StanBlog

Programsko inženjerstvo ak.god 2024./2025.

2. Revizija

Grupa: *Project Bajeet*

Tim:

- Jelena Glasovac
- Mauro Mohorovičić
- Lovre Rančev
- Bruno Ševčenko
- Bruno Tudor
- Marko Vukadin

Nastavnik: Goran Rajić

1. Opis projektnog zadatka

Glavni cilj aplikacije *StanBlog* je poboljšati komunikaciju i koordinaciju između suvlasnika stambenih zgrada, te omogućiti učinkovitiji način upravljanja zajedničkim aktivnostima i resursima unutar zgrade. Aplikacija će služiti kao centralizirana platforma putem koje će suvlasnici moći predlagati inicijative, raspravljati o relevantnim temama, organizirati glasanja i dogovarati sastanke s ciljem donošenja odluka koje su u interesu svih korisnika zgrade. Ujedno, aplikacija će omogućiti lakše praćenje statusa dogovorenih aktivnosti.

Vlasništvo u zajedničkim stambenim jedinicama često podrazumijeva probleme u koordinaciji i upravljanju zajedničkim sredstvima i troškovima, što dovodi do nesuglasica i zastoja u provođenju ključnih aktivnosti. Novi zakon o upravljanju zgradama dodatno propisuje procedure koje uključuju veću interakciju među sudionicima procesa upravljanja zgradom, što naglašava potrebu za digitalnim rješenjem.

1.1 Korisnici aplikacije

Korisnik aplikacije može biti **suvlasnik**, **predstavnik suvlasnika** ili **administrator**. Korisnici se prijavljuju pritiskom na neki od ponuđenih načina prijave:

U slučaju da se korisnik prijavi putem nekog od ponuđenih servisa, ali nije evidentiran kao korisnik sustava, može poslati zahtjev administratoru za unošenje u sustav. Na administratoru ostaje da korisnika prihvati i dodjeli odgovarajućem stanu, ili da odbije zahtjev.

Drugi način prijave je ostvaren na način da administrator samostalno upiše podatke korisnika, pri čemu stvara lokalni račun (koji nije vezan uz OAuth 2.0 vjerodajnice). Za takvu vrstu računa korisnik mora prvu prijavu izvršiti pomoću jednokratne lozinke koju dobiva od administratora. Lozinku je potrebno promijeniti, zbog čega će pri prvoj prijavi novi korisnik vidjeti sučelje za unos nove lozinke.

Korisničke lozinke su pohranjene su šifrirano koristeći Bcrypt algoritam i kao takve nisu ni na koji način vidljive administratoru ili nekom trećem licu. Iz tog razloga obavezno je jednokratnu lozinku resetirati budući da se šalje putem nesigurnog medija (e-pošta).

1.1.1 Suvlasnik i predstavnik suvlasnika

Uloge suvlasnika i predstavnika suvlasnika su funkcionalno gotovo jednake. Glavna razlika predstavnika suvlasnika u odnosu na običnog suvlasnika jest da predstavnik može inicirati službeni sastanak kada su ispunjeni uvjeti glasanja.

Suvlasnici mogu na oglasnoj ploči aplikacije započeti javnu ili privatnu diskusiju pritiskom na +, što im otvara izbornik za unos naslova, sadržaja te postavljanja određenih ograničenja.

Pod ograničenja spadaju:

- tip diskusije: **javna** ili **privatna**

- maksimalni broj odgovora po suvlasniku
- zabrana sudjelovanja određenih suvlasnika
- vidljivost (ako se radi o privatnoj diskusiji).

Javna diskusija i njezin sadržaj vidljivi su svima, ali je sudjelovanje suvlasnika ograničeno ovisno o postavkama diskusije.

Privatna diskusija vidljiva je svima, ali je njezin sadržaj dostupan samo odabranim korisnicima.

Dodavanjem suvlasnika u listu sudionika automatski se šalje elektronička pošta suvlasniku kako bi bio informiran. U primjerku e-pošte nalaze se naslov i sadržaj objave.

U objavljenoj diskusiji na oglasnoj ploči vidljivi su:

- datum i vrijeme objave
- naslov i sadržaj diskusije
- aktivna tema glasanja, opcije i broj odgovora na glasanje *(ako postoji)*.

Diskusija se može naknadno urediti pritiskom na *Uredi*, gdje je moguće urediti sve ranije navedene parametre. Ako se u diskusiji uređivanjem promijene naslov ili sadržaj, diskusija će biti označena datumom i vremenom zadnjeg uređivanja.

Sudionici u diskusiji sudjeluju pomoću komentara. Ako je suvlasniku dozvoljeno sudjelovanje u nekoj diskusiji, može objaviti komentar pod tu diskusiju. Komentar može uključivati i glasanje, koje se sastoji od pitanja te opcije potvrdnog i negativnog odgovora. Uz opcije odgovora vidljiv je i broj odgovora na određenu opciju, kao i postotak odgovora na pojedinu opciju u odnosu na ukupni broj odgovora.

Rezultat glasanja ažurira se u stvarnom vremenu, sa svakim novim glasom. Podaci o glasanju prikazani su anonimno, čime se osigurava neutralnost u donošenju odluka i sprječava pritisak na korisnike.

Ako broj pozitivnih odgovora premašuje četvrtinu svih odgovora, predstavnik suvlasnika može kreirati poziv na sastanak. Sastanak se ostvaruje pomoću aplikacije *StanPlan*, a pri stvaranju sastanka potrebno je unijeti temu, termin i tekst poziva na sastanak. Pritiskom na *Stvori* aplikacija šalje unesene podatke u odgovarajućem formatu na *StanPlan* server.

Suvlasnici i predstavnik suvlasnika pritiskom na ikonu osobe dobivaju dvije opcije:

- **Osobni podaci**, gdje mogu pregledati svoje osobne podatke i postaviti novu lozinku
- **Odjava**, pri čijem pritisku se korisnik odjavljuje iz sustava i preusmjerava na početnu stranicu za prijavu.

1.1.2 Administrator

Administrator je odgovoran za vođenje i održavanje sustava te ima potpunu kontrolu nad korisničkim pristupima. Administrator dodaje i uklanja nove korisnike, odobrava zahtjeve, uređuje podatke postojećih korisnika (poput adrese e-pošte ili prezimena), konfigurira OAuth 2.0 usluge i konfigurira integraciju sa StanPlan poslužiteljem.

Pritiskom na ikonu zupčanika u gornjem desnom kutu ekrana (dostupno samo administratoru), administrator dobiva pristup svojem administratorskom panelu.

U odjeljku *Korisnici* nalazi se popis svih registriranih korisnika, čije podatke može urediti pritiskom na ikonu olovke pored imena određenog korisnika. U ovom odjeljku mogu se i dodati novi lokalni korisnici (bez vezanih OAuth 2.0 vjerodajnica). Pritiskom na *Dodaj* otvara se izbornik u kojem je potrebno unijeti osobne podatke korisnika i adresu e-pošte, nakon čega se generira nasumična jednokratna lozinka koja se šalje na e-poštu novog korisnika.

U odjeljku *Zahtjevi* nalazi se popis svih zahtjeva za registraciju na čekanju. Pritiskom na *Odbij*, zahtjev neregistriranog korisnika se odbija (ako se npr. radi o nekome tko nije uopće stanar zgrade). Pritiskom na *Prihvati*, otvara se izbornik u kojem je potrebno unijeti osobne podatke, dodijeliti stan i ulogu novom suvlasniku. Nakon prihvaćanja zahtjeva, korisnik će biti obavješten o mogućnosti prijave na sustav.

U odjeljku *Postavke* nalaze se opcije za konfiguraciju OAuth 2.0 usluga, kao i polje za unos adrese StanPlan poslužitelja.

1.2 Moguće nadogradnje

- **Uvođenje mobilne verzije:** radi praktičnosti, gdje bi korisnici mogli sudjelovati u diskusijama i glasanjima putem mobilnih uređaja.
- **Napredna analitika:** za praćenje učestalosti glasanja i statističke podatke o sudjelovanju.
- **Integracija s financijskim modulom:** za detaljno praćenje stanja pričuve i potrošnje, što bi bilo dodatno korisno za predstavnike i suvlasnike.

2. Analiza zahtjeva

2.1 Funkcionalni zahtjevi

ID zahtjeva	Opis	Prioritet	Izvor	Kriteriji prihvaćanja
F-001	Aplikacija omogućuje neprijavljenom korisniku prijavu u sustav koristeći korisničko ime i lozinku ili OAuth 2.0	Visok	Zahtjev dionika	Neprijavljeni korisnik može se uspješno prijaviti u sustav koristeći odabrani način autentifikacije.
F-002	Aplikacija omogućuje suvlasniku pokretanje javne diskusije na oglasnoj ploči aplikacije	Visok	Dokument zahtjeva	Suvlasnik može kreirati novu javnu diskusiju vidljivu svim korisnicima.
F-003	Aplikacija omogućuje suvlasniku pokretanje privatne diskusije određivanjem liste sudionika	Visok	Dokument zahtjeva	Suvlasnik može kreirati privatnu diskusiju vidljivu samo odabranim suvlasnicima i predstavniku stanara.
F-004	Aplikacija omogućuje suvlasniku sudjelovanje u javnim i privatnim diskusijama na koje ima pristup	Visok	Dokument zahtjeva	Suvlasnik može pregledavati i objavljivati poruke u diskusijama za koje ima ovlasti.
F-005	Aplikacija omogućuje suvlasniku postavljanje parametara diskusije koju je inicirao	Srednji	Dokument zahtjeva	Suvlasnik može odrediti maksimalan broj odgovora i ograničiti sudjelovanje određenih korisnika u diskusiji.
F-006	Aplikacija omogućuje suvlasniku pokretanje glasanja unutar diskusije	Visok	Dokument zahtjeva	Suvlasnik može postaviti pitanje s odgovorom "da" ili "ne" i pokrenuti glasanje unutar diskusije.
F-007	Aplikacija omogućuje suvlasniku glasanje u diskusijama na koje ima pristup	Visok	Zahtjev dionika	Suvlasnik može dati svoj glas u aktivnom glasanju unutar diskusije.

F-008	Aplikacija omogućuje suvlasniku pregled rezultata glasanja	Visok	Zahtjev dionika	Suvlasnik može vidjeti ažurirane rezultate glasanja u realnom vremenu.
F-009	Aplikacija omogućuje suvlasniku iniciranje kreiranja poziva na sastanak nakon pozitivnog glasanja	Visok	Dokument zahtjeva	Sustav automatski omogućuje kreiranje sastanka ako broj pozitivnih glasova premašuje četvrtinu suvlasnika.
F-010	Aplikacija omogućuje suvlasniku promjenu inicijalne lozinke koristeći prethodnu lozinku	Visok	Zahtjev dionika	Suvlasnik može uspješno promijeniti svoju lozinku za pristup sustavu.
F-011	Aplikacija omogućuje suvlasniku odjavu iz sustava	Visok	Zahtjev dionika	Suvlasnik može se uspješno odjaviti iz aplikacije.
F-012	Aplikacija omogućuje administratoru kreiranje korisničkih računa za suvlasnike i predstavnika suvlasnika	Visok	Dokument zahtjeva	Administrator može dodati nove korisnike s korisničkim imenom, lozinkom i e-mail adresom.
F-013	Aplikacija omogućuje administratoru upravljanje korisničkim računima (uređivanje, brisanje)	Visok	Dokument zahtjeva	Administrator može urediti ili obrisati postojeće korisničke račune.
F-014	Aplikacija omogućuje administratoru pregled popisa svih korisnika	Visok	Dokument zahtjeva	Administrator može vidjeti ažurirani popis svih korisnika sustava.
F-015	Aplikacija omogućuje administratoru unos adrese StanPlan servera za integraciju	Srednji	Dokument zahtjeva	Administrator može uspješno postaviti adresu za integraciju sa StanPlan aplikacijom.
F-016	Aplikacija omogućuje administratoru podešavanje integracije s OAuth 2.0 servisima za autentifikaciju	Srednji	Zahtjev dionika	Administrator može konfigurirati vanjske servise za autentifikaciju korisnika.

F-017	Aplikacija omogućuje slanje zahtjeva StanPlan aplikaciji za kreiranje sastanka	Visok	Dokument zahtjeva	Sustav šalje zahtjev za kreiranje sastanka s potrebnim detaljima StanPlan aplikaciji.
F-018	Sustav elektroničke pošte šalje obavijesti korisnicima kada su dodani u privatnu diskusiju	Srednji	Zahtjev dionika	Korisnici primaju e-mail s informacijom o novoj privatnoj diskusiji u kojoj mogu sudjelovati.

2.2 Ostali zahtjevi

ID zahtjeva	Opis	Prioritet
NF-3.1	Sustav treba biti izgrađen u obliku web-aplikacije koristeći pritom objektno-orijentirane jezike	Visok
NF-3.2	Sustav treba koristiti hrvatski jezik i podržavati hrvatsku abecedu	Visok
NF-3.3	Sustav treba omogućiti istovremeni rad više korisnika	Visok
NF-3.4	Sustav treba biti intuitivan i jednostavan za korištenje	Visok
NF-3.5	Neispravno korištenje sustava ne smije narušiti rad sustava	Visok
NF-3.6	Autentikacijski i ostali privatni podaci moraju biti pravilno osigurani	Visok

2.3 Dionici

1. Suvlasnik
2. Predstavnik suvlasnika
3. Administrator
4. StanPlan aplikacija (vanjski sustav)
5. Naručitelj
6. Razvojni tim

Aktori i njihovi funkcionalni zahtjevi:

1. **Neprijavljeni korisnik (inicijator)** može:
    - Prijaviti se u sustav, za što su potrebni korisničko ime i lozinka, ili neki od podešenih OAuth 2.0 servisa.
2. **Suvlasnik (inicijator)** može:
    - Pokrenuti javnu diskusiju na oglasnoj ploči aplikacije.
    - Pokrenuti privatnu diskusiju određivanjem liste suvlasnika koji mogu sudjelovati.
        - Diskusija može, ali i ne mora uključivati predstavnika stanara.

    - Sudjelovati u javnim diskusijama.
    - Sudjelovati u privatnim diskusijama na koje ima pristup.
    - Postavljati parametre diskusije koju je inicirao:
        - Odrediti maksimalan broj odgovora koji svaki od suvlasnika može postaviti u diskusiju.
        - Ograničiti broj poruka ili zabraniti sudjelovanje određenih suvlasnika.
    - Pokrenuti glasanje u diskusiji postavljanjem pitanja s odgovorom "da" ili "ne".
    - Glasati u glasanju u diskusiji na koju ima pristup.
    - Pregledavati rezultate glasanja.
    - Ako broj pozitivnih odgovora premašuje četvrtinu svih suvlasnika, inicirati kreiranje poziva na sastanak.
    - Promijeniti inicijalnu lozinku za autentikaciju koristeći prethodnu lozinku.
    - Odjaviti se iz sustava.
3. **Predstavnik suvlasnika (inicijator)** može:
    - Sve što može i običan suvlasnik
    - Pružati uvid u procedure, povijest problema i financijski kontekst zgrade.
4. **Administrator (inicijator)** može:
    - Kreirati korisničke račune za suvlasnike i predstavnika suvlasnika, dodjeljujući korisničko ime, lozinku i adresu elektroničke pošte.
    - Unijeti adresu StanPlan servera za integraciju sa sučeljem za sastanke.
    - Upravljati korisničkim računima (uređivanje, brisanje).
    - Pregledati popis svih korisnika.
    - Podesiti integraciju s vanjskim servisima za autentifikaciju (OAuth 2.0).
5. **StanPlan aplikacija (vanjski sustav) (sudionik)**:
    - Prima zahtjev za kreiranje sastanka od StanBlog aplikacije putem sučelja.
    - Omogućuje kreiranje sastanka s naslovom, terminom, tekstom poziva, dnevnim redom i ciljevima sastanka.
    - Koristi serversko sučelje StanBlog aplikacije za preuzimanje liste diskusija s pozitivnim ishodom glasanja.
6. **Sustav elektroničke pošte (sudionik)**:
    - Šalje poruke elektroničke pošte korisnicima kada su dodani u privatnu diskusiju, informirajući ih da mogu sudjelovati u njoj.
7. **Baza podataka (sudionik)**:
    - Pohranjuje korisničke podatke i njihove uloge.
    - Pohranjuje diskusije, poruke, glasanja i njihove rezultate.
    - Pohranjuje postavke i parametre diskusija.
    - Pohranjuje informacije o integraciji sa StanPlan aplikacijom.

3. Specifikacija zahtjeva sustava

3.1 Obrasci uporabe

UC1 - Prijava putem OAuth 2.0 servisa

- **Glavni sudionik:** Korisnik
- **Cilj:** Pristupiti sustavu korištenjem vanjske autentikacije
- **Sudionici:** Vanjski servis za autentikaciju (OAuth 2.0)
- **Preduvjet:** Korisnik ima račun na vanjskom servisu za autentikaciju
- **Opis osnovnog tijeka:**
    - Korisnik odabire opciju "Sign in with Google" ili "Sign in with GitHub"
    - Sustav preusmjerava korisnika na vanjski servis za autentikaciju
    - Korisnik unosi svoje vjerodajnice na vanjskom servisu
    - Vanjski servis potvrđuje identitet korisnika i preusmjerava ga natrag u aplikaciju
    - Korisnik dobiva pristup sustavu
- **Opis mogućih odstupanja:**
    - **3.a** Korisnik unosi neispravne vjerodajnice
        1. Vanjski servis obavještava korisnika o pogrešci
        2. Korisnik ponovno pokušava prijavu ili odustaje

UC2 - Prijava putem lokalne autentikacije

- **Glavni sudionik**: Korisnik
- **Cilj:** Pristupiti sustavu korištenjem interne autentikacije
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik ima korisnički račun
- **Opis osnovnog tijeka:**
    - Korisnik unosi korisničko ime i lozinku
    - Sustav provjerava vjerodajnice u bazi podataka
    - Korisnik dobiva pristup sustavu
- **Opis mogućih odstupanja:**
    - **2.a** Korisnik unosi neispravne vjerodajnice ili nepostojeći račun
        1. Sustav obavještava korisnika o pogrešci
        2. Korisnik ponovno unosi vjerodajnice ili odustaje

UC3 - Registracija novog korisnika

- **Glavni sudionik:** Korisnik
- **Cilj:** Kreirati novi korisnički račun
- **Sudionici:** Baza podataka

- **Preduvjet:** -
- **Opis osnovnog tijeka:**
    - Korisnik odabire opciju "Registracija"
    - Korisnik unosi korisničko ime, e-mail adresu i lozinku
    - Sustav provjerava unesene podatke
    - Sustav sprema podatke u bazu podataka
    - Korisnik dobiva obavijest o uspješnoj registraciji
- **Opis mogućih odstupanja:**
    - **3.a** Korisnik unosi neispravne ili već korištene podatke
        1. Sustav obavještava korisnika o pogrešci
        2. Korisnik unosi nove podatke ili odustaje

UC4 - Promjena lozinke

- **Glavni sudionik:** Korisnik
- **Cilj:** Promijeniti inicijalnu lozinku za autentikaciju
- **Sudionici:** Baza podataka
- **Preduvjet:**
    - Korisnik je prijavljen u sustav
    - Korisnik ima inicijalnu lozinku
- **Opis osnovnog tijeka:**
    - Korisnik odabire opciju "Promjena lozinke"
    - Korisnik unosi trenutnu lozinku i novu lozinku
    - Korisnik potvrđuje novu lozinku
    - Sustav ažurira lozinku u bazi podataka
    - Korisnik dobiva obavijest o uspješnoj promjeni lozinke
- **Opis mogućih odstupanja:**
    - **2.a** Korisnik unosi netočnu trenutnu lozinku
        1. Sustav obavještava korisnika o pogrešci
        2. Korisnik ponovno unosi lozinku ili odustaje

UC5 - Kreiranje korisnika

- **Glavni sudionik:** Administrator
- **Cilj:** Kreirati korisničke račune za predstavnika suvlasnika i suvlasnike
- **Sudionici:** Baza podataka
- **Preduvjet:** Administrator je prijavljen u sustav
- **Opis osnovnog tijeka:**
    - Administrator odabire opciju "Kreiraj korisnika"
    - Administrator unosi korisničko ime, inicijalnu lozinku i e-mail adresu korisnika
    - Administrator odabire tip korisnika (predstavnik suvlasnika ili suvlasnik)
    - Sustav sprema podatke u bazu podataka
    - Korisnik dobiva obavijest putem e-pošte o kreiranju računa
- **Opis mogućih odstupanja:**

    - **2.a** Uneseni podaci su neispravni ili nepotpuni
        1. Sustav obavještava administratora o pogrešci
        2. Administrator ispravlja podatke i ponovno pokušava

---

UC6 - Iniciranje diskusije

- **Glavni sudionik:** Suvlasnik
- **Cilj:** Pokrenuti novu diskusiju na oglasnoj ploči
- **Sudionici:** Baza podataka, drugi suvlasnici, predstavnik suvlasnika
- **Preduvjet:** Korisnik je prijavljen u sustav
- **Opis osnovnog tijeka:**
    - Suvlasnik odabire opciju "Nova diskusija"
    - Suvlasnik unosi naslov i sadržaj diskusije
    - Suvlasnik određuje parametre diskusije:
        1. Tip diskusije (javna ili privatna)
        2. Maksimalan broj odgovora po korisniku
        3. Ograničenje broja poruka
        4. Zabrana sudjelovanja određenim suvlasnicima
    - Ako je diskusija privatna, suvlasnik odabire sudionike
    - Suvlasnik objavljuje diskusiju
    - Sustav sprema diskusiju i obavještava odabrane sudionike putem e-pošte
- **Opis mogućih odstupanja:**
    - **5.a** Suvlasnik ne unese sve potrebne podatke
        1. Sustav obavještava korisnika o nedostajućim podacima
        2. Suvlasnik unosi nedostajuće podatke i ponovno pokušava
    - **5.b** Suvlasnik pokuša napustiti stranicu prije objave diskusije
        1. Sustav obavještava korisnika da se uneseni podaci neće spremiti
        2. Korisnik odabire "Odbaci" i odbacuje sve unesene podatke, ili "Nastavi" te nastavlja s osnovnim tijekom

---

UC7 - Sudjelovanje u diskusiji

- **Glavni sudionik:** Suvlasnik, predstavnik suvlasnika
- **Cilj:** Sudjelovati u diskusiji odgovaranjem na poruke
- **Sudionici:** Baza podataka
- **Preduvjet:**
    - Korisnik je prijavljen u sustav
    - Diskusija je javna ili je korisnik uključen u privatnu diskusiju
    - Korisnik ima pravo sudjelovanja (nije zabranjen)
- **Opis osnovnog tijeka:**
    - Korisnik pristupa odabranoj diskusiji
    - Korisnik čita prethodne poruke
    - Korisnik unosi svoj odgovor
    - Sustav sprema odgovor i prikazuje ga u diskusiji

- **Opis mogućih odstupanja:**
    - **3.a** Korisnik je dosegao maksimalan broj odgovora u diskusiji
        1. Sustav obavještava korisnika da je dosegnut limit odgovora
        2. Korisnik ne može dodati novi odgovor

UC8 - Pokretanje glasovanja u diskusiji

- **Glavni sudionik:** Inicijator diskusije
- **Cilj:** Postaviti pitanje za glasovanje u diskusiji
- **Sudionici:** Baza podataka
- **Preduvjet:**
    - Inicijator je pokrenuo diskusiju
    - Diskusija je aktivna
- **Opis osnovnog tijeka:**
    - Inicijator odabire opciju "Pokreni glasovanje"
    - Inicijator unosi pitanje za glasovanje (odgovor da/ne)
    - Sustav dodaje glasovanje kao komentar u diskusiji
    - Sudionici diskusije vide pitanje i mogu glasati
- **Opis mogućih odstupanja:**
    - **2.a** Inicijator ne unese pitanje
        1. Sustav obavještava korisnika o potrebi unosa pitanja
        2. Inicijator unosi pitanje i nastavlja proces

UC9 - Glasovanje

- **Glavni sudionik:** Sudionik diskusije
- **Cilj:** Glasati na postavljeno pitanje u diskusiji
- **Sudionici:** Baza podataka
- **Preduvjet:**
    - Korisnik je sudionik diskusije
    - Postoji aktivno glasovanje
- **Opis osnovnog tijeka:**
    - Korisnik vidi pitanje za glasovanje
    - Korisnik odabire svoj odgovor (da ili ne)
    - Sustav bilježi glas korisnika
    - Sustav ažurira prikaz trenutnih rezultata glasovanja
- **Opis mogućih odstupanja:**
    - **2.a** Korisnik je već glasao
        1. Sustav obavještava korisnika da je već glasao
        2. Korisnik ne može ponovno glasati
    - **2.b** Glasovanje je završeno
        1. Sustav obavještava korisnika da je glasovanje završeno
        2. Korisnik ne može glasati

UC10 - Kreiranje sastanka

- **Glavni sudionik:** Inicijator diskusije
- **Cilj:** Kreirati sastanak nakon pozitivnog ishoda glasovanja
- **Sudionici:** Baza podataka, vanjska aplikacija (StanPlan)
- **Preduvjet:**
    - Broj pozitivnih glasova premašuje četvrtinu svih suvlasnika
    - Inicijator želi kreirati sastanak
- **Opis osnovnog tijeka:**
    - Inicijator odabire opciju "Kreiraj sastanak"
    - Sustav komunicira s vanjskom aplikacijom preko sučelja
    - Inicijator unosi detalje sastanka:
        1. Naslov
        2. Termin
        3. Tekst poziva (dnevni red, ciljevi)
    - Sustav šalje podatke vanjskoj aplikaciji (StanPlan)
    - Vanjska aplikacija kreira sastanak i vraća potvrdu
    - Sustav obavještava sudionike o kreiranom sastanku
- **Opis mogućih odstupanja:**
    - **2.a** Neuspješna komunikacija s vanjskom aplikacijom
        1. Sustav obavještava inicijatora o pogrešci
        2. Inicijator pokušava ponovno ili odustaje

UC11 - Pregled liste diskusija s pozitivnim ishodom glasanja (serversko sučelje)

- **Glavni sudionik:** Vanjska aplikacija
- **Cilj:** Preuzeti listu diskusija s pozitivnim ishodom glasanja
- **Sudionici:** Baza podataka
- **Preduvjet:**
    - Postoje diskusije s pozitivnim ishodom glasanja
    - Vanjska aplikacija ima pristup serverskom sučelju
- **Opis osnovnog tijeka:**
    - Vanjska aplikacija šalje zahtjev za listu diskusija
    - Sustav vraća listu koja uključuje:
        1. Naslov diskusije
        2. Poveznicu na diskusiju
        3. Pitanje s pozitivnim ishodom glasanja
- **Opis mogućih odstupanja:**
    - **1.a** Neispravan zahtjev od vanjske aplikacije
        1. Sustav vraća poruku o pogrešci
        2. Vanjska aplikacija ispravlja zahtjev i ponovno pokušava

---

UC12 - Promjena adrese StanPlan servera

- **Glavni sudionik:** Administrator
- **Cilj:** Unijeti ili promijeniti adresu StanPlan servera za integraciju
- **Sudionici:** Baza podataka
- **Preduvjet:** Administrator je prijavljen u sustav
- **Opis osnovnog tijeka:**
    - Administrator odabire opciju "Postavke integracije"
    - Administrator unosi ili mijenja adresu StanPlan servera
    - Sustav sprema adresu u bazu podataka
    - Sustav potvrđuje uspješnu promjenu
- **Opis mogućih odstupanja:**
    - **2.a** Unesena adresa je neispravna
        1. Sustav obavještava administratora o pogrešci
        2. Administrator ispravlja adresu i ponovno pokušava

UC13 - Obavještavanje sudionika putem e-pošte

- **Glavni sudionik:** Sustav
- **Cilj:** Obavijestiti korisnike o relevantnim događajima putem e-pošte
- **Sudionici:** Korisnici, e-mail servis
- **Preduvjet:**
    - Događaj koji zahtijeva obavijest (npr. dodavanje u diskusiju, kreiranje sastanka)
- **Opis osnovnog tijeka:**
    - Sustav detektira događaj koji zahtijeva obavijest
    - Sustav generira poruku e-pošte s relevantnim informacijama
    - Sustav šalje e-poštu korisnicima
- **Opis mogućih odstupanja:**
    - **3.a** Slanje e-pošte neuspješno
        1. Sustav bilježi pogrešku i pokušava ponovno slanje
        2. Ako ponovljeni pokušaji ne uspiju, sustav obavještava administratora

---

UC14 - Pregled diskusija

- **Glavni sudionik:** Korisnik
- **Cilj:** Pregledati postojeće diskusije
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav
- **Opis osnovnog tijeka:**
    - Korisnik odabire opciju "Diskusije"
    - Sustav prikazuje listu diskusija:
        1. Javne diskusije
        2. Privatne diskusije u kojima je korisnik sudionik

    - Korisnik odabire diskusiju za pregled
- **Opis mogućih odstupanja:**
    - **2.a** Nema dostupnih diskusija
        1. Sustav obavještava korisnika da nema dostupnih diskusija

---

UC15 - Odjava

- **Glavni sudionik:** Korisnik
- **Cilj:** Odjaviti se iz sustava
- **Sudionici:** -
- **Preduvjet:** Korisnik je prijavljen u sustav
- **Opis osnovnog tijeka:**
    1. Korisnik odabire opciju "Odjava"
    2. Sustav završava korisničku sesiju
    3. Korisnik biva preusmjeren na stranicu za prijavu

UC16 - Resetiranje lozinke

- **Glavni sudionik:** Korisnik
- **Cilj:** Resetirati zaboravljenu lozinku
- **Sudionici:** Baza podataka, e-mail servis
- **Preduvjet:** Korisnik postoji u sustavu
- **Opis osnovnog tijeka:**
    - Korisnik odabire opciju "Zaboravljena lozinka"
    - Korisnik unosi svoju e-mail adresu
    - Sustav šalje e-poštu s uputama za resetiranje lozinke
    - Korisnik slijedi upute iz e-pošte i postavlja novu lozinku
- **Opis mogućih odstupanja:**
    - **2.a** Unesena e-mail adresa nije povezana s korisničkim računom
        1. Sustav obavještava korisnika da račun ne postoji
        2. Korisnik ponovno unosi e-mail adresu ili odustaje
    - **4.a** Link za resetiranje lozinke je istekao ili nevažeći
        1. Sustav obavještava korisnika o pogrešci
        2. Korisnik ponovno pokreće proces resetiranja lozinke

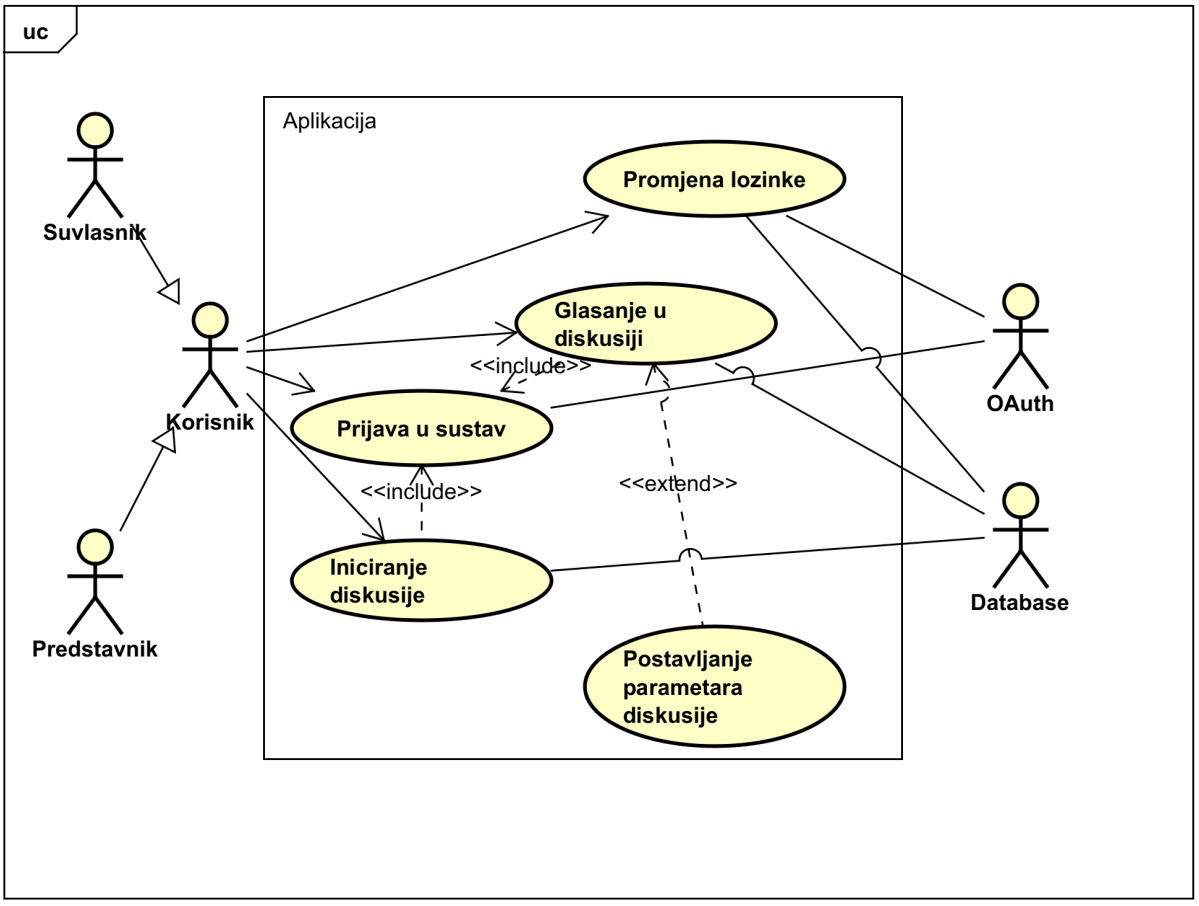
---

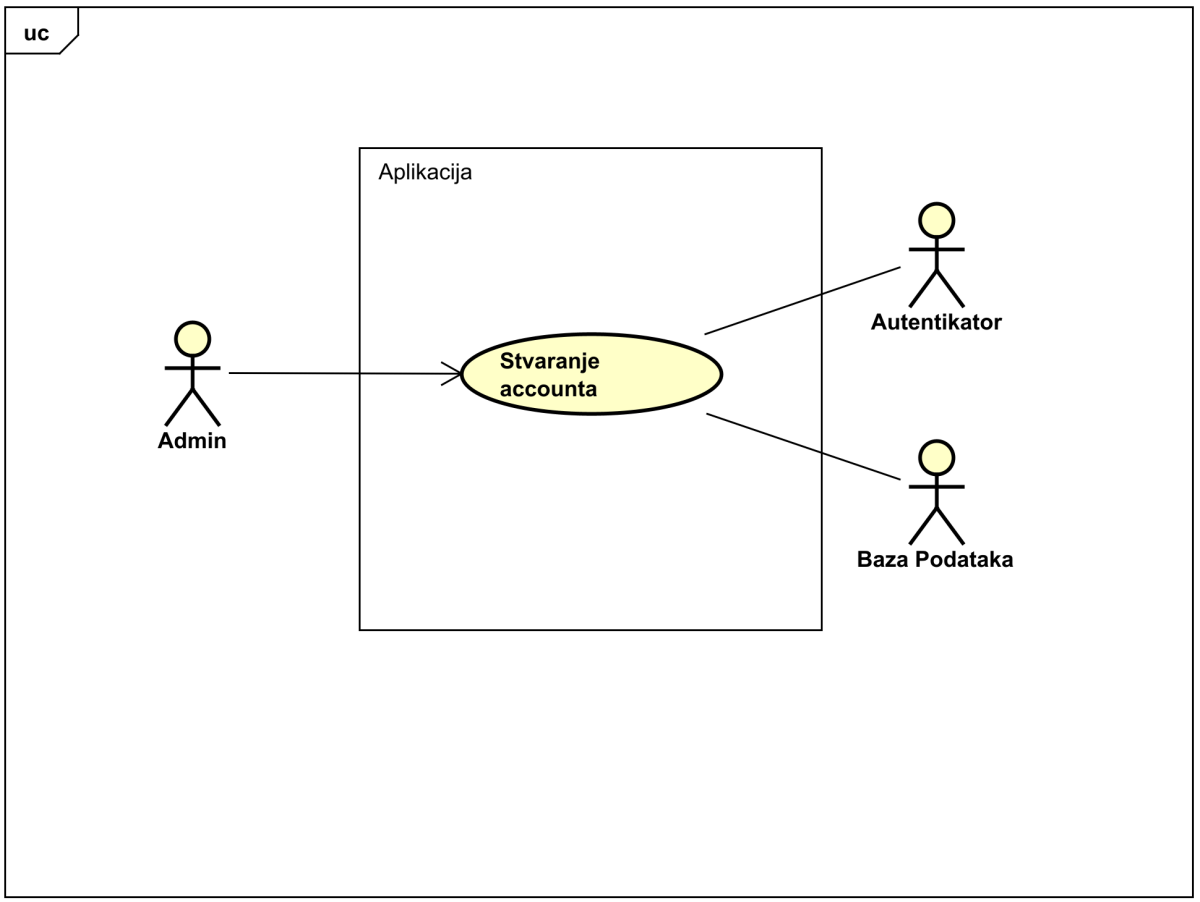
UC17 - Uređivanje parametara diskusije

- **Glavni sudionik:** Inicijator diskusije
- **Cilj:** Urediti parametre postojeće diskusije
- **Sudionici:** Baza podataka
- **Preduvjet:**
    - Diskusija je aktivna

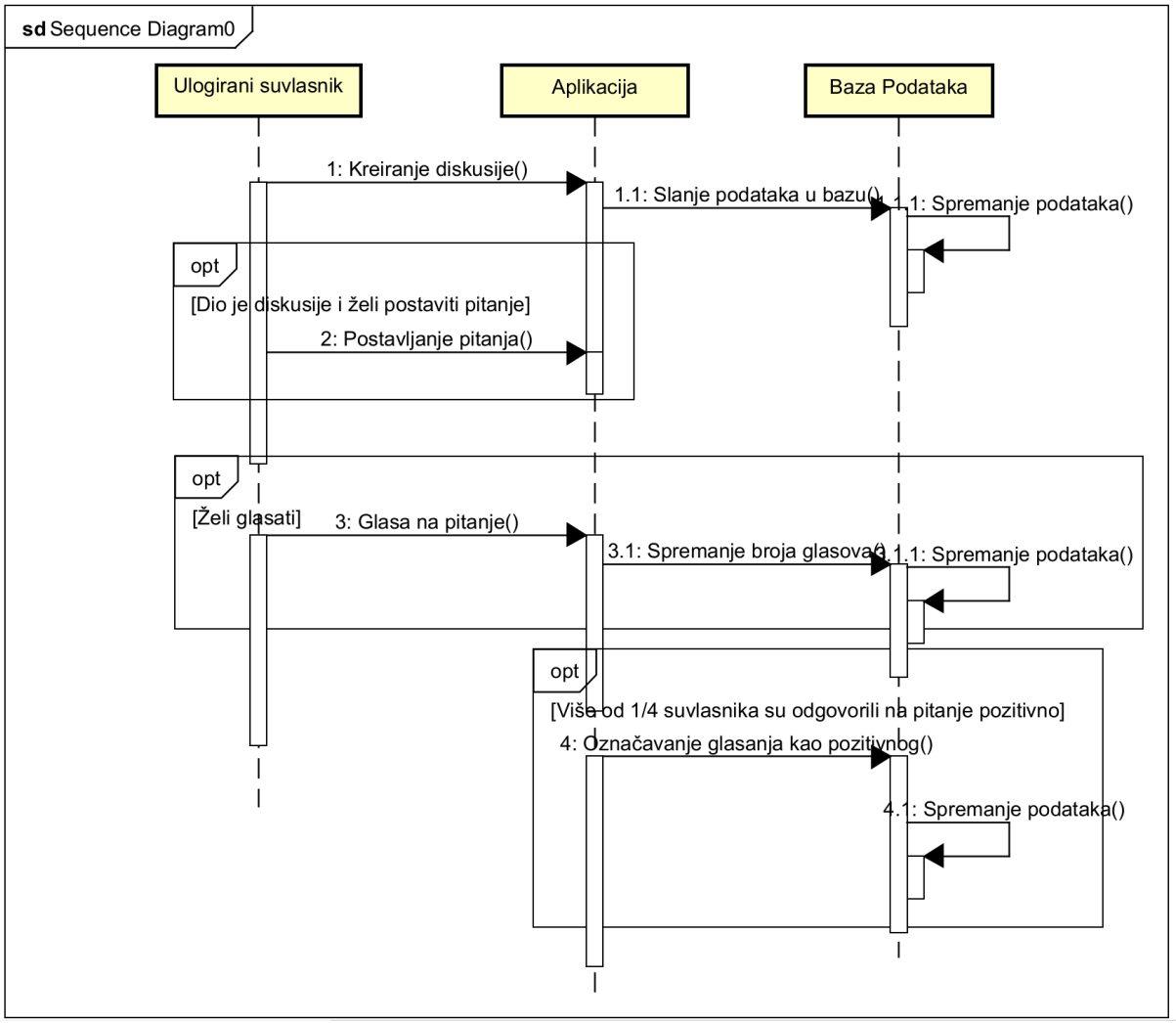
    - Korisnik je inicijator diskusije
- **Opis osnovnog tijeka:**
    - Inicijator odabire diskusiju koju je pokrenuo
    - Inicijator odabire opciju "Uredi diskusiju"
    - Inicijator mijenja parametre diskusije:
        1. Maksimalan broj odgovora po korisniku
        2. Ograničenje broja poruka
        3. Zabrana sudjelovanja određenim suvlasnicima
    - Inicijator sprema promjene
    - Sustav ažurira diskusiju u bazi podataka
- **Opis mogućih odstupanja:**
    - **3.a** Inicijator ne unese ispravne podatke
        1. Sustav obavještava inicijatora o pogrešci
        2. Inicijator ispravlja podatke i ponovno pokušava

3.2 Dijagrami obrazaca uporabe




uc
Aplikacija
Stvaranje accounta
Admin
Autentikator
Baza Podataka

3.3 Sekvencijski dijagrami



4. Arhitektura i dizajn sustava

Arhitektura sustava sastoji se od tri podsustava: web poslužitelj, klijent (web preglednik) i baza podataka.

- **Poslužitelj** je centar sustava. Komunicira s klijentom i bazom podataka, a ta se komunikacija zasniva na HTTP protokolu. Može vraćati zatražene resurse klijentu te prihvaćati poslane resurse i po potrebi spremati u bazu podataka.
- **Klijent** je web stranica preko koje korisnik ostvaruje interakciju sa sustavom. Može slati zahtjeve za resursima i slati vlastite podatke prema poslužitelju. Prilikom korištenja, promjene i dohvaćeni resursi se na klijentu odražavaju na odgovarajući način.
- **Baza podataka** koristi se prilikom skoro svih obrazaca uporabe te se koristi za spremanje podataka koje joj poslužitelj dostavlja. Podatci iz nje se mogu i čitati te obrađivati na poslužitelju.

Arhitektura se zasniva na modelu objektno orijentiranog programiranja. Detaljnije, komponente su podijeljene na:

- **modele**, koji određuju razrede objekta koje se obrađuje,
- **preglede**, koji sadrže razrede koji su zaduženi za prikazivanje i ispisivanje elemenata aplikacije i podataka
- **nadglednike**, koji sadrže razrede koji nadgledaju same funkcionalnosti sustava. To može uključivati obradu podataka, korisnikove zahtjeve i sl.

Ovaj je model izabran jer odgovara zahtjevima sustava koji se temelje na slanju i obradi te ponovnom prikazu poruka.

Za backend koristi se Java Spring, a frontend je izrađen u Reactu. Vanjski servisi s kojima aplikacija komunicira su:

- OAuth sustav za autentikaciju korisnika
- servis kojim će se slati e-mailovi.

4.1 Baza podataka

U projektu koristi se PostgreSQL baza podataka koja se bazira na relacijskom modelu. Ovaj je model izabran jer se na taj način jednostavno mogu povezati podaci u tablice i dohvaćati pomoću seta dostupnih ključeva, što omogućuje jednostavno sortiranje i odgovarajuću dostupnost traženih podataka.

Komponente baze su entiteti sa pripadajućim atributima. Neki skup atributa je pritom primarni ključ, no i drugi skupovi atributa se mogu koristiti za brzo i efikasno dohvaćanje podataka, što ovaj model čini izvrsnim za naše potrebe.

Entiteti od kojih se baza podataka sastoji su:

- **User**
- **Thread**
- **Message** (i **Poll**)
- **Board**

4.1.1 Opis tablica

4.1.1.1 User

User je entitet koji opisuje korisnika aplikacije. Ima sljedeće atribute: *userID* - njegov jedinstveni identifikator, *username* - njegovo jedinstveno korisničko ime te *email*, *password* i *role*.

Korisnik može biti dio više oglasnih ploča. Korisnik također može biti uključen u više diskusija, te može biti pošiljatelj više poruka. Korisnik može glasati na više pitanja (*Poll*) i pitanje može imati više glasača.

Atribut	Tip podatka	Opis varijable
userID (PK)	int	Jedinstveni identifikator korisnika
username (U)	varchar (20)	Jedinstveno korisničko ime
password	char (60) binary	Korisnička lozinka
email (U)	varchar (50)	E-mail adresa korisnika
role	enum (admin, tenant, representative)	Funkcija korisnika u sustavu (administrator, stanar ili predstavnik)

4.1.1.2 Thread

Thread je entitet koji opisuje neku diskusiju.

Sadrži sljedeće atribute: *threadID* - jedinstveni identifikator diskusije, te *maxResponses*, *maxMessages* i *Private* koji određuju parametre te diskusije.

Diskusija pripada točno jednoj oglasnoj ploči. S njom može interaktirati jedan ili više korisnika. Može sadržavati više tekstualnih poruka.

Atribut	Tip podatka	Opis varijable
threadID (PK)	int	Jedinstveni identifikator diskusije

maxResponses	int	Najveći dopušteni broj odgovora odnosno poruka koje jedan korisnik može poslati u diskusiju
maxMessages	int	Najveći ukupni broj poruka koji se može poslati u diskusiju
Private	bool	1 ili više ako je diskusija privatna, 0 ako je javna

4.1.1.3 Message

Message je entitet koji modelira tekstualne poruke koje mogu, ali ne moraju, biti specijalizirane u pitanje (Poll).

Sadrži sljedeće atribute: *messageID* - jedinstveni identifikator poruke, *title* - naslov poruke, te *threadID* i *userID* - identifikatori diskusije kojoj poruka pripada i identifikator korisnika koji je poslao tu poruku.

Atribut	Tip podatka	Opis varijable
messageID (PK)	int	Jedinstveni identifikator poruke
content	nvarchar	Sadržaj poruke
threadID (FK)	int	Jedinstveni identifikator diskusije kojoj poruka pripada
userID (FK)	int	Jedinstveni identifikator korisnika koji je poslao poruku

4.1.1.4 Poll

Poll je entitet koji opisuje specijalnu inačicu tekstualne poruke - pitanje. Uz *messageID*, koji je identifikator te poruke, ima i atribut *title* koji sadrži skraćeni tekst pitanja.

Atribut	Tip podatka	Opis varijable
messageID (PK)(FK)	int	Jedinstveni identifikator poruke koja je tipa pitanje
title	nvarchar	Tekst pitanja koje se postavlja

4.1.1.5 Board

Board je entitet koji opisuje oglasnu ploču. Kako oglasna ploča pripada jednoj zgradi, *boardID* je jedinstveni identifikator oglasne ploče, a *adress* je adresa zgrade kojoj ta oglasna ploča pripada.

Atribut	Tip podatka	Opis varijable

boardID (PK)	int	Jedinstveni identifikator oglasne ploče
adress (U)	nvarchar	Jedinstvena adresa zgrade za koju je načinjena oglasna ploča

4.1.1.6 CanInteractWith

CanInteractWith je relacija koja pridružuje sudionike diskusije s određenom diskusijom, temeljem njihovih jedinstvenih identifikatora.

Jedan korisnik može biti dio više diskusija, a jedna diskusija može uključivati jednog ili više korisnika.

Atribut	Tip podatka	Opis varijable
threadID (PK)(FK)	int	Jedinstveni identifikator diskusije
userID (PK)(FK)	int	Jedinstveni identifikator korisnika koji može interagirati s oglasnom pločom

4.1.1.7 IsAPartOf

IsAPartOf je relacija koja povezuje korisnike s oglasnim pločama kojima pripadaju, temeljem njihovih jedinstvenih identifikatora.

Jedan korisnik može biti dio više oglasnih ploča, a jedna oglasna ploča može sadržavati više korisnika.

Atribut	Tip podatka	Opis varijable
userID (PK)(FK)	int	Jedinstveni identifikator korisnika koji pripada oglasnoj ploči
boardID (PK)(FK)	int	Jedinstveni identifikator oglasne ploče

4.1.1.8 Vote

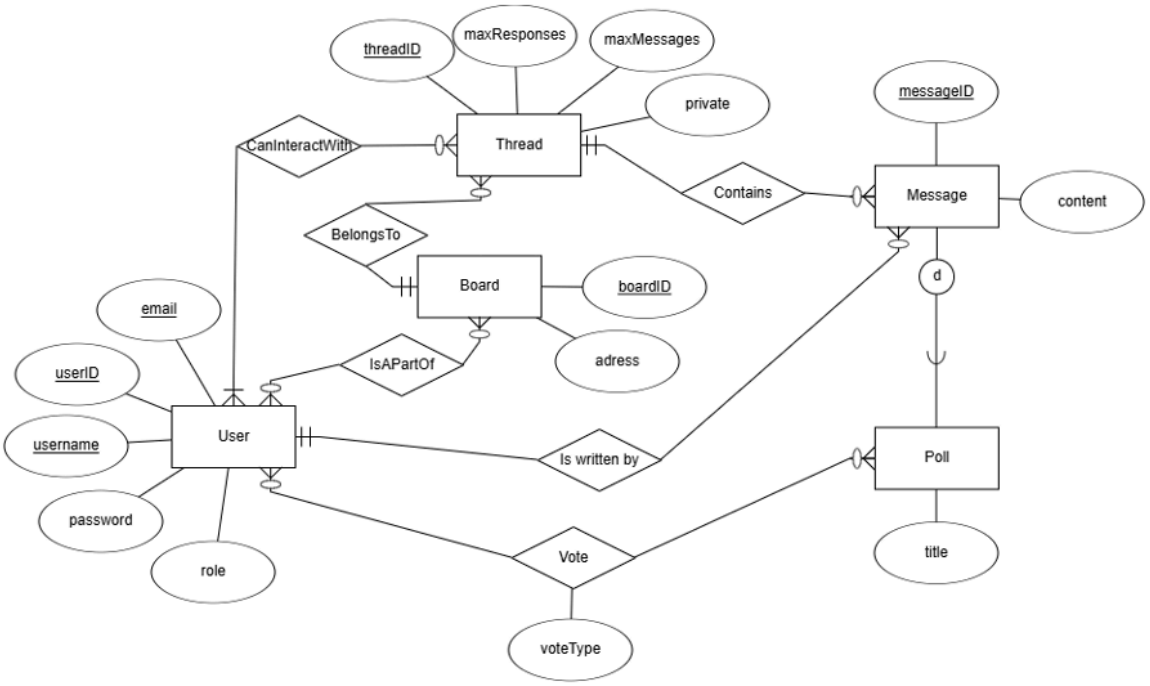
Vote je relacija koja pridružuje korisnike i pitanja na koja su glasali.

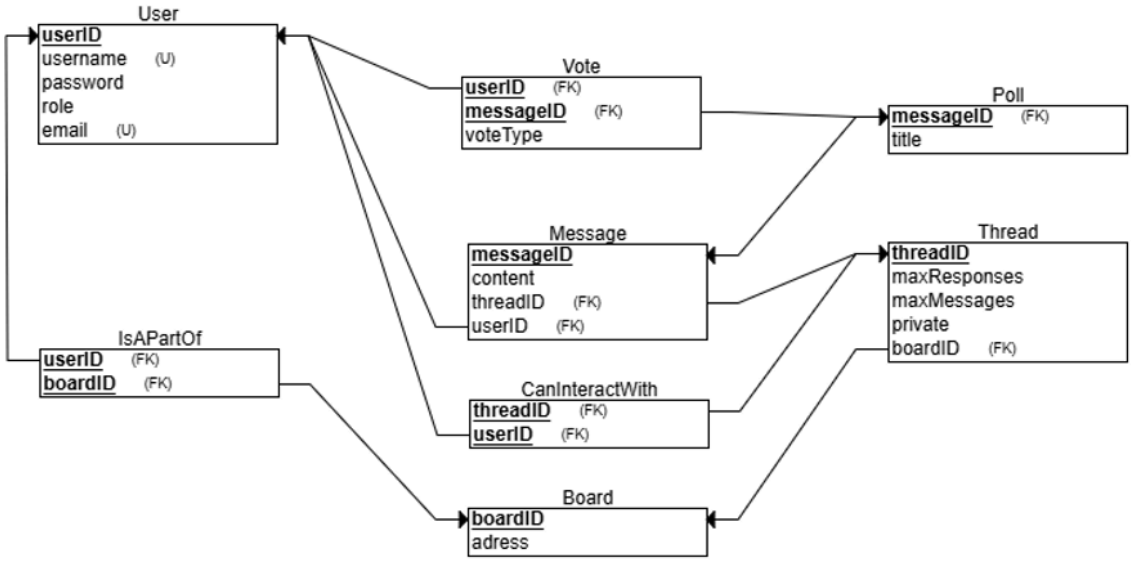
Korisnik može glasati na više pitanja i pitanje može imati više glasača.

Kao atribut ova veza sadrži vrstu glasa (*voteType*). Glas može biti pozitivan ili negativan, odnosno *upvote* ili *downvote*.

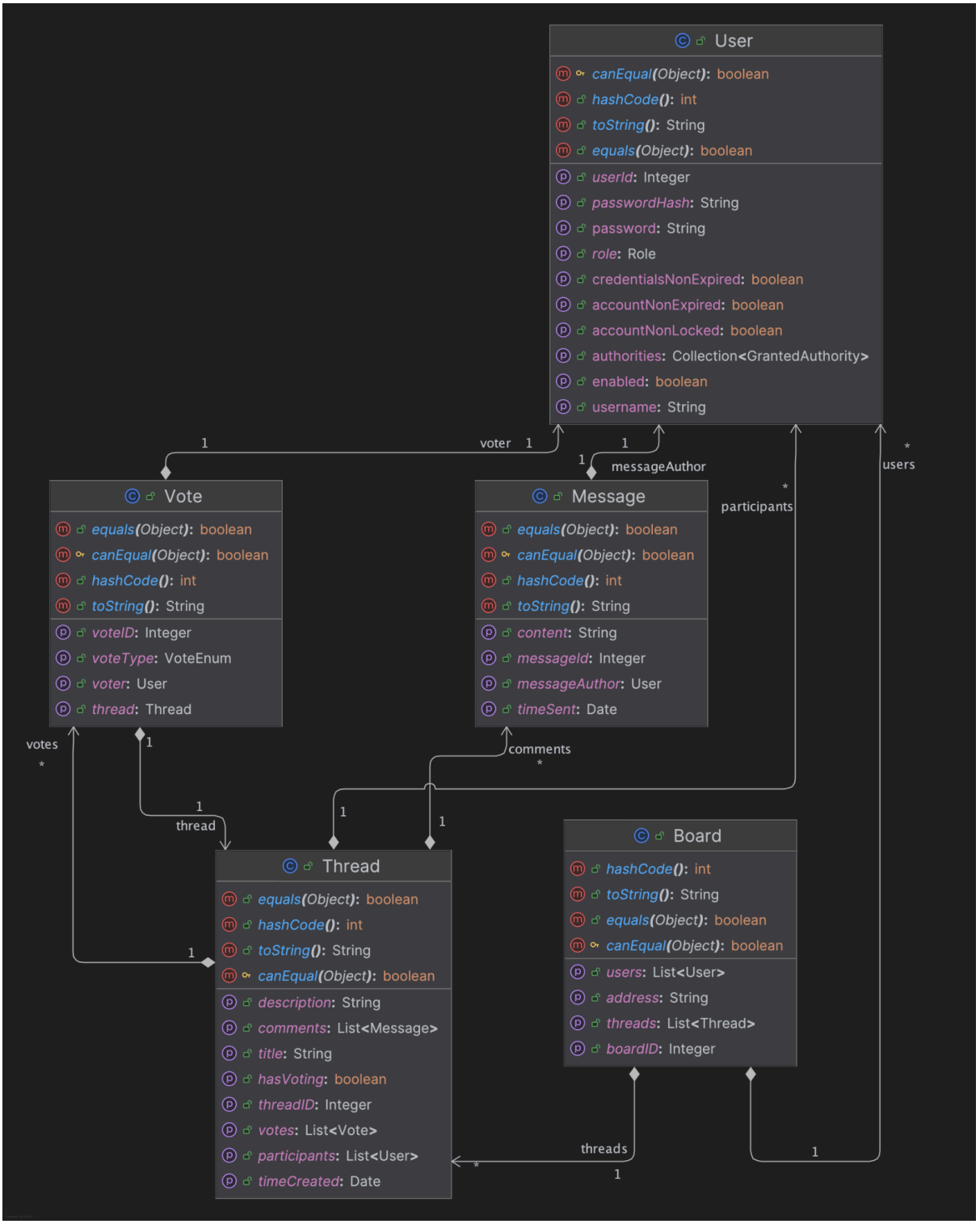
Atribut	Tip podatka	Opis varijable
messageID (PK)(FK)	int	Jedinstveni identifikator poruke
userID (PK)(FK)	int	Jedinstveni identifikator korisnika koji je glasao na poruku
voteType	Enum (upvote, downvote)	Vrsta glasa koja može biti pozitivni glas (*upvote*) ili negativni glas (*downvote*)

4.1.2 Dijagram baze podataka



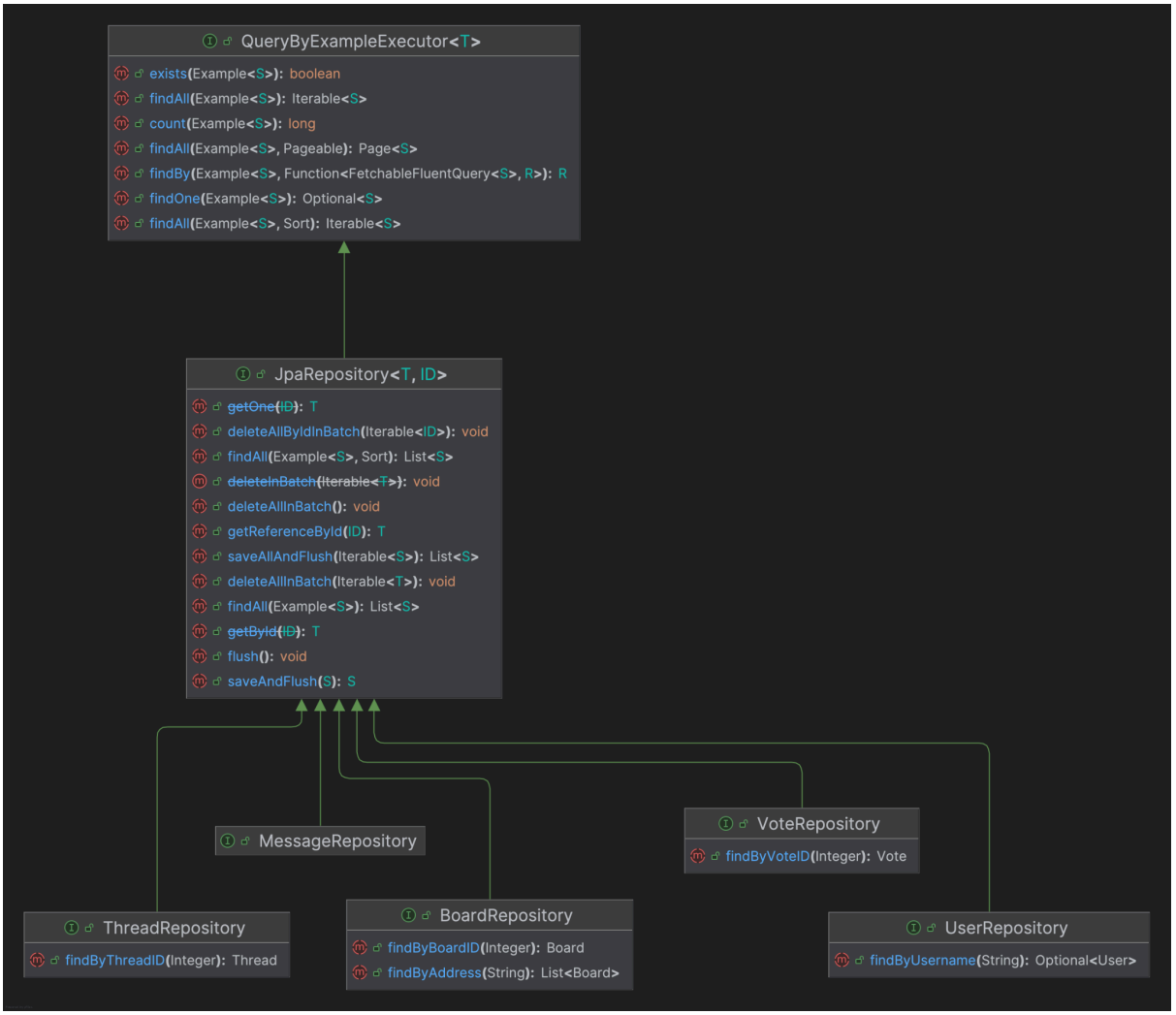


4.2 Dijagram razreda



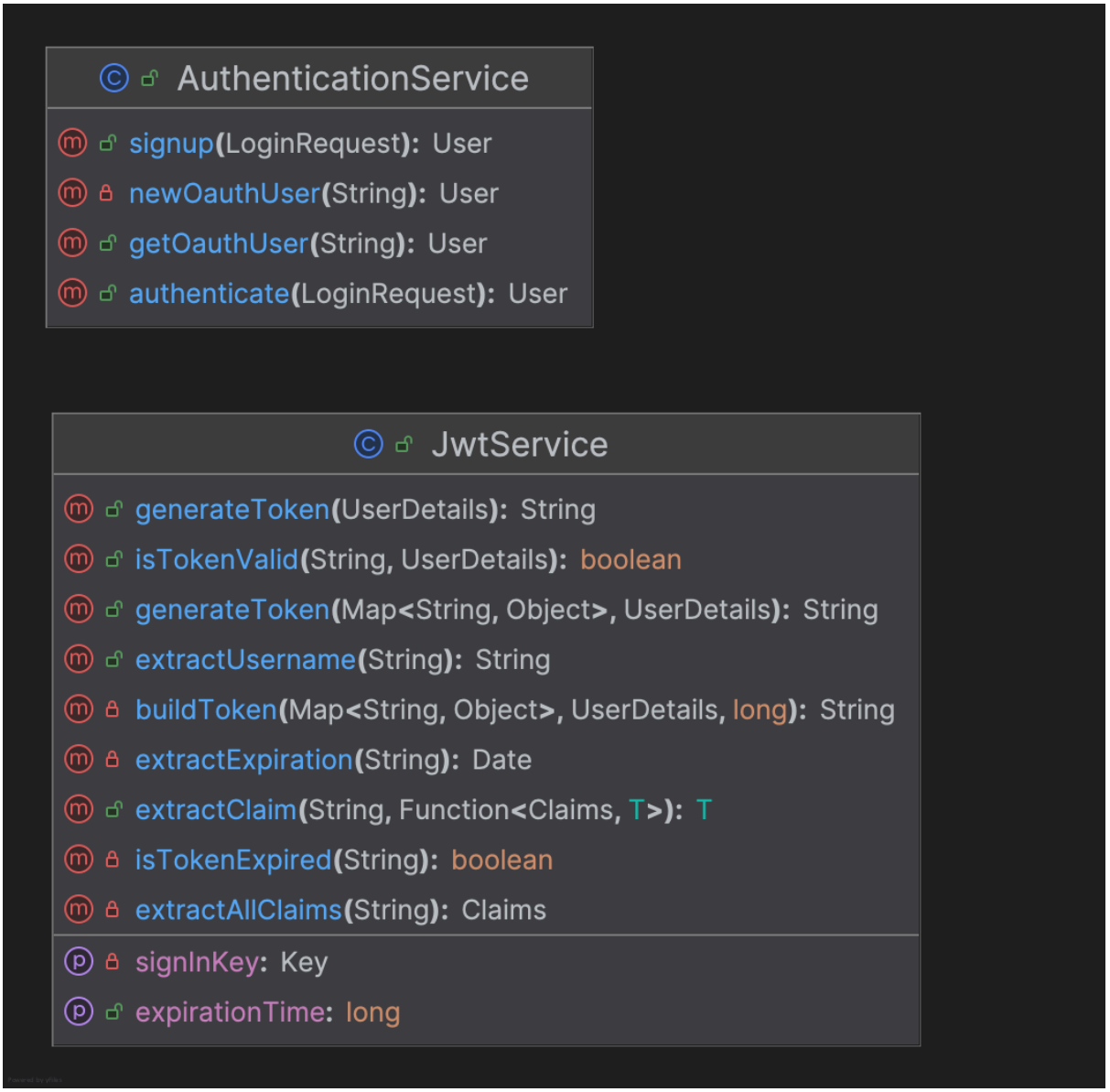
Dijagram razreda - sloj DataTypes

Ovaj sloj sadrži razrede koji predstavljaju entitete baze podataka. Svaki razred odgovara jednoj tablici u bazi podataka. Razredi su povezani relacijama koje odgovaraju relacijama među tablicama u bazi podataka.



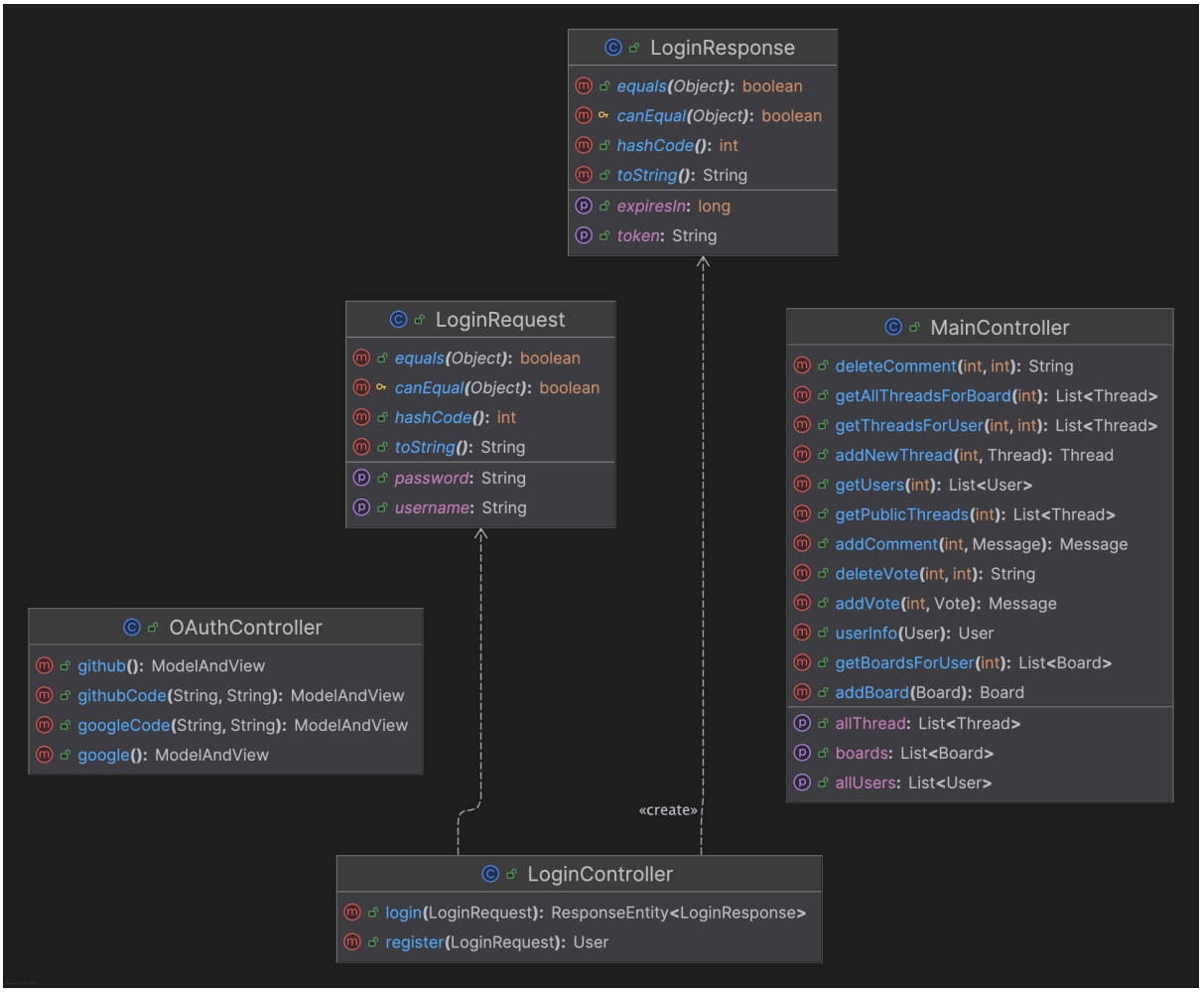
Dijagram razreda - sloj Repository

Ovaj sloj sadrži razrede koji upravljaju entitetima baze podataka. Razredi sadrže metode za dohvaćanje, spremanje, ažuriranje i brisanje podataka iz baze podataka. Implementirani su



Dijagram razreda - sloj Service

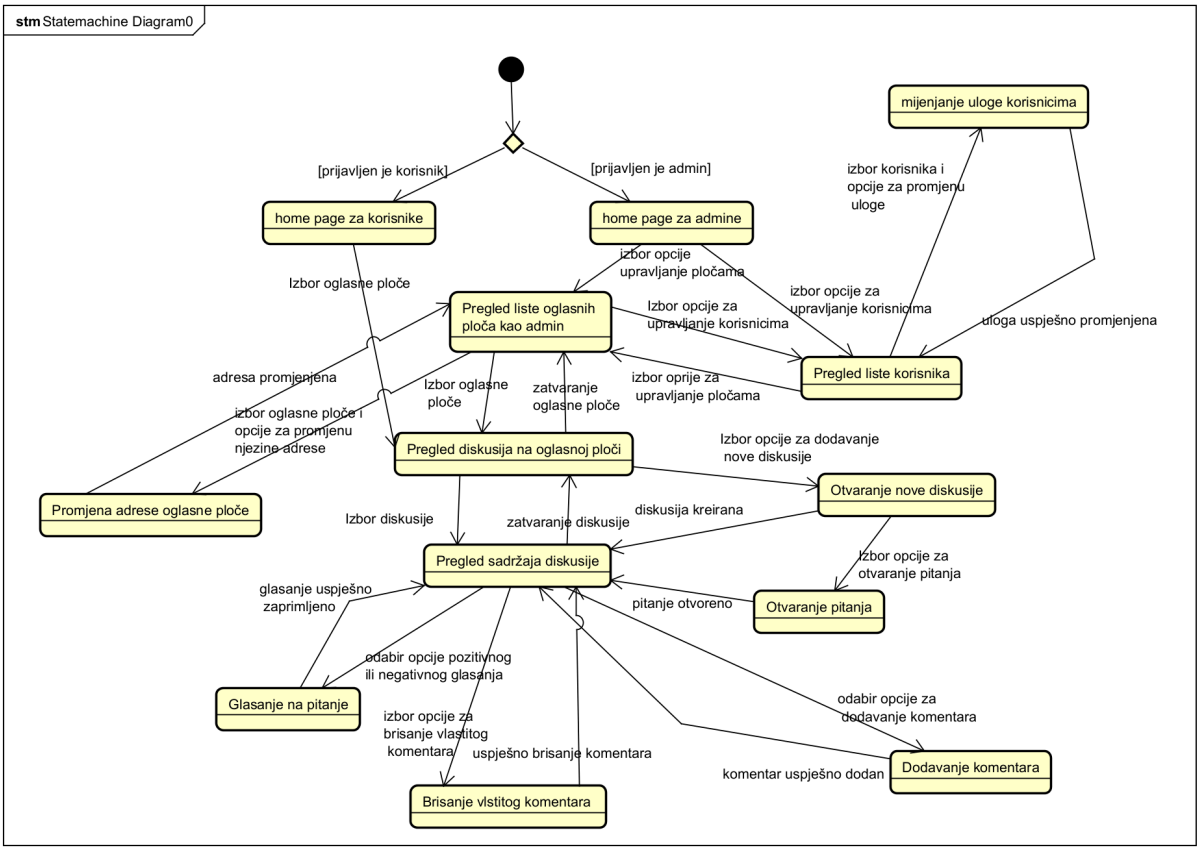
Ovaj sloj sadrži razrede koji upravljaju poslovnim logikama aplikacije. Razredi sadrže metode koje obrađuju poslovne logike, pozivaju metode sloja Repository i vraćaju rezultate sloju Controller.



Dijagram razreda - sloj Controller

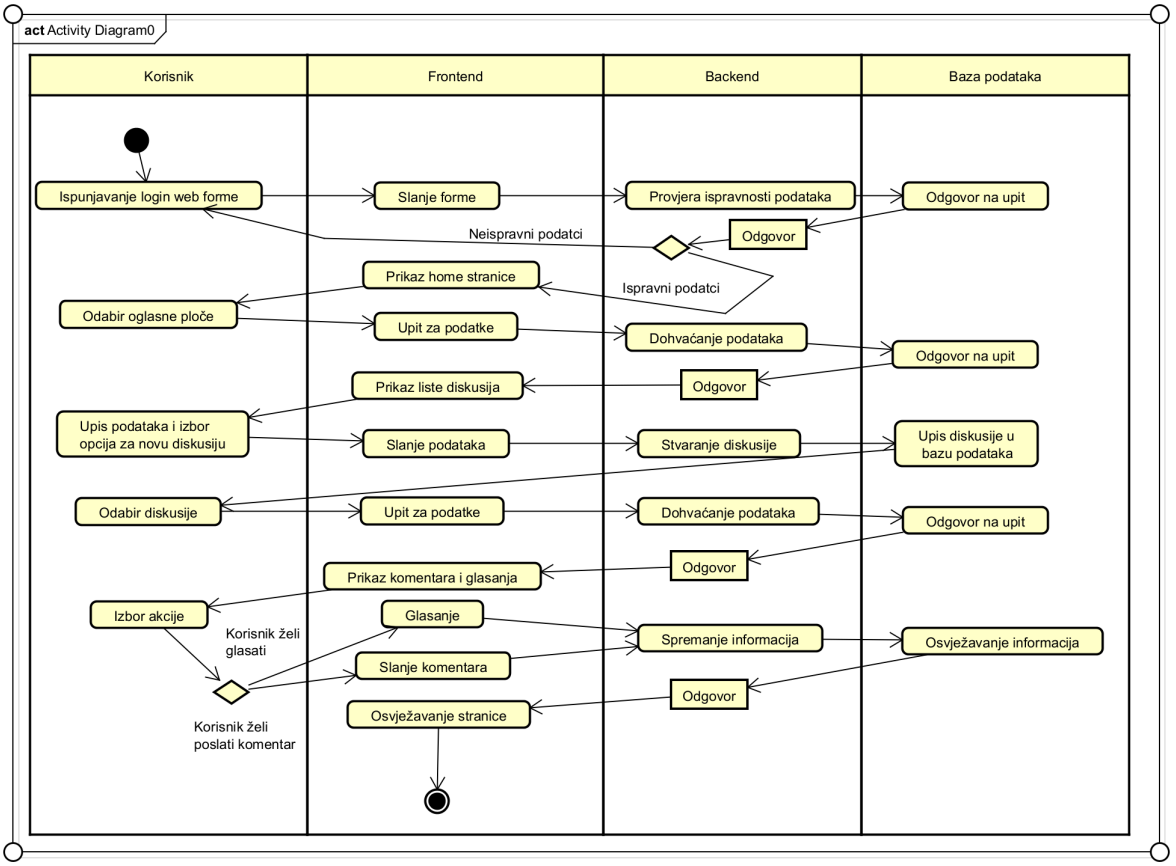
Ovaj sloj sadrži razrede koji upravljaju zahtjevima korisnika, pa je time najniži sloj u hijerarhiji. Razredi sadrže metode koje obrađuju zahtjeve korisnika, pozivaju metode sloja servisa i vraćaju odgovor korisniku.

4.3 Dijagram stanja



Prikazan dijagram stanja opisuje rad aplikacije koji počinje otvaranjem početne stranice koja ima dvije inačice i razlikuje se o ulozi korisnika. Običan korisnik odande može izabrati oglasnu ploču, pregledati diskusije na njoj i izabrati neku pojedinačnu diskusiju kako bi vidio detalje. U skolu diskusije vidi sve komentare i pitanja koji su postavljeni na njoj te može dodati svoj komentar, glasati na pitanje ili obrisati vlastiti komentar. Alternativno može otvoriti novu diskusiju i eventualno postaviti pitanje u sklopu nje. Admin, osim ovih mogućnosti, može pregledavati i listu oglasnih ploča i korisnika i upravljati njima.

4.4 Dijagram aktivnosti



Dijagram aktivnosti pokazuje aktivnosti koje sustav izvodi pri korištenju.

Prvo, korisnik se mora prijaviti upisivanjem korisničkog imena i lozinke ili korištenjem vanjskog servisa. Ispunjena forma se šalje poslužitelju koji traži podatke iz baze podataka i provjerava odgovaraju li upisani podaci traženima. Ako ne odgovaraju, traži se ponovni upis, a ako su podaci ispravni, korisniku se prikazuje početna stranica s listom dostupnih oglasnih ploča.

Korisnik odabire oglasnu ploču. Web klijent šalje upit poslužitelju za podacima diskusija unutar te oglasne ploče. Poslužitelj dohvaća podatke iz baze podataka, a zatim ih vraća web pregledniku koji ih ispisuje korisniku.

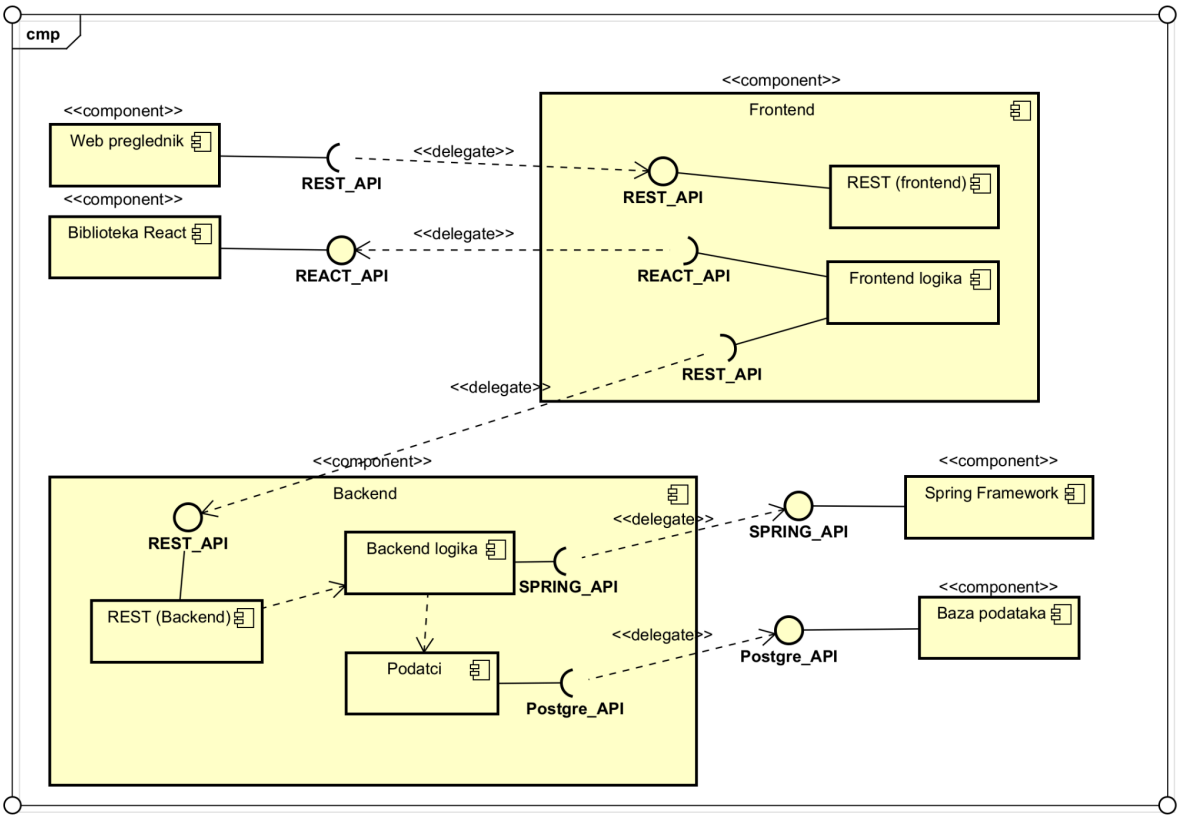
Korisnik otvara novu diskusiju upisom teme u okvir i izborom opcija. Ti se podaci šalju poslužitelju koji ih sprema u bazu podataka.

Korisnik zatim bira radnju: komentiranje ili glasanje. Ako bira komentirati, upisuje komentar koji šalje poslužitelju kako bi se spremio i prikazao drugim posjetiteljima stranice. Ako bira glasati - postupak je isti - samo što se umjesto teksta zapisuje broj pozitivnih i negativnih glasova.

Nakon svake radnje ažurira se sadržaj stranice.

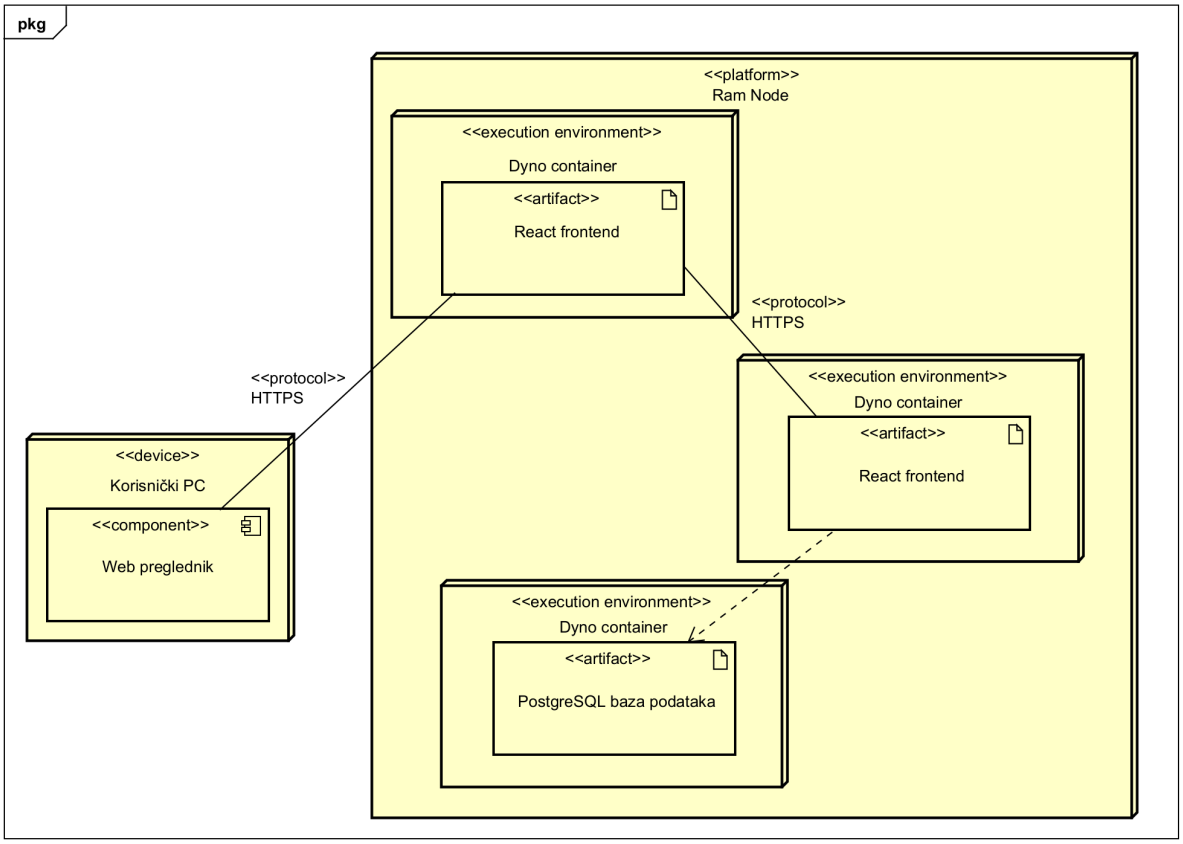
5. Arhitektura komponenata i razmještaja

5.1 Dijagram komponenata



Na slici prikazan je dijagram komponenata. Sustav je razdvojen u dvije komponente koje odgovaraju frontendu i backendu. Korisnik je u interakciji s frontentom, a zatim frontend komunicira s backendom kako bi ostvarili funkcionalnost sustava. Frontend je izgrađen koristeći REST API, a tako i komunicira s backendom koji je izgrađen koristeći radni okvir Spring Boot. Backend komunicira s bazom podataka koja se zasniva na temelju relacija.

5.2 Dijagram razmještaja



U dijagramu razmještaja vidljiva je arhitektura temeljena na sustavu klijent-poslužitelj. Klijentskom dijelu pristupa se web preglednikom na korisničkom računalu, a poslužitelj se nalazi na RamNode cloud hosting servisu. Protokol HTTPS se koristi za komunikaciju između poslužitelja i klijenata.

5.3 Implementacijski oblik

Naš sustav implementiran je na unajmljenom Virtual Private Serveru (VPS) pružatelja Ramnode. Na VPS-u je instaliran Ubuntu 22.04 (64-bit), a za web-server koristimo NGINX koji, osim posluživanja frontenda, radi kao reverse proxy za backend i upravlja SSL/TLS enkripcijom za HTTPS protokol.

5.3.1 Struktura

- **NGINX** prima HTTP/HTTPS zahtjeve (portovi 80 i 443 respektivno). Svi HTTP zahtjevi na port 80 se iz sigurnosnih razloga prepisuju u HTTPS zahtjeve na port 443.
- **Backend** sluša na istom poslužitelju na portu 8080. Zahtjevi koje NGINX dobije na putanju `/api/*` se prosljeđuju na backend putem *reverse proxy* mehanizma.
- **Frontend** se nalazi u folderu `/var/www/html` u obliku statičkih HTML/JS/CSS datoteka.

- **Baza podataka**, PostgreSQL 14, također se izvršava na istom poslužitelju i dostupna je na portu 5432 na *loopback* adresi. Baza pohranjuje korisničke podatke kako je opisano u poglavlju 4, *Arhitektura i dizajn sustava*.

5.3.2 Sigurnost i mrežna konfiguracija

- Korištenje HTTPS protokola osigurava kriptirani prijenos podataka između korisničkog preglednika i Nginx poslužitelja.
- VPS je konfiguriran s vatrozidom (UFW) tako da su otvoreni samo potrebni portovi (80 i 443).
- Interna komunikacija između backenda i PostgreSQL baze odvija se unutar samog VPS-a, čime se smanjuje vjerojatnost neovlaštenog pristupa.

5.3.3 Razlog izbora

- Ramnode VPS odabran je zbog svoje relativno pristupačne cijene za ovakav manji projekt.
- Ubuntu 22.04 se koristi zbog široke podrške i redovnih sigurnosnih ažuriranja, a NGINX jer je ispitano i pouzdano rješenje

5.3.4 Shema implementacije

1. **Korisnik** u pregledniku (na računalu ili mobilnom uređaju) otvara web stranicu putem https://projectbajeet.work.gd.
2. **NGINX** na Ramnode VPS-u prima zahtjeve i
    - statičke datoteke servira iz `/var/www/html/`, a
    - zahtjeve na `/api/` prosljeđuje na backend, na `localhost:8080`.
3. **Backend** obrađuje zahtjeve, po potrebi komunicira s **PostgreSQL** bazom i generira odgovarajuće odgovore.
4. Odgovor se vraća natrag preko Nginxa, a zatim putem HTTPS-a do korisničkog preglednika.

6. Ispitivanje programskog rješenja

Ovo poglavlje treba opissti provedena ispitivanja implementiranih funkcionalnosti na razini komponenti i sustava. Fokus je na odabiru i izvedbi ispitnih slučajeva koji obuhvaćaju redovne, rubne uvjete i testiranje grešaka, kao i upotrebu odgovarajućih alata za provedbu testiranja.

6.1 Ispitivanje komponenti

6.1.1 Izazivanje pogreške (Exception Throwing): Registracija već registriranim emailom

Ispitni slučaj:

- **Ulazni podaci:**
  - `username`: `"testKorisnik???"`
  - `lozinka`: `"testPassword???"`
- **Očekivani rezultati:**
  - Registracija neuspješna, izazivanje pogreške.
- **Dobiveni rezultat:**
  - Prolaz ispitivanja.

Obrazloženje:

Pokušaj registracije korisnika koji već postoji u bazi rezultira iznimkom `DataIntegrityViolationException`, što je očekivano ponašanje.

Kod:

```
@Test
void testExistingUser() {
    String username = "testKorisnik" + Math.random();
    String password = "testPassword" + Math.random();
    LoginRequest newUser = new LoginRequest(username, password);
    User testUser = authenticationService.signup(newUser);

    assertThrows(DataIntegrityViolationException.class, () -> {
        authenticationService.signup(newUser);
    });

    userRepo.deleteById(testUser.getUserId());
}
```

6.1.2 Registracija novog korisnika

Ispitni slučaj:

- **Ulazni podaci:**
  - `username`: `"testKorisnik???"`
  - `lozinka`: `"testPassword???"`
- **Očekivani rezultati:**
  - Uspješna registracija, novi korisnik dodan u bazu.
- **Dobiveni rezultat:**
  - Prolaz ispitivanja.

Obrazloženje:

Novog korisnika uspješno dodajemo u bazu te provjeravamo je li kreiran sa svim očekivanim atributima.

Kod:

```
@Test
void testNewUser() {
    String username = "testKorisnik" + Math.random();
    String password = "testPassword" + Math.random();
    LoginRequest newUser = new LoginRequest(username, password);

    User testUser = authenticationService.signup(newUser);

    Assertions.assertThat(testUser).isNotNull();
    Assertions.assertThat(testUser.getUsername()).isEqualTo(username);

    userRepo.deleteById(testUser.getUserId());
}
```

6.1.3 Brisanje korisnika

Ispitni slučaj:

- **Ulazni podaci:**
  - `username`: `"testKorisnik???"`
  - `lozinka`: `"testPassword???"`
- **Očekivani rezultati:**
  - Korisnik uspješno obrisan iz baze.
- **Dobiveni rezultat:**
  - Prolaz ispitivanja.

Obrazloženje:

Korisnika dodajemo u bazu, brišemo ga te provjeravamo je li uspješno uklonjen.

Kod:

```
@Test
void testDeleteUser() {
    String username = "testKorisnik" + Math.random();
    String password = "testPassword" + Math.random();
    LoginRequest newUser = new LoginRequest(username, password);

    User testUser = authenticationService.signup(newUser);

    userRepo.deleteById(testUser.getUserId());


Assertions.assertThat(userRepo.findByUsername(newUser.getUsername()).isPresent()).isF
alse();
}
```

6.1.4 Pronalaženje korisnika prema korisničkom imenu

Ispitni slučaj:

- **Ulazni podaci:**
  - `username`: `"testKorisnik???"`
  - `lozinka`: `"testPassword???"`
- **Očekivani rezultati:**
  - Uspješno pronalaženje korisnika u bazi.
- **Dobiveni rezultat:**
  - Prolaz ispitivanja.

Obrazloženje:

Dodani korisnik uspješno je pronađen putem metode `findByUsername`.

Kod:

```
@Test
void testFindByUsername() {
    String username = "testKorisnik" + Math.random();
    String password = "testPassword" + Math.random();
    LoginRequest newUser = new LoginRequest(username, password);
```

```
    User testUser = authenticationService.signup(newUser);

    Optional<User> foundUser = userRepo.findByUsername(username);

    Assertions.assertThat(foundUser.isPresent()).isTrue();
    Assertions.assertThat(foundUser.get().getUsername()).isEqualTo(username);

    userRepo.deleteById(testUser.getUserId());
}
```

6.1.5 Testiranje objekta "Board"

Ispitni slučaj:

- **Ulazni podaci:**
  - `Board` s adresom `"Adresa testiranja"`.
- **Očekivani rezultati:**
  - Spremanje i pronalaženje objekta `Board`.
- **Dobiveni rezultat:**
  - Prolaz ispitivanja.

Obrazloženje:

`Board` objekt uspješno se sprema, pronalazi i briše iz baze, pri čemu se sve promjene provjeravaju.

Kod:

```
@Test
void testBoard() {
    Board board = new Board();

    board.setAddress("Adresa testiranja");

    Board savedBoard = boardRepo.save(board);

    Optional<Board> foundBoard = boardRepo.findById(savedBoard.getBoardID());

    Assertions.assertThat(foundBoard.isPresent()).isTrue();
    Assertions.assertThat(foundBoard.get().getAddress()).isEqualTo(board.getAddress());

    boardRepo.deleteById(savedBoard.getBoardID());
```

```
Assertions.assertThat(boardRepo.findById(savedBoard.getBoardID()).isPresent()).isFalse();
}
```

6.1.6 Testiranje objekta "Thread"

Ispitni slučaj:

- **Ulazni podaci:**
  - `Thread` s naslovom `"Naslov testiranja threada"` i opisom `"Opis testiranja threada"`.
- **Očekivani rezultati:**
  - Spremanje i pronalaženje objekta `Thread` unutar `Board` objekta.
- **Dobiveni rezultat:**
  - Prolaz ispitivanja.

Obrazloženje:

`Thread` objekt se uspješno povezuje s `Board`, spremanjem u bazu i kasnijim pronalaženjem.

Kod:

```
@Test
void testThread() {
    Board board = new Board();

    board.setAddress("Adresa testiranja threada");

    Thread thread = new Thread();

    thread.setDescription("Opis testiranja threada");
    thread.setHasVoting(true);
    thread.setTitle("Naslov testiranja threada");
    thread.setPublic(true);

    List<Thread> threadovi = new ArrayList<>();
    threadovi.add(thread);
    board.setThreads(threadovi);

    Board savedBoard = boardRepo.save(board);

    Optional<Thread> foundThread =
threadRepo.findById(savedBoard.getThreads().get(0).getThreadID());
```

```
    Assertions.assertThat(foundThread.isPresent()).isTrue();
    Assertions.assertThat(foundThread.get().getTitle()).isEqualTo(thread.getTitle());

Assertions.assertThat(foundThread.get().getDescription()).isEqualTo(thread.getDescription()
);

    boardRepo.deleteById(savedBoard.getBoardID());


Assertions.assertThat(boardRepo.findById(savedBoard.getBoardID()).isPresent()).isFalse();
}
```

8.1.7 Testiranje objekta "Message"

Ispitni slučaj:

- **Ulazni podaci:**
  - `Message` s autorom i sadržajem `"Testiranje poruke"`.
- **Očekivani rezultati:**
  - Spremanje i pronalaženje objekta `Message` unutar `Thread` i `Board` objekta.
- **Dobiveni rezultat:**
  - Prolaz ispitivanja.

Obrazloženje:

`Message` objekt se uspješno povezuje s `Thread` i `Board` te je moguće dohvatiti sadržaj poruke iz baze.

Kod:

```
@Test
void testMessage() {
    Board board = new Board();

    board.setAddress("Adresa testiranja poruke");

    Thread thread = new Thread();

    thread.setDescription("Opis testiranja poruke");
    thread.setHasVoting(true);
    thread.setTitle("Naslov testiranja poruke");
    thread.setPublic(true);
```

```
    String username = "testKorisnik" + Math.random();
    String password = "testPassword" + Math.random();
    LoginRequest newUser = new LoginRequest(username, password);

    User testUser = authenticationService.signup(newUser);

    Message message = new Message();

    message.setMessageAuthor(testUser);

    String content = "Testiranje poruke";

    message.setContent(content);
    message.setHasVoting(false);

    List<Message> messages = new ArrayList<>();
    messages.add(message);

    thread.setComments(messages);

    List<Thread> threadovi = new ArrayList<>();
    threadovi.add(thread);
    board.setThreads(threadovi);

    Board savedBoard = boardRepo.save(board);

    Optional<Message> foundMessage =
messageRepo.findById(savedBoard.getThreads().get(0).getComments().get(0).getMessageI
d());

    Assertions.assertThat(foundMessage.isPresent()).isTrue();
    Assertions.assertThat(foundMessage.get().getContent()).isEqualTo(content);

    boardRepo.deleteById(savedBoard.getBoardID());


Assertions.assertThat(messageRepo.findById(savedBoard.getThreads().get(0).getComment
s().get(0).getMessageId()).isPresent()).isFalse();
}
```

6.2 Ispitivanje sustava

6.2.1 Prvi ispitni slučaj

1. Ulazi:
- Prijava kao korisnik `t1` sa lozinkom `t`
- Kreiranje novog Boarda sa nazivom `selenium test` i odabiranje svih korisnika
1. Koraci ispitivanja:
- Prijava
- Stvaranje novog Boarda
- Odjava
1. Očekivani izlaz:
- Stvoren novi board za sve korisnike
1. Dobiveni izlaz: Prolaz ispitivanja.

6.2.2 Drugi ispitni slučaj

1. Ulazi:
- Prijava kao korisnik `t2` sa lozinkom `t`
- Kreiranje novog Threada u prethodno napravljenom boardu sa nazivom `thread` i opisom `opis`
- Postavljanje pitanja sa tekstom `pitanje` u novonapravljenom threadu
1. Koraci ispitivanja:
- Prijava
- Stvaranje novog Threada
- Postavljanje pitanja
- Odjava
1. Očekivani izlaz:
- Stvoren novi javni thread za sve korisnike boarda sa postavljenim pitanjem
1. Dobiveni izlaz: Prolaz ispitivanja.

6.2.3 Treći ispitni slučaj

1. Ulazi:
- Prijava kao korisnik `t3` sa lozinkom `t`
- Ulaz u prethodno napravljeni board te thread
- Glasanje na postavljeno pitanje
- Komentar sa tekstom `svida mi se`
1. Koraci ispitivanja:
- Prijava
- Ulaz u thread
- Glasanje na postavljeno pitanje
- Postavljanje komentara
- Odjava
1. Očekivani izlaz:
- Stvoren novi board za sve korisnike
1. Dobiveni izlaz: Prolaz ispitivanja.

6.2.4 Četvrti ispitni slučaj

1. Ulazi:

- Prijava kao korisnik `t1` sa lozinkom `t`
- Promjena uloge korisnika `t5` iz `TENANT` u `ADMIN`

1. Koraci ispitivanja:

- Prijava
- Navigacija na stranicu na kojoj se upravlja korisnicima
- Promjena uloge korisnika
- Odjava

1. Očekivani izlaz:

- Stvoren novi board za sve korisnike

1. Dobiveni izlaz: Prolaz ispitivanja.

7. Tehnologije za implementaciju aplikacije

7.1 Programski jezici:

- **Java** 17 (https://www.oracle.com/java/)

  Java je objektno orijentirani programski jezik koji je korišten za poslužiteljsku stranu. Jednostavan je za naučiti, lako se koristi i u širokoj je primjeni. Razvojna okruženja dodatno olakšavaju rad s njim, a i povezivanje s bazom podataka i ostalim servisima je vrlo jednostavno.

- **JavaScript** ES2023 (https://www.javascript.com/)

  JavaScript je također objektno orijentirani jezik, korišten u kombinaciji s HTML-om i CSS-om. Koristi se prvenstveno za web aplikacije, a u tom kontekstu je korišten i u ovoj aplikaciji. Omogućuje interaktivnost i dinamičnost web stranica, a u kombinaciji s Reactom olakšava izradu frontenda.

- **HTML** 5 (https://html.com/)

  HTML je jezik koji se koristi za oblikovanje stranice i organizaciju komponenti. Standard je za svoju svrhu i u kombinaciji s CSS-om omogućuje stvaranje vizualno zanimljivih web-stranica koje čine korisničko sučelje odnosno klijentsku stranu aplikacije.

- **CSS** 3 (https://css3.com/)

  CSS oblikuje elemente stranice, određuje njihovu poziciju i izgled te se, kao i HTML, koristi za dizajn klijentske strane aplikacije. Cilj je približiti stranicu korisniku te jednostavnim i intuitivnim dizajnom stranice koji se postiže uporabom CSS-a olakšati njeno korištenje.

7.2 Radni okviri i biblioteke:

- **React** 18.3.1 (https://react.dev/)

  Klijentski dio aplikacije razvijen je koristeći React, JavaScript biblioteku koja omogućuje građenje sučelja iz komponenti. Na taj način komponente se mogu ponovo koristiti i kombinirati s postojećima kako bi se stvorile interaktivne web stranice. Korištenje alata olakšava uređivanje frontenda te stoga pozitivno doprinosi efikasnosti razvoja aplikacije.

- **Tailwind CSS** 3.4.14 (https://tailwindcss.com/)

  Tailwind je korišten za oblikovanje izgleda stranice. Nudi lakši način za dizajnirati stranicu tako što drži CSS i HTML povezanima. Na taj način olakšava unošenje promjena, pozicioniranje i oblikovanje objekata koji su vidljivi na stranici.

- **Axios** 1.7.7 (https://axios-http.com/docs/intro)

  Axios je HTTP klijent osnovan na temelju JavaScriptovih Promise API-jeva. Korišten je za slanje i prihvaćanje HTTP zahtjeva. Omogućuje direktno pretvaranje zahtjeva u formate u kojima ih je lakše obraditi, kao što su JSON, Multipart / FormData i drugi. Uključuje i funkcionalnosti automatske obrade zahtjeva i postavljanje vremenskih ograničenja.

- **Vite** 5.4.10 (https://vite.dev/)

  Vite je razvojni okvir za izgradnju web aplikacija čije su prednosti brzina, efikasnost i fleksibilnost s raznim dodacima. Korišten je kao baza na koju se lako može nadograditi i u kombinaciji s ostalim korištenim alatima izgraditi web stranica koja odgovara zahtjevima.

7.3 Baza podataka:

- **PostgreSQL** 17 (https://www.postgresql.org/)

  PostgreSQL baza je izabrana jer omogućuje sigurnu, efikasnu i pouzdanu pohranu podataka. Uputstva za uporabu su lako dostupna što ju čini pristupačnom, a i iznimno moćnom što se ističe prilikom rukovanja s velikom količinom podataka.

7.4 Razvojni alati:

- **Visual Studio Code** 1.96.2 (https://code.visualstudio.com)

  VS Code omogućuje brzo i jednostavno uređivanje i pisanje koda. Podržava brojna proširenja i jezike što ga čini jasnim izborom za razvojno okruženje. Olakšava pronalaženje grešaka, testiranje, i kombiniranje komponenti, što ga i čini jednim od najpopularnijih pomagala za programere.

- **Spring Boot** 3.0 (https://spring.io/projects/spring-boot)

  Spring je razvojno okruženje korišteno za backend aplikacije. Odlikuje se jednostavnošću i širinom primjene, a zbog toga je i izabran za izradu ove aplikacije. Zbog česte korištenosti materijali i uputstva za korištenje lako su dostupni, a rad u njemu uvelike olakšava razvoj aplikacije u odnosu na rad bez razvojnog okruženja.

- **GitHub** (https://github.com)

  GitHub olakšava upravljanje projektom i njegovim komponentama, osigurava podršku za zajednički rad i olakšava upravljanje inačicama. Okosnica je projekta zbog sigurnosti i pristupačnosti, a odlikuje se i izvrsnim alatima za praćenje napretka i doprinosa članova tima. Osigurava i pohranu koda na cloudu što smanjuje rizike povezane s gubljenjem napretka.

7.5 Alati za ispitivanje:

- **Selenium** 3.17.4 (https://www.selenium.dev/)

  Selenium je sustav za automatsko ispitivanje web stranica koji je korišten za ispitivanje stranice. Sukladno toplim preporukama mentora, izabran je zbog kompatibilnosti s velikim brojem programskih jezika, mogućnosti izvođenja brojnih integracijskih testova i dostupnosti materijala i uputa za korištenje.

7.6 Alati za razmještaj:

- **Maven** 5.0.0 (https://www.npmjs.com/package/maven) Maven je Node.js paket koji se koristi pri prevođenju backenda koji se pokreće s Javom na poslužitelju. Glavne su mu prednosti što osigurava standardiziranu i organiziranu strukturu direktorija i što pojednostavljuje rad s ovisnostima paketa koji se koriste. Organiziranost i strukturiranost je važna za uspjeh projekta, a baš to su prednosti mavena.
- **Node Package Manager** (https://www.npmjs.com/) NPS je repozitorij paketa dostupnih za korištenje u sklopu Node.jsa. Omogućuje korištenje već napisanog koda i time smanjuje opseg koda koji treba biti napisan u sklopu projekta. Osim očitog razloga koji je korištenje i upravljanje paketima, NPM je u širokoj uporabi zbog čega su dostupni mnogi materijali koji olakšavaju korištenje.
- **NGINX** 1.18.0 (https://nginx.org/en/) NGINX je open-source serversko sučelje. Podržava komunikaciju HTTPS protokolom, a koristi se i kao proxy server za komunikaciju e-mailovima. Prednosti su mu stabilnost i performanse koje ga čine odgovarajućim izborom za ovaj projekt.

7.7 Cloud platforma:

- **RamNode** (https://ramnode.com/)

  Cloud hosting servis RamNode je izabran zbog brzog deploya, jednostavnosti i fleksibilnosti. Na njemu je deploy aplikacije.

8. Upute za puštanje u pogon

8.1 Instalacija

- Preduvjeti *(zvjezdica označava build-time preduvjete)*:
  - backend:
    - JDK 17
    - Apache Maven (*)
    - Postgres >=16
  - frontend:
    - NodeJS >=18 (*)
    - NPM (Node Package Manager) (*)
    - reverse proxy po odabiru, mi ćemo u ovim uputama koristiti NGINX

8.1.2 Backend

Prvo je potrebno klonirati repozitorij:

git clone --depth=1 <https://github.com/ProjectBajeet/Project_Bajeet>
cd Project_Bajeet/IzvorniKod

Build backenda:

mvn clean package -Dmaven.test.skip=true

Izlazna .jar datoteka nalazit će se u folderu `target`:

$ ls -lah target/*jar
-rw-r--r-- 1 root root 54M Jan 13 15:48 target/project_bajeet-0.0.1-SNAPSHOT.jar

8.1.2 Frontend

Izvorni kod frontenda nalazi se u mapi `IzvorniKod/frontend`.

cd frontend
npm install
npx vite build

U folderu `build` nalazit će se root web direktorij, kojeg je potrebno učiniti dostupnim koristeći web server po izboru.

```
$ ls -lah build/
total 28K
drwxr-xr-x 3 root root 4.0K Jan 13 16:59 .
drwxr-xr-x 7 root root 4.0K Jan 13 16:59 ..
drwxr-xr-x 2 root root 4.0K Jan 13 16:59 assets
-rw-r--r-- 1 root root 2.7K Jan 13 16:59 diskusije.js
-rw-r--r-- 1 root root  394 Jan 13 16:59 index.html
-rw-r--r-- 1 root root  198 Jan 13 16:59 korisnikInfo.js
-rw-r--r-- 1 root root 1.5K Jan 13 16:59 vite.svg
```

8.2 Postavke

8.2.1 Postgres

U Postgresu potrebno je kreirati korisnika i bazu podataka. Shemu baze će aplikacija automatski kreirati (vidi odjeljak Baza podataka).

Na primjer, ako smo se odlučili za pristupne podatke `postgres:postgres123`, a za ime baze odabrali `projectbajeet`, ta konfiguracija bi izgledala ovako:

```
psql -U postgres -c "ALTER USER postgres WITH PASSWORD 'postgres123';"
psql -U postgres -c "CREATE DATABASE projectbajeet;"
```

8.2.2 Backend

Konfiguracija backenda aplikacije se nalazi u datoteci `IzvorniKod/src/main/resources/application.yml`. U nastavku su objašnjene stavke bitne za trenutnu konfiguraciju.

Ipak, kako se radi o Spring Boot aplikaciji, moguće je konfigurirati bilo koju od ovdje navedenih opcija.

8.2.3 Konfiguracija slušatelja

```
server:
  port: 8080
  address: 127.0.0.1
```

U odjeljku `server` moguće je konfigurirati Tomcat poslužitelja. Ovo je potrebno podesiti ovisno o konfiguraciji reverse proxyja.

Po zadanom, frontend očekuje da će backend biti dostupan po endpointovima `/api/`. Iz tog razloga backend nije potrebno (niti je preporučeno) učiniti dostupnim na vanjskoj adresi.

8.2.4 Konfiguracija Spring okruženja

```
spring:
  datasource:
    platform: postgres
    driverClassName: org.postgresql.Driver
    # ovo promijeniti prema konfiguraciji Postgres servera
    url: jdbc:postgresql://<address>:<port>/<dbname>
    username: <username>
    password: <password>
  sql:
    init:
      mode: always
  jpa:
    defer-datasource-initialization: true
    hibernate:
      # ovo promijeniti ovisno o okruženju
      ddl-auto: create-drop
    show-sql: true
    properties:
      hibernate:
        format_sql: true
        use_nationalized_character_data: true
        enable_lazy_load_no_trans: true
  application:
    name: ProjectBajeet
  security:
    oauth2:
      client:
        registration:
          google:
            client-id: ${GOOGLE_CLIENT_ID}
            client-secret: ${GOOGLE_CLIENT_SECRET_ID}
            redirect-uri: <https://projectbajeet.work.gd/api/oauth2/code/google>
            scope:
              - email
              - profile
          github:
            client-id: ${GITHUB_CLIENT_ID}
            client-secret: ${GITHUB_CLIENT_SECRET_ID}
            redirect-uri: <https://projectbajeet.work.gd/api/oauth2/code/github>
            scope:
```

- user:email
- read:user

8.2.5 Baza podataka

Baza podataka se konfigurira u odjeljku `datasource`. Potrebno je izmijeniti podatke poslužitelja:

- `<address>:<port>` - adresa i port Postgres servera (npr. `localhost:5432`)
- `<dbname>` - ime baze u Postgres serveru (npr. `projectbajeet`)
- `<username>` i `<password>` - pristupni podaci Postgres servera

Također je potrebno konfigurirati `jpa.hibernate.ddl-auto`, što kontrolira na koji način će se baza podataka kreirati ili uništiti pri pokretanju aplikacije.

- za **development okruženja** dobro je tu vrijednost postaviti na `create-drop`, što će stvoriti bazu pri pokretanju i uništiti ju pri gašenju
    - svako pokretanje će biti *"clean slate"*
- za **produkcijska okruženja** ovu vrijednost se preporučuje postaviti na `update`, što će kreirati shemu pri prvom pokretanju, a pri svakom sljedećem samo ažurirati ako dođe do promjena

Mijenjanje ostalih postavki vezanih za bazu podataka nije podržano i ne smatra se važećom konfiguracijom.

8.2.6 OAuth konfiguracija

Aplikacija podržava prijavu putem GitHuba i Googlea. Za ostvarenje ove funkcionalnosti potrebno je od navedenih servisa dobiti OAuth tokene.

Upute za dobivanje ovih tokena za GitHub su ovdje, a za Google su ovdje.

Te tokene je zatim potrebno učiniti dostupnima aplikaciji pomoću varijabla okruženja:

- `GOOGLE_CLIENT_ID` i `GOOGLE_CLIENT_SECRET_ID` za Google
- `GITHUB_CLIENT_ID` i `GITHUB_CLIENT_SECRET_ID` za GitHub.

8.2.7 Frontend / NGINX konfiguracija

Frontend nema konfiguracijske datoteke - bitno je samo da može pristupiti backendu preko `/api` endpointova. Za ostvarivanje ove funkcionalnosti može se koristiti bilo koji web server koji ima mogućnost reverse proxyja. U ovim uputama koristiti ćemo NGINX.

Pretpostavljajući da se datoteke frontenda nalaze u `/var/www/html`, osnovna konfiguracija bi izgledala ovako:

<details> <summary>Otvori primjer NGINX konfiguracije</summary>

```
## Na Debian ili Ubuntu, ova datoteka ide u:
## /etc/nginx/sites-available/projectbajeet
## I aktivira se komandom:
## ln -s /etc/nginx/sites-available/projectbajeet /etc/nginx/sites-enabled/
server {
  listen 80 default_server;

  # Ovo odkomentirati za SSL
  # listen 443 ssl;
  # ssl_certificate /etc/ssl/cert.crt;
  # ssl_certificate_key /etc/ssl/cert.key;

  # Isključujemo stvari koje nam nisu potrebne
  sendfile off;
  etag off;
  server_tokens off;

  # Folder u kojem su datoteke frontenda
  root /var/www/html;
  index index.html;

  # Ako je SSL uključen, i/ili ovaj server ima više
  # virtualnih hostova, server_name mora biti postavljen
  # na odgovarajuću domenu. U bilo kojem drugom slučaju je
  # ovo u redu.
  server_name _;

  ## Frontend datoteke
  location / {
    try_files $uri $uri/ =404;
  }

  ## Proxy za backend
  location /api/ {
    # Pretpostavka da backend sluša na portu 8080
    proxy_pass <http://localhost:8080/>;
    # Klasična NGINX konfiguracija za proxy
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection 'upgrade';
    proxy_set_header Host $host;
    proxy_cache_bypass $http_upgrade;
  }
}
```

</details>

8.3 Pokretanje aplikacije

8.3.1 Frontend

8.3.1.1 Razvojno okruženje

Za pokretanje razvojne verzije koja se automatski osvježava s promjenama potrebno je pokrenuti sljedeće:

npx vite dev

Razvojna verzija frontenda će biti dostupna na `http://localhost:3000`.

8.3.1.2 Produkcijsko okruženje

Pretpostavlja se korištenje ranije navedene NGINX konfiguracije.

Nakon što je frontend buildan (prema uputama u odjeljku Instalacija), potrebno je premjestiti datoteke u odgovarajući folder:

Ovisno o dopuštenjima, možda će biti potrebno koristiti sudo ispred cp
cp -r build/* /var/www/html/

8.3.2 Backend

8.3.2.1 Razvojno okruženje

Razvojna verzija aplikacije, sa automatskim osvježavanjem s promjenama, pokreće se ovako:

mvn spring-boot:run -Dmaven.test.skip=true

Backend će biti dostupan na `http://localhost:8080`, osim ako je konfiguriran drugačije (vidi sekciju Konfiguracija slušatelja).

8.3.2.2 Produkcijsko okruženje

Prvo je potrebno buildati zadnju verziju aplikacije, prema uputama u odjeljku Instalacija. Zatim, .jar datoteku je potrebno premjestiti na željeno mjesto:

cp target/project_bajeet-0.0.1-SNAPSHOT.jar /opt/project_bajeet.jar

Nakon toga, potrebno je napraviti *systemd* servis koji će pokretati aplikaciju. Poželjno je aplikaciju pokrenuti pod nekim ne-root korisnikom, kojeg je moguće kreirati ovako:

adduser --system --no-create-home projectbajeet

Primjer systemd servisa, koji se može kreirati u `/etc/systemd/system/projectbajeet.service`:

```
[Unit]
Description=Project Bajeet
After=syslog.target

[Service]
User=projectbajeet
ExecStart=/usr/bin/java -jar /opt/project_bajeet.jar
SuccessExitStatus=143
TimeoutStopSec=10
Restart=always
RestartSec=30

[Install]
WantedBy=multi-user.target
```

Nakon što je servis kreiran, potrebno ga je pokrenuti:

```
# Učitava naš novi systemd servis
systemctl daemon-reload
# Omogućuje automatsko pokretanje servisa pri bootanju servera
systemctl enable projectbajeet
# Pokreće servis
systemctl start projectbajeet
```

8.4 Upute za administratore

Smjernice za administratore aplikacije nakon puštanja u pogon:

8.4.1 Pristup administratorskom sučelju

- URL za admin panel (npr. /admin).
- Početni podaci za prijavu (ako postoje).

8.5 Redovito održavanje

Kao administrator aplikacije, imate odgovornost održavanja aplikacije i osiguravanja njezine dostupnosti. To uključuje:

- Redovito ažuriranje aplikacije.
- Redovito ažuriranje operativnog sustava.
- Redovito arhiviranje baze podataka.
- Redovito pregledavanje logova.
- Redovito testiranje sigurnosnih kopija.

8.5.1 Ažuriranje aplikacije

Ažuriranje aplikacije može se obaviti na sljedeći način:

```
# Pretpostavljamo da je ovo mjesto gdje se nalazi klonirani repozitorij
cd Project_Bajeet
# Preuzimanje najnovijih promjena
git pull origin main

# Ako je potrebno, ažurirati backend
cd IzvorniKod
mvn clean package -Dmaven.test.skip=true
cp target/project_bajeet*.jar /opt/project_bajeet.jar

# Ako je potrebno, ažurirati frontend
cd frontend
npm i && npx vite build
cp -r build/* /var/www/html/

# Restartati aplikaciju
systemctl restart projectbajeet
```

8.5.2 Provjera logova

Kako je backend pokrenut kao systemd servis, logovi backenda se mogu pregledati pomoću naredbe `journalctl`:

```
journalctl -u projectbajeet
```

Logovi NGINX-a (frontenda) nalaze se u `/var/log/nginx/error.log` i `/var/log/nginx/access.log`:

```
tail -f /var/log/nginx/error.log
# ili
tail -f /var/log/nginx/access.log
```

8.5.3 Arhiviranje baze podataka

Arhiviranje baze podataka može se obaviti na sljedeći način:

```
pg_dump -U postgres projectbajeet > projectbajeet.sql
```

Moguće je i automatsko arhiviranje baze podataka, npr. pomoću `cron` poslova.

Redovito testiranje sigurnosnih kopija je ključ pri održavanju aplikacije. Sigurnosne kopije se mogu testirati na sljedeći način (naravno, ne na produkcijskom sustavu):

```
# Vraćanje baze podataka iz sigurnosne kopije
psql -U postgres projectbajeet < projectbajeet.sql
```

8.6 Opis pristupa aplikaciji na javnom poslužitelju

Aplikacija je dostupna na javnom poslužitelju na adresi https://projectbajeet.work.gd. Na početnoj stranici nalazi se prijava i registracija korisnika. Nakon prijave korisniku je omogućen pristup svim funkcionalnostima aplikacije.

9. Zaključak i budući rad

Zadatak projektnog tima bio je razvoj web aplikacije StanBlog čija je svrha omogućiti jednostavnu komunikaciju suvlasnika zgrada. Aplikacija služi kao online oglasna ploča. Registrirani korisnici mogu pristupiti oglasnim pločama zgrada u kojima imaju suvlasništvo. Ondje mogu pregledati diskusije i započeti svoje. Svaka diskusija je zasebni kanal u kojem se može dodavati mišljenja u obliku komentara ili glasati na postavljena pitanja.

Tijekom semestra, kao i svaki tim, i naš je tim prošao kroz različite faze timskog rada. Prva faza bila je okupljanje tima (forming) u kojem smo razgovorom došli do okvirne podjele poslova, koja se kasnije u maloj mjeri mijenjala.

U drugoj fazi tima naišli smo na prve sukobe i probleme (storming). Iskustva i metode rada svih članova su tima bile različite što je dovelo do nesuglasica. Ovaj suboptimalan period timskog rada obilježile su nesuglasice oko organizacije koda i podjele posla, koja se na prvu činila nepoštena. Situacija se popravila tek krajem prve faze, kada smo komunikacijom postigli dogovore i prionuli na posao. Prvi je dio dokumentacije napisan prije same izrade aplikacije u ovoj fazi, te je na temelju nje izgrađen kostur aplikacije.

Treću fazu karakterizira konačno slaganje tima i prihvaćanje uloga u njemu (norming). Kako bismo odradili sve što je potrebno za prvu predaju, sredinom semestra bili smo prisiljeni skupiti se kao tim, prijeći preko razmirica i surađivati kako bismo napredovali prema cilju. U ovoj se fazi odvijala implementacija osnovnih funkcionalnosti, dok je dokumentacija osvježena na temelju stečenog uvida u kod.

Konačno, četvrta i posljednja faza prije završetka projekta je faza produktivnosti (performing). Svaki je član tima prionuo na svoj posao i radio kako bismo uspješno izvršili projektni zadatak. U ovoj su fazi implementirane preostale funkcionalnosti, dovršena je dokumentacija i obavljeno je testiranje. Na kraju su spojene sve komponente u jednu cjelinu.

U budućnosti preostaje još i faza raspuštanja tima (adjourning), no ona dolazi tek kada je projekt uspješno predan i završen.

Dvije funkcionalnosti nisu implementirane u konačnoj inačici aplikacije, a to su:

1) preuzimanje liste glasanja s pozitivnih ishodom

2) određivanje maksimalnog broja poruka u diskusiji kao funkcionalnog parametra diskusije

Ovaj je projekt bio prilika za učenje rada u timskom okruženju, rješavanje sukoba i suradnju. Te su vještine jednako važne kao i tehnički aspekti, poput pisanja dokumentacije, izrade dijagrama i rada na projektu s ograničenim vremenskim okvirom. Unatoč postignutom napretku, postoji prostor za unaprjeđenje komunikacije i upravljanja timom, što bi u budućnosti dodatno povećalo učinkovitost rada.

Također, projekt je mnogima bio uvod u korištenje GitHuba i prilika za naučiti raditi s UML dijagramima_dvije vještine neophodne za daljnji razvoj mladih perspektivnih inženjera.

U budućnosti aplikacija bi se mogla nadograditi s dodatkom mape koja prikazuje lokacije zgrada uz oglasne ploče, različitim kategorijama diskusija ili mogućnošću arhiviranja diskusija čime ih se zatvara bez gubitka sadržaja. Iako ove ideje premašuju opseg trenutačnog projekta, predstavljaju smjer u kojem bi aplikacija mogla rasti.

Na kraju, ponosni smo na ono što smo postigli i vjerujemo da smo postavili dobar temelj za buduće projekte.

A. Dnevnik promjena dokumentacije

Rev.	Opis promjene/dodatka	Autori	Datum
0.1	Napravljena početna stranica i podnožje	Bruno Ševčenko	24.10.2024.
0.2	Dodani predlošci svih poglavlja Opisani funkcionalni zahtjevi projekta	Bruno Ševčenko	25.10.2024.
0.3	Dodani i opisani obrasci uporabe	Bruno Ševčenko	26.10.2024.
0.4	Dodani ostali zahtjevi Ispravak numeriranja funkcionalnih zahtjeva	Bruno Ševčenko Jelena Glasovac	27.10.2024.
0.5	Inicijalni draft dijagrama obrazaca uporabe i sekvencijskih dijagrama	Jelena Glasovac	30.10.2024.
0.6	Dodan opis projektnog zadatka	Bruno Ševčenko	7.11.2024.
0.7	Dodan opis arhitekture sustava Dodane sheme baze podataka Dodan opis baze podataka i pripadajuće tablice relacija Formatiran i prenesen na GitHub dnevnik sastajanja Ažurirana tablica aktivnosti (za neke članove)	Jelena Glasovac	10.11.2024.
0.8	Izmijenjen opis baze podataka i pripadajući dijagrami Ažurirana tablica aktivnosti	Jelena Glasovac	11.11.2024.
0.9	Dodan dijagram razreda i njegov opis Izmijenjen opis baze podataka i pripadajući dijagrami Ažurirana tablica aktivnosti	Jelena Glasovac	12.11.2024.
0.9.1	Izmijenjena sekcija „Opis projektnog zadatka“ radi detaljnijeg opisa	Bruno Ševčenko	12.11.2024.
1.0	Restrukturiranje dokumentacije prema novom predlošku (dodan odjeljak "Analiza zahtjeva") Ažuriran dijagram razreda Verzija prve predaje	Bruno Ševčenko Jelena Glasovac	15.11.2024.
1.1	Popisani alati i tehnologije korišteni prilikom rada na projektu i dodani njihovi opisi, ispravljen pravopis u nekim poglavljima	Jelena Glasovac	9.1.2025.

1.2	Dodani dijagrami aktivnosti, komponenata i razmještaja Ažuriran dnevnik sastajanja Ažurirana tablica aktivnosti	Jelena Glasovac	15.1.2025.
1.2.1	Dodani opisi dijagrama aktivnosti, komponenata i razmještaja Ažurirana tablica aktivnosti	Jelena Glasovac	17.1.2025.
1.3	Dodane upute za puštanje u pogon	Bruno Ševčenko	17.1.2025.
1.4	Dodan zaključak	Jelena Glasovac	19.1.2025.
1.5	Dodan implementacijski oblik Dodane upute za build backenda u 8. poglavlje Dodani UCovi za lokalnu prijavu i registraciju	Bruno Ševčenko	20.1.2025.
1.5.1	Dodani ključni izazovi i rješenja	Jelena Glasovac	22.1.2025.
1.6	Dodane upute za produkcijsku konfiguraciju backenda i frontenda Dodane upute za održavanje aplikacije Ažuriran dijagram razreda	Bruno Ševčenko	23.1.2025.
1.7	Ažurirani alati i tehnologije Ažurirana tablica aktivnosti	Jelena Glasovac	23.1.2025.
1.8	Dodan dijagram stanja i pripadajući opis	Jelena Glasovac	24.1.2025.
2.0	Dodani dijagram "Contributors" sa GitHuba, osvježena tablica doprinosa	Jelena Glasovac	24.1.2025.

B. Popis literature

1. *Vlada prihvatila prijedlog Zakona o upravljanju i održavanju zgrada*, MPGI RH, https://mpgi.gov.hr/vijesti-8/vlada-prihvatila-prijedlog-zakona-o-upravljanju-i-odrzavanju-zgrada/17817
2. Programsko inženjerstvo, FER ZEMRIS, http://www.fer.hr/predmet/proinz
3. I. Sommerville, ”Software engineering”, 8th ed, Addison Wesley, 2007.
4. T.C.Lethbridge, R.Langaniere, ”Object-Oriented Software Engineering”, 2nd e. McGraw-Hill, 2005.
5. I. Marsic, Software engineering book“, Department of Electrical and Computer Engineering, Rutgers University, http://www.ece.rutgers.edu/˜marsic/books/SE
6. The Unified Modeling Language, https://www.uml-diagrams.org/
7. Astah Community, http://astah.net/editions/uml-new

C. Prikaz aktivnosti grupe

Dnevnik sastajanja

Kontinuirano osvježavanje

U ovom dijelu potrebno je redovito osvježavati dnevnik sastajanja prema predlošku.

1. sastanak

- Datum: 16. listopada 2024.
- Prisustvovali: Jelena Glasovac, Mauro Mohorovičić, Lovre Rančev, Bruno Ševčenko, Bruno Tudor, Marko Vukadin
- Teme sastanka:
    - upoznavanje i formiranje tima, izbor vođe
    - dogovorene metode komunikacije (WhatsApp grupa, GitHub projekt)
    - prema željama članova tima, podjeljene su uloge u timu:

Član tima	Uloga
Mauro Mohorovičić	backend
Marko Vukadin	backend
Lovre Rančev	frontend
Bruno Tudor	frontend
Jelena Glasovac	vođenje, dizajn baze podataka, pisanje dokumentacije
Bruno Ševčenko	pisanje dokumentacije

2. sastanak

- Datum: 23. listopada 2024.
- Prisustvovali: Jelena Glasovac, Mauro Mohorovičić, Bruno Ševčenko, Bruno Tudor, Marko Vukadin
- Teme sastanka:
    - opis i definiranje problema, izoliranje zahtjeva iz teksta, upoznavanje s zadatkom
    - potvrda podjele posla
    - podizanje GitHuba i dodavanje svih kao kolaboratora

3. sastanak

- Datum: 30. listopada 2024.
- Prisustvovali: Jelena Glasovac, Mauro Mohorovičić, Bruno Ševčenko
- Teme sastanka:
    - dosadašnji napredak s dokumentacijom i arhitekturom
    - voditeljica tima podjelila zaduženja za sljedećih 1/2 tjedna članovima tima (nije bilo zamjerki)

4. sastanak

- Datum: 6. studenog 2024.
- Prisustvovali: Jelena Glasovac, Mauro Mohorovičić, Bruno Ševčenko
- Teme sastanka:
    - analiza napretka
        - OAuth uključen i funkcionalan,
        - izabrana funkcionalnost koja će biti napravljena uz login - slanje poruka u bazu i njihovo čitanje na stranicu, login stranica pronađena među open source resursima i iskorištena
        - druga stranica dokumentacije napisana, preostaje treća stranica
    - članovi tima podsjećeni da je 1. revizija u tjednu od 11. do 15. 11. 2024.

5. sastanak

- Datum: 15. siječnja 2025.
- Prisustvovali: Jelena Glasovac, Mauro Mohorovičić, Bruno Ševčenko, Bruno Tudor, Marko Vukadin
- Teme sastanka:
    - analiza napretka
        - demonstracija backenda, izbor frontenda i dogovori oko daljnih unaprjeđenja
        - rasprava i priprema za crtanje dijagrama i preostale dokumentacije
        - planiranje rješenja problema pronađenih tjekom implementacije..

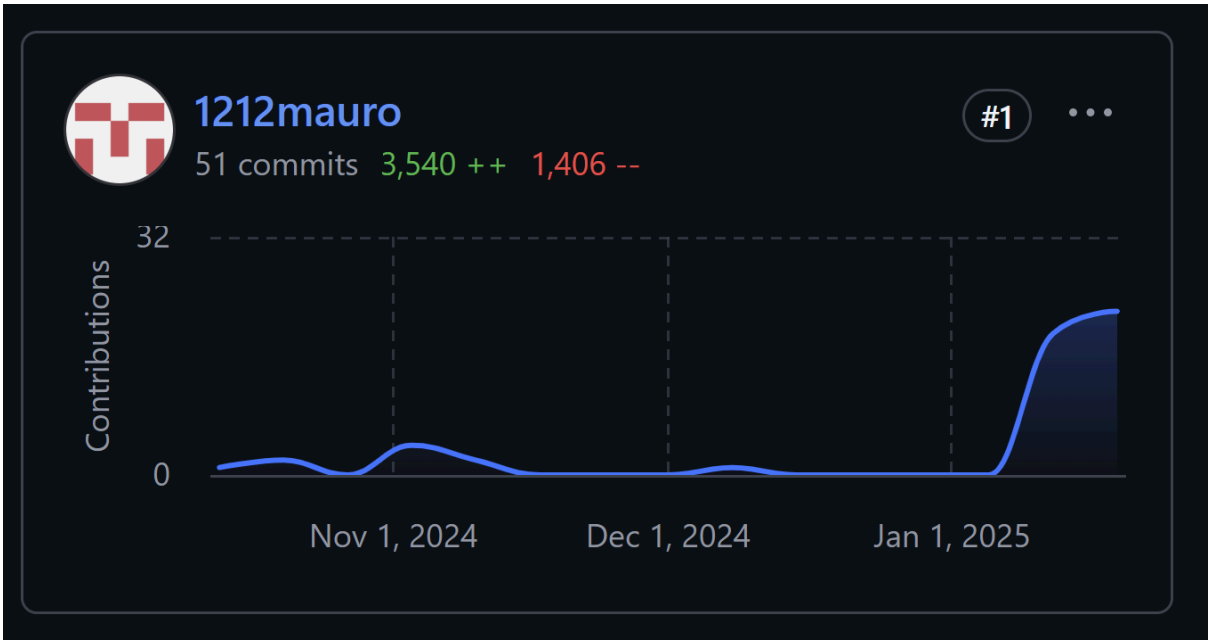
Tablica aktivnosti

Aktivnost	Jelena Glasovac	Mauro Mohorovičić	Lovre Rančev	Bruno Ševčenko	Bruno Tudor	Marko Vukadin
Upravljanje projektom	13	5	3	6	3	3
Opis projektnog zadatka	1	0	0	4	0	0
Funkcionalni zahtjevi	1	0	0	5	0	0
Opis pojedinih obrazaca	0	0	0	4	0	0
Dijagram obrazaca	2	0	0	0	0	0
Sekvencijski dijagrami	1	0	0	0	0	0
Opis ostalih zahtjeva	0	0	0	1	0	0
Arhitektura i dizajn sustava	2	1	0	2	0	0
Baza podataka	5	0	0	0	0	1
Dijagram razreda	3	0	0	2	0	0
Dijagram stanja	1	0	0	0	0	0
Dijagram aktivnosti	2	0	0	0	0	0
Dijagram komponenti	1.5	0	0	0	0	0
Korištene tehnologije i alati	3.5	0	0	0	0	0

Ispitivanje programskog rješenja	0	0	0	0	0	6
Dijagram razmještaja	1.5	0	0	0	0	0
Upute za puštanje u pogon	0	0	0	5	0	0
Dnevnik sastajanja	1.5	0	0	0	0	0
Zaključak i budući rad	2	0	0	1	0	0
Popis literature	0	0	0	1	0	0
Izrada početne stranice	0	0	10	0	0	0
Dizajn web stranica	1	0	1	0	0	0
Izrada baze podataka	0	1	0	0	0	0
Frontend (ostalo)	0	35	3	0	15	5
Spajanje s bazom podataka	0	1	0	0	0	0
Backend	0	105	0	0	0	17
Deploy	0	0	0	0	0	14
Izrada prezentacije projekta i uređivanje finalne verzije dokumentacije	3	0	0	0	0	0
TOTAL	**45**	**144**	**17**	**31**	**18**	**41**

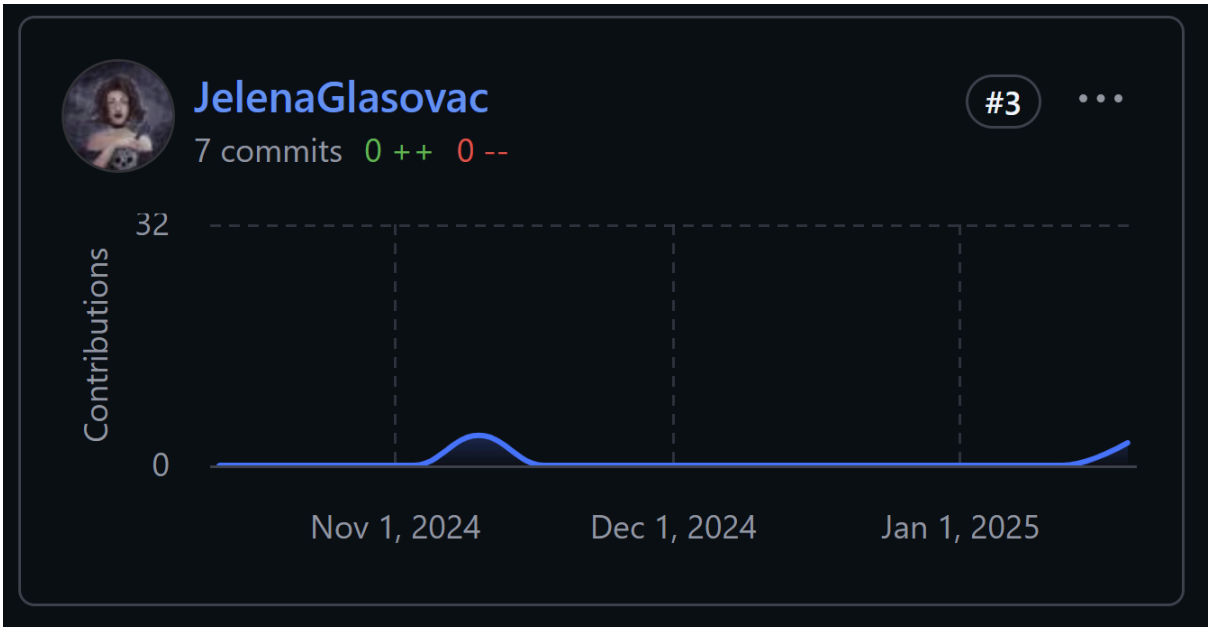
Dijagram pregleda promjena

- Mauro Mohorovičić

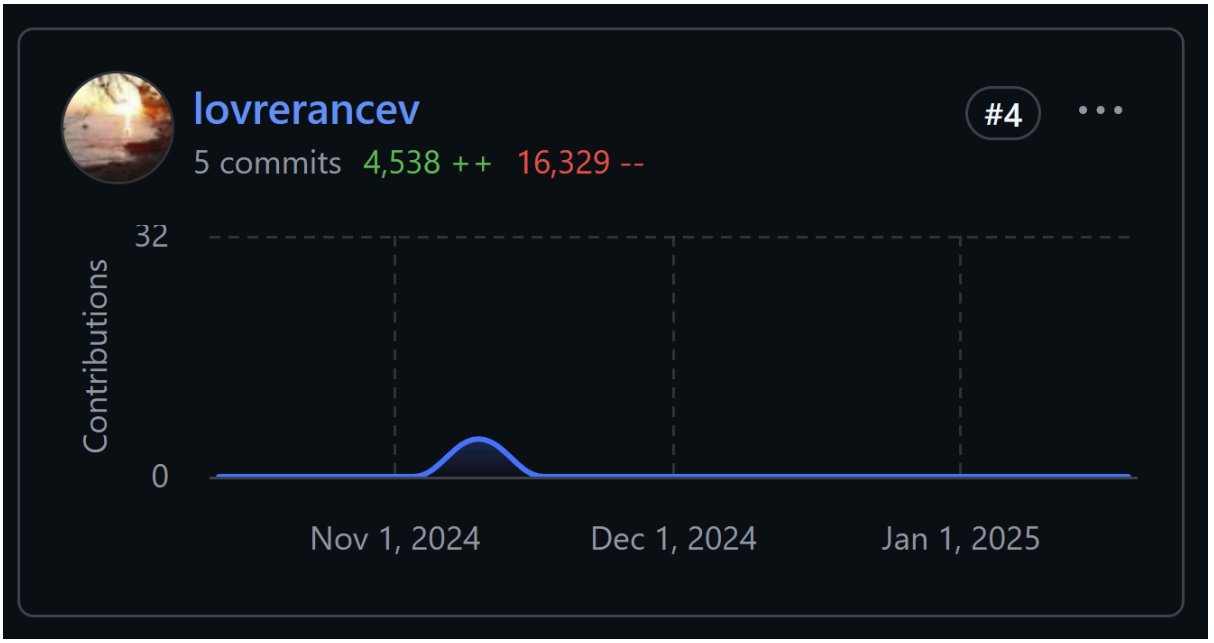


- Marko Vukadin

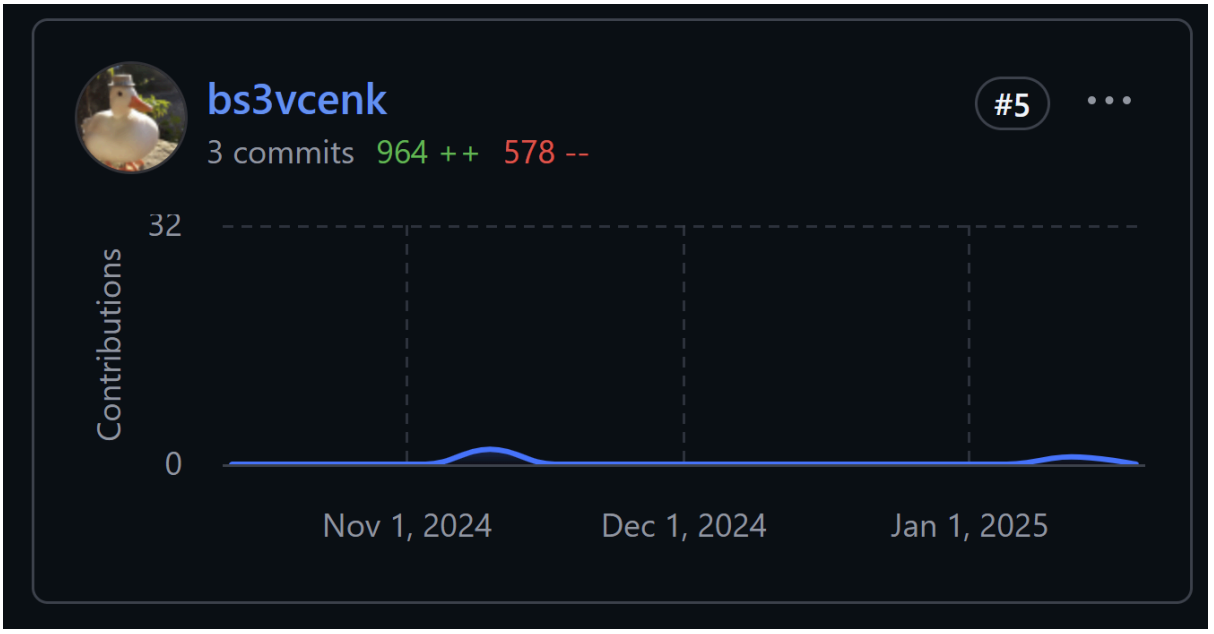
- Jelena Glasovac



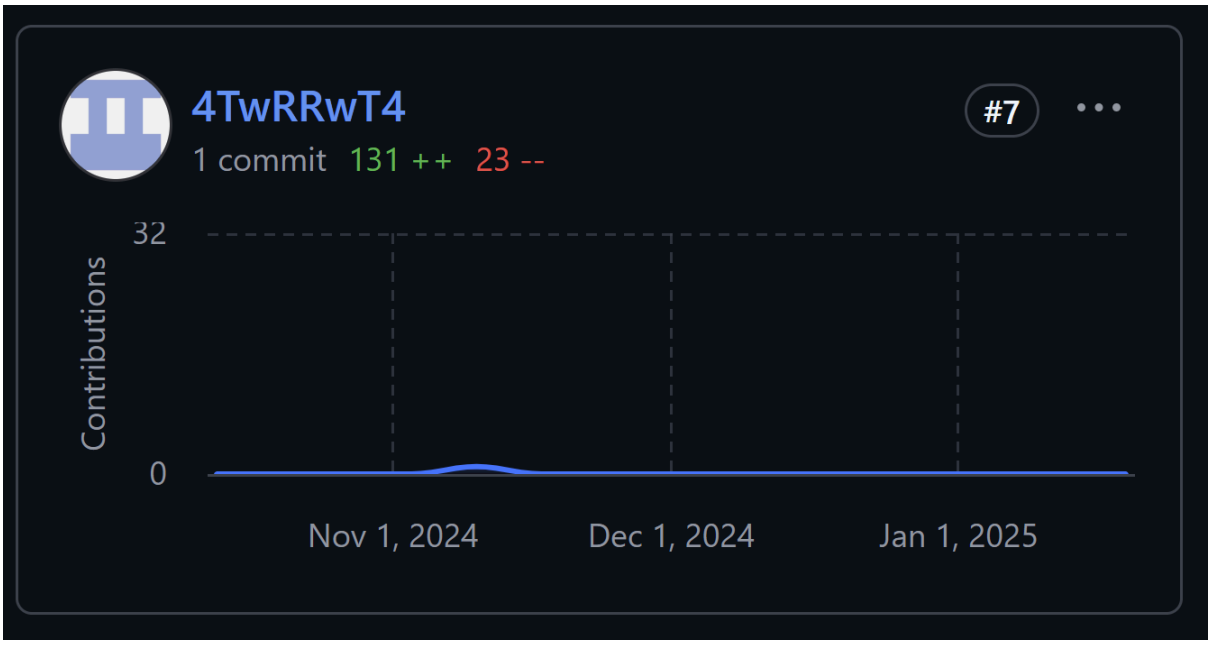
- Lovre Rančev



- Bruno Ševčenko



- Bruno Tudor



Ključni izazovi i rješenja

Dva su ključna izazova s kojima smo se morali suočiti prilikom izrade projekta.

Prvi izazov je manjak inicijative pri radu nekih članova tima i manjak suradnje. Zbog toga, komponente je teže povezivati i više se vremena gubi na popravljanje grešaka i

prilagođavanje. Manji broj članova u timu i veliki nerazmjer uloženog truda među članovima tima stvara velike izazove koje nismo adekvatno riješili dovoljno brzo.

Drugi izazov su kašnjenja izrada funkcionalnosti, zbog kojih je teže poštovati vremenske rokove i veliki je napor nadoknaditi izgubljeno vrijeme netom prije predaje. Tome se ne može doskočiti naknadnim rješenjem, osim ulaganja velike količine vremena pred kraj projekta.

Ipak, iz ovoga imali smo priliku naučiti lekcije o važnosti komunikacije unutar tima i redovitih sastanaka koje ćemo primijeniti na budućim projektima.