INFORMACIONI SISTEMI

DRUGA FAZA PROJEKTA

INFORMACIONI SISTEM ZA SERVIS MASOVNOG SLANJA SMS PORUKA

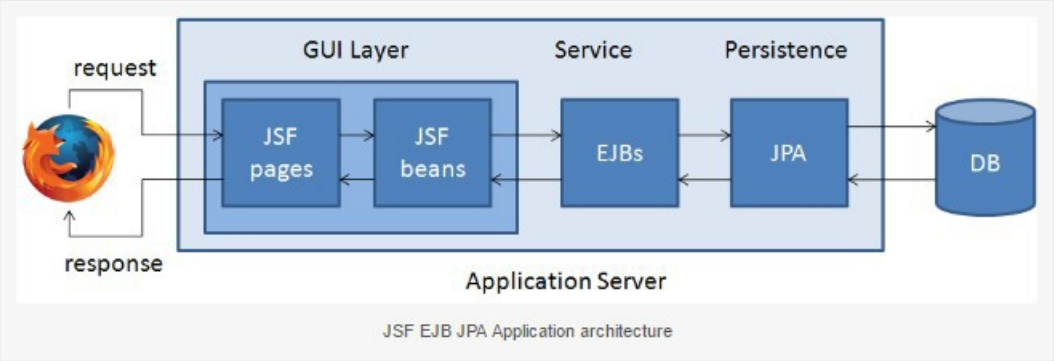
MENTOR: M.Sc. Nenad Petrović
STUDENT: Aleksandra Ristič, 16 313

Tekst zadatka:

182. Informacioni sistem za servis masovnog slanja reklamnih SMS poruka. Korisnik treba da definiše sadržaj poruke i listu brojeva mobilnih telefona na koji se šalju poruke. Svaki od korisnika ima ugovor sa nekim od operatera mobilne telefonije, tako da je u bazi potrebno voditi evidenciju o tome. Za svaku uslugu masovnog slanja SMS poruka se kreira faktura, koja sadrži totalnu cenu usluge, koja je definisana kao proizvod broja poslatih poruka i cene jedinicne poruke, koja zavisi od odabranog operatera. Takode, uzeti u obzir da za svakog operatora postoji i skala popusta
(recimo, za više od N poslatih poruka, primeniti popust od M%).

Obrazac: tok akcije

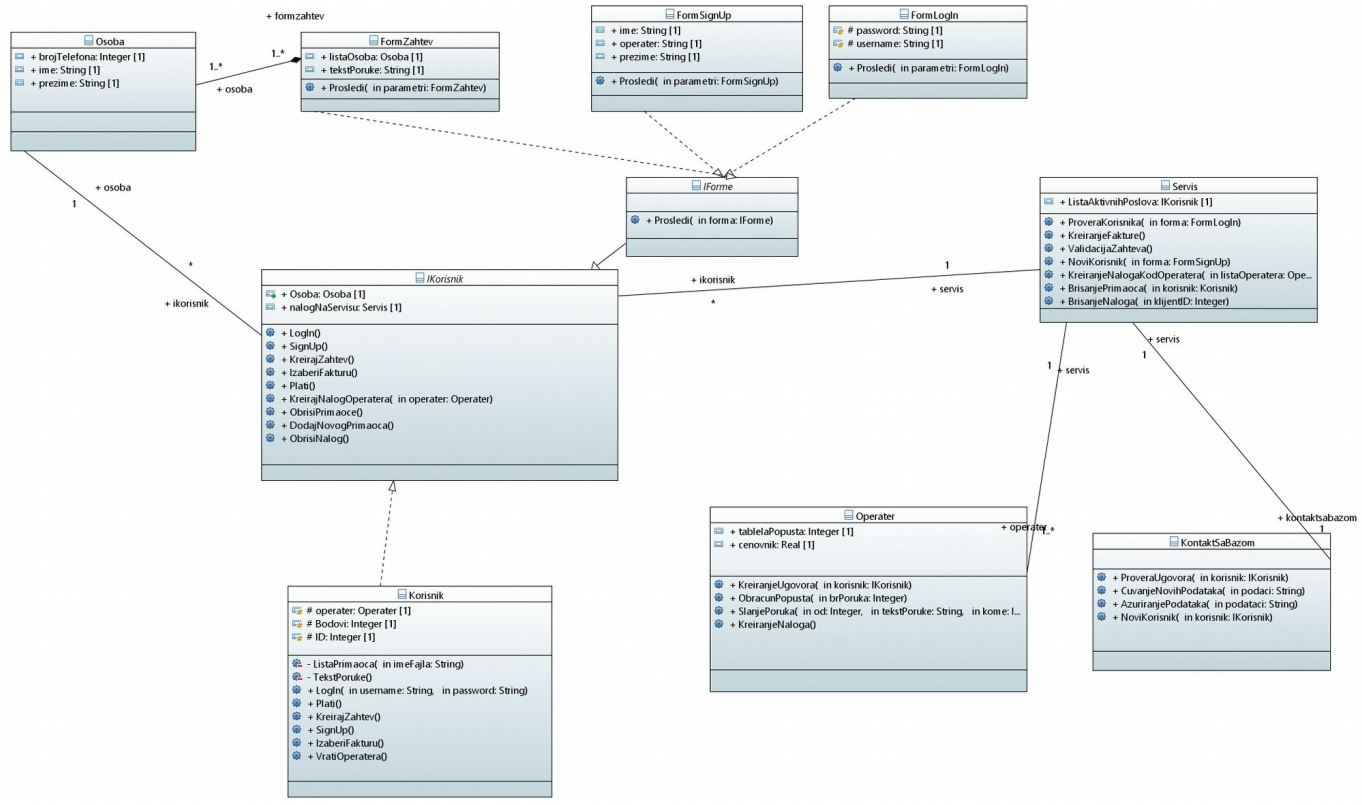
Arhitektura aplikacije



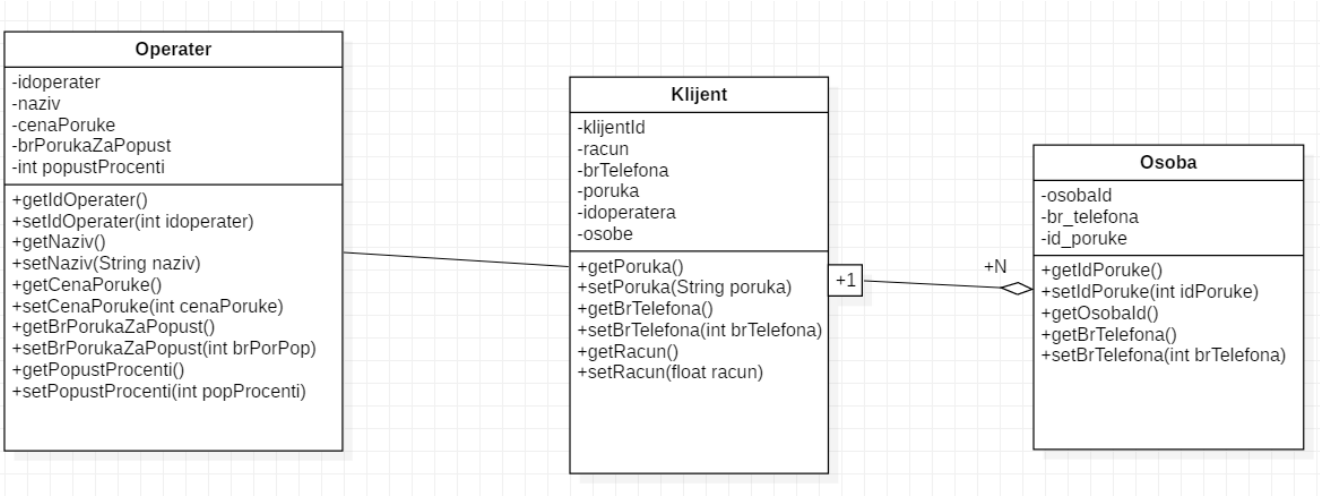
JSF EJB JPA Application architecture

Dijagram klasa

Inicijalni klasni dijagram:



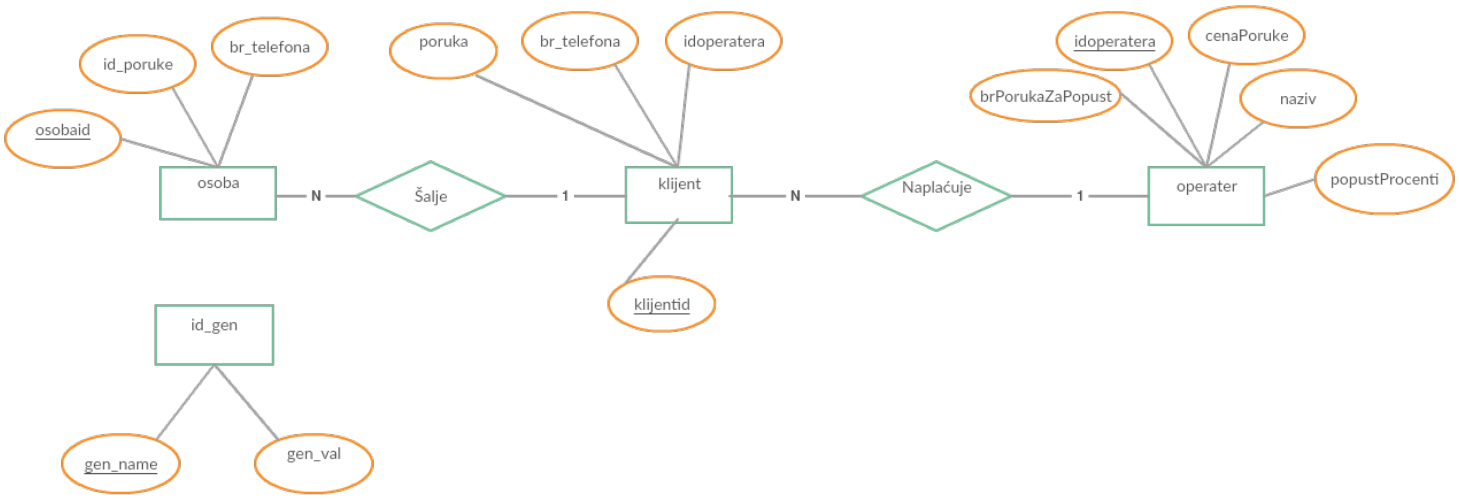
Primena obrasca:
S obzirom da je u zadatku dat obrazac *Tok akcije* akcenat je stavljen na *Flores*-ovoj drugoj fazi: *pregovori*. Naime, na osnovu podataka koje je korisnik sistema uneo vrši se obračun njegovog računa u koji je uključen i popust ukoliko je on ostvaren. Kako je ovaj sistem samo posrednik između operatera i korisnika, mogućnost smanjenja računa korisnika je onemogućena. Pregovori se zasnivaju na tome da ukoliko korisniku ne odgovara ponuđena cena njemu se daje savet da smanji broj targetiranih osoba čime će se smanjiti negov račun. Ukoliko se odluči za tu opciju sistem će ga odvesti na stranicu koja omogućava dodavanje, brisanje i izmenu korisnika. Na kraju pregovora korisniku se potvrdjuje njegov zahtev i obaveštava se da će usluga biti izvršena nakon uplate.



Šema baze podataka

Šema baze podataka pod nazivom firsttable nalazi se na MySQLServer-u.
Baza sadrži četiri tabele. Tabela ***klijent*** sadrži sve neophodne podatke vezane za klijenta kao i id operatera čiji je korisnik. U tabeli ***osoba*** se nalaze svi podaci vezani za targetiranu osobu kojoj će poruka biti poslata, kao i id poruke. Id poruke poklapa se sa id-jem klijenta, jer svaki klijent može imati samo jednu poruku pri jednom prijavljivanju na sistem. Tabela ***operater*** sadrzi cenovnik za odredjenog operatera, ova tabla služi samo za čitanje podataka. Takođe, tu je i tabela ***id_gen*** za automatsko generisanje primarnih ključeva za gore navedene tabele.

Na slici ispod se nalazi EER dijagram baze.



Testiranje aplikacije

Napomena: Da bi testiranje bilo moguće potrebno je da postoji kreirana tabela sa unetim podacima o operateru.

Testirani su sledeći slučajevi korišćenja:
Klijent:

- Dodavanje novog klijenta zajedno sa osobama
- Obračunavanje računa klijenta

Osoba:

- Azuriranje osobe
- Brisanje već postojeće osobe

Operater:

- Preuzimanje informacija o operateru

1. Dodavanje novog klijenta zajedno sa osobama

Opis:

Test treba da obezbedi dodavanje novog klijenta zajedno sa listom osoba u već postojeću bazu.

Koraci testa:

- Unos podataka o novom klijentu
- Pritiskom na dugme „Dalje“ se prebacuje na sledecu stranu u okviru koje se dodaju informacije o osobi.

- Klikom na dugme “Dodaj“ izvršava se funkcija klase *KlijentController* koja obezbeđuje da se nova osoba doda u listu osoba klijenta.

```
public void addOsoba(Osoba o)
{
	klijent.getOsobe().add(o);
}
```

- Pritiskom na dugme „Dalje“ pozova se funkcija iz klase *KlijentController* koja će novog klijenta dodati u bazu sa novim id-jem.

```
@EJB
private KlijentService service;
public void saveKlijent(Klijent kli)
{
	service.addKlijent(kli);
}
```

- U klasi *Klijent* definisana je unidirectional-na OneToMany veza između klijenta i liste osoba , dodavanjem klijenta u bazu biće dodate i osobe iz liste u tabeli osobe.

```
@OneToMany(
	cascade = CascadeType.ALL,
	orphanRemoval = true
	  )
```

Očekivani rezultati:

Jedan novi klijent u bazi i jedna ili više osoba u bazi u tabeli *osoba*

2. Obračunavanje računa klijenta

Napomena 1. :

Pre početka testa potrebno je imati u bazi u svim tabelama bar po jednu informaciju.

Napomena 2. :

U okviru ovog testa se izvodi i testiranje „Preuzimanje informacija o operateru“

Opis:

Test treba da obezbedi obračunavanje računa klijenta

Koraci testa:

- Pritiskom na dugme „Dalje“ u pozadini se izvršava funkcija iz klase *KlijentController*

```
public void izracunajRacun()
{
	int brOsoba=klijent.getBrojOsoba();
	Operater op=opservice.getOperater(klijent.getIdoperatera());
	int cenaPoruke=op.getCenaPoruke();
	float rez;
	rez=(float) (brOsoba*cenaPoruke);
	if(brOsoba>=op.getBrPorukaZaPopust())
	{
		rez=rez-op.getPopustProcenti()*rez;
	}
	klijent.setRacun(rez);
}
```

- Ostvaruje se interakcija sa funkcijom iz klase *OperaterService* čija je implementacija navedena u nastavku

```
@EJB
private OperaterService opservice;
```

```
public Operater getOperater(int id) {
	Operater op=em.find(Operater.class, id);
	return op;
}
```

Očekivani rezultati:

Iznos računa korisnika sa dodanim umanjenjem u zavisnosti od definisanog količinskog popusta određenog operatera.

3.Brisanje već postojeće osobe

Opis:

Test treba da obezbedi brisanje postojeće osobe iz baze.

Koraci testa:

- Unos id-a osobe koja se briše iz baze
- Pritiskom na dugme „Obriši“ u pozadini se izvršava funkcija iz klase *OsobaController*

```
@EJB
private OsobaService service;
public void removeOsoba(int id)
{
	service.deleteOsoba(id);
}
```

- Ostvaruje se interakcija sa funkcijom iz klase *OsobaService* čija je implementacija navedena u nastavku

```
private EntityManager em;
public void deleteOsoba(int id)
{
	Osoba e=em.find(Osoba.class, id);
	em.remove(e);

}
```

Očekivani rezultati:

Osoba nije u bazi.

4. Ažuriranje već postojeće osobe

Opis:

Test treba da obezbedi ažuriranje već postojeće osobe u bazi.

Koraci testa:

- Unos id-a osobe čiji se podaci ažuriraju
- Unos novih podataka osobe
- Pritiskom na dugme „Ažuriraj“ u pozadini se izvršava funkcija iz klase *OsobaController*

```
@EJB
private OsobaService service;
```

```
public void updateOsoba(int id, int newBrTelefona)
{
	service.updateOsoba(id, newBrTelefona);
}
```

- Ostvaruje se interakcija sa funkcijom iz klase *OsobaService* čija je implementacija navedena u nastavku

```
public void updateOsoba(int id, int newbrTelefona) {
	Osoba e=em.find(Osoba.class, id);
	e.setBrTelefona(newbrTelefona);
	}
```

Očekivani rezultati:

Ažuriran broj telefona osobe.