

Programsko inženjerstvo

Ak. god. 2023./2024.

CookBooked

Dokumentacija, Rev. 2

Grupa: *ErrorMasters*

Voditelj: *Matija Alojz Stuhne*

Datum predaje: 16. 11. 2023.

Nastavnik: *Nikolina Frid*

Sadržaj

1 Dnevnik promjena dokumentacije	3
2 Opis projektnog zadatka	8
3 Specifikacija programske potpore	11
3.1 Funkcionalni zahtjevi	11
3.1.1 Obrasci uporabe	13
3.1.2 Sekvencijski dijagrami	25
3.2 Ostali zahtjevi	29
4 Arhitektura i dizajn sustava	30
4.1 Baza podataka	32
4.1.1 Opis tablica	33
4.1.2 Dijagram baze podataka	39
4.2 Dijagram razreda	40
4.3 Dijagrami nakon implementacije programskih rješenja	44
4.4 Dijagram stanja	45
4.5 Dijagram aktivnosti	46
5 Implementacija i korisničko sučelje	50
5.1 Korištene tehnologije i alati	50
5.2 Ispitivanje programskog rješenja	51
5.2.1 Ispitivanje komponenti	51
5.2.2 Ispitivanje sustava	55
5.3 Dijagram razmještaja	56
5.4 Upute za puštanje u pogon	57
6 Zaključak i budući rad	63
Popis literature	65
Indeks slika i dijagonama	67

Dodatak: Prikaz aktivnosti grupe

68

1. Dnevnik promjena dokumentacije

Rev.	Opis promjene/dodataka	Autori	Datum
0.1	Napravljen predložak.	Matija Alojz Stuhne	23.10.2023.
0.1.1	Osvježen predložak s podatcima grupe.	Matija Alojz Stuhne	30.10.2023.
0.2	Dodan opis projektnog zadatka.	Matija Alojz Stuhne	30.10.2023.
0.3	Dodani dionici, aktori i njihovi funkcionalni zahtjevi	Matija Alojz Stuhne	31.10.2023.
0.4	Dodani obrasci uporabe	Matija Alojz Stuhne	31.10.2023.
0.4.1	Manje promjene dokumentacije	Matija Alojz Stuhne	01.11.2023.
0.5	Dodani dijagrami obrazaca uporabe	Marko Dananić	03.11.2023.
0.5.1	Dodani dijagrami obrazaca uporabe (u dokument) Manje promjene dokumentacije	Matija Alojz Stuhne	03.11.2023.

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

Rev.	Opis promjene/dodataka	Autori	Datum
0.5.2	Manje promjene dijagrama obrazaca uporabe	Marko Dananić, Matija Alojz Stuhne	04.11.2023
0.6	Dodani sekvencijski dijagrami	Matija Alojz Stuhne	04.11.2023.
0.6.1	Dodani opisi sekvencijskih dijagrama <i>Pushed by M. A. Stuhne</i>	Marko Dananić	07.11.2023.
0.6.2	Manje promjene sekvencijskih dijagrama	Matija Alojz Stuhne	15.11.2023.
0.6.3	Manje promjene dokumentacije - sastanci, dnevnik promjena	Matija Alojz Stuhne	15.11.2023.
0.7	Dodani prikazi relacija baze podataka	Matija Alojz Stuhne	15.11.2023.
0.7.1	Dodani opisi prikaza relacija baze podataka	Matija Alojz Stuhne	16.11.2023.
0.8.	Dodani dijagrami razreda <i>Pushed by M. A. Stuhne</i>	Marko Dananić, Petric Habjanec	16.11.2023.

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

Rev.	Opis promjene/dodataka	Autori	Datum
0.8.1	Dodani ostali zahtjevi sustava	Matija Alojz Stuhne	16.11.2023.
0.8.2	Manje promjene dijagrama obrazaca uporabe <i>Pushed by M. A. Stuhne</i>	Marko Dananić	16.11.2023.
0.8.3	Dodan uvodni tekst za dijagrame razreda	Matija Alojz Stuhne	16.11.2023.
1.0.0	Korigiranje teksta i provjera dokumentacije	Matija Alojz Stuhne	16.11.2023.
1.1.0	Započeta razrada 5. poglavlja	Matija Alojz Stuhne	2.1.2024.
1.2.0	Dodan dijagram stanja	Matija Alojz Stuhne	6.1.2024.
1.2.1	Dodan dijagram aktivnosti	Matija Alojz Stuhne	6.1.2024.
1.2.2	Dodan dijagram komponenti <i>Pushed by M. A. Stuhne</i>	Marko Dananić	10.1.2024.
1.2.3	Dodan dijagram razmještaja <i>Pushed by M. A. Stuhne</i>	Marko Dananić	10.1.2024.
1.3.0	Dodani JUnit testovi	Matija Alojz Stuhne	18.1.2024.

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

Rev.	Opis promjene/dodataka	Autori	Datum
1.3.1	Dodani rezultati JUnit testova	Matija Alojz Stuhne	18.1.2024.
1.4.0	Dodan zaključak	Matija Alojz Stuhne	18.1.2024.
1.5.0	Dodane upute za puštanje u pogon <i>Pushed by M. A. Stuhne</i>	Dario Huić	19.1.2024.
1.5.1	Ispravci dokumentacije	Matija Alojz Stuhne	19.1.2024.
1.5.2	Dodana tablica aktivnosti	Matija Alojz Stuhne	19.1.2024.
1.6.0	Dodani grafovi git aktivnosti	Matija Alojz Stuhne	19.1.2024.
1.6.1	Manje promjene dokumentacije - Selenium ERROR 404	Matija Alojz Stuhne	19.1.2024.
1.6.2	Dodan prikaz komponenti sustava nakon završetka razvoja	Matija Alojz Stuhne	19.1.2024.
1.6.3	Promjene grafova git aktivnosti	Matija Alojz Stuhne	19.1.2024.

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

Rev.	Opis promjene/dodataka	Autori	Datum
1.6.4	Dodatne promjene grafova git aktivnosti	Matija Alojz Stuhne	19.1.2024.
2.0.0	Korigiranje teksta i provjera dokumentacije	Matija Alojz Stuhne	19.1.2024.

2. Opis projektnog zadatka

Cilj ovog projekta je razviti programsku podršku za stvaranje web aplikacije "CookBooked" koja će omogućiti korisnicima razmjenu recepata za kuhanje i pečenje kolača te povezivanje i komunikaciju s autorima recepata. Platforma će ponuditi različit spektar mogućnosti ovisno o tome je li korisnik registriran ili nije.

Prilikom pokretanja sustava prikazuje se izbornik recepata po kategorijama (prigoda, namirnice, zemlja podrijetla recepta).

Neregistrirani korisnici mogu **samo pregledavati** recepte temeljem kategorija. Neregistrirani korisnici vide sve detalje recepata (naslov, sastojci, koraci pripreme, posebne oznake...). Kako bi korisniku bilo omogućeno vršiti dodatne radnje (objava recepata, komentiranje recepata, spremanje recepata...), potrebno se prijaviti sa postojećim korisničkim računom ili registracijom kreirati novi korisnički račun.

Za kreiranje novog korisničkog računa potrebni su sljedeći podatci:

- *korisničko ime*
- *lozinka*
- *ime*
- *prezime*
- *broj mobitela*
- *email adresa*

Registracijom u sustav korisniku se dodjeljuju prava klijenta, a naknadno mu se mogu dodijeliti prava vlasnika portala ili administratora. Registrirani korisnik može objavljivati recepte za kuhanje i pečenje, komentirati na ostale recepte itd.

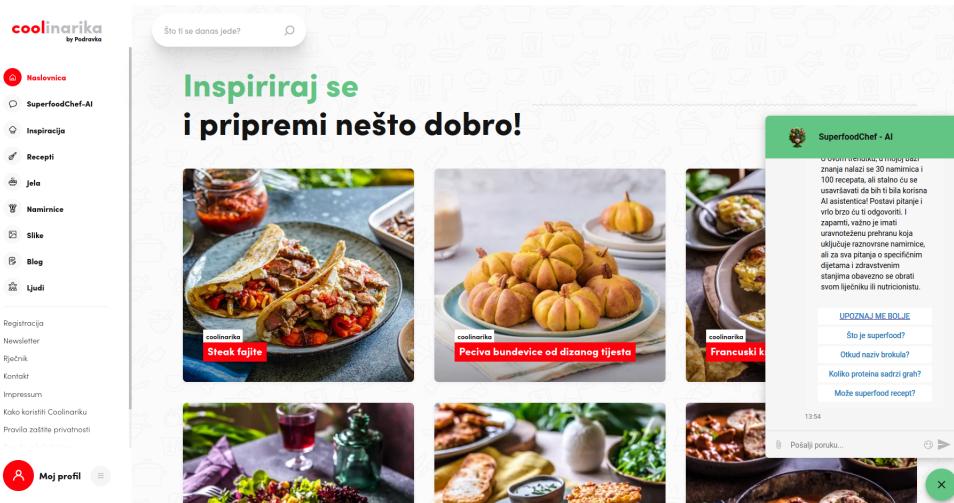
Klijent objavom recepta postaje njegov autor. Autori recepata mogu komunicirati s ostalim korisnicima vezano za svoje recepte (termin komunikacija = chat, videopozivi itd.). Klijent ima mogućnost označavanja, komentiranja i spremanja recepata za buduću referencu. Također, kao dodatna mogućnost, pruža se i opcija praćenja omiljenih autora (obavijesti). Klijent ima dvije vrste profila, javni i privatni. Na javnom su prikazani recepti, pratitelji i autori koje klijent prati; dok su na privatnom prikazane osobne informacije kojima klijent može upravljati (korisničko ime, lozinka, e-mail adresa, ime, prezime...).

Vlasnik portala ima dodatne mogućnosti brisanja neželjenih recepata i/ili komentara ispod njih te mogućnost brisanja i/ili dodavanja novih kategorija recepata.

Administrator sustava ima najveće ovlasti. On ima pristup bazi s popisom registriranih korisnika te ima mogućnost brisanja ili promijene korisničkih podataka. Također, ima svu funkcionalnost vlasnika web portala.

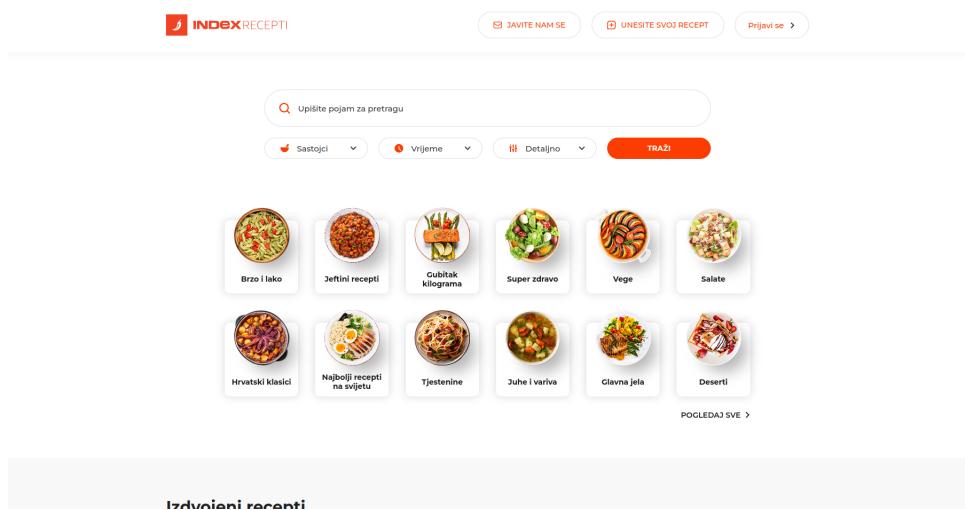
Postojeća slična rješenja:

<https://www.coolinarika.com/>



Slika 2.1: Izgled coolinarika web portala

<https://recepti.index.hr/>



Slika 2.2: Izgled index-recepti web portala

Postojeća rješenja baziraju se na proširenoj funkcionalnosti (osim kolača nude i recepte ostalih jela). Coolinarika nudi i dodatnu funkcionalnost razgovora sa AI ChatBot-om koji pomaže ljudima oko generalnog pojma prehrane. Također, Coolinarika ima i blog te pregled svojstava namirnica koje se koriste u receptima.

Mogućnost nadogradnje web portala CookBooked:

Web portal CookBooked primarno je zamišljen kao pomoćnik u kuhanju i pečenju kolača. Kao takav, usmjeren je na korisnike čiji je primarni interes područje slastica.

Svaka nadogradnja postojećeg portala morala bi biti dobro iskommunicirana sa postojećim klijentima.

Mogućnosti nadogradnje:

- *dodavanje recepata koji nisu vezani za kolače*
- *prijenos uživo pripremanja recepata*
- *dodavanje oglasnik poslova vezanih uz kulinarstvo/slastičarstvo*
- *dodavanje premium sadržaja dostupnog samo plaćenim klijentima*

Financiranje web portala CookBooked:

Održavanje i nadogradnja web portala CookBooked planiraju se financirati iz sljedećih izvora:

- *plaćeni oglasi na web portalu*
- *prodaja premium korisničkih mogućnosti (u planu)*
- *donacije*

3. Specifikacija programske potpore

3.1 Funkcionalni zahtjevi

Dionici:

1. Vlasnik web portala (naručitelj)
2. Klijenti web portala
3. Administrator
4. Razvojni tim

Aktori i njihovi funkcionalni zahtjevi:

1. Neregistrirani/neprijavljeni korisnik (inicijator) može:
 - (a) pregledavati recepte temeljem kategorija
 - (b) se registrirati u sustav, stvoriti korisnički račun za koji su mu potrebni korisničko ime, lozinka, ime, prezime, broj mobitela, adresa elektroničke pošte
2. Klijent (registriran i prijavljen korisnik) (inicijator) može:
 - (a) pregledavati recepte temeljem kategorija
 - (b) objavljivati vlastite recepte
 - i. komunicirati s ostalim klijentima
 - (c) označavati tuđe recepte
 - (d) komentirati tuđe recepte
 - (e) spremati recepte za buduću referencu
 - (f) pratiti ostale autore
 - (g) pregledavati i mijenjati osobne podatke
 - (h) izbrisati svoj korisnički račun
3. Vlasnik (inicijator) može:
 - (a) ***sva funkcionalnost Klijenta bez brisanje svog računa***
 - (b) brisati neželjene recepte

- (c) brisati neželjene komentare
- (d) brisati/dodavati kategorije

4. Administrator (inicijator) može:

- (a) ***sva funkcionalnost Vlasnika bez funkcionalost klijenta***
- (b) vidjeti popis svih registriranih korisnika i njihovih osobnih podataka
 - i. mijenjati korisničke podatke
- (c) brisati korisnike i mijenjati im razinu pristupa aplikaciji (klijent, vlasnik)

5. Baza podataka (sudionik):

- (a) pohranjuje sve podatke o korisnicima i njihovim ovlastima
- (b) pohranjuje sve podatke o receptima i kategorijama

3.1.1 Obrasci uporabe

Opis obrazaca uporabe

UC1 - Pregled recepata

- **Glavni sudionik:** Korisnik, klijent
- **Cilj:** Pregledati recepte ovisno o odabranim kategorijama
- **Sudionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
 1. Kategorije recepata su prikazane prilikom učitavanja aplikacije
 2. Korisnik/Klijent odabire željene kategorije
 3. Prikazuju se recepti iz željenih kategorija

UC2 - Registracija

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

UC3 - Prijava u sustav

- **Glavni sudionik:** Klijent
- **Cilj:** Dobiti pristup korisničkom sučelju
- **Sudionici:** Baza podataka
- **Preduvjet:** UC2 - Registracija

- **Opis osnovnog tijeka:**
 1. Unos korisničkog imena i lozinke
 2. Potvrda o ispravnosti unesenih podataka
 3. Pristup korisničkim funkcijama
- **Opis mogućih odstupanja:**
 - 2.a Neispravno korisničko ime/lozinka
 1. Sustav obavještava korisnika o neuspjelom upisu i vraća ga na stranicu za prijavu

UC4 - Pregled osobnih podataka

- **Glavni sudionik:** Klijent
- **Cilj:** Pregledati osobne podatke
- **Sudionici:** Baza podataka
- **Preduvjet:** UC3 - Prijava u sustav
- **Opis osnovnog tijeka:**
 1. Klijent odabire opciju osobni podatci
 2. Aplikacija prikazuje osobne podatke klijenta

UC5 - Promjena osobnih podataka

- **Glavni sudionik:** Klijent
- **Cilj:** Promijenti osobne podatke
- **Sudionici:** Baza podataka
- **Preduvjet:** UC4 - Pregled osobnih podataka
- **Opis osnovnog tijeka:**
 1. Klijent odabire opciju za promjenu podataka
 2. Klijent mijenja svoje osobne podatke
 3. Klijent spremi promjene
 4. Podatci prolaze validaciju
 5. Baza podataka se ažurira
- **Opis mogućih odstupanja:**
 - 2.a Unos neispravnih podataka
 1. Sustav obavještava korisnika o neuspjelom upisu i vraća ga na opciju za promjenu podataka

UC6 - Brisanje korisničkog računa

- **Glavni sudionik:** Klijent

- **Cilj:** Izbrisati korisnički račun
- **Sudionici:** Baza podataka
- **Preduvjet:** UC4 - Pregled osobnih podataka
- **Opis osnovnog tijeka:**
 1. Klijent odabire opciju brisanja korisničkog računa
 2. Klijent briše račun
 3. Posljedično se brišu i svi recepti te komentari klijenta
 4. Baza podataka se ažurira
 5. Otvara se stranica za registraciju

UC7 - Objavljivanje recepta

- **Glavni sudionik:** Klijent
- **Cilj:** Objaviti recept na web portalu
- **Sudionici:** Baza podataka
- **Preduvjet:** UC3 - Prijava u sustav
- **Opis osnovnog tijeka:**
 1. Klijent odabire opciju objavi recept
 2. Klijent unosi potrebne podatke o receptu
 3. Klijent objavljuje recept
 4. Klijent prima obavijest o uspješnoj objavi recepta
 5. Šalje se obavijest svim pratiteljima klijenta/autora
 6. Baza podataka se ažurira
 7. Otvara se stranica sa receptima korisnika
- **Opis mogućih odstupanja:**
 - 2.a Unos neispravnih podataka
 1. Sustav obaveštava korisnika o neuspjelom upisu i vraća ga na opciju za objavu recepta

UC8 - Uređivanje recepta

- **Glavni sudionik:** Klijent
- **Cilj:** Urediti recept na web portalu
- **Sudionici:** Baza podataka
- **Preduvjet:** UC3 - Prijava u sustav, UC7 - Objavljivanje recepta
- **Opis osnovnog tijeka:**
 1. Klijent otvara stranicu sa svojim receptima
 2. Klijent odabire opciju uređivanja recepta

3. Klijent mijenja određene podatke o receptu
 4. Klijent sprema promijene
 5. Klijent prima obavijest o uspješnoj promjeni recepta
 6. Baza podataka se ažurira
 7. Otvara se stranica sa receptima korisnika
- **Opis mogućih odstupanja:**
 - 3.a Unos neispravnih podataka
 1. Sustav obavještava korisnika o neuspjelom upisu i vraća ga na opciju za objavu recepta

UC9 - Brisanje recepta

- **Glavni sudionik:** Klijent
- **Cilj:** Izbrisati recept sa web portala
- **Sudionici:** Baza podataka
- **Preduvjet:** UC3 - Prijava u sustav, UC7 - Objavljivanje recepta
- **Opis osnovnog tijeka:**
 1. Klijent otvara stranicu sa svojim receptima
 2. Klijent odabire opciju brisanja recepta
 3. Klijent prima obavijest o uspješnom brisanju recepta
 4. Baza podataka se ažurira
 5. Klijenti koji su recept imali spremišten, više ga nemaju
 6. Otvara se stranica sa receptima korisnika

UC10 - Slanje poruke

- **Glavni sudionik:** Klijent
- **Cilj:** Poslati poruku drugom klijentu (autoru recepta)
- **Sudionici:** Baza podataka
- **Preduvjet:** UC3 - Prijava u sustav
- **Opis osnovnog tijeka:**
 1. Klijent odabire opciju slanja poruke drugom klijentu (autoru recepta)
 2. Klijent piše odgovarajuću poruku
 3. Klijent šalje poruku
 4. Klijent prima obavijest o uspješnom slanju poruke
 5. Baza podataka se ažurira

UC11 - Primanje poruke

- **Glavni sudionik:** Klijent
- **Cilj:** Primiti poruku od drugog klijenta
- **Sudionici:** Baza podataka
- **Preduvjet:** UC3 - Prijava u sustav
- **Opis osnovnog tijeka:**
 1. Klijent se prijavljuje u sustav
 2. Klijent odabire opciju provjere obavijesti
 3. Klijent čita poruku

UC12 - Komentiranje recepta

- **Glavni sudionik:** Klijent
- **Cilj:** Objaviti komentar na receptu
- **Sudionici:** Baza podataka
- **Preduvjet:** UC3 - Prijava u sustav
- **Opis osnovnog tijeka:**
 1. Klijent odabire recept koji želi komentirati
 2. Klijent piše željeni komentar
 3. Klijent objavljuje komentar
 4. Baza podataka se ažurira
- **Opis mogućih odstupanja:**
 - 3.a Unos neispravnih podataka (dopušteni znakovi)
 1. Sustav obaveštava korisnika o neuspjeloj objavi komentara

UC13 - Spremanje recepta

- **Glavni sudionik:** Klijent
- **Cilj:** Spremiti recept za kasniju referencu
- **Sudionici:** Baza podataka
- **Preduvjet:** UC3 - Prijava u sustav
- **Opis osnovnog tijeka:**
 1. Klijent odabire recept koji želi spremiti
 2. Klijent sprema recept
 3. Baza podataka se ažurira

UC14 - Uklanjanje recepta

- **Glavni sudionik:** Klijent
- **Cilj:** Maknuti recept iz spremljenih recepata

- **Sudionici:** Baza podataka
- **Preduvjet:** UC3 - Prijava u sustav, UC13 - Spremanje recepta
- **Opis osnovnog tijeka:**
 1. Klijent odabire recept koji želi maknuti iz spremnih
 2. Klijent miče recept
 3. Baza podataka se ažurira

UC15 - Označavanje recepta

- **Glavni sudionik:** Klijent
- **Cilj:** Označiti recept za kasniju referencu
- **Sudionici:** Baza podataka
- **Preduvjet:** UC3 - Prijava u sustav
- **Opis osnovnog tijeka:**
 1. Klijent odabire recept koji želi označiti kao jedan od svojih omiljenih
 2. Klijent označava recept
 3. Baza podataka se ažurira

UC16 - Odznačavanje recepta

- **Glavni sudionik:** Klijent
- **Cilj:** Maknuti oznaku sa recepta
- **Sudionici:** Baza podataka
- **Preduvjet:** UC3 - Prijava u sustav, UC14 - Označavanje recepta
- **Opis osnovnog tijeka:**
 1. Klijent odabire recept kojem želi maknuti oznaku
 2. Klijent miče recept
 3. Baza podataka se ažurira

UC17 - Praćenje autora

- **Glavni sudionik:** Klijent
- **Cilj:** Označiti autora od kojeg želi primati obavijesti
- **Sudionici:** Baza podataka
- **Preduvjet:** UC3 - Prijava u sustav
- **Opis osnovnog tijeka:**
 1. Klijent odabire autora kojeg želi zapratiti, tj. od njega primati obavijesti
 2. Klijent zaprati autora
 3. Baza podataka se ažurira

UC18 - Prestanak praćenja autora

- **Glavni sudsionik:** Klijent
- **Cilj:** Prestati pratiti autora, tj. od njega primati obavijesti
- **Sudsionici:** Baza podataka
- **Preduvjet:** UC3 - Prijava u sustav, UC17 - Praćenje autora
- **Opis osnovnog tijeka:**
 1. Klijent odabire autora kojeg želi prestati pratiti.
 2. Klijent prestaje pratiti autora
 3. Baza podataka se ažurira

UC19 - Slanje obavijesti

- **Glavni sudsionik:** Klijent
- **Cilj:** Poslati obavijest svojim pratiteljima
- **Sudsionici:** Baza podataka
- **Preduvjet:** UC3 - Prijava u sustav, UC7 - Objavljivanje recepta
- **Opis osnovnog tijeka:**
 1. Klijent objavljuje novi recept
 2. Njegovim pratiteljima stiže obavijest
 3. Baza podataka se ažurira

UC20 - Primanje obavijesti

- **Glavni sudsionik:** Klijent
- **Cilj:** Primiti obavijest praćenog autora
- **Sudsionici:** Baza podataka
- **Preduvjet:** UC3 - Prijava u sustav, UC15 - Praćenje autora
- **Opis osnovnog tijeka:**
 1. Klijent odabire opciju provjere obavijesti
 2. Klijent čita obavijest

UC21 - Dodavanje kategorije

- **Glavni sudsionik:** Vlasnik, Administrator
- **Cilj:** Dodati novu kategoriju recepata
- **Sudsionici:** Baza podataka
- **Preduvjet:** UC3 - Prijava u sustav (kao jedan od sudsionika)
- **Opis osnovnog tijeka:**

1. Vlasnik/Administrator odabire opciju dodavanja kategorija
 2. Vlasnik/Administrator dodaje novu kategoriju
 3. Baza podataka se ažurira
- 3.a Unos postojeće kategorije
 1. Sustav obavještava korisnika o neuspjelom dodavanju kategorije, vraća ga na stranicu s kategorijama
 - 3.b Unos neispravnih podataka (dopušteni znakovi)
 1. Sustav obavještava korisnika o neuspjelom dodavanju kategorije, vraća ga na stranicu s kategorijama

UC22 - Brisanje kategorije

- **Glavni sudionik:** Vlasnik, Administrator
- **Cilj:** Izbrisati kategoriju recepta
- **Sudionici:** Baza podataka
- **Preduvjet:** UC3 - Prijava u sustav (kao jedan od sudionika)
- **Opis osnovnog tijeka:**
 1. Vlasnik/Administrator odabire opciju brisanja kategorija
 2. Vlasnik/Administrator briše kategoriju
 3. Baza podataka se ažurira

UC23 - Brisanje komentara

- **Glavni sudionik:** Vlasnik, Administrator
- **Cilj:** Izbrisati komentar na recept
- **Sudionici:** Baza podataka
- **Preduvjet:** UC3 - Prijava u sustav (kao jedan od sudionika)
- **Opis osnovnog tijeka:**
 1. Vlasnik/Administrator odabire komentar koji želi obrisati
 2. Vlasnik/Administrator briše komentar
 3. Baza podataka se ažurira

UC24 - Pregled korisnika

- **Glavni sudionik:** Administrator
- **Cilj:** Pregledati popis klijenata
- **Sudionici:** Baza podataka
- **Preduvjet:** UC3 - Prijava u sustav (kao jedan od sudionika)
- **Opis osnovnog tijeka:**

1. Administrator se prijavljuje u sustav
2. Administrator odabire opciju pregleda korisnika
3. Prikaže se lista svih registriranih korisnika

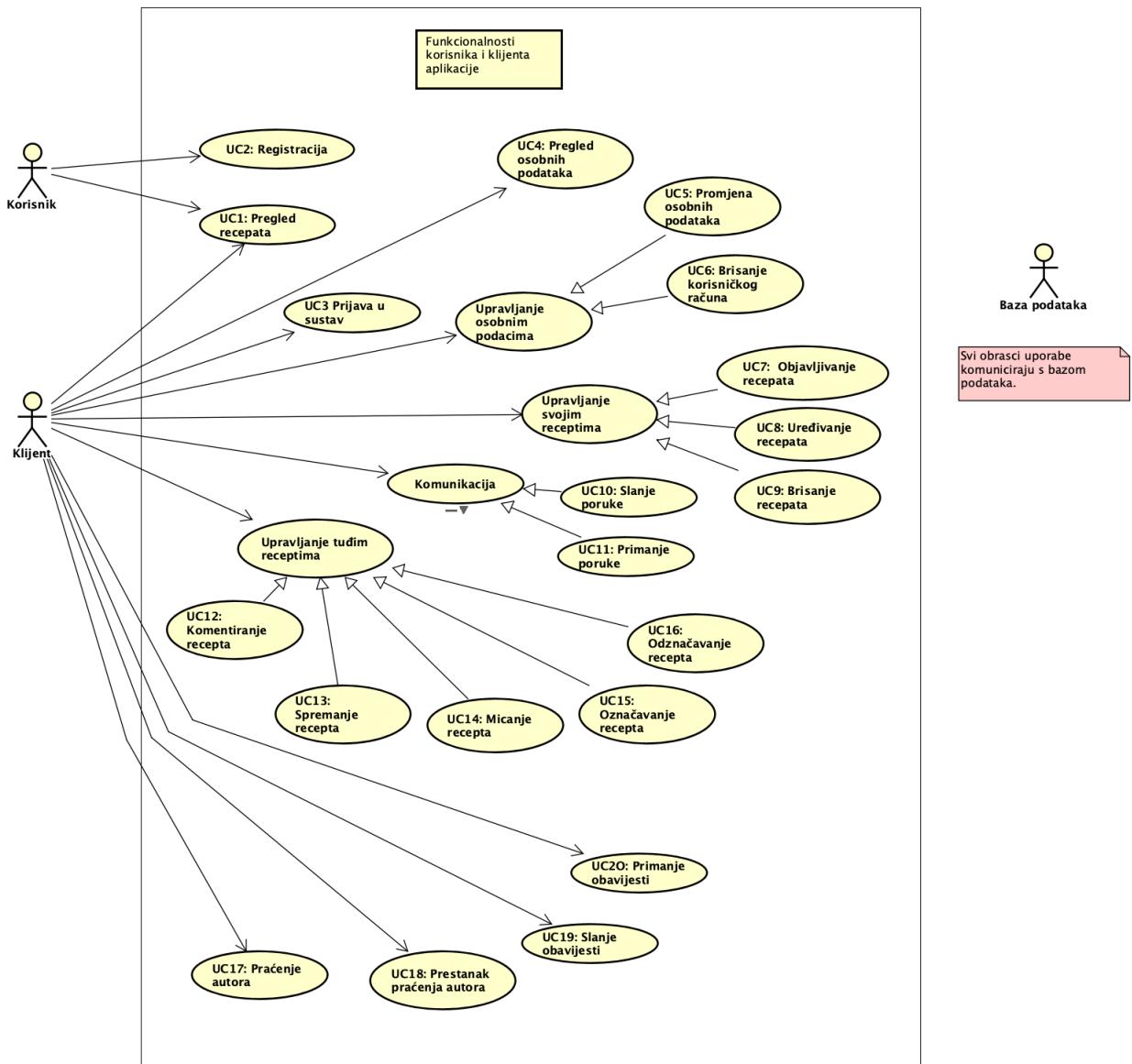
UC25 - Brisanje korisnika

- **Glavni sudionik:** Administrator
- **Cilj:** Obrisati korisnički račun klijenta
- **Sudionici:** Baza podataka
- **Preduvjet:** UC3 - Prijava u sustav (kao jedan od sudionika), UC24 - Pregled korisnika
- **Opis osnovnog tijeka:**
 1. Administrator odabire željenog korisnika
 2. Administrator odabire opciju brisanja korisnika
 3. Korisnički račun klijenta se obriše
 4. Posljedično se brišu i svi recepti te komentari klijenta
 5. Baza podataka se ažurira

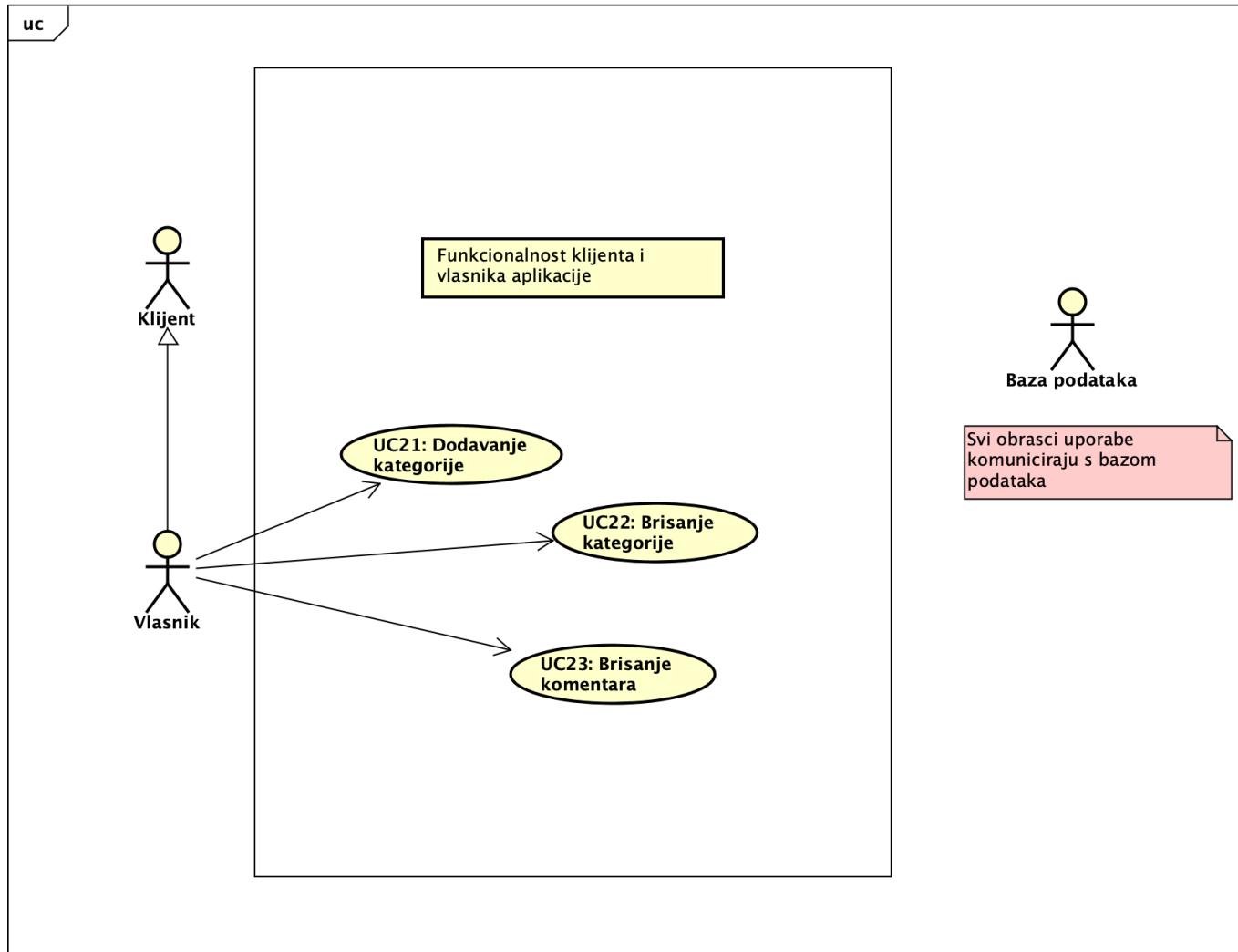
UC26 - Promjena prava korisnika

- **Glavni sudionik:** Administrator
- **Cilj:** Promijeniti razinu pristupa korisnika
- **Sudionici:** Baza podataka
- **Preduvjet:** UC3 - Prijava u sustav (kao jedan od sudionika), UC24 - Pregled korisnika
- **Opis osnovnog tijeka:**
 1. Administrator se prijavljuje u sustav
 2. Administrator odabire željenog korisnika
 3. Administrator odabire opciju promijene prava korisnika
 4. Administrator mijenja razinu pristupa korisnika
 5. Baza podataka se ažurira

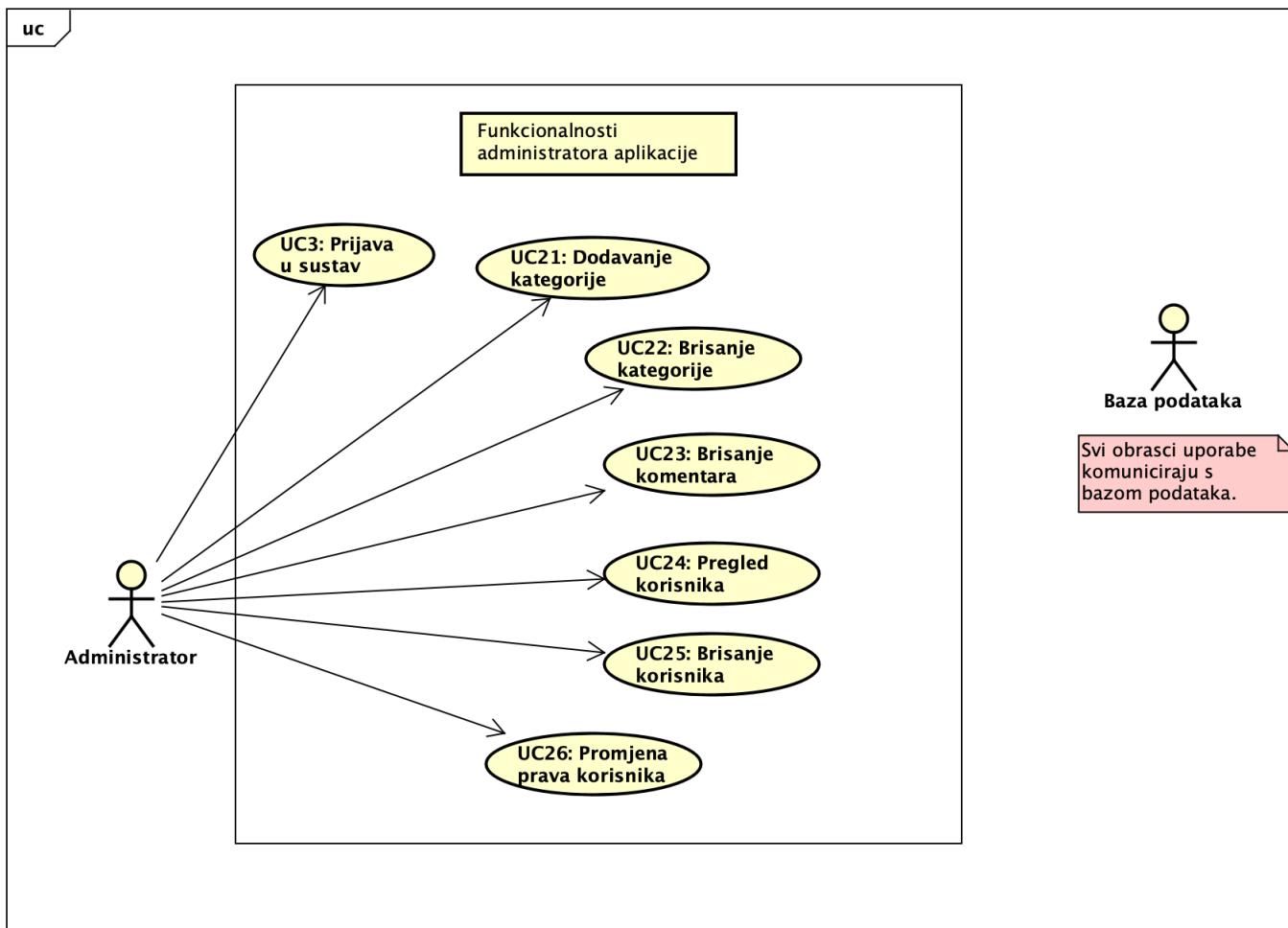
Dijagrami obrazaca uporabe



Slika 3.1: Dijagram obrazca uporabe, funkcionalnost korisnika i klijenta



Slika 3.2: Dijagram obrasca uporabe, funkcionalnost vlasnika

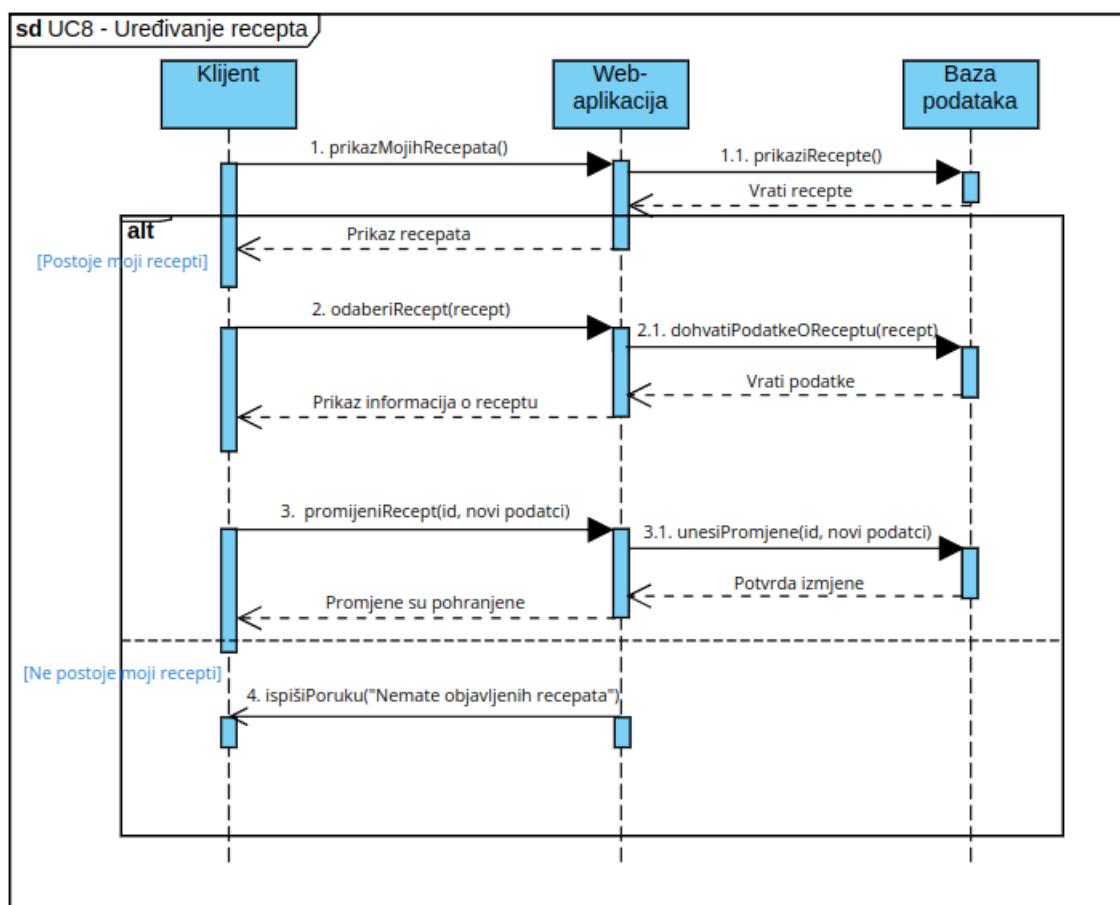


Slika 3.3: Dijagram obrasca uporabe, funkcionalnost administratora

3.1.2 Sekvencijski dijagrami

Obrazac uporabe UC8 – Uređivanje recepta

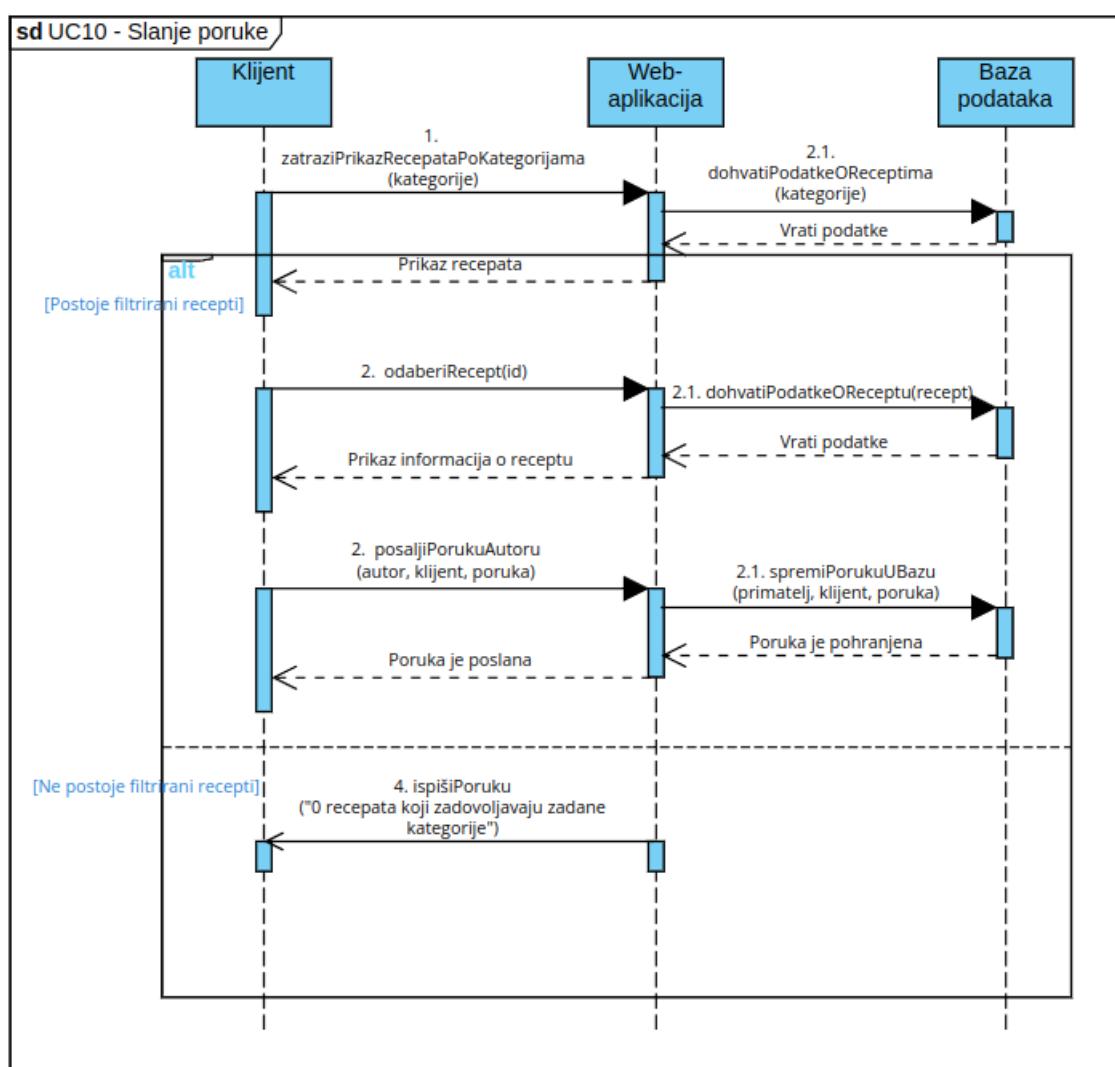
Klijent šalje zahtjev za prikaz svojih recepata kako bi mogao odabrati recept koji želi urediti. Poslužitelj dohvaća dostupne recepte i prikazuje ih ukoliko postoje. U slučaju da ne postoji niti jedan recept poslužitelj vraća odgovarajuću poruku. Odbijom recepta poslužitelj iz baze podataka dohvaća podatke o receptu i prikazuje ih korisniku. Korisnik unosi nove podatke o željenom receptu i šalje zahtjev za promjenom. Poslužitelj izmijeni recept u bazi podataka koja vraća potvrdu.



Slika 3.4: Sekvencijski dijagram za UC8

Obrazac uporabe UC10 – Slanje poruke

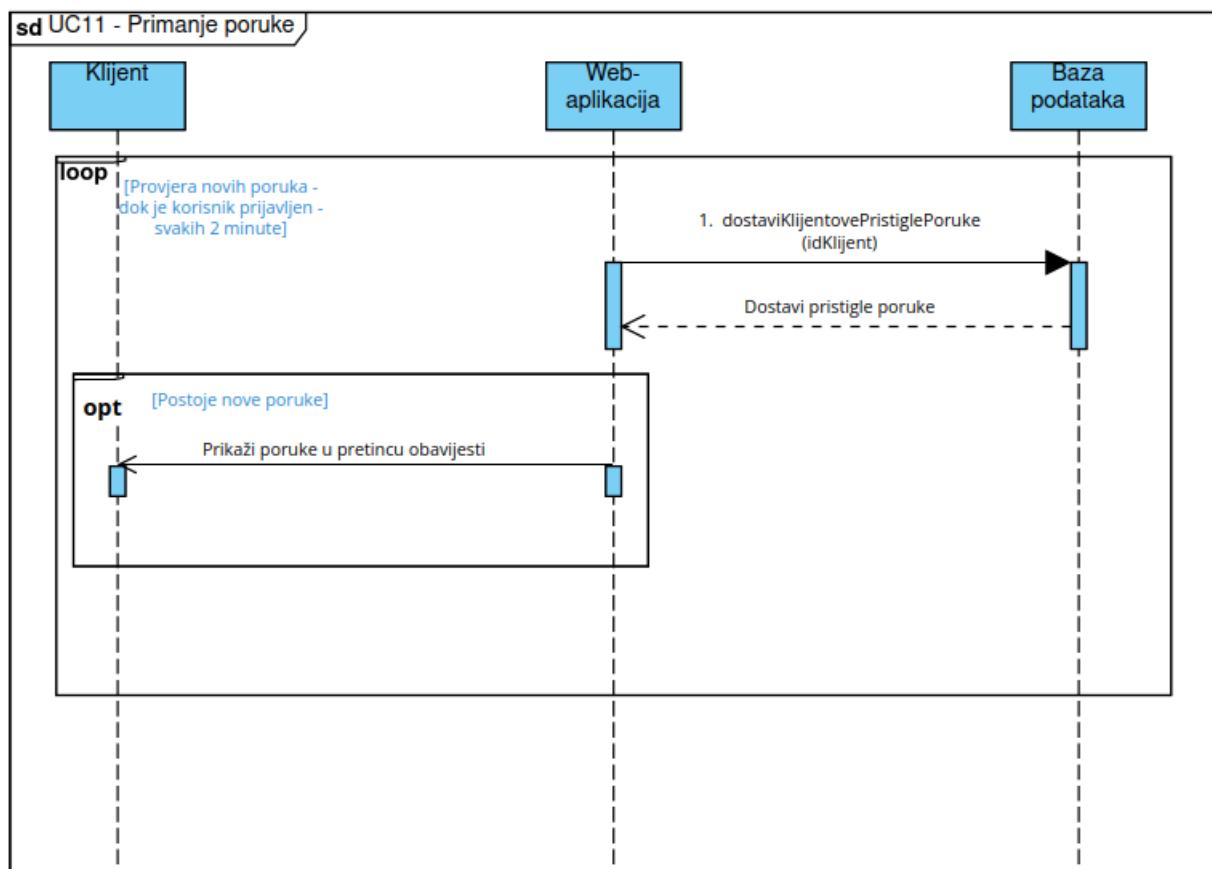
Klijent šalje zahtjev za prikaz recepata po kategorijama. Poslužitelj dohvaća dostupne recepte i prikazuje ih ukoliko postoje. U slučaju da ne postoji niti jedan recept poslužitelj vraća odgovarajuću poruku. Odabirom recepta poslužitelj iz baze podataka dohvaća podatke o receptu i prikazuje ih korisniku. Klijent šalje zahtjev za slanje poruke autoru recepta i samu poruku. Poslužitelj prosljeđuje poruku bazi podataka koja sprema poruku i vraća potvrdu što znači da je poruka posljana.



Slika 3.5: Sekvensijski dijagram za UC10

Obrazac uporabe UC11 – Primanje poruke

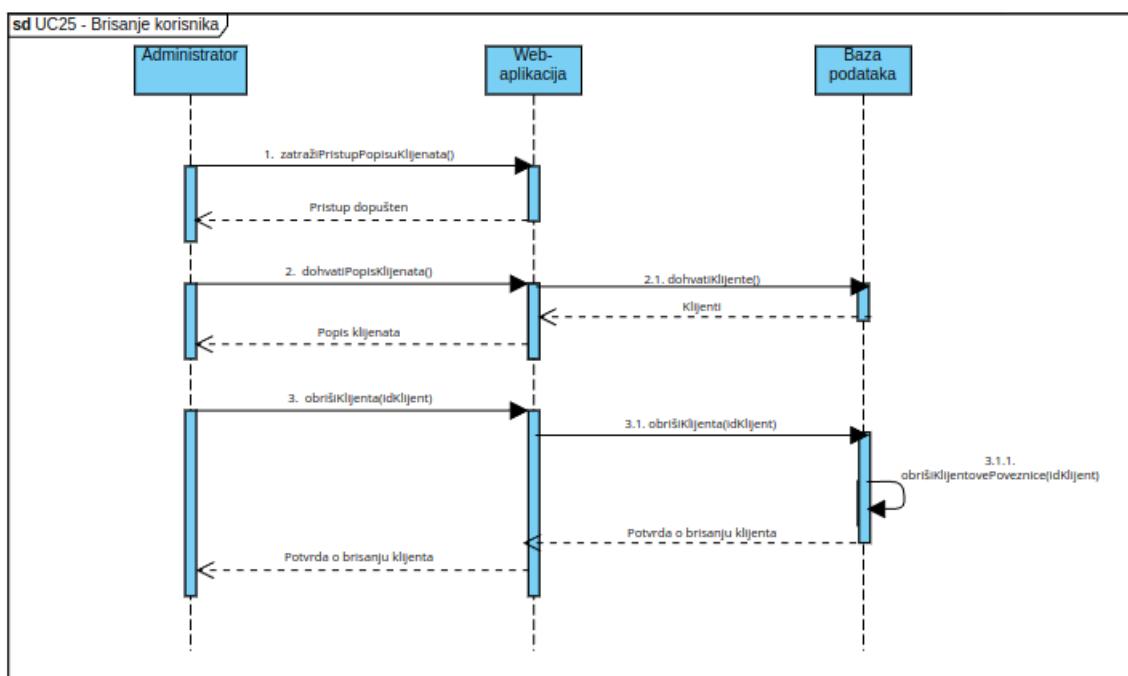
Poslužitelj svake 2 minute u bazi podataka provjerava za prijavljenog korisnika da li su pristigle neke poruke za njega. Ukoliko baza podataka pošalje potvrdu da je primljena poruka poslužitelj javlja klijentu i poruke se prikazuju u pretincu obavijesti.



Slika 3.6: Sekvencijski dijagram za UC11

Obrazac uporabe UC25 – Brisanje korisnika

Administrator traži od poslužitelja pristup popisu klijenata i ukoliko je sve u redu poslužitelj vraća potvrdu i odobrava mu pristup. Zatim administrator šalje zahtjev za prikaz popisa klijenata. Poslužitelj prosljeđuje zahtjev bazi podataka koja šalje potvrdu i administratoru se prikazuje popis klijenata. Odabirom klijenta administrator šalje zahtjev za brisanjem tog klijenta iz baze podataka. Poslužitelj prosljeđuje zahtjev bazi podataka koja briše odabranog klijenta i sve njegove poveznice te na kraju šalje potvrdu o uspješnom brisanju.



Slika 3.7: Sekvencijski dijagram za UC25

3.2 Ostali zahtjevi

1. Mogućnost rada više korisnika u stvarnom vremenu
2. Podržavanje hrvatske abecedu (dijakritičke znakove) pri unosu i prikazu tekstuallnog sadržaja
3. Pristup bazi podataka u ne više od nekoliko sekundi
4. Neispravno korištenje korisničkog sučelja ne izaziva prekid rada sustava
5. Otporna i sigurna veza s bazom podataka
6. Jednostavnost korištenja sustava
7. Oblikovanje sustava pomoću objektno-orientiranih jezika
8. Mogućnost naknadnog dodavanja funkcionalnosti

4. Arhitektura i dizajn sustava

Arhitektura se može podijeliti na tri podsustava:

- Web poslužitelj
- Web aplikacija
- Baza podataka

Web preglednik predstavlja program koji omogućuje korisnicima pregledavanje web-stranica i konzumiranje povezanih multimedijskih sadržaja. Svaki internetski preglednik djeluje kao prevoditelj, interpretirajući kôd web stranica napisan u određenom programskom jeziku i pretvarajući ga u vizualni format razumljiv svakom korisniku. Drugim riječima, stranice su izvorno napisane u kodu, a *web preglednik* ih interpretira kako bi korisnicima pružio čitljiv i interaktivan prikaz. Kada korisnik želi pristupiti određenoj web stranici, šalje zahtjev *web poslužitelju* putem *web preglednika*, što pokreće proces prijenosa informacija i omogućava korisnicima interakciju s odabranim sadržajem.

Web poslužitelj zadužen je za komunikaciju klijenta sa aplikacijom. Međusobna komunikacija odvija se putem HTTP (engl. *Hyper Text Transfer Protocol*) protokola, koji je jedan od protokola za prijenos informacija na webu.

Korisnik koristi *web aplikaciju* za obrađivanje željenih zahtjeva. *Web aplikacija* gleda zahtjev te ovisno o njemu, pristupa bazi podataka nakon čega preko *web poslužitelja* vraća korisniku odgovor u HTML dokumentu koji je vidljiv u *web pregledniku*.

Programski jezik koji smo odabrali za izradu naše *web aplikacije* je Java, zajedno sa Spring Boot radnim okvirom te programski jezik JavaScript zajedno sa React radnim okvirom.

Ovisno o ulogama u timu, odabrana razvoja okruženja su:

- Frontend - Visual Studio Code
- Backend - IntelliJ IDEA
- Dokumentacija - Visual Studio Code

Arhitekura sustava zasnivat će se na arh. obrascu MVC (Model-View-Controller). Spring Boot nudi veliku podršku za rad sa Spring MVC-om te kao takav odgovara našim zahtjevima (također olakšava konfiguraciju potrebnu za validan rad Spring MVC-a).

Arh. obrazac MVC omogućava nezavisan razvoj pojedinih dijelova aplikacije te kao takav olakšava ispitivanje, razvijanje i dodavanje novih svojstva u aplikaciju.

Temeljne komponente MVC koncepta su:

- Model - dinamičke strukture podataka, npr. entitet User, Ingredient...
- View - ono što korisnik vidi, vizualizacija podataka
- Controller - poveznica između Model-a i View-a, bavi se HTTP zahtjevima i odgovorima

4.1 Baza podataka

U našem sustavu, koristit ćemo relaciju bazu podataka koja je strukturon prilagođena modeliranju stvarnog svijeta. Osnovna gradivna jedinka baze je relacija, tj. tablica koja je identificirana svojim imenom i skupom atributa. Glavna svrha baze podataka je brza i jednostavna pohrana, izmjena te dohvata podataka za određenu uporabu.

Upravitelj bazom podataka:

- PostgreSQL

Entiteti baze podataka:

- users
- userFollow
- chatMessage
- communicationTime
- recipe
- recipeRating
- bookmarkedRecipe
- role
- category
- tag
- ingredient
- cuisine
- video
- image
- media

4.1.1 Opis tablica

Relacija **users** pohranjuje entitete modela koji predstavljaju korisnika. Ograničenje stranog ključa odnosi se na atribut roleId koji označava id razine pristup (Klijent, Vlasnik, Admin) koju korisnik ima; referencirana je relacija role, stupac id.

users		
id	INT	jedinstveni identifikator korisnika
firstName	VARCHAR	ime korisnika
lastName	VARCHAR	prezime korisnika
username	VARCHAR	korisničko ime korisnika
email	VARCHAR	adresa elektroničke pošte korisnika
password	VARCHAR	hash lozinke
roleId	VARCHAR	identifikator razine pristupa korisnika (role.id)

Relacija **userFollow** pohranjuje entitete modela koji predstavlja odnos praćenja korisnika i autora. Ograničenje stranog ključa odnosi se na attribute followerId i authorId koji označavaju id korisnika koji je inicirao praćenje, tj. korisnika koji je autor recepta; referencirana je relacija users, stupac id.

userFollow		
id	INT	jedinstveni identifikator modela praćenja
followerId	INT	id korisnika koji prati (user.id)
authorId	INT	id autora kojeg se prati (user.id)
followedAt	TIMESTAMP	vrijeme početka praćenja

Relacija **chatMessage** pohranjuje entitete modela koji predstavlja poruke koje se koriste pri komunikaciji. Ograničenje stranog ključa odnosi se na attribute senderId i receiverId koji označavaju id korisnika koji je posalo poruku, tj. korisnika koji poruku treba primiti; referencirana je relacija users, stupac id.

chatMessage		
id	INT	jedinstveni identifikator poruke (user.id)
senderId	INT	id pošiljatelja (user.id)
receiverId	INT	id primatelja (user.id)
content	VARCHAR	sadržaj poruke

Relacija **communicationTime** pohranjuje entitete modela koji predstavlja dostupnost korisnika za komunikaciju. Ograničenje stranog ključa odnosi se na atribut userId koji označava id korisnika koji prikazuje vrijeme u koje je dostupan za komunikaciju; referencirana je relacija users, stupac id.

communicationTime		
id	INT	jedinstveni identifikator
userId	INT	jedinstveni identifikator korisnika (user.id)
start	TIMESTAMP	vrijeme od
end	TIMESTAMP	vrijeme do

Relacija **recipe** pohranjuje entitete modela recept. Ograničenje stranog ključa odnosi se na atribute ingredientId, tagId, categoryId, cuisineId, mediaId te userId koji označavaju, redom, sastojke recepta, oznake recepta, kategoriju u koju recept spada, kuhinju u koju recept spada, multimedijiske sadržaje koje recept sadrži te autora recepta. Relacije koje se referenciraju su, redom, ingredient (stupac id), tag (stupac id), category (stupac id), cuisine (stupac id), media (stupac id), user (stupac id).

Zbog jednostavnosti vizualnog prikaza, odnosi @ManyToMany nisu prikazani zasebnom relacijom.

Redom:

- N:N sa ingredient
- N:N sa tag
- N:N sa media

recipe		
id	INT	jedinstveni identifikator recepta
ingredientId	List<INT>	id sastojka (ingredient.id)
tagId	INT	id oznake (tag.id)
categoryId	INT	id kategorije (category.id)
cuisineId	INT	id zemlje podrijetla (cuisine.id)
mediaId	INT	id multimedijске stavke (media.id)
userId	INT	id autora (user.id)
title	VARCHAR	naslov recepta
description	VARCHAR	opis recepta
cookingTime	INTERVAL	vrijeme kuhanja

Relacija **recipeRating** pohranjuje modele entiteta koji predstavljaju *feedback* na recept; ocjenu i komentar. Ograničenje stranog ključa odnosi se na atribut `userId` i `recipeId` koji označavaju id korisnika koji daje *feedback* i recept koji se komentira; referencirana je relacija `users`, stupac `id` te relacija `recipe`, stupac `id`.

recipeRating		
id	INT	jedinstveni identifikator ocjene
userId	INT	id klijenta koji je dao ocjenu (user.id)
recipeId	INT	id recepta (recipe.id)
rating	INT	ocjena
comment	VARCHAR	komentar
createdAt	TIMESTAMP	vrijeme nastanka ocjene

Relacija **bookmarkedRecipe** pohranjuje model entiteta koji predstavlja zabilježbu recepta od strane nekog korisnika. Ograničenje stranog ključa odnosi se na atribut `userId` i `recipeId` koji označavaju id korisnika koji zabilježuje recept te id recepta koji se zabilježuje; referencirana je relacija `users`, stupac `id` te relacija `recipe`, stupac `id`.

bookmarkedRecipe		
id	INT	jedinstveni identifikator zabilježbe recepta
userId	INT	id klijenta koji je dao ocjenu (user.id)
recipeId	INT	id recepta (recipe.id)
createdAt	TIMESTAMP	vrijeme nastanka ocjene

role Relacija pohranjuje model entiteta koji predstavlja "ulogu", tj. razinu prava pristupa korisnika (Klijent, Vlasnik, Autor).

role		
id	INT	jedinstveni identifikator razine pristupa
name	VARCHAR	ime razine pristupa

Relacija **category** pohranjuje model entiteta koji predstavlja kategoriju u koju recept može pripadati.

category		
id	INT	jedinstveni identifikator kategorije
name	VARCHAR	ime kategorije

Relacija **tag** pohranjuje model entiteta koji predstavlja oznaku koji recept može imati (npr. Vegan, Healthy, Quick...)

tag		
id	INT	jedinstveni identifikator oznake recepta
name	VARCHAR	opis oznake recepta

Relacija **ingredient** pohranjuje model entiteta koji predstavlja sastojak koji recept može sadržavati.

ingredient		
id	INT	jedinstveni identifikator sastojka
name	VARCHAR	ime sastojka

cuisine Relacija pohranjuje model entiteta koji predstavlja zemlju podrijetla recepta, tj. kuhinju iz koje recept potječe.

cuisine		
id	INT	jedinstveni identifikator zemlje kuhanja
name	VARCHAR	ime zemlje kuhanja

Relacija **video** pohranjuje model entiteta koji predstavlja medijski zapis oblika videa.

video		
id	INT	jedinstveni identifikator videa
duration	INTERVAL	trajanje videa

image Relacija pohranjuje model entiteta koji predstavlja medijski zapis oblika fotografije.

image		
id	INT	jedinstveni identifikator fotografije
width	INT	širina fotografije (px)
height	INT	visina fotografije (px)
format	VARCHAR	format fotografije

Relacija **media** pohranjuje model entiteta koji predstavlja medijski sadržaj koji je objavio neki korisnik, uz neki recept. Ograničenje stranog ključa odnosi se na atribut recipeId id recepta uz okji je medijski sadržaj postavljen; referencirana je relacija recipe, stupac id.

Medijska stavka može biti ili slika ili video.

media		
id	INT	jedinstveni identifikator multimedijске stavke
recipeId	INT	id recepta (px)
key	VARCHAR	naziv datoteke na oblaku
description	VARCHAR	opis multimedijске stavke
uploadDate	DATE	datum postavljanje multimedijске stavke

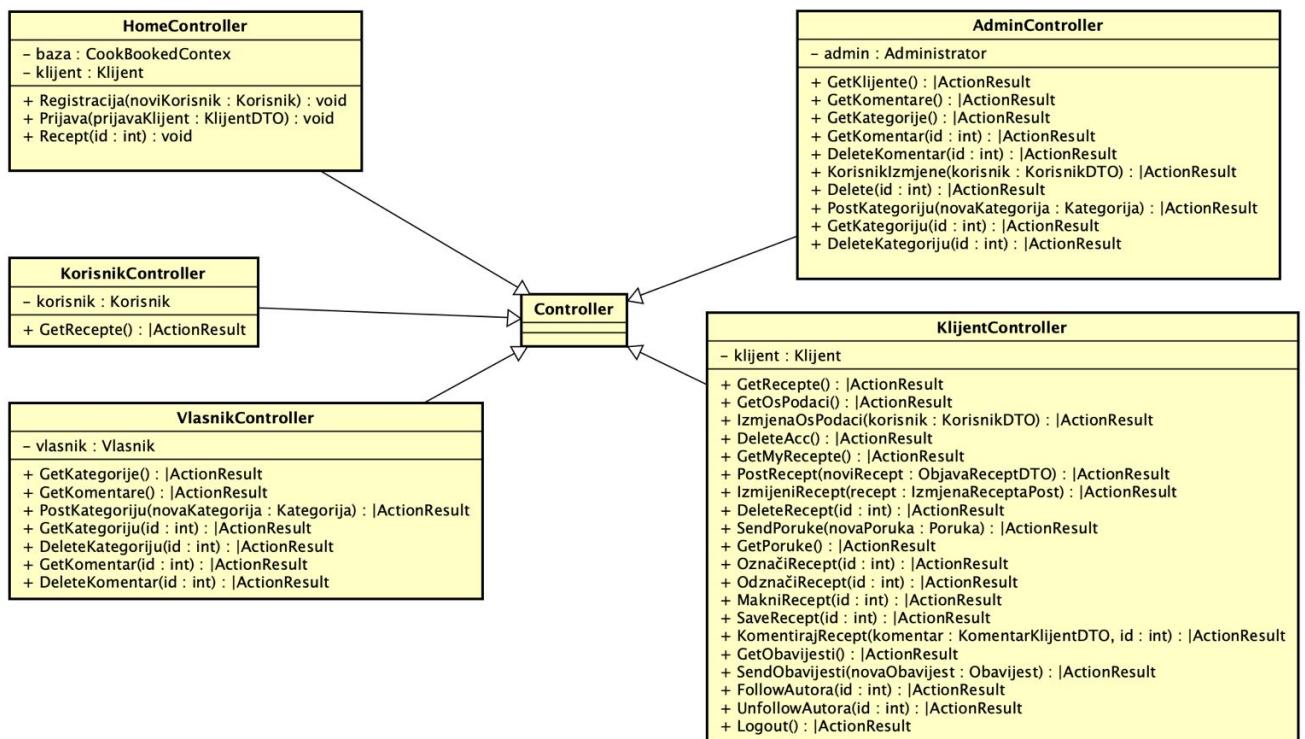
4.1.2 Dijagram baze podataka



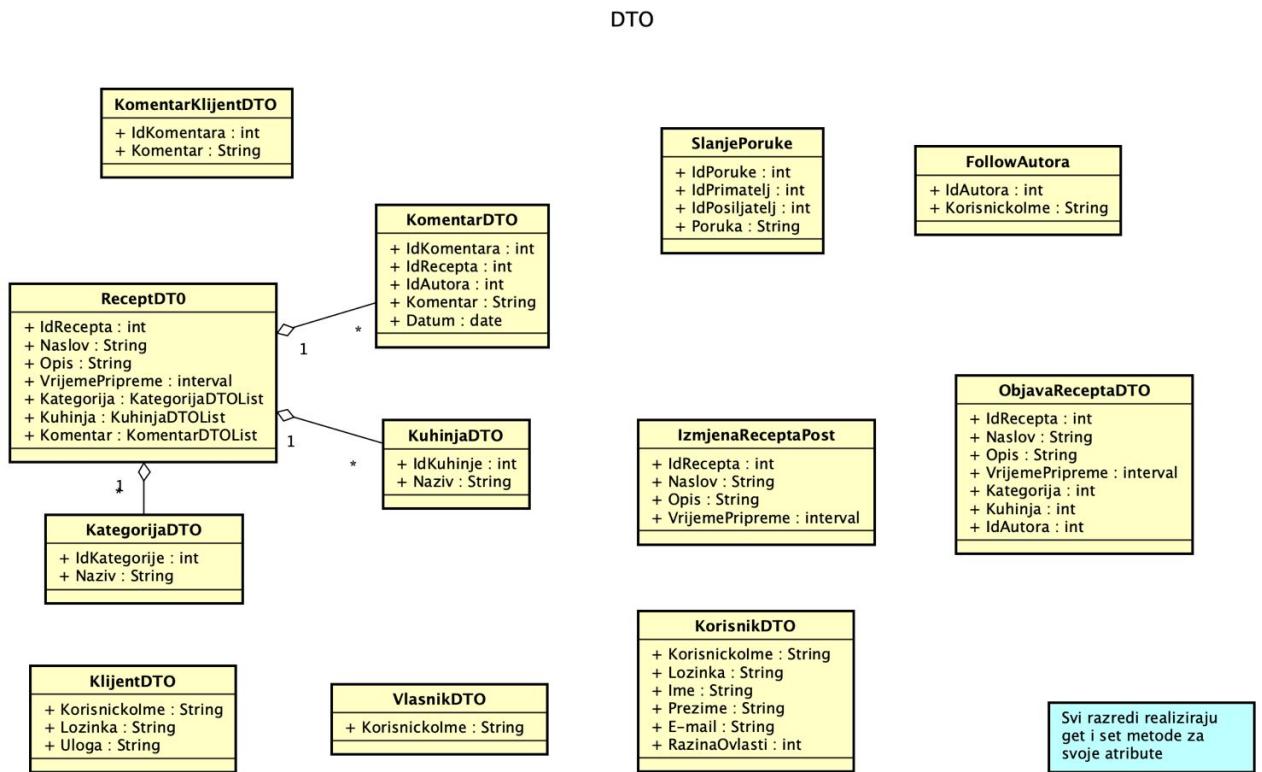
Slika 4.1: Dijagram baze podataka

4.2 Dijagram razreda

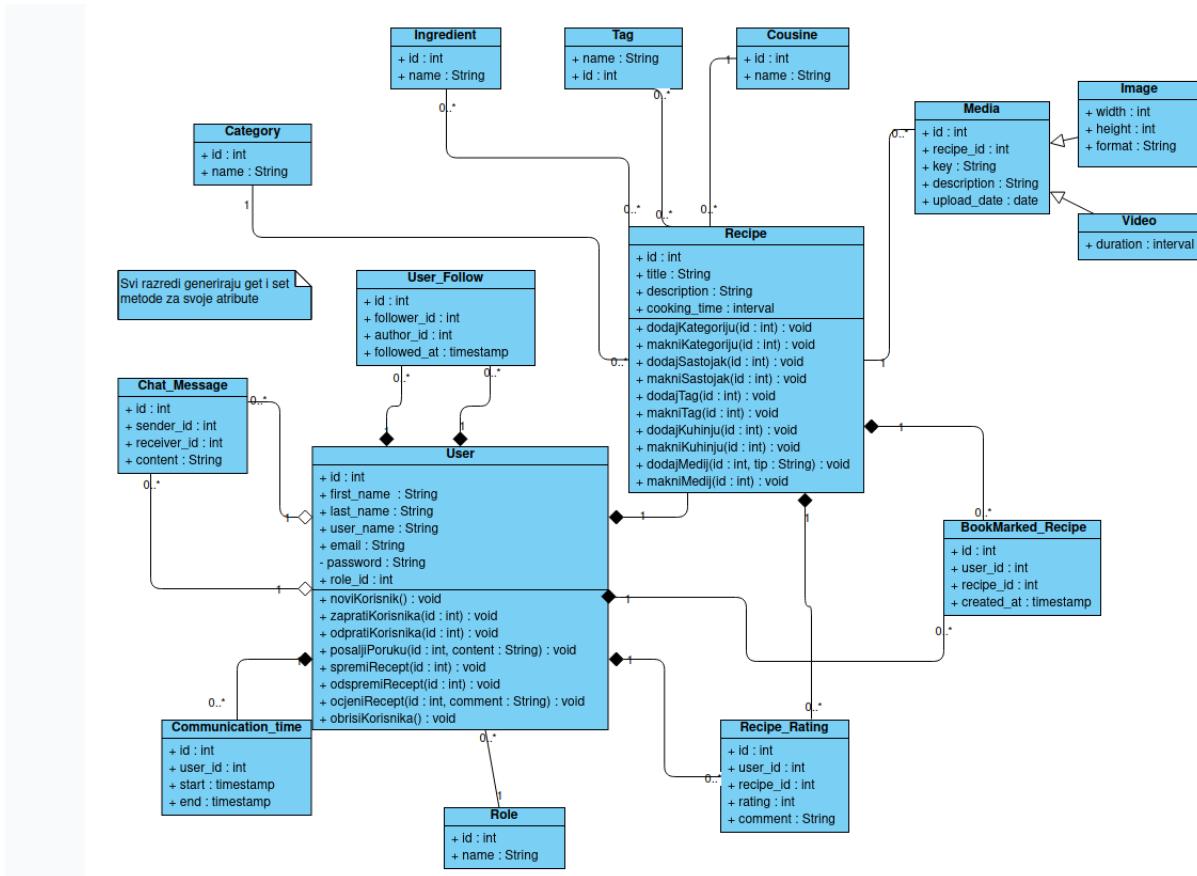
Slike 4.2, 4.3 i 4.4 prikazuju razrede koji pripadaju backend dijelu odabrane arhitektura sustava (MVC). Razredi prikazani na slici 4.3 nasljeđuju Controller razred. Metode unutar tih razreda služe za manipulaciju podatcima pomoću DTO-a (Data Transfer Object). Podatci se dobivaju kroz metode koje su implementirane u Model razredima. Metode unutar Controller razreda odgovorne su za generiranje JSON datoteka i odgovarajućeg HTML statusnog koda kao povratne informacije na njihovo izvršavanje.



Slika 4.2: Dijagram razreda - dio Controllers



Slika 4.3: Dijagram razreda - dio Data transfer objects



Slika 4.4: Dijagram razreda - dio Models

4.3 Dijagrami nakon implementacije programskih rješenja

Prikaz dijagrama razreda nakon završetka druge faze projekta nalazi se direktoriju ”/Dokumentacija” u dva oblika:

- CookBookedModel.plantuml
- CookBookedModelImage.png

Izvedba je u ovom obliku zbog preglednosti prikaza istih.

Grafovi su automatski generirani pomoću IntelliJ IDEA Ultimate razvojnog okruženja.

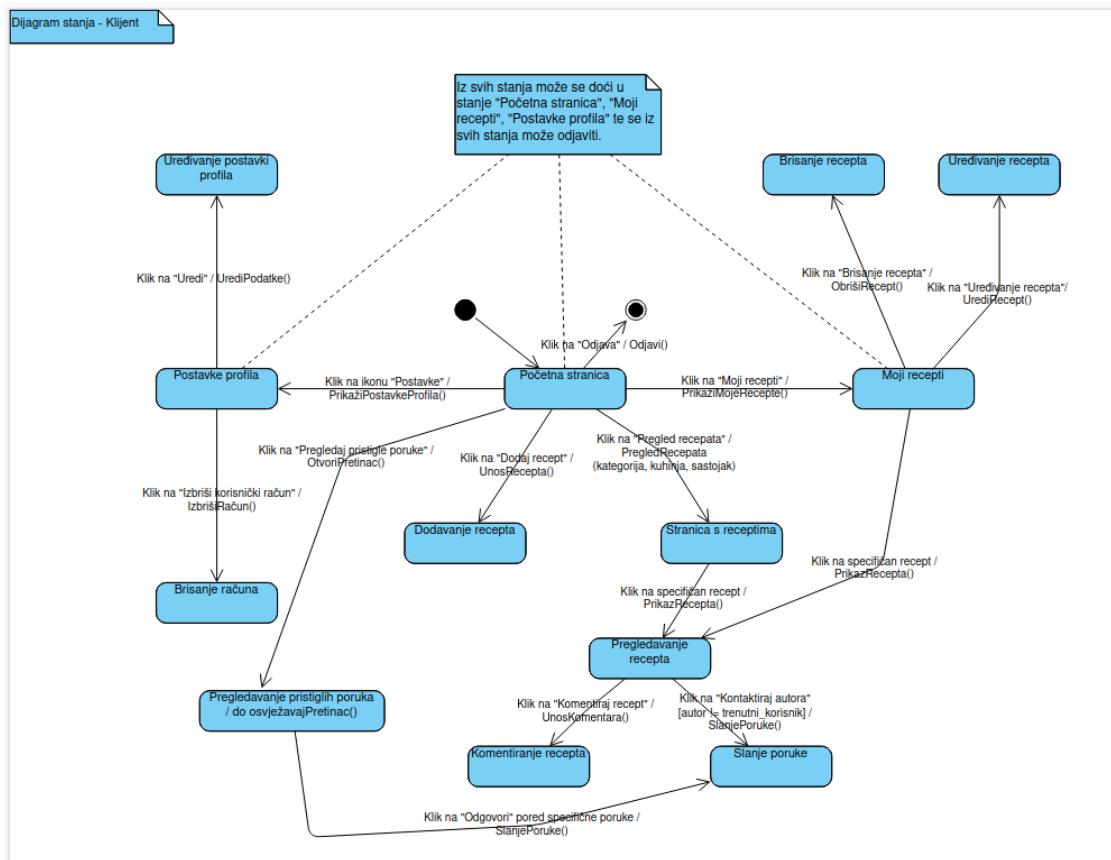
4.4 Dijagram stanja

Dijagram stanja prikazuje stanja objekata te prijelaze iz jednog stanja u drugo temeljene na događajima. Na slici 4.5 prikazan je dijagram stanja za registriranog korisnika.

Nakon što se prijavi, klijent je preusmjeren na početnu stranicu s koje može pristupiti ostalim stranicama ("Postavke profila", "Moji recepti", "Dodavanje recepta", "Stranica s receptima", "Pregledavanje pristiglih poruka").

Ako klijent odluči pregledavati recepte, iz padajućeg izbornika ima opciju filtrirati iste po kategoriji, kuhinji ili sastojcima recepta. Nakon prikaza svih filtriranih recepta, klijent može detaljnije pregledati svaki pojedinačni recept te objaviti komentar ili kontaktirati njegova autora.

Preostale stranice nude očekivane opcije vezane uz postavke korisničkog profila, brisanje korisničkog profila, uređivanje već objavljenih recepta, njihovo brisanje, dodavanje novih te komunikaciju sa korisnicima koji su temeljem već objavljenih recepta klijenta kontaktirali.

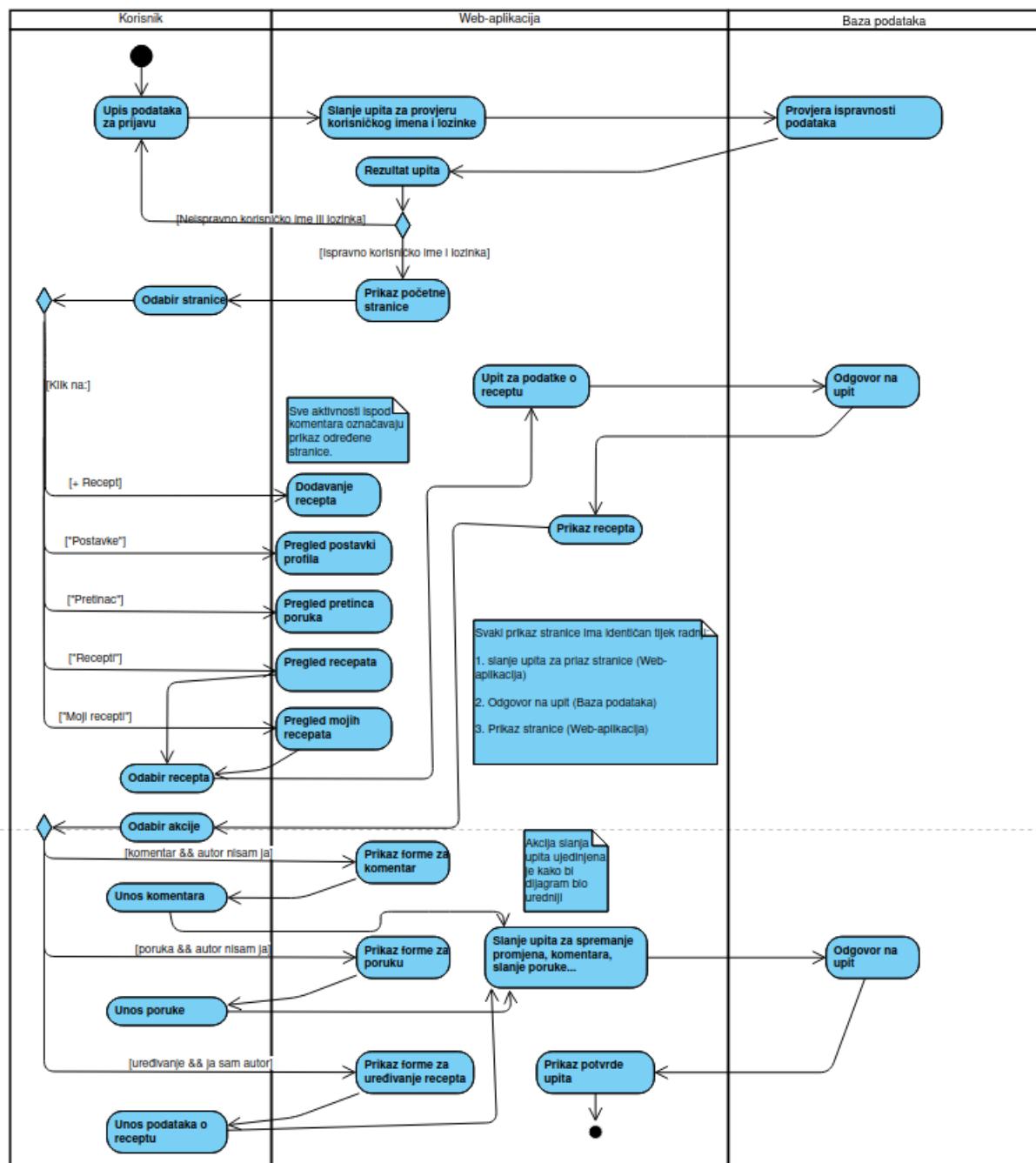


Slika 4.5: Dijagram stanja

4.5 Dijagram aktivnosti

Na slici 4.6 prikazi su procesi akcija koje korisnik može izvršiti ukoliko se nađe na početnoj stranici. Detaljnije je obrađen slučaj ako korisnik želi pregledati pojedine recepte. Korisnik se prijavi u sustav, odabere koju stranicu želi vidjeti (detaljnije obrađena stranica prikaza recepata), a odabirom prikaza recepata prikažu mu se recepti. Odabirom pojedinog recepta korisnik može komentirati recept, poslati poruku autoru ili, ako je recept njegov, urediti podatke o receptu.

Svaka od akcija uključuje ispunjenje određe forme od strane korisnika, slanje upita od strane web-aplikacije bazi podataka potom odgovor baze podataka na taj upit te prikaz tog odgovora od strane web-aplikacije.



Slika 4.6: Dijagram aktivnosti

Na slici 4.7 prikazan je dijagram komponenti koji opisuje organizaciju i međuovisnost komponenti, interne strukture i odnose prema okolini.

Sustavu se može pristupiti preko dva raziličita sučelja:

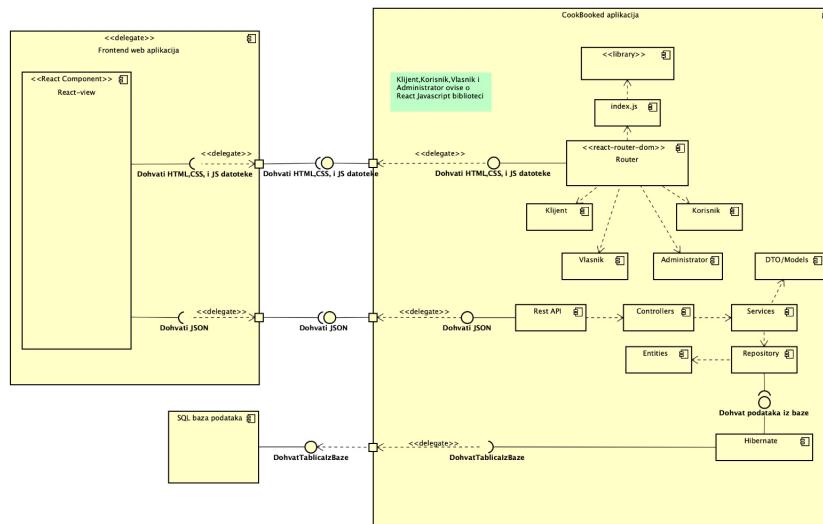
1. Sučelje za dohvata HTML, CSS i JS datoteka

- služi za posluživanje datoteka koje pripadaju *frontend* dijelu aplikacije
- router je komponenta koja ovisno o url-u u upitu poslužuje određene datoteke
- *frontend* dio sastoji se od niza JavaScript datoteka raspoređenih u logičke cjeline
- JavaScript datoteke ovise o React biblioteci

2. Sučelje za dohvata JSON podataka

- preko njega se pristupa REST API komponenti
- REST API poslužuje podatke koji pripadaju *backend* dijelu aplikacije
- Hibernate je zadužen za dohvaćanje tablica iz baze podataka pomoću SQL upita
- podaci koji su pristigli iz baze se šalju dalje MVC arhitekturi u obliku DTO-a i/ili modela

React-view komponenta preko dosputnih sučelja komunicira sa CookBooked aplikacijom, omogućuje ažuriranje prikaza ovisno o korisničkim radnjama te omogućuje dohvaćanje novih podataka ili datoteka prema potrebi.



Slika 4.7: Dijagram aktivnosti

5. Implementacija i korisničko sučelje

5.1 Korištene tehnologije i alati

Komunikacija u timu realizirana je korištenjem aplikacije Discord¹ te aplikacija WhatsApp². Za izradu UML dijagrama korišten je web servis Visual Paradigm Online³, a kao sustav za upravljanje izvornim kodom Git⁴. Udaljeni rezervorij projekta je dostupan na web platformi GitHub⁵.

Kao razvojna okruženja korišteni su:

- IntelliJ IDEA Ultimate⁶ - *backend*
- Microsoft Visual Studio⁷ - *frontend, dokumentacija*

Integrirana razvojna okruženja tvrtke JetBrains (1.), tj. Microsoft (2.).

Aplikacija je napisana koristeći radni okvir Spring Boot⁸ i jezik Java⁹ za izradu *backend-a*, tj. radni okvir React¹⁰ i jezik JavaScript¹¹ za izradu *frontend-a*.

Baza podataka nalazi se na poslužitelju u oblaku DigitalOcean¹².

¹<https://discord.com/>

²<https://www.whatsapp.com/>

³<https://online.visual-paradigm.com/>

⁴<https://git-scm.com/>

⁵<https://github.com/>

⁶<https://online.visual-paradigm.com/>

⁷<https://visualstudio.microsoft.com/>

⁸<https://spring.io/projects/spring-boot/>

⁹<https://www.java.com/en/>

¹⁰<https://reactjs.org/>

¹¹<https://www.javascript.com/>

¹²<https://www.digitalocean.com/>

5.2 Ispitivanje programskog rješenja

5.2.1 Ispitivanje komponenti

Svi testovi napravljeni su pomoću JUnit, AssertJ i Mockito dependancy-a.

Za repliciranje kontrolera, servisa i repozitorija korištene su @MockBean, @Mock te @InjectMocks anotacije.

U svim testovima korišten je "Given-When-Then" način testiranja.

Za svaki test prikazan je dio izvornog koda (u obliku slike zbog preglednosti). Način nazivanja svake test funkcije prikazuje gdje se testira, što se testira te koji se povratni rezulat očekuje.

AuthorizationControllerTest

```

3 usages
private LoginModel validLoginModel = new LoginModel( username: "valid", password: "valid");
3 usages
private LoginModel invalidLoginModel = new LoginModel( username: "invalid", password: "invalid");

└ mastuhne
@Test
public void AuthorizationController_Login_ReturnIsOk() throws Exception {
    String expectedToken = "generatedToken";
    Mockito.when(authorizationService.generateToken(validLoginModel.getUsername(), validLoginModel.getPassword()))
        .thenReturn(expectedToken);

    mockMvc.perform(MockMvcRequestBuilders
                    .post( urlTemplate: "/login")
                    .contentType(MediaType.APPLICATION_JSON)
                    .content(objectMapper.writeValueAsString(validLoginModel)))
        .andExpect(MockMvcResultMatchers.status().isOk())
        .andExpect(MockMvcResultMatchers.content().string(expectedToken));
}

└ mastuhne
@Test
public void AuthorizationController_Login_ReturnBadRequest() throws Exception {
    Mockito.when(authorizationService.generateToken(invalidLoginModel.getUsername(), invalidLoginModel.getPassword()))
        .thenThrow(new BadCredentialsException(""));

    mockMvc.perform(MockMvcRequestBuilders
                    .post( urlTemplate: "/login")
                    .contentType(MediaType.APPLICATION_JSON)
                    .content(objectMapper.writeValueAsString(invalidLoginModel)))
        .andExpect(MockMvcResultMatchers.status().isBadRequest());
}

```

Slika 5.1: AuthorizationControllerTest kod

RecipeControllerTest

```

└ mastuhne
@Test
public void RecipeController_GetById_ReturnRecipe() throws Exception {
    Integer givenId = 12345678;
    String givenTitle = "testName";
    Mockito.when(recipeService.getRecipeById(givenId))
        .thenReturn(Recipe.builder().title(givenTitle).id(givenId).build());

    mockMvc.perform(MockMvcRequestBuilders
                    .get( urlTemplate: "/recipes/{id}", givenId)
                    .accept(MediaType.APPLICATION_JSON))
        .andExpect(MockMvcResultMatchers.jsonPath( expression: "$.id").value(givenId))
        .andExpect(MockMvcResultMatchers.jsonPath( expression: "$.title").value(givenTitle))
        .andReturn();
}

```

Slika 5.2: RecipeControllerTest kod

CategoryServiceTest

```

6 usages
private List<Category> categories = new ArrayList<>();

▲ mastuhne
@BeforeEach
public void setup() {
    categories.add(new Category( id: 1, name: "first", new ArrayList<>()));
    categories.add(new Category( id: 2, name: "second", new ArrayList<>()));
    categories.add(new Category( id: 3, name: "third", new ArrayList<>()));
    categories.add(new Category( id: 4, name: "fourth", new ArrayList<>()));
    categories.add(new Category( id: 5, name: "fifth", new ArrayList<>()));
}

▲ mastuhne
@Test
void CategoryService_GetAll_ReturnCategories() {
    given(categoryRepostiory.findAll()).willReturn(categories);

    List<Category> categoryList = categoryService.getAllCategories();

    assertThat(categoryList).isNotNull();
    assertThat(categoryList.size()).isEqualTo( expected: 5);
}

```

Slika 5.3: CategoryServiceTest kod

RecipeServiceTest

```

▲ mastuhne
@Test
public void RecipeController_GetById_ReturnRecipe() throws Exception {
    Integer givenId = 12345678;
    String givenTitle = "testName";
    Mockito.when(recipeService.getRecipeById(givenId))
        .thenReturn(Recipe.builder().title(givenTitle).id(givenId).build());

    mockMvc.perform(MockMvcRequestBuilders
        .get( urlTemplate: "/recipes/{id}", givenId)
        .accept(MediaType.APPLICATION_JSON))
        .andExpect(MockMvcResultMatchers.jsonPath( expression: "$.id").value(givenId))
        .andExpect(MockMvcResultMatchers.jsonPath( expression: "$.title").value(givenTitle))
        .andReturn();
}

```

Slika 5.4: RecipeServiceTest kod

UserRepositoryTest

```
└─ mastuhne
  └─ Test
    └─ @DisplayName("UserRepository_Save_ReturnSavedUser")
      public void UserRepository_Save_ReturnSavedUser() {
        User user = new User( id: 1, firstName: "test", lastName: "test", phoneNumber: "test", username: "test", password: "test", email: "test", new Role());
        User savedUser = userRepository.save(user);
        Assertions.assertThat(savedUser).isNotNull();
        Assertions.assertThat(savedUser.getId()).isGreaterThan( other: 0);
      }
}
```

Slika 5.5: UserRepositoryTest kod

Prikaz rezultata testova:

```
[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 1.614 s - in com.fer.progi.errormasters.Cookbooked.controllers.AuthorizationControllerTest
[INFO] Running com.fer.progi.errormasters.Cookbooked.controllers.RecipeControllerTest
```

Slika 5.6: Rezultat AuthorizationControllerTest-a

```
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.602 s - in com.fer.progi.errormasters.Cookbooked.controllers.RecipeControllerTest
[INFO] Running com.fer.progi.errormasters.Cookbooked.CookbookedApplicationTests
```

Slika 5.7: Rezultat UserRepositoryTest-a

```
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 4.002 s - in com.fer.progi.errormasters.Cookbooked.CookbookedApplicationTests
[INFO] Running com.fer.progi.errormasters.Cookbooked.services.CategoryServiceTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.093 s - in com.fer.progi.errormasters.Cookbooked.services.CategoryServiceTest
[INFO] Running com.fer.progi.errormasters.Cookbooked.services.RecipeServiceTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.071 s - in com.fer.progi.errormasters.Cookbooked.services.RecipeServiceTest
```

Slika 5.8: Rezultat ... testa

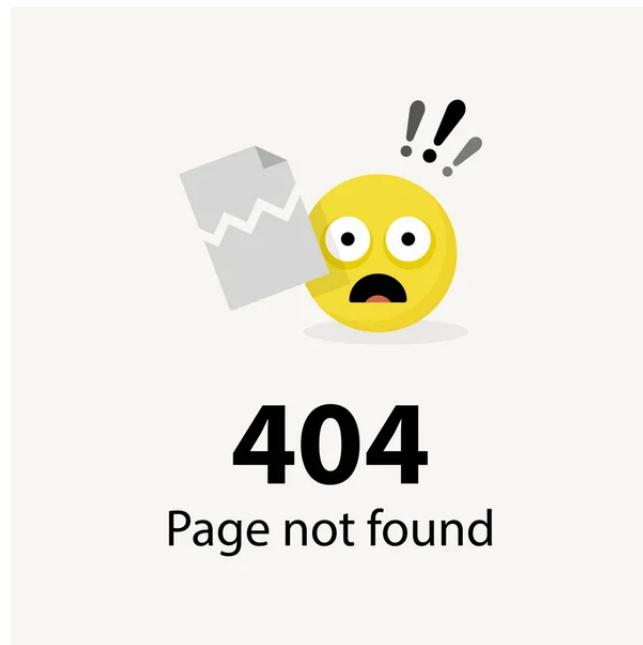
```
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 5.838 s - in com.fer.progi.errormasters.Cookbooked.repositories.UserRepositoryTest
[INFO] Running com.fer.progi.errormasters.Cookbooked.controllers.AuthorizationControllerTest
```

Slika 5.9: Rezultat UserRepositoryTest-a

```
[INFO] Results:
[INFO]
[INFO] Tests run: 7, Failures: 0, Errors: 0, Skipped: 0
[INFO]
```

Slika 5.10: Rezultat svih testova

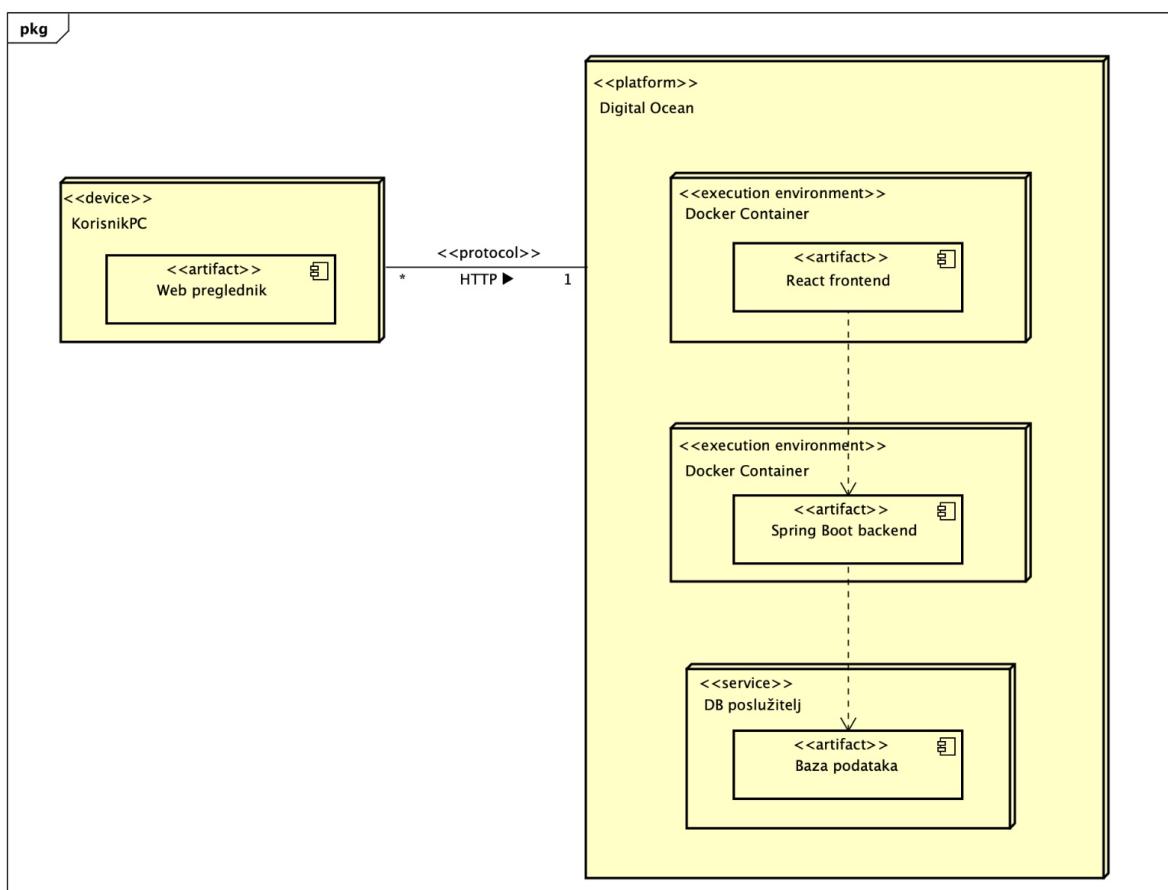
5.2.2 Ispitivanje sustava



Slika 5.11: Selenium

5.3 Dijagram razmještaja

Dijagram razmještaja prikazuje fizički razmještaj programskih artefakata na fizičkoj ili virtualnoj infrastrukturi. Na platformi DigitalOcean implementirano je rješenje koje koristi Docker kontejnere za ispunjavanje pozadinske funkcionalnosti web aplikacije. Klijenti koriste web preglednik kako bi pristupili web aplikaciji. Sustav je baziran na arhitekturu "klijent-poslužitelj", dok je komunikacija između računala korisnika i računala poslužitelja ostvarena putem HTTP veze.



Slika 5.12: Dijagram aktivnosti

5.4 Upute za puštanje u pogon

Instalacija i konfiguracija baze podataka Potrebno je otići na platformu koja nudi usluge poslužitelja baze podataka, u našem slučaju, *DigitalOcean*. Također, odlučili smo se koristiti PostgreSQL za pohranu podataka.

Kod stvaranja baze podataka treba odabrati slijedeće postavke:

- Regija podatkovnog centra (data center region) – Amsterdam
- Baza podataka – PostgreSQL v15
- CPU – Regular
- Plan – 1vCPU, 2 GB RAM

Create Database Cluster

The screenshot shows the 'Create Database Cluster' interface on the DigitalOcean website. The process is divided into several sections:

- Choose a datacenter region:** Shows 'Amsterdam • Datacenter 3 • AMS3' selected, with '1 resource' available.
- VPC network - default-ams3:** Shows 'All resources created in this datacenter will be members of the same VPC network'. A note says they can communicate securely over their Private IP addresses.
- Select a plan:** Shows two options:
 - \$13.00/mo: 1vCPU / 1 GB RAM / Storage minimum: 10 GB / Connection limit: 22
 - \$24.00/mo**: Selected option. 1vCPU / 2 GB RAM / Storage minimum: 30 GB / Connection limit: 47
- Choose a database engine:** Shows PostgreSQL v15 selected, with other options like MongoDB, MySQL, Redis, and Kafka available.
- CPU options:** Shows 'Regular' selected with 'Disk: SSD'.
- Premium AMD** and **Premium Intel** options are also listed.

Slika 5.13: Konfiguracija baze podataka

Nakon uspješnog stvaranja baze podataka platforma će nas proslijediti na stranicu na kojoj možemo provesti dodatnu konfiguraciju baze podataka. Na toj se stranici nalaze upute za spajanje (na bazu), koje ćemo iskoristiti kako bismo dobili izravan pristup bazi.

The screenshot shows the 'Overview' tab of a PostgreSQL cluster named 'db-postgresql-cookbooked'. The cluster is located in the 'cookbooked' region with 2 GB RAM, 1 vCPU, and 30 GB Disk, running on an AMS3 instance with PostgreSQL 15.

READ ONLY NODES: A section indicating no read-only nodes have been added yet, with a link to 'Add a read-only node'.

DATABASE CLUSTER TOTAL COST: Shows a monthly rate of '\$30.00'. A note states this amount is prorated and does not reflect month-to-date usage.

TRUSTED SOURCES: A warning message states the cluster is open to all incoming connections, with a link to 'Secure this database cluster by restricting access'.

CONNECTION DETAILS: A panel showing connection parameters:

```
username = doadmin
password = *****
show
host = db-postgresql-cookbooked-do-user-14073808-0.c.db.ondigitalocean.com
port = 25060
database = defaultdb
sslmode = require
```

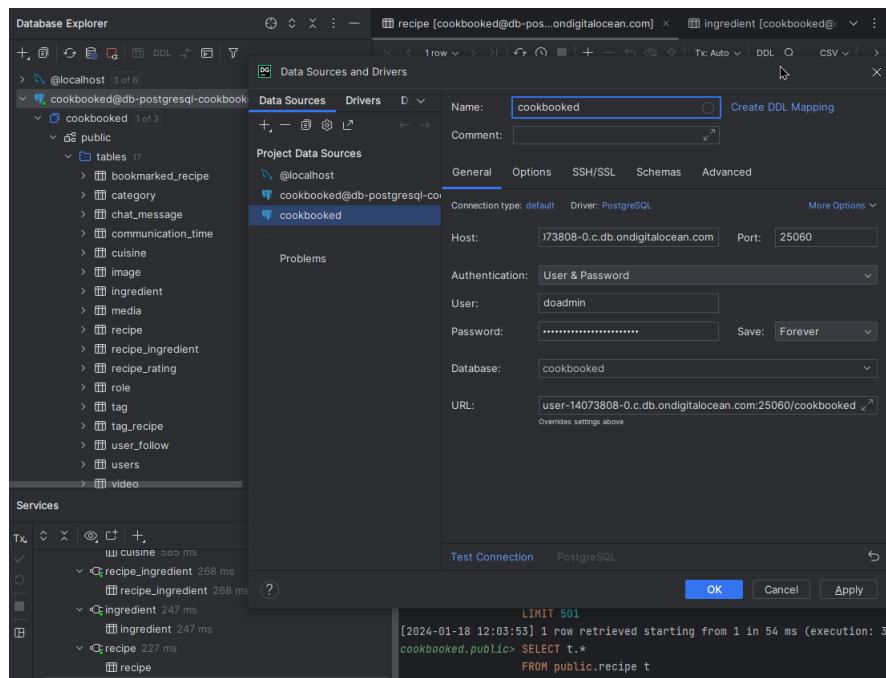
User: doadmin | Database/Pool: defaultdb

[Copy](#) [Download CA certificate](#)

Slika 5.14: Dodatna konfiguracija baze podataka

Spajanje na bazu podataka

Spajanje na bazu provodimo putem okoline za upravljanje bazom podataka. U našem slučaju koristili smo JetBrains-ov alat *DataGrip*. Navigacijom na *Database Explorer* odabiremo opciju *New* te u padajućem izborniku odaberemo opciju *DataSource - PostgreSQL*. U otvorenom prozoru unosimo podatke koje smo dobili kao povratnu informaciju prilikom kreiranja i konfiguiranja baze podataka.



Slika 5.15: Spajanje pomoću alata *DataGrip*

Kopirat ćemo URL, tj. JDBC URL koji ćemo kasnije upisati u *Spring Boot application properties* datoteku.

Pritiskom gumba "OK" stvara se novi unos u *Database Explorer*. Također, klikom na gumb "Test Connection" možemo testirati jesmo li sve podatke unijeli ispravno.

Punjjenje baze podataka

Odabirom unesene baze u *Database Explorer*-u možemo pronaći public shemu koja će biti glavno spremište naših podataka. Klikom desne tipke miša na public shemu, biti će prikazan izbornik u kojem odabiremo opciju *SQL Scripts - Run SQL Script*. Otvoriti će se nova kartica u kojoj ćemo staviti sadržaj "cookbooked.sql" datoteke koja predstavlja opis strukturnog modela baze.

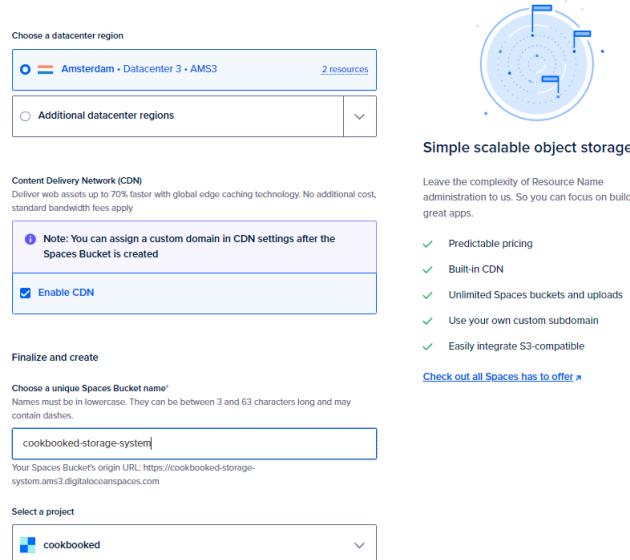
Konfiguracija *Spaces Object Storage*-a za spremanje multimedijskih datoteka

Za pohranu slika i videa koristimo *DigitalOcean*-ov *Spaces Object Storage*. Konfiguracija zahtjeva da odaberemo:

- Regiju – izaberemo najbliži server
- CDN – uključen
- Ime

Nakon što smo kreirali *Spaces Bucket*, biti će nam potreban *API-key* za taj *bucket*, kako bismo ga mogli koristiti na *Spring Boot backend*-u. *API-key* ćemo dobiti oda- birom opcije *API - Spaces Key* u izborniku platforme *DigitalOcean*. Tamo ćemo izgenerirati naše *API* ključeve.

Create a Spaces Bucket



Slika 5.16: Kreiranje *Spaces*

Instalacija i konfiguracija Spring Boot-a Nakon uspješnog konfiguiranja baze podataka i *Spaces*-a, potrebno je klonirati repozitorij aplikacije **CookBooked**. U direktoriju naziva *backend* potrebno je konfigurirati *application.properties* datoteku.

Potrebno je:

- postaviti `spring.datasource.url`
- postaviti `spring.datasource.username`
- postaviti `spring.datasource.password`
- postaviti `s3.access.key`, `s3.secret.key`, `s3.server.url`, `s3.bucket.name`, `s3.bucket.region`

```
# Database connection settings
spring.datasource.driver-class-name=org.postgresql.Driver
spring.datasource.url=jdbc:postgresql://db-postgresql-cookbooked-do-user-14073808-0.c.db.ondigitalocean.com:25060/cookbooked
spring.datasource.username=doadmin
spring.datasource.password=sifra

# Hibernate Postgres Optimisation
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect
spring.jpa.show-sql=false
spring.jpa.properties.hibernate.format_sql=true
spring.jpa.hibernate.naming.physical-strategy= org.hibernate.boot.model.naming.PhysicalNamingStrategyStandardImpl
hibernate.type.preferred_duration_jdbc_type=INTERVAL_SECOND

## DDL-auto (create, create-drop, validate, update)
spring.jpa.hibernate.ddl-auto=none

# endpoint prefix
server.servlet.context-path=/api/v1

# swagger openApi configuration
springdoc.api-docs.path=/api-docs
springdoc.swagger-ui.path=/swagger.html

## s3 configuration
s3.access.key = 0000JNJK0JN7UDBBjf77
s3.secret.key = secret
s3.server.url = ams3.digitalceanspaces.com
s3.bucket.name = cookbooked-storage
s3.bucket.region = ams3

# multipart file upload configuration
spring.servlet.multipart.enabled=true
spring.servlet.multipart.max-file-size=30MB
```

Slika 5.17: Izgled *application.properties* datoteke

Backend REST API-server je tada spreman te ga možemo pokrenuti naredbom `./mvnw spring-boot:run`. Nakon izvršavanja naredbe, *backend* je pokrenut na port-u 8080.

Instalacija i konfiguracija REACT frontend-a

Frontend je potrebno konfigurirati da šalje zahtjeve na *backend* server. To ćemo postići tako da u **.env** datoteci postavimo IP adresu ili domenu *backend* servera. Nakon toga moguće je pokrenuti naredbe **npm install** koja će učitati potrebne biblioteke, **npm build** koja će "izgraditi" našu aplikaciju te **npm run** koja će pokrenuti aplikaciju.

6. Zaključak i budući rad

Zadatak naše grupe bio je razvoj web aplikacije za razmjenu recepata za kuhanje i pečenje kolača te povezivanje s autorima recepata. Razvoj na projektu trajao je 14 tjedana te se je odvijao kroz dvije faze.

1. faza uključivala je:

- okupljanje razvojnog tima
- dodjelu projektnog zadatka
- intenzivan i detaljan rad na dokumentaciji zahtjeva

Kvalitetno provedena prva faza značajno je olakšala daljnji rad na projektu. Dijagrami (obrasci uporabe, sekvencijski dijagrami, dijagram baze podataka, dijagram razreda) koji su izrađeni u ovoj fazi pomogli su razvojnim timovima koji su radili na *frontend-u* i *backend-u* (pogotovo u drugoj fazi projekta).

2. faza uključivala je:

- intenzivan rad na implementaciji programskog rješenja
- dokumentiranje preostalih UML dijagrama
- dokumentiranje unit testova
- izradu popratne dokumentacija

Iako kraća od prve, druga faza pokazala se kao velik zalogaj svim članovima tima. Manjak iskustva većine članova u izradi sličnih implementacijskih rješenja usporio je sam razvoj te stvorio probleme koji su bili u vremenskoj domeni. Tim problemima doprinijela je i lošija organizacija nego u prvoj fazi. Navedeni problemi rezultirali su nedovršenošću Selenium testova te rješavanjem određenih implementacijskih problema "quick-fix" metodama. Unatoč tome, velika većina projekta uspješno je realizirana kroz drugu fazu.

Komunikacija između članova grupe odvijala se putem WhatsApp i Discord grupa; bila je svakodnevna te su u njoj aktivno sudjelovali svi članovi grupe.

Kao "zaključak zaključka" mogli bismo reći kako nam ovaj projekt predstavlja vrijedno iskustvo. Kroz rad na ovom projektu naučili smo da komunikacija ne ide uvijek glatko, da je vremenska organiziranost izuzetno bitna te da je koordinacija

između članova tima vjerojatno i najbitniji aspekt rada na sličnim projektima. Uz to smo naučili i mnogobrojne nove stvari i tehnologije (koje će nam olakšati budući rad) te smatramo kako možemo biti zadovoljni zbog onoga što smo postigli.

Popis literature

Kontinuirano osvježavanje

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

1. Programsко инжењерство, FER ZEMRIS, <http://www.fer.hr/predmet/proinz>

Indeks slika i dijagrama

2.1 Izgled coolinarika web portala	9
2.2 Izgled index-recepti web portala	9
3.1 Dijagram obrasca uporabe, funkcionalnost korisnika i klijenta	22
3.2 Dijagram obrasca uporabe, funkcionalnost vlasnika	23
3.3 Dijagram obrasca uporabe, funkcionalnost administratora	24
3.4 Sekvencijski dijagram za UC8	25
3.5 Sekvencijski dijagram za UC10	26
3.6 Sekvencijski dijagram za UC11	27
3.7 Sekvencijski dijagram za UC25	28
4.1 Dijagram baze podataka	39
4.2 Dijagram razreda - dio Controllers	41
4.3 Dijagram razreda - dio Data transfer objects	42
4.4 Dijagram razreda - dio Models	43
4.5 Dijagram stanja	45
4.6 Dijagram aktivnosti	47
4.7 Dijagram aktivnosti	49
5.1 AuthorizationControllerTest kod	52
5.2 RecipeControllerTest kod	52
5.3 CategoryServiceTest kod	53
5.4 RecipeServiceTest kod	53
5.5 UserRepositoryTest kod	54
5.6 Rezultat AuthorizationControllerTest-a	54
5.7 Rezultat UserRepositoryTest-a	54
5.8 Rezultat ... testa	54
5.9 Rezultat UserRepositoryTest-a	54
5.10 Rezultat svih testova	54
5.11 Selenium	55
5.12 Dijagram aktivnosti	56

5.13 Konfiguracija baze podataka	57
5.14 Dodatna konfiguracija baze podataka	58
5.15 Spajanje pomoću alata <i>DataGrip</i>	59
5.16 Kreiranje <i>Spaces</i>	60
5.17 Izgled <i>application.properties</i> datoteke	61
6.1 Svi git commit-ovi	70
6.2 Matija Alojz Stuhne	70
6.3 Dario Huić	70
6.4 Luka Ivezković	71
6.5 Valentina Ivanić	71
6.6 Petra Habjanec	71
6.7 Dominik Ožvald	71

Dodatak: Prikaz aktivnosti grupe

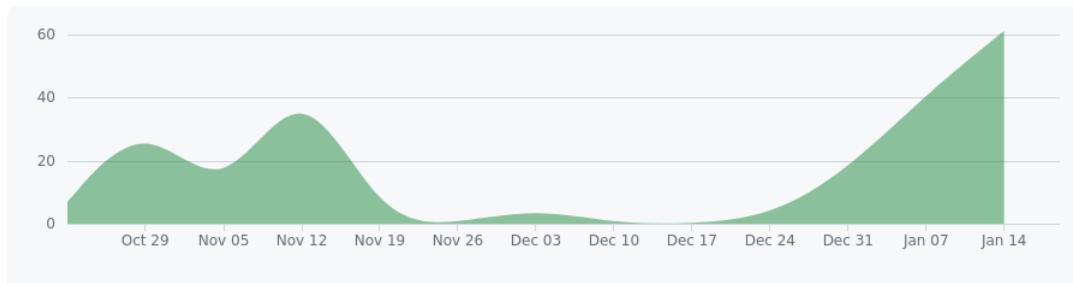
Dnevnik sastajanja

1. komunikacija - svakodnevna putem WhatsApp grupe
 - Datum: svakodnevno
 - Prisustvovali: M. Dananić, P. Habjanec, D. Huić, V. Ivanić, L. Ivezović, D. Ožvald, M. A. Stuhne
 - Teme sastanka:
 - napredak u područjima koda i dokumentacije
2. sastanak
 - Datum: 23.10.2023.
 - Prisustvovali: M. Dananić, P. Habjanec, D. Huić, V. Ivanić, L. Ivezović, D. Ožvald, M. A. Stuhne
 - Teme sastanka:
 - raspodjela posla
 - korištene tehnologije
 - initial commit
3. sastanak
 - Datum: 29.10.2023.
 - Prisustvovali: M. Dananić, P. Habjanec, D. Huić, V. Ivanić, L. Ivezović, D. Ožvald, M. A. Stuhne
 - Teme sastanka:
 - bitni zadatci do 2.11.2023.
 - dogovor oko dokumentacije

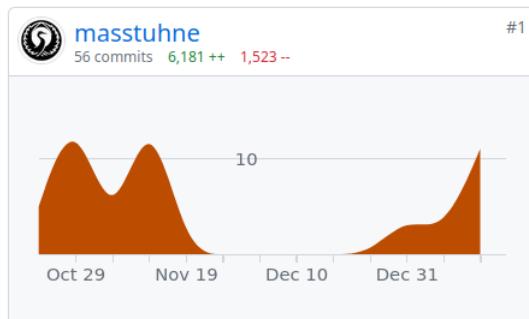
Tablica aktivnosti

		Matija Alojz Stuhne	Marko Dananić	Petra Habjanec	Dario Huić	Valentina Ivanić	Luka Ivezković	Dominik Ožvald
Backend	+				+		+	
Frontend			+	+	+	+		+
Dokumentacija	+	+	+	+				
Prezentacija	+							

Dijagrami pregleda promjena



Slika 6.1: Svi git commit-ovi



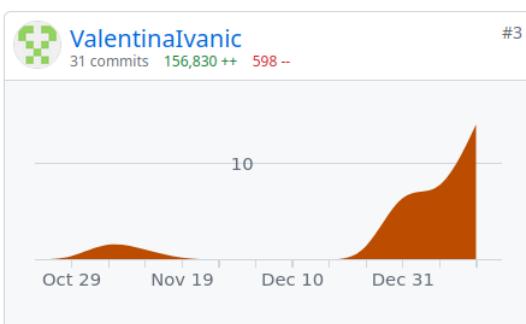
Slika 6.2: Matija Alojz Stuhne



Slika 6.3: Dario Huić



Slika 6.4: Luka Ivezković



Slika 6.5: Valentina Ivanić



Slika 6.6: Petra Habjanec



Slika 6.7: Dominik Ožvald