

1.Korisnički zahtev

Postoji potreba da se za rekreativni centar projektuje informacijski sistem. Informacijski sistem treba da omogućiti evidentiranje podataka o članovima rekreativnog centra i njegovim gostima, i uslugama koje oni koriste. Podatke o aktivnostima članova, kao što su datum korišćenja usluge, naziv usluge, cena usluge treba evidentirati u njihove članske karte. Za sve usluge koje koristi član treba kreirati račun i zapamtiti ga. Treba omogućiti da se vide svi računi jednog člana. Za goste rekreativnog centra treba omogućiti štampanje računa koji obuhvata spisak svih dnevnih usluga sa njihovim cenama i datum.

Opis zahteva pomoću modela slučajeva korišćenja

U modelu imamo sledeće slučajeve korišćenja:

1. Kreiranje korisničkog naloga
2. Poništavanje korisničkog naloga
3. Provera postojanja naloga
4. Ažuriranje korisničkog naloga
5. Storniranje naloga
6. Kreiranje usluge
7. Poništavanje usluge
8. Provera postojanja usluge
9. Ažuriranje usluge
10. Storniranje usluge
11. Logovanje
12. Kreiranje članske karte
13. Ažuriranje članske karte
14. Provera postojanja članske karte
15. Storniranje članske karte
16. Kreiranje računa
17. Provera postojanja računa
18. Storniranje računa

Slučaj korišćenja-Kreiranje korisničkog naloga

Naziv SK

Kreiranje korisničkog naloga

Aktori

Administrator

Učesnici

Administrator i sistem

Preduslov

Sistem je uključen i administrator je ulogovan pod svojom šifrom.

Osnovni scenario SK

1. Administrator poziva sistem da kreira novi nalog
2. Sistem kreira novi nalog
3. Sistem prikazuje prazan nalog

Alternativna scenarija

Slučaj korišćenja-Storniranje korisničkog naloga

Naziv SK

Storniranje korisničkog naloga

Aktori

Administrator

Ucesnici

Admonistrator i sistem

Preduslov

Sistem je uključen i administrator je ulogovan pod svojom šifrom.

Osnovni scenario SK

1. Administrator poziva sistem da stornira nalog
2. Sistem storniranovi nalog
3. Sistem prikazuje poruku da je nalog storniran

Alternativna scenarija

2.1 Ukoliko sistem nije u mogućnosti da stornira nalog prikazuje poruku da ne može da stornira nalog. Prekida izvršenje scenarija.

*Slučaj korišćenja-Poništavanje korisničkog naloga***Naziv SK**

Poništavanje korisničkog naloga

Aktori

Administrator

Ucesnici

Admonistrator i sistem

Preduslov

Sistem je uključen i administrator je ulogovan pod svojom šifrom.

Osnovni scenario

1. Administrator unosi šifru korisnika sistema koga želi da obriše
2. Administrator poziva sistem da proveri postojanje naloga
3. Sistem proverava postojanje datoga naloga
4. Sistem prikazuje dati nalog
5. Administrator poziva sistem da obriše dati nalog
6. Sistem briše nalog
7. Sistem prikazuje poruku da je uspesno obrisao nalog

Alternativna scenarija

3.1 Ukoliko sistem nije u mogućnosti da pronade nalog prikazuje poruku da ne može da nađe nalog.

*Slučaj korišćenja-Provera postojanja korisničkog naloga***Naziv SK**

Provera postojanja korisničkog naloga

Aktori

Administrator

Ucesnici

Admonistrator i sistem

Preduslov

Sistem je uključen i administrator je ulogovan pod svojom šifrom.

Osnovni scenario

1. Administrator unosi šifru korisnika sistema koga želi da vidi
2. Administrator poziva sistem da proveri postojanje naloga
3. Sistem proverava postojanje datoga naloga
4. Sistem prikazuje dati nalog

Alternativna scenarija

3.1 Ukoliko sistem nije u mogućnosti da pronade nalog prikazuje poruku da ne može da nađe nalog.

*Slučaj korišćenja-Ažuriranje korisničkog naloga***Naziv SK**

Ažuriranje korisničkog naloga

Aktori

Administrator

Ucesnici

Administrator i sistem

Preduslov

Sistem je uključen i administrator je ulogovan pod svojom šifrom.

Osnovni scenario

1. Administrator unosi šifru korisnika sistema koga želi da ažurira
2. Administrator poziva sistem da proveri postojanje naloga
3. Sistem proverava postojanje datoga naloga
4. Sistem prikazuje dati nalog
5. Administrator unosi podatke
6. Administrator poziva sistem da promeni podatke
7. Sistem menja podatke naloga
8. Sistem prikazuje poruku da je nalog promenjen

Alternativna scenarija

3.1 Ukoliko sistem nije u mogućnosti da pronađe nalog prikazuje poruku da ne može da nađe nalog.

Slučaj korišćenja-Logovanje

Naziv SK

Logovanje

Aktori

Asistent,Administrator

Ucesnici

Asistent,Administrator i sistem

Preduslov

Sistem je uključen i prikazuje formu za unos korisničkog imena i sifre

Osnovni scenario SK

1. Korisnik unosi podatke
2. Korisnik poziva sistem da proveri ispravnost podataka
3. Sistem proverava ispravnost podataka
4. Sistem prikazuje osnovnu formu

Alternativna scenarija

3.1 Ukoliko podaci nisu odgovarajući sistem prikazuje poruku da podaci nisu odgovarajući.

Slučaj korišćenja-Kreiranje usluge

Naziv SK

Kreiranje usluge

Aktori

Administrator

Ucesnici

Administrator i sistem

Preduslov

Sistem je uključen i administrator je ulogovan pod svojom šifrom.

Osnovni scenario SK

1. Administrator poziva sistem da kreira novu uslugu
2. Sistem kreira novu uslugu
3. Sistem prikazuje praznu formu sa poljima za unos podataka o usluzi

Alternativna scenarija

Slučaj korišćenja-Storniranje usluge

Naziv SK

Storniranje usluge

Aktori

Administrator

Ucesnici

Admonistrator i sistem

Preduslov

Sistem je uključen i administrator je ulogovan pod svojom šifrom.

Osnovni scenario SK

1. Administrator poziva sistem da stornira uslugu
2. Sistem stornira podatke u bazu podataka
3. Sistem prikazuje poruku da je stornirao podatke

Alternativna scenarija

2.1 Ukoliko sistem nije u mogućnosti da stornira uslugu prikazuje poruku da ne može da stornira uslugu. Prekida izvršenje scenarija.

Slučaj korišćenja-Ažuriranje usluge

Naziv SK

Ažuriranje usluge

Aktori

Administrator

Ucesnici

Admonistrator i sistem

Preduslov

Sistem je uključen i administrator je ulogovan pod svojom šifrom.

Osnovni scenario SK

1. Administrator unos šifru usluge koju želi da ažurira
2. Administrator poziva sistem da proveri postojanje usluge
3. Sistem proverava postojanje date usluge
4. Sistem prikazuje datu uslugu
5. Administrator unos podatke
6. Administrator poziva sistem da promeni podatke
7. Sistem menja podatke
8. Sistem prikazuje poruku da je promenio podatke

Alternativna scenarija

6.1 Ukoliko sistem nije u mogućnosti da nađe uslugu prikazuje poruku da ne može da nađe uslugu. Prekida izvršenje scenarija.

Slučaj korišćenja-Provera postojanja usluge

Naziv SK

Provera postojanja usluge

Aktori

Administrator

Ucesnici

Admonistrator i sistem

Preduslov

Sistem je uključen i administrator je ulogovan pod svojom šifrom.

Osnovni scenario SK

1. Administrator unos šifru usluge koju želi da pronade
2. Administrator poziva sistem da proveri postojanje usluge
3. Sistem proverava postojanje date usluge
4. Sistem prikazuje datu uslugu

Alternativna scenarija

4.1 Ukoliko sistem nije u mogućnosti da pronade uslugu prikazuje poruku da ne može da pronade uslugu. Prekida izvršenje scenarija.

Slučaj korišćenja-Poništavanje usluge

Naziv SK

Poništavanje usluge

Aktori

Administrator

Ucesnici

Adminstrator i sistem

Preduslov

Sistem je uključen i administrator je ulogovan pod svojom šifrom. Sistem prikazuje formu za obradu usluga.

Osnovni scenario SK

1. Administrator unosi šifru usluge koju želi da obriše
2. Administrator poziva sistem da pronađe uslugu
3. Sistem proverava postojanje usluge
4. Sistem prikazuje uslugu
5. Administrator poziva sistem da obriše uslugu
6. Sistem briše uslugu
7. Sistem prikazuje poruku da je obrisao uslugu

Alternativna scenarija

3.1 Ukoliko sistem nije u mogućnosti da pronade uslugu prikazuje poruku da ne može da pronade uslugu.

Slučaj korišćenja-Kreiranje članske karte

Naziv SK

Kreiranje članske karte

Aktori

Asistent

Ucesnici

Asistent i sistem

Preduslov

Sistem je uključen i asistent je ulogovan pod svojom šifrom.

Osnovni scenario SK

1. Asistent poziva sistem da kreira novu člansku kartu
2. Sistem kreira novu člansku kartu
3. Sistem prikazuje člansku kartu

Alternativna scenarija

Slučaj korišćenja-Storniranje članske karte

Naziv SK

Storniranje članske karte

Aktori

Asistent

Ucesnici

Asistent i sistem

Preduslov

Sistem je uključen i asistent je ulogovan pod svojom šifrom.

Osnovni scenario SK

1. Asistent poziva sistem da stornira člansku kartu
2. Sistem stornira podatke u bazu podataka
3. Sistem prikazuje poruku da je stornira kartu

Alternativna scenarija

2.1 Ukoliko sistem nije u mogućnosti da stornira člansku kartu prikazuje poruku da ne može da stornira člansku kartu. Prekida izvršenje scenarija.

Slučaj korišćenja-Ažuriranje članske karte

Naziv SK

Ažuriranje članske karte

Aktori

Asistent

Ucesnici

Asistent i sistem

Preduslov

Sistem je uključen i asistent je ulogovan pod svojom šifrom. Sistem prikazuje formu za obradu članskih karti.

Osnovni scenario SK

1. Asistent unosi šifru članske karte koju traži
2. Asistent poziva sistem da pronađe člansku kartu
3. Sistem proverava postojanje članske karte
4. Sistem prikazuje člansku kartu
5. Asistent menja željene podatke
6. Asistent poziva sistem da promeni podatke u članskoj karti
7. Sistem menja podatke u članskoj karti
8. Sistem prikazuje poruku

Alternativna scenarija

3.1 Ukoliko sistem nije u mogućnosti da pronađe člansku kartu prikazuje poruku da ne može da pronađe člansku kartu. Prekida izvršenje scenarija.

Slučaj korišćenja-Traženje članske karte

Naziv SK

Traženje članske karte

Aktori

Asistent

Ucesnici

Asistent i sistem

Preduslov

Sistem je uključen i asistent je ulogovan pod svojom šifrom.

Osnovni scenario SK

1. Asistent unosi šifru članske karte koju traži
2. Asistent poziva sistem da pronađe člansku kartu
3. Sistem proverava postojanje članske karte
4. Sistem prikazuje člansku kartu

Alternativna scenarija

3.1 Ukoliko sistem nije u mogućnosti da pronađe člansku kartu prikazuje poruku da ne može da pronađe člansku kartu.

Slučaj korišćenja-Kreiranje računa

Naziv SK

Kreiranje računa

Aktori

Asistent

Ucesnici

Asistent i sistem

Preduslov

Sistem je uključen i asistent je ulogovan pod svojom šifrom.

Osnovni scenario SK

1. Asistent poziva sistem da kreira novi račun
2. Sistem kreira novi račun
3. Sistem prikazuje novi račun

Alternativna scenarija

Slučaj korišćenja-Storniranje računa

Naziv SK

Storniranje računa

Aktori

Asistent

Ucesnici

Asistent i sistem

Preduslov

Sistem je uključen i asistent je ulogovan pod svojom šifrom.

Osnovni scenario SK

1. Asistent poziva sistem da stornira račun
2. Sistem stornira podatke u bazu
3. Sistem prikazuje poruku da je snimio račun

Alternativna scenarija

- 8.1 Ukoliko sistem nije u mogućnosti da stornira račun prikazuje poruku da ne može da stornira račun. Prekida izvršenje scenarija

Slučaj korišćenja-Traženje računa

Naziv SK

Traženje računa

Aktori

Asistent

Ucesnici

Asistent i sistem

Preduslov

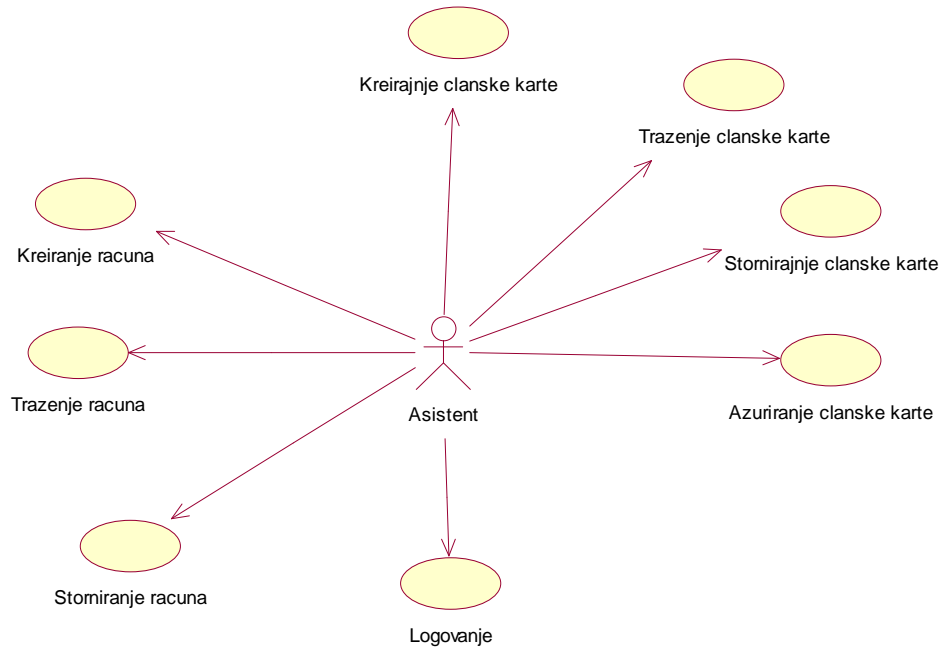
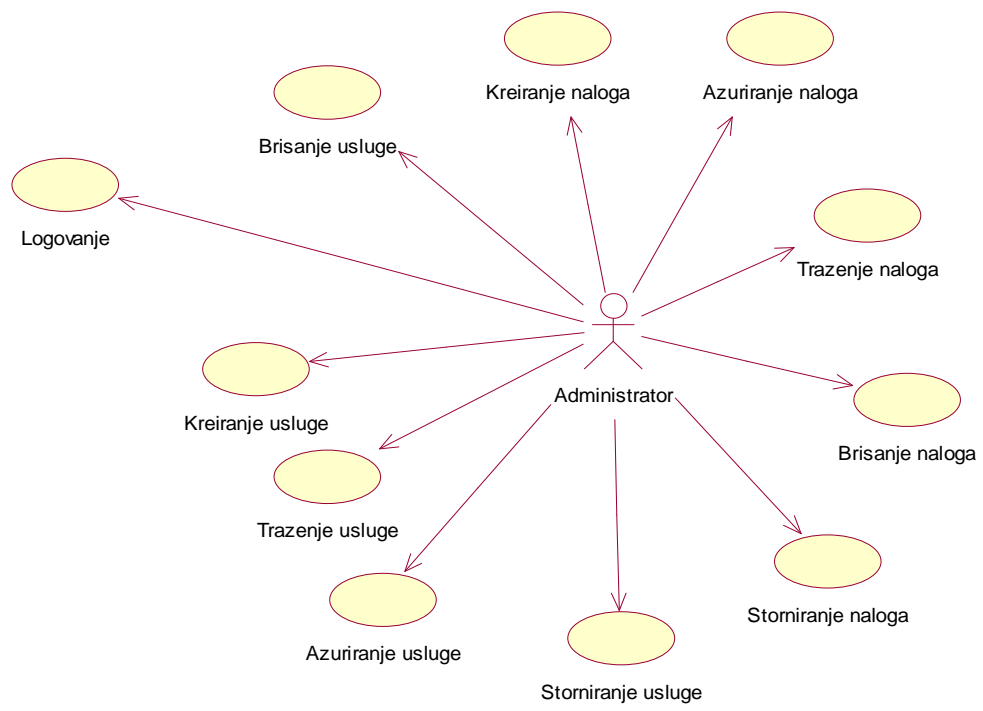
Sistem je uključen i asistent je ulogovan pod svojom šifrom. Sistem prikazuje formu za obradu računa.

Osnovni scenario SK

1. Asistent unos šifru računa koji traži
2. Asistent poziva sistem da pronađe račun
3. Sistem proverava postojanje računa
4. Sistem prikazuje račun

Alternativna scenarija

- 3.1 Ukoliko sistem nije u mogućnosti da pronađe račun prikazuje poruku da ne može da pronađe račun.



2.Analiza

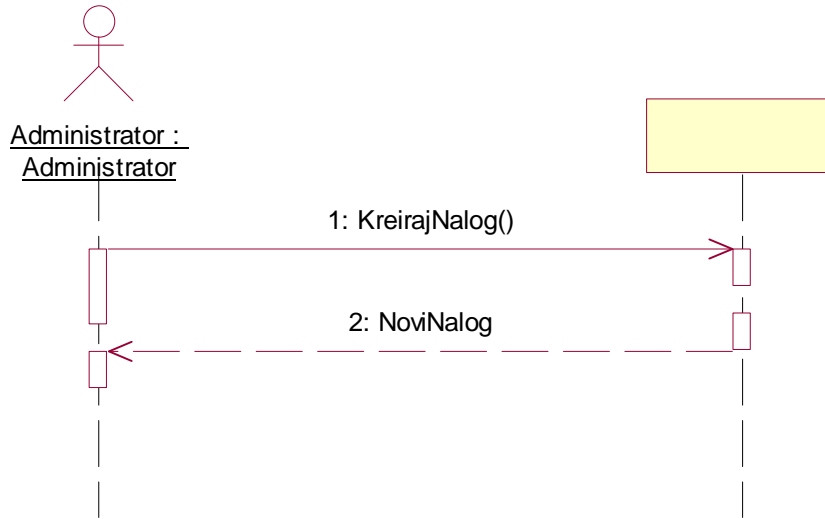
Ponašanje softverskog sistema-Sistemska dijagrama sekvenci

Dijagram sekvenci slučaja koršćenja-Kreiranje korisničkog naloga

Osnovni scenario

1. Administrator poziva sistem da kreira novi nalog
2. Sistem prikazuje prazan nalog

Alternativna scenarija



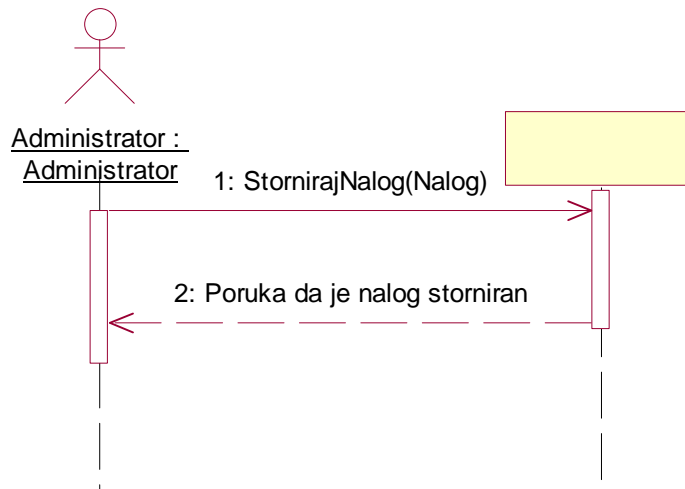
Operacije koje treba projektovati

1. NoviNalog KreirajNalog()

Dijagram sekvenci slučaja koršćenja-Storniranje korisničkog naloga

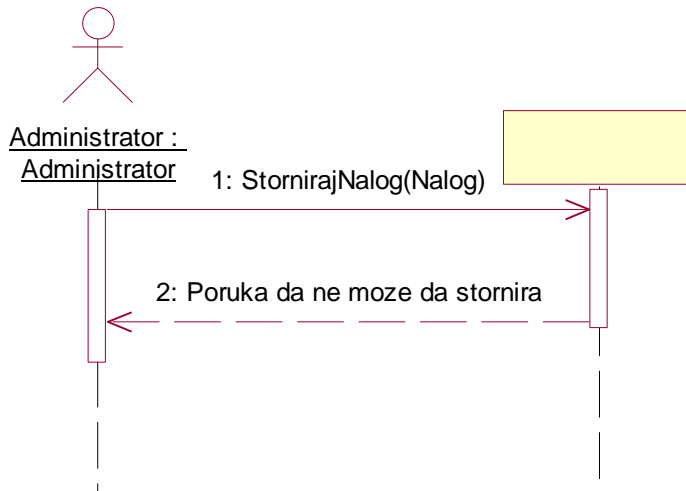
Osnovni scenario

1. Administrator poziva sistem da stornira korisnički nalog
2. Sistem prikazuje poruku da je nalog storniran



Alternativna scenarija

2.1 Ukoliko sistem nije u mogućnosti da stornira nalog prikazuje poruku da ne može da stornira nalog. Prekida izvršenje scenarija.



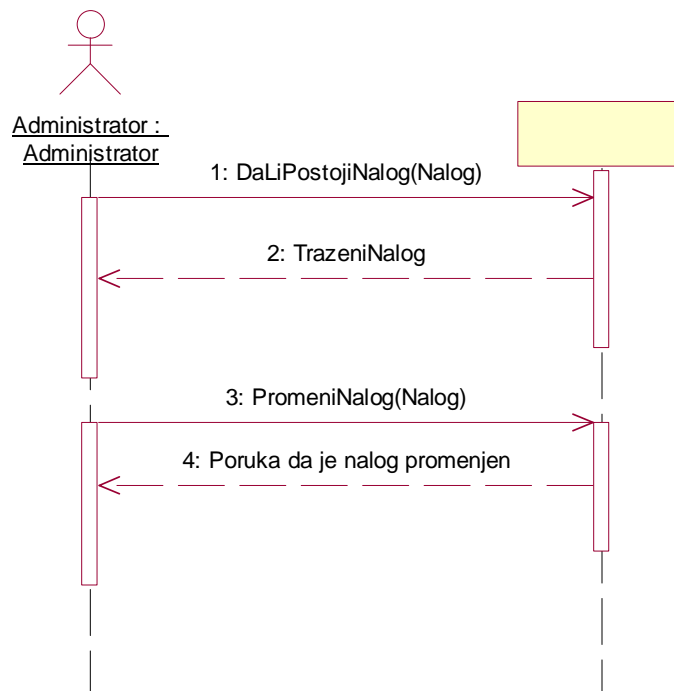
Operacije koje treba projektovati

1. Poruka StornirajNalog(Nalog)

Dijagram sekvenci slučaja koršćenja-Ažuriranje korisničkog naloga

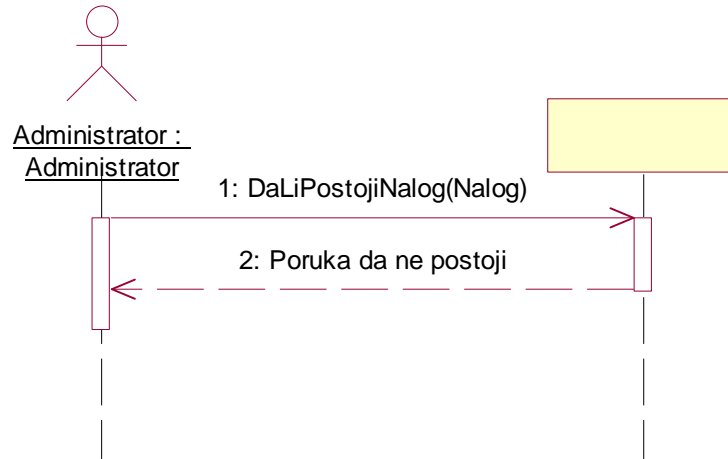
Osnovni scenario

1. Administrator poziva sistem da proveri postojanje naloga
2. Sistem prikazuje dati nalog
3. Administrator poziva sistem da promeni podatke
4. Poruka da je nalog promenjen



Alternativna scenarija

3.1 Ukoliko sistem nije u mogućnosti da pronađe nalog prikazuje poruku da ne može da nađe nalog.



Operacije koje treba projektovati

1. TrazeniNalog DaLiPostojiNalog(Nalog)
2. Poruka PromeniNalog(Nalog)

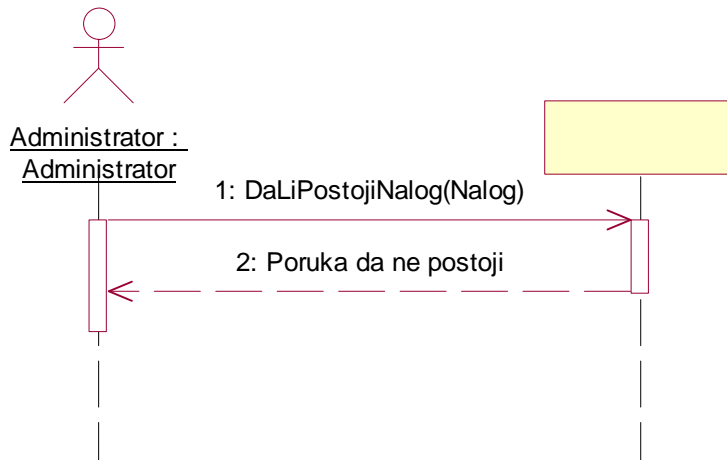
Dijagram sekvenci slučaja koršćenja-Poništavanje korisničkog naloga

Osnovni scenario

1. Administrator poziva sistem da proveri postojanje naloga
2. Sistem prikazuje dati nalog
3. Administrator poziva sistem da obriše dati nalog
4. Sistem prikazuje poruku da je uspesno obrisao nalog

Alternativna scenarija

2.1 Ukoliko sistem nije u mogućnosti da pronađe nalog prikazuje poruku da ne može da nađe nalog.



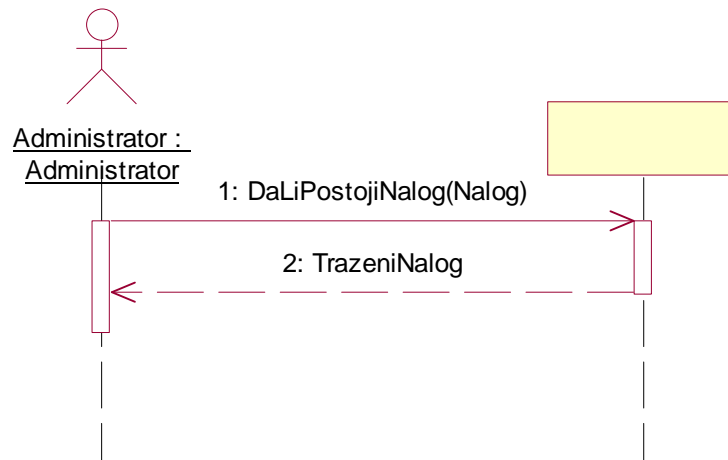
Operacije koje treba projektovati

1. TrazeniNalog DaLiPostojiNalog(Nalog)
2. Poruka ObrisiNalog(Nalog)

Dijagram sekvenci slučaja koršćenja-Provera postojanja korisničkog naloga

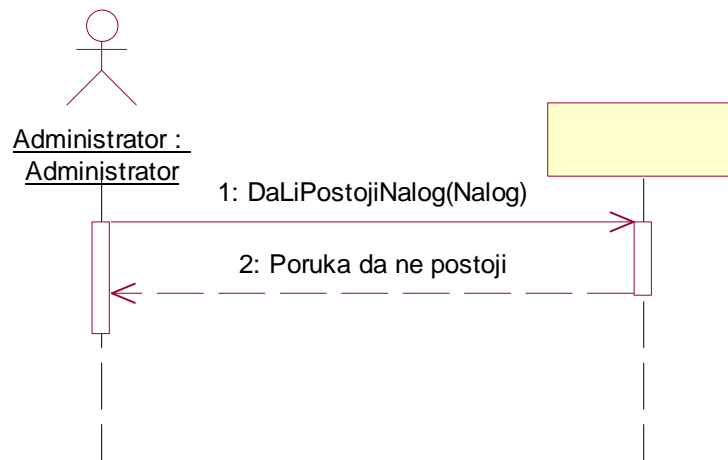
Osnovni scenario

1. Administrator poziva sistem da proveri postojanje naloga
2. Sistem prikazuje dati nalog



Alternativna scenarija

- 2.1 Ukoliko sistem nije u mogućnosti da pronade nalog prikazuje poruku da ne može da nađe nalog.



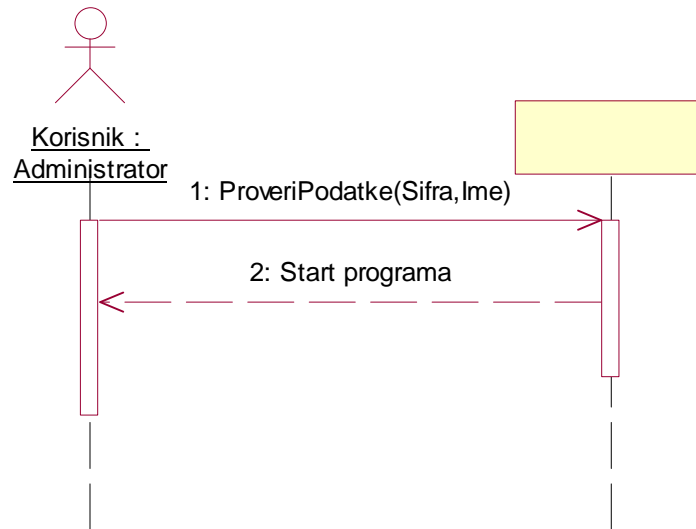
Operacije koje treba projektovati

1. TrazeniNalog DaLiPostojiNalog(Nalog)

Dijagram sekvenci slučaja koršćenja-Logovanje

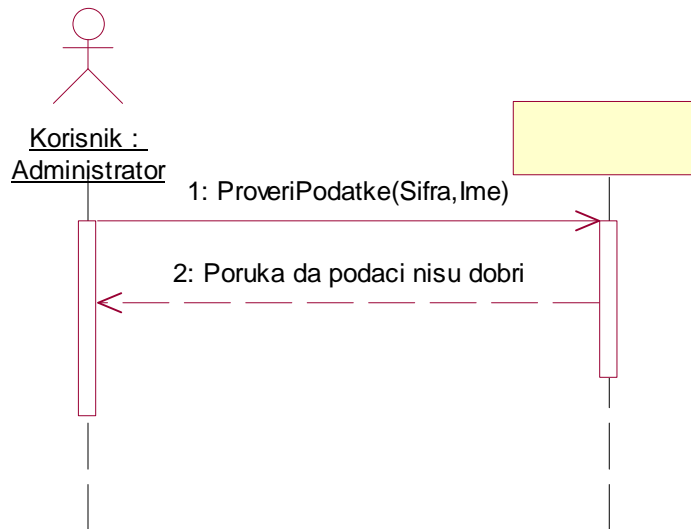
Osnovni scenario

1. Korisnik poziva sistem da proveri ispravnost podataka
2. Sistem prikazuje osnovnu formu



Alternativna scenarija

2.1 Ukoliko podaci nisu odgovarajući sistem prikazuje poruku da podaci nisu odgovarajući.



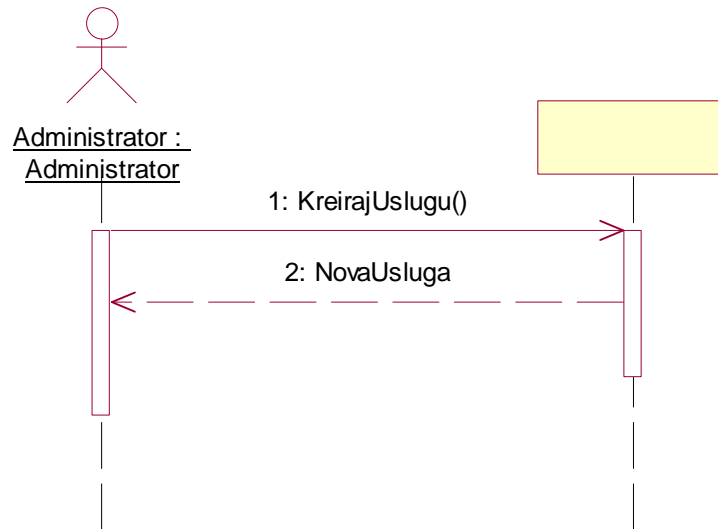
Operacije koje treba projektovati

1. OsnovnaForma ProveriIspravnostPodataka(Sifra,KorisnickoIme)

Dijagram sekvenci slučaja koršćenja-Kreiranje usluge

Osnovni scenario

1. Administrator poziva sistem da kreira novu uslugu
2. Sistem prikazuje prazanu formu sa poljima za unos podataka o usluzi



Alternativna scenarija

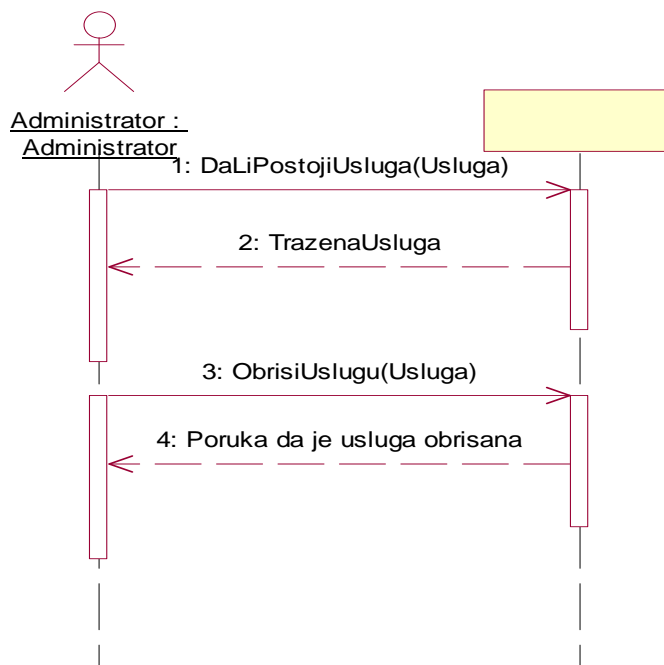
Operacije koje treba projektovati

1. NovaUsluga KreirajUslugu()

Dijagram sekvenci slučaja koršćenja-Poništavanje usluge

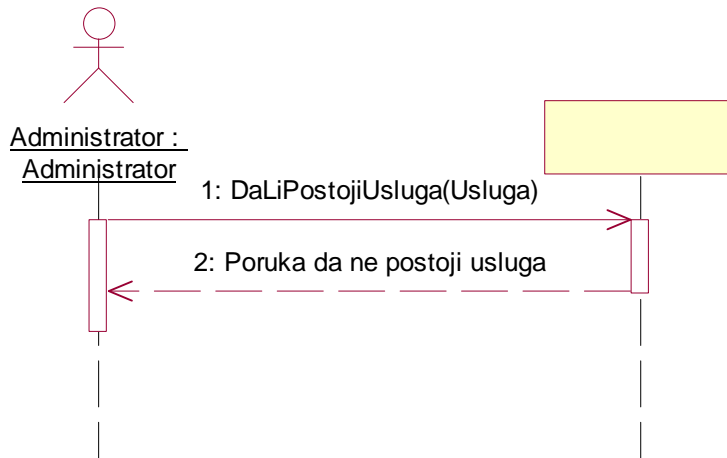
Osnovni scenario

1. Administrator poziva sistem da pronade uslugu
2. Sistem prikazuje uslugu
3. Administrator poziva sistem da obriše uslugu
4. Sistem prikazuje poruku da je obrisao uslugu



Alternativna scenarija

2.1 Ukoliko sistem nije u mogućnosti da pronade uslugu prikazuje poruku da ne može da pronade uslugu.



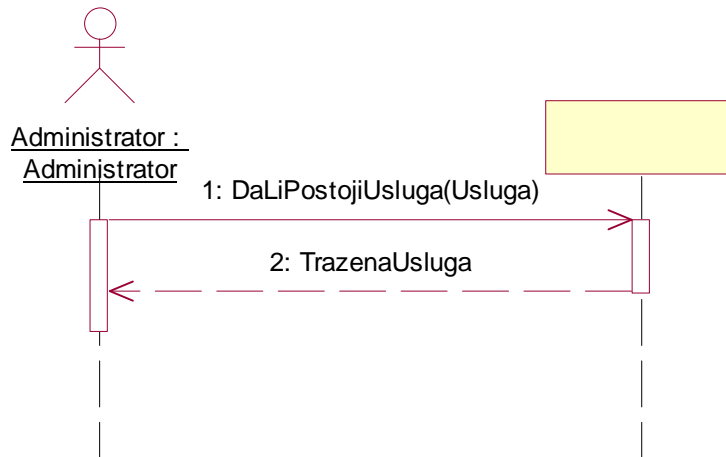
Operacije koje treba projektovati

1. TrazenaUsluga DaLiPostojiUsluga(Usluga)
2. Poruka ObrisiUslugu(Usluga)

Dijagram sekvenci slučaja koršćenja-Provera postojanja usluge

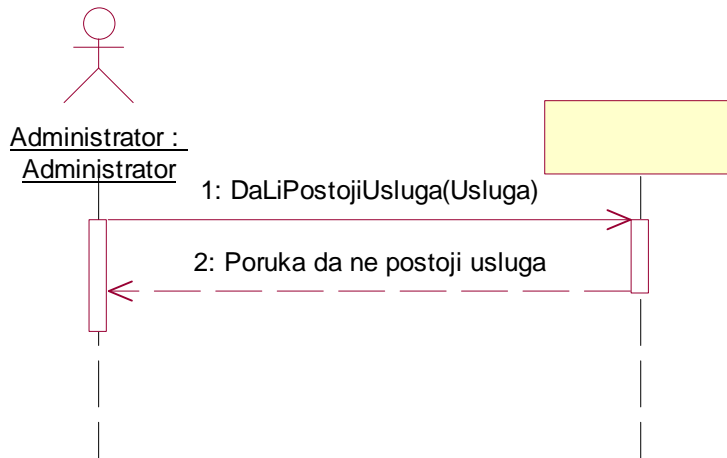
Osnovni scenario

1. Administrator poziva sistem da pronade uslugu
2. Sistem prikazuje uslugu



Alternativna scenarija

- 2.1 Ukoliko sistem nije u mogućnosti da pronade uslugu prikazuje poruku da ne može da pronade uslugu.



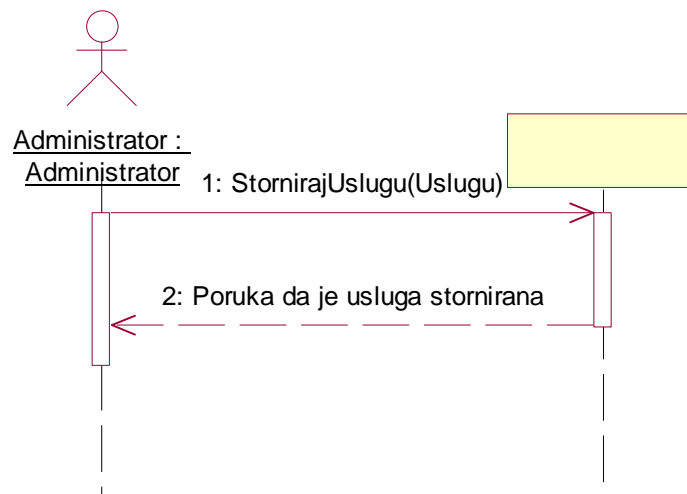
Operacije koje treba projektovati

1. TrazenaUsluga DaLiPostojiUsluga(Usluga)

Dijagram sekvenci slučaja koršćenja-Storniranje usluge

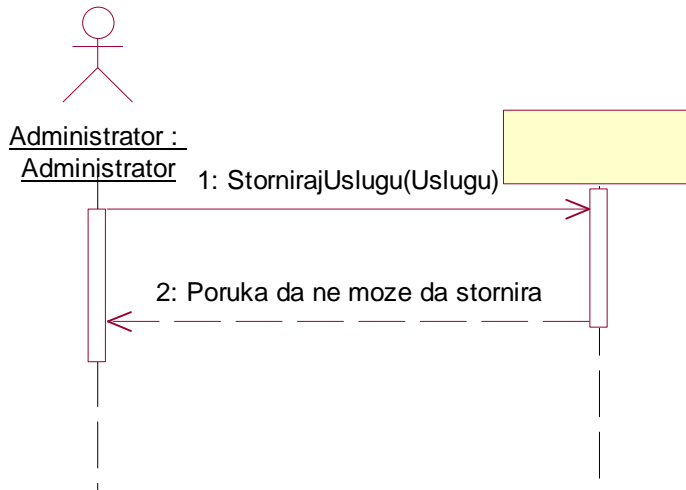
Osnovni scenario

1. Administrator poziva sistem da stornira uslugu
2. Sistem prikazuje poruku da je stornirao uslugu



Alternativna scenarija

- 2.1 Ukoliko sistem nije u mogućnosti da stornira uslugu prikazuje poruku da ne može da stornira uslugu.



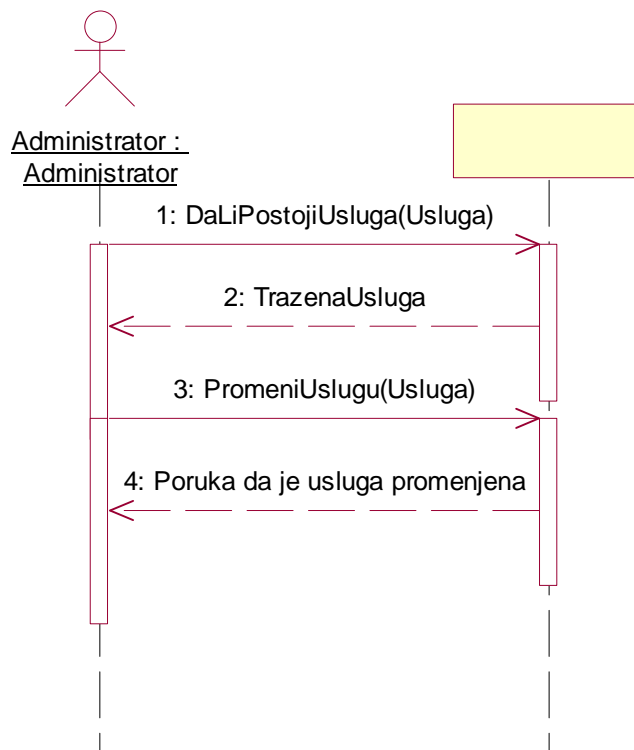
Operacije koje treba projektovati

1. Poruka StornirajUslugu(Uslugu)

Dijagram sekvenci slučaja koršćenja-Ažuriranje usluge

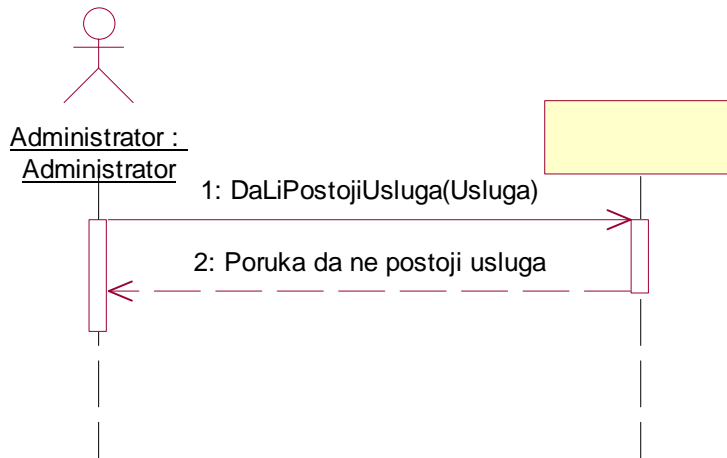
Osnovni scenarijo

1. Asistent poziva sistem da pronađe uslugu
2. Sistem prikazuje uslugu
3. Asistent poziva sistem da promeni podatke u usluzi
4. Poruka da je usluga promenjena



Alternativna scenarija

- 2.1 Ukoliko sistem nije u mogućnosti da pronađe uslugu prikazuje poruku da ne može da pronađe uslugu. Prekida izvršenje scenarija.



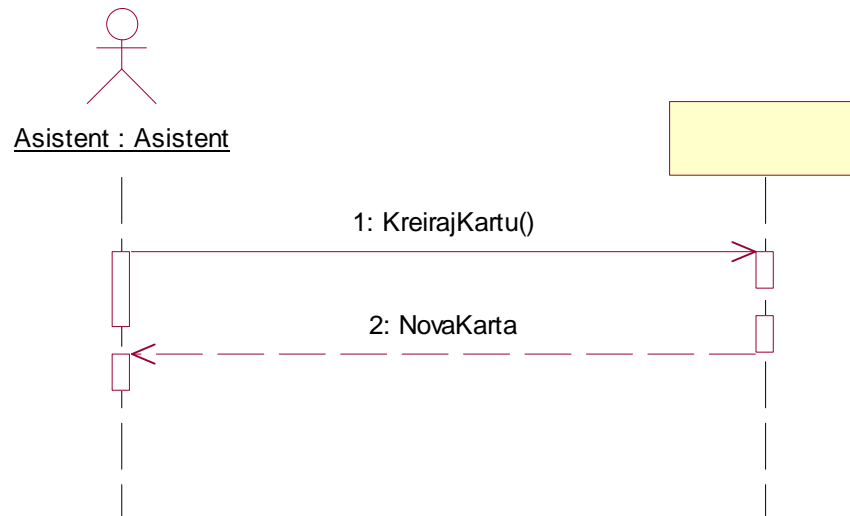
Operacije koje treba projektovati

1. TrazenaUsluga DaLiPostojiUsluga(Usluga)
2. Poruka PromeniUslugu(Usluga)

Dijagram sekvenci slučaja koršćenja-Kreiranje članske karte

Osnovni scenario

1. Asistent poziva sistem da kreira novu člansku kartu
2. Sistem prikazuje člansku kartu



Alternativna scenarija

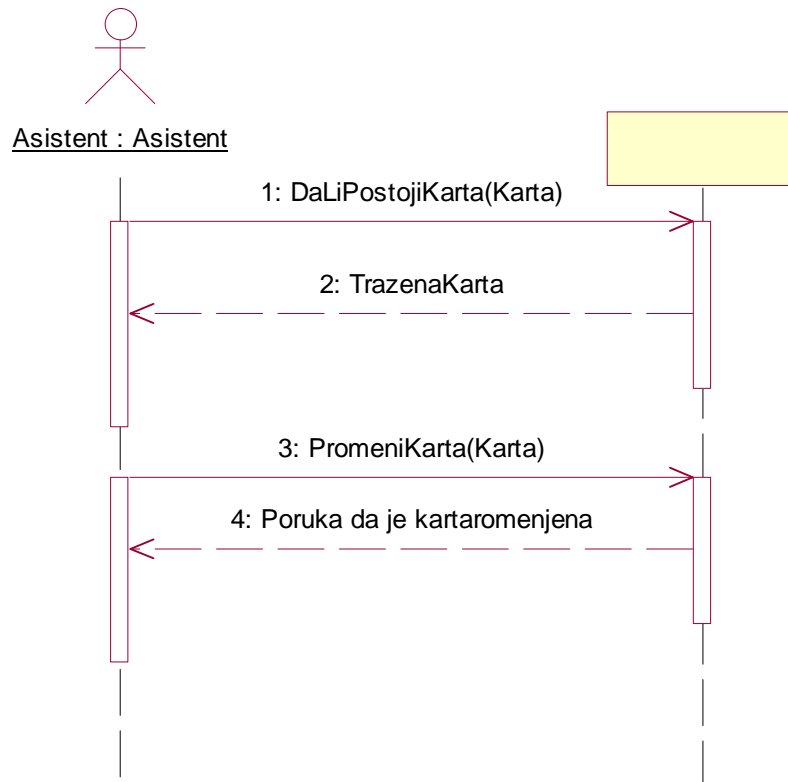
Operacije koje treba projektovati

1. NovaKarta KreirajKartu()

Dijagram sekvenci slučaja koršćenja-Ažuriranje članske karte

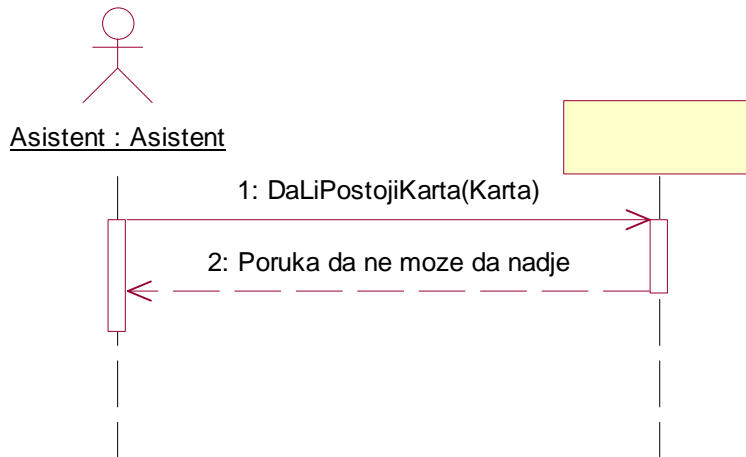
Osnovni scenario

5. Asistent poziva sistem da pronađe člansku kartu
6. Sistem prikazuje člansku kartu
7. Asistent poziva sistem da promeni podatke u članskoj karti
8. Sistem prikazuje promenjenu kartu



Alternativna scenarija

2.1 Ukoliko sistem nije u mogućnosti da pronađe člansku kartu prikazuje poruku da ne može da pronađe člansku kartu. Prekida izvršenje scenarija.



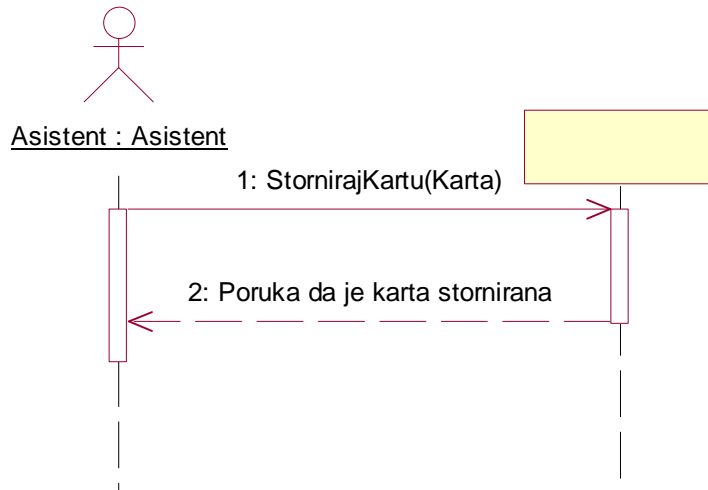
Operacije koje treba projektovati

1. TrazenaKarta DaLiPostojiKarta(Karta)
2. Poruka PromeniKartu(Karta)

Dijagram sekvenci slučaja koršćenja-Storniranje članske karte

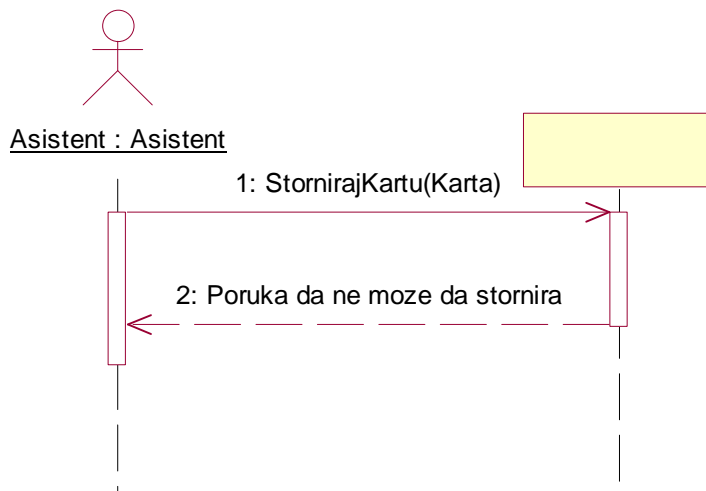
Osnovni scenario

1. Asistent poziva sistem da stornira člansku kartu
2. Sistem stornira člansku kartu



Alternativna scenarija

2.1 Ukoliko sistem nije u mogućnosti da stornira člansku kartu prikazuje poruku da ne može da stornira člansku kartu. Prekida izvršenje scenarija



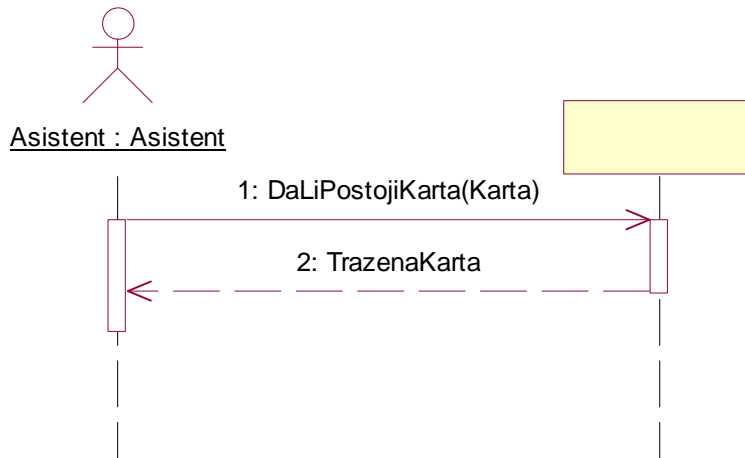
Operacije koje treba projektovati

2. Poruka StornirajKartu(Karta)

Dijagram sekvenci slučaja koršćenja-Traženje članske karte

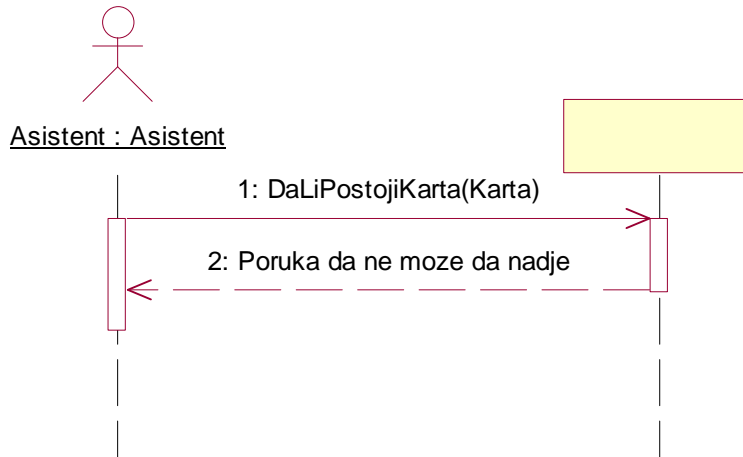
Osnovni scenario

1. Asistent poziva sistem da pronađe člansku kartu
2. Sistem prikazuje člansku kartu



Alternativna scenarija

2.1 Ukoliko sistem nije u mogućnosti da nađe člansku kartu prikazuje poruku da ne može da nađe člansku kartu.



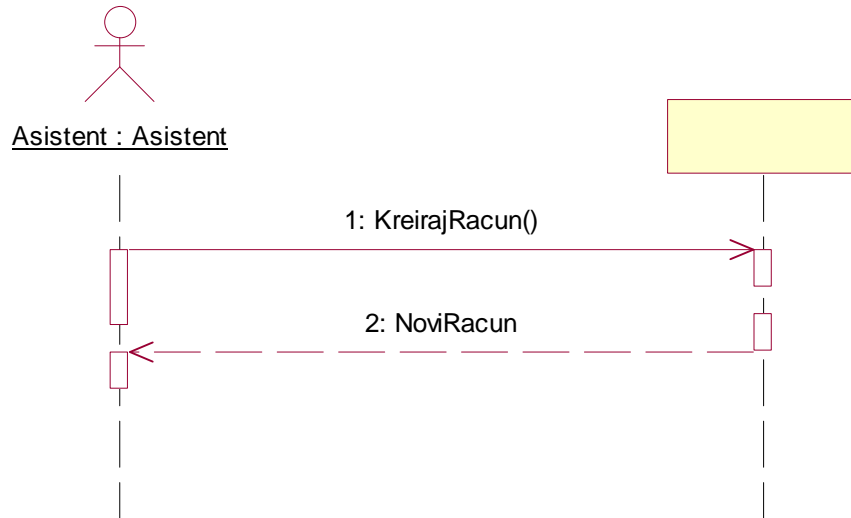
Operacije koje treba projektovati

1. TrazenaKarta DaLiPostojiKarta(Karta)

Dijagram sekvenci slučaja koršćenja-Kreiranje računa

Osnovni scenario

1. Asistent poziva sistem da kreira novi račun
2. Sistem prikazuje novi račun



Alternativna scenarija

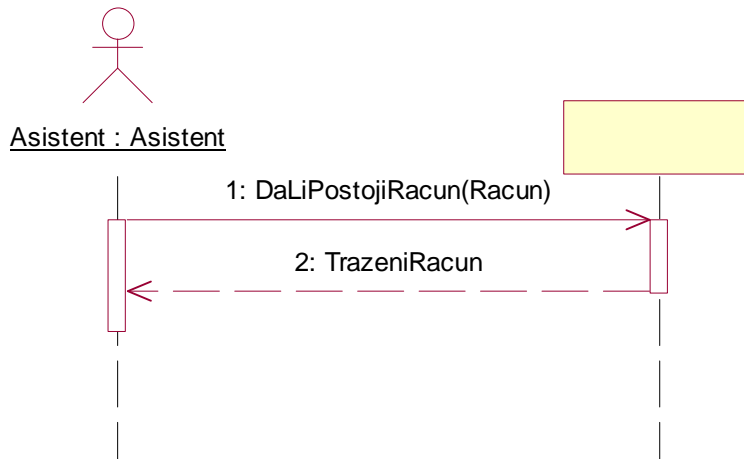
Operacije koje treba projektovati

1. NoviRacun KreirajRacun()

Dijagram sekvenci slučaja koršćenja-Traženje računa

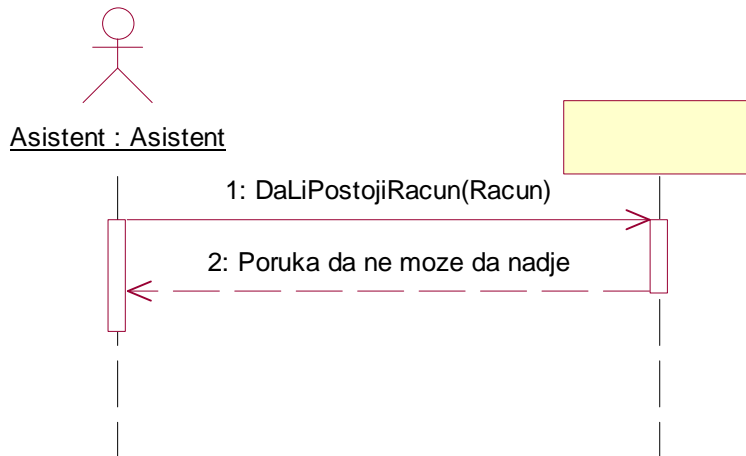
Osnovni scenario

1. Asistent poziva sistem da pronađe račun
2. Sistem prikazuje račun



Alternativna scenarija

- 2.1 Ukoliko sistem nije u mogućnosti da pronađe račun prikazuje poruku da ne može da pronađe račun.



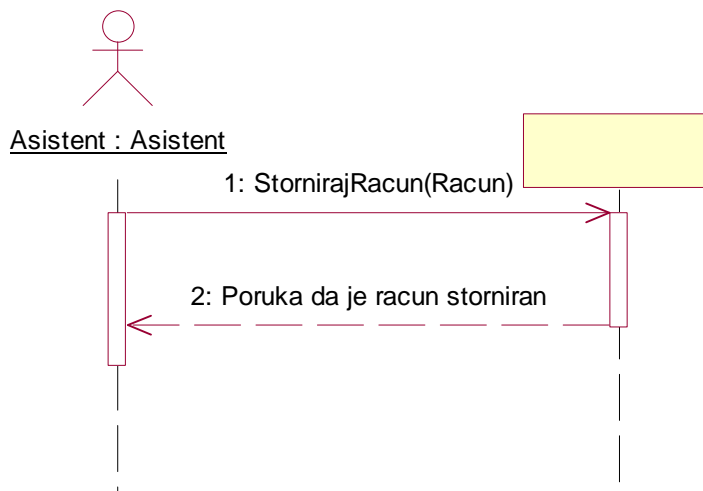
Operacije koje treba projektovati

1. TrazeniRacun DaLiPostojiRacun(Racun)

Dijagram sekvenci slučaja koršćenja-Storniranje računa

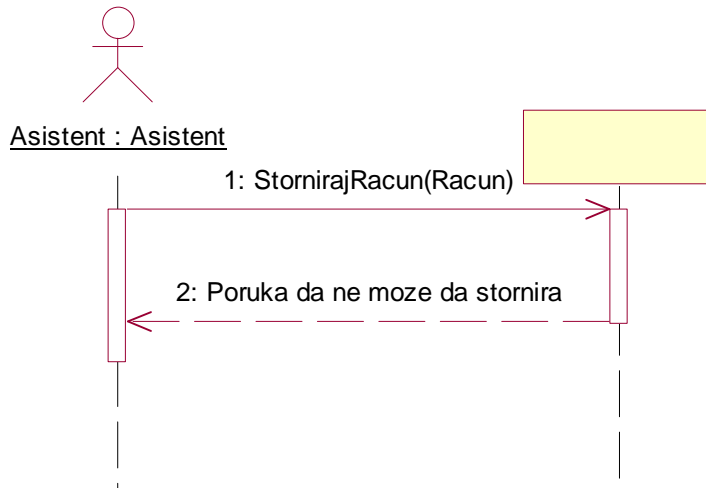
Osnovni scenario

2. Asistent poziva sistem da stornira račun
3. Sistem stornira račun



Alternativna scenarija

- 2.1 Ukoliko sistem nije u mogućnosti da stornira račun prikazuje poruku da ne može da stornira račun.



Operacije koje treba projektovati

2. Poruka StornirajRacun()

Rezultat analize sistemskih dijagrama sekvenci

Kao rezultat analize scenarija dobijeno je ukupno 18 sistemskih operacija koje treba projektovati:

1. NoviNalog KreirajNalog()
2. Poruka StornirajNalog(Nalog)
3. TrazeniNalog DaLiPostojiNalog(Nalog)
4. Poruka PromeniNalog(Nalog)
5. Poruka ObrisiNalog(Nalog)
6. Poruka ProveriIspravnostPodataka(Sifra,KorisnickoIme)
7. NovaUsluga KreirajUslugu()
8. Poruka StornirajUslugu(Usluga)
9. TrazenaUsluga DaLiPostojiUsluga(Usluga)
10. Poruka PromeniUslugu(Usluga)
11. Poruka ObrisiUslugu(Usluga)
12. NovaKarta KreirajKartu()
13. Poruka StornirajKartu(Karta)
14. TrazenaKarta DaLiPostojiKarta(Karta)
15. Poruka PromeniKartu(Karta)
16. NoviRacun KreirajRacun()
17. Poruka StornirajRacun(Racun)
18. TrazeniRacun DaLiPostojiRacun(Racun)

Ponašanje softverskog sistema-Definisanje ugovora o operacijama

UGOVOR: KreirajNalog

Operacija: KreirajNalog():Nalog

Preduslovi:

Postuslovi: Napravljen je novi nalog

UGOVOR: StornirajNalog

Operacija: StornirajNalog(Nalog):signal

Preduslovi:

Postuslovi: Storniran je novi nalog

UGOVOR: DaLiPostojiNalog
Operacija:DaLiPostojiNalog(Nalog):Nalog
Preduslovi:
Postuslovi:Prikazan je nalog ukoliko postoji

UGOVOR: PromeniNalog
Operacija:PromeniNalog(Nalog):signal
Preduslovi:Nalog treba da postoji da bi se izvršila SO
Postuslovi:Promenjen je postojeći nalog

UGOVOR: ObrišiNalog
Operacija:ObrišiNalog(Nalog): signal
Preduslovi:Nalog treba da postoji da bi se izvršila SO
Postuslovi:Obrisan je postojeći nalog

UGOVOR: ProveriIspravnostPodataka
Operacija:ProveriIspravnostPodataka(Sifra,KorisničkoIme):Signal
Preduslovi:
Postuslovi:Vraća se potvrda ako postoji nalog

UGOVOR: KreirajUslugu
Operacija:KreirajUslugu():Usluga
Preduslovi:
Postuslovi:Kreirana je nova usluga

UGOVOR: StornirajUslugu
Operacija:StornirajUslugu(Usluga):Signal
Preduslovi:
Postuslovi:Storniran je nova usluga

UGOVOR: DaLiPostojiUsluga
Operacija:DaLiPostojiUsluga(Usluga):Usluga
Preduslovi:
Postuslovi:Prikazana je usluga ukoliko postoji

UGOVOR: ObrišiUslugu
Operacija:ObrišiUslugu(Usluga):Signal
Preduslovi:Usluga treba da postoji da bi se izvršila SO
Postuslovi:Obrisana je postojeća usluga

UGOVOR: PromeniUslugu
Operacija:PromeniUslugu(Usluga):Signal
Preduslovi:Usluga treba da postoji da bi se izvršila SO
Postuslovi:Promenjena je postojeća usluga

UGOVOR: KreirajKartu
Operacija:KreirajKartu():ClanskaKarta
Preduslovi:
Postuslovi:Napravljena je nova karta

UGOVOR: StornirajKartu
Operacija:StornirajKartu(Karta):Signal
Preduslovi:
Postuslovi:Stornirana je nova karta

UGOVOR: DaLiPostojiKarta
Operacija:DaLiPostojiKarta(Karta):ClanskaKarta
Preduslovi:
Postuslovi:Prikazana je karta ukoliko postoji

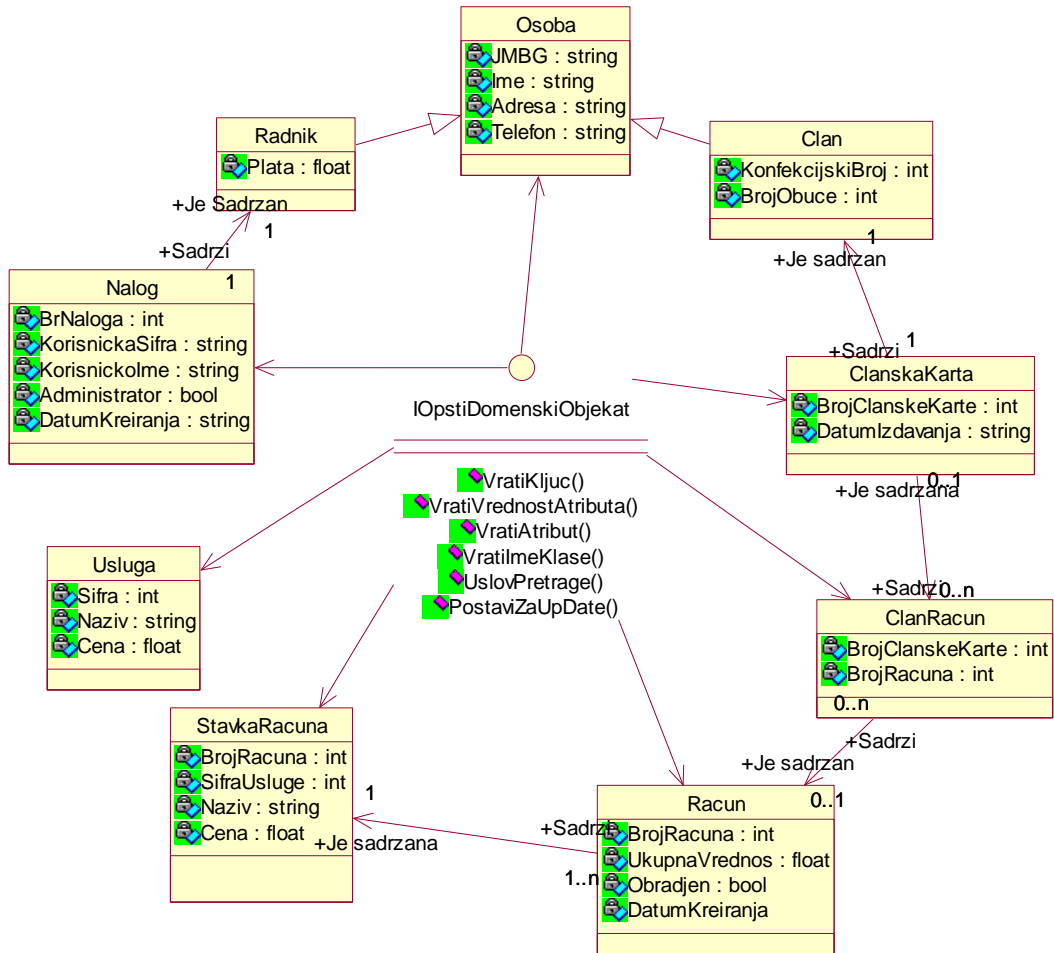
UGOVOR: PromeniKartu
Operacija:PromeniKartu(Kartu):Signal
Preduslovi:Karta treba da postoji da bi se izvršila SO
Postuslovi:Promenjena je postojeća karta

UGOVOR: KreirajRacun
Operacija:KreirajRacun():Racun
Preduslovi:Prethodni račun je obrđen
Postuslovi:Kreiran je novi račun

UGOVOR: StornirajRacun
Operacija:StornirajRacun(Racun):Signal
Preduslovi:
Postuslovi:Storniran je novi racun

UGOVOR: DaLiPostojiRacun
Operacija:DaLiPostojiRacun(Racun):Racun
Preduslovi:
Postuslovi:Prikazan je racun ukoliko postoji

Struktura softverskog sistema-Konceptualni model



Struktura softverskog sistema-Relacioni model

Nalog(BrNaloga,JMBG,Ime,Adresa,Telefon,Plata,DatumKreiranja,KorIme,KorSifra,Administrator)

ClanskaKarta(BrClKarte,JMBG,Ime,Adresa,Telefon,DatumKreiranja,KonfBroj,BrObuce)

ClanRacun(BrClKarte,BrRacuna)

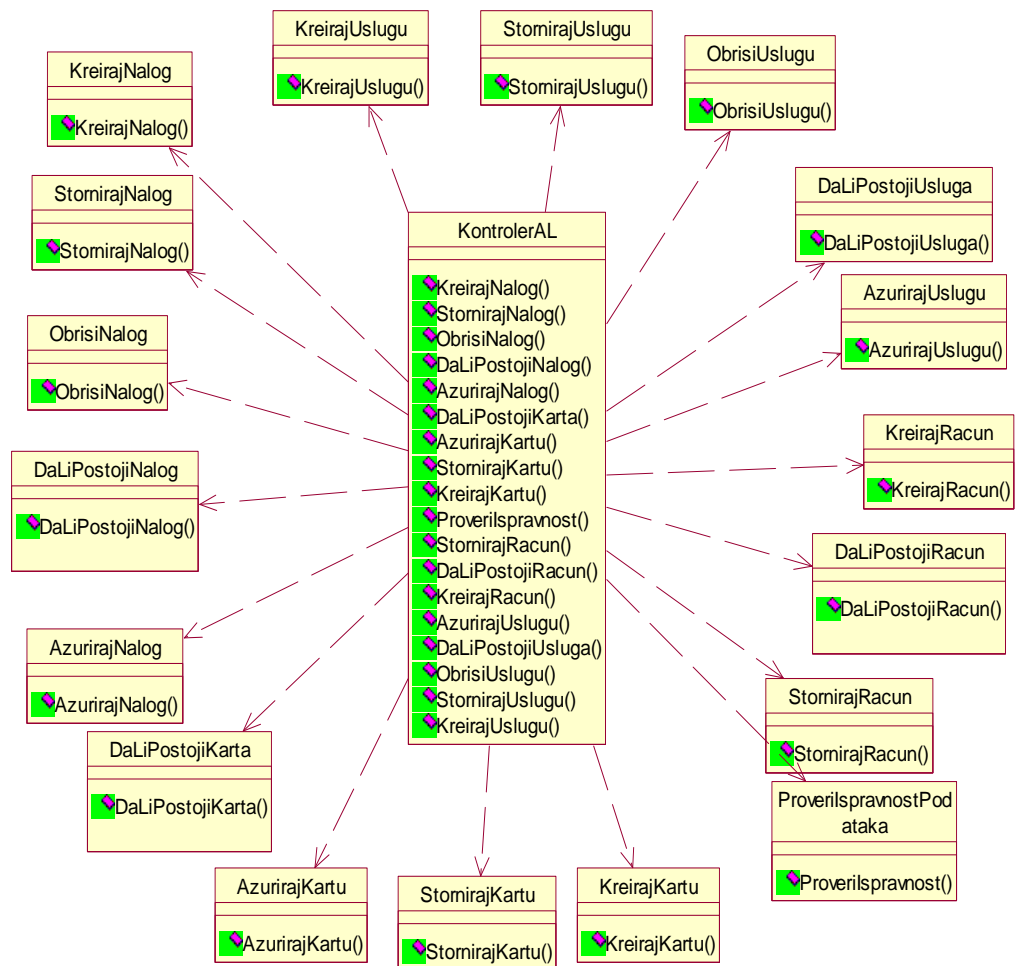
Racun(BrRacuna,UkVrednost,DatumKreiranja,Obradjen)

StavkaRacuna(BrRacuna,Sifra,Naziv,Cena)

Usluga(Sifra,Naziv,Cena)

3.Projektovanje

Projektovanje aplikacione logike-Kontroler



```

public abstract class KontrolerAL
{
    //Odeljak za kreiranje domenskih objekata

    public static Nalog KreirajNalog()
    {
        KreirajNalog kreiraj=new KreirajNalog();
        return kreiraj.Kreiraj();
    }

    public static ClanskaKarta KreirajKartu()
    {
        KreirajKartu kreiraj=new KreirajKartu();
        return kreiraj.Kreiraj();
    }

    public static Racun KreirajRacun()
    {
        KreirajRacun kreiraj=new KreirajRacun();
        return kreiraj.Kreiraj();
    }

    public static Usluga KreirajUslugu()

```

```

{
    KreirajUslugu kreiraj=new KreirajUslugu();
    return kreiraj.Kreiraj();
}

//Odeljak za storniranje domenskih objekata

public static bool StornirajNalog(Nalog n)
{
    StornirajNalog storniraj=new StornirajNalog();
    return storniraj.Storniraj(n);
}

public static bool StornirajKartu(ClanskaKarta c)
{
    StornirajKartu storniraj=new StornirajKartu();
    return storniraj.Storniraj(c);
}

public static bool StornirajUslugu(Usluga u)
{
    StornirajUslugu storniraj=new StornirajUslugu();
    return storniraj.Storniraj(u);
}

public static bool StornirajRacun(Racun r)
{
    StornirajRacun storniraj=new StornirajRacun();
    return storniraj.Storniraj(r);
}

public static bool StornirajRacunClana(ClanskaKarta ck,Racun r)
{
    StornirajClanRacun storniraj=new StornirajClanRacun();
    return storniraj.Storniraj(ck,r);
}

//Odeljak za pretrazivanje domenskih objekata

public static Nalog DaLiPostojiNalog(Nalog n)
{
    DaLiPostojiNalog dlp=new DaLiPostojiNalog();
    return dlp.DaLiPostoji(n);
}

public static ClanskaKarta DaLiPostojiKarta(ClanskaKarta c)
{
    DaLiPostojiKarta dlp=new DaLiPostojiKarta();
    return dlp.DaLiPostoji(c);
}

public static Racun DaLiPostojiRacun(Racun r)
{
    DaLiPostojiRacun dlp=new DaLiPostojiRacun();
    return dlp.DaLiPostoji(r);
}

public static Usluga DaLiPostojiUsluga(Usluga u)
{
    DaLiPostojiUsluga dlp=new DaLiPostojiUsluga();
    return dlp.DaLiPostoji(u);
}

//Odeljak za promenu domenskih objekata

public static bool PromeniNalog(Nalog n)
{
    PromeniNalog promeni=new PromeniNalog();
    return promeni.Promeni(n);
}

```

```

public static bool PromeniKartu(ClanskaKarta c)
{
    PromeniKartu promeni=new PromeniKartu();
    return promeni.Promeni(c);
}

public static bool PromeniUslugu(Usluga u)
{
    PromeniUslugu promeni=new PromeniUslugu();
    return promeni.Promeni(u);
}

//Odeljak za brisanje domenskih objekata

public static bool ObrisiUslugu(Usluga u)
{
    ObrisiUslugu obrisi=new ObrisiUslugu();
    return obrisi.Obrisi(u);
}

public static bool ObrisiNalog(Nalog n)
{
    ObrisiNalog obrisi=new ObrisiNalog();
    return obrisi.Obrisi(n);
}

//Provera sifre i korisnickog imena

public static object[] ProveriUserSifra(string usr,string sif)
{
    ProveriIspravnost pi=new ProveriIspravnost();
    return pi.Proveri(usr,sif);
}

//Operacije sa listom

public static void SnimiListu()
{
    ListOP lop=new ListOP();
    lop.SnimiListu();
}

public static void UcitajListu()
{
    ListOP lop=new ListOP();
    lop.UcitajListu();
}

public static void VратиUsluge(ArrayList lst)
{
    ListOP lop=new ListOP();
    lop.VратиUsluge(lst);
}

public static void VратиRacuneClana(ArrayList list,int brck)
{
    ListOP lop=new ListOP();
    lop.VратиRacuneClana(list,brck);
}

//Operacije sa datasetom

public static DataSet VратиClanove()
{
    VратиDataSet vds=new VратиDataSet();
    return vds.VратиClanove();
}

public static DataSet VратиRacune()
{

```

```

        VratiDataSet vds=new VratiDataSet();
        return vds.VratiRacune();
    }

    public static DataSet VratiNaloge()
    {
        VratiDataSet vds=new VratiDataSet();
        return vds.VratiNaloge();
    }

    public static DataSet VratiUsluge()
    {
        VratiDataSet vds=new VratiDataSet();
        return vds.VratiUsluge();
    }
}

```

Projektovanje strukture softverskog sistema-Domenske klase

Klasa OSOBA

```

public abstract class Osoba:IOpstiDomObj
{
    protected string jmbg;
    protected string ime;
    protected string adresa;
    protected string telefon;

    public abstract string JMBG
    {
        get;
        set;
    }

    public abstract string Ime
    {
        get;
        set;
    }

    public abstract string Adresa
    {
        get;
        set;
    }

    public abstract string Telefon
    {
        get;
        set;
    }

    public abstract string VratiImeKlase();
    public abstract string VratiVrednostAtributa();
    public abstract string VratiAtribut();
    public abstract string UslovPretrage();
    public abstract string PostaviZaUpdate();
    public abstract int VratiKljuc();
}

```

Klasa CLAN

```

[Serializable]
public class Clan:Osoba
{
    private int konfbr;
    private int brobuca;
}

```

```

public Clan(Clan c)
{
    this.brobeuce=c.brobeuce;
    this.konfbr=c.konfbr;
    base.adresa=c.adresa;
    base.ime=c.ime;
    base.jmbg=c.jmbg;
    base.telefon=c.telefon;
}

public Clan()
{
    this.brobeuce=0;
    this.konfbr=0;
    base.adresa=null;
    base.ime=null;
    base.jmbg=null;
    base.telefon=null;
}

public int KonfBr
{
    get{return this.konfbr;}
    set{this.konfbr=value;}
}

public int BrObuce
{
    get{return this.brobeuce;}
    set{this.brobeuce=value;}
}

public override string JMBG
{
    get{return base.jmbg;}
    set{base.jmbg=value;}
}

public override string Ime
{
    get{return base.ime;}
    set{base.ime=value;}
}

public override string Adresa
{
    get{return base.adresa;}
    set{base.adresa=value;}
}

public override string Telefon
{
    get{return base.telefon;}
    set{base.telefon=value;}
}

public override string VratiImeKlase()
{
    return "Clan";
}

public override string VratiVrednostAtributa()
{
    return null;
}

public override string VratiAtribut()
{
    //Nepotrebno u ovoj klasi
    return null;
}

```

```

    }

    public override string PostaviZaUpdate()
    {
        return null;
    }

    public override string UslovPretrage()
    {
        return "JMBG='"+this.jmbg+"'";
    }

    public override int VратиKljuc()
    {
        return Int32.Parse(this.jmbg);
    }
}

```

Klasa RADNIK

```

[Serializable]
public class Radnik:Osoba
{
    private string plata;

    public Radnik(Radnik r)
    {
        this.plata=r.plata;
        base.adresa=r.adresa;
        base.ime=r.ime;
        base.jmbg=r.jmbg;
        base.telefon=r.telefon;
    }

    public Radnik()
    {
        this.plata=null;
        base.adresa=null;
        base.ime=null;
        base.jmbg=null;
        base.telefon=null;
    }

    public string Plata
    {
        get{return plata;}
        set{plata=value;}
    }

    public override string JMBG
    {
        get{return base.jmbg;}
        set{base.jmbg=value;}
    }

    public override string Ime
    {
        get{return base.ime;}
        set{base.ime=value;}
    }

    public override string Adresa
    {
        get{return base.adresa;}
        set{base.adresa=value;}
    }

    public override string Telefon
    {
        get{return base.telefon;}
    }
}

```

```

        set{base.telefon=value;}
    }

    public override string VratiVrednostAtributa()
    {
        return null;
    }

    public override string VratiImeKlase()
    {
        return "Radnik";
    }

    public override string UslovPretrage()
    {
        return "JMBG='"+jmbg+"'";
    }

    public override string PostaviZaUpdate()
    {
        return null;
    }

    public override string VratiAtribut()
    {
        return null;
    }

    public override int VratiKljuc()
    {
        return Int32.Parse(base.jmbg);
    }
}

```

Klasa NALOG

```

[Serializable]
public class Nalog:IOpstiDomObj
{
    private string datumkreiranja;
    private string korisnickoime;
    private string korisnickasifra;
    private int brnaloga;
    private Radnik r;
    private bool administrator;

    public Nalog(Nalog n)
    {
        datumkreiranja=n.datumkreiranja;
        korisnickoime=n.korisnickoime;
        korisnickasifra=n.korisnickasifra;
        r=n.r;
        brnaloga=n.brnaloga;
    }

    public Nalog()
    {
        datumkreiranja=DateTime.Today.Date.ToString();
        korisnickoime=null;
        korisnickasifra=null;
        r=null;
        brnaloga=0;
    }

    public string DatumKreiranja
    {
        get{return datumkreiranja;}
        set{datumkreiranja=value;}
    }
}

```

```

public string KorisnickoIme
{
    get{return korisnickoime;}
    set{korisnickoime=value;}
}

public string KorisnickaSifra
{
    get{return korisnickasifra;}
    set{korisnickasifra=value;}
}

public int BrNaloga
{
    get{return brnaloga;}
    set{brnaloga=value;}
}

public bool Administrator
{
    get{return administrator;}
    set{administrator=value;}
}

public Radnik Radnik
{
    get{return r;}
    set{r=value;}
}

public string VratiVrednostAtributa()
{
    return
brnaloga+", "+r.JMBG+", "+r.Ime+", "+r.Adresa+", "+r.Telefon+", "+r.Plata+", "+this.
datumkreiranja+", "+this.korisnickoime+", "+this.korisnickasifra+", "+administrator;
}

public string VratiImeKlase()
{
    return "Nalog";
}

public string UslovPretrage()
{
    return "BrNaloga="+brnaloga;
}

public string PostaviZaUpdate()
{
    return
"BrNaloga="+brnaloga+", JMBG='"+r.JMBG+"', Ime='"+r.Ime+"', Adresa='"+r.Adresa+"', Telefon=' "
+r.Telefon+"', Plata='"+r.Plata+"', DatumKreiranja='"+datumkreiranja+"', KorIme='"+korisnick
oime+"', KorSifra='"+korisnickasifra+"', Administrator="+administrator;
}

public string VratiAtribut()
{
    return "BrNaloga";
}

public int VratiKljuc()
{
    return brnaloga;
}
}

```

Klasa CLANSKAKARTA

```

[Serializable]
public class ClanskaKarta:IOpstiDomObj
{
    private string datumizdavanja;
    private int brclkarte;
    private Clan cl;

    public ClanskaKarta(ClanskaKarta cla)
    {
        cl=cla.cl;
        brclkarte=cla.brclkarte;
        datumizdavanja=cla.datumizdavanja;
    }

    public ClanskaKarta()
    {
        cl=null;
        datumizdavanja=DateTime.Today.ToString();
        this.brclkarte=0;
    }

    public string DatumIzdavanja
    {
        get{return datumizdavanja;}
        set{datumizdavanja=value;}
    }

    public int BrClKarte
    {
        get{return brclkarte;}
        set{brclkarte=value;}
    }

    public Clan ClaN
    {
        get{return cl;}
        set{cl=value;}
    }

    public string VратиИмеКласе()
    {
        return "ClanskaKarta";
    }

    public string VратиVrednostAtributa()
    {
        return
brclkarte+"','"+cl.JMBG+"','"+cl.Ime+"','"+cl.Adresa+"','"+cl.Telefon+"','"+cl.KonfBr+"','"+c
l.BrObuce+"','"+datumizdavanja+"''";
    }

    public string VратиAtribut()
    {
        return "BrClKarte";
    }

    public string PostaviZaUpdate()
    {
        return
"BrClKarte="+brclkarte+",JMBG='"+cl.JMBG+"',Ime='"+cl.Ime+"',Adresa='"+cl.Adresa+"',Telef
on='"+cl.Telefon+"',KonfBr="+cl.KonfBr+",BrObuce="+cl.BrObuce+",DatumIzdavanja='"+datumiz
davanja+"''";
    }

    public string UslovPretrage()
    {
        return "BrClKarte="+brclkarte;
    }

    public int VратиKljuc()
    {

```

```
        return brclkarte;
    }
}
```

Klasa RACUN

```
[Serializable]
public class Racun:IOpstiDomObj
{
    private int brracuna;
    private float ukvrednost;
    private string datumizdavanja;
    private bool obradjen;
    private ArrayList srac;

    public Racun(Racun r)
    {
        this.srac=r.srac;
        this.datumizdavanja=r.datumizdavanja;
        this.obradjen=r.obradjen;
        this.ukvrednost=r.ukvrednost;
        this.brracuna=r.brracuna;
    }

    public Racun()
    {
        this.srac=new ArrayList();
        this.datumizdavanja=DateTime.Today.ToString();
        this.obradjen=false;
        this.ukvrednost=0;
        this.brracuna=0;
    }

    public int BrRacuna
    {
        get{return this.brracuna;}
        set{this.brracuna=value;}
    }

    public float UkVrednost
    {
        get{return this.ukvrednost;}
        set{this.ukvrednost=value;}
    }

    public int BrStavki
    {
        get{return this.srac.Count;}
    }

    public string DatumIzdavanja
    {
        get{return this.datumizdavanja;}
        set{this.datumizdavanja=value;}
    }

    public bool Obradjen
    {
        get{return this.obradjen;}
        set{this.obradjen=value;}
    }

    public void DodajStavku(StavkaRacuna sr)
    {
        this.srac.Add(sr);
        this.ukvrednost+=sr.Cena;
    }

    public StavkaRacuna VratiStavku(int i)
    {
        return (StavkaRacuna)this.srac[i];
    }
}
```

```

    }

    public string VratiVrednostAtributa()
    {
        return
this.brracuna+", "+this.ukvrednost+", '"+this.datumizdavanja+"', "+this.obradjen;
    }

    public string VratiImeKlase()
    {
        return "Racun";
    }

    public string UslovPretrage()
    {
        return "BrRacuna="+this.brracuna;
    }

    public string VratiAtribut()
    {
        return "BrRacuna";
    }

    public string PostaviZaUpdate()
    {
        return null;
    }

    public int UslovLista()
    {
        return this.brracuna;
    }

    public int VratiKljuc()
    {
        return this.brracuna;
    }
}

```

Klasa STAVKARACUNA

```

[Serializable]
public class StavkaRacuna:IOpstiDomObj
{
    private int brracuna;
    private int sifrausluge;
    private float cena;
    private string naziv;

    public StavkaRacuna(Racun rac,Usluga usl)
    {
        this.brracuna=rac.BrRacuna;
        this.sifrausluge=usl.Sifra;
        this.cena=usl.Cena;
        this.naziv=usl.Naziv;
    }

    public StavkaRacuna(Racun rac)
    {
        this.brracuna=rac.BrRacuna;
    }

    public int BrRacuna
    {
        get{return this.BrRacuna;}
    }

    public int Sifra
    {
        get{return this.sifrausluge;}
        set{this.sifrausluge=value;}
    }
}

```

```

    }

    public float Cena
    {
        get{return this.cena;}
        set{this.cena=value;}
    }

    public string Naziv
    {
        get{return this.naziv;}
        set{this.naziv=value;}
    }

    public string VratiImeKlase()
    {
        return "StavkaRacuna";
    }

    public string VratiVrednostAtributa()
    {
        return
this.brracuna+", "+this.sifrausluge+", "+this.cena+", '"+this.naziv+"'";
    }

    public string UslovPretrage()
    {
        return "BrRacuna="+this.brracuna;
    }

    public string VratiAtribut()
    {
        return "BrRacuna";
    }

    public string PostaviZaUpdate()
    {
        return null;
    }

    public int UslovLista()
    {
        return this.brracuna;
    }

    public int VratiKljuc()
    {
        return this.brracuna;
    }
}

```

Klasa USLUGA

```

[Serializable]
public class Usluga:IOpstiDomObj
{
    private int sifra;
    private string naziv;
    private float cena;

    public Usluga(Usluga u)
    {
        this.naziv=u.naziv;
        this.sifra=u.sifra;
        this.cena=u.cena;
    }

    public Usluga()
    {
        this.naziv=null;
        this.sifra=0;
    }
}

```

```

        this.cena=0F;
    }

    public int Sifra
    {
        get{return sifra;}
        set{sifra=value;}
    }

    public string Naziv
    {
        get{return naziv;}
        set{naziv=value;}
    }

    public float Cena
    {
        get{return cena;}
        set{cena=value;}
    }

    public string VratiVrednostAtributa()
    {
        return sifra+", '"+naziv+"', "+cena;
    }

    public string VratiImeKlase()
    {
        return "Usluga";
    }

    public string UslovPretrage()
    {
        return "Sifra="+sifra;
    }

    public string VratiAtribut()
    {
        return "Sifra";
    }

    public string PostaviZaUpdate()
    {
        return "Sifra="+sifra+",Naziv='"+naziv+"',Cena="+cena;
    }

    public int VratiKljuc()
    {
        return this.sifra;
    }
}

```

Klasa CLANRACUN

```

[Serializable]
public class ClanRacun:IOpstiDomObj
{
    private int brracuna;
    private int brclkarte;

    public ClanRacun(ClanskaKarta ck,Racun r)
    {
        this.brracuna=r.BrRacuna;
        this.brclkarte=ck.BrClKarte;
    }

    public int BrClKarte
    {
        get{return this.brclkarte;}
    }
}

```

```

public int BrRacuna
{
    get{return this.brracuna;}
}

public string VratiImeKlase()
{
    return "ClanRacun";
}

public string VratiVrednostAtributa()
{
    return this.brclkarte+","+this.brracuna;
}

public string UslovPretrage()
{
    return "BrClKarte="+this.brclkarte;
}

public string PostaviZaUpdate()
{
    return null;
}

public string VratiAtribut()
{
    return null;
}

public int UslovLista()
{
    return this.brclkarte;
}

public int VratiKljuc()
{
    return this.brclkarte;
}
}

```

Interfejs IOPSTIDOMOBJ

```

public interface IOpstiDomObj
{
    string VratiVrednostAtributa();
    string VratiImeKlase();
    string UslovPretrage();
    string VratiAtribut();
    string PostaviZaUpdate();
    int VratiKljuc();
}

```

Projektovanje ponašanja softverskog sistema-Sistemske operacije

```

public class KreirajNalog
{
    private DBBroker dbbr;

    public KreirajNalog()
    {
        //
        // TODO: Add constructor logic here
        //
        dbbr=new DBBroker();
    }

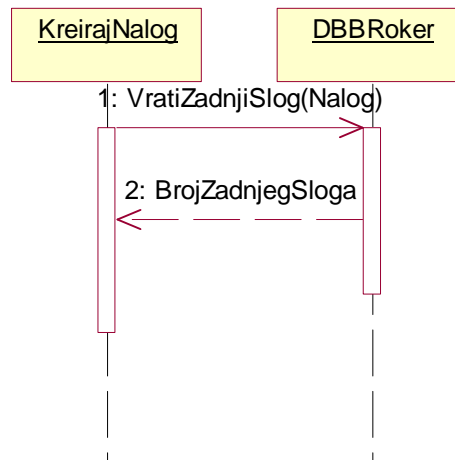
    public Nalog Kreiraj()
    {

```

```

        Nalog n=new Nalog();
        dbbr.OtvoriBazu();
        string pom=dbbr.VratiZadnjiSlog(n).ToString();
        dbbr.ZatvoriBazu();
        if(pom.Length==0)
            n.BrNaloga=1;
        else
            n.BrNaloga=Int32.Parse(pom)+1;
        Console.WriteLine("Nalog je uspesno kreiran.");
        Console.WriteLine("ENTER...");
        return n;
    }
}

```



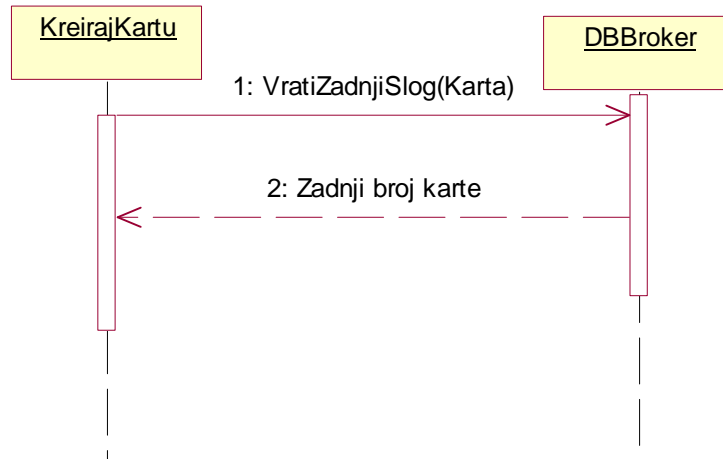
```

public class KreirajKartu
{
    private DBBroker dbbr;

    public KreirajKartu()
    {
        //
        // TODO: Add constructor logic here
        //
        dbbr=new DBBroker();
    }

    public ClanskaKarta Kreiraj()
    {
        ClanskaKarta c=new ClanskaKarta();
        dbbr.OtvoriBazu();
        string pom=dbbr.VratiZadnjiSlog(c).ToString();
        dbbr.ZatvoriBazu();
        if(pom.Length==0)
            c.BrClKarte=1;
        else
            c.BrClKarte=Int32.Parse(pom)+1;
        Console.WriteLine("Clanska karta je uspesno kreirana.");
        Console.WriteLine("ENTER...");
        return c;
    }
}

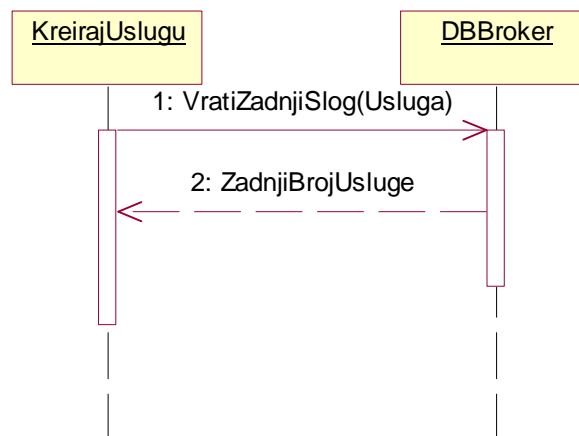
```



```

public class KreirajUslugu
{
    private DBBroker dbbr;
    public KreirajUslugu()
    {
        //
        // TODO: Add constructor logic here
        //
        dbbr=new DBBroker();
    }

    public Usluga Kreiraj()
    {
        Usluga u=new Usluga();
        dbbr.OtvoriBazu();
        string pom=dbbr.VratizadnjiSlog(u).ToString();
        dbbr.ZatvoriBazu();
        if(pom.Length==0)
            u.Sifra=100;
        else
            u.Sifra=Int32.Parse(pom)+1;
        return u;
    }
}
  
```



```

public class KreirajRacun
{
    private DBBroker dbbr;
    public KreirajRacun()
    {
  
```

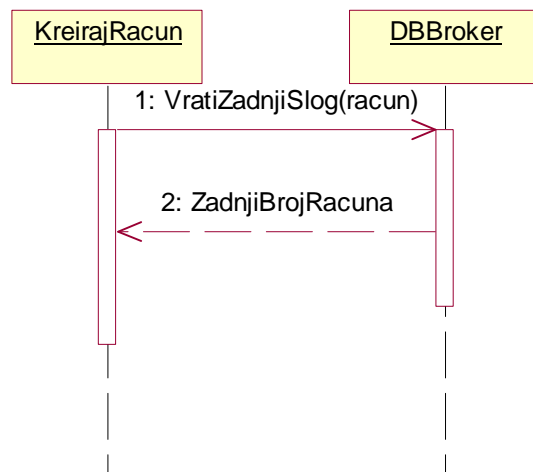
```

        //
        // TODO: Add constructor logic here
        //
        dbbr=new DBBroker();
    }

    public Racun Kreiraj()
    {
        Racun r=new Racun();
        dbbr.OtvoriBazu();
        string pom=dbbr.VratiZadnjiSlog(r).ToString();
        dbbr.ZatvoriBazu();
        if(pom.Length==0)
            r.BrRacuna=1;
        else
            r.BrRacuna=Int32.Parse(pom)+1;

        return r;
    }
}

```



```

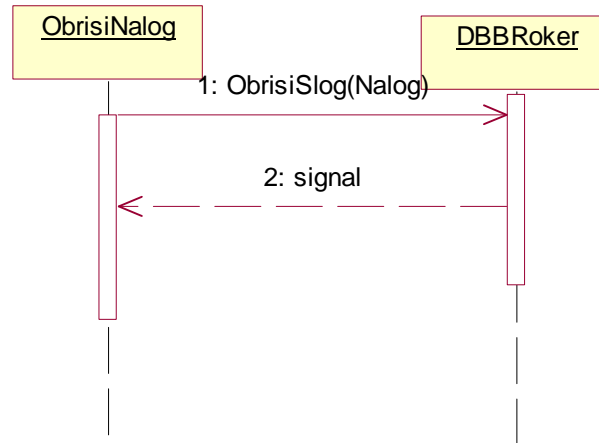
public class ObrisiNalog
{
    private DBBroker dbbr;

    public ObrisiNalog()
    {
        dbbr=new DBBroker();
    }

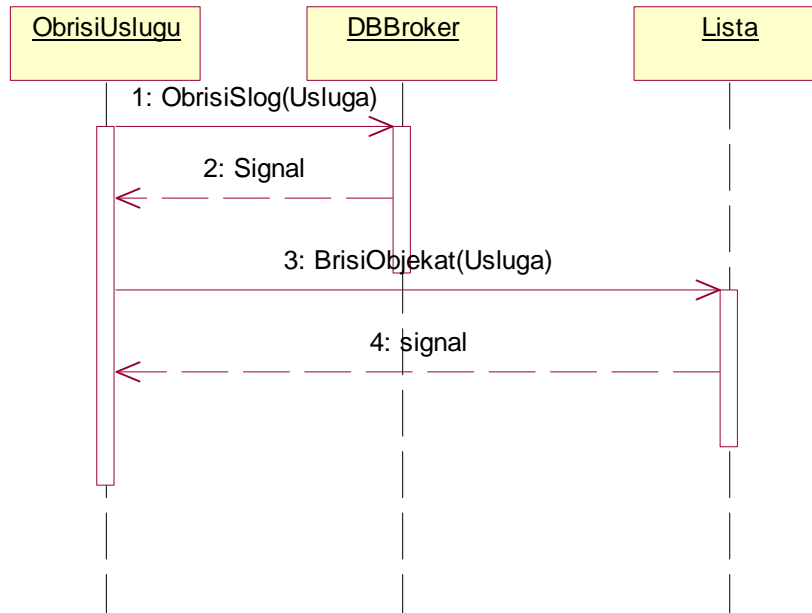
    public bool Obrisi(Nalog n)
    {
        try
        {
            dbbr.OtvoriBazu();
            dbbr.TransakcijaBegin();
            bool signal=dbbr.BrisiSlog(n);
            dbbr.TransakcijaCommit();
            Lista.BrisiObjekat(n);
            return signal;
        }
        catch(Exception e)
        {
            Console.WriteLine("Greska: "+e.Message);
            dbbr.TransakcijaRollBack();
            return false;
        }
        finally
        {
            dbbr.ZatvoriBazu();
        }
    }
}

```

```
}  
}  
}
```



```
public class ObrisiUslugu  
{  
    private DBBroker dbbr;  
  
    public ObrisiUslugu()  
    {  
        dbbr=new DBBroker();  
    }  
  
    public bool Obrisi(Usluga u)  
    {  
        try  
        {  
            dbbr.OtvoriBazu();  
            dbbr.TransakcijaBegin();  
            bool signal=dbbr.BrisiSlog(u);  
            dbbr.TransakcijaCommit();  
            Lista.BrisiObjekat(u);  
            return signal;  
        }  
        catch(Exception e)  
        {  
            Console.WriteLine("Greska:"+e.Message);  
            dbbr.TransakcijaRollBack();  
            return false;  
        }  
        finally  
        {  
            dbbr.ZatvoriBazu();  
        }  
    }  
}
```

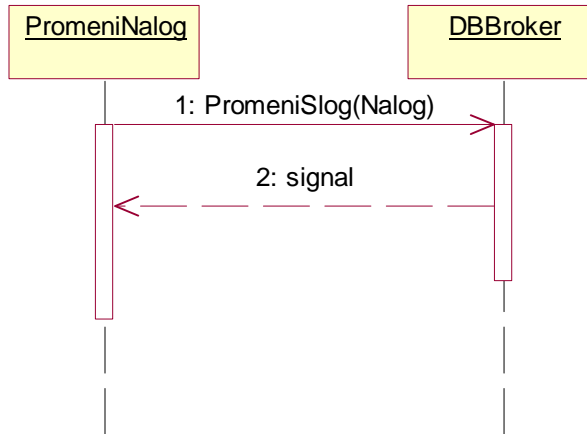


```

public class PromeniNalog
{
    private DBBroker dbbr;

    public PromeniNalog()
    {
        dbbr=new DBBroker();
    }

    public bool Promeni(Nalog n)
    {
        try
        {
            dbbr.OtvoriBazu();
            dbbr.TransakcijaBegin();
            bool signal=dbbr.PromeniSlog(n);
            dbbr.TransakcijaCommit();
            return signal;
        }
        catch(Exception e)
        {
            dbbr.TransakcijaRollBack();
            return false;
        }
        finally
        {
            dbbr.ZatvoriBazu();
        }
    }
}
  
```

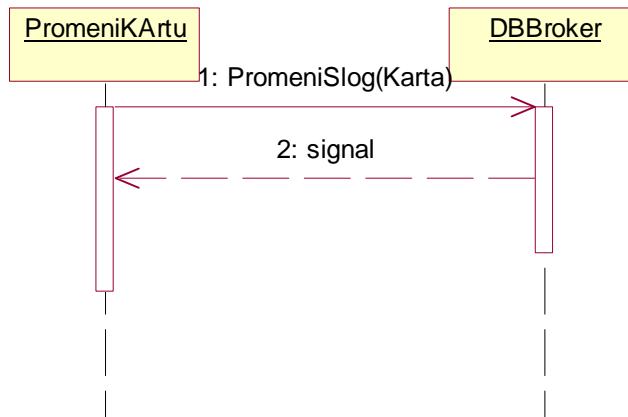


```

public class PromeniKartu
{
    private DBBroker dbbr;

    public PromeniKartu()
    {
        dbbr=new DBBroker();
    }

    public bool Promeni(ClanskaKarta ck)
    {
        try
        {
            dbbr.OtvoriBazu();
            dbbr.TransakcijaBegin();
            bool signal=dbbr.PromeniSlog(ck);
            dbbr.TransakcijaCommit();
            return signal;
        }
        catch(Exception e)
        {
            Console.WriteLine("GRESKA:"+e.Message);
            dbbr.TransakcijaRollBack();
            return false;
        }
        finally
        {
            dbbr.ZatvoriBazu();
        }
    }
}
  
```



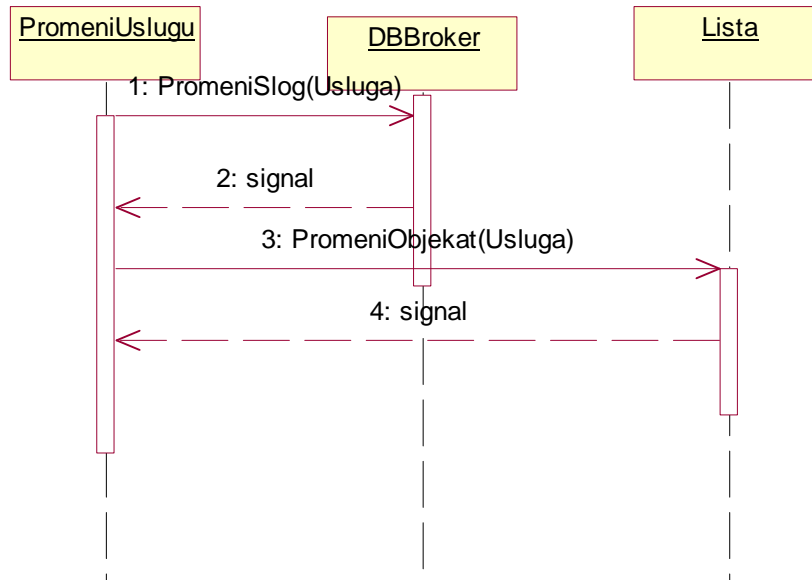
```

public class PromeniUslugu
{
    private DBBroker dbbr;

    public PromeniUslugu()
    {
        dbbr=new DBBroker();
    }

    public bool Promeni(Usluga u)
    {
        Lista.AzurirajObjekat(u);
        try
        {
            dbbr.OtvoriBazu();
            dbbr.TransakcijaBegin();
            bool signal=dbbr.PromeniSlog(u);
            dbbr.TransakcijaCommit();
            return signal;
        }
        catch(Exception e)
        {
            dbbr.TransakcijaRollBack();
            return false;
        }
        finally
        {
            dbbr.ZatvoriBazu();
        }
    }
}

```



```

public class StornirajNalog
{
    private DBBroker dbbr;

    public StornirajNalog()
    {
        dbbr=new DBBroker();
    }

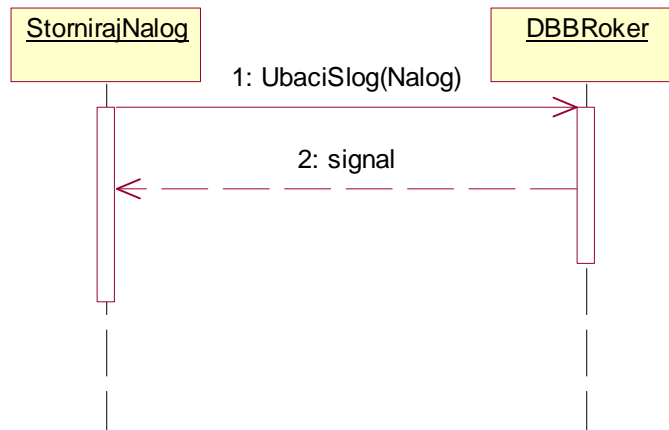
    public bool Storniraj(Nalog n)
    {
        try
        {

```

```

        dbbr.OtvoriBazu();
        dbbr.TransakcijaBegin();
        bool signal=dbbr.UbaciSlog(n);
        dbbr.TransakcijaCommit();
        return signal;
    }
    catch(Exception e)
    {
        Console.WriteLine("GRESKA:"+e.Message);
        dbbr.TransakcijaRollBack();
        return false;
    }
    finally
    {
        dbbr.ZatvoriBazu();
    }
}
}

```

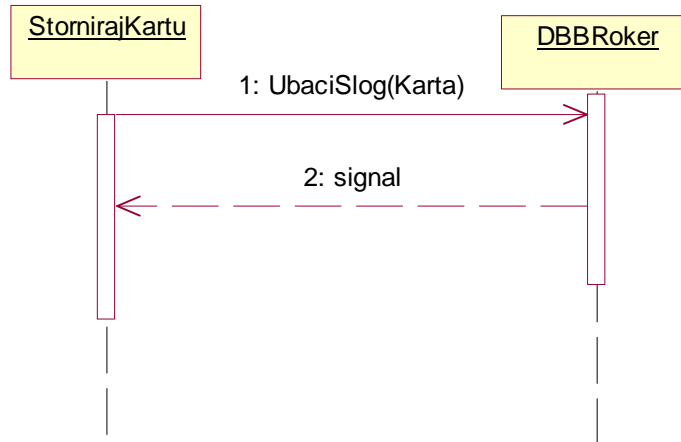


```

public class StornirajKartu
{
    private DBBroker dbbr;
    public StornirajKartu()
    {
        //
        // TODO: Add constructor logic here
        //
        dbbr=new DBBroker();
    }

    public bool Storniraj(ClanskaKarta c)
    {
        try
        {
            dbbr.OtvoriBazu();
            dbbr.TransakcijaBegin();
            bool signal=dbbr.UbaciSlog(c);
            dbbr.TransakcijaCommit();
            return signal;
        }
        catch(Exception e)
        {
            Console.WriteLine("GRESKA:"+e.Message);
            dbbr.TransakcijaRollBack();
            return false;
        }
        finally
        {
            dbbr.ZatvoriBazu();
        }
    }
}
}

```

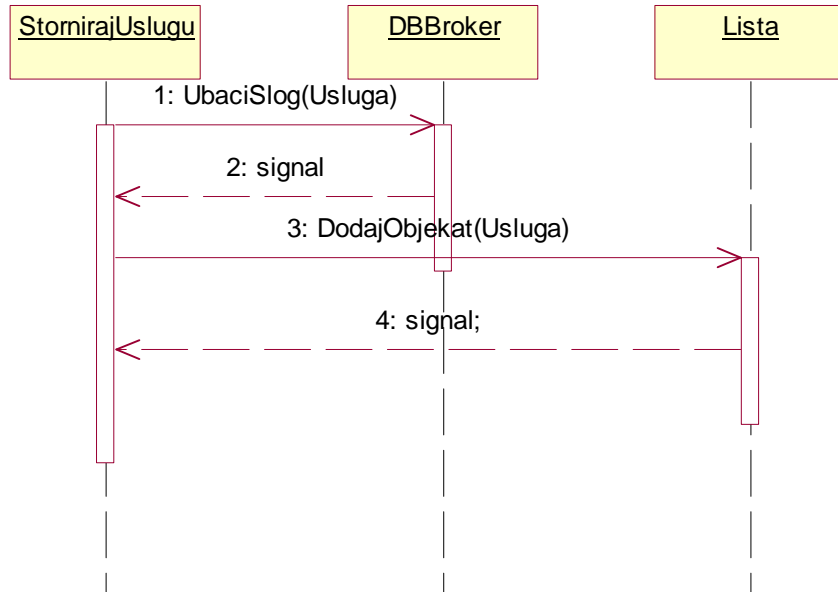


```

public class StornirajUslugu
{
    private DBBroker dbbr;

    public StornirajUslugu()
    {
        dbbr=new DBBroker();
    }

    public bool Storniraj(Usluga u)
    {
        try
        {
            dbbr.OtvoriBazu();
            dbbr.TransakcijaBegin();
            bool signal=dbbr.UbaciSlog(u);
            dbbr.TransakcijaCommit();
            Lista.DodajObjekat(u);
            return signal;
        }
        catch(Exception e)
        {
            Console.WriteLine("GRESKA:"+e.Message);
            dbbr.TransakcijaRollBack();
            return false;
        }
        finally
        {
            dbbr.ZatvoriBazu();
        }
    }
}
  
```

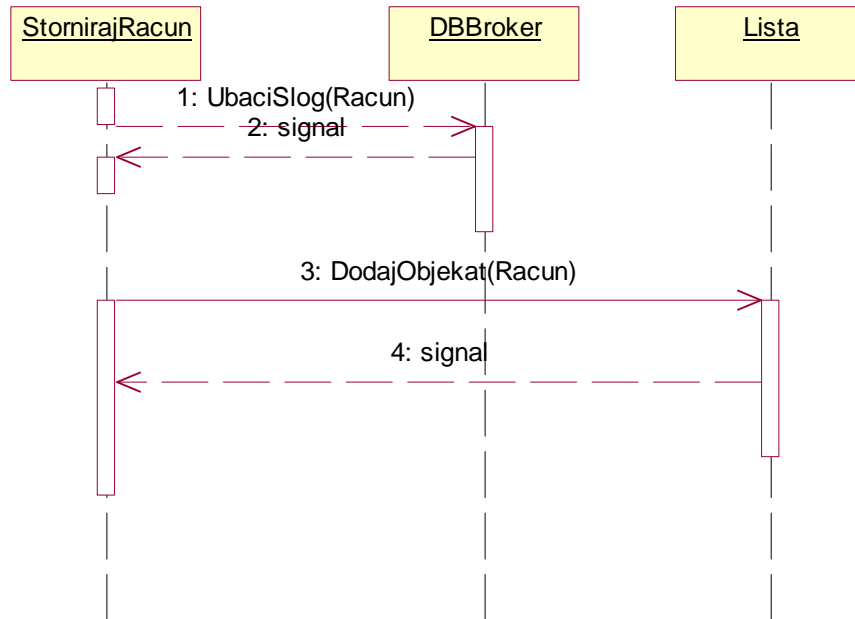


```

public class StornirajRacun
{
    private DBBroker dbbr;

    public StornirajRacun()
    {
        dbbr=new DBBroker();
    }

    public bool Storniraj(Racun r)
    {
        try
        {
            dbbr.OtvoriBazu();
            dbbr.TransakcijaBegin();
            dbbr.UbaciSlog(r);
            for(int i=0;i<r.BrStavki;i++)
            {
                StavkaRacuna pom=r.VratiStavku(i);
                dbbr.UbaciSlog(pom);
            }
            dbbr.TransakcijaCommit();
            Lista.DodajObjekat(r);
            return true;
        }
        catch(Exception e)
        {
            Console.WriteLine("GRESKA:"+e.Message);
            dbbr.TransakcijaRollBack();
            return false;
        }
        finally
        {
            dbbr.ZatvoriBazu();
        }
    }
}
  
```



```

public class StornirajClanRacun
{
    private DBBroker dbbr;

    public StornirajClanRacun()
    {
        dbbr=new DBBroker();
    }

    public bool Storniraj(ClanskaKarta ck,Racun r)
    {
        ClanRacun clr=new ClanRacun(ck,r);
        Lista.DodajObjekat(clr);
        Lista.DodajObjekat(r);
        try
        {
            dbbr.OtvoriBazu();
            dbbr.TransakcijaBegin();
            dbbr.UbaciSlog(clr);
            dbbr.UbaciSlog(r);
            for(int i=0;i<r.BrStavki;i++)
            {
                StavkaRacuna pom=r.VratiStavku(i);
                dbbr.UbaciSlog(pom);
            }
            dbbr.TransakcijaCommit();
            return true;;
        }
        catch(Exception e)
        {
            Console.WriteLine("GRESKA:"+e.Message);
            dbbr.TransakcijaRollBack();
            return false;
        }
        finally
        {
            dbbr.ZatvoriBazu();
        }
    }
}
  
```

```

public class ProveriIspravnost
{
    private DBBroker dbbr;
    public ProveriIspravnost()
    {
        //
        // TODO: Add constructor logic here
        //
        dbbr=new DBBroker();
    }

    public object[] Proveri(string usr,string sifra)
    {
        try
        {
            dbbr.OtvoriBazu();
            return dbbr.NadjiUserSifra(usr,sifra);
        }
        catch(Exception e)
        {
            return null;
        }
        finally
        {
            dbbr.ZatvoriBazu();
        }
    }
}

public class BackUp
{
    private Timer tajmer;
    private delegate void PozoviOperacije();

    public BackUp()
    {
        tajmer=new Timer(14400000);
        tajmer.AutoReset=true;
        tajmer.Elapsed+=new ElapsedEventHandler(TimerHandler);
    }

    private void TimerHandler(object sender,ElapsedEventArgs e)
    {
        PozoviOperacije po=new PozoviOperacije(BackUpBaze);
        po+=new PozoviOperacije(BackUpDatoteke);
        po();
    }

    private void BackUpBaze()
    {
        File.Copy(@"c:\Seminarski\SRCentar.mdb",@"C:\Seminarski\BackUp\SRCentar.mdb");
    }

    private void BackUpDatoteke()
    {
        File.Copy(@"C:\Seminarski\SRCentar.dat",@"C:\Seminarski\BackUp\SRCentar.dat");
    }

    private void StartTimer()
    {
        tajmer.Start();
    }

    private void StopTimer()
    {
        tajmer.Stop();
    }
}

```

```

public class VratiDataSet
{
    private DBBroker dbbr;
    public VratiDataSet()
    {
        //
        // TODO: Add constructor logic here
        //
        dbbr=new DBBroker();
    }

    public DataSet VratiClanove()
    {
        return dbbr.VratiClanove();
    }

    public DataSet VratiRacune()
    {
        return dbbr.VratiRacune();
    }

    public DataSet VratiNaloge()
    {
        return dbbr.VratiNaloge();
    }

    public DataSet VratiUsluge()
    {
        return dbbr.VratiUsluge();
    }
}

public class ListOP
{
    public ListOP()
    {
        //
        // TODO: Add constructor logic here
        //
    }

    public void SnimiListu()
    {
        Lista.SnimiUDatoteku();
    }

    public void UcitajListu()
    {
        Lista.UcitajIzDatoteke();
    }

    public void VratiUsluge(ArrayList lst)
    {
        Lista.VratiUslugu(lst);
    }

    public void VratiRacuneClana(ArrayList list,int brck)
    {
        Lista.VratiRacuneClana(list,brck);
    }
}

```

Projektovanje aplikacione logike-DataBase Broker

```

public class DBBroker
{
    private string konekcija;
    private OleDbConnection oledb;
    private OleDbCommand komanda;
    private OleDbTransaction transakcija;

```

```

public DBBroker()
{
    konekcija=@"Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=C:\Documents and Settings\Milos Pavlovic\My Documents\Visual Studio
Projects\Seminarski\SRCentar\SRCentar.mdb;Persist Security Info=False";
    oledb=new OleDbConnection(konekcija);
    komanda=null;
    transakcija=null;
}

public void TransakcijaBegin()
{
    transakcija=oledb.BeginTransaction();
}

public void TransakcijaCommit()
{
    transakcija.Commit();
}

public void TransakcijaRollBack()
{
    transakcija.Rollback();
}

public void OtvoriBazu()
{
    oledb.Open();
}

public void ZatvoriBazu()
{
    oledb.Close();
}

public bool UbaciSlog(IOpstiDomObj obj)
{
    try
    {
        string upit="INSERT INTO "+obj.VratiImeKlase()+" VALUES
("+obj.VratiVrednostAtributa()+")";
        komanda=new OleDbCommand(upit,oledb,transakcija);
        komanda.ExecuteNonQuery();
        return true;
    }
    catch(OleDbException e)
    {
        Console.WriteLine("Operacija storniranja nije
uspela:"+e.Message);
        return false;
    }
}

public bool BrisiSlog(IOpstiDomObj obj)
{
    try
    {
        string upit="DELETE FROM "+obj.VratiImeKlase()+" WHERE
"+obj.UslovPretrage();
        komanda=new OleDbCommand(upit,oledb,transakcija);
        komanda.ExecuteNonQuery();
        return true;
    }
    catch(OleDbException e)
    {
        Console.WriteLine("Operacija brisanja nije
uspela:"+e.Message);
        return false;
    }
}

```

```

public bool PromeniSlog(IOpstiDomObj obj)
{
    try
    {
        string upit=@"UPDATE "+obj.VratiImeKlase()+" SET
"+obj.PostaviZaUpdate()+" WHERE "+obj.UslovPretrage();
        komanda=new OleDbCommand(upit,oledb,transakcija);
        komanda.ExecuteNonQuery();
        return true;
    }
    catch(OleDbException e)
    {
        Console.WriteLine("Operacija promene nije
uspela:"+e.Message);
        return false;
    }
}

public object VratiZadnjiSlog(IOpstiDomObj obj)
{
    try
    {
        string upit=@"SELECT MAX("+obj.VratiAtribut()+") FROM
"+obj.VratiImeKlase();
        komanda=new OleDbCommand(upit,oledb);
        return komanda.ExecuteScalar();
    }
    catch(OleDbException e)
    {
        Console.WriteLine("Greska:"+e.Message);
        return null;
    }
}

public DataSet NadjiSlog(IOpstiDomObj obj)
{
    try
    {
        DataSet ds=new DataSet();
        string upit=@"SELECT * FROM "+obj.VratiImeKlase()+" WHERE
"+obj.UslovPretrage();
        OleDbDataAdapter da=new OleDbDataAdapter(upit,oledb);
        da.Fill(ds,obj.VratiImeKlase());
        return ds;
    }
    catch(Exception e)
    {
        Console.WriteLine("Graska:"+e.Message);
        return null;
    }
}

public object[] NadjiUserSifra(string usr,string sif)
{
    try
    {
        string upit="SELECT Administrator FROM Nalog WHERE
KorIme='"+usr+"' AND KorSifra='"+sif+"'";
        string upit1="SELECT COUNT(*) FROM Nalog WHERE
KorIme='"+usr+"' AND KorSifra='"+sif+"'";
        object[] pom=new object[2];
        OleDbDataReader citac=null;
        komanda=new OleDbCommand(upit1,oledb);
        pom[0]=komanda.ExecuteScalar();
        int i=(int)pom[0];
        if(i>0)
        {
            komanda=new OleDbCommand(upit,oledb);
            citac=komanda.ExecuteReader();
            citac.Read();
        }
    }
}

```

```

        pom[1]=citac[0];
        return pom;
    }
    else return null;
}
catch(OleDbException e)
{
    Console.WriteLine("Podaci nisu odgovarajuci:"+e.Message);
    return null;
}
}

public DataSet VratiClanove()
{
    try
    {
        string clan="SELECT * FROM ClanskaKarta";
        string clanracun="SELECT * FROM ClanRacun";
        DataSet ds=new DataSet("Clanovi");
        OleDbDataAdapter adapter=new OleDbDataAdapter(clan,oledb);
        adapter.Fill(ds,"Clan");
        adapter=new OleDbDataAdapter(clanracun,oledb);
        adapter.Fill(ds,"Racuni");
        ds.Relations.Add("Clanovi i
veze",ds.Tables["Clan"].Columns["BrClKarte"],
ds.Tables["Racuni"].Columns["BrClKarte"]);

        return ds;
    }
    catch(Exception e)
    {
        Console.WriteLine("Greska:"+e.Message);
        return null;
    }
}

public DataSet VratiRacune()
{
    try
    {
        string racun="SELECT * FROM Racun";
        string stavka="SELECT * FROM StavkaRacuna";
        DataSet ds=new DataSet("Racuni");
        OleDbDataAdapter adapter=new OleDbDataAdapter(racun,oledb);
        adapter.Fill(ds,"Racuni");
        adapter=new OleDbDataAdapter(stavka,oledb);
        adapter.Fill(ds,"Stavke");
        ds.Relations.Add("Racuni i
stavke",ds.Tables["Racuni"].Columns["BrRacuna"],
ds.Tables["Stavke"].Columns["BrRacuna"]);

        return ds;
    }
    catch(Exception e)
    {
        Console.WriteLine("Greska:"+e.Message);
        return null;
    }
}

public DataSet VratiNaloge()
{
    try
    {
        string nalog="SELECT * FROM Nalog";
        DataSet ds=new DataSet("Nalozi");
        OleDbDataAdapter adapter=new OleDbDataAdapter(nalog,oledb);
        adapter.Fill(ds,"Nalozi");

        return ds;
    }
    catch(Exception e)
    {

```

```

        Console.WriteLine("Greska:" + e.Message);
        return null;
    }
}

public DataSet VratiUsluge()
{
    try
    {
        string usluga="SELECT * FROM Usluga";
        DataSet ds=new DataSet("Usluge");
        OleDbDataAdapter adapter=new
OleDbDataAdapter(usluga,oledb);
        adapter.Fill(ds,"usluge");
        return ds;
    }
    catch(Exception e)
    {
        Console.WriteLine("Greska:" + e.Message);
        return null;
    }
}
}

```

Projektovanje aplikacione logike-List broker

```

public class Lista
{
    private static ArrayList ObjList=new ArrayList();

    public Lista()
    {
    }

    public static int BrElemenata
    {
        get{return ObjList.Count;}
    }

    public static void DodajObjekat(IOpstiDomObj obj)
    {
        ObjList.Add(obj);
    }

    public static int BrisiObjekat(IOpstiDomObj obj)
    {
        foreach(IOpstiDomObj pom in ObjList)
        {
            if(pom.VratiKljuc()==obj.VratiKljuc())
            {
                ObjList.Remove(obj);
                return 1;
            }
        }
        return 0;
    }

    public static IOpstiDomObj NadjiObjekat(IOpstiDomObj obj)
    {
        foreach(IOpstiDomObj pom in ObjList)
        {
            if(pom.VratiKljuc()==obj.VratiKljuc())
                return pom;
        }
        return null;
    }

    public static int AzurirajObjekat(IOpstiDomObj obj)
    {
        IOpstiDomObj pom=null;
    }
}

```

```

        for(int i=0;i<ObjList.Count;i++)
        {
            pom=(IOpstiDomObj)ObjList[i];
            if(pom.VratiKljuc()==obj.VratiKljuc())
            {
                ObjList[i]=obj;
                return 1;
            }
        }
        return 0;
    }

    public static void SnimiUDatoteku()
    {
        IFormatter formater=new BinaryFormatter();
        Stream tok=new
FileStream("Lista.dat",FileMode.Create,FileAccess.Write, FileShare.None);
        try
        {
            formater.Serialize(tok, ObjList);
        }
        catch(SerializationException e)
        {
            Console.WriteLine(e);
        }
        finally
        {
            tok.Close();
        }
        ObjList.Clear();
    }

    public static void UcitajIzDatoteke()
    {
        IFormatter formater=new BinaryFormatter();
        Stream tok=new
FileStream("Lista.dat",FileMode.Open,FileAccess.Read,FileShare.Read);
        try
        {
            ObjList=(ArrayList)formater.Deserialize(tok);
        }
        catch(SerializationException e)
        {
            Console.WriteLine(e);
        }
        finally
        {
            tok.Close();
        }
    }

    public static void VratiUslugu(ArrayList lst)
    {
        for(int i=0;i<ObjList.Count;i++)
        {
            if(ObjList[i] is Usluga)
                lst.Add(ObjList[i]);
        }
    }

    public static void VratiRacuneClana(ArrayList list,int brck)
    {
        for(int i=0;i<ObjList.Count;i++)
        {
            if(ObjList[i] is ClanRacun)
            {
                ClanRacun clr=(ClanRacun)ObjList[i];
                if(clr.BrClKarte==brck)
                {
                    for(int j=0;j<ObjList.Count;j++)
                    {

```



```

//Operacije za klasu Nalog
private static void KonvertujGrafikaNalog(Nalog nalog, TextBox
brnaloga, TextBox datkre,
    TextBox korime, TextBox korsif, TextBox jmbg, TextBox ime,
    TextBox adresa, TextBox telefon, TextBox plata, CheckBox admin)
{
    nalog.BrNaloga=Int32.Parse(brnaloga.Text);
    nalog.DatumKreiranja=datkre.Text;
    nalog.KorisnickoIme=korime.Text;
    nalog.KorisnickaSifra=korsif.Text;
    nalog.Administrator=admin.Checked;
    nalog.Radnik.JMBG=jmbg.Text;
    nalog.Radnik.Ime=ime.Text;
    nalog.Radnik.Adresa=adresa.Text;
    nalog.Radnik.Telefon=telefon.Text;
    nalog.Radnik.Plata=plata.Text;
}

private static void KonvertujNalogGrafika(Nalog nalog, TextBox
brnaloga, TextBox datkre,
    TextBox korime, TextBox korsif, TextBox jmbg, TextBox ime,
    TextBox adresa, TextBox telefon, TextBox plata, CheckBox admin)
{
    brnaloga.Text=nalog.BrNaloga.ToString();
    datkre.Text=nalog.DatumKreiranja;
    korime.Text=nalog.KorisnickoIme;
    korsif.Text=nalog.KorisnickaSifra;
    admin.Checked=nalog.Administrator;
    jmbg.Text=nalog.Radnik.JMBG;
    ime.Text=nalog.Radnik.Ime;
    adresa.Text=nalog.Radnik.Adresa;
    telefon.Text=nalog.Radnik.Telefon;
    plata.Text=nalog.Radnik.Plata;
}

public static void KreirajNalog(TextBox brnaloga, TextBox datkre,
    TextBox korime, TextBox korsif, TextBox jmbg, TextBox ime,
    TextBox adresa, TextBox telefon, TextBox plata, CheckBox admin)
{
    Nalog n=KontrolerAL.KreirajNalog();
    n.Radnik=new Radnik();
    KonvertujNalogGrafika(n, brnaloga, datkre, korime, korsif, jmbg,
        ime, adresa, telefon, plata, admin);
}

public static void StornirajNalog(TextBox brnaloga, TextBox datkre,
    TextBox korime, TextBox korsif, TextBox jmbg, TextBox ime,
    TextBox adresa, TextBox telefon, TextBox plata, CheckBox admin)
{
    Nalog n=new Nalog();
    n.Radnik=new Radnik();

    KonvertujGrafikaNalog(n, brnaloga, datkre, korime, korsif, jmbg, ime, adresa, telefon, plat
a, admin);

    if(KontrolerAL.StornirajNalog(n))
        MessageBox.Show("Nalog sacuvan");
    else
        MessageBox.Show("Nalog nije sacuvan");
}

public static void DaLiPostojiNalog(TextBox brnaloga, TextBox datkre,
    TextBox korime, TextBox korsif, TextBox jmbg, TextBox ime,
    TextBox adresa, TextBox telefon, TextBox plata, CheckBox admin)
{
    Nalog n=new Nalog();
    n.Radnik=new Radnik();

    KonvertujGrafikaNalog(n, brnaloga, datkre, korime, korsif, jmbg, ime, adresa, telefon, plat
a, admin);
}

```

```

        n=KontrolerAL.DaLiPostojiNalog(n);

        KonvertujNalogGrafika(n,brnaloga,datkre,korime,korsif,jmbg,ime,adresa,telefon,plata,admin);
    }

    public static void PromeniNalog(TextBox brnaloga,TextBox datkre,
        TextBox korime,TextBox korsif,TextBox jmbg,TextBox ime,
        TextBox adresa,TextBox telefon,TextBox plata,CheckBox admin)
    {
        Nalog n=new Nalog();
        n.Radnik=new Radnik();

        KonvertujGrafikaNalog(n,brnaloga,datkre,korime,korsif,jmbg,ime,adresa,telefon,plata,admin);

        if(KontrolerAL.PromeniNalog(n))
            MessageBox.Show("Nalog promenjen");
        else
            MessageBox.Show("Nalog nije promenjen");
    }

    public static void ObrisiNalog(TextBox brnaloga,TextBox datkre,
        TextBox korime,TextBox korsif,TextBox jmbg,TextBox ime,
        TextBox adresa,TextBox telefon,TextBox plata,CheckBox admin)
    {
        Nalog n=new Nalog();
        n.Radnik=new Radnik();

        KonvertujGrafikaNalog(n,brnaloga,datkre,korime,korsif,jmbg,ime,adresa,telefon,plata,admin);

        if(KontrolerAL.ObrisiNalog(n))
            MessageBox.Show("Nalog obrisan");
        else
            MessageBox.Show("Nalog nije obrisan");
    }

    //Operacije za klsu ClanskaKarta
    private static void KonvertujGrafikaKarta(ClanskaKarta ck,TextBox
brclk,TextBox datizd,
        TextBox jmbg,TextBox ime,TextBox adresa,TextBox telefon,TextBox
konfbr,TextBox brobuce)
    {
        ck.BrClKarte=Int32.Parse(brclk.Text);
        ck.DatumIzdavanja=datizd.Text;
        ck.ClaN.JMBG=jmbg.Text;
        ck.ClaN.Ime=ime.Text;
        ck.ClaN.Adresa=adresa.Text;
        ck.ClaN.Telefon=telefon.Text;
        ck.ClaN.KonfBr=Int32.Parse(konfbr.Text);
        ck.ClaN.BrObuce=Int32.Parse(brobuce.Text);
    }

    private static void KonvertujKartaGrafika(ClanskaKarta ck,TextBox
brclk,TextBox datizd,
        TextBox jmbg,TextBox ime,TextBox adresa,TextBox telefon,TextBox
konfbr,TextBox brobuce)
    {
        brclk.Text=ck.BrClKarte.ToString();
        datizd.Text=ck.DatumIzdavanja;
        jmbg.Text=ck.ClaN.JMBG;
        ime.Text=ck.ClaN.Ime;
        adresa.Text=ck.ClaN.Adresa;
        telefon.Text=ck.ClaN.Telefon;
        konfbr.Text=ck.ClaN.KonfBr.ToString();
        brobuce.Text=ck.ClaN.BrObuce.ToString();
    }

    public static void KreirajKartu(TextBox brclk,TextBox datizd,
        TextBox jmbg,TextBox ime,TextBox adresa,TextBox telefon,TextBox
konfbr,TextBox brobuce)

```

```

        {
            ClanskaKarta c=KontrolerAL.KreirajKartu();
            c.ClaN=new Clan();

            KonvertujKartaGrafika(c,brclk,datizd,jmbg,ime,adresa,telefon,konfbr,brobuce);
        }

        public static void StornirajKartu(TextBox brclk,TextBox datizd,
            TextBox jmbg,TextBox ime,TextBox adresa,TextBox telefon,TextBox
konfbr,TextBox brobuce)
        {
            ClanskaKarta c=new ClanskaKarta();
            c.ClaN=new Clan();

            KonvertujGrafikaKarta(c,brclk,datizd,jmbg,ime,adresa,telefon,konfbr,brobuce);
            if(KontrolerAL.StornirajKartu(c))
                MessageBox.Show("Clanska karta sacuvana");
            else
                MessageBox.Show("Clanska karta nije sacuvana");
        }

        public static void DaliPostaojiKarta(TextBox brclk,TextBox datizd,
            TextBox jmbg,TextBox ime,TextBox adresa,TextBox telefon,TextBox
konfbr,TextBox brobuce)
        {
            ClanskaKarta c=new ClanskaKarta();
            c.ClaN=new Clan();
            konfbr.Text="0";
            brobuce.Text="0";

            KonvertujGrafikaKarta(c,brclk,datizd,jmbg,ime,adresa,telefon,konfbr,brobuce);
            c=KontrolerAL.DaLiPostojiKarta(c);

            KonvertujKartaGrafika(c,brclk,datizd,jmbg,ime,adresa,telefon,konfbr,brobuce);
        }

        public static void PromeniKartu(TextBox brclk,TextBox datizd,
            TextBox jmbg,TextBox ime,TextBox adresa,TextBox telefon,TextBox
konfbr,TextBox brobuce)
        {
            ClanskaKarta c=new ClanskaKarta();
            c.ClaN=new Clan();

            KonvertujGrafikaKarta(c,brclk,datizd,jmbg,ime,adresa,telefon,konfbr,brobuce);
            if(KontrolerAL.PromeniKartu(c))
                MessageBox.Show("Clanska karta promenjena");
            else
                MessageBox.Show("Clanska karta nije promenjena");
        }

        //Operacije za klasu Usluga
        private static void KonvertujGrafikaUsluga(Usluga u,TextBox sifra,TextBox
naziv,TextBox cena)
        {
            u.Sifra=Int32.Parse(sifra.Text);
            u.Naziv=naziv.Text;
            u.Cena=Convert.ToSingle(cena.Text);
        }

        private static void KonvertujUslugaGrafika(Usluga u,TextBox sifra,TextBox
naziv,TextBox cena)
        {
            sifra.Text=u.Sifra.ToString();
            naziv.Text=u.Naziv;
            cena.Text=u.Cena.ToString();
        }

        public static void KreirajUslugu(TextBox sifra,TextBox naziv,TextBox cena)
        {
            Usluga u=KontrolerAL.KreirajUslugu();
            KonvertujUslugaGrafika(u,sifra,naziv,cena);
        }

```

```

    }

    public static void StornirajUslugu(TextBox sifra,TextBox naziv,TextBox
cena)
    {
        Usluga u=new Usluga();
        KonvertujGrafikaUsluga(u,sifra,naziv,cena);
        if(KontrolerAL.StornirajUslugu(u))
            MessageBox.Show("Usluga sacuvana");
        else
            MessageBox.Show("Usluga nije sacuvana");
    }

    public static void ObrisiUslugu(TextBox sifra,TextBox naziv,TextBox cena)
    {
        Usluga u=new Usluga();
        KonvertujGrafikaUsluga(u,sifra,naziv,cena);
        if(KontrolerAL.ObrisiUslugu(u))
            MessageBox.Show("Usluga obrisana");
        else
            MessageBox.Show("Usluga nije obrisana");
    }

    public static void DaLiPostojiUsluga(TextBox sifra,TextBox naziv,TextBox
cena)
    {
        Usluga u=new Usluga();
        cena.Text="0";
        KonvertujGrafikaUsluga(u,sifra,naziv,cena);
        u=KontrolerAL.DaLiPostojiUsluga(u);
        KonvertujUslugaGrafika(u,sifra,naziv,cena);
    }

    public static void PromeniUslugu(TextBox sifra,TextBox naziv,TextBox cena)
    {
        Usluga u=new Usluga();
        KonvertujGrafikaUsluga(u,sifra,naziv,cena);
        if(KontrolerAL.PromeniUslugu(u))
            MessageBox.Show("Usluga promenjena");
        else
            MessageBox.Show("Usluga nije promenjena");
    }

    //Operacije za klasu Racun
    private static void KonvertujGrafikaRacun(Racun r,TextBox brrac,
        TextBox datizd,TextBox ukvred,CheckBox obradjen,ListView stavke)
    {
        r.BrRacuna=Int32.Parse(brrac.Text);
        r.DatumIzdavanja=datizd.Text;
        r.Obradjen=obradjen.Checked;
        for(int i=0;i<stavke.Items.Count;i++)
        {
            StavkaRacuna srac=new StavkaRacuna(r);
            srac.Sifra=Int32.Parse(stavke.Items[i].Text);
            srac.Naziv=stavke.Items[i].SubItems[1].Text;

srac.Cena=Convert.ToSingle(stavke.Items[i].SubItems[2].Text);
            r.DodajStavku(srac);
        }
    }

    private static void KonvertujRacunGrafika(Racun r,TextBox brrac,
        TextBox datizd,TextBox ukvred,CheckBox obradjen,ListView stavke)
    {
        brrac.Text=r.BrRacuna.ToString();
        datizd.Text=r.DatumIzdavanja;
        obradjen.Checked=r.Obradjen;
        for(int i=0;i<r.BrStavki;i++)
        {
            stavke.Items.Add(r.VratiStavku(i).Sifra.ToString());
            stavke.Items[i].SubItems.Add(r.VratiStavku(i).Naziv);
        }
    }

```

```

stavke.Items[i].SubItems.Add(r.VratiStavku(i).Cena.ToString());
    }
    ukvred.Text=r.UkVrednost.ToString();
}

public static void KreirajRacun(TextBox brrac,
    TextBox datizd,TextBox ukvred,CheckBox obradjen,ListView stavke)
{
    Racun r=KontrolerAL.KreirajRacun();
    KonvertujRacunGrafika(r,brrac,datizd,ukvred,obradjen,stavke);
}

public static void StornirajRacun(TextBox brrac,
    TextBox datizd,TextBox ukvred,CheckBox obradjen,ListView stavke)
{
    Racun r=new Racun();
    KonvertujGrafikaRacun(r,brrac,datizd,ukvred,obradjen,stavke);
    if(KontrolerAL.StornirajRacun(r))
        MessageBox.Show("Racun sacuvan");
    else
        MessageBox.Show("Racun nije sacuvan");
}

public static void StornirajRacunClana(string [] racun,ListViewItem[]
stavke,TextBox brclk)
{
    Racun r=new Racun();
    r.BrRacuna=int.Parse(racun[0]);
    r.DatumIzdavanja=racun[2];
    r.Obradjen=bool.Parse(racun[3]);
    for(int i=0;i<stavke.Length;i++)
    {
        StavkaRacuna srac=new StavkaRacuna(r);
        srac.Sifra=int.Parse(stavke[i].SubItems[0].Text);
        srac.Cena=float.Parse(stavke[i].SubItems[2].Text);
        srac.Naziv=stavke[i].SubItems[1].Text;
        r.DodajStavku(srac);
    }
    ClanskaKarta c=new ClanskaKarta();
    c.BrClKarte=int.Parse(brclk.Text);
    if(KontrolerAL.StornirajRacunClana(c,r))
        MessageBox.Show("Racun clana sacuvan");
    else
        MessageBox.Show("Racun clana nije sacuvan");
}

public static void DaLiPostojiRacun(TextBox brrac,
    TextBox datizd,TextBox ukvred,CheckBox obradjen,ListView stavke)
{
    Racun r=new Racun();
    KonvertujGrafikaRacun(r,brrac,datizd,ukvred,obradjen,stavke);
    r=KontrolerAL.DaLiPostojiRacun(r);
    KonvertujRacunGrafika(r,brrac,datizd,ukvred,obradjen,stavke);
}

//Provera korisnickog imena i sifre
public static object[] ProveriUserSifra(TextBox usr,TextBox sif)
{
    return KontrolerAL.ProveriUserSifra(usr.Text,sif.Text);
}

//Operacije za rad sa listom
public static void UcitajListu()
{
    KontrolerAL.UcitajListu();
}

public static void SnimiListu()
{
    KontrolerAL.SnimiListu();
}

```

```

    }

    public static void NapuniListViewNalepnica(ListView l)
    {
        ArrayList lst=new ArrayList();
        KontrolerAL.VratiUsluge(lst);
        for(int i=0;i<lst.Count;i++)
        {
            Usluga u=(Usluga)lst[i];
            l.Items.Add(u.Sifra.ToString());
            l.Items[i].SubItems.Add(u.Naziv);
            l.Items[i].SubItems.Add(u.Cena.ToString());
        }
    }

    public static void NapuniListViewKarta(ListView l,TextBox brrac)
    {
        ArrayList lst=new ArrayList();
        int broj=int.Parse(brrac.Text);
        KontrolerAL.VratiRacuneClana(lst,broj);
        for(int i=0;i<lst.Count;i++)
        {
            Racun r=(Racun)lst[i];
            l.Items.Add(r.BrRacuna.ToString());
            l.Items[i].SubItems.Add(r.UkVrednost.ToString());
            l.Items[i].SubItems.Add(r.DatumIzdavanja);
            l.Items[i].SubItems.Add(r.Obradjen.ToString());
        }
    }

    public static void VratiUslugeNaRacun(ListViewItem[] items,ListView
    Stavke,TextBox vred)
    {
        float ukvred=float.Parse(vred.Text);
        float cena=0;
        for(int i=0;i<items.Length;i++)
        {
            Stavke.Items.Add(items[i].Text);
            Stavke.Items[i].SubItems.Add(items[i].SubItems[1].Text);
            Stavke.Items[i].SubItems.Add(items[i].SubItems[2].Text);
            cena+=float.Parse(items[i].SubItems[2].Text);
        }
        ukvred+=cena;
        vred.Text=ukvred.ToString();
    }

    public static void VratiRacunNaKartu(ListView racuni,string[] racun,int
    pozicija)
    {
        racuni.Items.Add(racun[0]);
        racuni.Items[pozicija].SubItems.Add(racun[1]);
        racuni.Items[pozicija].SubItems.Add(racun[2]);
        racuni.Items[pozicija].SubItems.Add(racun[3]);
    }

    //Rad sa Data Setom

    public static DataSet VratiClanove()
    {
        return KontrolerAL.VratiClanove();
    }

    public static DataSet VratiRacune()
    {
        return KontrolerAL.VratiRacune();
    }

    public static DataSet VratiNaloge()
    {
        return KontrolerAL.VratiNaloge();
    }

```

```
    }  
    public static DataSet VratiUsluge()  
    {  
        return KontrolerAL.VratiUsluge();  
    }  
}
```