Oblikovanje programske potpore

Ak. god. 2017./2018.

OneDayJob

Dokumentacija, Rev. 2.0

Grupa: *ProjectExceptionAI*

Voditelj: *Tin Ivan Križ*

Datum predaje: *18. 1. 2018*

Sadržaj

1. Dnevnik promjena dokumentacije

Rev.	Opis promjene/dodatka	Autor(i)	Datum
0.1	Napravljen predložak.	Križ	23.10.2017.
0.15	Napisana prva verzija opisa projektnog zadatka	Križ	3.11.2017
0.2	Napisani ostali zahtjevi	Križ	6.11.2017
0.25	Napisani opisi obrazaca uporabe	Kalafatić, Miličević	8.11.2017
0.4	Dodani sekvencijski dijagrami bez opisa	Vidak	13.11.2017
0.41	Dodan upotpunjen dnevnik sastajanja za prvi ciklus	Križ	13.11.2017
0.55	Dodan nacrt baze i opis	Butorac	15.11.2017
0.7	Dodan dijagram razreda i objekata	Križ	16.11.2017
0.85	Upotpunjeni funkcionalni zahtjevi sa svim dijagramima i opisima	Kalafatić, Miličević, Vidak	17.11.2017
1.0	Napisani dodatci, zaključak, pojmovnik, uređivanje dokumenta	Carin	17.11.2017
1.30	Dignuta funkcionalna baza	Butorac	5.1.2018
1.50	Povezana aplikacija s bazom	Miličević, Carin, Kalafatić, Križ	12.1.2018
1.80	Aplikacija dorađena za predaju	Miličević, Carin, Kalafatić, Križ	18.1.2018
2.0	Dokumentacija dorađena za predaju	Vidak	18.1.2018

2. Opis projektnog zadatka

Cilj projekta je razviti mobilnu aplikaciju za android uređaje koja služi kao oglasnik za jednodnevne poslove. Ona bi bila poveznica između dvije ciljane skupine - ljudi koji imaju viška vremena i voljni su to vrijeme unovčiti i ljudi koji su stalno u strci ili nisu u fizičkom stanju da odrade neke jednostavne i (načelno) kratke poslove.

Trenutno postoje njuškalo i slični oglasnici, ali svim takvim oglasnicima je cilj naći ljudima stalne poslove, dok je našoj aplikaciji cilj uposliti ljude na nekoliko sati i za takvu primjenu je optimirana. Korisnicima bi pronalazak posla trebao biti pojednostavljen bogatim mogućnostima pretrage. Korisnik može pretraživati ili filtrirati poslove na temelju naziva, lokacije, vremena trajanja, vremena početka rada, zarade, a može i vidjeti GPS kartu s poslovima u svojoj blizini.

Budući da ova aplikacija nužno zahtijeva interakciju između korisnika, polaznu aktivnost korisničkog sučelja predstavlja registracija korisnika ili njihova prijava u slučaju da je korisnik već registriran. Prije svega, jedna od esencijalnih točaka koju je neophodno dotaknuti pri registraciji korisničkog profila jest sigurnost. Da bi korisnici mogli postavljati ili prihvaćati poslove, moraju verificirati račun putem električne pošte i moraju imati postavljenu svoju sliku kao sliku profila na kojoj se jasno vidi čovjekovo lice. Također mora biti omogućen sustav ocjenjivanja korisnika kako bi se korisnicima omogućila što veća razina sigurnosti (korisnici puno više mogu vjerovati drugom korisniku ako drugi korisnik ima visoku ocjenu). Dakle, registracija korisnika nije završena sve dok se korisniku ne verificira slika i email.

Jednom kada posloprimac nađe posao koji ga zanima, on se mora javiti odgovarajućem poslodavcu. Dogovor između poslodavca i posloprimca bit će omogućen kroz samo aplikaciju sustavom poruka. Svaki korisnik će imati svoj sandučić gdje će vidjeti listu svih ljudi s kojima je zadnje pričao i odakle će moći slati i primati poruke.

Aplikacija razlikuje dvije vrste korisnika:

- Neprijavljeni korisnici – korisnici koji se nisu prijavili u sustav. Nemaju pravo pristupa sustavu sve dok se ne prijave.
- Prijavljeni korisnici – korisnici koji su se registrirali i verificirali račun, te prijavili u sustav. Imaju mogućnost koristiti sve mogućnosti koje aplikacija nudi.
- Administratori – korisnici koji imaju ulogu održavanja sustava. Imaju mogućnost koristiti aplikaciju kao bilo koji drugi prijavljeni korisnik, ali također mogu brisati poslove i korisnike zbog neprimjerenog sadržaja.

Ključne točke kod razvoja sustava su razvoj početne oglasne ploče poslova, mogućnost komunikacije između dviju zainteresiranih strana i sustav registracije i verifikacije. To su najbitniji dijelovi jer je već s njima omogućena temeljna funkcionalnost aplikacije. Kad su te tri točke riješene, onda je redom bitan razvoj filtriranja poslova, sustava ocjenjivanja korisnika, profila korisnika i na kraju je najmanje bitna implementacija GPS karte s prikazanim poslovima u blizini. Najmanje bitna je karta jer korisnik teoretski može filtrirati poslove po lokaciji i bez GPS karte.

U budućnosti bi se aplikacija mogla nadograditi sa sustavom naplate. Trenutni opseg aplikacije uključuje prihvaćanje poslova kao usmeni dogovor između posloprimca i poslodavca. Ovaj aspekt bi se mogao obogatiti sustavom naplate gdje obje stranke prihvate cifru po početku posla koja se onda isplati po završetku posla. Problem bi kod takvog pristupa bio pravne prirode. Naš trenutni opseg uključuje najčešće male poslove i male iznose koji bi se isplaćivali na licu mjesta bez računa. Uz ovakvu nadogradnju, postojala bi evidencija zarade koja bi se trebala i pravdati u očima države.

3. Pojmovnik

- Aktivnost (engl. *activity*) – poseban razred koji u mobilnim aplikacijama predstavlja jedan ekran. Ona sadrži svu logiku vezanu za pojedini ekran.
- Entitet (engl. *entity*) – objekt u sustavu kojeg želimo modelirati i o kojem želimo spremati podatke. Pojam se najčešće veže uz relacijske baze podataka.
- GUI (engl. *Graphical User Interface*) – korisničko sučelje. Skup svih grafičkih komponenata koje korisnik vidi i s kojima ulazi u interakciju.
- XML (engl. Extensible Markup Language) – metajezik u kojem je moguće jednostavno definirati izgled korisničkog sučelja.

4. Funkcionalni zahtjevi

Dionici :

- Neprijavljeni korisnik
- Prijavljeni korisnik
- Administrator

Aktori i njihovi funkcionalni zahtjevi:

- Neprijavljeni korisnik, inicijator
  - Može se registrirati
  - Može se prijaviti
- Prijavljeni korisnik, inicijator
  - Pregledavanje ponuđenih poslova
  - Stvaranje ponude za posao
  - Uređivanje profila
  - Ocjenjivanje drugih korisnika
  - Uređivanje liste poslova
  - Traženje
  - Filtriranje poslova
  - Otvaranje punog opisa posla
  - Označavanje posla zauzetim
  - Otvaranje sandučića poste
  - Otvaranje poruke
  - Slanje poruke
  - Pregledavanje korisničkih profila
  - Pregledavanje karte poslova
  - Odjava
- Administrator, inicijator

- Prijavljeni korisnik s većim ovlastima
- Brisanje svih poslova
- Brisanje korisničkih profila

- Baza podataka, sudionik
  - Spremanje i brisanje podataka

Opis obrazaca uporabe:

- UC1 – Registracija
  - **Glavni sudionik**: Neprijavljeni korisnik
  - **Cilj**: Registracija korisnika
  - **Sudionici** : Neprijavljeni korisnik, baza podataka
  - **Preduvjeti**: -
  - **Rezultat** : Podaci korisnika dodani su u bazu podataka
  - **Željeni scenarij**:
    1) Korisnik upisuje svoje podatke, te prilaže svoju sliku
    2) Baza podataka sa sustavom obavlja provjeru, te šalje korisniku na verifikaciju
    3) Korisnik obavlja potrebne korake verifikacije
    4) Korisnik je uspješno registriran
  - **Mogući drugi scenarij**: (Registracija postojećeg korisnika)
    1) Korisnik upisuje svoje podatke
    2) Baza podataka obavlja provjeru te utvrđuje da korisnik već postoji
    3) Korisnik biva obaviješten o neuspješnoj registraciji
  - **Mogući drugi scenarij**: (Korisnik ne verificira podatke)
    1) Korisnik upisuje svoje podatke te uploada svoju sliku
    2) Baza podataka sa sustavom obavlja provjeru, te šalje korisniku na verifikaciju
    3) Korisnik ne obavlja potrebne korake verifikacije
    4) Korisnik ne može koristiti ostale funkcionalnosti aplikacije dok se ne obavi verifikacija

- UC2 – Prijava
  - **Glavni sudionik**: Neprijavljeni korisnik
  - **Cilj**: Prijaviti se kako bi korisnik mogao pristupiti ostalom sadržaju aplikacije
  - **Sudionici** : Neprijavljeni korisnik, baza podataka
  - **Preduvjeti**: Korisnik obavio registraciju
  - **Rezultat** : Korisnik može pristupiti ostalom sadržaju aplikacije

- **Željeni scenarij**:
  1) Korisnik upisuje podatke za prijavu
  2) Sustav provjerava postoji li registrirani korisnik s tim podatcima
  3) Korisnik se uspješno prijavio i dobiva status prijavljenog korisnika
- **Mogući drugi scenarij**:
  1) Korisnik upisuje podatke za prijavu
  2) Sustav provjerava postoji li korisnik s tim podatcima
  3) Korisnik je upisao podatke koji se ne nalaze u bazi podataka, te mu sustav to dojavljuje. Korisnik ostaje na aktivnosti za prijavu.

- UC3 – Pregledavanje poslova

  - **Glavni sudionik**: Prijavljeni korisnik
  - **Cilj**: Pregledati popis ponuđenih poslova
  - **Sudionici** : Prijavljeni korisnik, baza podataka
  - **Preduvjeti**: -
  - **Rezultat** : Korisnik je pregledao popis ponuđenih poslova
  - **Željeni scenarij**:
    1) Korisnik pregledava ponuđene poslove na naslovnici
    2) Korisnik pronalazi željeni posao te ga ima mogućnost odabrati

- UC4 – Odjava

  - **Glavni sudionik**: Prijavljeni korisnik
  - **Cilj**: Odjaviti korisnika
  - **Sudionici** : Prijavljeni korisnik
  - **Preduvjeti**: Korisnik je prijavljen
  - **Rezultat** : Korisnik od prijavljenog postaje neprijavljeni korisnik
  - **Željeni scenarij**:
    1) Prijavljeni korisnik pritišće gumb za odjavu
    2) Korisnik postaje neprijavljeni korisnik

- UC5 – Stvaranje poslova

  - **Glavni sudionik**: Prijavljeni korisnik
  - **Cilj**: Dodati u sustav ponudu za posao
  - **Sudionici** : Prijavljeni korisnik, baza podataka
  - **Preduvjeti**: -
  - **Rezultat** : Ponuda za posao dodana u bazu podataka
  - **Željeni scenarij**:

1) Korisnik upisuje podatke potrebne za stvaranje posla
2) Korisnik podnosi zahtjev za dodavanje posla u bazu podataka
3) Posao je uspješno dodan

- **Mogući drugi scenarij**:
  1) Korisnik upisuje podatke potrebne za stvaranje posla
  2) Korisnik podnosi zahtjev za dodavanje posla u bazu podataka
  3) Posao nije dodan jer neki podatci nisu ispravno uneseni
- **Mogući drugi scenarij**:
  1) Korisnik upisuje podatke potrebne za stvaranje posla
  2) Korisnik odustaje od stvaranja posla
  3) Upisani podatci se odbacuju te je stanje u sustavu isto kao i prije

- UC6 – Uređivanje profila

  - **Glavni sudionik**: Prijavljeni korisnik
  - **Cilj**: Promijeniti podatke u sustavu vezane za opis korisnika
  - **Sudionici** : Prijavljeni korisnik, baza podataka
  - **Preduvjeti**: -
  - **Rezultat** : Opis korisnika je promijenjen
  - **Željeni scenarij**:
    1) Korisnik uređuje podatke vezane uz opis svojeg profila
    2) Korisnik sprema promjene
    3) Podatci se ažuriraju
  - **Mogući drugi scenarij**:
    1) Korisnik uređuje podatke vezane uz opis svojeg profila
    2) Korisnik napušta aktivnost
    3) Podatci nisu promijenjeni

- UC7 – Administratorsko brisanje posla

  - **Glavni sudionik**: Administrator
  - **Cilj**: Obrisati posao iz baze podataka
  - **Sudionici** : Administrator, baza podataka
  - **Preduvjeti**: Posao postoji u bazi podataka
  - **Rezultat** : Posao je uklonjen iz baze podataka
  - **Željeni scenarij**:
    1) Administrator podnosi zahtjev za brisanje posla iz baze
    2) Posao se uklanja iz baze podataka

- UC8 – Ocjenjivanje korisnika

- **Glavni sudionik**: Prijavljeni korisnik
- **Cilj**: Promijeniti podatke u sustavu vezane za ocjenu korisnika
- **Sudionici** : Prijavljeni korisnik, baza podataka
- **Preduvjeti**: Korisnik imao poslovnu interakciju s korisnikom kojega ocjenjuje
- **Rezultat** : Ocjena korisnika je ažurirana
- **Željeni scenarij**:
  1) Korisnik označava posao gotovim, te mu se nudi mogućnost ocjenjivanja korisnika
  2) Korisnik daje ocjenu
  3) Podatci se ažuriraju
- **Mogući drugi scenarij**:
  1) Korisnik označava posao gotovim, te mu se nudi mogućnost ocjenjivanja korisnika
  2) Korisnik napušta aktivnost
  3) Podatci nisu promijenjeni

- UC9 – Administratorsko brisanje korisnika

  - **Glavni sudionik**: Administrator
  - **Cilj**: Obrisati korisnika iz baze podataka
  - **Sudionici** : Administrator, baza podataka
  - **Preduvjeti**: Korisnik postoji u bazi
  - **Rezultat** : Korisnik je uklonjen iz baze
  - **Željeni scenarij**:
    1) Administrator podnosi zahtjev za brisanje korisnika iz baze
    2) Korisnik se uklanja iz baze podataka

- UC10 – Upravljanje "Mojim poslovima"

  - **Glavni sudionik**: Prijavljeni korisnik
  - **Cilj**: Urediti ili pregledati korisnikove poslove
  - **Sudionici** : Prijavljeni korisnik, baza podataka
  - **Preduvjeti**: Korisnik ima barem jedan posao u listi poslova
  - **Rezultat** : Korisnik je pregledao svoje poslove, te je potencijalno došlo do izmjena
  - **Željeni scenarij**:
    1) Korisnik pregledava i/ili uređuje statuse poslova tako što aktivne poslove može otkazati ili završiti, a neaktivne samo pregledati

2) Ako je došlo do promjena korisnik potvrđuje akciju, te se ona provodi
3) Podatci se ažuriraju

- **Mogući drugi scenarij**:
  1) Korisnik pregledava i/ili uređuje statuse poslova tako što aktivne poslove može otkazati ili završiti, a neaktivne samo pregledati
  2) Korisnik napušta aktivnost
  3) Podatci ostaju nepromijenjeni

- UC11 – Filtriranje

  - **Glavni sudionik**: Prijavljeni korisnik
  - **Cilj**: Popis poslova se mijenja, te ostaju samo poslovi koji zadovoljavaju kriterije filtera
  - **Sudionici** : Prijavljeni korisnik, baza podataka
  - **Preduvjeti**: -
  - **Rezultat** : Dobiven isfiltrirani popis poslova
  - **Željeni scenarij**:
    1) Korisnik ulazi u opciju filter
    2) Korisnik definira filtere, te šalje zahtjev za primjenu filtera
    3) Korisnik dobiva isfiltrirani popis poslova

- UC12 – Otvori potpuni opis posla

  - **Glavni sudionik**: Prijavljeni korisnik
  - **Cilj**: Pregledati opis posla
  - **Sudionici** : Prijavljeni korisnik, baza podataka
  - **Preduvjeti**: Postoji posao
  - **Rezultat** : -
  - **Željeni scenarij**:
    1) Korisnik otvara posao koji želi detaljnije pregledati

- UC13 – Dodjeljivanje posla

  - **Glavni sudionik**: Prijavljeni korisnik
  - **Cilj**: Dogovoriti posao
  - **Sudionici** : Prijavljeni korisnik, baza podataka
  - **Preduvjeti**: Jedan od prijavljenih korisnika je stvorio posao
  - **Rezultat** : Korisnici su dogovorili posao, posao je maknut iz ponude, te se korisniku ‘posloprimcu’ dogovoreni posao dodao u ‘moji poslovi’

- **Željeni scenarij**:
  1) Korisnik posloprimac, nakon što je pronašao željeni posao, kontaktira korisnika poslodavca putem poruke te započinje komunikacija.
  2) Oba korisnika dolaze do sporazuma
  3) Korisnik poslodavac, dodjeljuje posao korisniku posloprimcu
  4) Korisniku posloprimcu je dodijeljen spomenuti posao.
- **Mogući drugi scenarij**:
  1) Korisnik posloprimac, nakon što je pronašao željeni posao, kontaktira korisnika poslodavca putem poruke te započinje komunikacija
  2) Korisnik poslodavac ne reagira, komunikacija staje

- UC14 – Pregledavanje poštanskog sandučića

  - **Glavni sudionik**: Prijavljeni korisnik
  - **Cilj**: Pregledati sve razgovore
  - **Sudionici** : Prijavljeni korisnik, baza podataka
  - **Preduvjeti**: Korisnik ima poruku
  - **Rezultat** : -
  - **Željeni scenarij**:
    1) Korisnik otvara sandučić i pregledava ga

- UC15 – Slanje poruke

  - **Glavni sudionik**: Prijavljeni korisnik
  - **Cilj**: Poslati poruku drugom korisniku
  - **Sudionici**: Prijavljeni korisnik, baza podataka
  - **Preduvjeti**: Primatelj poruke nalazi se u bazi podataka
  - **Rezultat**: Poruka dodana u primateljev sandučić
  - **Željeni scenarij**:
    1) Korisnik upisuje podatke potrebne za slanje poruke: primatelja i sadržaj poruke
    2) Sustav validira podatke i poruka se dodaje u bazu podataka
  - **Mogući drugi scenarij**:
    1) Korisnik nije ispunio sve podatke potrebne za slanje poruke
    2) Korisnik dobiva obavijest da nije ispunio obavezne podatke ili su oni neispravni

- UC16 – Pregledavanje profila

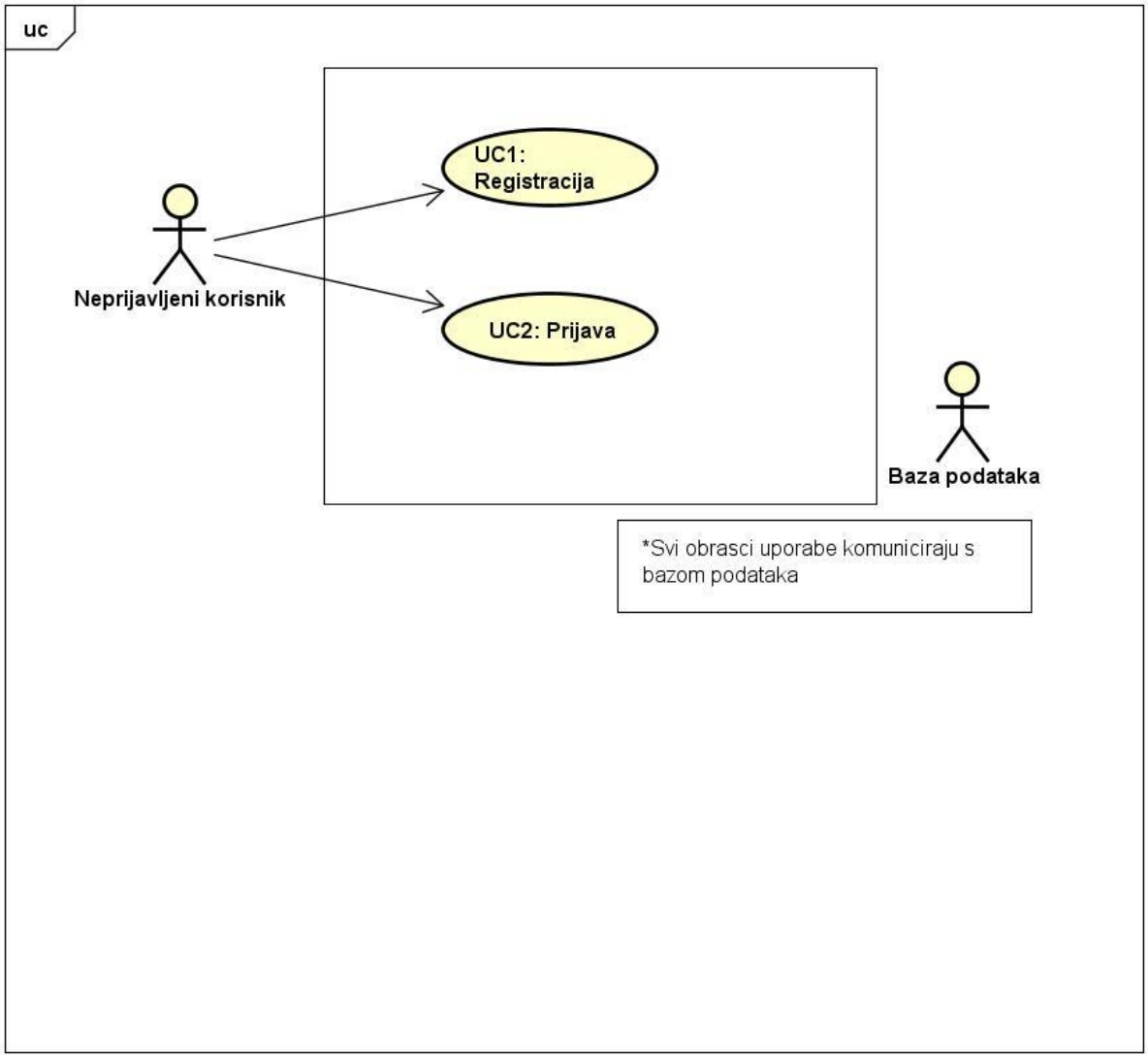
  - **Glavni sudionik**: Prijavljeni korisnik
  - **Cilj**: Pregledati profil korisnika

- **Sudionici** : Prijavljeni korisnik, baza podataka
- **Preduvjeti**: Traženi korisnik postoji
- **Rezultat** : -
- **Željeni scenarij**:
  1) Korisnik otvara i pregledava profil

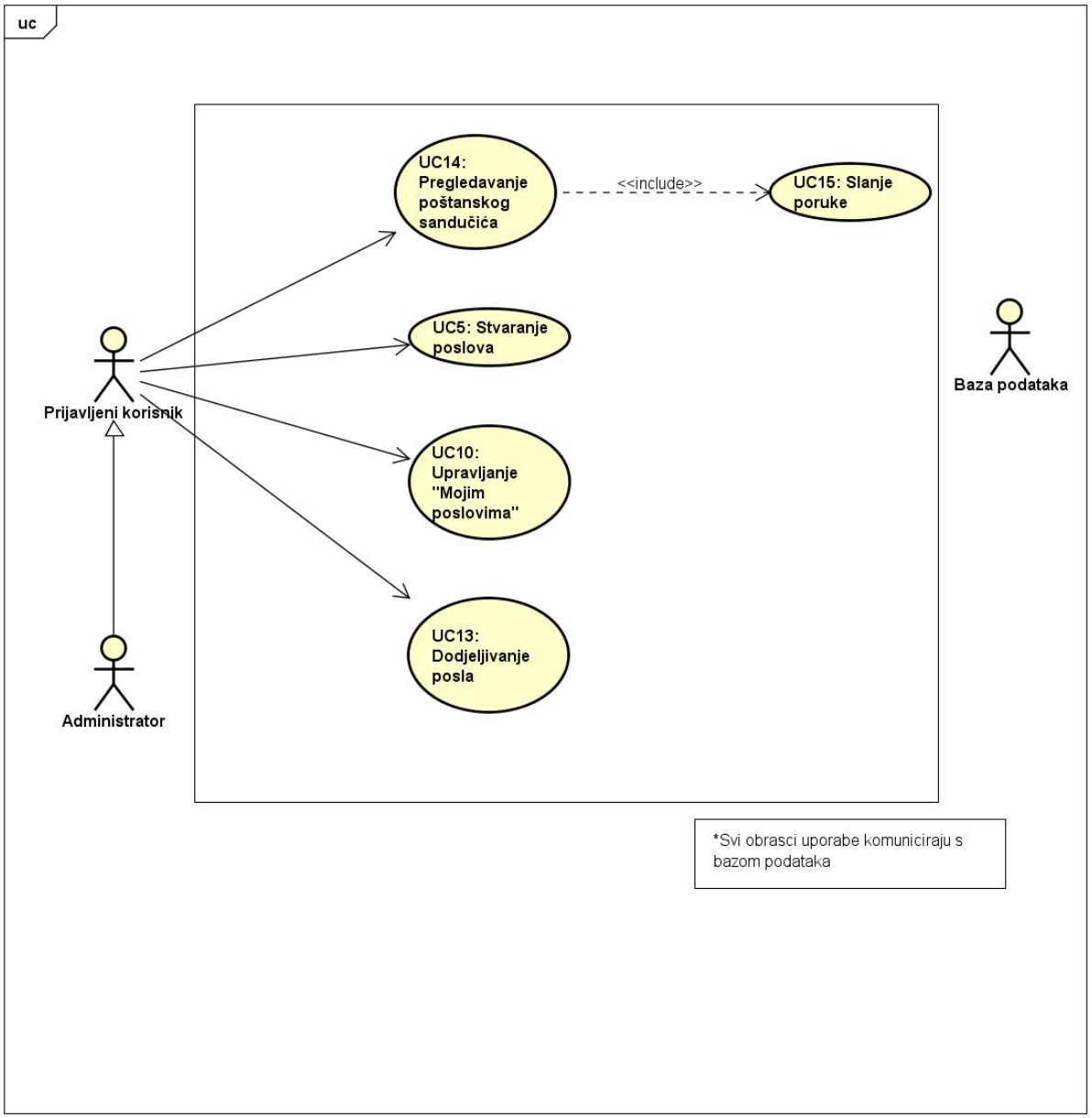
- UC17 – Pregledavanje GPS mape poslova

  - **Glavni sudionik**: Prijavljeni korisnik
  - **Cilj**: Pregledati GPS mapu s poslovima
  - **Sudionici** : Prijavljeni korisnik, baza podataka
  - **Preduvjeti**: -
  - **Rezultat** : -
  - **Željeni scenarij**:
    1) Korisnik otvara GPS mapu poslova i pregledava ponuđene poslove

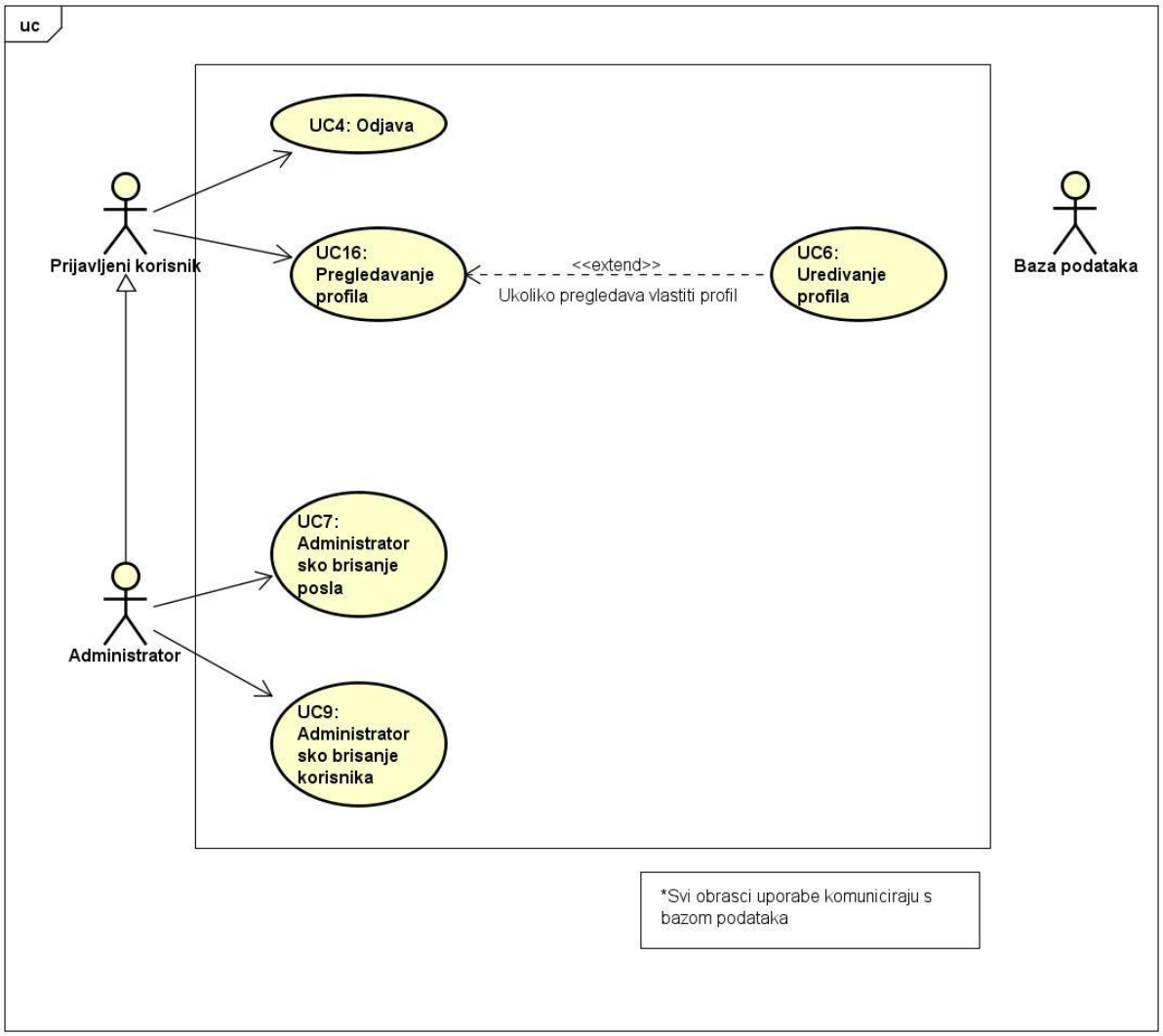
Dijagrami obrazaca uporabe



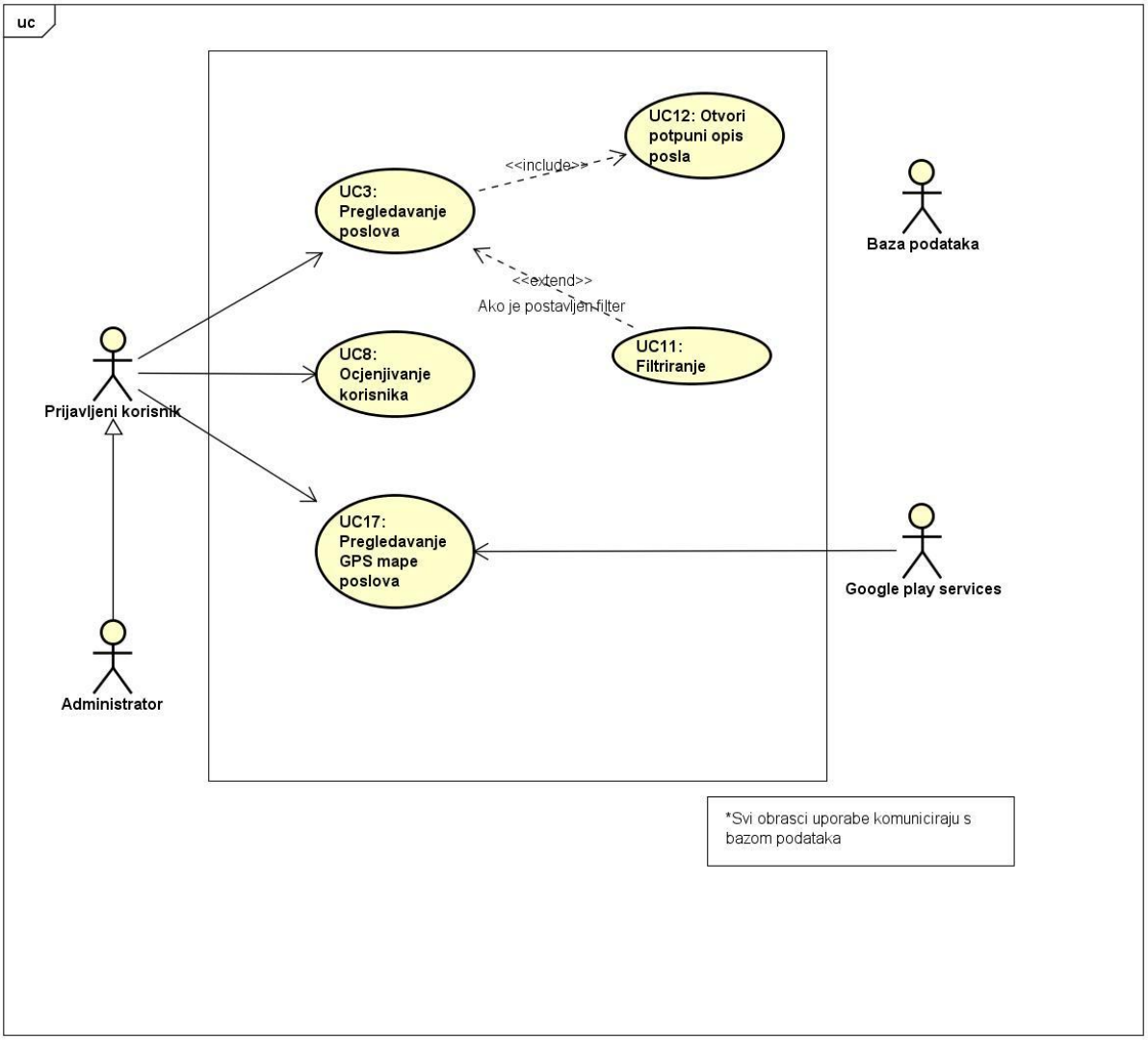
Slika 4.1.1: Dijagram obrazaca uporabe za neprijavljenog korisnika



Slika 4.1.2: Dijagram obrazaca uporabe za prijavljenog korisnika i administratora



Slika 4.1.3: Dijagram obrazaca uporabe za prijavljenog korisnika I administratora

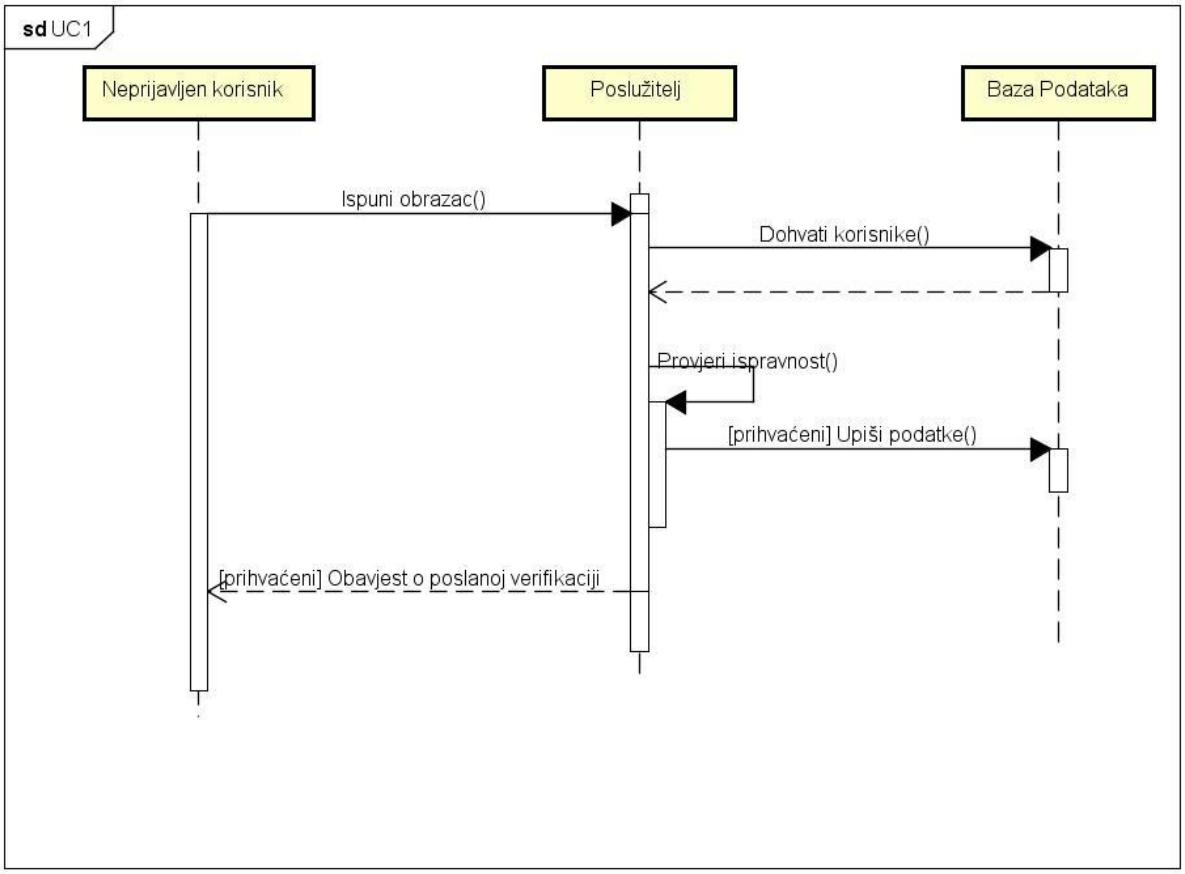


Slika 4.1.4: Dijagram obrazaca uporabe za prijavljenog korisnika I administratora

Sekvencijski dijagrami:

Obrazac uporabe UC1 (Registracija) :

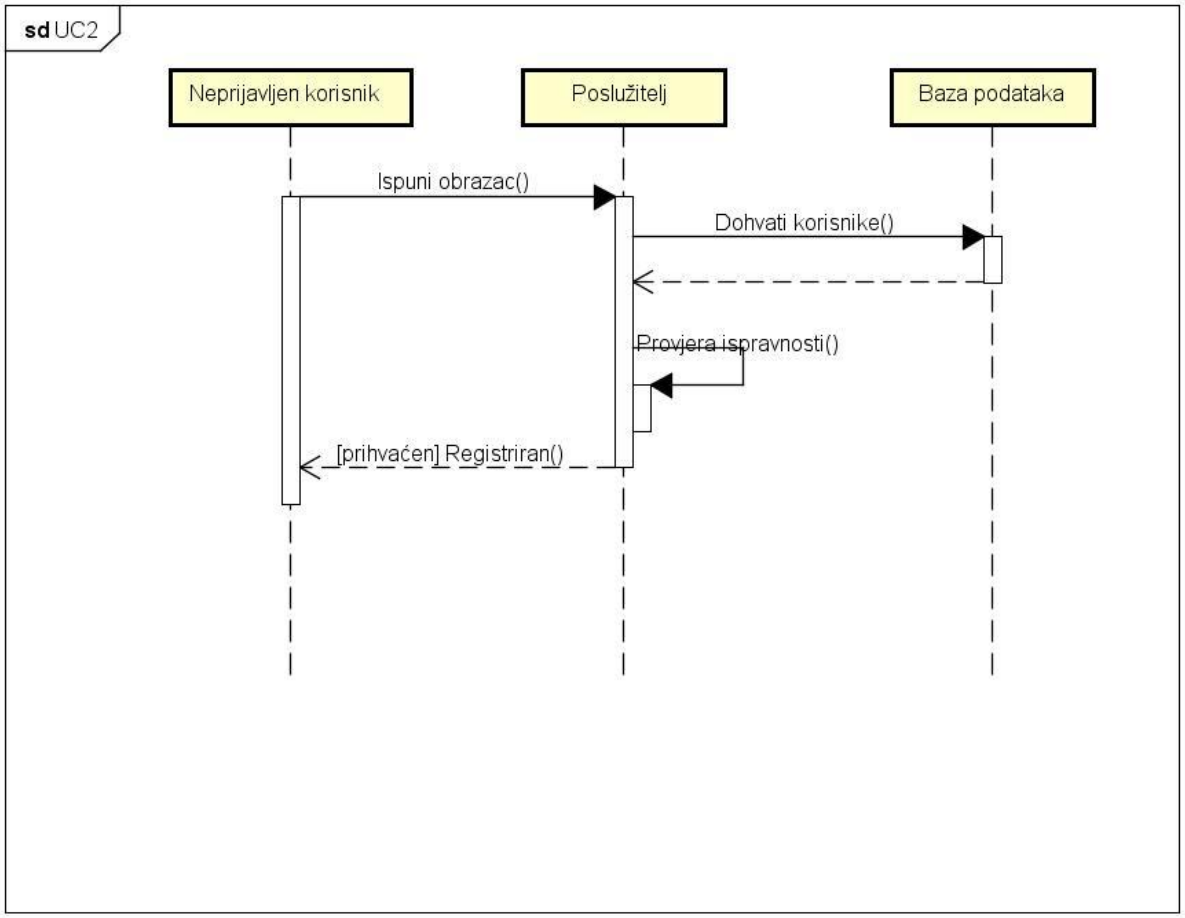
Neprijavljeni korisnik upisuje podatke potrebne za registraciju i dodaje svoju sliku. Sustav provjerava ispravnost podataka, te vraća odgovarajuću poruku. Ako su podatci ispravni korisnik se mora verificirati pomoću električne pošte nakon čega je unesen u bazu podataka. U slučaju da podatci nisu ispravni dojavljena je greška i korisnik može ponoviti proces registracije.



Slika 4.2.1: Sekvencijsku dijagram za registraciju

Obrazac uporabe UC2 (Prijava) :

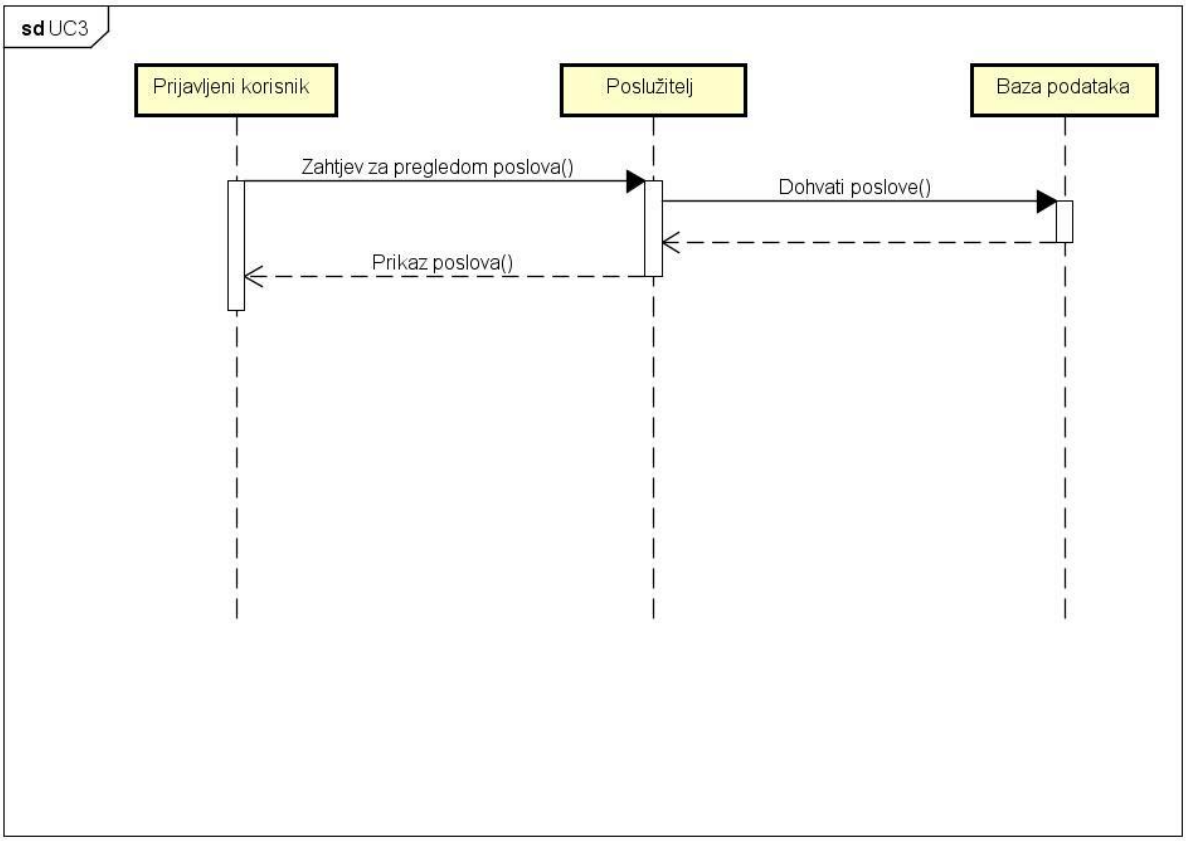
Neprijavljeni korisnik upisuje svoje podatke. Sustav provjerava postoji li korisnik s tim podatcima u bazi podataka. U slučaju da se podatci podudaraju korisnik je uspješno prijavljen, te je unaprijeđen u prijavljenog korisnika. Ako se podatci ne podudaraju korisniku se dojavljuje da prijava nije uspjela, te korisnik može ponoviti proces prijave.



Slika 4.2.2: Sekvencijski dijagram za prijavu

Obrazac uporabe UC3 (Pregledavanje poslova) :

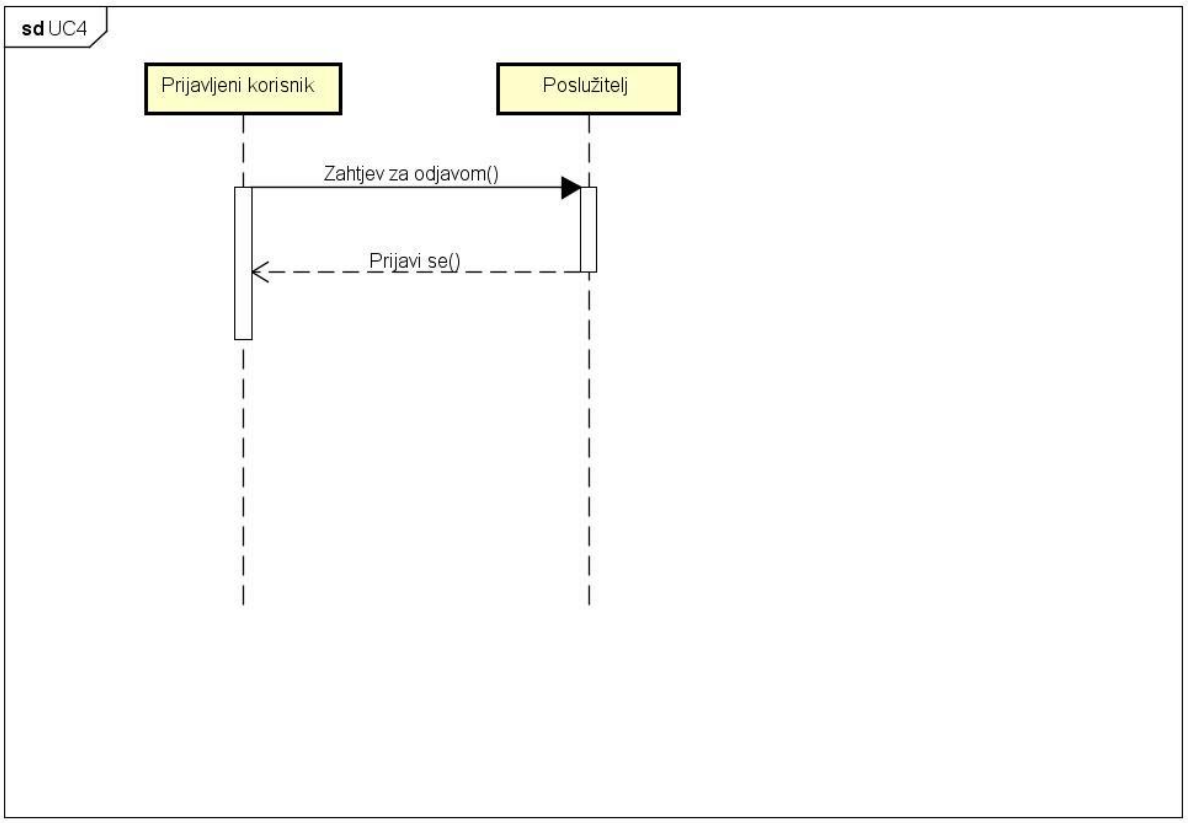
Prijavljeni korisnik šalje zahtjev za pregledavanje poslova i oni se dohvaćaju iz baze podataka. Poslovi se prikazuju na ekranu.



Slika 4.2.3: Sekvencijski dijagram za pregledavanje poslova

Obrazac uporabe UC4 (Odjava) :

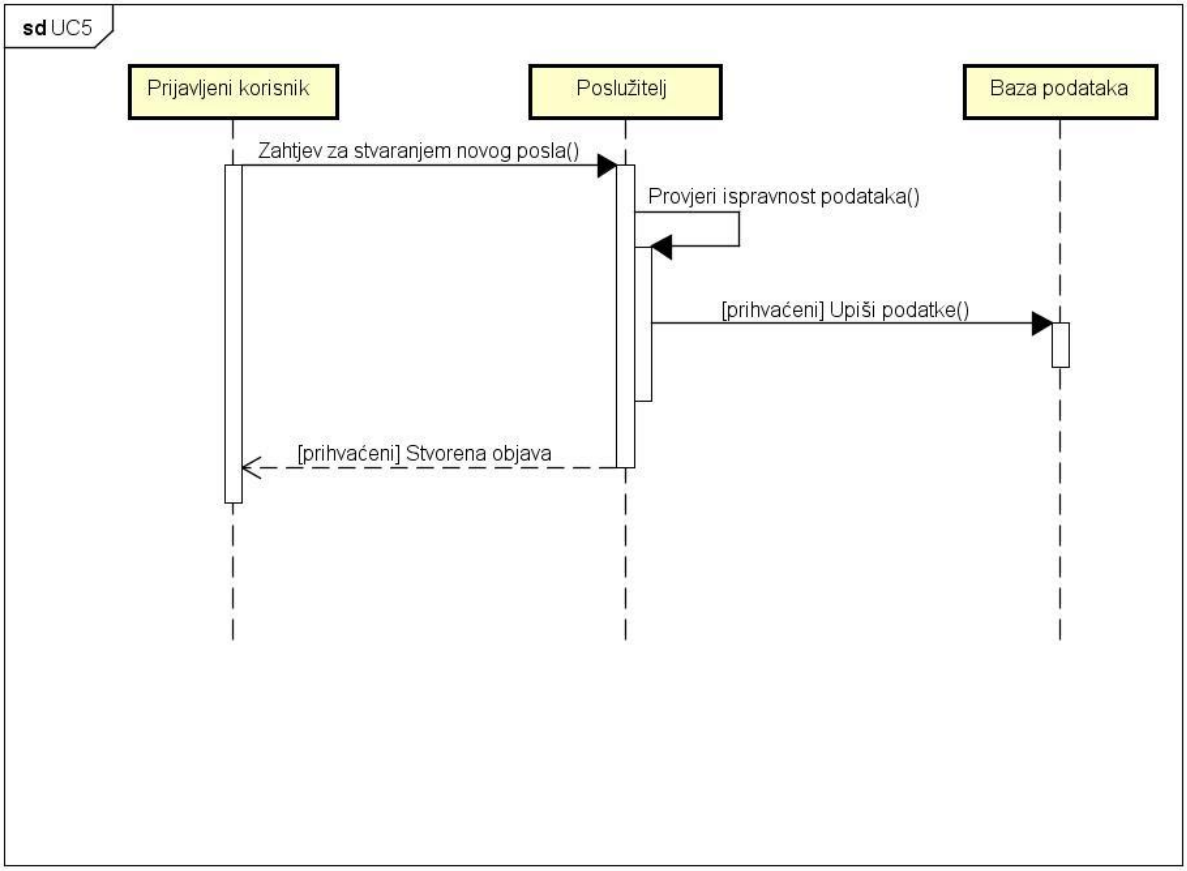
Prijavljeni korisnik podnosi zahtjev za odjavu. Poslužitelj obrađuje zahtjev te odjavljuje korisnika iz sustava. Korisnik je odjavljen, te je na početnom ekranu kao neprijavljeni korisnik.



Slika 4.2.4: Sekvencijski dijagram za odjavu

Obrazac uporabe UC5 (Stvaranje poslova) :

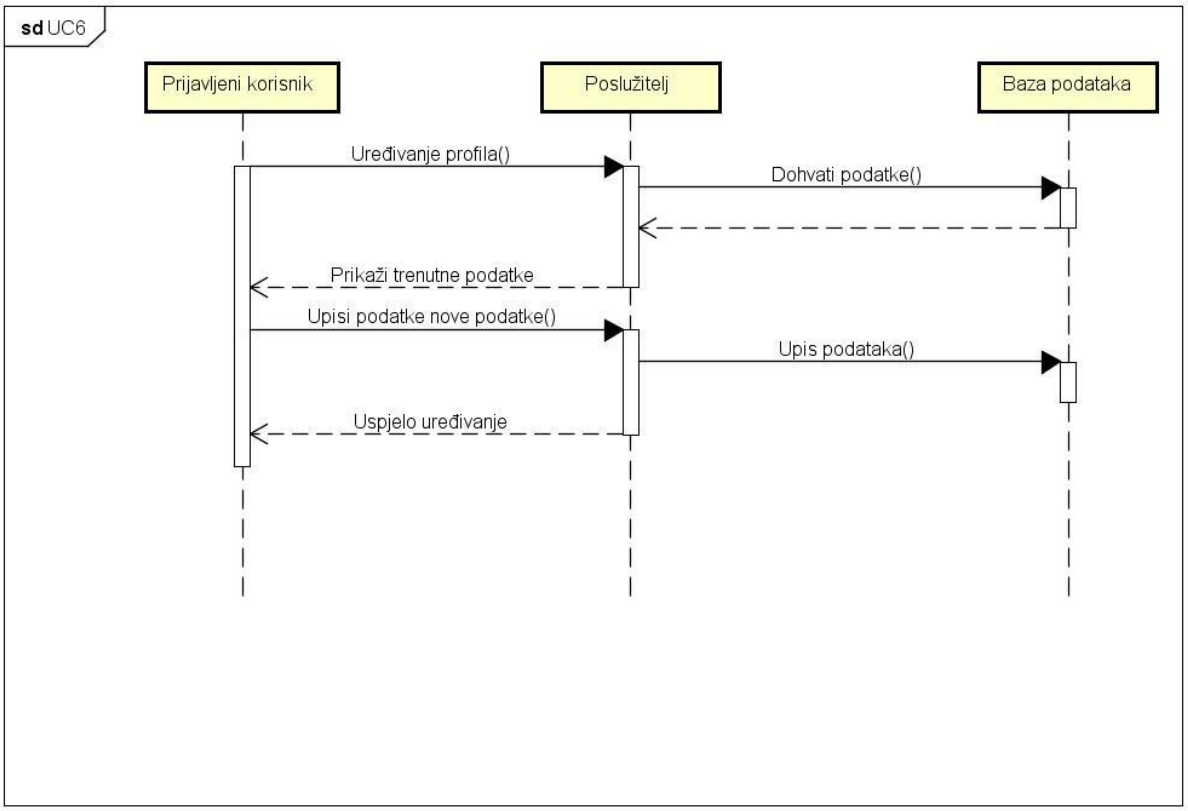
Prijavljeni korisnik podnosi zahtjev za stvaranjem novog posla i upisuje potrebne podatke. Nakon upisivanja potrebnih podataka šalje zahtjev za objavu posla. Sustav provjerava ispravnost podataka, te u slučaju da su podaci ispravni prihvaća zahtjev te upisuje podatke u bazu podataka što označava da je objava stvorena i prihvaćena.



Slika 4.2.5: Sekvencijski dijagram za stvaranje poslova

Obrazac uporabe UC6 (Uređivanje profila) :

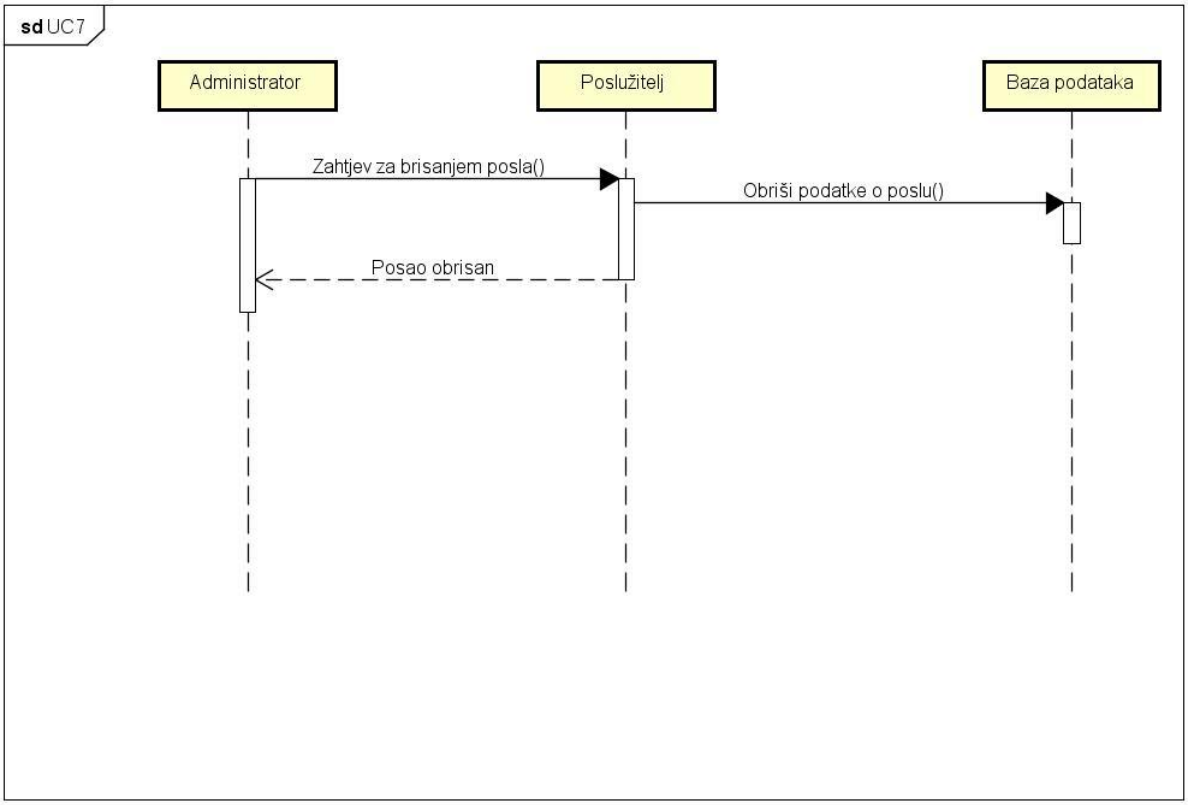
Prijavljeni korisnik odabire opciju uređivanje profila. Poslužitelj dohvaća trenutne podatke te ih prikazuje korisniku. Korisnik nakon toga mijenja, odnosno upisuje nove podatke, podnosi zahtjev za promjenom, odnosno upisom podataka. Podatci se upisuju u bazu podataka te je uređivanje gotovo.



Slika 4.2.6: Sekvencijski dijagram za uređivanje profila

Obrazac uporabe UC7 (Administratorsko brisanje posla) :

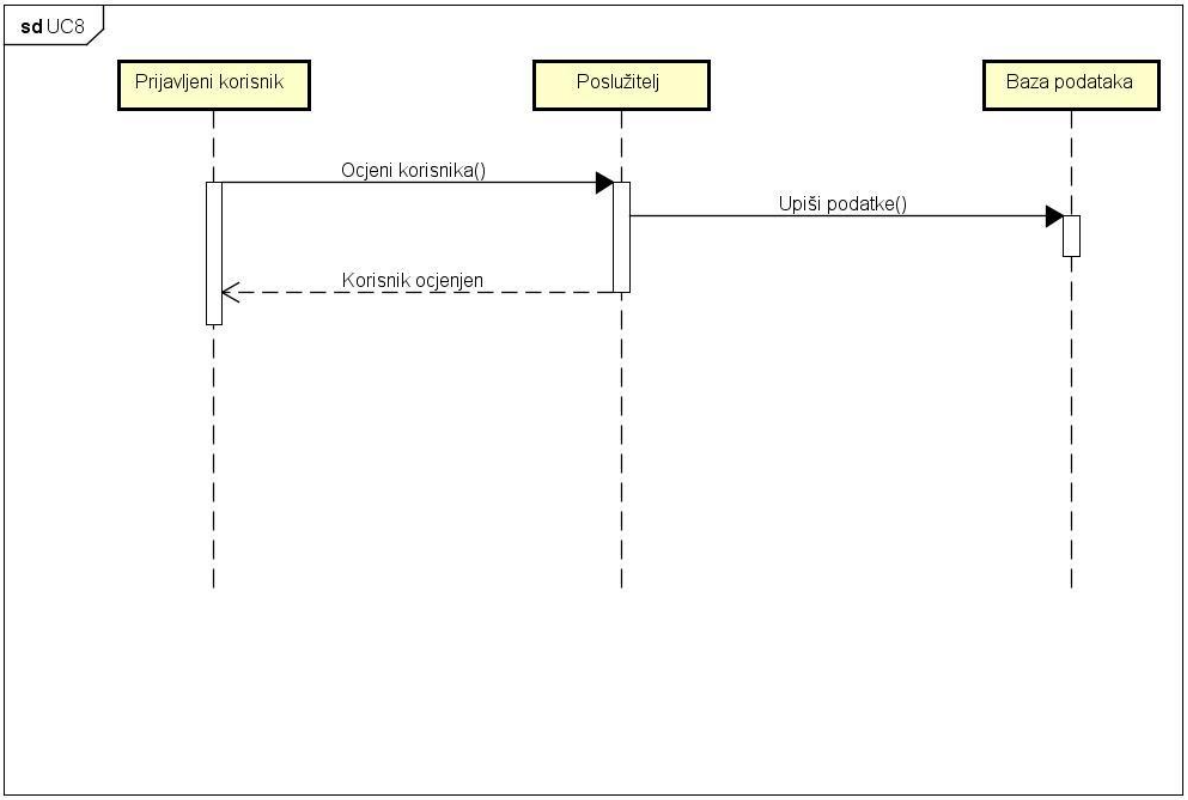
Nakon što je administrator odabrao posao koji se treba izbrisati, šalje zahtjev za brisanjem posla, baza podataka briše podatke, odnosno posao, te je u tom trenutku posao obrisan.



Slika 4.2.7: Sekvencijski dijagram za administratorsko brisanje poslova

Obrazac uporabe UC8 (Ocjenjivanje korisnika) :

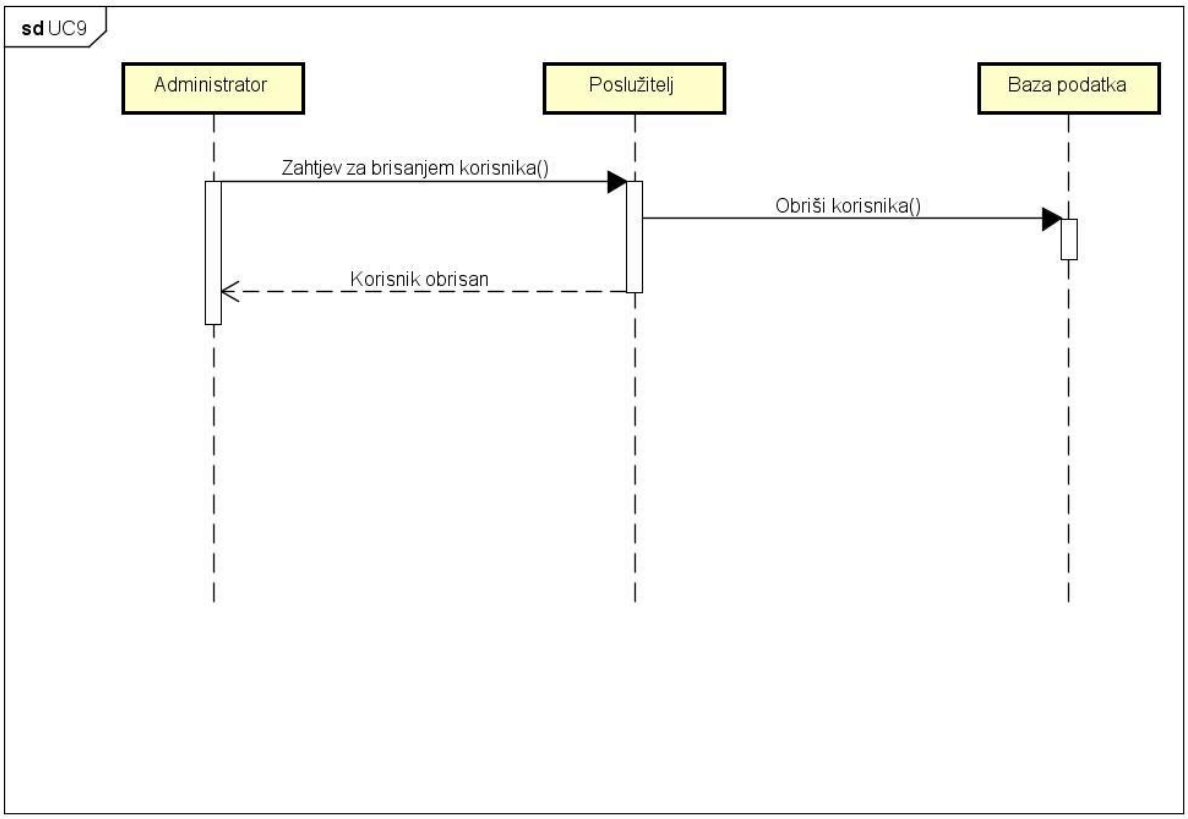
Nakon odrađenog posla, korisnik treba ocijeniti korisnika, nakon što je korisnik unio ocjenu, ona se upisuje u bazu podataka, te se mijenja ocjena korisnika, odnosno drugi korisnik je ocijenjen.



Slika 4.2.8: Sekvencijski dijagram za ocjenjivanje korisnika

Obrazac uporabe UC9 (Administratorsko brisanje korisnika) :

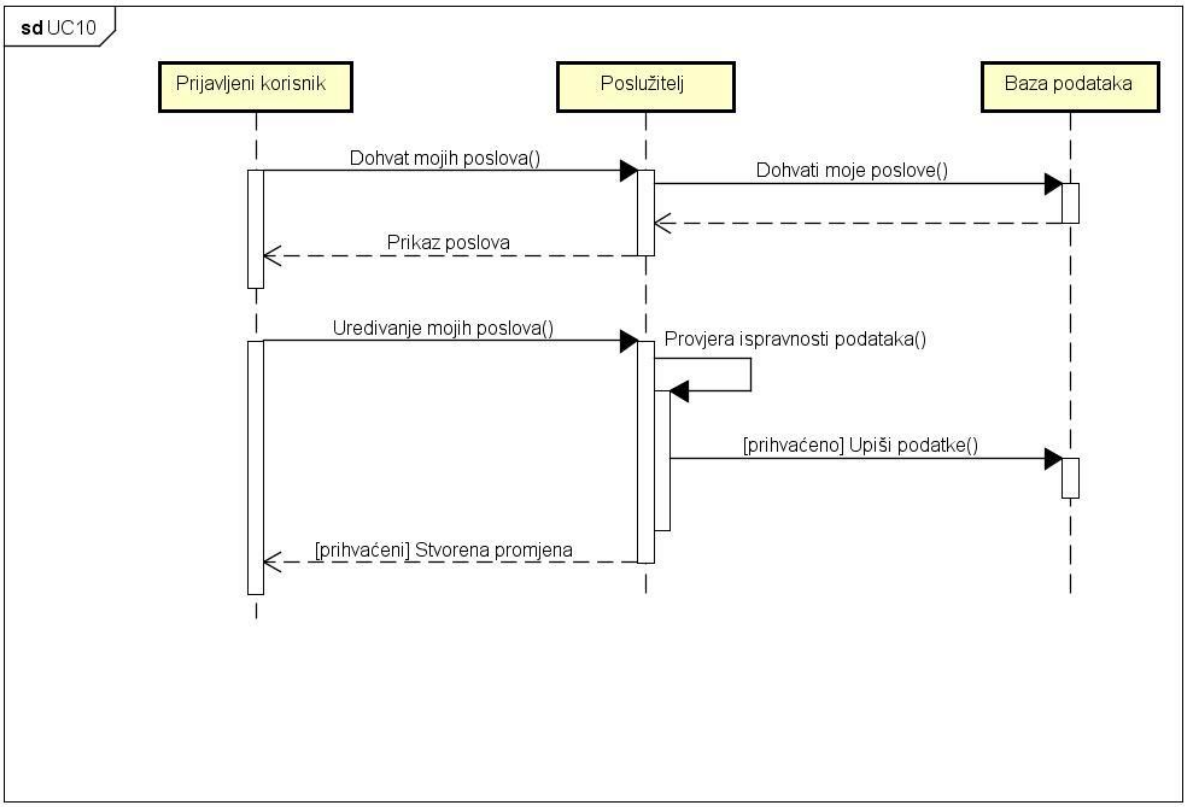
Nakon što je administrator odabrao korisnika kojeg se treba izbrisati, šalje zahtjev za brisanje korisnika, baza podataka briše podatke, odnosno korisnika, te je u tom korisnik posao obrisan.



Slika 4.2.9: Sekvencijski dijagram za administratorsko brisanje korisnika

Obrazac uporabe UC10 (Upravljanje „Mojim poslovima”) :

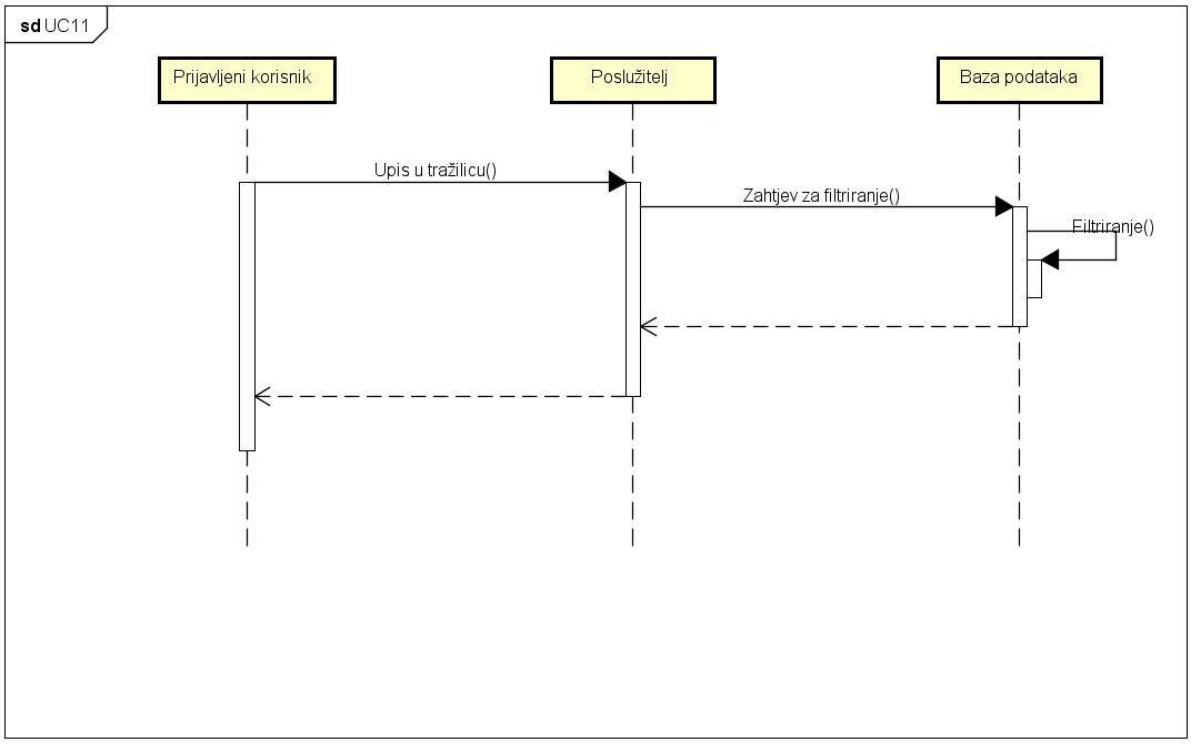
Prijavljeni korisnik dohvaća listu poslova te dobiva prikaz poslova. Nakon toga, prijavljeni korisnik ima opciju uređivanja odabranih poslova, izmjena se po potrebi provjerava te se unutar baze podataka osvježava stanje.



Slika 4.2.10: Sekvencijski dijagram za upravljanje “Mojim poslovima”

Obrazac uporabe UC11 (Filtriranje) :

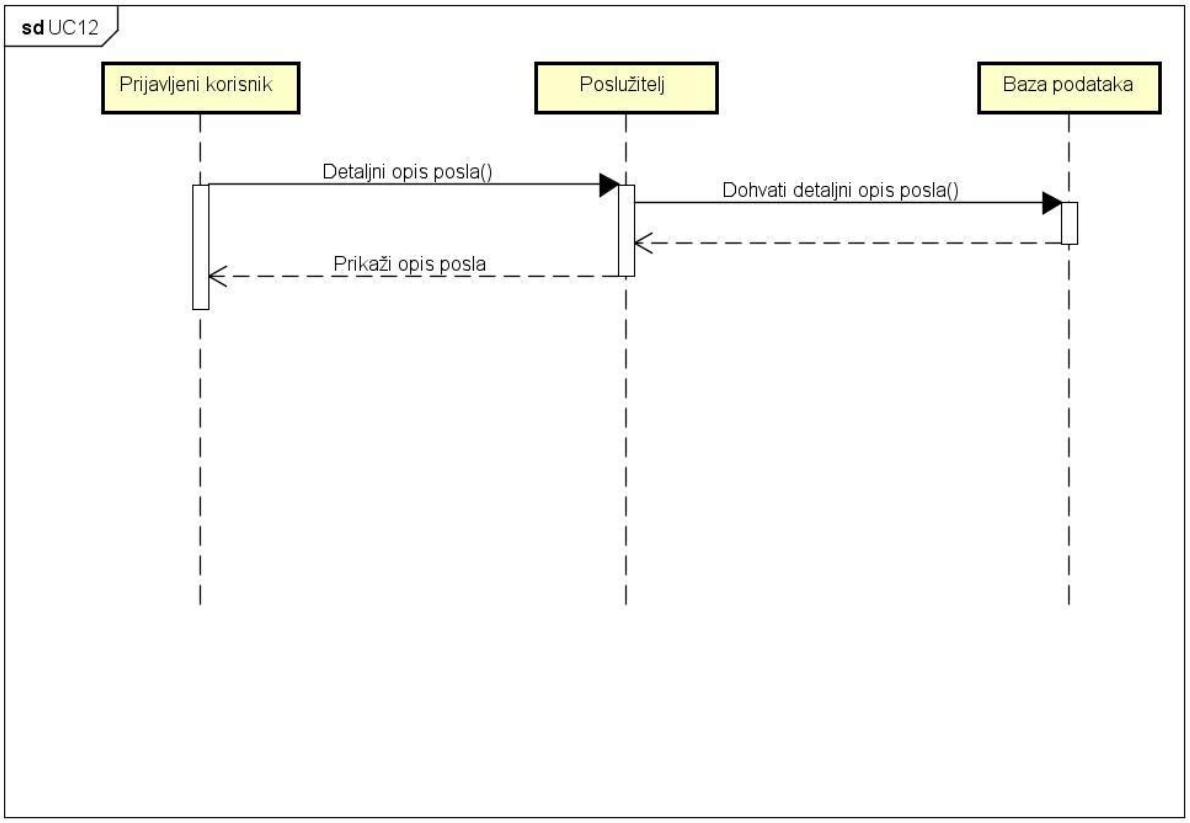
Prijavljeni korisnik definira filtere koje želi primijeniti te pošalje zahtjev za filtracijom. Sustav zatim primjeni korisnikov filter, te mu vrati isfiltrirani popis poslova.



Slika 4.2.11: Sekvencijski dijagram za filtriranje poslova

Obrazac uporabe UC12 (Otvori potpuni opis posla) :

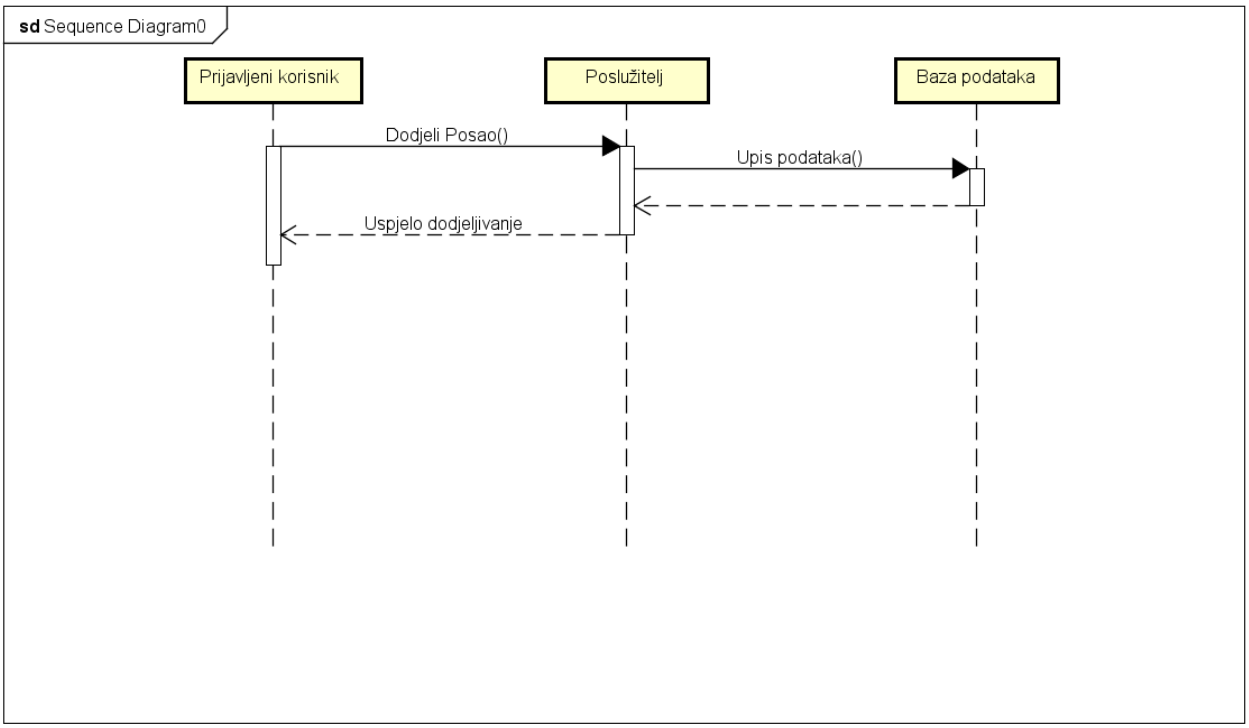
Odabirom nekog posla poslužitelj traži od baze podataka detaljni opis posla, te se korisniku taj opis prikazuje.



Slika 4.2.12: Sekvencijski dijagram za otvaranje detaljnog opisa posla

Obrazac uporabe UC13 (Dodjeljivanje posla) :

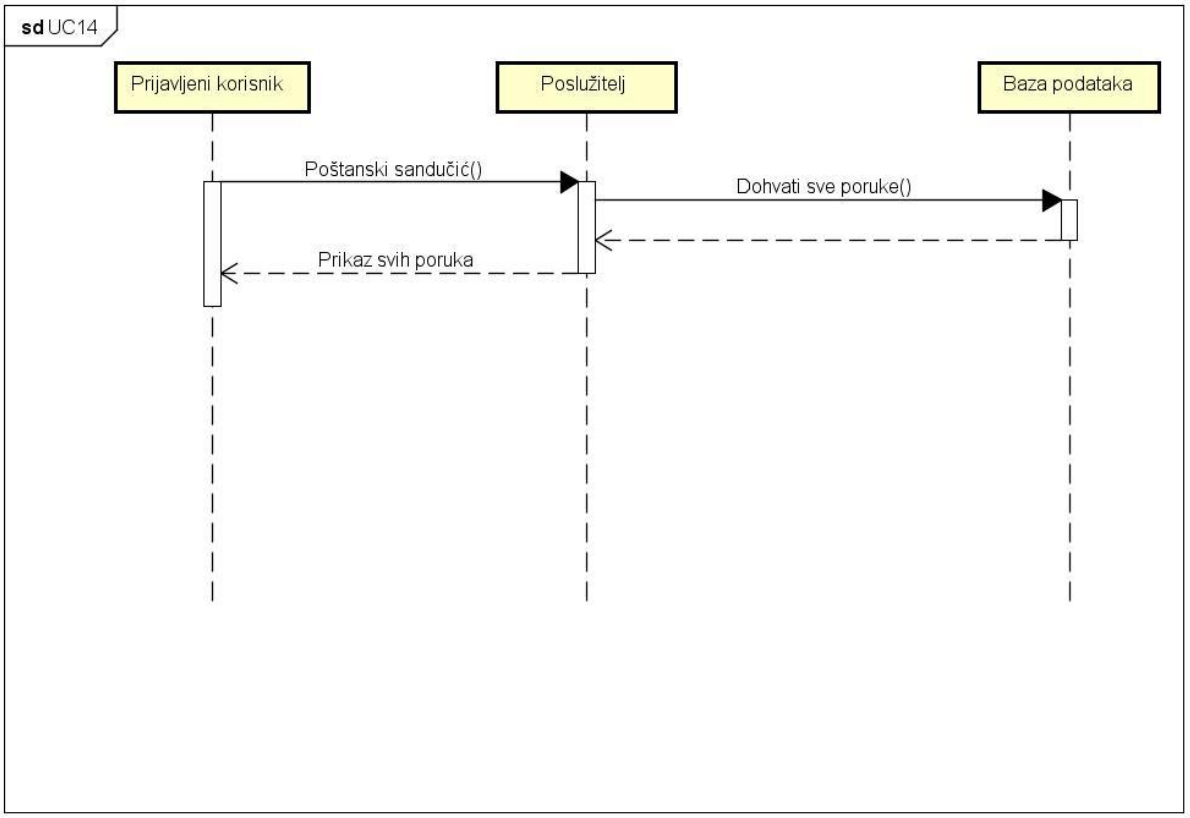
Dva prijavljena korisnika dogovaraju posao izmjenjivanjem poruka. Nakon što se postigao dogovor, korisnik poslodavac dodjeljuje posao korisniku posloprimcu tako što upiše njegov email u za to predviđeno područje. Time se ažuriraju podatci u bazi podataka i posao se smatra dodijeljenim korisniku posloprimcu.



Slika 4.2.13: Sekvencijski dijagram za dodjeljivanje posla

Obrazac uporabe UC14 (Pregledavanje poštanskog sandučića) :

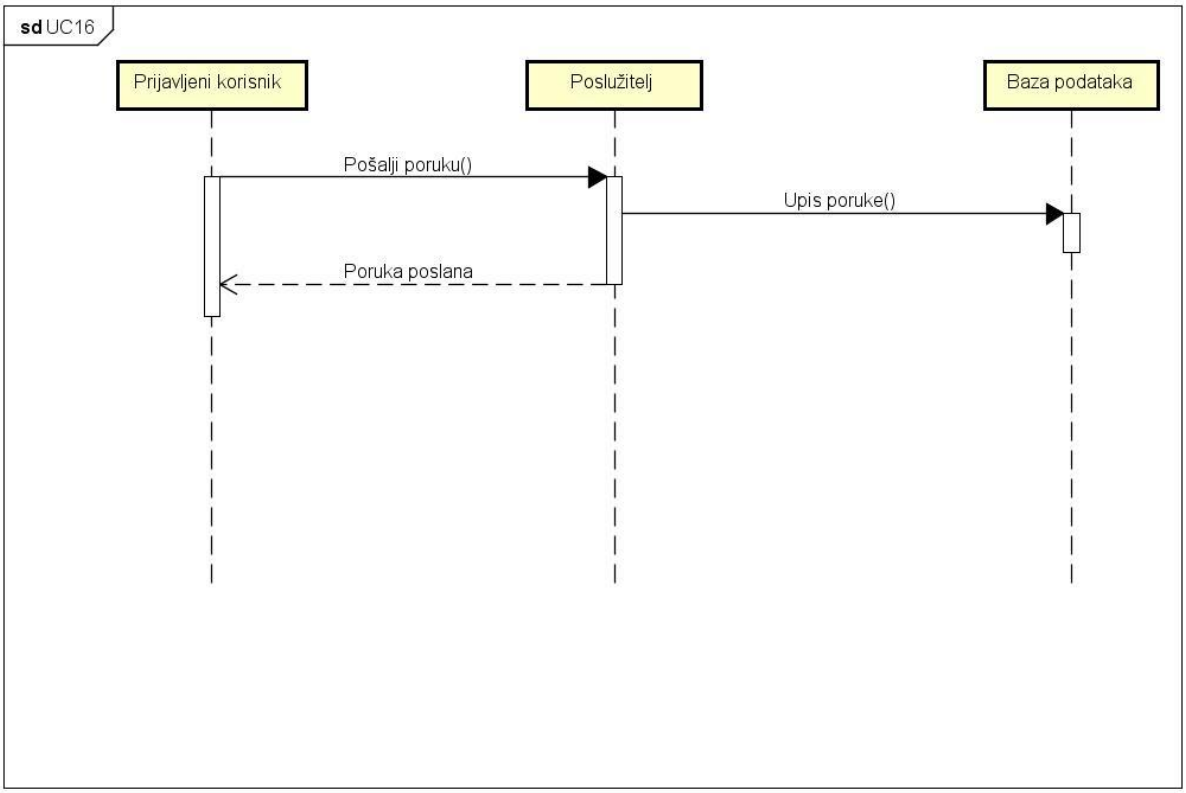
Dolaskom na poštanski sandučić poslužitelj traži od baze podataka dohvaćanje svih poruka od korisnika, te se korisniku te poruke prikazuju.



Slika 4.2.14: Sekvencijski dijagram za pregledavanje poštanskog sandučića

Obrazac uporabe UC15 (Slanje poruke) :

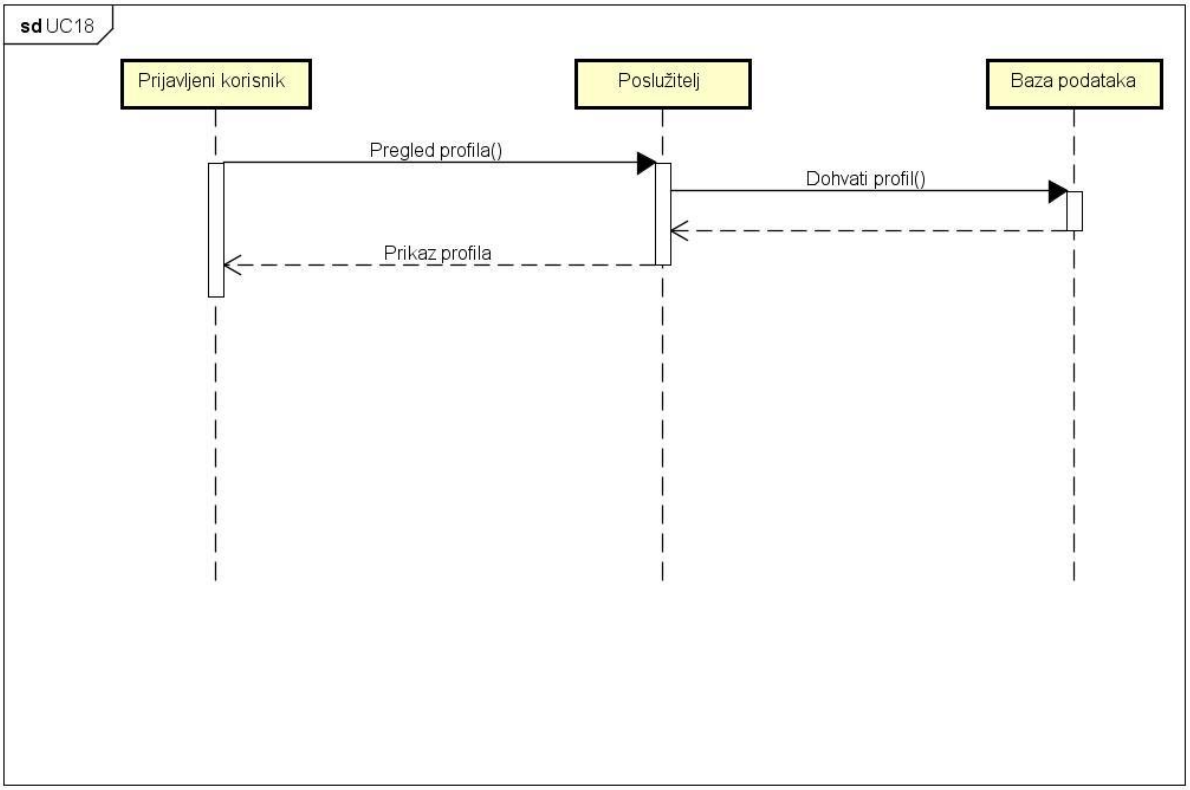
Prijavljeni korisnik, nakon što je sastavio poruku, šalje zahtjev ‘Pošalji poruku’, u bazu podataka se upisuje sadržaj poruke, te je poruka poslana, odnosno vidljiva drugom korisniku.



Slika 4.2.15: Sekvencijski dijagram za slanje poruke

Obrazac uporabe UC16 (Pregledavanje profila) :

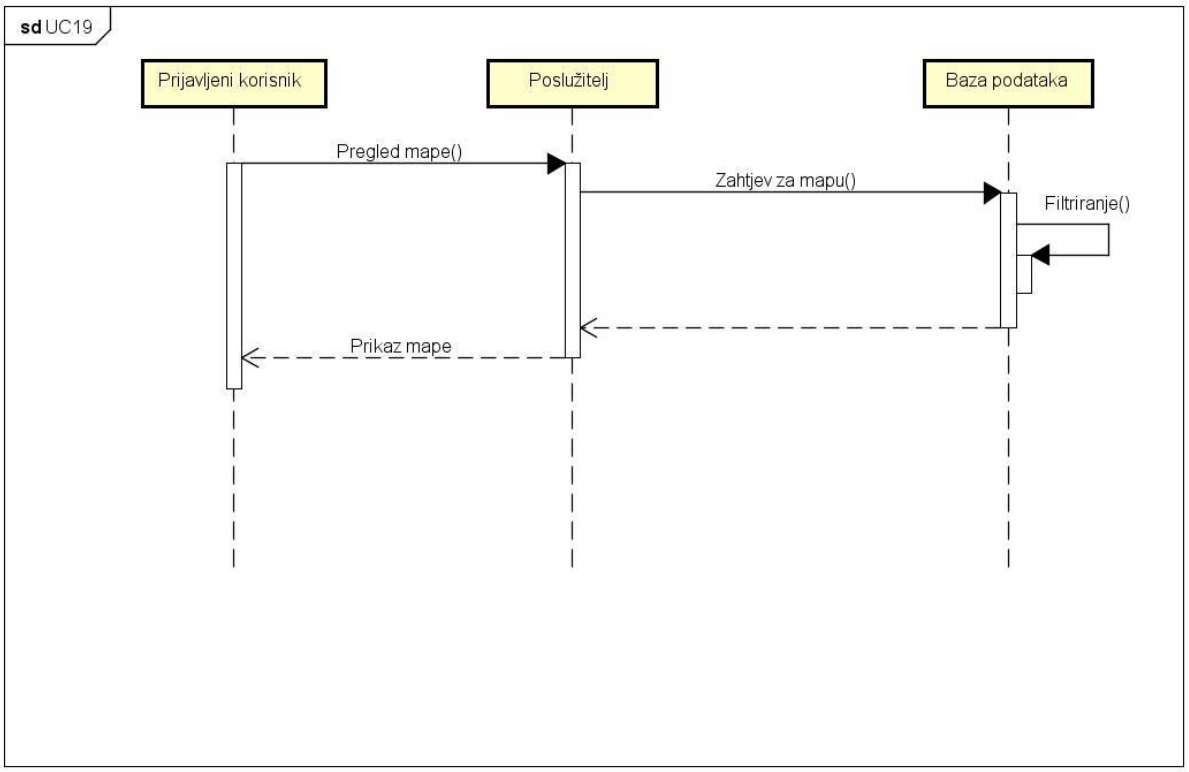
Prijavljeni korisnik šalje zahtjev poslužitelju za pregled određenog profila, poslužitelj zatim traži bazu podataka da mu preda podatke o profilu kojeg je korisnik zatražio, te podatke odnosno profil, prikazuje korisniku.



Slika 4.2.16: Sekvencijski dijagram za pregledavanje profila

Obrazac uporabe UC17 (Pregledavanje GPS mape poslova) :

Prijavljeni korisnik prilikom otvaranja GPS mape, šalje poslužitelju zahtjev za prikazom mape, odnosno poslova na mapi, baza podataka vraća popis poslova, te se korisniku prikazuje mapa s pinovima koji označavaju poslove.



Slika 4.2.17: Sekvencijski dijagram za pregledavanje GPS mape poslova

5. Ostali zahtjevi

- Neispravno korištenje aplikacije ne smije srušiti aplikaciju ili bazu podataka
- Omogućen rad s više korisnika u isto vrijeme
- Svaki pristup bazi podataka iz aplikacije mora biti kraći od nekoliko sekundi
- Osim filtriranja poslova po svojstvima posla, mora postojati mogućnost izravnog pretraživanja kroz koju se mogu pretraživati i korisnici i poslovi.
- Aplikacija mora podržavati hrvatsku abecedu
- Korisničko sučelje mora biti intuitivno kako bi ga mogle koristiti i starije osobe koje nemaju puno iskustva s tehnologijom
- Nadogradnja sustava mora biti jednostavna i ne smije ugroziti rad aplikacije
- Za potrebe testiranja, baza mora sadržavati barem po jednog korisnika od svake vrste (tri), gdje jedan korisnik mora imati po jedan posao od svake vrste (odrađeni, u tijeku i postavljeni) i nekoliko poruka s drugim korisnicima. Također, mora postojati barem deset poslova kako bi se moglo testirati filtriranje i prikaz na GPS karti.

6. Arhitektura i dizajn sustava

6.1. Svrha, opći prioriteti i skica sustava

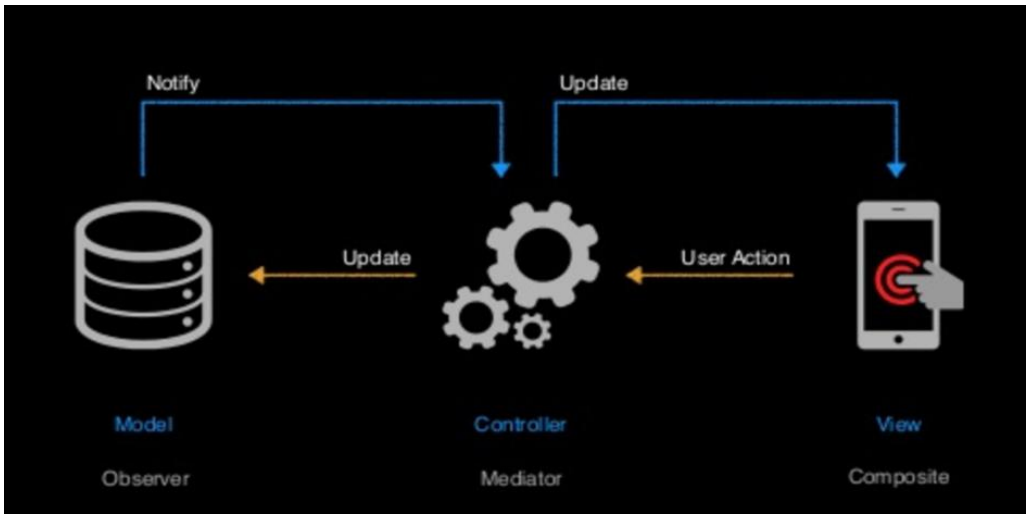
ARHITEKTURA I DIZAJN

Arhitektura programske potpore je jedan od najbitnijih koraka u oblikovanju programske potpore. Osnovna uloga arhitekture programske potpore je apstrakcija sustava na visokom nivou. Arhitektura programske potpore mora strukturirati projekt i samu programsku podršku. Izrađuje se prije detaljne specifikacije i osnova je za komunikaciju dionika. Prednosti definiranja arhitekture su što smanjuje cijenu oblikovanja, razvoja i održavanja programskog proizvoda.

S obzirom na to da je naš zadatak napraviti mobilnu aplikaciju koja ima bazu podataka na serveru, odlučili smo se za MVC (engl. Model-View-Controller) arhitekturu. MVC arhitektura je prilagođena objektno orijentiranoj paradigmi pa time dodatno olakšavamo oblikovanje i održavanje programskog proizvoda.

Arhitektura MVC se sastoji od 3 dijela:

1) Model - služi za dohvaćanje, umetanje i izmjenu podataka u bazi
2) View (hrv. *pogled*) - služi za prikaz informacija koje korisnik može vidjeti. U našem slučaju se pogled odnosi na xml datoteke koje oblikuju izgled mobilne aplikacije
3) Controller (hrv. *upravljač*) – sloj u kojem se nalazi cijela logika sustava. On služi kao posrednik između pogleda i modela. Ovaj sloj se u mobilnim aplikacijama implementira kroz posebne razrede – aktivnosti.



Slika 6.1.1: Opis načina rada MVC-a

BAZA PODATAKA

Za potrebe našeg sustave i mobilne aplikacije koristi ćemo relacijsku bazu podataka čija struktura uvelike olakšava modeliranje događaja i entiteta iz stvarnog svijeta i njihovih podataka. Kvant relacijske baze podataka je relacija, odnosno tablica koja je opisana svojim imenom i skupom pripadajućih atributa. Sve relacije u bazi su svedene na 3. normalnu formu stoga u bazi nemamo redundantnih podataka.

Slijedi prikaz i opis svih relacija i njihovih atributa te pripadajućih primarnih i stranih ključeva:

Korisnik

- korisnikID: VARCHAR(255) - autogenerirani primarni ključ korisnika
- ime: VARCHAR(255) - ime korisnika
- prezime: VARCHAR(255) - prezime korisnika
- email: VARCHAR(255) - email korisnika
- zaporkaHASH: VARCHAR(255) - heširana zaporka korisnika
- dob: SMALLINT(127) - dob korisnika
- slikaID: VARCHAR(255) - ključ slike spremljen u drugoj tablici
- opis: VARCHAR(255) - opis korisnika
- datumRegistracije: TIMESTAMP - datum kada se korisnik prvi puta registrirao
- jeValidiran: BOOLEAN - je li korisnik verificirao svoj korisnički račun
- brojTelefona: VARCHAR(15) - telefonski broj korisnika

PK {korisnikID**}**
K {email**}**
FK {slikaID**}**

Entitet "Korisnik" opisuje svakog korisnika aplikacije. Entitet "Korisnik" ima 2 jedinstvena ključa: korisnikID i email. KorisnikID je primarni ključ entiteta i generira se pri registraciji svakog korisnika. Svaki Entitet "Korisnik" sadrži osnovne podatke o korisniku: ime, prezime, email, dob, opis, datumRegistracije, brojTelefona, je li korisnik verificirao svoj račun. U bazi podata se ne pamti zaporka nego HASH zaporke. SlikaID je strani ključ entiteta "Slika".

Poruka

- porukaID: VARCHAR(255) - autogenerirani primarni ključ poruke
- posiljateljID: VARCHAR(255) - korisnikID korisnika koji je poslao poruku
- primateljID: VARCHAR(255) - korisnikID korisnika koji je primio poruku
- sadrzaj: VARCHAR(255) - sadržaj poruke
- datum: TIMESTAMP - datum kada je poruka poslana

PK {porukaID**}**
FK {primateljID,posiljateljID**}**

Entitet "Poruka" opisuje svaku poslanu poruku. Svaka poruka ima jedinstveni ključ porukaID. PorukaID se generira prilikom slanja svake poruke. Poruka ima sadržaj i datum kada je poslana. Svaka poruka ima korisnikID posiljatelja i primatelja iste.

Slika

- slikaID**:** VARCHAR(255) - autogenerirani primarni ključ slike
- slikaBLOB: BLOB - binarni objekt koji predstavlja sliku

PK {slikaID**}**

Entitet "Slika" predstavlja sliku. Svaka slika ima jedinstveni ključ slikaID. Entitet "Slika" ima i atribut slikaBLOB koji predstavlja binarni objekt slike.

KorisnikStatistika

- korisnikID: VARCHAR(255)-kljuc korisnika
- ukBrojPoslovaPonudenih: INT - broj poslova koje je korisnik ponudio
- ukBrojPoslovaOdradenih: INT - broj poslova koje je korisnik odradio
- ukNovacPotrosen: INT - ukupna količina novaca koju je korisnik potrošio
- ukNovacZaraden: INT - ukupna količina novaca koju je korisnik zaradio
- prosjecnaOcjena: Double - prosječna ocjena korisnika

PK{korisnikID**}**

Za svakog korisnika se vodi njegova statistika. Za svakog korisnika se pamti ukupan broj poslova koje je taj korisnik ponudio, ukupan broj poslova koje je taj korisnik odradio, ukupna kolicna novca koju je taj korisnik potrošio, ukupna količina novca koju je taj korisnik zaradio i prosječna ocjena korisnika.

Posao

- posaoID: VARCHAR(255) - autogenerirani primarni kljuc posla
- naslov: VARCHAR(255) - naslov posla
- opis: VARCHAR(1023) - opis posla
- lokacija: VARCHAR(255) - lokacija gdje će se posao održavati
- vrijeme: TIMESTAMP(255) - vrijeme kada će se posao održati
- trajanje: LONG - okvirno vrijeme potrebno za odrađivanje posla
- ponudeniNovac: INT - novac ponuđen za odrađivanje posla
- posaoGotov: BOOLEAN - zastavica je li posao odrađen
- kategorijaID: VARCHAR(255) - kljuc za kategoriju u koju spada posao

PK{posaoID**}**
FK{kategorijaID**}**

Entitet "Posao" opisuje svaki posao. Entitet PosaoID je primarni ključ entiteta i generira se pri izradi svakog posla. Svaki posao ima naslov,opis posla,lokaciju gdje se održava,vrijeme kad će se posao odraditi, okvirno vrijeme trajanja posla, ponuđeni novac za odradu posla, zastavicu je li posao već odrađen i kategoriju u koju posao spada.

Kategorija

- kategorijaID: VARCHAR(255) - primarni ključ kategorije
- naziv: VARCHAR(255) - naziv kategorije
- opis: VARCHAR(255) - opširan opis što sve kategorija podrazumijeva

PK{kategorijaID**}**

Entitet "Kategorija" opisuje svaku kategoriju koju posao može imati. Svaka kategorija ima svoj naziv i svoj opis.

Korisnik-Posao

- posaoID: VARCHAR(255) - posaoID posla
- poslodavacID: VARCHAR(255) - korisnikD korisnika koji je ponudio posao
- posloprimacID: VARCHAR(255) - korisnikID korisnika koji je odradio posao
- sklopljenPosaoTrenutak: TIMESTAMP - trenutak kada su poslodavac i posloprimac sklopili posao
- odrađenPosaoTrenutak: TIMESTAMP - trenutak kada je posao odrađen

PK {PosaoID**}**
FK {poslodavacID,posloprimacID**}**

Entitet "Korisnik-Posao" povezuje korisnike (posloprimce i poslodavce s poslom) i pamti kada je posao sklopljen između poslodavca i posloprimca i kada je posao odrađen.

Ocjena

- korisnikDaoID: VARCHAR(255)
- korisnikPrimioID: VARCHAR(255)
- posaoID: VARCHAR(255)
- ocjena: SMALLINT(5)
- datumOcijenjen: TIMESTAMP

PK{korisnikDaoID,korisnikPrimioID,posaoID**}**

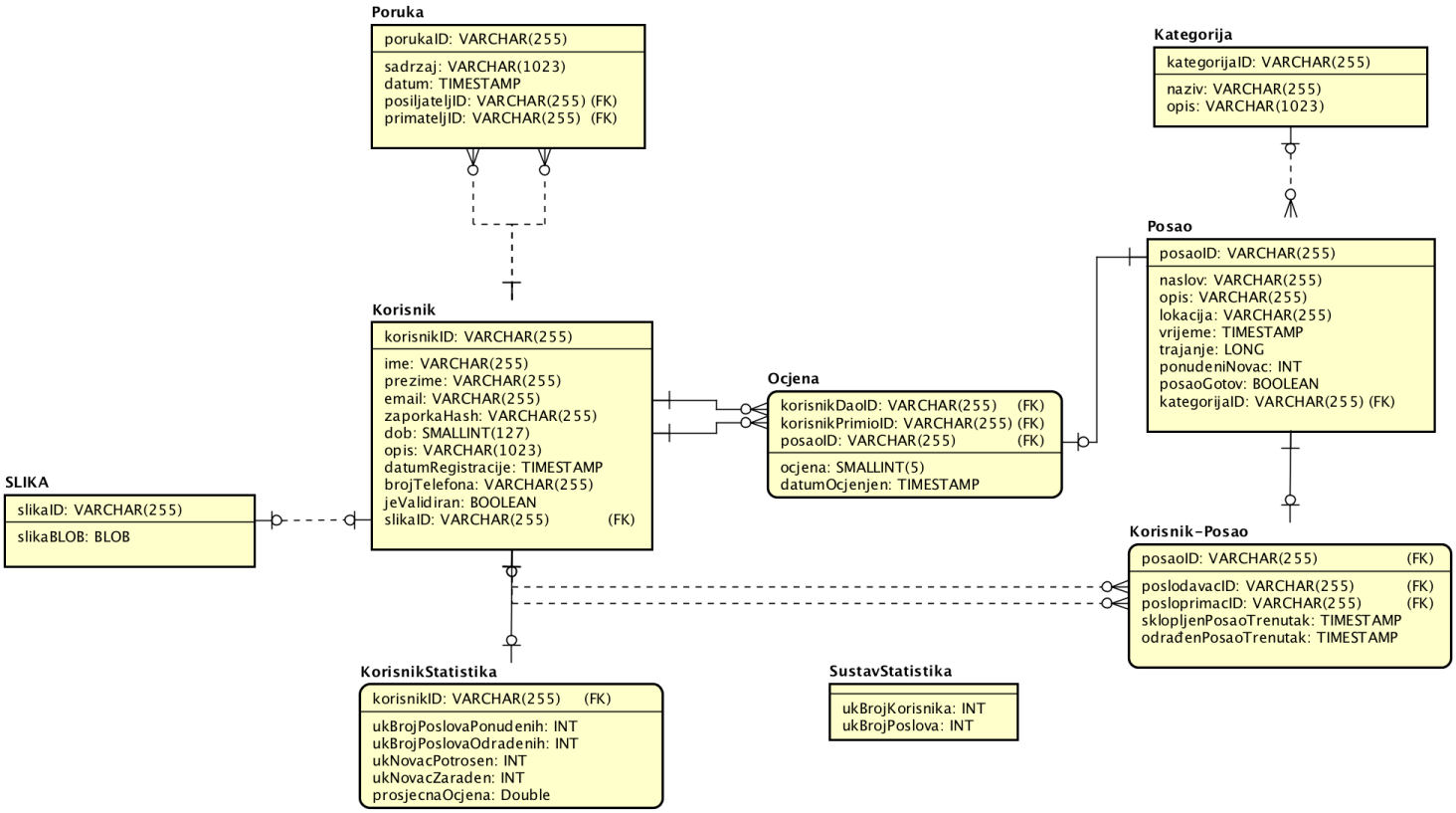
Entitet "Ocjena" opisuje svaku ocjenu korisnika. Entitet pamti koji je korisnik kojem korisniku dao ocjenu i za koji posao. Korisnik može ocijeniti drugog samo ako je između njih sklopljen posao. Ocjena je cijeli broj između 1 i 5. Entitet pamti trenutak kada je ocjena dodijeljena.

SustavStatistika

- ukBrojKorisnika: INT
- ukBrojPoslova: INT

Entitet "SustavStatistika" pamti osnovne informacije o sustavu. Pamti se ukupan broj korisnika aplikacije i ukupan broj poslova u bazi podataka.

ER DIJAGRAM


Poruka
porukaID: VARCHAR(255)
sadrzaj: VARCHAR(1023)
datum: TIMESTAMP
posiljateljID: VARCHAR(255) (FK)
primateljID: VARCHAR(255) (FK)
Kategorija
kategorijaID: VARCHAR(255)
naziv: VARCHAR(255)
opis: VARCHAR(1023)
Posao
posaoID: VARCHAR(255)
naslov: VARCHAR(255)
opis: VARCHAR(255)
lokacija: VARCHAR(255)
vrijeme: TIMESTAMP
trajanje: LONG
ponudeniNovac: INT
posaoGotov: BOOLEAN
kategorijaID: VARCHAR(255) (FK)
Korisnik
korisnikID: VARCHAR(255)
ime: VARCHAR(255)
prezime: VARCHAR(255)
email: VARCHAR(255)
zaporkaHash: VARCHAR(255)
dob: SMALLINT(127)
opis: VARCHAR(1023)
datumRegistracije: TIMESTAMP
brojTelefona: VARCHAR(255)
jeValidiran: BOOLEAN
slikaID: VARCHAR(255) (FK)
Ocjena
korisnikDaoID: VARCHAR(255) (FK)
korisnikPrimioID: VARCHAR(255) (FK)
posaoID: VARCHAR(255) (FK)
ocjena: SMALLINT(5)
datumOcjenjen: TIMESTAMP
SLIKA
slikaID: VARCHAR(255)
slikaBLOB: BLOB
Korisnik-Posao
posaoID: VARCHAR(255) (FK)
poslodavacID: VARCHAR(255) (FK)
posloprimacID: VARCHAR(255) (FK)
sklopljenPosaoTrenutak: TIMESTAMP
odradenPosaoTrenutak: TIMESTAMP
KorisnikStatistika
korisnikID: VARCHAR(255) (FK)
ukBrojPoslovaPonudenih: INT
ukBrojPoslovaOdradenih: INT
ukNovacPotrosen: INT
ukNovacZaraden: INT
prosjecnaOcjena: Double
SustavStatistika
ukBrojKorisnika: INT
ukBrojPoslova: INT

6.2. Dijagram razreda s opisom

Svi razredi koje aplikacija sadrži se mogu podijeliti u tri paketa:

1) Modeli
2) Aktivnosti (engl. *activity*)
3) Servisi (engl. *service*)

Modeli

U ovom se paketu nalaze razredi koji su samo objektno orijentirane inačice entiteta iz baze podataka. Modeli sadrže sve one atribute koje sadrže odgovarajući entiteti u bazi sa svim metodama potrebnim za uspješnu manipulaciju podacima kao što su metode dohvaćanja (engl. *geteri*) i postavljanja (engl. *seteri*). Oni također sadrže metodu stvaranja (engl. *konstruktori*) koja prima sve atribute kao parametar.

Potrebni modeli:
1) Korisnik
2) Poruka
3) Posao
4) Kategorija

Aktivnosti

Aktivnosti su posebni razredi koji predstavljaju poveznicu između korisničkog sučelja (GUI) s bazom podataka i sadrže svu funkcionalnost sustava. Oni nasljeđuju razred AppCompatActivity, a za svaki razred iz ovog paketa mora postojati odgovarajuća XML (*Extensible Markup Language*) datoteka koja sadrži podatke o izgledu i sadržaju korisničkog sučelja za pojedinu aktivnost. Ona opisuje izgled pojedine grafičke komponente i njihov međusobni odnos. Grafičke komponente mogu biti gumbi, slike, okviri za tekst itd.

Načelno, aktivnosti nemaju reference jedna na drugu. Kada jedna aktivnost poželi reći android operativnom sustavu da je došlo vrijeme prelaska na drugu aktivnost, onda im je dovoljna referenca na same sebe i ime aktivnosti na koju je potrebno skočiti. Sukladno tome, aktivnosti su načelno cjeline neovisne jedna od druge. Komunikaciju između aktivnosti je moguće ostvariti kroz pakete (engl. *bundle*) koji su implementirani kao *HashMape* – strukture temeljene na parovima ključ-vrijednost.

Potrebne aktivnosti:
1) PrijavaAktivnost – početni ekran koji zahtijeva od korisnika da unese korisničko ime i zaporku prije no što krene koristiti aplikaciju.
2) RegistracijaAktivnost – omogućuje korisniku da se registrira. Sadrži polja za unos podataka o korisniku i obavezno polje za unos slike.

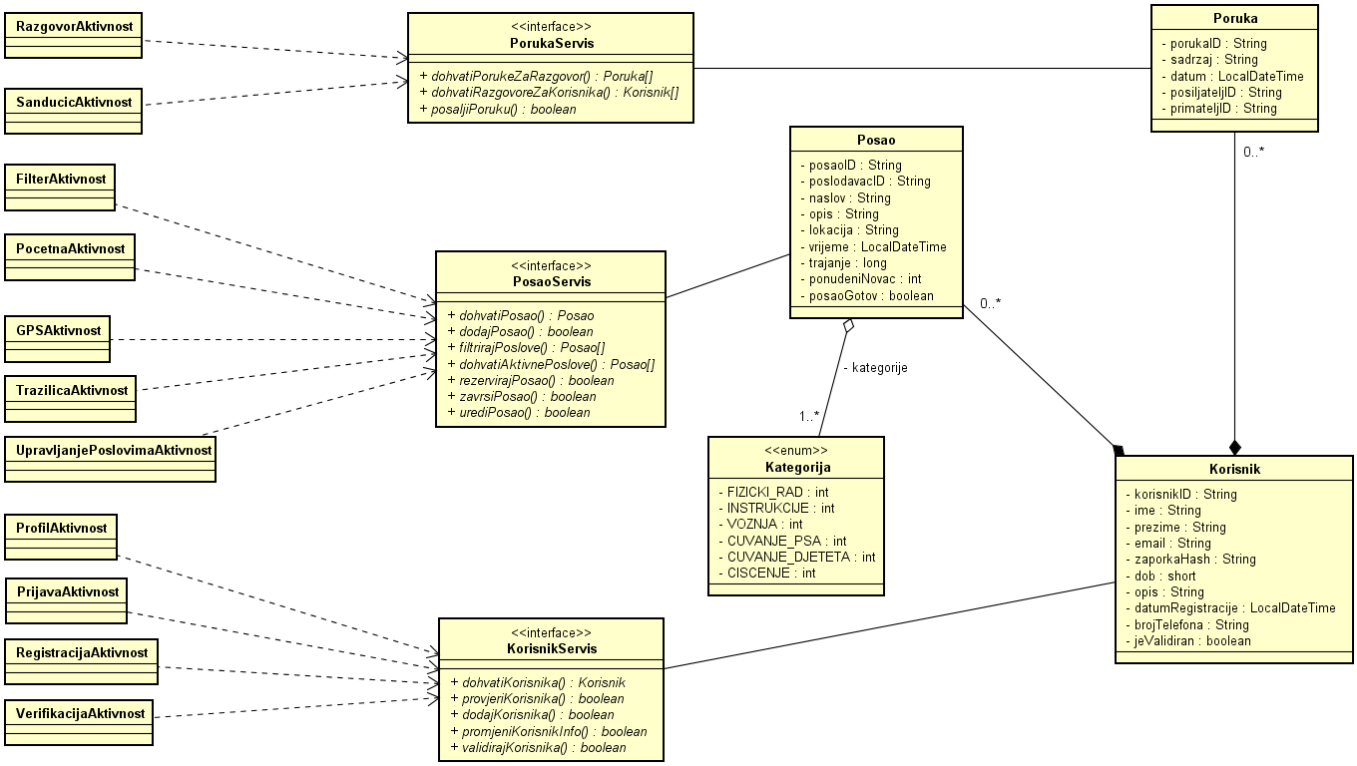
3) VerifikacijaAktivnost – ako je upisao dobre podatke za registraciju, korisnik u ovoj aktivnosti mora upisati verifikacijski kod koji mu je došao na e-mail adresu i kojime korisnik potvrđuje stvaranje profila.
4) PocetnaAktivnost – glavni i početni ekran u aplikaciji. Sadrži popis svih poslova poredanih po vremenu objavljivanja.
5) SanducicAktivnost – ekran koji prikazuje popis svih razgovora poredanih po vremenu zadnjeg dopisivanja.
6) RazgovorAktivnost – prikazuje sve poruke između dva korisnika.
7) FilterAktivnost – ekran koji sadrži opcije filtriranja kao što su filtriranje po vremenu početka posla, trajanju, plaćanju, lokaciji itd. Jednom kada se filter definira, aplikacija se vraća na početnu aktivnost, ali je popis poslova filtriran korisničkim unosom.
8) GPSAktivnost – GPS karta koja sadrži poslove u blizini u obliku pribadača na karti. Odavde je moguće vidjeti opis pojedinog posla pritiskom na jednu od obližnjih pribadača na karti.
9) TrazilicaAktivnost – ekran koji prikazuje rezultate traženja kroz tražilicu s vrha aplikacije. Ovdje se mogu nalaziti i korisnici i poslovi koji odgovaraju zadanom unosu u tražilicu.
10) UpravljanjePoslovimaAktivnost – prikaz svih poslova jednog korisnika. Na vrhu se nalaze prihvaćeni, a na dnu odrađeni poslovi. Ovdje korisnik može označiti neki posao odrađenim.
11) ProfilAktivnost – prikaz svih korisničkih podataka i statistika koje postoje u bazi. Ako je korisnik vlasnik profila kojeg promatra, on ima pravo izmijeniti neke podatke na profilu. Administrator ovdje vidi i poseban gumb za brisanje korisnika.

Svaka aktivnost sadrži:
1) Reference na grafičke komponente kao što su gumbi ili okviri za tekst
2) Javne metode - pozivaju se kod pritiska na neku grafičku komponentu
3) Privatne metode – omogućuju čitljiviji i organiziraniji kod

Servisi

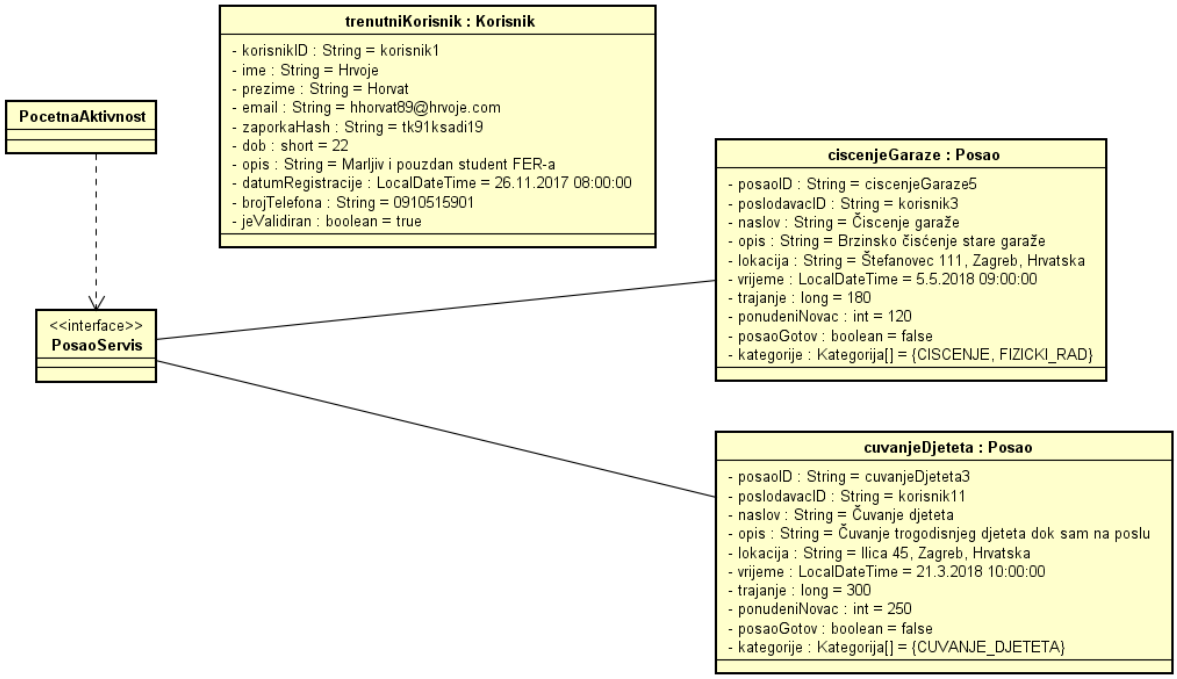
Posebna sučelja (engl. *interface*) namijenjena za komunikaciju s bazom podataka. Servisi omogućuju svim aktivnostima da dohvaćaju podatke ili mijenjaju bazu. Jedan servis je odgovoran za sve radnje vezane uz pojedini entitet. Tako je na primjer instanca koja implementira sučelje *KorisnikServis* dužna omogućiti radnje kao što su: mijenjanje podataka o pojedinom korisniku, dohvaćanje svih korisnika, dohvaćanje korisnika s određenim filterom itd.

Iako su servisi formalno sučelja, ona se dinamički instanciraju u aktivnostima pomoću biblioteke Retrofit, nakon čega se mogu dalje koristiti za komunikaciju s bazom.



Slika 6.2.1: Dijagram razreda za cijeli sustav s izostavljenim geterima, seterima i konstruktorima u modelima, te referencama na grafičke objekte i metodama u aktivnostima zbog jednostavnosti i čitkosti dijagrama.

6.3. Dijagram objekata



Slika 6.3.1: Dijagram objekata u trenutku pregledavanja poslova u početnoj aktivnosti. Kao što je već rečeno, početna aktivnost dinamički instancira objekt koji implementira sučelje PosaoServis pomoću Retrofita i koristi ga da bi dohvatio sve ponuđene poslove.

6.4. Ostali UML dijagrami

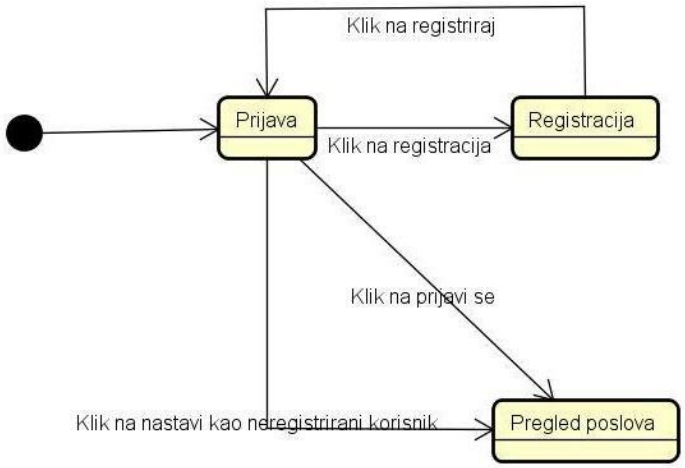
Dijagram stanja

Prilikom otvaranja aplikacije otvara se prozor u kojem je moguće ulogirati se u aplikaciju pomoću e-maila i lozinke ili se korisnik može registrirati nakon čega slijedi i dodatna verifikacija. Za korištenje same aplikacije korisnik obavezno mora biti registriran.

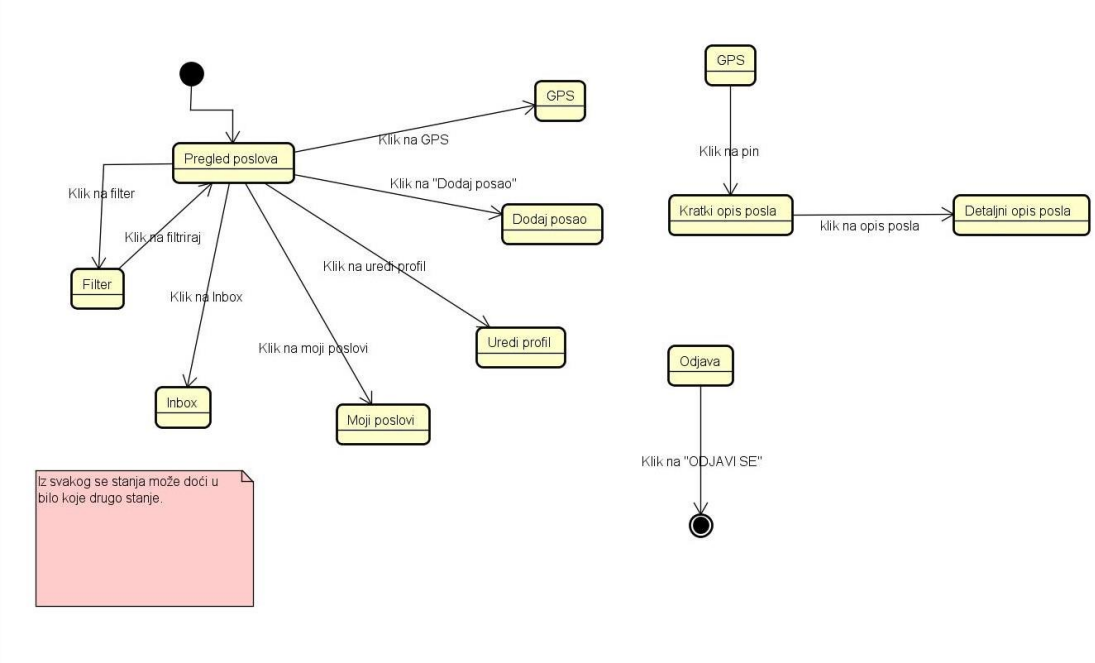
Nakon registracije korisnik je u mogućnosti: pregledavati poslove, dodavati nove poslove, uređivati vlastiti profil i prihvaćati i dogovarati se za poslove koje želi raditi.

Administratori imaju nešto veće ovlasti od registriranih osoba. Oni mogu raditi sve kao i registrirani korisnik ali još uz to mogu: brisati poslove (ako je posao neprimjeren), brisati profile registriranih korisnika (ako se korisnik neprimjereno ponaša) i slično.

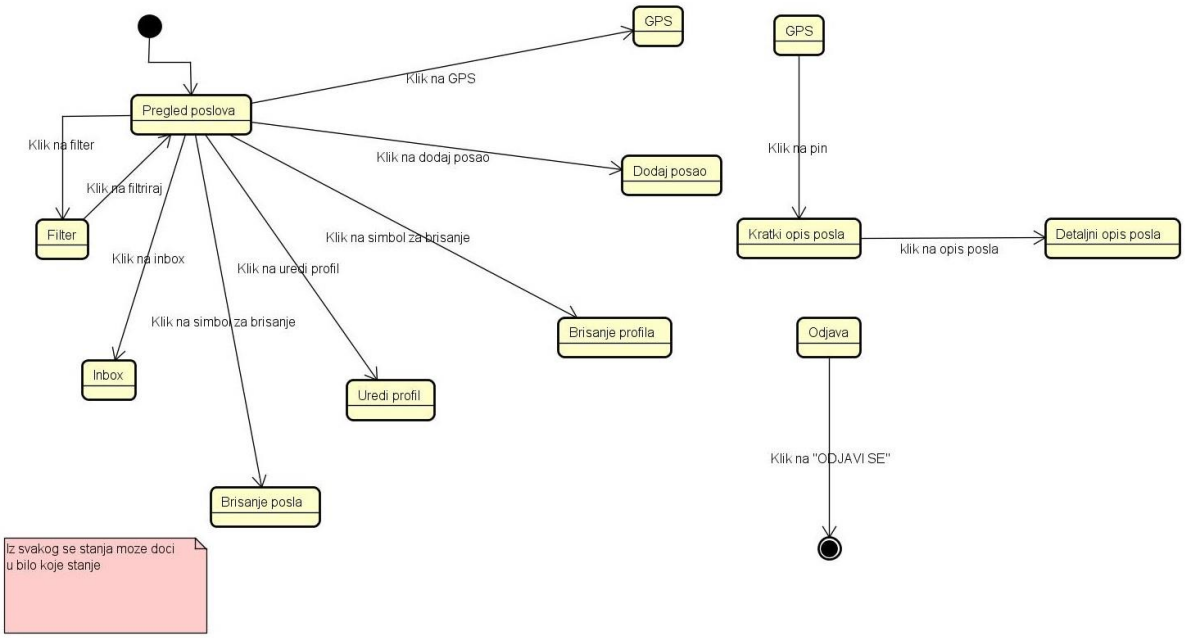
Ukoliko korisnik želi završiti sa radom na aplikaciji klikom na gumb „Odjava“ odjavljuju se sa svoga profila i dolazi na početnu stranicu same aplikacije (stranice za prijavu).



Slika 6.1. Dijagram stanja (Prvi prozor prilikom otvaranja aplikacije)



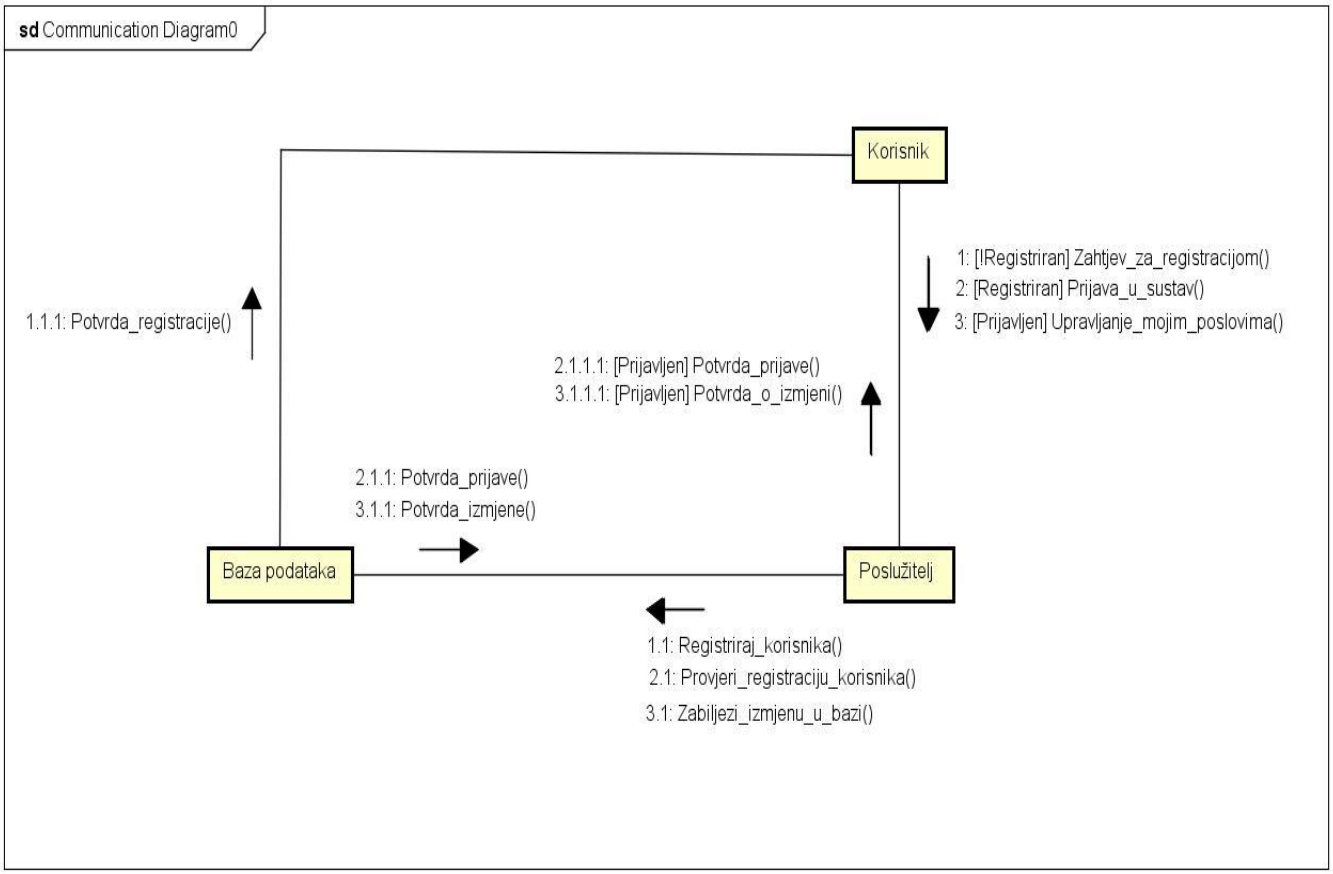
Slika 6.2. Dijagram stanja (Registrirani korisnik)



Slika 6.3. Dijagram stanja (Administrator)

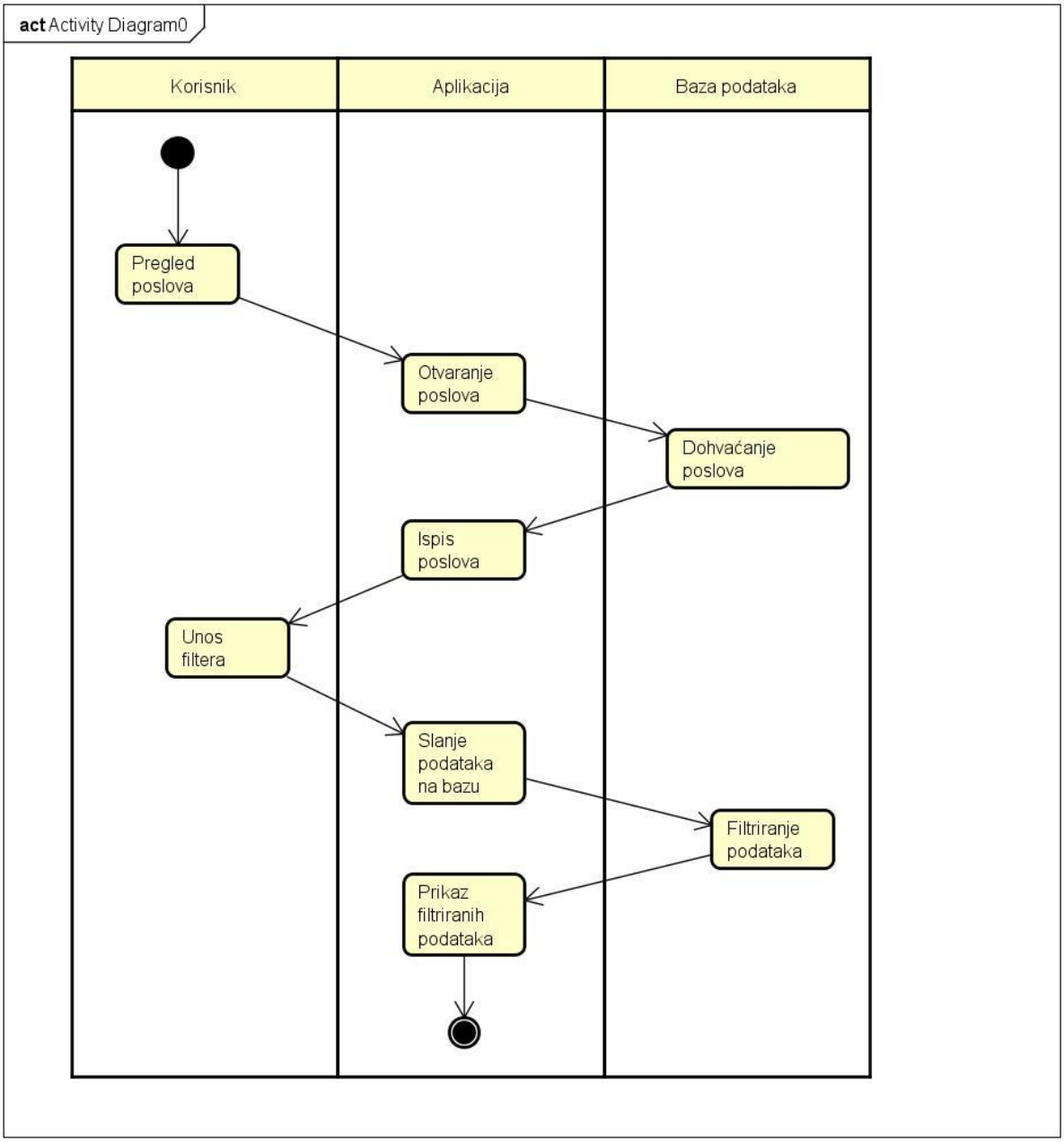
Komunikacijski dijagram

Komunikacijski dijagram na slici 6.4. prikazujem registraciju korisnika. Ako korisnik već nije registriran klikom na ikonu „Register“ to može napraviti unosom sljedećih podataka: ime, prezime, lozinka, ponovljena lozinka, e-mail adresa, broj godina, broj mobitela i kratak opis o korisniku. Sustav te podatke šalje bazi podataka. U slučaju dobro unesenih svih podataka baza podataka šalje korisniku email kojim korisnik dobiva verifikacijski kod.



Slika 6.4. Komunikacijski dijagram (registracija korisnika i izmjena podataka određenog posla)

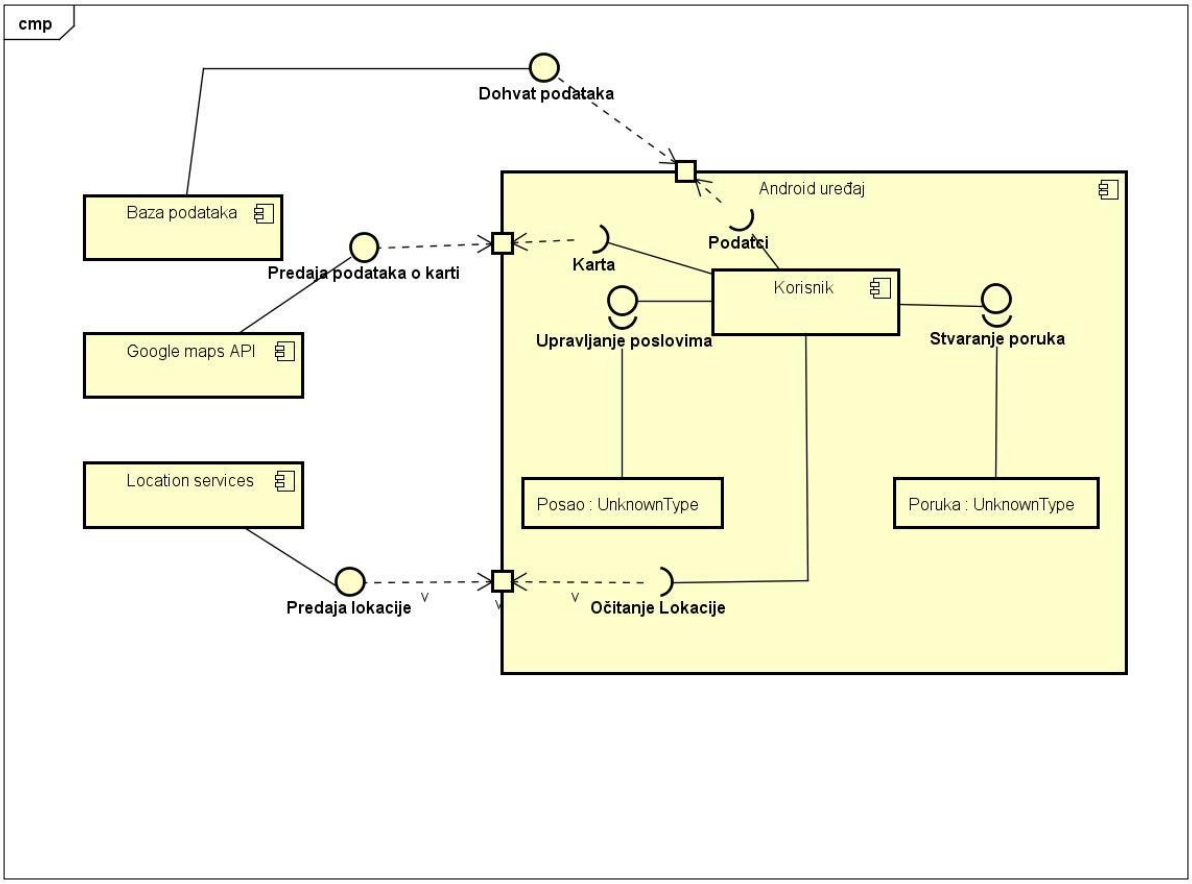
Dijagram aktivnosti



Slika 6.5 – Dijagram aktivnosti za UC 11

Korisnik pokreće aplikaciju, i u polju predviđenom za prijavu unosi svoj e-mail i lozinku. U slučaju unosa krivih podataka aplikacija će mu ponuditi novi pokušaj unosa. Nakon ispravnog unosa podataka korisniku su predočeni svi poslovi i korisnik je tada u mogućnosti napraviti svojevrstan filter poslova. Baza obrađuje i filtrira poslove te se oni prikazuju u aplikaciji.

Dijagram komponenti



Slika 6.6. – Dijagram komponenti

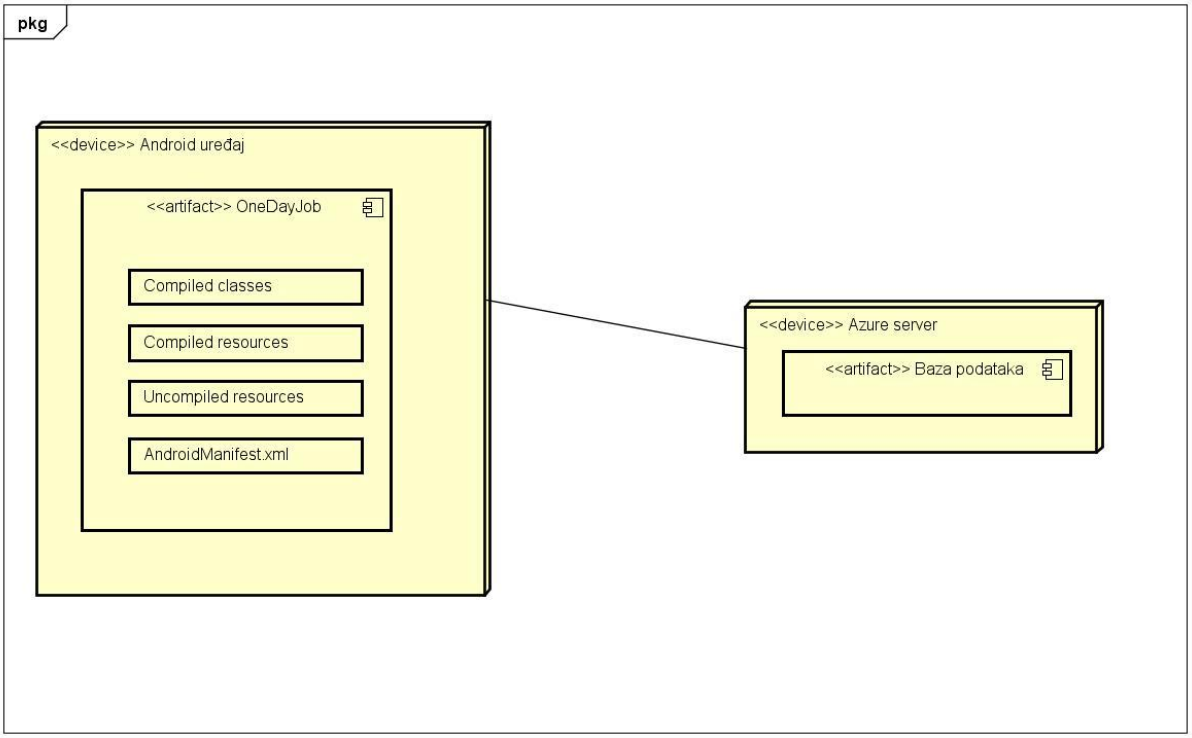
Sustav se sastoji od komponenti koje su ključne za rad aplikacije. Unutar android uređaja nalazi se komponenta korisnik koja upravlja poslovima, te može stvarati poruke. Također baza podataka, Google maps API i location services opskrbljuju aplikaciju s dodatnim podacima koji su potrebni za pravilan rad aplikacije.

- Android uređaj – uređaj na kojemu se pokreće aplikacija
- Baza podataka - nalazi se na serveru Azure, služi za pohranu podataka
- Google maps API – omogućuje aplikaciji korištenje interaktivne karte
- Location services – pružaju lociranje korisnika pomoću GPS tehnologije

7. Implementacija i korisničko sučelje

7.1. Dijagram razmještaja

Na slici je prikazana generalna topologija sustava. Komunikacija između aplikacije i servera odvija se preko HTTP veze. Baza podataka pisana je u SQL-u i nalazi se na Azure-ovim serverima. Na android uređaju instalirana je aplikacija pomoću koje korisnik odrađuje sve akcije koje mu aplikacija omogućava.



Slika 7.1. Dijagram razmještaja

7.2. Korištene tehnologije i alati

Pri izradi mobilne aplikacije koristili smo razvoju okolinu Android Studio u kojoj koristimo programski jezik Java za funkcionalnost i XML za grafički izgled i koristili smo Spring za backend.

Prilikom razvoja korišten je git sustav za kontrolu verzija na repozitoriju putem web usluge GitLab.

Prilikom izrade UML dijagrama koristili smo program Astah Community.

Java
Java je objektno orijentirani programski jezik. Velika prednost u odnosu na većinu dotadašnjih programskih jezika je to što se programi pisani u Javi mogu izvoditi bez preinaka na svim operativnim sustavima za koje postoji **JVM**(Java Virtual Machine), dok je klasične programe pisane primjerice u C-u potrebno prilagođavati platformi(Operacijskom sustavu) na kojem se izvode. Upravo zbog toga je popularna za razvoj programa na mobilnim telefonima.

XML
XML je kratica za Extendeble Markup Language odnosno jezik za označavanje podataka. Ideja je bila stvoriti jedan jezik koji će biti jednostavno čitljiv i ljudima i računalnim programima. Princip realizacije je vrlo jednostavan: odgovarajući sadržaj treba se uokviriti odgovarajućim oznakama koje ga opisuju i imaju poznato, ili lako shvatljivo značenje. Format oznaka u XMLu vrlo je sličan formatu oznaka u npr. HTML jeziku.

Spring
Spring je radni okvir za java platformu. U svojoj osnovi služi za izgradnju web i mobilnih aplikacija, ali ima i druge značajke. Iako ovaj okvir nema nikakav specifični programski model postao je popularan u kod programera koji koriste Javu. Springov okvir je open source

7.3. Isječak programskog koda vezan za temeljnu funkcionalnost sustava

Dizanje baze

Prilikom stvaranja baze podataka potrebno je definirati koje sve atribute entiteti moraju sadržavati. U 6. poglavlju definirani su svi atributi koje entiteti moraju imati, pa u ovom poglavlju se nećemo time baviti. Također, prilikom stvaranja baze podataka moramo definirati sve mogućnosti baze podataka. Svaki entitet se mora moći stvoriti, učitati iz baze, izmijeniti i pobrisati, zbog toga korišten je CrudRepository (engl. create, read, update, delete repository) koji implementira sve ove zahtjeve. Sve se entitete može pretraživati u bazi po njihovom identifikatoru. Ostale zahtjeve morali smo sami definirati.

Zahtjevi koje baza mora omogućiti pri dohvatu korisnika su sljedeći:

1. Pretraživanje po prezimenu
2. Pretraživanje po imenu
3. Pretraživanje po imenu i prezimenu
4. Pretraživanje po emailu
5. Pretraživanje po validaciji korisnika

Ovako to izgleda u kodu:

```
import org.springframework.data.repository.CrudRepository;

import server.models.Korisnik;


public interface KorisnikRepository extends CrudRepository<Korisnik, Long>{
    List<Korisnik> findByPrezime(String prezime);
    List<Korisnik> findByIme(String ime);
    List<Korisnik> findByJeValidiran(boolean jeValidiran);
    List<Korisnik> findByImeAndPrezime(String ime,String prezime);
    Korisnik findByEmail(String email);
}
```

Zahtjevi koje baza mora omogućiti pri dohvatu poruka su sljedeći:

1. Pretraga po pošiljatelju
2. Pretraga po primatelju
3. Pretraga po primatelju i pošiljatelju

```
public interface PorukaRepository extends CrudRepository <Poruka, Long>{
    List<Poruka> findByPosiljateljID(long posiljateljID);
    List<Poruka> findByPrimateljID(long primateljID);
    List<Poruka> findByPrimateljIDAndPosiljateljID(long posiljateljID,long primateljID);
}
```

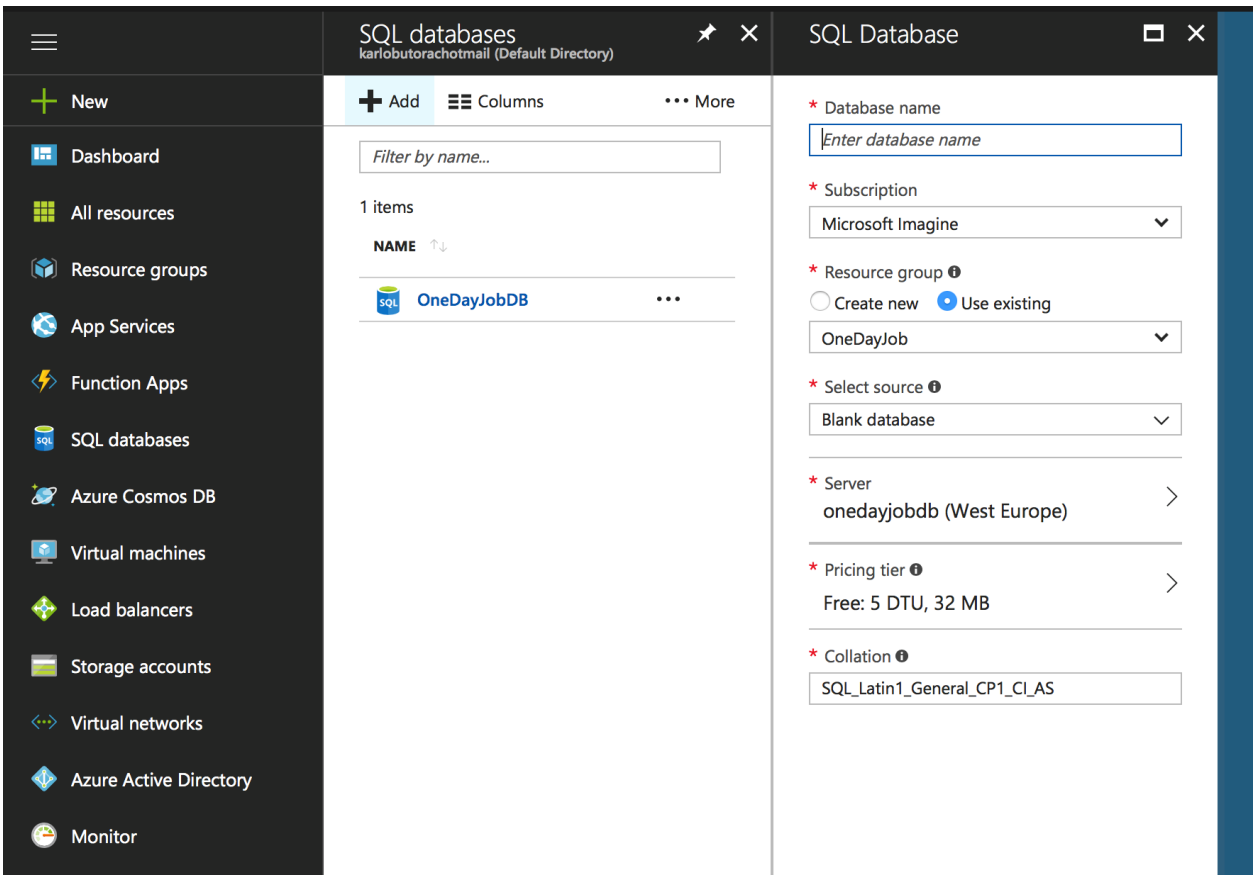
Zahtjevi koje baza mora omogućiti pri dohvatu poslova su sljedeći:

1. Pretraga po poslodavcu

2. Pretraga po posloprimcu
3. Pretraga po poslodavcu i posloprimcu
4. Pretraga po kategorijama
5. Pretraga po rezerviranosti posla
6. Pretraga po završenosti posla
7. Pretraga po kategoriji i završenosti posla

```
public interface PosaoRepositary extends CrudRepository<Posao,Long>{
    List<Posao> findByPoslodavacID(long poslodavacID);
    List<Posao> findByPosloprimacID(long posloprimacID);
    List<Posao> findByPoslodavacIDAndPosloprimacID(long poslodavac,long posloprimac);
    List<Posao> findByKategorijaID(long kategorijaID);
    List<Posao> findByPosaoRezerviran(boolean posaoRezerviran);
    List<Posao> findByPosaoGotov(boolean posaoGotov);
    List<Posao> findByKategorijaIDAndPosaoRezerviran(long kategorijaID,boolean posaoRezerviran);
}
```

Baza se na azuru stvara pritiscima na: SQL database -> Add , pa ispunjavanjem svih potrebnih podataka kao što je prikazano na sljedećoj slici:

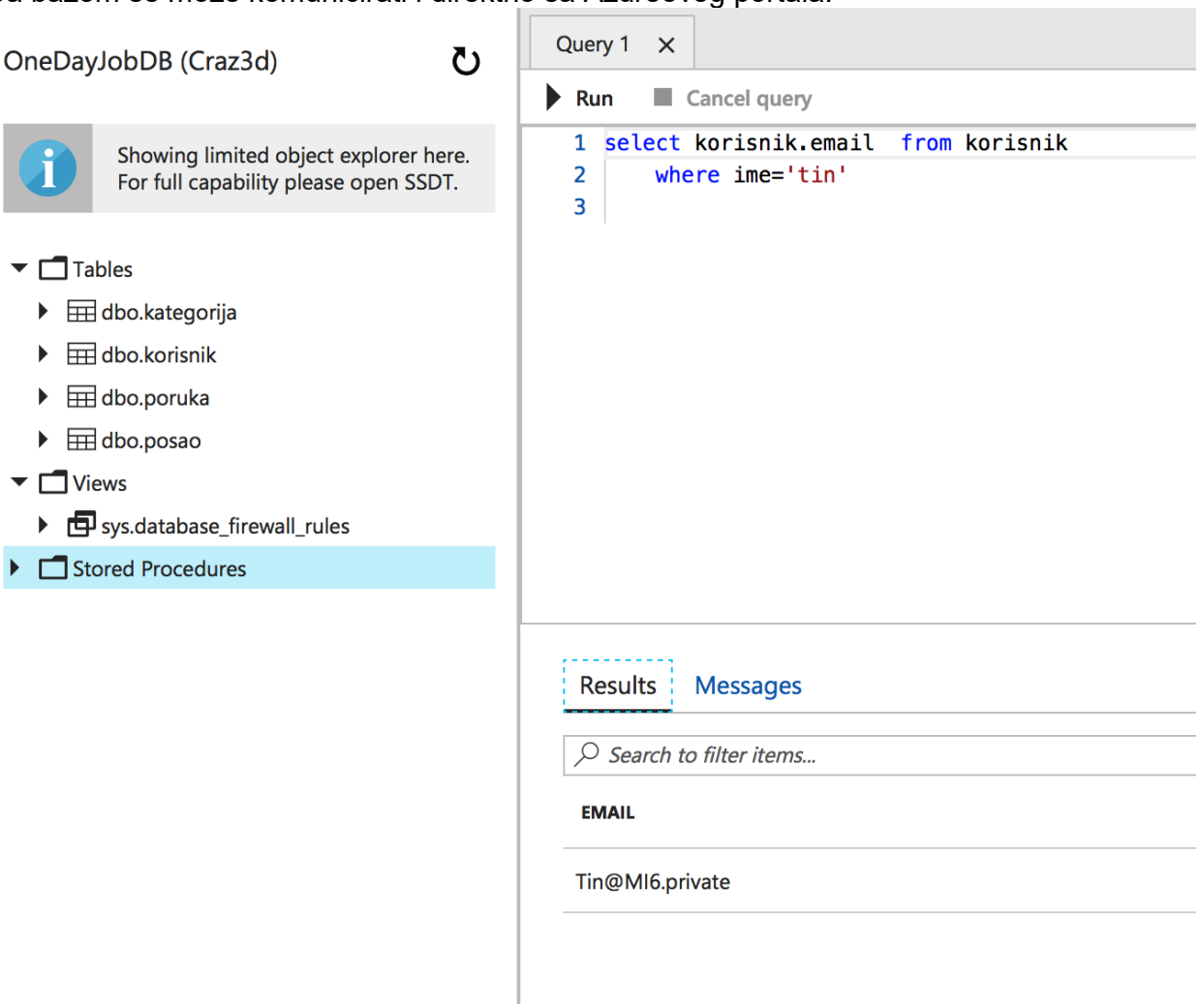


Kako bi se spojili sa serverom na stvorenu bazu podataka potrebno je u application.properties od servera dodati:

```
spring.datasource.url=jdbc:sqlserver://onedayjobdb.database.windows.net:1433;database=OneDayJobDB;encrypt=true;
spring.datasource.username=Username@onedayjobdb
spring.datasource.password=jakoTeskiPasswordKojiSeNeMozePogoditi123456789
spring.datasource.driver-class-name=com.microsoft.sqlserver.jdbc.SQLServerDriver
spring.jpa.generate-ddl=true
```

Ukoliko se komunicira sa bazom sa servera koji se pokreće lokalno (ne sa Microsoft Azurea) potrebno je dodati u postavke vatrozida (engl. firewall) propusnicu za IP-adresu klijenta.

Sa bazom se može komunicirati i direktno sa Azureovog portala:

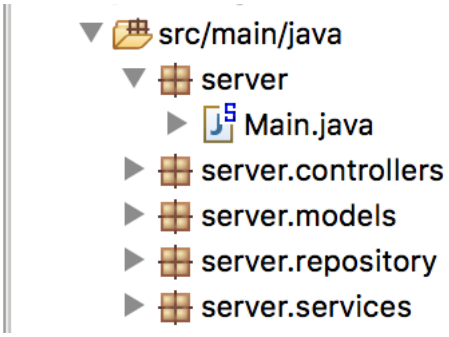


Dizanje servera

Server je organiziran na način da ima 3 sloja. Ti slojevi su:

1. Kontroleri
2. Servisi
3. Repozitoriji

Kontroleri razgovaraju sa mobilnom aplikacijom i prosljeđuju zahtjeve mobilne aplikacije servisima koji obrađuju zahtjeve i prosljeđuju ih repozitorijima koji su dužni za razgovor sa bazom podataka. U aplikaciji još postoji paket modeli koji modelira sve entitete. Na sljedećoj slici vidi se struktura servera:



Na sljedećoj slici nalazi se struktura KorisnikKontrolera:

Na sljedećoj slici nalazi se struktura KorisnikServisa:

Možemo primijetiti da su strukture servisa i kontrolera vrlo slične, razlog tome je što je zadaća kontrolera samo mapirati sve funkcije koje server pruža mobilnoj aplikaciji i proslijediti ih servisu koji ih onda obrađuje.

Za slanje elektroničke pošte potrebne za verifikaciju korisnika koristi se JavaMailSenderImpl čiju implementaciju nudi SpringFramework. Na sljedećoj slici nalazi se xml forma potrebna za povezivanje servera sa gmail API-em

```
<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-2.5.xsd">

<bean id="mailSender" class="org.springframework.mail.javamail.JavaMailSenderImpl">
    <property name="host" value="smtp.gmail.com" />
    <property name="port" value="587" />
    <property name="username" value="oneDayJobApp@gmail.com" />
    <property name="password" value="sifrasifra123" />

    <property name="javaMailProperties">
        <props>
                <prop key="mail.smtp.auth">true</prop>
                <prop key="mail.smtp.starttls.enable">true</prop>
            </props>
    </property>
</bean>

<bean id="mailMail" class="server.services.MailServis">
    <property name="mailSender" ref="mailSender" />
</bean>
</beans>
```

Komunikacija s bazom

U nastavku prikazujemo slijed nužnih koraka za komunikaciju s bazom. Primjer je vezan samo za model posla, ali za ostale modele (posao, korisnik) vrijedi analogan scenarij. Najprije moramo opisati model posla. On mora implementirati Serializable kako bi bio prenosiv u JSON obliku. Anotacije služe kako bi se označilo kako baza imenuje pojedine atribute.

```
public class Posao implements Serializable {
    @SerializedName("posaoID")
    private long posaoId;
    @SerializedName("poslodavacID")
    private long poslodavacId;
    @SerializedName("posloprimacID")
    private long posloprimacId;
    @SerializedName("naslov")
    private String naslov;
    @SerializedName("opis")
    private String opis;
    @SerializedName("lokacija")
    private String lokacija;
    @SerializedName("vrijeme")
    private long vrijeme;
    @SerializedName("trajanje")
    private long trajanje;
    @SerializedName("ponudeniNovac")
    private int ponudeniNovac;
    @SerializedName("posaoGotov")
    private boolean posaoGotov;
    @SerializedName("kategorijaID")
    private Long kategorijaID;
    @SerializedName("posaoRezerviran")
    private boolean posaoRezerviran;
```

Sučelje se piše vrlo jednostavno. Anotira se samo koji je oblik (GET, POST) i parametri zahtjeva, te očekivani odgovor.

```
public interface PosaoServis {

    @GET("/posao/aktivni/")
    Call<List<Posao>> getAktivniPoslovi();

    @POST("/dodajPosao")
    Call<Boolean> dodajPosao(@Body Posao posao);

    @POST("/posao/update")
    Call<Posao> updatePosao(@Body Posao posao);


}
```

Retrofit je biblioteka koja omogućuje komunikaciju s bazom na temelju stvorenih sučelja. Pružimo im URL na bazu i možemo koristiti servis za komunikaciju.

```
Retrofit retrofit = new Retrofit.Builder()
        .baseUrl("https://onedayjobapp2.azurewebsites.net")
        .addConverterFactory(GsonConverterFactory.create())
        .addConverterFactory(ScalarsConverterFactory.create())
        .client(httpClient.build())
        .build();
final PosaoServis service = retrofit.create(PosaoServis.class);
```

Isječak prikazuje primjer pokretanja dohvata s baze GET zahtjevom. Konkretno, dohvaćaju se aktivni poslovi za feed i provjerava se postoji li definirani filter. Dohvat se odrađuje asinkrono zato što se ne želi opteretiti glavna dretva.

```
new Thread(new Runnable() {
    @Override
    public void run() {
        service.getAktivniPoslovi().enqueue(new Callback<List<Posao>>() {
            @Override
            public void onResponse(Call<List<Posao>> call, Response<List<Posao>> response) {
                Log.d( tag: "Login",  msg: "onResponse: " + response.body());

                posloviTest = response.body();
                TheMainActivity.this.onMapReady(mMap);

                Intent intent = getIntent();
                Bundle bundle = intent.getExtras();
                Filter filter = null;
                filter =(Filter) bundle.get("filter");
                List<Posao> posloviFiltrirani = new ArrayList<>();


                if (filter != null){
                    for (Posao p : posloviTest){
                        if (p.getKategorijaID() == filter.getKategorijaID().longValue()){
                            posloviFiltrirani.add(p);
                        }
                    }

                    FeedAdapter feedAdapter = new FeedAdapter( context: TheMainActivity.this, R.layout.list_element, posloviFiltrirani);
                    listJobs.setAdapter(feedAdapter);
                    return;
                }
                FeedAdapter feedAdapter = new FeedAdapter( context: TheMainActivity.this, R.layout.list_element, posloviTest);
                listJobs.setAdapter(feedAdapter);
            }
```

Prikaz slanja posla na bazu POST zahtjevom.

```
new Thread((Runnable) () → {
        service.dodajPosao(posao).enqueue(new Callback<Boolean>() {
            @Override
            public void onResponse(Call<Boolean> call, Response<Boolean> response) {
                Toast.makeText( context: AddJobActivity.this,  text: "Dodan posao!", Toast.LENGTH_SHORT).show();
                Log.d( tag: "REG",  msg: "onResponse: dodao sam posao!");

                Intent intent = new Intent( packageContext: AddJobActivity.this, TheMainActivity.class);

                Bundle bundle = new Bundle();
                bundle.putSerializable("korisnik", korisnik);
                intent.putExtras(bundle);

                startActivity(intent);
            }

            @Override
            public void onFailure(Call<Boolean> call, Throwable t) {
                Log.d( tag: "REG",  msg: "onFailure: " + t.getMessage());
                Toast.makeText( context: AddJobActivity.this,  text: "Nisam uspio registrirati. Razlog: " + t.getMessage(), Toast.LE
            }
        });
}).run();
```

GPS

Dodavanje pinova na kartu.

```
        for (Posao p : posloviAktivni){
            try {
                results = geocoder.getFromLocationName(p.getLokacija(), maxResults: 2);
            } catch (IOException e) {
                continue;
            }
            if (results.isEmpty())continue;
            LatLng posaoLatLng = new LatLng(results.get(0).getLatitude(),results.get(0).getLongitude());
            Marker posaoMarker = mMap.addMarker(new MarkerOptions().position(posaoLatLng).title(p.getNaslov()));
        }
```

Spajanje pinova s poslovima kako bi svaki pin mogao odvesti na točno određeni posao.

```
GoogleMap.OnInfoWindowClickListener listener = new GoogleMap.OnInfoWindowClickListener() {
    @Override
    public void onInfoWindowClick(Marker marker) {

        //Ovo vjv moze bolje - Spajanje markera i posla
        String title = marker.getTitle();

        for (Posao pos : posloviAktivni){
            if (title.equals(pos.getNaslov())){

                Intent intent = new Intent( packageContext: TheMainActivity.this, JobActivity.class);

                Bundle bundle = new Bundle();

                Log.d( tag: "odabrani posao",  msg: "onClick: saljem posao" + pos.getNaslov());
                bundle.putSerializable("odabraniPosao", pos);
                intent.putExtras(bundle);

                startActivity(intent);
            }
        }
    }
};
```

7.4. Ispitivanje programskog rješenja

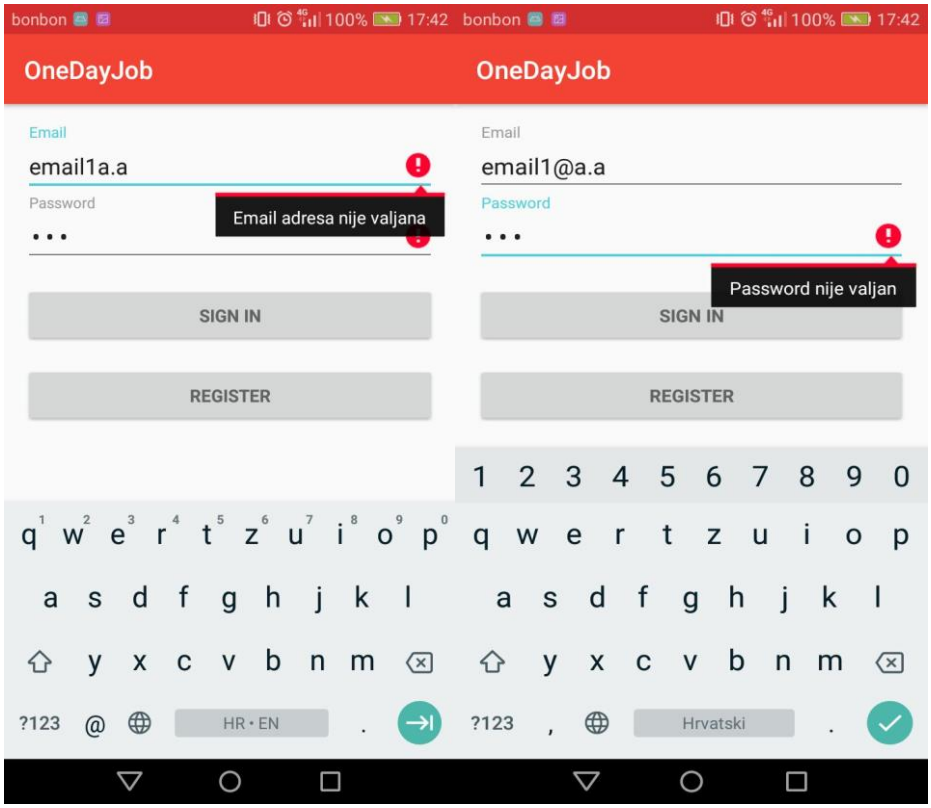
Ispitivat ćemo sve funkcionalnosti aplikacije s razine registriranog korisnika. Ispitat ćemo sve rubne razine sustava koje bi mogle dovesti do rušenja aplikacije. Ispod svakog dijela bit će opisan slučaj koji se ispituje te očekivani i dobiveni rezultat.

Slučaj 1

Opis: Prijava korisnika u sustav

Očekivani rezultat: Ako su korisničko ime i lozinka dobro uneseni (korisnik je registriran), korisnik je prijavljen u sustav.

Dobiveni rezultat: U slučaju unosa krivih podataka otvara se dijalog koji signalizira da su podatci krivi. Ako su podatci ispravni korisnik je ulogiran u sustav.



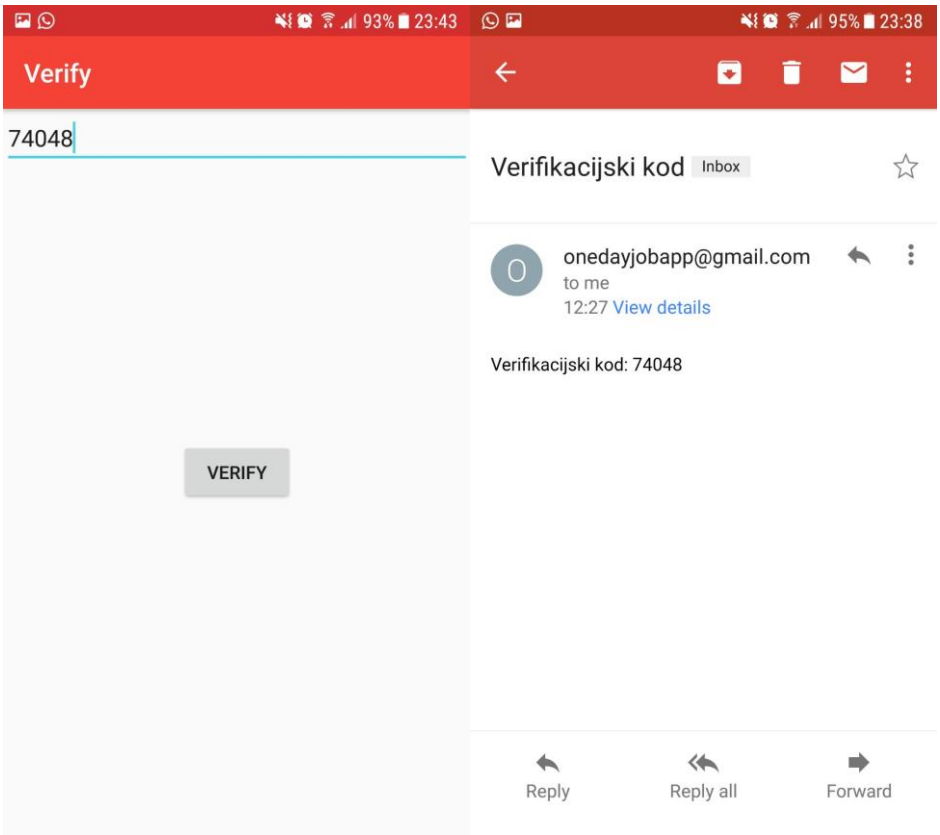
Slika 7.4.1. – Rezultat slučaja 1

Slučaj 2

Opis: Verifikacija korisnika

Očekivani rezultat: Dolazak verifikacijskog koda na mail, i upisivanje koda u aplikaciju.

Dobiveni rezultat: Nakon unosa verifikacijskog koda aplikacija nas vraća nas aktivnost za prijavu



Slika 7.4.2. Rezultat slučaja 2

Slučaj 3

Opis: Stvaranje novog posla

Očekivani rezultat: Korisnik dodaje posao i može ga vidjeti u feedu.

Dobiveni rezultat: Korisnik vidi posao u feedu.

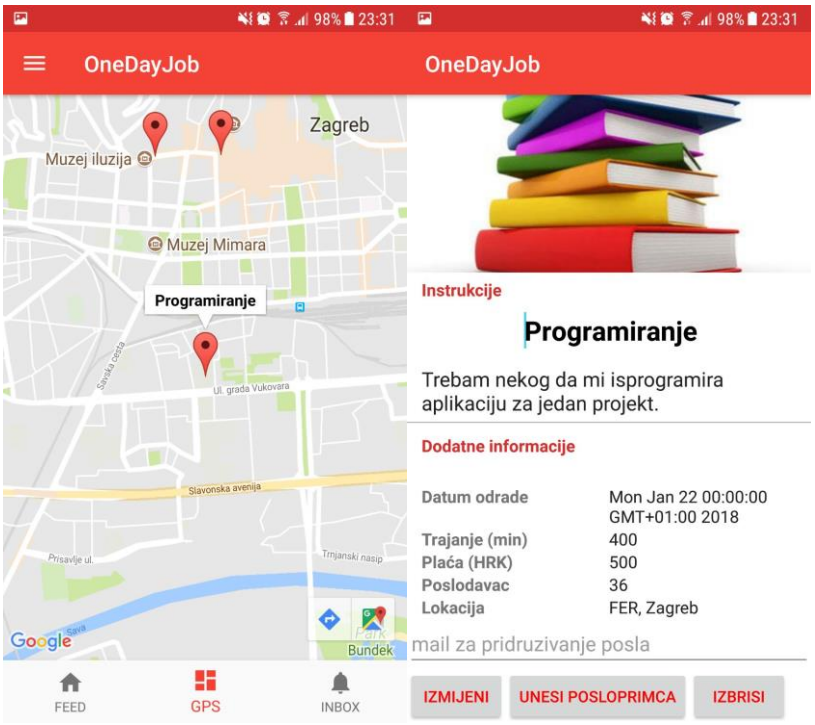
Slika 7.4.3 Slika slučaja 3

Slučaj 4

Opis: Ulazak u detaljni opis posla preko GPS mape

Očekivani rezultat: Korisnik ima prikaz detaljnog opisa posla.

Dobiveni rezultat: Korisnik nakon klika na pin u GPS mapi dobiva detaljni opis posla.



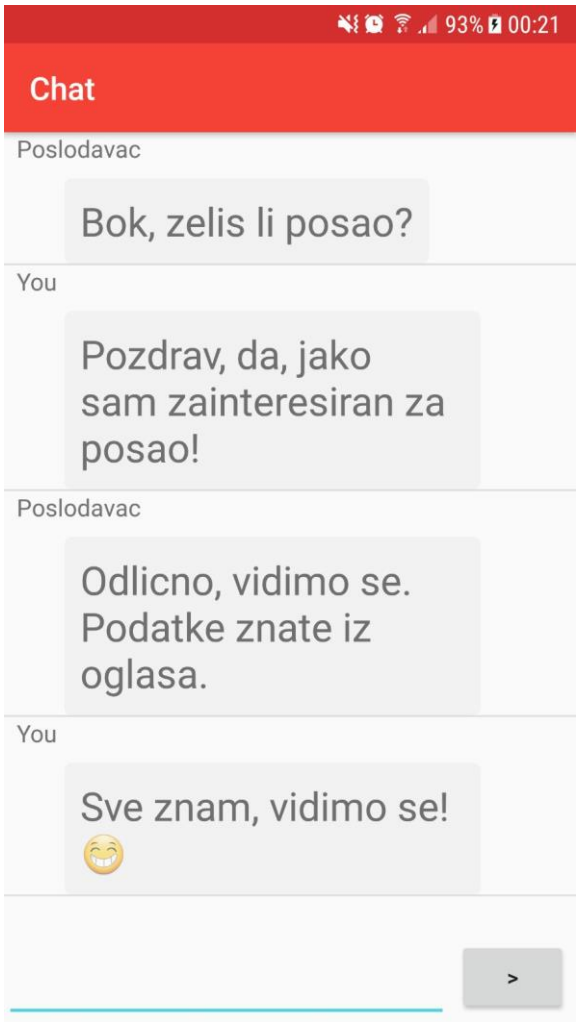
Slika 7.4.4. Slika slučaja 4

Slučaj 5

Opis: Dogovaranje između dva korisnika u inboxu

Očekivani rezultat: Oba korisnika vide cijeli tijek razgovora.

Dobiveni rezultat: Nakon slanja poruke oba korisnika vide tijek cijelog razgovora.



Slika 7.4.5. Slika slučaja 5

7.5. Upute za instalaciju

Aplikacija može biti preuzeta na dva načina. Prvi način je preko Google play store-a ili preuzimanjem aplikacije sa GitLab-a.

Preuzimanje preko Google play store-a

- Pokrenite Google play store i u traci pretrage upišite „OneDayJob“ ili aplikaciju potražite izravno na linku https://play.google.com/store/apps/details?id=hr.fer.opp.onedayjob
- Odaberite aplikaciju i pritisnite tipku „Instaliraj“
- Nakon završene instalacije možete pokrenuti aplikaciju

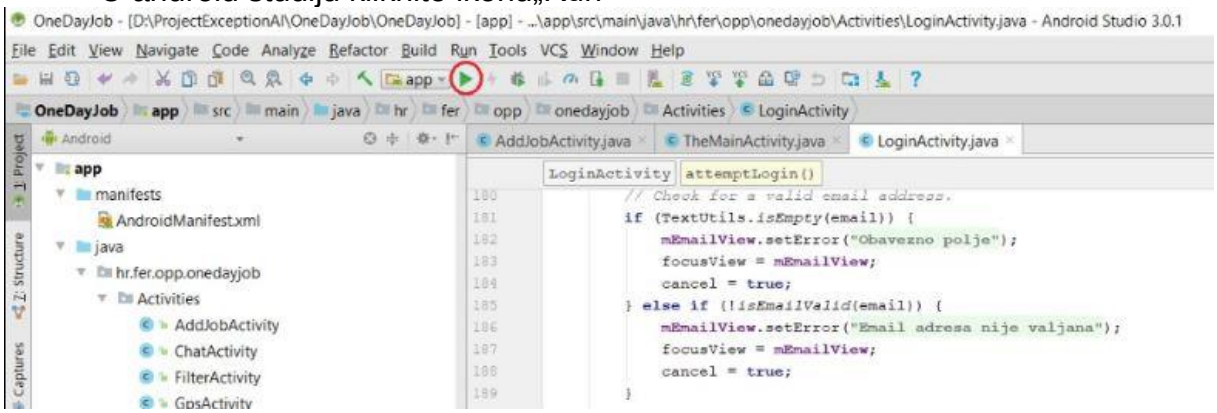
Preuzimanje preko GitLab-a

- Preko zadanog preglednika upišite web lokaciju https://gitlab.com/ProjectExceptionAI/OneDayJob
- Nakon otvaranja stranice odaberite gumb download

Slika 7.5.1. Gumb download

- Nakon što se aplikacija skinula uz pomoć android studija potrebno je otvoriti file u android studiju
- Spojite mobitel sa usb kabelom na računalo

- U android studiju kliknite ikonu„Run“

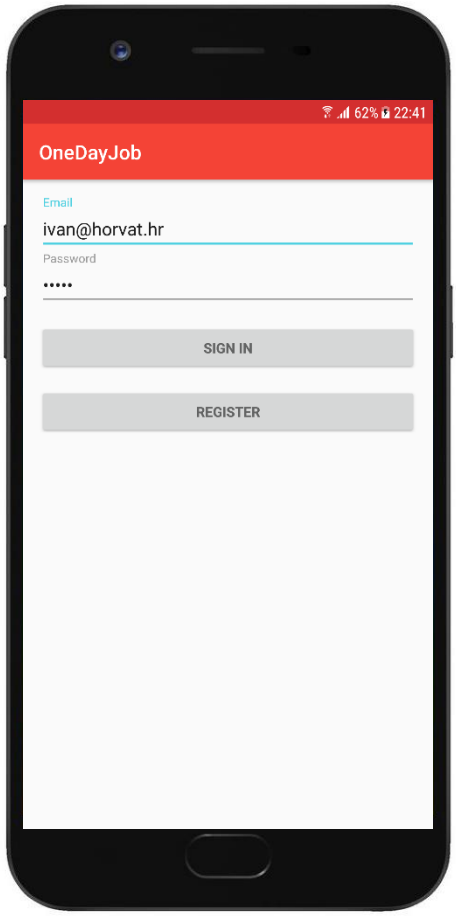


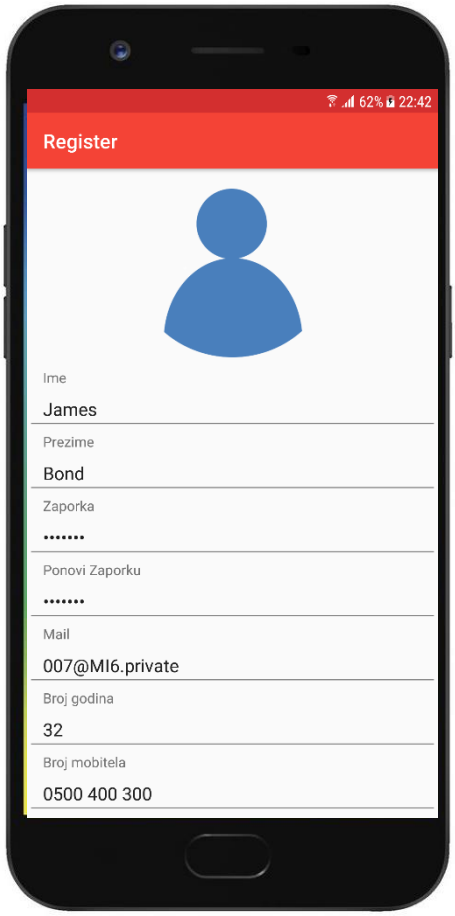
Slika 7.5.2. Ikona run

- Aplikacija će te automatski instalirati i pokrenuti

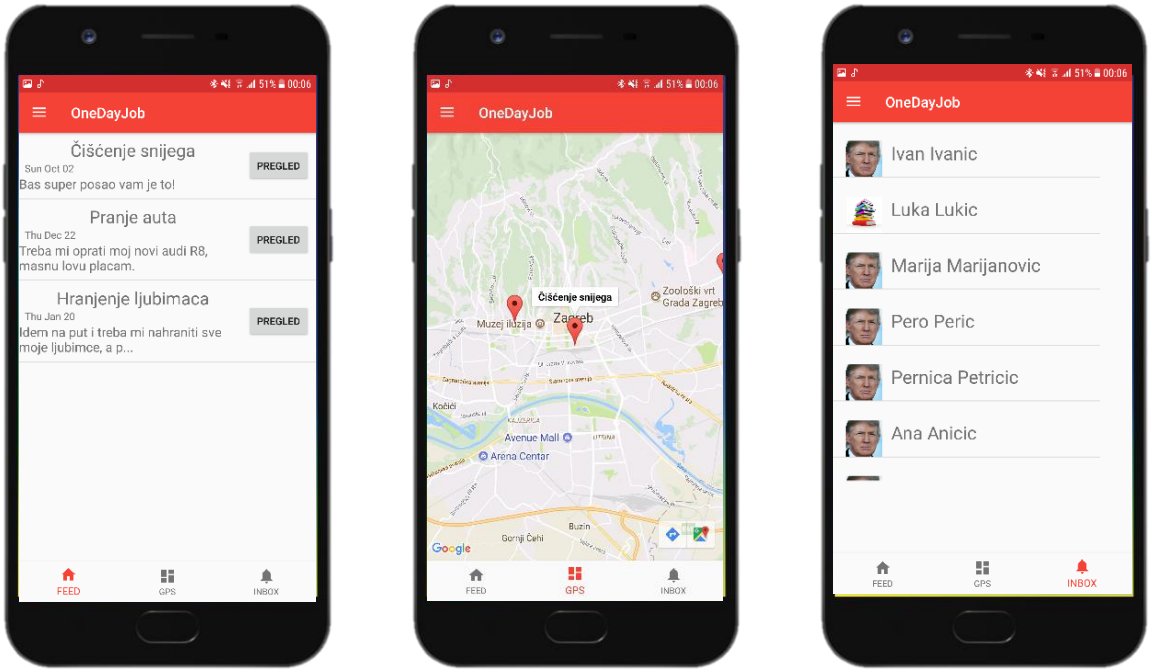
7.6. Korisničke upute

Pritiskom na ikonu aplikacije otvara se početni ekran aplikacije koji nudi dvije opcije: prijavu i registraciju. Pri prvom korištenju korisnik mora pritisnuti gumb za registraciju čime se otvara novi ekran. Ekran za registraciju zahtijeva od korisnika da upiše svoje korisničke podatke te unese sliku koja će predstavljati korisnikov avatar. Po završetku ispunjavanja forme korisnik pritišće na gumb REGISTER. Kako bi registracija bila uspješna potrebno je još izvršiti verifikaciju. Neposredno nakon registracije na korisnikov mail koji je upisao prilikom registracije šalje se verifikacijski kod koji mora upisati u ekran koji mu se otvorio u aplikaciji. Nakon upisanog ispravnog verifikacijskog koda registracija je uspješno završena te vas aplikacija vraća nazad na početni ekran za prijavu. Kako bi se korisnik prijavio u aplikaciju mora utipkati svoje podatke (e-mail i lozinka) koje je prethodno unio prilikom registracije te pritisnuti na tipku SIGN IN.





Nakon uspješne prijave otvara se glavni ekran aplikacije koji se sastoji od tri dijela: feed, gps i mailbox. Kada se otvori glavni ekran početno je uključen feed ekran. Pritiskom pri dnu ekrana na gumbe (FEED, GPS, INBOX) korisnik može prebacivati između tri glavna prikaza aplikacije.



- FEED prikaz omogućava korisniku da prelistava ponudu poslova koji još nisu obavljeni. Također, pritiskom na gumb PREGLED otvara se detaljan opis posla na koji smo kliknuli. Osim detaljnog opisa i dodatnih informacija korisniku se nudi opcija javljanja na posao klikom na gumb KONTAKTIRAJ POSLODAVCA. Tada se otvara aktivnost za komunikaciju s poslodavcem putem koje se možete dogovoriti za posao.
- GPS prikaz otvara kartu unutar aplikacije na kojoj su zapiknuti pinovi na lokacijama na kojima se nudi posao. Kartu je moguće približiti i udaljiti pomoću svima poznatih pokreta s dva prsta. Pritiskom na pojedini pin iznad njega pojavljuje se naziv posla. Pritiskom na naziv posla otvara se detaljan opis posla i dodatne informacije te gumb za javljanje na posao.
- Pritiskom na gumb INBOX otvara se korisnikov vlastiti sandučić u kojemu su svi njegovi prijašnji razgovori s ostalim korisnicima. Klikom na razgovor otvaraju se sve poruke razgovora s tom osobom. Ovim putem korisnik može komunicirati s ostalim korisnicima.

U gornjem lijevom kutu glavnog ekrana postoji gumb (tri vodoravne crtice) čijim pritiskom se otvaraju važne funkcionalnosti aplikacije kao što je filtriranje poslova, pregled „mojih“ poslova, dodavanje novog posla, uređivanje profila te odjava iz aplikacije.

- Pritiskom na filter otvara se novi ekran gdje se korisniku nude opcije po kojima može filtrirati poslove. Korisnik odabire kategoriju za koju želi pregledati poslove te pritisne na gumb FILTRIRAJ kako bi mu se prikazali samo oni poslovi koji njemu odgovaraju.
- Moji poslovi su popis poslova koje je korisnik kao poslodavac objavio u aplikaciji.

- Ako korisnik želi ponuditi novi posao ostalim korisnicima onda pritisne na gumb Dodaj posao te kada se otvori novi ekran ispuni sve potrebne informacije kako bi drugi korisnici znali o kakvom se poslu radi te jesu li zainteresirani za njega.
- Korisnik može u bilo kojem trenutku urediti svoj profil. Uređivanje profila podrazumijeva promjenu slike, imena, elektroničke pošte ili lozinke.
- Po završetku rada korisnik se može odjaviti iz aplikacije ili jednostavno izići iz nje pritiskom na (home) gumb mobilnog uređaja.

8. Zaključak i budući rad

Cilj projekta bila je android aplikacija koja bi trebala služiti kao oglasnik za kratke I načelno jednokratne poslove. Razvoj se odvijao u dvije faze

Prva faza je bila čisto teoretski pristup razvoju. Stvaranjem dijagrama obrazaca uporabe I dijagrama razreda, smo bili prisiljeni na razmišljanje
o svim mogućim problemima I željenim rezultatima. Ostalo je samo to implementirati.

Druga faza je predstavljala implementaciju sustava. Istaknuta timskim radom I trudom, ova faza je rezultirala željenim proizvodom.

Komunikacija između članova tima bila je konstantna, a ponajviše se odvijala uživo (sastancima), što je uvelike pridonijelo upoznatosti svih članova tima sa svim aspektima projekta i na međusobnom građenju dobrih odnosa. Prilikom rada na projektu, koristili smo sustav GitLab za upravljanje raznim verzijama koda.

Kao neka od mogućih proširenja koja sad nismo stigli implementirati bi bilo plaćanje preko aplikacije koje bi se moglo odvijati ili putem kreditne kartice ili putem paypala.

Sve u svemu, jako pozitivno iskustvo koje će nam sigurno biti korisno za naše buduće zanimanje. Naučili smo raditi u timu, koristiti GitLab I sve prednosti koje nam nosi stvaranje dokumentacije proizvoda prije same implementacije.
.

9. Popis literature

Popisati sve reference i literaturu koja je pomogla pri ostvarivanju projekta.

[1] Oblikovanje programske potpore, FER ZEMRIS, http://www.fer.hr/predmet/opp

[2] Oblikovanje programske potpore, FER ZEMRIS, http://www.zemris.fer.hr/predmeti/opp

[3] I. Sommerville, „Software engineering“, 8th ed, Addison Wesley, 2007.

[4] T.C.Lethbridge, R.Langaniere, „Object-Oriented Software Engineering“, 2nd ed. McGraw-Hill, 2005.

[5] Software engineering ,Rutgers University, http://www.ece.rutgers.edu/~marsic/Teaching/SE

[6] I. Marsic, „Software engineering book“, Department of Electrical and Computer Engineering, Rutgers University, http://www.ece.rutgers.edu/~marsic/books/SE

[7] Concepts: Requirements, http://www.upedu.org/upedu/process/gcncpt/co_req.htm

[8] UML 2 Class Diagram Guidelines, http://www.agilemodeling.com/style/classDiagram.htm

[9] Domain Class Diagram Modeling Standards and Guidelines, http://www.bced.gov.bc.ca/imb/downloads/classdiagramstandards.pdf

[10] Astah Community, http://astah.net/editions/community/

Dodatak A: Indeks (slika, dijagrama, tablica, ispisa kôda)

Dodatak B: Dnevnik sastajanja

Sastanak1: 16.10.2017. - FER

Prisutni: Karlo Butorac, Alen Carin, Josip Kalafatić, Tin Ivan Križ, Ivan Miličević, Jakov Vidak
Sažetak: Prvi sastanak i službeno upoznavanje unutar tima. Dogovor o mijenjanju teme i razrada iste.
Zaključci: Nakon upoznavanja svih sa svima, odlučili smo da ćemo uložiti vremena u to da radimo na vlastitoj temi. Složili smo PDF document na par stranica kao opis zadatka koji je kasnije bio prihvaćen kao naša tema.

Sastanak2: 25.10.2017. - FER

Prisutni: Alen Carin, Tin Ivan Križ, Ivan Miličević, Jakov Vidak
Sažetak: Daljnja razrada teme i upoznavanje članova s blagodatima Git sustava.
Zaključci: Instaliran Git na računalima svih članova, napravljen korisnički račun na GitLabu i napravljen Git repozitorij sa svim članovima prijavljenim kao developer članovi, te dodanim profesorom, asistentom i demonstratorom kao master članovi.

Sastanak3: 30.10.2017. - FER

Prisutni: Karlo Butorac, Alen Carin, Josip Kalafatić, Tin Ivan Križ, Ivan Miličević, Jakov Vidak
Sažetak: Dogovor o aplikaciji više u detalje. Organizacija ekrana, vrste korisnika, načelno koji entiteti su nam potrebni, željeni obrasci uporabe i dogovor o tome što nam je najbitnije implementirati. Podjela dokumentacije na članove.
Zaključci: Dogovoreno da korisnici moraju biti registrirani i verificirani da bi koristili našu aplikaciju, tj. da imamo tri vrste korisnika: neregistrirane, registrirane i admine. Dogovoreni obrasci uporabe, ekrani i okvirni entiteti. Dogovoreno da će komunikacija između korisnika biti omogućena kroz poruke slično emailu, a ne chat kao što je možda prvotno zamišljeno. Dogovoreno da je najmanje bitno implementirati GPS kartu. Dogovoreno da je sustav verifikacije ostvaren putem emaila i slike profila. Zaključena podjela rada je:

- Karlo Butorac – Arhitektura i dizajn sustava
- Alen Carin i Jakov Vidak – Dijagram stanja
- Josip Kalafatić i Ivan Miličević - Funkcionalni zahtjevi
- Tin Ivan Križ – Opis zadatka, ostali zahtjevi, dijagram razreda s opisom

Uz napomenu da svi mogu pomoći svima ako završe ranije.

Sastanak4: 6.11.2017. - FER

Prisutni: Karlo Butorac, Alen Carin, Josip Kalafatić, Tin Ivan Križ, Ivan Miličević, Jakov Vidak
Sažetak: Razgovor o dosadašnjem napretku i mogućim izmjenama. Dodatna razrada aplikacije. Dogovor o korištenju GitLab-a za dokumentaciju.

Zaključci: Svima je pojašnjena aplikacija do te razine da svi imaju jednoznačnu percepciju opsega i funkcionalnosti. Dogovoreno je da ćemo više koristiti Git za dokumentaciju, ali tako da se ne spremaju cijeli dokumenti nego da se spremaju fragmenti dokumentacije na kojoj se radilo i onda da ćemo kasnije spojiti sve fragmente u jedan dokument.

Sastanak5: 13.11.2017. - FER

Prisutni: Karlo Butorac, Alen Carin, Tin Ivan Križ, Ivan Miličević, Jakov Vidak
Sažetak: Razgovor o dosadašnjem napretku dokumentacije i budućoj raspodjeli raspoloživih ljudskih resursa.
Zaključci: Osobe koje su završile svoj dio sada mogu pomoći onima koji još nisu uspjeli ili imaju problema. Zaključena je sljedeća raspodjela preostalih poslova:

- Jakov Vidak – sekvencijski dijagrami
- Alen Carin i Tin Ivan Križ – dijagrami razreda i objekata
- Ivan Miličević i Josip Kalafatić – opisi sekvencijskih dijagrama i dijagram obrazaca uporabe
- Karlo Butorac – početak arhitekture s entitetima i opisima entiteta

Sastanak6: 7.12.2017. - FER

Prisutni: Karlo Butorac, Josip Kalafatić, Tin Ivan Križ, Ivan Miličević, Jakov Vidak
Sažetak: Razgovor o tome kako smo se snašli u Android studio-u. Male promijene u dogovoru oko podijele posla. Razgovor o stvaranju baze I gdje ćemo ju staviti.
Zaključci: Baza će biti dignuta na Azure.

Sastanak7: 19.12.2017. - FER

Prisutni: Karlo Butorac, Alen Carin, Tin Ivan Križ, Josip Kalafatić, Jakov Vidak,
Sažetak: Razgovor o napretku oko aplikacije i planu preko praznika
Zaključci: Napravljen dogovor o napretku aplikacije za vrijeme praznika.

Sastanak 8: 8.1.2018. - FER

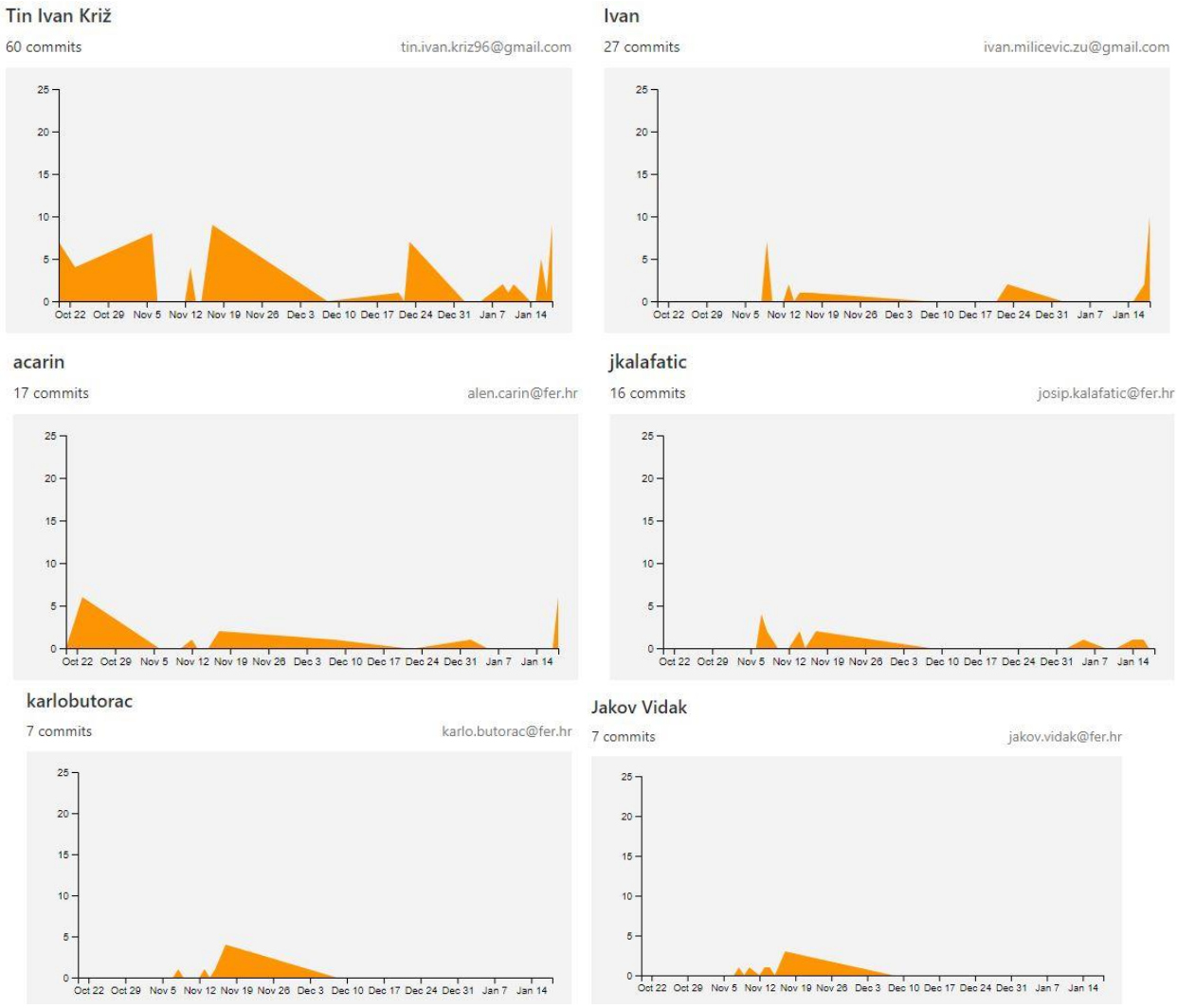
Prisutni: Karlo Butorac, Alen Carin, Tin Ivan Križ, Ivan Miličević, Jakov Vidak, Josip Kalafatić
Sažetak: Pregled tko je što napravio. Većina baze gotova, trebalo bi ju početi povezivati sa ostatkom aplikacije.
Zaključci: Počinjemo sa spajanjem baze i aplikacije. Nalaziti ćemo se više I imati radne seanse jer nam je puno lakše raditi kada smo zajedno, a I komunikacija između backend i frontenda će biti puno lakša..

Dodatak C: Prikaz aktivnosti grupe

Popis aktivnosti	Tin Ivan Križ	Karlo Butorac	Alen Carin	Josip Kalafatić	Ivan Miličević	Jakov Vidak
Upravljanje projektom	100%					
Opis projektnog zadatka	80%				20%	
Rječnik pojmova	14%	14%	14%	14%	14%	14%
Opis funkcionalnih zahtjeva				33%	33%	33%
Opis ostalih zahtjeva	100%					
Arhitektura i dizajn sustava						
Svrha, opći prioriteti i skica sustava		100%				
Dijagram razreda s opisom						
Dijagram objekata	100%					
Ostali UML dijagrami			50%			50%
Implementacija i korisničko sučelje						
Dijagram razmještaja			33	33		33
Korištene tehnologije i alati						100
Isječak programskog kôda		50				50
Ispitivanje programskog rješenja	14	14	14	14	14	14
Upute za instalaciju	25	25		25		25
Korisničke upute			33		33	33
Plan rada	100%					

Pregled rada i stanje ostvarenja	16	16	16	16	16	16
Zaključak i budući rad	100%					
Popis literature	16%	16%	16%	16%	16%	16%
Dodaci						
Indeks			100%			
Dnevnik sastajanja	100%					

Grafovi aktivnosti:



Dodatak D: Plan rada / Pregled rada i stanje ostvarenja

U rev. 1 smo odradili veliki dio dokumentacije i pripreme za samu implementaciju. U rev. 2 smo implementirali cijelu aplikaciju, te nadopunili dokumentaciju sa svim potrebnim informacijama, te mogućim izmjenama u odnosu na trenutni opis.