*Jesenji semestar, 2016/17*

*PREDMET: SE201 – Uvod u softversko inženjerstvo*

# Projektni zadatak na temu

Sistem za rezervaciju i narucivanje hrane i pica u restoranima

Profesor: Ime i prezime:

##### Prof. Dragan Domazet Mihajlo Rajkovic

Asistent: Broj indeksa:

**Bogdan Jankovic 5088**

# Sadrzaj

1. Uvod.....................................................................................................
2. Opis i svrha dokumenta........................................................................
3. Funkcije softvera...................................................................................
   1. Funkcije....................................................................................
   2. Korisnici sistema.......................................................................
      1. Opis za klijenta................................................................
      2. Opis za zaposlenog.........................................................
      3. Opis za administratora.....................................................
4. Specifikacija zahteva............................................................................
   1. Funkcionalni..............................................................................
   2. Nefunkcionalni..........................................................................
5. Slucajevi koriscenja..............................................................................
   1. UseCase administratora...........................................................
   2. UseCase zaposlenog................................................................
   3. UseCase klijenta.......................................................................
6. Sistem sekvencioni dijagrami...............................................................
   1. SSD administratora..................................................................
   2. SSD zaposlenog.......................................................................
   3. SSD klijenta..............................................................................
7. Klasni dijagram.....................................................................................
8. Arhitektura............................................................................................
9. Testiranje..............................................................................................
10. Zakljucak............................................................................................
11. Reference...........................................................................................

# Uvod

Cilj dokumenta je da detaljno predstavi novi rezervacije mesta i hrane u kaficima i restoranima. Ovaj projektni zadatak baviće se definisanjem osnovnih funkcionalnih i nefunkcionalnih zahteva koje softver mora da zadovolji kako bi se smatrao uspešnim. U dokumentu će biti detaljno opisan rad sistema. Za bolje razumevanje problema sa kojim se susrećemo kako bi projekat bio realizovan biće objašnjeni putem korišćenja dijagrama(Klasni, UseCase, sekvencijalni) koji bi trebalo detaljnije da objasne način i zamisao funkcionisanja sistema. Za potrebe izrade dijagrama koristiće se alata **PowerDesigner.** U projektnom zadatku biće predstavljene i mogućnosti svih korisnika sistema. Korisnici ovog sistema su klijenti, administratori i zaposljeni.

# Opis i svrha dokumenta

Svrha dokumenta je da jasno opiše moguće funkcionalnosti i mogućnosti softvera. Dokument se sastoji od samog uvoda gde se daje kratak opis cilja za kreiranje ovog sistema. Nakon toga, dokument će se sastojati od opisa sistema sa identifikovanim klasama korisnika i njihovim ovlašćenjima, detalje zahteva u kojima su opisani funkcionalni i nefunkcionalni zahtevi. Dokument će se sastojati i od specifikacije dizajna u okviru kog se nalaze dijagrami slučajeva korišćenja, klasni dijagram i svih mogućih sekvencijalnih dijagrama koji su vezani za rezervaciju i narucivanje hrane u restoranima.

Takođe, jako bitna stavka je razvoj test klasa u java programskom jeziku u kojem ćemo simulirati nekoliko funkcija od kojih će se sistem za praćenje prisustva sastojati.

3. Funkcije softvera

3.1. Funkcije softvera

Funkcije sistema su:

* Autentifikacija korisnika
* Rezervacija stolova
* Narucivanje hrane i pice unapred
* Registrovanje korisnika
* Placanja online putem
* Kreiranje novih korisnika
* Menjanje rezervacija
* Pregled istorije rezervacija
* Pregled cena
* Pregled slobodnih mesta

## Korisnici sistema

Postoje tri vrste korisnika sistema za praćenje prisustva studenata:

* + 1. Klijent
    2. Zaposleni
    3. Administrator

3.2.1. Opis za klijenta

Sistem je zamišljen da funkcioniše uz pomoć aplikacije kojom ce se birati restorani i detalji rezervacije. Klijent ce imati mogucnost dodavanja kartice putem koje moze placati online. Klijenti, zaposleni i administratori će moći da koriste softver, s tim što neće svako imati iste mogućnosti, gledano hijerarhijski , administrator će imati najviša prava, zatim zaposlen, pa klijent sa najmanjim pravima. Svaki klijent ce imati mogucnost skidanja aplikacije iz prodavnice aplikacija.

Svaki klijent će moći da pristupi aplikaciji sistema svojim nalogom koji kreira prilikom prvog paljenja alikacije, na kojoj mogu da vide sve restorane, da izvrše rezervaciju prema datumu i mogucnosti slobodnih mesta ili da pogledaju svoje prethodne rezervacije.

3.2.2. Opis za zaposlenog

Korisnik sa autoritetom zaposlenog ima veće mogućnosti od klijenta. Sistem zaposlenom pruža mogućnost da za njegov lokal azurira broj slobodnih mesta, obrise, doda ili izmeni rezervaciju. Zaposleni putem aplikacije ima uvid u sve trenutno prisutne klijente kao i detalje njihovih rezervacija. Može doći do problema da ukoliko klijent nije u mogućnosti da pokupi svoju rezervaciju, gde ce zaposleni moci da tu rezervaciju ponisti ili je uruci nekom drugom.

3.2.3. Opis za administratora

Korisnik sa autoritetom administrator ima najveće mogućnosti koje naš sistem pruža. Administrator je korisnik koji može da upravlja svim nalozima, čak i administratorskim nalazima i svim drugim funkcijama koje sistem pruža. Administrator može dodati novog klijenta, novi lokal, promeniti podatke nekog klijenta i obrisati klijenta, kao i lokala. Takođe, administrator može upravljati i vršiti promene u bazi podataka i upravljati serverom na kome se nalazi ceo sistem. Administrator takođe može da pruža pomoć svim ostalim klijentima ukoliko dođe do neke greške u sistemu ili prilikom registrovanja prisustva nekog studenata.

Administrator je takođe zadužen za oglašavanje datuma održavanja sistema, kao i da obaveštava o bitnim informacijama vezanim za sistem sve ostale korisnike.

# Specifikacija zahteva

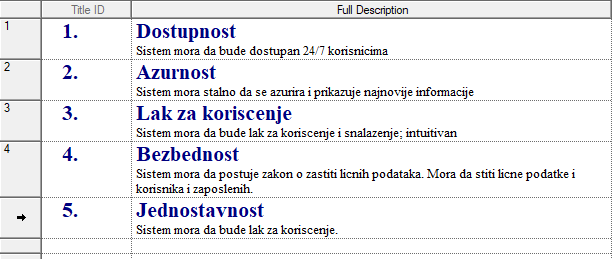
## 4.1. Funkcionalni zahtevi

U delu funkcionalnih zahteva biće prikazani svi funkcionalni zahtevi koje sistem za rezervisanje mesta i hrane treba da zadovolji. Funkcionalni zahtevi će biti prikazani po sektorima i to prema stakeholderima odnosno aktorima Sistema. Za izradu i prikaz funkcionalnih zahteva se koristi **Requirements model** koji zadovoljava sve standarde za pisanje istih generisan u PowerDesigner alatu.



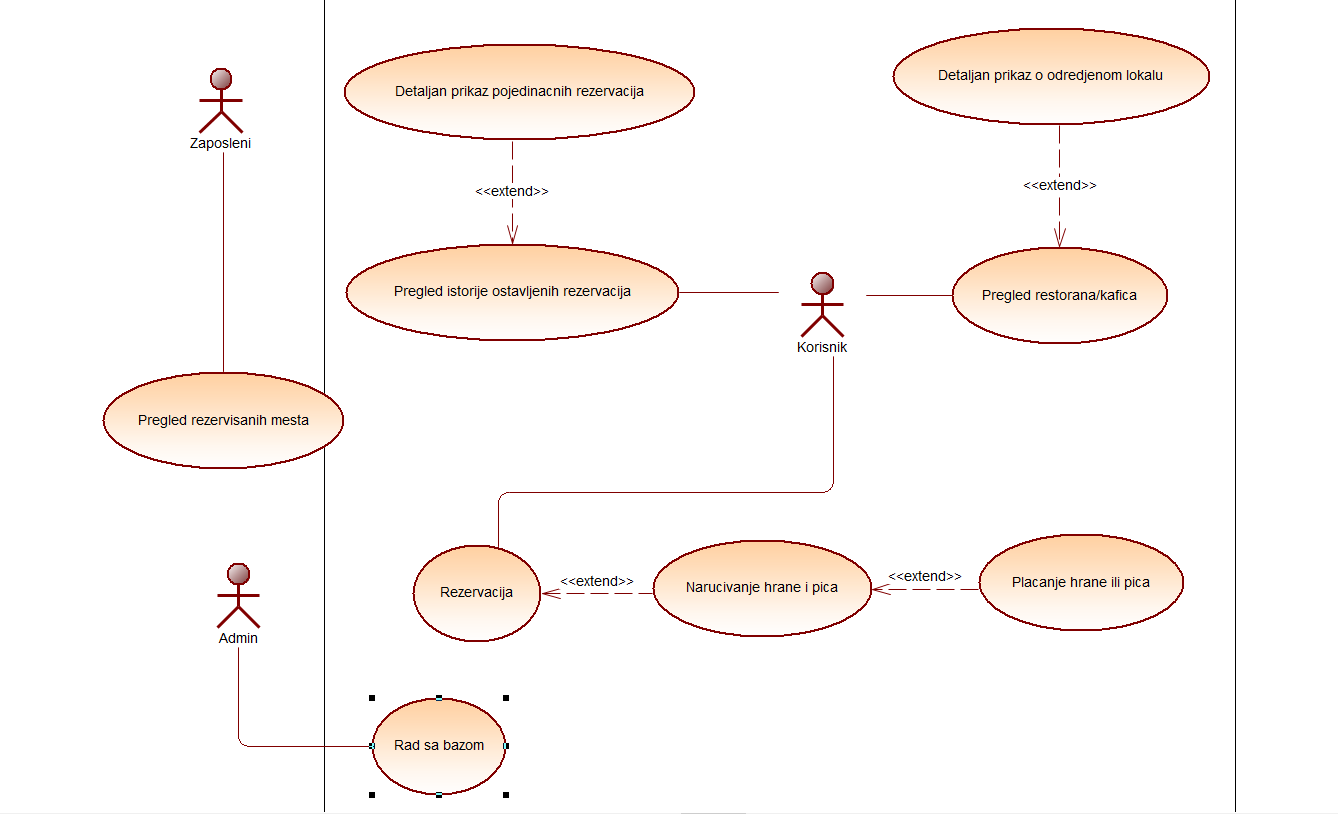
## 4.1. Funkcionalni zahtevi

Nefunkcionalni zahtevi su zahtevi koji su vezani za ceo sistem, to nisu neke akcije koje sistem preduzima. Pod nefunkcionalnim zahtevima spadaju perfomanse, bezbednost sistema, lakoća upotrebe, raspoloživost, pouzdanost.

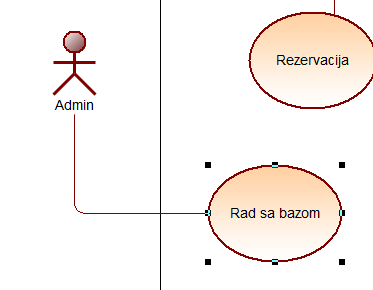


## Slučajevi korišćenja

Slučajevi korišćenja predstavljaju akcije koje akteri mogu da izvrše sa funkcionalnostima softvera. Slučajevi korišćenja su prikazani u UseCase dijagramu koristeći PowerDesigner alat.

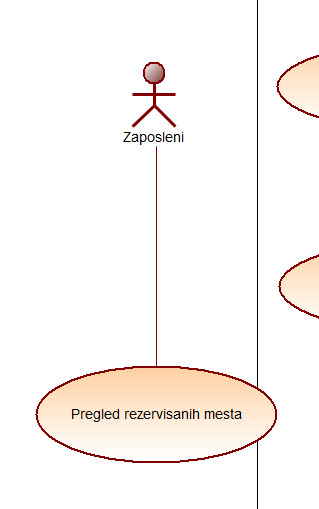


5.1. UseCase administratora



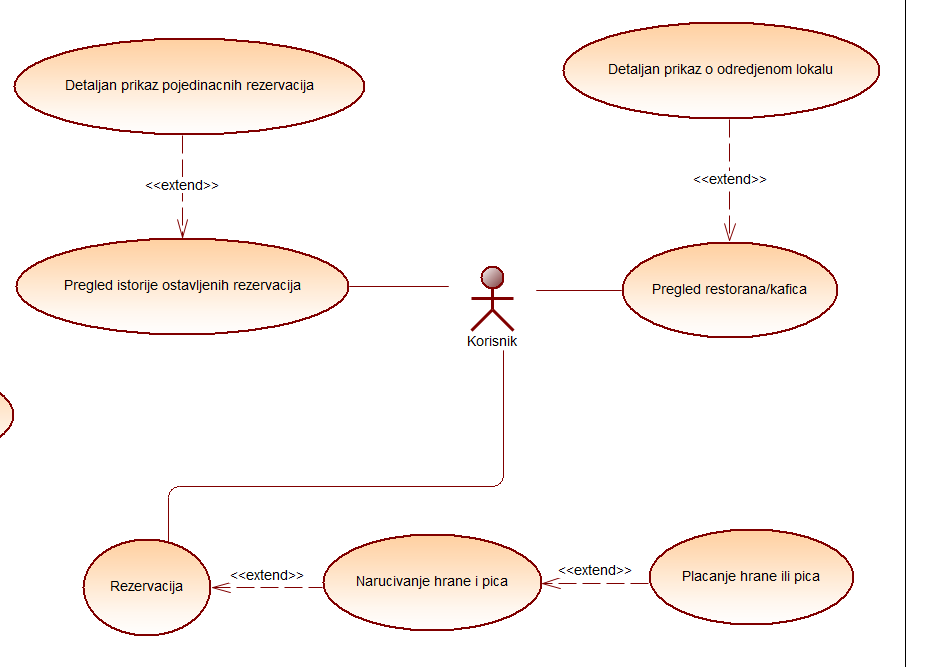
|  |  |
| --- | --- |
| Use Case | Rad sa bazom |
| Akteri | Admin |
| Cilj | Admin moze da dodaje nove kafice I restorane, moze da vrsi izmene I da brise tj. uklanja odredjene sa spiska koji vise ne nude online rezervaciju |
| Preduslovi | Admin mora da se uloguje kao admin nalog I na taj nacin dobije privilegije admina |
| Okidac | Admin bira opciju administracije podataka |
| Normalni tok | 1. Admin vrsi izbor da li dodaje, izmenjuje ili brise podatke 2. U zavisnosti od toga sta je izabrao vrsi dalji rad sa bazom |
| Alternativni tok |  |

5.2. UseCase zaposlenog



|  |  |
| --- | --- |
| Use Case | Pregled rezervisanih mesta |
| Akteri | Zaposleni u kaficu/restoranu |
| Cilj | Zaposleni ima uvid u to koji nalog je za koliko mesta u koje vreme ostavio rezervaciju u kaficu/restoranu gde zaposleni radi kako bi mogao fizicki u lokalu da rezervise mesto I ucini ga nedostupnim za druge korisnike |
| Preduslovi | Zaposleni mora da se uloguje na jedinstveni nalog restorana/kafica kom imaju pristup samo zaposljeni u njemu |
| Okidac | Zaposleni bira opciju pregleda rezervisanih mesta |
| Normalni tok | 1. Zaposleni uoci odredjenu rezervaciju |
| Alternativni tok |  |

5.3. UseCase klijenta



|  |  |
| --- | --- |
| Use Case | Narucivanje hrane i/ili pica |
| Akteri | Korisnik |
| Cilj | Korisnik moze pri rezervaciji I da pogleda meni kafica/restorana I prema tome unapred naruci hranu i/ili pice |
| Preduslovi | Korisnik mora da izabere opciju rezervisi I mora da popuni podatke o rezervaciji(vreme I datum, broj mesta) |
| Okidac | Korisnik mora da klikne na opciju naruci |
| Normalni tok | 1. Korisniku se otvara meni kafica/restorana u kom ostavlja rezervaciju I on ga pregleda 2. Vrsi odabir stavki koje zeli da unapred naruci I kolicinu 3. Unosi podatke o kreditnoj kartici 4. Potvrdjuje narucivanje I ostavljanje rezervacije |
| Alternativni tok | [ALT-1]  Narucivanje bez placanja preko sistema   1. Korisniku se otvara meni kafica/restorana u kom ostavlja rezervaciju I on ga pregleda 2. Vrsi odabir stavki koje zeli da unapred naruci I kolicinu 3. Potvrdjuje narucivanje I ostavljanje rezervacije |

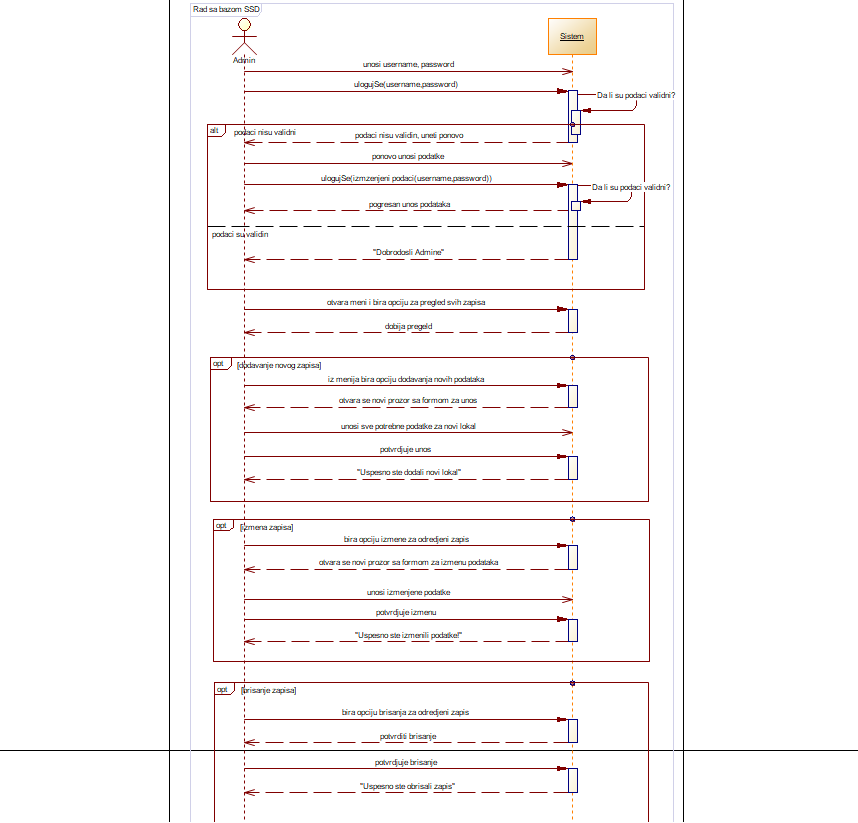
|  |  |
| --- | --- |
| Use Case | Pregled istorije ostavljenih rezervacija |
| Akteri | Korisnik |
| Cilj | Korisnik moze da pregleda sve rezervacije koje je ostavio preko svog naloga |
| Preduslovi | Korisnik mora da napravi nalog I da se uloguje |
| Okidac | Korisnik mora da izabere iz menija opciju pregleda istorije rezervacija |
| Normalni tok | 1. Korisnik otvara listu svih rezervacija koje je ostavio 2. Korisnik moze da klikne na odredjenu rezervaciju I vidi vise detalja o njoj(za kojim tacno stolom je bio, koliko stolova je ostavljeno naspram broja ljudi za koje je trazio rezervaciju, koliki je racun napravljen itd.) |
| Alternativni tok |  |

|  |  |
| --- | --- |
| Use Case | Pregled kafica/restorana |
| Akteri | Korisnik |
| Cilj | Korisnik moze da pregleda sve kafice I restorane koji nude opciju online rezervisanja mesta |
| Preduslovi | Korisnik mora da napravi nalog I da se uloguje |
| Okidac | Korisnik mora da izabere iz menija opciju pregleda kafica/restorana |
| Normalni tok | 1. Korisnik otvara listu svih restorana/kafica I moze da pregleda sve one koje nude opciju online rezervacije 2. Korisnik moze da klikne na odredjeni kafic/restoran I da pregleda njegove detalje kao sto su adresa, radno vreme, slike lokala itd. |
| Alternativni tok |  |

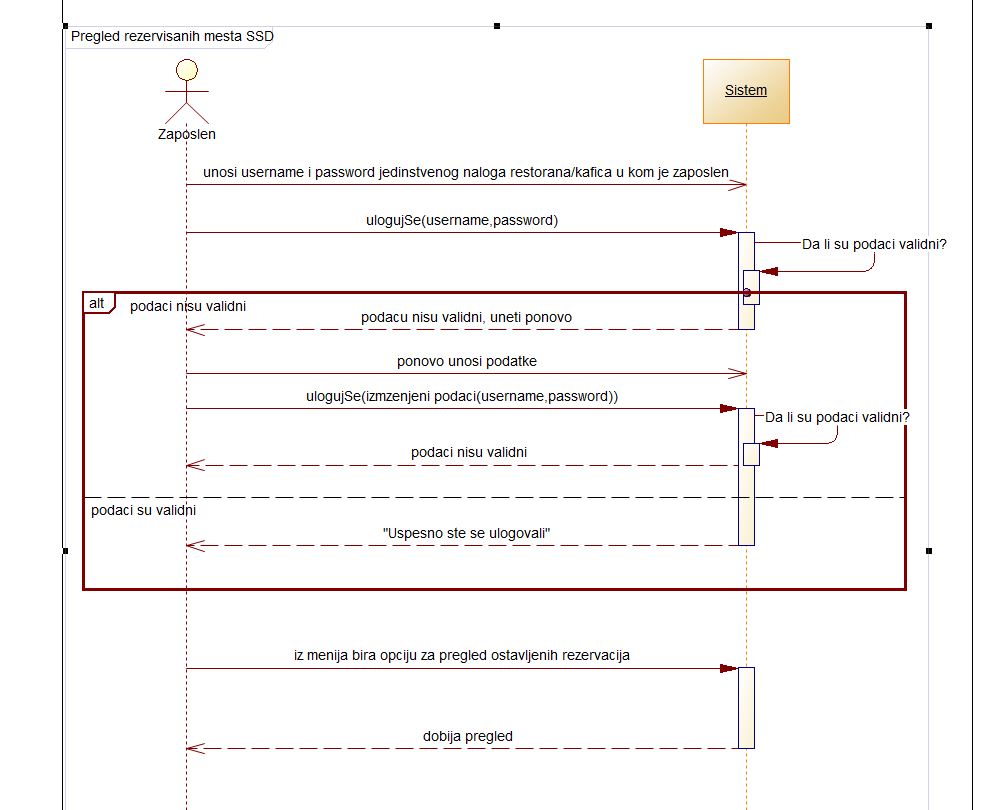
|  |  |
| --- | --- |
| Use Case | Rezervisanje mesta |
| Akteri | Korisnik |
| Cilj | Korisnik moze da rezervise mesto u zeljenom restoranu ili kaficu |
| Preduslovi | Korisnik mora da iz menija izabere opciju pregled kafica/restorana |
| Okidac | Korisnik mora da klikne na odredjeni kafic/restoran opciju rezervisi |
| Normalni tok | 1. Unosi broj mesta za koje hoce da ostavi rezervaciju 2. Unosi datum I vreme rezervacije 3. Potvrdjuje rezervaciju I zavrsava proces na taj nacin |
| Alternativni tok | [ALT-1]  Izmena podataka o vec ostavljenoj rezervaciji   1. Bira opciju izmeni rezervaciju 2. Automatski popunjene podatke prepravlja(broj mesta ili datum I vreme) 3. Povtrdjuje izmenu I zavrsava time proces |

## Sistem sekvencioni dijagrami

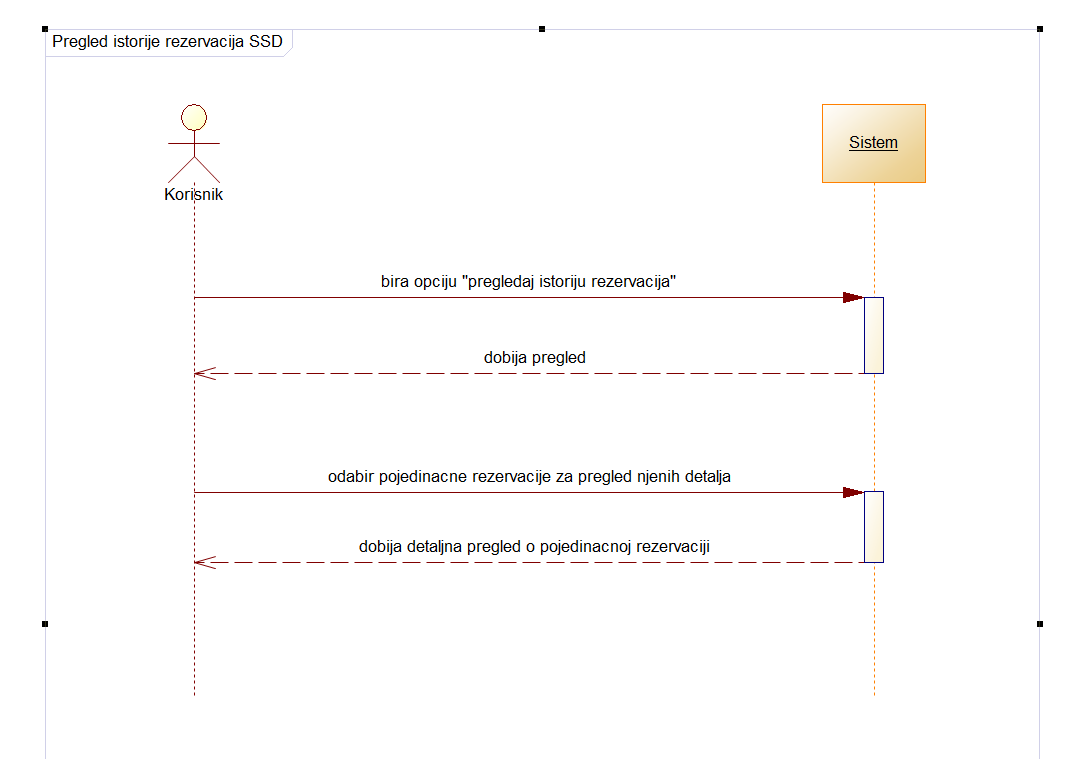
6.1. SSD administratora

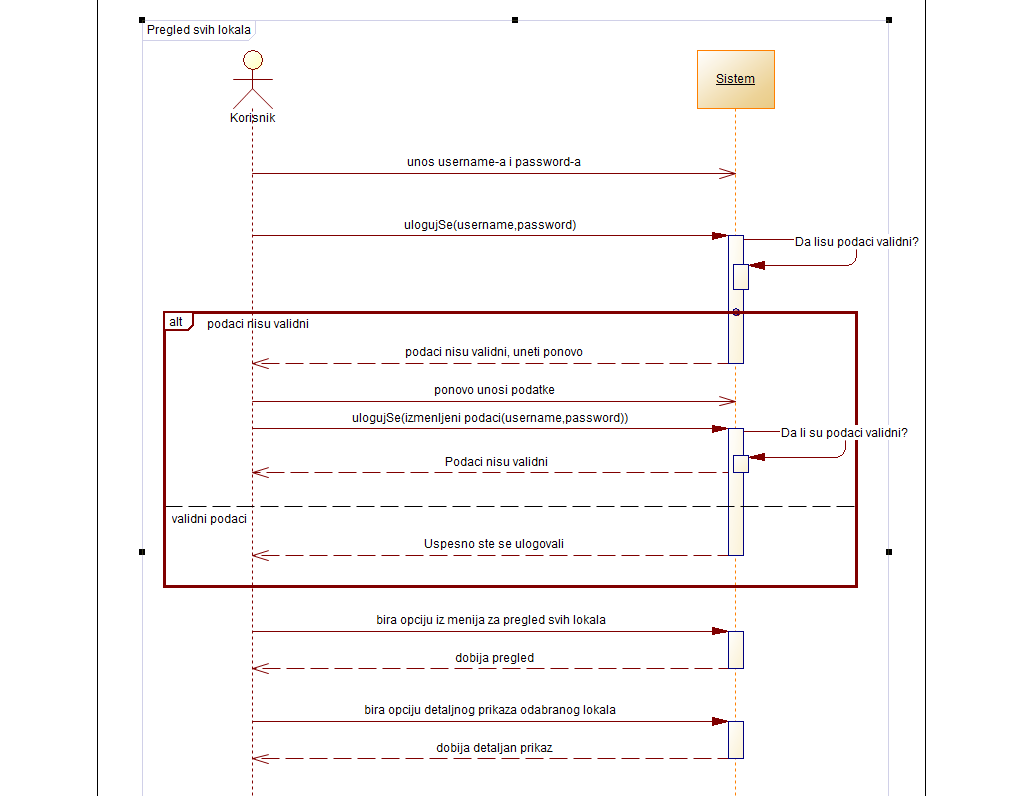


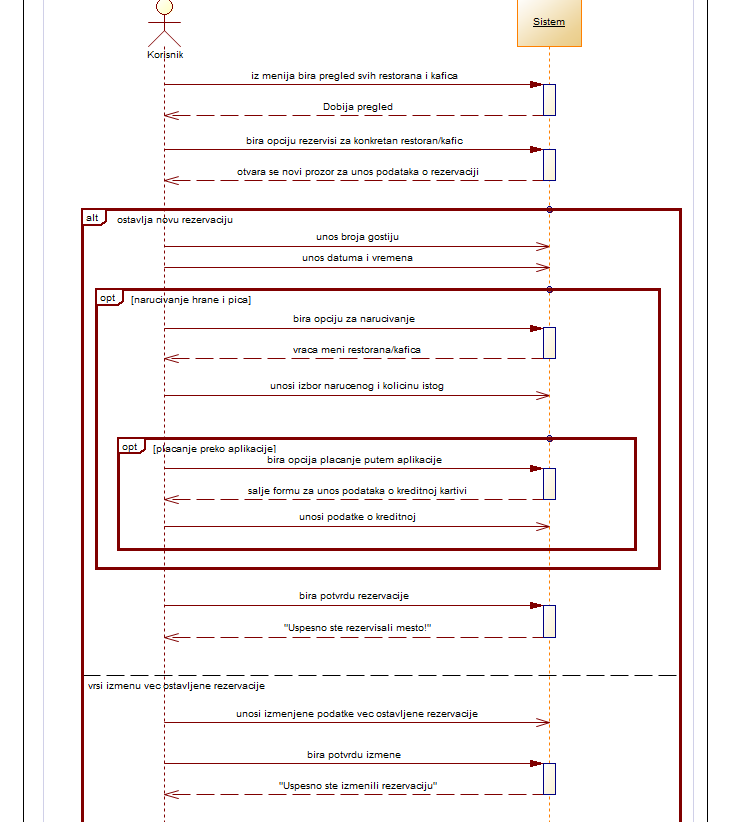
6.2. SSD zaposlenog



6.3. SSD klijenta



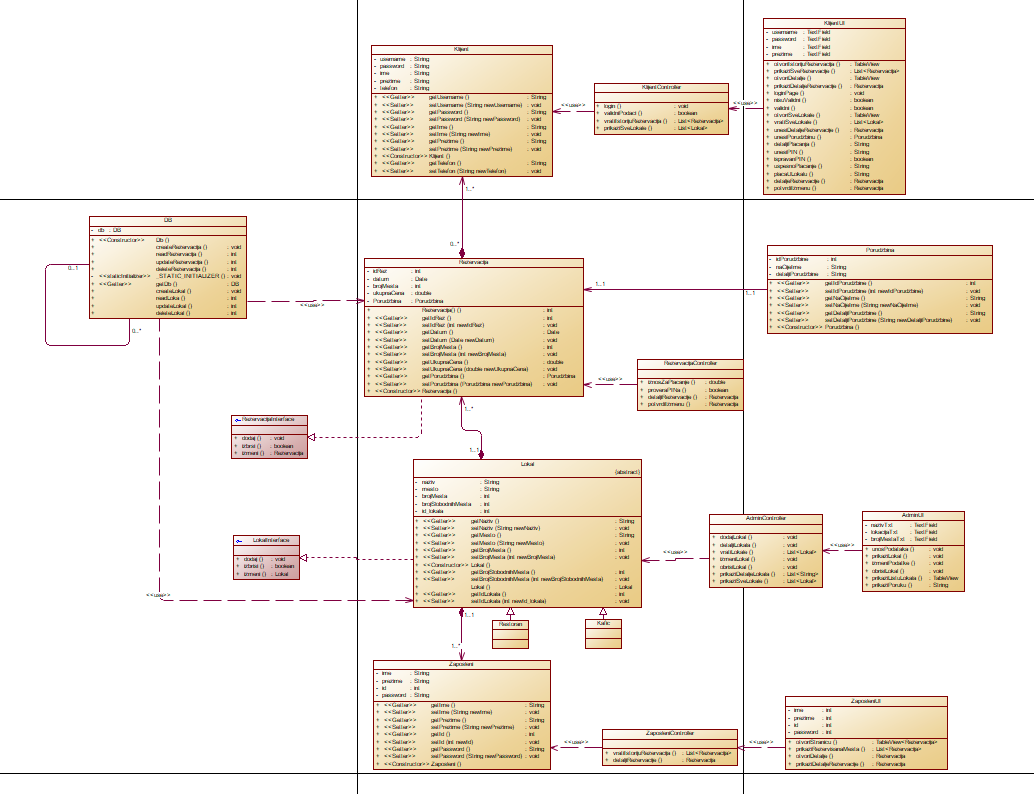




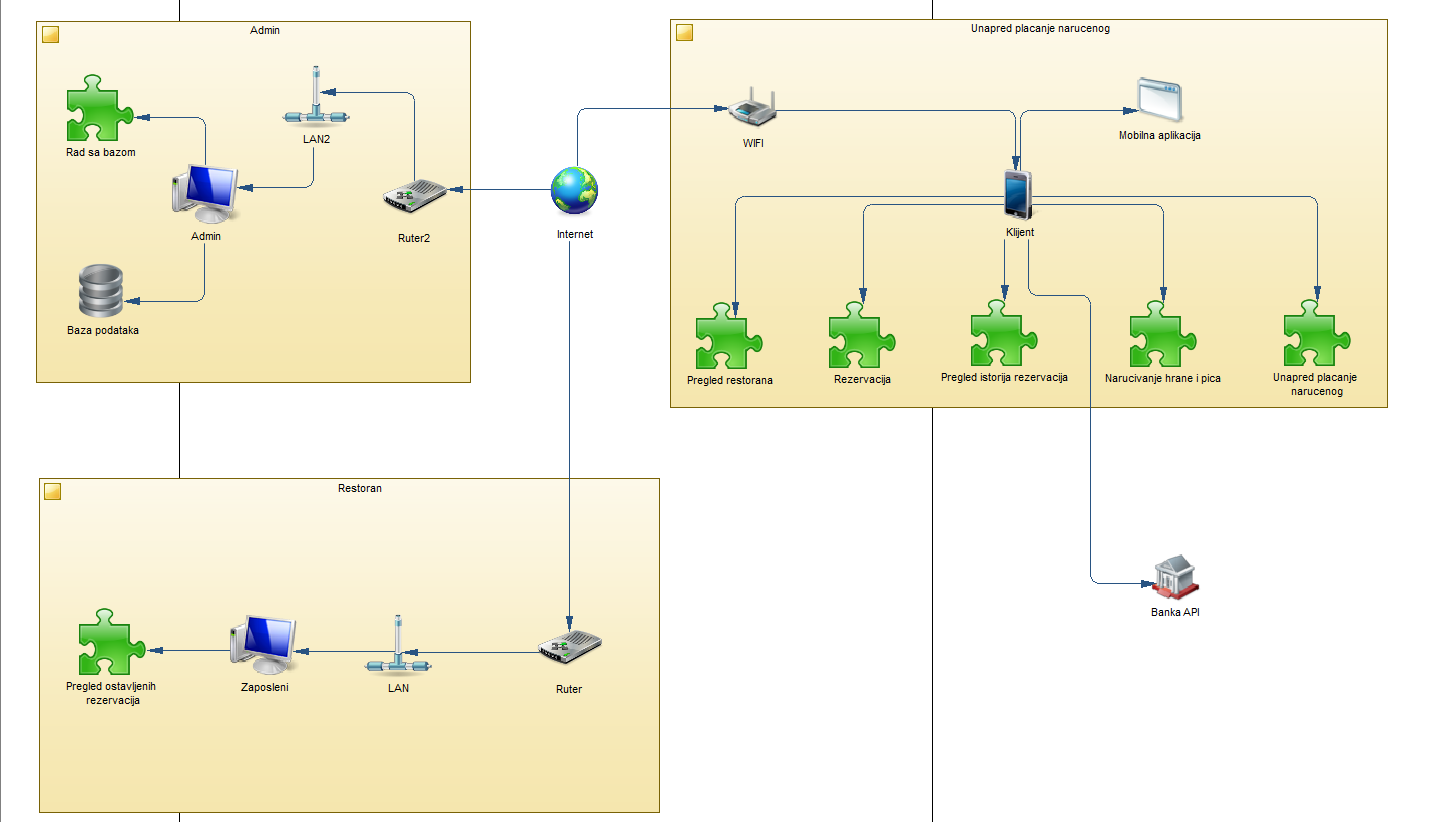
**NAPOMENA:** Zbog obimnosti i kompleksnosti detaljnih sekvencijalnih dijagrama oni radi čitljivosti i preglednosti samog dokumenta neće biti ovde prikazani već se nalaze kao prilogu u projektu.

## Klasni dijagram

U klasnom dijagramu se nalaze predviđene klase za kreiranje objekata kako bi se omogućilo potpuno funkcionisanje Lanca apoteka. Svaka klasa je detaljno analizirana i poseduju sve potrebne atribute i metode za ispunjenje ciljeva ovog projektnog zadatka. Pored samih klasa prikazane su i veze među njima kao i primena patterna **Singleton, Builder i Decorator**.



## Arhitektura



**Opis arhitekture**

Arhitektura sistema sastoji se iz tri bitna dela: klijent, admin I zaposleni. Klijent povezujuci se na internet putem Wifi ili mobilne mreze dobija pristup mobilnoj aplikaciji koja nudi sledece opcije(komponente na dijagramu): pregled restorana, rezervacija, pregled istorije ostavljenih rezervacija, narucivanje hrane I pica unapred I placanje istog. S obzirom na to da se preko aplikacije moze unapred platiti narucena hrana I pice potrebna je komponenta na koju se vezuje mobilni telefon a to je sistem banke kako bi prosla njegova uplata. Ovaj deo je opcionalan, tj. moze samo da se naruci hrana I pice bez placanja tako da u tom slucaju nije potrebna ova komponenta. Drugi deo arhitekture je deo za zaposlene koji preko rutera koji je povezan na internet dok kasnije prosledjuje podatke preko LAN mreze u okviru restorana u kome se ostavljaju rezervacije. Zaposleni je direktno povezan na taj nacin sa sistemom I ima svoju komponentu pregleda ostavljenih rezervacija kako bi kasnije mogao fizicki u objektu da rezervise odredjeni sto. Treci I poslednji deo ove arhitekture je admin deo. Ista je arhitektura kao kod dela za zaposlenog s tim sto se admin ne nalazi fizicki u tom restoranu u kome seostalvjaju rezervacije I zbog toga je preko svog rutera I LAN mreze povezan na sistem I samo on ima pristup bazi sa kojom moze da radi I upravlja.

Ceo sistem je web-orijentisan I zbog toga je lako prosiriv, I lako dostupan zbog cega je I izabrana ovakva vrsta pristupa.

## Testiranje

Testiranje služi da se utvrdi da li program radi ono bi trebalo da radi i da se otkriju greške u programu pre nego što se pusti u upotrebu. Testiranje se vrši puštanjem da softver radi, pri čemu se koriste „veštački“ podaci. Onda se proveravaju rezultati testiranja da bi se videlo da li oni ukazuju na neku grešku, nepravilnost i da bi se došlo do informacija o nefunkcionalnim svojstvima softvera.

Kada se integrišu sve komponente sistema, očekuje se da sistem pokaže ponašanje u skladu sa specifikacijom i navedenim zahtevima. To se mora testom utvrditi. Mora se razviti test za proveru da sistem radi samo ono što se od njega i očekuje.

Testiranje sistema se fokusira na testiranje interakcija komponenata sistema. Ovaj test interakcija bi trebalo da otkrije komponente sa greškama koje se otkrivaju samo kada je komponenta u interakciji sa drugim komponentama sistema. Testiranje sistema takođe doprinosi uklanjanju nerazumevanja među projektantima komponenata.

**Testiranje modula** (jedinično testiranje, unit test) proverava da li pojedinačne komponente ispravno funkcionišu sa svim očekivanim tipovima ulaza, u skladu sa dizajnom komponente. Jedinično testiranje treba vršiti u kontrolisanom okruženju tako da tim za testiranje može komponenti koja se testira da predaje unapred definisan skup podataka, i da posmatra izlazne akcije i rezultate. Zatim je potrebno da tim proveri strukturu podataka, logiku i uslove za ulazne i izlazne podatke. Govoreći o jediničnom testiranju, podrazumevamo da se bavimo testiranjem metoda ili klasa.

@Test

public void testIzmeniCenu1() throws CenaException {

System.out.println("izmeniCenu");

double cena = 500.0;

Rezervacija instance = new Rezervacija();

instance.setUkupnaCena(cena);

double expResult = 600.0;

double result = instance.izmeniCenu(100);

assertEquals(expResult, result, 0.0);

}

@Test

public void testIzmeniCenu2() throws CenaException {

System.out.println("izmeniCenu");

double cena = 200.0;

Rezervacija instance = new Rezervacija();

instance.setUkupnaCena(cena);

double expResult = 500.0;

double result = instance.izmeniCenu(50);

assertNotEquals(expResult, result, 0.0);

}

@Test(expected = CenaException.class)

public void testIzmeniCenu3() throws Exception {

System.out.println("izmeniCenu");

double cena = 200.0;

Rezervacija instance = new Rezervacija();

instance.setUkupnaCena(cena);

instance.izmeniCenu(0);

}

## Zakljucak

Projekat je rađen iz nedelje u nedelju, kako se i znanje iz softverskog inženjerstva proširivalo.

Ovakav pristup razvijanju projekta je definitivno po mom mišljenju mnogo praktičniji iako zahteva malo više vremena. Faze zahteva kao i dizajna smatram veoma bitnim za razvoj softverskih sistema jer baš one pružaju uvid u sve moguće, planirane i ne planirane puteve u razvoju dok je UML tu da svi učestici u timu mogu veoma lako i brzo da se sporazumevaju prateći samo osnove modelovanja. Takođe korišćenje ovako moćnih alata ima velike primene kako u modelovanju aspekata sistema tako i u generisanju inicijalnog koda. Smatram ovako razvijen i predstavljen projekat na temu sistema rezervacije mesta i hrane u restoranu i kaficu veoma primenljivim a isto tako i ono što je ovaj pristup razvoju omogućio je to da je sistem veoma lako nadogradiv i proširiv.

## Reference

<http://lams.metropolitan.ac.rs>

<https://refactoring.guru>