Programsko inženjerstvo

Ak. god. 2020./2021.

Pomozi mi

Dokumentacija, Rev. 2

Grupa: *TODO*

Voditelj: *Mihaela Bakšić*

Datum predaje: 14. 1. 2021.

Nastavnik: *Dunja Tounec*

Sadržaj

1. Dnevnik promjena dokumentacije

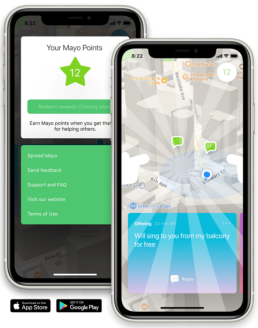
Kontinuirano osvježavanje

Rev.	Opis promjene/dodatka	Autori	Datum
0.1	Napravljen predložak.	Bakšić	16.10.2020.
0.2	Dodani opisi *Use Case* dijagrama.	Bakšić	28.10.2020.
0.3	Dodani funkcionalni zahtjevi.	Milde	31.10.2020.
0.4	Dodan opis projektnog zadatka.	Milde	31.10.2020.
0.5	Opisana arhitektura i dizajn sustava. Dodan opis i dijagram baze podataka.	Oreč	09.11.2020.
0.6	Dodani sekvencijski dijagrami i dijagrami obrazaca uporabe.	Rački	9.11.2020.
0.7	Dodani *Class* dijagrami.	Milde	11.11.2020.
0.8	Dodani opisi *Class* dijagrama.	Bakšić	11.11.2020.
1.0	Verzija za 1. ciklus predaje	Bakšić	11.11.2020.
1.1	Zaključak i budući rad	Bakšić	31.12.2020.
1.2	Dijagram stanja i revizija baze podataka	Oreč	3.1.2021.
1.3	Dijagram aktivnosti	Ilinović	4.1.2021.
1.4	Dijagrami komponenti i razmještaja	Bakšić	7.1.2021.
1.5	Revizija obrazaca uporabe i sekvencijskih dijagrama	Bakšić	9.1.2021.
1.6	Napisane korištene tehnologije i alati	Rački	10.1.2021.
1.7	Dijagrami klasa ažurirani i dodani	Milde	12.1.2021.
1.8	Testiranje komponenti	Ilinović	13.1.2021.
1.9	Testiranje sustava	Bakšić	13.1.2021.

2. Opis projektnog zadatka

Cilj ovog projekta je razvoj programske potpore za stvaranje web aplikacije *"Pomozi mi"* koja svojim korisnicima omogućuje potraživanje nečije pomoći, ali i pružanje vlastite pomoći. Primjerice, tako bi korisnik mogao otići po mlijeko za bolesnu susjedu koja je to zatražila, a ujedno nekoga iz IT sektora zamoliti da mu pomogne namjestiti postavke pisača.

Pretragom tržišta trenutno dostupnih aplikacija, nismo pronašli aplikaciju koja nudi slična rješenja. Većina pronađenih aplikacija nudi vrlo specifičan način pomoći, a aplikacije su isključivo za Android i iOS mobilne uređaje *(Be my eyes, Kindly, uCiC, Mayo)*. Najsličnija aplikacija našem rješenju je *Mayo (u razvoju, slika 2.1)*, ali i tu su vidljive velike razlike kao što su: registracija u spomenutu aplikaciju nije potrebna (anonimnost) te dostupnost samo za mobilne uređaje.



Slika 2.1: Primjer konkurentne aplikacije

Zbog prirode naše aplikacije, samo će korisnici s napravljenim računom moći pregledati zahtjeve za pomoć i objaviti svoje.

Da bi korisnik napravio svoj korisnički račun, bit će potrebni:

- Korisničko ime
- E-mail adresa
- Zaporka
- Ime
- Prezime
- Preferirana lokacija
- Kontakt broj mobitela

Kada korisnik stvori svoj račun, omogućena mu je prijava u aplikaciju unosom svojeg korisničkog imena i zaporke.

Prijavljenom korisniku sada se prikazuju zahtjevi za pomoć na početnom zaslonu, a pomoću intuitivnog korisničkog sučelja može postaviti svoj zahtjev za koji potražuje pomoć.

Opišimo sada kako će tipični korisnik navigirati našom aplikacijom. Kada korisnik želi pomoći, odabire zahtjev s liste svih aktivnih zahtjeva koji se nalaze unutar jednog kilometra od lokacije uređaja. Lista se može proširiti na veće geografsko područje. Korisnik koji će provesti odabrani zahtjev se smatra **izvršiteljem zahtjeva**. Valja napomenuti kako je svaka lista sortirana po određenom redoslijedu i omogućeno je filtriranje zahtjeva prema kategorijama. Primjerice, moguće je gledati virtualne ili zahtjeve s lokacijom. Kada korisnik traži pomoć, govorimo o ulozi **autora zahtjeva**. Prilikom zadavanja, autor unosi opis i kontakt podatke poput broja mobitela, adrese (opcionalno) i datum i/ili vrijeme do kada se zahtjev treba izvršiti (opcionalno). Budući da se više mogućih izvršitelja može javiti na jedan zahtjev, odabir konačnog vrši se metodom trostrukog rukovanja. To ćemo najbolje razjasniti primjerom:

- *Teta Marica želi da joj netko donese 3kg krumpira (autor zahtjeva).*
- *Ante i Marko vide da teta Marica treba pomoć i jave se na zahtjev (mogući izvršitelji).*
- *Marica vidi da su se obojica javila, ali je Ante bolji izvršitelj i njega odabere (Ante je izvršitelj).*

Kada je izvršitelj odabran, razmjenjuju se kontakt informacije i kreće proces izvršenja koji ostavljamo gore navedenim dionicima.

Postavlja se pitanje kako je teta Marica znala da je Ante "bolji" izvršitelj? Zahvaljujući sustavu međusobnog ocjenjivanja korisnika! Po izvršenju zahtjeva autor

označava da je zahtjev izvršen nakon čega se korisnici međusobno ocjenjuju ocjenama od 1-5 te opcionalno upisuju komentare. Kako novi korisnici ne bi ostali zakinuti za povjerenje budućih autora zahtjeva, ocjenjivanje bilo kojeg korisnika aplikacije moguće je u bilo kojem trenutku (ne samo nakon izvršenja zadatka) i ocjene se vide u detaljima profila, kao i komentari. Također, moguće je vidjeti i "lance povjerenja": da je korisnik kojeg ste vi visoko ocijenili ocijenio korisnika čiji profil gledate.

Kako bi se čovjek mogao prisjetiti svoje i tuđe humanosti, svaki korisnik može vidjeti listu zahtjeva koje je zadao i izvršio. Sustav omogućuje i dodatne izvještaje/preglede, posebno one koji omogućuju prikaz najboljeg pomagača u tekućoj godini.

Da bismo zajednicu učinili sigurnijom, uveli smo ulogu **administratora**. Administratori se brinu oko sadržaja koji se objavljuje. Imaju ovlasti brisanja zahtjeva koji se smatra opasnim, nemogućim, lažnim ili neetičkim te privremenog i trajnog blokiranja korisnika aplikacije. Administratori se dodjeljuju prema geografskim lokacijama.

Na kraju ovog opisa treba spomenuti nekoliko posebnosti. Ukoliko korisnik tijekom zadavanja zahtjeva ne želi unijeti lokaciju, taj zahtjev će se prikazivati svim korisnicima neovisno o njihovoj lokaciji. Sjetimo se čovjeka koji je tražio pomoć oko namještanja postavki pisača, taj zahtjev je razumno označiti kao virtualni. Također, kako naša spomenuta teta Marica ne bi morala uvijek unositi adresu kada stvara novi zahtjev, adresa zahtjeva može se povući iz adrese s kojom je registrirana i tako Marici uštedjeti nekoliko sekundi svakim zahtjevom.

3. Specifikacija programske potpore

3.1 Funkcionalni zahtjevi

Dionici:

1. Korisnik web aplikacije
2. Administrator
3. Razvojni tim

Aktori i njihovi funkcionalni zahtjevi:

1. Neregistrirani korisnik može:
   (a) napraviti korisnički račun
2. Registrirani korisnik može:
   (a) prijaviti se u sustav
   (b) zadati (objaviti) novi zahtjev
      i. odabrati lokaciju zahtjeva
   (c) pregledati vlastiti profil
      i. pregledati javljanja na tuđe zahtjeve
      ii. upravljati korisničkim računom
      iii. obrisati vlastiti korisnički račun
   (d) pregledati vlastite zahtjeve
      i. upravljati vlastitim zahtjevima
      ii. pregledati potencijalne izvršitelje
         A. prihvatiti izvršitelja
         B. odbiti izvršitelja
   (e) pregledavati listu aktivnih zahtjeva
      i. filtrirati zahtjeve
      ii. promijeniti lokaciju izvršavanja
      iii. javiti se na zahtjev

(f) pregledati profil drugog korisnika
(g) ocijeniti drugog korisnika
(h) izvršiti zahtjev
(i) pregledati statistiku
(j) odjaviti se iz sustava

3. Administrator

(a) izvršavati sve mogućnosti kao registrirani korisnik
(b) dodati novog administratora
(c) administrirati zahtjeve
(d) administrirati korisnika

3.1.1 Obrasci uporabe

Opis obrazaca uporabe

UC1 - Registracija

- **Glavni sudionik:** Javni korisnik
- **Cilj:** Stvoriti korisnički račun u aplikaciji
- **Sudionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
  1. Korisnik unosi potrebne podatke
  2. Potvrda unesenih podataka
  3. Upis podataka u bazu
  4. Nakon uspješne registracije korisnik se preusmjerava na stranicu zahtjeva
- **Opis mogućih odstupanja:**

  1.a Korisnik unosi već zauzeto korisničko ime i/ili e-mail ili nisu popunjena sva obavezna polja
    1. Prikaz odgovarajuće poruke
    2. Omogućavanje ponovnog unosa neodgovarajućih podataka

  4.a Mogućnost odustajanja od registracije

UC2 - Prijava u sustav

- **Glavni sudionik:** Korisnik
- **Cilj:** Prijava korisnika u sustav
- **Sudionici:** Baza podataka
- **Preduvjet:** Javni korisnik ima izrađen korisnički račun u sustavu
- **Opis osnovnog tijeka:**
  1. Javni korisnik odabire opciju prijave
  2. Javni korisnik upisuje korisničko ime i lozinku
  3. Korisnik potvrđuje unos
- **Opis mogućih odstupanja:**

  2.a Unos pogrešnog korisničkog imena i/ili lozinke
    1. Javnom korisniku ispisuje se poruka o pogrešci lozinke ili korisničkog imena

UC3.1 - Pregled potencijalnih izvršitelja

- **Glavni sudionik:** Korisnik autor
- **Cilj:** Pregledati listu korisnika koji su se javili kao potencijalni izvršitelji
- **Sudionici:**-
- **Preduvjet:** Korisnik pregledava zahtjev kojem je autor
- **Opis osnovnog tijeka:**
    1. Korisnik u prikazu zahtjeva bira opciju za prikaz potencijalnih izvršitelja
    2. Prikaz potencijalnih izvršitelja s opcijama za prihvaćanje i odbijanja

UC3.1.1 - Odbijanje javljanja na zahtjev

- **Glavni sudionik:** Korisnik autor
- **Cilj:** Odbijanje pojedinog ili više potencijalnih izvršitelja
- **Sudionici:** Korisnik izvršitelj
- **Preduvjet:** Postoji barem jedan potencijalni izvršitelj za zahtjev
- **Opis osnovnog tijeka:**
    1. Prikaz liste potencijalnih izvršitelja
    2. Korisnik odbija pojedino javljanje na zahtjev
    3. Odbijenom korisniku šalje se obavijest o odbijanju

UC3.1.2 - Prihvaćanje javljanja na zahtjev

- **Glavni sudionik:** Korisnik autor
- **Cilj:** Prihvaćanje javljanja na zahtjev za pomoć
- **Sudionici:** Korisnik izvršitelj
- **Preduvjet:** Postoji barem jedan potencijalni izvršitelj za zahtjev, Zahtjev je aktivan
- **Opis osnovnog tijeka:**
    1. Prikaz liste potencijalnih izvršitelja
    2. Korisnik prihvaća pojedino javljanje na zahtjev
    3. Slanje obavijesti prihvaćenom korisniku
    4. Automatsko odbijanje ostalih potencijalnih izvršitelja
    5. Slanje obavijesti odbijenim korisnicima
    6. Automatsko dodavanje prihvaćenog korisnika kao izvršitelja zahtjeva

UC3.2 - Javljanje na zahtjev

- **Glavni sudionik:** Korisnik
- **Cilj:** Inicijalizirati komunikaciju s korisnikom autorom
- **Sudionici:** Korisnik autor zahtjeva, Baza podataka

- **Preduvjet:** Oglas je aktivan, korisnik nije autor zahtjeva kojeg pregledava
- **Opis osnovnog tijeka:**
  1. Korisnik odabire zahtjev za pregled
  2. Korisnik odabire opciju javljanja na zahtjev
  3. Korisnik se uvodi u bazu podataka kao potencijalni izvršitelj za odabrani zahtjev
  4. Korisniku autoru dolazi obavijest o novom potencijalnom izvršitelju
- **Opis mogućih odstupanja:**
  1.a Korisnik autor može biti blokiran
    1. Zahtjevi blokiranih korisnika ne prikazuju se na glavnoj stranici zahtjeva
    2. Zahtjevi blokiranih korisnika vidljivi na njegovom profilu ne mogu biti odabrani za izvršavanje

UC3.3 - Brisanje/blokiranje zahtjeva

- **Glavni sudionik:** Korisnik autor
- **Cilj:** Mogućnost blokiranja ili brisanja zahtjeva
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik pregledava vlastiti zahtjev
- **Opis osnovnog tijeka:**
  1. Korisnik odabire zahtjev za pregled
  2. Korisnik odabire brisanje/blokiranje zahtjeva
  3. Unos izmjena u bazu podataka
- **Opis mogućih odstupanja:**
  2.a Pokušaj brisanja/blokiranja zahtjeva za koje postoje aktivni izvršitelji
    1. Zahtjevi za koje postoje aktivni izvršitelji mogu biti samo blokirani
    2. Svim potencijalnim ili aktivnim izvršiteljima se šalje obavijest o blokiranju zahtjeva

UC4 - Zadavanje novog zahtjeva

- **Glavni sudionik:** Korisnik
- **Cilj:** Unijeti i opisati svoj zahtjev za pomoć
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen
- **Opis osnovnog tijeka:**
  1. Korisnik otvara komponentu za unos zahtjeva

2. Unos opisa zahtjeva
3. Unos vremena isteka
4. Potvrda zahtjeva
5. Unos zahtjeva u bazu podataka

- **Opis mogućih odstupanja:**

2.a Unos zahtjeva sa praznim opisom

1. Prikaz poruke o minimalnoj duljini opisa od dva znaka

UC4.1 - Odabir lokacije zahtjeva

- **Glavni sudionik:** Korisnik
- **Cilj:** Opcionalan odabir lokacije zahtjeva
- **Sudionici:** Baza podataka
- **Preduvjet:** U tijeku je zadavanje novog zahtjeva
- **Opis osnovnog tijeka:**
  1. Korisnik se odlučuje za postavljanje lokacije
  2. Odabir korištenja adrese korisnika ili unosa na karti
  3. Otvaranje karte za unos lokacije
  4. Potvrda lokacije
  5. Nastavak zadavanja zahtjeva
- **Opis mogućih odstupanja:**

1.a Korisnik se odlučuje ne unositi lokaciju

1. Zahtjevi bez lokacije vode se kao virtualni i prikazuju se svim korisnicima
2. Virtualni zahtjevi polaze od pretpostavke da je lokacija irelevantna za uspješno izvršavanje

3.a Unos prazne lokacije

1. Ukoliko se korisnik odlučio za unos lokacije putem karte, a nije to učinio kao adresa zahtjeva se vodi nulti meridijan i ekvator.

UC5 - Pregled profila

- **Glavni sudionik:** Korisnik
- **Cilj:** Pregled profila korisnika
- **Sudionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
  1. Odabir korisnika za prikaz

2. Prikaz osnovnih podataka o korisniku, njegovih zahtjeva i zahtjeva koje je on izvršio
3. Korisnik može pregledavati zahtjeve profila
4. Korisnik može pregledati lanac povjerenja, komentare i ocjenu profila korisnika

UC5.1 - Brisanje korisničkog računa

- **Glavni sudionik:** Korisnik
- **Cilj:** Brisanje korisničkog računa iz sustava
- **Sudionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
  1. Korisnik odabire opciju brisanja računa
  2. Unos brisanja u bazu podataka

UC5.2 - Upravljanje korisničkim podacima

- **Glavni sudionik:** Korisnik
- **Cilj:** Izmjena korisničkih podataka
- **Sudionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
  1. Korisnik odabire opciju izmjene korisničkih podataka
  2. Korisnik unosi nove podatke
  3. Korisnik potvrđuje novi unos
- **Opis mogućih odstupanja:**

  2.a Unos podataka u krivom formatu ili neispunjenje obaveznih polja
  1. Prikaz odgovarajuće poruke
  2. Ponovna mogućnost unosa podataka

UC6 - Promjena lokacije izvršenja

- **Glavni sudionik:** Korisnik
- **Cilj:** Izmjena korisnikove lokacije
- **Sudionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
  1. Korisnik odabire izmjenu lokacije

2. Korisnik unosi novu lokaciju u tekstualnom obliku ili odabirom na karti
3. Nova lokacija pohranjuje se u bazu
4. Prikaz zahtjeva u odnosu na novu lokaciju

- **Opis mogućih odstupanja:**

3.a Unos nevaljane lokacije

1. Unos lokacije u bazu se stornira
2. Korisniku se prikazuje odgovarajuća poruka

UC7 - Izvršavanje zahtjeva

- **Glavni sudionik:** Korisnik
- **Cilj:** Označavanje zahtjeva izvršenim
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je postavljen kao autor zahtjeva
- **Opis osnovnog tijeka:**
  1. Korisnik odabire zahtjev za pregled
  2. Korisnik označuje zahtjev izvršenim
  3. Korisniku izvršitelju šalje se obavijest o izvršenju
  4. Inicira se ocjenjivanje korisnika
- **Opis mogućih odstupanja:**

1.a Zahtjev je istekao prije nego što je korisnik potvrdio izvršenje

1. Zahtjevi za koje je korisnik odabran kao izvršitelj prikazuju se unatoč istjecanju i mogu se odabrati za izvršavanje

1.b Zahtjev za koji je korisnik odabran kao izvršitelj je blokiran

1. Blokirani zahtjevi se ne mogu izvršavati i ne prikazuju se među korisnikovim zahtjevima za izvršavanje

UC8 - Ocjenjivanje korisnika

- **Glavni sudionik:** Korisnik
- **Cilj:** Ocjena korisnika i/ili ocjena izvršenja zahtjeva uz popratni komentar
- **Sudionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
  1. Korisnik inicira ocjenjivanje na profilu drugog korisnika ili se ocjenjivanje pokreće nakon izvršenja zahtjeva
  2. Korisnik izabire ocjenu od 1 do 5
  3. Korisnik opcionalno unosi komentar

4. Korisnik potvrđuje svoj odabir
5. Ocjena i komentar se unose u bazu podataka

- **Opis mogućih odstupanja:**

2.a Ocjena se može, ali ne mora odnositi na izvršavanje specifičnog zahtjeva

1. Ukoliko se ocjena odnosi na izvršavanje zahtjeva, u bazu se upisuje o kojem se zahtjevu radi

UC9 - Pregled statistike

- **Glavni sudionik:** Korisnik
- **Cilj:** Prikaz statistika o korisnicima
- **Sudionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
  1. Korisnik odabire opciju prikaza statistike
  2. Prikaz statistike u aplikaciji
  3. Povratak na prethodnu stranicu

UC10 - Administriranje zahtjeva

- **Glavni sudionik:** Administrator
- **Cilj:** Brisanje neprihvatljivih zahtjeva
- **Sudionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
  1. Administrator odabire opciju brisanja zahtjeva
  2. Administrator potvrđuje odabir
  3. Autoru zahtjeva dolazi obavijest o brisanju zahtjeva
  4. Potencijalnim izvršiteljima se šalje obavijest o brisanju zahtjeva
- **Opis mogućih odstupanja:**

2.a Javljanje na zahtjev je već prihvaćeno i izvršitelj je postavljen

1. Izvršitelj dobiva obavijest o brisanju zahtjeva

UC11 - Administriranje korisnika

- **Glavni sudionik:** Administrator
- **Cilj:** Upravljanje korisnicima
- **Sudionici:** Korisnici, Baza podataka
- **Preduvjet:** -

- **Opis osnovnog tijeka:**
  1. Na profilu korisnika administrator odabire opciju administriranja korisnika
  2. Administrator bira opciju privremenog ili trajnog blokiranja korisnika
  3. Administrator potvrđuje odabir
  4. Unos blokiranja u bazu podataka
  5. Svi aktivni zahtjevi blokiranog korisnika za koje nije još odabran izvršitelj se brišu
- **Opis mogućih odstupanja:**
  - 1.a Korisnik je već privremeno blokiran
    1. Administrator može odblokirati privremeno blokiranog korisnika
  - 1.b Korisnik je već trajno blokiran
    1. Administrator ne može odblokirati trajno blokiranog korisnika

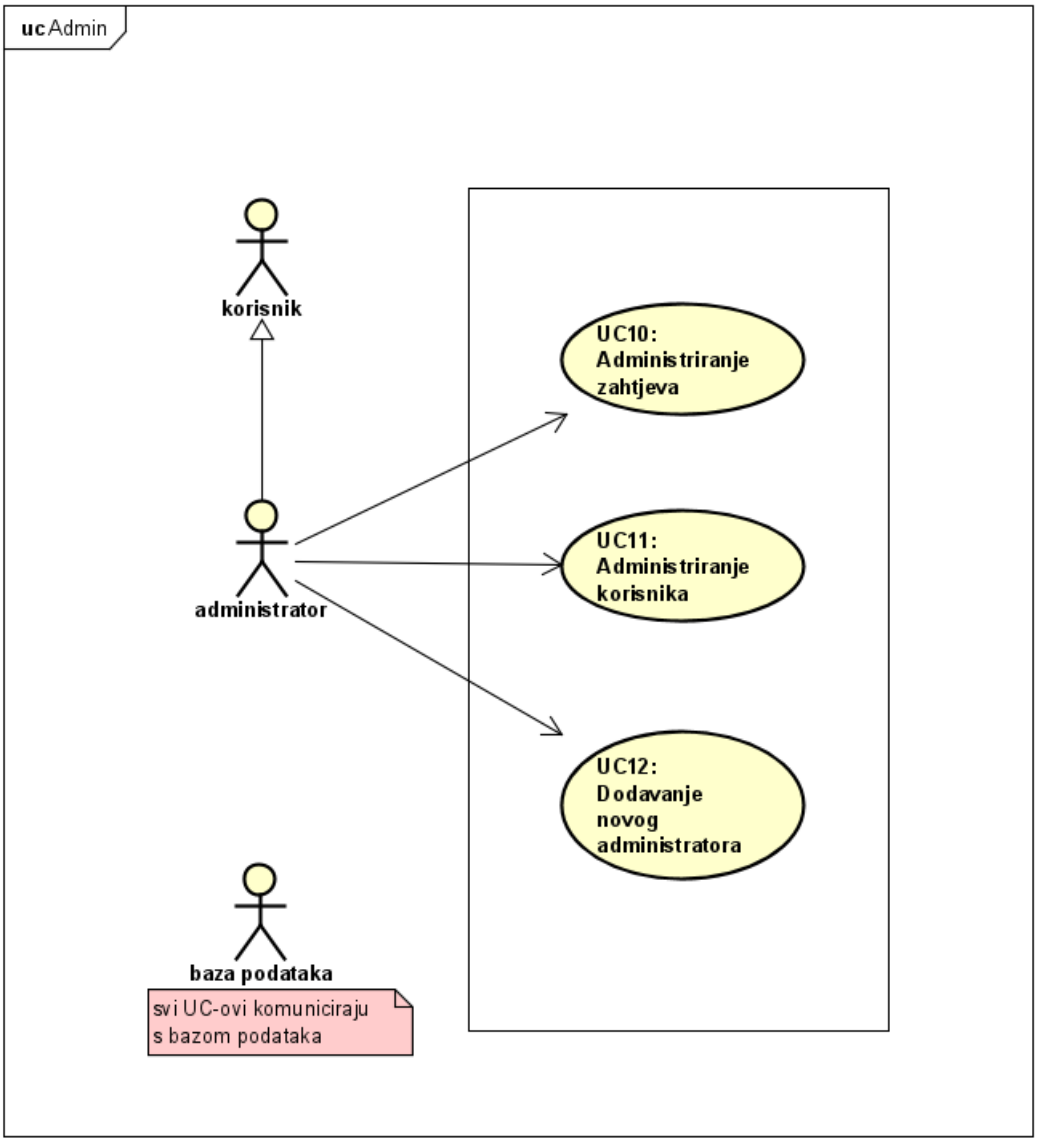
UC12 - Dodavanje novog administratora

- **Glavni sudionik:** Administrator
- **Cilj:** Postaviti nekog korisnika kao administratora
- **Sudionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
  1. Administrator na profilu korisnika odabire opciju postavljanja administratorskih ovlasti
  2. Administrator potvrđuje odabir
- **Opis mogućih odstupanja:**
  - 1.a Pregledavanje profila korisnika koji već ima dodijeljene administratorske ovlasti
    1. Administratoru se ne omogućava ponovo postavljanje ovlasti

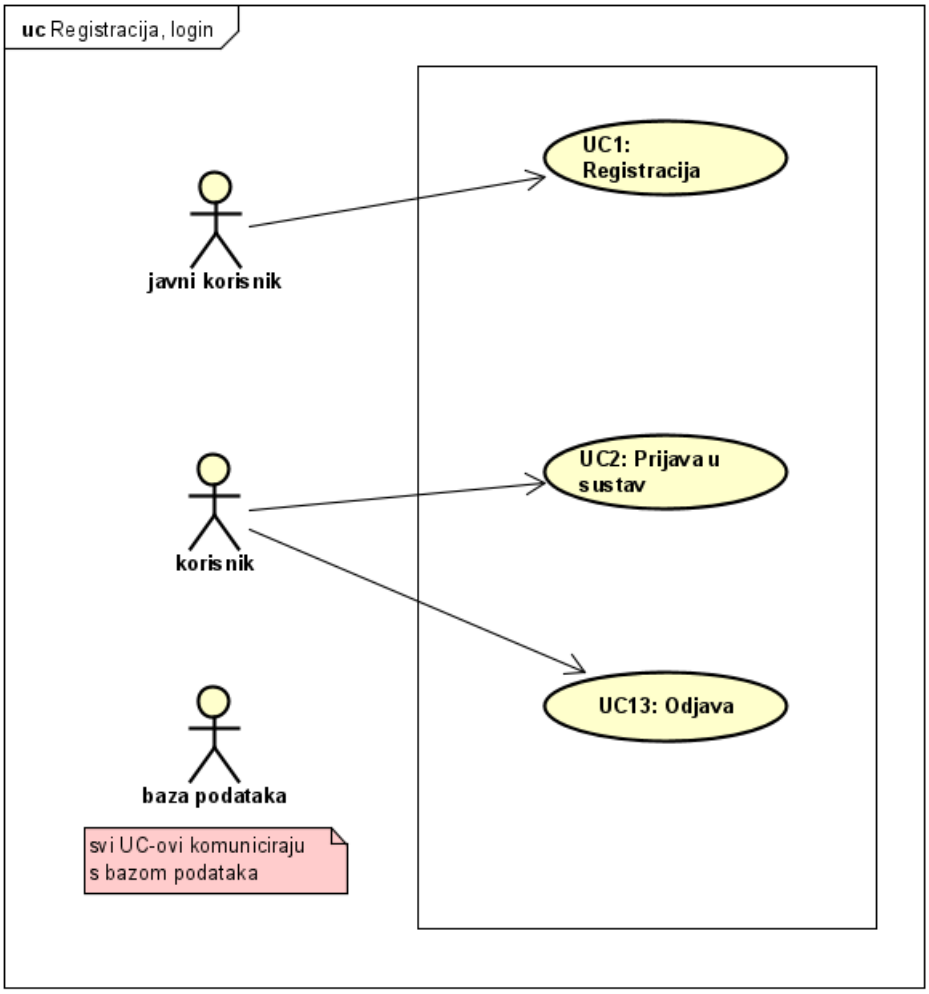
UC13 - Odjava

- **Glavni sudionik:** Korisnik
- **Cilj:** Odjava iz sustava
- **Sudionici:** -
- **Preduvjet:** Korisnik je prijavljen u sustav
- **Opis osnovnog tijeka:**
  1. Korisnik odabire opciju odjave
  2. Korisnik se preusmjerava na stranicu za prijavu

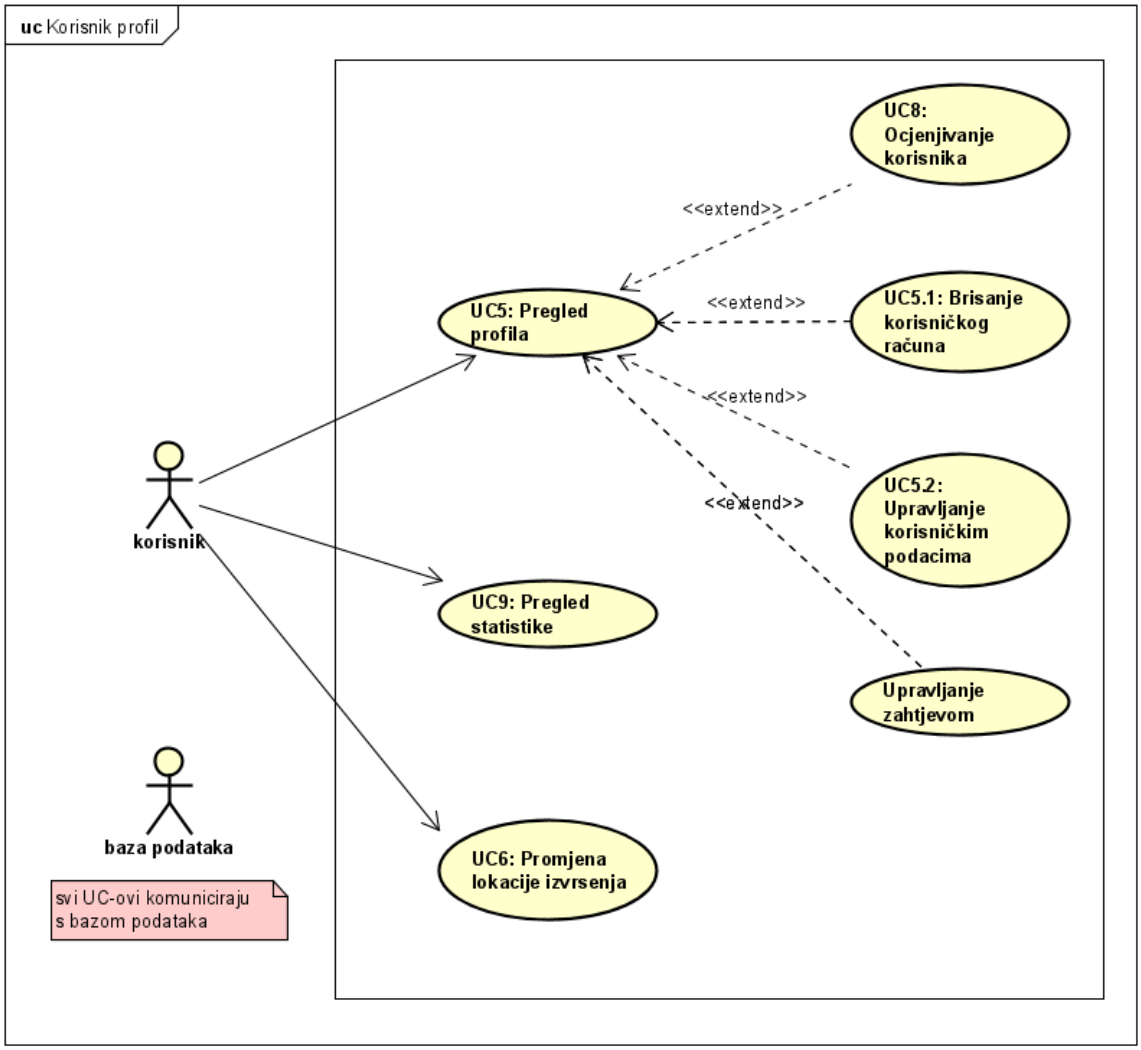
Dijagrami obrazaca uporabe



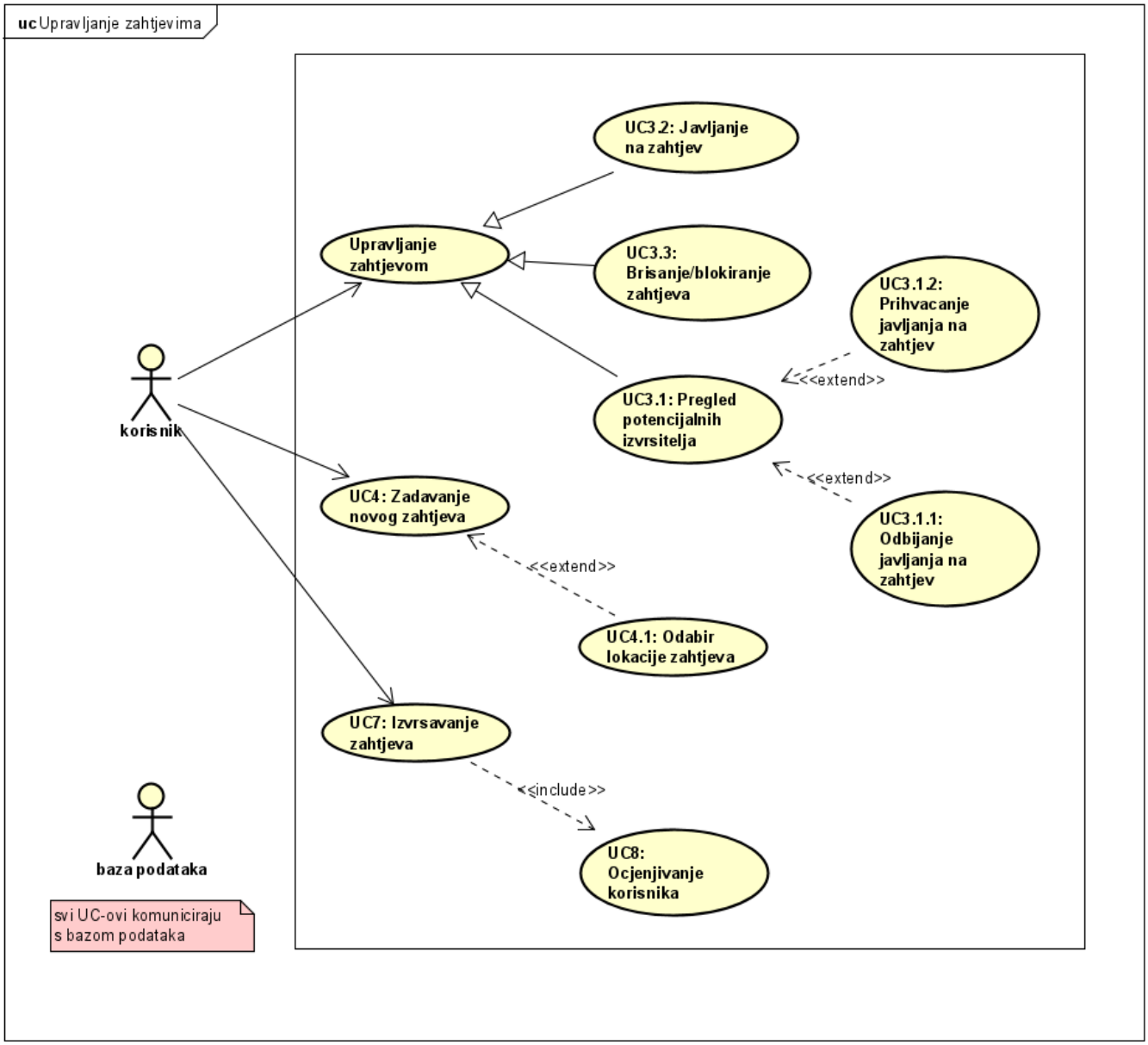
Slika 3.1: Admin



Slika 3.2: Registracija, login



Slika 3.3: Korisnik i profil

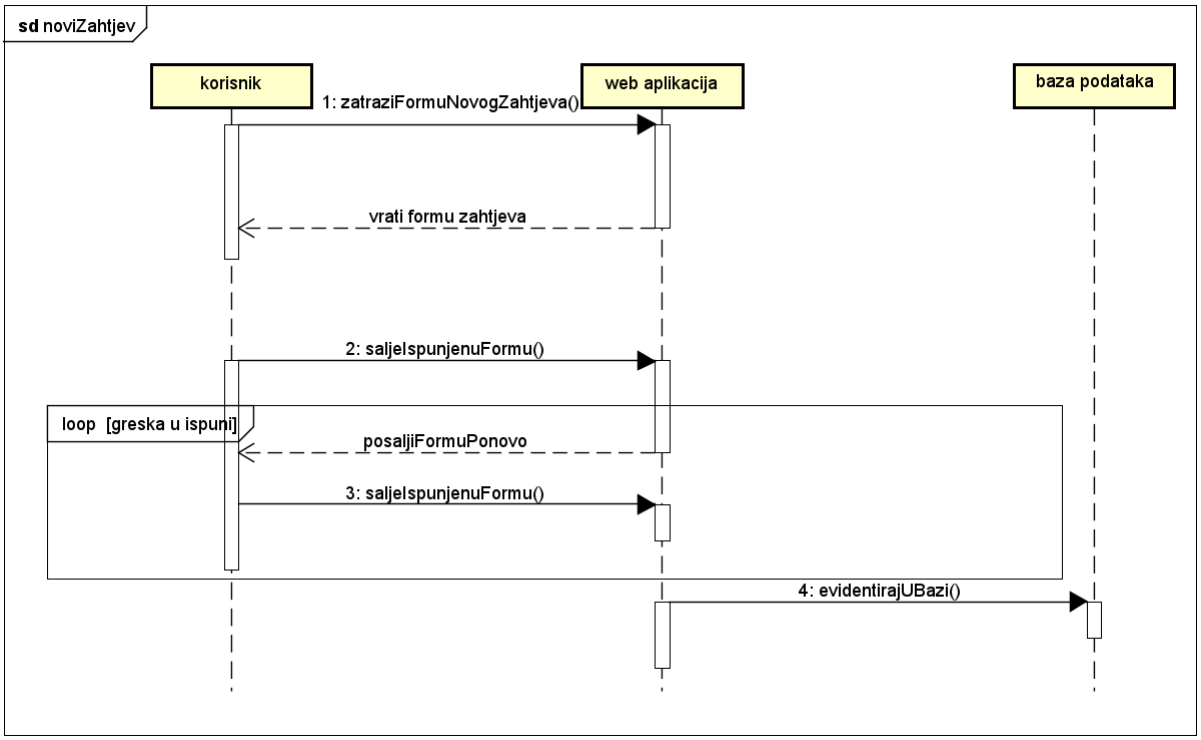


Slika 3.4: Upravljanje zahtjevima

3.1.2 Sekvencijski dijagrami

Novi zahtjev

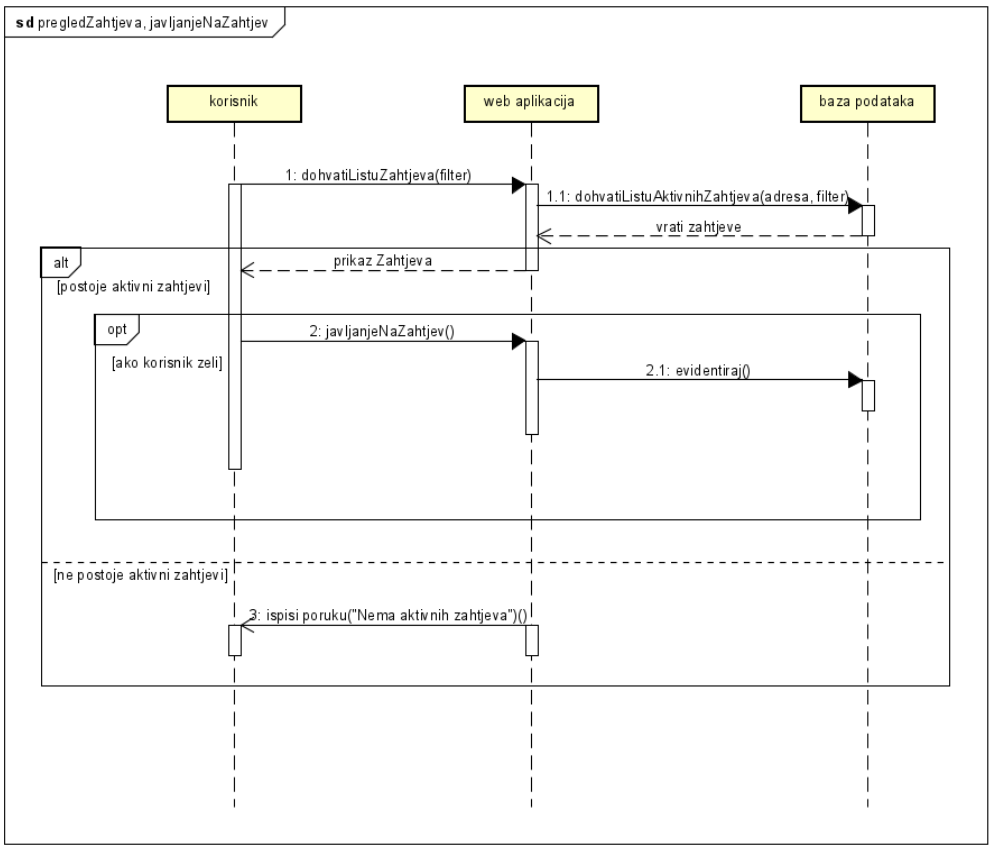
Pri postavljanju novog zahtjeva, korisnik odabire opciju za dodavanje novog zahtjeva te mu poslužitelj šalje formu novog zahtjeva. Nakon toga, korisnik popunjava formu te ga sustav traži da popravi unos dok god su neka polja krivo popunjena. Nakon uspješne popune svih polja u formi, događa se evidentiranje zahtjeva u bazi, te se nakon toga zahtjev može prikazivati drugim korisnicima aplikacije.



Slika 3.5: Novi zahtjev

Pregled zahtjeva, javljanje na zahtjev

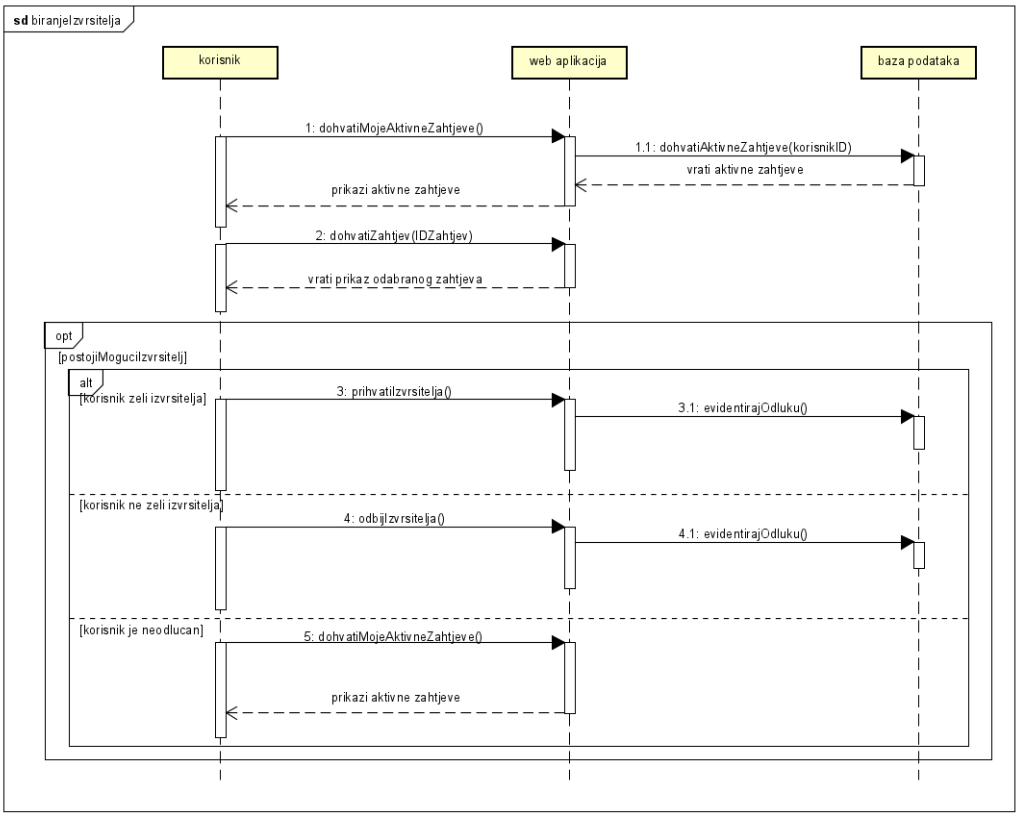
Korisnik pri uobičajenom korištenju aplikacije može pregledavati listu dostupnih zahtjeva. Korisnik tada odabire opciju prikaza zahtjeva, a poslužitelj od baze podataka traži odgovarajuće zahtjeve obzirom na kriterije filtriranja zahtjeva za prikaz. U slučaju da postoje aktivni zahtjevi sa odgovarajućim uvjetima korisnik ima mogućnost javiti se na zahtjev tako što odabere tu opciju u aplikaciji, a poslužitelj to evidentira u bazi. Nakon što se javio na zahtjev, autor zahtjeva može vidjeti kontakte potencijalnih izvršitelja i kontaktirati ih. U slučaju da nema aktivnih zahtjeva korisniku se u aplikaciji ispisuje odgovarajuća poruka te nema daljnjih mogućnosti za javljanje na zahtjev.



Slika 3.6: Pregled zahtjeva, javljanje na zahtjev

Biranje izvršitelja

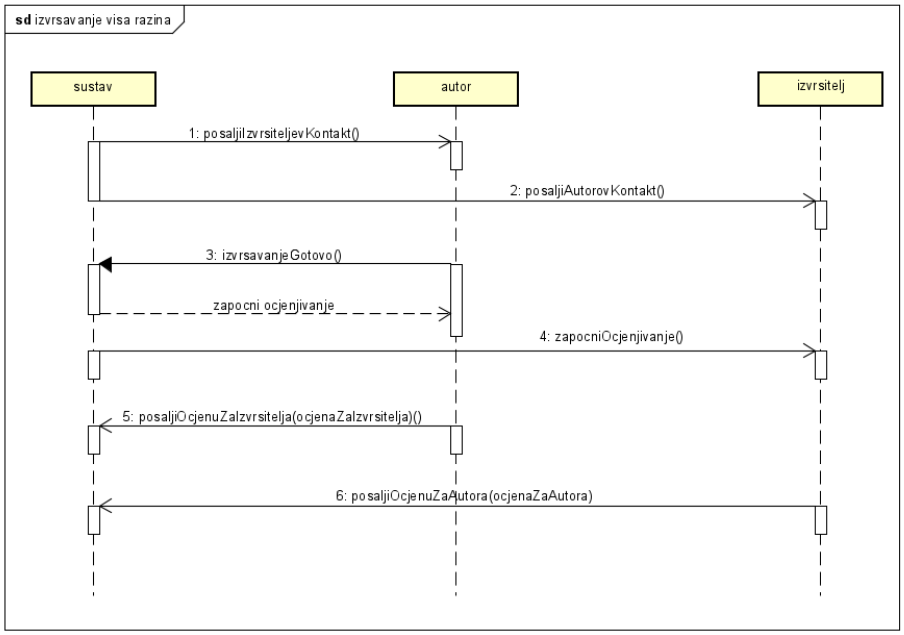
Isprva korisnik šalje zahtjev za prikazivanjem njegovih aktivnih zahtjeva, poslužitelj ih pronalazi u bazi i prosljeđuje korisniku. Ako korisnik želi vidjeti neki pojedini zahtjev detaljnije, odabire taj zahtjev te mu se u slučaju da postoje ljudi koji su se javili na zahtjev nude opcije odbijanja i prihvaćanja pojedinog izvršitelja te može isto tako odgoditi odluko oko biranja izvršitelja i vratiti se na prikaz svih njegovih aktivnih zahtjeva.



Slika 3.7: Biranje izvršitelja

Izvršavanje viša razina

Izvršavanje kreće tako da sustav šalje kontakte izvršitelju i autoru. Nakon izvršavanja autor označava da je zahtjev izvršen, čime se omogućava ocjenjivanje. Ocjenjivanje uključuje odabir ocjene od 1 do 5 te opcionalan unos komentara.



Slika 3.8: Izvršavanje viša razina

3.2 Ostali zahtjevi

- Aplikacija treba biti izvedena kao web aplikacija prilagođena mobilnom uređaju.
- Sustav mora podržavati rad više korisnika u stvarnom vremenu.
- Procesiranje bilo kakve korisničke interakcije sa sustavom ne bi trebalo trajati duže od par sekundi.
- Administratori su dodijeljeni po geografskim lokacijama.
- Sustav mora podržavati hrvatske dijakritičke znakove.
- Informacije o zahtjevima moraju biti redovno ažurirane.
- Korisničko sučelje treba biti jednostavno za korištenje.

4. Arhitektura i dizajn sustava

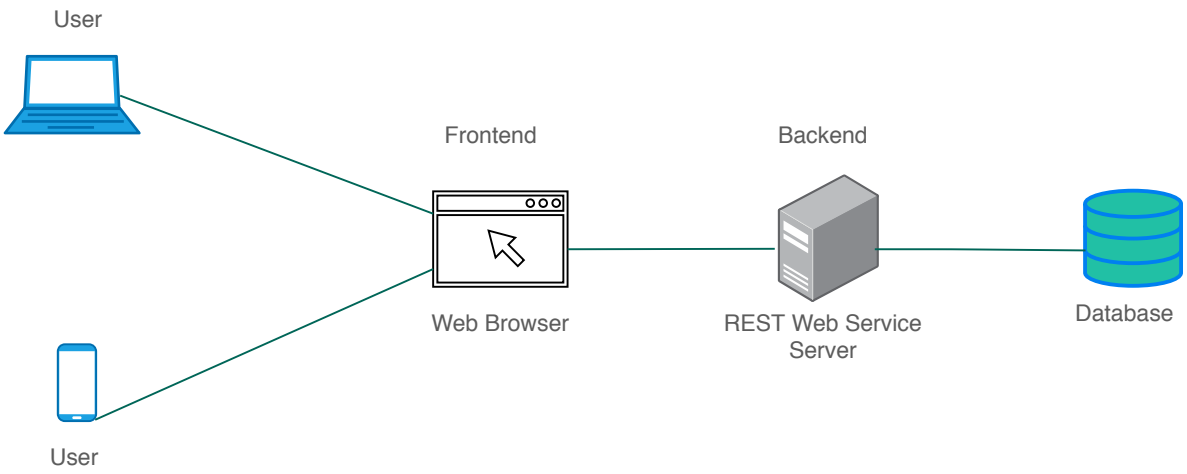
Odabir odgovarajuće arhitekture je važna odluka koja utječe na cjelokupnu funkcionalnost sustava. Budući da je cilj aplikacije široka dostupnost, odabrali smo web aplikaciju. Web aplikacija funkcionira neovisno o platformi, što oslobađa razvojni tim višeplatformnog razvoja.
Korisnik aplikaciji pristupa pomoću web preglednika. Web preglednik je program koji omogućava korisniku pregled i prikaz web aplikacije na uređaju. Kako bi se omogućila komunikacija klijenta(korisnika) s aplikacijom koristi se web poslužitelj koji koristi HTTP protokol. Aplikacija komunicira sa backend-om preko REST API-ja. Backend dohvaća sve potrebne podatke iz baze podataka(više o njoj u sljedećoj sekciji) nakon čega preko poslužitelja i preglednika prikazuje te podatke korisniku u obliku HTML dokumenta.
Aplikacija je pisana u programskom jeziku Java i programskom okruženju Spring Boot, te React-u(JavaScript biblioteka za izgradnju korisničkih sučelja), a od razvojnih okruženja odabrali smo Visual Studio Code za frontend te IntelliJ IDEA za ostatak posla.

Logika backenda izgrađena je na temelju REST konvencije, stoga je razrađena u tri sloja: Repository, Service i Controller.

- **Repository** – Sloj Repository primarno je zadužen za komunikaciju s bazom podataka. U njemu se nalaze standardne metode za dohvaćanje, spremanje i izmjenu podataka u bazi. Te metode temelje se na atributima entiteta za koji je izgrađen pojedini repository. Također, svaki repository može biti nadopunjen vlastitim metodama.

- **Service** – U Service sloju odvija se poslovna logika i manipulacija podacima koji dolaze s baze. To uključuje kreiranje klasa, promjenu tipova, omatanje i sl.

- **Controller** – prima i obrađuje zahtjeve poslužitelja. Svi se zahtjevi prosljeđuju koristeći JSON format. Shodno pojedinom zahtjevu, controller zatražuje podatke sa repositoryja, izvodi metode koje mu pruža Service. Na kraju podatke



Slika 4.1: Arhitektura sustava

u obliku objekata šalje kao odgovor web poslužitelju.

React je javascript knjižnica za izgradnju korisničkih sučelja. Temelji se na deklariranju komponenata koje se prikazuju korisniku preko web preglednika.

4.1 Baza podataka

Sve je podatke potrebno negdje spremiti kako bi se mogli dinamički dohvaćati. Za ovo nam služi baza podataka, koju također smatramo dijelom MVC obrasca. Naša aplikacija u pozadini koristi, kao relacijsku bazu podataka, PostgreSQL.
U bazi podataka nalaze se sljedeći entiteti:

- Korisnik
- Adresa
- Ocjena
- Zahtjev
- Obavijest
- Potencijalni

4.1.1 Opis tablica

Korisnik Ovaj entitet modelira jednog korisnika aplikacije.
Sadrži atribute: korisnikID, ime, prezime, e-posta, lozinka, korisnickoIme, jeAdmin, telefon, slika, status, i adresaID koji predstavlja strani ključ na entitet Adresa. Korisnik je povezan s entitetom Ocjena vezama ”prima” i ”daje” (*One-to-Many*) preko atributa korisnikID. Korisnik se također veže s entitetom Zahtjev vezama ”autorski” i ”izvršiteljski” (*One-to-Many*) preko atributa korisnikID i vezom ”potencijalni” (*Many-to-Many*) preko atributa korisnikID. Ovaj entitet se još veže uz entitet Adresa vezom *Many-to-One* preko atributa adresaID. Također je u vezi s entitetom Obavijest vezom *One-To-Many* preko atributa korisnikID.

Korisnik		
korisnikID	INT	jedinstveni identifikator svakog korisnika
ime	VARCHAR	ime korisnika
prezime	VARCHAR	prezime korisnika
e-posta	VARCHAR	e-mail adresa korisnika
lozinka	VARCHAR	hash lozinke
korisnickoIme	VARCHAR	korisnicko ime
jeAdmin	BOOLEAN	oznaka je li korisnik administrator
telefon	VARCHAR	broj mobitela korisnika
slika	BOOLEAN	oznaka je li korisnik ima sliku profila
status	VARCHAR	oznaka statusa korisničkog računa
adresaId	VARCHAR	adresa prebivališta korisnika

Adresa Ovaj entitet modelira adresu prebivališta pojedinog korisnika aplikacije. Sadrži sljedeće atribute: adresaID, opis, xKoordinata, yKoordinata. Entitet Adresa je u vezi *One-to-Many* s entitetom Zahtjev preko atributa adresaID i u vezi *One-to-Many* s entitetom Korisnik pomoću atributa adresaID.

Adresa		
adresaID	INT	jedinstveni identifikator adrese korisnika
opis	VARCHAR	Detaljniji opis adrese korisnika
xKoordinata	NUMERIC	X koordinata adrese korisnika predočena na karti
yKoordinata	NUMERIC	Y koordinata adrese korisnika predočene na karti

Ocjena Ovaj entitet predstavlja ocjenu koju jedan korisnik daje drugome. Sadrži atribute: ocjenaID, komentar, ocjena, korisnikID, zahtjevID, primakorisnikID. Ovaj entite sadrži tri strana ključa, a to su: korisnikID(predstavlja korisnika koji ocjenjuje), primakorisnikID(predstavlja korisnika kojeg se ocjenjuje) i zahtjevID(predstavlja zahtjev koji se izvršava). Ocjena je u dvije veze s entitetom Korisnik, a to su "prima" (*Many-to-One*) i "daje" (*Many-to-One*) preko atributa korisnikID. Ocjena se još veže uz Zahtjev vezom *One-to-One* preko atributa zahtjevID.

Ocjena		
ocjenaID	INT	jedinstveni identifikator svake ocjene
komentar	VARCHAR	komentar kojeg korisnik ostavlja uz ocjenu
ocjena	INT	ocjena koju korisnik dodjeljuje
korisnikID	INT	korisnik koji ocjenjuje
zahtjevID	INT	zahtjev koji se izvršava
primakorisnikID	INT	korisnik kojeg se ocjenjuje

Zahtjev Ovaj entitet predstavlja jedan zahtjev kojeg korisnik aplikacije zadaje ili izvršava. Sadrži atribute:zahtjevID, opis, datumVrPocetka, naslov, datumIsteka, status, adresaID, korisnikID, autorskikorisnikID. Kao i entitet ocjena i ovaj entitet sadrži tri strana ključa: adresaID(predstavlja adresu korisnika koji je zadao zahtjev, nije obavezno kako bi se kreirao zahtjev), korisnikID(predstavlja izvršitelja zahtjeva) i autorskikorisnikID(predstavlja samog kreatora zahtjeva). Entitet je u vezi s Ocjenom preko veze *One-to-One* pomoću atributa zahtjevID. Zahtjev se veže uz entitet Korisnik preko tri veze "autorski" (*Many-to-One*), veze "izvršiteljski" (*Many-to-One*) i veze "potencijalni" (*Many-to-Many*) svaka preko atributa korisnikID. Još se veže uz entitet Adresa *One-to-Many* preko atributa adresaID. Također je u vezi s entitetom Obavijest vezom *One-To-Many* preko atributa zahtjveID.

Zahtjev		
zahtjevID	INT	jedinstveni identifikator svakog zahtjeva
opis	VARCHAR	opis zahtjeva
datumVrPocetka	DATETIME	trenutak postavljanja zahtjeva na aplikaciju
datumIsteka	DATE	datum kada zahtjev istječe, te se ne može više izvršavati
naslov	VARCHAR	naslov zahtjeva
status	VARCHAR	status zahtjeva
adresaID	INT	adresa autora zahtjeva
korisnikID	INT	izvršitelj zahtjeva
autorskikorisnikID	INT	autor zahtjeva

Potencijalni Ovaj entitet predstavlja sve potencijalne izvršitelje jednog zahtjeva. Sadrži atribute: zahtjevID, korisnikID. Oba atributa su strani ključevi. Prvi predstavlja zahtjev kojeg korisnik želi izvršiti, drugi predstavlja korisnika koji želi izvršit zahtjev. Ovaj entitet nastaje radi *Many-to-Many* veze "potencijalni" između Zahtjeva i Korisnika.

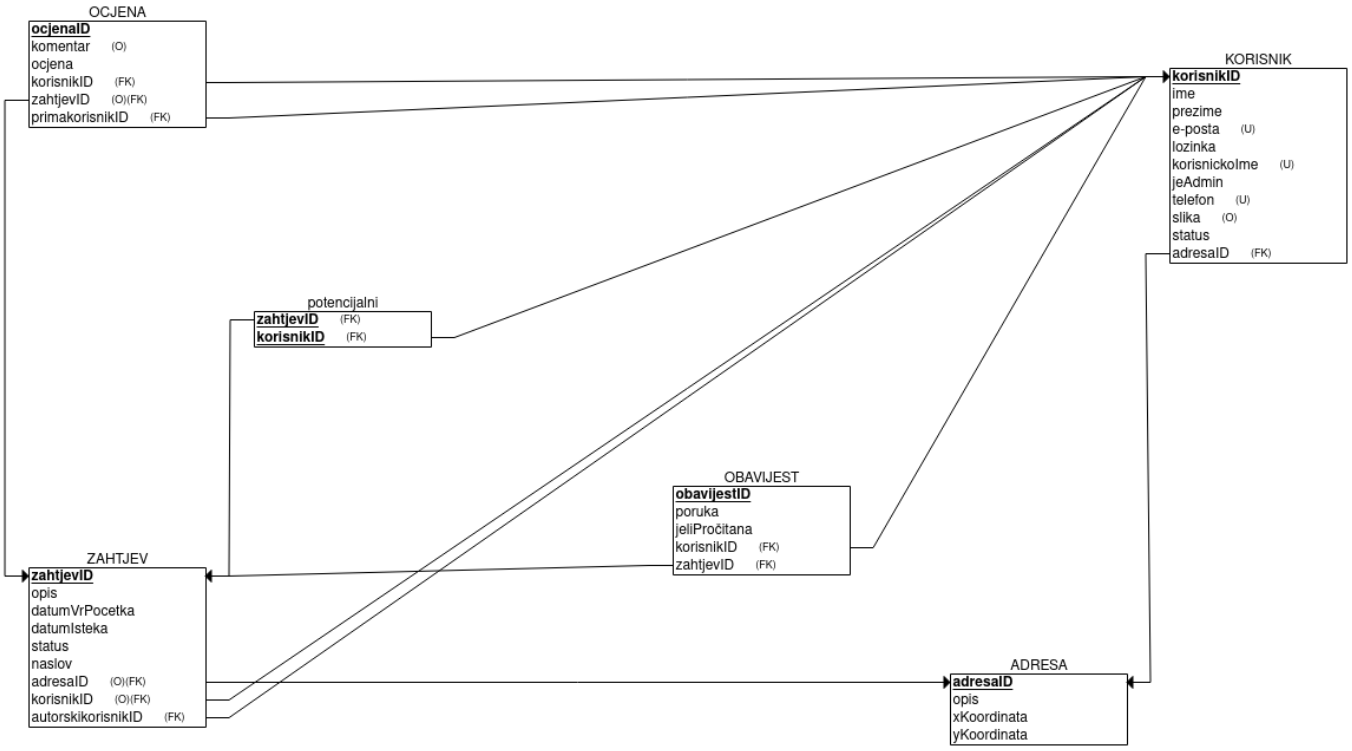
Potencijalni		
zahtjevID	INT	zahtjev kojeg korisnik želi izvršiti
korisnikID	INT	potencijalni izvršitelj zahtjeva

Obavijest Ovaj entitet predstavlja sve moguće obavijesti koje jedan korisnik može dobiti. Sadrži atribute: obavijestID, korisnikID, poruka, zahtjevID, jeliPročitana. Ovaj entite sadrži dva strana ključa: korisnikID(predstavlja korisnika kojem je obavijest namijenjena), te zahtjevID(predstvavlja zahtjev na kojeg se obavijest odnosi). Entitet je u vezi s entitetom Korisnik vezom *Many-To-One* pomoću atributa korisnikID, te s entitetom Zahtjev vezom *Many-To-One* preko atributa zahtjevID.

Obavijest		
obavijestID	INT	jedinstveni identifikator obavijesti
poruka	VARCHAR	predstavlja detaljniji opis obavijesti
jeliPročitana	BOOLEAN	označuje jeli korisnik pročitao obavijest
korisnikID	INT	korisnik kojem je obavijest namijenjena.
zahtjevID	INT	zahtjev na kojeg se obavijest odnosi

4.1.2 Dijagram baze podataka

Podcrtani elementi su ključevi, elementi koji imaju (O) nisu obavezni za unos u bazu podataka, elementi koji imaju (FK) su strani ključevi i elementi s oznakom (U) moraju biti jedinstveni.

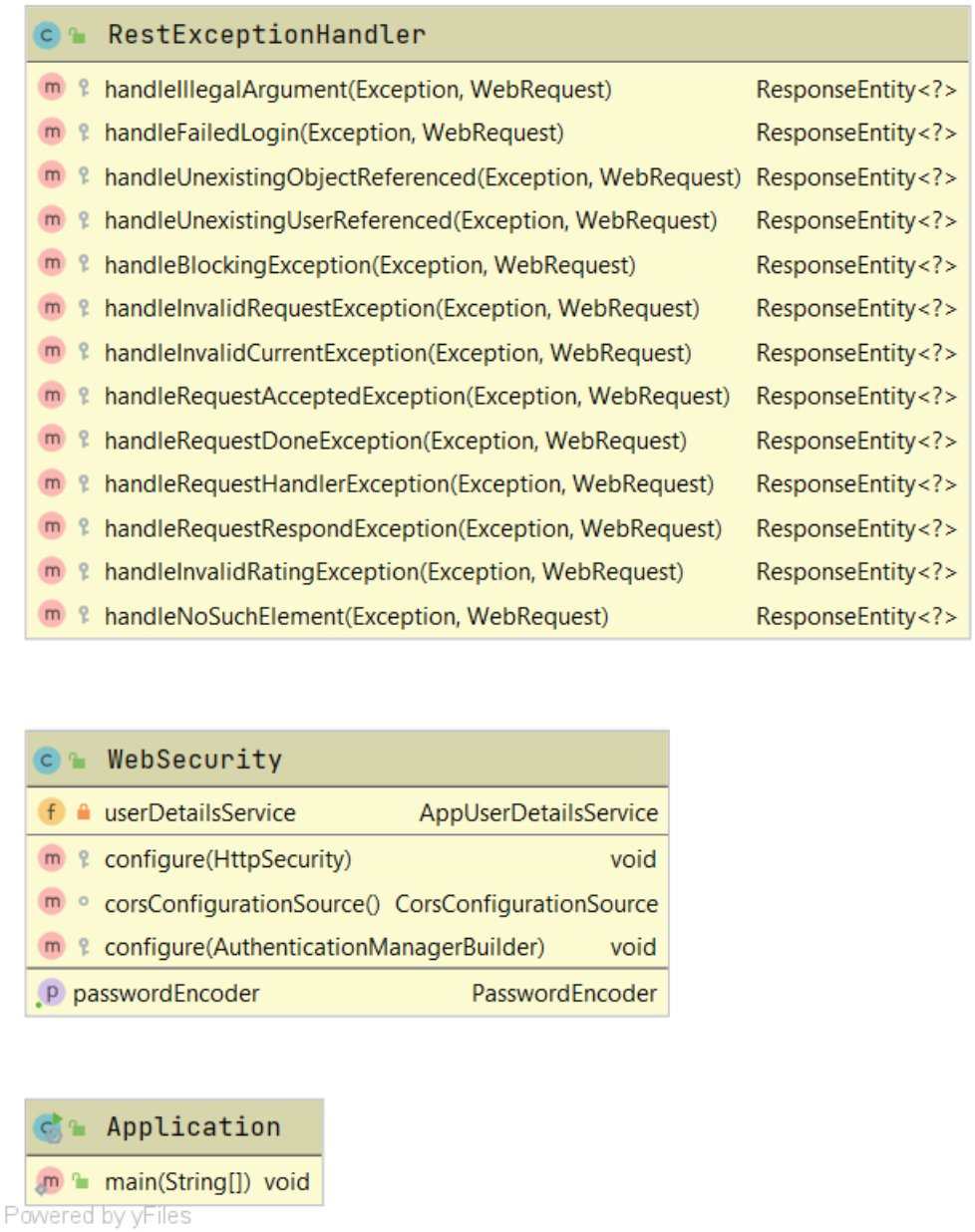


Slika 4.2: Relacijski model baze podataka

4.2 Dijagram razreda

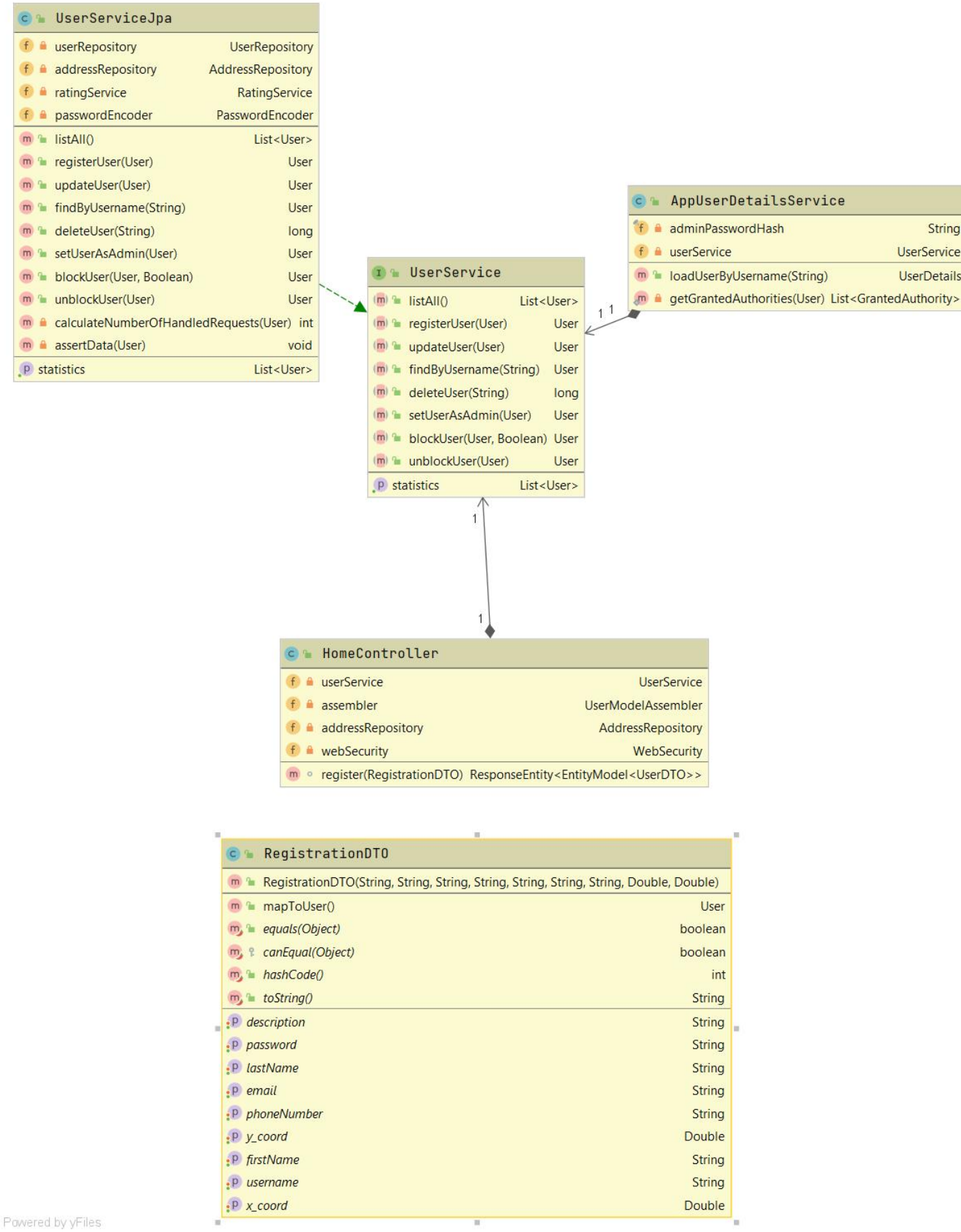
U ovom poglavlju opisana je struktura *backenda* aplikacije te opisana glavna funkcionalnost pojedinih klasa.

Na slici 4.3 prikazana je konfiguracija aplikacije *Pomozi mi*. Konfiguracija aplikacije temelji se na postavkama u klasi *WebSecurity*, kojom su definiranja dopuštenja pristupa za pojedini *path*, ponašanje pri *loginu* i *logoutu*.
Klasom RestExceptionHandler regulirano je ponašanje sustava pri pojavi iznimke. Pojava iznimki rezultira *responseom* u kojem je sadržana poruka iznimke.



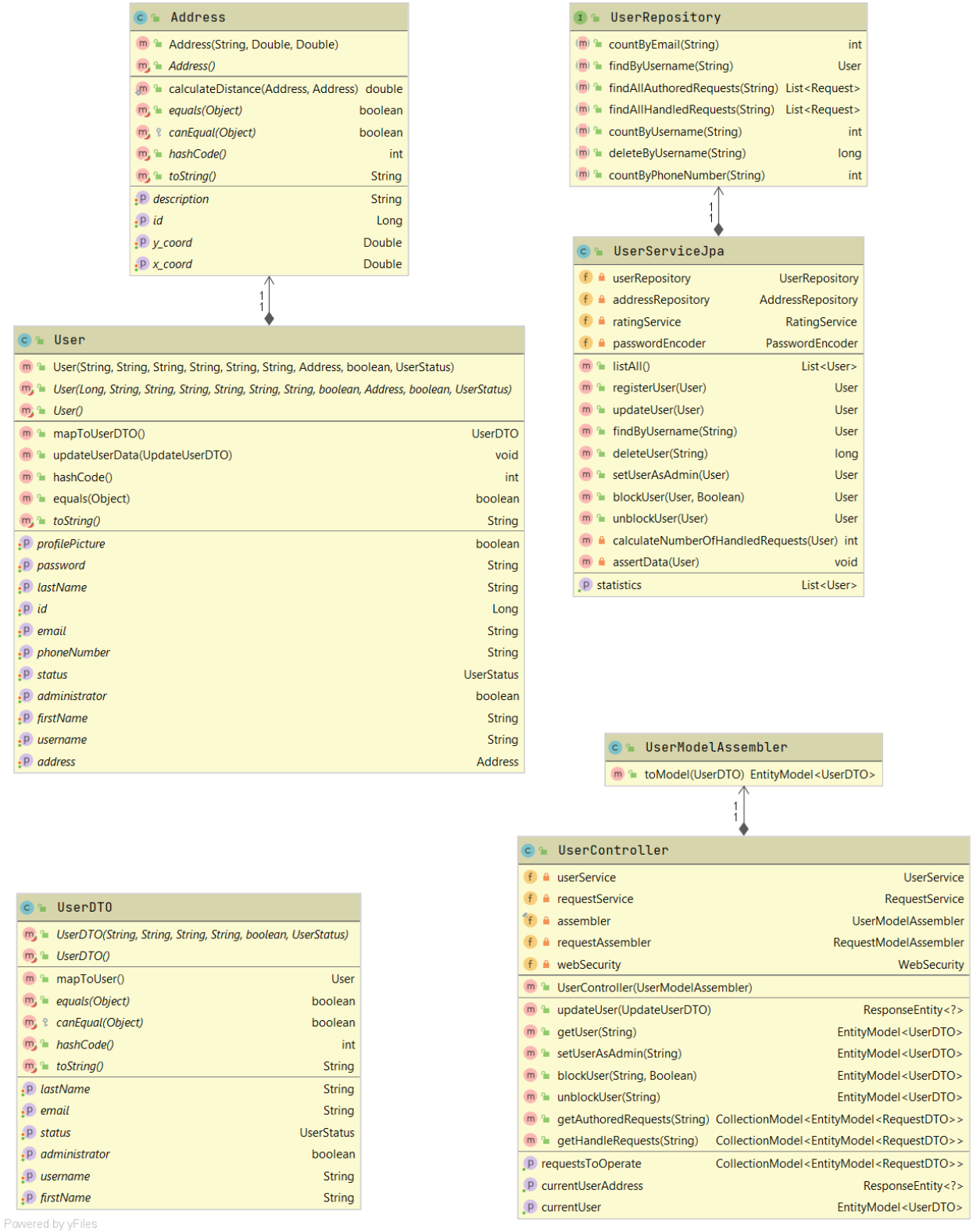
Slika 4.3: Struktura aplikacije i konfiguracija

Glavni preduvjet za obavljanje bilo kakve aktivnosti unutar aplikacije je da korisnik ima izrađen korisnički račun i prijavljen je u sustav. Klasom *RegistrationDTO* modeliraju se podaci koje korisnik unosi prilikom registracije. Podaci takvog objekta služe kao predložak za stvaranje korisničkog računa. Za prijavu u sustav, provjeru korisničkog imena i lozinke te dodjeljivanje uloga zadužena je klasa *AppUserDetailsService*. Korisniku može biti dodjeljena uloga ROLE_USER i ROLE_ADMIN kojima su određene ovlasti slanja zahtjeva na kontrolere.



Slika 4.4: Klase koje reguliraju registraciju i prijavu

Slika 4.5 sadržava klase koje su ključne za aktivnosti i rad s korisnicima. Klasa *User* predstavlja sam entitet kojim je opisan pojedini korisnik. Sadržava sve relevantne atribute za opis korisnika, konstruktore te funkciju *mapToUserDTO* kojom se objekt *User* pretvara u objekt pogodniji za komunikaciju sa *frontendom*. *UserDTO* predstavlja glavni model komunikacije s vanjskim korisnikom te skriva informacije o pojedinom *Useru* koje nisu relevantne za prikaz na *frontendu*. Omogućena je i pretvorba u suprotnom smjeru, iz klase *UserDTO* u klasu *User*. Klasa *UserModelAssembler* omata klasu *UserDTO* u oblik najprikladniji za interaktivnost prikaza, uključujući dodatne linkove. Pristup i manipulacija zapisima korisnika u bazi podataka ostvaren je klasom *UserRepository* koja sadrži sve standardne metode za rad sa zapisima. Poslovna logika koja se tiče korisnika ostvarena je u klasi *UserServiceJpa*. *UserController* zadužen je za obrađivanje zahtjeva. On inicijalizira komunikaciju sa servisom i bazom podataka i oblikuje podatke o korisnicima u odgovor na zahtjev.

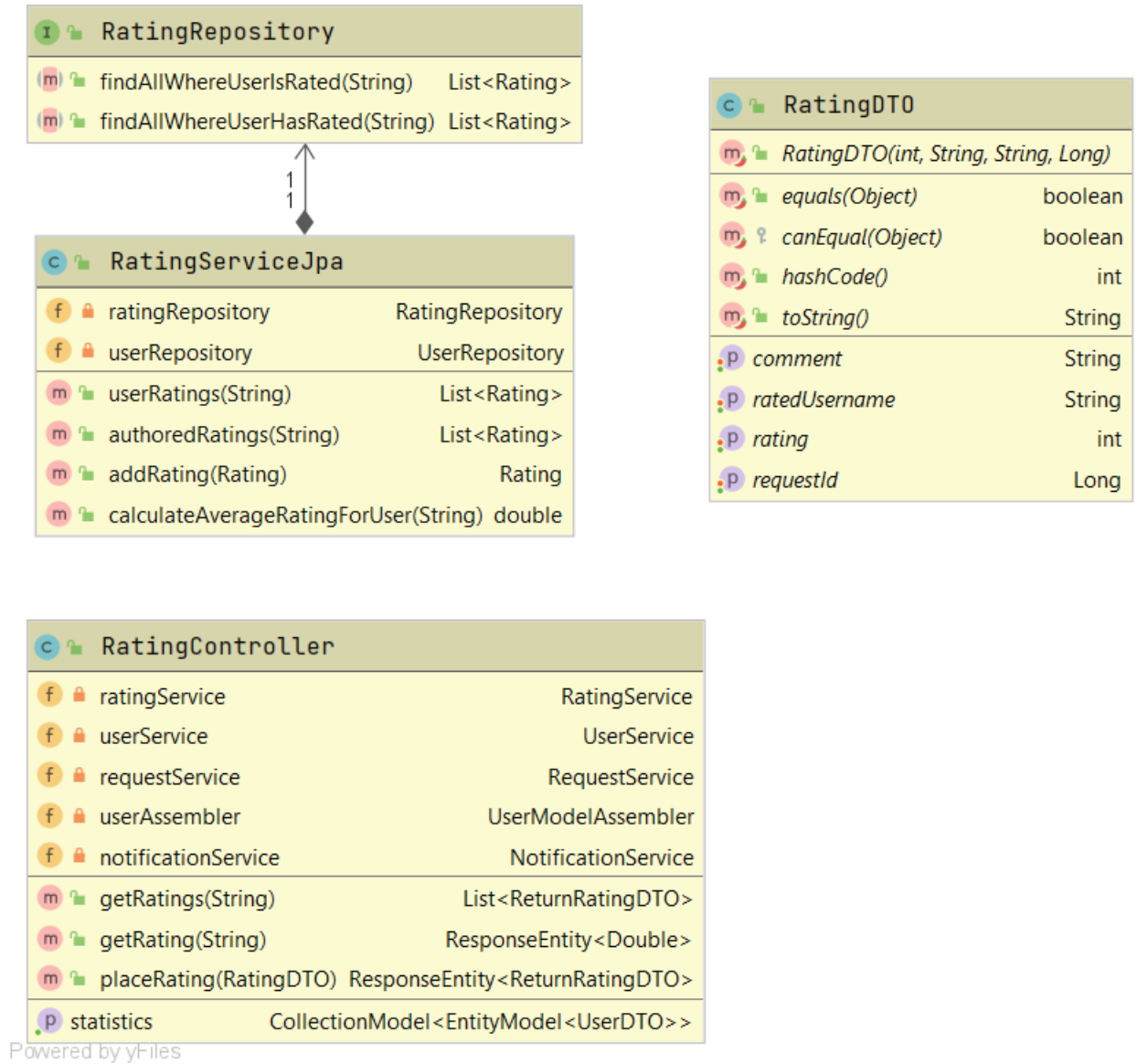


Slika 4.5: Klase koje reguliraju rad s korisnicima sustava

Klasa *Request* modelira zahtjeve korisnika. Kao i u slučaju sa klasom *User*, ova klasa ima pripadajuću klasu DTO-a, repozitorija, servisa i kontrolera. Prilikom zadavanja zahtjeva, na kontoroler dolazi zahtjev koji u tijelu sadržava objekt tipa *CreateRequestDTO* koji sadrži isključivo podatke koje unosi sam korisnik. Taj objekt je potrebno stvoriti objekt tipa *Request* koristeći dobivene podatke i predefinirane podatke koji opisuju stanje zahtjeva kako bi se on mogao spremiti u bazu podataka.

Slika 4.6: Klase koje reguliraju rad sa zahtjevima

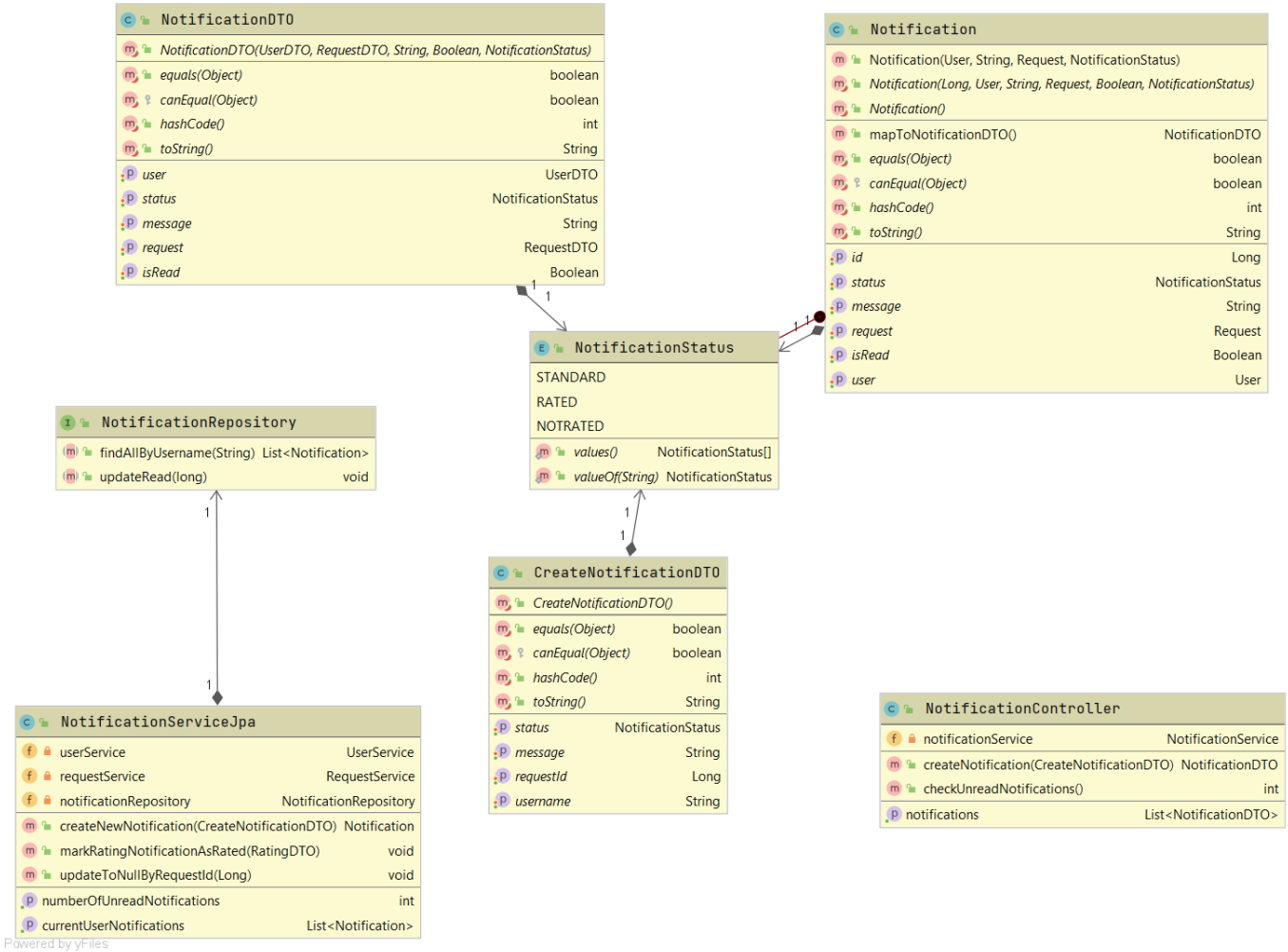
Korisnicima je omogućeno međusobno ocjenjivanje te su na slici 4.7 prikazane klase koje služe modeliranju ocjena i ocjenjivanja. Svaka instancla klase *RatingDTO* ima postavljenog korisnika koji je ocjenjen,ocjenu, komentar te opcionalno zahtjev na koji se odnosi. *RatingServiceJpa* omogućuje prikaz ocjena koje je pojedini korisnik autorirao te ocjena koje je pojedini korisnik dobio.



Slika 4.7: Klase koje reguliraju ocjenjivanje korisnika

Korisnicima za razne aktivnosti sudjelovanja na stranici(stvaranje zahtjeva, javljanje na zahtjev i mnogi drugi) dolaze obavijesti. Na slici 4.8 prikazane su sve klase koje služe za modeliranje i funkcioniranje obavijesti. Razred *Notification* modelira jednu obavijest. Razred *NotificationDTO* sadrži sve informacije kako bi se obavijest poslala korisniku, tj. sadrži korisnika kojemu je obavijest namijenjena, status obavijesti, poruku koju obavijest nosi, zahtjev za koju je obavijest vezana i na kraju jeli korisnik pročitao obavijest. Kao i razred *Request* sadrži pripadajuće razrede repozitorija, servisa, kontrolera i DTO-a za stvaranje(*CreateNotificationDTO*).

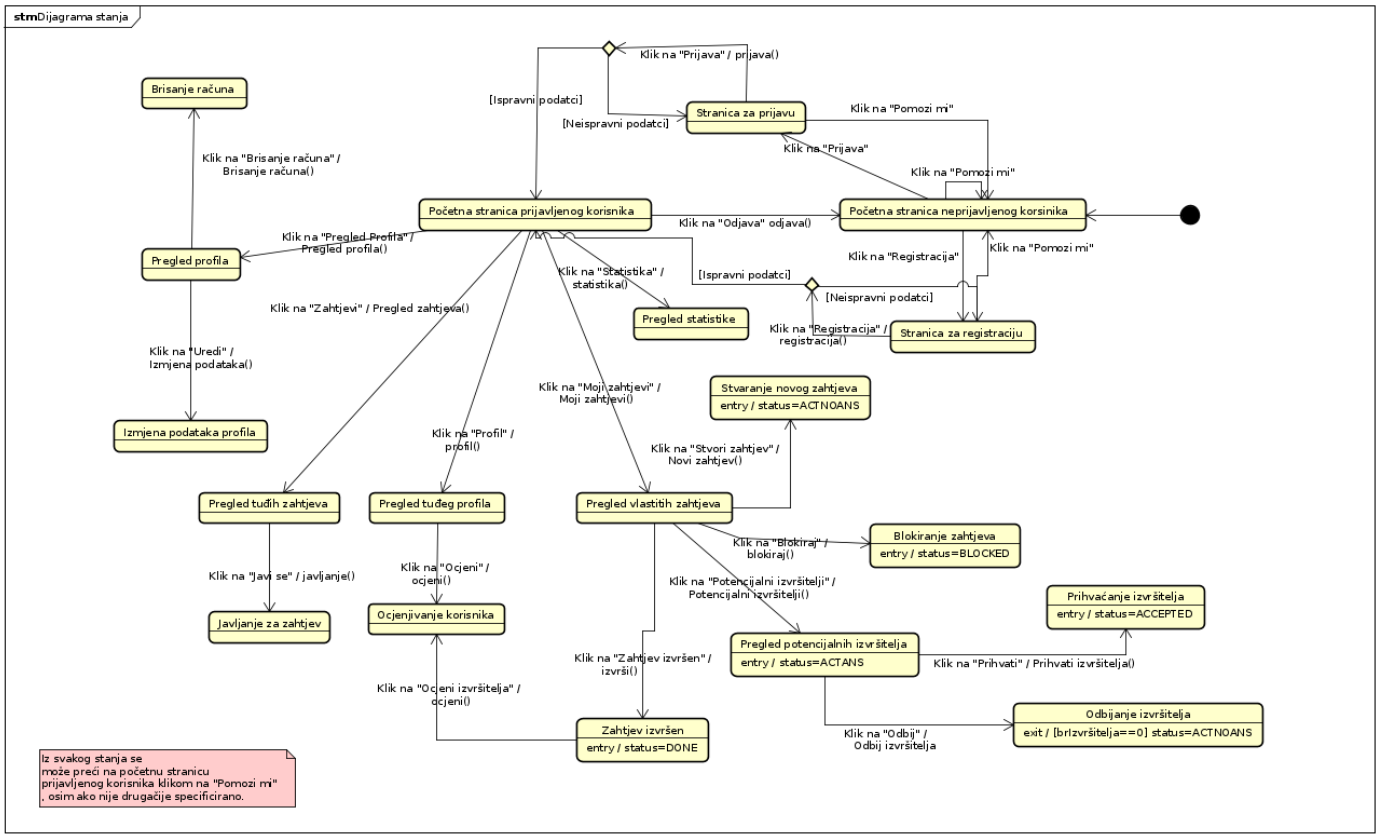
.



Slika 4.8: Klase koje reguliraju obavijesti

4.3 Dijagram stanja

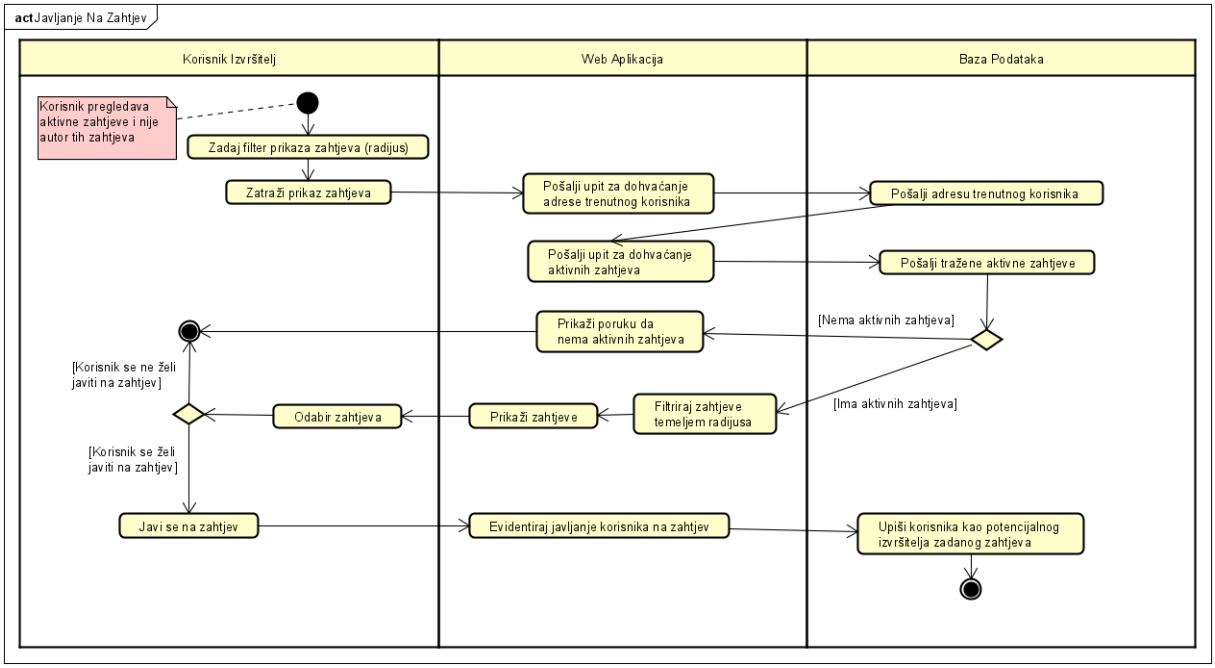
Dijagram stanja je tip dijagrama koji služi za opis ponašanja sustava pomoću stanja u koja se može preći nekim događajem. Na slici 4.8 prikazan je dijagram stanja jednog korisnika(koji nije administrator).Nakon prijave(ili registracije ako korisnik nema korisnički račun), korisnik odlazi na početnu stranicu aplikacije. Na početnoj stranici korisnik može: pregledati vlastiti profil, pregledati vlastite zahtjeve, pregledati profile i zahtjeve drugih korisnika, te pregledati statistiku. Klikom na "Pregled profila" korisniku se otvara vlastiti profil, te ga može izbrisati ili izmijeniti. Klikom na "Moji zahtjevi" korisnik može pregledati vlastite zahtjeve. Uz to može stvoriti novi zahtjev, blokirati postojeći, pregledati potencijalne izvršitelje nekog zahtjeva(i time prihvatiti ili odbiti nekog od potencijalnih izvršitelja) i na kraju postaviti zahtjev kao izvršen(i ocijeniti izvršitelja zahtjeva). Klikom na "Zahtjevi" korisniku se omogućuje pregled zahtjeva drugih korisnika i samim time se omogućava javljanje za zahtjeve. Klikom na "Profil" korisniku se otvara profil korisnika kojeg želi pregledati, te može vidjeti cjelokupnu ocjenu tog korisnika i sam ga može ocijeniti. Na kraju korisnik može pregledati i statistiku klikom na "Statistika" gdje se prikazuju 3 najbolje ocijenjena korisnika.



Slika 4.9: Dijagram stanja

4.4 Dijagram aktivnosti

Slika 4.8 prikazuje aktivnost korisnika izvršitelja (ubuduće korisnik) pri javljanju na jedan aktivni zahtjev. Na početku aktivnosti korisnik pregledava aktivne zahtjeve i nije autor tih zahtjeva. Nakon toga zadaje radijus koji će se koristiti za prikaz samo onih zahtjeva koji su unutar tog radijusa u odnosu na korisnikovu lokaciju. Ukoliko nema takvih zahtjeva, korisniku se prikaže poruka da nema aktivnih zahtjeva i aktivnost se završava. Ukoliko ima takvih zahtjeva oni se prikazuju korisniku, te se čeka njegov odabir. Ukoliko korisnik ne želi odabrati zahtjev, akcija se tu završava. Ukoliko korisnik odabere neki od tih zahtjeva, korisnik će biti evidentiran kao potencijalni izvršitelj.



Slika 4.10: Aktivnost javljanja na zahtjev

4.5 Dijagram komponenti

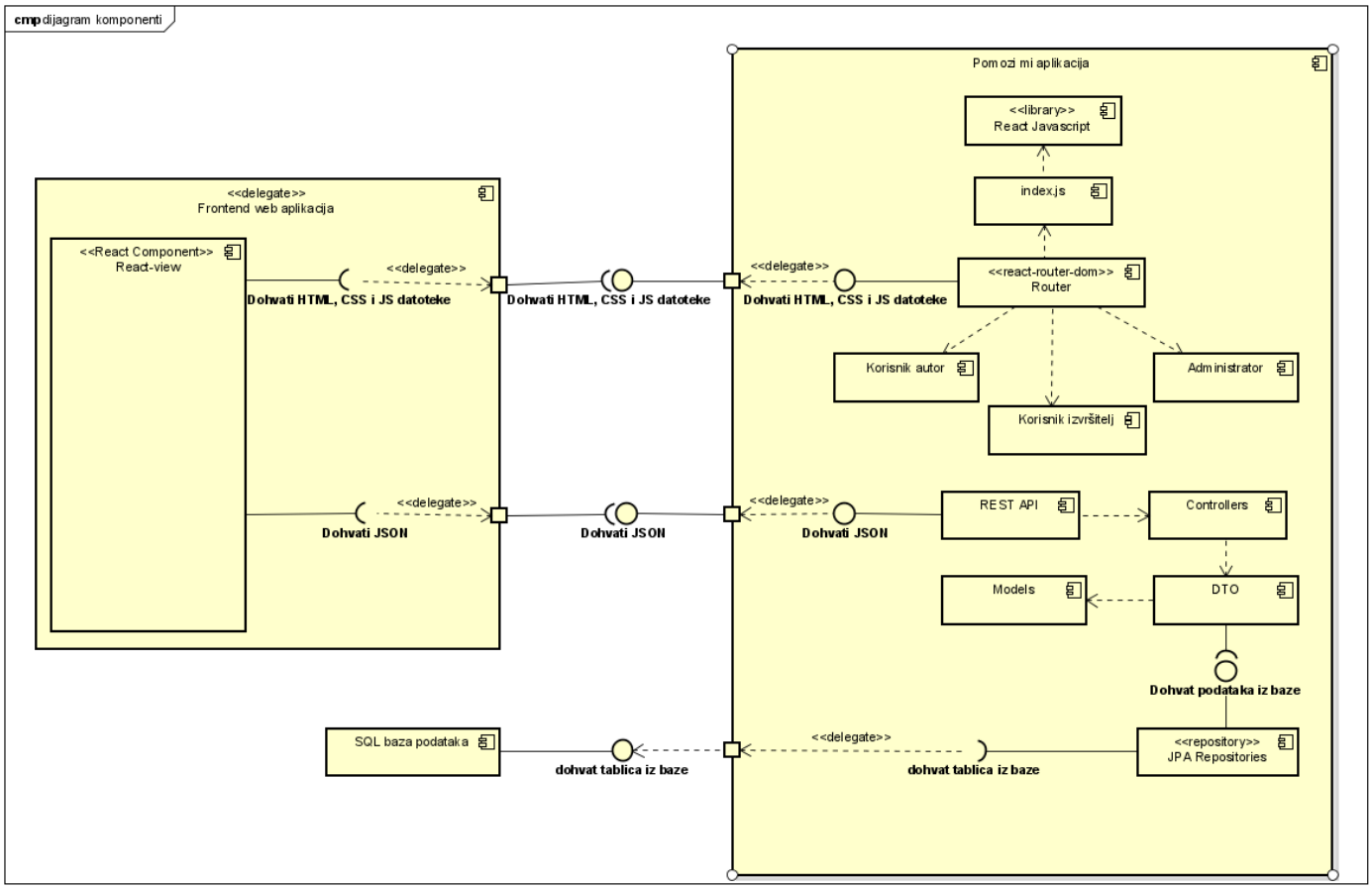
Pomoću dijagrama komponenti prikazanog na slici 4.9 prikazan je međuodnos glavnih komponenti sustava. Najgrublja podjela komponenata koje međusobno komuniciraju u *ModelViewController* (MVC) obrascu je na:

- Model koji predstavlja skup svih entiteta koji se spremaju u bazi te se nad njima vrše operacije.
- View kojeg predstavlja komponenta Frontend web aplikacija.
- Controller koji služi za primanje HTTP zahtjeva i slanje odgovora.

Frontend web aplikacija komunicira sa ostatkom aplikacije pri posluživanju statičkog sadržaja poput HTML, CSS i JS datoteka te sa backend dijelom aplikacije u formatu JSON. Posluživanje sadržaja frontend dijelu aplikacije ovisi o ulozi korisnika kojemu se poslužuje te shodno ulozi komponenta Router odlučuje o tome koje se datoteke poslužuju. Također, u to su uključene i datoteke JavaScript libraryja React koji se koristi u oblikovanju korisničkog sučelja.

Format JSON glavni je format za komunikaciju sa REST API-jem kojeg aplikacija koristi. REST API po primitku JSON objekta mapira ga u pripadajući objekt u Javi. Vrijedi i obrnuto, prilikom slanja Java objekta on se prethodno pretvara u pripadajući JSON i u tom obliku šalje HTTP protokolom.

Komponenta kontrolera putem DTO objekata ostvaruje komunikaciju sa REST API-jem i poslovnom logikom aplikacije. Pristup bazi na backendu nudi se sučeljem s komponentom *JPARepositories* te ona služi kao glavna komunikacija aplikacije i SQL baze podataka u jeziku JPQL.



Slika 4.11: Dijagram komponenti

5. Implementacija i korisničko sučelje

5.1 Korištene tehnologije i alati

Komunikacija našem u timu je realizirana korištenjem aplikacija Discord[1] i Whatsapp[2].
Za izradu UML dijagrama korišten je alat Astah Professional[3], a kao sustav za upravljanje izvornim kodom Git[4]. Udaljeni repozitorij projekta je dostupan na web platformi GitLab[5], a kao razvojno okruženje korišten je IntelliJ IDEA[6] - tvrtke JetBrains[7] te Microsoftov Visual Studio Code[8].
Naša aplikacija napisana je koristeći Spring Framework[9] i jezik Java[10] za izradu backenda te React[11] - JavaScript library za izradu frontenda. React je open-source, front end, JavaScript knjižnica za izgradnju korisničkog sučelja ili UI komponenti. Održavaju ga Facebook i zajednica pojedinačnih programera i tvrtki. Spring razvojno okruženje je Java platforma koja pruža široki panel opcija kao podrški razvoju Java aplikacija. Spring rukuje infrastrukturom, tako da programer može usmjeriti svoj fokus na razvoj aplikacije.
Tijekom izrađivanja aplikacije koristili smo i Postman[12] za simuliranje HTTP zahtjeva nad aplikacijom, Google Drive[13] za efikasno dijeljenje organizacijskih informacija i raznih datoteka. Baza podataka se nalazi na poslužitelju u oblaku Amazon Web Services[14].

[1] https://discord.com/
[2] https://www.whatsapp.com/
[3] http://astah.net/editions/professional
[4] https://git-scm.com/
[5] https://gitlab.com/
[6] https://www.jetbrains.com/idea/
[7] https://www.jetbrains.com/
[8] https://visualstudio.microsoft.com/
[9] https://spring.io/
[10] https://www.oracle.com/java/
[11] https://reactjs.org/
[12] https://www.postman.com/
[13] https://workspace.google.com/products/drive/
[14] https://aws.amazon.com/

5.2 Ispitivanje programskog rješenja

dio 2. revizije

U ovom poglavlju je potrebno opisati provedbu ispitivanja implementiranih funkcionalnosti na razini komponenti i na razini cijelog sustava s prikazom odabranih ispitnih slučajeva. Studenti trebaju ispitati temeljnu funkcionalnost i rubne uvjete.

5.2.1 Ispitivanje komponenti

Unit testovi za Service sloj

Da bi napravili unit testove za svaki razred posebno, koristiti ćemo framework Mockito. Koristiti ćemo ga tako da ”simuliramo” rad objekata koji testni razred koristi. @RunWith(MockitoJUnitRunner.class) je anotacija koja se stavlja iznad razreda u kojem su unit testovi. @InjectMocks anotacija nam označuje objekt nad kojim će se vršiti unit test (System under test). @Mock anotacija stvara objekt koji je ”lažan”, on neće izvoditi svoje normalne metode, već ćemo mu mi kako će se ponašati na svaki poziv metode.

UserServiceJpa unit test

listAll() - metoda listAll() vraća listu korisnika (List¡User¿). U unit testu stvaramo 3 nova korisnika i simuliramo metodu listAll() da kao rezultat vrati listu od ta 3 korisnika. Zatim pozovemo metodu listAll() i provjerimo vraća li ta metoda dobre podatke nazad.

registerUser() - unit testovi za ovu metodu ispituju baca li IllegalArgumentException kada mu predamo korisnika kojemu fali neki od obaveznih podataka za registraciju ili kada već postoji korisnik s tim korisničkim imenom, emailom ili brojem telefona. Ispituje se i radi li metoda sve dobro kada mu predamo korisnika koji ima sve podatke, i ne postoje već korisnici s takvim korisničkim imenom, emailom i brojem telefona.

updateUser() - unit testovi za ovu metodu ispituju baca li IllegalArgumentException kada mu predamo korisnika kojemu fali neki od obaveznih podataka za ažuriranje podataka. Ispituje se i radi li metoda očekivano kada se preda korisnik kojemu bi se trebalo bez greške ažurirati podatke.

findByUsername() - unit testovi za ovu metodu ispituju baca li IllegalArgumentException kada mu predamo korisničko ime koje je null, vraća li null kada mu predamo korisničko ime za kojeg ne postoji registrirani korisnik i radi li metoda očekivano kada se preda korisničko ime za kojeg postoji registrirani korisnik.

deleteUser() - unit testovi za ovu metodu ispituju vraća li ispravno broj obrisanih korisnika. Ukoliko mu predamo korisničko ime za koje ne postoji registrirani korisnik treba vraćati da je broj obrisanih korisnika 0, a ako mu predamo korisničko ime za koje postoji registrirani korisnik treba vraćati da je broj obrisanih korisnika 1.

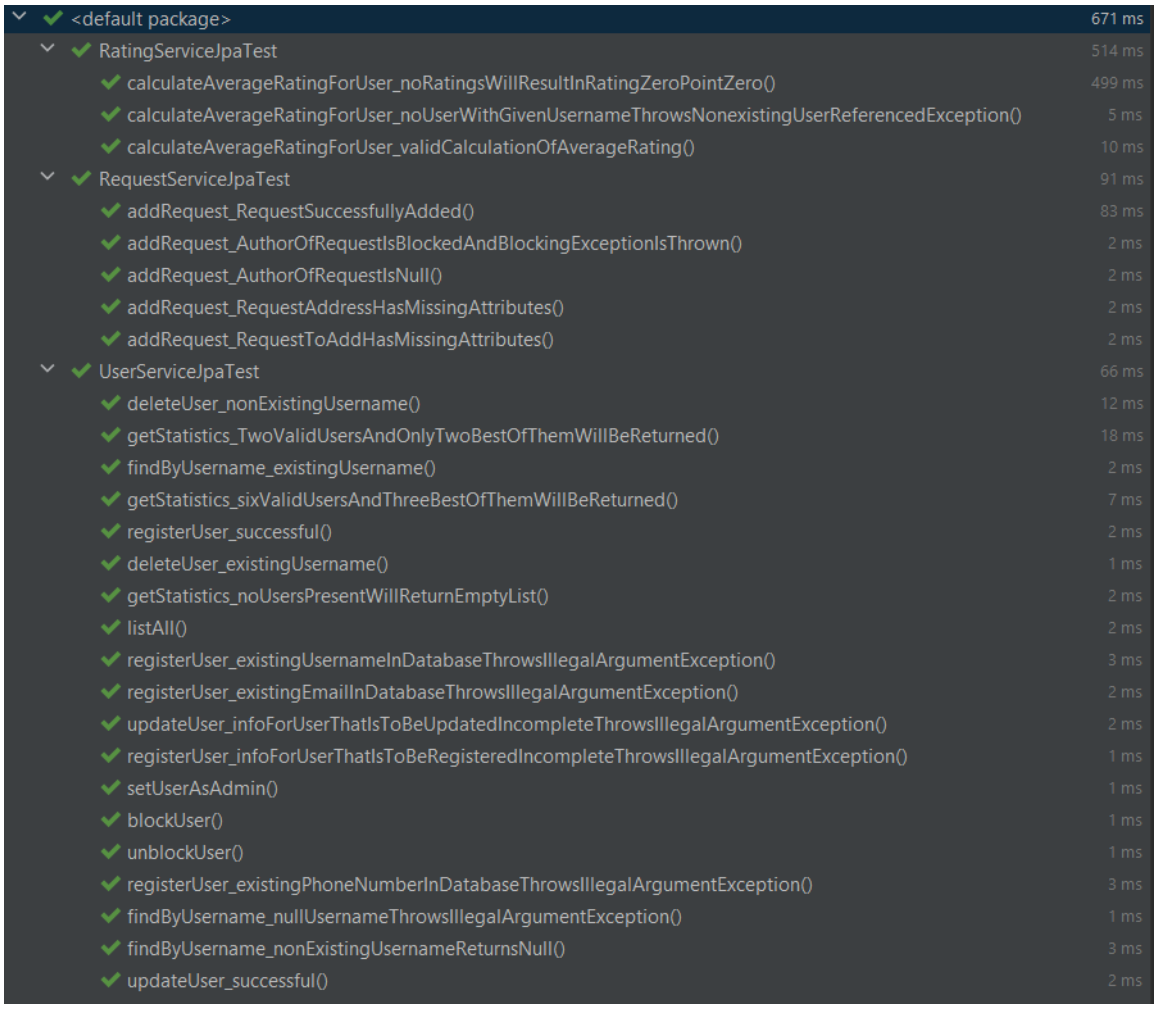
getStatistics() - Ukoliko ne postoje registrirani korisnici pri pozivu ove metode, ona bi trebala vratiti praznu listu. Ako postoje registrirani korisnici, ali ne njih više od dva, metoda bi trebala vratiti samo listu od tog jednog ili dva korisnika, tako da je prvi onaj bolje ocjenjeni. Ako postoji tri ili više registriranih korisnika metoda bi trebala vratiti listu od tri najbolje ocjenjena korisnika. Testira se i u kojem poretku je metoda vratila najbolje ocijenjene korisnike, tu se testira je li dobro implementirano sortiranje korisnika.

RatingServiceJpa unit test

calculateAverageRatingForUser() - Ukoliko se preda korisnik čije korisničko ime nema registrirano u bazi podataka baca se NonexistingUserReferencedException. Ako postoji korisnik, ali nema nikakvih ocjena za njega, metoda bi trebala vratiti ocjenu 0.0 . Testira se i računa li metoda ispravno prosječnu ocjenu za korisnika koji postoji i za kojega ima više ocjena spremljenih u bazi.

RequestServiceJpa unit test

addRequest() - Ako se metodi preda zahtjev čiji je autor postavljen na null, metoda bi trebala postaviti ulogiranog korisnika kao autora zahtjeva. Ako adresa zahtjeva ne sadrži opis, x kordinatu ili y kordinatu, metoda bi trebala baciti IllegalArgumentException. Ako je predan zahtjev koji je potpun i točan metoda bi trebala uspješno dodati zahtjev.



Slika 5.1: Rezultati unit testova

5.2.2 Ispitivanje sustava

Za ispitivanje sustava korišten je alat za automatizirano testiranje Selenium WebDriver. Ispitni slučajevi pisani su u jeziku Java.

Cilj ispitivanja sustava je ispitati ponašanje sustava u cjelini, onako kako ga vidi korisnik aplikacije te dostupnost svih funkcionalnosti koje korisnik može koristiti. Napisani ispitni slučajevi prije svega su ispitivali pravilan izgled komponenti i funkcionalnosti koje se nude kroz korisničko sučelja u ovisnosti o stanju sustava, stanju pojedinog zahtjeva i ovlastima korisnika koji je prijavljen u sustav. Testiranje je koristilo dva korisnika već registrirana u sustav, jednog s ovlastima običnog korisnika i drugog s ovlastima administratora.

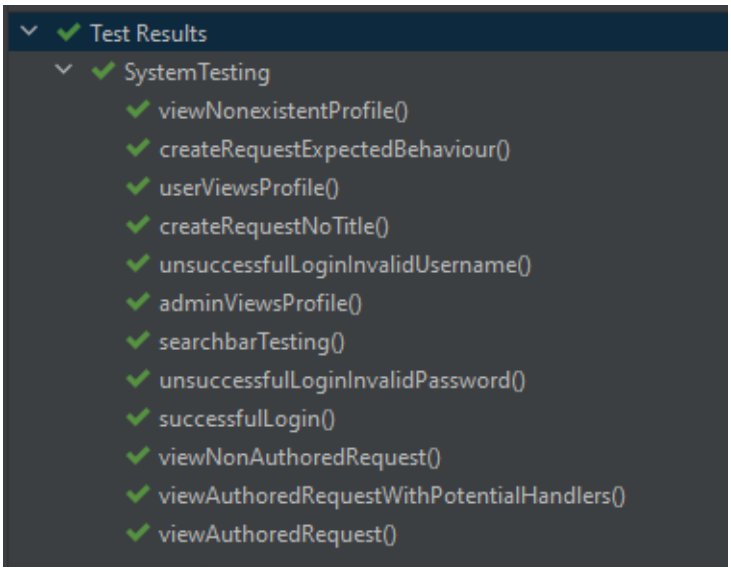
Komponente koje najviše ovise o ovlastima i stanju sustava su komponenta za prikaz pojedinog zahtjeva i komponenta profila korisnika. Za primjer, funkcionalnosti rada sa zahtjevima testiraju metode:

1. *createRequestExpectedBehaviour()* - očekivani unos u formu zahtjeva
2. *createRequestNoTitle()* - neispravan unos u formu zahtjeva
3. *viewNonAuthoredRequest()* - prisutnost komponenti za javljanje na zahtjev
4. *viewAuthoredRequest()* - prisutnost komponenti za blokiranje zahtjeva
5. *viewAuthoredRequestWithPotentialHandlers()* - prisutnost komponenti za pregled potencijalnih izvršitelja te sam pregled

Testni slučajevi fokusirali su se na različite situacije i rubne slučajeve u prikazu i funkcionalnostima. Rubni slučajevi poput pogrešnog unosa u formu ili nedostatka nekog unosa se u aplikaciji signaliziraju različitim porukama o pogrešci. Testne metode za takve slučajeve prije svega su se fokusirali na traženje odgovarajuće poruke o pogrešci. Konkretan primjer za to su testni slučajevi unosa pogrešnih korisničkih podataka u formu za prijavu.

Neimplementirana funkcionalnost koja je također testirana je traka za pretraživanje. Testom je pokazano da akcijama nad njom se ne mjenja stanje sustava i ne remeti se normalan tijek funkcioniranja aplikacije.

Glavnu prepreku za automatizaciju ispitivanja sustava predstavljala je priroda testiranog sustava. Sustav se sastoji od brojnih aktivnosti koje se mogu izvoditi samo jednom, poput javljanja na pojedini zahtjev ili biranja izvršitelja za zahtjev te aktivnosti koje zahtjevaju slijednu interakciju više korisnika. Takve akcije teško je automatizirano ispitivati zato što to neminovno u sustavu akumulira velik broj zahtjeva samo za testiranje ili testove nije moguće ponavljati više puta, čime se u potpunosti gubi njihova svrha. Stoga su navedene funkcionalnosti izostavljene iz automatiziranog ispitvanja alatom Selenium i odrađene ručno.



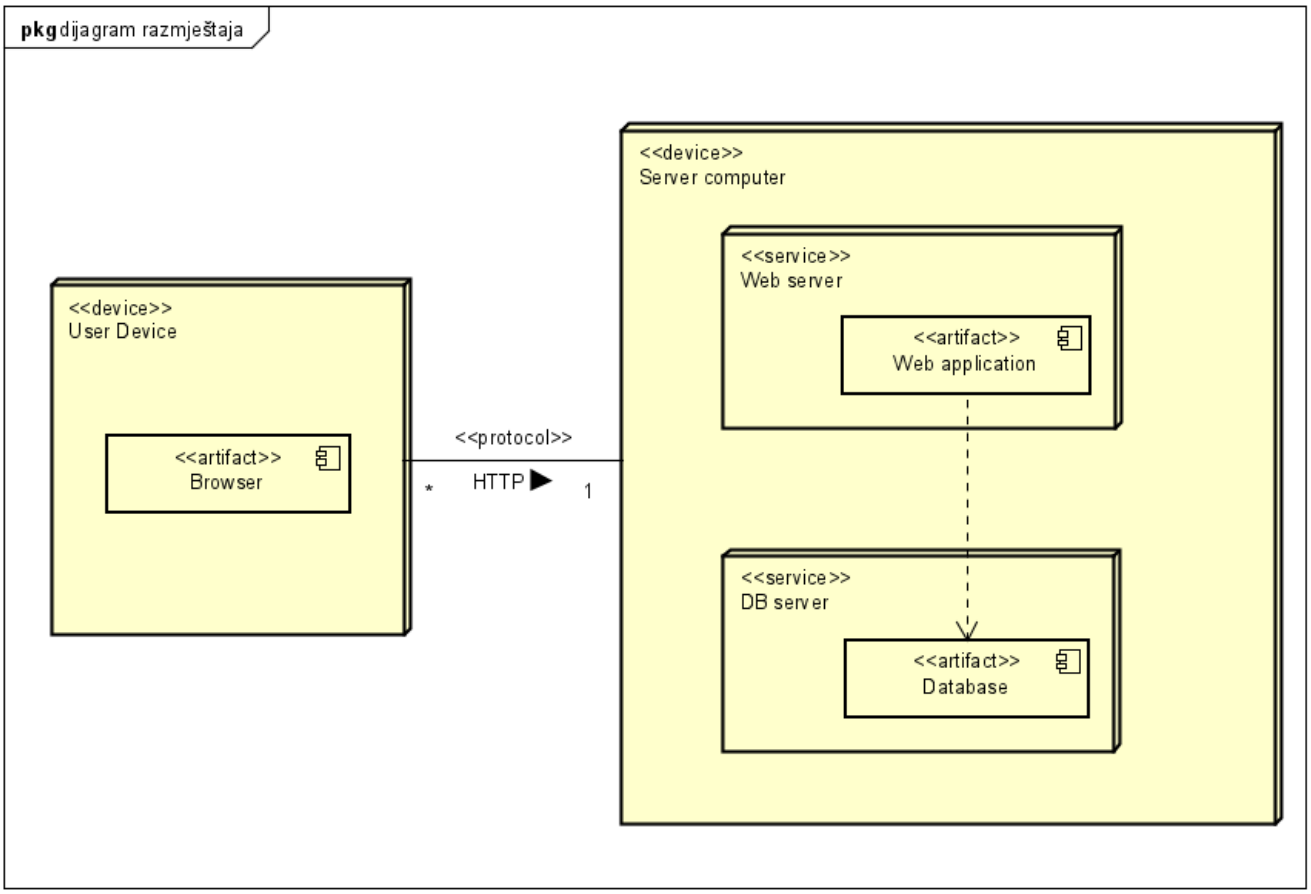
Slika 5.2: Rezultati testova sustava

5.3 Dijagram razmještaja

Na slici 5.1 nalazi se dijagram razmještaja koji prikazuje raspored programske podrške aplikaciji unutar sklopovlja.

Sa strane korisnika aplikacije, sklopovlje predstavlja njegovo računalo, mobilni telefon ili bilo koji drugi uređaj s pristupom internetu i instaliranim browserom. Internetski browser zadužen je za uspostavu konekcije sa poslužiteljem koja će omogućiti slanje i posluživanje zahtjeva. Sa strane poslužitelja sklopovlje predstavlja serversko računalo. Dvije glavne usluge koje se nalaze na serverskom računalu su baza podataka i web server koje su nužne za posluživanje zahtjeva.

Komunikacija između korisnika i poslužitelja vrši se putem *stateless* HTTP protokola. Tijekom slanja zahtjeva serveru, korisnikov browser ima otvorenu jednu HTTP vezu prema serveru. S druge strane server može posluživati više zahtjeva iz više različitih izvora te stoga može imati i više aktivnih HTTP veza.



Slika 5.3: Dijagram razmještaja

5.4 Upute za puštanje u pogon

dio 2. revizije

U ovom poglavlju potrebno je dati upute za puštanje u pogon (engl. deployment) ostvarene aplikacije. Na primjer, za web aplikacije, opisati postupak kojim se od izvornog kôda dolazi do potpuno postavljene baze podataka i poslužitelja koji odgovara na upite korisnika. Za mobilnu aplikaciju, postupak kojim se aplikacija izgradi, te postavi na neku od trgovina. Za stolnu (engl. desktop) aplikaciju, postupak kojim se aplikacija instalira na računalo. Ukoliko mobilne i stolne aplikacije komuniciraju s poslužiteljem i/ili bazom podataka, opisati i postupak njihovog postavljanja. Pri izradi uputa preporučuje se ***naglasiti korake instalacije uporabom natuknica*** *te koristiti što je više moguće* ***slike ekrana*** *(engl. screenshots) kako bi upute bile jasne i jednostavne za slijediti.*

Dovršenu aplikaciju potrebno je pokrenuti na javno dostupnom poslužitelju. Studentima se preporuča korištenje neke od sljedećih besplatnih usluga: Amazon AWS, Microsoft Azure ili Heroku. Mobilne aplikacije trebaju biti objavljene na F-Droid, Google Play ili Amazon App trgovini.

6. Zaključak i budući rad

dio 2. revizije

U ovom poglavlju potrebno je napisati osvrt na vrijeme izrade projektnog zadatka, koji su tehnički izazovi prepoznati, jesu li riješeni ili kako bi mogli biti riješeni, koja su znanja stečena pri izradi projekta, koja bi znanja bila posebno potrebna za brže i kvalitetnije ostvarenje projekta i koje bi bile perspektive za nastavak rada u projektnoj grupi.

Potrebno je točno popisati funkcionalnosti koje nisu implementirane u ostvarenoj aplikaciji.

6.1 Izrada projekta i tehnički izazovi

Izrada projekta protezala se kroz period od 12 tjedana. Prvi dio izrade fokusirao se na definiranje korisničkih zahtjeva i zahtjeva sustava, dok se drugi dio okrenuo ka implementaciji definiranog sustava.

Imajući na umu da nitko iz projektne grupe nije bio upoznat sa tehnologijama koje su se koristile (Spring Framework i React), glavni izazov predstavljalo je upoznavanje s novim tehnologijama.
Inicijalno, to je otežavalo vremensko planiranje i nošenje sa klasičnim problemima uporabe Springa i Reacta koji se mogu zaobići korištenjem koncepta *best practice*.

6.1.1 Tehnički izazovi korištenja radnog okvira Spring

Najveći izazov prilikom korištenja radnog okvira Spring bili su u uspostavi autentikacije i autorizacije. Radi jednostavnost izvedbe inicijalne inačice, odlučeno je za konfiguraciju sigurnosti aplikacije koristiti već postojeću Springovu klasu *WebSecurityConfigurerAdapter*. Najduže je trajalo uspostavljanje konfiguracije autentikacije putem *UserDetailsSerivce*-a iz razloga nepotpunosti i manjkavosti informacija o funkcionalnostima navedenog service-a na internetu i čestih neočekivanih

ponašanja koja su posljedica različitih konfiguracija sigurnosti u referentnim primjerima i našoj aplikaciji. Autentikacija je uspješno uspostavljena pregledavanjem implementacije korištenih klasa i službene dokumentacije. Manjkavost koju uviđamo u našoj izvedbi sigurnosti je nekorištenje tokena, već svaki zahtjev mora sadržavati autentikacijske podatke *username* i *password*.

6.1.2 Izazovi korištenja knjižnice React

Glavnina tehničkih izazova korištenja knjižnice React je proizašla iz potrebe za adaptacijom komponenti koje React nudi. Kao kompromis između pisanja vlastitih i relativno teške adaptacije već postojećih komponenti, glavni *layout* web stranice izrađen je ručno i uz pomoć *Bootstrapa*, dok su se za pojedine komponente unutar stranice koristile komponente *Semantic-UI*.

6.2 Budući rad

S obzirom na kratko vrijeme za izvedbu aplikacije i ograničeno prethodno iskustvo projektnog tima, prva inačica aplikacije ostaje otvorena za neka poboljšanja. U ovom odjeljku opisani su prijedlozi za restrukturiranje dijela funkcionalnosti te koje dodatne funkcionalnosti bi poboljšale aplikaciju.

Restrukturiranje sigurnosti i izmjene potrebne za uvođenje *JWT tokena* kao sredstva autentikacije prva je i najbitnija potrebna izmjena. Time se osigurava veća sigurnost podataka u odnosu na slanje *passworda* u svakom HTTP zahtjevu. Dodatno, e-mail verifikacijom unesene adrese prilikom registracije potrebno je osigurati da se nitko osim stvarnog posjednika adrese ne može registrirati u sustav.

Kako bi se osigurala kvalitetnija komunikacija i interakcija između korisnika aplikacije te bolji *user experience* dodatne implementacija funkcionalnosti pretraživanja korisnika i zahtjeva te *chat* koje nisu izvedene u prvoj inačici je nužna.

Od zahtjevanih i predviđenih funkcionalnosti aplikacije u prvoj inačici nisu implementirane:

1. Brisanje korisničkog računa
2. Prikaz lanaca povjerenja

Predviđene i neimplementirane funkcionalnosti bit će uključene u iduću inačicu aplikacije.

6.3 Zaključak

Izvođenje ovog projekta uvelike je pridonjelo praktičnom iskustvu svih članova projektne grupe. Nadasve najbitnije znanje koje je projektna grupa dobila je uvid u cjelokupni proces razvoja određenog programskog proizvoda, počevši sa planiranjem, razradom pa sve do *deployment*-a. To znanje je učinilo dosadašnja znanja članova projektne grupe kompletnijima i primjenjivijima. Također, projekt je potaknuo grupu na aktivnije korištenje već gotovih knjižnica i komponenata i njihovu adaptaciju za specifične potrebe projekta.

Popis literature

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

1. Programsko inženjerstvo, FER ZEMRIS, http://www.fer.hr/predmet/proinz
2. I. Sommerville, "Software engineering", 8th ed, Addison Wesley, 2007.
3. T.C.Lethbridge, R.Langaniere, "Object-Oriented Software Engineering", 2nd ed. McGraw-Hill, 2005.
4. I. Marsic, Software engineering book", Department of Electrical and Computer Engineering, Rutgers University, http://www.ece.rutgers.edu/~marsic/books/SE
5. The Unified Modeling Language, https://www.uml-diagrams.org/
6. Astah Community, http://astah.net/editions/uml-new

Indeks slika i dijagrama

Dodatak: Prikaz aktivnosti grupe

Dnevnik sastajanja

1. sastanak
   - Datum: 4. listopada 2020.
   - Prisustvovali: H.Rom, Ž.Rački, M.Stanić, D.Oreč, D.Milde, A.Ilinović, M.Bakšić
   - Teme sastanka:
     - upoznavanje
     - dogovor o tehnologijama za projekt
     - dogovor oko uloga
2. sastanak
   - Datum: 13. listopada 2020.
   - Prisustvovali: H.Rom, Ž.Rački, M.Stanić, D.Oreč, D.Milde, A.Ilinović, M.Bakšić
   - Teme sastanka:
     - upoznavanje s Gitom
     - definiranje korisničkih zahtjeva
3. sastanak
   - Datum: 20. listopada 2020.
   - Prisustvovali: H.Rom, Ž.Rački, M.Stanić, D.Oreč, D.Milde, A.Ilinović, M.Bakšić
   - Teme sastanka:
     - baza podataka
     - UC dijagrami
4. sastanak
   - Datum: 27. listopada 2020.
   - Prisustvovali: H.Rom, Ž.Rački, M.Stanić, D.Oreč, D.Milde, A.Ilinović, M.Bakšić
   - Teme sastanka:

  - Sekvencijski dijagrami
  - UC dijagrami

5. sastanak
   - Datum: 30. listopada 2020.
   - Prisustvovali: H.Rom, Ž.Rački, M.Stanić, D.Oreč, D.Milde, A.Ilinović, M.Bakšić
   - Teme sastanka:
     - login
     - registracija
     - podjela posla na backendu
     - frontend pregled dizajna
6. sastanak
   - Datum: 3. studenog 2020.
   - Prisustvovali: H.Rom, Ž.Rački, M.Stanić, D.Oreč, D.Milde, A.Ilinović, M.Bakšić
   - Teme sastanka:
     - Ispravak sekvencijskih dijagrama
     - frontend forms
     - Pregled napravljenog i prijedlozi za izmjene
7. sastanak
   - Datum: 5. studenog 2020.
   - Prisustvovali: H.Rom, Ž.Rački, M.Stanić, D.Oreč, D.Milde, A.Ilinović, M.Bakšić
   - Teme sastanka:
     - Pregled napravljenog i prijedlozi za izmjene
     - Izdavanje novih zaduženja
8. sastanak
   - Datum: 10. studenog 2020.
   - Prisustvovali: H.Rom, Ž.Rački, M.Stanić, D.Oreč, D.Milde, A.Ilinović, M.Bakšić
   - Teme sastanka:
     - Prepravke i finalizacija dokumentacije

U drugom ciklusu većina komunikacije je bila asinkrona putem Discord servera.

9. sastanak
    - Datum: 2. prosinca 2020.
    - Prisustvovali: H.Rom, Ž.Rački, M.Stanić, D.Oreč, D.Milde, A.Ilinović, M.Bakšić
    - Teme sastanka:
        - Restrukturiranje timova
        - Dogovor zadataka za sljedeća dva tjedna
10. sastanak
    - Datum: 16. prosinca 2020.
    - Prisustvovali: H.Rom, Ž.Rački, M.Stanić, D.Oreč, D.Milde, A.Ilinović, M.Bakšić
    - Teme sastanka:
        - Pregled napravljenog i prijedlozi izmjena
        - Definiranje daljnjih zadataka

Tablica aktivnosti

Kontinuirano osvježavanje

Napomena: Doprinose u aktivnostima treba navesti u satima po članovima grupe po aktivnosti.

	Mihaela Bakšić	**Antonio Ilinović**	**Dario Oreč**	**Hrvoje Rom**	**Mateo Stanić**	**Dominik Milde**	**Željko Rački**
Upravljanje projektom	15						
Opis projektnog zadatka						4	
Funkcionalni zahtjevi						2	
Opis pojedinih obrazaca	8						
Dijagram obrazaca	3						
Sekvencijski dijagrami							6
Opis ostalih zahtjeva			0.5		0.5	0.5	
Arhitektura i dizajn sustava			3		0.5	0.5	0.5
Baza podataka			4			1	
Dijagram razreda						3	
Dijagram stanja			2				
Dijagram aktivnosti		6					
Dijagram komponenti	2						
Korištene tehnologije i alati						2	1
Ispitivanje programskog rješenja		8					
Dijagram razmještaja	2						
Upute za puštanje u pogon							
Dnevnik sastajanja	2						
Zaključak i budući rad	2				1		
Popis literature							

	Mihaela Bakšić	**Antonio Ilinović**	**Dario Oreč**	**Hrvoje Rom**	**Mateo Stanić**	**Dominik Milde**	**Željko Rački**
Dodatne stavke kako ste podijelili izradu aplikacije							
izrada početne stranice					5	3	
front end					35	30	15
dizajn					5	15	
izrada baze podataka							3
spajanje s bazom podataka							
back end	40	20	20		0.1	3	
deployment						1.5	

Dijagrami pregleda promjena

dio 2. revizije

Prenijeti dijagram pregleda promjena nad datotekama projekta. Potrebno je na kraju projekta generirane grafove s gitlaba prenijeti u ovo poglavlje dokumentacije. Dijagrami za vlastiti projekt se mogu preuzeti s gitlab.com stranice, u izborniku Repository, pritiskom na stavku Contributors.