

Programsko inženjerstvo

Ak. god. 2021./2022.

SkillEtCooking

Dokumentacija, Rev. 2.

Grupa: *Progimeri*

Voditelj: *Karlo Frankola*

Datum predaje: *14. siječnja 2022.*

Nastavnik: *Eugen Vušak, mag. ing. comp*

Sadržaj

1 Dnevnik promjena dokumentacije	3
2 Opis projektnog zadatka	5
3 Specifikacija programske potpore	10
3.1 Funkcionalni zahtjevi	10
3.1.1 Obrasci uporabe	12
3.1.2 Sekvencijski dijagrami	21
3.2 Ostali zahtjevi	25
4 Arhitektura i dizajn sustava	26
4.1 Baza podataka	28
4.1.1 Opis tablica	28
4.1.2 Dijagram baze podataka	32
4.2 Dijagram razreda	33
4.3 Dijagram stanja	36
4.4 Dijagram aktivnosti	37
4.5 Dijagram komponenti	39
5 Implementacija i korisničko sučelje	40
5.1 Korištene tehnologije i alati	40
5.2 Ispitivanje programskog rješenja	41
5.2.1 Ispitivanje komponenti	41
5.2.2 Ispitivanje sustava	46
5.3 Dijagram razmještaja	47
5.4 Upute za puštanje u pogon	48
6 Zaključak i budući rad	53
Popis literature	54
Indeks slika i dijagrama	55

Dodatak: Prikaz aktivnosti grupe

56

1. Dnevnik promjena dokumentacije

Rev.	Opis promjene/dodatka	Autori	Datum
0.1	Napravljen predložak	Karlo Frankola	15.10.2021.
0.2	Opis projektnog zadatka	Dario Dugonjevac	28.10.2021.
0.3	Dodani funkcionalni zahtjevi	Borna Colarić	31.10.2021.
0.4	Dodani obrasci uporabe	Toni Serezlija	01.11.2021.
0.5	Napravljeni dijagrami obrazaca uporabe	Marko Tunjić	02.11.2021.
0.6	Sekvencijski dijagrami	Dario Dugonjevac	12.11.2021.
0.7	Definirani nefunkcionalni zahtjevi	Toni Serezlija	13.11.2021.
0.8	Opis tablica baze podataka	Karlo Frankola	14.11.2021.
0.9	Dijagram baze podataka	Marko Tunjić	15.11.2021.
0.10	Dijagram razreda	Jan Brkić	16.11.2021.
0.10.1	Dopunjeni dijagrami razreda	Karlo Frankola	19.11.2021.

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

Rev.	Opis promjene/dodatka	Autori	Datum
1.0	Dopuna dnevnika aktivnosti i prva predaja	Karlo Frankola	19.11.2021.
1.1	Dodana dokumentacija za ispitivanje komponenti	Marko Tunjić	13.01.2021.
1.2	Dodani dijagrami stanja i aktivnosti	Jan Brkić	13.01.2021.
1.3	Dodan dijagram razreda i razmještaja, upute za pokretanje	Karlo Frankola	14.01.2021.
1.4	Dodan zaključak, dijagram komponenti, korištene tehnologije	Dario Dugonjevac	14.01.2021.
1.5	Popravak dijagrama stanja	Jan Brkić	14.01.2021.
1.6	Dopuna prikaza aktivnosti grupe	Karlo Frankola	14.01.2021.
2.0	Predaja druge revizije	Karlo Frankola	14.01.2021.

2. Opis projektnog zadatka

Cilj ovog projekta je razviti programsku potporu za stvaranje web aplikacije "SkillEtCooking" koja će korisniku olakšati snalaženje u kuhinji i optimizaciju iskorištavanja sastojaka koje osoba već posjeduje. Na taj način korisnik neće morati provesti cijeli dan razmišljajući što i kako napraviti nego to aplikacija radi umjesto njega.

Sustav će podržavati više vrsta korisnika:

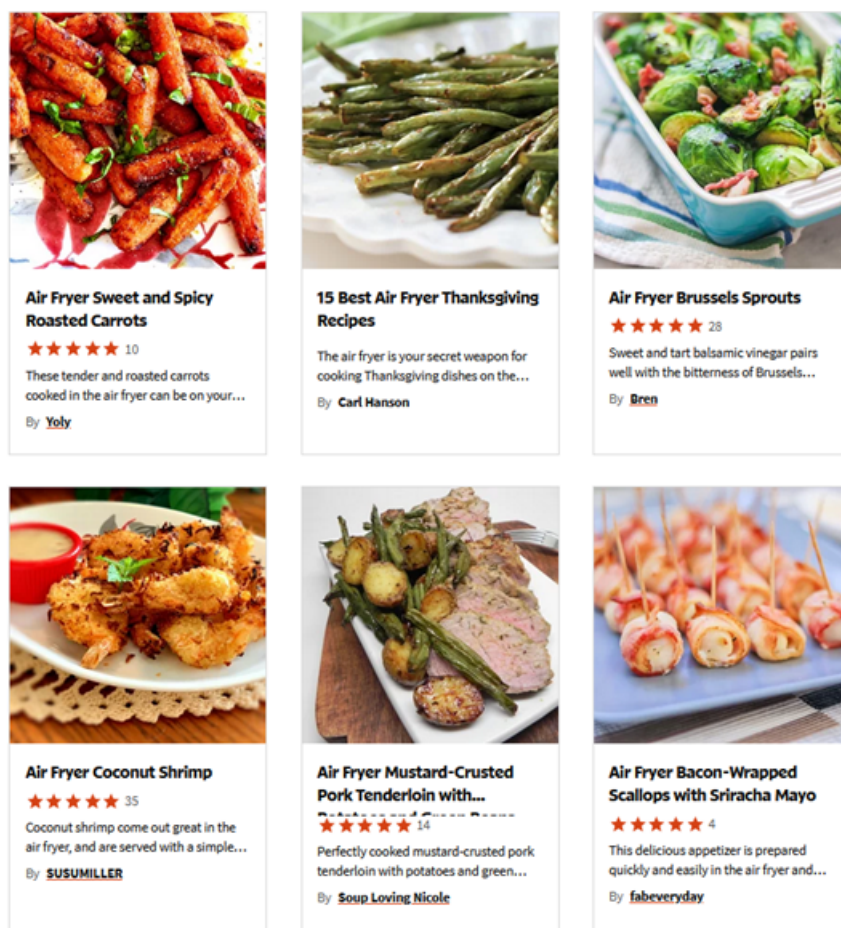
- registrirani
- neregistrirani
- moderatori.

Neregistriranim korisnicima se prilikom otvaranja aplikacije otvori početna stranica na kojoj se nalaze svi recepti (6.1). Recepti se mogu sortirati na razne načine:

- popularnost
- prosječna ocjena recepta
- oznaka "Preporučeno".

Popularnost se određuje na osnovu broja otvaranja pojedinog recepta dok se sortiranje po oznaci "Preporučeno" određuje na osnovu prosjeka popularnosti i prosječne ocjene koristeći funkciju:

- $\text{ocjena} / \text{prosjecnaOcjena} + \text{brojOtvaranja} / \text{prosjecniBrojOtvaranja}$



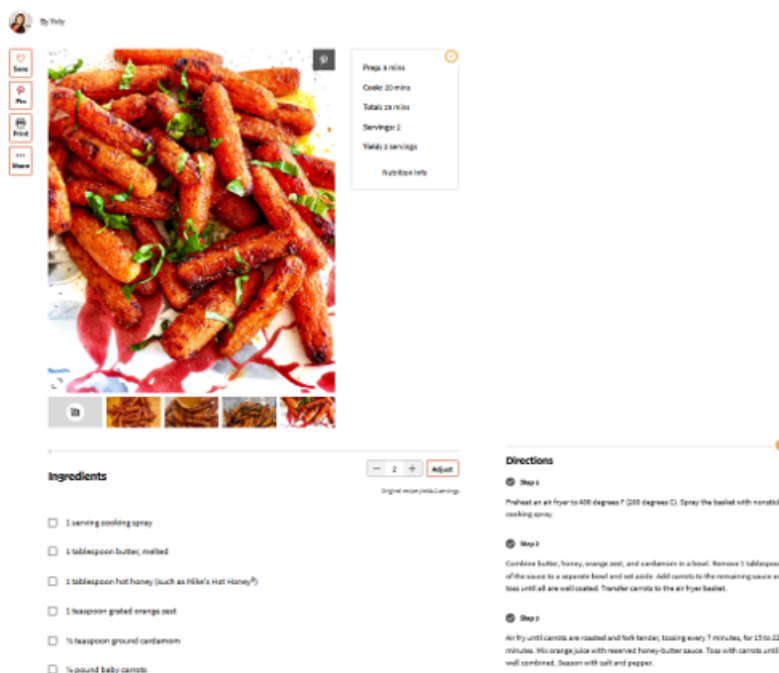
Slika 2.1: Prikaz svih recepata

Neregistrirani korisnici imaju mogućnost filtriranja recepata upisivanjem imena recepta ili sastojka/sastojaka. Nakon filtriranja skupa recepata po sastojcima, recepti se sortiraju po vrijednosti Jaccardovog indeksa sličnosti s time da najbližnji dolazi prvi te se izbacuju svi recepti čija je vrijednost indeksa sličnosti manja od praga. Za to vrijeme, ostali izbori sortiranja su onemogućeni, ali pretraživanje po naslovu je i dalje u funkciji.

Sljedeća mogućnost neregistriranih korisnika je da im se pritiskom na naziv recepta prikazuje stranica s detaljima recepta. Na toj stranici vidljivi su svi podatci vezani za recept:

- slika
- naziv
- procijenjeno vrijeme kuhanja
- sastojci i količina svakog sastojka
- koraci pripreme s kratkim opisom

- ocjena (2.2).

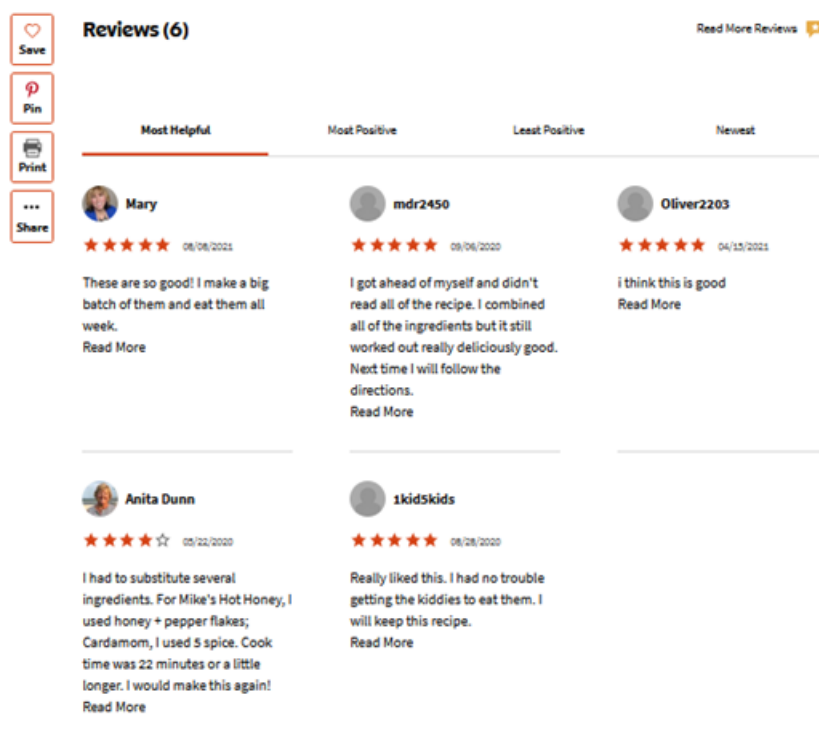


Slika 2.2: Primjer otvorenog recepta

Stranica sadrži prostor sa komentarima i ocjenama gdje je moguće vidjeti iskustva drugih korisnika što će biti izvedeno korištenjem Disqus usluge (2.3). Tu je vidljiva prva razlika između registriranog i neregistriranog korisnika, a to je da komentare i ocjene mogu ostaviti samo registrirani korisnici. Također ukoliko je autor ostavio komentar na svoj recept, komentar će biti dodatno naglašen i bolje uočljiv ostalim korisnicima.

Osim toga, stranica sadrži i mogućnost da klikom na gumb korisnik može pregledati sve ostale recepte s istim autorom kao i odabrani recept.

Registrirani korisnik se može prijaviti, a neregistrirani će imati mogućnost registracije u sustav (2.4). Prijavljeni korisnik može koristiti aplikaciju na jednak način kao i neprijavljeni korisnik, ali prijavljeni korisnik također može dodavati i uređivati vlastite recepte, uređivati vlastite podatke te koristiti prethodno objašnjenu funkcionalnost ostavljanja komentara. Pravila prilikom unosa recepta su da korisnik mora unijeti sliku jela (maksimalno 5), barem jedan sastojak i količinu te barem jedan korak pripreme.



Slika 2.3: Komentari na recept

LOG IN	SIGN UP
<div>Email Address</div> <div>Password</div> <div><input type="checkbox"/> Remember me</div> <div>LOG IN</div>	<div>LOG IN</div> <div>SIGN UP</div> <div> <div>First Name*</div> <div>Last Name*</div> </div> <div>Email Address*</div> <div>Username*</div> <div> <div>Password*</div> <div>Retype Password*</div> </div> <div>SUBMIT</div>

Slika 2.4: Prijava i registracija u sustav

Sustav koriste i moderatori koji su interni zaposlenici projekta, a njihovi računi se dodaju direktno u bazu podataka. Moderatori se prijavljuju u sustav jednako kao i ostali korisnici. Oni mogu dodati komentar na nekome receptu koji će kao i kod autora biti dodatno vizualno označen, ali ne mogu ocjenjivati recepte.

Dodatne mogućnosti moderatora su:

- brisanje komentara
- brisanje recepata
- pregled korisnika sustava.

Također sustav mora biti funkcionalan ukoliko je više korisnika prijavljeno u isto vrijeme neovisno o tome jesu li moderatori ili obični registrirani korisnici.

3. Specifikacija programske potpore

3.1 Funkcionalni zahtjevi

Dionici:

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

Aktori i njihovi funkcionalni zahtjevi:

1. Neregistrirani/neprijavljeni korisnik (inicijator) može:
 - (a) Pregledati sve recepte
 - (b) Sortirati i filtrirati recepte po raznim argumentima
 - (c) Filtrirati recepte po sastojcima
 - (d) Pretražiti recepte po imenu/autoru
 - (e) Registrirati se
2. Registrirani/prijavljeni korisnik (inicijator) može:
 - (a) Raditi sve kao i neregistrirani/neprijavljeni korisnik
 - (b) Dodati vlastiti recept (Dodani recept mora sadržavati maksimalno 5 slika, te po jedan sastojak i korak pripreme)
 - (c) Urediti vlastiti recept (obrisati/dodati sastojke, urediti i dodati korake i opis)
 - (d) Dodavati komentare na recepte
 - (e) Brisati vlastite komentare

3. Moderator (inicijator) ima mogućnost:

- (a) Brisanja i dodavanja komentara, ali ne i ocjenjivanja recepata
- (b) Brisanja recepata
- (c) Brisanja korisničkih računa
- (d) Pregleda svih korisnika

4. Baza podataka (sudionik) mora moći:

- (a) Spremati sve podatke o receptima
- (b) Spremati sve podatke o korisnicima
- (c) Spremati sve podatke o moderatorima
- (d) Izvršiti zadani upit i vratiti rezultat

3.1.1 Obrasci uporabe

Opis obrazaca uporabe

UC1 - Sortiranje recepata po popularnosti

- **Glavni sudionik:** Korisnik
- **Cilj:** Filtrirati recepte po popularnosti
- **Sudionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
 1. Prilikom učitavanja je prikazana početna stranica na kojoj se nalazi lista recepata
 2. Korisnik na karti odabire način sortiranja: prema popularnosti
 3. Prikazuju se recepti sortirani silazno prema broju otvaranja

UC2 - Sortiranje recepata prema prosječnoj ocjeni

- **Glavni sudionik:** Korisnik
- **Cilj:** Filtrirati recepte prema prosječnoj ocjeni
- **Sudionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
 1. Prilikom učitavanja je prikazana početna stranica na kojoj se nalazi lista recepata
 2. Korisnik na karti odabire način sortiranja: prema prosječnoj ocjeni
 3. Prikazuju se recepti sortirani silazno prema prosječnoj ocjeni korisnika

UC3 - Sortiranje recepata prema oznaci "Preporučeno"

- **Glavni sudionik:** Korisnik
- **Cilj:** Filtrirati recepte prema oznaci "Preporučeno"
- **Sudionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
 1. Prilikom učitavanja je prikazana početna stranica na kojoj se nalazi lista recepata
 2. Korisnik na karti odabire način sortiranja: prema prosječnoj ocjeni

3. Prikazuju se recepti sortirani silazno prema oznaci “Preporučeno” koja se određuje kao funkcija popularnosti i ocjene koja je prepuštena projektantima

UC4 – Pretraživanje recepata po naslovu

- **Glavni sudionik:** Korisnik
- **Cilj:** Prikazati sve recepte koje sadrže upisani termin
- **Sudionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
 1. Prilikom učitavanja je prikazana početna stranica na kojoj se nalazi lista recepata
 2. Korisnik pretražuje određeni termin tako da upiše naslov recepta kojeg želi otvoriti
 3. Prikazuju se svi recepti koji sadrže upisani termin

UC5 – Pretraživanje recepata po sastojcima

- **Glavni sudionik:** Korisnik
- **Cilj:** Prikazati sve recepte čiji sastojci su najbližnji unesenim sastojcima
- **Sudionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
 1. Prilikom učitavanja je prikazana početna stranica na kojoj se nalazi lista recepata
 2. Korisnik upisuje sastojke koje ima pri ruci
 3. Prikazuju se recepti sortirani po vrijednosti indeksa sličnosti

UC6 – 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 te se podaci spremaju u bazu podataka
- **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 obavještava korisnika o neuspjelom upisu i vraća ga na stranicu za registraciju
 2. Korisnik mijenja potrebne podatke te završava unos ili odustaje od registracije

UC7 – Prijava u sustav

- **Glavni sudionik:** Korisnik
- **Cilj:** Dobiti pristup dodatnim funkcionalnostima poput unosa, pregledavanja i uređivanja
- **Sudionici:** Baza podataka
- **Preduvjet:** Registracija
- **Opis osnovnog tijeka:**
 1. Unos korisničkog imena i lozinke
 2. Potvrda o ispravnosti unesenih podataka
 3. Pristup korisničkim funkcijama
- **Opis mogućih odstupanja:**
 - 1.a Neispravno korisničko ime ili lozinka
 1. Sustav obavještava korisnika o neuspjelom upisu i omogućuje mu ponovno prijavu u sustav

UC8 – Pregled pojedinog recepta

- **Glavni sudionik:** Korisnik
- **Cilj:** Prikazati sve podatke vezane za recept
- **Sudionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
 1. Korisnik odabere jedan od ponuđenih recepata
 2. Otvori se stranica sa svim informacijama o receptu

UC9 – Komentiranje i ocjenjivanje recepata

- **Glavni sudionik:** Korisnik
- **Cilj:** Ostaviti komentar i ocjenu za određeni recept
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen
- **Opis osnovnog tijeka:**
 1. Korisnik napiše recenziju i ocijeni recept
 2. Ocjena i osvrt se pohranjuju u bazu podataka i ažurira se prosječna ocjena recepta

UC10 – Pregled svih recepata određenog autora

- **Glavni sudionik:** Korisnik
- **Cilj:** Prikazati sve recepte određenog autora
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik mora biti na stranici recepta
- **Opis osnovnog tijeka:**
 1. Korisnik se nalazi na stranici recepta
 2. Korisnik odabire opciju “Prikaži sve recepte autora“
 3. Prikazuju se svi recepti autora uključujući i odabrani recept

UC11 – Dodavanje recepta

- **Glavni sudionik:** Korisnik
- **Cilj:** Dodati novi recept
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen
- **Opis osnovnog tijeka:**
 1. Korisnik odabire opciju za dodavanje novog recepta
 2. Korisnik prilaže maksimalno 5 slika jela, sastojke (makar 1) i količinu te korake pripreme (makar 1)
 3. Ukoliko je korisnik zadovoljio uvjete, podaci se spremaju u bazu podataka te su vidljivi u aplikaciji
- **Opis mogućih odstupanja:**
 - 2.a Korisnik nije priložio niti jedan sastojak ili jedan korak pripreme
 1. Sustav obavještava korisnika o uvjetima koji nisu zadovoljeni
 2. Korisnik dodaje potrebne podatke te završava unos ili odustaje od dodavanja recepta

UC12 – pregled vlastitih recepata

- **Glavni sudionik:** Korisnik
- **Cilj:** Prikazati sve vlastite recepte
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik mora biti prijavljen
- **Opis osnovnog tijeka:**
 1. Korisnik odabire opciju “Pregledaj recepte”
 2. Korisniku se prikazuju svi recepti kojima je on autor

UC13 – Uređivanje recepta

- **Glavni sudionik:** Korisnik
- **Cilj:** Urediti željeni recept
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je autor recepta
- **Opis osnovnog tijeka:**
 1. Korisnik odabire opciju uređivanja recepta
 2. Korisnik unosi željene podatke
 3. Ukoliko su uvjeti zadovoljeni, korisnik prima obavijest o uspješnom uređivanju recepta
- **Opis mogućih odstupanja:**
 - 2.a Korisnik je prilikom uređivanja uklonio sve sastojke ili korake pripreme
 1. Sustav obavještava korisnika o uvjetima koji nisu zadovoljeni
 2. Korisnik dodaje potrebne podatke te završava izmjene ili odustaje od uređivanja recepta te recept ostaje nepromijenjen

UC14 – Dodavanje komentara moderatora

- **Glavni sudionik:** Korisnik
- **Cilj:** Dodati komentar na određeni recept
- **Sudionici:** Baza podataka
- **Preduvjet:** Moderator mora biti prijavljen
- **Opis osnovnog tijeka:**
 1. Moderator ostavlja komentar koji je u konačnici vizualno dodatno naglašen
 2. Komentar se pohranjuje u bazu podataka

UC15 – Brisanje komentara

- **Glavni sudionik:** Korisnik
- **Cilj:** Obrisati komentare koji nisu primijenjeni
- **Sudionici:** Baza podataka
- **Preduvjet:** Moderator mora biti prijavljen
- **Opis osnovnog tijeka:**
 1. Moderator odabire željeni komentar za određeni recept
 2. Klikom na komentar prikaže mu se opcija “Izbriši”
 3. Komentar se uklanja iz baze podataka

UC16 – Brisanje vlastitih komentara

- **Glavni sudionik:** Korisnik
- **Cilj:** Autor želi izbrisati svoj komentar
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik mora biti prijavljen
- **Opis osnovnog tijeka:**
 1. Korisnik na stranici recepta odabire vlastiti komentar
 2. Klikom na komentar prikaže mu se opcija “Izbriši”
 3. Komentar se uklanja iz baze podataka

UC17 - Brisanje recepata

- **Glavni sudionik:** Moderator
- **Cilj:** Obrisati željene recepte
- **Sudionici:** Baza podataka
- **Preduvjet:** Moderator mora biti prijavljen
- **Opis osnovnog tijeka:**
 1. Moderator odabire željeni recept
 2. Klikom na opciju “Izbriši” recept se uklanja iz baze podataka

UC18 - Brisanje vlastitih recepata

- **Glavni sudionik:** Korisnik
- **Cilj:** Autor želi obrisati svoj recept
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik mora biti prijavljen
- **Opis osnovnog tijeka:**

1. Korisnik odabire recept kojemu je on autor
2. Klikom na opciju "Izbriši" recept se uklanja iz baze podataka

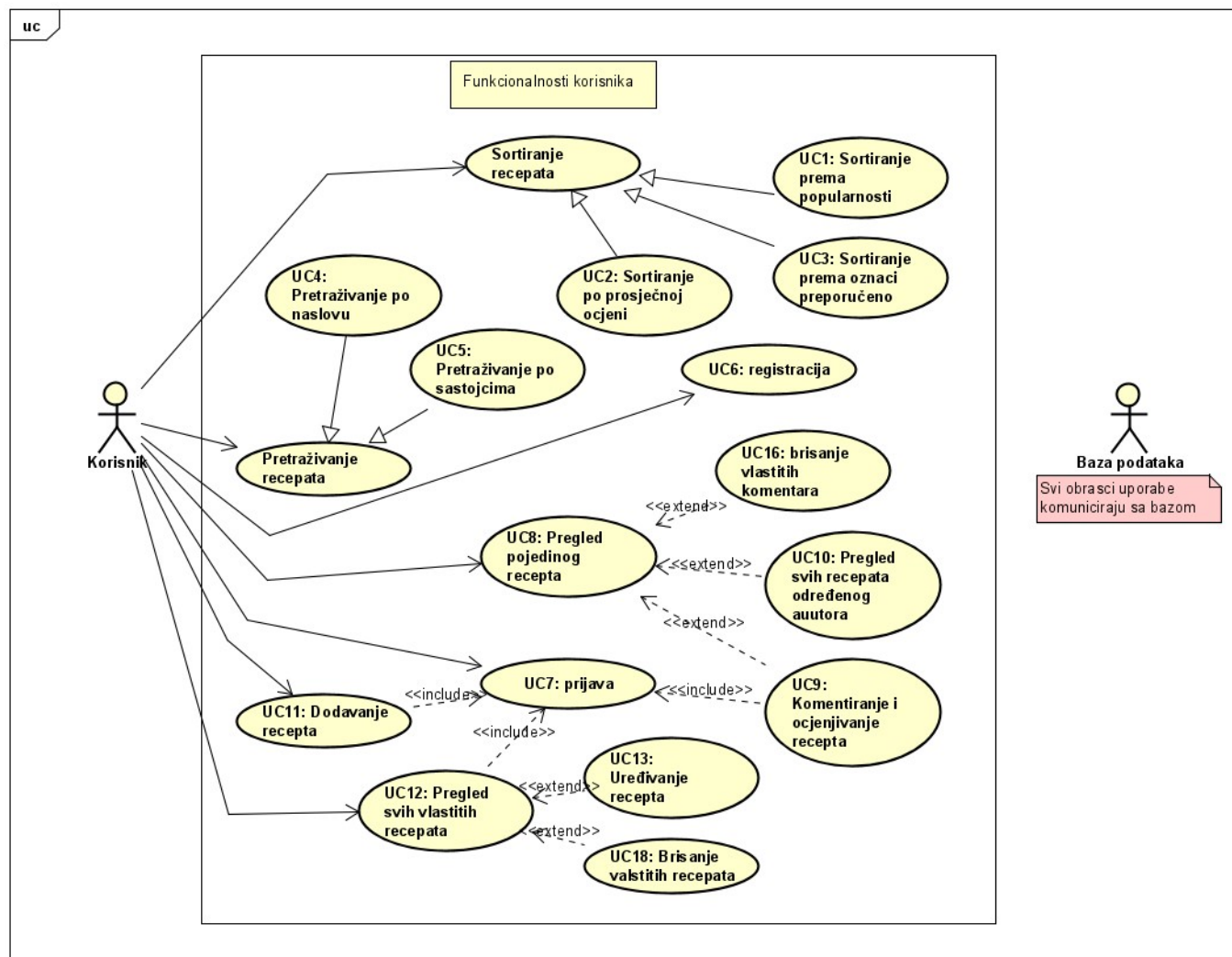
UC19 - Pregled korisnika sustava

- **Glavni sudionik:** Moderator
- **Cilj:** Pregledati registrirane korisnike
- **Sudionici:** Baza podataka
- **Preduvjet:** Moderator mora biti prijavljen
- **Opis osnovnog tijeka:**
 1. Moderator odabire opciju pregledavanja korisnika
 2. Prikaže se lista svih ispravno registriranih korisnika s osobnim podacima

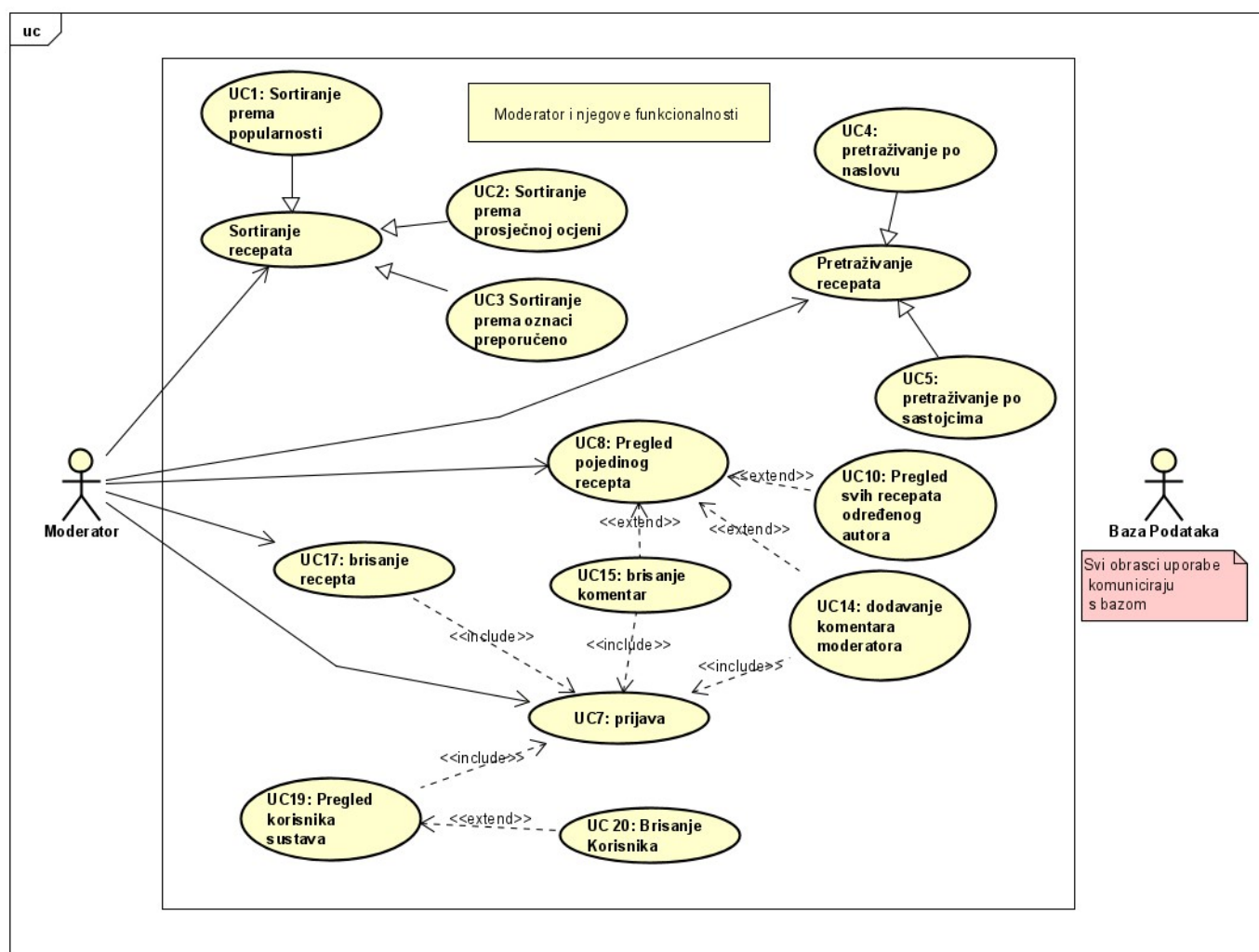
UC20 - Brisanje korisnika

- **Glavni sudionik:** Moderator
- **Cilj:** Obrisati korisnika
- **Sudionici:** Baza podataka
- **Preduvjet:** Moderator mora biti prijavljen
- **Opis osnovnog tijeka:**
 1. Moderator odabire opciju uklanjanja korisnika
 2. Moderator pronalazi željenog korisnika
 3. Moderator uklanja željenog korisnika i njegove podatke iz baze podataka

Dijagrami obrazaca uporabe



Slika 3.1: Dijagram obrasca uporabe, funkcionalnost korisnika

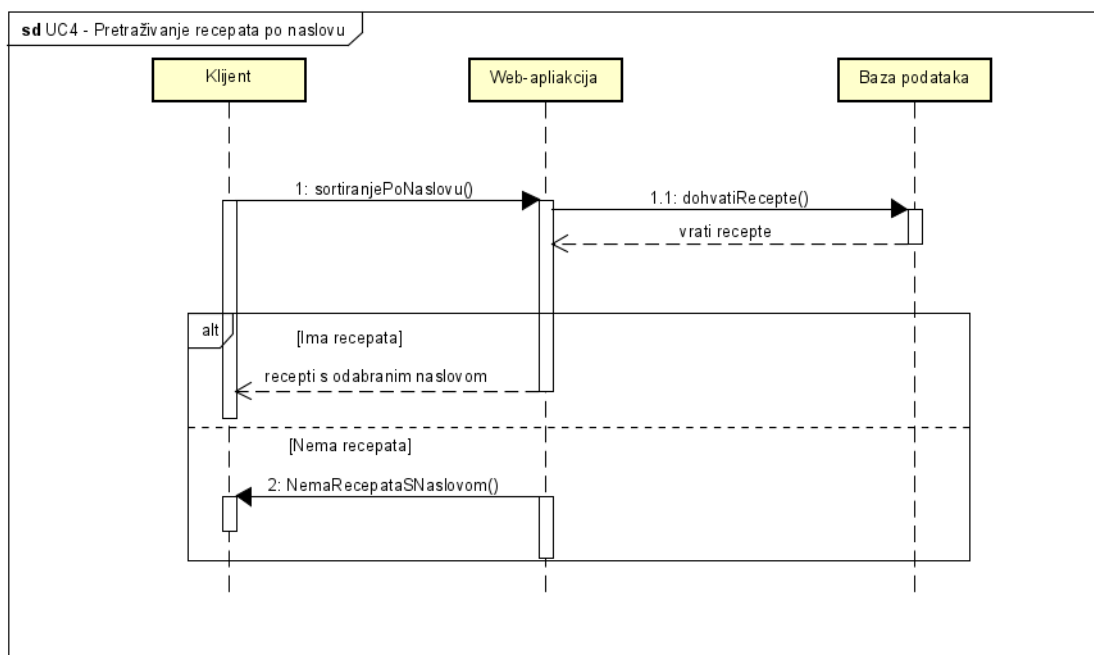


Slika 3.2: Dijagram obrasca uporabe, funkcionalnost moderatora

3.1.2 Sekvencijski dijagrami

Obrazac uporabe UC4 - Pretraživanje recepata po naslovu

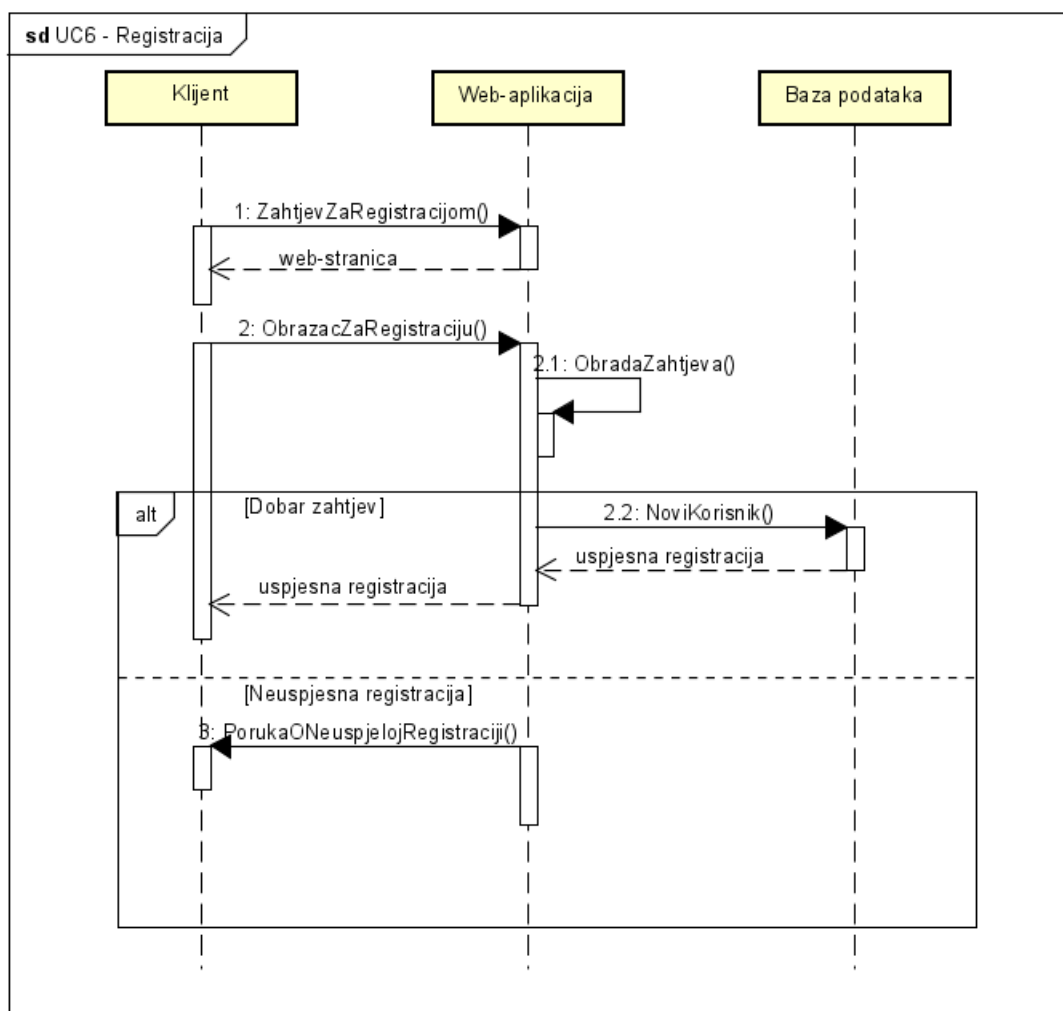
Klijent na početnoj stranici ima uvid u sve recepte, ali ima i mogućnost filtriranja tih recepata. Jedno od mogućnosti filtriranja jest filtriranje recepata po naslovu. Klijent upisuje naslov ili dio naslova kojim želi naći određeni recept te pritisne tipku pretraži. Pritiskom na tipku, web-aplikacija šalje upit bazi podataka. Baza podataka pretražuje sve recepte i traži one recepte koji imaju sve riječi ili samo jednu riječ upisanu u tražilicu. Ukoliko baza podataka ne pronade niti jedan recept nakon pretraživanja, klijent će biti obaviješten o tome. U suprotnome, baza podataka vraća web-aplikaciji sve recepte koje je pronašla nakon pretraživanja. Web-aplikacija potom ispisuje sve recepte klijentu te se time završava pretraživanje.



Slika 3.3: Sekvencijski dijagram za UC4

Obrazac uporabe UC6 - Registracija

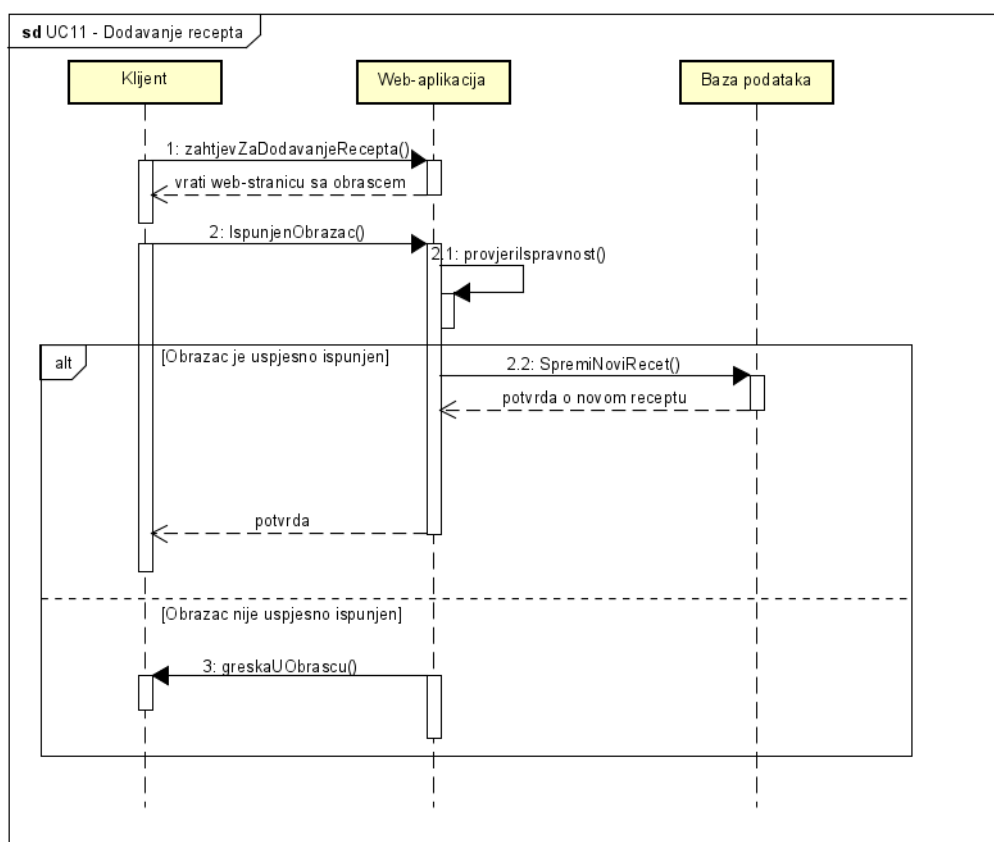
Klijent ima mogućnost registracije. Registracija je postupak stvaranja računa po prvi puta u web-aplikaciji. Klijent pritiskom na gumb "prijava" otvara novu web-stranicu gdje mu je omogućena prijava. Osim prijave, klijentu je omogućena i registracija, ukoliko nema već postojeći račun. Klikom na registraciju, pojavljuje se obrazac za stvaranje računa gdje je potrebno unijeti podatke koji će se potom spremati u bazu podataka. Kako bi obrazac bio uspješno napisan potrebno je: ispuniti sva polja, upisati nepostojeće korisničko ime i lozinku koja se pridržava propisanih pravila. Ako je klijent uspješno ispunio zahtjev, njegov račun će se spremiti u bazu podataka i dobit će obavijest o uspješnoj registraciji. Ako klijent nije uspješno ispunio zahtjev, dobit će poruku o odgovarajućoj pogrešci.



Slika 3.4: Sekvencijski dijagram za UC6

Obrazac uporabe UC11 - Dodavanje recepta

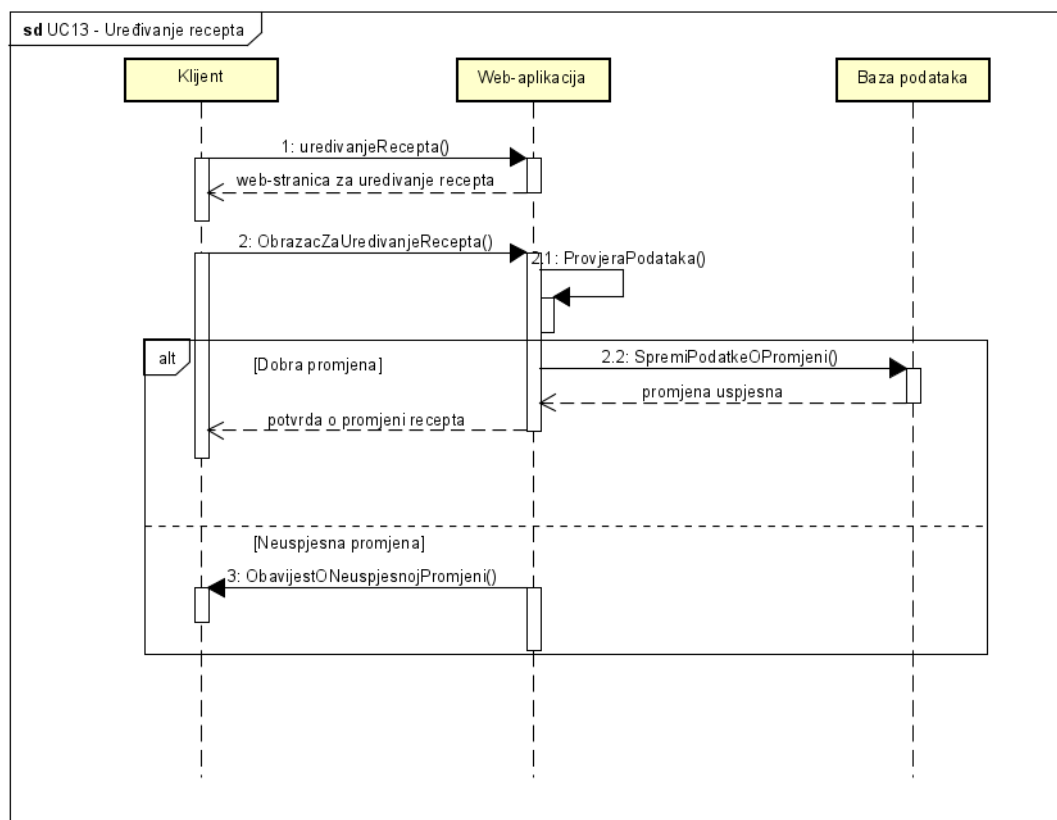
Registrirani korisnik (nadalje klijent) ima mogućnost dodavanja recepta. Dodavanje recepta je vidljivo na profilu klijenta. Odabirom usluge dodavanja recepta, otvara se web-stranica sa obrascem za dodavanje recepta. U obrascu je potrebno popuniti označena polja, te priložiti do maksimalno 5 slika. Ako je klijent uspješno popunio obrazac, recept se dodaje u bazu podataka i klijent dobiva obavijest o uspješnome dodavanju recepta. Ako klijent nije uspješno popunio obrazac, ispisuje mu se odgovarajuća pogreška, ali se postojeći podaci ne brišu te klijent ima mogućnost ispravljanja pogreške.



Slika 3.5: Sekvencijski dijagram za UC11

Obrazac uporabe UC13 - Uređivanje recepta

Registrirani korisnik (nadalje klijent) ima mogućnost uređivanja vlastitog recepta. Uređivanje recepta je vidljivo na stranici klijentovog recepta. Odabirom usluge uređivanja recepta, otvara se web-stranica s već postojećim obrascem. Podaci u već postojećem obrascu izvučeni su iz baze podataka. Klijent može dodavati ili brisati podatke u obrascu. Klijent mora poštovati pravila za uređivanje obrasca kao i kod dodavanja recepta. Ako je klijent napravio prihvatljivu promjenu, promjena će se spremiti u bazu podataka i klijent dobiva obavijest o uspješnome uređivanju recepta. Ako je klijent napravio neprihvatljivu promjenu, promjena neće biti spremljena i klijent će dobiti obavijest o pogrešci.



Slika 3.6: Sekvencijski dijagram za UC13

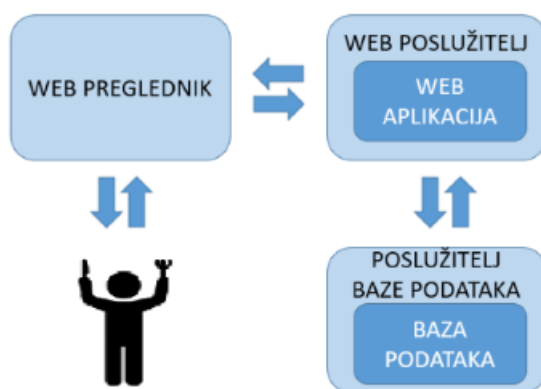
3.2 Ostali zahtjevi

- *Sustav treba omogućiti rad više korisnika u stvarnom vremenu*
- *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*
- *Neispravno korištenje korisničkog sučelja ne smije narušiti funkcionalnost i rad sustava*
- *Sustav treba biti jednostavan za korištenje, korisnici se moraju znati koristiti sučeljem bez opširnih uputa*
- *Nadogradnja sustava ne smije narušavati postojeće funkcionalnosti 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.*

4. Arhitektura i dizajn sustava

Arhitektura se može podijeliti na tri podsustava:

- Web poslužitelj
- Web aplikacija
- Baza podataka



Slika 4.1: Arhitektura sustava

Web preglednik je program koji korisniku omogućuje pregled web-stranica i multimedijalnih sadržaja vezanih uz njih. Svaki internetski preglednik je prevođitelj. Dakle, stranica je pisana u kodu koji preglednik nakon toga interpretira kao nešto svakome razumljivo. Korisnik putem web preglednika šalje zahtjev web poslužitelju.

Web poslužitelj osnova je rada web aplikacije. Njegova primarna zadaća je komunikacija klijenta s aplikacijom. Komunikacija se odvija preko HTTP (engl. Hyper Text Transfer Protocol) protokola, što je protokol u prijenosu informacija na webu. Poslužitelj je onaj koji pokreće web aplikaciju te joj prosljeđuje zahtjev.

Korisnik koristi web aplikaciju za obrađivanje željenih zahtjeva. Web aplikacija obrađuje zahtjev te ovisno o zahtjevu, pristupa bazi podataka nakon čega preko poslužitelja vraća korisniku odgovor u obliku HTML dokumenta vidljivog u web pregledniku.

Programski jezik kojeg smo odabrali za izradu naše web aplikacije je Java zajedno sa Spring Boot radnim okvirom. Odabrano razvojno okruženje je Visual Studio Code. Arhitektura sustava temeljiti će se na MVC (Model-View-Controller) konceptu.

Karakteristika MVC koncepta je nezavisan razvoj pojedinih dijelova aplikacije što za posljedicu ima jednostavnije ispitivanje kao i jednostavno razvijanje i dodavanje novih svojstava u sustav.

MVC koncept sastoji se od:

- **Model** - Središnja komponenta sustava. Predstavlja dinamičke strukture podataka, neovisne o korisničkom sučelju. Izravno upravlja podacima, logikom i pravilima aplikacije. Također prima ulazne podatke od Controllera.
- **View** - Bilo kakav prikaz podataka, poput grafa. Mogući su različiti prikazi iste informacije poput grafičkog ili tabličnog prikaza podataka.
- **Controller** - Prima ulaze i prilagođava ih za prosljeđivanje Modelu ili Viewu. Upravlja korisničkim zahtjevima i temeljem njih izvodi daljnju interakciju s ostalim elementima sustava.

4.1 Baza podataka

Za potrebe našeg 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. Zadaća baze podataka je brza i jednostavna pohrana, izmjena i dohvat podataka za daljnju obradu. Baza podataka ove aplikacije sastoji se od sljedećih entiteta:

- Users
- Roles
- Images
- Ingredients
- Recipes
- Ratings
- Recipe_steps

4.1.1 Opis tablica

Users Ovaj entitet sadržava sve važne informacije o korisniku aplikacije. Sadrži attribute: `first_name`, `last_name`, `date_of_birth`, `email`, `id`, `password.hash`, `username` `role_id`. Ovaj entitet u vezi je *Many-to-One* s entitetom Roles preko atributa `role_id`, u vezi *One-to-Many* s entitetom Recipes preko atributa `created_by` te je u vezi *One-to-Many* s entitetom Ratings preko atributa `user_id`.

Users		
<code>id</code>	SERIAL	jedinstveni identifikator korisnika
<code>first_name</code>	VARCHAR	ime korisnika
<code>last_name</code>	VARCHAR	prezime korisnika
<code>date_of_birth</code>	DATE	datum rođenja korisnika
<code>email</code>	VARCHAR	korisnikov email
<code>password.hash</code>	VARCHAR	korisnikova lozinka
<code>username</code>	VARCHAR	korisničko ime
<code>role_id</code>	INT	strani ključ na entitet Roles

Recipes Ovaj entitet sadržava sve važne informacije o receptu unutar aplikacije. Sadrži attribute: `id`, `popularity`, `title`, `created_at`, `last_updated_at`, `recipe_description`, `estimated_time` i `created_by`. Ovaj entitet u vezi je *One-to-Many* s entitetom `Ingredients` preko atributa `recipe_id`, u vezi *One-to-Many* s entitetom `Images` preko atributa `recipe_id`, u vezi *Many-to-One* s entitetom `Users` preko atributa `created_by`, u vezi *One-to-Many* s entitetom `Ratings` preko atributa `recipe_id` te je u vezi *One-to-Many* s entitetom `Recipe_steps` preko atributa `recipe_id`.

Recipes		
<code>id</code>	SERIAL	jedinstveni identifikator recepta
<code>popularity</code>	INT	popularnost recepta
<code>title</code>	VARCHAR	naslov recepta
<code>created_at</code>	TIMESTAMP	datum nastanka recepta
<code>last_updated_at</code>	TIMESTAMP	datum zadnje izmjene recepta
<code>recipe_description</code>	VARCHAR	kratki opis recepta
<code>estimated_time</code>	INT	procijenjeno vrijeme za pripremu
<code>created_by</code>	INT	strani ključ na entitet <code>Users</code>

Ingredients Ovaj entitet predstavlja jedan sastojak unutar nekog recepta. Sadrži attribute: `id`, `ingredient_name`, `ingredient_measure`, `ingredient_quantity` i `ingredient_order`. Ovaj entitet u vezi je *Many-to-One* s entitetom `Recipes` preko atributa `recipe_id`.

Ingredients		
<code>id</code>	SERIAL	jedinstveni identifikator sastojka
<code>ingredient_name</code>	VARCHAR	ime sastojka
<code>ingredient_quantity</code>	INT	količina sastojka
<code>ingredient_measure</code>	VARCHAR	ime mjere sastojka(ml, g, kg)
<code>ingredient_order</code>	INT	redni broj sastojka unutar recepta

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

Ingredients		
recipe_id	INT	strani ključ na entitet Recipes

Ratings Ovaj entitet predstavlja ocjenu na određeni recept. Sadrži attribute: id, rating_value, user_id i recipe_id. Ovaj entitet u vezi je *Many-to-One* s entitetom Recipes preko atributa recipe_id te je u vezi *Many-to-One* s entitetom Users preko atributa user_id

Ratings		
id	SERIAL	jedinstveni identifikator ocjene
rating_value	INT	vrijednost ocjene
user_id	INT	strani ključ na entitet Users
recipe_id	INT	strani ključ na entitet Recipes

Images Ovaj entitet predstavlja jednu sliku koja se dodaje prilikom dodavanja recepta. Sadrži attribute: id, image_data, image_order i recipe_id. Ovaj entitet u vezi je *Many-to-One* s entitetom Recipes preko atributa recipe_id.

Images		
id	SERIAL	jedinstveni identifikator slike
image_data	BYTEA	prostor koji zauzima slika u memoriji
image_order	INT	poredak slike prilikom dodavanja
recipe_id	INT	strani ključ na entitet Recipes

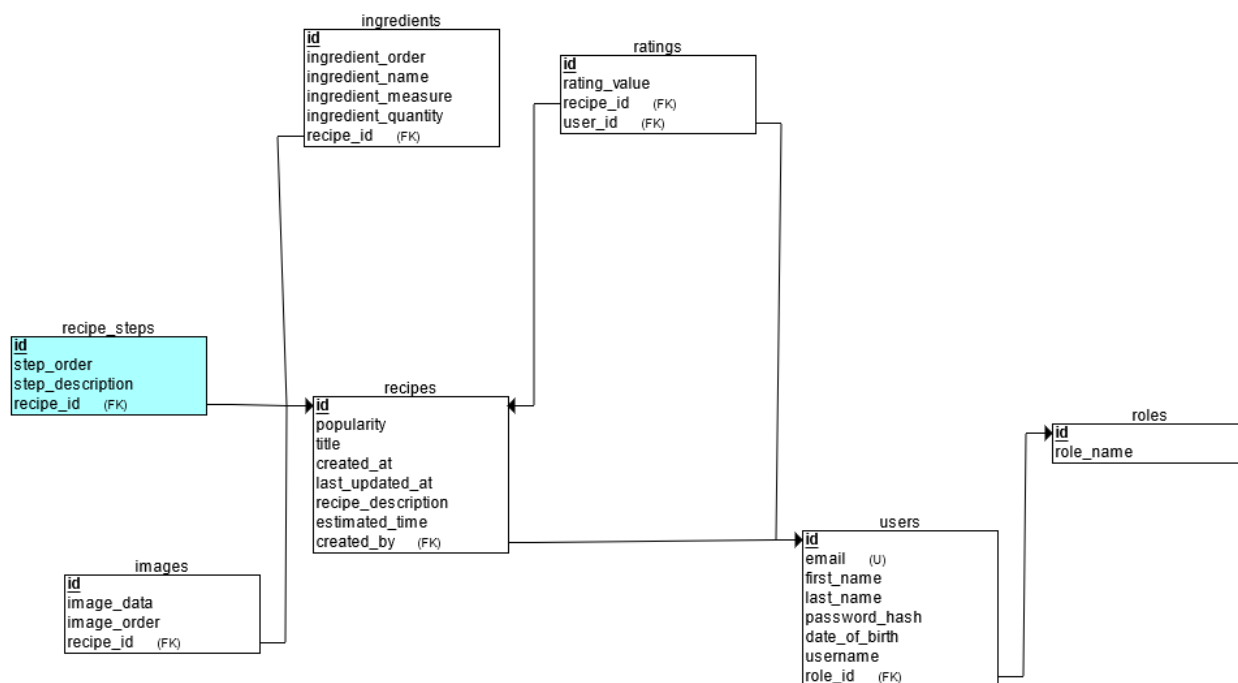
Recipe_steps Ovaj entitet predstavlja jedan korak pripreme u receptu. Sadrži attribute: id, step_order, step_description i recipe_id. Ovaj entitet u vezi je *Many-to-One* s entitetom Recipes preko atributa recipe_id.

Recipe_steps		
id	SERIAL	jedinstveni identifikator koraka pripreme
step_order	INT	redni broj koraka pripreme
step_description	VARCHAR	kratki opis koraka pripreme
recipe_id	INT	strani ključ na entitet Recipes

Roles Ovaj entitet predstavlja ulogu korisnika u sustavu. Sadrži attribute: id i role_name. Ovaj entitet u vezi je *One-to-Many* s entitetom Users preko atributa role_id.

Roles		
id	SERIAL	jedinstveni identifikator uloge
role_name	VARCHAR	ime uloge (korisnik, moderator)

4.1.2 Dijagram baze podataka

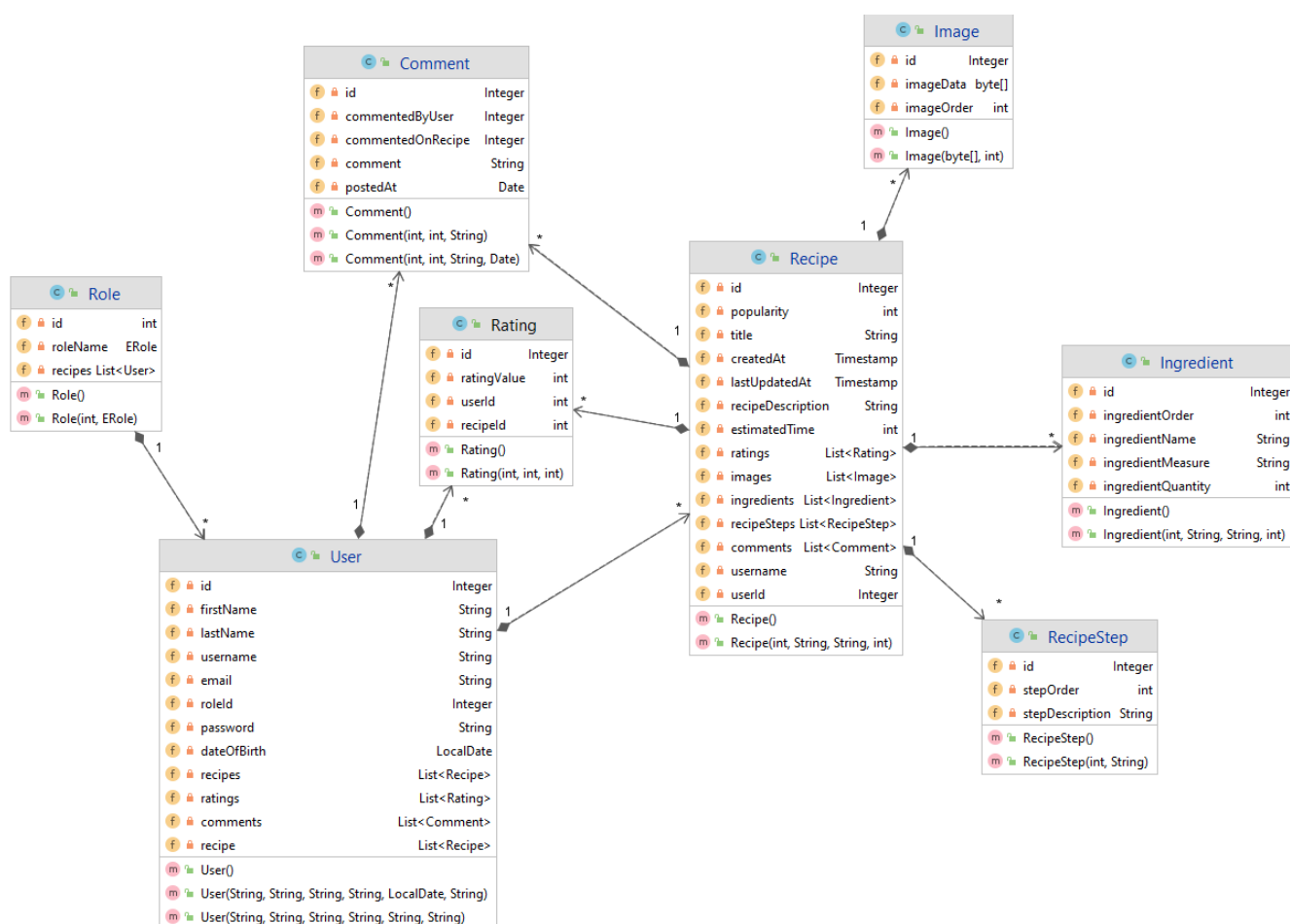


Slika 4.2: E-R dijagram baze podataka

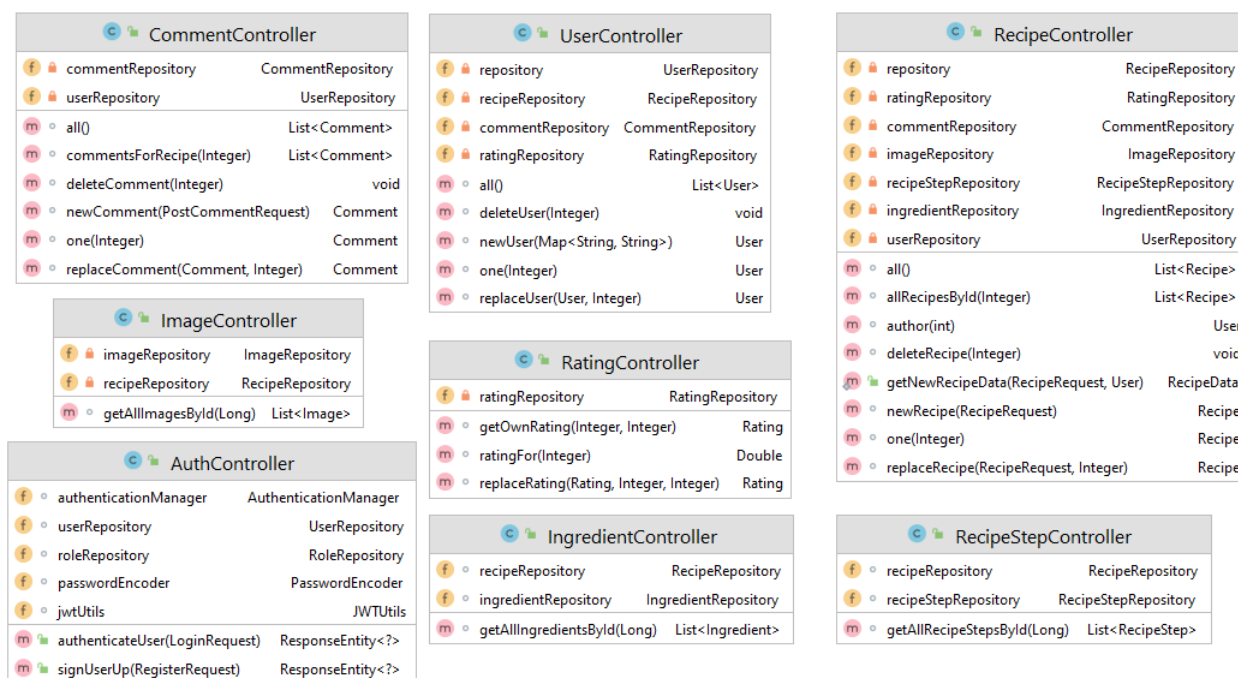
4.2 Dijagram razreda

Na slikama 4.3, 4.4 i 4.5 prikazani su razredi koji pripadaju backend dijelu MVC arhitekture. Razredi prikazani na 4.3 su Model razredi koji su implementacije entiteta baze. Razredi prikazani na slici 4.3 su Controller razredi te oni implementiraju funkcionalnosti u ovisnosti na HTTP upit i vraćaju valjani odgovor u JSON formatu. Bazi se pristupa preko Interfacea koji nasljeđuju JpaRepository.

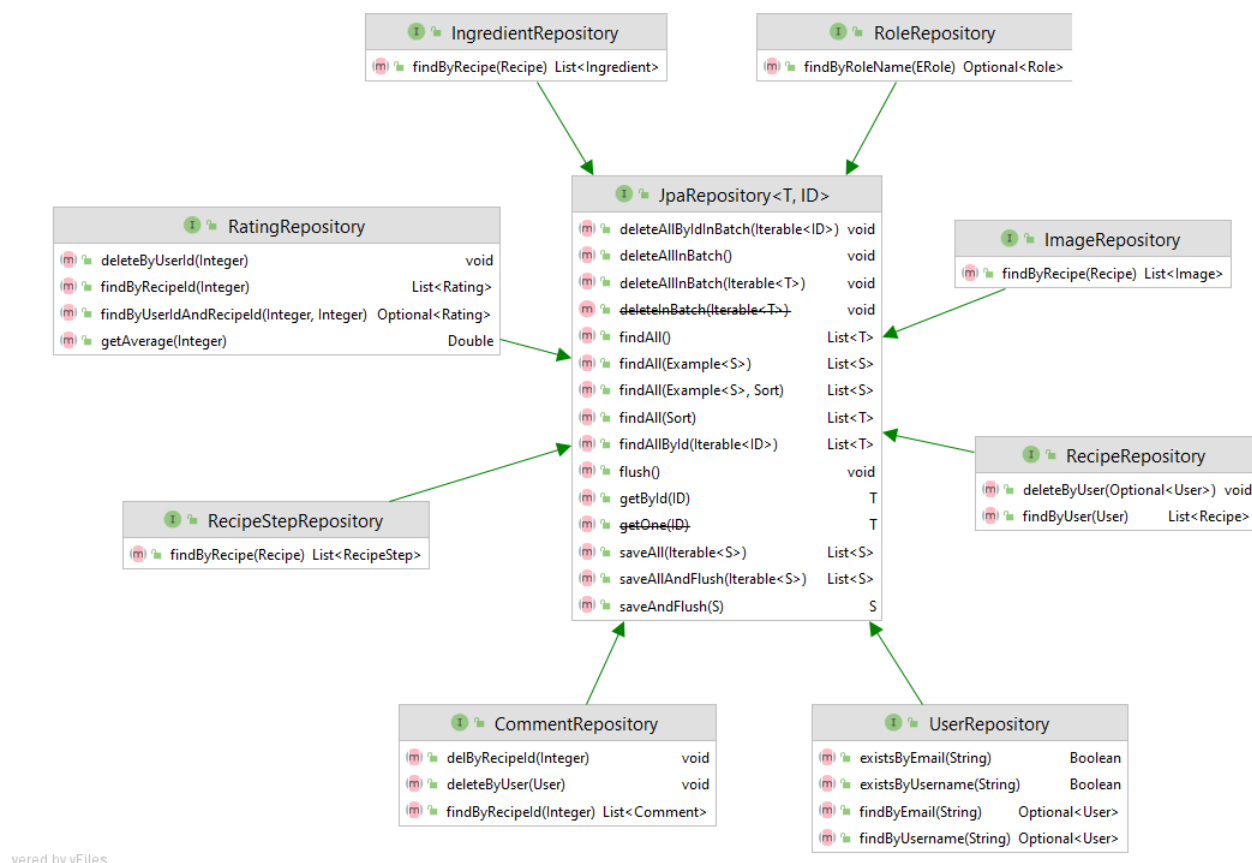
Zbog lakše organizacije razredi su podijeljeni na tri logičke jedinice te su prikazani odvojenim dijagramima. Iz naziva i tipova atributa mogu se zaključiti ostale ovisnosti među razredima.



Slika 4.3: Dijagram razreda - Models



Slika 4.4: Dijagram razreda - Controllers

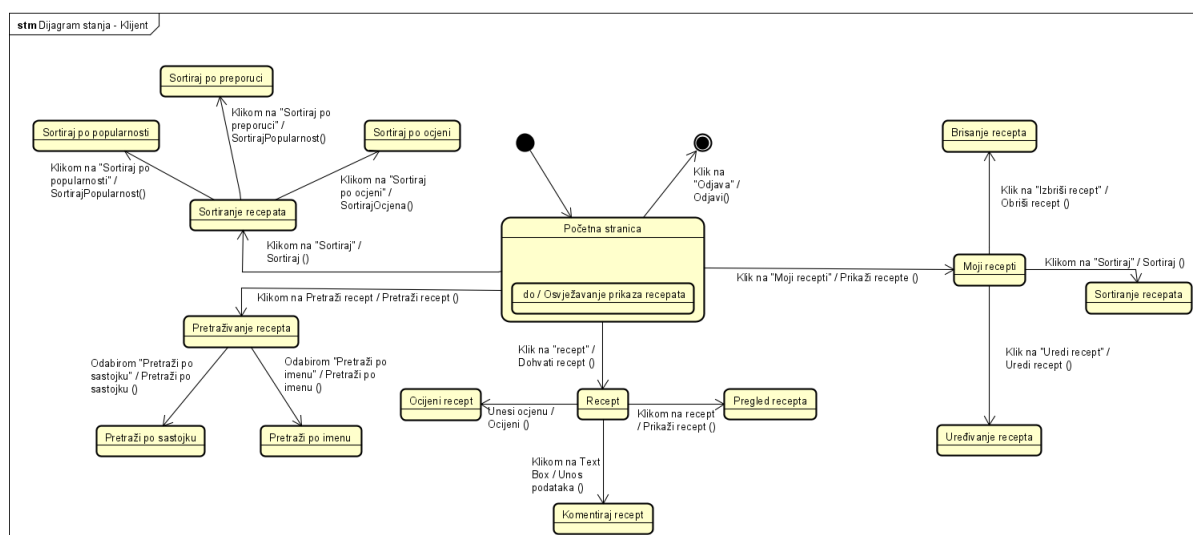


Slika 4.5: Dijagram razreda - Repositories

4.3 Dijagram stanja

Dijagram stanja prikazuje stanja objekta te prijelaze iz jednog stanja u drugo temeljene na događajima. Na slici 4.6 vidljiv je dijagram stanja registriranog korisnika. Korisnikove mogućnosti, osim pregleda svih recepata koji se prikazuju pri učitavanju stranice, su i pregled vlastitih recepata. Klikom na Sortiraj, korisnik može sortirati recepte po ocjeni, preporuci i popularnosti te mu je omogućen lakši pregled istih.

Na istoj stranici, korisnik može pretražiti recepte te se tu otvaraju dvije mogućnosti, pretraživanje po sastojku te pretraživanje po imenu sve u svrhu lakšeg pretraživanja. Klijent izlistane recepte može ocijeniti, pregledati i komentirati. Klikom na "Moji recepti", korisniku se prikazuju svi njegovi dosadašnji recepti. Svoje recepte korisnik može obrisati, klikom na "Izbriši recept", sortirati klikom na "Sortiraj" te urediti pojedini recept.

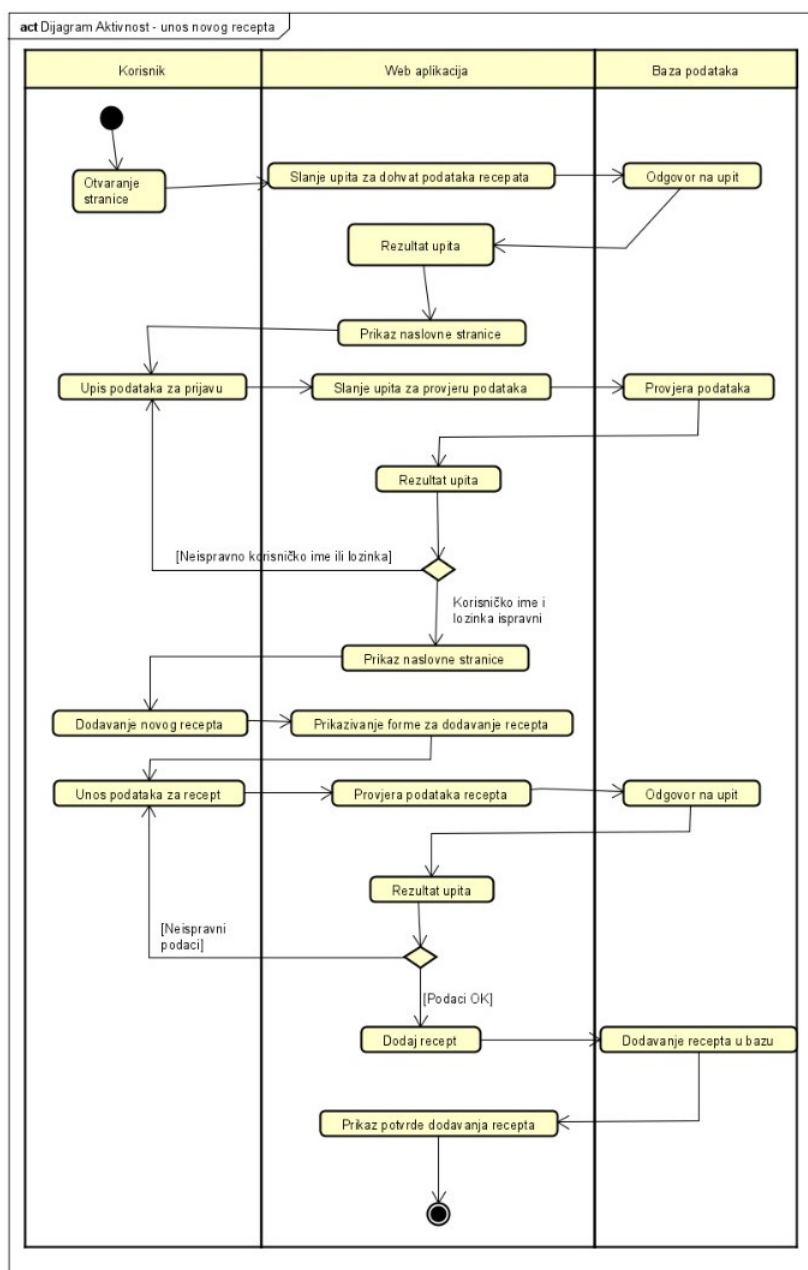


Slika 4.6: Dijagram stanja

4.4 Dijagram aktivnosti

Dijagram aktivnosti modelira ponašanja nizom akcija. Primjenjuje se za modeliranje poslovnih procesa te upravljačkog i podatkovnog toka. Na slici 4.7 prikazan je Dijagram aktivnosti za unos novog recepta. Korisnik na početku učitava stranicu na koju nije prijavljen. Neprijavljeni korisnik i dalje ima mnoge mogućnosti ali da bi dodao svoj recept on se mora prijaviti u sustav.

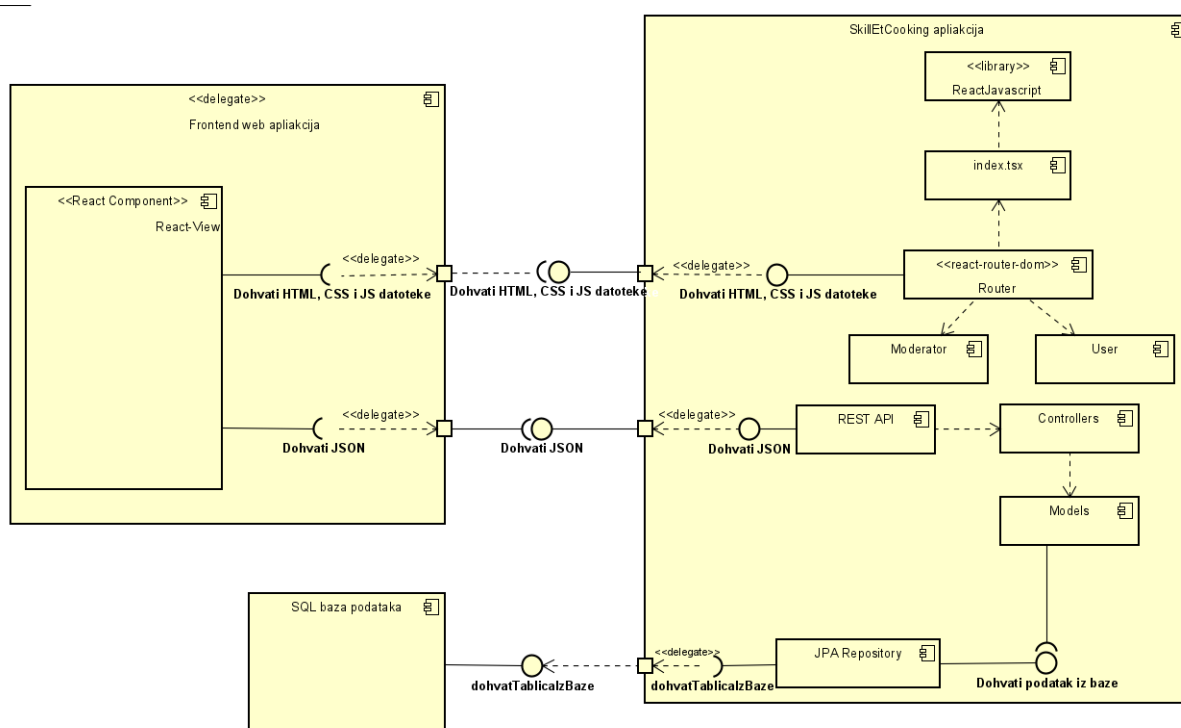
Korisnik se prijavljuje klikom na "Prijava" te se učitava stranica za prijavu. Podaci prijave se šalju Bazi podataka na validaciju te nakon njene potvrde, korisniku je omogućen daljnji pristup unosa recepta. Korisnik klikom na dodaj recept unosi podatke o receptu te pri tome mora zadovoljiti uvjete kao što su maksimalnih 5 slika koje može a i ne mora unijeti. Ujedno, potrebno je unijeti barem jedan korak pripreme te se uneseni podaci provjeravaju u Bazi podataka. Nakon zadovoljavajućeg unosa podataka recepta, šalje se poruka da je recept valjano dodan.



Slika 4.7: Dijagram aktivnosti

4.5 Dijagram komponenti

Dijagram komponenti prikazan na slici 4.8 opisuje organizaciju i međusobnu ovisnost komponenti, unutrašnje strukture i odnose prema okolini. Preko sučelja "Dohvati HTML, CSS i JS datoteke" poslužuju se datoteke za *frontend* dio web aplikacije. *Frontend* dio web aplikacije se sastoji od niza TypeScript datoteka koje predstavljaju aktore koji tim datotekama mogu pristupiti. Preko sučelja "Dohvati JSON" pristupa se REST API komponenti. REST API poslužuje podatke koje pripadaju *backend* dijelu web aplikacije. JPA Repository služi za dohvaćanje tablica iz baze podataka koristeći SQL upite.



Slika 4.8: Dijagram komponenti

5. Implementacija i korisničko sučelje

5.1 Korištene tehnologije i alati

Za potrebe komunikacije među članovima tima korištena je aplikacija WhatsApp¹. Za izradu UML dijagrama korišten je alat Astah Professional², a kao sustav za upravljanje izvornim kodom Git³. Udaljeni repozitorij projekta dostupan je na web platformi GitLab⁴.

Za razvojno okruženje odlučili smo se za Microsoft Visual Studio Code⁵. Visual Studio Code, ili kraće VSC, integrirano je razvojno okruženje (IDE) tvrtke Microsoft. Značajke VSC-a uključuju podršku za otklanjanje pogrešaka, isticanje sintakse, inteligentno dovršavanje koda, isječke, refaktoriranje koda i ugrađeni Git.

Aplikacija je napisana koristeći radni okvir Spring Boot⁶ i jezik Java⁷ za izradu *backend*a te React⁸ i jezik TypeScript⁹ za izradu *frontend*a. React je biblioteka u JavaScriptu za izradu korisničkih sučelja. Održavaju ga Meta i zajednica pojedinačnih programera i tvrtki. React se najčešće koristi kao osnova u razvoju web i mobilnih aplikacija. React se bavi samo upravljanjem stanjem i prikazivanjem tog stanja te prilikom izrade složenijih aplikacija zahtjeva korištenje dodatnih biblioteka za interakciju s API-jem. Radni okvir Spring boot je sredstvo kojim se olakšava i ubrzava izrada web aplikacija. Ima mogućnost autokonfiguracije, odlučan pristup konfiguraciji i mogućnost izrade samostalnih aplikacija. Ove značajke rade zajedno kako bi vam pružile alat koji vam omogućuje postavljanje aplikacije koja se temelji na Springu uz minimalnu konfiguraciju i postavljanje.

Baza podataka se nalazi na poslužitelju u oblaku na AWS¹⁰

¹<https://www.whatsapp.com/>

²<https://astah.net/products/astah-professional/>

³<https://git-scm.com>

⁴<https://about.gitlab.com>

⁵<https://code.visualstudio.com>

⁶<https://spring.io>

⁷<https://www.java.com/en/>

⁸<https://reactjs.org>

⁹<https://www.typescriptlang.org>

¹⁰<https://aws.amazon.com/>

5.2 Ispitivanje programskog rješenja

5.2.1 Ispitivanje komponenti

Obavljeno je ispitivanje funkcionalnosti metoda iz klase: *RecipeController.java*. Poseban naglasak na metodi dodavanja novog recepta koja prima podatke iz HTTP zahtjeva provjerava ih i priprema za spremanje u bazu. Ta metoda se zove `+getNewRecipeData(RecipeRequest, User):RecipeData` i nalazi se u razredu *RecipeController.java*. Prvi ispitni slučaj je namijenjen provjeri sigurnosti tako da se ne dopusti funkcionalnost dodavanja recepta neprijavljenim korisnicima, a u zadnjem ispitnom slučaju je demonstrirana da ukoliko neprijavljeni korisnik pristupa domeni umjesto očekivanog odgovora: *HTTP: 200 OK* dobije se odgovor *HTTP: 401 UNAUTHORIZED*. Nadalje drugi ispitni slučaj provjera je li rezultat metode u skladu s očekivanim u slučaju kada joj se pošalju ispravni podatci. Svi ostali ispitni slučajevi ispituju rubne slučajeve i slučajeve u kojima metoda ne smije dati nikakv rezultat nego mora baciti iznimku. Primjeri takvih slučajeva su: slanje zahtjeva sa pet ili jednom slikom (*oni predstavljaju rubne slučajeve jer je minimalan broj slika jedna, a maksimalan pet*), slanje zahtjeva bez ijednog sastojka, slanje zahtjeva u kojima su sastojci nepotpuni (*ne sadrže naziv, količinu ili mjernu jedinicu*), slanje zahtjeva bez ijednog koraka pripreme...

Izvorni kod testova:

```
1 package hr.fer.proinz.skilletcooking.controllers;
2
3 import static org.junit.jupiter.api.Assertions.assertEquals;
4 import static org.junit.jupiter.api.Assertions.assertNotNull;
5 import static org.junit.jupiter.api.Assertions.assertThrows;
6
7 import org.junit.jupiter.api.Test;
8 import org.springframework.beans.factory.annotation.Autowired;
9 import org.springframework.boot.test.autoconfigure.web.servlet.AutoConfigureMockMvc;
10 import org.springframework.boot.test.context.SpringBootTest;
11 import org.springframework.http.MediaType;
12 import org.springframework.mock.web.MockMultipartFile;
13 import org.springframework.test.web.servlet.MockMvc;
14 import org.springframework.test.web.servlet.request.MockMvcRequestBuilders;
15
16 import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.post;
17 import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.status;
18
19 import java.io.IOException;
20
21 import javax.transaction.Transactional;
22
23 import com.google.gson.Gson;
24
25 import org.springframework.web.multipart.MultipartFile;
26
27 import hr.fer.proinz.skilletcooking.controllers.RecipeController.RecipeData;
28 import hr.fer.proinz.skilletcooking.models.User;
29 import hr.fer.proinz.skilletcooking.payload.Request.RecipeRequest;
30 import hr.fer.proinz.skilletcooking.repository.UserRepository;
```

```
31
32 @SpringBootTest
33 @AutoConfigureMockMvc
34 public class RecipeControllerTest {
35
36     @Autowired
37     private MockMvc mockMvc;
38
39     @Autowired
40     private UserRepository userRepository;
41
42     @Test
43     void testPostDeniedWhenLoggedIn() throws Exception {
44         mockMvc.perform(post("/api/recipes/").andExpect(status().is4xxClientError()));
45     }
46
47     @Test
48     void testGetNewRecipeDataWhenPassingNoData() {
49         assertThrows(IllegalArgumentException.class,
50             () -> RecipeController.getNewRecipeData(new RecipeRequest(), new User()));
51     }
52
53     @Test
54     void testGetNewRecipeDataWhenPassingValidData() throws IOException {
55         RecipeRequest request = new RecipeRequest();
56         request.setTitle("New");
57         request.setRecipeDescription("New recipe");
58         request.setEstimatedTime(100);
59         request.setDescription(new String[] { "First step" });
60         request.setName(new String[] { "Flour", "Milk" });
61         request.setQuantity(new int[] { 1, 2 });
62         request.setMeasure(new String[] { "kg", "l" });
63         request.setUserId(3);
64         request.setPictures(
65             new MultipartFile[] { new MockMultipartFile("keksi", "keksi.png", MediaType.IMAGE.PNG.VALUE,
66                 this.getClass().getResourceAsStream("images/keksi.png").readAllBytes()) });
67         User user = userRepository.findById(request.getUserId()).get();
68         RecipeData data = RecipeController.getNewRecipeData(request, user);
69         assertNotNull(data);
70         assertEquals(data.getRecipe().getTitle(), "New");
71         assertEquals(data.getRecipe().getUser(), user);
72         assertEquals(true,
73             data.getIngredients().stream().allMatch(ingredient -> ingredient.getRecipe().equals(data.getRecipe())));
74         assertEquals(true,
75             data.getImages().stream().allMatch(image -> image.getRecipe().equals(data.getRecipe())));
76         assertEquals(true,
77             data.getSteps().stream().allMatch(step -> step.getRecipe().equals(data.getRecipe())));
78     }
79
80     @Test
81     void testGetNewRecipeDataWhenPassingTooManyImages() throws IOException {
82         RecipeRequest request = new RecipeRequest();
83         request.setTitle("New");
84         request.setRecipeDescription("New recipe");
85         request.setEstimatedTime(100);
86         request.setDescription(new String[] { "First step" });
87         request.setName(new String[] { "Flour" });
88         request.setQuantity(new int[] { 1 });
89         request.setMeasure(new String[] { "kg" });
90         request.setUserId(3);
91         MultipartFile[] pictures = new MultipartFile[6];
92         for (int i = 0; i < pictures.length; i++)
93             pictures[i] = new MockMultipartFile("keksi", "keksi.png", MediaType.IMAGE.PNG.VALUE,
94                 this.getClass().getResourceAsStream("images/keksi.png").readAllBytes());
95         request.setPictures(pictures);
96         User user = userRepository.findById(request.getUserId()).get();
97         assertThrows(IllegalArgumentException.class, () -> RecipeController.getNewRecipeData(request, user));
98     }
99
100     @Test
101     void testGetNewRecipeDataWhenPassingEmptyImages() throws IOException {
102         RecipeRequest request = new RecipeRequest();
103         request.setTitle("New");
```

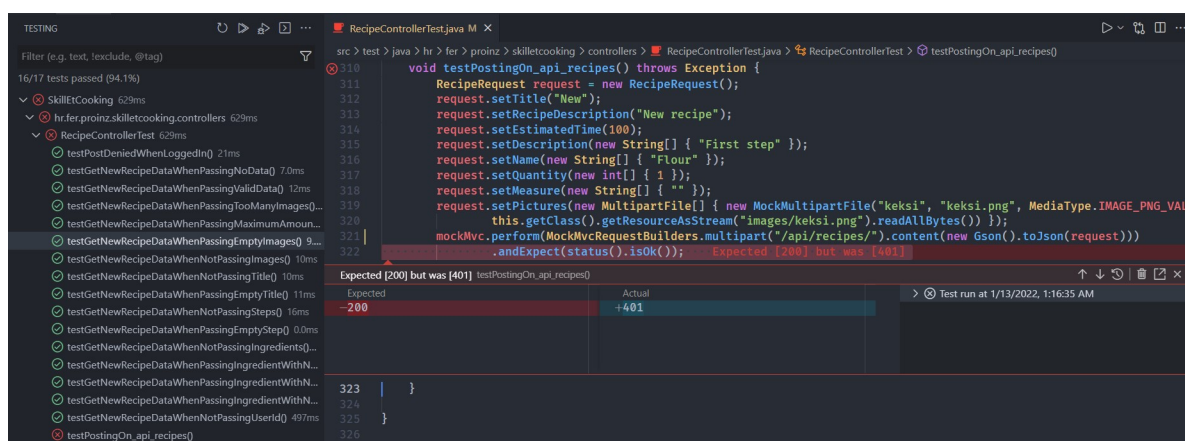
```
104         request.setRecipeDescription("New recipe");
105         request.setEstimatedTime(100);
106         request.setDescription(new String[] { "First step" });
107         request.setName(new String[] { "Flour" });
108         request.setQuantity(new int[] { 1 });
109         request.setMeasure(new String[] { "kg" });
110         request.setUserId(3);
111         request.setPictures(new MultipartFile[] {});
112         User user = userRepository.findById(request.getUserId()).get();
113         assertThrows(IllegalArgumentException.class, () -> RecipeController.getNewRecipeData(request, user));
114     }
115
116     @Test
117     void testGetNewRecipeDataWhenNotPassingImages() throws IOException {
118         RecipeRequest request = new RecipeRequest();
119         request.setTitle("New");
120         request.setRecipeDescription("New recipe");
121         request.setEstimatedTime(100);
122         request.setDescription(new String[] { "First step" });
123         request.setName(new String[] { "Flour" });
124         request.setQuantity(new int[] { 1 });
125         request.setMeasure(new String[] { "kg" });
126         request.setUserId(3);
127         User user = userRepository.findById(request.getUserId()).get();
128         assertThrows(IllegalArgumentException.class, () -> RecipeController.getNewRecipeData(request, user));
129     }
130
131     @Test
132     void testGetNewRecipeDataWhenNotPassingTitle() throws IOException {
133         RecipeRequest request = new RecipeRequest();
134         request.setRecipeDescription("New recipe");
135         request.setEstimatedTime(100);
136         request.setDescription(new String[] { "First step" });
137         request.setName(new String[] { "Flour" });
138         request.setQuantity(new int[] { 1 });
139         request.setMeasure(new String[] { "kg" });
140         request.setUserId(3);
141         request.setPictures(new MultipartFile[] { new MockMultipartFile("keksi", "keksi.png", MediaType.IMAGE.PNG.VALUE,
142             this.getClass().getResourceAsStream("images/keksi.png").readAllBytes()) });
143         User user = userRepository.findById(request.getUserId()).get();
144         assertThrows(IllegalArgumentException.class, () -> RecipeController.getNewRecipeData(request, user));
145     }
146
147     @Test
148     void testGetNewRecipeDataWhenPassingEmptyTitle() throws IOException {
149         RecipeRequest request = new RecipeRequest();
150         request.setTitle("");
151         request.setRecipeDescription("New recipe");
152         request.setEstimatedTime(100);
153         request.setDescription(new String[] { "First step" });
154         request.setName(new String[] { "Flour" });
155         request.setQuantity(new int[] { 1 });
156         request.setMeasure(new String[] { "kg" });
157         request.setUserId(3);
158         request.setPictures(new MultipartFile[] { new MockMultipartFile("keksi", "keksi.png", MediaType.IMAGE.PNG.VALUE,
159             this.getClass().getResourceAsStream("images/keksi.png").readAllBytes()) });
160         User user = userRepository.findById(request.getUserId()).get();
161         assertThrows(IllegalArgumentException.class, () -> RecipeController.getNewRecipeData(request, user));
162     }
163
164     @Test
165     void testGetNewRecipeDataWhenNotPassingSteps() throws IOException {
166         RecipeRequest request = new RecipeRequest();
167         request.setTitle("New");
168         request.setRecipeDescription("New recipe");
169         request.setEstimatedTime(100);
170         request.setName(new String[] { "Flour" });
171         request.setQuantity(new int[] { 1 });
172         request.setMeasure(new String[] { "kg" });
173         request.setUserId(3);
174         request.setPictures(new MultipartFile[] { new MockMultipartFile("keksi", "keksi.png", MediaType.IMAGE.PNG.VALUE,
175             this.getClass().getResourceAsStream("images/keksi.png").readAllBytes()) });
176         User user = userRepository.findById(request.getUserId()).get();
```

```
177         assertThrows(IllegalArgumentException.class, () -> RecipeController.getNewRecipeData(request, user));
178     }
179
180     @Test
181     void testGetNewRecipeDataWhenPassingEmptyStep() throws IOException {
182         RecipeRequest request = new RecipeRequest();
183         request.setTitle("New");
184         request.setRecipeDescription("New recipe");
185         request.setEstimatedTime(100);
186         request.setDescription(new String[] { "" });
187         request.setName(new String[] { "Flour" });
188         request.setQuantity(new int[] { 1 });
189         request.setMeasure(new String[] { "kg" });
190         request.setUserId(3);
191         request.setPictures(new MultipartFile[] { new MockMultipartFile("keksi", "keksi.png", MediaType.IMAGE_PNG.VALUE,
192             this.getClass().getResourceAsStream("images/keksi.png").readAllBytes()) });
193         User user = userRepository.findById(request.getUserId()).get();
194         assertThrows(IllegalArgumentException.class, () -> RecipeController.getNewRecipeData(request, user));
195     }
196
197     @Test
198     void testGetNewRecipeDataWhenNotPassingIngredients() throws IOException {
199         RecipeRequest request = new RecipeRequest();
200         request.setTitle("New");
201         request.setRecipeDescription("New recipe");
202         request.setEstimatedTime(100);
203         request.setDescription(new String[] { "First step" });
204         request.setUserId(3);
205         request.setPictures(new MultipartFile[] { new MockMultipartFile("keksi", "keksi.png", MediaType.IMAGE_PNG.VALUE,
206             this.getClass().getResourceAsStream("images/keksi.png").readAllBytes()) });
207         User user = userRepository.findById(request.getUserId()).get();
208         assertThrows(IllegalArgumentException.class, () -> RecipeController.getNewRecipeData(request, user));
209     }
210
211     @Test
212     void testGetNewRecipeDataWhenPassingIngredientWithNoName() throws IOException {
213         RecipeRequest request = new RecipeRequest();
214         request.setTitle("New");
215         request.setRecipeDescription("New recipe");
216         request.setEstimatedTime(100);
217         request.setDescription(new String[] { "First step" });
218         request.setName(new String[] { "" });
219         request.setQuantity(new int[] { 1 });
220         request.setMeasure(new String[] { "kg" });
221         request.setUserId(3);
222         request.setPictures(new MultipartFile[] { new MockMultipartFile("keksi", "keksi.png", MediaType.IMAGE_PNG.VALUE,
223             this.getClass().getResourceAsStream("images/keksi.png").readAllBytes()) });
224         User user = userRepository.findById(request.getUserId()).get();
225         assertThrows(IllegalArgumentException.class, () -> RecipeController.getNewRecipeData(request, user));
226     }
227
228     @Test
229     void testGetNewRecipeDataWhenPassingIngredientWithNoQuantity() throws IOException {
230         RecipeRequest request = new RecipeRequest();
231         request.setTitle("New");
232         request.setRecipeDescription("New recipe");
233         request.setEstimatedTime(100);
234         request.setDescription(new String[] { "First step" });
235         request.setName(new String[] { "Flour" });
236         request.setQuantity(new int[] {});
237         request.setMeasure(new String[] { "kg" });
238         request.setUserId(3);
239         request.setPictures(new MultipartFile[] { new MockMultipartFile("keksi", "keksi.png", MediaType.IMAGE_PNG.VALUE,
240             this.getClass().getResourceAsStream("images/keksi.png").readAllBytes()) });
241         User user = userRepository.findById(request.getUserId()).get();
242         assertThrows(IllegalArgumentException.class, () -> RecipeController.getNewRecipeData(request, user));
243     }
244
245     @Test
246     void testGetNewRecipeDataWhenPassingIngredientWithNoMeasure() throws IOException {
247         RecipeRequest request = new RecipeRequest();
248         request.setTitle("New");
249         request.setRecipeDescription("New recipe");
```

```

250     request.setEstimatedTime(100);
251     request.setDescription(new String[] { "First step" });
252     request.setName(new String[] { "Flour" });
253     request.setQuantity(new int[] { 1 });
254     request.setMeasure(new String[] { "" });
255     request.setUserId(3);
256     request.setPictures(new MultipartFile[] { new MockMultipartFile("keksi", "keksi.png", MediaType.IMAGE_PNG.VALUE,
257         this.getClass().getResourceAsStream("images/keksi.png").readAllBytes()) });
258     User user = userRepository.findById(request.getUserId()).get();
259     assertThrows(IllegalArgumentException.class, () -> RecipeController.getNewRecipeData(request, user));
260 }
261
262 @Test
263 void testGetNewRecipeDataWhenNotPassingUserId() throws IOException {
264     RecipeRequest request = new RecipeRequest();
265     request.setTitle("New");
266     request.setRecipeDescription("New recipe");
267     request.setEstimatedTime(100);
268     request.setDescription(new String[] { "First step" });
269     request.setName(new String[] { "Flour" });
270     request.setQuantity(new int[] { 1 });
271     request.setMeasure(new String[] { "" });
272     request.setPictures(new MultipartFile[] { new MockMultipartFile("keksi", "keksi.png", MediaType.IMAGE_PNG.VALUE,
273         this.getClass().getResourceAsStream("images/keksi.png").readAllBytes()) });
274     User user = userRepository.findById(3).get();
275     assertThrows(IllegalArgumentException.class, () -> RecipeController.getNewRecipeData(request, user));
276 }
277
278 @Test
279 @Transactional
280 void testPostingOn_api_recipes() throws Exception {
281     RecipeRequest request = new RecipeRequest();
282     request.setTitle("New");
283     request.setRecipeDescription("New recipe");
284     request.setEstimatedTime(100);
285     request.setDescription(new String[] { "First step" });
286     request.setName(new String[] { "Flour" });
287     request.setQuantity(new int[] { 1 });
288     request.setMeasure(new String[] { "" });
289     request.setPictures(new MultipartFile[] { new MockMultipartFile("keksi", "keksi.png", MediaType.IMAGE_PNG.VALUE,
290         this.getClass().getResourceAsStream("images/keksi.png").readAllBytes()) });
291     mockMvc.perform(MockMvcRequestBuilders.multipart("/api/recipes/").content(new Gson().toJson(request)))
292         .andExpect(status().isOk());
293 }
294
295 }

```

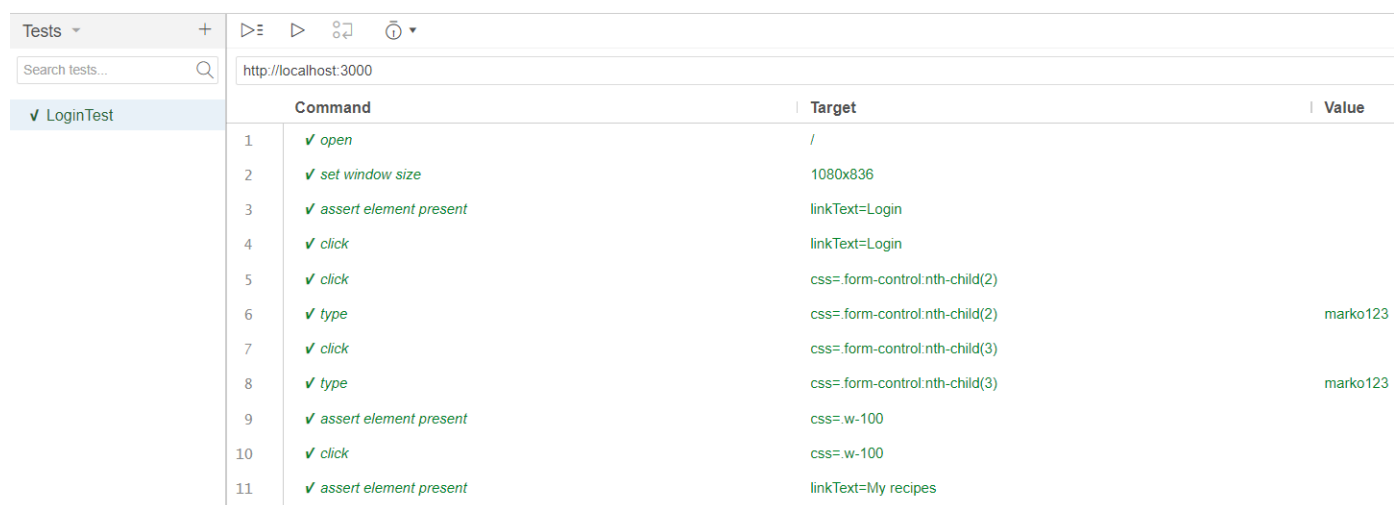


Slika 5.1: Rezultati izvođenja testova u uređivaču VSCode.

5.2.2 Ispitivanje sustava

1. Ispitivanje prijave u sustav (UC-7)

U ovom ispitnom slučaju je provedena prijava. Scenarij započinje otvaranjem početne stranice i pod pretpostavkom da korisnik nije prijavljen u sustav biti će vidljiva hiperveza koja vodi na stranicu za prijavu. Nakon toga unošenjem korisničkog imena i lozinke te klikom na gumb za prijavu korisnik bi trebao biti preusmjeren na početnu stranicu ali ovaj put uz dodatne mogućnosti što se provjerava postojanjem hiperveze koja vodi na vlastite recepte.



	Command	Target	Value
1	✓ open	/	
2	✓ set window size	1080x836	
3	✓ assert element present	linkText=Login	
4	✓ click	linkText=Login	
5	✓ click	css=.form-control:nth-child(2)	
6	✓ type	css=.form-control:nth-child(2)	marko123
7	✓ click	css=.form-control:nth-child(3)	
8	✓ type	css=.form-control:nth-child(3)	marko123
9	✓ assert element present	css=w-100	
10	✓ click	css=w-100	
11	✓ assert element present	linkText=My recipes	

Slika 5.2: Ispitivanje prijave u sustav.

2. Ispitivanje pretraživanja recepta po sastojcima (UC-5)

U ovom ispitnom slučaju je provedeno pretraživanje recepta po sastojcima. Scenarij započinje otvaranjem početne stranice na kojoj se nalazi HTML element za unos sastojaka. Pod pretpostavkom da u bazi postoji recept sa sastojcima: peršin, češnjak, špageti i vrhnje (za-kuhanje), nakon odabira upravo tih sastojaka na početnoj stranici bi trebao biti vidljiv taj recept što se provjerava njegovom prisutnošću.

✓ Jaccard test*		http://localhost:3000	
	Command	Target	Value
1	✓ open	/	
2	✓ set window size	1090x835	
3	✓ click	css=.css-6j8wv5-Input	
4	peršin		
5	✓ click	css=.css-6j8wv5-Input	
6	češnjak		
7	✓ click	css=.css-6j8wv5-Input	
8	\$pageti		
9	✓ click	css=.css-6j8wv5-Input	
10	vrhnje(za-kuhanje)		
11	✓ assert element present	css=.card-img-top	
12	✓ click	id=nameFilter	
13	✓ type	id=nameFilter	spaghetti
14	✓ assert element present	css=.card-img-top	

Slika 5.3: Ispitivanje pretraživanja receptata po sastojcima.

3. Ispitivanje registracije u sustav (UC-6)

U ovom ispitnom slučaju je provedena registracija korisnika u sustav. Scenarij započinje otvaranjem početne stranice gdje su prikazani svi recepti. Nakon toga korisnik odabire hipervezu registracija koja ga vodi na formu za registraciju. Korisnik popunjava podatke te ako su zadovoljeni svi uvjeti korisnik će biti uspješno registriran te će ga se potom preusmjeriti na početnu stranicu, ali će ovaj put biti prijavljen u sustav. Nakon toga će biti vidljiva hiperveza My recipes koja se pojavljuje samo prijavljenim korisnicima.

✓ registracija	http://localhost:3000		
	Command	Target	Value
1	✓ open	/	
2	✓ set window size	1400x1040	
3	✓ assert element present	linkText=Register	
4	✓ click	linkText=Register	
5	✓ click	name=lastName	
6	✓ type	name=lastName	Kucic
7	✓ type	name=email	mlucic@blue
8	✓ type	name=firstName	Mario
9	✓ click	name=dateOfBirth	
10	✓ type	name=dateOfBirth	2000-11-09
11	✓ click	name=username	
12	✓ type	name=username	kuk
13	✓ mouse down at	name=email	125.5.20.40625
14	✓ mouse move at	name=email	125.5.20.40625
15	✓ mouse up at	name=email	125.5.20.40625
16	✓ click	name=email	
17	✓ click	name=email	
18	✓ click	css=main>div	
19	✓ click	name=password	
20	✓ type	name=password	lozinka
21	✓ click	css=form-control:nth-child(3)	
22	✓ type	css=form-control:nth-child(3)	lozinka
23	✓ assert element present	css=main>div	
24	✓ click	css=main>div	
25	✓ assert element present	css=main>div	
		linkText=My recipes	

Slika 5.4: Ispitivanje registracije u sustav.

4. Ispitivanje dodavanja komentara moderatora (UC-14)

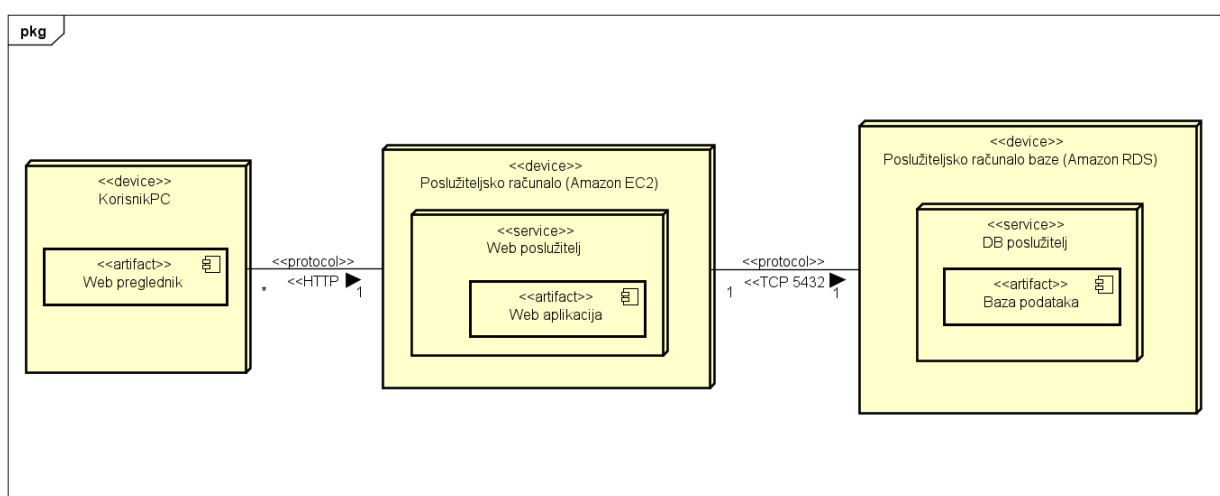
U ovom ispitnom slučaju je provedeno dodavanje komentara moderatora. Moderator se poput ostalih korisnika mora prijaviti u sustav. Nakon prijave moderator odabire opciju prikaza određenog recepta. U polju za komentare upisuje komentar po želji i pritiskom na hipervezu objavi komentar dodaje komentar u bazu. Pritiskom na tipku osvježi komentar će se prikazati uz ostale komentare toga recepta. Također, komentar će biti dodatno naglašen oznakom za moderatore (kruna).

✓ dodavanjeKomentaraModeratora*		http://localhost:3000		
	Command	Target	Value	
1	✓ open	/		
2	✓ set window size	1407x1040		
3	✓ assert element present	linkText=Login		
4	✓ click	linkText=Login		
5	✓ click	css= form-control:nth-child(2)		
6	✓ type	css= form-control:nth-child(2)	ivo	
7	✓ click	css= form-control:nth-child(3)		
8	✓ type	css= form-control:nth-child(3)	ivo	
9	✓ assert element present	css= w-100		
10	✓ click	css= w-100		
11	✓ assert element present	linkText=Pojedinosti o receptu		
12	✓ click	linkText=Pojedinosti o receptu		
13	✓ assert element present	id=comment-textarea		
14	✓ click	id=comment-textarea		
15	✓ type	id=comment-textarea	Dodavanje komentara	
16	✓ assert element present	css= post-comment-form > btn		
17	✓ click	css= post-comment-form > btn		
18	✓ assert element present	id=refresh-comments-btn		
19	✓ click	id=refresh-comments-btn		
20	✓ mouse over	id=refresh-comments-btn		
21	✓ mouse out	id=refresh-comments-btn		
22	✓ assert element present	css= comment-icon		

Slika 5.5: Ispitivanje dodavanja komentara moderatora.

5.3 Dijagram razmještaja

Dijagrami razmještaja opisuju topologiju sklopovlja i programsku potporu koja se koristi u implementaciji sustava u njegovom radnom okruženju. Upogonjenje smo izvršili na AWS-u pa imamo dijagram s 3 uređaja. Servis web poslužitelja pokrenut je na Amazon EC2 instanci koji s bazom pokrenutoj preko Amazon RDS-a komunicira preko TCP protokola na vratima 5432. Klijenti koriste web preglednik na vlastitim računalima kako bi pristupili web aplikaciji. Sustav je baziran na arhitekturi "klijent -poslužitelj", a komunikacija između računala korisnika i poslužitelja odvija se preko HTTP veze.



Slika 5.6: Dijagram razmještaja

5.4 Upute za puštanje u pogon

Instalacija potrebnih programa

Prije lokalnog pokretanja potrebno je instalirati sljedeće programske alate:

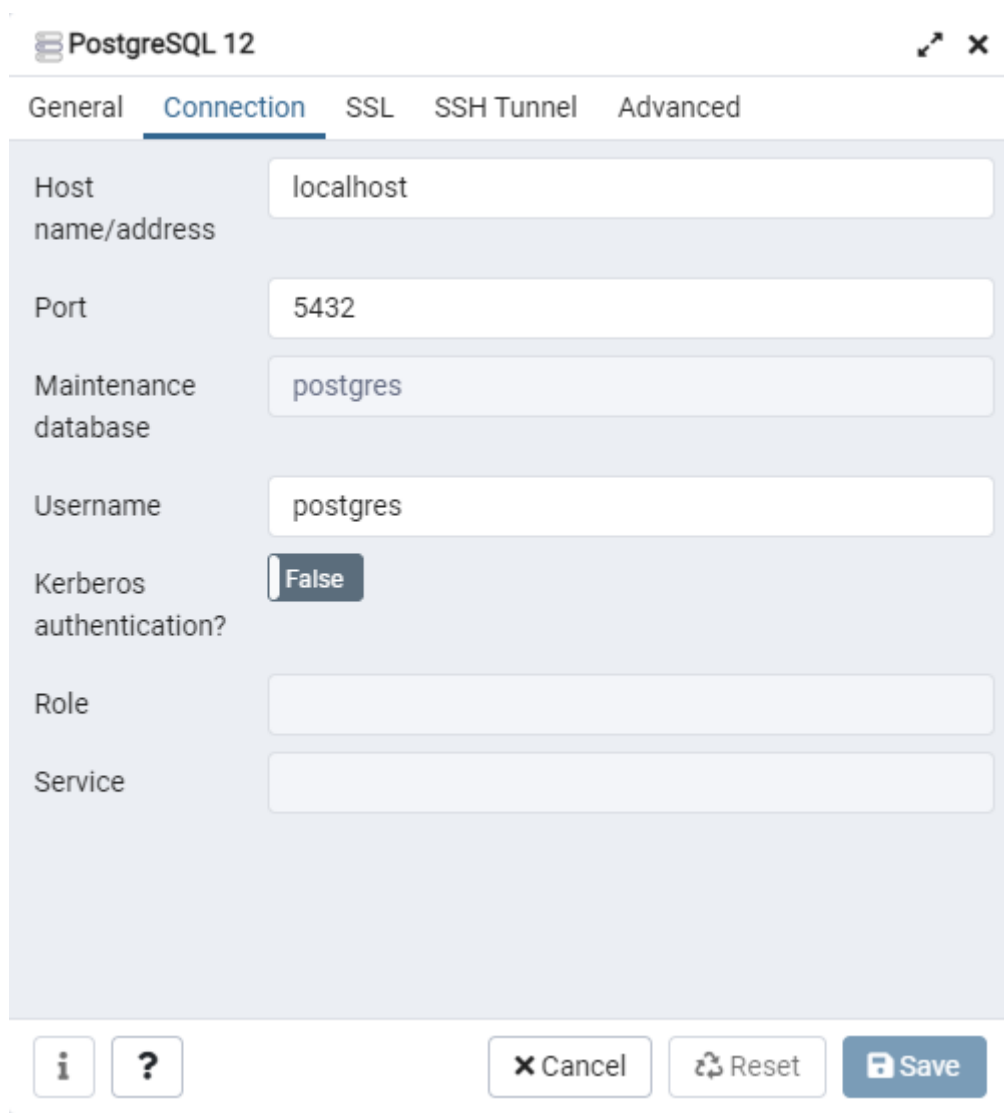
- Node + NPM
- Java 11
- Maven
- PostgreSQL 12 + PgAdmin
- git

Za kloniranje repozitorija i koda koristimo sljedeću naredbu te se authenticiramo našim GitLab podacima:

```
git clone https://gitlab.com/progimeri/proinz-projekt.git
```

Konfiguracija baze podataka

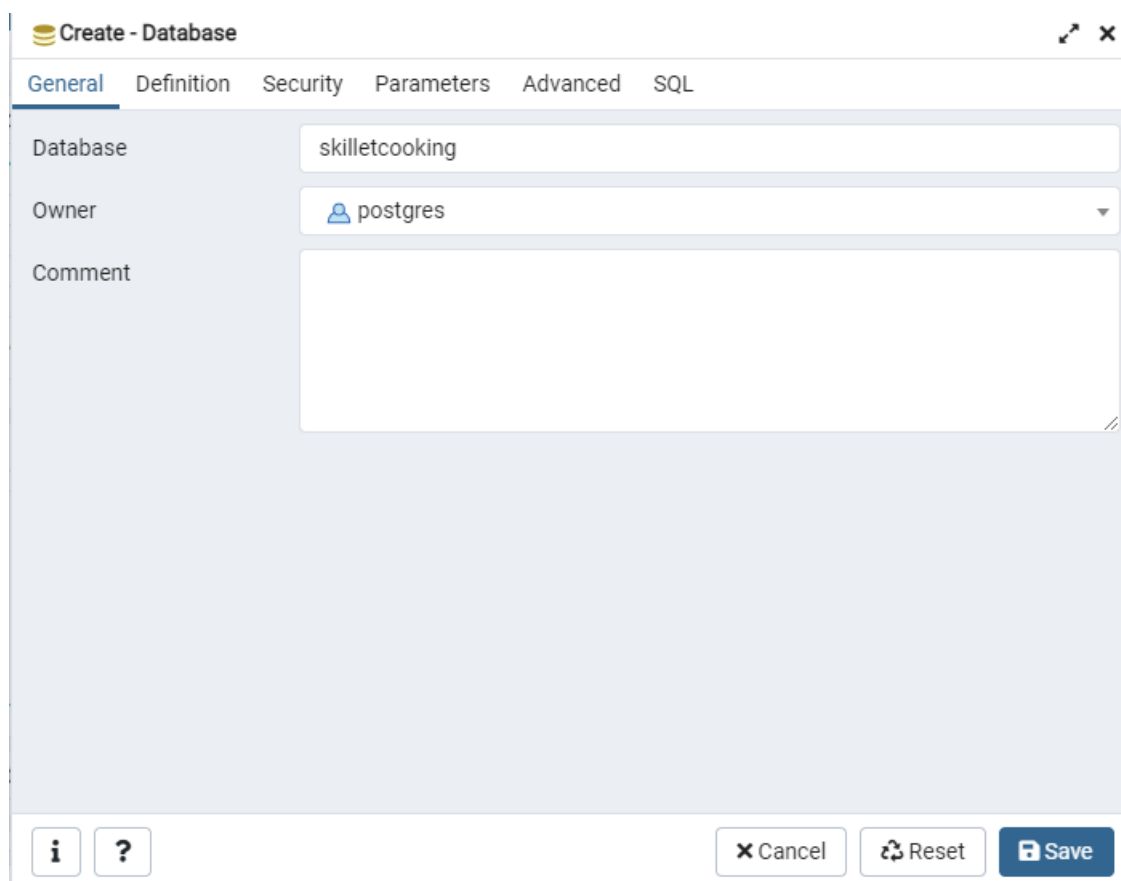
Prije pokretanja same web aplikacije moramo konfigurirati lokalnu bazu podataka. Konfiguraciju poslužitelja baze možemo napraviti proizvoljno te poslije specificirati u kodu, ali preporuča se korištenje zadanih podataka:



The image shows the 'PostgreSQL 12' connection configuration window. It has a title bar with a maximize icon and a close button. Below the title bar are five tabs: 'General', 'Connection' (which is selected and highlighted in blue), 'SSL', 'SSH Tunnel', and 'Advanced'. The 'Connection' tab contains several fields: 'Host name/address' with the value 'localhost', 'Port' with the value '5432', 'Maintenance database' with the value 'postgres', 'Username' with the value 'postgres', 'Kerberos authentication?' with a dropdown menu set to 'False', 'Role' with an empty text box, and 'Service' with an empty text box. At the bottom of the window are three buttons: 'Cancel', 'Reset', and 'Save'.

Slika 5.7: Postavke poslužitelja baze

Nakon stvaranja poslužitelja baze možemo stvoriti bazu bilo kojeg imena, ali preporuka je koristiti ime 'skilletcooking' s vlasnikom 'postgres':



Slika 5.8: Postavke baze podataka

Konfiguracija stražnjeg dijela

Samu shemu baze popuniti će Flyway migracija prilikom prvog pokretanja back-end poslužitelja. Kako bi poslužitelj spojili na lokalnu bazu podataka trebamo definirati da je aktivni profil dev u application.properties datoteci:

```
spring.profiles.active=dev
```

Samim time ako se zbog nekog razloga želimo spojiti na bazu podataka koja je pokrenuta na AWS-u to možemo jednostavno tako da zamijenimo aktivni profil sa prod:

```
spring.profiles.active=prod
```

Prijašnje postavke koje smo definirali pri izradi baze podataka sada moramo

specificirati u application-dev.properties datoteci u izvornom kodu back-enda i izmijeniti sljedeće linije.

```
spring.datasource.url=jdbc:postgresql://{host_name}/{database_name}
spring.datasource.username={database_user_name}
spring.datasource.password={lozinka}

spring.flyway.url=jdbc:postgresql://{host_name}/{database_name}
spring.flyway.user={database_user_name}
spring.flyway.password={lozinka}
```

Za predložene podatke i korisnika postgres kojemu je lozinka 'bazepodataka' to bi izgledalo ovako:

```
spring.datasource.url=jdbc:postgresql://localhost:5432/skilletcooking
spring.datasource.username=postgres
spring.datasource.password=bazepodataka

spring.flyway.url=jdbc:postgresql://localhost:5432/skilletcooking
spring.flyway.user=postgres
spring.flyway.password=bazepodataka
```

Kada je cijela konfiguracija namještena napokon možemo pokrenuti back-end koji će pokrenuti generiranje sheme u bazi. To možemo tako da se u konzoli pozicioniramo u mapu izvorniKod/back-end i pokrenemo naredbu `mvn spring-boot:run` koja će pokrenuti Java Spring poslužitelj na vratima 8080. Back-end se također može pokrenuti korištenjem modernih naprednih IDE-jeva poput IntelliJ-a.

Kada je servis uspješno pokrenut i nakon popunjenja sheme još jedan preduvjet prije pokretanja cjelokupne aplikacije je definicija "uloga" koje korisnici mogu imati, a to su **USER** i **MODERATOR**. Te uloge možemo jednostavno dodati korištenjem alata za upite u PgAdminu i pokretanju sljedećih upita:

```
INSERT INTO roles(role_name) VALUES ( 'USER' );
INSERT INTO roles(role_name) VALUES ( 'MODERATOR' );
```

Pokretanje prednjeg dijela

Sada je cijeli back-end spreman i možemo pokrenuti front-end. Pokrenimo novu konzolu i pozicioniramo se u mapu izvorniKod/front-end. Pokrećemo naredbu

npm install koja će instalirati sve potrebne pakete s npm-a te nakon toga možemo izvršiti naredbu npm start koja će pokrenuti poslužitelj za prednju stranu na vratima 3000 i njemu možemo pristupiti na adresi localhost:3000.

Javni poslužitelj

Aplikacija je pokrenuta na javnom poslužitelju preko AWS-a i dostupna je na stranici: <http://34.244.45.75/> .

6. Zaključak i budući rad

Zadatak naše grupe bio je razvoj web aplikacije za pregled recepata uz mogućnost sortiranja, filtriranja, dodavanja vlastitih recepata, komentiranja i ocijenjivanja recepata. Za ovaj projekt dobili smo vrijeme od 17 tjedana. Za to vrijeme naš je tim uspješno završio zadatak i razvio zadanu aplikaciju. Tijek odvijanja projekta može se podijeliti u 2 faze.

Prva faza je faza okupljanja tima i rad na zahtjevima projekta. Dokumentiranje zahtjeva je olakšalo kasniju preraspodjelu poslova kao i lakše osmišljavanje izgleda i funkcionalnosti web aplikacije. Kasniji problemi u drugoj fazi prilikom izrade web aplikacije su se lakše rješavali uz pomoć već postojećih dijagrama čime smo smanjili gubitak vremena.

Druga faza uključivala je izradu i ispitivanje web aplikacije uz dokumentiranje projekta. Naglasak je više bio na samostalnome radu za razliku od prve faze zbog čega je ponekad dolazilo do problema. Neiskustvo u radu u timu kao i u radu na projektu je predstavljao najveći izazov, ali su svi članovi bili tu jedni za druge te nije bilo pretjeranih problema.

Komunikacija u timu je bila jako dobra. Svi članovi tima bili su pozitivni i nije dolazilo do neželjenih sukoba.

Sudjelovanje u ovakvome projektu bilo je od iznimne koristi svim članovima tima. Budući da je svatko radio u svim fazama i na svim dijelovima web aplikacije, puno smo toga imali prilike naučiti. Raditi na ovome projektu bilo je dobro iskustvo i za buduće timske radove.

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	Prikaz svih recepata	6
2.2	Primjer otvorenog recepta	7
2.3	Komentari na recept	8
2.4	Prijava i registracija u sustav	8
3.1	Dijagram obrasca uporabe, funkcionalnost korisnika	19
3.2	Dijagram obrasca uporabe, funkcionalnost moderatora	20
3.3	Sekvencijski dijagram za UC4	21
3.4	Sekvencijski dijagram za UC6	22
3.5	Sekvencijski dijagram za UC11	23
3.6	Sekvencijski dijagram za UC13	24
4.1	Arhitektura sustava	26
4.2	E-R dijagram baze podataka	32
4.3	Dijagram razreda - Models	33
4.4	Dijagram razreda - Controllers	34
4.5	Dijagram razreda - Repositories	35
4.6	Dijagram stanja	36
4.7	Dijagram aktivnosti	38
4.8	Dijagram komponenti	39
5.1	Rezultati izvođenja testova u uređivaču VSCode.	45
5.2	Dijagram razmještaja	47
5.3	Postavke poslužitelja baze	49
5.4	Postavke baze podataka	50
6.1	Aktivnost	60

Dodatak: Prikaz aktivnosti grupe

Dnevnik sastajanja

1. sastanak

- Datum: 4. listopada 2021.
- Prisustvovali: K.Frankola, B.Colarić, D.Dugonjevac, T.Serezlija, M.Tunjić
- Teme sastanka:
 - Međusobno upoznavanje i predstavljanje
 - Rasprava o projektu i zajedničkim aktivnostima (u grubo)
 - Prijava na stranicu raspodjele grupa

2. sastanak

- Datum: 11. listopada 2021.
- Prisustvovali: K.Frankola, J.Brkić, B.Colarić, D.Dugonjevac, P.Gavran, T.Serezlija, M.Tunjić
- Teme sastanka:
 - Upoznavanje novih članova
 - Ponovna rasprava o odabranim tehnologijama
 - Podjela aktivnosti po članovima

3. sastanak

- Datum: 27. listopada 2021.
- Prisustvovali: K.Frankola, J.Brkić, B.Colarić, D.Dugonjevac, P.Gavran, T.Serezlija, M.Tunjić
- Teme sastanka:
 - Daljnja podjela aktivnosti (dokumentacija i programiranje)
 - Razmatranje arhitekture i ponašanja sustava

4. sastanak

- Datum: 6. studenoga 2021.
- Prisustvovali: K.Frankola, J.Brkić, B.Colarić, D.Dugonjevac, P.Gavran, T.Serezlija, M.Tunjić
- Teme sastanka:

- Osnovni dijagram baze
- Osnovni dijagram ponašanja sustava
- Aktivnosti koje je potrebno napraviti do predaje

5. sastanak

- Datum: 9. prosinca 2021.
- Prisustvovali: K.Frankola, J.Brkić, B.Colarić, D.Dugonjevac, P.Gavran, T.Serezlija, M.Tunjić
- Teme sastanka:
 - Podjela aktivnosti za alfa verziju

6. sastanak

- Datum: 10. siječnja 2022.
- Prisustvovali: K.Frankola, J.Brkić, B.Colarić, D.Dugonjevac, P.Gavran, T.Serezlija, M.Tunjić
- Teme sastanka:
 - Finalni dogovori oko podjele zadataka
 - Završavanje implementacije i nedostaci

Tablica aktivnosti

Kontinuirano osvježavanje

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

	Karlo Frankola	Jan Brkić	Borna Colarić	Dario Dugonjevac	Petar Gavran	Toni Serezlija	Marko Tunjić
Upravljanje projektom	8						
Opis projektnog zadatka			1	2			2
Funkcionalni zahtjevi			3				2
Opis pojedinih obrazaca						3	
Dijagram obrazaca							1
Sekvencijski dijagrami				2			
Opis ostalih zahtjeva						1	
Arhitektura i dizajn sustava						2	
Baza podataka						2	
Dijagram razreda	1	1.5					
Dijagram stanja		1					
Dijagram aktivnosti		1					
Dijagram komponenti				1			
Korištene tehnologije i alati		1					
Ispitivanje programskog rješenja						4	
Dijagram razmještaja	1						

Nastavljeno na idućoj stranici

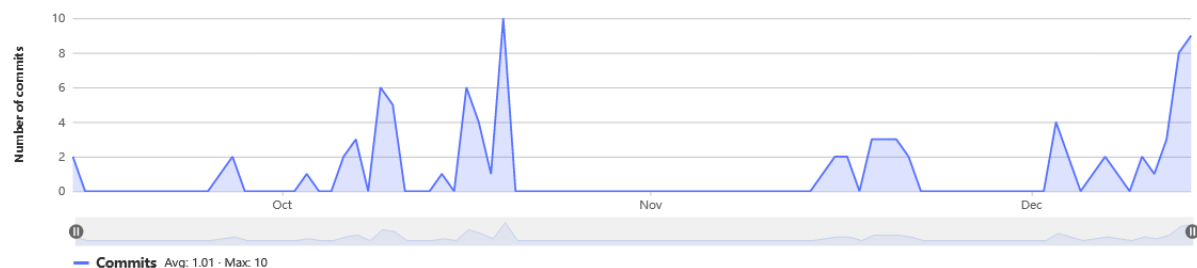
Nastavljeno od prethodne stranice

	Karlo Frankola	Jan Brkić	Borna Colarić	Dario Dugonjevac	Petar Gavran	Toni Serezlija	Marko Tunjić
Upute za puštanje u pogon	1						
Dnevnik sastajanja	1						
Zaključak i budući rad				1			
Popis literature							
<i>Izrada početne stranice</i>	4	1					
<i>Izrada baze podataka</i>	2		1	1		1	1
<i>Spajanje s bazom podataka</i>	2						
<i>Back End</i>	12		7	7		7	8
<i>Front End</i>	8	7	3	7		8	7
<i>Puštanje u pogon</i>	7						

Dijagrami pregleda promjena

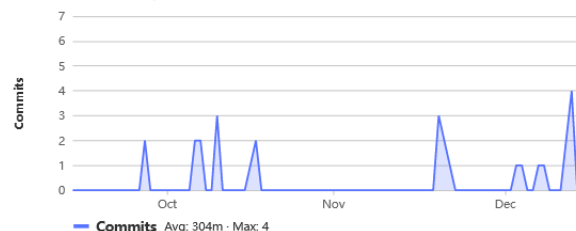
Commits to main

Excluding merge commits. Limited to 6,000 commits.



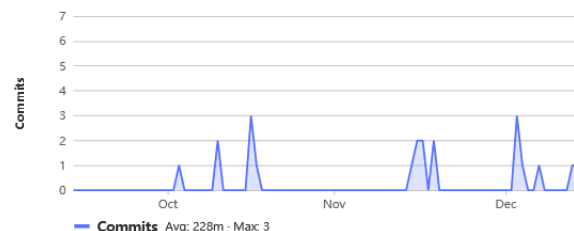
Marko

28 commits (marko.tunjic@fer.hr)



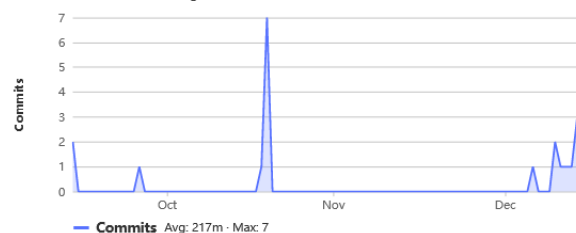
toniSerezlija

21 commits (ts52728@fer.hr)



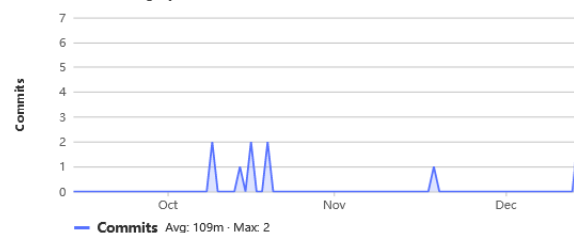
fkarlo

20 commits (karlo.frankola@gmail.com)



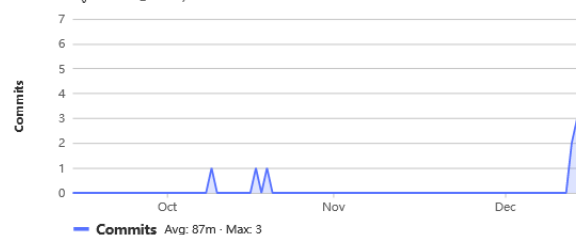
Dario

10 commits (dario.dugonjevac@fer.hr)



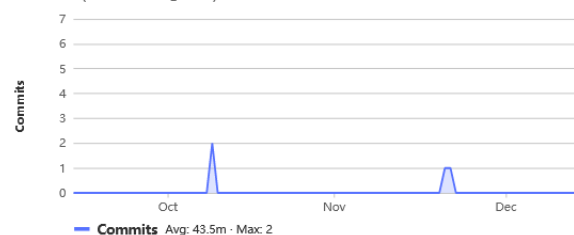
jan.brkic

8 commits (jan.brkic@fer.hr)



bcolaric

4 commits (borna.colaric@fer.hr)



Slika 6.1: Aktivnost