Programsko inženjerstvo

Ak. god. 2022./2023.

Sinappsa

Dokumentacija, Rev. *2*

Grupa: *Sheeshmishi*

Voditelj: *Ivan Krešo*

Datum predaje: *13.siječnja.2023.*

Nastavnik: *Laura Majer, mag. ing. comp*

Sadržaj

1 Dnevnik promjena dokumentacije **3**

2 Opis projektnog zadatka **6**

3 Specifikacija programske potpore **9**
3.1 Funkcionalni zahtjevi 9
3.1.1 Obrasci uporabe 11
3.1.2 Sekvencijski dijagrami 21
3.2 Ostali zahtjevi 24

4 Arhitektura i dizajn sustava **25**
4.1 Baza podataka 26
4.1.1 Opis tablica 27
4.1.2 Dijagram baze podataka 30
4.2 Dijagram razreda 31
4.3 Dijagram stanja 35
4.4 Dijagram aktivnosti 36
4.5 Dijagram komponenti 37

5 Implementacija i korisničko sučelje **38**
5.1 Korištene tehnologije i alati 38
5.2 Ispitivanje programskog rješenja 39
5.2.1 Ispitivanje komponenti 39
5.2.2 Ispitivanje sustava 42
5.3 Dijagram razmještaja 45
5.4 Upute za puštanje u pogon 46
5.4.1 Konfiguracija poslužitelja baze podataka 46
5.4.2 Konfiguracija baze podataka 46
5.4.3 Punjenje baze podataka 47
5.4.4 Pokretanje poslužitelja baze podataka 48
5.4.5 Konfiguracija poslužitelja serverske strane 48

5.4.6 Konfiguracija veze s bazom podataka 49
5.4.7 Konfiguracija poslužitelja klijentske strane 49
5.4.8 Definiranje adrese spajanja 49
5.4.9 Pokretanje aplikacije 50

6 Zaključak i budući rad **52**

Popis literature **54**

Indeks slika i dijagrama **55**

Dodatak: Prikaz aktivnosti grupe **56**

1. Dnevnik promjena dokumentacije

Rev.	Opis promjene/dodatka	Autori	Datum
0.1	Napravljen predložak.	Svi	27.10.2022.
0.2	Dodan 1. dio poglavlja: Opis projektnog zadatka.	Karlo Dimjašević	28.10.2022.
0.3	Uređena naslovna stranica dokumentacije. Dodani dionici i aktori. Dodani funkcionalni zahtjevi	Marko Prosenjak, Ivan Krešo	28.10.2022.
0.4	Dovršen Opis projektnog zadatka	Barbara Kralj	29.10.2022.
0.5	Dodan popis obrazaca uporabe.	Marin Čičak, Matea Kranjčić	01.11.2022.
0.5.1	Dodani opisi za obrasce uporabe UC1-8	Lovro Gaćina	05.11.2022.
0.5.2	Dodani opisi za obrasce uporabe UC16-19	Lovro Gaćina	06.11.2022.
0.5.3	Dodani opisi za obrasce uporabe UC11, 12, 21, 22	Marko Prosenjak	06.11.2022.
0.5.4	Dodani opisi za obrasce uporabe UC9, 10, 13-15	Barbara Kralj	06.11.2022.
0.5.5	Dodani obrasci uporabe (16 -> 16, 17)	Marin Čičak, Matea Kranjčić	07.11.2022.
0.6	Dodani dijagrami obrasca uporabe	Marin Čičak, Matea Kranjčić	07.11.2022.

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

Rev.	Opis promjene/dodatka	Autori	Datum
0.7	Dodan opis i relacijski dijagram baze podataka	Ivan Krešo	14.11.2022.
0.8	Dodan opis ostalih zahtjeva	Ivan Krešo	16.11.2022.
0.9	Dodan opis arhitekture i dizajna sustava	Barbara Kralj	17.11.2022.
0.10	Dodani dijagrami razreda	Barbara Kralj	17.11.2022.
0.11	Manje ispravke i dodaci u opisima UC-ova	Lovro Gaćina	18.11.2022.
0.12	Dodana skica dizajna homepagea u opis projektnog zadatka	Barbara Kralj	18.11.2022.
0.13	Dodani sekvencijski dijagrami za obrasce uporabe UC5 i UC17	Marin Čičak, Matea Kranjčić	18.11.2022.
0.13.1	Dodan sekvencijski dijagram za obrazac uporabe UC7	Lovro Gaćina	18.11.2022.
1.0	Ispravak gramatičkih grešaka i provjera dokumentacije	Ivan Krešo	18.11.2022.
1.1	Promijenjen relacijski dijagram i opis baze podataka	Ivan Krešo	06.01.2023.
1.2	Dodano poglavlje Zaključak i budući rad	Ivan Krešo	11.01.2023.
1.3	Korigirani dijagrami razreda	Barbara Kralj	11.01.2023.
1.4	Opisane korištene tehnologije i alati	Barbara Kralj	11.01.2023.
1.5	Ažuriran popis literature i dnevnik sastajanja	Marin Čičak	11.01.2023.
1.6	Dodan dijagram razmještaja	Marko Prosenjak	11.01.2023.

Nastavljeno na idućoj stranici

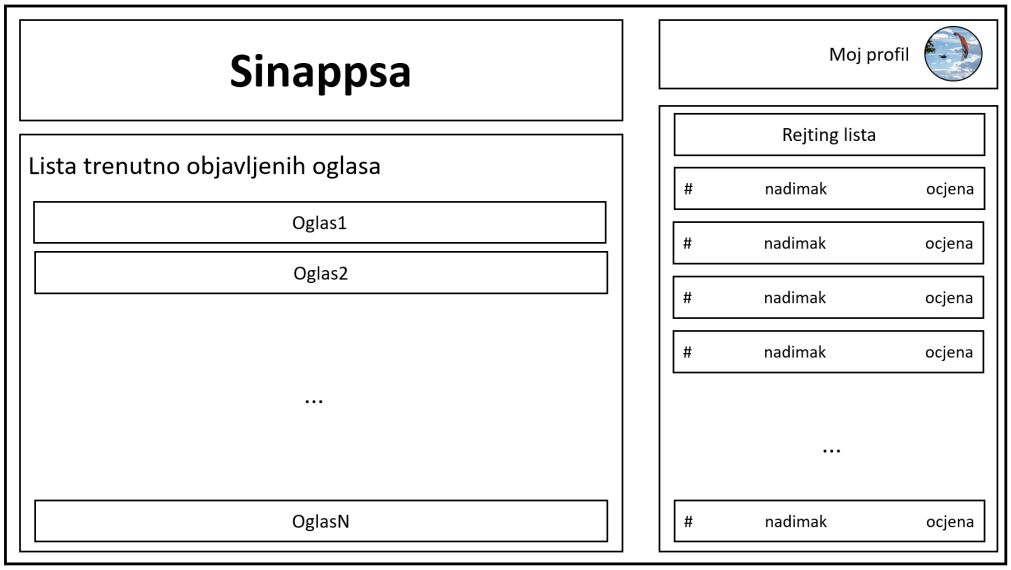
Nastavljeno od prethodne stranice

Rev.	Opis promjene/dodatka	Autori	Datum
1.7	Puštanje aplikacije u pogon	Karlo Dimjašević	12.01.2023.
1.8	Dodan JUnit test i izlaz testa za registraciju	Marin Čičak	12.01.2023.
1.9	Dorađen sekvencijski dijagram za obrazac uporabe UC5.	Marin Čičak	12.01.2023.
1.10	Ažurirana tablica aktivnosti	Marin Čičak	12.01.2023.
1.11	Dodani dijagrami aktivnosti i komponenti	Lovro Gaćina	12.01.2023.
1.12	Dodan Selenium test i izlaz testa za registraciju	Matea Kranjčić	12.01.2023.
1.13	Dodan dijagram stanja	Matea Kranjčić	13.01.2023.
1.14	Dodani dijagrami pregleda promjena	Barbara Kralj	13.01.2023.
1.15	Dorađeni sekvencijski dijagrami za obrasce uporabe UC7 i UC17	Marin Čičak	13.01.2023.
1.16	Dodani opisi dijagrama aktivnosti i komponenti	Lovro Gaćina	13.01.2023.
1.17	Dodan opis ispitivanja sustava	Matea Kranjčić	13.01.2023.
2.0	Konačna verzija		

2. Opis projektnog zadatka

Cilj ovog projekta je razviti programsku podršku za stvaranje web aplikacije „Sinappsa“ putem koje studenti mogu ponuditi ili tražiti pomoć oko specifičnog gradiva, laboratorijske vježbe ili kolegija općenito, a osim toga sadrži kompetitivni element zbog *rejting* liste najuspješnijih *studenata-pomagača*. Ideja za aplikaciju potekla je iz želje studenata da njihova pomoć drugim studentima bude opažena (reakcije na postove i slično).

Prilikom pokretanju aplikacije, prikazuje se lista trenutno objavljenih oglasa zajedno s *rejting* listom *studenata-pomagača*.



Slika 2.1: Skica dizajna homepagea

Aplikaciji mogu pristupiti tri vrste korisnika:

- neregistrirani korisnik
- registrirani korisnik
- moderator

Svaka vrsta ima vlastiti niz funkcionalnosti kojima se može koristiti i upravljati.

Neregistrirani korisnik u pokrenutom sustavu ima mogućnost vidjeti oglase i *rejting* liste, ali mu je onemogućeno kontaktiranje davatelja oglasa. Oglase je moguće filtrirati po nekoliko stavki. Postoji mogućnost filtriranja ovisno o smjeru:

- računarstvo
- elektrotehnika

Potom, oglase je također moguće filtrirati po kolegiju i kategoriji. Dostupne kategorije su sljedeće:

- laboratorijska vježba
- blic
- gradivo
- kontinuirani ispit
- ispitni rok

Neregistrirani korisnik obavlja registraciju unoseći podatke:

- ime
- prezime
- korisničko ime
- avatar
- službena FER e-mail adresa
- lozinka

Sustav će pri okončavanju prijave provjeriti je li upisana službena e-mail adresa u FER domeni. Korisnik će potom dobiti e-mail poruku. U toj e-mail poruci nalazit će se poveznica. Pritiskom na poveznicu, korisnik potvrđuje svoj identitet. Tim korakom registracija uspješno završava.

Registriranom korisniku, za razliku od neregistriranog korisnika, sustav nudi širi spektar funkcionalnosti. Takva najznačajnija funckionalnost je stvaranje i objavljivanje oglasa.

Prilikom stvaranja oglasa navodi se:

- naslov
- opis
- kolegij
- kategorija oglasa
  - oglas za traženje pomoći
  - oglas za pružanje pomoći

Oglas može biti aktivan i neaktivan. Inicijalno stanje oglasa je: aktivan. Oglas je moguće izmijeniti. Izmjenu može napraviti isključivo vlasnik oglasa. Također, oglas mora biti aktivan kako bi izmjena bila moguća. U izmjeni se mogu promijeniti naslov i opis oglasa. Korisnik, osim izmjene, može i obrisati svoj oglas.

Sljedeća funkcionalnost koja se nudi isključivo registriranom korisniku je odgovaranje na oglas drugih korisnika. Odabirom opcije odgovora na oglas, korisnik unosi proizvoljnu dodatnu poruku. Vlasnik oglasa prima e-mail s upitom i kontakt podacima zainteresiranog studenta. Daljna komunikacija i dogovori između studenata odvija se izvan aplikacije.

Osim e-maila, odgovor na oglas (upit) je također vidljiv u aplikaciji osobi koja ga je poslala i koja ga je primila. Korisniku se nudi prikaz svih njegovih trenutnih upita. Upit može biti jedno od sljedećeg stanja:

- u tijeku
- čeka ocjenjivanje
- prihvaćen
- odbijen

Inicijalno stanje upita je *u tijeku*. Upit je prihvaćen tek onda kada se instrukcije održe. Dakle, student koji traži pomoć ocjenjuje *student-pomagača* tek pri završetku instrukcija. Student može odbiti upit i prije završetka instrukcija. Odgovor na upit postavlja student koji je objavio oglas!

Student koji je nudio pomoć te čiji je rad pozitivno prihvaćen, biva ocijenjen od drugog studenta. Ta se ocjena pribraja u dosadašnje ocjene i računa se prosjek. Prosjeci ocjena svih korisnika spremaju se u rejting listi. Rejting lista prikazana u web aplikaciji sadrži 10 registriranih članova s najvećim rejtingom i ažurira se redovito.

Prijavljenom korisniku moguće je odabirom izbornika *Moj profil* vidjeti prikaz aktivnih i neaktivnih oglasa, trenutnih upita i informacija o profilu. Također, u istom izborniku korisnik može u svakom trenutku promijeniti lozinku, korisničko ime ili avatar.

Osim neprijavljenih i prijavljenih korisnika, uloga koja još postoji u sustavu je uloga moderatora. Ista ima za pravo uklanjanje nepravilnih i/ili neprikladnih oglasa. Nakon uklonjenog oglasa, korisniku se šalje e-mail s moderatorovim obrazloženjem zašto je oglas uklonjen. Moderator također može ručno dodavati kolegije ovisno o smjeru kojem pripadaju.

3. Specifikacija programske potpore

3.1 Funkcionalni zahtjevi

Dionici:

1. Korisnici (studenti)
   (a) Neregistrirani
   (b) Registrirani
2. Moderator
3. Razvojni tim

Aktori i njihovi funkcionalni zahtjevi:

1. Neregistrirani korisnik (inicijator) može:
   (a) Filtrirati oglase po smjeru (računarstvo ili elektrotehnika), po kolegiju i po kategoriji (laboratorijska vježba, blic, gradivo, kontinuirani ispit ili ispitni rok)
   (b) Pregledati listu trenutno objavljenih oglasa i rejting listu studenata-pomagača (pod nadimicima)
   (c) Registrirati se u aplikaciju za što su potrebni ime, prezime, korisničko ime, avatar i službena FER e-mail adresa te lozinka
2. Registrirani korisnik (inicijator) može:
   (a) Prijaviti se u aplikaciju, za što je potrebno unijeti registrirano korisničko ime te lozinku
   (b) Odjaviti se iz aplikacije
   (c) Objavljivati oglase, potrebno navesti nudi li ili traži pomoć, te još naslov, opis, kolegij i kategoriju
   (d) Odgovarati na oglase pri čemu može upisati dodatnu poruku
   (e) Pregledati vlastite oglase i upite
   (f) Izmijeniti vlastite podatke: korisničko ime, lozinku ili avatar
   (g) Obrisati svoj oglas

(h) Ocijeniti studenta-pomagača (ocjene od 1 do 10)

3. Moderator (inicijator) može:

   (a) Ukloniti nepravilne ili neprikladne oglase
   (b) Dodati kolegije, koje označava s obzirom na smjer (računarstvo ili elektrotehnika)

4. Baza podataka (sudionik) može:

   (a) Pohraniti sve podatke o korisnicima te njima pridruženim ovlastima (moderator ili registrirani korisnik)
   (b) Pohraniti sve podatke o oglasima i upitima
   (c) Pohraniti sve podatke o kolegijima i pripadajućim smjerovima

3.1.1 Obrasci uporabe

Opis obrazaca uporabe

UC1 - Pregled svih objavljenih oglasa

- **Glavni sudionik:** Korisnik, moderator
- **Cilj:** Pregledavanje trenutno objavljenih oglasa
- **Sudionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
  1. Trenutno objavljeni oglasi se prikazuju nakon učitavanja aplikacije

UC2 - Pregled rejting liste

- **Glavni sudionik:** Korisnik
- **Cilj:** Pregledavanje liste najbolje ocjenjenih korisnika
- **Sudionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
  1. Na početnoj stranici se korisnicima pored popisa objavljenih oglasa pokaže *rejting* lista *studenata-pomagača*

UC3 - Registracija korisnika

- **Glavni sudionik:** Korisnik
- **Cilj:** Stvaranje novog korisničkog računa
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik nije prijavljen
- **Opis osnovnog tijeka:**
  1. Korisnik odabire opciju registracije i preusmjerava se na stranicu za registraciju
  2. Korisnik unosi svoje ime, prezime, korisničko ime, službeni e-mail i lozinku te odabire avatar
  3. Šalje se poruka za potvrdu na unesenu e-mail adresu
  4. Korisnik klikom na link u poruci potvrđuje registraciju
  5. Korisnik se prijavljuje u novoregistrirani račun i preusmjeruje na početnu stranicu

- **Opis mogućih odstupanja:**
  - 2.a Unesena e-mail adresa nije na domeni FER-a
    1. Korisnik unosi novu e-mail adresu
  - 2.b Uneseno korisničko ime već pripada drugom računu
    1. Korisnik unosi novo korisničko ime
  - 2.c Korisnik unosi lozinku manju od pet znakova
    1. Korisnik unosi novu lozinku

UC4 - Filtriranje oglasa

- **Glavni sudionik:** Korisnik
- **Cilj:** Prikazati samo objave vezane uz odabrani smjer, kategoriju ili kolegij
- **Sudionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
  1. Korisnik klikne na gumb za filtriranje
  2. Korisnik odabire željeni smjer, kategoriju ili kolegij iz izbornika
  3. Na početnoj stranici prikazuju se samo oglasi za oznakom odabranog smjera, kategorije ili kolegija

UC5 - Prijava registriranog korisnika

- **Glavni sudionik:** Korisnik
- **Cilj:** Prijava korisnika u postojeći račun
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je registriran
- **Opis osnovnog tijeka:**
  1. Korisnik odabire opciju za prijavu
  2. Korisnik unosi korisničko ime i lozinku
  3. Korisnik je prijavljen i preusmjeruje se na početnu stranicu
- **Opis mogućih odstupanja:**
  - 2.a Uneseno je pogrešno korisničko ime ili lozinka
    1. Korisnik ispravlja korisničko ime ili lozinku

UC6 - Odjava registriranog korisnika

- **Glavni sudionik:** Korisnik
- **Cilj:** Odjaviti korisnika sa računa

- **Sudionici:** -
- **Preduvjet:** Korisnik je prijavljen
- **Opis osnovnog tijeka:**
  1. Korisnik bira opciju za odjavu
  2. Korisnik se odjavljuje i preusmjerava na početnu stranicu

UC7 - Stvaranje i objavljivanje oglasa

- **Glavni sudionik:** Korisnik
- **Cilj:** Stvoriti oglas i objaviti ga
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen
- **Opis osnovnog tijeka:**
  1. Korisnik odabire opciju pregleda vlastitih oglasa
  2. Korisniku se otvara popis vlastitih oglasa
  3. Korisnik odabire opciju ”Stvori oglas”
  4. Korisnik unosi naslov i opis, kolegij te kategoriju oglasa
  5. Korisnik odabire opciju ”Objavi oglas”
  6. Oglas se objavljuje i upisuje u bazu podataka

UC8 - Pregled vlastitih oglasa

- **Glavni sudionik:** Korisnik
- **Cilj:** Prikazati korisniku sve njegove oglase
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen
- **Opis osnovnog tijeka:**
  1. Korisnik odabire opciju za prikaz informacija o profilu
  2. Prikazuju se korisnički podaci (podaci o profilu)
  3. Korisnik odabire opciju ”Aktivni oglasi” ili ”Neaktivni oglasi”
  4. Korisniku se prikazuju svi njegovi aktivni ili neaktivni oglasi

UC9 - Mijenjanje vlastitih aktivnih oglasa

- **Glavni sudionik:** Korisnik
- **Cilj:** Izmjeniti podatke o oglasu
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen i ima aktivan oglas

- **Opis osnovnog tijeka:**
  1. Korisnik odabire posebni izbornik ”Moj profil”
  2. Korisniku se prikazuje popis vlastitih oglasa
  3. Korisnik odabire opciju izmjene oglasa
  4. Nakon mijenjanja potvrđuje izmjenu
  5. Promjene se upisuju u bazu podataka

UC10 - Brisanje vlastitih oglasa

- **Glavni sudionik:** Korisnik
- **Cilj:** Brisanje oglasa iz popisa vlastitih oglasa
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen
- **Opis osnovnog tijeka:**
  1. Korisnik odabire posebni izbornik ”Moj profil”
  2. Korisniku se prikazuje popis vlastitih oglasa
  3. Korisnik odabire opciju brisanja oglasa
  4. Oglas se briše iz baze podataka

UC11 - Odgovaranje na oglas

- **Glavni sudionik:** Korisnik
- **Cilj:** Odgovoriti na aktivni oglas drugog korisnika
- **Sudionici:** Baza podataka, drugi korisnik (autor oglasa)
- **Preduvjet:** Korisnik je prijavljen
- **Opis osnovnog tijeka:**
  1. Korisniku se prikazuju svi objavljeni oglasi
  2. Korisnik pronalazi aktivan oglas
  3. Korisnik odabire opciju odgovora na oglas
  4. Korisnik unosi proizvoljnu dodatnu poruku
  5. Autor objavljenog oglasa prima e-mail s upitom i kontakt podacima korisnika
  6. Upit se dodaje u posebni izbornik ”Moj profil” i korisniku i autoru oglasa
  7. Promjene se upisuju u bazu podataka

UC12 - Pregled trenutnih korisnikovih upita

- **Glavni sudionik:** Korisnik
- **Cilj:** Prikazati trenutne korisnikove upite na oglase
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen
- **Opis osnovnog tijeka:**
  1. Korisnik odabire opciju za prikaz informacija o profilu
  2. Korisnik odabire opciju za prikaz svojih upita
  3. Prikazuju se svi korisnikovi upiti na oglase drugih korisnika

UC13 - Pregled trenutnih upita korisniku

- **Glavni sudionik:** Korisnik
- **Cilj:** Prikazati trenutne upite na korisnikove oglase
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen
- **Opis osnovnog tijeka:**
  1. Korisnik odabire opciju za prikaz informacija o profilu
  2. Korisnik odabire opciju za prikaz upita na svoje oglase
  3. Prikazuju se svi upiti na korisnikove oglase

UC14 - Odgovaranje na upit

- **Glavni sudionik:** Korisnik (autor oglasa)
- **Cilj:** Odgovoriti na upit, tj. promijeniti njegovo stanja
- **Sudionici:** Drugi korisnik (autor upita), baza podataka
- **Preduvjet:** Autor oglasa je prijavljen i ima objavljen oglas za koji je postavljen upit
- **Opis osnovnog tijeka:**
  1. Autor oglasa odabire oglas i upit
  2. Autor oglasa postavlja novo stanje upita (*prihvaćen* ili *odbijen*)

UC15 - Ocjenjivanje student-pomagača

- **Glavni sudionik:** Korisnik koji je zatražio pomoć (Autor oglasa ili upita)
- **Cilj:** Ocijeniti korisnika koji je pružao pomoć
- **Sudionici:** Student-pomagač (Autor oglasa ili upita), baza podataka
- **Preduvjet:** Glavni korisnik je registriran, upit je postavljen u stanje *prihvaćen*
- **Opis osnovnog tijeka:**

1. Glavni korisnik ocjenjuje studenta-pomagača brojčanom ocjenom
2. Ažurira se rejting lista studenata

UC16 - Pregled korisnikovih podataka

- **Glavni sudionik:** Korisnik
- **Cilj:** Pregledati podatke korisničkog profila
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen
- **Opis osnovnog tijeka:**
  1. Korisnik odabire opciju za prikaz informacija o profilu
  2. Prikazuju se informacije o korisničkom profilu (ime, prezime, korisničko ime, avatar, e-mail adresa)

UC17 - Izmjena korisnikovih podataka

- **Glavni sudionik:** Korisnik
- **Cilj:** Izmijeniti podatke korisničkog profila
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen
- **Opis osnovnog tijeka:**
  1. Korisnik odabire opciju za prikaz informacija o profilu
  2. Prikazuju se informacije o korisničkom profilu (ime, prezime, korisničko ime, avatar, e-mail adresa)
  3. Korisnik odabire opciju za promjenu informacija
  4. Korisnik unosi novo korisničko ime, lozinku i/ili avatar
  5. Informacije o profilu se izmijenjuju
- **Opis mogućih odstupanja:**

  4.a Uneseno korisničko ime već pripada drugom računu
    1. Korisnik unosi novo korisničko ime

UC18 - Uklanjanje neprikladnih i nepravilnih oglasa

- **Glavni sudionik:** Moderator
- **Cilj:** Uklanjanje neprikladnih i nepravilnih oglasa iz aplikacije
- **Sudionici:** Baza podataka, korisnik
- **Preduvjet:** Moderator je prijavljen te je oglas objavljen (postoji u bazi podataka)

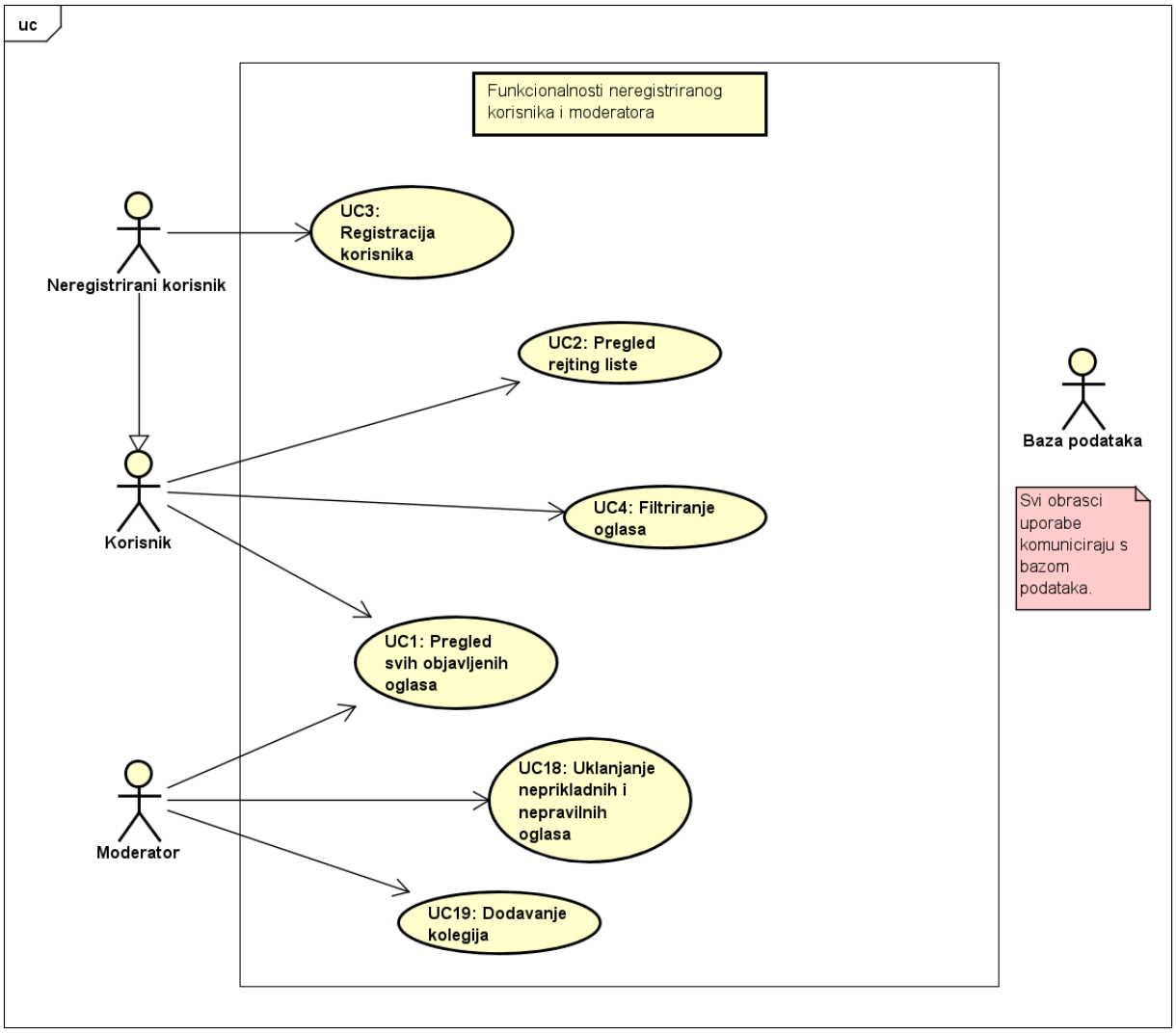
- **Opis osnovnog tijeka:**
  1. Moderator odabire opciju ”Ukloni oglas” (gumb koji se nalazi pored oglasa)
  2. Moderator navodi razlog brisanja oglasa
  3. Nakon što je moderator naveo razlog brisanja, gumb ”Ukloni” postaje aktivan (enabled)
  4. Moderator pritiskom na gumb ”Ukloni” briše oglas
  5. Korisniku čiji je oglas obrisan šalje se e-mail s objašnjenjem moderatora
  6. Oglas se uklanja iz baze podataka te više nije vidljiv u aplikaciji

UC19 - Dodavanje kolegija

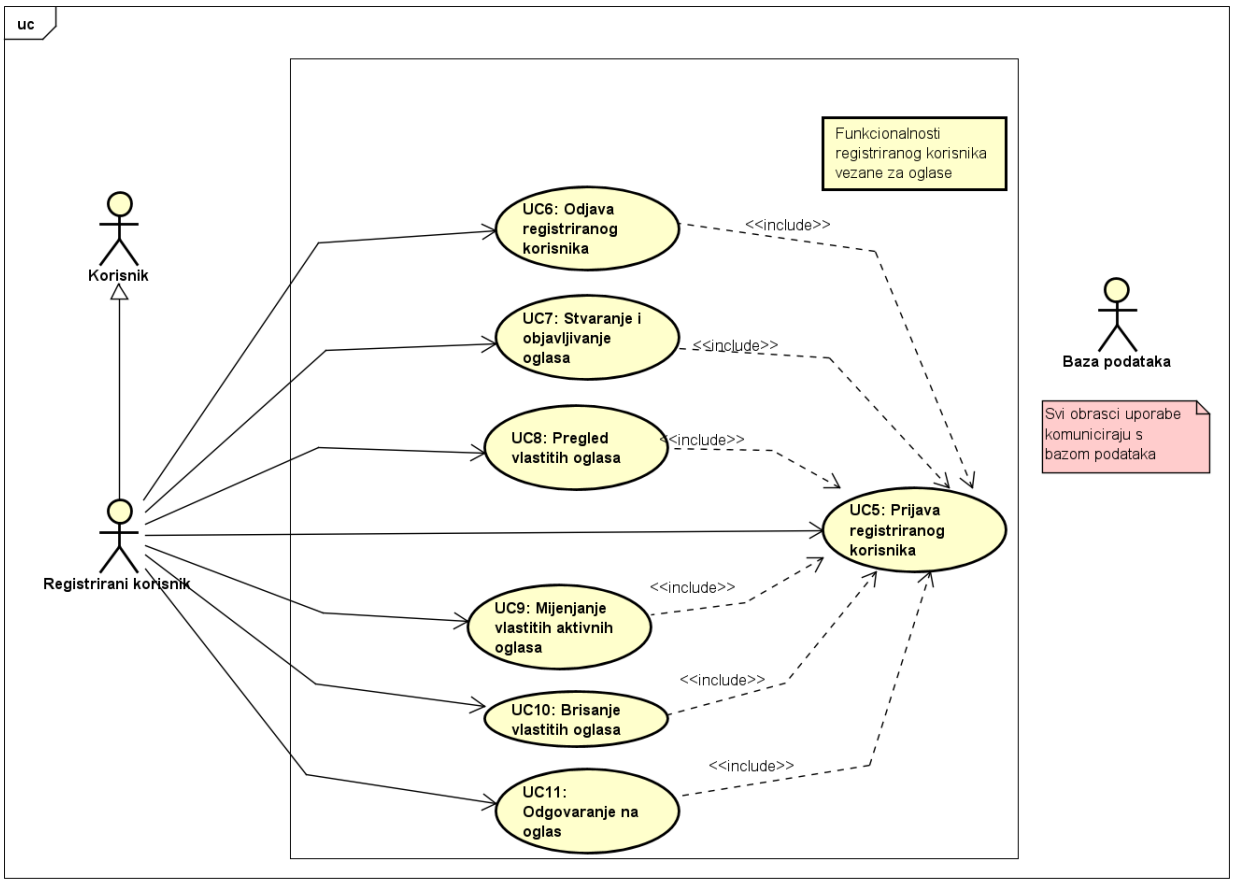
- **Glavni sudionik:** Moderator
- **Cilj:** Dodavanje kolegija u aplikaciju
- **Sudionici:** Baza podataka
- **Preduvjet:** Moderator je prijavljen
- **Opis osnovnog tijeka:**
  1. Moderator odabire opciju ”Dodaj kolegij”
  2. Moderator navodi ime i smjer kolegija
  3. Nakon što je moderator naveo ime kolegija i smjer, gumb ”Dodaj” postaje aktivan (enabled)
  4. Moderator pritiskom na gumb ”Dodaj” dodaje kolegij
  5. Kolegij se dodaje u bazu podataka i postaje vidljiv u aplikaciji
- **Popis mogućih odstupanja:**

  4.a Kolegij postoji u sustavu
  - (a) Sustav obavještava moderatora o neuspjelom dodavanju kolegija

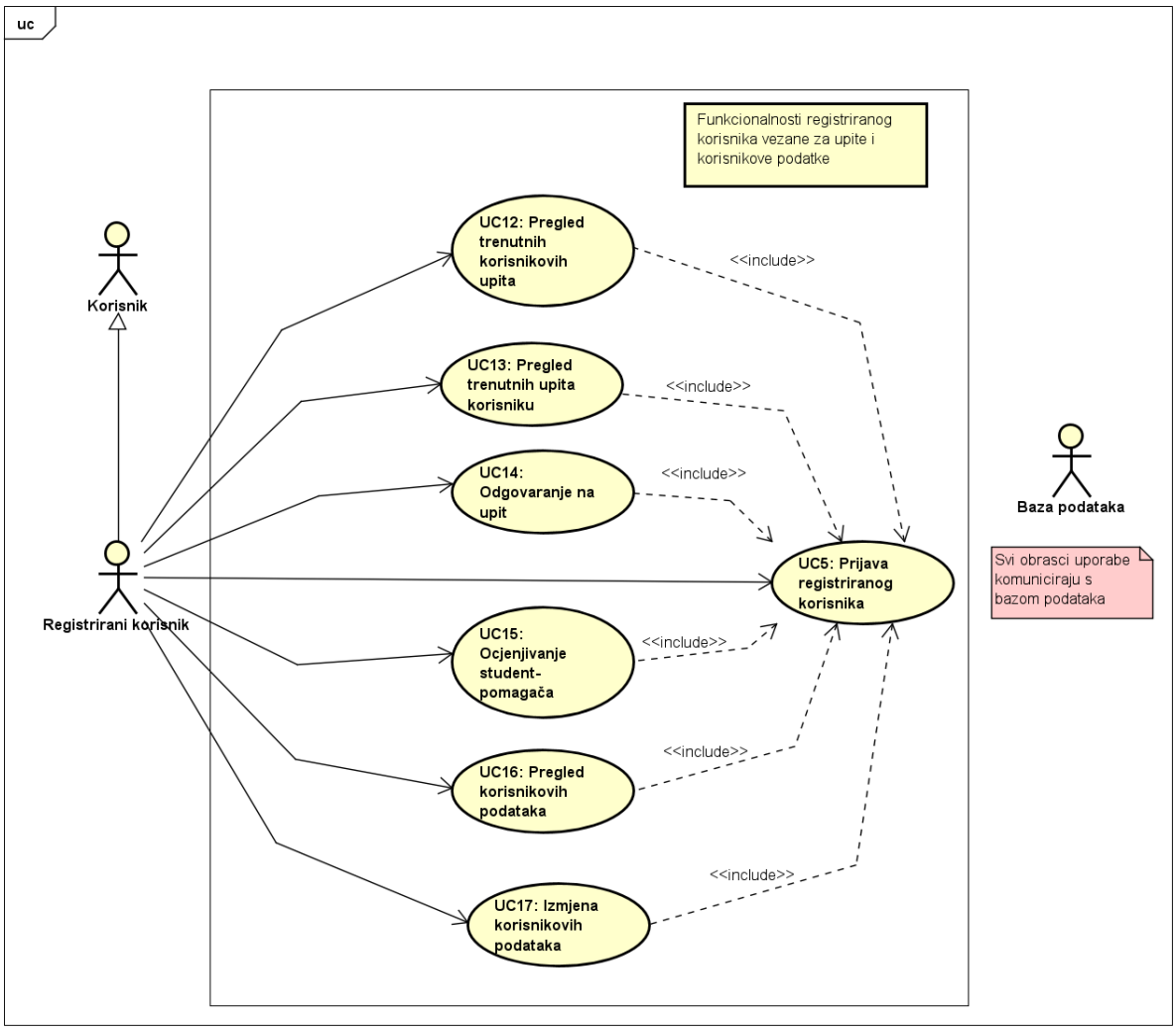
Dijagrami obrazaca uporabe



Slika 3.1: Dijagram obrasca uporabe, funkcionalnosti neregistriranog korisnika i moderatora



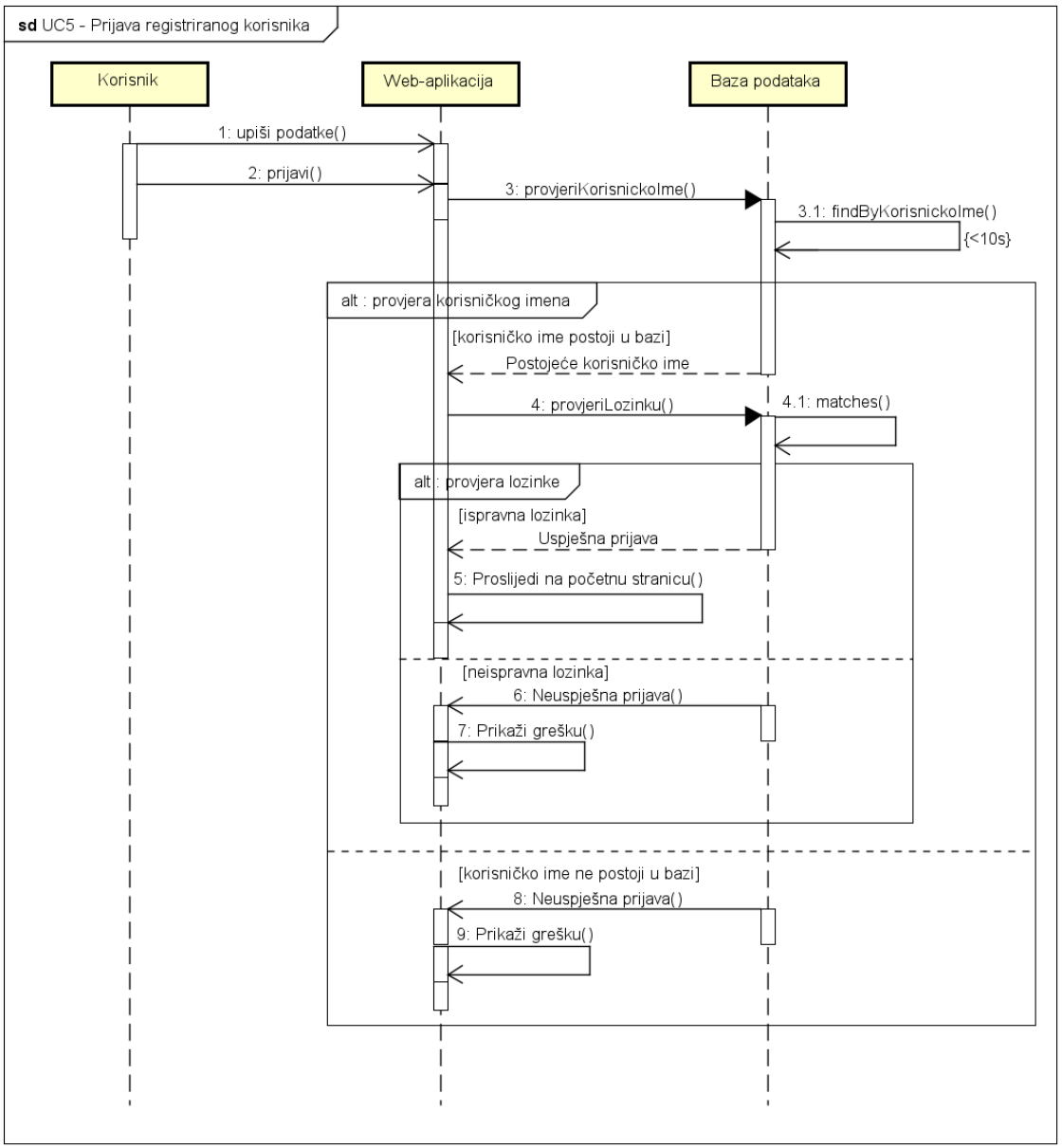
Slika 3.2: Dijagram obrasca uporabe, funkcionalnosti registriranog korisnika vezane uz oglase



Slika 3.3: Dijagram obrasca uporabe, funkcionalnosti registriranog korisnika vezane uz upite

3.1.2 Sekvencijski dijagrami

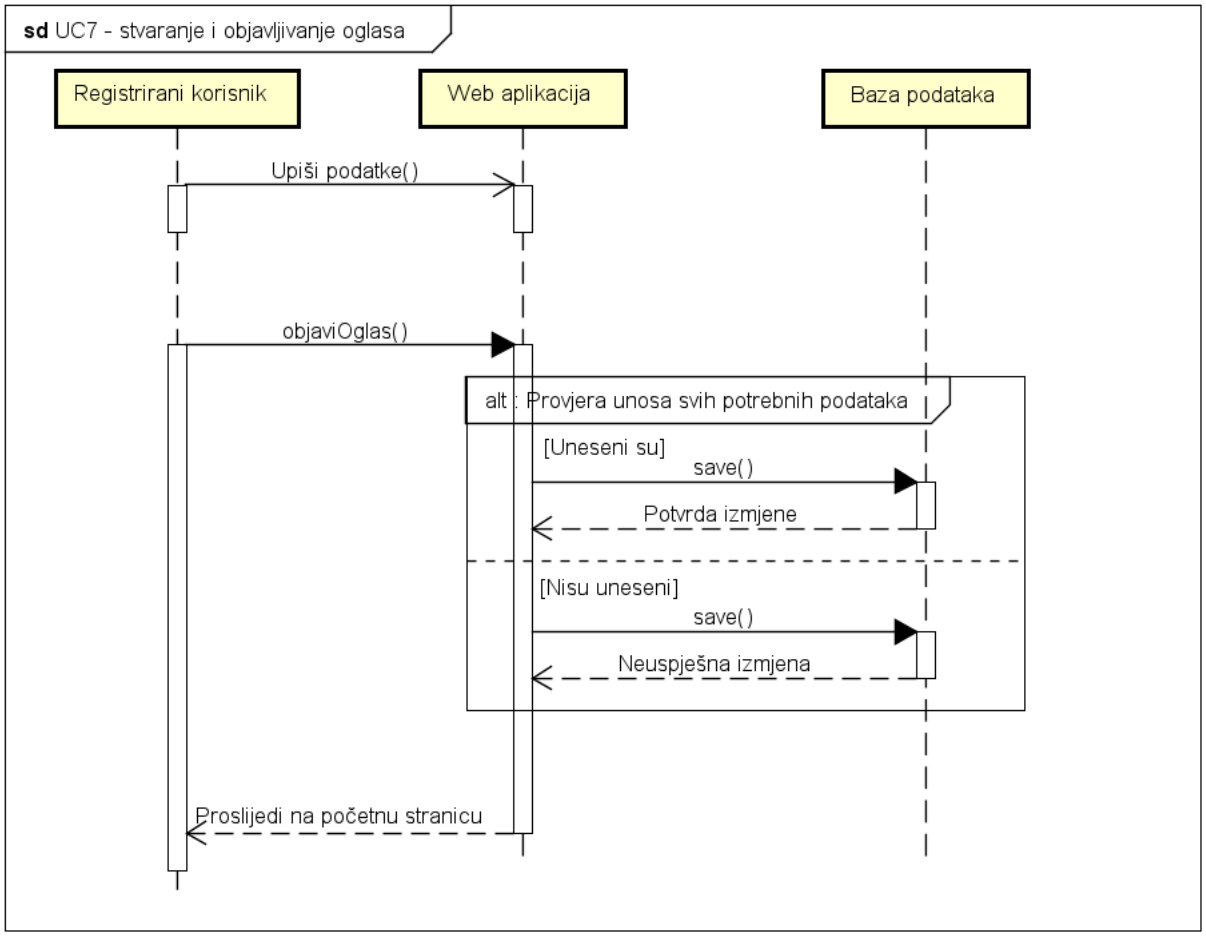
Obrazac uporabe UC5 - Prijava registriranog korisnika.
Korisnik će prilikom odabira funkcionalnosti prijave biti preusmjeren na formu gdje će trebati upisati valjano korisničko ime i ispravnu lozinku. Podaci se šalju na poslužitelj, te dalje se provjeravaju s podacima u bazi podataka. Korisnik će biti informiran o pogrešnom upisivanju korisničkog imena i/ili lozinke. U tom slučaju korisnik i dalje ostaje na stranici za prijavu. Nakon uspješne prijave, bit će preusmjeren na početnu stranicu.



Slika 3.4: Sekvencijski dijagram za UC5

Obrazac uporabe UC7 - Stvaranje i objavljivanje oglasa

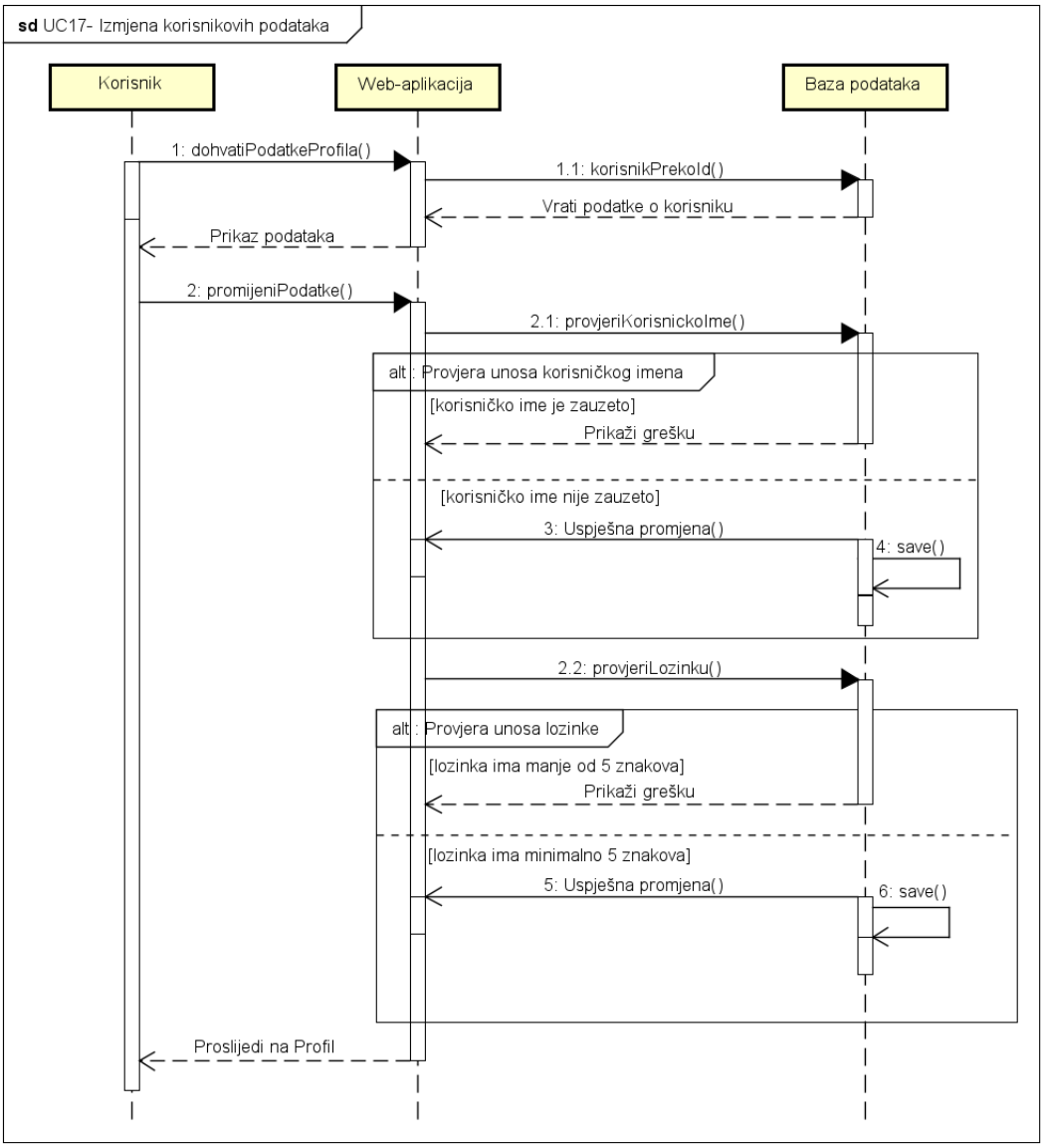
Korisnik odabire opciju ”Kreiraj oglas”, koja se nalazi na početnoj stranici. Opcija ga vodi na formu gdje korisnik unosi tražene podatke: naslov, opis, kolegij, kategoriju te parametar nudi li ili traži pomoć. Podatke šalje poslužitelju, koji provjerava jesu li ispravno odabrani i uneseni svi podaci. Nakon toga poslužitelj sprema oglas u bazu podataka i proslijeđuje korisnika na početnu stranicu. Oglas se sada vidi na početnoj stranici, a također i na stranici ”Profil”.



Slika 3.5: Sekvencijski dijagram za UC7

Obrazac uporabe UC17 - Izmjena korisnikovih podataka

Korisnik odabire opciju prikaza informacija o profilu. Poslužitelj dohvaća iz baze podataka podatke o korisniku i prikazuje mu ih na stranici. Korisnik bira opciju za izmjenu vlastitih podataka. Upisuje novo korisničko ime, lozinku i/ili avatar. Podaci se šalju na poslužitelj gdje se provjerava postoji li već uneseno korisničko ime. Također se provjerava odgovara li nova unesena lozinka zahtjevima sustava (minimalni broj znakova). Nakon provjere, poslužitelj korisniku vraća poruku o grešci, ako je uneseno postojeće korisničko ime ili prikazuje uspješnu izmjenu podataka. Podaci se ažuriraju u bazi podataka nakon uspješno izvršene izmjene.



Slika 3.6: Sekvencijski dijagram za UC17

3.2 Ostali zahtjevi

- Korisničko sučelje i sustav moraju podržavati hrvatsku abecedu (uključujući dijakritičke znakove) pri unosu podataka te prikazu podataka
- Aplikacija mora biti prilagođena za različite veličine ekrana
- Izvršavanje upita prema bazi, kao što su dohvaćanje, unos ili mijenjanje podataka ne smije trajati duže od nekoliko sekundi
- Sustav treba biti jednostavan za korištenje, nitko ne bi trebao imati problema sa snalaženjem po korisničkom sučelju
- Implementacija dodatnih funkcionalnosti ne smije narušiti postojeće funkcionalnosti sustava
- Lozinke korisnika ne bi smjele u bazi podataka biti spremljene u njihovom izvornom obliku, nego moraju biti enkriptirane da bi se osigurala njihova privatnost
- Korisnik ne smije moći pomoću korisničkog sučelja učiniti nikakvu akciju koja bi dovela do greške tj. iznimke u programskom kodu niti narušila očuvanost i pouzdanost podataka u bazi podataka
- Sustav mora biti implementiran kao web aplikacija poštujući objektno-orijentiranu paradigmu
- Arhitektura sustava mora biti smislena i pregledna

4. Arhitektura i dizajn sustava

Arhitektura je podijeljena na tri dijela:

- Internetski preglednik
- Web poslužitelj
- Baza podataka

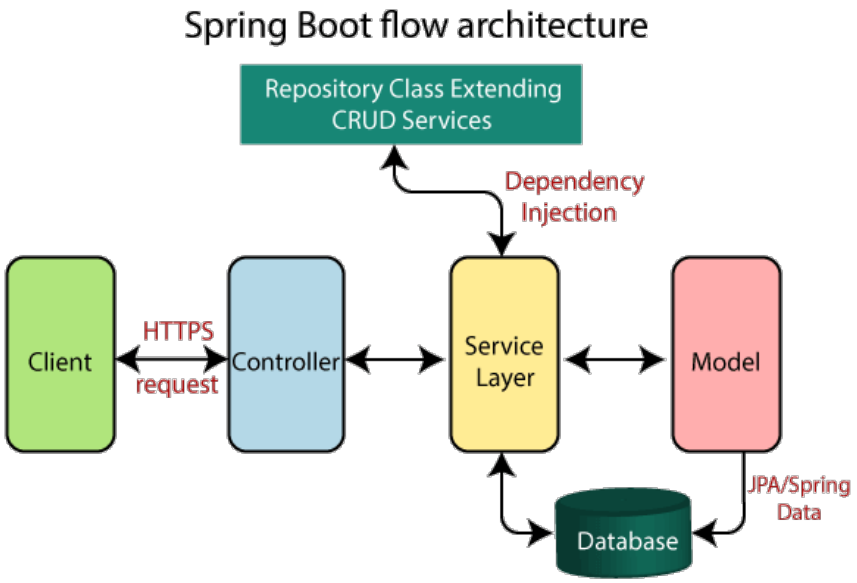
Internetski preglednik odnosno web preglednik je program koji korisniku omogućuje pregled web stranica i multimedijalnih sadržaja vezanih uz njih. Svi internetski preglednici su i grafički programi, jer osim teksta mogu prikazivati i vizualne sadržaje. Korisnik putem preglednika šalje zahtjev web poslužitelju.

Web poslužitelj je namjensko računalo ili softver koje šalje i prima podatke od mnogostrukih klijenata te je osnova za rad web aplikacije. Za komunikaciju koristi http protokol odnosno prenosi informacije na webu.

Web aplikacija je program koji se izvodi u internet pregledniku. Ona obrađuje korisnikove zahtjeve. Ovisno o zahtjevu, pristupa bazi podataka te preko poslužitelja vraća korisniku HTML odgovor vidljiv u web pregledniku.

Programski jezik koji smo odabrali za izradu backend dijela naše web aplikacije je Java, preciznije Spring Boot razvojni okvir, za frontend koristimo programski jezik JavaScript, preciznije React razvojni okvir. Razvojna okruženja koja koristimo su IntelliJ IDEA Ultimate za Spring te Microsoft Visual Studio Code za React.

Arhitektura sustava se temelji na MVC arhitekturi. Spring ima razvijenu podršku za MVC arhitekturu i malo različitu terminologiju. Na slici se vide ključni elementi arhitekture.



Slika 4.1: Spring Boot Arhitektura

- **Model** - Sadrži opise entiteta koji se automatski preslikaju u tablice u bazi podataka. U tome nam pomažu Spring Boot notacije kao što su @Entity (označava tablicu), @Column (atribut entiteta, tj. stupac tablice), @Id (primarni ključ) i slični.
- **Controller** - Zadužen za svu komunikaciju sa Web klijentom, to ukjučuje primanje svih HTTP zahtjeva, koje onda šalje dalje (prema *Service*-ima) te od njih dobiva odgovor i onda odgovara natrag Web klijentu. Također se brine za autentikaciju, ne mogu svi korisnici sustava slati sve instance HTTP zahtjeva.
- **Service** - Najvažniji dio sustava, u ovom dijelu se nalazi sva poslovna logika. Prima pozive od *Controller*-a, radi sve potrebne provjere. Po potrebi mijenja podatke u bazi što postiže zvanjem metoda *Repository*-a.
- **Repository** - Sve promjene (unos, brisanje ili mijenjanje) podataka u bazi idu preko *Repository* dijela. Tu se pišu sve SQL naredbe koje onda *Service* može zvati.

4.1 Baza podataka

Podatke u našem sustavu trebamo negdje pamtiti te imati mogućnost čitanja, dodavanja, izmjene i brisanja tih podataka. Da bi to postigli, odlučili smo koristiti

relacijsku bazu podataka. Sastoji se od 4 entiteta, tj. tablice, svaka ta tablica ima svoje atribute te su tablice međusobno povezane stranim ključevima.

Baza podataka se sastoji od sljedećih entiteta:

- RegistriraniKorisnik
- Oglas
- Upit
- Kolegij

4.1.1 Opis tablica

RegistriraniKorisnik Ovaj entitet sadrži informacije o studentima koji su registrirani korisnici te o moderatoru. Sadrži atribute: id, korisnickoIme, ime, prezime, avatar, email, lozinka, moderator, brojPrimljenihRecenzija, sumaPrimljenihRecenzija, verificationCode, enabled. Ovaj entitet je u vezi *One-to-Many* sa entitetima Oglas i Upit preko id-a korisnika.

Atribut	Tip	Opis atributa
id	LONG	Jedinstveni brojčani identifikator
korisnickoIme	VARCHAR	Jedinstveno korisničko ime korisnika
ime	VARCHAR	Ime korisnika
prezime	VARCHAR	Prezime korisnika
avatar	VARCHAR	Ime slike koja se koristi kao avatar
email	VARCHAR	Jedinstvena e-mail adresa korisnika
lozinka	VARCHAR	Lozinka korisnika
moderator	BOOLEAN	True označava da je korisnik moderator, false da nije
brojPrimljenihRecenzija	INT	Broj koliko je puta korisnik ocjenjivan od drugih korisnika
sumaPrimljenihRecenzija	INT	Zbroj svih ocjena kojima je korisnik ocijenjen
verificationCode	VARCHAR	Jedinstveni verifikacijski kod (niz slova) korisnika
enabled	BOOLEAN	True ako je korisnički račun aktiviran, false ako nije

Oglas Ovaj entitet sadrži informacije o svim oglasima. Sadrži atribute: id, naslov, opis, kolegij, kategorija, idKreatora, aktivan i trazimPomoc. Entitet je u vezi

One-to-Many sa entitetom Upit preko id-a oglasa. Osim toga, u vezi je *Many-to-One* sa entitetom RegistriraniKorisnik preko idKreatora, te je u vezi *Many-to-One* sa entitetom Kolegij preko kolegij atributa.

Atribut	Tip	Opis atributa
id	LONG	Jedinstveni brojčani identifikator
naslov	VARCHAR	Naslov oglasa
opis	VARCHAR	Kratki opis oglasa
kolegij	VARCHAR	Ime kolegija na kojeg se oglas odnosi (Kolegij.ime)
kategorija	KATEGORIJA	Poprima jednu od pet mogućih vrijednosti (LABOS, BLIC, GRADIVO, ISPITNI_ROK, KONTINUIRANI_ISPIT)
idKreatora	LONG	Id registriranog korisnika koji stvorio ovaj oglas (RegistriraniKorisnik.id)
aktivan	BOOLEAN	True ako je oglas i dalje aktivan, false ako nije
trazimPomoc	BOOLEAN	True ako je oglas kreiran od korisnika koji traži pomoć, false ako kreator nudi pomoć

Upit Ovaj entitet sadrži informacije o svim odgovorima na oglas, tzv. upitima. Sadrži atribute: id, idAutoraUpita, idOglasa, poruka, stanje i ocjena. Entitet je u vezi *Many-to-One* sa entitetom RegistriraniKorisnik preko idAutoraUpita. Također je u *Many-to-One*e odnosu sa entiteom Oglas preko atributa idKreatora.

Atribut	Tip	Opis atributa
id	LONG	Jedinstveni brojčani identifikator
idAutoraUpita	LONG	Identifikator korisnika koji je objavio ovaj upit (RegistriraniKorisnik.id)
idOglasa	LONG	Identifikator oglasa na kojeg se ovaj upit odnosi (Oglas.id)
poruka	VARCHAR	Kratka poruka autora upita koja se

Nastavljeno na idućoj stranici

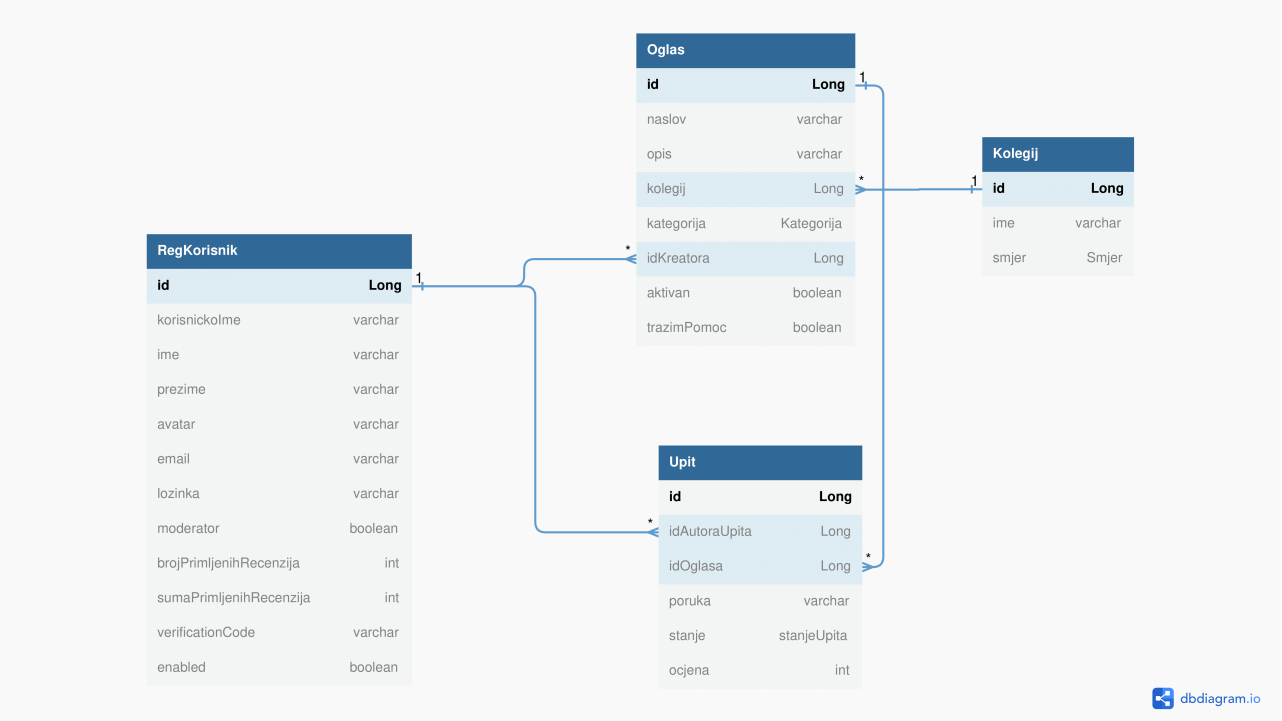
Nastavljeno od prethodne stranice

Atribut	Tip	Opis atributa
		šalje autoru Oglasa
stanje	STANJEUPITA	Poprima jednu od četiri moguće vrijednosti (U_TIJEKU, PRIHVACEN, ODBIJEN, CEKA_OCJENJIVANJE)
ocjena	INT	Ocjena kojom je korisnik koji je primio pomoć ocijenio korisnika koji nudi pomoć, poprima vrijednost null ako upit još nije u stanju PRIHVACEN

Kolegij Ovaj entitet sadrži informacije o svim kolegijima. Sastoji se od 3 atributa: id, ime i smjer. Entitet je u vezi *One-to-Many* sa entitetom Oglas preko atributa ime.

Atribut	Tip	Opis atributa
id	LONG	Jedinstveni brojčani identifikator
ime	VARHCHAR	Ime kolegija
smjer	SMJER	Smjer kolegija (R ili E)

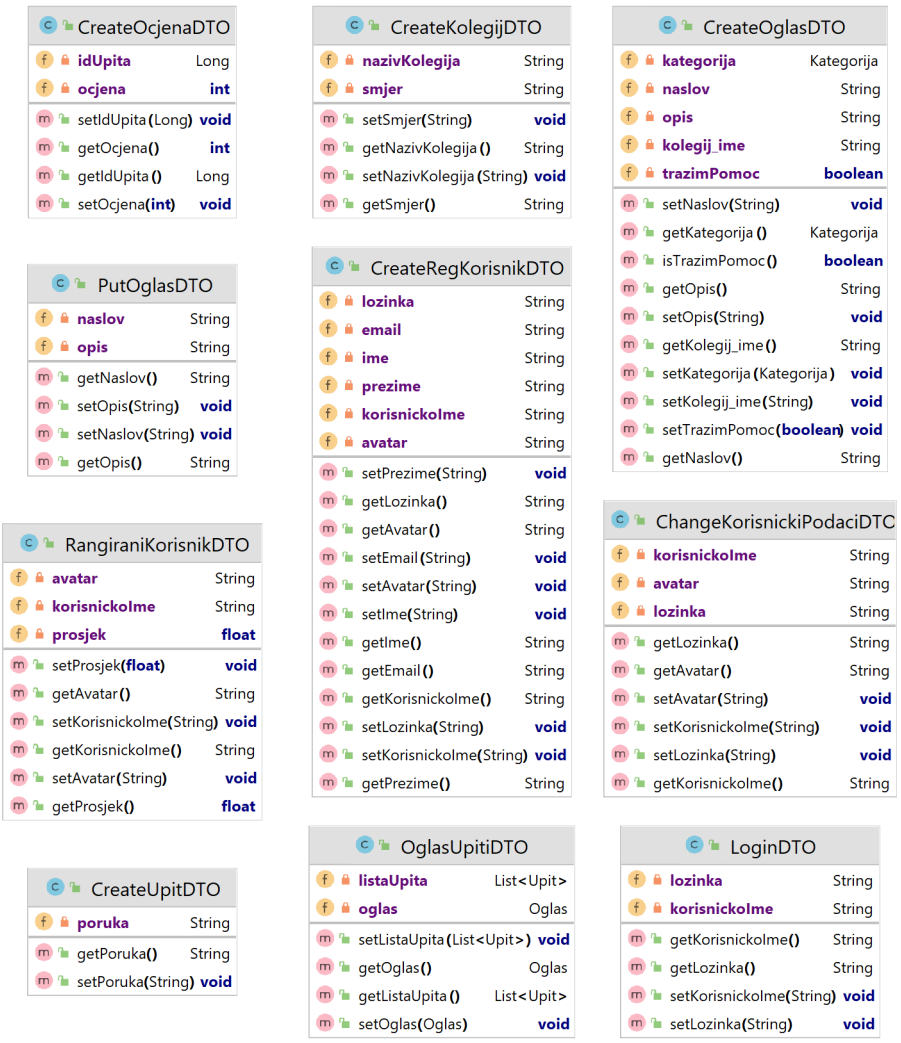
4.1.2 Dijagram baze podataka



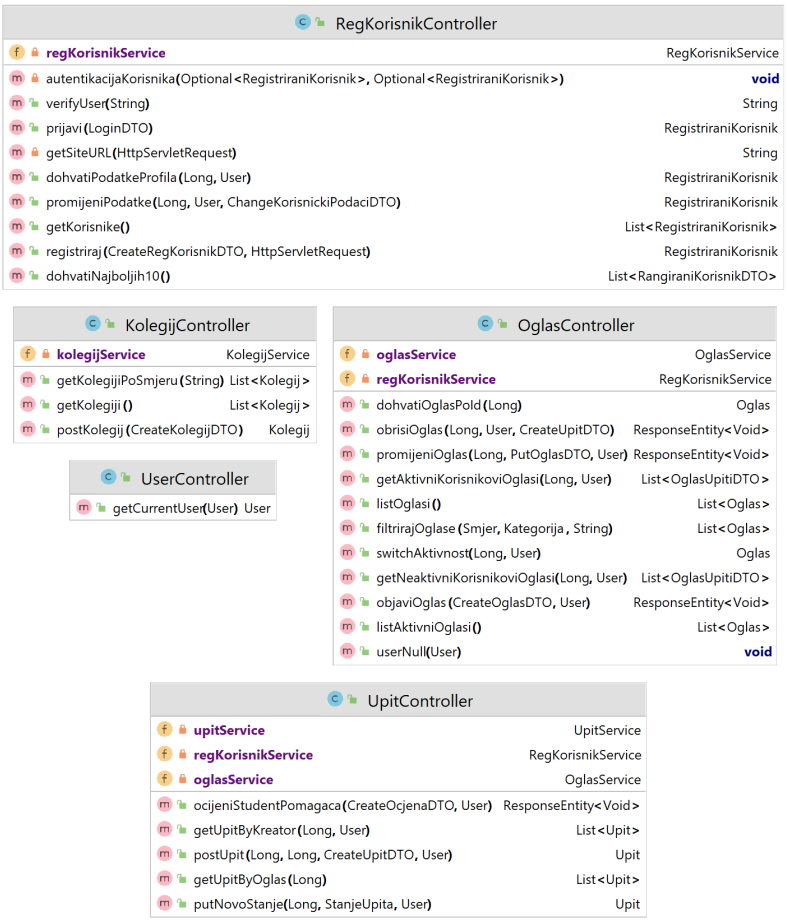
Slika 4.2: Relacijski dijagram baze podataka

4.2 Dijagram razreda

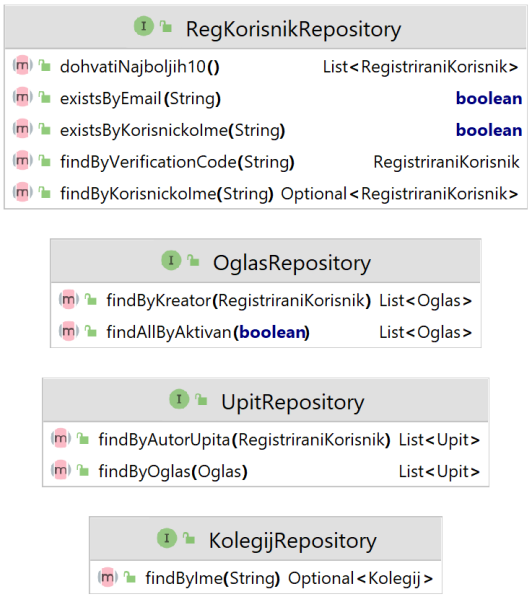
Slika 4.3: Dijagram razreda - dio Models



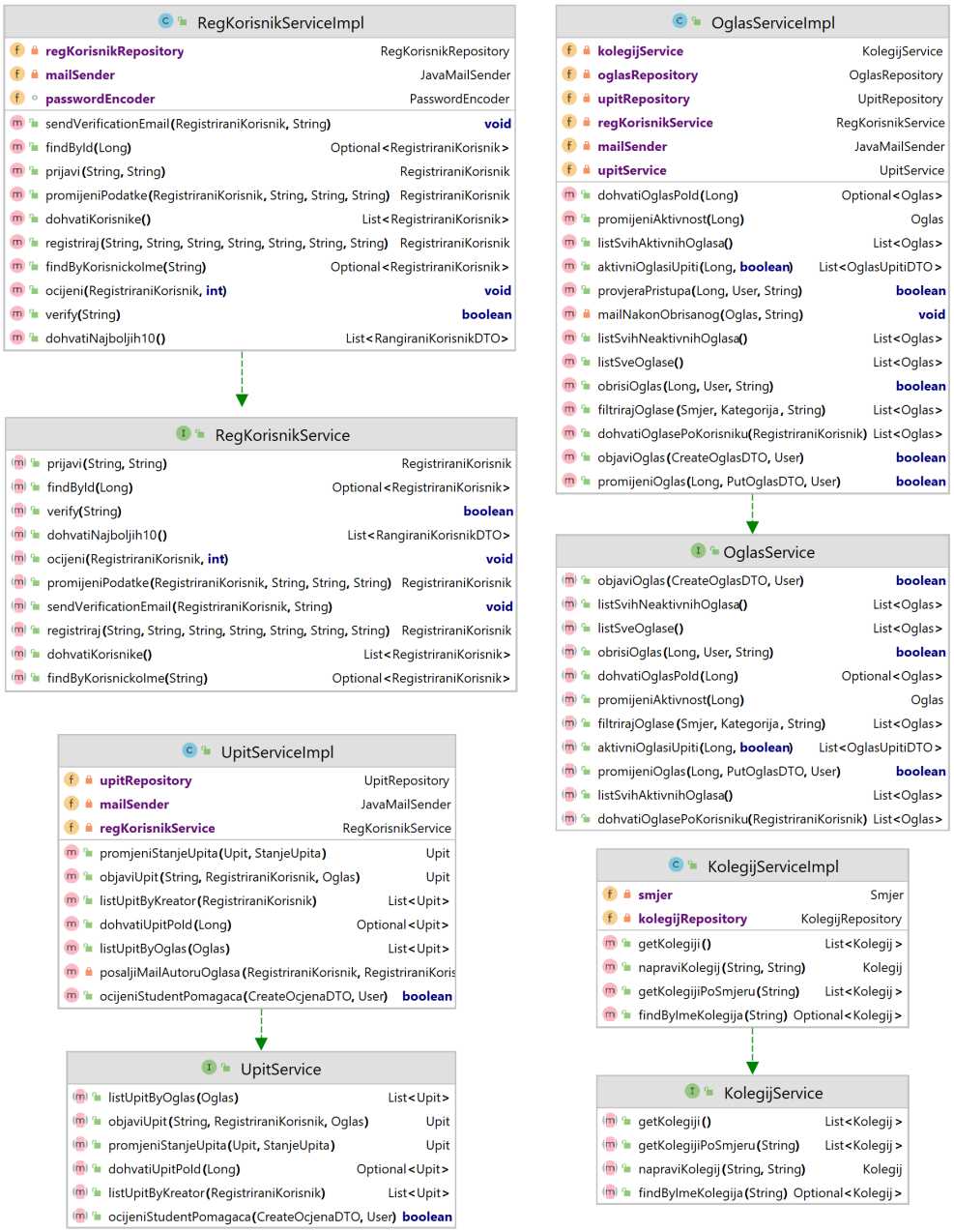
Slika 4.4: Dijagram razreda - dio Data Transfer Objects



Slika 4.5: Dijagram razreda - dio Controllers



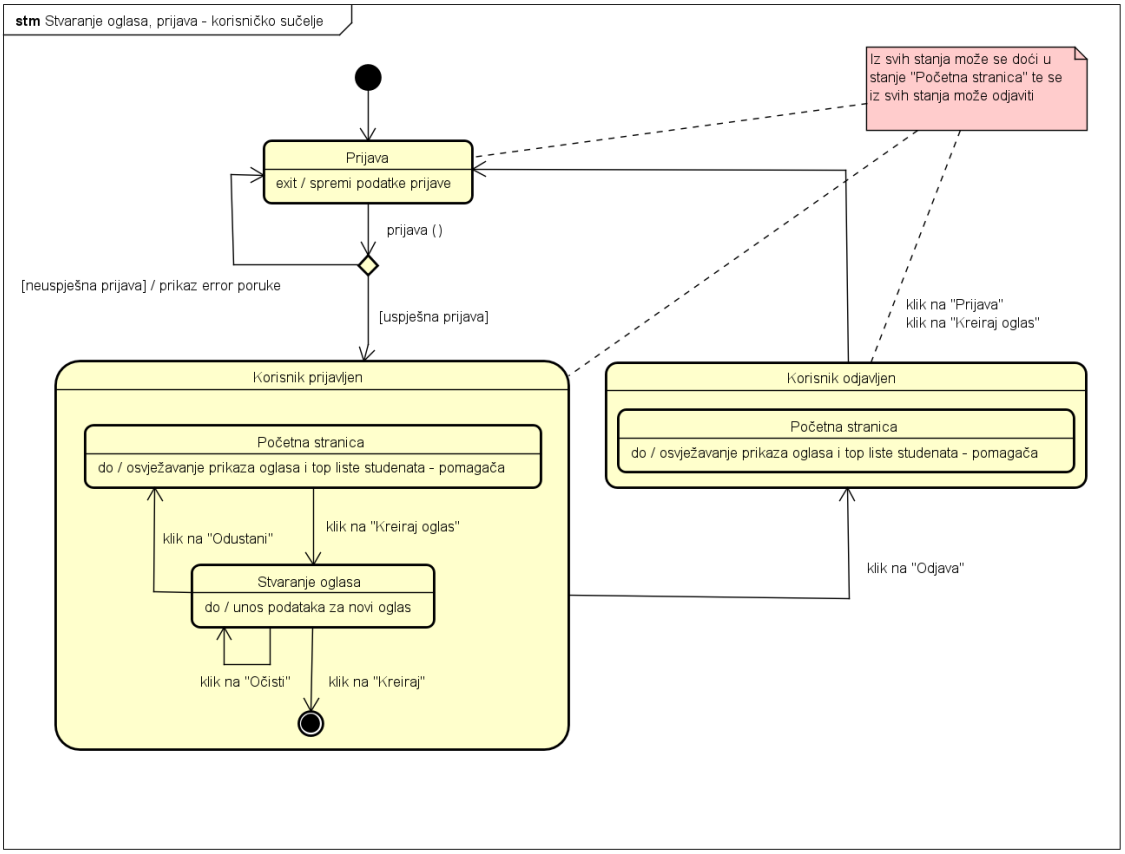
Slika 4.6: Dijagram razreda - dio Repositories



Slika 4.7: Dijagram razreda - dio Services

4.3 Dijagram stanja

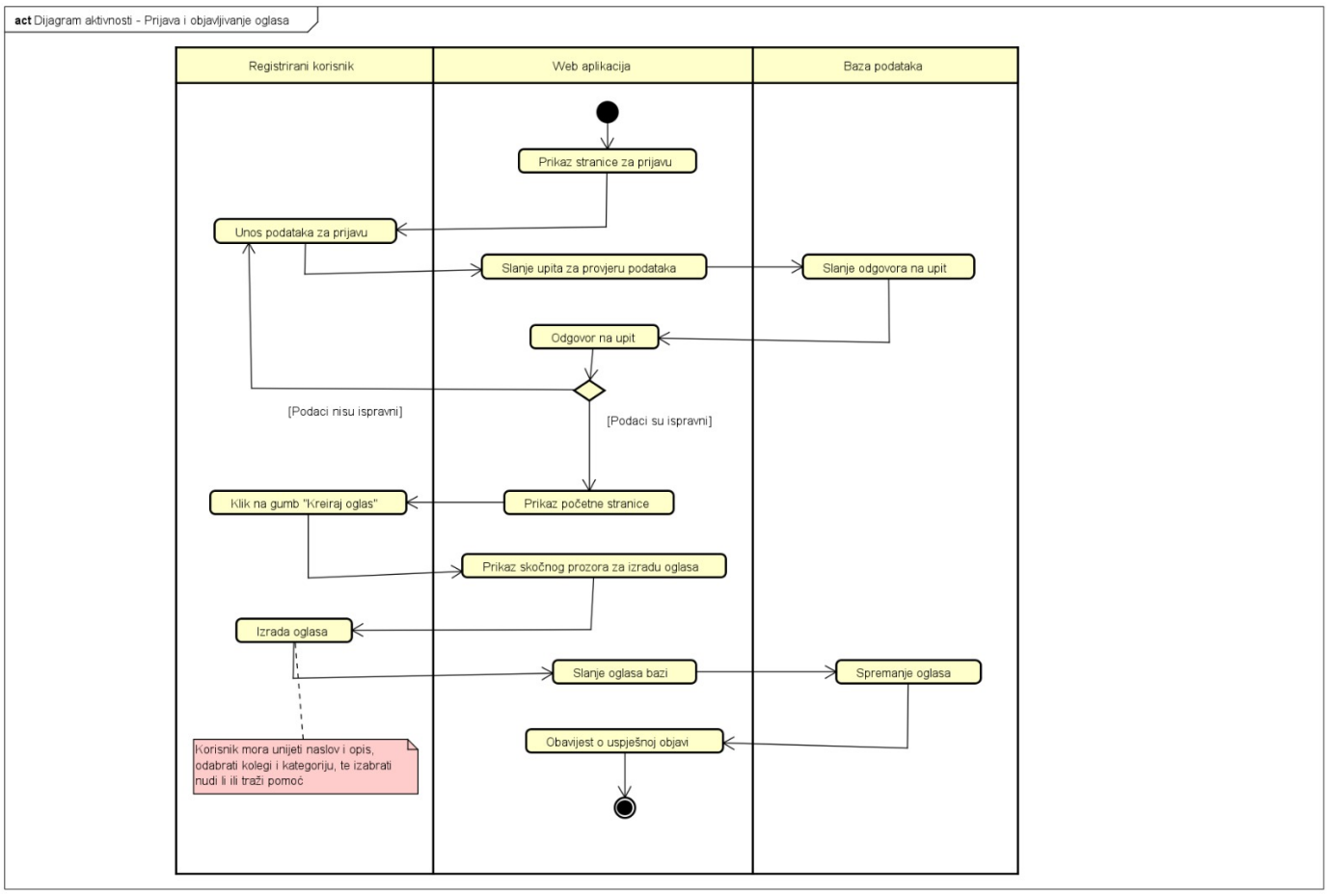
Dijagram stanja prikazuje stanja objekta te prijelaze iz jednog stanja u drugo temeljene na događajima. Na slici 4.8 prikazan je dijagram stanja korisničkog sučelja te tijek jedne od ključnih funkcionalnosti - stvaranje oglasa. Neprijavljenom korisniku omogućen je prikaz trenutno aktivnih oglasa te rang liste studenata-pomagača. Prijavljeni korisnik, uz funkcionalnosti neprijavljenog korisnika, može stvoriti oglas klikom na gumb ”Kreiraj oglas”. Tada se prikazuje forma za unos podataka o oglasu. Nakon svih unesenih podataka i provjera, korisnik klikom na gumb ”Kreiraj” objavljuje oglas, a ako odluči kliknuti na gumb ”Očisti” dosad uneseni podaci se brišu te je prazna forma ponovno spremna za unos podataka. Također u svakom trenutku prilikom unosa podataka može se i odustati od kreiranja oglasa. Za to je predviđen gumb ”Odustani” koji vraća korisnika na početnu stranicu.



Slika 4.8: Dijagram stanja - stvaranje oglasa i prijava korisnika

4.4 Dijagram aktivnosti

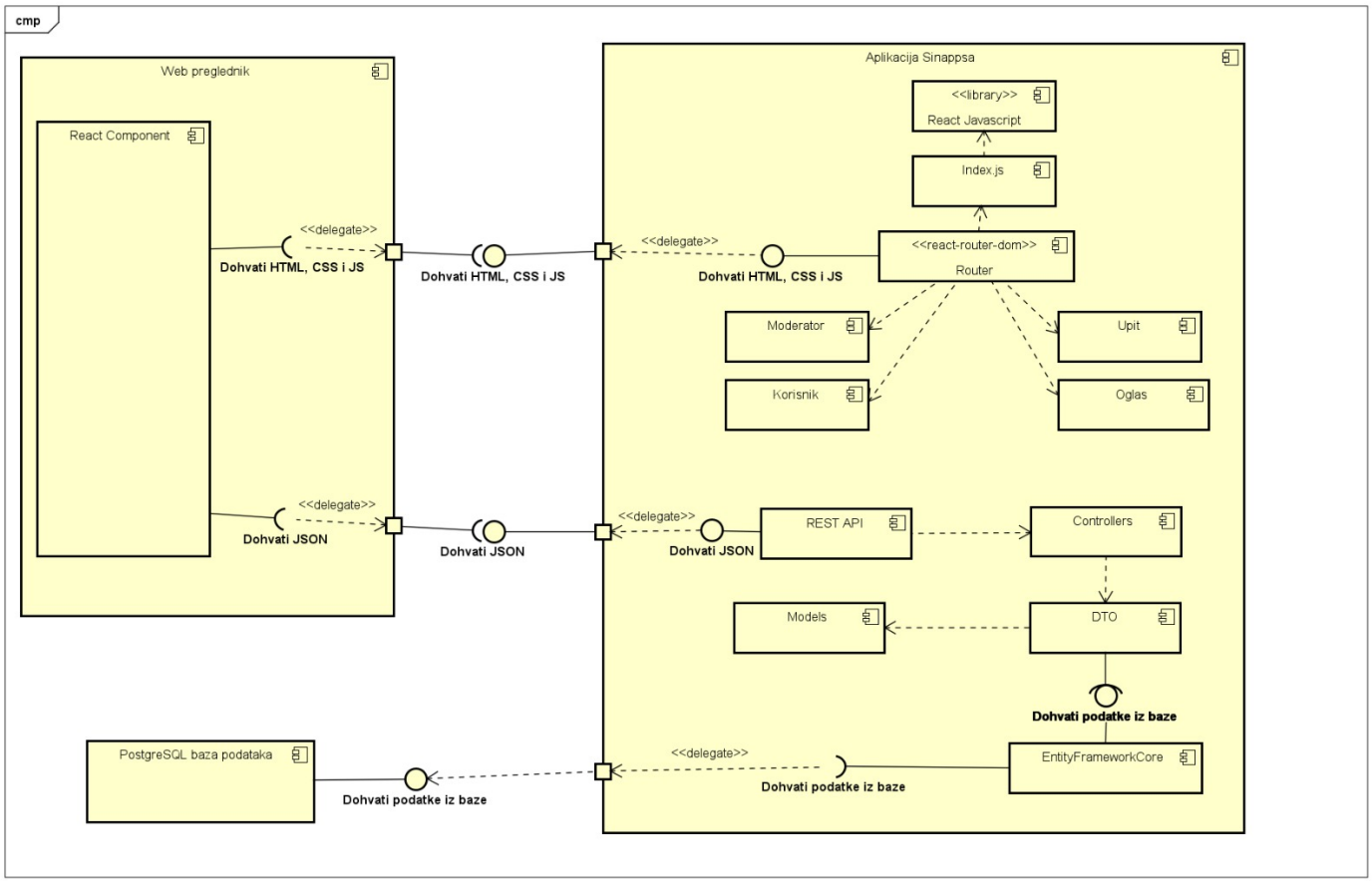
Dijagram aktivnosti primjenjuje se za opis modela toka upravljanja ili toka podataka. U modeliranju toka upravljanja svaki novi korak poduzima se nakon završenog prethodnog, a naglasak je na jednostavnosti. Na dijagramu aktivnosti 4.9 prikazan je proces objavljivanja oglasa. Registrirani korisnik se prije stvaranja oglasa prijavljuje u sustav svojim korisničkim imenom i lozinkom. Nakon uspješne prijave, kreira oglas: upisuje njegov naslov i opis, bira kolegij i kategoriju, te označava nudi li ili traži pomoć. Nakon klika na gumb za objavu, oglas se sprema u bazu podataka, a korisnik je obaviješten o objavi.



Slika 4.9: Dijagram aktivnosti - Prijava i objava oglasa

4.5 Dijagram komponenti

Dijagram komponenti prikazan na slici 4.10 opisuje organizaciju i meduovisnost komponenti, interne strukture i odnose prema okolini. Aplikaciji se pristupa pomoću dva sučelja. Frontend dijelu aplikacije pristupa se sučeljem za dohvat HTML, CSS i JS datoteka. Komponenta Router temeljem URL-a odlučuje koja će se datoteka poslužiti na to sučelje. Frontend se sastoji od niza datoteka podijeljenih u logičke cijeline nazvane po aktoru ili dijelu stranice koji ih koristi. Sve komponente frontenda ovise o React Javascript biblioteci. Preko sučelja za dohvat JSON podataka pristupa se REST API komponenti. REST API poslužuje podatke koji pripadaju backend dijelu aplikacije. EntityFrameworkCore dohvaća podatke iz baze podataka koristeći SQL upite i odgovarajuće sučelje. Prosljeđuje dobivene podatke ostalim komponentima kao DTO (data transfer object). Unutar web preglednika komponenta React-view pomoću odgovarajućih sučelja komunicira sa frontend i backend dijelom aplikacije i dohvaća datoteke temeljem korisnikovih akcija.



Slika 4.10: Dijagram komponenti

5. Implementacija i korisničko sučelje

5.1 Korištene tehnologije i alati

Tehnologija korištena u svrhu komunikacije unutar tima je aplikacija WhatsApp[1], dok je komunikacija s mentorom realizirana korištenjem aplikacije Microsoft Teams[2]. Repozitorij projekta se nalazi na GitLabu[3] koji je korišten i kao sustav za upravljanje kodom. Overleaf[4] je poslužio kao online LaTex uređivač prilikom pisanja potrebne dokumentacije.

Za izgradnju sustava korišten je Visual Studio Code[5]. VS Code uređivač je izvornog koda tvrtke Microsoft. Dolazi s ugrađenom podrškom za JavaScript, TypeScript i Node.js te ima bogat ekosustav proširenja za druge jezike. IntelliJ IDEA[6] je integrirano razvojno okruženje (IDE), korišteno u projektu za izgradnju komponenti, za razvoj računalnog softvera napisanog u Javi, Kotlinu, Groovyju i drugim jezicima koji se temelje na JVM-u. Razvila ga je tvrtka JetBrains.

Za izradu aplikacije backend je realiziran radnim okvirom Java Spring Boot[7] i jezikom Java[8] dok frontend koristi React[9] i jezik JavaScript[10]. Spring je projekt otvorenog koda koji pruža pojednostavljen, modularan pristup za izradu aplikacija s Javom. Samo ime, Spring, obično se odnosi na sam okvir aplikacije ili cijelu skupinu projekata, odnosno modula. Spring Boot je jedan specifičan modul koji je izgrađen kao proširenje Spring okvira. React je JavaScript biblioteka te se koristi za izradu web stranica ili mobilnih aplikacija. Uz React korištena je dodatna biblioteka React Bootstrap[11] za brz i pregledan UI.

Baza podataka je kreirana u PostgreSQL-u[12].

[1] https://www.whatsapp.com/
[2] https://www.microsoft.com/hr-hr/microsoft-teams/log-in
[3] https://about.gitlab.com/
[4] https://www.overleaf.com/
[5] https://www.overleaf.com/
[6] https://www.jetbrains.com/idea/
[7] https://spring.io/projects/spring-boot
[8] https://www.java.com/en/
[9] https://reactjs.org/
[10] https://www.javascript.com/
[11] https://react-bootstrap.github.io/
[12] https://www.postgresql.org/

5.2 Ispitivanje programskog rješenja

5.2.1 Ispitivanje komponenti

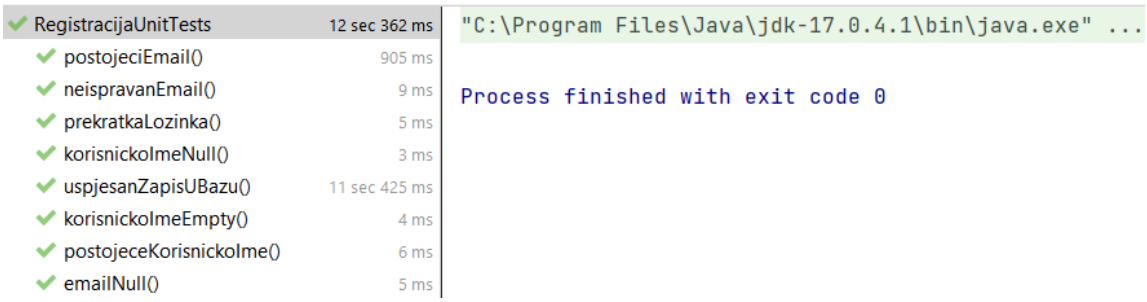
Napisali smo 8 JUnit testova koji pokrivaju funkcionalnost registracije. Testovi su izvršeni automatski. Test *korisnickoImeNull()* ispituje je li upisano korisničko ime jednako **null**. Također postoji sličan test koji provjerava je li e-mail jednak **null**. Takav test smo mogli provesti i na ostalim parametrima registracije, kao što su ime, prezime, lozinka i avatar. Rezultat bi bio isti kao i na ova 2 testa. Uz navedena 2 testa, sličan test njima jest test *korisnickoImeEmpty()* koji provjerava je li poslani podatak (u ovome slučaju korisničko ime) prazan. Također i ovaj test se može provesti na ostale već spomenute parametre, a rezultat bi opet bio isti. Tu je i test koji provjerava valjanost upisane lozinke (minimalni broj upisanih znakova). Imamo 2 testa koji provjeravaju nalaze li se uneseni podaci u bazi podataka. U našem slučaju to su podaci o korisničkom imenu i e-mailu. Uz test koji provjerava ispravnost unesenog e-maila (pravilo da je iz *fer.hr* domene), imamo test kojim uspješno stvaramo i zapisujemo u bazu novog korisnika. U nastavku vidimo izvorni kod svih testova i izlaz pri njihovom pokretanju.

```
import com.example.projekt.dao.RegKorisnikRepository;
import com.example.projekt.domain.RegistriraniKorisnik;
import com.example.projekt.service.RegKorisnikService;
import com.example.projekt.service.RequestDeniedException;
import com.example.projekt.service.impl.RegKorisnikServiceImpl;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import org.junit.jupiter.api.extension.ExtendWith;
import org.mockito.Mock;
import org.mockito.junit.jupiter.MockitoExtension;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

import javax.mail.MessagingException;
import java.io.UnsupportedEncodingException;
import java.util.Arrays;
import java.util.List;
import java.util.concurrent.ThreadLocalRandom;

import static org.mockito.ArgumentMatchers.any;
import static org.mockito.Mockito.when;
import static org.junit.jupiter.api.Assertions.*;

@ExtendWith(MockitoExtension.class)
public class RegistracijaUnitTests {

    @Mock
    private RegKorisnikRepository regKorisnikRepository;
    RegKorisnikService regKorisnikService;
    List<RegistriraniKorisnik> korisnikArrayList;

    @BeforeEach
    void initService() {
        regKorisnikService = new RegKorisnikServiceImpl(regKorisnikRepository, new BCryptPasswordEncoder());
        korisnikArrayList = Arrays.asList(new RegistriraniKorisnik("emailAna@fer.hr", "anchy123","Ana", "Anic", "
            tajnaLozinka", "1"),
              new RegistriraniKorisnik("emailIvo@fer.hr", "ivek44","Ivo", "Ivic", "tajnaLozinka", "2"),
```

```
            new RegistriraniKorisnik("emailBrane@fer.hr", "brat_99","Brane", "Branic", "tajnaLozinka", "2"));
    }

    @Test
    public void korisnickoImeNull() {
        Throwable exception = assertThrows(RequestDeniedException.class, () -> regKorisnikService.registriraj("
            emailLuka@fer.hr", null,"Luka", "Lukic", "tajnaLozinka", "4", "siteURL"));
        assertEquals("Sva polja moraju biti ispunjena", exception.getMessage());
    }

    @Test
    public void emailNull() {
        Throwable exception = assertThrows(RequestDeniedException.class, () -> regKorisnikService.registriraj(null, "lux"
            ,"Luka", "Lukic", "tajnaLozinka", "4", "siteURL"));
        assertEquals("Sva polja moraju biti ispunjena", exception.getMessage());
    }

    @Test
    public void korisnickoImeEmpty() {
        Throwable exception = assertThrows(RequestDeniedException.class, () -> regKorisnikService.registriraj("
            emailLuka@fer.hr", "","Luka", "Lukic", "tajnaLozinka", "4", "siteURL"));
        assertEquals("Polja ne smiju biti prazna", exception.getMessage());
    }

    @Test
    public void prekratkaLozinka() {
        when(regKorisnikRepository.existsByEmail("emailLuka@fer.hr")).thenReturn(false);
        when(regKorisnikRepository.existsByKorisnickoIme("lux")).thenReturn(false);

        Throwable exception = assertThrows(RequestDeniedException.class, () -> regKorisnikService.registriraj("
            emailLuka@fer.hr", "lux","Luka", "Lukic", "abcd", "4", "siteURL"));
        assertEquals("Lozinka mora imati barem 5 znakova.", exception.getMessage());
    }

    @Test
    public void postojeciEmail() {
        when(regKorisnikRepository.existsByEmail("emailBrane@fer.hr")).thenReturn(true);

        Throwable exception = assertThrows(RequestDeniedException.class, () -> regKorisnikService.registriraj("
            emailBrane@fer.hr", "lux","Luka", "Lukic", "abcde", "4", "siteURL"));
        assertEquals("Ova e-mail adresa je vec iskoristena.", exception.getMessage());
    }

    @Test
    public void postojeceKorisnickoIme() {
        when(regKorisnikRepository.existsByKorisnickoIme("ivek44")).thenReturn(true);

        Throwable exception = assertThrows(RequestDeniedException.class, () -> regKorisnikService.registriraj("
            emailLuka@fer.hr", "ivek44","Luka", "Lukic", "abcde", "4", "siteURL"));
        assertEquals("Ovo korisnicko ime je vec zauzeto.", exception.getMessage());
    }

    @Test
    public void neispravanEmail() {
        Throwable exception = assertThrows(RequestDeniedException.class, () -> regKorisnikService.registriraj("
            emailLuka@gmail.com", "lux","Luka", "Lukic", "abcde", "4", "siteURL"));
        assertEquals("E-mail adresa " + "emailLuka@gmail.com" + " nije iz FER-ove domene.", exception.getMessage());
    }

    @Test
    public void uspjesanZapisUBazu() throws MessagingException, UnsupportedEncodingException {
        Long randomLong = ThreadLocalRandom.current().nextLong(10, 21);
        RegistriraniKorisnik korisnikUBazi = new RegistriraniKorisnik("emailLuka@fer.hr", "lux","Luka", "Lukic", "abcde",
            "4");
        korisnikUBazi.setId(randomLong);

        when(regKorisnikRepository.save(any(RegistriraniKorisnik.class))).thenReturn(korisnikUBazi);

        assertEquals(randomLong, regKorisnikService.registriraj("emailLuka@fer.hr", "lux","Luka", "Lukic", "abcde", "4",
            "siteURL").getId());
    }
```

```
}
```

Listing 5.1: JUnit test za registraciju



Slika 5.1: Izlaz pokretanja JUnit testa za registraciju

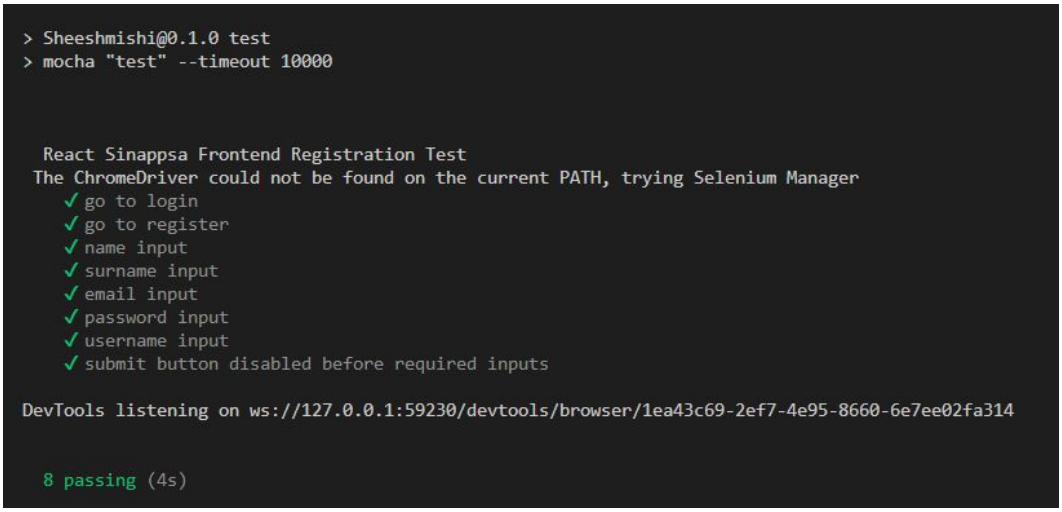
5.2.2 Ispitivanje sustava

Koristeći radni okvir **Selenium** smo ispitali rad dijela sustava. Napisali smo osam testova koji pokrivaju funkcionalnost registracije korisnika. Testovi su izvršeni automatski. Koristili smo **Selenium WebDriver** - podršku za pisanje testova, a njih smo napisali koristeći jezik JavaScript. Cilj nam je bio provjeriti podudarnosti sustava s funkcijskim zahtjevom. Testovima smo provjerili postoji li gumb "Prijava" pri učitavanju stranice - "`go_to_login`" test, postoji li gumb "Registriraj se ovdje" nakon klika na gumb "Prijava" - "`go_to_register`" test, onemogućenost odabira gumba "Registriraj se" prije nego su svi potrebni podaci uneseni - "`submit_button_disabled_before_required_inputs`" test te mogu li se, odlaskom na registraciju, upisati sljedeći podaci: ime - "`name_input` " test, prezime - "`surname_input`" test, e-mail - "`email_input`" test, lozinka - "`password_input`" test, korisničko ime - "`username_input`" test. U nastavku vidimo izvorni kod svih testova te izlaz pri njihovu pokretanju.

```
var assert = require("assert");
const webdriver = require("selenium-webdriver");

const rootUrl = "http://localhost:3000";
const loginUrl = rootUrl + "/login";
const registerUrl = rootUrl + "/register";

describe("React Sinappsa Frontend Registration Test", () => {
  var driver;

  before(() => {
    driver = new webdriver.Builder().forBrowser("chrome").build();
  });

  it("go_to_login", () => {
    driver.get(rootUrl).then(() => {
      driver
        .findElement(webdriver.By.linkText("Prijava"))
        .then((element) => element.click())
        .then(() => driver.getCurrentUrl())
        .then((url) => assert.equal(url, loginUrl))
        .catch(() => assert.fail("Test failed"));
    });
  });

  it("go_to_register", () => {
    driver.get(loginUrl).then(() => {
      driver
        .findElement(webdriver.By.linkText("Registriraj se ovdje"))
        .then((element) => element.click())
        .then(() => driver.getCurrentUrl())
        .then((url) => assert.equal(url, registerUrl))
        .catch(() => assert.fail("Test failed"));
    });
  });

  it("name_input", () => {
    driver.get(registerUrl).then(() => {
      driver
        .findElement(webdriver.By.id("ime"))
        .sendKeys("Testno Ime\n")
        .then(() => {
          driver.getPageSource().then((source) => {
```

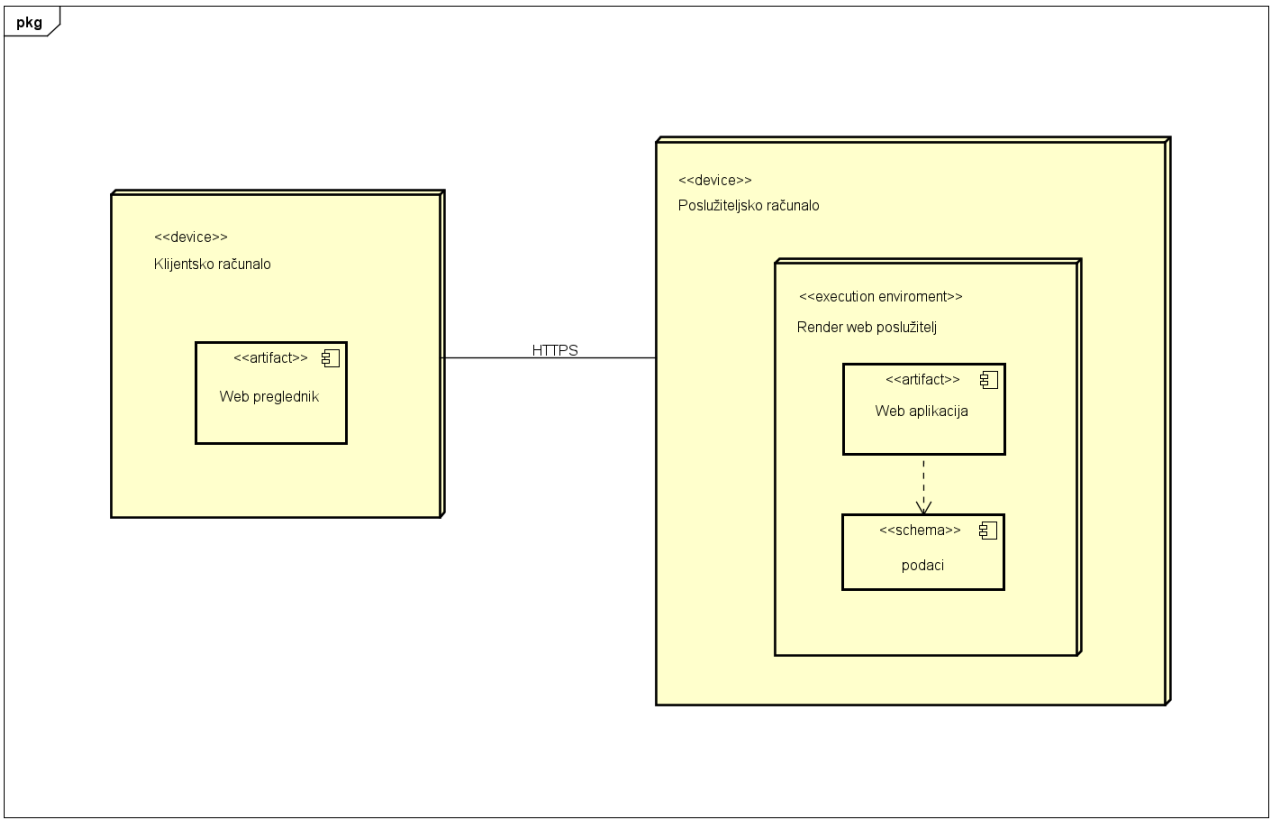
```
            assert.equal(source.includes("Testno Ime"), true);
          });
        });
    });
  });
  it("surname input", () => {
    driver.get(registerUrl).then(() => {
      driver
        .findElement(webdriver.By.id("prezime"))
        .sendKeys("Testno Prezime\n")
        .then(() => {
          driver.getPageSource().then((source) => {
            assert.equal(source.includes("Testno Prezime"), true);
          });
        });
    });
  });

  it("email input", () => {
    driver.get(registerUrl).then(() => {
      var email = driver.findElement(webdriver.By.id("email"));
      email.sendKeys("luka.lukic@fer.hr")
        .then(() => {
          var value = email.getAttribute("value");
          assert.equal(value.contains("@fer"), true);
        });
    });
  });

  it("password input", () => {
    driver.get(registerUrl).then(() => {
      var password = driver.findElement(webdriver.By.id("lozinka"));
      password.sendKeys("12345")
        .then(() => {
          var value = password.getAttribute("value");
          assert.equal(value.length >= 5 ? true : false, true);
        });
    });
  });

  it("username input", () => {
    driver.get(registerUrl).then(() => {
      var username = driver.findElement(webdriver.By.id("username"));
      username.sendKeys("testni username")
        .then(() => {
          var value = username.getAttribute("value");
          assert.equal(value === "testni username" ? true : false, true);
        });
    });
  });

  it("submit button disabled before required inputs", () => {
    driver.get(registerUrl).then(() => {
      var submit = driver.findElement(webdriver.By.id("submit"))
      .then(() => {
        var submitState = submit.getAttribute("disabled");
        assert.equal(submitState === true ? true : false, true)
      })
    })
  })

  after(() => driver.quit());
});
```

Listing 5.2: Selenium test za registraciju



Slika 5.2: Izlaz pokretanja Selenium testa za registraciju

5.3 Dijagram razmještaja



Slika 5.3: Dijagram razmještaja

5.4 Upute za puštanje u pogon

5.4.1 Konfiguracija poslužitelja baze podataka

Bazu podataka smo podigli na servisu Render. Potrebno je izraditi korisnički račun kako bi se dobila mogućnost puštanja aplikacije u pogon. Kako bismo dodali novu bazu podataka na Render servis, odabiremo karticu *Dashboard* te potom *New*. Pojavit će se padajuća lista sa servisima koje je moguće pustiti u pogon. Odabiremo *PostgreSQL* kao bazu podataka. Ispunimo redom podatke za ime našeg servisa kojeg puštamo u pogon, ime same baze podataka te korisničko ime. Regiju ostaviti na *Frankfurt (EU Central)* ili namjestiti na istu ako nije tako postavljeno kao inicijalna vrijednost. Ostala polja možemo zanemariti. Potom odaberemo besplatnu inačicu servisa koja će za naše potrebe biti i više nego dovoljna. Na dnu stranice nalazi se gumb *Create Database* čijim se pritiskom stvara naša baza podataka na Render servisu. Ako se vratimo na stranicu *Dashboard*, vidjet ćemo naš novostvoreni servis, njegov status, tip, itd. U nastavku ćemo objasniti konfiguraciju baze podataka.

5.4.2 Konfiguracija baze podataka

Sada je potrebno povući konfiguracijske parametre novostvorenog servisa za bazu podataka. To napravimo klikom na naš servis. Otvara se stranica s raznim informacijama o našem servisu. Ono što će nama biti potrebno za uspješno povezivanje baze podataka i poslužitelja naše aplikacije nalazi se u odjeljku *Connections*.

Otvorimo sada datoteku *application.properties* koja se nalazi u *./IzvorniKod/backend/src/main/resources* mapi. Pogledajmo sljedeći dio datoteke:

```
spring.datasource.url=...
spring.datasource.password=...
spring.datasource.username=...
```

Prva linija definira *url* naše baze podataka na Render servisu. Url je oblika:

```
jdbc:postgresql://
<Hostname>.frankfurt-postgres.render.com<Port>/
<Database>
```

Vrijednosti unutar <> je ime podatka sa Render stranice. Sljedeće dvije linije definiraju zaporku i korisničko ime kojim ćemo pri pokretanju poslužitelja pristupati našoj bazi podataka s Render servisa. *Username* i *Password* vidljivi su na Render stranici.

Također, osim navedenih, potrebno je još definirati i sljedeće konfiguracijske parametre:

```
spring.datasource.driverClassName=org.postgresql.Driver
spring.jpa.database-platform=org.hibernate.dialect.PostgreSQLDialect
spring.jpa.show-sql=false
spring.jpa.hibernate.ddl-auto=create
spring.jpa.defer-datasource-initialization=true
```

Ukratko, ovdje konfiguriramo koji upravljač za bazu podataka koristimo, želimo li ispisivati sql-naredbe, način rada baze podataka, itd. Važnije svojstvo je *spring.jpa.hibernate.ddl-auto*. Ako je postavljeno na *create*, baza podataka će se nanovo puniti podacima. Druga opcija je *update*; ona definira čuvanje starih podataka prilikom novog pokretanja poslužitelja. Postoje i druge opcije, no o njima ovdje neće biti riječi.

5.4.3 Punjenje baze podataka

Bazu podataka punimo koristeći datoteku *data.sql* (lokacija: *./IzvorniKod/backend /src/main/resources*). Prilikom pokretanja poslužitelja, Spring Boot će automatski provjeriti postoji li datoteka *data.sql* te ako postoji, slijedno će izvršavati sql-naredbe navedene u istoj datoteci. Pritom će Spring Boot koristiti konfiguracijske parametre objašnjene u prethodnom odlomku. Važno je napomenuti kako svojstvo *spring.jpa.hibernate.ddl-auto* treba biti postavljeno na *create* kako bi se punjenje baze podataka pokrenulo.

5.4.4 Pokretanje poslužitelja baze podataka

Poslužitelj baze podataka (Render servis) automatski se pokreće nakon stvaranja servisa za bazu podataka. Da je to doista tako, možemo provjeriti log našeg servisa za bazu podataka odlaskom na *Dashboard - <ImeServisaZaBazuPodataka> - Logs*.

5.4.5 Konfiguracija poslužitelja serverske strane

Poslužiteljsku stranu naše web aplikacije (engl. *backend*) također puštamo u pogon preko servisa Render. Proces je nešto složeniji nego što je to bio slučaj s bazom podataka. Također odlazimo na *Dashboard*, potom *New*, te odabiremo *Web Service*. Prvi korak je spojiti se na jedan od nama dostupnih repozitorija na *GitLab-u*. Podsjetimo, uvjet za korištenje Render servisa je imati korisnički račun, a da bi isti napravili, potrebno je povezati *GitLab* račun. Nakon što smo odabrali jedan od repozitorija, pojavit će nam se obrazac sličan onome kao kod baze podataka. Unesemo ime koje želimo za naš servis, ostavimo regiju kako je predloženo (*Frankfurt*) te odaberemo granu (engl. *branch*) *GitLab* repozitorija iz koje ćemo htjeti povući razvojni kod. Opcionalno, ako se korijenska mapa naše poslužiteljske aplikacije razlikuje od korijenske mape repozitorija, možemo ju navesti pod *Root Directory*. Za okruženje (*Environment*) odabiremo *Docker*. Preduvjet je preuzeti docker skriptu koju su pripremili kolege iz CROZ firme te ju postaviti u vlastiti projekt. Ponovno odabiremo besplatnu vrstu servisa te odabiremo opciju *Advanced*. Otvorit će se obrazac s dodatnim informacijama, poput varijabli okruženja. Dodat ćemo jednu varijablu okruženja, imena *PORT*, koja će specificirati na kojem se portu vrti naš servis. Pogledamo li ponovno *application.properties* datoteku, vidjet ćemo sljedeći redak:

```
server.port=${PORT:8080}
```

Render koristi ovu informaciju kako bi na ispravan način postavio port našeg web poslužitelja. Također, potrebno je definirati *Dockerfile Path* unutar našeg repozitorija. U njemu se nalazi već spomenuta *Docker* skripta. Ostala polja možemo zanemariti ili ostaviti inicijalne vrijednosti. Pritiskom na *Create Web Service* započinjemo izgradnju našeg web poslužitelja. Postupak će trajati nekoliko minuta, nakon čega će se poslužiteljska aplikacija automatski pokrenuti i biti aktivna neko vrijeme.

5.4.6 Konfiguracija veze s bazom podataka

Spring Boot, framework koji koristimo za izgradnju poslužiteljske strane naše web aplikacije, koristi datoteku *application.properties* unutar koje se mogu definirati razne konfiguracijske postavke. Prisjetimo se, u jednom od prethodnih poglavlja definirali smo upravo jedne takve konfiguracijske postavke za bazu podataka. Spring Boot će iste koristiti za uspostavljanje veze s bazom podataka.

5.4.7 Konfiguracija poslužitelja klijentske strane

Postupak konfiguracije i izgradnje klijentske strane web aplikacije gotovo je isti kao i izradnja poslužiteljske strane web aplikacije. Glavna razlika su *Build Command* te *Start Command*. Koristimo *yarn* kao upravitelja projektom (engl. *package manager*). Stoga kao naredbu izgranje koristimo *yarn build*, a kao naredbu pokretanja *yarn start-prod*. Što one zapravo pokreću u pozadini definirano je u datoteci *package.json*:

```
...
scripts: {
    ...
    "build": "yarn install && react-scripts build",
    "start-prod": "node app.js",
    ...
}
...
```

5.4.8 Definiranje adrese spajanja

Preostaje objasniti kako spojiti klijentsku stranu web aplikacije s poslužiteljskom. Pogledajmo prvo kako je to učinjeno na poslužiteljskoj strani. Znamo kako je za prihvaćanje http-zahtjeva u Spring Boot modelu zadužen *Controller* sloj. On je ulazna i izlazna točka svakog zahtjeva (engl. *http-request*) i odgovora (engl. *http-response*). S obzirom da lokacije naše klijentske i poslužiteljske strane nisu iste (engl. *url*), potrebno je eksplicitno definirati s koje adrese se dopušta prolaz zahtjeva. Drugim rječima, definiramo s kime poslužitelj smije komunicirati. Ova funkcionalnost implementirana je korištenjem anotacija nad svim klasama koje predstavljaju upravljač (engl. *controller*). Evo primjera:

```
@CrossOrigin(origins = "https://sheeshmishi-fe.onrender.com")
@RestController
@RequestMapping("/user")
public class UserController { ... }
```

Prvi redak (prva anotacija) definira upravo to: dopuštena je komunikacija između našeg poslužitelja te klijenta koji je pokrenut na navedenoj lokaciji.
Obrnuta priča nešto je jednostavnija. Potrebno je samo koristiti lokaciju poslužitelja prilikom korištenja ugrađene funkcije *fetch* koju podržava *javascript* i *nodejs*. Kako ne bismo *„tvrdokodirali“* (engl. *hard-code*) lokaciju našeg poslužitelja, istu smo izdvojili u posebnu datoteku, *config.json*:

```
{
    "hostname": {
        "dev": "http://localhost:3000/api",
        "prod": "https://sheeshmishi.onrender.com/api"
    },
    "profile": "prod"
}
```

Postoje dvije inačice imena poslužitelja (engl. *hostname*): *dev* (skraćeno od *development*) i *prod* (skraćeno od *production*). U slučaju puštanja u pogon, želimo koristiti drugu inačicu, a to konfiguriramo pod svojstvom *profile : prod*. Također, važno je spomenuti *proxy* koristimo u našoj web aplikaciji. Pogledamo li ponovno datoteku *application.properties*, možemo uočiti redak:

```
server.servlet.context-path=/api
```

On definira sljedeće ponašanje: svaki zahtjev (engl. *http-request*) koji poželimo poslati našem poslužitelju, zapravo šaljemo na lokaciju poslužitelja te dodatno još */api*. To je ukratko moderan i učinkovit pristup povezivanja klijentske i poslužiteljske strane.

5.4.9 Pokretanje aplikacije

Aplikaciju pokrećemo odlaskom na poveznicu:

`https://sheeshmishi-fe.onrender.com`

Moguće je da će zbog načina na koji Render servis radi (i besplatne inačice koju smo uzeli prilikom izgradnje servisa) aplikaciji trebati nekoliko minuta da se pokrene, s obzirom da se i klijentski i poslužiteljski servisi nakon otprilike 20 minuta neaktivnosti ugase.

6. Zaključak i budući rad

Naš zadatak je bio napraviti web aplikaciju koju bi koristili studenti FER-a da bi jedni drugima pomagali oko labosa, ispita i učenja gradiva. Ideja je bila da studenti mogu objaviti oglase i da im drugi studenti onda mogu odgovoriti na te oglase, tj. postaviti upite. Da bi bilo zanimljivije postoji i ocjenjivanje te rang lista 10 najbolje ocjenjenih studenata-pomagača. Nakon više mjeseci rada smo zadovoljni završnom verzijom naše aplikacije, iako smo svjesni da bi neke funkcionalnosti mogli poboljšati.

Izradu aplikacije bi mogli podijelit u dvije faze, između kojih je bilo 1. kolokviranje projekta. U prvoj fazi smo se više bavili dokumentacijom, tada smo detaljno raspisali obrasce uporabe te zapisali sve funkcionalne i nefunkcionalne zahtjeve. Izradili smo i dijagram obrazaca uporabe, sekvencijski dijagram te dijagram razreda. Također smo na *backendu* napravili kostur arhitekture te smo implementirali osnovne funkcionalnosti: registraciju i prijavu korisnika. Da bi to uspjeli smo morali napravit relacijski dijagram baze podataka. Smatramo da smo jako dobro napravili kostur arhitekture i entitete u bazi podataka pa u drugoj fazi nismo imali problema s programiranjem na *backendu*. Na *frontendu* smo također implementirali registraciju i prijavu da bi sve to zajedno funkcioniralo.

U drugoj fazi smo se puno više orijentirali na samo programiranje, podijelili smo poslove i svaki član grupe je doprinio konačnom proizvodu. Uložili smo puno vremena i truda u projekt, ponajviše zbog toga što nismo otprije upoznati sa alatima koje smo koristili niti smo ikad samostalno izrađivali web aplikaciju. Zadnji dio rada na aplikaciji prije pisanja dokumentacije je bilo testiranje, što je također bilo novo i zanimljivo svima nama. Zatim smo i dovršili dokumentaciju projekta, napravili smo preostale dijagrame te napisali upute za *deploy* te ostale tekstualne dijelove dokumentacije, također smo doradili neke dijelove prve verzije dokumentacije.

Komunikacija između članova grupe se odvijala preko Whatsappa, a ponekad smo se nalazili i na video pozivima ili uživo da bi neke stvari razjasnili. Imali smo i jedan Word dokument koji smo svi mogli čitati i uređivati u kojem smo za svaki HTTP zahtjev napisali putanju, tip zahtjeva, očekivani ulaz (body u JSON

obliku) te izlaz (također u JSON obliku), na taj način smo osigurali da komunikacija između *backenda* i *frontenda* bude ispravna i jednoznačna.

Budući rad na projektu bi se sastojao od toga da popravimo neke elemente završne web aplikacije. Što se tiče samih funkcionalnih zahtjeva, oni su svi ispunjeni, no što se tiče nefunkcionalnih zahtjeva, neke stvari se mogu popraviti. To su npr. responzivnost aplikacije, te izgled stranice 'Moj profil'. Također u *deploy* verziji aplikacije treba nekoliko sekundi da bi se izvršile neke akcije, npr. registracija ili otvaranje 'Moj profil' stranice, iako to je možda manje naša greška, a više do toga da smo koristili besplatne (zato i slabije verzije) alate za *deploy*. Mogla bi se napraviti i mobilna aplikacija jer bi to pojednostavilo korištenje aplikacije mnogim studentima. Još jedna ideja je da se aplikacija može poopćiti i da ne bude samo za FER, nego da i ostali faksevi imaju svoje verzije.

Smatramo da je svima nama sudjelovanje u ovom projektu bilo jako korisno iskustvo za naše buduće studiranje i karijeru. Izvježbali smo se u korištenju GIT-a, upoznali smo se s izazovima rada u grupi, te mislim da je najvažnija činjenica da smo se upoznali s nekim alatima i radnim okvirima. Na backendu smo koristili *Spring*, te možemo reći da se nama na *backendu* svidjelo koristiti taj radni okvir, i da želimo u budućnosti još više stvari raditi i naučiti puno više o *Springu*. Također su kolege s *frontenda* sada upoznali *React* koji je jedan od najpopularnijih radnih okvira za *frontend*. Definitivno smo zadovoljni našim radom, naučenim stvarima, iskustvom koje smo stekli te konačnom verzijom aplikacije.

Popis literature

1. Programsko inženjerstvo, FER ZEMRIS, http://www.fer.hr/predmet/proinz
2. Spring Framework, https://spring.io/
3. React JavaScript library, https://reactjs.org/
4. React Framework, https://react-bootstrap.github.io/
5. Java tutorials, code examples and sample projects, https://www.codejava.net/
6. Baeldung, https://www.baeldung.com/
7. Amigoscode, https://amigoscode.com/
8. GeeksForGeeks portal, https://www.geeksforgeeks.org/
9. Stack overflow portal, https://stackoverflow.com/
10. Hrvoje Šimić, Progi project teams - backend, https://gitlab.com/hrvojesimic/progi-project-teams-backend
11. Josip Tomić, Progi project teams - frontend, https://gitlab.com/jtomic/opp-project-teams-frontend
12. Progi deploy demo, https://gitlab.com/progi-deploy-demo
13. Selenium dokumentacija, https://www.selenium.dev/selenium/docs/api/py/api.html
14. Selenium automatizacija testova, https://medium.com/@oyetoketoby80/automating-your-
15. JUnit 5 tutorial, https://www.vogella.com/tutorials/JUnit/article.html

Indeks slika i dijagrama

2.1 Skica dizajna homepagea 6

3.1 Dijagram obrasca uporabe, funkcionalnosti neregistriranog korisnika i moderatora 18
3.2 Dijagram obrasca uporabe, funkcionalnosti registriranog korisnika vezane uz oglase 19
3.3 Dijagram obrasca uporabe, funkcionalnosti registriranog korisnika vezane uz upite 20
3.4 Sekvencijski dijagram za UC5 21
3.5 Sekvencijski dijagram za UC7 22
3.6 Sekvencijski dijagram za UC17 23

4.1 Spring Boot Arhitektura 26
4.2 Relacijski dijagram baze podataka 30
4.3 Dijagram razreda - dio Models 31
4.4 Dijagram razreda - dio Data Transfer Objects 32
4.5 Dijagram razreda - dio Controllers 33
4.6 Dijagram razreda - dio Repositories 33
4.7 Dijagram razreda - dio Services 34
4.8 Dijagram stanja - stvaranje oglasa i prijava korisnika 35
4.9 Dijagram aktivnosti - Prijava i objava oglasa 36
4.10 Dijagram komponenti 37

5.1 Izlaz pokretanja JUnit testa za registraciju 41
5.2 Izlaz pokretanja Selenium testa za registraciju 44
5.3 Dijagram razmještaja 45

6.1 prikaz aktivnosti na repozitoriju - grana develop 60
6.2 prikaz aktivnosti na repozitoriju - grana devdoc 61

Dodatak: Prikaz aktivnosti grupe

Dnevnik sastajanja

1. sastanak
    - 20.listopada 2022.
    - Prisustvovali: Ivan Krešo, Karlo Dimjašević, Matea Kranjčić, Marko Prosenjak, Barbara Kralj, Lovro Gaćina, Marin Čičak
    - Teme sastanka:
        - Pročitali zadatak i diskutirali nejasnoće
        - Zapsali pitanja za asistenticu
2. sastanak
    - 27.listopada 2022.
    - Prisustvovali: Ivan Krešo, Karlo Dimjašević, Matea Kranjčić, Marko Prosenjak, Barbara Kralj, Lovro Gaćina, Marin Čičak
    - Teme sastanka:
        - Popisali funkcionalnosti
        - Rasporedili funkcije po korisnicima (neregistrirani, registrirani i moderator
3. sastanak
    - 30.listopada 2022.
    - Prisustvovali: Ivan Krešo, Karlo Dimjašević, Matea Kranjčić, Marko Prosenjak, Barbara Kralj, Lovro Gaćina, Marin Čičak
    - Teme sastanka:
        - Dogovorili i rasporedili zadatke po članovima vezane za dijagrame obrazaca uporabe
4. sastanak
    - 30.listopada 2022.
    - Prisustvovali: Ivan Krešo, Karlo Dimjašević, Matea Kranjčić, Marko Prosenjak, Barbara Kralj, Lovro Gaćina, Marin Čičak
    - Teme sastanka:

  - Dogovorili i rasporedili članove na *frontend* i *backend* dijelove programiranja

5. sastanak
   - 3.prosinca 2022.
   - Prisustvovali: Ivan Krešo, Karlo Dimjašević, Matea Kranjčić, Marko Prosenjak, Barbara Kralj, Lovro Gaćina, Marin Čičak
   - Teme sastanka:
     - Diskusija projekta prije prvog kolokviranja
6. sastanak
   - 21.prosinca 2022.
   - Prisustvovali: Ivan Krešo, Karlo Dimjašević, Matea Kranjčić, Marko Prosenjak, Barbara Kralj, Lovro Gaćina, Marin Čičak, Laura Majer
   - Teme sastanka:
     - Demonstracija alfa verzije asistentu

Tablica aktivnosti

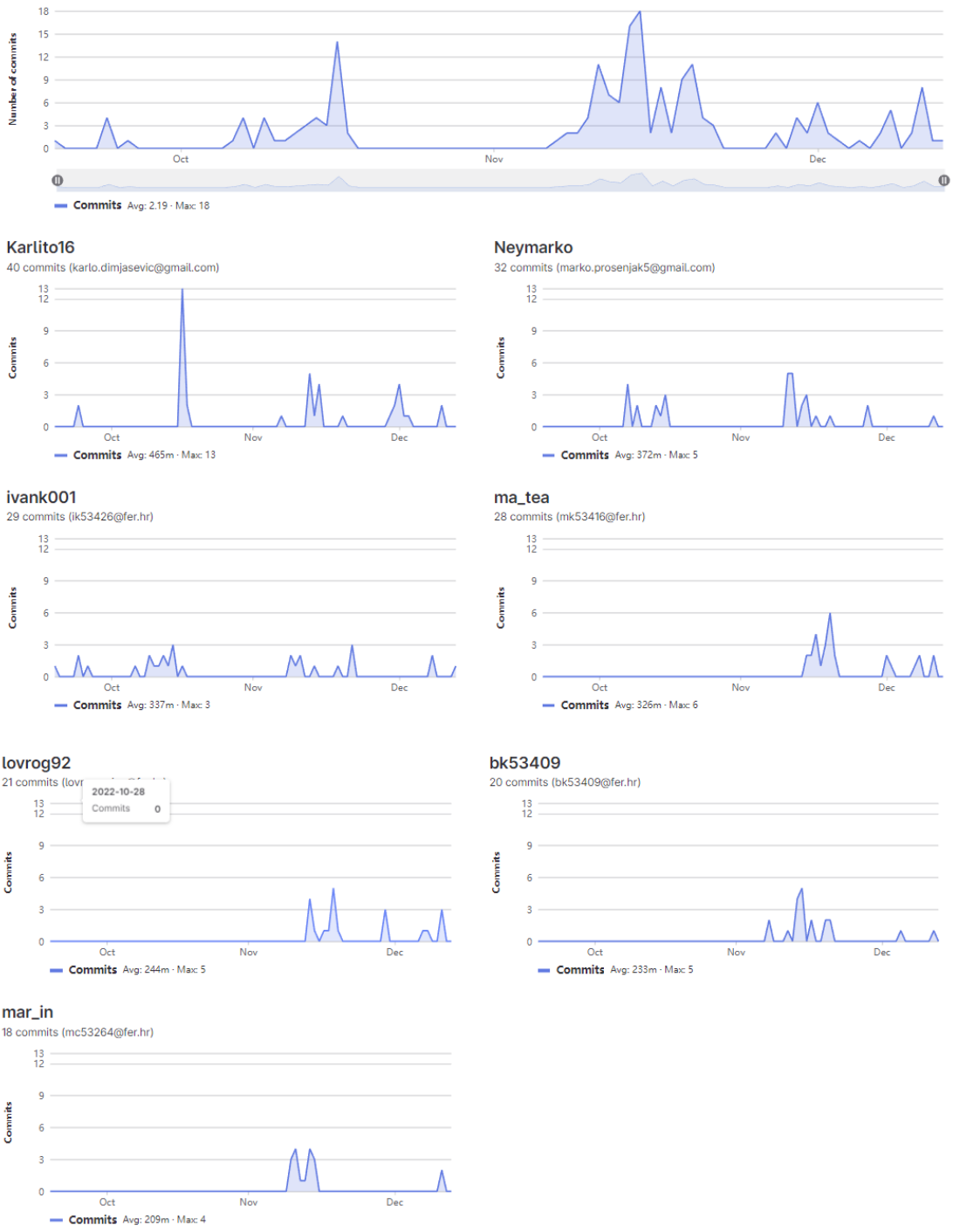
	Voditelj: Ivan Krešo	Lovro Gaćina	Karlo Dimjaševi	Matea Kranjčić	Barbara Kralj	Marko Prosenjak	Marin Čičak
Upravljanje projektom	2.5						
Opis projektnog zadatka			1		1		
Funkcionalni zahtjevi	1					1	
Opis pojedinih obrazaca		2			2	2	
Dijagram obrazaca				1.5			1.5
Sekvencijski dijagrami		1.5		1.5			2.5
Opis ostalih zahtjeva	1						
Arhitektura i dizajn sustava	0.5				1.5		
Baza podataka	2						
Dijagram razreda					1		
Dijagram stanja				1.5			
Dijagram aktivnosti		2					
Dijagram komponenti		2					
Korištene tehnologije i alati					2		
Ispitivanje programskog rješenja	3.5	3.5	3.5	3.5	3.5	3.5	3
Dijagram razmještaja						1.5	
Upute za puštanje u pogon			4				
Dnevnik sastajanja							1
Zaključak i budući rad	2						
Popis literature							1

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

	Voditelj: Ivan Krešo	**Lovro Gaćina**	**Karlo Dimjašević**	**Matea Kranjčić**	**Barbara Kralj**	**Marko Prosenjak**	**Marin Čičak**
back end	18				16		15
front end		13	26	18		18.5	6

Dijagrami pregleda promjena



Slika 6.1: prikaz aktivnosti na repozitoriju - grana develop

Slika 6.2: prikaz aktivnosti na repozitoriju - grana devdoc