

Programsko inženjerstvo

Ak. god. 2022./2023.

Sinappsa

Dokumentacija, Rev. 2.

Grupa: *BijeliCasio*

Voditelj: *Borna Nikolić*

Datum predaje: *18.11.2022.*

Nastavnik: *Laura Majer*

Sadržaj

1 Dnevnik promjena dokumentacije	3
2 Opis projektnog zadatka	6
3 Specifikacija programske potpore	12
3.1 Funkcionalni zahtjevi	12
3.1.1 Obrasci uporabe	14
3.1.2 Sekvencijski dijagrami	24
3.2 Ostali zahtjevi	28
4 Arhitektura i dizajn sustava	29
4.1 Baza podataka	31
4.1.1 Opis tablica	31
4.1.2 Dijagram baze podataka	35
4.2 Dijagram razreda	36
4.3 Dijagram stanja	39
4.4 Dijagram aktivnosti	40
4.5 Dijagram komponenti	42
5 Implementacija i korisničko sučelje	44
5.1 Korištene tehnologije i alati	44
5.2 Ispitivanje programskog rješenja	46
5.2.1 Ispitivanje komponenti	46
5.2.2 Ispitivanje sustava	52
5.3 Dijagram razmještaja	56
5.4 Upute za puštanje u pogon	57
6 Zaključak i budući rad	62
Popis literature	63
Indeks slika i dijagrama	65

Dodatak: Prikaz aktivnosti grupe

66

1. Dnevnik promjena dokumentacije

Rev.	Opis promjene/dodatka	Autori	Datum
0.1	Napravljen predložak	Ana Grčević	28.10.2022.
0.2	Dodan opis projektnog zadatka	Ana Grčević	01.11.2022.
0.3	Dodani obrasci uporabe	Ana Grčević, Martina Majdiš, Tin Margetić, Borna Nikolić, Matija Pavičić, Mihael Petričević, Milica Vuković	02.11.2022.
0.4	Ispravak obrazaca uporabe	Ana Grčević	10.11.2022.

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

Rev.	Opis promjene/dodatka	Autori	Datum
0.5	Dodani sekvencijski dijagrami	Ana Grčević, Martina Majdiš, Tin Margetić, Borna Nikolić, Matija Pavičić, Mihael Petričević, Milica Vuković	10.11.2022.
0.6	Dodani dionici, aktori i funkcionalni zahtjevi	Ana Grčević	13.11.2022.
0.7	Dodani ostali zahtjevi	Ana Grčević, Milica Vuković	13.11.2022.
0.8	Dodani dijagram i opis baze podataka	Borna Nikolić	13.11.2022.
0.9	Dodan opis arhitekture sustava	Ana Grčević	15.11.2022.
0.10	Dodan dijagram razreda	Ana Grčević, Borna Nikolić	16.11.2022.
1.0	Verzija samo s bitnim dijelovima za 1. ciklus		

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

Rev.	Opis promjene/dodatka	Autori	Datum
1.1	Dodane korištene tehnologije i alati	Ana Grčević	6.1.2023.
1.2	Dodan dijagram stanja	Ana Grčević	7.1.2023.
1.3	Dodane upute za puštanje u pogon	Ana Grčević, Borna Nikolić	7.1.2023.
1.4	Dodan dijagram razmještaja	Ana Grčević	7.1.2023.
1.5	Dodan dijagram aktivnosti	Ana Grčević	7.1.2023.
1.6	Dodan dijagram komponenti	Borna Nikolić	7.1.2023.
1.7	Updejtan dijagram razreda	Borna Nikolić	7.1.2023.
1.8	Dodano ispitivanje programskog rješenja	Milica Vuković	9.1.2023.
1.9	Dodan zaključak	Ana Grčević	9.1.2023.
1.10	Korigiranje teksta i provjera dokumentacije	Ana Grčević	11.1.2023.
2.0	Konačni tekst predložka dokumentacije		

2. Opis projektnog zadatka

Sinappsa je web aplikacija koja omogućuje studentima FER-a da traže ili pružaju pomoć oko specifičnog gradiva, laboratorijskih vježbi ili kolegija općenito. Student-pomagač će moći staviti oglas na koji će se drugi studenti moći javljati. Nakon obavljenih instrukcija student daje recenziju studentu-pomagaču. Ta recenzija ulazi u cjelokupni rejting studenta koji se zatim prikazuje na rejting listi na web stranici. Prikaz rejting liste sadrži 10 registriranih članova s najvećim rejtingom i ažurira se redovito.

Aplikacija se sastoji od tri vrste korisnika, a to su:

1. neregistrirani korisnici
2. registrirani korisnici
3. moderatori

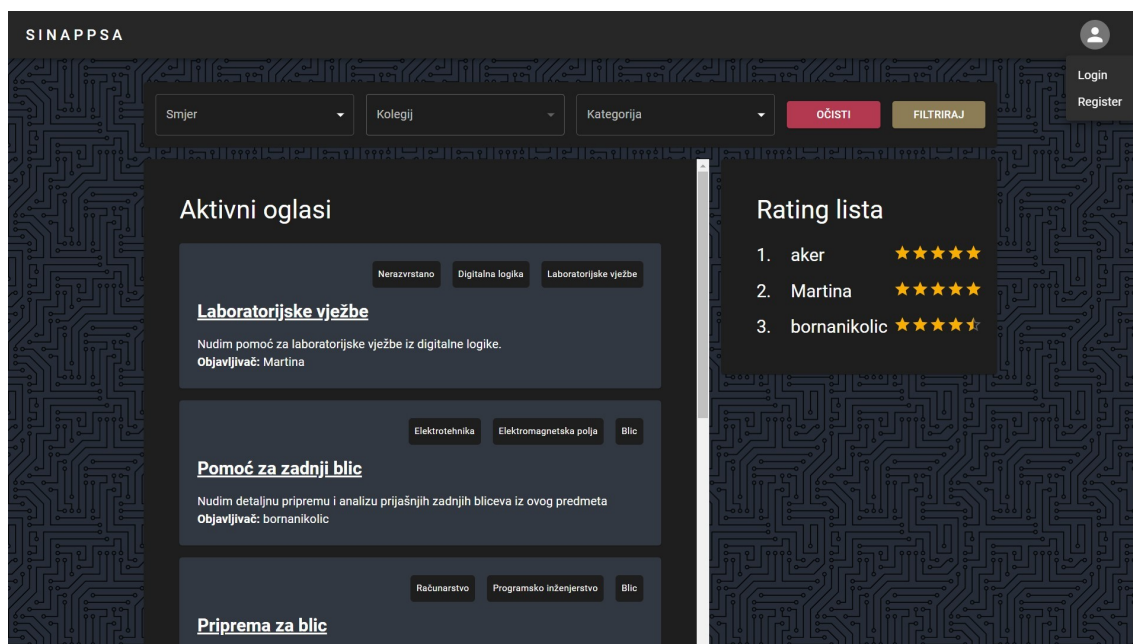
Neregistrirani korisnici

Neregistrirani korisnici na stranici vide objavljene oglase i rejting listu pod korisničkim imenom. Mogu filtrirati oglase po smjeru, kolegiju i kategoriji. Za smjer, padajući izbornik daje opcije elektrotehnika, računarstvo i nerazvrstani, a za kategoriju su moguće opcije laboratorijska vježba, blic, gradivo, kontinuirani ispit i ispitni rok.

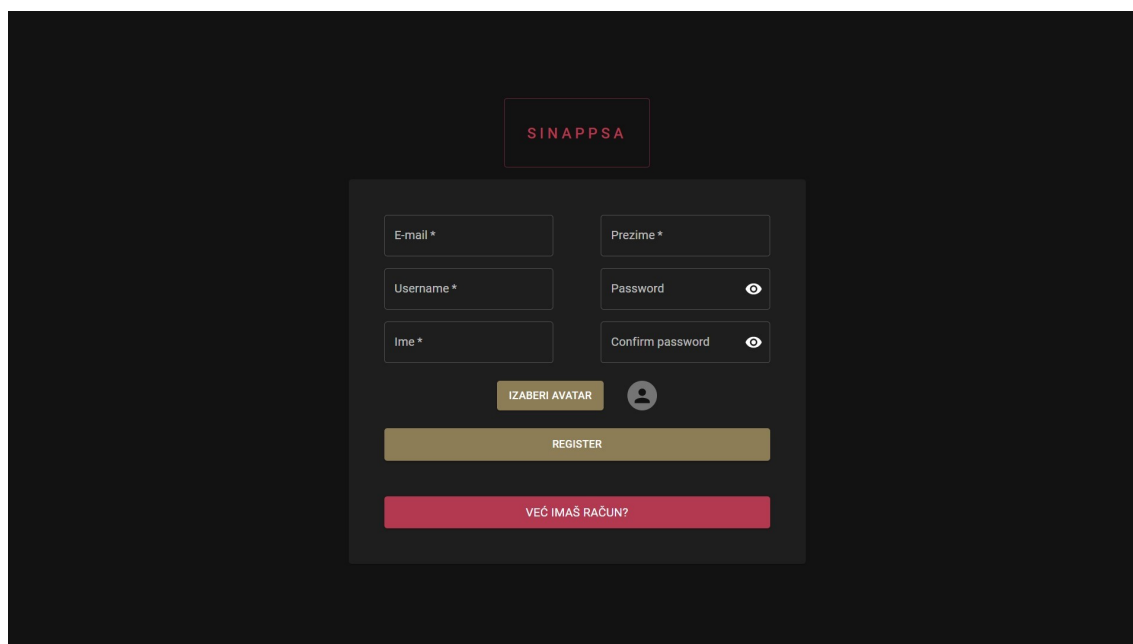
Ovim korisnicima nije dopušteno javljanje na oglase koje vide već se za tu mogućnost moraju registrirati službenom FER e-mail adresom. Opcija registracije ili prijave pojavljuje im se u padajućem izborniku klikom miša na ikonu u gornjem desnom kutu. Odabirom jedne od opcija, odlazi se na formu za registraciju ili prijavu.

Već prije registrirani korisnik ima mogućnost prijave na svoj račun preko nadimka. Ako korisnik koji još nema račun završi na formi za prijavu, klikom na gumb „Nemaš račun?“ korisnik se preusmjerava na formu za registraciju.

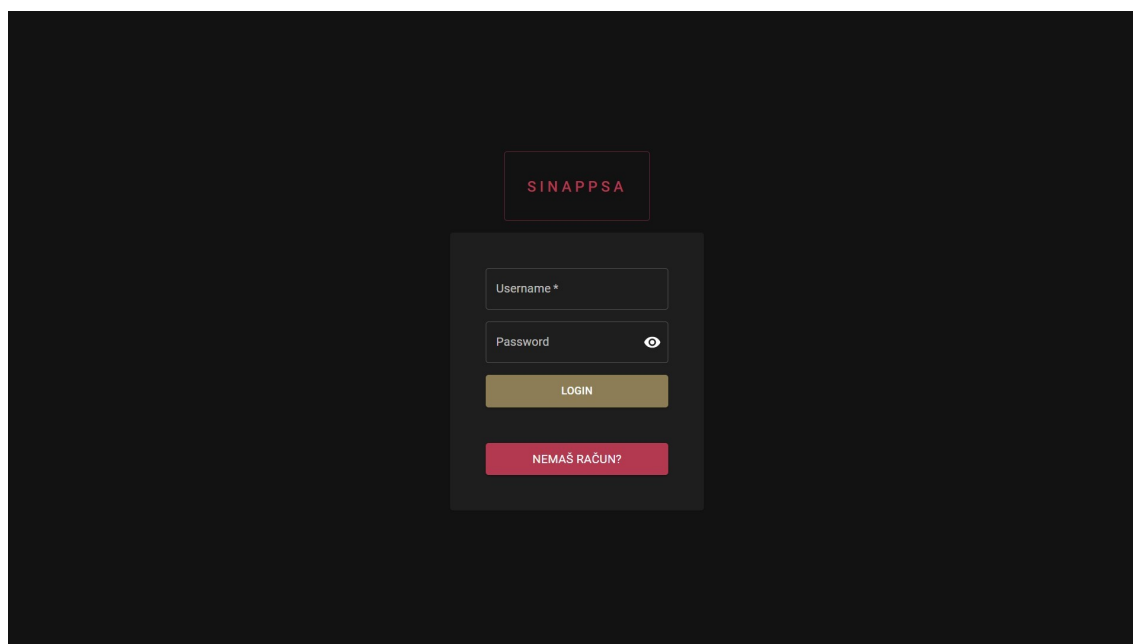
Na formi za registraciju korisnik treba unijeti svoje ime, prezime, korisničko ime, avatara, e-mail i lozinku. Sustav provjerava pripada li e-mail adresa FER domeni. Nakon zahtjeva za registraciju, korisnik klikom na link u primljenom e-mail potvrđuje svoju registraciju.



Slika 2.1: Početna stranica za neregistriranog korisnika



Slika 2.2: Forma za registraciju

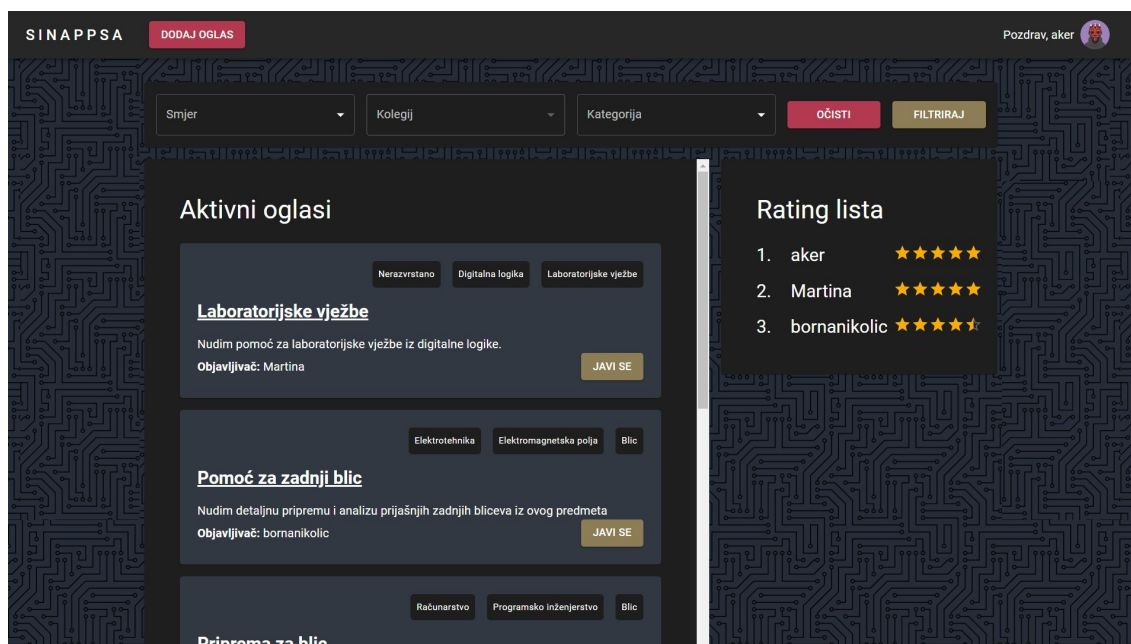


Slika 2.3: Forma za prijavu

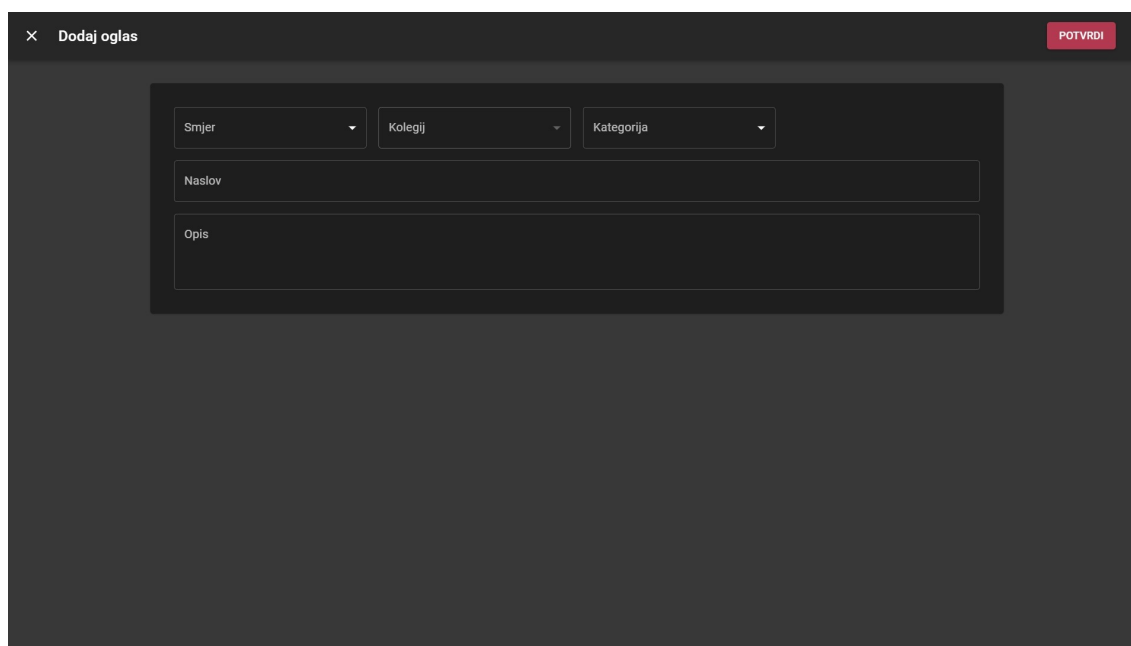
Registrirani korisnici

Registrirani korisnici, uz funkcionalnost neregistriranih korisnika, mogu objavljivati oglase i odgovarati na njih. Opcija stvaranja oglasa pojavljuje se u zaglavlju kao gumb „Dodaj oglas“. Pri stvaranju oglasa korisnik navodi naslov, opis, kolegij i kategoriju oglasa.

Registrirani korisnici također mogu i odgovarati na oglase. Prilikom odgovora na oglas unosi se proizvoljna poruka koja se zajedno s kontakt informacijama korisnika šalje na e-mail adresu studentu-pomagaču. Daljnja komunikacija obavlja se preko e-maila. Upit na oglas može biti prihvaćen, odbijen ili u tijeku. Ako je korisnik dobio pozitivan odgovor, nakon odslušanih instrukcija, osoba ocjenjuje iste te se ta ocjena pribraja u cjelokupni rejting studenta-pomagača.

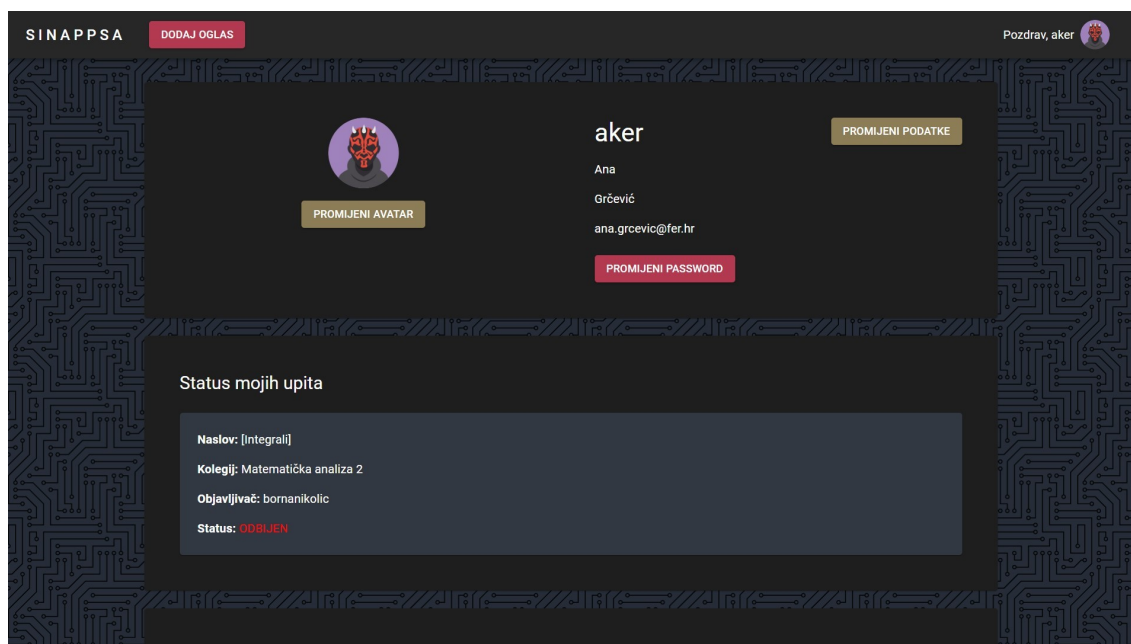


Slika 2.4: Početna stranica za registriranog korisnika

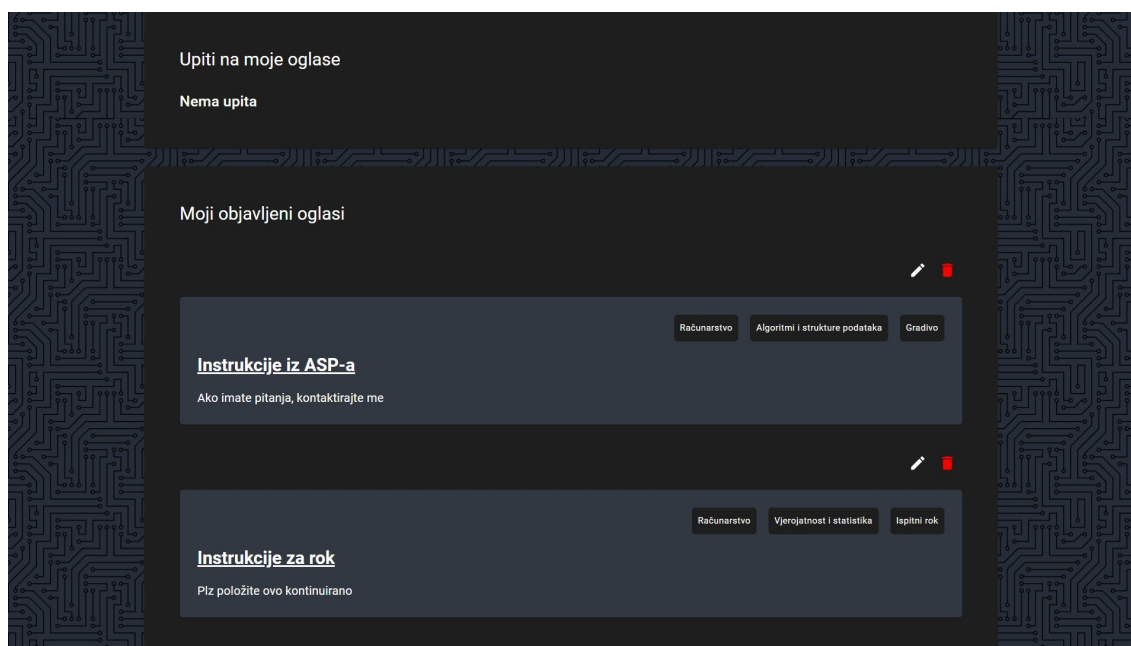


Slika 2.5: Forma za dodavanje oglasa

Na svom profilu, korisniku se prikazuju njegovi aktivni oglasi, upiti na njegove oglase kao i statusi upita koje je korisnik poslao na druge oglase. Aktivnim oglasima može mijenjati naslov i opis te je u mogućnosti obrisati ih. Uz informacije o oglasima na profilu se nalaze i osobne informacije korisnika. Korisniku je omogućeno naknadno mijenjanje lozinke, korisničkog imena i avatara.



Slika 2.6: Izgled profila - prvi dio



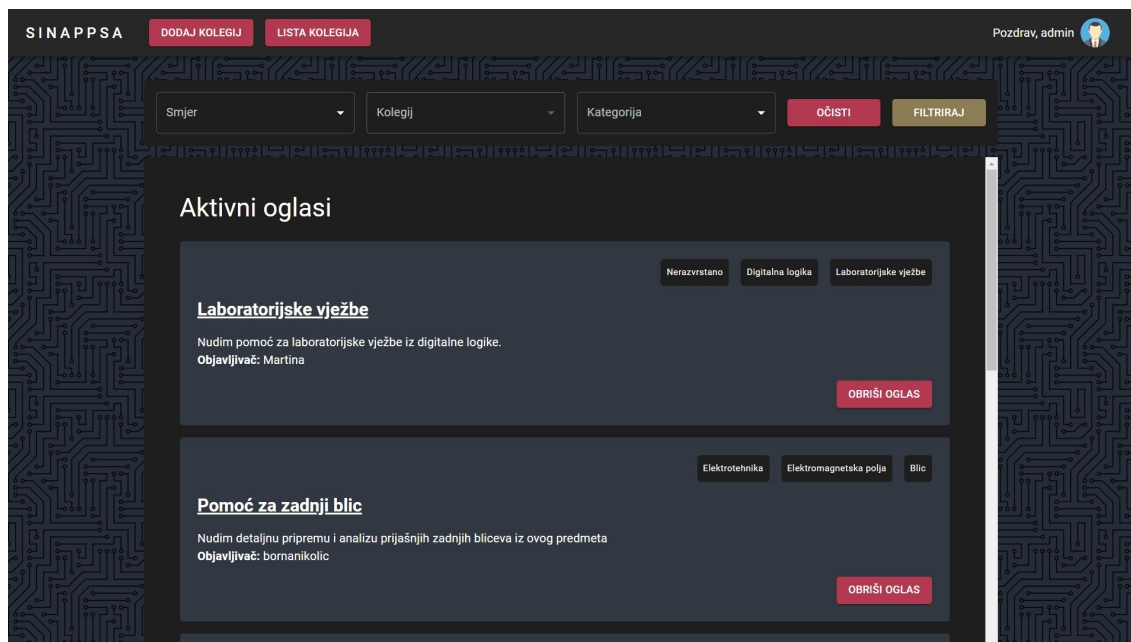
Slika 2.7: Izgled profila - drugi dio

Moderator

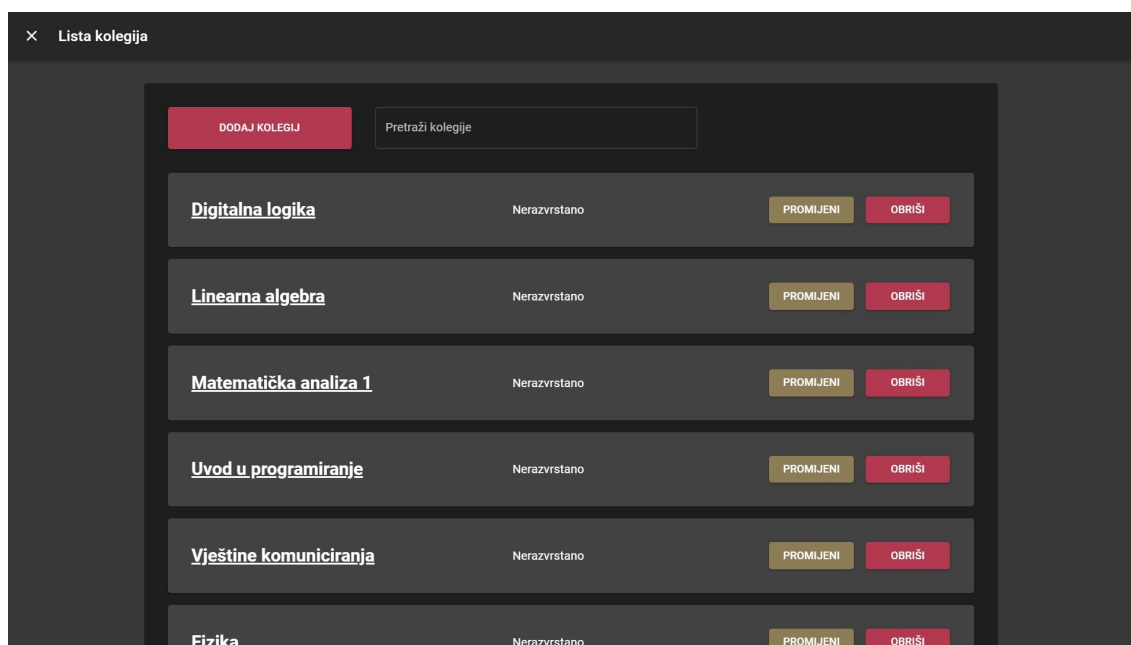
Moderator ima ulogu uklanjanja nepravilnih ili neprikladnih oglasa. Kada moderator odabere opciju brisanja oglasa pojavljuje se skočni prozor u koji upisuje razlog zašto je oglas obrisao te se isti šalje na e-mail adresu objavljiivaču oglasa.

Moderator također ima opciju dodavanja kolegija. Nakon odabira opcije „Dodaj

kolegij“ koja se nalazi u zaglavlju, moderatoru se otvara skočni prozor gdje upisuje naziv kolegija koji želi dodati. Pored opcije ”Dodaj kolegij” nalazi se gumb ”Lista kolegija” čijim pritiskom se moderatoru prikazuje popis postojećih kolegija koje može uređivati, obrisati ili pretraživati.



Slika 2.8: Početna stranica za moderatora



Slika 2.9: Lista dodanih kolegija

3. Specifikacija programske potpore

3.1 Funkcionalni zahtjevi

Dionici:

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

Aktori i njihovi funkcionalni zahtjevi:

1. Neregistrirani/neprijavljeni korisnik (inicijator) može:
 - (a) pregledati objavljene oglase pod nadimkom
 - (b) filtrirati oglase po smjeru, kolegiju i kategoriji
 - (c) pregledati rejting listu studenata- pomagača pod njihovim nadimkom
 - (d) se registrirati u sustav, stvoriti novi korisnički račun za koji su mu potrebni ime, prezime, korisničko ime, avatara, e-mail i lozinka
2. Registrirani korisnik (inicijator) može:
 - (a) javljati se na objavljene oglase
 - (b) objavljivati oglase
 - (c) ocjenjivati studente-pomagače
 - (d) pregledavati i mijenjati osobne podatke
 - (e) pregledavati i mijenjati svoje aktivne oglase
 - (f) obrisati svoje aktivne oglase
 - (g) promijeniti status upita na oglas (prihvaćen ili odbijen)
3. Moderator (inicijator) može:
 - (a) uklanjati nepravilne i neprikladne oglase
 - (b) dodavati, mijenjati i brisati kolegije

(c) pretraživati kolegije

4. Baza podataka (sudionik):

(a) pohranjuje sve podatke o korisnicima

(b) pohranjuje sve podatke o oglasima

(c) pohranjuje kolegije

3.1.1 Obrasci uporabe

Opis obrazaca uporabe

UC1 - Pregled objavljenih oglasa

- **Glavni sudionik:** Neregistrirani korisnik
- **Cilj:** Mogućnost pregleda objavljenih oglasa
- **Sudionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
 1. Dohvaćanje aktivnih oglasa iz baze podataka
 2. Prikaz aktivnih oglasa neregistriranom korisniku
- **Opis mogućih odstupanja:**
 - 1.a Nema aktivnih oglasa
 1. Sustav obavještava korisnika da nema aktivnih oglasa

UC2 - Pregled rejting-liste

- **Glavni sudionik:** Neregistrirani korisnik
- **Cilj:** Mogućnost pregleda rejting-liste studenata pomagača pod nadimcima
- **Sudionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
 1. Neregistrirani korisnik zahtjeva prikaz liste 10 najboljih student-pomagača
 2. Lista student-pomagača dohvaća se iz baze podataka
 3. Rangiranje studenata-pomagača unutar web aplikacije
 4. Prikaz liste neregistriranom korisniku
- **Opis mogućih odstupanja:**
 - 2.a Nema registriranih student-pomagača
 1. Sustav obavještava korisnika da ne postoje registrirani student-pomagači
 - 3.a Studente-pomagače nije moguće rangirati
 1. Sustav prikazuje praznu rejting listu

UC3 - Filtriranje oglasa

- **Glavni sudionik:** Neregistrirani korisnik
- **Cilj:** Mogućnost filtriranja oglasa po smjeru, kolegiju i kategoriji
- **Sudionici:** Baza podataka, Web aplikacija

- **Preduvjet:** Korisniku se prikazuju aktivni oglasi
- **Opis osnovnog tijeka:**
 1. Korisnik odabire opciju smjer(računarstvo ili elektrotehnika), kolegij i/ili kategoriju(laboratorijska vježba, blic, gradivo, kontinuirani ispit ili ispitni rok)
 2. Korisnik odabire opciju „Filtriraj“
 3. Web aplikacija filtrira listu oglasa po danim kriterijima i osvježava prikaz filtriranih oglasa
- **Opis mogućih odstupanja:**
 - 2.a Korisnik filtrira, a nije izabrao nijednu od opcija (smjer, kolegij, kategorija):
 1. Prikazuju se svi oglasi

UC4 - Registracija

- **Glavni sudionik:** Neregistrirani korisnik
- **Cilj:** Stvoriti korisnički račun za pristup sustavu
- **Sudionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
 1. Korisnik odabire opciju za registraciju
 2. Korisnik unosi podatke potrebne za registraciju
 3. Aplikacija provjerava ispravni format unesenih podataka
 4. Podaci se upisuju u bazu podataka
 5. Korisniku se šalje e-mail pošta za potvrdu registracije računa
- **Opis mogućih odstupanja:**
 - 2.a Korisnik je unio već zauzeto korisničko ime i/ili e-mail ili je unio e-mail koji nije iz fer.hr domene
 1. Sustav obavještava korisnika o pogrešci te briše polje lozinke

UC5 - Pregled svojih objavljenih oglasa

- **Glavni sudionik:** Registrirani korisnik
- **Cilj:** Pregledati sve svoje objavljene oglase
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen
- **Opis osnovnog tijeka:**
 1. Korisnik odlazi na rubriku „Moji objavljeni oglasi“

2. Aplikacija šalje upit bazi za prikaz svih aktivnih oglasa
3. Aktivni oglasi prikazuju se korisniku
- **Opis mogućih odstupanja:**
 - 2.a Nema aktivnih oglasa za prikaz
 1. Aplikacija ispisuje poruku „Nema oglasa za prikaz“

UC6 - Objava oglasa

- **Glavni sudionik:** Registrirani korisnik
- **Cilj:** Objaviti oglas
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen
- **Opis osnovnog tijeka:**
 1. Korisnik odabire opciju „Objavi oglas“
 2. Unosi sadržaj svog oglasa (naslov, opis, kolegij, kategorija)
 3. Podaci se upisuju u bazu podataka
- **Opis mogućih odstupanja:**
 - 2.a Korisnik nije unio sve obvezne komponente oglasa
 1. Sustav obavještava korisnika o podacima koji nisu ispravno uneseni

UC7 - Izmjena objavljenih oglasa

- **Glavni sudionik:** Registrirani korisnik
- **Cilj:** Izmijeniti značajke oglasa
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen i oglas je objavljen
- **Opis osnovnog tijeka:**
 1. Korisnik odlazi na rubriku „Moji objavljeni oglasi“
 2. Korisnik odabire oglas koji želi izmijeniti
 3. Korisnik odabere opciju za izmjenu oglasa
 4. Korisnik izmjenjuje željene značajke oglasa
 5. Korisnik sprema promjene
 6. Baza podataka se ažurira
- **Opis mogućih odstupanja:**
 - 4.a Korisnik je obrisao sav sadržaj oglasa i pokušava ga ponovno objaviti
 1. Sustav obavještava korisnika da ne može objaviti prazan oglas

UC8 - Brisanje oglasa

- **Glavni sudionik:** Registrirani korisnik
- **Cilj:** Obrisati objavljeni oglas
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen i oglas je objavljen
- **Opis osnovnog tijeka:**
 1. Korisnik odlazi na rubriku „Moji objavljeni oglasi“
 2. Korisnik odabire oglas koji želi obrisati
 3. Korisnik stisne gumb „Obriši oglas“
 4. Sustav izbacuje skočni prozor s pitanjem „Jeste li sigurni da želite obrisati ovaj oglas?“
 5. Korisnik odabire „Da“ ili „Ne“
 6. Baza podataka se ažurira

UC9 - Odgovaranje na oglase

- **Glavni sudionik:** Registrirani korisnik
- **Cilj:** Odgovoriti na objavljeni oglas
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen i oglas je objavljen
- **Opis osnovnog tijeka:**
 1. Korisnik bira oglas na koji želi odgovoriti
 2. Korisnik upisuje svoj odgovor
 3. Sustav šalje e-mail objavljiivaču oglasa
 4. Daljnja komunikacija odvija se e-mailom između korisnika i studenta-pomagača

UC10 - Pregled statusa upita

- **Glavni sudionik:** Registrirani korisnik
- **Cilj:** Pogledati status poslanog upita
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen i poslan je upit na oglas
- **Opis osnovnog tijeka:**
 1. Korisnik odlazi na rubriku „Status mojih upita“
 2. Aplikacija šalje upit bazi za prikaz statusa upita
 3. Korisniku se prikazuje status njegovog upita (prihvaćen, odbijen ili u tijeku)

UC11 - Promjena statusa upita

- **Glavni sudionik:** Registrirani korisnik
- **Cilj:** Promijeniti status primljenog upita
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen i primljen je upit na oglas
- **Opis osnovnog tijeka:**
 1. Korisnik odlazi na rubriku „Upiti na moje oglase“
 2. Aplikacija šalje upit bazi za prikaz upita
 3. Korisniku se prikazuje upiti na njegove oglase
 4. Korisnik prihvaća ili odbija upit na oglas
 5. Baza podataka se ažurira

UC12 - Ocjenjivanje pomagača

- **Glavni sudionik:** Registrirani korisnik
- **Cilj:** Ocijeniti studenta-pomagača
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen i objavljen je bar jedan oglas na temelju kojeg se pomagač ocjenjuje
- **Opis osnovnog tijeka:**
 1. Korisnik nakon pozitivnog odgovora ocjenjuje studenta-pomagača
 2. Korisnik odabire ocjenu koju mu želi dati
 3. Ažurira se rejting-lista i srednja ocjena studenta-pomagača
 4. Baza podataka se ažurira

UC13 - Izmjena informacija o profilu

- **Glavni sudionik:** Registrirani korisnik
- **Cilj:** Izmijeniti informacije o svom profilu
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen
- **Opis osnovnog tijeka:**
 1. Korisnik odabire opciju za promjenu svojih podataka
 2. Korisnik mijenja svoje podatke
 3. Korisnik sprema promjene
 4. Baza podataka se ažurira

UC14 - Uklanjanje nepravilnih oglasa

- **Glavni sudionik:** Moderator

- **Cilj:** Ukloniti nepravilne ili neprikladne oglase
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen i dodijeljena su mu prava moderatora
- **Opis osnovnog tijeka:**
 1. Moderator pregledava oglase i utvrđuje nepravilnosti
 2. Moderator piše objašnjenje za brisanje oglasa
 3. Moderator odabire opciju „Obriši oglas”
 4. Korisniku se šalje e-mail s objašnjenjem zašto je njegov oglas uklonjen
 5. Baza podataka se ažurira
- **Opis mogućih odstupanja:**
 - 1.a Nema aktivnih oglasa
 1. Sustav obavještava moderatora da nema aktivnih oglasa

UC15 - Dodavanje kolegija

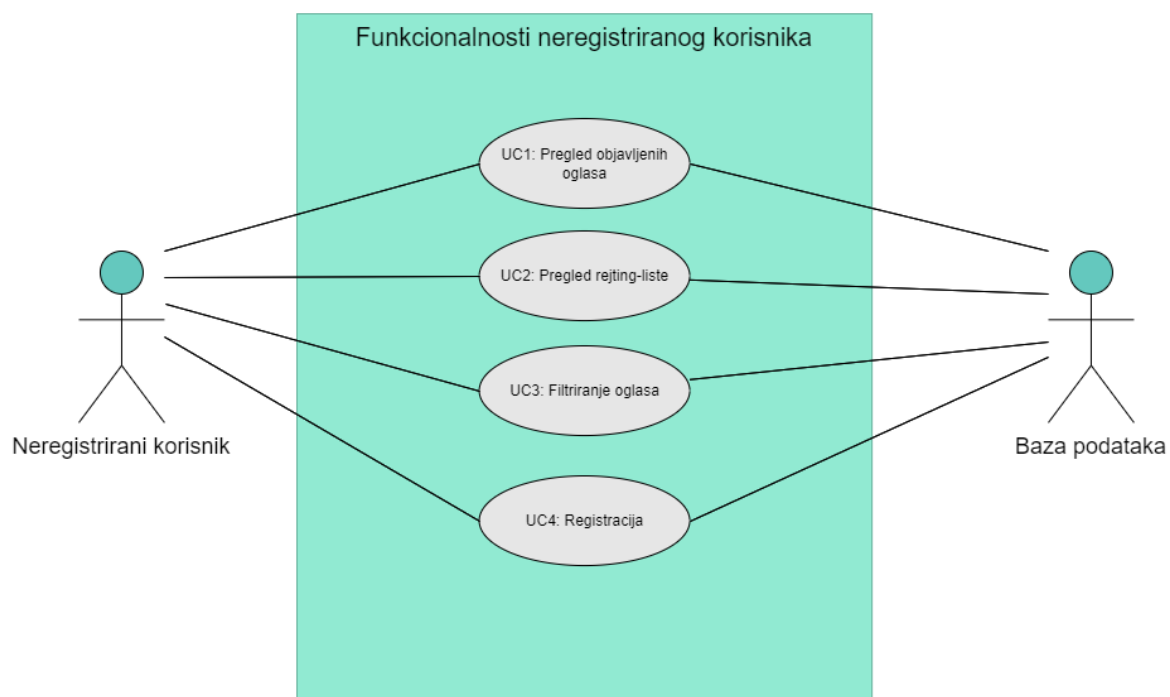
- **Glavni sudionik:** Moderator
- **Cilj:** Dodati novi kolegij
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen i dodijeljena su mu prava moderatora
- **Opis osnovnog tijeka:**
 1. Moderator odabire opciju „Dodaj kolegija”
 2. Moderator unosi podatke o novom kolegiju
 3. Moderator odabire opciju „Unesi kolegij”
 4. Podaci se upisuju u bazu podataka
- **Opis mogućih odstupanja:**
 - 3.a Kolegij već postoji u bazi podataka
 1. Sustav obavještava moderatora o neuspješnom unosu kolegija u bazu podataka

UC16 - Kategorizacija kolegija

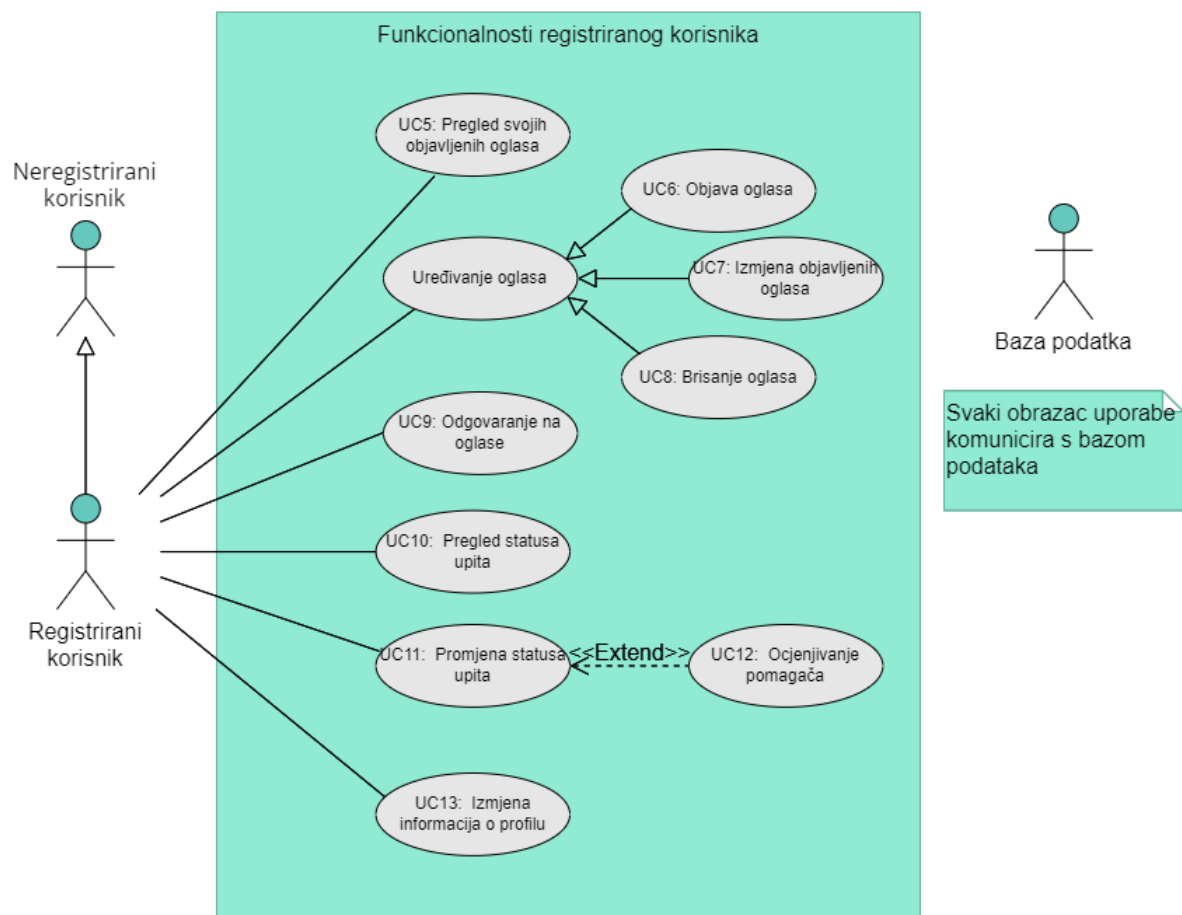
- **Glavni sudionik:** Moderator
- **Cilj:** Kategorizirati kolegij po smjerovima
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen i dodijeljena su mu prava moderatora, kolegij postoji u bazi podataka
- **Opis osnovnog tijeka:**
 1. Moderator odabire prikaz liste kolegija

2. Moderator odabire kolegij iz liste kolegija
3. Moderator odabire smjer (jedan od preddiplomskih ili diplomskih smjerala)
4. Podaci se upisuju u bazu podataka

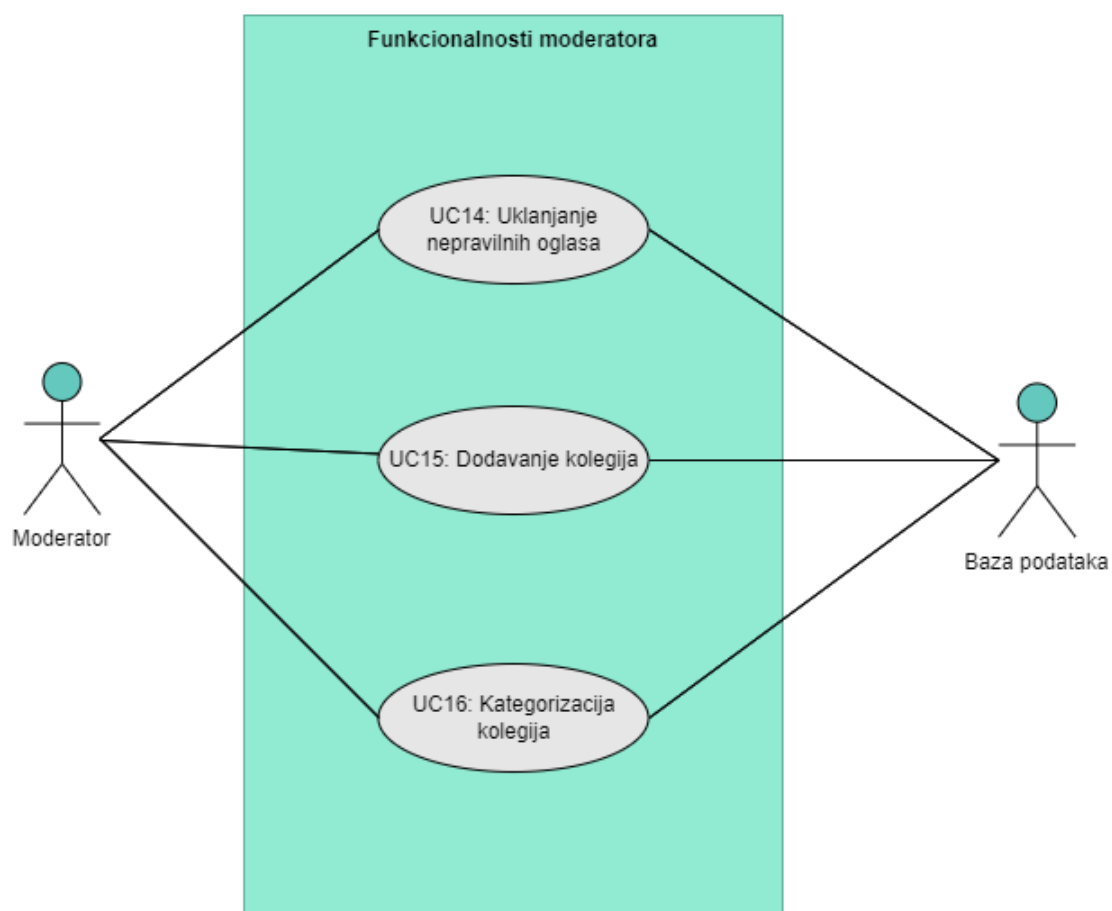
Dijagrami obrazaca uporabe



Slika 3.1: Dijagram obrasca uporabe, funkcionalnosti neregistriranog korisnika



Slika 3.2: Dijagram obrasca uporabe, funkcionalnosti registriranog korisnika

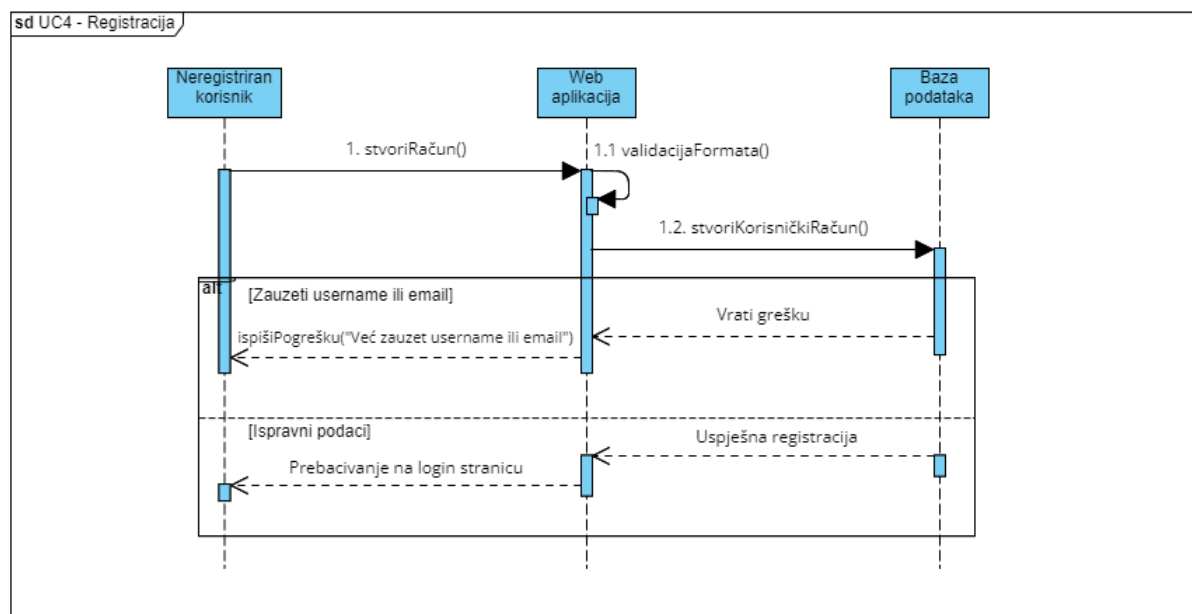


Slika 3.3: Dijagram obrasca uporabe, funkcionalnosti moderatora

3.1.2 Sekvencijski dijagrami

Obrazac uporabe UC4 - Registracija

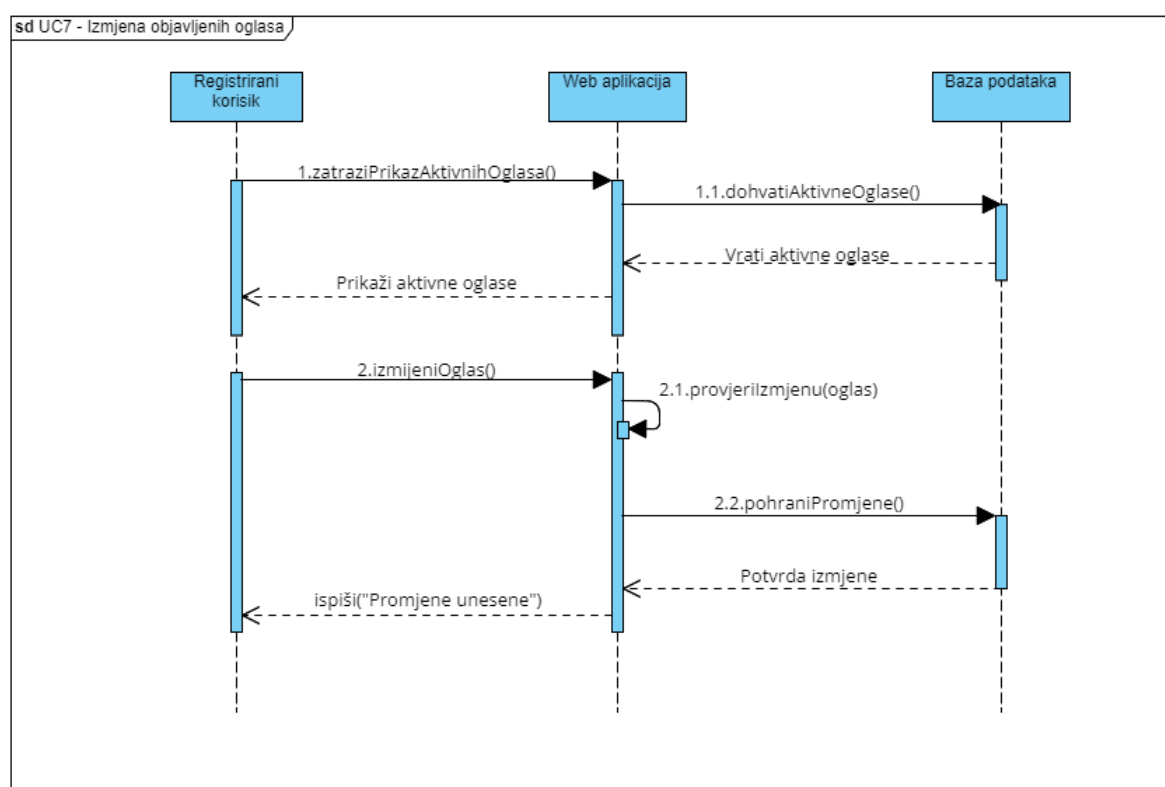
Korisnik otvara formu za registraciju i upisuje svoje podatke. Poslužitelj provjerava je li čitava forma popunjena i nalazi li se e-mail u FER-ovoj domeni. Poslužitelj pohranjuje podatke u bazu podataka. Ukoliko su zauzeti korisničko ime ili e-mail adresa sustav obavještava korisnika o tome i ispisuje poruku.



Slika 3.4: Sekvencijski dijagram za UC4

Obrazac uporabe UC7 - Izmjena objavljenih oglasa

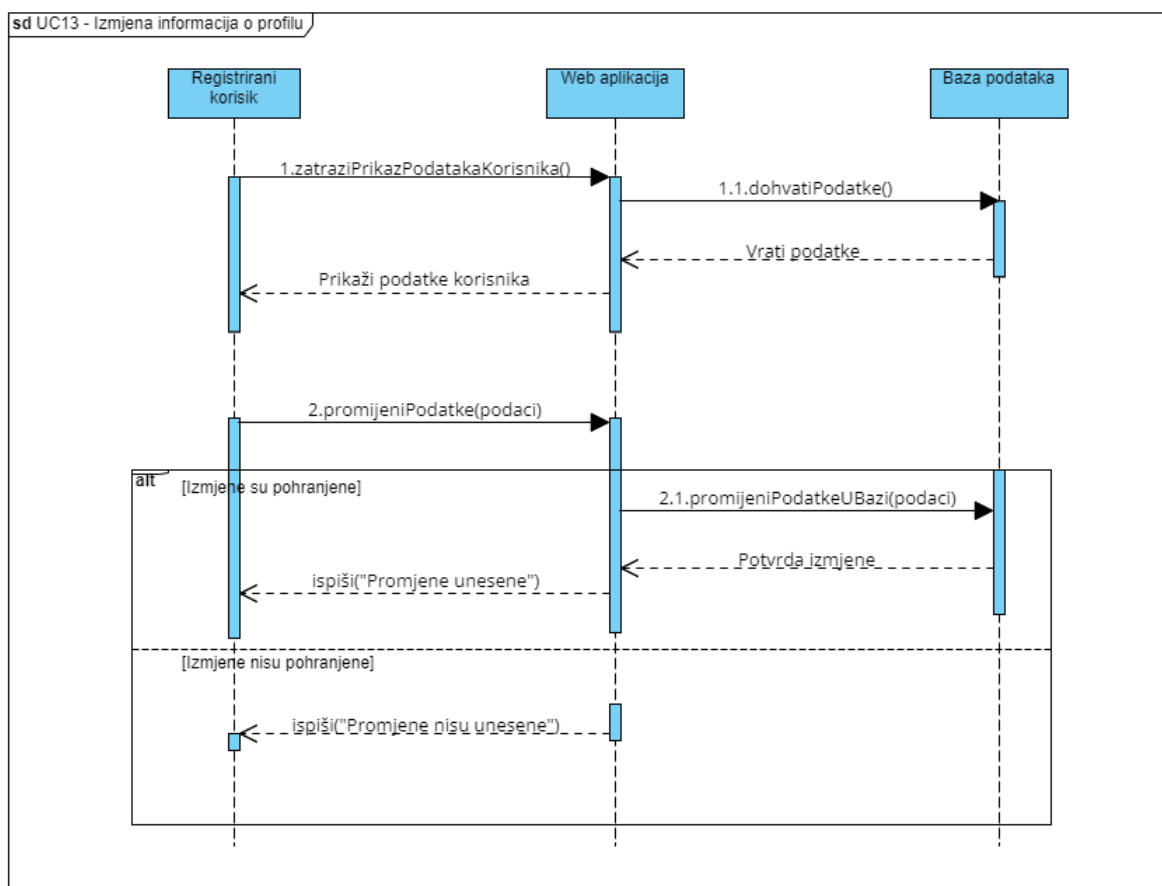
Korisnik šalje zahtjev za prikaz aktivnih oglasa. Poslužitelj dohvaća oglase i prikazuje ih korisniku. Korisnik mijenja neke od komponenata oglasa. Poslužitelj provjerava je li koja od komponenata prazna, tj. izbrisana. Ako je promjena validna, ona se pohranjuje u bazu podataka te korisnik dobiva poruku „Promjene unesene“.



Slika 3.5: Sekvencijski dijagram za UC7

Obrazac uporabe UC13 - Izmjena informacija o profilu

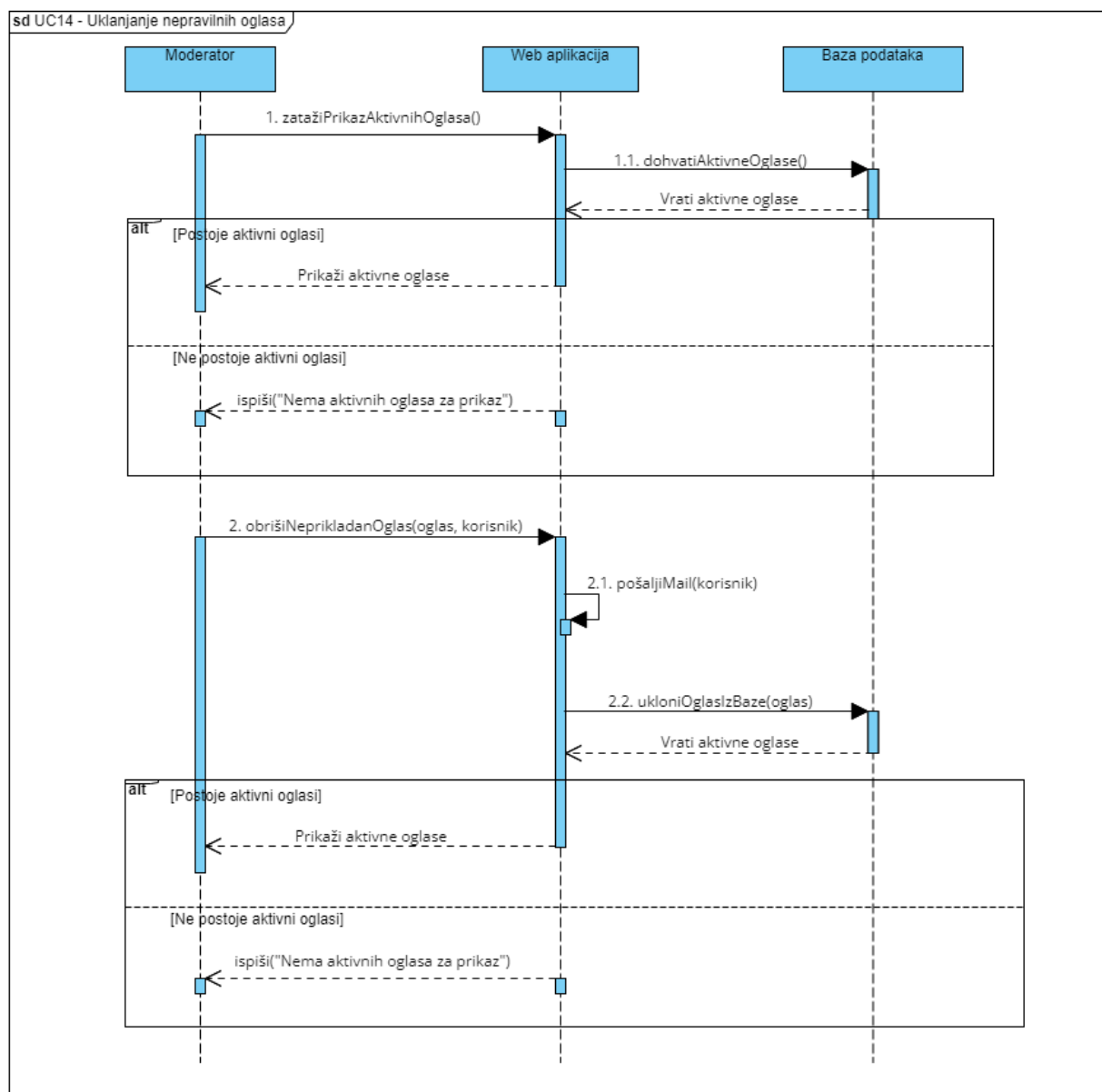
Korisnik šalje zahtjev za prikaz aktivnih oglasa. Poslužitelj dohvaća oglase i prikazuje ih korisniku. Korisnik mijenja neke od komponenata oglasa. Poslužitelj provjerava je li koja od komponenata prazna, tj. izbrisana. Ako je promjena validna, ona se pohranjuje u bazu podataka te korisnik dobiva poruku „Promjene unesene“. Ukoliko dođe do greške u obradi zahtjeva sustav obavještava korisnika da promjene nisu unesene.



Slika 3.6: Sekvencijski dijagram za UC13

Obrazac uporabe UC14 - Uklanjanje nepravilnih oglasa

Moderator šalje zahtjev za prikaz aktivnih oglasa. Poslužitelj dohvaća oglase i prikazuje ih korisniku. Ukoliko ne postoje aktivni oglasi, moderator dobiva poruku „Nema aktivnih oglasa za prikaz“. Moderator briše oglas i piše razlog zašto je oglas obrisao. Razlog se šalje korisniku na mail te se oglas briše iz baze podataka. Sustav moderatoru vraća prikaz aktivnih oglasa. Ukoliko su svi oglasi obrisani, moderator dobiva poruku „Nema aktivnih oglasa za prikaz“.



Slika 3.7: Sekvencijski dijagram za UC14

3.2 Ostali zahtjevi

- Sustav mora biti napravljena kao web aplikacija koristeći objektno-orijentirane jezike
- Aplikacija mora biti prilagodljiva uređajima različitih veličina
- Korisničko sučelje mora biti jednostavno za intuitivno korištenje svim korisnicima
- Korisničko sučelje i sustav moraju podržavati hrvatsku abecedu pri unosu i prikazu tekstualnog sadržaja
- Aplikacija treba omogućiti rad više korisnika u stvarnom vremenu
- Neispravno korištenje korisničkog sučelja ne smije narušiti funkcionalnost i rad sustava
- Izvršavanje dijela programa u kojem se pristupa bazi podataka ne smije trajati duže od nekoliko sekundi
- Slanje e-maila (verifikacijski e-mail ili obrazloženje brisanja oglasa) korisniku ne smije trajati duže od 10 sekundi

4. Arhitektura i dizajn sustava

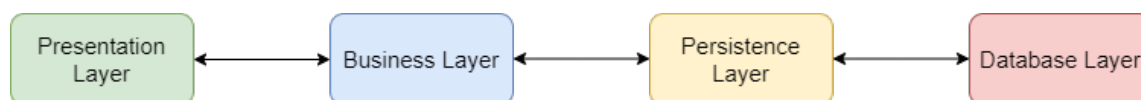
Glavni dijelovi arhitekture sustava su web preglednik, web poslužitelj i baza podataka.

Web preglednik omogućuje korisniku da pronađe određenu stranicu. Njegova funkcija je prevođenje stranica u svima razumljiv sadržaj. Neki od popularnih web preglednika su Google Chrome, Safari, Microsoft Edge, Opera.

Web poslužitelj je softverski program koji ima zadatak da pohranjuje, obrađuje i dostavlja klijentima web stranice. Komunikacija između web preglednika i klijenta odvija se HTTPS protokolom.

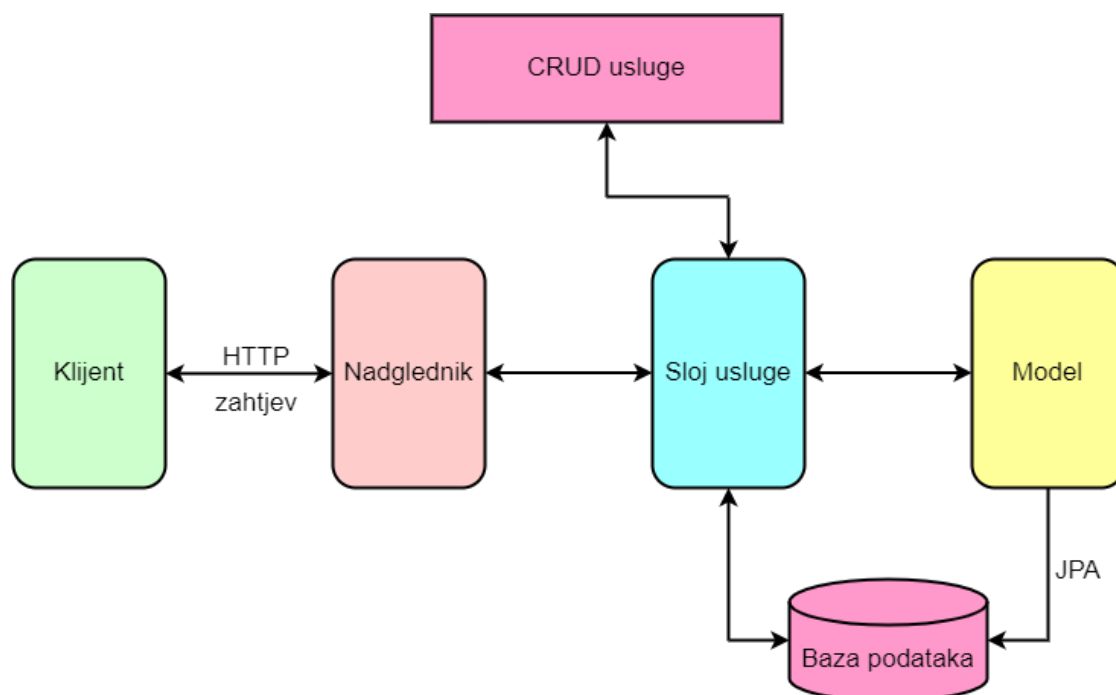
Programski jezik koji smo odabrali za implementaciju ove web aplikacije je radni okvir Spring Boot koji koristi višeslojnu arhitekturu. Spring Boot slijedi slojevitost arhitekture u kojoj svaki sloj komunicira sa slojem neposredno ispod ili iznad njega. Postoje četiri sloja u Spring Boot arhitekturi, a to su:

- **Prezentacijski sloj** (engl. *Presentation Layer*) - Ovo je najviši sloj arhitekture. Sastoji se od front-end dijela aplikacije. Obrađuje HTTP zahtjeve, provjerava autentičnost zahtjeva te pretvara JSON parametar u Java objekt. Nakon što provjeri autentičnost zahtjeva, dalje ga prosljeđuje poslovnom sloju.
- **Poslovni sloj** (engl. *Business Layer*) - Ovaj sloj obrađuje svu poslovnu logiku te je zaslužan za autorizaciju i validaciju.
- **Sloj postojanosti** (engl. *Persistence Layer*) - Sadrži svu logiku pohrane te prevodi poslovne objekte iz i u bazu podataka.
- **Sloj baze podataka** (engl. *Database Layer*) - Može se sastojati od više baza podataka. Zaslužan za provođenje CRUD (engl. *Create, Read/Retrieve, Update and Delete*) operacija.



Slika 4.1: Slojevi Spring Boota

Komunikacija u Spring Bootu prikazana je na slici 4.2. Ona započinje kada klijent postavi HTTPS zahtjev koji se dalje šalje nadgledniku. Nadglednik mapira taj zahtjev i obrađuje ga. U sloju usluge obavlja se sva poslovna logika. Spring Boot izvodi svu logiku nad podacima baze podataka koja je preslikana u klasu modela putem Java Persistence biblioteke. U posljertku, ako nije došlo do pogreške, Java Spring Boot stranica se prikazuje klijentu.



Slika 4.2: Komunikacija u Spring Bootu

Uz Spring Boot koristili smo React koji je JavaScript biblioteka za izradu korisničkih sučelja. Primarna svrha Reacta je omogućiti razvojnom programeru stvaranje korisničkog sučelja koristeći čisti JavaScript.

Kao razvojne okoline koristili smo IntelliJ IDEA za rad sa Spring Bootom te Visual Studio Code za rad u Reactu.

4.1 Baza podataka

Za potrebe aplikacije Sinappsa koristit ćemo relacijsku bazu podataka. Osnovna jedinica baze je relacija koja se definira svojim imenom i skupom atributa. Baza podataka pomaže u pohrani, izmjeni i dohvatu podataka koji su nam potrebni za obradu zahtjeva. Baza podataka aplikacije Sinappsa sastoji se od sljedećih entiteta:

- Korisnik
- Profil
- Kolegij
- Smjer
- Kategorija
- Oglas
- Upit

4.1.1 Opis tablica

Korisnik

Ovaj entitet sadrži osnovne informacije o korisniku aplikacije. Sadrži jedinstveni identifikator korisnika, njegov email, lozinku i korisničko ime. Osim ovih atributa ova tablica sadrži i boolean zastavicu moderator koja označava je li korisnik moderator sustava ili nije.

Korisnik		
id	BIGINT	korisnikov jedinstveni id
email	VARCHAR	korisnikov e-mail
lozinka	VARCHAR	korisnikova lozinka
username	VARCHAR	korisnikovo korisničko ime
moderator	BOOLEAN	oznaka je li korisnik moderator

Profil

Ovaj entitet sadrži ostale informacije o korisniku i u odnosu je One-To-One sa tablicom Korisnik. Atributi koje ova tablica sadrži su id, ime, prezime, prosječna ocjena, avatar profila, zastavicu potvrđena registracija i strani ključ tablice korisnik idKorisnika.

Profil		
id	BIGINT	jedinstvena oznaka profila
ime	VARCHAR	ime korisnika
prezime	VARCHAR	prezime korisnika
ocjena	VARCHAR	prosječna ocjena studenta-pomagača
potvrđenaRegistracija	BOOLEAN	oznaka o potvrđenoj registraciji
avatar	VARCHAR	slika profila
idKorisnika	BIGINT	referenca na tablicu Korisnik

Kolegij

Entitet Kolegij sadrži informacije o kolegijima u sustavu. Sadrži attribute: id, naziv kolegija i referencu na tablicu Smjer koja označava na kojem smjeru se predaje kolegij. Relacija Kolegij je u odnosu Many-To-One sa relacijom Smjer.

Kolegij		
id	BIGINT	jedinstvena oznaka kolegija
nazivKolegija	VARCHAR	naziv kolegija
idSmjera	BIGINT	referenca na relaciju Smjer

Smjer

Entitet Smjer sadrži dva atributa, id i nazivSmjera. Ova tablica pohranjuje smjerove koji se predaju na FER-u kako bi se kolegiji mogli svrstati po tim smjerovima.

Smjer		
id	BIGINT	jedinstvena oznaka smjera
nazivSmjera	VARCHAR	naziv smjera

Kategorija

Entitet Kategorija pohranjuje informacije o različitim kategorijama za koje se može napraviti oglas. Atributi u ovoj tablici su id i nazivKategorije. Mogući nazivi kategorija su: laboratorijska vježba, blic, gradivo, kontinuirani ispit, ispitni rok.

Kategorija		
id	BIGINT	jedinstvena oznaka kategorije
nazivKategorije	VARCHAR	naziv kategorije

Oglas

Ovaj entitet predstavlja oglas koji student-pomagač objavljuje u aplikaciji. Sadrži attribute id, naslov, opis i reference na relacije Kategorija, Kolegiji i Profil. Pomoću reference na relaciju Kategorija oglasu se određuje u koju kategoriju oglasa spada: laboratorijska vježba, blic, gradivo, kontinuirani ispit, ispitni rok. Određeni oglas objavljen je za određeni kolegij te se ta informacija pamti pomoću reference na relaciju Kolegij. Također pomoću reference na relaciju profil pamti se informacija koji student-pomagač je objavio oglas. Sa entitetima Kategorija, Kolegij, Profil ovaj entitet ima odnos One-To-One.

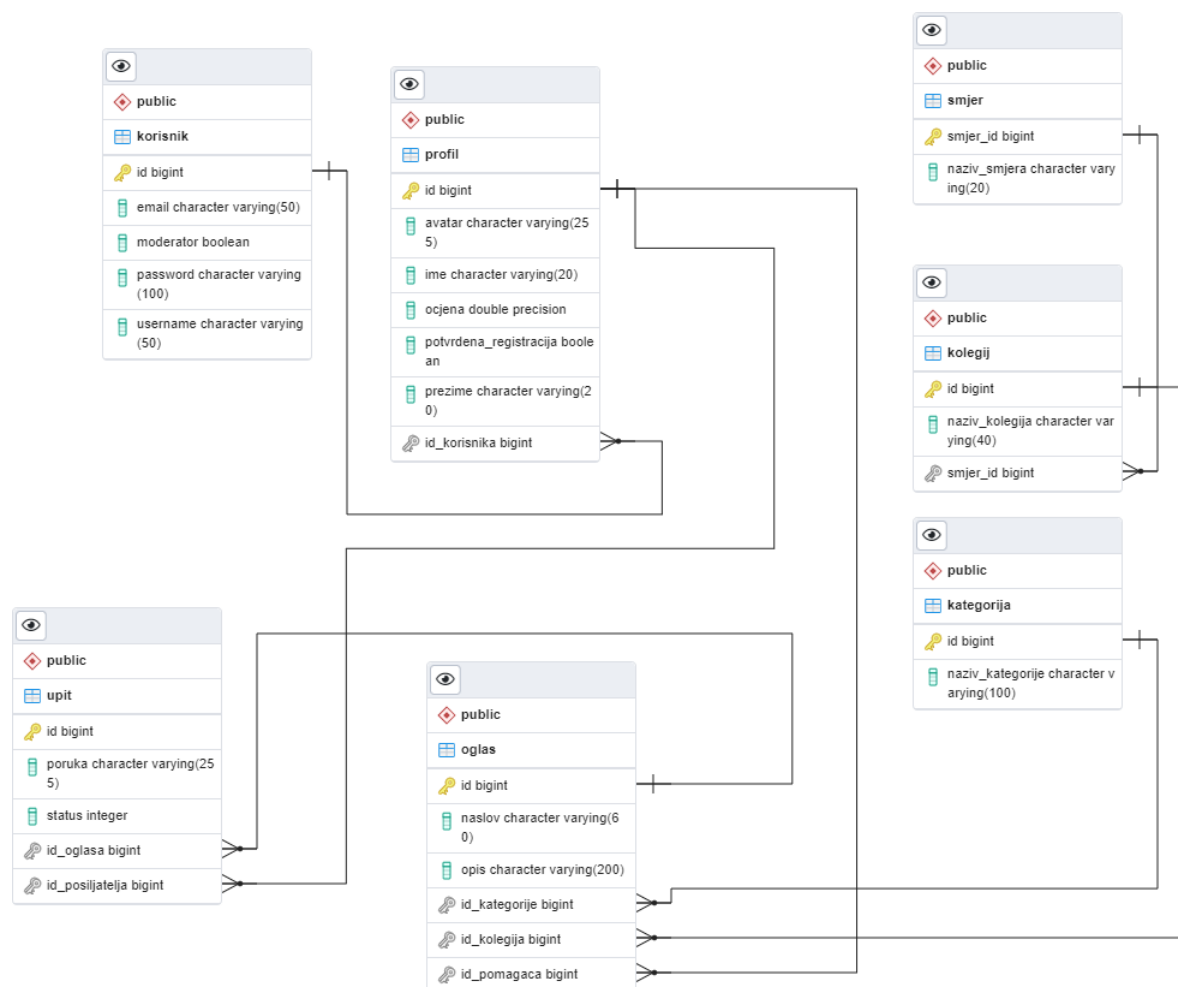
Oglas		
id	BIGINT	jedinstvena oznaka oglasa
naslov	VARCHAR	naslov oglasa
opis	VARCHAR	opis oglasa
idKategorije	BIGINT	referenca na relaciju Kategorija
idKolegija	BIGINT	referenca na relaciju Kolegij
idPomagača	BIGINT	referenca na relaciju Profil

Upit

Entitet upit predstavlja sve upite koje studenti pošalju na jedan oglas. Sadrži sljedeće atribute: id, poruka, statusUpita i reference na relacije Oglas i Profil. Atribut id je jedinstveni identifikator upita. Atribut poruka sadrži poruku koju student koji se javlja na oglas šalje studentu-pomagaču koji je objavio oglas. Atribut statusUpita može imati jednu od 3 vrijednosti: 0 – U tijeku (upit čeka odgovor studenta pomagača), 1 – Odbijen (student-pomagač je odbio upit) i 2 – Prihvaćen (student-pomagač je odobrio upit). Pomoću reference na relaciju Oglas baza pamti za koji oglas student šalje upit, a pomoću reference na relaciju Profil pamti se koji student šalje upit. S obje relacije veza je One-To-One.

Upit		
id	BIGINT	jedinstvena oznaka upita
poruka	VARCHAR	poruka koju šalje pošiljatelj
statusUpita	INTEGER	status upita
idOglas	BIGINT	referenca na relaciju Oglas
idPošiljatelja	BIGINT	referenca na relaciju Profil

4.1.2 Dijagram baze podataka

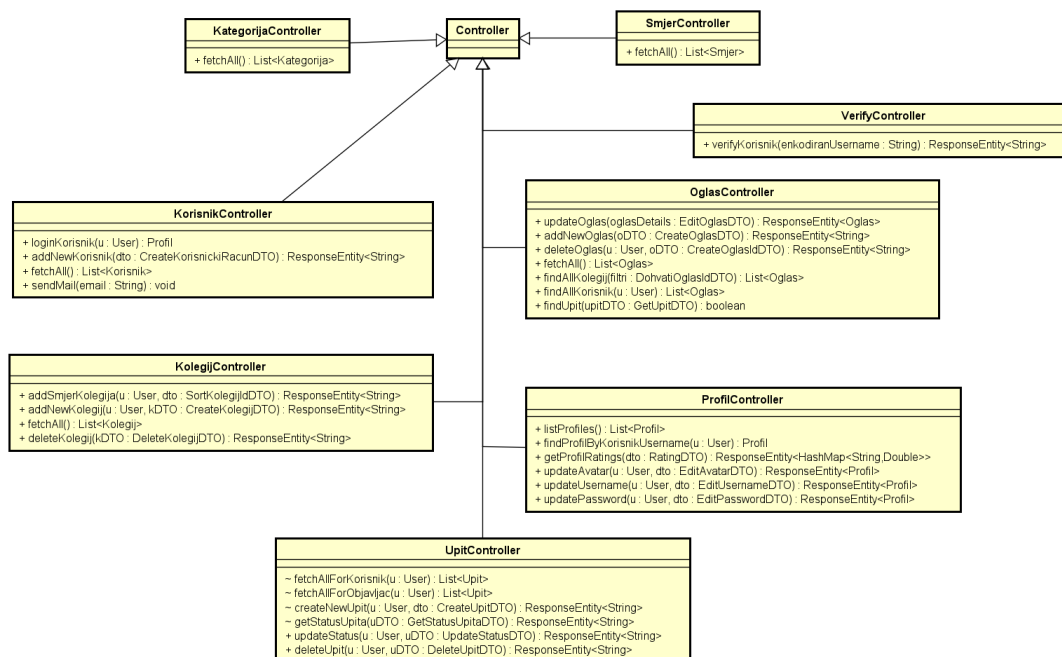


Slika 4.3: Dijagram baze podataka

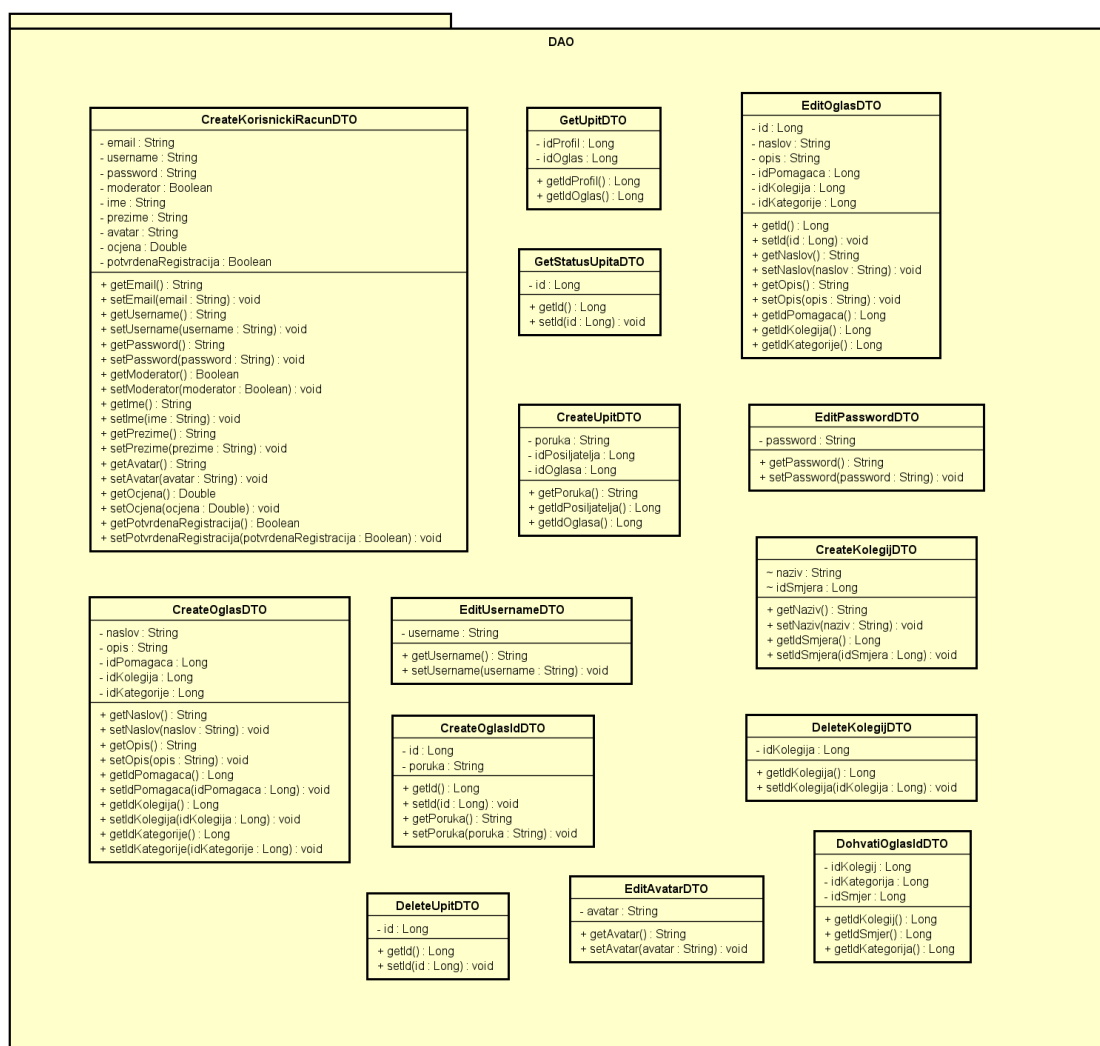
4.2 Dijagram razreda

Na slikama 4.4, 4.5 i 4.6 prikazani su razredi koji pripadaju backend dijelu MVC arhitekture. Razredi su podijeljeni u tri dijela: Controllers, DAO i Modele. Razredi u Controllers dijelu nasljeđuju razred Controller i upravljaju pristiglim zahtjevima na backend dio aplikacije. Metode implementirane u tim razredima manipuliraju s DAO (Data Access Object) razredima. DAO razredi su modelirani i dohvaćeni pomoću metoda koje su implementirane u Model razredima. Povratne vrijednosti metoda u Controller razredima su JSON datoteke s HTML status kodom.

Zbog lakšeg prikaza i organizacije, razredi su logički podijeljeni po pravu pristupa metodama određenih aktera te su prikazane samo ovisnosti između razreda koji pripadaju istom dijelu dijagrama. Iz naziva i tipova atributa u razredima može se zaključiti vrsta ovisnosti među različitim razredima.



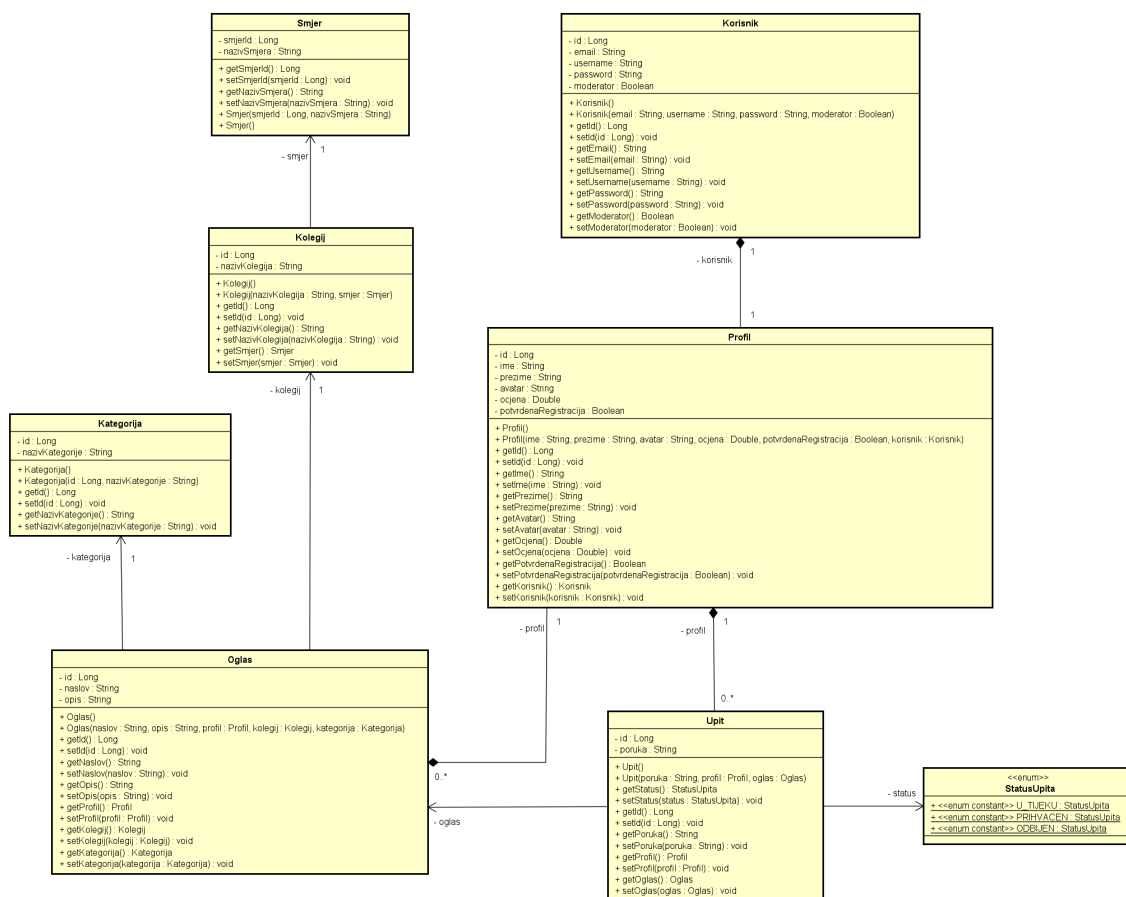
Slika 4.4: Dijagram razreda - Controllers



Slika 4.5: Dijagram razreda - Dana Access Object

Razredi modela preslikavaju strukturu baze podataka u aplikaciju. Implementirane metode komuniciraju s bazom te vraćaju tražene podatke. Razred Korisnik predstavlja registriranog korisnika i njegovo osnovne informacije važne za prijavu i slanje e-maila: e-mail adresa, korisničko ime i zaporka. Razred Profil sadrži više dodatnih informacija o registriranom korisniku kao što su ime, prezime, odabrani avatar i status je li registracija potvrđena putem maila ili nije. Razred Smjer predstavlja dostupne smjerove na studiju. Razred Kolegij sadrži informacije o kolegijima koji su dodani u bazu. Dostupne informacije su naziv kolegija i smjer kojem taj kolegij pripada. Razred Kategorija predstavlja kategoriju u koju možemo svrstati oglas. U sustavu postoji 5 različitih kategorija: laboratorijska vježba, blic, gradivo, kontinuirani ispit i ispitni rok. Razred Oglas predstavlja informacije koje

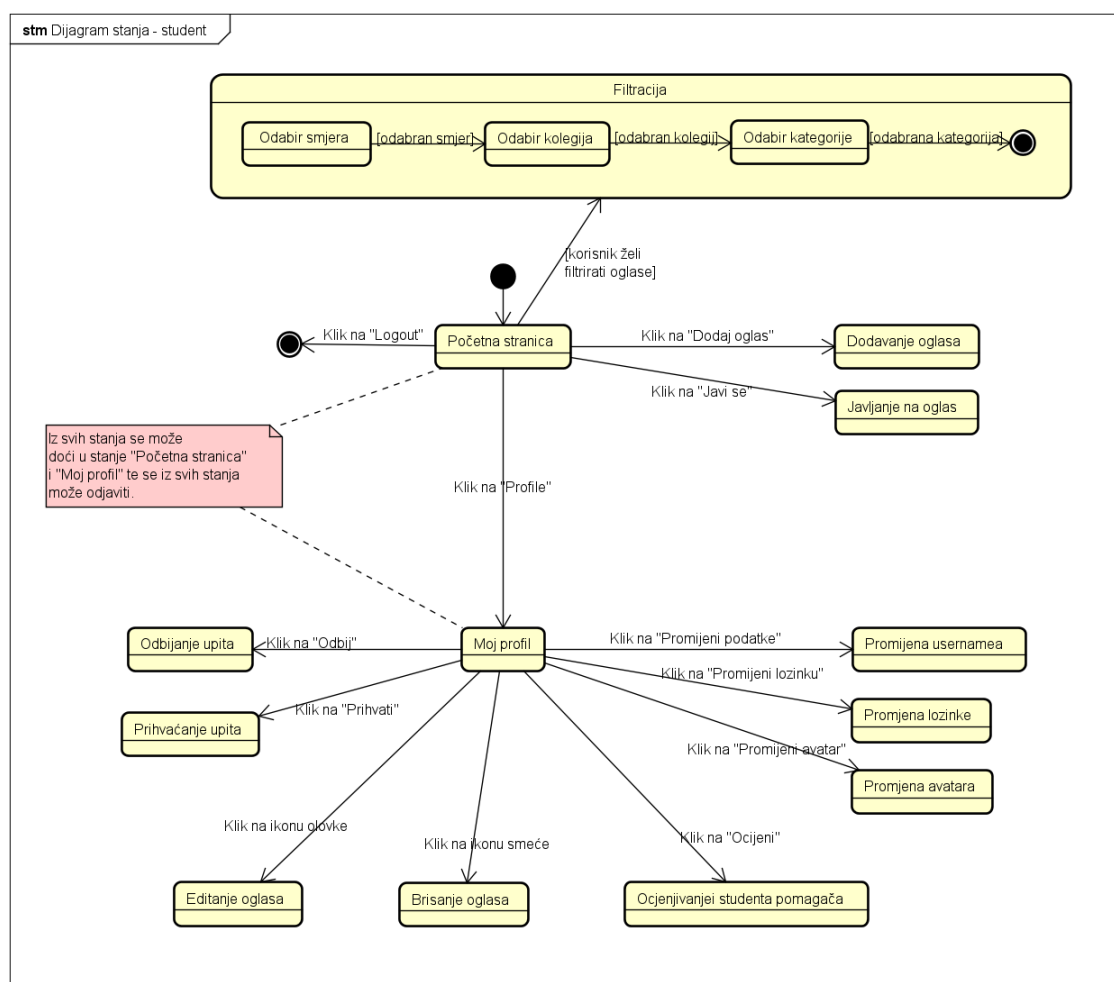
se pohranjuju u bazu za pojedini oglas, a to su: naziv oglasa, opis oglasa, kolegij za koji se oglas objavljuje, kategorija kojoj oglas pripada i profil koji je poslao upit za taj oglas. Razred Upit pamti podatke vezane uz javljanje na određeni oglas. Sadrži oglas za koji se šalje upit, sadrži profil koji stvara upit, poruku s kojom korisnik šalje upit i status u kojem se nalazi upit. Razred StatusUpita predstavlja enumeraciju pomoću koje se modelira jedan od 3 stanja upita: u tijeku, prihvaćen i odbijen.



Slika 4.6: Dijagram razreda - Models

4.3 Dijagram stanja

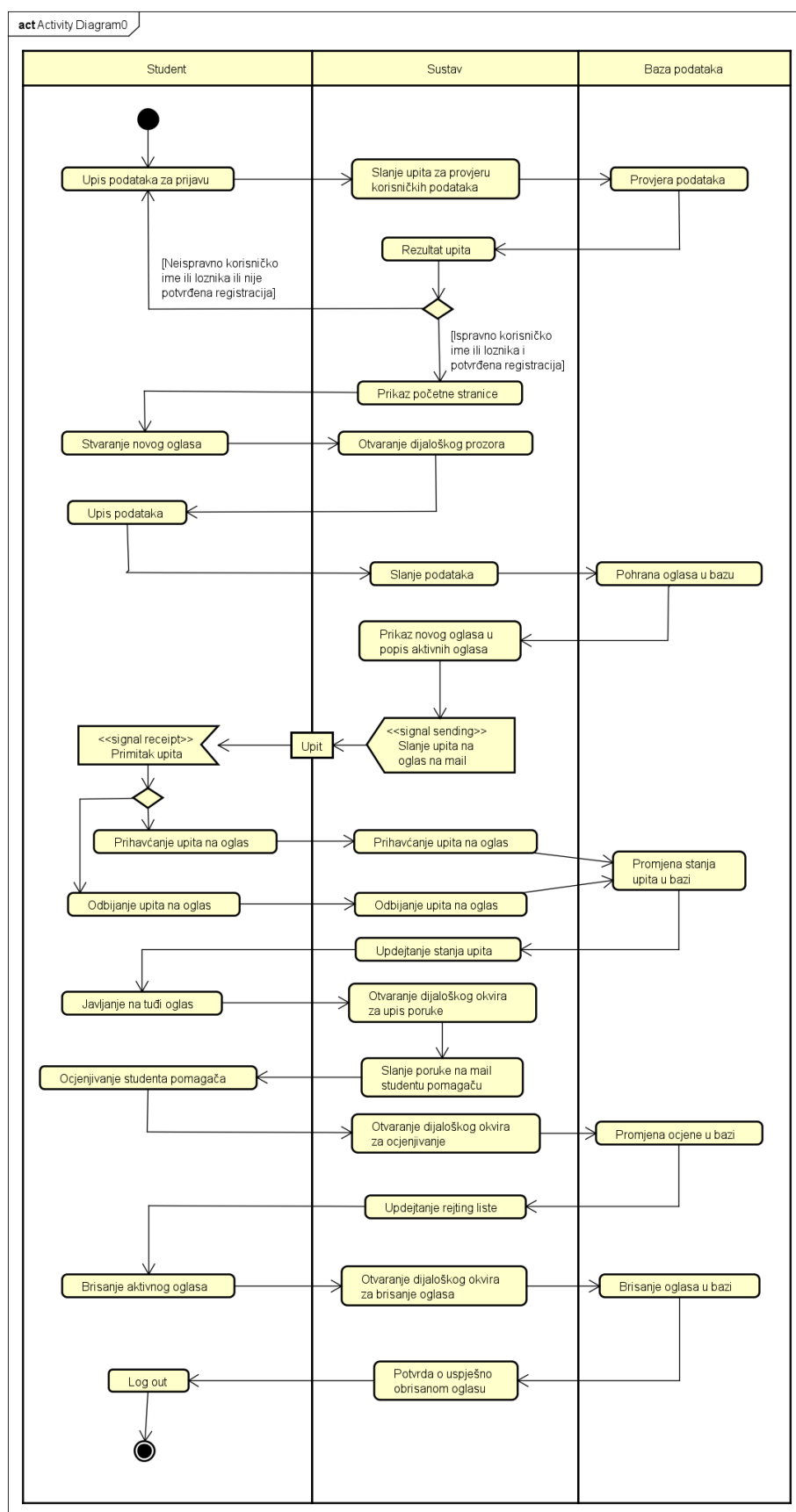
Na slici 4.7 prikazan je dijagram stanja. On prikazuje dinamičko ponašanje dijela sustava. Dijagram opisuje stanja za registriranog korisnika. Korisnik nakon prijave dolazi na početnu stranicu. Tamo ima mogućnost stvaranja novog oglasa, javljanja na tuđe oglase, filtriranja po smjeru, kolegiju i kategoriji te odlazak na stranicu „Profile“. Na stranici profila korisnik ima mogućnost uređivanja svojih podataka, tj. mijenjanja avatara, lozinke i nadimka. Ako postoji upit na oglas koje je napisao korisnik, onda korisnik ima mogućnost prihvatiti ili odbiti taj upit. U dijelu „Moji objavljeni oglasi“ korisnik ima mogućnost uređivanja oglasa, klikom na ikonu olovke, ili brisanja oglasa, klikom na ikonu smeća. Korisnik iz svih stanja može doći u stanje početne stranice i stranice profila te se iz svih stanja može odjaviti.



Slika 4.7: Dijagram stanja

4.4 Dijagram aktivnosti

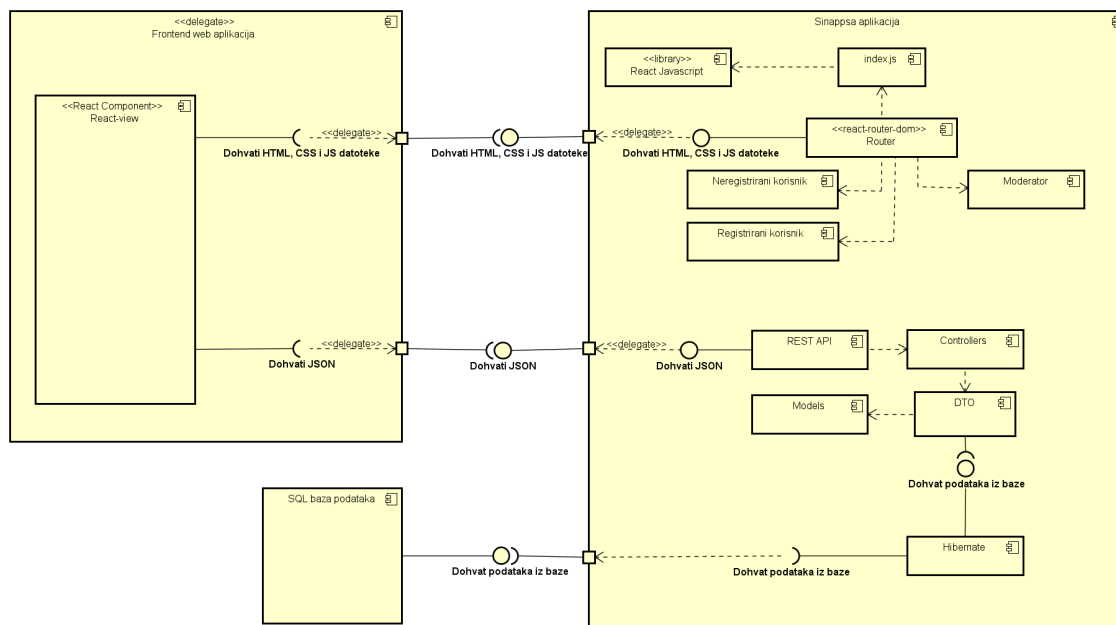
Dijagram aktivnosti koristi se za modeliranje poslovnih procesa ili upravljačkog i podatkovnog toka. Na slici 4.8 prikazan je dijagram aktivnosti registriranog korisnika. Korisnik nakon prijave stvara novi oglas nakon čega preko e-maila dobiva upit na stvoreni oglas. Korisnik odabire hoće li prihvatiti ili obiti upit na oglas. Zatim se korisnik javlja na oglas nekog drugog studenta pomagača te nakon potvrđenog upita na oglas ocjenjuje instrukcije. Na posljepku korisnik briše svoj oglas te se odjavljuje.



Slika 4.8: Dijagram aktivnosti

4.5 Dijagram komponenti

Dijagram komponenti prikazan na slici 4.9 opisuje međusobnu ovisnost komponenti, internu strukturu komponenti i njihov odnos prema okolini. Sustavu se pristupa preko dva sučelja. Prvo sučelje služi za dohvat HTML, CSS i JS datoteka. To sučelje pripada frontend dijelu aplikacije. Router je komponenta koja na temelju url određuje koje datoteke će sučelje vratiti kao rezultat upita. Frontend dio aplikacije sastoji se od niza JavaScript datoteka koje su raspoređene u logičke cjeline. Te logičke cjeline nazvane su po različitim tipovima aktera koji pristupaju aplikaciji. Sve JavaScript datoteke u logičkim cjelinama ovise o React biblioteci pomoću koje se dohvaćaju gotove komponente kao što su polja za upis podataka, gumbi, odlomci, itd. Drugo sučelje pomoću kojeg se može pristupiti aplikaciji je sučelje koje služi za dohvat JSON podataka. Preko tog sučelja pristupa se REST API komponenti sustava. REST API poslužuje podatke koji pripadaju backend dijelu aplikacije. Hibernate je implementacija JPA putem kojeg se pomoću SQL upita može pristupiti i dohvatiti podatke iz baze podataka. Tako dohvaćeni podaci iz baze šalju se dalje MVC arhitekturi u obliku DAO (Data Access Object). React-view je komponenta koja preko dostupnih sučelja komunicira sa Sinappsa aplikacijom te ovisno o korisnikovim akcijama osvježava prikaz i dohvaća nove podatke od aplikacije.



Slika 4.9: Dijagram komponenti

5. Implementacija i korisničko sučelje

5.1 Korištene tehnologije i alati

Za komunikaciju između članova tima korištene su aplikacije WhatsApp¹ i Microsoft Teams². Upravljanje programskim kodom se obavljalo preko distribuiranog sustava Git³ te je korišten udaljeni repozitorij GitLab⁴.

Za izradu korisničkih sučelja, tj. frontenda, korištena je JavaScript biblioteka React ili React.js⁵ koja se koristi za brzu i učinkovitu izgradnju interaktivnih korisničkih sučelja i web aplikacija uz znatno manje napisanog koda. U Reactu aplikacije se razvijaju stvaranjem komponenti za višestruku uporabu. React je izgrađen pomoću JSX-a – kombinacije JavaScripta i XML-a. Razvio ga je Facebook. Razvojna okolina korištena za rad u Reactu je Visual Studio Code⁶. VS Code besplatno je Microsoftovo razvojno okruženje otvorenog koda (engl. open source). Danas jedan od najpopularnijih uređivača izvornog koda koji omogućuje programiranje u velikom izboru programskih jezika.

Implementacija ove web aplikacije, tj. backend, napravljena s radnim okvirom Spring Boot⁷. Spring Boot je mikro framework otvorenog koda koji održava tvrtka Pivotal. Koristi Javu te je zgrađen na temelju Spring okvira. On pruža lakši i brži način za postavljanje, konfiguriranje i pokretanje aplikacija. Za pisanje backenda korišteno je razvojno okruženje IntelliJ IDEA⁸. To je integrirano razvojno okruženje (IDE) napisano u Javi za razvoj računalnog softvera napisanog u Javi, Kotlinu, Groovyju i drugim jezicima koji se temelje na JVM-u. Razvio ga je JetBrains. Pruža određene značajke kao što je dovršavanje koda analizom konteksta, navigacija koda koja omogućuje izravno skakanje u klasu ili deklaraciju u kodu, refaktoriranje koda, otklanjanje pogrešaka koda i opcije za ispravljanje nedosljed-

¹<https://www.whatsapp.com/>

²<https://www.microsoft.com/en-us/microsoft-teams/group-chat-software>

³<https://git-scm.com/>

⁴<https://about.gitlab.com/>

⁵<https://reactjs.org/>

⁶<https://code.visualstudio.com/>

⁷<https://spring.io/projects/spring-boot>

⁸<https://www.jetbrains.com/idea/>

nosti putem prijedloga.

Za pisanje dokumentacije korišten je markup jezik LaTeX pisan u TeXstudio⁹ uređivaču teksta. Za izradu dijagrama potrebnih u dokumentaciji korišteni su alati Astah Professional¹⁰ i Visual Paradigm Online¹¹. Ispitivanje programskog rješenja provedeno je uz radni okvir Selenium WebDriver¹².

Za deploy frontenda korišten je Netlify¹³, a za deploy backenda i baze podataka Render¹⁴.

⁹<https://www.texstudio.org/>

¹⁰<https://astah.net/products/astah-professional/>

¹¹<https://online.visual-paradigm.com/>

¹²<https://www.selenium.dev/documentation/webdriver/>

¹³<https://www.netlify.com/>

¹⁴<https://render.com/>

5.2 Ispitivanje programskog rješenja

5.2.1 Ispitivanje komponenti

Ispitivanje komponenti provodi se s ciljem ispitivanja pojedinih metoda i klasa, na način da se metodi daju ulazni podaci (ispravni ili neispravni) i provjerava se daje li metoda očekivani izlaz.

Prilikom testiranja repository sloja koristili smo KorisnikRepository interface. Najprije smo provjerili koliko korisnika već ima u bazi kako bismo znali je li se naš novi korisnik dodao u bazu. Napravili smo novog ispravnog korisnika i pokušali ga spremiti u bazu.

```
@Test
@Order(1)
void setData() {
    brojKorisnika = korisnikRepositoryTest.findAll().size();
    korisnik = new Korisnik("email@proba.com", "Test ime",
        "testPassword1234", false);
    Assertions.assertNull(korisnik.getId());
    korisnik = korisnikRepositoryTest.save(korisnik);
}
```

Zatim smo provjerili postoji li jedan korisnik više u bazi i spremili njegov id kako bismo ga na kraju mogli obrisati.

```
@Test
@Order(2)
void isSave() {
    Assertions.assertEquals(brojKorisnika + 1,
        korisnikRepositoryTest.findAll().size());
    Assertions.assertNotNull(korisnik.getId());
    idKorisnika = korisnik.getId();
}
```

Potom smo pokušali spremiti korisnika s prekratkom lozinkom što nije uspjelo, kao što je i očekivano.

```
@Test
@Order(3)
void prekratkaLozinka() {
    korisnik = new Korisnik("prekratkaLozinka@proba.com",
        "Prekratka lozinka", "test12345", false);
    Assertions.assertEquals(brojKorisnika + 1,
        korisnikRepositoryTest.findAll().size());
    try {
        korisnikRepositoryTest.save(korisnik);
    } catch (Exception e) {
        System.out.println("Lozinka je prekratka!\n" + e);
    }
    Assertions.assertEquals(brojKorisnika + 1,
        korisnikRepositoryTest.findAll().size());
}
```

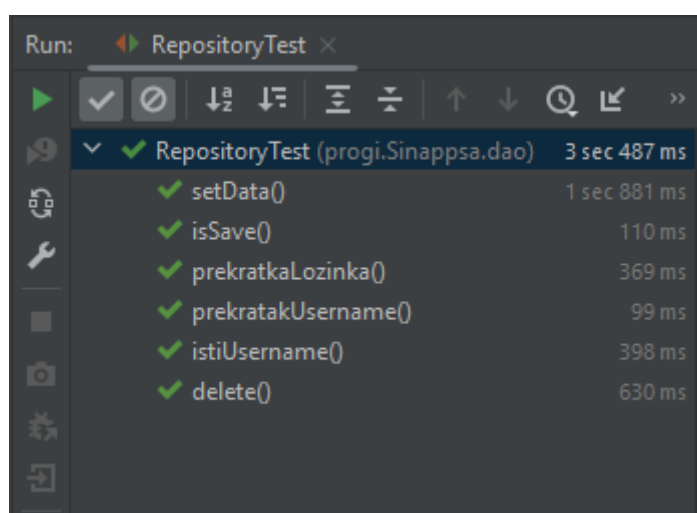
Testirali smo i slučaj kada je username prekratak. Spremanje takvog korisnika također nije uspjelo.

```
@Test
@Order(4)
void prekratakUsername() {
    try {
        korisnik = new Korisnik("prekratakUsername@proba.com",
            "", "testLozinka12345", false);
        Assertions.assertEquals(brojKorisnika + 1,
            korisnikRepositoryTest.findAll().size());
        korisnikRepositoryTest.save(korisnik);
    } catch (IllegalArgumentException e) {
        System.out.println("Username je prekratak!\n" + e);
    }
    Assertions.assertEquals(brojKorisnika + 1,
        korisnikRepositoryTest.findAll().size());
}
```

Na kraju smo još jednom provjerili broj korisnika u bazi, te obrisali jedinog korisnika kojeg smo uspjeli uspješno spremiti.

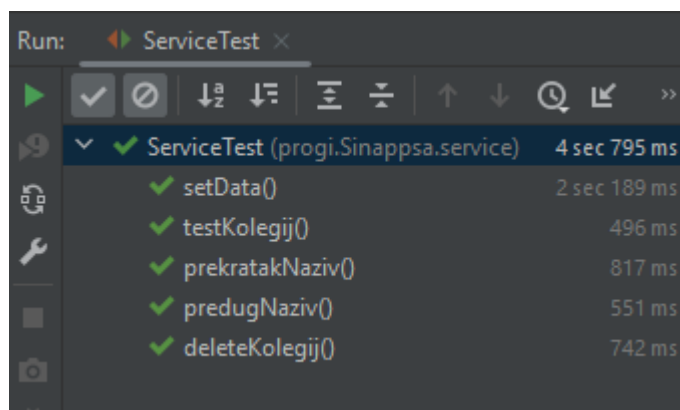

```
@Test
@Order(6)
void delete() {
    Assertions.assertEquals(brojKorisnika + 1,
        korisnikRepositoryTest.findAll().size());
    korisnikRepositoryTest.deleteById(idKorisnika);
    Assertions.assertEquals(brojKorisnika,
        korisnikRepositoryTest.findAll().size());
    List<Korisnik> korisnici =
        korisnikRepositoryTest.findAll().stream().filter(k ->
            k.getId().equals(idKorisnika)).toList();
    Assertions.assertEquals(0, korisnici.size());
}
```

Kada pokrenemo sve testove vidimo da se oni ispravno izvode.



Slika 5.1: Rezultati RepositoryTest testova

Veoma slične testove napisali smo i za provjeru service sloja, gdje smo provjeravali KolegijService i SmjerService interface i koji također svi ispravno rade.



Slika 5.2: Rezultati ServiceTest testova

Za provjeru controller sloja smo koristili OglasController. Na početku svakog testa zadali smo odgovarajući URL stranice, odredili vrstu HTTP zahtjeva, te postavili potrebne ulazne podatke. Za dodavanje novog oglasa najprije smo u string varijablu u JSON formatu spremili naslov, opis, idPomagaca, idKolegija i idKategorije te to postavili kao entitet zahtjeva. Nakon što se zahtjev pošalje i dobijemo odgovor provjeravamo je li se on izvršio dobro. Najprije provjeravamo je li statusLine 200, te vraća li se u odgovoru "OK" kako je i zadano u controller-u.

```
@Test
@Order(1)
public void addNewOglas() throws NoSuchAlgorithmException,
    KeyStoreException, KeyManagementException, IOException {
    String pageUrl = "http://localhost:8080/api/oglas/create";

    TrustStrategy acceptingTrustStrategy = (X509Certificate[]
        chain, String authType) -> true;
    SSLContext sslContext = org.apache.http.ssl.SSLContexts.
        custom().loadTrustMaterial(null, acceptingTrustStrategy)
        .build();
    SSLConnectionSocketFactory csf = new
        SSLConnectionSocketFactory(sslContext);
    CloseableHttpClient httpClient = HttpClients.custom()
        .setSSLSocketFactory(csf).build();

    HttpPost httpPost = new HttpPost(pageUrl);
    String JSON_STRING = "{\"naslov\":\"Oglas
```

```
        iz Controller testa\"," + "\"opis\":" + "\"Novi  
        opis iz Controller testa\"," + "\"idPomagaca\  
        \":164,\"idKolegija\":" + "140,\"idKategorije\":" + "3}";  
        HttpEntity stringEntity = new StringEntity(JSON_STRING,  
        ContentType.APPLICATION_JSON);  
        httpPost.setEntity(stringEntity);  
  
        CloseableHttpResponse response = httpClient.execute(httpPost);  
  
        logger.info("code={}",  
        response.getStatusLine().getStatusCode());  
        logger.info("statusLine={}",  
        response.getStatusLine());  
        Assertions.assertEquals(200,  
        response.getStatusLine().getStatusCode());  
        String jsonString =  
        EntityUtils.toString(response.getEntity());  
        logger.info("jsonString={}", jsonString);  
        Assertions.assertEquals("OK", jsonString);  
    }  
}
```

U ControllerTestu smo još pokušali dohvatiti sve oglase za prijavljenog korisnika kako bismo provjerili provjerava li se autorizacija na dobar način tamo gdje je ona potrebna. U header smo postavili username i password korisnika iz baze te nakon što smo dobili odgovor kojem je statusLine 200 provjeravamo počinje li odgovor sa id-om korisnika kojeg smo mu zadali.

```
@Test  
@Order(2)  
public void getOglasKorisnika() throws  
    NoSuchAlgorithmException, KeyStoreException,  
    KeyManagementException, IOException, JSONException {  
    //dohvati sve oglase za prijavljenog korisnika  
    String pageUrl = "http://localhost:8080/api/oglas  
        /dohvati/korisnik";  
  
    TrustStrategy acceptingTrustStrategy =
```

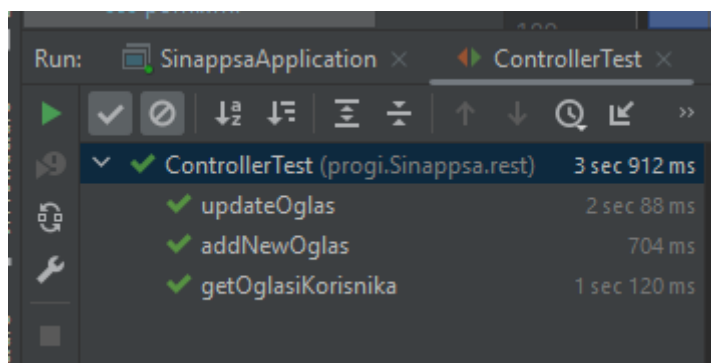
```
(X509Certificate[] chain, String authType) -> true;
SSLContext sslContext = org.apache.http.ssl
    .SSLContexts.custom().loadTrustMaterial(null,
        acceptingTrustStrategy).build();
SSLConnectionSocketFactory csf = new
    SSLConnectionSocketFactory(sslContext);
CloseableHttpClient httpClient = HttpClients
    .custom().setSSLSocketFactory(csf).build();

HttpGet httpGet = new HttpGet(pageUrl);
String auth = username + ":" + password;
byte[] encodedAuth =
    Base64.getEncoder().encode(auth.getBytes
        (StandardCharsets.ISO_8859_1));
String authHeader = "Basic " + new String(encodedAuth);
httpGet.setHeader(HttpHeaders.AUTHORIZATION, authHeader);

CloseableHttpResponse response = httpClient
    .execute(httpGet);

logger.info("code={}", response.getStatusLine()
    .getStatusCode());
logger.info("statusLine={}", response.getStatusLine());
Assertions.assertEquals(200,
    response.getStatusLine().getStatusCode());
String jsonString = EntityUtils.toString(response.getEntity());
logger.info("jsonString={}", jsonString);
Assertions.assertTrue(jsonString
    .contains("\"id\":164,\"ime\":\"Milica\""));
}
```

Controller testovi također svi rade ispravno.



Slika 5.3: Rezultati ControllerTest testova

5.2.2 Ispitivanje sustava

Ispitivanje cijelog sustava napravili smo pomoću Selenium WebDrivera unutar Junit testova.

Na početku imamo dva testa koja testiraju ispravnu i neispravnu prijavu. U svakom od njih naprijed definiramo postavke samo drivera, te mu zadamo URL stranice sa koje započinje test. Nakon što pronade element u koji se upisuje username, pošaljemo mu podatke za username, te isto učinimo i za password. Budući da su podaci ispravni, aplikacija automatski preusmjerava korisnika na početnu stranicu pa se to i provjerava u posljednjem retku testa.

```
@Test
```

```
public void testLogin() throws InterruptedException {  
    System.setProperty("webdriver.chrome.driver",  
        "C:\\\\Program Files (x86)\\\\Chrome Driver\\\\chromedriver.exe");  
    WebDriver driver = new ChromeDriver();  
    driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);  
    driver.get("https://sinappsa.netlify.app/prijava");  
    WebElement element = driver.findElement(By.name("usernameOrEmail"));  
    element.sendKeys("vmilica");  
    element = driver.findElement(By.name("password"));  
    element.sendKeys("vmilica1234");  
    driver.findElement(By.cssSelector("Button[type='submit']")).click();  
    Thread.sleep(3000);  
    String url = driver.getCurrentUrl();  
    driver.quit();  
    Assertions.assertEquals("https://sinappsa.netlify.app/", url);  
}
```

```
}
```

Test za neispravnu prijavu je skoro identičan kao za ispravnu, jedino što mu se zadaju neispravni podaci. U ovom slučaju to je neispravna lozinka i on mora na samom kraju ostati na stranici za prijavu, a ne biti preusmjeren na početnu stranicu aplikacije.

```
@Test
public void failLogin() throws InterruptedException {
    System.setProperty("webdriver.chrome.driver",
        "C:\\Program Files (x86)\\Chrome Driver\\chromedriver.exe");
    WebDriver driver = new ChromeDriver();
    driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
    driver.get("https://sinappsa.netlify.app/prijava");
    WebElement element = driver.findElement(By.name("usernameOrEmail"));
    element.sendKeys("vmilica");
    element = driver.findElement(By.name("password"));
    element.sendKeys("vmilica123456");
    driver.findElement(By.cssSelector("Button[type='submit']")).click();
    Thread.sleep(3000);
    String url = driver.getCurrentUrl();
    driver.quit();
    Assertions.assertEquals("https://sinappsa.netlify.app/prijava", url);
}
```

Testirali smo i ponaša li se aplikacija kako treba ako se ne upišu svi potrebni podaci prilikom dodavanja novog oglasa. Naravno, za dodavanje novog oglasa korisnik mora biti prijavljen, stoga na početku testa, nakon postavki drivera slijedi prijava u aplikaciju identična prvom testu. Nakon što je prijava bila uspješna na početnoj stranici odabire se gumb “Dodaj oglas” i u tražena polja upisuju se samo naslov i opis oglasa dok smjer, kolegij i kategorija ostaju prazni što aplikacija ne bi trebala podržati. Budući da je test uhvatio iznimku, vidimo da takav oglas nije dodan u bazu podataka i zaključujemo da aplikacija radi ispravno.

```
@Test
void dodavanjeOglasa() throws InterruptedException {
    //neuspjelo
```

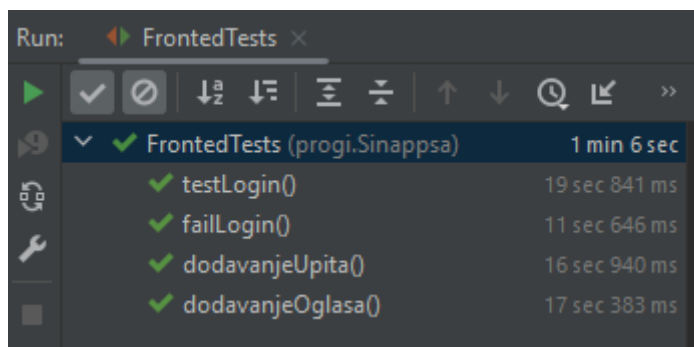
```
System.setProperty("webdriver.chrome.driver",
"C:\\Program Files (x86)\\Chrome Driver\\chromedriver.exe");
WebDriver driver = new ChromeDriver();
driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
driver.get("https://sinappsa.netlify.app/prijava");
WebElement element = driver.findElement(By.name("usernameOrEmail"));
element.sendKeys("vmilica");
element = driver.findElement(By.name("password"));
element.sendKeys("vmilica1234");
driver.findElement(By.cssSelector("Button[type='submit']")).click();
Thread.sleep(3000);
String url = driver.getCurrentUrl();
Assertions.assertEquals("https://sinappsa.netlify.app/", url);
driver.findElement(By.xpath("//Button[text()='Dodaj oglas']")).click();
element = driver.findElement(By.name("naslov"));
element.sendKeys("Selenium naslov oglasa");
element = driver.findElement(By.name("opis"));
element.sendKeys("Selenium opis novog oglasa");
try {
    driver.findElement(By.xpath("//Button[text()='Potvrdi']")).click();
} catch (Exception e) {}
System.out.println("Uhvatio sam iznimku!!");
driver.findElement(By.xpath("//button[text()='Ok']")).click();
Assertions.assertEquals("https://sinappsa.netlify.app/", url);
driver.quit();
}
```

Posljednji test sustava provjerava javljanje na upite već postavljenih oglasa. Također se ni na upite ne može javljati korisnik koji nije prijavljen, stoga najprije moramo provesti valjanu prijavu. Nakon što je prijava uspjela, a to vidimo preusmjerenjem na početnu stranicu, odabiremo oglas koji nije naš i na koji se ranije nismo javili, upisujemo poruku korisniku koji je oglas objavio i šaljemo upit.

```
@Test
void dodavanjeUpita() throws InterruptedException {
    //nije uspjelo
    System.setProperty("webdriver.chrome.driver",
```

```
"C:\\Program Files (x86)\\Chrome Driver\\chromedriver.exe");
WebDriver driver = new ChromeDriver();
driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
    driver.get("https://sinappsa.netlify.app/prijava");
WebElement element = driver.findElement(By.name("usernameOrEmail"));
element.sendKeys("vmilica");
element = driver.findElement(By.name("password"));
element.sendKeys("vmilica1234");
driver.findElement(By.cssSelector("Button[type='submit']")).click();
Thread.sleep(3000);
String url = driver.getCurrentUrl();
Assertions.assertEquals("https://sinappsa.netlify.app/", url);
Thread.sleep(3000);
driver.findElement(By.xpath("//button[text()='Javi se']")).click();
element = driver.findElement(By.xpath("//textarea"));
element.sendKeys("Javljam se na ovaj upit iz Seleniuma!");
        driver.findElement(By.xpath("//button[text()='Submit']")).click();
driver.quit();
}
```

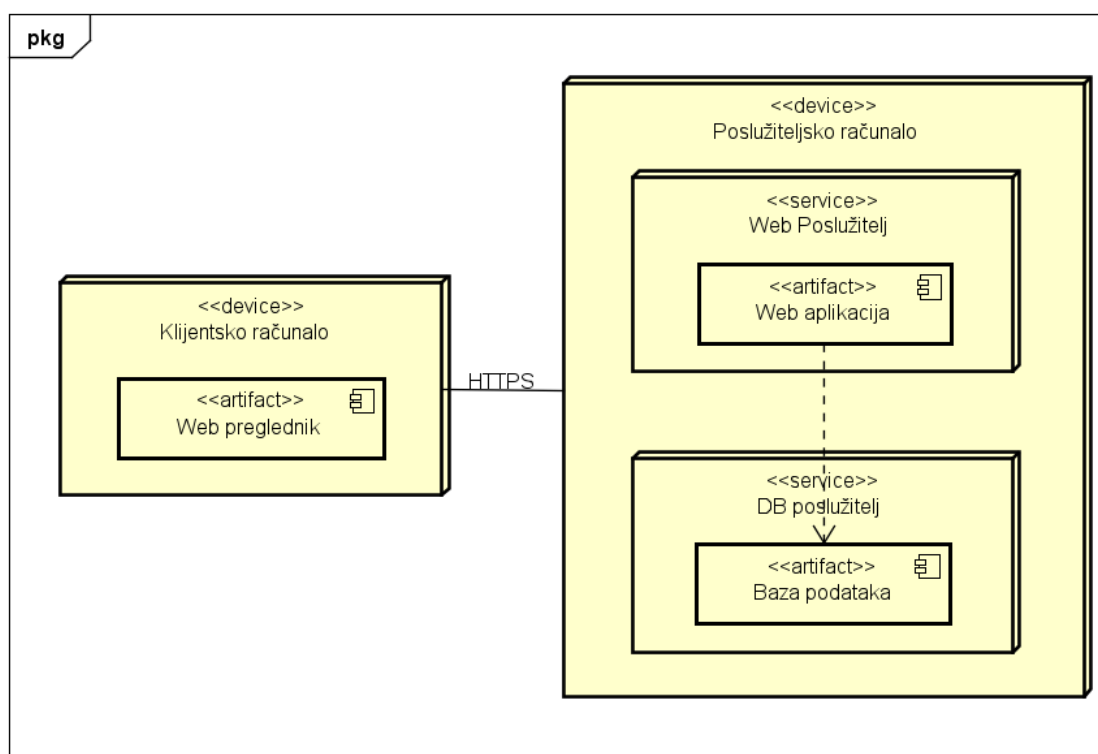
Nakon pokretanja svih testova sustava vidimo da i oni ispravno rade.



Slika 5.4: Rezultati testova sustava

5.3 Dijagram razmještaja

Dijagram razmještaja je strukturni statički UML dijagram koji opisuje topologiju sustava i usredotočen je na odnos sklopovskih i programskih dijelova. Na klijentskoj strani je PC računalo na kojem je pokrenut web preglednik. Klijent se protokolom HTTPS spaja na poslužitelja. Web poslužitelj i poslužitelj baze podataka nalaze se na istom računalu. Sustav je baziran na arhitekturi „klijent – poslužitelj“.



Slika 5.5: Dijagram razmještaja

5.4 Upute za puštanje u pogon

Instalacija poslužitelja baze podataka

Na poslužitelju Render potrebno je napraviti poslužitelj baze podataka. Odabirom *New* -> *PostgreSQL* radi se nova SQL baza. Potrebno je unijeti sljedeće podatke:

- Ime: ime pod kojim će se baza spremati na Renderu
- Baza: postavljanje atributa dbname koji ćemo koristiti kod povezivanja s backend dijelom aplikacije
- PostgreSQL: odabir verzije PostgreSQL

Opcionalno može se dodati ime korisnika (u suprotnom će se naziv korisnika slučajno generirati).

New PostgreSQL

Name ⓘ	<input type="text"/>
Database ⓘ	<input type="text" value="randomly generated unless specified"/>
User	<input type="text" value="randomly generated unless specified"/>
Region <small>The region where your Database runs. Services must be in the same region to communicate privately and you currently have services running in Frankfurt.</small>	<input type="text" value="Frankfurt (EU Central)"/>
PostgreSQL Version	<input type="text" value="15"/>
Datadog API Key ⓘ	<input type="text"/>

Slika 5.6: Prikaz dijaloškog okvira stvaranja baze podataka

General

Name	Progi_sinappsa	Edit
Created	2 months ago	
Expiration	February 13, 2023 ⓘ	
Status	Available	
PostgreSQL Version	15	
Region	Frankfurt (EU Central)	
Read Replica	Add Read Replica ⓘ	
Storage	6.48% used out of 1.0 GiB	

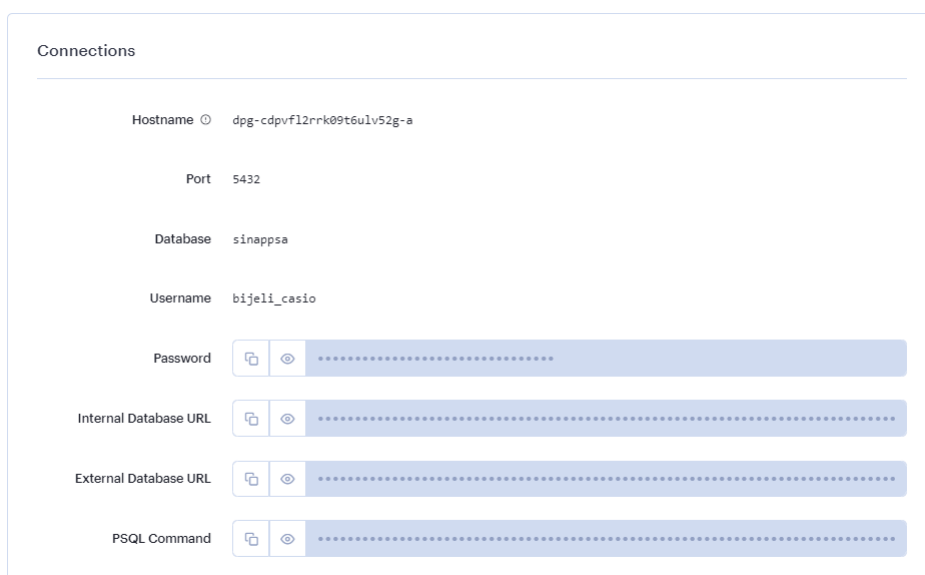
Slika 5.7: Uspješno dodan poslužitelj baze podataka

Povezivanje backend dijela aplikacije s bazom podataka

U backend dijelu aplikacije, kako bismo se uspješno povezali s bazom, u Spring Boot aplikaciju u datoteku `Sinappsa/src/resources/deploy/application.properties` treba dodati 3 atributa:

- `spring.datasource.url`
- `spring.datasource.username`
- `spring.datasource.password`

Ove attribute treba postaviti u skladu s podacima za povezivanje na poslužitelj baze koji su dostupni na Renderu.



The screenshot shows a 'Connections' form with the following fields and values:

Field	Value
Hostname	dpg-cdpvf12rrk09t6ulv52g-a
Port	5432
Database	sinappsa
Username	bijeli_casio
Password	[Redacted]
Internal Database URL	[Redacted]
External Database URL	[Redacted]
PSQL Command	[Redacted]

Slika 5.8: Prikaz podataka potrebnih za povezivanje na poslužitelj baze podataka

Atribut `spring.datasource.url` treba postaviti na string koji će predstavljati URL adresu poslužitelja baze podataka. Taj URL započinje sa stringom `jdbc:postgresql://` te se na taj dio zalijepi *Hostname* poslužitelja zajedno s portom (deafultni port je 5432) te se nakon toga doda ime baze podataka. Atribut `spring.datasource.username` postavlja se na *Username* koji je dodijelio poslužitelj baze podataka. Atribut `spring.datasource.password` poprima vrijednosti generirane zaporce koja služi za autentifikaciju pri spajanju na poslužitelj baze podataka.

Instalacija poslužitelja backend dijela aplikacije

Za instalaciju backend poslužitelja koristimo Render. Kako Render ne pruža podršku za Javu, prije same instalacije poslužitelja backenda potrebno je Java kod omotati u Docker izvršnu okolinu pomoću Dockerfile-a na način prikazan na slici 5.9.

```
1 # Container za izgradnju (build) aplikacije
2 FROM openjdk:17-alpine AS builder
3
4 # Kopiranje izvornog koda u container
5 COPY ./IzvorniKod/Backend/Sinappsa/.mvn .mvn
6 COPY ./IzvorniKod/Backend/Sinappsa/mvnw .
7 COPY ./IzvorniKod/Backend/Sinappsa/pom.xml .
8 COPY ./IzvorniKod/Backend/Sinappsa/src src
9 RUN chmod +x mvnw
10
11 # Pokretanje builda
12 RUN ./mvnw clean package
13
14 # Stvaranje containera u kojem ce se vrtiti aplikacija
15 FROM openjdk:17-alpine
16
17 ## Ovdje je moguće instalirati alate potrebne za rad aplikacije. Vjerojatno vam neće trebati, no dobro je znati.
18 ## Linux distro koji se koristi je Alpine, stoga se kao package manager koristi apk
19 #RUN apk install <nesto>
20
21 # Kopiranje izvršnog JAR-a iz build containera u izvršni container
22 COPY --from=builder target/*.jar /app.jar
23
24 # Izlaganje porta
25 EXPOSE 8080
26
27 # Naredba kojom se pokrene aplikacija
28 ENTRYPOINT ["java","-jar","/app.jar"]
29
```

Slika 5.9: Prikaz Dockerfilea

Kako bismo uspješno postavili poslužitelj potrebno je odabrati opciju *New-Web Service*. Zatim se potrebno povezati sa Git repozitorijem na kojem se nalazi kod backend aplikacije. Nakon toga Render zahtjeva sljedeće parametre:

- *Name*: ime web servisa
- *Branch*: grana git repozitorija na kojoj se nalazi web servis
- Korijenski direktorij: put do direktorija u kojem se nalazi napravljeni Dockerfile
- *Environment*: odabir izvršne okoline (budući da Render nema direktnu podršku za Javu ovdje odabiremo Docker)

You are deploying a web service for [bijelicasio/projektrepozitorij](#).

Name A unique name for your web service.	<input type="text" value="example-service-name"/> <small>Required</small>
Region The region where your web service runs. Services must be in the same region to communicate privately and you currently have services running in Frankfurt.	<input type="text" value="Frankfurt (EU Central)"/>
Branch The repository branch used for your web service.	<input type="text" value="main"/>
Root Directory <small>Optional</small> Defaults to repository root. When you specify a root directory that is different from your repository root, Render runs all your commands in the specified directory and ignores changes outside the directory.	<input type="text" value="e.g., src"/>
Environment The runtime environment for your web service.	<input type="text" value="Docker"/>

Slika 5.10: Prikaz podataka potrebnih za instalaciju poslužitelja backend dijela aplikacije

Klikom na gumb *Advanced* proširuju se mogućnosti za postavljene web servise. U *Advanced* postavkama pod karticom *Environment variables* potrebno je dodati dvije varijable izvršne okoline:

- DB_PASS
- DB_USERNAME

Ove dvije varijable postavljamo na vrijednosti *usernamea* i *passworda* s poslužitelja baze podataka. Odabirom opcije *Create servis* započinje stvaranje servisa i stvaranje tablica u bazi podataka.

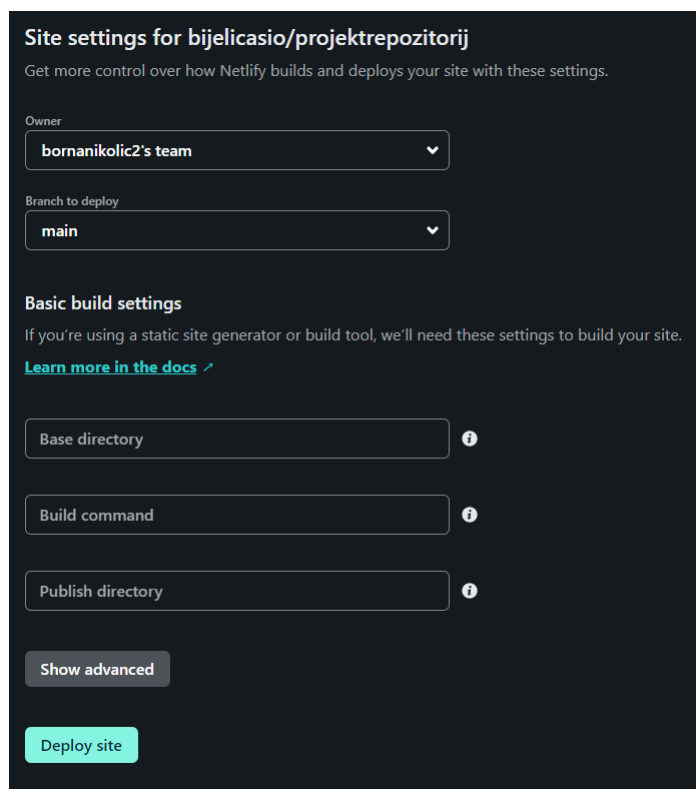
Pokretanje backend aplikacije i stvaranje tablica u bazi

Pokretanjem Spring Boot backend aplikacije pomoću Java Persistence API-ja koji služi za perzistenciju podataka aplikacije pomoću objektno relacijskog preslikavanja (eng. Object Relational Mapping - ORM) u povezanoj bazi podataka stvaraju se tablice iz klasa definiranih u aplikaciji koje su anotirane oznakom *@Entity*.

Uspješnim deployom backend aplikacije ona postaje dostupna za rad preko interneta. Pomoću kartice *Logs* na Renderu moguće je pratiti zahtjeve koji pristižu na servis te odgovore na zahtjeve i eventualne greške.

Instalacija poslužitelja frontenda

Za pogon frontend dijela aplikacije ne koristimo Render, već Netlify. Nakon uspješnog stvaranja tima na Netlifyu odabirom opcije *Add new Site* pod kategorijom *Site* možemo deployati stranicu. Nakon *Add new Site* odabiremo opciju *Import an existing project* te se povežemo sa Git repozitorijem. Nakon odabira repozitorija odabiremo granu u repozitoriju na kojem se nalazi izvorni kod dijela stranice i u polje *Base directory* stavljamo putanju do mape u kojoj se nalazi izvorni kod. Za *build command* postavljamo naredbu *npm run build*.



The screenshot shows the 'Site settings for bijelicasio/projektrepozitorij' page on Netlify. It includes a header with the title and a subtitle 'Get more control over how Netlify builds and deploys your site with these settings.' Below this, there are two dropdown menus: 'Owner' set to 'bormanikolic2's team' and 'Branch to deploy' set to 'main'. A section titled 'Basic build settings' follows, with a subtitle 'If you're using a static site generator or build tool, we'll need these settings to build your site.' and a link 'Learn more in the docs'. There are three input fields: 'Base directory', 'Build command', and 'Publish directory', each with an information icon. Below these is a 'Show advanced' button and a 'Deploy site' button.

Slika 5.11: Prikaz podataka potrebnih za instalaciju frontend poslužitelja

Odabirom opcije *Deploy site* kreće se u *build frontend* i po uspješnom završetku web aplikacija je puštena u pogon na javnom poslužitelju.

6. Zaključak i budući rad

Naš projektni zadatak bio je realizacija web aplikacije Sinappsa koja omogućuje studentima FER-a da traže ili pružaju pomoć vezanu za fakultet. Nakon skoro cijelog semestra rada na aplikaciji ponosno možemo reći da smo implementirali i zadovoljili sve zahtjeve radnog zadatka.

Prvih sedam tjedana rada na aplikaciji išli su sporijim tempom. Prije početka samog programiranja aplikacije, proučili smo projektni zadatak, napravili obrasce uporabe te se dogovorili kako će aplikacija izgledati i od kojih komponenata će se sastojati. Zatim smo se podijelili na frontend, backend i dokumentaciju. Nakon podjele krenuli smo se upoznavati s tehnologijama s kojima je većina nas prvi put stupila u kontakt. Prije prve revizije imali smo osposobljene osnovne funkcionalnosti aplikacije, a to su bile registracija, prijava i dizajn stranice.

Nakon prve revizije, u dva tjedna napravljene su gotovo sve funkcionalnosti aplikacije kako bi se mogla predati alfa inačica aplikacije. Nakon manjih prepravaka u kodu, aplikacija je bila gotova i spremna za deploy. Rad na dokumentaciji prvenstveno se sastojao od crtanja i opisivanja UML dijagrama arhitekture aplikacije. Na kraju je učinjeno ispitivanje sustava i komponenti kako bi se provjerio rad aplikacije.

Uz komunikaciju preko WhatsApp-a, skoro smo svaki tjedan imali sastanke uživo ili preko Microsoft Teamsa kako bi rasporedili zadatke za nadolazeći tjedan. Česti sastanci natjerali su nas da radimo redovito te su poticali što bolju komunikaciju među članovima tima. Rad na ovom projektu bio je iznimno koristan jer nam je dao uvid u stvarni svijet web developera, od dizajniranja stranice do izrade dokumentacije. Također je prikazao važnost harmonijskog rada u grupi kako bi sve bilo izrađeno i predano na vrijeme. Iako u određenim trenutcima težak, ovaj projekt je bio veoma bitan kao primjer simulacije rada u industriji.

Popis literature

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>
7. What is React.js?, <https://blog.hubspot.com/website/react-js>
8. What Is Spring Boot?, <https://stackify.com/what-is-spring-boot/>
9. Spring Boot - Architecture, <https://www.geeksforgeeks.org/spring-boot-architecture/>

Indeks slika i dijagrama

2.1	Početna stranica za neregistriranog korisnika	7
2.2	Forma za registraciju	7
2.3	Forma za prijavu	8
2.4	Početna stranica za registriranog korisnika	9
2.5	Forma za dodavanje oglasa	9
2.6	Izgled profila - prvi dio	10
2.7	Izgled profila - drugi dio	10
2.8	Početna stranica za moderatora	11
2.9	Lista dodanih kolegija	11
3.1	Dijagram obrasca uporabe, funkcionalnosti neregistriranog korisnika	21
3.2	Dijagram obrasca uporabe, funkcionalnosti registriranog korisnika .	22
3.3	Dijagram obrasca uporabe, funkcionalnosti moderatora	23
3.4	Sekvencijski dijagram za UC4	24
3.5	Sekvencijski dijagram za UC7	25
3.6	Sekvencijski dijagram za UC13	26
3.7	Sekvencijski dijagram za UC14	27
4.1	Slojevi Spring Boota	29
4.2	Komunikacija u Spring Bootu	30
4.3	Dijagram baze podataka	35
4.4	Dijagram razreda - Controllers	36
4.5	Dijagram razreda - Dana Access Object	37
4.6	Dijagram razreda - Models	38
4.7	Dijagram stanja	39
4.8	Dijagram aktivnosti	41
4.9	Dijagram komponenti	43
5.1	Rezultati RepositoryTest testova	48
5.2	Rezultati ServiceTest testova	49
5.3	Rezultati ControllerTest testova	52

5.4	Rezultati testova sustava	55
5.5	Dijagram razmjesta	56
5.6	Prikaz dijaloškog okvira stvaranja baze podataka	57
5.7	Uspješno dodan poslužitelj baze podataka	57
5.8	Prikaz podataka potrebnih za povezivanje na poslužitelj baze podataka	58
5.9	Prikaz Dockerfilea	59
5.10	Prikaz podataka potrebnih za instalaciju poslužitelja backend dijela aplikacije	59
5.11	Prikaz podataka potrebnih za instalaciju frontend poslužitelja	61
6.1	Prikaz aktivnosti na repozitoriju	70

Dodatak: Prikaz aktivnosti grupe

Dnevnik sastajanja

1. sastanak

- Datum: 20.listopada 2022.
- Prisustvovali: L.Majer, A.Zaja, B.Nikolić, A.Grčević, M.Majdiš, T.Margetić, M.Pavičić, M.Petričević, M.Vuković
- Teme sastanka:
 - pojašnjenje dobivenog zadatka

2. sastanak

- Datum: 31.listopada 2022.
- Prisustvovali: B.Nikolić, A.Grčević, M.Majdiš, T.Margetić, M.Pavičić, M.Petričević, M.Vuković
- Teme sastanka:
 - podjela zadataka
 - dogovor oko korištenih tehnologija
 - skiciranje izgleda stranice

3. sastanak

- Datum: 3.studenoga 2022.
- Prisustvovali: L.Majer, B.Nikolić, A.Grčević, M.Pavičić
- Teme sastanka:
 - ispravak sekvencijskih dijagrama
 - ispravak baze podataka
 - objašnjenje nejasnoća

4. sastanak

- Datum: 9.studenoga 2022.
- Prisustvovali: B.Nikolić, A.Grčević, M.Majdiš, T.Margetić, M.Pavičić, M.Petričević, M.Vuković
- Teme sastanka:
 - pregled napravljene stranice

- raspodjela daljnjih zadataka

5. sastanak

- Datum: 12.prosinca 2022.
- Prisustvovali: B.Nikolić, M.Majdiš, T.Margetić, M.Pavičić, M.Vuković
- Teme sastanka:
 - rješavanje problema spajanja s bazom
 - raspodjela daljnjih zadataka

6. sastanak

- Datum: 15.prosinca 2022.
- Prisustvovali: B.Nikolić, M.Majdiš, T.Margetić, M.Petričević, M.Vuković
- Teme sastanka:
 - dogovor oko testiranja i završnih rokova
 - raspodjela daljnjih zadataka

7. sastanak

- Datum: 2.siječnja 2023.
- Prisustvovali: B.Nikolić, A.Grčević, M.Majdiš, T.Margetić, M.Petričević, M.Vuković
- Teme sastanka:
 - prezentacije napisanih kodova
 - raspodjela daljnjih zadataka

8. sastanak

- Datum: 7.siječnja 2023.
- Prisustvovali: B.Nikolić, A.Grčević
- Teme sastanka:
 - pisanje projektne dokumentacije

Tablica aktivnosti

	Borna Nikolić	Ana Grčević	Martina Majdiš	Tin Margetić	Matija Pavičić	Mihael Petričević	Milica Vuković
Upravljanje projektom	16h					4h	
Opis projektnog zadatka		3.5h					
Funkcionalni zahtjevi	10h	2h		3h	2h	3h	
Opis pojedinih obrazaca		1h					
Dijagram obrazaca	4h	2.5h	1.5h	1.5h	1.5h	1.5h	1h
Sekvencijski dijagrami	8h	3h	2h	2h	2h	2h	
Opis ostalih zahtjeva		0.2h					0.5h
Arhitektura i dizajn sustava		4.5h					
Baza podataka	3h						
Dijagram razreda	3.5h	1h					
Dijagram stanja		1.5h					
Dijagram aktivnosti		2h					
Dijagram komponenti	1.5h						
Korištene tehnologije i alati		2h					
Ispitivanje programskog rješenja		1h					1h
Dijagram razmještaja		1h					
Upute za puštanje u pogon	1.5h	1h					
Dnevnik sastajanja		0.5h					
Zaključak i budući rad		1h					
Popis literature		0.5h					

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

	Borna Nikolić	Ana Grčević	Martina Majdiš	Tin Margetić	Matija Pavičić	Mihael Petričević	Milica Vuković
<i>izrada početne stranice</i>				6h		3h	
<i>izrada registracijske stranice</i>				5h		3h	
<i>registracija (avatar ikone)</i>						3h	
<i>izrada login stranice</i>				4h		3h	
<i>izrada baze podataka</i>	12h						
<i>spajanje s bazom podataka</i>						23h	
<i>back end</i>			30h		25h	19h	10h
<i>deploy baze podataka</i>	4.5h						
<i>deploy front enda i back enda</i>	17.5h					4h	
<i>ispitivanje programskog rješenja</i>							20h
<i>izrada profil stranice</i>						5h	
<i>izrada forme za dodavanje oglasa</i>				3h			
<i>izrada admin stranice</i>				5h			

Dijagrami pregleda promjena

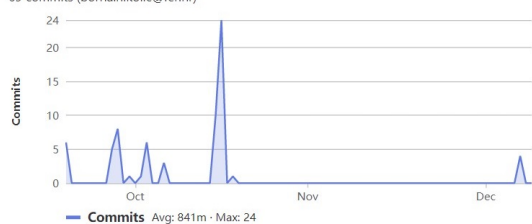
Commits to main

Excluding merge commits. Limited to 6,000 commits.



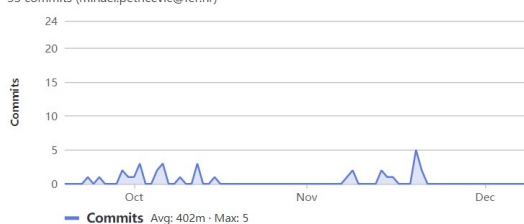
bornanikolic2

69 commits (borna.nikolic@fer.hr)



mihael53cevic

33 commits (mihael.petricevic@fer.hr)



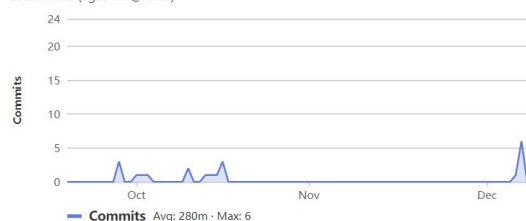
Martina Majdiš

17 commits (martina.majdis@fer.hr)



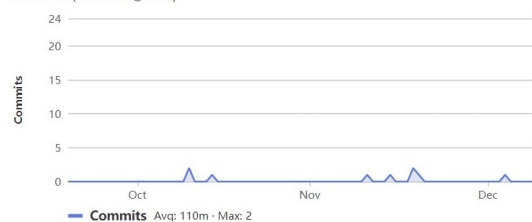
Ana Grcevic

23 commits (ag53417@fer.hr)



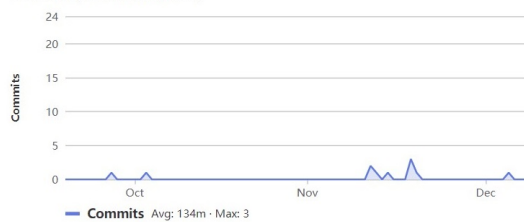
marti164

9 commits (mm53181@fer.hr)



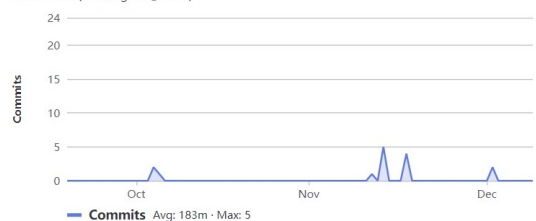
Milica

11 commits (milica.vukovic@fer.hr)



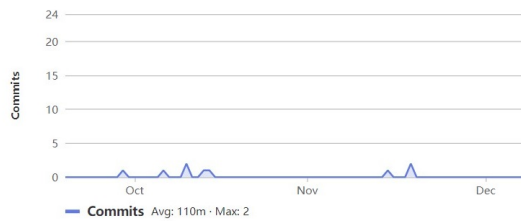
Tin

15 commits (tin.margetic@fer.hr)



Matija

9 commits (mata.pavicic@gmail.com)



Slika 6.1: Prikaz aktivnosti na repozitoriju