

Reci-pa-ispeci

Nastavnik: Vlado Sruk

Cilj ovoga projekta je razviti web aplikaciju „Reci pa ispeci“ koja će korisnicima olakšati svakodnevnu pripremu jela, te uz to omogućiti interakciju s ostalim korisnicima koji dijele strast prema dijeljenju kulinarskih ideja, pripremi hrane i (naravno) konzumaciji iste. Na ovaj način se potiče stvaranje zajednice ljudi koja dijeli strast prema kulinarstvu.

Današnji ubrzani način života donosi brojne izazove, kada je u pitanju svakodnevna priprema jela. Često se suočavamo s pitanjem: „Što učiniti s dostupnim namirnicama?“ ili jednostavno nemamo inspiraciju za sljedeći obrok. Ove situacije nas nerijetko demotiviraju, dovodeći do ponavljanja istih recepata ili nepotrebnog bacanja hrane.

Jedno od trenutačno korištenih rješenja su bilježnice u koje se zapisuju recepti. Standardne bilježnice s receptima su vrlo često nepraktične iz nekoliko razloga. Neki od razloga su:

- bilježnice nemaju mogućnost brzog i učinkovitog pretraživanja recepata
- nakon nekog vremena bilježnica postaje nepregledna, jer način zapisivanja često nije sistematiziran
- recepti zapisani u bilježnici nisu lako dostupni drugima

Iz ovih par razloga se jasno vidi zašto ovaj način, u današnjem dobu, nije održiv. Uz pomoć aplikacije "Reci pa ispeci" moguće je lako i brzo pretraživanje recepata (način na koji su recepti zapisani je sistematiziran i lako čitljiv) te postoji mogućnost objaviti javno recept, kako bi drugima bio također dostupan.

Drugi primjer rješenja je web aplikacija: „<https://www.recepti.com/>“. U navedenoj aplikaciji korisnik može napisati sistematizirani recept, koji se šalje na odobravanje. Uz tekstualne podatke moguće je priložiti i sliku jela unutar recepta.

Dodajte recept

Naziv:

Kategorija:

Težina pripreme:

Vreme pripreme: ☐ MIN ☐ Porcije ☐ Posao

Kuhinja:

Glavna slika: **Dodaj**

Grupa sastojaka

Naziv:

Količina: Sastojak: **+**

Nova grupa

Pripremni postupak

Korak 1: **+**

Slika: **Dodaj**

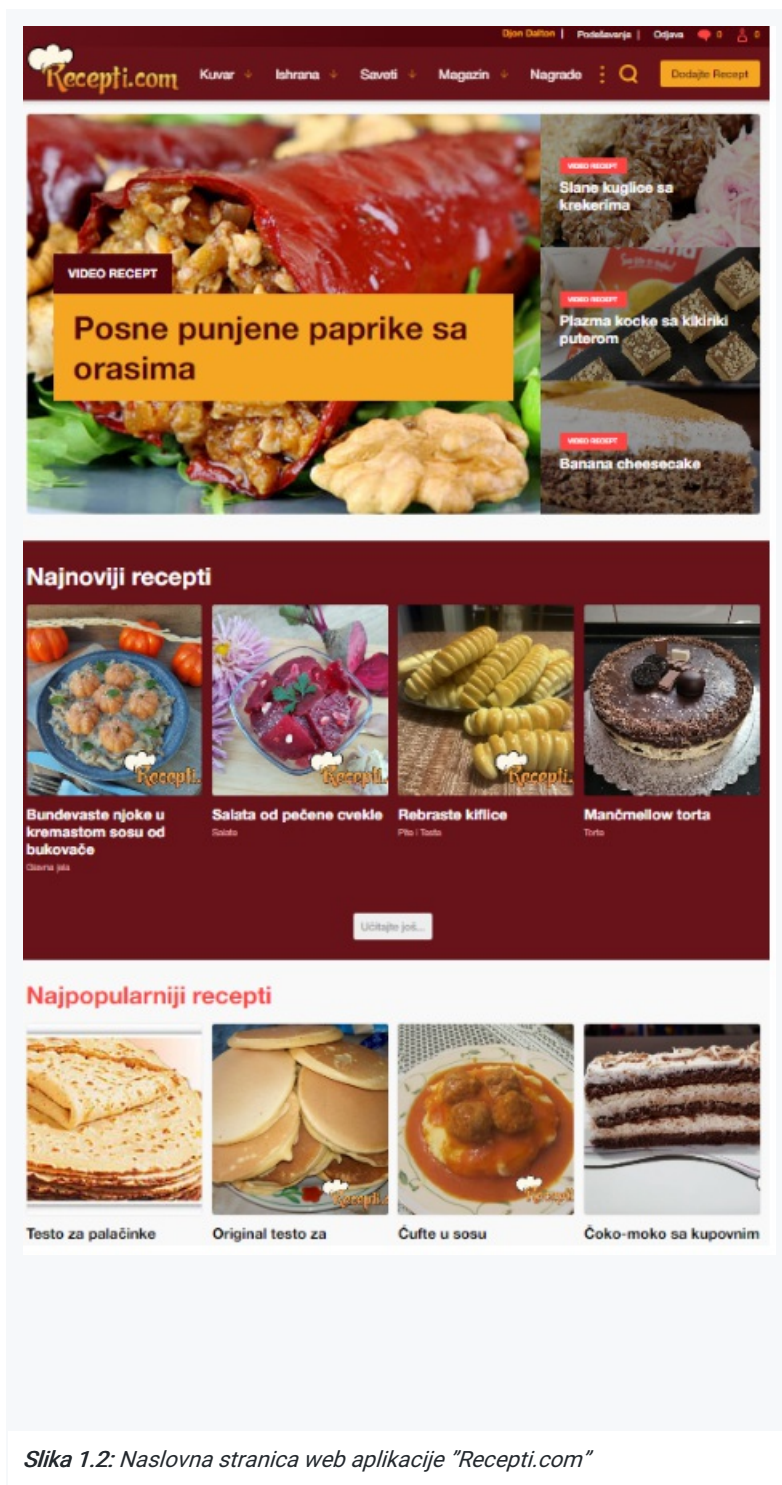
Savet

☐ Priznatim uslove sajta

Postavite recept

Slika 1.1: Stranica za prijedlog recepta web aplikacije "Recepti.com"

Na aplikaciji je moguće vidjeti tuđe recepte i filtrirati izbor recepata koji se prikazuje, također je moguće stvoriti vlastitu listu recepata.



Slika 1.2: Naslovna stranica web aplikacije "Recepti.com"

Iako je ovo rješenje naprednije od bilježnica, ima nekoliko nedostataka. Primjerice, ne podržava AI chat koji bi odgovarao na kulinarske upite, te ne postoji mogućnost stvaranja recepata dostupnih samo korisniku. Ovaj projekt sadrži funkcionalnosti koje ispravljaju navedene poteškoće u već spomenutim rješenjima, kao što su implementacija AI chata, brzo i jednostavno pretraživanje vlastitih recepata te mogućnost komunikacije između korisnika.

Ciljani klijenti su sve od kuhara početnika do profesionalaca, pa čak i onih koji se samo žele okušati u izvedbi nekoga recepta.

U aplikaciji postoje 3 uloge:

- Korisnici
- Kuhari
- Administratori

Korisnici imaju pristup svojoj privatnoj listi recepata, s mogućnošću dodavanja novih recepata prema vlastitim željama. Također, mogu pregledavati i pretraživati javnu listu recepata te po potrebi predložiti promjene ili dodati nove recepte.

Kuhari su korisnici s pravom odobravanja prijedloga recepata. To su osobe koje su prepoznate kao relevantne u ovom području, te imaju pravo na kontrolu javne liste recepata. Oni slobodno mogu dodati novi recept u javnu listu ili izbrisati neki već postojeći.

Administratori su osobe koje imaju mogućnost izmjenjivati detalje recepata, brisati određene recepte iz liste i po potrebi dodavati nove recepte. Pod njihovom kontrolom su svi korisnički računi i svi dijelovi aplikacije. Mogu nekog korisnika promovirati u kuhara ili ga (po potrebi) degradirati natrag u običnog korisnika.

Svi korisnici imaju mogućnost pisanja u dva različita chata. Jedan u kojem mogu komunicirati s ostalim korisnicima kuharice i drugi u kojem je moguće predložiti recept koji će biti prepoznat od strane AI-a, te će se automatski kreirati novi prijedlog recepta. AI chat se dodatno može koristiti i za ostale upite vezane uz kulinarstvo. Također, svi imaju mogućnost dodavanja dodatnih informacija o svojem profilu (npr. dodavanje slike profila i dodavanje opisa o sebi) te mogu pregledavati profile drugih korisnika. Na samome profilu piše koja je uloga korisnika (jedna od tri već spomenute uloge).

Neprijavljeni korisnici imaju pristup samo javnome popisu recepata te se mogu registrirati ili prijaviti u aplikaciju.

Korisnici mogu pretraživati recepte prema nazivu jela, dostupnim sastojcima i vremenu pripreme. Na stranici svakog recepta mogu ostaviti ocjene i komentare. Korisnici s učestalo dobrim ocjenama mogu napredovati do statusa kuhara, dok se kuhari s velikim brojem loših ocjena mogu vratiti na osnovnu korisničku ulogu. Moguće je označiti pojedine sastojke kao omiljene, te postoji mogućnost pregledavanja personalizirane liste pod nazivom „Preporučeni recepti“ (koja je prilagođena svakom korisniku ovisno o najdražim sastojcima).

Aplikacija „Reci pa ispeci“ predstavlja suvremeno rješenje za organizaciju recepata, poticanje kulinarske kreativnosti i olakšavanje svakodnevnih izazova u kuhinji. Cilj je stvoriti zajednicu korisnika koja dijeli strast prema kuhanju, čime se unaprjeđuje kulinarsko iskustvo za sve sudionike.

Moguće dodatne funkcionalnosti za aplikaciju, koje se mogu nadodati nakon izvršavanja jezgrenih funkcionalnosti, su:

- Omogućavanje korisnicima da biraju između hrvatskog i engleskog jezika za korisničko sučelje, čime se povećava pristupačnost aplikacije za širi krug korisnika.
- Korisnicima omogućiti izvoz recepata u PDF formatu ili njihovo dijeljenje putem e-maila.
- Dodavanje mogućnosti prilagodbe korisničkog sučelja, npr. tamni način rada, raspored elemenata.

Funkcionalni zahtjevi

ID zahtjeva	Opis	Prioritet	Izvor	Kriteriji prihvaćanja
F-001	Sustav omogućuje korisnicima pregledavanje javne liste recepata na stranici.	Visok	Dokument zahtjeva	Korisnik može pregledavati javne recepte prema različitim kriterijima pretraživanja.
F-002	Sustav omogućuje prikaz javnog profila drugog korisnika.	Srednji	Dokument zahtjeva	Korisnik može vidjeti javni profil drugog korisnika.
F-003	Sustav omogućuje korisnicima registraciju i prijavu putem OAuth 2.0 standarda.	Visok	Dionik zahtjeva	Korisnik može uspješno autorizirati prijavu putem Googlea.
F-004	Sustav omogućuje pregled korisničkih podataka na stranici profila.	Visok	Dokument zahtjeva	Korisnik može pregledati svoje osobne podatke na stranici „Moj profil“.
F-005	Sustav omogućuje korisnicima ažuriranje njegovih podataka na profilu.	Srednji	Dokument zahtjeva	Korisnik može uspješno promijeniti i spremiti nove osobne podatke na profilu.
F-006	Sustav omogućuje korisnicima slanje prijedloga recepata.	Visok	Dokument zahtjeva	Korisnik može predložiti novi recept, koji se šalje na odobravanje.
F-007	Sustav omogućuje komunikaciju među korisnicima putem chata.	Srednji	Dokument zahtjeva	Korisnik može poslati i primiti poruke u chatu s drugim korisnicima.
F-008	Sustav omogućuje korisnicima upotrebu AI chata.	Srednji	Dokument zahtjeva	Korisnik može koristiti specijalizirani AI chat za predlaganje recepata i kulinarske savjete.
F-009	Sustav omogućuje spremanje recepata na privatnu listu.	Visok	Dokument zahtjeva	Korisnik može dodati recepte na privatnu listu.
F-010	Sustav omogućuje pregled vlastite liste recepata.	Visok	Dokument zahtjeva	Korisnik može pregledati sve recepte koje je označio kao favorite.
F-011	Sustav omogućuje ocjenjivanje recepata s komentarom.	Srednji	Dokument zahtjeva	Korisnik može ocijeniti recept i ostaviti popratni komentar.
F-012	Sustav omogućuje dodavanje sastojaka na listu najdražih sastojaka.	Srednji	Dokument zahtjeva	Korisnik može dodati sastojke na listu favorita.
F-013	Sustav omogućuje pregled recepata na temelju liste najdražih sastojaka.	Srednji	Dokument zahtjeva	Korisnik može pregledavati personaliziranu listu recepata temeljenih na omiljenim sastojcima.
F-014	Sustav omogućuje brisanje sastojaka iz liste omiljenih sastojaka.	Srednji	Dokument zahtjeva	Korisnik može ukloniti sastojke iz liste omiljenih sastojaka.
F-015	Sustav omogućuje prihvaćanje ili odbijanje recepata korisnika.	Visok	Dokument zahtjeva	Kuhar (ili administrator) može odlučiti hoće li prihvatiti ili odbiti recept.
F-016	Sustav omogućuje promjenu detalja recepta.	Srednji	Dokument zahtjeva	Korisnik može izmijeniti detalje recepta.
F-017	Sustav omogućuje dodavanje novih sastojaka u bazu podataka.	Srednji	Zahtjev dionika	Kuhar (ili administrator) može dodati nove sastojke u bazu podataka.

F-018	Sustav omogućuje administratoru promociju korisnika u kuhara.	Visok Prioritet	Dokument zahtjeva	Administrator može promovirati korisnika u kuhara i prihvatanja
F-019	Sustav omogućuje administratoru degradaciju kuhara u korisnika.	Visok	Dokument zahtjeva	Administrator može degradirati kuhara u običnog korisnika.
F-020	Sustav omogućuje korisniku brisanje recepta	Visok	Dokument zahtjeva	Korisnik može obrisati recept iz liste

Ostali zahtjevi

ID zahtjeva	Opis	Prioritet
NF-1.1	Sustav treba koristiti OAuth 2.0 standard za autentikaciju i autorizaciju korisnika.	Visok
NF-2.1	Sustav mora podržati rad do 100 korisnika bez smanjenja performansi.	Visok
NF-3.1	Sustav mora biti dostupna većinu vremena tijekom godine (osim za planirano održavanje).	Visok
NF-4.1	Sustav treba imati jasnu dokumentaciju.	Visok
NF-4.2	Sustav treba biti oblikovan tako da je lako održiv.	Visok
NF-5.1	Sustav mora ograničiti korisnika na pristup jedino onim resursima kojima ima pristup.	Visok
NF-5.2	Sustav treba biti oblikovan tako da omogućuje jednostavno korištenje	Visok

Dionici

1. Korisnici sustava (D-1)
- Neregistrirani korisnik
 - Korisnik
 - Kuhar
2. Administratori sustava (D-2)
3. Razvojni tim (D-3)
4. Naručitelji sustava (D-4)

Aktori i njihove funkcionalnosti

Korisnici sustava (A-1)

- Funkcije: F-001, F-002, F-003, F-004, F-005, F-006, F-007, F-008, F-009, F-010, F-011, F-012, F-013, F-014, F-016, F-020

Kuhari (A-2)

- Funkcije: F-001, F-002, F-003, F-004, F-005, F-006, F-007, F-008, F-009, F-010, F-011, F-012, F-013, F-014, F-015, F-017, F-020

Administratori (A-3)

- Funkcije: F-001, F-002, F-003, F-004, F-005, F-006, F-007, F-008, F-009, F-010, F-011, F-012, F-013, F-014, F-015, F-017, F-018, F-019, F-020

Neprijavljeni korisnici (A-4)

- Funkcije: F-001, F-002, F-003

Obrasci uporabe

Svaka web aplikacija ima određeni set funkcija koje korisnici, administratori i ostali sudionici mogu obavljati. U slučaju ove web aplikacije korisnici se dijele na goste, prijavljene korisnike, kuhare i administratore.

Svaki tip korisnika ima određene funkcije koje može obavljati, pri čemu su neke funkcije dostupne svima, a neke su specifične za jedan ili više tipova.

Opis obrazaca uporabe nalazi se u nastavku.

Use Cases: Opis obrazaca uporabe

UC1: Pregledavanje recepata na stranici

- **Glavni sudionik:** Korisnik/Gost
- **Cilj:** pregledavanje i pretraga recepata na početnoj stranici
- **Sudionici:** baza podataka, web aplikacija
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
 1. Učitavanje početne stranice web-aplikacije
 2. Korisnik pretražuje recepte po sastojcima/vremenu pripreme ili imenu.
 3. Korisniku se prikazuje lista recepata sa zadanim parametrima

UC2: Prikaz javnog profila drugog korisnika

- **Glavni sudionik:** Korisnik/Gost
- **Cilj:** Pregledati profil drugog korisnika
- **Sudionici:** Web aplikacija, baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
 1. Korisnik otvara profil drugog korisnika preko stranice recepta. (F-011)
 2. Prikazuju se informacije, ocjene, uloga i ostali podaci sa stranice profila

UC3: Prijava u sustav

- **Glavni sudionik:** Gost
- **Cilj:** Pristup korisničkom sučelju
- **Sudionici:** Baza podataka, Google servis za autorizaciju
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
 1. Ulazak na stranicu za prijavu
 2. Prijava putem Google-a (ako se radi o prvoj prijavi podaci se pohranjuju u bazu podataka) - OAuth 2.0 Login
 3. Pristupa se korisničkom sučelju i funkcijama
- **Opis mogućih odstupanja:**
 1. Ako je pri autorizaciji preko Google-a došlo do pogreške korisnik može pokušati ponovno

UC4: Pregled korisničkih podataka

- **Glavni sudionik:** Korisnik
- **Cilj:** Pregled osobnih podataka
- **Sudionici:** Baza podataka, web aplikacija
- **Preduvjet:** Korisnik je prijavljen
- **Opis osnovnog tijeka:**
 1. Korisnik odlazi na stranicu profila klikom na ikonu u gornjem desnom uglu
 2. Web-aplikacija prikazuje stranicu s podacima profila

UC5: Promjena informacija na profilu

- **Glavni sudionik:** Korisnik
- **Cilj:** Promjena/ažuriranje informacija profila
- **Sudionici:** Baza podataka, web aplikacija
- **Preduvjet:** Korisnik prijavljen na račun
- **Opis osnovnog tijeka:**
 1. Korisnik je na stranici profila
 2. Korisnik odabire opciju uređivanja informacija
 3. Korisnik dodaje nove osobne podatke ili mijenja postojeće
 4. Nakon promjene šalje potvrdu o promjeni
 5. Podaci se ažuriraju na stranici profila i u bazi podataka
 6. Korisnik dobiva pop-up potvrdu o uspješnoj promjeni podataka
- **Opis mogućih odstupanja:**
 1. (3) ako se korisnik odluči da ipak ne želi promijeniti podatke može odbaciti promjene

UC6: Slanje prijedloga recepta od strane korisnika

- **Glavni sudionik:** Korisnik
- **Cilj:** Prijedlog novog recepta
- **Sudionici:** Baza podataka, kuhari
- **Preduvjet:** Korisnik prijavljen na račun
- **Opis osnovnog tijeka:**
 1. Korisnik na korisničkom sučelju odabire opciju dodavanja recepta
 2. Korisniku se otvori obrazac za prijavu novog recepta, pri čemu sam upisuje atribute recepta (ime, sastojci, opis, vrijeme kuhanja, itd.) te dodaje sliku recepta
 3. Korisnik zatim šalje obrazac kuharu na provjeru te dobiva potvrdu o poslanom obrascu
 4. U bazi podataka se pojavljuje recept koji čeka na odobrenje (zastavica awaitingApproval = true)
- **Opis mogućih odstupanja:**
 1. (2) Korisnik pri dodavanju recepta može odustati od slanja zahtjeva povratkom na početnu stranicu ili odlaskom na stranicu profila

UC7: Komunikacija u chatu

- **Glavni sudionik:** Korisnik
- **Cilj:** Komunicirati s drugim korisnicima
- **Sudionici:** Sustav, drugi korisnici
- **Preduvjet:** Korisnik je prijavljen
- **Opis osnovnog tijeka:**
 1. Korisnik odabire chat opciju chata s drugim korisnicima

2. Korisnik unosi poruku i šalje je
3. Poruka se prikazuje svim korisnicima

UC8: Komunikacija u AI chatu

- **Glavni sudionik:** Korisnik
- **Cilj:** Koristiti AI chat radi prijedloga novih recepata i slično
- **Sudionici:** Sustav, AI modul
- **Preduvjet:** Korisnik je prijavljen
- **Opis osnovnog tijeka:**
 1. Korisnik odabire chat opciju AI chat
 2. Korisnik unosi poruku i šalje je
 3. Ako koristi AI chat, AI obrađuje upit i daje odgovor ili predlaže recept
 4. Korisnik prima povratnu informaciju ako je koristio AI chat

UC9: Spremanje vlastitog recepta na listu

- **Glavni sudionik:** Korisnik
- **Cilj:** Spremiti recept na privatnu listu bez provjere kuhara
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen
- **Opis osnovnog tijeka:**
 1. Korisnik u istom obrascu za prijedlog recepata postavi vidljivost recepta na "Privatno".
 2. Sustav sprema recept na listu vlastitih recepata korisnika

UC10: Pregled vlastite liste recepata

- **Glavni sudionik:** Korisnik
- **Cilj:** Pregledati vlastitu listu recepata
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen
- **Opis osnovnog tijeka:**
 1. Korisnik otvara stranicu profila.
 2. Pregledava sve recepte koje je sam napravio.

UC11: Ocjenjivanje i komentiranje recepta

- **Glavni sudionik:** Korisnik
- **Cilj:** Dati ocjenu receptu koji je isprobao.
- **Sudionici:** Web aplikacija, baza podataka
- **Preduvjet:** Korisnik je prijavljen i pregledava recept
- **Opis osnovnog tijeka:**
 1. Korisnik odabire ocjenu za recept
 2. Korisnik ispunjava polje za komentar
 3. Sustav bilježi ocjenu i ažurira prosječnu ocjenu recepta
 4. Drugi korisnici mogu vidjeti komentar

UC12: Dodavanje sastojaka u listu najdražih sastojaka

- **Glavni sudionik:** Korisnik
- **Cilj:** Stvaranje liste najdražih sastojaka korisnika
- **Sudionici:** Sustav, baza podataka
- **Preduvjet:** Korisnik je prijavljen i nalazi se na stranici profila
- **Opis osnovnog tijeka:**
 1. Korisnik često korištene/najdraže sastojke može dodati u listu najdražih sastojaka
 2. Po tim sastojcima mu sustav daje novu listu preporučenih recepata koji sadrže njegove najdraže namirnice

UC13: Pregledavanje recepata na temelju liste najdražih sastojaka

- **Glavni sudionik:** Korisnik
- **Cilj:** Pregled recepata sastavljenih po listi najdražih sastojaka
- **Sudionici:** Korisnik, baza podataka
- **Preduvjet:** Korisnik je prijavljen
- **Opis osnovnog tijeka:**
 1. Korisnik na temelju liste najdražih sastojaka na početnoj stranici dobiva personaliziranu listu recepata "Za tebe"
 2. Korisnik pregledava recepte s dobivene liste

UC14: Brisanje sastojaka iz liste omiljenih sastojaka

- **Glavni sudionik:** Korisnik
- **Cilj:** Brisanje sastojaka iz liste omiljenih sastojaka
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen
- **Opis osnovnog tijeka:**
 1. Korisnik po želji može mijenjati najdraže sastojke, bira opciju brisanja nekog od sastojaka klikom na sastojak koji ima u listi
 2. Sastojak se briše iz liste
 3. Lista personaliziranih recepata se nakon toga ažurira

UC15: Prihvaćanje/odbijanje recepta od strane kuhara

- **Glavni sudionik:** Kuhar
- **Cilj:** Prihvaćanje ili odbijanje novih recepata koje su poslali korisnici
- **Sudionici:** Baza podataka, korisnik
- **Preduvjet:** Kuhar prijavljen na račun, korisnik poslao prijedlog
- **Opis osnovnog tijeka:**
 1. Kuhar na sučelju ima listu poslanih predloženih recepata
 2. Iz liste bira recept, pregledava ga te ako takav recept već ne postoji u bazi podataka, prihvaća ga
 3. Nakon prihvaćanja recepta u bazi podataka se ažurira zastavica čekanja provjere (false)
 4. Novi recept učitava se na stranici, ažurira se baza podataka
- **Opis mogućih odstupanja:**

Kuhar procjenjuje da recept nije dovoljno dobar za objavu te odbija recept, pri čemu se recept briše iz baze podataka Sustav javlja kuharu da sličan recept već postoji, kuhar provjerava te ako se radi o istom receptu odbija ga

UC16: Izmjena detalja recepta

- **Glavni sudionik:** Korisnik
- **Cilj:** Izmijeniti detalje recepta
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik prijavljen na račun, nalazi se na stranici recepta
- **Opis osnovnog tijeka:**
 1. Korisnik nalazi na neku grešku u receptu
 2. Korisnik popravljiva grešku
 3. Recept se ažurira u bazi podataka

UC17: Dodavanje sastojaka u bazu podataka

- **Glavni sudionik:** Kuhar
- **Cilj:** Dodati neki novi sastojak u bazu podataka
- **Sudionici:** Baza podataka
- **Preduvjet:** Kuhar prijavljen na račun
- **Opis osnovnog tijeka:**
 1. Kuhar primjećuje da neki sastojak nedostaje u bazi podataka ili mu netko javi da doda određeni sastojak
 2. Kuhar dodaje novi sastojak
 3. Lista sastojaka se ažurira u bazi podataka
 4. Korisnici mogu pretraživati pomoću novog sastojka ili ga dodati na listu

UC18: Promocija korisnika u kuhara

- **Glavni sudionik:** Administrator
- **Cilj:** Promovirati korisnika u kuhara na temelju pozitivnih ocjena recepata
- **Sudionici:** Administrator, korisnik, sustav
- **Preduvjet:** Korisnik ima visoku ocjenu recepata
- **Opis osnovnog tijeka:**
 1. Administrator pregledava korisnikove ocjene i aktivnosti
 2. Ako zadovoljava uvjete, administrator promiče korisnika u kuhara
 3. Korisnik dobiva obavijest o promociji

UC19: Degradacija kuhara u običnog korisnika

- **Glavni sudionik:** Administrator
- **Cilj:** Vratiti kuhara u status običnog korisnika zbog niskih ocjena
- **Sudionici:** kuhar, sustav
- **Preduvjet:** Kuhar ima nizak prosjek ocjena svojih recepata
- **Opis osnovnog tijeka:**
 1. Administrator pregledava ocjene recepata kuhara.
 2. Administrator donosi odluku o degradaciji
 3. Kuhar dobiva obavijest o promjeni statusa
 4. Novi status kuharevog profila je korisnik, ažurira se u bazi podataka

UC20: Brisanje recepta

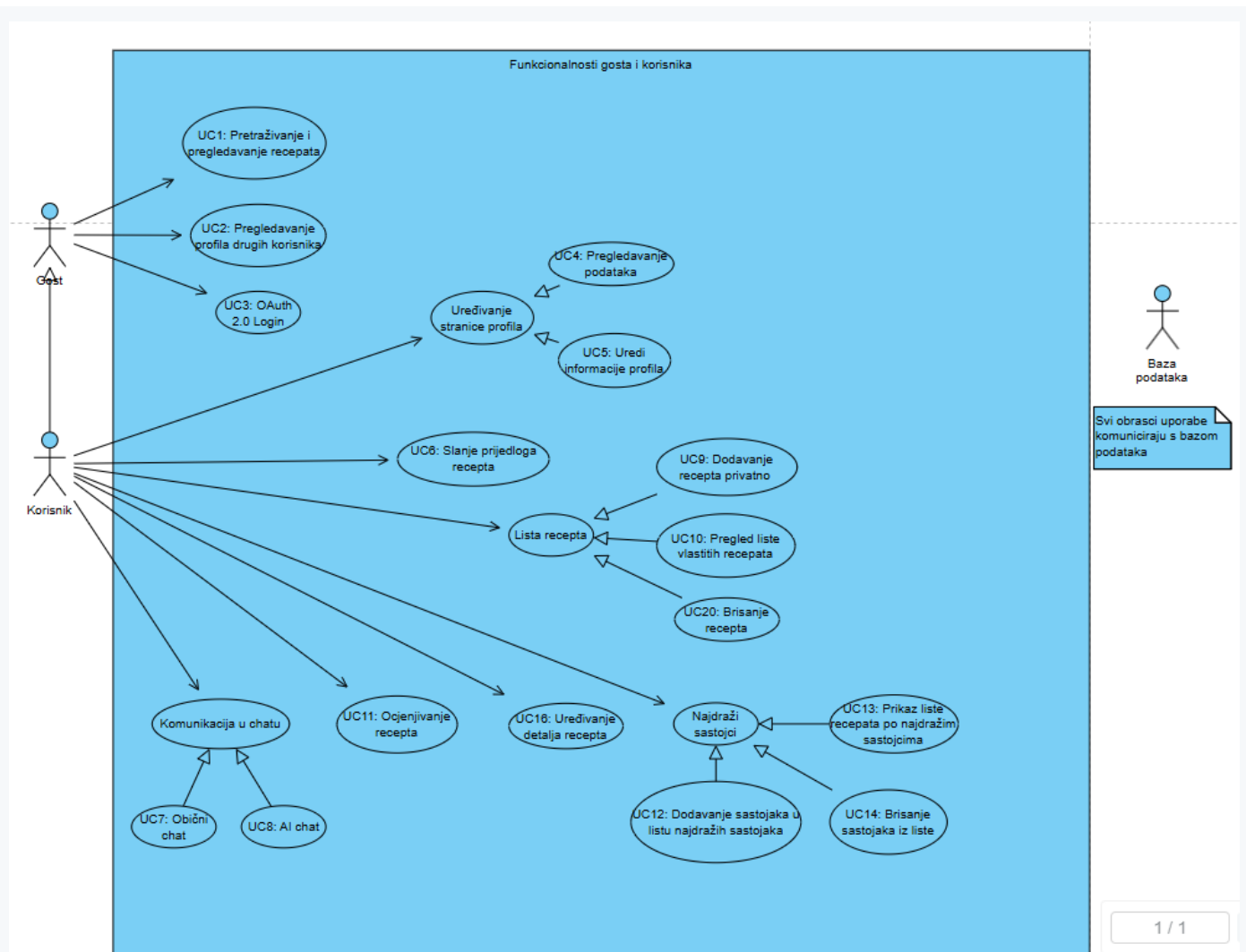
- **Glavni sudionik:** Korisnik
- **Cilj:** Ukloniti recept
- **Sudionici:** Baza podataka, sustav
- **Preduvjet:** Korisnik ulogiran na račun i pregledava recept
- **Opis osnovnog tijeka:**

Korisnik pregledava recept.

Odabire opciju za brisanje

Sustav uklanja recept s liste i briše ga iz baze podataka

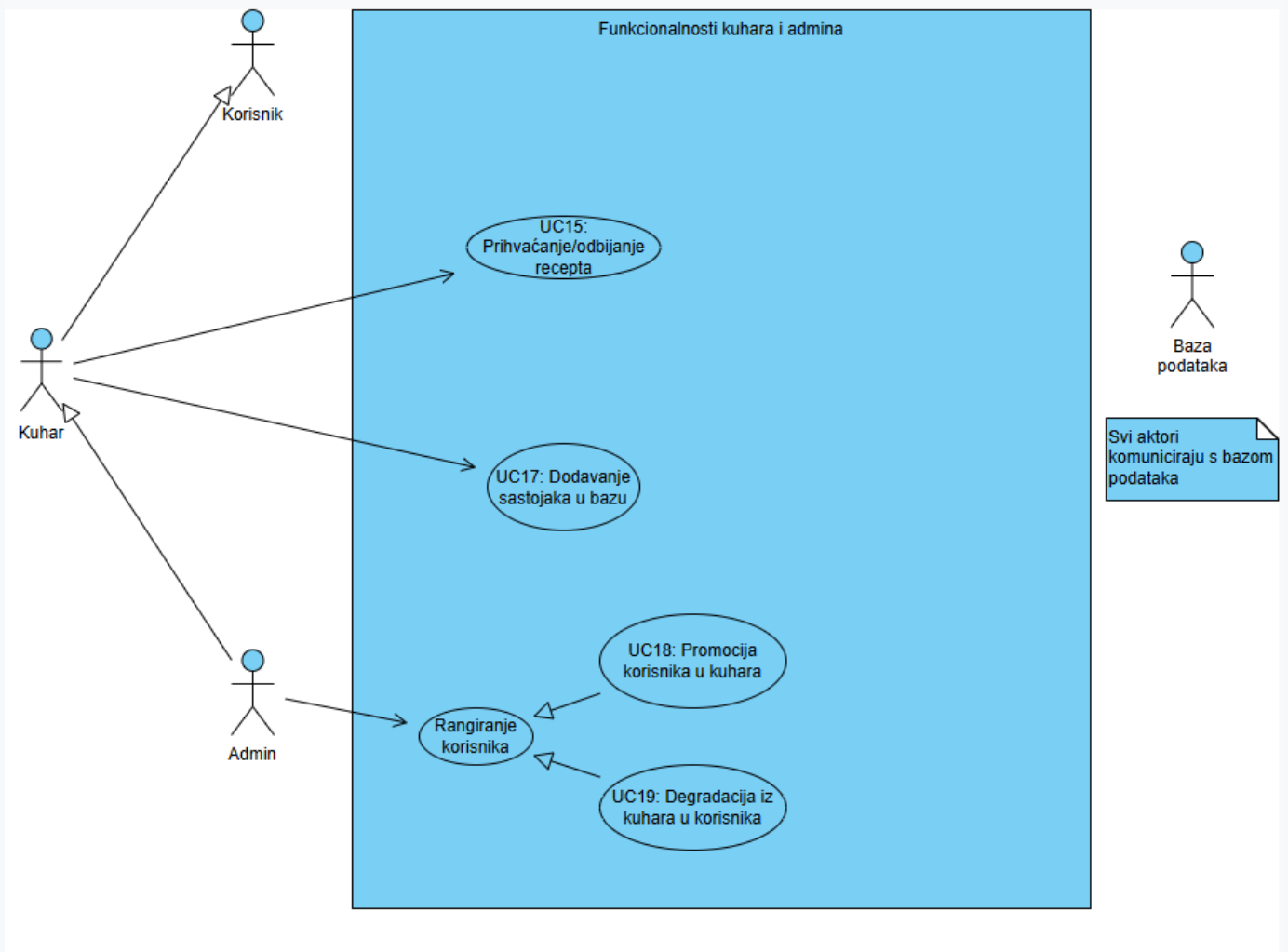
Dijagrami obrazaca uporabe



Dijagram 1: Funkcionalnosti gosta i korisnika

Opis prvog dijagrama obrazaca uporabe

- Ovim dijagramom prikazane su osnovni obrasci uporabe za goste i korisnike stranice
- Korisnik ima sve funkcionalnosti gosta (pretraga i pregled receptata, pregledavanje tuđih profila, login)
- Korisnik ima posebne obrasce uporabe koji uključuju uređivanje vlastitog profila, slanje prijedloga receptata
- Stvaranje liste vlastitih receptata i pregled iste, dodavanje i brisanje receptata iz vlastite liste
- Dodavanje i brisanje sastojaka iz liste najdražih sastojaka te prikaz liste receptata po najdražim sastojcima
- Ostavljanje recenzija na tuđim receptima
- Komunikacija u običnom chatu te komunikacija s AI chatbotom

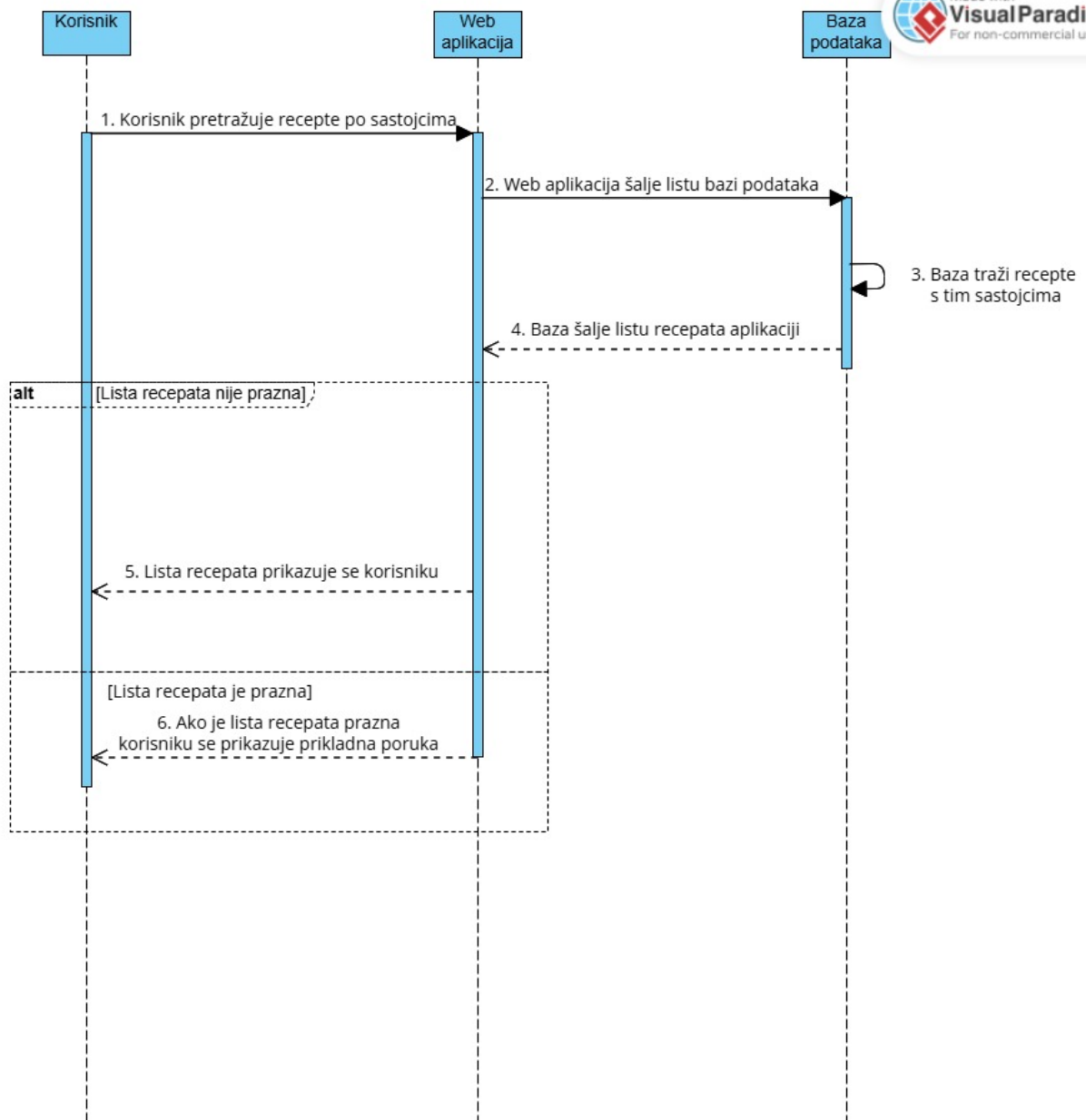


Dijagram 2: Funkcionalnosti kuhara i admina

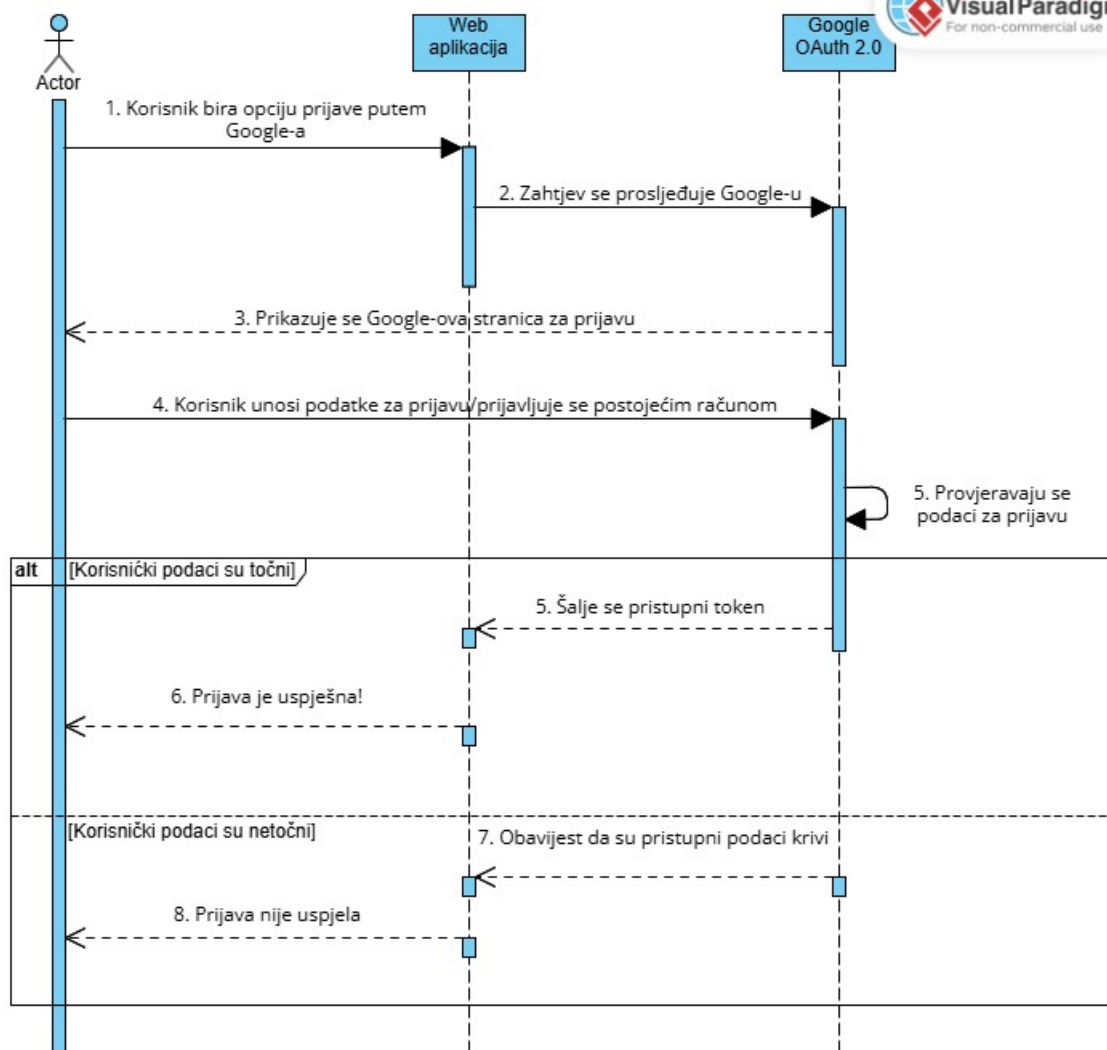
Opis drugog dijagrama obrazaca uporabe

- Ovim dijagramom prikazani su posebni obrasci uporabe verificiranog kuhara i administratora
- Kuhari imaju sve funkcionalnosti korisnika te dodatne kuharske ovlasti
- Administratori imaju sve funkcionalnosti kuhara te dodatne administratorske ovlasti
- Kuhari mogu prihvaćati i odbijati prijedloge recepata koje šalju korisnici, mijenjati detalje recepata te dodavati nove sastojke u bazu sastojaka
- Administratori mogu upravljati rangom korisničkih računa, promovirati korisnike u kuhare i degradirati kuhare u korisnike
- Administratori imaju kontrolu nad javnom listom recepata

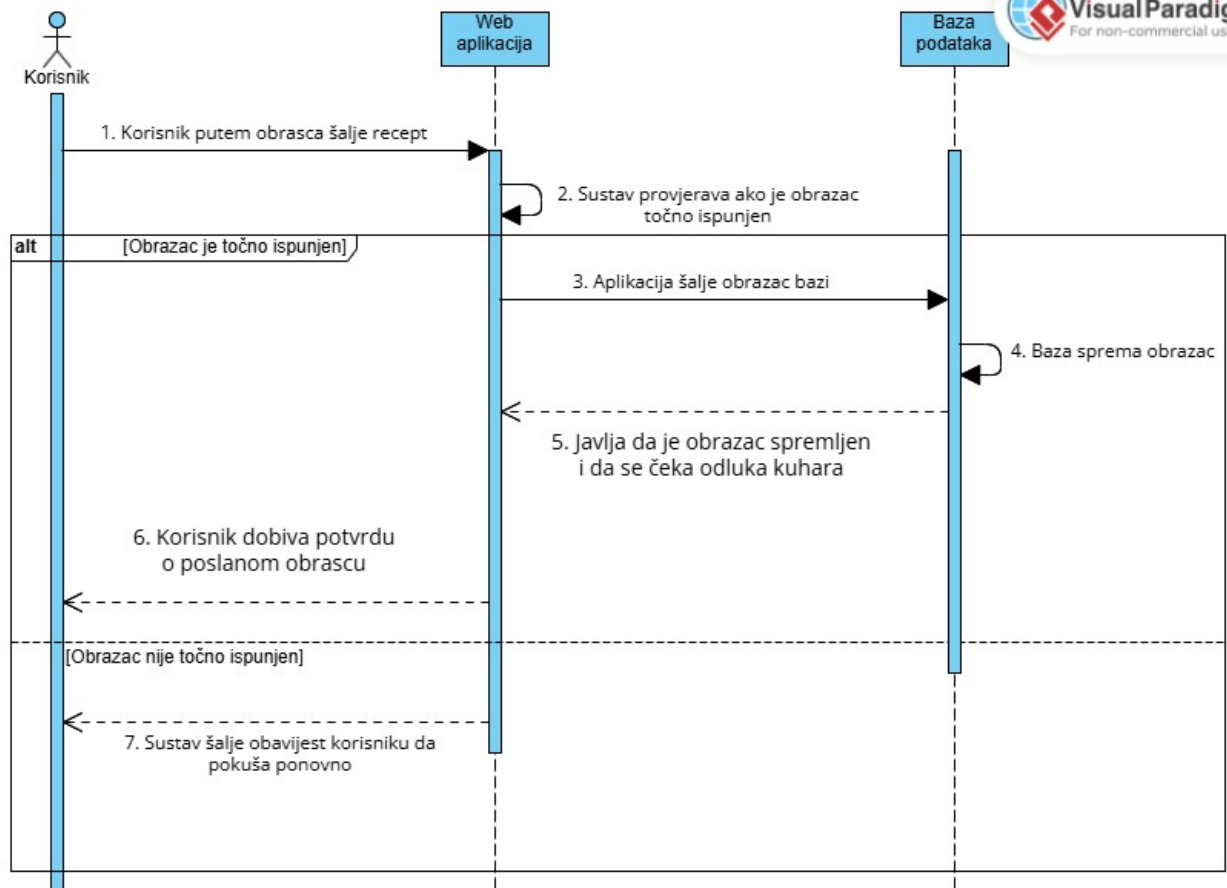
Sekvencijski dijagrami



Dijagram 3: Sekvencijski dijagram pretrage recepta po sastojcima

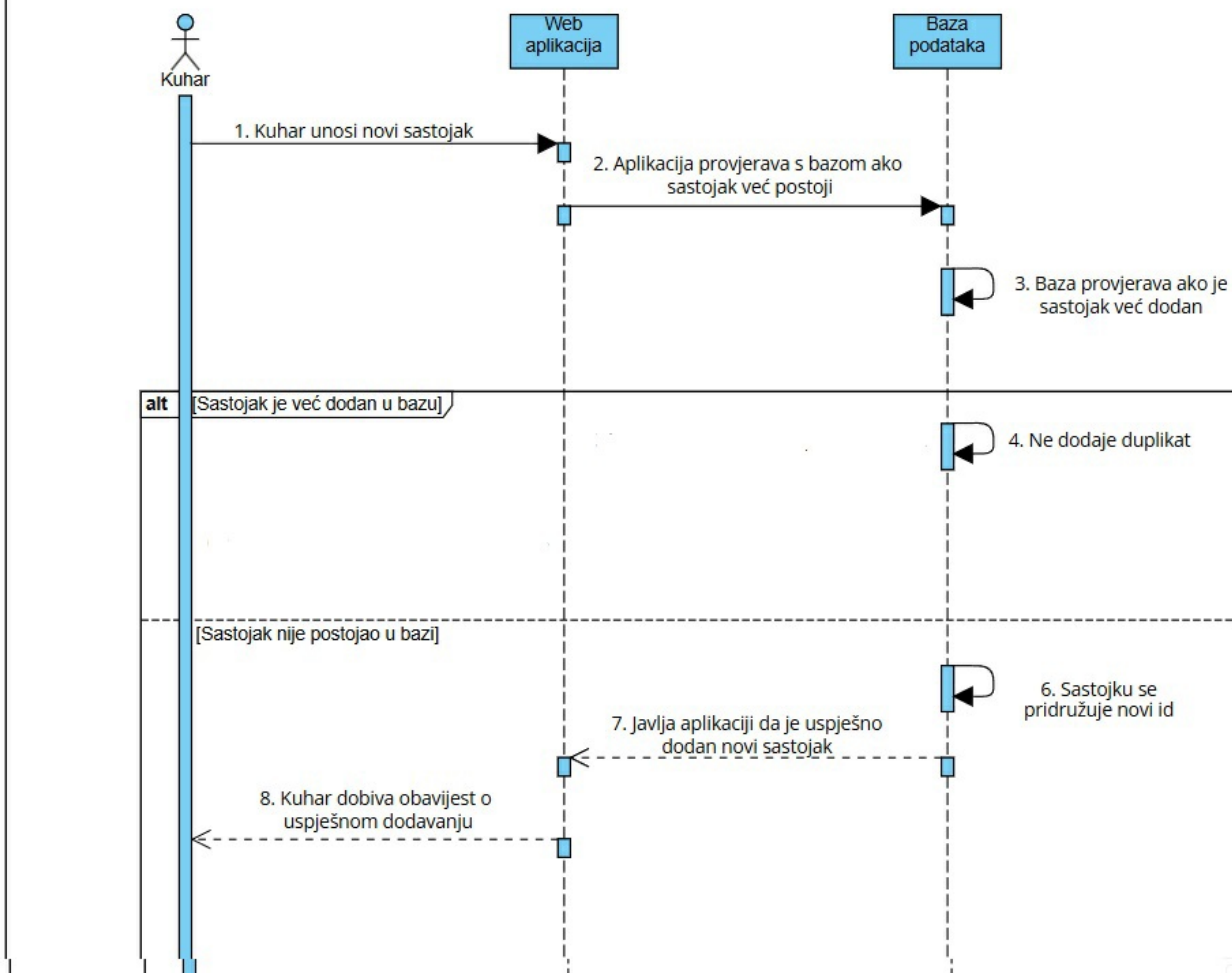


Dijagram 4: Sekvencijski dijagram prijave na korisnički račun preko Google OAuth2.0



Dijagram 5: Sekvencijski dijagram slanja prijedloga recepta korisnika

sd UC19: Dodavanje novih sastojaka u bazu podataka



Dijagram 6: Sekvencijski dijagram dodavanja novih sastojaka

Provjera uključenosti ključnih funkcionalnosti u obrasce uporabe

Use-Case id	Ime Use-Casea	Funkcionalni zahtjevi
UC1	Pregledavanje recepata na stranici	F-001
UC2	Prikaz javnog profila drugog korisnika	F-002

UC3 Use-Case id UC4	Prijava u sustav Ime Use-Casea Pregled korisničkih podataka	F-003 Funkcionalni zahtjevi F-004
UC5	Promjena osobnih podataka na profilu	F-005
UC6	Slanje prijedloga recepta od strane korisnika	F-006
UC7	Komunikacija u chatu	F-007
UC8	Komunikacija u AI chatu	F-008
UC9	Spremanje vlastitog recepta na listu	F-009
UC10	Pregled vlastite liste recepata	F-010
UC11	Ocjenjivanje i komentiranje recepta	F-011
UC12	Dodavanje sastojaka u listu najdražih sastojaka	F-012
UC13	Pregledavanje recepata na temelju liste najdražih sastojaka	F-013
UC14	Brisanje sastojaka iz liste omiljenih sastojaka	F-014
UC15	Prihvatanje/odbijanje recepta od strane kuhara	F-015
UC16	Izmjena detalja recepta	F-016
UC17	Dodavanje sastojaka u bazu podataka	F-017
UC18	Promocija korisnika u kuhara	F-018
UC19	Degradacija kuhara u običnog korisnika	F-019
UC20	Brisanje recepta	F-020

4. Arhitektura i dizajn sustava

Arhitektura sustava može se podijeliti na tri glavna dijela:

- **Web preglednik (klijent):** Web preglednik služi kao korisničko sučelje za interakciju s aplikacijom. Korisnici pristupaju web stranicama putem preglednika, što olakšava komunikaciju s poslužiteljem po modelu zahtjev-odgovor. Klijent pokreće zahtjeve, šalje podatke poslužitelju i obrađuje odgovore za prikaz traženih informacija ili potvrdu prijena podataka. Preglednik također upravlja ispravnim prikazivanjem elemenata korisničkog sučelja.
- **Web poslužitelj (backend):** Djeluje kao posrednik između klijenta i baze podataka te je odgovoran za obradu dolaznih zahtjeva i slanje odgovora natrag klijentu. Upravlja poslovnom logikom i manipulacijom podacima, koristeći RESTful API preko HTTP protokola. Po primitku zahtjeva, poslužitelj ili dohvaća podatke iz baze podataka ili pohranjuje podatke koje je poslao klijent.
- **Baza podataka:** Baza podataka je glavni sloj za pohranu u kojem se održavaju svi podaci aplikacije. Sadrži trajne podatke, uključujući korisničke informacije, postavke aplikacije i sadržaj. Sloj baze podataka podržava učinkovito postavljanje upita i osigurava dosljednost podataka. Gotovo svi zahtjevi kojima upravlja web poslužitelj uključuju komunikaciju s bazom podataka za čitanje ili pohranjivanje informacija.

Odabrani stil arhitekture: **Klijent-poslužitelj s RESTful API-jem**. Ovaj stil je idealan jer omogućuje:

1. **Razdvajanje problema:** Klijentska i poslužiteljska strana su razdvojene, što znači da se razvijaju, implementiraju i skaliraju neovisno jedni o drugima, omogućujući učinkovito upravljanje resursima.
2. **Učinkovito rukovanje podacima:** Poslužiteljska strana djeluje kao pružatelj podataka, ustupajući API-je klijentskoj strani. Ovaj pristup je uobičajen u Single Page Applications (SPA), gdje klijentska strana obrađuje korisničke interakcije i dinamički ažurira korisničko sučelje na temelju API podataka.
3. **Poboljšano korisničko iskustvo:** SPA pristup smanjuje ponovno učitavanje stranica.

Tehnologije korištene u razvoju

- **Poslužiteljski dio:** Ostvarujemo programskim jezikom Java uz Spring radni okvir s pomoću alata Spring-Boot. Za razvoj backenda koristimo IntelliJ IDEA razvojno okruženje jer podržava Spring-Boot alate.
- **Klijentski dio:** Ostvarujemo programskim jezikom JavaScript uz React radni okvir, zbog jednostavnosti implementacije SPA pristupa. Za razvojno okruženje koristimo WebStorm, koje je također razvijeno od strane JetBrains-a, promovirajući konzistentnost u korištenju tehnologija.
- **Dizajn korisničkog sučelja:** Za dizajn korisničkog sučelja koristimo alat Figma.

Baza podataka

Koristimo relacijsku bazu podataka, implementiranu pomoću PostgreSQL-a. Relacijska baza podataka organizira podatke u strukturirane tablice definirane imenom i skupom atributa. Baza nam služi za brzu pohranu, izmjenu i dohvat podataka u skladu s korisničkim zahtjevima. Entiteti unutar naše baze podataka su: - Person - Role - Rating - Recipe - Ingredient - FavoriteIngredients

Opis tablica

Person

Predstavlja korisnika platforme, uključujući opće članove i kuhare. Sadrži osobne podatke poput imena, biografije, e-adrese te njihove uloge na platformi. Tablica je povezana: Many-to-One s Role preko RoleId, One-to-Many s Recipe preko PersonId, One-to-Many s Rating preko PersonId i Many-to-Many s Ingredient preko IngredientId unutar FavoriteIngredients. Atributi: PersonId, FirstName, LastName, About, Username, Email, ProfileImage, RoleId (FK).

Atribut	Tip podatka	Opis varijable
PersonId	BIGSERIAL	Jedinstveni identifikator svakog korisnika/osobe
FirstName	VARCHAR(50)	Ime osobe
LastName	VARCHAR(50)	Prezime osobe
About	VARCHAR	Opis o vlasniku profila
Username	VARCHAR(50)	Korisničko ime - jedinstveno
Email	VARCHAR(100)	Adresa e-pošte osobe
ProfileImage	VARCHAR(255)	Profilna slika osobe
RoleId	INTEGER	Identifikator uloge osobe

Role

Definira ulogu osobe na platformi, kao što je "Kuhar" (Chef) ili "Korisnik" (User). To pomaže u upravljanju korisničkim dopuštenjima i razlikovanju vrsta korisnika. Tablica je povezana: One-to-Many sa Person preko RoleId. Atributi: RoleId, Name.

Atribut	Tip podatka	Opis varijable
RoleId	BIGSERIAL	Jedinstveni identifikator svake uloge
Name	VARCHAR(50)	Naziv uloge

Rating

Korisnicima omogućuje ocjenjivanje i komentiranje recepata. Atribut "Grade" predstavlja danu ocjenu, a "Comment" je izborna tekstualno polje za povratne informacije. Tablica je povezana: Many-to-One sa Recipe preko Recipeld, Many-to-One sa Person preko PersonId. Atributi: Recipeld (FK), PersonId (FK), Grade, Comment.

Atribut	Tip podatka	Opis varijable
Recipeld	INTEGER	Jedinstveni identifikator recepta za kojeg je dana recenzija
PersonId	INTEGER	Jedinstveni identifikator osobe
Grade	SMALLINT	Ocjena recepta, vrijednosti od 1 do 5
Comment	VARCHAR(1000)	Komentar uz ocjenu

Recipe

Pohranjuje detalje o receptu, uključujući korake za pripremu, potrebno vrijeme, opis i povezanog kuhara ili korisnika koji ga je predložio. Atribut "Publicity" označava je li recept javan ili privatn, a "WaitingApproval" označava je li za njega potrebno odobrenje administratora. Tablica je povezana: Many-to-One s Person preko UserId, One-to-Many s Rating preko PersonId i Recipeld, Many-to-Many s Ingredient preko IngredientId unutar RecipeIngredients. Atributi: Recipeld, Procedure, Publicity, TimeToCook, Title, Description, WaitingApproval, RecipeImage, ChefId (FK), UserId (FK).

Atribut	Tip podatka	Opis varijable
Recipeld	BIGSERIAL	Jedinstveni identifikator svakog recepta
Procedure	VARCHAR	Procedura izrade jela
Publicity	BOOLEAN	Je li recept javan ili privatn
TimeToCook	INTEGER	Opis o vlasniku profila
Title	VARCHAR(50)	Naslov recepta
Description	VARCHAR(100)	Kratki opis recepta
WaitingApproval	BOOLEAN	Čeka li recept dopuštenje kuhara za javnu objavu
RecipeImage	VARCHAR(255)	Slika recepta
ChefId	INTEGER	Identifikator kuhara
UserId	INTEGER	Identifikator korisnika

Ingredient

Popis sastojaka koji se koriste u receptima. Svaki sastojak ima jedinstveni identifikator i naziv. Tablica je povezana: Many-to-Many s Recipe preko Recipeld unutar RecipeIngredients, Many-to-Many s Person preko PersonId unutar FavoriteIngredients. Atributi: IngredientId, Name.

Atribut	Tip podatka	Opis varijable
IngredientId	BIGSERIAL	Jedinstveni identifikator svakog sastojka
Name	VARCHAR(50)	Naziv sastojka

FavoriteIngredients

Definira odnos Many-to-Many između ljudi i njihovih omiljenih sastojaka, spaja tablice Person i Ingredient. Atributi: PersonId (FK), IngredientId (FK).

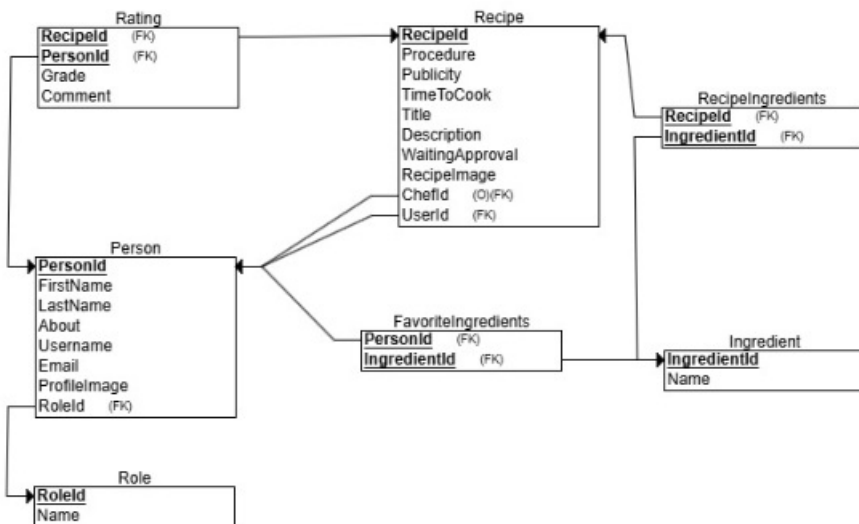
Atribut	Tip podatka	Opis varijable
PersonId	INTEGER	Jedinstveni identifikator osobe
IngredientId	INTEGER	Jedinstveni identifikator sastojka

RecipeIngredients

Definira odnos Many-to-Many između recepata i sastojaka, određujući koji se sastojci koriste u svakom receptu. Atributi: Recipeld (FK), IngredientId (FK).

Atribut	Tip podatka	Opis varijable
Recipeld	INTEGER	Jedinstveni identifikator recepta
IngredientId	INTEGER	Jedinstveni identifikator sastojka

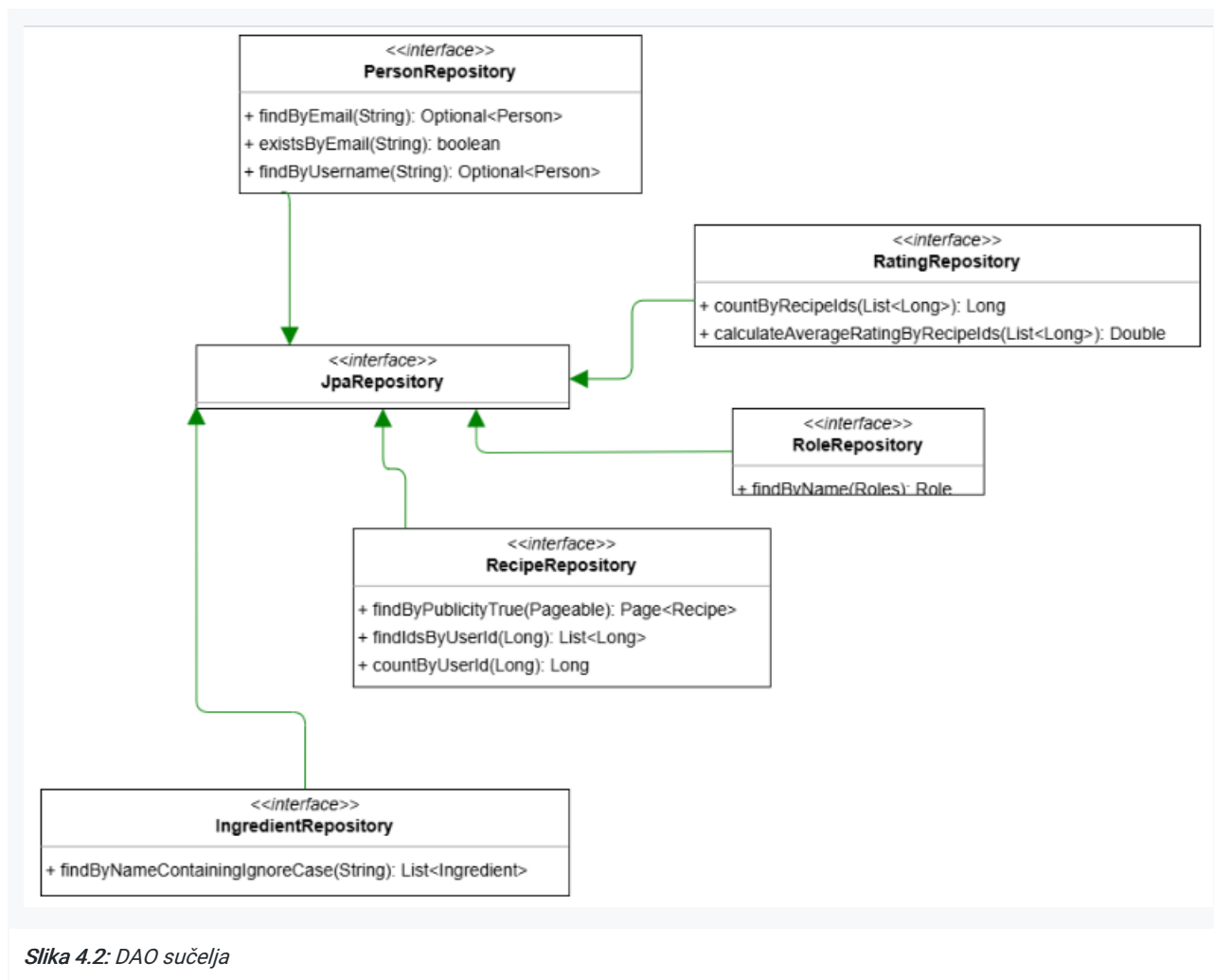
Dijagram baze podataka



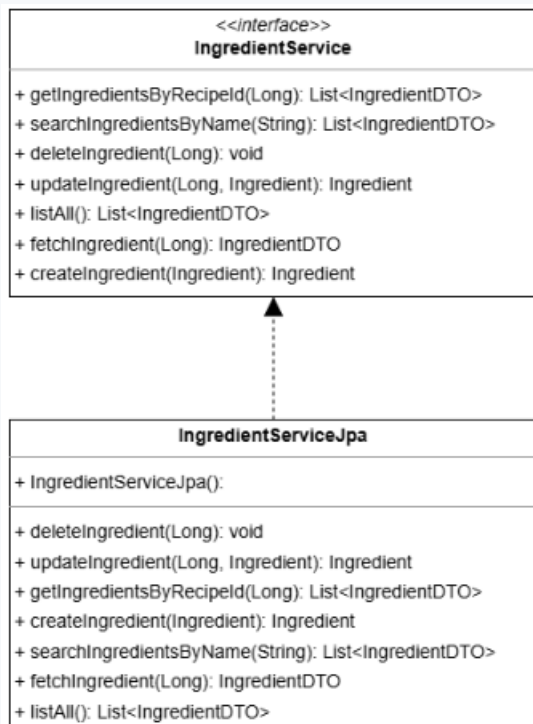
Slika 4.1: E-R dijagram baze podataka

Dijagram razreda

Na slikama 4.2 i 4.3 prikazani su dijagrami razreda odvojeni po paketima unutar pozadinskog djela (*backend*) ove Spring Java aplikacije. Na slici 4.2 prikazani su razredi unutar dao paketa, tj *data access objects*. Ovo su zapravo sučelja (*interface*) koja nasljeđuju Spring Data JPA repozitorije i pomažu u pojednostavljenju CRUD operacija. Sljedeće unutar dto paketa, koji je unutar domain paketa, su tzv. *data transfer objects*. Ove klase koriste se za prijenos podataka između slojeva i odvajanje prednjeg dijela aplikacije od pozadinskog. Pojedini element baze može imati više DTO-a za različite primjene, zato u primjeru naše aplikacije Recipe element ima čak 4 DTO klase. Dalje, isto unutar domain paketa imamo i mapper klase, one mapiraju entity klase, klase koje reprezentiraju tablice unutar baze podataka, u dto objekte. Zatim imamo rest paket koj sadrži REST Controller klase, one definiraju REST krajnje točke (*endpoints*) i obrađuju dolazne HTTP zahtjeve, komuniciraju sa uslužnim slojevima i šalju odgovore klijentu. Zadnji paket jest service paket čije su klase prikazane na slici 4.6 i one reprezentiraju uslužni sloj te upravljaju poslovnom logikom aplikacije. Imamo [NazivObjekta]Service sučelje koje implementiraju [NazivObjekta]ServiceJpa klase, one upravljaju funkcijama u kojima pozivamo objekte iz baze koristeći *DAO* klase (repozitorije), te ih mapiramo pomoću *mapper* objekata. Za potrebe implementacije live chat-a i ai-chat-a koristimo klase unutar components i config paketa, one upravljaju nad radom s websocketima pomoću kojih je ostvarena komunikacija. Klase za impčmentaciju chat-a prikazane su na slikama 4.5 i 4.6.

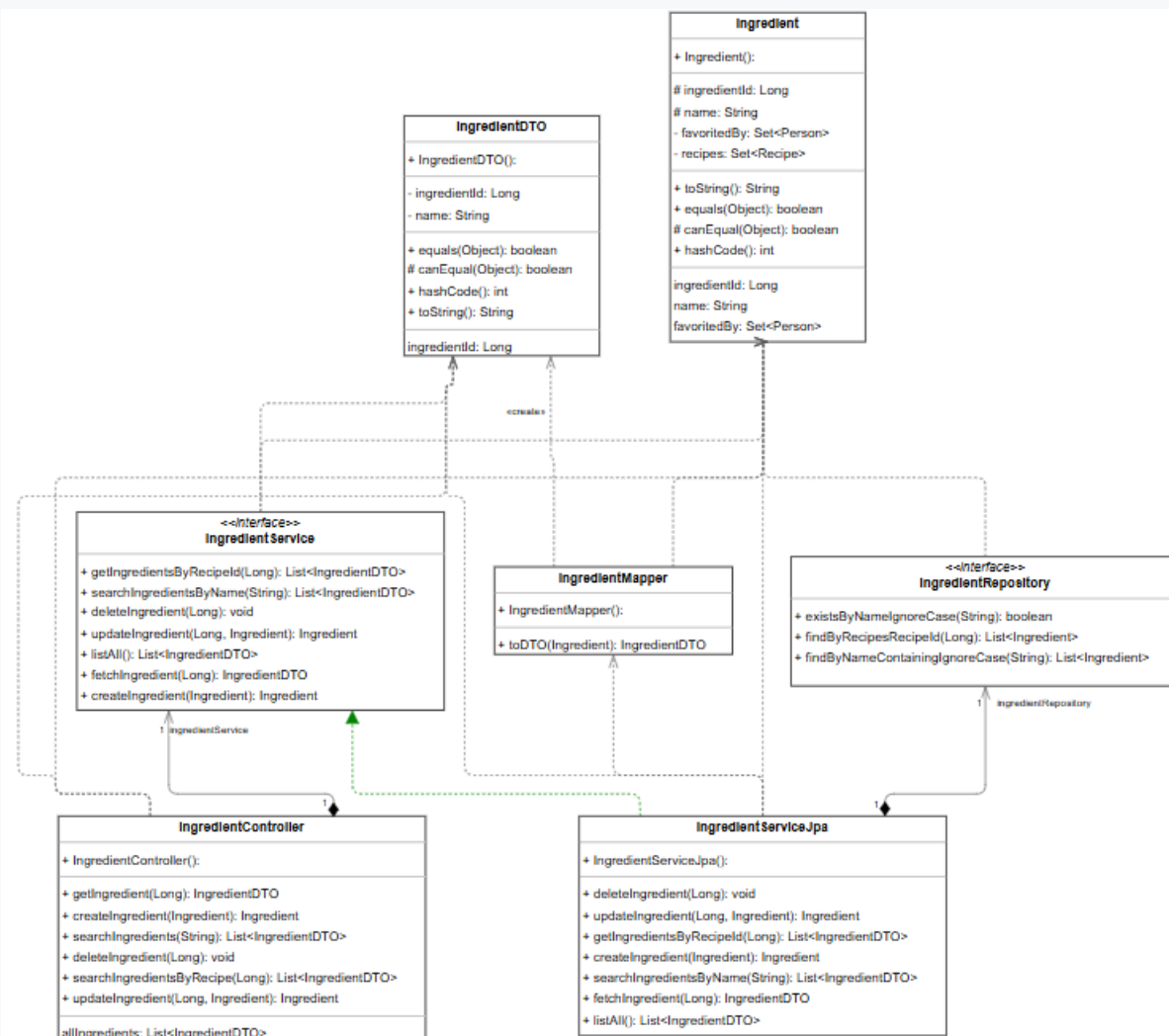


Slika 4.2: DAO sučelja

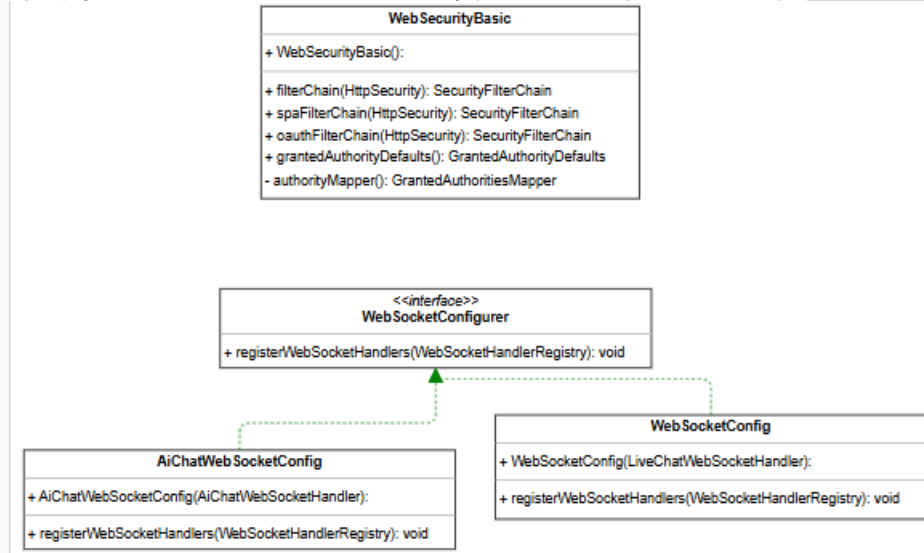


Slika 4.3: Prikaz Service klasa

Za prikaz cijelog sustava koristit ćemo sliku 4.4 koja prikazuje rad cijele aplikacije na manjoj skali, tj. na primjeru *Ingredients* entiteta. Ovaj prikaz dosljedan je tomu kako su i drugi entiteti povezani sa ostalim, prije navedenim i prikazanim, slojevima i klasama. Naime, zato što sveukupno ima oko 50 klasa među svim paketima, zbog preglednosti smo za prikaz cijele aplikacije odabrali ovako umanjenu skalu.



Uz ovakav prikaz važno je napomenuti da za primjere *Rating*, *Recipe* i *Person* imamo dodatne klase unutar DTO sloja. Ove dodatne klase povezuju se sa zasebnim *Controller*, *Service* i *Mapper* klasama, kako je prikazano na slici 4.8 na primjeru *Recipe* klase.



Slika 4.6: Prikaz config klasa

Dinamičko ponašanje aplikacije

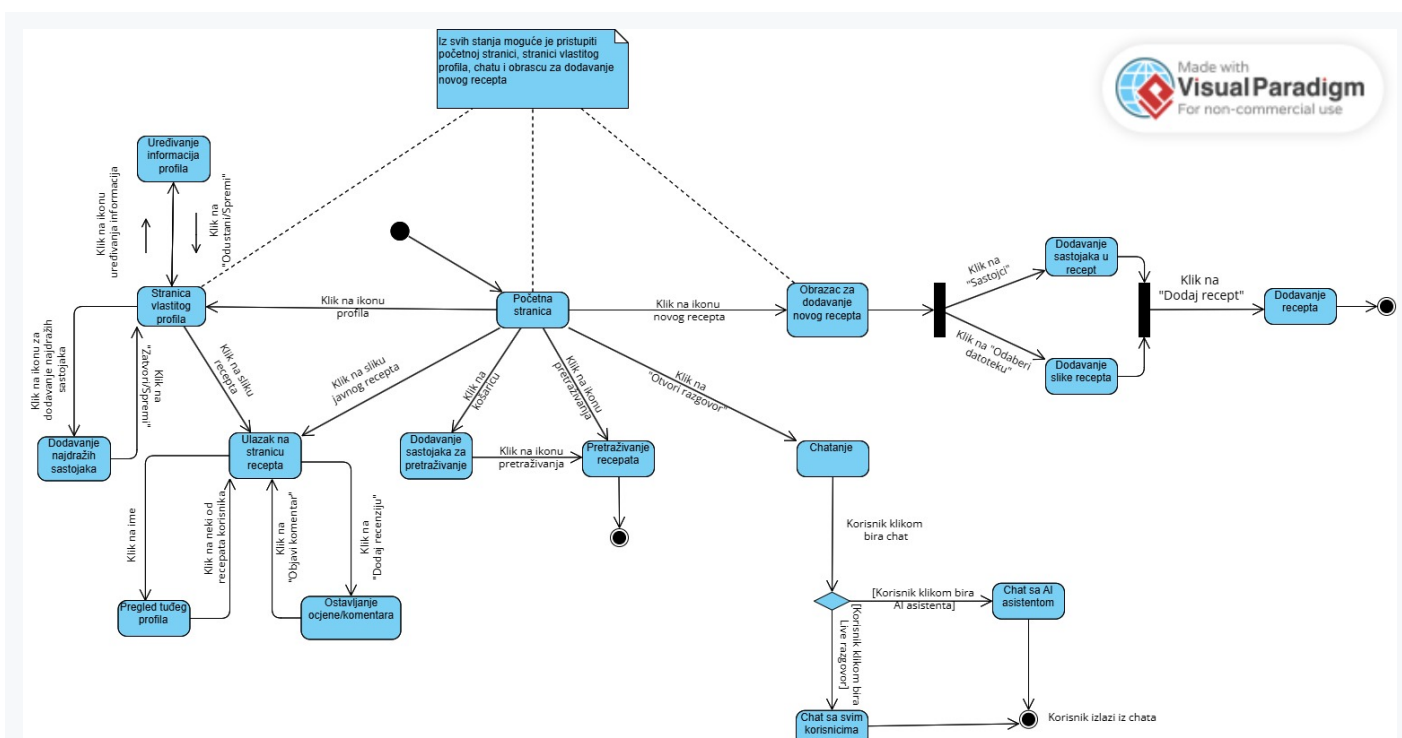
Dijagram stanja

Na slici 5.1 nalazi se dijagram stanja za web aplikaciju. Korisnik u svakom trenutku ima pristup početnoj stranici, vlastitom profilu, chatu i obrascu za dodavanje recepta.

Na stranici profila korisnik može uređivati informacije, gledati vlastite recepte i dodavati najdraže sastojke.

Na obrascu za stvaranje novog recepta korisnik može dodavati sastojke, može dodati sliku i u konačnici može poslati zahtjev za dodavanjem recepta. Korisnik također može s početne stranice ući na neki od recepata, te ga onda može komentirati i ostaviti ocjenu.

Na početnoj stranici može pretraživati recepte, po sastojcima, imenu recepta ili po dostupnom vremenu.



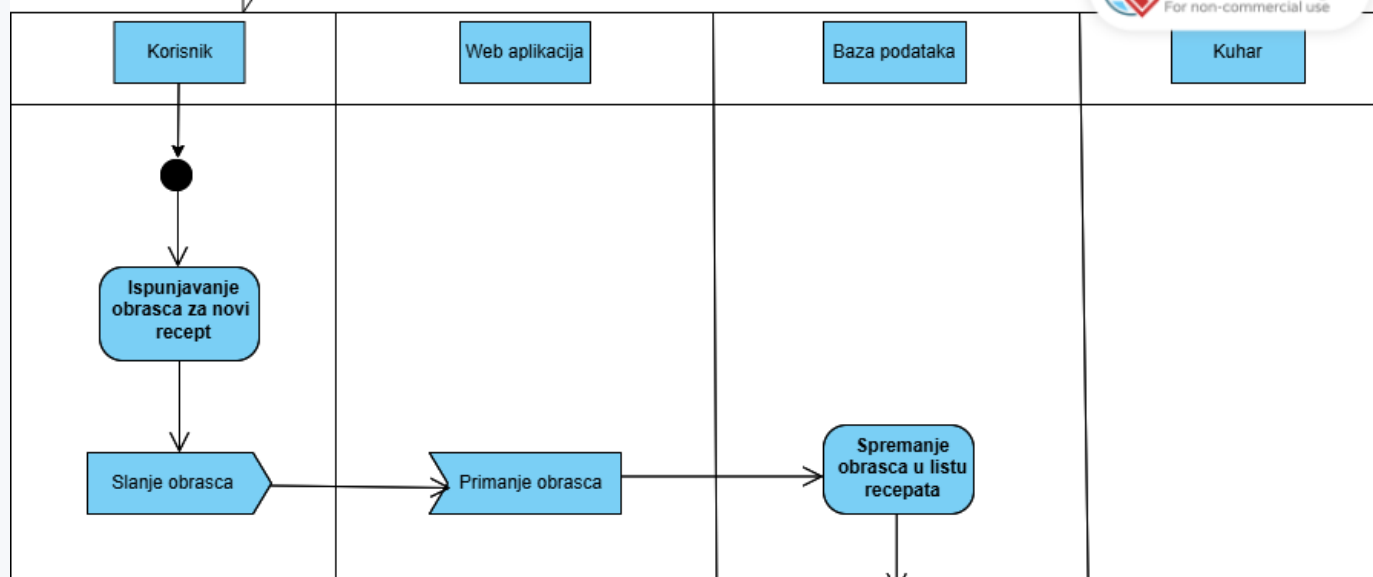
Slika 5.1: Dijagram stanja

Dijagram aktivnosti

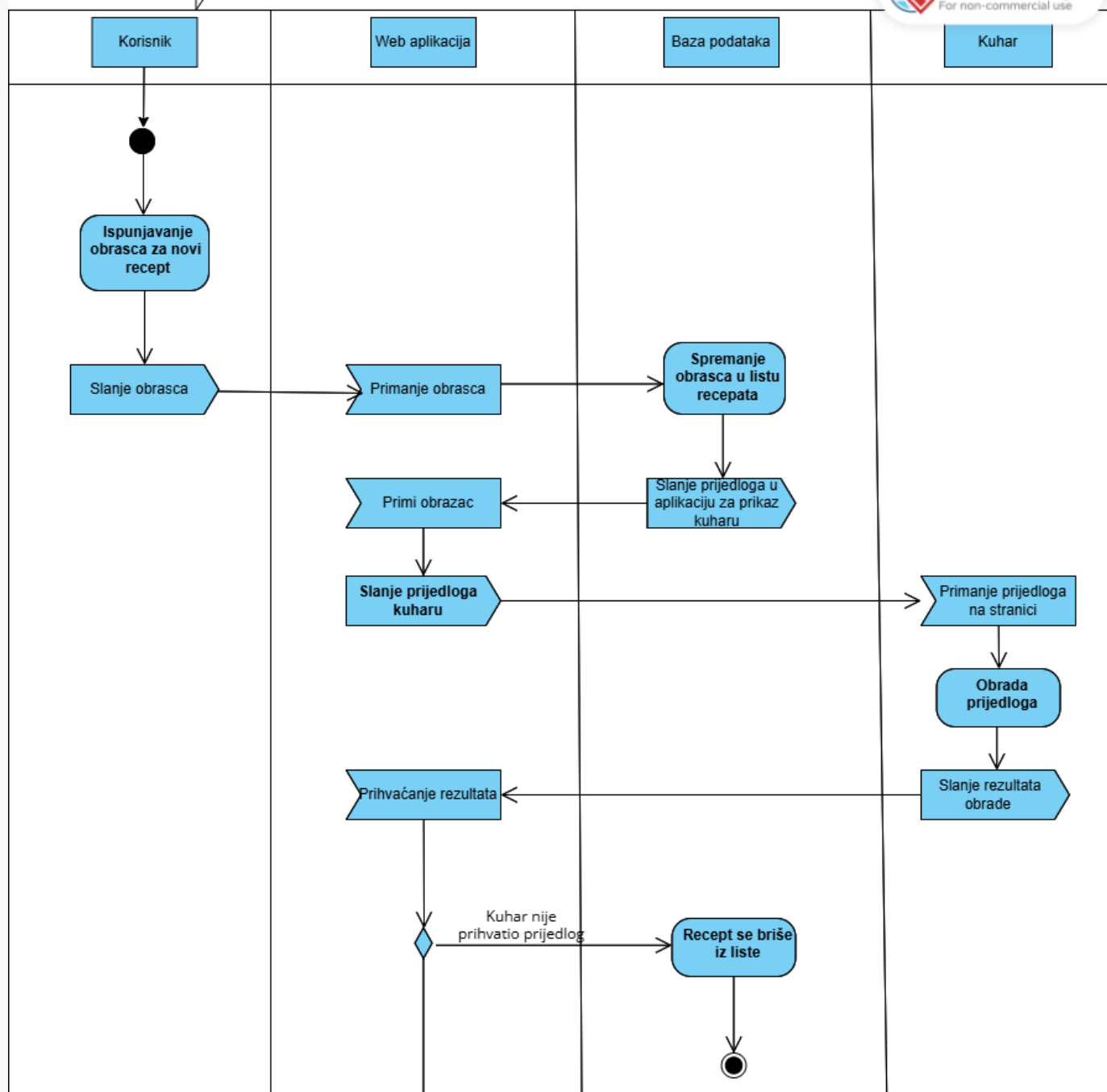
Na slici 5.2 nalazi se dijagram aktivnosti za web aplikaciju. Dijagram aktivnosti prikazuje slučaj korisnikovog slanja prijedloga recepta na provjeru. Aktori su korisnik, web aplikacija, baza podataka i kuhar koji provjerava recept. Interakcija se obavlja u idućih nekoliko koraka:

1. Korisnik šalje obrazac na provjeru u web aplikaciji
2. Web aplikacija prihvata zahtjev i sprema u bazu prijedlog recepta u listu recepta (recept nije verificiran stoga nije vidljiv)
3. Kuhar u web aplikaciji odobrava/poništava prijedlog
4. U bazi recept postaje vidljiv/briše iz liste, ovisno o odluci kuhara
5. U slučaju prihvatanja, nakon ažuriranja vidljivosti recepta web aplikacija prikazuje korisnicima recept

act - Dijagram aktivnosti: Slanje prijedloga recepta



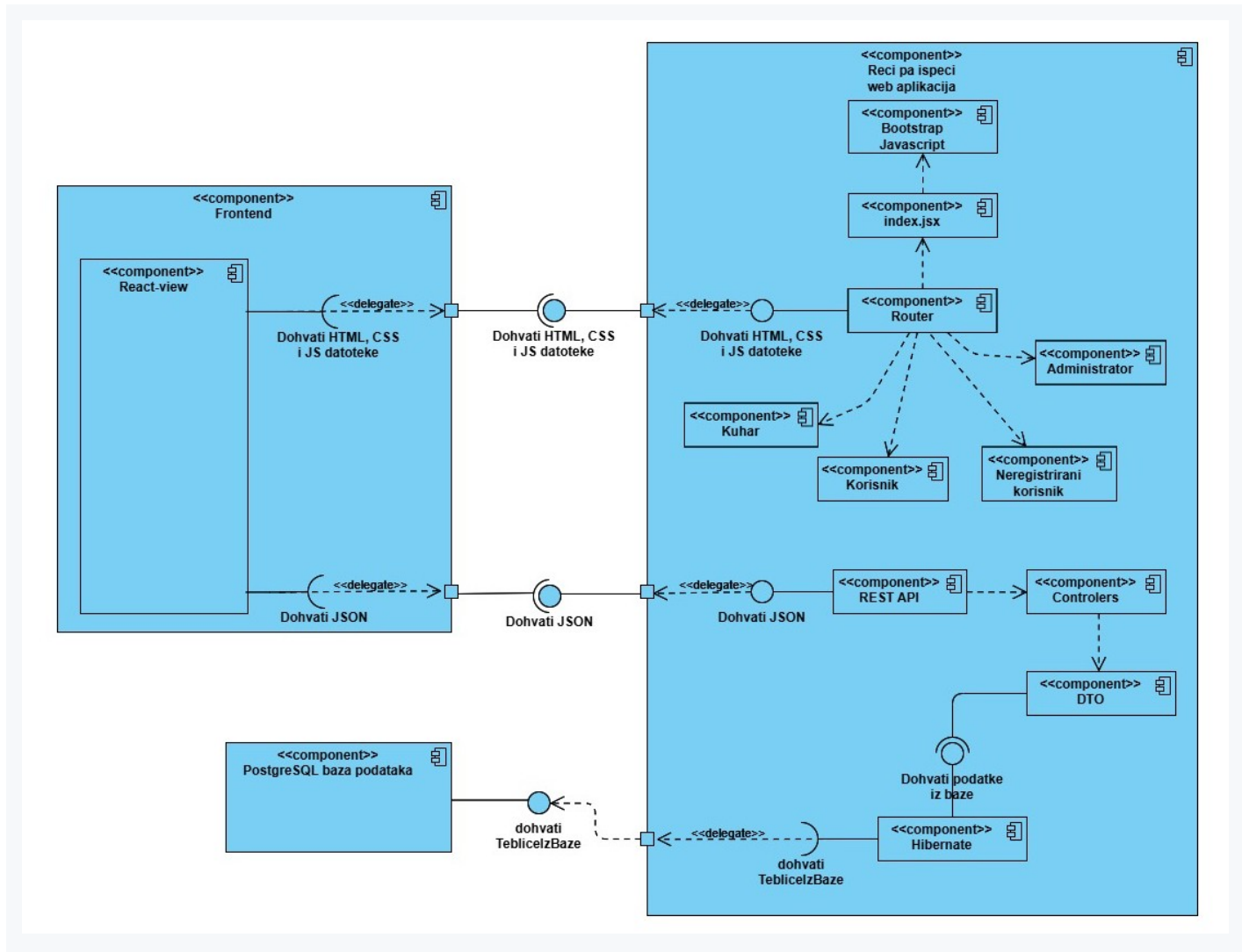
act - Dijagram aktivnosti: Slanje prijedloga recepta



Arhitektura sustava predstavlja temeljni okvir za razumijevanje i implementaciju svih njegovih funkcionalnosti. U kontekstu razvojne dokumentacije aplikacija, dijagrami komponenata i razmještaja odlučujući su za prikaz povezanosti i rasporeda različitih komponenata sustava. Ovi dijagrami omogućuju sudionicima projekta razumijevanje i vizualizaciju fizičkog i logičkog dizajna sustava, uključujući interakcije između dijelova aplikacije, što je odlučujuće za efikasnu implementaciju i dugoročnu održivost sustava.

Dijagram komponenata

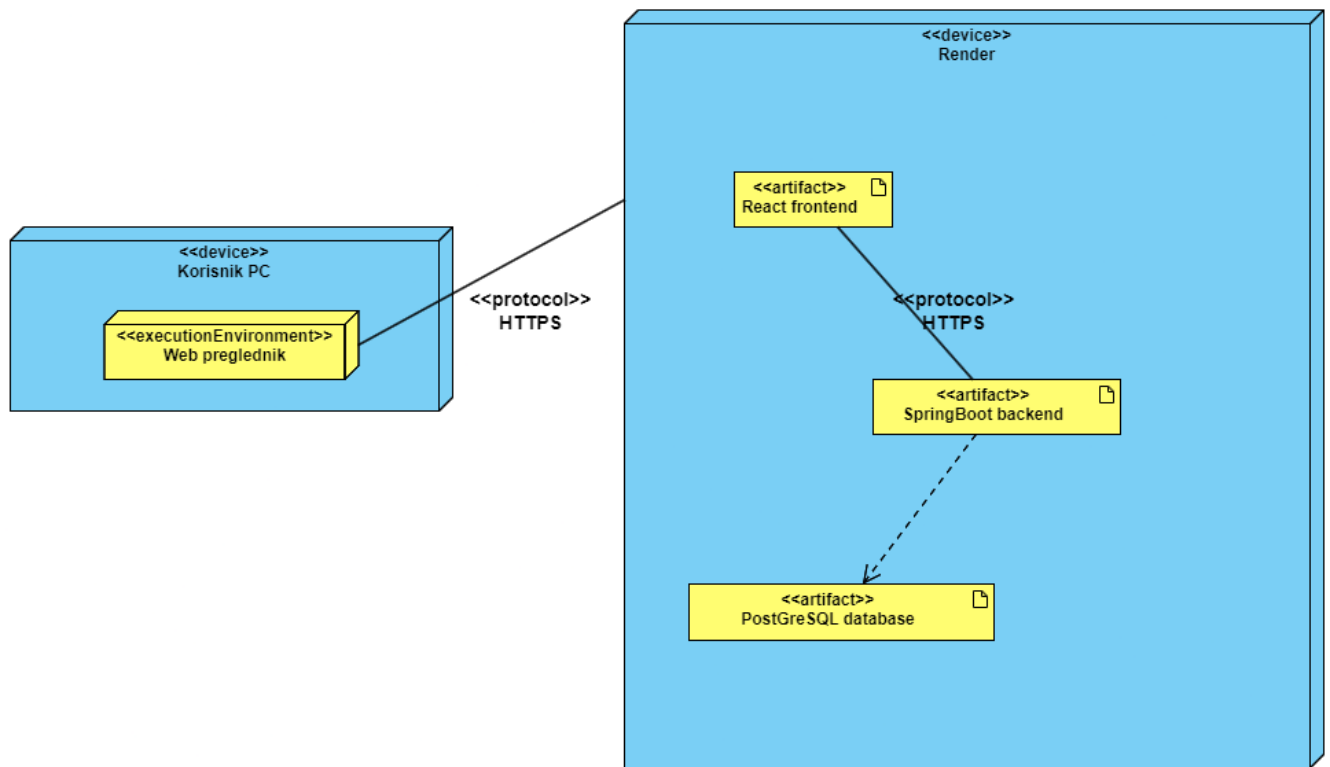
Dijagram komponenti opisuje organizaciju i međusobnu povezanost dijelova sustava, uključujući njihovu strukturu i odnose prema vanjskom okruženju. Ovome sustavu se pristupa preko dva različita sučelja. Preko sučelja za dohvat HTML, CSS i JavaScript datoteka poslužuju se datoteke koje pripadaju frontend dijelu aplikacije. Router u frontend aplikaciji određuje koje će komponente i datoteke biti prikazane korisniku na temelju URL-a. Frontend se sastoji od niza JavaScript datoteka organiziranih prema tipovima korisnika koji im pristupaju, dok se za implementaciju elemenata korisničkog sučelja poput gumba, formi i sličnoga koristi Bootstrap biblioteka. React-view komponenta koristi REST API za komunikaciju s backendom i na temelju korisničkih akcija osvježavaju prikaz sučelja, dohvaćaju nove podatke ili resurse. Preko sučelja za dohvat JSON podataka pristupa se REST API komponenti, koja predstavlja backend dio aplikacije izrađen pomoću Spring Boota. REST API koristi arhitekturu MVC (Model-View-Controller), gdje kontroler obrađuje korisničke zahtjeve i komunicira s modelima. Za prijenos podataka između slojeva aplikacije koriste se DTO-ovi (Data Transfer Objects). Backend koristi ORM alat Hibernate za dohvaćanje i upravljanje podacima iz PostgreSQL baze podataka. Dohvaćeni podaci obrađuju se unutar backenda i vraćaju u frontend aplikaciju putem REST API-ja u obliku JSON objekata.



Slika 5.1: Prikaz dijagrama komponenata

Dijagram razmještaja

Dijagram razmještaja se korist kako bismo razjasnili arhitekturu sustava. Sustav je baziran na arhitekturi "klijent-poslužitelj". Korisnički pristup aplikaciji odvija se preko web preglednika. Na platformi Render smješteni su poslužitelj za frontend, poslužitelj za backend i poslužitelj za bazu podataka. Preko HTTPS protokola ostvarena je komunikacija između korisnika i poslužitelja. Frontend šalje HTTPS zahtjeve backendu koji obrađuje zahtjeve, pristupa bazi podataka za dohvaćanje ili pohranu podataka te vraća odgovore natrag frontendu.



Slika 5.2: Prikaz dijagrama razmještaja

6. Ispitivanje programskog rješenja

Ispitivanje komponenti

PersonServiceTests

1. TEST: Dodavanje najdražih sastojaka za korisnika koji ne postoji

1. Funkcionalnost koju testiramo:

Testira se kako sustav rukuje situacijom kada korisnik s određenim personId ne postoji.

2. Ispitni slučaj:

Ulazni podaci:

- Long personId = 1L
- List ingredientIds = List.of(1L, 2L)

Očekivani rezultat:

- Bacanje RuntimeException s porukom:
- "Person not found with id: 1"

Dobiveni rezultat:

- Prolaz ispitivanja (metoda baca očekivanu iznimku s ispravnom porukom)

3. Postupak provođenja ispitivanja:

Ovaj ispitni slučaj testira funkcionalnost dodavanja najdražih sastojaka korisniku koji ne postoji u bazi podataka. Testiranje se obavlja simulacijom metode `personRepository.findById(personId)` tako da vraća `Optional.empty()`, čime se predstavlja scenarij u kojem korisnik s ID-jem 1 nije pronađen u bazi podataka. Nakon toga pozvana je metoda `personService.addFavoriteIngredients(personId, ingredientIds)`. Kako bi se provjerilo ispravno rukovanje greškom, korišten je `assertThrows` kao potvrda da metoda baca iznimku `RuntimeException`. Zatim se provjerava poruka iznimke pomoću `assertEquals` kako bi se osiguralo da odgovara očekivanoj vrijednosti. Nakon usporedbe rezultata s očekivanjima, potvrđeno je da sustav ispravno detektira situaciju kada korisnik ne postoji i ispravno rukuje pogreškom.

```
@Test new *
void addFavoriteIngredients_PersonNotFound() {
    Long personId = 1L;
    List<Long> ingredientIds = List.of(1L, 2L);

    when(personRepository.findById(personId)).thenReturn(Optional.empty());

    Exception exception = assertThrows(RuntimeException.class,
        () -> personService.addFavoriteIngredients(personId, ingredientIds));

    assertEquals("expected: " + "Person not found with id: " + personId, exception.getMessage());
}
```

Slika 6.1: Kod ispitnog slučaja 1

2. TEST: Dodavanje najdražih sastojaka, ali taj sastojak ne postoji u bazi podataka

1. Funkcionalnost koju testiramo:

Testira se dodavanje najdražih sastojaka, pri čemu neki sastojci ne postoje u bazi podataka.

2. Ispitni slučaj:

Ulazni podaci:

- Long personId = 1L
- List ingredientIds = List.of(1L, 2L, 3L)

Očekivani rezultat:

- Bacanje RuntimeException s porukom:
- "Some ingredients were not found with the provided IDs"

Dobiveni rezultat:

- Prolaz ispitivanja (metoda baca očekivanu iznimku s ispravnom porukom)

3. Postupak provođenja ispitivanja:

Ovaj ispitni slučaj testira funkcionalnost dodavanja najdražih sastojaka, pri čemu neki od sastojaka iz ulaznih podataka ne postoje u bazi podataka. U ovom testu simulirano je da metoda `personRepository.findById(personId)` vraća objekt korisnika s ID-jem 1, dok metoda `ingredientRepository.findAllById(ingredientIds)` vraća samo dva od tri sastojka koji su traženi (nedostaje sastojak s ID-jem 3). Nakon toga pozvana je metoda `personService.addFavoriteIngredients(personId, ingredientIds)`. Kako bi se provjerilo pravilno rukovanje greškom, korišten je `assertThrows` za potvrdu da metoda baca iznimku `RuntimeException`. Poruka iznimke se provjerava pomoću metode `assertEquals`, kako bi se osiguralo da odgovara očekivanoj vrijednosti: "Some ingredients were not found with the provided IDs". Rezultati su uspoređeni s očekivanima, te je potvrđeno da sustav ispravno prepoznaje situaciju kada neki sastojci nisu pronađeni u bazi te da ispravno rukuje takvom greškom.

```
@Test new *
void addFavoriteIngredients_IngredientNotFound() {
    Long personId = 1L;
    List<Long> ingredientIds = List.of(1L, 2L, 3L);

    Person mockPerson = new Person();
    mockPerson.setPersonId(personId);

    when(personRepository.findById(personId)).thenReturn(Optional.of(mockPerson));
    when(ingredientRepository.findAllById(ingredientIds)).thenReturn(List.of(new Ingredient(), new Ingredient()));

    Exception exception = assertThrows(RuntimeException.class,
        () -> personService.addFavoriteIngredients(personId, ingredientIds));

    assertEquals("expected: \"Some ingredients were not found with the provided IDs\", exception.getMessage());
}
```

Slika 6.2: Kod ispitnog slučaja 2

3. TEST: Dodavanje najdražeg sastojka, ali on je već najdraži tom korisniku

1. Funkcionalnost koju testiramo:

- Testira se dodavanje sastojka na popis najdražih sastojaka korisnika, pri čemu je taj sastojak već dodan kao najdraži korisniku.

2. Ispitni slučaj:

Ulazni podaci:

- Long personId = 1L
- List ingredientIds = List.of(1L)

Očekivani rezultat:

- Bacanje IllegalArgumentException s porukom:
- "Ingredient already added to favorites: Salt"

Dobiveni rezultat:

- Prolaz ispitivanja (metoda baca očekivanu iznimku s ispravnom porukom)

3. Postupak provođenja ispitivanja:

Ovaj ispitni slučaj testira funkcionalnost dodavanja najdražeg sastojka korisniku, pri čemu je taj sastojak već zabilježen u bazi podataka kao njegov najdraži. U testu je simulirano da metoda `personRepository.findById(personId)` vraća objekt korisnika s ID-jem 1, a metoda `ingredientRepository.findAllById(ingredientIds)` vraća sastojak s ID-jem 1 i nazivom "Salt". Nadalje, u testu je postavljeno da korisnik već ima taj sastojak ("Salt") na popisu svojih najdražih sastojaka (`mockPerson.getFavoriteIngredients().add(ingredient)`). Nakon toga pozvana je metoda `personService.addFavoriteIngredients(personId, ingredientIds)`. Kako bi se provjerilo pravilno rukovanje greškom, korišten je `assertThrows` kao potvrda da metoda baca iznimku `IllegalArgumentException`. Poruka iznimke provjerena je pomoću `assertEquals`, kako bi se osiguralo da odgovara očekivanoj vrijednosti: "Ingredient already added to favorites: Salt". Nakon što su rezultati uspoređeni s očekivanjima, potvrđeno je da sustav ispravno prepoznaje situaciju kada se pokušava dodati već postojeći najdraži sastojak te ispravno rukuje takvom greškom.


```

@Test new *
void addFavoriteIngredients_IngredientAlreadyAdded() {
    Long personId = 1L;
    List<Long> ingredientIds = List.of(1L);

    Person mockPerson = new Person();
    mockPerson.setPersonId(personId);

    Ingredient ingredient = new Ingredient();
    ingredient.setIngredientId(1L);
    ingredient.setName("Salt");

    mockPerson.getFavoriteIngredients().add(ingredient);

    when(personRepository.findById(personId)).thenReturn(Optional.of(mockPerson));
    when(ingredientRepository.findAllById(ingredientIds)).thenReturn(List.of(ingredient));

    Exception exception = assertThrows(IllegalArgumentException.class,
        () -> personService.addFavoriteIngredients(personId, ingredientIds));

    assertEquals("expected: 'Ingredient already added to favorites: Salt', exception.getMessage());
}

```

Slika 6.3: Kod ispitnog slučaja 3

4. TEST: Uspješno dodavanje najdražih sastojaka za korisnika

1. Funkcionalnost koju testiramo:

- Testira se uspješno dodavanje najdražih sastojaka.

2. Ispitni slučaj:

Ulazni podaci:

- Long personId = 1L
- List ingredientIds = List.of(1L, 2L)

Očekivani rezultat:

- Uspješno dodavanje sastojaka "Salt" i "Sugar" na popis najdražih sastojaka korisnika.

Dobiveni rezultat:

- Prolaz ispitivanja (sastojci "Salt" i "Sugar" uspješno dodani na popis korisnikovih najdražih sastojaka)

3. Postupak provođenja ispitivanja:

Ovaj ispitni slučaj testira uspješno dodavanje najdražih sastojaka korisniku. U testu je simulirano da metoda `personRepository.findById(personId)` vraća objekt korisnika s ID-jem 1, a metoda `ingredientRepository.findAllById(ingredientIds)` vraća dva sastojka: "Salt" s ID-jem 1 i "Sugar" s ID-jem 2. Nakon što se simulira dohvat korisnika i sastojaka, metoda `personRepository.save(mockPerson)` je također simulirana kako bi vratila spremljenog korisnika. Nakon toga pozvana je metoda `personService.addFavoriteIngredients(personId, ingredientIds)`. U završnom koraku korišten je `assertTrue` za provjeru da popis korisnikovih

najdražih sastojaka sadrži oba sastojka: "Salt" i "Sugar". Sustav ispravno dodaje nove sastojke na popis najdražih sastojaka.

```
@Test new *
void addFavoriteIngredients_SuccessfulAddition() {
    Long personId = 1L;
    List<Long> ingredientIds = List.of(1L, 2L);

    Person mockPerson = new Person();
    mockPerson.setPersonId(personId);

    Ingredient ingredient1 = new Ingredient();
    ingredient1.setIngredientId(1L);
    ingredient1.setName("Salt");

    Ingredient ingredient2 = new Ingredient();
    ingredient2.setIngredientId(2L);
    ingredient2.setName("Sugar");

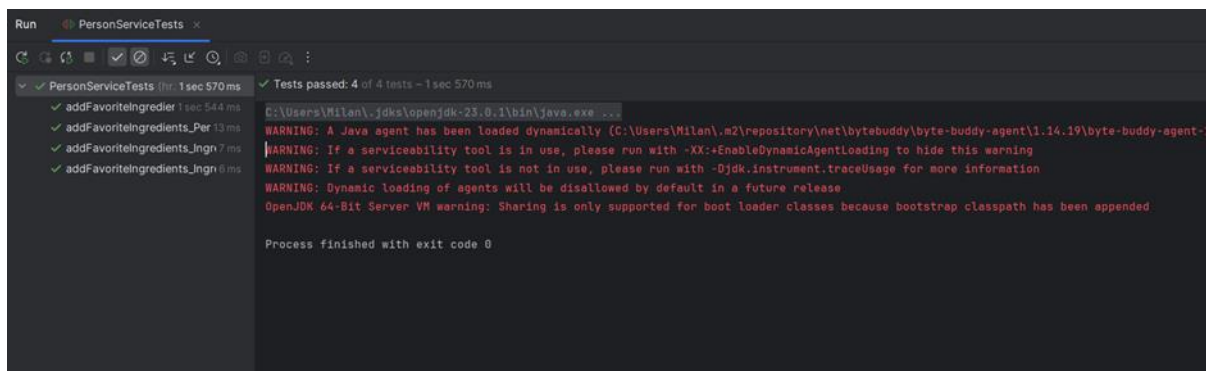
    when(personRepository.findById(personId)).thenReturn(Optional.of(mockPerson));
    when(ingredientRepository.findAllById(ingredientIds)).thenReturn(List.of(ingredient1, ingredient2));
    when(personRepository.save(mockPerson)).thenReturn(mockPerson);

    personService.addFavoriteIngredients(personId, ingredientIds);

    assertTrue(mockPerson.getFavoriteIngredients().contains(ingredient1));
    assertTrue(mockPerson.getFavoriteIngredients().contains(ingredient2));
}
```

Slika 6.4: Kod ispitnog slučaja 4

Na **slici 6.5** je prikaz rezultata izvršenih testova klase `PersonServiceTests`. Sva četiri testna slučaja su uspješno prošla, što je označeno zelenim oznakama pored naziva testova. Proces je završio s exit code 0, što označava uspješno izvršavanje testova.



Slika 6.5: Rezultati izvršenih testova

RecipeServiceTests

5. TEST: Testiranje graničnih vrijednosti za `recipeService.findSimilarRecipes()`

1. Funkcionalnost koju testiramo:

Testira se ispravno filtriranje recepata koji se podudaraju u 80% ili više sastojaka.

2. Ispitni slučaj:

Ulazni podaci:

- Long recipeld = 1L
- Recept "mainRecipe" s ID-jem 1 i sastojcima: "Salt", "Sugar", "Flour", "Butter", "Milk".
- Slični recepti:
 - "similarRecipe1" s 80% podudaranja,
 - "similarRecipe2" s 100% podudaranja,
 - "similarRecipe3" s 40% podudaranja (manje od 80%).

Očekivani rezultat:

- Funkcija vraća listu od 2 recepta: "similarRecipe1" i "similarRecipe2" jer se podudaraju u 80% ili više sastojaka s glavnim receptom.

Dobiveni rezultat:

- Prolaz ispitivanja (metoda ispravno vraća listu od 2 recepta koji zadovoljavaju uvjet sličnosti)

3. Postupak provođenja ispitivanja:

Ovaj ispitni slučaj testira metodu `recipeService.findSimilarRecipes()` za pronalazak sličnih recepata na temelju sličnosti sastojaka. U testu je simulirano da metoda `recipeRepository.existsById(recipeld)` potvrđuje postojanje glavnog recepta s ID-jem 1. Također je simulirano da metoda `recipeRepository.findSimilarRecipes(recipeld)` vraća listu recepata: "Similar Recipe 1", "Similar Recipe 2", i "Similar Recipe 3". Nakon toga pozvana je metoda `recipeService.findSimilarRecipes(recipeld)`. Na kraju, korišteni su `assertNotNull` za potvrdu da rezultat nije prazan te `assertEquals` za provjeru veličine liste rezultata (očekivano 2). Potvrđeno je da sustav ispravno prepozna recepte s dovoljnom sličnosti (80% ili više) i vraća ispravnu listu rezultata.

```

@Test
void testFindSimilarRecipes() {
    Long recipeId = 1L;
    Ingredient ingredient1 = createIngredient(ingredientId: 1L, name: "Salt");
    Ingredient ingredient2 = createIngredient(ingredientId: 2L, name: "Sugar");
    Ingredient ingredient3 = createIngredient(ingredientId: 3L, name: "Flour");
    Ingredient ingredient4 = createIngredient(ingredientId: 4L, name: "Butter");
    Ingredient ingredient5 = createIngredient(ingredientId: 5L, name: "Milk");

    Recipe mainRecipe = createRecipe(recipeId, title: "Main Recipe", Set.of(ingredient1, ingredient2, ingredient3,
        ingredient4, ingredient5));
    Recipe similarRecipe1 = createRecipe(recipeId: 2L, title: "Similar Recipe 1", Set.of(ingredient1, ingredient2,
        ingredient3, ingredient4));
    Recipe similarRecipe2 = createRecipe(recipeId: 3L, title: "Similar Recipe 2", Set.of(ingredient1, ingredient2,
        ingredient3, ingredient4, ingredient5));
    Recipe similarRecipe3 = createRecipe(recipeId: 2L, title: "Similar Recipe 3", Set.of(ingredient1, ingredient2));

    when(recipeRepository.existsById(recipeId)).thenReturn(true);
    when(recipeRepository.findSimilarRecipes(recipeId)).thenReturn(findSimilarRecipes(mainRecipe,
        List.of(similarRecipe1, similarRecipe2, similarRecipe3)));

    List<RecipeDTO> result = recipeService.findSimilarRecipes(recipeId);

    assertNotNull(result);
    assertEquals(expected: 2, result.size(), message: "80% or more similarity should return recipes.");
}

```

Slika 6.6: Kod ispitnog slučaja 5

6. TEST: Predaje se nevažeći recipeld

1. Funkcionalnost koju testiramo:

Testira se ispravno rukovanje, kada je unos nevažeći.

2. Ispitni slučaj:

Ulazni podaci:

- Long recipeld = null

Očekivani rezultat:

- Bacanje `ResponseStatusException` kako bi se označilo da je ulazni podatak nevažeći.

Dobiveni rezultat:

- Prolaz ispitivanja (metoda ispravno baca `ResponseStatusException` u slučaju nevažećeg ulaznog podatka)

3. Postupak provođenja ispitivanja:

Ovaj ispitni slučaj testira metodu `recipeService.findSimilarRecipes()` za scenarij kada je `recipeld` nevažeći. U testu je metoda direktno pozvana s `recipeld`, kojemu je dodijeljena vrijednost `null`. Kako bi se provjerilo pravilno rukovanje greškom, korišten je `assertThrows` za potvrdu da metoda baca iznimku `ResponseStatusException`. Sustav ispravno prepoznaje nevažeći unos i na ispravan način rukuje s pogreškom.

```
@Test new *
void testFindSimilarRecipes_InvalidRecipeId() {
    Long recipeId = null;
    assertThrows(ResponseStatusException.class, () -> recipeService.findSimilarRecipes(recipeId));
}
```

Slika 6.7: Kod ispitnog slučaja 6

7. TEST: Recept ne postoji

1. Funkcionalnost koju testiramo:

Testira se ispravno rukovanje kada traženi recept nije pronađen.

2. Ispitni slučaj:

Ulazni podaci:

- Long recipeId = 1L

Očekivani rezultat:

- Bacanje ResponseStatusException, kako bi se označilo da recept s dodijeljenim recipeId ne postoji.

Dobiveni rezultat:

- Prolaz ispitivanja (metoda ispravno baca ResponseStatusException u slučaju kada recept ne postoji)

3. Postupak provođenja ispitivanja:

Ovaj ispitni slučaj testira metodu recipeService.findSimilarRecipes() za scenarij kada je uneseni recipeId povezan s receptom koji ne postoji u bazi podataka. U testu je simulirano da metoda recipeRepository.existsById(recipeId) vraća false, što označava da recept s ID-jem 1 nije pronađen. Nakon toga pozvana je metoda recipeService.findSimilarRecipes(recipeId). Kako bi se provjerilo pravilno rukovanje greškom, korišten je assertThrows za potvrdu da metoda baca iznimku ResponseStatusException. Sustav ispravno prepoznaje nepostojanje recepta odgovarajuće rukuje greškom.

```
@Test new *
void testFindSimilarRecipes_NonExistingRecipe() {
    Long recipeId = 1L;
    when(recipeRepository.existsById(recipeId)).thenReturn(false);
    assertThrows(ResponseStatusException.class, () -> recipeService.findSimilarRecipes(recipeId));
}
```

Slika 6.8: Kod ispitnog slučaja 7

8. TEST: Brisanje recepta s recipeId = null

1. Funkcionalnost koju testiramo:

Testira se operacija brisanja u slučaju nevažećeg ulaznog podatka.

2. Ispitni slučaj:

Ulazni podaci:

- Long recipeld = null

Očekivani rezultat:

- Nije predviđeno što se događa kada se preda null, ali pretpostavka je da Spring framework sprječava pozivanje metode deleteById te da se brisanje ne izvršava.

Dobiveni rezultat:

- Prolaz ispitivanja (metoda deleteById iz repozitorija nije pozvana, čime je potvrđeno očekivano ponašanje)

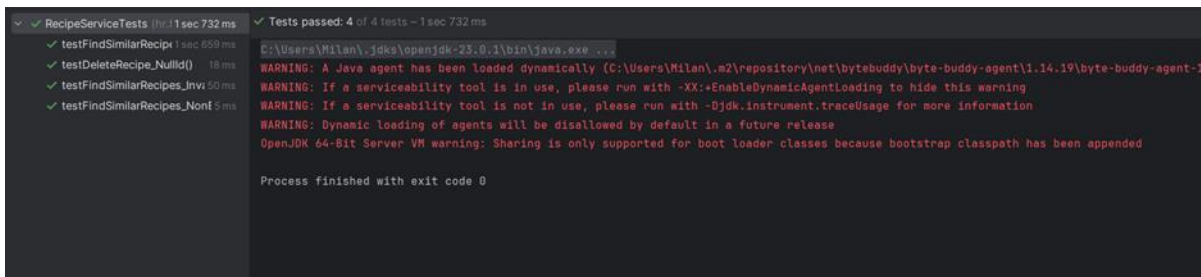
3. Postupak provođenja ispitivanja:

Ovaj ispitni slučaj testira metodu recipeService.deleteRecipe() za scenarij kada se preda null. U testu je metoda recipeService.deleteRecipe(null) pozvana s null vrijednošću. Nakon toga, korišten je verify za provjeru da se metoda deleteById iz recipeRepository nikada ne poziva (never()), čime se osigurava da operacija brisanja nije pokrenuta. Potvrđeno je da sustav ispravno rukuje unosom null tako da ne pokušava izvršiti operaciju brisanja.

```
@Test new *
void testDeleteRecipe_NullId() {
    recipeService.deleteRecipe(id: null);
    verify(recipeRepository, never()).deleteById(anyLong());
}
```

Slika 6.9: Kod ispitnog slučaja 8

Na **slici 6.10** je prikaz rezultata izvršenih testova klase RecipeServiceTests. Sva četiri testna slučaja su uspješno prošla, što je označeno zelenim oznakama pored naziva testova. Proces je završio s exit code 0, što označava uspješno izvršavanje testova.



Slika 6.10: Rezultati izvršenih testova

Ispitivanje sustava

Za ispitivanje sustava odlučili smo koristiti Selenium Web IDE (proširenje na web pregledniku). Razvili smo 4 testna slučaja: uspješna prijava, neuspješna prijava, dodavanje komentara na recept i dodavanje najdražih sastojaka.

1. Uspješna prijava

- **Opis:** Korisnik se prijavljuje s Google korisničkim imenom i lozinkom za prijavu. U testu se koristi testni korisnik test.recipaispeci.

- **Koraci:**

Nakon otvaranja prozora stišće se na gumb za pristup profilu.

Tada se otvori Googleov OAuth 2.0 login stranica na kojoj se unosi korisničko ime i lozinka.

Nakon uspješne prijave, provjerava se je li na stranici prisutan gumb za dodavanje recepata.

Ukoliko je prisutan, to sigurno znači da je korisnik prijavljen.

- **Očekivani izlaz:** test završen uspješno

	Command	Target	Value
1	✓ open	/	
2	✓ set window size	1552x832	
3	✓ click	css=.cursor-pointer > path:nth-child(2)	
4	✓ pause		30000
5	✓ type	id=identifierId	test.recipaispeci
6	✓ send keys	id=identifierId	\${KEY_ENTER}
7	✓ wait for element visible	name=Passwd	30000
8	✓ type	name=Passwd	TestReciPalspeci123
9	✓ send keys	name=Passwd	\${KEY_ENTER}
10	✓ wait for element visible	css=.font-2rem	30000
11	✓ assert element present	css=.font-2rem	
12	✓ close		

Slika 1: Koraci izvršeni u Seleniumu

1. open on / OK
2. setWindowSize on 1552x832 OK
3. click on css=.cursor-pointer > path:nth-child(2) OK
4. pause with value 30000 OK
5. type on id=identifierId with value test.recipaispeci OK
6. sendKeys on id=identifierId with value \${KEY_ENTER} OK
7. waitForElementVisible on name=Passwd with value 30000 OK
8. type on name=Passwd with value TestReciPalspeci123 OK
9. sendKeys on name=Passwd with value \${KEY_ENTER} OK
10. waitForElementVisible on css=.font-2rem with value 30000 OK
11. assertElementPresent on css=.font-2rem OK
12. close OK

'Uspjesan_login' completed successfully

Slika 2: Log uspješne prijave u Seleniumu

2. Neuspješna prijava

- **Opis:** Korisnik se pokušava prijaviti s Google korisničkim imenom i lozinkom za prijavu, ali unosi krivu lozinku. U testu se koristi testni korisnik test.recipaispeci.
- **Koraci:**

Nakon otvaranja prozora stišće se na gumb za pristup profilu.

Tada se otvori Googleov OAuth 2.0 login stranica na kojoj se unosi korisničko ime i lozinka.

Kako prijava nije uspješna, korisnik dobiva upozorenje u obliku teksta da je zaporka pogrešna.

Ukoliko dobijemo takav tekst, jasno je da koristimo krivu lozinku.

- **Očekivani izlaz:** Uspješan završetak testa, odnosno neuspješan login

	Command	Target	Value
1	✓ open	/	
2	✓ set window size	1552x832	
3	✓ click	css= cursor-pointer > path:nth-child(2)	
4	✓ pause		30000
5	✓ type	id=identifierId	test.recipaispeci
6	✓ send keys	id=identifierId	\${KEY_ENTER}
7	✓ wait for element visible	name=Passwd	30000
8	✓ type	name=Passwd	TestReciPaispeci122
9	✓ send keys	name=Passwd	\${KEY_ENTER}
10	✓ pause		30000
11	✓ assert text	css=div:nth-child(2) > span	Zaporka je pogrešna. Pokušajte ponovo ili kliknite "Zaboravili ste zaporku u?" da biste je poništili.
12	✓ close		

Slika 3: Tijek koraka u testu Neuspješan_Login

1. open on / OK
 2. setWindowSize on 1552x832 OK
 3. click on css=.cursor-pointer > path:nth-child(2) OK
 4. pause with value 30000 OK
 5. type on id=identifierId with value test.recipaispeci OK
 6. sendKeys on id=identifierId with value \${KEY_ENTER} OK
 7. waitForElementVisible on name=Passwd with value 30000 OK
 8. type on name=Passwd with value TestReciPaispeci122 OK
 9. sendKeys on name=Passwd with value \${KEY_ENTER} OK
 10. pause with value 30000 OK
 11. assertText on css=div:nth-child(2) > span with value Zaporka je pogrešna. Pokušajte ponovo ili kliknite "Zaboravili ste zaporku?" da biste je poništili. OK
 12. close OK
- 'Neuspjesan_login' completed successfully

Slika 4: Log neuspješne prijave u Seleniumu

3. Ostavljanje komentara

- **Opis:** Korisnik dodaje komentar i ocjenu na neki recept. U ovom slučaju tekst komentara glasi "Nije nešto", a korisnički račun koji smo koristili je milan.vidakovic04
- **Koraci:**

Korisnik prolazi sučeljem do pojedinog recepta.

Navedeni korisnik ostavlja recenziju s određenim brojem zvjezdica i komentarom.

Nakon dodavanja komentara, provjerava se je li ostavljen komentar kakav se pokušao dodati.

Provjerom se vidi da je milan.vidakovic04 uistinu ostavio komentar „Nije nešto“.

- **Očekivani izlaz:** Uspješan prolaz ispitivanja, odnosno ostavljen komentar

	Command	Target	Value
1	✓ open	/	
2	✓ set window size	1552x832	
3	✓ mouse down at	css=.col-xl-3:nth-child(1) .hover-text	168.02499771118164,67.39999389648438
4	✓ mouse move at	css=.col-xl-3:nth-child(1) .hover-text	168.02499771118164,67.39999389648438
5	✓ mouse up at	css=.col-xl-3:nth-child(1) .hover-text	168.02499771118164,67.39999389648438
6	✓ click	css=.col-xl-3:nth-child(1) .hover-text	
7	✓ click	css=.btn	
8	✓ click	css=.form-select	
9	✓ select	css=.form-select	label=★★★
10	✓ click	css=.form-control	
11	✓ type	css=.form-control	Nije nešto
12	✓ click	css=.btn-primary	
13	✓ assert text	css=.border-bottom:nth-child(5) > .text-muted	Nije nešto
14	✓ assert text	css=.border-bottom:nth-child(5) .fw-bold	milan.vidakovic04

Slika 5: Tijek koraka u testu Recept_komentar

1. open on / OK
2. setWindowSize on 1552x832 OK
3. Trying to find css=.col-xl-3:nth-child(1) .hover-text... OK
4. mouseMoveAt on css=.col-xl-3:nth-child(1) .hover-text with value 168.02499771118164,67.39999389648438 OK
5. mouseUpAt on css=.col-xl-3:nth-child(1) .hover-text with value 168.02499771118164,67.39999389648438 OK
6. click on css=.col-xl-3:nth-child(1) .hover-text OK
7. Trying to find css=.btn... OK
8. click on css=.form-select OK
9. select on css=.form-select with value label=★★★ OK
10. click on css=.form-control OK
11. type on css=.form-control with value Nije nešto OK
12. click on css=.btn-primary OK
11. type on css=.form-control with value Nije nešto OK
12. click on css=.btn-primary OK
13. Trying to find css=.border-bottom:nth-child(5) > .text-muted... OK
14. assertText on css=.border-bottom:nth-child(5) .fw-bold with value milan.vidakovic04 OK

'Recept_komentar' completed successfully

Slika 6: Selenium logovi uspješnog testiranja

4. Dodavanje najdražih sastojaka

- **Opis:** Testiramo opciju dodavanja najdražih sastojaka koja se nalazi na profilu korisnika. U ovom slučaju, korisniku test.recipaispeci dodali smo najdraže sastojke: borovnica, čokolada za kuhanje i kava
- **Koraci:**

Korisnik test.recipaispeci treba dodati najdraže sastojke.

Na početku testiranja se utvrđuje da korisnik još nema najdraže sastojke što se očituje time da nema recepata za prikaz na odjeljku „Za tebe“, a jednako tako i na stranici profila piše da još nema najdraže sastojke.

Nakon što korisnik doda najdraže sastojke „Borovnica“, „Čokolada za kuhanje“, „Kava“, utvrđuje se da su ti sastojci uistinu sada dodani na stranici profila.

Osim toga, sada je i recept prikazan na stranici „Za tebe“ čime se utvrđuje ispravna funkcionalnost.

- **Očekivani izlaz:** Uspješan prolaz testa - sastojci su dodani na profil korisnika

	Command	Target	Value
1	✓ open	/	
2	✓ set window size	1536x816	
3	✓ click	css=.mx-3:nth-child(2)	
4	✓ assert text	css=.align-content-center	Nema recepata za prikaz
5	✓ click	css=.rounded-circle	
6	✓ assert text	css=.col-xl-4 .card-text	@test.recipaispeci još nema najdraže sastojke!
7	✓ click	css=.col-xl-4 .ml-5	
8	✓ click	css=.ts-control	
9	✓ click	id=select-ing-opt-1	
10	✓ select	id=select-ing	label=Borovnica
11	✓ click	id=select-ing-ts-control	
12	✓ click	id=select-ing-opt-24	
13	✓ select	id=select-ing	label=Čokolada za kuhanje
14	✓ click	id=select-ing-ts-control	
15	✓ click	id=select-ing-opt-4	
16	✓ select	id=select-ing	label=Kava
17	✓ click	css=.btn-primary	
18	✓ assert text	css=.py-2:nth-child(1)	Borovnica
19	✓ assert text	css=.py-2:nth-child(2)	Čokolada za kuhanje
20	✓ assert text	css=.py-2:nth-child(3)	Kava
21	✓ click	css=.title-navbar	
22	✓ click	css=.mx-3:nth-child(2)	
23	✓ wait for element present	css=.col-xl-3:nth-child(1) .hover-text	30000
24	✓ assert element present	css=.col-xl-3:nth-child(1) .hover-text	

|Slika 7: Koraci izvršeni pri testiranju dodavanja najdražih sastojaka|

1. open on / OK
2. setWindowSize on 1536x816 OK
3. click on css=.mx-3:nth-child(2) OK
4. Trying to find css=.align-content-center... OK
5. click on css=.rounded-circle OK
6. assertText on css=.col-xl-4 .card-text with value @test.recipaispeci još nema najdraže sastojke! OK
7. click on css=.col-xl-4 .ml-5 OK
8. click on css=.ts-control OK
9. click on id=select-ing-opt-1 OK
10. select on id=select-ing with value label=Borovnica OK
11. click on id=select-ing-ts-control OK
12. click on id=select-ing-opt-24 OK
13. select on id=select-ing with value label=Čokolada za kuhanje OK
14. click on id=select-ing-ts-control OK
15. click on id=select-ing-opt-4 OK
16. select on id=select-ing with value label=Kava OK
17. click on css=.btn-primary OK
18. assertText on css=.py-2:nth-child(1) with value Borovnica OK
19. Trying to find css=.py-2:nth-child(2)... OK
20. assertText on css=.py-2:nth-child(3) with value Kava OK
21. click on css=.title-navbar OK
22. click on css=.mx-3:nth-child(2) OK
23. waitForElementPresent on css=.col-xl-3:nth-child(1) .hover-text with value 30000 OK
24. assertElementPresent on css=.col-xl-3:nth-child(1) .hover-text OK

'Dodavanje_najdrazih_recepata' completed successfully

|Slika 8: logovi uspješnog dodavanja sastojaka|

Korištene tehnologije i alati

1. Programski jezici:

- **JavaScript** – dinamički, višenamjenski programski jezik koji se koristi za razvoj web aplikacija. Omogućuje stvaranje interaktivnih i dinamičnih web stranica, uz kompletan razvoj frontenda i backenda.
- **Java 17** – jedan od najkorištenijih objektno orijentiranih programskih jezika u razvoju web aplikacija, podržava razne radne okvire te je široko poznata po svojoj pouzdanosti, sigurnosti i prenosivosti.

- **HTML 5** – standardni skriptni jezik za izradu web stranica. Omogućava web preglednicima interpretaciju i prikaz sadržaja korisnicima. Koristi se u kombinaciji s CSS-om i JavaScriptom.
- **CSS 4.15** – jezik korišten za određivanje i stilizaciju izgleda svih elemenata na web stranicama. U njemu programer odabire fontove, boje, poravnanja, veličine i druge vizualne aspekte stranice.

2. Radni okviri i biblioteke:

- **React 18.3.1** – JavaScript biblioteka za izgradnju interaktivnih korisničkih sučelja. Omogućuje stvaranje samostalnih i neovisnih komponenti koje se mogu višestruko koristiti i lako ih je ažurirati, što čini razvoj učinkovitijim.
- **Spring Boot** - specijalizirani dio Spring radnog okvira koji ubrzava i pojednostavljuje razvoj web aplikacija. Jedan je od najpopularnijih alata za izradu web aplikacija.

3. Baza podataka:

- Koristimo relacijsku bazu podataka, implementiranu pomoću **PostgreSQL-a 16**. PostgreSQL je alat za stvaranje i razvoj baze podataka, koji podržava složene upite, transakcije između baze i aplikacije. Besplatan je za korištenje te je stoga pogodan razvojnim timovima i studentima.
- **Liquibase** (Open source verzija) – alat otvorenog koda za jednostavno upravljanje bazom podataka. Alat pomoću datoteka changelogs omogućava praćanje svih promjena u bazi podataka te podržava direktno pisanje XML, JSON i SQL formatima. Podržava mnoge baze podataka, među kojima se nalazi i PostgreSQL. Omogućava unazadne promjene baze zvane rollback ako nešto pođe po krivu pri mijenjanju baze.

4. Razvojni alati:

- **Intelij IDEA v2024.3** – IDE koji ima odličnu podršku i razvojno okruženje za Javu.
- **Figma** – besplatan online alat za jednostavan i pristupačan dizajn web aplikacija i pojedinih stranica u aplikaciji. Korišten je za izradu okvirnog dizajna i izgleda stranice.
- **WebStorm 2024.3** – IDE specijaliziran za razvoja frontend dijela web aplikacija. Alat je idealan jer podržava frameworkove kao što su React, Angular i sl.

5. Alati za ispitivanje:

- **Selenium Web extension** – web proširenje za testiranje web-aplikacija, odnosno za simulaciju korisničkih interakcija s web-stranicom. Podržava mnoge preglednike te omogućava razvojnom timu detaljno i višestruko ispitivanje aplikacije.
- **JUnit 5** - radni okvir otvorenog koda koji se također koristi za testiranje web aplikacija pisanih u Javi. Pomoću ovog alata testirane su komponente aplikacije.

Za testiranje i verzioniranje koda korišteni su Git i Github.

- **Git (2.45.1.windows.1)** – lokalni alat za praćenje i kontrolu verzija projekta, omogućava grananje i spajanje (branching and merging) za rad na paralelnim komponentama projekta.
- **Github** – online platforma namijenjena za pohranu projekata koji koriste Git. Omogućava timovima učinkovitu suradnju na projektima i strukturiran pristup

podacima i kodu. Zajedno s Gitom omogućava pushanje (slanje) i pullanje (dohvaćanje) koda između lokalnog i udaljenog repozitorija.

6. Host:

- Aplikacija je hostana na online servisu **Render**, koji omogućava brz i lagan razvoj i deployment aplikacija uz podršku HTTPS-u i raznim programskim jezicima.

7. Komunikacija:

- Komunikacija se odvijala pomoću idućih aplikacija: Microsoft Teams, Discord, Whatsapp.

8. Dijagrami:

- Dijagrami obrazaca uporabe, sekvencijski dijagrami, dijagrami aktivnosti i stanja rađeni su pomoću alata **Visual Paradigm Online** (<https://online.visual-paradigm.com/>). Alat je osmišljen za jednostavno crtanje dijagrama online te smo ga koristili radi jednostavnosti. Za svaki tip dijagrama alat ima poseban skup elemenata za crtanje, što olakšava stvaranje svih potrebnih tipova dijagrama.
- Dijagrami baze podataka i razreda rađeni su pomoću online alata **draw.io** zbog fleksibilnosti alata za izradu dijagrama.

1. Instalacija

- **Preduvjeti**

Softver i verzije:

Node.js 16

Java 17 (za backend)

PostgreSQL 16 (za bazu podataka)

Liquibase (za upravljanje migracijama baze podataka)

Git (za preuzimanje repozitorija)

Razvojni alati:

IntelliJ IDEA

WebStorm

- **Preuzimanje: Upute za preuzimanje izvornog koda**

Primjer:

```

PS D:\repo> git clone https://github.com/alexMatika-code/Reci-pa-ispeci
Cloning into 'Reci-pa-ispeci'...
remote: Enumerating objects: 5119, done.
remote: Counting objects: 100% (177/177), done.
remote: Compressing objects: 100% (89/89), done.
remote: Total 5119 (delta 97), reused 99 (delta 78), pack-reused 4942 (from 2)
Receiving objects: 100% (5119/5119), 30.49 MiB | 701.00 KiB/s, done.
Resolving deltas: 100% (2351/2351), done.
PS D:\repo> cd .\Reci-pa-ispeci\
PS D:\repo\Reci-pa-ispeci>

```

Slika 8.1: Kloniranje Git repozitorija

Instalacija ovisnosti:

```

PS D:\GitHub\Reci-pa-ispeci\dev\teams-frontend\teams-frontend> npm i

removed 1875 packages, changed 18 packages, and audited 329 packages in 17s

111 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

```

Slika 8.2: Instaliranje ovisnosti

2. Postavke

- **Detaljne upute za konfiguraciju aplikacije:**

- Postavke su upravljane pomoću environment varijabli.
- Ključne varijable:

BACKEND_URL

DB_URL

DB_USERNAME

DB_PASSWORD

GOOGLE_CLIENT_SECRET

OAUTH_REDIRECT_URI

FRONTEND_URL

- Primjer .env datoteke:

U ovom projektu nisu korištene .env datoteke za pohranu konfiguracijskih podataka. Umjesto toga, svi potrebni parametri, poput podataka za povezivanje s bazom podataka ili API ključeva, unose se ručno direktno u Render platformu. Ove vrijednosti se definiraju kao Environment Variables unutar postavki Render servisa.

- Baza podataka

Za postavljanje baze podataka korišten je Render, gdje je kreirana PostgreSQL baza. Upravljanje strukturom baze i unos entiteta realizirani su pomoću Liquibase-a, koristeći changelog-master.xml datoteku. Ova datoteka služi kao glavna referenca za sve promjene u bazi, omogućavajući primjenu migracija i početnih podataka kroz verzionirane XML skripte. Promjene su automatski primijenjene prilikom pokretanja aplikacije, što osigurava konzistentnost baze u razvojnim i produkcijskim okruženjima.

3. Pokretanje aplikacije

- Upute za pokretanje aplikacije u različitim okruženjima:
- Razvojno okruženje:

```
PS D:\GitHub\Reci-pa-ispeci\dev\teams-frontend\teams-frontend> npm run dev

> teams-frontend@0.0.0 dev
> vite

VITE v5.4.10 ready in 267 ms

➔ Local:   http://localhost:3000/
➔ Network: use --host to expose
➔ press h + enter to show help
```

Slika 8.3: Pokretanje aplikacije u razvojnom okruženju

- Produkcijsko okruženje:

```
PS D:\GitHub\Reci-pa-ispeci\dev\teams-frontend\teams-frontend> npm run build

> teams-frontend@0.0.0 build
> vite build

vite v5.4.10 building for production...
[plugin:vite:resolve] [plugin:vite:resolve] Module "net" has been externalized for browser compatibility, imported by "D:\GitHub\Reci-pa-ispeci\dev\teams-frontend\teams-frontend\node_modules\stripe\lib\stripe-node.js". See https://vite.dev/guide/troubleshooting.html#module-externalized-for-browser-compatibility for more details.
- 572 modules transformed.
vite/index.html                8.72 kB | gzip:  0.48 kB
vite/assets/placeholder-50d4tq7.jpg 30.88 kB
vite/assets/logo-vu2d4ve.png      42.82 kB
vite/assets/vy-9-vvff48.jpg       2,120.83 kB
vite/assets/index-CMDI-B4z.css     268.45 kB | gzip:  30.18 kB
vite/assets/index-Bkx4tqf.js      421.22 kB | gzip: 137.71 kB
✓ build in 2.86s
```

Slika 8.4: Priprema aplikacije za produkcijsko okruženje

4. Upute za administratore

- Smjernice za administratore aplikacije nakon puštanja u pogon:
- Pristup administratorskom sučelju:
 - **Admin Panel:** <https://reci-pa-ispeci.onrender.com/user-control>
 - Za prijavu se koristi OAuth 2.0 autentifikacija putem Googlea. Administratori se moraju prijaviti s Google računom koji ima odgovarajuće dozvole za administraciju aplikacije. Početna stranica aplikacije dostupna je na: <https://reci-pa-ispeci.onrender.com/>.

- **Održavanje:**

Pregled logova:

Logovi aplikacije dostupni su putem Render sučelja, gdje se mogu pregledati za dijagnosticiranje grešaka i praćenje aktivnosti.

Ažuriranje aplikacije:

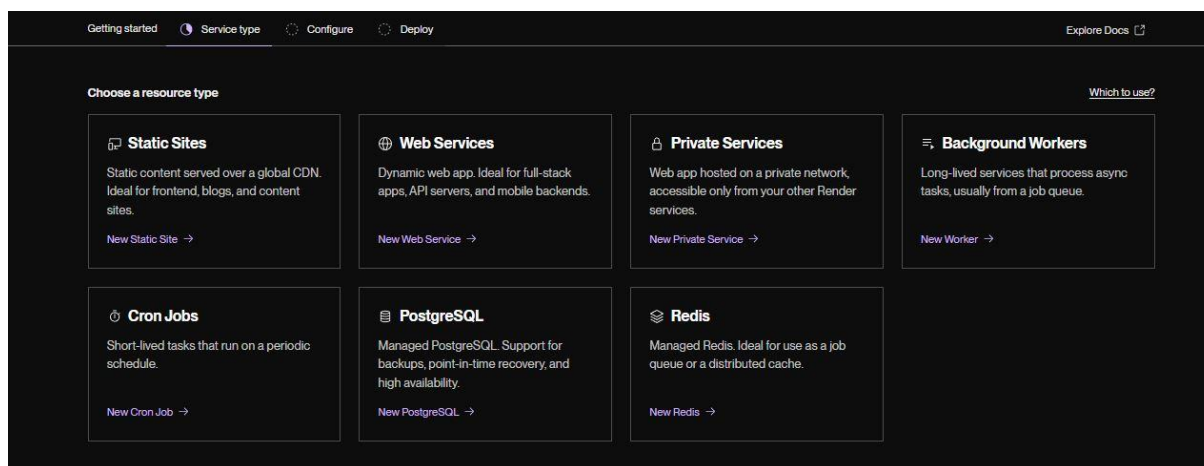
Aplikacija na Render platformi automatski se ponovno pokreće i ažurira svaki put kada se promjene iz Git repozitorija izvrše i budu commitane u glavnu granu (main).

Rješavanje problema:

Rješavanje problema na Renderu započinje pregledom logova putem Render dashboarda kako bi se identificirala greška i dobile detaljne informacije o njezinoj prirodi. Nakon što se problem prepozna, ažurira se ili klonira lokalna kopija koda, gdje se simulira situacija koja je uzrokovala grešku. Zatim ispravljamo probleme u kodu, čime se osigurava ispravnost funkcionalnosti lokalno. Kada je greška ispravljena, kod se detaljno testira kako bi se potvrdilo da problem više ne postoji. Zatim se promjene šalju u Git repozitorij, nakon čega Render automatski pokreće novi deployment s ažuriranim kodom. Na kraju se ponovno pregledavaju logovi na Renderu kako bi se potvrdilo da je problem uspješno riješen.

5. Primjer za Render platformu (Cloud Deploy)

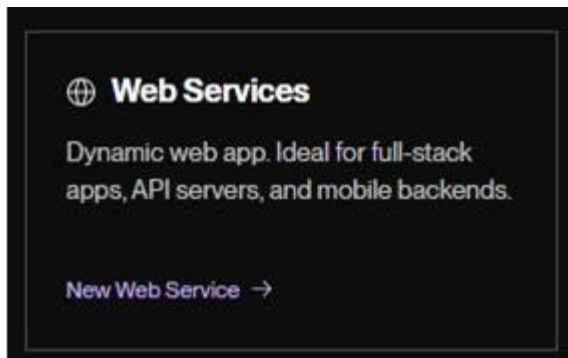
Za postavljanje web aplikacije korišten je Render, koji omogućuje jednostavno i učinkovito upravljanje infrastrukturom za frontend, backend i bazu podataka. Frontend aplikacija razvijena u Reactu postavljena je kao Web Services na Renderu. Backend aplikacija, izrađena pomoću Spring Boota, također je postavljena na Render kao Web Services, pri čemu se automatski pokreće prilikom deploymenta i povezuje s frontendom putem HTTP zahtjeva. Za bazu podataka korištena je PostgreSQL baza, koju Render pruža kao zasebnu uslugu. Baza je konfigurirana putem environment varijabli, čime je osigurana sigurna i jednostavna komunikacija s backend aplikacijom.



Slika 8.5: Odabir vrste resursa na Renderu

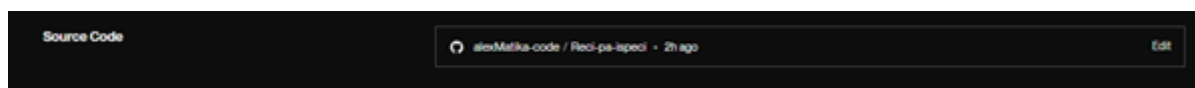
Postavljanje Frontenda na Render

Za pokretanje frontenda potrebno je odabrati opciju Web Services na Renderu.



Slika 8.6: Odabir Web Services kao vrste resursa

Zatim je potrebno povezati repozitorij svog frontenda sa Renderom.



Slika 8.7: Odabir repozitorija za povezivanje frontenda sa Renderom

Nakon odabira repozitorija potrebno je popuniti ponuđena polja. Zatim popunjavamo polja obaveznih podataka kao što su ime servisa, regiju, jezik, branch i root directory. U našem slučaju je odabrano ime Recipaispeci, regija Frankfurt (EU Central), jezik je postavljen na Docker, branch je postavljen na main i root directory je postavljen na dev/teams-frontend/teams-frontend.

Configure and deploy your new Web Service

Getting started
Service type
Configure
Deploy
Need help? Docs

Source Code
alexMatlika-code / Reci-pa-ispeci • 38m ago
Edit

Name
Reci-pa-ispeci
A unique name for your web service.

Project
Optional
Add this web service to a project once it's created.

Create a new project to add this to?

You don't have any projects in this workspace. Projects allow you to group resources into environments so you can better manage related resources.

+ Create a project

Language
Docker

Branch
main
The Git branch to build and deploy.

Region
Frankfurt (EU Central)
Your services in the same region can communicate over a private network.

Root Directory
Optional
dev/teams-frontend/teams-frontend
If set, Render runs commands from this directory instead of the repository root. Additionally, code changes outside of this directory do not trigger an auto-deploy. Most commonly used with a monorepo.

Slika 8.8: Popunjena polja u Renderu

Zatim je potrebno odabrati tip instance, našem slučaju je kod frontenda korišten besplatni tip instance.

Instance Type

For hobby projects

Free
\$0 / month
512 MB (RAM)
0.1 CPU

Upgrade to enable more features
Free instances spin down after periods of inactivity. They do not support SSH access, scaling, one-off jobs, or persistent disks. Select any paid instance type to enable these features.

For professional use
For more power and to get the most out of Render, we recommend using one of our paid instance types. All paid instances support:

- Zero Downtime
- SSH Access
- Scaling
- One-off jobs
- Support for persistent disks

Starter
\$7 / month
512 MB (RAM)
0.5 CPU

Standard
\$25 / month
2 GB (RAM)
1 CPU

Pro
\$85 / month
4 GB (RAM)
2 CPU

Pro Plus
\$175 / month
8 GB (RAM)
4 CPU

Pro Max
\$225 / month
16 GB (RAM)
4 CPU

Pro Ultra
\$450 / month
32 GB (RAM)
8 CPU















Need a custom instance type? We support up to 512 GB RAM and 64 CPUs.

Slika 8.9: Odabir tipa instance


Nakon toga smo postavili vrijednosti environment varijabli kao na slici 8.10.

Environment Variables

Set environment-specific config and secrets (such as API keys), then read those values from your code. [Learn more.](#)

BACKEND_URL	value	 Generate	
DB_URL	value	 Generate	
DB_USERNAME	value	 Generate	
DB_PASSWORD	value	 Generate	
GOOGLE_CLIENT_SECRET	value	 Generate	
OAuth_REDIRECT_URI	value	 Generate	
FRONTEND_URL	value	 Generate	

+ Add Environment Variable

 Add from .env

Slika 8.10: Postavljanje environment varijabli

Konačno, postavili smo potrebna polja u dijelu "Advanced" na sljedeći način. Dockerfile path je postavljen na dev/teams-frontend/teams-frontend/ ./Dockerfile i docker build context directory je unesen kao dev/teams-frontend/teams-frontend/ .

Nakon toga je moguće odabrati opciju Deploy Web Service i tako kreiramo web servis za frontend.

^ Advanced

Secret Files

Store plaintext files containing secret data (such as a `.env` file or a private key).

Access during builds and at runtime from your app's root, or from `/etc/secrets/<filename>`.

+ Add Secret File

Health Check Path

Provide an HTTP endpoint path that Render messages periodically to monitor your service. [Learn More](#).

Registry Credential

If your service pulls private Docker images from a registry, specify a credential that can access those images.

Manage your credentials in [Settings](#).

No credential

Docker Build Context Directory

The path to your service's Docker build context, relative to the repo root. Defaults to the root.

Dockerfile Path

The path to your service's Dockerfile, relative to the repo root. Defaults to `./Dockerfile`.

Docker Command

Optionally override your Dockerfile's `CMD` and `ENTRYPOINT` instructions with a different command to start your service.

Pre-Deploy Command

Render runs this command before the start command. Useful for database migrations and static asset uploads.

Auto-Deploy

By default, Render automatically deploys your service whenever you update its code or configuration. Disable to handle deploys manually. [Learn more](#).

Yes

Build Filters

Include or ignore specific paths in your repo when determining whether to trigger an auto-deploy. Paths are relative to your repo's root directory. [Learn more](#).

Included Paths

Changes that match these paths will trigger a new build.

+ Add Included Path

Ignored Paths

Changes that match these paths will not trigger a new build.

+ Add Ignored Path

Deploy Web Service

Slika 8.11: Popunjena polja u dijelu "Advanced"

Postavljanje Backenda na Render

Za pokretanje backenda potrebno je odabrati opciju Web Services na Renderu (*slika 8.6*).

Zatim je potrebno povezati repozitorij svog backenda sa Renderom (*slika 8.7*).

Nakon odabira repozitorija potrebno je popuniti ponuđena polja. Zatim popunjavamo polja obaveznih podataka kao što su ime servisa, regiju, jezik, branch i root directory. U našem slučaju je odabrano ime Recipa-ispeci, regija Frankfurt (EU Central), jezik je postavljen na Docker, branch je postavljen na main i root directory je postavljeno na dev/teams-backend/teams-backend.

Configure and deploy your new Web Service

Getting started **Service type** **Configure** Deploy Need help? Docs

Source Code

alexMatika-code / Recl-pa-ispeci • 38m ago Edit

Name
A unique name for your web service.

Recl-pa-ispeci

Project Optional
Add this web service to a project once it's created.

Create a new project to add this to?

You don't have any projects in this workspace. Projects allow you to group resources into environments so you can better manage related resources.

+ Create a project

Language

Docker

Branch
The Git branch to build and deploy.

main

Region
Your services in the same region can communicate over a private network.

Frankfurt (EU Central)

Root Directory Optional
If set, Render runs commands from this directory instead of the repository root. Additionally, code changes outside of this directory do not trigger an auto-deploy. Most commonly used with a monorepo.

dev/teams-backend/teams-backend

Slika 8.12: Popunjena polja u Renderu

Zatim je potrebno odabrati tip instance, našem slučaju je kod backenda korišten "Starter" tip instance.

Instance Type

For hobby projects

Free
\$0 / month
512 MB (RAM)
0.1 CPU

For professional use
For more power and to get the most out of Render, we recommend using one of our paid instance types. All paid instances support:

- Zero Downtime
- SSH Access
- Scaling
- One-off jobs
- Support for persistent disks

Starter \$7 / month 512 MB (RAM) 0.5 CPU	Standard \$25 / month 2 GB (RAM) 1 CPU
Pro \$85 / month 4 GB (RAM) 2 CPU	Pro Plus \$175 / month 8 GB (RAM) 4 CPU
Pro Max \$225 / month 16 GB (RAM) 4 CPU	Pro Ultra \$450 / month 32 GB (RAM) 8 CPU

Need a custom instance type? We support up to 512 GB RAM and 64 CPUs.

Slika 8.13: Odabir tipa instance

Nakon toga smo postavili vrijednosti environment varijabli kao na *slici 8.10*.

Konačno, postavili smo potrebna polja u dijelu "Advanced" na sljedeći način. Dockerfile path je postavljen na dev/teams-backend/teams-backend/ ./Dockerfile i docker build context directory je unesen kao dev/teams-backend/teams-backend .

Nakon toga je moguće odabrati opciju Deploy Web Service i tako kreiramo web servis za backend.

^ Advanced

Secret Files

Store plaintext files containing secret data (such as a `.env` file or a private key).

Access during builds and at runtime from your app's root, or from `/etc/secret/<filename>`.

Add Secret File

Disk

Mount an SSD to persist your service's filesystem data across deploys. This prevents zero-downtime deploys. [Learn More](#).

Disks are charged at \$0.25/GB per month.

Add disk

Health Check Path

Provide an HTTP endpoint path that Render messages periodically to monitor your service. [Learn More](#).

/healthz

Registry Credential

If your service pulls private Docker images from a registry, specify a credential that can access those images.

Manage your credentials in [Settings](#).

No credential

Docker Build Context Directory

The path to your service's Docker build context, relative to the repo root. Defaults to the root.

dev/teams-backend/teams-backend/ .

Dockerfile Path

The path to your service's Dockerfile, relative to the repo root. Defaults to `./Dockerfile`.

dev/teams-backend/teams-backend/ ./Dockerfile

Docker Command

Optionally override your Dockerfile's `CMD` and `ENTRYPOINT` instructions with a different command to start your service.

Pre-Deploy Command

Render runs this command before the start command.

Useful for database migrations and static asset uploads.

dev/teams-backend/teams-backend/ \$

Auto-Deploy

By default, Render automatically deploys your service whenever you update its code or configuration. Disable to handle deploys manually. [Learn more](#).

Yes

Build Filters

Include or ignore specific paths in your repo when determining whether to trigger an auto-deploy. Paths are relative to your repo's root directory. [Learn more](#).

Included Paths

Changes that match these paths will trigger a new build.

Add Included Path

Ignored Paths

Changes that match these paths will not trigger a new build.

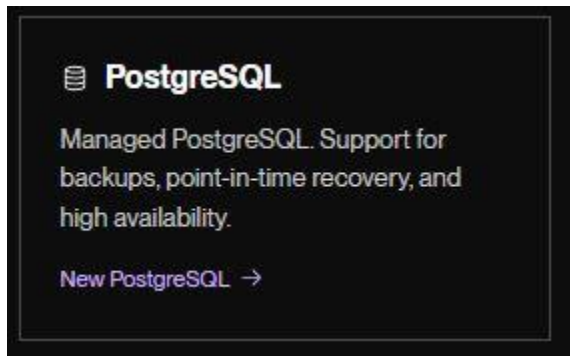
Add Ignored Path

Deploy Web Service

Slika 8.14: Popunjena polja u dijelu "Advanced"

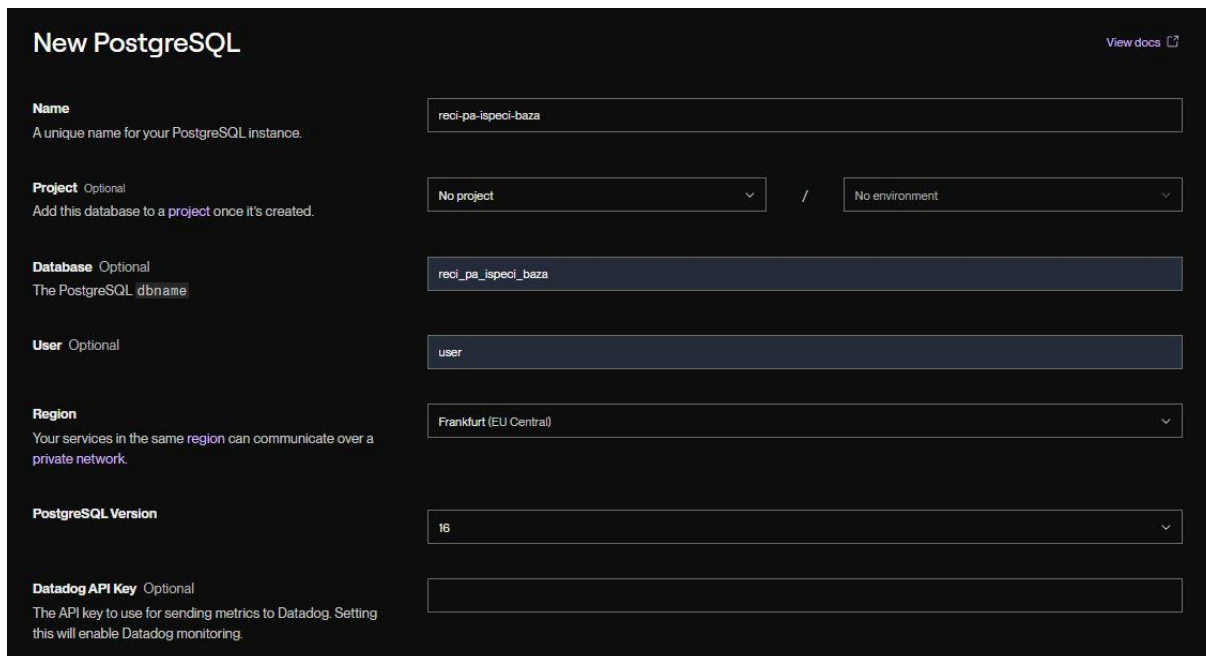
Postavljanje baze podataka na Render

Za pokretanje baze podataka potrebno je odabrati opciju PostgreSQL na Renderu.



Slika 8.15: Odabir PostgreSQL kao vrste resursa

Nakon toga je potrebno popuniti ponuđena polja koja traže podatke o imenu, regiji i verziji PostgreSQL.



New PostgreSQL [View docs](#)

Name
A unique name for your PostgreSQL instance.

Project Optional
Add this database to a project once it's created.
 /

Database Optional
The PostgreSQL dbname.

User Optional

Region
Your services in the same region can communicate over a private network.

PostgreSQL Version

Datadog API Key Optional
The API key to use for sending metrics to Datadog. Setting this will enable Datadog monitoring.

Slika 8.16: Popunjavanje polja za bazu podataka

Nakon toga odabrali smo "Basic plan" koji nudi 256 mb i 0.1 CPU, a kao kapacitet baze smo odabrali 1 gb.

Konačno, idemo na opciju kreiranja Baze podataka.

Plan Options

Instance Type

Set your database's RAM and CPU. You can change your instance type later.

New

You can now set your database's storage separately from its instance type [Learn more](#).

☐ **Free**
For testing out PostgreSQL on Render

☒ **Basic**
Reliability and performance for hobby projects. Starting at \$6 / month plus storage.

☐ **Pro**
Perfect for production use cases at any scale. Starting at \$55 / month plus storage.

☐ **Accelerated**
Memory-optimized plans offer significant performance improvements. Starting at \$160 / month plus storage.

Need a custom instance type? We support up to 1024 GB RAM, 128 CPUs, and 5 TB storage.

Memory and CPU

Basic-256mb \$6 / month	256 MB (RAM) 0.1 CPU	Basic-1gb \$19 / month	1 GB (RAM) 0.5 CPU
Basic-4gb \$75 / month		4 GB (RAM) 2 CPU	

Storage

Your database's capacity, in GB. You can increase storage at any time, but you can't decrease it. Specify 1 GB or any multiple of 5 GB.

Enable High Availability

Run a standby instance of your database and automatically fail over to it if the primary encounters an issue. [Learn more](#)

\$0.30 / month

☐ Disabled

ⓘ Only available for Pro instances and higher.

Monthly Total

Database instances are billed and prorated by the second.

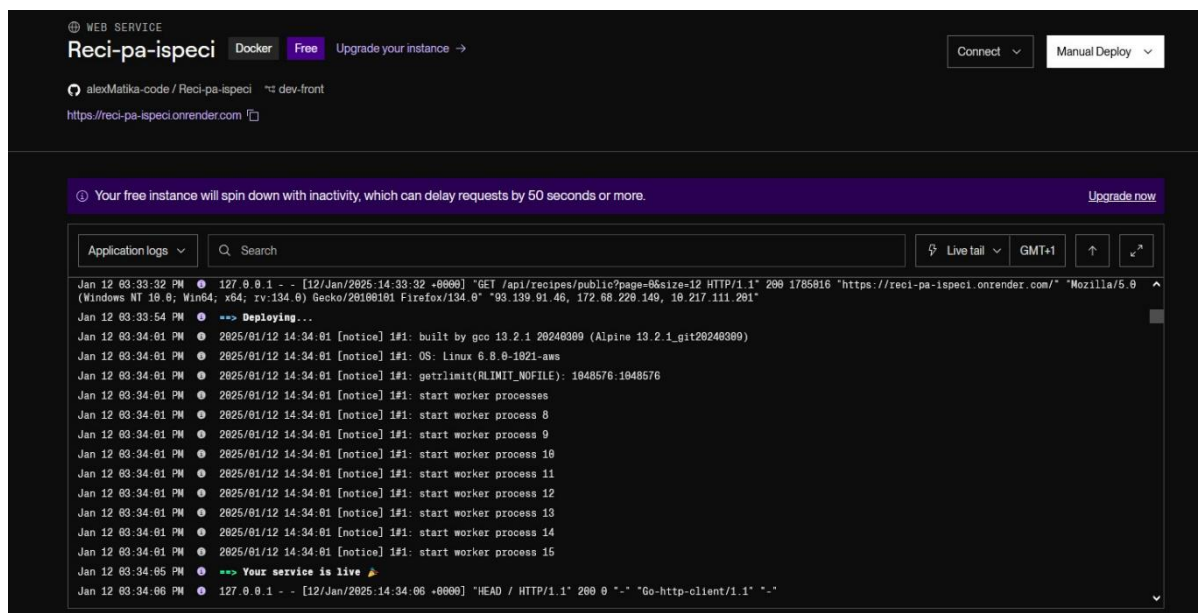
Instance: Basic-256mb	\$6 / month
Storage: 1 GB	\$0.30 / month
Total	\$6.30 / month

Create Database

Slika 8.17: Odabir plana i kapaciteta za bazu podataka

Kako bi se osigurala uspješna komunikacija između frontenda i backenda, bilo je nužno kreirati API koji će zaprimati zahtjeve i prosljeđivati ih backendu. Za pokretanje API-ja, prvo je bilo potrebno odabrati opciju za pokretanje web servisa. Taj servis se zatim povezuje s GitHub repozitorijem, odakle se preuzima sav potreban kod. Nakon toga, potrebno je unijeti ključne informacije, kao što su naziv servisa, regija, grana repozitorija s koje će se dohvaćati kod te, najvažnije, odabrati odgovarajući "Runtime" za web servis. U ovom slučaju, odabran je Node, dok smo za "Build Command" koristili npm build.

Dodatno, moguće je postaviti posebne environmental varijable koje omogućuju upravljanje verzijama instaliranog Nodea ili drugih "Runtimea". Kada koristimo Node, ključno je da se u repozitoriju nalazi datoteka *package.json*, koja Renderu omogućuje automatsko preuzimanje svih potrebnih biblioteka za pokretanje API-ja. Na kraju, nakon što se kod povuče iz repozitorija, pristupa se konzoli koja služi za upravljanje i kontrolu rada web servisa.



Slika 8.18: Pristup konzoli

Opis pristupa aplikaciji na javnom poslužitelju

- **Ograničenja:** Aplikacija Reci-pa-ispici ima određena ograničenja zbog resursa (CPU i RAM) na kojima je deployana. Ova konfiguracija prikladna je za manji broj korisnika, ali nije za visoki promet. Za veći broj korisnika potrebno je povećati dostupne resurse kako bi se osigurale bolje performanse i stabilnost aplikacije.
- **Korisnici** mogu pristupiti aplikaciji koristeći sljedeći URL: <https://reci-pa-ispici.onrender.com/>
- **Administratori** se moraju prijaviti s Google računom koji ima odgovarajuće dozvole za administraciju aplikacije. URL: <https://reci-pa-ispici.onrender.com/user-control>

Zaključak

- Zadatak ove grupe bio je napraviti stranicu za recepte koja može ići uz bok najboljim takvim stranicama na Balkanu i šire. Nakon 3 mjeseca rada i truda, nekoliko problema oko implementacije, testiranja i usuglašavanja u tome smo i uspjeli.
- U prvom ciklusu nastave smislili smo vlastitu ideju i sastavili popis osnovnih funkcionalnosti naše web aplikacije. Nakon prvog sastanka, podijelili smo se u tri grupe – Frontend, Backend i dokumentaciju. Svi timovi su u prvom ciklusu odradili svoj posao: dokumentacija je izradila potrebne dijagrame, funkcionalne zahtjeve i detaljne opise dijelova sustava, Backend je osigurao podizanje stranice, autorizaciju, bazu podataka i velik dio funkcionalnosti, dok je Frontend dio napravio moderan i intuitivan dizajn i izgled svih stranica aplikacije, razvoj korisničkog sučelja i integraciju s Backendom.
- U drugom ciklusu na ostalo je dovršiti ostale funkcionalnosti i dokumentaciju te ažurirati i nadograditi izgled stranica u skladu s novo dodanim funkcionalnostima. Uz veoma dobru komunikaciju unutar tima, puno rada i česte dogovore uspješno smo ostvarili zadani cilj.
- Prilikom razvoja aplikacije naišli smo na razne probleme u svim područjima razvoja projekta. Tim je imao problema s uspostavljanjem uspješne autorizacije putem

Google OAuth 2.0 servisa pri deployu te nekih problema pri samom deployu. Manjak iskustva rada u timu također je povećao broj sati utrošenih u razvoj aplikacije, konkretno zbog proučavanja kompliciranih alata i materijala s kojima se nismo nikad susreli. Svi tehnički problemi uspješno su riješeni dodatnim angažmanom članova tima i podrškom asistenata projekta.

- Svi članovi tima ovim projektom upoznali su se s raznim alatima i načinima rada s kojima se do sada nisu susreli, ali još bitnije od znanja rada u raznim alatima je iskustvo rada i komunikacije s razvojnim timom koje će svima biti korisna priprema za rad u struci.
- Za unaprjeđenje budućeg rada tima i brži razvoj projekta najvažniji dio je uspješna komunikacija između svih članova tima, kako bi se smanjila konfuzija i povećala učinkovitost kod implementacije promjena.
- Što se funkcionalnosti aplikacije tiče, uspješno smo savladali sve funkcionalnosti koje smo zamislili na početku rada, od sustava recepata, chatova, hijerarhije korisnika i ostaloga, osim NF-2.1 (stranicu može koristiti do 100 korisnika bez gubitka performansi), zato što nismo vidjeli smisao plaćanja skuplje verzije Rendera.

Rev.	Opis promjene/dodatka	Autori	Datum
0.1	Napravljen template i Github	grupno	23.10.2024.
0.2	Napravljen Github Wiki	Patrik Krčma	31.10.2024.
0.3	Napravljena specifikacija zahtjeva sustava	Patrik Krčma	07.11.2024.
0.4	Napravljena opis projektnog zadatka	Fran Čopčić	08.11.2024.
0.5	Napravljena dnevnik promjena dokumentacije	Fran Čopčić	08.11.2024.
0.6	Napravljena prikaz aktivnosti grupe	Fran Čopčić	08.11.2024.
0.7	Napravljena folder za fotografije na wikiju	Fran Čopčić	08.11.2024.
0.8	Napravljena analiza zahtjeva	Fran Čopčić	10.11.2024.
0.9	Napravljene promjene obrazaca uporabe	Patrik Krčma	10.11.2024.
1.0	Napravljeni dijagrami obrazaca uporabe	Patrik Krčma	10.11.2024.
1.1	Uređivanje wikija i dodavanje rješenja na wiki	Patrik Krčma	11.11.2024.

Rev.	Opis promjene/dodatka	Autori	Datum
1.2	Napravljeni i dodani sekvencijski dijagrami	Patrik Krčma	11.11.2024. i 12.11.2024.
1.3	Ažuriran izgled wiki-a po novom predlošku profesora	Patrik Krčma	24.12.2024.
1.4	Dodan dijagram stanja i opis, kratko ažuriranje 3. točke	Patrik Krčma	11.01.2025.
1.5	Dodan dijagram aktivnosti i njegov opis	Patrik Krčma	11.01.2025.
1.6	Dodani dijagrami komponenata i razmještaja te njihov opis	Fran Čopčić	14.01.2025.
1.7	Dodavanje opisa instalacije u uputama za puštanje u pogon	Fran Čopčić	14.01.2025.
1.8	Uređivanje funkcionalnih zahtjeva	Fran Čopčić	15.01.2025.
1.9	Ispravljanje sitnih pogrešaka kod dijagrama komponenata i popunjavanje uputa za puštanje aplikacije u pogon	Fran Čopčić	16.01.2025.
2.0	Dodavanje opisa slikama u uputama za puštanje u pogon	Fran Čopčić	17.01.2025.
2.1	Ažuriranje dnevnika sastajanja	Fran Čopčić	17.01.2025.
2.2	Dodane "Korištene tehnologije i alati"	Patrik Krčma	18.01.2025.
2.3	Izmijenjene uključenosti ključnih funkcionalnosti u obrascima uporabe	Patrik Krčma	18.01.2025.
2.4	Izmijenjeni detalji popisa korištenih alata i tehnologija	Patrik Krčma	20.01.2025.
2.5	Unos sadržaja pod točku "Ispitivanje komponenti"	Fran Čopčić	22.01.2025.
2.6	Dodavanje ključnih izazova i rješenja.	Fran Čopčić	22.01.2025.
2.7	Napravljeni testovi za ispitivanje komponenti i sustava	Milan Vidaković	22. i 24.01.2025.
2.8	Testovi za ispitivanje sustava dodani na wiki	Patrik Krčma	24.01.2025.

Rev. Opis promjene/dodatka	Autori	Datum
2.9 Formatiranje slika i sadržaja na Wikiju	Patrik Krčma	24.01.2025.
3.0 Popunjavanje uputa za puštanje u pogon.	Fran Čopčić	24.01.2025.
3.1 Dodavanje dijagrama pregleda promjena.	Fran Čopčić	24.01.2025.

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.Langanieri, "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/](http://www.ece.rutgers.edu/~marsic/books/SE) books/SE
5. The Unified Modeling Language, <https://www.uml-diagrams.org/>
6. Visual Paradigm online, <https://www.visual-paradigm.com/>
7. [draw.io] (<https://app.diagrams.net/>) alat za crtanje dijagrama
8. YouTube (<https://www.youtube.com/>)
9. Wiki profesora Sruka <https://github.com/VladoSruk/Programsko-inzenjerstvo/wiki>

Dnevnik sastajanja

1. Sastanak

- **Datum:** 9.10.2024.
- **Prisustvovali:** Matija Lovreković, Alex Matika, Franko Ćosić, Marta Leš, Patrik Krčma, Fran Čopčić, Milan Vidaković
- **Teme sastanka:**
 - međusobno upoznavanje
 - osmišljavanje teme za projektni zadatak
 - uspostavljena WhatsApp grupa za daljnju komunikaciju između članova grupe

2. Sastanak

- **Datum:** 15.10.2024.
- **Prisustvovali:** Matija Lovreković, Alex Matika, Franko Ćosić, Marta Leš, Patrik Krčma, Fran Čopčić, Milan Vidaković
- **Teme sastanka:**
 - sastanak s asistentom i demosom – prodiskutirati osmišljenu temu i što treba nadodati ili izmijeniti

3. Sastanak

- **Datum:** 16.10.2024.
- **Prisustvovali:** Matija Lovreković, Alex Matika, Franko Ćosić, Marta Leš, Patrik Krčma, Fran Čopčić, Milan Vidaković
- **Teme sastanka:**
 - podjela zadataka na projektu
 - rasprava o alatima koji će biti korišteni

4. Sastanak

- **Datum:** 5.11.2024.
- **Prisustvovali:** Matija Lovreković, Alex Matika, Franko Ćosić, Marta Leš, Patrik Krčma, Fran Čopčić, Milan Vidaković
- **Teme sastanka:**
 - dogovor oko konačnih izmjena na projektu, prije prve predaje
 - raspodjela zadataka koji trebaju biti obavljeni do prve predaje

5. Sastanak

- **Datum:** 10.11.2024.
- **Prisustvovali:** Matija Lovreković, Alex Matika, Franko Ćosić, Marta Leš, Patrik Krčma, Fran Čopčić, Milan Vidaković
- **Teme sastanka:**
 - dodavanje završnih funkcionalnosti prve predaje
 - dogovor o prezentiranju prve predaje

6. Sastanak

- **Datum:** 17.12.2024.
- **Prisustvovali:** Matija Lovreković, Alex Matika, Franko Ćosić, Marta Leš, Patrik Krčma, Fran Čopčić, Milan Vidaković
- **Teme sastanka:**
 - podjela zadataka na projektu u drugom ciklusu

7. Sastanak

- **Datum:** 13.01.2025.
- **Prisustvovali:** Matija Lovreković, Alex Matika, Franko Ćosić, Marta Leš, Patrik Krčma, Fran Čopčić, Milan Vidaković
- **Teme sastanka:**
 - predavljanje dosadašnjega doprinosa članova tima
 - analiza i planiranje preostalih zadataka do završetka projekta

8. Sastanak

- **Datum:** 23.01.2025.
- **Prisustvovali:** Matija Lovreković, Alex Matika, Franko Ćosić, Marta Leš, Patrik Krčma, Fran Čopčić, Milan Vidaković
- **Teme sastanka:**
 - konačni popravci aplikacije
 - dogovor u vezi prezentiranja projekta 28.1.

Tablica aktivnosti

Zadatak	Matija Lovreković	Alex Matika	Franko Ćosić	Marta Leš	Patrik Krčma	Fran Čopčić	Milan Vidaković
Upravljanje projektom	23						
Opis projektnog zadatka		1				6	
Funkcionalni zahtjevi		4				3	
Ostali zahtjevi		2				1	
Dizajn aplikacije	5			10	2	6	
Opis pojedinih obrazaca					8		
Dijagram obrazaca					6		
Sekvencijski dijagrami					5		
Dionici						2	
Arhitektura i dizajn sustava		2	1	3			3
Baza podataka		1		3			
Dijagram razreda				14			
Dijagram stanja					7		
Dijagram aktivnosti					5		
Dijagram komponenti						9	

Dijagram razmještaja						4	
Popis korištenih tehnologija i aplikacija				4			
Upute za puštanje u pogon						6	
Zaključak				2			
Izrada prezentacije				2			
Dnevnik sastajanja						2	
Popis literature						1	
Sastanci	12	12	12	12	12	12	12
Istraživanje informacija i tehnologija	5	10	3	8	9	8	10
Deployment	8	25	5				
Izrada baze podataka			4				
Spajanje s bazom podataka			3				
Ispitivanje komponenti i sustava		2			1	1	5
Backend	2	10	58	25			25
Frontend	65	20	1	40			

Dijagram pregleda promjena



Ključni izazovi i rješenja

Opis izazova

1. Tehnički problemi:

- Implementacija autorizacije putem Google OAuth 2.0 servisa prilikom deploya
- Problemi s deployanjem same aplikacije

2. Manjak iskustva:

Članovi tima nisu imali dovoljno iskustva u radu s alatima i tehnologijama korištenima u projektu Nedostatak iskustva u timskom radu

Rješenja

Rješavanje tehničkih problema:

- Problemi s autorizacijom i deployem riješeni su dodatnim angažmanom članova tima i konzultacijama s projektnim asistentima

Poboljšanje timske suradnje:

- Redoviti sastanci i otvorena komunikacija unutar tima dovelo je do bolje koordinacije i smanjenja nesporazuma
- Kroz suradnju članovi su naučili koristiti nove alate i metode, što je rezultiralo poboljšanjem timske dinamike

Zaključno

Unatoč izazovima, projekt je uspješno završen u zadanom vremenskom okviru. Kroz projekt su ostvarene sve planirane funkcionalnosti, uključujući sustav recepata, hijerarhiju korisnika i integraciju chat sustava. Tim je razvio vrijedne vještine u timskoj suradnji, rješavanju problema i radu s naprednim tehnologijama. Dobrodošli u Reci-pa-ispeci wiki!