PROJEKTNI ZADATAK

Školska godina 2021/2022

Sistem za online prodaju karata za bioskope,koncerte,pozorište

Ime: Bojana Stajić
Broj Indeksa: 4596

Predmet: Uvod u softversko inženjerstvo
Šifra predmeta: SE201

Sadržaj

Contents

1. Uvod

Sistem omogućava korisnicima rezervaciju, odnosno kupovinu karata za bioskope, pozorišta i koncerte putem aplikacije online kako bi izbegli čekanje u redovima u okviru prodajnih mesta. Kako bi sistem bio bolje pojašnjen biće upotrebljeni UseCase dijagrami, sekvencijalni dijagrami, klasni dijagrami kao i dokument sa funkcionalnim i nefunkcionalnim zahtevima. Za izradu pomenutih dokumenata koristićemo PowerDesigner program.

2. Svrha dokumenta

Svrha dokumenta ja da se što bolji objasni funkcionalnost i sama ideja sistema za rezervaciju karata u okviru različitih tipova objekata.Dokument se sastoji od uvoda u kome objašnjavamo šta će sve ovaj dokument obuhvatiti, svrhe dokumenta i opisa samog sistema. Zatim sekvencijalni i klasni dijagrami, kao i funkcionalnih i nefunkcionalnih zahtevi koji bliže definišu rad ovog sistema. Na kraju se nalaze Design patterns, testiranje sistema i zaključak.

2.1 Definicije, skraćenice i akronimi

U dokumentu ćemo koristiti profesionalnu terminologiju kao i skraćenice, pa u svrhu upoznavanja i lakšeg razumevanja čitaoca, predstavićemo značenja istih koji se pojavljuju u dokumentu:

- PowerDesigner - grafičko okruženje za modelovanje aplikacija, podataka i poslovnih procesa, koje omogućava i generisanje koda i mnoge druge mogućnosti.
  .
- SOPK – Sistem za online prodaju karata(bioskop,pozorište,koncert)

3. Opis SOPK

Sistem omogućava korisnicima da rezervišu karte u pozorištima, bioskopima I koncertima putem interneta i da na taj način izbegnu redove u istim kao I da unapred znaju ima li karata ili ne. Pri ulasku na sistem, od korisnika će biti zatraženo da se uloguje ukoliko već ima nalog, a ukoliko nema, biće mu ponuđena mogućnost da se registruje. Nakon uspešnog procesa logovanja/registracije korisniku će biti prikazane tri opcije: prikaz rezervacija, prikaz korisnika I prikaz objekata. Prikaz rezervacija prikazuje sve dosadašnje rezervacije, prikaz korisnika prikazuje sve registrovane korisnike u sistemu

kao I mogućnost izmene,brisanja I dodavanja istih, dok prikaz objekata prikazuje sve objekte iz baze podataka sa svojim podacima I opcijom da se filtriraju u svrhu lakše pretrage dok klikom na objekat korisnik ima opciju rezervacije gde ga sistem vodi u novi pop-up window koji u vidu forme traži da se popune detalji vezani za rezervaciju I konačno, realizovanje iste. Nakon toga, korisnik može da se odjavi i izađe iz aplikacije.

4. Funkcionalni zahtevi

U delu funkcionalnih zahteva biće prikazani svi funkcionalni zahtevi koje aplikacija treba da zadovolji. Za izradu i prikaz funkcionalnih zahteva se koristi Requirements model koji je generisan u PowerDesigner alatu.

	Title ID	Full Description	Code	Priority	Workload	Risk	Status
→	⊟ 1.	**Zahtevi korisnika**	REQ_0021	Undefined	0	Undefined	Draft
2	1.1	**Registracija korisnika** Korisnik ima mogucnost registracije na sistem ukoliko mu prvi put pristupa.	REQ_0002	Undefined		Undefined	Draft
3	1.2	**Login korisnika** Korisnici koji su vec prosli proces registracije loguju se na sistem sa korisnickim imenom i sifrom.	REQ_0003	Undefined		Undefined	Draft
4	⊟ 1.3	**Odabir iz liste** Korisnik unutar aplikacije bira podatke.	REQ_0004	Undefined	0	Undefined	Draft
5	1.3.1	**Tip objekta** Korisnik bira dogadjaj: koncert, pozoriste, bioskop	REQ_0005	Undefined		Undefined	Draft
6	1.3.2	**Lokacija** Korisnik ima opciju da izabere konkertnu lokaciju/adresu odabranog objekta.	REQ_0006	Undefined		Undefined	Draft
7	1.3.3	**Sediste** Korisnik bira sediste ukoliko je slobodno.	REQ_0007	Undefined		Undefined	Draft
8	1.3.4	**Vreme** Korisnik bira vreme dolaska na zeljeni dogadjaj.	REQ_0016	Undefined		Undefined	Draft
9	1.3.5	**Metod placanja** Korisnik unosi metod placanja: paypal, kartica ili kes.	REQ_0022	Undefined		Undefined	Draft
10	1.4	**Rezervacija** Korisnik finalno sumira podatke i realizuje rezervaciju.	REQ_0009	Undefined		Undefined	Draft
11	⊟ 2.	**Zaposleni**	REQ_0011	Undefined	0	Undefined	Draft
12	2.1	**Pracenje rezervacija** Zaposleno lice ima evidenciju i pristup trenutno rezervisanim mestima u datom lokalu.	REQ_0012	Undefined		Undefined	Draft

13	⊟ 3.	**Admin**	REQ_0017	Undefined	0	Undefined	Draft
14	3.1	**Pregled rezervacija** Admin ima uvid u listu rezervacija(Korisnik, institucija, broj sedista,datum i vreme rezervacije)	REQ_0018	Undefined		Undefined	Draft
15	3.2	**Upravljanje podacima** Admin moze da brise, menja ili dodaje podatke vezane za rad i rezervacije u okviru pomenutih institucija.	REQ_0019	Undefined		Undefined	Draft
16	3.3	**Pregled liste korisnika** Admin ima uvid u listu korisnika i njegove podatke koji su dati u okviru registracije.	REQ_0020	Undefined		Undefined	Draft
17	3.4	**Evidencija objekata** Admin vodi evidenciju pomenutih institucija/objekata.	REQ_0023	Undefined		Undefined	Draft
18	⊟ 3.5	**Prikaz objekata** Admin ima uvid u listu postojecih objekata.	REQ_0024	Undefined	0	Undefined	Draft
19	⊟ 3.5.1	**Filtriranje objekata** Admin ima mogucnost filtriranja objekata po razlicitim kriterijumima	REQ_0025	Undefined	0	Undefined	Draft
20	3.5.1.	**Filtriranje po adresi** Admin filtrira institucije po odgovarajucim adresama istih.	REQ_0026	Undefined		Undefined	Draft
→	3.5.1.	**Filtriranje po tipu objekta** Admin filtrira institucije u zavisnosti od njihovog tipa: bioskop, pozoriste, koncertna hala.	REQ_0027	Undefined		Undefined	Draft

Funkcionalni zahtevi:

- **(1) Zahtevi korisnika**

- **(1.1) Registracija korisnika**

Korisnik ima mogućnost registracije na sistem ukoliko mu prvi put pristupa.

- **(1.2) Login korisnika**

Korisnici koji su već prosli proces registracije loguju se na sistem sa korisničkim imenom i šifrom.

- **(1.3) Odabir iz liste**

Korisnik unutar aplikacije bira podatke.

- **(1.3.1) Tip objekta**

Korisnik ima opciju da izabere koncertnu halu, bioskop ili pozorište.

- **(1.3.2) Lokacija**

Korisnik ima opciju da izabere konkertnu lokaciju/adresu odabranog objekta.

- **(1.3.3) Sedište**

Korisnik bira sedište ukoliko je slobodno.

- **(1.3.4) Vreme**

Korisnik bira vreme dolaska na željeni događaj.

- **(1.3.5) Metod plaćanja**

Korisnik unosi metod plaćanja: paypal, kartica ili keš.

- **(1.4) Rezervacija**

Korisnik finalno sumira podatke i realizuje rezervaciju.

- **(2) Zaposleni**
- **(2.1) Praćenje rezervacija**

Zaposleno lice ima evidenciju i pristup trenutno rezervisanim mestima u datom lokalu.

- **(3) Admin**
- **(3.1) Pregled rezervacije**

Admin ima uvid u listu rezervacija(Korisnik, objekat, broj sedišta,datum i vreme rezervacije)

- **(3.2) Upravljanje podacima**

Admin moze da briše, menja ili dodaje podatke vezane za rad i rezervacije u okviru pomenutih objekata.

- **(3.3) Pregled liste korisnika**

Admin ima uvid u listu korisnika i njegove podatke koji su dati u okviru registracije.

- **(3.4) Evidencija objekata**

Admin vodi evidenciju pomenutih objekata.

- **(3.5) Prikaz objekata**

Admin ima uvid u listu postojećih objekata.

- **(3.5.1) Filtriranje objekata**

Admin ima mogucnost filtriranja objekata po različitim kriterijumima

- **(3.5.1.1) Filtriranje po adresi**

Admin filtrira objekte po odgovarajućim adresama istih.

- **(3.5.1.2) Filtriranje po tipu objekta**

Admin filtrira objekte u zavisnosti od njihovog tipa: bioskop, pozorište, koncertna hala.

5. Nefunkcionalni zahtevi

Nefunkcionalni zahtevi su zahtevi koji su vezani za celi sistem. Pod nefunkcionalnim zahtevima spadaju bezbednost sistema, lakoća upotrebe, raspoloživost i brzina.

	Title ID	Full Description	Code	Priority	Workload	Risk	Status
→	1.	**Raspolozivost** Aplikacija treba da bude raspoloziva korisniku 90% vremena na mesecnom nivou.	REQ_0004	Undefined		Undefined	Draft
2	2.	**Bezbednost** Licni podaci korisnika moraju biti zasticeni u svakom trenutku od neautorizovanog pristupa.	REQ_0002	Undefined		Undefined	Draft
3	3.	**Lakoca upotrebe** Korisnicki interfejs mora biti jednostavan za koriscenje i lak za razumevanje i rukovanje korisnicima razlicitih starosnih dobi, i razlicitih razumevanja IT sistema.	REQ_0003	Undefined		Undefined	Draft
4	4.	**Brzina** Sistem treba da omoguci blagovremenost i brzinu odaziva za razlicite zadatke koje treba da izvrsi, konkretno, rezervaciju sedista za dogadjaje u pomenutim institucijama.	REQ_0005	Undefined		Undefined	Draft

Nefunkcionalni zahtevi

- **Raspoloživost**

  Aplikacija treba da bude raspoloživa korisniku 90% vremena na mesečnom nivou.

- **Bezbednost**

Lični podaci korisnika moraju biti zastićeni u svakom trenutku od neautorizovanog pristupa.

- **Lakoća upotrebe**

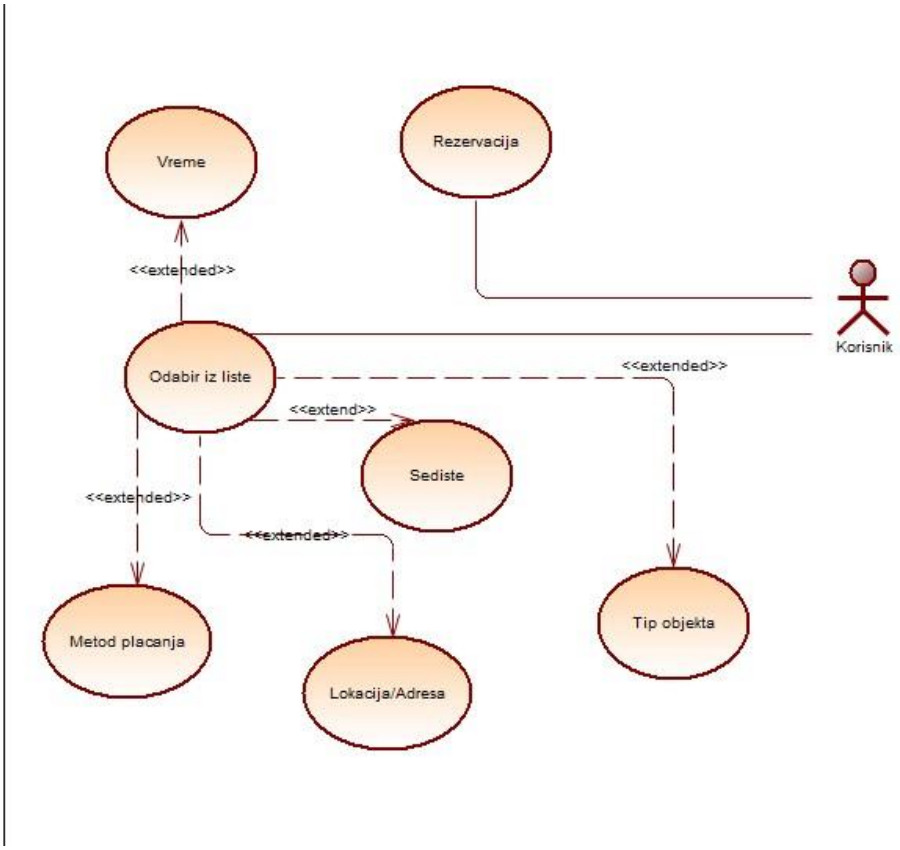
Korisnički interfejs mora biti jednostavan za korišćenje i lak za razumevanje i rukovanje korisnicima različitih starosnih dobi, i različitih razumevanja IT sistema.

- **Brzina**

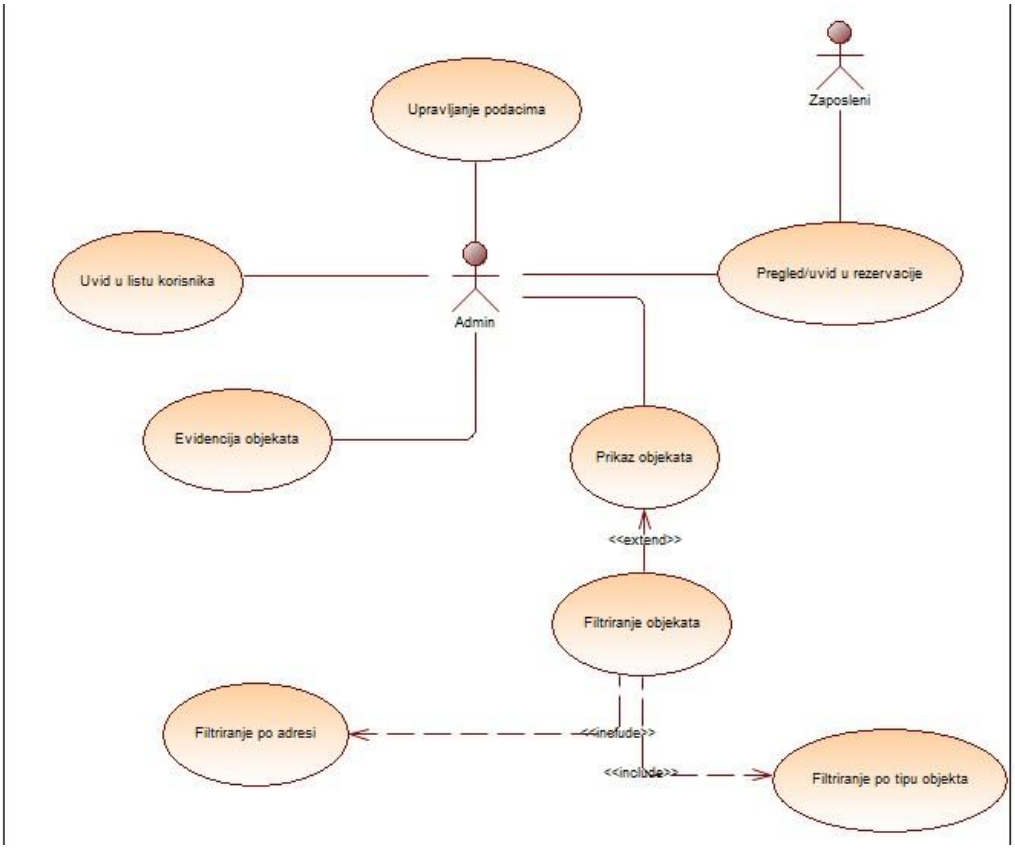
Sistem treba da omogući blagovremenost i brzinu odaziva za različite zadatke koje treba da izvrši, konkretno, rezervaciju sedišta za događaje u pomenutim institucijama.

6. Use Case Dijagram

Slučajevi korišćenja predstavljaju operacije koje akteri mogu da izvrše sa funkcionalnostima softvera. Slučajevi korišćenja su prikazani u UseCase dijagramu sa PowerDesigner alatom. Na slici ispod je prikazan osnovni, odnosno opšti slučaj korišćenja za ovaj sistem.



Slučajevi korišćenja za korisnika



Use Case dijagram za admina i zaposlenog

Primarni scenario korisnika:

- Korisnik moze da rezerviše određeni objekat
- Korisnik može da pretražuje objekte na osnovu filtera(po broju sedišta, tipu objekta, gradu)
- Korisnik unosi podatke vezane za rezervaciju(vreme,broj sedišta,adresa,metod plaćanja)

Primarni scenario administratora:

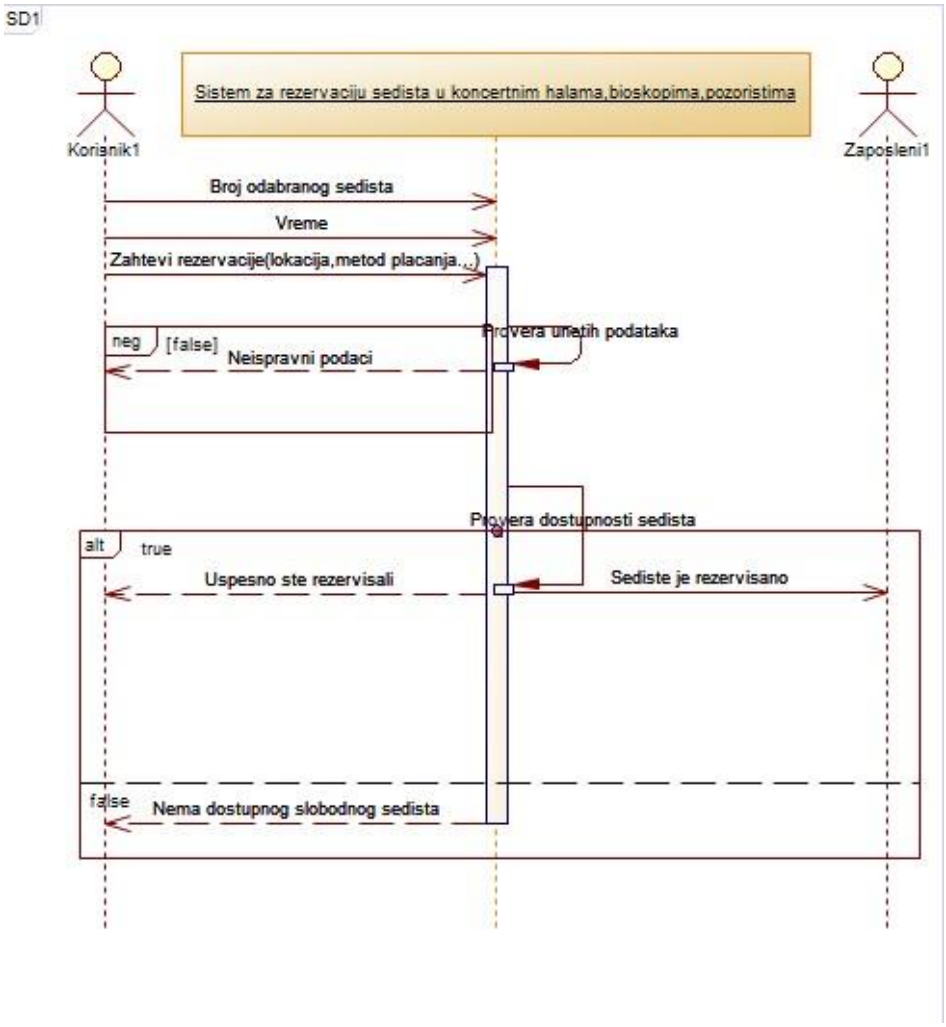
- Admin ima uvid u listu rezervacija(Korisnik, objekat, broj sedišta, datum i vreme dolaska)

- Admin može da briše, menja i dodaje podatke
- Admin ima uvid u listu korisnika i njegove podatke

Primarni scenario zaposlenog:

- Zaposleni ima uvid u listu rezervacija(Korisnik, objekat, broj sedišta, datum i vreme dolaska)

Sekundarni scenario predstavlja greške koje se mogu desiti u realizaciji osnovnog scenarija interakcije. Sekundarne scenarije treba definisati da bi se kasnije lakše razrešile problematične situacije.



Sekundarni scenario

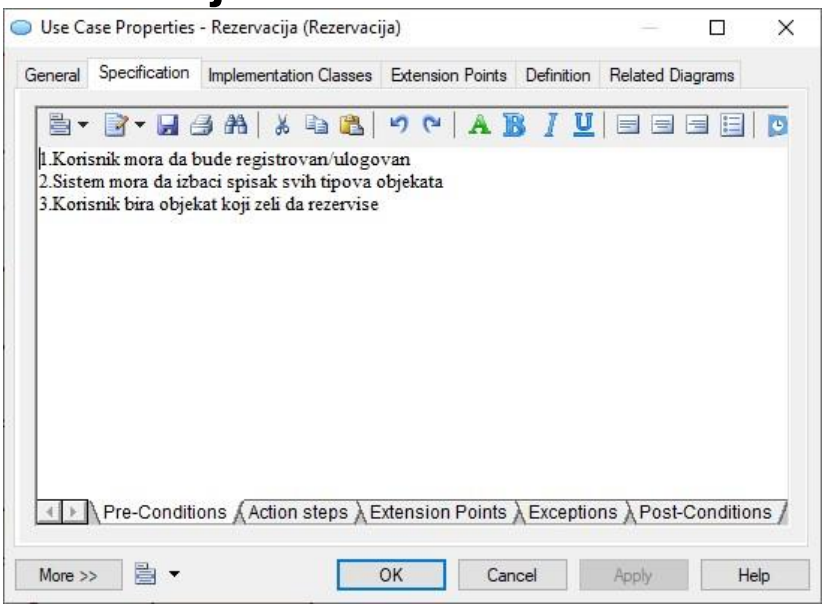
Sekundarni scenario:

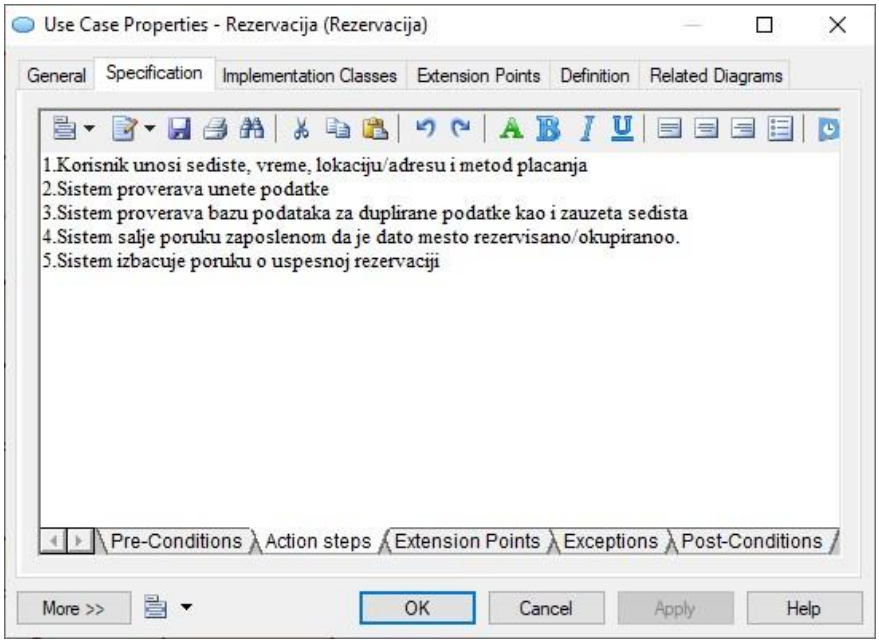
- Korisnik unosi broj sedišta i vreme dolaska, kao i metod plaćanja i adresu objekta
- Sistem proverava ispravnost podataka koje je korisnik uneo
- Ukoliko su podaci neispravni korisnik dobija poruku o neispravnosti podataka

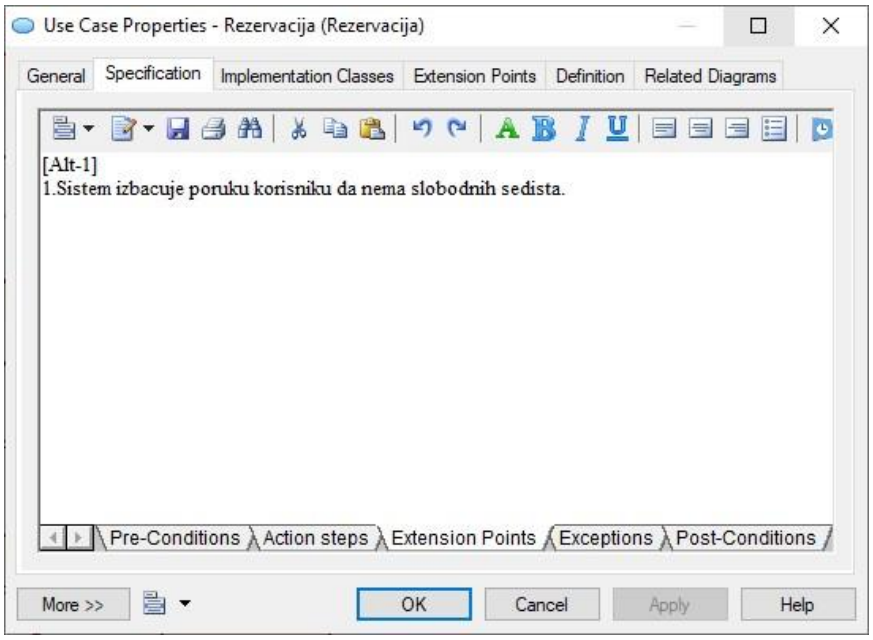
- Sistem proverava da li u bazi ima slobodnih sedišta za odabran datum i vreme
- Ukoliko u bazi nema slobodnih sedišta korisnik dobija poruku o tome
- U slučaju da ima, korisnik dobija poruku da je uspešno rezervisao sedište

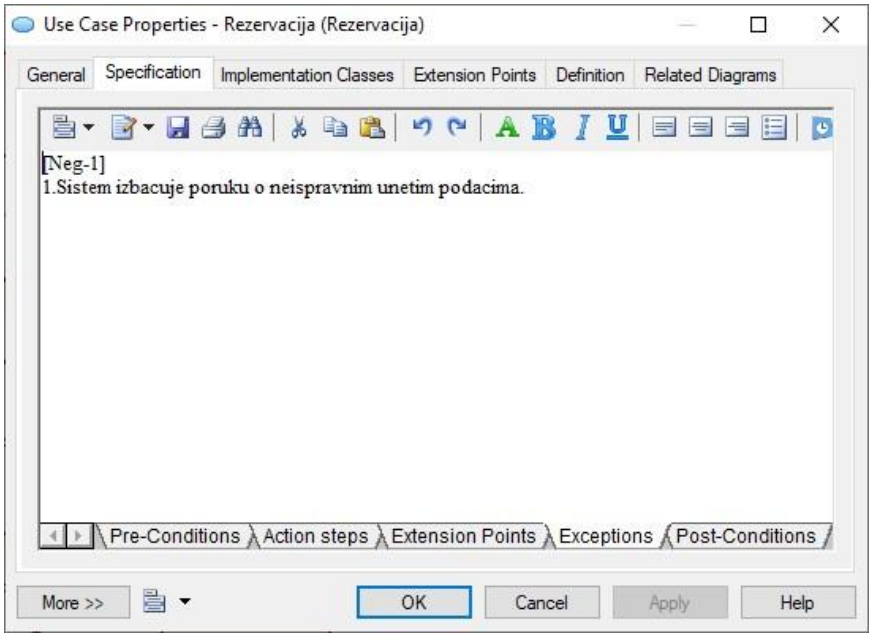
Scenariji(tabele koraka)

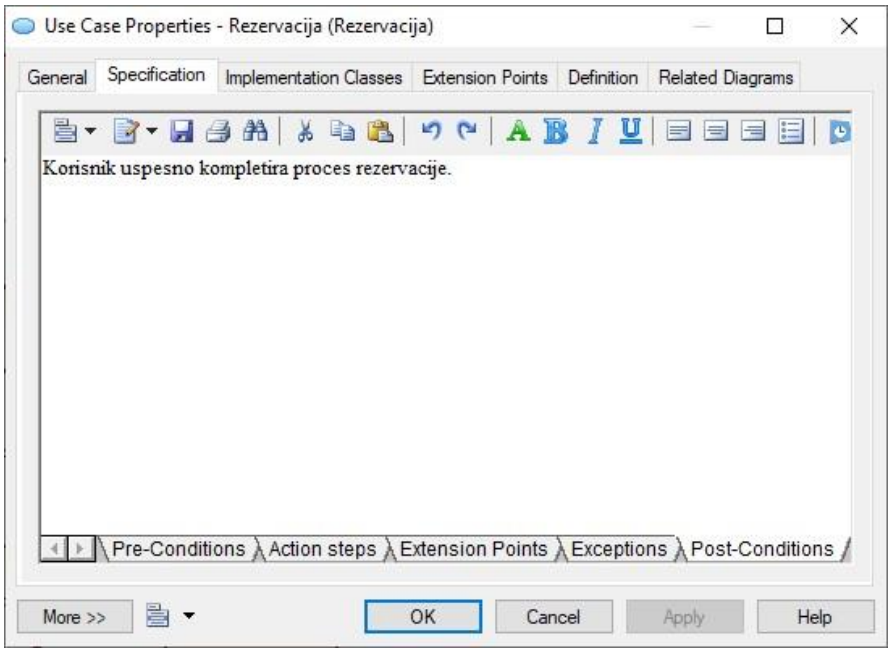
Rezervacija






Use Case Properties - Rezervacija (Rezervacija)
General
Specification
Implementation Classes
Extension Points
Definition
Related Diagrams
[Alt-1]
1.Sistem izbacuje poruku korisniku da nema slobodnih sedista.
Pre-Conditions
Action steps
Extension Points
Exceptions
Post-Conditions
More >>
OK
Cancel
Apply
Help

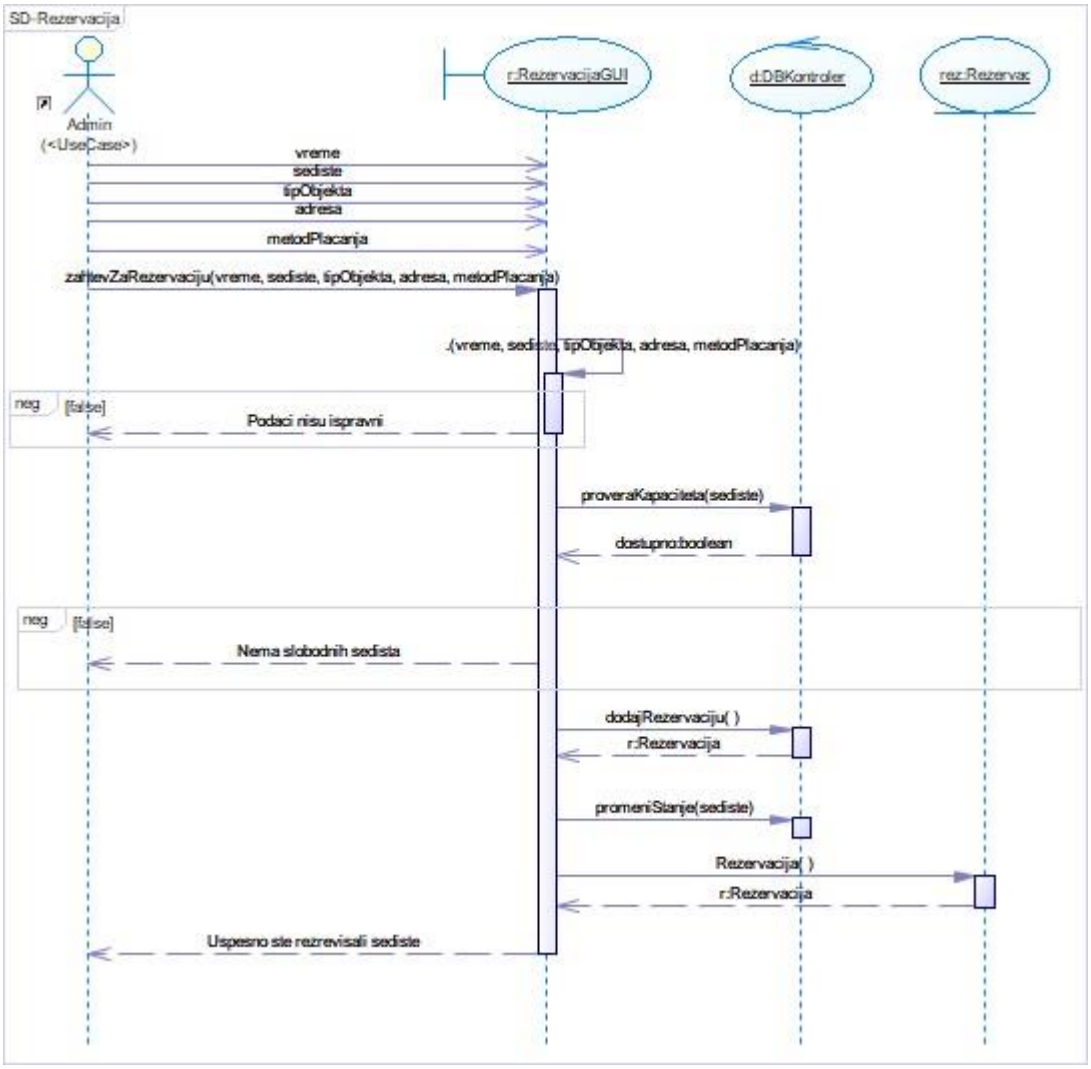

Use Case Properties - Rezervacija (Rezervacija)
General
Specification
Implementation Classes
Extension Points
Definition
Related Diagrams
[Neg-1]
1.Sistem izbacuje poruku o neispravnim unetim podacima.
Pre-Conditions
Action steps
Extension Points
Exceptions
Post-Conditions
More >>
OK
Cancel
Apply
Help



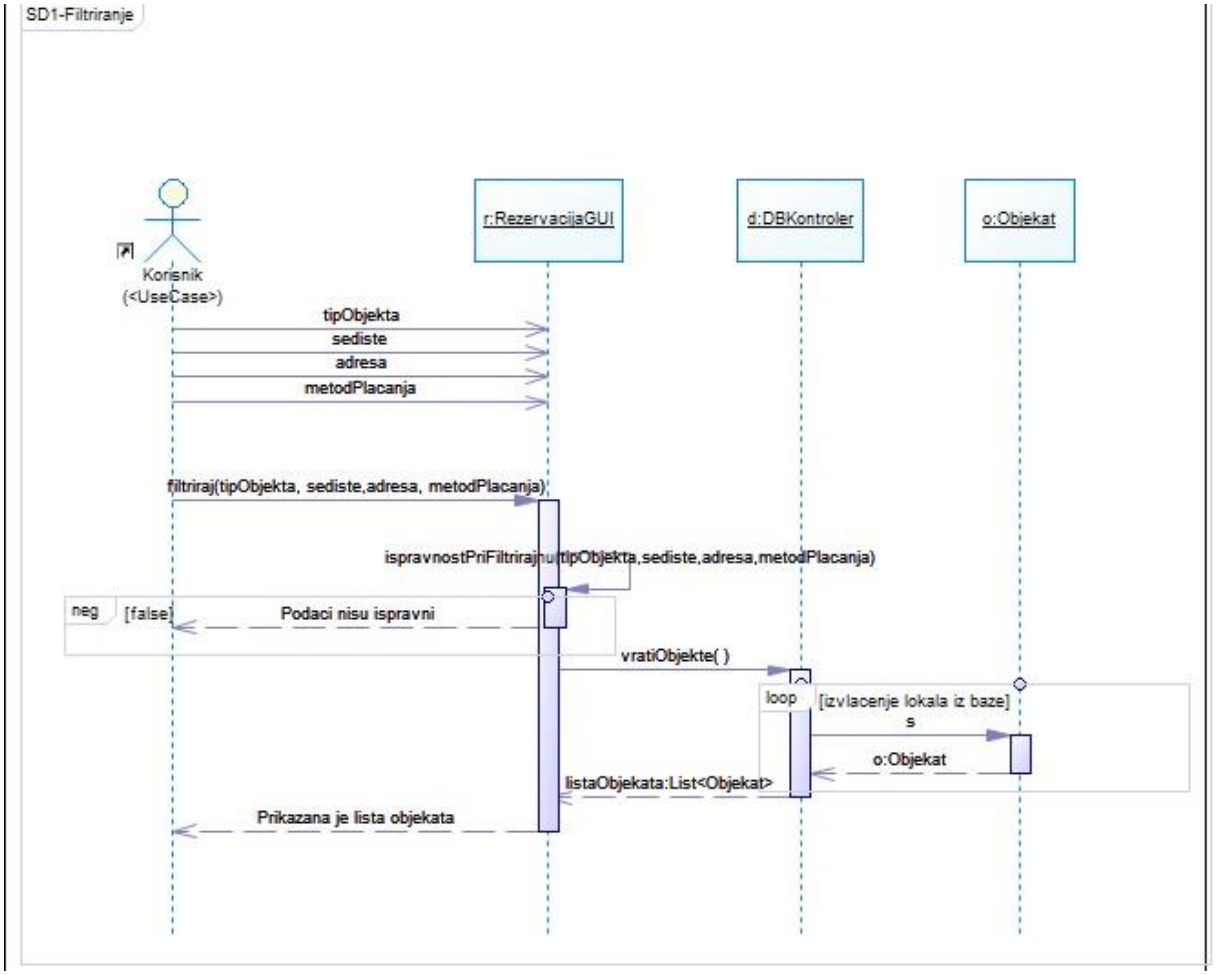
7. Sekvencijalni dijagram

Najšeći oblik dijagrama interakcije koji se koristi u praksi jeste dijagram sekvenci (eng. Sequence Diagram). Ovaj dijagram obično prikazuje jedan scenario koji obuhvata izvestan broj objekata i poruka koje oni razmenjuju u okviru slučaja upotrebe. Korišćenjem sekvencijalnog dijagrama može se opisati koje interakcije se izvršavaju kada se pojedinačni slučaj upotrebe izvršava i u po kom redosledu se ove interakcije izvršavaju.
Sekvencijalni dijagram – prikazujemo sistem kao objekat i aktere koji razmenjuju poruke sa sistemom.
Detaljni sekvencijalni dijagram - prikazuje konkretne klase iz klasnog dijagrama, a poruke u njemu odgovaraju metodama klasnog dijagrama.

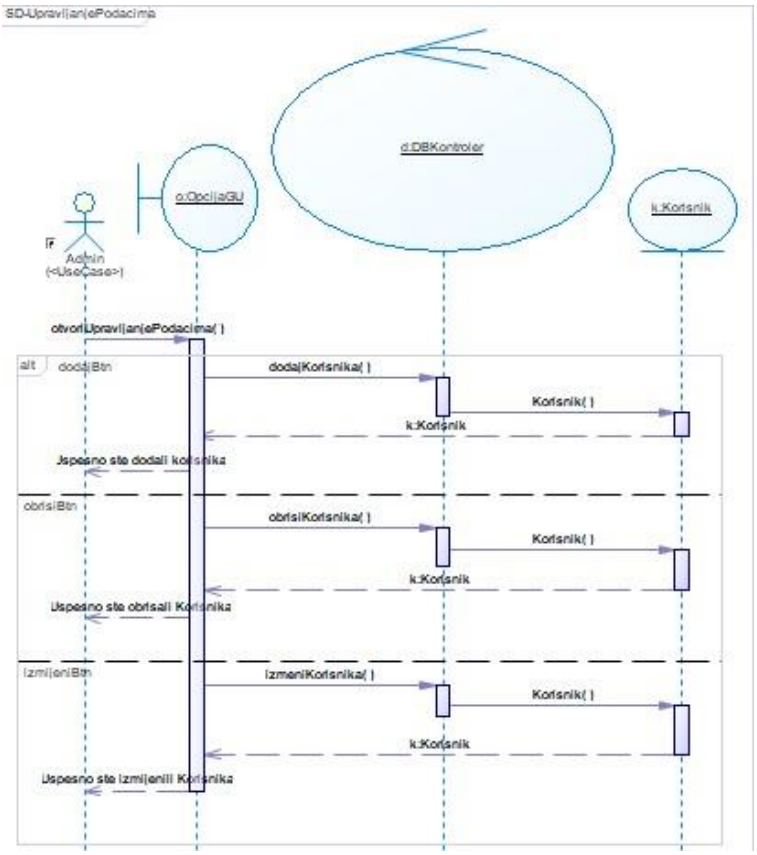
SD-Rezervacija(slika ispod)


SD-Rezervacija
Admin
(<UseCase>)
r:RezervacijaGUI
d:DBKontroler
rez:Rezervac
vreme
sediste
tipObjekta
adresa
metodPlacanja
zahtevZaRezervaciju(vreme, sediste, tipObjekta, adresa, metodPlacanja)
(vreme, sediste, tipObjekta, adresa, metodPlacanja)
neg
[false]
Podaci nisu ispravni
proveraKapaciteta(sediste)
dostupno:boolean
neg
[false]
Nema slobodnih sedista
dodajRezervaciju( )
r:Rezervacija
promeniStanje(sediste)
Rezervacija( )
r:Rezervacija
Uspesno ste rezrevisali sediste

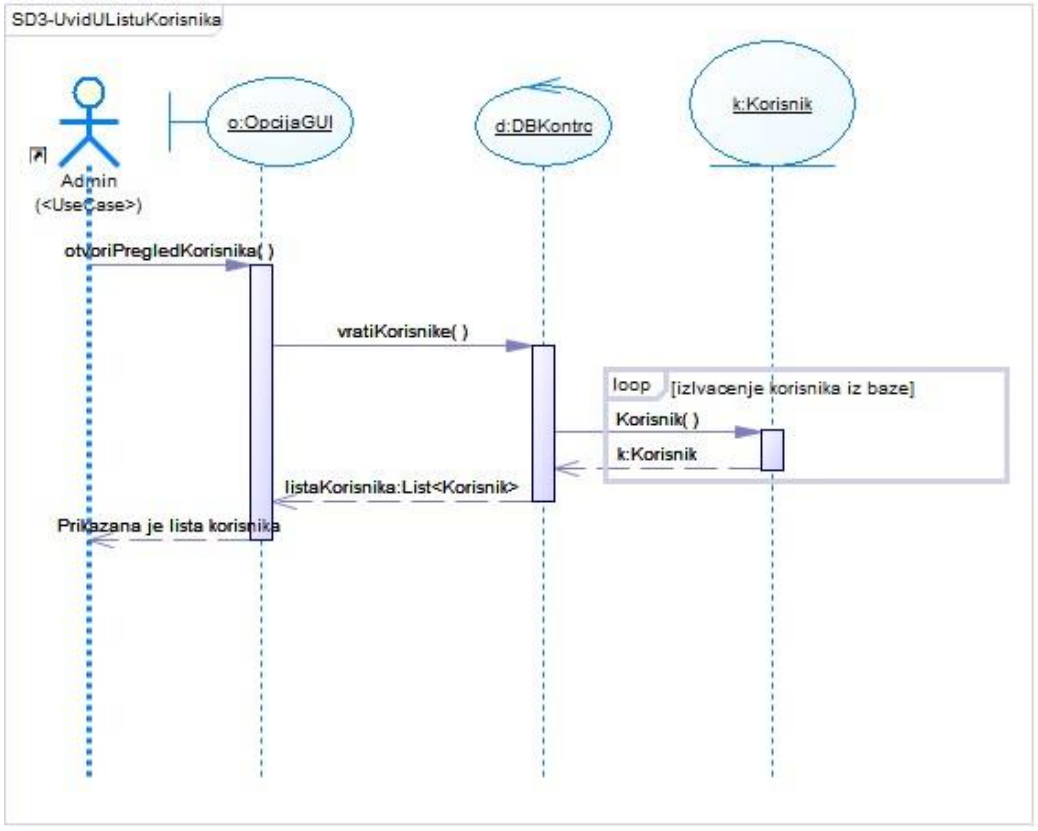
SD – Filtriranje

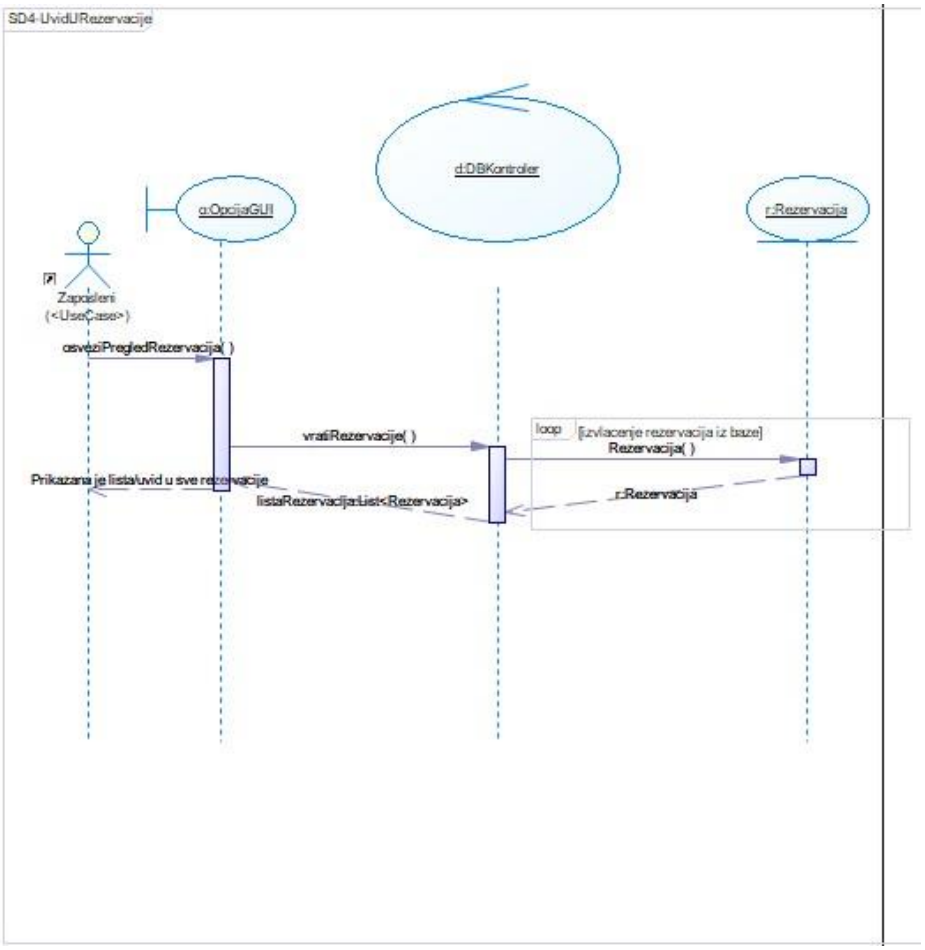


SD-Upravljanje Podacima



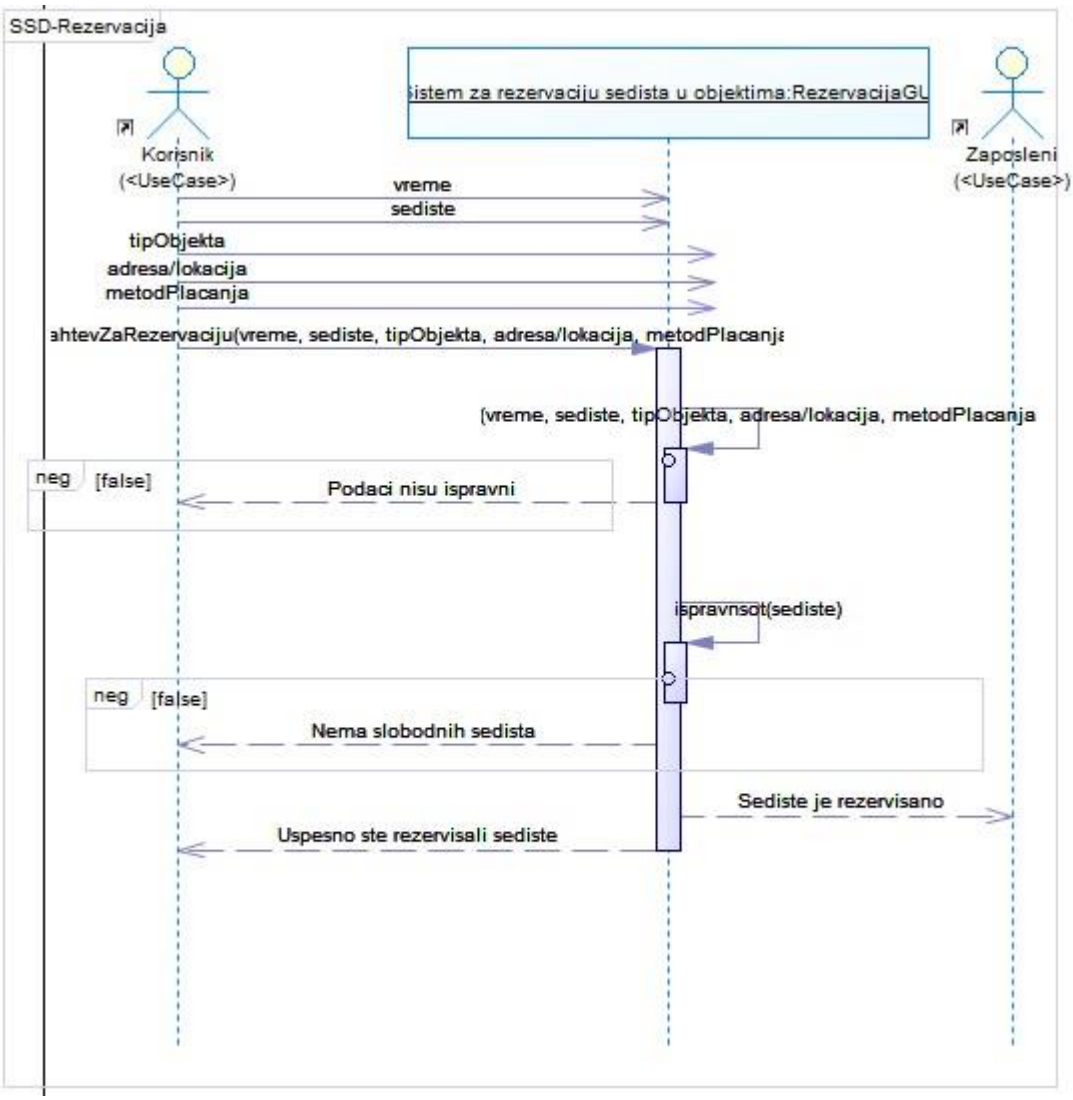
SD-Uvid u Listu Korisnika





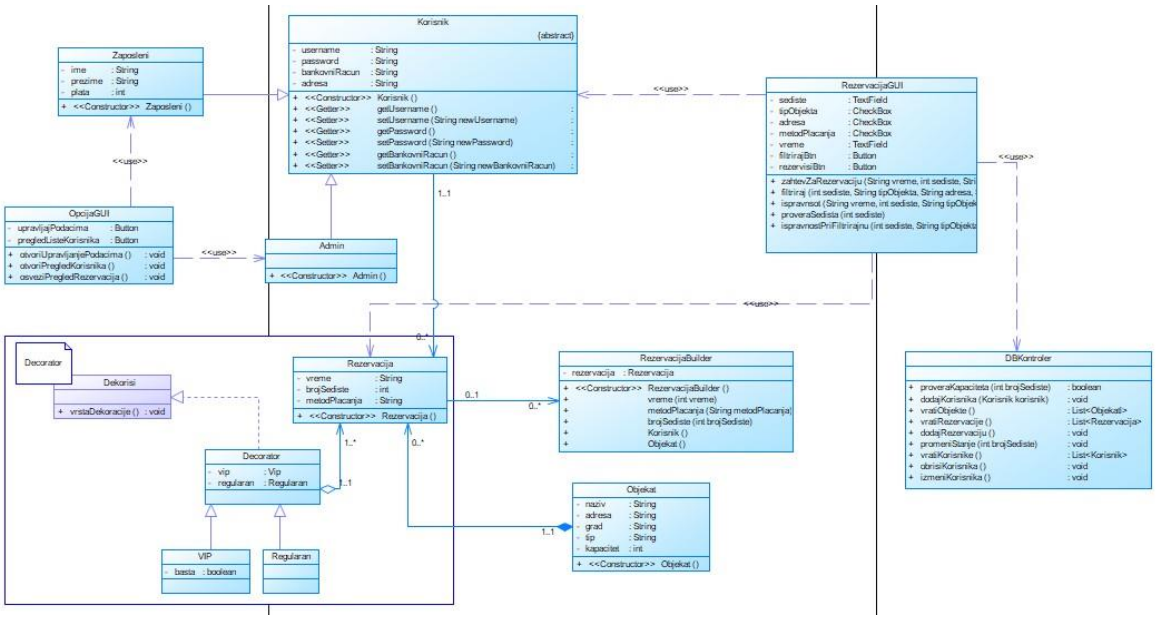
SD- Uvid u Rezervacije

SD-Rezervacija(Korisnik perspektiva)



8. Klasni dijagram

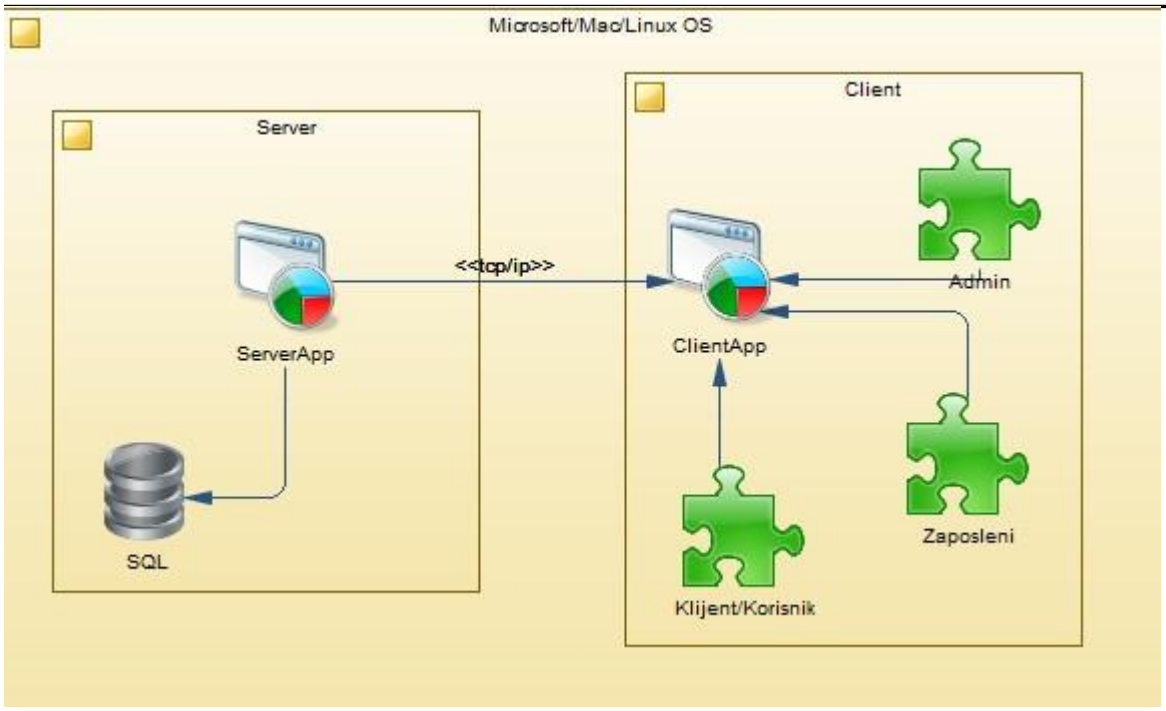
Klasni dijagram je statički strukturni UML dijagram koji opisuje klase sistema, njihove atribute i operacije (metode), kao i relacije između objekata. Još jednom koristimo PowerDesigner



alat.

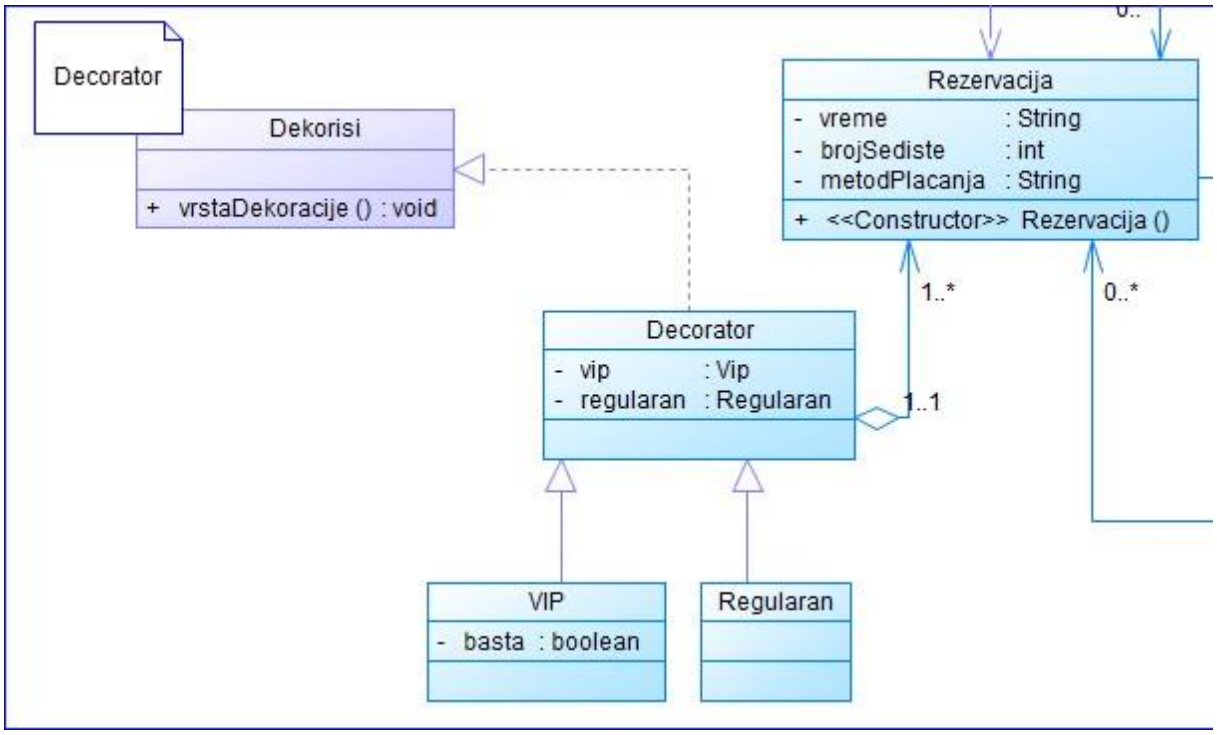
9. Arhitektura Sistema

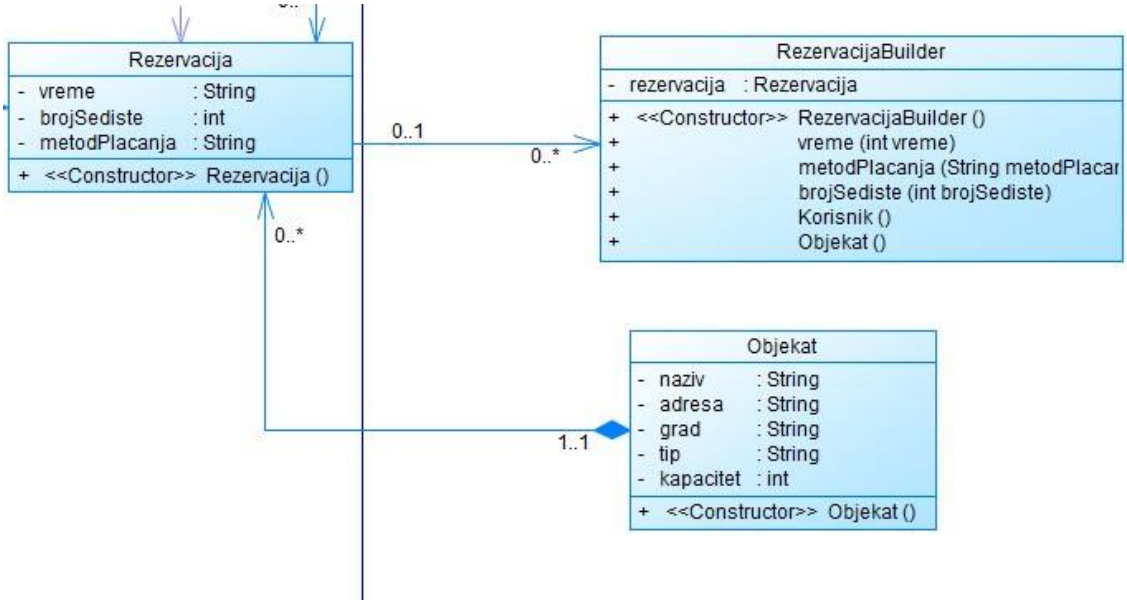
Na slici ispod je prikazan dijagram arhotekture za online prodaju karata za bioskope, pozorište i koncerte . Klijent, zaposleni i admin preko iste forme odnosno aplikacije pristupaju sistemu. U zavisnosti od uloge, korisnici ovog sistema mogu izvršavati određene akcije. Svi podaci se uzimaju iz baze podataka ili unose u bazu preko sistema.



10. Design patterns – šabloni projektovanja

Decorator





Builder pattern

11. Testiranje

Testiranje nam služi da vidimo da li program radi kako treba I da li se u određenim situacijama ponaša kako treba. Testiranje služi I da na vreme utvrdimo greške, I eventualno ih ispravimo.

U ovom slučaju koristimo jedinično testiranje – testiranje pojedinačnih objekata. Dakle, imamo test koji testira odgovarajuću klasu, odnosno metode koje nešto rade u okviru te klase.

Za testiranje će se koristiti JUnit. To je framework (skup alata) pomoću kojih možemo da vršimo testiranje.

```
public boolean raspolozivoStanje(Objekat o, int brojSedista) throws ClassNotFoundException, SQLException {

    DBKontroler dbKontr = new DBKontroler();
    if (brojSedista <= (o.getKapacitet() - dbKontr.ukupnoRezervisano(o)) && brojSedista > 0) {
        System.out.println((o.getKapacitet() - dbKontr.ukupnoRezervisano(o)));
        return true;
    } else {
        return false;
    }
}
```

Metoda raspolozivoStanje() ima zadatak da proveri da li objekat ima na raspolaganju traženi broj sedišta. Moguće su 2 greške: da korisnik unese pogrešan podatak I da objekat nema dovoljno slobodnih mesta što rešavaju 2 izuetka koje smo napravili.

```
public class GreskaUUnosu extends Exception {

    public GreskaUUnosu(String message) {
        super(message);
    }
}
```

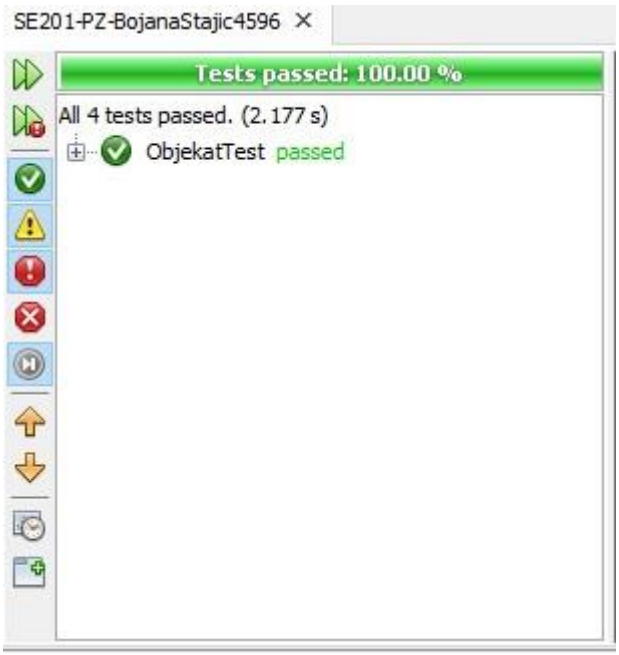
```
public class NemaSlobodnihMesta extends Exception {

    public NemaSlobodnihMesta(String message) {
        super(message);
    }
}
```

U JUnit-u pravimo četiri testa:

1. test1() - u ovom testu se očekuje da objekat na raspolaganju ima traženi broj sedišta
2. test2 () - u ovom testu se očekuje da dođe do izuzetka(NemaSlobodnihMesta) jer objekat na raspolaganju nema traženi broj sedišta
3. test3 () - u ovom testu se očekuje da dođe do izuzetka(GreskaUUnosu) jer je korisnik upisao pogrešne podatke(-150)
4. test4 () - u ovom testu se očekuje da dođe do izuzetka(GreskaUUnosu) jer je korisnik upisao pogrešne podatke(0)

```
@Test
public void test1() throws ClassNotFoundException, SQLException, NemaSlobodnihMesta, GreskaUUnosu {
    Objekat objekat = new Objekat(1, "Atelje 212", "Svetogorksa 21", "Beograd", "pozoriste", 150);
    DBKontroler db = new DBKontroler();

    boolean exp = true;
    boolean result = db.raspolozivoStanje(objekat, 50);
    assertEquals(exp, result);
}

@Test
public void test2() throws ClassNotFoundException, SQLException, NemaSlobodnihMesta, GreskaUUnosu {
    Objekat objekat = new Objekat(1, "Atelje 212", "Svetogorksa 21", "Beograd", "pozoriste", 150);
    DBKontroler db = new DBKontroler();

    boolean exp = false;
    boolean result = db.raspolozivoStanje(objekat, 120);
    assertEquals(exp, result);
}
```

```
@Test
public void test3() throws ClassNotFoundException, SQLException, NemaSlobodnihMesta, GreskaUUnosu {
    Objekat objekat2 = new Objekat(2, "O2 Arena", "12th Street", "London", "koncertna hala", 150);
    DBKontroler db = new DBKontroler();

    boolean exp = false;
    boolean result = db.raspolozivoStanje(objekat2, -150);
    assertEquals(exp, result);
}

@Test
public void test4() throws ClassNotFoundException, SQLException, NemaSlobodnihMesta, GreskaUUnosu {
    Objekat objekat2 = new Objekat(2, "O2 Arena", "12th Street", "London", "koncertna hala", 150);
    DBKontroler db = new DBKontroler();

    boolean exp = false;
    boolean result = db.raspolozivoStanje(objekat2, 0);
    assertEquals(exp, result);
}
```



Test je uspešno obavljen.

12. Zaključak

Na samom kraju, u procesu realizacije ovog projektnog zadatka, naučila sam dosta o softverskim procesima prilikom kreiranja jednog funkcionalnog sistema, unapredivši prethodno znanje. Implementiran je jedan složen I funkcionalan sistem. Naučila sam I ozbiljniji pristup kodiranju I implementaciji kao I značaj planiranja I postupnog rešavanja određenij segmenata. Jako puno sam naučila i u vezi PowerDesigner alata. Nadam se da ću u budućnosti moći da razvijem još naprednije I bolje sisteme.

13. Literatura

1. Ian Sommervile, Software Engineering, Tenth Edition, Pearson Education Inc., 2016.
2. Partha Kuchana, Software Architecture Design patterns in Java