

# Programsko inženjerstvo

Ak. god. 2023./2024.

## Što želiš čitati?

Dokumentacija, Rev. 2

Grupa: *DaVincijevi koderi*

Voditelj: *Andrea Gazibarić*

Datum predaje: *19.1.2024.*

Nastavnik: *Alan Jović*

# Sadržaj

<b>1</b>	<b>Dnevnik promjena dokumentacije</b>	<b>3</b>
<b>2</b>	<b>Opis projektnog zadatka</b>	<b>6</b>
<b>3</b>	<b>Specifikacija programske potpore</b>	<b>10</b>
3.1	Funkcionalni zahtjevi . . . . .	10
3.1.1	Obrasci uporabe . . . . .	12
3.1.2	Sekvencijski dijagrami . . . . .	23
3.2	Ostali zahtjevi . . . . .	29
<b>4</b>	<b>Arhitektura i dizajn sustava</b>	<b>30</b>
4.1	Komponente . . . . .	30
4.2	Izbor arhitekture . . . . .	34
4.3	Organizacija sustava . . . . .	36
4.4	Organizacija aplikacije . . . . .	38
4.5	Baza podataka . . . . .	40
4.5.1	Opis tablica . . . . .	40
4.5.2	Dijagram baze podataka . . . . .	42
4.6	Dijagram razreda . . . . .	44
4.7	Dijagram stanja . . . . .	47
4.8	Dijagram aktivnosti . . . . .	49
4.9	Dijagram komponenti . . . . .	51
<b>5</b>	<b>Implementacija i korisničko sučelje</b>	<b>53</b>
5.1	Korištene tehnologije i alati . . . . .	53
5.2	Ispitivanje programskog rješenja . . . . .	55
5.2.1	Ispitivanje komponenti . . . . .	55
5.2.2	Ispitivanje sustava . . . . .	61
5.3	Dijagram razmještaja . . . . .	67
5.4	Upute za puštanje u pogon . . . . .	68

<b>6 Zaključak i budući rad</b>	<b>70</b>
<b>Popis literature</b>	<b>72</b>
<b>Indeks slika i dijagrama</b>	<b>73</b>
<b>Dodatak: Prikaz aktivnosti grupe</b>	<b>74</b>

# 1. Dnevnik promjena dokumentacije

Rev.	Opis promjene/dodatka	Autori	Datum
0.1	Napravljen predložak.	Andrea Gazibarić	26.10.2023.
0.2	Dodani funkcionalni zahtjevi, aktori i di-onici.	Luka Blagaić	27.10.2023.
0.2.1	Ispravak funkcionalnih zahtjeva.	Luka Blagaić	28.10.2023.
0.2.2	Dodan opis funkcionalnih zahtjeva.	Andrea Gazibarić	30.10.2023.
0.2.3	Ispravak opisa funkcionalnih zahtjeva.	Andrea Gazibarić	02.11.2023.
0.2.4	Dodani dijagrami obrazaca uporabe.	Andrea Gazibarić	02.11.2023.
0.3	Dodani ER i REL dijagrami baze podataka.	Luka Blagaić	02.11.2023.
0.4	Dodani sekvencijski dijagrami.	Renato Balun	05.11.2023.
0.4.1	Dodan opis sekvencijskih dijagrama.	Renato Balun	06.11.2023.
0.5	Arhitektura i dizajn sustava, algoritmi i strukture podataka.	Davor Marenić	16.11.2023.
0.6	Dodan opis nefunkcionalnih zahtjeva.	Luka Zelić	17.11.2023.

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

Rev.	Opis promjene/dodatka	Autori	Datum
0.7	Dodan opis projektnog zadatka.	Šima Vu- letić	17.11.2023.
0.8	Dodani dijagrami razreda.	Filip Šturlić	17.11.2023.
0.9	Implementacija.	Luka Zelić, Renato Balun, Davor Marenić	*
1.0	Verzija samo s bitnim dijelovima za 1. ciklus.	*	17.11.2023.
1.1	Dodan dijagram razmještaja.	Renato Balun	29.12.2023.
1.2	Dodana inačica UML dijagrama razreda.	Luka Bla- gaić	02.01.2024.
1.3	Dodani dijagrami stanja i aktivnosti.	Andrea Gazibarić	02.01.2024.
1.4	Dodan novi ER model baze podataka.	Luka Bla- gaić	05.01.2024.
1.5	Dodana inačica UML dijagrama razreda.	Luka Bla- gaić	12.01.2024.
1.5	Dodan dijagram komponenti.	Šima Vu- letić	12.01.2024.
1.6	Dodan opis korištenih tehnologija.	Luka Bla- gaić	19.01.2024.

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

Rev.	Opis promjene/dodatka	Autori	Datum
1.7	Dodan opis puštanja aplikacije u pogon.	Luka Zelić	19.01.2024.
1.8	Dodano ispitivanje komponenti i sustava.	Šima Vuletić, Renato Balun	19.01.2024.
1.9	Implementacija.	Andrea Gazibarić, Luka Zelić, Luka Blagaić, Filip Šturlić, Renato Balun, Šima Vuletić	*.
2.0	Završna verzija.	*	19.01.2024.

## 2. Opis projektnog zadatka

Cilj ovog projekta je razviti programsku podršku za stvaranje web aplikacije "Što želiš čitati?" kojoj je glavni zadatak omogućiti korisniku pronalazak ponuditelja knjige u blizini koji ima u prodaji traženu knjigu na hrvatskom ili srodnom jeziku. Također omogućuje i međusobno povezivanje ponuditelja knjiga radi proširenja međusobnih ponuda. Naime, ljubitelji čitanja koji žele čitati stranu literaturu imaju problem s pronalaskom željenih knjiga na hrvatskom zato što mnogi naslovi nisu prevedeni na hrvatski ili srodni jezik poput srpskog i bosanskog, a u slučaju da postoji prijevod zbog loše ažurnosti web stranica ponuditelja knjiga nije ih lako pronaći.

Stoga je ideja ove web aplikacije da omogući čitatelju unos značajke željene knjige, imena i ponuditelja knjige. Također je omogućena pretraga dostupnosti po navedenim značajkama. Potencijalna korist ovog projekta je što olakšava i ubrzava čitateljima pronalazak tražene knjige, a u slučaju da ona ne postoji ne oduzima im puno vremena, te tako poboljšava njihovo iskustvo. Dodatna potencijalna korist je što čitatelju omogućava proširivanje horizonata i upoznavanje sa stranom kulturom čitajući stranu literaturu. Rezultat toga bi bila promocija kulture čitanja na području Republike Hrvatske. S druge strane, ponuditelji knjiga, poput izdavača, antikvarijata i preprodavača, imaju priliku proširiti svoj doseg na širu publiku te povećavajući prodaju knjiga iz svoje ponude. To bi pomoglo malim poduzetnicima koji drže antikvarijate i nezavisne knjižare da ostanu u poslu, a osnažilo bi izdavače da prošire svoju ponudu i prevedu još i više stranih naslova, te bi samim time i kupci bili zadovoljniji.

U aplikaciji postoje tri uloge:

- *Neregistrirani korisnik: potraživač knjiga, kupac*
- *Registrirani korisnik: ponuditelj knjiga*
- *Administrator*

Neregistrirani korisnik ima mogućnost pristupu web sjedištu aplikacije, te može pretraživati ponude knjiga. Pretraga se provodi po značajkama knjige koju neregistrirani korisnik unosi i po nazivu ponuditelja. Neregistriranom korisniku

se nudi pregled karte na kojoj su označene lokacije (adrese) svih ponuditelja knjiga te ih on može izabrati. Nakon odabira ponuditelja neregistriranom korisniku prikazuje se popis, to jest ponuda svih knjiga izabranog ponuditelja. Ponuđene knjige sadržavaju sljedeće značajke:

- *naziv,*
- *autore,*
- *godinu izdanja,*
- *izdavača,*
- *kategorija izdavača (domaći, strani),*
- *žanr,*
- *ISBN,*
- *broj izdanja,*
- *stanje očuvanosti,*
- *tekstni opis,*
- *sliku korica,*
- *oznaku vrste knjige,*
- *listu ponuda.*

S time da oznaku vrste knjige možemo okarakterizirati u 5 različitih kategorija:

- *Knjiga je na stranom jeziku (npr. engleski), a ne postoji izdanje na hrvatskom ili srodnom jeziku (bosanski, srpski)*
- *Knjiga je izdana na hrvatskom jeziku i dobavljiva je na području Hrvatske*
- *Knjiga je izdana na hrvatskom jeziku, ali nije dobavljiva na području Hrvatske (npr. izdavač je rasprodao izdanje)*
- *Knjiga je izdana na srodnom jeziku, dobavljiva je samo na njihovom tržištu*
- *Knjiga je izdana na srodnom jeziku, dobavljiva je u Hrvatskoj, ne postoji na hrvatskom jeziku*

Dok lista ponude svake knjige treba uključivati:

- *naziv ponuditelja,*
- *broj dostupnih primjeraka,*
- *cijenu knjige kod dotičnog ponuditelja.*

Uz to da ako knjiga nema više dostupnih registriranih ponuditelja knjigu se ne prikazuje u ponudi prilikom pretrage u sustavu. Osim odabira ponuditelja i pretrage ponuđenih knjiga, neregistriranom korisniku se daje i mogućnost da kroz sučelje



aplikacije zatraži od registriranog korisnika izdavača da stavi u ponudu prijevod željene knjige i samim time u dogovoru sa stranim izdavačem prevede knjigu na hrvatski jezik. Zahtjevi za prijevod se jednostavno bilježe na stranici i akumuliraju se za svaku knjigu na stranom jeziku kako bi ponuditelj imao uvid o potražnji prijevoda. Važno je još napomenuti da ostala komunikacija vezana uz nabavu knjige između neregistriranog korisnika i ponuditelja se ne odvaja kroz aplikaciju već preko drugih kanala poput e-pošte i telefonskim pozivom.

Registrirani korisnik je isključivo ponuditelj knjiga koji spada u jednu od tri kategorije:

- *Izdavač*
- *Antikvarijat*
- *Preprodavač*

Ponuditelj bira svoju kategoriju tijekom registracije, pri čemu registraciju treba odobriti administrator nakon pregleda zahtjeva. Nadalje registrirani korisnik unosi potrebne informacije za kontakt ponuditelja, a to su:

- *naziv,*
- *e-pošta,*
- *adresa,*
- *izdavača,*
- *broj telefona.*

Nakon toga registrirani korisnik pristupa web aplikaciji uz pomoć korisničkog imena i lozinke. Svaki ponuditelj ima mogućnost ponuditi neograničeni broj naslova knjiga i njezinih primjeraka u skladu sa svojom kategorijom. Tijekom kreiranja ponude svaki ponuditelj je primoran ispravno odrediti oznaku vrste knjige, te pri svakom dodavanju nove knjige ili primjeraka već postojeće knjige u web aplikaciju. Uz to pri svakoj izmjeni ponude knjiga poput dodavanja novih primjeraka već postojeće knjige ili uklanjanju primjeraka koji su prodani ponuditelj je dužan provjeriti da oznaka vrste knjige odgovara stvarnosti i ako to nije slučaj, izmijeniti ju. Ponuditelj također nije obavezan ponuditi sve knjige koje ima u ponudi, to jest u svojoj ponudi može imati više knjiga nego što ih je ponudio na web stranici. Izdavač u svojoj ponudi smije nuditi knjige isključivo knjige na hrvatskom jeziku, te na temelju prikupljenih zahtjeva neregistriranih korisnika smije zatražiti izdavača strane knjige za dozvolu prijevoda knjige sa stranog ili srodnog jezika na hrvatski jezik. Antikvarijat u svojoj ponudi smije imati knjige na stranom jeziku, srodnom

jeziku ili hrvatskom jeziku, s time da se mora svojom adresom nalaziti isključivo na području Hrvatske. Preprodavač u svojoj ponudi može imati sve vrste knjiga neovisno o prijevodu, te i one koje nisu na drugačiji način dobavljive na području Hrvatske. Uz to adresa mu može biti u Hrvatskoj i u zemljama sa srodnim jezikom, odnosno Srbiji i Bosni i Hercegovini.

Skup korisnika koji bi mogao biti zainteresiran za ostvareno rješenje se može raščlaniti na četiri skupa:

- *Ljubitelji knjiga: Oni koji traže stranu literaturu prevedene na hrvatski jezik ili srodne jezike.*
- *Izdavači: Oni koji žele proširiti dostupnost svojih knjiga na novo tržište.*
- *Antikvarijati: Oni koji imaju rijetke ili stare knjige koje žele ponuditi zainteresiranim čitateljima i kolekcionarima.*

Aplikacija bi se u budućnosti mogla proširiti na različite načine koji bi poboljšali zadovoljstvo neregistriranih i registriranih korisnika: U slučaju da u ponudi nema tražene knjige čitatelju bi se moglo ponuditi knjiga u ponudi od istog autora ili slične tematike, te na taj način čitatelj bi dobio štivo za čitanje, a ponuditelj bio ostvario dodatnu prodaju. Mogla bi se omogućiti komunikacija kupca i ponuditelja unutar aplikacije kako bi se kupcu olakšala kupnja. Moglo bi se omogućiti neregistriranim korisnicima da uz zahtjev za prijevodom određenog naslova mogu unijeti email kako bi mogli biti obaviješteni o izlasku prijevoda i proširenju ponude.

## 3. Specifikacija programske potpore

### 3.1 Funkcionalni zahtjevi

#### Dionici:

1. Neregistrirani korisnici
2. Registrirani korisnici
  - (a) Izdavači
  - (b) Antikvarijati
  - (c) Preprodavači
3. Administratori
4. Baza podataka

#### Aktori i njihovi funkcionalni zahtjevi:

1. Neregistrirani korisnik (inicijator) može:
  - (a) pretraživati ponudu knjiga po:
    - i. značajkama knjige
    - ii. ponuditelju
  - (b) pretraživati ponuditelje na karti
  - (c) odabrati ponuditelja i izlistati sve knjige u njegovoj ponudi
  - (d) zahtjevati od izdavača da kontaktira stranog izdavača za prijevod
  - (e) registrirati se
    - unijeti podatke
2. Izdavač (inicijator) može:
  - (a) zatražiti od stranog izdavača dozvolu za prijevod
  - (b) ponuditi knjige na hrvatskom jeziku
  - (c) maknuti knjige iz svoje ponude
3. Antikvarijat (inicijator) može:
  - (a) ponuditi knjige na hrvatskom i srodnim jezicima
  - (b) maknuti knjige iz svoje ponude

4. Preprodavač (inicijator) može:
  - (a) ponuditi knjige na bilo kojem jeziku
  - (b) maknuti knjige iz svoje ponude
5. Administrator (inicijator) može:
  - (a) odobriti registraciju
    - provjeriti je li adresa antikvarijata u RH i
    - jesu li ostali podatci dobri
  - (b) ukloniti registrirane korisnike
  - (c) mijenjati vrstu korisnika
  - (d) pristupiti bazi podataka
6. Baza podataka (sudionik) pohranjuje podatke o:
  - (a) registriranim korisnicima
  - (b) knjigama (uključujući i oznake knjige)
  - (c) zahtjevima za prijevod

### 3.1.1 Obrasci uporabe

#### Opis obrazaca uporabe

##### UC1 - Pretraživanje knjiga po značajkama

- **Glavni sudionik:** Neregistrirani korisnik
- **Cilj:** Pregledati ponudu knjiga
- **Sudionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
  1. odabir značajki knjige po kojima se želi pretražiti
  2. prikazuje se ponuda knjiga s odabranim značajkama

##### UC2 - Pretraživanje knjiga po ponuditelju

- **Glavni sudionik:** Neregistrirani korisnik
- **Cilj:** Pregledati ponudu knjiga
- **Sudionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
  1. odabir ponuditelja čiju ponudu želi pregledati
  2. prikazuje se ponuda knjiga odabranog poslužitelja

##### UC3 - Pretraživanje ponuditelja na karti

- **Glavni sudionik:** Neregistrirani korisnik
- **Cilj:** Pregledati ponuditelje u blizini
- **Sudionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
  1. otvaranje karte s označenim ponuditeljima
  2. odabir lokacije i ponuditelja
  3. ispis podataka o ponuditelju i njihove ponude

##### UC4 - Zahtjev za prijevodom

- **Glavni sudionik:** Neregistrirani korisnik
- **Cilj:** Zatražiti prijevod knjige na stranom jeziku
- **Sudionici:** Izdavač, baza podataka

- **Preduvjet:** Knjiga nije prevedena na hrvatski
- **Opis osnovnog tijeka:**
  1. odabir "Zahtjevaj prijevod" u sučelju aplikacije
  2. zahtjev se bilježi u bazu podataka
  3. izdavač dobija poruku o novom zahtjevu

### UC5 - Registracija

- **Glavni sudionik:** Neregistrirani korisnik
- **Cilj:** Registrirati se kao ponuditelj
- **Sudionici:** Baza podataka, administrator
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
  1. odabir "Sign up" u sučelju
  2. unos vrste ponuditelja (izdavač, antikvarijat, preprodavač)
  3. unos podataka (naziv, e-pošta, adresa, broj telefona)
  4. unos korisničkog imena i lozinke
  5. administrator odobrava račun
  6. poruka na e-pošti o odobrenju ili neodobrenju računa
- **Opis mogućih odstupanja:**
  - 3.a adresa antikvarijata nije u RH
    1. nemogućnost nastavka registracije bez promjene podataka
    2. upozorenje
  - 3.b adresa preprodavača nije u zemlji srodnog jezika
    1. nemogućnost nastavka registracije bez promjene podataka
    2. upozorenje
  - 4.a korisničko ime već postoji
    1. nemogućnost nastavka registracije bez promjene podataka
    2. upozorenje

### UC6 - Ponuda naslova knjiga

- **Glavni sudionik:** Ponuditelj
- **Cilj:** Ponuditi neograničen broj naslova knjiga
- **Sudionici:** Baza podataka
- **Preduvjet:** Registrirati se kao ponuditelj
- **Opis osnovnog tijeka:**
  1. Ponuditelj odabire "Ponudi naslov knjige" u sučelju aplikacije

2. Ponuditelj odabire naslove koje želi ponuditi
3. U bazu podataka se pohrani promjena

#### **UC7 - Ponuda primjeraka knjiga**

- **Glavni sudionik:** Ponuditelj
- **Cilj:** Ponuditi neograničen broj primjeraka knjiga
- **Sudionici:** Baza podataka
- **Preduvjet:** Registrirati se kao ponuditelj
- **Opis osnovnog tijeka:**
  1. Ponuditelj odabire "Ponudi primjerak knjige" u sučelju aplikacije
  2. Ponuditelj odabire primjerke knjiga
  3. U bazu podataka se pohrani promjena

#### **UC8 - Pregled želja neregistriranih korisnika**

- **Glavni sudionik:** Izdavač
- **Cilj:** Prikupiti želje neregistriranih korisnika
- **Sudionici:** Baza podataka
- **Preduvjet:** Registrirati se kao izdavač
- **Opis osnovnog tijeka:**
  1. Praćenje aktivnosti neregistriranog korisnika
  2. Bilježenje i prikupljanje njegovih želja

#### **UC9 - Zahtjev za dozvolu prijevoda sa stranog jezika**

- **Glavni sudionik:** Izdavač
- **Cilj:** Zatražiti prijevod knjige sa stranog jezika na hrvatski
- **Sudionici:** Baza podataka
- **Preduvjet:** Registrirati se kao izdavač
- **Opis osnovnog tijeka:**
  1. Izdavač pregleda želje neregistriranih korisnika
  2. Izdavač odabire knjige
  3. Izdavač zatraži dozvolu za prijevodom

#### **UC10 - Zahtjev za dozvolu prijevoda sa srodnog jezika**

- **Glavni sudionik:** Izdavač
- **Cilj:** Zatražiti prijevod knjige sa srodnog jezika na hrvatski
- **Sudionici:** Baza podataka

- **Preduvjet:** Registrirati se kao izdavač
- **Opis osnovnog tijeka:**
  1. Izdavač pregleda želje neregistriranih korisnika
  2. Izdavač odabire knjige
  3. Izdavač zatraži dozvolu za prijevodom

#### UC11 - Ponuda knjiga na hrvatskom jeziku od strane izdavača

- **Glavni sudionik:** Izdavač
- **Cilj:** Ponuditi knjigu na hrvatskom jeziku
- **Sudionici:** Baza podataka
- **Preduvjet:**
  1. Registrirati se kao izdavač
  2. Imati u svojoj ponudi knjige na hrvatskom jeziku
- **Opis osnovnog tijeka:**
  1. Izdavač odabire knjige za ponudu
  2. Izdavač nudi odabrane knjige
  3. U bazu podataka se pohrani promjena

#### UC12 - Ponuda knjige na stranom jeziku od strane antikvarijata

- **Glavni sudionik:** Antikvarijat
- **Cilj:** Ponuditi knjige na stranom jeziku
- **Sudionici:** Baza podataka
- **Preduvjet:** Uspješna registracija kao antikvarijata
- **Opis osnovnog tijeka:**
  1. Antikvarijat odabire knjige na stranom jeziku
  2. Antikvarijat nudi odabrane knjige
  3. U bazu podataka se pohrani promjena

#### UC13 - Ponuda knjige na srodnom jeziku od strane antikvarijata

- **Glavni sudionik:** Antikvarijat
- **Cilj:** Ponuditi knjige na srodnom jeziku
- **Sudionici:** Baza podataka
- **Preduvjet:** Uspješna registracija kao antikvarijata
- **Opis osnovnog tijeka:**
  1. Antikvarijat odabire knjige na srodnom jeziku
  2. Antikvarijat nudi odabrane knjige



3. U bazu podataka se pohrani promjena

#### UC14 - Ponuda knjige na hrvatskom jeziku od strane antikvarijata

- **Glavni sudionik:** Antikvarijat
- **Cilj:** Ponuditi knjige na hrvatskom jeziku
- **Sudionici:** Baza podataka
- **Preduvjet:** Uspješna registracija kao antikvarijata
- **Opis osnovnog tijeka:**
  1. Antikvarijat odabire knjige na hrvatskom jeziku
  2. Antikvarijat nudi odabrane knjige
  3. U bazu podataka se pohrani promjena

#### UC15 - Ponuda knjige na stranom jeziku od strane preprodavača

- **Glavni sudionik:** Preprodavač
- **Cilj:** Ponuditi knjige na stranom jeziku
- **Sudionici:** Baza podataka
- **Preduvjet:** Uspješna registracija kao preprodavača
- **Opis osnovnog tijeka:**
  1. Preprodavač odabire knjige na stranom jeziku
  2. Preprodavač nudi odabrane knjige
  3. U bazu podataka se pohrani promjena

#### UC16 - Ponuda knjige na srodnom jeziku od strane preprodavača

- **Glavni sudionik:** Preprodavač
- **Cilj:** Ponuditi knjige na srodnom jeziku
- **Sudionici:** Baza podataka
- **Preduvjet:** Uspješna registracija kao preprodavača
- **Opis osnovnog tijeka:**
  1. Preprodavač odabire knjige na srodnom jeziku
  2. Preprodavač nudi odabrane knjige
  3. U bazu podataka se pohrani promjena

#### UC17 - Ponuda knjige na hrvatskom jeziku od strane preprodavača

- **Glavni sudionik:** Preprodavač
- **Cilj:** Ponuditi knjige na hrvatskom jeziku
- **Sudionici:** Baza podataka

- **Preduvjet:** Uspješna registracija kao preprodavača
- **Opis osnovnog tijeka:**
  1. Preprodavač odabire knjige na hrvatskom jeziku
  2. Preprodavač nudi odabrane knjige
  3. U bazu podataka se pohrani promjena

#### UC18 - Određivanje ispravne oznake nove knjige

- **Glavni sudionik:** Ponuditelj
- **Cilj:** Ispravno odabrati oznaku vrste knjige kod dodavanja nove knjige
- **Sudionici:** Baza podataka
- **Preduvjet:** Registrirati se kao ponuditelj
- **Opis osnovnog tijeka:**
  1. Ponuditelj odabire novu knjigu
  2. Ponuditelj odabire ispravnu vrstu oznake za tu knjigu
  3. U bazu podataka se pohrani promjena

#### UC19 - Određivanje ispravne oznake primjerka već dodane knjige

- **Glavni sudionik:** Ponuditelj
- **Cilj:** Ispravno odabrati oznaku vrste knjige za taj primjerak
- **Sudionici:** Baza podataka
- **Preduvjet:** Registrirati se kao ponuditelj
- **Opis osnovnog tijeka:**
  1. Ponuditelj odabire već postojeću knjigu
  2. Ponuditelj odabire ispravnu vrstu oznake za tu knjigu
  3. U bazu podataka se pohrani promjena

#### UC20 - Ispravak oznake novog primjerka pri dodavanju

- **Glavni sudionik:** Ponuditelj
- **Cilj:** Izmijeniti pogrešno odabranu oznaku
- **Sudionici:** Baza podataka
- **Preduvjet:** Registrirati se kao ponuditelj
- **Opis osnovnog tijeka:**
  1. Ponuditelj odabire primjerak s pogrešnom oznakom
  2. Ponuditelj ispravlja oznaku za taj primjerak knjige
  3. Ponuditelj dodaje primjerak knjige
  4. U bazu podataka se pohrani promjena

**UC21 - Ispravak oznake novog primjerka pri uklanjanju**

- **Glavni sudionik:** Ponuditelj
- **Cilj:** Izmijeniti pogrešno odabranu oznaku
- **Sudionici:** Baza podataka
- **Preduvjet:** Registrirati se kao ponuditelj
- **Opis osnovnog tijeka:**
  1. Ponuditelj odabire primjerak s pogrešnom oznakom
  2. Ponuditelj ispravlja oznaku za taj primjerak knjige
  3. Ponuditelj uklanja primjerak knjige
  4. Iz baze podataka se uklanja primjerak knjige

**UC22 - Odobravanje registracije**

- **Glavni sudionik:** Administrator
- **Cilj:** Odobriti registraciju korisnika
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je registriran i dodijeljena su mu prava administratora
- **Opis osnovnog tijeka:**
  1. Administrator pregleda zahtjev za registraciju
  2. Administrator provjerava odgovaraju li uneseni podaci
  3. Posebno za antikvarijata provjerava je li mu adresa u RH
  4. Ako je prijava ispravna, administrator odobrava registraciju korisnika

**UC23 - Brisanje korisnika**

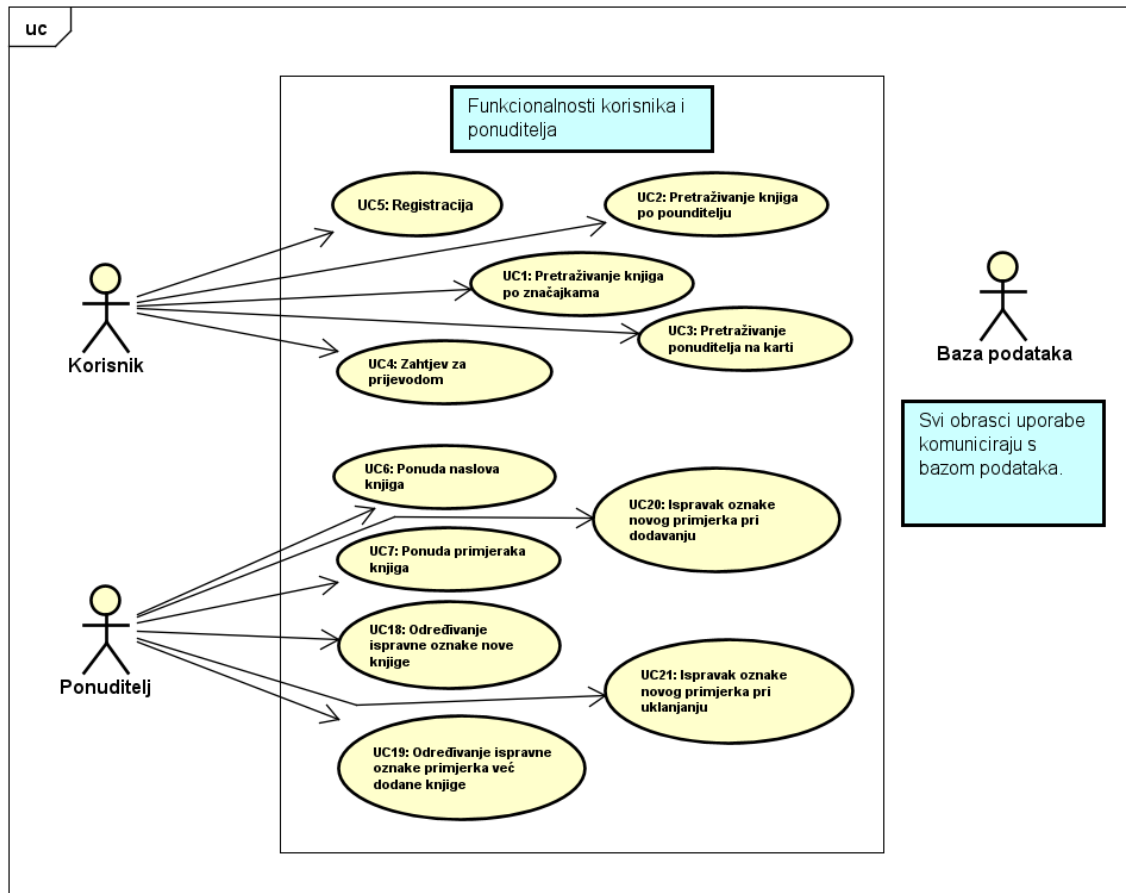
- **Glavni sudionik:** Administrator
- **Cilj:** Izbrisati registriranog korisnika
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je registriran i dodijeljena su mu prava administratora
- **Opis osnovnog tijeka:**
  1. Administrator odabire opciju uklanjanja korisnika
  2. Administrator pronalazi željenog korisnika
  3. Administrator uklanja željenog korisnika i njegove podatke iz baze podataka

**UC24 - Promjena vrste korisnika**

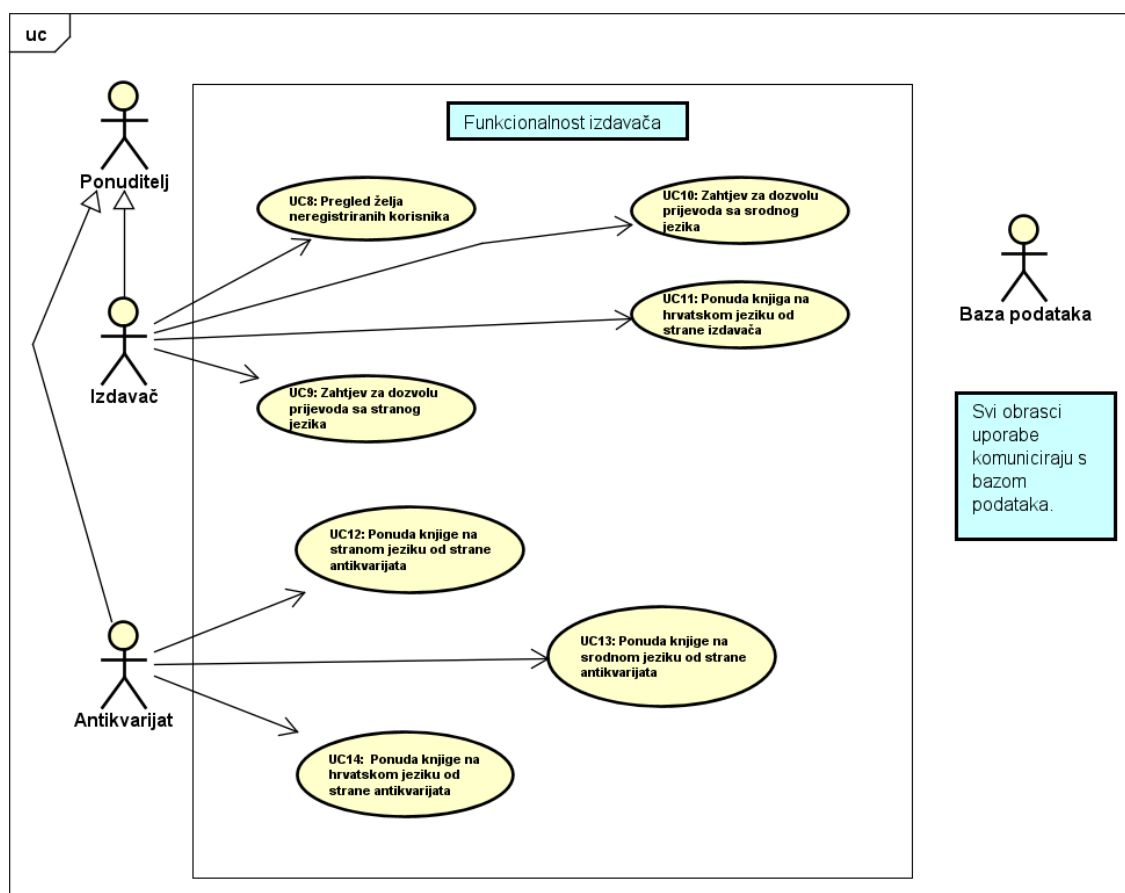
- **Glavni sudionik:** Administrator

- **Cilj:** Izmijeniti vrstu i prava korisnika
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je registriran i dodijeljena su mu prava administratora
- **Opis osnovnog tijeka:**
  1. Administrator pronalazi željenog korisnika
  2. Administrator mijenja vrstu i prava korisnika

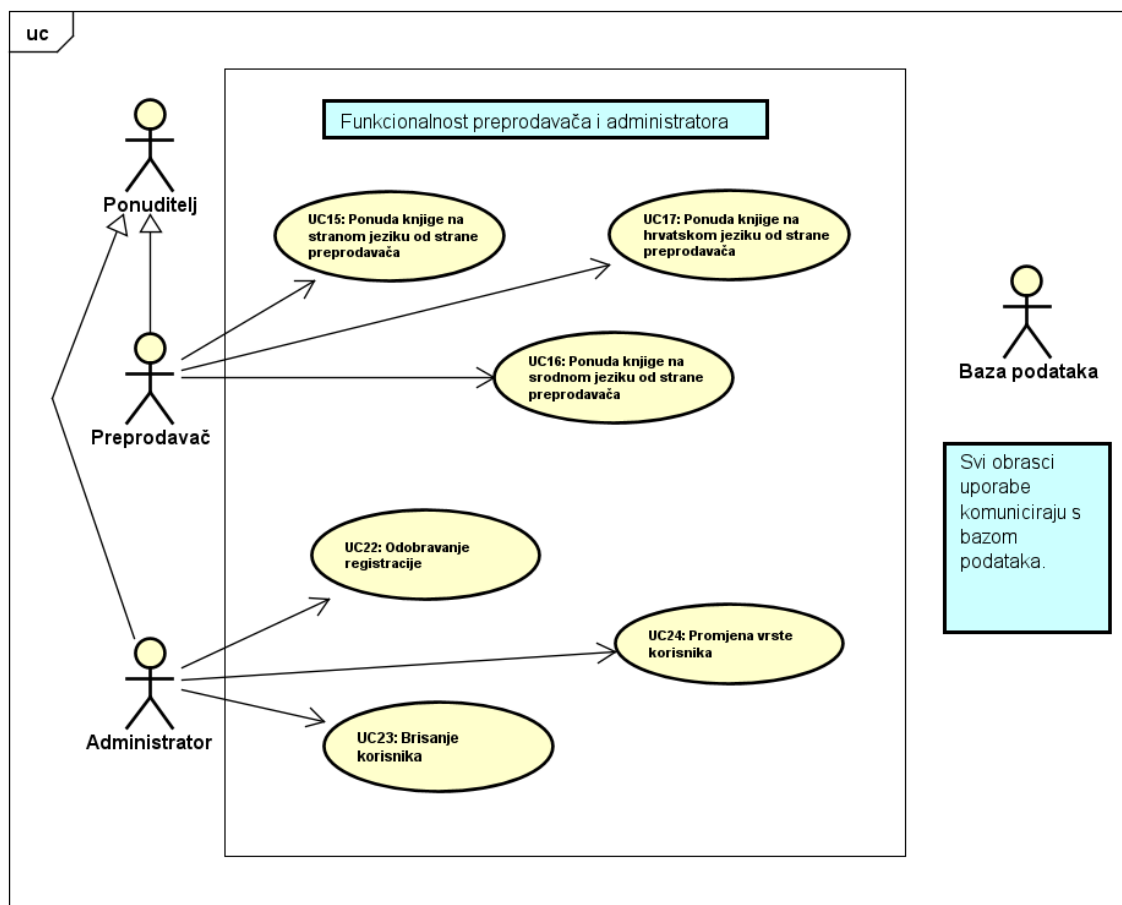
## Dijagrami obrazaca uporabe



Slika 3.1: Dijagram obrasca uporabe, funkcionalnost korisnika i ponuditelja



Slika 3.2: Dijagram obrasca uporabe, funkcionalnost izdavača i antikvarijata

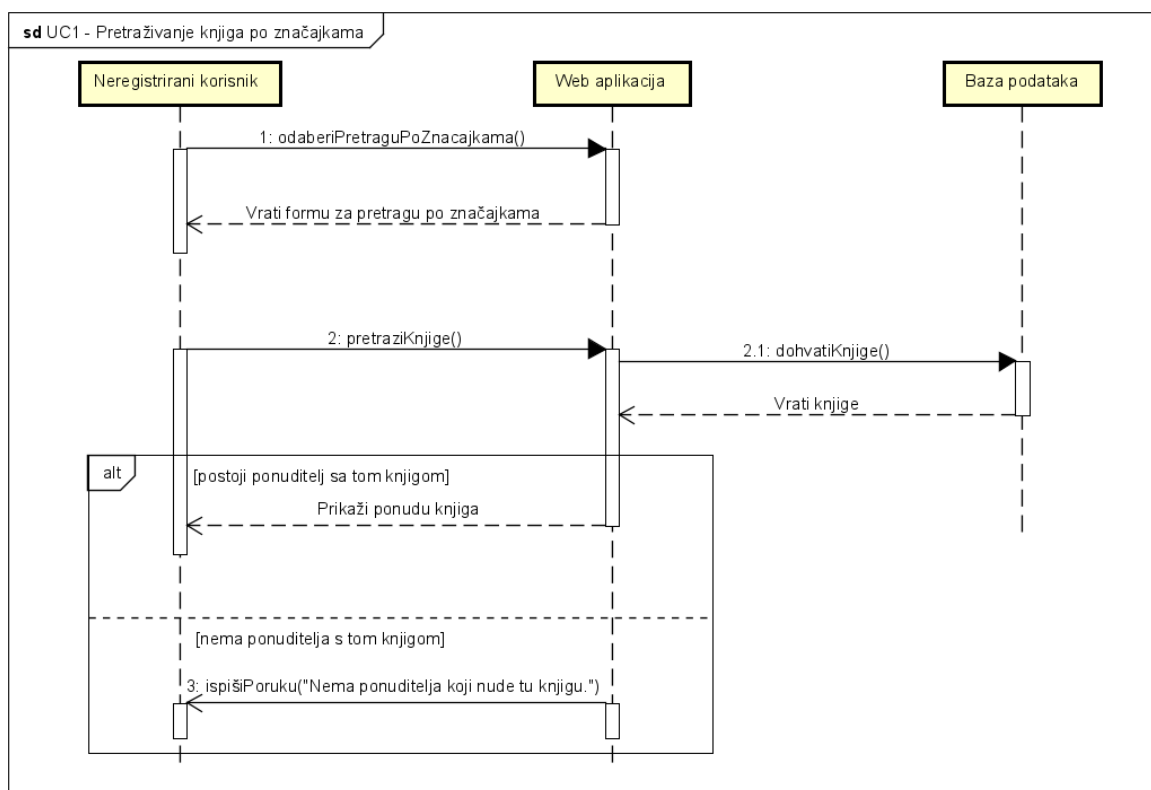


Slika 3.3: Dijagram obrasca uporabe, funkcionalnost preprodavača i administratora

### 3.1.2 Sekvencijski dijagrami

#### Obrazac uporabe UC1 - Pretraživanje knjiga po značajkama

Neregistrirani korisnik odabire pretraživanje knjige po značajkama. Na ekranu mu se pojavljuje forma za odabir značajki knjige. Korisnik odabire značajke knjige i klikne na dugme za pretraživanje kako bi pretražio knjige sa željenim značajkama. Web aplikacija zaprima zahtjev i šalje upit prema bazi podataka. Od baze podataka prema web aplikaciji se vraća rezultat. Ako u bazi podataka postoji ponuditelj koji nudi knjigu ili više knjiga s odabranim značajkama na ekranu mu se pokazuje ponuda knjiga. Ako u bazi podataka ne postoji ponuditelj koji nudi knjigu s odabranim značajkama na ekranu se ispisuje poruka da ne postoji registrirani ponuditelj koji nudi knjigu sa željenim značajkama.

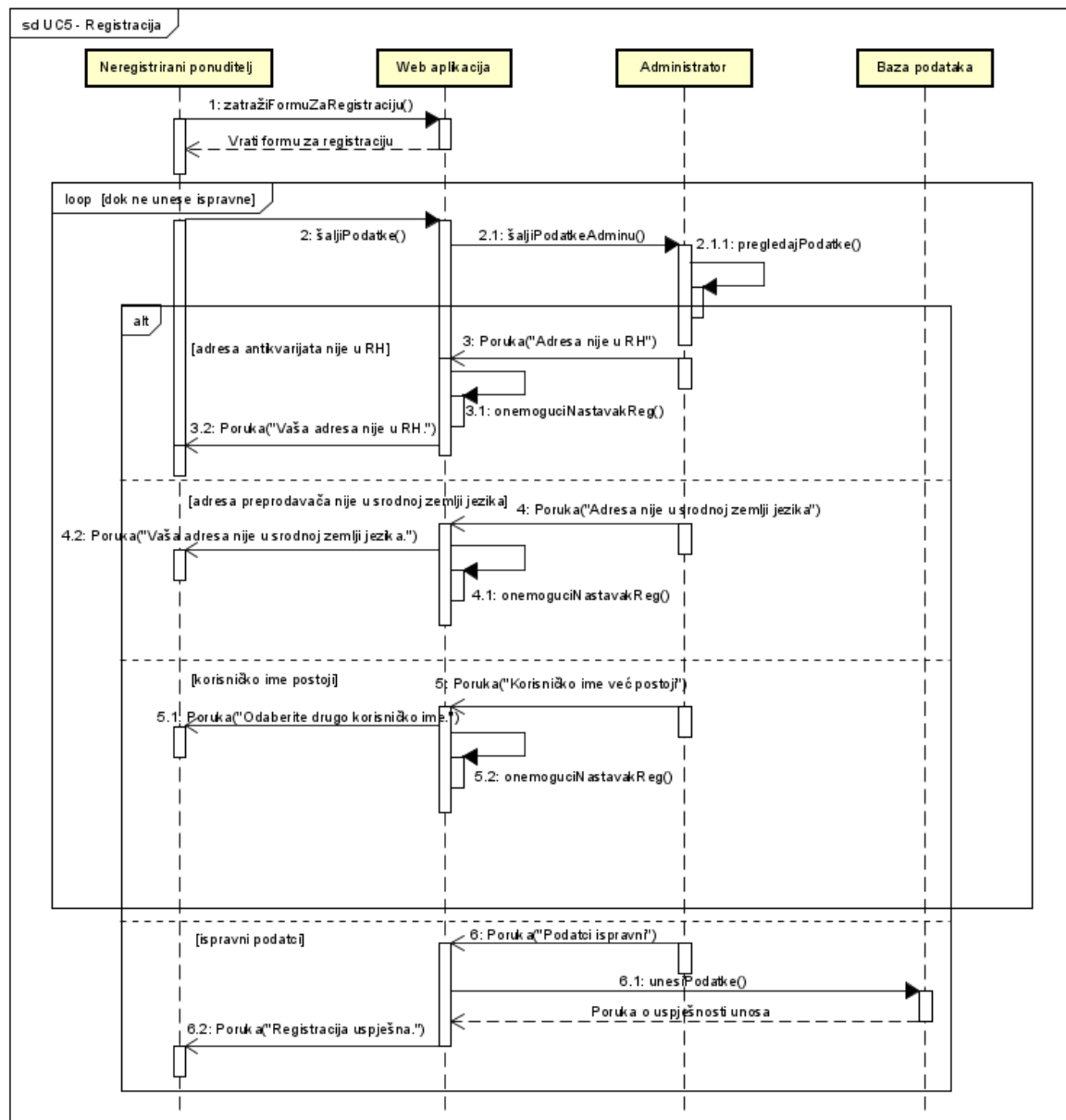


Slika 3.4: Sekvencijski dijagram za UC1



**Obrazac uporabe UC5 - Registracija**

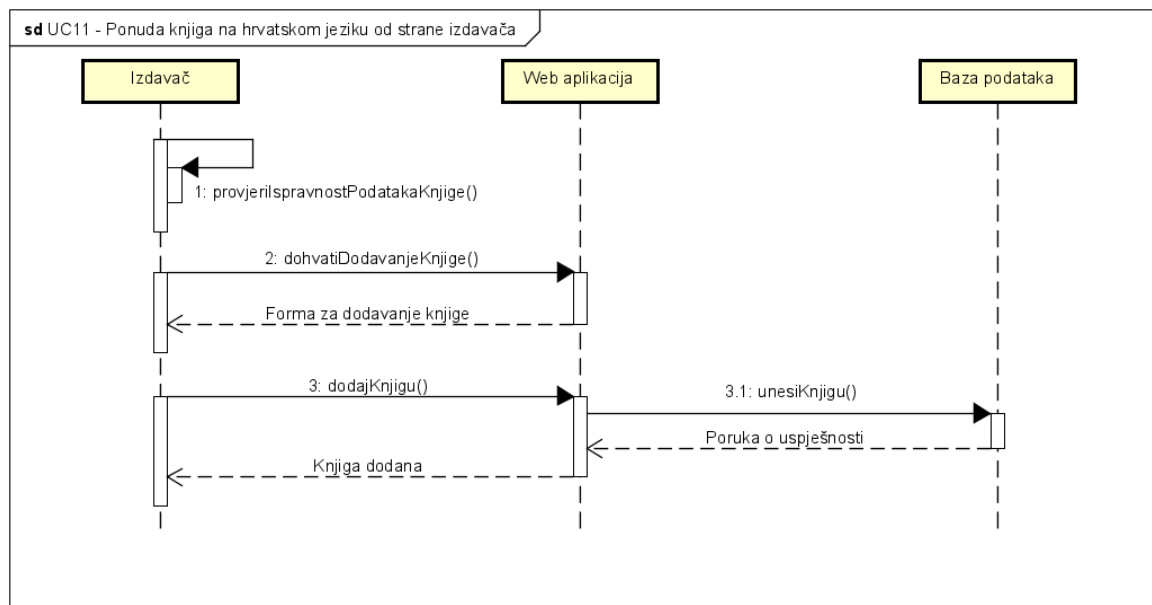
Neregistrirani ponuditelj odabire opciju registracije. Na ekranu mu se prikazuje forma za registraciju ponuditelja. Ponuditelj unosi potrebne podatke za registraciju, a to su: vrsta ponuditelja(izdavač, antikvarijat, preprodavač), naziv, e-pošta, adresa i broj telefona. Ponuditelj šalje podatke za registraciju klikom na odgovarajuće dugme. Web aplikacija šalje podatke administratoru. Administrator pregledava podatke. Ako adresa antikvarijata nije u Republici Hrvatskoj administrator šalje poruku da adresa ne odgovara uvjetima. Web aplikacija onemogućava nastavak registracije i šalje poruku da adresa ne odgovara uvjetima. Ako adresa preprodavača nije u srodnoj zemlji jezika administrator šalje poruku da adresa ne odgovara uvjetima. Web aplikacija onemogućava nastavak registracije i šalje poruku ponuditelju da adresa ne odgovara uvjetima. Ako korisničko ime već postoji u sustavu administrator šalje poruku da ponuditelj treba izmijeniti korisničko ime. Web aplikacija onemogućava nastavak registracije i šalje poruku ponuditelju da promijeni korisničko ime. Ako su podatci ispravni, administrator šalje poruku da su podatci ispravni. Podatci se unose u bazu podataka. Web aplikacija šalje poruku ponuditelju da je njegova registracija uspješno izvršena.



Slika 3.5: Sekvencijski dijagram za UC5

**Obrazac uporabe UC11 - Ponuda knjiga na hrvatskom jeziku od strane izdavača**

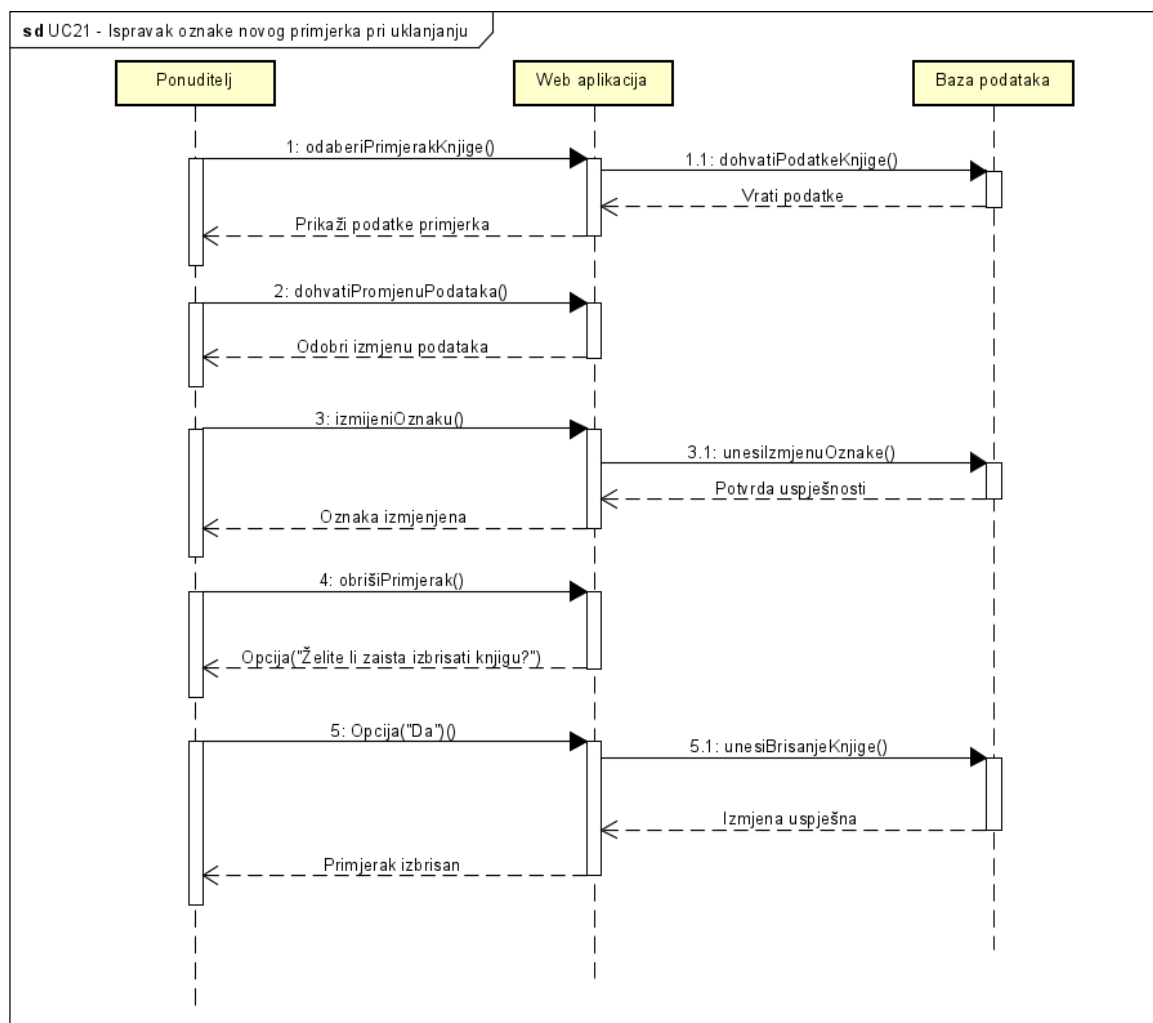
Izdavač provjerava ispravnost podataka knjige, to je njegova odgovornost. Nakon provjere podataka knjige, izdavač odabire opciju dodavanja knjige. Na ekranu mu se pokazuje forma za dodavanje knjige. Izdavač unosi značajke knjige i odabire dugme za dodavanje knjige. Web aplikacija šalje upit prema bazi podataka. U bazu podataka se sprema knjiga. Izdavač prima poruku da je njegova knjiga uspješno dodana.



Slika 3.6: Sekvencijski dijagram za UC11

**Obrazac uporabe UC21 - Ispravak oznake novog primjerka pri uklanjanju**

Ponuditelj odabire primjerak knjige kojem želi ispraviti oznaku prije uklanjanja. Web aplikacija dohvaća podatke iz baze podataka i prikazuje ih ponuditelju. Ponuditelj odabire opciju promjene podataka klikom na dugme. Web aplikacija odobrava izmjenu podataka. Ponuditelj unosi izmijenjenu oznaku knjige i sprema ju klikom na dugme za spremanje podataka. Web aplikacija šalje upit i sprema izmijenjenu oznaku u bazu podataka. Ponuditelj odabire opciju uklanjanja knjige klikom na dugme. Web aplikacija mu šalje opciju u slučaju da je slučajno kliknuo na dugme za uklanjanje. Ponuditelj odabire opciju uklanjanja knjige klikom na dugme "Da". Web aplikacija šalje upit i uklanja primjerak knjige iz baze podataka. Ponuditelj prima poruku da je uklanjanje primjerka odabrane knjige uspješno izvršeno.



Slika 3.7: Sekvencijski dijagram za UC21

## 3.2 Ostali zahtjevi

- Sustav treba omogućiti rad više korisnika u stvarnom vremenu
- Korisničko sučelje i sustav moraju podržavati hrvatsku abecedu pri unosu i prikazu tekstualnog sadržaja
- Izvršavanje dijela programa u kojem se pristupa bazi podataka ne smije trajati duže od nekoliko sekundi
- Sustav treba biti implementiran kao web aplikacija koristeći objektno-orijentirane jezike
- Neispravno korištenje korisničkog sučelja ne smije narušiti funkcionalnost i rad sustava
- Sustav treba biti jednostavn za korištenje, korisnici se moraju znati koristiti sučeljem bez detaljnih uputa
- Nadogradnja sustava ne smije narušavati postojeće funkcionalnosti sustava
- Sustav kao valutu koristi EUR
- Veza s bazom podataka mora biti kvalitetno zaštićena, brza i otporna na vanjske greške
- Pristup sustavu mora biti omogućen iz javne mreže pomoću HTTPS

## 4. Arhitektura i dizajn sustava

Da bismo razradili arhitekturu web aplikacije za poboljšanje dostupnosti knjiga prevedenih na hrvatski i srodne jezike, detaljnije ćemo opisati njene komponente, organizaciju i međusobnu komunikaciju. Evo ključnih elemenata:

### 4.1 Komponente

#### Stil arhitekture

Stil arhitekture bi mogao biti mikroservisni ili slojeviti (npr. MVC - Model-View-Controller). Mikroservisni pristup omogućava modularnost i lako skaliranje, dok slojeviti pristup olakšava razvoj i održavanje. Prilikom razvoja naše aplikacije odabran je MVC model.

#### Podsustavi

Podsustavi se mogu podijeliti na:

- Korisničko sučelje (UI): Front-end komponenta zadužena za interakciju s korisnicima.
- Poslovna logika: Središnji dio koji upravlja funkcionalnostima aplikacije.
- Baza podataka: Spremište podataka gdje se čuvaju informacije o knjigama, korisnicima, ponuditeljima, itd.
- Autentikacija i autorizacija: Sustav za upravljanje korisničkim pristupima i ovlastima.
- Integracija vanjskih usluga: Za funkcionalnosti poput prikaza karte (npr. integracija OpenStreetMap).

## Preslikavanje na radnu platformu

Komponente se raspoređuju na servere i klijentske uređaje:

- Server: Hostira poslovnu logiku, bazu podataka, autentikaciju i API za integraciju vanjskih usluga.
- Klijent: Uređaji korisnika (mobiteli, tableti, računala) s web preglednikom za pristup UI.

## Spremišta podataka

Centralizirana baza podataka (SQL ili NoSQL) koja pohranjuje:

- Podatke o knjigama.
- Informacije o korisnicima i ponuditeljima.
- Zahtjeve za prijevod i ponude.

## Mrežni protokoli

- HTTP/HTTPS: Za komunikaciju između klijenta i servera.
- API protokoli (REST, GraphQL): Za upite i manipulaciju podacima.

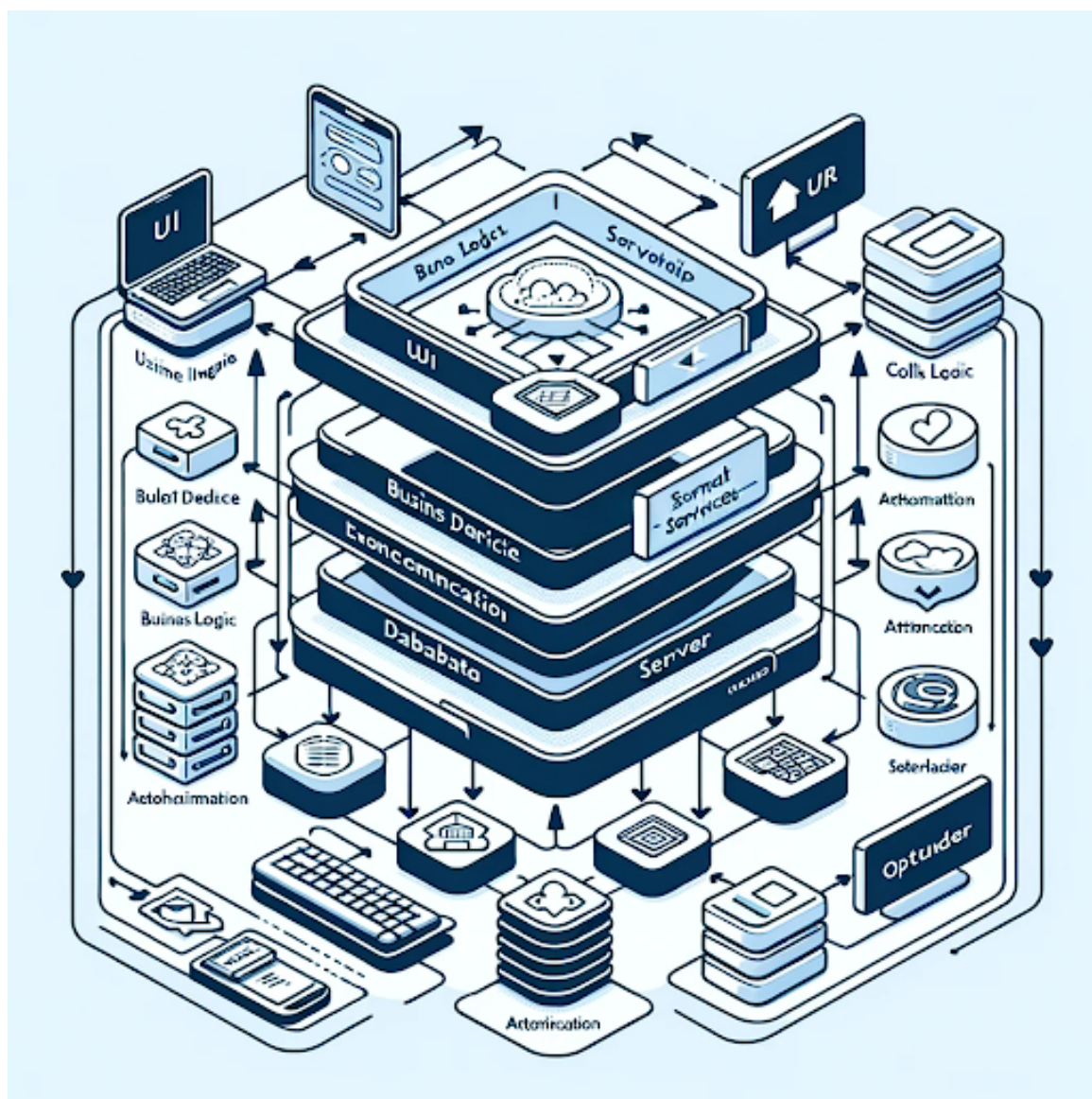
## Globalni upravljački token

Sustav bi koristio zahtjev-odgovor model za komunikaciju između klijenta i servera. Autentikacija i autorizacija korisnika odvijaju se na početku sesije.

## Sklopovsko-programski zahtjevi

- Server: Pouzdan i skalabilan, s dovoljno resursa za obradu zahtjeva i pohranu podataka.
- Klijent: Kompatibilnost s modernim web preglednicima i prilagodljivost različitim veličinama ekrana.
- Prikazuje raspored glavnih komponenti: UI, poslovna logika, baza podataka, autentikacija.
- Ilustrira komunikaciju između klijenta i servera.





Slika 4.1: Osnova arhitekture

Skica ilustrira sljedeće komponente i njihovu međusobnu komunikaciju:

- Korisničko sučelje (UI): Na klijentskim uređajima (mobitel, tablet, računalo).
- Poslovna logika: Dio koji se nalazi na serveru.
- Baza podataka: Spremište podataka također na serveru.
- Autentikacija: Dio za upravljanje korisničkim pristupom.
- Integracija vanjskih usluga: Na primjer, OpenStreetMap za prikaz karata.

Izbor Model-View-Controller (MVC) arhitekture za ovu web aplikaciju temelji se na nekoliko ključnih principa oblikovanja koje sam uzimao u obzir:

## 4.2 Izbor arhitekture

### Separacija briga (Separation of concerns)

MVC arhitektura odlično odvajaju različite aspekte aplikacije:

- **Model:** Predstavlja podatke i poslovnu logiku aplikacije. U ovom slučaju, model bi upravljao podacima knjiga, korisnika, ponuda i zahtjeva za prijevod.
- **View:** Odnosi se na korisničko sučelje. U našoj aplikaciji, ovo bi bilo mjesto gdje korisnici pretražuju knjige, pregledavaju ponude, i komuniciraju sa sustavom.
- **Controller:** Djeluje kao posrednik između Modela i Viewa, obrađuje korisničke zahtjeve, ažurira model i konačno šalje podatke natrag viewu.

Ova jasna podjela pomaže u organizaciji koda, olakšava održavanje i omogućava lakše testiranje pojedinih komponenti.

### Skalabilnost i fleksibilnost

MVC podržava skalabilnost. Kako se aplikacija razvija, svaki dio (Model, View, Controller) može se neovisno skalirati ili modificirati. Ovo je posebno korisno za aplikacije koje imaju potencijal rasti, kao što je naš slučaj s web platformom za knjige.

### Podrška za više klijentskih platformi

S obzirom na to da se aplikacija mora prilagoditi različitim uređajima (mobilnim telefonima, tabletima, računalima), MVC omogućava razvoj različitih vrsta viewova koji mogu koristiti isti model i controller logiku. To znači da možemo imati različite korisničke sučelja za mobilne uređaje i desktop računala, dok se poslovna logika i obrada podataka ne mijenjaju.

## Održavanje i nadogradnja

MVC struktura pojednostavljuje ažuriranja i održavanje. Ako trebate promijeniti poslovnu logiku, to se može učiniti u modelu bez utjecaja na view. Slično, izmjene u korisničkom sučelju ne utječu na poslovnu logiku aplikacije.

## Zajednica i resursi

MVC je popularan i dobro dokumentiran arhitekturni obrazac. Postoji obilje resursa, biblioteka i okvira koji podržavaju MVC, što može značajno ubrzati razvoj i pružiti bolju podršku tijekom razvojnog procesa.

Zbog ovih razloga, MVC se čini kao optimalan izbor za arhitekturu ove web aplikacije, pružajući dobru osnovu za jasno strukturiran, održiv i fleksibilan razvoj.

Organizacija sustava za opisanu web aplikaciju s najviše razine apstrakcije može se podijeliti u četiri glavne komponente: klijent-poslužitelj model, baza podataka, datotečni sustav, i grafičko sučelje. Ove komponente zajedno omogućavaju funkcionalnost, skalabilnost i korisničku interakciju potrebnu za uspješan rad aplikacije.

## 4.3 Organizacija sustava

### Klijent-poslužitelj model

U klijent-poslužitelj modelu, aplikacija je podijeljena na dva glavna dijela: klijent (front-end) koji se izvodi na korisničkom uređaju i poslužitelj (back-end) koji obrađuje zahtjeve i šalje odgovore.

- Klijent (Front-end): Koristi web preglednik ili mobilnu aplikaciju za pristup aplikaciji. Odgovoran je za prikupljanje korisničkih ulaza, prikazivanje podataka korisnicima i slanje zahtjeva poslužitelju.
- Poslužitelj (Back-end): Hostira aplikaciju, obrađuje zahtjeve, vrši operacije nad bazom podataka, i vraća rezultate klijentu. Također upravlja autentikacijom, autorizacijom, i sigurnošću.

### Baza podataka

Baza podataka je središnje mjesto za pohranu svih podataka koji su relevantni za aplikaciju. Uključuje:

- Podatke o knjigama: Nazive, autore, žanrove, godine izdanja, itd.
- Korisničke podatke: Informacije o registriranim i neregistriranim korisnicima.
- Ponude i zahtjeve: Informacije o dostupnosti knjiga, ponuditeljima, i zahtjevima za prijevod.

### Datotečni sustav

Datotečni sustav se koristi za pohranu i upravljanje datotekama koje nisu direktno povezane s bazom podataka, poput:

- Slika korica knjiga: Pohranjene kao datoteke koje se mogu prikazivati u korisničkom sučelju.
- Dokumenti: Uključujući priručnike, upute za korisnike, i druge dokumente.

## Grafičko sučelje

Grafičko sučelje (GUI) je ono što korisnik vidi i s čime interagira. Uključuje:

- Web stranice: Dizajnirane za jednostavno korištenje i navigaciju, s funkcionalnostima kao što su pretraga, pregled knjiga, i slanje zahtjeva.
- Mobilno sučelje: Prilagođeno za manje ekrane i osjetljivo na dodir, s sličnim funkcionalnostima kao web verzija.

Svaka od ovih komponenti igra ključnu ulogu u uspješnom funkcioniranju aplikacije, omogućujući efikasnu obradu podataka, intuitivnu korisničku interakciju i sigurnost informacija.

## 4.4 Organizacija aplikacije

Organizacija aplikacije koja koristi Model-View-Controller (MVC) arhitekturu s jasno definiranim frontend i backend slojevima omogućava efikasno upravljanje kodom i funkcionalnostima. Evo kako bi se to moglo strukturirati:

### Frontend (klijentska strana)

Frontend je ono što korisnik vidi i s čime interagira. U MVC arhitekturi, to uglavnom obuhvaća "View" komponentu.

- Tehnologije: HTML, CSS, JavaScript, i frameworki poput React, Angular ili Vue.js.
- Zadaće:
  - Prikaz podataka: Dinamički prikazuje podatke dobivene s backend-a.
  - Korisničko sučelje: Omogućava interakciju s korisnikom, prikuplja korisničke zahtjeve i šalje ih backend-u.
  - Odgovor na korisničke akcije: Ažuriranje sučelja na temelju korisničkih interakcija i podataka s backend-a.

### Backend (poslužiteljska strana)

Backend se bavi obradom podataka, poslovnom logikom i interakcijom s bazom podataka. U MVC arhitekturi, to uključuje "Model" i "Controller" komponente.

- Tehnologije: Server-side jezici poput Node.js, Python (Django, Flask), Ruby on Rails, Java (Spring), itd.
- Zadaće:
  - Controller:
    - \* Upravlja zahtjevima s frontend-a.
    - \* Prosljeđuje podatke između View-a i Modela.
    - \* Upravlja tokom podataka i obradi zahtjeva.
  - Model:
    - \* Predstavlja poslovnu logiku i podatke.

- \* Interakcija s bazom podataka.
- \* Obrada podataka (CRUD operacije - Create, Read, Update, Delete).

### **Komunikacija između frontend-a i backend-a**

Komunikacija između frontend-a i backend-a obično se odvija preko API-ja (Application Programming Interface), koristeći HTTP/HTTPS protokole.

- API (najčešće REST ili GraphQL): Definira kako frontend šalje zahtjeve backend-u i kako backend vraća odgovore.
- JSON format: Često korišten format za slanje podataka između frontend-a i backend-a.

### **Baza podataka**

Iako tehnički nije dio MVC strukture, baza podataka je ključna komponenta koja surađuje s Modelom. Ona pohranjuje sve podatke potrebne za aplikaciju.

- Tehnologije: SQL (MySQL, PostgreSQL) ili NoSQL (MongoDB, Cassandra) baze podataka.
- Zadaće: Pohrana i dohvaćanje podataka potrebnih za aplikaciju.

U ovakvoj organizaciji, svaki sloj ima jasno definiranu ulogu, što doprinosi boljem razumijevanju koda, lakšem održavanju i skaliranju aplikacije. Frontend i backend mogu se neovisno razvijati i optimizirati, što pruža fleksibilnost i efikasnost u razvojnem procesu.



## 4.5 Baza podataka

### 4.5.1 Opis tablica

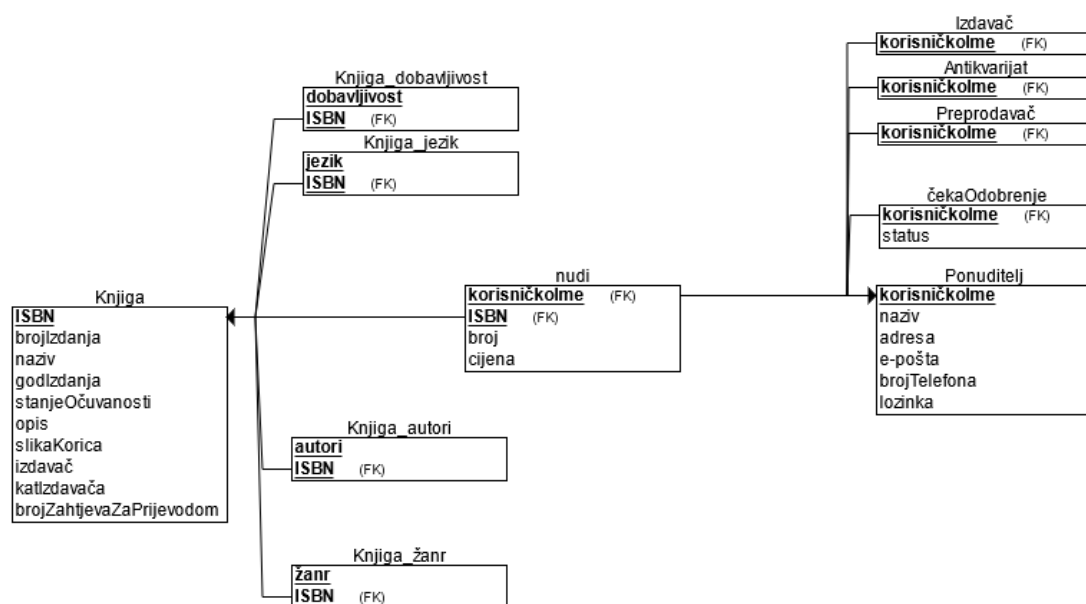
Korisnik		
	VARCHAR(13)	
userId	INT	Jedinstveni identifikator, autogeneriran od strane baze
username	VARCHAR	Jedinstveno korisničko ime
password	VARCHAR	Jedinstveno lozinka
naziv	VARCHAR	Naziv korisnika
adresa	VARCHAR	Adresa korisnika
telefon	INT	Telefon korisnika
email	VARCHAR	Email korisnika
tip	VARCHAR	Tip korisnika
Knjiga		
	VARCHAR(13)	
bookId	INT	Jedinstveni identifikator, autogeneriran od strane baze
naslov	VARCHAR	Naziv knjige
autor	VARCHAR	Ime autora
izdavač	VARCHAR	Ime izdavača
brojIzdanja	INT	Broj izdanja knjige
godIzdanja	INT	Godina izdanja knjige
kategorija	CHAR(3)	Kategorija izdavača
zahtjevi	INT	Zahtjev za prijevod knjige
očuvanost	VARCHAR	Očuvanost knjige

Nastavljeno na idućoj stranici

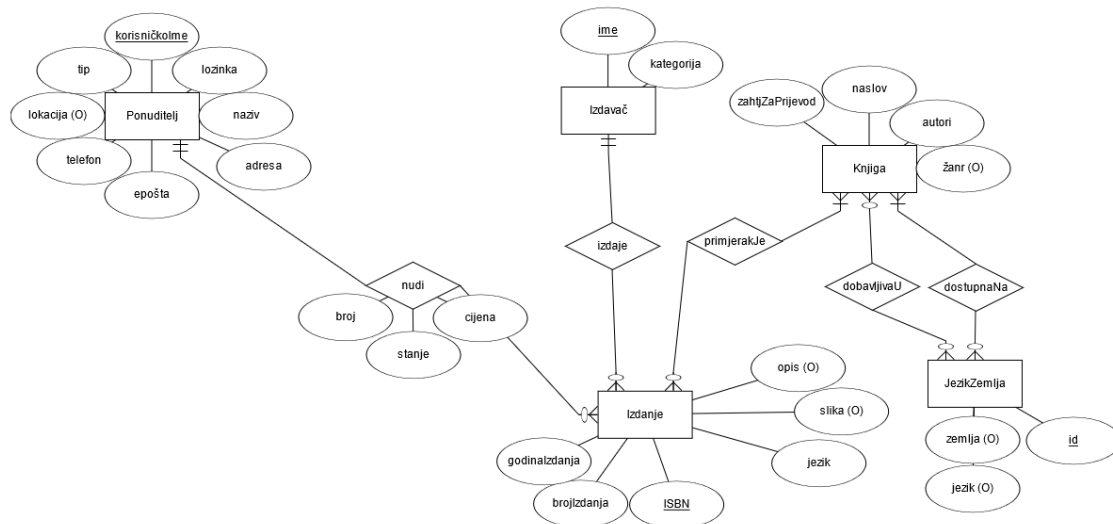
Nastavljeno od prethodne stranice

Korisnik		
opis	VARCHAR	Kratki opis knjige
slika	VARCHAR	Prikaz knjige
žanr	VARCHAR	Žanr knjige
isbn	INT	ISBN knjige
Ponuda		
	VARCHAR(32)	
offerId	INT	Jedinstveni identifikator, autogeneriran od strane baze
nazivPonuditelja	VARCHAR	Naziv ponuditelja
brojPrimjeraka	INT	Broj primjeraka knjige
cijena	INT	Cijena knjige

## 4.5.2 Dijagram baze podataka



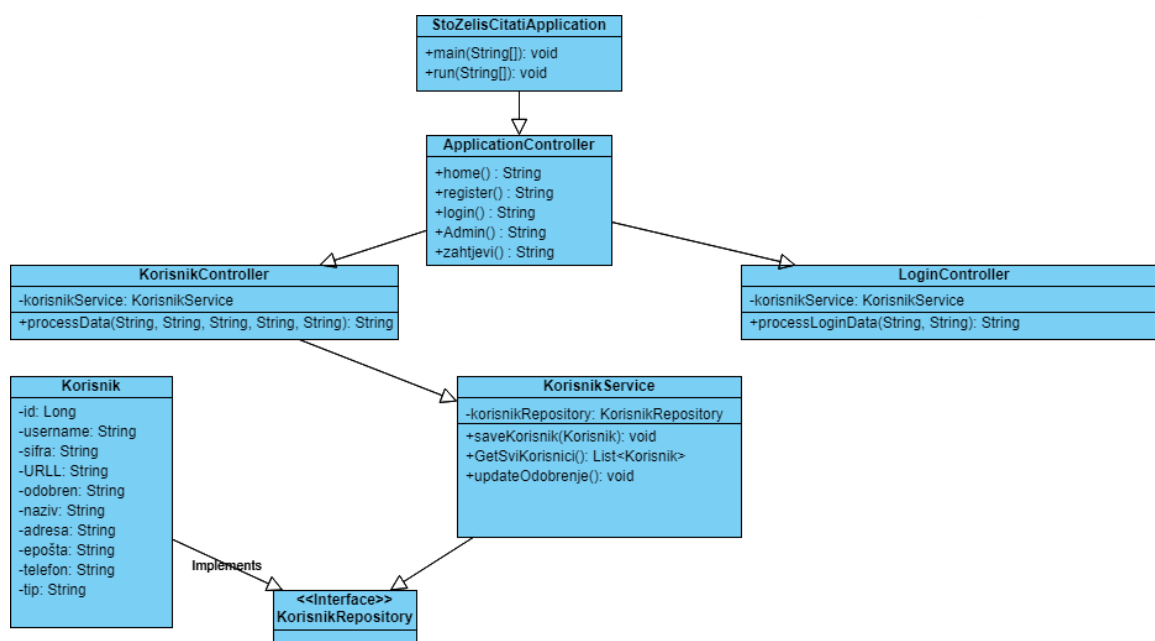
Slika 4.2: REL dijagram baze podataka



Slika 4.3: ER dijagram baze podataka

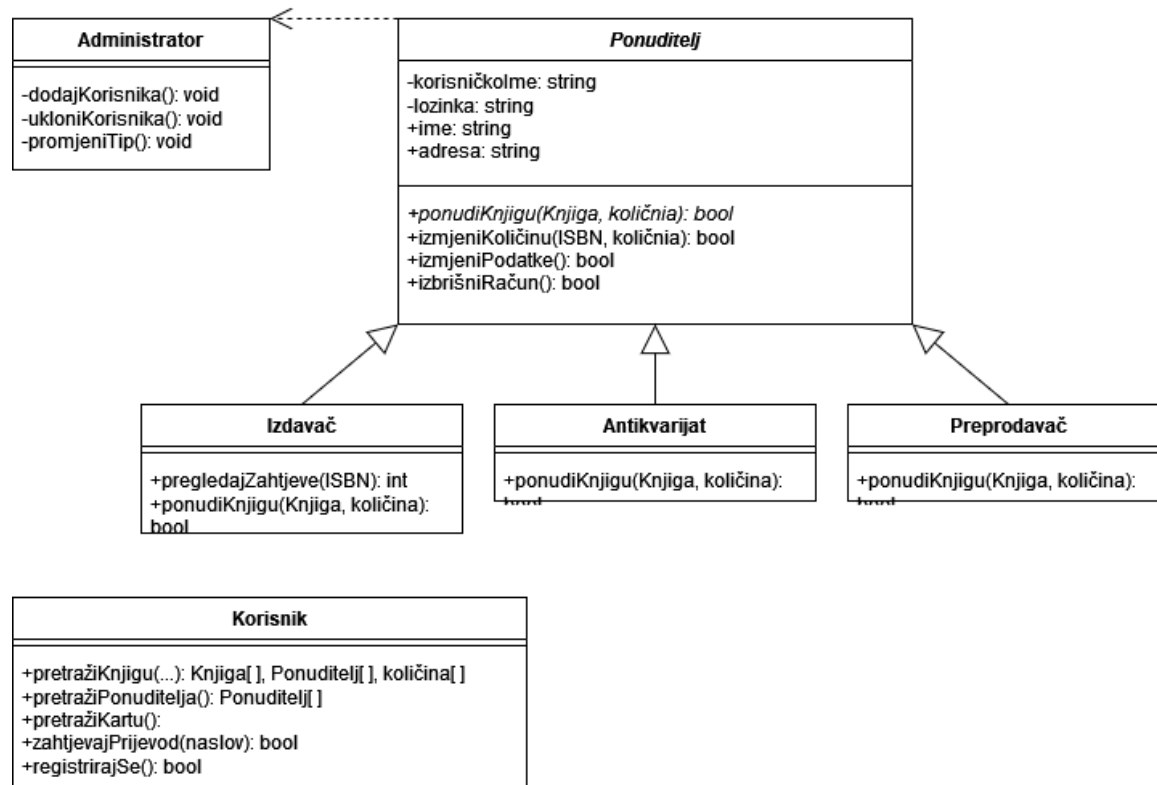
## 4.6 Dijagram razreda

Idejno razrađeni dijagram razreda opisuje strukturu web aplikacije. Uključuje ApplicationController za upravljanje zahtjevima vezanim uz navigaciju na web stranici, KorisnikController za obrađivanje korisničkih zahtjeva, LoginController za autentikaciju korisnika te Korisnik za predstavljanje entiteta korisnika. KorisnikService brine o logici poslovne logike vezane uz korisnike, dok KorisnikRepository predstavlja sučelje za pristup podacima korisnika u bazi podataka. StoZelisCitatiApplication predstavlja glavni razred aplikacije s main metodom za pokretanje, dok su veze između razreda usmjerene prema ovisnostima i suradnji između njih. Ovo omogućava bolje razumijevanje strukture aplikacije, odvojenost odgovornosti i međusobnu interakciju između komponenti.

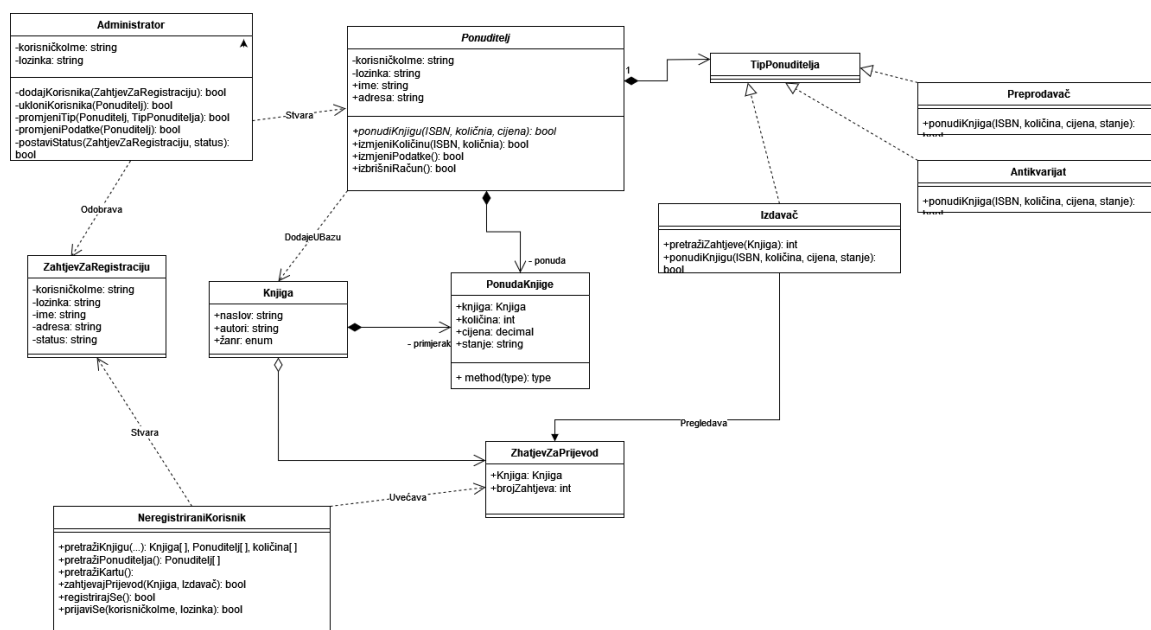


Slika 4.4: Dijagram razreda

Idejno razrađeni dijagram razreda opisuje sve tri vrste ponuditelja te njihove funkcionalnosti. Predstavljena je ovisnost i suradnja između razreda.



Slika 4.5: Dijagram razreda

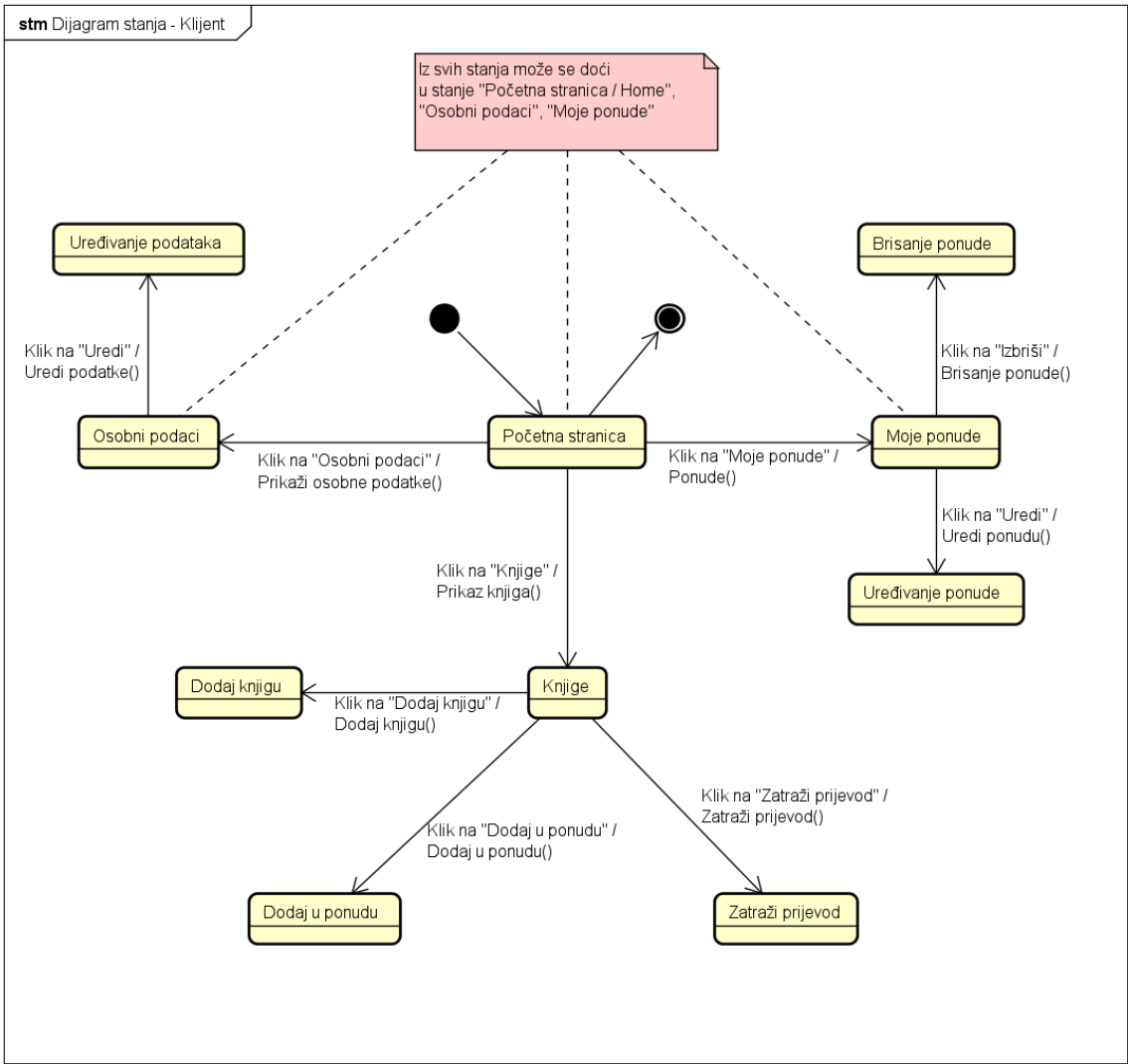


Slika 4.6: Dijagram razreda

## 4.7 Dijagram stanja

Dijagram stanja se koristi za modeliranje ponašanja sustava ili entiteta s obzirom na različita stanja kroz koje može proći. Sadrži stanja, prijelaze između stanja, događaje koji pokreću prijelaze te akcije koje se izvršavaju u svakom stanju ili prijelazu. Na slici je prikazan dijagram stanja za registriranog korisnika. Nakon uspješne prijave, korisniku se prikazuje početna stranica na kojoj može vidjeti svoje ponude knjiga te napraviti novu ponudu knjige. Za odabranu knjigu potrebno je provjeriti treba li zatražiti prijevod iste ili ne te zatim unijeti potrebne podatke i zaključiti ponudu. Korisniku se klikom na "Osobni podaci" prikazuju njegovi podaci koje može urediti, a klikom na "Moje ponude" može pregledati sve svoje ponude knjiga.

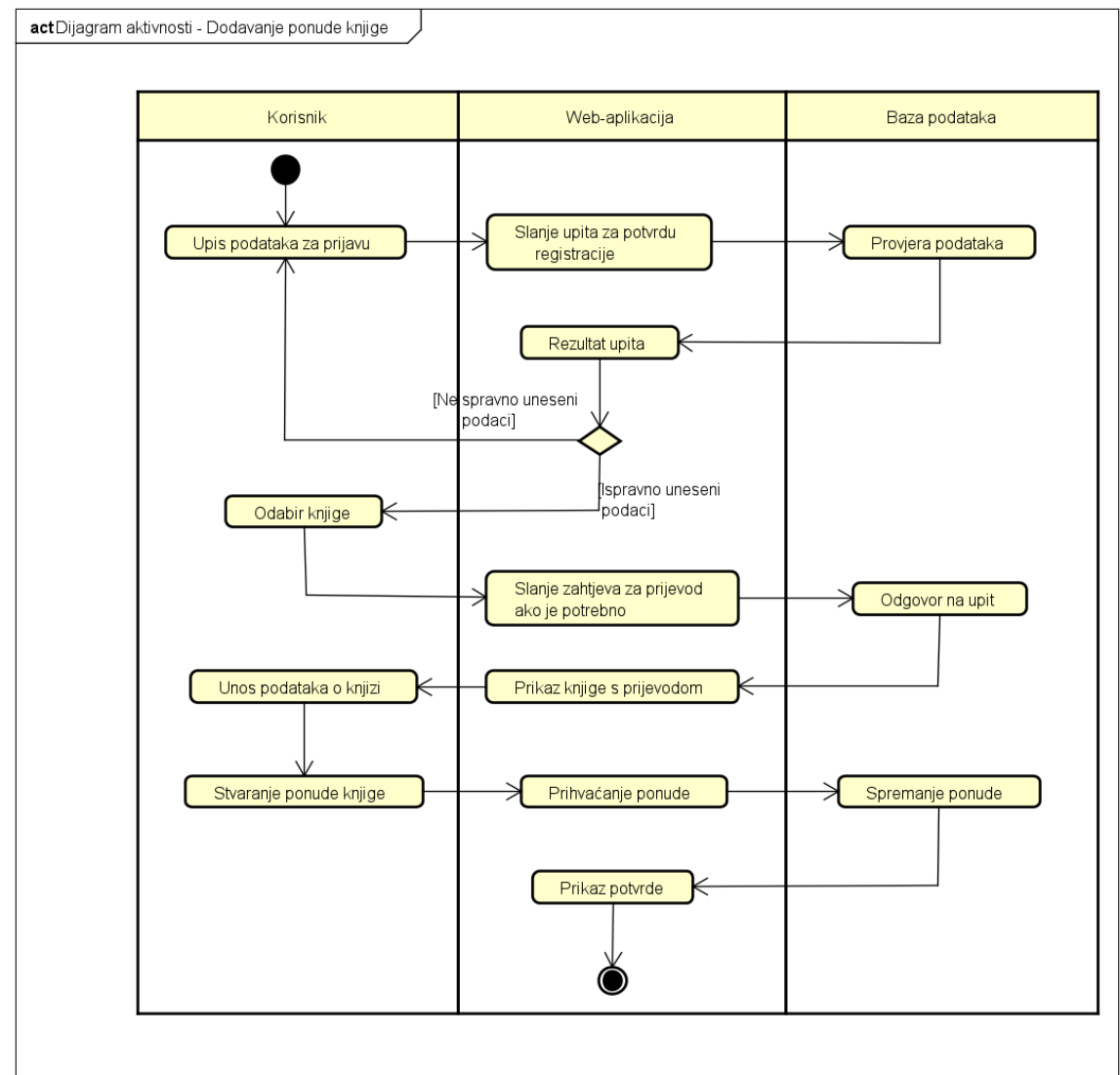




Slika 4.7: Dijagram stanja

## 4.8 Dijagram aktivnosti

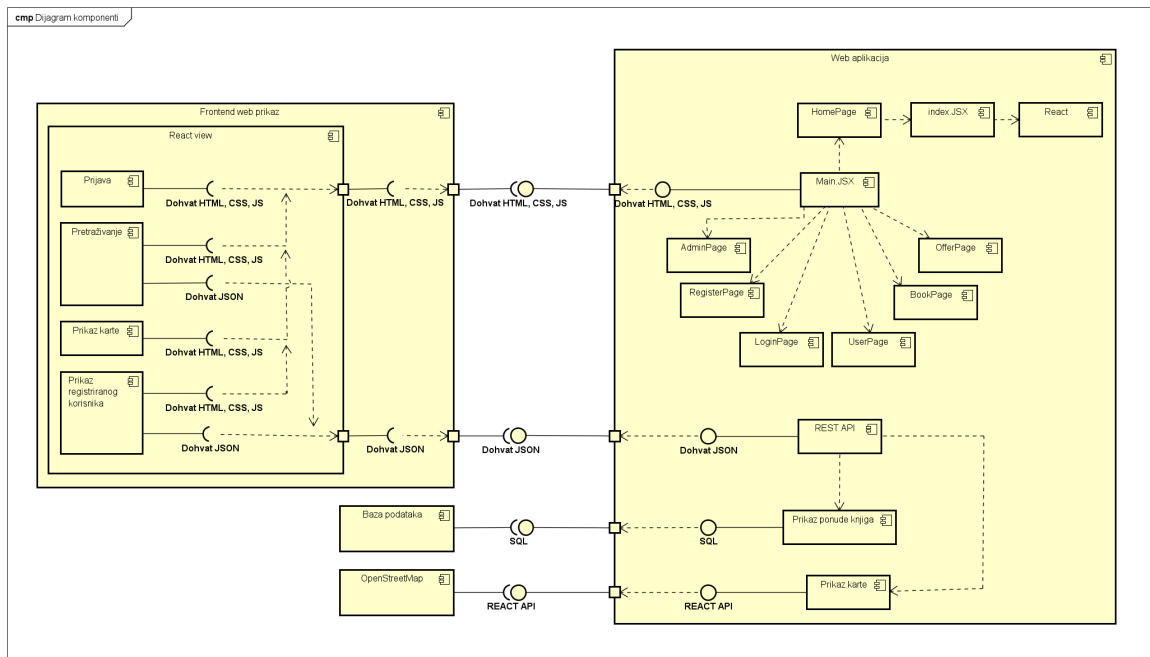
Dijagram aktivnosti se koristi za modeliranje poslovnih procesa, algoritama ili drugih sekvenci aktivnosti unutar sustava što je vrlo korisno za opisivanje tokova rada ili procesa izvođenja zadataka. Sadrži stanja, aktivnosti, odluke, ulazne i izlazne tokove te usklađivačke elemente koji pomažu u opisu redoslijeda aktivnosti. Na sljedećem dijagramu prikazan je proces kreiranja nove ponude knjige. Korisnik, nakon prijave u sustav te odobravanja iste, odabire knjigu koju želi dodati u ponudu. Također će zatražiti zahtjev za prijevod ako je to potrebno. Nakon unosa svih potrebnih podataka, korisnik zaključuje ponudu.



Slika 4.8: Dijagram aktivnosti

## 4.9 Dijagram komponenti

Dijagram komponenti je vrsta strukturnog dijagrama koja prikazuje logičke komponente sustava i njihove odnose. Na ovom dijagramu su vdiljive komponente Prijava, Pretraživanje, Prikaz karte i Prikaz registriranog korisnika koje djeluju unutar React view komponente unutar Frontend web prikaza komponente. One su preko sučelja za dohvat HTML, CSS i JS povezane s komponentom Main.JSX koja funkcionira kao router koji upravlja prikazom i funkcionalnošću grafičkog prikaza. Ovisno o potrebi upravlja ih na komponente AdminPage, RegisterPage, LoginPage, UserPage, BookPage, OfferPage, te HomePage. Sve te komponente se nalaze u Web aplikaciji. Dodatno u Web aplikaciji je komponenta REST API koja je povezana s komponentama za pretraživanje i prikaz registriranog korisnika preko sučelja dohvat JSON. Ona ovisno o potrebi se nadovezuje na Prikaz ponude knjiga koja je preko sučelja SQL povezana s vanjskom komponentom Baza podataka. Ili Prikaz karte koja je preko sučelja REACT API povezana s vanjskom komponento OpenStreetMap.



Slika 4.9: Dijagram komponenti

## 5. Implementacija i korisničko sučelje

### 5.1 Korištene tehnologije i alati

Komunikacija u timu odvijala se uživo te preko aplikacija Whatsapp i Discord.

UML dijagrami izrađeni su u aplikaciji Astah<sup>1</sup>, koja izdaje besplatnu student-sku licencu, te pomoću stranica VisualParadigm<sup>2</sup> i draw.io<sup>3</sup>. Dijagram baze podataka izrađen je u online alatu ERDPlus<sup>4</sup>. Git<sup>5</sup> je služio kao sustav za upravljanje izvornim kodom, a udaljeni repozitorij projekta dostupan je na GitHubu.

Razvojno okruženje bio je Microsoft Visual Studio<sup>6</sup>, a korišten je i uređivač koda Visual Studio Code<sup>7</sup>, Microsoftov program pogodan za izradu web aplikacija, koji, iako nije IDE, proširenjima (extensions) nudi podršku za mnoge programske jezike i alate kao što su debuggeri i intellisense.

Baza podataka izrađena je u sustavu Postgres<sup>8</sup>, koji proširuje jezik SQL i ima reputaciju pouzdanog i moćnog alata za upravljanje bazama podataka, a deployana je na poslužitelju u oblaku Render<sup>9</sup>. Backend je napisan u Javi te je korišten Spring Boot<sup>10</sup>. Spring Boot je framework temeljen na Java frameworku, dizajniran da automatskom konfiguracijom i metodom convention-over-configuration pojednostavi izradu aplikacija.

Za frontend je upotrebljen su standardni jezici HTML, CSS i JavaScript te React<sup>11</sup>, široko upotrebljavana biblioteka za izradu web i native korisničkih sučelja, temeljena na izradni pojedinačnih, jednostavnijih komponenti koje zajedno čine složeni UI, a koju održava Meta.

Pri izradi, to jest lokalnom testiranju, korišten je Vite<sup>12</sup>. Vite je lokalni server

---

<sup>1</sup><https://astah.net>

<sup>2</sup><https://online.visual-paradigm.com>

<sup>3</sup><https://app.diagrams.net>

<sup>4</sup><https://erdplus.com>

<sup>5</sup><https://github.com>

<sup>6</sup><https://visualstudio.microsoft.com>

<sup>7</sup><https://code.visualstudio.com>

<sup>8</sup><https://www.postgresql.org>

<sup>9</sup><https://render.com>

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

<sup>11</sup><https://react.dev>

<sup>12</sup><https://vitejs.dev>

koji prati promjene u datoteci i nakon spremanja automatski osvježava browser kako bi te promjene odmah bile vidljive.

Za ostvarenje interaktivne karte i geokodiranje, korištene su usluge platforme Mapbox<sup>13</sup>.

---

<sup>13</sup><https://www.mapbox.com>

## 5.2 Ispitivanje programskog rješenja

### 5.2.1 Ispitivanje komponenti

Sastavne programske komponente formirane su od nekoliko jedinica, uključujući osnovne dijelove programa poput razreda, pripadajućih metoda i atributa, koje međusobno intenzivno surađuju. Interakcija se ostvaruje kroz proceduralna sučelja, gdje svaka komponenta enkapsulira određeni skup funkcionalnosti koje druge komponente mogu pozivati. Evaluacija komponenata izvodi se s pomoću Junit radnog okvira, koji automatizira proces testiranja. Upotrebom dviju vrsta testova, jednih u uobičajenim uvjetima rada i drugih s unosom neispravnih i rubnih uvjeta, analizira se očekivano ponašanje razreda i unaprijed definirani izlaz metoda. Za testiranje BookService metoda koristili smo sučelja Knjiga, Ponuda, i BookRepository. Prvo smo testirali saveBook metodu.

```
@Test
@DisplayName("Testiranje saveBook metode")
void testSaveBook() {
    // Priprema podataka
    Knjiga knjiga = new Knjiga();
    List<Ponuda> ponude = new ArrayList<>();
    knjiga.setNaziv("Naslov knjige");
    knjiga.setIsbn("100");
    knjiga.setAutor("autor");
    knjiga.setIzdavac("izdavac");
    knjiga.setBrojIzdanja(1);
    knjiga.setOpis("opis");
    knjiga.setGodIzdanja(2005);
    knjiga.setSlikaURL("url");
    knjiga.setId(1L);
    knjiga.setOznaka("oznaka");
    knjiga.setKategorija("kategorija");
    knjiga.setStanjeOcuvanosti("st_oc");
    knjiga.setZahtjevi(50);
    knjiga.setZanr("zanr");
    knjiga.setPonude(ponude);

    when(bookRepository.save(any(Knjiga.class))).thenReturn(knjiga);

    Knjiga savedKnjiga = bookService.saveBook(knjiga);

    verify(bookRepository, times(1)).save(knjiga);
    assertNotNull(savedKnjiga);
    assertEquals("Naslov knjige", savedKnjiga.getNaziv());
    assertEquals("100", savedKnjiga.getIsbn());
```



```
assertEquals("autor", savedKnjiga.getAutor());
assertEquals("izdavac", savedKnjiga.getIzdavac());
assertEquals(1, savedKnjiga.getBrojIzdanja());
assertEquals("opis", savedKnjiga.getOpis());
assertEquals(2005, savedKnjiga.getGodIzdanja());
assertEquals("url", savedKnjiga.getSlikaURL());
assertEquals(1L, savedKnjiga.getId());
assertEquals("oznaka", savedKnjiga.getOznaka());
assertEquals("kategorija", savedKnjiga.getKategorija());
assertEquals("st_oc", savedKnjiga.getStanjeOcuvanosti());
assertEquals(50, savedKnjiga.getZajtjevi());
assertEquals("zanr", savedKnjiga.getZanr());
assertNotNull(savedKnjiga.getPonude());

}
```

A zatim smo testirali deleteBook metodu.

```
@Test
@DisplayName("Testiranje deleteBook metode.")
void testDeleteBook() {
    Long bookIdToDelete = 1L;
    bookService.deleteBook(bookIdToDelete);
    verify(bookRepository).deleteById(bookIdToDelete);
}
```

Zatim smo testirali KorisnikService. Za to smo koristili sučelja Korisnik, Ponuda, i KorisnikRepository.

Prvo smo testirali dohvaćanje korisnika.

```
@Test
@DisplayName("Test dohvacanja korisnika.")
void testGetUserById() {
    Long userId = 1L;
    Korisnik mockKorisnik = new Korisnik();
    List<Ponuda> ponude = new ArrayList<>();

    mockKorisnik.setId(userId);
    mockKorisnik.setUsername("username");
    mockKorisnik.setPassword("password");
    mockKorisnik.setEmail("email@gmail.com");
    mockKorisnik.setOdobren(false);
    mockKorisnik.setTip("tip");
    mockKorisnik.setAdresa("adresa");
    mockKorisnik.setTelefon("123456789");
    mockKorisnik.setPonude(ponude);
    mockKorisnik.setPrijavljen(false);
    mockKorisnik.setURL("url");
    mockKorisnik.setNaziv("naziv");

    when(korisnikRepository.findById(userId)).thenReturn(Optional.of(mockKorisnik));

    Korisnik result = korisnikService.getUserById(userId);
}
```

```
assertEquals(userId, result.getId());
assertEquals("adresa", result.getAdresa());
assertEquals("123456789", result.getTelefon());
assertEquals("tip", result.getTip());
assertEquals("email@gmail.com", result.getEmail());
assertEquals("password", result.getPassword());
assertEquals("username", result.getUsername());
assertEquals(false, result.getOdobren());
assertNotNull(ponude);
assertEquals(false, result.isPrijavljen());
assertEquals("url", result.getURLL());
assertEquals("naziv", result.getNaziv());
}
```

Zatim smo testirali funkcionalnost spremanja korisnika.

@Test

@DisplayName("Test spremanja korisnika.")

void saveUserTest(){

Long userId = 1L;

Korisnik mockKorisnik = new Korisnik();

List<Ponuda> ponude = new ArrayList<>();

mockKorisnik.setId(userId);

mockKorisnik.setUsername("username");

mockKorisnik.setPassword("password");

mockKorisnik.setEmail("email@gmail.com");

mockKorisnik.setOdobren(false);

mockKorisnik.setTip("tip");

mockKorisnik.setAdresa("adresa");

mockKorisnik.setTelefon("123456789");

mockKorisnik.setPonude(ponude);

mockKorisnik.setPrijavljen(false);

mockKorisnik.setURLL("url");

mockKorisnik.setNaziv("naziv");

when(korisnikRepository.save(any(Korisnik.class))).thenReturn(mockKorisnik);

Korisnik savedKorisnik = korisnikService.saveKorisnik(mockKorisnik);

verify(korisnikRepository, times(1)).save(mockKorisnik);

assertEquals(userId, savedKorisnik.getId());

assertEquals("adresa", savedKorisnik.getAdresa());

assertEquals("123456789", savedKorisnik.getTelefon());

assertEquals("tip", savedKorisnik.getTip());

assertEquals("email@gmail.com", savedKorisnik.getEmail());

assertEquals("password", savedKorisnik.getPassword());

assertEquals("username", savedKorisnik.getUsername());

assertEquals(false, savedKorisnik.getOdobren());

assertNotNull(ponude);

assertEquals(false, savedKorisnik.isPrijavljen());

assertEquals("url", savedKorisnik.getURLL());

assertEquals("naziv", savedKorisnik.getNaziv());

```
}
```

Nakon toga smo testirali ponudu, to jest OfferService. Za te testove smo koristili sučelja Knjiga, Korisnik, Ponuda, PonudaDto, BookRepository, KorisnikRepository, OfferRepository i OfferService. Prvo smo testirali brisanje ponude.

```
@Test
@DisplayName("Testiranje ^brisanja ^ponude.")
void deleteOfferTest() {
    Long offerId = 1L;

    offerService.deleteOffer(offerId);

    verify(offerRepository, times(1)).deleteById(offerId);
}
```

A zatim smo testirali spremanje ponude.

```
@Test
@DisplayName("Testiranje ^spremanja ^ponude.")
void saveOffer() {
    // Arrange
    PonudaDto ponudaDto = new PonudaDto();
    ponudaDto.setPonuditeljId(1L);
    ponudaDto.setKnjigaId(2L);
    ponudaDto.setCijena(100);
    ponudaDto.setBrojPrimjeraka(5);

    Korisnik mockKorisnik = new Korisnik();
    Knjiga mockKnjiga = new Knjiga();
    Ponuda mockPonuda = new Ponuda();

    when(korisnikRepository.findById(ponudaDto.getPonuditeljId())).thenReturn(java.util.Optional.of(mockKorisnik));
    when(bookRepository.findById(ponudaDto.getKnjigaId())).thenReturn(java.util.Optional.of(mockKnjiga));
    when(offerRepository.save(any(Ponuda.class))).thenReturn(mockPonuda);

    Ponuda resultPonuda = offerService.saveOffer(ponudaDto);

    verify(offerRepository).save(any(Ponuda.class));

    assertEquals(mockPonuda, resultPonuda);
}
```

Zatim smo testirali kada spremamo ponudu, a ne postoji taj userId.

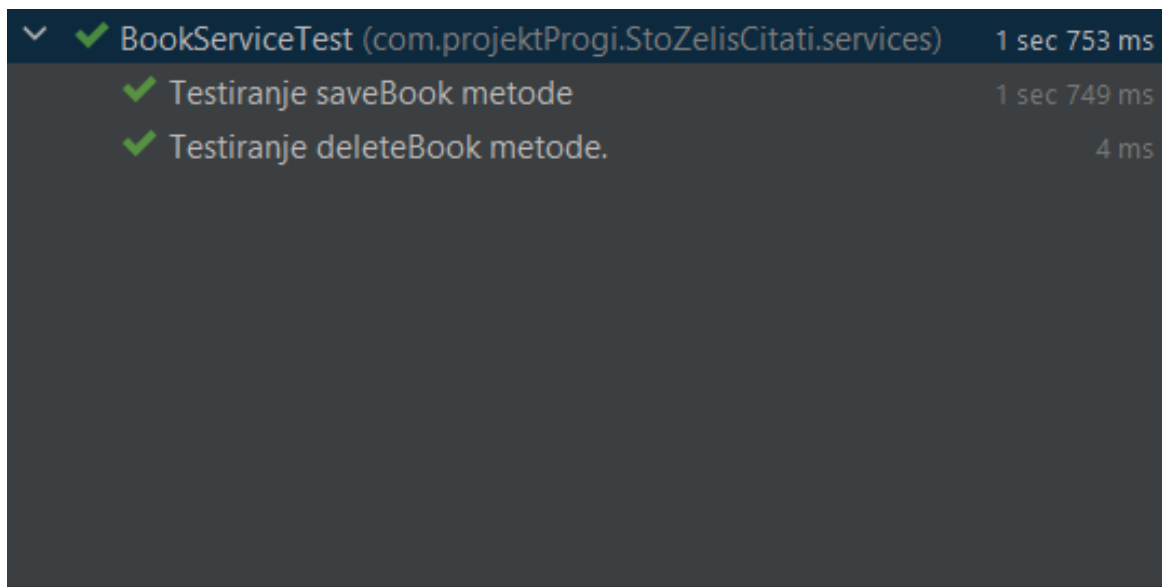
```
@Test
@DisplayName("Testiranje ^kada ^spremamo ^ponudu. ^a ^ne ^postoji ^taj ^userId.")
void saveOfferInvalid() {
    PonudaDto ponudaDto = new PonudaDto();
    ponudaDto.setPonuditeljId(1L);
    ponudaDto.setKnjigaId(2L);
    ponudaDto.setCijena(100);
    ponudaDto.setBrojPrimjeraka(5);
```

```
when(korisnikRepository.findById(ponudaDto.getPonuditeljId())).thenReturn(Optional.empty());

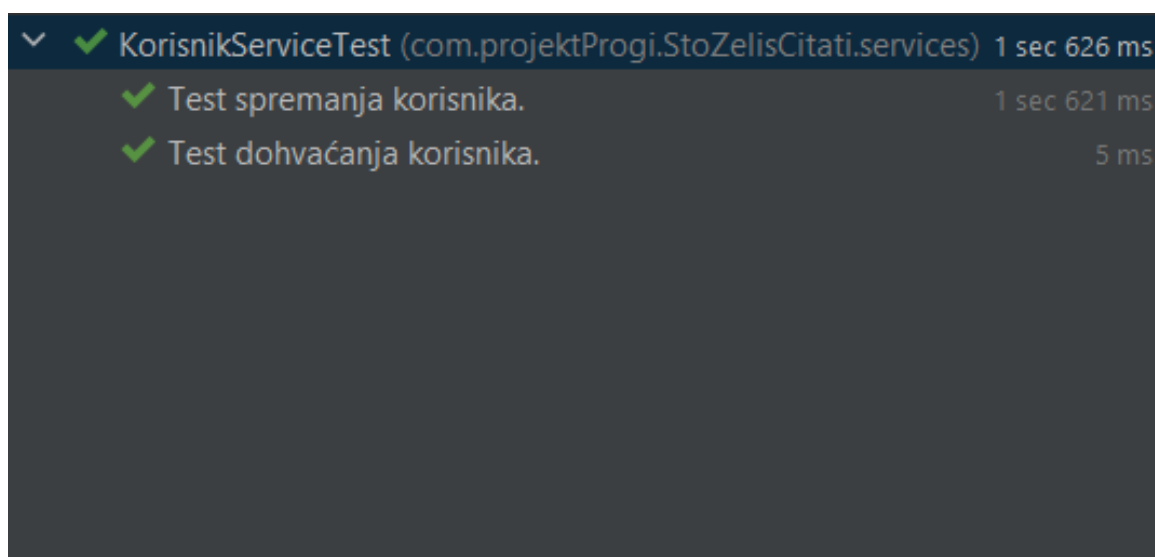
// Act & Assert
// Verify that ResponseStatusException is thrown with the correct status and message
ResponseStatusException exception = Assertions.assertThrows(ResponseStatusException.class, () -> {
    offerService.saveOffer(ponudaDto);
});

assertEquals(HttpStatus.NOT_FOUND, exception.getStatusCode());
assertEquals("Korisnik not found", exception.getReason());
}
```

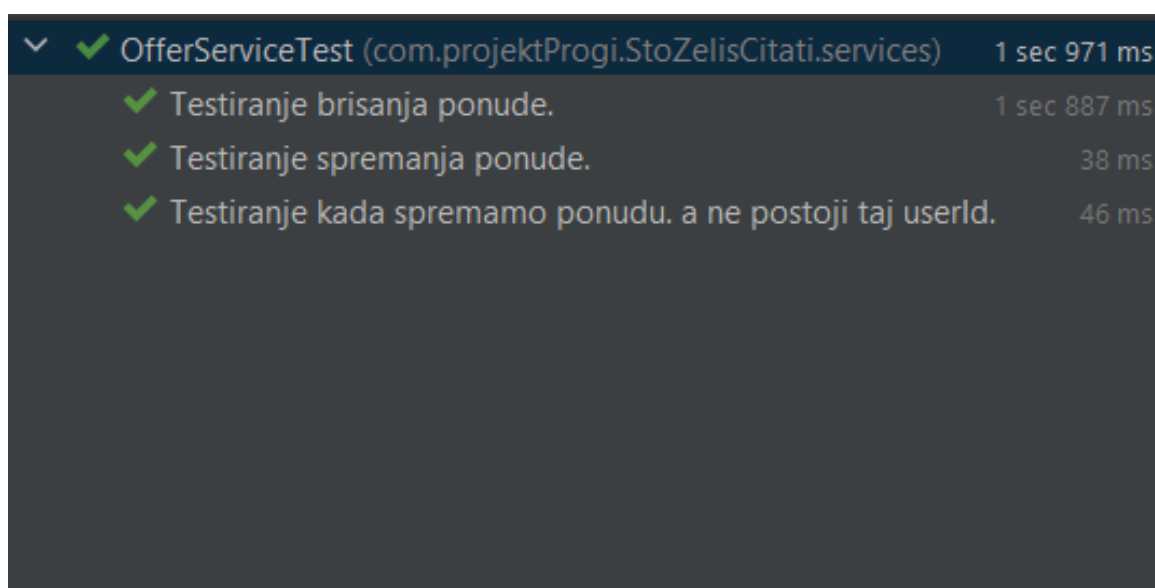
Svi testovi su se ispravno izveli s očekivanim rezultatima.



Slika 5.1: Rezultat testiranja BookService



Slika 5.2: Rezultat testiranja KorisnikService



Slika 5.3: Rezultat testiranja OfferService

## 5.2.2 Ispitivanje sustava

Testiranje cjelokupnog sustava obavljeno je korištenjem Selenium WebDriver-a unutar JUnit testova. U svakom od testova unaprijed postavljamo konfiguraciju drivera, uključujući URL aplikacije. Zatim driver automatski locira prethodno definirane elemente, a test mu prosljeđuje potrebne podatke.

Prvo smo testirali logiranje već prije registriranog korisnika.

```

9
public class SeleniumTest1 {
    public static void main(String[] args) {
        System.setProperty("webdriver.chrome.driver", "C:\\Faks\\TrecGod\\PROGI\\Projekt\\chromedriver-win64");
        WebDriver webDriver = new ChromeDriver();
        webDriver.manage().timeouts().implicitlyWait(10L, TimeUnit.SECONDS);
        boolean result = loginTest(webDriver, "user", "pass");
        System.out.println("Test pass:~" + result);

        WebDriver webDriver2 = new ChromeDriver();
        webDriver2.manage().timeouts().implicitlyWait(10L, TimeUnit.SECONDS);
        result = loginTest(webDriver2, "kriviuser", "krivasifra");
        System.out.println("Test pass:~" + result);
        webDriver.quit();
        webDriver2.quit();
    }

    public static boolean loginTest(WebDriver driver, String username, String password){

        try {
            driver.get("http://localhost:5173/login/");
            WebElement element = driver.findElement(By.name("username"));
            element.sendKeys(new CharSequence[]{username});
            element = driver.findElement(By.name("password"));
            element.sendKeys(new CharSequence[]{password});
            driver.findElement(By.name("submit")).click();
        } catch (UnhandledAlertException e){
        }

        try {
            Thread.sleep(5000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        String redirectionURL = driver.getCurrentUrl();
        System.out.println("Trenutni URL~je:~" + redirectionURL);

        return redirectionURL.contains("user");
    }
}

```

Zatim smo testirali registraciju novoga korisnika.

```

public class SeleniumTest2 {
    public static void main(String[] args) {
        System.setProperty("webdriver.chrome.driver", "C:\\Faks\\TrecGod\\PROGI\\Projekt\\chromedriver-win64");
    }
}

```

```

        WebDriver webDriver = new ChromeDriver();
        webDriver.manage().timeouts().implicitlyWait(10L, TimeUnit.SECONDS);
        boolean result = registerTest(webDriver, "user", "pass", "naziv", "adresa", "tel", "mail@gmail.com",
            "antikvarijat");
        System.out.println("Test✓pass✓" + result);
        webDriver.quit();
    }

    public static boolean registerTest(WebDriver driver, String username, String password, String naziv, String email, String tip){
        String redirectionURL = "";
        try{ driver.get("http://localhost:5173/register/");
            WebElement element = driver.findElement(By.name("username"));
            element.sendKeys(new CharSequence[]{username});
            element = driver.findElement(By.name("password"));
            element.sendKeys(new CharSequence[]{password});
            element = driver.findElement(By.name("naziv"));
            element.sendKeys(new CharSequence[]{naziv});
            element = driver.findElement(By.name("adresa"));
            element.sendKeys(new CharSequence[]{adresa});
            element = driver.findElement(By.name("telefon"));
            element.sendKeys(new CharSequence[]{telefon});
            element = driver.findElement(By.name("email"));
            element.sendKeys(new CharSequence[]{email});
            driver.findElement(By.name("tip")).click();
            driver.findElement(By.name(tip)).click();
            driver.findElement(By.name("submit")).click();
        } catch (UnhandledAlertException e){
            System.out.println(e.getAlertText());
        }

        try {
            Thread.sleep(5000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        redirectionURL = driver.getCurrentUrl();
        System.out.println("Trenutni✓URL✓je:✓" + redirectionURL);
        return redirectionURL.contains("login");
    }
}

```

Nakon toga testiramo ulogiravanje registriranog korisnika, te nakon ulogiravanja dodajemo knjigu.

```

public class SeleniumTest3 {
    public static void main(String[] args) {
        System.setProperty("webdriver.chrome.driver", "C:\\Faks\\TrecuGod\\PROGI\\Projekt\\chromedriver-win64.exe");
        WebDriver webDriver = new ChromeDriver();
        String oznaka = "hrv+dobavljiva";
        String kategorija = "domaci";
        String stanje = "izvrsno";
        webDriver.manage().timeouts().implicitlyWait(10L, TimeUnit.SECONDS);
        boolean result = addBookTest(webDriver, "user", "pass", "naziv", "autor", "izdavac", kategorija,
            "zanr", "opis", oznaka, "url", "2005", "1", stanje);
    }
}

```

```
        System.out.println("Test✓pass✓" + result);
        webDriver.quit();
    }

    private static boolean addBookTest(WebDriver driver, String username, String password, String naziv, String
        String zanr, String opis, String oznaka, String url, String godina, String brojIzdanja) {
        driver.get("http://localhost:5173/login/");
        WebElement element = driver.findElement(By.name("username"));
        element.sendKeys(new CharSequence[]{username});
        element = driver.findElement(By.name("password"));
        element.sendKeys(new CharSequence[]{password});
        driver.findElement(By.name("submit")).click();
        try {
            Thread.sleep(5000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        String redirectionURL = driver.getCurrentUrl();
        System.out.println("Trenutni✓URL✓je:✓" + redirectionURL);

        driver.findElement(By.name("addButton")).click();

        element = driver.findElement(By.name("naziv"));
        element.sendKeys(new CharSequence[]{naziv});
        element = driver.findElement(By.name("autor"));
        element.sendKeys(new CharSequence[]{autor});
        element = driver.findElement(By.name("izdavac"));
        element.sendKeys(new CharSequence[]{izdavac});
        element = driver.findElement(By.name("zanr"));
        element.sendKeys(new CharSequence[]{zanr});
        element = driver.findElement(By.name("opis"));
        element.sendKeys(new CharSequence[]{opis});
        element = driver.findElement(By.name("slikaURL"));
        element.sendKeys(new CharSequence[]{url});
        element = driver.findElement(By.name("godIzdanja"));
        element.sendKeys(new CharSequence[]{godina});
        element = driver.findElement(By.name("brojIzdanja"));
        element.sendKeys(new CharSequence[]{broj});

        driver.findElement(By.name("kategorija")).click();
        driver.findElement(By.name(kategorija)).click();

        driver.findElement(By.name("oznaka")).click();
        driver.findElement(By.name(oznaka)).click();

        driver.findElement(By.name("stanjeOcuvanosti")).click();
        driver.findElement(By.name(stanje)).click();

        driver.findElement(By.name("submit")).click();
        try {
            Thread.sleep(5000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
```



```
String redirectURL = driver.getCurrentUrl();
System.out.println("Trenutni URL je: " + redirectURL);
return redirectURL.contains("user");

}
```

Te za kraj smo testirali unošenje zahtjeva za prevođenje knjige.

```
public class SeleniumTest4 {
    public static void main(String[] args) {
        System.setProperty("webdriver.chrome.driver", "C:\\Faks\\TrecGod\\PROGI\\Projekt\\chromedriver-win64.exe");
        WebDriver webDriver = new ChromeDriver();
        webDriver.manage().timeouts().implicitlyWait(10L, TimeUnit.SECONDS);
        boolean result = testTranslationRequest(webDriver, "naziv");
        System.out.println("Test pass" + result);
        webDriver.quit();
    }

    private static boolean testTranslationRequest(WebDriver driver, String naziv){
        driver.get("http://localhost:5173/");
        driver.findElement(By.className("request_translation")).click();
        WebElement element = driver.findElement(By.name("naziv"));
        element.sendKeys(new CharSequence[]{naziv});
        driver.findElement(By.name("submit")).click();
        try {
            Thread.sleep(10000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        String redirectURL = driver.getCurrentUrl();
        System.out.println("Trenutni URL je: " + redirectURL);

        return redirectURL.equals("http://localhost:5173/");
    }
}
```

Svi testovi su se ispravno izveli s očekivanim rezultatima.

```
Trenutni URL je: http://localhost:5173/user  
Test pass: true  
Trenutni URL je: http://localhost:5173/login/  
Test pass: false  
  
Process finished with exit code 0
```

Slika 5.4: Rezultat testiranja logina

```
Trenutni URL je: http://localhost:5173/login  
Test pass true  
  
Process finished with exit code 0
```

Slika 5.5: Rezultat testiranja registriranja

```
Trenutni URL je: http://localhost:5173/user  
Trenutni URL je: http://localhost:5173/user  
Test pass true  
  
Process finished with exit code 0
```

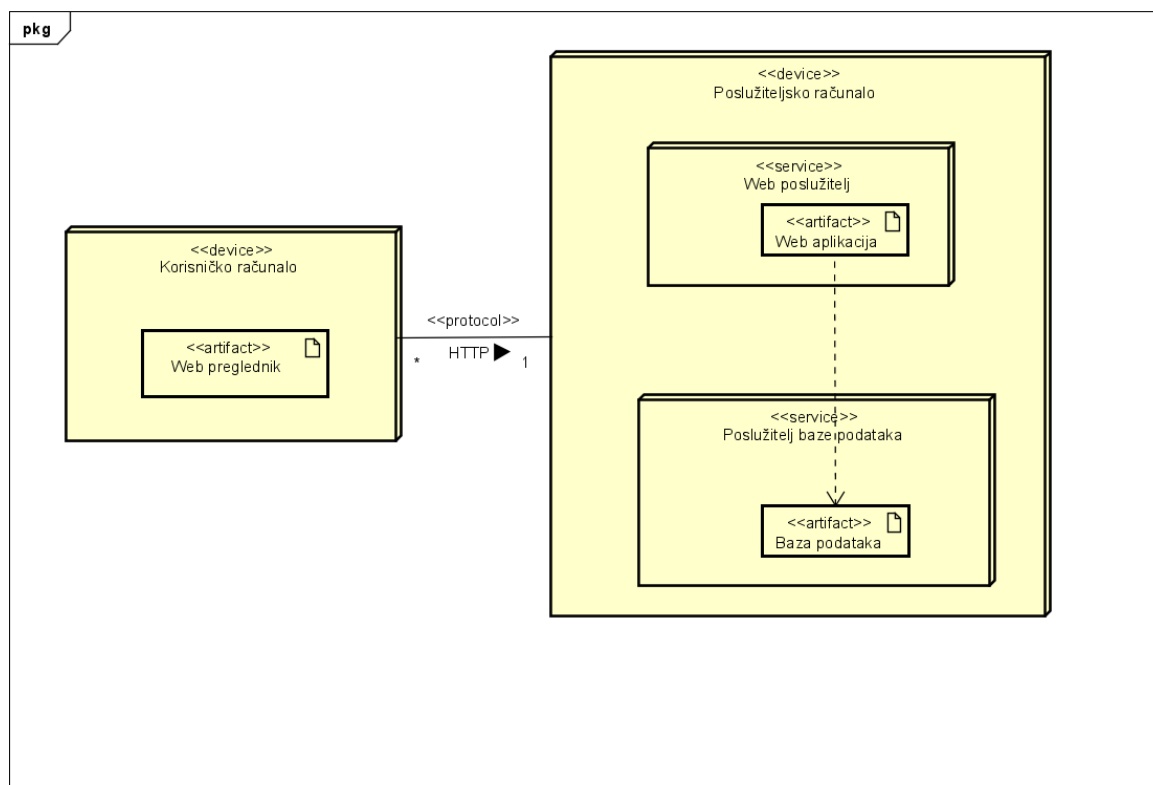
Slika 5.6: Rezultat testiranja logina uz dodavanje knjige

```
Trenutni URL je: http://localhost:5173/  
Test pass true  
  
Process finished with exit code 0
```

Slika 5.7: Rezultat testiranja dodavanje zahtjeva za prevođenje

## 5.3 Dijagram razmještaja

Dijagrami razmještaja prikazuju fizičku arhitekturu i topologiju programskog sustava te programsku potporu koja se koristi u njegovom radnom okruženju. Na korisničkom računalu se nalazi web preglednik koji služi za pristupanje web aplikaciji. Na poslužiteljskom računalu se nalaze web poslužitelj i poslužitelj baze podataka koji međusobno komuniciraju. Sustav koristi arhitekturu "klijent - poslužitelj", a komunikacija između korisnika(klijent, izdavač, administrator) i poslužitelja se odvija preko HTTP protokola.



Slika 5.8: Dijagram razmještaja

## 5.4 Upute za puštanje u pogon

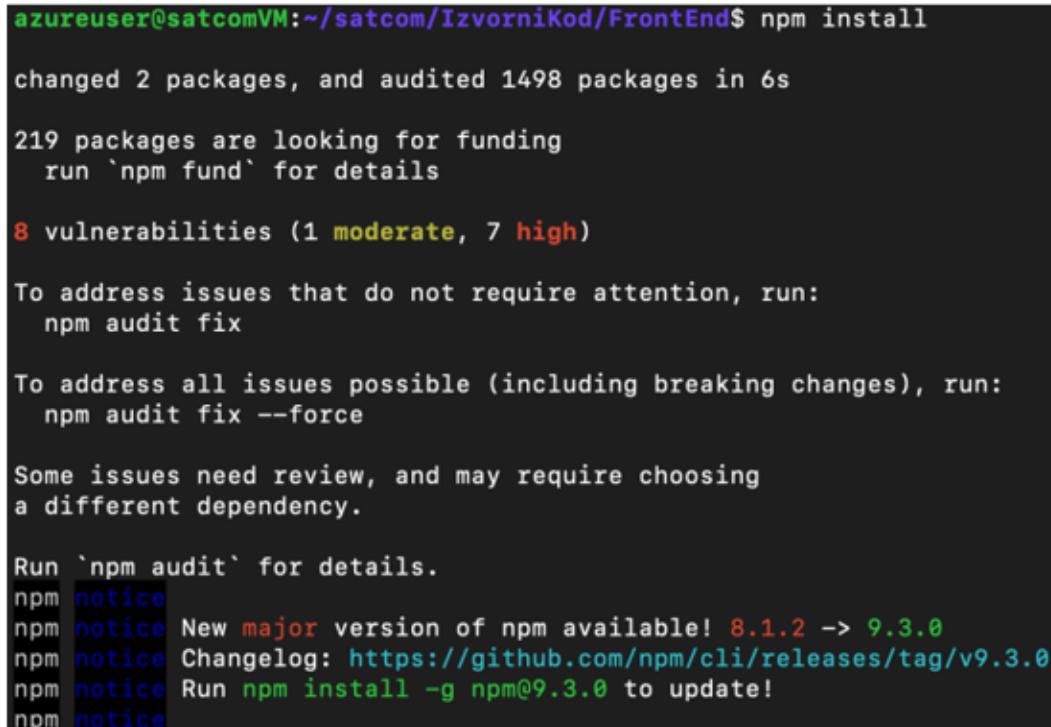
Za deploy cjelokupne aplikacije korištena je usluga Render čime je omogućeno da aplikacija bude javno dostupna. Backend aplikacije je napisan u Springboot-u te smo koristili IDE IntelliJ. Frontend je napisan s pomoću tehnologije React.js te je pisan u JavaScriptu. Baza podataka se nalazi na javnom Render serveru.

### Konfiguracija Backenda:

S pomoću IDE IntelliJ je ostvarena backend podrška, te se radi o Maven projektu. Da bi se lokalno pokrenulo na računalu potrebno je instalirati Maven dependencies (Intellij ovo radi automatski pri pokretanju projekta). Stoga je potrebno kliknuti *run/build project* te će se backend automatski povezati s bazom podataka. Nakon toga je potrebno pokrenuti frontend.

### Konfiguracija Frontenda:

Što se tiče frontenda potrebno je imati instaliran Node.js (preporučljiva najnovija verzija). Da bi se stranica pokrenula na localhostu prvo je potrebno pozicionirati se u direktorij "frontend" te utipkati u terminal *npm install*.



```
azureuser@satcomVM:~/satcom/Izvornikod/FrontEnd$ npm install

changed 2 packages, and audited 1498 packages in 6s

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

8 vulnerabilities (1 moderate, 7 high)

To address issues that do not require attention, run:
  npm audit fix

To address all issues possible (including breaking changes), run:
  npm audit fix --force

Some issues need review, and may require choosing
a different dependency.

Run `npm audit` for details.
npm notice
npm notice New major version of npm available! 8.1.2 -> 9.3.0
npm notice Changelog: https://github.com/npm/cli/releases/tag/v9.3.0
npm notice Run npm install -g npm@9.3.0 to update!
npm notice
```

Slika 5.9: Prikaz terminala nakon unosa naredbe *npm install*

Te nakon toga u terminal napisati: *npm run dev*. Ovo će pokrenuti React.js frontend dio aplikacije koristeći najnoviju verziju VITE-a na default *localhostu:5137*

## 6. Zaključak i budući rad

Naša grupa za projektni zadatak je dobila izradu i razvoj web aplikacije pod nazivom: „Što želiš čitati?“ koja omogućuje čitateljima pretragu ponuditelja knjiga na mapi, te pretragu knjiga na temelju njihove dostupnosti na hrvatskom jeziku i tržištu, također im i omogućavam zahtjev za prijevod knjige ako on već ne postoji. S druge strane ponuditeljima knjiga, izdavačima, antikvarijatima i preprodavačima omogućava ponudu knjiga većoj publici i tako poboljšanje poslovanja. Kroz četrnaest tjedana timskog rada uspješno smo ostvarili cilj i uz to smo stekli nova znanja i vještine. Projekt se razvijao u dvije faze.

Prva faza je započela okupljanjem tima, dijeljenjem ideja o razvoju projekta, određivanju ciljeva, te izražavanju pojedinačnih interesa i kompetencija na području razvoja web aplikacije, te razvoja samog projekta. Uzevši sve u obzir raspodijelili smo potrebni posao po mogućnostima i željama svakoga člana, te smo odredili rokove predaje. Podijelili smo se u dva tima, prvi od četiri člana zadužen za dokumentaciju i drugi od tri člana zaduženi za početnu implementaciju. Prva faza je trajala do kolokvija uz malo kašnjenje pri predaji.

Nakon što smo dobili ocjenu prve faze i pomoćne savjete što popraviti i kako se poboljšati u drugoj fazi, sastali smo se, raspravili smo o tome što je dobro bilo u prvoj fazi, a što je pošlo po zlu i uzrokovalo kašnjenje, te što možemo učiniti u drugoj fazi da to izbjegnemo. Zaključili smo da bi nam češći sastanci, i raspodjela zadataka na manje komponente s češćim rokovima predaje pomogla da uspješno i na vrijeme odradimo vlastite zadatke. I to bi nam omogućilo praćenje ako tko ima kakvih problema sa svojim zadatkom i brzu intervenciju, kako bi izbjegli veće zaostatke u razvoju. Budući da smo se svi planirali baviti implementacijom u drugoj fazi, tim za implementaciju iz prve faze nam je predstavio što su oni bili implemen-tirali i predložio kako da se raspodijelimo. Međusobno smo raspodijelili pravljenje potrebne dokumentacije, te smo ju napravili na početku druge faze kako bi se mogli nesmetano prebaciti na implementaciju. S time smo bili uglavnom uspješni uz nedostatak par funkcionalnosti poput provjere adresa registriranih korisnika.

To jest ne ograničavamo registrirane korisnike na područje djelovanja njihove vrste kako je specijalizirano u zadatku. Također omogućili smo da sve knjige imaju mogućnost zahtjeva za prijevod, a ne samo one na stranom jeziku. Te kod registracije ne provjeravamo je li željena e-mail adresa ili ime već zauzeto.

Sve u svemu sudjelovanje u ovom projektu je vrijedno iskustvo za svakoga člana, osim što smo naučili kako funkcionira izrada web aplikacije, kako se koristiti JavaScript bibliotekom React, okvir za razvoj Java aplikacija Spring, LaTeX, Git i GitHub te ostale tehnologije, naučili smo kako je to raditi u timu. Spoznali smo važnost dobre komunikacije i kako dobro postavljeni rokovi i dobro definirani zadatci olakšavaju izvođenje projekta. Svjesni smo da ima prostora za poboljšanje, ali zadovoljni smo postignutim uspjehom i znanjima koje smo stekli, te se veselimo budućem radu i potencijalnoj suradnji.



# Popis literature

1. Programsko inženjerstvo, FER ZEMRIS, <http://www.fer.hr/predmet/proinz>
2. I. Sommerville, "Software engineering", 8th ed, Addison Wesley, 2007.
3. T.C.Lethbridge, R.Langaniere, "Object-Oriented Software Engineering", 2nd ed. McGraw-Hill, 2005.
4. I. Marsic, Software engineering book", Department of Electrical and Computer Engineering, Rutgers University, <http://www.ece.rutgers.edu/~marsic/books/SE>
5. The Unified Modeling Language, <https://www.uml-diagrams.org/>
6. Astah Community, <http://astah.net/editions/uml-new>
7. Junit5, <https://junit.org/junit5/>

# Indeks slika i dijagrama

3.1	Dijagram obrasca uporabe, funkcionalnost korisnika i ponuditelja . . . . .	20
3.2	Dijagram obrasca uporabe, funkcionalnost izdavača i antikvarijata . . . . .	21
3.3	Dijagram obrasca uporabe, funkcionalnost preprodavača i adminis- tratora . . . . .	22
3.4	Sekvencijski dijagram za UC1 . . . . .	23
3.5	Sekvencijski dijagram za UC5 . . . . .	25
3.6	Sekvencijski dijagram za UC11 . . . . .	26
3.7	Sekvencijski dijagram za UC21 . . . . .	28
4.1	Osnova arhitekture . . . . .	32
4.2	REL dijagram baze podataka . . . . .	42
4.3	ER dijagram baze podataka . . . . .	43
4.4	Dijagram razreda . . . . .	44
4.5	Dijagram razreda . . . . .	45
4.6	Dijagram razreda . . . . .	46
4.7	Dijagram stanja . . . . .	48
4.8	Dijagram aktivnosti . . . . .	50
4.9	Dijagram komponenti . . . . .	52
5.1	Rezultat testiranja BookService . . . . .	59
5.2	Rezultat testiranja KorisnikService . . . . .	60
5.3	Rezultat testiranja OfferService . . . . .	60
5.4	Rezultat testiranja logina . . . . .	65
5.5	Rezultat testiranja registriranja . . . . .	65
5.6	Rezultat testiranja logina uz dodavanje knjige . . . . .	66
5.7	Rezultat testiranja dodavanje zahtjeva za prevođenje . . . . .	66
5.8	Dijagram razmještaja . . . . .	67
5.9	Prikaz terminala nakon unosa naredbe <i>npm install</i> . . . . .	68
6.1	Graf GitHub aktivnosti . . . . .	78
6.2	Graf GitHub aktivnosti . . . . .	79

# Dodatak: Prikaz aktivnosti grupe

## Dnevnik sastajanja

### 1. sastanak

- Datum: 21. listopada 2023.
- Prisustvovali: R. Balun, L. Blagaić, A. Gazibarić, D. Marenić, F. Šturlić, Š. Vuletić, L. Zelić
- Teme sastanka:
  - upoznavanje s zadatkom
  - odabir baznih tehnologija
  - proučavanje dokumentacije i podjela poslova

### 2. sastanak

- Datum: 5. studenoga 2023.
- Prisustvovali: R. Balun, L. Blagaić, A. Gazibarić, D. Marenić, F. Šturlić, Š. Vuletić, L. Zelić
- Teme sastanka:
  - dodatni dogovor oko raspodjele zadataka
  - proučavanje implementacije i podjela poslova
  - dogovor oko korištenih tehnologija za implementaciju

### 3. sastanak

- Datum: 12. studenoga 2023.
- Prisustvovali: R. Balun, L. Blagaić, A. Gazibarić, D. Marenić, F. Šturlić, Š. Vuletić, L. Zelić
- Teme sastanka:
  - provjera valjanosti dokumentacije
  - provjera valjanosti implementacije
  - dogovor oko puštanja aplikacije u pogon

### 4. sastanak

- Datum: 26. prosinca 2023.
- Prisustvovali: R. Balun, L. Blagaić, A. Gazibarić, D. Marenić, F. Šturlić,

Š. Vuletić, L. Zelić

- Teme sastanka:
  - dogovor oko budućeg rada
  - predložene promjene oko izrade aplikacije
  - raspodjela zadataka vezanih za dokumentaciju

#### 5. sastanak

- Datum: 30. prosinca 2023.
- Prisustvovali: R. Balun, L. Blagaić, A. Gazibarić, D. Marenić, F. Šturlić, Š. Vuletić, L. Zelić
- Teme sastanka:
  - detaljan opis implementacije i korištenih tehnologija
  - raspodjela zadataka vezanih za implementaciju

#### 6. sastanak

- Datum: 8. siječnja 2024.
- Prisustvovali: A. Gazibarić, D. Marenić, F. Šturlić, Š. Vuletić, L. Zelić
- Teme sastanka:
  - zajedničko implementiranje zadataka
  - dogovor oko sljedećih zadataka
  - raspodjela poslova

#### 7. sastanak

- Datum: 17. siječnja 2024.
- Prisustvovali: R. Balun, L. Blagaić, A. Gazibarić, F. Šturlić, Š. Vuletić, L. Zelić
- Teme sastanka:
  - dovršavanje zasebnih zadataka implementacije
  - dogovor oko puštanja aplikacije u pogon

## Tablica aktivnosti

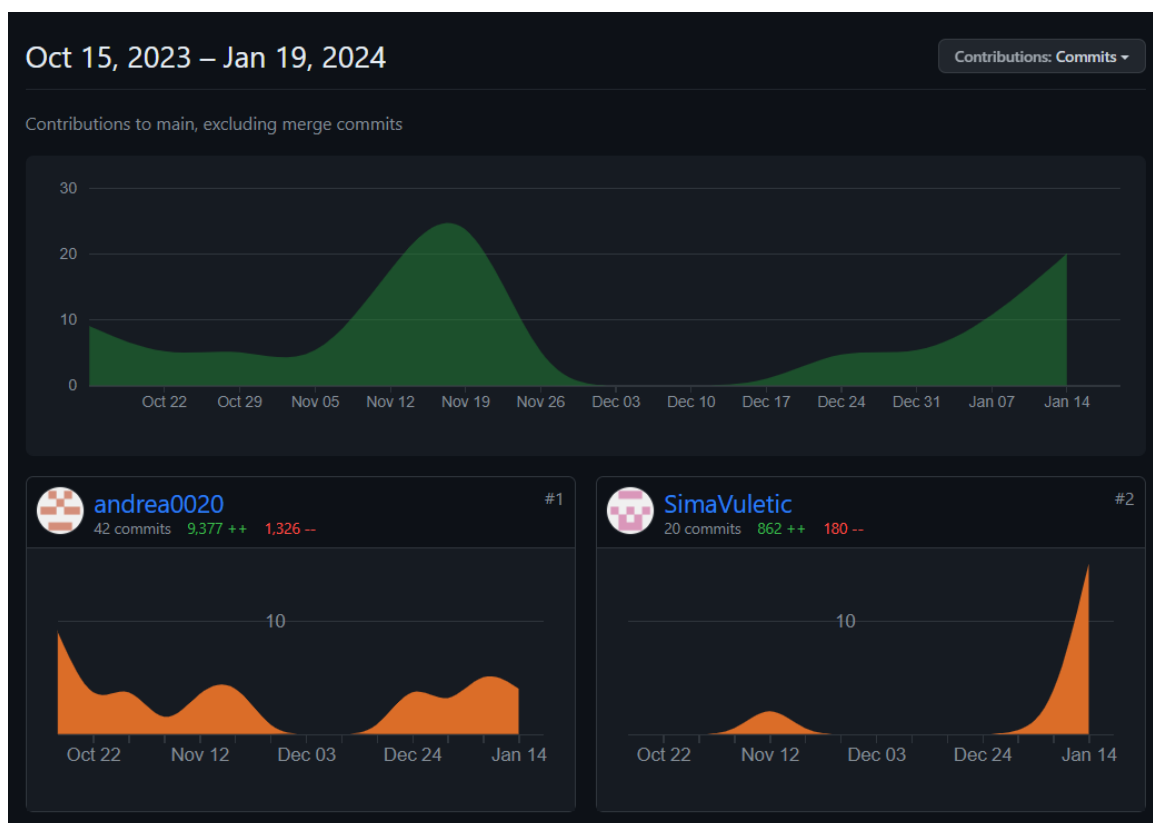
	Andrea Gazibarić	Renato Balun	Luka Blagać	Davor Marenić	Filip Šturlić	Šima Vuletić	Luka Zelić
Opis projektnog zadatka						2	
Funkcionalni zahtjevi			2				
Opis pojedinih obrazaca	3						
Dijagram obrazaca	2						
Sekvencijski dijagrami		2					
Opis ostalih zahtjeva							1
Arhitektura i dizajn sustava				2			
Baza podataka			3				
Dijagram razreda			2				
Dijagram stanja	2						
Dijagram aktivnosti	2						
Dijagram komponenti						2	
Korištene tehnologije i alati			2				
Ispitivanje programskog rješenja		3				3	
Dijagram razmještaja		2					
Upute za puštanje u pogon							1
Dnevnik sastajanja	1						
Zaključak i budući rad						2	
Popis literature						1	

Nastavljeno na idućoj stranici

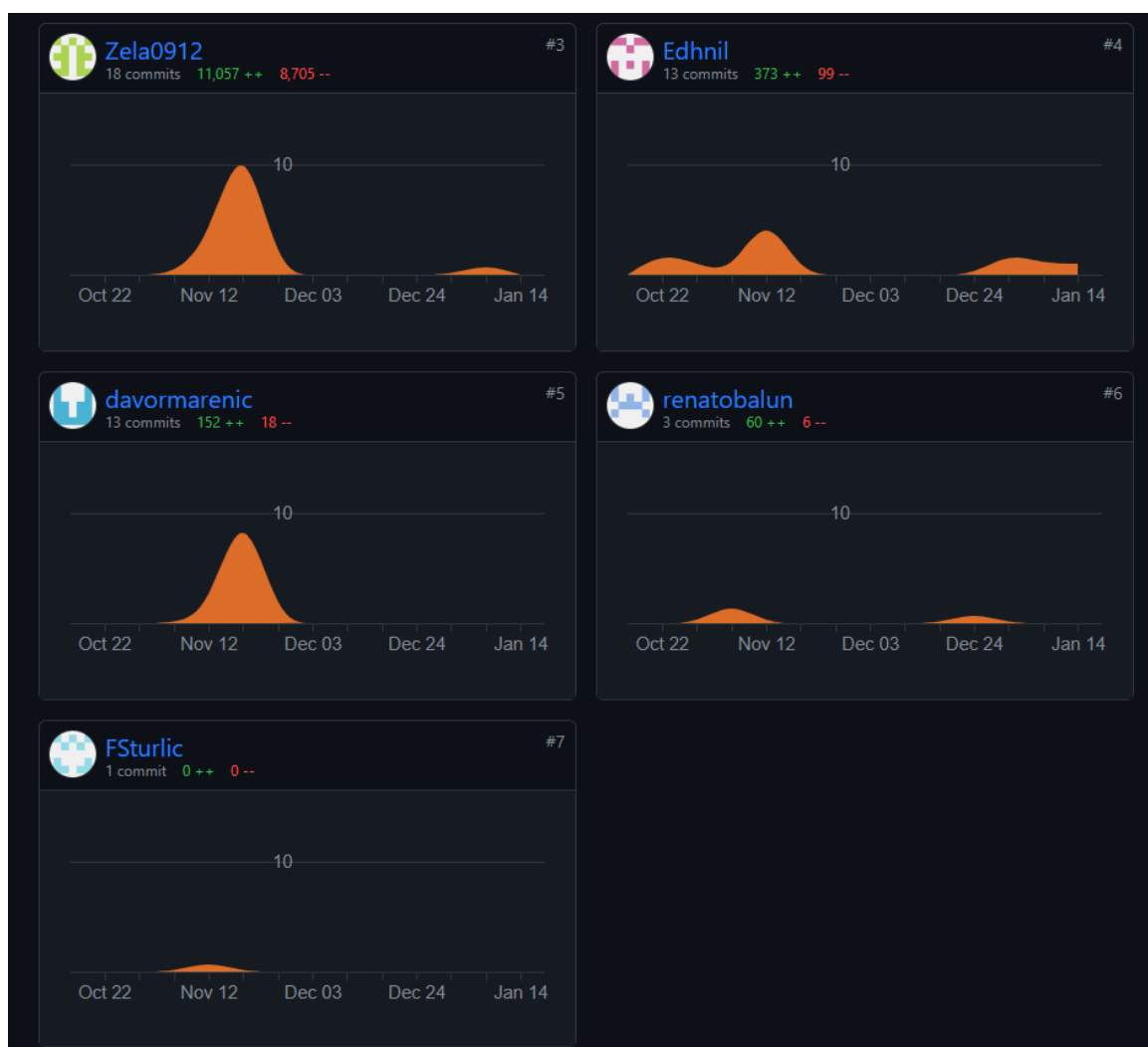
Nastavljeno od prethodne stranice

	Andrea Gazibarić	Renato Balun	Luka Blagaić	Davor Marenić	Filip Šturlić	Šima Vuletić	Luka Zelić
back end	40	5	5	2	2	2	35
front end	40	5	5	2	5	5	35
izrada baze podataka			5				
deployment	4						10
izrada dijagrama	5	5	5		5	5	
istraživanje	5	5	5	5	5	5	5

## Dijagrami pregleda promjena



Slika 6.1: Graf GitHub aktivnosti



Slika 6.2: Graf GitHub aktivnosti