Programsko inženjerstvo ak.god 2024./2025.

Triolingo

Grupa: <G17.1>

Tim: Petrovi

Nastavnik: Vlado Sruk

Cilj projektnog zadatka

Projekt *Triolingo* ima za cilj razviti fleksibilnu, online platformu koja povezuje učenike i učitelje jezika. Glavni cilj je omogućiti dostupno i personalizirano učenje jezika, prilagođeno potrebama korisnika. U današnjem globaliziranom svijetu, poznavanje stranih jezika postalo je ključna vještina, a *Triolingo* olakšava pristup učenju kroz centralizirani sustav pretrage, komunikacije i zakazivanja lekcija.

Problematika zadatka

Tradicionalni pristupi učenju jezika, kao što su škole jezika, često su skupi i zahtijevaju fiksni raspored koji ne odgovara svakom korisniku. Pronalaženje odgovarajućeg učitelja predstavlja dodatni izazov zbog ograničenih opcija i nedostatka transparentnosti u kvalifikacijama i metodama poučavanja. Uvođenjem online platforme, *Triolingo* omogućava:

- Pristup velikom broju kvalificiranih učitelja
- Fleksibilnost u odabiru vremena i stila učenja
- Transparentne informacije o učiteljima (godine iskustva, kvalifikacije, ocjene i komentari učenika, satnica koju naplaćuju)
- Lakšu komunikaciju i postavljanje ciljeva učenja prema potrebama učenika.

Potencijalna korist projekta

- **Učenici**: Omogućuje pristup učenju jezika u različitim formatima (privatne lekcije, grupne sesije), prilagođenih njihovim potrebama u terminima koji im odgovaraju.
- **Učitelji**: Imaju mogućnost proširiti svoj doseg i povećati prihode kroz platformu koja im omogućuje fleksibilno zakazivanje i komunikaciju s učenicima. Također uključuje feedback sustav koji učiteljima pomaže u poboljšanju njihovih metoda podučavanja.

- **Šira zajednica**: Pristupačno učenje jezika doprinosi profesionalnom razvoju, kulturnoj raznolikosti i olakšava međukulturalno razumijevanje, što je korisno u poslovnim i osobnim međunarodnim interakcijama.

Postojeća slična rješenja

Neka od često korištenih rješenja, tj. aplikacija, su:

- **iTalki**: Omogućava korisnicima pronalaženje učitelja jezika, ali nedostaju personalizirane opcije filtriranja prema ciljevima učenja.
- **Verbling**: Platforma je slična, no manje je fleksibilna u pogledu kalendara i zakazivanja termina prema preferencijama učenika.
- **Duolingo**: Fokusiran je na samostalno učenje jezika putem aplikacije, no ne nudi pristup stvarnim učiteljima ili individualne lekcije.

Razlike: *Triolingo* se ističe personaliziranim pristupom koji omogućuje učenicima individualno prilagođene lekcije s pravim učiteljima. Platforma stavlja naglasak na fleksibilnost rasporeda i detaljno filtriranje učitelja prema specifičnim jezičnim i obrazovnim potrebama učenika, pružajući korisničko iskustvo koje premašuje mogućnosti trenutačno dostupnih aplikacija.

Skup korisnika

Potencijalni korisnici *Triolingo* platforme uključuju:

- **Profesionalci**: Koriste platformu za poslovnu komunikaciju ili pripremu za međunarodne projekte.
- **Studenti i učenici**: Uče jezik za akademske potrebe ili pripremu za putovanja.
- **Imigranti**: Osobe koje se prilagođavaju životu u stranoj zemlji, koristeći jezik kao alat za bolju integraciju.
- **Hobisti**: Osobe koje uče nove jezike iz osobne želje za proširenjem znanja i upoznavanja novih kultura.

Mogućnost prilagodbe rješenja

- **Dodavanje novih jezika**: Platforma je konstruirana tako da se lako može nadograditi s novim jezicima.

- **Uvođenje novih funkcionalnosti**: Modularan dizajn omogućava integraciju dodatnih značajki, poput grupnih sesija, gamifikacije (bodovni sustavi) ili virtualne učionice s mogućnošću video poziva.
- **Prilagodba stila učenja**: Učitelji mogu odabrati između različitih metoda podučavanja koje se mogu prilagoditi specifičnim potrebama svakog učenika, poput konverzacijskih lekcija ili priprema za ispite.

Opseg projektnog zadatka

Projekt uključuje sljedeće funkcionalnosti koje obogaćuju korisničko iskustvo:

- Kreiranje korisničkog profila za učenike, učitelje i sistemske administratore.
- Pretraga i filtriranje učitelja prema različitim kriterijima.
- Zakazivanje i pregled lekcija u kalendaru, te arhiviranje lekcija nakon što su završene.
- Ugrađeni chat omogućuje izravnu komunikaciju između učenika i učitelja.
- Feedback sustav omogućava učenicima da ocjenjuju učitelje i pružaju povratne informacije nakon lekcija.
- Administracijski sustav za upravljanje korisnicima i nadzor aktivnosti na platformi.

Moguće nadogradnje projektnog zadatka

- **Videokonferencijske lekcije**: Integracija video poziva direktno unutar platforme.
- **Gamifikacija učenja**: Uvođenje bodova i nagrada za učenike, kako bi motivirali na kontinuirano učenje.
- **Napredni sustav preporuka**: Korištenje jednostavnih algoritama za preporuke učitelja/lekcija temeljenih na povijesti korisnika.
- **Multijezičnost sučelja**: Podrška za više jezika unutar korisničkog sučelja, uključujući automatsku detekciju jezika korisnika.

Funkcionalni zahtjevi

ID zahtjeva	**Opis**	**Prioritet**	**Izvor**	**Kriteriji prihvaćanja**
F-001	Sustav omogućuje korisnicima registraciju s pomoću e-mail adrese i lozinke.	Visok	Zahtjev dionika	Korisnik se može registrirati putem e-maila, primiti e-mail za potvrdu i uspješno se prijaviti.

ID zahtjeva	Opis	Prioritet	Izvor	Kriteriji prihvaćanja
F-002	Sustav omogućuje korisnicima prijavu s e-mail adresom, lozinkom ili putem OAuth usluge.	Visok	Zahtjev dionika	Korisnik se može prijaviti unoseći ispravne podatke ili koristeći OAuth servis te pristupiti svom računu.
F-003	Sustav omogućuje korisnicima oporavak lozinke putem e-mail adrese.	Srednji	Zahtjev dionika	Korisnik može zatražiti resetiranje lozinke, primiti poveznicu za resetiranje i uspješno resetirati lozinku.
F-004	Sustav omogućuje korisnicima pretraživanje liste učitelja s filtrima (jezik, cijena, dostupnost).	Visok	Zahtjev dionika	Korisnik može pretražiti i vidjeti listu učitelja filtriranu prema unesenim kriterijima.
F-005	Sustav omogućuje pregled profila učitelja s detaljima o kvalifikacijama, iskustvu i dostupnosti.	Visok	Povratne informacije korisnika	Korisnik može otvoriti profil učitelja i vidjeti ažurirane informacije.
F-006	Sustav omogućuje studentima unos preferencija učenja i ciljeva učenja na njihovim profilima.	Visok	Dokument zahtjeva	Student može dodati ili ažurirati preferencije i ciljeve učenja te ih spremiti na svom profilu.
F-007	Sustav omogućuje učiteljima unos kvalifikacija, iskustva i uređivanje kalendara dostupnosti.	Visok	Dokument zahtjeva	Učitelj može dodati ili izmijeniti svoje kvalifikacije, iskustvo i slobodne termine te ih uspješno spremiti.
F-008	Administrator može uređivati i brisati profile korisnika.	Visoki	Dokument zahtjeva	Administrator može urediti ili obrisati korisničke podatke i profil će biti ažuriran ili obrisan.
F-009	Sustav omogućuje studentima slanje zahtjeva za lekciju učitelju putem kalendara dostupnosti.	Visok	Zahtjev dionika	Student može odabrati termin i poslati zahtjev, a učitelj dobiva obavijest.

ID zahtjeva	Opis	Prioritet	Izvor	Kriteriji prihvaćanja
F-010	Sustav omogućuje učiteljima prihvaćanje ili odbijanje zahtjeva za lekciju.	Visok	Dokument zahtjeva	Učitelj može pregledati zahtjev, prihvatiti ili odbiti, a sustav obavještava studenta o statusu.
F-011	Sustav omogućuje studentima ocjenjivanje i komentiranje učitelja nakon održane lekcije.	Srednji	Povratne informacije korisnika	Student može ocijeniti učitelja i ostaviti komentar, koji se prikazuje na profilu učitelja.

Nefunkcionalni zahtjevi

ID zahtjeva	Opis	Prioritet
NF-001	Sustav treba imati intuitivno korisničko sučelje na engleskom jeziku koje prati standard “ISO 9241-210:2019 (Human-centred design for interactive systems)”.	Visok
NF-002	Sustav treba podržavati Unicode standard za prikaz i unos teksta.	Visok
NF-003	Sustav treba biti responzivan i prilagođen za sve desktop i mobilne uređaje.	Visok
NF-004	Sustav treba podržavati rad većeg broja korisnika istovremeno bez pada performansi.	Visok
NF-005	Sustav treba biti dostupan 99.9 % vremena kako bi se osigurala pouzdanost usluge.	Visok
NF-006	Sustav treba podržavati barem 30 jezika za koje je moguće održati lekciju.	Visok
NF-007	Sustav treba biti oblikovan tako da omogućuje jednostavno održavanje.	Srednji
NF-007.1	Sustav treba biti popraćen dovoljnom dokumentacijom.	Srednji
NF-007.1.1	Kod sustava treba biti dokumentiran prema “Code Conventions for the Java Programming Language”.	Srednji
NF-007.1.2	Pokretanje sustava za potpuni deployment je opisano kroz svaki korak.	Srednji

Dionici

Dionici: 1. Učenici - aktivni korisnici platforme koji uče jezike 2. Učitelji - pružatelji obrazovnih usluga na platformi 3. Administratori - odgovorni za održavanje i upravljanje platformom 4. Naručitelj (asistent) - odgovoran za specifikacije i zahtjeve projekta 5. Razvojni tim - realizira tehničke aspekte projekta

Aktori i njihovi funkcionalni zahtjevi

A-1 Guest (inicijator) može:

- **Funkcionalnost 1:** Registracija (F-001).
- **Funkcionalnost 2:** Pretraživanje liste učitelja (F-004).

A-2 Student (inicijator) može:

- **Funkcionalnost 1:** Prijava (F-002).
- **Funkcionalnost 2:** Pretraživanje liste učitelja (F-004).
- **Funkcionalnost 3:** Uređivanje profila:
  - **Podfunkcionalnost 1:** Unos preferencija učenja (F-006).
  - **Podfunkcionalnost 2:** Unos ciljeva učenja (F-007).
- **Funkcionalnost 4:** Dogovaranje lekcija:
  - **Podfunkcionalnost 1:** Slanje zahtjeva za lekciju (F-009).
- **Funkcionalnost 5:** Komentiranje i ocjenjivanje učitelja nakon lekcije (F-011).

A-3 Teacher (inicijator) može:

- **Funkcionalnost 1:** Prijava (F-002).
- **Funkcionalnost 2:** Uređivanje profila:
  - **Podfunkcionalnost 1:** Unos iskustva i kvalifikacija (F-008).
  - **Podfunkcionalnost 2:** Uređivanje kalendara (F-010).
- **Funkcionalnost 3:** Dogovaranje lekcija:
  - **Podfunkcionalnost 1:** Prihvaćanje zahtjeva za lekciju (F-010).

A-4 Admin (inicijator) može:

- **Funkcionalnost 1:** Upravljanje korisnicima:
  - **Podfunkcionalnost 1:** Uređivanje korisničkog profila (F-012).
  - **Podfunkcionalnost 2:** Brisanje korisničkog profila (F-013).

Obrasci uporabe

UC1 - Registracija

- Glavni sudionik: Guest
- Cilj: stvaranje računa kao Student ili Teacher
- Sudionici:
- Preduvjet: važeća adresa e-pošte
- Opis osnovnog tijeka: > 1. Unošenje osnovnih podataka > 2. Odabir vrste profila (učitelj ili učenik)

UC2 - Prijava

- Glavni sudionik: Student/Teacher
- Cilj: otključavanje svih funkcionalnosti sustava
- Sudionici:
- Preduvjet: obavljena registracija
- Opis osnovnog tijeka: > 1. Prijava u račun adresom e-pošte

UC3 - Pretraživanje liste učitelja

- Glavni sudionik: Guest/Student
- Cilj: Pregled svih dostupnih učitelja
- Sudionici:
- Preduvjet: nema
- Opis osnovnog tijeka: > 1. Pretraživanje liste učitelja > 2. Filtriranje po jeziku, kvalifikacijama, stilu podučavanja, satnici i kalendaru dostupnosti

UC4 - Uređivanje profila

- Glavni sudionik: Student/Teacher
- Cilj: nadopunjavanje profila bitnim informacijama
- Sudionici:
- Preduvjet: korisnik je prijavljen u sustav
- Opis osnovnog tijeka: > 1. Student označava jezike koje želi učiti, trenutačnu razinu znanja i preferirani stil podučavanja (UC4.1) > 2. Student postavlja ciljeve učenja (UC4.2) > 3. Učitelj dodaje jezike koje podučava, godine iskustva, kvalifikacije, stil podučavanja, sliku profila te satnicu koju naplaćuje (UC4.3) > 4. Učitelj uređuje svoj kalendar, označavajući kad je dostupan (UC4.4)

UC5 - Dogovaranje lekcije

- Glavni sudionik: Student i Teacher
- Cilj: Dogovaranje lekcije
- Sudionici:
- Preduvjet: korisnici su prijavljeni u sustav, učitelj predaje jezik kojeg studenta zanima, učitelj je slobodan u terminu kojeg student predlaže

- Opis osnovnog tijeka: > 1. Student odabire datum i vrijeme te šalje zahtjev za lekcijom (UC5.1) > 2. Učitelj prihvaća (UC5.2) ili odbija

UC5.2 - Prihvaćanje zahtjeva za lekcijom

- Glavni sudionik: Teacher
- Cilj: Prihvaćanje studentovog zahtjeva za lekcijom
- Sudionici: Student
- Preduvjet: Student je poslao zahtjev za lekcijom
- Opis osnovnog tijeka: > 1. Učitelj prihvaća lekciju > 2. Student je obaviješten o prihvaćanju (UC5.3) > 3. Otvara se chat funkcija između njih (UC5.4)

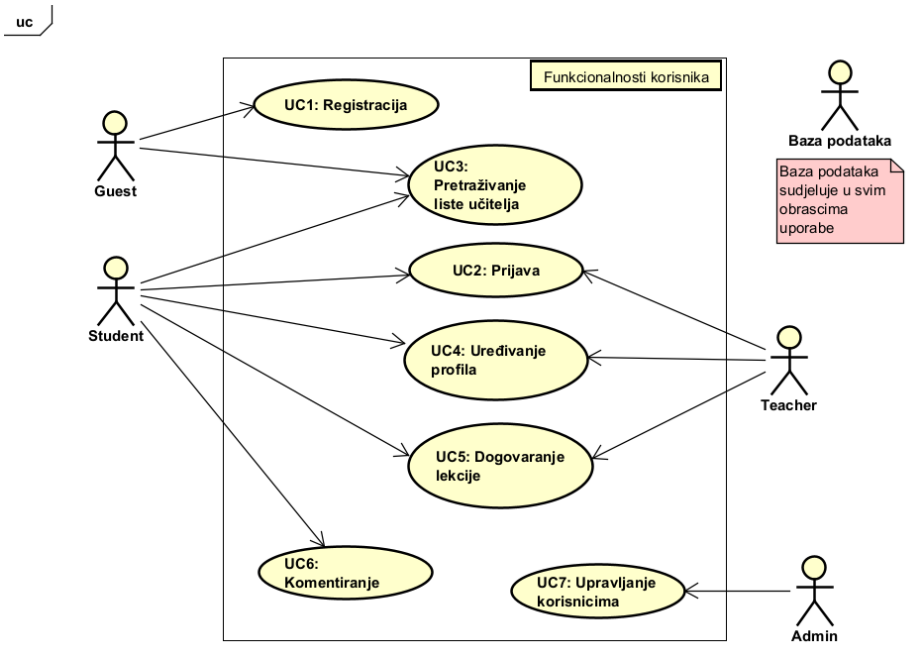
UC6 - Komentiranje i ocjenjivanje učitelja nakon lekcije

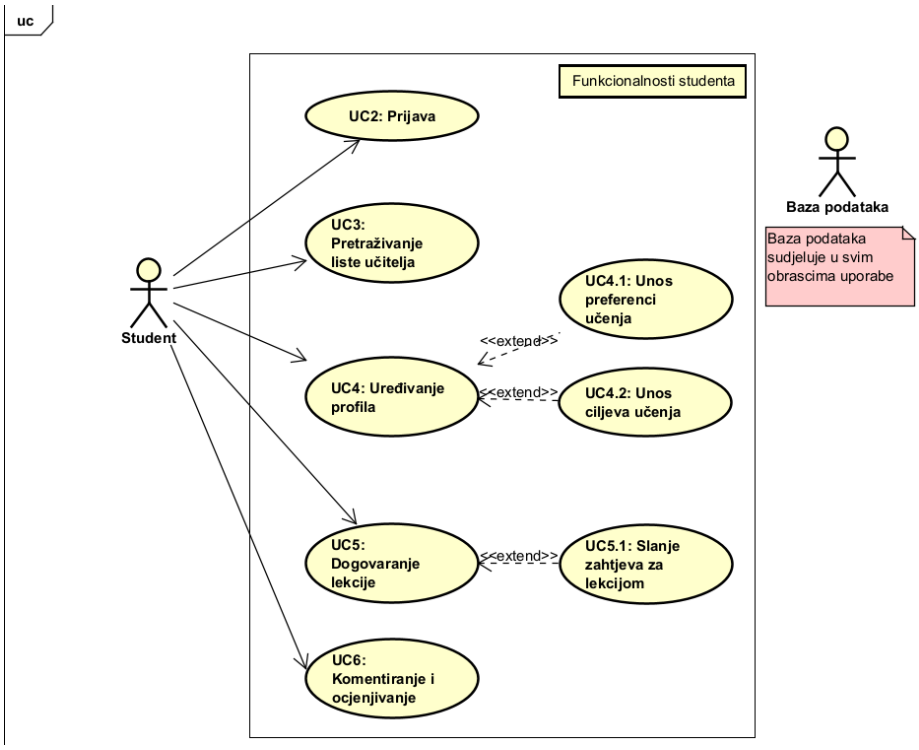
- Glavni sudionik: Student
- Cilj: ostavljanje dojma nakon lekcije
- Sudionici:
- Preduvjet: korisnik prijavljen u sustav kao Student, korisnik odradio barem jednu lekciju s učiteljem
- Opis osnovnog tijeka: > 1. Student komentira učitelja > 2. Student ostavlja ocjenu za učitelja

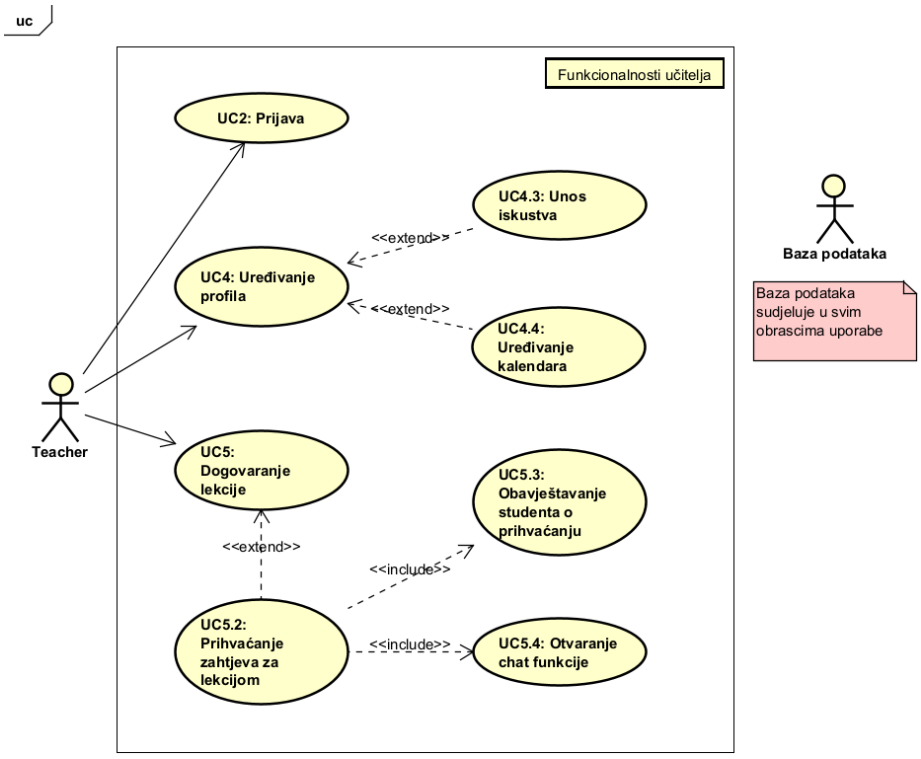
UC7 - Upravljanje korisnicima

- Glavni sudionik: Admin
- Cilj: kontrola nad korisničkim računima
- Sudionici:
- Preduvjet: korisnik prijavljen u sustav kao Admin
- Opis osnovnog tijeka: > 1. Admin uređuje korisnički profil > 2. Admin briše korisnički profil

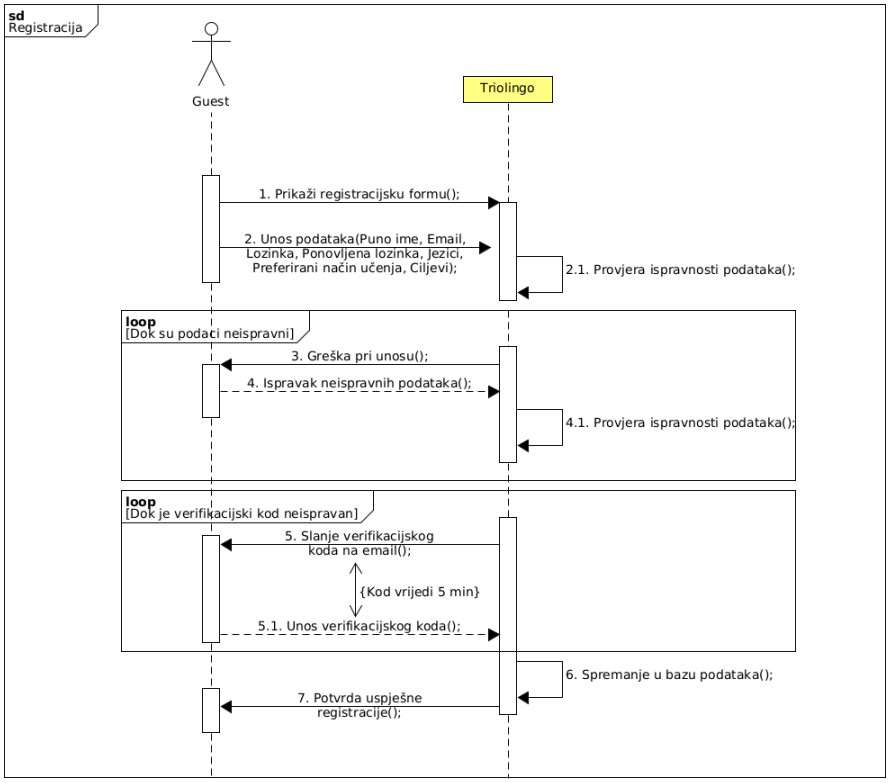
Dijagrami obrazaca uporabe

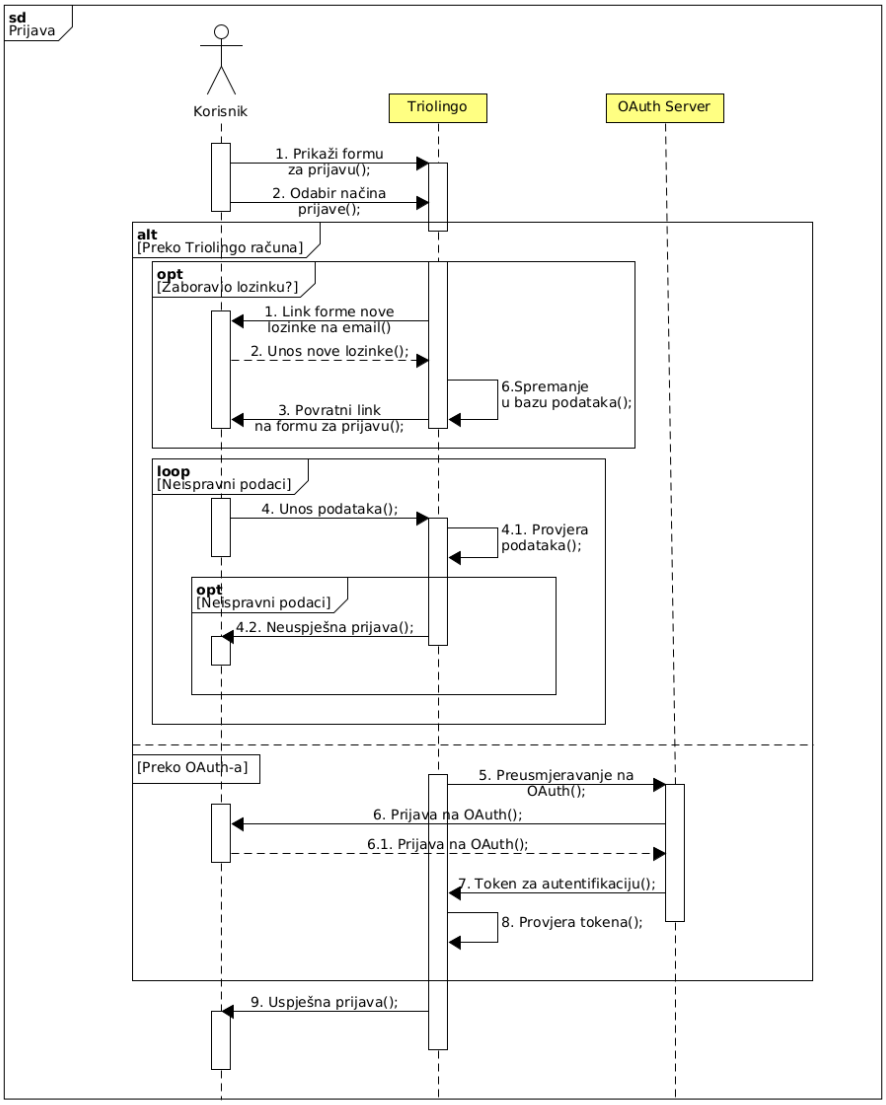


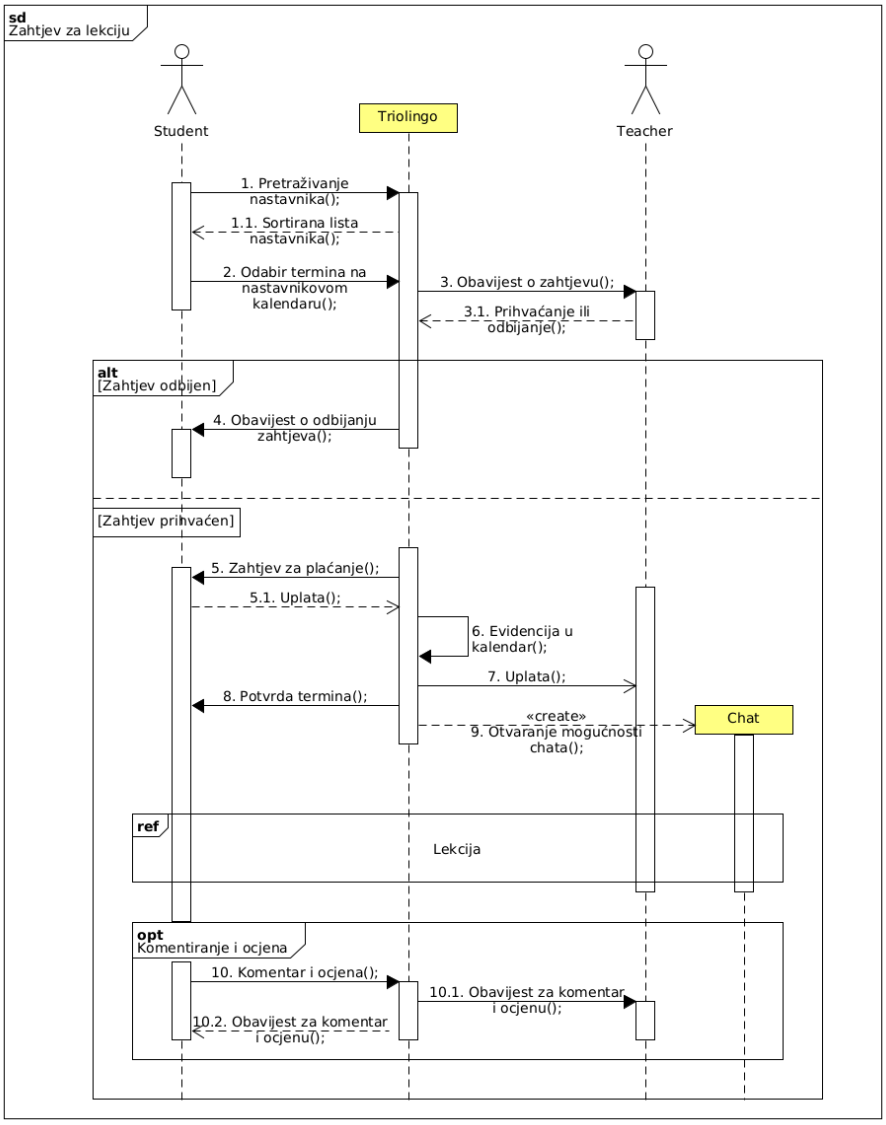

uc
Funkcionalnosti studenta
UC2: Prijava
UC3: Pretraživanje liste učitelja
UC4.1: Unos preferenci učenja
<<extend>>
UC4: Uređivanje profila
<<extend>>
UC4.2: Unos ciljeva učenja
UC5: Dogovaranje lekcije
<<extend>>
UC5.1: Slanje zahtjeva za lekcijom
UC6: Komentiranje i ocjenjivanje
Student
Baza podataka
Baza podataka sudjeluje u svim obrascima uporabe


uc
Funkcionalnosti učitelja
UC2: Prijava
UC4.3: Unos iskustva
<<extend>>
UC4: Uređivanje profila
<<extend>>
UC4.4: Uređivanje kalendara
Baza podataka
Baza podataka sudjeluje u svim obrascima uporabe
Teacher
UC5: Dogovaranje lekcije
UC5.3: Obavještavanje studenta o prihvaćanju
<<extend>>
<<include>>
UC5.2: Prihvaćanje zahtjeva za lekcijom
<<include>>
UC5.4: Otvaranje chat funkcije

Sekvencijski dijagrami




sd
Prijava
Korisnik
Triolingo
OAuth Server
1. Prikaži formu za prijavu();
2. Odabir načina prijave();
alt
[Preko Triolingo računa]
opt
[Zaboravio lozinku?]
1. Link forme nove lozinke na email()
2. Unos nove lozinke();
6.Spremanje u bazu podataka();
3. Povratni link na formu za prijavu();
loop
[Neispravni podaci]
4. Unos podataka();
4.1. Provjera podataka();
opt
[Neispravni podaci]
4.2. Neuspješna prijava();
[Preko OAuth-a]
5. Preusmjeravanje na OAuth();
6. Prijava na OAuth();
6.1. Prijava na OAuth();
7. Token za autentifikaciju();
8. Provjera tokena();
9. Uspješna prijava();



Opis Arhitekture

Izbor arhitekture

Za Triolingo odabrana je dvostruka arhitektura s odvojenim frontend i backend serverima, kako bi se osiguralo jasno razdvajanje odgovornosti između korisničkog sučelja i poslovne logike. Ovaj pristup poboljšava skalabilnost i omogućava neovisni razvoj i implementaciju frontend i backend dijelova.

Dodatno, Docker kontejnerizacija osigurava dosljednost razvojnih i produkcijskih okruženja, pojednostavljuje proces orkestracije i omogućava lako skaliranje.

Organizacija sustava s najviše razine apstrakcije

Klijent-poslužitelj

Sustav koristi klijent-poslužitelj arhitekturu gdje klijent (frontend u Reactu) komunicira s backend serverom putem REST API-ja. Backend server vrši sve poslovne operacije i vraća JSON odgovore natrag frontend serveru, čime se podržava učinkovita komunikacija i manji prijenos podataka.

Spremišta podataka

Koristi se relacijska baza podataka SQLite, uz Hibernate kao ORM koji omogućava rad s podacima kroz Java objekte. Relacijski model podržava složene relacije i transakcijske zahtjeve platforme, čime se osigurava integritet podataka za korisnike i lekcije.

Mrežni protokoli

HTTPS se koristi za sigurnu komunikaciju između klijenta i servera. HTTP REST protokol koristi se za komunikaciju između frontenda i backenda, čime se postiže jednostavna razmjena podataka u JSON formatu.

Organizacija aplikacije

Slojevi frontend i backend

Frontend sloj: React aplikacija na frontend serveru odgovorna je za prikaz korisničkog sučelja. Koristi se Vite za brzu kompilaciju i optimizaciju aplikacije. React aplikacija poslužuje se putem NGINX-a koji preusmjerava zahtjeve za statičke datoteke.

Backend sloj: Spring Boot API-only aplikacija na backend serveru odgovorna je za obradu poslovne logike, autentifikaciju korisnika, upravljanje rezervacijama i povratnim informacijama.

MVC arhitektura

Backend aplikacija koristi Model-View-Controller (MVC) arhitekturu, gdje:

- Model predstavlja poslovne podatke i logiku (upravljanje korisnicima, lekcijama).
- View nije prisutan jer je backend API-only, već svi odgovori idu kao JSON podaci klijentu.
- Controller obrađuje HTTP zahtjeve, izvodi operacije na modelu i vraća podatke u JSON formatu.

Identifikacija komponenata arhitekture

Podsustavi: Frontend server (React, NGINX), backend server (Spring Boot, Hibernate), i baza podataka (SQLite).

Preslikavanje na radnu platformu: Docker koristi se za deployment frontend i backend servera, omogućujući izolaciju, dosljednost i skalabilnost.

Globalni upravljački tok: Klijentski zahtjevi se primaju na frontend serveru, prenose se na backend putem HTTP-a, obrađuju i odgovori vraćaju frontend serveru, koji ih prikazuje korisniku.

Sklopovsko-programski zahtjevi: Docker omogućava fleksibilnost u razmještaju, dok Docker Compose može olakšati orkestraciju i nadzor nad cijelim okruženjem.

Baza podataka

Za potrebe našeg sustava koristit ćemo relacijsku bazu podataka SQLite. Baza podataka je dizajnirana kako bi omogućila efikasno pohranjivanje, izmjenu i dohvat podataka neophodnih za funkcionalnost aplikacije. S obzirom na relacijsku prirodu, baza omogućava modeliranje stvarnog svijeta kroz dobro definirane relacije (tablice), koje su opisane atributima (stupcima). SQLite je odabran zbog svoje portabilnosti, efikasnosti i jednostavnosti upotrebe, pogodan za aplikacije gdje paralelni pristup bazi nije prioritet.

Opis tablica

1. Korisnik

Naziv entiteta klase: `User`

Tablica **Korisnik** sadrži osnovne informacije o svim korisnicima unutar sustava, uključujući studente, učitelje i administratore. Svaki korisnik u sustavu jedinstveno je identificiran ID-om, a pohranjeni su i njegovo puno ime, e-mail adresa i šifrirana lozinka.

- **id**: Automatski se povećava, primarni ključ.
- **fullName**: Polje ne smije biti prazno.
- **email**: Polje ne smije biti prazno, validira se prema uzorku za provjeru valjanosti e-mail adrese.
- **password**: Polje ne smije biti prazno, pohranjuje se u šifriranom obliku.

Korisnicima su dodijeljene uloge koje određuju njihove dozvole unutar sustava. Uloge uključuju: - USER - ADMIN - TEACHER - STUDENT

Atribut	Tip Podatka	Opis
id	LONG	Jedinstveni identifikator korisnika (PK)
fullName	VARCHAR	Puno ime korisnika

Atribut	Tip Podatka	Opis
email	VARCHAR	E-mail adresa korisnika, mora biti valjana
password	VARCHAR	Šifrirana lozinka za autentikaciju korisnika

2. Student

Naziv entiteta klase: `Student`

Tablica **Student** proširuje tablicu Korisnik dodatnim specifičnostima koje se odnose na studente unutar aplikacije. Osim osnovnih informacija naslijeđenih iz klase Korisnik, svaki student ima dodatne informacije koje se odnose na njegov proces učenja.

- **id**: Primarni ključ, naslijeđen iz klase `User`.
- **learningLanguages**: Many-to-Many veza s entitetom `LearningLanguage` preko tablice `student_learning_language`. Veza definira koje jezike student trenutačno uči.
- **preferredTeachingStyle**: Enumeracija koja opisuje stil podučavanja.
- **learningGoals**: Tekstualno polje koje može sadržavati opisne ciljeve učenja.
- **LearningLanguages**: Tablica `student_learning_language` povezuje studente s jezicima koje uče. Svaki jezik je instance entiteta `LearningLanguage` (FK).

Svaki student je definiran unutar ovog sustava s namjerom da prati njegov edukativni napredak i preferencije, što doprinosi personaliziranom pristupu učenju.

Atribut	Tip Podatka	Opis
id	LONG	Jedinstveni identifikator korisnika, naslijeđen iz `User` (PK)
learningLanguages	List	Popis jezika koje student uči (FK)
preferredTeachingStyle	ENUM	Preferirani stil podučavanja (GROUP, INDIVIDUAL, FLEXIBLE)
learningGoals	TEXT	Opisani ciljevi učenja koje student želi postići

3. Učitelj

Naziv entiteta klase: `Teacher`

Tablica **Učitelj** proširuje tablicu Korisnik dodatnim specifičnostima koje se odnose na učitelje unutar aplikacije. Osim osnovnih informacija naslijeđenih iz

klase Korisnik, svaki učitelj ima dodatne informacije koje se odnose na njegovu profesionalnu kvalifikaciju i iskustvo u podučavanju.

- **id**: Primarni ključ, naslijeđen iz klase `User`.
- **languages**: Many-to-Many veza s entitetom `Language` preko tablice `teacher_language`. Ova veza definira koje jezike učitelj može podučavati.
- **yearsOfExperience**: Broj godina iskustva u podučavanju.
- **qualifications**: Kvalifikacije koje učitelj posjeduje.
- **teachingStyle**: Stil podučavanja koji učitelj preferira.
- **hourlyRate**: Satnica koju učitelj naplaćuje za svoje usluge.
- **profilePictureHash**: Hash vrijednost slike profila.
- **Languages**: Tablica `teacher_language` povezuje učitelje s jezicima koje podučavaju. Svaki jezik je instance entiteta `Language` (FK).

Atribut	Tip Podatka	Opis
id	LONG	Jedinstveni identifikator korisnika, naslijeđen iz `User` (PK)
languages	List	Jezici koje učitelj može podučavati (FK)
yearsOfExperience	INTEGER	Broj godina iskustva u podučavanju
qualifications	TEXT	Kvalifikacije učitelja
teachingStyle	ENUM	Preferirani stil podučavanja (GROUP, INDIVIDUAL, FLEXIBLE)
profilePictureHash	VARCHAR	Hash vrijednost slike profila
hourlyRate	DOUBLE	Satnica za usluge podučavanja

4. Jezik

Naziv entiteta klase: `Language`

Tablica **Language** sadrži informacije o različitim jezicima dostupnim u aplikaciji. Svaki jezik je jedinstveno identificiran svojim ID-om i imenom, što omogućava jednostavno referenciranje i korištenje u drugim dijelovima sustava kao što su lekcije i materijali za učenje.

- **id**: Primarni ključ.
- **name**: Ime jezika, ne smije biti prazno polje.
- **Language**: Ne povezuje se izravno s drugim entitetima putem stranih ključeva u ovoj tablici, ali se koristi kao referenca u entitetima poput `LearningLanguage` i `Teacher` za identifikaciju jezika koji se uči ili podučava.

Atribut	Tip Podatka	Opis
id	LONG	Jedinstveni identifikator jezika (PK)
name	VARCHAR	Ime jezika, mora biti valjano i nije null

5. Učenje jezika

Naziv entiteta klase: `LearningLanguage`

Tablica **LearningLanguage** sadrži informacije o jezicima koje učenici uče, uključujući specifični jezik i razinu znanja. Ovaj entitet povezuje učenike s jezicima koje uče, što omogućuje prilagodbu učenja individualnim potrebama i sposobnostima.

- **id**: Automatski se povećava, primarni ključ.
- **language**: One-to-One veza s entitetom `Language` koji definira jezik koji se uči.
- **knowledgeLevel**: Enumeracija koja definira razinu znanja jezika (početno, srednje, napredno).
- **Language**: Ova veza omogućuje identifikaciju specifičnog jezika koji student uči. Svaki jezik je instanca entiteta `Language` (FK).

Atribut	Tip Podatka	Opis
id	LONG	Jedinstveni identifikator jezika koji se uči (PK)
language	Language	Jezik koji se uči (FK)
knowledgeLevel	ENUM	Razine znanja (BEGINNER, INTERMEDIATE, ADVANCED)

6. Recenzija

Naziv entiteta klase: `Review`

Tablica **Recenzija** sadrži povratne informacije koje korisnici ostavljaju za usluge ili proizvode unutar sustava. Svaka recenzija u sustavu jedinstveno je identificirana ID-om, i povezana je s korisnikom koji ju je ostavio. Recenzije pomažu u poboljšanju kvalitete usluga i proizvoda te služe kao direktan feedback od korisnika.

- **id**: Automatski se povećava, primarni ključ.
- **userId**: Identifikator korisnika koji je ostavio recenziju. Veza s tablicom **Korisnik** (FK).
- **content**: Tekst recenzije. Ovo polje može biti prazno ako korisnik nije ostavio tekstualni komentar.

- **rating**: Numerička ocjena koju korisnik daje usluzi ili proizvodu. Ocjene su tipično u rasponu od 1 do 5.
- **reviewDate**: Datum kada je recenzija ostavljena. Ne smije biti prazno.

Atribut	Tip Podatka	Opis
id	LONG	Jedinstveni identifikator recenzije (PK)
userId	LONG	Identifikator korisnika koji je ostavio recenziju (FK)
content	VARCHAR	Tekst recenzije
rating	INTEGER	Numerička ocjena recenzije (1-5)
reviewDate	DATE	Datum kada je recenzija ostavljena

7. Lekcija

Naziv entiteta klase: `Lesson`

Tablica **Lekcija** služi za upravljanje pojedinačnim nastavnim sesijama. Svakoj lekciji se određuje vremenski termin te status, što omogućava učiteljima i učenicima da imaju pregled planiranih i odrađenih lekcija.

- **id**: Automatski se povećava, PK.
- **Date**: Datum održavanja lekcije.
- **Time**: Vrijeme održavanja lekcije.
- **Status**: Status lekcije (ENUM: Scheduled, Completed, In progress, Cancelled).
- **TeacherID**: Identifikator učitelja koji vodi lekciju (FK).
- **StudentID**: Identifikator učenika koji sudjeluje u lekciji (FK).
- **LanguageName**: Naziv jezika koji se uči tijekom lekcije (FK).
- **TeacherPayment**: Iznos plaćen učitelju za lekciju.
  - Svaka lekcija je povezana s jednim učiteljem i jednim učenikom, što ukazuje na veze Many-to-One iz Lekcije prema Učitelju i Učeniku.

Atribut	Tip Podatka	Opis
id	LONG	Jedinstveni identifikator lekcije (PK)
Date	DATE	Datum održavanja lekcije
Time	TIME	Vrijeme održavanja lekcije
Status	ENUM	Status lekcije (Scheduled, Completed, In progress, Cancelled)
TeacherID	LONG	ID učitelja koji vodi lekciju (FK)
StudentID	LONG	ID učenika koji sudjeluje u lekciji (FK)

Atribut	Tip Podatka	Opis
LanguageName	VARCHAR	Naziv jezika koji se uči (FK)
TeacherPaymen	LONG	Iznos plaćen učitelju za lekciju

ER Dijagram baze podataka

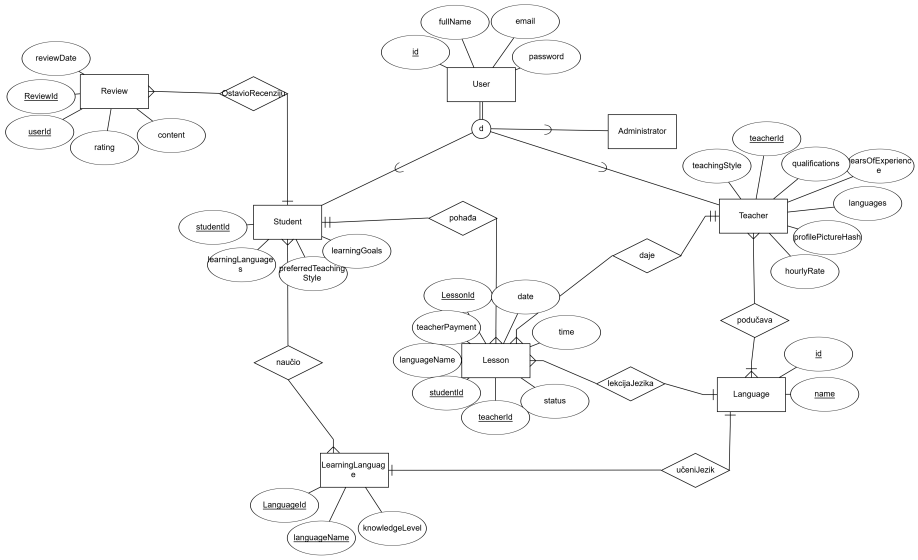


Figure 1: ER Dijagram baze podataka

Dijagram razreda

Na slikama prikazani su dijagrami razreda u backend dijelu arhitekture. Slika 1 prikazuje entitete. Razredi na slikama 2-9 opisuju controllere. Metode tih razreda manipuliraju DTO-ima (Data transfer object) koji su dohvaćeni s pomoću metoda implementiranih u Model razredima. Metode implementirane u Controller razredima vraćaju JSON datoteke s HTTP statusnim kodom. Slike 10-12 prikazuju dijagram DTO razreda. Slika 13 prikazuje dijagram security razreda.

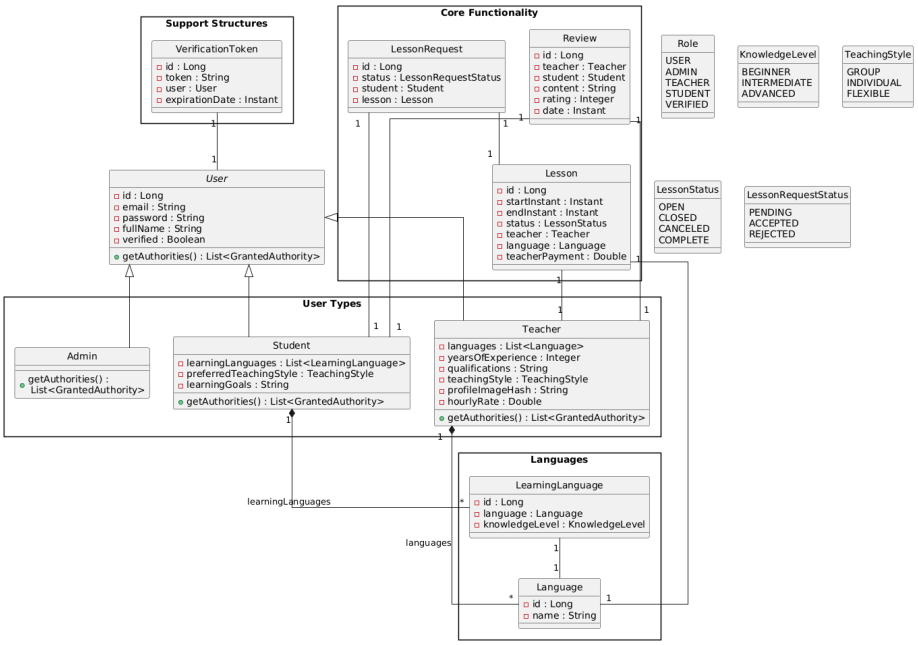
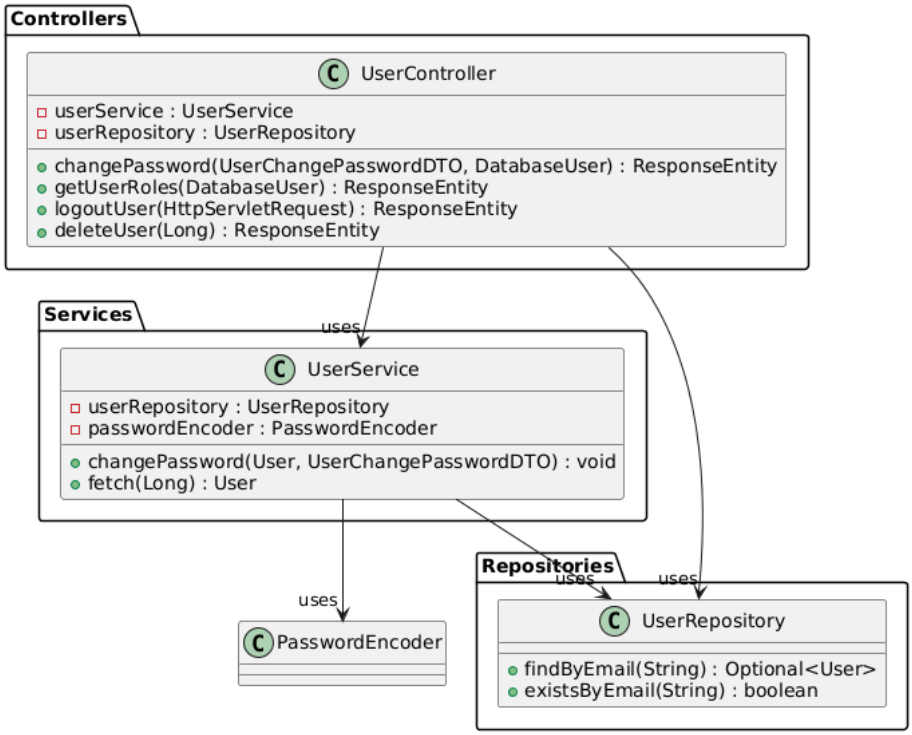
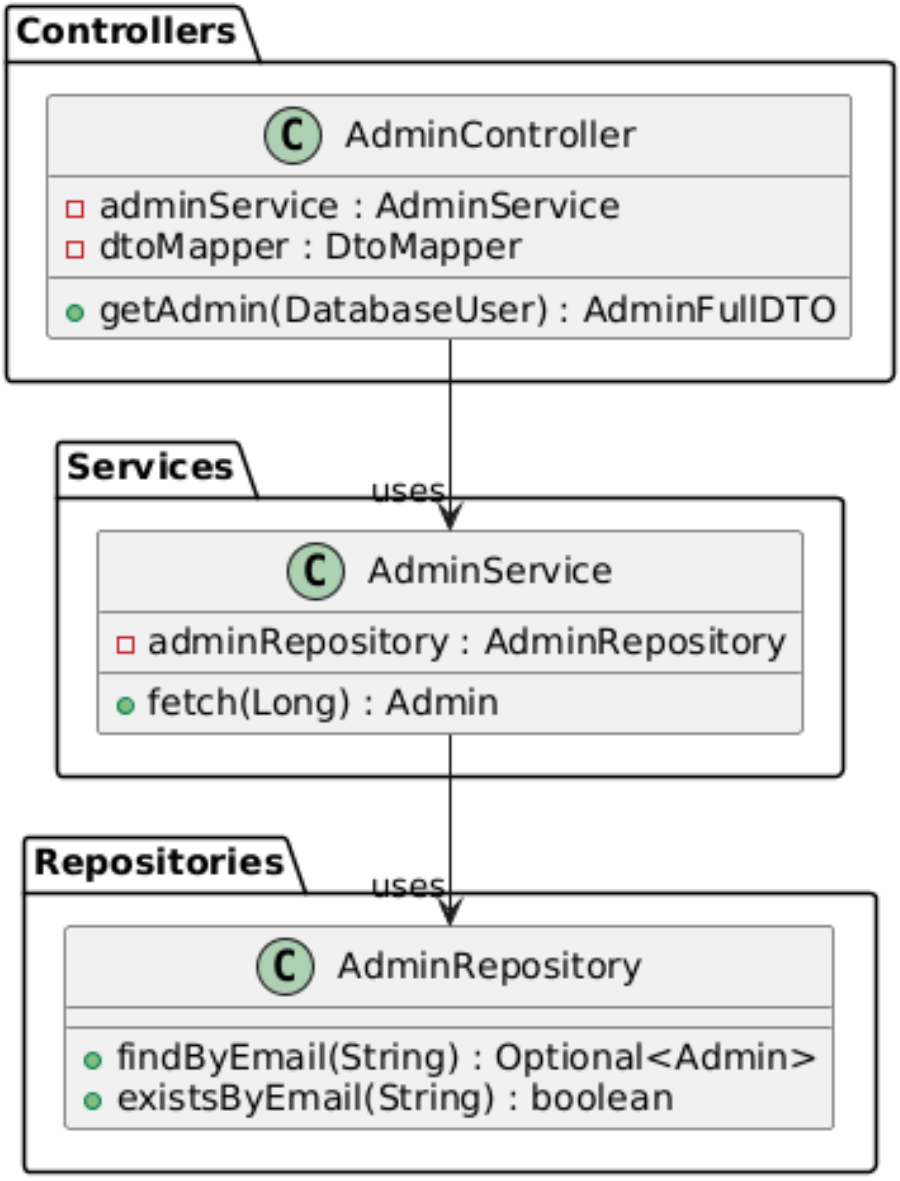
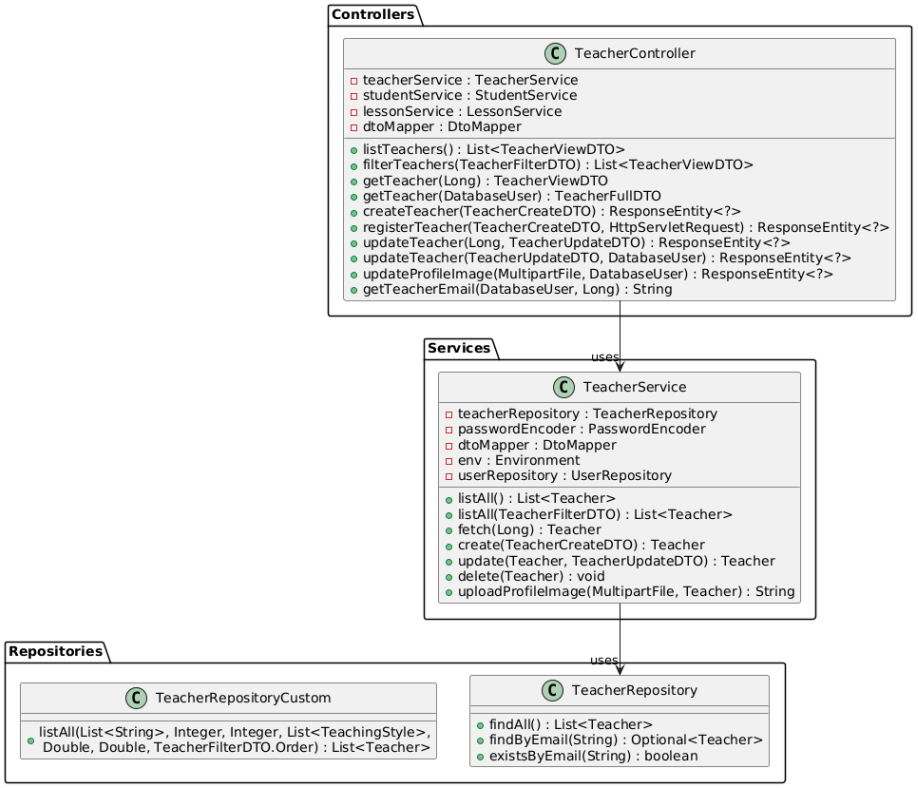
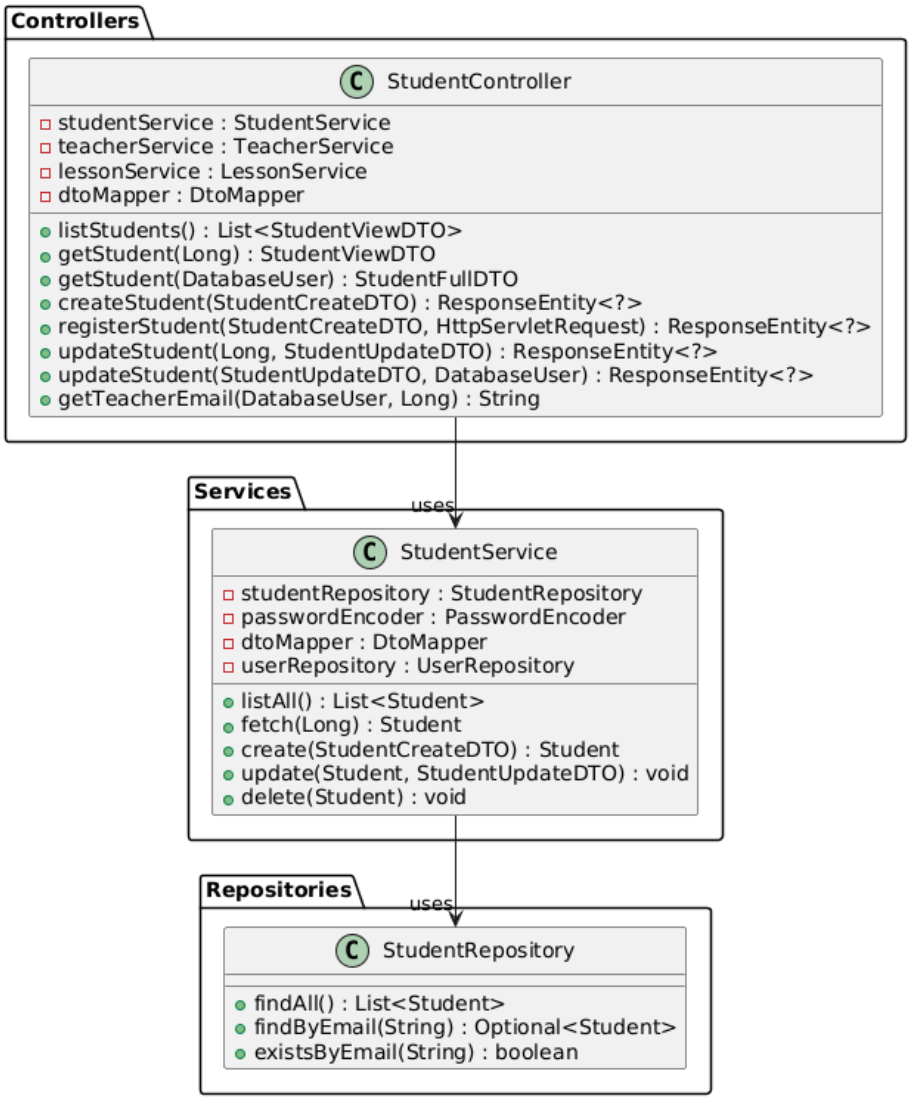


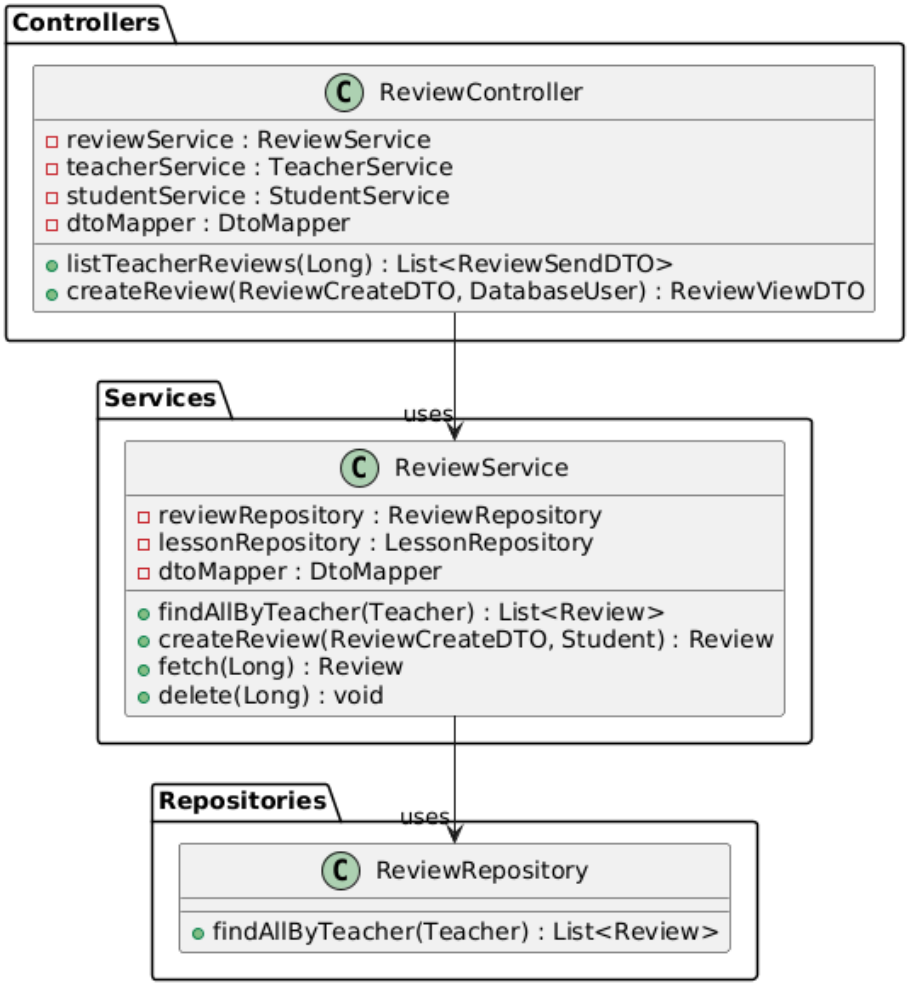
Figure 2: Dijagram razreda

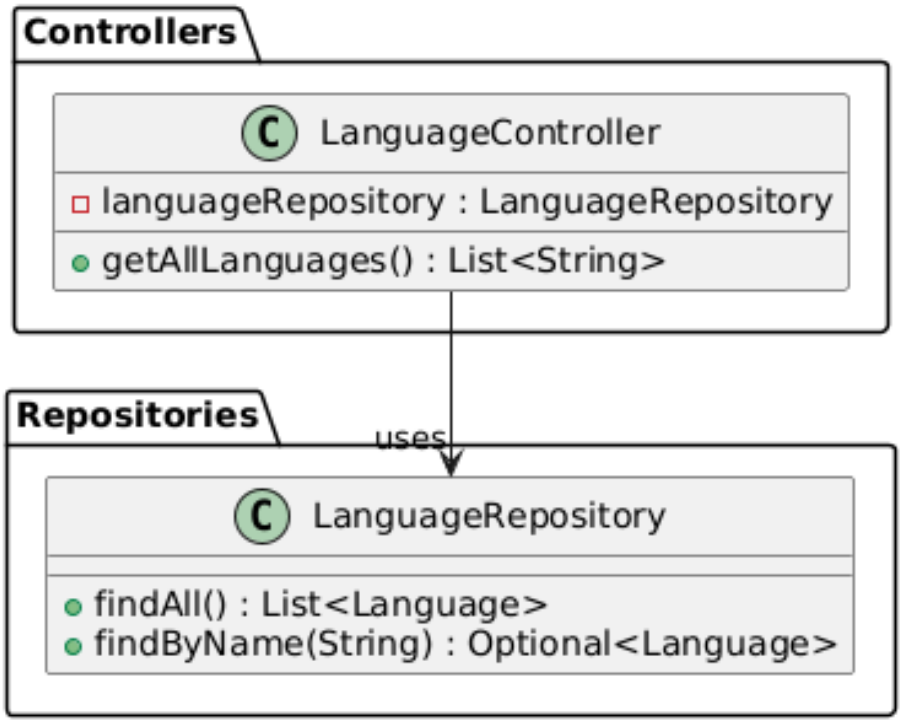


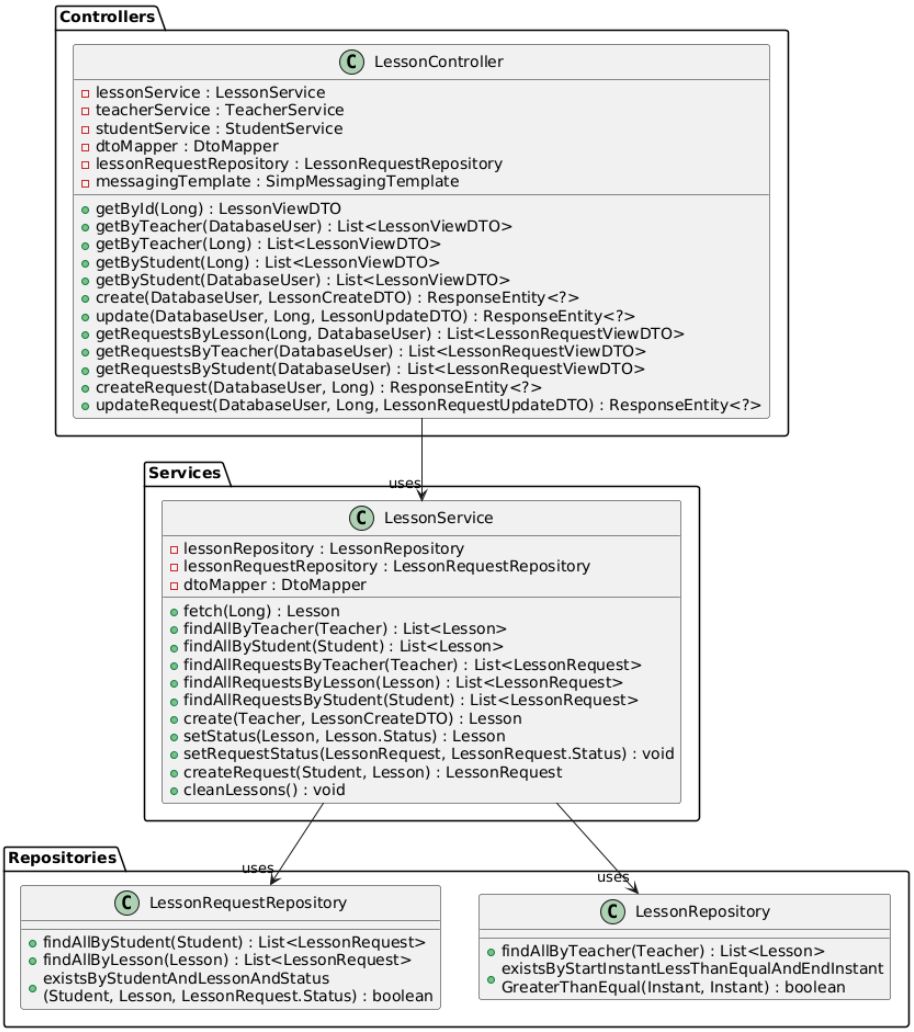

Controllers
AdminController
adminService : AdminService
dtoMapper : DtoMapper
getAdmin(DatabaseUser) : AdminFullDTO
uses
Services
AdminService
adminRepository : AdminRepository
fetch(Long) : Admin
uses
Repositories
AdminRepository
findByEmail(String) : Optional<Admin>
existsByEmail(String) : boolean

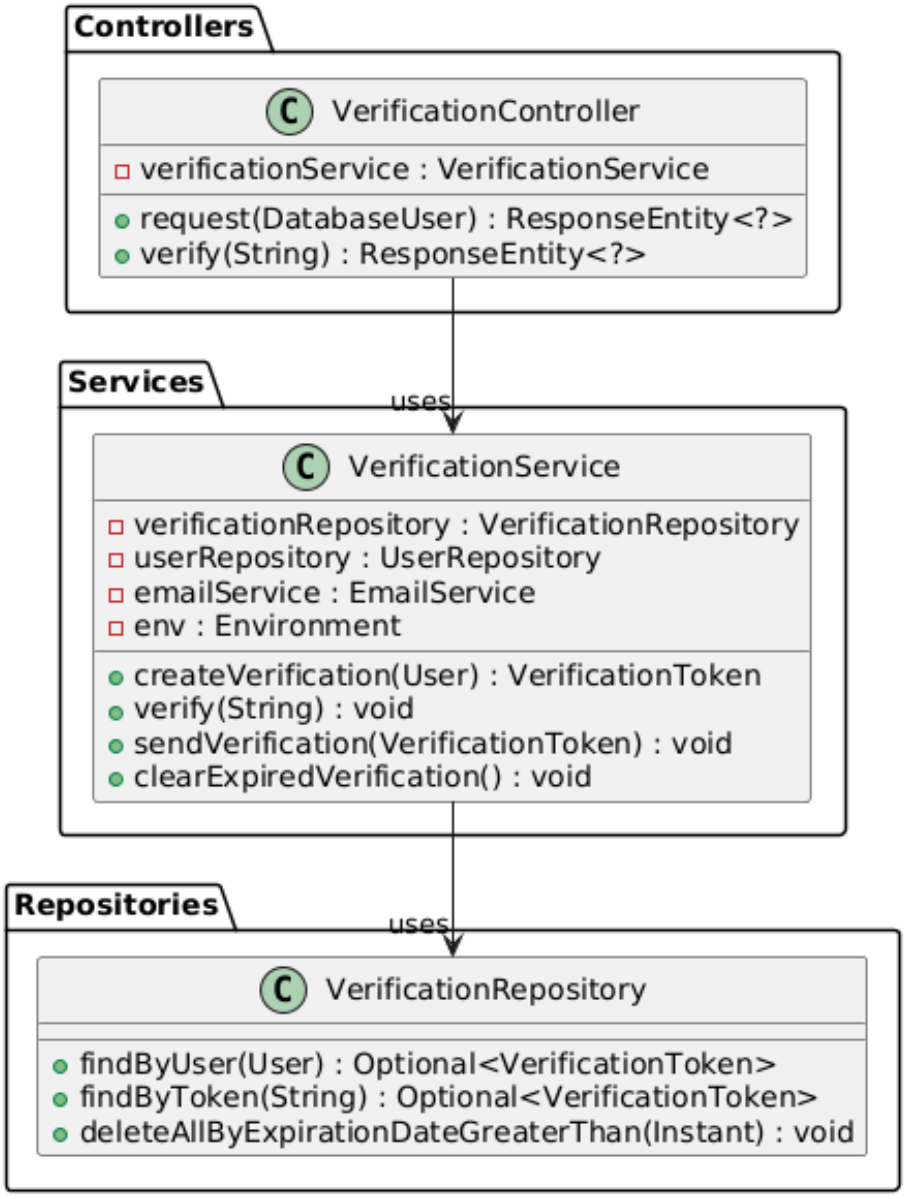

Controllers
TeacherController
- teacherService : TeacherService
- studentService : StudentService
- lessonService : LessonService
- dtoMapper : DtoMapper
+ listTeachers() : List<TeacherViewDTO>
+ filterTeachers(TeacherFilterDTO) : List<TeacherViewDTO>
+ getTeacher(Long) : TeacherViewDTO
+ getTeacher(DatabaseUser) : TeacherFullDTO
+ createTeacher(TeacherCreateDTO) : ResponseEntity<?>
+ registerTeacher(TeacherCreateDTO, HttpServletRequest) : ResponseEntity<?>
+ updateTeacher(Long, TeacherUpdateDTO) : ResponseEntity<?>
+ updateTeacher(TeacherUpdateDTO, DatabaseUser) : ResponseEntity<?>
+ updateProfileImage(MultipartFile, DatabaseUser) : ResponseEntity<?>
+ getTeacherEmail(DatabaseUser, Long) : String
uses
Services
TeacherService
- teacherRepository : TeacherRepository
- passwordEncoder : PasswordEncoder
- dtoMapper : DtoMapper
- env : Environment
- userRepository : UserRepository
+ listAll() : List<Teacher>
+ listAll(TeacherFilterDTO) : List<Teacher>
+ fetch(Long) : Teacher
+ create(TeacherCreateDTO) : Teacher
+ update(Teacher, TeacherUpdateDTO) : Teacher
+ delete(Teacher) : void
+ uploadProfileImage(MultipartFile, Teacher) : String
uses
Repositories
TeacherRepositoryCustom
+ listAll(List<String>, Integer, Integer, List<TeachingStyle>, Double, Double, TeacherFilterDTO.Order) : List<Teacher>
TeacherRepository
+ findAll() : List<Teacher>
+ findByEmail(String) : Optional<Teacher>
+ existsByEmail(String) : boolean

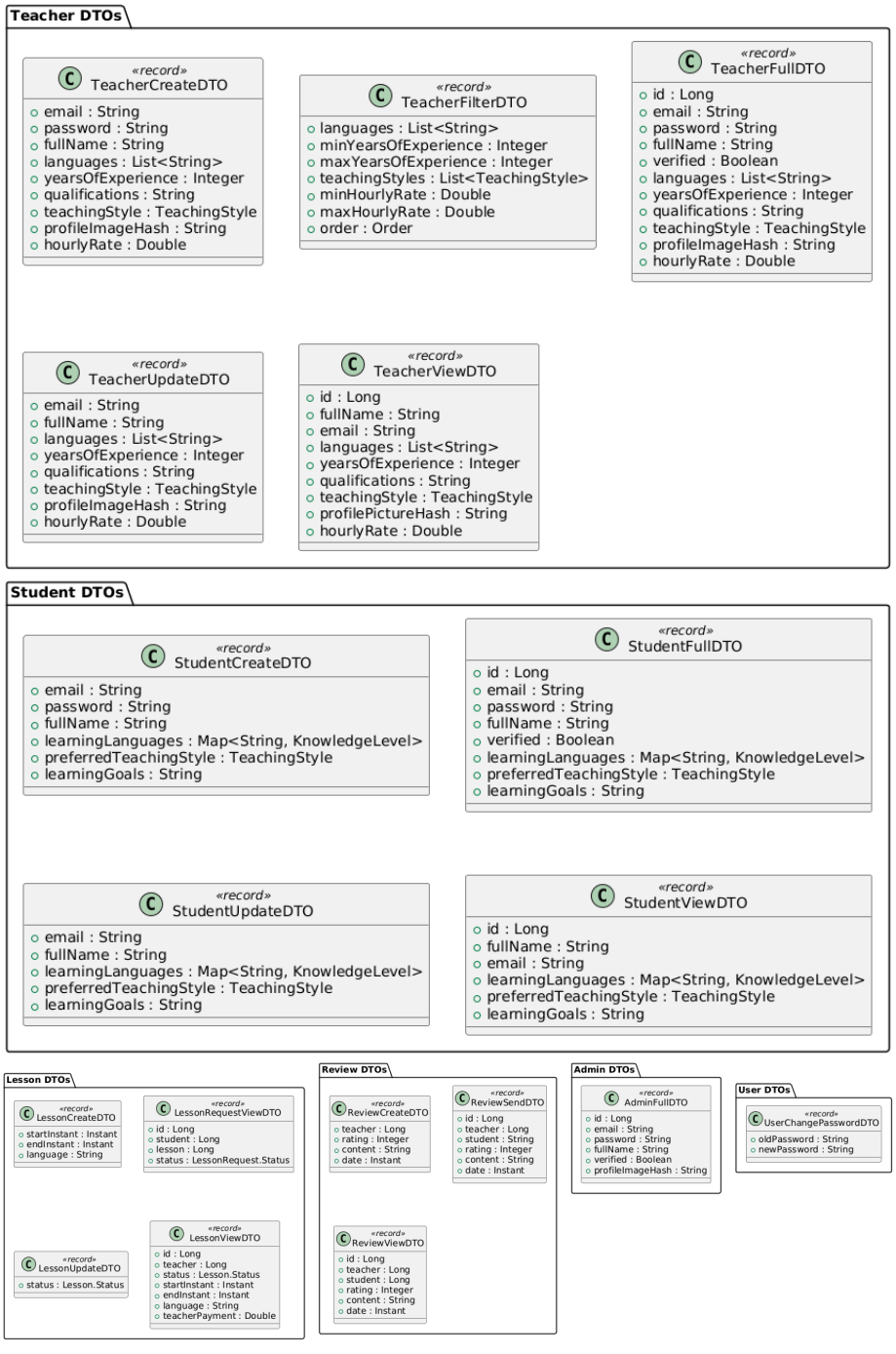

Controllers
StudentController
- studentService : StudentService
- teacherService : TeacherService
- lessonService : LessonService
- dtoMapper : DtoMapper
+ listStudents() : List<StudentViewDTO>
+ getStudent(Long) : StudentViewDTO
+ getStudent(DatabaseUser) : StudentFullDTO
+ createStudent(StudentCreateDTO) : ResponseEntity<?>
+ registerStudent(StudentCreateDTO, HttpServletRequest) : ResponseEntity<?>
+ updateStudent(Long, StudentUpdateDTO) : ResponseEntity<?>
+ updateStudent(StudentUpdateDTO, DatabaseUser) : ResponseEntity<?>
+ getTeacherEmail(DatabaseUser, Long) : String
uses
Services
StudentService
- studentRepository : StudentRepository
- passwordEncoder : PasswordEncoder
- dtoMapper : DtoMapper
- userRepository : UserRepository
+ listAll() : List<Student>
+ fetch(Long) : Student
+ create(StudentCreateDTO) : Student
+ update(Student, StudentUpdateDTO) : void
+ delete(Student) : void
uses
Repositories
StudentRepository
+ findAll() : List<Student>
+ findByEmail(String) : Optional<Student>
+ existsByEmail(String) : boolean


Controllers
ReviewController
reviewService : ReviewService
teacherService : TeacherService
studentService : StudentService
dtoMapper : DtoMapper
listTeacherReviews(Long) : List<ReviewSendDTO>
createReview(ReviewCreateDTO, DatabaseUser) : ReviewViewDTO
uses
Services
ReviewService
reviewRepository : ReviewRepository
lessonRepository : LessonRepository
dtoMapper : DtoMapper
findAllByTeacher(Teacher) : List<Review>
createReview(ReviewCreateDTO, Student) : Review
fetch(Long) : Review
delete(Long) : void
uses
Repositories
ReviewRepository
findAllByTeacher(Teacher) : List<Review>


Controllers
LanguageController
languageRepository : LanguageRepository
getAllLanguages() : List<String>
uses
Repositories
LanguageRepository
findAll() : List<Language>
findByName(String) : Optional<Language>


Controllers
LessonController
- lessonService : LessonService
- teacherService : TeacherService
- studentService : StudentService
- dtoMapper : DtoMapper
- lessonRequestRepository : LessonRequestRepository
- messagingTemplate : SimpMessagingTemplate
+ getById(Long) : LessonViewDTO
+ getByTeacher(DatabaseUser) : List<LessonViewDTO>
+ getByTeacher(Long) : List<LessonViewDTO>
+ getByStudent(Long) : List<LessonViewDTO>
+ getByStudent(DatabaseUser) : List<LessonViewDTO>
+ create(DatabaseUser, LessonCreateDTO) : ResponseEntity<?>
+ update(DatabaseUser, Long, LessonUpdateDTO) : ResponseEntity<?>
+ getRequestsByLesson(Long, DatabaseUser) : List<LessonRequestViewDTO>
+ getRequestsByTeacher(DatabaseUser) : List<LessonRequestViewDTO>
+ getRequestsByStudent(DatabaseUser) : List<LessonRequestViewDTO>
+ createRequest(DatabaseUser, Long) : ResponseEntity<?>
+ updateRequest(DatabaseUser, Long, LessonRequestUpdateDTO) : ResponseEntity<?>
uses
Services
LessonService
- lessonRepository : LessonRepository
- lessonRequestRepository : LessonRequestRepository
- dtoMapper : DtoMapper
+ fetch(Long) : Lesson
+ findAllByTeacher(Teacher) : List<Lesson>
+ findAllByStudent(Student) : List<Lesson>
+ findAllRequestsByTeacher(Teacher) : List<LessonRequest>
+ findAllRequestsByLesson(Lesson) : List<LessonRequest>
+ findAllRequestsByStudent(Student) : List<LessonRequest>
+ create(Teacher, LessonCreateDTO) : Lesson
+ setStatus(Lesson, Lesson.Status) : Lesson
+ setRequestStatus(LessonRequest, LessonRequest.Status) : void
+ createRequest(Student, Lesson) : LessonRequest
+ cleanLessons() : void
uses
uses
Repositories
LessonRequestRepository
+ findAllByStudent(Student) : List<LessonRequest>
+ findAllByLesson(Lesson) : List<LessonRequest>
+ existsByStudentAndLessonAndStatus (Student, Lesson, LessonRequest.Status) : boolean
LessonRepository
+ findAllByTeacher(Teacher) : List<Lesson>
+ existsByStartInstantLessThanEqualAndEndInstant GreaterThanEqual(Instant, Instant) : boolean


Controllers
VerificationController
verificationService : VerificationService
request(DatabaseUser) : ResponseEntity<?>
verify(String) : ResponseEntity<?>
uses
Services
VerificationService
verificationRepository : VerificationRepository
userRepository : UserRepository
emailService : EmailService
env : Environment
createVerification(User) : VerificationToken
verify(String) : void
sendVerification(VerificationToken) : void
clearExpiredVerification() : void
uses
Repositories
VerificationRepository
findByUser(User) : Optional<VerificationToken>
findByToken(String) : Optional<VerificationToken>
deleteAllByExpirationDateGreaterThan(Instant) : void


Teacher DTOs
«record»
TeacherCreateDTO
email : String
password : String
fullName : String
languages : List<String>
yearsOfExperience : Integer
qualifications : String
teachingStyle : TeachingStyle
profileImageHash : String
hourlyRate : Double
«record»
TeacherFilterDTO
languages : List<String>
minYearsOfExperience : Integer
maxYearsOfExperience : Integer
teachingStyles : List<TeachingStyle>
minHourlyRate : Double
maxHourlyRate : Double
order : Order
«record»
TeacherFullDTO
id : Long
email : String
password : String
fullName : String
verified : Boolean
languages : List<String>
yearsOfExperience : Integer
qualifications : String
teachingStyle : TeachingStyle
profileImageHash : String
hourlyRate : Double
«record»
TeacherUpdateDTO
email : String
fullName : String
languages : List<String>
yearsOfExperience : Integer
qualifications : String
teachingStyle : TeachingStyle
profileImageHash : String
hourlyRate : Double
«record»
TeacherViewDTO
id : Long
fullName : String
email : String
languages : List<String>
yearsOfExperience : Integer
qualifications : String
teachingStyle : TeachingStyle
profilePictureHash : String
hourlyRate : Double
Student DTOs
«record»
StudentCreateDTO
email : String
password : String
fullName : String
learningLanguages : Map<String, KnowledgeLevel>
preferredTeachingStyle : TeachingStyle
learningGoals : String
«record»
StudentFullDTO
id : Long
email : String
password : String
fullName : String
verified : Boolean
learningLanguages : Map<String, KnowledgeLevel>
preferredTeachingStyle : TeachingStyle
learningGoals : String
«record»
StudentUpdateDTO
email : String
fullName : String
learningLanguages : Map<String, KnowledgeLevel>
preferredTeachingStyle : TeachingStyle
learningGoals : String
«record»
StudentViewDTO
id : Long
fullName : String
email : String
learningLanguages : Map<String, KnowledgeLevel>
preferredTeachingStyle : TeachingStyle
learningGoals : String
Lesson DTOs
«record»
LessonCreateDTO
startInstant : Instant
endInstant : Instant
language : String
«record»
LessonRequestViewDTO
id : Long
student : Long
lesson : Long
status : LessonRequest.Status
«record»
LessonUpdateDTO
status : Lesson.Status
«record»
LessonViewDTO
id : Long
teacher : Long
status : Lesson.Status
startInstant : Instant
endInstant : Instant
language : String
teacherPayment : Double
Review DTOs
«record»
ReviewCreateDTO
teacher : Long
rating : Integer
content : String
date : Instant
«record»
ReviewSendDTO
id : Long
teacher : Long
student : String
rating : Integer
content : String
date : Instant
«record»
ReviewViewDTO
id : Long
teacher : Long
student : Long
rating : Integer
content : String
date : Instant
Admin DTOs
«record»
AdminFullDTO
id : Long
email : String
password : String
fullName : String
verified : Boolean
profileImageHash : String
User DTOs
«record»
UserChangePasswordDTO
oldPassword : String
newPassword : String

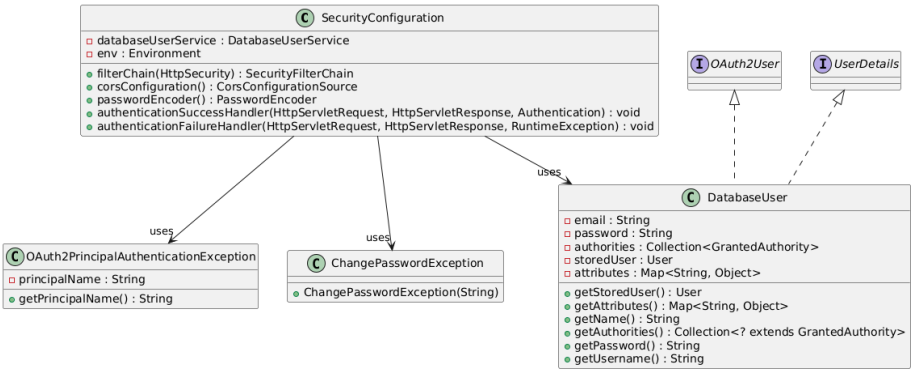
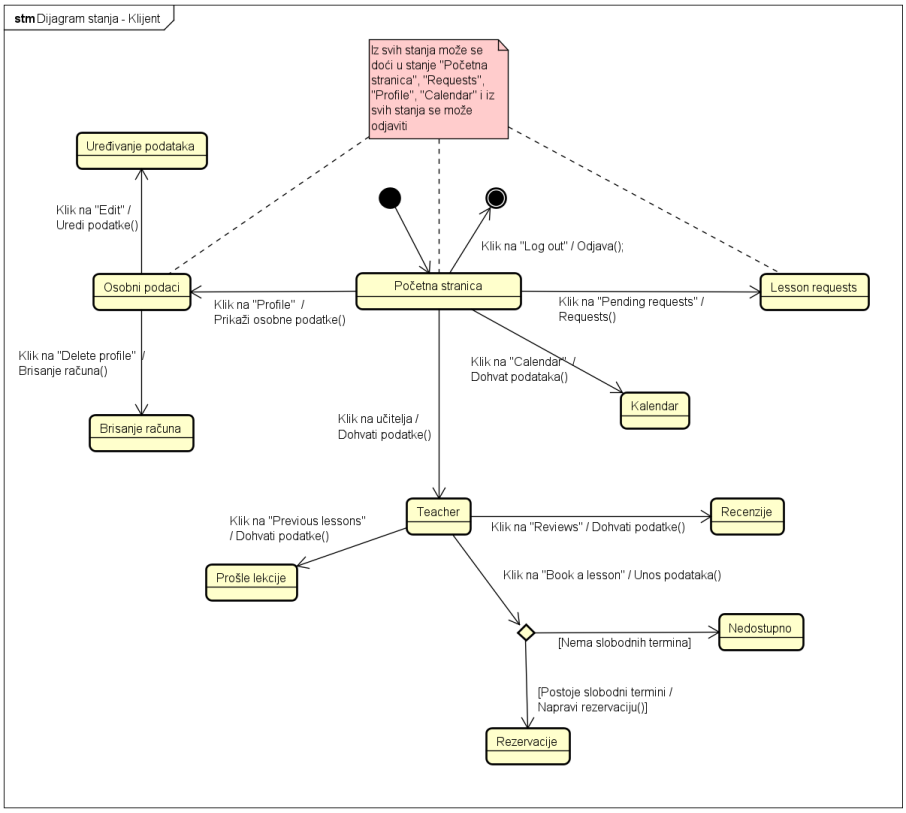


Figure 3: Security

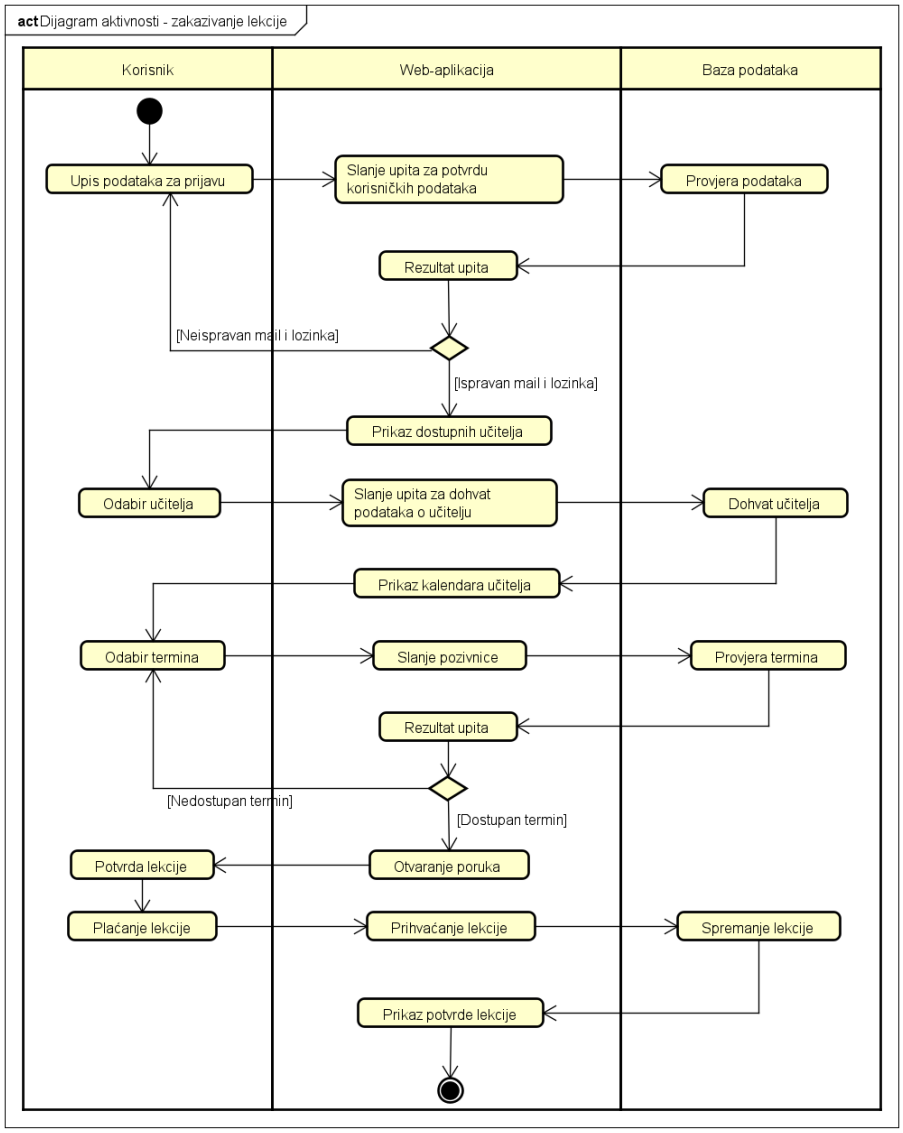
Dijagram stanja

Dijagram stanja prikazuje stanja objekta te prijelaze iz jednog stanja u drugo temeljene na događajima. Na slici je prikazan dijagram stanja za registriranog korisnika (studenta). Nakon prijave, klijentu se prikazuje početna stranica na kojoj može pregledati učitelje i filtrirati ih. Za odabranog učitelja ima opciju pregledavanja recenzija i ocjena učitelja te opciju za stvaranje rezervacije lekcije. Također, klijent može na početnoj stranici odabrati neku od ponuđenih mogućnosti. Klikom na ”Osobni podatci” prikazuju mu se njegovi podatci, koje može urediti ili odabrati opciju brisanja računa. Odabirom ”Pending requests” otvara mu se popis zahtjeva za rezervaciju lekcije. Klikom na “Calendar” može pregledati termine svojih lekcija.



Dijagram aktivnosti

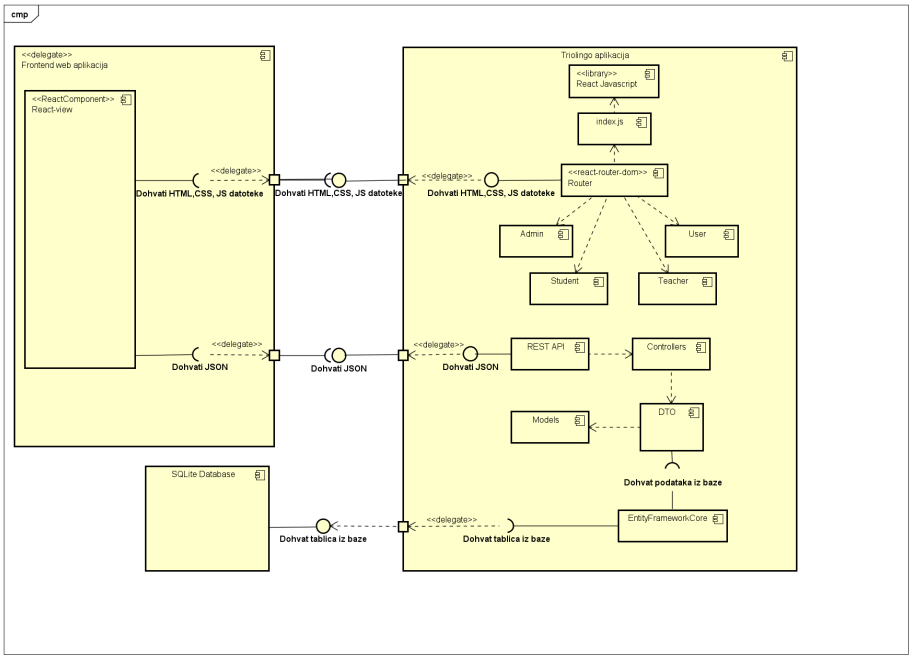
Dijagram aktivnosti koristi se za prikaz modela toka upravljanja ili toka podataka, naglašavajući jednostavnost u izvršenju koraka nakon završetka prethodnog. U prikazanom dijagramu aktivnosti detaljno je ilustriran proces zakazivanja lekcije na Triolingo platformi. Korisnik se prijavljuje, bira učitelja i termin, te potvrđuje i plaća lekciju. Web-aplikacija i baza podataka koordiniraju provjeru podataka, prikaz učitelja i kalendara, slanje pozivnica, te spremanje i ažuriranje podataka o lekciji.



Dijagram komponenti

Dijagram komponenti ilustrira kako je aplikacija podijeljena na frontend i backend segmente, koristeći klijent-poslužitelj arhitekturu za efikasno upravljanje korisničkim interakcijama i poslovnim logikama. Frontend se temelji na Reactu, s NGINX poslužiteljem koji upravlja statičkim datotekama i dinamičkim rutama putem React Routera. Ova konfiguracija omogućava dinamičko ažuriranje korisničkog sučelja i interakciju s koris-

nicima. Backend je razvijen koristeći Spring Boot za kreiranje RESTful API-ja, s Hibernate ORM za interakciju s SQLite bazom podataka. Ovo omogućava jednostavno upravljanje podacima i njihov siguran prijenos preko HTTPS-a u JSON formatu koristeći DTO za izolaciju aplikacijskih slojeva.



Korištene tehnologije i alati

Programski jezici

TypeScript

Frontend je razvijen u TypeScriptu jer omogućuje statičko tipiziranje, što smanjuje broj grešaka tijekom razvoja. Također, TypeScript je nadskup JavaScripta i time možemo koristiti bogat ekosustav biblioteka i radnih okvira.

Java 21

Backend je napisan u Javi 21 jer je pouzdan jezik s dugom poviješću u razvoju enterprise aplikacija. Najveća prednost je to što Java omogućuje korištenje Spring Boot okvira, što ubrzava razvoj, olakšava integraciju sigurnosnih značajki i upravljanje bazama podataka.

Biblioteke i radni okviri

React

React 18.3 korišten je za frontend jer omogućuje stvaranje modularnih komponenti koje se lako održavaju i ponovno koriste. Njegova popularnost i velika zajednica osiguravaju pristup brojnim resursima i bibliotekama, što ubrzava razvoj i smanjuje vrijeme potrebno za učenje.

Vite

Vite je korišten za upravljanje projektom i razvojni server jer je brz i jednostavan za konfiguraciju, što povećava produktivnost frontend razvoja.

Spring Boot

Spring Boot 3.3.4 korišten je na backendu jer omogućuje brzo kreiranje RESTful servisa i upravljanje bazama podataka. Spring Security pojednostavljuje integraciju sigurnosnih standarda, dok Spring Data JPA olakšava rad s bazama podataka.

Razvojna okruženja

Visual Studio Code

Visual Studio Code odabran je za frontend jer je lagan, prilagodljiv i ima odličnu podršku za TypeScript putem ekstenzija.

IntelliJ IDEA

IntelliJ IDEA koristi se za backend zbog naprednih značajki za razvoj u Javi, uključujući integraciju s alatima poput Mavena i Dockera. Njegova integrirana podrška za Spring Boot omogućuje brži razvoj i ispitivanje.

Git

Git je odabran za verzioniranje koda jer omogućuje praćenje promjena, suradnju na projektu i povratak na starije verzije koda.

Baza podataka

SQLite

SQLite je korišten jer je lagana baza podataka koja ne zahtijeva složenu instalaciju i pogodna je za lokalni razvoj manjih aplikacija. Njegova jednostavna integracija sa Spring Bootom dodatno ubrzava rad na projektu.

ERDPlus

ERDPlus se koristi za izradu ER dijagrama zbog intuitivnog sučelja i besplatnog pristupa.

Ispitivanje

Postman

Postman se koristi za ispitivanje REST API-ja jer omogućuje jednostavnu izradu i izvršavanje HTTP zahtjeva te analizu odgovora.

Selenium WebDriver

Selenium WebDriver odabran je za ispitivanje korisničkog sučelja jer omogućuje automatizaciju testiranja na različitim preglednicima i platformama.

Spring Boot starter test framework

Spring Boot test framework koristi se za unit-ispitivanje backend aplikacije jer omogućuje jednostavno ispitivanje REST API-ja i baze podataka. Uključuje najbolje biblioteke za ispitivanje kao što su JUnit, AssertJ i Mockito, čime se osigurava kvalitetan i robustan kod.

Komunikacija

Discord

Discord je korišten za komunikaciju tima jer omogućuje održavanje sastanaka, dijeljenje datoteka i organizaciju putem kanala.

Trello

Trello je korišten za praćenje zadataka i organizaciju projekta. Vizualno sučelje omogućuje timovima praćenje napretka i raspodjelu odgovornosti na intuitivan način.

Pogon

VPS

GalaxyGate

GalaxyGate se koristi kao VPS provider. Na njemu imamo server s Debian 12 operativnim sustavom.

TLS certifikat

Let's Encrypt

nginx

CertBot

Na serveru je pokrenut nginx reverse proxy koji je, s pomoću CertBota, namješten da koristi besplatni TLS certifikat od non-profit organizacije Let's Encrypt.

Deployment

lemon-dm

Docker

Na poslužitelju je pokrenuta naša vlastita aplikacija lemon-dm koja brzo može izgraditi i dockerizirati bilo koju javnu Git granu. S pomoću ove aplikacije je pokrenuta stranica koju vide korisnici, ali i koristi se i za testiranje aplikacije u produkcijskim uvjetima.

Domena

Domain.com

Domena "triolinog.space" je kupljena pomoću Domain.com.

Email

Zoho Mail

Zoho Mail hosta SMTP poslužitelja namještenog na našu domenu, što našoj aplikaciji daje mogućnost slanja poruka korisnicima, ali i nama daje mogućnost da imamo službene adrese za admine sustava.

Ispitivanje programskog rješenja

Ispitivanje komponenti

Ispitni slučaj 1 - Unos studenta/učitelja direktno u bazu podataka

```java
@Test  stjepanbonic +1
public void ShouldSaveTeacher() {
    Teacher teacher = new Teacher();
    teacher.setFullName("John Doe");

    Teacher savedTeacher = teacherRepository.save(teacher);
    savedTeacher = teacherRepository.findById(savedTeacher.getId()).get();

    assertThat(savedTeacher.getId()).isNotNull();
    assertThat(savedTeacher.getFullName()).isEqualTo( expected: "John Doe");
}

@Test  stjepanbonic +1
public void ShouldSaveStudent() {
    Student student = new Student();
    student.setFullName("Igor");

    Student savedStudent = studentRepository.save(student);
    savedStudent = studentRepository.findById(savedStudent.getId()).get();

    assertThat(savedStudent.getId()).isNotNull();
    assertThat(savedStudent.getFullName()).isEqualTo( expected: "Igor");
}
```

Slika 1a:

Kod za test unosa studenta/učitelja u bazu podataka

Slika 1b: Rezultat

Ispitni slučaj 2 - Registracija studenta

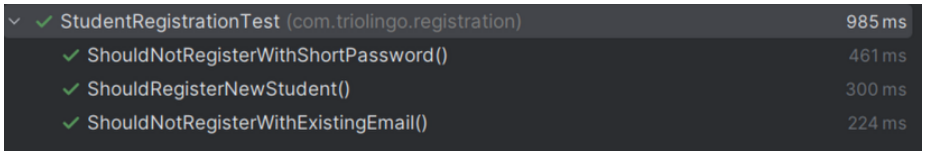
```java
@Test  Petar Stika
void ShouldRegisterNewStudent() {
    StudentCreateDTO studentCreateDTO = new StudentCreateDTO(
            email: "john.doe@gmail.com",
            password: "12345678",
            fullName: "John Doe",
            Map.of( k1: "Spanish", KnowledgeLevel.BEGINNER),
            TeachingStyle.FLEXIBLE,
            learningGoals: "Learn Espanol"
    );

    ResponseEntity<String> response = restTemplate.postForEntity( url: "/student/register", studentCreateDTO, String.class);
    assertThat(response.getStatusCode()).isEqualTo(HttpStatus.CREATED);
}
```

Slika 2a.1: Kod za uspješnu registraciju studenta

```java
@Test  Petar Stika
void ShouldNotRegisterWithExistingEmail() {
    StudentCreateDTO studentCreateDTO = new StudentCreateDTO(
            email: "john.doe@gmail.com",
            password: "12345678",
            fullName: "John Doe",
            Map.of( k1: "Spanish", KnowledgeLevel.BEGINNER),
            TeachingStyle.FLEXIBLE,
            learningGoals: "Learn Espanol"
    );

    ResponseEntity<String> response1 = restTemplate.postForEntity( url: "/student/register", studentCreateDTO, String.class);
    assertThat(response1.getStatusCode()).isEqualTo(HttpStatus.CREATED);

    ResponseEntity<String> response2 = restTemplate.postForEntity( url: "/student/register", studentCreateDTO, String.class);
    assertThat(response2.getStatusCode()).isEqualTo(HttpStatus.BAD_REQUEST);
}
```

```java
@Test  Petar Stika
void ShouldNotRegisterWithShortPassword() {
    StudentCreateDTO studentCreateDTO = new StudentCreateDTO(
            email: "john.doe@gmail.com",
            password: "1234",
            fullName: "John Doe",
            Map.of( k1: "Spanish", KnowledgeLevel.BEGINNER),
            TeachingStyle.FLEXIBLE,
            learningGoals: "Learn Espanol"
    );

    ResponseEntity<String> response = restTemplate.postForEntity( url: "/student/register", studentCreateDTO, String.class);
    assertThat(response.getStatusCode()).isEqualTo(HttpStatus.BAD_REQUEST);
}
```

Slika 2a.2 i 2a.3: Kod za neuspješnu registraciju studenta



Slika 2b: Rezultat

Ispitni slučaj 3 - Registracija učitelja

```
@Test  Petar Štika
void ShouldRegisterNewTeacher() {
    TeacherCreateDTO teacherCreateDTO  = new TeacherCreateDTO(
            email: "john.doe@gmail.com",
            password: "12345678",
            fullName: "John Doe",
            List.of("English", "Spanish"),
            yearsOfExperience: 5,
            qualifications: "PhD English",
            TeachingStyle.FLEXIBLE,
            profileImageHash: null,
            hourlyRate: 15.5
    );

    ResponseEntity<String> response = restTemplate.postForEntity( url: "/teacher/register", teacherCreateDTO, String.class);
    assertThat(response.getStatusCode()).isEqualTo(HttpStatus.CREATED);
}
```

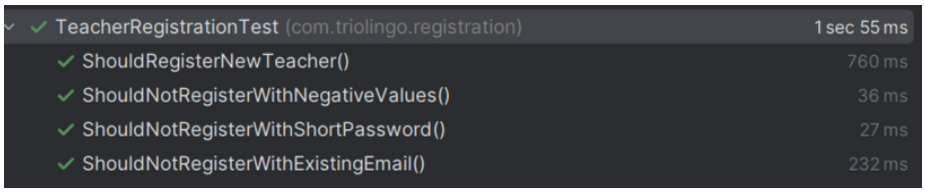
Slika 3a.1: Kod za uspješnu registraciju učitelja

```
@Test  Petar Štika
void ShouldNotRegisterWithExistingEmail() {
    TeacherCreateDTO teacherCreateDTO  = new TeacherCreateDTO(
            email: "john.doe@gmail.com",
            password: "12345678",
            fullName: "John Doe",
            List.of("English", "Spanish"),
            yearsOfExperience: 5,
            qualifications: "PhD English",
            TeachingStyle.FLEXIBLE,
            profileImageHash: null,
            hourlyRate: 15.5
    );

    ResponseEntity<String> response1 = restTemplate.postForEntity( url: "/teacher/register", teacherCreateDTO, String.class);
    assertThat(response1.getStatusCode()).isEqualTo(HttpStatus.CREATED);

    ResponseEntity<String> response2 = restTemplate.postForEntity( url: "/teacher/register", teacherCreateDTO, String.class);
    assertThat(response2.getStatusCode()).isEqualTo(HttpStatus.BAD_REQUEST);
}
```

```
@Test  Petar Štika
void ShouldNotRegisterWithShortPassword() {
    TeacherCreateDTO teacherCreateDTO  = new TeacherCreateDTO(
            email: "john.doe@gmail.com",
            password: "1234",
            fullName: "John Doe",
            List.of("English", "Spanish"),
            yearsOfExperience: 5,
            qualifications: "PhD English",
            TeachingStyle.FLEXIBLE,
            profileImageHash: null,
            hourlyRate: 15.5
    );

    ResponseEntity<String> response = restTemplate.postForEntity( url: "/teacher/register", teacherCreateDTO, String.class);
    assertThat(response.getStatusCode()).isEqualTo(HttpStatus.BAD_REQUEST);
}
```

```java
@Test  Petar Stika
void ShouldNotRegisterWithNegativeValues() {
    // Negative years of experience
    TeacherCreateDTO teacherCreateDTO1  = new TeacherCreateDTO(
            email: "john.doe@gmail.com",
            password: "12345678",
            fullName: "John Doe",
            List.of("English", "Spanish"),
            yearsOfExperience: -5,
            qualifications: "PhD English",
            TeachingStyle.FLEXIBLE,
            profileImageHash: null,
            hourlyRate: 15.5
    );

    ResponseEntity<String> response1 = restTemplate.postForEntity( url: "/teacher/register", teacherCreateDTO1, String.class);
    assertThat(response1.getStatusCode()).isEqualTo(HttpStatus.BAD_REQUEST);

    // Negative hourly rate
    TeacherCreateDTO teacherCreateDTO2  = new TeacherCreateDTO(
            email: "john.doe@gmail.com",
            password: "12345678",
            fullName: "John Doe",
            List.of("English", "Spanish"),
            yearsOfExperience: 5,
            qualifications: "PhD English",
            TeachingStyle.FLEXIBLE,
            profileImageHash: null,
            hourlyRate: -15.5
    );

    ResponseEntity<String> response2 = restTemplate.postForEntity( url: "/teacher/register", teacherCreateDTO2, String.class);
    assertThat(response2.getStatusCode()).isEqualTo(HttpStatus.BAD_REQUEST);
}
```

Slika 3a.2, 3a.3 i 3a.4: Kod za neuspješnu registraciju učitelja



Slika 3b: Rezultat

Ispitni slučaj 4 - Prijava administratora

```java
@Test  Petar Stika
void ShouldLoginAsAdmin() {
    MultiValueMap<String, String> userCredentials = new LinkedMultiValueMap<>();
    userCredentials.add("email", "admin@triolingo.space");
    userCredentials.add("password", "123456789");

    HttpHeaders headers = new HttpHeaders();
    headers.setContentType(MediaType.APPLICATION_FORM_URLENCODED);
    HttpEntity<MultiValueMap<String, String>> request = new HttpEntity<>(userCredentials, headers);
    ResponseEntity<String> response = restTemplate.postForEntity( url: "/login", request, String.class);
    assertThat(response.getStatusCode()).isEqualTo(HttpStatus.OK);
}
```

Slika 4a: Kod za uspješnu prijavu administratora

Slika 4b: Rezultat

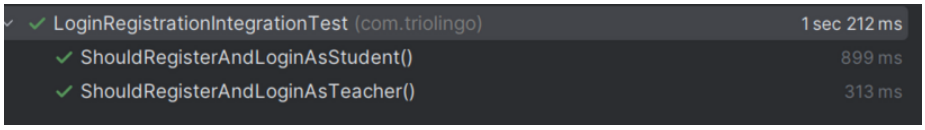
Ispitni slučaj 5 - Prijava studenta

```
@Test  Petar Štika
void ShouldRegisterAndLoginAsStudent() {
    // Registration part
    StudentCreateDTO studentCreateDTO = new StudentCreateDTO(
            email: "john.doe@gmail.com",
            password: "12345678",
            fullName: "John Doe",
            Map.of( k1: "Spanish", KnowledgeLevel.BEGINNER),
            TeachingStyle.FLEXIBLE,
            learningGoals: "Learn Espanol"
    );

    ResponseEntity<String> registerResponse = restTemplate.postForEntity( url: "/student/register", studentCreateDTO, String.class);
    assertThat(registerResponse.getStatusCode()).isEqualTo(HttpStatus.CREATED);

    // Login part
    MultiValueMap<String, String> userCredentials = new LinkedMultiValueMap<>();
    userCredentials.add("email", "john.doe@gmail.com");
    userCredentials.add("password", "12345678");

    HttpHeaders headers = new HttpHeaders();
    headers.setContentType(MediaType.APPLICATION_FORM_URLENCODED);
    HttpEntity<MultiValueMap<String, String>> request = new HttpEntity<>(userCredentials, headers);

    ResponseEntity<String> loginResponse = restTemplate.postForEntity( url: "/login", request, String.class);
    assertThat(loginResponse.getStatusCode()).isEqualTo(HttpStatus.OK);
}
```

Slika 5a.1: Kod za uspješnu registraciju i prijavu studenta

```
@Test  Petar Štika
void ShouldRegisterAndLoginAsTeacher() {
    // Registration part
    TeacherCreateDTO teacherCreateDTO  = new TeacherCreateDTO(
            email: "john.doe@gmail.com",
            password: "12345678",
            fullName: "John Doe",
            List.of("English", "Spanish"),
            yearsOfExperience: 5,
            qualifications: "PhD English",
            TeachingStyle.FLEXIBLE,
            profileImageHash: null,
            hourlyRate: 15.5
    );

    ResponseEntity<String> registerResponse = restTemplate.postForEntity( url: "/teacher/register", teacherCreateDTO, String.class);
    assertThat(registerResponse.getStatusCode()).isEqualTo(HttpStatus.CREATED);

    // Login part
    MultiValueMap<String, String> userCredentials = new LinkedMultiValueMap<>();
    userCredentials.add("email", "john.doe@gmail.com");
    userCredentials.add("password", "12345678");

    HttpHeaders headers = new HttpHeaders();
    headers.setContentType(MediaType.APPLICATION_FORM_URLENCODED);
    HttpEntity<MultiValueMap<String, String>> request = new HttpEntity<>(userCredentials, headers);

    ResponseEntity<String> loginResponse = restTemplate.postForEntity( url: "/login", request, String.class);
    assertThat(loginResponse.getStatusCode()).isEqualTo(HttpStatus.OK);
}
```

Slika 5a.2: Kod za uspješnu registraciju i prijavu učitelja



Slika 5b: Rezultat

Ispitivanje sustava

Prvi ispitni slučaj - prijava i promjena lozinke

1. Ulazi:

- podaci testnog korisnika (ucenik123@gmail.com, lozinka!)

2. Koraci ispitivanja:

- Otvoriti aplikaciju
- Otvoriti login stranicu
- Upisati email
- Upisati lozinku
- Kliknuti login gumb
- Otvoriti stranicu profila
- Otvoriti dijalog za promjenu lozinke
- Upisati staru lozinku
- Upisati novu lozinku
- Potvrditi novu lozinku
- Kliknuti gumb za promjenu lozinke

3. Očekivani rezultat:

- Korisnik je vraćen na stranicu profila s promijenjenom lozinkom

4. Dobiveni rezultat:

```
Otvoren login page
Upisan email
Upisana lozinka
Korisnik ulogiran
Otvorena stranica profila
Otvoren dijalog za promjenu lozinke
Upisana stara lozinka ('lozinka!')
Upisana nova lozinka ('lozinka!!')
Potvrdena nova lozinka ('lozinka!!')
Promijenjena lozinka
Korisnik vracen na stranicu profila, test uspjesan
```

Drugi ispitni slučaj - prijava i objavljivanje termina lekcije

1. Ulazi:

- podaci testnog korisnika (ucitelj123@gmail.com, lozinka!)

2. Koraci ispitivanja:

- Otvoriti aplikaciju
- Otvoriti login stranicu
- Upisati email
- Upisati lozinku
- Kliknuti login gumb
- Otvoriti stranicu kalendara
- Odabrati vrijeme početka lekcije
- Odabrati vrijeme kraja lekcije
- Odabrati jezik koji će se podučavati

3. Očekivani rezultat:

- Učitelj je vraćen na stranicu kalendara s objavljenim terminom lekcije

4. Dobiveni rezultat:

```
Otvoren login page
Upisan email
Upisana lozinka
Ucitelj ulogiran
Otvorena stranica kalendara
Otvoren kalendar za pocetak lekcije
Postavljen datum pocetka lekcije na 31.1.2025.
Postavljeno vrijeme pocetka lekcije na 10:00
Otvoren kalendar za kraj lekcije
Postavljen datum kraja lekcije na 31.1.2025.
Postavljeno vrijeme pocetka lekcije na 11:00
Otvoren selektor jezika
Odabran japanski jezik
Objavljen termin lekcije, test uspjesan
```

Treći ispitni slučaj - slanje zahtjeva za termin lekcije

1. Ulazi:

- podaci testnog korisnika (ucenik123@gmail.com, lozinka!)

2. Koraci ispitivanja:

- Otvoriti aplikaciju
- Otvoriti login stranicu
- Upisati email
- Upisati lozinku
- Kliknuti login gumb
- Otvoriti stranicu učitelja

- Otvoriti listu dostupnih termina za lekciju tog učitelja
- Poslati zahtjev za lekciju

3. Očekivani rezultat:

- Pošalje se zahtjev za lekciju

4. Dobiveni rezultat:

```
Otvoren login page
Upisan email
Upisana lozinka
Korisnik ulogiran
Otvorena stranica ucitelja
Otvorena lista dostupnih termina za lekciju tog ucitelja
Poslan zahtjev za lekciju, test uspjesan
```

Četvrti ispitni slučaj - prihvaćanje zahtjeva za lekciju

1. Ulazi:

- podaci testnog korisnika (ucitelj123@gmail.com, lozinka!)

2. Koraci ispitivanja:

- Otvoriti aplikaciju
- Otvoriti login stranicu
- Upisati email
- Upisati lozinku
- Kliknuti login gumb
- Otvoriti popis zahtjeva za lekciju
- Prihvatiti zahtjev za lekciju

3. Očekivani rezultat:

- Korisnik ostaje na stranici zahtjeva za lekciju, taj zahtjev se obriše

4. Dobiveni rezultat:

```
Otvoren login page
Upisan email
Upisana lozinka
Ucitelj ulogiran
Otvoren popis zahtjeva za lekciju
Prihvacen zahtjev za lekciju
```

Peti ispitni slučaj - ostavljanje recenzije

1. Ulazi:

- podaci testnog korisnika (ucenik123@gmail.com, lozinka!)

2. Koraci ispitivanja:

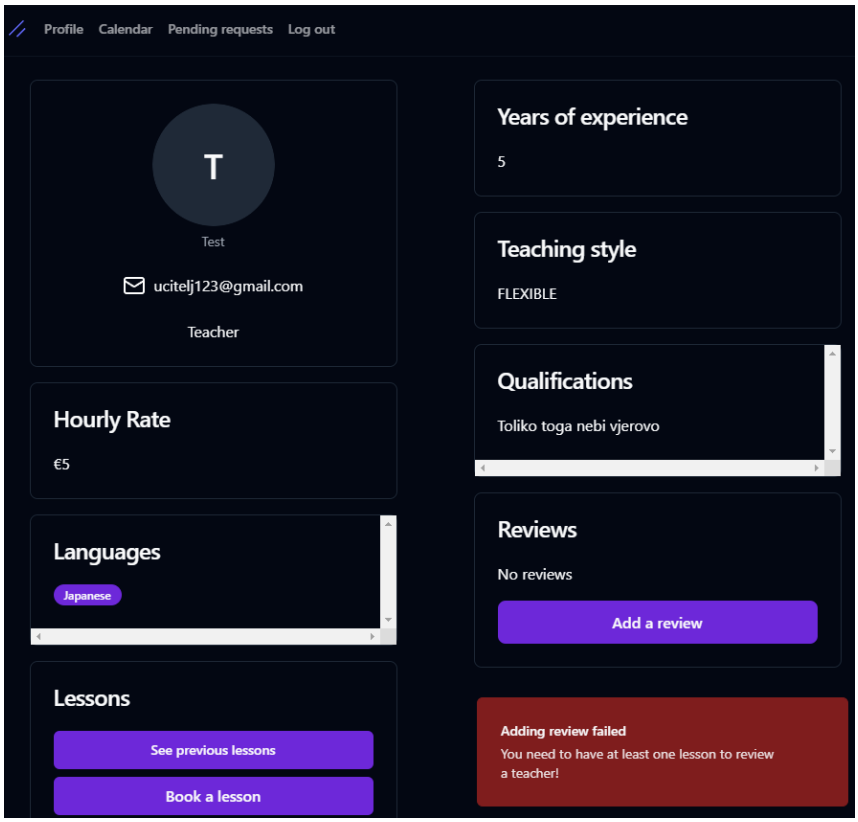
- Otvoriti aplikaciju
- Otvoriti login stranicu
- Upisati email
- Upisati lozinku
- Kliknuti login gumb
- Otvoriti stranicu učitelja
- Otvoriti dijalog za ostavljanje recenzije
- Napisati recenziju
- Predati recenziju

3. Očekivani rezultat:

- Korisnik ostaje na stranici učitelja, a recenzija je evidentirana

4. Dobiveni rezultati:

```
Otvoren login page
Upisan email
Upisana lozinka
Korisnik ulogiran
Otvorena stranica ucitelja
Otvoren dijalog za ostavljanje recenzije
Napisana recenzija ucitelja
Test neuspjesan, nije obavljena niti jedna lekcija izmedu tog ucenika i ucitelja
```



Šesti ispitni slučaj - ostavljanje recenzije nakon provedene lekcije

1. Ulazi:

- podaci testnog korisnika (ucenik123@gmail.com, lozinka!)

2. Koraci ispitivanja:

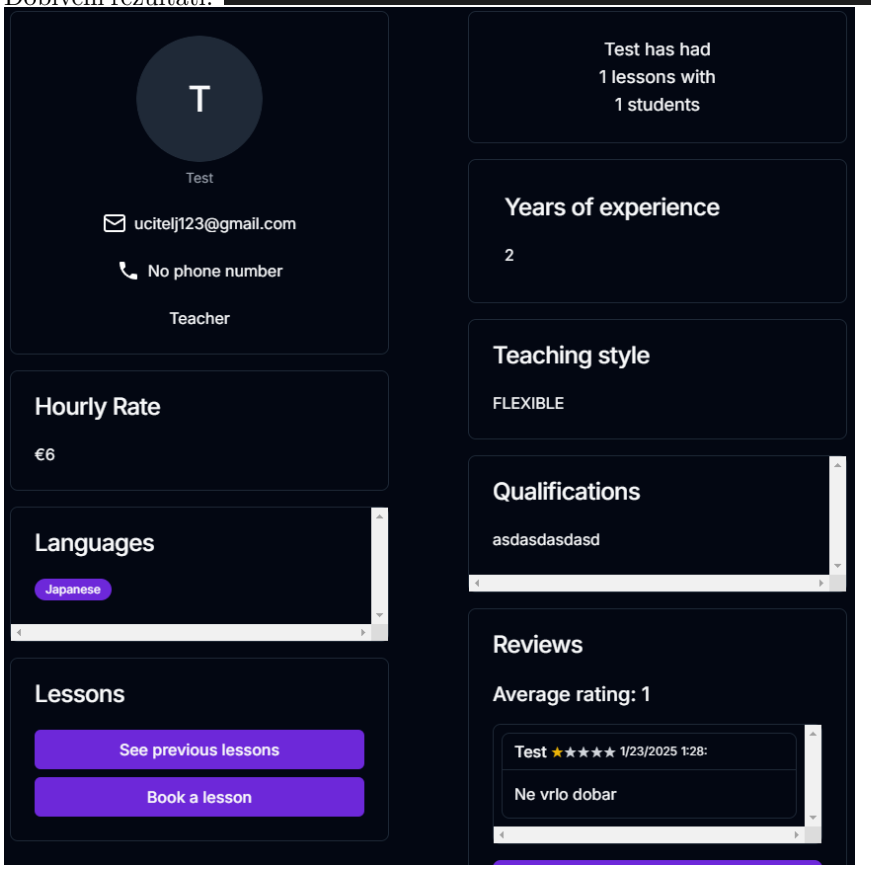
- Otvoriti aplikaciju
- Otvoriti login stranicu
- Upisati email
- Upisati lozinku
- Kliknuti login gumb
- Otvoriti stranicu učitelja
- Otvoriti dijalog za ostavljanje recenzije
- Napisati recenziju
- Predati recenziju

3. Očekivani rezultat:

- Korisnik ostaje na stranici učitelja, a recenzija je evidentirana

4. Dobiveni rezultati:

```
Otvoren login page
Upisan email
Upisana lozinka
Korisnik ulogiran
Otvorena stranica ucitelja
Otvoren dijalog za ostavljanje recenzije
Napisana recenzija ucitelja
Recenzija je ostavljena, test uspjesan
```



Dijagram razmještaja

Dijagram razmještaja prikazuje strukturu sklopovlja i programsku potporu koja se koristi u implementaciji sustava unutar radnog okruženja. Klijenti pristupaju sustavu koristeći web preglednike, komunicirajući s web poslužiteljem preko HTTPS veze. Na poslužiteljskom računalu implementirani su nginx kao web poslužitelj i Docker kontejneri za frontend i backend dijelove aplikacije, uključu-

jući bazu podataka koja omogućava pohranu i dohvat podataka. Arhitektura sustava temelji se na modelu „klijent-poslužitelj“.

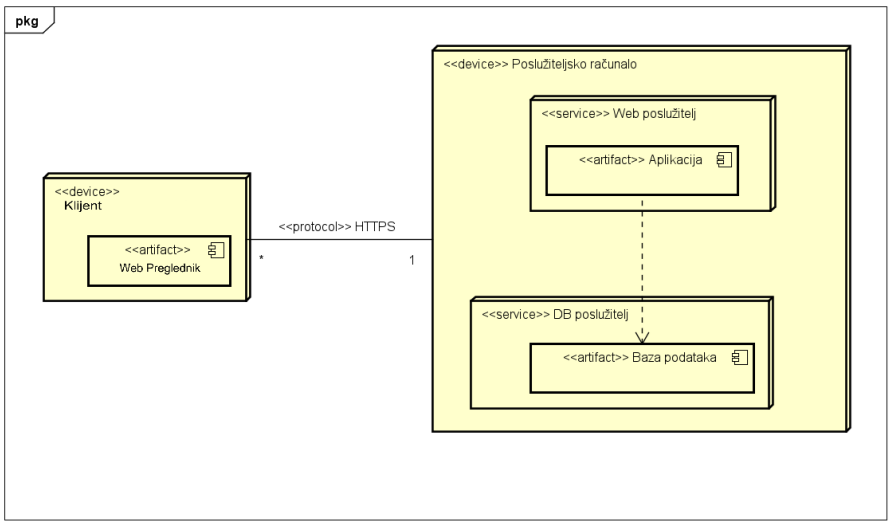


Figure 4: Dijagram razmještaja

Upute za puštanje u pogon

Puštanje u pogon s pomoću lemon-dm aplikacije

Ove upute pretpostavljaju da korisnik ima server s nekom verzijom Linuxa, registriranu domenu, dostupan SMTP poslužitelj te registriranu OAuth GitHub aplikaciju.

Postavljanje lemon-dm aplikacije na serveru

lemon-dm aplikacija može se instalirati samo preko sourcea na GitHub-u, tako da je najbolji način za dobiti aplikaciju klonirati git repozitorij na server s pomoću SSH konekcije. U lemon-dm aplikaciji nalazi se “projects.json” file, njega je potrebno populirati svim “projektima” koje želite pokrenuti. Primjer se nalazi u

```
{
        "docker-test": {
                "name": "docker-test",
                "gitUrl": "git@github.com:LeoTheMoldyLemon/docker-test.git",
                "dockerfilePath": "."
        },
        "triolingo-frontend": {
                "name": "triolingo-frontend",
                "gitUrl": "git@github.com:Progi-Petrovi/triolingo.git",
                "dockerfilePath": "./frontend/deploy"
        },
        "triolingo-backend": {
                "name": "triolingo-backend",
                "gitUrl": "git@github.com:Progi-Petrovi/triolingo.git",
                "dockerfilePath": "./backend/deploy"
        }
}
```

nastavku.

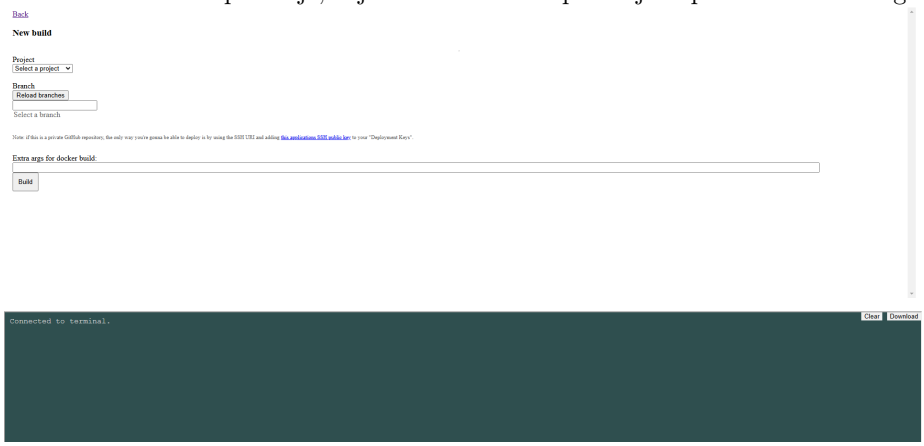
lemon-dm je node.js aplikacija, te ju kao takvu možete pokrenuti na mnogo načina. Na našem serveru aplikacija je pokrenuta s pomoću PM2 process managera.

Zatim, u web pregledniku, posjetite port 5001 vašeg servera. Ako se lemon-dm

pravilno pokrenuo, ovo je što biste trebali vidjeti:

Kreiranje builda za frontend i backend aplikacije

Pritiskom na link “Create new build” korisnik pred sobom vidi form kojim može kreirati novi build aplikacije, kojim će lemon-dm aplikacija napraviti docker image.

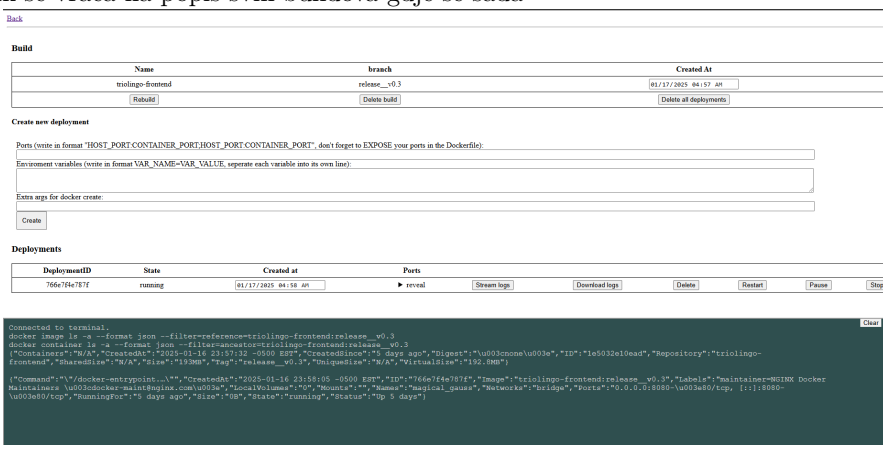


Kako bismo kreirali frontend build potrebno je odabrati “triolingo-frontend”(ili kako ste ga nazvali prilikom popunjavanja datoteke “projects.json”) pod “Project” te željenu verziju aplikacije pod “Branch”. Ako su novi branch-evi stvoreni, može se reload-ati ova stranica, ili pritisnuti “Reload branches”. Za frontend build, pod “Extra args for docker build:” potrebno je staviti `--build-arg VITE_API_URL="https://{DOMAIN}/api/"` gdje `{DOMAIN}` zamjenjujete s domenom koju imate registriranu. Nakon pritiska na “Build” tipku čekati završetak build-a. Napredak te potencijalne greške moguće je pratiti na danom terminalu.

Postupak ponoviti za backend, ali ostaviti “Extra args for docker build:” polje prazno.

Pokretanje build-ova

Pritiskom na tipku “Back”, korisnik se vraća na popis svih buildova gdje se sada



nalaze dva upravo izrađena builda.

Pritiskom na “deployments” link uz “triolingo-frontend” dolazimo do stranice za pokretanje navedenog builda. Tu je dovoljno pod “Ports” staviti `8080:80` te pritisnuti “Create”. Nakon izrade docker containera (koja se može pratiti u terminalu) pojavit će se nova stavka u “Deployments” tablici. Pritiskom na tipku “Start” docker container je pokrenut.

Isti ovaj postupak može se ponoviti za backend, jedino je potrebno u “Ports” umjesto `8080:80` staviti `5000:5000` te pod “Environment variables” staviti

```
cors.allowedOrigin=https://*.{DOMAIN}, https://{DOMAIN}, https://www.{DOMAIN}
path.frontend.base=https://{DOMAIN}
redirect-uri=https://{DOMAIN}/api/login/oauth2/code/github

spring.security.oauth2.client.registration.github.clientSecret={GITHUB_CLIENT_SECRET}
spring.security.oauth2.client.registration.github.clientId={GITHUB_CLIENT_ID}

email.host={EMAIL_HOST}
email.port={EMAIL_PORT}
email.user={EMAIL_USER}
email.password={EMAIL_PASSWORD}
```

s time da sve u vitičastim zagradama treba zamijeniti vlastitim vrijednostima.

Nakon uspješnih pokretanja, aplikacija bi trebala biti dostupna na postavljenoj domeni.

Zaključak i budući rad

Zadatak naše grupe bio je razviti web-aplikaciju koja učenicima i učiteljima omogućava učenje i poučavanje stranih jezika. Učenici biraju učitelja u jednom

od dostupnih termina u kalendaru, dok učitelji prihvaćaju ili odbijaju takve zahtjeve. Također smo omogućili dvosmjernu komunikaciju putem chata i prijavu putem OAuth autentifikacije.

Naš je tim imao sedam članova (Stjepan Bonić, Borna Gojšić, Petar Knežević, Josip Skledar, Leonardo Šimunović, Petar Štika i Karla Lučić) koji su surađivali tijekom zimskog semestra akademske godine 2024./2025. Iako su pojedini članovi posjedovali duboko znanje o razvoju aplikacija u svojem području, nitko nije imao iskustvo u timskom razvoju projekata. Zbog toga smo stavili veliki naglasak na upravljanje projektom i komunikaciju unutar grupe.

U prvoj smo fazi definirali opis projektnog zadatka, skupili zahtjeve i pripremili ostalu dokumentaciju potrebnu za razvoj aplikacije. Za organizaciju zadataka upotrijebili smo Trello, a za brzu komunikaciju Discord. Svaka se linija koda postavljala putem pull requesta na GitHub, a barem jedan član tima je pregledao i odobrio ili odbio promjene, što je osiguralo kvalitetu i konzistentnost koda. Uz to, razvili smo osnovne funkcionalnosti aplikacije, dizajn korisničkog sučelja i jednostavno pogon aplikacije na serveru.

U drugoj fazi dodali smo zahtjevnije funkcionalnosti poput chata, kalendara i sličnih mogućnosti. Temeljito smo ih ispitali pomoću ispitnih slučajeva na razini komponenti i cijeloga sustava.

Na kraju smatramo da smo uspješno razvili našu aplikaciju i dobili odlično iskustvo o timskom razvoju projekata, upravljanju projektom i međusobnoj komunikaciji. Ostvarili smo sve prvotno planirane funkcionalnosti, a uveli smo i neke dodatne. Budući rad mogao bi uključivati izradu mobilne verzije, integraciju sustava plaćanja unutar aplikacije te više opcija za OAuth autentifikaciju.

Tijekom rada susreli smo se s nekoliko tehničkih izazova, poput integracije chata, verifikacije putem e-pošte i OAuth autentifikacije, za koje nismo imali dovoljno iskustva. Ipak, uz dokumentaciju i online primjere uspješno smo riješili sve probleme. Projekt je bio korisno iskustvo za svakog člana tima jer smo naučili i razmijenili brojne vještine iz područja razvoja web-aplikacija i vođenja dugoročnog projekta. Iako smo zadovoljni rezultatima, prepoznajemo mogućnosti za poboljšanje poput: naglasak na ranije testiranje, realističniji planiranje vremena i češćem održavanju sastanaka.

Popis literature

1. Programsko inženjerstvo, FER ZEMRIS, http://www.fer.hr/predmet/proinz
2. I. Sommerville, "Software engineering", 8th ed, Addison Wesley, 2007.
3. T.C.Lethbridge, R.Langaniere, "Object-Oriented Software Engineering", 2nd ed. McGraw-Hill, 2005.
4. I. Marsic, Software engineering book", Department of Electrical and Computer Engineering, Rutgers University, http://www.ece.rutgers.edu/~marsic/books/SE

5. The Unified Modeling Language, https://www.uml-diagrams.org/
6. Astah Community, http://astah.net/editions/uml-new
7. TypeScript dokumentacija, https://www.typescriptlang.org/docs/
8. Java 21 dokumentacija, https://docs.oracle.com/en/java/javase/21/docs/api/index.html
9. React 18.3 dokumentacija, https://react.dev/reference/react
10. Vite dokumentacija, https://vite.dev/guide/
11. Spring Boot dokumentacija, https://docs.spring.io/spring-boot/documentation.html
12. Git dokumentacija, https://git-scm.com/doc
13. SQLite dokumentacija, https://www.sqlite.org/docs.html
14. Postman dokumentacija, https://learning.postman.com/docs/getting-started/introduction/
15. Selenium dokumentacija, https://www.selenium.dev/documentation/en/
16. Spring Boot test framework dokumentacija, https://docs.spring.io/spring-boot/docs/2.0.4.RELEASE/reference/html/boot-features-testing.html
17. Docker dokumentacija, https://docs.docker.com/

Dnevnik sastajanja

Kontinuirano osvježavanje

1. Sastanak

- Datum: 18. listopada 2024.
- Prisustvovali: S.Bonić, B.Gojšić, P.Knežević, J.Skledar, L.Šimunović, P.Štika, K.Lučić
- Teme sastanka:
  - upoznavanje
  - podjela zadataka

2. Sastanak

- Datum: 22. listopada 2024.
- Prisustvovali: S.Bonić, B.Gojšić, P.Knežević, J.Skledar, L.Šimunović, P.Štika, K.Lučić
- Teme sastanka:
  - UML dijagram za “BecomeFluent”

3. Sastanak

- Datum: 9. studenoga 2024.
- Prisustvovali: S.Bonić, B.Gojšić, P.Knežević, J.Skledar, L.Šimunović, P.Štika, K.Lučić
- Teme sastanka
  - Pregled odrađenog
  - Dogovor strukture autentifikacijskog sustava
  - Određivanje hijerarhije entiteta
  - Planiranje nastavka

4. Sastanak

- Datum: 10. siječnja 2025.
- Prisustvovali: S.Bonić, B.Gojšić, P.Knežević, J.Skledar, L.Šimunović, P.Štika, K.Lučić
- Teme sastanka
  - Pregled odrađenog
  - Podjela nedovršenih zadataka za alfa verziju
  - Planiranje nastavka

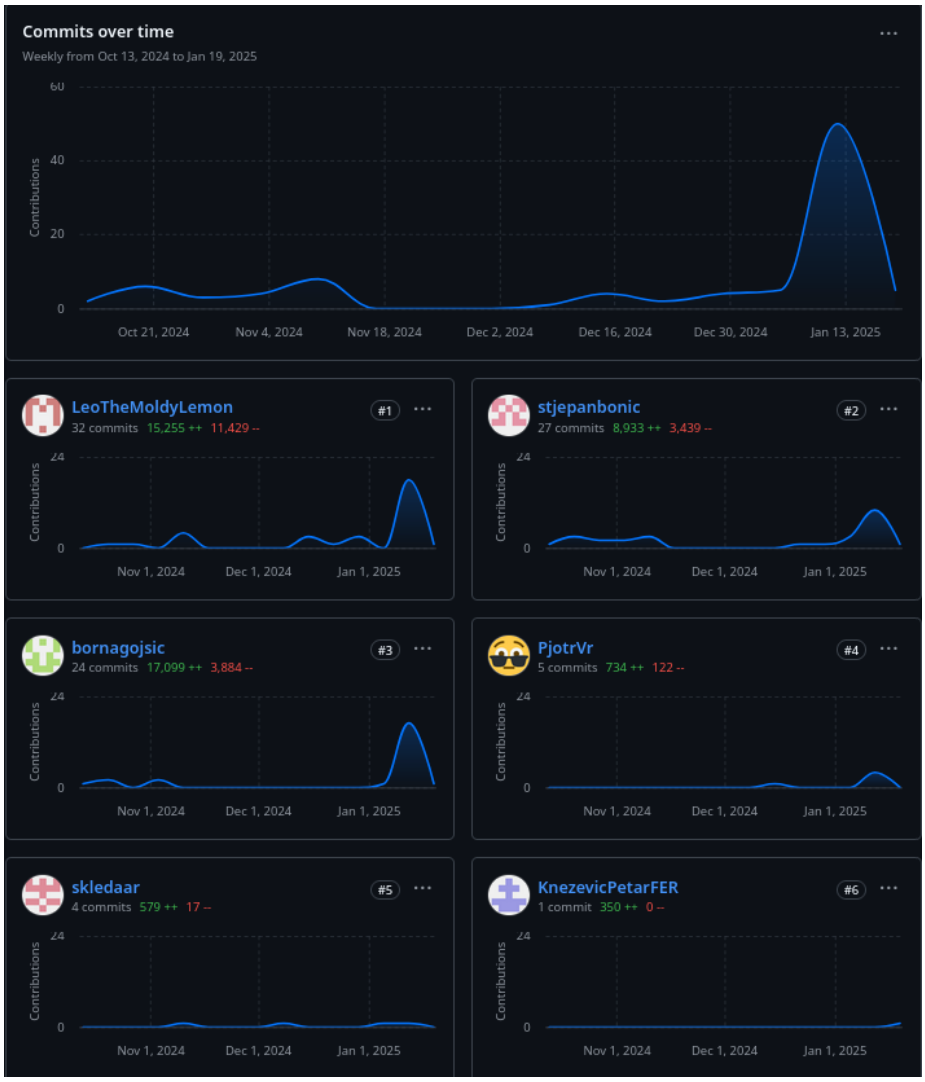
Tablica aktivnosti

Kontinuirano osvježavanje

Aktivnost	Petar Štika	Petar Knežević	Stjepan Bonić	Leonardo Šimunović	Borna Gojšić	Karla Lučić	Josip Skledar
Upravljanje projektom	5	5	5	5	5	4	4
Opis projektnog zadatka	1	3	1	0.5	1	0.5	1
Funkcionalni zahtjevi	4	3	2	1	2	1	4
Opis pojedinih obrazaca	1.5	2	1	1	1	1	2
Dijagram obrazaca uporabe	3					4	2
Sekvencijski dijagrami	4						2
Opis ostalih zahtjeva	3	3			1	1	
Arhitektura i dizajn sustava		4	5	3	2	6	
Baza podataka		3		2		4	
Dijagram razreda		2				6	
Dijagram stanja		2				3	
Dijagram aktivnosti		2				3	
Dijagram komponenti		2				4	

Aktivnost	Petar Štika	Petar Knežević	Stjepan Bonić	Leonardo Šimunović	Borna Gojšić	Karla Lučić	Josip Skledar
Korištene tehnologije i alati	2			2			
Ispitivanje programskog rješenja	4	8					
Dijagram razmještaja	3					3	
Upute za puštanje u pogon				1			
Dnevnik sastajanja	0.5						
Zaključak i budući rad	0.5						
Popis literature	1						
Dizajn pojedinih stranica			10	2	10		
Izrada baze podataka			3	6			
Backend	8	8	25	80	5		6
Frontend	3		30	8	35		12
Puštanje u pogon				10			

Dijagram pregleda promjena



Dnevnik promjena dokumentacije

Rev.	Opis promjene/dodatka	Autori	Datum
0.1	Napravljen predložak	Leonardo Šimunović	25.10.2024
0.2	Dodan footer za sranice	Leonardo Šimunović	25.10.2024

Rev.	Opis promjene/dodatka	Autori	Datum
0.3	Dodani funkcionalne zahtjeve	Josip Skledar	27. 10. 2024
0.4	Dodani obrasce uporabe i sekvencijski dijagram	Josip Skledar	27. 10. 2024
0.5	Dodan opis projekta i analiza zahtjeva	Petar Štika	27. 10. 2024
0.6	Promjena rasporeda stranica	Petar Štika	27. 10. 2024
0.7	Ispravak identacije	Josip Skledar	27. 10. 2024
0.8	Dodan opis arhitekture	Petar Knežević	28. 10. 2024
0.9	Ispravak gramatike	Karla Lučić	12. 11. 2024
1.1	Dodan opis baze podataka	Karla Lučić	13. 11. 2024, 14. 11. 2024 i 15. 11. 2024
1.2	Dodani novi funkcionalni i nefunkcionalni zahtjevi	Petar Štika	12. 12. 2024
1.3	Dodani novi sekvencijski dijagrami	Petar Štika	13. 12. 2024
1.4	Dodani novi dijagrami razreda	Karla Lučić	16. 1. 2025
1.5	Dodan dijagram razmještaja	Petar Štika	16. 1. 2025
1.6	Dodane korištene tehnologije i alati	Petar Štika	16. 1. 2025
1.7	Dodani ispitni slučajevi sustava	Petar Knežević	16. 1. 2025
1.8	Dodani dijagrami stanja i aktivnosti	Karla Lučić	17. 1. 2025
1.9	Dodani ispitni slučajevi komponenti	Petar Štika	17. 1. 2025
1.10	Dodan dijagram komponenti	Karla Lučić	17. 1. 2025
1.11	Dodane tehnologije za pogon	Leonardo Šimunović	22. 1. 2025
1.12	Dodane upute za puštanje u pogon	Leonardo Šimunović	22. 1. 2025
1.13	Ispravak dijagrama razmještaja	Leonardo Šimunović	22. 1. 2025
1.14	Dodani dodatni ispitni slučajevi sustava	Petar Knežević	23. 1. 2025
1.15	Ispravak gramatike	Karla Lučić	23. 1. 2025