

Dobrodošli na Book-n-Bite wiki!

Programsko inženjerstvo ak.god 2024./2025

Sveučilište u Zagrebu

Fakultet elektrotehnike i računarstva

Book n' Bite

Tim: <TG10.2>

Ime tima: Book n' Bite

Nastavnik: Vlado Sruk

Asistent: Goran Rajić

Cilj projekta

Cilj ovog projekta je razviti programsku podršku za stvaranje web aplikacije „Book 'n Bite“ koja će grupama korisnika olakšati izbor restorana. Zamislite scenarij u kojem se nalazite u grupi prijatelja i svi želite nešto pojesti, ali se ne možete dogovoriti u koji restoran želite otići ili iz kojeg restorana želite nešto naručiti. Glavna zadaća ove aplikacije je olakšati vam izbor te na temelju osobnih preferencija svih članova, odrediti koji restoran najviše odgovara grupi u cijelini. Svaki korisnik - ocjenjivač u aplikaciji dobiva listu proizvoda iz raznih restorana te svaki od ponuđenih proizvoda ocjenjuje zvjezdicama od 1 do 5, ovisno o tome koliko mu se to jelo ili piće sviđa i koliko ga u tom trenutku želi konzumirati. Kada svi članovi grupe prođu ovaj postupak, aplikacija na temelju ocjena određuje koji restoran najviše odgovara grupi odnosno ispisuje cijelu rang listu restorana.

Opis i opseg projekta

Aplikacija podržava stvaranje i prijavu na više vrsta profila: * ocjenjivač, * restoran, * administrator.

Ovisno o vrsti profila na koji korisnik izvrši prijavu, unutar aplikacije će mu biti dostupne različite funkcionalnosti. Svim korisnicima se nakon ulaska u aplikaciju prikazuje početni zaslon s gumbom "Nastavi koristeći Google" pomoću kojeg će biti preusmjereni na Googlov OAuth2 te poslije toga na registraciju gdje će nadopuniti podatke o sebi i odabratи jesu li restoran ili ocjenjivač. Stari korisnici će poslije autentifikacije odmah biti preusmjereni na svoju glavnu stranicu. Sustav vrši provjeru unesenih podataka te ukoliko je to potrebno prikazuje korisniku poruku o neispravnosti unesenih podataka i dozvoljava mu da ih prepravi. Poruka o pogrešci se prikazuje u slučaju da korisnik ne unese sve navedene podatke ili unese određene podatke u pogrešnom formatu. Ukoliko su svi podaci ispravno uneseni postupak registracije se uspješno provodi

te se ocjenjivača preusmjeruje na izbornik za grupe, dok se profile restorana preusmjeruje na stranicu za unos dodatnih podatka o restoranu.

Na stranici za unos podataka o restoranu potrebno je unijeti naziv restorana, radno vrijeme te lokaciju. Ovo su podaci koje je obvezno unijeti te u suprotnom sustav ispisuje poruku o pogrešci. Također postoje i opcionalni podaci koje restoran može unijeti u koje spadaju broj telefona te proizvoljne poveznice koje će se prikazivati na stranici restorana kao što su poveznica na vanjsku stranicu restorana ili poveznice na aplikacije za dostavu. Uspješnim unosom ovih podataka restoran ulazi u postupak verifikacije te se restoranu prikazuje zaslon čekanja na verifikaciju i onemogućuje daljnje korištenje aplikacije dok administrator ne verificira njegov profil.

Računi za prijavu administratora su unaprijed određeni i dostupni nadležnim osobama. Prijava korisnika se vrši neovisno o njihovoј podjeli tj. i ocjenjivač i restoran prolaze samo Googlovu autentifikaciju te su preusmjereni na svoje glavne stranice.

Nakon prijave ocjenjivača prikazuje im se stranica koja se sastoji od polja za unos koda grupe kojoj se žele pridružiti, gumba za stvaranje nove grupe te ikone koja im omogućava pregled vlastitog profila. Pritiskom na gumb za pregled vlastitog profila korisniku se prikazuje stranica sa njegovim informacijama koje je unio pri postupku registracije.

Na početnoj stranici unosom koda grupe ocjenjivači se mogu direktno pridružiti postojećoj grupi te tada direktno odlaze na stranicu za ocjenjivanje proizvoda. U slučaju unosa koda koji ne odgovara ni jednoj grupi ocjenjivaču se ispisuje pogreška.

Ocenjivač koji želi stvoriti novu grupu može to učiniti pritiskom na gumb „Stvor novu grupu“ koji će ga odvesti na sljedeću stranicu. Ova stranica sadrži izbornik kategorije proizvoda te sučelje za odabir lokacije. Ocjenjivač koji stvara grupu jedini ima pravo izabrati ove podatke te će se oni nadalje odnositi na grupu u cijelini. Nakon što odabere navedene podatke ocjenjivač se prebacuje na stranicu za ocjenjivanje jela te od tog trenutka stvorena grupa postaje aktivna. Pri vrhu stranice za ocjenjivanje jela kreatoru grupe ali i ostalim članovima koji se pridruže grupi ispisuje se generirani kod grupe. Ovaj kod grupe unose ostali korisnici koji se žele pridružiti stvorenoj grupi.

Nakon grupiranja, svim ocjenjivačima iste grupe prikazuje se ista lista nasumično odabralih proizvoda iz različitih restorana. Ispod svakog proizvoda nalazi se interaktivna skala u obliku 5 zvjezdica pomoću koje ocjenjivač može iskazati koliko je zainteresiran za ponuđeni proizvod. Nakon što je korisnik ocijenio sve ponuđene proizvode pritišće gumb „Završi“. Sustav u tom trenutku pregledava jesu li svi proizvodi ocjenjeni (ukoliko nisu šalje poruku o pogrešci) te zatim šalje prikupljene ocjene u bazu podataka. Nakon toga ocjenjivaču se prikazuje zaslon čekanja na ostale korisnike. U trenutku kada svi članovi grupe izvrše postupak ocjenjivanja jela, sustav izvršava potrebne kalkulacije te zatim svima prikazuje stranicu sa rang listom restorana. Ova rang lista je jedinstvena za

cijelu grupu te redoslijedom od najboljeg prema najgorem prikazuje restorane koji bi odgovarali grupi. Odabirom restorana s rang liste otvara se stranica tog restorana opisana u nastavku.

Nakon što administrator verificira određeni restoran on se tada može prijaviti na svoj račun. Nakon prijave restoranima se prikazuje vlastita stranica restorana na kojoj su vidljive sve informacije koje restoran unosi tijekom registracije te popis proizvoda navedenog restorana. Restoranu se također prikazuje ikona za uređivanje navedenih informacija. Ukoliko restoran pritisne na navedenu ikonu ponovo ga se odvodi na stranicu za ispunu informacija o restoranu kao i prilikom registracije samo što ovaj put izmjene ne treba verificirati administrator. Lokacija restorana prikazuje se na karti te klikom na nju korisnik može dobiti interaktivni prikaz lokacije. Uz popis proizvoda restoranima se također prikazuje i gumb za dodavanje novih proizvoda. Prilikom dodavanja novog proizvoda otvara se stranica za unos osnovnih informacija o proizvodu: naziv proizvoda, opis, cijena, tip, alergeni i kategorija proizvoda. Restoran također ima mogućnost prenijeti sliku proizvoda. Odabirom na opciju "Stvorи proizvod" novi proizvod se dodaje u bazu podataka te je vidljiv na stranici restorana. Izgled stranice restorana je jednak i kada je pregledavaju ocjenjivači izuzev ikona za izmjenu informacija i dodavanja proizvoda.

Posljednji od mogućih profila na koje se korisnik može prijaviti je administrator. Administratoru se nakon prijave prikazuje početni zaslon s tri opcije: „Pregled liste ocjenjivača“, „Pregled liste restorana“ te „Pregled liste za verifikaciju“. Ukoliko administrator odluči pregledati listu ocjenjivača prikazuje mu se lista kartica sa korisničkim imenima svih ocjenjivača koji su stvorili račun unutar aplikacije. Klikom na bilo koju od njih otvara se prethodno opisana stranica profila navedenog korisnika. Uz to na svakoj kartici administrator ima gume koji omogućavaju blokiranje ili trajno brisanje računa navedenih korisnika ukoliko se utvrde kršenja smjernica aplikacije. Ukoliko administrator odluči trajno obrisati neki od profila on se uklanja iz baze podataka, prijava ne njega više nije moguća te se uklanja s administratorove liste. Ukoliko administrator odluči blokirati određenog korisnika njegov račun se u bazi podataka označava kao neaktiviran te prijava tog korisnika više nije moguća. U administratorovom prikazu gumb za blokiranje se tada pretvara u gumb za odblokiranje pomoću kojeg administrator u bilo kojem trenutku ima mogućnost poništiti navedenu akciju blokiranja. Administratoru se odabirom na opciju „Pregled liste restorana“ nude analogne opcije opisane u „Pregled liste ocjenjivača“ samo što se na listi sada nalaze profili restorana. U pregledu liste za verifikaciju administrator može vidjeti popis kartica svih restorana koji trenutno čekaju na verifikaciju, klikom na njih pregledati njihove informacije te se zatim odlučiti želi li ih verificirati ili ne. Svoj odabir izražava pritiskom na jedan od dva gumba prikazana kraj kartice svakog restorana: „Potvrdi“ ili „Odbaci“.

Potencijalna korist ovog projekta

Projekt donosi višestruke koristi za sve dionike uključene u sustav, kao što su ocjenjivači, restorani, i administratori platforme. Specifične koristi uključuju:

- Za ocjenjivače: Aplikacija nudi jedinstvenu platformu za istraživanje restorana, dijeljenje iskustava i savjeta s prijateljima kroz grupnu funkcionalnost, te lakše donošenje odluka o odabiru restorana. Korisnici mogu pristupiti ocjenama i recenzijama restorana, što im pomaže u odlučivanju o najboljim opcijama za njihov planirani izlazak. Također, mogućnost stvaranja grupa za zajedničko ocjenjivanje omogućuje veću interaktivnost i socijalnu povezanost među korisnicima.
- Za restorane: Restorani imaju priliku poboljšati svoju vidljivost na tržištu. Platforma omogućava restoranu da prikazuje svoje proizvode, odgovara na recenzije korisnika i upravlja svojim ocjenama. Ovo može povećati njihovu reputaciju i privući nove kupce koji traže provjerene i dobro ocijenjene restorane.
- Za administratore: Administratori imaju potpunu kontrolu nad pristupom platformi, uključujući registraciju restorana, verifikaciju korisničkih računa i nadzor nad sadržajem. Ovaj nadzor osigurava da platforma ostane sigurna, povjerljiva i pouzdana za korisnike.

Postojeća slična rješenja

Na tržištu već postoje razna rješenja koja omogućuju korisnicima da pretražuju restorane, ocjenjuju ih i dijele recenzije. Među najpoznatijim aplikacijama u ovom području su Hungree, Restaurant roulette, TripAdvisor i Yelp. Iako su ove aplikacije popularne i nude mnoge slične funkcionalnosti, postoje ključne razlike u odnosu na naš projekt.

Naša aplikacija uključuje jedinstvenu funkcionalnost za grupno odlučivanje i ocjenjivanje restorana, dok postojeće aplikacije većinom omogućuju individualne recenzije.

Pružanje verifikacije restorana i potpuna administracija pristupa platformi nudi veću kontrolu nad sadržajem i sigurno okruženje za korisnike, što nije uvijek slučaj u drugim rješenjima.

- Hungree je aplikacija najsličnija našem projektu. Koristi mehanizam grupnog glasanja za odabir restorana, gdje članovi grupe mogu predlagati restorane i glasati za opcije koje im se sviđaju. Glavna razlika u odnosu na "Book 'n Bite" je ta što Hungree nema ocjenjivanje pojedinačnih jela, već se fokusira na cjelokupni izbor restorana. Također, "Book 'n Bite" koristi sustav ocjenjivanja s ocjenama od 1 do 5 za svaki proizvod iz različitih restorana, omogućavajući preciznije prilagođavanje preferencijama članova grupe.

Hungree is a **fun and easy way to decide what to eat**

Create rooms and share with all your friends

Let's find some restaurants

11:46 4G Yes, help me decide

Welcome, we hope you are hungree

Rooms Now In Progress Decided

Birthday Dinner

4:11 4G

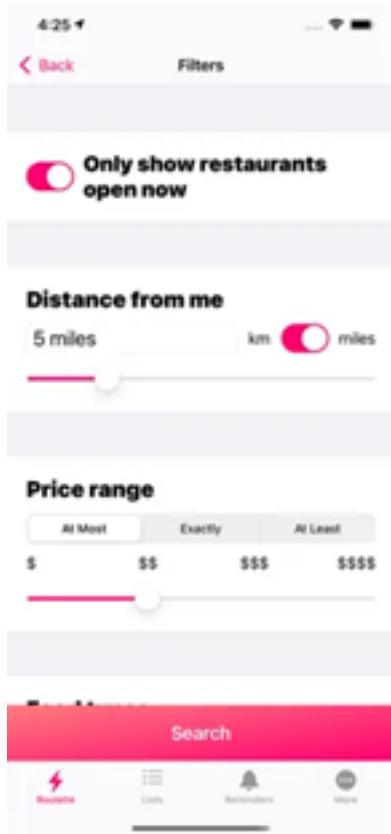
Results

Members Alyssa B Keanan M

Ranking

Restaurant	Emoji	Count
Boston Pizza	😊	1
McDonald's	😊	1
Domino's Pizza	😊	1
Starbucks	😊	1
Popeye's Louisiana Kitchen	😊	1
Dairy Queen	😐	1

Use My Location Set Location By Map Search By Zip Add My Own Remove Ads



Skup korisnika koji bi mogao biti zainteresiran za ostvareno rješenje

Proizvod će biti koristan širokom spektru korisnika:

- Opća populacija korisnika koji traže praktične informacije i preporuke za restorane.
- Grupe prijatelja i obitelji koje žele zajednički odabrati restorane na temelju zajedničkih preferencija i ocjena.
- Ljubitelji hrane koji traže nove restorane s visokom ocjenom i detaljnim recenzijama.
- Korisnici sa specifičnim prehrabbenim preferencijama (npr. vegani, bezglutenci), koji žele lako pronaći restorane koji udovoljavaju njihovim dijetetskim zahtjevima.
- Turističke grupe koje često istražuju gastronomске mogućnosti.
- Restorani koji žele privući korisnike putem kvalitetnih recenzija i preporuka te bolje upravljati svojim profilima na platformi.

Mogućnost prilagodbe rješenja

Aplikacija uključuje mogućnosti prilagodbe za različite vrste korisnika:

- Široke ponude ocjenjivačkih kategorija: Ocjenjivačima omogućavaju filtriranje po kriterijima kao što su vrste kuhinje, dijetetski zahtjevi (veganski, bezglutenski itd.).
- Geografske prilagodbe: Sustav može filtrirati restorane prema udaljenosti ili omogućiti korisnicima pretragu po gradovima i regijama.

Moguće nadogradnje projektnog zadataka

Nadogradnje koje bi mogle biti implementirane u budućnosti uključuju:

- Integracija s platformama za naručivanje hrane (npr. Uber Eats, Glovo), kako bi korisnici mogli naručiti iz restorana direktno kroz aplikaciju.
- Povezivanje s društvenim mrežama za dijeljenje recenzija i preporuka među prijateljima.
- Povećanje personalizacije s naprednjim sustavima preporuka baziranim na ponašanju korisnika. Aplikacija bi mogla analizirati prethodna ocjenjivanja korisnika i preporučiti restorane slične onima koje su korisnici ranije ocijenili pozitivno.
- Implementacija loyalty programa za restorane koji žele nagraditi vjerne kupce.
- Integracija s vanjskim sustavima za rezervaciju stolova.

Funkcionalni zahtjevi

ID za- ht- jeva	Opis	Priro- itet Zahtjev ocjenji- vača	Kriteriji prihvaćanja
F- 001	Sustav omogućuje neregistriranom ocjenjivaču registraciju putem Google računa (Google OAuth 2.0).	Visok Zahtjev ocjenji- vača	Ocenjivač se može registrirati putem Google računa.
F- 002	Sustav omogućuje registriranom ocjenjivaču prijavu na svoj račun putem Google računa (Google OAuth 2.0).	Visok Zahtjev ocjenji- vača	Ocenjivač se može prijaviti putem Google računa.

ID za- ht- jeva	Opis	Prioritet Zahtjev ocjenji- vača	Kriteriji prihvaćanja
F-003	Sustav omogućuje registriranom ocjenjivaču generiranje koda za pristup grupi.	Visok Zahtjev ocjenji- vača	Ocenjivač može generirati jedinstveni kod za pristup grupi.
F-004	Sustav omogućuje registriranom ocjenjivaču unos koda za pristup grupi.	Visok Zahtjev ocjenji- vača	Ocenjivač može unijeti ispravan kod za pridruživanje grupi i uspješno se pridružiti.
F-005	Sustav omogućuje registriranom ocjenjivaču odabir vrste proizvoda i lokacije za pretraživanje restorana.	Visok Zahtjev ocjenji- vača	Ocenjivač može odabrati vrstu proizvoda i lokaciju te dobiti listu relevantnih restorana.
F-006	Sustav omogućuje registriranom ocjenjivaču ocjenjivanje liste proizvoda.	Srednji Zahtjev ocjenji- vača	Ocenjivač može ocijeniti listu proizvoda, što će ažurirati rang liste proizvoda u sustavu.
F-007	Sustav omogućuje registriranom ocjenjivaču pretraživanje rang liste restorana.	Srednji Zahtjev ocjenji- vača	Ocenjivač može pregledati rang listu restorana na temelju različitih kriterija.
F-008	Sustav omogućuje korisnicima odjavu iz sustava.	Visok Zahtjev ocjenji- vača	Korisnik se može uspješno odjaviti iz sustava.
F-009	Sustav omogućuje korisnicima pregled vlastitog profila.	Srednji Zahtjev ocjenji- vača	Korisnici mogu vidjeti svoje podatke na vlastitom profilu.
F-010	Sustav omogućuje korisnicima ocjenjivanje restorana.	Srednji Zahtjev ocjenji- vača	Korisnici mogu ocijeniti restorane i ostaviti komentare.
F-011	Sustav omogućuje korisnicima pregled lokacije pojedinog restorana.	Srednji Zahtjev ocjenji- vača	Korisnici mogu vidjeti geolokaciju restorana.
F-012	Sustav omogućuje korisnicima pregled liste proizvoda za ocjenjivanje.	Srednji Zahtjev ocjenji- vača	Korisnici mogu vidjeti proizvode koje trebaju ocijeniti.

ID za- ht- jeva	Opis	Prioritet	Kriteriji prihvaćanja
F-013	Sustav omogućuje neregistriranom restoranu registraciju putem Google računa (Google OAuth 2.0).	Visok Zahtjev	Restoran se može registrirati restorana putem Google računa.
F-014	Sustav omogućuje registriranom restoranu prijavu na svoj račun putem Google računa (Google OAuth 2.0).	Visok Zahtjev	Restoran se može prijaviti restorana putem Google računa.
F-015	Sustav omogućuje registriranom restoranu dodavanje proizvoda na svoju stranicu.	Srednji Zahtjev	Restoran može dodati novi restorana proizvod sa svim potrebnim detaljima na svoju stranicu u sustavu.
F-016	Sustav omogućuje registriranom restoranu ažuriranje podataka o svom restoranu.	Srednji Zahtjev	Restoran može urediti restorana informacije o svojoj ponudi i radnom vremenu.
F-017	Sustav omogućuje restoranima uklanjanje proizvoda.	Srednji Zahtjev	Restoran može ukloniti restorana proizvod sa svog profila.
F-018	Sustav omogućuje restoranima postavljanje svoje lokacije.	Srednji Zahtjev	Restorani mogu unijeti restorana svoju lokaciju pomoći kartografske značajke.
F-019	Sustav omogućuje administratoru registraciju putem Google računa (Google OAuth 2.0).	Visok Zahtjev administra-tora	Administrator se može registrirati putem Google računa.
F-020	Sustav omogućuje administratoru prijavu u sustav putem Google računa (Google OAuth 2.0).	Visok Zahtjev administra-tora	Administrator se može prijaviti putem Google računa.
F-021	Sustav omogućuje administratoru dohvaćanje liste ocjenjivača.	Visok Zahtjev administra-tora	Administrator može pregledati i filtrirati listu svih ocjenjivača u sustavu.
F-022	Sustav omogućuje administratoru dohvaćanje liste restorana.	Visok Zahtjev administra-tora	Administrator može pregledati i filtrirati listu svih restorana u sustavu.

ID zahtjeva	Opis	Prioritet	Kriteriji prihvaćanja
F-023	Sustav omogućuje administratoru uklanjanje korisničkog profila ili restorana.	Visok Zahtjev administra-tora	Administrator može deaktivirati ili ukloniti korisnički profil ili restoran iz sustava.
F-024	Sustav omogućuje administratoru pregled profila drugih korisnika.	Visok Zahtjev administra-tora	Administrator može pristupiti profilu korisnika i vidjeti njihove javne podatke.
F-025	Sustav omogućuje administratoru pregled liste restorana za verifikaciju.	Visok Zahtjev administra-tora	Administrator može vidjeti sve restorane koji čekaju verifikaciju.
F-026	Sustav omogućuje administratoru blokiranje korisnika.	Visok Zahtjev administra-tora	Administrator može označiti korisnika kao blokiranog, čime mu onemogućava daljnju prijavu.
F-027	Sustav omogućuje administratoru odblokiranje korisnika.	Visok Zahtjev administra-tora	Administrator može ukloniti status blokade s korisnika i omogućiti mu ponovno prijavljivanje.
F-028	Sustav omogućuje administratoru prihvatanje zahtjeva za verifikaciju restorana.	Visok Zahtjev administra-tora	Administrator može potvrditi restoran i dodati ga u sustav.
F-029	Sustav omogućuje administratoru odbijanje zahtjeva za verifikaciju restorana.	Visok Zahtjev administra-tora	Administrator može odbiti zahtjev restorana za pristup sustavu.

Ostali zahtjevi

Zahtjevi za korištenje

ID zahtjeva	Opis	Prioritet
NF-1.1	Aplikacija mora biti jednostavna za korištenje.	Visok
NF-1.2	Aplikacija mora biti javno dostupna.	Visok
NF-1.3	Aplikacija se mora moći pokrenuti na svakome web pregledniku.	Visok

Zahtjevi za performanse

ID zahtjeva	Opis	Prioritet
NF-2.1	Sustav mora omogućiti rad više korisnika u stvarnom vremenu.	Visok

Zahtjevi za sigurnost

ID zahtjeva	Opis	Prioritet
NF-3.1	Korisnici se moraju moći sigurno prijaviti u sustav putem vanjske autentifikacije (OAuth 2.0).	Visok
NF-3.2	Svi osjetljivi podaci moraju biti šifrirani u bazi podataka.	Visok

Zahtjevi za održavanje

ID zahtjeva	Opis	Prioritet
NF-4.1	Sustav treba biti oblikovan tako da omogućuje jednostavno održavanje.	Visok
NF-4.1.1	Sustav treba imati dovoljnu dokumentaciju.	Visok

Zahtjevi za lokalizaciju

ID zahtjeva	Opis	Prioritet
NF-5.1	Sustav mora podržavati hrvatsku abecedu.	Srednji

Zahtjevi za integraciju

ID zahtjeva	Opis	Prioritet
NF-6.1	Sustav mora biti integriran s Google Maps API-jem za prikaz lokacija restorana.	Visok
NF-6.2	Sustav mora biti integriran s OAuth2 za vanjsku autentifikaciju putem pružatelja kao što je Google.	Visok

Zahtjevi za razvoj

ID zahtjeva	Opis	Prioritet
NF-7.1	Sustav mora biti ostvaren koristeći objektno orijentirane jezike.	Visok

Dionici

1. Ocjenjivači

Ocenjivači su dionici koji koriste aplikaciju za pridruživanje grupama, ocjenjivanje proizvoda i sudjelovanje u kreiranju rang lista restorana prema zajedničkim preferencijama.

2. Restorani

Restorani predstavljaju dionike koji koriste sustav za upravljanje svojim profilom i prezentaciju proizvoda, čime privlače potencijalne posjetitelje.

3. Administrator

Administrator je odgovoran za upravljanje korisničkim i restoranskim profilima, održavanje stabilnosti sustava te osiguravanje pridržavanja smjernica.

4. Razvojni tim

Razvojni tim zadužen je za izradu, testiranje i održavanje aplikacije, a njihov interes uključuje stabilnost, skalabilnost i pouzdanost aplikacije.

Aktori i njihovi funkcionalni zahtjevi

Aktor	Opis	Funkcionalni zahtjevi
A-1 Ocenj- jivač (inicija- tor)	Korisnik koji se registrira, prijavljuje, pridružuje grupama i ocjenjuje proizvode.	F-001, F-002, F-003, F-004, F-005, F-006, F-007, F-008, F-009, F-010, F-011, F-012
A-2 Restoran (inicija- tor)	Korisnik koji upravlja profilom restorana i Restoran proizvodima dostupnim na svojoj stranici.	F-013, F-014, F-015, F-016, F-017
A-3 Ad- minis- trator (inicija- tor)	Osoba odgovorna za nadzor nad sustavom, pregled korisnika i restorana te upravljanje pristupima.	F-018, F-019, F-020, F-021, F-022, F-023, F-024, F-025, F-026, F-027, F-028, F-029

Aktor	Opis	Funkcionalni zahtjevi
A-4 Baza podataka (sudionik)	Omogućava pohranu i dohvata podataka o korisnicima, restoranima, grupama i ocjenama.	Povezana sa svim funkcionalnim zahtjevima koji uključuju pohranu ili dohvata podataka.

Obrasci uporabe

Opis obrazaca uporabe

UC1 - Registracija ocjenjivač

- Glavni sudionik: Neregistrirani ocjenjivač
- Cilj: Napraviti korisnički račun kojim se pristupa sustavu
- Sudionici: Baza podataka, vanjski servis za autentifikaciju (Google OAuth2)
- Preduvjet: Korisnik nije registriran
- Opis osnovnog tijeka: > 1. Korisnik odabire opciju za prijavu putem vanjskog servisa > 2. Aplikacija preusmjerava korisnika na servis za autentifikaciju > 3. Korisnik unosi potrebne podatke za autentifikaciju u vanjskom servisu > 4. Vanjski servis potvrđuje identitet i vraća podatke aplikaciji > 5. Stvara se novi korisnički račun čiji se podatci pohranjuju u bazu podataka > 6. Aplikacija preusmjerava korisnika na početnu stranicu
- Opis mogućih odstupanja: > 3.a Korisnik ne postoji ili je lozinka netočna > 1. Sustav javlja korisniku da provjeri podatke koje je upisao

UC2 - Registracija restorana

- Glavni sudionik: Neregistrirani restoran
- Cilj: Napraviti korisnički račun kojim se pristupa sustavu
- Sudionici: Administrator, baza podataka, vanjski servis za autentifikaciju (Google OAuth2)
- Preduvjet: Korisnik nije registriran
- Opis osnovnog tijeka: > 1. Korisnik odabire opciju za prijavu putem vanjskog servisa > 2. Aplikacija preusmjerava korisnika na servis za autentifikaciju > 3. Korisnik unosi potrebne podatke za autentifikaciju u vanjskom servisu > 4. Vanjski servis potvrđuje identitet i vraća podatke aplikaciji > 5. Korisnik ispunjava formu sa informacijama o restoranu > 6. Neregistrirani korisnik čeka potvrdu verifikacije od strane administratora > 7. Stvara se novi korisnički račun čiji se podatci pohranjuju u bazu podataka > 8. Aplikacija preusmjerava korisnika na početnu stranicu
- Opis mogućih odstupanja: > 3.a Korisnik ne postoji ili je lozinka netočna > 1. Sustav javlja korisniku da provjeri podatke koje je upisao

6.a Administrator odbija zahtjev za verifikaciju restorana 1. Neregistrirani korisnik dobiva obavijest o odbijanju verifikacije

UC3 - Prijava

- Glavni sudionik: Neprijavljeni korisnik
- Cilj: Dobiti pristup određenim korisničkim funkcijama
- Sudionici: Baza podataka, vanjski servis za autentifikaciju (Google OAuth2)
- Preduvjet: Korisnik se registrirao
- Opis osnovnog tijeka: > 1. Korisnik odabire opciju za prijavu putem vanjskog servisa > 2. Aplikacija preusmjerava korisnika na servis za autentifikaciju > 3. Korisnik unosi potrebne podatke za autentifikaciju u vanjskom servisu > 4. Vanjski servis potvrđuje identitet i vraća podatke aplikaciji > 5. Stvara se novi korisnički račun čiji se podaci pohranjuju u bazu podataka > 6. Aplikacija preusmjerava korisnika na početnu stranicu
- Opis mogućih odstupanja: > 3.a Korisnik ne postoji ili je lozinka netočna > 1. Sustav javlja korisniku da provjeri podatke koje je upisao

UC4 - Odjava

- Glavni sudionik: Korisnik
- Cilj: Odjaviti se iz sustava
- Sudionici: Baza podataka
- Preduvjet: Korisnik je prijavljen
- Opis osnovnog tijeka: > 1. Korisnik odabire opciju "Odjavi se" > 2. Korisnik napušta sustav

UC5 - Pridruživanje postojećoj grupi

- Glavni sudionik: Ocjenjivač
- Cilj: Pridružiti se već stvorenoj grupi ocjenjivača
- Sudionici: Baza podataka
- Preduvjet: Korisnik je prijavljen, postoji grupa kojoj se korisnik želi pridružiti
- Opis osnovnog tijeka: > 1. Korisnik klikom na polje za unos upisuje kod grupe kojoj se želi pridružiti > 2. Klikom na gumb "Pridruži se" korisnik postaje pripadnik grupe
- Opis mogućih odstupanja: > 1.a Korisnik unosi neispravan ili ne postojeći kod grupe > 1. Korisniku se ispisuje poruka o pogrešci > 2. Korisnik ponovo unosi kod grupe

UC6 - Izrada nove grupe

- Glavni sudionik: Ocjenjivač
- Cilj: Stvoriti novu grupu korisnika za odabir restorana
- Sudionici: Baza podataka

- Preduvjet: Korisnik je prijavljen
- Opis osnovnog tijeka: > 1. Korisnik odabire opciju "Stvori novu grupu" > 2. Korisniku se prikazuje stranica na kojoj odabire kategoriju proizvoda i lokaciju na karti te zatim odabire "Stvori grupu" > 3. Generira se kod grupe koji se zatim ispisuje korisniku > 4. Korisnik dijeli generirani kod drugim korisnicima

UC7 - Pregled profila drugih korisnika

- Glavni sudionik: Administrator
- Cilj: Pregledati informacije o drugom korisniku
- Sudionici: Baza podataka
- Preduvjet: Administrator je prijavljen
- Opis osnovnog tijeka: > 1. Administrator pronađe korisnika čiji profil želi pregledati > 2. Administrator odlazi na profil tog korisnika > 3. Administrator pregledava profil drugog korisnika, njegove javno dostupne podatke

UC8 - Pregled vlastitog profila

- Glavni sudionik: Ocjenjivač, administrator
- Cilj: Pregledati korisničke podatke svog profila
- Sudionici: Baza podataka
- Preduvjet: Ocjenjivač/administrator je prijavljen
- Opis osnovnog tjedna > 1. Korisnik odabire opciju "Moj profil" > 2. Korisniku se prikazuje stranica vlastitog profila i svi njegovi korisnički podaci > 3. Korisnik pregledava informacije o svom profilu

UC9 - Uređivanje vlastitog profila

- Glavni sudionik: Ocjenjivač, administrator
- Cilj: Promijeniti korisničke podatke
- Sudionici: Baza podataka
- Preduvjet: Korisnik/administrator je prijavljen
- Opis osnovnog tijeka: > 1. Korisnik na stranici svog profila odabire opciju "Uredi profil" > 2. Korisnik mijenja svoje korisničke podatke i njihovu dostupnost > 3. Korisnik potvrđuje promjene odabirom opcije "Spremi promjene" > 4. Baza podataka se ažurira
- Opis mogućih odstupanja
 - > 2.a Unos podataka u nedozvoljenom formatu ili unos već zauzetog nadimka ili e-mail adrese
 - > 1. Korisnik dobiva odgovarajuću obavijest o neispravnosti podataka > 2. Korisnik mijenja potrebne podatke i završava s unosom ili odustaje od promjene
 - 3.a Korisnik odabere opciju "Odustani" 1. Promjene se ne spremaju
 - 3.a Korisnik odabere opciju "Odustani" 1. Promjene se ne spremaju

UC10 - Dodavanje novog proizvoda

- Glavni sudionik: Restoran
- Cilj: Stvoriti novi proizvod koje će biti dio ponude restorana
- Sudionici: Baza podataka
- Preduvjet: Restoran je prijavljen
- Opis osnovnog tijeka: > 1. Restoran unutar pregleda vlastite stranice odabire opciju “Dodaj novi proizvod” > 2. Prikazuje se stranica na kojoj restoran popunjava informacije o proizvodu te dodaje sliku proizvoda > 3. Odabirom na opciju “Stvori proizvod” novi proizvod se dodaje u bazu podataka te je vidljiv na stranici restorana

UC11 - Uklanjanje postojećeg proizvoda

- Glavni sudionik: Restoran
- Cilj: Ukloniti prethodno stvoreni proizvod iz ponude restorana
- Sudionici: Baza podataka
- Preduvjet: Restoran je prijavljen, postoji proizvod za uklanjanje
- Opis osnovnog tijeka: > 1. Restoran na popisu proizvoda pronađe proizvod koji želi ukloniti i odabire njemu pridruženu ikonu za uklanjanje proizvoda > 2. Prikazuje se dijaloški okvir potvrde uklanjanja proizvoda > 3. Restoran unutar dijaloškog okvira odabire opciju “Ukloni” čime potvrđuje svoj odabir > 4. Proizvod se uklanja iz baze podataka te se više ne prikazuje na popisu proizvoda
- Opis mogućih odstupanja
 - > 3.a Restoran unutar dijaloškog okvira potvrde odabire opciju “Prekid”
 - > 1. Akcija uklanjanja proizvoda se ne provodi

UC12 - Pregled proizvoda pojedinog restorana

- Glavni sudionik: Korisnik
- Cilj: Pregledati proizvode koji se nalaze u ponudi pojedinog restorana
- Sudionici: Baza podataka
- Preduvjet: Korisnik je prijavljen
- Opis osnovnog tijeka: > 1. Korisnik (može biti i sam restoran) odlazi na profil restorana > 2. Korisniku se prikazuju proizvodi iz ponude tog restorana

UC13 - Pregled stranice restorana

- Glavni sudionik: Ocjenjivač, administrator
- Cilj: Pregled stranice pojedinog restorana i informacija dostupnih na njoj
- Sudionici: Restoran, baza podataka
- Preduvjet: Ocjenjivač/administrator je prijavljen
- Opis osnovnog tijeka: > 1. Korisnik pronađe restoran čiji profil želi pregledati > 2. Korisnik odlazi na profil tog restorana > 3. Korisnik pregledava profil restorana, njegove javno dostupne podatke

UC14 - Ocjenjivanje restorana

- Glavni sudionik: Ocjenjivač, administrator
- Cilj: Ocijeniti restoran po završetku posjeta restoranu
- Sudionici: Restoran, baza podataka
- Preduvjet: Ocjenjivač/administrator je prijavljen, korisnik je već posjetio restoran koji želi ocijeniti
- Opis osnovnog tijeka:
 1. Korisnik ocjenjuje restoran i ostavlja komentar o restoranu
 2. Recenzija je vidljiva na profilu restorana
 3. Ažurira se baza podataka

UC15 - Pregled liste ocjenjivača

- Glavni sudionik: Administrator
- Cilj: Pregled liste ocjenjivača te njihovih informacija
- Sudionici: Ocjenjivač, baza podataka
- Preduvjet: Administrator je prijavljen
- Opis osnovnog tijeka: > 1. Administrator odabire opciju “Korisnici” u svojoj nadzornoj ploči

UC16 - Pregled liste restorana

- Glavni sudionik: Administrator
- Cilj: Pregled liste restorana te njihovih informacija
- Sudionici: Restoran, baza podataka
- Preduvjet: Administrator je prijavljen
- Opis osnovnog tijeka: > 1. Administrator odabire opciju “Restorani” u svojoj nadzornoj ploči

UC17 - Pregled liste restorana za verifikaciju

- Glavni sudionik: Administrator
- Cilj: Omogućiti administratoru uvid u restorane koji žele pristupiti sustavu
- Sudionici: Administrator, baza podataka
- Preduvjet: Administrator je prijavljen, restoran je poslao zahtjev za verifikacijom
- Opis osnovnog tijeka: > 1. Administrator odabire opciju “Restorani” u svojoj nadzornoj ploči > 2. Administratoru se otvara nova stranica s listom svih restorana > 3. Administrator odabire opciju “Čeka verifikaciju”

UC18 - Pregled rang liste restorana

- Glavni sudionik: Ocjenjivač
- Cilj: Prikaz korisnicima rang pojedinih restorana
- Sudionici: Baza podataka
- Preduvjet: Restoran je prijavljen, ocjenjivač je prijavljen, stvorena je grupa

- Opis osnovnog tijeka: > 1. Ocjenjivač je napustio stranicu za čekanje > 2. Ocjenjivaču se prikazuje rang lista restorana

UC19 - Odabir lokacije restorana

- Glavni sudionik: Restoran
- Cilj: Restoran postavlja svoju geolokaciju
- Sudionici: Baza podataka
- Preduvjet: Restoran je prijavljen
- Opis osnovnog tijeka: > 1. Restoran odabire opciju “Odaberi lokaciju” > 2. Restoranu se prikazuje karta > 3. Restoran odabire svoju geolokaciju > 4. Restoran odabire opciju “Potvrди”
- Opis mogućih odstupanja: > 3.1. Restoran odabire pogrešnu geolokaciju

UC20 - Brisanje profila ocjenjivača

- Glavni sudionik: Ocjenjivač, administrator
- Cilj: Brisanje svog korisničkog računa iz baze podataka aplikacije
- Sudionici: Baza podataka
- Preduvjet: Administrator je prijavljen pri brisanju određenog ocjenjivača, ocjenjivač je prijavljen pri brisanju vlastitog računa
- Opis osnovnog tijeka: > 1. Ocjenjivač odlučuje obrisati svoj profil odabirom opcije “Obriši profil” > 2. Ocjenjivač potvrđuje svoju odluku > 3. Profil ocjenjivača se uklanja iz baze podataka > 4. Otvara se stranica za ponovnu registraciju

UC21 - Brisanje profila restorana

- Glavni sudionik: Restoran, administrator
- Cilj: Restoranu omogućiti brisanje vlastitog profila
- Sudionici: Administrator
- Preduvjet: Administrator je prijavljen pri brisanju određenog restorana, restoran je prijavljen pri brisanju vlastitog profila
- Opis osnovnog tijeka: > 1. Restoran odlučuje obrisati svoj profil odabirom opcije “Obriši profil” > 2. Restoran potvrđuje svoju odluku > 3. Profil restoran se uklanja iz baze podataka > 4. Otvara se stranica za ponovnu registraciju

UC22 - Blokiranje korisnika

- Glavni sudionik: Administrator
- Cilj: Korisniku onemogućiti pristup aplikaciji
- Sudionici: Korisnik, baza podataka
- Preduvjet: Administrator je prijavljen, korisnički račun postoji
- Opis osnovnog tijeka: > 1. Administrator pristupa listi svih prijavljenih korisnika > 2. Administrator odabire određenog korisnika > 3. Adminis-

trator odabire opciju "Blokiraj korisnika" > 4. Korisnik je u bazi podataka označen kao blokiran te mu je onemogućena ponovna prijava

- Opis mogućih odstupanja: > 3.1. Korisnik je već blokiran

UC23 - Odblokiranje korisnika

- Glavni sudionik: Administrator
- Cilj: Korisniku ponovno omogućiti pristup aplikaciji
- Sudionici: Korisnik, baza podataka
- Preduvjet: Administrator je prijavljen, postojao je korisnički račun
- Opis osnovnog tijeka: > 1. Administrator pristupa listi svih blokiranih korisnika > 2. Administrator odabire određenog korisnika > 3. Administrator odabire opciju "Odblokiraj korisnika" > 4. Korisniku je u bazi podataka izbrisana oznaka da je blokiran te mu je omogućena ponovna prijava
- Opis mogućih odstupanja: > 3.1. Korisnik je već odblokiran

UC24 - Prihvatanje zahtjeva za stvaranje restorana

- Glavni sudionik: Administrator
- Cilj: Prihvatanje restorana u aplikaciju te spremanje u bazu podataka
- Sudionici: Restoran, baza podataka
- Preduvjet: Administrator je prijavljen, restoran nije verificiran, restoran je poslao zahtjev za verifikacijom
- Opis osnovnog tijeka: > 1. Administrator pristupa listi svih zahtjeva za stvaranje restorana > 2. Administrator odabire određeni restoran > 3. Administrator odabire opciju "Prihvati" > 4. Restoranu je prihvaćen zahtjev te se dodaje u bazu podataka
- Opis mogućih odstupanja: > 3.1. Restoran je već prihvaćen
> 3.2. Restoran je već odbijen

UC25 - Odbijanje zahtjeva za stvaranje restorana

- Glavni sudionik: Administrator
- Cilj: Odbijanje restoranu pristup u aplikaciju
- Sudionici: Restoran, baza podataka
- Preduvjet: Administrator je prijavljen, restoran nije verificiran, restoran je poslao zahtjev za verifikacijom
- Opis osnovnog tijeka: > 1. Administrator pristupa listi svih zahtjeva za stvaranje restorana > 2. Administrator odabire određeni restoran > 3. Administrator odabire opciju "Odbij" > 4. Restoranu nije prihvaćen zahtjev
- Opis mogućih odstupanja: > 3.1. Restoran je već prihvaćen
> 3.2. Restoran je već odbijen

UC26 - Odabir kategorije proizvoda

- Glavni sudionik: Ocjenjivač
- Cilj: Ocjenjivač koji je stvorio grupu bira željenu kategoriju proizvoda
- Sudionici: Restoran, baza podataka
- Preduvjet: Ocjenjivač je prijavljen, grupa je stvorena
- Opis osnovnog tijeka: > 1. Ocjenjivač pristupa listi svih kategorija proizvoda > 2. Ocjenjivač odabire željenu kategoriju proizvoda pritiskom na kategoriju
- Opis mogućih odstupanja: > 2.1. Ocjenjivač je već odabrao određenu kategoriju proizvoda

UC27 - Pregled lokacije pojedinog restorana

- Glavni sudionik: Ocjenjivač
- Cilj: Prikaz geolokacijskog položaja određenog restorana
- Sudionici: Restoran, baza podataka
- Preduvjet: Ocjenjivač je prijavljen, grupa je stvorena
- Opis osnovnog tijeka: > 1. Ocjenjivač odabire restoran na rang listi restorana > 2. Ocjenjivač odabire opciju "Prikaži lokaciju"

UC28 - Pregled liste proizvoda za ocjenjivanje

- Glavni sudionik: Ocjenjivač
- Cilj: Ocjenjivač ima mogućnost vidjeti proizvode koje će ocjeniti
- Sudionici: Restoran, baza podataka
- Preduvjet: Ocjenjivač je prijavljen, grupa je stvorena
- Opis osnovnog tijeka: > 1. Ocjenjivač se nalazi u grupi > 2. Ocjenjivač pristupa listi proizvoda za ocjenjivanje

UC29 - Ocjenjivanje proizvoda

- Glavni sudionik: Ocjenjivač
- Cilj: Ocjenjivač daje ocjenu odredenom proizvodu
- Sudionici: Restoran, baza podataka
- Preduvjet: Ocjenjivač je prijavljen, Ocjenjivač je posjetio restoran
- Opis osnovnog tijeka: > 1. Ocjenjivač odabire proizvod > 2. Ocjenjivač daje ocjenu proizvodu
- Opis mogućih odstupanja: > 2.1. Ocjenjivač je već ocijenio proizvod

Dijagrami obrazaca uporabe

Slika 3.1: Dijagram obrazaca uporabe - Funkcionalnost ocjenjivača

Slika 3.2: Dijagram obrazaca uporabe - Funkcionalnosti neregistriranih korisnika i restorana

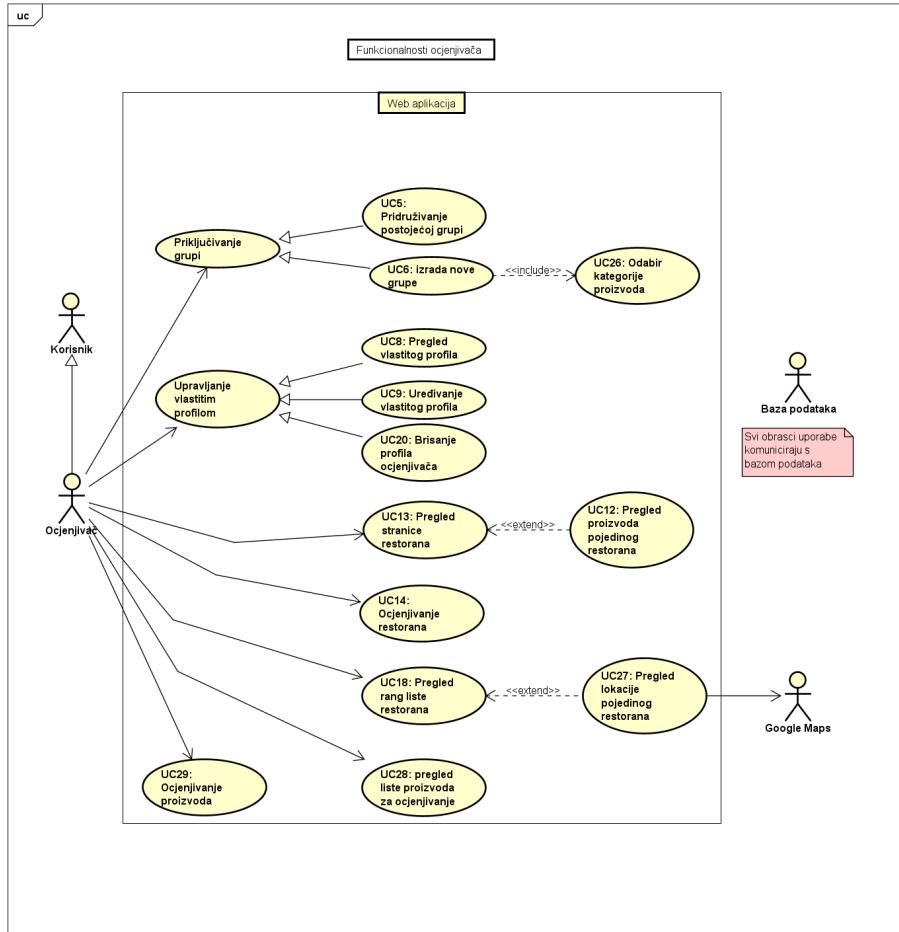


Figure 1: Dijagram obrazaca uporabe - Funkcionalnost ocjenjivača

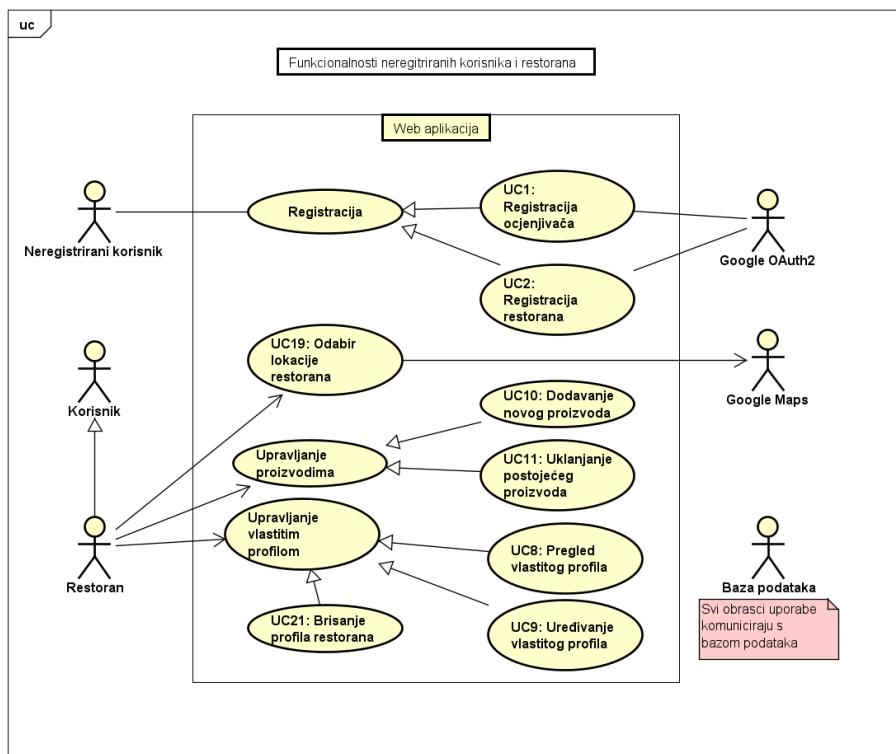


Figure 2: Dijagram obrazaca uporabe - Funkcionalnosti neregistriranih korisnika i restorana

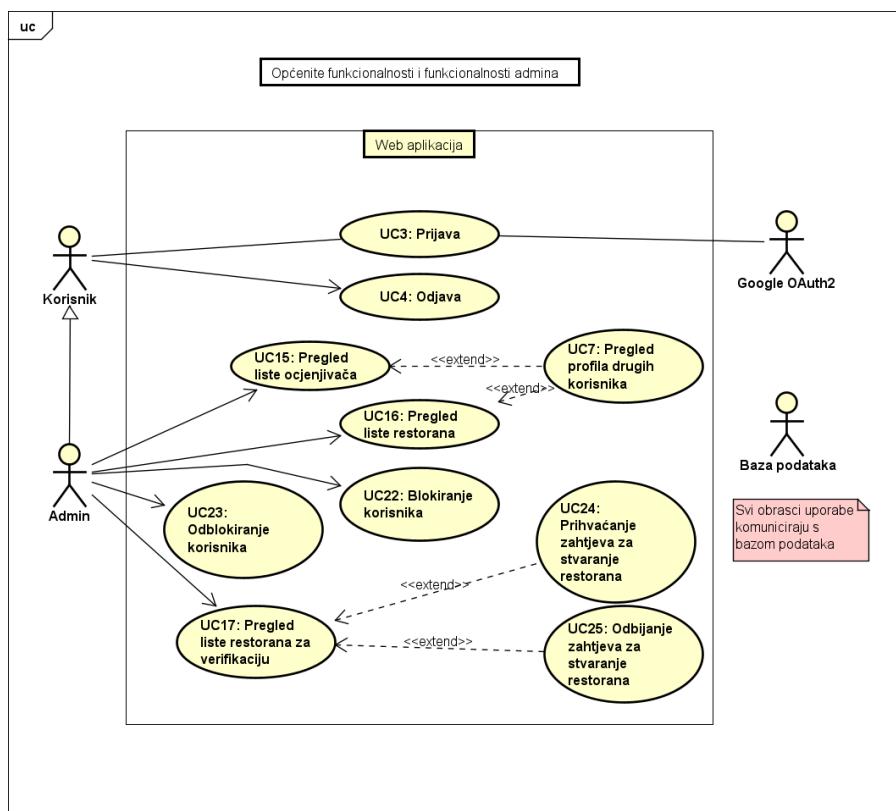


Figure 3: Dijagram obrazaca uporabe - Općenite funkcionalnosti i funkcionalnosti admina

Slika 3.3: Dijagram obrazaca uporabe - Općenite funkcionalnosti i funkcionalnosti admina

Sekvencijski dijagrami

Funkcionalnosti admina

Ovaj dijagram prikazuje aktivnosti koje administrator može obavljati kako bi nadzirao korisnike i restorane unutar aplikacije.

Administrator prvo pristupa svojoj nadzornoj ploči, gdje može pregledati popis svih ocjenjivača i restorana registriranih u aplikaciji te izvršavati potrebne administrativne radnje. Administrator može pregledavati listu ocjenjivača i odabrati profil pojedinog ocjenjivača kako bi provjerio njegove aktivnosti. Ako utvrdi kršenje pravila ili zloupotrebu, administrator može ukloniti ocjenjivača iz sustava, čime mu trajno onemogućava pristup aplikaciji.

Osim upravljanja ocjenjivačima, administrator može pregledavati i listu restorana. Pristupom listi restorana, administrator može pregledavati detaljne profile svakog restorana, uključujući informacije koje su restorani unijeli tijekom registracije. Ako pronađe restoran koji krši pravila aplikacije, administrator ga može ukloniti iz sustava, čime restoran postaje nedostupan ostalim korisnicima.

Ovaj dijagram jasno prikazuje kako administrator koristi aplikaciju za nadzor korisnika i osigurava da aplikacija ostane sigurna i pouzdana za sve korisnike.

Slika 3.4: Sekvencijski dijagram - Admin funkcionalnosti

Funkcionalnosti ocjenjivača

Ovaj dijagram opisuje postupke koje ocjenjivač poduzima u aplikaciji kako bi kreirao ili se pridružio grupi za ocjenjivanje, ocjenjivao proizvode te odabrao restoran na temelju rang liste.

Ocenjivač započinje interakciju tako što odabire opciju za kreiranje nove grupe. Aplikacija potom stvara novu grupu i generira jedinstveni kod koji ocjenjivač može podijeliti s drugim korisnicima kako bi im omogućio pridruživanje grupi. Alternativno, ako se želi pridružiti postojećoj grupi, ocjenjivač unosi kod grupe. Aplikacija provjerava kod, i ako je ispravan, pridružuje ga grupi.

Nakon pridruživanja grupi, ocjenjivaču se prikazuje popis proizvoda za ocjenjivanje. Ocjenjivač pregledava proizvode jedan po jedan i dodjeljuje ocjene prema vlastitim preferencijama. Ovaj proces ocjenjivanja ponavlja se dok ocjenjivač ne ocjeni sve proizvode s liste.

Kada svi članovi grupe završe ocjenjivanje, aplikacija obrađuje sve prikupljene ocjene i generira rang listu restorana, poredanu prema ukupnim preferencijama članova grupe. Ocjenjivač zatim može pregledavati profile restorana s rang liste i vratiti se na rang listu kako bi usporedio ostale opcije. Kada odabere restoran koji mu se najviše sviđa, aplikacija bilježi njegov odabir i vraća ga na glavnu stranicu.

Slika 3.5: Sekvencijski dijagram - Ocjenjivač funkcionalnosti

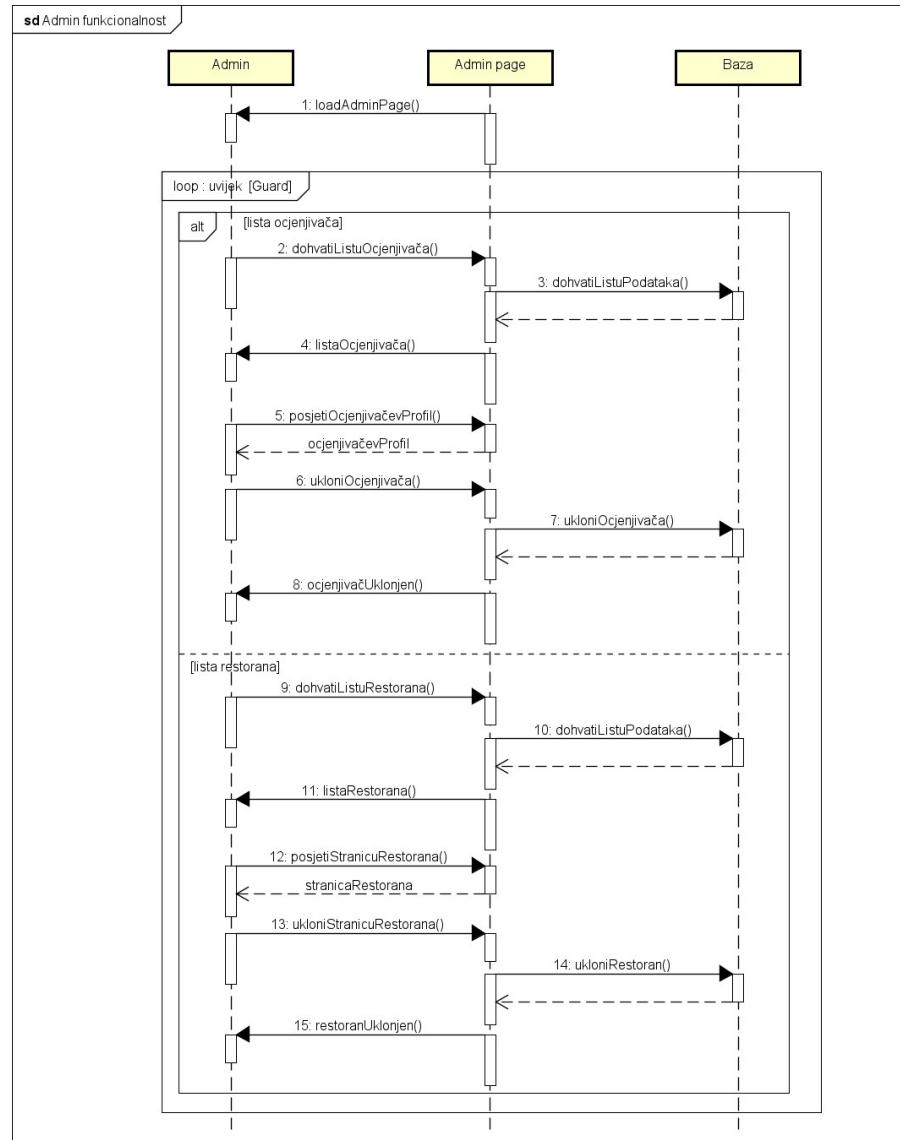


Figure 4: Sekvencijski dijagram - Admin funkcionalnosti

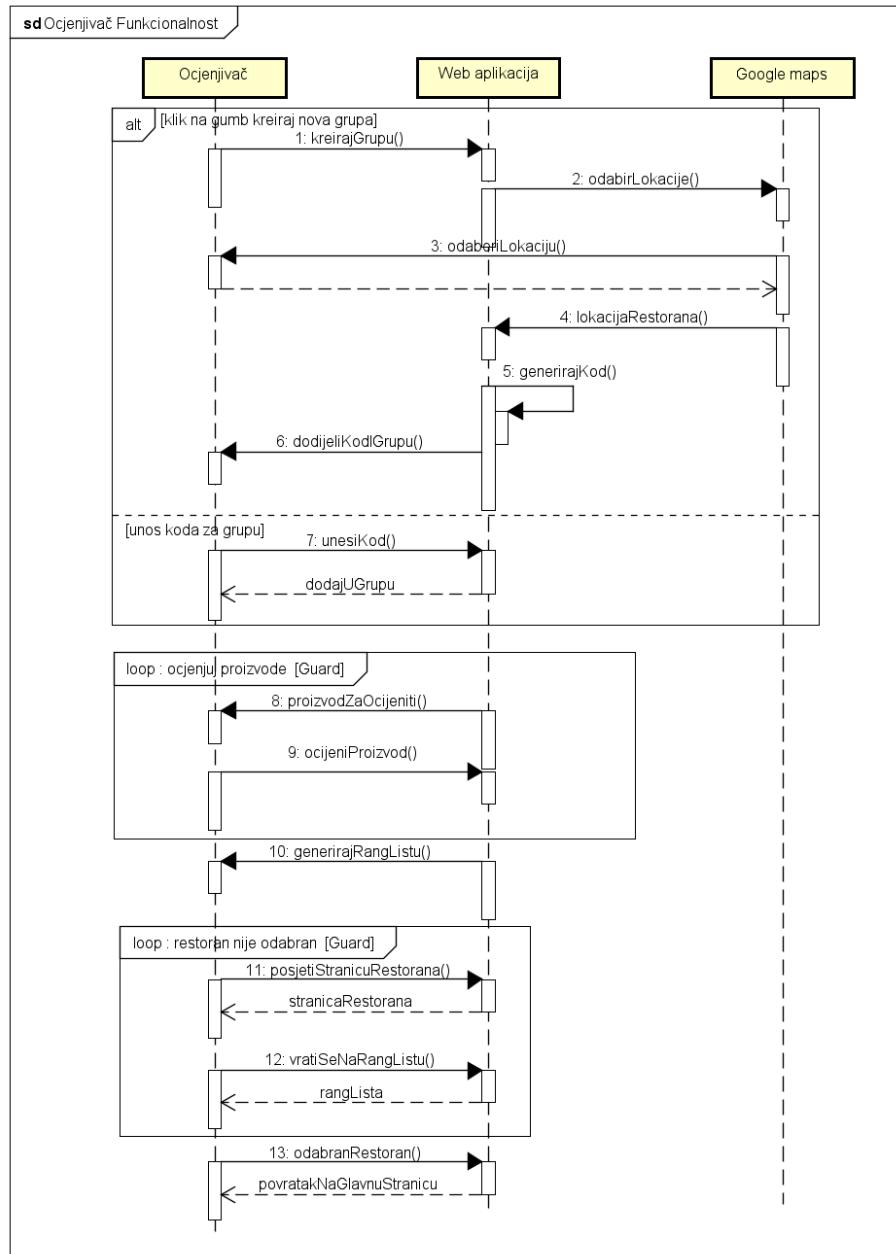


Figure 5: Sekvencijski dijagram - Ocjenjivač funkcionalnosti

Funkcionalnosti restorana

Ovaj dijagram opisuje osnovne funkcionalnosti koje su dostupne korisnicima s profilom restorana, uključujući unos i ažuriranje informacija o restoranu te dodavanje novih proizvoda.

Nakon prijave, restoranu se prikazuje forma za unos informacija o njegovom profilu, uključujući naziv, radno vrijeme, lokaciju i dodatne podatke poput broja telefona i poveznica na dostavne servise. Restoran popunjava formu, a podaci se spremaju u aplikaciju, omogućujući restoranu da prikaže te informacije korisnicima na svojoj stranici.

Restoran može u bilo kojem trenutku urediti svoje podatke. Kada odabere opciju za uređivanje, prikazuje mu se forma s postojećim podacima koje može izmijeniti. Nakon što unese nove informacije, ažurirani podaci se spremaju u sustav, a restoran se vraća na svoju stranicu s ažuriranim informacijama.

Također, restoran može dodati novi proizvod na svoju stranicu. Kada odabere opciju za dodavanje proizvoda, prikazuje mu se forma za unos informacija o proizvodu, uključujući naziv, opis, cijenu, tip, alergene i sliku proizvoda. Nakon što ispuni formu, proizvod se dodaje u bazu podataka i postaje vidljiv korisnicima na stranici restorana.

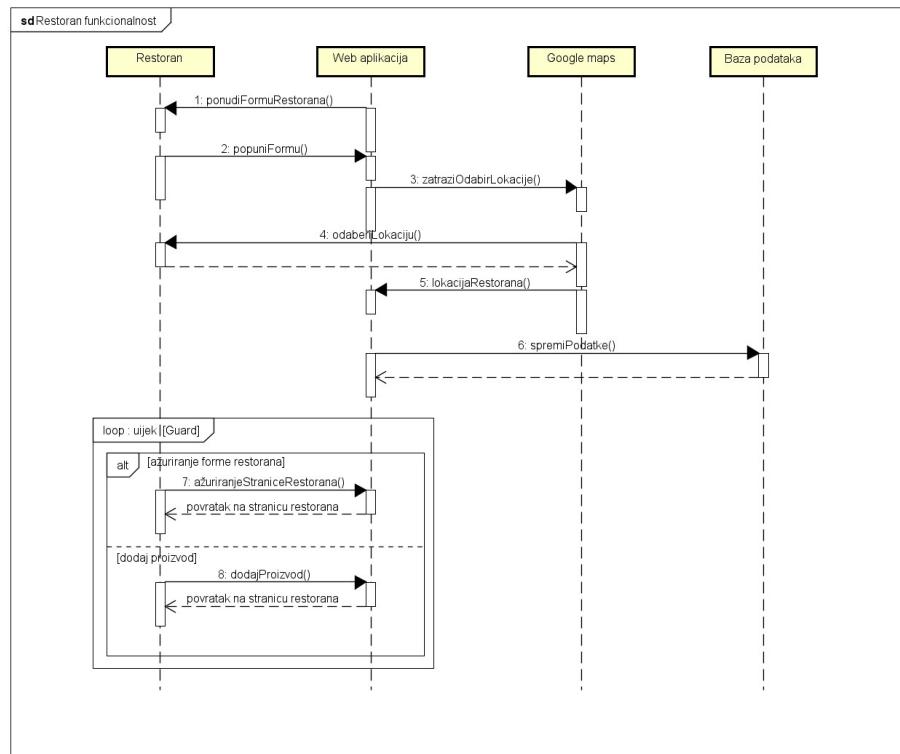


Figure 6: Sekvencijski dijagram - Restoran funkcionalnost

Slika 3.6: Sekvencijski dijagram - Restoran funkcionalnosti

Registracija i login

Ovaj dijagram prikazuje proces registracije i prijave korisnika, uključujući postupke za provjeru ispravnosti podataka i upravljanje pogreškama.

Korisnik započinje proces registracije ili prijave odabirom odgovarajuće opcije u aplikaciji. Ako odabere registraciju, prikazuje mu se forma za unos potrebnih podataka. Korisnik unosi svoje podatke, a aplikacija provjerava njihovu ispravnost, osiguravajući da su svi podaci u valjanom formatu i da nisu zauzeti. Ako postoji problem s podacima, korisnik prima obavijest o grešci i može ispraviti podatke te pokušati ponovno. Kada su podaci ispravni, aplikacija ih sprema u bazu podataka i potvrđuje korisnikov račun, nakon čega ga preusmjerava na glavnu stranicu.

Ako korisnik odabere prijavu, aplikacija mu prikazuje formu za unos podataka za prijavu. Nakon što unese podatke, aplikacija ih provjerava i osigurava da su točni. Ako postoje pogreške, korisnik prima obavijest o neispravnim podacima i može ih ispraviti. Kada su podaci za prijavu točni, aplikacija korisniku omogućava pristup i preusmjerava ga na glavnu stranicu.

Slika 3.7: Sekvencijski dijagram - Registracija i Login

Provjera uključenosti ključnih funkcionalnosti u obrasce uporabe

ID Obrasca Uporabe	Naziv Obrasca Uporabe	Povezani Funkcionalni Zahtjevi
UC1	Registracija ocjenjivača	F-001
UC2	Registracija restorana	F-013
UC3	Prijava	F-002, F-014, F-020
UC4	Odjava	F-008
UC5	Pridruživanje postojećoj grupi	F-004
UC6	Izrada nove grupe	F-003
UC7	Pregled profila drugih korisnika	F-024
UC8	Pregled vlastitog profila	F-009
UC9	Uređivanje vlastitog profila	F-009, F-016
UC10	Dodavanje novog proizvoda	F-015
UC11	Uklanjanje postojećeg proizvoda	F-017
UC12	Pregled proizvoda pojedinog restorana	F-012
UC13	Pregled stranice restorana	F-012

ID Obrasca Uporabe	Naziv Obrasca Uporabe	Povezani Funkcionalni Zahtjevi
UC14	Ocjenvivanje restorana	F-010
UC15	Pregled liste ocjenjivača	F-021
UC16	Pregled liste restorana	F-022
UC17	Pregled liste restorana za verifikaciju	F-025
UC18	Pregled rang liste restorana	F-007
UC19	Odabir lokacije restorana	F-018
UC20	Brisanje profila ocjenjivača	F-023
UC21	Brisanje profila restorana	F-023
UC22	Blokiranje korisnika	F-026
UC23	Odblokiranje korisnika	F-027
UC24	Prihvatanje zahtjeva za stvaranje restorana	F-028
UC25	Odbijanje zahtjeva za stvaranje restorana	F-029
UC26	Odabir kategorije proizvoda	F-005
UC27	Pregled lokacije pojedinog restorana	F-011
UC28	Pregled liste proizvoda za ocjenjivanje	F-012
UC29	Ocjenvivanje proizvoda	F-006

Arhitektura sustava

Opis arhitekture

Za projekt “Book ’n Bite” odabrana je MVC (Model-View-Controller) arhitektura. MVC arhitektura omogućava strukturiranje sustava kroz jasno definirane slojeve – Model, View i Controller – što olakšava organizaciju aplikacijskih pravila, korisničkog sučelja i upravljanje podacima. Koristeći ovu arhitekturu, aplikacija može lako podržavati različite korisničke uloge (ocjenjivač, restoran i administrator) s različitim funkcionalnostima. MVC stil je također prikladan za projekte koji zahtijevaju jasnu podjelu odgovornosti i fleksibilnu proširivost.

Podsistavi

Projekt “Book ’n Bite” organiziran je u tri glavna podsustava:

- Frontend (View): Frontend je razvijen u Reactu i zadužen je za prikaz korisničkog sučelja i interakciju s korisnicima. Komponente React-a prikazuju različite dijelove aplikacije kao što su stranice za registraciju, prijavu, upravljanje grupama, ocjenjivanje proizvoda i pregled rang lista restorana. Virtualni DOM, kojeg koristi React, omogućava brze promjene prikaza, čime se poboljšavaju performanse aplikacije.

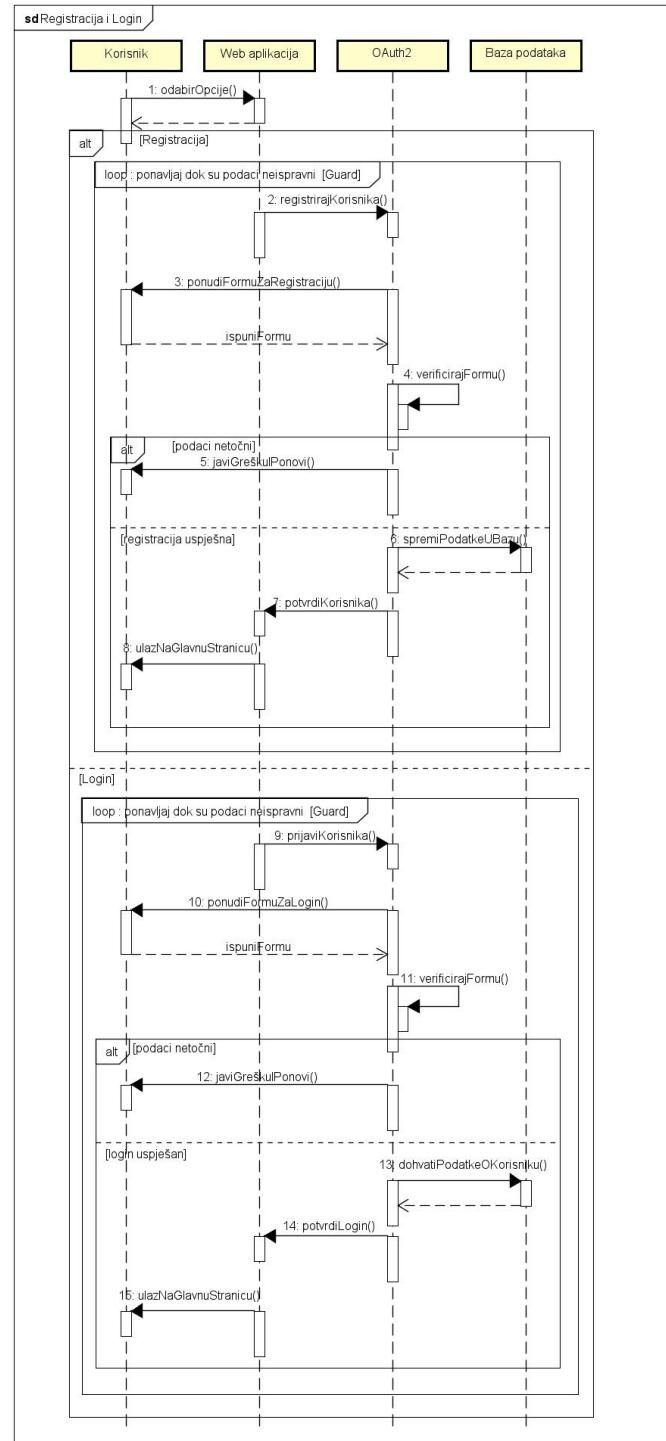


Figure 7: Sekvencijski dijagram - Registracija i Login
30

- Backend (Controller): Backend koristi Spring Boot, koji omogućava brzu izradu web aplikacija i pruža potrebne servise za upravljanje logikom sustava i obradom korisničkih zahtjeva. Kontroleri u Spring Boot-u primaju i obrađuju zahtjeve te upravljaju interakcijama između frontend-a i baze podataka. Servisi provode logiku obrade podataka i validaciju unosa, dok modeli predstavljaju strukturu podataka za komunikaciju s bazom.
- Sloj pristupa podacima (Model): Sloj pristupa podacima zadužen je za pohranu podataka u PostgreSQL bazu podataka, relacijsku bazu koja pruža efikasan pristup i integraciju s modelima aplikacije. U bazi podataka pohranjuju se ključni podaci o korisnicima, restoranima, proizvodima, grupama i ocjenama, omogućujući brzo i učinkovito dohvaćanje podataka za pravila rada sustava i prikaz korisničkog sučelja.

Sinteza ovih slojeva - korisničkog sučelja na razini Reacta, web-aplikacijskog sloja u Spring Bootu i sloja pristupa podacima u Spring Bootu - stvara temelj za razvoj visoko skalabilnih i funkcionalnih web-aplikacija. Korisnici ostvaruju interakciju s aplikacijom preko intuitivnog React korisničkog sučelja, dok Spring Boot preuzima odgovornost za obradu njihovih zahtjeva i poslovne logike. Istovremeno, sloj pristupa podacima omogućuje efikasnu komunikaciju s bazom podataka, omogućujući pohranu i dohvat podataka s pouzdanošću i učinkovitšćocu.

Obrazloženje odabira arhitekture

Odabir MVC arhitekture temelji se na ključnim čimbenicima kao što su:

* Jasna podjela odgovornosti između slojeva, što omogućava lakše održavanje, testiranje i nadogradnju aplikacije. * Skalabilnost: MVC arhitektura omogućava da se svaki sloj proširuje ili mijenja bez potrebe za velikim promjenama u ostatku sustava. * Fleksibilnost: Različite korisničke uloge (ocjenjivači, restorani, administratori) zahtijevaju različite funkcionalnosti i pristupe sustavu. MVC arhitektura podržava fleksibilno dodavanje i ažuriranje funkcionalnosti specifičnih za svaku ulogu.

Principi kao što su visoka kohezija i niska povezanost podržavaju održivost i modularnost aplikacije, omogućavajući da svaki sloj bude neovisan, čime se olakšava rad na specifičnim funkcionalnostima.

Organizacija sustava na visokoj razini

Aplikacija "Book 'n Bite" organizirana je prema klijent-poslužitelj arhitekturi, u kojoj klijent (frontend) upravlja prikazom i interakcijom s korisnikom, dok poslužitelj (backend) upravlja poslovnom logikom i komunikacijom s bazom podataka.

Baza podataka: PostgreSQL relacijska baza podataka pohranjuje sve podatke o ocjenjivačima, restoranima, proizvodima, grupama i ocjenama te omogućava efikasno izvršavanje upita za kreiranje rang lista na temelju ocjena. Baza je ključna za čuvanje podataka potrebnih za ispravan rad aplikacije.

Korisničko sučelje aplikacije (frontend sloj) omogućava interakciju putem preglednika, čime je aplikacija dostupna bez potrebe za dodatnim softverom na korisnikovom uređaju.

Organizacija aplikacije

Aplikacija se sastoji od tri glavne komponente: frontend-a, backend-a i baze podataka. Frontend sloj aplikacije ostvaren je Javascript bibliotekom React koja omogućava izradu dinamičkih interaktivnih web aplikacija. Backend sloj ostvaren je razvojnim okvirom Spring boot u programsom jeziku Java te je zadužen za obradu zahtjeva koje šalje frontend te za statičko posluživanje datoteka. Baza podataka nalazi se na poslužitelju Supabase. Ostvarena je kao relacijska baza podataka pomoću tehnologije PostgreSQL. Interakcija između slojeva odvija se tako što korisnik putem preglednika šalje zahtjev na Heroku server. Ako zahtjev traži frontend resurse, Heroku poslužuje React aplikaciju. React aplikacija šalje HTTP zahtjeve backendu putem REST API-ja za podatke ili akcije koje nisu dostupne na klijentu. Backend obrađuje zahtjev, komunicira s bazom podataka na Supabaseu, te vraća rezultat frontend aplikaciji. Frontend ažurira korisničko sučelje na temelju odgovora. Za odvajanje logike korisničkog sučelja, poslovne logike i podataka koristi se MVC arhitektura. Model obuhvaća entitete poput Restoran, Grupa, Proizvod, koji predstavljaju podatke iz baze. Controller čine Spring Boot kontroleri, a oni obrađuju HTTP zahtjeve iz frontenda, pozivaju odgovarajuće servise i vraćaju odgovore korisnicima. View se sastoji od frontend dijela aplikacije implementiranog u Reactu, čija je uloga prezentacija podataka korisnicima kroz interaktivno korisničko sučelje te prikaz odgovora backend servisa i poruka o greškama. Ovaj pristup omogućuje jasno razdvajanje odgovornosti između frontend i backend slojeva, dok MVC arhitektura osigurava čistoću koda i olakšava razvoj i održavanje.

Baza podataka

Za potrebe našeg sustava odlučili smo koristiti relacijsku bazu podataka i sustav PostgreSQL. Relacijske baze podataka koriste relacijski model za strukturiranje i upravljanje podatcima pri čemu se podaci pohranjuju u tablicama sastavljenim od redaka i stupaca, organizirani u relacijama. Relacijske baze omogućavaju jednostavnu pohranu, umetanje, izmjenu i dohvata podataka za daljnju obradu, osiguravajući konzistentnost i integritet podataka. PostgreSQL, kao moćan i popularan open-source sustav, idealan je za upravljanje takvom bazom. Baza podataka naše aplikacije sastoji se od nekoliko tablica koje organiziraju sve potrebne podatke kroz relacije i atribute.

Baza podataka naše aplikacije sastoji se od sljedećih tablica: * KORISNIK * ADMINISTRATOR * OCJENJIVAČ * GRUPA * OCJENA * PROIZVOD * RESTORAN * LISTA PROIZVODA

Opis tablica

Korisnik

Entitet KORISNIK opisuje korisnika i sadrži najbitnije informacije o njemu. Korisnik može biti ili administrator ili ocjenjivač ili restoran.

Ime tablice: KORISNIK

Atribut	Tip podatka	Opis varijable
korisnikId	int	Jedinstveni identifikator
korisnickoIme	varchar	Jedinstveno korisničko ime korisnika
lozinka	varchar	Lozinka za prijavu korisnika

Administrator

Entitet ADMINISTRATOR sadrži sve informacije o korisnicima s administrativnim ovlastima unutar sustava. Administrator ima hijerarhijsku vezu s entitetom korisnik, odnosno administrator je podtip korisnika.

Ime tablice: ADMINISTRATOR

Atribut	Tip podatka	Opis varijable
korisnikId	int	Jedinstveni identifikator administratora

Ocenjivač

Entitet OCJENJIVAČ sadrži sve informacije o ocjenjivačima uključujući ime, prezime i E – mail ocjenjivača. Povezan je sa vezom Many- to-many sa entitetom Grupa preko korisnikId i idGrupa, Many-to-one sa entitetom Ocjena preko korisnikId.

Ime tablice: OCJENJIVAČ

Atribut	Tip podatka	Opis varijable
korisnikId	int	Jedinstveni identifikator ocjenjivača
ime	varchar	Ime ocjenjivača
prezime	varchar	Prezime ocjenjivača
eMail	varchar	Jedinstvena email adresa ocjenjivača
brOcjene	int	Ocjena koju je ocjenjivač dao za proizvod

Grupa

Entitet GRUPA sadrži osnovne informacije o grupi, uključujući kod grupe i kategoriju grupe. Povezan je sa vezom Many-to-many sa Ocjenjivač preko korisnikId

i idGrupa. Vezom One-to-One povezan je s entitetom LISTA PROIZVODA preko listaProizvodaId

Ime tablice: GRUPA

Atribut	Tip podatka	Opis varijable
idGrupa	int	Identifikator grupe
kodGrupe	int	Jedinstveni kod za ulaz u grupu
kategorijaGrupa	varchar	Kategorija grupe
lokacijaGrupa	varchar	Lokacija grupe
listaProizvodaId	int	Identifikator liste proizvoda

Ocjena

Entitet OCJENA sadrži informaciju o ocjeni kojom su korisnici ocijenili proizvod. Povezan je sa vezom Many-to-many sa Proizvod preko brOcjene i NazivProizvoda, One-to-many sa entitetom Ocjenjivač preko korisnikId.

Ime tablice: OCJENA

Atribut	Tip podatka	Opis varijable
brOcjene	int	Ocjena koju je ocjenjivač dao za proizvod

Proizvod

Entitet PROIZVOD sadrži osnovne informacije o proizvodu, uključujući naziv proizvoda, kategoriju proizvoda i cijenu. Povezan je sa vezom Many-to-many sa Restoran preko KorisnikId i NazivProizvoda. Vezom Many-to-Many povezan je s entitetom LISTA PROIZVODA preko listaProizvodaId i nazivProizvodaId, a vezom Many-to-Many povezan je s entitetom OCJENA preko brOcjene i nazivProizvoda.

Ime tablice: PROIZVOD

Atribut	Tip podatka	Opis varijable
kategorijaProizvoda	varchar	Kategorija proizvoda
nazivProizvoda	varchar	Ime proizvoda
restoran	varchar	Ime restorana u kojem je taj proizvod
cijena	numeric	Cijena proizvoda
poveznicaSlike	varchar	Slika proizvoda

Restoran

Entitet RESTORAN sadrži osnovne informacije o restoranu, uključujući naziv restorana, lokaciju, radno vrijeme itd... Povezan je sa vezom Many-to-many sa preko KorisnikId i NazivProizvoda.

Ime tablice: RESTORAN

Atribut	Tip podatka	Opis varijable
korisnikId	int	Jedinstveni identifikator restorana
brojTelefona	int	Jedinstveni kontakt broj restorana
cijenovniRang	Numeric(10,2)	Raspon cijena u restoranu
radnoVrijeme	varchar	Radno vrijeme restorana
nazivRestorana	varchar	Naziv restorana
lokacija	varchar	Lokacija restorana
povezniceRestoran	varchar	Poveznica na stranicu restorana

Lista proizvoda

Entitet LISTA PROIZVODA sadrži atribut ListaProizvodaId. Povezan je vezom One-to-One sa entitetom GRUPA preko ListaProizvodaId, te vezom Many-to-Many sa entitetom PROIZVOD preko nazivProizvoda i listaProizvoda.

Ime tablice: LISTA PROIZVODA

Atribut	Tip podatka	Opis varijable
listaProizvodaId	int	Identifikator liste proizvoda

Dijagram baze podataka

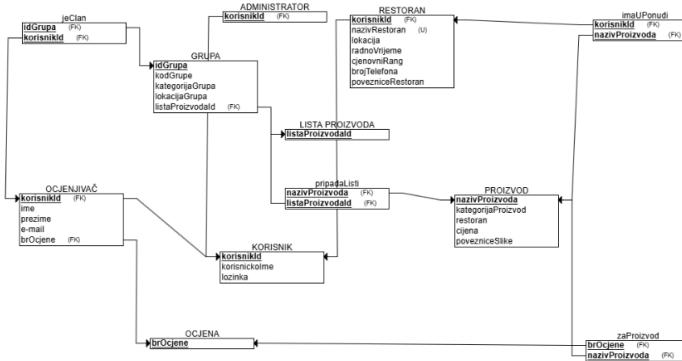


Figure 8: Relacijska shema baze podataka

Slika 4.1: Prikaz dijagrama baze podataka

Dijagram razreda

Dijagram razreda prikazuje glavne klase sustava i njihove međusobne odnose. Klasa Korisnik predstavlja osnovnu klasu za korisnike, s metodama za registraciju, prijavu i odjavu. Iz nje nasljeđuju specifične klase korisnika: Ocjenjivač, Restoran i Administrator. Klasa Ocjenjivač omogućuje pretraživanje restorana, pregled rangova i ocjenjivanje proizvoda. Klasa Restoran omogućuje upravljanje proizvodima putem metoda za dodavanje, uklanjanje i ažuriranje informacija o proizvodima. Klasa Administrator dodaje metode za administrativne aktivnosti, kao što su blokiranje i deblokiranje korisnika te verifikacija restorana. Dodatno, klasa Proizvod sadrži osnovne atribute naziv i cijena te je povezana s klasom Restoran, što označava kojem restoranu proizvod pripada. Klasa Grupa sadrži listu članova i povezana je s klasom ListaJela, koja predstavlja kolekciju proizvoda koje ocjenjuje određena grupa korisnika. Ova struktura omogućuje jednostavno proširivanje i upravljanje korisnicima, restoranim, proizvodima i grupama unutar sustava.

Backend aplikacije izgrađen je uz korištenje radnog okvira Spring Boot, što rezultira višeslojnim arhitekturnim pristupom.

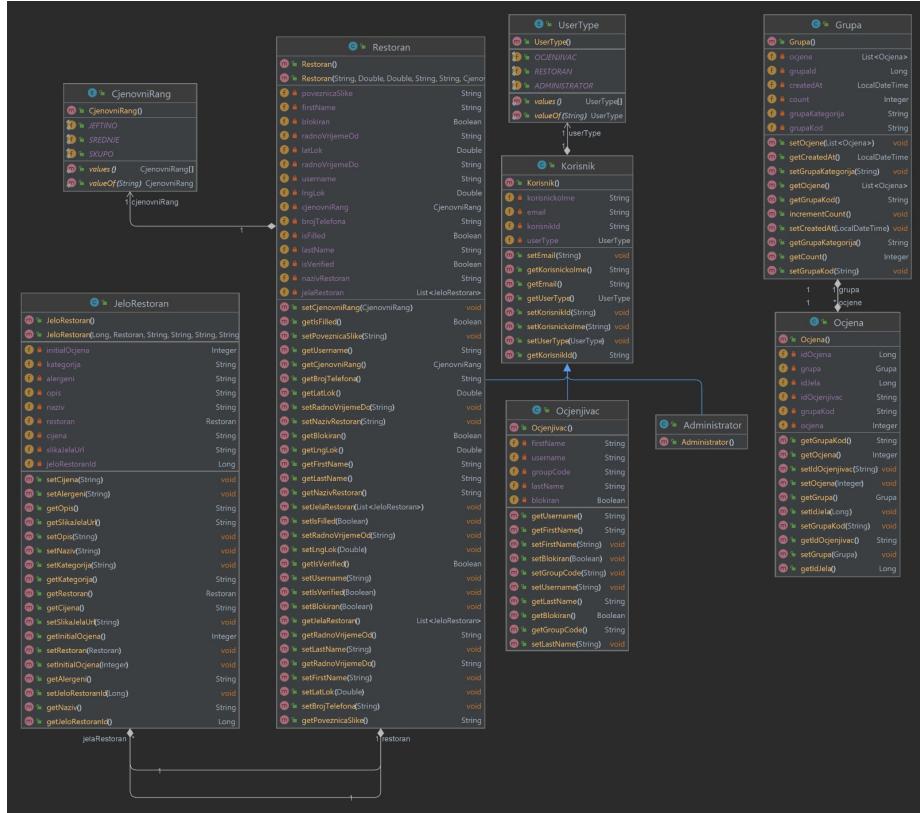
Na slici 4.2 prikazana je struktura entiteta i njihovih međusobnih odnosa unutar aplikacije. Svaki entitet predstavlja specifičnu domenu podataka, poput korisnika, restorana, ocjena i grupe. Entiteti su strukturirani tako da sadrže osnovne članske varijable te pripadajuće metode poput gettera i settera, dok složenija logika ostaje izvan njih. Neki entiteti, poput "Korisnik" i "Restoran", uključuju dodatne metode za specifične funkcionalnosti, dok enumeracije kao "CjenovniRang" i "UserType" pružaju jasno definirane vrijednosti za određena polja.

Sloj Repository, prikazan na slici 4.3, sadrži razrede definirane kao sučelja. Ova sučelja pružaju metode za dohvati podataka iz baze te za izvođenje izmjena.

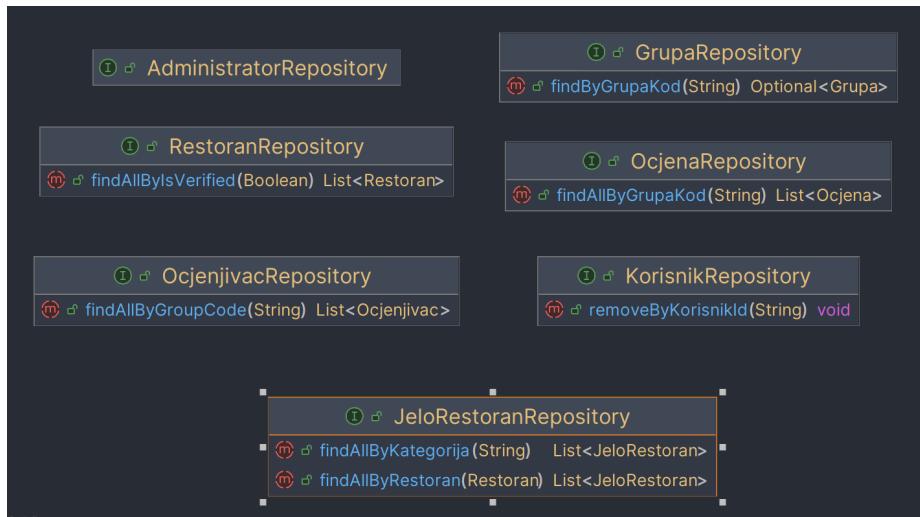
Centralnu logiku aplikacije sadrži sloj Service, prikazan na slici 4.4. Ovaj sloj upravlja entitetima i poziva metode iz sloja Repository kako bi obavio potrebne operacije.

Najniži sloj u hijerarhiji je sloj Controller, prikazan na slici 4.5. Ovaj sloj služi kao poveznica između frontenda i backenda. Njegova uloga je obrada HTTP zahtjeva, pozivanje metoda iz sloja Service te slanje odgovora na obrađene zahtjeve.

Na slici 4.6 prikazan je pregled različitih zahtjeva koji se koriste u aplikaciji. Svaki od njih sadrži specifične atribute prilagođene funkcionalnostima aplikacije. Na primjer, zahtjevi za kreiranje korisnika, restorana ili jela definiraju podatke potrebne za njihovu inicijalizaciju, dok DTO objekti omogućuju prijenos podataka između slojeva aplikacije. Ovi zahtjevi olakšavaju manipulaciju podacima i osiguravaju konzistentnost u komunikaciji između različitih dijelova sustava.



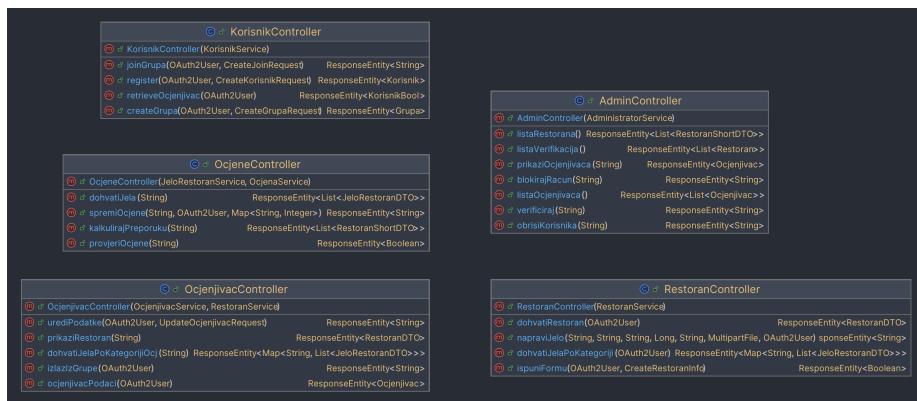
Slika 4.2: Dijagram razreda - modeli



Slika 4.3: Dijagram razreda - sloj Repository



Slika 4.4: Dijagram razreda - sloj Service



Slika 4.5: Dijagram razreda - sloj Controller

CreateKorisnikRequest	UpdateOcenjivacRequest
(@ CreateKorisnikRequest)	(@ UpdateOcenjivacRequest)
@ username String	@ firstName String
@ userType UserTipe	@ lastName String
@ lastName String	@ username String
@ firstName String	@ firstName String
@ firstName String	@ lastName String
@ lastName String	@ username String
@ username String	@ userType String
@ userType UserTipe	
CreateRestoranInfo	JeloRestoranDTO
(@ CreateRestoranInfo)	(@ JeloRestoranDTO)
@ nazivRestoran String	@ opisJela String
@ brTelefon String	@ jeloRestoranId Long
@ latLok Double	@ imageSrc String
@ doVrijeme String	@ cijena String
@ odVrijeme String	@ kategorija String
@ IngLok Double	@ nazivJela String
@ link String	@ alergeni String
@ latLok Double	@ alergeni String
@ brTelefon String	@ kategorija String
@ nazivRestoran String	@ cijena String
@ doVrijeme String	@ imageSrc String
@ link String	@ jeloRestoranId Long
@ IngLok Double	@ opisJela String
@ odVrijeme String	@ nazivJela String
CreateJeloRestoranRequest	KorisnikBool
(@ CreateJeloRestoranRequest)	(@ KorisnikBool)
(@ CreateJeloRestoranRequest, String, String, String, Long, String)	
@ opis String	@ isVerified Boolean
@ alergeni String	@ isFilled Boolean
@ kategorija String	@ ocjenjivacId String
@ naziv String	@ ocjenjivacIme String
@ cijena Long	@ userType UserTipe
@ imageSrc MultipartFile	@ isRegistered boolean
@ alergeni String	@ blokiran Boolean
@ naziv String	@ email String
@ opis String	(@ getIsRegistered() boolean)
@ imageSrc MultipartFile	(@ setIsRegistered(boolean) void)
@ kategorija String	@ ocjenjivacId String
@ cijena Long	@ blokiran Boolean
RestoranDTO	
(@ RestoranDTO)	(@ RestoranDTO)
@ nazivRestoran String	@ nazivRestoran String
@ radnoVrijemeDo String	@ radnoVrijemeOd String
@ radnoVrijemeOd String	@ poveznicaSlike String
@ poveznicaSlike String	@ latLok Double
@ cijenoviRang CijenoviRang	@ cijenoviRang CijenoviRang
@ firstName String	@ firstName String
@ IngLok Double	@ IngLok Double
@ brojTelefona String	@ brojTelefona String
@ IngLok Double	@ id String
@ id String	@ username String
@ lastName String	@ lastName String
@ latLok Double	@ latLok Double
@ filled Boolean	@ filled Boolean
@ brojTelefona String	@ id String
@ IngLok Double	@ lastName String
@ id String	@ nazivRestoran String
@ lastName String	@ radnoVrijemeOd String
@ verified Boolean	@ verified Boolean
@ firstName String	@ username String
@ username String	@ cijenoviRang CijenoviRang
@ poveznicaSlike String	@ poveznicaSlike String
@ radnoVrijemeDo String	@ radnoVrijemeDo String
RestoranShortDTO	
(@ RestoranShortDTO)	(@ RestoranShortDTO)
@ nazivRestoran String	@ nazivRestoran String
@ korisnikId String	@ korisnikId String
@ ocjena Double	@ ocjena Double
@ blokiran Boolean	@ blokiran Boolean
@ ocjena Double	@ nazivRestoran String
@ nazivRestoran String	@ korisnikId String

Slika 4.6: Dijagram razreda - Requests

Dinamičko ponašanje aplikacije

UML dijagrami stanja

Dijagram stanja prikazuje stanja objekta i događaje koji omogućavaju prijelaze iz jednog stanja u drugo. Na slici je prikazan dijagram stanja za prijavljenog ocjenjivača. Nakon prijave se ocjenjivaču prikazuje početna stranica na kojoj je ponuđeno više mogućnosti za nastavak korištenja aplikacije. Ima mogućnost otvaranja profila korisnika. Na stranici svojeg korisničkog profila ima mogućnost odjave, brisanja računa i uređivanja osobnih podataka (korisničko ime, ime, prezime) nakon čega odabirom opcije „Spremi promjene“ se vraća na stranicu profila. Odabirom opcije „Natrag“ vraća se na početnu stranicu. Druga mogućnost je unos koda grupe kojoj se ocjenjivač želi pridružiti u za to zadani okvir. Prikazuje im se stranica za ocjenjivanje proizvoda. Svakom proizvodu mora se dodijeliti ocjena. Klikom na gumb „Završi“ ocjenjivaču se prikazuje lista preporučenih restorana. Svaki restoran ima opciju „Posjeti restoran“ koja ocjenjivača vodi na stranicu restorana gdje može vidjeti više detalja o restoranu. Odabirom opcije „Stvori novu grupu“ na početnoj stranici, ocjenjivač može odabrat kategoriju proizvoda koja se ocjenjuje za novu grupu nakon čega se otvara stranica za ocjenjivanje proizvoda.

Slika 4.7: Dijagram stanja

UML dijagrami aktivnosti

Ovaj dijagram aktivnosti prikazuje tijek pridruživanja grupi putem web stranice koristeći Google OAuth2 za autentifikaciju, pri čemu se podaci korisnika i grupe pohranjuju i provjeravaju u bazi podataka. Dijagram je podijeljen u više “swimlane-ova” koji predstavljaju različite aktore u procesu: ocjenjivača (korisnika), web stranicu, Google OAuth2 i bazu podataka.

Glavni koraci procesa:

Odabir opcije za prijavu: Korisnik na web stranici odabire opciju prijave putem Google OAuth2, nakon čega ga web stranica preusmjerava na servis za autentifikaciju.

Autentifikacija: Korisnik unosi svoje podatke za prijavu na Google OAuth2:

* **Ako su podaci neispravni:** Google OAuth2 vraća obavijest o pogrešci web stranici, koja korisniku prikazuje poruku o pogrešnim podacima. Korisnik tada ima mogućnost ponoviti prijavu unosom ispravnih podataka.

* **Ako su podaci ispravni:** Google OAuth2 vraća pozitivan ishod web stranici, čime se omogućuje nastavak procesa.

Kreiranje korisničkog računa: Ako su podaci za autentifikaciju ispravni, web stranica kreira korisnički račun.

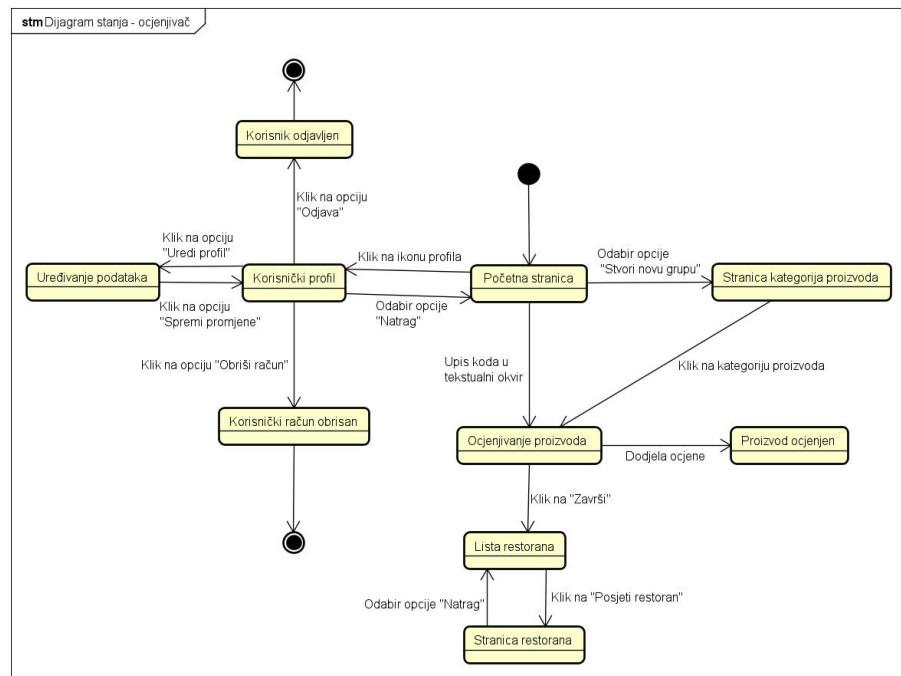


Figure 9: UML dijagram stanja

Unos podataka: Korisnik unosi podatke potrebne za kompletiranje profila te se oni pohranjuju u bazu podataka.

Unos koda grupe: Korisnik unosi kod grupe, koji web stranica šalje bazi podataka na provjeru.

Provjera koda grupe: Baza podataka provjerava ispravnost unesenog koda i vraća rezultat web stranici:

* **Ako je kod ispravan:** Korisnik se povezuje s grupom u bazi podataka, a otvara mu se stranica grupe.

* **Ako je kod neispravan:** Web stranica korisniku prikazuje poruku o pogrešci te omogućuje ponovni unos koda.

Pohrana povezanosti: U slučaju ispravnog koda, baza podataka pohranjuje povezanost korisnika s odgovarajućom grupom.

Cijeli proces osigurava siguran i pouzdan postupak autentifikacije, validacije podataka i povezivanja korisnika s grupom. Svaka faza uključuje odgovarajuću provjeru i obradu podataka kako bi se zaštitala sigurnost sustava i osigurao ispravan tijek aktivnosti.

Slika 4.8: Dijagram aktivnosti - Pridruživanje grupi

Dijagram komponenata

Dijagram komponenti prikazuje arhitekturu sustava i način na koji se komponente međusobno povezuju. Na slici 5.1 prikazan je UML dijagram komponenti sustava. Na njemu je vidljivo da se komunikacija s aplikacijom ostvaruje iz vanjske komponente web preglednika putem sučelja HTTP request. Aplikacija se sastoji od dvije glavne unutarnje komponente, Frontend i Backend, koje međusobno komuniciraju preko sučelja REST API.

Frontend aplikacije izgrađen je pomoću React biblioteke i sastoji se od komponente React view, koja je zadužena za prikaz korisničkog sučelja, React routera, koji upravlja navigacijom, i komponente App.js, koja integrira sve elemente frontenda. Komunikacija s vanjskom komponentom Google auth service ostvaruje se putem sučelja Google API.

Backend aplikacije implementiran je koristeći Spring okvir i sastoji se od nekoliko komponenti. Komponenta Rest prima zahtjeve preko sučelja REST API te ih proslijedi unutar sustava. Logika je implementirana unutar komponente Backend logic, koja se povezuje s repozitorijskom komponentom putem JPA sučelja. Komponenta Repository koristi JPA za interakciju s bazom podataka, dok je domena aplikacije definirana kroz komponentu Domain. Backend se putem sučelja PostgreSQL povezuje s bazom podataka radi pohrane i dohvatanja podataka.

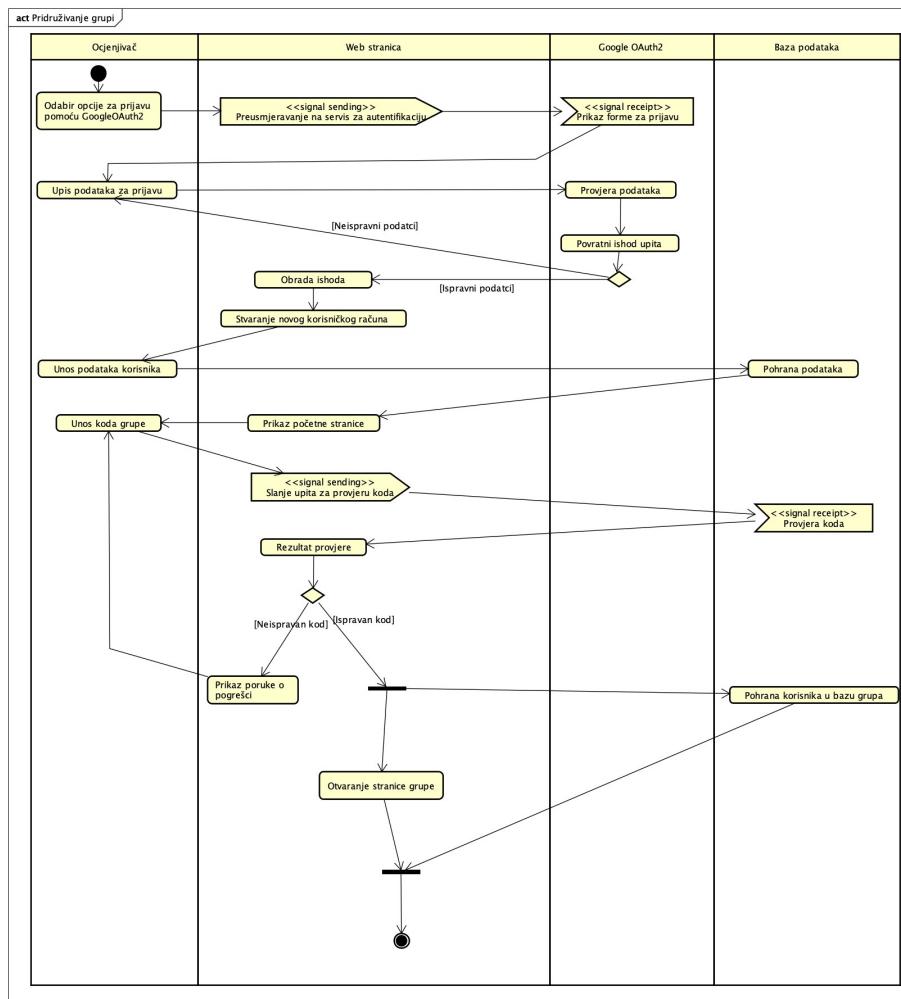
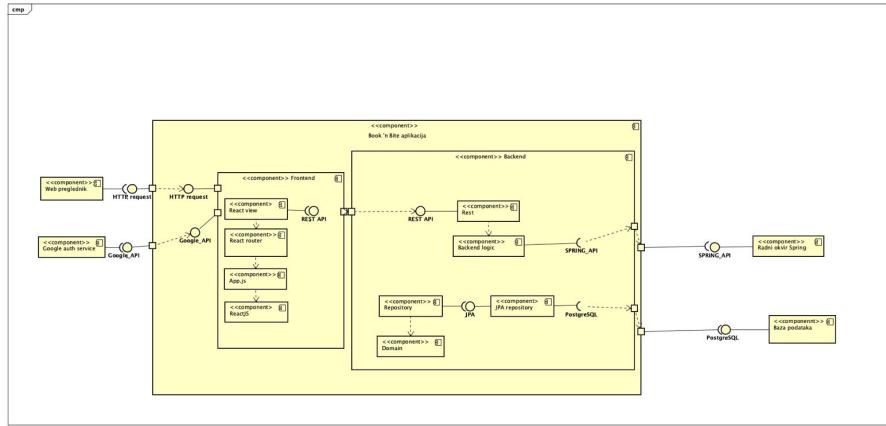


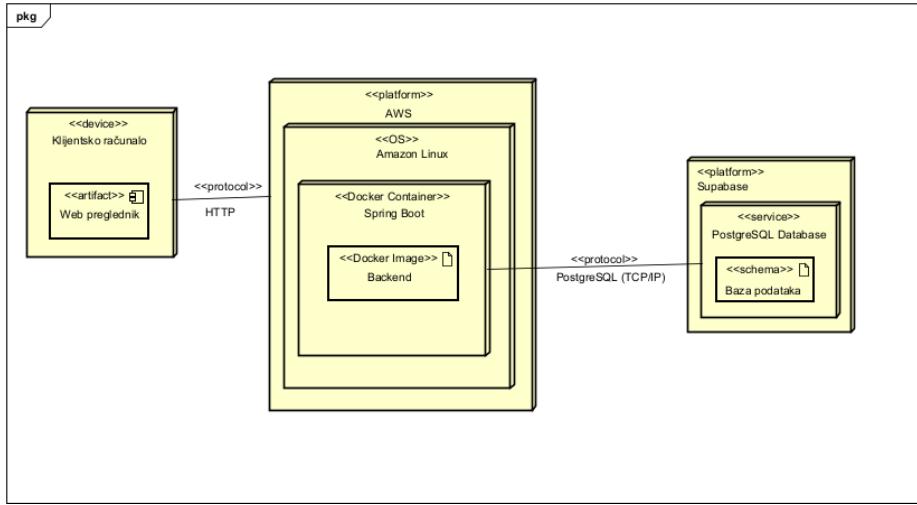
Figure 10: UML dijagram aktivnosti



Slika 5.1: Dijagram komponenata

Dijagram razmještaja

Dijagram razmještaja (engl. UML Deployment Diagram) predstavlja statički strukturni prikaz koji opisuje topologiju sustava, s posebnim naglaskom na povezanost hardverskih i softverskih elemenata. Njegova svrha je prikazati kako se sustav realizira kroz kod i podatke koji se nalaze i izvršavaju na različitim računalnim resursima. Ovakav dijagram pruža uvid u hardverske komponente, komunikacijske veze te način na koji se softverski artefakti raspoređuju i izvode unutar sustava. U ovom slučaju, dijagram prikazuje kako klijent putem web preglednika koristi HTTP protokol za komunikaciju s backend aplikacijom implementiranom u Docker kontejneru, koji se izvršava na Amazon Linux OS-u unutar AWS platforme. Backend aplikacija koristi PostgreSQL bazu podataka, smještenu na Supabase platformi, pri čemu komunikacija između aplikacije i baze podataka koristi TCP/IP protokol.



Slika 5.2: Dijagram razmještaja

Ispitivanje komponenti

Ispitivanje komponenti ostvareno je pomoću JUnit5 alata za ispitivanje.

Ispitni slučaj 1: Test provjere svojstva grupe

Prvi test:

- **Opis funkcionalnosti:** Testira inicijalno stanje objekta Grupa, provjerava da svojstvo grupaKod nije postavljeno tj. da je vrijednost `null`.
- **Ulazni podaci:** Nema ulaznih podataka, samo je inicijalizacija objekta Grupa.
- **Očekivani rezultat:** Metoda `getGrupaKod()` vraća `null`.
- **Rezultat:** Prolaz ispitivanja.
- **Postupak:**
 1. Inicijalizacija objekta Grupa bez postavljanja atributa.
 2. Poziv metode `getGrupaKod()` i usporedba povratne vrijednosti s `null` pomoću metode `assertNull()`.

Drugi test:

- **Opis funkcionalnosti:** Test postavljanja i dohvaćanja vrijednosti za svojstva grupaKod i grupaKategorija.
- **Ulazni podaci:**
 - grupaKod = "LD6658"
 - grupaKategorija = "Roštilj"
- **Očekivani rezultat:**
 - Metoda `getGrupaKod()` vraća "LD6658".
 - Metoda `getGrupaKategorija()` vraća "Roštilj".
- **Rezultat:** Prolaz ispitivanja.

- **Postupak:**

1. Inicijalizacija objekta **Grupa**.
2. Postavljanje vrijednosti za svojstva **grupaKod** i **grupaKategorija**.
3. Usporedba povratnih vrijednosti metoda **getGrupaKod()** i **getGrupaKategorija()** s ulaznim podacima koristeći metodu **assertEquals()**.

```
@Test
void testGrupaPropertiesNull() {
    Grupa grupa = new Grupa();

    assertNull(grupa.getGrupaKod());
}

@Test
void testGrupaProperties() {
    Grupa grupa = new Grupa();
    grupa.setGrupaKod("LD6658");
    grupa.setGrupaKategorija("Roštilj");
    assertEquals( expected: "LD6658", grupa.getGrupaKod());
    assertEquals( expected: "Roštilj", grupa.getGrupaKategorija());
}
```

Slika 6.1: Test provjere svojstva grupe

Ispitni slučaj 2: Test provjere svojstva jela

- **Opis funkcionalnosti:** Test postavljanja i dohvaćanja vrijednosti za svojstva **jeloRestoranId**, **restoran** (Restoran), **naziv**, **opis**, **kategorija**, **cijena**, **alergeni**, **slikaJelaUrl**.
- **Ulazni podaci:**
 - **jeloRestoranId** = 1
 - Restoran **restoranExample**
 - **naziv** = "Pizza Example"
 - **opis** = "Opis pizze"
 - **kategorija** = "obicni"
 - **cijena** = "10 Eur"
 - **alergeni** = "Gluten, Laktoza"
 - **slikaJelaUrl** = "ExampleUrl"
- **Očekivani rezultat:** Metode za dohvata podataka vraćaju ispravne vrijednosti; metoda **getJeloRestoranId()** vraća 1, metoda **getRestoran()** vraća **restoranExample**, metoda **getNaziv()** vraća „Pizza Example“, metoda **getOpis()** vraća „Opis pizze“, metoda **getKategorija()** vraća „obicni“, metoda **getCijena()** vraća „10 Eur“, metoda **getAlergeni()** vraća „Gluten,

- Laktoza“, metoda `getSlikaJelaUrl()` vraća „ExampleUrl“
- **Rezultat:** Prolaz ispitivanja.
 - **Postupak:**
 1. Inicijalizacija objekta `Restoran` i `jeloRestoran`.
 2. Postavljanje vrijednosti za svojstva objekta `jeloRestoran`.
 3. Usporedba povratnih vrijednosti `get` metoda s ulaznim podacima koristeći metodu `assertEquals()`.

```

@Test
void testJeloProperties() {
    Restoran restoranExample = new Restoran();
    JeloRestoran jeloRestoran = new JeloRestoran(
        jeloRestoranId: 1L,
        restoranExample,
        naziv: "Pizza Example",
        opis: "Opis pizze",
        kategorija: "obicni",
        cijena: "10 Eur",
        alergeni: "Gluten, Laktoza",
        slikaJelaUrl: "ExampleUrl"
    );

    assertEquals(Long.valueOf(1), jeloRestoran.getJeloRestoranId());
    assertEquals(restoranExample, jeloRestoran.getRestoran());
    assertEquals(expected: "Pizza Example", jeloRestoran.getNaziv());
    assertEquals(expected: "Opis pizze", jeloRestoran.getOpis());
    assertEquals(expected: "obicni", jeloRestoran.getKategorija());
    assertEquals(expected: "10 Eur", jeloRestoran.getCijena());
    assertEquals(expected: "Gluten, Laktoza", jeloRestoran.getAlergeni());
    assertEquals(expected: "ExampleUrl", jeloRestoran.getSlikaJelaUrl());
}

}

```

Slika 6.2: Test provjere svojstva jela

Ispitni slučaj 3: Test provjere svojstva ocjene

Prvi test:

- **Opis funkcionalnosti:** Testira inicijalno stanje objekta `Ocjena`, provjerava da svojstvo `grupaKod` nije postavljeno tj. da je vrijednost `null`.
- **Ulazni podaci:** Nema ulaznih podataka, samo je inicijalizacija objekta `Grupa`.
- **Očekivani rezultat:** Metoda `getGrupaKod()` vraća `null`.
- **Rezultat:** Prolaz ispitivanja.

- **Postupak:**

1. Inicijalizacija objekta **Grupa** bez postavljanja atributa.
2. Poziv metode `getGrupaKod()` i usporedba povratne vrijednosti s `null` pomoću metode `assertNull()`.

Drugi test:

- **Opis funkcionalnosti:** Test postavljanja i dohvaćanja vrijednosti za svojstva `ocjena`, `idJela`, `idOcenjivac`, `grupa` (`Grupa`), `grupaKod`.
- **Ulazni podaci:**
 - `ocjena = 3`
 - `idJela = 111`
 - `idOcenjivac = "001"`
 - `Grupa grupa`
 - `grupaKod = "LD6658"`
- **Očekivani rezultat:** Povratne vrijednosti metoda odgovaraju ulaznim podacima; metoda `getOcjena()` vraća 3, metoda `getIdJela()` vraća 111, metoda `getIdOcenjivac()` vraća „001“, metoda `getGrupa()` vraća `grupa`, metoda `getGrupaKod()` vraća „LD6658“
- **Rezultat:** Prolaz ispitivanja.
- **Postupak:**
 1. Inicijalizacija objekata `Ocjena` i `Grupa`.
 2. Postavljanje vrijednosti za svojstva.
 3. Usporedba povratnih vrijednosti metoda `getGrupaKod()`, `getOcjena()`, `getIdJela()`, `getIdOcenjivac()`, `getGrupa()` s ulaznim podacima koristeći metodu `assertEquals()`.

```

@Test
void testOcjenaPropertiesNull() {
    Ocjena ocjena = new Ocjena();
    ocjena.setOcjena(3);
    assertNull(ocjena.getGrupaKod());
}

@Test
void testOcjenaProperties() {
    Ocjena ocjena = new Ocjena();
    Grupa grupa = new Grupa();
    ocjena.setOcjena(3);
    ocjena.setIdJela(111L);
    ocjena.setIdOcjenjivac("001");
    ocjena.setGrupa(grupa);
    ocjena.setGrupaKod("LD6658");

    assertEquals( expected: 3, ocjena.getOcjena());
    assertEquals( expected: 111L, ocjena.getIdJela());
    assertEquals( expected: "001", ocjena.getIdOcjenjivac());
    assertEquals(grupa, ocjena.getGrupa());
    assertEquals( expected: "LD6658", ocjena.getGrupaKod());
}

```

Slika 6.3: Test ocjena

Ispitni slučaj 4: Test provjere svojstva ocjenjivača

- **Opis funkcionalnosti:** Test postavljanja i dohvaćanja vrijednosti za svojstva username, firstName, lastName, email, korisnikId, UserType.
- **Ulazni podaci:**
 - username = "testUser"
 - firstName = "Test"
 - lastName = "User"
 - email = "test@test.com"
 - korisnikId = "001"
 - UserType.OCJENJIVAC
- **Očekivani rezultat:**
 - Metoda getUsername() vraća "testUser".

- Metoda `getFirstName()` vraća "Test".
- Metoda `getLastName()` vraća "User".
- Metoda `getEmail()` vraća "test@test.com".
- Metoda `getKorisnikId()` vraća "001".
- Metoda `getUserType()` vraća `UserType.OCJENJIVAC`.
- **Rezultat:** Prolaz ispitivanja.
- **Postupak:**
 1. Inicijalizacija objekta `Ocjjenjivac`.
 2. Postavljanje vrijednosti za svojstva `username`, `firstName`, `lastName`, `email`, `korisnikId`, `UserType`.
 3. Usporedba povratnih vrijednosti metoda `getUsername()`, `getFirstName()`, `getLastName()`, `getEmail()`, `getKorisnikId()`, `getUserType()` s ulaznim podacima koristeći metodu `assertEquals()`.

```
@Test
void testOcjjenjivacProperties() {
    Ocjjenjivac ocjenjivac = new Ocjjenjivac();
    ocjenjivac.setUsername("testUser");
    ocjenjivac.setFirstName("Test");
    ocjenjivac.setLastName("User");
    ocjenjivac.setEmail("test@test.com");
    ocjenjivac.setKorisnikId("001");
    ocjenjivac.setUserType(UserType.OCJENJIVAC);

    assertEquals(ocjenjivac.getUsername(), actual: "testUser");
    assertEquals(ocjenjivac.getFirstName(), actual: "Test");
    assertEquals(ocjenjivac.getLastName(), actual: "User");
    assertEquals(ocjenjivac.getEmail(), actual: "test@test.com");
    assertEquals(ocjenjivac.getKorisnikId(), actual: "001");
    assertEquals(ocjenjivac.getUserType(), UserType.OCJENJIVAC);
}
```

Slika 6.4: Test ocjenjivač

Ispitni slučaj 5: Test provjere svojstva restorana

- **Opis funkcionalnosti:** Test postavljanja i dohvaćanja vrijednosti za svojstva `nazivRestoran`, `latLok`, `lngLok`, `radnoVrijemeOd`, `radnoVrijemeDo`, `cjenovniRang`, `brojTelefona`, `poveznicaSlike`, `username`, `firstName`, `lastName`, `email`, `isVerified`, `isFilled`.
- **Ulazni podaci:**
 - `nazivRestoran` = "Test"
 - `latLok` = 45.00
 - `lngLok` = 16.00

- radnoVrijemeOd = "10:00"
- radnoVrijemeDo = "00:00"
- cjenovniRang = cjenovniRang.JEFTINO
- brojTelefona = "01 234 567"
- poveznicaSlike = "example.jpg"
- username = "testRestoran"
- firstName = "Test"
- lastName = "Restoran"
- email = "testRestoran@gmail.com"
- isVerified = false
- isFilled = false

- **Očekivani rezultat:**

- Metode getNazivRestoran(), getLatLok(), getLngLok(), getRadnoVrijemeOd(), getRadnoVrijemeDo(), getCjenovniRang(), getBrojTelefona(), getPoveznicaSlike(), getUsername(), getFirstName(), getLastname(), getEmail() vraćaju ispravne vrijednosti.

- **Rezultat:** Prolaz ispitivanja.

- **Postupak:**

1. Inicijalizacija objekta Ocjenjivac.
2. Postavljanje vrijednosti za svojstva.
3. Usporedba povratnih vrijednosti metoda s ulaznim podacima koristeći metodu assertEquals().

```

@Test
void testRestoran() {
    Restoran restoran = new Restoran();
    restoran.setNazivRestoran("Test");
    restoran.setLatLok(45.00);
    restoran.setLngLok(16.00);
    restoran.setRadnoVrijemeOd("10:00");
    restoran.setRadnoVrijemeDo("00:00");
    restoran.setCjenovniRang(CjenovniRang.JEFTINO);
    restoran.setBrojTelefona("01 234 567");
    restoran.setPoveznicaSlike("example.jpg");
    restoran.setUsername("testRestoran");
    restoran.setFirstName("Test");
    restoran.setLastName("Restoran");
    restoran.setEmail("testRestoran@gmail.com");
    restoran.setIsVerified(false);
    restoran.setIsFilled(false);
    assertEquals( expected: "Test", restoran.getNazivRestoran());
    assertEquals( expected: 45.00, restoran.getLatLok());
    assertEquals( expected: 16.00, restoran.getLngLok());
    assertEquals( expected: "10:00", restoran.getRadnoVrijemeOd());
    assertEquals( expected: "00:00", restoran.getRadnoVrijemeDo());
    assertEquals(CjenovniRang.JEFTINO, restoran.getCjenovniRang());
    assertEquals( expected: "01 234 567", restoran.getBrojTelefona());
    assertEquals( expected: "example.jpg", restoran.getPoveznicaSlike());
    assertEquals( expected: false, restoran.getIsVerified());
    assertEquals( expected: false, restoran.getIsFilled());
    assertEquals( expected: "testRestoran", restoran.getUsername());
    assertEquals( expected: "Test", restoran.getFirstName());
    assertEquals( expected: "Restoran", restoran.getLastName());
    assertEquals( expected: "testRestoran@gmail.com", restoran.getEmail());
}

```

Slika 6.5: Test restoran

Ispitni slučaj 6: Test provjere funkcionalnosti verifikacije restorana

- **Opis funkcionalnosti:** Test ispituje funkciju `verificiraj`.
- **Ulazni podaci:**
 - Mock objekt `restoranRepository`.
 - Objekt `AdministratorServiceImpl` s injektiranim `administratorService` pomoću `@InjectMocks`.
 - Objekt `Restoran`.
- **Očekivani rezultat:** Funkcija `verificiraj("1")` vraća string “Restoran je verificiran.” i postavlja varijablu `isVerified` na `true`.
- **Rezultat:** Potvrda uspješne verifikacije restorana.

- **Postupak:**

1. Inicijalizacija Mock objekta `restoranRepository`, objekta `administratorService` i objekta `Restoran` unutar funkcije `testVerificiraj()`.
2. Pronalaženje restorana sa ID-jem 1.
3. Pozivanje metode `verificiraj("1")`, koja vraća string i postavlja varijablu `isVerified` na vrijednost `true`.
4. Provjera rezultata funkcije `verificiraj("1")` pomoću:
 - Funkcije `assertEquals()`, koja uspoređuje rezultat s očekivanom vrijednosti “Restoran je verificiran.”
 - Funkcije `assertTrue()`, koja provjerava ima li varijabla restorana `isVerified()` vrijednost `true`.

```
class AdministratorServiceImplTest {
    @Mock 1 usage
    private OcjenjivacRepository ocjenjivacRepository;

    @Mock 3 usages
    private RestoranRepository restoranRepository;

    @InjectMocks 2 usages
    private AdministratorServiceImpl administratorService;

    @BeforeEach
    void setUp() {
        MockitoAnnotations.openMocks(this);
    }

    @Test
    public void testVerificiraj() {
        Restoran restoran = new Restoran();
        restoran.setIsVerified(false);
        when(restoranRepository.findById("1")).thenReturn(Optional.of(restoran));
        String result = administratorService.verificiraj(id: "1");
        assertEquals(expected: "Restoran je verificiran.", result);
        assertTrue(restoran.getIsVerified());
        verify(restoranRepository, times(wantedNumberOfInvocations: 1)).findById("1");
        verify(restoranRepository, times(wantedNumberOfInvocations: 1)).save(restoran);
    }
}
```

Slika 6.6: Test verifikacija restorana

Ispitni slučaj 7: Test provjere funkcionalnosti liste ocjenjivača

- **Opis funkcionalnosti:** Test ispituje funkciju `listaOcenjivaca()`.
- **Ulazni podaci:**
 - Mock objekt `ocjenjivacRepository`.
 - Objekt `AdministratorServiceImpl` s injektiranim `administratorService`

- pomoću `@InjectMocks`.
- Lista ocjenjivača s 3 ocjenjivača.
 - **Očekivani rezultat:** Funkcija `listaOcenjivaca()` vraća listu s jednim brojem članova kao ulazni podatak.
 - **Rezultat:** Potvrda jednakosti veličine povratne liste.
 - **Postupak:**
 1. Inicijalizacija objekta `mockListaOcenjivaca`.
 2. Dodavanje 3 objekta `Ocenjivac`.
 3. Provjera rezultata metode.

```
@Test
public void testListaOcenjivaca() {
    List<Ocenjivac> mockListaOcenjivaca = new ArrayList<>();
    mockListaOcenjivaca.add(new Ocenjivac());
    mockListaOcenjivaca.add(new Ocenjivac());
    mockListaOcenjivaca.add(new Ocenjivac());

    when(ocjenjivacRepository.findAll()).thenReturn(mockListaOcenjivaca);

    List<Ocenjivac> rezultat = administratorService.listaOcenjivaca();
    assertEquals( expected: 3, rezultat.size());
}
```

Slika 6.7: Test lista ocjenjivača

Ispitni slučaj 8: Test provjere svojstva ocjenjivača

- **Opis funkcionalnosti:** Metoda `retrieveOcenjivac` koristi podatke iz OAuth2 tokena za dohvaćanje korisnika iz baze.
- **Ulazni podaci:**
 - Token s atributima (`sub = "user123", name = "userExample", email = "user@example.com"`).
 - Korisnik u `KorisnikRepository` s ID-jem “user123”.
- **Očekivani rezultat:**
 - Metode `getOcenjivacIme()`, `getOcenjivacId()`, `getEmail()`, `getUserType()` vraćaju ispravne vrijednosti.
- **Rezultat:** Uspješni prolaz ispitivanja.
- **Postupak:**
 1. Inicijalizira se **Mock** objekt pomoću `MockitoAnnotations.openMocks(this)`.
 2. Postavljaju se atributi objektu token.
 3. Simulira se ponašanje repozitorija pomoću funkcija:
 - `KorisnikRepository.findById("user123")`
 - `OcenjivacRepository.getReferenceById("user123")`
 4. Poziva se metoda `retrieveOcenjivac` iz servisa `korisnikService`.
 5. Uspoređuju se očekivani i dobiveni rezultati za vrijednosti:
 - `ocjenjivacIme`

- ocjenjivacId
- email
- UserType
pomoću funkcije assertEquals().

```
class KorisnikServiceImplTest {
    @Mock 1 usage
    private OcjenjivacRepository ocjenjivacRepository;
    @Mock 1 usage
    private KorisnikRepository korisnikRepository;
    @InjectMocks 1 usage
    private KorisnikServiceImpl korisnikService;

    @Mock 4 usages
    private OAuth2User token;

    @BeforeEach
    public void KorisnikServiceImplTest() {
        MockitoAnnotations.openMocks(this);
    }
    @Test
    public void retrieveOcenjivacTest() {
        when(token.getAttribute("sub")).thenReturn("user123");
        when(token.getAttribute("name")).thenReturn("userExample");
        when(token.getAttribute("email")).thenReturn("user@example.com");

        Korisnik mockUser = new Ocjenjivac();
        when(korisnikRepository.findById("user123")).thenReturn(Optional.of(mockUser));
        when(ocjenjivacRepository.getReferenceById("user123")).thenReturn((Ocenjivac) mockUser);

        KorisnikBool result = korisnikService.retrieveOcenjivac(token);

        assertEquals("userExample", result.getOcenjivacIme());
        assertEquals("user123", result.getOcenjivacId());
        assertEquals("user@example.com", result.getEmail());
        assertEquals(UserType.OCJENJIVAC, result.getUserType());
    }
}
```

Slika 6.8: Korisnik service test

Ispitivanje sustava

Ispitivanje sustava provedeno je uz pomoć dodatka za preglednik Selenium IDE. U nastavku su prikazani provedeni ispitni slučajevi.

Testni slučaj 1: Registracija ocjenjivača

- Ulazi:
 - username = "testUser"
 - firstName = "test"
 - lastName = "User"
- Simulacija korisničkih akcija:
 1. Otvoriti URL aplikacije: <http://booknbite.site/>.

2. Postaviti veličinu prozora na 1051x798.
 3. Kliknuti na polje za unos korisničkog imena.
 4. Unijeti vrijednost "testUser".
 5. Kliknuti na polje za unos imena.
 6. Unijeti vrijednost "test".
 7. Kliknuti na polje za unos prezimena.
 8. Unijeti vrijednost "User".
 9. Kliknuti na gumb za potvrdu.
 10. Zatvoriti preglednik.
- **Očekivani izlaz:** Korisnik je preusmjeren na početnu stranicu nakon registracije.

	Command	Target	Value
1	✓ open	http://booknbite.site/	
2	✓ set window size	1051x798	
3	✓ click	id=username	
4	✓ type	id=username	testUser
5	✓ click	id=firstName	
6	✓ type	id=firstName	test
7	✓ click	id=lastName	
8	✓ type	id=lastName	User
9	✓ click	css=rounded-button	
10	✓ close		

Slika 6.9: Test registracija ocjenjivača; parametri

Running 'RegistracijaOcenjivacTest'

1. open on http://booknbite.site/ OK
2. setWindowSize on 1051x798 OK
3. click on id=username OK
4. type on id=username with value testUser OK
5. click on id=firstName OK
6. type on id=firstName with value test OK
7. click on id=lastName OK
8. type on id=lastName with value User OK
9. click on css=.rounded-button OK
10. close OK

'RegistracijaOcenjivacTest' completed successfully

Slika 6.10: Test registracija ocjenjivača; rezultati

Testni slučaj 2: Nedovoljni podaci pri registraciji

- Ulazi:

– username = "invalidUser"

- **Simulacija korisničkih akcija:**

1. Otvoriti URL aplikacije: <http://booknbite.site/>.
2. Postaviti veličinu prozora na 1051x798.
3. Kliknuti na polje za unos korisničkog imena.
4. Unijeti vrijednost "invalidUser".
5. Kliknuti na gumb za potvrdu.

- **Koraci ispitivanja:**

1. Otvoriti aplikaciju u pregledniku pomoću URL-a.
2. Postaviti veličinu prozora na 1054x800 piksela.
3. Kliknuti na polje za unos korisničkog imena.
4. Unijeti vrijednost invalidUser u polje.
5. Provjeriti da element za validaciju nije prisutan, iako se zapravo očekuje da je prisutan (lokator: `css=.label-input:nth-child(4) > label`).
6. Kliknuti na gumb za potvrdu registracije (lokator: `css=.rounded-button`).

- **Očekivani izlaz:**

Na stranici bi se trebao prikazati validacijski element koji javlja da nedostaje podatak za ime.

- **Dobiveni izlaz:**

Validacijski element je pronađen.

Http://booknbite.site/		
	Command	Target
1	✓ open	http://booknbite.site/
2	✓ set window size	1054x800
3	✓ click	id=username
4	✓ type	id=username
5	X verify element not present	css=.label-input:nth-child(4) > label
6	✓ click	css=.rounded-button

Slika 6.11: Test registracija nedovoljno podataka; parametri

Running 'MissingInputRegistrationTest'

1. open on <http://booknbite.site/> OK
2. setWindowSize on 1054x800 OK
3. click on id=username OK
4. type on id=username with value invalidUser OK
5. verifyElementNotPresent on `css=.label-input:nth-child(4) > label` Failed:
Element with locator `css=.label-input:nth-child(4) > label` was found
6. click on `css=.rounded-button` OK

'MissingInputRegistrationTest' ended with 1 error(s)

Slika 6.12: Test registracija nedovoljno podataka; rezultat

Testni slučaj 3: Uređivanje korisničkog profila ocjenjivača

- **Ulazi:**

- username = "testUser1"
 - lastName = "User1"

- **Simulacija korisničkih akcija:**

1. Otvaranje aplikacije u pregledniku.
2. Klik na polje korisničkog profila.
3. Klik na gumb za uređivanje profila.
4. Klik na polje za uređivanje korisničkog imena.
5. Unos novog korisničkog imena.
6. Klik na polje za uređivanje prezimena.
7. Unos novog prezimena.
8. Spremanje izmjena.

- **Koraci ispitivanja:**

1. Otvoriti aplikaciju putem URL-a: <http://booknbite.site/>.
2. Postaviti veličinu prozora na 1054x800 piksela.
3. Kliknuti na polje za uređivanje korisničkog imena (lokator: `css=.username-text`).
4. Kliknuti na gumb za unos (lokator: `css=.rounded-button:nth-child(2)`).
5. Kliknuti na polje za unos korisničkog imena (lokator: `id=username`).
6. Unijeti novo korisničko ime: `testUser1`.
7. Kliknuti na polje za unos prezimena (lokator: `id=lastName`).
8. Unijeti novo prezime: `User1`.
9. Kliknuti na gumb za spremanje promjena (lokator: `css=.save-button`).
10. Zatvoriti aplikaciju.

- **Očekivani izlaz:**

Povratak na stranicu korisničkog profila. Uneseni novi podaci su ispravno spremljeni.

- **Dobiveni izlaz:**

Koraci testa su uspješno provedeni, a svi podaci su ispravno uneseni i spremljeni.

	Command	Target	Value
1	✓ open	http://booknbite.site/	
2	✓ set window size	1054x800	
3	✓ click	css=.username-text	
4	✓ click	css=.rounded-button:nth-child(2)	
5	✓ click	id=username	
6	✓ type	id=username	testUser1
7	✓ click	id=lastName	
8	✓ type	id=lastName	User1
9	✓ click	css=.save-button	
10	✓ close		

Slika 6.13: Test uređivanje profila ocjenjivača; parametri

Running 'UrediProfilOcenjivacTest'

1. open on http://booknbite.site/ OK
2. setWindowSize on 1054x800 OK
3. click on css=.username-text OK
4. click on css=.rounded-button:nth-child(2) OK
5. click on id=username OK
6. type on id=username with value testUser1 OK
7. click on id=lastName OK
8. type on id=lastName with value User1 OK
9. click on css=.save-button OK
10. close OK

'UrediProfilOcenjivacTest' completed successfully

Slika 6.14: Test uređivanje profila ocjenjivača; rezultati

Testni slučaj 4: Stvaranje grupe, ocjenjivanje proizvoda i otvaranje Google karte

- Ulazi:
 - Nema specifičnog ulaznog podatka.
- Simulacija korisničkih akcija:
 1. Stvaranje nove grupe klikom na gumb za stvaranje grupe.
 2. Odabir kategorije proizvoda.
 3. Ocjenjivanje proizvoda.
 4. Klik na gumb „Završi“.
 5. Klik na gumb za otvaranje prvog restorana na listi.
 6. Klik na prikaz lokacije na Google karti.

- **Koraci ispitivanja:**

1. Otvoriti aplikaciju u pregledniku putem URL-a: `http://booknbite.site/`.
2. Postaviti veličinu prozora preglednika na: 1051x798.
3. Kliknuti na:
 - Neodređeni CSS element: `css=.undefined`.
 - Kategoriju: `css=.category:nth-child(1) > .overlay`.
 - Element menija: `css=span:nth-child(5)`.
4. Kliknuti na gumb za završavanje radnje: `css=.zavrsi-button`.
5. Kliknuti na zaobljeni gumb: `css=.rounded-button:nth-child(1)`.
6. Kliknuti na element unutar GM stila: `css=.gm-style > div > div:nth-child(2)`.
7. Aktivirati gumb za prelazak u puni ekran: `css=.gm-fullscreen-control`.
8. Izvršiti skriptu za pomicanje stranice: `window.scrollTo(0,137.60000610351562)`.
9. Ponovno kliknuti na gumb za puni ekran: `css=.gm-fullscreen-control`.
10. Zatvoriti preglednik.

Command		Target
1	<code>✓ open</code>	<code>http://booknbite.site/</code>
2	<code>✓ set window size</code>	<code>1051x798</code>
3	<code>✓ click</code>	<code>css=.undefined</code>
4	<code>✓ click</code>	<code>css=.category:nth-child(1) > .overlay</code>
5	<code>✓ click</code>	<code>css=span:nth-child(5)</code>
6	<code>✓ click</code>	<code>css=.zavrsi-button</code>
7	<code>✓ click</code>	<code>css=.rounded-button:nth-child(1)</code>
8	<code>✓ click</code>	<code>css=.gm-style > div > div:nth-child(2)</code>
9	<code>✓ click</code>	<code>css=.gm-fullscreen-control</code>
10	<code>✓ run script</code>	<code>window.scrollTo(0,137.60000610351562)</code>
11	<code>✓ click</code>	<code>css=.gm-fullscreen-control</code>
12	<code>✓ close</code>	

Slika 6.15: Test stvaranje grupe i ocjenjivanje jela; parametri

1. open on <http://booknbite.site/> OK
 2. setWindowSize on 1051x798 OK
 3. click on css=undefined OK
 4. click on css=.category:nth-child(1) > .overlay OK
 5. click on css=span:nth-child(5) OK
 6. click on css=.zavrsi-button OK
7. Trying to find css=.rounded-button:nth-child(1)... OK
 8. click on css=.gm-style > div > div:nth-child(2) OK
 9. click on css=.gm-fullscreen-control OK
 10. runScript on window.scrollTo(0,137.60000610351562) OK
 11. click on css=.gm-fullscreen-control OK
 12. close OK

Slika 6.16: Test stvaranje grupe i ocjenjivanje jela; rezultati

Korištene tehnologije i alati

Uspješna suradnja tima ostvarena je korištenjem aplikacije **Discord**, koja pruža jednostavnu i organiziranu komunikaciju putem funkcionalnosti poput “channel” i “thread”.

Kako bismo kvalitetno dokumentirali izradu projekta, među ostalim smo koristili alate **Astah UML** i **Visual Paradigm for UML** za izradu UML dijagrama. Te alate smo odabrali zbog intuitivnog sučelja, napredne mogućnosti za analizu i detaljnu dokumentaciju.

Ovaj projekt koristi **Java 23** kao glavni programski jezik za razvoj backend sustava. Java je odabrana zbog svoje sigurnosti, stabilnosti, aktualnosti i kompatibilnosti s radnim okvirom Spring Boot. Ove karakteristike omogućuju jednostavno održavanje i proširenje sustava, kao i izgradnju pouzdanih aplikacija.

Za radni okvir odabran je **Spring Boot** u verziji 3.3.5. Spring Boot je popularan alat za izgradnju REST API-ja i backend sustava, poznat po svojoj jednostavnosti i bogatom ekosustavu. U sklopu ovog projekta korišteno je više modula. Modul **spring-boot-starter-web** omogućuje razvoj web aplikacija i REST API-ja, dok **spring-boot-starter-data-jpa** pruža podršku za rad s bazama podataka putem Java Persistence API-ja (JPA). Za sigurnost aplikacije korišten je modul **spring-boot-starter-security**, koji implementira autentifikaciju i autorizaciju. Nadalje, integracija OAuth2 protokola realizirana je pomoću modula **spring-boot-starter-oauth2-resource-server** i **spring-boot-starter-oauth2-client**. Alati za testiranje aplikacije uključeni su u modul **spring-boot-starter-test**, koji podržava jedinično i integracijsko testiranje.

Projekt koristi **Supabase** kao bazu podataka zbog njene jednostavne integracije,

podrške za transakcije i kompatibilnosti s JPA. Supabase podržava različite verzije, uključujući Auth (verzija 2.168.0), PostgREST (verzija 12.2.3) i Postgres (verzija 15.6.1.135), što omogućuje sigurnu i skalabilnu obradu podataka.

Za upravljanje ovisnostima i automatizaciju izgradnje aplikacije koristi se **Maven**, čija je verzija definirana unutar Spring Boot parent konfiguracije. Kao razvojni alati koriste se **IntelliJ IDEA**, zbog integrirane podrške za Maven i Spring Boot, te VS Code i WebStorm kao alternativni alati. Verzioniranje koda osigurano je pomoću alata **Git** u verziji 2.47.0, koji omogućuje učinkovitu suradnju unutar tima i praćenje promjena.

Tijekom izrade dizajna korisničkog sučelja, prema kojem je razvijen frontend, koristili smo alat **Figma** zbog njegove intuitivnosti i mogućnosti timske suradnje. Za razvoj interaktivnog korisničkog sučelja koristi se **React** u verziji **18.3.1**. React je odabran zbog svoje jednostavnosti, brzine i velikog ekosustava alata i biblioteka. **React Router 6.28.0** koristi se za upravljanje rutama unutar aplikacije, dok **Axios 1.7.7** omogućuje upravljanje HTTP zahtjevima prema backend sustavu. Za responzivan dizajn korisničkog sučelja koristi se **Bootstrap** u verziji **5.3.3**.

Razvojni alati za frontend uključuju **ESLint**, koji se koristi za statičku analizu koda i održavanje konzistentnog stila, te **Prettier**, koji automatski formatira kod kako bi se povećala čitljivost i smanjile mogućnosti za pogreške.

Za testiranje **React** komponenti koriste se **Jest** i **React Testing Library**. Ovi alati omogućuju pisanje jediničnih testova kako bi se osigurala stabilnost i funkcionalnost korisničkog sučelja. Ispitivanje sustava provedeno je uz pomoć dodatka za preglednik **Selenium IDE 3.17.2**. Ispitivanje komponenti ostvareno je pomoću **JUnit5** alata za ispitivanje.

Aplikacija je hostana na **AWS-u**, čime je osigurana pouzdana i skalabilna infrastruktura za Spring Boot aplikaciju i PostgreSQL bazu podataka. AWS je odabran zbog svoje fleksibilnosti, pouzdanosti i integracijskih mogućnosti koje nudi.

Instalacija i pokretanje aplikacije

Popis potrebnih programa i alata * Node.js 18 * npm 9.8 * Java 23 * Maven 3.9 * Docker 27.3

1. Kloniranje repozitorija

Potrebno je klonirati repozitorij s Githuba koristeći naredbu:

```
git clone https://github.com/IvoGabud/Book-n-Bite.git
```

2. Frontend

Nakon toga slijedi izgradnja frontend dijela aplikacije koja se provodi u sljedećim koracima:

- Pozicioniranje u direktorij: `Book-n-Bite/client/book-n-bite/`
- Instalacija ovisnosti: `npm install`
- Izgradnja aplikacije: `npm run build`

Nakon toga potrebno je kopirati sve stvorene datoteke iz `Book-n-Bite/client/book-n-bite/build` direktorija u `Book-n-Bite/server/booknbite/src/main/resources/static` direktorij.

3. Backend

Sada je moguće izgraditi backend dio aplikacije unutar Docker kontejnera provodeći sljedeće korake:

- Pozicioniranje u direktorij: `Book-n-Bite/server/booknbite`
- Izgradnja slike Docker kontejnera `docker buildx build --platform linux/amd64 -t korisnicko_ime/booknbite:1.0 .`
- Objava stvorene slike na Docker hub `docker push korisnicko_ime/booknbite:1.0`

4. Konfiguracija AWS poslužitelja

Potrebno je otici na stranicu AWS upravljačke ploče (Console) te se zatim prijaviti ili stvoriti novi korisnički račun. Nakon toga treba odabrati uslugu EC2 (Elastic Compute Cloud) te stvoriti novu sigurnosnu grupu (Security group). Sigurnosnu grupu potrebno je konfigurirati kao što je prikazano slikom.

Zatim u kartici Key pairs potrebno je stvoriti novi par sigurnosnih ključeva.

Stvorenu datoteku book-n-bite-aws.pem potrebno je pohraniti na sigurno mjesto na računalu. Konačno u kartici Instances potrebno je pritisnuti Launch instance te podešiti dane opcije na sljedeći način:

5. Pokretanje aplikacije na poslužitelju

Potrebno se pozicionirati u direktorij unutar kojeg se nalazi `book-n-bite-aws.pem` datoteka te se zatim povezati s poslužiteljem pomoću ssh klijenta koristeći naredbu:

```
ssh -i book-n-bite-aws.pem ec2-user@ip_adresa_poslužitelja
```

Na poslužitelju je zatim potrebno instalirati te pokrenuti Docker servis pomoću sljedećih naredba:

```
sudo yum install docker
```

Create security group Info

A security group acts as a virtual firewall for your instance to control inbound and outbound traffic. To create a new security group, complete the fields below.

Basic details

Security group name Info
book-n-bite-sg
Name cannot be edited after creation.

Description Info
book-n-bite-sg

VPC Info
vpc-

Inbound rules Info

Type	Protocol	Port range	Source	Description - optional
HTTP	TCP	80	Any... <input type="button" value="Delete"/>	0.0.0.0/0 <input type="button" value="Delete"/>
SSH	TCP	22	My IP <input type="button" value="Delete"/>	93.139. <input type="button" value="Delete"/>

[Add rule](#)

Outbound rules Info

Type	Protocol	Port range	Destination	Description - optional
All traffic	All	All	Any... <input type="button" value="Delete"/>	0.0.0.0/0 <input type="button" value="Delete"/>

[Add rule](#)

Tags - optional

A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

No tags associated with the resource.

[Add new tag](#)

You can add up to 50 more tags

[Cancel](#) [Create security group](#)

Figure 11: Security group konfiguracija

Create key pair Info

Key pair
A key pair, consisting of a private key and a public key, is a set of security credentials that you use to prove your identity when connecting to an instance.

Name
book-n-bite-aws
The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.

Key pair type Info
 RSA ED25519

Private key file format
 .pem For use with OpenSSH
 .ppk For use with PuTTY

Tags - optional
No tags associated with the resource.

[Add new tag](#)

You can add up to 50 more tags.

[Cancel](#) [Create key pair](#)

Figure 12: Key pair konfiguracija

Launch an instance [Info](#)

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags [Info](#)

Name
Book'n bite

Add additional tags

Summary

Number of instances [Info](#)

1

Application and OS Images (Amazon Machine Image) [Info](#)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Search our full catalog including 1000s of application and OS images

Recent [Quick Start](#)

Amazon Linux	macOS	Ubuntu	Windows	Red Hat	SUSE Linux	Debian

Browse more AMIs
Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

Amazon Linux 2023 AMI
ami-0e54671bdf3c8ed8d (64-bit (x86), uefi-preferred) / ami-0e54671bdf3c8ed8d (64-bit (Arm), uefi)
Virtualization type: HVM enabled: true Root device type: ebs

Free tier eligible

Description

Amazon Linux 2023 is a modern, general purpose Linux-based OS that comes with 5 years of long term support. It is optimized for AWS and designed to provide a secure, stable and high-performance execution environment to develop and run your cloud applications.

Amazon Linux 2023 AMI 2023.6.20241212.0.x86_64 HVM kernel-6.1

Architecture: 64-bit (x86) Boot mode: uefi-preferred AMI ID: ami-0e54671bdf3c8ed8d Username: ec2-user Verified provider:

Instance type [Info](#) | [Get advice](#)

Instance Type

12.micro	Family: t2 1 vCPU 1 GiB Memory Current generation: true Free tier eligible	All generations
	On-Demand Windows 10 Pro base pricing: \$0.0100 USD per Hour On-Demand SUSE base pricing: \$0.0114 USD per Hour On-Demand Ubuntu Pro base pricing: \$0.0152 USD per Hour On-Demand Linux base pricing: \$0.0134 USD per Hour On-Demand RHEL base pricing: \$0.0278 USD per Hour	Compare instance types

Additional costs apply for AMIs with pre-installed software

Key pair (login) [Info](#)

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - required
book-n-bite-aws [Create new key pair](#)

Network settings [Info](#)

Network [Info](#)

vpc-
Subnet [Info](#)
No preference (Default subnet in any availability zone)

Auto-assign public IP [Info](#)

Enable
Additional charges apply when outside of free tier allowance

Firewall security groups [Info](#)
A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

Create security group Select existing security group

Common security groups [Info](#)
Select security groups [Compare security group rules](#)

book-n-bite-sg [X](#)

Security groups that you add or remove here will be added to or removed from all your network interfaces.

Configure storage [Info](#)

Advanced

1x GiB [gp3](#) [Root volume \(Not encrypted\)](#)

(Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage)

Figure 13: EC2 instance konfiguracija

```
sudo service docker start
```

Nakon toga moguće je pokrenuti Docker kontejner.

```
sudo docker run -d -p 80:8080 korisnicko_ime/booknbite:1.0
```

Promjena postavki aplikacije

Konfiguracija baze podataka

U spring boot serveru u application.properties datoteci postavlja se datasource URL u liniji spring.datasource.url=datasource-url te se zatim postavi username i password za uspješnu autentifikaciju (spring.datasource.username=username spring.datasource.password=password).

Opis pristupa aplikaciji na javnom poslužitelju

Aplikaciji je moguće pristupiti upisom ip adrese AWS poslužitelja u adresnu traku internet preglednika. Navedenu adresu moguće je pronaći unutar AWS konzole u kartici Instances usluge EC2. Alternativno moguće je registrirati domenu te je pridružiti navedenoj ip adresi. U tom slučaju aplikaciji je moguće pristupiti upisom dobivenog domenskog imena (<http://booknbite.site/>) u adresnu traku internet preglednika.

Korisnici aplikacije mogu joj pristupiti putem bilo kojeg internetskog preglednika. Nije potrebno instalirati dodatne programe ili ekstenzije za korištenje aplikacije. Aplikacija je također optimizirana za rad na različitim uređajima. Jedino ograničenje pristupa aplikaciji jest obvezna autentifikacija putem Google OAuth2. Nakon uspješne autentifikacije, korisnici se prijavljuju na svoj profil i mogu koristiti funkcionalnosti aplikacije ovisno o vrsti profila (ocjenjivač, restoran ili administrator).

Upute za administratore

Pristup administratorskom sučelju - Administratori se prijavljuju putem **Google autentifikacije**, jednako kao i regularni korisnici. - Ako korisnički račun korišten za autentifikaciju postoji na popisu administratora, aplikacija će automatski otvoriti konzolu za administratore. - **Početni podaci za prijavu:** - **Email:** test.booknbite@gmail.com - **Password:** blackRisotto

Redovito održavanje

- Pregled logova:** Za pregled logova aplikacije unutar Docker kontejnera koristite naredbu:
`sudo docker logs [container_id]`

- **Ažuriranje aplikacije**

Ažuriranje aplikacije uključuje sljedeće korake:

1. **Preuzimanje najnovijih promjena iz repozitorija:**

Povucite najnovije promjene s glavne grane:

```
git pull origin main
```

2. **Ponovna izgradnja aplikacije:**

Nakon povlačenja promjena, slijedite korake za instalaciju i pokretanje aplikacije opisane u sekcijama:

- 2. Frontend
- 3. Backend
- 5. Pokretanje aplikacije na poslužitelju

3. **Provjera aplikacije:** Nakon što pokrenete aplikaciju, slijedite upute za pristup aplikaciji opisane u sekciji:

- Opis pristupa aplikaciji na javnom poslužitelju

Razvoj aplikacije Book 'n Bite bio je zahtjevan projekt koji nas je izložio novim tehnologijama i tehničkim izazovima, ali i pružio priliku za značajan osobni i timski razvoj. Cilj izrade aplikacije bio je pružiti jednostavno i intuitivno rješenje grupama korisnika za donošenje zajedničkih odluka pri odabiru restorana. Za izradu aplikacije imali smo otprilike 15 tjedana, a projekt je bio podijeljen u dvije glavne faze. U prvoj fazi naglasak je bio na izradi projektne dokumentacije, definiranju funkcionalnih zahtjeva, dijagrama i arhitekture sustava, dok je druga faza bila usmjerena na implementaciju ključnih funkcionalnosti, testiranje aplikacije i pripremu za prezentaciju projekta.

Nakon uvodne vježbe s mentorom i međusobnog upoznavanja, naš tim je samostalno osmislio temu projekta, vođen idejom kreiranja aplikacije koja će olakšati grupno donošenje odluka prilikom odabira restorana. Jedan od prvih izazova s kojima smo se susreli bio je kvalitetno definiranje svih funkcionalnih zahtjeva aplikacije. Ovaj proces zahtijevao je detaljne rasprave unutar tima kako bismo osigurali da aplikacija ispunjava potrebe svih korisničkih grupa – ocjenjivača, restorana i administratora. Dodatni izazov bio je osmišljavanje baze podataka koja je trebala podržati složene funkcionalnosti aplikacije. Iako je inicijalna struktura baze bila funkcionalna, tijekom razvoja bilo je potrebno doraditi i prilagoditi model baze kako bi zadovoljio nove zahtjeve.

Kako bismo postigli ciljeve projekta, bilo je potrebno upoznati se s tehnologijama koje većina članova nije koristila prije. Kroz istraživanje, predavanja i suradnju unutar tima, stekli smo potrebna znanja za uspješnu realizaciju aplikacije. Prva revizija projekta završila je uspješnom implementacijom osnovnih značajki, poput sustava prijave i registracije korisnika pomoću integracije Google OAuth2 autentifikacije. Ovime smo postavili temelje za daljnji razvoj aplikacije.

Druga faza rada na projektu započela je detaljnim planiranjem i usklađivanjem individualnih odgovornosti svakog člana tima. Iako nije bilo formalnih podtimova, svaki član tima preuzeo je specifične zadatke, poput implementacije

dizajna, izrade UML dijagrama ili integracije s Google Mapsom za prikaz lokacija restorana. Zadatci su često bili međusobno ovisni, što je zahtijevalo visok stupanj koordinacije i suradnje među članovima. Tijekom implementacije složenijih funkcionalnosti, poput prikaza lokacija na karti, bilo je potrebno dodatno istraživanje kako bismo pronašli optimalna rješenja.

Svi članovi tima aktivno su doprinosili uspješnom razvoju aplikacije. Komunikacija unutar tima odvijala se putem Discorda, uz redovite sastanke na kojima smo raspravljali o napretku, rješavali nedoumice i prilagođavali planove kako bismo osigurali pravovremeno izvršenje zadataka. Ključna je bila pametna raspodjela poslova unutar tima, kako bi svaki član mogao doprinijeti na najbolji mogući način, a pritom se držati zadanih rokova. Povremeno su se javljali izazovi, poput zastoja uzrokovanih čekanjem na dovršetak prethodnih zadataka. Ove situacije rješavali smo jasnijim planiranjem i učinkovitijom međusobnom komunikacijom.

Razvoj aplikacije Book 'n Bite bio je vrijedno iskustvo koje nam je omogućilo stjecanje praktičnih znanja u radu s modernim tehnologijama poput Reacta i Spring Boota, kao i dublje razumijevanje procesa projektiranja i implementacije softverskih rješenja. Kroz projekt smo unaprijedili tehničke, timske i organizacijske vještine, uključujući planiranje, raspodjelu zadataka i učinkovitu suradnju unutar tima. Projekt nas je pripremio za izazove u programskom inženjerstvu te pružio uvide u realne probleme i njihova rješenja. Kao tim, zadovoljni smo postignutim ciljem te svjesni da aplikacija ima potencijal za daljnji razvoj i unaprjeđenje. Budući rad na aplikaciji obuhvaća niz mogućnosti poput integracije s platformama za dostavu hrane, razvoja personaliziranih preporuka temeljenih na povijesti korisničkih ocjena i uvođenja mobilne verzije aplikacije kako bi se proširila dostupnost i olakšalo korištenje. Dodatno, razmatra se selektiranje restorana prema lokaciji, pri čemu bi rang lista prikazivala samo restorane unutar zadanog radijusa koji korisnik definira. Ove nadogradnje omogućile bi aplikaciji širu primjenu i još bolje prilagođavanje potrebama korisnika.

Zaključno, projekt Book 'n Bite omogućio nam je stvaranje funkcionalne i korisne aplikacije koja ima potencijal postati rješenje za širok raspon korisnika. Iskustva stečena tijekom razvoja bila su jako važna za naš tehnički i profesionalni razvoj te su nam pružila čvrste temelje za buduće projekte.

Rev.	Opis promjene/dodataka	Autori	Datum
0.1	Stvorene stranice Home, Opis projektnog zadatka i Analiza zahtjeva, dodana prva verzija opisa projektnog zadatka i nefunkcionalni zahtjevi	Ivana Renić	25.10.2024.
0.2	Dodani početni funkcionalni zahtjevi	Rene Filipović	25.10.2024
0.3	Dopunjena stranica Home	Ivana Renić	28.10.2024.

Rev.	Opis promjene/dodataka	Autori	Datum
0.4	Stvorene stranice Dnevnik promjena dokumentacije, Popis literature, Prikaz aktivnosti grupe, Specifikacija zahtjeva sustava, Arhitektura i dizajn sustava	Ivana Renić	28.10.2024.
0.5	Dodan je popis obrazaca uporabe	Ivana Renić	30.10.2024.
0.6	Dodan je opis baze podataka	Ivo Gabud	03.11.2024.
0.7	Dodana je relacijska shema baze podataka	Ivo Gabud	04.11.2024.
0.8	Napisani opisi svih obrazaca uporabe	Ivo Gabud, Ivana Renić, Gabrijel Leko	06.11.2024.
0.9	Dodani dijagrami obrazaca uporabe i sekvensijski dijagrami	Ivo Gabud	06.11.2024.
0.10	Popravak funkcionalnih zahtjeva	Filip Knapić	06.11.2024.
0.11	Proširen opis projekta i dodane slike sličnih aplikacija	Ivo Gabud	07.11.2024.
0.12	Proširen opis projekta	Ivana Renić	11.11.2024.
0.13	Ažuriran opis baze podataka	Adrian Ambroz	12.11.2024.
0.14	Prepravljeni i dodani sekvensijski dijagrami te odgovarajući opisi, dopunjeno popis literature	Ivana Renić	12.11.2024.
0.15	Izmjena slike relacijske sheme baze podataka	Adrian Ambroz	12.11.2024.
0.16	Dodana arhitektura sustava, popravljeni nefunkcionalni zahtjevi i dodani dionici	Ivana Renić	12.11.2024.
0.17	Dodan dijagram razreda, prepravljeni funkcionalni zahtjevi i obrasci uporabe	G.Leko, I.Gabud, R.Filipović, A.Ambroz, F.Knapić, J. Ivanković, I.Renić	15.11.2024.
1.0	Dovršena sva dokumentacija za prvu reviziju	G.Leko, I.Gabud, R.Filipović, A.Ambroz, F.Knapić, J. Ivanković, I.Renić	15.11.2024.
1.1	Napravljen osnovni dio uputa za puštanje u pogon	Ivo Gabud	24.12.2024.

Rev.	Opis promjene/dodataka	Autori	Datum
1.2	Prepravljeni dijagrami obrazaca uporabe	Jelena Ivanković	24.12.2024.
1.3	Dodan dijagram aktivnosti	Gabrijel Leko, Ivana Renić	07.01.2025.
1.4	Prepravljeni sekvencijski dijagrami	Rene Filipović, Ivana Renić	09.01.2025.
1.5	Dodan dijagram komponenata	Filip Knapić, Ivana Renić	09.01.2025.
1.6	Dodan dijagram stanja	Adrian Ambroz, Ivana Renić	13.01.2025.
1.7	Dodane korištene tehnologije i alati	Ivana Renić	14.01.2025.
1.8	Dopunjene upute za puštanje u pogon	Ivana Renić	22.01.2025.
1.9	Dodan dijagram razmještaja	Ivana Renić	23.01.2025.
1.10.	Dodano ispitivanje programskog rješenja	Ivana Renić, Adrian Ambroz	23.01.2025.
1.11.	Dodan je zaključak i budući rad	Ivana Renić	23.01.2025.
1.12.	Ispravljen je dijagram stanja	Adrian Ambroz	24.01.2025.
1.13.	Dodani novi dijagrami razreda	Ivana Renić	24.01.2025.

1. Programsko inženjerstvo, FER ZEMRIS, http://www.fer.hr/predmet/pro_inz
2. I. Sommerville, "Software engineering", 8th ed, Addison Wesley, 2007.
3. T.C.Lethbridge, R.Langaniere, "Object-Oriented Software Engineering", 2nd ed. McGraw-Hill, 2005.
4. I. Marsic, Software engineering book“, Department of Electrical and Computer Engineering, Rutgers University, http://www.ece.rutgers.edu/%CB%9Cmarsic/_books/SE
5. The Unified Modeling Language, <https://www.uml-diagrams.org/>
6. Astah Community, <http://astah.net/editions/uml-new>
7. Spring: <https://gitlab.com/hrvojesimic/progi-project-teams-backend>
8. React: <https://gitlab.com/jatomic/opp-project-teams-frontend>
9. Deployment: <https://github.com/progi-devops/progi-monorepo>

Dnevnik sastajanja

1. sastanak
 - Datum: 16. listopada 2024.

- Prisustvovali: G.Leko, I.Gabud, R.Filipović, A.Ambroz, F.Knapić, J. Ivanković, I.Renić
- Teme sastanka: > * Dogovorena je okvirna tema projekta i ime tima > * Ivo Gabud postavljen je za voditelja tima

2. sastanak

- Datum: 19. listopada 2024.
- Prisustvovali: G.Leko, I.Gabud, R.Filipović, A.Ambroz, F.Knapić, J. Ivanković, I.Renić
- Teme sastanka: > * Uspostavljena je Discord grupa svih članova
> * Određeni su funkcionalni i nefunkcionalni zahtjevi te zahtjevi domene
> * Odabrane su tehnologije za realizaciju projekta (Front-end: React.js, Back-end: Spring Boot) > * Napravljena je kratka prezentacija koja sadrži ključne informacije o projektu

3. sastanak

- Datum: 29. listopada 2024.
- Prisustvovali: G.Leko, I.Gabud, R.Filipović, A.Ambroz, F.Knapić, J. Ivanković, I.Renić
- Teme sastanka: > * Razrada arhitekture sustava > * Raspodjela poslova za nadolazeći tjedan;
> 1. Obrasci uporabe: Ivo Gabud, Ivana Renić, Gabriel Leko > 2. Baza podataka i njen dijagram: Adrian Ambroz, Filip Knapić
> 3. Okvirni dizajn stranice u Figmi: Jelena Ivanković > 4. Sekvencijski dijagrami: Rene Filipović > 5. Dijagrami obrazaca uporabe: Jelena Ivanković

4. sastanak

- Datum: 29. listopada 2024.
- Prisustvovali: I.Gabud, I.Renić
- Teme sastanka: > * Određeni obrasci uporabe

5. sastanak

- Datum: 29. listopada 2024.
- Prisustvovali: F.Knapić, A.Ambroz
- Teme sastanka: > * Izrada prve inačice ER modela baze podataka

6. sastanak

- Datum: 30. listopada 2024.
- Prisustvovali: F.Knapić, A.Ambroz
- Teme sastanka: > * Dopuna ER modela baze podataka > * Izrada opisa i dijagrama baze podataka

7. sastanak

- Datum: 7. studenog 2024.

- Prisustvovali: G.Leko, I.Gabud, R.Filipović, A.Ambroz, F.Knapić, J. Ivanković, I.Renić

- Teme sastanka: > * Pregled dizajna aplikacije u Figmi
> * Pregled dosad postojeće dokumentacije
> * Dogovor oko izmjena obrazaca uporabe
> * Rasподjela poslova za nadolazeći tjedan

8. sastanak

- Datum: 9. studenog 2024.

- Prisustvovali: G.Leko, R.Filipović
- Teme sastanka: > * Razvitak backenda > * Povezivanje backenda s bazom podataka > * Povezivanje backenda s frontendom

9. sastanak

- Datum: 10. studenog 2024.

- Prisustvovali: G.Leko, I.Gabud, R.Filipović, A.Ambroz, F.Knapić, J. Ivanković, I.Renić

- Teme sastanka: > * Promjene u bazi podataka > * Popravljeni sekvencijski dijagrami > * Promjene dizajna u Figmi > * Pregled dokumentacije i rasподjela posla oko nedovršenih djelova

10. sastanak

- Datum: 11. studenog 2024.
- Prisustvovali: I.Gabud, R.Filipović
- Teme sastanka: > * Dogovor oko komunikacije frontend-a i backend-a > * Dogovor implementacijskih detalja

11. sastanak

- Datum: 12. studenog 2024.
- Prisustvovali: I.Gabud, R.Filipović
- Teme sastanka: > * Dogovor oko deploymenta aplikacije > * Rasprava o postojećoj implementaciji

12. sastanak

- Datum: 15. studenog 2024.

- Prisustvovali: G.Leko, I.Gabud, R.Filipović, A.Ambroz, F.Knapić, J. Ivanković, I.Renić
- Teme sastanka: > * Završni sastanak prije prve predaje > * Dogovor o finalnim promjenama za prvu predaju

13. sastanak

- Datum: 11. prosinca 2024.
- Prisustvovali: G.Leko, I.Gabud, R.Filipović, A.Ambroz, F.Knapić, J. Ivanković, I.Renić
- Teme sastanka: > * Potreban popravak sekvenčkih dijagrama (R.Filipović) > * Potreban popravak dijagrama obrazaca uporabe (J.Ivanković) > * Planiranje daljnog rada na projektu i razvoja aplikacije

14. sastanak

- Datum: 22. prosinca 2024.
- Prisustvovali: G.Leko, I.Gabud, R.Filipović, A.Ambroz, F.Knapić, J. Ivanković, I.Renić
- Teme sastanka: > Raspodjela poslova; > * Popraviti bazu podataka (F.Knapić) > * Napraviti dijagram komponenti (F.Knapić) > * Upute za puštanje u pogon (I.Gabud) > * Dijagram aktivnosti (G.Leko) > * Dijagram stanja (A.Ambroz) > * Dijagram razmještaja (I.Renić) > * Dovršiti dizajn (J.Ivanković) > * Implementirati dizajn (F.Knapić) > * Popraviti modele i napisati endpointove za preostale funkcionalnosti (R.Filipović) > * Dovršiti dokumentaciju (I.Renić) > * Istražiti i implementirati integraciju s Google maps (G.Leko) > * Napisati integracijske testove (A.Ambroz) > * Implementirati frontend funkcionalnosti (I.Gabud)

15. sastanak

- Datum: 8. siječnja 2025.
- Prisustvovali: I.Gabud, F.Knapić, J.Ivanković
- Teme sastanka: > * Pregled napravljenog dizajna aplikacije u Figma i dogovor oko implementacije dizajna

16. sastanak

- Datum: 10. siječnja 2025.
- Prisustvovali: G.Leko, I.Gabud, R.Filipović, A.Ambroz, F.Knapić, J.Ivanković, I.Renić
- Teme sastanka: > * Pregled dosad napravljenog > * Planiranje daljnog rada na projektu prije pokazivanja alfa verzije

17. sastanak

- Datum: 12. siječnja 2025.
- Prisustvovali: I.Gabud, A.Ambroz, R.Filipović
- Teme sastanka:
> * Testovi za ispitivanje programskog rješenja

18. sastanak

- Datum: 13. siječnja 2025.

- Prisustvovali: I.Gabud, G.Leko
- Teme sastanka:
 > * Integracija s vanjskim poslužiteljem (Google Maps)

19. sastanak

- Datum: 15. siječnja 2025.
- Prisustvovali: G.Leko, I.Gabud, R.Filipović, A.Ambroz, F.Knapić, J.Ivanković, I.Renić
- Teme sastanka: > * Planiranje dalnjeg rada na projektu nakon pokazivanja alfa verzije

20. sastanak

- Datum: 21. siječnja 2025.
- Prisustvovali: G.Leko, I.Gabud, R.Filipović, A.Ambroz, F.Knapić, J.Ivanković, I.Renić
- Teme sastanka:
 > * Izrada prezentacije > * Popravak Uputa za puštanje u pogon > * Dijagram razmještaja > * Popravci na frontendu

Tablica aktivnosti

Aktivnost	Ivo Gabud	Gabriel Leko	Rene Filipović	Adrian Ambroz	Filip Knapić	Jelena Ivanković	Ivana Renić
Upravljanje projektom	25h	0h	0h	0h	0h	0h	0h
Opis projektnog zadatka	5h	0h	0h	0h	0h	0h	4h
Funkcionalni zahtjevi	0h	0h	2h	0h	2h	0h	3h
Opis pojedinih obrazaca	4h	4h	0h	0h	0h	0h	6h
Dijagram obrazaca	0h	0h	0h	0h	0h	4.5h	0h
Sekvencijski dijagrami	0h	0h	3h	0h	0h	0h	2h
Opis ostalih zahtjeva	0h	2h	0h	1h	0h	0h	2h
Dionici	0h	0h	0h	0h	0h	0h	1.5h
Arhitektura i dizajn sustava	0h	0h	0h	0h	3h	0h	2h
Baza podataka	0h	1h	1h	4h	5h	0h	0h
Dijagram razreda	1h	2h	0h	1h	0h	0h	1h

Aktivnost	Ivo Gabud	Gabriel Leko	Rene Filipović	Adrian Ambroz	Filip Knapić	Jelena Ivanković	Ivana Renić
Dijagram stanja aktivnosti	0h	0h	0h	2h	0h	0h	1h
Dijagram komponenti	0h	0h	0h	0h	3h	0h	2h
Usvajanje tehnologija i alata	1h	1h	1h	1h	1h	1h	2h
Korištene tehnologije i alati	0h	2h	0h	0h	0h	0h	2h
Ispitivanje programskog rješenja	1h	0.5h	1h	5h	0h	0h	4h
Dijagram razmještaja	0h	0h	0h	0h	0h	0h	4h
Upute za puštanje u pogon	6h	0h	1h	0h	0h	0h	2.5h
Dnevnik sastajanja	0h	0h	0h	0h	0h	0h	2h
Zaključak i budući rad	0h	0h	0h	0h	0h	0h	4h
Ključni izazovi i rješenja	0h	0h	0h	0h	0h	0h	1h
Popis literature	0h	0h	0h	0h	0h	0h	0.5h
Izrada baze podataka	0h	0h	2h	3h	6h	0h	0h
Spajanje s bazom podataka	0h	0.5h	1h	0h	0h	0h	0h
Izrada prezentacije	2h	2h	0h	0h	0h	1h	0h
Dizajn aplikacije	0h	0h	0h	0h	0h	26h	0h
Front end	60h	6h	0h	0h	40h	0h	0h
Back end	0h	12h	50h	0h	0h	0h	0h
Postavljanje web aplikacije na poslužitelj	10h	2h	5h	0h	0h	0h	0h
Ažuriranje README-a	0h	0h	0h	0h	0h	0h	0.5h
Uređivanje wiki-ja	1h	2h	1h	2h	1h	0.5h	9h

Aktivnost	Ivo Gabud	Gabriel Leko	Rene Filipović	Adrian Ambroz	Filip Knapić	Jelena Ivanković	Ivana Renić
Dnevnik promjena dokumentacije	0h	0h	0h	0h	0h	0h	2h

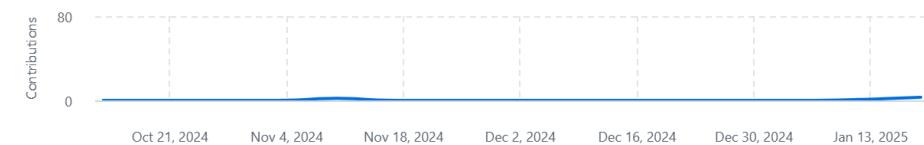
Dijagrami pregleda promjena

Commits over time

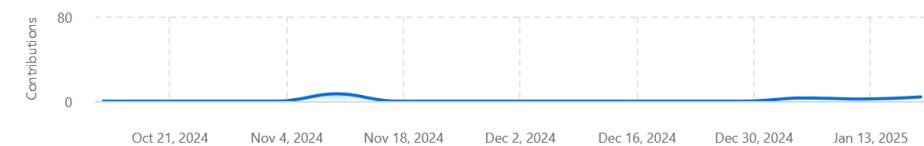
Weekly from Oct 13, 2024 to Jan 19, 2025



AmbrozAdrian's Commits



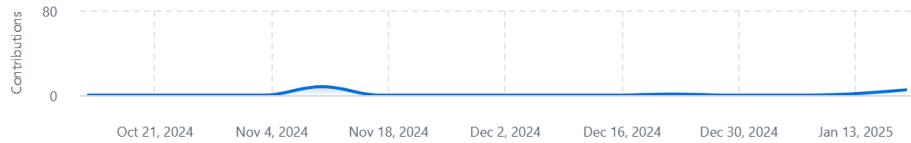
FilipKnapić's Commits



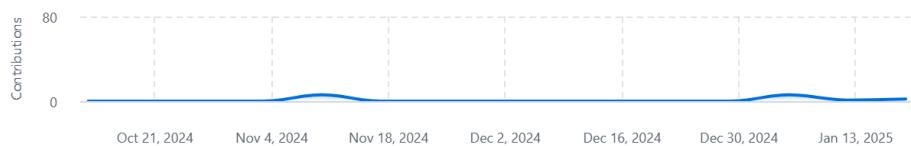
IvoGabud's Commits



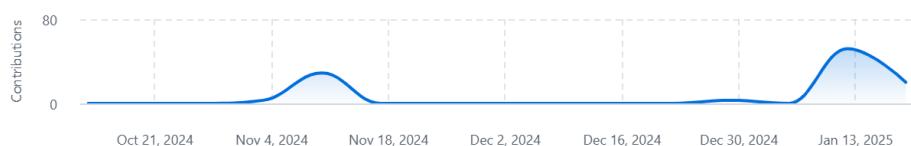
jelena-i's Commits



ivanarenic's Commits



RFilipovic's Commits



Rielriel's Commits



- Gabrijel Leko pod imenom Rielriel

Ključni izazovi i rješenja

Prvi i ključni izazov s kojim smo se susreli bio je savladavanje novih tehnologija potrebnih za izradu kvalitetne web-aplikacije. Mnogi članovi tima prvi su se put susreli s alatima poput Reacta, Bootstrapa i GitHuba. Također, rijetki od nas su prije radili na ovakvom projektu. Ovaj izazov rješili smo postupnim učenjem kroz praksi i vlastite pogreške, uz međusobno pomaganje i suradnju, pri čemu su iskusniji članovi tima aktivno pomagali onima s manje iskustva.

Dodatni izazov bila je i pametna raspodjela poslova unutar tima, kako bi svaki član mogao doprinijeti na najbolji mogući način, a pritom se držati zadanih rokova. Održavanje jasne komunikacije i dogovora među članovima bilo je ključno za uspješnu realizaciju projekta.

Ovaj projekt pružio nam je ne samo tehničko znanje u razvoju web-aplikacije, već i dragocjeno iskustvo u organizaciji, suradnji i upravljanju vremenom. Naučili smo važnost timskog rada, dobre komunikacije i međusobnog povjerenja kao ključnih elemenata za postizanje zajedničkog cilja.

Book n' Bite