

Programsko inženjerstvo

Ak. god. 2023./2024.

KuhajIT

Dokumentacija, Rev. 2

Grupa: 30Bodova

Voditelj: Hana Ivančić

Datum predaje: 19. 01. 2024.

Nastavnik: mag. ing. Hrvoje Nuić

Sadržaj

1 Dnevnik promjena dokumentacije	3
2 Opis projektnog zadatka	5
2.1 Potencijalna korist ovog projekta	5
2.2 Postojeća slična rješenja	5
2.3 Skup korisnika koji bi mogao biti zainteresiran za ostvareno rješenje	7
2.4 Opseg projektnog zadatka	7
2.5 Moguće nadogradnje projektnog zadatka	9
3 Specifikacija programske potpore	11
3.1 Funkcionalni zahtjevi	11
3.1.1 Obrasci uporabe	13
3.1.2 Sekvencijski dijagrami	22
3.2 Ostali zahtjevi	25
4 Arhitektura i dizajn sustava	26
4.1 Baza podataka	28
4.1.1 Opis tablica	29
4.2 Dijagram razreda	38
4.3 Dijagram stanja	42
4.4 Dijagram aktivnosti	44
4.5 Dijagram komponenti	45
5 Implementacija i korisničko sučelje	46
5.1 Korištene tehnologije i alati	46
5.2 Ispitivanje programskog rješenja	48
5.2.1 Ispitivanje komponenti	48
5.3 Dijagram razmještaja	56
5.4 Upute za puštanje u pogon	58
6 Zaključak i budući rad	67

Popis literature	69
Dodatak: Prikaz aktivnosti grupe	70

1. Dnevnik promjena dokumentacije

Rev.	Opis promjene/dodataka	Autori	Datum
0.1	Napravljen predložak.	Dunja Petrović	21.10.2023.
0.2	Dopisane upute za povijest dokumentacije. Dodane reference.	Dunja Petrović	22.10.2023.
0.3	Dodani <i>Use Case</i> dijagrami i sekvensijski dijagrami, funkcionalni i nefunkcionalni zahtjevi	Dunja Petrović	29.10.2023.
0.4	Dopunjeni <i>Use Case</i> dijagrami i sekvensijski dijagrami, funkcionalni i nefunkcionalni zahtjevi	Hana Ivančić	08.11.2023.
0.5.	Dodan opis arhitekture sustava.	Dunja Petrović	11.11.2023.
0.6.	Dodan opis baze podataka.	Dunja Petrović	13.11.2023.
0.7.	Dodan dijagram razreda modela i kontrolera.	Hana Ivančić	14.11.2023.
0.8.	Dodan opis dijagrama razreda modela i kontrolera, nadopunjeno opis arhitekture sustava.	Dunja Petrović	14.11.2023.
0.9	Dodan dijagram razred DTO.	Hana Ivančić	15.11.2023.
0.10	Dodan opis dijagrama razreda DTO.	Dunja Petrović	16.11.2023.
1.0	Ispravljen dio grešaka iz prve revizije.	Dunja Petrović	10.12.2023.
1.1	Ispravljen dio grešaka iz prve revizije.	Dunja Petrović	14.12.2023.

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

Rev.	Opis promjene/dodataka	Autori	Datum
1.2	Dodan dijagram stanja.	Dunja Petrović	26.12.2023.
1.3	Dodan dijagram komponenti, dijagram razmještaja i dijagram aktivnosti.	Dunja Petrović	27.12.2023.
1.4	Dodane upute za pogon.	Dunja Petrović	29.12.2023.
1.5	Dodano poglavlje Korištene tehnologije, nadopunjene upute za puštanje u pogon.	Dunja Petrović	2.1.2024.
1.6	Započeto poglavlje Zaključak i budući rad.	Dunja Petrović	4.1.2024.
1.7	Započet opis JUnit testova.	Dunja Petrović	6.1.2024.
1.8	Dovršen opis JUnit testova.	Dunja Petrović	16.01.2024.
1.9	Dodan opis Selenium testova i dovršeno poglavlje Zaključak i budući rad.	Dunja Petrović	19.01.2024.

2. Opis projektnog zadatka

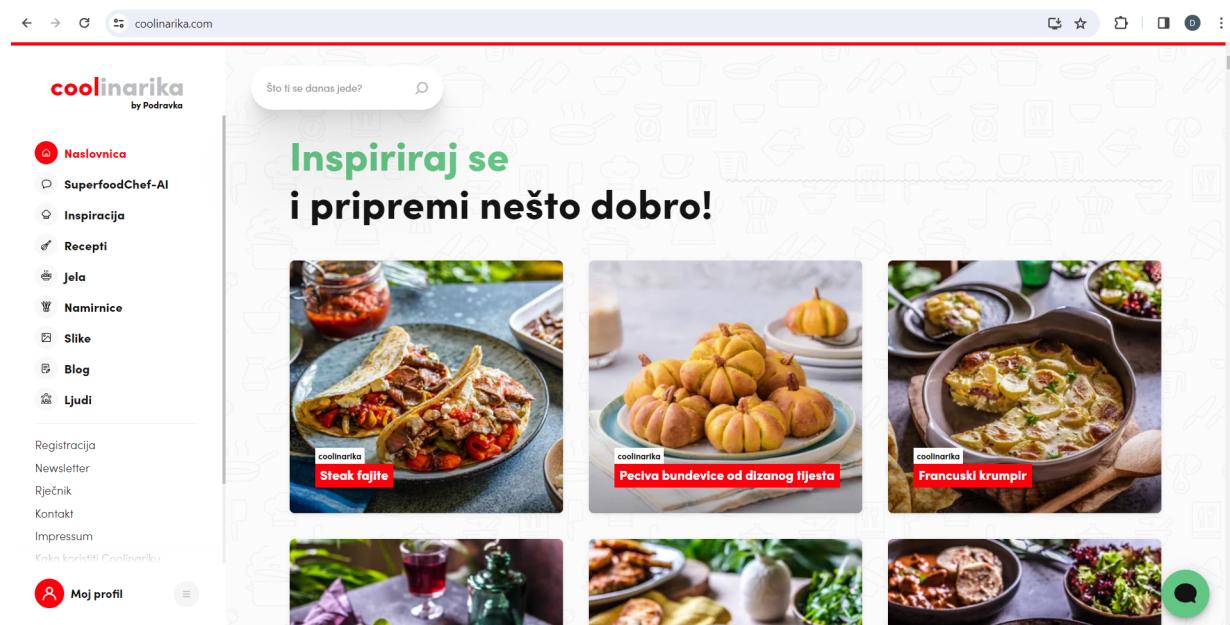
2.1 Potencijalna korist ovog projekta

U današnje doba, ljudima je ponekad teško pronaći druge sličnih interesa. KuhajIT je platforma upravo za pronađak ljudi koji dijele interes za kuhanjem, nutricionizmom i ukusnom hranom. Cilj ovog projekta je razviti tu platformu i učiniti ju što lakšom za korištenje, kako bi kuharima entuzijastima te nutricionistima maksimalno uljepšali i pojednostavili korištenje platforme. Ono što bismo razvijanjem platforme KuhajIT dobili jest velika zajednica ljudi koji razmjenjuju iskustva, recepte, nova saznanja o nutritivnim vrijednostima, i sve im je to nadohvat ruke, udaljeno samo nekoliko klikova miša. Također, KuhajIT bi bila izvrsna platforma za one koji ne znaju kuhati ili žele usavršiti svoje vještine kuhanja, no ne znaju otkud početi.

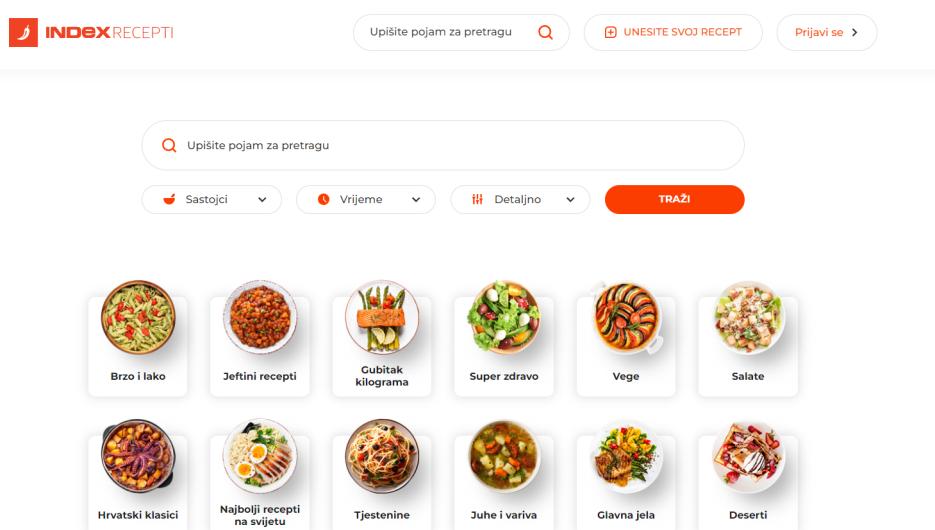
2.2 Postojeća slična rješenja

Trenutno se u dubinama interneta nalazi nekolicina platformi koje djeluju po principu po kojem će djelovati platforma KuhajIT. Najsličnija platforma bila bi coolinarika. coolinarika je platforma koju je razvila Podravka za objavljivanje recepata od strane korisnika. Objavljene recepte ostali korisnici mogu recenzirati i ocjenjivati. To je ono po čemu su coolinarika i KuhajIT slične. Ono po čemu se coolinarika i KuhajIT razlikuju su različite vrste korisnika koje KuhajIT omogućava, dok coolinarika ne razlikuje svoje korisnike. Na taj način svatko tko se registrira može objaviti svoj recept, za razliku od KuhajIT platforme, gdje samo prijavljeni kulinarski entuzijasti mogu objavljivati svoje recepte, što platformi daje dodatni kredibilitet. Također, na platformi coolinarika ne postoji ni uloga 'nutricionista', koji na platformi KuhajIT slaže dijete i na taj način pomaže ostalim korisnicima. Platforma coolinarika ne nudi niti opciju skeniranja bar koda proizvoda za najbrži mogući dolazak do recepta koji je korisnik u danom trenutku u mogućnosti pripremiti. Druga slična platforma jest Index Recepti. Ta platforma također ne razlikuje svoje korisnike već svi imaju iste uloge. To znači da korisnik ne može raditi svoju

kuharicu koja sadrži recepte, no može na stranicu dodati neki recept i na njega staviti određene tagove koji pomažu grupirati slične recepte različitih korisnika, što na KuhajIT nije omogućeno. Također, Index Recepti korisnicima daje mogućnost da njihovi recepti budu javni ili privatni, što znači da ih vidjeti mogu samo korisnici koji prime direktnu poveznicu na njih, dok na platformi KuhajIT svi korisnici mogu vidjeti sve recepte, tj. svi su recepti javni. Index Recepti, kao ni coolinarika, ne nudi opciju skeniranja bar koda proizvoda.



Slika 2.1: Sučelje platforme coolinarika



Slika 2.2: Sučelje platforme Index Recepti

2.3 Skup korisnika koji bi mogao biti zainteresiran za ostvareno rješenje

Za ovakvo rješenje mogli bi biti zainteresirani putni entuzijasti. Uz minimalne promjene platforme njezina tema bi se mogla promijeniti. Za temu putovanja promijenili bi recepte u mjestu koje bi ljudi mogli posjetiti. Tako bi umjesto sastojaka, uputstava i ostalih koraka vezanih za kuhanje, korisnici sada vidjeli znamenitosti određenog mesta, parkove prirode ili nacionalne parkove u blizini i dodatne zanimljivosti koje bi bilo vrijedno pogledati. Kuharica bi sada bila regija ili država, a korisnici bi se dijelili na putopisce, kulturologe i avanturiste. Kulturolozi bi zamijenili kulinarske entuzijaste i mogli bi stvarati svoje "kuharice" tj. najbolja mjesta za posjetiti u određenoj regiji. Putopisci bi zamijenili nutricioniste, i umjesto "dijete" mogli bi korisnicima predložiti personaliziranu rutu ovisno o tome koje regije sve žele posjetiti i koji su njihovi interesi, odnosno u kojim dijelovima tih regija bi najviše mogli uživati. Klijenti bi ovdje bili avanturisti, ljudi koji žele putovati i na ovoj platformi traže ideje za mesta vrijedna posjeta, kao i mogući putni vodič.

2.4 Opseg projektnog zadatka

Na platformi KuhajIT postoji opcija registracije. Neregistrirani korisnik može poslati zahtjev za registraciju u kojoj navodi i za koju se ulogu želi prijaviti. Moguće uloge su:

- klijent
- kulinarski entuzijast
- nutricionist

Ako se neregistrirani korisnik želi prijaviti kao klijent, potrebno mu je sljedeće:

- korisničko ime
- lozinka
- ime korisnika
- prezime korisnika

Međutim, ako se neregistrirani korisnik želi prijaviti kao kulinarski entuzijast ili nutricionist, potrebno je još:

- slika

- kratka biografija
- email

Kako bi neregistrirani korisnik postao kulinarski entuzijast ili nutricionist na platformi KuhajIT, kao takvog ga mora potvrditi ADMINISTRATOR. Svi su registrirani korisnici, bili oni klijenti, kulinarski entuzijasti ili nutricionisti, u mogućnosti mijenjati podatke unutar svog KuhajIT profila. Neregistrirani korisnici nemaju tu mogućnost iz jednostavnog razloga što nisu registrirani, stoga ne posjeduju račun čije bi podatke mogli mijenjati. Svi korisnici, bili oni registrirani ili neregistrirani, mogu pretraživati i pregledavati recepte koje su objavili kulinarski entuzijasti. Na platformi postoji i mogućnost ostavljanja recenzija i ocjena na recepte i kuharice, koje mogu ostaviti svi korisnici, a autor recepta (kulinarски entuzijast koji je objavio recept) ima mogućnost odgovoriti na recenzije korisnika. Ako recenziju ostavi neregistrirani korisnik, on se vodi kao anoniman. Zadatak nutricionista je da unosi informacije o proizvodima (sastojcima jela), razvrstava ih u kategorije s oznakama te na temelju dostupnih informacija o proizvodima izrađuje dijete za klijente i kulinarske entuzijaste platforme KuhajIT. Informacije o nutritivnim vrijednostima svakog proizvoda definirane su na 100g, a uključuju sljedeće:

- energija
- masnoće
- zasićene masne kiseline
- ugljikohidrati
- šećeri
- bjelančevine
- sol

Osim navedenog, svaki proizvod mora imati i priloženu fotografiju, masu i neke dodatke oznake koje ga svrstavaju u različite kategorije. Neke od dodatnih oznaka su:

- "sadrži kikiriki"
- "sadrži gluten"
- "riba"
- "tjestenina"

Dijeta, koju nutricionist stvara, može se definirati s ograničenjima na određene proizvode, kategorije proizvoda, proizvode s nedopuštenim količinama sastojaka,

te dnevnim limitom za određene nutritivne vrijednosti proizvoda. Dijeta sadrži i opis od nutricionista koji ju je stvorio. Kuharice su skupine recepata koje stvaraju kulinarski entuzijasti i nose tematski naziv, primjerice "Variva". Stvaranjem kuharice u njoj se ne nalazi niti jedan recept, a recepte u kuharice također dodaju njihovi autori, kulinarski entuzijasti. Međutim, to je opcionalno, odnosno ne mora svaki recept biti u pripadnoj kuharici, no isto tako, jedan recept može biti u više različitih kuharica istog kulinarskog entuzijasta. Tako se, primjerice, recept za varivo od graška može nalaziti u kuharici "Brzi ručkovi", a istovremeno i u kuharici "Priprema jela unaprijed". Valja napomenuti da je kulinarski entuzijast koji stvara kuharicu i sprema recepte u nju sam odgovoran za tematsku povezanost recepata u kuharici. Tako kulinarski entuzijast u kuharicu "Brzi ručkovi" teoretski može staviti i recept za čokoladnu tortu. Svaki recept je napisan po istom kalupu: najprije su navedeni svi potrebni sastojci, zajedno s njihovom količinom, i ukupno vrijeme kuhanja, zatim slijede koraci pripreme jela, popraćeni slikama koje pobliže objašnjavaju postupak. Naposljeku je navedena veličina jedne porcije jela uz popratnu sliku gotovog jela. Klijent platforme može skenirati bar kodove sastojaka, ili, ako neki od sastojaka nema bar kod, unijeti ga ručno iz dostupnih kategorija, koje ima doma i pomoću kojih želi pripremiti jelo, a na temelju tih sastojaka i ograničenja koje postavlja dijeta registriranog korisnika, platforma će pronaći i poredati recepte po njihovoj prihvatljivosti. Nakon što klijent odabere i pripremi odabrano jelo, on ga može označiti kao pripravljenog u tom danu, na temelju čega se, kroz dulji vremenski period, generira statistika konzumiranih nutritivnih vrijednosti u tom periodu. Neregistriranim se korisnicima prilikom otvaranja početne stranice prikazuju najnovije kuharice objavljene od strane kulinarskih entuzijasta. S druge strane, registriranim korisnicima se na početnoj stranici prikazuju isprobani recepti, informacije o dijeti koju prate, nove kuharice i recepti od kulinarskih entuzijasta koje prate.

2.5 Moguće nadogradnje projektnog zadatka

Neke od mogućih nadogradnji projektnog zadatka za platformu KuhajIT uključuju:

- dodavanje video recepata
- chat u kojem se registrirani korisnici mogu dopisivati s nutricionistima, kulinarskim entuzijastima i ostalima
- obavještavanje korisnika kada neki od kulinarskih entuzijasta koje prate objave

novi recept ili kuharicu

- mogućnost ispisa željenog recepta
- "explore" dio na početnoj stranici registriranih korisnika, u kojem mogu otkriti neke nove kulinarske entuzijaste koji objavljaju recepte koji bi im se mogli svidjeti
- povezanost platforme s nekom internetskom trgovinom za kupnju namirnica, primjerice "Konzum Klik"

3. Specifikacija programske potpore

3.1 Funkcionalni zahtjevi

Dionici:

1. Administrator
2. Razvojni tim
3. Svi zainteresirani za kvalitetnu prehranu
4. Korisnici

Aktori i njihovi funkcionalni zahtjevi:

1. Neregistrirani korisnik (inicijator) može:
 - (a) pretraživati i pregledavati profile kulinarskih entuzijasta te njihove recepte i kuharice
 - (b) ostaviti anonimnu recenziju i ocjenu na recept
 - (c) poslati zahtjev za registraciju na platformu
2. Klijent (inicijator) može:
 - (a) mijenjati podatke na svom korisničkom profilu
 - (b) pretraživati i pregledavati profile kulinarskih entuzijasta te njihove recepte i kuharice
 - (c) ostaviti recenziju i ocjenu na recept pod svojim korisničkim imenom
 - (d) unositi proizvode koje ima doma i koji su spremni za uporabu u receptu
 - i. skeniranjem bar koda
 - ii. ručnim unosom iz dostupnih kategorija
 - (e) zatražiti od nutricionista da im složi dijetu
 - (f) označiti koje su recepte konzumirali u kojem danu
3. Kulinarski entuzijast (inicijator) može:
 - (a) obavljati iste akcije kao i klijent
 - (b) stvarati nove tematske kuharice i u njima recepte

4. Nutricionist (sudionik) može:

- (a) obavljati iste akcije kao i klijent
- (b) izrađivati dijete na temelju dostupnih informacija o proizvodima
- (c) unositi podatke o proizvodima
- (d) razvrstavati proizvode u kategorije s oznakama

5. Administrator (incijator) može:

- (a) potvrditi zahtjev za registraciju korisnika koji se želi registrirati kao ku-
linarski entuzijast ili nutricionist
- (b) vidjeti popis svih registriranih korisnika i njihovih osobnih podataka
- (c) brisati recenzije koje su u suprotnosti s pravilima korištenja aplikacije

3.1.1 Obrasci uporabe

Opis obrazaca uporabe

UC1 - Zahtjev za registraciju

- **Glavni sudionik:** Neregistrirani korisnik
- **Cilj:** Stvoriti korisnički profil
- **Sudionici:** Baza podataka, administrator
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
 1. Korisnik odabire za koju se ulogu želi registrirati
 2. Korisnik unosi podatke potrebne za registraciju
 3. Korisnik šalje zahtjev za registraciju
 4. Korisnik dobiva potvrdu od administratora da je uspješno registriran
- **Opis mogućih odstupanja:**
 - 2.a email ili korisničko ime koje je korisnik unio je već zauzeto ili u krivom obliku
 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
 - 3.a uvjeti za registraciju korisnika kao kulinarskog entuzijasta ili nutricionista nisu zadovoljeni
 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

UC2 - Prijava u sustav

- **Glavni sudionik:** Klijent
- **Cilj:** Uspješno se prijaviti u sustav
- **Sudionici:** Baza podataka
- **Preduvjet:** Registracija
- **Opis osnovnog tijeka:**
 1. Korisnik unosi podatke za prijavu
 2. Potvrda o ispravnosti unesenih podataka

3. Pristup korisničkim funkcijama
- **Opis mogućih odstupanja:**
 - 1.a Neispravni podaci za prijavu
 1. Sustav obaveštava korisnika o neuspjeloj prijavi i vraća ga na stranicu za prijavu

UC3 - Promjena osobnih podataka

- **Glavni sudionik: Svi registrirani korisnici (klijenti, kulinarski entuziasti, nutricionisti)**
- **Cilj: Promijeniti željene osobne podatke**
- **Sudionici: Baza podataka**
- **Preduvjet: Prijava**
- **Opis osnovnog tijeka:**
 1. Registrirani korisnik odabire opciju za promjenu podataka
 2. Registrirani korisnik mijenja željene podatke
 3. Registrirani korisnik sprema promjene
 4. Baza podataka se ažurira
- **Opis mogućih odstupanja:**
 - 3.a Registrirani korisnik ne spremi promjene
 1. Sustav obaveštava registriranog korisnika da nije spremio podatke prije izlaska iz prozora

UC4 - Pretraživanje profila kulinarskih entuzijasta

- **Glavni sudionik: Svi orisnici**
- **Cilj: Pretražiti profile kulinarskih entuzijasta**
- **Sudionici: Baza podataka**
- **Preduvjet: -**
- **Opis osnovnog tijeka:**
 1. Korisnik u tražilicu upisuje kategoriju po kojoj želi pretraživati
 2. Na ekran se izlistavaju profili koji odgovaraju upisanoj kategoriji

UC5 - Ostavljanje recenzije na recept

- **Glavni sudionik: Registrirani korisnik**
- **Cilj: Ostaviti recenziju na recept**
- **Sudionici: Baza podataka**
- **Preduvjet: Prijava**

- **Opis osnovnog tijeka:**

1. Registrirani korisnik odabire recept na kojem želi ostaviti recenziju
2. Registrirani korisnik ostavlja ocjenu ili recenziju na odabrani recept

UC6 - Odgovor autora na recenziju

- **Glavni sudionik: Kulinarski entuzijast**

- **Cilj: Odgovoriti na recenziju koju je na recept kulinarskog entuzijasta ostavio drugi korisnik**

- **Sudionici: Baza podataka**

- **Preduvjet: Prijava**

- **Opis osnovnog tijeka:**

1. Kulinarski entuzijast odabere recept na kojem je ostavljena recenzija
2. Kulinarski entuzijast odabere opciju odgovora na recenziju
3. Kulinarski entuzijast odgovara na recenziju
4. Kulinarski entuzijast objavljuje odgovor na recenziju

UC7 - Stvaranje nove kuharice

- **Glavni sudionik: Kulinarski entuzijast**

- **Cilj: Stvoriti novu tematsku kuharicu**

- **Sudionici: Baza podataka**

- **Preduvjet: Prijava**

- **Opis osnovnog tijeka:**

1. Kulinarski entuzijast odabire opciju za stvaranje nove kuharice
2. Kulinarski entuzijast stvara novu kuharicu
3. Baza podataka se ažurira

UC8 - Stvaranje novog recepta

- **Glavni sudionik: Kulinarski entuzijast**

- **Cilj: Stvoriti novi recept**

- **Sudionici: Baza podataka**

- **Preduvjet: Prijava**

- **Opis osnovnog tijeka:**

1. Kulinarski entuzijast odabire opciju za stvaranje novog recepta
2. Kulinarski entuzijast unosi sve potrebne podatke za stvaranje novog recepta
3. Kulinarski entuzijast stvara novi recept

4. Baza podataka se ažurira

UC9 - Dodavanje recepta u kuharicu

- **Glavni sudionik:** Kulinarski entuzijast
- **Cilj:** Dodati recept u kuharicu koja mu tematski odgovara
- **Sudionici:** Baza podataka
- **Preduvjet:** Prijava
- **Opis osnovnog tijeka:**
 1. Kulinarski entuzijast odabire recept koji želi dodati u kuharicu
 2. Kulinarski entuzijast odabire opciju dodavanja recepta u kuharicu
 3. Kulinarski entuzijast iz prikazane liste dostupnih kuharica odabire onu u koju želi dodati recept
 4. Kulinarski entuzijast dodaje recept u odabranoj kuharici
 5. Baza podataka se ažurira

UC10 - Brisanje kuharice

- **Glavni sudionik:** Kulinarski entuzijast
- **Cilj:** Obrisati kuharicu
- **Sudionici:** Baza podataka
- **Preduvjet:** Prijava
- **Opis osnovnog tijeka:**
 1. Kulinarski entuzijast odabire kuharicu koju želi obrisati
 2. Kulinarski entuzijast briše odabranoj kuharici, no ne i recepte koji se nalaze u njoj
 3. Baza podataka se ažurira

UC11 - Brisanje recepta iz baze

- **Glavni sudionik:** Kulinarski entuzijast
- **Cilj:** Obrisati recept iz svih kuharica i baze
- **Sudionici:** Baza podataka
- **Preduvjet:** Prijava
- **Opis osnovnog tijeka:**
 1. Kulinarski entuzijast odabire recept koji želi obrisati
 2. Kulinarski entuzijast briše odabrani recept
 3. Baza podataka se ažurira

UC12 - Brisanje recepta iz pojedine kuharice

- **Glavni sudionik:** Kulinarski entuzijast
- **Cilj:** Obrisati recept iz samo jedne kuvarice
- **Sudionici:** Baza podataka
- **Preduvjet:** Prijava
- **Opis osnovnog tijeka:**
 1. Kulinarski entuzijast odabire kuvaricu iz koje želi obrisati recept
 2. Kulinarski entuzijast odabire recept koji želi obrisati
 3. Kulinarski entuzijast briše recept iz kuvarice
 4. Baza podataka se ažurira

UC13 - Skeniranje bar koda proizvoda

- **Glavni sudionik:** Klijent
- **Cilj:** Unos proizvoda
- **Sudionici:** Baza podataka
- **Preduvjet:** Prijava
- **Opis osnovnog tijeka:**
 1. Klijent odabire opciju skeniranja bar koda proizvoda kojeg ima doma
 2. Klijent skenira proizvod
- **Opis mogućih odstupanja:**
 - 2.a Bar kod se ne može očitati
 1. Sustav obavještava korisnika da je skeniranje bar koda neuspješno i da pokuša ponovno ili ručno unese proizvod

UC14 - Ručni unos proizvoda

- **Glavni sudionik:** Klijent
- **Cilj:** Unos proizvoda
- **Sudionici:** Baza podataka
- **Preduvjet:** Prijava
- **Opis osnovnog tijeka:**
 1. Klijent odabire opciju ručnog unosa proizvoda kojeg ima doma
 2. Klijent ručno unosi proizvod
 3. Baza podataka se ažurira
- **Opis mogućih odstupanja:**
 - 2.a Proizvod nije prepoznat
 1. Sustav obavještava korisnika da proizvod nije prepoznat

UC15- Unos podataka o proizvodu

- **Glavni sudionik:** Nutricionist
- **Cilj:** Unos podataka o proizvodu
- **Sudionici:** Baza podataka
- **Preduvjet:** Prijava
- **Opis osnovnog tijeka:**
 1. Nutricionist odabire opciju unosa podataka o proizvodu
 2. Nutricionist unosi sve potrebne podatke o proizvodu
 3. Nutricionist odabire kojoj kategoriji proizvod pripada
 4. Baza podataka se ažurira
- **Opis mogućih odstupanja:**
 - 2.a Nutricionist nije unio neki od podataka
 1. Sustav obaveštava nutricionista da nije unio sve potrebne podatke i vraća ga na stranicu za unos podataka o proizvodu

UC16 - Označivanje recepta kao konzumiranog u određenom danu

- **Glavni sudionik:** Klijent
- **Cilj:** Generiranje statistike konzumiranih nutritivnih vrijednosti kroz vrijeme na temelju informacija o tome koji je recept konzumiran koji dan
- **Sudionici:** Baza podataka
- **Preduvjet:** Prijava
- **Opis osnovnog tijeka:**
 1. Klijent odabire recept koji je konzumirao
 2. Klijent odabire opciju odabira datuma kada je odabrani recept konzumiran
 3. Klijent unosi datum konzumiranja odabranog recepta
 4. Baza podataka se ažurira

UC17 - Brisanje proizvoda

- **Glavni sudionik:** Nutricionist
- **Cilj:** Brisanje proizvoda iz baze podataka
- **Sudionici:** Baza podataka
- **Preduvjet:** Prijava
- **Opis osnovnog tijeka:**
 1. Nutricionist pregledava kategorije proizvoda
 2. Nutricionist odabire kategoriju proizvoda kojeg želi obrisati
 3. Nutricionist odabire proizvod iz odabrane kategorije

4. Nutricionist briše odabrani proizvod
5. Baza podataka se ažurira
6. Otvara se stranica s pregledom svih kategorija proizvoda

UC18 - Brisanje korisničkog računa

- **Glavni sudionik:** Klijent
- **Cilj:** Obrisati svoj korisnički račun
- **Sudionici:** Baza podataka
- **Preduvjet:** Prijava
- **Opis osnovnog tijeka:**
 1. Klijent pregledava osobne podatke
 2. Klijent odabire opciju brisanja korisničkog računa
 3. Klijent briše svoj korisnički račun
 4. Baza podataka se ažurira
 5. Otvara se stranica za registraciju

UC19 - Izrada dijete

- **Glavni sudionik:** Nutricionist
- **Cilj:** Izraditi dijetu koju će klijenti pratiti
- **Sudionici:** Baza podataka
- **Preduvjet:** Prijava
- **Opis osnovnog tijeka:**
 1. Nutricionist odabire opciju izrade nove dijete
 2. Nutricionist upisuje sva ograničenja na proizvode i ostala ograničenja koja je zamislio u dijeti
 3. Nutricionist spremi novu dijetu, baza podataka se ažurira

UC20 - Potvrda zahtjeva za registracijom

- **Glavni sudionik:** Administrator
- **Cilj:** Odobriti registraciju kulinarskog entuzijasta ili nutricionista
- **Sudionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
 1. Administrator odabire zahtjev za registraciju koji želi pregledati
 2. Ako su uneseni podatci zadovoljavajući, administrator potvrđuje zahtjev za registracijom

- **Opis mogućih odstupanja:**

1.a Neispravni podaci za registraciju

1. Administrator odbija zahtjev, a sustav obavještava korisnika o neuspjeloj registraciji

UC21 - Prikaz popisa registriranih korisnika i njihovih podataka

- **Glavni sudionik: Administrator**

- **Cilj: Pristupiti podatcima svih registriranih korisnika**

- **Sudionici: Baza podataka**

- **Preduvjet: Autorizacija**

- **Opis osnovnog tijeka:**

1. Administrator šalje upit u bazu za podatcim svih registriranih korisnika
2. Administrator dobiva popis svih podataka.

UC22 - Brisanje recenzija

- **Glavni sudionik: Administrator**

- **Cilj: Ukloniti recenzije koje su u suprotnosti s pravilima korištenja aplikacije**

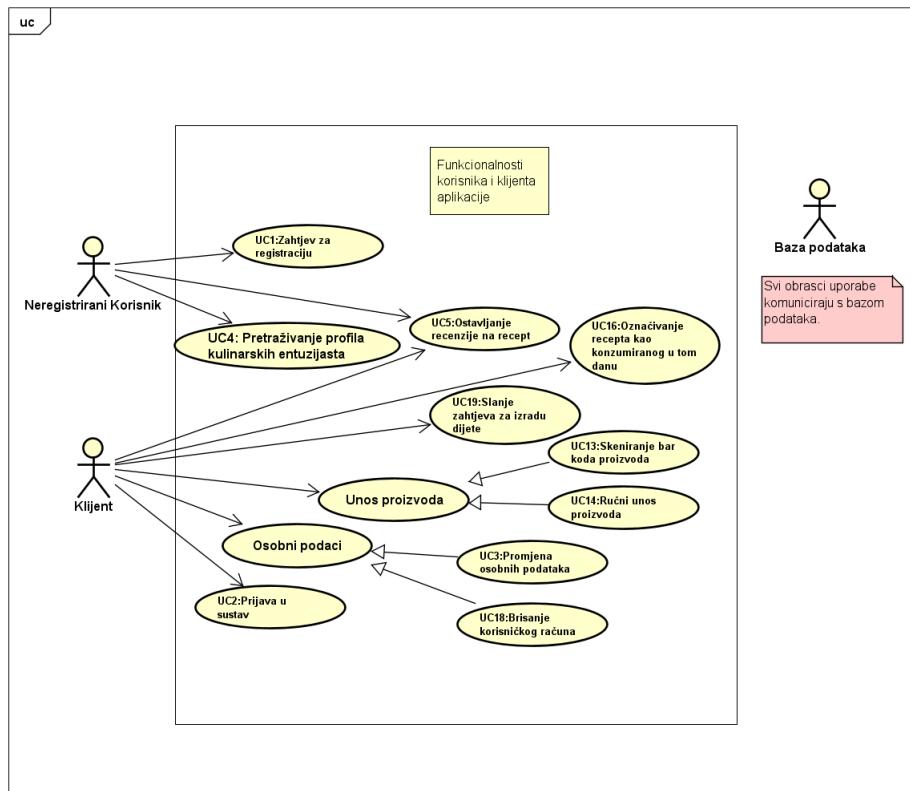
- **Sudionici: Baza podataka**

- **Preduvjet: Autorizacija**

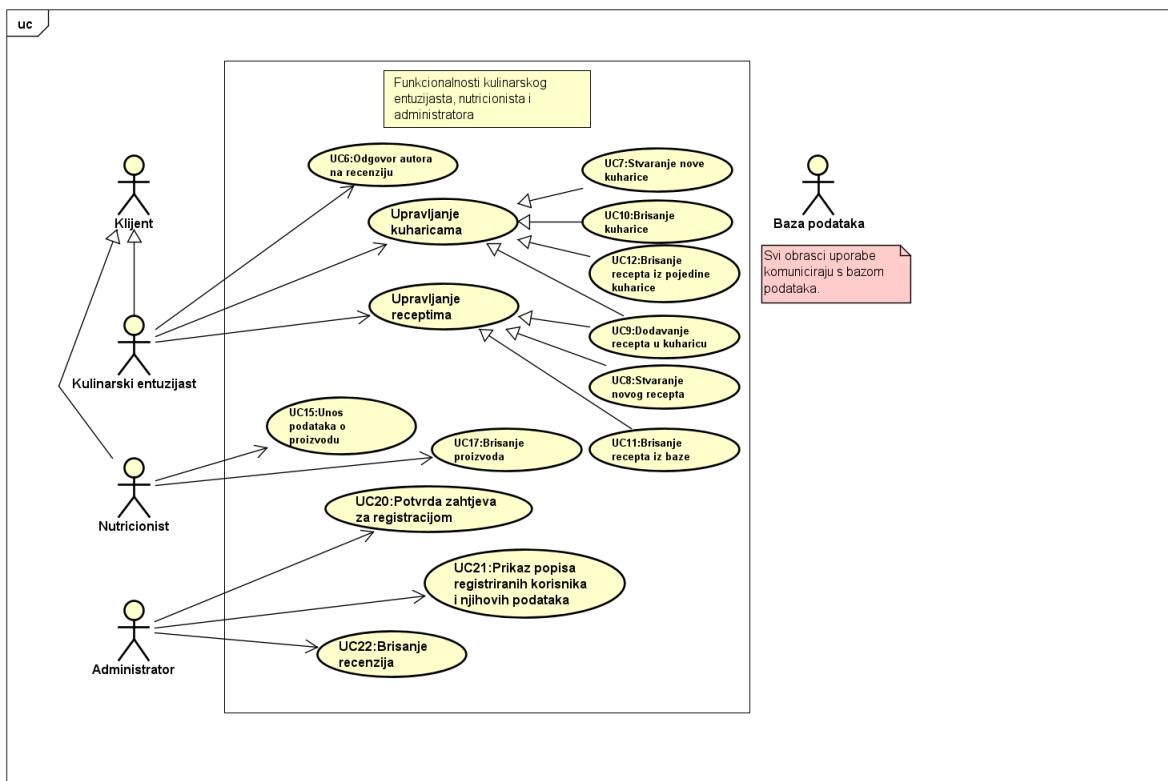
- **Opis osnovnog tijeka:**

1. Administrator odabire recenziju koju treba obrisati
2. Administrator briše odabranu recenziju
3. Baza podataka se ažurira

Dijagrami obrazaca uporabe



Slika 3.1: Funkcionalnosti korisnika i klijenta aplikacije

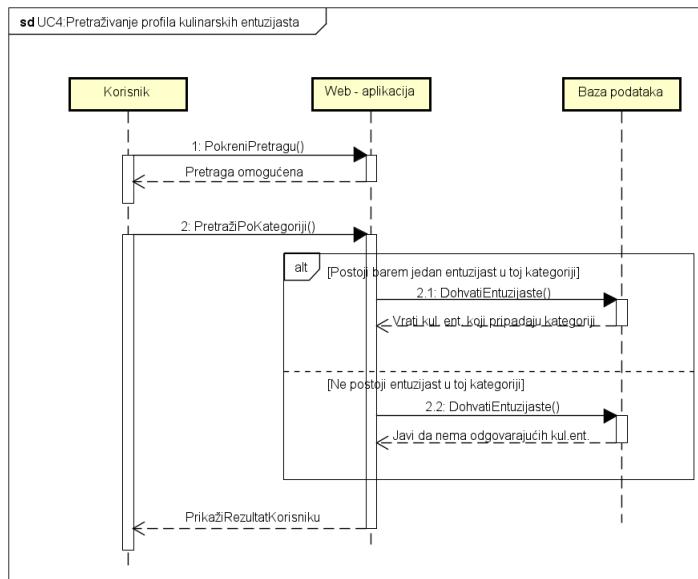


Slika 3.2: Funkcionalnosti kulinarskog entuzijasta, nutricionista i administratora aplikacije

3.1.2 Sekvencijski dijagrami

Obrazac uporabe UC4 - Pretraživanje profila kulinarskih entuzijasta

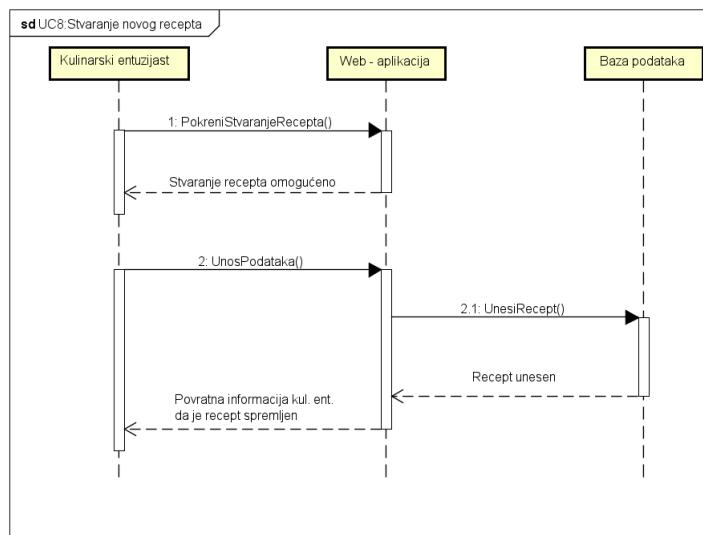
Korisnik šalje zahtjev za pretragom profila kulinarskih entuzijasta. Poslužitelj mu omogućava pristup i pretragu. Korisnik u tražilicu upisuje kategoriju po kojoj želi pretražiti kulinarske entuzijaste. Poslužitelj iz baze dohvata sve kulinarske entuzijaste koji odgovaraju upisanoj kategoriji i prikazuje ih korisniku.



Slika 3.3: Sekvencijski dijagram za UC4

Obrazac uporabe UC8 - Stvaranje novog recepta

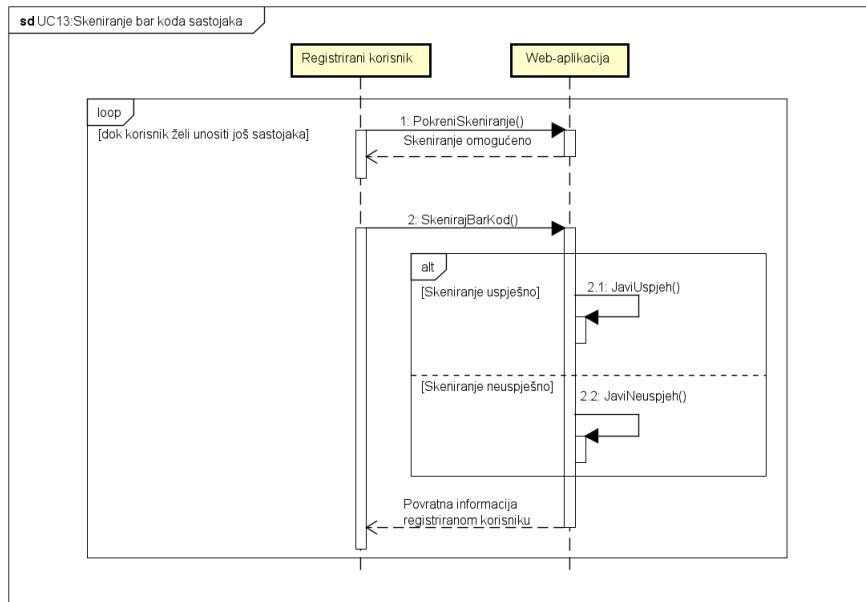
Kulinarski entuzijast šalje zahtjev za stvaranjem novog recepta. Poslužitelj mu omogućava pristup i stvaranje novog recepta. Kulinarski entuzijast upisuje sve podatke potrebne za stvaranje novog recepta: potrebne sastojke i njihovu količinu, korake pripreme, veličinu porcije, vrijeme kuhanja i galeriju popratnih slika. Nakon potvrde podataka, poslužitelj unosi novi recept u bazu podataka.



Slika 3.4: Sekvencijski dijagram za UC8

Obrazac uporabe UC13 - Skeniranje bar koda proizvoda

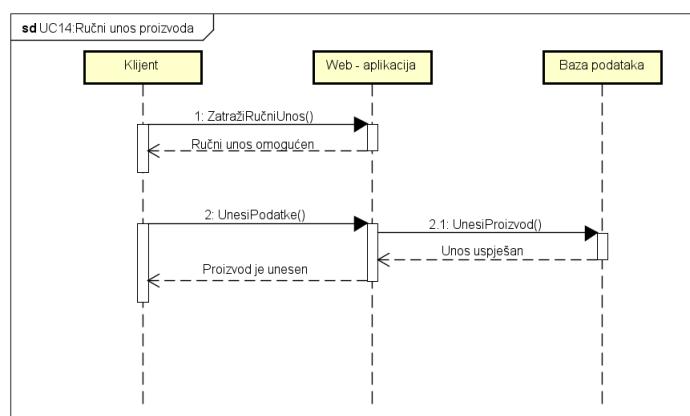
Klijent šalje zahtjev za skeniranjem bar koda proizvoda kojeg ima doma. Poslužitelj mu omogućava pristup i skeniranje bar koda proizvoda. Klijent skenira bar kod proizvoda, i ako je bar kod čitljiv i proizvod prepoznat, poslužitelj unosi skenirani proizvod u bazu podataka.



Slika 3.5: Sekvencijski dijagram za UC13

Obrazac uporabe UC14 - Ručni unos proizvoda

Klijent šalje zahtjev za ručnim unosom proizvoda kojeg ima doma. Poslužitelj mu omogućava pristup i unos proizvoda. Klijent unosi podatke o proizvodu ručno, primjerice njegov naziv, naziv proizvođača, masu, energetsku vrijednost itd. Nakon potvrde unosa, poslužitelj unosi novi recept u bazu podataka.



Slika 3.6: Sekvencijski dijagram za UC14

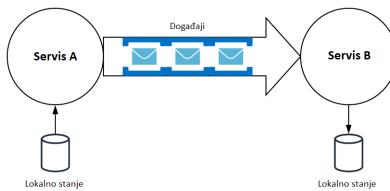
3.2 Ostali zahtjevi

- Sustav treba podržavati rad više korisnika u stvarnom vremenu.
- Sustav treba implementirati kao web aplikaciju koristeći objektno-orientirane jezike.
- Sustav mora biti prilagođena veličini ekrana na kojem je prikazana.
- Sustav mora biti u potpunosti responzivan, bez dugog čekanja na podatke.
- Sustav mora biti što jednostavniji i prilagođeniji korisniku za korištenje.
- Sustav mora podržavati znakove hrvatske abecede pri unosu i prikazu sadržaja.
- Sustav mora imati omogućen pristup iz javne mreže.
- Sustav mora biti otporan na neispravno korištenje korisničkog sučelja, u smislu da neispravno korištenje ne smije narušiti njegovu funkcionalnost.
- Sustav mora ograničiti korisnika na pristup jedino onim resursima kojima ima pristup.

4. Arhitektura i dizajn sustava

Za ostvarenje naše aplikacije odabrali smo arhitekturu zasnovanu na događajima. Prednosti ovog tipa arhitekture su mnoge:

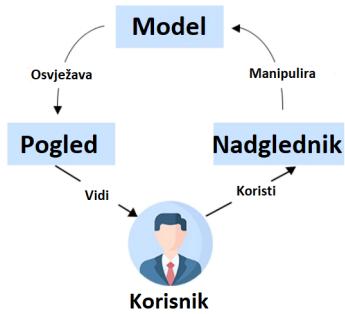
- svaki od servisa arhitekture može biti neovisan entitet, što povećava fleksibilnost
- događaji se mogu pratiti u stvarnom vremenu, što olakšava analizu sustava
- događaji predstavljaju promjene stanja pa je komponentama omogućeno da reagiraju na te promjene



Slika 4.1: Prikaz arhitekture zasnovane na događajima

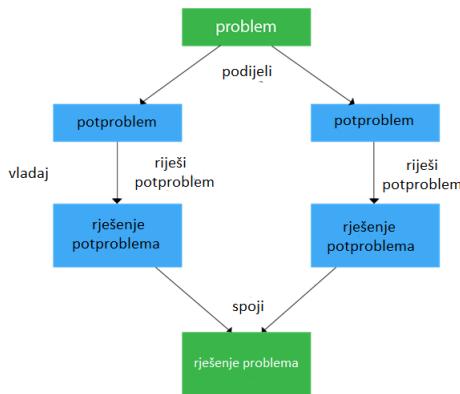
Specifično, radi se o MVC (Model-View-Controller) obrascu, koji odvaja korisničko sučelje od ostatka sustava. On dodatno smanjuje međuvisnost U/I sučelja i ostatka sustava. Sastoji se od tri dijela:

- model - sadrži razrede čiji se objekti obrađuju različitim operacijama. Odgovoran je za održavanje dosljednosti podataka te obavljanje poslovnih operacija i pravila
- view (pogled) - odgovoran za prezentaciju podataka korisnicima (korisničko sučelje, web-aplikacija, grafovi...). On ne bi trebao sadržavati nikakvu poslovnu logiku, već prikazuje informacije korisnicima na razumljiv i privlačan način
- controller (nadglednik) - sadrži razrede koji upravljaju i rukuju korisničkom interakcijom s pogledom i modelom



Slika 4.2: Prikaz načina rada MVC obrasca

Po principu oblikovanja arhitekture odabrali smo *Podijeli pa vladaj* arhitekturu, kako bismo se unutar tima mogli podijeliti u manje timove koji rade na određenim problemima i kako bismo, ako to bude bilo potrebno, jednostavno zamijenili željene dijelove bez opsežne intervencije u cijeli sustav.



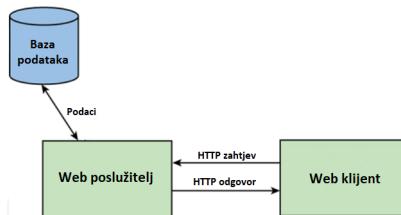
Slika 4.3: Prikaz principa djelovanja "podijeli pa vladaj"

Arhitektura našeg sustava dijeli se na tri komponente:

- Web preglednik - softverska aplikacija koja omogućuje korisnicima pregleđavanje i interakciju sa svim sadržajima Interneta. Glavna funkcija web preglednika je prikazivanje web stranica koje su oblikovane u obliku programskog koda na korisniku jasan način. Neki poznati web-preglednici uključuju Google Chrome, Mozilla Firefox, Opera... Web preglednik služi i kao medijator između korisnika, koji šalje zahtjeve, i web poslužitelja, koji prima zahtjeve i na njih odgovara.
- Web poslužitelj - program koji proslijeđuje web sadržaje web klijentu/pregledniku na njegov zahtjev putem protokola HTTP. On od klijenta dobije zahtjev za podacima, pristupa bazi podataka (kojoj također ima pristup), dohvaća tražene

podatke iz baze te ih vraća klijentu u obliku HTTP odgovora. Vraćeni podaci se zatim prikazuju klijentu (ako nije došlo do poteškoća).

- Baza podataka - omogućuje organiziranu pohranu podataka koje je potrebno pohraniti (za KuhajIT su to podaci o korisničkim profilima, recepti, kuharice, osvrti na recepte, proizvodi i ostali)



Slika 4.4: Prikaz principa djelovanja podsustava arhitekture

Čitavi backend naše aplikacije izrađen je u programskom jeziku *Java*, u radnom okviru *Spring Boot*. Razvojno okruženje korišteno za razvoj backenda je *Eclipse IDE*. Frontend je izrađen u programskom jeziku *JavaScript*, u biblioteci *React*. Razvojno okruženje korišteno za razvoj frontenda je *Visual Studio Code*.

4.1 Baza podataka

Za ostvarenje naše aplikacije KuhajIT odabrali smo relacijsku bazu podataka PostgreSQL, u alatu *PGadmin4*, koja svojom jednostavnom i lako razumljivom strukturom olakšava pregled i upravljanje podacima. Ključne komponente relacijske baze podataka su entiteti (tablice okarakterizirane imenom i skupom atributa) te veze među entitetima (strukture koje povezuju podatke iz različitih entiteta pomoću ključeva).

Entiteti naše baze podataka su:

- User
- Role
- Recipe
- Recipe Ingredient
- Ingredient

- Cookbook
- Cookbook Recipe
- Review
- Response
- Image
- Recipe Image
- Diet
- Diet Ingredient
- Diet allowed recipes
- Consumed recipes
- Step of making
- Category
- Label

4.1.1 Opis tablica

User Entitet *User* sadrži atribute važne za svakog registriranog korisnika aplikacije KuhajIT. Ti atributi su: ID korisnika, korisničko ime, lozinku, ime, prezime, email, uloga, biografija, ID osobne fotografije koju je priložio, oznaka potvrđenosti i ID dijete koje se pridržava. U vezi je *One-to-Many* s entitetom *Review* preko korisničkog ID-ja, *One-to-Many* s entitetom *Cookbook* preko korisničkog ID-ja, *Recipe* preko korisničkog ID-ja, *One-to-Many* s entitetom *Response* preko korisničkog ID-ja, *One-to-One* s entitetom *Image* preko ID-ja učitane fotografije, *Many-to-One* s entitetom *Role* preko ID-ja uloge, *Many-to-One* s entitetom *Diet* preko ID-ja dijete i *One-to-Many* s entitetom *Consumed Recipes* preko ID-ja korisnika.

User		
id	INT	ID korisnika, sekvencijski generiran.

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

User		
username	VARCHAR	Korisničko ime korisnika, koje mora biti jedinstveno, korisnik si ga bira sam pri registraciji, no može ga promijeniti.
password	VARCHAR	Lozinka za pristup korisničkom računu.
name	VARCHAR	Ime korisnika
surname	VARCHAR	Prezime korisnika
email	VARCHAR	Ako se korisnik želi registrirati kao kulinarski entuzijast/nutricionist, mora priložiti email, inače atribut poprima NULL vrijednost.
confirmed	BOOLEAN	Oznaka je li administrator potvrdio registraciju kulinarskog entuzijasta/nutricionista.
biography	VARCHAR	Ako se korisnik želi registrirati kao kulinarski entuzijast/nutricionist, mora priložiti kratku biografiju, inače atribut poprima NULL vrijednost.
image_id	INT	ID osobne fotografije koju je korisnik koji želi biti kulinarski entuzijast ili nutricionist učitao.
role_id	INT	ID uloge koju korisnik registracijom želi posjedovati (nutricionist, kulinarski entuzijast ili klijent).
diet_id	INT	ID dijete koje se korisnik pridržava.

Role Entitet *Role* je šifarnik uloga koje korisnik registracijom može dobiti. To su:

- klijent
- nutricionist
- kulinarski entuzijast

Uz navedene, u tablici se nalazi i uloga administratora, koju nije moguće dobiti registracijom u sustav, već je predefiniran. U vezi je *One-to-Many* s entitetom *User*, preko ID-ja uloge.

Role		
id	INT	Jedinstveni ID uloge, sekvencijski generiran.
name	VARCHAR	Naziv uloge.

Recipe Entitet *Recipe* sadrži atribute važne za svaki recept objavljen na web aplikaciji KuhajIT. Ti atributi su: ID recepta, ime recepta, vrijeme pripreme, ID autora recepta, veličina porcije, ID kategorije kojoj pripada recept. U vezi je *One-to-Many* s entitetom *Recipe Image* preko ID-ja recepta, *One-to-Many* s entitetom *Cookbook Recipe* preko ID-ja recepta, *One-to-Many* s entitetom *Recipe Ingredient* preko ID-ja recepta, *Many-to-One* s entitetom *User* preko ID-ja korisnika i *Many-to-One* s entitetom *Category*.

Recipe		
id	INT	Jedinstveni ID recepta, sekvencijski generiran.
name	VARCHAR	Ime recepta.
cook_time	INT	Vrijeme pripreme recepta.
portion_size	INT	Veličina porcije pripremljenog recepta.
steps_of_making	VARCHAR	Koraci pripreme recepta.
creator_id	INT	Korisnički ID autora recepta.
category_id	INT	ID kategorije kojoj recept pripada.

Recipe Ingredient Entitet *Recipe Ingredient* sadrži atribute važne za pohranu sastojaka koji se koriste u pojedinom receptu objavljenom na web aplikaciji KuhajIT. Ti atributi su: ID sastojka recepta, količina sastojka, ID recepta i ID sastojka. U vezi je *Many-to-One* s entitetom *Recipe* preko ID-ja recepta i *Many-to-One* s entitetom *Ingredient*.

Recipe Ingredient		
ingredient_id	INT	ID sastojka.
recipe_id	INT	ID recepta.
quantity	INT	Količina sastojka potrebnog za recept.

Ingredient Entitet *Ingredient* sadrži atribute važne za svaki sastojak. Ti atributi su: ID sastojka i ime sastojka. U vezi je *One-to-Many* s entitetom *Recipe Ingredient* preko ID-ja sastojka, *Many-to-One* s entitetom *Image* preko ID-ja fotografije i *Label* preko ID-ja oznake.

Ingredient		
id	INT	Jedinstveni ID sastojka, sekvencijski generiran.
name	VARCHAR	Naziv sastojka.
image_id	INT	ID fotografije sastojka.
label_id	INT	ID oznake sastojka.

Cookbook Entitet *Cookbook* sadrži atribute važne za svaku kuharicu stvorenu od kulinarskog entuzijasta na web aplikaciji KuhajIT. Ti atributi su: ID kuharice, ID kategorije kuharice, naziv kuharice i ID autora. U vezi je *One-to-Many* s entitetom *Cookbook Recipe* preko ID-ja kuharice, *Many-to-One* s entitetom *User* preko ID-ja korisnika (kulinar skog entuzijasta koji ju je stvorio) i *Many-to-One* s entitetom *Category* preko ID-ja kategorije.

Cookbook		
id	INT	Jedinstveni ID kuharice, sekvencijski generiran.
category	VARCHAR	Kategorija kuharice.
name	VARCHAR	Naziv kuharice.
creator_id	INT	Korisnički ID autora kuharice.
category_id	INT	ID kategorije kuharice.

Cookbook Recipe Entitet *Cookbook Recipe* sadrži atribute važne za pohranu recepta u pojedinu kuharicu na web aplikaciji KuhajIT. Ti atributi su: ID kuharice

i ID recepta. U vezi je *Many-to-One* s entitetom *Cookbook* preko ID-ja kuharice i *Many-to-One* s entitetom *Recipe* preko ID-ja recepta koji se u kuharici nalazi.

Cookbook Recipe		
cookbook_id	INT	ID kuharice.
recipe_id	INT	ID recepta.

Review Entitet *Review* sadrži atributе važne za svaku recenziju ostavljenu na recept na web aplikaciji KuhajIT. Ti atributi su: ID recenzije, ocjena recepta dodijeljena u recenziji, poruka ostavljena u recenziji, ID autora recenzije i ID recepta na koji je recenzija ostavljena. U vezi je *One-to-Many* s entitetom *Cookbook Recipe* preko ID-ja kuharice i *Many-to-One* s entitetom *User* preko ID-ja korisnika koji je ostavio recenziju, *Many-to-One* s entitetom *Recipe* preko ID-ja recepta na koji je recenzija ostavljena i *One-to-One* s entitetom *Response* preko ID-ja recenzije.

Review		
id	INT	Jedinstveni ID recenzije, sekvencijski generiran.
mark	INT	Ocjena ostavljena u recenziji.
message	VARCHAR	Poruka ostavljena u recenziji.
creator_id	INT	Korisnički ID autora recenzije, ako je recenziju ostavio neregistrirani korisnik, poprima vrijednost NULL.
recipe_id	INT	ID recepta na kojeg je recenzija ostavljena.
response_id	INT	ID odgovora na recenziju.

Response Entitet *Response* sadrži atributе važne za svaki odgovor na recenziju ostavljenu na recept na web aplikaciji KuhajIT. Ti atributi su: ID odgovora, poruka ostavljena u odgovoru, ID autora odgovora (kulinarski entuzijast na čiji je recept ostavljena recenzija) i ID recenzije na koju je odgovor ostavljen. U vezi je *One-to-One* s entitetom *Review* preko ID-ja recenzije i *Many-to-One* s entitetom *User* preko ID-ja korisnika koji odgovara na recenziju.

Response		
id	INT	Jedinstveni ID odgovora na recenziju, sekvensijski generiran.
message	VARCHAR	Poruka ostavljena u odgovoru na recenziju.
creator_id	INT	Korisnički ID autora odgovora na recenziju.
review_id	INT	ID recenzije na koju autor recepta odgovara.

Image Entitet *Image* sadrži atribute važne za svaku fotografiju učitanu u sklopu recepta na web aplikaciji KuhajIT. Ti atributi su: ID fotografije i URL fotografije.

Image		
id	INT	Jedinstveni ID učitane fotografije, sekvensijski generiran.
url	VARCHAR	URL učitane fotografije.
description	VARCHAR	Opis učitane fotografije.

Recipe Image Entitet *Recipe Image* sadrži referencu na slike koje je kulinarski entuzijast učitao pri stvaranju recepta. U vezi je *One-to-One* s entitetom *Image* i *Many-to-One* s entitetom *Recipe*.

Recipe Image		
image_id	INT	ID učitane fotografije.
recipe_id	INT	ID recepta kojem učitana fotografija pripada.

Diet Entitet *Diet* sadrži atribute važne za svaku dijetu koju kreira nutricionist. Ti atributi su: ID dijete, naziv dijete, opis dijete i ID nutricionista koji ju je stvorio. U vezi je *Many-to-One* s entitetom *User* preko ID-ja korisnika i *One-to-Many* s entitetom *Diet Ingredient* preko ID-ja dijete..

Diet		
id	INT	ID stvorene dijete.
name	VARCHAR	Naziv dijete.
description	VARCHAR	Opis dijete.
creator_id	INT	ID nutricionista koji je stvorio dijetu.

Diet Ingredient Entitet *Diet Ingredient* sadrži atribute važne za pohranu sastojaka koji su dozvoljeni i preporučeni za konzumiranje u pojedinoj dijeti objavljenoj na web-aplikaciji KuhajIT. Ti atributi su: ID dijete, ID sastojka i maksimalna dopuštena gramaza sastojka. U vezi je *Many-to-One* s entitetom Diet preko ID-ja dijete i *Many-to-One* s entitetom Ingredient preko ID-ja sastojka.

Diet Ingredient		
diet_id	INT	ID dijete.
ingredient_id	INT	ID sastojka.
max_amount_grams	INT	Maksimalna dopuštena gramaza .

Diet allowed recipes Entitet *Diet allowed recipes* sadrži atribute važne za pohranu recepata koji su dozvoljeni za konzumiranje u pojedinoj dijeti objavljenoj na KuhajIT web-aplikaciji. Ti atributi su: ID recepta i ID dijete. U vezi je *Many-to-One* s entitetom Diet preko ID-ja dijete i *Many-to-One* s entitetom Recipe.

Diet allowed recipes		
diet_id	INT	ID dijete.
recipe_id	INT	ID recepta.

Consumed recipes Entitet *Consumed recipes* sadrži atribute važne za pohranu recepata konzumiranih od registriranog korisnika. Ti atributi su: ID recepta koji je konzumiran, ID korisnika koji ga je konzumirao te datum konzumiranja. U vezi je *Many-to-One* s entitetom Recipe preko ID-ja recepta te *Many-to-One* s entitetom User preko ID-ja korisnika.

Consumed recipes		
recipe_id	INT	ID recepta.
user_id	INT	ID korisnika.
date	DATE	Datum unosa.

Step of making Entitet *Step of making* sadrži atribute važne za korak priprave recepta. Ti atributi su: ID koraka, tekstualni opis koraka, redni broj koraka, ID fotografije koja odgovara tom koraku i ID recepta na koji se korak odnosi. U vezi je *Many-to-One* s entitetom *Recipe* preko ID-ja recepta i *Many-to-One* s entitetom *Image* preko ID-ja fotografije.

Step of making		
id	INT	ID koraka izrade.
description	VARCHAR	Tekstualni opis koraka.
step_num	INT	Redni broj koraka.
image_id	INT	ID fotografije koraka.
recipe_id	INT	ID recepta na koji se korak odnosi.

Category Entitet *Category* sadrži atribute važne za pohranu kategorija recepata i kuharica. Ti atributi su: ID kategorije i naziv kategorije. U vezi je *One-to-Many* s entitetom *Recipe* preko ID-ja kuharice i *One-to-Many* s entitetom *Cookbook* preko ID-ja kuharice.

Category		
id	INT	ID kategorije.
name	VARCHAR	Naziv kategorije.

Label Entitet *Label* sadrži atribute važne za pohranu oznaka proizvoda koje unosi nutricionist. Ti atributi su: ID oznake i naziv oznake. U vezi je *One-to-Many* s entitetom *Ingredient* preko ID-ja oznake.

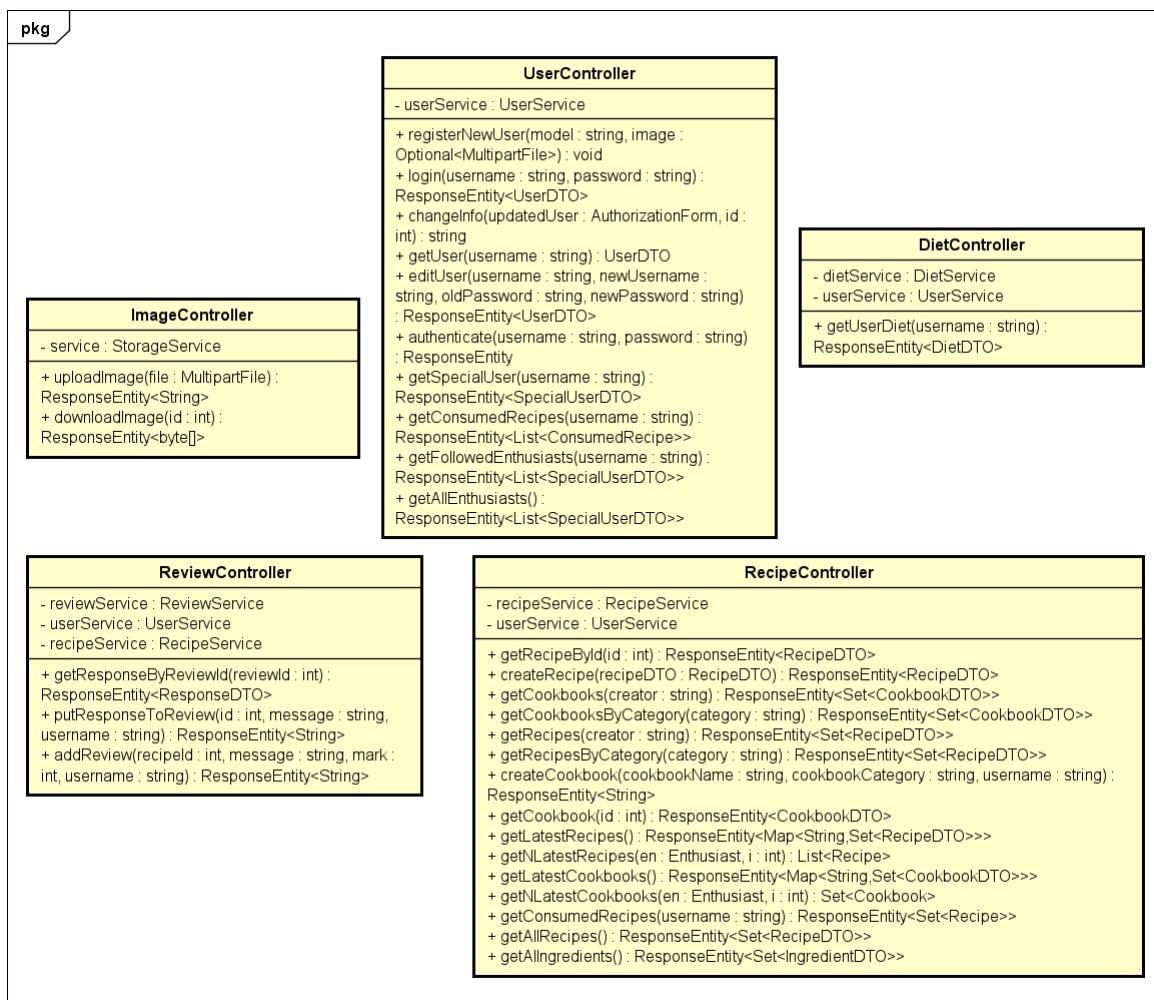
Label		
id	INT	ID oznake.
name	VARCHAR	Naziv oznake.



Slika 4.5: Prikaz ER dijagrama baze podataka

4.2 Dijagram razreda

Na slikama 4.6, 4.7, 4.8, 4.9 i 4.10 prikazani su dijagrami razreda koji backend arhitekturu ostvarenog sustava. Dijagram razreda na slici 4.6 prikazuje razred koji nasljeđuje razred *Controller*. Kontroleri su odgovorni za prihvatanje zahtjeva od klijenta, izvršavanje određene poslovne logike i generiranje odgovora koji se šalje natrag klijentu. Drugim riječima, kontroleri obrađuju HTTP zahtjeve i odgovaraju na njih.



Slika 4.6: Dijagram razreda - dio Controllers

Dijagram razreda na slici 4.7 prikazuje *Models* dio razreda.

Razredi *Enthusiast* i *Nutritionist* specifikacija su razreda *SpecialUser*, stoga nasljeđuju javne metode i atribute tog razreda, a usto sadrže i metode specifične ulozi.

Razredi *SpecialUser*, *Client* i *Administrator* specifikacija su razreda *User*, stoga nasljeđuju javne metode i atribute tog razreda.

Razred *User* predstavlja korisnika koji se može prijaviti kao klijent, kulinarski entuzijast ili nutricionist te, sukladno svojoj ulozi, obavljati različite funkcije opisane u Specifikaciji programske potpore.

Instanca razreda *User*, ako se radi o registriranom kulinarskom entuzijastu, može stvarati recepte, koji pripadaju razredu *Recipe*, i kuharice, koje pripadaju razredu *Cookbook*.

Razred *Recipe* sadrži sve metode potrebne za stvaranje i objavljivanje recepta.

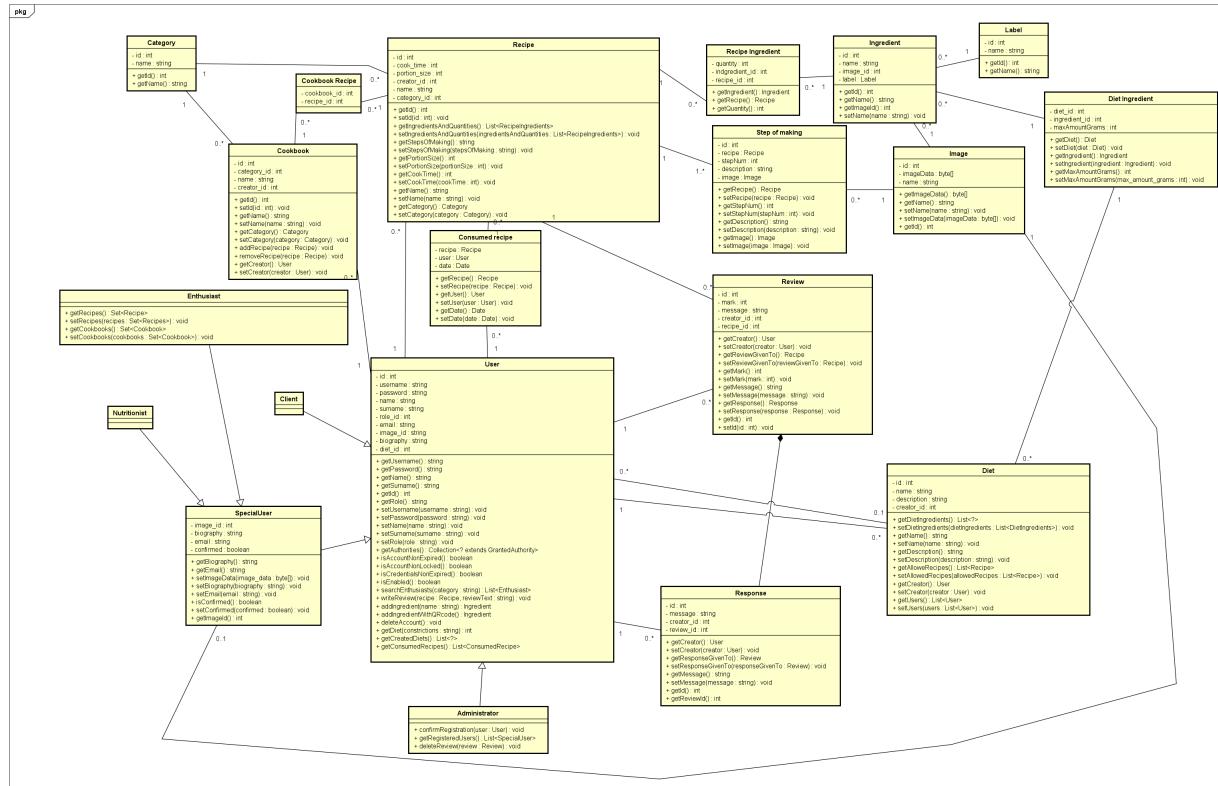
Razred *Cookbook* sadrži sve metode potrebne za stvaranje tematske kuharice i spremanje recepata u istu.

Razred *Image* sadrži atribute važne za pohranu slika.

Razred *Ingredient* sadrži atribute i metode važne za pohranu i dohvaćanje proizvoda i sastojaka.

Razred *Review* sadrži atribute i metode važne za ostavljanje recenzije na recept kulinarskog entuzijasta.

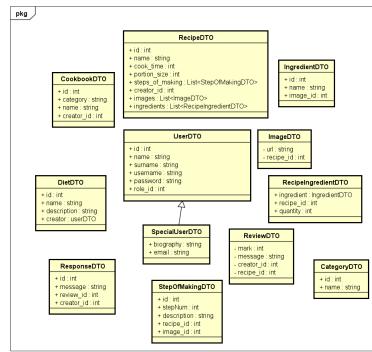
Razred *Response* sadrži atribute i metode važne za odgovaranje kulinarskog entuzijasta na recenziju ostavljenu na njegov recept. Razred *Diet* sadrži atribute i metode važne za stvaranje dijete od nutricionista. Razred *Step of making* sadrži atribute i metode važne za svaki korak izrade recepta od kulinarskog entuzijasta. Razredi *Consumed Recipe*, *Cookbook Recipe*, *Recipe Ingredient* i *Diet Ingredient* predstavljaju vezne klase između klasa koje povezuju (*Many-to-Many* veze u bazi podataka).



Slika 4.7: Dijagram razreda - dio Models

Dijagram razreda na slici 4.8 prikazuje *Data Transfer Object* dio razreda.

DTO su objekti koji prenose podatke između sustava. Konkretno, kada dođe do zahtjeva za podacima od korisnika na frontendu, DTO razredi koriste se za prijenos tih podataka iz backenda na frontend, koji se zatim prikazuju korisniku koji je poslao zahtjev za njima.

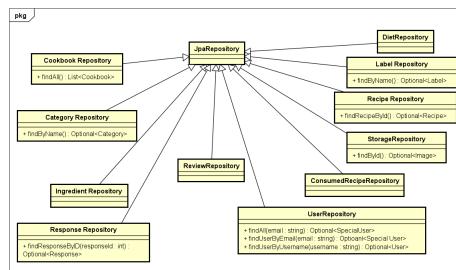


Slika 4.8: Dijagram razreda - dio DTO

Dijagram razreda na slici 4.9 prikazuje *Repository* dio razreda.

Sučelje *Repository* u Spring Boot-u sadrži deklaracije metoda za osnovne operacije

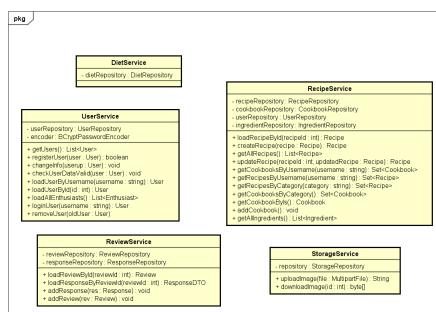
vezane uz pristup podacima (čitanje, pisanje, ažuriranje i brisanje).



Slika 4.9: Dijagram razreda - dio Repository

Dijagram razreda na slici 4.10 prikazuje *Service* dio razreda.

Sučelje servisa definira metode koje su dostupne ostatku aplikacije za izvršavanje implementiranih funkcionalnosti. Često komunicira s metodama iz sučelja *Repository* za pristup podacima, umjesto da izravno komunicira s bazom.



Slika 4.10: Dijagram razreda - dio Service

4.3 Dijagram stanja

Na slici 4.9 prikazan je dijagram stanja aplikacije implementirane aplikacije KuhajIT. Dijagram stanja je dijagram kojim se prikazuje diskretno ponašanje objekta ili sustava putem prelazaka između konačnog broja stanja. Koristi se za modeliranje ponašanja entiteta tijekom vremena, naglašavajući odgovor na događaje i okidače.

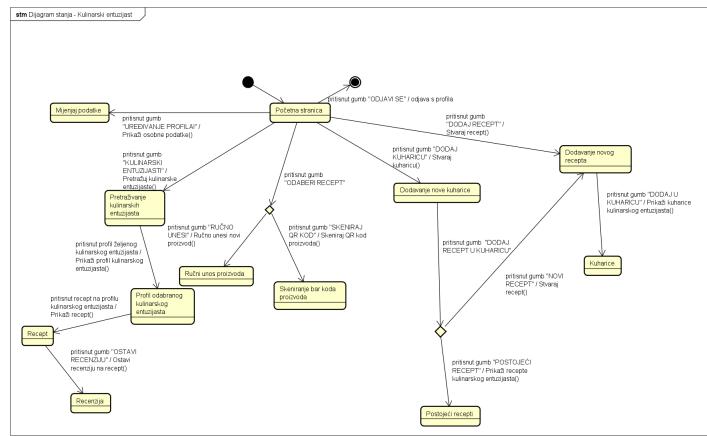
Mi smo odlučili prikazati dijagram stanja za kulinarskog entuzijasta, pošto on ima najviše funkcionalnosti koje su zajedničke svim registriranim korisnicima (klijentima).

Prijavljenom korisniku se na početnoj stranici prikazuju isprobani recepti, informacije o dijeti koju prate, nove kuharice i recepti od drugih kulinarskih entuzijasta koje prate. Također im se nudi sljedeće opcije:

- prikazivanje osobnih podataka
- dodavanje nove kuharice
- dodavanje novog recepta
- pretraživanje ostalih kulinarskih entuzijasta
- odabir recepta / skeniranje QR koda proizvoda
- odjava

Pri prikazivanju osobnih podataka, klijentu se nudi opcija uređivanja istih. Pri dodavanju nove kuharice, kulinarski entuzijast može odabrati želi li odmah dodati i recept u nju, i ako želi, odabrati hoće li to biti neki od njegovih već objavljenih recepata ili želi stvoriti novi recept. Pri dodavanju novog recepta, kulinarski entuzijast ima priliku odabrati želi li recept dodati nekoj od svojih postojećih kuharica. Pri dodavanju novog proizvoda, klijent može odabrati želi li skenirati bar kod proizvoda ili ga želi ručno unijeti.

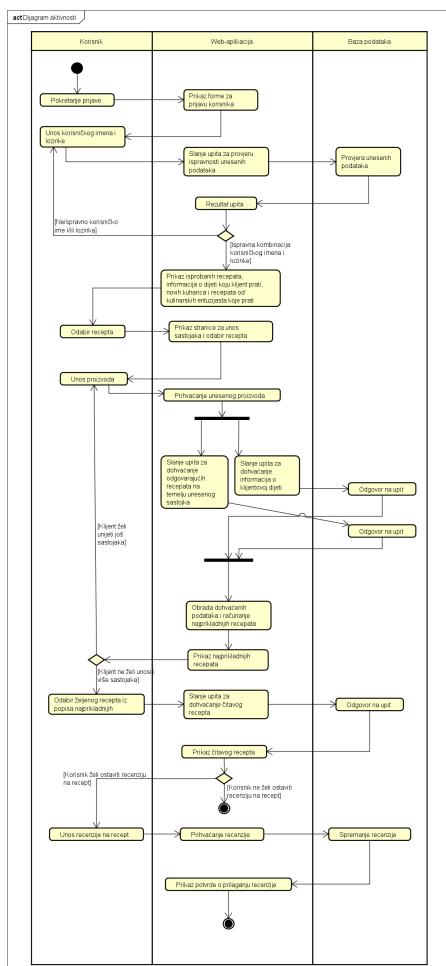
Opcije odjave i prikaza osobnih podataka su svakom prijavljenom korisniku omogućene u bilo kojem trenutku korištenja aplikacije.



Slika 4.11: Dijagram stanja - Kulinarski entuzijast

4.4 Dijagram aktivnosti

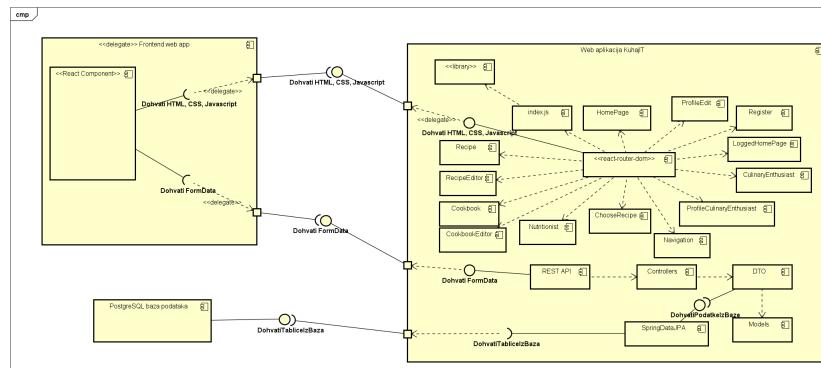
Na slici 4.10 prikazan je dijagram aktivnosti aplikacije KuhajIT. Dijagrami aktivnosti upotrebljavaju se za modeliranje i grafički prikaz dinamičkog ponašanja sustava. Konkretno, na ovom je dijagramu prikazana aktivnost prijave klijenta, unos proizvoda (ručno ili skeniranjem bar koda) te izlistavanje najprikladnijih recepata na temelju unesenih proizvoda i dijete koje se prijavljeni korisnik pridržava.



Slika 4.12: Dijagram aktivnosti

4.5 Dijagram komponenti

Na slici 4.11 prikazan je dijagram komponenti. Dijagrami komponenti koriste se za modeliranje arhitekture programskega sustava na visokoj razini. Pravljaju jasan i sažet način vizualizacije različitih komponenti ili građevnih blokova sustava i njihovih odnosa.



Slika 4.13: Dijagram komponenti

5. Implementacija i korisničko sučelje

5.1 Korištene tehnologije i alati

Za međusobnu komunikaciju tijekom izrade naše KuhajIT aplikacije korištene su dvije aplikacije: [Discord](#)¹ i [WhatsApp](#)².

WhatsApp je aplikacija za razmjenu poruka koja omogućava korisnicima slanje tekstualnih poruka, medijskih datoteka (poput slika, videa i audio zapisa), te poziva i video poziva putem internetskog povezivanja. Mi smo ju koristili za interno dogovaranje o onome što je iduće potrebno obaviti te o terminima sastanaka uživo.

Discord je platforma za komunikaciju putem interneta koja je prvobitno dizajnirana za zajednice igrača, ali se brzo proširila na različite druge vrste korisnika. Omogućava korisnicima stvaranje privatnih ili javnih "servera" (prostorija) gdje se mogu razgovarati putem teksta, glasa ili videa. Mi smo ju koristili za hitne sastanke u slučaju da je netko od članova pri izradi naišao na problem bilo kakve vrste i trebala mu je asistencija drugog člana tima.

Upravljanje izvornim kodom ostvareno je sustavom [Git](#)³, a udaljeni repozitorij čitavog projekta dostupan nam je bio na web platformi [GitHub](#)⁴.

Git je sustav za upravljanje verzijama koji se široko koristi u razvoju softvera. Omogućava programerima i timovima praćenje promjena u izvornom kodu tijekom vremena, praćenje različitih verzija projekta te suradnju među programerima, te smo ga mi na taj način koristili.

GitHub je web platforma za upravljanje verzijama i suradnju na razvoju softvera. Osnovna funkcionalnost GitHub-a leži u Git-u, koji omogućuje razvojnom timu praćenje promjena u izvornom kodu tijekom vremena. Za izradu svih UML dijagrama korišten je alat [Astah UML](#)⁵.

Astah UML je vrsta alata koji pomaže programerima, analitičarima sustava i drugima da vizualiziraju, modeliraju i dokumentiraju arhitekturu softvera. Podržava

¹<https://discord.com/download>

²<https://www.whatsapp.com/download/>

³<https://git-scm.com/downloads>

⁴<https://github.com/>

⁵<https://astah.net/downloads/>

različite vrste dijagrama, od kojih smo mi koristili dijagrame razreda, stanja, aktivnosti, komponenti, dijagrame obrazaca uporabe te sekvencijske dijagrame.

Odabrana razvojna okruženja pri izradi naše aplikacije bila su [Visual Studio Code](#)⁶ za frontend i [Eclipse](#)⁷ za backend.

Visual Studio Code je razvojno okruženje za pisanje koda razvijen od strane Microsofta. Odlikuje ga bogata podrška za web tehnologije, kao što su HTML, CSS, JavaScript, Typescript itd.

Eclipse je razvojno okruženje koje podržava brojne programske jezike, a neki od njih su: Java, C++, PHP, Python, Ruby...

Kako je već prije navedeno, za razvoj frontenda naše aplikacije koristili smo [JavaScript](#)⁸ i [React](#)⁹.

JavaScript je visokoprogamski jezik koji se često koristi za izgradnju dinamičkih web stranica i web aplikacija, razvijen u Netscape Communications Corporation-u. Najčešće se koristi za programiranje na strani klijenta.

React je JavaScript biblioteka za izgradnju korisničkih sučelja, koju karakterizira komponentna arhitektura koja olakšava organizaciju koda i upravljanje stanjem aplikacije.

Za razvoj backenda naše aplikacije korišten je [Spring Boot](#)¹⁰.

Spring Boot je open-source okvir za izgradnju Java web aplikacija i mikroservisa, dio šireg Spring ekosustava. Posebno je fokusiran na pojednostavljenje konfiguracije, ubrzanje razvoja i olakšavanje izgradnje samostalnih aplikacija.

Baza podataka razvijena je pomoću sustava za upravljanje bazom podataka [PostgreSQL](#)¹¹.

⁶<https://code.visualstudio.com/Download>

⁷<https://www.eclipse.org/downloads/>

⁸<https://www.javascript.com/>

⁹<https://react.dev/>

¹⁰<https://spring.io/projects/spring-boot/>

¹¹<https://www.postgresql.org/download/>

5.2 Ispitivanje programskog rješenja

5.2.1 Ispitivanje komponenti

Test 1: Uspješna registracija

Provedeno je testiranje uspješne registracije novog, dotad neregistriranog korisnika u sustav. Registracija je uspješna ako ne baca iznimku i ako metoda "registerUser" klase UserService vraća istinitu vrijednost. Na slikama ispod prikazani su test uspješne registracije i metoda "registerUser" klase UserService, kao i potvrda uspješne provedbe testa.

```
public boolean registerUser(User user) {
    checkUserDataValid(user);
    userRepository.save(user);
    return false;
}
```

Slika 5.1: Metoda "registerUser" klase UserService

```
@SpringBootTest
class UserRegistrationTest {
    @MockBean
    private UserRepository userRepository;

    @Autowired
    private UserService userService;

    @dunjap2003
    @Test
    public void registrationValidTest(){
        User newUser = new User();
        newUser.setUsername("registeredUser");
        newUser.setPassword("registeredPassword");
        newUser.setName("registeredName");
        newUser.setSurname("registeredSurname");

        when(userRepository.findUserByUsername(newUser.getUsername())).thenReturn(Optional.empty());

        assertDoesNotThrow(() -> userService.registerUser(newUser));
        boolean registrationSuccessful = userService.registerUser(newUser);
        assertTrue(registrationSuccessful);
    }
}
```

Slika 5.2: Test uspješne registracije korisnika



Slika 5.3: Potvrda uspješnog izvođenja testa registracije

Test 2: Uspješna prijava korisnika

Provedeno je testiranje ispravne prijave korisnika koji je registriran i prisutan u bazi. Prijava je uspješna ako metoda "loginUser" klase UserService ne vraća null

vrijednost i ako su svi atributi prethodno stvorenog objekta korisnika i atributi korisnika koji je spremlijen u oponašan repozitorij korisnika jednaki. Na slikama ispod prikazani su test uspješne prijave korisnika i metoda "loginUser" klase UserService.

```
private UserRepository userRepository;
@Autowired
private UserService userService;
@Autowired
private BCryptPasswordEncoder encoder;
15 usages
static User newUser;
± dunjaj2003
@BeforeAll
public static void initUser(){
    newUser = new User();
    newUser.setUsername("newUser");
    newUser.setPassword("newPassword");
    newUser.setName("New");
    newUser.setSurname("User");
}
± Fran +1
@Test
public void loginValidTest(){
    User encryptedUser = new User();
    encryptedUser.setUsername(newUser.getUsername());
    String password = encoder.encode(newUser.getPassword());
    encryptedUser.setPassword(password);
    when(userRepository.findUserByUsername(newUser.getUsername())).thenReturn(Optional.of(encryptedUser));

    User savedUser = userService.loginUser(newUser.getUsername(), newUser.getPassword());

    assertNotNull(savedUser);
    assertTrue(encoder.matches(newUser.getPassword(), savedUser.getPassword()));
}
```

Slika 5.4: Test uspješne prijave korisnika

```
public User loginUser(String username, String password){
    Optional<User> user = userRepository.findUserByUsername(username);
    if(user.isEmpty()){
        return null;
    }

    String cryptedPassword = encoder.encode(password);
    if(!cryptedPassword.equals(user.get().getPassword())){
        return null;
    }

    return user.get();
}
```

Slika 5.5: Metoda "loginUser" klase UserService

Test 3: Neuspješna prijava korisnika (nepostojeće korisničko ime)

Provedeno je testiranje neuspješne prijave korisnika zbog unosa neispravnog korisničkog imena, tj. korisničkog imena koje se ne nalazi u oponašanom repozitoriju korisnika koji je simuliran u sklopu testiranja. Neuspješna prijava korisnika manifestira se kroz dohvaćenu null vrijednost prilikom dohvaćanja traženog korisnika iz oponašanog repozitorija korisnika. Na slici ispod prikazan je test neuspješne prijave korisnika zbog nepostojećeg korisničkog imena. Metoda "loginUser" klase UserService prikazana je u slici iznad.

```

@MockBean
private UserRepository userRepository;

@Autowired
private UserService userService;

14 usages
static User newUser;

▲ dunjap2003
@BeforeAll
public static void initUser(){
    newUser = new User();
    newUser.setUsername("newUser");
    newUser.setPassword("newPassword");
    newUser.setName("New");
    newUser.setSurname("User");
}

new *
@Test
public void loginInvalidUsernameTest(){
    lenient().when(userRepository.findUserByUsername(newUser.getUsername())).thenReturn(Optional.of(newUser));

    User savedUser = userService.loginUser(username: "falseUsername", password: "newPassword");

    assertNull(savedUser);
}

```

Slika 5.6: Test neuspješne prijave korisnika zbog nepostojećeg korisničkog imena

Test 4: Neuspješna prijava korisnika (pogrešna lozinka)

Provedeno je testiranje neuspješne prijave korisnika zbog unosa ispravnog korisničkog imena, ali neispravne lozinke koja bi omogućila pristup profilu s tim korisničkim imenom. Neuspješna prijava korisnika zbog neispravne lozinke manifestira se kroz dohvaćenu null vrijednost prilikom usporedbi lozinki spremljenog korisnika s tim korisničkim imenom i upisane lozinke za to korisničko ime. Na slici ispod prikazan je test neuspješne prijave korisnika zbog upisane neispravne lozinke. Metoda "loginUser" klase UserService prikazana je u slici iznad.

```

@MockBean
private UserRepository userRepository;

@Autowired
private UserService userService;

14 usages
static User newUser;

▲ dunjap2003
@BeforeAll
public static void initUser(){
    newUser = new User();
    newUser.setUsername("newUser");
    newUser.setPassword("newPassword");
    newUser.setName("New");
    newUser.setSurname("User");
}

new *
@Test
public void loginInvalidPasswordTest(){
    lenient().when(userRepository.findUserByUsername(newUser.getUsername())).thenReturn(Optional.of(newUser));

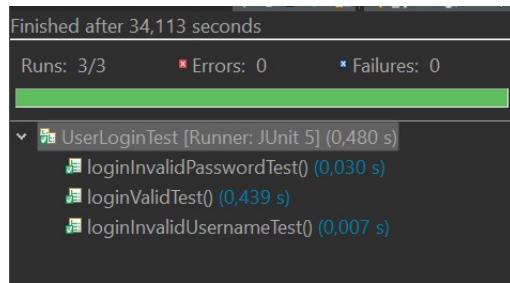
    User savedUser = userService.loginUser(username: "newUsername", password: "falsePassword");

    assertNull(savedUser);
}

```

Slika 5.7: Test neuspješne prijave korisnika zbog unosa neispravne lozinke

Na slici ispod prikazane su potvrde uspješnog izvođenja testova vezanih uz login.



Slika 5.8: Potvrda uspješnog izvođenja testova vezanih uz login

Test 5: Uspješno dohvaćanje podataka o korisniku iz baze podataka

Provedeno je testiranje uspješnog dohvaćanja podataka o korisniku iz baze podataka. Dohvaćanje podataka o korisniku iz baze podataka je uspješno ako metoda "loadUserByUsername" klase UserService vratí objekt korisnika koji je pronađen putem korisničkog imena i ako su atributi korisnika dohvaćenog iz oponašanog reponzitorija korisnika i atributi onog korisnika koji je inicijalno spremljen u oponašani repozitorij jednaki. Na slikama ispod prikazani su test uspješnog dohvaćanja podataka o korisniku iz baze podataka te metoda "loadUserByUsername" klase UserService, kao i potvrda uspješno izvedenog testa o dohvaćanju podataka o korisniku iz baze podataka.

```
public User loadUserByUsername(String username) throws UsernameNotFoundException {
    // TODO Auto-generated method stub
    Optional<User> user = userRepository.findUserByUsername(username);
    if (!user.isPresent()) return null;
    return user.get();
}
```

Slika 5.9: Metoda "loadUserByUsername" klase UserService

```
@MockBean
private UserRepository userRepository;

@Autowired
private UserService userService;

@Author("dunjap2003")
@Test
public void UserRetrieveDataValidTest(){
    User newUser = new User();
    newUser.setUsername("newUser");
    newUser.setPassword("newPassword");
    newUser.setName("New");
    newUser.setSurname("User");

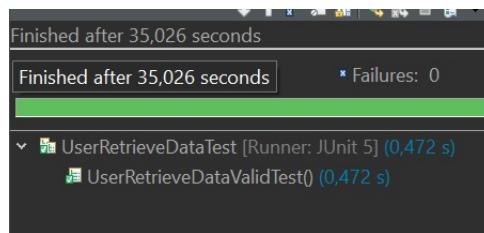
    when(userRepository.findUserByUsername(newUser.getUsername())).thenReturn(Optional.of(newUser));

    User userDetails = userService.loadUserByUsername(newUser.getUsername());

    assertNotNull(userDetails);
    compareUsers(newUser, userDetails);
}

1 usage: @Author("dunjap2003")
private void compareUsers(User newUser, User userDetails) {
    assertEquals(newUser.getUsername(), userDetails.getUsername());
    assertEquals(newUser.getPassword(), userDetails.getPassword());
    assertEquals(newUser.getName(), userDetails.getName());
    assertEquals(newUser.getSurname(), userDetails.getSurname());
}
```

Slika 5.10: Test uspješnog dohvaćanja podataka o korisniku na temelju korisničkog imena iz baze podataka



Slika 5.11: Potvrda uspješnog izvođenja testa dohvaćanja podataka o korisniku

Test 6: Uspješna promjena podataka korisnika

Provedeno je testiranje uspješne promjene podataka korisnika koji se nalazi u bazi. Promjena podataka je uspješna ako metoda "changeInfo" klase UserService uspješno osvježi željene podatke u bazi podataka (podaci o korisniku u oponašanom repozitoriju korisnika odgovaraju podacima koji su zadani kao zamjena originalnima). Na slikama ispod prikazani su test uspješne promjene podataka korisnika te metoda "changeInfo" klase UserService, kao i potvrda uspješnog izvođenja testa o promjeni podataka korisnika.

```

public void changeInfo(User userup){
    Optional<User> user1=userRepository.findById(userup.getId());
    if (!user1.isPresent()) throw new IllegalArgumentException("Cannot update non-existent user.");
    User user = user1.get();
    user.setName(userup.getName());
    user.setPassword(encoder.encode(userup.getPassword()));
    user.setSurname(userup.getSurname());
    user.setUsername(userup.getUsername());

    if (userup instanceof SpecialUser) {
        SpecialUser specialUser = (SpecialUser) user;
        SpecialUser specialUserUp = (SpecialUser) userup;
        specialUser.setPhoto_url(specialUserUp.getImage());
        specialUser.setBiography(specialUserUp.getBiography());
        specialUser.setEmail(specialUserUp.getEmail());
        user = specialUser;
    }
    userRepository.deleteById(user.getId());
    checkUserValid(user);
    userRepository.save(user);
}

```

Slika 5.12: Metoda "changeInfo" klase UserService

```

@.Autowired
private UserService userService;

@MockBean
private UserRepository userRepository;

@MockBean
private PasswordEncoder encoder;

@Test
public void userChangeInfoValidTest(){
    when(encoder.encode(anyString())).thenReturn("encodedUpdatedPassword");

    User originalUser = new User();
    originalUser.setId(1);
    originalUser.setUsername("originalUsername");
    originalUser.setPassword("originalPassword");
    originalUser.setName("originalName");
    originalUser.setSurname("originalSurname");

    User updatedUser = new User();
    updatedUser.setId(1);
    updatedUser.setUsername("updatedUsername");
    updatedUser.setPassword("originalPassword");
    updatedUser.setName("updatedName");
    updatedUser.setSurname("originalSurname");

    when(userRepository.findById(originalUser.getId())).thenReturn(Optional.of(originalUser));
    when(userRepository.save(any())).thenReturn(updatedUser);
}

```

Slika 5.13: Test uspješne promjene podataka korisnika u bazi, 1. dio

```

when(userRepository.save(any())).thenReturn(updatedUser);

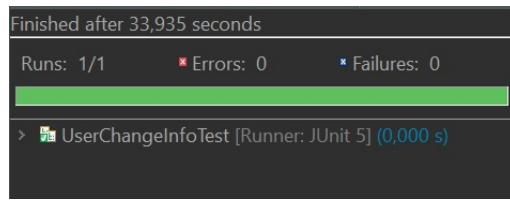
userService.changeInfo(updatedUser);

verify(userRepository, times(wantedNumberOfInvocations: 1)).findById(originalUser.getId());
verify(userRepository, times(wantedNumberOfInvocations: 1)).save(any(User.class));

assertEquals(expected: "updatedUser", originalUser.getUsername());
assertEquals(expected: "encodedUpdatedPassword", originalUser.getPassword());
assertEquals(expected: "Updated", originalUser.getName());
assertEquals(expected: "User", originalUser.getSurname());

```

Slika 5.14: Test uspješne promjene podataka korisnika u bazi, 2. dio



Slika 5.15: Potvrda uspješnog izvođenja testova vezanih uz promjenu podataka korisnika

Test 7: Uspješno dohvaćanje recepata po korisničkom imenu autora recepata

Provedeno je testiranje uspješnog dohvaćanja recepata po korisničkom imenu autora recepata. Dohvaćanje recepata je uspješno ako dohvaćeni set recepata kojima je autor stvoren korisnik nije null vrijednost te ako su očekivani set recepata i dohvaćeni set recepata jednaki. Na slikama ispod prikazani su test uspješnog dohvaćanja recepata po korisničkom imenu autora recepata te metoda "getRecipesByUsername" klase RecipeService.

```

@MockBean
private UserRepository userRepository;

@MockBean
private RecipeRepository recipeRepository;

@Autowired
private RecipeService recipeService;

@Fran +1
@Test
public void loadRecipesUsernameValidTest(){
    Enthusiast newUser = new Enthusiast(username: "newUsername", password: "newPassword", name: "newName", surname: "newSurname", photo: null, bio: null);
    Enthusiast otherUser = new Enthusiast(username: "otherUsername", password: "otherPassword", name: "otherName", surname: "otherSurname", photo: null, bio: null);
    Recipe tortillas = new Recipe(name: "Mexican tortillas", portionSize: 4, cookTime: 60, newUser);
    Recipe chilli = new Recipe(name: "Chilli con carne", portionSize: 2, cookTime: 120, newUser);
    Recipe enchiladas = new Recipe(name: "Chicken enchiladas", portionSize: 4, cookTime: 45, otherUser);
    Set<Recipe> listNew = new HashSet<>();
    listNew.add(tortillas);
    listNew.add(chilli);
    newUser.setRecipes(listNew);

    when(userRepository.findUserByUsername(newUser.getUsername())).thenReturn(Optional.of(newUser));

    Set<Recipe> fetchedSet = recipeService.getRecipesByUsername(newUser.getUsername());

    assertNotNull(fetchedSet);
    assertEquals(listNew, fetchedSet);
}

```

Slika 5.16: Test uspješnog dohvaćanja recepata po korisničkom imenu

```

public Set<Recipe> getRecipesByUsername(String username) {
    Optional<User> u = userRepository.findUserByUsername(username);
    if (u.isEmpty()) return null;
    if (u.get().getRole().getName().equalsIgnoreCase(Role.ENTHUSIAST.getName())) return ((Enthusiast)u.get()).getRecipes();
    return null;
}

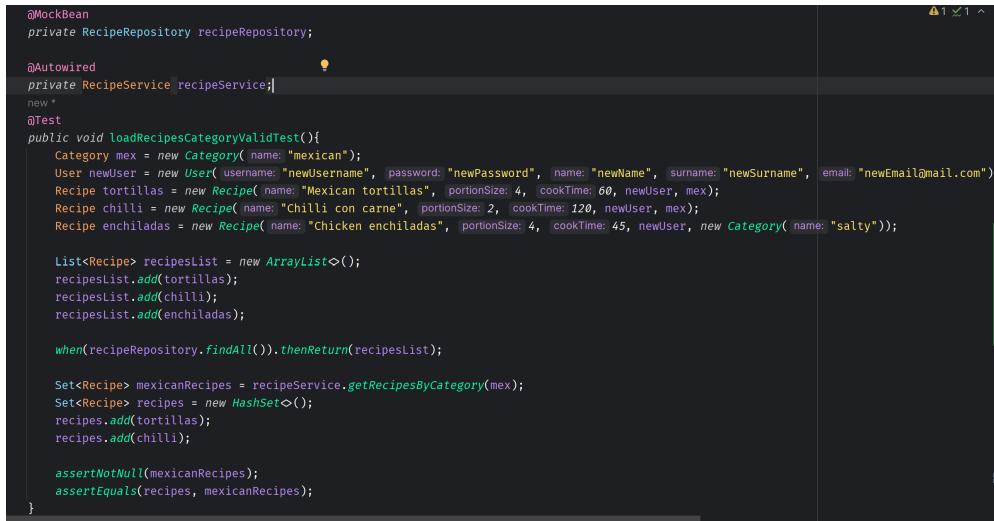
```

Slika 5.17: Metoda "getRecipesByUsername" klase RecipeService

Test 8: Uspješno dohvaćanje recepata po kategoriji recepta

Provedeno je testiranje uspješnog dohvaćanja recepata po kategoriji recepta. Dohvaćanje recepata je uspješno ako dohvaćeni set recepata iz oponašanog repozitorija recepata s određenom kategorijom nije null vrijednost te ako su očekivani set

recepata i dohvaćeni set recepata jednaki. Na slikama ispod prikazani su test uspješnog dohvaćanja recepata po kategoriji te metoda "getRecipesByCategory" klase RecipeService.



```

@MockBean
private RecipeRepository recipeRepository;

@Autowired
private RecipeService recipeService;

@Test
public void loadRecipesCategoryValidTest(){
    Category mex = new Category( name: "mexican");
    User newUser = new User( username: "newUsername", password: "newPassword", name: "newName", surname: "newSurname", email: "newEmail@mail.com");
    Recipe tortillas = new Recipe( name: "Mexican tortillas", portionSize: 4, cookTime: 60, newUser, mex);
    Recipe chilli = new Recipe( name: "Chilli con carne", portionSize: 2, cookTime: 120, newUser, mex);
    Recipe enchiladas = new Recipe( name: "Chicken enchiladas", portionSize: 4, cookTime: 45, newUser, new Category( name: "salty"));

    List<Recipe> recipesList = new ArrayList<>();
    recipesList.add(tortillas);
    recipesList.add(chilli);
    recipesList.add(enchiladas);

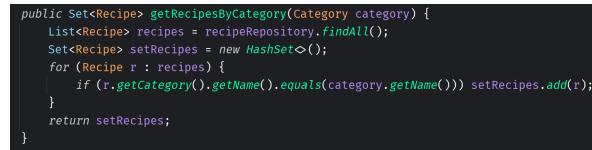
    when(recipeRepository.findAll()).thenReturn(recipesList);

    Set<Recipe> mexicanRecipes = recipeService.getRecipesByCategory(mex);
    Set<Recipe> recipes = new HashSet<>();
    recipes.add(tortillas);
    recipes.add(chilli);

    assertEquals(recipes, mexicanRecipes);
}

```

Slika 5.18: Test uspješnog dohvaćanja recepata po kategoriji



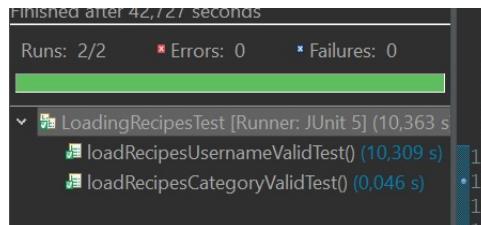
```

public Set<Recipe> getRecipesByCategory(Category category) {
    List<Recipe> recipes = recipeRepository.findAll();
    Set<Recipe> setRecipes = new HashSet<>();
    for (Recipe r : recipes) {
        if (r.getCategory().getName().equals(category.getName())) setRecipes.add(r);
    }
    return setRecipes;
}

```

Slika 5.19: Metoda "getRecipesByCategory" klase RecipeService

Na slici ispod prikazana je potvrda uspješnosti izvođenja testova o dohvaćanju recepata na temelju korisničkog imena/kategorije.



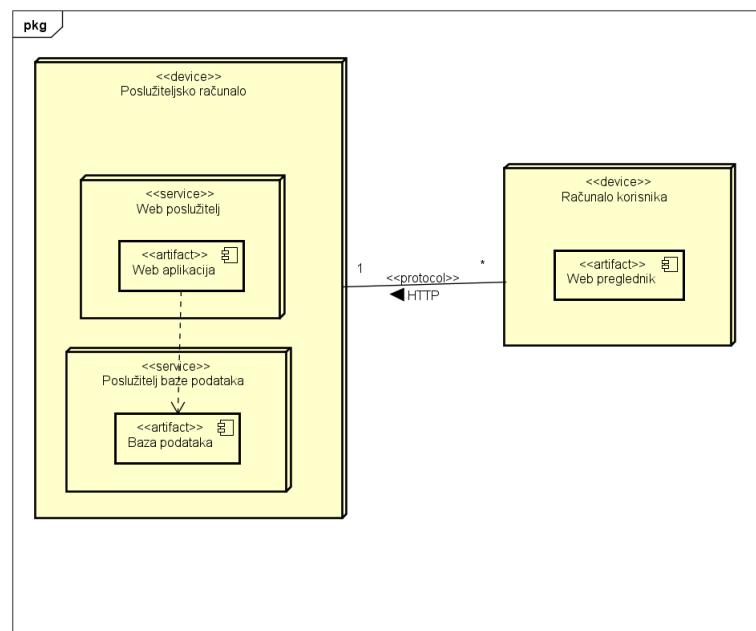
Slika 5.20: Potvrda uspješnog izvođenja testova vezanih uz dohvaćanje recepata

5.3 Dijagram razmještaja

Na slici 5.1 prikazan je dijagram razmještaja. Dijagrami razmještaja prikazuju fizičku arhitekturu programskog sustava, prikazujući razmještaj programskih ar-

tefakata na sklopovskim čvorovima ili virtualnim okruženjima. Glavna svrha dijagrama razmještaja u programskom inženjerstvu je pružiti razumijevanje arhitekture razmještaja sustava.

Naš dijagram razmještaja sastoji se od dvije glavne komponente: računalo korisnika i poslužiteljsko računalo. Na računalu korisnika, on putem web preglednika pristupa aplikaciji KuhajIT. Komunikacija između računala korisnika i poslužiteljskog računala uspostavlja se i odvija pomoću protokola HTTP. Unutar poslužiteljskog računala razlikujemo dvije podkomponente: web poslužitelj i poslužitelj baze podataka.



Slika 5.21: Dijagram razmještaja

5.4 Upute za puštanje u pogon

Naša KuhajIT web-aplikacija, razvijena u okviru ovog projekta, puštena je u pogon putem web platforme [Render](#)¹². Render je platforma koja pruža usluge za deploy, hosting i skaliranje web aplikacija, servisa i web stranica. Istiće se jednostavnosć upotrebe, brzim implementacijama i podrškom za različite jezike i okvire, i upravo smo ju zbog tog razloga i odabrali za tzv. deploy. Na Render-u je postavljen Spring Boot poslužitelj, kojega smo spremili u Docker kontejner, a koji odgovara na upite korisnika, statička stranica, koja služi za prikaz React web aplikacije te PostgreSQL baza podataka. Ispod su priložene dvije slike komponenata postavljenih na Render.

SERVICE NAME	STATUS	TYPE	RUNTIME	REGION	LAST DEPLOYED	
KuhajITBackend	Deployed	Web Service	Docker	Frankfurt	5 minutes ago	***
KuhajIT	Deployed	Web Service	Node	Frankfurt	7 minutes ago	***

Slika 5.22: Pregled prenesenih komponenti, frontend i backend

SERVICE NAME	STATUS	TYPE	RUNTIME	REGION	LAST DEPLOYED	
kuhajIT	Available	PostgreSQL	PostgreSQL 15	Frankfurt	15 hours ago	***

Slika 5.23: Pregled prenesenih komponenti, baza podataka

Postavljanje Spring Boot poslužitelja

Dio razloga zbog kojeg smo odabrali baš Render za deploy je jednostavnost postavljanja komponenti na njega. Tako je bilo i s postavljanjem Spring Boot poslužitelja koji komunicira sa React aplikacijom, spremljenom u [Docker](#)¹³ "kontejner". Korištenje Docker-a nam je uvelike olakšalo cijeli proces deploy-a, jer Docker nudi izolirane okoline, zvane kontejnerima, koje sadrže sve potrebno za pokretanje aplikacije, stoga se nije potrebno oslanjati na ono što je instalirano na Render poslužitelju. Na slici ispod prikazan je kod Dockerfile-a. Dockerfile je dokument koji sadrži sve naredbe koje bi se trebale pozvati u naredbenom retku kako bi se aplikacija ispravno i cjelovito pokrenula.

¹²<https://render.com/>

¹³<https://docker.com/>

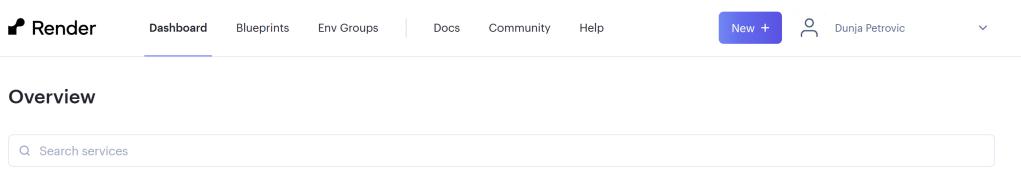
```

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

```

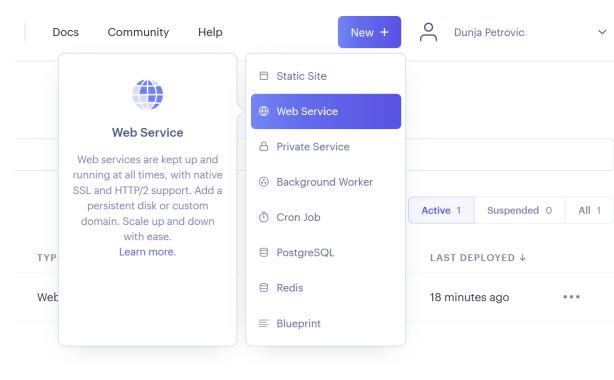
Slika 5.24: Dockerfile

Na početku, potrebno je u gornjem desnom kutu web platforme Render odabratи opciju New, kako je prikazano na slici ispod.



Slika 5.25: Dodavanje nove komponente

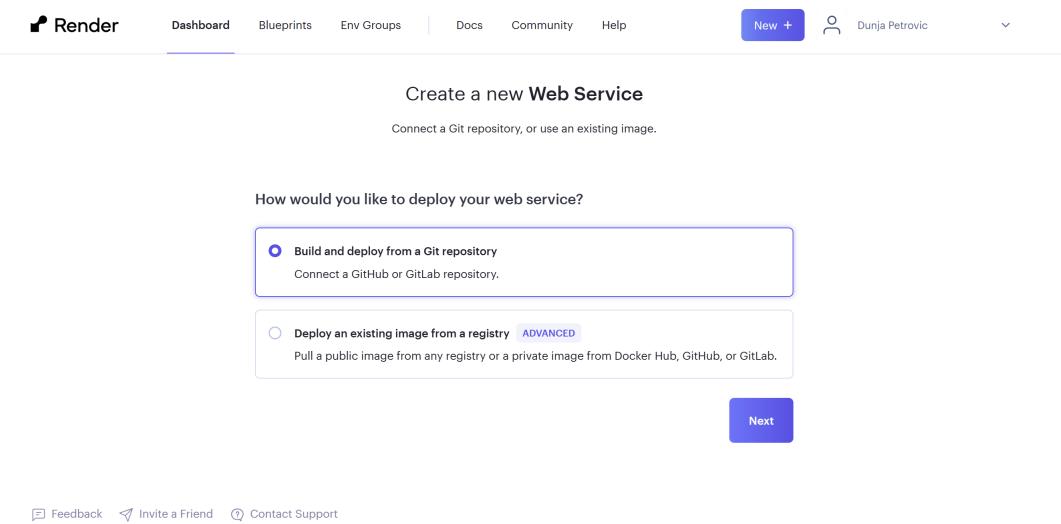
Zatim, u izborniku koji se prikaže, potrebno je odabrati opciju Web Service, kako je prikazano na slici ispod.



Slika 5.26: Odabir opcije Web Service

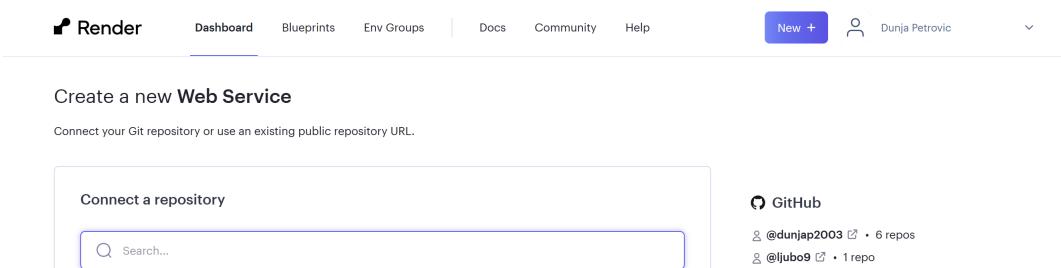
Kada nam se otvorи stranica za dodavanje nove komponente tipa "Web Service", nudi nam se opcija odabira GitHub repozitorija iz koje želimo dodati komponentu,

koju je potrebno i označiti, kako je prikazano na slici ispod.



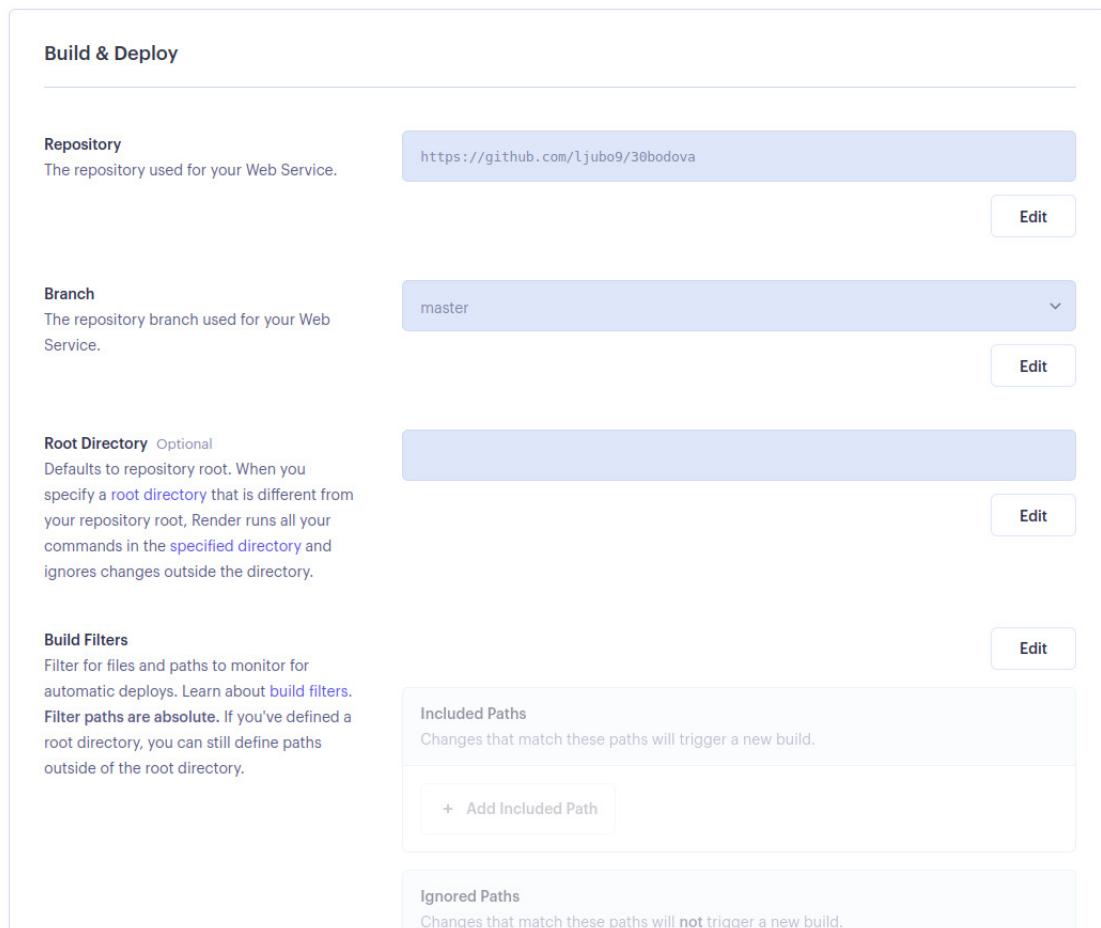
Slika 5.27: Odabir opcije dodavanja iz postojećeg GitHub repozitorija

Nakon povezivanja Render računa s GitHub računom, slijedi odabir repozitorija, kako je prikazano na slici ispod. Mi smo odabrali repozitorij ljubo9/30bodova.



Slika 5.28: Odabir repozitorija

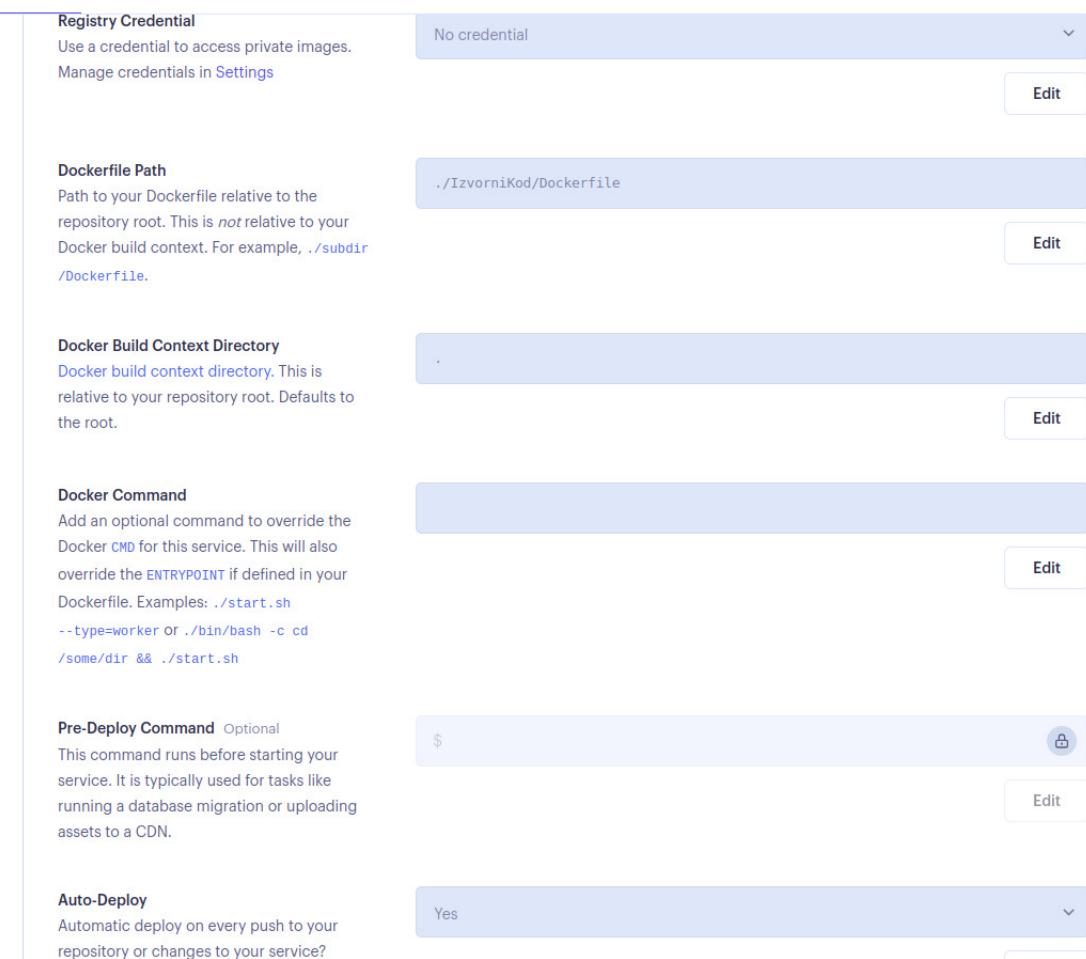
Odabir repozitorija slijedi podešavanje svih potrebnih opcija kako bi se Spring Boot poslužitelj ispravno pokrenuo. Na slici ispod prikazane su postavke koje je potrebno postaviti za ispravan rad poslužitelja.



Slika 5.29: Postavke za postavljanje Spring Boot poslužitelja, 1. dio

Na gornjoj je slici još jednom vidljiv odabrani repozitorij, odabrana grana (mi smo odabrali granu "master") i korijenski direktorij (nama to nije bilo važno pa smo ga ostavili praznog).

Na slici ispod može se vidjeti drugi dio potrebnih postavki za postavljanje Spring Boot poslužitelja.



Slika 5.30: Postavke za postavljanje Spring poslužitelja, 2. dio

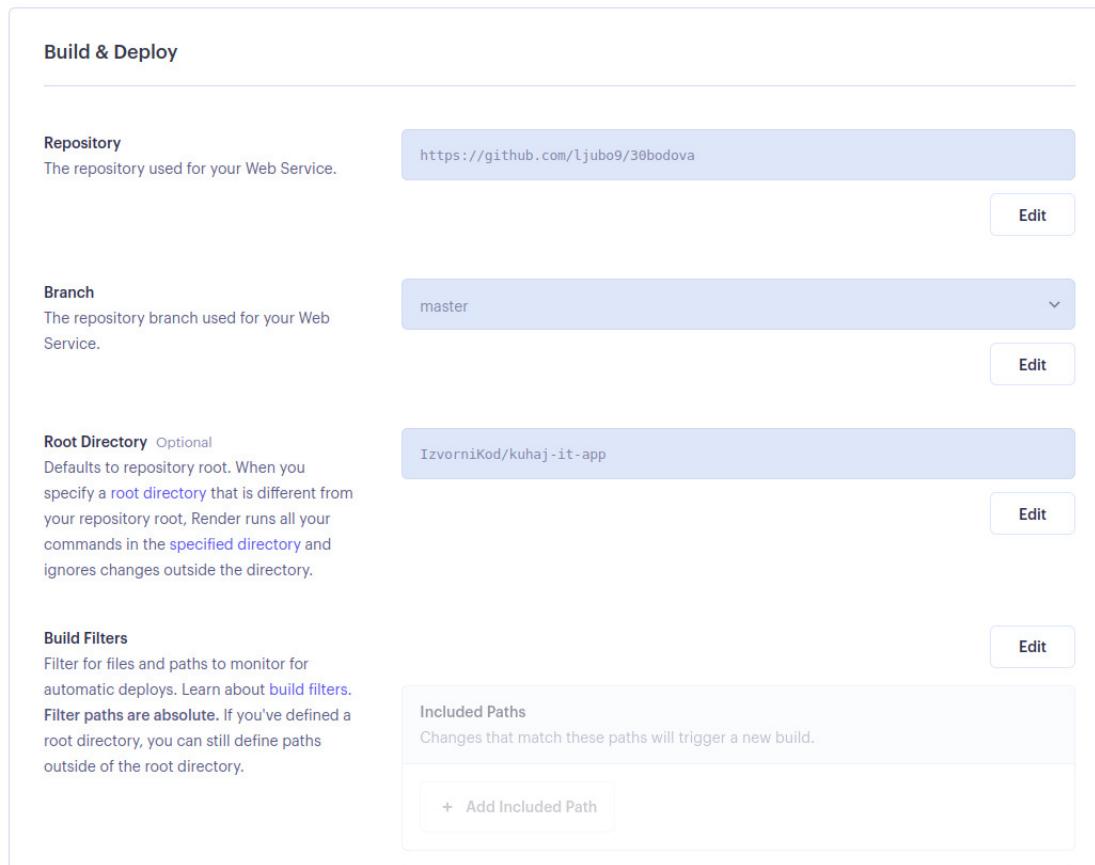
Ono što je bilo važno podešiti je bila lokacija Dockerfile-a, koju smo mi podešili na `./IzvorniKod/Dockerfile`. Usto, označili smo i opciju "Auto-Deploy" s "Yes", tako da se svaki put kada se neka nova izmjena napravi u kodu, Spring Boot poslužitelj se iznova "deploy-a". Time je deploy Spring Boot poslužitelja priveden kraju.

Postavljanje React web aplikacije

Kao i postavljanje Spring Boot poslužitelja, postavljanje React web aplikacije na Render je izrazito jednostavno. Svi koraci do podešavanja postavki za ispravan rad React web aplikacije jednaki su kao za postavljanje Spring Boot poslužitelja: dodavanje novog "web service-a", odabir opcije spajanja s GitHub repozitorijem, spajanje sa željenim GitHub računom te odabir GitHub repozitorija u kojem se nalazi projekt.

Ono što se razlikuje je podešavanje ispravnih postavki za pokretanje React web

aplikacije. Na slici ispod prikazane su potrebne postavke.



Slika 5.31: Postavke za postavljanje React web aplikacije, 1. dio

Odabrani repozitorij isti je kao za postavljanje Spring Boot poslužitelja, kao i odabrana grana repozitorija. Za razliku od postavljanja Spring Boot poslužitelja, ovdje smo za korijenski direktorij odabrali direktorij IzvorniKod/kuhaj-it-app. Drugi dio potrebnih postavki prikazan je na slici ispod.

The screenshot shows the GitHub Actions settings for a repository named 'IzvorniKod/kuhaj-it-app'. It includes sections for Build Command, Pre-Deploy Command, Start Command, Auto-Deploy, and Deploy Hook.

- Build Command:** \$ npm install. This command runs in the root directory of the repository when a new version of your code is pushed, or when you deploy manually. It is typically a script that installs libraries, runs migrations, or compiles resources needed by your app.
- Pre-Deploy Command (Optional):** This command runs before starting your service. It is typically used for tasks like running a database migration or uploading assets to a CDN.
- Start Command:** \$ npm start. This command runs in the root directory of your app and is responsible for starting its processes. It is typically used to start a webserver for your app. It can access environment variables defined by you in Render.
- Auto-Deploy:** Yes. Automatic deploy on every push to your repository or changes to your service? Select "No" to handle your deploys manually. Automatic deploys not working as expected? Troubleshoot your GitHub connection.
- Deploy Hook:** https://api.render.com/deploy/srv-cl8eb72uuipc73epuf10?key=VUm_0SnT1gw. Your private URL to trigger a deploy for this server. Remember to keep this a secret. Regenerate Hook button.

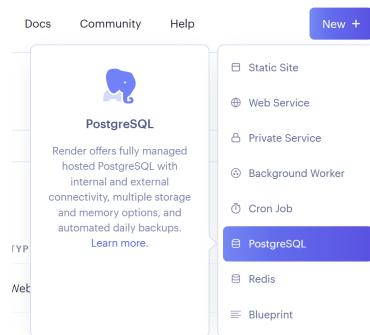
Slika 5.32: Postavke za postavljanje React web aplikacije, 2. dio

Na gornjoj je slici vidljivo da smo za "naredbu izgradnje", naredbu koja će instalirati sve biblioteke, pokrenuti migracije i kompajlirati sve potrebno za ispravan rad naše React web aplikacije, odabrali naredbu "npm install". Ono što je još trebalo podesiti je naredba pokretanja, koju smo podesili na "npm start". Isto kao i kod podešavanja Spring Boot poslužitelja, odabrali smo opciju Auto-Deploy.

Postavljanje baze podataka

Iako nešto drugačije od "deploy-anja" Spring Boot poslužitelja i React web aplikacije, postavljanje PostgreSQL baze podataka na platformu Render također je izrazito jednostavno.

Za početak, potrebno je na početnoj stranici stisnuti gumb New, a zatim odabrati opciju PostgreSQL, kako je prikazano na slici ispod.

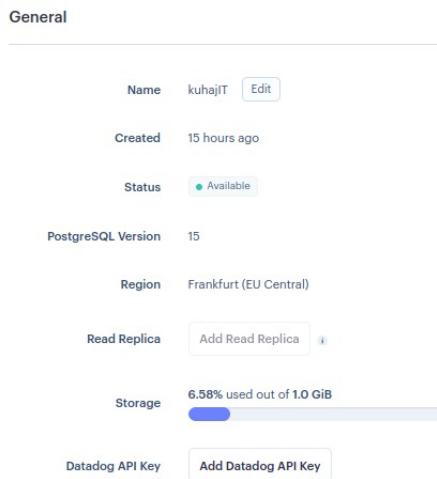


Slika 5.33: Dodavanje nove PostgreSQL baze podataka

Odabirom opcije PostgreSQL, otvara se novi prozor na kojem je moguće unijeti razne podatke o bazi podataka: ime, korisnika, verziju PostgreSQL-a itd. Podatke koje smo mi unijeli prikazani su na slikama ispod.

A screenshot of the 'Connections' configuration page for a PostgreSQL service. The page title is 'Connections'. There are several input fields: 'Hostname' set to 'dpg-cmksasuct0pc73d3q1d0-a', 'Port' set to '5432', 'Database' set to 'kuhajit_76wy', 'Username' set to 'ivan', and 'Password' (redacted). Below these are fields for 'Internal Database URL', 'External Database URL', and 'PSQL Command', each with a redacted value.

Slika 5.34: Postavke baze podataka na Render-u, 1. dio



Slika 5.35: Postavke baze podataka na Render-u, 2. dio

Postavka "Password" šifra je usera koja služi za spajanje na bazu i vidljiva je u datoteci application.properties, dok su polja Internal Database URL, External Database URL i PSQL Command generirana od strane platforme Render.

Stvorena baza podataka je prazna, a mi ju moramo napuniti podacima. Skripta za izravno punjenje baze podataka (inicijaliziranje svih tablica i veza između njih te punjenje nekim testnim podacima) nalazi se u datoteci Script.sql, a mi smo ju pokretali izravno iz razvojnog okruženja Eclipse IDE.

6. Zaključak i budući rad

17 tjedana nakon prvog okupljanja projektne grupe, naš je zadatak priveden kraju. Implementirali smo web aplikaciju za sve one koji su zainteresirani za kvalitetnu prehranu. Ostvarena je registracija i prijava korisnika u tri kategorije: "običnih" klijenata, kulinarskih entuzijasta i nutricionista. "Obični" klijent na našoj KuhajIT aplikaciji ima, uz mogućnost pretraživanja kulinarskih entuzijasta i pregleda njihovih recepata, što može i neregistrirani korisnik, mogućnost skeniranja ili ručnog unosa sastojaka koje ima doma i koje želi upotrijebiti za pripravu recepta, na temelju kojih mu aplikacija generira najprikladnije recepte. Također, klijent ima mogućnost i označiti konzumirane recepte, zapratiti željene kulinarske entuzijaste te putem aplikacije pratiti dijetu koju kreira nutricionist. Kulinarski entuzijast ima sve mogućnosti "običnog" klijenta, no tu je i mogućnost stvaranja kuharica i recepata, koje objavljuje na aplikaciji. Nutricionist također ima sve mogućnosti "običnog" klijenta, uz opciju unošenja nutritivnih podataka o sastojcima te kreiranje dijeta sa željenim ograničenjima. Izrada čitave ove funkcionalnosti bila je podijeljena u dva dijela: tzv. "prvi" i "drugi" ciklus. Prvi ciklus nije bio toliko zahtjevan što se same implementacije tiče, no ima ogromnu težinu zbog važnosti svih odluka donesenih upravo tijekom njega, od okupljanja projektne grupe, preko konceptualnog osmišljavanja same implementacije, pa sve do postavljanja čvrstih temelja na koje se naprednija implementacija nadograđivala. Značajan dio rada u prvom ciklusu bilo je opširno dokumentiranje čitavog projektnog zadatka: opis projektnog zadatka, specifikacija programske potpore (koja je uključivala obrasce uporabe te pripadajuće dijagrame obrazaca uporabe te sekvencijske dijagrame) i opis arhitekture (dizajn baze podataka te dijagrami razreda). Drugi ciklus bio je prilično implementacijski zahtjevniji. Većina konkretne implementacije web aplikacije ostvarena je upravo u drugom ciklusu, uz manje, ali ne manje važne dopune dokumentaciji (dijagrami stanja, aktivnosti, komponenti, razmještaja itd.). U ovom se ciklusu većina članova tima susrela s prvim ozbiljnijim radom u njima nepoznatim tehnologijama, što je iziskivalo određeno vrijeme upoznavanja, istraživanja i prilagodbe novome. Ne treba zanemariti niti vrijeme uloženo u pažljivo testiranje rada aplikacije pomoću unit i selenium testova, a koje

je opisano u jednom od gornjih poglavlja, kao niti tzv. "deployment" naše web aplikacije. Tijekom implementacije, dogovori između članova tima tekli su putem WhatsApp grupe, što nam se pokazalo vrlo korisnim jer su na taj način svi članovi tima, čak i oni koji u dotičnom dijelu implementacije nisu direktno sudjelovali, bili upućeni u trenutno stanje. Prostora za napredak uvijek ima, neke od ideja koje smo imali su stvaranje personaliziranih dijeta na zahtjev klijenata, stvaranje mobilne verzije aplikacije KuhajIT, unapređenje sučelja i dr. Uz sve gore navedeno, upoznavanje s novim tehnologijama, učenje pravilnog dokumentiranja, "deployanja" i slično, najvažnija tekovina ovog projekta svim je članovima upravo rad u timu. Iako nije u svim trenucima bilo jednostavno, svi smo naučili vrijedne lekcije o timskom radu, međusobnoj potpori, organizaciji vremena i vrijednostima koje ćemo tražiti i u nekim budućim projektnim grupama. S obzirom na to da nam je svima to prvo iskustvo ovakvoga tipa, a i na ostale obaveze koje svi imamo na fakultetu, ponosni smo na obavljenou.

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>

Dodatak: Prikaz aktivnosti grupe

Dnevnik sastajanja

1. sastanak

- Datum: u ovom formatu: 16.listopada 2023.
- Prisustvovali: H.Ivančić, D.Ljubić, I.Pavlović, M.Perković, D.Petrović, K.Raštegorac, F.Širić
- Teme sastanka:
 - upoznavanje s temom

2. sastanak

- Datum: u ovom formatu: 25.listopada 2023.
- Prisustvovali: H.Ivančić, I.Pavlović, M.Perković, D.Petrović, K.Raštegorac, F.Širić
- Teme sastanka:
 - raspodjela posla
 - početak rada

3. sastanak

- Datum: 2.studenoga 2023.
- Prisustvovali: H.Ivančić, D.Ljubić, I.Pavlović, M.Perković, D.Petrović, K.Raštegorac, F.Širić
- Teme sastanka:
 - provjera dotad završene dokumentacije
 - detaljnija raspodjela posla

4. sastanak

- Datum: 9.studenoga 2023.
- Prisustvovali: H.Ivančić, D.Ljubić, I.Pavlović, M.Perković, D.Petrović, K.Raštegorac, F.Širić
- Teme sastanka:
 - testiranje dotad dovršene implementacije
 - provjera dotad završene dokumentacije

5. sastanak

- Datum: 17.studenoga 2023.
- Prisustvovali: H.Ivančić, D.Ljubić, I.Pavlović, M.Perković, D.Petrović, K.Raštegorac, F.Širić
- Teme sastanka:
 - posljednja provjera ispravnosti dokumentacije
 - posljednja provjera implementacije funkcionalnosti

6. sastanak

- Datum: 11.prosinca 2023.
- Prisustvovali: H.Ivančić, D.Ljubić, I.Pavlović, M.Perković, D.Petrović, K.Raštegorac, F.Širić
- Teme sastanka:
 - dogovor o dalnjem tijeku rada
 - zadavanje ciljeva implementacije za 2. ciklus

7. sastanak

- Datum: 18.prosinca 2023.
- Prisustvovali: H.Ivančić, D.Ljubić, I.Pavlović, M.Perković, D.Petrović, K.Raštegorac, F.Širić
- Teme sastanka:
 - statusni sastanak
 - plan daljnog rada

8. sastanak

- Datum: 29.prosinca 2023.
- Prisustvovali: H.Ivančić, D.Ljubić, M.Perković, K.Raštegorac, F.Širić
- Teme sastanka:
 - dogovor o dijelovima frontend aplikacije

9. sastanak

- Datum: 5.siječnja 2024.
- Prisustvovali: H.Ivančić, D.Ljubić, I.Pavlović, M.Perković, D.Petrović, K.Raštegorac, F.Širić
- Teme sastanka:
 - statusni sastanak
 - konzultacije o izradi preostalog dijela dokumentacije

10. sastanak

- Datum: 12.siječnja 2024.
- Prisustvovali: H.Ivančić, D.Ljubić, I.Pavlović, M.Perković, D.Petrović, K.Raštegorac, F.Širić
- Teme sastanka:
 - briefing dotad implementiranog
 - posljednje izmjene baze podataka

11. sastanak

- Datum: 13.siječnja 2024.
- Prisustvovali: H.Ivančić, D.Ljubić, I.Pavlović, M.Perković, D.Petrović, K.Raštegorac, F.Širić
- Teme sastanka:
 - briefing dotad implementiranog
 - posljednje izmjene baze podataka

12. sastanak

- Datum: 16.siječnja 2024.
- Prisustvovali: H.Ivančić, D.Ljubić, I.Pavlović, M.Perković, D.Petrović, K.Raštegorac, F.Širić
- Teme sastanka:
 - briefing dotad implementiranog

13. sastanak

- Datum: 19.siječnja 2024.
- Prisustvovali: H.Ivančić, D.Ljubić, I.Pavlović, M.Perković, D.Petrović, K.Raštegorac, F.Širić
- Teme sastanka:
 - posljednje izmjene dokumentacije
 - posljednja provjera inmplementacije

Tablica aktivnosti

Kontinuirano osvježavanje

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

	Hana Ivančić	David Ljubić	Ivan Pavlović	Marija Perković	Dunja Petrović	Karlo Raštegorac	Fran Širić
Upravljanje projektom	10	1			2		
Opis projektnog zadatka					4		
Funkcionalni zahtjevi	3				7		
Opis pojedinih obrazaca	1				5		
Dijagram obrazaca	6				2		
Sekvencijski dijagrami	5				2		
Opis ostalih zahtjeva					1		
Arhitektura i dizajn sustava					8		
Baza podataka					5		
Dijagram razreda	6				2		
Dijagram stanja							
Dijagram aktivnosti							
Dijagram komponenti							
Korištene tehnologije i alati	2	2	2	2	2	2	5
Ispitivanje programskog rješenja		2		5		2	10
Dijagram razmještaja							
Upute za puštanje u pogon							

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

	Hana Ivančić	David Ljubić	Ivan Pavlović	Marija Perković	Dunja Petrović	Karlo Raštegorac	Fran Širić
Dnevnik sastajanja					1		
Zaključak i budući rad							
Popis literature					1		
<i>Dodatne stavke kako ste podijelili izradu aplikacije</i>							
<i>npr. izrada početne stranice</i>		10		12		10	
<i>izrada baze podataka</i>			16				5
<i>spajanje s bazom podataka</i>			2				2
<i>back end</i>							30
<i>deploy</i>	10	1	1	1	1	1	1