

Programsko inženjerstvo

Ak. god. 2023./2024.

# FlipMemo

Dokumentacija, Rev. 2

Grupa: *Olimplusplus*

Voditelj: *Duje Štolfa*

Datum predaje: 19. 1. 2024.

Nastavnik: *Goran Rajić*

# Sadržaj

<b>1 Dnevnik promjena dokumentacije</b>	<b>3</b>
<b>2 Opis projektnog zadatka</b>	<b>5</b>
2.1 Ponavljanje s rastućim vremenskim razmakom . . . . .	5
2.2 Opis aplikacije . . . . .	7
2.2.1 Načini učenja . . . . .	7
2.3 Slična rješenja i tržište . . . . .	9
<b>3 Specifikacija programske potpore</b>	<b>10</b>
3.1 Funkcionalni zahtjevi . . . . .	10
3.1.1 Obrasci uporabe . . . . .	12
3.1.2 Sekvencijski dijagrami . . . . .	24
3.2 Ostali zahtjevi . . . . .	30
<b>4 Arhitektura i dizajn sustava</b>	<b>31</b>
4.1 Baza podataka . . . . .	32
4.1.1 Opis tablica . . . . .	33
4.1.2 Dijagram baze podataka . . . . .	36
4.2 Dijagram razreda . . . . .	38
4.2.1 Razredi na backendu . . . . .	38
4.2.2 Razredi na frontendu . . . . .	40
4.3 Dijagram stanja . . . . .	45
4.4 Dijagram aktivnosti . . . . .	47
4.5 Dijagram komponenti . . . . .	48
<b>5 Implementacija i korisničko sučelje</b>	<b>49</b>
5.1 Korištene tehnologije i alati . . . . .	49
5.1.1 Korišteni alati . . . . .	49
5.1.2 Korištene tehnologije . . . . .	49
5.1.3 Poveznice na korištene tehnologije i alate . . . . .	50
5.2 Ispitivanje programskog rješenja . . . . .	52

5.2.1	Ispitivanje komponenti . . . . .	52
5.2.2	Ispitivanje sustava . . . . .	60
5.3	Dijagram razmještaja . . . . .	64
5.4	Upute za puštanje u pogon . . . . .	65
5.4.1	Baza podataka . . . . .	65
5.4.2	Poslužiteljski dio aplikacije . . . . .	65
5.4.3	Klijentski dio aplikacije . . . . .	66
<b>6</b>	<b>Zaključak i budući rad</b>	<b>75</b>
<b>Popis literature</b>		<b>76</b>
<b>Indeks slika i dijagrama</b>		<b>78</b>
<b>Dodatak: Prikaz aktivnosti grupe</b>		<b>79</b>

# 1. Dnevnik promjena dokumentacije

Rev.	Opis promjene/dodatka	Autori	Datum
0.1	Napravljen predložak.	Duje Štolfa	25.10.2023.
0.2	Opis projektnog zadatka.	Gabrijel Čobanov Karlo Kuzle	28.10.2023.
0.3	Razrađeni funkcionalni zahtjevi, aktori i dionici. Nabrojani <i>Use Caseovi</i> .	Nina Bulić Duje Štolfa	29.10.2023.
0.4	Opisani obrasci uporabe i ostali zahtjevi.	Frane Kuzmanić Gabrijel Čobanov Nikša Brala Karlo Kuzle Nina Bulić Duje Štolfa Ivo Žilić	31.10.2023.
0.7	Dijagrami obrazaca uporabe	Frane Kuzmanić	31.11.2023.
0.6	Dodani sekvencijski dijagrami i njihovi opisi	Nina Bulić Duje Štolfa Ivo Žilić	2.11.2023.
0.7	Dijagram baze podataka	Nikša Brala	5.11.2023.
0.8	Opis baze i tablica baze podataka	Karlo Kuzle	7.11.2023.

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

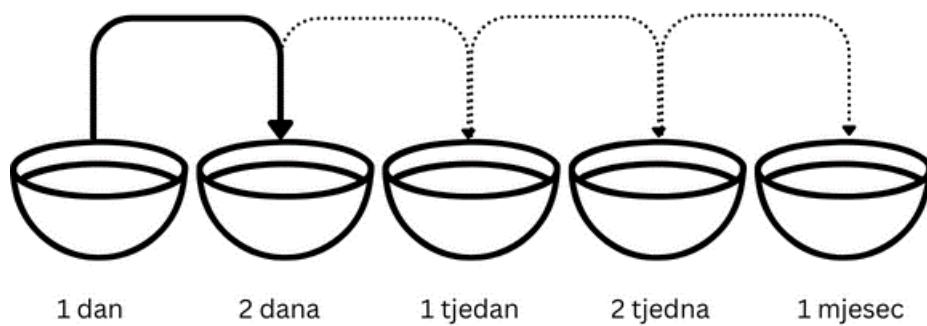
<b>Rev.</b>	<b>Opis promjene/dodataka</b>	<b>Autori</b>	<b>Datum</b>
0.9	Dodan opis arhitekture sustava	Duje Štolfa Gabrijel Čobanov Frane Kuzmanić	10.11.2023.
0.10	Dijagrami razreda	Nikša Brala	11.11.2023.
0.11	Opis dijagrama razreda	Gabrijel Čobanov Nina Bulić Ivo Žilić Duje Štolfa	13.11.2023.
<b>1.0</b>	Prva revizija dokumentacije	*	17.11.2023.
1.1	Dijagrami stanja i aktivnosti	Duje Štolfa	17.12.2023.
1.2	Dijagram komponenti	Nikša Brala	10.1.2024.
1.3	Ispitivanje komponenti	Duje Štolfa	13.1.2024.
1.4	Dijagram razmještaja	Nikša Brala	16.1.2024.
1.5	Prva inačica testiranja sustava	Karlo Kuzle	16.1.2024.
1.6	Dijagrami razreda	Nikša Brala Ivo Žilić	17.1.2024.
1.7	Druga inačica testiranja sustava	Karlo Kuzle	18.1.2024.
1.8	Zaključak i korištene tehnologije	Gabrijel Čobanov	18.1.2024.
1.9	Prezentacija projekta	Nina Bulić	18.1.2024.
1.10	Upute za puštanje u pogon	Duje Štolfa	19.1.2024.
<b>2.0</b>	Druga revizija dokumentacije	*	19.1.2024.

## 2. Opis projektnog zadatka

Cilj ovog projekta je razvoj aplikacije FlipMemo u svrhu učenja riječi na stranom jeziku. FlipMemo je namijenjen svima koji žele unaprijediti svoje znanje na interaktivan i efikasan način. Aplikacija omogućuje početnicima kao i naprednim korisnicima razvijanje svoje vještine razumijevanja, čitanja i pisanja stranog jezika. Učenje se oslanja na tehniku ponavljanja s rastućim vremenskim razmacima. Aplikaciju će moći koristiti bilo tko zainteresiran za učenje jezika, te sve što je potrebno je adresa elektroničke pošte.

### 2.1 Ponavljanje s rastućim vremenskim razmakom

SR (iz engleskog „spaced repetition“) tehnika je napravljena tako da iskoristi psihološki efekt prisjećanja. Naime, ljudi prirodno zaboravljaju informacije koje ne koriste često, jer ljudska memorija nije beskonačna, pa stvari koje nisu dio svakodnevice postaju dio zaborava. Ta činjenica još više vrijedi za nove informacije. Informacija koju ne koristimo ili ne vežemo s iskustvom se jednostavno ne smatra važnom, stoga se lako zaboravi. SR tehnika rješava ovaj problem tako što informacije koje želimo naučiti (u našem slučaju riječi jezika) pretvara u dio svakodnevice.



Slika 2.1: Ilustracija sustava za učenje s pet posuda

Kako bismo lakše razumjeli ovaj način učenja, zamislimo nekoliko posuda, kao na slici 2.1. U svakoj posudi se nalaze kartice s informacijama koje korisnik želi naučiti. Na karticama se nalazi pitanje i odgovor. Svaka posuda je označena sa svojim vremenskim intervalom. Svaka kartica na koju se da točan odgovor ide u

sljedeću posudu. Svaka kartica na koju se da netočan odgovor, ide u prvu posudu. Ako se na karticu odgovori točno, a bila je izvađena iz zadnje posude, kartica izlazi iz sistema i ta informacija se smatra naučenom. Korisnik na početku sesije učenja vadi sve kartice iz prve posude, te odgovara na pitanja. One na koje odgovori točno, stavlja u posudu od dva dana, one na koje odgovori netočno, ostavlja u posudu od jednog dana. Kad završi sa svim pitanjima, sesija je za taj dan gotova. Ako korisnik odluči učiti odmah dan nakon, može vaditi kartice iz prve posude, ali ne iz druge. Tek nakon prolaska dva dana može opet uzeti kartice iz druge posude.

Cilj je u tome da se informacije koje teže ili manje pamte, viđaju češće. Stvari koje korisnik zna ili razumije lakše, viđa manje, čime se može fokusirati na nedostatke u svom znanju.

## 2.2 Opis aplikacije

Aplikacija je kolekcija rječnika iz kojih se mogu učiti riječi odabranog jezika. Rječnik je kolekcija međusobno značenjem povezanih riječi. Svaka riječ ima svoj prijevod na hrvatski jezik, definiciju, pomoćne rečenice unutar kojih se ta riječ koristi i audiozapis izgovora riječi na svakom od njih. Korisnici imaju pristup svim rječnicima odabranog jezika za učenje, a učenje mogu provoditi na četiri načina.

Riječi i rječnici su odgovornost administratora; on ih može stvarati, brisati i raspoređivati u rječnike. Pri dodavanju riječi u sustav, administrator je dužan pobrinuti se za točnost prijevoda i definicije, kao i pomoćne rečenice i audiozapisa izgovora riječi. U tom poslu mu pomaže API.

Jednu riječ administrator može dodati u više rječnika. U tom se slučaju stanje "naučenosti" neke riječi prenosi među rječnicima. Primjerice, ako je učenik neku riječ u jednom rječniku naučio u toj mjeri da je ona u trećoj posudi, ona će u toj posudi biti i kada na nju nađe u nekom drugom rječniku.

Kako se za svaku riječ bilježi u kojoj je posudi i točno vrijeme (datum, sat i minuta) kada je stavljena u tu posudu, riječi su međusobno neovisne i mogu postati dostupne u različitim vremenima iako se nalaze u istoj posudi.

Dizajn aplikacije je fokusiran na intuitivnost i jednostavnost. Cilj je napraviti proizvod koji ne pruža prilike za pogrešno korištenje, te korisnicima pruža sigurnu i efikasnu uslugu. Korisničko sučelje se razlikuje ovisno o vrsti prijavljenog korisnika. Administratorsko sučelje pruža isključivo funkcionalnosti izmjene i dorađivanja riječi i rječnika, dok učeničko sučelje ima samo pristup opcijama vezanim za učenje. Svi vizualni elementi se oslanjaju na minimalnost.

### 2.2.1 Načini učenja

U aplikaciji su predviđena četiri načina učenja, kako bi savladavanje gradiva išlo što lakše i brže:

**Prijevod strane riječi:** Korisniku je prikazan detaljan opis riječi stranog jezika (rijec, definicija i sve fraze). Ponođeno mu je nekoliko riječi na hrvatskom jeziku od kojih on mora odabrati onu koja je najbolji prijevod strane riječi. Riječi koje se nude kao odgovori biraju se iz trenutno aktivnog rječnika, tj. rječnika iz kojeg se uči.

**Prijevod na strani jezik:** Korisniku je prikazana riječ na hrvatskom jeziku, te mu

je ponuđeno nekoliko riječi na stranom jeziku. Korisnik od njih mora odabrat onu koja je točan prijevod riječi na hrvatskom jeziku.

**Slušanje i pisanje:** Korisnik sluša audiozapis izgovora trenutne riječi na odabranom jeziku učenja. Korisnik treba napisati riječ koju je čuo. Time se provjera razumije li korisnik razlike u izgovoru riječi kao i njihovo pisanje.

**Snimanje izgovora riječi:** Korisniku je prikazan detaljan opis riječi (rijec, definicija i sve fraze) na jeziku koji uči i opciju za snimanje glasa. Potrebno je točno izgovoriti riječ prikazanu na ekranu. Izgovor se ocjenjuje brojem od jedan do deset, čime korisnik dobiva povratnu informaciju o svom izgovoru.

Nakon svakog odgovorenog pitanja, korisnik dobiva povratnu informaciju o točnosti odgovora.

## 2.3 Slična rješenja i tržište

Kao što je već spomenuto, aplikaciju može koristiti bilo tko ima email adresu. Može biti korisna učenicima u školama, kao sredstvo za samostalni rad, ali mogu je i sami nastavnici preporučivati ili čak integrirati u nastavu. Aplikacija također može biti korisna predanim turistima koji se spremaju za putovanje u državu u kojoj ne poznaju jezik, ili čak poslovnim ljudima koji imaju čest dodir sa stranim kulturama u svakodnevici.

S dugo razvijanim i vrlo složenim algoritmom za ponavljanje s odmakom, jedna od najznačajnijih aplikacija u *spaced repetition* svijetu zasigurno je Anki. U Ankiju korisnici mogu samostalno raditi kolekcije kartica za učenje (*flashcards*) i dijeliti ih međusobno. S jedne strane, to je dobra stvar, jer se može učiti bilo koja tema koja se poželi, no s druge strane, ako korisnik nije voljan ili nema vremena praviti kartice za učenje, a netko nije napravio kvalitetnu kolekciju i ponudio ostalim korisnicima, taj korisnik ne može učiti. Specijalizacija naše aplikacije na jezike osigurava kvalitetu učenja i kvalitetu kartica.

The screenshot shows the main interface of the Anki application. At the top, there is a navigation bar with tabs: Decks, Add, Browse, Stats, and Sync. Below the navigation bar is a table titled "Deck" with columns for "Deck", "New", "Learn", and "Due". The table lists several decks:

Deck	New	Learn	Due
+ Geography	20	0	124
- Languages	6	19	73
English	3	3	35
German	3	6	18
+ Spanish	0	10	20
My Deck	3	5	42
+ Other	0	11	100

At the bottom of the interface, a message states: "Studied 0 cards in 0 seconds today (0s/card)".

Slika 2.2: Špilovi u Ankiju pandan su našim rječnicima

# 3. Specifikacija programske potpore

## 3.1 Funkcionalni zahtjevi

Dionici:

1. Vlasnik (naručitelj)
2. Učenici
3. Administratori
4. Razvojni tim

Aktori i njihovi funkcionalni zahtjevi:

1. Administrator (inicijator) može:

- (a) pregledati i odabrati jezik te unutar tog jezika
  - i. stvoriti, urediti, brisati i pregledati riječi (riječ, opis, fraze, prijevod)
  - ii. stvoriti, urediti, brisati i pregledati rječnike (naziv, broj riječi)
  - iii. dodati ili ukloniti riječ/i iz jednog ili više rječnika
  - iv. stvoriti administratora
  - v. brisati administratora

2. Učenik (inicijator) može:

- (a) pregledati i odabrati dostupne jezike
- (b) pregledati i odabrati dostupne rječnike odabranog jezika
- (c) započeti učenje odabirom jednog od četiri načina učenja
- (d) izbrisati svoj korisnički račun

3. Baza podataka (sudionik) može:

- (a) pohraniti sve podatke o učenicima i njihovim (ne)naučenim riječima
- (b) pohraniti sve podatke o učenicima o administratorima
- (c) pohraniti sve podatke o riječima i rječnicima

4. API za rječnike (sudionik) može:

(a) dohvatiti dodatne podatke o riječima (opis, fraza)

5. Neregistrirani korisnik (inicijator) može:

(a) registrirati se:

- i. potvrditi registraciju prijavom s inicijalnom lozinkom
- ii. promijeniti inicijalnu lozinku

6. Servis za ocjenu izgovora (sudionik) može:

(a) ocijeniti korisnikov izgovor neke riječi ili izraza

7. Davatelj e-pošte (sudionik) može:

(a) pruža uslugu stvaranja i korištenja e-mail računa

### 3.1.1 Obrasci uporabe

#### Opis obrazaca uporabe

##### UC1 - Registracija

- **Glavni sudionik:** Neregistrirani korisnik
- **Cilj:** Stvaranje učeničkog računa
- **Sudionici:** Baza podataka, davatelj e-pošte
- **Preduvjet:** Učenik nije registriran niti prijavljen u sustav
- **Opis osnovnog tijeka:**
  1. Korisnik odabire opciju "Registracija"
  2. Korisnik unosi ime, prezime, e-mail i potvrđuje prijavu
  3. Korisnik na svoj e-mail dobiva privremenu generiranu lozinku
  4. Korisnik se prijavljuje u sustav s generiranom lozinkom (UC3 Prijava u sustav)
  5. Sustav učeniku prikazuje obrazac za promjenu inicijalne lozinke (nastavak UC2 Promjena lozinke)
- **Opis mogućih odstupanja:**
  - 2.a Korisnik se pokušava registrirati e-mail adresom na koju je već prijavljen račun
    1. Sustav upozorava korisnika i onemogućuje mu registraciju

##### UC2 - Promjena lozinke

- **Glavni sudionik:** Učenik
- **Cilj:** promjena lozinke
- **Sudionici:** Baza podataka
- **Preduvjet:** UC1 Registracija
- **Opis osnovnog tijeka:**
  1. Učenik u postavkama svog računa bira opciju "Promjena lozinke"
  2. Učeniku se prikazuje dijaloški okvir u kojem treba unijeti i potvrditi svoju novu lozinku
  3. Sustav preusmjerava korisnika na zaslon za odabir jezika (UC7 Pregledavanje i odabir jezika)
- **Opis mogućih odstupanja:**
  - 1.a Učenik gasi aplikaciju
    1. Ne dolazi do promjene lozinke, učenik mora ponoviti proces

- 2.a Polja "lozinka" i "potvrdi lozinku" se razlikuju
  1. Sustav upozorava učenika i traži ispravak unosa

### UC3 - Prijava u sustav

- **Glavni sudionik:** Učenik ili administrator
- **Cilj:** Dobivanje pristupa učeničkom ili administratorskom sučelju
- **Sudionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
  1. Korisnik aplikacije unosi svoj e-mail i lozinku
  2. Sustav provjerava postoji li račun s istim podacima
  3. Ako su podaci ispravni, sustav preusmjerava korisnika na zaslon za odabir jezika (nastavak UC7 Pregledavanje i odabir jezika)
- **Opis mogućih odstupanja:**
  - 2.a Neispravan e-mail ili lozinka
    1. Sustav obavještava korisnika o grešci

### UC4 - Brisanje korisničkog računa

- **Glavni sudionik:** Učenik
- **Cilj:** Iz sustavske baze podataka obrisati sve zapise o učeničkom računu i naučenim riječima
- **Sudionici:** Baza podataka
- **Preduvjet:** UC3 Prijava u sustav
- **Opis osnovnog tijeka:**
  1. Korisnik odabire opciju "obriši moj račun"
  2. Korisniku se prikazuje dijaloški okvir brisanja računa gdje mu se opisuje koji podatci će biti izbrisani i obavještava ga se da sve riječi koje je naučio će također biti obrisane te mu se prikazuju gumbovi "Obriši" i "Odustani"
  3. Korisnik potvrđuje brisanje klikom na gumb "Obriši"
- **Opis mogućih odstupanja:**
  - 3.a Korisnik odustaje od brisanja
    1. Korisnički račun ostaje neizbrisani

### UC5 - Dodavanje administratora

- **Glavni sudionik:** Administrator

- **Cilj:** Stvaranje administratora
- **Sudionici:** Baza podataka
- **Preduvjet:** Prijava administratora u sustav (UC3 Prijava u sustav)
- **Opis osnovnog tijeka:**
  1. Administrator odabire opciju "stvorи administratora "
  2. Administrator dolazi na stranicu u kojoj mora upisati ime, prezime, email adresu i lozinku novog administratora
  3. Administrator potvrđuje stvaranje novog administratora
  4. Podaci se spremaju u bazu podataka
- **Opis mogućih odstupanja:**
  - 2.a Admin upisuje email adresu već postojićeg admina
    1. Sustav ga upozorava da već postoji admin s upisanom email adresom

### UC6 - Brisanje administratora

- **Glavni sudionik:** Administrator
- **Cilj:** Uklanjanje administratora iz sustava
- **Sudionici:** Baza podataka
- **Preduvjet:** Prijavljen admin u sustav (UC3 Prijava u sustav), postojanje admina kojeg se želi izbrisati
- **Opis osnovnog tijeka:**
  1. Administrator odabire opciju "pregled administratora" i prikazuje mu se popis admina
  2. Administrator za specifičnog administratora odabire opciju "Obriši"
  3. Administratoru se prikazuje dijaloški okvir s porukom upozorenja o posljedicama brisanja te gumbovima "Obriši" i "Odustani"
  4. Administrator potvrđuje brisanje klikom na gumb "Obriši"
- **Opis mogućih odstupanja:**
  - 4.a Admin odustaje od brisanja
    1. Račun se ne briše

### UC7 - Pregledavanje i odabir jezika

- **Glavni sudionik:** Učenik ili administrator
- **Cilj:** Odabrati željeni jezik iz popisa dostupnih jezika
- **Sudionici:** Baza podataka
- **Preduvjet:** UC3 Prijava u sustav

- **Opis osnovnog tijeka:**

1. Sustav korisniku prikazuje popis dostupnih jezika
2. Korisnik odabire jezik
3. Sustav preusmjerava:
  - učenika na zaslon za odabir rječnika (UC8 Pregledavanje i odabir rječnika)
  - administratora na zaslon za upravljanje riječima

### UC8 - Pregledavanje i odabir rječnika

- **Glavni sudsionik:** Učenik ili administrator

- **Cilj:** Odabrati rječnik iz popisa rječnika nekog jezika

- **Sudsionici:** Baza podataka

- **Preduvjet:** UC7 Pregledavanje i odabir jezika

- **Opis osnovnog tijeka:**

1. Korisniku su prikazani svi rječnici u odabranom jeziku
2. Korisnik odabire jedan od njih
3. Sustav preusmjerava:
  - učenika na zaslon za odabir načina učenja (UC13 Odabir načina učenja)
  - administratora na obrazac za uređivanje rječnika (UC11 Promjena naziva rječnika, UC10 Promjena sadržaja rječnika)

- **Opis mogućih odstupanja:**

- 1.a Ne postoji nijedan rječnik u odabranom jeziku

1. Korisniku je prikazana informativna poruka

2. Preskaču se koraci 2. i 3.

- 2.a Administrator odabire opciju "Obriši rječnik" pokraj jednog od rječnika

1. Administrator nastavlja s brisanjem rječnika (UC12 Brisanje rječnika)

2. Preskače se korak 3.

- 2.b Administrator odabire opciju "Dodaj novi rječnik"

1. Administrator nastavlja sa stvaranjem rječnika (UC9 Stvaranje rječnika)

2. Preskače se korak 3.

### UC9 - Stvaranje rječnika

- **Glavni sudsionik:** Administrator

- **Cilj:** Dodati novi rječnik u odabrani jezik

- **Sudsionici:** Baza podataka

- **Preduvjet:** UC8 Pregledavanje i odabir rječnika
- **Opis osnovnog tijeka:**
  1. Administrator je prikazan obrazac za stvaranje rječnika
  2. Administrator upisuje naziv rječnika i potvrđuje unos
  3. Rječnik se dodaje u bazu podataka
- **Opis mogućih odstupanja:**
  - 2.a Administrator je upisao prazan naziv rječnika
    1. Sustav javlja administratoru da naziv rječnika ne smije biti prazan

### UC10 - Promjena sadržaja rječnika

- **Glavni sudionik:** Administrator
- **Cilj:** Izmjena popisa riječi u rječniku
- **Sudionici:** Baza podataka
- **Preduvjet:** UC8 Pregledavanje i odabir rječnika
- **Opis osnovnog tijeka:**
  1. U obrascu je prikazan popis svih riječi u rječniku
  2. Administrator odabire opciju "Dodaj riječ"
  3. Administratoru je prikazan okvir s popisom svih riječi trenutnog jezika koje nisu u rječniku
  4. Administrator odabire jednu ili više riječi
  5. Administrator potvrđuje odabir
  6. Odabrane riječi dodaju se u rječnik
- **Opis mogućih odstupanja:**
  - 2.a Admin pokraj već dodane riječi odabire opciju "Ukloni riječ"
    1. Prikazuje se okvir za potvrdu brisanja
    2. Koraci 3.-6. se preskaču
  - 3.a Sve su riječi odabranog jezika već u rječniku
    1. Administratoru je prikazana informativna poruka umjesto popisa
    2. Administrator odustaje od mijenjanja rječnika
    3. Koraci 4.-6. se preskaču
  - 5.a Administrator odustaje od promjena
    1. Promjene se ne spremaju u bazu

### UC11 - Promjena naziva rječnika

- **Glavni sudionik:** Administrator
- **Cilj:** Promijeniti naziv odabranog rječnika

- **Sudionici:** Baza podataka
- **Preduvjet:** UC8 Pregledavanje i odabir rječnika
- **Opis osnovnog tijeka:**
  1. Administrator u obrascu odabire opciju "Promijeni naziv"
  2. Administrator upisuje novi naziv i potvrđuje promjene
  3. Sustav sprema promjene u bazu
- **Opis mogućih odstupanja:**
  - 2.a Admin pokušava spremiti prazan naziv rječnika
    1. Sustav javlja administratoru da naziv rječnika ne smije biti prazan
  - 2.b Administrator odustaje od promjena
    1. Promjene se ne spremaju u bazu

### UC12 - Brisanje rječnika

- **Glavni sudionik:** Administrator
- **Cilj:** Brisanje rječnika iz sustava
- **Sudionici:** Baza podataka
- **Preduvjet:** UC8 Pregledavanje i odabir rječnika
- **Opis osnovnog tijeka:**
  1. Administrator odabire opciju brisanje rječnika
  2. Administratoru se prikazuje dijaloški okvir s porukom upozorenja o posljedicama brisanja te gumbovima "Obriši" i "Odustani"
  3. Administrator potvrđuje brisanje klikom na gumb "Obriši"
  4. Sustav iz baze briše odabrani rječnik
- **Opis mogućih odstupanja:**
  - 3.a Admin odustaje od brisanja
    1. Rječnik ostaje neizbrisani

### UC13 - Odabir načina učenja

- **Glavni sudionik:** Učenik
- **Cilj:** Započeti dnevnu sesiju učenja u odabranom načinu
- **Sudionici:** -
- **Preduvjet:** Postojanje korisničkog računa, prijava u sustav s korisničkim računom, odabir jezika i rječnika (UC8 pregledavanje i odabir rječnika)
- **Opis osnovnog tijeka:**
  1. Učenik bira jedan od četiri gumba koji predstavljaju načine rada

2. Nakon odabira, učenika se preusmjerava na stranicu tog specifičnog načina rada i započinje učenje

#### UC14 - Prijevod strane riječi

- **Glavni sudionik:** Učenik
- **Cilj:** Učenik je naučio prijevod strane riječi ili izraza na hrvatski
- **Sudionici:** Baza podataka
- **Preduvjet:** Učenik je prijavljen, učenik je prethodno odabrao koji će jezik učiti, iz kojeg će rječnika učiti te je odabrao način učenja Prijevod strane riječi (UC13 Odabir načina učenja)
- **Opis osnovnog tijeka:**
  1. Korisniku je prikazana riječ na stranom jeziku s nekoliko ponuđenih prijevoda na hrvatski
  2. Korisnik odabire jedan od ponuđenih prijevoda
  3. Sustav korisnika obavještava je li odabrao ispravan prijevod
  4. Sustav korisniku nudi nastavak učenja ili povratak na odabir načina učenja

#### UC15 - Prijevod na strani jezik

- **Glavni sudionik:** Učenik
- **Cilj:** Učenik je naučio prijevod strane riječi ili izraza na hrvatski
- **Sudionici:** Baza podataka
- **Preduvjet:** Učenik je prijavljen, učenik je prethodno odabrao koji će jezik učiti, iz kojeg će rječnika učiti te je odabrao način učenja Prijevod na strani jezik (UC13 Odabir načina učenja)
- **Opis osnovnog tijeka:**
  1. Korisniku je prikazana riječ na hrvatskom s nekoliko ponuđenih prijevoda na strani jezik
  2. Korisnik odabire jedan od ponuđenih prijevoda
  3. Sustav korisnika obavještava je li odabrao ispravan prijevod
  4. Sustav korisniku nudi nastavak učenja ili povratak na odabir načina učenja

#### UC16 - Prijevod zvučnog zapisa

- **Glavni sudionik:** Učenik
- **Cilj:** Učenik može točno napisati riječ na stranom jeziku

- **Sudionici:** Baza podataka
- **Preduvjet:** Učenik je prijavljen, učenik je prethodno odabrao koji će jezik učiti, iz kojeg će rječnika učiti te je odabrao način učenja Prijevod zvučnog zapisa (UC13 Odabir načina učenja)
- **Opis osnovnog tijeka:**
  1. Korisniku klikom na simbol zvučnika čuje riječ na stranom jeziku
  2. Korisnik upisuje riječ na stranom jeziku
  3. Sustav korisnika obavještava je li upisana riječ ispravno napisana
  4. Sustav korisniku nudi nastavak učenja ili povratak na odabir načina učenja

### UC17 - Snimanje izgovora riječi

- **Glavni sudionik:** Učenik
- **Cilj:** Učenik pravilno izgovara riječi na stranom jeziku
- **Sudionici:** Baza podataka, Servis za ocjenu izgovora
- **Preduvjet:** Učenik je prijavljen, učenik je prethodno odabrao koji će jezik učiti, iz kojeg će rječnika učiti te je odabrao način učenja Snimanje izgovora (UC13 Odabir načina učenja)
- **Opis osnovnog tijeka:**
  1. Korisniku je ponuđena riječ na stranom jeziku
  2. Korisnik pritiskom na simbol mikrofona započinje snimanje zvuka
  3. Korisnik izgovara ponuđenu riječ na stranom jeziku
  4. Korisnik pritiskom na simbol mikrofona završava snimanje zvuka
  5. Sustav pomoću servisa za ocjenu izgovora korisnika obavještava o kvaliteti njegova izgovora
  6. Sustav korisniku nudi nastavak učenja ili povratak na odabir načina učenja

### UC18 - Stvaranje riječi

- **Glavni sudionik:** Administrator
- **Cilj:** Mogućnost pregledavanja svih riječi za zadani jezik
- **Sudionici:** Baza podataka, API za rječnike
- **Preduvjet:** Prijavljanje u administratorski račun, odabran jezik, administrator se nalazi na stranici za pregled riječi
- **Opis osnovnog tijeka:**
  1. Administrator odabire opciju dodavanja riječi u sustav

2. Preusmjerava se na zaslon s obrascem za stvaranje riječi
3. Administrator unosi riječ
4. Administrator unosi prijevod riječi, pri čemu pomaže API
5. Administrator unosi pomoćne fraze
6. Administrator unosi zvučnu datoteku izgovora riječi na odabranom jeziku

### UC19 - Pregledavanje riječi

- **Glavni sudionik:** Administrator
- **Cilj:** Mogućnost pregledavanja svih riječi za zadani jezik
- **Sudionici:** Baza podataka
- **Preduvjet:** Prijavljanje u administratorski račun, postojanje riječi u sustavu, odabran jezik
- **Opis osnovnog tijeka:**
  1. Administrator odabire opciju pregleda riječi
  2. Na ekranu mu se prikazuje lista svih riječi u odabranom jeziku koje se nalaze u sustavu (bazi podataka)

### UC20 - Brisanje riječi

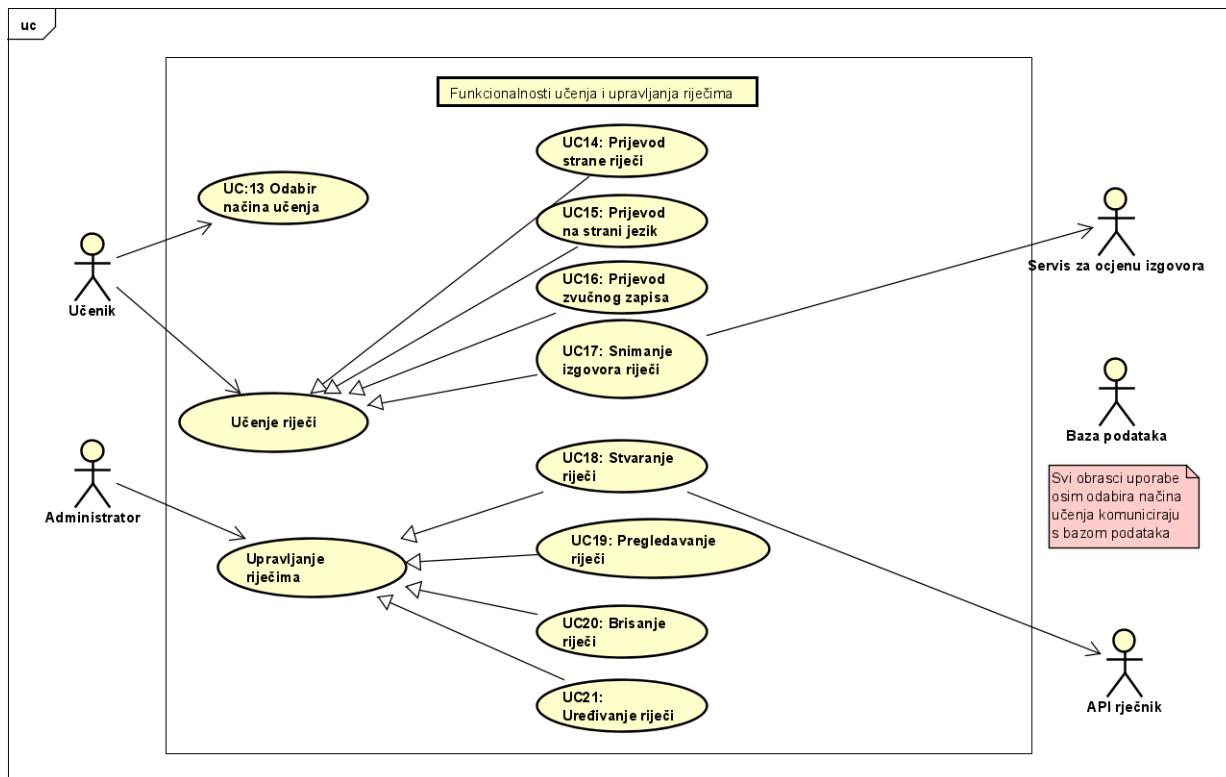
- **Glavni sudionik:** Administrator
- **Cilj:** Brisanje riječi iz odabranog jezika
- **Sudionici:** Baza podataka
- **Preduvjet:** Prijavljanje u administratorski račun, postojanje riječi u sustavu, odabran jezik
- **Opis osnovnog tijeka:**
  1. Administrator odabire opciju pregleda riječi
  2. Na ekranu mu se prikazuje lista svih riječi u odabranom jeziku koje se nalaze u sustavu (bazi podataka)
  3. Pored svake riječi se nalazi gumb za brisanje riječi
  4. Klikom na gumb adminu se prikazuje dijaloški okvir s porukom upozorenja o posljedicama brisanja te gumbovima "Obriši" i "Odustani"
  5. Administrator potvrđuje brisanje klikom na gumb "Obriši"
- **Opis mogućih odstupanja:**
  - 5.a Admin odustaje od brisanja
    1. Riječ ostaje neizbrisana

### UC21 - Uređivanje riječi

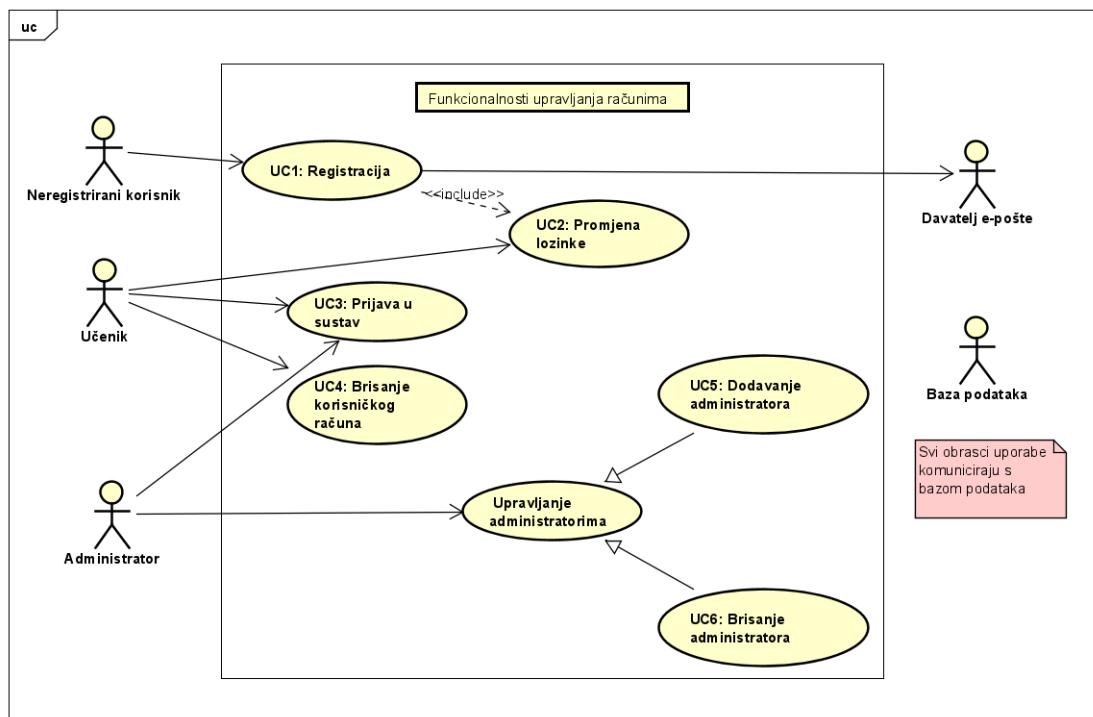
---

- **Glavni sudionik:** Administrator
- **Cilj:** Mogućnost pregledavanja svih riječi za zadani jezik
- **Sudionici:** Baza podataka
- **Preduvjet:** Prijavljanje u administratorski račun, postojanje riječi u sustavu, odabran jezik
- **Opis osnovnog tijeka:**
  1. Administrator odabire opciju pregleda riječi
  2. Na ekranu mu se prikazuje lista svih riječi u odabranom jeziku koje se nalaze u sustavu (bazi podataka)
  3. Administrator klikom na određenu riječ ide na stranicu za uređivanje
  4. Ima opciju promijeniti riječ, njen prijevod na strani jezik ili pomoćne fraze
  5. Nakon napravljenih željenih promjena, administrator potvrdi svoj rad i vraća se na pregled riječi
- **Opis mogućih odstupanja:**
  - 5.a Admin odustaje od uređivanja
    1. Riječ ostaje nepromijenjena

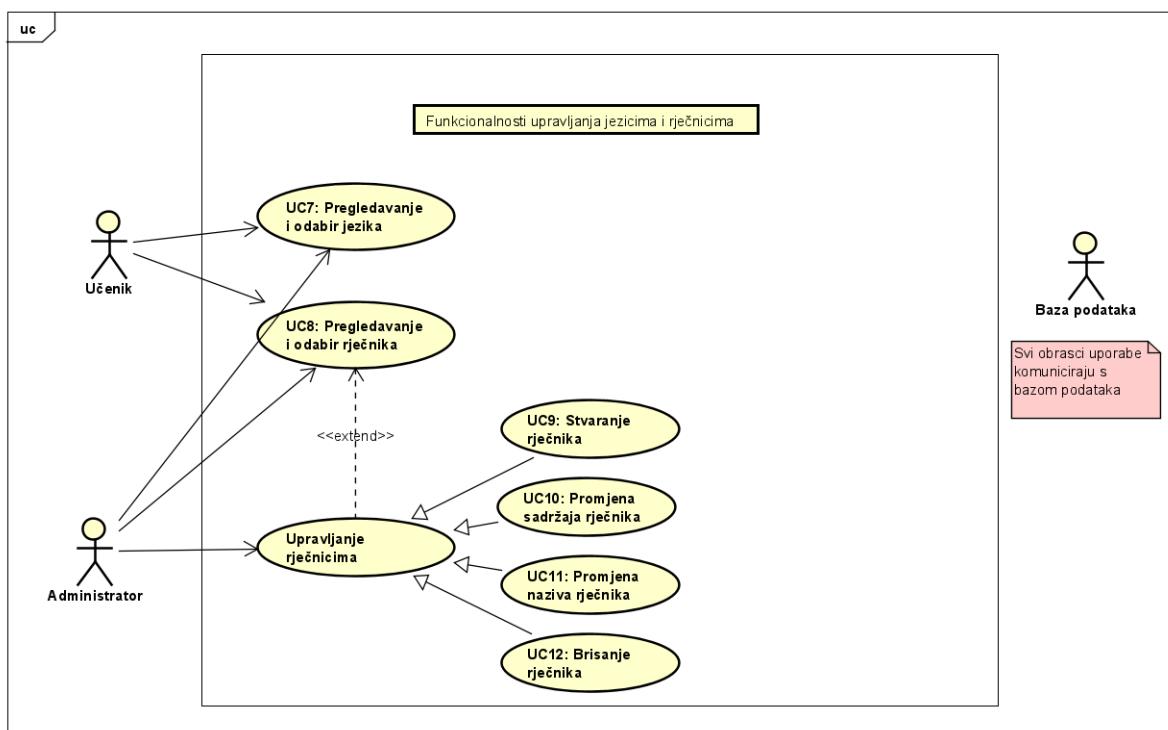
## Dijagrami obrazaca uporabe



Slika 3.1: Funkcionalnosti učenja i upravljanja riječima



Slika 3.2: Funkcionalnosti upravljanja računima

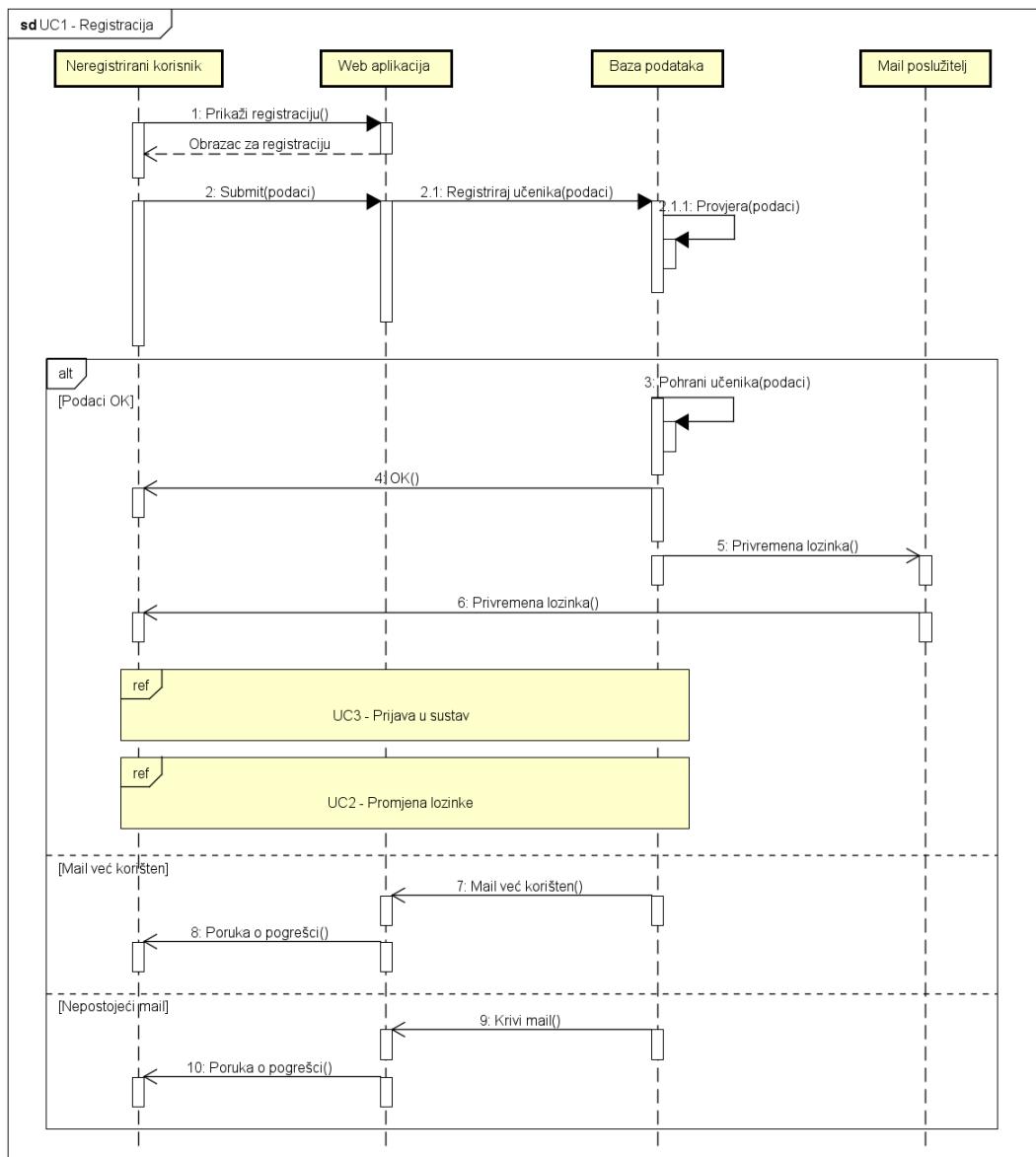


Slika 3.3: Funkcionalnosti upravljanja jezicima i rječnicima

### 3.1.2 Sekvencijski dijagrami

#### Obrazac uporabe 1: Registracija

Neregistrirani korisnik započinje registraciju odabirom opcije "Registracija" na korisničkom sučelju. Nakon što je opcija odabrana, poslužitelj šalje zahtjev korisniku da unese svoje osobne podatke, uključujući ime, prezime i e-mail adresu. Korisnik unosi tražene podatke i nakon toga potvrđuje svoju registraciju putem korisničkog sučelja. Poslužitelj započinje proces provjere unesenih podataka. Prvo, provjerava dostupnost e-mail adrese u bazi podataka kako bi utvrdio postoje li odstupanja. U slučaju da e-mail adresa već postoji u bazi podataka ili nije ispravna, poslužitelj obavještava korisnika o nemogućnosti registracije i sprječava daljnji napredak u procesu. Ukoliko su svi uneseni podaci ispravni, poslužitelj generira privremenu lozinku za korisnika, koja se šalje na njegovu e-mail adresu. Nakon što korisnik primi privremenu lozinku putem e-maila, on se prijavljuje u sustav koristeći svoje korisničko ime (e-mail adresu) i privremenu lozinku. Odmah nakon uspješne prijave, korisniku se prikazuje obrazac za promjenu inicijalne lozinke. Korisnik unosi novu lozinku koju želi koristiti za pristup sustavu i poslužitelj ažurira podatke u bazi.

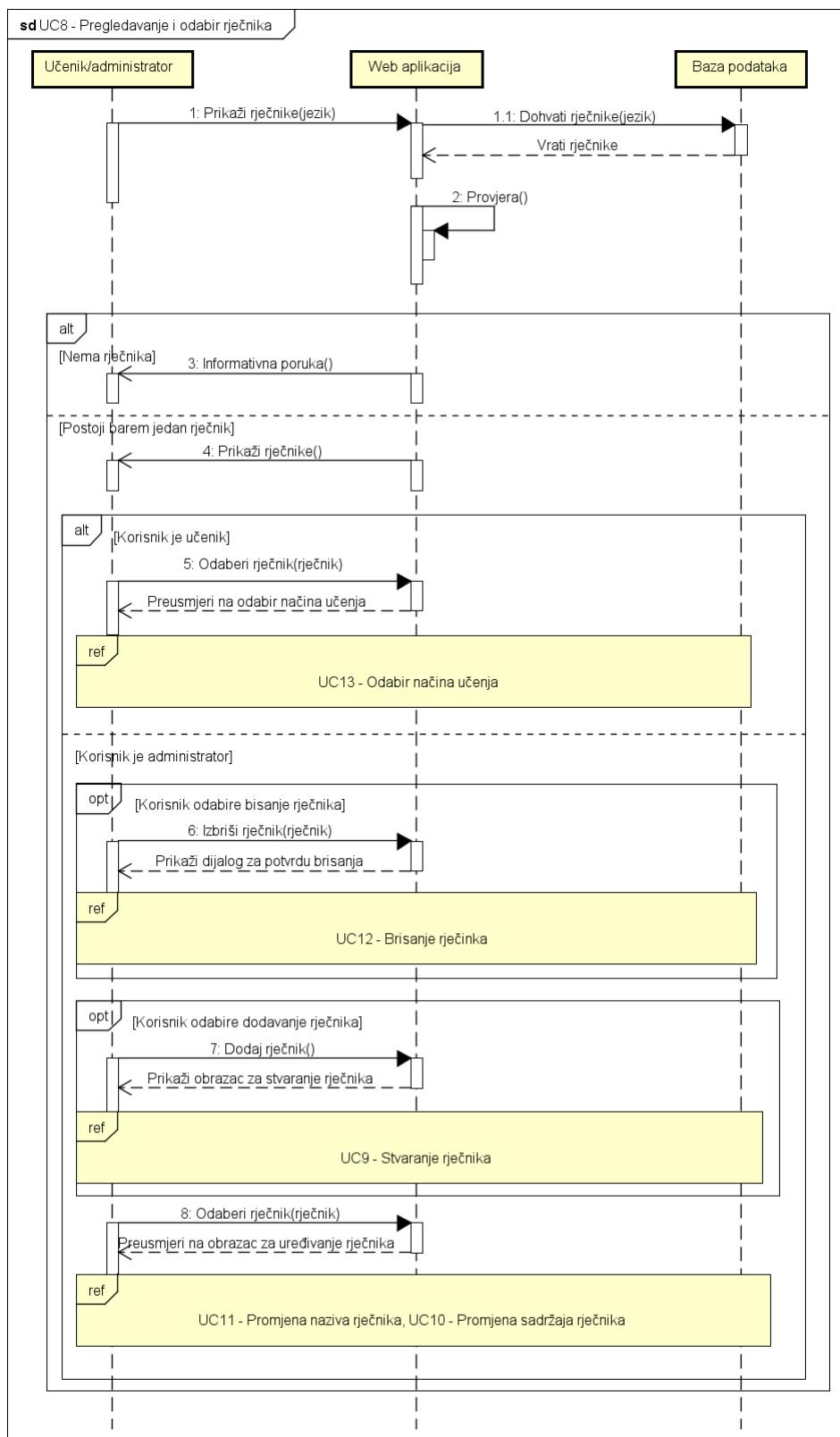


Slika 3.4: Sekvencijski dijagram, UC1 Registracija

### Obrazac uporabe 8 – Pregledavanje i odabir rječnika

Nakon što je odabrao jezik, korisniku se treba prikazati popis svih dostupnih rječnika za taj jezik, a web aplikacija te podatke dohvata slanjem upita na bazu podataka. Prije prikazivanja podataka korisniku, web aplikacija provjerava koliko je rječnika vratila baza. U slučaju da baza nije vratila nijedan rječnik, korisniku se prikazuje poruka da za odabrani jezik trenutno ne postoji nijedan rječnik. Inače se korisniku prikazuje popis rječnika.

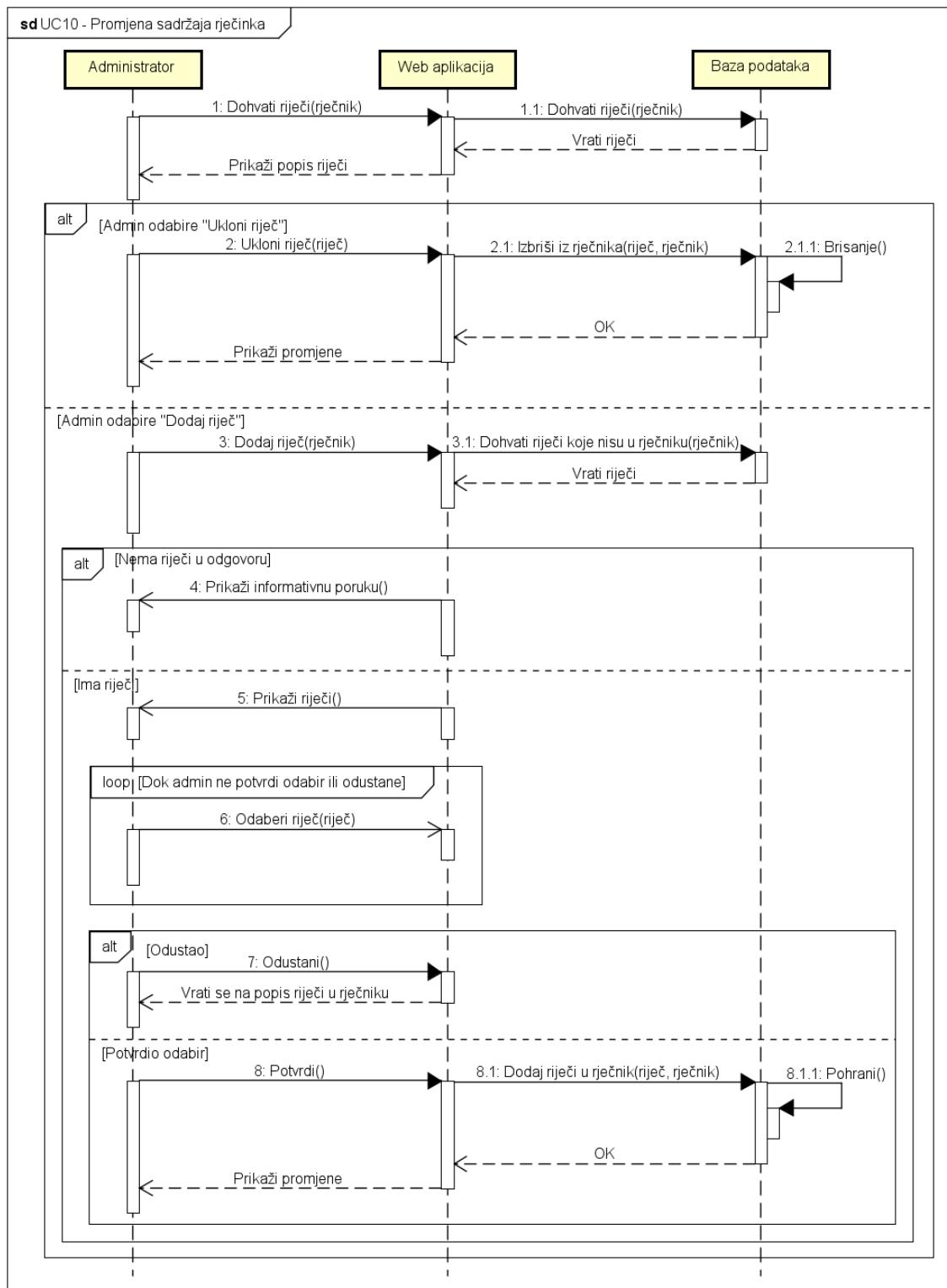
I administratori i učenici mogu pristupati odabiru rječnika. Odabirom rječnika web aplikacija preusmjerava učenika na zaslon za odabir načina učenja, dok administratora preusmjerava na zaslon sa detaljnijim sadržajem rječnika (popisom riječi u rječniku, nazivom rječnika). U slučaju da administrator ne odabere rječnik, već zatraži brisanje rječnika, web aplikacija mu prikazuje okvir za potvrdu brisanja rječnika preko kojeg može izbrisati rječnik iz sustava. Administrator također može odabratи opciju za stvaranje novog rječnika, nakon čega ga web aplikacija preusmjerava na zaslon s obrascem za stvaranje rječnika.



Slika 3.5: Sekvencijski dijagram, UC8 Pregledavanje i odabir rječnika

**Obrazac uporabe 10 - Promjena sadržaja rječnika**

Administrator šalje zahtjev web aplikaciji za popis svih riječi u rječniku. Web aplikacija dohvaća popis riječi iz baze podataka te ga prikazuje korisniku. Ako administrator odabere opciju "ukloni" pored određene riječi, tada šalje zahtjev za uklanjanje aplikaciji, a aplikacija šalje zahtjev za brisanje riječi u bazi podataka i potvrđuje brisanje administratoru. U slučaju da administrator odabere opciju "dodaj riječ", aplikacija dohvaća riječi koje se ne nalaze u rječniku iz baze podataka i prikazuje ih administratoru. Ako nema dostupnih riječi, aplikacija šalje informativnu poruku administratoru umjesto popisa. U suprotnom, administrator odabire jednu ili više riječi, potvrđuje odabir, i riječi se pohranjuju u bazu podataka putem aplikacije.



Slika 3.6: Sekvencijski dijagram, UC10 Promjena sadržaja rječnika

### 3.2 Ostali zahtjevi

- sustav korisniku daje povratnu informaciju o točnosti njegova odgovora
- sustav u razumnom vremenu prezentira riječi nakon odabira načina rada
- sustav mora imati potporu hrvatskih dijakritičkih znakova
- sustavu iz javne mreže pristupamo protokolom HTTPS
- sustav je dovoljno jednostavan i intuitivan za bilo koju dobnu skupinu korisnika
- za korištenje sustava korisniku je potrebno poznавanje hrvatskog jezika
- prijevodi riječi moraju biti ispravni

## 4. Arhitektura i dizajn sustava

Naš se sustav može podijeliti na tri podsustava:

**Web preglednik:** Lokalno instalirani program koji omogućuje prikaz sadržaja sa interneta. Pomoću tog programa korisnik može poslati zahtjeve za resursima ili poslati neke podatke web poslužitelju. Web preglednik, nakon što dobije zatražene informacije od web poslužitelja, korisniku prikazuje sadržaj.

**Web poslužitelj:** Centralni podsustav aplikacije koji obrađuje višestruke zahtjeve korisnika. Komunikaciju s klijentima ostvaruje preko HTTP protokola. Web poslužitelj može, na korisnički zahtjev, poslati datoteke ili primiti neke podatke koje može spremiti u bazu podataka.

**Baza podataka:** Koristi se za pohranjivanje podataka web aplikacije. Sastavljena je od relacija koje predstavljaju entitete s određenim atributima koji su međusobno povezani definiranim odnosima. Web aplikacija često komunicira s bazom, povlačeći i davajući podatke bazi.

Klijent-server arhitektura uobičajen je način organizacije web aplikacija. U takvoj je arhitekturi sustav organiziran kao skup jednog ili više servisa koji mrežom komuniciraju s odgovarajućim klijentima. Glavna prednost ove arhitekture je to što distribuiranjem odgovornosti na više manjih servisa gradimo modularni sustav koji olakšava razvoj i otporniji je na greške; promjenom ili dodavanjem određenog servisa ne utječemo na ostatak sustava. Iako u našoj implementaciji ne iskorištavamo sve prednosti ove arhitekture jer nam je potreban samo jedan server (servis za pristup bazi podataka, backend) i jedan programski klijent (sustav za prikazivanje podataka u web pregledniku, frontend), odvajanje odgovornosti znatno nam je olakšalo razvoj aplikacije.

Servis za pristup bazi podataka razvijen je u Pythonu, koristeći programski mikrookvir Flask. U svojoj osnovi Flask je znatno manji od sličnih okvira (Django, FastAPI) te pruža lakši i brži razvoj uz sve potrebne funkcionalnosti za razvoj HTTP programskog sučelja. Od dostupnih ORM-ova, Flask se najbolje može integrirati sa SQLAlchemyjem pa je on korišten za spajanje na PostgreSQL bazu podataka, stvaranje i izvršavanje upita te upravljanje preslikavanjem relacija iz baze

u Python objekte. Za serijalizaciju objekata iz baze (posebice prilikom oblikovanja HTTP odgovora) korišten je Marshmallow kao vrlo malena i učinkovita biblioteka.

Web sučelje aplikacije razvijeno je u okruženju React, a pokreće ga Node.js. Razvijen u Meti, React je Javascript biblioteka za izgradnju korisničkih sučelja kompozicijom komponenti pomoću kojih se optimizira proces prikazivanja i osvježavanja sučelja. S velikom zajednicom koja ga koristi i razvija te iznimno kvalitetnom podrškom React je bio najpouzdaniji odabir za tehnologiju za razvoj frontenda. U razvoju smo koristili Typescript kao pouzdaniju inačicu Javascripta u smislu izbjegavanja logičkih pogreški prilikom razvoja strogom i statičkom provjrom tipova podataka.

Kako bismo centralizirali pohranjivanje stanja aplikacije i odvojili logiku upravljanja tim stanjem od logike izgradnje komponenti, uz React koristili smo i Redux. Uvođenjem Reduxa, MVC arhitektura nametnula se kao intuitivan način za organizaciju logike prikazivanja podataka korisniku. Naime, iako se određene funkcionalnosti Reacta mogu koristiti za pohranjivanje stanja aplikacije i upravljanje njime, korištenjem samo Reacta otežali smo razvoj i skaliranje aplikacije. Tako se u našoj implementaciji stanje aplikacije pohranjuje u Reduxovom storeu (model), prikazuje se korisniku preko sučelja napravljenog u Reactu (view), a osvježavanje ili mijenjanje prikazanih podataka odgovornost je Reduxovih reducera (controller).

## 4.1 Baza podataka

Naš sustav koristi bazu podataka koja se temelji na relacijskom modelu podataka i koristi tablice (relacije) za organizaciju i pohranu podataka. Entiteti naše baze su:

- Jezik
- Rječnik
- Riječ
- Korisnik
- Fraza
- Posuda

#### 4.1.1 Opis tablica

**Korisnik** Entitet Korisnik sadrži sve informacije o korisniku. Atributi korisnika su: userId(primarni ključ), ime, prezime, email, lozinka, korisnikStvorenNa, uloga i promijenioLozinku. Korisnik je u *Many-To-One* vezi s tablicom StanjeRiječi.

Korisnik		
userId	INT	Identifikacijski ključ korisnika
ime	VARCHAR	Korisnikovo ime
prezime	VARCHAR	Korisnikovo prezime
email	VARCHAR	Korisnikov email
lozinka	VARCHAR	Korisnikova lozinka
korisnikStvorenNa	DATETIME	Datum stvaranja računa
uloga	VARCHAR	Korisnikova uloga (učenik ili admin)
promijenioLozinku	BOOLEAN	Oznaka je li korisnik promijenio lozinku

**Rječnik** Entitet Rječnik sadrži sve informacije o rječniku. Njegovi atributi su su: rječnikId(primarni ključ), nazivRječnik, rječnikStvorenNa i jezikId. Rječnik je u *Many-To-One* vezi s tablicom RječnikRiječi te je u *Many-To-One* vezi s entitetom Jezik.

Rječnik		
rječnikId	INT	Identifikacijski ključ rječnika
nazivRječnik	VARCHAR	Naziv rječnika
rječnikStvorenNa	DATE	Datum stvaranja rječnika
jezikId	INT	Identifikacijski ključ jezika rječnika

**Riječ** Entitet Riječ sadrži sve informacije o riječi u sustavu. Njeni atributi su su: riječId(primarni ključ), hrvNaziv, straniNaziv, audioPath i jezikId. Entitet Riječ je u *Many-To-One* vezi s entitetom Jezik, u *Many-To-One* vezi s tablicom RiječRječnik, u *One-To-Many* vezi s entitetom Fraza te u *Many-To-One* vezi s tablicom StanjeRiječi.

Riječ		
riječId	INT	Identifikacijski ključ riječi
hrvNaziv	VARCHAR	Naziv riječi na hrvatskom
straniNaziv	VARCHAR	Naziv rječi na stranom jeziku
audioPath	VARCHAR	Put do audio datoteke s izgovorom riječi
jezikId	INT	Identifikacijski ključ jezika rječnika

**Jezik** Entitet Jezik sadrži sve informacije o nekom jeziku u sustavu. Njegovi atributi su su: jezikId(primarni ključ), nazivJezik i isoOznaka. Entitet Jezik je u *One-To-Many* vezi s entitetom Rječnik te u *One-To-Many* vezi s entitetom Riječ.

Jezik		
jezikId	INT	Identifikacijski ključ jezika
nazivJezik	VARCHAR	Naziv jezika
isoOznaka	CHAR(2)	ISO oznaka jezika

**Fraza** Entitet Fraza sadrži sve informacije o nekoj frazi u sustavu. Njeni atributi su su: frazaId(primarni ključ), fraza i riječId. Entitet Fraza je u *Many-To-One* vezi s entitetom Riječ.

Fraza		
frazaId	INT	Identifikacijski ključ fraze
fraza	VARCHAR	Puna fraza
rijecId	INT	Identifikacijski ključ riječi iz fraze

**RiječRječnik** Tablica RiječRječnik povezuje sve riječi s pripadnim rječnikom. Njeni atributi su su: riječId(primarni ključ) te rječnikId(primarni ključ). Tablica RiječRječnik je u *One-To-Many* vezi s entitetom Riječ i u *One-To-Many* vezi s entitetom Rječnik.

RiječRječnik		
riječId	INT	Identifikacijski ključ riječi
rječnikId	INT	Identifikacijski ključ rječnika

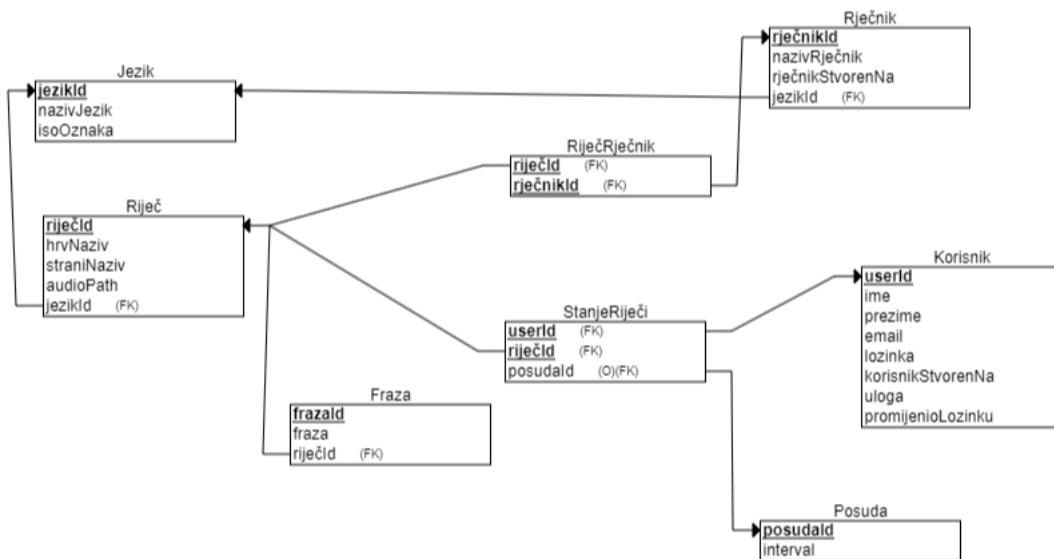
**StanjeRiječi** Tablica StanjeRiječi povezuje korisnika s riječima i njima pripadnim "posudama". Njeni atributi su su: userId(primarniKljuč), riječId(primarniKljuč) i posudaId. Tablica StanjeRiječi je u *One-To-Many* vezi s entitetom Riječ, u *One-To-Many* vezi s entitetom Korisnik te u *One-To-One* vezi s entitetom Posuda.

StanjeRiječi		
userId	INT	Identifikacijski ključ korisnika
rijecId	INT	Identifikacijski ključ riječi
posudaId	INT	Identifikacijski ključ posude

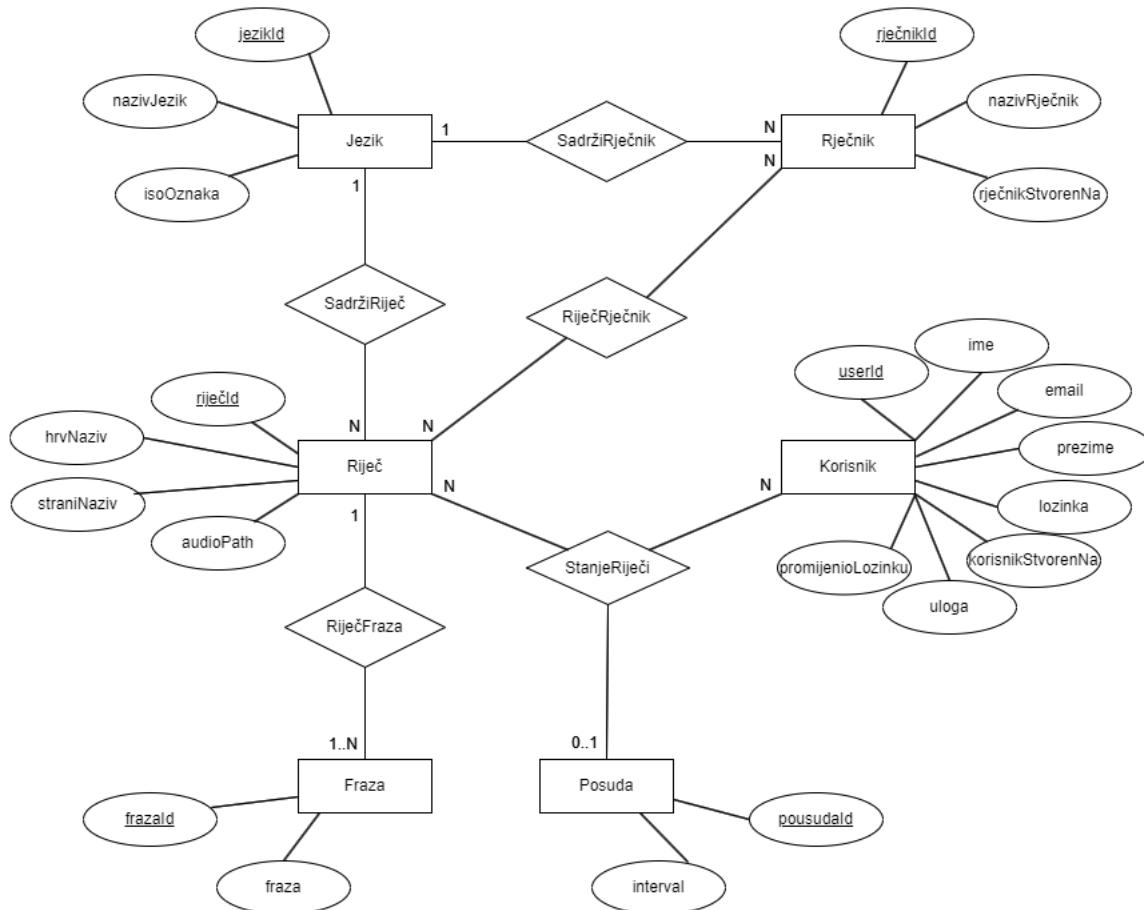
**Posuda** Entitet Posuda predstavlja sve posude i njihove intervale. Njeni atributi su su: posudaId(primarni ključ) i interval. Tablica Posuda je u *One-To-One* vezi s tablicom StanjeRiječi.

Posuda		
posudaId	INT	Identifikacijski ključ posude
interval	INT	Interval čekanja do ponovnog pojavljivanja riječi

#### 4.1.2 Dijagram baze podataka



Slika 4.1: Relacijska shema baze podataka



Slika 4.2: ER model baze podataka

## 4.2 Dijagram razreda

### 4.2.1 Razredi na backendu

Ova se skupina razreda (Slika 4.3) izravno preslikava u bazu podataka tako da se za svaki razred stvara jedna tablica u bazi. Dakle, svi su razredi svedeni na treću normalnu formu i ni u kojem slučaju ne mogu biti povezani pridruživanjem (agregacijom ili asocijacijom).

#### **db.Model**

Razred koji dolazi iz SQLAlchemy koji koristimo u pozadini za komunikaciju s bazom podataka. Ovaj razred nasljeđuju svi razredi u skupini modela na backendu zato što on pruža potrebne funkcionalnosti za slanje i povlačenje podataka u bazu.

#### **Razred Jezik**

Služi za modeliranje jezika te je opisan atributima identifikatora, naziva jezika i ISO oznake. Atributi su privatni, te ima javni konstruktor.

#### **Razred Riječ**

Služi za modeliranje riječi, najmanje jedinice učenja u aplikaciji. Razred sadrži attribute: identifikator riječi, hrvatski naziv, strani naziv, putanju na audiodatoteku izgovora riječi na stranom jeziku i identifikator jezika na kojem su audiozapis i strani naziv. Atributi su privatni i ima javni konstruktor.

#### **Razred Rječnik**

Služi za modeliranje rječnika, tj. kolekcije riječi. Razred je opisan atributima identifikatora, naziva rječnika, datuma njegovog stvorenja i jezika riječi koje sadrži. Atributi su privatni te ima javni konstruktor.

#### **Razred StanjeRiječi**

Služi kao razred koji prati stanje riječi, tj. je li riječ naučena. Opisan je s atributima: identifikatora korisnika, identifikatora riječi i identifikatora posude. Atributi su privatni i ima javni konstruktor.

## Razred Posuda

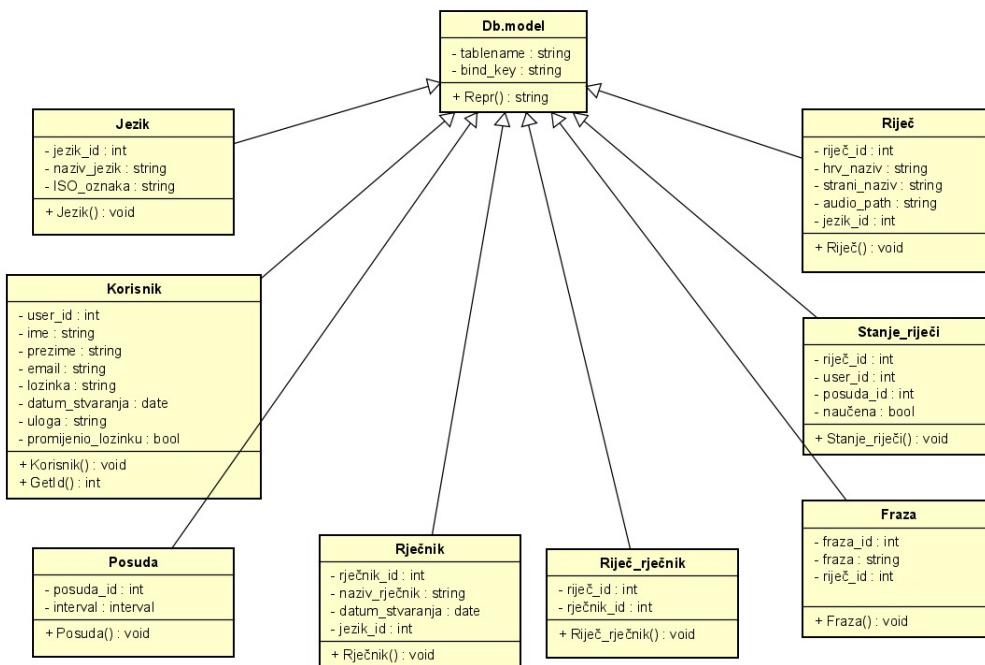
Služi za modeliranje posude, opisan sa identifikatorom posude i intervalom. Atributi su privatni i ima javni konstruktor.

## Razred Fraza

Služi za modeliranje fraze koja prati riječ na stranom jeziku za lakše učenje konteksta. Razred je opisan s atributima identifikatora fraze, samom frazom i identifikatorom riječi uz koju je fraza povezana. Atributi su privatni i ima javni konstruktor.

## Razred Korisnik

Služi za modeliranje korisnika. Baza podataka razlikuje administratore i učenike po njihovoj ulozi, ne smatra ih različitim entitetima. Razred opisuju atributi identifikatora korisnika, imena, prezimena, adrese elektroničke pošte, lozinke za račun, trenutka stvaranja korisnika, uloge i zastavice koja provjerava promjenu lozinke. Atributi su privatni i razred ima javni konstruktor.



Slika 4.3: Dijagram razreda za backend dio aplikacije

#### 4.2.2 Razredi na frontendu

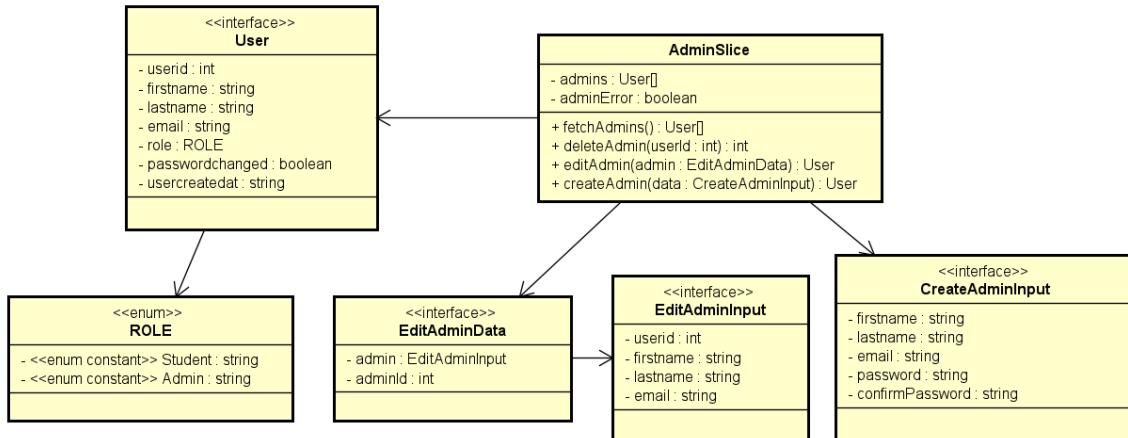
Razredi na frontend dijelu aplikacije podijeljeni su u tri skupine. Isječke (slices) u kojima se spremaju podaci tipa Models, a čije metode primaju podatke iz Inputs skupine razreda. Kako su isječci međusobno neovisni, ali i radi preglednosti, razredi na klijentskom dijelu aplikacije prikazani su u više dijagrama, ovisno o isječku u kojem se koriste.

**Modeli** su reprezentacije entiteta s backend dijela aplikacije. Backend modeli ne preslikavaju se izravno u odgovarajuće frontend modele, već se proširuju s dodatnim podacima iz povezanih tablica. Tako, primjerice, dok razredi Riječ i Fraza na backendu nisu povezani, razred Riječ na frontendu u sebi ima listu objekata razreda Fraza.

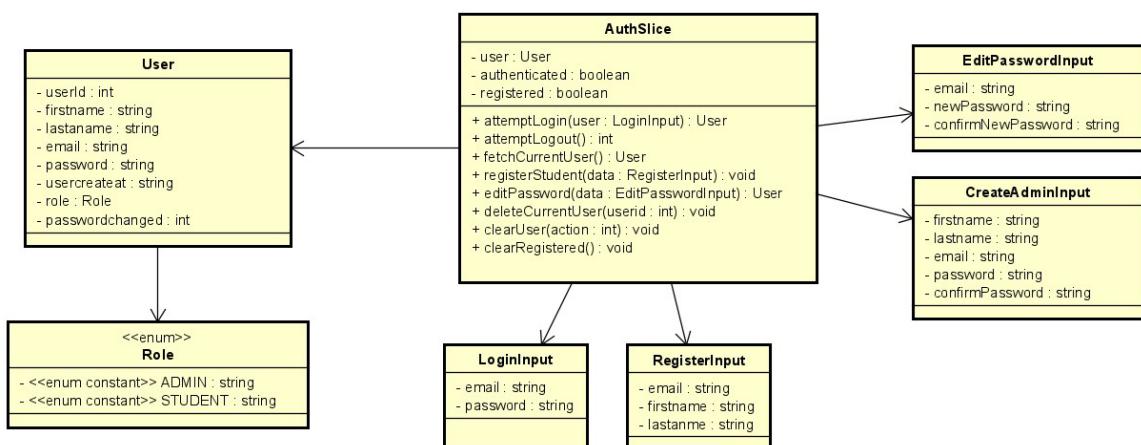
**Input** razredi predstavljaju *data transfer* objekte. Objekti tog razreda izravno se mogu pohraniti u tijelo HTTP zahtjeva i odgovaraju onoj strukturi u kojoj pojedina pristupna točka na backendu očekuje primiti podatke.

**Slice** razredi specifični su za implementiranu arhitekturu koja koristi Redux za upravljanje stanjem aplikacije. S ciljem odvajanja odgovornosti, stanje cijele aplikacije podijeljeno je na proizvoljan broj samostalnih cjelina (isječak, *slice*) koje upravljuju i pohranjuju podatke iz isključivo jedne domene. Stanje u određenom isječku može se mijenjati samo preko njegovih metoda, a unutar njih se nerijetko pozivaju *axios* metode za slanje HTTP zahtjeva backend dijelu aplikacije jer se rezultati tih poziva moraju spremati upravo u stanje aplikacije. Ukupno je implementirano sedam razreda za isječke stanja:

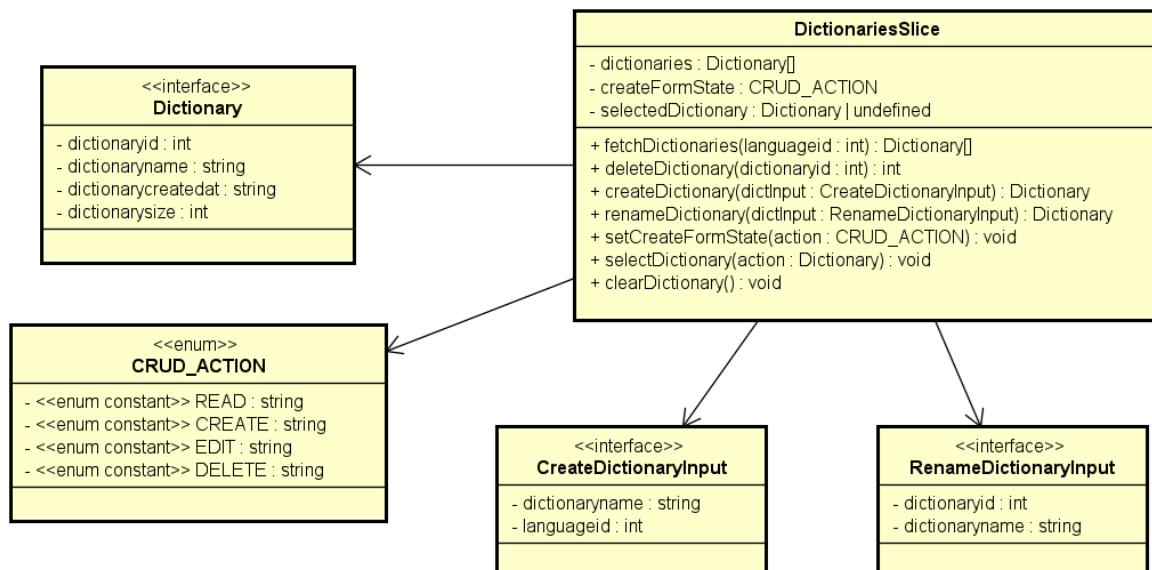
1. *Admin* – upravljanje administratorima
2. *Auth* – podaci o trenutno prijavljenom korisniku
3. *Dictionaries* – upravljanje rječnicima
4. *Language* – dostupni jezici i trenutno odabran jezik
5. *Student Dictionaries* – rječnici nekog jezika s dodatnim podacima koji se prikazuju učeniku
6. *Study Session* – podaci potrebni za učenje u bilo kojem načinu učenja
7. *Words* – upravljanje riječima



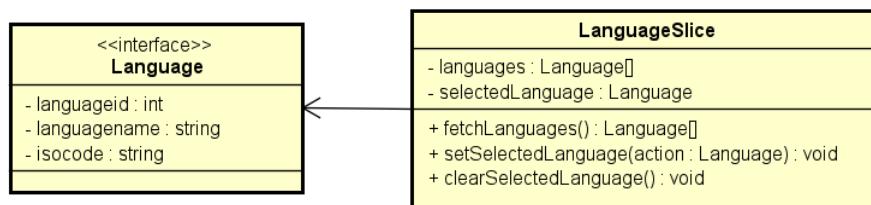
Slika 4.4: Dijagram razreda za slice Admin



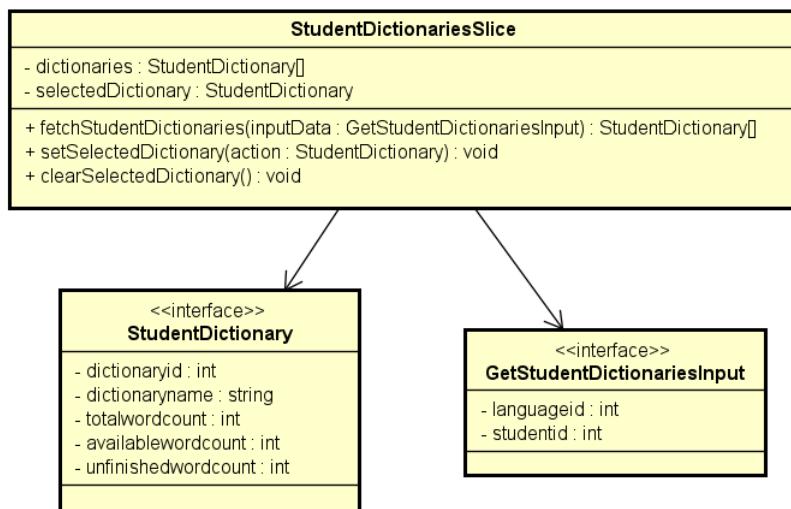
Slika 4.5: Dijagram razreda za slice Auth



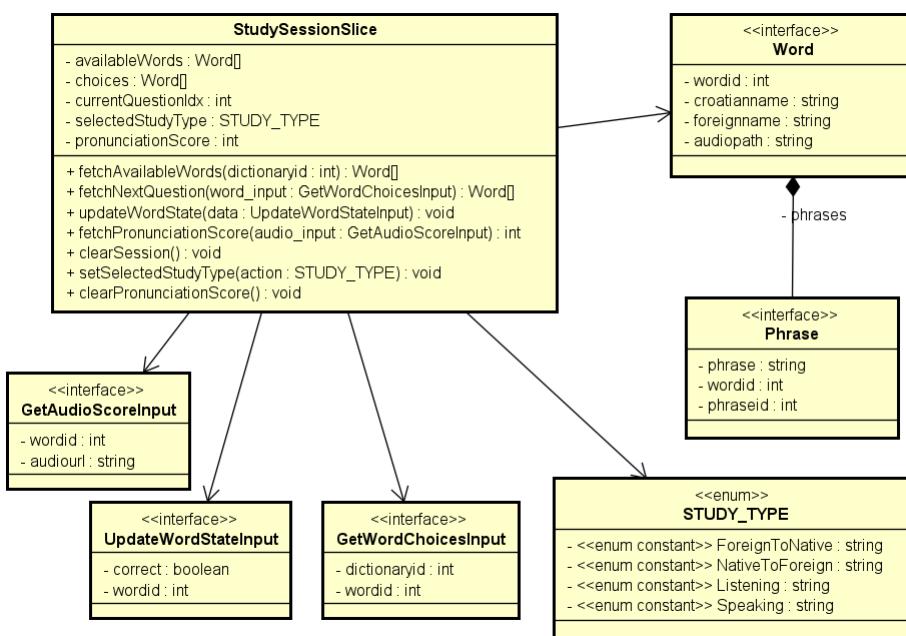
Slika 4.6: Dijagram razreda za slice Dictionaries



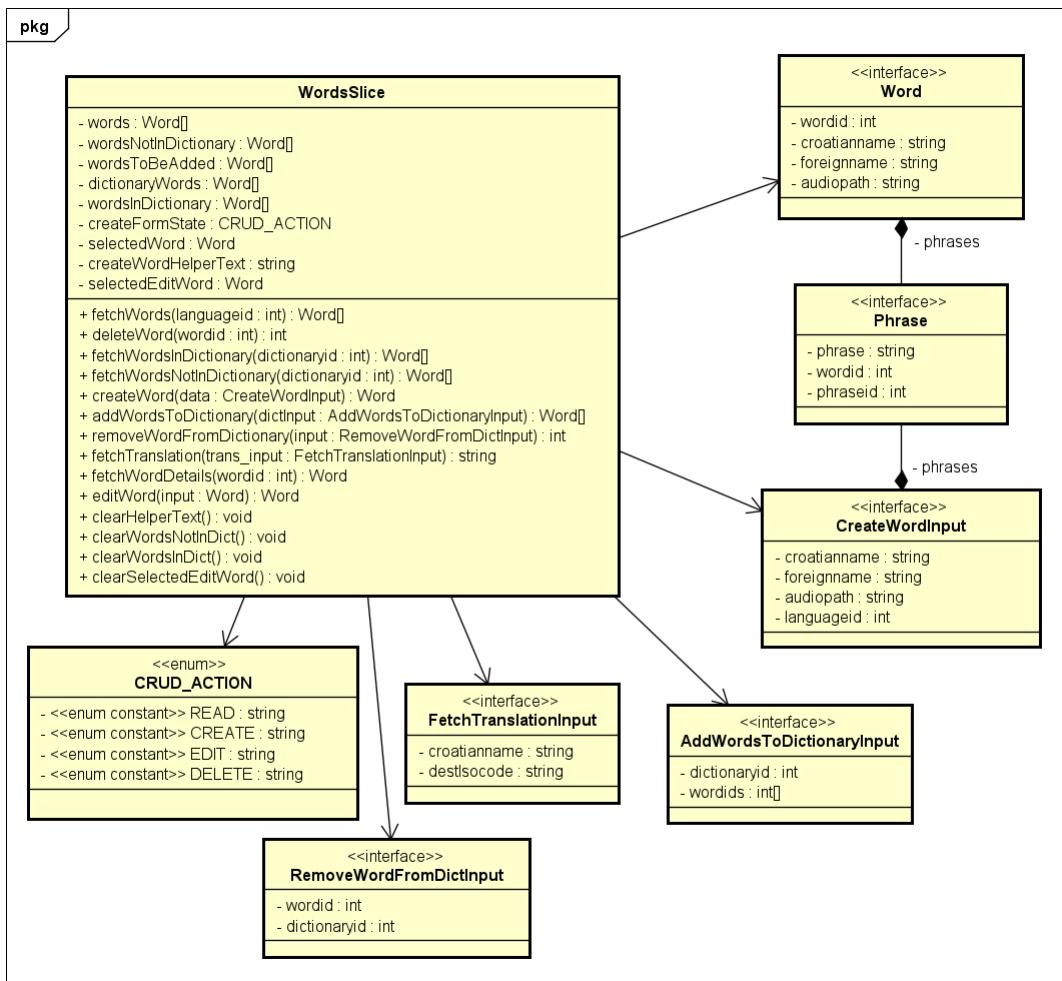
Slika 4.7: Dijagram razreda za slice Language



Slika 4.8: Dijagram razreda za slice Student Dictionaries



Slika 4.9: Dijagram razreda za slice Study Session

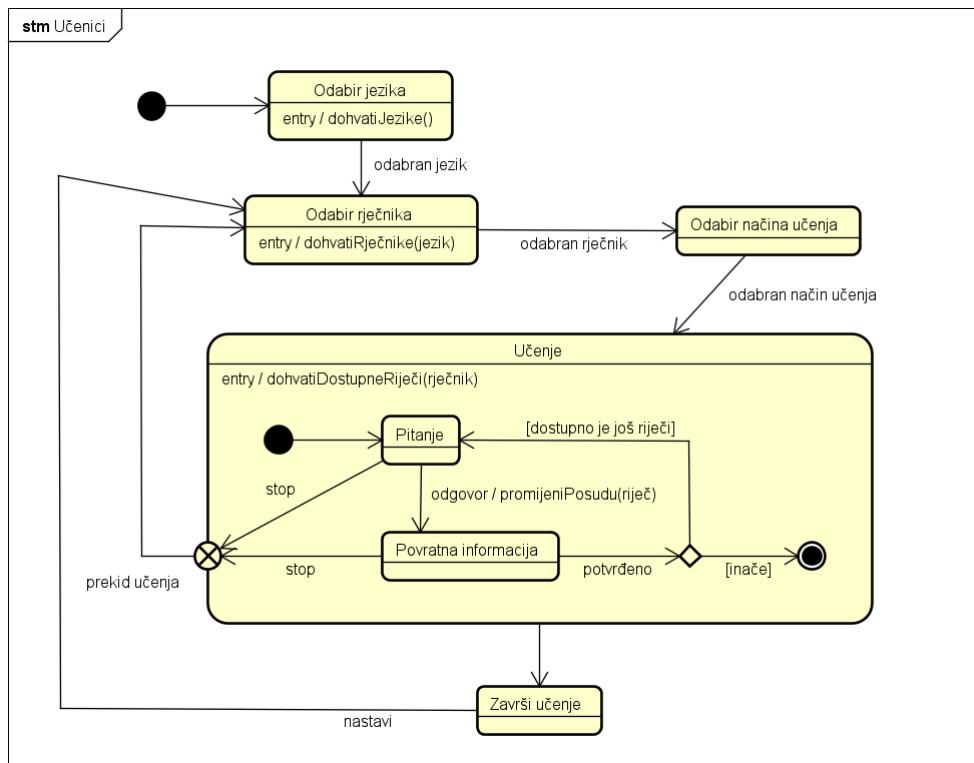


Slika 4.10: Dijagram razreda za slice Words

## 4.3 Dijagram stanja

Slika 4.11 prikazuje implementirane funkcionalnosti korisničkog sučelja za učenike. Učenici započinju učenje tako što redom odabiru jezik, rječnik i način učenja. Kako bi se prikazali dostupni jezici i rječnici, prije ulaska u svako od tih stanja poziva se metoda koja dohvaća odgovarajuće podatke iz baze podatka. Netom prije početka učenja dohvaćaju se sve riječi koje su učeniku trenutno dostupne za učenje.

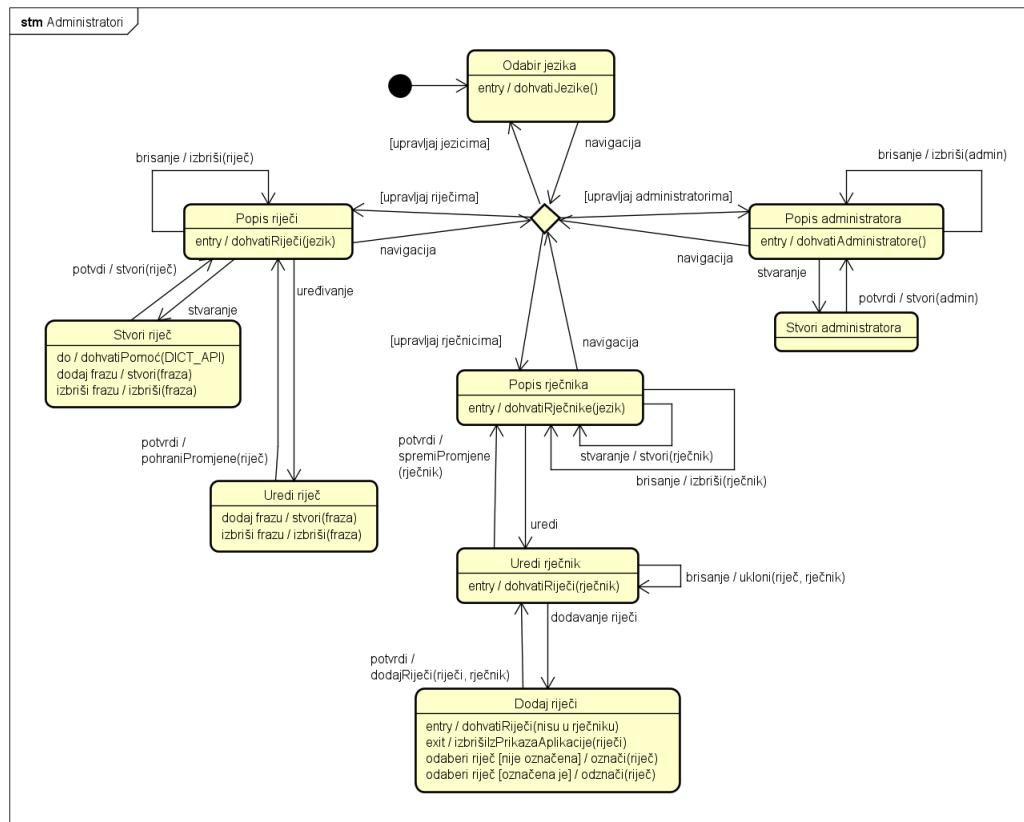
Tijekom učenja učeniku se naizmjence prikazuju pitanja i povratne informacije o unesenim odgovorima sve dok ima dostupnih riječi ili dok učenik ne zaustavi učenje. Nakon unesenog odgovora poziva se metoda koja premješta riječ u odgovarajući posudu. Učenik može zaustaviti učenje prije i nakon odgovora na pitanje, pri čemu odlazi u stanje za odabir rječnika. U to stanje odlazi i nakon što je odgovorio na sve trenutno dostupne riječi.



Slika 4.11: Dijagram stanja - sučelje za učenike

Stanja administratorskog sučelja prikazana su Slikom 4.12. Četiri glavna stanja u kojima se to sučelje može nalaziti su *Odabir jezika*, *Popis administratora*, *Popis rječnika* i *Popis riječi*. Administrator može doći u svako od tih stanja preko navigacijske trake, a ulaskom u to stanje dohvaćaju se podaci koji se trebaju prikazati na

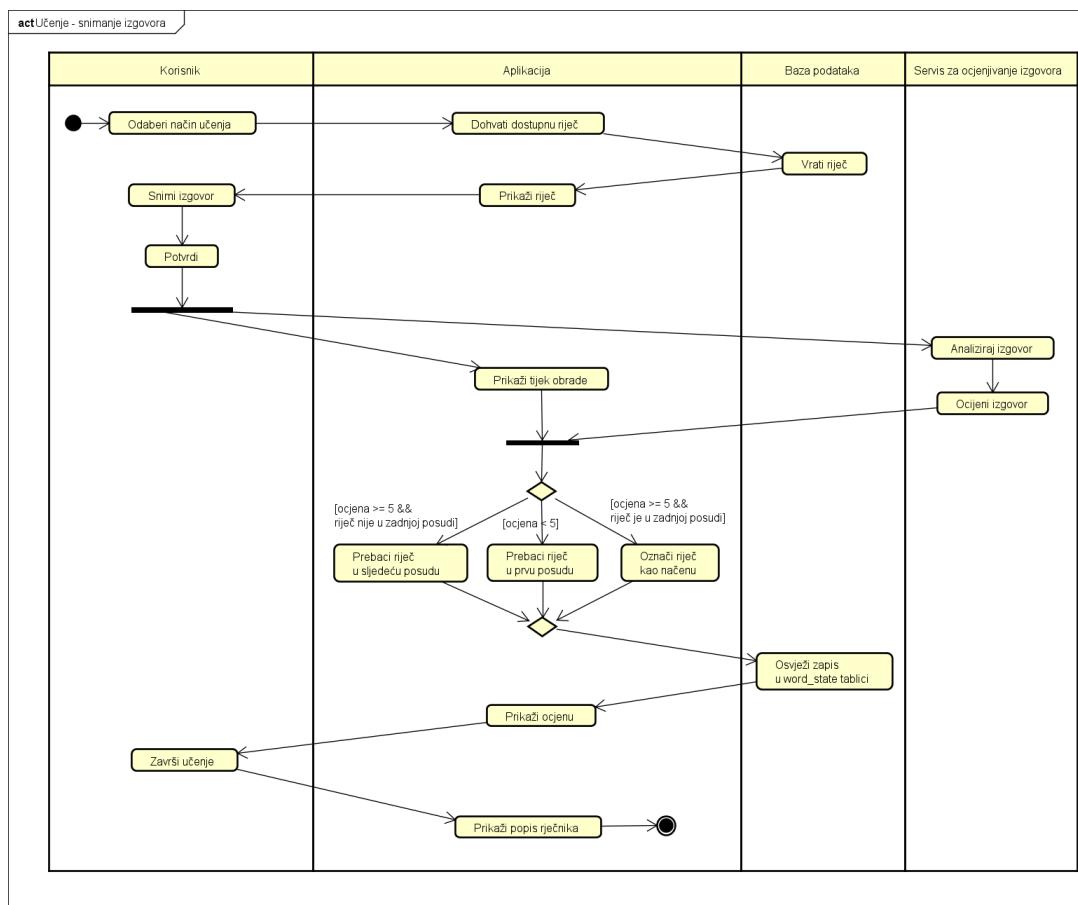
sučelju. Izuzev odabira jezika, administrator u svakom od tih stanja može brisati i stvarati odgovarajuće zapise. Pri stvaranju riječi dohvaćaju se podaci iz vanjskog API-ja i mogu se dodavati ili brisati fraze. Riječi i rječnici mogu se uređivati. Uređivanjem rječnika administrator iz njega uklanja riječi ili dodaje nove.



Slika 4.12: Dijagram stanja - sučelje za administratore

## 4.4 Dijagram aktivnosti

Slikom 4.13 prikazano je učenje snimanjem izgovora. Odabirom načina učenja aplikacija dohvaća dostupne riječi i učeniku prikazuje pitanje. Nakon što učenik snimi izgovor i potvrdi unos, sustav obrađuje njegov odgovor: servis za ocjenjivanje izgovora analizira unos i vraća ocjenu, dok se učeniku prikazuje tijek obrade. U slučaju da je ocjena strogo manja od 5, riječ se prebacuje u prvi posudu. U protivnom se riječ prebacuje u sljedeću posudu ako ona postoji, inače se označava kao u potpunosti naučena. Promjena posude sprema se u bazu podataka, a zatim se učeniku prikazuje ocjena. Klikom na gumb završi učenje aplikacija preusmjerava korisnika na stranicu s popisom rječnika.

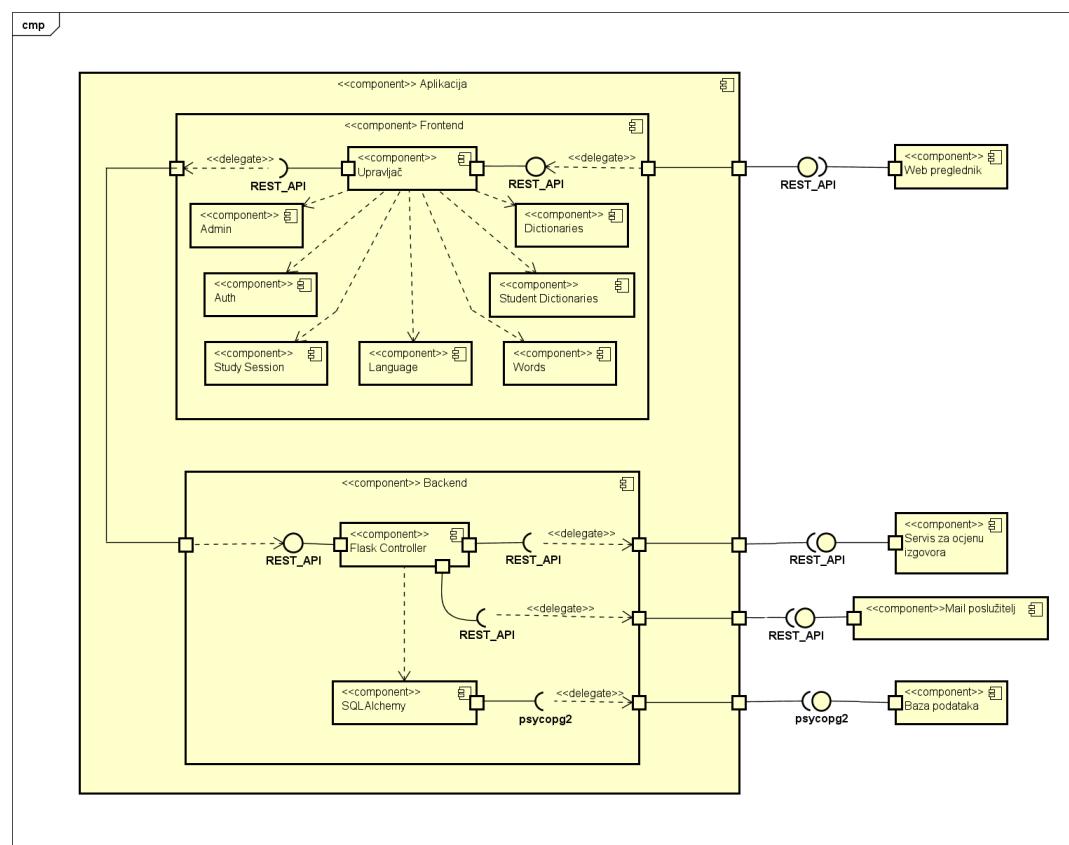


Slika 4.13: Dijagram aktivnosti za učenje sa snimanjem izgovora

## 4.5 Dijagram komponenti

Klijentska komponenta aplikacije organizirana je u međusobno neovisne module za pohranu stanja aplikacije i jedne upravljačke komponente preko kojeg moduli komuniciraju s ostatkom aplikacije. Svaki modul zadužen je za određeni isječak cijelog stanja aplikacije. Upravljačka komponenta nudi sučelje preko kojeg preglednik može dohvaćati podatke o stanju, a zahtjeva sučelje preko kojeg može slati zahtjeve na krajnje točke serverskog dijela aplikacije.

Serverska komponenta sadrži upravljačku komponentu Flask za primanje i obradu HTTP zahtjeva. Metode te komponente koriste modul SQLAlchemy za izvršavanje SQL upita i pristup bazi podataka. Baza podataka nudi određeno sučelje, a SQLAlchemy ga zahtijeva. Komponenta Flask pri obradi određenih zahtjeva komunicira sa servisom za ocjenu izgovora i mail poslužiteljem te od njih zahtijeva REST API sučelje.



Slika 4.14: Dijagram komponenti

# 5. Implementacija i korisničko sučelje

## 5.1 Korištene tehnologije i alati

### 5.1.1 Korišteni alati

Alati koji su bili korišteni za izradu aplikacije nisu divergirali od standarda u industriji. Za pisanje programskog koda i skripti, sa poslužiteljske i korisničke strane, korišten je Microsoftov Visual Studio Code. Jednostavan program čija osnovna funkcionalnost uključuje sve klasične operacije nad tekstrom koji predstavlja kod: automatsko naglašavanje teksta bojom ovisno o programskom jeziku koji se koristi, mnoštvo kratica na tipkovnici za manipulaciju teksta, ubacivanje ili uklanjanje komentara, tabulatora i sl. VSC se također može specijalizirati za razvoj bilo koje vrste projekta pomoću dodataka koji su javno dostupni svima i mogu se opcionalno instalirati.

Za upravljanje bazom podataka korišten je pgAdmin. Program unutar kojeg se koristi SQL kako bi se definirala relacijska baza podataka, sve njene tablice, te konačno, podatci. Poveznicu između njih čini kod napisan za poslužiteljsku stranu u Pythonu, čije su krajnje API točke testirane pomoću alata Postman. Postman omogućuje slanje HTTP zahtjeva na bilo koju adresu, te dopušta absolutnu kontrolu nad poslanim paketom. Pomoću tog alata je testirano prima li poslužiteljska strana pravilno informacije koje se pošalju (npr. informacije za registraciju), ili dobivaju li se točni podatci iz baze kad ih se zatraži (npr. sve riječi nekog rječnika).

### 5.1.2 Korištene tehnologije

Korištene tehnologije su sve u skladu razvoja RESTful aplikacije, što samo zapravo znači da se komunikacija klijenta i poslužitelja odvija preko HTTP ili HTTPS protokola, sa GET, POST, PUT i DELETE zahtjevima.

Na korisničkoj strani je korišten javascript radni okvir React, koji je razvila Meta (bivši Facebook), koji je zasnovan na komponentama. Svaka komponenta može imati svoju definiranu funkcionalnost, te može sadržavati druge predefinirane ili korisnički definirane komponente. Od klijentskih programskih jezika korišten je

TypeScript zbog pouzdanosti.

Za neke dijelove korisničkog sučelja je korišten MUI (Material User Interface) koji ima nekolicinu predefiniranih komponenti, kao npr. gume, tablice ili kartice za prikaz podataka. MUI je napravljen za React i ne može se koristiti sa nekom drugom bibliotekom. Biblioteka Styled Components korištena je za proširivanje komponenti MUI-ja CSS-om.

Za upravljanje stanjem aplikacije (tzv. state) korišten je Redux. Redux je mala biblioteka sa jednostavnim API-jem. Bazirana je na logici da je svako novo stanje aplikacije redukcija prijašnjeg, što zapravo znači da se svako novo stanje računa pomoću redukcijske funkcije. Ukupno stanje aplikacije se dijeli u isječke (slices), od kojih je svaki odgovoran za praćenje stanja jednog dijela aplikacije.

Korišteni HTTP klijent je Axios. Za upravljanje formama u aplikaciji korištena je biblioteka React Hook Form, a za routing React Router.

Na poslužiteljskoj strani implementacijski jezik je Python 3, a korištene su biblioteke Flask i nekolicina popratnih tehnologija koje služe za proširivanje funkcionalnosti Flaska. Korisničke sesije ostvarene su na poslužiteljskoj strani pomoću biblioteke Flask-Login. Od ORM-ova odabran je Flask-SQLAlchemy, a za serijalizaciju podataka korišten je Marshmallow. Migracijama baze podataka upravljali smo bibliotekom Flask-Migrate.

Google Translate Ajax API (*googletrans*) korišten je kao vanjski API za prevodenje. Za slanje mailova korišten je Brevo (bivši Sendinblue).

Ispitivanje komponenti provedeno je pomoću biblioteke pytest, dok je za ispitivanje sustava korišten Selenium WebDriver.

### 5.1.3 Poveznice na korištene tehnologije i alate

- Visual Studio Code
- PgAdmin
- Postman
- Python
- TypeScript
- React
- MUI

- Styled Components
- Redux
- Axios
- React Hook Form
- Flask
- Marshmallow
- Flask-SQLAlchemy
- Flask-Migrate
- Flask-Login
- googletrans
- Brevo
- pytest
- Selenium Webdriver

## 5.2 Ispitivanje programskog rješenja

### 5.2.1 Ispitivanje komponenti

Testiranje komponenti proveli smo koristeći razvojni okvir pytest. Specifičnost tog okvira je korištenje kontekstnih funkcija (engl. fixture). Kontekstnim funkcijama definiramo kontekst za izvršavanje ispitnih funkcija: koji će podaci biti dostupni testu, što će se dogoditi prije, a što nakon testa. Kako se u kontekstnoj funkciji naredbom *yield* poziva ispitna funkcija, sve naredbe prije nje predstavljaju *setup* tog testa, a naredbe nakon *teardown*.

Osnovne kontekstne funkcije koje koristimo tijekom testiranja komponenti su one za prijavu administratora odnosno učenika (Slika 5.1)<sup>1</sup>. Također, za pristup kontekstu objekta aplikacije Flask koristimo funkciju *app*, dok funkcija *client* omogućava emulaciju HTTP zahtjeva prema aplikaciji bez pokretanja backenda (Slika 5.2).

Pri definiciji kontekstne funkcije definiramo njen djelokrug, tj. kada će se u procesu testiranja ona pozivati. Zadani djelokrug je *function*, a kontekstna funkcija s tim djelokrugom pozvat će se prije izvršavanja svake ispitne funkcije, dok će se funkcija s djelokrugom *module* pozivati jednom po ispitnom skupu.

Ispitne funkcije organizirane su u više ispitnih skupova (modula) tako da svaki skup testira jedan dio funkcionalnosti aplikacije. Implementirano je ukupno šest ispitnih funkcija u tri ispitna skupa. Rezultati testova prikazani su na Slici 5.12.

#### Ispitni skup *test\_dictionary.py*

- Stvaranje rječnika s praznim nazivom (rubni slučaj, Slika 5.3)
  - Obrascem uporabe 9 definirano je da zahtjev za stvaranje novog rječnika čiji je naziv prazan niz znakova treba odbiti. Očekivani rezultat je HTTP odgovor sa statusom *400 Bad request* jer su podaci u tijelu zahtjeva u neispravnom formatu.
- Stvaranje rječnika bez prethodne prijave (pogreška, Slika 5.4)
  - Kako ispitna funkcija nema *login\_admin* kontekst, očekivano je da će takav zahtjev za stvaranje rječnika biti odbijen i da će status HTTP odgovora biti *401 Unauthorised*.

<sup>1</sup>Radi se o zadanim (default) korisnicima i osigurano je da će njihovi korisnički računi postojati u bazi podataka

```
@pytest.fixture(scope="module")
def login_student(app):
    with app.test_client() as client:
        response = client.post(
            "/api/auth/login",
            json={"email": "ucenik@ucenik.com", "password": "progi123"},
        )

    yield response

    client.post("/api/auth/logout")

@pytest.fixture(scope="module")
def login_admin(app):
    with app.test_client() as client:
        response = client.post(
            "/api/auth/login",
            json={"email": "admin@admin.com", "password": "progi123"},
        )

    yield response

    client.post("/api/auth/logout")
```

Slika 5.1: Kontekstne funkcije za autentifikaciju korisnika

```
@pytest.fixture(scope="session")
def app():
    flask_app.config.update(
        {
            "TESTING": True,
        }
    )

    yield flask_app

@pytest.fixture()
def client(app):
    return app.test_client()
```

Slika 5.2: Osnovne kontekstne funkcije

```
def test_create_dictionary_empty_name(login_admin, client):
    create_dict_input = {"dictionaryname": ""}

    resp = client.post("/api/dictionaries/1", json=create_dict_input)

    assert resp.status_code == 403
```

Slika 5.3: Stvaranje rječnika s praznim nazivom

```
def test_create_dictionary_unauthorised(client):
    create_dict_input = {"dictionaryname": "Test rjecnik"}

    with client:
        resp = client.post("/api/dictionaries/1", json=create_dict_input)

        assert resp.status_code == 401
```

Slika 5.4: Stvaranje rječnika bez prethodne prijave

### Ispitni skup *test\_users.py*

- Stvaranje administratora s nepotpunim podacima (pogreška, Slika 5.5)
  - Za stvaranje administratora u tijelu zahtjeva trebaju biti polja *email*, *firstname*, *lastname* i *password*. U slučaju da barem jedan od tih polja nije prisutan, zahtjev se ne može izvršiti i završava s *400 Bad request* odgovorom.
- Promjena lozinke učenika (redovni slučaj, Slika 5.6)
  - Kako bi stanje u aplikaciji nakon uspješno provedenog testa ostalo isto kao i prije testa, napisana je kontekstna funkcija *revert\_student\_password\_change* (Slika 5.7) koja nakon testa izvršava upit koji u bazu podataka spremi korisnikovu staru lozinku. Uspješan zahtjev za promjenu lozinke u odgovoru vraća podatke o korisniku.

```
def test_create_admin_missing_field(login_admin, client):
    create_admin_input = {
        "firstname": "Test",
        "lastname": "Admin",
        "password": "pass",
    }

    resp = client.post("/api/users/create-admin", json=create_admin_input)

    assert resp.status_code == 400
```

Slika 5.5: Stvaranje administratora s nepotpunim podacima

```
def test_edit_password(login_student, client, revert_student_password_change):
    edit_password_input = {
        "newPassword": "nova loznika",
        "email": "ucenik@ucenik.com",
    }

    resp = client.put("/api/users/edit-password", json=edit_password_input)

    assert resp.status_code == 200
    assert resp.json["email"] == edit_password_input["email"]
```

Slika 5.6: Promjena lozinke učenika

```
@pytest.fixture()
def revert_student_password_change(app, login_student):
    yield

    old_password_data = {"newPassword": "progi123", "email": "ucenik@ucenik.com"}

    with app.app_context():
        user = db.session.execute(
            db.select(User).where(User.email == old_password_data["email"]))
        .scalar()

        user.password = bcrypt.hashpw(
            old_password_data["newPassword"].encode("utf-8"), bcrypt.gensalt())
        .decode()
    db.session.commit()
```

Slika 5.7: Kontekstna funkcija za poništavanje promjene lozinke

### Ispitni skup *test\_words.py*

- Stvaranje riječi (redovni slučaj, Slika 5.8)
  - Za stvaranje nove riječi u tijelu zahtjeva potrebno je imati id jezika u kojem spremamo riječ, hrvatski naziv, strani naziv i naziv spremljene audiodatoteke. U odgovoru se vraća stvorena riječ koja mora odgovarati podacima iz zahtjeva.
  - Kontekstna funkcija *revert\_create\_word* (Slika 5.9) nakon uspješnog testa briše stvorenu riječ iz sustava.
- Započni učenje nad rječnikom (rubni slučaj, Slika 5.11)
  - Kako se u određenim načinima učenja na pitanje odgovara odabirom jedne od četiri ponuđene riječi iz istog rječnika, domenu veličine rječnika možemo podijeliti na particije od  $[0, 3]$  riječi i  $> 4$  riječi.
  - Za rječnike s tri ili manje riječi nije moguće prikazati ponuđene odgovore pa nije moguće započeti učenje nad tim rječnikom i očekujemo da će takav zahtjev biti odbijen sa statusom *403 Forbidden*.
  - Kontekstna funkcija *study\_session\_setup\_teardown* (Slika 5.10) osigurava kontekst za izvršavanje ovog testa: prije izvršavanja stvara novi rječnik, u njega dodaje tri riječi te testu prosljeđuje podatke o rječniku i stvorenim riječima. Nakon testa dodane riječi i rječnik brišu se iz sustava.

```
def test_create_word(login_admin, client, revert_create_word):
    croatianname = revert_create_word

    create_word_input = {
        "croatianname": croatianname,
        "foreignname": "Test foreign name",
        "audiopath": "Dummy audio path",
    }
    languageid = 1

    resp = client.post(f"api/words/{languageid}", json=create_word_input)

    assert resp.status_code == 200
    assert resp.json["croatianname"] == croatianname
    assert resp.json["foreignname"] == "Test foreign name"
    assert resp.json["audiopath"] == "Dummy audio path"
    assert resp.json["languageid"] == languageid
```

Slika 5.8: Stvaranje riječi

```
@pytest.fixture()
def revert_create_word(app, login_admin):
    croatianname = "Test hrvatski naziv"

    yield croatianname

    with app.app_context():
        created_word = db.session.execute(
            db.select(Word).where(Word.croatianname == croatianname)
        ).scalar()

        if created_word != None:
            db.session.delete(created_word)
            db.session.commit()
```

Slika 5.9: Kontekstna funkcija koja nakon testa briše stvorenu riječ

```
@pytest.fixture()
def study_session_setup_teardown(login_student, client):
    # Dodaj rječnik
    dictionary_data = {"dictionaryname": "Test dictionary"}

    dictionary = client.post("/api/dictionaries/1", json=dictionary_data)
    dictionaryid = dictionary.json["dictionaryid"]

    # Dodaj riječi
    words_data = [
        {
            "croatianname": "Cro 1",
            "foreignname": "Foreign 1",
            "audiopath": "Audio 1",
        },
        {
            "croatianname": "Cro 2",
            "foreignname": "Foreign 2",
            "audiopath": "Audio 2",
        },
        {
            "croatianname": "Cro 3",
            "foreignname": "Foreign 3",
            "audiopath": "Audio 3",
        },
    ]
    wordids = []
    for word in words_data:
        added_word = client.post("/api/words/1", json=word)
        wordids.append(added_word.json["wordid"])

    # Dodaj riječi u rječnik
    add_to_dict_input = {
        "dictionaryid": dictionaryid,
        "wordids": wordids,
    }

    client.post("/api/dictionaries/add-words", json=add_to_dict_input)

    # Pokreni test
    yield dictionaryid, wordids[0]

    # Izbriši rječnike
    client.delete(f"/api/dictionaries/{dictionaryid}")

    # Izbriši riječi
    for wordid in wordids:
        client.delete(f"/api/words/{wordid}")
```

Slika 5.10: Kontekstna funkcija za početak učenja

```
def test_start_study_session_unavailable(  
    login_student, study_session_setup_teardown, client  
):  
    dictionaryid, wordid = study_session_setup_teardown  
  
    resp = client.get(f"/api/words/choice/{dictionaryid}/{wordid}")  
  
    assert resp.status_code == 403
```

Slika 5.11: Započni učenje nad rječnikom

```
● (.venv) PS C:\Users\Duje\Desktop\fer\5. semestar\progi\olimplusplus\izvornikod\server> pytest -v --disable-warnings  
===== test session starts =====  
platform win32 -- Python 3.11.4, pytest-7.4.4, pluggy-1.3.0 -- C:\Users\Duje\Desktop\fer\5. semestar\progi\olimplusplus\izvo  
rnikod\server\.venv\Scripts\python.exe  
cachedir: .pytest_cache  
rootdir: C:\Users\Duje\Desktop\fer\5. semestar\progi\olimplusplus\izvornikod\server  
collected 6 items  
  
tests/test_dictionaries.py::test_create_dictionary_unauthorised PASSED [ 16%]  
tests/test_dictionaries.py::test_create_dictionary_empty_name PASSED [ 33%]  
tests/test_users.py::test_create_admin_missing_field PASSED [ 50%]  
tests/test_users.py::test_edit_password PASSED [ 66%]  
tests/test_words.py::test_create_word PASSED [ 83%]  
tests/test_words.py::test_start_study_session_unavailable PASSED [100%]  
  
===== 6 passed, 7 warnings in 3.16s =====
```

Slika 5.12: Dijagram razreda za slice Admin

### 5.2.2 Ispitivanje sustava

Ispitivanje sustava ostvareno je Selenium WebDriverom u Pythonu. Selenium WebDriver je alat za automatizaciju web preglednika. WebDriver koristi preglednikove vlastite mehanizme za kontrolu, čime osigurava visoku razinu realističnosti u testiranju, što ga čini ključnim alatom u razvoju softvera i osiguravanju kvalitete web aplikacija. Provedeno je pet testova, a njihov kod priložen je u nastavku. Rezultati testova prikazani su na Slici 5.18.

- Prvi ispitni slučaj obrađuje ispravnu prijavu. Očekivani izlaz je uspješna prijava i preusmjeravanje. (Slika 5.13)
- Drugi ispitni slučaj obrađuje neispravnu registraciju. Unosom maila u krivom formatu očekivani izlaz je neuspjela registracija. (Slika 5.14)
- Treći ispitni slučaj obrađuje uspješnu promjenu lozinke. Preduvjet za promjenu lozinke je uspješna prijava u sustav kao student. Unosom podudarajućih lozinki očekivani izlaz je izmjena lozinke i preusmjeravanje. (Slika 5.15)
- Četvrti ispitni slučaj koji obrađuje neispravno stvaranje admina. Preduvjet za stvaranje admina je uspješna prijava u sustav kao admin. Unosom nepodudarajućih lozinki u predložak, očekivani izlaz je neuspjela kreacija admina. (Slika 5.16)
- Peti ispitni slučaj obrađuje stvaranje rječnika s praznim imenom. Preduvjet za stvaranje rječnika je uspješna prijava u sustav kao admin. Bez unosa imena novog rječnika očekivani izlaz je nepromijenjeno stanje tablice rječnika u sustavu. (Slika 5.17)

```
# Ispravna prijava
def test1(email, password) -> bool:
    url = "http://localhost:3000/login"
    driver.get(url)
    driver.find_element(By.ID, "email").send_keys(email)
    driver.find_element(By.ID, "password").send_keys(password)
    driver.find_element(By.CSS_SELECTOR, "button[type='submit']").click()
    time.sleep(0.5)

    if driver.current_url.endswith("/login"):
        return False
    else:
        return True
```

Slika 5.13: Ispravna prijava

```
# Neispravna registracija
def test2() -> bool:
    url = "http://localhost:3000/login"
    driver.get(url)
    driver.find_element(By.CSS_SELECTOR, "button[type='button']").click()
    driver.find_element(By.ID, "firstname").send_keys("Osoba")
    driver.find_element(By.ID, "lastname").send_keys("Osoba")
    driver.find_element(By.ID, "email").send_keys("osoba.com")
    driver.find_element(By.CSS_SELECTOR, "button[type='submit']").click()

    if driver.current_url.endswith("/register"):
        return True
    else:
        return False
```

Slika 5.14: Neispravna registracija

```
# Uspješna promjena lozinke
def test3() -> bool:
    # Student login
    test1("ucenik@ucenik.com", "progi123")
    url = "http://localhost:3000/select-language/student"
    driver.get(url)
    # Open dropdown
    driver.find_element(By.XPATH, '//*[@id="root"]/nav/div/div/ul/li[1]/button').click()
    # Go to change password
    driver.find_element(By.XPATH, '/html/body/div[2]/div[3]/ul/ul/div[2]/li[2]').click()
    driver.find_element(By.ID, "newPassword").send_keys("progi123")
    driver.find_element(By.ID, "confirmNewPassword").send_keys("progi123")
    driver.find_element(By.CSS_SELECTOR, "button[type='submit']").click()
    time.sleep(0.5)

    if driver.current_url.endswith("/edit-password"):
        return False
    else:
        return True
```

Slika 5.15: Uspješna promjena lozinke

```
# Neispravno stvaranje admina
def test4() -> bool:
    # Admin login
    test1("admin@admin.com", "progi123")
    url = "http://localhost:3000/select-language/admin"
    driver.get(url)
    # Select language
    driver.find_element(By.XPATH, '//*[@id="root"]/div/div/div/button').click()
    # Admin-list
    driver.find_element(By.XPATH, '//*[@id="root"]/nav/div/div/ul/li[4]/div/a').click()
    # Go to add admin
    driver.find_element(By.XPATH, '//*[@id="root"]/div/div/div[1]/button').click()
    time.sleep(0.5)
    driver.find_element(By.ID, "firstname").send_keys("Admin")
    driver.find_element(By.ID, "lastname").send_keys("Admin")
    driver.find_element(By.ID, "email").send_keys("admin123@admin.com")
    driver.find_element(By.ID, "password").send_keys("admin123")
    driver.find_element(By.ID, "confirmPassword").send_keys("progi123")
    driver.find_element(By.CSS_SELECTOR, "button[type='submit']").click()
    time.sleep(0.5)

    if driver.current_url.endswith("/admin-list"):
        return True
    else:
        return False
```

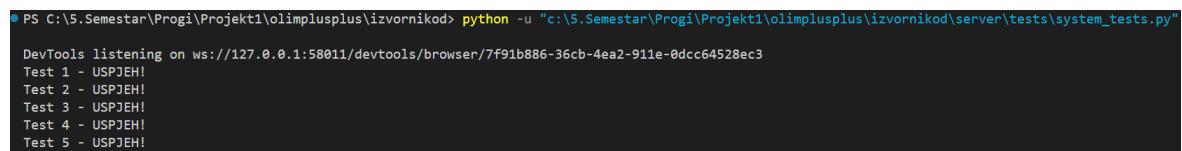
Slika 5.16: Neispravno stvaranje administratora

```
# Neuspješno stvaranje rječnika
def test5():
    # Admin login
    test1("admin@admin.com", "progi123")
    url = "http://localhost:3000/select-language/admin"
    driver.get(url)
    time.sleep(0.5)
    # Select language
    driver.find_element(By.XPATH, '//*[@id="root"]/div/div/div/button').click()
    # Dictionaries
    driver.find_element(By.XPATH, '//*[@id="root"]/nav/div/div/ul/li[3]/div/a').click()
    dict_table = driver.find_element(By.XPATH, '//*[@id="root"]/div/div[2]/table/tbody')
    dict_table_before = dict_table.find_elements(By.CSS_SELECTOR, """#root > div > div
                                > div.MuiPaper-root.MuiPaper-elevation
                                .MuiPaper-rounded.MuiPaper-elevation1
                                .sc-jSFipO.hkiwdC.css-1ps6pg7-MuiPaper-root
                                > table > tbody > tr""")
    driver.find_element(By.XPATH, '//*[@id="root"]/div/div/div[1]/button').click()

    driver.find_element(By.CSS_SELECTOR, "button[type='submit']").click()
    time.sleep(0.5)
    dict_table_after = dict_table.find_elements(By.CSS_SELECTOR, """#root > div > div
                                > div.MuiPaper-root.MuiPaper-elevation
                                .MuiPaper-rounded.MuiPaper-elevation1
                                .sc-jSFipO.hkiwdC.css-1ps6pg7-MuiPaper-root
                                > table > tbody > tr""")

    if len(dict_table_before) != len(dict_table_after):
        return False
    else:
        return True
```

Slika 5.17: Stvaranje rječnika s praznim imenom

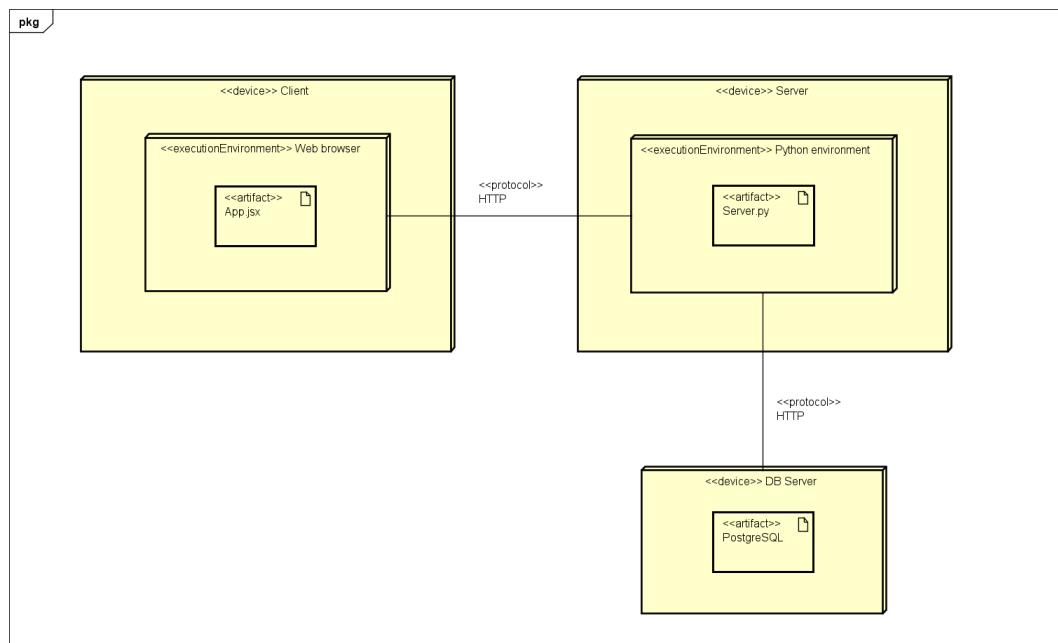
A screenshot of a terminal window titled "PS C:\5.Semestar\Progi\Projekt1\olimplusplus\izvornikod". The command "python -u "c:\5.Semestar\Progi\Projekt1\olimplusplus\izvornikod\server\tests\system\_tests.py"" was run. The output shows the DevTools listening on ws://127.0.0.1:58011/devtools/browser/7f91b886-36cb-4ea2-911e-0dcc64528ec3 message, followed by five "Test 1 - USPJEH!" messages.

```
PS C:\5.Semestar\Progi\Projekt1\olimplusplus\izvornikod> python -u "c:\5.Semestar\Progi\Projekt1\olimplusplus\izvornikod\server\tests\system_tests.py"
DevTools listening on ws://127.0.0.1:58011/devtools/browser/7f91b886-36cb-4ea2-911e-0dcc64528ec3
Test 1 - USPJEH!
Test 2 - USPJEH!
Test 3 - USPJEH!
Test 4 - USPJEH!
Test 5 - USPJEH!
```

Slika 5.18: Terminal nakon pokretanja testova ispitivanja sustava

### 5.3 Dijagram razmještaja

Aplikacija je organizirana u tri razine: klijentsku, serversku i podatkovnu. Klijentska razina izvršava se u web pregledniku koji pokreće React aplikaciju. Serverska razina pokreće se na odvojenom poslužitelju u Python 3.11.7 okruženju koji komunicira s PostgreSQL 15 bazom podataka puštenom u pogon na trećem računalu. Komunikacija između izvršnih okruženja odvija se protokolom HTTP.



Slika 5.19: Dijagram razmještaja

## 5.4 Upute za puštanje u pogon

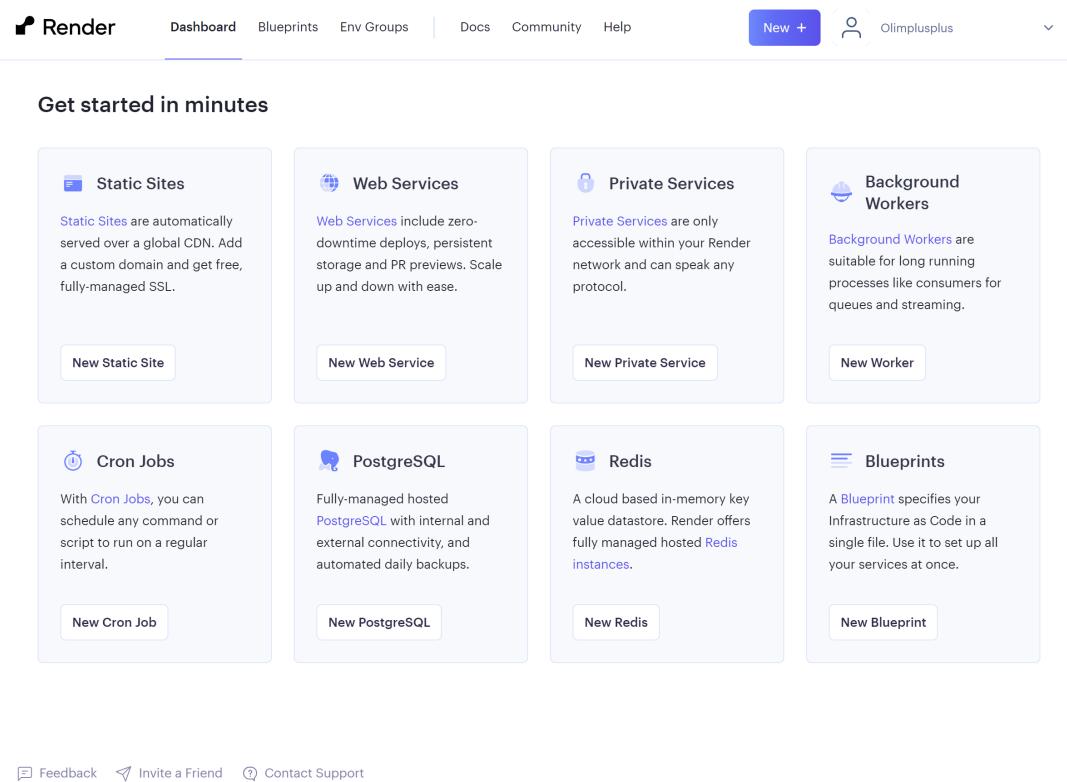
Aplikaciju puštamo u pogon preko usluge Render koju povezujemo s repozitorijem na GitHubu. Ovakva konfiguracija omogućava nam da jednom postavljenu aplikaciju možemo automatski ponovno pustiti u pogon pushem na granu koju smo odabrali kao produkcijsku. Puštanje u pogon odvija se u tri faze.

### 5.4.1 Baza podataka

- Nakon registracije i prijave u servis Render odabiremo gumb *New PostgreSQL* (Slika 5.20)
- Bazu podataka konfiguriramo prema Slici 5.21 i potvrdimo unos
- Nakon uspješnog puštanja u pogon, iz postavki kopiramo vanjski URL za pristup bazi (Slika 5.22)
- Inicijalizaciju baze provodimo pomoću biblioteke Flask-Migrate. URL za pristup bazi pohranimo lokalno u varijablu okruženja na serverskom dijelu aplikacije. Izvršavanjem naredbe `flask –app server db upgrade` u lokalnom okruženju izvršavaju se sve stvorene *up* migracije i u produkcijskoj bazi stvaraju se sve potrebne tablice.

### 5.4.2 Poslužiteljski dio aplikacije

- Poslužiteljsku stranu puštamo u pogon kao web servis. Nakon odabira opcije *Web Service* iz izbornika *New* odabiremo opciju puštanja u pogon s GitHub repozitorija (Slika 5.23).
- Render povežemo s GitHub računom (Slika 5.24). Nakon uspješnog povezivanja moći ćete odabrati repozitorij kojeg želite pustiti u pogon (Slika 5.25).
- Web servis konfiguriramo prema slici Slika 5.26 i potvrdimo unos.
- U postavkama stvorenog servisa pod odjeljkom *Environments* trebamo stvoriti novu datoteku u koju ćemo spremiti sve potrebne varijable okruženja za pokretanje aplikacije, uključujući i URL za pristup bazi podataka (Slika 5.27).



[Feedback](#) [Invite a Friend](#) [Contact Support](#)

Slika 5.20: Dashboard zaslon nakon uspješne registracije

### 5.4.3 Klijentski dio aplikacije

- Prije puštanja fontenda u pogon potrebno je proširiti datoteku *tsconfig.json* prema Slici 5.28 i promjene pushati na git.
- Ovaj dio aplikacije također puštamo u pogon kao web servis i konfiguriramo ga prema Slici 5.29 .

Nakon navedenih koraka sve tri razine aplikacije trebale bi biti uspješno puštene u pogon (Slika 5.30).

New PostgreSQL

Name: Progi db

Database: olimplusplus

User: flipmemo

Region: Frankfurt (EU Central)

PostgreSQL Version: 15

Datadog API Key: (Optional)

Instance Type

For hobby projects:

<b>Free</b> \$0 / month	256 MB (RAM) 0.1 CPU 1 GB (Storage)
----------------------------	-------------------------------------------

For professional use:

<b>Starter</b> \$7 / month	256 MB (RAM) 0.1 CPU 1 GB (Storage)
<b>Standard</b> \$20 / month	1 GB (RAM) 1 CPU 16 GB (Storage)
<b>Pro</b> \$95 / month	4 GB (RAM) 2 CPU 96 GB (Storage)
<b>Pro Plus</b> \$185 / month	8 GB (RAM) 4 CPU 256 GB (Storage)

Upgrades available for Pro and Pro Plus instances.

Need a [custom instance type](#)? We support up to 512 GB RAM, 64 CPUs, and 5 TB storage.

Access more features like [Point-in-Time Recovery](#) and [High Availability](#) by upgrading to a team plan.

Create a team

Create Database

Feedback | Invite a Friend | Contact Support

Slika 5.21: Konfiguracija baze podataka

The screenshot shows the Render interface with the 'Connections' section selected. It displays the following database configuration:

- Hostname: dpg-cml9mjn109ks73a85ck0-a
- Port: 5432
- Database: olimplusplus
- Username: flipmemo
- Password: [REDACTED]
- Internal Database URL: [REDACTED]
- External Database URL: [REDACTED]
- PSQL Command: [REDACTED]

Below the connections section is an 'Access Control' section indicating that 1 IP range is allowed from outside of your private network.

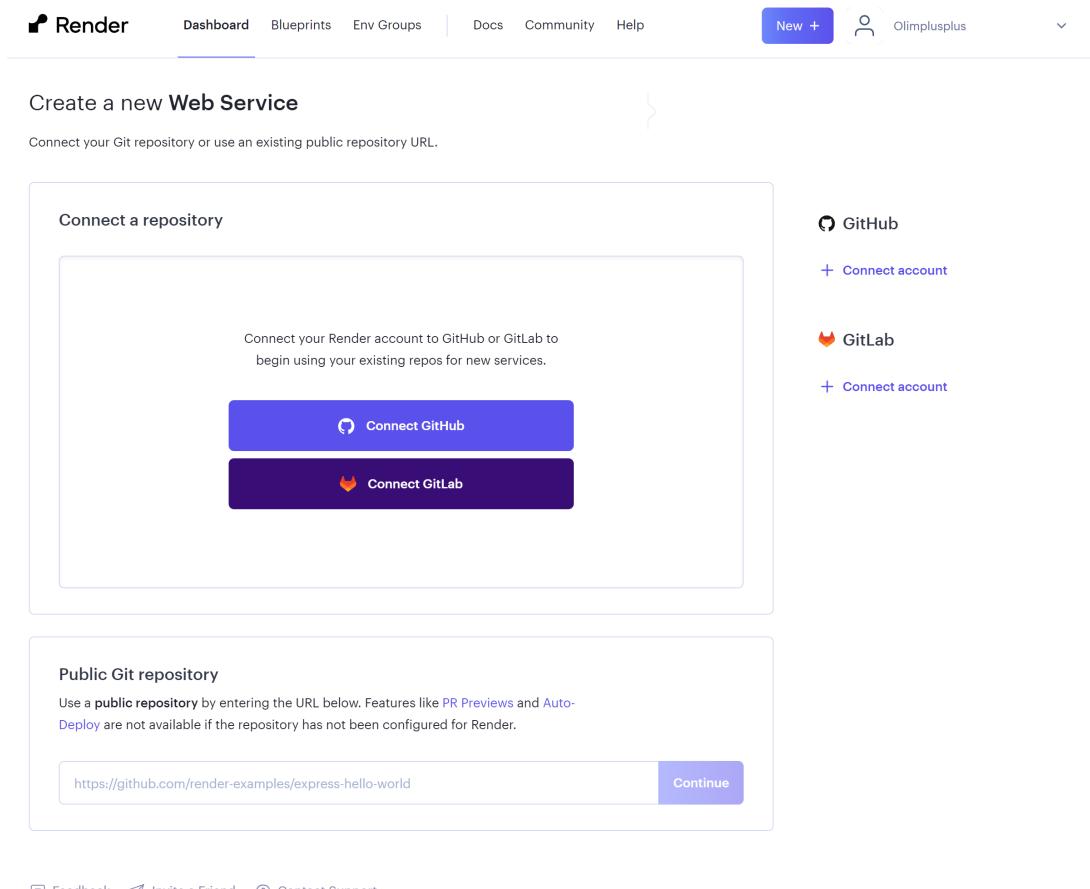
Slika 5.22: Podaci za pristup bazi podataka

The screenshot shows the 'Create a new Web Service' page. It asks how you would like to deploy your web service, with two options:

- Build and deploy from a Git repository  
Connect a GitHub or GitLab repository.
- Deploy an existing image from a registry ADVANCED  
Pull a public image from any registry or a private image from Docker Hub, GitHub, or GitLab.

A 'Next' button is visible at the bottom right.

Slika 5.23: Stvaranje web servisa



Slika 5.24: Sučelje bez povezanog GitHub računa

The screenshot shows the Render web interface for creating a new Web Service. At the top, there is a navigation bar with links for Dashboard, Blueprints, Env Groups, Docs, Community, Help, a New button, and a user profile for Olimplusplus. Below the navigation bar, a section titled "Create a new Web Service" asks to "Connect your Git repository or use an existing public repository URL." On the left, a "Connect a repository" panel features a search bar and a list of repositories. It shows a recent connection from "DujeStolfa / olimplusplus" made 14 hours ago, with a "Connect" button. To the right, sections for GitHub and GitLab are displayed, each with a "Configure account" link and a "Connect account" link. At the bottom of the main area, a "Public Git repository" section provides instructions for entering a URL and includes a "Continue" button. At the very bottom of the page, there are links for Feedback, Invite a Friend, and Contact Support.

Slika 5.25: Uspješno povezan GitHub račun

You are deploying a web service for [DujeStolfa/olimplusplus](#).

**Name**  
A unique name for your web service.  
Progi server

**Region**  
The region where your web service runs. Services must be in the same region to communicate privately and you currently have services running in Frankfurt.  
Frankfurt (EU Central)

**Branch**  
The repository branch used for your web service.  
main

**Root Directory** Optional  
Defaults to repository root. When you specify a root directory that is different from your repository root, Render runs all your commands in the specified directory and ignores changes outside the directory.  
izvornikod/server

**Runtime**  
The runtime for your web service.  
Python 3

**Build Command**  
This command runs in the root directory of your repository when a new version of your code is pushed, or when you deploy manually. It is typically a script that installs libraries, runs migrations, or compiles resources needed by your app.  
izvornikod/server/ \$ pip install -r requirements.txt

**Start Command**  
This command runs in the root directory of your app and is responsible for starting its processes. It is typically used to start a webserver for your app. It can access environment variables defined by you in Render.  
izvornikod/server/ \$ gunicorn server:app

**Instance Type**

For hobby projects	Free \$0 / month	512 MB (RAM) 0.1 CPU	⚠️ Upgrade to enable more features Free instances spin down after periods of inactivity. They do not support SSH access, scaling, one-off jobs, or persistent disks. Select any paid instance type to enable these features.
<b>For professional use</b> For more power and to get the most out of Render, we recommend using one of our paid instance types. All paid instances support:	<b>Starter</b> \$7 / month	512 MB (RAM) 0.5 CPU	<b>Standard</b> \$25 / month 2 GB (RAM) 1 CPU
<ul style="list-style-type: none"> <li>• Zero Downtime</li> <li>• SSH Access</li> <li>• Scaling</li> <li>• One-off jobs</li> <li>• Support for persistent disks</li> </ul>	<b>Pro</b> \$85 / month	4 GB (RAM) 2 CPU	<b>Pro Plus</b> \$175 / month 8 GB (RAM) 4 CPU
	<b>Pro Max</b> \$225 / month	16 GB (RAM) 4 CPU	<b>Pro Ultra</b> \$450 / month 32 GB (RAM) 8 CPU

Need a [custom instance type](#)? We support up to 512 GB RAM and 64 CPUs.

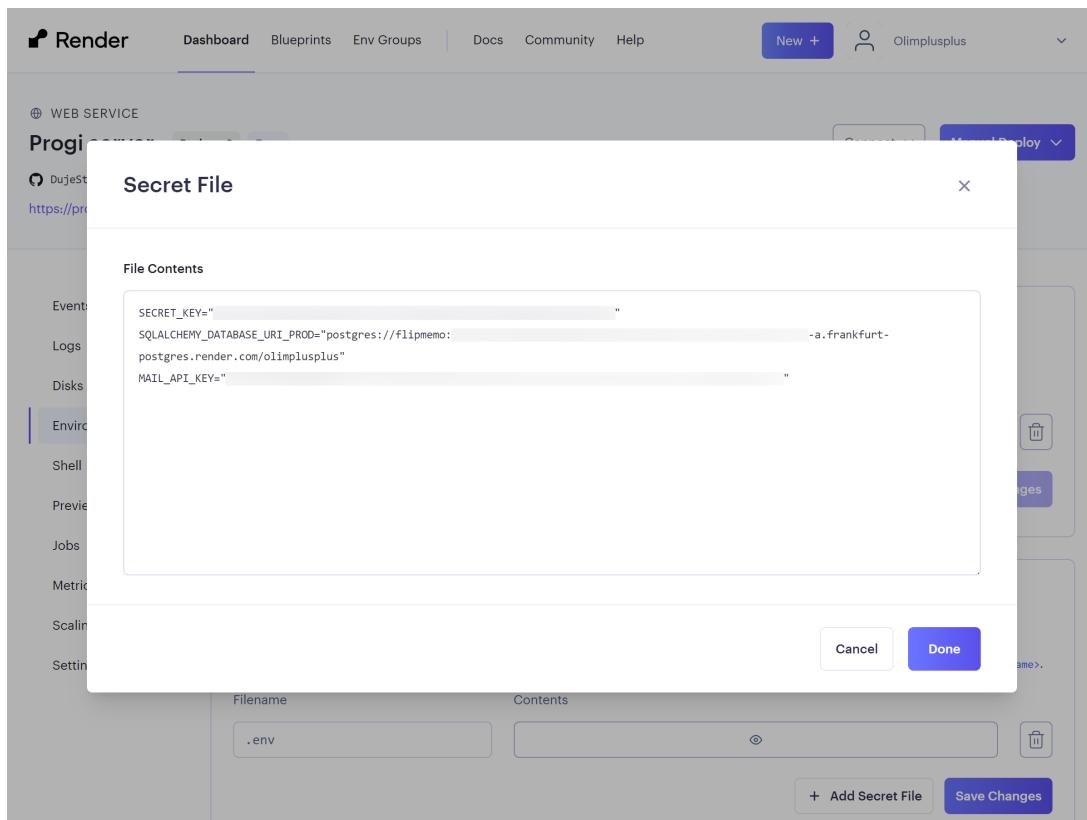
**Environment Variables** Optional  
Set environment-specific config and secrets (such as API keys), then read those values from your code. [Learn more](#).

**Advanced**

**Create Web Service**

[Feedback](#) [Invite a Friend](#) [Contact Support](#)

Slika 5.26: Konfiguracija serverskog dijela aplikacije



Slika 5.27: Dodavanje varijabli okruženja na backend

```
"scripts": {
  "start": "react-scripts start",
  "start-prod": "serve -s build",
  "build": "npm install && react-scripts build",
  "test": "react-scripts test",
  "eject": "react-scripts eject"
},
"engines": {
  "node": ">=18.18.0 <19.0.0"
},
```

Slika 5.28: Producčijska konfiguracija React aplikacije

**Render** Dashboard Blueprints Env Groups Docs Community Help New + Olimplusplus

You are deploying a web service for [DujeStolfa/olimplusplus](#).

Name	Progi client												
Region	Frankfurt (EU Central)												
Branch	main												
Root Directory	Optional Defaults to repository root. When you specify a <a href="#">root directory</a> that is different from your repository root, Render runs all your commands in the <a href="#">specified directory</a> and ignores changes outside the directory.												
Runtime	Node												
Build Command	izvornikod/client/ \$ yarn build												
Start Command	izvornikod/client/ \$ yarn start-prod												
<b>Instance Type</b>													
For hobby projects	<table border="1"> <tbody> <tr> <td><b>Free</b> \$0 / month</td> <td>512 MB (RAM) 0.1 CPU</td> <td><b>Upgrade to enable more features</b> Free instances spin down after periods of inactivity. They do not support SSH access, scaling, one-off jobs, or persistent disks. Select any paid instance type to enable these features.</td> </tr> <tr> <td><b>Starter</b> \$7 / month</td> <td>512 MB (RAM) 0.5 CPU</td> <td><b>Standard</b> \$25 / month 2 GB (RAM) 1 CPU</td> </tr> <tr> <td><b>Pro</b> \$85 / month</td> <td>4 GB (RAM) 2 CPU</td> <td><b>Pro Plus</b> \$175 / month 8 GB (RAM) 4 CPU</td> </tr> <tr> <td><b>Pro Max</b> \$225 / month</td> <td>16 GB (RAM) 4 CPU</td> <td><b>Pro Ultra</b> \$450 / month 32 GB (RAM) 8 CPU</td> </tr> </tbody> </table>	<b>Free</b> \$0 / month	512 MB (RAM) 0.1 CPU	<b>Upgrade to enable more features</b> Free instances spin down after periods of inactivity. They do not support SSH access, scaling, one-off jobs, or persistent disks. Select any paid instance type to enable these features.	<b>Starter</b> \$7 / month	512 MB (RAM) 0.5 CPU	<b>Standard</b> \$25 / month 2 GB (RAM) 1 CPU	<b>Pro</b> \$85 / month	4 GB (RAM) 2 CPU	<b>Pro Plus</b> \$175 / month 8 GB (RAM) 4 CPU	<b>Pro Max</b> \$225 / month	16 GB (RAM) 4 CPU	<b>Pro Ultra</b> \$450 / month 32 GB (RAM) 8 CPU
<b>Free</b> \$0 / month	512 MB (RAM) 0.1 CPU	<b>Upgrade to enable more features</b> Free instances spin down after periods of inactivity. They do not support SSH access, scaling, one-off jobs, or persistent disks. Select any paid instance type to enable these features.											
<b>Starter</b> \$7 / month	512 MB (RAM) 0.5 CPU	<b>Standard</b> \$25 / month 2 GB (RAM) 1 CPU											
<b>Pro</b> \$85 / month	4 GB (RAM) 2 CPU	<b>Pro Plus</b> \$175 / month 8 GB (RAM) 4 CPU											
<b>Pro Max</b> \$225 / month	16 GB (RAM) 4 CPU	<b>Pro Ultra</b> \$450 / month 32 GB (RAM) 8 CPU											
For professional use	<p>For more power and to get the most out of Render, we recommend using one of our paid instance types. All paid instances support:</p> <ul style="list-style-type: none"> <li>• Zero Downtime</li> <li>• SSH Access</li> <li>• Scaling</li> <li>• One-off jobs</li> <li>• Support for persistent disks</li> </ul>												
Need a <a href="#">custom instance type</a> ? We support up to 512 GB RAM and 64 CPUs.													
Environment Variables	Optional Set environment-specific config and secrets (such as API keys), then read those values from your code. <a href="#">Learn more</a> .												
<input type="text" value="NAME_OF_VARIABLE"/> <input type="text" value="value"/> <input type="button" value="Generate"/> <input type="button" value="Delete"/> <a href="#">+ Add Environment Variable</a>													
<input type="button" value="Advanced"/>													
<input type="button" value="Create Web Service"/>													

[Feedback](#) [Invite a Friend](#) [Contact Support](#)

Slika 5.29: Konfiguracija web servisa za frontend

The screenshot shows the Render Dashboard. At the top, there is a navigation bar with links for Dashboard, Blueprints, Env Groups, Docs, Community, Help, and a New + button. A user profile for 'Olimplusplus' is also visible. Below the navigation is a search bar labeled 'Search services'. Underneath is a table titled 'Overview' showing three active services:

Service Name	Status	Type	Runtime	Region	Last Deployed	Actions
Progi client	Deployed	Web Service	Node	Frankfurt	a few seconds ago	...
Progi server	Deployed	Web Service	Python 3	Frankfurt	6 minutes ago	...
Progi db	Available	PostgreSQL	PostgreSQL 15	Frankfurt	21 minutes ago	...

At the bottom of the dashboard, there are links for Feedback, Invite a Friend, and Contact Support.

Slika 5.30: Aktivne aplikacije u Renderu

## 6. Zaključak i budući rad

Razvoj projekta je bilo važno iskustvo za sve članove tima. Dobili smo uvid u alate za komunikaciju ideja, kao što su stečene i vještine sinkronizacije sa drugim članovima. Naučeno je puno o konkretnom razvoju aplikacija na webu, kao i o Reactu. Vrijeme razvoja je bilo definirano količinama znanja individualnih članova tima, te da ponovno pravimo isti projekt sa svime naučenim, išlo bi brže.

Puno vremena je bilo uloženo u idejnu razradu aplikacije, te smo se posvetili detaljima funkcionalnih zahtjeva i razrade svih obrazaca uporabe prije nego što smo krenuli programirati, što je u konačnici uštedilo puno vremena. Nismo imali dvoumljenja oko implementacije, samo smo mogli pisati po već razrađenim obrascima.

Za bržu i kvalitetniju izradu projekta bi samo bilo potrebno da svi članovi imaju istu količinu znanja vezanu za odabrani radni okvir, kao i sličnu količinu iskustva u izradi web aplikacija. Puno stvari u razvoju na webu se ponavlja, projekti imaju slične strukture datoteka, koriste se isti programski obrasci kao rješenje dobro poznatih problema. Izrada baze podataka je donekle standardiziran postupak, povezivanje na istu i povlačenje podataka se može napraviti na više načina. Sve se svodi na odabir nekih od svih dostupnih tehnologija, te iskustvo s radom u istim. Korisničko sučelje se može napraviti relativno brzo uz pomoć biblioteka i radnih okvira. Puno gotovih rješenja za web već postoji, vještina ih je znati spojiti i znati kako te biblioteke međusobno komuniciraju.

Glede perspektiva za nastavak rada u projektnoj grupi, bilo kakav budući rad može biti samo lakši, budući da je tim prošao kroz svoje ranije faze, te svi članovi lakše znaju komunicirati jedni s drugima i uskočiti tamo gdje treba. Svi su upoznati s vlastitim snagama i slabostima.

Sve tražene funkcionalnosti projektnog zadatka su implementirane i razrađene.

# Popis literature

1. Programsko inženjerstvo, FER ZEMRIS, <http://www.fer.hr/predmet/proinzh>
2. I. Sommerville, "Software engineering", 8th ed, Addison Wesley, 2007.
3. S.R. Schach, "Object-Oriented and Classical Software Engineering", 8th ed. McGraw-Hill, 2010.
4. G. Booch, I. Jacobson, and J. Rumbaugh, "The Unified Modeling Language User Guide", 2nd ed., Pearson Education, 2005.
5. Astah Community, <http://astah.net/editions/uml-new>
6. N. Frid, A. Jović "Modeliranje programske potpore UML-dijagramima", Sveučilište u Zagrebu, 2023.

# Indeks slika i dijagrama

2.1 Ilustracija sustava za učenje s pet posuda . . . . .	5
2.2 Špilovi u Ankiju pandan su našim rječnicima . . . . .	9
3.1 Funkcionalnosti učenja i upravljanja riječima . . . . .	22
3.2 Funkcionalnosti upravljanja računima . . . . .	23
3.3 Funkcionalnosti upravljanja jezicima i rječnicima . . . . .	23
3.4 Sekvencijski dijagram, UC1 Registracija . . . . .	25
3.5 Sekvencijski dijagram, UC8 Pregledavanje i odabir rječnika . . . . .	27
3.6 Sekvencijski dijagram, UC10 Promjena sadržaja rječnika . . . . .	29
4.1 Relacijska shema baze podataka . . . . .	36
4.2 ER model baze podataka . . . . .	37
4.3 Dijagram razreda za backend dio aplikacije . . . . .	39
4.4 Dijagram razreda za slice Admin . . . . .	41
4.5 Dijagram razreda za slice Auth . . . . .	41
4.6 Dijagram razreda za slice Dictionaries . . . . .	42
4.7 Dijagram razreda za slice Language . . . . .	42
4.8 Dijagram razreda za slice Student Dictionaries . . . . .	42
4.9 Dijagram razreda za slice Study Session . . . . .	43
4.10 Dijagram razreda za slice Words . . . . .	44
4.11 Dijagram stanja - sučelje za učenike . . . . .	45
4.12 Dijagram stanja - sučelje za administratore . . . . .	46
4.13 Dijagram aktivnosti za učenje sa snimanjem izgovora . . . . .	47
4.14 Dijagram komponenti . . . . .	48
5.1 Kontekstne funkcije za autentifikaciju korisnika . . . . .	53
5.2 Osnovne kontekstne funkcije . . . . .	54
5.3 Stvaranje rječnika s praznim nazivom . . . . .	54
5.4 Stvaranje rječnika bez prethodne prijave . . . . .	54
5.5 Stvaranje administratora s nepotpunim podacima . . . . .	55
5.6 Promjena lozinke učenika . . . . .	55

5.7 Kontekstna funkcija za poništavanje promjene lozinke . . . . .	56
5.8 Stvaranje riječi . . . . .	57
5.9 Kontekstna funkcija koja nakon testa briše stvorenu riječ . . . . .	57
5.10 Kontekstna funkcija za početak učenja . . . . .	58
5.11 Započni učenje nad rječnikom . . . . .	59
5.12 Dijagram razreda za slice Admin . . . . .	59
5.13 Ispravna prijava . . . . .	60
5.14 Neispravna registracija . . . . .	61
5.15 Uspješna promjena lozinke . . . . .	61
5.16 Neispravno stvaranje administratora . . . . .	62
5.17 Stvaranje rječnika s praznim imenom . . . . .	62
5.18 Terminal nakon pokretanja testova ispitivanja sustava . . . . .	63
5.19 Dijagram razmještaja . . . . .	64
5.20 Dashboard zaslon nakon uspješne registracije . . . . .	66
5.21 Konfiguracija baze podataka . . . . .	67
5.22 Podaci za pristup bazi podataka . . . . .	68
5.23 Stvaranje web servisa . . . . .	68
5.24 Sučelje bez povezanog GitHub računa . . . . .	69
5.25 Uspješno povezan GitHub račun . . . . .	70
5.26 Konfiguracija serverskog dijela aplikacije . . . . .	71
5.27 Dodavanje varijabli okruženja na backend . . . . .	72
5.28 Producjska konfiguracija React aplikacije . . . . .	72
5.29 Konfiguracija web servisa za frontend . . . . .	73
5.30 Aktivne aplikacije u Renderu . . . . .	74

# Dodatak: Prikaz aktivnosti grupe

## Dnevnik sastajanja

### 1. sastanak

- Datum: 15. listopada 2023.
- Prisustvovali: Cijela grupa
- Teme sastanka:
  - rasprava o funkcijskim zahtjevima
  - razrješavanje nedoumica oko teksta zadatka

### 2. sastanak

- Datum: 25. listopada 2023.
- Prisustvovali: N.Bulić, D.Štolfa, I.Žilić
- Teme sastanka:
  - popis obrazaca uporabe
  - razrada aktora, dionika i funkcijskih zahtjeva

### 3. sastanak

- Datum: 30. listopada 2023.
- Prisustvovali: Cijela grupa
- Teme sastanka:
  - Git i LaTeX instalacija, postavljanje okruženja

### 4. sastanak

- Datum: 6. studenog 2023.
- Prisustvovali: N.Brala, K.Kuzle, G.Čobanov, F.Kuzmanić
- Teme sastanka:
  - dijagrami obrazaca uporabe
  - dijagram baze podataka

### 5. sastanak

- Datum: 9. prosinca 2023.
- Prisustvovali: Cijela grupa
- Teme sastanka:

- analiza izvornog koda prve revizije projekta
- rješavanje zadataka iz dev odjeljka u README datoteci
- podjela početnih zadataka

#### 6. sastanak

- Datum: 18. prosinca 2023.
- Prisustvovali: N.Bulić, D.Štolfa
- Teme sastanka:
  - zaslon za promjenu lozinke
  - struktura serverskog dijela aplikacije

#### 7. sastanak

- Datum: 21. prosinca 2023.
- Prisustvovali: Cijela grupa
- Teme sastanka:
  - recap o tijeku zadataka
  - uklanjanje pogrešaka
  - podjela novih zadataka

#### 8. sastanak

- Datum: 8. siječnja 2024.
- Prisustvovali: N.Brala, D.Štolfa
- Teme sastanka:
  - plan razrade dijagrama
  - početak izrade dijagrama komponenti
  - analiza arhitekture sustava

#### 9. sastanak

- Datum: 12. siječnja 2024.
- Prisustvovali: N. Bulić, D. Štolfa
- Teme sastanka:
  - zaslon za upravljanje rječnicima
  - analiza klijentske strane aplikacije
  - Redux mehanizmi

#### 10. sastanak

- Datum: 15. siječnja 2024.
- Prisustvovali: N.Brala, I.Žilić
- Teme sastanka:
  - dijagrami razreda

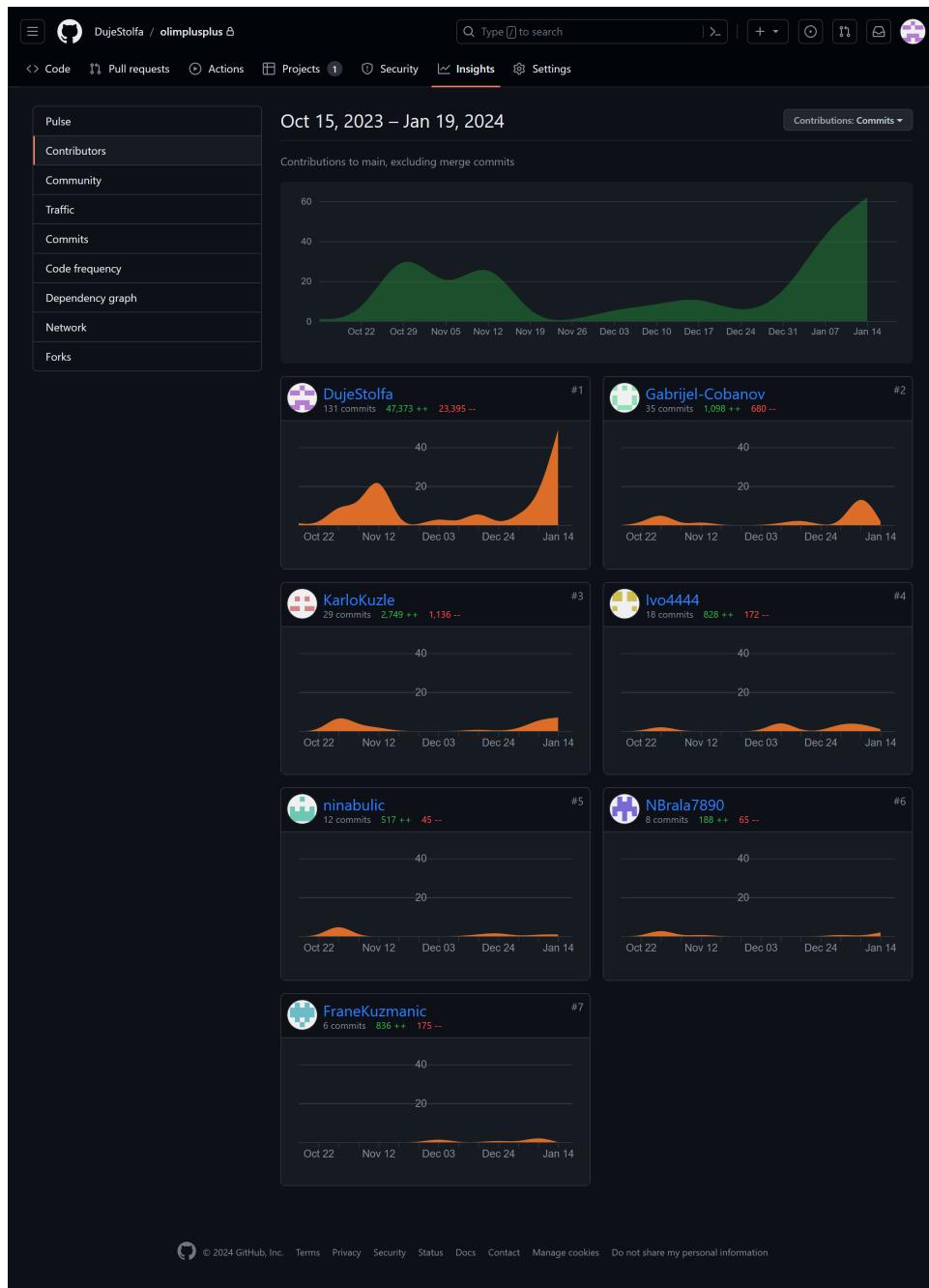
## 11. sastanak

- Datum: 17. siječnja 2024.
- Prisustvovali: N.Brala, K.Kuzle, D.Štolfa, G.Čobanov
- Teme sastanka:
  - ispitivanje sustava
  - dijagrami razreda
  - provjera izrađene dokumentacije

## Tablica aktivnosti

	Duje Štolfa	Karlo Kuzle	Ivo Žilić	Frane Kuzmanić	Nina Bulić	Gabrijel Čobanov	Nikša Brala
Upravljanje projektom							
Opis projektnog zadatka							
Funkcionalni zahtjevi							
Opis pojedinih obrazaca							
Dijagram obrazaca							
Sekvencijski dijagrami							
Opis ostalih zahtjeva							
Arhitektura i dizajn sustava							
Baza podataka							
Dijagram razreda							
Dijagram stanja							
Dijagram aktivnosti							
Dijagram komponenti							
Korištene tehnologije i alati							
Ispitivanje programskog rješenja							
Dijagram razmještaja							
Upute za puštanje u pogon							
Dnevnik sastajanja							
Zaključak i budući rad							
Popis literature							

# Dijagrami pregleda promjena



*Napomena: Doprinosi s devdoc grane squashani su u jedan commit prilikom spajanja u main granu, pa isti nisu vidljivi u dijagramu, već samo na devdoc grani na GitHubu.*