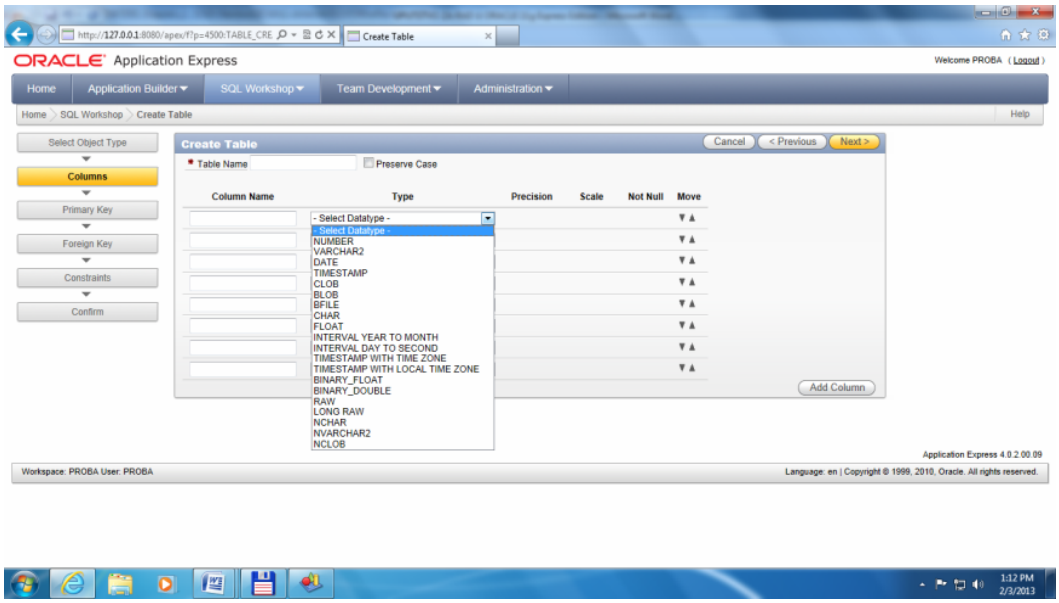
Koristimo opcije odeljka SQL Workshop.



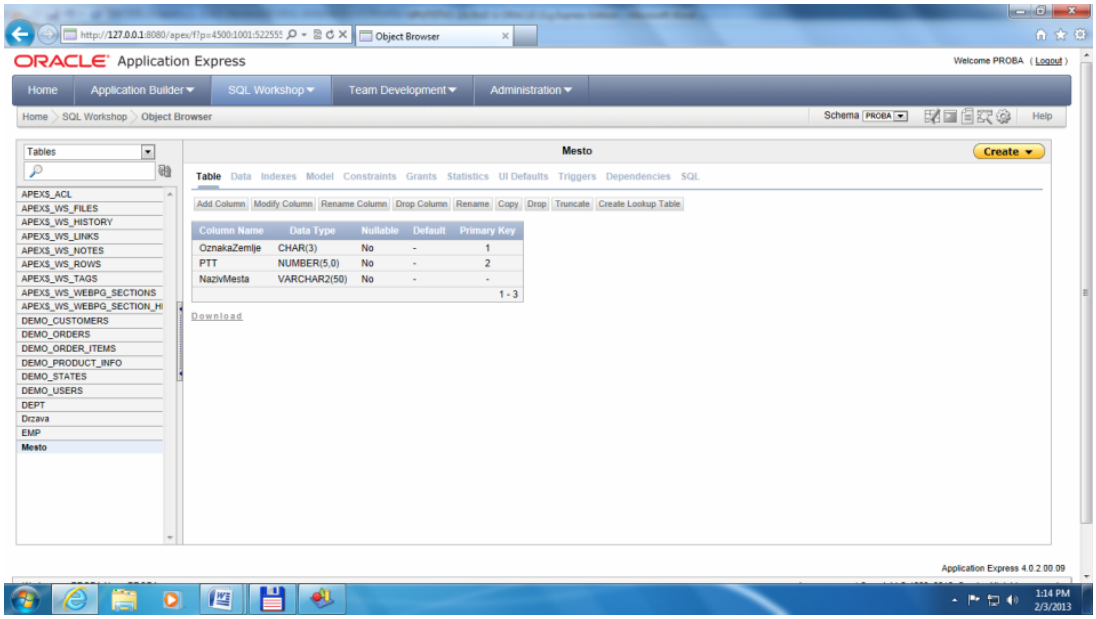
Kreiranje tabela moze da se vrši i koristeći korisnički interfejs – SQL Workshop, Object Browser, Create opcija u desnom uglu:



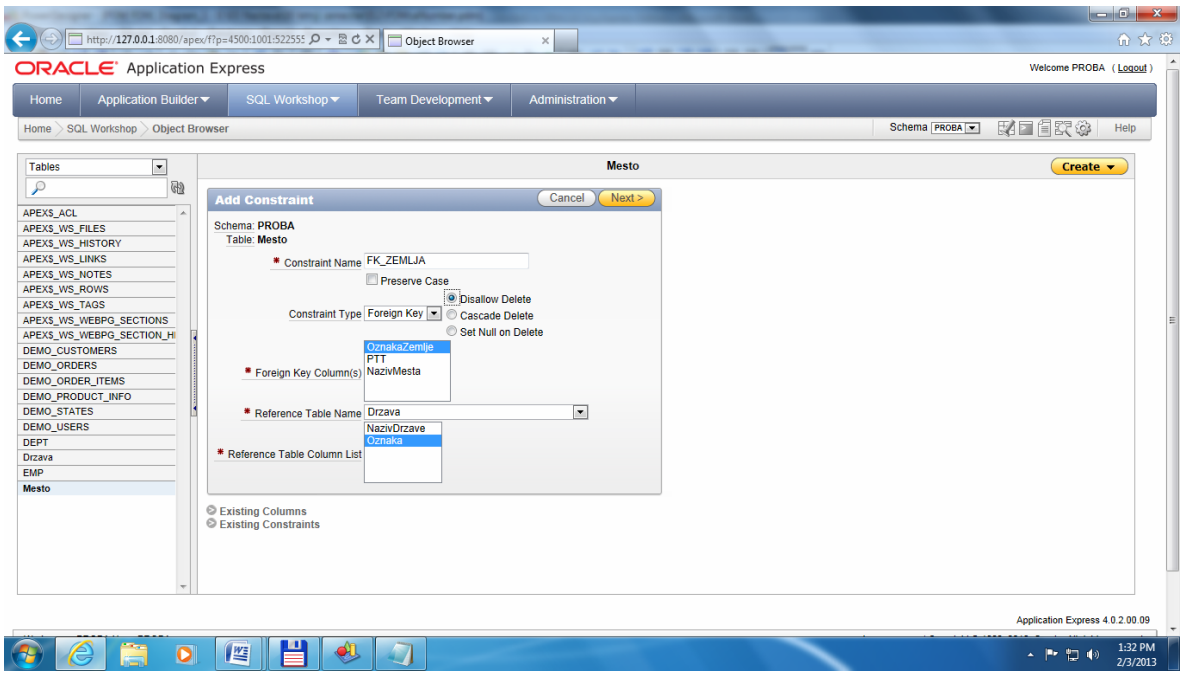
Ako izaberemo Create – Table – treba da dodelimo naziv tabeli i odredimo polja, tipove podataka itd:



Za svaku kreiranu tabelu iz baze podataka u SQL workshop – Object Browseru imamo detaljni prikaz, klikom na naziv tabele sa levog spiska:



Nad tabelom Mesto dodajemo novi Constraint – Constraints, Add Constraint:



7.3.2.2. Kreiranje baze podataka pomoću SQL naredbi

Prethodni primeri iz MS SQL Server baze podataka podržani su istim ili sličnim SQL upitima u ORACLE bazi podataka, što je prikazano u nastavku.

```
CREATE TABLE proizvod(
sifra_proizvoda nvarchar2(10) not null,
naziv_proizvoda nvarchar2(100) not null,
sifra_jedinice_mere int null,
kolicina_po_pakovanju int null,
datum_pocetka_proizvodnje date null
)

ALTER TABLE proizvod
```

```
DROP COLUMN kolicina_po_pakovanju

ALTER TABLE proizvod
ADD kolicina_po_pakovanju float


ALTER TABLE proizvod
ADD kolicina_na_lageru int not null

CREATE TABLE JedinicaMere(
sifra int not null,
naziv nvarchar2(40) not null
)

ALTER TABLE JedinicaMere
ADD oznaka nvarchar2(10) not null

ALTER TABLE JedinicaMere
ADD CONSTRAINT PK_JM PRIMARY KEY (sifra)

ALTER TABLE proizvod
ADD CONSTRAINT PK_Proizvod PRIMARY KEY (sifra_proizvoda)

ALTER TABLE JedinicaMere
ADD CONSTRAINT AK_naziv unique (naziv)

ALTER TABLE proizvod
ADD CONSTRAINT AK_nazivproizvoda unique (naziv_proizvoda)

ALTER TABLE proizvod
ADD CONSTRAINT FK_proizvod_JM FOREIGN KEY (sifra_jedinice_mere)
REFERENCES JedinicaMere (sifra)

CREATE INDEX FK_proizvodJM
ON proizvod(sifra_jedinice_mere)


ALTER TABLE proizvod
ADD CONSTRAINT ck_kolicinapakovanje CHECK (kolicina_po_pakovanju>0)
```

TRIGER:

IDENTIFIKATOR +1

```
CREATE OR REPLACE TRIGGER  "INSERT_DEMO_CUST"
BEFORE INSERT ON demo_customers
FOR EACH ROW
DECLARE
  cust_id number;
BEGIN
  SELECT demo_cust_seq.nextval
    INTO cust_id
    FROM dual;
  :new.CUSTOMER_ID := cust_id;
END;
/
ALTER TRIGGER  "INSERT_DEMO_CUST" ENABLE;
```

AZURIRANJE UKUPNOG IZNOSA U ZAGLAVLJU U ODNOSU NA STANJE STAVKI

```
CREATE OR REPLACE TRIGGER  "UPDATE_ORDER_TOTAL"
after insert or update or delete on demo_order_items
begin

-- Update the Order Total when any order item is changed

update demo_orders set order_total =
  (select sum(unit_price*quantity) from demo_order_items
    where demo_order_items.order_id = demo_orders.order_id);
end;
/
ALTER TRIGGER  "UPDATE_ORDER_TOTAL" ENABLE;
```

POGLED

```
CREATE VIEW RobaKojeImaNaLageru
AS
SELECT proizvod.naziv_proizvoda as "Naziv proizvoda", proizvod.kolicina_na_lageru as
"Kolicina" FROM proizvod WHERE proizvod.kolicina_na_lageru>0
```

NAPOMENA:
U ORACLE bazi podataka više reči koje su odvojene razmakom navodimo u okviru navodnika.

PROCEDURA

```
create or replace procedure "ROBAZAJM"
(jm IN VARCHAR2)
is
begin
SELECT proizvod.naziv_proizvoda FROM proizvod INNER JOIN JedinicaMere ON
proizvod.sifra_jedinice_mere=JedinicaMere.sifra WHERE JedinicaMere.naziv = JM;
end;
```

7.3.3. Rad sa podacima

UNOS PODATAKA

```
INSERT INTO JedinicaMere (sifra, naziv, oznaka) VALUES (1, 'kilogram', 'kg')

INSERT INTO proizvod
VALUES ('P1', 'pasteta', 1, 0.3, 5, '1/3/2013')
```

NAPOMENA:
U ORACLE bazi podataka numeričke vrednosti se navode bez apostrofa, realan broj sa tačkom, string i datum sa apostrofima. Datum se navodi u američkom stilu sa kosim crtama.

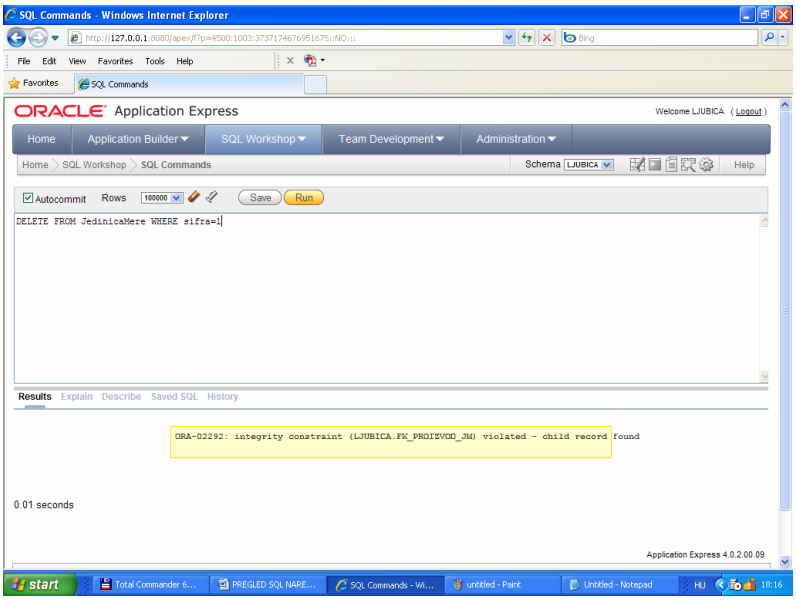
IZMENA PODATAKA

```
UPDATE proizvod
SET kolicina_po_pakovanju=0.2 WHERE sifra_proizvoda='P1'
```

BRISANJE PODATAKA

```
DELETE FROM JedinicaMere WHERE sifra=1
```

primer: Pokusacemo obrisati zapis iz tabele Jedinica mere, za koji postoje vezani podaci u tabeli proizvod i testirati pravila očuvanja referencijalnog integriteta.



Očigledno, kao i kod MS SQL Server baze podataka, nije moguće obrisati zapis u tabeli na strani 1, a da se pri tome ne naruši referenijalni integritet.

8. KONEKCIJA NA BAZU PODATAKA

8.1. Konekcija iz korisničkog interfejsa direktnim povezivanjem sa bazom podataka

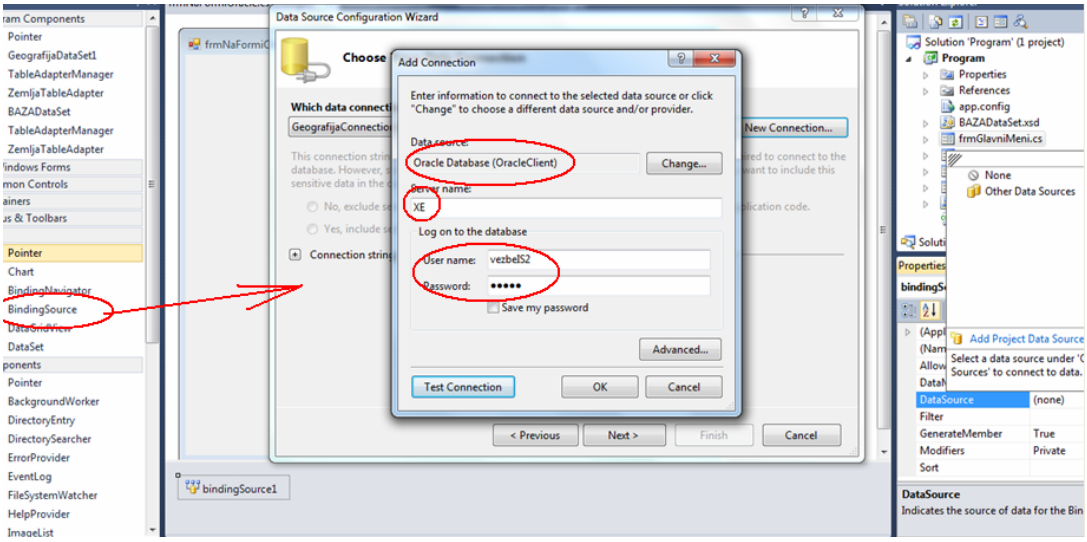
Ukoliko bi konekcija sa bazom podataka bila ostvarena putem binding source na korisničkom interfejsu, dobili bismo "project data source" i odgovarajuće stringove konekcije u app.config fajlu. Primeri Connection string-ova dobijenih na ovaj način:

ACCESS
Provider=Microsoft.Jet.OLEDB.4.0;Data Source="E:\03 Nastava\00 letnji semestar\osnovne IS2\CASOVI\05 cas KONEKCIJA\Baze podataka\Access\Geografija.mdb"

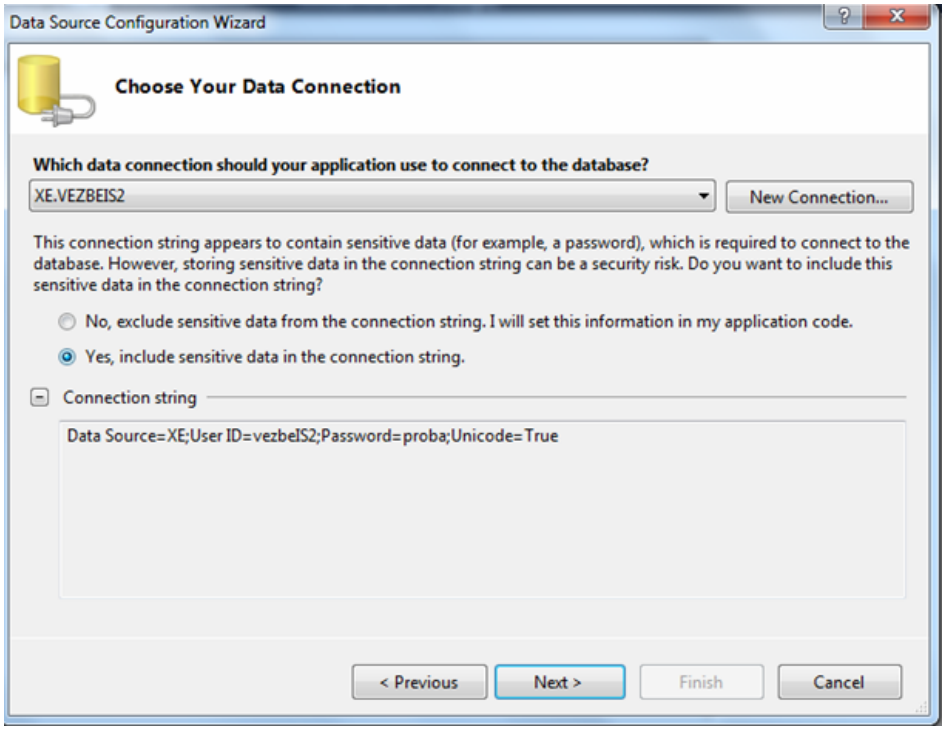
SQL
Data Source=.\SQLEXPRESS;AttachDbFilename="E:\03 Nastava\00 letnji semestar\osnovne IS2\CASOVI\05 cas KONEKCIJA\Baze podataka\SQL Server\BAZA.mdf";Integrated Security=True;Connect Timeout=30;User Instance=True

ORACLE
Data Source=XE;User ID=vezbeIS2;Password=proba;Unicode=True

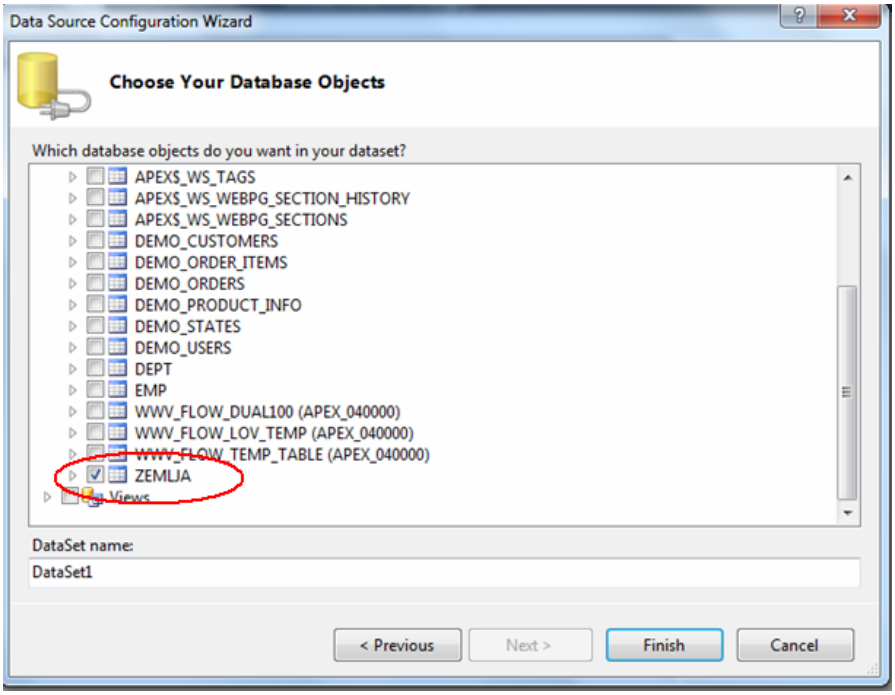
Naredna slika prikazuje ostvarivanje konekcije na ORACLE bazu podataka koristeći binding source.



Nakon toga dobijamo:



A zatim možemo birati tabelu iz baze podataka sa kojom bismo radili:



8.2. Konekcija programskim kodom pomoću gotovih klasa

Industrijske klase su klase iz gotovih biblioteka klasa u okviru razvojnog okruženja Visual Studio .NET. Za ostvarivanje konekcije ka bazi podataka možemo koristiti:

1. Za Ms Access bazu podataka – OleDBConnection klasu
2. Za SQL Server bazu podataka – SqlConnection
3. Za Oracle bazu podataka - OracleConnection

ACCESS

```
      OleDbConnection AccessKonekcija = new OleDbConnection();
      string putanjaAccessBaze = "F:\\03 Nastava\\00 letnji semestar\\osnovne IS2\\CASOVI\\02 cas KREIRANJE BAZE
PODATAKA\\0  kreiranje baze podataka\\PRIMER mesto zemlja\\MS Access\\IzCASEAccess2000.mdb";
 ILI
        string AccessStringKonekcije = "Jet OLEDB:Global Partial Bulk Ops=2;Jet OLEDB:Registry Path=;Jet
OLEDB:Database Locking Mode=1;Data Source='" + putanjaAccessBaze + "';Mode=Share Deny None;Jet
OLEDB:Engine Type=5;Provider='Microsoft.Jet.OLEDB.4.0';Jet OLEDB:System database=;Jet OLEDB:SFP=False;persist
security info=False;Extended Properties=;Jet OLEDB:Compact Without Replica Repair=False;Jet OLEDB:Encrypt
Database=False;Jet OLEDB:Create System Database=False;Jet OLEDB:Don't Copy Locale on Compact=False;User
ID=Admin;Jet OLEDB:Global Bulk Transactions=1";

        AccessKonekcija.ConnectionString = AccessStringKonekcije;
        try
        {
           AccessKonekcija.Open();
           MessageBox.Show("uspeh Access konekcije!");
        }
        catch (Exception ex)
        {
           MessageBox.Show("NASTUPILA JE GRESKA:" + ex.Message.ToString());
        }
```

SQL SERVER

```
         SqlConnection SQLKonekcija = new SqlConnection();
        string SQLStringKonekcije = "Data Source=ASUS\\SQLEXPRESS;Initial Catalog=Konferencija;Integrated
Security=True";
        SQLKonekcija.ConnectionString = SQLStringKonekcije;
        try
        {
           SQLKonekcija.Open();
           MessageBox.Show("uspeh SQL konekcije!");
        }
        catch (Exception ex)
        {
           MessageBox.Show("NASTUPILA JE GRESKA:" + ex.Message.ToString());
        }
```

ORACLE

```
        Oracle.DataAccess.Client.OracleConnection OraConn = new Oracle.DataAccess.Client.OracleConnection();
         string oradb="Data Source=XE;User Id=system;Password=oracleadmin;";
        ILI
         string oradb = "Data Source=(DESCRIPTION=(ADDRESS_LIST="
        + "(ADDRESS=(PROTOCOL=TCP)(HOST=XE)(PORT=4950)))"
        + "(CONNECT_DATA=(SERVER=DEDICATED)(SERVICE_NAME=ORCL)));"
        + "User Id=system;Password=oracleadmin;";
        ILI
        string oradb = "Data
Source=(DESCRIPTION=((PROTOCOL=TCP)(CONNECT_DATA=((SERVER=DEDICATED)(SERVICE_NAME=XE))));"
        OraConn.ConnectionString = oradb;
         try
         {
         OraConn.Open();
         }
         catch (Exception ex)
         {
              MessageBox.Show ("NASTUPILA JE GRESKA:" + ex.Message.ToString () );
           }
```

9. KREIRANJE BIBLIOTEKE KLASA SLOJA PODATAKA

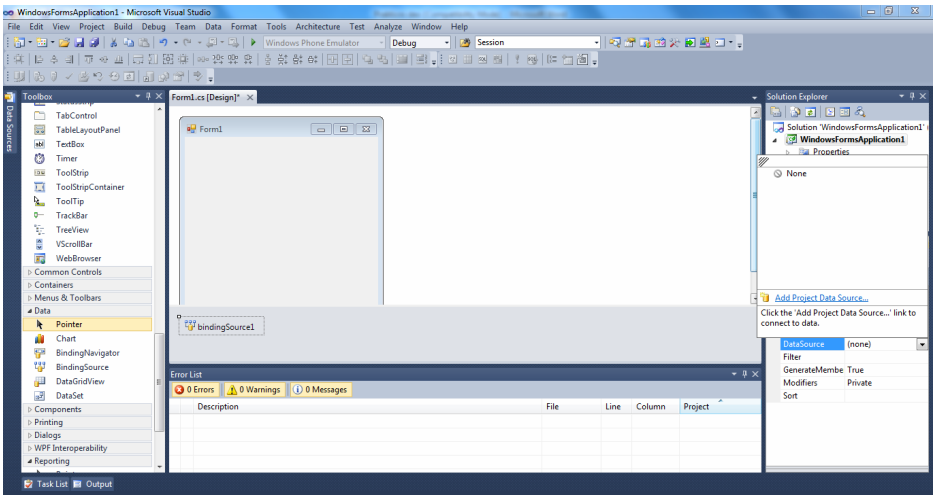
S obzirom na generalnu orjentaciju razvoja aplikacija u Visual Studio NET okruzenju, uputstva za rad se odnose na Visual Studio NET 2010.

U zavisnosti od načina rada sa klasama sloja podataka, možemo izabrati opcije:

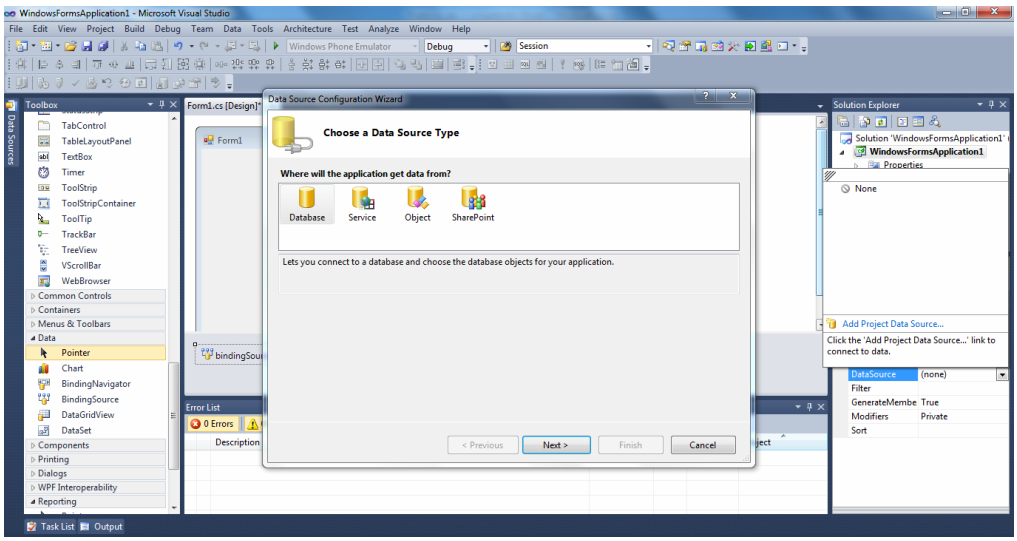
1. Korišćenje generatora programskog koda klasa, gde je generisana biblioteka klasa sastavni deo projekta korisničkog interfejsa i direktno je povezana sa bazom podataka (Entity Framework).
2. Kreiranje sopstvenih klasa korišćenjem ClassLibrary projekta.

9.1. Generisanje klasa korišćenjem ENTITY FRAMEWORK-a

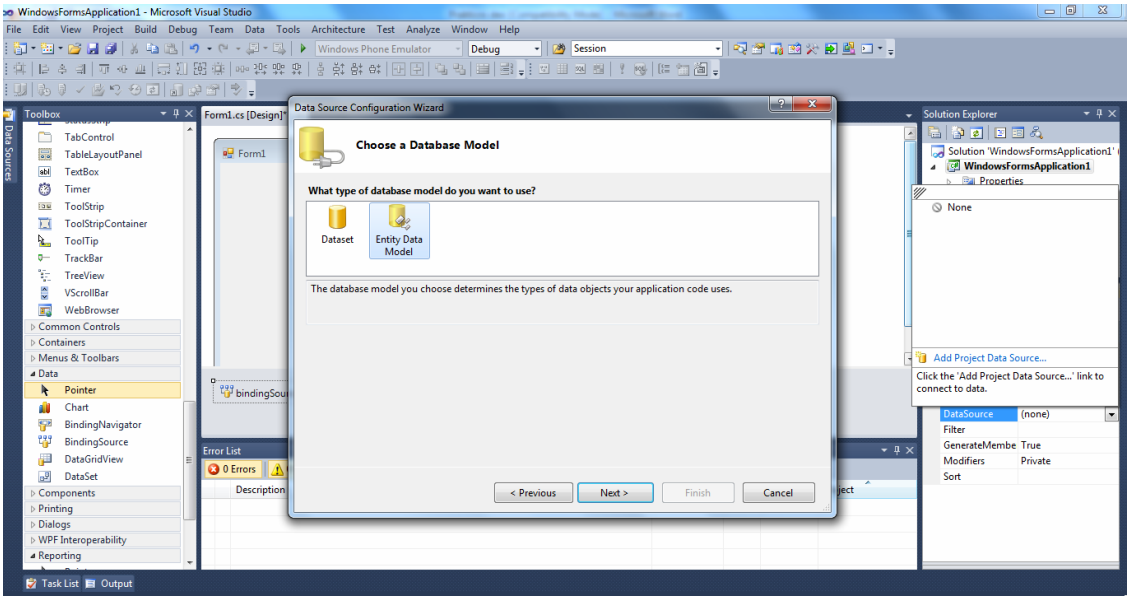
1. Način - Koristeći generatore koda koji generišu klase za klasa tehnološki deo i deo sličan strukturi tabela u bazama podataka, koristeći Entity framework na sledeći način:
   - Postavljanjem binding source na korisnički interfejs



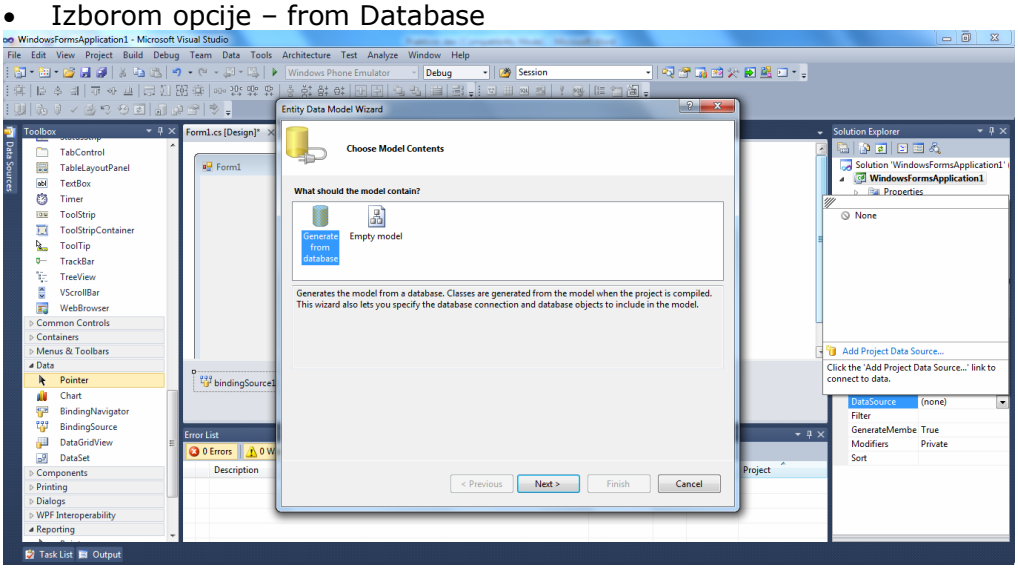
   - Izborom opcije Add project data source… biranjem Data source - Database



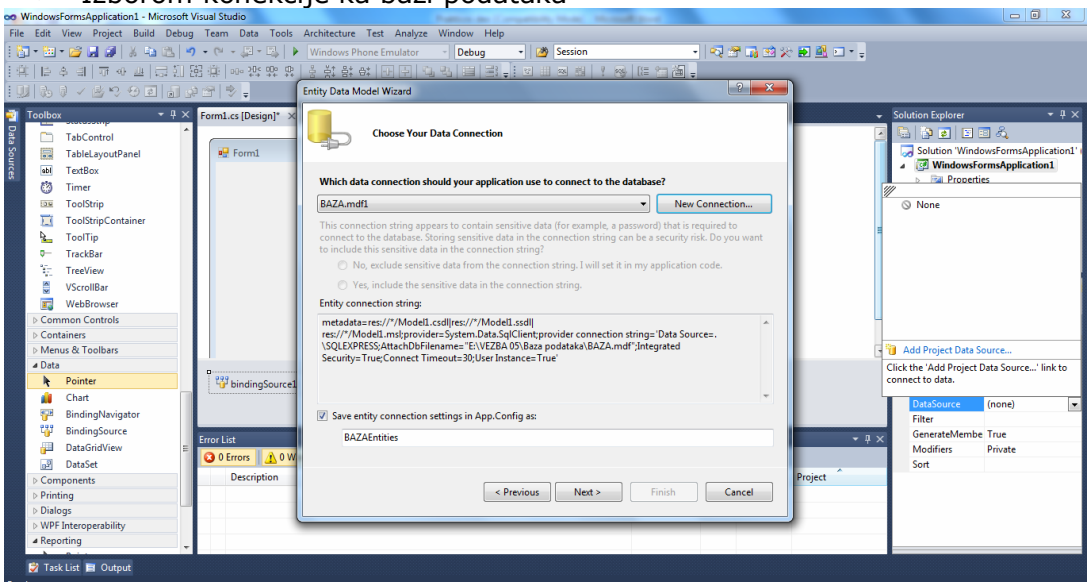
   - Izborom opcije Entity Data Model



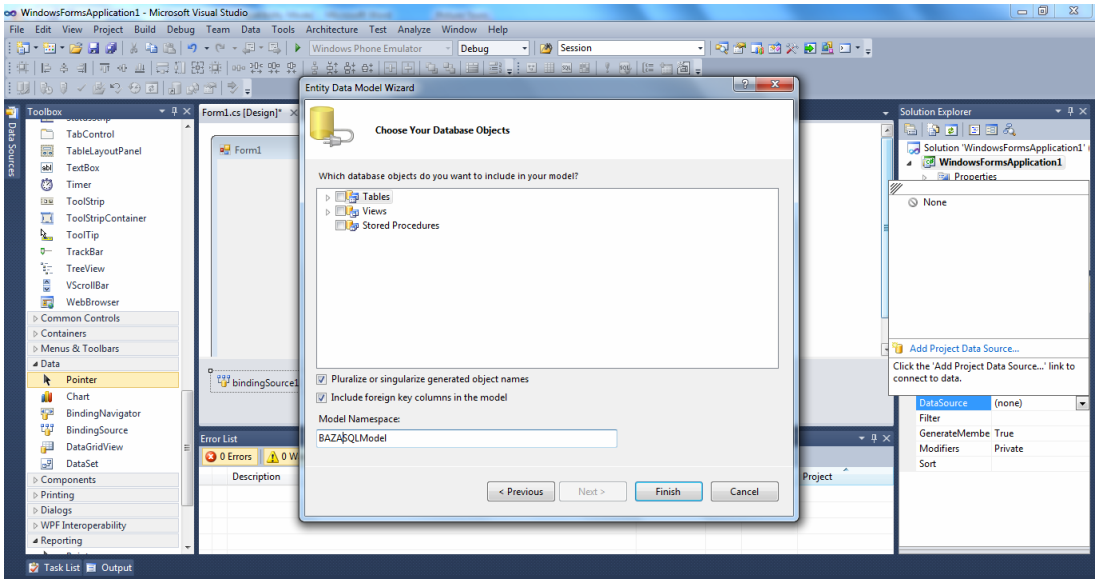
- Izborom opcije – from Database



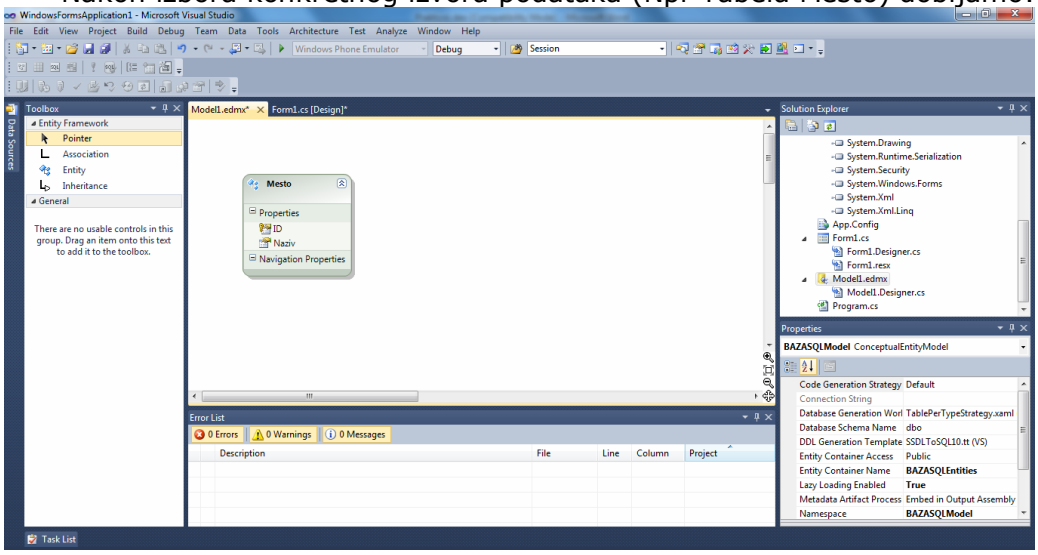
- Izborom konekcije ka bazi podataka



- Izborom izvora podataka – tabele, pogledi ili stored procedure



- Nakon izbora konkretnog izvora podataka (Npr Tabela Mesto) dobijamo:



- Konačno, dobijamo generisan kod klase BAZASQLEntities koja predstavlja zapravo kontejnersku klasu za ostale klase entitete, u ovom slučaju Mesto. Ova klasa je direktno povezana sa bazom podataka, a čiji se izvorni kod nalazi u okviru projekta korisničkog interfejsa:

```
//------------------------------------------------------------------------------
// <auto-generated>
//    This code was generated from a template.
//
//    Manual changes to this file may cause unexpected behavior in your application.
//    Manual changes to this file will be overwritten if the code is regenerated.
// </auto-generated>
//------------------------------------------------------------------------------

using System;
using System.Data.Objects;
using System.Data.Objects.DataClasses;
using System.Data.EntityClient;
using System.ComponentModel;
using System.Xml.Serialization;
using System.Runtime.Serialization;
```

```
[assembly: EdmSchemaAttribute()]

namespace WindowsFormsApplication1
{
    #region Contexts

    /// <summary>
    /// No Metadata Documentation available.
    /// </summary>
    public partial class BAZASQLEntities : ObjectContext
    {
        #region Constructors

        /// <summary>
        /// Initializes a new BAZASQLEntities object using the connection string found in the 'BAZASQLEntities' section of the application configuration file.
        /// </summary>
        public BAZASQLEntities() : base("name=BAZASQLEntities", "BAZASQLEntities")
        {
            this.ContextOptions.LazyLoadingEnabled = true;
            OnContextCreated();
        }

        /// <summary>
        /// Initialize a new BAZASQLEntities object.
        /// </summary>
        public BAZASQLEntities(string connectionString) : base(connectionString, "BAZASQLEntities")
        {
            this.ContextOptions.LazyLoadingEnabled = true;
            OnContextCreated();
        }

        /// <summary>
        /// Initialize a new BAZASQLEntities object.
        /// </summary>
        public BAZASQLEntities(EntityConnection connection) : base(connection, "BAZASQLEntities")
        {
            this.ContextOptions.LazyLoadingEnabled = true;
            OnContextCreated();
        }

        #endregion

        #region Partial Methods

        partial void OnContextCreated();

        #endregion

        #region ObjectSet Properties

        /// <summary>
        /// No Metadata Documentation available.
        /// </summary>
        public ObjectSet<Mesto> Mestoes
        {
            get
            {
                if ((_Mestoes == null))
                {
                    _Mestoes = base.CreateObjectSet<Mesto>("Mestoes");
                }
                return _Mestoes;
            }
        }
        private ObjectSet<Mesto> _Mestoes;

        #endregion
        #region AddTo Methods
```

```
        /// <summary>
        /// Deprecated Method for adding a new object to the Mestoes EntitySet. Consider using the .Add method of the associated ObjectSet&lt;T&gt; property instead.
        /// </summary>
        public void AddToMestoes(Mesto mesto)
        {
            base.AddObject("Mestoes", mesto);
        }

        #endregion
    }


    #endregion

    #region Entities

    /// <summary>
    /// No Metadata Documentation available.
    /// </summary>
    [EdmEntityTypeAttribute(NamespaceName="BAZASQLModel", Name="Mesto")]
    [Serializable()]
    [DataContractAttribute(IsReference=true)]
    public partial class Mesto : EntityObject
    {
        #region Factory Method

        /// <summary>
        /// Create a new Mesto object.
        /// </summary>
        /// <param name="id">Initial value of the ID property.</param>
        public static Mesto CreateMesto(global::System.String id)
        {
            Mesto mesto = new Mesto();
            mesto.ID = id;
            return mesto;
        }

        #endregion
        #region Primitive Properties

        /// <summary>
        /// No Metadata Documentation available.
        /// </summary>
        [EdmScalarPropertyAttribute(EntityKeyProperty=true, IsNullable=false)]
        [DataMemberAttribute()]
        public global::System.String ID
        {
            get
            {
                return _ID;
            }
            set
            {
                if (_ID != value)
                {
                    OnIDChanging(value);
                    ReportPropertyChanging("ID");
                    _ID = StructuralObject.SetValidValue(value, false);
                    ReportPropertyChanged("ID");
                    OnIDChanged();
                }
            }
        }
        private global::System.String _ID;
        partial void OnIDChanging(global::System.String value);
        partial void OnIDChanged();

        /// <summary>
        /// No Metadata Documentation available.
        /// </summary>
```

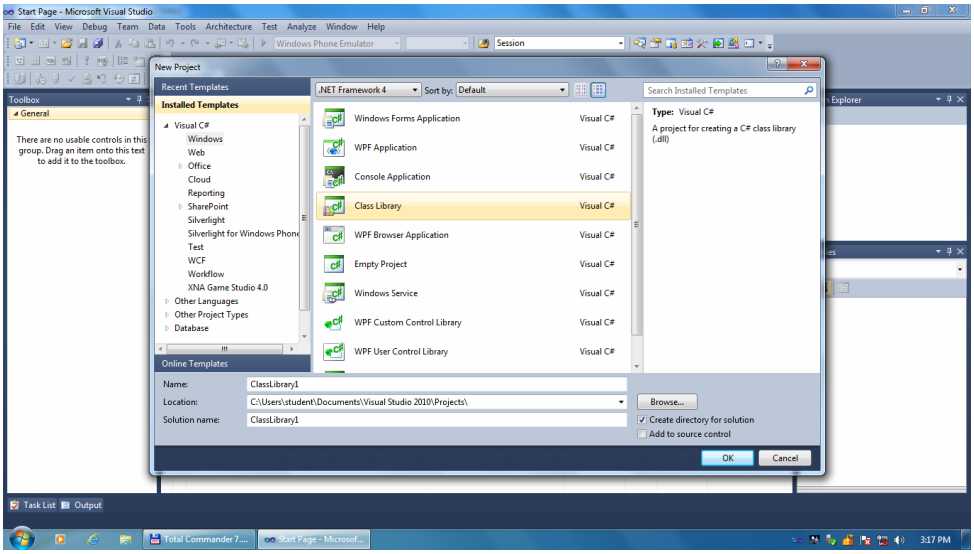
```
        [EdmScalarPropertyAttribute(EntityKeyProperty=false, IsNullable=true)]
        [DataMemberAttribute()]
        public global::System.String Naziv
        {
            get
            {
                return _Naziv;
            }
            set
            {
                OnNazivChanging(value);
                ReportPropertyChanging("Naziv");
                _Naziv = StructuralObject.SetValidValue(value, true);
                ReportPropertyChanged("Naziv");
                OnNazivChanged();
            }
        }
        private global::System.String _Naziv;
        partial void OnNazivChanging(global::System.String value);
        partial void OnNazivChanged();

        #endregion

    }

    #endregion

}
```

9.2. Kreiranje sopstvenih opštih tehnoloških klasa

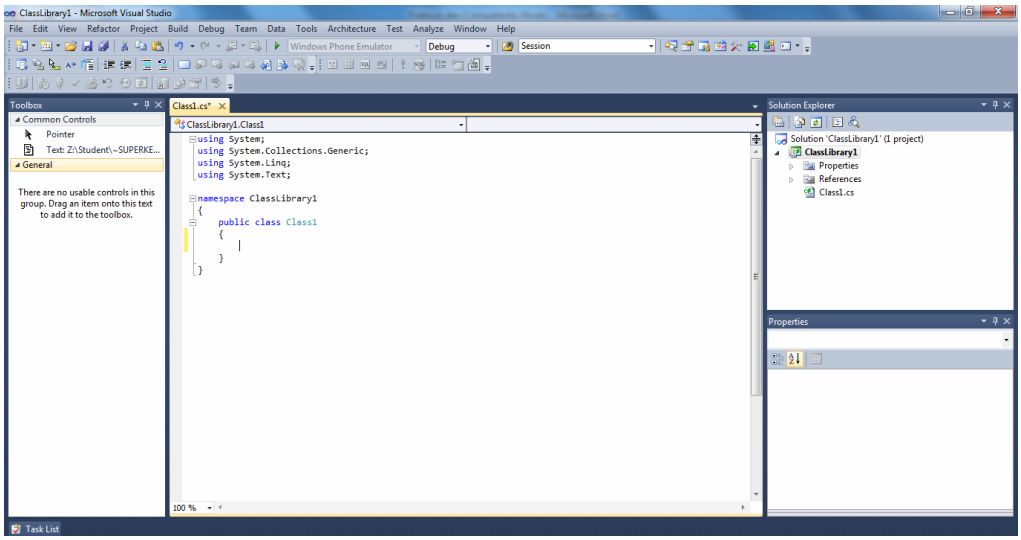
Drugi način odnosi se na kreiranje sopstvenih klasa koristeći poseban ClassLibrary tip projekta. Drugi način je pogodniji zbog podrške upravljanju promenama i zastupljen je u okviru praktične nastave na nastavnom predmetu Informacioni sistemi 2. Omogućena je podrška:

- Promeni lokacije (putanje, naziva DBMS servera) baze podataka - realizovana je koncepcija da se putanja baze podataka i ostala podešavanja u vezi konekcije na bazu podataka čuvaju u eksternom konfiguracionom fajlu (za Ms Access bazu podataka dovoljna je samo putanja koja se čuva u „putanjabaze.txt" fajlu, dok se za SQL Server bazu podataka čuvaju u XML fajlu.
- Promeni tipa baze podataka – kompletna realizacija programskog koda korisničkog interfejsa koja se odnosi na rad sa podacima dodeljena je klasama iz sloja podataka. U okviru odgovarajuće biblioteke klasa (odnosno dll-a) nalaze se klase koje su tehnološki zavisne od tzv. Industrijskih bazičnih klasa za rad sa odgovarajućim tipom DBMS-a, a semantički nezavisne od naziva i strukture baze podataka. Ukoliko bi došlo do potrebe za promenom tipa baze podataka, odnosno DBMS-a, tada bismo jednostavnom zamenom jednog dll-a dobili mogućnost da korisnički interfejs može da radi sa drugim tipom baze podataka bez potrebe za ponovnim programiranjem. Dovoljno je imati biblioteke klasa za podršku različitim DBMS-ovima.

  - Biramo opciju Windows / Class Library.

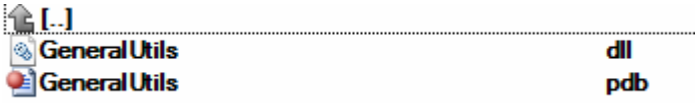


- Dobijamo prostor za pisanje programskog koda klasa.



Nakon kreiranja klasa, kompajliranjem dobijamo blioteku klasa kao poseban projekat, odnosno *.dll fajl u izvrsnom obliku. Svaki projekat tipa Class Library automatski dobija podfoldere, a u bin/debug podfolderu:

Nalazi se odgovarajuci *.dll fajl.



Kreirane su sopstvene biblioteke klasa :

1. GeneralUtils – rad sa tekstualnim datotekama i druge opste usluzne klase
2. DataInterOpUtils – rad sa datotekama za razmenu podataka (interoperabilnost) – XML, XLS
3. SQLDBUtils – univerzalne klase za rad sa SQL Server bazom podataka

Izvorni kod klasa dat je u nastavku.

9.2.1. Klase za rad sa TXT, XML i XLS fajlovima

U nastavku je dat izvorni kod klasa za rad sa tekstualnim datotekama (BIBLIOTEKA KLASA GeneralUtils), kao i za podršku interoperabilnosti aplikacija omogućavanjem eksportovanja podataka u XML i XLS format (BIBLIOTEKA KLASA DataInteropUtils)..

BIBLIOTEKA - GENERAL UTILS

KLASA ZA RAD SA TEKSTUALNOM DATOTEKOM

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace GeneralUtils
{
   public static class clsTXTfajl
   {
      public static string ProcitajFajl(string PutanjaFajla)
      {
         string sadrzajFajla;
         System.IO.StreamReader file = new System.IO.StreamReader(PutanjaFajla);
         sadrzajFajla = file.ReadToEnd();
         file.Close();
         return sadrzajFajla;
      }
   }
}
```

BIBLIOTEKA - DATA INTEROP UTILS

KLASA ZA RAD SA XML FAJLOVIMA

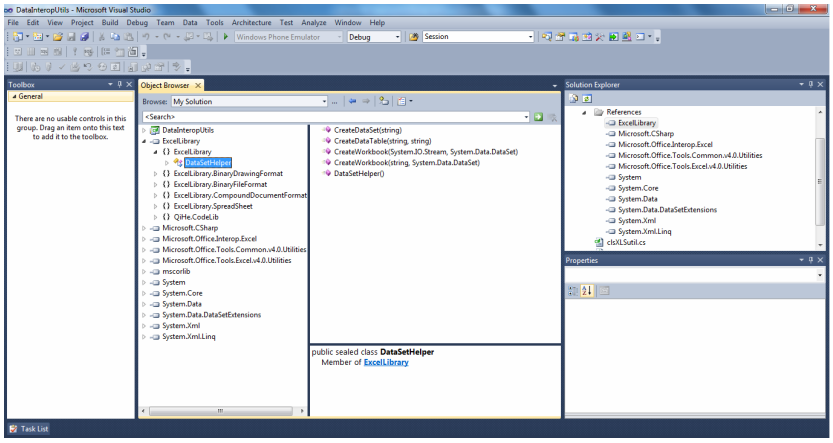
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace DataInteropUtils
{
   public static class clsXMLUtil
   {
      public static void ObrisiXMLfajl(string PutanjaiNazivXMLFajla)
      {
         System.IO.FileInfo pomfile = new System.IO.FileInfo(PutanjaiNazivXMLFajla);
         pomfile.Delete();
      }

      public static string KreirajXMLfajl(System.Data.DataSet ds, bool ProveraPostojanja, string PutanjaiNazivXMLFajla,
string PutanjaiNazivXMLSheme)
      {
         System.IO.FileInfo pomfile = new System.IO.FileInfo(PutanjaiNazivXMLFajla);

         if ((ProveraPostojanja == true) && (pomfile.Exists))
         {
            return "Ne mozete kreirati fajl jer je vec kreiran. Zadnji put kreiran fajl - datum:" +
pomfile.LastWriteTime.Date.Day.ToString() + "." + pomfile.LastWriteTime.Date.Month.ToString() + "." +
```

```
pomfile.LastWriteTime.Date.Year.ToString() + " vreme:" + pomfile.LastWriteTime.Hour.ToString() + ":" +
pomfile.LastWriteTime.Minute.ToString() + ":" + pomfile.LastWriteTime.Second.ToString();
            }
            else
            {
                ds.WriteXml(PutanjaiNazivXMLFajla);
                ds.WriteXmlSchema(PutanjaiNazivXMLSheme);
                return "Uspesno ste kreirali XML fajl";
            }
        }

        public static System.Data.DataSet UcitajXMLuDataset(string PutanjaINazivXMLfajla)
        {
            System.Data.DataSet ds = new System.Data.DataSet();
            ds.ReadXml(PutanjaINazivXMLfajla);
            return ds;
        }

    }
}
```

KLASA ZA RAD SA XLS (EXCEL) FAJLOVIMA

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
//
using ExcelLibrary;

namespace DataInteropUtils
{
    public static class clsXLSutilALT
    {
        public static bool SnimiXLS(System.Data.DataSet ds, string nazivXLSfajla)
        {
            try
            {
                ExcelLibrary.DataSetHelper.CreateWorkbook(nazivXLSfajla, ds);
            }
            catch
            {
                return false;
            }
            return true;
        }
    }
}
```

Za realizaciju ove klase koristi se (ukljucena je u References deo projekta biblioteke klasa DataInteropUtils) dodatna biblioteka ExcelLibrary koja ima klasu DatasetHelper i usluzne metode za rad sa XLS fajlovima.



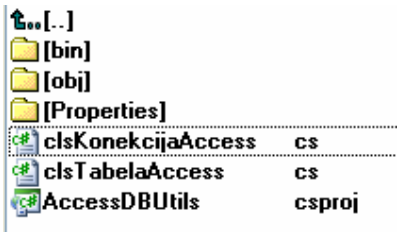
9.2.2. Klase za rad sa MS Access bazom podataka

Industrijske klase razvojnog okruženja Visual Studio .NET koje omogućavaju rad sa MS Access bazom podataka nalaze se u biblioteci klasa ***System.Data.OleDb.*** To su sledeće klase:

- OleDbConnection
- OleDbCommand
- OleDbTransaction
- OleDbDataAdapter

Takođe, koristi se i *univerzalna* klasa ***System.Data.DataSet*** – memorijska kolekcija slogova.

Kreiran je projekat biblioteke klasa koji bi olakšao rad sa industrijskim klasama. Biblioteka klasa se zove AccessDBUtils i sastoji se od dve klase: clsKonekcijaAccess i clsTabelaAccess.



Suština je u tome da se najpre pomoću klase clsKonekcijaAccess uspostavlja konekcija ka bazi podataka, koja se nalazi na putanji koja je zapisana u eksternom putanjabaze.txt fajlu. Čitanje ovog fajla može da se vrši iz same klase, ali u ovom slučaju je predviđeno da se čita eksterno (u okviru korisničkog interfejsa) I prosleđuje klasi samo putanja baze. Nakon uspešne konekcije, koristimo klase clsTabelaAccess kako bismo čitali ili ažurirali podatke iz baze. Realizacija metode IzvrsiAzuriranje pokriva sve operacije – unos, brisanje I izmena podataka, a sama implementacija radjena je koristeci transakcije (mada u ovom slucaju jedne tabele nije bitno da bude realizovano koristeci transakcije).

Izvorni kod klasa dat je u nastavku:

KLASA clsKonekcijaAccess ZA RAD SA KONEKCIJOM NA BAZU PODATAKA

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
// dodato
using System.Data.OleDb;

namespace AccessDBUtils
{
   public class clsKonekcijaAccess

   {
      /* ODGOVORNOST: Konekcija na celinu baze podataka, Access tipa  */

      #region Atributi
      private OleDbConnection pKonekcija;
      private string pPutanjaAccessBaze;
      #endregion

      #region Konstruktor
      public clsKonekcijaAccess(string putanjaAccessBaze)
      {
         pPutanjaAccessBaze = putanjaAccessBaze;
      }
      #endregion
```

```
      #region Privatne metode
      private string DajStringKonekcije(string putanjaAccessBaze)
      {
         string mStringKonekcije;
         // iz app.config:
         //Provider=Microsoft.Jet.OLEDB.4.0;Data Source=&amp;quot;C:\moje\Nastava\Lernji semestrar 2011\5
cas\RESENJE 5 cas\Baza podataka\Probna2000.mdb&amp;quot;
         // druga varijanta koja takodje radi je:
         mStringKonekcije = "Jet OLEDB:Global Partial Bulk Ops=2;Jet OLEDB:Registry Path=;Jet OLEDB:Database
Locking Mode=1;Data Source='" + putanjaAccessBaze + "';Mode=Share Deny None;Jet OLEDB:Engine
Type=5;Provider='Microsoft.Jet.OLEDB.4.0';Jet OLEDB:System database=;Jet OLEDB:SFP=False;persist security
info=False;Extended Properties=;Jet OLEDB:Compact Without Replica Repair=False;Jet OLEDB:Encrypt
Database=False;Jet OLEDB:Create System Database=False;Jet OLEDB:Don't Copy Locale on Compact=False;User
ID=Admin;Jet OLEDB:Global Bulk Transactions=1";
         return mStringKonekcije;
      }
      #endregion

      #region Javne metode
      public bool OtvoriKonekciju()
      {
         bool uspeh;
         pKonekcija = new OleDbConnection();
         pKonekcija.ConnectionString = DajStringKonekcije(pPutanjaAccessBaze);

         try
         {
            pKonekcija.Open();
            uspeh = true;
         }
         catch
         {
            uspeh = false;
         }
         return uspeh;
      }

      public OleDbConnection DajKonekciju()
      {
         return pKonekcija;
      }

      public void ZatvoriKonekciju()
      {
         pPutanjaAccessBaze = "";
         pKonekcija.Close();
         pKonekcija.Dispose();
      }

      #endregion

   }
}
```

KLASA clsTabelaAccess ZA RAD SA TABELAMA

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
// dodato
using System.Data.OleDb;

namespace AccessDBUtils
{
   public class clsTabelaAccess
   {
      #region Atributi

      private string pNazivTabele;
      private clsKonekcijaAccess pKonekcija;
```

```
private OleDbDataAdapter pAdapter;
private System.Data.DataSet pDataSet;

#endregion

#region Konstruktor

public clsTabelaAccess(clsKonekcijaAccess Konekcija, string NazivTabele)
{
   pKonekcija = Konekcija;
   pNazivTabele = NazivTabele;
}

#endregion

#region Privatne metode

private void KreirajAdapter(string SelectUpit, string InsertUpit, string DeleteUpit, string UpdateUpit)
{
   OleDbCommand mSelectKomanda, mInsertKomanda, mDeleteKomanda, mUpdateKomanda;

   mSelectKomanda = new OleDbCommand();
   mSelectKomanda.CommandText = SelectUpit;
   mSelectKomanda.Connection = pKonekcija.DajKonekciju();

   mInsertKomanda = new OleDbCommand();
   mInsertKomanda.CommandText = InsertUpit;
   mInsertKomanda.Connection = pKonekcija.DajKonekciju();

   mDeleteKomanda = new OleDbCommand();
   mDeleteKomanda.CommandText = DeleteUpit;
   mDeleteKomanda.Connection = pKonekcija.DajKonekciju();

   mUpdateKomanda = new OleDbCommand();
   mUpdateKomanda.CommandText = UpdateUpit;
   mUpdateKomanda.Connection = pKonekcija.DajKonekciju();

   pAdapter = new OleDbDataAdapter();
   pAdapter.SelectCommand = mSelectKomanda;
   pAdapter.InsertCommand = mInsertKomanda;
   pAdapter.UpdateCommand = mUpdateKomanda;
   pAdapter.DeleteCommand = mDeleteKomanda;
}

private void KreirajDataset()
{
   pDataSet = new System.Data.DataSet();
   pAdapter.Fill(pDataSet, pNazivTabele);

}

private void ZatvoriAdapterDataset()
{
   pAdapter.Dispose();
   pDataSet.Dispose();
}

#endregion

#region Javne metode

public System.Data.DataSet DajPodatke(string SelectUpit)
   // izdvaja podatke u odnosu na dat selectupit
{
   KreirajAdapter(SelectUpit, "", "", "");
   KreirajDataset();
   return pDataSet;
}

public int DajBrojSlogova()
{
```

```
            int BrojSlogova = pDataSet.Tables[0].Rows.Count;
            return BrojSlogova;
        }

        public bool IzvrsiAzuriranje(string Upit)
            // izvrzava azuriranje unos/brisanje/izmena u odnosu na dati i upit
        {

            //
            bool uspeh = false;
            OleDbConnection mKonekcija;
            OleDbCommand Komanda;
            OleDbTransaction mTransakcija = null;
            try
            {
                mKonekcija = pKonekcija.DajKonekciju();
                // aktivan kod

                // povezivanje
                Komanda = new OleDbCommand();
                Komanda.Connection = mKonekcija;
                Komanda = mKonekcija.CreateCommand();
                // pokretanje
                // NE TREBA OPEN JER DOBIJAMO OTVORENU KONEKCIJU KROZ KONSTRUKTOR
                // mKonekcija.Open();
                mTransakcija = mKonekcija.BeginTransaction();
                Komanda.Transaction = mTransakcija;
                Komanda.CommandText = Upit;
                Komanda.ExecuteNonQuery();
                mTransakcija.Commit();
                uspeh = true;
            }
            catch
            {
                mTransakcija.Rollback();
                uspeh = false;
            }
            return uspeh;
        }


        #endregion

    }
}
```

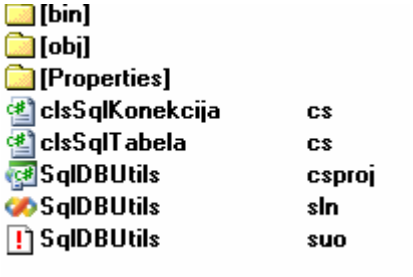
9.2.3. Klase za rad sa MS SQL Server bazom podataka

Industrijske klase razvojnog okruženja Visual Studio .NET koje omogućavaju rad sa MS SQL Server bazom podataka nalaze se u biblioteci klasa ***System.Data.SqlClient .*** To su sledeće klase:

- SqlConnection
- SqlDataAdapter
- SqlCommand
- SqlTransaction

Takođe, koristi se i *univerzalna* klasa ***System.Data.DataSet*** – memorijska kolekcija slogova.

Kreiran je projekat biblioteke klasa koji bi olakšao rad sa industrijskim klasama. Biblioteka klasa se zove SqlDBUtils i sastoji se od dve klase: clsSqlKonencija i clsSqlTabela.



Suština je u tome da se najpre pomoću klase clsSqlKonekcija uspostavlja konekcija ka bazi podataka, na osnovu putanje I naziva baze, kao I naziva MS SQL Server instance koji se nalaze u podesavanjebaze.XML fajlu. Svi parametri konekcije se preuzimaju u konstruktoru klase, a čitaju u korisničkom interfejsu iz XML fajla sa podešavanjima.

KLASA ZA RAD SA KONEKCIJOM

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
//
using System.Data.SqlClient;

namespace SqlDBUtils
{
   public class clsSqlKonekcija
   {
      /* ODGOVORNOST: Konekcija na celinu baze podataka, SQL server tipa  */

      #region Atributi
      private SqlConnection pKonekcija;
      //
      private string pPutanjaSQLBaze;
      private string pNazivBaze;
      private string pNazivSQL_DBMSinstance;
      #endregion

      #region Konstruktor
      public clsSqlKonekcija(string nazivSQL_DBMSInstance, string putanjaSqlBaze, string NazivBaze)
      {
         pPutanjaSQLBaze = putanjaSqlBaze;
         pNazivBaze = NazivBaze;
         pNazivSQL_DBMSinstance = nazivSQL_DBMSInstance;
      }
      #endregion

      #region Privatne metode
      private string DajStringKonekcije()
      {
         string mStringKonekcije;
         if (pPutanjaSQLBaze.Length.Equals(0) || pPutanjaSQLBaze==null)
         {
            mStringKonekcije = "Data Source=" + pNazivSQL_DBMSinstance + " ;Initial Catalog=" + pNazivBaze +
";Integrated Security=True";
         }
         else
         {
            mStringKonekcije = "Data Source=.\\" + pNazivSQL_DBMSinstance + ";AttachDbFilename=" +
pPutanjaSQLBaze + "\\" + pNazivBaze +  ";Integrated Security=True;Connect Timeout=30;User Instance=True";
         }
         return mStringKonekcije;
      }
      #endregion

      #region Javne metode
      public bool OtvoriKonekciju()
```

```
      {
         bool uspeh;
         pKonekcija = new SqlConnection();
         pKonekcija.ConnectionString = DajStringKonekcije();

         try
         {
            pKonekcija.Open();
            uspeh = true;
         }
         catch
         {
            uspeh = false;
         }
         return uspeh;
      }

      public SqlConnection DajKonekciju()
      {
         return pKonekcija;
      }

      public void ZatvoriKonekciju()
      {
         pPutanjaSQLBaze = "";
         pKonekcija.Close();
         pKonekcija.Dispose();
      }

      #endregion

   }
}
```

KLASA ZA RAD SA TABELAMA

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
//
using System.Data.SqlClient;

namespace SqlDBUtils
{
   public class clsSqlTabela
   {
      #region Atributi

      private string pNazivTabele;
      private clsSqlKonekcija pKonekcija;
      private SqlDataAdapter pAdapter;
      private System.Data.DataSet pDataSet;

      #endregion

      #region Konstruktor

      public clsSqlTabela(clsSqlKonekcija Konekcija, string NazivTabele)
      {
         pKonekcija = Konekcija;
         pNazivTabele = NazivTabele;
      }

      #endregion

      #region Privatne metode

      private void KreirajAdapter(string SelectUpit, string InsertUpit, string DeleteUpit, string UpdateUpit)
      {
         SqlCommand mSelectKomanda, mInsertKomanda, mDeleteKomanda, mUpdateKomanda;
```

```
    mSelectKomanda = new SqlCommand();
    mSelectKomanda.CommandText = SelectUpit;
    mSelectKomanda.Connection = pKonekcija.DajKonekciju();

    mInsertKomanda = new SqlCommand();
    mInsertKomanda.CommandText = InsertUpit;
    mInsertKomanda.Connection = pKonekcija.DajKonekciju();

    mDeleteKomanda = new SqlCommand();
    mDeleteKomanda.CommandText = DeleteUpit;
    mDeleteKomanda.Connection = pKonekcija.DajKonekciju();

    mUpdateKomanda = new SqlCommand();
    mUpdateKomanda.CommandText = UpdateUpit;
    mUpdateKomanda.Connection = pKonekcija.DajKonekciju();

    pAdapter = new SqlDataAdapter();
    pAdapter.SelectCommand = mSelectKomanda;
    pAdapter.InsertCommand = mInsertKomanda;
    pAdapter.UpdateCommand = mUpdateKomanda;
    pAdapter.DeleteCommand = mDeleteKomanda;
}

private void KreirajDataset()
{
    pDataSet = new System.Data.DataSet();
    pAdapter.Fill(pDataSet, pNazivTabele);

}

private void ZatvoriAdapterDataset()
{
    pAdapter.Dispose();
    pDataSet.Dispose();
}

#endregion

#region Javne metode

public System.Data.DataSet DajPodatke(string SelectUpit)
    // izdvaja podatke u odnosu na dat selectupit
{
    KreirajAdapter(SelectUpit, "", "", "");
    KreirajDataset();
    return pDataSet;
}

public int DajBrojSlogova()
{
    int BrojSlogova = pDataSet.Tables[0].Rows.Count;
    return BrojSlogova;
}

public bool IzvrsiAzuriranje(string Upit)
    // izvrzava azuriranje unos/brisanje/izmena u odnosu na dati i upit
{

    //
    bool uspeh = false;
   SqlConnection mKonekcija;
   SqlCommand Komanda;
   SqlTransaction mTransakcija = null;
    try
    {
        mKonekcija = pKonekcija.DajKonekciju();
        // aktivan kod

        // povezivanje
        Komanda = new SqlCommand();
```

```
            Komanda.Connection = mKonekcija;
            Komanda = mKonekcija.CreateCommand();
            // pokretanje
            // NE TREBA OPEN JER DOBIJAMO OTVORENU KONEKCIJU KROZ KONSTRUKTOR
            // mKonekcija.Open();
            mTransakcija = mKonekcija.BeginTransaction();
            Komanda.Transaction = mTransakcija;
            Komanda.CommandText = Upit;
            Komanda.ExecuteNonQuery();
            mTransakcija.Commit();
            uspeh = true;
         }
         catch
         {
            mTransakcija.Rollback();
            uspeh = false;
         }
         return uspeh;
      }
      #endregion

   }
}
```

NAPOMENE:
Prethodni kod ilustruje primenu objekta tipa transakcije. Naravno, transakcije se koriste kod skupa sql naredbi koje treba izvršiti paketno, a u ovom slučaju je dat primer korišćenja transakcije samo radi ilustracije načina korišćenja odgovarajuće klase i njenih metoda.

Izvorni kod navedenih klasa mogao bi se transformisati radi korišćenja „Stored procedura" za ažuriranje podataka.

PRIMER:

```
using (SqlConnection con = new SqlConnection(Konekcija))
{
        using (SqlCommand cmd = new SqlCommand("StoredProcedura_Dodaj_Kontakt", con))
        {
                cmd.CommandType = CommandType.StoredProcedure;
                cmd.Parameters.Add("@FirstName", SqlDbType.VarChar).Value = txtFirstName.Text;
                cmd.Parameters.Add("@LastName", SqlDbType.VarChar).Value = txtLastName.Text;
                con.Open();
                cmd.ExecuteNonQuery();
        }
}
```

9.2.4. KLASE ZA RAD SA ORACLE BAZOM PODATAKA

Fajl sa gotovim bibliotekama klasa "Oracle.DataAccess.dll" sadrži [1]:

- The Oracle.DataAccess.Client projekat sadrži ODP.NET classes and enumerations.
- The Oracle.DataAccess.Types projekat sadrzi the Oracle Data Provider for .NET Types (ODP.NET Types).

Oracle.DataAccess.Client klase - objasnjenje sa izvora [1] u originalu (na engleskom jeziku):

Class	**Description**
OracleCommand Class	An OracleCommand object represents a SQL command, a stored procedure or function, or a table name
OracleCommandBuilder Class	An OracleCommandBuilder object provides automatic SQL generation for the OracleDataAdapter when updates are made to the database
OracleConnection Class	An OracleConnection object represents a connection to an Oracle database
OracleDataAdapter Class	An OracleDataAdapter object represents a data provider object that communicates with the DataSet
OracleDataReader Class	An OracleDataReader object represents a forward-only, read-only, in-memory result set

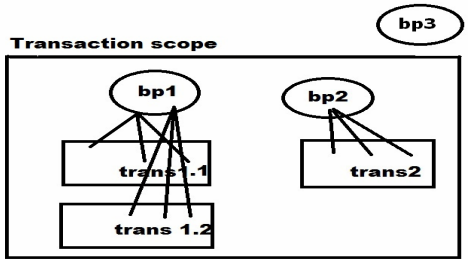
Class	Description
OracleError Class	The OracleError object represents an error reported by an Oracle database
OracleErrorCollection Class	An OracleErrorCollection object represents a collection of OracleErrors
OracleException Class	The OracleException object represents an exception that is thrown when Oracle Data Provider for .NET encounters an error
OracleFailoverEventArgs Class	The OracleFailoverEventArgs object provides event data for the OracleConnection.Failover event
OracleFailoverEventHandler Delegate	The OracleFailoverEventHandler delegate represents the signature of the method that handles theOracleConnection.Failover event
OracleGlobalization Class	The OracleGlobalization class is used to obtain and set the Oracle globalization settings of the session, thread, and local computer (read-only)
OracleInfoMessageEventHandler Delegate	The OracleInfoMessageEventHandler delegate represents the signature of the method that handles theOracleConnection.InfoMessage event
OracleInfoMessageEventArgs Class	The OracleInfoMessageEventArgs object provides event data for the OracleConnection.InfoMessage event
OracleParameter Class	An OracleParameter object represents a parameter for an OracleCommand
OracleParameterCollection Class	An OracleParameterCollection object represents a collection of OracleParameters
OracleRowUpdatedEventArgs Class	The OracleRowUpdatedEventArgs object provides event data for the OracleDataAdapter.RowUpdated event
OracleRowUpdatedEventHandler Delegate	The OracleRowUpdatedEventHandler delegate represents the signature of the method that handles theOracleDataAdapter.RowUpdated event
OracleRowUpdatingEventArgs Class	The OracleRowUpdatingEventArgs object provides event data for the OracleDataAdapter.RowUpdating event
OracleRowUpdatingEventHandler Delegate	The OracleRowUpdatingEventHandler delegate represents the signature of the method that handles theOracleDataAdapter.RowUpdating event
OracleTransaction Class	An OracleTransaction object represents a local transaction
OracleXmlQueryProperties Class	An OracleXmlQueryProperties object represents the XML properties used by the OracleCommand class when theXmlCommandType property is Query
OracleXmlSaveProperties Class	An OracleXmlSaveProperties object represents the XML properties used by the OracleCommand class when theXmlCommandType property is Insert, Update, or Delete

Na osnovu navedenih gotovih klasa, može se po analogiji sa već ranije kreiranim bibliotekama klasa za rad sa MS Access i MS SQL Server bazom podataka napraviti biblioteka klasa koja bi olašala rad sa gotovim klasama za rad sa ORACLE bazom podataka.

LITERATURA:
[1] http://docs.oracle.com/html/B14164_01/intro002.htm

9.2.5. KLASE ZA RAD SA TRANSAKCIJAMA

U okviru rada sa transakcijama razlikujemo:
1. Rad sa transakcijama nad jednom tabelom (klase za rad sa transakcijama)
2. Rad sa transakcijama nad više tabele, odnosno rad sa skupom povezanih transakcija nad više tabela, koje mogu biti distribuirane (klase za obradu "Transaction scope")



Sa prethodne slike se vidi da u okviru jedne baze podataka paralelno se može izvršavati više transakcija, a takođe se transakcije mogu vršiti na više različitih baza podataka istovremeno. Udruživanje raznih transakcija u jednu celinu vrši objekat klase Transaction Scope.

9.2.5.1. Korišćenje gotovih klasa za rad sa transakcijama

U nastavku je dat primer programskog koda za rad sa transakcijama, gde se koriste gotove industrijske klase:

```
// preuzimanje podataka sa korisnickog interfejsa
        // i odgovarajuce konverzije

        string BrojIndeksa = txtBrojIndeksa.Text;
        string Prezime = txtPrezime.Text;
        string Ime = txtIme.Text;
        string ptt = DajPTT(cmbMesto.Text);
        string Adresa = txtAdresa.Text;
        string Telefon = txtTelefon.Text;
        string Zaposlen = cmbZaposlen.Text;
        string DatumUpisa = "#" + txtMesec.Text + "/" + txtDan.Text + "/" + txtGodina.Text + "#";
        string SifraSmera = DajSifruSmera(cmbSmer.Text);
        string NivoStudija = cmbNivoStudija.Text;
        string StatusStudija = cmbStatusStudija.Text;
        string GodinaStudija = cmbGodinaStudija.Text;

        // IZVRSAVANJE TRANSAKCIJA

        // ucitavanje putanje do baze
        clsTXTfajl objTxtFajl = new clsTXTfajl();
        string putanjaAccessBaze = objTxtFajl.ProcitajFajl(Application.StartupPath +
"\\putanja\\PutanjaBazeOsnovna.txt");

        // otvaranje konekcije
        OleDbConnection objAccessConn1 = new OleDbConnection();
        objAccessConn1.ConnectionString = "Jet OLEDB:Global Partial Bulk Ops=2;Jet OLEDB:Registry Path=;Jet
OLEDB:Database Locking Mode=1;Data Source='" + putanjaAccessBaze + "';Mode=Share Deny None;Jet
OLEDB:Engine Type=5;Provider='Microsoft.Jet.OLEDB.4.0';Jet OLEDB:System database=;Jet OLEDB:SFP=False;persist
security info=False;Extended Properties=;Jet OLEDB:Compact Without Replica Repair=False;Jet OLEDB:Encrypt
Database=False;Jet OLEDB:Create System Database=False;Jet OLEDB:Don't Copy Locale on Compact=False;User
ID=Admin;Jet OLEDB:Global Bulk Transactions=1";
        objAccessConn1.Open();

        // kreiranje transakcije

        OleDbTransaction objAccessTransakcija = objAccessConn1.BeginTransaction();


        // izvrsavanje transakcije
        string stringUpita1 = "INSERT INTO Student(BrojIndeksa, Prezime, Ime, ptt,  Adresa, Telefon, Zaposlen,
DatumUpisa, SifraSmera, NivoStudija, StatusStudija, GodinaStudija) VALUES ('" + BrojIndeksa + "','" + Prezime + "','"
+ Ime + "'," + ptt + ",'" + Adresa + "','" + Telefon + "','" + Zaposlen + "'," + DatumUpisa + ",'" + SifraSmera + "','" +
NivoStudija + "','" + StatusStudija + "'," + GodinaStudija + ")";
        OleDbCommand cmdAccess1 = new OleDbCommand(stringUpita1, objAccessConn1);

        cmdAccess1.Transaction = objAccessTransakcija;

        cmdAccess1.ExecuteNonQuery();

        objAccessTransakcija.Commit();
```

9.2.5.2. Korišćenje gotovih klasa za rad sa distribuiranim transakcijama

U nastavku je dat primer programskog koda za rad sa distribuiranim transakcijama, odnosno sa skupom transakcija nad više različitih baza podataka koje su udružene u okviru zajedničkog "Transaction scope".

```
// preuzimanje podataka sa korisnickog interfejsa
        // i odgovarajuce konverzije

        string BrojIndeksa = txtBrojIndeksa.Text;
```

```
        string Prezime = txtPrezime.Text;
        string Ime = txtIme.Text;
        string ptt = DajPTT(cmbMesto.Text);
        string Adresa = txtAdresa.Text;
        string Telefon = txtTelefon.Text;
        string Zaposlen = cmbZaposlen.Text;
        string DatumUpisa = "#" + txtMesec.Text + "/" + txtDan.Text + "/" + txtGodina.Text + "#";
        string SifraSmera = DajSifruSmera(cmbSmer.Text);
        string NivoStudija = cmbNivoStudija.Text;
        string StatusStudija = cmbStatusStudija.Text;
        string GodinaStudija = cmbGodinaStudija.Text;

        // IZVRSAVANJE DISTRIBUIRANIH TRANSAKCIJA

        TransactionScope tsInstance = new TransactionScope();
        using (tsInstance)
        {

            // ********************** OSNOVNA BAZA *************************
            // *************************************************************

            // ucitavanje putanje do baze
            clsTXTfajl objTxtFajl1 = new clsTXTfajl();
            string putanjaAccessBaze1 = objTxtFajl1.ProcitajFajl(Application.StartupPath +
"\\putanja\\PutanjaBazeOsnovna.txt");

            // otvaranje konekcije
            OleDbConnection objAccessConn1 = new OleDbConnection();
            objAccessConn1.ConnectionString = "Jet OLEDB:Global Partial Bulk Ops=2;Jet OLEDB:Registry Path=;Jet
OLEDB:Database Locking Mode=1;Data Source='" + putanjaAccessBaze1 + "';Mode=Share Deny None;Jet
OLEDB:Engine Type=5;Provider='Microsoft.Jet.OLEDB.4.0';Jet OLEDB:System database=;Jet OLEDB:SFP=False;persist
security info=False;Extended Properties=;Jet OLEDB:Compact Without Replica Repair=False;Jet OLEDB:Encrypt
Database=False;Jet OLEDB:Create System Database=False;Jet OLEDB:Don't Copy Locale on Compact=False;User
ID=Admin;Jet OLEDB:Global Bulk Transactions=1";
            objAccessConn1.Open();

            // kreiranje transakcije

            OleDbTransaction objAccessTransakcija1 = objAccessConn1.BeginTransaction();


            // izvrsavanje transakcije
            string stringUpita1 = "INSERT INTO Student(BrojIndeksa, Prezime, Ime, ptt,  Adresa, Telefon, Zaposlen,
DatumUpisa, SifraSmera, NivoStudija, StatusStudija, GodinaStudija) VALUES ('" + BrojIndeksa + "','" + Prezime + "','"
+ Ime + "'," + ptt + ",'" + Adresa + "','" + Telefon + "','" + Zaposlen + "'," + DatumUpisa + ",'" + SifraSmera + "','" +
NivoStudija + "','" + StatusStudija + "'," + GodinaStudija + ")";
            OleDbCommand cmdAccess1 = new OleDbCommand(stringUpita1, objAccessConn1);

            cmdAccess1.Transaction = objAccessTransakcija1;

            cmdAccess1.ExecuteNonQuery();

            objAccessTransakcija1.Commit();

            // ********************** OSNOVNA BAZA *************************
            // *************************************************************

            // ucitavanje putanje do baze
            clsTXTfajl objTxtFajl2 = new clsTXTfajl();
            string putanjaAccessBaze2 = objTxtFajl2.ProcitajFajl(Application.StartupPath +
"\\putanja\\PutanjaBazeOsnovna2.txt");

            // otvaranje konekcije
            OleDbConnection objAccessConn2 = new OleDbConnection();
            objAccessConn2.ConnectionString = "Jet OLEDB:Global Partial Bulk Ops=2;Jet OLEDB:Registry Path=;Jet
OLEDB:Database Locking Mode=1;Data Source='" + putanjaAccessBaze2 + "';Mode=Share Deny None;Jet
OLEDB:Engine Type=5;Provider='Microsoft.Jet.OLEDB.4.0';Jet OLEDB:System database=;Jet OLEDB:SFP=False;persist
security info=False;Extended Properties=;Jet OLEDB:Compact Without Replica Repair=False;Jet OLEDB:Encrypt
Database=False;Jet OLEDB:Create System Database=False;Jet OLEDB:Don't Copy Locale on Compact=False;User
ID=Admin;Jet OLEDB:Global Bulk Transactions=1";
            objAccessConn2.Open();
```

```
            // kreiranje transakcije

            OleDbTransaction objAccessTransakcija2 = objAccessConn2.BeginTransaction();


            // izvrsavanje transakcije
            string stringUpita2 = "INSERT INTO Student(BrojIndeksa, Prezime, Ime, ptt,  Adresa, Telefon, Zaposlen,
DatumUpisa, SifraSmera, NivoStudija, StatusStudija, GodinaStudija) VALUES ('" + BrojIndeksa + "','" + Prezime + "','"
+ Ime + "'," + ptt + ",'" + Adresa + "','" + Telefon + "','" + Zaposlen + "'," + DatumUpisa + ",'" + SifraSmera + "','" +
NivoStudija + "','" + StatusStudija + "'," + GodinaStudija + ")";
            OleDbCommand cmdAccess2 = new OleDbCommand(stringUpita2, objAccessConn2);

            cmdAccess2.Transaction = objAccessTransakcija2;

            cmdAccess2.ExecuteNonQuery();

            objAccessTransakcija2.Commit();

            // --------------------POTVRDA SVIH TRANSAKCIJA--------------------------
            // ************************************************************************

            tsInstance.Complete();
```

9.2.5.3. Kreiranje sopstvene klase za rad sa transakcijama

U nastavku je dat programski kod klase kreirane da pojednostavi rad sa transakcijama nad jednom bazom podataka.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
//
using System.Data.OleDb;

namespace AccessDBUtils
{
   public class clsTransakcija
   {
      #region Atributi
      OleDbConnection pKonekcija = new OleDbConnection();
      OleDbCommand komanda = new OleDbCommand();
      OleDbTransaction transakcija;
      string[] listaupita = new string[5];
      int RedniBroj = 0;
      #endregion
      #region Javne metode

      public void DodajUpit(string noviUpit)
      {
         listaupita.SetValue(noviUpit, RedniBroj);
         RedniBroj++;
      }

      public bool IzvrsiTransakciju()
      {
         // podrazumeva se da je konekcija otvorena i kao takva prosledjena
         //pKonekcija.Open();
         bool uspeh = false;
         try
         {
            komanda = pKonekcija.CreateCommand();
            transakcija = pKonekcija.BeginTransaction();
            komanda.Transaction = transakcija;
            for (int i = 0; i < RedniBroj; i++)
            {
               komanda.CommandText = listaupita.GetValue(i).ToString();
               komanda.ExecuteNonQuery();
            }
            transakcija.Commit();
            //pKonekcija.Close();
```

```
            uspeh = true;

        }
        catch
        {
            uspeh = false;
        }

        return uspeh;
    }
    #endregion

  }
}
```

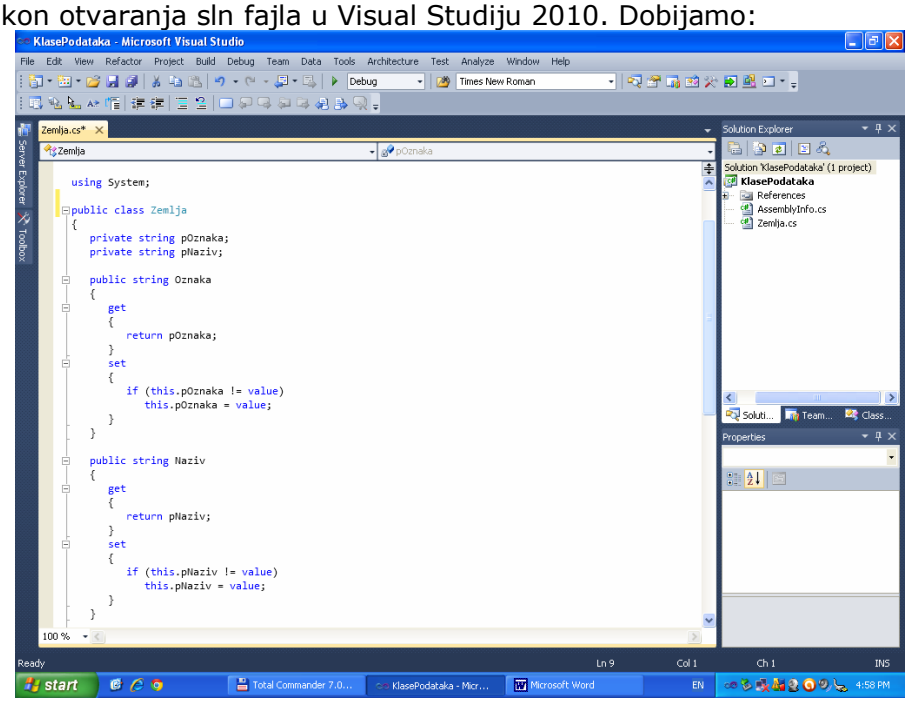
9.3. BIBLIOTEKA KLASA PODATAKA

U okviru odeljka 6.4.3. generisan je programski kod klasa iz CASE alata. U ovom odeljku opisan je postupak uređivanja kreiranog koda klasa, radi korišćenja u daljem programiranju. Naime, koncepcija vežbi je takva da za svaku tabelu iz baze podataka može biti 3 klase u biblioteci klasa podataka:

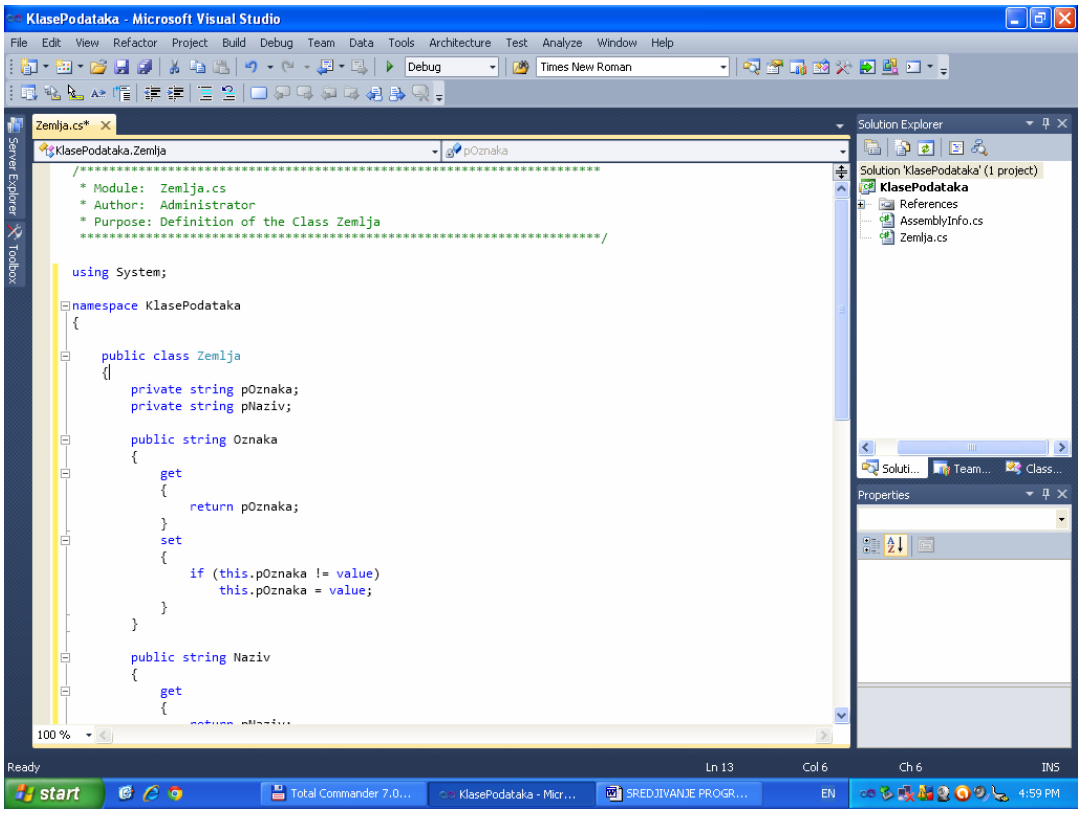
1. Klasa tipa "pojedinac" – ima samo set i get metode (dobija se generisanjem na osnovu dijagama klasa iz CASE alata ili u okviru Entity Framework-a)
2. Klasa tipa "lista" - uređena lista objekata klasa pojedinac
3. Klasa tipa "DB" – Klasa koja sadrži SQL upite za rad sa tabelom baze podataka na koju se odnosi.

Ove klase se prave nezavisno od buduće primene za konkretnu aplikaciju. Opšti princip je kreiranje biblioteke klasa podataka koja može biti korišćena za razne aplikacije, a olakšava programiranje korisničkog interfejsa podelom odgovornosti na slojeve klasa, prema principima višeslojnog programiranja.

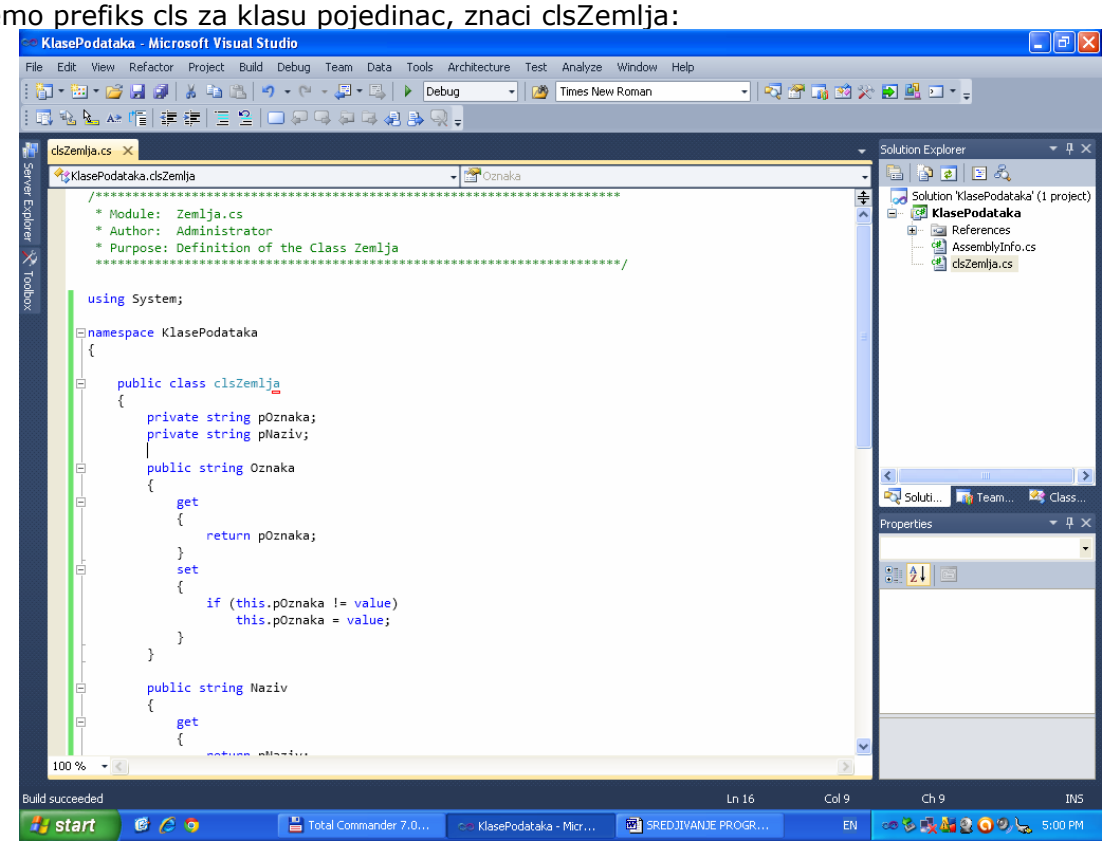
Postupak kreiranja biblioteke klasa podataka (ilustrujemo na jednostavnom primeru klase Zemlja na osnovu tabele Zemlja) započinjemo pokretanjem projekta (*.sln fajl) koji se ucitava u Visual Studio.NET 2010.Generisan je kod ranije generacije C# projekta, pa se pokrece konverzija, nakon otvaranja sln fajla u Visual Studiju 2010. Dobijamo:



Dodajemo namespace KlasePodataka - pripadnost klase projektu koji se zove "KlasePodataka" (konverzija to nije dodala, a nije postojalo u ranijim verzijama, a sada je potrebno).



dodajemo prefiks cls za klasu pojedinac, znaci clsZemlja:



Kreiramo klasu clsArrayListaZemalja

```
using System;
using System.Collections.Generic;
using System.Text;
//
using System.Collections;

namespace KlasePodataka
{
   public class clsArrayListaZemalja
   {
      // atribut
      ArrayList listaZemalja;

      // konstruktor
      public clsArrayListaZemalja()
      {
         listaZemalja = new ArrayList();
      }

      // metode

      public void ObrisiArrayListuZemalja()
      {
         listaZemalja.Clear();
      }

      public int DajBrojElemenataListeZemalja()
      {
         return listaZemalja.Count;
      }

      public void DodajElementArrayListeZemalja(clsZemlja objNovaZemlja)
      {
         listaZemalja.Add(objNovaZemlja);
      }

      public void ObrisiElementArrayListeZemalja(clsZemlja objZemljaZaBrisanje)
      {
         listaZemalja.Remove(objZemljaZaBrisanje);
      }

      public void ObrisiElementArrayListeZemaljaSaPozicije(int pozicija)
      {
         listaZemalja.RemoveAt(pozicija);
      }

      public void UmetniElementArrayListeZemaljaNaPoziciju(clsZemlja objNovaZemlja, int pozicija)
      {
         listaZemalja.Insert(pozicija, objNovaZemlja);
      }

      public void ObrisiNizElemenataSaPozicija(int pocetnapozicija, int duzina)
      {
         listaZemalja.RemoveRange(pocetnapozicija, duzina);

      }


   }
}
```

Kreiramo klasu clsListaZemalja

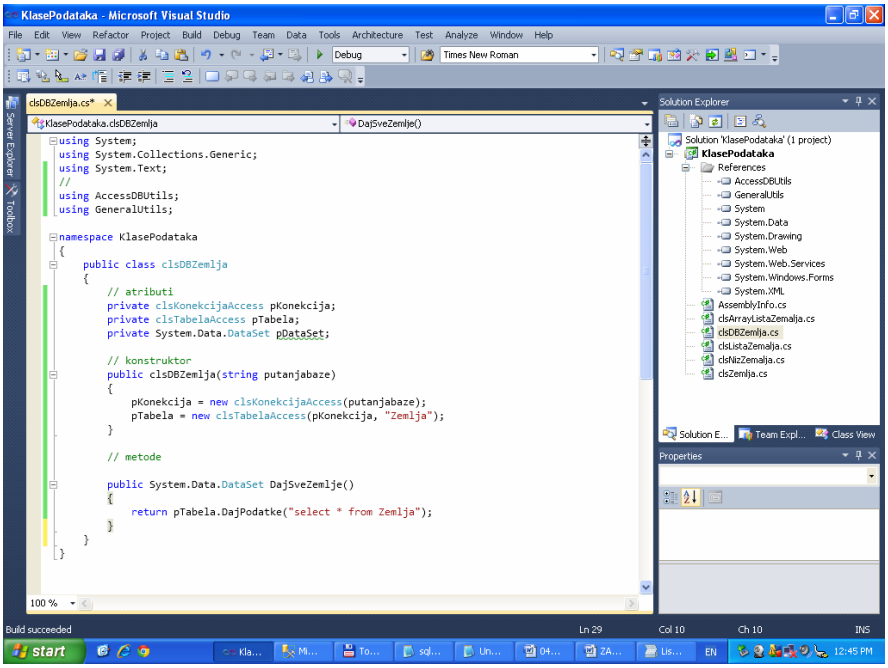
```
using System;
using System.Collections.Generic;
using System.Text;
//
using System.Collections;
```

```
namespace KlasePodataka
{
   public class clsListaZemalja
   {
      //NAPOMENA:
      // List<tip> moze imati elemente samo <tip> tipa
      //


      //atributi
      List<clsZemlja> objListaZemalja;

      //konstruktor
      public clsListaZemalja()
      {
         objListaZemalja= new List<clsZemlja>();
      }

      // metode
      public void DodajElementListe(clsZemlja objNovaZemlja)
      {
         objListaZemalja.Add(objNovaZemlja);
      }

      // dodajemo dalje metode po zelji....
   }
}
```

Kreiramo klasu clsNizZemalja, radi upoređivanja sa klasom clsListaZemalja.

```
using System;
using System.Collections.Generic;
using System.Text;

namespace KlasePodataka
{
   public class clsNizZemalja
   {
      // NAPOMENA:
      // niz zemalja ima ogranicen broj elemenata koji se moraju unapred odrediti
      //

      // atributi
      clsZemlja[] objNizZemalja;
      int pBrojElemenataNiza; // ukupno 100, ali oznake idu od 0-99

      // konstruktor
      public clsNizZemalja(int BrojElemenataNiza)
      {
         pBrojElemenataNiza = BrojElemenataNiza;
         objNizZemalja = new clsZemlja[pBrojElemenataNiza];
      }
      public void DodajZemlju(int pozicija, clsZemlja objNovaZemlja)
      {
         if (pozicija < pBrojElemenataNiza)
         {
            objNizZemalja[pozicija] = objNovaZemlja;
         }
      }

   }
}
```

Kreiramo klasu clsDBZemlja




```
using System;
using System.Collections.Generic;
using System.Text;
//
using AccessDBUtils;
using GeneralUtils;

namespace KlasePodataka
{
    public class clsDBZemlja
    {
        // atributi
        private clsKonekcijaAccess pKonekcija;
        private clsTabelaAccess pTabela;
        private System.Data.DataSet pDataSet;

        // konstruktor
        public clsDBZemlja(string putanjabaze)
        {
            pKonekcija = new clsKonekcijaAccess(putanjabaze);
            pTabela = new clsTabelaAccess(pKonekcija, "Zemlja");
        }

        // metode

        public System.Data.DataSet DajSveZemlje()
        {
            return pTabela.DajPodatke("select * from Zemlja");
        }
    }
}
```

Posebno možemo obratiti pažnju na klase podataka koje kao svoje atribute imaju objekte drugih klasa (videti generisane klase iz odeljka 6.4.3):

ODNOS ASOCIJACIJE sa dijagrama klasa

```
public class StavkeRacuna
{
  public TipRobe tipRobe; <--- atribut se zove tipRobe i predstavlja objekat klase TipRobe

  private float pKolicina;
  private float pCenaStavke;
```

ODNOS KOMPOZICIJE sa dijagrama klasa

```
public class Racun
{
  public System.Collections.ArrayList stavkaRacuna;  <---- atribut je uređena lista objekata klase stavkaRacuna
```

Naravno, u ovim slučajevima možemo izvršiti korekcije:

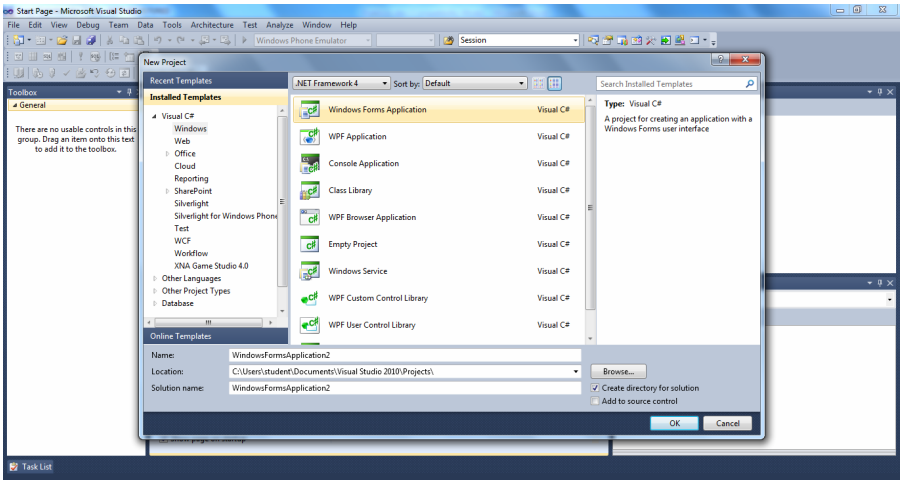
1. Preimenovati naziv atributa
2. Koristiti klasu UredjenaListaStavkiRacuna kao tip podatka za atribut, umesto direktnog navodjenja sistemske kolekcije ArrayList

10. IMPLEMENTACIJA WINDOWS APLIKACIJE u Visual Studio .NET

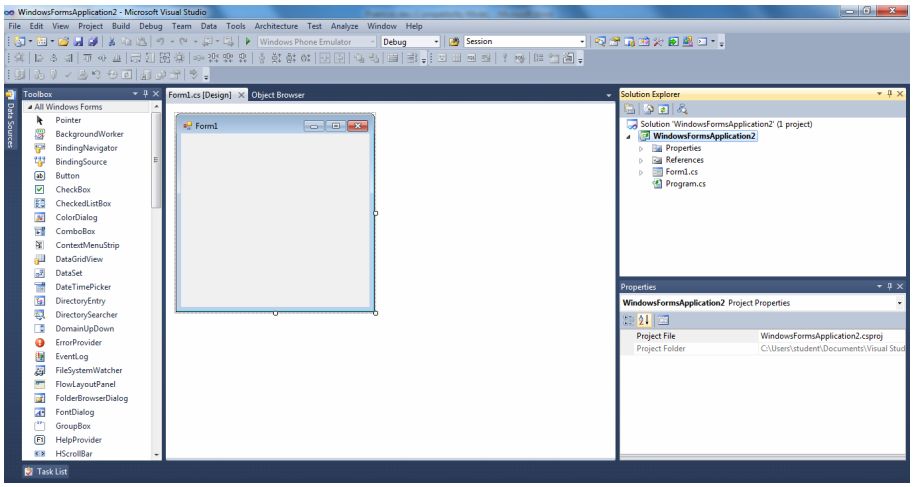
10.1. Radno okruženje

U nastavku će biti prikazan način izrade windows aplikacije u okviru Visual Studio .NET 2010 razvojnog okruženja.

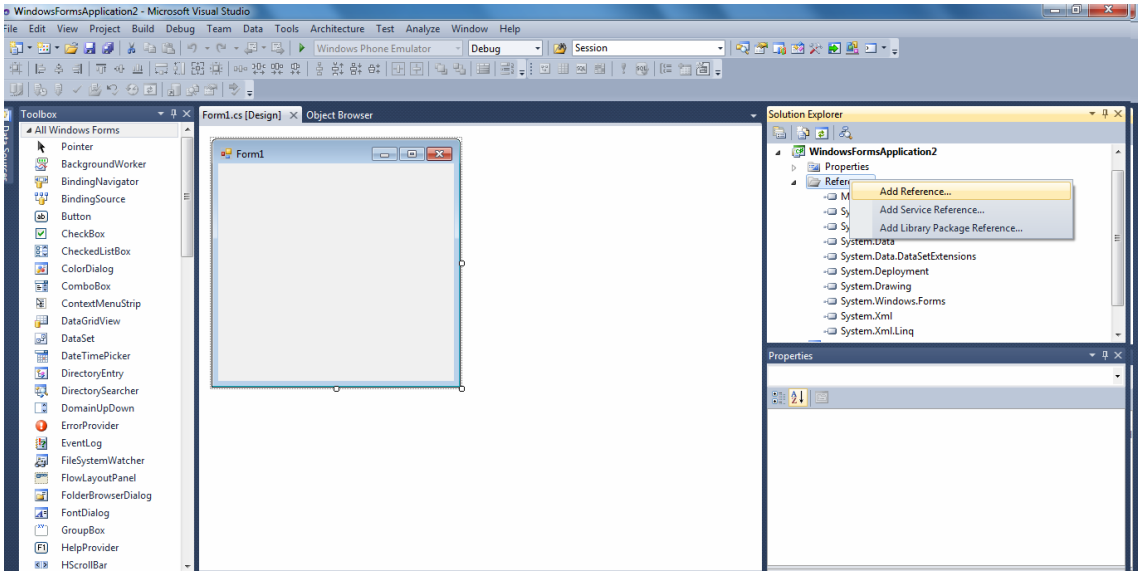
Nakon startovanja razvojnog okruženja dobijamo dijalog prozor za izbor zeljenog tipa aplikacije:



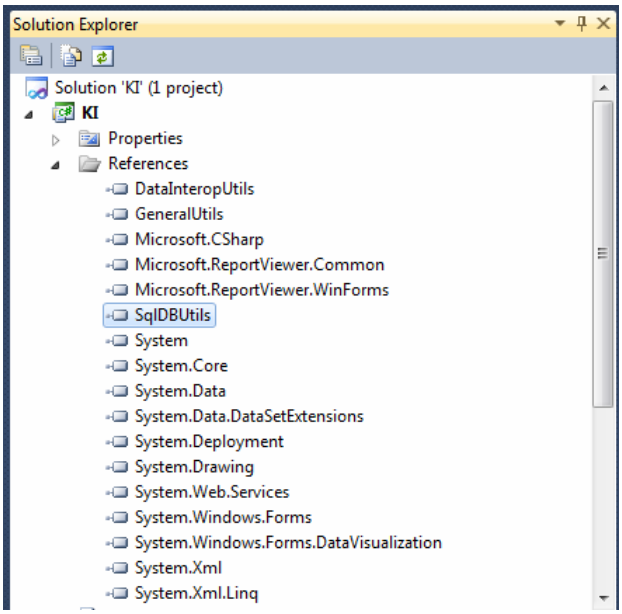
Nakon izbora tipa aplikacije i snimanja početnih fajlova projekta aplikacije na željenoj lokaciji, dobijamo prostor za dizajn i programiranje ekranskih formi. Sa palete biramo elemente korisnickog interfejsa i kreiramo vizuelni dizajn formi.



U odeljku Solution – References koristimo desni taster i opciju „Add reference” kako bismo dodali referencu na biblioteke klasa koje smo prethodno kreirali:

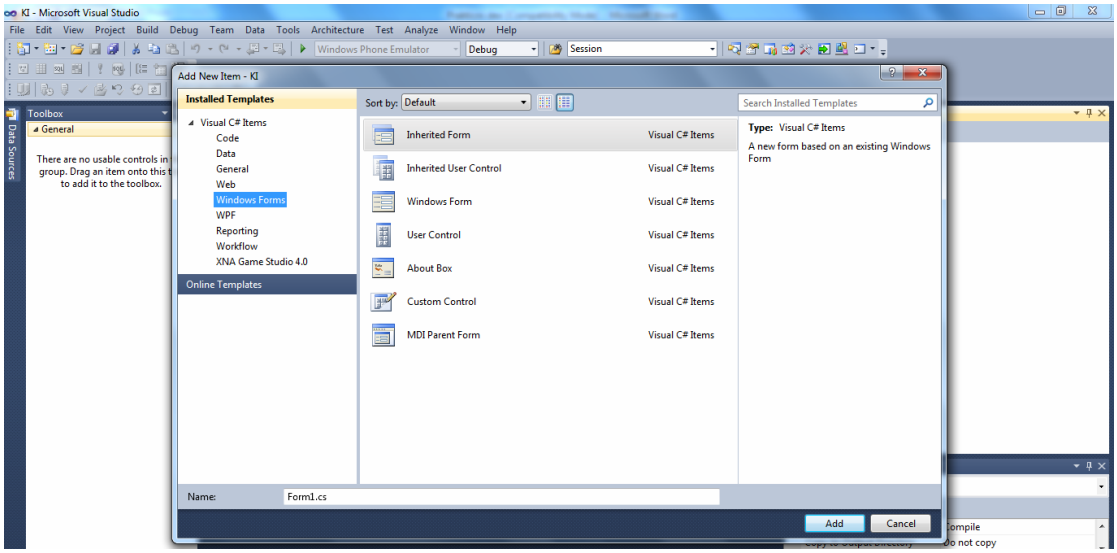


Dodajemo sve prethodno kreirane dll fajlove (kompajlirane biblioteke klasa) i navedene biblioteke klasa se javljaju na spisku referenci:

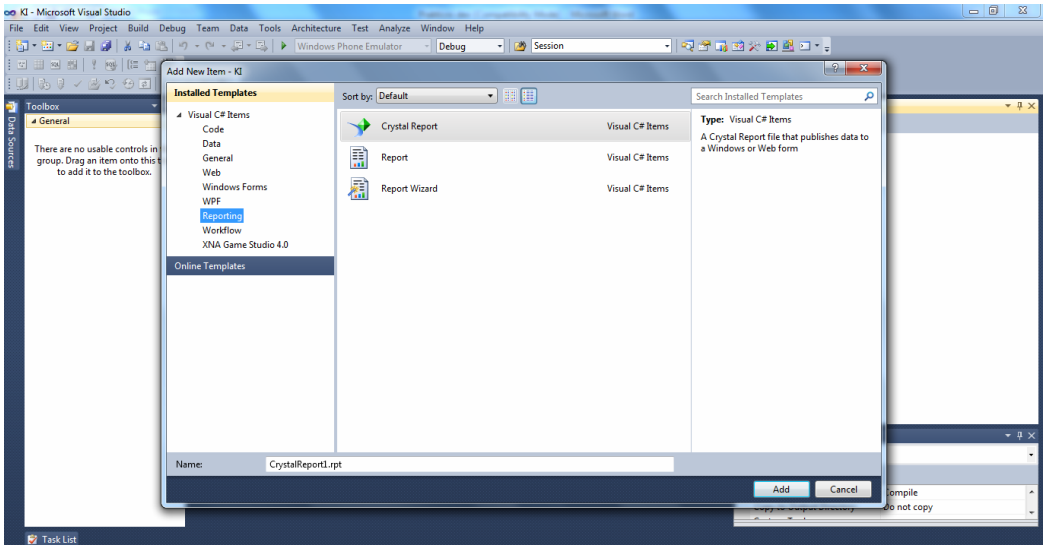


Radi dodavanja elemenata projektu koristimo opciju sa glavnog menija razvojnog okruzenja Project – Add New Item:

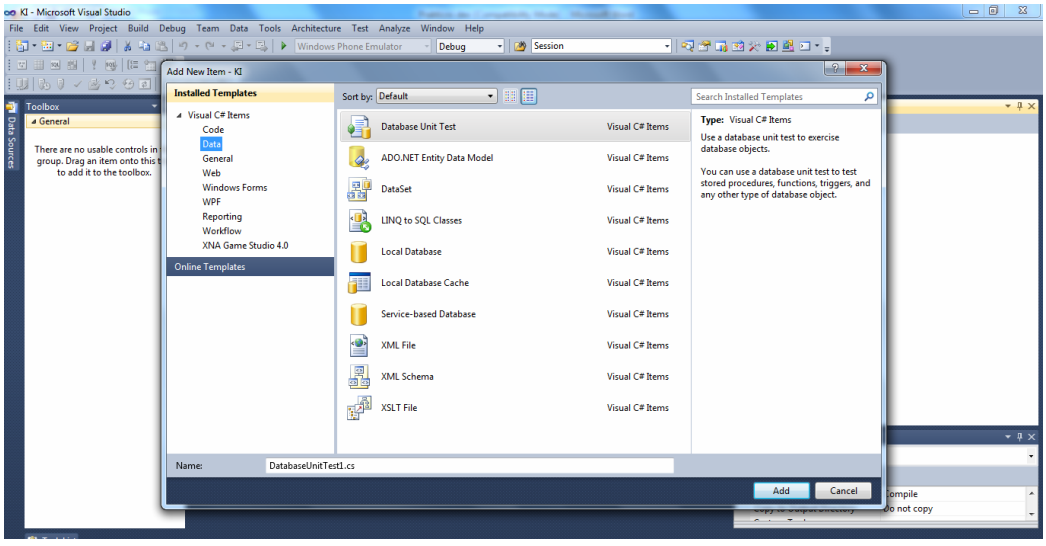
Za Windows forme:



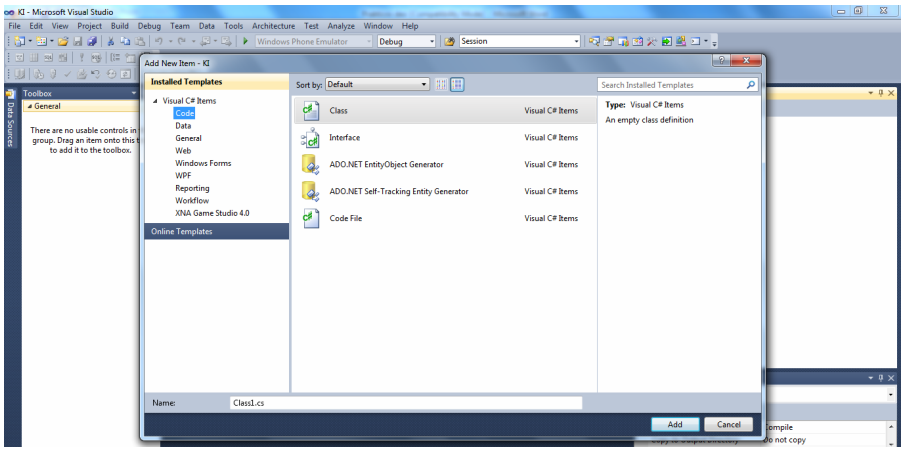
Za Izvestaje:



Za podatke:

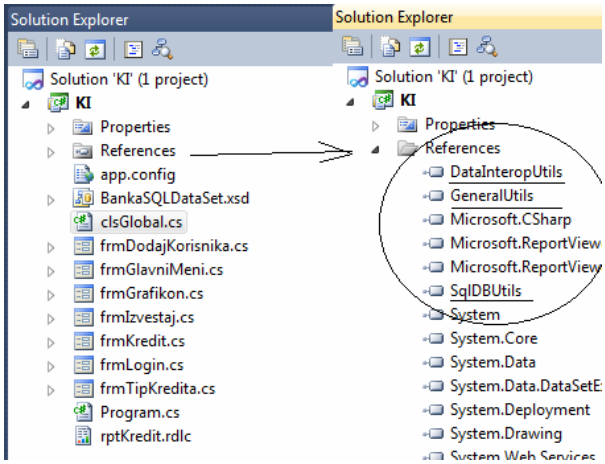


Za programski kod:



10.2. Fajlovi u strukturi aplikacije

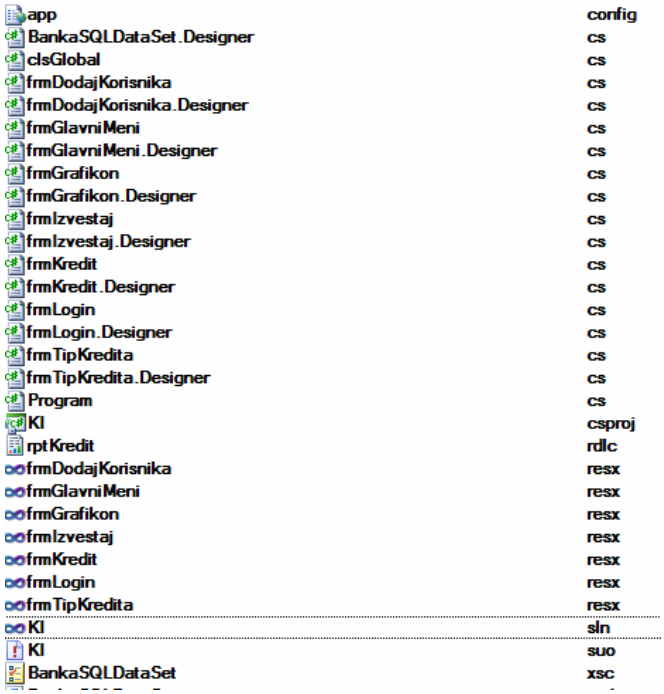
Konacni skup elemenata projekta koji čine resenje korisnickog interfejsa se sastoji iz:



Gde je:

clsGlobal.cs	Klasa sa globalnim promenljivima i procedurama, prvenstveno zajednickim zbog otvaranja konekcije jednom na nivou cele aplikacije
FrmDodajKorisnika.cs	Forma za dodavanje korisnika sistema
FrmGlavniMeni.cs	Forma sa glavnim menijem aplikacije
FrmGrafikon.cs	Forma sa grafikonom
frmIzvestaj.cs	Forma za rad sa izvestajima
frmKredit.cs	Forma za rad sa Kreditima – osnovna forma za demonostraciju rada sa svim osnovnim funkcijama
FrmLogin.cs	Forma za prijavu korisnika na sistem
Frm TipKredita.cs	Forma za rad sa pomocnim (sifarnickim tabelama) – npr. Tip Kredita
Program.cs	Fajl gde se definise pocetna forma za pokretanje aplikacije
rptKredit.rdlc	Fajl izvestaja

Odnosno od fajlova:



Gde su:

App.config	Konfiguracioni fajl koji prvenstveno sadrzi string konekcije, kreira se i azurira prilikom direktnog konektovanja iz korisnickog interfejsa do baze podataka
BankaSQLDatased.Designer.cs	Struktura Dataseta, kada se kreira prilikom direktnog konektovanja iz korisnickog interfejsa do baze podataka
frmDodajKorisnika.cs	Programski kod funkcionalnog dela ekranske forme
frmDodajKorisnika.Designer.cs	Programski kod koji se odnosi samo na elemente korisnickog interfejsa forme
KI.csproj	Osnovni fajl celine projekta
KI.sln	Osnovni fajl resenja (solution) koji moze da sadrzi vise projekata

10.3. Globalna podešavanja i otvaranje konekcije

U okviru projekta nalazi se klasa gde su definisane globalne promenljive i procedure, „clsGlobal.cs". Programski kod klase clsGlobal.cs dat je u nastavku.

NAPOMENA: Da bismo koristili usluge biblioteka klasa iz References, moramo dodati "using" i naziv odgovarajuce biblioteke.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
// DODATO
using SqlDBUtils;
using DataInteropUtils;
using System.Windows.Forms;
```

```
namespace KI
{
   public static class clsGlobal
   {
      // GLOBALNE PROMENLJIVE
      public static clsSqlKonekcija Konekcija;
      public static string KorisnickoIme;
      public static string StatusKorisnika;

      // GLOBALNE PROCEDURE
      public static bool OtvoriKonekciju()
        // otvara konekciju jednom na nivou cele aplikacije
      // ovde se procedura OtvoriKonekciju zove isto kao i metoda klase konekcija,sto je dozvoljeno
      {
         System.Data.DataSet dsParametri = clsXMLUtil.UcitajXMLuDataset(Application.StartupPath + "
\\Podesavanja\\PodesavanjaSQL.XML");
         //
         string pNazivSQLinstance = dsParametri.Tables[0].Rows[0].ItemArray[0].ToString();
         string pNazivBaze = dsParametri.Tables[0].Rows[0].ItemArray[1].ToString();
         string pPutanjaBaze = dsParametri.Tables[0].Rows[0].ItemArray[2].ToString();

         Konekcija = new clsSqlKonekcija(pNazivSQLinstance, pPutanjaBaze, pNazivBaze);

         bool uspeh = Konekcija.OtvoriKonekciju();
         return uspeh;
      }

      public static bool Karakter(KeyPressEventArgs e, string tipKaraktera)
        // sluzi za proveru ispravnosti tipa pritisnutih karaktera u nekom unosu
      {
         bool ispravan = false;
         switch (tipKaraktera)
         {
            case "VelikoSlovoAlfabet":
               if (e.KeyChar < 65 || e.KeyChar > 90)
               {
                  ispravan = true;
               }
               else
               {
                  ispravan = false;
               }
               break;
            case "MaloSlovoAlfabet":
               if (e.KeyChar < 97 || e.KeyChar > 122)
               {
                  ispravan = true;
               }
               else
               {
                  ispravan = false;
               }
               break;
            case "YUslovo":
               string YULat = "ČčĆćŠšĐđŽž";
               bool yuslovalat = YULat.Contains((char)e.KeyChar);
               if (yuslovalat == true)
               {
                  ispravan = true;
               }
               else
               {
                  ispravan = false;
               }
               break;
            case "cifra":
               if (e.KeyChar < 48 || e.KeyChar > 57)
               {
                  ispravan = false;
               }
```

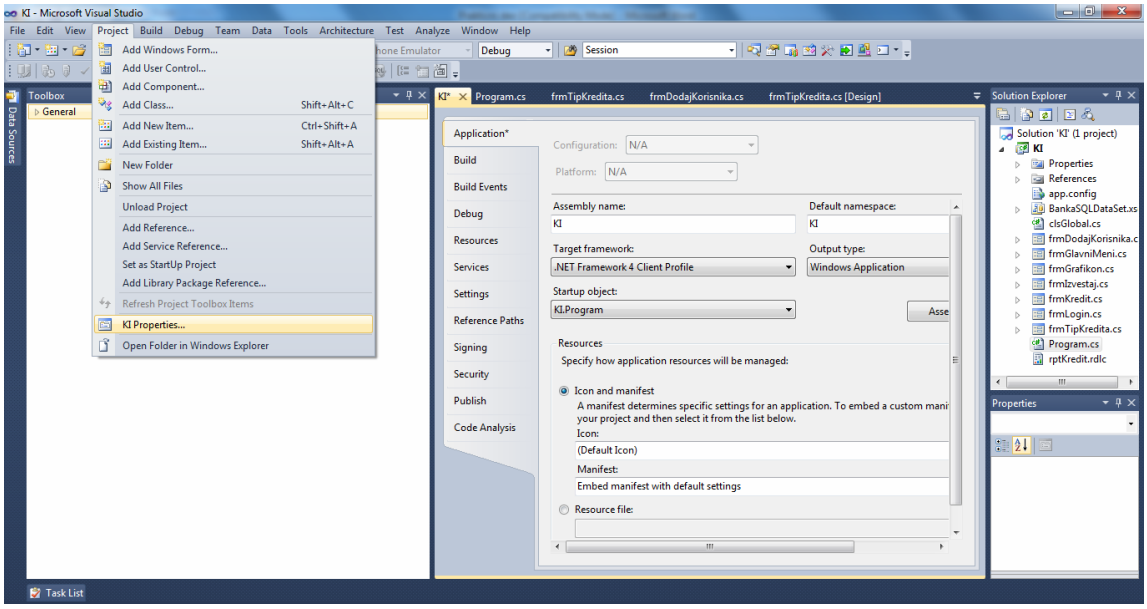
```
                else
                {
                    ispravan = true;
                }
                break;
            case "ENTER":
                if (e.KeyChar != 13)
                {
                    ispravan = false;
                }
                else
                {
                    ispravan = true;
                }
                break;
            case "BACKSPACE":
                if (e.KeyChar != 8)
                {
                    ispravan = false;
                }
                else
                {
                    ispravan = true;
                }
                break;
            case "BLANKO":
                if (e.KeyChar != 32)
                {
                    ispravan = false;
                }
                else
                {
                    ispravan = true;
                }
                break;
        }
        return ispravan;
    }
  }
}
```

10.4. POKRETANJE APLIKACIJE

Fajl Program.cs sadrži podešavanje početne forme, u našem slučaju "frmLogin".

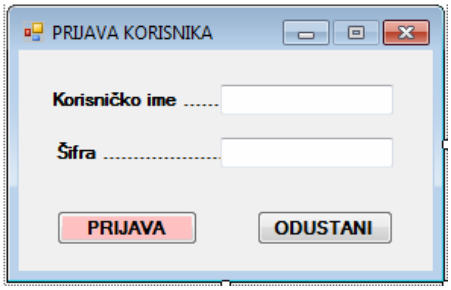
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Forms;

namespace KI
{
  static class Program
  {
    /// <summary>
    /// The main entry point for the application.
    /// </summary>
    [STAThread]
    static void Main()
    {
      Application.EnableVisualStyles();
      Application.SetCompatibleTextRenderingDefault(false);
      Application.Run(new frmLogin());
    }
  }
}
```

Podešavanje na nivou projekta se mora izvršiti tako da se za čitanje podataka o početnoj formi uzima Program.cs:

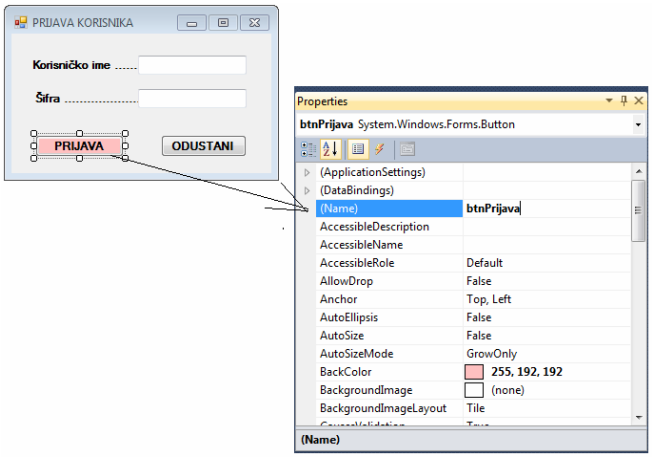


10.5. PRIJAVA NA SISTEM

Ekranska forma za prijavu sastoji se od labela, text box-ova i tastera:



NAPOMENA:
Nastojimo u dizajnu korisnickog interfejsa da i kontrole u Properties prozoru (osobina „Name") nazivamo mnemonicki, npr. Button Prijava je btnPrijava.



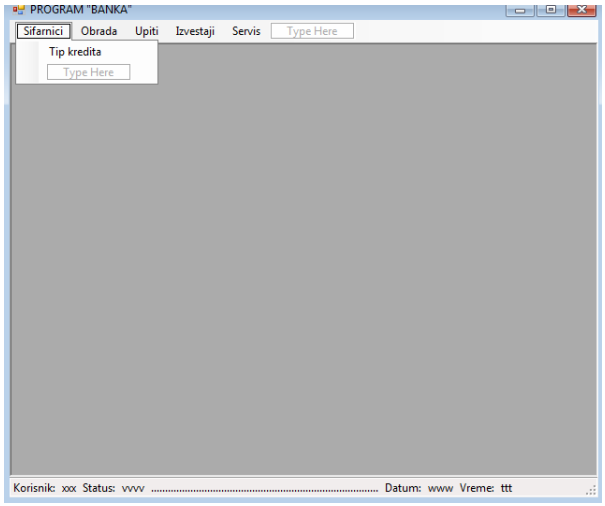
Programski kod forme za login prikazan je u nastavku. U okviru form load dogadjaja otvara se konekcija prvi i jedini put na nivou cele aplikacije. Nakon uspešne prijave na tasteru btnPrijava,

otvara se glavni meni, a čuvaju u globalnoj promenljivoj podaci o aktuelnom korisniku.

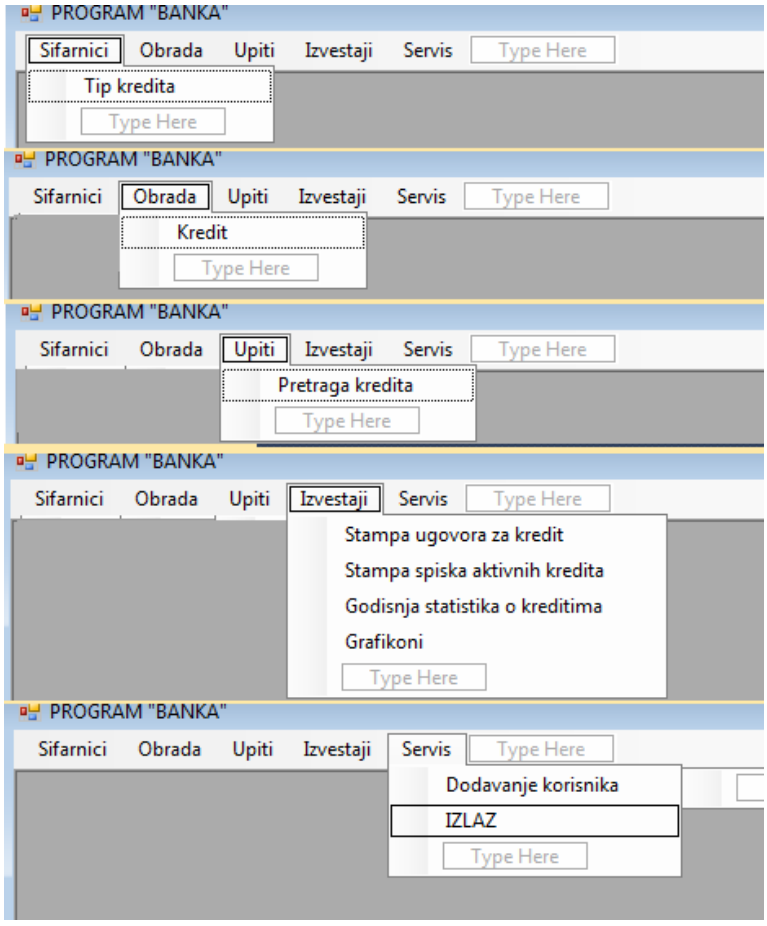
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
//
using SqlDBUtils;

namespace KI
{
   public partial class frmLogin : Form
   {
      public frmLogin()
      {
         InitializeComponent();
      }

      private void frmLogin_Load(object sender, EventArgs e)
      {
         clsGlobal.OtvoriKonekciju();
      }

      private void btnOdustani_Click(object sender, EventArgs e)
      {
         txtKorisnickoIme.Text = "";
         txtSifra.Text ="";

      }

      private void btnPrijava_Click(object sender, EventArgs e)
      {
         clsSqlTabela pTabela = new clsSqlTabela(clsGlobal.Konekcija, "Korisnik");

         // privremeno:
         System.Data.DataSet dsKorisnik = pTabela.DajPodatke("select * from Korisnik where KorisnickoIme='" +
txtKorisnickoIme.Text + "' and Sifra='" + txtSifra.Text + "'");
         if (dsKorisnik.Tables[0].Rows.Count == 0)
         {
            MessageBox.Show("Ne postoji korisnik sa datim korisnickim imenom i/ili sifrom!", "UPOZORENJE");
         }
         else
         {
            clsGlobal.KorisnickoIme = dsKorisnik.Tables[0].Rows[0].ItemArray[0].ToString();
            clsGlobal.StatusKorisnika = dsKorisnik.Tables[0].Rows[0].ItemArray[2].ToString();
            frmGlavniMeni formaGlavniMeni = new frmGlavniMeni();
            formaGlavniMeni.ShowDialog();
            this.Visible = false;
         }
      }

      private void txtSifra_KeyPress(object sender, KeyPressEventArgs e)
      {
         if (clsGlobal.Karakter(e, "ENTER"))
         {
            btnPrijava.PerformClick();
         }
      }
   }
}
```

10.6. Glavni meni

Izgled glavnog menija cini "menustrip" (gornji deo), "status strip" (donji deo) i timer kontrola. U okviru formload dogadjaja se prikazuju u podaci u statusnoj liniji, a timer kontrola se aktivira, kako bi prikazivala tacan datum i vreme u statusnoj liniji. Takođe, u skladu sa pravima korisnika, pojedine stavke glavnog menija se postavljaju vidljivim.



Struktura menija se najčešće sastoji od šifarnika (opšti podaci), obrada (podrška primitivnim poslovnim procesima), upiti (razne vrste pretraga), izveštaji (razne vrste dokumenata, statistika i grafikona) i servis (obrada korisnika sistema, backup i slično):



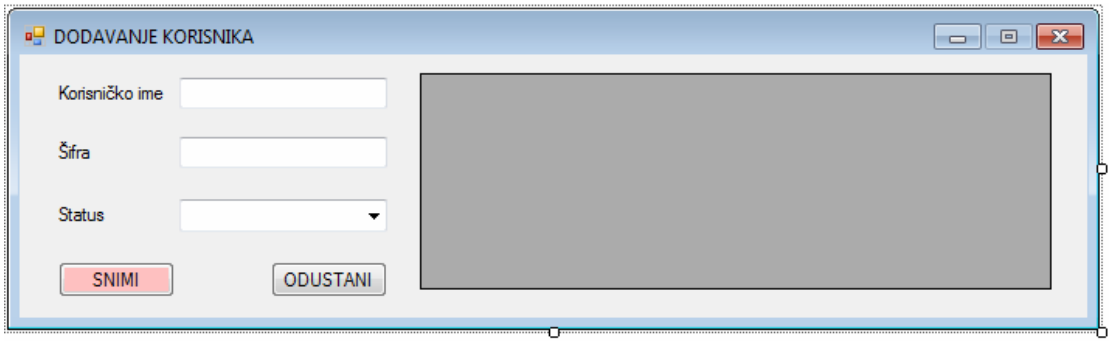
Programski kod forme glavnog menija dat je u nastavku:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace KI
{
   public partial class frmGlavniMeni : Form
   {
      public frmGlavniMeni()
      {
         InitializeComponent();
      }

      private void frmGlavniMeni_Load(object sender, EventArgs e)
      {
         tsslKorisnickoIme.Text = clsGlobal.KorisnickoIme;
         tsslStatusKorisnika.Text = clsGlobal.StatusKorisnika;
         tsslDatum.Text = DateTime.Now.Day.ToString() + "." + DateTime.Now.Month.ToString() + "." +
DateTime.Now.Year.ToString();
         timer1.Start();
         //
         //regulisanje prava prema statusima/profilima korisnika
         if (clsGlobal.StatusKorisnika.Equals ("administrator"))
         {
            sifarniciToolStripMenuItem.Visible = true;
            dodavanjeKorisnikaToolStripMenuItem.Visible = true;
         }
         else
         {
            sifarniciToolStripMenuItem.Visible = false;
            dodavanjeKorisnikaToolStripMenuItem.Visible = false;

         }
      }

      private void timer1_Tick(object sender, EventArgs e)
      {
         tsslVreme.Text =DateTime.Now.Hour.ToString() + ":" + DateTime.Now.Minute.ToString() + ":" +
DateTime.Now.Second.ToString();
      }

      private void iZLAZToolStripMenuItem_Click(object sender, EventArgs e)
      {
         Application.Exit();
      }

      private void dodavanjeKorisnikaToolStripMenuItem_Click(object sender, EventArgs e)
      {
         frmDodajKorisnika formaDodajKorisnika = new frmDodajKorisnika();
         formaDodajKorisnika.ShowDialog();
      }

      private void tipKreditaToolStripMenuItem_Click(object sender, EventArgs e)
      {
         frmTipKredita formTipKredita = new frmTipKredita();
         formTipKredita.ShowDialog();
      }

      private void kreditToolStripMenuItem_Click(object sender, EventArgs e)
      {
         frmKredit formaKredit = new frmKredit();
         formaKredit.ShowDialog();
      }
```

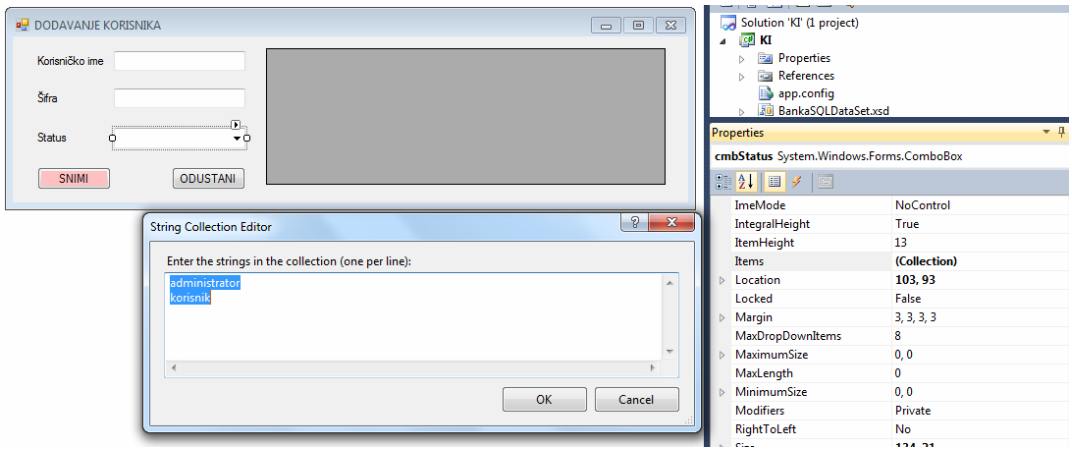
```
        private void grafikoniToolStripMenuItem_Click(object sender, EventArgs e)
        {
            frmGrafikon formaGrafikon = new frmGrafikon();
            formaGrafikon.ShowDialog();
        }
    }
}
```

10.7. ADMINISTRACIJA KORISNIKA SISTEMA

U nastavku je data ekranska forma za administraciju korisnika sistema (slika 11.7.1). Ova ekranska forma ima combo box sa fiksnim vrednostima unetim u properties, Items, collection (slika 11.7.2). Takođe, sadrži i DataGridView kontrolu.



Slika 11.7.1. Ekran za administraciju korisnika



Slika 11.7.2. Dodavanje stavki combo box-a u okviru Properties, Items (Collection)

U nastavku je dat programski kod forme za administraciju korisnika.

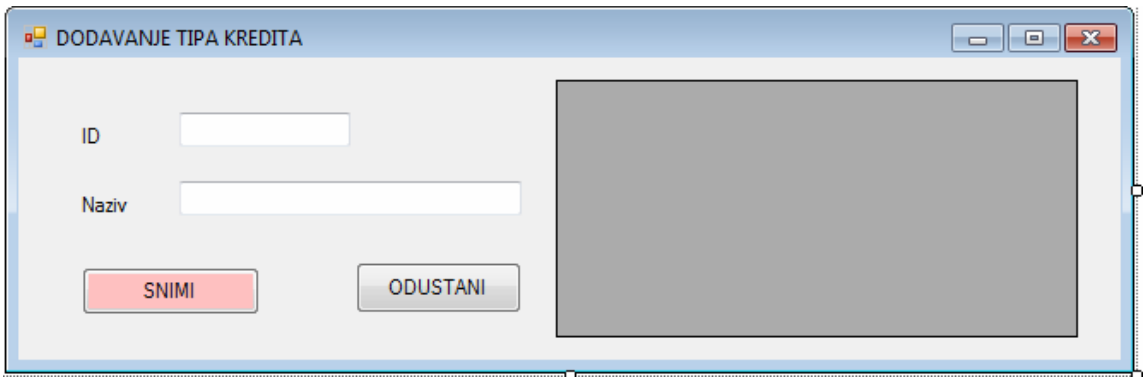
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
//
using System.Data.SqlClient;
using SqlDBUtils;


namespace KI
{
    public partial class frmDodajKorisnika : Form
    {
```

```
        // atributi
        clsSqlTabela pTabela;
        System.Data.DataSet dsKorisnik;

        // konstruktor
        public frmDodajKorisnika()
        {
            InitializeComponent();
        }

        // nase procedure
        private bool IzvrsiTransakciju(string KorisnickoIme, string Sifra, string Status)
        {
            bool uspeh;
            SqlConnection Konekcija;
            SqlCommand Komanda;
            SqlTransaction Transakcija = null;
            try
            {
                Konekcija = clsGlobal.Konekcija.DajKonekciju();
                // aktivan kod

                // povezivanje
                Komanda = new SqlCommand();
                Komanda.Connection = Konekcija;
                Komanda = Konekcija.CreateCommand();
                // pokretanje
                Transakcija = Konekcija.BeginTransaction();
                Komanda.Transaction = Transakcija;
                Komanda.CommandText = "INSERT INTO Korisnik VALUES ('" + KorisnickoIme + "','" + Sifra + "','" + Status
+ "')";
                Komanda.ExecuteNonQuery();
                Transakcija.Commit();
                uspeh = true;
            }
            catch
            {
                Transakcija.Rollback();
                uspeh = false;
            }
            return uspeh;
        }

        private bool IzvrsiStoredProceduru(string KorisnickoIme, string Sifra, string Status)
        {
            bool uspeh;
            SqlConnection Konekcija;
            SqlCommand Komanda;
            try
            {
                Konekcija = clsGlobal.Konekcija.DajKonekciju();

                // povezivanje
                Komanda = new SqlCommand();
                Komanda.Connection = Konekcija;
                Komanda = Konekcija.CreateCommand();
                Komanda.Parameters.Add("@KorIme", SqlDbType.NVarChar).Value = KorisnickoIme;
                Komanda.Parameters.Add("@Sifra", SqlDbType.NVarChar).Value = Sifra;
                Komanda.Parameters.Add("@Status", SqlDbType.NVarChar).Value = Status;
                Komanda.CommandType = CommandType.StoredProcedure;
                Komanda.CommandText = "UnosKorisnika";

                // aktivan kod
                Komanda.ExecuteNonQuery();
                uspeh = true;
            }
            catch
            {
                uspeh = false;
            }
            return uspeh;
```

```
        }

        private void NapuniGrid()
        {
            pTabela = new clsSqlTabela(clsGlobal.Konekcija, "Korisnik");
            dsKorisnik = pTabela.DajPodatke("Select * from Korisnik");
            // povezivanje grida sa datasetom
            dtgKorisnici.DataSource = dsKorisnik.Tables[0];
            dtgKorisnici.Refresh();
        }

        // dogadjaji
        private void btnPotvrdi_Click(object sender, EventArgs e)
        {
            // 1. nacin
            // bool uspeh= IzvrsiTransakciju(txtKorisničkoIme.Text, txtSifra.Text, cmbStatus.Text);

            // 2. nacin
            bool uspeh = IzvrsiStoredProceduru(txtKorisničkoIme.Text, txtSifra.Text, cmbStatus.Text);
            if (uspeh)
            {
                MessageBox.Show("Uspesno dodat korisnik!");
                NapuniGrid();
            }
            else
            {
                MessageBox.Show("NEUSPEH dodavanja korisnika!");
            }
        }

        private void frmDodajKorisnika_Load(object sender, EventArgs e)
        {
          NapuniGrid();
        }
    }
}
```

10.8. RAD SA SIFARNIKOM

Pomoćna šifarnička tabela TipKredita se ažurira sa forme gde je podržan unos i tabelarni prikaz podataka.



U nastavku je dat programski kod ekranske forme:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
//
```

```
using System.Data.SqlClient;
using SqlDBUtils;

namespace KI
{
   public partial class frmTipKredita : Form
   {
      // atributi
      clsSqlTabela pTabela;
      System.Data.DataSet dsTipKredita;

      // konstruktor
      public frmTipKredita()
      {
         InitializeComponent();
      }

      // nase procedure
      private bool IzvrsiTransakciju(string ID, string Naziv)
      {
         bool uspeh;
         SqlConnection Konekcija;
         SqlCommand Komanda;
         SqlTransaction Transakcija = null;
         try
         {
            Konekcija = clsGlobal.Konekcija.DajKonekciju();
            // aktivan kod

            // povezivanje
            Komanda = new SqlCommand();
            Komanda.Connection = Konekcija;
            Komanda = Konekcija.CreateCommand();
            // pokretanje
            Transakcija = Konekcija.BeginTransaction();
            Komanda.Transaction = Transakcija;
            Komanda.CommandText = "INSERT INTO TipKredita VALUES ('" + ID + "','" + Naziv + "')";
            Komanda.ExecuteNonQuery();
            Transakcija.Commit();
            uspeh = true;
         }
         catch
         {
            Transakcija.Rollback();
            uspeh = false;
         }
         return uspeh;
      }

      private void NapuniGrid()
      {
         pTabela = new clsSqlTabela(clsGlobal.Konekcija, "TipKredita");
         dsTipKredita = pTabela.DajPodatke("Select * from TipKredita");
         // povezivanje grida sa datasetom
         dtgTipoviKredita.DataSource = dsTipKredita.Tables[0];
         dtgTipoviKredita.Refresh();
      }

      // dogadjaji
      private void btnOdustani_Click(object sender, EventArgs e)
      {
         txtID.Text = "";
         txtNaziv.Text = "";
      }

      private void btnSnimi_Click(object sender, EventArgs e)
      {
         // 1. nacin
         bool uspeh= IzvrsiTransakciju(txtID.Text, txtNaziv.Text);

         if (uspeh)
```

```
        {
            MessageBox.Show("Uspesno dodat tip kredita!");
            NapuniGrid();
        }
        else
        {
            MessageBox.Show("NEUSPEH dodavanja tipa kredita!");
        }

    }

    private void frmTipKredita_Load(object sender, EventArgs e)
    {
        NapuniGrid();
    }
  }
}
```

10.9. OSNOVNI EKRAN ZA RAD

Osnovni ekran za rad sastoji se od Tab kontrole sa 3 ekranske kartice – ažuriranje, pretraga i štampa.

10.9.1. Osnovna podešavanja i stanja forme

U nastavku su dati delovi programskog koda ekranske forme sa objašnjenjima:

Using deo – uključujemo deo za referenciranje biblioteka klasa.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
// - dodato
using GeneralUtils;
using SqlDBUtils;
using DataInteropUtils;
```

U strukturi ekranske forme koja je zapravo klasa razlikujemo atribute i metode. Postavljamo region atributa na početak, zatim region „naših procedura", konstruktora i događaja.

```
namespace KI
{
  public partial class frmKredit : Form
  {

      #region Atributi

- razlikujemo stanja forme za unos, prikaz, brisanje, izmenu, pa globalna promenljiva na nivou forme «stanjeForme» čuva vrednost o aktuelnom stanju forme u svakom trenutku.
      private string stanjeForme;
` objekat klase clsSQLTabela koristi se za rad sa tabelama iz MSSQlServer baze podataka
      private clsSqlTabela  pTabela;
` Dataset predstavlja memorijsku kolekciju zapisa, odgovara strukturi podataka tabele ili SQL upita na osnovu kojih se podaci preuzimaju iz baze podataka u DAtaset
      private System.Data.DataSet dsKredit;
- promenljiva RedniBrojSloga vodi računa o tekućem broju sloga kod navigacije, tj. pozicioniranja
      private int RedniBrojSloga;
- promenljiva Ukupno vodi računa o ukupnom broju zapisa u tabeli, odnosno dataset-u
      private int Ukupno;
- promeljiva kliknuto se koristi kod provere da li je izabrana stavka sa grida
      private bool kliknuto;
```

```
- promeljiva StariBrojRacuna se koristi  da bi se sacuvala stara vrednost identifikacionog obelezja radi pozicioniranja kod izmene podataka
      private string StariBrojRacuna;
` koristimo gotov Select upit sa svim navedenim poljima za tabelarni prikaz
      private string SelectUpit="select Kredit.BrojRacuna, TipKredita.Naziv, Kredit.JMBG, Kredit.DatumPocetka, Kredit.TrajanjeMeseci, Kredit.Kamata, Kredit.Iznos from Kredit inner join TipKredita on Kredit.IdTipaKredita=TipKredita.Id";
      #endregion
```

` konstruktor je procedura koja se prva izvršava i najčešće sadrži inicijalizaciju promenljivih

```
      // konstruktor
      public frmKredit()
      {
         InitializeComponent();
         // inicijalizacija promenljivih
         stanjeForme = "prikaz";
         RedniBrojSloga = 0;
         Ukupno = 0;
         kliknuto = false;
         StariBrojRacuna = "";

      }
```

` postavljanje forme u određeno stanje

```
      private void PostaviStanjeForme(string stanje)
      // set procedura za atribut
      {
         stanjeForme = stanje;
         switch (stanjeForme)
         {
            case "prikaz":
               tsbPrvi.Enabled = true;
               tsbPrethodni.Enabled = true;
               tsbSledeci.Enabled = true;
               tsbPoslednji.Enabled = true;
               //
               tsbDodaj.Enabled = true;
               tsbIzmeni.Enabled = true;
               tsbObrisi.Enabled = true;
               //
               tsbSnimi.Enabled = false;
               tsbOdustani.Enabled = false;
               //
               btnDodajTipKredita.Enabled = false;
               //
               txtBrojRacuna.Enabled = false;
               cmbTipKredita.Enabled = false;
               txtJMBG.Enabled = false;
               txtDan.Enabled = false;
               txtMesec.Enabled = false;
               txtGodina.Enabled = false;
               txtTrajanjeMeseci.Enabled = false;
               txtKamata.Enabled = false;
               txtIznos.Enabled = false;
               break;
            case "unos":
               tsbPrvi.Enabled = false;
               tsbPrethodni.Enabled = false;
               tsbSledeci.Enabled = false;
               tsbPoslednji.Enabled = false;
               //
               tsbDodaj.Enabled = false;
               tsbIzmeni.Enabled = false;
               tsbObrisi.Enabled = false;
               //
               tsbSnimi.Enabled = true;
               tsbOdustani.Enabled = true;
               //
```

```
                btnDodajTipKredita.Enabled = true;
                //
                 txtBrojRacuna.Enabled = true;
                 cmbTipKredita.Enabled = true;
                 txtJMBG.Enabled = true;
                 txtDan.Enabled = true;
                 txtMesec.Enabled = true;
                 txtGodina.Enabled = true;
                 txtTrajanjeMeseci.Enabled = true;
                 txtKamata.Enabled = true;
                 txtIznos.Enabled = true;
                //
                IsprazniKontroleUnosa();
                break;
            case "izmena":
                tsbPrvi.Enabled = false;
                tsbPrethodni.Enabled = false;
                tsbSledeci.Enabled = false;
                tsbPoslednji.Enabled = false;
                //
                tsbDodaj.Enabled = false;
                tsbIzmeni.Enabled = false;
                tsbObrisi.Enabled = false;
                //
                tsbSnimi.Enabled = true;
                tsbOdustani.Enabled = true;
                //
                btnDodajTipKredita.Enabled = true;
                //
                txtBrojRacuna.Enabled = true;
                 cmbTipKredita.Enabled = true;
                 txtJMBG.Enabled = true;
                 txtDan.Enabled = true;
                 txtMesec.Enabled = true;
                 txtGodina.Enabled = true;
                 txtTrajanjeMeseci.Enabled = true;
                 txtKamata.Enabled = true;
                 txtIznos.Enabled = true;
                break;
            case "brisanje":
                tsbPrvi.Enabled = false;
                tsbPrethodni.Enabled = false;
                tsbSledeci.Enabled = false;
                tsbPoslednji.Enabled = false;
                //
                tsbDodaj.Enabled = false;
                tsbIzmeni.Enabled = false;
                tsbObrisi.Enabled = false;
                //
                tsbSnimi.Enabled = true;
                tsbOdustani.Enabled = true;
                //
                btnDodajTipKredita.Enabled = false;
                //
                txtBrojRacuna.Enabled = false;
                cmbTipKredita.Enabled = false;
                txtJMBG.Enabled = false;
                txtDan.Enabled = false;
                txtMesec.Enabled = false;
                txtGodina.Enabled = false;
                txtTrajanjeMeseci.Enabled = false;
                txtKamata.Enabled = false;
                txtIznos.Enabled = false;
                break;
        }
        tsbPoruka.Text = "U toku je " + stanjeForme + " podataka";
    }
```

` procedure za punjenje lista combo boxa

```
private void NapuniComboBrojRacuna()
```

```
    {
        //
        clsSqlTabela pTabelaKredit = new clsSqlTabela(clsGlobal.Konekcija, "Kredit");
        System.Data.DataSet dsKredit = pTabelaKredit.DajPodatke("select BrojRacuna from Kredit");

        //praznjenje liste
        cmbBrojRacunaFilter.Items.Clear();

        //Popunjavanje liste  kredita sa nazivima iz objekta tipa DataSet
        for (int i = 0; i <= dsKredit.Tables[0].Rows.Count - 1; i++)
        {
            cmbBrojRacunaFilter.Items.Add(dsKredit.Tables[0].Rows[i].ItemArray[0].ToString());
         }

    }


    private void NapuniComboTipKredita()
    {
        clsSqlTabela pTabelaTipKredita = new clsSqlTabela(clsGlobal.Konekcija, "TipKredita");
        System.Data.DataSet dsTipKredita = pTabelaTipKredita.DajPodatke("select * from TipKredita");

        // praznjenje tekuce liste
        cmbTipKredita.Items.Clear();
        cmbFilterTipKredita.Items.Clear();

        //Popunjavanje liste tipova kredita sa nazivima iz objekta tipa DataSet
        for (int i = 0; i <= dsTipKredita.Tables[0].Rows.Count - 1; i++)
        {
            cmbTipKredita.Items.Add(dsTipKredita.Tables[0].Rows[i].ItemArray[1].ToString());
            cmbFilterTipKredita.Items.Add(dsTipKredita.Tables[0].Rows[i].ItemArray[1].ToString());
        }

    }
```

Učitavanje i osvežavanje liste combo boxa

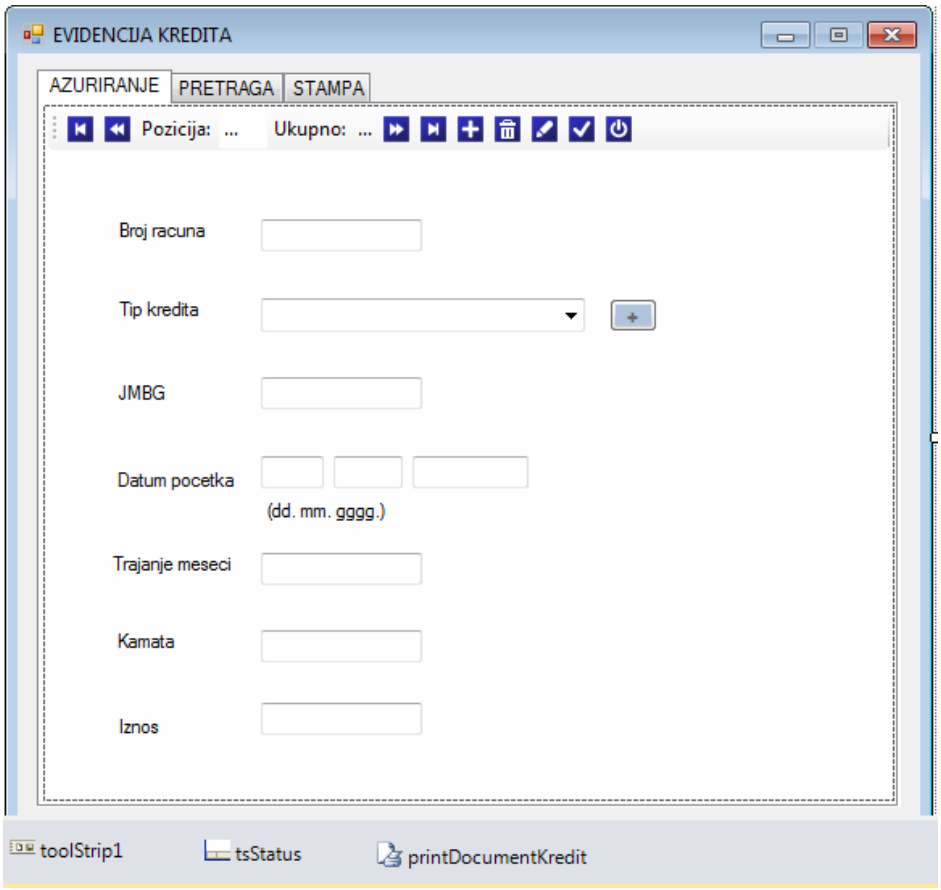
```
    private void frmKredit_Activated(object sender, EventArgs e)
    {
        NapuniComboTipKredita();
    }

    private void tabControl1_Selected(object sender, TabControlEventArgs e)
    {
        if (e.TabPageIndex.Equals(2))
        {
            NapuniComboBrojRacuna();
        }
    }
```

` procedura za osvežavanje prikaza podataka – bitna je zbog dopune kod unosa

```
    private void OsveziPrikazPodataka()
    {
        kliknuto = false;
        dsKredit = pTabela.DajPodatke(SelectUpit);
        NapuniGrid(dsKredit);
        PrikaziUkupanBrojSlogova();
        NapuniComboBrojRacuna();

    }
```

10.9.2. AŽURIRANJE PODATAKA

Slika 1. prikazuje karticu ažuriranje, koja sadrži tool strip sa slicicama koje ilustruju funkcionalnost odgovarajucih tastera, a takodje i labele za ispis pozicije i ukupnog broja zapisa u navigaciji. Takođe sadrzi i text box-ove, combo box gde se vrednosti dinamicki citaju i kroz kod pune iz baze podataka. Povodom datuma koriste se 3 text boxa umesto Date/time picker kontrole. Dodatni taster pored combo boxa omogucuje pokretanje pomocnog ekrana za rad sa sifarnikom tipova korisnika, kojim se lista combo boxa moze prosiriti.



U nastavku je dat prikaz ključnih delova programskog koda.

- pražnjenje kontrola radi pripreme unosa

```
#region Nase procedure

private void IsprazniKontroleUnosa()
{
   txtBrojRacuna.Text = "";
   cmbTipKredita.Text = "";
   txtJMBG.Text = "";
   txtDan.Text = "";
   txtMesec.Text = "";
   txtGodina.Text = "";
   txtTrajanjeMeseci.Text = "";
   txtKamata.Text = "";
   txtIznos.Text = "";
}
```

` preračunavanje naziva iz combo boxa u sifru (ID) zbog omogucavanja da se sifra upise u bazu podataka

```
private string DajIDTipaKredita(string NazivTipaKredita)
      // potrebna zbog snimanja podataka
      //- kada se preuzmu podaci iz combo bude naziv
      // a potrebna je sifra
      {
         string ID = "";
         clsSqlTabela  pTabelaTipKredita = new clsSqlTabela (clsGlobal.Konekcija, "TipKredita");
         System.Data.DataSet dsTipKredita = pTabelaTipKredita.DajPodatke("select Id from TipKredita where Naziv='"
+ NazivTipaKredita + "'");
         // moze biti samo jedan slog koji odgovara tako da je rows od 0, a posto u select ima samo 1 polje,onda je
itemarray od 0
         ID = dsTipKredita.Tables[0].Rows[0].ItemArray[0].ToString();
         return ID;
      }
```

` provera da li su sve kontrole popunjene prilikom unosa

```
      private bool ImaPraznaKontrolaUnosa()
      //vraca true ako je samo bar jedna
      {
         bool uspeh, uspeh1, uspeh2, uspeh3, uspeh4, uspeh5, uspeh6, uspeh7, uspeh8, uspeh9;
         uspeh1= (txtBrojRacuna.Text == "");
         uspeh2 = (cmbTipKredita.Text == "");
         uspeh3 = (txtJMBG.Text == "");
         uspeh4 = (txtDan.Text == "");
         uspeh5 = (txtMesec.Text == "");
         uspeh6 = (txtGodina.Text == "");
         uspeh7 = (txtTrajanjeMeseci.Text == "");
         uspeh8 = (txtKamata.Text == "");
         uspeh9 = (txtIznos.Text == "");
         // logicki AND = &, logicki OR = ||
         uspeh = uspeh1 || uspeh2 || uspeh3 || uspeh4 || uspeh5 || uspeh6 || uspeh7 || uspeh8 || uspeh9;
         return uspeh;
      }
```

` provere ispravnosti podataka prilikom unosa

```
private bool PodatakSaComboListe(string UnetiNazivTipaKredita)
      // provera da li je podatak koji treba da bude
      // snimljen jedan od podataka iz liste combo boxa
      {
         bool SaListe = false;
         clsSqlTabela pTabelaTipKredita = new clsSqlTabela(clsGlobal.Konekcija, "TipKredita");
         System.Data.DataSet dsTipKredita = pTabelaTipKredita.DajPodatke("select * from TipKredita where Naziv='" +
UnetiNazivTipaKredita + "'");
         SaListe = (dsTipKredita.Tables[0].Rows.Count > 0);
         return SaListe;
      }

      private bool PostojiVrednostPrimarnogKljuca(string UnetiBrojRacuna)
      //provera da li vec postoji ranije unet slog
      // koji ima istu vrednost primarnog kljuca
      {
         clsSqlTabela pTabelaKreditPK = new clsSqlTabela(clsGlobal.Konekcija, "Kredit");
         System.Data.DataSet dsKreditPK = pTabelaKreditPK.DajPodatke("select * from Kredit where BrojRacuna='" +
UnetiBrojRacuna + "'");
         bool postoji = (dsKreditPK.Tables[0].Rows.Count > 0);
         return postoji;
      }

      private bool PostojiVrednostAltPK(string IdTipaKredita, string JMBG, DateTime DatumPocetka)
      //provera da li vec postoji ranije unet slog
      // koji ima istu vrednost alternativnog kljuca
      // u ovom konkretnom slucaju nije postavljen alt pk
      // nad nazivom mesta, jer mora se omoguciti da
      // postoje 2 mesta sa istim nazivom, jer je tako u realnosti
      // ali ova procedura ipak proverava da li vec postoji
```

```
      // i sluzi samo za upozorenje
      {
         clsSqlTabela pTabelaKreditALTK = new clsSqlTabela(clsGlobal.Konekcija, "Kredit");
         System.Data.DataSet dsKreditALTK = pTabelaKreditALTK.DajPodatke("select * from Kredit where
IdTipaKredita='" + IdTipaKredita + "' and JMBG='" + JMBG + "' and Year(DatumPocetka)=" +
DatumPocetka.Year.ToString() + " and Month(DatumPocetka)=" + DatumPocetka.Month.ToString() + " and
Day(DatumPocetka)=" + DatumPocetka.Day.ToString ());
         bool postoji = (dsKreditALTK.Tables[0].Rows.Count > 0);
         return postoji;
      }


      private bool IspravniPodaci()
      // objedinjuje sve procedure za proveru potpunosti
      // i ispravnosti podataka
      {
         bool uspeh = false;

         if ((stanjeForme == "unos") || (stanjeForme == "izmena"))
         {

            bool prazno = ImaPraznaKontrolaUnosa();
            if (prazno == true)
            {
               MessageBox.Show("Nisu sva polja popunjena!");
               uspeh = false;
               txtBrojRacuna.Focus();
               return uspeh;
            }
            else // sve je popunjeno
            {
               bool SaListe = PodatakSaComboListe(cmbTipKredita.Text);
               if (SaListe == false)
               {
                  MessageBox.Show("Nije unet podatak o Tipu kredita sa dozvoljene liste vrednosti!");
                  uspeh = false;
                  cmbTipKredita.Focus();
                  return uspeh;
               }
               else // jeste sa liste
               {
                  if (stanjeForme.Equals("unos"))
                  {
                     bool postoji = PostojiVrednostPrimarnogKljuca(txtBrojRacuna.Text);
                     if (postoji == true)
                     {
                        MessageBox.Show("Vec postoji uneti kredit sa istom vrednoscu  Broja racuna!");
                        uspeh = false;
                        txtBrojRacuna.Focus();
                     }
                     else
                     {
                        uspeh = true;
                     };
                     return uspeh;
                  }
                  else // nije stanje unosa i ne proverava se jedinstvenost primarnog kljuca
                  {
                     uspeh = true;
                     return uspeh;
                  }
               }
            }
         }
         else // stanje forme =brisanje
         {
            uspeh = true;
            return uspeh;
         }
      }
```

` procedura za snimanje podataka u bazu podataka, odnosno brisanje, izmenu i unos

```
private void SnimiPodatke()
{
```

- preuzimamo podatke u promenljive tipa string, jer ce te vrednosti biti "zalepljene" za string SQL upita i nema potrebe za konverzijama u druge tipove podataka
` ovde samo imamo konverziju naziva u sifru iz combo box-a

```
// preuzimanje podataka sa korisnickog interfejsa
        // i odgovarajuce konverzije
        string BrojRacuna = txtBrojRacuna.Text  ;
        string IDTipaKredita = DajIDTipaKredita (cmbTipKredita.Text);
        string JMBG=txtJMBG.Text;
        string TrajanjeMeseci= txtTrajanjeMeseci.Text ;
        string Kamata =txtKamata.Text;
        string Iznos = txtIznos.Text;
```

` kod formatiranja datumske vrednosti koristimo americki stil mesec/dan/godina, a pre i posle stavljamo simbol # kod MS Access baze podataka, a kod MS SQL Server baze podataka koristimo apostrof – jednostruki navodnik

```
        // za Access bazu sa tarabom #
        //string DatumPocetka = "#" + txtMesec.Text + "/" + txtDan.Text + "/" + txtGodina.Text + "#";
        // za SQL server bazu sa apostrofom '
        string DatumPocetka = "'" + txtMesec.Text + "/" + txtDan.Text + "/" + txtGodina.Text + "'";

        // izvrsavanje snimanja
        clsSqlTabela pTabelaAzuriranje = new clsSqlTabela(clsGlobal.Konekcija, "Kredit");
        string Upit = "";
        switch (stanjeForme)
        {
```

NAPOMENE:

- *` kod formiranja SQL upita vodimo racuna da samo string tip podatka iz baze podataka ima apostrofe oko vrednosti, dok ostala polja drugih tipova podataka nemaju apostrofe (tacnije, numericke nemaju, a datumske su vec ranije kreirane sa # ili apostrofom).*
- *` kod insert into ne moramo navoditi spisak naziva svih polja iza naziva tabele, ali u okviru Values moramo postovati redosled navodjenja polja iz dizajna baze podataka*
- *Ako bismo imali potrebu za konverzijama tipova podataka npr. Radi numeričkih operacija, možemo npr. koristiti: float.Parse(txtUkupanIznos.Text);*

```
            case "unos":
                Upit = "INSERT INTO Kredit VALUES ('" + BrojRacuna + "','" + IDTipaKredita + "','" + JMBG + "'," +
DatumPocetka  + "," + TrajanjeMeseci + "," + Kamata + "," + Iznos + ")";
                break;
            case "brisanje":
                Upit = "DELETE * from Kredit where BrojRacuna='" +  BrojRacuna + "'";
                break;
            case "izmena":
                Upit = "UPDATE Kredit SET BrojRacuna='" + BrojRacuna + "', IdTipaKredita='" + IDTipaKredita + "',
JMBG='" + JMBG + "', DatumPocetka=" + DatumPocetka  + ", TrajanjeMeseci=" + TrajanjeMeseci + ", Kamata=" +
Kamata + ", Iznos=" + Iznos + " WHERE BrojRacuna='" + StariBrojRacuna + "'";
                break;
        }
        pTabelaAzuriranje.IzvrsiAzuriranje(Upit);
        MessageBox.Show("Uspesno je zavrsen " + stanjeForme + " podataka");

        // regulisanje stanja forme nakon uspesnog zavrsetka posla
        switch (stanjeForme)
        {
            case "unos":
                Ukupno++;
                RedniBrojSloga++;
                break;
            case "brisanje":
                Ukupno--;
                RedniBrojSloga--;
                break;
            case "izmena":
                //
```

```
                break;
            }
            OsveziPrikazPodataka();
            PrikaziLokacijuSloga(RedniBrojSloga);



            switch (stanjeForme)
            {
               case "unos":
                  PrikaziAktuelnog(BrojRacuna);
                  break;
               case "brisanje":
                  IsprazniKontroleUnosa();
                  PrikaziPrvog();
                  break;
               case "izmena":
                  PrikaziAktuelnog(BrojRacuna);
                  break;
            }
            PostaviStanjeForme("prikaz");
      }
```

` procedure – događaji za pokretanje opcija za ažuriranje

```
private void tsbDodaj_Click(object sender, EventArgs e)
      {
         PostaviStanjeForme("unos");
      }

      private void tsbObrisi_Click(object sender, EventArgs e)
      {
         PostaviStanjeForme("brisanje");
      }

      private void tsbIzmeni_Click(object sender, EventArgs e)
      {
         // preuzimanje starih vrednosti radi pristupa kod izmene
         StariBrojRacuna = txtBrojRacuna.Text;

         PostaviStanjeForme("izmena");

      }

      private void tsbSnimi_Click(object sender, EventArgs e)
      {
         DialogResult rezultat = MessageBox.Show("Da li ste sigurni da zelite da snimite podatke?", "UPOZORENJE",
MessageBoxButtons.YesNo);
         if (rezultat.ToString().Equals("Yes"))
         {
            try
            {
               // aktivan kod koji moze izazvati gresku

               bool ispravniPodaci = IspravniPodaci();
               if (ispravniPodaci == true)
               {
                  // za Access i SQL server za transformaciju u datumski tip DateTime.Parse koristi # -  kod datumske
vrednosti
                  DateTime DatumPocetka = DateTime.Parse( "#" + txtMesec.Text + "/" + txtDan.Text + "/" +
txtGodina.Text + "#");
                  bool postojiAltPK = PostojiVrednostAltPK(DajIDTipaKredita (cmbTipKredita.Text),txtJMBG.Text,
DatumPocetka);
                  if ((postojiAltPK == true) && (stanjeForme.Equals("unos")))
                  {
                     // ovde nije zabrana,vec samo upozorenje
                     DialogResult odgovor = MessageBox.Show("Vec postoji Kredit sa  istim JMBG, tipom kredita i
datumom pocetka. Da li ste sigurni da zelite da snimite podatke o Kreditu sa identicnim podacima?", "UPOZORENJE",
MessageBoxButtons.YesNo);
                     if (rezultat.ToString().Equals("Yes"))
                     {
                        SnimiPodatke();
```

```
                    }
                    else // postoji sa istim nazivom i korisnik hoce da ipak promeni
                    {
                        txtBrojRacuna.Focus();
                    }
                } //
                else  // ne postoji AltPK i sve je u redu sa podacima
                {
                    SnimiPodatke();
                }
            }
            else // NEISPRAVNI PODACI
            {
                txtBrojRacuna.Focus();
            }
        }

        catch (Exception Greska)
        {
            MessageBox.Show("Neuspesno je zavrsen " + stanjeForme + " podataka. Postoji greska:" +
Greska.Message);
        }

    }

}

private void tsbOdustani_Click(object sender, EventArgs e)
{
    DialogResult rezultat = MessageBox.Show("Da li ste sigurni da zelite da odustanete?", "UPOZORENJE",
MessageBoxButtons.YesNo);
    if (rezultat.ToString().Equals("Yes"))
    {
        if (stanjeForme.Equals("unos"))
        {
            IsprazniKontroleUnosa();
            PrikaziPrvog();
        }
        if (stanjeForme.Equals("izmena"))
        {
            PrikaziTekuceg();
        }
        PostaviStanjeForme("prikaz");
    }
}
```

` procedure za sprecavanje unosa neispravnih karaktera

```
private void txtBrojRacuna_KeyPress(object sender, KeyPressEventArgs e)
{

    if (e.KeyChar == (char)Keys.Enter)
    {
        txtBrojRacuna.Focus();
    }

    // Dozvoljeni su: cifre 0-9, Backspace i ENTER
    // AKO SU NEDOZVOLJENI KARAKTERI, ONDA NE REAGUJE
    if (!clsGlobal.Karakter(e, "cifra") && !clsGlobal.Karakter(e, "BACKSPACE") && !clsGlobal.Karakter(e, "ENTER")
&& !e.KeyChar.Equals("-"))
    {
        e.Handled = true;
    }

}

private void txtJMBG_KeyPress(object sender, KeyPressEventArgs e)
{
```

```
        if (e.KeyChar == (char)Keys.Enter)
        {
            txtJMBG.Focus();
        }

        // Dozvoljeni su: cifre 0-9, Backspace i ENTER
        // AKO SU NEDOZVOLJENI KARAKTERI, ONDA NE REAGUJE
        if (!clsGlobal.Karakter(e, "cifra") && !clsGlobal.Karakter(e, "BACKSPACE") && !clsGlobal.Karakter(e, "ENTER"))
        {
            e.Handled = true;
        }

    }

    private void txtDan_KeyPress(object sender, KeyPressEventArgs e)
    {

        if (e.KeyChar == (char)Keys.Enter)
        {
            txtDan.Focus();
        }

        // Dozvoljeni su: cifre 0-9, Backspace i ENTER
        // AKO SU NEDOZVOLJENI KARAKTERI, ONDA NE REAGUJE
        if (!clsGlobal.Karakter(e, "cifra") && !clsGlobal.Karakter(e, "BACKSPACE") && !clsGlobal.Karakter(e, "ENTER"))
        {
            e.Handled = true;
        }

    }

    private void txtMesec_KeyPress(object sender, KeyPressEventArgs e)
    {

        if (e.KeyChar == (char)Keys.Enter)
        {
            txtMesec.Focus();
        }

        // Dozvoljeni su: cifre 0-9, Backspace i ENTER
        // AKO SU NEDOZVOLJENI KARAKTERI, ONDA NE REAGUJE
        if (!clsGlobal.Karakter(e, "cifra") && !clsGlobal.Karakter(e, "BACKSPACE") && !clsGlobal.Karakter(e, "ENTER"))
        {
            e.Handled = true;
        }

    }

    private void txtGodina_KeyPress(object sender, KeyPressEventArgs e)
    {

        if (e.KeyChar == (char)Keys.Enter)
        {
            txtGodina.Focus();
        }

        // Dozvoljeni su: cifre 0-9, Backspace i ENTER
        // AKO SU NEDOZVOLJENI KARAKTERI, ONDA NE REAGUJE
        if (!clsGlobal.Karakter(e, "cifra") && !clsGlobal.Karakter(e, "BACKSPACE") && !clsGlobal.Karakter(e, "ENTER"))
        {
            e.Handled = true;
        }

    }

    private void txtTrajanjeMeseci_KeyPress(object sender, KeyPressEventArgs e)
    {

        if (e.KeyChar == (char)Keys.Enter)
        {
            txtTrajanjeMeseci.Focus();
```

```
        }

        // Dozvoljeni su: cifre 0-9, Backspace i ENTER
        // AKO SU NEDOZVOLJENI KARAKTERI, ONDA NE REAGUJE
        if (!clsGlobal.Karakter(e, "cifra") && !clsGlobal.Karakter(e, "BACKSPACE") && !clsGlobal.Karakter(e, "ENTER"))
        {
           e.Handled = true;
        }

     }

     private void txtKamata_KeyPress(object sender, KeyPressEventArgs e)
     {

        if (e.KeyChar == (char)Keys.Enter)
        {
           txtKamata.Focus();
        }

        // Dozvoljeni su: cifre 0-9, Backspace i ENTER
        // AKO SU NEDOZVOLJENI KARAKTERI, ONDA NE REAGUJE
        if (!clsGlobal.Karakter(e, "cifra") && !clsGlobal.Karakter(e, "BACKSPACE") && !clsGlobal.Karakter(e, "ENTER")
&& !e.KeyChar.Equals(",") && !e.KeyChar.Equals("."))
        {
           e.Handled = true;
        }

     }

     private void txtIznos_KeyPress(object sender, KeyPressEventArgs e)
     {

        if (e.KeyChar == (char)Keys.Enter)
        {
           txtIznos.Focus();
        }

        // Dozvoljeni su: cifre 0-9, Backspace i ENTER
        // AKO SU NEDOZVOLJENI KARAKTERI, ONDA NE REAGUJE
        if (!clsGlobal.Karakter(e, "cifra") && !clsGlobal.Karakter(e, "BACKSPACE") && !clsGlobal.Karakter(e, "ENTER")
&& !e.KeyChar.Equals(",") && !e.KeyChar.Equals("."))
        {
           e.Handled = true;
        }

     }
```

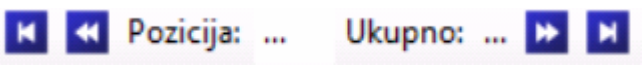
` procedura – dogadjaj za pokretanje forme za dopunu liste combo boxa

```
     private void btnDodajTipKredita_Click(object sender, EventArgs e)
     {
        frmTipKredita formaTipKredita = new frmTipKredita();
        formaTipKredita.ShowDialog();
     }
```

10.9.3. NAVIGACIJA I POZICIONIRANJE

U nastavku je dat prikaz programskog koda za realizaciju navigacije.

1. **Način – pozicioniranje korišćenjem tastera PRVI, PRETHODNI, SLEDEĆI, POSLEDNJI**



- procedura za prikaz ukupnog broja slogova

```
private void PrikaziUkupanBrojSlogova()
      {
         Ukupno = pTabela.DajBrojSlogova();
         tslUkupno.Text = Ukupno.ToString();
      }
```

` procedura za prikaz tekuce pozicije zapisa

```
      private void PrikaziLokacijuSloga(int RB)
      {
         int lokacija = RB + 1;
         tstPozicija.Text = lokacija.ToString();
      }
```

` procedura za preracunavanje sifre iz baze podataka u naziv, radi prikazivanja u combo boxu prilikom navigacije

```
      private string DajNazivTipaKredita(string ID)
      // potrebno zbog prikaza podataka analiticki
      // u bazi je sifra, a u korisnickom interfejsu je naziv
      {
         string Naziv = "";
         clsSqlTabela  pTabelaTipKredita = new clsSqlTabela(clsGlobal.Konekcija, "TipKredita");
         System.Data.DataSet dsTipKredita = pTabelaTipKredita.DajPodatke("select Naziv from TipKredita where Id='"
+ ID + "'");
         // moze biti samo jedan slog koji odgovara tako da je rows od 0, a posto u select ima samo 1 polje,onda je
itemarray od 0
         Naziv = dsTipKredita.Tables[0].Rows[0].ItemArray[0].ToString();
         return Naziv;
      }
```

` procedura koja se koristi kod navigacije radi prikaza podataka tekuceg zapisa

```
      private void PrikaziTekuceg()
      {
         // PREDUSLOV: prethodno je dsMesto napunjen bilo sa select * ili konkretnim poljima

         // gde god da se nalazi, prikazati u toolstripu lokaciju
         PrikaziLokacijuSloga(RedniBrojSloga);

         // 1. varijanta: ako postavimo da dsKredit sadrzi: "select * from Kredit" onda:
         //txtBrojRacuna.Text = dsKredit.Tables[0].Rows[RedniBrojSloga].ItemArray[0].ToString();
         //cmbTipKredita.Text =
DajNazivTipaKredita(dsKredit.Tables[0].Rows[RedniBrojSloga].ItemArray[1].ToString());
         //txtJMBG.Text = dsKredit.Tables[0].Rows[RedniBrojSloga].ItemArray[2].ToString();

         //DateTime datumPocetka =
DateTime.Parse(dsKredit.Tables[0].Rows[RedniBrojSloga].ItemArray[3].ToString());
         //txtDan.Text = datumPocetka.Day.ToString();
         //txtMesec.Text = datumPocetka.Month.ToString();
         //txtGodina.Text = datumPocetka.Year.ToString();

         //txtTrajanjeMeseci.Text = dsKredit.Tables[0].Rows[RedniBrojSloga].ItemArray[4].ToString();
         //txtKamata.Text = dsKredit.Tables[0].Rows[RedniBrojSloga].ItemArray[5].ToString();
         //txtIznos.Text = dsKredit.Tables[0].Rows[RedniBrojSloga].ItemArray[6].ToString();

         // OVDE IMAMO 2. VARIJANTU:
         // private string SelectUpit="select Kredit.BrojRacuna, TipKredita.Naziv, Kredit.JMBG, Kredit.DatumPocetka,
Kredit.TrajanjeMeseci, Kredit.Kamata, Kredit.Iznos from Kredit inner join TipKredita on
Kredit.IdTipaKredita=TipKredita.Id order by Kredit.DatumPocetka desc";
         // 2. varijanta: dsKredit sadrzi: "select KONKRETNA POLJA from..." onda:
         txtBrojRacuna.Text = dsKredit.Tables[0].Rows[RedniBrojSloga].ItemArray[0].ToString();
         cmbTipKredita.Text = dsKredit.Tables[0].Rows[RedniBrojSloga].ItemArray[1].ToString();
         txtJMBG.Text = dsKredit.Tables[0].Rows[RedniBrojSloga].ItemArray[2].ToString();

         DateTime datumPocetka = DateTime.Parse(dsKredit.Tables[0].Rows[RedniBrojSloga].ItemArray[3].ToString());
```

```
            txtDan.Text = datumPocetka.Day.ToString();
            txtMesec.Text = datumPocetka.Month.ToString();
            txtGodina.Text = datumPocetka.Year.ToString();

            txtTrajanjeMeseci.Text = dsKredit.Tables[0].Rows[RedniBrojSloga].ItemArray[4].ToString();
            txtKamata.Text = dsKredit.Tables[0].Rows[RedniBrojSloga].ItemArray[5].ToString();
            txtIznos.Text = dsKredit.Tables[0].Rows[RedniBrojSloga].ItemArray[6].ToString();
        }
```

` procedura za prikaz prvog zapisa

```
private void PrikaziPrvog()
        {
            RedniBrojSloga = 0;
            Ukupno = pTabela.DajBrojSlogova();
            if (Ukupno > 0)
            {
                PrikaziTekuceg();
            }
            else
            {
                IsprazniKontroleUnosa();
            };
        }
```

procedura koja na osnovu vrednosti identifikacionog obelezja daje redni broj zapisa u datasetu

```
        private int DajRBSloga(string BrojRacuna)
        {
            int RB = 0;
            for (int i = 0; i <= dsKredit.Tables[0].Rows.Count - 1; i++)
            {
                if (dsKredit.Tables[0].Rows[i].ItemArray[0].ToString() == BrojRacuna)
                {
                    RB = i;
                }
            }

            return RB;
        }
```

procedura za prikaz aktuelnog zapisa

```
        private void PrikaziAktuelnog(string BrojRacuna)
        {
            Ukupno = pTabela.DajBrojSlogova();
            if (Ukupno > 0)
            {
                RedniBrojSloga = DajRBSloga(BrojRacuna);
                PrikaziTekuceg();
            }
            else
            {
                IsprazniKontroleUnosa();
            };
        }
```

procedure – dogadjaji za pokretanje navigacije

```
        private void tsbPrvi_Click(object sender, EventArgs e)
        {
            RedniBrojSloga = 0;
            PrikaziTekuceg();
            PostaviStanjeForme("prikaz");
        }

        private void tsbPrethodni_Click(object sender, EventArgs e)
        {
            if (RedniBrojSloga > 0)
            {
                RedniBrojSloga--;
                PrikaziTekuceg();
```

```
        }
        PostaviStanjeForme("prikaz");
    }

    private void tsbSledeci_Click(object sender, EventArgs e)
    {
        if (RedniBrojSloga < Ukupno - 1)
        {
            RedniBrojSloga++;
            PrikaziTekuceg();
        }
        PostaviStanjeForme("prikaz");
    }

    private void tsbPoslednji_Click(object sender, EventArgs e)
    {
        RedniBrojSloga = Ukupno - 1;
        PrikaziTekuceg();
        PostaviStanjeForme("prikaz");
    }
```

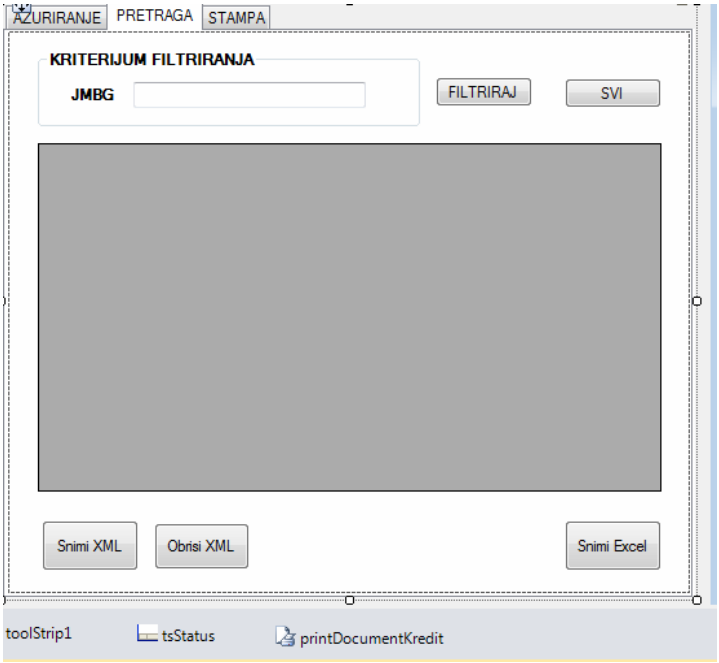
2. NAČIN – izborom željenog zapisa iz grida

- pozicioniranje na zapis klikom na stavku grida

```
    private void dtgKredit_CurrentCellChanged(object sender, EventArgs e)
    {
        // tek kada se klikne na neki red,
        // treba te podatke prikazati analiticki
        // a posto se ovaj dogadjaj moze pokrenuti
        // i prilikom formload-a ili ucitavanja grida
        // na neki drugi nacin, onda moramo spreciti da se izvrsi
        // ovaj programski kod u tim drugim situacijama
        // jer puca program zato sto nema current row index vrednost
        if (kliknuto == true)
        {
            RedniBrojSloga = dtgKredit.CurrentRow.Index;
            PrikaziLokacijuSloga(RedniBrojSloga);
            PrikaziTekuceg();
        }

    }

    private void dtgKredit_MouseDown(object sender, MouseEventArgs e)
    {
        // potrebno zbog regulisanja problema na CurrentCellChanged
        kliknuto = true;

    }
```

10.9.4. FILTRIRANJE PODATAKA

Ekranski deo za pretragu sadrzi text box za unos kriterijuma pretrage i tastera za filtriranje i prikaz svih podataka, snimanje I brisanje XML fajlova i snimanje Excel fajla.



` procedura za punjenje grida podacima iz dataset-a

```
private void NapuniGrid(System.Data.DataSet ds)
{
   // povezivanje grida sa datasetom
   dtgKredit.DataSource = ds.Tables[0];
   dtgKredit.Refresh();
}
```

` procedura – dogadjaj za pokretanje prikaz svih zapisa u gridu

```
private void btnSvi_Click(object sender, EventArgs e)
{
   pTabela = new clsSqlTabela (clsGlobal.Konekcija, "Kredit");
   dsKredit = pTabela.DajPodatke(SelectUpit);
   // povezivanje grida sa datasetom
   NapuniGrid(dsKredit);
   txtFilter.Text = "";

}
```

` procedura – dogadjaj za pokretanje prikaz filtriranih zapisa u gridu

NAPOMENA: Posebno obratiti pažnju na kriterijum filtriranja. Ako je filtriranje u odnosu na identifikaciona obeležja tipa JMBG, tada se koristi simbol „=", a kada je filtriranje prema nazivu (npr. prezimenu ili nazivu firme i slicno) može se koristiti «LIKE» operator i džoker simboli.

Primer:

```
«.....JMBG='" + txtKriterijumFiltera.Text + "'"
"......NazivFirme LIKE '%" + txtKriterijumFiltera.Text + "%'"
```

```
private void btnFiltriraj_Click(object sender, EventArgs e)
{
   pTabela = new clsSqlTabela(clsGlobal.Konekcija, "Kredit");
   // moze ovako:
```

```
        dsKredit = pTabela.DajPodatke("select Kredit.BrojRacuna, TipKredita.Naziv, Kredit.JMBG, Kredit.DatumPocetka,
Kredit.TrajanjeMeseci, Kredit.Kamata, Kredit.Iznos from Kredit inner join TipKredita on
Kredit.IdTipaKredita=TipKredita.Id where JMBG='" + txtFilter.Text  + "' order by Kredit.DatumPocetka desc");
        // povezivanje grida sa datasetom
        NapuniGrid(dsKredit);

    }
```

10.9.5. RAD SA XML I XLS FAJLOVIMA

Sa ekrana za pretragu podataka mogu se podaci eksportovati u XML ili XLS format podataka. Potrebno je da postoji globalna promenljiva na nivou forme tipa dataset koja se puni na tasterima SVI i Filtriraj, a vrednosti čitaju na tasterima za rad sa XML i XLS fajlovima radi eksportovanja.

` procedura – dogadjaj za rad sa XML

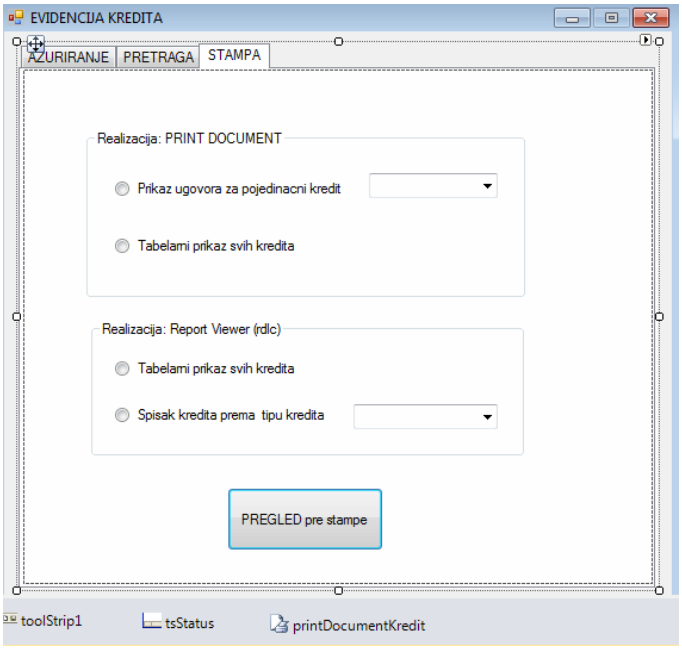
```
    private void btnObrisiXML_Click(object sender, EventArgs e)
    {
        clsXMLUtil.ObrisiXMLfajl("Krediti.XML");
        MessageBox.Show("Uspesno ste obrisali XML fajl sa spiskom kredita!");

    }

    private void btnSnimiXML_Click(object sender, EventArgs e)
    {
        string PorukaUspeha = clsXMLUtil.KreirajXMLfajl(dsKredit, true, "Krediti.XML", "KreditiShema.XML");
        MessageBox.Show(PorukaUspeha);


    }
```

` procedura - dogadjaj za rad sa Excelom

```
    private void btnExcel_Click(object sender, EventArgs e)
    {
        clsXLSutilALT.SnimiXLS(dsKredit, "Krediti.XLS");
        MessageBox.Show("Uspesno ste kreirali XLS fajl!");
    }
```

10.9.6. STAMPA

Ovaj deo ekrana daje mogucnost prikaza izvestaja koristeci Report viewer ili Print Document tehniku. Dati su prikazi u formi dokumenta sa tabelarnim prikazom svih podataka ili filtriranim prikazom pojedinacnog zapisa u formi obrasca sa podacima.



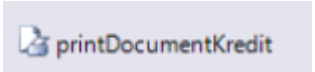
Na ovom ekranu se mogu birati načini rada sa izveštajima, kao i parametri rada – kriterijumi filtriranja podataka za izveštaje.

Taster print preview:

```
private void btnPrintPreview_Click(object sender, EventArgs e)
{
   if ((rdbPojedinacniUgovor.Checked) || (rdbTabelarniSvi.Checked))
   {
      PrintPreviewDialog objStampaIme = new PrintPreviewDialog();
      objStampaIme.Document = printDocumentKredit;
      objStampaIme.Width = 800;
      objStampaIme.Height = 600;
      objStampaIme.Top = 10;
      objStampaIme.Left = 10;
      objStampaIme.ShowDialog();
   }

   if (rdbTabelarniReport.Checked)
   {
      frmIzvestaj formaIzvestaj = new frmIzvestaj();
      formaIzvestaj.ShowDialog();
   }
}
```

10.9.6.1. KREIRANJE IZVESTAJA KORISTECI PRINT DOCUMENT

Da bismo radili sa izveštajima tipa Print Document, na ekransku formu postavljamo sa palete alata odgovarajući alat:



U nastavku je dat programski kod:

```
private void printDocumentKredit_PrintPage(object sender, System.Drawing.Printing.PrintPageEventArgs e)
{
   try
   {

      // ************** ZAGLAVLJE **********************
```

```
            // slika - logo firme
            e.Graphics.DrawImage(Image.FromFile("LOGO.bmp"), Rectangle.FromLTRB(30, 30, 130, 130));

            // podaci o firmi

            e.Graphics.DrawString("Velika probna banka",
           new Font("Tahoma", 14, FontStyle.Bold),
           Brushes.Black, e.MarginBounds.Left + 80,
           e.MarginBounds.Top - 80 + e.Graphics.MeasureString("Xxxx",
           new Font("Tahoma", 14, FontStyle.Bold),
           e.MarginBounds.Width).Height - 13);

            e.Graphics.DrawString("VPB",
              new Font("Tahoma", 14, FontStyle.Bold),
              Brushes.Black, e.MarginBounds.Left + 80,
              e.MarginBounds.Top - 60 + e.Graphics.MeasureString("Xxxx",
              new Font("Tahoma", 14, FontStyle.Bold),
              e.MarginBounds.Width).Height - 13);

            e.Graphics.DrawString("7. Jul bb",
             new Font("Tahoma", 14, FontStyle.Bold),
             Brushes.Black, e.MarginBounds.Left + 80,
             e.MarginBounds.Top - 40 + e.Graphics.MeasureString("Xxxx",
             new Font("Tahoma", 14, FontStyle.Bold),
             e.MarginBounds.Width).Height - 13);

            e.Graphics.DrawString("Zrenjanin",
             new Font("Tahoma", 14, FontStyle.Bold),
             Brushes.Black, e.MarginBounds.Left + 80,
             e.MarginBounds.Top - 20 + e.Graphics.MeasureString("Xxxx",
             new Font("Tahoma", 14, FontStyle.Bold),
             e.MarginBounds.Width).Height - 13);

            e.Graphics.DrawString("Telefon: 023/525-989",
            new Font("Tahoma", 14, FontStyle.Bold),
            Brushes.Black, e.MarginBounds.Left + 80,
            e.MarginBounds.Top + e.Graphics.MeasureString("Xxxx",
            new Font("Tahoma", 14, FontStyle.Bold),
            e.MarginBounds.Width).Height - 13);


            // *************** NASLOV

            if (rdbPojedinacniUgovor.Checked)
            {
                e.Graphics.DrawString("UGOVOR ZA KREDIT PO RACUNU BROJ:" + cmbBrojRacunaFilter.Text,
           new Font("Tahoma", 16, FontStyle.Bold),
           Brushes.Black, e.MarginBounds.Left + 80,
           e.MarginBounds.Top + 60 + e.Graphics.MeasureString("Xxxx",
           new Font("Tahoma", 16, FontStyle.Bold),
           e.MarginBounds.Width).Height - 13);
            }


            if (rdbSpisakPoTipu.Checked)
            {
                e.Graphics.DrawString("SPISAK KREDITA PREMA TIPU:" +  cmbFilterTipKredita.Text,
            new Font("Tahoma", 16, FontStyle.Bold),
            Brushes.Black, e.MarginBounds.Left + 80,
            e.MarginBounds.Top + 60 + e.Graphics.MeasureString("Xxxx",
            new Font("Tahoma", 16, FontStyle.Bold),
            e.MarginBounds.Width).Height - 13);
            }

            if (rdbTabelarniSvi.Checked)
            {
                e.Graphics.DrawString("SPISAK SVIH KREDITA",
            new Font("Tahoma", 16, FontStyle.Bold),
            Brushes.Black, e.MarginBounds.Left + 80,
            e.MarginBounds.Top + 60 + e.Graphics.MeasureString("Xxxx",
            new Font("Tahoma", 16, FontStyle.Bold),
```

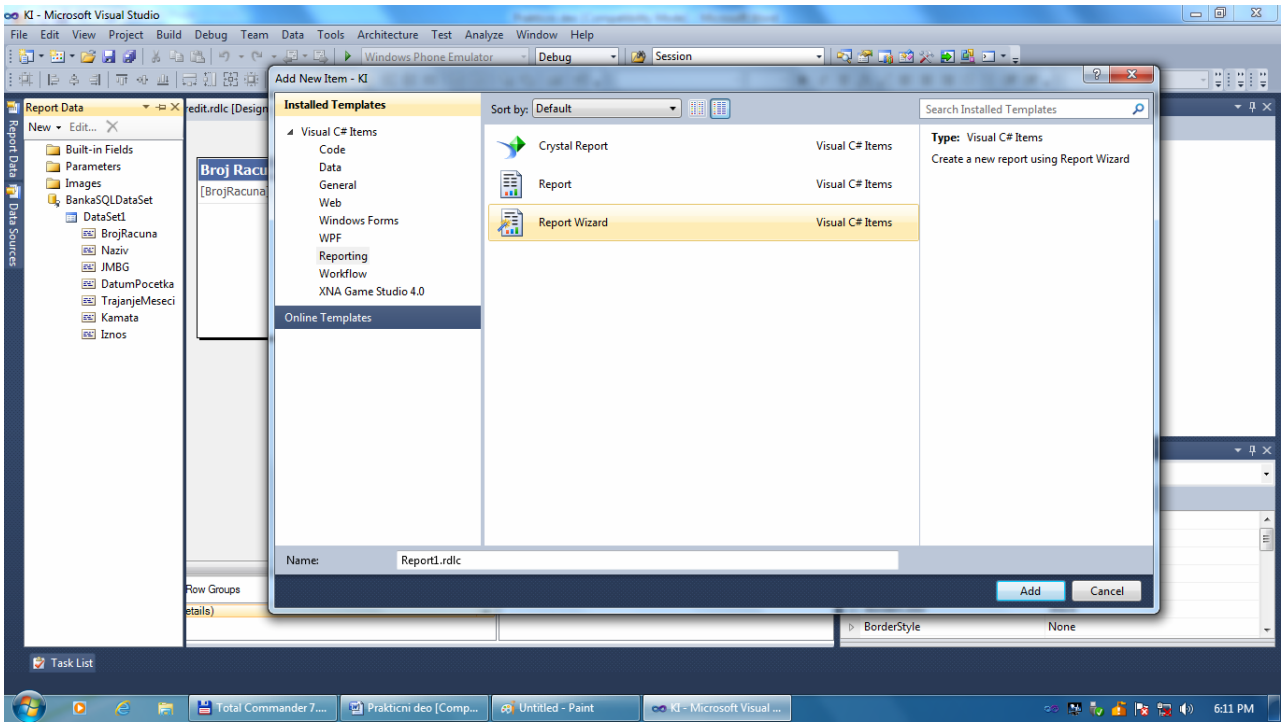
```
            e.MarginBounds.Width).Height - 13);
              }

              // ------------ LINIJA

              e.Graphics.DrawLine(Pens.Blue, e.MarginBounds.Left+ 60, e.MarginBounds.Top + 120, 700,
e.MarginBounds.Top + 120);


              // ###################################################
              // *************** GLAVNI DEO  **********************
               // ###################################################

              if (rdbPojedinacniUgovor.Checked)
              {

              // preuzimanje podataka

                  clsSqlTabela pTabelaKredit = new clsSqlTabela(clsGlobal.Konekcija, "Kredit");
            System.Data.DataSet dsKreditStampa = pTabelaKredit.DajPodatke("select Kredit.BrojRacuna, TipKredita.Naziv,
Kredit.JMBG, Kredit.DatumPocetka, Kredit.TrajanjeMeseci, Kredit.Kamata, Kredit.Iznos from Kredit inner join TipKredita
on Kredit.IdTipaKredita=TipKredita.Id where Kredit.BrojRacuna='" + cmbBrojRacunaFilter.Text + "'" );

              // pojedinacno svako polje

            e.Graphics.DrawString("Broj racuna:  " + dsKreditStampa.Tables[0].Rows[0].ItemArray[0].ToString(),
              new Font("Tahoma", 16, FontStyle.Bold),
              Brushes.Black, e.MarginBounds.Left + 90,
              e.MarginBounds.Top + 150 + e.Graphics.MeasureString("Xxxx",
              new Font("Tahoma", 16, FontStyle.Bold),
              e.MarginBounds.Width).Height - 13);

                   e.Graphics.DrawString("Tip kredita:  " + dsKreditStampa.Tables[0].Rows[0].ItemArray[1].ToString () ,
              new Font("Tahoma", 16, FontStyle.Bold),
              Brushes.Black, e.MarginBounds.Left + 90,
              e.MarginBounds.Top + 180 + e.Graphics.MeasureString("Xxxx",
              new Font("Tahoma", 16, FontStyle.Bold),
              e.MarginBounds.Width).Height - 13);

                  e.Graphics.DrawString("JMBG:  " + dsKreditStampa.Tables[0].Rows[0].ItemArray[2].ToString(),
              new Font("Tahoma", 16, FontStyle.Bold),
              Brushes.Black, e.MarginBounds.Left + 90,
              e.MarginBounds.Top + 210 + e.Graphics.MeasureString("Xxxx",
              new Font("Tahoma", 10, FontStyle.Bold),
              e.MarginBounds.Width).Height - 13);

                  e.Graphics.DrawString("Datum pocetka:  " + dsKreditStampa.Tables[0].Rows[0].ItemArray[3].ToString(),
              new Font("Tahoma", 16, FontStyle.Bold),
              Brushes.Black, e.MarginBounds.Left + 90,
              e.MarginBounds.Top + 240 + e.Graphics.MeasureString("Xxxx",
              new Font("Tahoma", 10, FontStyle.Bold),
              e.MarginBounds.Width).Height - 13);

                  e.Graphics.DrawString("Trajanje (meseci):  " +
dsKreditStampa.Tables[0].Rows[0].ItemArray[4].ToString(),
              new Font("Tahoma", 16, FontStyle.Bold),
              Brushes.Black, e.MarginBounds.Left + 90,
              e.MarginBounds.Top + 270 + e.Graphics.MeasureString("Xxxx",
              new Font("Tahoma", 10, FontStyle.Bold),
              e.MarginBounds.Width).Height - 13);

                  e.Graphics.DrawString("Kamata:  " + dsKreditStampa.Tables[0].Rows[0].ItemArray[5].ToString(),
            new Font("Tahoma", 16, FontStyle.Bold),
            Brushes.Black, e.MarginBounds.Left + 90,
            e.MarginBounds.Top + 300 + e.Graphics.MeasureString("Xxxx",
            new Font("Tahoma", 10, FontStyle.Bold),
            e.MarginBounds.Width).Height - 13);

                  e.Graphics.DrawString("Iznos:  " + dsKreditStampa.Tables[0].Rows[0].ItemArray[6].ToString(),
            new Font("Tahoma", 16, FontStyle.Bold),
            Brushes.Black, e.MarginBounds.Left + 90,
```

```
            e.MarginBounds.Top + 330 + e.Graphics.MeasureString("Xxxx",
            new Font("Tahoma", 10, FontStyle.Bold),
            e.MarginBounds.Width).Height - 13);


                e.Graphics.DrawString("POTPIS KLIJENTA:",
          new Font("Tahoma", 16, FontStyle.Bold),
          Brushes.Black, e.MarginBounds.Left + 90,
          e.MarginBounds.Top + 480 + e.Graphics.MeasureString("Xxxx",
          new Font("Tahoma", 10, FontStyle.Bold),
          e.MarginBounds.Width).Height - 13);

                // ------------ LINIJA

                e.Graphics.DrawLine(Pens.Black, 200, e.MarginBounds.Top + 580, 450, e.MarginBounds.Top + 580);

            };


            if (rdbSpisakPoTipu.Checked)
            {

            };



          if (rdbTabelarniSvi.Checked)
            {

            // ************ NASLOVNA LINIJA TABELE ****************

                e.Graphics.DrawString("BROJ RACUNA   TIP KREDITA  JMBG   DATUM POCETKA" ,
            new Font("Tahoma", 10, FontStyle.Bold),
            Brushes.Black, e.MarginBounds.Left + 30,
            e.MarginBounds.Top + 150 + e.Graphics.MeasureString("Xxxx",
            new Font("Tahoma", 10, FontStyle.Bold),
            e.MarginBounds.Width).Height - 13);

            // *************** PREUZIMANJE PODATAKA ******************

            clsSqlTabela pTabelaKredit2 = new clsSqlTabela(clsGlobal.Konekcija, "Kredit");
            System.Data.DataSet dsKreditStampa2 = pTabelaKredit2.DajPodatke("select Kredit.BrojRacuna,
TipKredita.Naziv, Kredit.JMBG, Kredit.DatumPocetka, Kredit.TrajanjeMeseci, Kredit.Kamata, Kredit.Iznos from Kredit
inner join TipKredita on Kredit.IdTipaKredita=TipKredita.Id order by Kredit.BrojRacuna");

            // ************* PRIKAZ SVIH PODATAKA ZA JEDAN SLOG U JEDNOM REDU

            for (int i = 0; i < dsKreditStampa2.Tables[0].Rows.Count; i++)
            {
                string BrojRacuna = dsKreditStampa2.Tables[0].Rows[i].ItemArray[0].ToString();
                string TipKredita = dsKreditStampa2.Tables[0].Rows[i].ItemArray[1].ToString();
                string JMBG = dsKreditStampa2.Tables[0].Rows[i].ItemArray[2].ToString();
                string DatumPocetka = dsKreditStampa2.Tables[0].Rows[i].ItemArray[3].ToString();
                string TrajanjeMeseci = dsKreditStampa2.Tables[0].Rows[i].ItemArray[4].ToString();
                string Kamata = dsKreditStampa2.Tables[0].Rows[i].ItemArray[5].ToString();
                string Iznos = dsKreditStampa2.Tables[0].Rows[i].ItemArray[6].ToString();



                e.Graphics.DrawString(BrojRacuna + "   " + TipKredita + "    " +  JMBG  + "    " +  DatumPocetka,
            new Font("Tahoma", 10, FontStyle.Bold),
            Brushes.Black, e.MarginBounds.Left + 30,
            e.MarginBounds.Top + 180 + i*30 + e.Graphics.MeasureString("Xxxx",
            new Font("Tahoma", 10, FontStyle.Bold),
            e.MarginBounds.Width).Height - 13);
            }



            };
```

```
        }
        catch (Exception ex)
        {
            MessageBox.Show("Greska:" + ex.Message);
            return;
        }
    }

  }
}
```

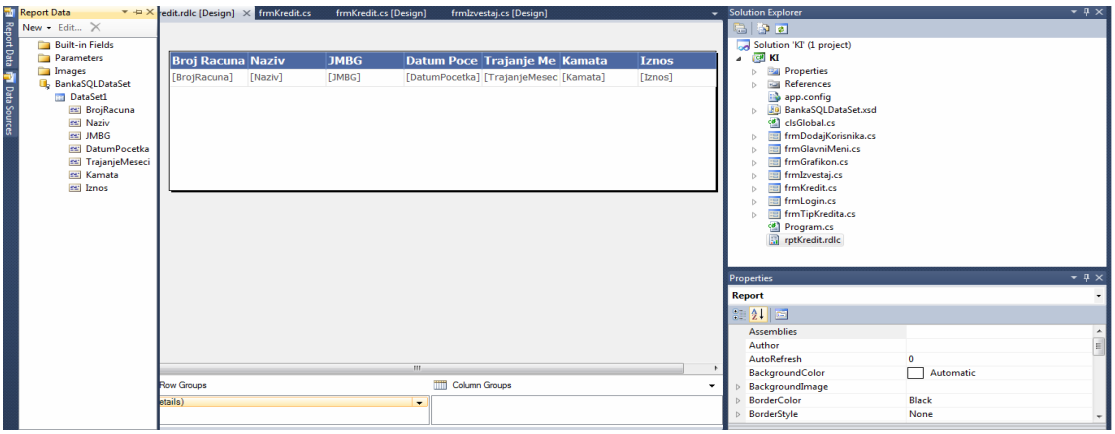
10.9.6.2. Rad sa rdlc izveštajima

Dizajn rdlc izvestaja radimo koristeci opciju sa glavnog menija Project – Add New Item, Reporting, Report Wizard.



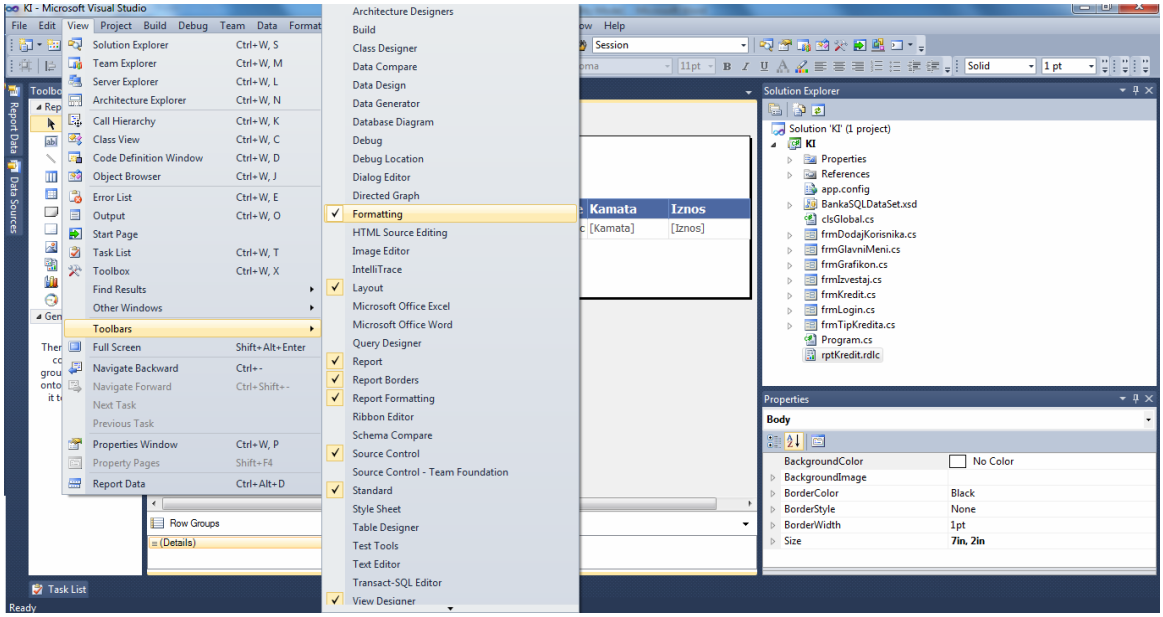
U okviru Wizarda konektujemo se iz aplikacije direktno sa bazom podataka, tako da dobijamo na formi za rad sa izvestajem binding source, adapter i dataaset.

Izgled jednostavnog dizajna tabelarnog prikaza podataka iz jedne tabele:

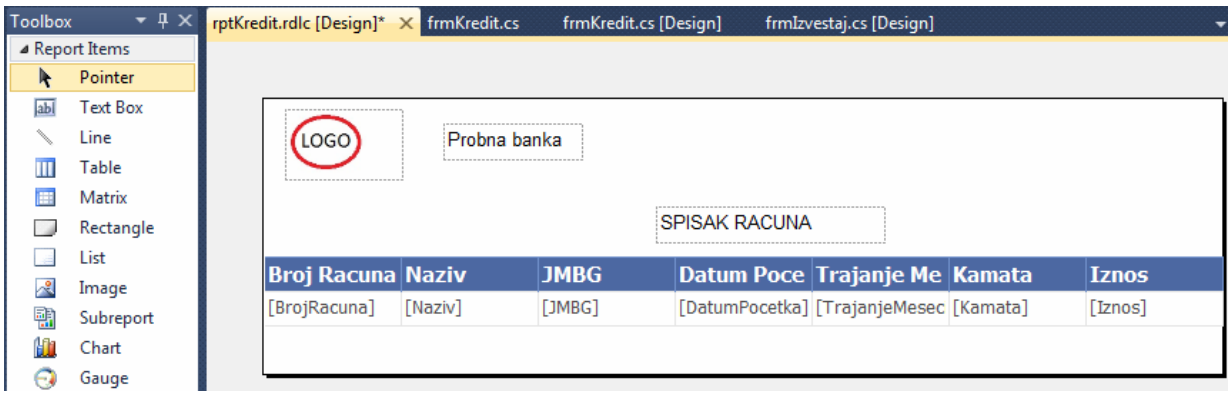


NAPOMENA:
Ukoliko želimo da prikažemo podatke iz spoja tabela, kreiraćemo i snimiti sql upit u samu bazu podataka pod nekim imenom, a zatim u okviru Wizard-a koristiti konekciju na pogled (upit), gde su uključeni podaci iz spoja tabela.

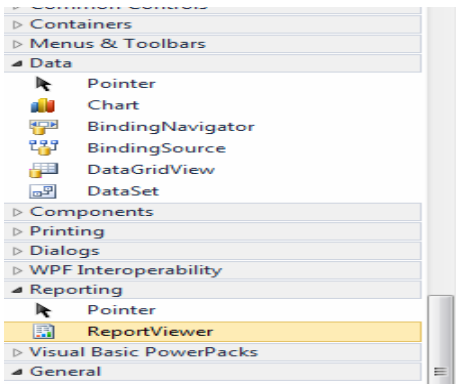
Dodavanje palete za vizualno doterivanje izveštaja:



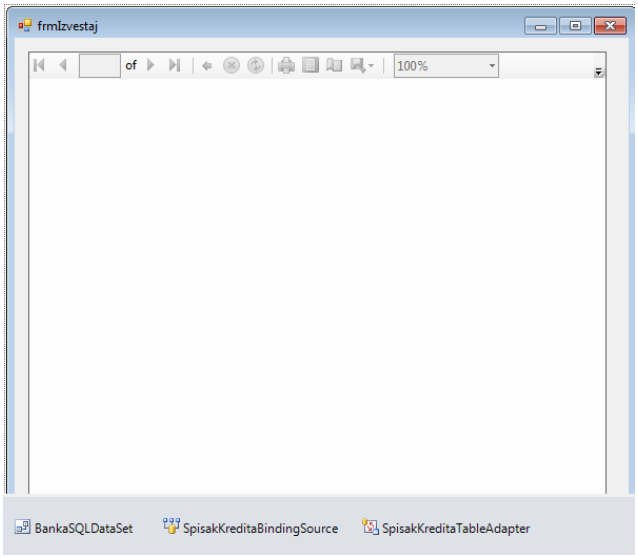
Dodajemo elemente dizajna:



Fajl se pokreće u okviru posebnog ekrana za rad sa izveštajima. Kad postavimo kontrolu (iz odeljka Toolboxa – reporting) – „Report viewer" na formu:



Odmah dobijamo i dijalog prozor gde se konektujemo na ranije kreiran rdlc fajl izveštaja.



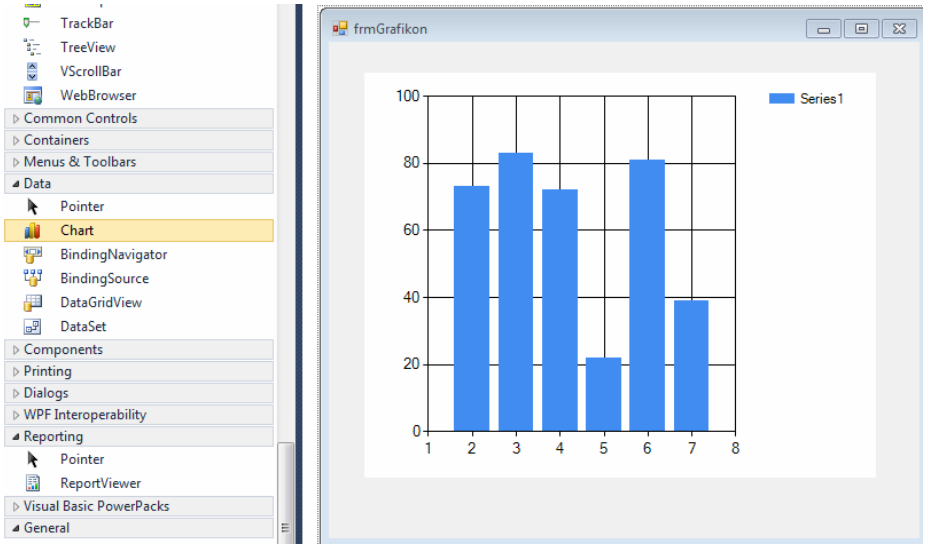
Programski kod forme za rad za izvestajima:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace KI
{
   public partial class frmIzvestaj : Form
   {
      public frmIzvestaj()
      {
         InitializeComponent();
      }

      private void frmIzvestaj_Load(object sender, EventArgs e)
      {
         // TODO: This line of code loads data into the 'BankaSQLDataSet.SpisakKredita' table. You can move, or
remove it, as needed.
         this.SpisakKreditaTableAdapter.Fill(this.BankaSQLDataSet.SpisakKredita);

         this.reportViewer1.RefreshReport();
         this.reportViewer1.RefreshReport();
      }
   }
}
```

10.10. Rad sa grafikonima

Da bismo radili sa grafikonima, podaci moraju biti statisticki obradjeni, tj, moramo kreirati SQL upit (npr. Pogled u MS SQL Serveru) kojim se dobijaju sumarni podaci, primenom nekih od agregatnih funkcija SQL jezika. U ovom primeru kreiran je pogled koji ima naziv KreditiPotipu koji grupise podatke o kreditima po tipu kredita, a zatim prebrojava broj kredita po odgovarajucem tipu.

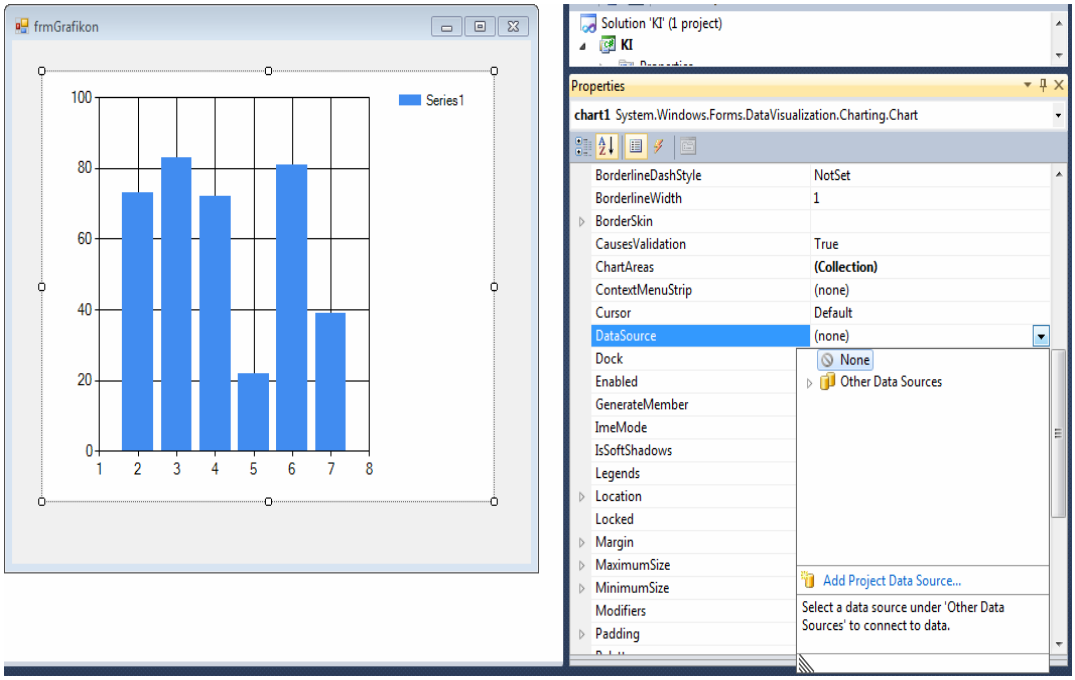
Sa palete postavljamo chart kontrolu:



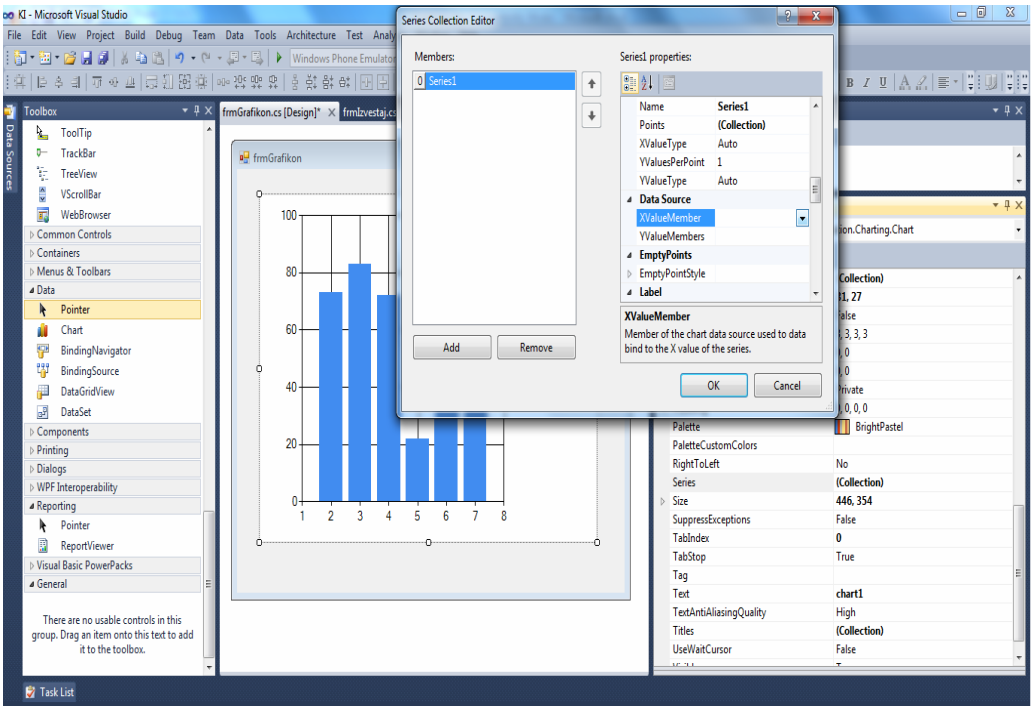
Programski kod za rad sa izveštajima:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
//
using SqlDBUtils;

namespace KI
{
   public partial class frmGrafikon : Form
   {
      public frmGrafikon()
      {
         InitializeComponent();
      }

      private void frmGrafikon_Load(object sender, EventArgs e)
      {
         clsSqlTabela objTabela = new clsSqlTabela (clsGlobal.Konekcija, "KreditiPoTipu");
         System.Data.DataSet dsPodaci = objTabela.DajPodatke ("select * from KreditiPoTipu");
         chart1.DataSource = dsPodaci.Tables[0];
         chart1.Series[0].XValueMember = "Tip kredita";
         chart1.Series[0].YValueMembers = "Ukupan broj racuna";
         chart1.Series[0].Name = "Ukupan broj racuna prema tipu kredita";
      }
   }
}
```

Umesto programskog koda za konekciju dinamicki, možemo povezati grafikon direktno sa bazom podataka – DATA SOURCE.



A zatim u Series – collection, Data source x i y value member podesiti izvor podataka iz odgovarajucih polja sql upita.



10.11. Forma Master-Detail

Forme tipa master-detail imaju mogućnost istovremenog unosa i snimanja podataka u 2 tabele baze podataka – zaglavlje i stavke. Tipičan primer je forma za ažuriranje podataka o računima, fakturama i slično.

PRVI PRIMER - FAKTURA (izgled dokumenta):

FAKTURA

Broj fakture _____

Datum fakturisanja _____

Datum isporuke _____

Kupac:

RB	Artikl	Količina	JM	Osnovna cena	PDV	CENA

Vrednost bez PDV-a

PDV

Vrednost sa PDV-om

Slovima:

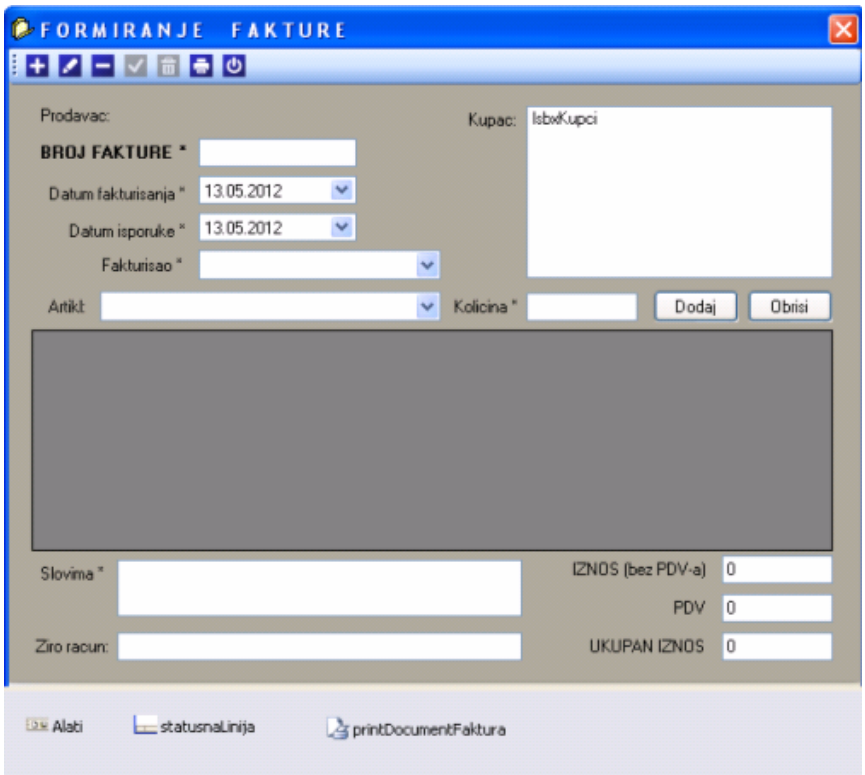
Uplatu možete izvršiti na žiro račun:

NAČIN PLAĆANJA - ŽIRALNO
ROK PLAĆANJA - 15 DANA
Za ne izmirivanje dugovanja nadležan je Osnovni Sud u Zrenjaninu.

_____ MP _____ MP

Fakturisao **Primio**

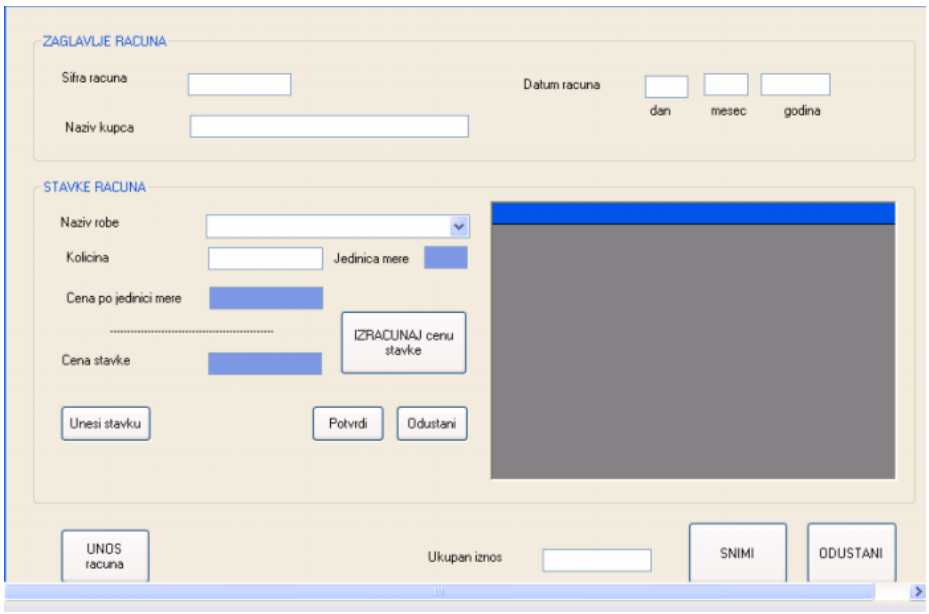
Ekranska forma:



Programski kod za realizaciju transakcije za upis podataka u master –detail formu za azuriranje faktura dat je u nastavku:

```
private void tsbPotvrdi_Click(object sender, EventArgs e)
        if (Transakcija == "UNOS")
        {
            OleDbConnection konekcija = new OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=Fakturisanje2012.mdb");
            konekcija.Open();
            OleDbCommand komanda = konekcija.CreateCommand();
            OleDbTransaction transakcija = null;
            try
            {
                transakcija = konekcija.BeginTransaction();
                komanda.Transaction = transakcija;
                komanda.CommandText = "Insert into Faktura Values ('" + txtBrojFakture.Text + "', '" +
dtDatumFakturisanja.Value.ToShortDateString() + "', '" + dtDatumIsporuke.Value.ToShortDateString() + "', '" +
dsProdavac.Tables[0].Rows[0].ItemArray[0].ToString() + "', '" +
dsKupci.Tables[0].Rows[lsbxKupci.SelectedIndex].ItemArray[0].ToString() + "', " +
dsZaposleni.Tables[0].Rows[cmbFakturisao.SelectedIndex].ItemArray[0].ToString() + ", " + txtIznosBezPDVa.Text + ",
" + txtPDV.Text + ", " + txtUkupanIznos.Text + ", '" + txtSlovima.Text + "', 'Ziralno', '15 dana')";

                komanda.ExecuteNonQuery();

                for (int i = 0; i < RedniBroj; i++)
                {
                    komanda.CommandText = "Insert into [Stavka fakture] Values ('" + txtBrojFakture.Text + "', " +
dsStavke.Tables[0].Rows[i].ItemArray[0].ToString() + ", " + dsStavke.Tables[0].Rows[i].ItemArray[2].ToString() + ", "
+ dsStavke.Tables[0].Rows[i].ItemArray[4].ToString() + ", " + dsStavke.Tables[0].Rows[i].ItemArray[7].ToString() +
", " + dsStavke.Tables[0].Rows[i].ItemArray[8].ToString() + ")";
                    komanda.ExecuteNonQuery();
                }
                transakcija.Commit();
                MessageBox.Show("Uspešno snimljena faktura.", "Informacija korisniku", MessageBoxButtons.OK,
MessageBoxIcon.Information);
```

DRUGI PRIMER – račun:



U drugom primeru data je mogucnost da se podaci stavke privremeno snimaju u pomocnu bazu podataka ili XML, a nakon zavrsetka celog racuna snimaju u bazu podataka.

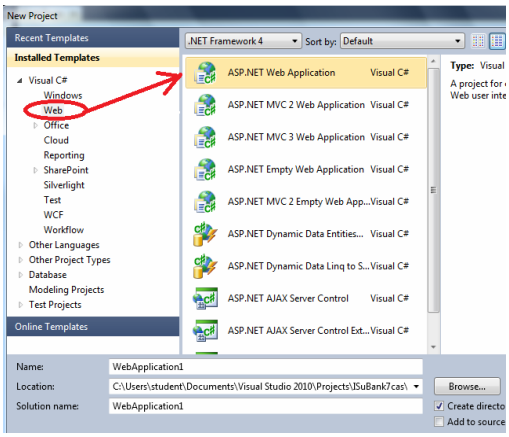
```
private void btnDodajStavku_Click(object sender, EventArgs e)
      {
         // cuvanje zaglavlja u XML
         // ----------brisanje
         System.IO.FileInfo pomfile = new System.IO.FileInfo("Zaglavlje.XML");
         pomfile.Delete();
         // ----------upis
         dsZaglavlje.WriteXml("Zaglavlje.XML");

         //-------------------------
         ObrisiKontroleStavki();
         cmbNazivRobe.Enabled = true;
         txtKolicina.Enabled = true;
         NapuniComboNazivRobe();
         cmbNazivRobe.Focus();
      }


    private void btnPotvrdiStavku_Click(object sender, EventArgs e)
      {
         // ------------------------------------------------
         // -------------- Dodavanje PODATAKA O Stavkama RACUNA -------
         // ------------------------------------------------

         // -------------- na nivou dataseta ----------------------

         // prethodno je kreiran objKonekcija i objTabela
         // kao globalni


         //Dodavanje na nivou eksternog dataseta
         DataRow row = dsStavke.Tables[0].NewRow();
         row["SifraRacuna"] = int.Parse(txtSifraRacuna.Text);
         // datumska vrednost

         row["SifraRobe"] = DajSifruRobe(cmbNazivRobe.Text);

         row["Kolicina"] = float.Parse(txtKolicina.Text);
         row["CenaStavke"] = float.Parse(txtCenaStavke.Text);
         dsStavke.Tables[0].Rows.Add(row);

         // ---------------- prikaz u gridu ------------------
         dtgStavke.DataSource = dsStavke.Tables[0];
         dtgStavke.Refresh();
         // ---------------- snimanje u XML privremeno
         // ----------brisanje
         System.IO.FileInfo pomfile = new System.IO.FileInfo("Stavke.XML");
         pomfile.Delete();
         // ----------upis
         dsStavke.WriteXml("Stavke.XML");

         // ------------------ sabiranje ukupnog iznosa
         UkupanIznos = UkupanIznos + float.Parse(txtCenaStavke.Text);
         txtUkupanIznos.Text = UkupanIznos.ToString();
      }


private void btnSnimiRacun_Click(object sender, EventArgs e)
      {
         // ------------------------------------------------
         // -------------- UPIS PODATAKA O ZAGLAVLJU RACUNA -------
         // ------------------------------------------------

         UcitajZaglavlje();

         //Dodavanje na nivou dataseta - generise HasChanges
         DataRow row = objTabelaRacun.dsDataSet.Tables[0].NewRow();
         row["SifraRacuna"] = int.Parse (txtSifraRacuna.Text);
         // datumska vrednost
```

```
         // - formatiranje uvek u americkom stilu
         string datumAmer = "#" + txtMesec.Text + "/" + txtDan.Text + "/" + txtGodina.Text + "#";
         System.IFormatProvider format = new System.Globalization.CultureInfo("en-US", true);

         row["DatumRacuna"] = DateTime.Parse(datumAmer, format);
         // realan broj - transformacija stringa u realan broj sa float.parse

        row["NazivKupca"] = txtNazivKupca.Text;

        row["UkupanIznos"] = float.Parse(txtUkupanIznos.Text);
         objTabelaRacun.dsDataSet.Tables[0].Rows.Add(row);

         // dodavanje na nivou sql-a i baze podataka
         objTabelaRacun.DeleteUpit = "";
         // ako je u Regional Settings Americki stil koristimo datumAmer formatiran string
         string upitRacun = "INSERT INTO RACUN (SifraRacuna, DatumRacuna, NazivKupca, UkupanIznos) VALUES (" +
txtSifraRacuna.Text + ", " + datumAmer + ", '" + txtNazivKupca.Text  + "', " + txtUkupanIznos.Text  + ")";
         objTabelaRacun.InsertUpit = upitRacun;
         objTabelaRacun.UpdateUpit = "";
         bool uspeh = objTabelaRacun.UpisiUBazu(); //

         // ------------------------------------------------
         // UPIS PODATAKA O STAVKAMA
         // - vec se nalaze u objTabelaStavke.dsDataSet.Tables[0].Rows kolekciji
         // - znaci, radimo samo donji deo, odnosno sql i bazu
         // ------------------------------------------------
         string upitStavke ="";
         bool uspehStavke=false;
         objTabelaStavke.DeleteUpit = "";
         objTabelaStavke.UpdateUpit = "";
         string pomSifraRacuna = "";
         string pomSifraRobe="";
         string pomKolicina = "";
         string pomCenaStavke = "";
         // idemo kroz for ciklus nad privremenim datasetom dsStavke
         // da bi za svaku stavku napravili insert into
         for (int i = 0; i < dsStavke.Tables[0].Rows.Count; i++)
         {

            // dataset iz klase treba da dobije HAS CHANGES stanje
            // tako da bi moglo da se izvrsi snimanje
            DataRow rowStavke = objTabelaStavke.dsDataSet.Tables[0].NewRow();
            rowStavke["SifraRacuna"] = int.Parse (dsStavke.Tables[0].Rows[i].ItemArray[0].ToString());
            // datumska vrednost

            rowStavke["SifraRobe"] = dsStavke.Tables[0].Rows[i].ItemArray[1].ToString();

            rowStavke["Kolicina"] = float.Parse (dsStavke.Tables[0].Rows[i].ItemArray[2].ToString());
            rowStavke["CenaStavke"] = float.Parse(dsStavke.Tables[0].Rows[i].ItemArray[3].ToString());
            objTabelaStavke.dsDataSet.Tables[0].Rows.Add(rowStavke);

            pomSifraRacuna = dsStavke.Tables[0].Rows[i].ItemArray[0].ToString();
            pomSifraRobe = dsStavke.Tables[0].Rows[i].ItemArray[1].ToString();
            pomKolicina = dsStavke.Tables[0].Rows[i].ItemArray[2].ToString();
            pomCenaStavke = dsStavke.Tables[0].Rows[i].ItemArray[3].ToString();
            upitStavke = "INSERT INTO Stavke (SifraRacuna, SifraRobe, Kolicina, CenaStavke) VALUES (" +
pomSifraRacuna  + ", '" + pomSifraRobe  + "', " + pomKolicina  + ", " + pomCenaStavke  + ")";
            objTabelaStavke.InsertUpit = upitStavke;
            MessageBox.Show(upitStavke);
            uspehStavke = objTabelaStavke.UpisiUBazu();
         }

         MessageBox.Show("Uspesno snimljeni podaci o racunu i stavkama racuna!");
      }
```

11. IMPLEMENTACIJA WEB APLIKACIJE

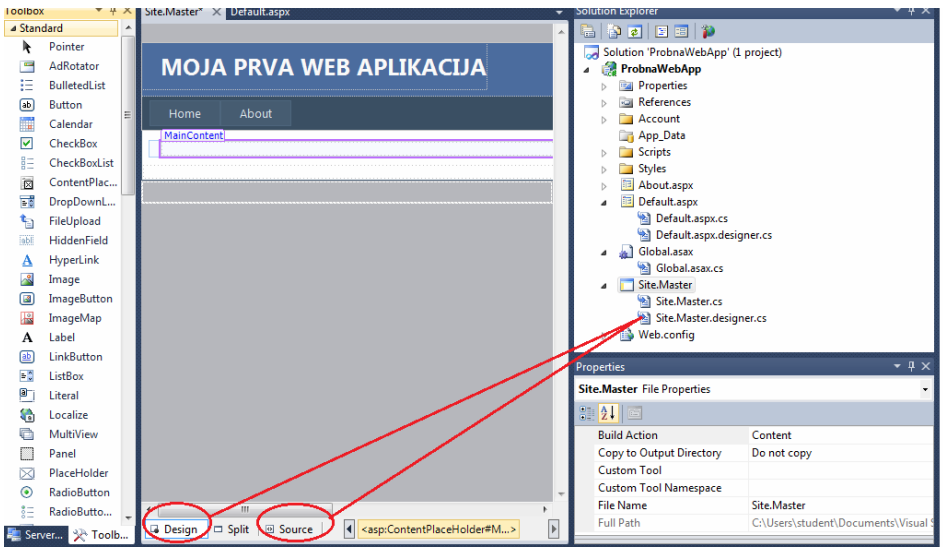
11.1. Radno okruženje i kreiranje ASPX aplikacije

1. Kreiranje ASPX aplikacije kao tipa projekta



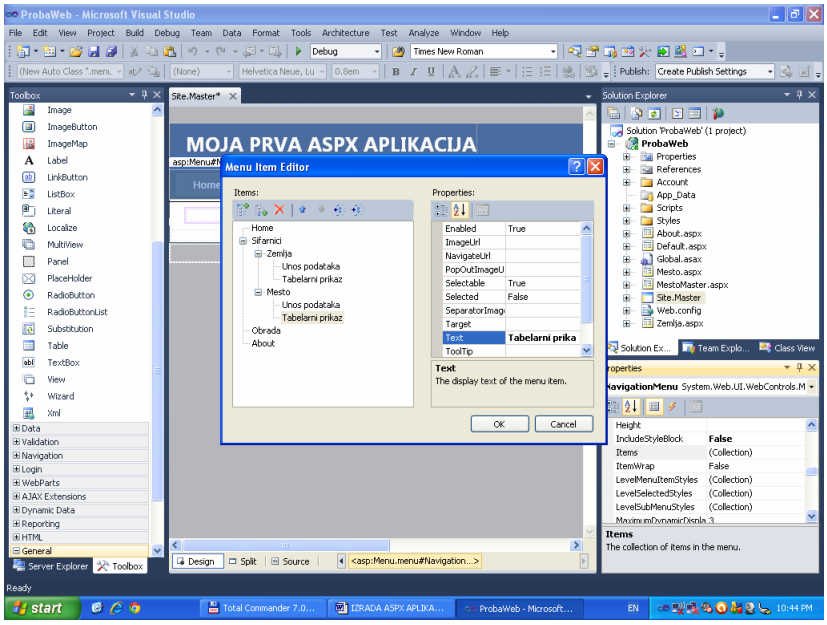
2. Automatski su kreirane osnovne stranice:
   - About.aspx - podaci o autoru
   - Default.aspx – prva stranica koja se pokreće nakon pokretanja aplikacije
   - Global.asax – mesto definisanja globalnih promenljivih i procedura na nivou cele aplikacije, a takodje i podesavanje koda koji se pokreće na pocetku izvršavanja aplikacije
   - Site.Master – zaglavlje svake stranice – kada dodajemo nove stranice, one mogu da budu obične ili da svaka ima zajedničko zaglavlje sa nazivom aplikacije i glavnim menijem, koji se nalazi u ovom fajlu Site.Master
   - Web.config – podaci za konfiguraciju aplikacije.

Svaka stranica ima design - vizualni dizajn stranice i source deo - programski kod koji se odnosi na izgled dizajna i automatski se generiše na osnovu korisšćenja alata za vizualni dizajn sa toolbox-a (npr. Default.aspx.designer.cs). U posebnom fajlu nalazi se izvorni kod programa koji se odnosi na programiranje događaja koji mi programiramo (Default.aspx.cs).

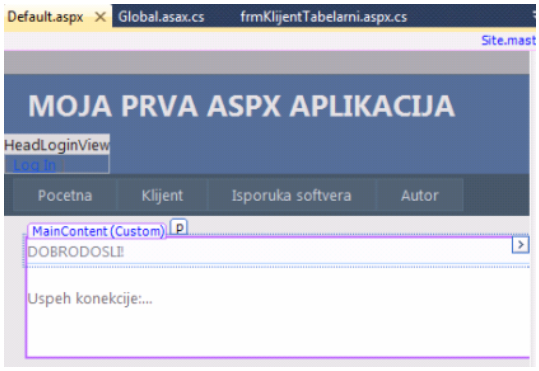


3. Editovanje dizajna **Site.Master** fajla koji automatski odmah sadrži glavni meni aplikacije sa opcijama Home i About.

- Menjamo naslov koji ce se pojavljivati na svakoj formi.
- Dodavanje nove stavke menija - Navigation Menu - Items collection editor.



3. Editovanje dizajna prve stranice koja se pokreće, tj. **Default.aspx**.



5. Editovanje dizajna**About.aspx** stranice – postavljamo podatke o autoru.

6. Build solution. Pokrecemo aplikaciju.

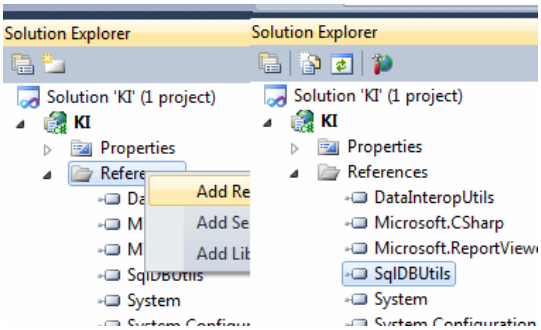
- Pokrece se aplikacija na localhost portu.

7. Za pokretanje jedne stranice iz druge možemo na nekom događaju (npr. button click) postaviti sledeći programski kod:

```
Server.Transfer("NazivForme.aspx");
```

11.2. Konekcija na bazu podataka putem klasa sloja podataka

1. Da bismo omogucili pristup podacima putem programskog koda (bez direktne konekcije sa forme), koristimo kreirane klase sloja podataka. Dodajemo reference na dll-ove u Solution Exploreru, odeljak References, desni taster – Add reference.



2. Da bismo omogućili uspostavljanje konekcija ka bazi podataka jednom na nivou cele aplikacije, koristimo Global.asaxfajl, tacnije, pišemo programski kod u okviru Global.asax.cs. Na pocetku u using sekciji postavljamo nazive biblioteka klasa za rad sa SqlServer bazom podataka SqlDBUtils i za rad sa XML fajlovima DataInteropUtils. S obzirom da je Global zapravo klasa, ima atribute i metode. Postavljamo u sekciju atributa globalne promenljive, a zatim u sekciju metoda proceduru za otvaranje konekcije, koja se pokrece u okviru Application_Start dogadjaja. Ovaj događaj se aktivira kada se aplikacija pokrene.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Security;
using System.Web.SessionState;
//
using SqlDBUtils;
using DataInteropUtils;

namespace KI
{
   public class Global : System.Web.HttpApplication
   {
      #region Atributi

      public static clsSqlKonekcija Konekcija;
      public static bool uspesnoOtvorenaKonekcija;
      public static string Putanja = "C:\\Users\\student\\Documents\\Visual Studio 2010\\Projects\\IS2_16 cas\\WEB
sql\\ki\\KI\\App_Data";
      #endregion


      #region Nase Procedure

      public static bool OtvoriKonekcijuDoBazePodataka()
      // ovde se procedura OtvoriKonekciju zove isto kao i metoda klase konekcija
      // to je dozvoljeno
      {
         System.Data.DataSet dsParametri = clsXMLUtil.UcitajXMLuDataset(Putanja + "\\PodesavanjaSQL.XML");
         //
         string pNazivSQLinstance = dsParametri.Tables[0].Rows[0].ItemArray[0].ToString();
         string pNazivBaze = dsParametri.Tables[0].Rows[0].ItemArray[1].ToString();
         string pPutanjaBaze = dsParametri.Tables[0].Rows[0].ItemArray[2].ToString();

         Konekcija = new clsSqlKonekcija(pNazivSQLinstance, pPutanjaBaze, pNazivBaze);
```

```
        bool uspeh = Konekcija.OtvoriKonekciju();
        return uspeh;
    }

    #endregion



    void Application_Start(object sender, EventArgs e)
    {
        uspesnoOtvorenaKonekcija = OtvoriKonekcijuDoBazePodataka();

    }
```

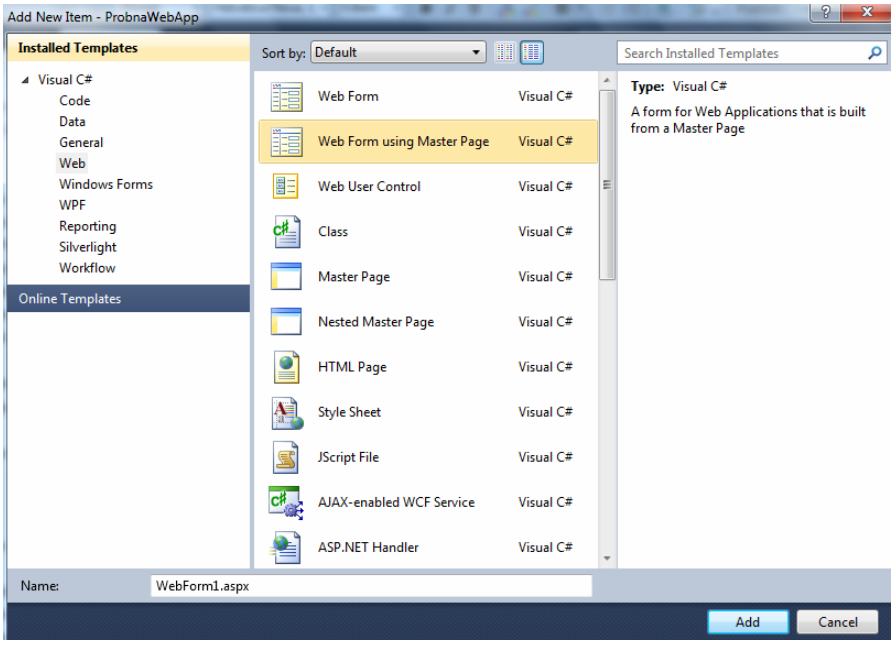
3. U okviru Default.aspx forme proveravamo uspeh konekcije.

Na prazan prostor forme kliknemo i dobijamo prostor za pisanje programskog koda dogadjaja Page_Load, koji je sličan sa Form Load kod Windows aplikacija. Ukoliko se uspešno ostvari konekcija (čitamo vrednost iz globalne promenljive uspesnoOtvorenaKonekcija, odnosno atributa klase Global), u labeli ce biti ispisana odgovarajuća poruka.

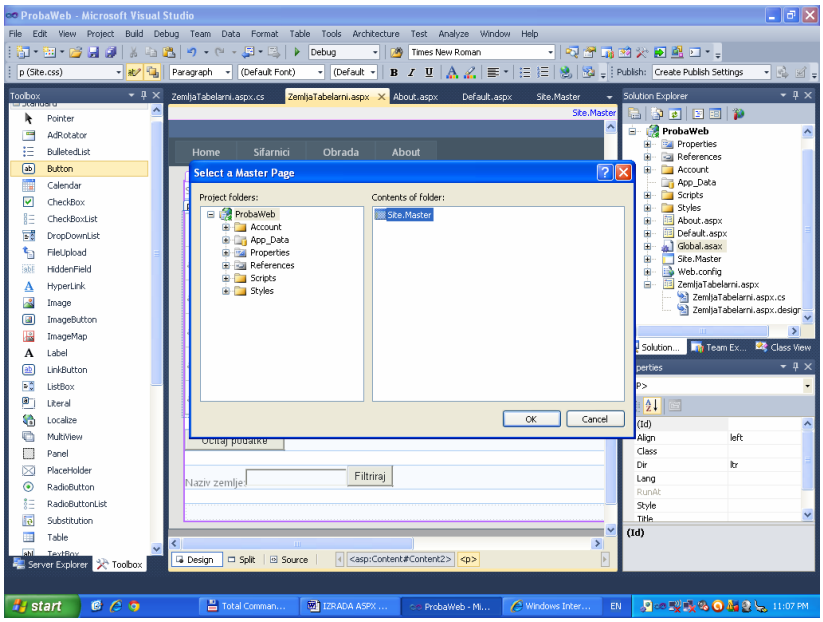
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace KI
{
    public partial class _Default : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            if (Global.uspesnoOtvorenaKonekcija)
            {
                lblStatusKonekcije.Text = "Uspesno otvorena konekcija!";
            }
            else
            {
                lblStatusKonekcije.Text = "NEUSPEH otvaranja konekcije!";
            }
        }
    }
}
```

11.3. Kreiranje forme i povezivanje sa menijem

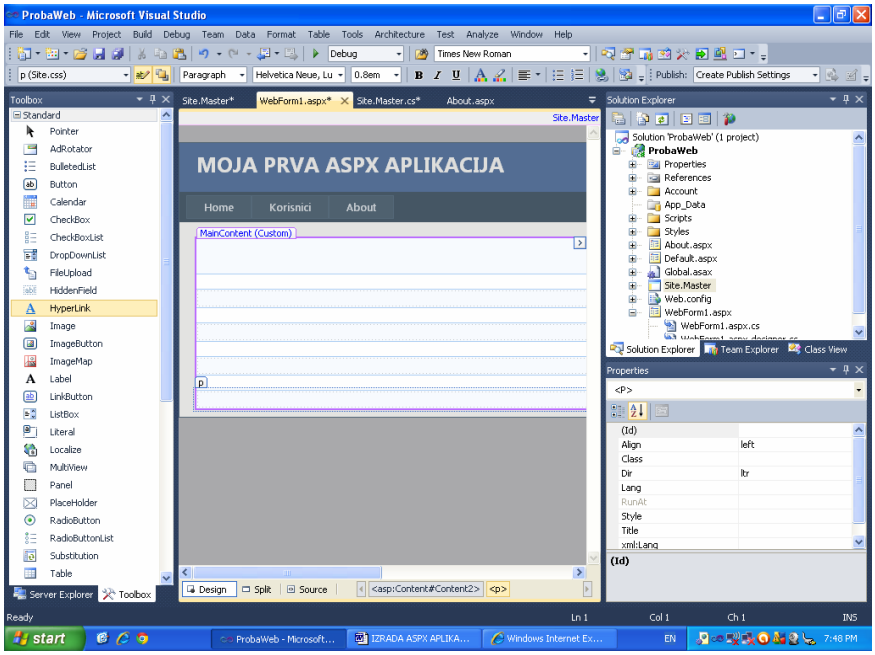
1. Dodajemo novu formu - Project, Add New Item, Web. Možemo birati Web Form ili Web form using Master Page. Biramo drugu opciju.



Otvara se dijalog za izbor master stranice, s obzirom da u jednom projektu može biti više Site.Master fajlova.

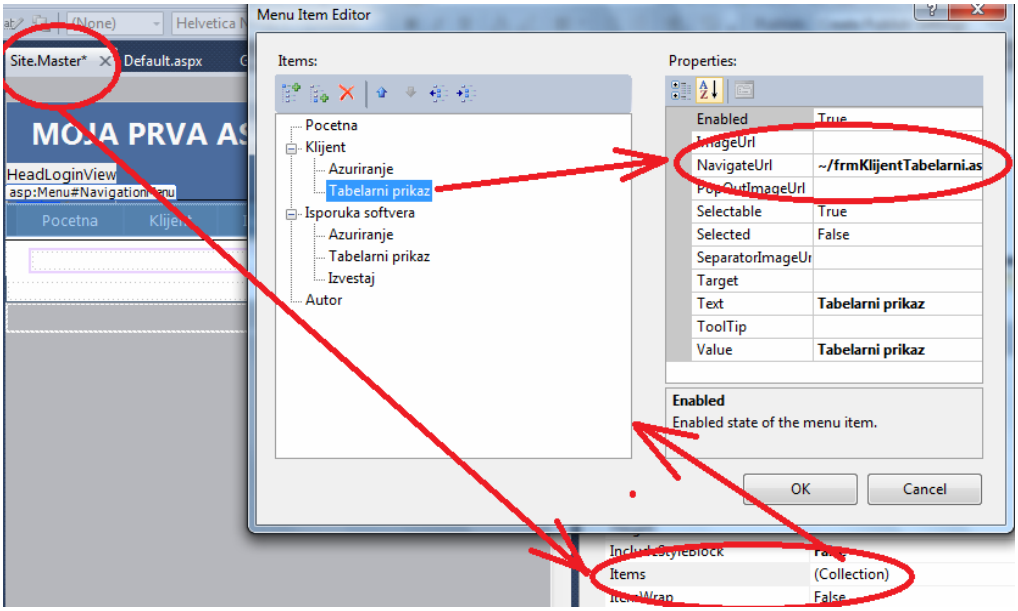


2.Dobijamo novu formu za rad. Pripremimo radnu površinu (tasterom "enter" radi proširenja).



Menjamo naziv u Solution exploreru (desni taster, Rename).

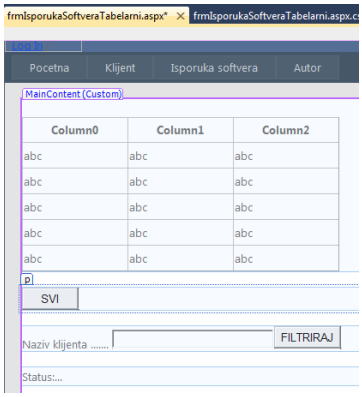
Povezujemo u okviru Site.Master (NavigationMenu – osobina Items – Collection) stavku glavnog menija sa stranicom koristeći NavigateURL.



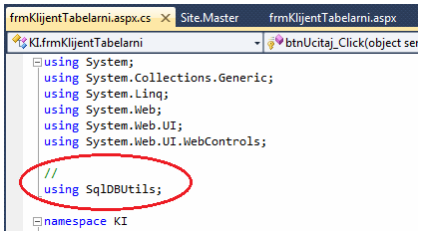
11.4. Tabelarni prikaz podataka sa filtriranjem

Postavljamo **gridview kontrolu**.sa tooboxa iz "Data" kartice i dodajemo i **button**.

Dizajn forme sa filtriranjem:



Dodajemo u **using sekciji** reference na biblioteke klasa koje smo prethodno dodali u add reference.



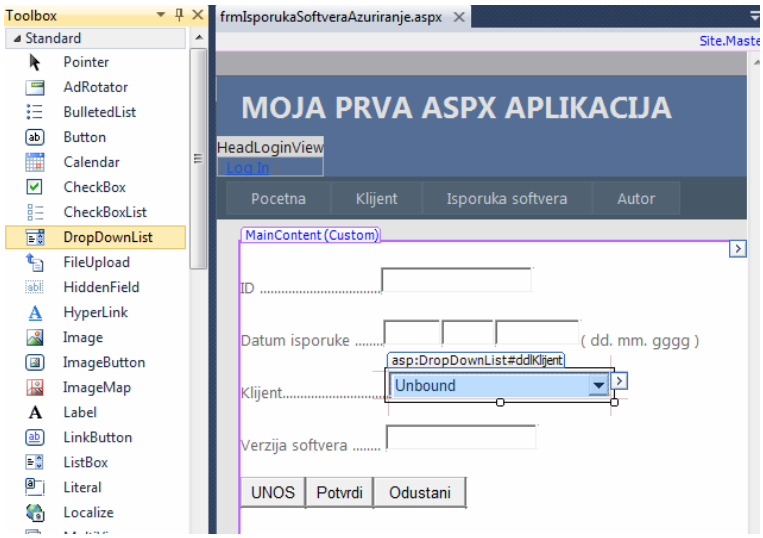
Programski kod forme sa filtriranjem:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
//
using SqlDBUtils;
using DataInteropUtils;

namespace KI
{
   public partial class frmIsporukaSoftveraTabelarni : System.Web.UI.Page
   {

     // ATRIBUTI

       private clsSqlTabela pTabela;
       private System.Data.DataSet dsIsporukaSoftvera;



       // NASE PROCEDURE


       private void NapuniGrid(System.Data.DataSet ds)
       {
          // povezivanje grida sa datasetom
          grdIsporukaSoftvera.DataSource = ds.Tables[0];
          grdIsporukaSoftvera.DataBind();
       }


       protected void Page_Load(object sender, EventArgs e)
       {
          pTabela = new clsSqlTabela(Global.Konekcija, "IsporukaSoftvera");
```

```
            dsIsporukaSoftvera = pTabela.DajPodatke("select Klijent.Naziv, IsporukaSoftvera.DatumIsporuke,
IsporukaSoftvera.VerzijaSoftvera from Klijent inner join IsporukaSoftvera on
Klijent.Sifra=IsporukaSoftvera.SifraKlijenta");
            NapuniGrid(dsIsporukaSoftvera);
        }

        protected void btnSvi_Click(object sender, EventArgs e)
        {
            if (Global.uspesnoOtvorenaKonekcija == true)
            {
                // punimo grid podacima
                pTabela = new clsSqlTabela(Global.Konekcija, "IsporukaSoftvera");
                dsIsporukaSoftvera = pTabela.DajPodatke("select Klijent.Naziv, IsporukaSoftvera.DatumIsporuke,
IsporukaSoftvera.VerzijaSoftvera from Klijent inner join IsporukaSoftvera on
Klijent.Sifra=IsporukaSoftvera.SifraKlijenta");
                NapuniGrid(dsIsporukaSoftvera);
            }

        }

        protected void btnFiltriraj_Click(object sender, EventArgs e)
        {
            if (Global.uspesnoOtvorenaKonekcija == true)
            {
                // punimo grid podacima
                pTabela = new clsSqlTabela(Global.Konekcija, "IsporukaSoftvera");
                // KRITERIJUM FILTERA: kod ID brojeva =, kod naziva LIKE
                dsIsporukaSoftvera = pTabela.DajPodatke("select Klijent.Naziv, IsporukaSoftvera.DatumIsporuke,
IsporukaSoftvera.VerzijaSoftvera from Klijent inner join IsporukaSoftvera on
Klijent.Sifra=IsporukaSoftvera.SifraKlijenta where Klijent.Naziv like '%" + txtFilter.Text + "%'");
                NapuniGrid(dsIsporukaSoftvera);
            }

        }


    }
}
```

12.5. Unos podataka na formi sa combo box-om

1. Postavljanje elemenata dizajna forme, za combo koristimo Drop Down list.



Programski kod forme:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
//
using SqlDBUtils;

namespace KI
{
   public partial class frmIsporukaSoftveraAzuriranje : System.Web.UI.Page
   {
      // atributi

      // nase procedure
      private void NapuniCombo()
      {
         clsSqlTabela pTabela = new clsSqlTabela(Global.Konekcija, "Klijent");
         System.Data.DataSet dsTipKredita = pTabela.DajPodatke("select * from Klijent");
         //Popunjavanje liste mesta sa nazivima iz objekta tipa DataSet
         for (int i = 0; i <= dsTipKredita.Tables[0].Rows.Count - 1; i++)
         {
            ddlKlijent.Items.Add(dsTipKredita.Tables[0].Rows[i].ItemArray[1].ToString());
         }

      }

      private string DajIDkredita(string Naziv)
      {
         {
            string ID = "";
            clsSqlTabela pTabela = new clsSqlTabela(Global.Konekcija, "Klijent");
            System.Data.DataSet dsTipKredita = pTabela.DajPodatke("select Sifra from Klijent where Naziv='" + Naziv
+ "'");
            // moze biti samo jedan slog koji odgovara tako da je rows od 0, a posto u select ima samo 1 polje,onda je
itemarray od 0
            ID = dsTipKredita.Tables[0].Rows[0].ItemArray[0].ToString();
            return ID;
         }
      }

      private void SnimiPodatke()
      {
         // preuzimanje podataka sa korisnickog interfejsa
         // i odgovarajuce konverzije
         string Id =txtID.Text;
         string DatumIsporuke = "'" + txtMesec.Text + "/" + txtDan.Text + "/" + txtGodina.Text + "'";
         string SifraKlijenta = DajIDkredita(ddlKlijent.Text);
         string VerzijaSoftvera = txtVerzijaSoftvera.Text;

         // izvrsavanje snimanja
         clsSqlTabela pTabelaAzuriranje = new clsSqlTabela(Global.Konekcija, "IsporukaSoftvera");
         string Upit = "INSERT INTO IsporukaSoftvera VALUES (" + Id + "," + DatumIsporuke + ",'" + SifraKlijenta +
"','" + VerzijaSoftvera + "')";
         pTabelaAzuriranje.IzvrsiAzuriranje(Upit);
         lblStatus.Text = "Uspesno je zavrsen unos podataka!";
      }

      private void IsprazniKontrole()
      {
         txtID.Text = "";
         txtDan.Text = "";
         txtMesec.Text = "";
         txtGodina.Text = "";
         txtVerzijaSoftvera.Text = "";

      }

      // dogadjaji
      protected void Page_Load(object sender, EventArgs e)
```

```
        {
            NapuniCombo();
        }

        protected void btnUnos_Click(object sender, EventArgs e)
        {
            IsprazniKontrole();
            txtID.Focus();
        }

        protected void btnOdustani_Click(object sender, EventArgs e)
        {
            IsprazniKontrole();
        }

        protected void btnPotvrdi_Click(object sender, EventArgs e)
        {
            SnimiPodatke();
        }
    }
}
```

11.6. Rad sa sesijama

U okviru izvršavanja web aplikacija, svaki pojedinačni korisnik koji pristupi i koristi funkcije u aplikaciji ima svoj skup vrednosti globalnih promenljivih aplikacije koje se odnose samo na njega. Te promenljive se mogu čuvati u sesiji. Sesija se jednostavno koristi navođenjem na sledeći način:

1. dodeljivanje vrednosti

Session [“NazivPromenljive”] = vrednost;

```
primeri:
Session["KorisnickoIme"] = txtKorisnickoIme.Text;
Session["IDKorisnika"] = dsKorisnik.Tables[0].Rows[0].ItemArray[0].ToString();
Session["Konekcija"] = Global.Konekcija;
Session["UspehKonekcije"] = Global.uspesnoOtvorenaKonekcija;
```

2. čitanje vrednosti – s obzirom da je vrednost čuvana u okviru promenljive sesije «neodređenog, opšteg tipa objekta», mora se primeniti type cast.

(tip podatka ili klasa) Session [“NazivPromenljive”]

```
primeri:
pTabela = new clsSqlTabela((clsSqlKonekcija)Session["Konekcija"], "Zemlja");
lblKorisnik.Text = (string)Session["KorisnickoIme"];
UcitajUcesnika((string)Session["KorisnickoIme"]);
```

12.LITERATURA

- ANSI/IEEE Std 1471-2000, *Recomended Practice for architectural description of Sofware-Intensive systems,* 2000.
- Balaban N, Ristić Ž, Đurković J: *Principi informatike*, Savremena administracija, 1996.
- Balduino R: *Basic Unified Process*, IBM, 2007.
- Bock D: *Entity-Relationship Modelling and Normalization Errors*, Journal of Database Management, 1997.
- Carnegie: *How do You Define Software Architecture?*, Software Engineering Institute, Carnegie Mellon University, December 2000, http://www.sei.cmu.edu/architecture/definitions.html
- CCTA: *Managing Successful Projects with PRINCE2*, Key Skills Limited, 1999.
- *Cobit 5 metodologija*, http://www.isaca.org/cobit/pages/default.aspx
- Elmasri R., Navathe S.B., *Fundamentals of Database Systems*, 5th edition, Pearson International Edition, 2007
- Eriksson, Hans-Erik, and Magnus Penker: *Business Modeling with UML: Business Patterns at Work*, John Wiley and Sons, ISBN 0-471-29551-5, 2000.
- Jauković M: *Uvod u informacione sisteme*, Tehnička knjiga, Beograd, 1992.
- Jovanović P: *Upravljanje projektom*, Fakultet organizacionih nauka, Beograd, 2006.
- Kazi Lj: *Razvoj višeslojnog web baziranog informacionog sistema za podršku upravljanju projektima*, Master rad, Tehnički fakultet «Mihajlo Pupin» Zrenjanin, 2009.
- Kazi Lj, *Integracija metoda za konceptualno modelovanje podataka u nastavi informacionih sistema*, Pupin Almanah, Tehnicki fakultet Mihajlo Pupin Zrenjanin, 2012.
- Kazi Lj, Kazi Z, Radulović B, Letić D, Bhatt M: *Applying Integration Of Conceptual Data Modelling Methods Within Information System Development: A Case Study*, Metalurgia International, ISSN 1582 - 2214 vol. XVII no. 6 (2012) pp. 67-74
- Kazi Lj., Radulović B., Radosav D., Bhatt M., Grmuša N., Štiklica N.: *Business Process Model and Elements Of Software Design: The Mapping Approach*, Konferencija «Applied Internet and Information Technologies» AIIT 2012, Zrenjanin
- Lazarević B, Jovanović V, Vučković M: *Projektovanje informacionih sistema I deo*, Naučna knjiga, Beograd, 1988.
- Lazarević B, Marjanović Z., Aničić N, Babarogić S: *Baze podataka*, Fakultet organizacionih nauka, Beograd, 2003.
- Lockemann P.: *Information system Architectures: From Art to Science*, Conference BTW2003 (Business, Technologie and Web) Proceeedings 2003., http://doeseno.informatik.uni-leipzig.de/proceedings/paper/keynote_lockeman.pdf
- Marshall, Chris: *Enterprise Modeling with UML: Designing Successful Software through Business Analysis*, Addison Wesley Longman, ISBN 0 201-43313-3, 2000
- Mogin P, Luković I, Govedarica M: *Principi projektovanja baze podataka*, Fakultet Tehnickih nauka, Novi Sad, 2000.
- Mogin P, Luković I: *Principi baza podataka*, University of Novi Sad, School for technical sciences, Novi Sad, Serbia, 1996.
- Naiburg E, Maksimchuk R: *UML for Database Design*, Addison-Wesley, 2001
- Project Management Institute: *Project Management Body of Knowledge*, Upper Darby, 1987.
- Radulović B, Kazi Lj, Kazi Z: *Informacioni sistemi – odabrana poglavlja*, Tehnički fakultet Mihajlo Pupin, Zrenjanin, 2011.
- Ristić D: *Proizvodno poslovni sistemi – osnovi organizacije rada*, Edicija YU Experti, IBN centar, Beograd, 1991.
- Siegelaub J. M: *How PRINCE 2 can complement PMBOK and your PMP*, PMI/Westchester chapter, 2004.
- Stanojević M: *Osnovi projektovanja informacionih sistema*, Naučna knjiga, Beograd, 1986.
- Stankić R: *Poslovna informatika*, Ekonomski fakultet Univerziteta u Beogradu, 2005.
- Tissot, Florence, and Wes Crump, *An Integrated Enterprise Modeling Environment*, P. Bernus, K. Mertins, G. Schmidt (Eds.), Handbook on Architectures of Information Systems, Springer, pp.59-79, ISBN 3-540-64453-9, 1998.
- *Uputstvo za izradu i usvajanje projekata informacionih sistema organa uprave*, Službeni glasnik SRS, broj 49/89
- Van Belle J-P: *A Framework for the Evaluation of Business Models and its Empirical Validation*, The Electronic Journal Information Systems Evaluation 2006, Volume 9 Issue 1, pp 31-44, www.ejise.com
- Vasconcelos A, Soursa P, Tribolet J: *Information system Architecture Metrics: an Enterprise Engineering Evaluation Approach*, The Electronic Journal of Information Systems Evaluation, Vol 10, Issue 1, pp 91-122, ISSN 1566-6379, www.ejise.com
- Vernadat, François, *Enterprise Modeling and Integration*, London, Chapman and Hall, 1996.
- Yeong A: *The marriage proposal of PRINCE2 and PMBOK*, 2007.
- Zachman, John, *A Framework for Information System Architecture*, IBM system journal Vol.26 Nº 3, 1987, p.276 – 292.