Univerzitet u Beogradu

Fakultet organizacionih nauka

Laboratorija za softversko inženjerstvo

Seminarski rad

Predmet: Softverski proces

Tema: Softverski sistem za evidenciju faktura o prodaji

|  |  |
| --- | --- |
| Profesor: | Student: |
| prof dr Siniša Vlajić | Jelena Đorđević 2013/3716 |

Beograd, 2015.

**Sadržaj**

[Prikupljanje zahteva 4](#_Toc432710614)

[Verbalni opis 4](#_Toc432710615)

[Slučajevi korišćenja 4](#_Toc432710616)

[Slučaj korišćenja 1: Unos nove fakture 5](#_Toc432710617)

[Slučaj korišćenja 2: Izmena fakture 6](#_Toc432710618)

[Slučaj korišćenja 3: Storniranje fakture 7](#_Toc432710619)

[Slučaj korišćenja 4: Pretraga faktura 8](#_Toc432710620)

[Slučaj korišćenja 5: Unos novog proizvoda 9](#_Toc432710621)

[Slučaj korišćenja 6: Izmena proizvoda 10](#_Toc432710622)

[Slučaj korišćenja 7: Unos poslovnog partnera 11](#_Toc432710623)

[Slučaj korišćenja 8: Izmena poslovnog partnera 12](#_Toc432710624)

[Slučaj korišćenja 9: Prijavljivanje korisnika 13](#_Toc432710625)

[Analiza 14](#_Toc432710626)

[Dijagrami sekvenci 14](#_Toc432710627)

[DS1: Dijagram sekvenci slučaja korišćenja –Unos fakture 14](#_Toc432710628)

[DS2: Dijagram sekvenci slučaja korišćenja – Izmena fakture 16](#_Toc432710629)

[DS3: Dijagram sekvenci slučaja korišćenja – Storniranje fakture 18](#_Toc432710630)

[DS4: Dijagram sekvenci slučaja korišćenja – Pretraga faktura 20](#_Toc432710631)

[DS5: Dijagram sekvenci slučaja korišćenja – Unos novog proizvoda 21](#_Toc432710632)

[DS6: Dijagram sekvenci slučaja korišćenja – Izmena proizvoda 23](#_Toc432710633)

[DS7: Dijagram sekvenci slučaja korišćenja – Unos poslovnog partnera 25](#_Toc432710634)

[DS8: Dijagram sekvenci slučaja korišćenja – Izmena poslovnog partnera 26](#_Toc432710635)

[DS9: Dijagram sekvenci slučaja korišćenja – Prijavljivanje korisnika 28](#_Toc432710636)

[Ugovori o sistemskim operacijama 29](#_Toc432710637)

[Ugovor UG1: KreirajNovu 29](#_Toc432710638)

[Ugovor UG2: Zapamti 29](#_Toc432710639)

[Ugovor UG3: Obradi 29](#_Toc432710640)

[Ugovor UG4: PretraziFakture 29](#_Toc432710641)

[Ugovor UG5: NadjiFakturu 29](#_Toc432710642)

[Ugovor UG6: Storniraj 30](#_Toc432710643)

[Ugovor UG7: KreirajNoviProizvod 30](#_Toc432710644)

[Ugovor UG8: ZapamtiProizvod 30](#_Toc432710645)

[Ugovor UG9: PretraziProizvode 30](#_Toc432710646)

[Ugovor UG10: NadjiProizvod 30](#_Toc432710647)

[Ugovor UG11: KreirajNovogPoslovnogPartnera 30](#_Toc432710648)

[Ugovor UG12: ZapamtiPoslovnogPartnera 31](#_Toc432710649)

[Ugovor UG13: PretraziPoslovnePartnere 31](#_Toc432710650)

[Ugovor UG14: NadjiPoslovnogPartnera 31](#_Toc432710651)

[Ugovor UG15: NadjiKorisnika 31](#_Toc432710652)

[Konceptualni model 32](#_Toc432710653)

[Relacioni model 32](#_Toc432710654)

[Projektovanje 36](#_Toc432710655)

[Arhitektura softverskog sistema 36](#_Toc432710656)

[Projektovanje korisničkog interfejsa 37](#_Toc432710657)

[MVC (Model – View - Controller) patern 37](#_Toc432710658)

[Projektovanje ekranskih formi 38](#_Toc432710659)

[Projektovanje kontrolera korisničkog interfejsa 91](#_Toc432710660)

[Projektovanje aplikacione logike 95](#_Toc432710661)

[Generics mehanizam 95](#_Toc432710662)

[Kontroler aplikacione logike 95](#_Toc432710663)

[Poslovna logika 97](#_Toc432710664)

[Broker baze podataka 104](#_Toc432710665)

[Projektovanje skladišta podataka 109](#_Toc432710666)

[Principi, metode i strategije projektovanja softvera 111](#_Toc432710667)

[Principi (tehnike) projektovanja softvera 111](#_Toc432710668)

[Apstrakcija 111](#_Toc432710669)

[Spojenost i kohezija 115](#_Toc432710670)

[*Dekompozicija i modularizacija* 117](#_Toc432710671)

[Učaurenje / Sakrivanje informacija 119](#_Toc432710672)

[*Odvajanje interfejsa i implementacije* 120](#_Toc432710673)

[Dovoljnost, kompletnost i jednostavnost 120](#_Toc432710674)

[Strategije projektovanja softvera 121](#_Toc432710675)

[Metode projektovanja softvera 122](#_Toc432710676)

[Objektno orijentisano projektovanje 123](#_Toc432710677)

[Principi objektno orijentisanog projektovanja klasa 123](#_Toc432710678)

# Prikupljanje zahteva

## Verbalni opis

Potrebno je napraviti aplikaciju koja će omogućiti vođenje evidencije o izlaznim fakturama preduzeća, unos nove fakture, izmenu postojeće, storniranje fakture i pretragu unetih faktura. Za svaku fakturu treba uneti poslovnog partnera, datum prodaje i proizvode koji su predmet poslovanja. Takođe treba voditi evidenciju o vrednosti PDV – a za svaku fakturu.

U aplikaciji treba voditi evidenciju o proizvodima koje preduzeće proizvodi i prodaje, njihovo dodavanje i izmenu. Svaki proizvod ima svoj jedinstveni identifikator, naziv, cenu i jedinicu mere.

Zatim treba voditi evidenciju o poslovnim partnerima sa kojima preduzeće posluje, njihovo dodavanje i izmenu. O poslovnim partnerima treba čuvati podatke: naziv, PIB, matični broj preduzeća, adresu i kontakt.

Da bi korisnik koristio aplikaciju mora da bude prijavljen na sistem.

## Slučajevi korišćenja

Identifikovani su sledeći slučajevi korišćenja:

1. Unos nove fakture
2. Izmena fakture
3. Pretraga faktura
4. Storniranje fakura
5. Unos novog proizvoda
6. Izmena proizvoda
7. Unos poslovnog partnera
8. Izmena poslovnog partnera
9. Prijavljivanje korisnika



Slika : Dijagram slučajeva korišćenja

### Slučaj korišćenja 1: Unos nove fakture

##### **Naziv SK**

Unos nove fakture

##### **Aktori SK**

Korisnik

##### **Učesnici SK**

Korisnik i sistem (program)

##### **Preduslov**: Sistem je uključen i korisnik je prijavljen pod svojom šifrom. Sistem prikazuje početnu formu.

##### **Osnovni scenario SK**

1. Korisnik **poziva** sistem da kreira novu fakturu. (APSO)
2. Sistem **kreira** novu fakturu. (SO)
3. Sistem **prikazuje** korisniku novu fakturu i poruku: “Sistem je kreirao novu fakturu”. (IA)
4. Korisnik **unosi** podatke u novu fakturu. (APUSO)
5. Korisnik **kontroliše** da li je korektno uneo podatke u novu fakturu. (ANSO)
6. Korisnik **poziva** sistem da zapamti podatke o fakturi. (APSO)
7. Sistem **pamti** podatke o fakturi. (SO)
8. Sistem **prikazuje** korisniku zapamćenu fakturu i poruku: “Sistem je sačuvao fakturu”. (IA)
9. Korisnik **poziva** sistem da obradi fakturu. (APSO)
10. Sistem **obrađuje** fakturu. (SO)
11. Sistem **prikazuje** korisniku obrađenu fakturu i poruku: “Sistem je obradio fakturu”. (IA)

##### **Alternativna scenarija**

3.1 Ukoliko sistem ne može da kreira fakturu on prikazuje korisniku poruku: “Sistem ne može da kreira novu fakturu”. Prekida se izvršenje scenaria. (IA)

8.1 Ukoliko sistem ne može da zapamti podatke o fakturi on prikazuje korisniku poruku “Sistem ne može da zapamti fakturu”. Prekida se izvršenje scenaria. (IA)

11.1 Ukoliko sistem ne može da obradi fakturu on prikazuje korisniku poruku “Sistem ne može da obradi fakturu”. (IA)

### Slučaj korišćenja 2: Izmena fakture

##### **Naziv SK**

Izmena fakture

##### **Aktori SK**

Korisnik

##### **Učesnici SK**

Korisnik i sistem (program)

##### **Preduslov**: Sistem je uključen i korisnik je prijavljen pod svojom šifrom. Sistem prikazuje formu za pretragu faktura.

##### **Osnovni scenario SK**

1. Korisnik **unosi** vrednost po kojoj pretražuje fakture. (APUSO)
2. Korisnik **poziva** sistem da nađe fakture po zadatoj vrednosti. (APSO)
3. Sistem **traži** fakturu po zadatoj vrednosti. (SO)
4. Sistem **prikazuje** korisniku pronađene fakture. (IA)
5. Korisnik **bira** fakturu. (APUSO)
6. Korisnik **poziva** sistem da nađe sve podatke o izabranoj fakturi. (APSO)
7. Sistem **traži** podatke o izabranoj fakturi.(SO)
8. Sistem **prikazuje** korisniku sve podatke o fakturi. (IA)
9. Korisnik **unosi (menja)** podatke o fakturi. (APUSO)
10. Korisnik **kontroliše** da li je korektno uneo podatke o fakturi. (ANSO)
11. Korisnik **poziva** sistem da zapamti podatke o fakturi. (APSO)
12. Sistem **pamti** podatke o fakturi. (SO)
13. Sistem **prikazuje** korisniku zapamćenu fakturu i poruku: “Sistem je zapamtio fakturu.” (IA)
14. Korisnik **poziva** sistem da obradi fakturu. (APSO)
15. Sistem **obrađuje** fakturu. (SO)
16. Sistem **prikazuje** prodavcu obrađenu fakturu i poruku: “Sistem je obradio fakturu.” (IA)

##### **Alternativna scenarija**

4.1 Ukoliko sistem ne može da nađe nijednu fakturu on prikazuje korisniku poruku: “Sistem ne može da nađe fakturu po zadatoj vrednosti”. (IA)

8.1 Ukoliko sistem ne može da nađe podatke o fakturi on prikazuje korisniku poruku: “Sistem ne može da nađe fakturu po zadatoj vrednosti”. (IA)

13.1 Ukoliko sistem ne može da zapamti podatke o izmenjenoj fakturi on prikazuje korisniku poruku “Sistem ne može da zapamti novu fakturu”. Prekida se izvršenje scenaria. (IA)

16.1 Ukoliko sistem ne može da obradi fakturu on prikazuje korisniku poruku: “Sistem ne može da obradi fakturu.” (IA)

### Slučaj korišćenja 3: Storniranje fakture

##### **Naziv SK**

Storniranje fakture

##### **Aktori SK**

Korisnik

##### **Učesnici SK**

Korisnik i sistem (program)

##### **Preduslov**: Sistem je uključen i korisnik je prijavljen pod svojom šifrom. Sistem prikazuje formu za pretragu faktura.

##### **Osnovni scenario SK**

1. Korisnik **unosi** vrednost po kojoj pretražuje fakture. (APUSO)
2. Korisnik **poziva** sistem da nađe fakture po zadatoj vrednosti. (APSO)
3. Sistem **traži** fakturu po zadatoj vrednosti. (SO)
4. Sistem **prikazuje** korisniku pronađene fakture. (IA)
5. Korisnik **bira** fakturu. (APUSO)
6. Korisnik **poziva** sistem da nađe sve podatke o izabranoj fakturi. (APSO)
7. Sistem **traži** podatke o izabranoj fakturi.(SO)
8. Sistem **prikazuje** korisniku sve podatke o fakturi. (IA)
9. Korisnik **poziva** sistem da stornira zadatu fakturu. (APSO)
10. Sistem **stornira** fakturu. (SO)
11. Sistem **prikazuje** korisniku storniranu fakturu i poruku: “Sistem je stornirao fakturu.” (IA)

##### **Alternativna scenarija**

4.1 Ukoliko sistem ne može da nađe nijednu fakturu on prikazuje korisniku poruku: “Sistem ne može da nađe fakturu po zadatoj vrednosti”. (IA)

8.1 Ukoliko sistem ne može da nađe podatke o fakturi on prikazuje korisniku poruku: “Sistem ne može da nađe fakturu po zadatoj vrednosti”. (IA)

11.1 Ukoliko sistem ne može da stornira fakturu on prikazuje korisniku poruku: “Sistem ne može da stornira fakturu”. (IA)

### Slučaj korišćenja 4: Pretraga faktura

##### **Naziv SK**

Pretraga faktura

##### **Aktori SK**

Korisnik

##### **Učesnici SK**

Korisnik i sistem (program)

##### **Preduslov**: Sistem je uključen i korisnik je prijavljen pod svojom šifrom. Sistem prikazuje formu za pretragu faktura.

##### **Osnovni scenario SK**

1. Korisnik **unosi** vrednost po kojoj pretražuje fakture. (APUSO)
2. Korisnik **poziva** sistem da nađe fakturu po zadatoj vrednosti. (APSO)
3. Sistem **traži** fakturu po zadatoj vrednosti. (SO)
4. Sistem **prikazuje** korisniku pronađene fakture. (IA)
5. Korisnik **bira** fakturu. (APUSO)
6. Korisnik **poziva** sistem da nađe sve podatke o izabranoj fakturi. (APSO)
7. Sistem **traži** podatke o izabranoj fakturi. (SO)
8. Sistem **prikazuje** korisniku sve podatke o fakturi. (IA)

##### **Alternativna scenarija**

4.1 Ukoliko sistem ne može da nađe nijednu fakturu on prikazuje korisniku poruku: “Sistem ne može da nađe fakturu po zadatoj vrednosti”. (IA)

8.1 Ukoliko sistem ne može da nađe podatke o fakturi on prikazuje korisniku poruku: “Sistem ne može da nađe fakturu po zadatoj vrednosti”. (IA)

### Slučaj korišćenja 5: Unos novog proizvoda

##### **Naziv SK**

Unos novog proizvoda

##### **Aktori SK**

Korisnik

##### **Učesnici SK**

Korisnik i sistem (program)

##### **Preduslov**: Sistem je uključen i korisnik je prijavljen pod svojom šifrom. Sistem prikazuje početnu formu.

##### **Osnovni scenario SK**

1. Korisnik **poziva** sistem da kreira novu proizvod. (APSO)
2. Sistem **kreira** novi proizvod. (SO)
3. Sistem **prikazuje** korisniku novi proizvod i poruku: “Sistem je kreirao novi proizvod”. (IA)
4. Korisnik **unosi** podatke o novom proizvodu. (APUSO)
5. Korisnik **kontroliše** da li je korektno uneo podatke o proizvodu. (ANSO)
6. Korisnik **poziva** sistem da zapamti podatke o proizvodu. (APSO)
7. Sistem **pamti** podatke o proizvodu. (SO)
8. Sistem **prikazuje** korisniku zapamćeni proizvod i poruku: “Sistem je sačuvao proizvod”. (IA)

##### **Alternativna scenarija**

3.1 Ukoliko sistem ne može da kreira proizvod on prikazuje korisniku poruku: “Sistem ne može da kreira novi proizvod”. Prekida se izvršenje scenaria. (IA)

8.1 Ukoliko sistem ne može da zapamti podatke o proizvodu on prikazuje korisniku poruku “Sistem ne može da zapamti proizvod”. Prekida se izvršenje scenaria. (IA)

### Slučaj korišćenja 6: Izmena proizvoda

##### **Naziv SK**

Izmena proizvoda

##### **Aktori SK**

Korisnik

##### **Učesnici SK**

Korisnik i sistem (program)

##### **Preduslov**: Sistem je uključen i korisnik je prijavljen pod svojom šifrom. Sistem prikazuje formu za pretragu proizvoda.

##### **Osnovni scenario SK**

1. Korisnik **unosi** vrednost po kojoj pretražuje proizvode. (APUSO)
2. Korisnik **poziva** sistem da nađe proizvode po zadatoj vrednosti. (APSO)
3. Sistem **traži** proizvod po zadatoj vrednosti. (SO)
4. Sistem **prikazuje** korisniku pronađene proizvode. (IA)
5. Korisnik **bira** proizvod. (APUSO)
6. Korisnik **poziva** sistem da nađe sve podatke o izabranom proizvodu. (APSO)
7. Sistem **traži** podatke o izabranom proizvodu. (SO)
8. Sistem **prikazuje** korisniku sve podatke o proizvodu. (IA)
9. Korisnik **unosi (menja)** podatke o proizvodu. (APUSO)
10. Korisnik **kontroliše** da li je korektno uneo podatke o proizvodu. (ANSO)
11. Korisnik **poziva** sistem da zapamti podatke o proizvodu. (APSO)
12. Sistem **pamti** podatke o proizvodu. (SO)
13. Sistem **prikazuje** korisniku zapamćen proizvod i poruku: “Sistem je zapamtio proizvod.” (IA)

##### **Alternativna scenarija**

4.1 Ukoliko sistem ne može da nađe nijedan proizvod on prikazuje korisniku poruku: “Sistem ne može da nađe proizvod po zadatoj vrednosti”. (IA)

8.1 Ukoliko sistem ne može da nađe podatke o proizvodu on prikazuje korisniku poruku: “Sistem ne može da nađe proizvod po zadatoj vrednosti”. (IA)

13.1 Ukoliko sistem ne može da zapamti podatke o izmenjenom proizvodu on prikazuje korisniku poruku “Sistem ne može da zapamti izmenjeni proizvod”. Prekida se izvršenje scenaria. (IA)

### Slučaj korišćenja 7: Unos poslovnog partnera

##### **Naziv SK**

Unos novog poslovnog partnera

##### **Aktori SK**

Korisnik

##### **Učesnici SK**

Korisnik i sistem (program)

##### **Preduslov**: Sistem je uključen i korisnik je prijavljen pod svojom šifrom. Sistem prikazuje početnu formu.

##### **Osnovni scenario SK**

1. Korisnik **poziva** sistem da kreira novog poslovnog partnera. (APSO)
2. Sistem **kreira** novog poslovnog partnera. (SO)
3. Sistem **prikazuje** korisniku novog poslovnog partnera i poruku: “Sistem je kreirao novog poslovnog partnera”. (IA)
4. Korisnik **unosi** podatke o novom poslovnom partneru. (APUSO)
5. Korisnik **kontroliše** da li je korektno uneo podatke o poslovnom partneru. (ANSO)
6. Korisnik **poziva** sistem da zapamti podatke o poslovnom partneru. (APSO)
7. Sistem **pamti** podatke o poslovnom partneru. (SO)
8. Sistem **prikazuje** korisniku zapamćenog poslovnog partnera i poruku: “Sistem je sačuvao poslovnog partnera”. (IA)

##### **Alternativna scenarija**

3.1 Ukoliko sistem ne može da kreira poslovnog partnera on prikazuje korisniku poruku: “Sistem ne može da kreira novog poslovnog partnera”. Prekida se izvršenje scenaria. (IA)

8.1 Ukoliko sistem ne može da zapamti podatke o poslovnom partneru on prikazuje korisniku poruku “Sistem ne može da zapamti poslovnog partnera”. Prekida se izvršenje scenaria. (IA)

### Slučaj korišćenja 8: Izmena poslovnog partnera

##### **Naziv SK**

Izmena poslovnog partnera

##### **Aktori SK**

Korisnik

##### **Učesnici SK**

Korisnik i sistem (program)

##### **Preduslov**: Sistem je uključen i korisnik je prijavljen pod svojom šifrom. Sistem prikazuje formu za pretragu poslovnih partnera.

##### **Osnovni scenario SK**

1. Korisnik **unosi** vrednost po kojoj pretražuje poslovne partnere. (APUSO)
2. Korisnik **poziva** sistem da nađe poslovne partnere po zadatoj vrednosti. (APSO)
3. Sistem **traži** poslovne partnere po zadatoj vrednosti. (SO)
4. Sistem **prikazuje** korisniku pronađene poslovne partnere. (IA)
5. Korisnik **bira** poslovnog partnera. (APUSO)
6. Korisnik **poziva** sistem da nađe sve podatke o poslovnom partneru. (APSO)
7. Sistem **traži** podatke o izabranom poslovnom partneru. (SO)
8. Sistem **prikazuje** korisniku sve podatke o poslovnom partneru. (IA)
9. Korisnik **unosi (menja)** podatke o poslovnom partneru. (APUSO)
10. Korisnik **kontroliše** da li je korektno uneo podatke o poslovnom partneru. (ANSO)
11. Korisnik **poziva** sistem da zapamti podatke o poslovnom partneru. (APSO)
12. Sistem **pamti** podatke o poslovnom partneru. (SO)
13. Sistem **prikazuje** korisniku zapamćenog poslovnog partnera i poruku: “Sistem je zapamtio poslovnog partnera.” (IA)

##### **Alternativna scenarija**

4.1 Ukoliko sistem ne može da nađe nijedanog poslovnog partnera on prikazuje korisniku poruku: “Sistem ne može da nađe poslovnog partnera po zadatoj vrednosti”. (IA)

8.1 Ukoliko sistem ne može da nađe podatke o poslovnom partneru on prikazuje korisniku poruku: “Sistem ne može da nađe poslovnog partnera po zadatoj vrednosti”. (IA)

13.1 Ukoliko sistem ne može da zapamti podatke o izmenjenom poslovnom partneru on prikazuje korisniku poruku “Sistem ne može da zapamti izmenjenog poslovnog partnera ”. Prekida se izvršenje scenaria. (IA)

### Slučaj korišćenja 9: Prijavljivanje korisnika

##### **Naziv SK:**

Prijavljivanje korisnika

##### **Aktori SK:**

Korisnik

##### **Učesnici SK:**

Korisnik i sistem (program)

##### **Preduslov:** Sistem je uključen i prikazuje formu za prijavu.

##### **Osnovni scenario SK:**

1. Korisnik **unosi** podatke za autentifikaciju korisnika. (APUSO)
2. Korisnik **poziva** sistem da pronađe korisnika sa zadatim podacima. (APSO)
3. Sistem **pretražuje** korisnike. (SO)
4. Sistem **prikazuje** poruku da je korisnik uspešno prijavljen i početnu formu . (IA)

##### **Alternativna scenarija:**

4.1 Ukoliko sistem ne može da nađe korisnika prikazuje poruku da korisnik nije uneo tačne podatke. (IA)

# Analiza

## Dijagrami sekvenci

### DS1: Dijagram sekvenci slučaja korišćenja –Unos fakture

1. Korisnik **poziva** sistem da kreira novu fakturu. (APSO)
2. Sistem **prikazuje** korisniku novu fakturu i poruku: “Sistem je kreirao novu fakturu”. (IA)
3. Korisnik **poziva** sistem da zapamti podatke o fakturi. (APSO)
4. Sistem **prikazuje** korisniku zapamćenu fakturu i poruku: “Sistem je sačuvao fakturu”. (IA)
5. Korisnik **poziva** sistem da obradi fakturu. (APSO)
6. Sistem **prikazuje** korisniku obrađenu fakturu i poruku: “Sistem je obradio fakturu”. (IA)

##### 

Slika - Dijagram sekvenci SK Unos fakture, osnovni scenario

##### **Alternativna scenarija**

2.1 Ukoliko sistem ne može da kreira fakturu on prikazuje korisniku poruku: “Sistem ne može da kreira novu fakturu”. Prekida se izvršenje scenarija. (IA)



Slika - - Dijagram sekvenci SK Unos fakture, alternativni scenario

4.1 Ukoliko sistem ne može da zapamti podatke o fakturi on prikazuje korisniku poruku “Sistem ne može da zapamti fakturu”. Prekida se izvršenje scenarija. (IA)



Slika - Dijagram sekvenci SK Unos fakture, alternativni scenario

6.1 Ukoliko sistem ne može da obradi fakturu on prikazuje korisniku poruku “Sistem ne može da obradi fakturu”. Prekida se izvršenje scenarija. (IA)



Slika - Dijagram sekvenci SK Unos fakture, alternativni scenario

Sa navedenih sekvencnih dijagrama uočavaju se 3 sistemske operacije koje treba projektovati:

1. signal **KreirajNovu**(Faktura);
2. signal **Zapamti**(Faktura);
3. signal **Obradi**(Faktura);

### DS2: Dijagram sekvenci slučaja korišćenja – Izmena fakture

1. Korisnik **poziva** sistem da nađe fakture po zadatoj vrednosti. (APSO)
2. Sistem **prikazuje** korisniku pronađene fakture. (IA)
3. Korisnik **poziva** sistem da nađe sve podatke o izabranoj fakturi. (APSO)
4. Sistem **prikazuje** korisniku sve podatke o fakturi. (IA)
5. Korisnik **poziva** sistem da zapamti podatke o fakturi. (APSO)
6. Sistem **prikazuje** korisniku zapamćenu fakturu i poruku: “Sistem je zapamtio fakturu.” (IA)
7. Korisnik **poziva** sistem da obradi fakturu. (APSO)
8. Sistem **prikazuje** prodavcu obrađenu fakturu i poruku: “Sistem je obradio fakturu.” (IA)

##### 

Slika - Dijagram sekvenci SK Izmena fakture, osnovni scenario

##### **Alternativna scenarija**

2.1 Ukoliko sistem ne može da nađe nijednu fakturu on prikazuje korisniku poruku: “Sistem ne može da nađe fakturu po zadatoj vrednosti”. Prekida se izvršenje scenarija. (IA)



Slika - Dijagram sekvenci SK Izmena fakture, alternativni scenario

4.1 Ukoliko sistem ne može da nađe podatke o fakturi on prikazuje korisniku poruku: “Sistem ne može da nađe fakturu po zadatoj vrednosti”. Prekida se izvršenje scenarija. (IA)



Slika - Dijagram sekvenci SK Izmena fakture, alternativni scenario

6.1 Ukoliko sistem ne može da zapamti podatke o izmenjenoj fakturi on prikazuje korisniku poruku “Sistem ne može da zapamti novu fakturu”. Prekida se izvršenje scenarija. (IA)



Slika - Dijagram sekvenci SK Izmena fakture, alternativni scenario

8.1 Ukoliko sistem ne može da obradi fakturu on prikazuje korisniku poruku: “Sistem ne može da obradi fakturu”. Prekida se Izvršenje scenarija. (IA)



Slika - Dijagram sekvenci SK Izmena fakture, alternativni scenario

Sa navedenih sekvencnih dijagrama uočavaju se 4 sistemske operacije:

1. signal PretraziFakture(List<Faktura>, KriterijumPretrage);
2. signal NadjiFakturu(Faktura, KriterijumPretrage);
3. signal Zapamti(Faktura);
4. signal Obradi(Faktura);

### DS3: Dijagram sekvenci slučaja korišćenja – Storniranje fakture

1. Korisnik **poziva** sistem da nađe fakture po zadatoj vrednosti. (APSO)
2. Sistem **prikazuje** korisniku pronađene fakture. (IA)
3. Korisnik **poziva** sistem da nađe sve podatke o izabranoj fakturi. (APSO)
4. Sistem **prikazuje** korisniku sve podatke o fakturi. (IA)
5. Korisnik **poziva** sistem da stornira zadatu fakturu. (APSO)
6. Sistem **prikazuje** korisniku storniranu fakturu i poruku: “Sistem je stornirao fakturu.” (IA)

##### 

Slika - Dijagram sekvenci SK Storniranje fakture, osnovni scenario

##### **Alternativna scenarija**

2.1 Ukoliko sistem ne može da nađe nijednu fakturu on prikazuje korisniku poruku: “Sistem ne može da nađe fakturu po zadatoj vrednosti”. Prekida se izvršenje scenarija. (IA)



Slika - Dijagram sekvenci SK Storniranje fakture, alternativni scenario

4.1 Ukoliko sistem ne može da nađe podatke o fakturi on prikazuje korisniku poruku: “Sistem ne može da nađe fakturu po zadatoj vrednosti”. Prekida se izvršenje scenarija. (IA)



Slika - Dijagram sekvenci SK Storniranje fakture, alternativni scenario

6.1 Ukoliko sistem ne može da stornira fakturu on prikazuje korisniku poruku: “Sistem ne može da stornira fakturu”. Prekida se izvršenje scenarija. (IA)



Slika - Dijagram sekvenci SK Storniranje fakture, alternativni scenario

Sa navedenih sekvencnih dijagrama uočavaju se 3 sistemske operacije:

1. signal PretraziFakture(List<Fatura>, KriterijumPretrage);
2. signal NadjiFakturu(Faktura, KriterijumPretrage);
3. signal Storniraj(Faktura);

### DS4: Dijagram sekvenci slučaja korišćenja – Pretraga faktura

1. Korisnik **poziva** sistem da nađe fakturu po zadatoj vrednosti. (APSO)
2. Sistem **prikazuje** korisniku pronađene fakture. (IA)
3. Korisnik **poziva** sistem da nađe sve podatke o izabranoj fakturi. (APSO)
4. Sistem **prikazuje** korisniku sve podatke o fakturi. (IA)

##### 

Slika - Dijagram sekvenci SK Pretraga faktura, osnovni scenario

##### **Alternativna scenarija**

2.1 Ukoliko sistem ne može da nađe nijednu fakturu on prikazuje korisniku poruku: “Sistem ne može da nađe fakturu po zadatoj vrednosti”. Prekida se izvršenje scenarija. (IA)



Slika - Dijagram sekvenci SK Pretraga faktura, alternativni scenario

4.1 Ukoliko sistem ne može da nađe podatke o fakturi on prikazuje korisniku poruku: “Sistem ne može da nađe fakturu po zadatoj vrednosti”. Prekida se izvršenje scenarija. (IA)



Slika - Dijagram sekvenci SK Pretraga faktura, alternativni scenario

Sa navedenih sekvencnih dijagrama uočavaju se 2 sistemske operacije:

1. signal PretraziFakture(List<Faktura>, KriterijumPretrage);
2. signal NadjiFakturu(Faktura, KriterijumPretrage);

### DS5: Dijagram sekvenci slučaja korišćenja – Unos novog proizvoda

1. Korisnik **poziva** sistem da kreira novu proizvod. (APSO)
2. Sistem **prikazuje** korisniku novi proizvod i poruku: “Sistem je kreirao novi proizvod”. (IA)
3. Korisnik **poziva** sistem da zapamti podatke o proizvodu. (APSO)
4. Sistem **prikazuje** korisniku zapamćeni proizvod i poruku: “Sistem je sačuvao proizvod”. (IA)

##### 

Slika - Dijagram sekvenci SK Unos novog proizvoda, osnovni scenario

##### **Alternativna scenarija**

2.1 Ukoliko sistem ne može da kreira proizvod on prikazuje korisniku poruku: “Sistem ne može da kreira novi proizvod”. Prekida se izvršenje scenarija. (IA)



Slika - Dijagram sekvenci SK Unos novog proizvoda, alternativni scenario

4.1 Ukoliko sistem ne može da zapamti podatke o proizvodu on prikazuje korisniku poruku “Sistem ne može da zapamti proizvod”. Prekida se izvršenje scenarija. (IA)



Slika - Dijagram sekvenci SK Unos novog proizvoda, alternativni scenario

Sa navedenih sekvencnih dijagrama uočavaju se 2 sistemske operacije:

1. signal KreirajNoviProizvod(Proizvod);
2. signal ZapamtiProizvod(Proizvod);

### DS6: Dijagram sekvenci slučaja korišćenja – Izmena proizvoda

1. Korisnik **poziva** sistem da nađe proizvode po zadatoj vrednosti. (APSO)
2. Sistem **prikazuje** korisniku pronađene proizvode. (IA)
3. Korisnik **poziva** sistem da nađe sve podatke o izabranom proizvodu. (APSO)
4. Sistem **prikazuje** korisniku sve podatke o proizvodu. (IA)
5. Korisnik **poziva** sistem da zapamti podatke o proizvodu. (APSO)
6. Sistem **prikazuje** korisniku zapamćen proizvod i poruku: “Sistem je zapamtio proizvod.” (IA)

##### 

Slika - Dijagram sekvenci SK Izmena proizvoda, osnovni scenario

##### **Alternativna scenarija**

2.1 Ukoliko sistem ne može da nađe nijedan proizvod on prikazuje korisniku poruku: “Sistem ne može da nađe proizvod po zadatoj vrednosti”. (IA)



Slika - Dijagram sekvenci SK Izmena proizvoda, alternativni scenario

4.1 Ukoliko sistem ne može da nađe podatke o proizvodu on prikazuje korisniku poruku: “Sistem ne može da nađe proizvod po zadatoj vrednosti”. Prekida se izvršenje scenarija. (IA)



Slika - Dijagram sekvenci SK Izmena proizvoda, alternativni scenario

6.1 Ukoliko sistem ne može da zapamti podatke o izmenjenom proizvodu on prikazuje korisniku poruku “Sistem ne može da zapamti izmenjeni proizvod”. Prekida se izvršenje scenarija. (IA)



Slika - Dijagram sekvenci SK Izmena proizvoda, alternativni scenario

Sa navedenih sekvencnih dijagrama uočavaju se 3 sistemske operacije:

1. signal PretraziProizvode(List<Proizvod>, KriterijumPretrage);
2. signal NadjiProizvod(Proizvod, KriterijumPretrage);
3. signal ZapamtiProizvod(Proizvod);

### DS7: Dijagram sekvenci slučaja korišćenja – Unos poslovnog partnera

1. Korisnik **poziva** sistem da kreira novog poslovnog partnera. (APSO)
2. Sistem **prikazuje** korisniku novog poslovnog partnera i poruku: “Sistem je kreirao novog poslovnog partnera”. (IA)
3. Korisnik **poziva** sistem da zapamti podatke o poslovnom partneru. (APSO)
4. Sistem **prikazuje** korisniku zapamćenog poslovnog partnera i poruku: “Sistem je sačuvao poslovnog partnera”. (IA)

##### 

Slika - Dijagram sekvenci SK Unos poslovnog partnera, osnovni scenario

##### **Alternativna scenarija**

2.1 Ukoliko sistem ne može da kreira poslovnog partnera on prikazuje korisniku poruku: “Sistem ne može da kreira novog poslovnog partnera”. Prekida se izvršenje scenarija. (IA)



Slika - Dijagram sekvenci SK Unos poslovnog partnera, alternativni scenario

4.1 Ukoliko sistem ne može da zapamti podatke o poslovnom partneru on prikazuje korisniku poruku “Sistem ne može da zapamti poslovnog partnera”. Prekida se izvršenje scenarija. (IA)



Slika - Dijagram sekvenci SK Unos poslovnog partnera, alternativni scenario

Sa navedenih sekvencnih dijagrama uočavaju se 2 sistemske operacije:

1. signal KreirajNovogPoslovnogPartnera(PoslovniPartner);
2. signal ZapamtiPoslovnogPartnera(PoslovniPartner);

### DS8: Dijagram sekvenci slučaja korišćenja – Izmena poslovnog partnera

1. Korisnik **poziva** sistem da nađe poslovne partnere po zadatoj vrednosti. (APSO)
2. Sistem **prikazuje** korisniku pronađene poslovne partnere. (IA)
3. Korisnik **poziva** sistem da nađe sve podatke o poslovnom partneru. (APSO)
4. Sistem **prikazuje** korisniku sve podatke o poslovnom partneru. (IA)
5. Korisnik **poziva** sistem da zapamti podatke o poslovnom partneru. (APSO)
6. Sistem **prikazuje** korisniku zapamćenog poslovnog partnera i poruku: “Sistem je zapamtio poslovnog partnera.” (IA)

##### 

Slika - Dijagram sekvenci SK Izmena poslovnog partnera, osnovni scenario

##### **Alternativna scenarija**

2.1 Ukoliko sistem ne može da nađe nijedanog poslovnog partnera on prikazuje korisniku poruku: “Sistem ne može da nađe poslovnog partnera po zadatoj vrednosti”. Prekida se izvršenje scenarija. (IA)



Slika - Dijagram sekvenci SK Izmena poslovnog partnera, alternativni scenario

4.1 Ukoliko sistem ne može da nađe podatke o poslovnom partneru on prikazuje korisniku poruku: “Sistem ne može da nađe poslovnog partnera po zadatoj vrednosti”. Prekida se izvršenje scenarija. (IA)



Slika - Dijagram sekvenci SK Izmena poslovnog partnera, alternativni scenario

6.1 Ukoliko sistem ne može da zapamti podatke o izmenjenom poslovnom partneru on prikazuje korisniku poruku “Sistem ne može da zapamti izmenjenog poslovnog partnera ”. Prekida se izvršenje scenarija. (IA)



Slika - Dijagram sekvenci SK Izmena poslovnog partnera, alternativni scenario

Sa navedenih sekvencnih dijagrama uočavaju se 3 sistemske operacije:

1. signal PretraziPoslovnePartnere(List<PoslovniPartner>, KriterijumPretrage);
2. signal NadjiPoslovnogPartnera(PoslovniPartner, KriterijumPretrage);
3. signal ZapamtiPoslovnogPartnera(PoslovniPartner);

### DS9: Dijagram sekvenci slučaja korišćenja – Prijavljivanje korisnika

1. Korisnik **poziva** sistem da pronađe korisnika sa zadatim podacima. (APSO)
2. Sistem **prikazuje** poruku da je korisnik uspešno prijavljen i početnu formu . (IA)

##### 

Slika - Dijagram sekvenci SK Prijavljivanje korisnika, osnovni scenario

##### **Alternativna scenarija**

2.1 Ukoliko sistem ne može da nađe korisnika prikazuje poruku da korisnik nije uneo tačne podatke. (IA)



Slika - Dijagram sekvenci SK Prijavljivanje korisnika, alternativni scenario

Sa navedenih sekvencnih dijagrama uočava se 1 sistemska operacija:

1. signal NadjiKorisnika(Korisnik, KriterijumPretrage);

## Ugovori o sistemskim operacijama

### Ugovor UG1: KreirajNovu

**Operacija: KreirajNovu**(Faktura):signal;

**Veza sa SK**: SK1

**Preduslovi**:

**Postuslovi**: Napravljena je nova faktura.   
 Zadovoljena strukturna ograničenja nad tabelom Faktura.

### Ugovor UG2: Zapamti

**Operacija: Zapamti**(Faktura):signal;

**Veza sa SK**: SK1, SK2

**Preduslovi**: Ako je faktura obrađena ili stornirana ne može se izvršiti sistemska operacija.  
 Prosto vrednosno ograničenje nad *Faktura* objektom mora biti zadovoljeno.

**Postuslovi**: Izračunata je vrednost svake od stavki fakture.  
 Izračunata je ukupna vrednost fakture.   
 Zadovoljena strukturna ograničenja nad tabelom Faktura.

### Ugovor UG3: Obradi

**Operacija: Obradi**(Faktura):signal;

**Veza sa SK**: SK1, SK2

**Preduslovi**: Ako je faktura obrađena ili stornirana ne može se izvršiti sistemska operacija.  
 Prosto vrednosno ograničenja nad Faktura objektom mora biti zadovoljeno.

**Postuslovi**: Izračunata je vrednost svake od stavki fakture.  
 Izračunata je ukupna vrednost fakture.  
 Faktura je obrađena.  
 Zadovoljena strukturna ograničenj nad tabelom Faktura.

### Ugovor UG4: PretraziFakture

**Operacija: PretraziFakture**(List<Faktura>, kriterijumPretrage):signal;

**Veza sa SK**: SK2, SK3, SK4

**Preduslovi**:

**Postuslovi**:

### Ugovor UG5: NadjiFakturu

**Operacija: NadjiFakturu**(Faktura, kriterijumPretrage):signal;

**Veza sa SK**: SK2, SK3, SK4

**Preduslovi**:

**Postuslovi**:

### Ugovor UG6: Storniraj

**Operacija: Storniraj**(Faktura):signal;

**Veza sa SK**: SK3

**Preduslovi**: Ako je faktura stornirana ne može se izvršiti sistemska operacija.

**Postuslovi**: Faktura je stornirana.   
 Zadovoljena strukturna ograničenja nad tabelom Faktura.

### Ugovor UG7: KreirajNoviProizvod

**Operacija: KreirajNoviProizvod**(Proizvod):signal;

**Veza sa SK**: SK5

**Preduslovi**:

**Postuslovi**: Kreiran je novi proizvod.   
 Zadovoljena strukturna ograničenja nad tabelom Proizvod.

### Ugovor UG8: ZapamtiProizvod

**Operacija: ZapamtiProizvod**(Proizvod):signal;

**Veza sa SK**: SK5, SK6

**Preduslovi**: Prosto vrednosno ograničenje nad *Proizvod* objektom mora biti zadovoljeno.

**Postuslovi**: Zapamćen je novi proizvod.   
 Zadovoljena strukturna ograničenja nad tabelom Proizvod.

### Ugovor UG9: PretraziProizvode

**Operacija: PretraziProizvode**(List<Proizvod>, kriterijumPretrage):signal;

**Veza sa SK**: SK6

**Preduslovi**:

**Postuslovi**:

### Ugovor UG10: NadjiProizvod

**Operacija: NadjiProizvod**(Proizvod, kriterijumPretrage):signal;

**Veza sa SK**: SK6

**Preduslovi**:

**Postuslovi**:

### Ugovor UG11: KreirajNovogPoslovnogPartnera

**Operacija: KreirajNovogPoslovnogPartnera**(PoslovniPartner):signal;

**Veza sa SK**: SK7

**Preduslovi**:

**Postuslovi**: Kreiran je novi poslovni partner.   
 Zadovoljena strukturna ograničenja nad tabelom PoslovniPartner.

### Ugovor UG12: ZapamtiPoslovnogPartnera

**Operacija: ZapamtiPoslovnogPartnera**(PoslovniPartner):signal;

**Veza sa SK**: SK7, SK8

**Preduslovi**: Prosto vrednosno ograničenje nad *PoslovniPartner* objektom mora biti zadovoljeno.

**Postuslovi**: Zapamćen je novi poslovni partner.   
 Zadovoljena strukturna ograničenja nad tabelom PoslovniPartner.

### Ugovor UG13: PretraziPoslovnePartnere

**Operacija: PretraziPoslovnePartnere**(List<PoslovniPartner>, kriterijumPretrage):signal;

**Veza sa SK**: SK8

**Preduslovi**:

**Postuslovi**:

### Ugovor UG14: NadjiPoslovnogPartnera

**Operacija: NadjiPoslovnogPartnera**(PoslovniPartner, kriterijumPretrage):signal;

**Veza sa SK**: SK8

**Preduslovi**:

**Postuslovi**:

### Ugovor UG15: NadjiKorisnika

**Operacija: NadjiKorisnika**(Korisnik, kriterijumPretrage):signal;

**Veza sa SK**: SK9

**Preduslovi**:

**Postuslovi**:

## Konceptualni model

Struktura softverskog sistema prikazana je konceptualnim modelom na sledećoj slici.



Slika - Konceptualni model

## Relacioni model

Na osnovu konceptualnog modela dobijamo relacioni model:

**Korisnik**(korisnikID, ime, prezime, email, korisnickoIme, korisnickaSifra)

**Mesto**(ptt, nazivMesta)

**PoslovniPartner**(poslovniPartnerID, pib, naziv, ulica, broj, kontakt, *ptt*)

**Proizvod**(proizvodID, naziv, cena, jedMere)

**Faktura**(idFakture, brojFakture, brojOtpremnice, datum,osnovica, pdv, ukupno, obradjena, stornirana, *poslovniPartnerID*)

**StavkaFakture**(brojFakture, rb, kolicina, rabat, iznosRabata, poreskaOsnovica, pdv, iznosPDVa, vrednostSaPDVom, *proizvodID*)

Tabela - Ograničenja nad tabelom Korisnik

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Tabela Korisnik | | Prosto vrednosno ograničenje | | Složeno vrednosno ograničenje | | Strukturno ograničenje |
| Atributi | Ime | Tip atributa | Vrednost atributa | Međuzavisnost atributa jedne tabele | Međuzavisnost atributa više tabela | INSERT /  UPDATE/  DELETE/ |
| korisnikID | Int | not null |  |  |
| Ime | String | not null |  |  |
| Prezime | String | not null |  |  |
| Email | String | not null |  |  |
| korisnickoIme | String | not null |  |  |
| korisnickaSifra | String | not null |  |  |

Tabela - Ograničenja nad tabelom Mesto

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Tabela Mesto | | Prosto vrednosno ograničenje | | Složeno vrednosno ograničenje | | Strukturno ograničenje |
| Atributi | Ime | Tip atributa | Vrednost atributa | Međuzavisnost atributa jedne tabele | Međuzavisnost atributa više tabela | INSERT /  UPDATE CASCADES PoslovniPartner  DELETE RESTRICT PoslovniPartner |
| Ptt | Int | Not null |  |  |
| naziv | String | Not null |  |  |

Tabela - Ograničenja nad tabelom PoslovniPartner

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Tabela PoslovniPartner | | Prosto vrednosno ograničenje | | Složeno vrednosno ograničenje | | Strukturno ograničenje |
| Atributi | Ime | Tip atributa | Vrednost atributa | Međuzavisnost atributa jedne tabele | Međuzavisnost atributa više tabela | INSERT RESTRICTED Mesto  UPDATE CASCADES Faktura  RESTRICTED Mesto  DELETE RESTRICTED Faktura  RESTRICTED Mesto |
| poslovniPartnerID | Int | Not null |  |  |
| pib | String | Not null |  |  |
| naziv | String | Not null |  |  |
| ulica | String | Not null |  |  |
| broj | String | Not null |  |  |
| kontakt | String | Not null |  |  |
| ptt | int | Not null |  |  |

Tabela - Ograničenja nad tabelom Proizvod

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Tabela Proizvod | | Prosto vrednosno ograničenje | | Složeno vrednosno ograničenje | | Strukturno ograničenje |
| Atributi | Ime | Tip atributa | Vrednost atributa | Međuzavisnost atributa jedne tabele | Međuzavisnost atributa više tabela | INSERT /  UPDATE CASCADES StavkaFakture  DELETE  RESTRICTED StavkaFakture |
| proizvodID | Int | Not null |  |  |
| naziv | String | Not null |  |  |
| cena | Double | Not null and >0 |  |  |
| jedMere | String | Not null |  |  |

Tabela 5 - Ograničenja nad tabelom StavkaFakture

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Tabela StavkaFakture | | Prosto vrednosno ograničenje | | Složeno vrednosno ograničenje | | Strukturno ograničenje |
| Atributi | Ime | Tip atributa | Vrednost atributa | Međuzavisnost atributa jedne tabele | Međuzavisnost atributa više tabela | INSERT RESTRECTED Faktura RESTRICTED Proizvod  UPDATE RESTRECTED Faktura  DELETE  RESTRICTED Proizvod |
| brojFakture | String | Not null |  |  |
| rb | Int | Not null and >0 |  |  |
| kolicina | Double | >0 (default: 0) |  |  |
| rabat | Double | >0 (default: 7) |  |  |
| iznosRabata | Double | >0 (default: 0) |  | iznosRabata = kolicina \* proizvod.cena \*rabat |
| poreskaOsnovica | Double | >0 (default: 0) |  | poreskaOsnovica = kolicina \* proizvod.cena - iznosRabata |
| pdv | Double | >0, <1 (default: 0.2) |  |  |
| iznosPDVa |  |  | iznosPDVa = poreskaOsnovica \* pdv |  |
| vrednostSaPDVom | Double | >0 (default: 0) | vrednostSaPDVom = poreskaOsnovica + iznosPDVa |  |
| proizvodID | Int | Not null |  |  |

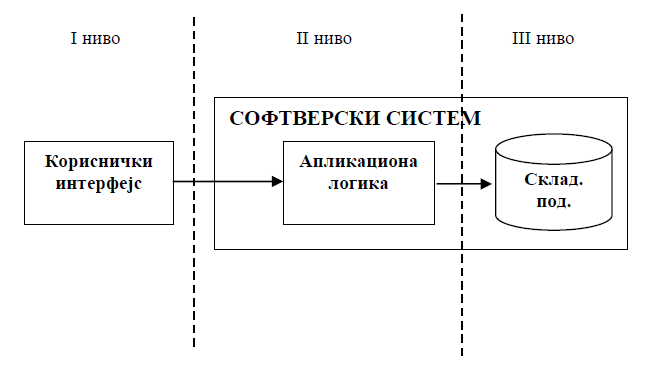
Tabela 6 - Ograničenja nad tabelom Faktura

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Tabela Faktura | | Prosto vrednosno ograničenje | | Složeno vrednosno ograničenje | | Strukturno ograničenje |
| Atributi | Ime | Tip atributa | Vrednost atributa | Međuzavisnost atributa jedne tabele | Međuzavisnost atributa više tabela | INSERT RESTRECTED PoslovniPartner  UPDATE CASCADES StavkaFakture  RESTRICTED PoslovniPartner  DELETE CASCADES StavkaFakture  RESTRICTED  PoslovniPartner |
| idFakture | Int | Not null |  |  |
| brojFakture | String | Not null |  |  |
| datum | Date | Not null |  |  |
| osnovica | Double | (default:0) |  | osnovica = SUM( stavkaFakture.  poreskaOsnovica) |
| pdv | Double | (default:0) |  | pdv = SUM( stavkaFakture.  iznosPDVa) |
| ukupno | Double | (default:0) | Ukupno = osnovica + pdv |  |
| obradjena | Boolean | (default:false) |  |  |
| stornirana | Boolean | (default:false) |  |  |
| poslovniPartnerID | int | Not null |  |  |

# Projektovanje

### Arhitektura softverskog sistema

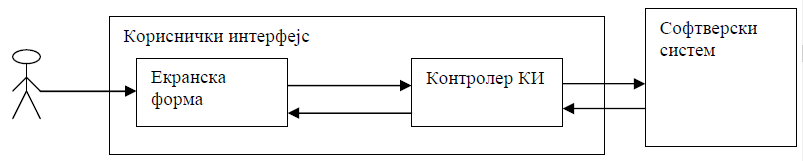
U ovom seminarskom radu korišćena je klasična tronivojska arhitektura, detaljnije prikazana na narednoj slici.



Slika - Tronivojska arhitektura

## Projektovanje korisničkog interfejsa

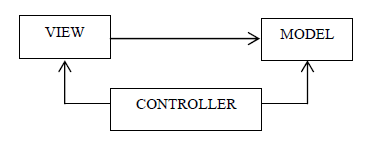
Na slici 36 je prikazana struktura korisničkog interfejsa.



Slika - struktura korisničkog interfejsa

### MVC (Model – View - Controller) patern

Ovde je implementiran MVC patern, koji je predstavljen na narednoj slici:



Slika – MVC patern

MVC je makrostrukturalni patern koji deli softverski sistem u tri dela:

* **View** obezbeđuje korisniku interfejs (ekransku formu) pomoću koga će korisnik da unosi podatke i poziva odgovarajuće operacije koje treba da se izvrše nad modelom; prikazuje stanje modela.
* **Controller** osluškuje i prihvata zahtev od klijenta za izvršenje operacije. Nakon toga poziva operaciju koja je definisana u modelu. Ukoliko model promeni stanje kontroler obaveštava view da je promenjeno stanje.
* **Model** predstavlja stanje sistema. Stanje mogu menjati neki od operacija modela.

U slučaju korišnjenja “Unos poslovnog partnera” **View** je FmUnosPoslovnogPartnera preko koga korisnik unosi podatke i poziva željenu operaciju nad modelom. View samo prikazuje stanje modela i ne sadrži logiku. KontorlerKIUnosPoslovnogPartnera je **Controller**, koji osluškuje ekransku formu i reaguje na korisnikovu akciju. U njemu se nalazi sva potrebna logika. Poslovni partner ovde predstavlja **Model**. Pomenuti kontroler je dat u sekciji *Projektovanje kontrolera korisničkog interfejsa*.

### Projektovanje ekranskih formi

#### Slučaj korišćenja 1: Unos nove fakture

##### **Naziv SK**

Unos nove fakture

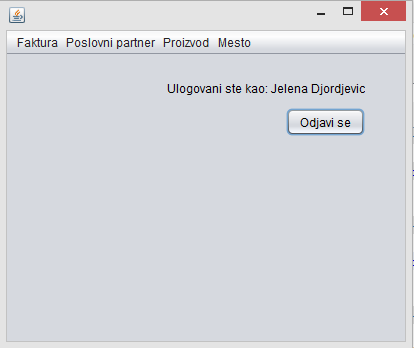
##### **Aktori SK**

Korisnik

##### **Učesnici SK**

Korisnik i sistem (program)

##### **Preduslov**: Sistem je uključen i korisnik je prijavljen pod svojom šifrom. Sistem prikazuje početnu formu.

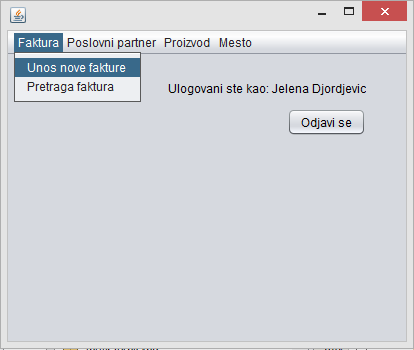


Slika - Početna forma

##### **Osnovni scenario SK**

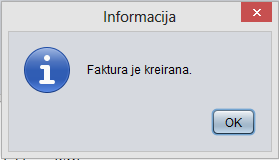
1. Korisnik **poziva** sistem da kreira novu fakturu. (APSO)

*Opis akcije*: Klikom na stavku menija Faktura, *Unos nove fakture,* korisnik poziva sistemsku operaciju ***kreiraj(Faktura)*** koja pravi novu fakturu.

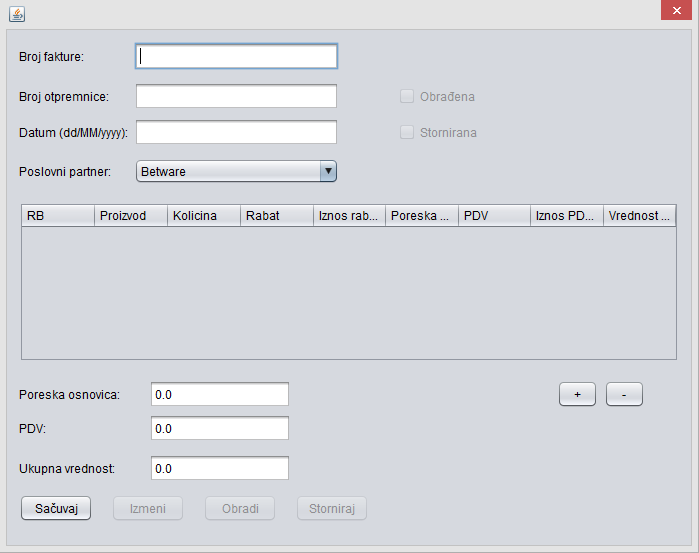


Slika - Kreiranje nove fakture

1. Sistem **kreira** novu fakturu. (SO)
2. Sistem **prikazuje** korisniku novu fakturu i poruku: “Sistem je kreirao novu fakturu”. (IA)

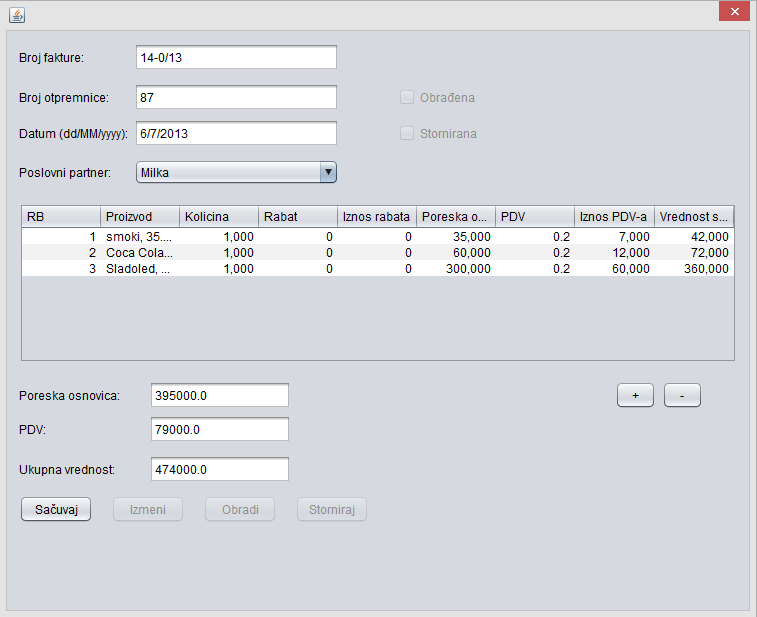


Slika - Poruka



Slika - Nova faktura

1. Korisnik **unosi** podatke u novu fakturu. (APUSO)

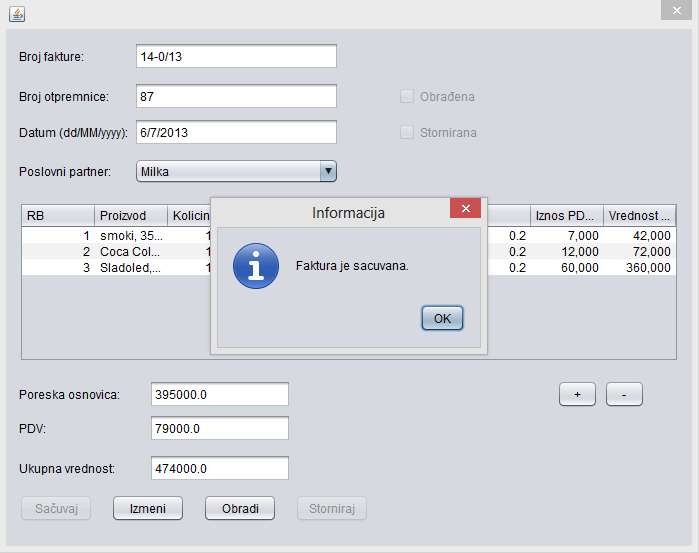


Slika - Unos podataka u novu fakturu

1. Korisnik **kontroliše** da li je korektno uneo podatke u novu fakturu. (ANSO)
2. Korisnik **poziva** sistem da zapamti podatke o fakturi. (APSO)

*Opis akcije*: Korisnik klikom na dugme „Sačuvaj“ poziva sistemsku operaciju ***Zapamti(Faktura)*** koja čuva novu fakturu.

1. Sistem **pamti** podatke o fakturi. (SO)
2. Sistem **prikazuje** korisniku zapamćenu fakturu i poruku: “Sistem je sačuvao fakturu”. (IA)

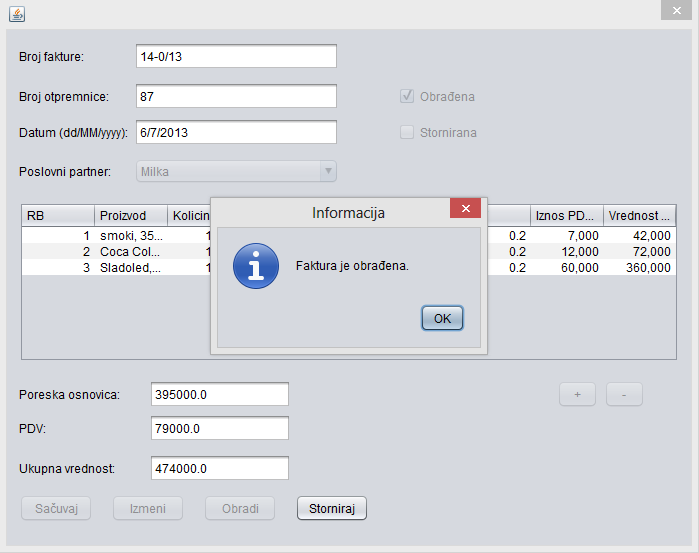


Slika - Sačuvana faktura

1. Korisnik **poziva** sistem da obradi fakturu. (APSO)

*Opis akcije*: Korisnik klikom na dugme „Obradi“ poziva sistemsku operaciju ***Obradi(Faktura)*** koja obrađuje novu fakturu.

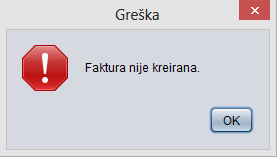
1. Sistem **obrađuje** fakturu. (SO)
2. Sistem **prikazuje** korisniku obrađenu fakturu i poruku: “Sistem je obradio fakturu”. (IA)



Slika - Obrađena faktura

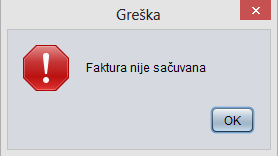
##### **Alternativna scenarija**

3.1 Ukoliko sistem ne može da kreira fakturu on prikazuje korisniku poruku: “Sistem ne može da kreira novu fakturu”. Prekida se izvršenje scenaria. (IA)



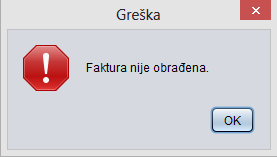
Slika - Greška

8.1 Ukoliko sistem ne može da zapamti podatke o fakturi on prikazuje korisniku poruku “Sistem ne može da zapamti fakturu”. Prekida se izvršenje scenaria. (IA)



Slika - Greška

11.1 Ukoliko sistem ne može da obradi fakturu on prikazuje korisniku poruku “Sistem ne može da obradi fakturu”. (IA)



Slika - Greška

#### Slučaj korišćenja 2: Izmena fakture

##### **Naziv SK**

Izmena fakture

##### **Aktori SK**

Korisnik

##### **Učesnici SK**

Korisnik i sistem (program)

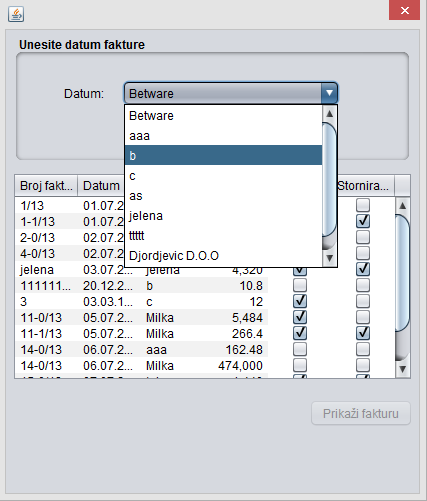
##### **Preduslov**: Sistem je uključen i korisnik je prijavljen pod svojom šifrom. Sistem prikazuje formu za pretragu faktura.



Slika - Forma za pretragu faktura

##### **Osnovni scenario SK**

1. Korisnik **unosi** vrednost po kojoj pretražuje fakture. (APUSO)

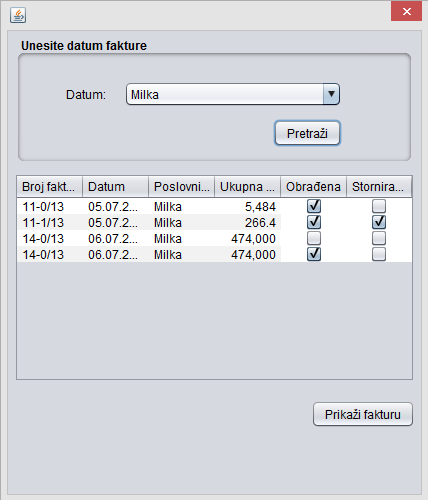


Slika - Korisnik bira vrednost za pretragu

1. Korisnik **poziva** sistem da nađe fakture po zadatoj vrednosti. (APSO)

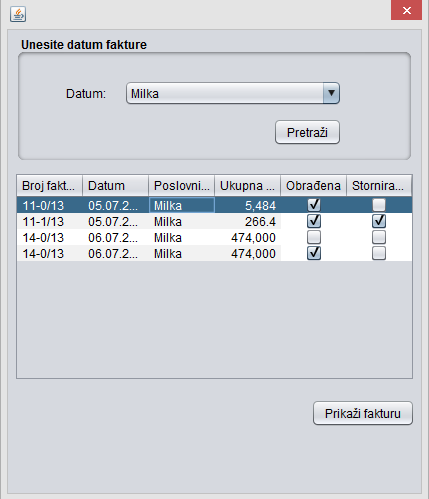
*Opis akcije:* Klikom na dugme “Pretraži” korisnik poziva sistemsku operaciju PretraziFakture(List<Faktura>, KriterijumPretrage) koja nalazi fakture po zadatoj vrednosti.

1. Sistem **traži** fakturu po zadatoj vrednosti. (SO)
2. Sistem **prikazuje** korisniku pronađene fakture. (IA)



Slika - Sistem prikazuje fakture

1. Korisnik **bira** fakturu. (APUSO)

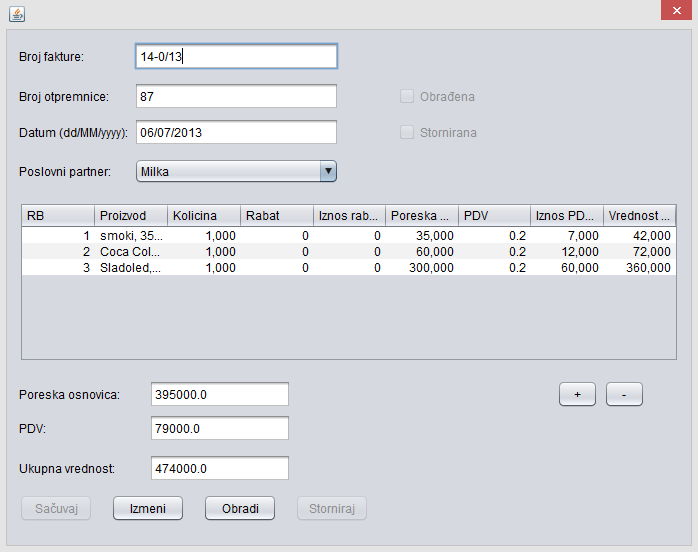


Slika - Korisnik je izabrao fakturu

1. Korisnik **poziva** sistem da nađe sve podatke o izabranoj fakturi. (APSO)

*Opis akcije*: Klikom na dugme “Prikaži fakturu” korisnik poziva sistemsku operaciju NadjiFakturu(Faktura, KriterijumPretrage) koja nalazi podatke za izabranu fakturu.

1. Sistem **traži** podatke o izabranoj fakturi.(SO)
2. Sistem **prikazuje** korisniku sve podatke o fakturi. (IA)

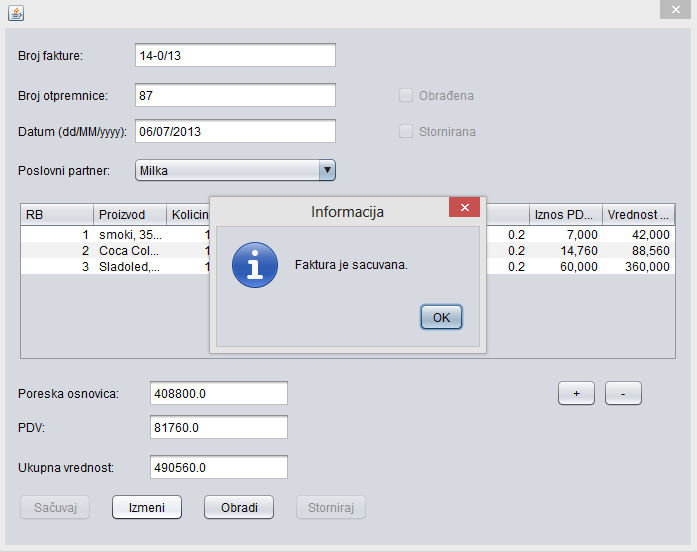


Slika - Prikaz fakture

1. Korisnik **unosi (menja)** podatke o fakturi. (APUSO)
2. Korisnik **kontroliše** da li je korektno uneo podatke o fakturi. (ANSO)
3. Korisnik **poziva** sistem da zapamti podatke o fakturi. (APSO)

*Opis akcije*: Korisnik klikom na dugme „Izmeni“ poziva sistemsku operaciju ***Zapamti(Faktura)*** koja čuva izmenjenu fakturu.

1. Sistem **pamti** podatke o fakturi. (SO)
2. Sistem **prikazuje** korisniku zapamćenu fakturu i poruku: “Sistem je zapamtio fakturu.” (IA)

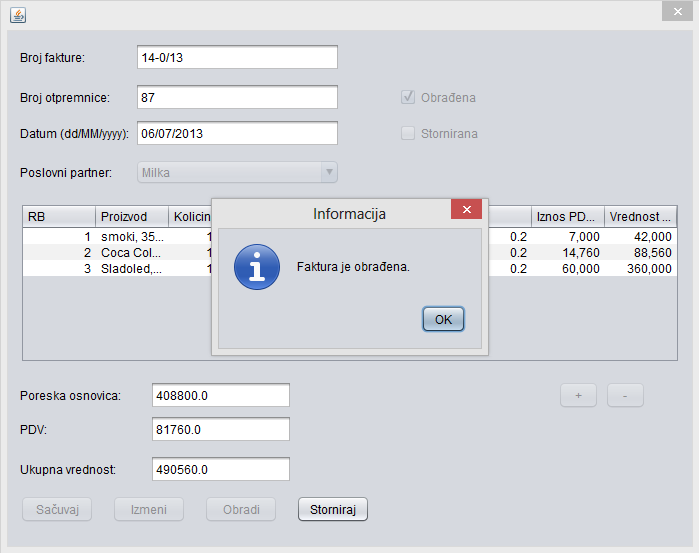


Slika - Faktura je sačuvana

1. Korisnik **poziva** sistem da obradi fakturu. (APSO)

*Opis akcije*: Korisnik klikom na dugme „Obradi“ poziva sistemsku operaciju ***Obradi(Faktura)*** koja obrađujefakturu.

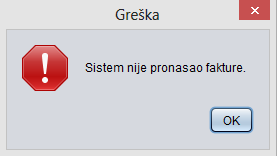
1. Sistem **obrađuje** fakturu. (SO)
2. Sistem **prikazuje** prodavcu obrađenu fakturu i poruku: “Sistem je obradio fakturu.” (IA)



Slika – Faktura je obrađena

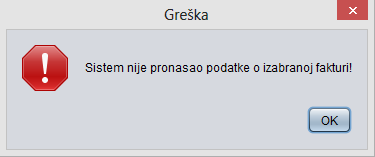
##### **Alternativna scenarija**

4.1 Ukoliko sistem ne može da nađe nijednu fakturu on prikazuje korisniku poruku: “Sistem ne može da nađe fakturu po zadatoj vrednosti”. (IA)



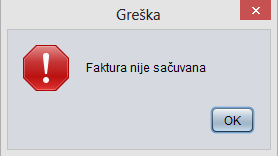
Slika - Greška

8.1 Ukoliko sistem ne može da nađe podatke o fakturi on prikazuje korisniku poruku: “Sistem ne može da nađe fakturu po zadatoj vrednosti”. (IA)



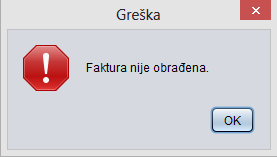
Slika - Greška

13.1 Ukoliko sistem ne može da zapamti podatke o izmenjenoj fakturi on prikazuje korisniku poruku “Sistem ne može da zapamti novu fakturu”. Prekida se izvršenje scenaria. (IA)



Slika - Greška

16.1 Ukoliko sistem ne može da obradi fakturu on prikazuje korisniku poruku: “Sistem ne može da obradi fakturu.” (IA)



Slika - Greška

#### Slučaj korišćenja 3: Storniranje fakture

##### **Naziv SK**

Storniranje fakture

##### **Aktori SK**

Korisnik

##### **Učesnici SK**

Korisnik i sistem (program)

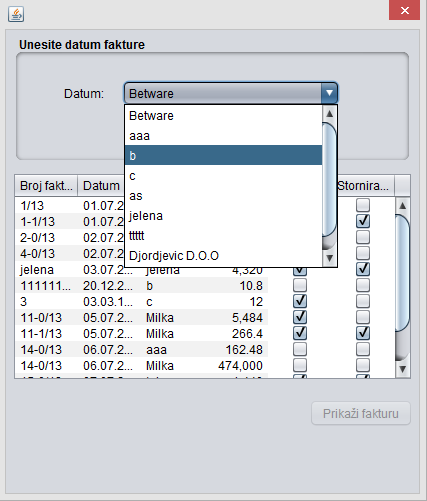
##### **Preduslov**: Sistem je uključen i korisnik je prijavljen pod svojom šifrom. Sistem prikazuje formu za pretragu faktura.



Slika - Forma za pretragu faktura

##### **Osnovni scenario SK**

1. Korisnik **unosi** vrednost po kojoj pretražuje fakture. (APUSO)

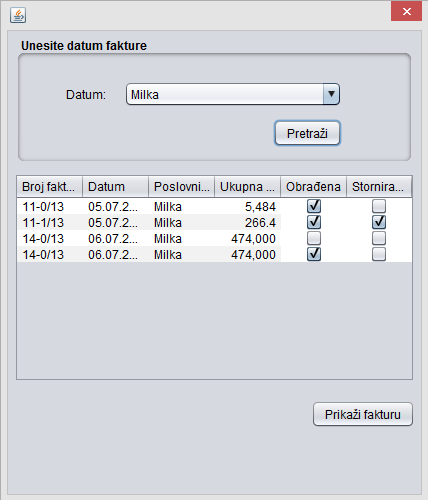


Slika - Korisnik bira vrednost za pretragu

1. Korisnik **poziva** sistem da nađe fakture po zadatoj vrednosti. (APSO)

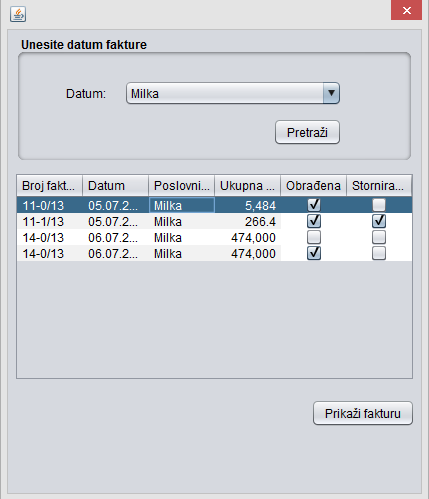
*Opis akcije:* Klikom na dugme “Pretraži” korisnik poziva sistemsku operaciju PretraziFakture(List<Faktura>, KriterijumPretrage) koja nalazi fakture po zadatoj vrednosti.

1. Sistem **traži** fakturu po zadatoj vrednosti. (SO)
2. Sistem **prikazuje** korisniku pronađene fakture. (IA)



Slika - Sistem prikazuje fakture

1. Korisnik **bira** fakturu. (APUSO)

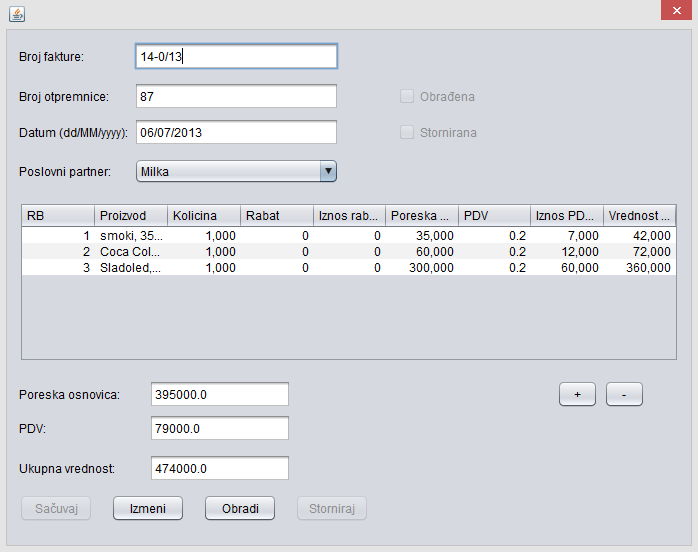


Slika - Korisnik je izabrao fakturu

1. Korisnik **poziva** sistem da nađe sve podatke o izabranoj fakturi. (APSO)

*Opis akcije*: Klikom na dugme “Prikaži fakturu” korisnik poziva sistemsku operaciju NadjiFakturu(Faktura, KriterijumPretrage) koja nalazi podatke za izabranu fakturu.

1. Sistem **traži** podatke o izabranoj fakturi.(SO)
2. Sistem **prikazuje** korisniku sve podatke o fakturi. (IA)

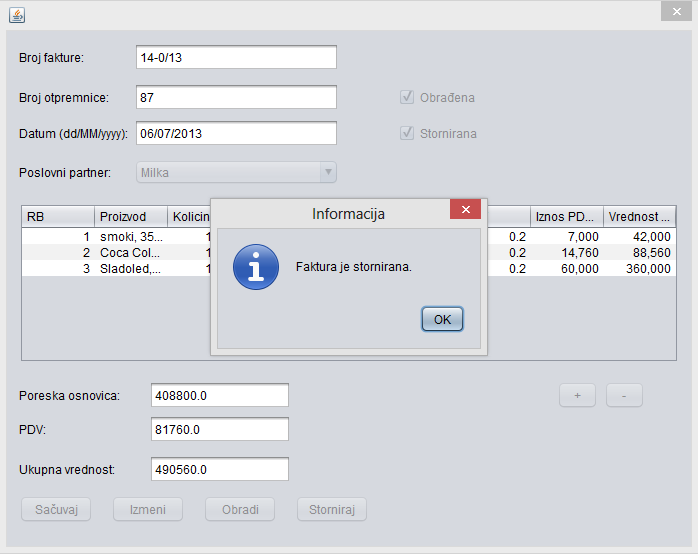


Slika - Prikaz fakture

1. Korisnik **poziva** sistem da stornira zadatu fakturu. (APSO)
2. Sistem **stornira** fakturu. (SO)

*Opis akcije*: Klikom na dugme “Storniraj” korisnik poziva sistemsku operaciju Storniraj(Faktura) koja stornira fakturu.

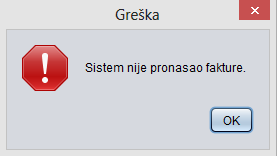
1. Sistem **prikazuje** korisniku storniranu fakturu i poruku: “Sistem je stornirao fakturu.” (IA)



Slika - Faktura je stornirana

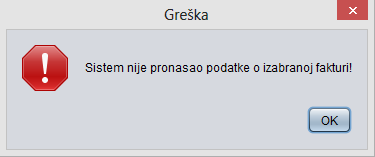
##### **Alternativna scenarija**

4.1 Ukoliko sistem ne može da nađe nijednu fakturu on prikazuje korisniku poruku: “Sistem ne može da nađe fakturu po zadatoj vrednosti”. (IA)



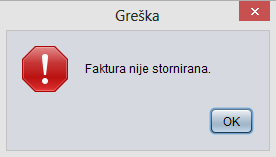
Slika - Greška

8.1 Ukoliko sistem ne može da nađe podatke o fakturi on prikazuje korisniku poruku: “Sistem ne može da nađe fakturu po zadatoj vrednosti”. (IA)



Slika - Greška

11.1 Ukoliko sistem ne može da stornira fakturu on prikazuje korisniku poruku: “Sistem ne može da stornira fakturu”. (IA)



Slika - Greška

#### Slučaj korišćenja 4: Pretraga faktura

##### **Naziv SK**

Pretraga faktura

##### **Aktori SK**

Korisnik

##### **Učesnici SK**

Korisnik i sistem (program)

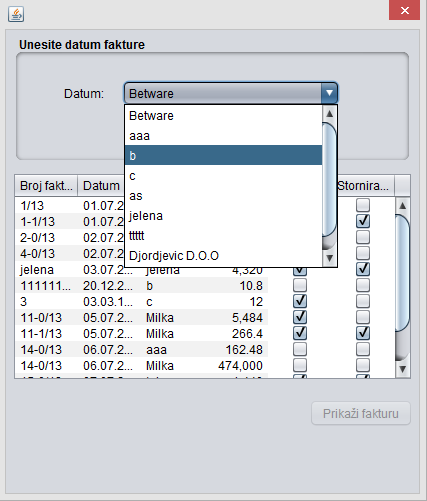
##### **Preduslov**: Sistem je uključen i korisnik je prijavljen pod svojom šifrom. Sistem prikazuje formu za pretragu faktura.



Slika - Forma za pretragu faktura

##### **Osnovni scenario SK**

1. Korisnik **unosi** vrednost po kojoj pretražuje fakture. (APUSO)

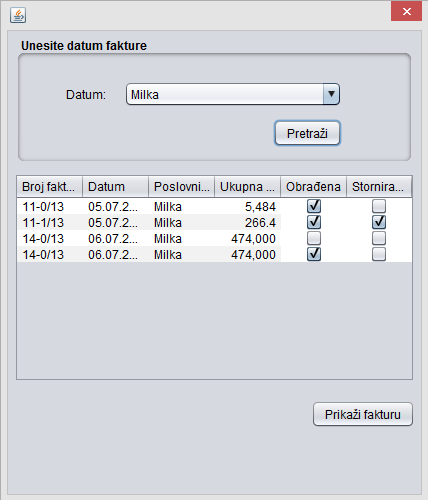


Slika - Korisnik bira vrednost za pretragu

1. Korisnik **poziva** sistem da nađe fakture po zadatoj vrednosti. (APSO)

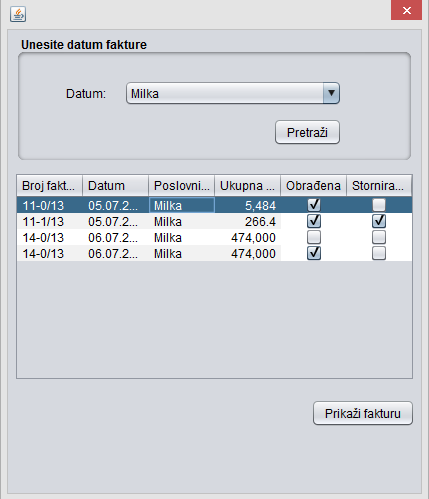
*Opis akcije:* Klikom na dugme “Pretraži” korisnik poziva sistemsku operaciju PretraziFakture(List<Faktura>, KriterijumPretrage) koja nalazi fakture po zadatoj vrednosti.

1. Sistem **traži** fakturu po zadatoj vrednosti. (SO)
2. Sistem **prikazuje** korisniku pronađene fakture. (IA)



Slika - Sistem prikazuje fakture

1. Korisnik **bira** fakturu. (APUSO)

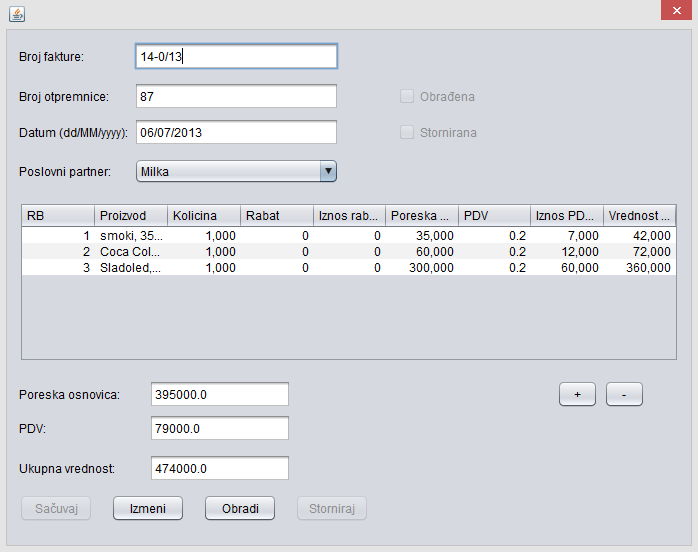


Slika - Korisnik je izabrao fakturu

1. Korisnik **poziva** sistem da nađe sve podatke o izabranoj fakturi. (APSO)

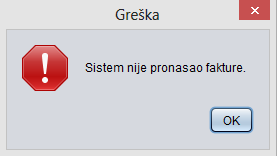
*Opis akcije*: Klikom na dugme “Prikaži fakturu” korisnik poziva sistemsku operaciju NadjiFakturu(Faktura, KriterijumPretrage) koja nalazi podatke za izabranu fakturu.

1. Sistem **traži** podatke o izabranoj fakturi.(SO)
2. Sistem **prikazuje** korisniku sve podatke o fakturi. (IA)



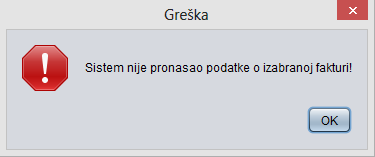
##### **Alternativna scenarija**

4.1 Ukoliko sistem ne može da nađe nijednu fakturu on prikazuje korisniku poruku: “Sistem ne može da nađe fakturu po zadatoj vrednosti”. (IA)



Slika - Greška

8.1 Ukoliko sistem ne može da nađe podatke o fakturi on prikazuje korisniku poruku: “Sistem ne može da nađe fakturu po zadatoj vrednosti”. (IA)



Slika - Greška

#### Slučaj korišćenja 5: Unos novog proizvoda

##### **Naziv SK**

Unos novog proizvoda

##### **Aktori SK**

Korisnik

##### **Učesnici SK**

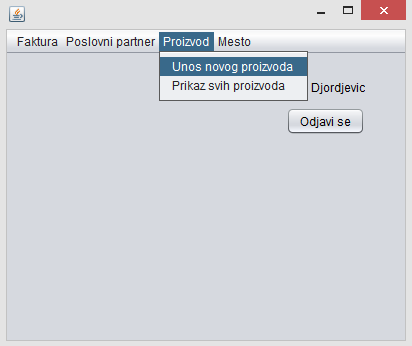
Korisnik i sistem (program)

##### **Preduslov**: Sistem je uključen i korisnik je prijavljen pod svojom šifrom. Sistem prikazuje početnu formu.

##### **Osnovni scenario SK**

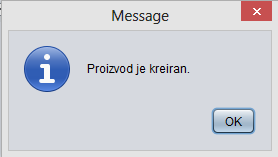
1. Korisnik **poziva** sistem da kreira novu proizvod. (APSO)

*Opis akcije*: Klikom na stavku menija Proizvod, *Unos novog proizvoda,* korisnik poziva sistemsku operaciju ***kreiraj(Proizvod)*** koja pravi novi proizvod.

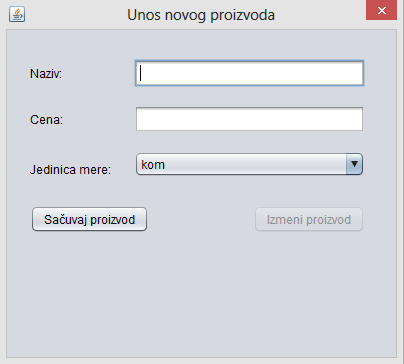


Slika - Kreiranje novog proizvoda

1. Sistem **kreira** novi proizvod. (SO)
2. Sistem **prikazuje** korisniku novi proizvod i poruku: “Sistem je kreirao novi proizvod”. (IA)



Slika – Poruka



Slika - Kreiran novi proizvod

1. Korisnik **unosi** podatke o novom proizvodu. (APUSO)

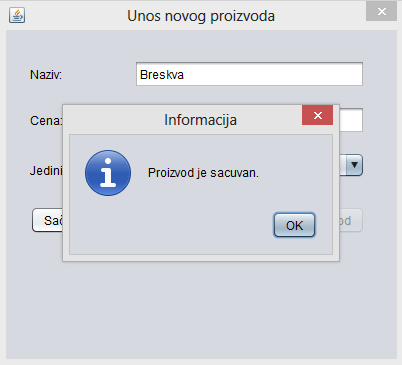


Slika - Unos podataka

1. Korisnik **kontroliše** da li je korektno uneo podatke o proizvodu. (ANSO)
2. Korisnik **poziva** sistem da zapamti podatke o proizvodu. (APSO)

*Opis akcije*: Korisnik klikom na dugme „Sačuvaj“ poziva sistemsku operaciju ***Zapamti(Proizvod)*** koja čuva novi proizvod.

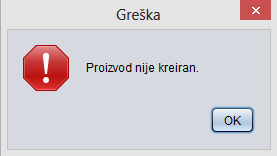
1. Sistem **pamti** podatke o proizvodu. (SO)
2. Sistem **prikazuje** korisniku zapamćeni proizvod i poruku: “Sistem je sačuvao proizvod”. (IA)



Slika - Proizvod je sačuvan

##### **Alternativna scenarija**

3.1 Ukoliko sistem ne može da kreira proizvod on prikazuje korisniku poruku: “Sistem ne može da kreira novi proizvod”. Prekida se izvršenje scenaria. (IA)



Slika - Greška

8.1 Ukoliko sistem ne može da zapamti podatke o proizvodu on prikazuje korisniku poruku “Sistem ne može da zapamti proizvod”. Prekida se izvršenje scenaria. (IA)



Slika - Greška

#### Slučaj korišćenja 6: Izmena proizvoda

##### **Naziv SK**

Izmena proizvoda

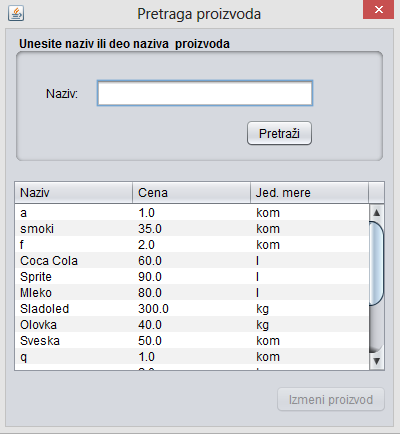
##### **Aktori SK**

Korisnik

##### **Učesnici SK**

Korisnik i sistem (program)

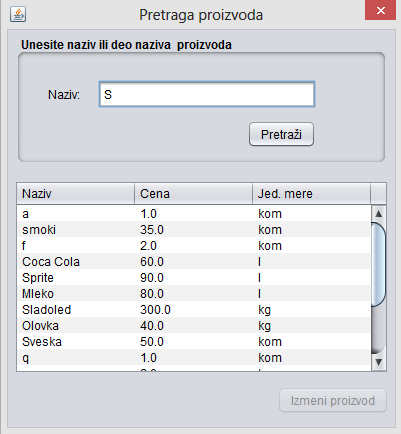
##### **Preduslov**: Sistem je uključen i korisnik je prijavljen pod svojom šifrom. Sistem prikazuje formu za pretragu proizvoda.



Slika - Forma za pretragu proizvoda

##### **Osnovni scenario SK**

1. Korisnik **unosi** vrednost po kojoj pretražuje proizvode. (APUSO)

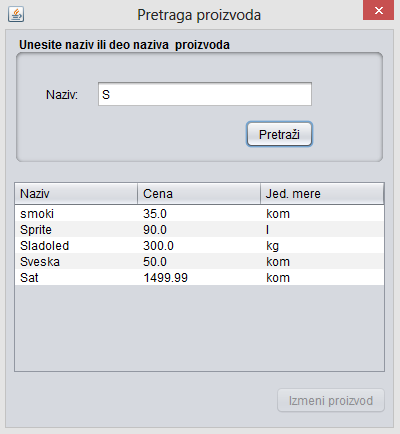


Slika - Unos vrednosti za pretragu

1. Korisnik **poziva** sistem da nađe proizvode po zadatoj vrednosti. (APSO)

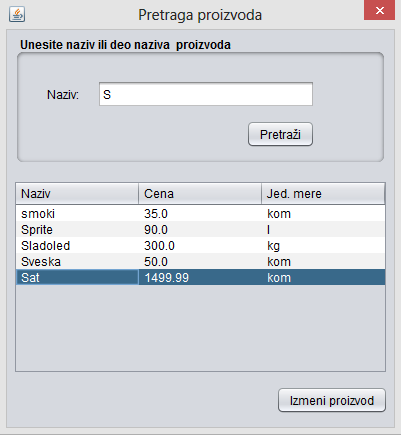
Opis akcije: Korisnik klikom na dugme “Pretraži” poziva sistemsku operaciju PretraziProizvode(List<Proizvod>, KriterijumPretrage) koja vraća listu proizvoda.

1. Sistem **traži** proizvod po zadatoj vrednosti. (SO)
2. Sistem **prikazuje** korisniku pronađene proizvode. (IA)



Slika - Pronađeni proizvodi

1. Korisnik **bira** proizvod. (APUSO)

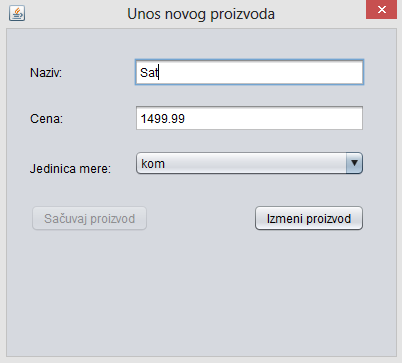


Slika - Korisnik bira proizvod

1. Korisnik **poziva** sistem da nađe sve podatke o izabranom proizvodu. (APSO)

Opis akcije: Korisnik klikom na dugme “Izmeni proizvod” poziva sistemsku operaciju NadjiProizvod(Proizvod, KriterijumPretrage) koja vraća podatke o izabranom proizvodu.

1. Sistem **traži** podatke o izabranom proizvodu. (SO)
2. Sistem **prikazuje** korisniku sve podatke o proizvodu. (IA)



Slika - Podaci o izabranom proizvodu

1. Korisnik **unosi (menja)** podatke o proizvodu. (APUSO)
2. Korisnik **kontroliše** da li je korektno uneo podatke o proizvodu. (ANSO)
3. Korisnik **poziva** sistem da zapamti podatke o proizvodu. (APSO)

*Opis akcije*: Korisnik klikom na dugme „Izmeni proizvod“ poziva sistemsku operaciju ***Zapamti(Proizvod)*** koja čuva proizvod.

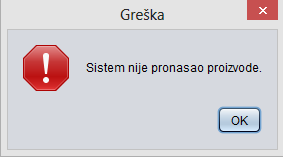
1. Sistem **pamti** podatke o proizvodu. (SO)
2. Sistem **prikazuje** korisniku zapamćen proizvod i poruku: “Sistem je zapamtio proizvod.” (IA)



Slika - Proizvod je sačuvan

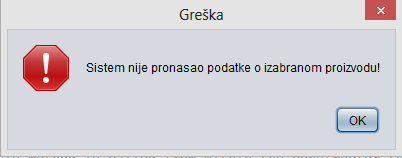
##### **Alternativna scenarija**

4.1 Ukoliko sistem ne može da nađe nijedan proizvod on prikazuje korisniku poruku: “Sistem ne može da nađe proizvod po zadatoj vrednosti”. (IA)



Slika - Greška

8.1 Ukoliko sistem ne može da nađe podatke o proizvodu on prikazuje korisniku poruku: “Sistem ne može da nađe proizvod po zadatoj vrednosti”. (IA)



Slika - Greška

13.1 Ukoliko sistem ne može da zapamti podatke o izmenjenom proizvodu on prikazuje korisniku poruku “Sistem ne može da zapamti izmenjeni proizvod”. Prekida se izvršenje scenaria. (IA)



Slika - Greška

#### Slučaj korišćenja 7: Unos poslovnog partnera

##### **Naziv SK**

Unos novog poslovnog partnera

##### **Aktori SK**

Korisnik

##### **Učesnici SK**

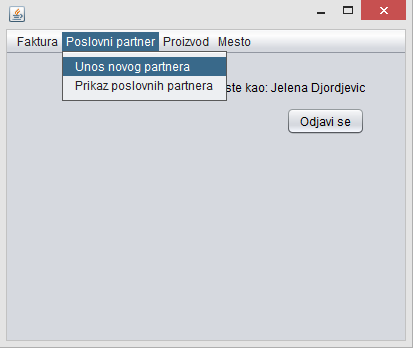
Korisnik i sistem (program)

##### **Preduslov**: Sistem je uključen i korisnik je prijavljen pod svojom šifrom. Sistem prikazuje početnu formu.

##### **Osnovni scenario SK**

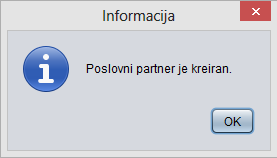
1. Korisnik **poziva** sistem da kreira novog poslovnog partnera. (APSO)

*Opis akcije*: Klikom na stavku menija Poslovni partner, *Unos novog partnera,* korisnik poziva sistemsku operaciju ***kreiraj(PoslovniPartner)*** koja pravi novog proizvod.

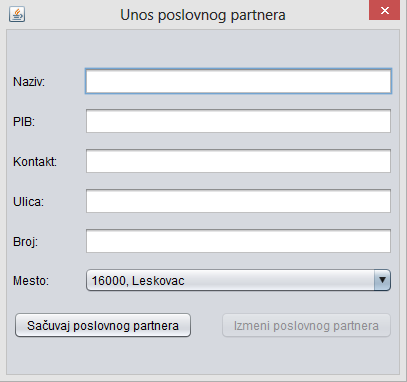


Slika - Kreiranje novog partnera

1. Sistem **kreira** novog poslovnog partnera. (SO)
2. Sistem **prikazuje** korisniku novog poslovnog partnera i poruku: “Sistem je kreirao novog poslovnog partnera”. (IA)



Slika – Poruka



Slika - Kreiran poslovni partner

1. Korisnik **unosi** podatke o novom poslovnom partneru. (APUSO)



Slika - Unos podataka

1. Korisnik **kontroliše** da li je korektno uneo podatke o poslovnom partneru. (ANSO)
2. Korisnik **poziva** sistem da zapamti podatke o poslovnom partneru. (APSO)

*Opis akcije*: Korisnik klikom na dugme „Sačuvaj“ poziva sistemsku operaciju ***Zapamti(PoslovniPartner)*** koja čuva novog poslovnog partnera.

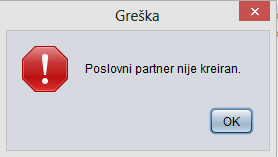
1. Sistem **pamti** podatke o poslovnom partneru. (SO)
2. Sistem **prikazuje** korisniku zapamćenog poslovnog partnera i poruku: “Sistem je sačuvao poslovnog partnera”. (IA)



Slika - Partner je sačuvan

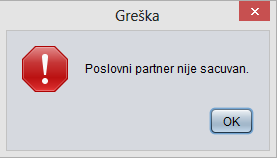
##### **Alternativna scenarija**

3.1 Ukoliko sistem ne može da kreira poslovnog partnera on prikazuje korisniku poruku: “Sistem ne može da kreira novog poslovnog partnera”. Prekida se izvršenje scenaria. (IA)



Slika - Greška

8.1 Ukoliko sistem ne može da zapamti podatke o poslovnom partneru on prikazuje korisniku poruku “Sistem ne može da zapamti poslovnog partnera”. Prekida se izvršenje scenaria. (IA)



Slika – Greška

#### Slučaj korišćenja 8: Izmena poslovnog partnera

##### **Naziv SK**

Izmena poslovnog partnera

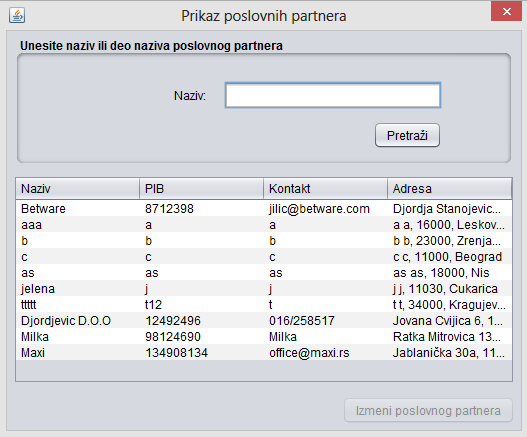
##### **Aktori SK**

Korisnik

##### **Učesnici SK**

Korisnik i sistem (program)

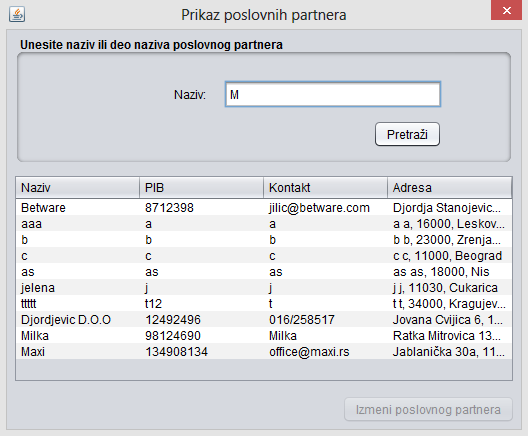
##### **Preduslov**: Sistem je uključen i korisnik je prijavljen pod svojom šifrom. Sistem prikazuje formu za pretragu poslovnih partnera.



Slika - Forma za pretragu poslovnih partnera

##### **Osnovni scenario SK**

1. Korisnik **unosi** vrednost po kojoj pretražuje poslovne partnere. (APUSO)

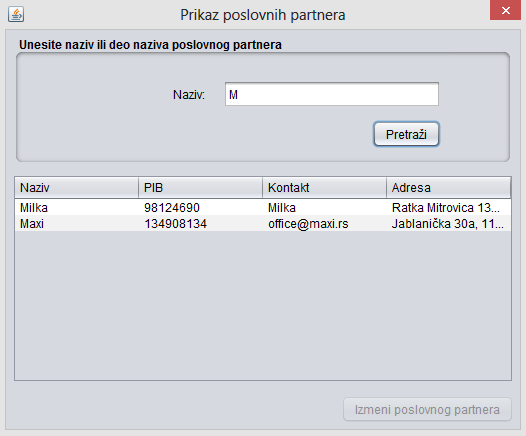


Slika - Korisnik unosi kriterijum pretrage

1. Korisnik **poziva** sistem da nađe poslovne partnere po zadatoj vrednosti. (APSO)

Opis akcije: Korisnik klikom na dugme “Pretraži” poziva sistemsku operaciju Pretrazi (List<PoslovniPartner>, KriterijumPretrage) koja vraća listu poslovnih partnera.

1. Sistem **traži** poslovne partnere po zadatoj vrednosti. (SO)
2. Sistem **prikazuje** korisniku pronađene poslovne partnere. (IA)



Slika - Pronađeni partneri

1. Korisnik **bira** poslovnog partnera. (APUSO)

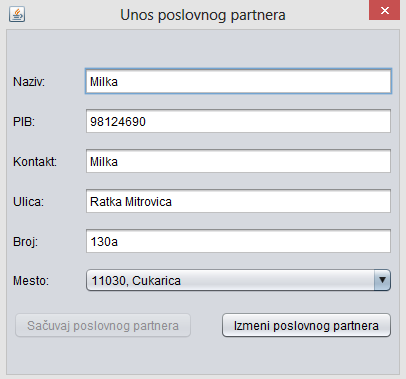


Slika - Korisnik bira partnera

1. Korisnik **poziva** sistem da nađe sve podatke o poslovnom partneru. (APSO)

Opis akcije: Korisnik klikom na dugme “Izmeni poslovnog partnera” poziva sistemsku operaciju Nadji (PoslovniPartner, KriterijumPretrage) koja vraća podatke o izabranom poslovnom partneru.

1. Sistem **traži** podatke o izabranom poslovnom partneru. (SO)
2. Sistem **prikazuje** korisniku sve podatke o poslovnom partneru. (IA)



Slika - Podaci o poslovnom partneru

1. Korisnik **unosi (menja)** podatke o poslovnom partneru. (APUSO)
2. Korisnik **kontroliše** da li je korektno uneo podatke o poslovnom partneru. (ANSO)
3. Korisnik **poziva** sistem da zapamti podatke o poslovnom partneru. (APSO)

*Opis akcije*: Korisnik klikom na dugme „Izmeni poslovnog partnera“ poziva sistemsku operaciju ***Zapamti(PoslovniPartner)*** koja čuva poslovnog partnera.

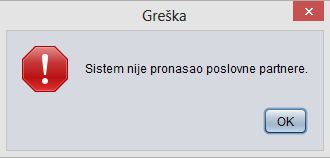
1. Sistem **pamti** podatke o poslovnom partneru. (SO)
2. Sistem **prikazuje** korisniku zapamćenog poslovnog partnera i poruku: “Sistem je zapamtio poslovnog partnera.” (IA)



Slika - Poslovni partner je izmenjen

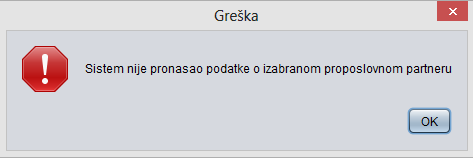
##### **Alternativna scenarija**

4.1 Ukoliko sistem ne može da nađe nijedanog poslovnog partnera on prikazuje korisniku poruku: “Sistem ne može da nađe poslovnog partnera po zadatoj vrednosti”. (IA)



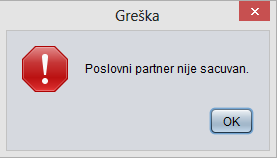
Slika - Greška

8.1 Ukoliko sistem ne može da nađe podatke o poslovnom partneru on prikazuje korisniku poruku: “Sistem ne može da nađe poslovnog partnera po zadatoj vrednosti”. (IA)



Slika - Greška

13.1 Ukoliko sistem ne može da zapamti podatke o izmenjenom poslovnom partneru on prikazuje korisniku poruku “Sistem ne može da zapamti izmenjenog poslovnog partnera ”. Prekida se izvršenje scenaria. (IA)



Slika - Greška

#### Slučaj korišćenja 9: Prijavljivanje korisnika

##### **Naziv SK:**

Prijavljivanje korisnika

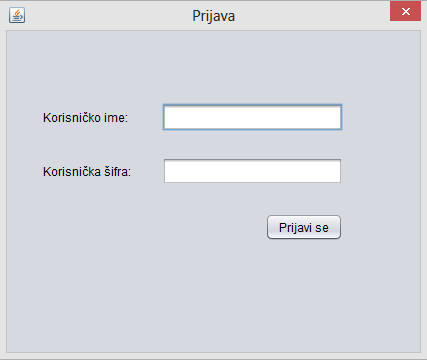
##### **Aktori SK:**

Korisnik

##### **Učesnici SK:**

Korisnik i sistem (program)

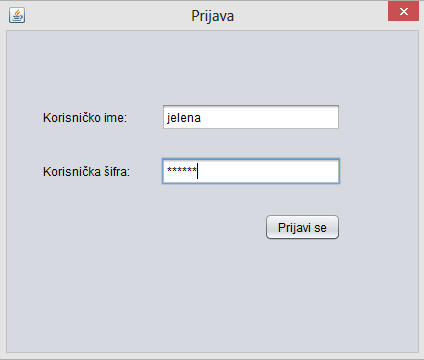
##### **Preduslov:** Sistem je uključen i prikazuje formu za prijavu.



Slika - Forma za prijavu

##### **Osnovni scenario SK:**

1. Korisnik **unosi** podatke za autentifikaciju korisnika. (APUSO)

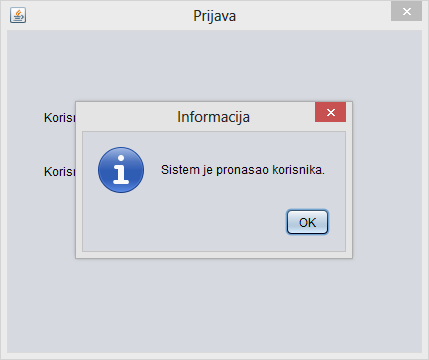


Slika - Unos podataka za autentifikaciju

1. Korisnik **poziva** sistem da pronađe korisnika sa zadatim podacima. (APSO)

*Opis akcije*: Korisnik klikom na dugme „Prijavi se“ poziva sistemsku operaciju ***NadjiKorisnika(Korisnik)***.

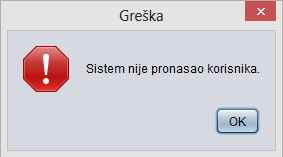
1. Sistem **pretražuje** korisnike. (SO)
2. Sistem **prikazuje** poruku da je korisnik uspešno prijavljen i početnu formu . (IA)



Slika - Uspešna prijava

##### **Alternativna scenarija:**

4.1 Ukoliko sistem ne može da nađe korisnika prikazuje poruku da korisnik nije uneo tačne podatke. (IA)



Slika - Greška

### Projektovanje kontrolera korisničkog interfejsa

Kontroler korisničkog interfejsa predstavlja sponu između ekranske forme i aplikacione logike. U daljem tekstu dat je primer dva kontrolera.

#### KontorlerKIUnosPoslovnogPartnera

public class KontorlerKIUnosPoslovnogPartnera {

private final PoslovniPartner model;

private final FmUnosPoslovnogPartnera view;

public KontorlerKIUnosPoslovnogPartnera(Frame mainView){

model = new PoslovniPartner();

view = new FmUnosPoslovnogPartnera(mainView, true, model);

kreirajPoslovnogPartnera();

view.postaviVrednosti(false);

}

public KontorlerKIUnosPoslovnogPartnera(Frame mainView, PoslovniPartner poslovniPartner){

model = poslovniPartner;

view = new FmUnosPoslovnogPartnera(mainView, true, model);

view.postaviVrednosti(true);

}

public void pokreniFormu(){

postaviOsluskivace();

prikaziFormu();

}

private void prikaziFormu() {

view.setVisible(true);

}

private void kreirajPoslovnogPartnera(){

try {

int poslovniPartnerId = KontrolerKI.generisiID(model);

model.setId(poslovniPartnerId);

TransferObjekat toZahtev = new TransferObjekat();

toZahtev.setOperacija(konstante.Operacije.KREIRAJ\_POSLOVNOG\_PARTNERA);

toZahtev.setParametar(model);

Komunikacija.vratiObjekat().posalji(toZahtev);

TransferObjekat toOdgovor = Komunikacija.vratiObjekat().procitaj();

String poruka = toOdgovor.getPoruka();

if (toOdgovor.getIzuzetak() != null) {

poruka += "\n" + ((Exception)toOdgovor.getIzuzetak()).getMessage();

JOptionPane.showMessageDialog(view, poruka, "Greška", JOptionPane.ERROR\_MESSAGE);

} else {

JOptionPane.showMessageDialog(view, poruka, "Informacija", JOptionPane.INFORMATION\_MESSAGE);

}

} catch (Exception ex) {

ex.printStackTrace();

JOptionPane.showMessageDialog(view, ex.getMessage(), "Greška", JOptionPane.ERROR\_MESSAGE);

}

}

private void zapamtiPoslovnogPartnera() {

try {

model.setNaziv(KonverterTipova.konvertuj(view.getTxtNaziv(), String.class));

model.setPib(KonverterTipova.konvertuj(view.getTxtPib(), String.class));

model.setKontakt(KonverterTipova.konvertuj(view.getTxtKontakt(), String.class));

model.getAdresa().setUlica(KonverterTipova.konvertuj(view.getTxtUlica(), String.class));

model.getAdresa().setBroj(KonverterTipova.konvertuj(view.getTxtBroj(), String.class));

if(view.getCbMesto().getSelectedIndex() != -1)

model.getAdresa().setMesto(KonverterTipova.konvertuj(view.getCbMesto(), Mesto.class));

if (model.getNaziv().isEmpty() || model.getPib().isEmpty() || model.getKontakt().isEmpty() ||

model.getAdresa().getUlica().isEmpty() || model.getAdresa().getBroj().isEmpty()) {

throw new Exception("Sva polja su obavezna.");

}

TransferObjekat<PoslovniPartner> toZahtev = new TransferObjekat();

toZahtev.setOperacija(Operacije.SACUVAJ\_POSLOVNOG\_PARTNERA);

toZahtev.setParametar(model);

Komunikacija.vratiObjekat().posalji(toZahtev);

TransferObjekat toOdogovor = Komunikacija.vratiObjekat().procitaj();

String poruka = toOdogovor.getPoruka();;

if (toOdogovor.getIzuzetak() != null) {

poruka += "\n" + ((Exception)toOdogovor.getIzuzetak()).getMessage();

JOptionPane.showMessageDialog(view, poruka, "Greška", JOptionPane.ERROR\_MESSAGE);

} else {

JOptionPane.showMessageDialog(view, poruka, "Informacija", JOptionPane.INFORMATION\_MESSAGE);

}

} catch (Exception ex) {

ex.printStackTrace();

JOptionPane.showMessageDialog(view, ex.getMessage(), "Greška", JOptionPane.ERROR\_MESSAGE);

}

}

public PoslovniPartner getModel() {

return model;

}

public FmUnosPoslovnogPartnera getView() {

return view;

}

private void postaviOsluskivace() {

view.getBtnSnimi().addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent ae) {

zapamtiPoslovnogPartnera();

}

});

view.getBtnIzmeni().addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent ae) {

int o = JOptionPane.showConfirmDialog(view, "Da li ste sigurni da želite da izmenite poslovnog partnera?",

"Potvrda", JOptionPane.YES\_NO\_CANCEL\_OPTION, JOptionPane.QUESTION\_MESSAGE);

switch (o) {

case JOptionPane.YES\_OPTION:

zapamtiPoslovnogPartnera();

break;

case JOptionPane.NO\_OPTION:

view.dispose();

break;

default:

break;

}

}

});

}

}

#### KontrolerKIPrikazPoslovnihPartnera

public class KontrolerKIPrikazPoslovnihPartnera {

private final List<PoslovniPartner> model;

private final FmPrikazPartneraTabela view;

public KontrolerKIPrikazPoslovnihPartnera(Frame mainView) {

model = vratiPoslovnePartnere(null);

view = new FmPrikazPartneraTabela(mainView, true, model);

view.postaviVrednosti();

}

public void pokreniFormu(){

postaviOsluskivace();

prikaziFormu();

}

public static List<PoslovniPartner> vratiPoslovnePartnere(Dialog dialog) {

try {

TransferObjekat kto = new TransferObjekat();

kto.setOperacija(Operacije.VRATI\_SVE\_POSLOVNE\_PARTNERE);

kto.setParametar(new PoslovniPartner());

Komunikacija.vratiObjekat().posalji(kto);

TransferObjekat sto = Komunikacija.vratiObjekat().procitaj();

if (sto.getIzuzetak() != null) {

throw (Exception)sto.getIzuzetak();

}

List<PoslovniPartner> lista = (List<PoslovniPartner>)sto.getRezultat();

return lista;

} catch (Exception ex) {

ex.printStackTrace();

JOptionPane.showMessageDialog(dialog, ex.getMessage(), "Greška", JOptionPane.ERROR\_MESSAGE);

}

return Collections.EMPTY\_LIST;

}

private void prikaziPoslovnePartnere(){

try {

TransferObjekat toZahtev = new TransferObjekat();

toZahtev.setOperacija(Operacije.PRETRAZI\_POSLOVNE\_PARTNERE);

PoslovniPartner p = new PoslovniPartner();

p.setNaziv(view.getTxtNaziv1().getText());

toZahtev.setParametar(p);

Komunikacija.vratiObjekat().posalji(toZahtev);

TransferObjekat toOdgovor = Komunikacija.vratiObjekat().procitaj();

if (toOdgovor.getIzuzetak() != null) {

throw (Exception)toOdgovor.getIzuzetak();

}

List<PoslovniPartner> lista = (List<PoslovniPartner>)toOdgovor.getRezultat();

view.getTblPartneri().setModel(new ModelTabelePoslovniPartner(lista));

} catch (Exception ex) {

ex.printStackTrace();

JOptionPane.showMessageDialog(view, ex.getMessage(), "Greška", JOptionPane.ERROR\_MESSAGE);

}

}

private void prikaziPodatke(PoslovniPartner poslovniPartner){

try {

TransferObjekat toZahtev = new TransferObjekat();

toZahtev.setOperacija(Operacije.PRIKAZI\_POSLOVNOG\_PARTNERA);

toZahtev.setParametar(poslovniPartner);

Komunikacija.vratiObjekat().posalji(toZahtev);

TransferObjekat toOdgovor = (TransferObjekat) Komunikacija.vratiObjekat().procitaj();

if (toOdgovor.getIzuzetak() != null) {

JOptionPane.showMessageDialog(view, toOdgovor.getPoruka(), "Greška", JOptionPane.ERROR\_MESSAGE);

} else {

JOptionPane.showMessageDialog(view, toOdgovor.getPoruka(), "Informacija", JOptionPane.INFORMATION\_MESSAGE);

PoslovniPartner p = (PoslovniPartner) toOdgovor.getRezultat();

KontorlerKIUnosPoslovnogPartnera k = new KontorlerKIUnosPoslovnogPartnera((Frame) view.getParent(), p);

k.pokreniFormu();

}

} catch (Exception ex) {

ex.printStackTrace();

JOptionPane.showMessageDialog(view, ex.getMessage(), "Greška", JOptionPane.ERROR\_MESSAGE);

}

}

private void postaviOsluskivace() {

view.getBtnPretraga1().addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent ae) {

prikaziPoslovnePartnere();

}

});

view.getBtnIzmeni().addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent ae) {

if (view.getTblPartneri().getSelectedRow() == -1) {

return;

}

try {

PoslovniPartner poslovniPartner = ((ModelTabelePoslovniPartner) view.getTblPartneri().getModel()).

getPoslovniPartner(view.getTblPartneri().getSelectedRow());

prikaziPodatke(poslovniPartner);

prikaziPoslovnePartnere();

view.getBtnIzmeni().setEnabled(false);

} catch (Exception ex) {

ex.printStackTrace();

}

}

});

view.getTblPartneri().addMouseListener(new MouseAdapter() {

@Override

public void mouseClicked(MouseEvent me) {

if (view.getTblPartneri().getSelectedRow() != -1) {

view.getBtnIzmeni().setEnabled(true);

}else{

view.getBtnIzmeni().setEnabled(false);

}

}

});

}

private void prikaziFormu() {

view.setVisible(true);

}

}

## Projektovanje aplikacione logike

### Generics mehanizam

Pri projektovanju aplikacione logike korišnjen je Generics mehanizam. On omogućava da se parametrizuju interfejsi i klase kako be se omogućilo pisanje generičkih metoda čija logika ne zavisi od tipova njenih parametara.

Broker baze podataka i kontroler aplikacione logike sadrže generičke metode, dok su sistemske operacije implementirane kao generički tipovi. Ovo omogućava da povratna vrednost metoda kontrolera bude tačno određenog tipa. Na primer, ako metodi *vratiSveObjekte* prosledimo objekat tipa Mesto, povratna vrednost je List<Mesto>, a ne List<OpstiDomenskiObjekat>. Na taj način se izbegava kastovanje i obezbeđuje sigurnost tipova u vreme kompajliranja (eng. Type Safety).

### Kontroler aplikacione logike

public class Kontroler {

private static Kontroler k;

private OpstaSistemskaOperacija oso;

private Kontroler() {

}

public static Kontroler vratiObjekat() {

if (k == null) {

k = new Kontroler();

}

return k;

}

public void sacuvajObjekat(OpstiDomenskiObjekat odo) throws Exception{

oso = new SacuvajObjekat(odo);

oso.izvrsenjeSO();

}

public <T extends OpstiDomenskiObjekat> List<T> vratiSveObjekte(T odo) throws Exception {

oso = new VratiSveObjekte(odo);

oso.izvrsenjeSO();

return ((VratiSveObjekte)oso).getLista();

}

public <T extends OpstiDomenskiObjekat> List<T> vratiObjekteSaVezanimObjektom(T odo)throws Exception{

oso = new VratiObjekteSaVezanimObjektom(odo);

oso.izvrsenjeSO();

return ((VratiObjekteSaVezanimObjektom)oso).getLista();

}

public <T extends OpstiDomenskiObjekat> T kreirajObjekat(T odo) throws Exception {

KreirajNoviObjekat<T> oso = new KreirajNoviObjekat<>(odo);

oso.izvrsenjeSO();

return ((KreirajNoviObjekat<T>)oso).getObjekat();

}

public int generisiID(OpstiDomenskiObjekat odo) throws Exception {

oso = new GenerisiId(odo);

oso.izvrsenjeSO();

return ((GenerisiId)oso).getId();

}

public void kreirajISacuvajObjekat(OpstiDomenskiObjekat odo) throws Exception {

oso = new KreirajIsacuvajObjekat(odo);

oso.izvrsenjeSO();

}

public <T extends OpstiDomenskiObjekat> List<T> pretraziObjekte(T odo) throws Exception {

oso = new PretraziObjekte(odo);

oso.izvrsenjeSO();

return ((PretraziObjekte)oso).getLista();

}

public <T extends OpstiDomenskiObjekat> List<T> pretraziSaVezanimObjektom(T odo) throws Exception {

oso = new PretraziObjekteSaVezanimObjektom(odo);

oso.izvrsenjeSO();

return ((PretraziObjekteSaVezanimObjektom)oso).getLista();

}

public <T extends OpstiDomenskiObjekat> T prikaziPodatke(T odo) throws Exception {

oso = new PrikaziPodatke(odo);

oso.izvrsenjeSO();

return ((PrikaziPodatke<T>)oso).getObjekat();

}

public <T extends OpstiDomenskiObjekat> T prikaziPodatkeSaVezanimObjektom(T odo) throws Exception {

oso = new PrikaziPodatkeSaVezanimObjektom(odo);

oso.izvrsenjeSO();

return ((PrikaziPodatkeSaVezanimObjektom<T>)oso).getObjekat();

}

public void sacuvajObjekatSaStavkama(OpstiDomenskiObjekat odo) throws Exception{

oso = new SacuvajFakturuSaStavkama(odo);

oso.izvrsenjeSO();

}

public <T extends OpstiDomenskiObjekat> T prikaziPodatkeSaVezanimObjektomIStavkama(T odo) throws Exception{

oso = new PrikaziPodatkeSaVezanimObjektomIStavkama<>(odo);

oso.izvrsenjeSO();

return ((PrikaziPodatkeSaVezanimObjektomIStavkama<T>)oso).getObjekat();

}

public void obradi(OpstiDomenskiObjekat odo) throws Exception {

oso = new ObradiFakturu(odo);

oso.izvrsenjeSO();

}

public void storniraj(OpstiDomenskiObjekat odo) throws Exception {

oso = new StornirajFakturu(odo);

oso.izvrsenjeSO();

}

}

### Poslovna logika

Za svaki od ugovora o sistemskim operacijama koje smo prethodno definisali, projektuje se konceptualno rešenje.

#### Opšta sistemska operacija

public abstract class OpstaSistemskaOperacija <T extends OpstiDomenskiObjekat>{

protected DatabaseBroker dbbr;

protected T odo;

public OpstaSistemskaOperacija(T odo) {

dbbr = new DatabaseBroker();

this.odo = odo;

}

public void izvrsenjeSO() throws Exception{

otvoriBazu();

try{

izvrsiOperaciju();

potvrdi();

}catch(Exception e){

odbaci();

throw e;

}

zatvoriBazu();

}

private void otvoriBazu() throws Exception {

dbbr.ucitajDriver();

dbbr.otvoriKonekciju();

}

public abstract void izvrsiOperaciju() throws Exception;

private void potvrdi() throws Exception {

dbbr.commitTransakcije();

}

private void odbaci() throws Exception {

dbbr.rollbackTransakcije();

}

private void zatvoriBazu() throws Exception {

dbbr.zatvoriKonekciju();

}

}

#### KreirajNoviObjekat

public class KreirajNoviObjekat<T extends OpstiDomenskiObjekat> extends OpstaSistemskaOperacija{

T objekat;

public KreirajNoviObjekat(T odo) {

super(odo);

}

@Override

public void izvrsiOperaciju() throws Exception {

dbbr.kreiraj(odo);

}

public T getObjekat() {

return objekat;

}

}

#### SacuvajObjekat

public class SacuvajObjekat extends OpstaSistemskaOperacija{

public SacuvajObjekat(OpstiDomenskiObjekat odo) {

super(odo);

}

@Override

public void izvrsiOperaciju() throws Exception {

if (!odo.vrednosnaOgranicenja()) {

throw new Exception("Nisu zadovoljena vrednosna ograničenja!");

}

dbbr.sacuvaj(odo);

}

}

#### PretraziObjekte

public class PretraziObjekte<T extends OpstiDomenskiObjekat> extends OpstaSistemskaOperacija<T>{

List<T> lista;

public PretraziObjekte(T odo) {

super(odo);

}

@Override

public void izvrsiOperaciju() throws Exception {

lista = dbbr.pretraziObjekte(odo);

}

public List<T> getLista() {

return lista;

}

}

#### PrikaziPodatke

public class PrikaziPodatke <T extends OpstiDomenskiObjekat> extends OpstaSistemskaOperacija<T>{

T objekat;

public PrikaziPodatke(T odo) {

super(odo);

}

@Override

public void izvrsiOperaciju() throws Exception {

objekat = dbbr.dajPodatke(odo);

}

public T getObjekat() {

return objekat;

}

}

#### SacuvajFakturuSaStavkama

public class SacuvajFakturuSaStavkama extends OpstaSistemskaOperacija {

public SacuvajFakturuSaStavkama(OpstiDomenskiObjekat odo) {

super(odo);

}

@Override

public void izvrsiOperaciju() throws Exception {

if (!preduslov()) {

throw new Exception("Nisu ispiunjeni preduslovi!");

}

if (!odo.vrednosnaOgranicenja()) {

throw new Exception("Nisu zadovoljena vrednosna ograničenja!");

}

dbbr.brisiStavke(odo);

for (int i = 0; i < odo.vratiBrojStavki(); i++) {

dbbr.brisiSlabObjekat(odo.vratiStavku(i), odo);

}

dbbr.sacuvaj(odo);

for (int i = 0; i < odo.vratiBrojStavki(); i++) {

dbbr.sacuvajSlabObjekat(odo.vratiStavku(i), odo);

}

}

private boolean preduslov() throws Exception {

return !(dbbr.vratiLogickuVrednostAtributa("obradjena", odo)

|| dbbr.vratiLogickuVrednostAtributa("stornirana", odo));

}

}

#### PrikaziPodatkeSaVezanimObjektomIStavkama

public class PretraziObjekteSaVezanimObjektom <T extends OpstiDomenskiObjekat> extends OpstaSistemskaOperacija<T>{

List<T> lista;

public PretraziObjekteSaVezanimObjektom(T odo) {

super(odo);

}

@Override

public void izvrsiOperaciju() throws Exception {

lista = dbbr.pretraziObjekte(odo);

for (T opstiDomenskiObjekat : lista) {

opstiDomenskiObjekat.setVezaniObjekat(dbbr.dajPodatke(opstiDomenskiObjekat.vratiVezaniObjekat()));

}

}

public List<T> getLista() {

return lista;

}

}

#### VratiSveObjekte

public class VratiSveObjekte <T extends OpstiDomenskiObjekat> extends OpstaSistemskaOperacija<T>{

private List<T> lista;

public VratiSveObjekte(T odo) {

super(odo);

}

@Override

public void izvrsiOperaciju() throws Exception {

lista = dbbr.vratiSve(odo);

}

public List<T> getLista() {

return lista;

}

}

#### ObradiFakturu

public class ObradiFakturu extends OpstaSistemskaOperacija {

public ObradiFakturu(OpstiDomenskiObjekat odo) {

super(odo);

}

@Override

public void izvrsiOperaciju() throws Exception {

if (!preduslov()) {

throw new Exception("Nisu ispiunjeni preduslovi!");

}

if (!odo.vrednosnaOgranicenja()) {

throw new Exception("Nisu zadovoljena vrednosna ograničenja!");

}

dbbr.brisiStavke(odo);

for (int i = 0; i < odo.vratiBrojStavki(); i++) {

dbbr.brisiSlabObjekat(odo.vratiStavku(i), odo);

}

dbbr.sacuvaj(odo);

for (int i = 0; i < odo.vratiBrojStavki(); i++) {

dbbr.sacuvajSlabObjekat(odo.vratiStavku(i), odo);

}

}

private boolean preduslov() throws Exception {

return !(dbbr.vratiLogickuVrednostAtributa("obradjena", odo)

|| dbbr.vratiLogickuVrednostAtributa("stornirana", odo));

}

}

#### StornirajFakturu

public class StornirajFakturu extends OpstaSistemskaOperacija {

public StornirajFakturu(OpstiDomenskiObjekat odo) {

super(odo);

}

@Override

public void izvrsiOperaciju() throws Exception {

if (!preduslov()) {

throw new Exception("Nisu ispiunjeni preduslovi!");

}

if (!odo.vrednosnaOgranicenja()) {

throw new Exception("Nisu zadovoljena vrednosna ograničenja!");

}

dbbr.brisiStavke(odo);

for (int i = 0; i < odo.vratiBrojStavki(); i++) {

dbbr.brisiSlabObjekat(odo.vratiStavku(i), odo);

}

dbbr.sacuvaj(odo);

for (int i = 0; i < odo.vratiBrojStavki(); i++) {

dbbr.sacuvajSlabObjekat(odo.vratiStavku(i), odo);

}

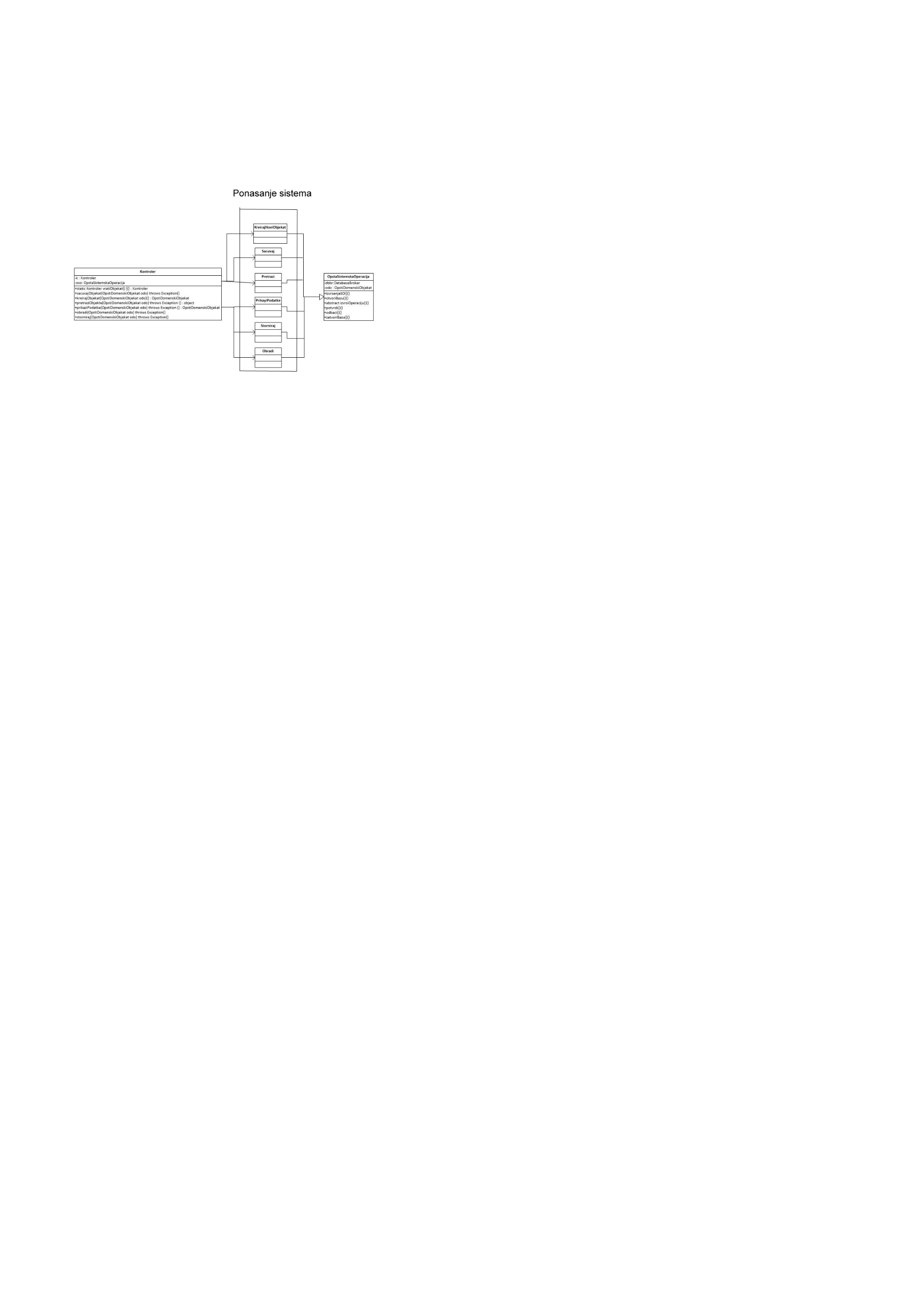
}

private boolean preduslov() throws Exception {

return !(dbbr.vratiLogickuVrednostAtributa("stornirana", odo));

}

}



Slika 109 - Klase koje su odgovorne za SO nasledjuju SistemskuOperaciju

#### Projektovanje strukture softverskog sistema

Na osnuvu konceptualnih klasa (slika 34), prave se softverske klase strukture:

##### Faktura:

public class Faktura {

private int idFakture;

private String brojFakture;

private PoslovniPartner poslovniPartner;

private double ukupnaVrednost;

private boolean obradjena;

private boolean stornirana;

private List<StavkaFakture> stavke;

private double osnovica;

private double pdv;

private String brojOtpremnice;

private Date datum;

public Faktura(int idFakture, String brojFakture, PoslovniPartner poslovniPartnerId, double ukupnaVrednost, boolean obradjena, boolean stornirana, List<StavkaFakture> stavke, double osnovica, double pdv, String brojOtpremnice, Date datum) {

this.idFakture = idFakture;

this.brojFakture = brojFakture;

this.poslovniPartner = poslovniPartnerId;

this.ukupnaVrednost = ukupnaVrednost;

this.obradjena = obradjena;

this.stornirana = stornirana;

this.stavke = stavke;

this.osnovica = osnovica;

this.pdv = pdv;

this.brojOtpremnice = brojOtpremnice;

this.datum = datum;

}

}

##### StavkaFakture

public class StavkaFakture {

private int rb;

private double kolicina;

private double rabat;

private double iznosRabata;

private double poreskaOsnovica;

private double pdv;

private double iznosPdva;

private double vrednostSaPdvom;

private Proizvod proizvod;

public StavkaFakture(int rb, double kolicina, double rabat, double iznosRabata, double poreskaOsnovica, double pdv, double iznosPdva, double vrednostSaPdvom, Proizvod proizvod) {

this.rb = rb;

this.kolicina = kolicina;

this.rabat = rabat;

this.iznosRabata = iznosRabata;

this.poreskaOsnovica = poreskaOsnovica;

this.pdv = pdv;

this.iznosPdva = iznosPdva;

this.vrednostSaPdvom = vrednostSaPdvom;

this.proizvod = proizvod;

}

}

##### Proizvod

public class Proizvod {

private int id;

private String naziv;

private double cena;

private JedMere jedinicaMere;

public Proizvod(int id, String naziv, double cena, JedMere jedinicaMere) {

this.id = id;

this.naziv = naziv;

this.cena = cena;

this.jedinicaMere = jedinicaMere;

}

}

##### PoslovniPartner

public class PoslovniPartner {

private int id;

private String pib;

private String naziv;

private String kontakt;

private Adresa adresa;

public PoslovniPartner(int id, String pib, String naziv, String kontakt, Adresa adresa) {

this.id = id;

this.pib = pib;

this.naziv = naziv;

this.kontakt = kontakt;

this.adresa = adresa;

}

}

##### Mesto

public class Mesto{

private int ptt;

private String naziv;

public Mesto(int ptt, String naziv) {

this.ptt = ptt;

this.naziv = naziv;

}

}

##### Adresa

public class Adresa {

private String ulica;

private String broj;

private Mesto mesto;

public Adresa(String ulica, String broj, Mesto mesto) {

this.ulica = ulica;

this.broj = broj;

this.mesto = mesto;

}

}

##### JedMere

public enum JedMere {

kom, m, l, kg, m2, m3;

}

### Broker baze podataka

public class DatabaseBroker {

private Connection konekcija;

public void ucitajDriver() throws Exception {

Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

}

public void otvoriKonekciju() throws Exception {

konekcija = DriverManager.getConnection("jdbc:odbc:bazaProSoftSem");

konekcija.setAutoCommit(false); // Zahteva eksplicitnu potvrdu transakcije

}

public void zatvoriKonekciju() throws Exception {

konekcija.close();

}

public void commitTransakcije() throws Exception {

try {

konekcija.commit();

} catch (SQLException ex) {

throw new Exception("Transakcija nije potvrđena!");

}

}

public void rollbackTransakcije() throws Exception {

try {

konekcija.rollback();

} catch (SQLException ex) {

throw new Exception("Transakcija nije potvrđena!");

}

}

public void sacuvaj(OpstiDomenskiObjekat odo) throws Exception {

try {

String sql = "UPDATE " + odo.vratiNazivTabele()

+ " SET " + odo.vratiAtributeSaVrednostima() + " WHERE "

+ odo.vratiNazivKljuca() + " = " + odo.vratiVrednostKljuca();

System.out.println(sql);

Statement sqlNaredba = konekcija.createStatement();

sqlNaredba.executeUpdate(sql);

sqlNaredba.close();

} catch (SQLException ex) {

ex.printStackTrace();

throw new Exception("Greška pri čuvanju!");

}

}

public <T extends OpstiDomenskiObjekat> List<T> vratiSve(T odo) throws Exception {

try {

String sql = "SELECT \* FROM " + odo.vratiNazivTabele();

System.out.println(sql);

Statement sqlNaredba = konekcija.createStatement();

ResultSet rs = sqlNaredba.executeQuery(sql);

List<T> lista = odo.napuni(rs);

return lista;

} catch (SQLException ex) {

ex.printStackTrace();

throw new Exception("Greška pri učitavanju!");

}

}

public <V extends OpstiDomenskiObjekat> V kreiraj(V odo) throws Exception {

try {

String upit = "INSERT INTO " + odo.vratiNazivTabele() + "("

+ odo.vratiNazivKljuca() + ")" + " VALUES (" + odo.vratiVrednostKljuca() + ")";

System.out.println(upit);

Statement sqlNaredba = konekcija.createStatement();

sqlNaredba.executeUpdate(upit);

sqlNaredba.close();

return odo;

} catch (SQLException ex) {

ex.printStackTrace();

throw new Exception("Greška pri kreiranju objekta!");

}

}

public int generisiId(OpstiDomenskiObjekat odo) throws Exception {

try {

int broj = 0;

String upit = "SELECT LAST(" + odo.vratiNazivKljuca() + ") AS poslednjiId FROM " + odo.vratiNazivTabele();

System.out.println(upit);

Statement sqlNaredba = konekcija.createStatement();

ResultSet rs = sqlNaredba.executeQuery(upit);

if (rs.next()) {

broj = rs.getInt("poslednjiId") + 1;

}

return broj;

} catch (SQLException ex) {

ex.printStackTrace();

throw new Exception("Greška pri generisanju ID-a!");

}

}

public void kreirajISacuvaj(OpstiDomenskiObjekat odo) throws Exception {

try {

String upit = "INSERT INTO " + odo.vratiNazivTabele() + " VALUES (" + odo.vratiVrednostiAtributa() + ")";

System.out.println(upit);

Statement sqlNaredba = konekcija.createStatement();

sqlNaredba.executeUpdate(upit);

sqlNaredba.close();

} catch (SQLException ex) {

ex.printStackTrace();

throw new Exception("Greška pri kreiranju i čuvanju objekta!");

}

}

public <T extends OpstiDomenskiObjekat> List<T> pretraziObjekte(T odo) throws Exception {

try {

String sql = "SELECT \* FROM " + odo.vratiNazivTabele() + " WHERE " + odo.vratiKriterijumPretrage();

System.out.println(sql);

Statement sqlNaredba = konekcija.createStatement();

ResultSet rs = sqlNaredba.executeQuery(sql);

return odo.napuni(rs);

} catch (SQLException ex) {

ex.printStackTrace();

throw new Exception("Greška pri pretraživanju!");

}

}

public <T extends OpstiDomenskiObjekat> T dajPodatke(T odo) throws Exception {

try {

String upit = "SELECT \* FROM " + odo.vratiNazivTabele() + " WHERE " + odo.vratiNazivKljuca()

+ "=" + odo.vratiVrednostKljuca();

System.out.println(upit);

Statement sqlNaredba = konekcija.createStatement();

ResultSet rs = sqlNaredba.executeQuery(upit);

List<T> lista = odo.napuni(rs);

return lista.get(0);

} catch (Exception ex) {

ex.printStackTrace();

throw new Exception("Greška pri izvlačenju podataka!");

}

}

public void brisiStavke(OpstiDomenskiObjekat odo) throws Exception {

try {

String upit;

OpstiDomenskiObjekat odo2;

for (int i = 0; i < odo.vratiBrojStavki(); i++) {

odo2 = odo.vratiStavku(i);

upit = "DELETE FROM " + odo2.vratiNazivTabele() + " WHERE " + odo2.vratiNazivKljuca()

+ "=" + odo.vratiVrednostKljuca() + " AND " + odo.vratiNazivKljuca() + "="

+ odo.vratiVrednostKljuca();

System.out.println(upit);

Statement sqlNaredba = konekcija.createStatement();

sqlNaredba.executeUpdate(upit);

}

} catch (Exception ex) {

ex.printStackTrace();

throw new Exception("Greška pri brisanju!");

}

}

public void brisiSlabObjekat(OpstiDomenskiObjekat slab, OpstiDomenskiObjekat jak) throws Exception {

try {

String upit = "DELETE FROM " + slab.vratiNazivTabele() + " WHERE " + slab.vratiNazivKljuca()

+ "=" + slab.vratiVrednostKljuca() + " AND " + jak.vratiNazivKljuca() + "="

+ jak.vratiVrednostKljuca();

System.out.println(upit);

Statement sqlNaredba = konekcija.createStatement();

sqlNaredba.executeUpdate(upit);

} catch (Exception ex) {

ex.printStackTrace();

throw new Exception("Greška pri brisanju!");

}

}

public void sacuvajSlabObjekat(OpstiDomenskiObjekat slab, OpstiDomenskiObjekat jak) throws Exception {

try {

String upit = "INSERT INTO " + slab.vratiNazivTabele() + " VALUES ("

+ slab.vratiVrednostiAtributa() + ", " + jak.vratiVrednostKljuca() + ")";

System.out.println(upit);

Statement sqlNaredba = konekcija.createStatement();

sqlNaredba.executeUpdate(upit);

sqlNaredba.close();

} catch (SQLException ex) {

ex.printStackTrace();

throw new Exception("Greška pri kreiranju i čuvanju objekta!");

}

}

public <T extends OpstiDomenskiObjekat> List<T> vratiSlabeObjekte(T odo) throws Exception {

try {

String upit = "SELECT \* FROM " + odo.vratiTabeluStavke() + " WHERE " + odo.vratiNazivKljuca()

+ "=" + odo.vratiVrednostKljuca();

System.out.println(upit);

Statement sqlNaredba = konekcija.createStatement();

ResultSet rs = sqlNaredba.executeQuery(upit);

return odo.vratiJednuStavku().napuni(rs);

} catch (Exception ex) {

ex.printStackTrace();

throw new Exception("Greška pri izvlačenju podataka!");

}

}

public boolean vratiLogickuVrednostAtributa(String atribut, OpstiDomenskiObjekat odo) throws Exception {

try {

String upit = " SELECT \*"

+ " FROM " + odo.vratiNazivTabele()

+ " WHERE " + odo.vratiNazivKljuca() + "=" + odo.vratiVrednostKljuca();

System.out.println(upit);

boolean s = false;

Statement sqlNaredba = konekcija.createStatement();

ResultSet rs = sqlNaredba.executeQuery(upit);

rs.next();

s = rs.getBoolean(atribut);

return s;

} catch (Exception ex) {

ex.printStackTrace();

throw new Exception("Greška pri izvlačenju podataka!");

}

}

}

U procesu pravljenja generičkih metoda DatabaseBroker klase dobili smo metode interfejsa OpstiDomenskiObjekat:

public interface OpstiDomenskiObjekat {

public String vratiNazivTabele();

public String vratiAtributeSaVrednostima();

public <T extends OpstiDomenskiObjekat> List<T> napuni(ResultSet rs) throws Exception;

public String vratiNazivKljuca();

public Object vratiVrednostKljuca();

public String vratiVrednostiAtributa();

public String vratiKriterijumPretrage()

public int vratiBrojStavki();

public OpstiDomenskiObjekat vratiStavku(int rb);

public OpstiDomenskiObjekat vratiVezaniObjekat();

public <T extends OpstiDomenskiObjekat> void setVezaniObjekat(T odo);

public String vratiTabeluStavke();

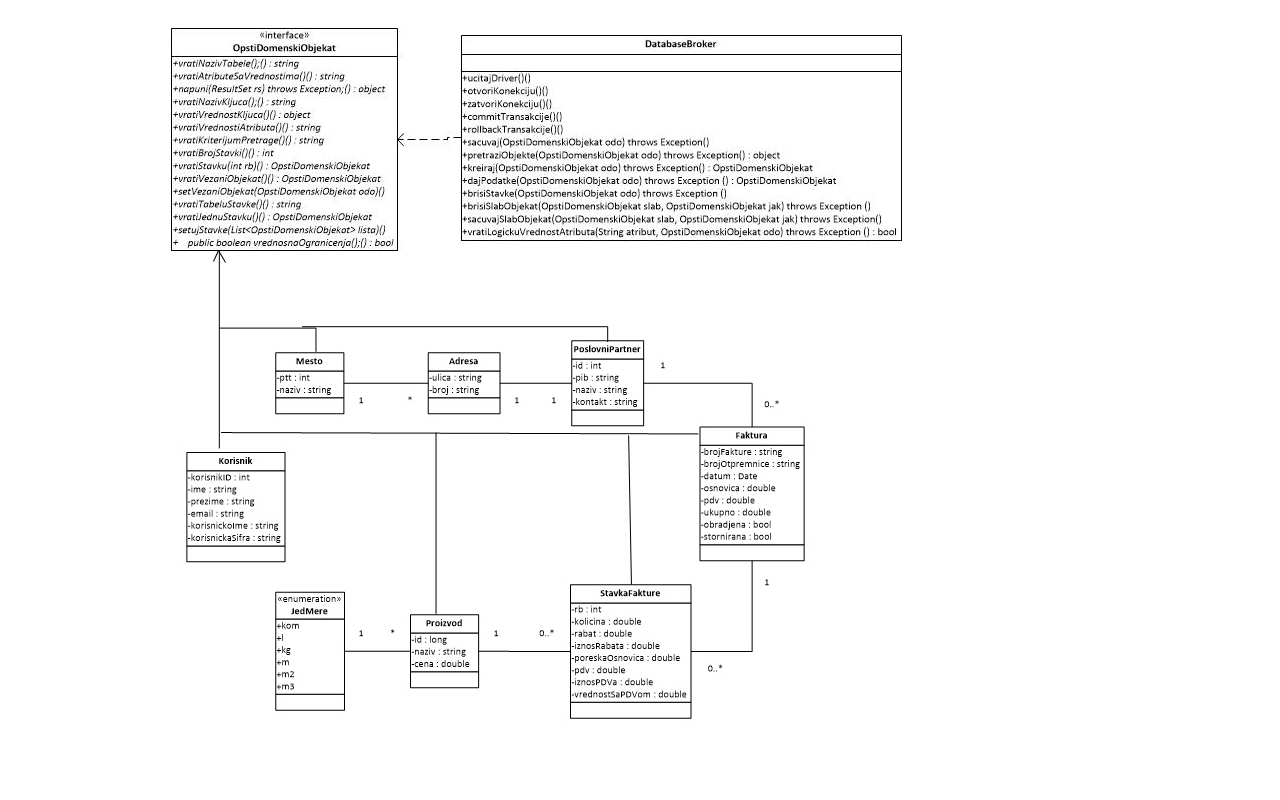
public OpstiDomenskiObjekat vratiJednuStavku();

public <T extends OpstiDomenskiObjekat> void setujStavke(List<T> lista);

public boolean vrednosnaOgranicenja();

}

Kao rezultat projektovanja klase DatabaseBrokera i interfejsa OpstiDomenskiObjekat dobijaju se sleći dijagrami:



Slika 110 - DatabaseBroker se povezuje sa klasom OpstiDomenskiObjekat

## Projektovanje skladišta podataka

Na osnovu softverskih klasa strukture projektovali smo tabele relacionog sistema za upravljanje bazom podataka.

#### Table: Faktura

**Columns**

Name Type Size

idFakture Long Integer 4

brojFakture Text 15

brojOtpremnice Text 15

datum Date/Time 8

osnovica Double 8

ukupnoPdv Double 8

ukupnaVrednost Double 8

obradjena Yes/No 1

stornirana Yes/No 1

poslovniPartnerId Long Integer 4

**Table Indexes**

Name Number of Fields

idFakture 1

Fields:

idFakture Ascending

PoslovniPartnerFaktura 1

Fields:

poslovniPartnerId Ascending

poslovniPartnerId 1

Fields:

poslovniPartnerId Ascending

PrimaryKey 1

Fields:

idFakture Ascending

#### Table: StavkaFakture

**Columns**

Name Type Size

rb Long Integer 4

kolicina Double 8

rabat Double 8

iznosRabata Double 8

poreskaOsnovica Double 8

pdv Double 8

iznosPdva Double 8

vrednostSaPdvom Double 8

proizvodId Long Integer 4

idFakture Long Integer 4

**Table Indexes**

Name Number of Fields

FakturaStavkaFakture 1

Fields:

idFakture Ascending

PrimaryKey 2

Fields:

rb Ascending

idFakture Ascending

proizvodId 1

Fields:

proizvodId Ascending

ProizvodStavkaFakture 1

Fields:

proizvodId Ascending

#### Table: PoslovniPartner

**Columns**

Name Type Size

poslovniPartnerId Long Integer 4

pib Text 9

naziv Text 50

kontakt Text 50

ulica Text 50

broj Text 10

ptt Long Integer 4

**Table Indexes**

Name Number of Fields

MestoPoslovniPartner 1

Fields:

ptt Ascending

PrimaryKey 1

Fields:

poslovniPartnerId Ascending

#### Table: Proizvod

**Columns**

Name Type Size

proizvodId Long Integer 4

naziv Text 50

cena Double 8

jedinicaMere Text 20

**Table Indexes**

Name Number of Fields

PrimaryKey 1

Fields:

proizvodId Ascending

#### Table: Mesto

**Columns**

Name Type Size

ptt Long Integer 4

naziv Text 50

**Table Indexes**

Name Number of Fields

PrimaryKey 1

Fields:

ptt Ascending

#### Table: Korisnik

**Columns**

Name Type Size

ID Long Integer 4

ime Text 30

prezime Text 30

korisnickoIme Text 15

korisnickaSifra Text 15

email Text 50

**Table Indexes**

Name Number of Fields

PrimaryKey 1

Fields:

ID Ascending

# Principi, metode i strategije projektovanja softvera

## Principi (tehnike) projektovanja softvera

### Apstrakcija

Apstrakcija je “proces svesnog zaboravljanja informacija tako da stvari koje su različite mogu biti tretirane kao da su iste“, pri čemu se pod zaboravljanjem informacija podrazumeva izdvajanje opštih od specifičnih informacija. Opšte informacije se izdvajaju kako bi se naglasila suština neke pojave.

U kontekstu softverskog projektovanja , postoje dva ključna mehanizma apstrakcije:

* parametrizacija i
* specifikacija

Apstrakcija specifikacijom vodi do tri glavne vrste apstrakcija:

* proceduralna apstrakcija
* apstrakcija podataka
* apstrakcija kontrolom

Parametrizacija je apstrakcija koja iz nekog skupa elemenata izdvaja njihova opšta svojstva koja su predstavljena preko parametara.

Različite vrste parametrizacije su:

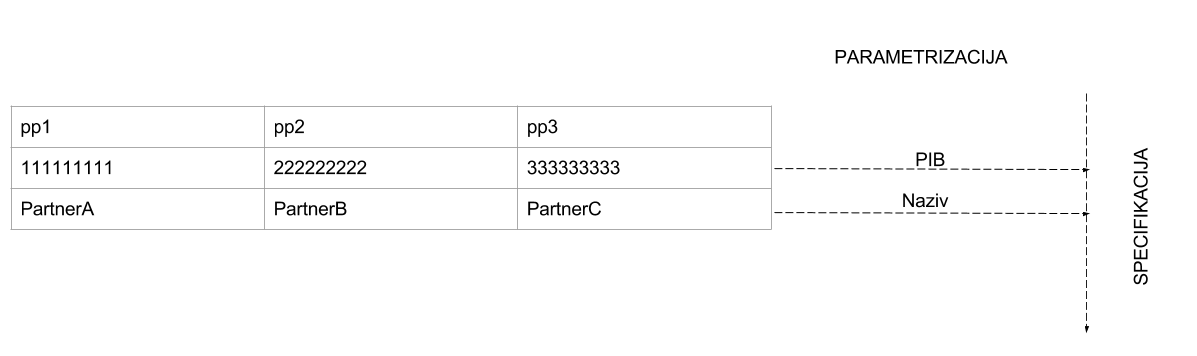
* parametrizacija skupa elemenata prostog tipa
* parametrizacija skupa elemenata složenog tipa
* parametrizacija skupa operacija
* parametrizacija skupa procedura
* parametrizacija skupa naredbi

Parametrizacija skupa elemenata prostog tipa:  
Skup celih brojeva se *1, 2, 3,...* se može prestaviti preko parametra *x* koji je tada opšti predstavnik svih celih brojeva. Ovo se može prestaviti kodom:  
*int x;*  
To znači da je deklarisana promenljiva x koja je celobrojnog tipa. Deklarisanjem promenljive prostog tima implicitno se ukazuje na skup operacija koje se mogu izvršiti nad njom.

Parametrizacija skupa elementa složenog tipa

Ako imamo skup poslovnih partnera ((11111111, PartnerA), (222222222, PartnerB), (333333333, PartnerC)), tada se parametrizacijom dobijaju njihova opšta svojstva (atributi): *PIB* i *Naziv.* Navedeni skup je označen sa pp1, pp2, pp3.

Parametrizacija se radi za svaki skup vrednosti atributa. Skup PIB vrednosti (11111111, 222222222, 333333333) se parametrizuje preko svojstva *PIB*, dok se skup naziva (PartnerA, PartnerB, PartnerC) parametrizuje preko svojstva *Naziv*.



Parametrizacijom dolazimo do opštih svojstava elemenata skupa. Navođenje opštih svojstava elemenata skupa prestavlja specifikaciju skupa. Apstrakcija podataka je definisana imenom i specifikacijom skupa: Poslovni Partner (PIB, Naziv).

Može se primetiti da parametrizacija prethodi specifikaciji, jer specifikacija koristi rezultate parametrizacije.

Parametrizacija skupa operacija

Parametrizacijom skupa nekih operacija ((5+2),(7+1),(3+8)) dobija se opšta operacija: *x+y*. Elementi operacije su operandi *x* i *y* i operator *+*. Navedeni skup konkretnih operacija je označen sa *op1, op2, op3*.

Parametrizacija se radi za svaki skup vrednosti elemenata operacije. Skup vrednosti (5, 7, 3) se parametrizuje preko svojstva x, dko se skup vrednosti (2, 1, 8) parametrizuje preko svojstva y. Operator + se koristi kod svake od navedenih operacija pa se zbog toga zadržava kao zajedničko svojstvo svih navedenih operacija.



Rezultat parametrizacije je x + y, opšta operacija za sabiranje dva broja.

Parametrizacijom dolazimo do opštih svojstava operacije. Navođenje opštih svojstava operacije predstavlja specifikaciju operacije. Specifikacija operacije nabraja parametre (elemente) operacije (operande i operator).

Parametrizacija skupa procedura

Ukoliko imamo skup nekih procedura:

*T dajPodatke(T odo) {metoda koja vraća objekat tipa Mesto}  
T dajPodatke(T odo) { metoda koja vraća objekat tipa Korisnik}  
T dajPodatke(T odo) { metoda koja vraća objekat tipa Proizvod}*

tada se parametrizacijom dobija opšta procedura:

*T dajPodatke(T odo)*

Elementi neke procedure su: šta vraća, naziv procedure, argumenti i telo procedure.

Parametrizacija se radi za svaki skup vrednosti elemenata procedure.



Za rezultat dobijamo:  
T dajPodatke(T odo)

Parametrizacijom dolazimo do opštih svojstava procedura. Navođenjem opštih svojstava procedura dolazi se do potpisa procedure. Potpis procedure je jedan od delova specifikacije procedure (proceduralna apstrakcija).

Parametrizacija skupa naredbi

Ukoliko imamo skup naredbi:

System.out.println(1 + “. Proizvod je “ + proizvodi[0].getNaziv());  
System.out.println(2 + “. Proizvod je “ + proizvodi[1].getNaziv());  
System.out.println(3 + “. Proizvod je “ + proizvodi[2].getNaziv());

koje prikauju elemente niza, tada se parametrizacijom dobija opšta naredba:

System.out.println(**i** + “. Proizvod je “ + proizvodi[**i**].getNaziv());

Parametrizacija se radi za svaki skup vrednosti elemenata naredbe.



Kao rezultat dobija se:

for(int i=0; i<3; i++)  
 System.out.println(**i** + “. Proizvod je “ + proizvodi[**i**].getNaziv());

Parametrizacijom dolazimo do opštih svojstava naredbi. Navođenjem opštih svojstava naredbi dolazi se do specifikacije naredbe koja postaje opšta naredba. Navedena specifikacija je ustvari apstrakcija naredbi, jedan od oblika apstrakcije kontrolom.

Specifikacija je apstrakcija koja izdvaja iz nekog skupa elemenata njihova opšta svojstva koja mogu biti predstavljena preko procedure, podataka ili kontrole.

Proceduralna apstrakcija  
Ovim apstrakcijom se izdvaja iz nekog skupa procedura ono što su njihova opšta svojstva:

* tip onoga što vraća procedura
* ime procedure
* argumenti procedure

Proceduralnom apstrakcijom se dobija **potpis** procedure. Pored potpisa proceduralna apstrakcija obuhvata i dopunske sekcije koje detaljno opisuju **šta** radi procedura i koji su **uslovi** izvršenja procedure. Opšti slučaj proceduralne apstrakcije ima sledeće elemente, shodno uprošćenoj Larmanovoj metodi: Potpis procedure, Veza sa SK, Preduslovi i Postuslovi.

Kao primer možemo navesti ugovor o sistemskoj operaciji KreirajNoviProizvod:

**Potpis procedure: KreirajNoviProizvod**(Proizvod):signal;

**Veza sa SK**: SK5

**Preduslovi**:

**Postuslovi**: Kreiran je novi proizvod.   
 Zadovoljena strukturna ograničenja nad tabelom Proizvod.

Apstrakcija podataka  
Apstrakcijom podataka se izdvaja iz nekog skupa podataka ono što su njegova opšta svojstva.

Ako imamo skup poslovnih partnera ((11111111, PartnerA), (222222222, PartnerB), (333333333, PartnerC)), tada se parametrizacijom dobijaju njihova opšta svojstva (atributi): *PIB* i *Naziv.*

Navođenje opštih svojstava elemenata skupa (PIB i Naziv) predstavlja specifikaciju skupa. Apstrakcija podataka je definisana imenom (Poslovni partner) skupa i specifikacijom skupa (PIB i Naziv).

Ukoliko elementi skupa imaju i strukturu i ponašanje (objekti) tada se nad njima radi i proceduralna apstrakcija i apstrakcija podataka. Kao rezultat navedenih apstrakcija se dobija klasa. **Klasa** se sastoji iz atributa i procedura (metoda) koji su učaureni u klasi.

Ukoliko elementi skupa imaju samo ponašanje tada se nad njima radi proceduralna apstrakcija koja kao rezultat daje interfejs. **Interfejs** se sastoji od skupa potpisa procedura (operacija).

Apstrakcija kontrolom  
Postoje dva oblika apstrakcije kontrole:

* Apstrakcija naredbi, kojom se izdvaja iz nekog skupa naredbi ono što su njihova opšta svojstva i predstavljaju se preko kontrolne strukture i opšte naredbe.
* Apstrakcija struktura podataka, kojoj se izdvaja iz nekog skupa struktura podataka ono što su njihova opšta svojstva i predstavljaju se preko iteratora, koji u opštem smislu kontroliše kretanje kroz strukturu podataka

### Spojenost i kohezija

U razvoju objektno-orijentisanog softvera potrebno je postići dva cilja:

* treba izgraditi klase koje imaju visoku koheziju (high cohesion);
* klase treba da budu slabo povezane (weak coupling).

Kohezija je mera kojom se utvrđuje koliko su metode međusobno povezane. Data je klasa X:

*class X*

*{*

*public m1(){m11();m12();m13();…}*

*private m11(){…}*

*m12(){…}*

*m13(){…}*

*}*

Metode m11, m12 i m13 učestvuju u ponašanju m1, pa se za ovakvu klasu kaže da ima visoku koheziju. Metode m11, m12 i m13 su međusobno povezane jer zajedno obezbeđuju neko ponašanje, ali je i metoda m1 povezana sa metodama m11, m12 i m13 jer one obezbeđuju njeno ponašanje.

Visoka kohezija se može uočiti kod klasa koje su odgovorne za izvršenje sistemskih operacija. Dat je primer apstraktne klase *OpstaSistemskaOperacija:*

*public abstract class OpstaSistemskaOperacija <T extends OpstiDomenskiObjekat>{*

*...*

*public void izvrsenjeSO() throws Exception{*

*otvoriBazu();*

*try{*

*izvrsiOperaciju();*

*potvrdi();*

*}catch(Exception e){*

*odbaci();*

*throw e;*

*}*

*zatvoriBazu();*

*}*

*private void otvoriBazu() throws Exception {*

*dbbr.ucitajDriver();*

*dbbr.otvoriKonekciju();*

*}*

*public abstract void izvrsiOperaciju() throws Exception;*

*private void potvrdi() throws Exception {*

*dbbr.commitTransakcije();*

*}*

*private void odbaci() throws Exception {*

*dbbr.rollbackTransakcije();*

*}*

*private void zatvoriBazu() throws Exception {*

*dbbr.zatvoriKonekciju();*

*}*

*}*

Sve metode u klasi učestvuju u realizaciji metode *izvrsenjeSO*.

Gubitak kohezije nastaje kada klasa (Y) treba da obezbedi više različitih ponašanja, koja nisu međusobno povezana. Ovakve klase su teške za održavanje i nadogradnju.

*class Y*

*{ …*

*public m1(){m11();m12();m13();…}*

*m2(){m21();m22(); …}*

*m3(){m31();m32(); …}*

*private m11(){…}*

*m12(){…}*

*m13(){…}*

*…*

*m21(){…}*

*m22() {…}*

*…*

*m31(){…}*

*m32() {…}*

*…*

*}*

U datom primeru vidimo da su metode koje su grupisane po ponašanju povezane. Tako su metode m11, m12 i m13 povezane preko metode m1. Metode m21 i m22 su povezane preko metode m2, dok su metode m31 i m32 povezane preko m3. Između metoda iz različitih grupa ne postoji povezanost.*.*

Spojenost znači da su klase u međusobnoj zavisnosti. To znači da klasa ne može da obavi neku operaciju ako ne postoji neka druga klasa.

Na primer, data je klasa X koja je treba da obezbedi neko ponašanje definisano metodom m1() tako što poziva metodu m2 klase Y. Klasa X je zavisna od klase Y.

**class X**

{ Y y;

X(){y=new Y();}

public m1(){y.m2();}

}

**class Y**

{

m2(){...}

}

Povezanost je mera vezanosti klase sa drugim klasama, kojom se utvrđuje koliko je jako klasa povezana sa drugim klasama. Klasa sa visokom ili snažnom povezanošću zavisi od drugih klasa. Takve klase nisu poželjne jer dovode do sledećih problema:

* Promene povezanih klasa utiču na promenu klasa koje su zavisne od njih.
* Teško je takvu klasu posmatrati izolovanu.
* Teško je ponovo koristiti takvu klasu jer ona zahteva prisustvo drugih klasa od kojih zavisi.

Klasa sa niskom ili slabom povezanošću ne zavisi mnogo od drugih klasa. Takva klasa se može lakše ponovo koristiti u razvoju drugih aplikacija. Zbog toga treba svesti povezanost klasa na najmanju moguću meru. U primeru klasa odgovornih za izvršenje sistemskih operacija se može videti da su te klase slabo povezane.

### Dekompozicija i modularizacija

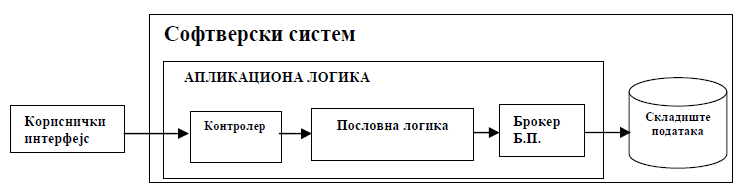
Dekompozicija (raščlanjivanje) je u najopštijem smislu proces koji početni problem deli u skup podproblema, koji se nezavisno rešavaju. Tako rešeni podproblemi omogućavaju da se lakše reši početni problem. Kada dekompoziciju posmatramo u kontekstu razvoja softverskog sistema, može se reći da se dekompozicijom softverski sistem deli u više modula. To znači da **modularizacija** softverskog sistema nastaje kao rezultat procesa dekompozicije.

Dekompozicija se javlja u nekoliko različitih aspekata razvoja softvera:

Dekompozicija kod prikupljanja zahteva   
Korisnički zahtev se dekomponuje na skup zahteva koji se opisuju preko slučajeva korišćenja. Na narednoj slici je dat primer korisničkog zahteva koje je dekomponovan na slučajeve korišćenja.



Dekompozicija kod projektovanja softvera   
Arhitektura softverskog sistema se obično dekomponuje u tri modula: korisnički interfejs, aplikaciona logika i skladište podataka. Ti moduli se dalje dekomponuju (npr. kod Larmanove metode aplikaciona logika se dekomponuje na kontroler korisničkog interfejsa, poslovnu logiku i broker baze podataka). Svaki do modula koji je dobijen dekompozicijom može se nezavisno projektovati i implementirati u odnosu na druge module.

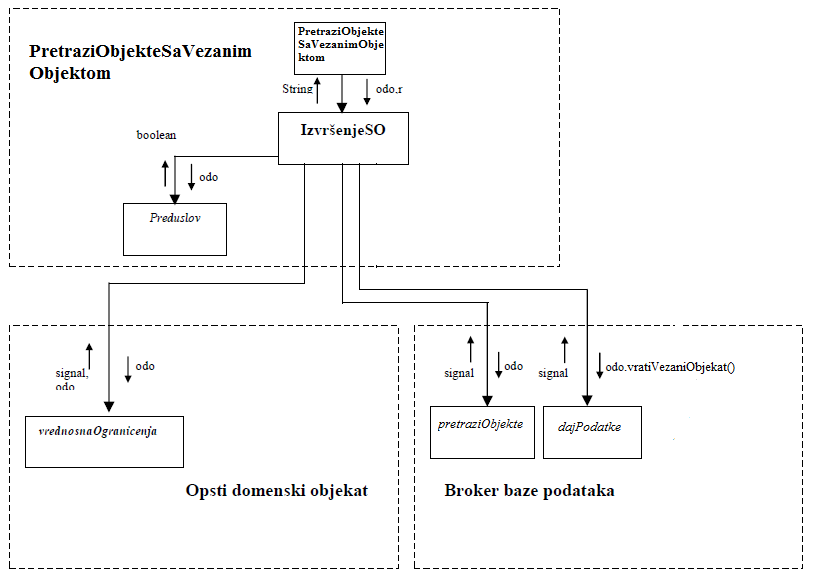


Kada se softverski sistem dekomponuje u module, principi koje treba zadovoljiti su:

* moduli treba da imaju jaku koheziju
* moduli treba da budu što je moguće slabije povezani
* svaki modul treba da čuva svoje interne informacije.

Rezultat procesa dekompozicije softverskog sistema je deljenje softverskog sistema u module. U tom smislu kažemo da je modularizacija rezulat procesa dekompozicije.

Dekompozicija funkcija   
Ako je funkcija (metoda) neke klase složena, može se dekomponovati u više podfunkcija. Ove podfunkcije se zatim nezavisno rešavaju, da bi se na kraju sve integrisale u jednu celinu kako bi se realizovala početna funkcija.



### Učaurenje / Sakrivanje informacija

Učaurenje je proces u kome se razdvajaju osobine modula (npr. klase) koje su javne za druge module od osobina koje su sakrivene od drugih modula sistema. Osobine koje su javne ukazuju na suštinska određenja modula dok osobine koje nisu javne ukazuju na detalje modula. Osobine modula koje nisu javne ne mogu koristiti drugi moduli. **Sakrivanje informacija** nastaje kao rezultat procesa učaurenja, gde se sakrivaju osobine modula koje nisu javne za korisnika modula.

U narednom primeru KontrolerKIMesta ima javni konstruktor i metodu pokreniFormu, dok se atributi model i view privatni. Korisnik ove klase (KontrolerKIGlavna) može pristupiti samo kostruktoru i javnoj metodi.

public class KontrolorKIMesta {

private final List<Mesto> model;

private final FmPrikazMesta view;

public KontrolorKIMesta(Frame mainView) {

model = vratiMesta(null);

view = new FmPrikazMesta(mainView, true, model);

view.postaviVrednosti();

}

...

public void pokreniFormu() {

view.setVisible(true);

}

}

public class KontrolerKIGlavna {

private final FmGlavna view;

private void postaviOsluskivace() {

...

view.getMiPrikazMesta().addActionListener(new ActionListener() {

...

@Override

public void actionPerformed(ActionEvent ae) {

KontrolorKIMesta k = new KontrolorKIMesta(view);

k.pokreniFormu();

}

});

}

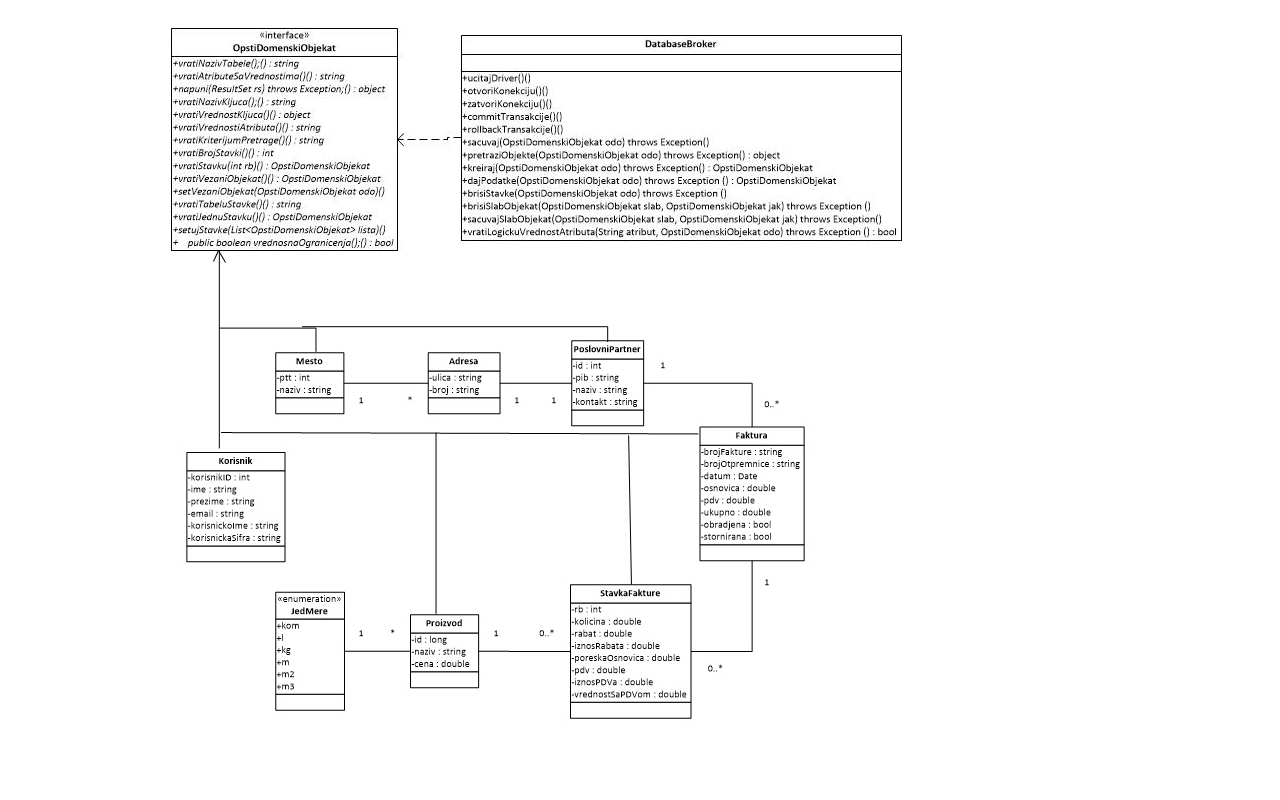
}

Korisnik modula treba da zana samo suštinske detalje modula a ne mora da zna detalje o strukturi (atributi model i view) i implementaciji modula (privatne metode). Moduli dobijeni kao rezultat procesa učaurenja izgledaju kao "crna kutija" za druge module u sistemu.

### Odvajanje interfejsa i implementacije

Interfejs se odvaja od implementacije i izlaže se korisniku. Korisnik ne zna kako su operacije interfejsa implementirane. U narednom primeru korisnik je DatabaseBroker, dok je OpštiDomenskiObjekat interfejs koji izlaže operacije korisniku. Database ne zna kako su operacije domenskih klasa (Račun, StavkaRačuna) implementirane. Navedene domenske klase realizuju interfejs OpstiDomenskiObjekat.

Primer:



### Dovoljnost, kompletnost i jednostavnost

Dovoljnost, kompletnost i jednostavnost ukazuju na osobine softverske komponente koja je **jednostavna** za održavanje i nadogradnju i **dovoljna** i **kompletna** da obezbedi željenu funkcionalnost.

## Strategije projektovanja softvera

Neke od strategija projektovanja softvera su:

* Podeli i pobedi
* S vrha na dole
* Odozgo nagore
* Iterativno - inkrementalni pristup

Podeli i pobedi  
Ova strategija se zasniva na principu dekompozicije koja početni problem u skup podproblema, koji se nezavisno rešavaju. Na taj način se lakše rešava početni problem.

Primer: Ova strategija se najčešće primenjuje u sledećim fazama razvoja softvera:

1. Prikupljanje zahteva – Zahtevi se opisuju preko skupa nezavisnih slučajeva korišćenja
2. Analiza – Struktura se opisuje preko skupa koncepata koji čine konceptualni model. Ponašanje se opisuje preko skupa nezavisnih sistemskih operacija
3. Projektovanje – Arhitektura sistema se deli u tri nivoa: korisnički interfejs, aplikaciona logika i skladište podataka. Korisnički interfejs se deli na dva dela: ekranske forme i kontroler KI. Aplikaciona logika se deli u tri dela: kontroler aplikacione logike, poslovna logika i broker baze podataka.

S vrha na dole  
Ova strategija se zasniva na principu dekompozicije funkcija koja početnu funkciju dekomponuje u više podfunkcija. Ove podfunkcije se zatim nezavisno rešavaju, da bi se na kraju sve podfunkcije integrisale u jednu celinu kako bi se realizovala početna funkcija.

Primer: projektovanje metode izvrsenjeSO() apstraktne klase OpstaSistemskaOperacija, koju realizuju podfunkcije otvoriBazu(), izvrsiOperaciju(), ...

*public abstract class OpstaSistemskaOperacija <T extends OpstiDomenskiObjekat>{*

*...*

*public void izvrsenjeSO() throws Exception{*

*otvoriBazu();*

*try{*

*izvrsiOperaciju();*

*potvrdi();*

*}catch(Exception e){*

*odbaci();*

*throw e;*

*}*

*zatvoriBazu();*

*}*

*private void otvoriBazu() throws Exception {*

*dbbr.ucitajDriver();*

*dbbr.otvoriKonekciju();*

*}*

*public abstract void izvrsiOperaciju() throws Exception;*

*private void potvrdi() throws Exception {*

*dbbr.commitTransakcije();*

*}*

*private void odbaci() throws Exception {*

*dbbr.rollbackTransakcije();*

*}*

*private void zatvoriBazu() throws Exception {*

*dbbr.zatvoriKonekciju();*

*}*

*}*

Odozgo nagore  
Ova strategija se zasniva na principu generalizacije koji u nekoj funkciji, koja je složena, uočava jednu ili više logičkih celina koje proglašava za funkcije. Na taj način je jedna složena funkcija dekomponovana u više nezavisnih funkcija. Kompozicija tih funkcija obezbeđuje istu funkcionalnost kao i složena funkcija.

Primer: projektovanje metode izvrsenjeSO() apstraktne klase OpstaSistemskaOperacija.

Iterativno - inkrementalni pristup  
Sistem se deli u više mini projekata koji se nezavisno razvijaju i prolaze kroz sve faze razvoja softvera. Mini projekat može biti vezan za slučaj korišćenja ili za složenu sistemsku operaciju.

Svaki mini projekat prolazi kroz više **iteracija**, čiji rezultat, interno izdanje (release) ili build predstavlja inkrement za sistem. Na kraju se implementirani mini projekti integrišu u softverski sistem.

Primer: Pri projektovanju softverskog sistema za evidenciju faktura o prodaji imali smo nekoliko nezavisnih slučajeva korišćenja iz kojih smo dobili nekoliko nezavisnih sistemskih operacija. Razvoj pojedinačne sistemske operacije nije zavisio od razvoja bilo koje druge sistemske operacije, odnosno neka sistemska operacija može biti implementirana i integrisana u sistem dok druga sistemska operacija može tek biti identifikovana u fazi analize.

## Metode projektovanja softvera

Najznačajnije metode projektovanja softvera su:

* Funkcionalno orijentisano projektovanje
* Objektno orijentisano projektovanje
* Projektovanje zasnovano na strukturi podataka
* Projektovanje zasnovano na komponentama

Funkcionalno orijentisano projektovanje je zasnovano na funkcijama. Problem se posmatra iz perspektive njegovog ponašanja odnosno funkcionalnosti. Prvo se uočavaju funkcije sistema, zatim se određuju strukture podataka nad kojima se izvršavaju te funkcije.

Objektno orijentisano projektovanje je zasnovano na objektima (klasama). Objekti mogu da reprezentuju i strukturu (atribute) i ponašanje (metode) softverskog sistema. Kod objektno orijentisanog projektovanja paralelno se razvijaju i struktura i ponašanje.

Projektovanje zasnovano na strukturi podataka problem posmatra iz perspektive strukture. Prvo se uočava struktura sistema, zatim se određuju funkcije koje se izvršavaju nad tom strukturom.

Projektovanje zasnovano na komponentama problem posmatra iz perspektive postojećih komponenti koje se mogu (ponovo) koristiti u rešavanju problema. Prvo se uočavaju delovi problema koji se mogu 96 realizovati preko postojećih komponenti a nakon toga se implementiraju delovi za koje nije postojalo rešenje.

### Objektno orijentisano projektovanje

Objektno orijentisano projektovanje je zasnovano na objektima (klasama). Objekti mogu da reprezentuju i strukturu (atribute) i ponašanje (metode) softverskog sistema. Kod ovog načina projektovanja se paralelno razvijaju i struktura i ponašanje.

Primer: Kod projektovanja softverskog sistema za evidenciju faktura o prodaji, kao rezultat analize se dobija logička struktura i ponašanje softverskog sistema. Struktura je predstavljena konceptualnim modelom, a ponašanje sekvencnim dijagramima. Konceptualni model se sastoji od skupa međusobno povezanih koncepata (klasa, odnosno objekata), dok sekvencni dijagram opisuje interakciju izmeu objekata (aktora i softverskog sistema). To znači da su u osnovi logičke strukture i ponašanja softverskog sistema objekti. Zato se ovakva analiza naziva **objektno-orijentisana analiza**. Kasnije se u fazi projektovanja i implementacije određuju klase koje će da realizuju strukturu (domenske klase) i ponašanje (klase koje realizuju sistemske operacije). To znači da su u osnovi fizičke strukture i ponašanja softverskog sistema takođe nalaze objekti, odnosno klase. Zato se ovakvo projektovanje naziva **objektno-orijentisano projektovanje**.

### Principi objektno orijentisanog projektovanja klasa

Kod objektno-orijentisanog programiranja postoje sledeći principi:

1. Princip Otvoreno-Zatvoreno
2. Princip zamene Barbare Liskov
3. Princip inverzije zavisnosti
4. Princip umetanja zavisnosti
5. Princip izdvajanja interfejsa

Princip Otvoreno-Zatvoreno

Po ovom principu modul treba da bude otvoren za proširenje ali zatvoren za modifikaciju.

Ovaj princip se može videti na primeru klase OpstaSistemskaOperacija čije ponašanje se ne može promeniti ali se može proširiti u njenim izvedenim klasama (npr. SacuvajObjekat).

*public abstract class OpstaSistemskaOperacija <T extends OpstiDomenskiObjekat>{*

*...*

*public void izvrsenjeSO() throws Exception{*

*otvoriBazu();*

*try{*

*izvrsiOperaciju();*

*potvrdi();*

*}catch(Exception e){*

*odbaci();*

*throw e;*

*}*

*zatvoriBazu();*

*}*

*private void otvoriBazu() throws Exception {*

*dbbr.ucitajDriver();*

*dbbr.otvoriKonekciju();*

*}*

*public abstract void izvrsiOperaciju() throws Exception;*

*private void potvrdi() throws Exception {*

*dbbr.commitTransakcije();*

*}*

*private void odbaci() throws Exception {*

*dbbr.rollbackTransakcije();*

*}*

*private void zatvoriBazu() throws Exception {*

*dbbr.zatvoriKonekciju();*

*}*

*}*

*public class SacuvajObjekat extends OpstaSistemskaOperacija{*

*public SacuvajObjekat(OpstiDomenskiObjekat odo) {*

*super(odo);*

*}*

*@Override*

*public void izvrsiOperaciju() throws Exception {*

*if (!odo.vrednosnaOgranicenja()) {*

*throw new Exception("Nisu zadovoljena vrednosna ograničenja!");*

*}*

*dbbr.sacuvaj(odo);*

*}*

*}*

Princip zamene Barbare Liskov

Po ovom principu podklase treba da budu zamenjive sa njihovim nadklasama.

Za primer ovog principa može se navesti metoda *vratiSve* DatabaseBroker klase, koja za ulazni argument prima objekat tipa OpstiDomenskiObjekat ali i njegove podklase.

*public <T extends OpstiDomenskiObjekat> List<T> vratiSve(T odo) throws Exception {*

*try {*

*String sql = "SELECT \* FROM " + odo.vratiNazivTabele();*

*System.out.println(sql);*

*Statement sqlNaredba = konekcija.createStatement();*

*ResultSet rs = sqlNaredba.executeQuery(sql);*

*List<T> lista = odo.napuni(rs);*

*return lista;*

*} catch (SQLException ex) {*

*ex.printStackTrace();*

*throw new Exception("Greška pri učitavanju!");*

*}*

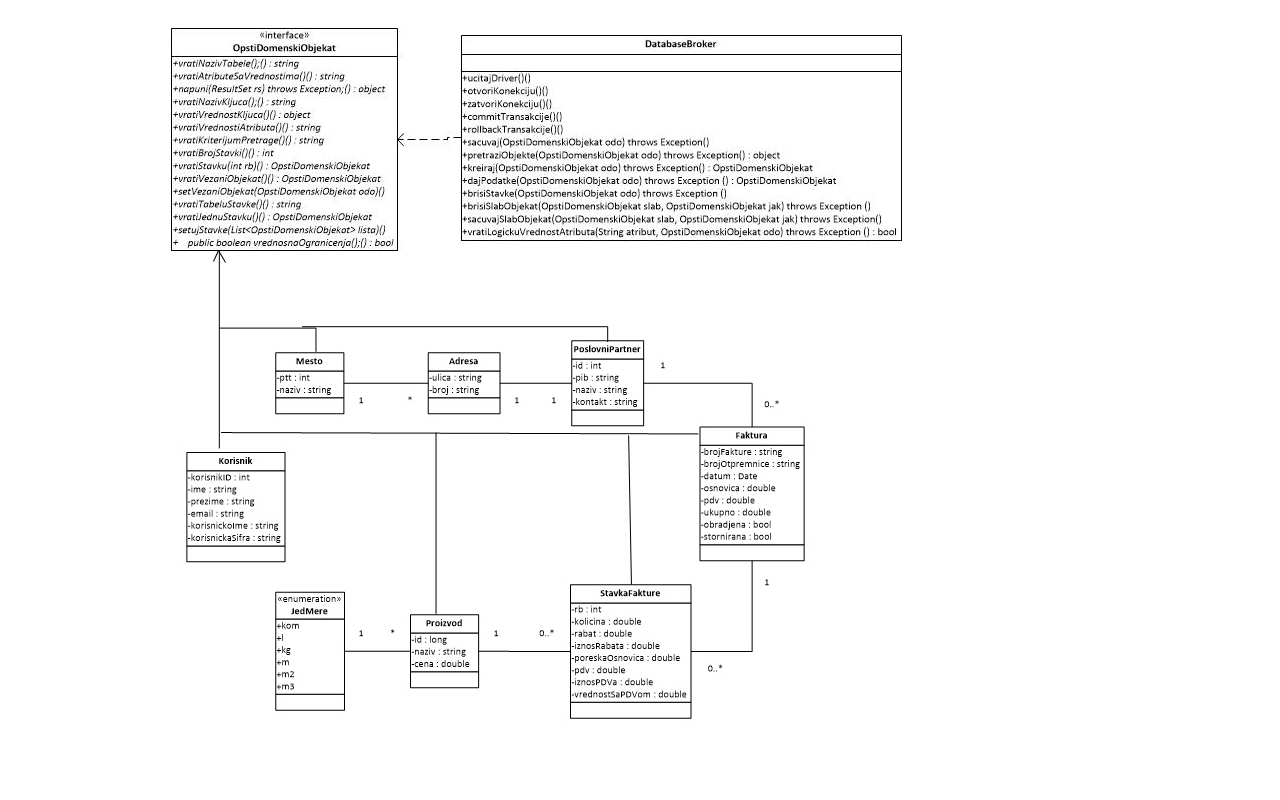
*}*

Princip inverzije zavisnosti

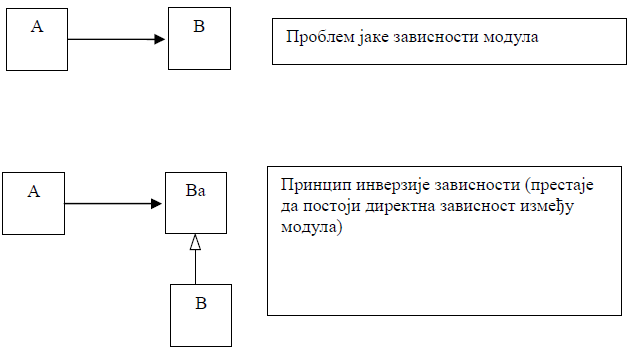
Princip glasi: Zavisi od apstrakcije a ne od konkretizacije.

Moduli najvišeg nivoa ne treba da zavise od modula nižeg nivoa. Oba treba da zavise od apstrakcije. Apstrakcije ne treba da zavise od detalja. Detalji treba da zavise od apstrakcije.

U datom primeru se izbegava direktna zavisnost između DatabaseBroker klase (modul višeg nivoa) i domenskih klasa (moduli nižih nivoa), već se broker povezuje sa interfejsom OpstiDomenskiObjekat koga realizuju domenske klase. Na taj način je DatabaseBroker posredno povezan sa svakom klasom koja implementira OpstiDomenskiObjekat.



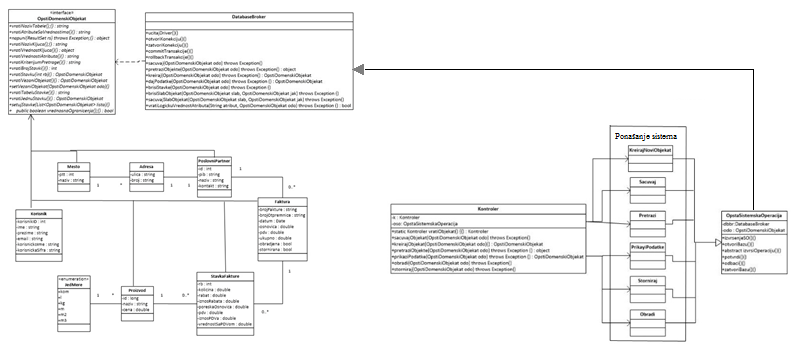
Opšti slučaj:



Princip umetanja zavisnosti

Po ovom principu zavisnosti između dve komponente programa se uspostavljaju u vreme izvršenja programa preko neke treće komponente.

Primer: Zavisnost između DatabaseBroker klase i konkretne domenske klase se uspostavlja u vreme izvršenja programa preko klase OpstaSistemskaOperacija.



Opšti slučaj:

