UNIVERZITET U SARAJEVU
ELEKTROTEHNIČKI FAKULTET SARAJEVO

# DOKUMENTACIJA PROJEKTA

PREDMET: OBJEKTNO ORIJENTISANA ANALIZA I DIZAJN

STUDENTI: ILMA SPAHIĆ
SELMA VUČIJAK
FARUK SMAJLOVIĆ

GRUPA: 10

MENTOR: JASMINA BAJRAMOVIĆ

## OPIS TEME:

Pošta je ustanova u kojoj se stvaraju velike gužve zbog mnogo interakcije sa ljudima. Zbog toga smo osmislili softversko rješenje koje će simulirati njen rad, prije svega da bi se olakšala organizacija i ubrzale njene funkcionalnosti, ali i da bismo modernizovali njene usluge i bili u korak sa vremenom.
Kako bi se to ostvarilo potrebna je aplikacija koja će istovremeno biti efikasna i jednostavna za korištenje. Cilj je olakšati potrošačima plaćanje računa, slanje paketa, te kupovinu razglednica i markica. Poštarima će sve informacije o paketima i ostaloj pošti biti na dohvat ruke, što će im uveliko olakšati i ubrzati posao. Uposlenicima će rad biti automatizovan što će smanjiti mogućnost greške.

UNIVERZITET U SARAJEVU
ELEKTROTEHNIČKI FAKULTET SARAJEVO

# PROTOTIP

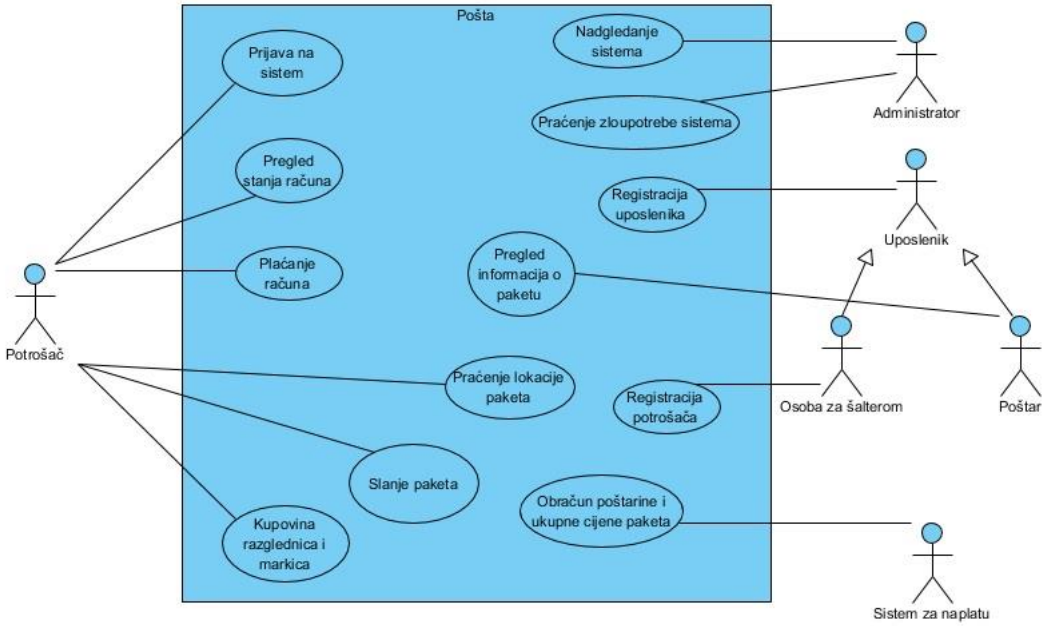
Planirano je korištenje adaptivnog layouta, te implementacija Viewova korištenjem XAML.

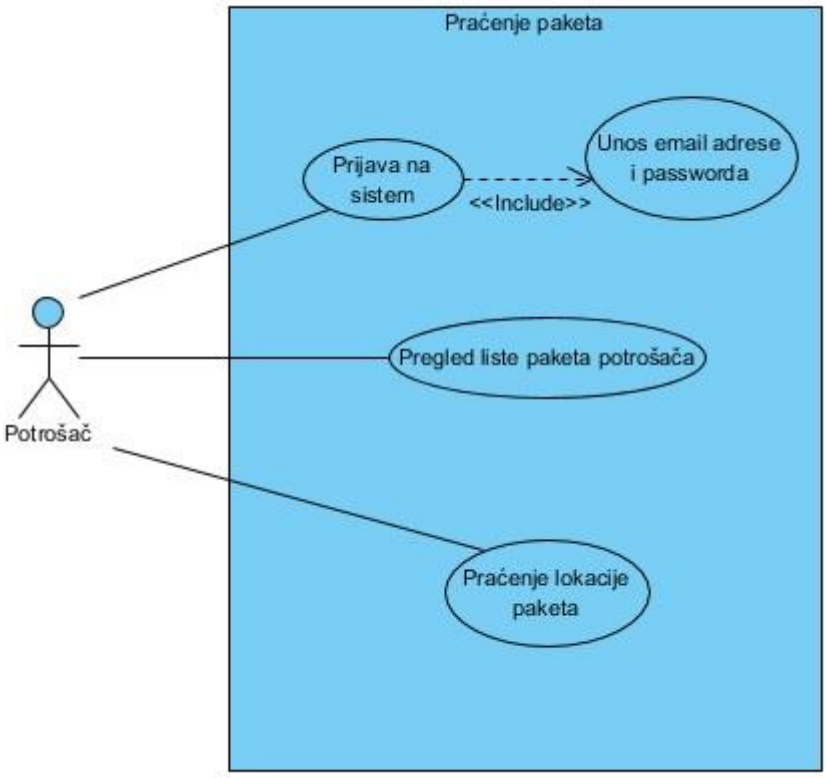
Radili: Ilma Spahić, Selma Vučijak, Faruk Smajlović

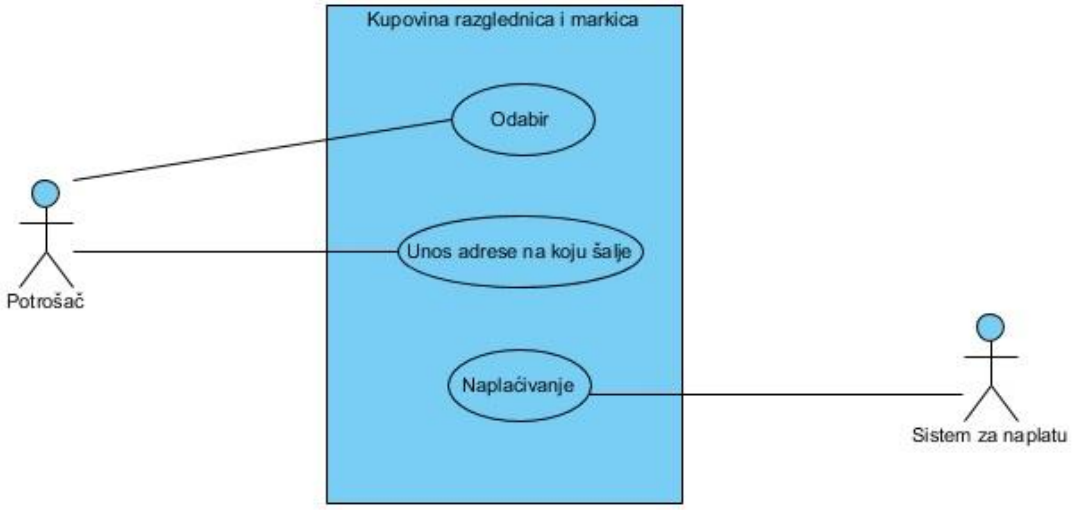
## DIJAGRAMI

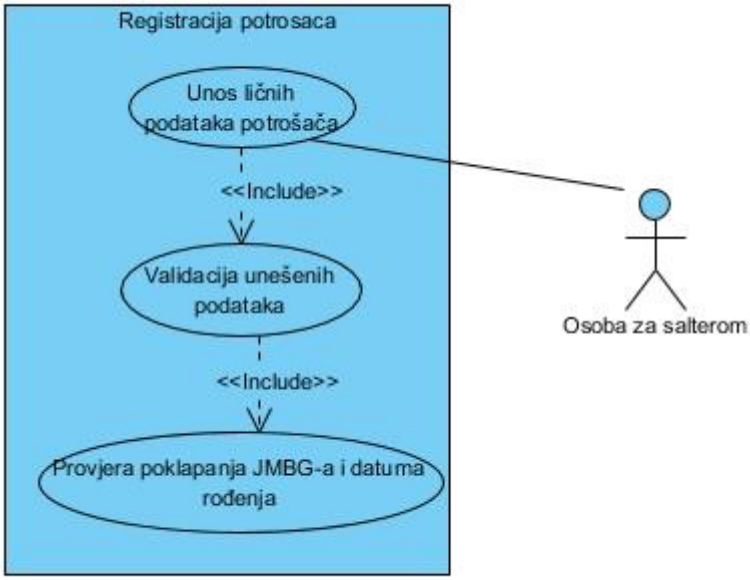
Dijagrami su rađeni u Visual Paradigm.

### *USE CASE*



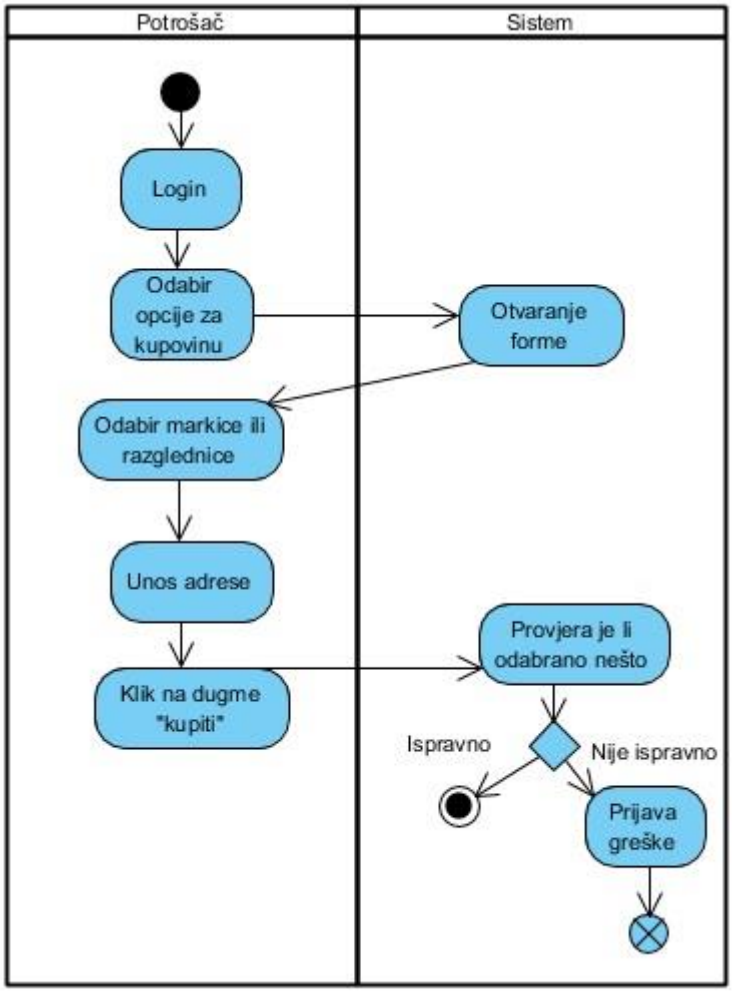

Praćenje paketa
Prijava na sistem
Unos email adrese i passworda
<<Include>>
Potrošač
Pregled liste paketa potrošača
Praćenje lokacije paketa

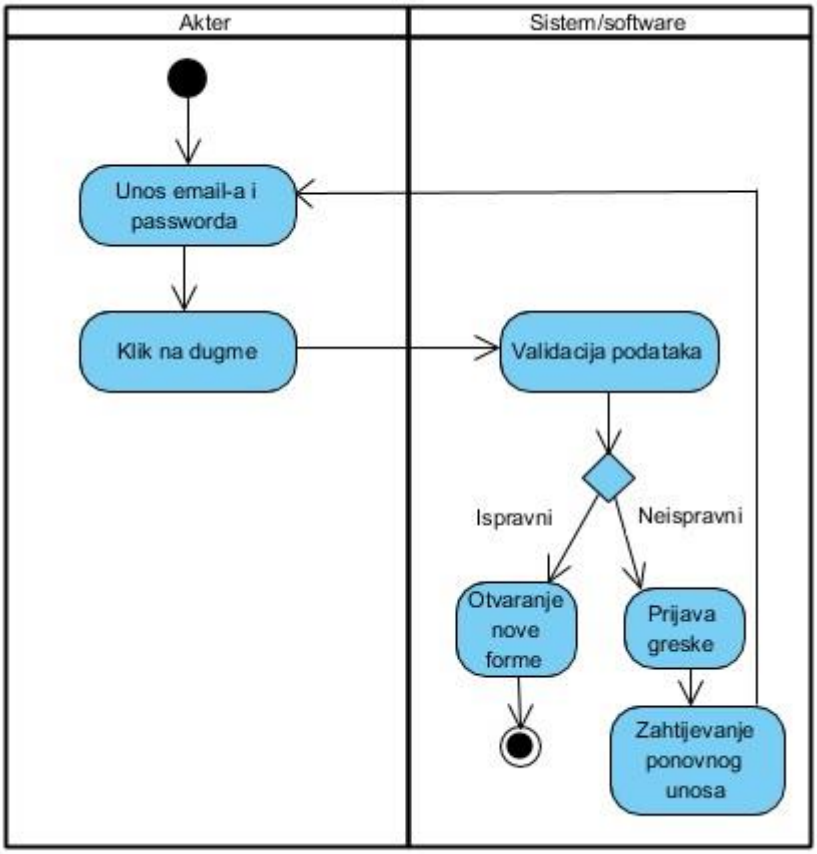

Kupovina razglednica i markica
Odabir
Potrošač
Unos adrese na koju šalje
Naplaćivanje
Sistem za naplatu

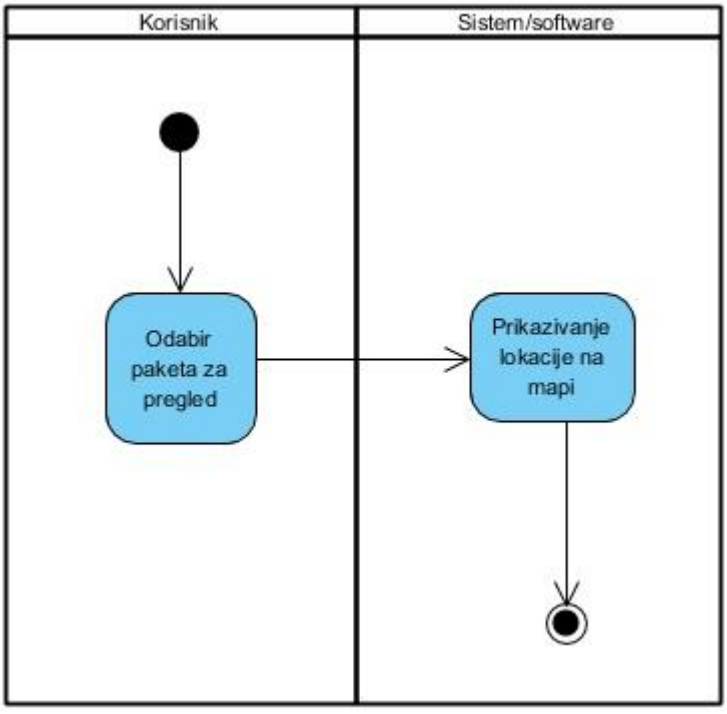


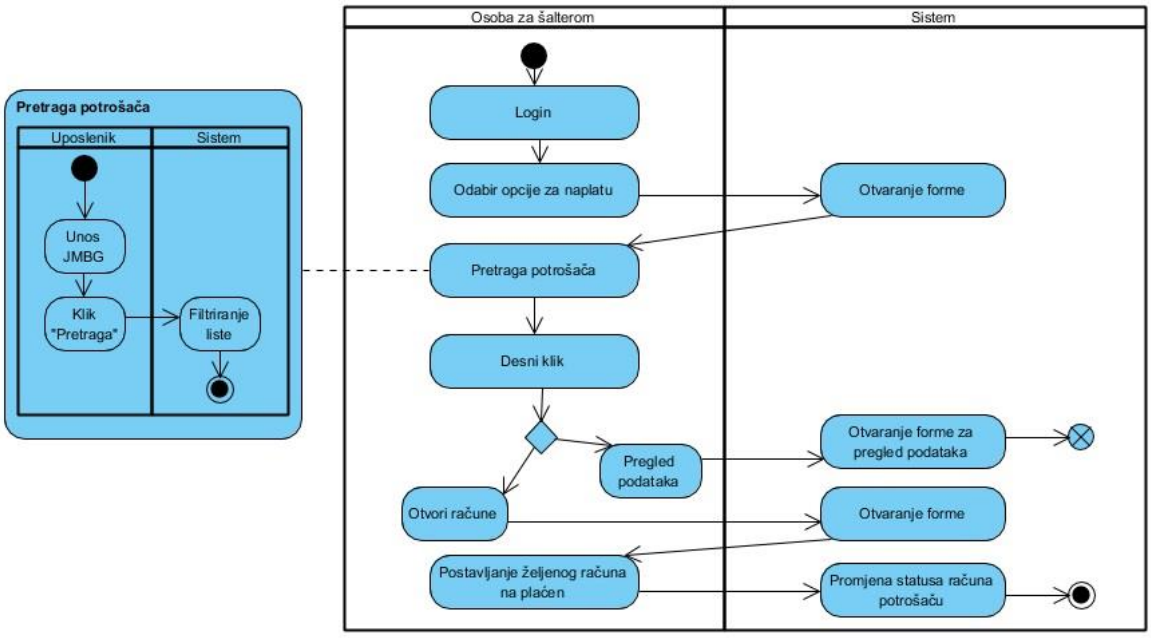

Radili: Selma Vučijak, Ilma Spahić, Faruk Smajlović

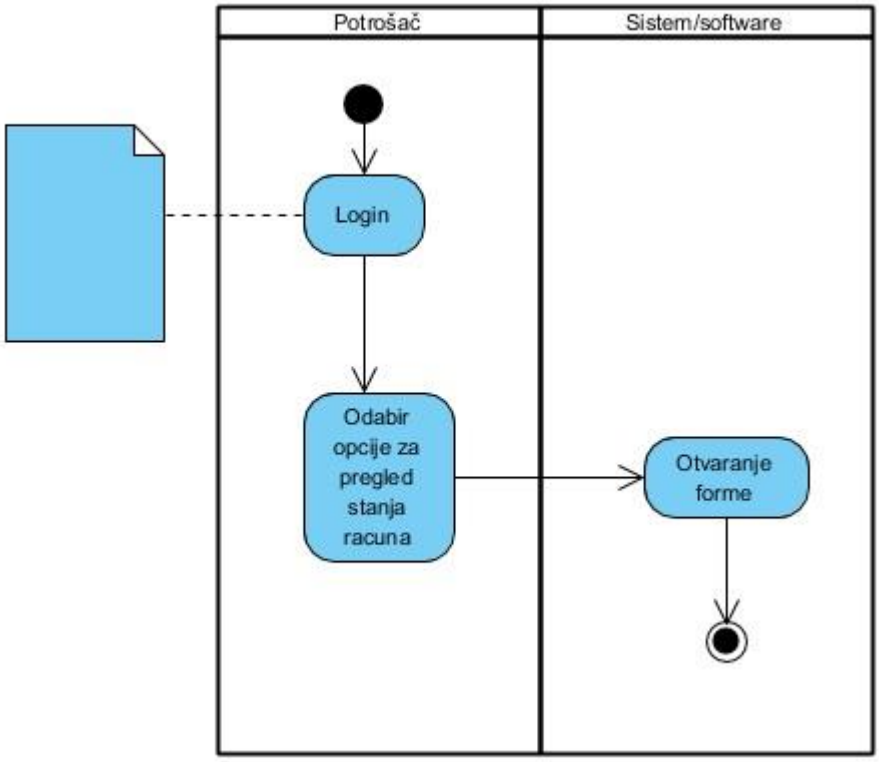
***ACTIVITY***

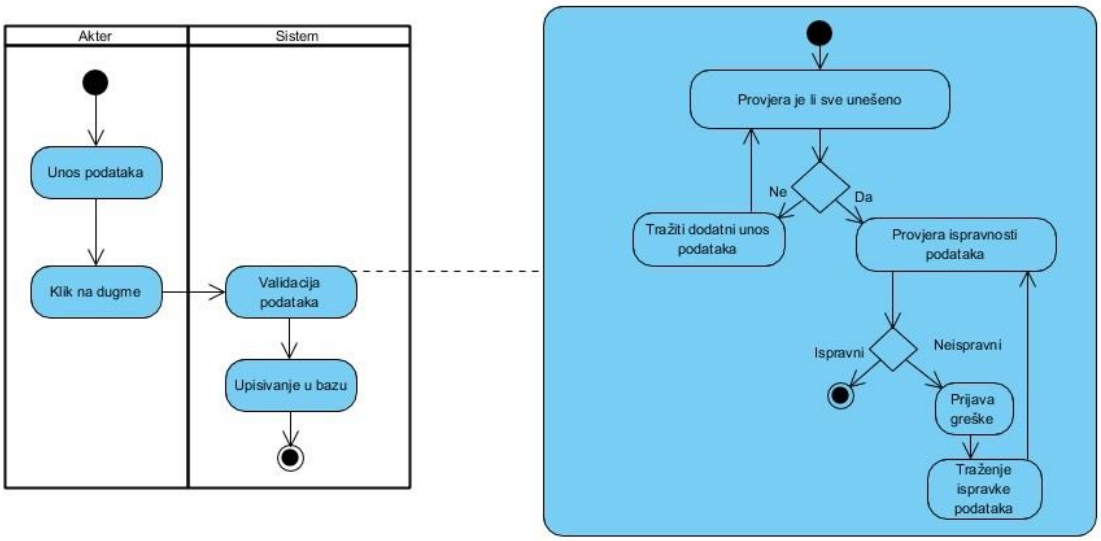


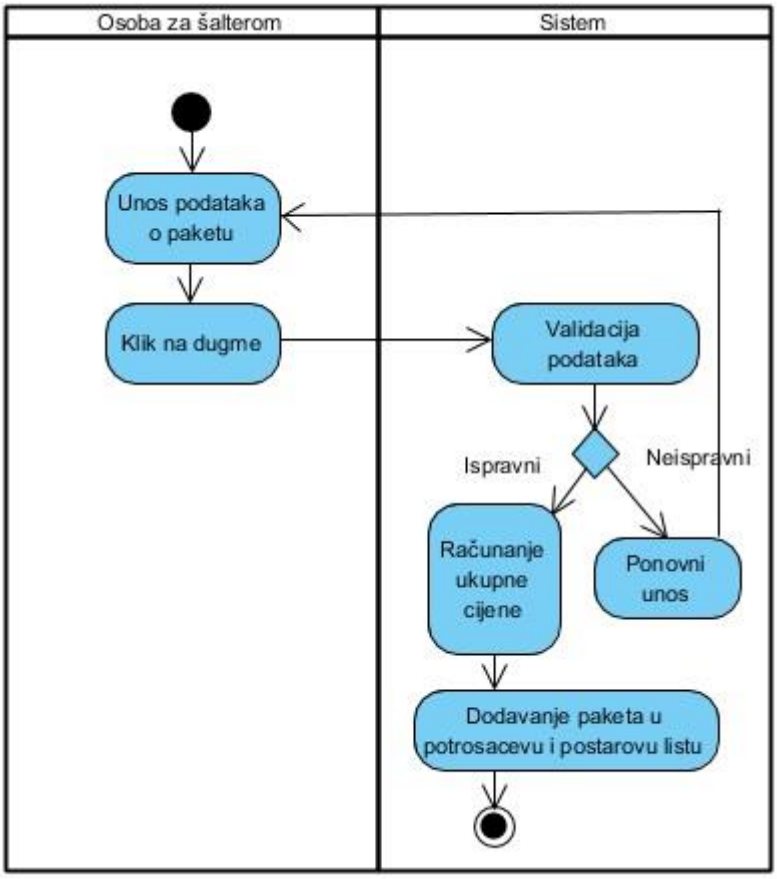

Akter
Sistem/software
Unos email-a i passworda
Klik na dugme
Validacija podataka
Ispravni
Neispravni
Otvaranje nove forme
Prijava greske
Zahtijevanje ponovnog unosa


Korisnik
Sistem/software
Odabir paketa za pregled
Prikazivanje lokacije na mapi


Pretraga potrošača
Uposlenik
Sistem
Unos JMBG
Klik "Pretraga"
Filtriranje liste
Osoba za šalterom
Sistem
Login
Odabir opcije za naplatu
Otvaranje forme
Pretraga potrošača
Desni klik
Pregled podataka
Otvaranje forme za pregled podataka
Otvori račune
Otvaranje forme
Postavljanje željenog računa na plaćen
Promjena statusa računa potrošaču


Potrošač
Sistem/software
Login
Odabir opcije za pregled stanja racuna
Otvaranje forme

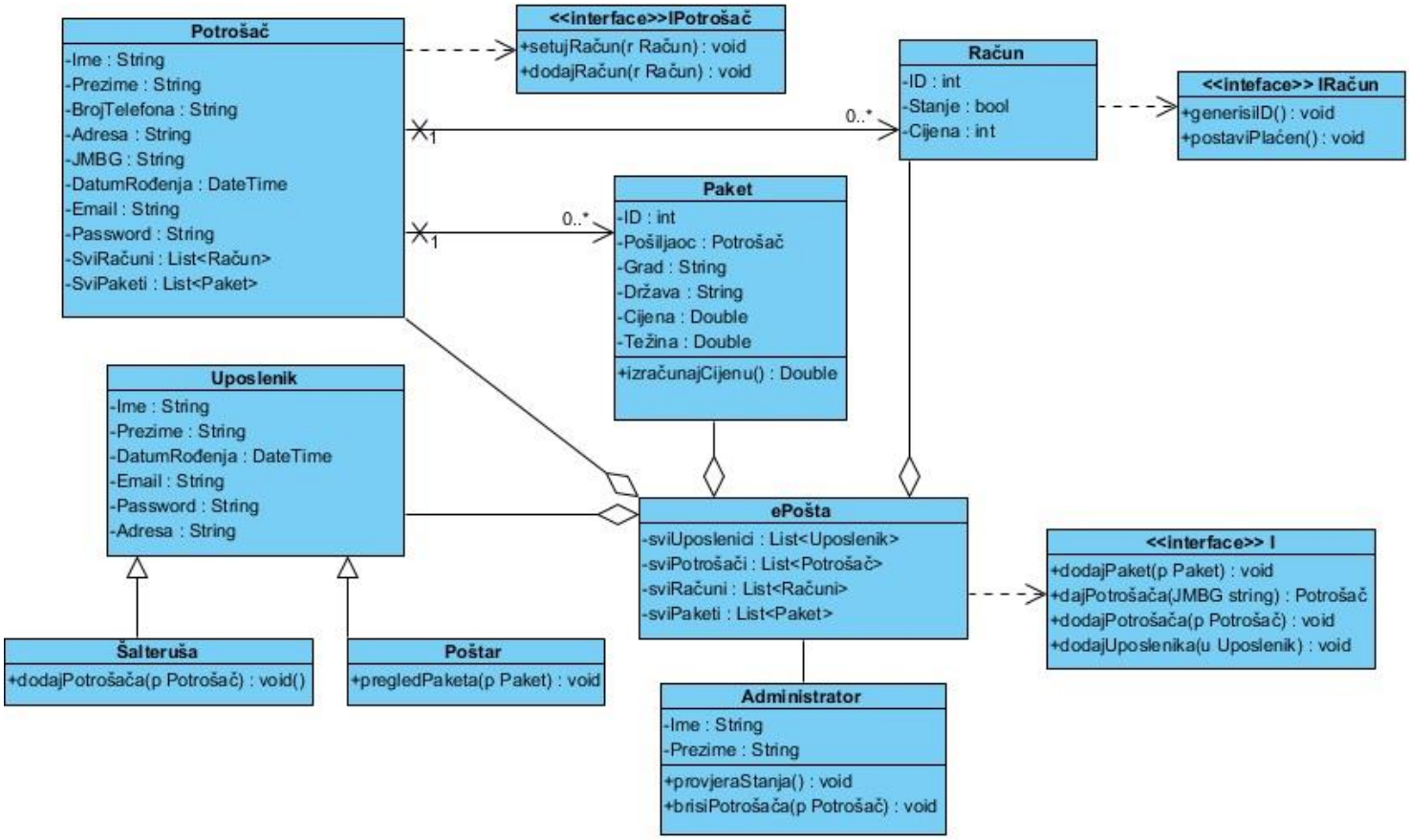



Radili: Selma Vučijak, Ilma Spahić, Faruk Smajlović

UNIVERZITET U SARAJEVU
ELEKTROTEHNIČKI FAKULTET SARAJEVO

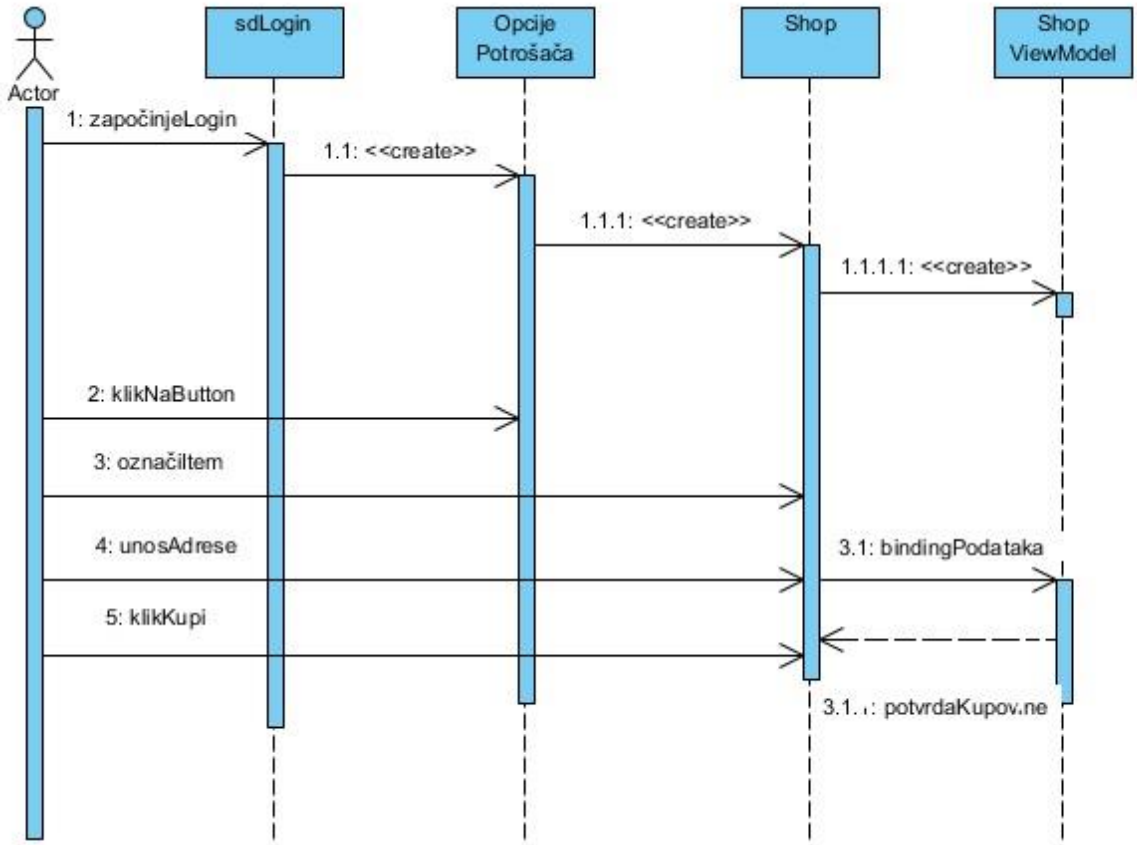
*CLASS*

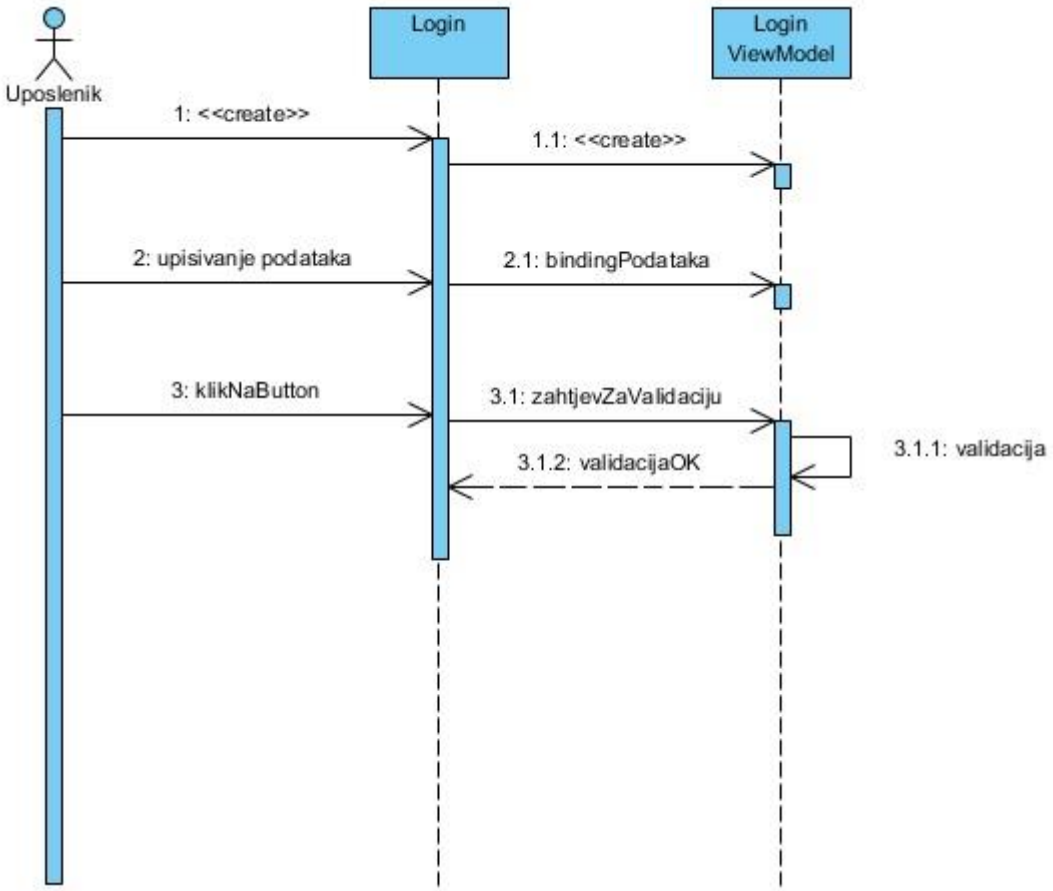


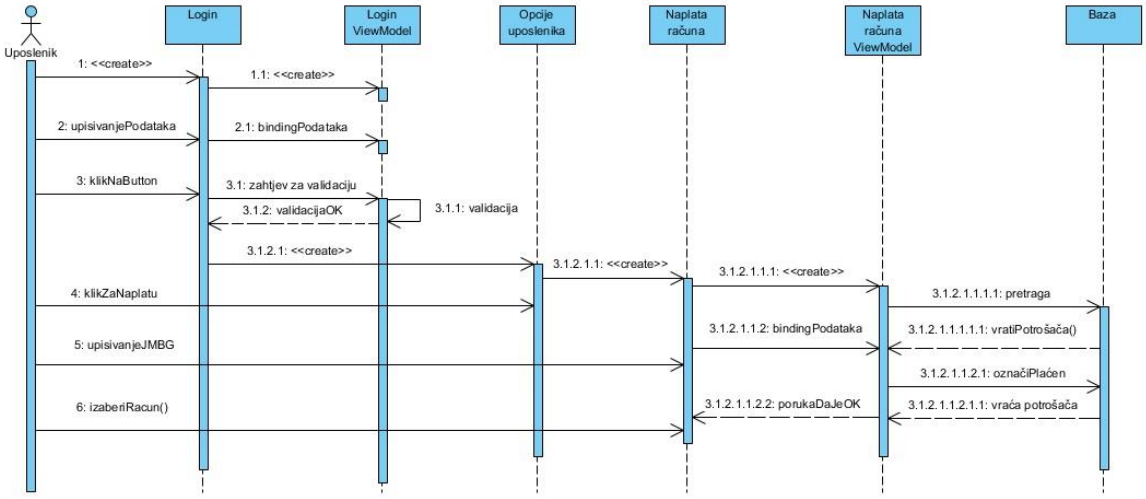

**MVVM** na linku

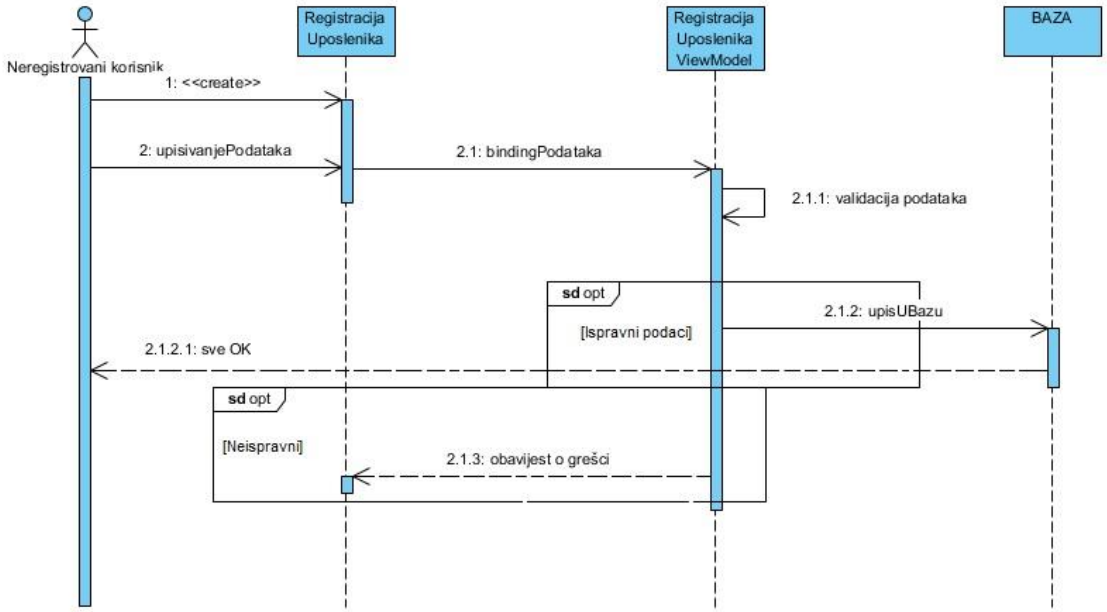
Radili: Selma Vučijak, Ilma Spahić, Faruk Smajlović

*SEQUENCE*


Actor
sdLogin
Opcije Potrošača
Shop
Shop ViewModel
1: započinjeLogin
1.1: <<create>>
1.1.1: <<create>>
1.1.1.1: <<create>>
2: klikNaButton
3: označiItem
4: unosAdrese
3.1: bindingPodataka
5: klikKupi
3.1.1: potvrdaKupovine

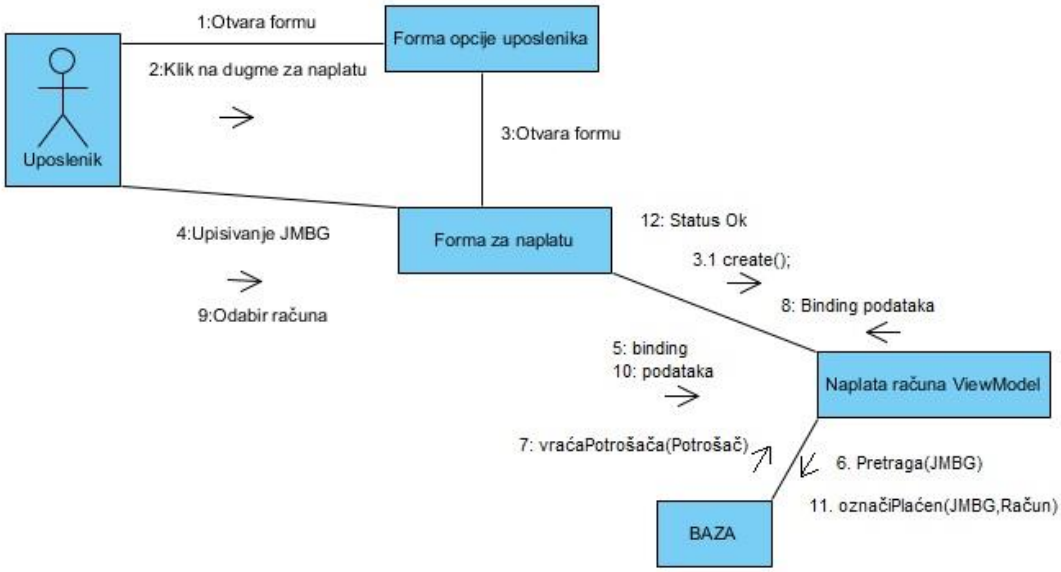

Uposlenik
Login
Login ViewModel
1: <<create>>
1.1: <<create>>
2: upisivanje podataka
2.1: bindingPodataka
3: klikNaButton
3.1: zahtjevZaValidaciju
3.1.1: validacija
3.1.2: validacijaOK

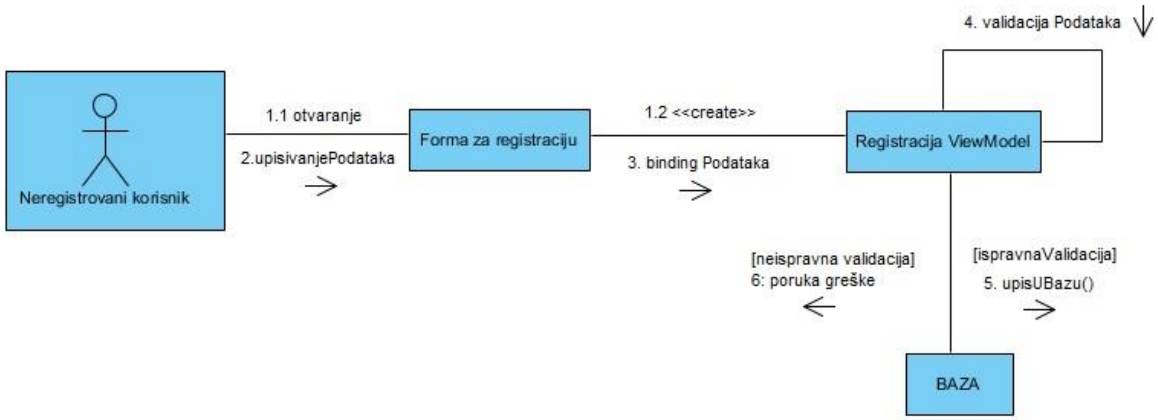

Uposlenik
Login
Login ViewModel
Opcije uposlenika
Naplata računa
Naplata računa ViewModel
Baza
1: <<create>>
1.1: <<create>>
2: upisivanjePodataka
2.1: bindingPodataka
3: klikNaButton
3.1: zahtjev za validaciju
3.1.1: validacija
3.1.2: validacijaOK
3.1.2.1: <<create>>
3.1.2.1.1: <<create>>
3.1.2.1.1.1: <<create>>
4: klikZaNaplatu
3.1.2.1.1.1.1: pretraga
3.1.2.1.1.2: bindingPodataka
3.1.2.1.1.1.1.1: vratiPotrošača()
5: upisivanjeJMBG
3.1.2.1.1.2.1: označiPlaćen
6: izaberiRacun()
3.1.2.1.1.2.2: porukaDaJeOK
3.1.2.1.1.2.1.1: vraća potrošača

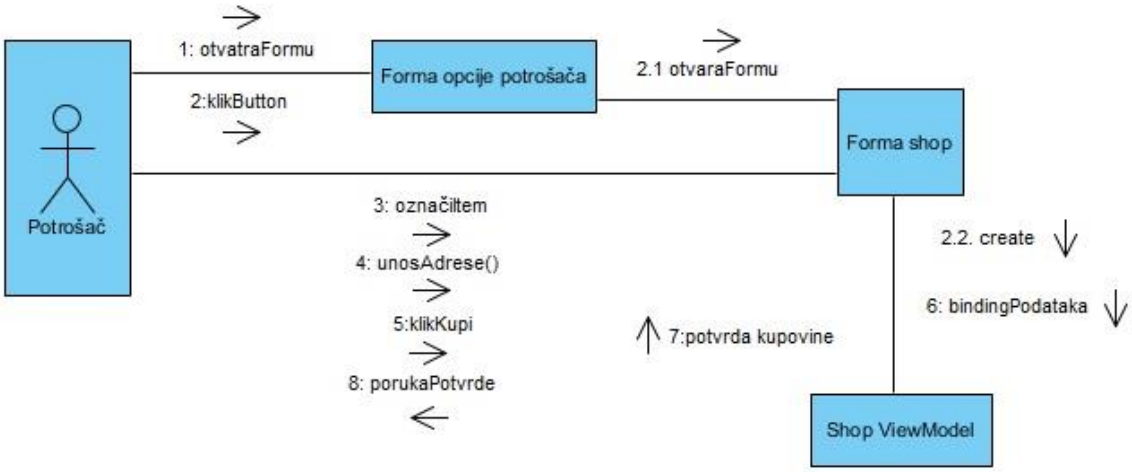



Radili: Selma Vučijak, Ilma Spahić, Faruk Smajlović

***COMMUNICATION***

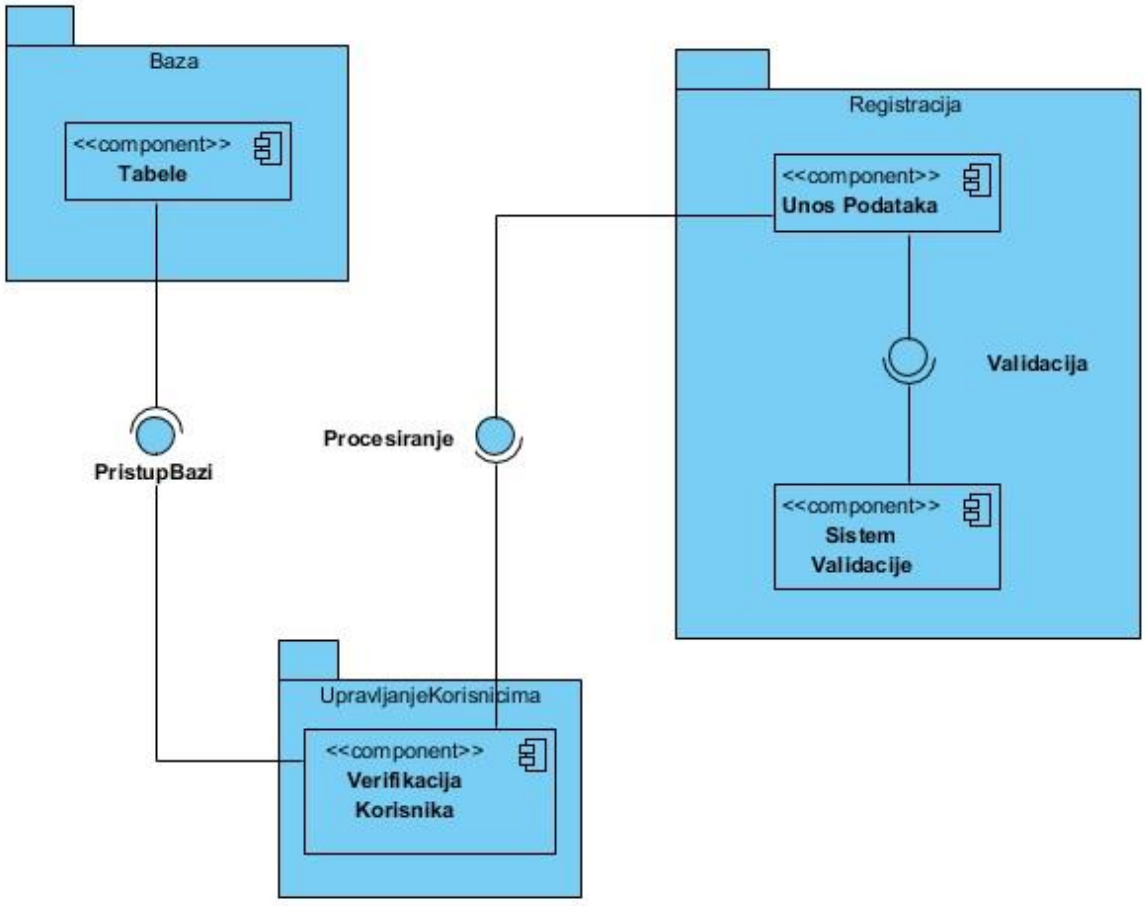




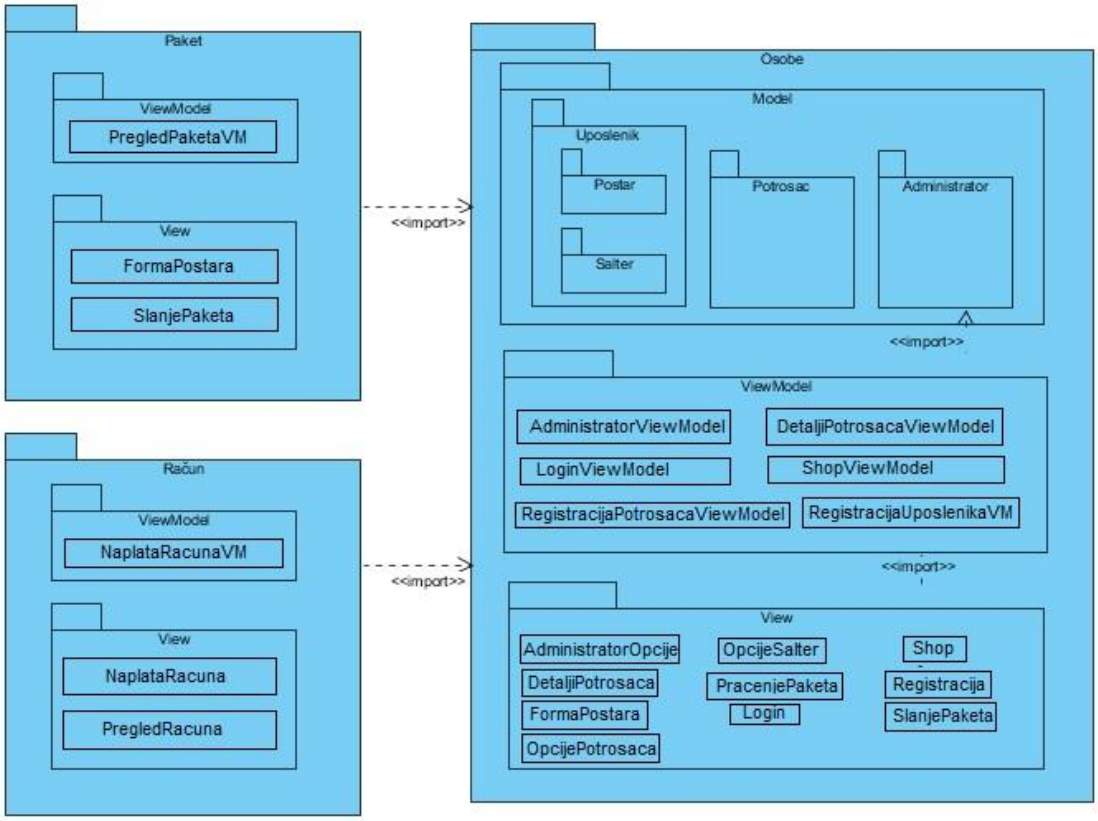

Radili: Selma Vučijak, Ilma Spahić, Faruk Smajlović

## *COMPONENT*



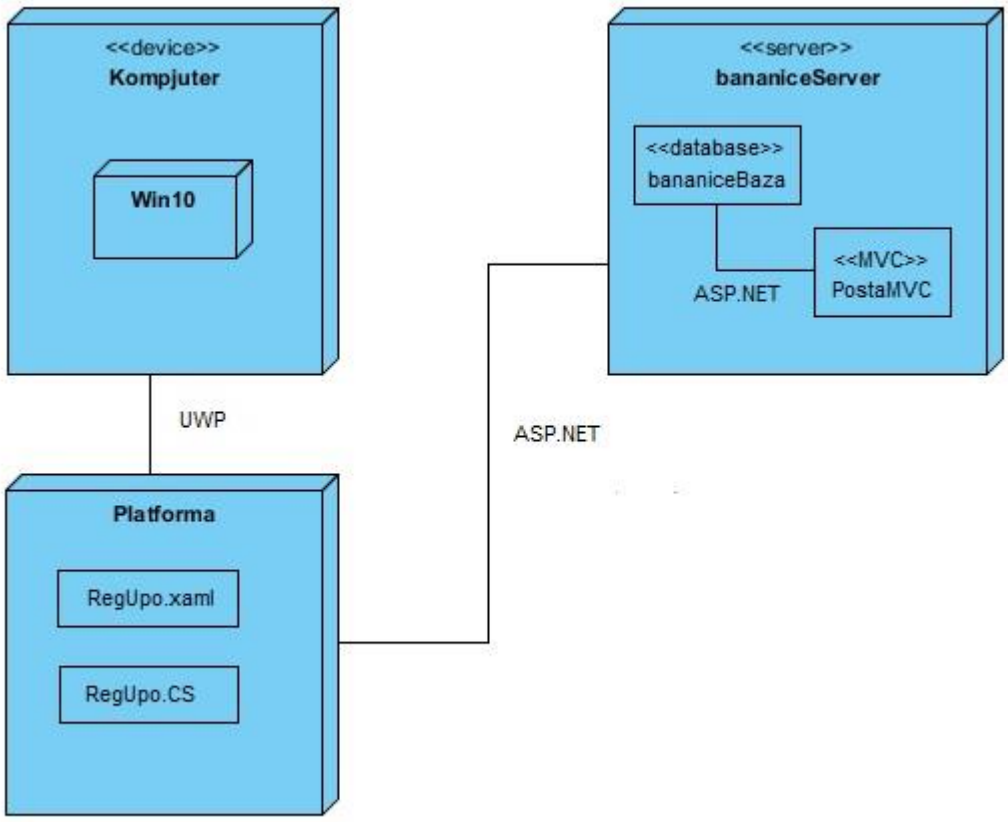

Radili: Selma Vučijak, Ilma Spahić, Faruk Smajlović

## *PACKAGE*




Radili: Selma Vučijak, Ilma Spahić, Faruk Smajlović

***DEPLOYMENT***




Radili: Selma Vučijak, Ilma Spahić, Faruk Smajlović

## *ANALIZA I PRIJEDLOZI UPOTREBE PATERNA*

Design Patterns

1. Singleton - Kreacijski pattern
Uloga Singleton paterna je da osigura da se klasa moze instancirati samo jednom i da osigura globalni pristup kreiranoj instanci klase.
Postoji vise objekata koje je potrebno samo jednom instancirati i nad kojim je potrebna jedinstvena kontrola pristupa.
Pattern smo iskoristili u nasem projektu kod klase za rad sa bazom jer nam je potrebna samo jedna klasa u citavom projektu.

2. Factory Method - Kreacijski pattern
Uloga Factory Method patterna je da omoguci kreiranje objekata na nacin da podklase odluce koju klasu instancirati. Razlicite podklase mogu na razlicite nacine implementirati interfejs. Factory Method cesto se primjenjuje jer dopusta projektu da blisko slijedi SOLID principe.
Factory Method instancira odgovarajucu podklasu(izvedenu klasu) preko posebne metode na osnovu informacije od strane klijenta ili na osnovu tekuceg stanja.
Ovaj pattern nismo koristili u nasem projektu, ali smo ga mogli iskoristiti prilikom registracije uposlenika obzirom da imamo dvije vrste uposlenika, osobu za salterom i postara.

3. Command - Pattern ponasanja
Razdvaja klijenta koji zahtjeva operaciju i omogucava slanje zahtjeva razlicitim primateljima. Komanda, to je objekt koji zna primaoca i poziva njegovu metodu.
Primalac(reciever) kada se pozove metoda execute() u komandi, izvrsava. Pozivatelj zna kako izvrsiti komandu, ali ne zna nista o konkretnoj komadi, zna samo o njenom interfejsu.

Objekti pozivatelja i primaoca se trze u klijent objektu, a klijent odlucuje sta se iz primaoca salje komandi, a koje komande se dodjeljuju pozivatelju, odlucuje koje
komande se izvrsavaju u kojem trenutku. Da bi izvrsio komandu, mora proslijediti komandu pozivatelju.
Ovaj tip patterna nismo iskoristili u nasem projektu, ali smo mogli prilikom odjavljivanja korisnika.

4. Decorator pattern- Strukturalni pattern
Osnovna namjena Decorator patterna je da omoguci dinamicko dodavanje novih elemenata i ponasanja (funkcionalnosti) postojecim
objektima. Objekat pri tome ne zna da je uradjena dekoracija sto je veoma korisno za iskoristljivost i ponovnu upotrebu
komponenti softverskog sistema. Ovaj tip patterna nismo iskoristili u nasem projektu. Mogli smo ga iskoristiti prilikom registracije. Ukoliko se registruje uposlenik, da se doda odabir tipa posla: Osoba za salterom ili postar.

5. Observer Pattern
Uloga Observer paterna je da uspostavi relaciju izmedu objekata tako kada jedan objekat

promijeni stanje, drugi zainteresirani objekti se obavještavaju..
Ovaj patter nismo koristili u nasem projektu, ali smo ga mogli koristiti prilikom implementacije funkcionalnosti pracenja paketa, ukoliko je paket stigao na adresu da korisnik dobije obavjestenje.

6. Interpreter Pattern
Interpreter pattern služi za evaluiranje gramatike nekog jezika i jezickih izraza.
Obicno se koriste TeminalExpression i CompoundExpression klase koje služe za rekurzivno rješavanje problema interpretacije nekog jezickog izraza gdje je CompoundExpression neki dio pravila koji poziva drugo pravilo za obradu izraza, dok je TerminalExpression bazni slucaj za neko pravilo.
Nismo koristili ovaj pattern u nasem projektu. Mozemo ga iskoristiti za provjeravanje validnosti JMBG broja i datuma rodjenja, ili provjere duzine passworda.

7. Adapter Pattern
Osnovna namjena Adapter paterna je da omogući širu upotrebu već postojećih klasa. Adapter patern kreira novu adapter klasu koja služi kao posrednik između
originalne klase i željenog interfejsa. Tim postupkom se dobija željena funkcionalnost bez izmjena na originalnoj klasi i bez ugrožavanja integriteta cijele aplikacije.
Nismo koristili ovaj pattern unutar našeg projekta zbog već dobro osmišljenih klasa koje ne zahtijevaju promjene.

8. Proxy Pattern
Namjena Proxy paterna je da omogući pristup i kontrolu pristupa stvarnim objektima. Proxy je obično mali javni surogat objekat koji predstavlja kompleksni objekat čija aktivizacija se postiže na osnovu
postavljenih pravila. Proxy patern rješava probleme kada se objekt ne može instancirati direktno. Ovaj pattern nije iskoristen, ali se mogao iskoristiti za restrikciju brisanja potrošača svima osim administratoru sistema.

9. Bridge Pattern
Osnovna namjena Bridge paterna je da omogući odvajanje apstrakcije i implementacije neke klase tako da ta klasa može posjedovati više različitih apstrakcija i više različitih implementacija za pojedine
apstrakcije. Nismo koristili ovaj patter unutar našeg projekta, ali smo ga mogli iskoristiti za razmjenu poruka između korisnika i sistema.

- Mogućnost prijave na sistem
- Mogućnost registracije novih korisnika sistema
- Mogućnost kupovine razglednica i markica
- Mogućnost plaćanja računa
- Mogućnost pregleda stanja računa
- Mogućnost praćenja lokacije paketa
- Mogućnost podnošenja zahtjeva za slanje paketa
- Mogućnost pregleda svih informacija o paketu
- Mogućnost obračuna poštarine i ukupne cijene paketa

| **Projektni zadatak** | **Članovi tima** | **Implementirano** |
|---|---|---|
| Adaptivni layout | Selma Vučijak, Ilma Spahić, Faruk Smajlović | Da |
| MVVM – Views | Selma Vučijak, Ilma Spahić, Faruk Smajlović | Da |
| MVVM – Models | Selma Vučijak, Ilma Spahić, Faruk Smajlović | Da |
| MVVM – ViewModels | Selma Vučijak, Ilma Spahić, Faruk Smajlović | Da |
| Entity framework | Selma Vučijak, Ilma Spahić, Faruk Smajlović | Da |
| Azure | Selma Vučijak, Ilma Spahić, Faruk Smajlović | Da |
| Vanjski uređaj | Selma Vučijak, Ilma Spahić, Faruk Smajlović | Ne |

## SCENARIJ

Scenarij mozete vidjeti na naredna tri linka 1, 2 , 3.