

Programsko inženjerstvo

Ak. god. 2020./2021.

Pomozi mi

Dokumentacija, Rev. 2

Grupa: *NULL*

Voditelj: *Iva Bokšić*

Datum predaje: *14. 1. 2021.*

Nastavnik: *Dunja Tounec*

Sadržaj

1 Dnevnik promjena dokumentacije	3
2 Opis projektnog zadatka	5
3 Specifikacija programske potpore	9
3.1 Funkcionalni zahtjevi	9
3.1.1 Obrasci uporabe	11
3.1.2 Sekvencijski dijagrami	27
3.2 Ostali zahtjevi	32
4 Arhitektura i dizajn sustava	33
4.1 Baza podataka	36
4.1.1 Opis tablica	36
4.1.2 Dijagram baze podataka	41
4.2 Dijagram razreda	42
4.3 Dijagram stanja	49
4.4 Dijagram aktivnosti	50
4.5 Dijagram komponenti	51
5 Implementacija i korisničko sučelje	52
5.1 Korištene tehnologije i alati	52
5.2 Ispitivanje programskog rješenja	53
5.2.1 Ispitivanje komponenti	53
5.2.2 Ispitivanje sustava	73
5.3 Dijagram razmještaja	84
5.4 Upute za puštanje u pogon	85
6 Zaključak i budući rad	86
Popis literature	88
Indeks slika i dijagrama	90

Dodatak: Prikaz aktivnosti grupe

91

1. Dnevnik promjena dokumentacije

Rev.	Opis promjene/dodatka	Autori	Datum
0.1	Napravljen predložak.	Roček	14.10.2020.
0.2	Opis projektnog zadatka.	Oreč	15.10.2020.
0.2.1	Započeta razrada UC dijagrama.	Bokšić Roček Ćurić Đaković	15.10.2020.
0.2.2	Izmjena predložka dokumentacije.	Bokšić	15.10.2020.
0.2.3	Opis obrazaca uporabe	Lipovac	15.10.2020.
0.2.4	Funkcionalni zahtjevi	Roček Jakas	15.10.2020.
0.3	Dijagram obrazaca uporabe	Roček	20.10.2020.
0.4	Ispravak opisa obrazaca uporabe.	Lipovac Bokšić	22.10.2020.
0.4.1	Unesen dnevnik sastajanja.	Bokšić	22.10.2020.
0.4.2	Izmjene na opisu i funkcijskim zahtjevima	Oreč	22.10.2020.
0.5	Sekvencijski dijagram za registraciju.	Đaković	23.10.2020.
0.5.1	Sekvencijski dijagram za zadavanje zahtjeva.	Jakas	24.10.2020.
0.6	Izmjena UC dijagrama	Oreč	30.10.2020.
0.7	Svi sekvencijski dijagrami	Ćurić	30.10.2020.
0.8	Izmjena opisa obrazaca uporabe	Lipovac	30.10.2020.
0.8.1	Opisi sekvencijskih dijagrama	Bokšić	03.11.2020.
0.8.2	Dodane tablice za bazu. Dodana slika baze.	Bokšić	03.11.2020.
0.8.3	Unos dnevnika sastajanja.	Bokšić	03.11.2020.
0.9	Izmjena opisa zadatka i dodani ostali zahtjevi	Oreč	04.11.2020.
0.9.1	Opis tablica baze podataka	Lipovac Ćurić	04.11.2020.
0.9.2	Opis arhitekture sustava	Bokšić	05.11.2020.

Rev.	Opis promjene/dodatka	Autori	Datum
0.9.3	Izmijenjena baza podataka i tablice entiteta	Bokšić	11.11.2020.
0.10	Završeni dijagrami razreda i opisi istih	Oreč Lipovac	11.11.2020
1.0	Verzija samo s bitnim dijelovima za 1. ciklus	Bokšić	12.11.2020.
1.1	Dodana implementacija i korisničko sučelje	Ćurić	28.12.2020.
1.1.1	Promjene na bazi i arhitekturi	Roček	28.12.2020.
1.2	Dodani dijagrami stanja, komponenti, razmještaja i aktivnosi	Ćurić	30.12.2020.
1.2.1	Prepravljeni dijagrami i dokumentacija vezana uz njih	Ćurić	4.1.2021.
1.3	Dokumentacija JUnit testova	Oreč, Lipovac	6.1.2021.
1.3.1	Promjene na bazi i provjera ostatka dokumentacije	Roček	6.1.2021.
1.4.	Selenium testiranje	Oreč, Lipovac	10.1.2021.
1.5	Prepravljeni dijagrami stanja i aktivnosti	Ćurić	12.1.2021.
1.5.1	Popravci cijele dokumentacije	Bokšić	12.1.2021.
1.6	Pisanje zaključka i budućeg rada	Lipovac, Oreč	13.1.2021.
1.7	Izmjena slika dijagrama razreda	Roček, Bokšić	14.1.2021.
1.8	Izmjena slika UC dijagrama i opisa istih	Bokšić	14.1.2021.
1.9	Puštanje u pogon	Roček	14.1.2021.
1.9.1	Dijagrami pregleda promjena	Bokšić	14.1.2021.
2.0	Konačni tekst predložka dokumentacije	Bokšić	14.1.2021.

2. Opis projektnog zadatka

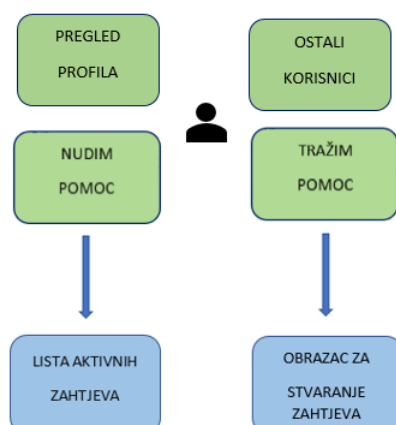
U okviru projekta za razvoj programske potpore cilj nam je napraviti web aplikaciju “Pomozi mi” čiji je primarni zadatak mogućnost traženja pomoći te pružanje same usluge od strane ljudi koji žele pomagati. Potencijalna korist mogla bi biti i samo zbližavanje korisnika te širenje osjećaja sigurnosti i povezanosti same zajednice. Na ovaj način mnogi ljudi kojima je odlazak u trgovinu ili košnja trave veliki izazov pronalaze način da dođu do osoba koji svoj višak vremena žele iskoristiti za pomaganje potrebitijima. Iako danas postoje mnoge društvene mreže, ova stranica nema nepotreban sadržaj te je maksimalno prilagođena svojoj funkciji uz jednostavno korištenje.

Prvi korak prema korištenju usluga same aplikacije je prijava u sustav. Svaki korisnik aplikacije ukoliko nema otvoreni profil (javni korisnik) se mora registrirati. Prilikom registracije ostavlja osnovne podatke o sebi :

- *Ime*
- *Prezime*
- *Lozinka*
- *Adresa*
- *e-mail*

Za svako sljedeće korištenje dovoljna je prijava pomoću e-mail adrese i lozinke.

Nakon prijave korisnik može pristupiti pregledu svog profila, listi korisnika ili odabrati neku od dvije aktivnosti same aplikacije: nudi li pomoć ili je traži te s obzirom na odabir se otvara odgovarajuća stranica.



Slika 2.1: odabir nakon prijave

Kada korisnik traži pomoć ispunjava obrazac za stvaranje zahtjeva. Na njemu unosi sljedeće podatke:

- Kratki opis posla
- Kontakt: broj mobitela / e-mail
- Opcionalno datum i vrijeme
- Opcionalno lokacija

Ukoliko je lokacija važna za izvršavanje usluge može se dohvatiti s uređaja ili označiti na karti.

Nakon objave korisnik je autor zahtjeva te je u mogućnosti zahtjev obrisati trajno iz baze podataka ili ga trenutno blokirati. Objavljeni zahtjev se nalazi na listi aktivnih zahtjeva.

Svi koji žele pomoći, na stranici aktivnih zahtjeva pronalaze koju uslugu bi mogli izvršiti. Prilikom pregleda liste moguće je lako dohvatiti profil korisnika koji je objavio zahtjev. Svakom korisniku se prikazuju samo zahtjevi čija je lokacija unutar jednog kilometra od uređaja korisnika te zahtjevi za čije je izvršavanje lokacija nebitna. Kada korisnik pronade zadovoljavajuću ponudu usluge odabire izvršavanje čime postaje izvršitelj zahtjeva.



Slika 2.2: proces izvršavanja zahtjeva

Odabirom posla šalje se automatska obavijest autoru zahtjeva o potencijalnom izvršitelju. Izvršitelj i autor mogu dodatno pojasniti sam posao i ostale detalje izvedbe. Nakon što je usluga obavljena autor zahtjev označava kao izvršen te se briše s liste aktivnih.

Nakon izvršenja posla moguće je ocijeniti drugog korisnika ocjenama 1-5 te ostaviti komentar. Ocjenjivanje je moguće i kad si korisnici nisu međusobno pomagali, a sve ocjene i komentari su vidljivi na profilu korisnika.

Svaki korisnik svoj profil može vidjeti u svakom trenutku, a ostale korisnike može pretražiti ili ih dohvatiti među aktivnim zahtjevi. Na tuđim profilima su vidljivi :

- Ime
- Prezime
- Ocjena korisnika
- "lanac povjerenja"

"Lanac povjerenja" prikazuje kako su ljudi koje sami ocijenimo visoko ocijenili profil koji trenutno gledamo. Na taj način korisnik se sigurnije osjeća za daljnju komunikaciju s korisnikom profila.

Kada korisnik gleda vlastiti profil ima mogućnost pristupa listi vlastitih zahtjeva. Unutar liste vlastitih zahtjeva bira pregled izvršenih ili aktivnih kojima može dodatno upravljati. Zahtjevi su sortirani vremenski, a moguće su još dodatne opcije filtriranja poput kategorije i lokacije. Svi zahtjevi koji su aktivni, izvršeni ili blokirani ostaju pohranjeni u bazi podataka.

Osim korisnika važnu ulogu imaju administratori koji su dodijeljeni za određenu geografsku lokaciju. Oni brinu da sadržaj koji se objavljuje ne bude lažan ili opasan te imaju mogućnost brisanja zahtjeva i profila. Oni mogu privremeno ili trajno blokirati sve korisnike aplikacije.

Ova aplikacija se lako po potrebi može proširiti i na druga područja uz male preinake vrsta zahtjeva. Također ukoliko se uoči potreba moguće je lako stvoriti i mobilnu aplikaciju koja bi još dodatno olakšala sam pristup korisnicima.

3. Specifikacija programske potpore

3.1 Funkcionalni zahtjevi

Dionici:

1. Razvojni tim
2. Administrator
3. Korsinik
4. Vanjski suradnici - CROZ

Aktori i njihovi funkcionalni zahtjevi:

1. Korisnik (inicijator) može:

(a) biti javni korisnik

i. može se registrirati

(b) biti registrirani korisnik

i. prijavljuje se u sustav

ii. pregled liste korisnika

iii. pregled vlastitog profila

iv. zadavanje zahtjeva za pomoć

v. prikaz liste aktivnih zahtjeva

vi. razmjenjivati notifikacije kada izvršitelj odabere zahtjev

vii. dohvatiti profil drugog korisnika u trenutku pregleda zahtjeva

viii. dohvatiti profile ostalih korisnika iz liste korisnika

ix. ocjenjivati i komentirati druge korisnike

x. vidjeti "lanac povjerenja"

xi. pregledati listu svojih izvršenih i ponuđenih zahtjeva

2. Administrator (inicijator) može:

- (a) pregled liste korisnika
- (b) prikaz liste aktivnih zahtjeva
- (c) kontrolirati sadržaj koji se objavljuje
- (d) brisanje zahtjeva
- (e) blokiranje korisnika pristupu aplikaciji

3. Baza podataka (sudionik) može:

- (a) spremanje podataka o profilima, zahtjevima, razmjeni notifikacija

4. Poslužitelj (sudionik) može:

- (a) spremanje podataka o profilima, zahtjevima, razmjeni notifikacija

3.1.1 Obrasci uporabe

Opis obrazaca uporabe

UC1 - Registracija

- **Glavni sudionik:** Javni korisnik
- **Cilj:** Stvoriti korisnički račun za pristup sustavu
- **Sudionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
 1. Korisnik odabire opciju za registraciju
 2. Korisnik unosi potrebne korisničke podatke
 3. Korisnik prima obavijest o uspješnoj registraciji
- **Opis mogućih odstupanja:**
 - 2.a Odabir već zauzetog korisničkog imena i/ili e-maila, unos korisničkog podatka u nedozvoljenom formatu ili unos neispravnoga e-maila
 1. Sustav obavještava korisnika o neispravnom unosu i ponovno ga vraća na stranicu za registraciju
 2. Korisnik mijenja potrebne podatke i završava unos ili odustaje od registracije

UC2 -Prijava

- **Glavni sudionik:** Registrirani korisnik
- **Cilj:** Dobiti pristup korisničkom sučelju
- **Sudionici:** Baza podataka
- **Preduvjet:** Registracija
- **Opis osnovnog tijeka:**
 1. Unos korisničkog imena i lozinke
 2. Potvrda o ispravnosti unesenih podataka
 3. Pristup korisničkim funkcijama
- **Opis mogućih odstupanja:**
 - 1.a Neispravan unos korisničkog imena ili lozinke
 1. sustav obavještava korisnika o neispravnom unosu i ponovno ga vraća na stranicu za prijavu

UC3 -Zadavanje zahtjeva za pomoć

- **Glavni sudionik:** Registrirani korisnik
- **Cilj:** Kreirati novi zahtjev za pomoć
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav
- **Opis osnovnog tijeka:**
 1. Korisnik odabire opciju da traži pomoć
 2. Pojavljuje se obrazac za kreiranje zahtjeva za pomoći
 3. Korisnik odabire opciju "Objavi zahtjev"

UC3.1 - Zadavanje lokacije

- **Glavni sudionik:** Karta
- **Cilj:** Dohvatiti trenutnu lokaciju korisnika
- **Sudionici:** Registrirani korisnik
- **Preduvjet:** Korisnik je prijavljen u sustav, zadaje zahtjev za pomoć
- **Opis osnovnog tijeka:**
 1. Korisnik omogućava aplikaciji pristup vlastitoj lokaciji
 2. Korisnik potvrđuje očitanoj lokaciju
 3. Očitana lokacija se sprema u bazu
- **Opis mogućih odstupanja:**
 - 1.a Korisnik ne odobrava pristup njegovoj lokaciji, ona nije dostupna ili je krivo očitana
 1. Aplikacija nudi opciju postavljanja lokacije pomoću karte

UC4 - Pretraživanje korisnika

- **Glavni sudionik:** Registrirani korisnik
- **Cilj:** Pregledati listu registriranih korisnika
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav
- **Opis osnovnog tijeka:**
 1. Korisnik u bilo kojem trenutku, pritiskom na gumb može pretražiti registrirane korisnike
 2. Iz baze se dohvaća lista profila korisnika

UC5 - Prikaz liste aktivnih zahtjeva

- **Glavni sudionik:** Registrirani korisnik
- **Cilj:** Pregledati aktivne zahtjeve i ponuditi pomoć
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav
- **Opis osnovnog tijeka:**
 1. Korisnik odabire ulogu izvršitelja zahtjeva
 2. Iz baze se dohvaća lista aktivnih zahtjeva autora koji trebaju pomoć te se nalaze unutar jednog kilometra od lokacije izvršitelja
- **Opis mogućih odstupanja:**
 - 2.a Ne postoje aktivni zahtjevi u označenom području
 1. Otvara se mogućnost proširenja liste na veće geografsko područje

UC6 - Profil

- **Glavni sudionik:** Registrirani korisnik
- **Cilj:** Dobiti pristup korisničkom profilu
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav
- **Opis osnovnog tijeka:**
 1. korisnik odabire opciju pregleda profila
 2. Iz baze se dohvaća profil i osobni podaci te se prikazuje korisniku

UC6.1 - Pregled vlastitog profila

- **Glavni sudionik:** Registrirani korisnik
- **Cilj:** Pregledati vlastiti profil
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav
- **Opis osnovnog tijeka:**
 1. Korisnik odabire opciju pregleda vlastitog profila
 2. Iz baze se dohvaća korisnikov profil i vlastiti podaci

UC6.1.1 - Dodatni izvještaji

- **Glavni sudionik:** Registrirani korisnik, administrator
- **Cilj:** Pregled dodatnih informacija o korisniku : ocjena, broj izvršenih i broj zadanih zahtjeva, rang na listi za najboljeg pomagača godine
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav
- **Opis osnovnog tijeka:**
 1. Korisnik odabire opciju na profilu za pregled dodatnih informacija
 2. Otvara mu se prozor na kojem su vidljive dodatne informacije korisnika

UC6.1.2 - Pregled aktivnih i izvršenih zahtjeva

- **Glavni sudionik:** Registrirani korisnik
- **Cilj:** Pregled vlastitih zahtjeva
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav i kreirao je zahtjeve
- **Opis osnovnog tijeka:**
 1. Na vlastitom profilu korisnika vidljivi su mu svi zahtjevi podijeljeni u kategorije.

UC6.1.3 - Kandidiranje

- **Glavni sudionik:** Registrirani korisnik
- **Cilj:** Kandidiranje za pomagača godine
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav
- **Opis osnovnog tijeka:**
 1. Na vlastitom profilu korisnik aktivira ili deaktivira gumb za kandidiranje.

UC6.2 - Pregled profila drugog korisnika

- **Glavni sudionik:** Registrirani korisnik
- **Cilj:** Dobiti pristup tuđim korisničkim profilima
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen, prikaz liste aktivnih zahtjeva, pregled liste korisnika
- **Opis osnovnog tijeka:**

1. Otvara se opcija pregleda profila autora zahtjeva
2. Iz baze se dohvaća profil o autoru zahtjeva

UC6.2.1 - Lanac povjerenja

- **Glavni sudionik:** Registrirani korisnik
- **Cilj:** Moguće je vidjeti je li neki korisnik kojega smo prije pozitivno ocijenili, pozitivno ocijenio korisnika čiji profil gledamo
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen, pregled profila drugog korisnika
- **Opis osnovnog tijeka:**
 1. Na profilu se prikazuje opcija "lanca povjerenja"
- **Opis mogućih odstupanja:**
 - 1.a Korisnik koji gleda drugi profil nije još nikoga ocijenio
 1. U prostoru za "lanac povjerenja" se ništa ne ispisiuje

UC7 - Pregled zahtjeva

- **Glavni sudionik:** Administrator
- **Cilj:** Pregledati aktivne zahtjeve
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen kao administrator
- **Opis osnovnog tijeka:**
 1. Administrator odabire opciju pregleda liste aktivnih zahtjeva
 2. Iz baze se dohvaća lista aktivnih zahtjeva autora koji trebaju pomoć te se nalaze unutar jednog kilometra od lokacije administratora
- **Opis mogućih odstupanja:**
 - 2.a Ne postoje aktivni zahtjevi u označenom području
 1. Otvara se mogućnost proširenja liste na veće geografsko područje

UC7.1 - Brisanje zahtjeva

- **Glavni sudionik:** Administrator
- **Cilj:** Obrisati zahtjev
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je registriran i prijavljen kao administrator
- **Opis osnovnog tijeka:**
 1. Administrator u listi zahtjeva odabire željeni zahtjev
 2. Odabire opciju "Obriši zahtjev"
 3. Zahtjev se uklanja iz baze podataka

UC8 - Pregled korisnika

- **Glavni sudionik:** Administrator
- **Cilj:** Pregledati registrirane korisnike
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je registriran i prijavljen kao administrator
- **Opis osnovnog tijeka:**
 1. Administrator odabire opciju pregledavanja korisnika
 2. Iz baze se dohvaća lista svih ispravno registriranih korisnika s osobnim podacima

UC8.1 - Blokiranje korisnika

- **Glavni sudionik:** Administrator
- **Cilj:** Mogućnost blokiranja korisnika
- **Sudionici:** -
- **Preduvjet:** Korisnik je registriran i prijavljen kao administrator, administratorski pregled korisnika
- **Opis osnovnog tijeka:**
 1. Pri pregledu korisnika, administratoru se omogućuje blokiranje korisničkih računa

UC9 - Dodjela administrativnog područja

- **Glavni sudionik:** Administrator
- **Cilj:** dodjela administrativnog područja prema geografskoj lokaciji
- **Sudionici:** Sustav
- **Preduvjet:** Korisnik je registriran i prijavljen kao administrator
- **Opis osnovnog tijeka:**
 1. Administratoru preko vlastite lokacije sustav dodjeljuje administrativno područje

UC10 - Ocjenjivanje

- **Glavni sudionik:** Registrirani korisnik
- **Cilj:** Ocjenjivanje korisnika kako bi drugi korisnici vidjeli ukupnu ocjenu
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav
- **Opis osnovnog tijeka:**
 1. Korisnik odabire opciju za ocjenjivanje između 1 i 5
 2. Pohranjuje se dodjeljena ocjena u bazu podataka

UC10.1 - Komentiranje

- **Glavni sudionik:** Registrirani korisnik
- **Cilj:** Davanje subjektivne predodžbe o primljenim/zadanim uslugama u sh-vrhu unaprijeđenja istih
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav
- **Opis osnovnog tijeka:**
 1. Korisnik piše komentar
 2. Komentar se pohranjuje u bazu podataka

UC11 - Filtriranje

- **Glavni sudionik:** Registrirani korisnik
- **Cilj:** Dobiti pregledniju listu
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav, pretraživanje liste aktivnih zah-tjeva ili korisnika
- **Opis osnovnog tijeka:**

1. Nad listom korisnika ili zahtjeva odabire se filtriranje.
2. Filtrirana lista se prikazuje korisniku

UC11.1 - Filtriranje po radijusu udaljenosti

- **Glavni sudionik:** Registrirani korisnik
- **Cilj:** Preglednija lista aktivnih zahtjeva ili korisnika s obzirom na radijus udaljenosti
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav, pretraživanje liste aktivnih zahtjeva ili korisnika
- **Opis osnovnog tijeka:**
 1. Nad listom korisnika ili zahtjeva odabire se filtriranje po radijusu udaljenosti tako da korisnik unosi radijus s obzirom na njegovu lokaciju
 2. Filtrirana lista se prikazuje korisniku
- **Opis mogućih odstupanja:**
 - 1.a Sustav ne može očitati korisnikovu lokaciju
 1. Korisniku se prikazuju samo zahtjevi bez lokacije

UC11.2 - Filtriranje po kategorijama

- **Glavni sudionik:** Registrirani korisnik
- **Cilj:** Preglednija lista aktivnih zahtjeva ili korisnika s obzirom kategorije
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav, pretraživanje liste aktivnih zahtjeva ili korisnika
- **Opis osnovnog tijeka:**
 1. Nad listom korisnika ili zahtjeva odabire se filtriranje po kategorijama koje obuhvaćaju datum zadavanja zahtjeva, vrstu zahtjeva, ocjenu korisnika
 2. Filtrirana lista se prikazuje korisniku

UC12 - Upravljanje aktivnim zahtjevima

- **Glavni sudionik:** Registrirani korisnik
- **Cilj:** Kontroliranje vlastitih zahtjeva
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav i nalazi se na vlastitom profilu
- **Opis osnovnog tijeka:**

1. Korisnik ima pregled vlastitih zahtjeva nad kojim pojedinačno može vršiti akcije

UC12.1 -Brisanje zahtjeva

- **Glavni sudionik:** Registrirani korisnik
- **Cilj:** Obrisati kreirani zahtjev
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav i nalazi se na vlastitom profilu
- **Opis osnovnog tijeka:**
 1. Klikom na gumb „Obriši“ korisnik briše kreirani zahtjev.

UC12.2 - Blokiranje zahtjeva

- **Glavni sudionik:** Registrirani korisnik
- **Cilj:** Blokirati kreirani zahtjev da ga drugi korisnici ne mogu odabrati za izvršavanje
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav, nalazi se na vlastitom profilu i ima kreirani zahtjev
- **Opis osnovnog tijeka:**
 1. Klikom na gumb „Blokiraj“ korisnik blokira zahtjev i on nestaje s liste aktivnik i nije ga moguće odabrati za izvršavanje.

UC12.3 - Označi izvršen

- **Glavni sudionik:** Registrirani korisnik
- **Cilj:** Pohranjivanje podatka da je zahtjev izvršen i on postaje trajno neaktivan
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav, zahtjev je odabran za izvršavanje od strane korisnika
- **Opis osnovnog tijeka:**
 1. Nakon što je korisnik izvršio zadatak odabire gumb „Označi izvršen“.

UC13 - Odabir zahtjeva za izvršavanje

- **Glavni sudionik:** Registrirani korisnik
- **Cilj:** Nestajanje zahtjeva s liste aktivnih zahtjeva i početak razmjene notifikacija među korisnicima
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav, ima uvid u pregled zahtjeva
- **Opis osnovnog tijeka:**
 1. Između liste svih aktivnih zahtjeva korisnik odabire one koje je spreman izvršiti.
 2. Zahtjev postaje neaktivan.
 3. Slijedi dogovor korisnika oko zahtjeva

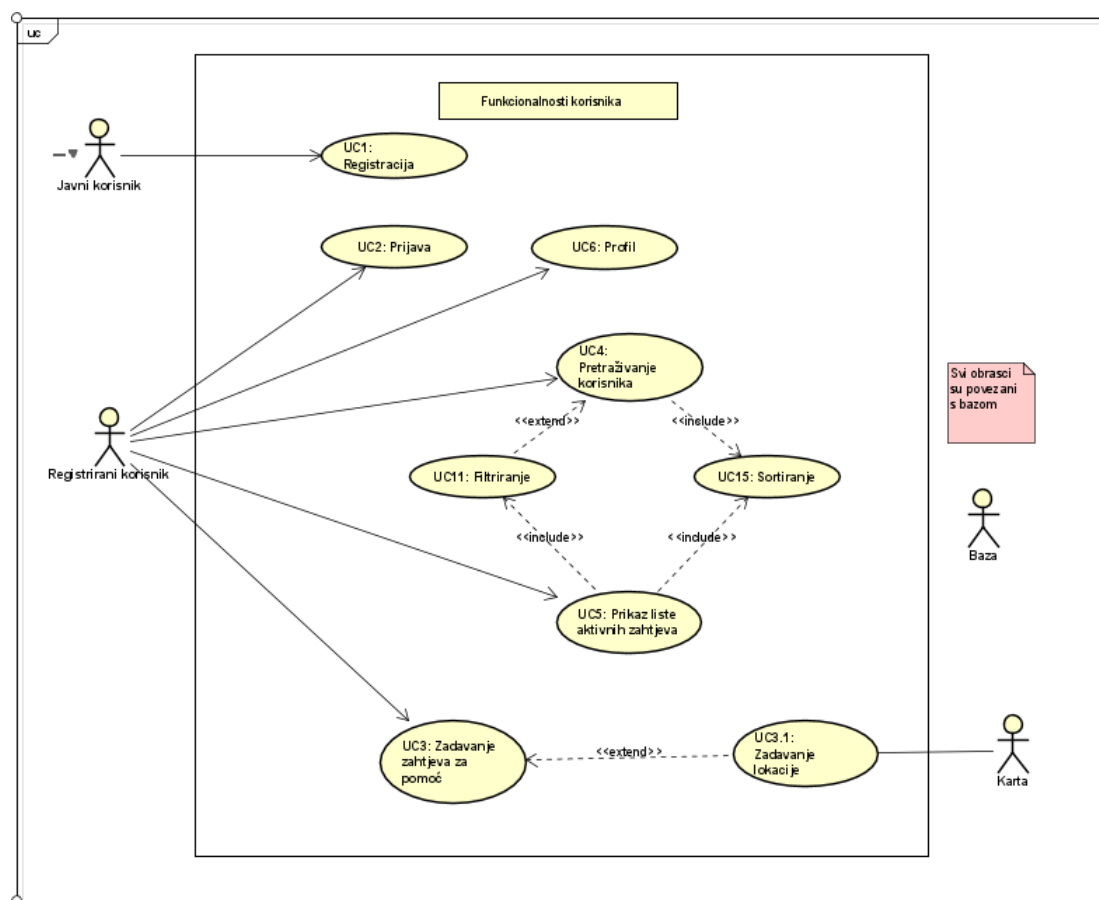
UC14 - Razmjena notifikacija

- **Glavni sudionik:** Registrirani korisnik
- **Cilj:** Dogovor oko izvršavanja zahtjeva
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav, korisnik je odabrao zahtjev za izvršavanje, autor zahtjeva ga je potvrdio za izvršitelja
- **Opis osnovnog tijeka:**
 1. Nakon odabira zahtjeva kojeg korisnik želi izvršiti omogućuje se razmjena notifikacija radi dogovora oko izvršenja zahtjeva

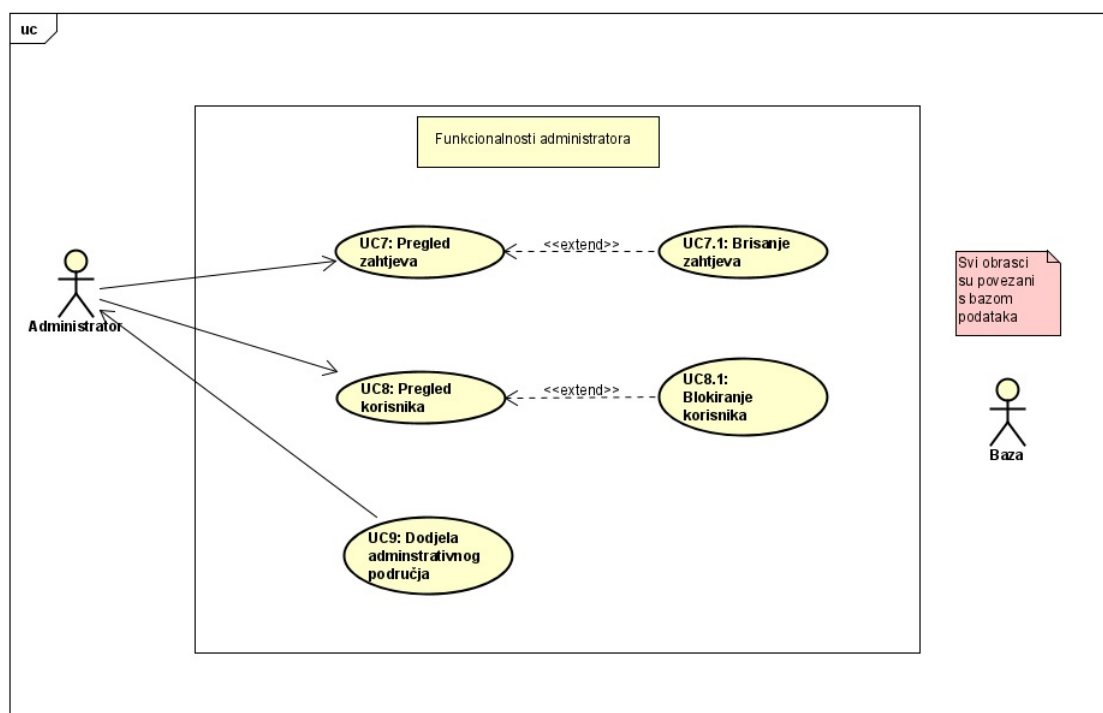
UC15 - Sortiranje

- **Glavni sudionik:** Registrirani korisnik, administrator
- **Cilj:** Urediti prikaz liste aktivnih zahtjeva radi bolje preglednosti.
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav na pregledu je liste zahtjeva ili korisnika
- **Opis osnovnog tijeka:**
 1. Korisnik odabire jednu od opcija za sortiranje
 2. Lista aktivnih zahtjeva se sortira s obzirom na odabir

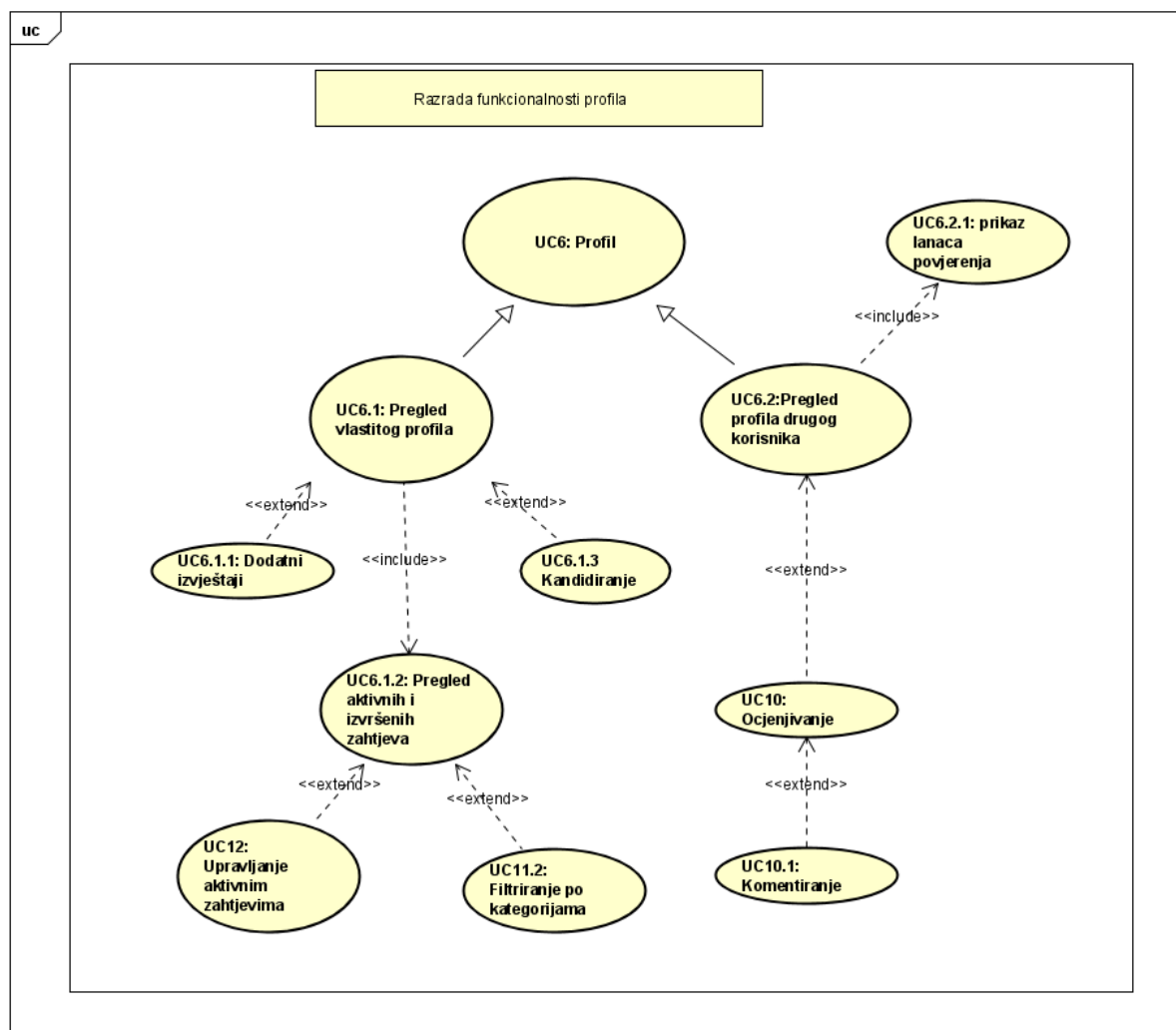
Dijagrami obrazaca uporabe



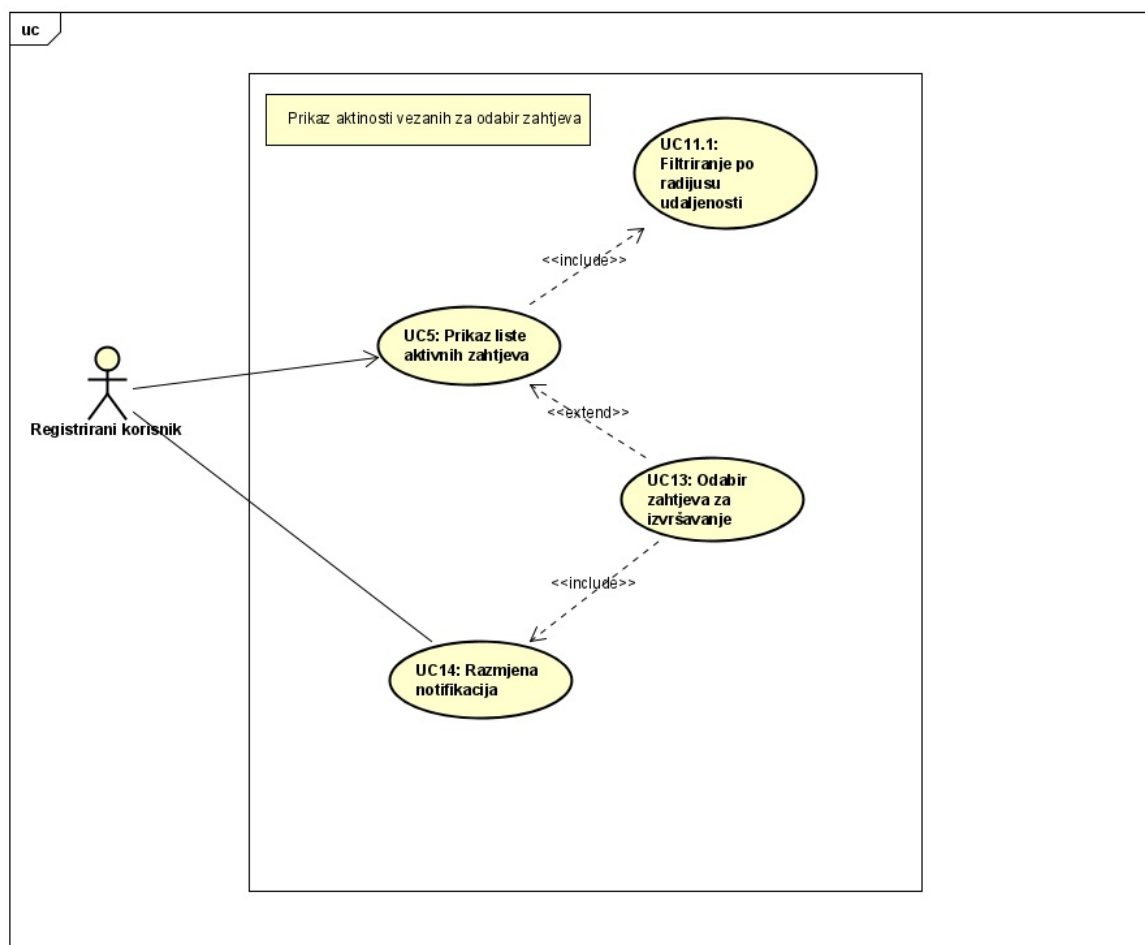
Slika 3.1: Dijagram obrasca uporabe, funkcionalnosti korisnika



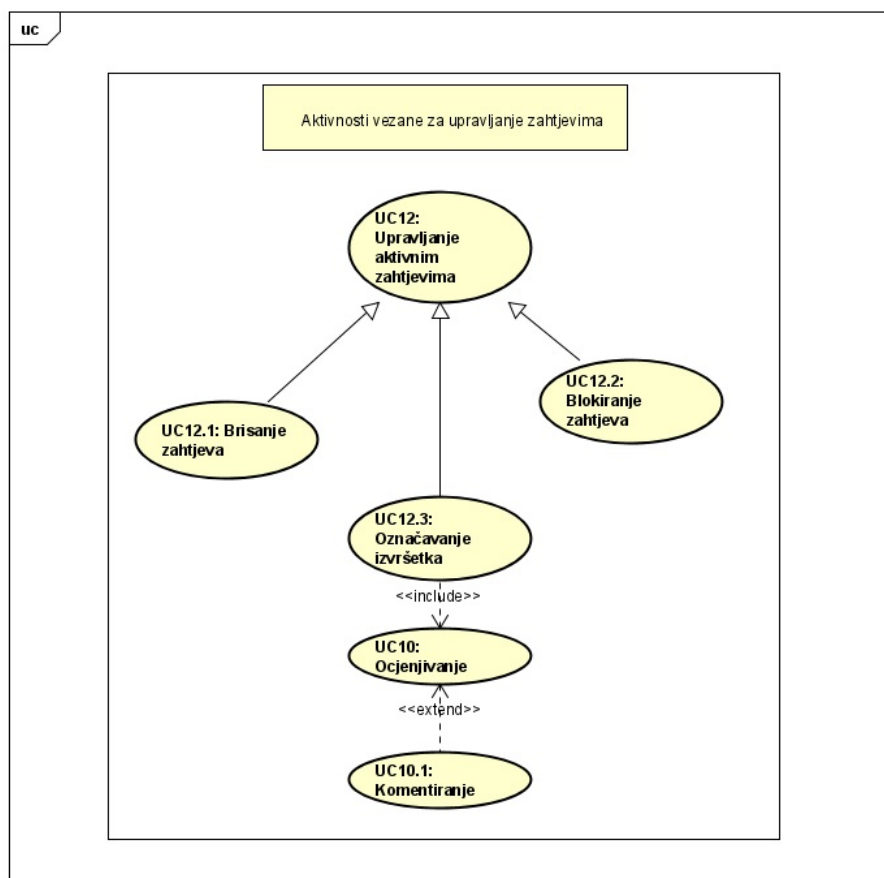
Slika 3.2: Dijagram obrasca uporabe, funkcionalnosti administratora



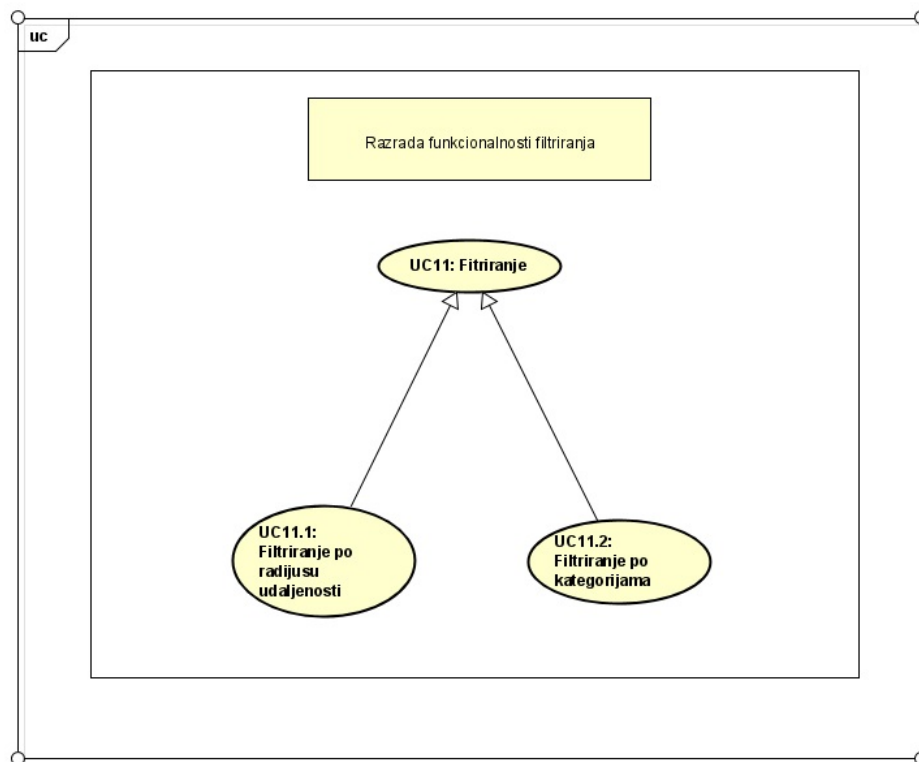
Slika 3.3: Dijagram obrasca uporabe, razrada funkcionalnosti profila



Slika 3.4: Dijagram obrasca uporabe, prikaz aktivnosti vezanih za upravljanje zahtjeva



Slika 3.5: Dijagram obrasca uporabe, prikaz aktivnosti vezanih za odabir zahtjeva

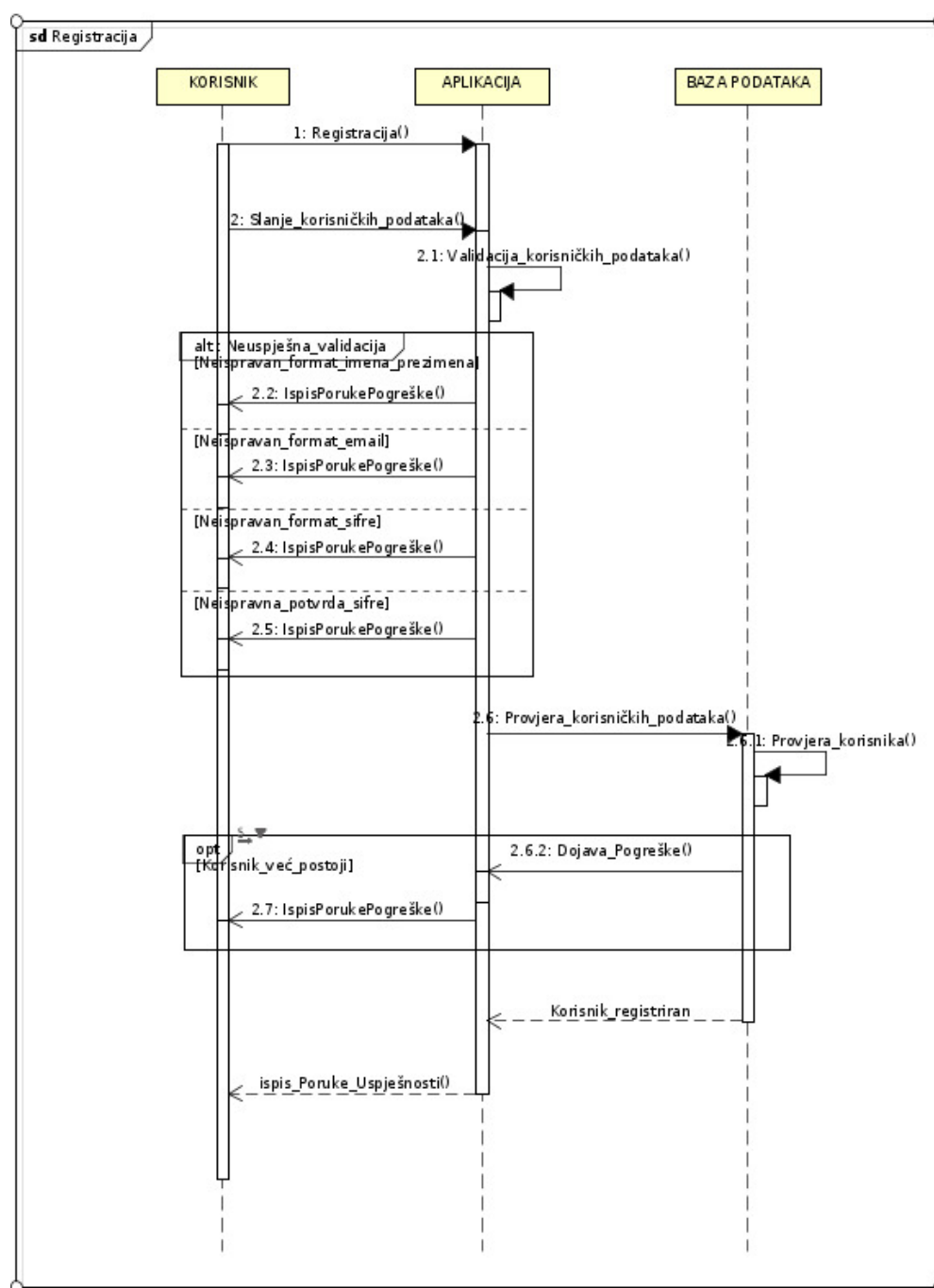


Slika 3.6: Dijagram obrasca uporabe, razrada funkcionalnosti filtriranja

3.1.2 Sekvencijski dijagrami

Obrazac uporabe UC1 - Registracija

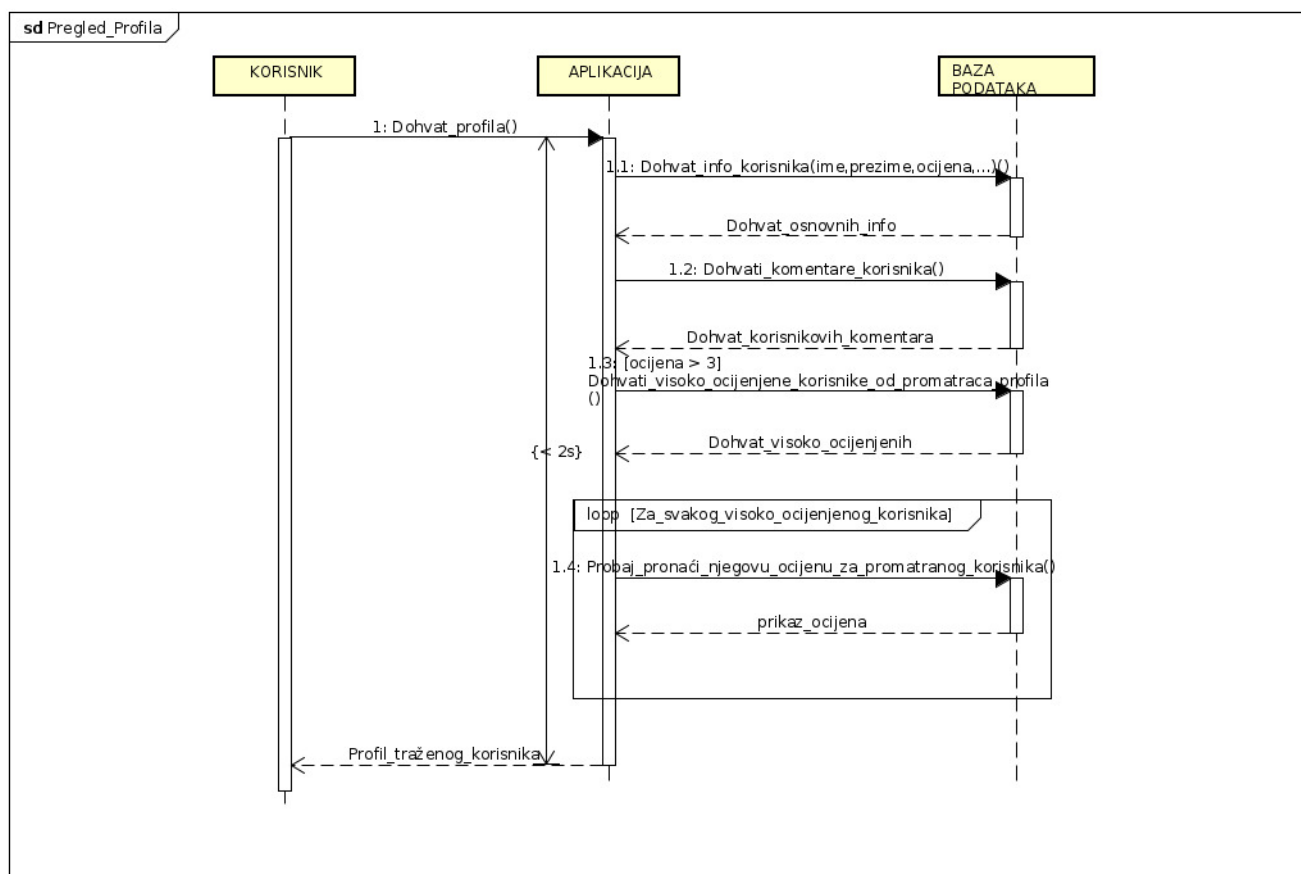
Korisnik šalje zahtjev za registracijom na sustav za primanje ili pružanje pomoći. Upisom registracijskih podataka, poslužitelj validira primljene podatke od strane korisnika. Ako je došlo do neuspješne validacije, poslužitelj o tome obavještava korisnika ispisom poruke pogreške. Ukoliko su podaci ispravno validirani na strani poslužitelja, on prosljeđuje podatke prema bazi podataka. Baza podataka provjerava postojanje korisnika s identičnim ključnim podacima (email, lozinka). U slučaju postojanja korisnika s identičnim ključnim podacima, baza podataka dojavljuje pogrešku sustavu, koji ona poruku prosljeđuje korisniku. Ako je provjera na strani baze uspješna, baza dojavljuje sustavu da je korisnik uspješno registriran, dok sustav korisniku ispisuje poruku uspješnosti.



Slika 3.7: Sekvencijski dijagram za UC1

Obrazac uporabe UC6 - Profil

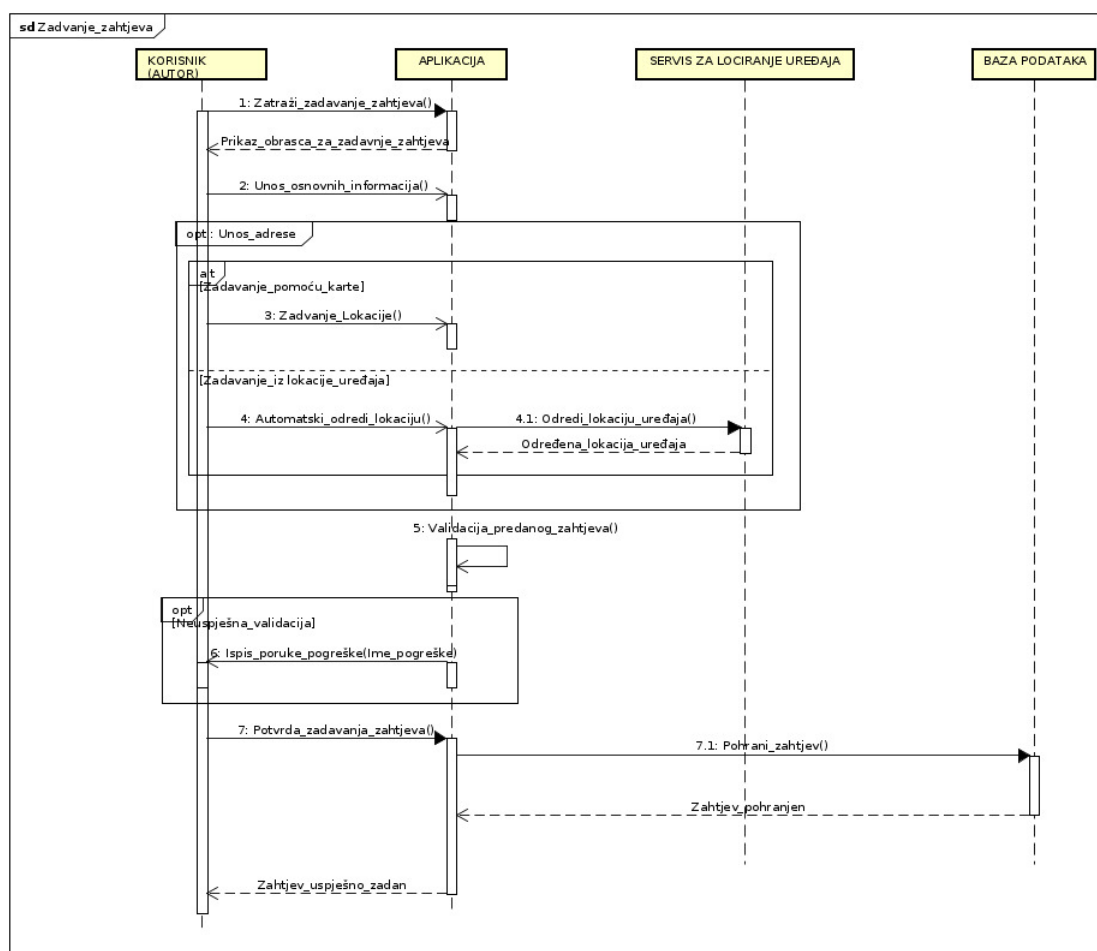
Korisnik prema poslužitelju šalje zahtjev za prikazom korisničkog profila. Poslužitelj iz baze dohvaća osnovne informacije o korisniku, komentare korisnika te dohvaća visoko ocijenjene korisnike korisnika koji je zatražio prikaz profila. Poslužitelj na temelju visoko ocijenjenih korisnika pretražuje eventualne njihove ocijene prema korisniku čiji se profil želi dohvatiti i tako izgrađuje lanac povjerenja. Poslužitelj vraća osnovne informacije, komentare te lanac povjerenja korisniku koji je zatražio dohvat profila.



Slika 3.8: Sekvencijski dijagram za UC6

Obrazac uporabe UC3 - Zadavanje zahtjev za pomoć

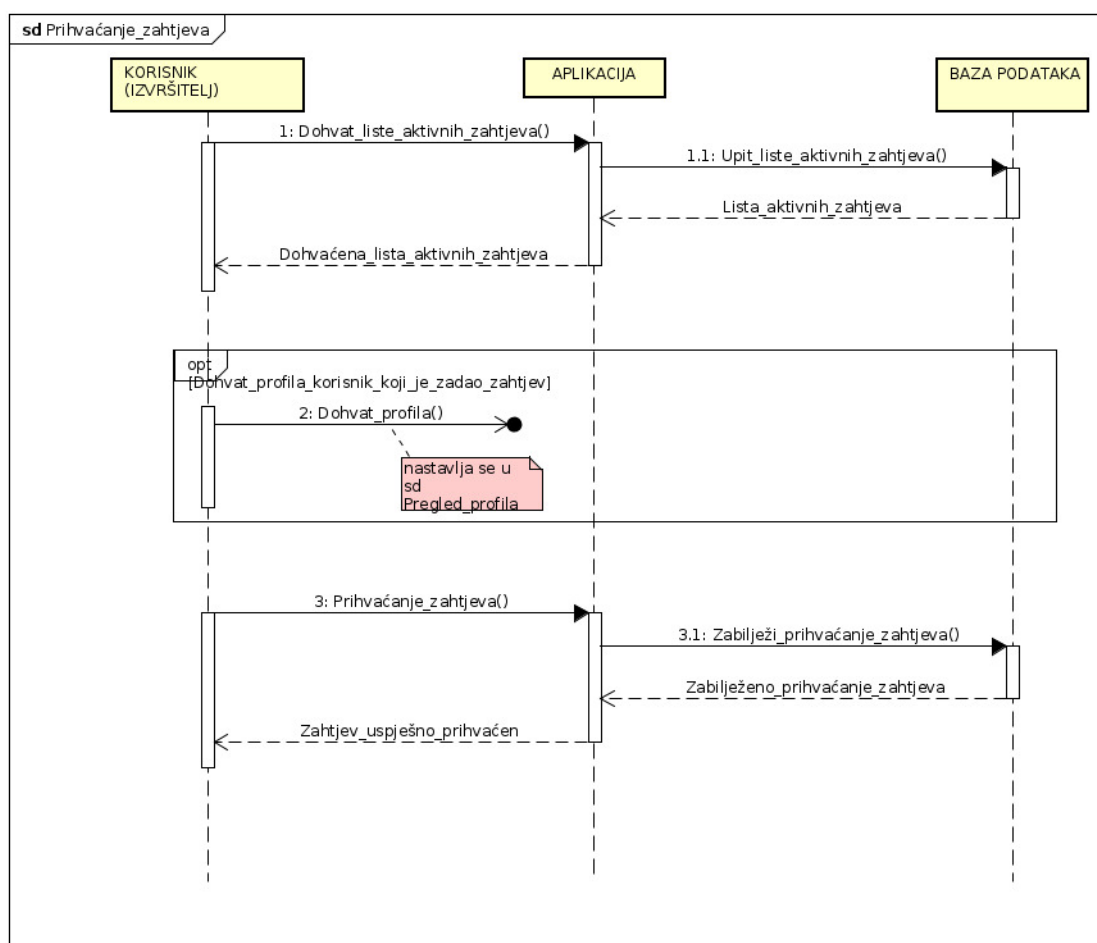
Korisnik(autor) želi zadati novi zahtjev za pomoć. Unosi osnovne informacije koje su potrebne za zahtjev te opcionalno unosi lokaciju ako ju smatra bitnom za zahtjev. Korisniku željenu lokaciju predaje poslužitelj u obliku geografskih koordinata ili poslužitelj može automatski pomoću servisa za određivanje lokacije odrediti geografsku lokaciju uređaja s kojeg je zahtjev zadan. Poslužitelj validira unesene informacije, ukoliko su unesene informacije neuspješno validiran poslužitelj dojavljuje pogrešku korisniku. Na kraju korisnik potvrđuje svoj zahtjev za pomoć. Poslužitelj prima zadani zahtjev te prosljeđuje pohranu zahtjeva prema bazi podataka. Baza podataka potvrđuje uspješnu pohranu zahtjeva dok poslužitelj korisniku ispisuje poruku uspjehnosti.



Slika 3.9: Sekvencijski dijagram za UC3

Obrazac uporabe UC13 - Odabir zahtjeva za pomoć

Korisnik šalje poslužitelju zahtjev za dohvat liste aktivnih zahtjeva. Poslužitelj dohvaća listu aktivnih zahtjeva iz baze podataka. Korisnik pregledava listu aktivnih zahtjeva uz mogućnost pregleda profila korisnika koji je zadao zahtjev. Korisnik odabire zahtjev s liste i prihvaća ga. Poslužitelj prima prihvatanje zahtjeva te prosljeđuje promjenu u bazu podataka. Poslužitelj obavještava korisnika o uspješnoj pohrani prihvatanja zahtjeva.



Slika 3.10: Sekvencijski dijagram za UC13

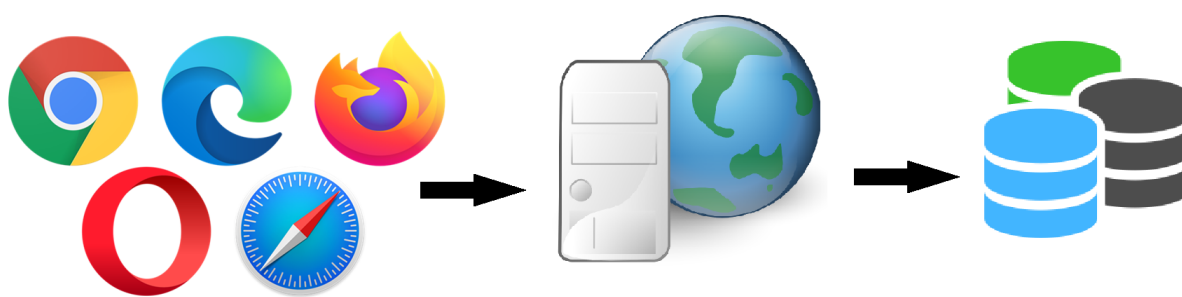
3.2 Ostali zahtjevi

- Aplikaciju je moguće koristiti od strane više korisnika istovremeno
- Funkcionalnost sustava je neovisna o eventualnim greškama korisnika
- Baza podataka mora biti dobro konfigurirana i sigurna
- Korisnici pristupaju određenom geografskom području
- Implementacija je bazirana na objektno-orijentiranom jeziku
- Sustav mora omogućiti jednostavnu nadogradnju novih funkcionalnosti
- Aplikacija se koristi na standardnom hrvatskom jeziku

4. Arhitektura i dizajn sustava

Arhitekturu sustava možemo podijeliti na tri podsustava:

- Web preglednik
- Web poslužitelj
- Baza podataka



Slika 4.1: Arhitektura sustava

Korisnik pomoću web preglednika po vlastitom izboru dobiva mogućnost pristupa web aplikaciji na Internetu. Preglednik zapravo šalje zahtjev HTTP protokolom web poslužitelju na kojem se nalazi web aplikacija i on ju pokreće i omogućuje korisniku daljnji rad na aplikaciji.

Pred korisnikom se nalaze sve funkcionalnosti koje aplikacija pruža. Onovoljno odabire pojedinu funkcionalnost te tako šalje zahtjev aplikaciji koja pristupa bazi podataka i prikuplja zatražene podatke i osvježava mu pregled aplikacije s prikupljenim podacima.

Sama aplikacija je sastavljena po načelu objektno usmjerene arhitekture. Razlog odabira takve arhitekture je trenutna sveprisutnost u industriji te predstavlja standard razvoja složenih programskih zahtjeva. Budući da je razvoj aplikacije

zamišljen kao rad više ljudi u timu objektno usmjerena arhitektura nudi da se sustav logički razdijeli na više cjelina te time omogućuje paralelnost razvoja. Također nudi recikliranje koda, mogućnost dodavanja novih funkcionalnosti bez poteškoća i greške se lako detektiraju i ispravljaju.

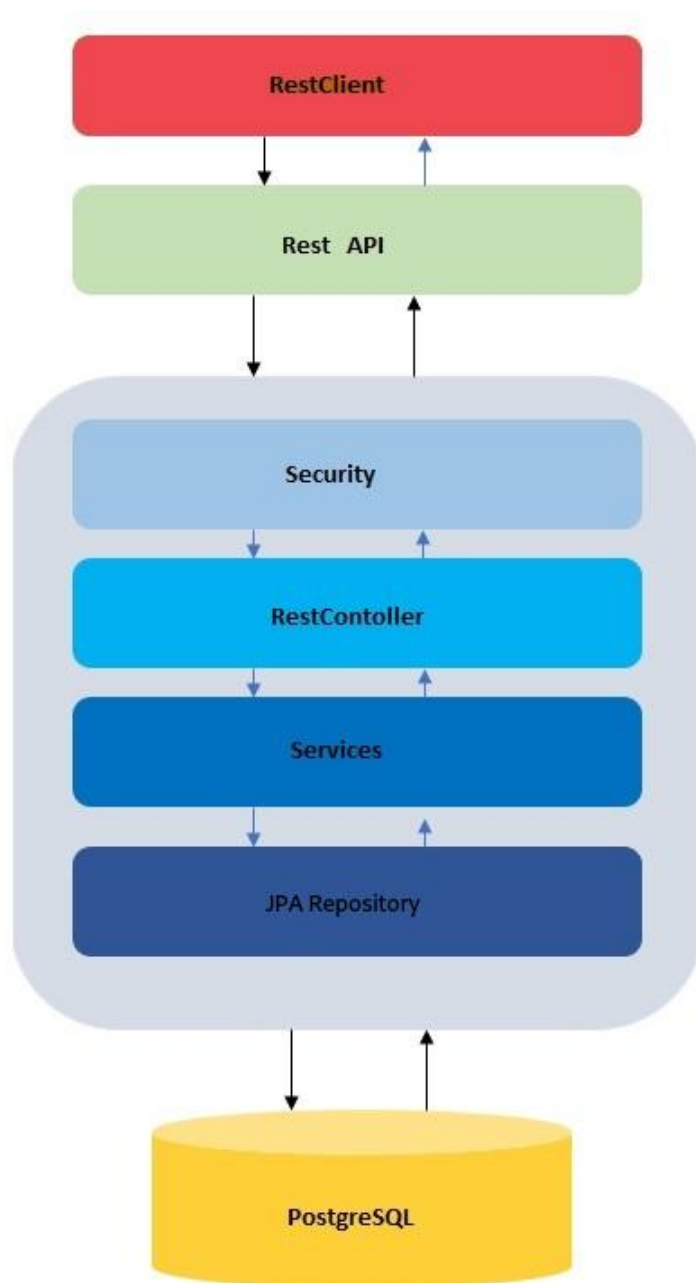
Aplikacija je razvijena u programskom jeziku Java 11 te kao razvojni okvir koristi Java Spring Boot verzije 2.3.4. Za sigurnost i sprječavanje vanjskih napada koristi se Spring Security. Kako bi se olakšala komunikacija s bazom koristi se Spring JPA. Za razvojno okruženje koristi se Eclipse, IntelliJ i VSCode.

Frontend je pisan u React-u. React je open-source, frontend, JavaScript knjižnica koja služi za izgradnju korisničkog sučelja. Koristi HTML, CSS, JSX i JavaScript kako bi modelirao sučelje.

Komunikacija između Jave na backendu i Reacta na frontendu je omogućena korištenjem REST api-ja.



Slika 4.2: Arhitektura sustava



Slika 4.3: Arhitektura sustava

4.1 Baza podataka

Implementacija baze podataka ostvarena je uporabom PostgreSQL zbog jednostavnosti korištenja i pozitivnog iskustva članova tima.

Baza podataka sastoji se od sljedećih tablica:

- Korisnik
- Zahtjev
- Lokacija
- Kandidatura
- Kandidiranje
- Ocjenjivanje
- ImaUlogu
- Uloga
- Obavijest

4.1.1 Opis tablica

Korisnik Ovaj entitet sadržava sve važne informacije o korisniku aplikacije. Sadrži attribute: ID_Korisnik, ime, prezime, email, lozinka, aktivan, token, slika te ID_Lokacija. Ovaj entitet je u vezi *One-to-Many* s entitetom Zahtjev preko atributa ID_Korisnik gdje korisnik ima dvije uloge, u vezi *Many-to-One* s entitetom Lokacija preko atributa ID_Lokacija, u vezi *Many-to-Many* s entitetom Kandidatura preko vezne tablice Kandidiranje i atributa ID_Korisnik, u vezi *One-to-Many* s entitetom Ocjenjivanje preko atributa ID_Korisnik gdje korisnik ima dvije uloge, u vezi *Many-to-Many* s entitetom Uloga preko vezne tablice ImaUlogu i atributa ID_Korisnik te u vezi *One-to-Many* s entitetom Obavijest preko atributa ID_Korisnik.

Korisnik		
ID_Korisnik	BIGSERIAL	jedinstveni identifikator korisnika
ime	VARCHAR	ime korisnika
prezime	VARCHAR	prezime korisnika
email	VARCHAR	e-mail adresa korisnika
lozinka	VARCHAR	hash lozinke
aktivan	BOOLEAN	status korisnika
token	VARCHAR	JWT refresh token koji se koristi pri autentifikaciji korisnika

Korisnik		
slika	VARCHAR	putanja do fotografije korisnika
<u>ID_Lokacija</u>	BIGINT	identifikator lokacije

Zahtjev Ovaj entitet sadržava sve važne informacije o korisnikovu zahtjevu. Sadrži attribute: ID_Zahtjev, opis, tstmp, status, brojMobitela, ID_Autor, ID_Izvršitelj, potvrđen, ID_Lokacija, execTstmp. Ovaj entitet je u vezi *Many-to-one* s entitetom Lokacija preko atributa ID_Lokacija, također u vezi *One-to-Many* s entitetom Ocjenjivanje preko atributa ID_Zahtjev te u dvije veze *Many-to-One* s entitetom Korisnik preko atributa ID_Korisnik što rezultira pojavom atributa ID_Autor i ID_Izvršitelj u tablici Zahtjev.

Zahtjev		
ID_Zahtjev	BIGSERIAL	jedinstveni identifikator zahtjeva
opis	VARCHAR	opis zahtjeva
tstmp	TIMESTAMP	vremenska točka do kada se treba izvršiti zahtjev
status	VARCHAR	status zahtjeva
brojMobitela	VARCHAR	broj mobitela autora zahtjeva
<u>ID_Autora</u>	BIGINT	jedinstveni identifikator korisnika(autora zahtjeva)
<u>ID_Izvršitelj</u>	BIGINT	jedinstveni identifikator korisnika(izvršitelja zahtjeva)
potvrđen	BOOLEAN	zastavica status potvrđen
<u>ID_Lokacija</u>	BIGINT	identifikator lokacije
execTstmp	TIMESTAMP	točka u vremenu kada je zahtjev izvršen

Lokacija Ovaj entitet sadržava sve važne informacije o lokaciji. Sadrži attribute: ID_Lokacija, duljina i sirina, drzava, naselje te adresa. Ovaj entitet je u vezi *One-to-Many* s entitetom Zahtjev preko atributa ID_Lokacija, također je u vezi *One-to-Many* s entitetom Korisnikom preko atributa ID_Lokacija te u vezi *One-to-Many* s entitetom Kandidatura preko atributa ID_Lokacija.

Lokacija		
ID_Lokacija	BIGSERIAL	jedinstveni identifikator lokacije
duljina	NUMERIC	geografska dužina
sirina	NUMERIC	geografska sirina
drzava	VARCHAR	naziv države
naselje	VARCHAR	naziv naselja
adresa	VARCHAR	adresa na kojoj treba odraditi zahtjev

Kandidatura Ovaj entitet sadržava sve važne informacije o kandidaturi za korisnika godine. Sadrži attribute: ID_Kandidatura, godina i ID_Lokacija. Ovaj entitet je u vezi *Many-to-Many* s entitetom Korisnik preko vezne tablice Kandidiranja i atributa ID_Kandidatura te u vezi *Many-to-One* s Lokacijom preko atributa ID_Lokacija.

Kandidatura		
<u>ID_Lokacija</u>	BIGINT	identifikator lokacije
godina	INT	godina kandidature
ID_Kandidatura	BIGSERIAL	jedinstveni identifikator lokacije

Kandidiranje Ovaj entitet je vezna tablica koja povezuje entitete Korisnik i Kandidatura. Sadrži dva strana ključa: ID_Korisnik i ID_Kandidatura.

Kandidiranje		
ID_Korisnik	BIGINT	identifikator korisnika
ID_Kandidatura	BIGINT	identifikator kandidature

Ocjenjivanje Ovaj entitet sadržava sve važne informacije o međusobnom ocjenjivanju korisnika. Sadrži attribute: ID_Ocjenjivanje, ID_Zahtjev, ocjena, komentar, ID_Ocjenjivac i ID_Ocjenjeni. Ovaj entitet je u dvostrukoj vezi *Many-to-One* s entitetom Korisnik preko atributa ID_Ocjenjeni i ID_Ocjenjivac te u vezi *Many-to-One* s entitetom Zahtjev preko atributa ID_Zahtjev.

Ocjenjivanje		
<u>ID_Ocjenjivanje</u>	BIGSERIAL	jedinstveni identifikator ocjenjivanja
<u>ID_Zahtjev</u>	BIGINT	identifikator zahtjeva
ocjena	INT	ocjena usluge
komentar	VARCHAR	komentar ocjenitelja
<u>ID_Ocjenjivac</u>	BIGINT	identifikator ocjenjivaca
<u>ID_Ocjenjeni</u>	BIGINT	identifikator ocjenjenog

Uloga Entitet sadržava informaciju o ulozi korisnika tijekom korištenja aplikacije. Uloga može biti administrator ili obični korisnik. Sadrži attribute: ID_Uloga i naziv. U vezi je *Many-to-Many* s entitetom Korisnik preko vezne tablice ImaUlogu i atributa ID_Uloga.

Uloga		
<u>ID_Uloga</u>	BIGSERIAL	jedinstveni identifikator uloge
naziv	VARCHAR	naziv uloge

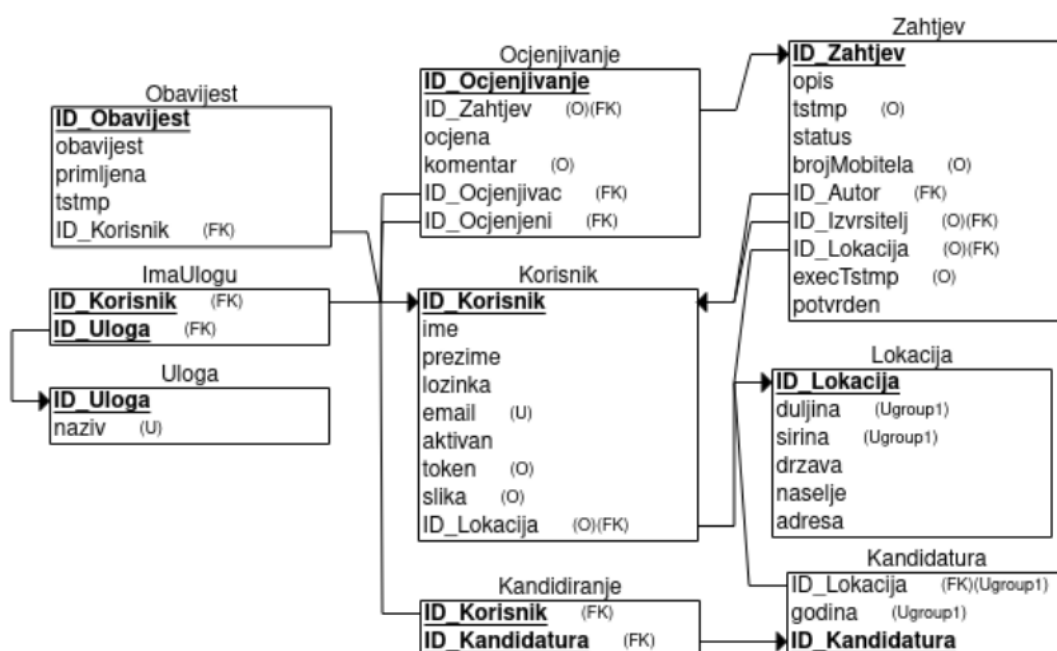
ImaUlogu je vezna tablica koja sadržava informacije koji korisnik ima koju ulogu unutar aplikacije. Sadrži attribute - strane ključeve: ID_Korisnik i ID_Uloga.

ImaUlogu		
<u>ID_Korisnik</u>	BIGINT	identifikator korisnika
<u>ID_Uloga</u>	BIGINT	identifikator uloge

Obavijest je tablica koja sadržava obavijesti korisnika. Sadrži attribute: ID_Obavijest, obavijest, primljena, tstmp i ID_Korisnik. Ovaj entitet je u vezi *Many-to-One* s entitetom Korisnik preko atributa ID_Korisnik.

Obavijest		
<u>ID_Obavijest</u>	BIGSERIAL	jedinstveni identifikator
obavijest	VARCHAR	text obavijesti
primljena	BOOLEAN	zastavica primljenosti poruke
tstmp	TIMESTAMP	timestamp slanja obavijesti
<u>ID_Korisnik</u>	BIGINT	identifikator korisnika primatelja

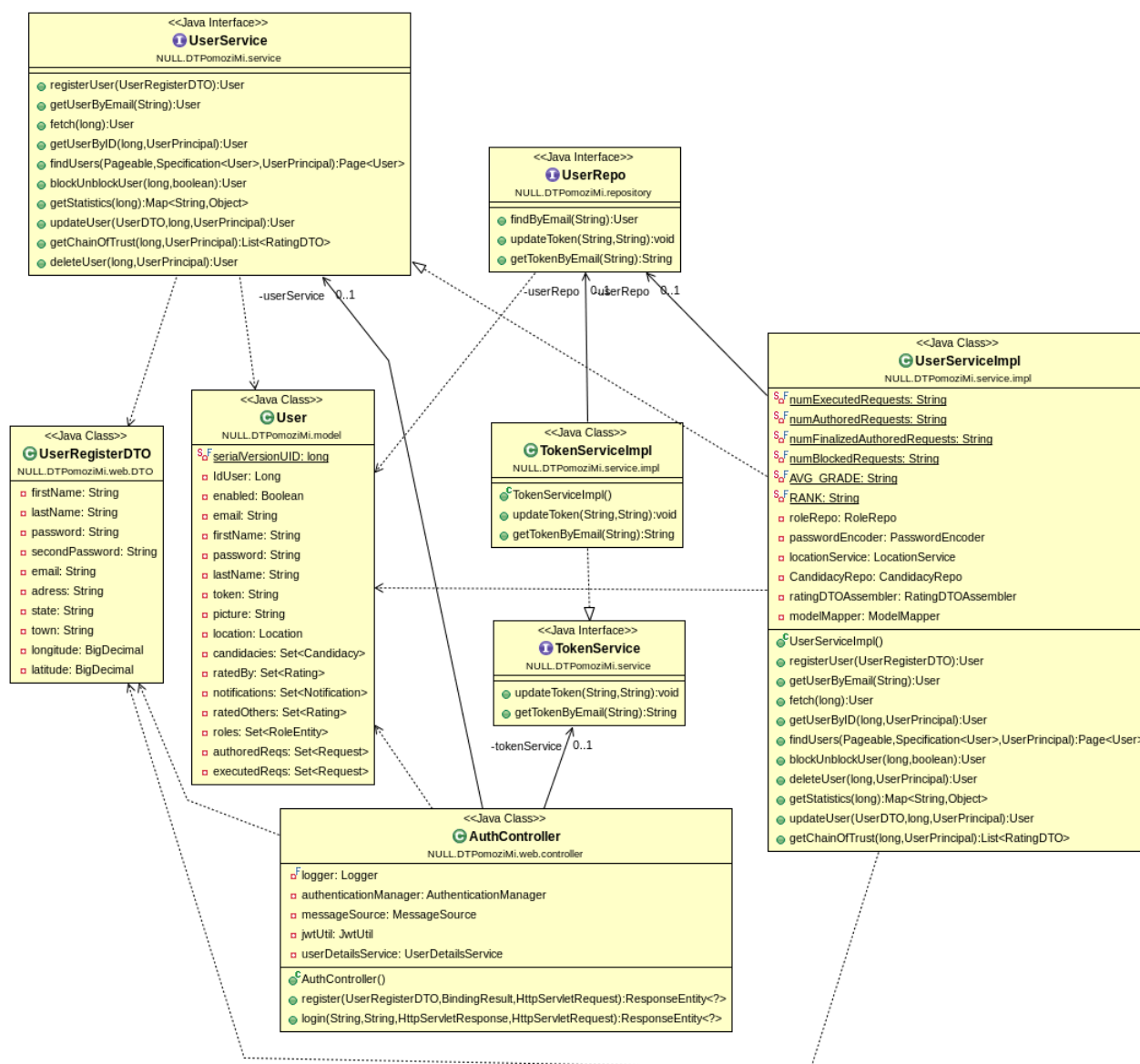
4.1.2 Dijagram baze podataka



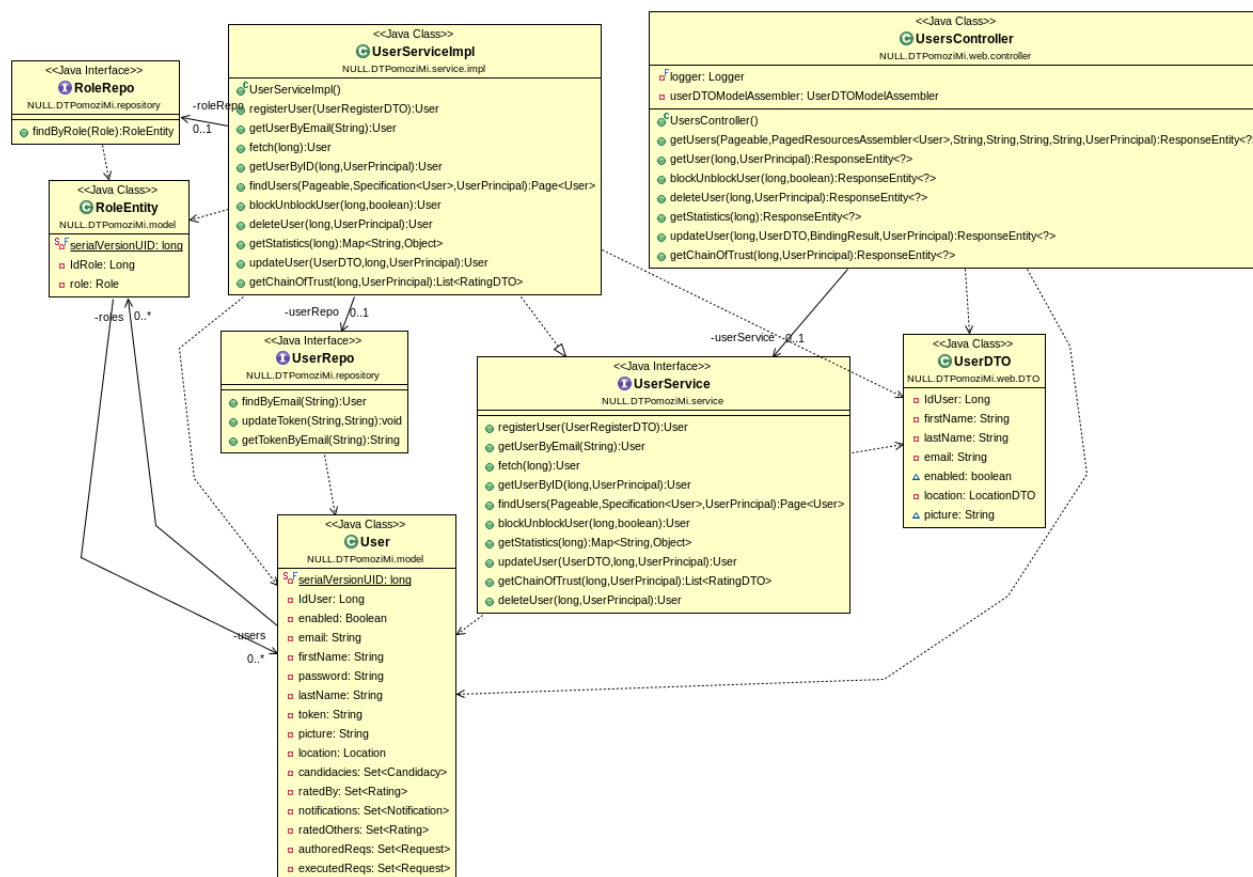
Slika 4.4: Dijagram baze podataka

4.2 Dijagram razreda

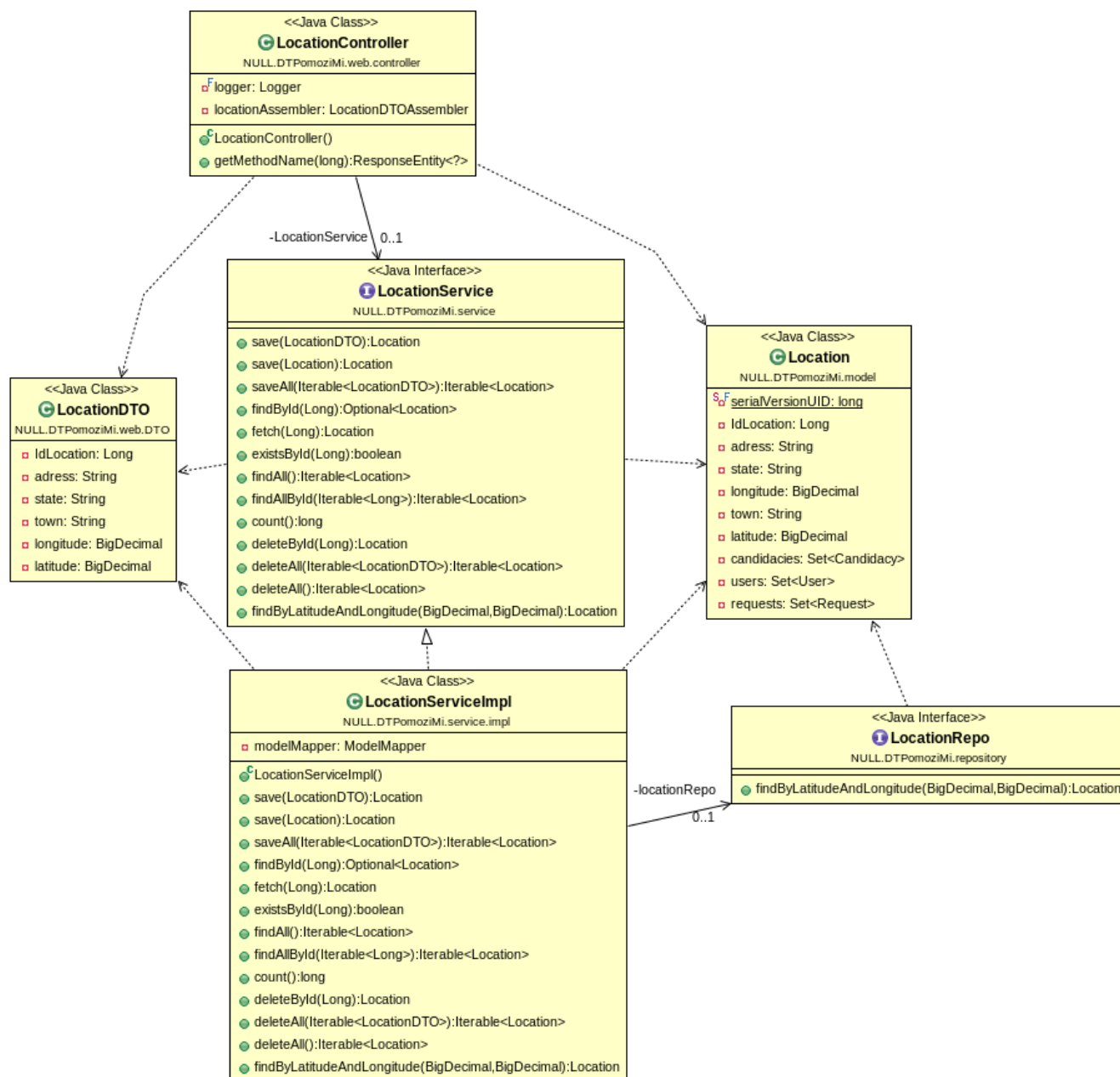
Na slikama 4.5, 4.6, 4.7, 4.8, 4.9, 4.10 i 4.11 su prikazani dijagrami razreda za backend. Raspoređeni su u više slika radi bolje preglednosti i slične funkcionalnosti. Iz samih slika lakše se može uočiti povezanost samih stavki, njihova ovisnost te sadržaj atributa i funkcionalnosti.



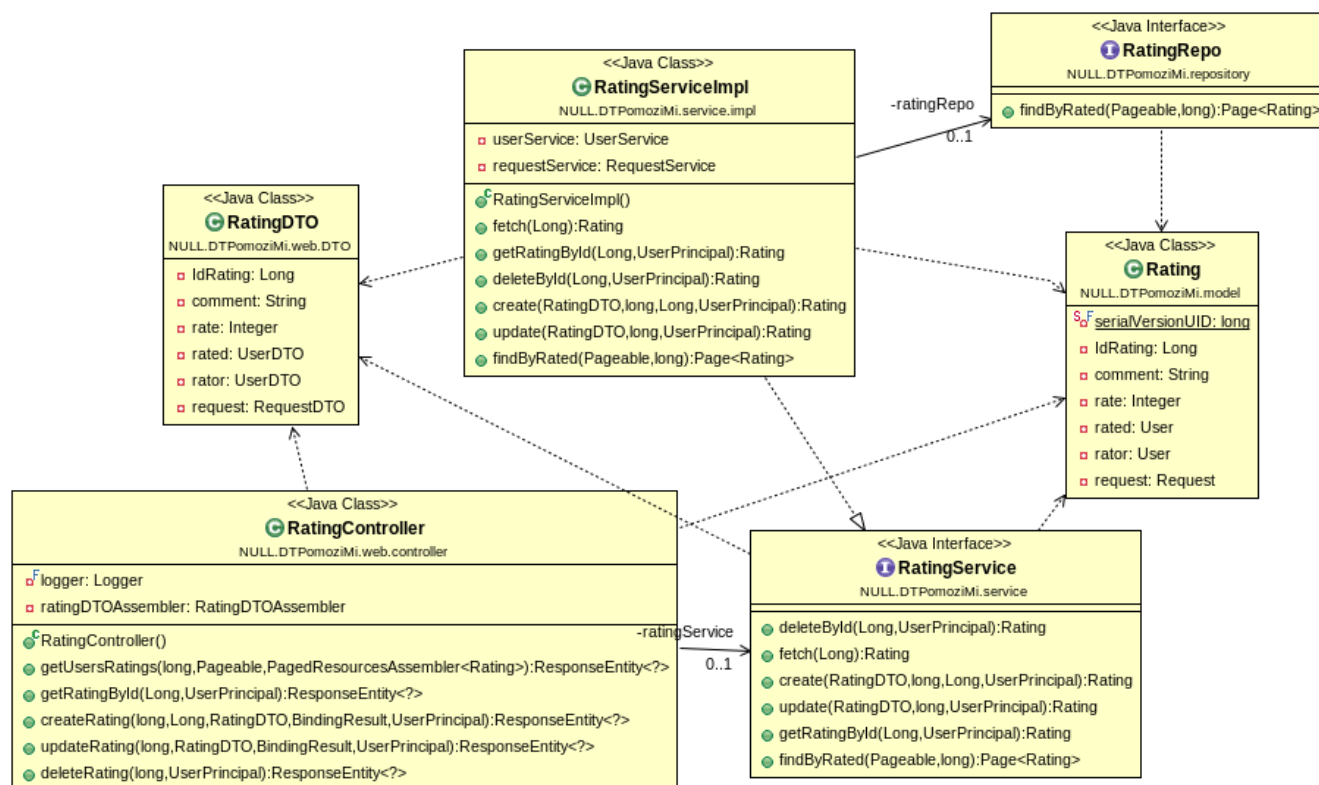
Slika 4.5: Dijagram razreda za autentifikaciju



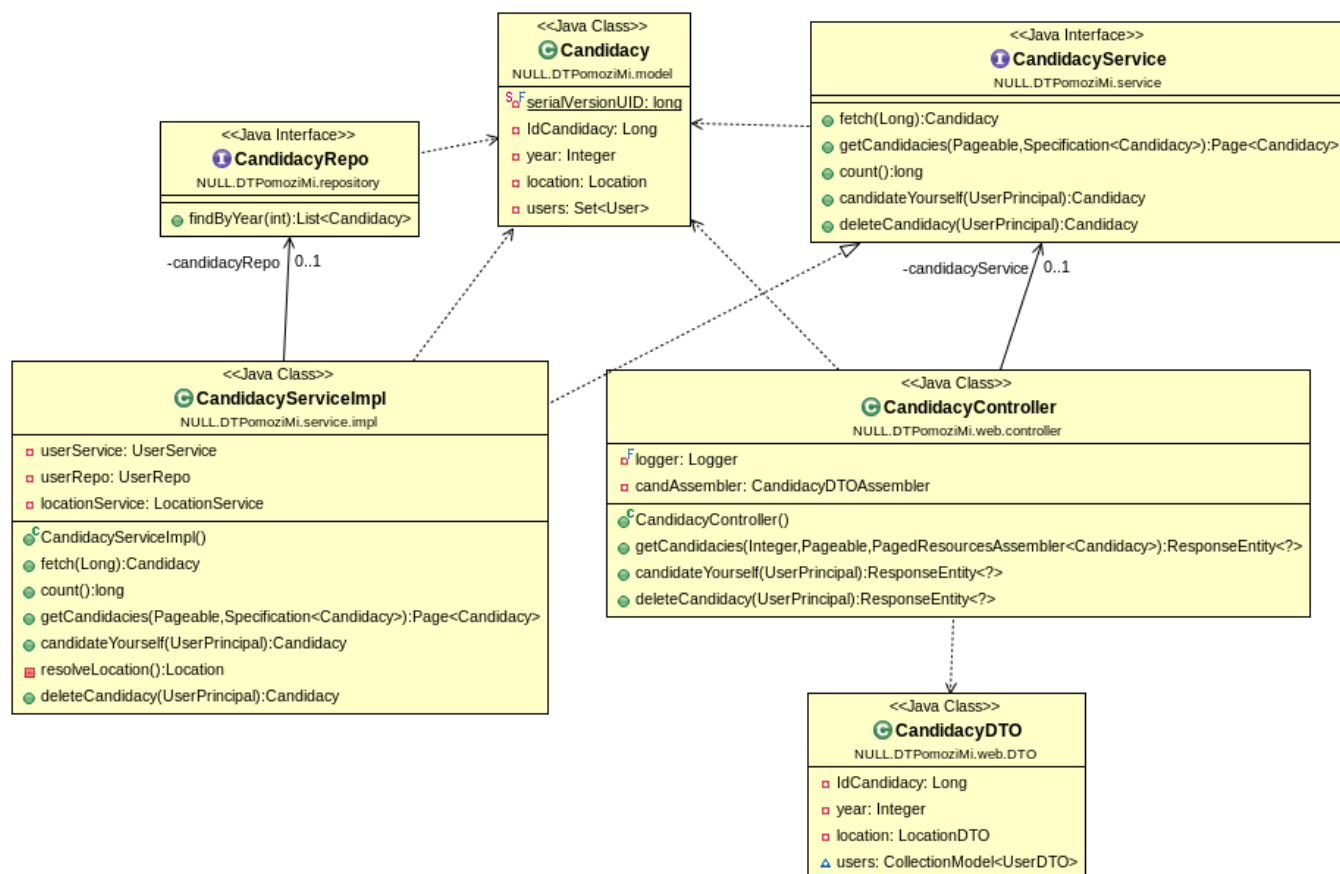
Slika 4.6: Dijagram razreda za korisnika



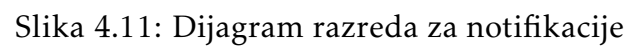
Slika 4.8: Dijagram razreda za lokaciju



Slika 4.9: Dijagram razreda za ocenjivanje

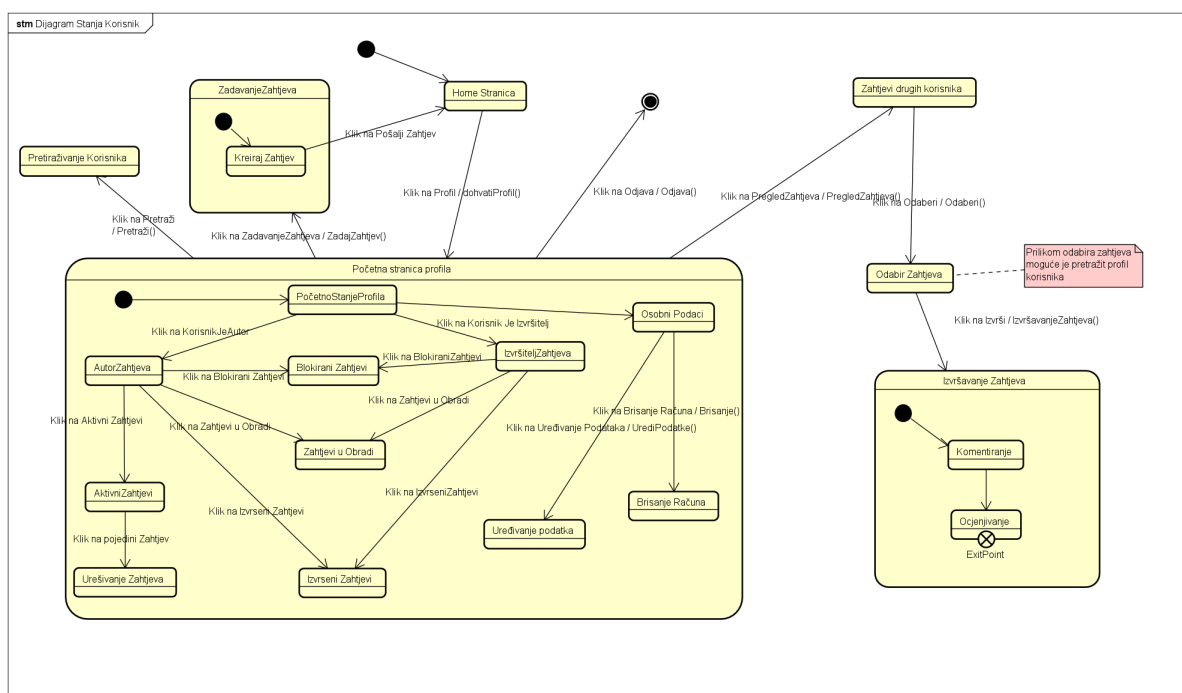


Slika 4.10: Dijagram razreda za kandidiranje



4.3 Dijagram stanja

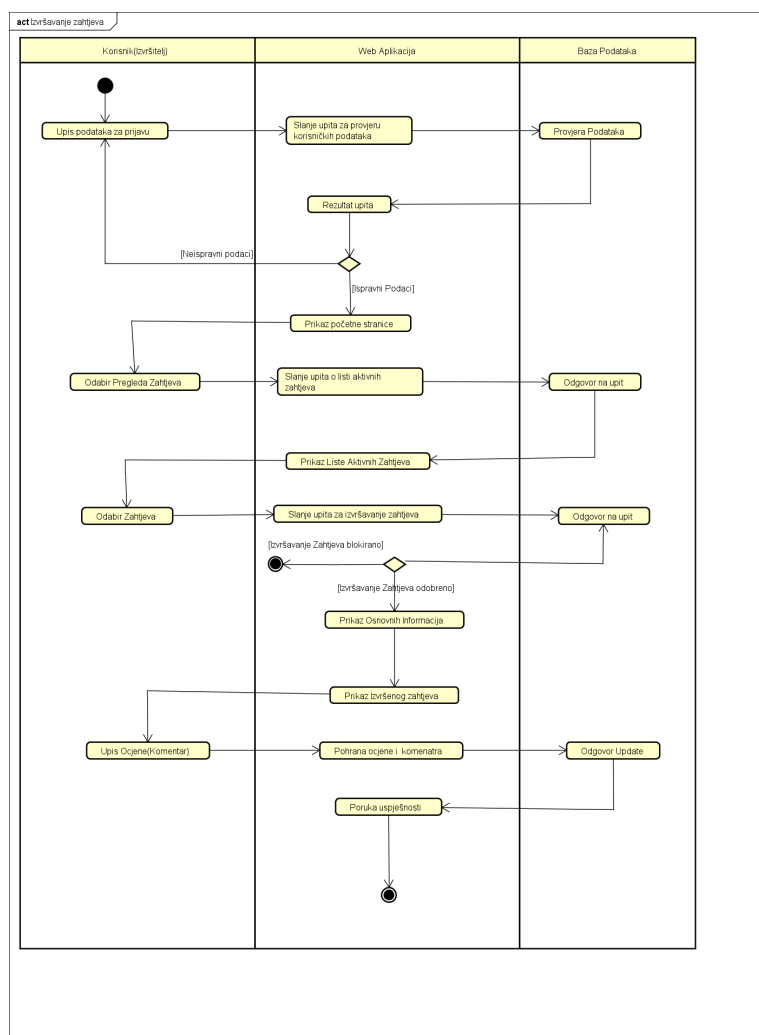
Dijagram stanja prikazuje stanja objekta te prijelaze iz jednog stanja u drugo temeljene na događajima. Na slici 4.12 prikazan je dijagram stanja registriranog korisnika. Nakon prijave, korisniku se prikazuje početna stranica (Home page). Korisnik klikom na Profil otvara svoju profilnu stranicu. Korisnik može odabirom Autor je Zahtjeva pregledati vlastite aktivne zahtjeve, izvršene, zahtjeve u obradi i blokirane. Isto tako odabirom Izvršitelj je zahtjeva može pregledati zahtjeve koje je izvršio. Korisniku se klikom na Pregled Zahtjeva nudi pregled aktivnih zahtjeva te mogućnost odabira nekoga od zahtjeva s liste. Nakon izvršavanja zahtjeva korisnik mora ocijeniti drugog korisnika i opcionalno ostaviti komentar te se komentari i ocjene vide u profilu korisnika. Korisniku se nudi i mogućnost uređivanja osobnih podataka kao i brisanje korisničkog računa, dok klikom na Search korisnik može pretraživati druge korisnike aplikacije. Također klikom na Zadavanje Zahtjeva korisnik može zadati novi zahtjev.



Slika 4.12: Dijagram stanja

4.4 Dijagram aktivnosti

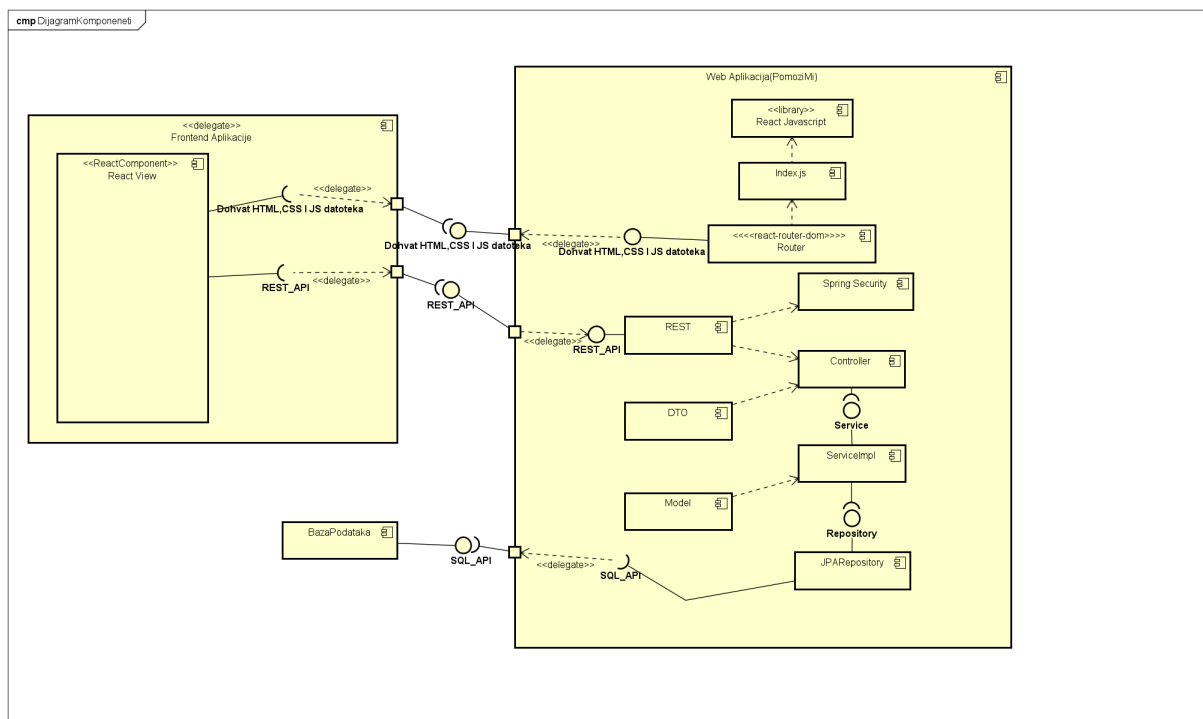
Dijagram aktivnosti primjenjuje se za opis modela toka upravljanja ili toka podataka. Ne upotrebljava se za modeliranje događajima poticanog ponašanja. U modeliranju toka upravljanja svaki novi korak poduzima se nakon završenog prethodnog, a naglasak je na jednostavnosti. Na dijagramu 4.13 prikazan je proces izvršavanja zahtjeva za pomoć. Korisnik se prijavljuje u sustav, prikazujemo se početna stranica, nakon koje on odabire Pregled zahtjeva. Kada je zamijetio zahtjev kojeg može izvršiti, odabire taj zahtjev, ako mu je zahtjev za izvršavanjem odobren poslužuju se dodatne informacije o zahtjevu kako bi mogao izvršiti zahtjev. Nakon izvršenog zahtjeva, korisnik treba ocijeniti i opcionalno komentirati.



Slika 4.13: Dijagram Aktivnosti

4.5 Dijagram komponenti

Dijagram komponenti prikazan na slici 4.14 opisuje organizaciju i međuovisnost komponenti, interne strukture i odnose prema okolini. Sustavu se pristupa preko dva različita sučelja. Preko sučelja za dohvat HTML, CSS i JS datoteka poslužuju se datoteke koje pripadaju *frontend* dijelu aplikacije. Router je komponenta koja na upit s url određuje koja datoteka će se poslužiti na sučelje. Sve JavaScript datoteke ovise o React biblioteci iz koje dohvaćaju gotove komponente kao što su gumbi, forme i slično. REST API poslužuje podatke koji pripadaju *backend* dijelu aplikacije. Pomoću JPA-a dohvaćaju se tablice iz baze podataka. Podaci u bazi obliku se pomoću komponente Model te se prosljeđuju dalje po REST arhitekturi. DTO(Data Transfer Object) omata podatke te se prosljeđuje prema *frontend* strani koja ih prikladno prikazuje. Za autorizaciju REST servis koristi Spring Security.



Slika 4.14: Dijagram Komponeneti

5. Implementacija i korisničko sučelje

5.1 Korištene tehnologije i alati

Komunikacija u timu realizirana je korištenjem aplikacije [Discord](https://discord.com/)¹, [Microsoft Teams](https://www.microsoft.com/en-us/microsoft-365/microsoft-teams)² i [WhatsApp](https://www.whatsapp.com/)³. Za izradu UML dijagrama korišten je alat [Astah Professional](https://astah.net/)⁴, a kao sustav za upravljanje izvornim kodom [Git](https://git-scm.com/)⁵. Udaljeni repozitorij projekta je dostupan na web platformi [GitLab](https://about.gitlab.com/)⁶. Kao razvojno okruženje korišteni su [IntelliJ](https://www.jetbrains.com/idea/)⁷ - integrirano razvojno okruženje (IDE) tvrtke JetBrains. [Spring Tool Suite](https://spring.io/tools)⁸ (Eclipse) - integrirano razvojno okruženje namijenjeno za razvoj aplikacija koje koriste Spring kao radni okvir te [VSCode](https://code.visualstudio.com/)⁹ - integrirano razvojno okruženje pogodno za razvoj frontend-a.

Aplikacija je napisana koristeći radni okvir [Spring Boot](https://spring.io/projects/spring-boot)¹⁰ i jezik [Javu](https://www.java.com/)¹¹ za izradu *backenda* te [React](https://reactjs.org/)¹² i jezik [JavaScript](https://www.javascript.com/)¹³ za izradu *frontenda*. Spring Boot je open-source Javin radni okvir koji omogućuje izgradnju mikro servisa. Sam Spring Boot dolazi s već predkonfiguriranim značajkama koje programerima omogućuju konvencijonalnost te mogućnost pokretanja aplikacije bez dodatnog posla. React, također poznat kao React.js ili ReactJS, je biblioteka u JavaScriptu za izgradnju korisničkih sučelja. Složene aplikacije u React-u obično zahtijevaju korištenje dodatnih biblioteka za interakciju s API-jem.

Baza podataka i aplikacija se nalaze na poslužitelju [Heroku](https://www.heroku.com/)¹⁴.

¹<https://discord.com/>

²<https://www.microsoft.com/en-us/microsoft-365/microsoft-teams>

³<https://www.whatsapp.com/>

⁴<https://astah.net/>

⁵<https://git-scm.com/>

⁶<https://about.gitlab.com/>

⁷<https://www.jetbrains.com/idea/>

⁸<https://spring.io/tools>

⁹<https://code.visualstudio.com/>

¹⁰<https://spring.io/projects/spring-boot>

¹¹<https://www.java.com/>

¹²<https://reactjs.org/>

¹³<https://www.javascript.com/>

¹⁴<https://www.heroku.com/>

5.2 Ispitivanje programskog rješenja

5.2.1 Ispitivanje komponenti

Provesli smo ispitivanje-unit testing nad razredima koji implementiraju temeljne funkcionalnosti, a to su *User* i *Request*. Ispitali smo njihovih 7 metoda te njihove redovne slučajeve, rubne slučajeve i one koji izazivaju pogrešku. Pri testiranju su korišteni pripremljeni podatci u bazi koji se unose automatski pri svakom pokretanju testa.

Ispitni slučaj 1: Testiranje funkcionalnosti metode `updateUser()`

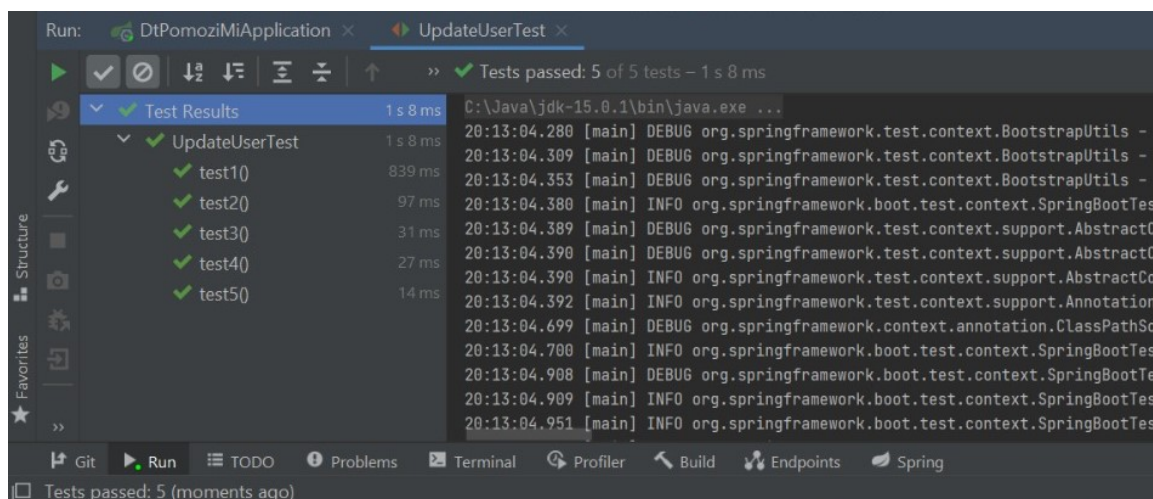
Ulaz:

1. vrijednosti `UserDTO`
2. Id korisnika čiji se profil uređuje
3. `User principal`

Očekivani rezultat:

1. uređeni user
2. pripadajuća pogreška

Rezultat: Očekivani rezultat je zadovoljen, aplikacija je prošla sve testove



Slika 5.1: `UpdateUserTest()`

Kod unit testa:

```
package NULL.DTPomoziMi.service.impl;

import static org.junit.jupiter.api.Assertions.assertEquals;
import static org.junit.jupiter.api.Assertions.assertThrows;

import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.dao.DataIntegrityViolationException;
import org.springframework.security.test.context.support.WithUserDetails;
import org.springframework.test.context.ActiveProfiles;

import NULL.DTPomoziMi.exception.IllegalAccessException;
import NULL.DTPomoziMi.model.User;
import NULL.DTPomoziMi.security.UserPrincipal;
import NULL.DTPomoziMi.service.UserService;
import NULL.DTPomoziMi.util.UserPrincipalGetter;
import NULL.DTPomoziMi.web.DTO.UserDTO;
import NULL.DTPomoziMi.web.assemblers.UserDTOModelAssembler;

@SpringBootTest
@ActiveProfiles("dev")
public class UpdateUserTest {

    @Autowired
    private UserService service;

    @Autowired
    private UserDTOModelAssembler assembler;

    private UserPrincipal principal;

    private UserDTO userDTO;

    public void setup() { userDTO = assembler.toModel(service.fetch(11)); }
```

```
//korisnik ima pristup uredivanju profila
@Test
@WithUserDetails(value = "marija.orec@gmail.com",
userDetailsServiceBeanName = "myUserDetailsService")
public void test1() {
    setup();
    principal = UserPrincipalGetter.getPrincipal();

    userDTO.setEmail("bananko@gmail.com");
    userDTO.setFirstName("Janko");
    userDTO.setLastName("bananko");
    userDTO.setPicture("aaa");

    service.updateUser(userDTO, 11, principal);

    User user = service.fetch(11);

    assertEquals(user.getEmail(), userDTO.getEmail());
    assertEquals(user.getFirstName(), userDTO.getFirstName());
    assertEquals(user.getLastName(), userDTO.getLastName());
    assertEquals(user.getPicture(), userDTO.getPicture());
    assertEquals(user.getIdUser(), userDTO.getIdUser());
}

// korisnik je postavio već postojeći email
@Test
@WithUserDetails(value = "bananko@gmail.com",
userDetailsServiceBeanName = "myUserDetailsService")
public void test2() {
    setup();
    principal = UserPrincipalGetter.getPrincipal();

    userDTO.setEmail("matea.lipovac@gmail.com");
    assertThrows(DataIntegrityViolationException.class,
```



```
() -> service.updateUser(userDT0, 11, principal));
}

//korisnik nema pristup uredivanju profila
@Test
@WithUserDetails(value = "matea.lipovac@gmail.com",
userDetailsServiceBeanName = "myUserDetailsService")
public void test3() {
    setup();
    principal = UserPrincipalGetter.getPrincipal();
    assertThrows(IllegalAccessException.class,
    () -> service.updateUser(userDT0, 11, principal));
}

//userDT0 i id se ne poklapaju
@Test
@WithUserDetails(value = "bananko@gmail.com",
userDetailsServiceBeanName = "myUserDetailsService")
public void test4() {
    setup();
    principal = UserPrincipalGetter.getPrincipal();
    assertThrows(IllegalArgumentException.class,
    () -> service.updateUser(userDT0, 3, principal));
}

//neprijavljen korisnik
@Test
public void test5() {
    principal = UserPrincipalGetter.getPrincipal();
    assertThrows(AuthenticationCredentialsNotFoundException.class,
    () -> service.updateUser(userDT0, 11, principal));
}
}
```

Ispitni sličaj 2: Testiranje funkcionalnosti metode updateRequest()

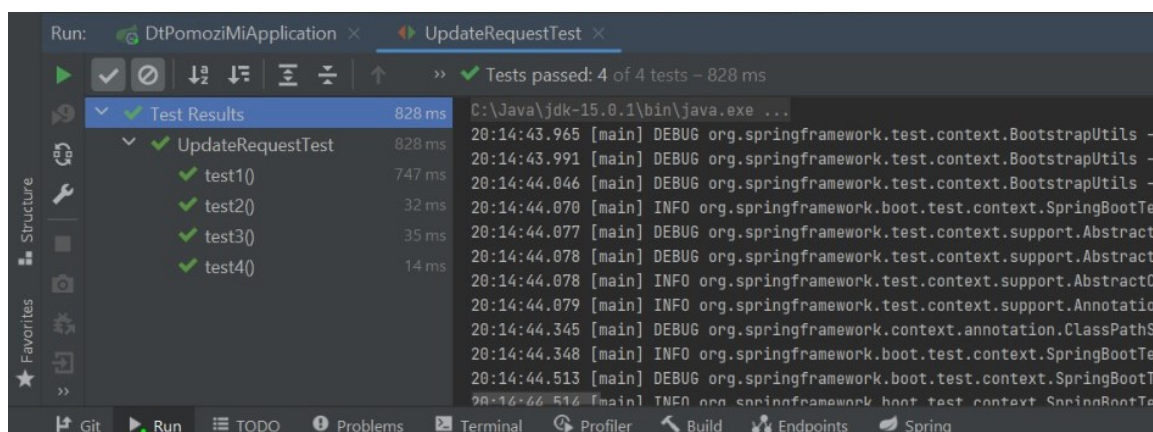
Ulaz:

1. id zahtjeva koji se uređuje
2. vrijednosti RequestDTO
3. User principal

Očekivani rezultat:

1. uređeni zahtjev
2. pripadajuća pogreška

Rezultat: Očekivani rezultat je zadovoljen, aplikacija je prošla sve testove



Slika 5.2: UpdateRequestTest()

Kod unit testa:

```
package NULL.DTPomoziMi.service.impl;

import static org.junit.jupiter.api.Assertions.assertEquals;
import static org.junit.jupiter.api.Assertions.assertNotEquals;
import static org.junit.jupiter.api.Assertions.assertNotNull;
import static org.junit.jupiter.api.Assertions.assertNull;
import static org.junit.jupiter.api.Assertions.assertThrows;

import java.time.LocalDateTime;

import org.junit.jupiter.api.Test;
```

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.security.authentication.
AuthenticationCredentialsNotFoundException;
import org.springframework.security.test.context.support.WithUserDetails;
import org.springframework.test.context.ActiveProfiles;

import NULL.DTPomoziMi.exception.IllegalAccessException;
import NULL.DTPomoziMi.model.Request;
import NULL.DTPomoziMi.model.RequestStatus;
import NULL.DTPomoziMi.security.UserPrincipal;
import NULL.DTPomoziMi.util.UserPrincipalGetter;
import NULL.DTPomoziMi.web.DTO.RequestDTO;
import NULL.DTPomoziMi.web.assemblers.RequestDTOAssembler;

@SpringBootTest
@ActiveProfiles("dev")
public class UpdateRequestTest {
    @Autowired
    private RequestServiceImpl service;

    @Autowired
    private RequestDTOAssembler assembler;

    private RequestDTO request;

    public void setup() { request = assembler.toModel(service.fetch(27)); }

    //uspjsan update
    @Test
    @WithUserDetails(value = "dominik.curic@gmail.com",
        userDetailsServiceBeanName = "myUserDetailsService")
    public void test1() {
        setup();
        UserPrincipal principal = UserPrincipalGetter.getPrincipal();
```

```
final String phone = "091";
final LocalDateTime ldt = LocalDateTime.of(2022, 1, 1, 1, 1, 1);
final String desc = "abc";

request.setDescription(desc);
request.setPhone(phone);
request.setTstamp(ldt);
request.setLocation(null);
request.setAuthor(null);
request.setStatus(RequestStatus.FINALIZED);

service.updateRequest(27, request, principal);
Request updated = service.fetch(27);

assertEquals(phone, updated.getPhone());
assertEquals(ldt, updated.getTstamp());
assertEquals(desc, updated.getDescription());
assertNull(updated.getLocation());
assertNotEquals(123L, updated.getIdRequest());
assertNotNull(updated.getAuthor());
assertNotEquals(RequestStatus.FINALIZED, updated.getIdRequest());

}

//id requesta i id requestDT0a razliciti
@Test
@WithUserDetails(value = "dominik.curic@gmail.com",
userDetailsServiceBeanName = "myUserDetailsService")
public void test2() {
    setup();
    UserPrincipal principal = UserPrincipalGetter.getPrincipal();
    assertThrows(IllegalArgumentException.class,
    () -> service.updateRequest(3, request, principal));
}
```

```
//nije autor
@Test
@WithUserDetails(value = "jan.rocek@gmail.com",
userDetailsServiceBeanName = "myUserDetailsService")
public void test3() {
    setup();
    UserPrincipal principal = UserPrincipalGetter.getPrincipal();
    assertThrows(IllegalAccessException.class,
    () -> service.updateRequest(27, request, principal));
}

//neprijavljen korisnik
@Test
public void test4() {
    UserPrincipal principal = UserPrincipalGetter.getPrincipal();
    assertThrows(AuthenticationCredentialsNotFoundException.class,
    () -> service.updateRequest(19, request, principal));
}
}
```

Ispitni sličaj 3: Testiranje funkcionalnosti metode pickForExecution()

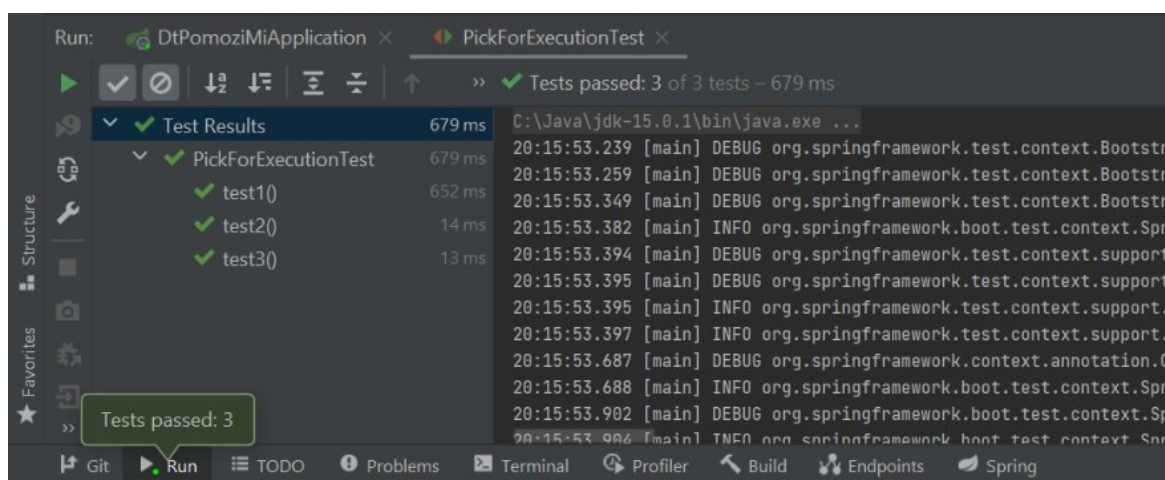
Ulaz:

1. id zahtjeva koji se odabire
2. User principal

Očekivani rezultat:

1. zahtjev je u statusu izvršavanja (executing)
2. pripadajuća pogreška

Rezultat: Očekivani rezultat je zadovoljen, aplikacija je prošla sve testove



Slika 5.3: PickForExecutionTest()

Kod unit testa:

```
package NULL.DTPomoziMi.service.impl;
```

```
import static org.junit.jupiter.api.Assertions.assertEquals;
```

```
import static org.junit.jupiter.api.Assertions.assertThrows;
```

```
import org.junit.jupiter.api.Test;
```

```
import org.springframework.beans.factory.annotation.Autowired;
```

```
import org.springframework.boot.test.context.SpringBootTest;
```

```
import org.springframework.security.authentication.
```

```
AuthenticationCredentialsNotFoundException;
```

```
import org.springframework.security.test.context.support.WithUserDetails;
```

```
import org.springframework.test.context.ActiveProfiles;
```

```
import NULL.DTPomoziMi.exception.IllegalActionException;
```

```
import NULL.DTPomoziMi.model.RequestStatus;
```

```
import NULL.DTPomoziMi.security.UserPrincipal;
```

```
import NULL.DTPomoziMi.service.RequestService;
```

```
import NULL.DTPomoziMi.util.UserPrincipalGetter;
```

```
@SpringBootTest
```

```
@ActiveProfiles("dev")
```

```
public class PickForExecutionTest {
```

```
@Autowired
private RequestService service;

//uspjesno odabrano izvršavanje
@Test
@WithUserDetails(value = "iva.boksic@gmail.com",
userDetailsServiceBeanName = "myUserDetailsService")
public void test1() {
    UserPrincipal principal = UserPrincipalGetter.getPrincipal();
    service.pickForExecution(14, principal).getStatus();
    assertEquals(RequestStatus.EXECUTING, service.fetch(14).getStatus());
}

//neaktivan zahtjev
@Test
@WithUserDetails(value = "iva.boksic@gmail.com",
userDetailsServiceBeanName = "myUserDetailsService")
public void test2() {
    UserPrincipal principal = UserPrincipalGetter.getPrincipal();
    assertThrows(IllegalActionException.class,
    () -> service.pickForExecution(24, principal));
}

//neprijavljen korisnik
@Test
public void test3() {
    UserPrincipal principal = UserPrincipalGetter.getPrincipal();
    assertThrows(AuthenticationCredentialsNotFoundException.class,
    () -> service.deleteRequest(14, principal).getStatus());
}
}
```

Ispitni sličaj 4: Testiranje funkcionalnosti metode `getUserById()`

Ulaz:

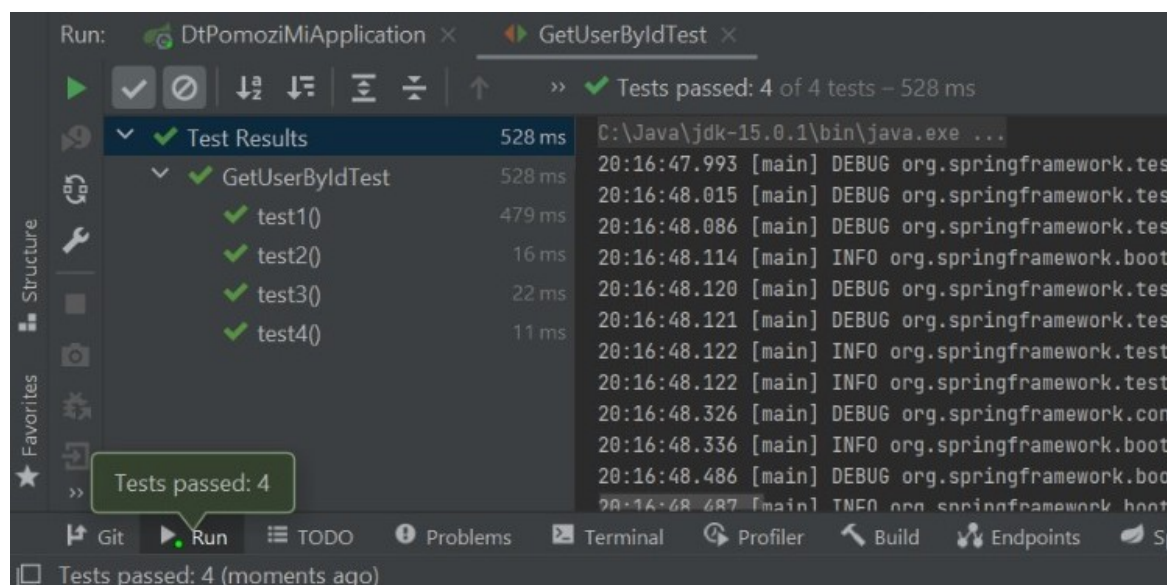
1. id korisnika koji se dohvaća

2. User principal

Očekivani rezultat:

1. korisnik s traženim idom
2. pripadajuća pogreška

Rezultat: Očekivani rezultat je zadovoljen, aplikacija je prošla sve testove



Slika 5.4: GetUserByIdTest()

Kod unit testa:

```
package NULL.DTPomoziMi.service.impl;
```

```
import static org.junit.jupiter.api.Assertions.assertEquals;
import static org.junit.jupiter.api.Assertions.assertNull;
import static org.junit.jupiter.api.Assertions.assertThrows;
```

```
import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.security.authentication.
AuthenticationCredentialsNotFoundException;
import org.springframework.security.test.context.support.WithUserDetails;
import org.springframework.test.context.ActiveProfiles;
```



```
import NULL.DTPomoziMi.exception.EntityMissingException;
import NULL.DTPomoziMi.security.UserPrincipal;
import NULL.DTPomoziMi.service.UserService;
import NULL.DTPomoziMi.util.UserPrincipalGetter;

@SpringBootTest
@ActiveProfiles("dev")
class GetUserByIdTest {

    @Autowired
    private UserService service;

    private UserPrincipal principal;

    //dobar id
    @Test
    @WithUserDetails(value = "jan.rocek@gmail.com",
        userDetailsServiceBeanName = "myUserDetailsService")
    public void test1() {
        principal = UserPrincipalGetter.getPrincipal();
        assertEquals("jan.rocek@gmail.com", service.getUserByID(3, principal).getEmail());
    }

    //nepostojeci id
    @Test
    @WithUserDetails(value = "jan.rocek@gmail.com",
        userDetailsServiceBeanName = "myUserDetailsService")
    public void test2() {
        principal = UserPrincipalGetter.getPrincipal();
        assertThrows(EntityMissingException.class,
            () -> service.getUserByID(1101, principal));
    }

    // ne-adminski dohvat tuđeg profila rezultira skrivanjem lokacije
```

```

@Test
@WithUserDetails(value = "matea.lipovac@gmail.com",
userDetailsServiceBeanName = "myUserDetailsService")
public void test3() { principal = UserPrincipalGetter.getPrincipal();
assertNull(service.getUserByID(7, principal).getLocation()); }

//neprijavljen korisnik
@Test
public void test4() { assertThrows(AuthenticationCredentialsNotFoundException.class
() -> service.getUserByID(1101, principal)); }

}

```

Ispitni sličaj 5: Testiranje funkcionalnosti metode fetch()

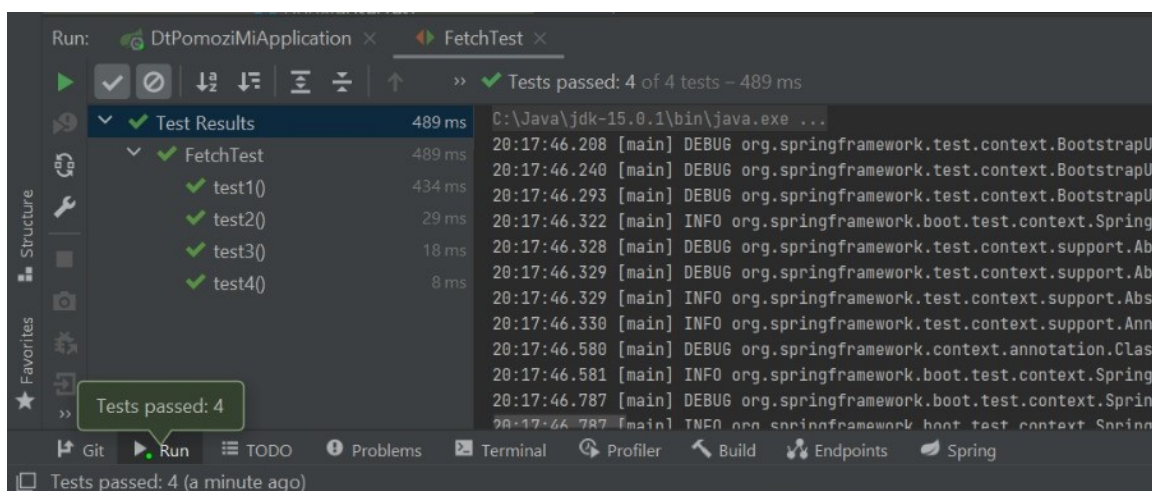
Ulaz:

1. Id korisnika kojeg se dohvaća

Očekivani rezultat:

1. korisnik s traženim idom
2. pripadajuća pogreška

Rezultat: Očekivani rezultat je zadovoljen, aplikacija je prošla sve testove



Slika 5.5: FetchTest()

Kod unit testa:

```
package NULL.DTPomoziMi.service.impl;

import static org.junit.jupiter.api.Assertions.assertEquals;
import static org.junit.jupiter.api.Assertions.assertThrows;

import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.security.authentication.
AuthenticationCredentialsNotFoundException;
import org.springframework.security.test.context.support.WithUserDetails;
import org.springframework.test.context.ActiveProfiles;

import NULL.DTPomoziMi.exception.EntityMissingException;
import NULL.DTPomoziMi.repository.UserRepo;
import NULL.DTPomoziMi.security.UserPrincipal;
import NULL.DTPomoziMi.service.UserService;
import NULL.DTPomoziMi.util.UserPrincipalGetter;

@SpringBootTest
@ActiveProfiles("dev")
public class FetchTest {

    @Autowired
    private UserService service;

    @Autowired
    private UserRepo userRepo;

    //dohvat samog sebe
    @Test
    @WithUserDetails(value = "jan.rocek@gmail.com",
userDetailsServiceBeanName = "myUserDetailsService")
    public void test1() {
```

```
UserPrincipal principal = UserPrincipalGetter.getPrincipal();
assertEquals(principal.getUser(), service.fetch(3));
}

//dohvat postojećeg korisnika
@Test
@WithUserDetails(value = "jan.rocek@gmail.com",
userDetailsServiceBeanName = "myUserDetailsService")
public void test2() { assertEquals(userRepo.findByEmail("matea.lipovac@gmail.com"),
service.fetch(12)); }

//dohvat nepostojećeg korisnika
@Test
@WithUserDetails(value = "jan.rocek@gmail.com",
userDetailsServiceBeanName = "myUserDetailsService")
public void test3() { assertThrows(EntityMissingException.class,
() -> service.fetch(90)); }

//neprijavljeni korisnik
@Test
public void test4() { assertThrows(AuthenticationCredentialsNotFoundException.class,
() -> service.fetch(12)); }
}
```

Ispitni slučaj 6: Testiranje funkcionalnosti metode deleteRequest()

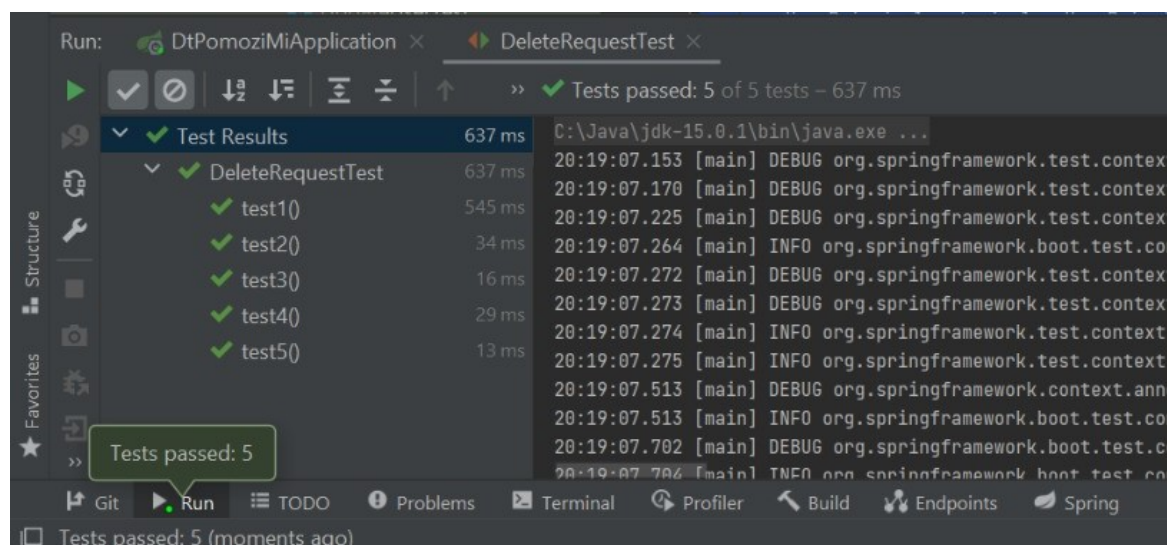
Ulaz:

1. id zahtjeva koji se briše
2. User principal

Očekivani rezultat:

1. zahtjev sa statusom obrisani (deleted)
2. pripadajuća pogreška

Rezultat: Očekivani rezultat je zadovoljen, aplikacija je prošla sve testove



Slika 5.6: DeleteRequestTest()

Kod unit testa:

```

package NULL.DTPomoziMi.service.impl;

import static org.junit.jupiter.api.Assertions.assertEquals;
import static org.junit.jupiter.api.Assertions.assertThrows;

import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.security.authentication.
AuthenticationCredentialsNotFoundException;
import org.springframework.security.test.context.support.WithUserDetails;
import org.springframework.test.context.ActiveProfiles;

import NULL.DTPomoziMi.exception.EntityMissingException;
import NULL.DTPomoziMi.exception.IllegalAccessException;
import NULL.DTPomoziMi.exception.IllegalActionException;
import NULL.DTPomoziMi.model.RequestStatus;
import NULL.DTPomoziMi.security.UserPrincipal;
  
```

```
import NULL.DTPomoziMi.util.UserPrincipalGetter;

@SpringBootTest
@ActiveProfiles("dev")
public class DeleteRequestTest {

    @Autowired
    private RequestServiceImpl service;

    //brisanje vlastitog aktivnog zahtjeva
    @Test
    @WithUserDetails(value = "matea.lipovac@gmail.com",
        userDetailsServiceBeanName = "myUserDetailsService")
    public void test1() {
        UserPrincipal principal = UserPrincipalGetter.getPrincipal();

        assertEquals(RequestStatus.DELETED,
            service.deleteRequest(35, principal).getStatus());
        assertThrows(EntityMissingException.class, () -> { service.fetch(35); });
    }

    //admin brise zahtjev
    @Test
    @WithUserDetails(value = "jan.rocek@gmail.com",
        userDetailsServiceBeanName = "myUserDetailsService")
    public void test2() {
        UserPrincipal principal = UserPrincipalGetter.getPrincipal();

        assertEquals(RequestStatus.DELETED,
            service.deleteRequest(11, principal).getStatus());
        assertThrows(EntityMissingException.class, () -> { service.fetch(11); });
    }

    //nevlasteno brisanje tuđeg zahtjeva
    @Test
```

```
@WithUserDetails(value = "robert.dakovic@gmail.com",
userDetailsServiceBeanName = "myUserDetailsService")
public void test3() {
    UserPrincipal principal = UserPrincipalGetter.getPrincipal();
    assertThrows(IllegalAccessException.class,
    () -> service.deleteRequest(25, principal).getStatus());
}

// zahtjev već ima izvršitelja
@Test
@WithUserDetails(value = "matea.lipovac@gmail.com",
userDetailsServiceBeanName = "myUserDetailsService")
public void test4() {
    UserPrincipal principal = UserPrincipalGetter.getPrincipal();

    assertThrows(IllegalActionException.class,
    () -> service.deleteRequest(34, principal).getStatus());
}

//neprijavljen korisnik
@Test
public void test5() {
    UserPrincipal principal = UserPrincipalGetter.getPrincipal();
    assertThrows(AuthenticationCredentialsNotFoundException.class,
    () -> service.deleteRequest(6, principal).getStatus());
}
}
```

Ispitni slučaj 7: Testiranje funkcionalnosti metode `blockUser()`

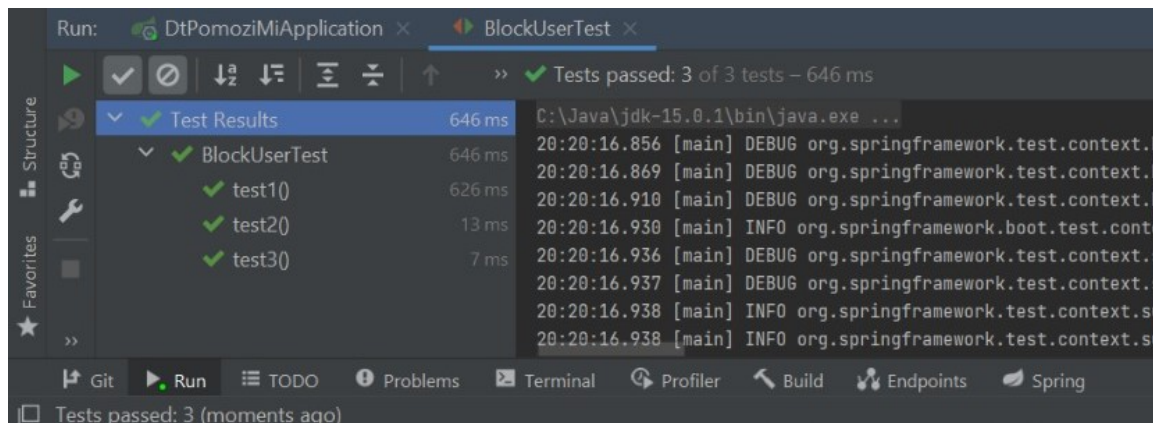
Ulaz:

1. id korisnika koji se blokira
2. enabled status - false

Očekivani rezultat:

1. enabled status blokiranog korisnika je false
2. pripadajuća pogreška

Rezultat: Očekivani rezultat je zadovoljen, aplikacija je prošla sve testove



Slika 5.7: BlockUserTest()

Kod unit testa:

```
package NULL.DTPomoziMi.service.impl;

import static org.junit.jupiter.api.Assertions.assertEquals;
import static org.junit.jupiter.api.Assertions.assertThrows;

import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.security.access.AccessDeniedException;
import org.springframework.security.authentication.
AuthenticationCredentialsNotFoundException;
import org.springframework.security.test.context.support.WithUserDetails;
import org.springframework.test.context.ActiveProfiles;

import NULL.DTPomoziMi.service.UserService;

@SpringBootTest
@ActiveProfiles("dev")
public class BlockUserTest {
```



```
@Autowired
```

```
private UserService service;
```

```
// korisnik sa id=12 postoji u bazi jer se prilikom pokretanja aplikacije  
nad bazom izvode upiti
```

```
// uneseni u data.sql file ==> INSERT INTO korisnik (...) VALUES (12, ...)
```

```
//admin blokira korisnika
```

```
@Test
```

```
@WithUserDetails(value = "jan.rocek@gmail.com",  
userDetailsServiceBeanName = "myUserDetailsService")
```

```
public void test1() {
```

```
assertEquals(false,service.blockUnblockUser(12, false).getEnabled());
```

```
}
```

```
//korisnik koji nije admin pokusava blokirati drugog korisnika
```

```
@Test
```

```
@WithUserDetails(value = "robert.dakovic@gmail.com",  
userDetailsServiceBeanName = "myUserDetailsService")
```

```
public void test2() {
```

```
assertThrows(AccessDeniedException.class,()->service.blockUnblockUser(12, false));
```

```
}
```

```
//neprijavljen korisnik
```

```
@Test
```

```
public void test3() {
```

```
assertThrows(AuthenticationCredentialsNotFoundException.class,
```

```
() -> service.blockUnblockUser(12, false));
```

```
}
```

```
}
```

5.2.2 Ispitivanje sustava

Ispitivanje sustava izvedeno je koristeći radni okvir Selenium IDE kako bi se provjerila osnovna funkcionalnost sustava po obrascima uporabe.

Ispitni slučaj 1: Registracija korisnika

Ulaz:

1. Otvara se obrazac za registraciju
2. Ime
3. Prezime
4. Email
5. Zaporka i njena potvrda
6. Država
7. Mjesto
8. Adresa
9. Pritisak na gumb "Registracija"

Očekivani rezultat:

1. Uspješna registracija i prikaz početne(home) stranice
2. Pripadajuća pogreška

Rezultat: Očekivani rezultat je zadovoljen, aplikacija je prošla sve testove

Na slici 5.8 je prikaz uspješno provedenog testa registracije u Selenium IDE.

Na slici 5.9 je prikaz obrasca prilikom neispravne registracije te pripadajuća greška.

Selenium IDE - Pomozi_mi* - Mozilla Firefox

Project: Pomozi_mi*

Tests +

Search tests...

https://null-pomozi-mi.herokuapp.com

	Command	Target	Value
7	type	id=lastName	Anic
8	click	id=email	
9	type	id=email	ana.anic@gmail.com
10	click	id=password	
11	type	id=password	AnaAnic1@
12	click	id=secondPassword	

Command

Target

Value

Description

Log Reference

16. click on id=town OK 21:44:52

17. type on id=town with value Split OK 21:44:53

18. click on id=address OK 21:44:54

19. type on id=address with value Ilica 24 OK 21:44:56

20. click on css=button:nth-child(2) OK 21:44:57

'Registracija' completed successfully 21:44:57

Slika 5.8: Uspješan prolazak testa registracije

Izradite svoj **PomoziMi** račun

Ime*: Ana

Prezime*: Dodic

E-mail*: anabanana
Unesite e-mail ispravnog formata

Zaporka*: ●●●●●
Zaporka mora imati barem 8 znakova

Potvrdi*: ●●●
Potvrda mora biti jednaka zaporci

Država: Crna Gora

Mjesto: Mjesto

Slika 5.9: Krivi unos u polja registracije

Ispitni slučaj 2: Prijava korisnika u sustav**Ulaz:**

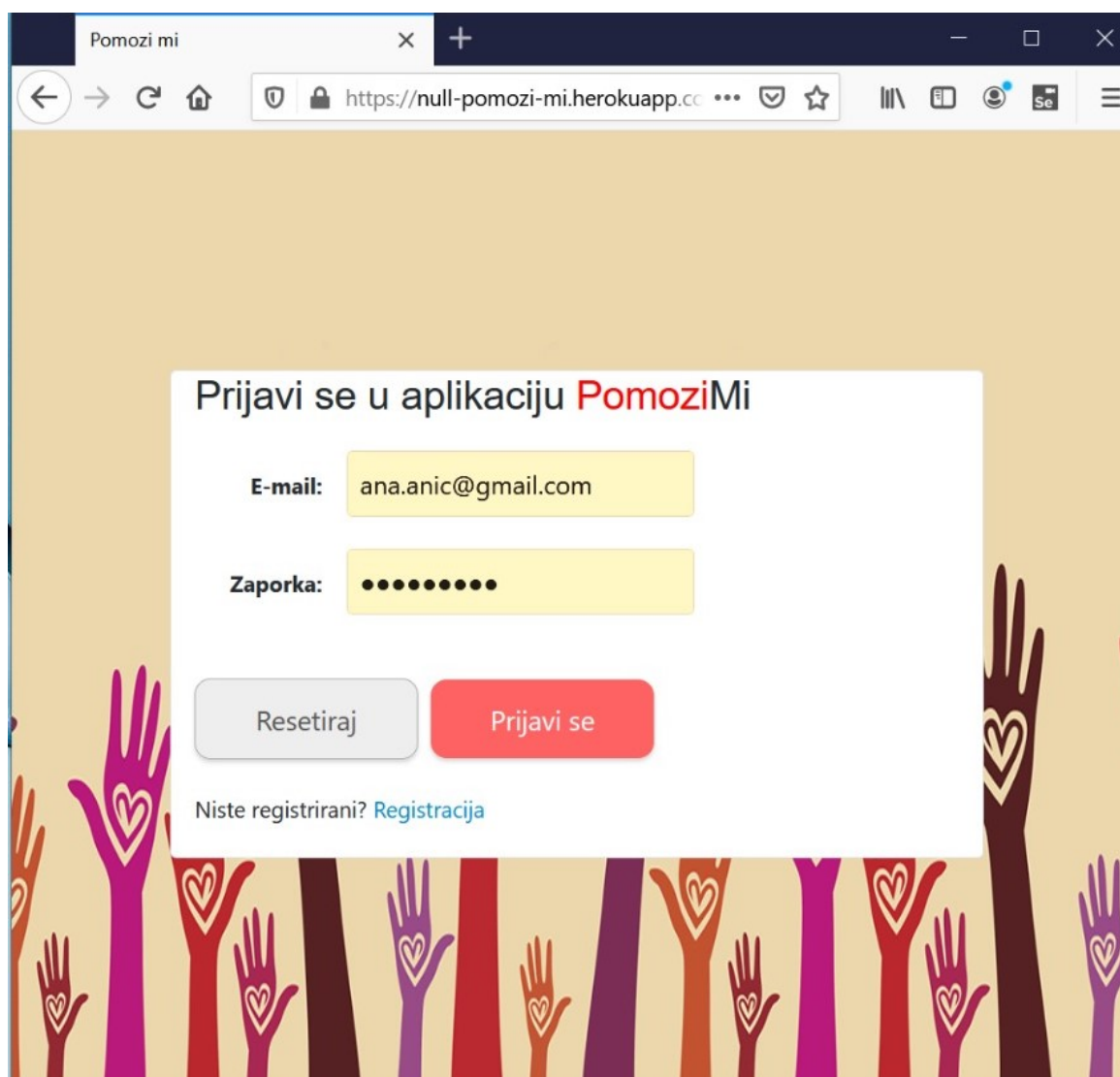
1. Otvara se obrazac za prijavu
2. Email
3. Lozinka
4. Pritisak na gumb "Prijavi se"

Očekivani rezultat:

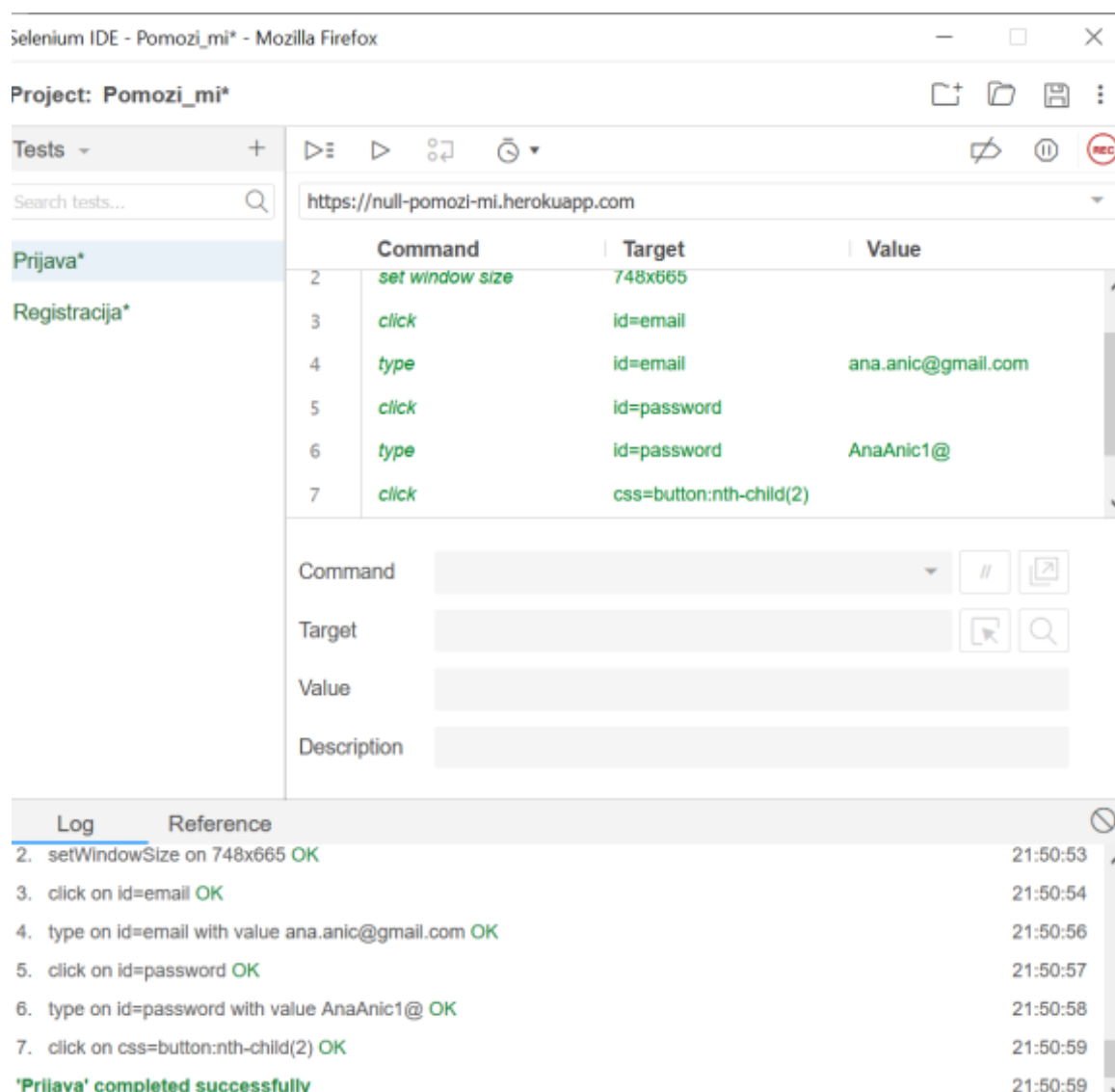
1. Prikazuje se obrazac za prijavu korisnika u sustav
2. Nakon uspješne prijave prikaz glavne stranice

Rezultat: Očekivani rezultat je zadovoljen, aplikacija je prošla sve testove

Na slici 5.10 vidi se obrazac za prijavu korisnika u sustav, a na slici 5.11 kako u Seleniumu IDE prolazi test za uspješnu prijavu.



Slika 5.10: Prijava korisnika u sustav



Slika 5.11: Uspješan prolazak testa prijave

Ispitni slučaj 3: Zadavanje zahtjeva

Ulaz:

1. Na izborniku odabrati "Zadavanje zahtjeva"
2. Unos broja mobitela
3. Unos opisa zahtjeva
4. Lokacija : Drzava, mjesto adresa ili odabir na karti
5. Spremanje podataka
6. Odjava iz aplikacije

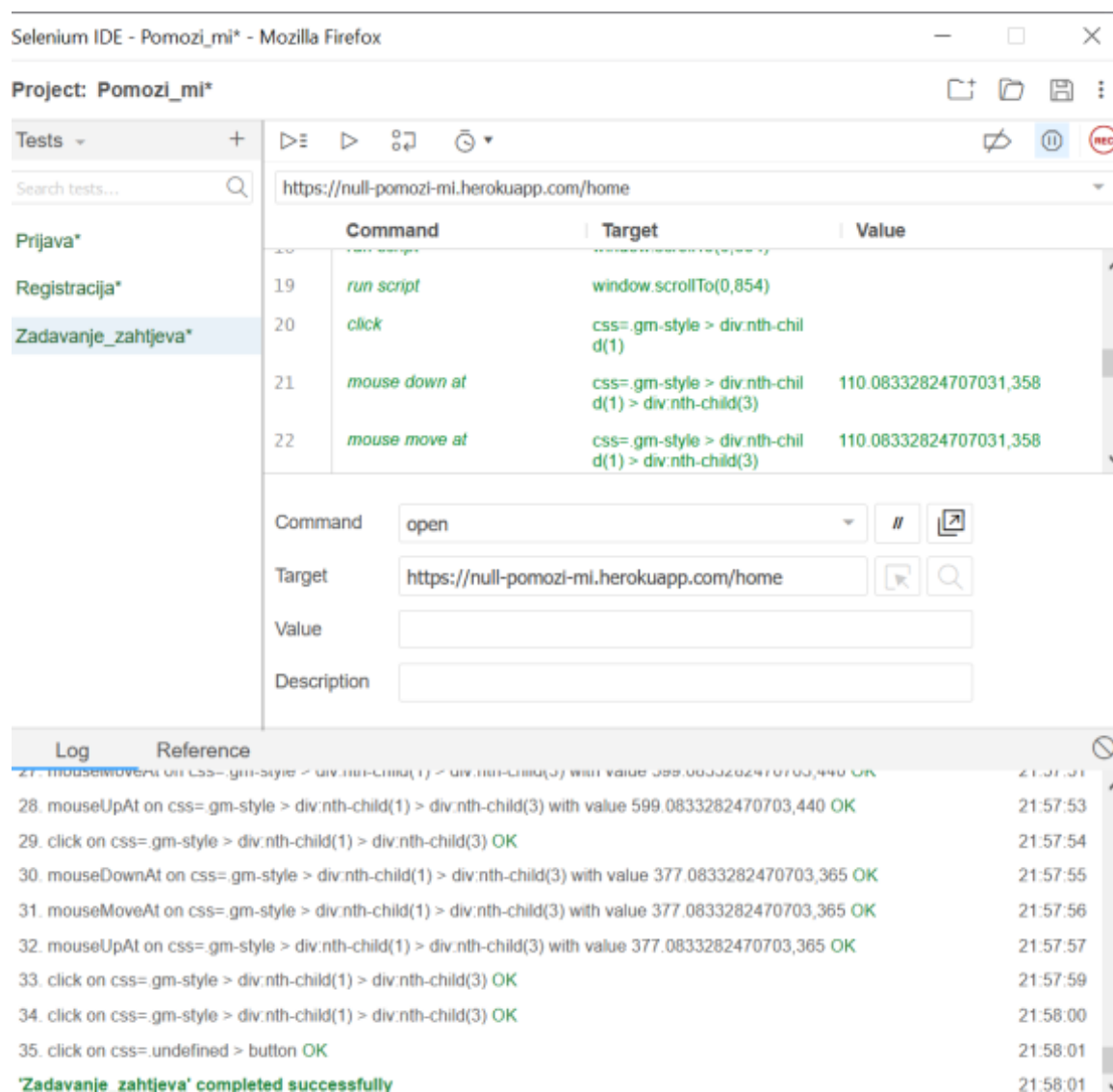
Očekivani rezultat:

1. Prikazuje se obrazac za zadavanje zahtjeva za pomoć

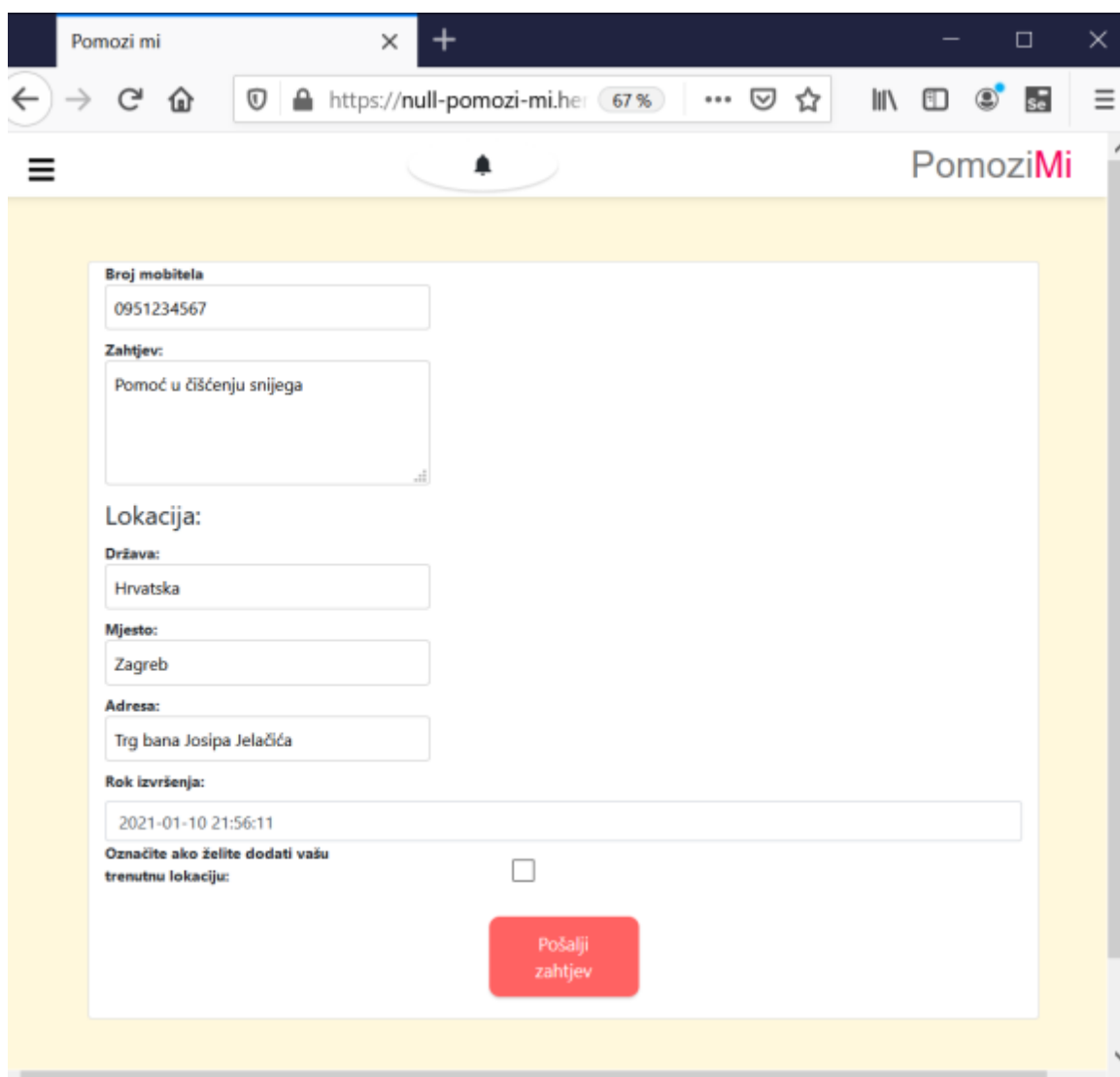
2. Opcionalni prikaz karte za lokaciju
3. Nakon uspješnog unosa podataka, pritisak na gumb "Pošalji zahtjev"

Rezultat: Očekivani rezultat je zadovoljen, aplikacija je prošla sve testove

Na slici 5.12 vidi se kako u Seleniumu IDE prolazi test za uspješnu prijavu, a na slici 5.13 obrazac za zadavanje zahtjeva.



Slika 5.12: Uspješan prolazak testa zadavanja zahtjeva



The screenshot shows a web browser window with the address bar displaying 'https://null-pomozi-mi.hr'. The page has a yellow background and a white form area. The form contains the following fields and elements:

- Broj mobitela:** A text input field containing '0951234567'.
- Zahtjev:** A text area containing 'Pomoć u čišćenju snijega'.
- Lokacija:** A section header for location details.
- Država:** A text input field containing 'Hrvatska'.
- Mjesto:** A text input field containing 'Zagreb'.
- Adresa:** A text input field containing 'Trg bana Josipa Jelačića'.
- Rok izvršenja:** A text input field containing '2021-01-10 21:56:11'.
- Označite ako želite dodati vašu trenutnu lokaciju:** A checkbox that is currently unchecked.
- Pošalji zahtjev:** A red button with white text.

Slika 5.13: Obrazac zadavanja zahtjeva za pomoć

Ispitni slučaj 4: Odabir zahtjeva za izvršavanje

Ulaz:

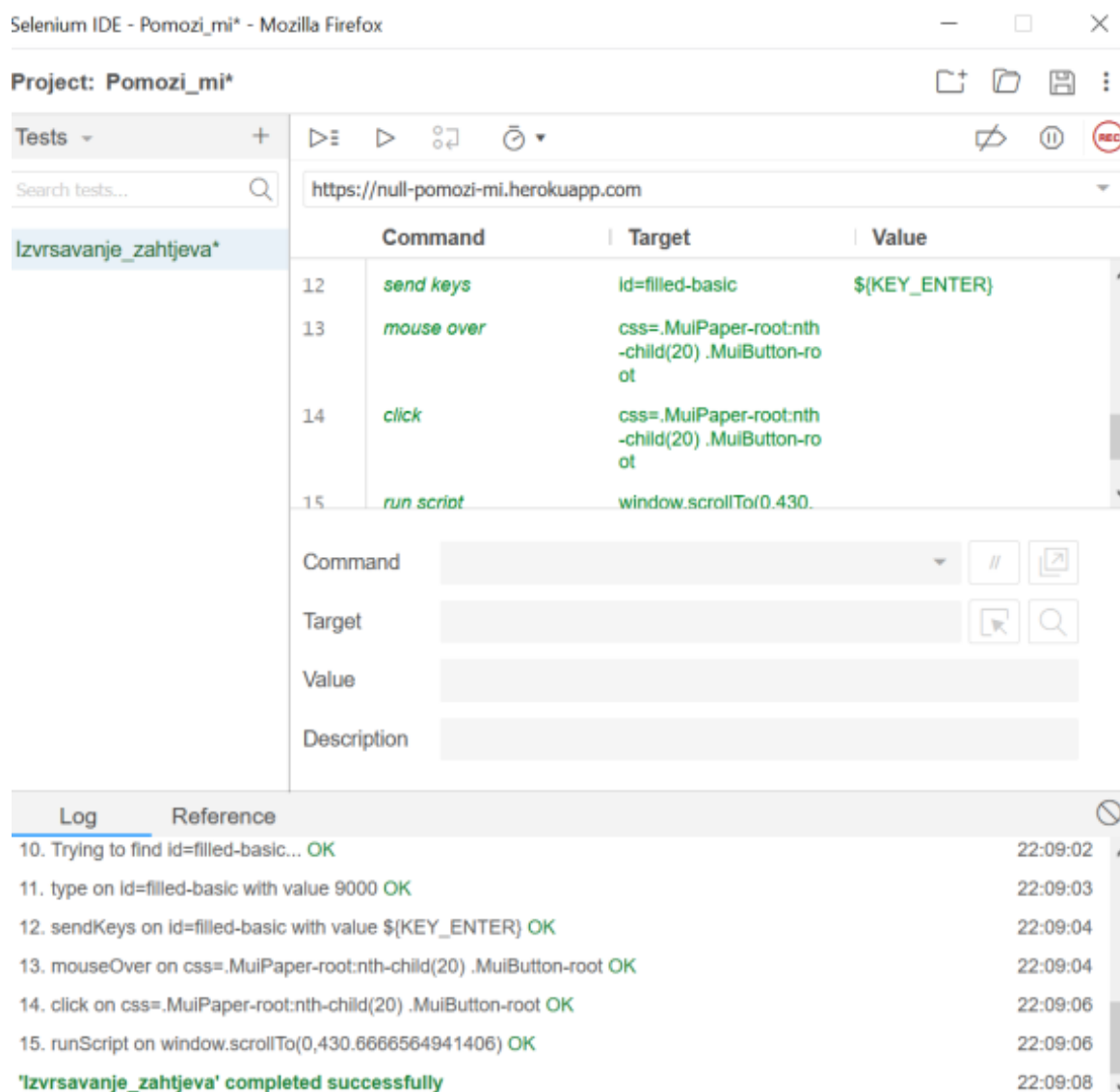
1. Na izborniku odabrati "Pregled zahtjeva"
2. Opcionalno unos radijusa za filtraciju
3. Odabir gumba "izvrši zahtjev"

Očekivani rezultat:

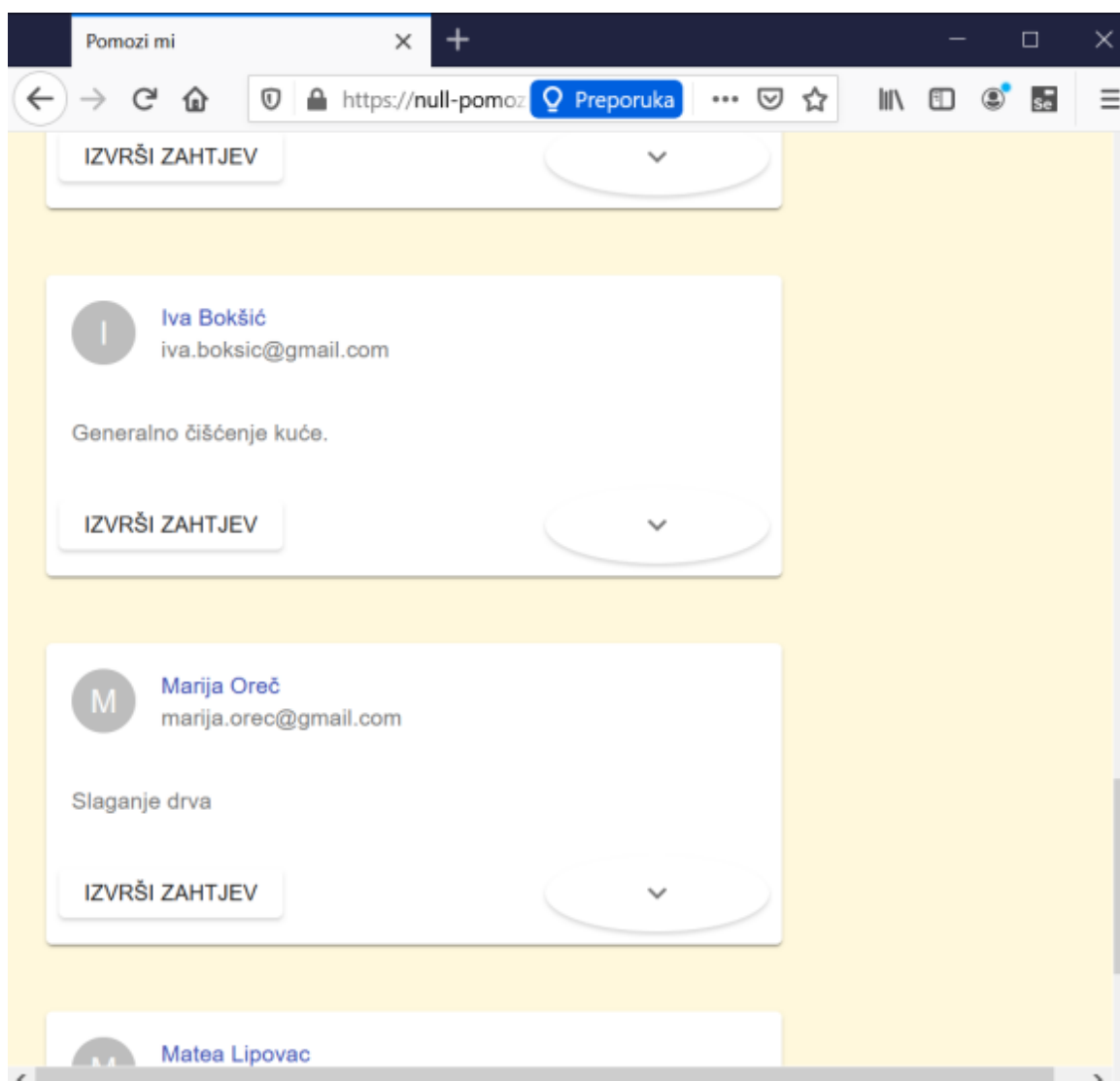
1. Prikazuje se izbornik na početnoj stranici
2. Prikaz liste aktivnih zahtjeva
3. Opcionalni prikaz zahtjeva za unešeni radijus
4. Nakon odabira izvršavanja šalje se notifikacija autoru

Rezultat: Očekivani rezultat je zadovoljen, aplikacija je prošla sve testove

Na slici 5.14 vidi se kako u Seleniumu IDE prolazi test za uspješan odabir izvršavanja zahtjeva, a na slici 5.15 prikaz liste aktivnih zahtjeva koje se pritiskom na gumb "Izvrši zahtjev" može odabrati za izvršavanje.



Slika 5.14: Uspješan prolazak testa izvršavanja zahtjeva



Slika 5.15: Prikaz liste aktivnih zahtjeva

Ispitni slučaj 5: Ocjenjivanje korisnika

Ulaz:

1. Na izborniku odabrati "Korisnici"
2. Kliknuti na ime tražene osobe
3. Na profilu korisnika stisnuti gumb za ocjenjivanje 4. odabrati ocijena 5. stisnuti gumb "ocijeni"

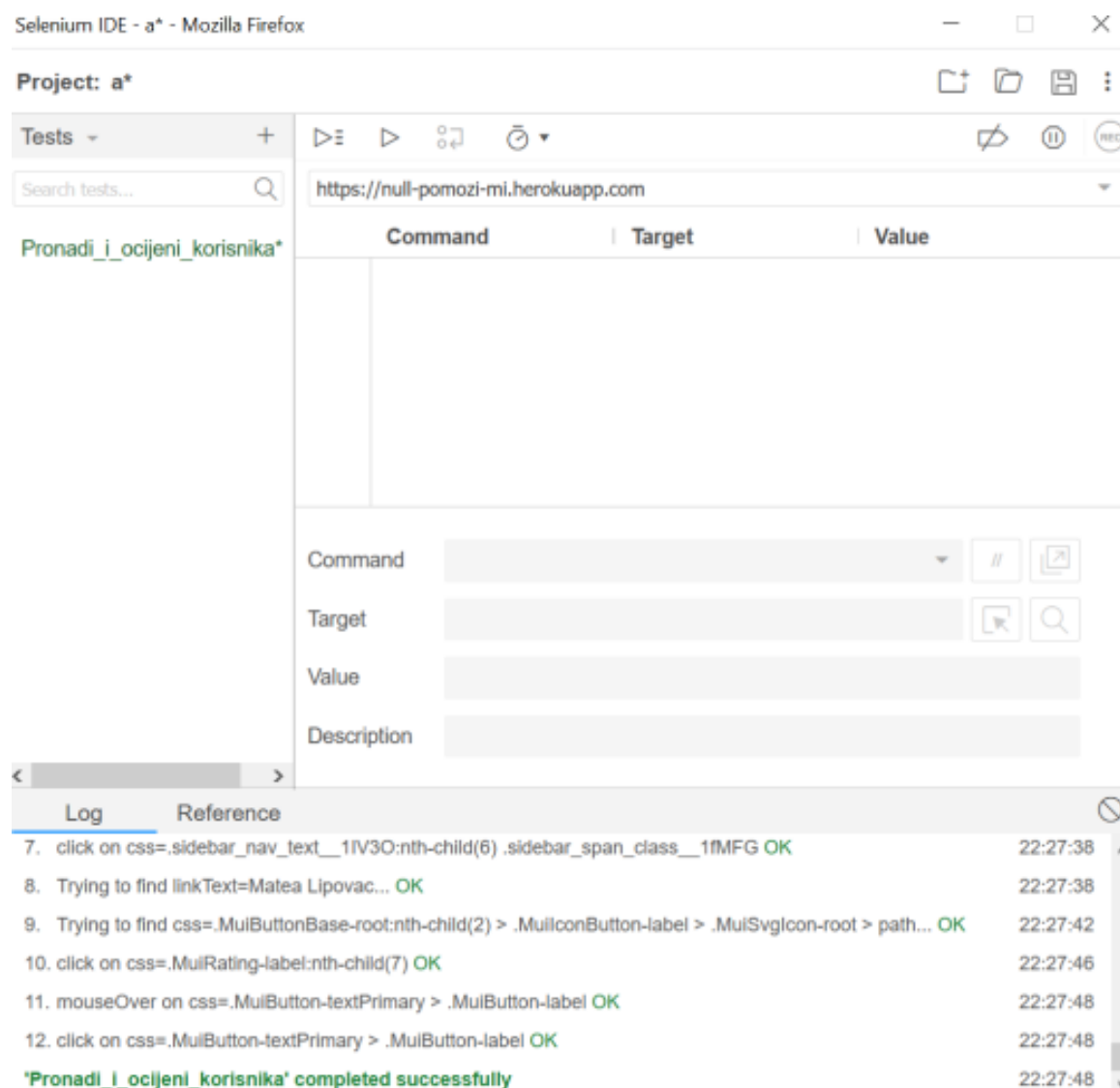
Očekivani rezultat:

1. Prikazuje se izbornik na početnoj stranici
2. Prikaz liste korisnika
3. Nakon odabira imena prikaz profila

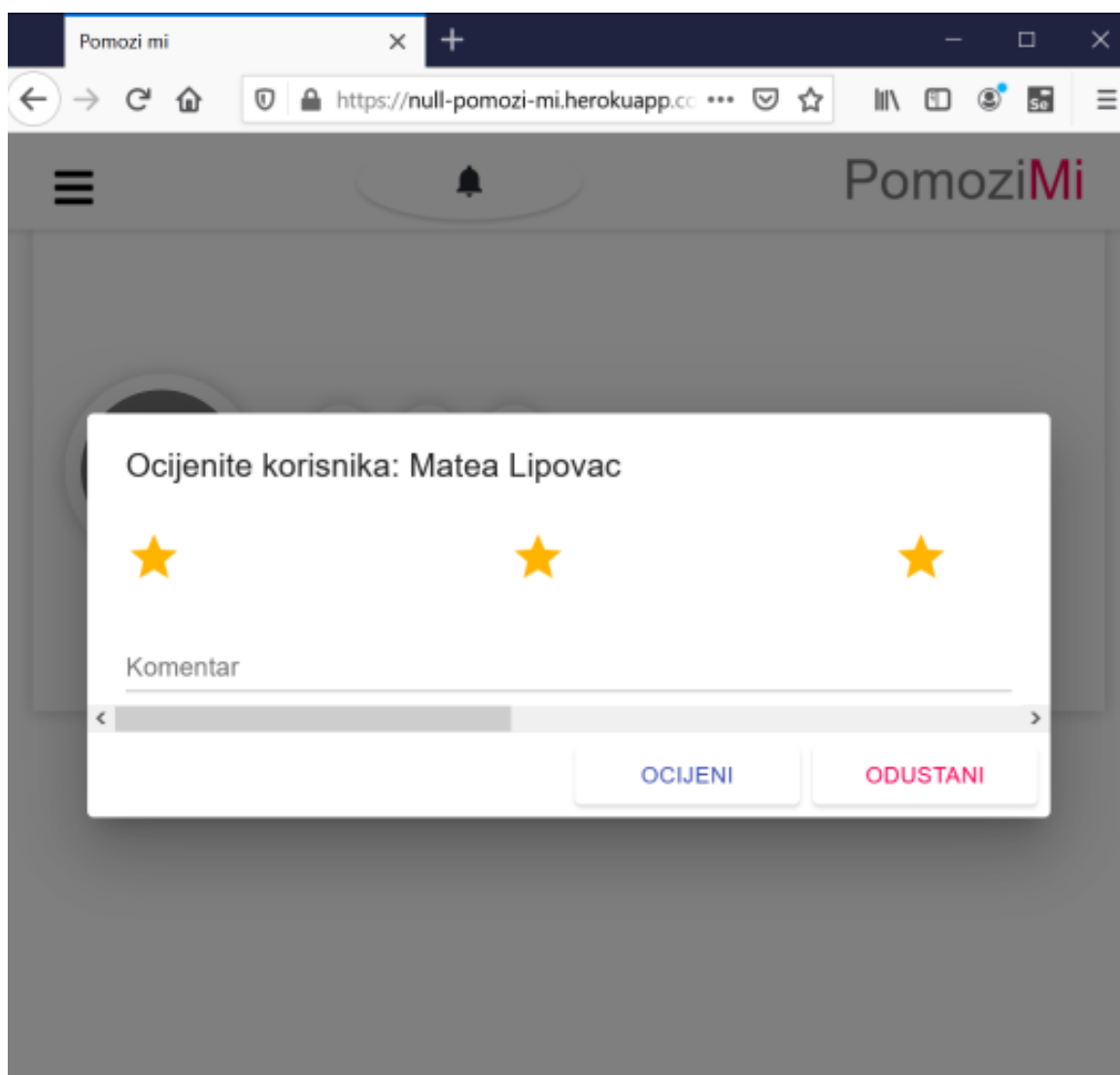
4. Mogućnost ocjenjivanja

Rezultat: Očekivani rezultat je zadovoljen, aplikacija je prošla sve testove

Na slici 5.14 vidi se kako u Seleniumu IDE prolazi test za ocjenjivanje drugog korisnika, a na slici 5.15 prikaz obrasca za ocjenjivanje.



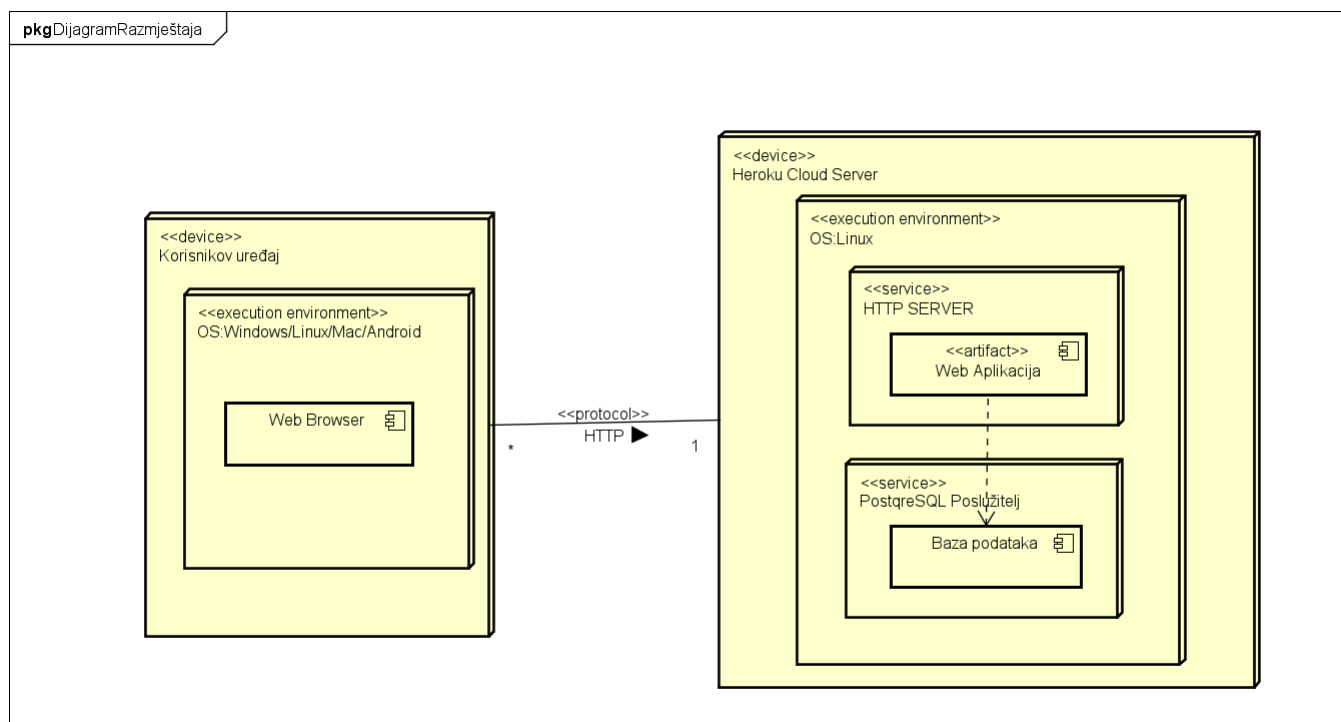
Slika 5.16: Uspješan prolazak testa ocjenjivanja korisnika



Slika 5.17: Obrazac za ocjenjivanje

5.3 Dijagram razmještaja

Dijagrami razmještaja opisuju topologiju sklopovlja i programsku potporu koja se koristi u implementaciji sustava u njegovom radnom okruženju. Na poslužitelju (HerokuCloudServer) je postavljen Linux operacijski sustav te se na njemu nalaze HTTP Server i poslužitelj baze podataka. Klijenti koriste web preglednik na svom uređaju(PC ili mobitel) kako bi pristupili web aplikaciji. Sustav je baziran na arhitekturi "klijent – poslužitelj", a komunikacija između računala korisnika (klijent, zaposlenik, vlasnik, administrator) i poslužitelja odvija se preko HTTP veze.



Slika 5.18: Dijagram Razmještaja

5.4 Upute za puštanje u pogon

U poglavlju 5.4 bit će opisan postupak puštanja Spring-Boot aplikacije u pogon na Heroku poslužitelj. Koraci koje je potrebno izvesti prilikom te akcije su navedeni ispod ovog teksta, a određene točke su dodatno pojašnjenje u ovom poglavlju ispod točaka.

1. Registrirati se s besplatnim računom na servisu Heroku¹⁵
2. Instalirati Heroku CLI ¹⁶
3. Prijaviti se na Heroku servis pomoću Heroku CLI (naredba: `heroku login`)
4. Dodati Postgresql add-on (bazu) na Heroku
5. Inicijalizirati git repozitorij, dodati kod aplikacije i napraviti prvi commit
6. Kreirati heroku app naredbom: `heroku create`
7. Deploy-ati aplikaciju naredbom: `git push heroku master`

4. Dodavanje Postgresql na Heroku:

Sve što je potrebno napraviti u ovom koraku je izvesti sljedeću naredbu:

```
$ heroku addons:create heroku-postgresql
```

Slika 5.19: Naredba za dodavanje Postgresql-a

Nakon izvođenja te naredbe, Heroku će automatski kreirati varijable okruženja `SPRING_DATASOURCE_URL`, `SPRING_DATASOURCE_USERNAME` i `SPRING_DATASOURCE_PASSWORD`. Te varijable će omogućiti aplikaciji da se spoji na Postgresql bazu dokle god imamo dodan *postgresql* dependency u `pom.xml`-u.

5. Inicijalizacija git repozitorija i prvi commit:

```
$ git init
$ git add .
$ git commit -m "first commit"
```

Slika 5.20: Naredba za dodavanje Postgresql-a

¹⁵<https://devcenter.heroku.com/>

¹⁶<https://devcenter.heroku.com/articles/heroku-cli>

6. Kreiranje aplikacije na Heroku:

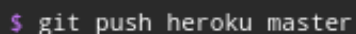
A dark-themed terminal window showing the command `$ heroku create` in a light-colored font.

Slika 5.21: Naredba za dodavanje Postgresql-a

Ova naredba će kreirati aplikaciju na Heroku i pripremiti sve za prvi deploy. Nakon izvršavanja te naredbe, naš lokalni git repozitorij koji smo u koraku prije inicijalizirali će biti također povezan na remote heroku git repozitorij pripremljen za našu aplikaciju.

7. Puštanje u pogon:

Sada smo spremni za deploy koji obavljamo izvršavanjem sljedeće naredbe:

A dark-themed terminal window showing the command `$ git push heroku master` in a light-colored font.

Slika 5.22: Naredba za dodavanje Postgresql-a

Navedena naredba će poslati kod aplikacije na Heroku poslužitelj te će biti pokrenut build postupak. Kada build postupak završi, naša aplikacija se pokreće te joj je moguće pristupiti na web adresi koja će biti ispisana u komandnoj liniji nakon završetka izvođenja naredbe.

6. Zaključak i budući rad

Naš projektni zadatak predmeta "Programsko inženjerstvo" bio je napraviti web aplikaciju "Pomozi mi" koja bi omogućila korisnicima međusobno pružanje pomoći. Pomoć se odnosila na svakodnevne obveze ili poteškoće. Korisnici u svakom trenutku imaju mogućnost pregleda aktivnih zahtjeva i njihovog izvršavanja. Osim toga, mogu sami zatražiti pomoć ispunjavanjem obrasca u kojemu ga pobliže opisuju zajedno sa svim nužnim informacijama.

Naš prvi susret sastojao se od međusobnog upoznavanja, komentiranja dane teme te organiziranja buduće komunikacije. Za raspodjelu poslova često smo koristili Gitlab, ali većina komunikacije i sastanaka odvijala se putem MS Teams i WhatsApp-a. U prvom ciklusu našega rada uglavnom smo se usredotočili na pisanje dokumentacije i upoznavanje sa novim tehnologijama. Svima nam je to bio prvi susret sa složenijim projektom, pa samim time i dokumentacijom, ali smo uz pomoć predavanja na kolegiju i kontinuiranu komunikaciju sa asistentima to uspješno savladali. Time smo dobili uvid u važnost dokumentiranja projekta unaprijed te kako komponente poput obrazaca uporabe ili različitih dijagrama pružaju pregled funkcionalnosti. Sve navedeno nam je poslužilo kao okvir za budući rad i uvelike olakšalo implementaciju.

U drugoj fazi naglasak je bio na izradi same aplikacije. Neki članovi su se susreli sa alatima koje smo koristili za programsku implementaciju, no i oni su naišli na puno novih prepreka te proširili svoje znanje. U ovom razdoblju organizacija je bila ključna te se svaki član trudio držati dogovorenih rokova. Nakon napisanog koda uslijedilo je testiranje sustava, nadopunjavanje dokumentacije i završni prepravci. Međusobno smo si pomagali, što je rezultiralo ugodnom i produktivnijom atmosferom.

Nakon 8 tjedana rada, aplikacija je bila završena. Iako je malo drugačija od početne vizije, tražene funkcionalnosti su uspješno implementirane. Jedna od svari koje smo naučili je da se prvobitni plan često mijenja i prilagođava tijekom rada na samom projektu. Neki od problema s kojima smo se susretali su bili rad s Gitom, uključivanje karte u aplikaciju i povezivanje korisničkog sučelja sa poslužiteljem. Sve prepreke smo na kraju uspješno savladali i iz njih stekli znanje. Cijelo iskus-

tvo dalo nam je uvid u timski rad i sve faze izrade projekta te će nam svima biti koristan za osobni rast kao budućih inženjera.

Popis literature

1. Programsko inženjerstvo, FER ZEMRIS, <http://www.fer.hr/predmet/proinz>
2. I. Sommerville, "Software engineering", 8th ed, Addison Wesley, 2007.
3. The Unified Modeling Language, <https://www.uml-diagrams.org/>
4. Astah Community, <http://astah.net/editions/uml-new>
5. Sprint Boot, <https://docs.spring.io/spring-boot/docs/current-SNAPSHOT/reference/htmlsingle/>
6. Puštanje u pogon Spring-Boot aplikacije, <https://devcenter.heroku.com/articles/deploying-spring-boot-apps-to-heroku>

Indeks slika i dijagrama

2.1	odabir nakon prijave	6
2.2	proces izvršavanja zahtjeva	7
3.1	Dijagram obrasca uporabe, funkcionalnosti korisnika	21
3.2	Dijagram obrasca uporabe, funkcionalnosti administratora	22
3.3	Dijagram obrasca uporabe, razrada funkcionalnosti profila	23
3.4	Dijagram obrasca uporabe, prikaz aktivnosti vezanih za upravljanje zahtjeva	24
3.5	Dijagram obrasca uporabe, prikaz aktivnosti vezanih za odabir zah- tjeva	25
3.6	Dijagram obrasca uporabe, razrada funkcionalnosti filtriranja	26
3.7	Sekvencijski dijagram za UC1	28
3.8	Sekvencijski dijagram za UC6	29
3.9	Sekvencijski dijagram za UC3	30
3.10	Sekvencijski dijagram za UC13	31
4.1	Arhitektura sustava	33
4.2	Arhitektura sustava	34
4.3	Arhitektura sustava	35
4.4	Dijagram baze podataka	41
4.5	Dijagram razreda za autentifikaciju	42
4.6	Dijagram razreda za korisnika	43
4.7	Dijagram razreda za zahtjeve	44
4.8	Dijagram razreda za lokaciju	45
4.9	Dijagram razreda za ocjenjivanje	46
4.10	Dijagram razreda za kandidiranje	47
4.11	Dijagram razreda za notifikacije	48
4.12	Dijagram stanja	49
4.13	Dijagram Aktivnosti	50
4.14	Dijagram Komponeneti	51

5.1	UpdateUserTest()	53
5.2	UpdateRequestTest()	57
5.3	PickForExecutionTest()	61
5.4	GetUserByIdTest()	63
5.5	FetchTest()	65
5.6	DeleteRequestTest()	68
5.7	BlockUserTest()	71
5.8	Uspješan prolazak testa registracije	74
5.9	Krivi unos u polja registracije	75
5.10	Prijava korisnika u sustav	76
5.11	Uspješan prolazak testa prijave	77
5.12	Uspješan prolazak testa zadavanja zahtjeva	78
5.13	Obrazac zadavanja zahtjeva za pomoć	79
5.14	Uspješan prolazak testa izvršavanja zahtjeva	80
5.15	Prikaz liste aktivnih zahtjeva	81
5.16	Uspješan prolazak testa ocjenjivanja korisnika	82
5.17	Obrazac za ocjenjivanje	83
5.18	Dijagram Razmještaja	84
6.1	Dijagram pregleda promjena na grani master	96

Dodatak: Prikaz aktivnosti grupe

Dnevnik sastajanja

1. sastanak

- Datum: 2. listopada 2020.
- Prisustvovali: I.Bokšić, I.Jakas, M.Lipovac, M.Oreč, J.Roček, R.Đaković, D.Ćurić
- Teme sastanka:
 - upoznavanje tima
 - razgovor o projektu i trenutnim tehnološkim znanjima članova

2. sastanak

- Datum: 8. listopada 2020.
- Prisustvovali: I.Bokšić, I.Jakas, M.Lipovac, M.Oreč, J.Roček, R.Đaković, D.Ćurić
- Teme sastanka:
 - razgovor s asistenticom i demonstratorom
 - analiza projekta
 - razješavanje prvih nedoumica oko zadatka

3. sastanak

- Datum: 10. listopada 2020.
- Prisustvovali: I.Bokšić, I.Jakas, M.Lipovac, M.Oreč, J.Roček, R.Đaković, D.Ćurić
- Teme sastanka:
 - razgovor o tehnologijama koje ćemo koristiti i njihova instalacija
 - okvirna podjela rada

4. sastanak

- Datum: 14. listopada 2020.
- Prisustvovali: I.Bokšić, I.Jakas, M.Lipovac, M.Oreč, J.Roček, R.Đaković, D.Ćurić
- Teme sastanka:

- definiranje funkcionalnih zahtjeva
- kreiranje use case dijagrama

5. sastanak

- Datum: 16. listopada 2020.
- Prisustvovali: I.Bokšić, M.Lipovac, M.Oreč, J.Roček, R.Đaković, D.Ćurić
- Teme sastanka:
 - konačno definiranje use case dijagrama
 - nova podjela zadataka

6. sastanak

- Datum: 26. listopada 2020.
- Prisustvovali: D.Ćurić
- Teme sastanka:
 - razgovor s asistenticom i demonstratorom
 - rasprava o sekvencijskim dijagramima

7. sastanak

- Datum: 3. studenog 2020.
- Prisustvovali: I.Bokšić, M.Lipovac, R.Đaković, D.Ćurić
- Teme sastanka:
 - raspodijela posla na dokumentaciji
 - prezentiranje login i sign in stranice

8. sastanak

- Datum: 3. studenog 2020.
- Prisustvovali: J.Roček
- Teme sastanka:
 - razgovor s demonstratorom
 - rasprava oko implementacije JWT-a

9. sastanak

- Datum: 5. studenog 2020.
- Prisustvovali: I.Bokšić, I.Jakas, M.Lipovac, M.Oreč, J.Roček, R.Đaković, D.Ćurić
- Teme sastanka:
 - razgovor s asistenticom i demonstratorom
 - komentiranje dijagrama baze podataka
 - sugestija oko izrade dijagrama razreda

10. sastanak

- Datum: 10. prosinca 2020.
- Prisustvovali: J.Roček, D.Ćurić, M.Oreč, M.Lipovac
- Teme sastanka:
 - dogovor oko implementacijskih detalja vezanih uz backend

11. sastanak

- Datum: 15. prosinac 2020.
- Prisustvovali: J.Roček, D.Ćurić
- Teme sastanka:
 - pregled napravljenih dijelova backend
 - definiranje novih zadataka za backend

12. sastanak

- Datum: 17. prosinca 2020.
- Prisustvovali: J.Roček, M.Oreč, M.Lipovac
- Teme sastanka:
 - pregled napravljenih dijelova backend
 - definiranje novih zadataka za backend

13. sastanak

- Datum: 22. prosinca 2020.
- Prisustvovali: J.Roček, D.Ćurić
- Teme sastanka:
 - pregled napravljenih dijelova backend
 - modificiranje postojećih metoda
 - definiranje novih zadataka za backend

14. sastanak

- Datum: 23. prosinca 2020.
- Prisustvovali: J.Roček, I.Jakas, R.Đaković, I.Bokšić
- Teme sastanka:
 - pregled napravljenih dijelova backend
 - definiranje zadataka za frontend

15. sastanak

- Datum: 27. prosinca 2020.
- Prisustvovali: I.Jakas, R.Đaković, I.Bokšić
- Teme sastanka:
 - pregled rada na frontu
 - definiranje novih zadataka na frontu

16. sastanak

- Datum: 3. siječnja 2021.
- Prisustvovali: J.Roček, M.Oreč, M.Lipovac
- Teme sastanka:
 - dogovor oko pisanja JUnit testova
 - dogovor oko same implementacije JUnit testova

17. sastanak

- Datum: 4. siječnja 2020.
- Prisustvovali: I.Jakas, R.Đaković, I.Bokšić, J.Roček
- Teme sastanka:
 - pregled fronta
 - novo definiranje zadataka

18. sastanak

- Datum: 7. siječnja 2020.
- Prisustvovali: I.Jakas, R.Đaković, I.Bokšić, J.Roček, M.Lipovac, M.Oreč, D.Ćurić
- Teme sastanka:
 - demonstriranje alfa verzije asistentici
 - popravci funkcionalnosti

19. sastanak

- Datum: 13. siječnja 2021.
- Prisustvovali: J.Roček, R.Đaković, I. Bokšić
- Teme sastanka:
 - konačni pregled aplikacije i popravci
 - rasprava oko dokumentacije

20. sastanak

- Datum: 14. siječnja 2021.
- Prisustvovali: J.Roček, I. Bokšić
- Teme sastanka:
 - pokazivanje novih funkcionalnosti asistentici
 - pregled dokumentacije

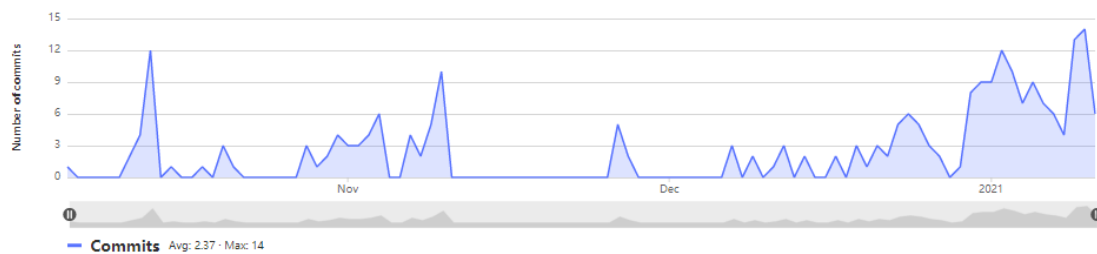
Tablica aktivnosti

	Iva Bokšić	Jan Roček	Ivan Jakas	Robert Đaković	Matea Lipovac	Dominik Čurić	Marija Oreč
Upravljanje projektom	2	3	2	1			
Opis projektnog zadatka							5
Funkcionalni zahtjevi	1	1	1	1	1	1	1
Opis pojedinih obrazaca	4				6		
Dijagram obrazaca	2	3		1		1	2
Sekvencijski dijagrami			2	2		5	
Opis ostalih zahtjeva							1
Arhitektura i dizajn sustava	4	2					
Baza podataka	2	5			1	1	
Dijagram razreda	1				2		3
Dijagram stanja						7	
Dijagram aktivnosti						4	
Dijagram komponenti						4	
Korištene tehnologije i alati						1	
Ispitivanje programskog rješenja		1			10		10
Dijagram razmještaja						1	
Upute za puštanje u pogon	1	1					
Dnevnik sastajanja	1					1	
Zaključak i budući rad					2		2
Popis literature	1						
Izrada login stranice	1		3	1			
Izrada baze podataka	1	2					
Integriranje fronta i backa		15					
Backend		100			20	20	20
Front	50	10	70	60			

Dijagrami pregleda promjena

Commits to master

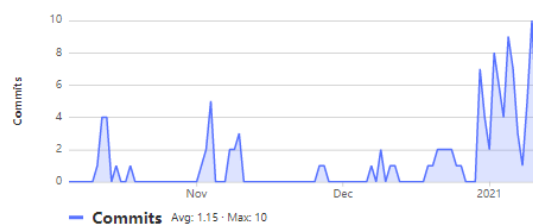
Excluding merge commits. Limited to 6,000 commits.



Slika 6.1: Dijagram pregleda promjena na grani master

Jan Roček

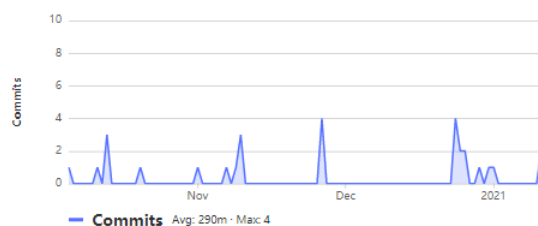
115 commits (jan.rocek@fer.hr)

**ivan**

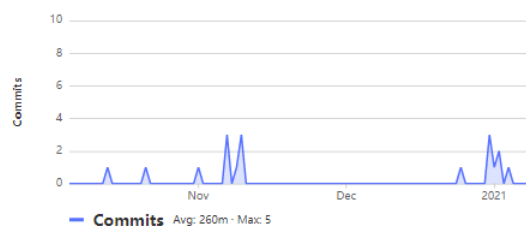
31 commits (ij51105@fer.hr)

**Iva Bokšić**

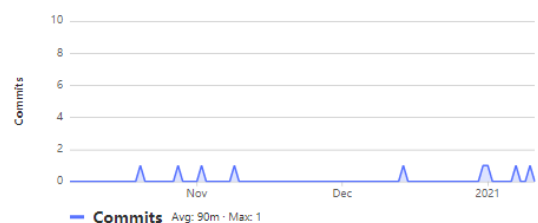
29 commits (ib51425@fer.hr)

**Robert**

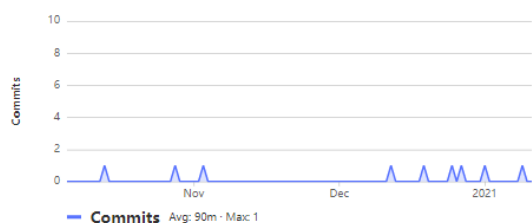
26 commits (rd51779@fer.hr)

**matealipovac**

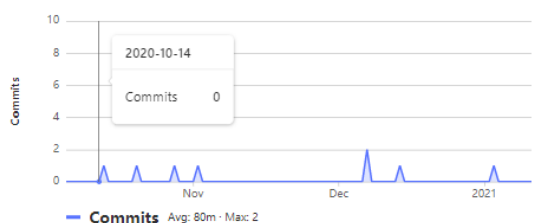
9 commits (72604162+matealipovac@users.noreply.github.com)

**Dominik12108**

9 commits (dominik.curic@fer.hr)

**MarijaOrec**

8 commits (marija.orec@fer.hr)

**IvaBoksic**

6 commits (iva.boksic099@gmail.com)

