

Programsko inženjerstvo

Ak. god. 2022./2023.

Sinappsa

Dokumentacija, Rev. 2

Grupa: *TurboBageri*

Voditelj: *Fran Žužič*

Datum predaje: *13.01.2022.*

Nastavnik: *Laura Majer*

Sadržaj

1 Dnevnik promjena dokumentacije	2
2 Opis projektnog zadatka	4
3 Specifikacija programske potpore	8
3.1 Funkcionalni zahtjevi	8
3.1.1 Obrasci uporabe	10
3.1.2 Sekvencijski dijagrami	19
3.2 Ostali zahtjevi	24
4 Arhitektura i dizajn sustava	25
4.1 Baza podataka	27
4.1.1 Opis tablica	27
4.1.2 Dijagram baze podataka	31
4.2 Dijagram razreda	32
4.3 Dijagram stanja	38
4.4 Dijagram aktivnosti	39
4.5 Dijagram komponenti	41
5 Implementacija i korisničko sučelje	43
5.1 Korištene tehnologije i alati	43
5.2 Ispitivanje programskog rješenja	44
5.2.1 Ispitivanje komponenti	44
5.2.2 Ispitivanje sustava	46
5.3 Dijagram razmještaja	53
5.4 Upute za puštanje u pogon	54
6 Zaključak i budući rad	64
Popis literature	65
Indeks slika i dijagrama	66

1. Dnevnik promjena dokumentacije

Kontinuirano osvježavanje

Rev.	Opis promjene/dodatka	Autori	Datum
0.1	Napravljen predložak.	Adrian Golem, Lovre Marković	24.10.2022.
0.2	Dopisane upute za povijest dokumentacije. Dodane reference.	Adrian Golem, Lovre Marković	24.10.2022.
0.5	Dodan <i>Use Case</i> dijagram i jedan sekvencijski dijagram, funkcionalni i nefunkcionalni zahtjevi i dodatak A	Lovre Marković	30.10.2022.
0.6	Arhitektura i dizajn sustava, algoritmi i strukture podataka	Adrian Golem	04.11.2022.
0.7	Opisi obrazaca uporabe	Adrian Golem, Lovre Marković	07.11.2022.
0.8	Sekvencijski dijagrami	Lovre Marković	10.11.2022.
0.9	Dijagrami razreda	Adrian Golem	14.11.2022.

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

Rev.	Opis promjene/dodatka	Autori	Datum
1.0	Verzija samo s bitnim dijelovima za 1. ciklus	Adrian Golem, Lovre Marković	18.11.2022.
1.1	Dodan dijagram stanja, dijagram aktivnosti	Adrian Golem, Lovre Marković	21.12.2022.
1.2	Dodan dijagram komponenti	Lovre Marković	27.12.2022.
1.3	Ispitivanje komponenti, ispitivanje sustava	Adrian Golem, Lovre Marković	08.01.2023.
1.4	Dodan dijagram razmještaja	Lovre Marković	10.01.2023.
1.5	Dodane upute za puštanje u pogon	Adrian Golem	12.01.2023.
2.0	Konačni tekst predložka dokumentacije	Adrian Golem, Lovre Marković	13.01.2023.

2. Opis projektnog zadatka

Cilj ovog projekta je stvaranje programske podrške odnosno razvijanje aplikacije "Sinappsa". "Sinappsa" je web – forum studenata FER-a. Osmišljena je uzorom na nekadašnji forum "Studoši". Aplikacija služi kako bi pomogla studentima da razmjenjuju svoje znanje slanjem oglasa preko kojih se mogu dogovoriti na instrukcije ili bilo kakvu pomoć vezanu za gradivo. Na ovaj način bi studiranje bilo olakšano većini studenata.

Kada se aplikacija pokrene prikažu se trenutni aktivni oglasi na forumu te rang-lista 10 najboljih studenata pomagača. Korisnici se dijele na :

- neregistrirane
- registrirane
- moderatore

Neregistrirani korisnik kada pristupi aplikaciji može vidjeti sve oglase te rang-listu. No da bi odgovorio na oglas odnosno stvorio oglas mora biti prijavljen. Korisnik se mora registrirati tako da unese određene podatke :

- Ime
- Prezime
- Korisničko ime
- Avatar
- Službenu e-mail adresu fakulteta
- Lozinku

Nakon registracije korisnik (klijent) se može prijaviti u sustav. Prijavom u sustav klijent dobiva mogućnosti odgovaranja na oglase te postavljanje oglasa. Klijent može mijenjati osobne podatke, točnije : lozinku , korisničko ime te avatar.

- Korisničko ime
- Lozinku
- Avatar

Klijent ima mogućnost obrisati svoj korisnički račun.

Registrirani korisnici imaju mogućnost stvaranja oglasa. Oglas se stvara odabirom opcije "Stvori oglas" nakon čega se klijentu prikaže forma koju mora popuniti. Klijent ispunjava formu tako što odabire naslov oglasa, opis oglasa, odabire kolegij te kategoriju oglasa. Kategorija se odabire u padajućem izborniku. Kategorija može biti :

- Laboratorijska vježba
- Blic
- Gradivo
- Kontinuirani ispit
- Ispitni rok

Klijenti imaju opcije izmjenjivanja oglasa te brisanja oglasa.

Klijenti imaju opciju odgovaranja na oglas, slanjem upita. Klijent odabire opciju "Odgovori" te unosi odgovarajuću poruku (upit). Korisniku koji je stvorio oglas se šalje e-mail sa podacima korisnika koji je poslao upit. Nadalje se ova dva korisnika dogovaraju nepovezano s aplikacijom. No vidljivo je u aplikaciji da se događa komunikacija između njih. To se događa preko statusa upita. Status može biti :

- Prihvaćen
- Odbijen
- U tijeku

Status je vidljiv za oba korisnika.

Korisnici mogu vidjeti sve svoje upite, primljene i poslane. Prvobitni status upita je "U tijeku". Ako se korisnici ne uspiju dogovoriti oko termina instrukcija/pomoći onda korisnik koji je stvorio oglas odabire opciju "Odbij". Tom radnjom se status upita mijenja u "Odbijen". Tek nakon odrađenih instrukcija korisnik koji je stvorio oglas može odabrati opciju "Prihvaćen".

Nakon odrađenih instrukcija te nakon promjene statusa upita u "Prihvaćen" pošiljatelju upita se javlja opcija da ocijeni studenta-pomagača. Studentu se dodjeljuje ocjena od 1 do 10. Nešto slično kao i na aplikaciji „Bolt“ u kojoj se može ocijeniti vozača.



Slika 2.1: Primjer ocjenjivanja u Bolt aplikaciji

Vidljiv je prikaz najboljih 10 studenata-pomagača na rang-listi. Rang-lista se ažurira redovito.

Registriranim korisnicima je omogućena opcija "Moj Profil".

Tamo klijenti mogu vidjeti svoje osobne podatke, oglase i upite. Sljedeće osobne podatke mogu izmjenjivati: (lozinku, korisničko ime te avatar).

- Lozinku
- Korisničko ime
- Avatar

Oglase mogu vidjeti pod opcijama "Aktivni" i "Neaktivni". Oglase također mogu izmjenjivati, ali samo naslov i opis te ih obrisati.

Klijenti u svome profilu imaju uvid u svoje upite te njihov status. Klijenti se odjavljuju sa foruma odabirom opcije "Odjavi se".

Moderator je vrsta klijenta koja ima opcije da uklanja neželjene oglase iz foruma.

Ta radnja se odvija tako da Moderator odabere opciju za uklanjanje oglasa te mu se javlja opcija za slanje željene poruke korisniku. Moderator može ostaviti generičku poruku ili napisati neku svoju poruku te je poslati na e-mail adresu autora oglasa.

Moderator ima opciju pristupa bazi podataka te može dodavati nove kolegije s obzirom na smjer te brisati kolegije. Također ima opciju pregleda svih registriranih korisnika.

Forum će biti izveden kao web-aplikacija i biti će prilagođena za različite veličine ekrana.

Aplikacija će generalno biti na usluzi studentima FER-a. Postoji mogućnost proširenja aplikacije da bude dostupna korisnicima svih fakulteta , da svaki fakultet ima svoj forum.

To rješenje bi sigurno koristilo svim studentima raznih fakulteta. Fakulteti bi se nalazili u bazi podataka te bi korisnik registracijom odabrao fakultet. Nadalje student može gledati samo oglase za svoj fakultet.

3. Specifikacija programske potpore

3.1 Funkcionalni zahtjevi

Dionici:

1. Neregistrirani korisnici
2. Registrirani korisnici - oglašavači, odgovarači
3. Moderator
4. Razvojni tim

Aktori i njihovi funkcionalni zahtjevi:

1. Neregistrirani korisnik (inicijator) može:
 - (a) Vidjeti listu trenutno objavljenih oglasa
 - (b) Vidjeti rating student-pomagača
 - (c) Filtrirati oglase (prema smjeru, kolegijima, kategorijama)
 - (d) Registrirati se u sustav sa službenom FER e-mail adresom te navesti ime, prezime, korisničko ime, avatar i službenu e-mail adresu te željenu lozinku
2. Registrirani korisnik (Oglašavač / Odgovarač) (inicijator) može:
 - (a) Pregledavati i mijenjati osobne podatke(Moj profil)
 - (b) Izbrisati svoj korisnički račun
 - (c) Stvarati oglas (naslov, opis, kolegij, kategorija)
 - (d) Mogu pregledavati svoje oglase(aktivne, neaktivne, mijenjati ih te brisati)
 - (e) Mogu slati upit na oglas te mogu vidjeti sve svoje upite te njihove statuse(prihvaćen, odbijen, u tijeku.)
 - (f) Može ocijeniti student-pomagača koji mu je održao instrukcije
3. Moderator (inicijator) može:
 - (a) vidjeti popis svih registriranih korisnika i njihovih osobnih podataka

- (b) Uklanjati sve neželjene sadržaje te neprimjerene oglase
- (c) Slati mail korisniku čiji je oglas uklonio
- (d) Dodavati kolegije te označavati s obzirom na smjer

3.1.1 Obrasci uporabe

Opis obrazaca uporabe

UC1 - Pregled oglasa

- **Glavni sudionik:** Korisnik, klijent
- **Cilj:** Pregledati oglase
- **Sudionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
 1. Oglasi se prikazuju prilikom ulaska na forum
 2. Korisnik može filtrirati oglase
 3. Prikazuju se informacije o oglasima

UC2 - Pregled ratinga student-pomagača

- **Glavni sudionik:** Korisnik, klijent
- **Cilj:** Pregledati ratinge
- **Sudionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
 1. Korisnik ima prikazanu listu ratinga prikazanu po nadimcima

UC3 - Registracija

- **Glavni sudionik:** Korisnik
- **Cilj:** Stvoriti korisnički račun za pristup sustavu
- **Sudionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
 1. Korisnik ima prikazanu listu ratinga prikazanu po nadimcima
 2. Korisnik unosi potrebne korisničke podatke
 3. Korisnik prima obavijest o uspješnoj registraciji
- **Opis mogućih odstupanja:**
 - 2.a Odabir već zauzetog korisničkog imena i/ili e-maila, unos korisničkog podatka u nedozvoljenom formatu ili pružanje neispravnog e-maila
 1. Sustav obavještava korisnika o neuspjelom upisu
 2. Korisnik mijenja potrebne podatke te završava unos ili odustaje od registracije

UC4 - Prijava u sustav

- **Glavni sudionik:** Klijent
- **Cilj:** Dobiti pristup korisničkom sučelju
- **Sudionici:** Baza podataka
- **Preduvjet:** Registracija
- **Opis osnovnog tijeka:**
 1. Upis korisničkog imena i lozinke
 2. Potvrda o ispravnosti podataka
 3. Pristup korisničkim funkcijama
- **Opis mogućih odstupanja:**
 - 2.a Neispravno korisničko ime i/ili lozinka
 1. Sustav obavještava korisnika o neuspjelom upisu i vraća ga na stranicu za registraciju

UC5 - Pregled osobnih podataka

- **Glavni sudionik:** Klijent
- **Cilj:** Pregledati osobne podatke
- **Sudionici:** Baza podataka
- **Preduvjet:** Prijava u sustav
- **Opis osnovnog tijeka:**
 1. Korisnik odabire opciju "Profil"
 2. Korisnik može vidjeti svoje podatke

UC6 - Promjena osobnih podataka

- **Glavni sudionik:** Klijent
- **Cilj:** Promijeniti osobne podatke
- **Sudionici:** Baza podataka
- **Preduvjet:** Prijava u sustav
- **Opis osnovnog tijeka:**
 1. Korisnik odabere opciju za promjenu podataka
 2. Korisnik mijenja svoje osobne podatke
 3. Korisnik sprema promjene
 4. Baza podataka se ažurira
- **Opis mogućih odstupanja:**
 - 2.a Korisnik promijeni svoje osobne podatke, ali ne odabere opciju "Spremi promjenu"

1. Sustav obavještava korisnika da nije spremio podatke prije izlaska iz prozora

UC7 - Brisanje korisničkog računa

- **Glavni sudionik:** Klijent
- **Cilj:** Obrisati korisnički račun
- **Sudionici:** Baza podataka
- **Preduvjet:** Prijava u sustav
- **Opis osnovnog tijeka:**
 1. Korisnik pregledava osobne podatke
 2. Otvara se stranica s osobnim podacima korisnika
 3. Korisnik brise račun
 4. Korisnički račun se izbriše iz baze podataka
 5. Otvara se stranica za registraciju

UC8 - Stvaranje oglasa

- **Glavni sudionik:** Klijent
- **Cilj:** Stvoriti i objaviti oglas
- **Sudionici:** Baza podataka
- **Preduvjet:** Prijava u sustav
- **Opis osnovnog tijeka:**
 1. Korisnik odabire opciju za stvaranje novog oglasa
 2. Korisnik odabire kolegij te određeni kriterij za svoj oglas
 3. Korisnik objavljuje svoj oglas
 4. Oglas se sprema u bazu podataka

UC9 - Pregled svojih oglasa

- **Glavni sudionik:** Klijent
- **Cilj:** Pregledati sve oglase
- **Sudionici:** Baza podataka
- **Preduvjet:** Prijava u sustav
- **Opis osnovnog tijeka:**
 1. Korisnik odabire opciju "Moj Profil"
 2. Korisnik odabire opciju "Moji oglasi"
 3. Korisnik odabire opciju "Aktivni" ili "Neaktivni"
 4. Korisnik vidi oglase koje je objavio s obzirom na opciju

- **Opis mogućih odstupanja:**

- 2.a Korisnik nije objavio nijedan oglas

- 1. Korisniku se pod moji oglasi prikaže da nema objavljenih oglasa

UC10 - Promjena oglasa

- **Glavni sudionik:** Klijent
- **Cilj:** Promijeniti detalje izvornog oglasa
- **Sudionici:** Baza podataka
- **Preduvjet:** Prijava u sustav, objavljen oglas (postoji oglas u aktivnim oglasima)
- **Opis osnovnog tijeka:**
 - 1. Korisnik odabire opciju "Moji oglasi/aktivni oglasi"
 - 2. Korisnik odabire opciju "Izmjeni" pored oglasa kojeg želi izmijeniti
 - 3. Korisnik može mijenjati naslov i opis
 - 4. Korisnik odabire opciju spremi izmjene
 - 5. Oglas se izmjenjuje te se prikazuje izmijenjeni oglas
 - 6. Prvotni oglas se briše u bazi te se izmijenjeni sprema
- **Opis mogućih odstupanja:**
 - 2.a Korisnik želi promijeniti kolegij ili kriterij no to nije moguće
 - 1. Korisniku se prikazuje da mora obrisati oglas

UC11 - Brisanje oglasa

- **Glavni sudionik:** Klijent
- **Cilj:** Obrisati oglas
- **Sudionici:** Baza podataka
- **Preduvjet:** Klijent je prijavljen i napravio je barem jedan oglas
- **Opis osnovnog tijeka:**
 - 1. Klijent odabire opciju "Moji Oglasi"
 - 2. Prikaze se lista oglasa
 - 3. Klijent odabire oglas gdje odabire opciju obriši oglas
 - 4. Oglas se briše iz baze podataka

UC12 - Odgovaranje na oglas

- **Glavni sudionik:** Klijent odgovarač, klijent oglašavač
- **Cilj:** Poslati upit na oglas
- **Sudionici:** Baza podataka

- **Preduvjet:** Klijent je prijavljen
- **Opis osnovnog tijeka:**
 1. Korisnik na određenom oglasu odabere opciju "Odgovori na oglas"
 2. Korisnik upiše svoj upit te odabere opciju "Pošalji upit"
 3. Klijentu oglašavaču se pošalje e-mail sa porukom iz upita te podacima klijenta koji je poslao upit
 4. Pošiljatelju upita se upit pokaže u "Moji upiti/poslani" te status (prihvaćen, odbijen ili u tijeku)
 5. Primatelju se upit pojavi u "Moji upiti/primljeni"
 6. Upiti se spremaju u bazu podataka

UC13 - Pregled svojih upita

- **Glavni sudionik:** Klijent
- **Cilj:** Pregledati sve upite
- **Sudionici:** Baza podataka
- **Preduvjet:** Prijava u sustav, poslani upiti i/ili objavljeni oglasi
- **Opis osnovnog tijeka:**
 1. Korisnik odabire opciju "Moj Profil"
 2. 2 Korisnik odabire opciju "Moji Upiti"
 3. Korisnik odabire opciju "Poslani" ili "Primljeni"
 4. Korisnik vidi upite s obzirom na opciju
 5. U opciji Poslani pored upita još piše i status upita
- **Opis mogućih odstupanja:**
 - 2.a Korisnik nije objavio nijedan oglas ni poslao nijedan upit
 1. Korisniku se pod moji oglasi prikaže da nema objavljenih oglasa u primljeni
 2. Korisniku se prikazuje da nije poslao nijedan upit

UC14 - Prihvaćanje upita

- **Glavni sudionik:** Klijent
- **Cilj:** Prihvatiti upit nakon dogovorenog termina
- **Sudionici:** Baza podataka
- **Preduvjet:** Prijava u sustav, dogovoriti instrukcije i održati instrukcije
- **Opis osnovnog tijeka:**
 1. Klijent koji se nije uspio dogovoriti sa pošiljateljem upita ili nije u mogućnosti održati instrukcije pored određenog upita odabire opciju odbij

2. Ako se dogovore onda korisnik ništa ne mijenja nego ostaje status "U tijeku"
 3. Nakon održanih instrukcija klijent odabire opciju Prihvati
 4. Status upita se sprema u bazu podataka
 5. Status je vidljiv i pomagaču i pošiljatelju upita
- **Opis mogućih odstupanja:**
 - 2.a Korisnik nije odradio instrukcije, a objavio je da je
 1. Ovu opciju potvrđuje pošiljatelj upita te ako on kaže da ih nije održao, pomagaču se automatski upisuje 1 zvjezdica te mu se tim smanjuje rating

UC15 - Ocjenjivanje studenta-pomagača

- **Glavni sudionik:** Klijent
- **Cilj:** Ocijeniti studenta pomagača
- **Sudionici:** Baza podataka
- **Preduvjet:** Prijava u sustav, upit je "prihvaćen"
- **Opis osnovnog tijeka:**
 1. Nakon održanih instrukcija pored upita se pojavi "prihvaćen"
 2. Klijent klikom na status odgovara jeli primio instrukcije ili nije
 3. Ako je onda može odabrati opciju "Ocijeni mentora"
 4. Ako nije klijentu se šalje isprika te da će pomagač biti kažnjen
 5. Klijent koji je primio instrukcije pored prihvaćenog upita može odabrati opciju Ocijeni Mentora
 6. Pojavljuje se opcija odabrati broj zvjezdica te neobavezno poruku ispod zvjezdica
 7. Klijent objavi svoju ocjenu koja se sprema u bazu podataka
 8. Ažuriraju se ocjene na rating listi studenta-pomagača

UC16 - Pregled korisnika

- **Glavni sudionik:** Moderator
- **Cilj:** Pregledati registrirane korisnike
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je registriran i dodijeljena su mu prava moderatora
- **Opis osnovnog tijeka:**
 1. Moderator odabire opciju pregledavanja korisnika

2. Prikaže se lista svih ispravno registriranih korisnika s osobnim podacima

UC17 - Uklanjanje svih neželjenih oglasa

- **Glavni sudionik:** Moderator
- **Cilj:** Obrisati recenziju
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je registriran i dodijeljena su mu prava Moderator
- **Opis osnovnog tijeka:**
 1. Moderator odabere oglas za željeni kolegij
 2. Klikom na oglas prikaze mu se opcija "Izbriši"
 3. Nakon brisanja pojavi mu se opcija Odgovori oglašavaču
 4. Oglas se uklanja iz baze podataka

UC18 - Odgovaranje na neželjen oglas

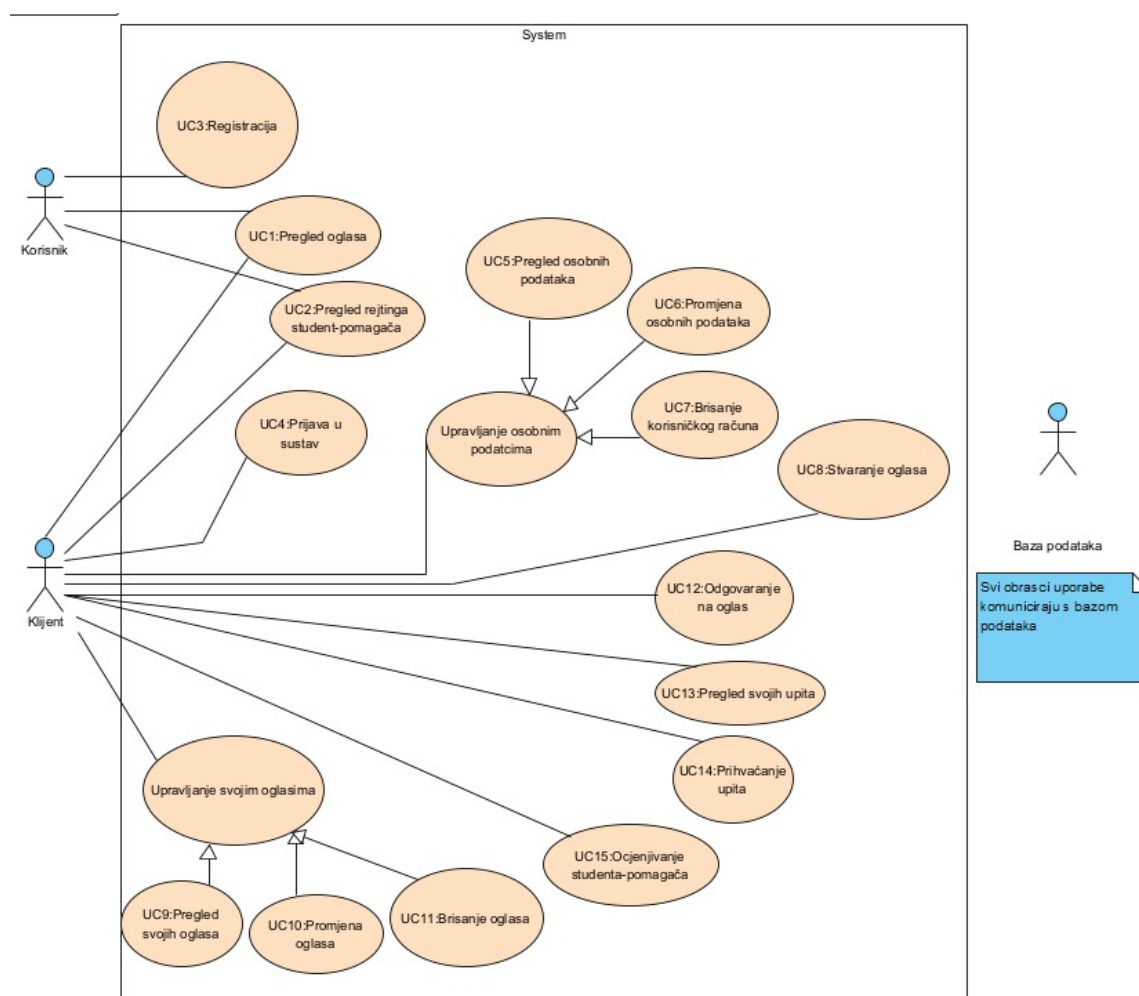
- **Glavni sudionik:** Moderator
- **Cilj:** Obrisati recenziju
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je registriran i dodijeljena su mu prava moderatora
- **Opis osnovnog tijeka:**
 1. Moderator odabire opciju Odgovori oglašavaču
 2. Moderatoru se javlja opcija gdje da napiše odgovor
 3. Kao placeholder je poruka "Vaš oglas nije u skladu sa pravima aplikacije"
 4. Nakon što Moderator odabere opciju "Pošalji", mail se šalje oglašavaču

UC19 - Dodavanje kolegija

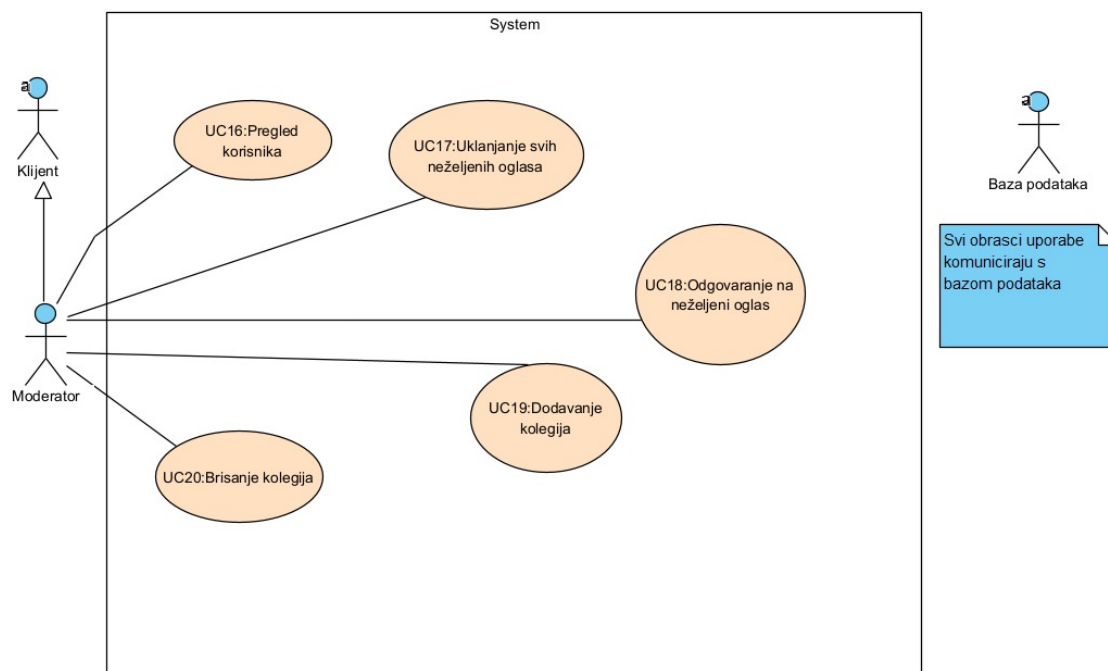
- **Glavni sudionik:** Moderator
- **Cilj:** Dodati kolegij
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je registriran i dodijeljena su mu prava moderatora
- **Opis osnovnog tijeka:**
 1. Moderator odabire opciju Dodaj kolegij
 2. Moderatoru se javlja opcija gdje mora upisati ime kolegija i smjer
 3. Moderator odabire opciju "Dodaj"
 4. Kolegij se sprema u bazu podataka

UC20 - Brisanje kolegija

- **Glavni sudionik:** Moderator
- **Cilj:** Obrisati kolegij
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je registriran i dodijeljena su mu prava moderatora
- **Opis osnovnog tijeka:**
 1. Moderator odabire opciju Obriši kolegij
 2. Moderatoru se javlja opcija gdje mora izabrati odgovarajući kolegij
 3. Moderator odabire opciju "Obriši"
 4. Kolegij se briše iz baze podataka

Dijagrami obrazaca uporabe

Slika 3.1: Dijagram obrasca uporabe : Korisnik - Klijent

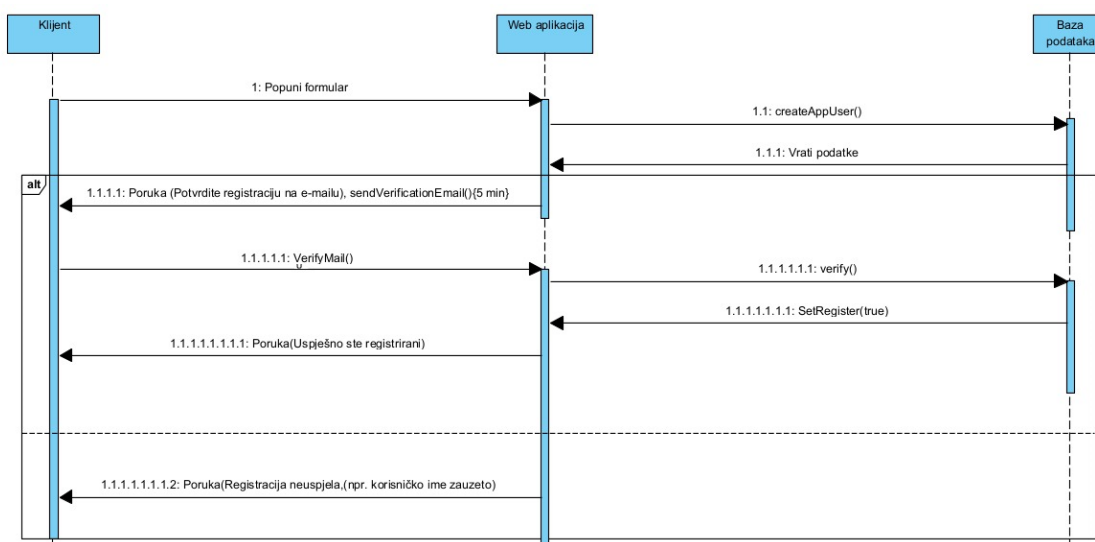


Slika 3.2: Dijagram obrasca uporabe: Moderator - Klijent

3.1.2 Sekvencijski dijagrami

Obrazac uporabe UC3 – Registracija

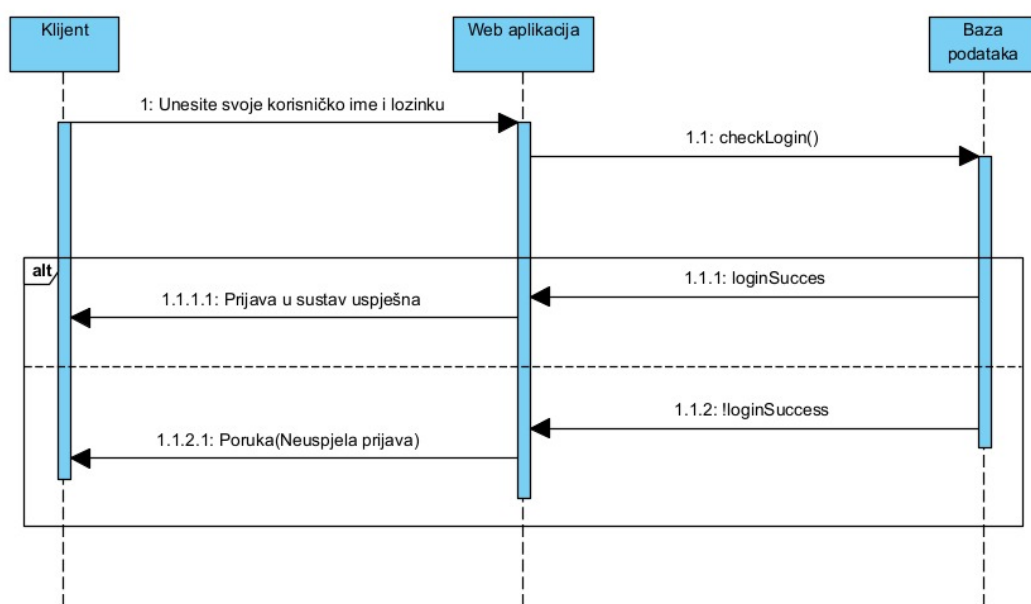
Klijent traži dopuštenje da odgovori na oglas, te mu se javlja da mora biti prijavljen. Korisnik nema korisnički račun te se mora registrirati. Korisnik ispunjava formular registracije. Podatci se šalju u bazu podataka te provjerava e-mail i korisničko ime. Ako se te tri stavke registracije ne podudaraju sa stavkama drugih klijenta onda se korisnik uspješno registrirao te dobiva e-mail poruku na adresu te mora kliknuti link na svome računalu da bih se uspio registrirati. Ako korisnik koristi isti dio registracije s nekim dobije odgovarajuću poruku na ekranu. Ako je registracija uspješna klijent može odgovarati te stvarati oglase.



Slika 3.3: Sekvencijski dijagram za UC3

Obrazac uporabe UC4 – Prijava u sustav

Da bi korisnik stvorio oglas mora biti prijavljen. Korisniku se klikom na opciju stvori oglas javlja da se mora prijaviti u sustav. Ako nema korisnički račun mora se registrirati. Korisnik se prijavljuje u sustav korištenjem korisničkog imena te lozinke. Podatci se šalju aplikacijom te se provjerava u bazi podataka postoji li račun sa odgovarajućim imenom te lozinkom. Ako postoji korisniku se dozvoljava prijava te stvaranje oglasa.

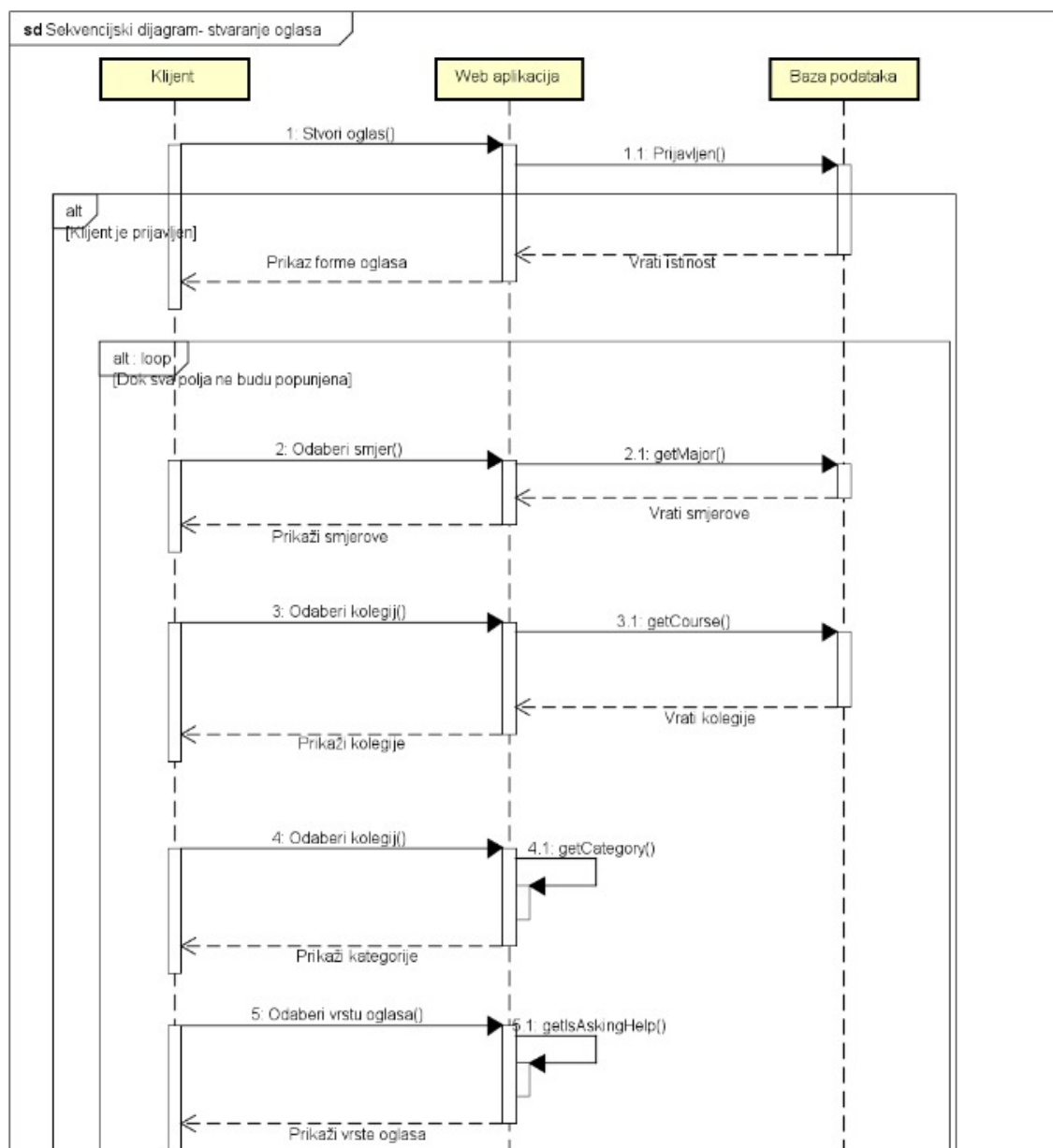


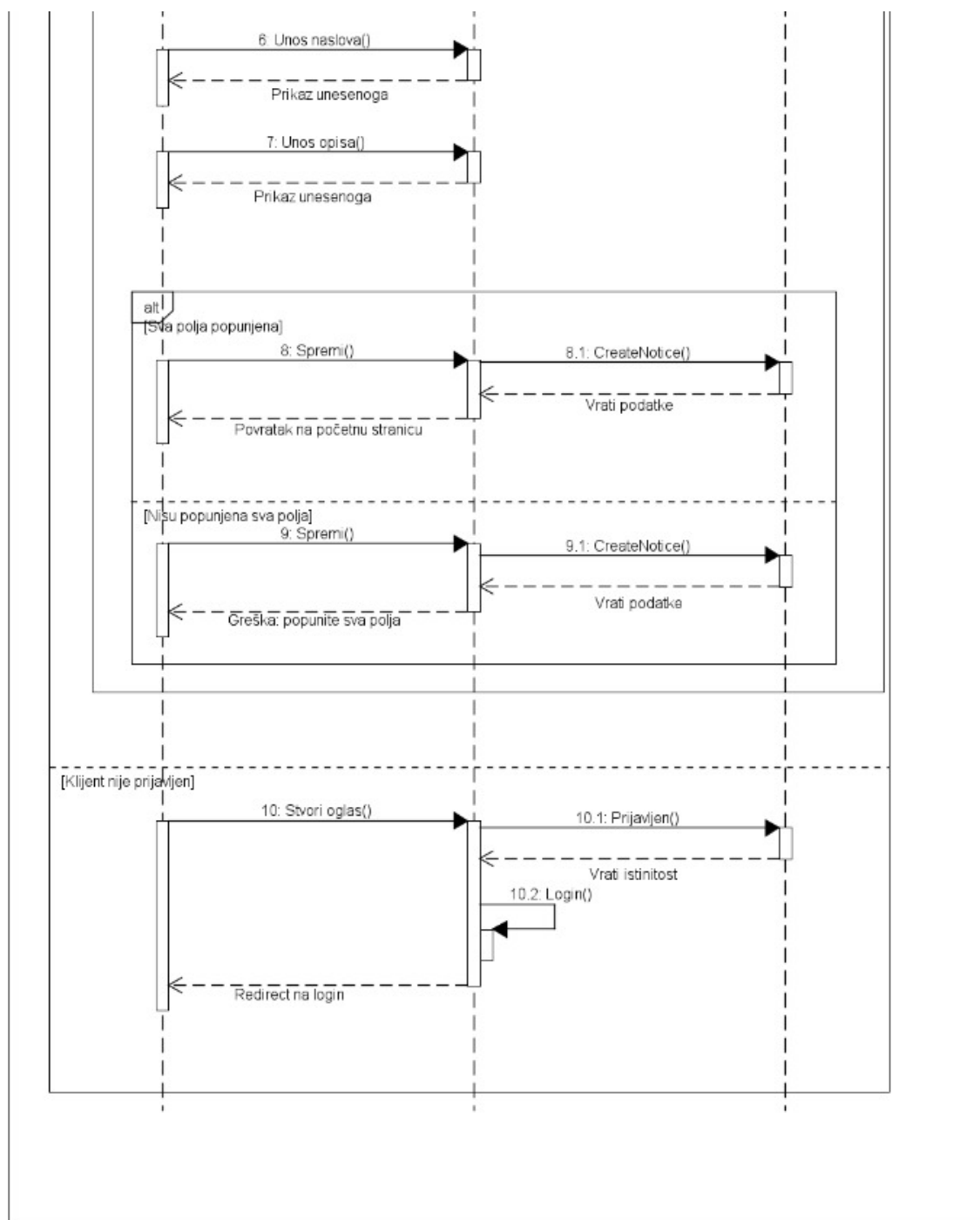
Slika 3.4: Sekvencijski dijagram za UC4

Obrazac uporabe UC8 – Stvaranje oglasa

Klijent odabire opciju "Stvori oglas" te ako je prijavljen u sustav otvara mu se forma za stvaranje oglasa koju mora popuniti. Ako nije prijavljen, aplikacija ga vodi na login dio gdje se može prijaviti.

Forma sadrži 4 padajuće liste te dva polja gdje klijent mora upisati tekst. Klijent u padajućim listama bira smjer, kolegij, kategorija te vrstu oglasa. Nakon toga obavezan je popuniti naslov oglasa i opis oglasa. Ako nije popunio opis i naslov, a kliknuo je opciju „Spremi“, aplikacija ga upozorava koja polja mora popuniti te mu ne dopušta da objavi takav oglas. Tek nakon što je klijent popunio sva polja, klikom na „Spremi“ se njegov oglas sprema u bazu podataka te ga aplikacija vraća na početnu stranicu. Nakon svega navedenog, klijent se nalazi na početnoj stranici gdje može vidjeti sve oglase objavljene te u njima može pronaći i svoj novonastali oglas.





Slika 3.5: Sekvencijski dijagram za UC8

3.2 Ostali zahtjevi

- Sustav treba omogućiti rad više korisnika u stvarnom vremenu
- Aplikacija mora biti izvedena kao web aplikacija
- Korisničko sučelje i sustav moraju podržavati hrvatsku abecedu (dijakritičke znakove) pri unosu i prikazu tekstualnog sadržaja
- Izvršavanje dijela programa u kojem se pristupa bazi podataka ne smije trajati duže od nekoliko sekundi
- Sustav treba biti implementiran kao web aplikacija koristeći objektno-orijentirane jezike
- Aplikacija mora biti prilagođena za različite veličine ekrana (responsive design)
- Neispravno korištenje korisničkog sučelja ne smije narušiti funkcionalnost i rad sustava
- Veza s bazom podataka mora biti kvalitetno zaštićena, brza i otporna na vanjske greške
- Pristup sustavu mora biti omogućen iz javne mreže pomoću HTTPS
- Sustav treba biti jednostavan za korištenje, korisnici se moraju znati koristiti sučeljem bez opširnih uputa
- Nadogradnja sustava ne smije narušavati postojeće funkcionalnosti sustava

4. Arhitektura i dizajn sustava

Općenito, arhitektura se može podijeliti na 3 podsustava:

- *Baza podataka*
- *Web poslužitelj*
- *Web aplikacija*

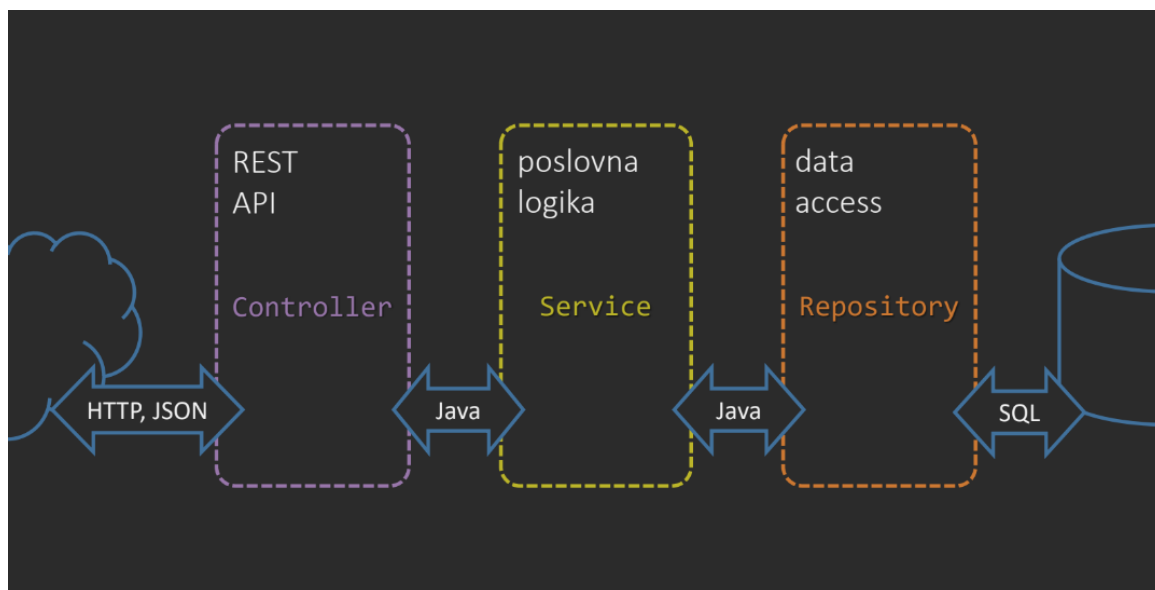
Web preglednik je program koji korisniku omogućuje pregled web-stranica i multimedijalnih sadržaja vezanih uz njih. Osim što omogućuje pregledavanje web sadržaja on je ujedno i prevoditelj koji kod stranice prevodi u format koji je običnom korisniku jasan za pregledavanje. Putem web preglednika korisnik šalje zahtjev web poslužitelju.

Web poslužitelj je centralni dio rada web aplikacije. Njegov zadatak je omogućiti komunikaciju između klijenta i aplikacije preko HTTP protokola. Zahtjev koji je dobio od web preglednika šalje web aplikaciji za obradu.

Web aplikacija zadužena je za potpunu obradu zahtjeva. Može također i pristupiti bazi podataka ako je to za pojedini zahtjev potrebno. Nakon obrade šalje odgovor korisniku preko poslužitelja u obliku HTML dokumenta kojeg web preglednik prevodi.

Prilikom izrade web stranice primarno ćemo koristiti programski jezik Java uz Spring Boot radni okvir te programski jezik JavaScript uz biblioteku React. Koristit ćemo ih primarno u razvojnem okruženju IntelliJ IDEA uz manje korištenje Microsoft Visual Studio Code-a.

Sustav je temeljen na Controller-Service-Repository arhitekturi. Koncept funkcionira ovako :



Slika 4.1: Controller-Service-Repository arhitektura

- **Controller** - isključivo odgovoran za izlaganje funkcionalnosti vanjskim entitetima, koristi se za pozivanje Service-a
- **Service** - u Service-u se nalazi kompletna poslovna logika, ako se navedena logika sastoji od dohvaćanja/spremanja podataka, poziva se Repository
- **Repository** - odgovoran za pristup bazi podataka te pohranu i dohvaćanje podataka

4.1 Baza podataka

Za potrebe naseg sustava koristit ćemo relacijsku bazu podataka koja svojom strukturom olakšava modeliranje stvarnog svijeta. Gradivna jedinka baze je relacija, odnosno tablica koja je definirana svojim imenom i skupom atributa. Zadatak baze podataka je brza i jednostavna pohrana, izmjena i dohvat podataka za daljnju obradu. Baza podataka ove aplikacije sastoji se od sljedećih entiteta:

- APP_USER
- NOTICE
- COURSE
- MAJOR
- QUERY
- APP_USER_Grade

4.1.1 Opis tablica

APP_USER Ovaj entitet sadržava sve važne informacije o korisniku aplikacije. Sadrži attribute: jedinstveni identifikator, korisničko ime, adresu elektronske pošte, ime, prezime, avatar, lozinku, ostvarene bodove, verifikacijski kod te informaciju je li korisnik registriran i ako je moderator. Ovaj entitet je u vezi *One-to-Many* s entitetom NOTICE preko atributa id korisnika. Također je u vezi *One-to-Many* s entitetom QUERY preko atributa id korisnika.

APP_USER	tip podatka	opis atributa
id	BIGINT	jedinstveni identifikator korisnika
nickname	VARCHAR	jedinstveno korisničko ime
email	VARCHAR	službena adresa elektronske pošte
first_name	VARCHAR	ime korisnika
last_name	VARCHAR	prezime korisnika
avatar	VARCHAR	sličica koju korisnik bira za prikaz na profilu
password	VARCHAR	lozinka za prijavu u sustav

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

APP_USER	tip podatka	opis atributa
ranking_points	INT	bodovi stečeni pomaganjem i sudjelovanjem u raspravama
verification_code	VARCHAR	verifikacijski kod za registraciju korisnika
is_registered	BOOLEAN	podatak o statusu registracije korisnika (registracija je važeća nakon potvrde verifikacijskog koda)
is_moderator	BOOLEAN	podatak o statusu moderatora

NOTICE Ovaj entitet sadržava sve važne informacije o objavljenom oglasu. Sadrži attribute: jedinstveni identifikator, smjer studija, kategoriju oglasa, kolegij za koji je oglas vezan, naslov oglasa, opis oglasa te informaciju o aktivnosti oglasa. Ovaj entitet je u vezi *Many-to-One* s entitetom APP_USER sa stranim ključem *nickname* koji se referencira na istoimenu atribut u tablici APP_USER. Nadalje, u vezi je *One-to-Many* s entitetom QUERY preko identifikatora oglasa i u vezi *Many-to-One* s entitetom COURSE preko identifikatora kolegija. Također je u vezi *Many-to-One* s entitetom MAJOR preko identifikatora smjera.

NOTICE	tip podatka	opis atributa
id	BIGINT	jedinstveni identifikator oglasa
user_id	BIGINT	jedinstveno identifikator korisnika (APP_USER.id)
course_id	BIGINT	jedinstveni identifikator kolegija (COURSE.id)
major_id	BIGINT	jedinstveni identifikator smjera (MAJOR.id)
category	VARCHAR	kategorija oglasa (laboratorijska vježba, blic, gradivo, kontinuirani ispit, ispitni rok)
title	VARCHAR	naslov oglasa

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

NOTICE	tip podatka	opis atributa
description	VARCHAR	opis oglasa
is_active	BOOLEAN	podatak o aktivnosti oglasa

COURSE Ovaj entitet sadržava sve važne informacije o kolegiju. Sadrži attribute: jedinstveni identifikator kolegija, identifikator smjera te ime kolegija. Ovaj entitet je u vezi *Many-to-One* s entitetom MAJOR preko identifikatora smjera. Također je u vezi *One-to-Many* s entitetom NOTICE preko identifikatora kolegija.

COURSE	tip podatka	opis atributa
id	BIGINT	jedinstveni identifikator kolegija
major_id	BIGINT	jedinstveni identifikator smjera (MAJOR.id)
course_name	VARCHAR	ime kolegija

MAJOR Ovaj entitet sadržava sve važne informacije o smjeru. Sadrži attribute: jedinstveni identifikator smjera te ime smjera koje može biti "računarstvo" ili "elektrotehnika". Ovaj entitet je u vezi *One-to-Many* s entitetom COURSE preko atributa id smjera. Također je u vezi *One-to-Many* s entitetom NOTICE preko identifikatora smjera.

MAJOR	tip podatka	opis atributa
id	BIGINT	jedinstveni identifikator upita na oglas
major_name	VARCHAR	ime smjera

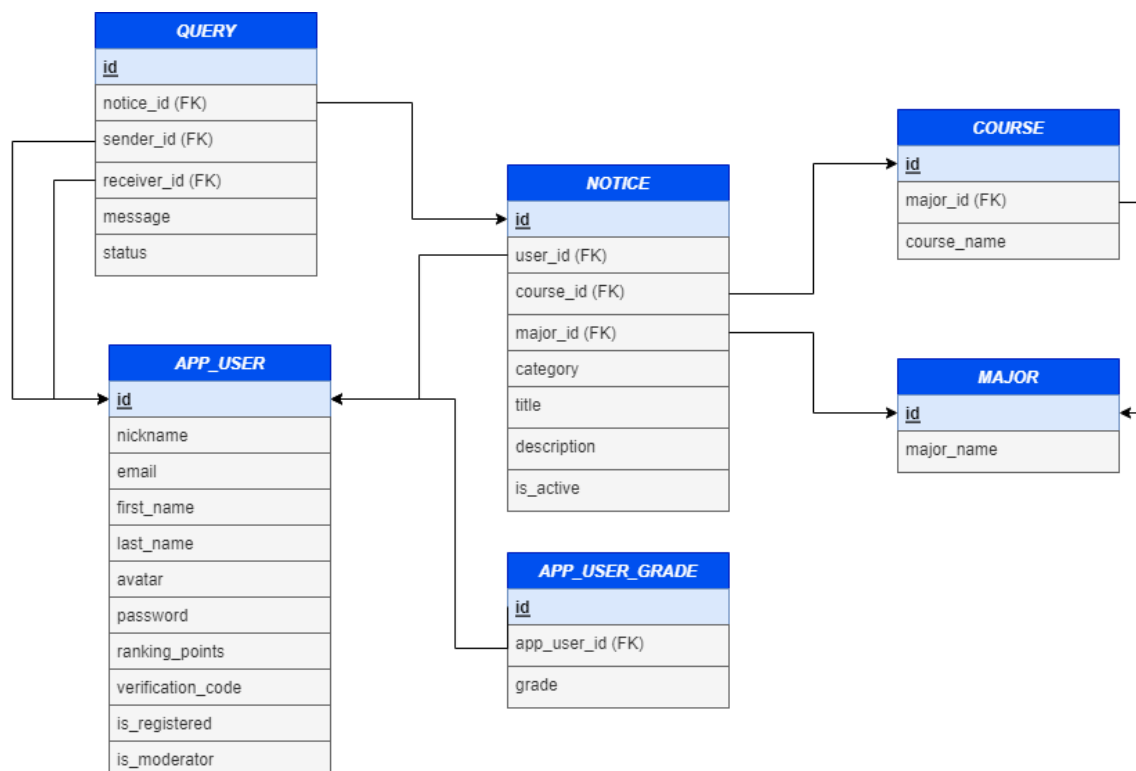
QUERY Ovaj entitet sadržava sve važne informacije o upitu na oglas. Sadrži attribute: jedinstveni identifikator upita, identifikator oglasa, identifikatore pošiljatelja i primatelja, te poruku i status oglasa. Ovaj entitet je u vezi *Many-to-One* s entitetom NOTICE preko atributa id oglasa. Također je u vezi *Many-to-One* s entitetom APP_USER preko identifikatora pošiljatelja i primatelja.

QUERY	tip podatka	opis atributa
id	BIGINT	jedinstveni identifikator upita na oglas
notice_id	BIGINT	jedinstveni identifikator oglasa (NOTICE.id)
sender_id	BIGINT	jedinstveni identifikator korisnika (APP_USER.id)
receiver_id	BIGINT	jedinstveni identifikator korisnika (APP_USER.id)
message	VARCHAR	poruka upita na oglas
status	VARCHAR	status upita na oglas

APP_USER_GRADE Ovaj entitet sadržava informacije o korisnikovim ocjenama. Sadrži attribute: jedinstveni identifikator ocjene, identifikator korisnika te ocjenu. Ovaj entitet je u vezi *Many-to-One* s entitetom APP_USER preko identifikatora korisnika.

APP_USER_GRADE	tip podatka	opis atributa
id	BIGINT	jedinstveni identifikator ocjene
app_user_id	BIGINT	jedinstveni identifikator korisnika (APP_USER.id)
grade	DOUBLE	korisnikova ocjena

4.1.2 Dijagram baze podataka



Slika 4.2: E-R dijagram baze podataka

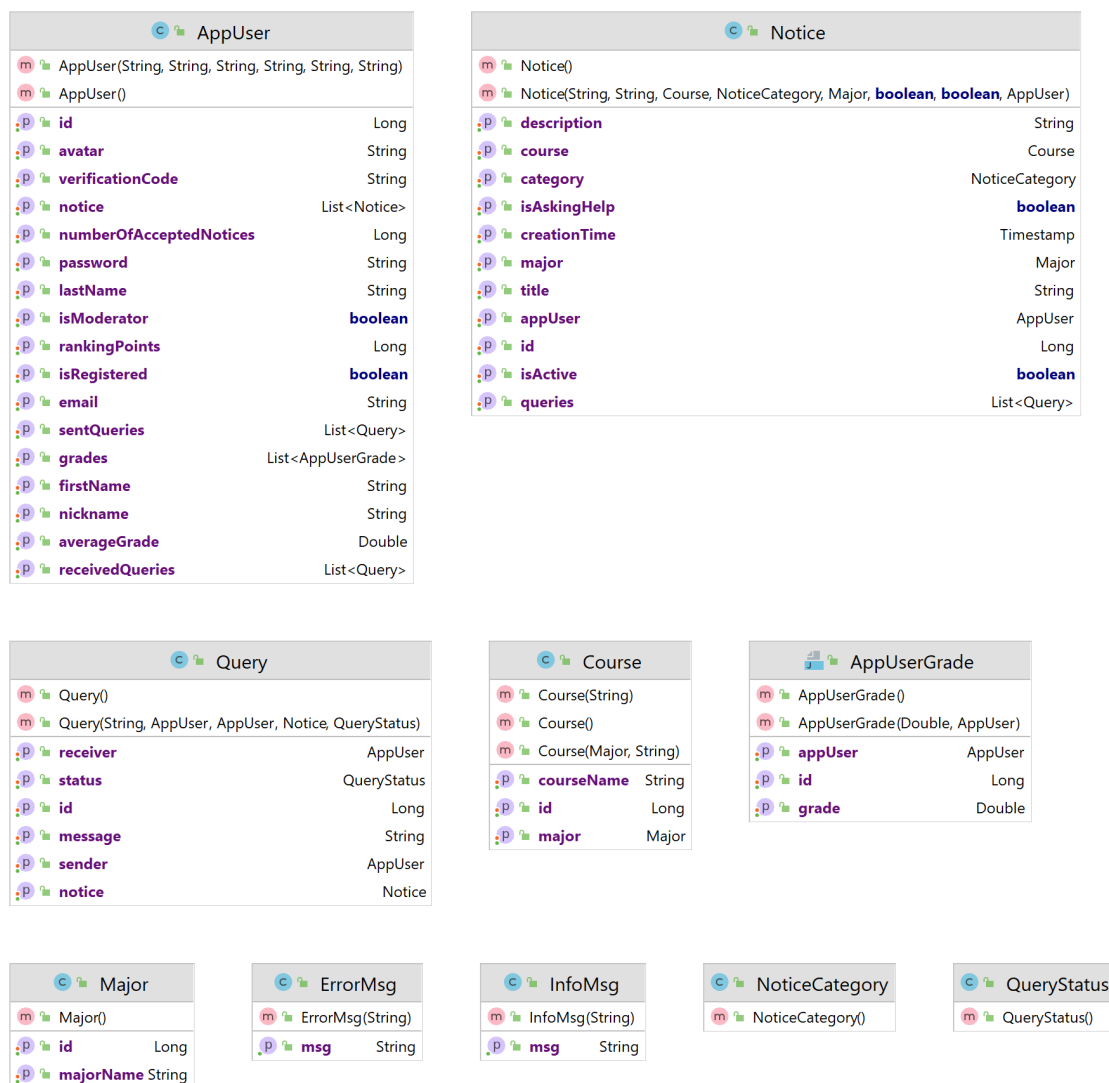
4.2 Dijagram razreda

Na slikama 4.3, 4.4 i 4.5, 4.6 i 4.7 i 4.8 su prikazani razredi koji pripadaju backend dijelu *Controller-Service-Repository* arhitekture. Razredi prikazani na slici 4.4 nalaze se u controller folderu. Metode implementirane u tim razredima manipuliraju klasama iz service foldera (slika 4.5). Nadalje, metodama iz service foldera pristupa se repository dijelu korištene arhitekture. Metode implementirane u controller razredima vraćaju JSON datoteke s html status kodom. '

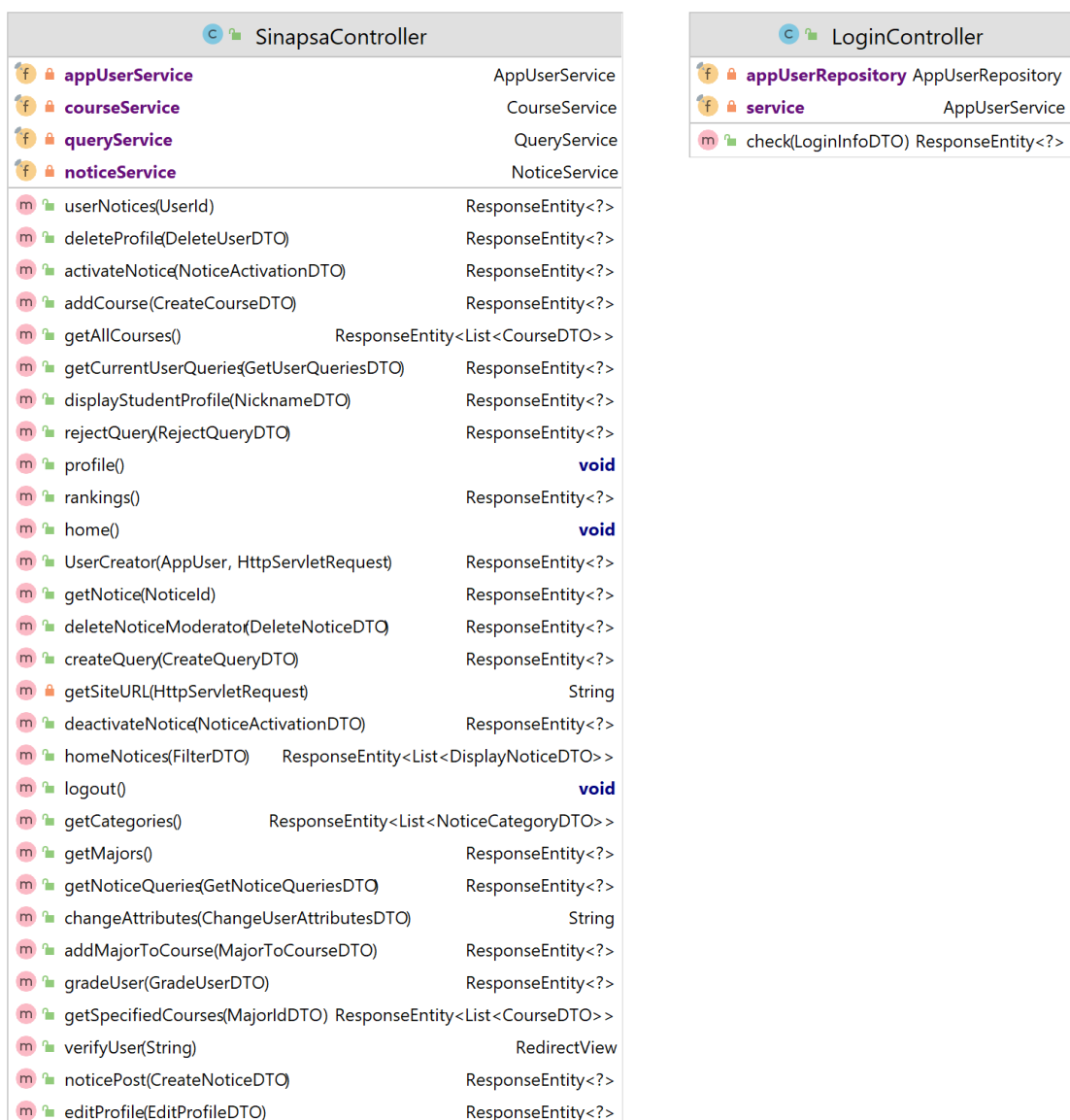
Zbog lakše organizacije, razredi su podijeljeni logički po pravu pristupa metodama određenih aktera.

Kako bi se smanjila prenapučenost unutar dijagrama, prikazane su samo ovisnosti između razreda koji pripadaju istom dijelu dijagrama.

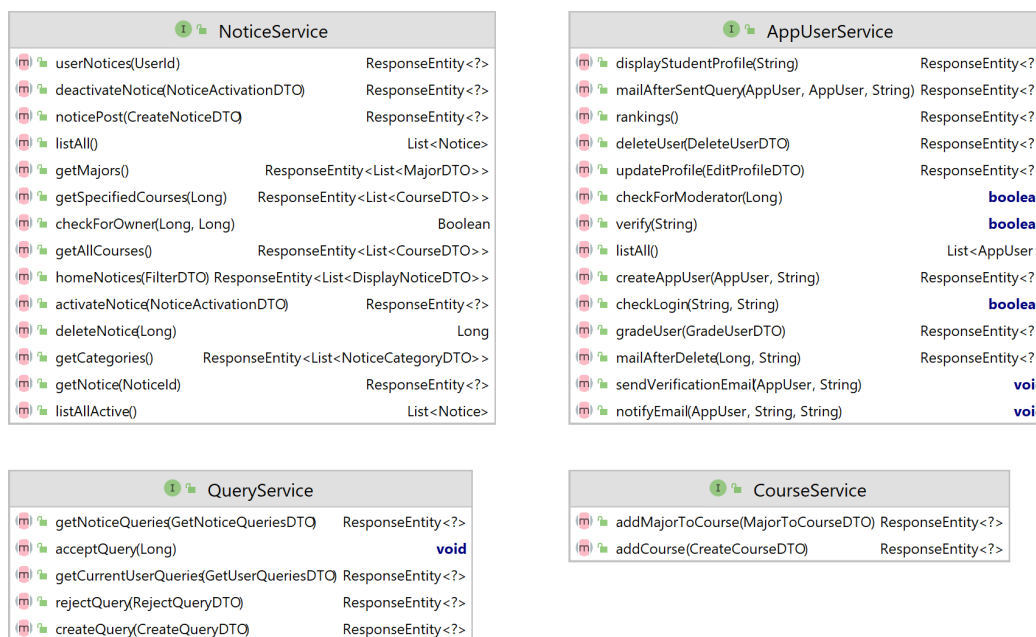
Iz naziva i tipova atributa u razredima može se zaključiti vrsta ovisnosti među različitim razredima.



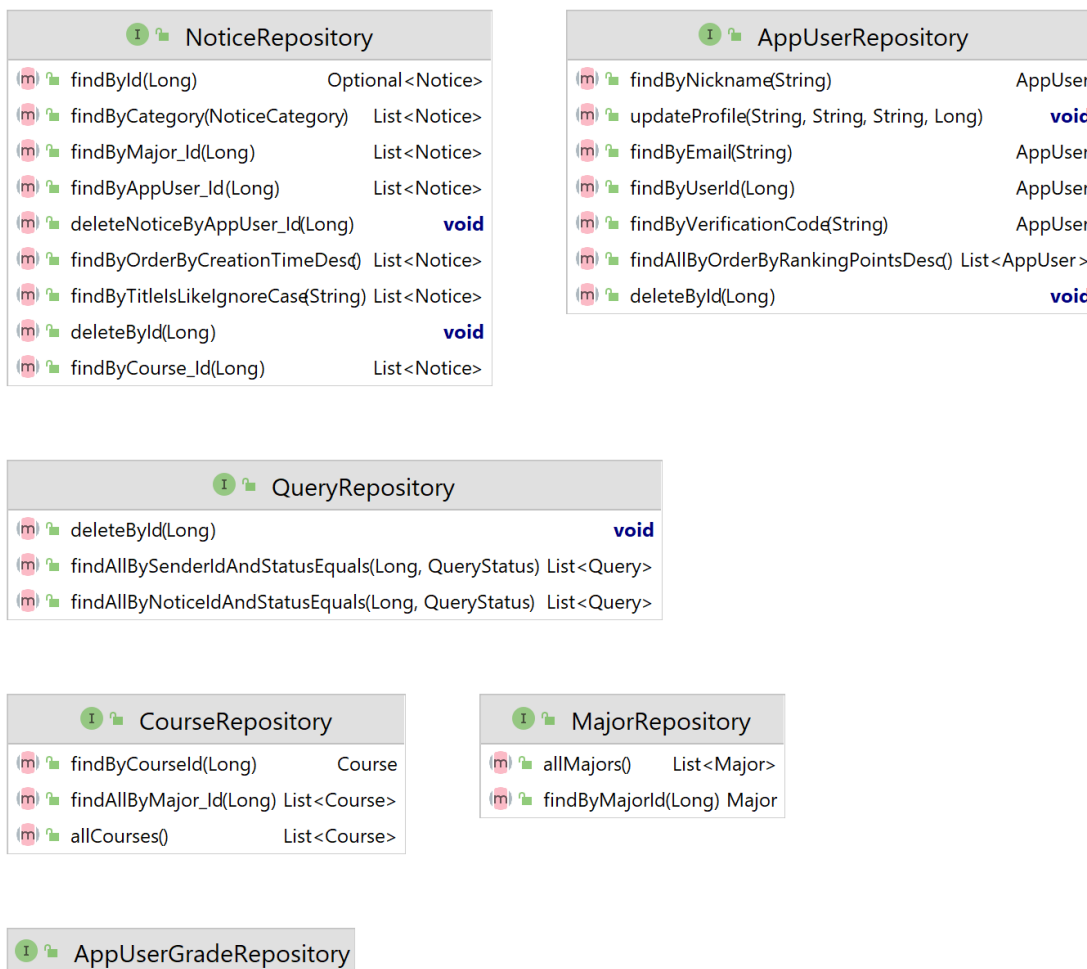
Slika 4.3: Dijagram razreda - dio Entities



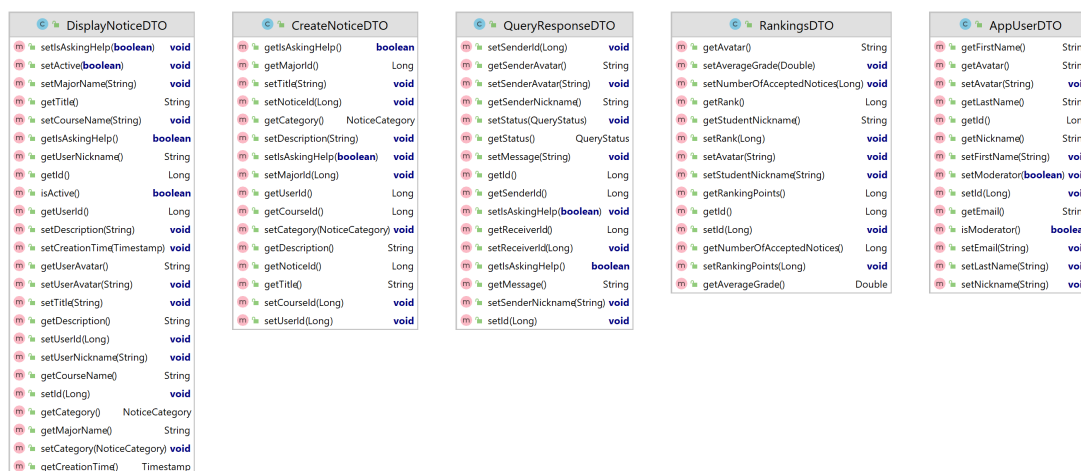
Slika 4.4: Dijagram razreda - dio Controllers

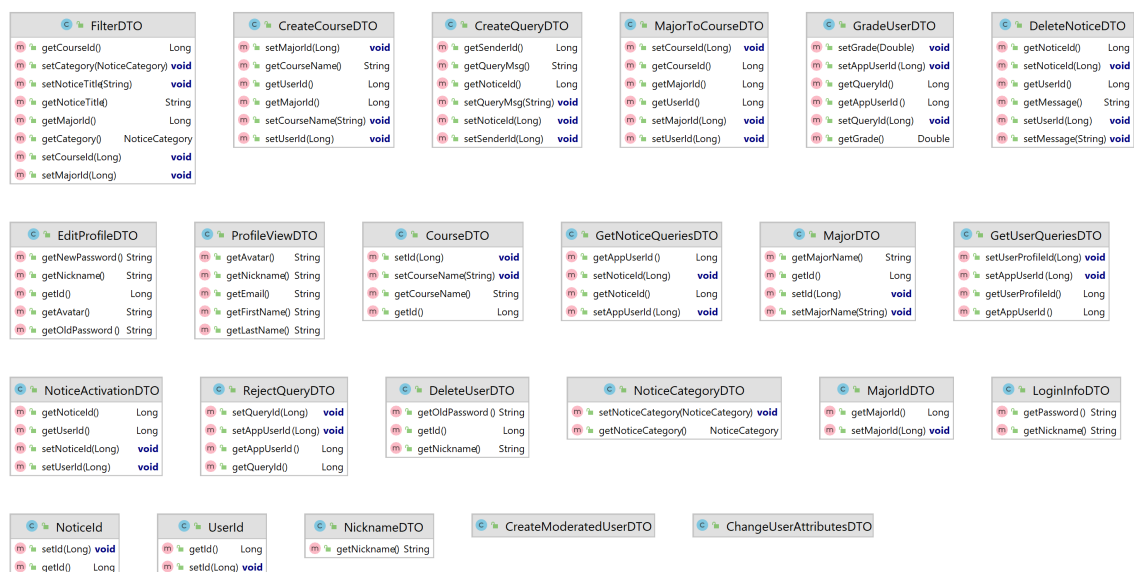


Slika 4.5: Dijagram razreda - dio Service



Slika 4.6: Dijagram razreda - dio Repository





Slika 4.7: Dijagram razreda - dio Data Transfer Objects

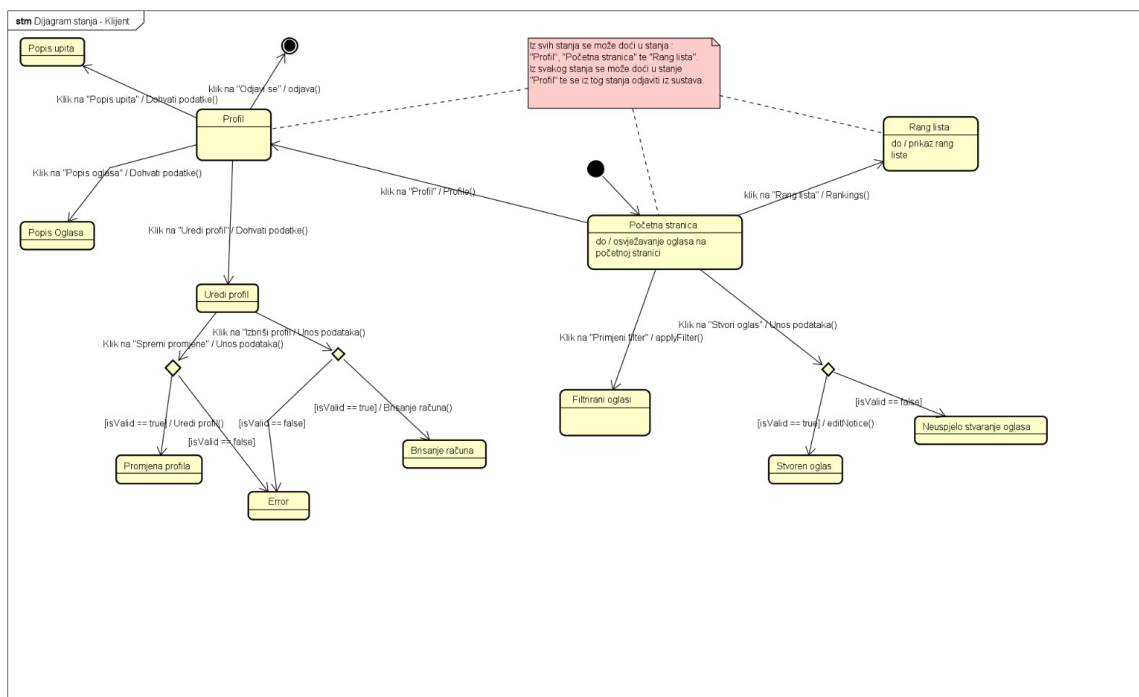


Slika 4.8: Dijagram razreda - dio Security

4.3 Dijagram stanja

Dijagram stanja prikazuje stanja objekta te prijelaze iz stanja temeljeno na događajima. Na slici je prikazan dijagram stanja za klijenta aplikacije Sinapps.

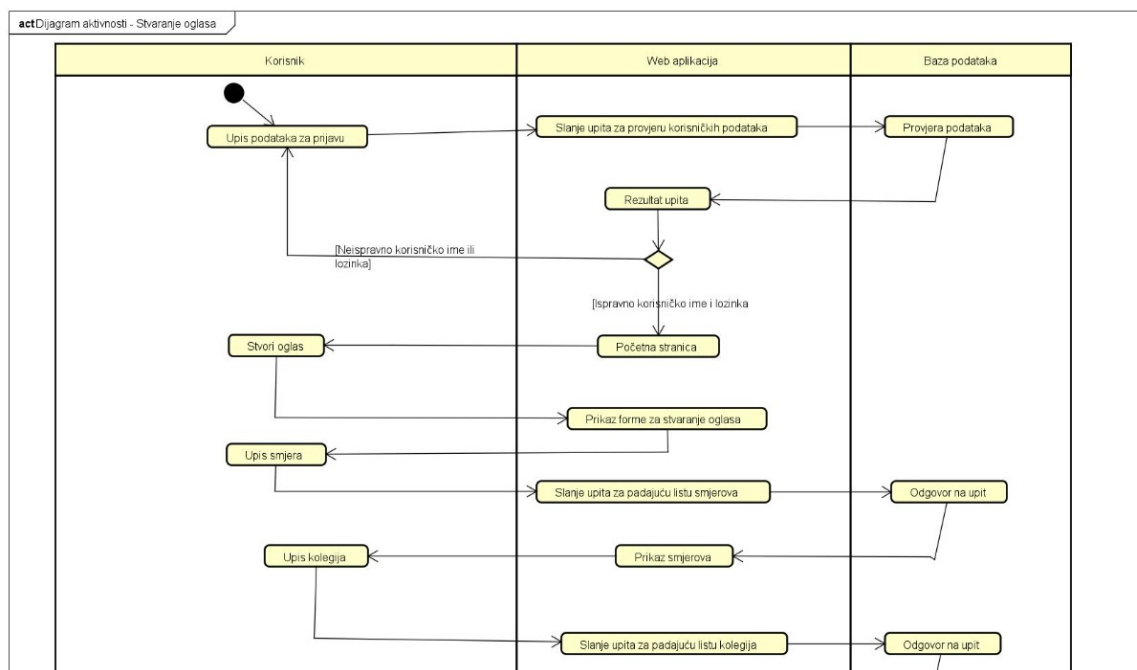
Nakon prijave klijent se nalazi na početnoj stranici gdje su mu vidljivi svi oglasi koji su objavljeni. Ima mogućnost primijeniti filter te vidjeti filtrirane oglase. Također ima opciju stvaranja oglasa te ako klikne tu opciju javlja mu se forma za stvaranje oglasa. Nakon što ju ispravno popuni vraća se na početnu stranicu gdje može vidjeti svoj oglas. Klijent ima još dvije opcije u zaglavlju početne stranice. Može odabrati opciju „Rang lista“ te može vidjeti trenutni poredak svih studenata-pomagača. Zadnju opciju koju klijent ima je da može odabrati opciju „Profil“. Ako odabere tu opciju, može vidjeti svoje podatke, svoje upite te oglase. Također ima opciju uređivanja osobnih podataka te naravno i mogućnost odjave.

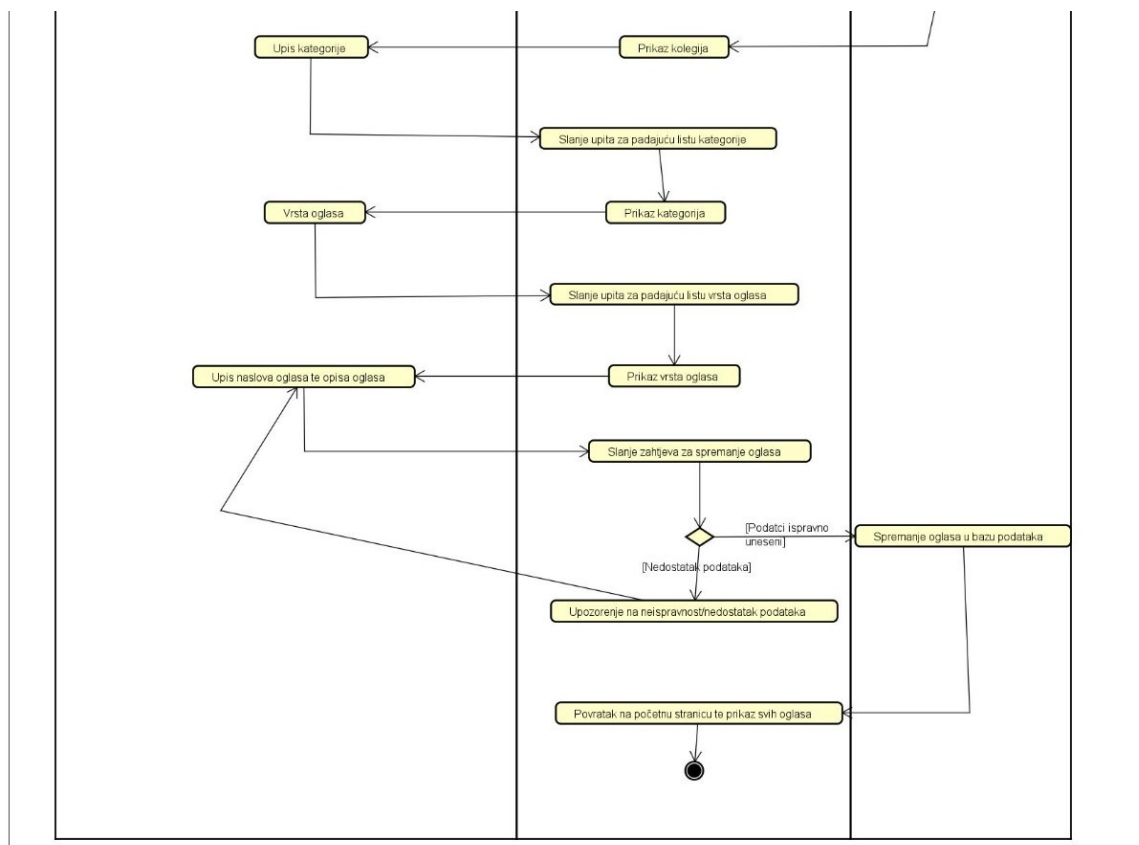


Slika 4.9: Dijagram stanja klijenta

4.4 Dijagram aktivnosti

Dijagram aktivnosti koristi se za opis toka podataka. Upotrebljava se za modeliranje toka te se svaki novi korak može započeti izvršavati tek nakon što je prethodni završen. Dijagram aktivnosti teži jednostavnosti. Na slici je prikazan dijagram aktivnosti kod stvaranje oglasa. Klijent se prijavi u sustav te nakon toga odabire opciju "Stvori oglas". Prikazuje mu se forma oglasa koju mora popuniti da bi mogao objaviti svoj oglas. Klijent unese sve potrebne podatke te odabire opciju "Spremi". Ako nešto nije u redu ispunjeno ostaje na formi te mu se javlja odgovarajuća poruka. Ako je sve u redu, aplikacija vraća klijenta na početnu stranicu gdje će moći pronaći svoj oglas u svim oglasima.

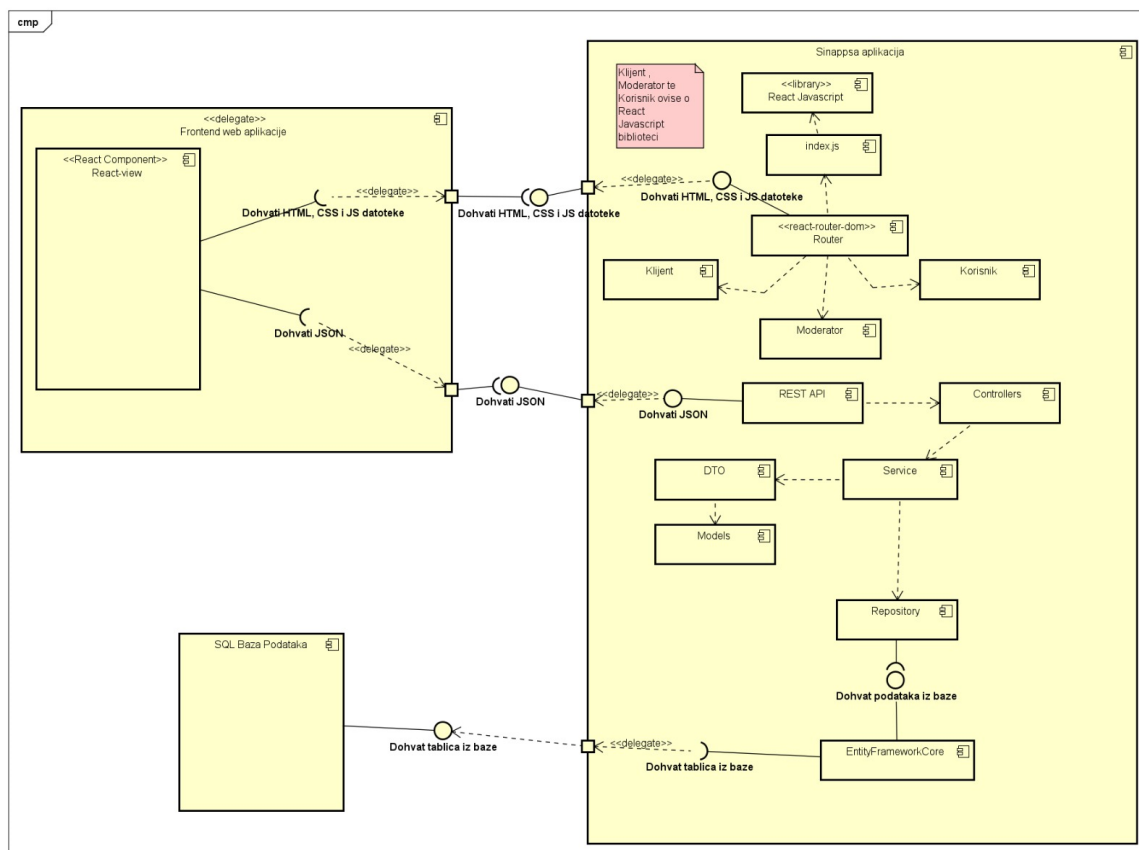




Slika 4.10: Dijagram aktivnosti - Stvaranje oglasa

4.5 Dijagram komponenti

Dijagram komponenti prikazuje organizaciju i ovisnost među komponentama te prikazuje odnos komponenata sa okolinom. Frontend dio aplikacije sadrži Router koji na upit s url određuje koja će se datoteka poslužiti na sučelje. To sučelje služi za dohvat HTML, CSS i JS datoteka. Frontend dio još sadrži niz JavaScript datoteka koje su raspoređene u određene logičke cjeline. Podijeljene su po tipovima aktera koji im pristupaju. Sve JavaScript datoteke ovise o React biblioteci koja sadrži gotove komponente (gumbi, forme i sl.). REST API komponentama se pristupa preko sučelja za dohvat JSON podataka. REST API poslužuje podatke koji pripadaju backend dijelu aplikacije. EntityFrameworkCore je zadužen za dohvaćanje tablica iz baze podataka pomoću SQL naredbi. Podaci iz baze dalje se šalju u "Repository" dio backend dijela aplikacije. Nadalje se "service" dio backenda služi podacima iz "Repository" dijela te odrađuje glavni dio posla na backend dijelu. "Controler" dio backenda povezuje REST API sa "service" dijelom te je tako povezan frontend i backend.



Slika 4.11: Dijagram komponenti

5. Implementacija i korisničko sučelje

5.1 Korištene tehnologije i alati

Komunikacija u timu je realizirana putem aplikacije Whatsapp. Za izradu UML dijagrama korišteni su alati Astah Professional te Visual Paradigm. Sustav za upravljanje izvornim kodom je Git. Udaljeni repozitorij je dostupan na platformi GitLab. Kao razvojno okruženje korišten je Microsoft Visual Studio - integrirano je razvojno okruženje (IDE) tvrtke Microsoft. Prvenstveno se koristi za razvoj računalnih programa za operacijski sustav Windows, kao i za web-stranice, web-aplikacije, web-usluge i mobilne aplikacije. Visual Studio za razvoj softvera koristi Microsoftove platforme kao što su Windows API, Windows Forms, Windows Presentation Foundation, Windows Store i Microsoft Silverlight.

Aplikacija je ostvarena korištenjem radnog okvira Spring Boot za backend dio aplikacije te jezik Java. Za frontend dio aplikacije korišten je React te jezik JavaScript. React, odnosno ReactJS je biblioteka jezika JavaScript za izgradnju korisničkih sučelja. React se koristi kao osnova za razvoj web ili mobilnih aplikacija, a složenije aplikacije koriste i dodatne biblioteke za interakciju s API-jem. Spring Boot okvir nudi programerima već gotova rješenja nekih metoda te znatno ubrzava rad i razvoj aplikacije. Omogućava rad aplikacije bez da aplikacija mora ovisiti o vanjskom web serveru ugrađivanjem web server poput Tomcat-a u aplikaciju tijekom inicijalizacije.

Baza podataka se nalazi na Renderu.

5.2 Ispitivanje programskog rješenja

5.2.1 Ispitivanje komponenti

U sljedećem kodu testiramo `getNoticeQueries()` metodu u servisu. U bazi imamo spremljen notice i njegov pripadajući oglas i provjeravamo hoće li nam test vratiti pripadajući query za traženi notice.

```
@Test
    @Order(1)
    void getNoticeQueries() {
        GetNoticeQueriesDTO getNoticeQueriesDTO = new
            GetNoticeQueriesDTO(5L, 1L);
        ResponseEntity<?> response =
            queryServiceJpa.getNoticeQueries(getNoticeQueriesDTO);

        Assertions.assertEquals(response.getStatusCode(), HttpStatus.OK);
    }
```

U priloženom kodu testiramo funkciju `gradeUser()` gdje ocjenjujemo korisnika. U bazi imamo spremljenog korisnika te njegovu prosječnu ocjenu koju uspoređujemo sa ocjenom koju smo dohvatili.

```
@Test
    @Order(2)
    void gradeUser() {
        GradeUserDTO gradeUserDTO = new GradeUserDTO(6L, 1L, 4D);
        appUserServiceJpa.gradeUser(gradeUserDTO);

        AppUser gradedUser = appUserRepository.findById(2L);

        Assertions.assertEquals(gradedUser.getAverageGrade(), 4);
    }
```

U funkciji `editNotice()` stvaramo oglas. Iz baze dohvaćamo oglas te provjeravamo jednakost kategorija oglasa. Ako se one podudaraju prolazi test, a ako ne test pada.

```
@Test
    @Order(4)
```

```
void editNotice() {  
    CreateNoticeDTO createNoticeDTO = new CreateNoticeDTO("Notice1",  
        "testni notice 1",4L ,3L, NoticeCategory.GRADIVO, 1L, 5L, true);  
    noticeServiceJpa.noticePost(createNoticeDTO);  
  
    Notice notice = noticeRepository.findById(5L).get();  
  
    Assertions.assertEquals(notice.getCategory(),  
        NoticeCategory.GRADIVO);  
}
```

Funkcijom `getUserNotices()` testiramo dohvaćanje svih oglasa nekog korisnika. U funkciji učitalamo upite korisnika kojeg imamo u bazi. Testiramo listu, ako je prazna znači da test ne prolazi, a ako nije prazna znači da test prolazi.

```
@Test  
@Order(5)  
void getUserNotices() {  
    UserId id = new UserId();  
    id.setId(1L);  
    noticeServiceJpa.userNotices(id);  
    List<Notice> notices = noticeRepository.findAll();  
  
    Assertions.assertFalse(notices.isEmpty());  
}
```

U priloženom kodu funkcijom `getHomeNotices()` testiramo dohvaćanje oglasa sa početne stranice. Oglas koji imamo u bazi učitalamo u listu. Ako je ta lista prazna znači da nam test pada, a ako sadrži oglas znači da test prolazi.

```
@Test  
@Order(6)  
void getHomeNotices() {  
    FilterDTO filterDTO = new FilterDTO("Notice1", null, null, null);  
    noticeServiceJpa.homeNotices(filterDTO);  
    List<Notice> notices = noticeRepository.findAll();  
  
    Assertions.assertFalse(notices.isEmpty());  
}
```

Funkcijom `deleteNotice()` testiramo brisanje notice-a. U bazi imamo notice spremljen te ga brišemo. Učitavamo sve notice te testiramo hoće li lista biti prazna, ako je prazna onda funkcija ispravno radi.

```
@Test
```

```
    @Order(7)
```

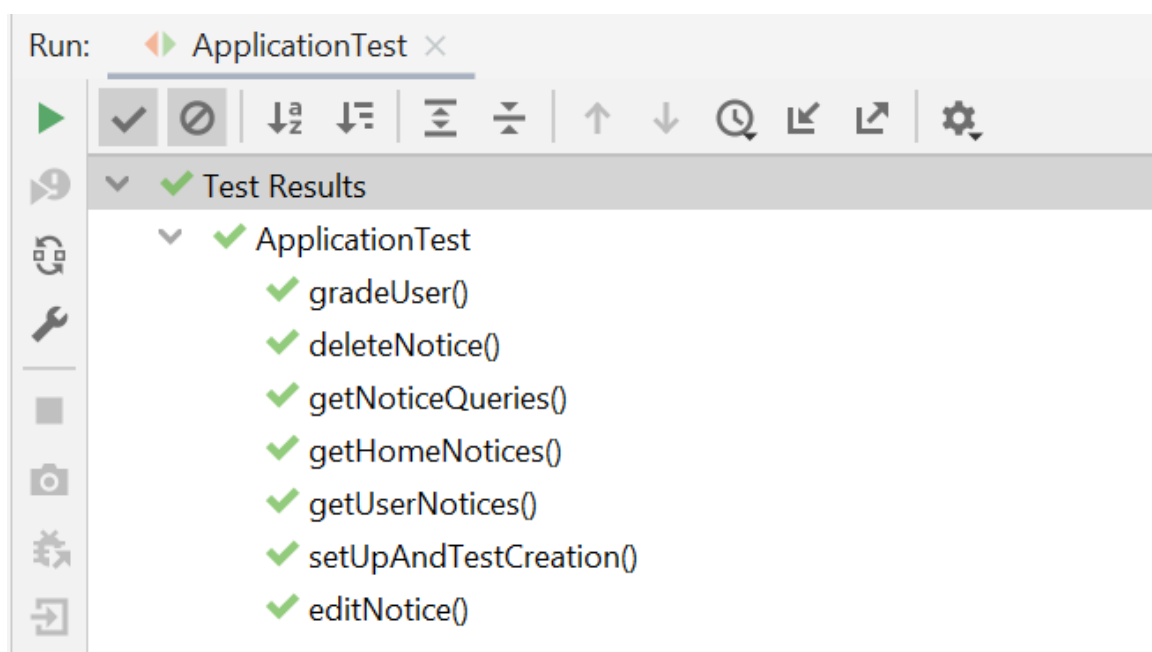
```
    void deleteNotice() {
```

```
        noticeServiceJpa.deleteNotice(5L);
```

```
        List<Notice> notices = noticeRepository.findAll();
```

```
        Assertions.assertTrue(notices.isEmpty());
```

```
    }
```



Slika 5.1: Rezultati ispitivanja komponenti

5.2.2 Ispitivanje sustava

Metoda `loginWithGoodCredentials()` logira se u deployanu aplikaciju sa ispravnim korisničkim imenom i lozinkom. Tada bi nas aplikacija trebala preusmjeriti na home stranicu.

```
@Test
```

```
void loginWithGoodCredentials() throws InterruptedException{
```

```
System.setProperty("webdriver.chrome.driver", "C:/Program Files
    (x86)/ChromeDriver/chromedriver.exe");
WebDriver driver = new ChromeDriver();

driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

driver.get("https://sinapsafront.onrender.com/sinapsa/login");
WebElement element;

element = driver.findElement(By.id("nickname"));
element.sendKeys("giento");

element = driver.findElement(By.id("password"));
element.sendKeys("keksudin");

driver.findElement(By.cssSelector("button[type='submit']")).click();
Thread.sleep(4000);

String redirectURL = driver.getCurrentUrl();
boolean goodCredentials = !redirectURL.contains("login");

assertTrue(goodCredentials);

driver.quit();
}
```

loginWithWrongPassword() je metoda za test logiranja sa pogrešnom lozinkom, u tom slučaju ostajemo na istoj login stranici uz grešku da je uneseno pogrešno korisničko ime ili lozinka.

@Test

```
void loginWithWrongPassword() throws InterruptedException{
    System.setProperty("webdriver.chrome.driver", "C:/Program Files
        (x86)/ChromeDriver/chromedriver.exe");
    WebDriver driver = new ChromeDriver();

    driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
```



```
driver.get("https://sinappsafrent.onrender.com/sinapsa/login");
WebElement element;

element = driver.findElement(By.id("nickname"));
element.sendKeys("giento");

element = driver.findElement(By.id("password"));
element.sendKeys("jurudin");

driver.findElement(By.cssSelector("button[type='submit']")).click();
Thread.sleep(4000);

String redirectURL = driver.getCurrentUrl();
boolean wrongPassword = redirectURL.contains("login");

assertTrue(wrongPassword);

driver.quit();
}
```

createNotice() stvara novi oglas na ispravan način (postavljen naslov i opis oglasa). Nakon ispravnog objavljivanja oglasa, aplikacija će nas preusmjeriti na home stranicu.

```
@Test
void createNotice() throws InterruptedException{
    System.setProperty("webdriver.chrome.driver", "C:/Program Files
        (x86)/ChromeDriver/chromedriver.exe");
    WebDriver driver = new ChromeDriver();

    driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

    //LOGGING IN
    driver.get("https://sinappsafrent.onrender.com/sinapsa/login");
    WebElement element;

    element = driver.findElement(By.id("nickname"));
    element.sendKeys("giento");
```

```
element = driver.findElement(By.id("password"));
element.sendKeys("keksudin");

driver.findElement(By.cssSelector("button[type='submit']")).click();
Thread.sleep(4000);

driver.findElement(By.name("createNotice-btn")).click();
Thread.sleep(5000);

//CREATING NOTICE
element = driver.findElement(By.id("title"));
element.sendKeys("Naslov oglasa");

element = driver.findElement(By.id("description"));
element.sendKeys("Opis Oglasa");

driver.findElement(By.cssSelector("button[type='submit']")).click();
Thread.sleep(2000);

String redirectURL = driver.getCurrentUrl();
boolean hasDescription = !redirectURL.contains("editNotice");

assertTrue(hasDescription);

driver.quit();
}
```

createNoticeWithNoDescription() testira ponašanje objave oglasa ako se radi o pokušaju objave oglasa bez da je napisan opis oglasa. U tom slučaju ostajemo i dalje na stranici za kreiranje oglasa, odnosno oglas se neće objaviti i nećemo biti preusmjereni na home stranicu.

```
@Test
void createNoticeWithNoDescription() throws InterruptedException{
    System.setProperty("webdriver.chrome.driver", "C:/Program Files
        (x86)/ChromeDriver/chromedriver.exe");
    WebDriver driver = new ChromeDriver();
```

```
driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

//LOGGING IN
driver.get("https://sinappsafont.onrender.com/sinapsa/login");
WebElement element;

element = driver.findElement(By.id("nickname"));
element.sendKeys("giento");

element = driver.findElement(By.id("password"));
element.sendKeys("keksudin");

driver.findElement(By.cssSelector("button[type='submit']")).click();
Thread.sleep(4000);

driver.findElement(By.name("createNotice-btn")).click();
Thread.sleep(2000);

//CREATING NOTICE
element = driver.findElement(By.id("title"));
element.sendKeys("Naslov oglasa");

driver.findElement(By.cssSelector("button[type='submit']")).click();
Thread.sleep(2000);

String redirectURL = driver.getCurrentUrl();
boolean hasDescription = !redirectURL.contains("editNotice");

assertFalse(hasDescription);

driver.quit();
}
```

changeNicknameToUsedNickname() pokušava promijeniti korisnikov nickname u nickname koji već koristi neki drugi korisnik. U tom slučaju nećemo biti preusmjereni dalje, već ćemo dobiti obavijest da je navedeni nickname zauzet i ostati na stranici za promjena podataka.

@Test

```
void changeNicknameToUsedNickname() throws InterruptedException{
    System.setProperty("webdriver.chrome.driver", "C:/Program Files
        (x86)/ChromeDriver/chromedriver.exe");
    WebDriver driver = new ChromeDriver();

    driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

    //LOGGING IN
    driver.get("https://sinappsafont.onrender.com/sinapsa/login");
    WebElement element;

    element = driver.findElement(By.id("nickname"));
    element.sendKeys("giento");

    element = driver.findElement(By.id("password"));
    element.sendKeys("keksudin");

    driver.findElement(By.cssSelector("button[type='submit']")).click();
    Thread.sleep(4000);

    //CREATING NOTICE
    driver.get("https://sinappsafont.onrender.com/sinapsa/editProfile");

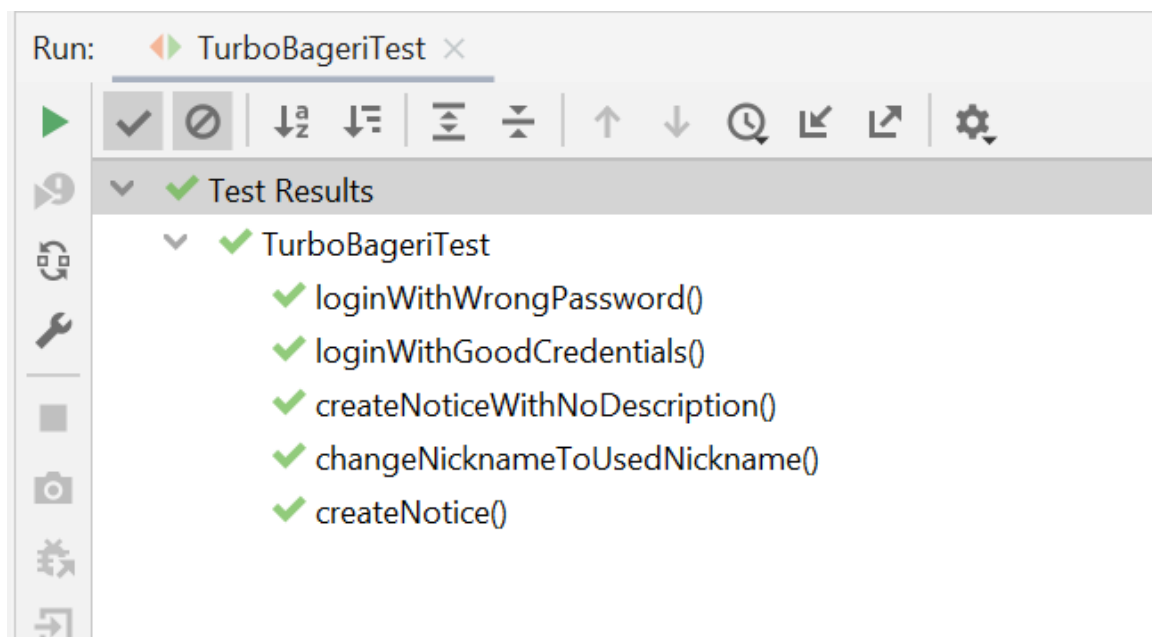
    element = driver.findElement(By.id("nickname"));
    element.clear();
    element.sendKeys("Keksa");

    element = driver.findElement(By.id("newPassword"));
    element.sendKeys("keksudin");

    element = driver.findElement(By.id("oldPassword"));
    element.sendKeys("keksudin");

    driver.findElement(By.name("Update")).click();
    Thread.sleep(2000);
```

```
String redirectURL = driver.getCurrentUrl();  
boolean sameNickname = redirectURL.contains("editProfile");  
  
assertTrue(sameNickname);  
  
driver.quit();  
}
```

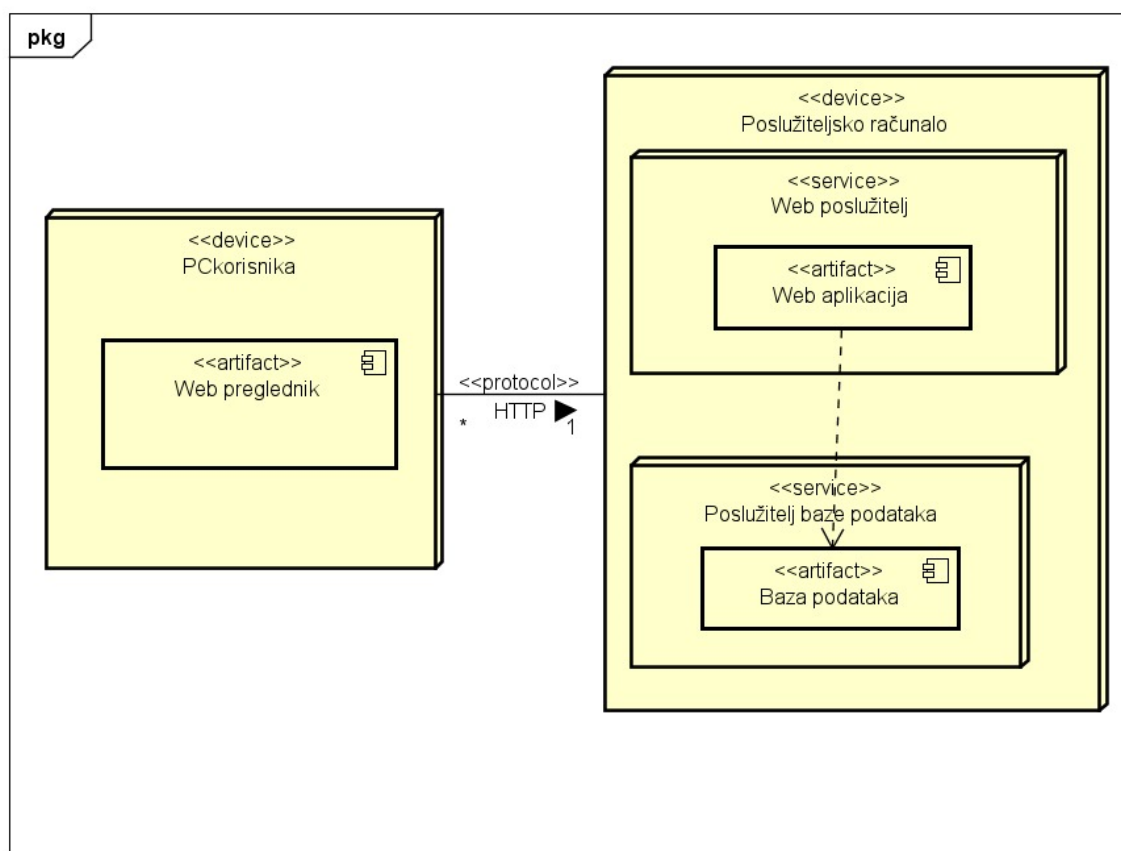


Slika 5.2: Rezultati ispitivanja sustava

5.3 Dijagram razmještaja

Dijagram razmještaja opisuje topologiju sklopovlja i programsku potporu koja se koristi u implementaciji sustava u njegovom radnom okruženju.

Dijele se na: specifikacijski, razmještaja instanci, implementacijski te dijagram mrežne arhitekture. Specifikacijski dijagram na slici opisuje aplikaciju "Sinapps". Klijent preko svog web preglednika pristupa web aplikaciji. Na poslužiteljskoj strani se nalazi web poslužitelj te poslužitelj baze podataka. Klijent i poslužitelj komuniciraju pomoću HTTP veza.



Slika 5.3: Dijagram razmještaja

5.4 Upute za puštanje u pogon

Izrada Render korisničkog računa

Za početak, potrebno je kreirati račun na [linku](#). Preporučeno je registrirati se putem GitLab računa ili spojiti svoj GitLab račun nakon registracije kako bi se moglo pristupiti projektu.

Priprema frontenda za deploy

Potrebno je u *package.json* dodati dependencye potrebne za deploy, posebnu pažnju treba obratiti na *http-proxy-middleware*, *dotenv* i *express*.

```
{
  "dependencies": {
    "@testing-library/jest-dom": "^5.16.5",
    "@testing-library/react": "^13.4.0",
    "@testing-library/user-event": "^13.5.0",
    "@types/jest": "^27.5.2",
    "@types/node": "^16.11.65",
    "@types/react": "^18.0.21",
    "@types/react-dom": "^18.0.6",
    "axios": "^0.24.0",
    "dotenv": "^16.0.3",
    "express": "^4.18.2",
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "react-scripts": "5.0.1",
    "typescript": "^4.8.4",
    "web-vitals": "^2.1.4",
    "http-proxy-middleware": "^2.0.6"
  }
}
```

Zatim je potrebno dodati *setupProxy.js* koji služi kao proxy server za lokalni razvoj (preusmjera API pozive na *localhost:8080*).

```
// setupProxy.js
const { createProxyMiddleware } = require("http-proxy-middleware");

module.exports = function (app) {
  app.use(
    "/api",
    createProxyMiddleware({
      target: "http://localhost:8080/",
      changeOrigin: true,
    })
  );
};
```

Treba dodati i *app.js*, u kojem se nalazi express server za produkcijski proxy i posluživanje frontenda.

```
const express = require("express");
const { createProxyMiddleware } = require("http-proxy-middleware");
require("dotenv").config();
const path = require("path")

const app = express();

// Configuration
const { PORT } = process.env;
const { HOST } = process.env;
const { API_BASE_URL } = process.env;

// Proxy
app.use(
  "/api",
  createProxyMiddleware({
    target: API_BASE_URL,
    changeOrigin: true,
  })
);

app.use(express.static(path.join(__dirname, 'build')))

app.listen(PORT, HOST, () => {
  console.log(`Starting Proxy at ${HOST}:${PORT}`);
});

app.get("*", async (req, res) => {
  res.sendFile(path.join(__dirname, 'build', 'index.html'))
})
);
```

Po potrebi, u *package.json*, treba dodati i sljedeće skripte (posebnu pažnju obratiti na *build* i *start-prod*):

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

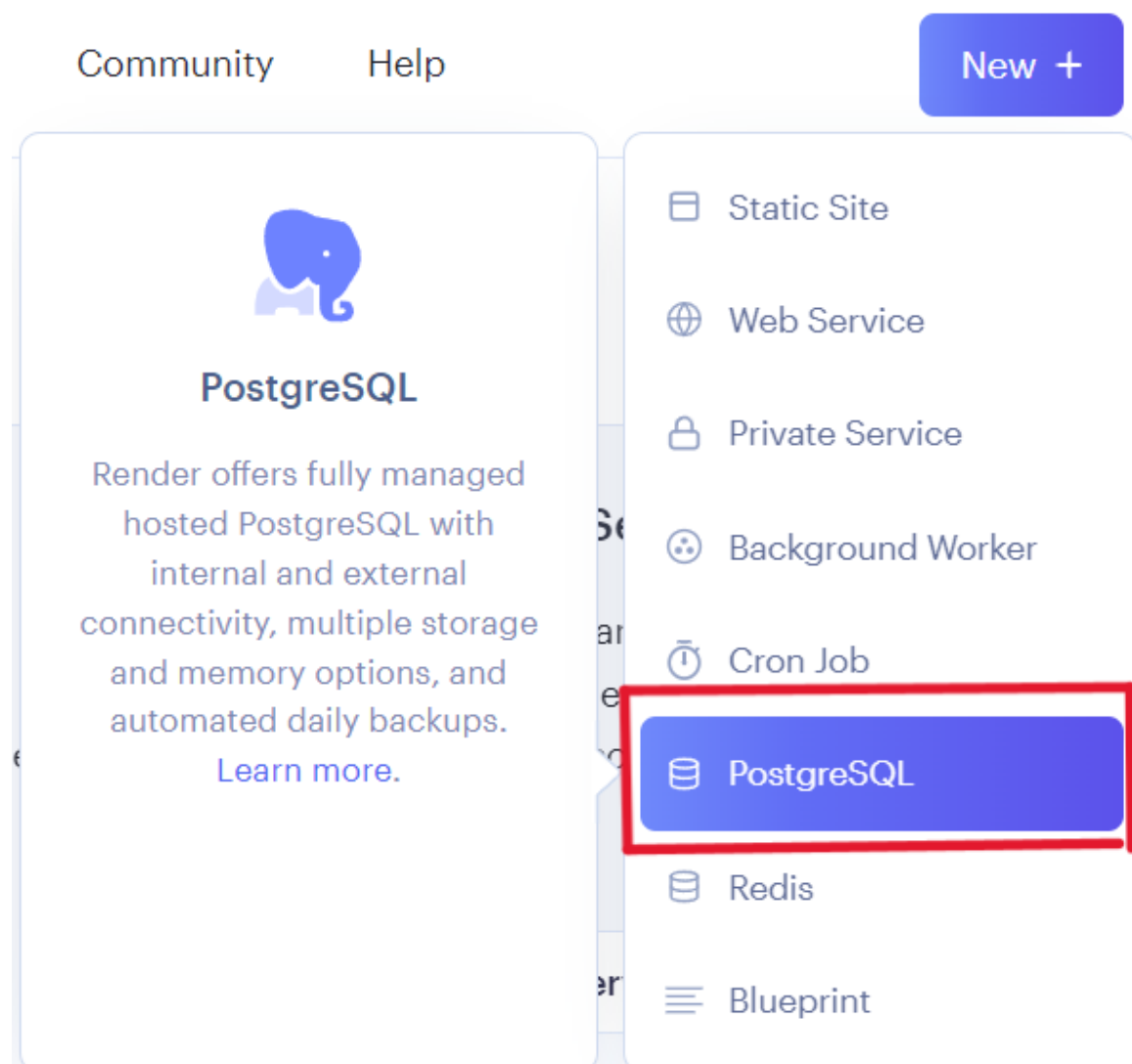
Priprema backenda za deploy

Po potrebi mogu se dodati environment varijable u konfiguraciju razvojnog okruženja (IDE). Zatim je potrebno dodati Dockerfile (u ovom slučaju za Maven, inače može i Gradle). Ako se mijenja lokacija Dockerfilea, paziti na putanje unutar COPY naredbi u Dockerfile skripti.

```
# Container za izgradnju aplikacije  
FROM openjdk:17-alpine AS builder  
COPY ../../.mvn .mvn  
COPY ../../mvnw .  
COPY ../../pom.xml .  
COPY ../../src src  
RUN chmod +x mvnw  
# Pokretanje builda  
RUN ./mvnw clean package  
# Stvaranje containera u kojem ce se vrtiti aplikacija  
FROM openjdk:17-alpine  
# Kopiranje izvrasnog JAR-a iz build containera u izvrsni container  
COPY --from=builder target/*.jar /app.jar  
# Izlaganje porta  
EXPOSE 8080  
  
# Naredba kojom se pokrece aplikacija  
ENTRYPOINT ["java", "-jar", "/app.jar"]
```

Kreiranje baze podataka

U Render dashboardu potrebno je otići na New - PostgreSQL



U sljedećem prozoru, postavljamo ime baze i opcionalno username za korisnika baze, dok se password automatski generira. Region postavljamo na *Frankfurt*.





New PostgreSQL

Name ⓘ	sinappsadb
Database ⓘ	sinappsa
User	turbobager
Region <small>The region where your Database runs.</small>	Frankfurt (EU Central)
PostgreSQL Version	15

Nakon toga kreiramo bazu klikom na *Create Database*.

Kreiranje backenda U Render dashboardu idemo na New - Web Service

Ako već niste, potrebno je povezati GitLab račun, nakon čega su za odabir dostupni svi projekti na koje imate prava pristupa. Potom stisnuti connect pored odgovarajućeg projekta.

 prog11 / frontend • 33 minutes ago	Connect
 prog11 / backend • an hour ago	Connect
 prog11 / meetings • a month ago	Connect
 prog11 / documentation • 2 months ago	Connect

Postaviti ime za servis (postat će dio web adrese), root directory postavljamo na *progi-be*, Environment na *Docker*, Region na *Frankfurt*.

Name A unique name for your web service.	<input type="text" value="sinappsaback"/>
Region The region where your web service runs.	<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="progi-be"/>
Environment The runtime environment for your web service.	<input type="text" value="Docker"/>

Na dnu proširujemo *Advanced*, dodajemo potrebne environment varijable (DB username, password, URL) i kopiramo vrijednosti iz postavki baze podataka na Renderu. Moramo ripaziti jer URL koji je prikazan na Renderu nije JDBC URL, a za ovaj primjer treba biti u formatu *jdbc:postgresql://hostname:port/database*.

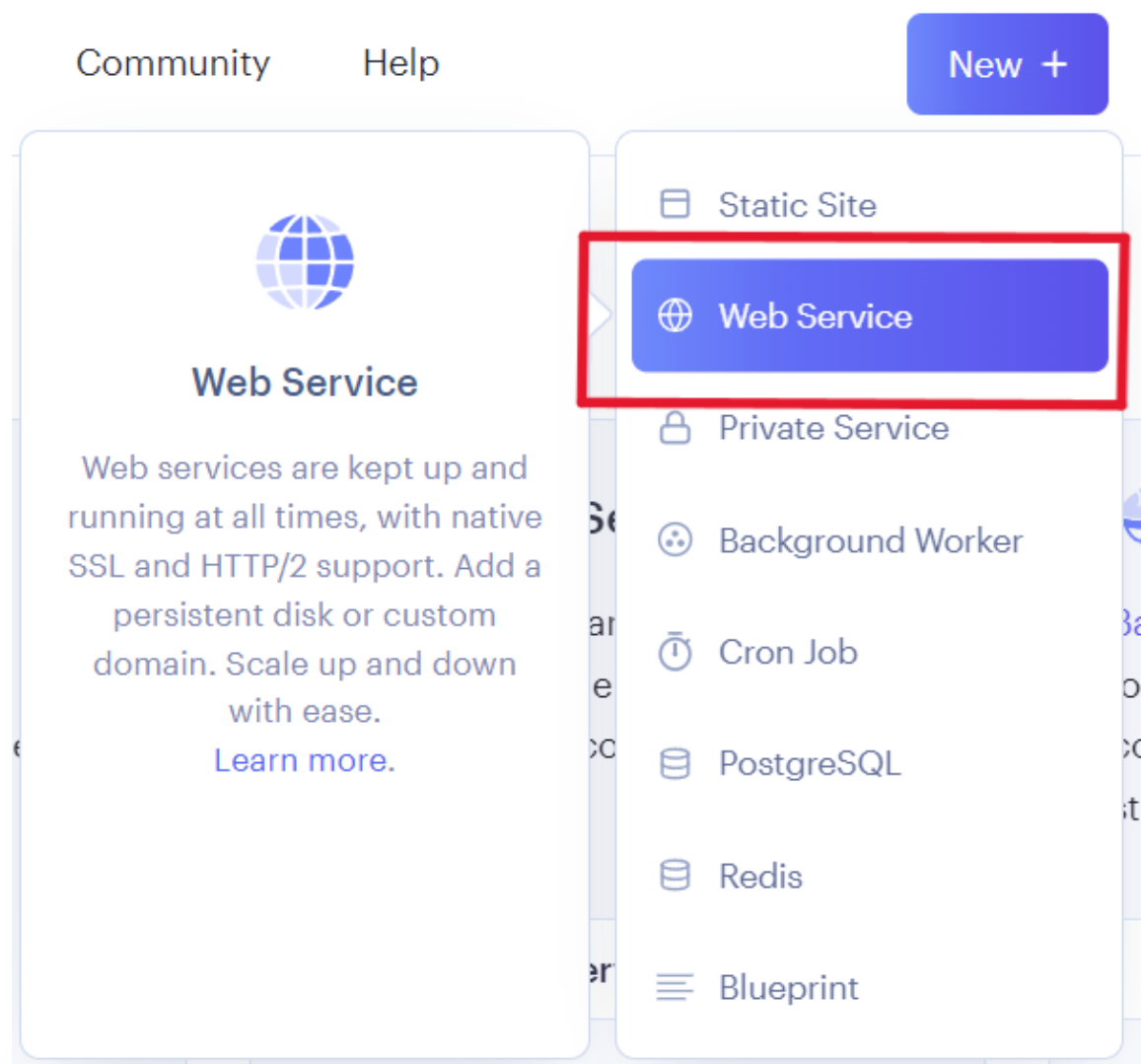
Nakon toga postavljamo */api/actuator/health* kao Health Check Path.

Putanju za Dockerfile postaviti na *./docker/maven/Dockerfile*.

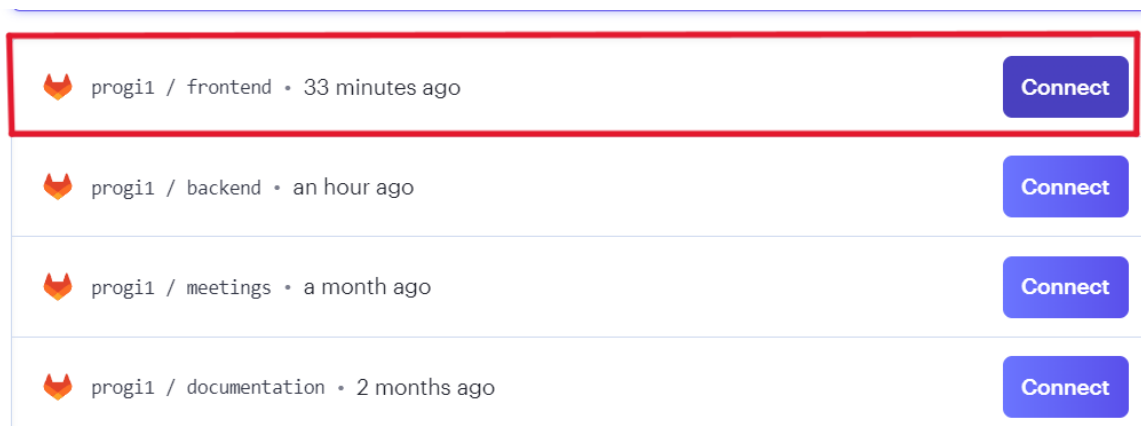
Stisnuti Create Web Service.

Kreiranje frontenda

U Render dashboardu potrebno je potrebno je otići na New - Web Service



U kreiranju backenda već smo spojili GitLab račun te su nam za odabir dostupni svi projekti na koje imamo prava pristupa. Potom je potrebno stisnuti connect pored odgovarajućeg projekta.



Postaviti ime za web servis (postat će dio web adrese). Root directory je potrebno postaviti na *progi-fe*. Environment se postavlja na *Node*, a Region na *Frankfurt*.

Name A unique name for your web service.	<input type="text" value="sinappsafont"/>
Region The region where your web service runs.	<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="progi-fe"/>
Environment The runtime environment for your web service.	<input type="text" value="Node"/>

Zatim je potrebno Build Command postaviti na *yarn build*, a Start Command na *yarn start-prod*.

Build Command This command runs in the root directory of your repository when a new version of your code is pushed, or when you deploy manually. It is typically a script that installs libraries, runs migrations, or compiles resources needed by your app.	<input type="text" value="progi-fe/ \$ yarn build"/>
Start Command This command runs in the root directory of your app and is responsible for starting its processes. It is typically used to start a webserver for your app. It can access environment variables defined by you in Render.	<input type="text" value="progi-fe/ \$ yarn start-prod"/>

Na dnu stranice potrebno je još proširiti *Advanced* i dodati potrebne environment varijable. `API_BASE_URL` postavlja se na adresu deployanog backenda aplikacije (adresa je dostupna na Render Dashboardu).

Advanced ✕

Use environment variables to store API keys and other configuration values and secrets. You can access them in your code like regular environment variables, for example with `os.getenv()` in Python or `process.env` in Node.

Add Environment Variable

You can store secret files (like `.env` or `.npmrc` files and private keys) in Render. These files can be accessed during builds and in your code just like regular files.

All secret files you create are available to read at the root of your repo (or Docker context). They are also available to load by absolute path at `/etc/secrets/<filename>`.

Add Secret File

Health Check Path
If you're running a server, enter the path where your server will always return a `200 OK` response. We use it to monitor your app and for zero downtime deploys.

/healthz

Nakon odrađenih svih prethodno navedenih koraka, potrebno je stisnuti *Create Web Service*.

6. Zaključak i budući rad

Zadatak naše grupe je bio da napravimo web forum za studente FER-a gdje će oni moći nuditi te tražiti pomoć. Naša aplikacija se zove „Sinappsa“ te je ostvarena kroz 17 tjedana rada. Podjela rada te cjelokupni projekt bio je odrađen u dvije faze.

Prva faza projekta uključivala je okupljanje tima za razvoj aplikacije, podjela poslova te određivanje krajnjih rokova za odrađivanje posla. Iznimna priprema te izvrsno postavljen kostur projekta omogućio nam je da rad ide bez prevelikih smetnji. UML dijagrami koje je obuhvaćala prva faza (obrasci uporabe, sekvencijski dijagrami, model baze podataka, dijagram razreda) projekta uistinu su pomogli programerima (dio tima koji se bavio frontend i backend dijelovima). Za svaku nedoumicu koju su imali oko implementacije i izgleda aplikacije konzultirali su se sa dokumentacijom projekta te im je uštedjelo dosta vremena.

Druga faza projekta je bila kraća od prve te je zahtijevala puno više rada nego prva. Radi manjka iskustva pojedinih članova tima bilo je doista zahtjevno dovršiti aplikaciju kako je bila zamišljena. Članovi tima su morali konstantno učiti nove tehnologije, alate, metode kako bi riješili problem koji se nalazio ispred njih te su mnoštvo toga naučili. Što se tiče dokumentiranja projekta, u drugoj fazi bilo je potrebno nadograditi određene dijelove iz prve faze. Također je bilo potrebno nadodati još 4 vrste dijagrama te dovršiti dokumentiranje projekta kako bi budućim korisnicima aplikacije bilo lakše koristiti našu aplikaciju.

Komunikacija među članovima tima realizirana je putem Whatsapp grupe te razgovorima uživo na prostorima fakulteta. Aplikaciju je moguće proširiti razvijanjem ju u mobilnu aplikaciju čime bi se i proširila upotreba te dostupnost aplikacije.

Kroz sudjelovanje na ovom projektu tim je uz funkcionalnu aplikaciju pokazao i timski rad. Uz sve znanje što smo stekli u našem području, najviše smo naučili kako je to zapravo raditi u timu. Naučili smo koliko nam organizacija te dobro postavljeni temelji znače na nekom ovakvom projektu. Zadovoljni smo s onim što imamo za predstaviti, iako uz neiskustvo smatramo da stojimo iza kvalitetnog rada na kojeg svi možemo biti ponosni.

Popis literature

Kontinuirano osvježavanje

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

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

Indeks slika i dijagrama

2.1	Primjer ocjenjivanja u Bolt aplikaciji	6
3.1	Dijagram obrasca uporabe : Korisnik - Klijent	17
3.2	Dijagram obrasca uporabe: Moderator - Klijent	18
3.3	Sekvencijski dijagram za UC3	19
3.4	Sekvencijski dijagram za UC4	20
3.5	Sekvencijski dijagram za UC8	23
4.1	Controller-Service-Repository arhitektura	26
4.2	E-R dijagram baze podataka	31
4.3	Dijagram razreda - dio Entities	33
4.4	Dijagram razreda - dio Controllers	34
4.5	Dijagram razreda - dio Service	35
4.6	Dijagram razreda - dio Repository	36
4.7	Dijagram razreda - dio Data Transfer Objects	37
4.8	Dijagram razreda - dio Security	37
4.9	Dijagram stanja klijenta	38
4.10	Dijagram aktivnosti - Stvaranje oglasa	40
4.11	Dijagram komponenti	42
5.1	Rezultati ispitivanja komponenti	46
5.2	Rezultati ispitivanja sustava	52
5.3	Dijagram razmještaja	53