Programsko inženjerstvo ak.god 25/26

Sveučilište u Zagrebu

Fakultet elektrotehnike i računarstva

Naziv projekta: StanBlog

Tim: <TG10.2>

Ime tima: stanari123

Nastavnik: Vlado Sruk

Opis projektnog zadatka

Uvod

U višestambenim zgradama živi velik broj ljudi koji dijele zajedničke dijelove zgrade, poput stubišta, hodnika, dizala, parkirališta i vanjskih površina. Iako svaki stanar koristi svoj privatni prostor, zajednički prostori zahtijevaju stalno i organizirano održavanje. Financijski model za to postoji – pričuva koju plaćaju svi suvlasnici – no mnoge zgrade i dalje ostaju zapuštene, što se jasno vidjelo nakon potresa u Zagrebu.

Glavni uzrok problema nije nedostatak sredstava, nego slaba i neučinkovita komunikacija među suvlasnicima. Sastanci stanara često su slabo posjećeni, informacije se prenose sporim i nepouzdanim kanalima, a mnogi suvlasnici nemaju uvid u probleme koji se događaju izvan njihovog kata ili ulaza. Zbog toga brojni kvarovi i potrebne intervencije ostaju neprimijećeni ili neraspravljeni.

Ovaj projekt bavi se upravo tom problematikom – stvaranjem centralizirane, transparentne i suvremene platforme koja olakšava prijavu problema, komunikaciju suvlasnika i donošenje zajedničkih odluka.

Cilj projektnog zadatka

Cilj projekta je razviti web aplikaciju koja:

- služi kao digitalna oglasna ploča za sve suvlasnike,
- omogućuje pokretanje i vođenje diskusija o problemima u zgradi,
- omogućuje glasanje o prijedlozima,
- automatski generira prijedlog za sastanak kada više od 25 % suvlasnika glasa pozitivno,
- omogućuje integraciju s vanjskim sustavom StanPlan radi formalnog vođenja sastanaka,
- administraciji omogućuje upravljanje korisnicima i postavkama integracije.

Korištenjem ove aplikacije komunikacija postaje brža, jednostavnija i preglednija, što dovodi do kvalitetnijeg održavanja zgrade i učinkovitijeg donošenja odluka.

Opis problematike

Suvlasnici često nemaju uvid u stanje cijele zgrade i oslanjaju se na usmene informacije, oglasne ploče ili rad predstavnika suvlasnika. Ovakav pristup uzrokuje:

- nedovoljno prijavljivanje problema,
- nisku informiranost suvlasnika,
- spor proces donošenja odluka,
- nedostatak transparentnosti,
- lošu koordinaciju među stanarima i predstavnicima.

Dodatni problem nastaje zbog neredovitog održavanja sastanaka, niskog odaziva i nepostojanja digitalnog traga o raspravama i odlukama.

Sve navedeno rezultira lošim održavanjem zajedničkih prostora, sporim rješavanjem intervencija i stvaranjem nezadovoljstva među suvlasnicima.

Opis predloženog rješenja

Predloženo rješenje je web aplikacija koja omogućuje suvlasnicima:

Diskusije

- pokretanje rasprave o bilo kojem problemu ili prijedlogu,
- otvorene i privatne diskusije,
- određivanje ograničenja (broj poruka, zabrana sudjelovanja),
- sudjelovanje predstavnika suvlasnika u javnim diskusijama,
- vidljivost postojanja privatne diskusije svim suvlasnicima uz skrivanje sadržaja.

Glasanje

- pokretanje glasanja unutar diskusije,
- glasanje “Za” ili “Protiv”,
- automatski poziv za sastanak ako više od 25 % suvlasnika glasa pozitivno.

Sastanci

- generiranje prijedloga sastanka s naslovom, dnevnim redom i ciljem,
- povezivanje s diskusijom u kojoj je glasanje pokrenuto,
- pravno valjani zaključci sastanka kroz integraciju sa StanPlan aplikacijom,
- slanje e-mail obavijesti svim suvlasnicima.

Administracija

- izrada i upravljanje korisnicima (suvlasnici i predstavnik suvlasnika),
- izmjena korisničkih podataka,
- postavljanje StanPlan integracijskog URL-a,
- nadzor sigurnosnih aspekata aplikacije.

Autentifikacija

- prijava putem OAuth 2.0 servisa,
- sigurno rukovanje korisničkim identitetima bez lokalnog spremanja lozinki.

Potencijalna korist projekta

Implementacija aplikacije donosi višestruke koristi:

Suvlasnicima:

- veća informiranost o stanju zgrade,
- mogućnost sudjelovanja u raspravama bez odlaska na fizički sastanak,
- transparentnost u odlučivanju,
- brže rješavanje problema.

Predstavniku suvlasnika:

- manji administrativni teret,
- bolja evidencija svih prijedloga i odluka,
- jednostavnija organizacija sastanaka,
- povećano povjerenje zajednice.

Zgradi i zajednici:

- bolje održavanje zajedničkih prostora,
- racionalnije korištenje pričuve,
- brža reakcija u hitnim slučajevima,
- veći stupanj organiziranosti i suradnje.

Postojeća slična rješenja na tržištu

Trenutno postoje aplikacije koje pokrivaju dio funkcionalnosti potrebnih za upravljanje zgradama, poput:

Susjed42

- naglasak na osnovnoj međususjedskoj komunikaciji,
- ne podržava strukturirane diskusije niti automatizirano sazivanje sastanaka.

mSuvlasnik

- fokusiran na financijsko poslovanje i komunikaciju s upraviteljem,
- komunikacija među suvlasnicima nije u središtu.

E-upravitelj

- alat namijenjen profesionalnim upraviteljima,
- nudi dokumentaciju i praćenje radova, ali nema demokratičnu oglasnu ploču.

Ključna razlika našeg rješenja:
Demokratizira komunikaciju dopuštajući svakom suvlasniku pokretanje rasprava i iznošenje prijedloga, uz automatizirane mehanizme glasanja i sazivanja sastanka.

Skup korisnika

Aplikacija je namijenjena:

- suvlasnicima stanova u višestambenim zgradama,
- predstavnicima suvlasnika,
- upraviteljima zgrada,
- velikim stambenim kompleksima,
- potencijalno i poslovnim zgradama koje imaju zajedničke prostore.

Korisnici su tehnički raznoliki – od starijih osoba do digitalno aktivnih suvlasnika – pa je aplikacija dizajnirana da bude jednostavna, intuitivna i pristupačna.

Opseg projektnog zadatka

Ovaj projekt obuhvaća:

- razvoj serverske logike,
- izradu baze podataka i poslovnih pravila,
- mogućnost pokretanja diskusija i glasanja,
- automatizirano generiranje sastanaka nakon glasanja,
- integraciju s OAuth 2.0 sustavom,
- administracijske funkcije,
- integraciju sa StanPlan sustavom.

Ovaj projekt **ne** obuhvaća:

- rješavanje financijskog dijela upravljanja zgradom,
- vođenje tehničke dokumentacije zgrade,
- komercijalne funkcionalnosti upravitelja nekretninama.

Mogućnosti prilagodbe i nadogradnje

Arhitektura sustava osmišljena je tako da omogućuje jednostavnu prilagodbu i buduće proširenje funkcionalnosti. Modularna podjela frontend i backend komponenti, korištenje REST API-ja te integracija s vanjskim servisima omogućuju dodavanje novih mogućnosti bez potrebe za značajnim izmjenama postojećeg koda. Sustav se može nadograditi dodatnim modulima kao što su naprednije opcije glasanja, prošireni administracijski alati, sustav obavijesti ili dodatne integracije s platformama za upravljanje zgradama. Ovakav pristup osigurava dugoročnu skalabilnost, održivost i prilagodljivost rješenja potrebama korisnika.

Zaključno

Ovaj projekt predstavlja moderan odgovor na stvarne probleme komunikacije u višestambenim zgradama. Digitalizacijom oglasne ploče, omogućavanjem strukturiranih diskusija i uvođenjem transparentnog glasanja, aplikacija potiče angažiranost suvlasnika i ubrzava donošenje odluka. Integracija s postojećim StanPlan sustavom proširuje funkcionalnost na formalne procese donošenja odluka, čime aplikacija postaje snažan alat za suvremeno upravljanje zgradama.

Rješenje je skalabilno, prilagodljivo i lako nadogradivo te predstavlja temelj za budući razvoj naprednijih funkcionalnosti.

Analiza zahtjeva

Na ovoj se stranici nalazi detaljna analiza zahtjeva za aplikaciju za upravljanje komunikacijom i održavanjem višestambenih zgrada.
Cilj je definirati funkcionalne i nefunkcionalne zahtjeve, dionike sustava te zahtjeve vezane uz održavanje i sigurnost.
Nefunkcionalni zahtjevi i zahtjevi domene primjene dopunjuju funkcionalne zahtjeve. Oni opisuju kako se sustav treba ponašati i koja ograničenja treba poštivati (performanse, korisničko iskustvo, pouzdanost, standardi kvalitete, sigurnost itd.).

Analiza zahtjeva definira sve funkcionalne, nefunkcionalne i održavne aspekte sustava te identificira ključne dionike.
Tablični prikaz zahtjeva omogućuje transparentnost, lakšu provjeru i osigurava da svi zahtjevi projekta budu precizno dokumentirani i pratljivi tijekom razvoja.
Dodatno, opisani nefunkcionalni zahtjevi i zahtjevi domene primjene osiguravaju kvalitetu, sigurnost, pouzdanost i jednostavnu nadogradnju sustava.

2. Funkcionalni zahtjevi

ID zahtjeva	Opis	Prioritet	Kriteriji prihvaćanja
F-001	Sustav omogućuje administratoru stvaranje korisničkih računa - predstavnik suvlasnika i suvlasnici.	Visok	Administrator može stvoriti korisničko ime, lozinku i email adresu, a stvoreni korisnik može pristupiti sustavu.
F-002	Sustav omogućuje korisnicima promjenu lozinke pomoću korisničkog imena i stare lozinke.	Visok	Korisnik može promijeniti lozinku i pristupiti sustavu.
F-003	Sustav podržava autentifikaciju putem vanjskog servisa OAuth 2.0.	Srednji	Korisnik se može prijaviti u sustav pomoću servisa za autentifikaciju.
F-004	Sustav omogućuje korisnicima prijavu putem korisničkog imena i lozinke.	Visok	Korisnik se može prijaviti u sustav pomoću svoje lozinke i korisničkog imena.
F-005	Sustav omogućuje suvlasnicima pokretanje javne i privatne diskusije na oglasnoj ploči.	Visok	Korisnik može kreirati temu, postaviti naslov, opis i odrediti parametre diskusije.
F-006	Inicijator diskusije može ograničiti broj poruka po korisniku ili zabraniti sudjelovanje određenih suvlasnika.	Visok	Inicijator može postaviti parametre diskusije.
F-007	Sustav omogućuje pokretanje privatne diskusije vidljive samo odabranim sudionicima.	Visok	Samo korisnici na listi mogu čitati i komentirati privatnu diskusiju.

ID zahtjeva	Opis	Prioritet	Kriteriji prihvaćanja
F-008	Sustav automatski šalje email poruku suvlasnicima koji su dodani u privatnu diskusiju.	Srednji	Korisnik koji je dodan u diskusiju prima email obavijest.
F-009	Sustav omogućuje pokretanje glasanja unutar diskusije.	Visok	Inicijator može pokrenuti glasanje s pozitivnim i negativnim odgovorom. Glasovi se broje i prikaz se mijenja u stvarnom vremenu.
F-010	Sustav omogućuje prikaz trenutnog broja pozitivnih i negativnih glasova.	Nizak	Nakon svakog glasa, broj glasova se ažurira na sučelju.
F-011	Ako broj pozitivnih glasova premašuje 25%, može se sazvati sastanak suvlasnika.	Nizak	Sustav omogućuje kreiranje zahtjeva za sastanak ako ima više od 25% pozitivnih glasova.
F-012	Sustav koristi vanjsku aplikaciju StanPlan radi kreiranja sastanaka.	Visok	Sustav šalje podatke (naslov, termin, dnevni red, ciljeve) putem sučelja StanPlan.
F-013	Administrator unosi adresu servera StanPlan aplikacije radi integracije.	Srednji	Administrator može upisati adresu servera StanPlan.
F-014	Sustav omogućuje preuzimanje liste diskusija s pozitivnim rezultatom glasanja.	Nizak	Sustav šalje listu s naslovom, pitanjem i poveznicom na diskusiju.
F-015	Sustav omogućuje pregled svih diskusija.	Visok	Korisnik može vidjeti sve diskusije na oglasnoj ploči.
F-016	Sustav vodi evidenciju broja poruka po sudioniku u diskusiji.	Srednji	Broj poruka po sudioniku se automatski bilježi i prikazuje.
F-017	Sustav omogućuje upload datoteka uz poruke (slike, dokumenti).	Srednji	Datoteka se uspješno sprema i povezuje s odgovarajućom porukom, vidljiva sudionicima diskusije.
F-018	Administrator može dodavati i uređivati uloge korisnika.	Srednji	Administrator može mijenjati uloge i ovlasti korisnika.

3. Nefunkcionalni zahtjevi i zahtjevi domene primjene

ID zahtjeva	Opis	Prioritet	Kriteriji prihvaćanja
N-001	Sustav mora biti dostupan 24/7 uz minimalne zastoje.	Visok	Sustav je dostupan većinu vremena, downtime < 2h mjesečno.

ID zahtjeva	Opis	Prioritet	Kriteriji prihvaćanja
N-002	Aplikacija mora biti responzivna i raditi na mobilnim i desktop uređajima.	Visok	Sučelje se pravilno prikazuje na svim tipičnim rezolucijama.
N-003	Sustav mora koristiti siguran protokol (HTTPS) i enkripciju korisničkih podataka.	Visok	Komunikacija je uvijek enkriptirana, lozinke i tokeni su sigurni.
N-004	Sustav podržava više jezika korisničkog sučelja (minimalno hrvatski i engleski).	Srednji	Korisnik može odabrati jezik sučelja, tekstovi su prikazani ispravno.
N-005	Sustav mora imati vrijeme odziva do 2 sekunde za tipične operacije (pregled diskusija, glasanje, učitavanje stranice).	Visok	Sučelje se učitava i operacije izvršavaju brzo bez primjetnog kašnjenja.
N-006	Sustav mora biti kompatibilan s najčešćim web preglednicima i mobilnim platformama (Chrome, Firefox, Edge, iOS, Android).	Visok	Aplikacija se ispravno prikazuje i funkcionira na navedenim platformama.
N-007	Sustav mora imati audit log za praćenje aktivnosti korisnika.	Srednji	Sve aktivnosti korisnika su zapisane u logu za reviziju.
N-008	Sustav mora podržavati ograničen broj korisnika primjeren opsegu studentskog projekta.	Nizak	Sustav može posluživati tipičan broj korisnika u testnom okruženju (do 50-100 korisnika).
N-009	Sustav mora poštivati standarde kvalitete i sigurnosne protokole te biti skalabilan za moguće buduće nadogradnje.	Srednji	Sustav je razvijen u skladu sa sigurnosnim standardima i modularnošću.

4. Zahtjevi za održavanjem

ID zahtjeva	Opis	Prioritet	Kriteriji prihvaćanja
M-001	Sustav mora imati dokumentaciju za administratore i programere.	Visok	Dokumentacija je dostupna i ažurirana.
M-002	Sustav mora omogućiti jednostavno ažuriranje i nadogradnju baze podataka i aplikacije.	Visok	Novi update se može instalirati bez gubitka podataka ili funkcionalnosti.
M-003	Sustav mora imati procedure za backup i oporavak podataka.	Visok	Podaci se mogu obnoviti iz backupa u slučaju kvara.

ID zahtjeva	Opis	Prioritet	Kriteriji prihvaćanja
M-004	Sustav mora omogućiti praćenje grešaka i anomalija.	Srednji	Sustav generira logove i upozorenja za eventualne probleme.

5. Aktori

Aktori predstavljaju sve subjekte koji izravno komuniciraju sa sustavom i sudjeluju u obrascima upotrebe. Oni iniciraju radnje, primaju informacije ili razmjenjuju podatke sa sustavom.

- Suvlasnik
  Osnovni korisnik aplikacije. Pokreće i pregledava diskusije, sudjeluje u raspravama, glasa o prijedlozima i prima obavijesti.
- Predstavnik suvlasnika
  Napredni korisnik s dodatnim ovlastima. Koordinira održavanje, pregledava sve diskusije, inicira sastanke i upravlja procesima glasanja.
- Administrator
  Upravlja korisničkim računima, ulogama i integracijama. Odgovoran je za konfiguraciju sustava, sigurnosne postavke i održavanje aplikacije.
- StanPlan sustav
  Vanjski sustav za formalno vođenje sastanaka. Prima podatke o sastancima i vraća potvrde o kreiranju termina i dnevnog reda.
- Autentifikacijski servis (OAuth 2.0)
  Vanjski servis za provjeru identiteta korisnika. Omogućuje sigurnu prijavu bez lokalnog spremanja lozinki.

6. Dionici

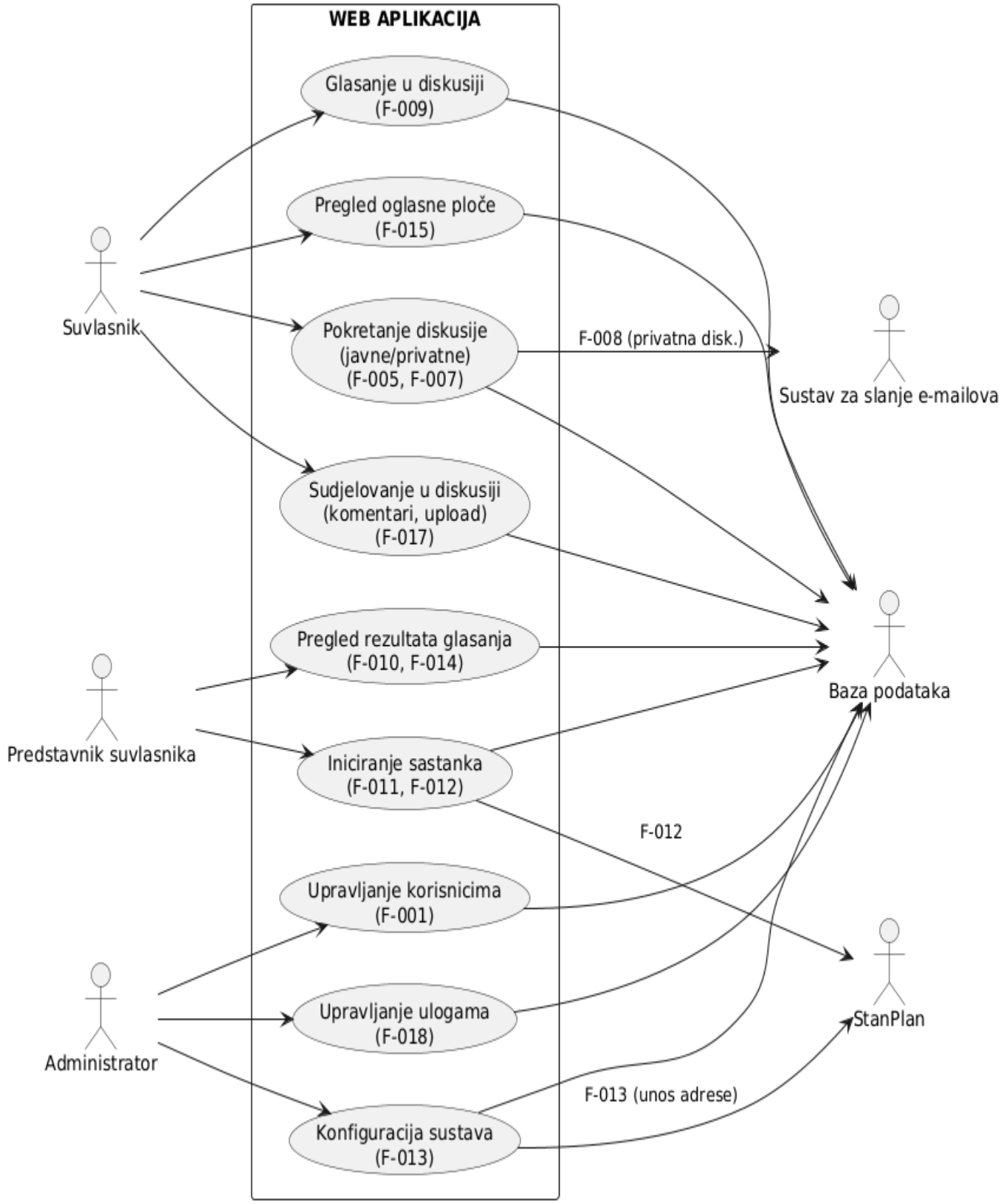
Dionik	Uloga / Interes
Suvlasnici	Glavni korisnici aplikacije; prijavljuju probleme, sudjeluju u diskusijama i glasanjima.
Predstavnici suvlasnika	Koordiniraju održavanje, pregledavaju diskusije i iniciraju sastanke.
Administrator	Upravljanje korisnicima, integracijama i održavanjem sustava.
Upravitelji zgrada	Koriste aplikaciju za pregled problema i planiranje održavanja (opcionalno).
Razvojni tim	Razvija i održava aplikaciju, implementira nove funkcionalnosti i ispravke bugova.

Obrasci uporabe

Dijagrami obrazaca uporabe

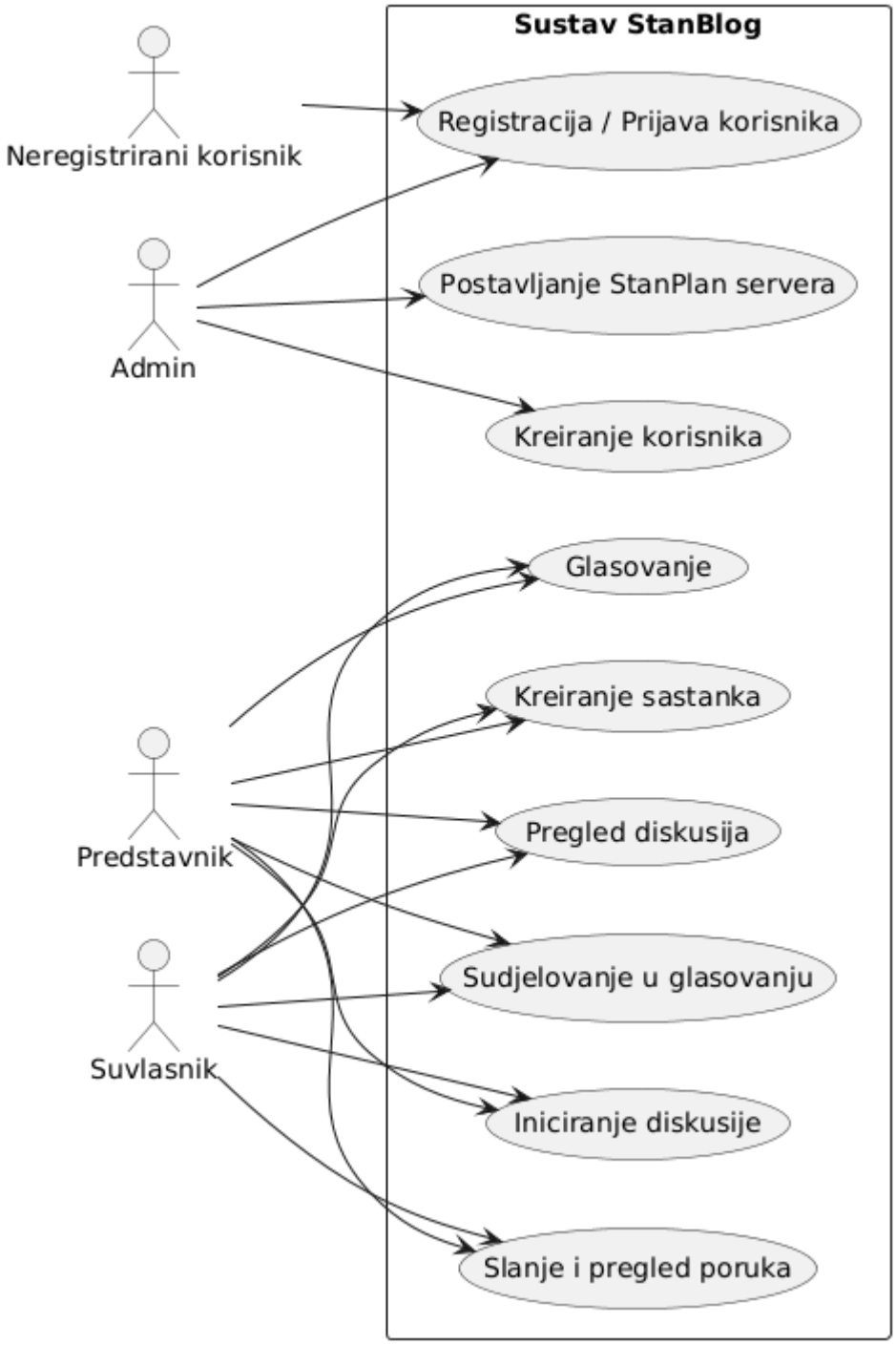
Prikazati odnos aktora i obrazaca uporabe odgovarajućim UML dijagramom. Modelirati po razinama apstrakcije i skupovima srodnih funkcionalnosti.

1. Visokorazinski dijagram obrazaca uporabe cijelog sustava



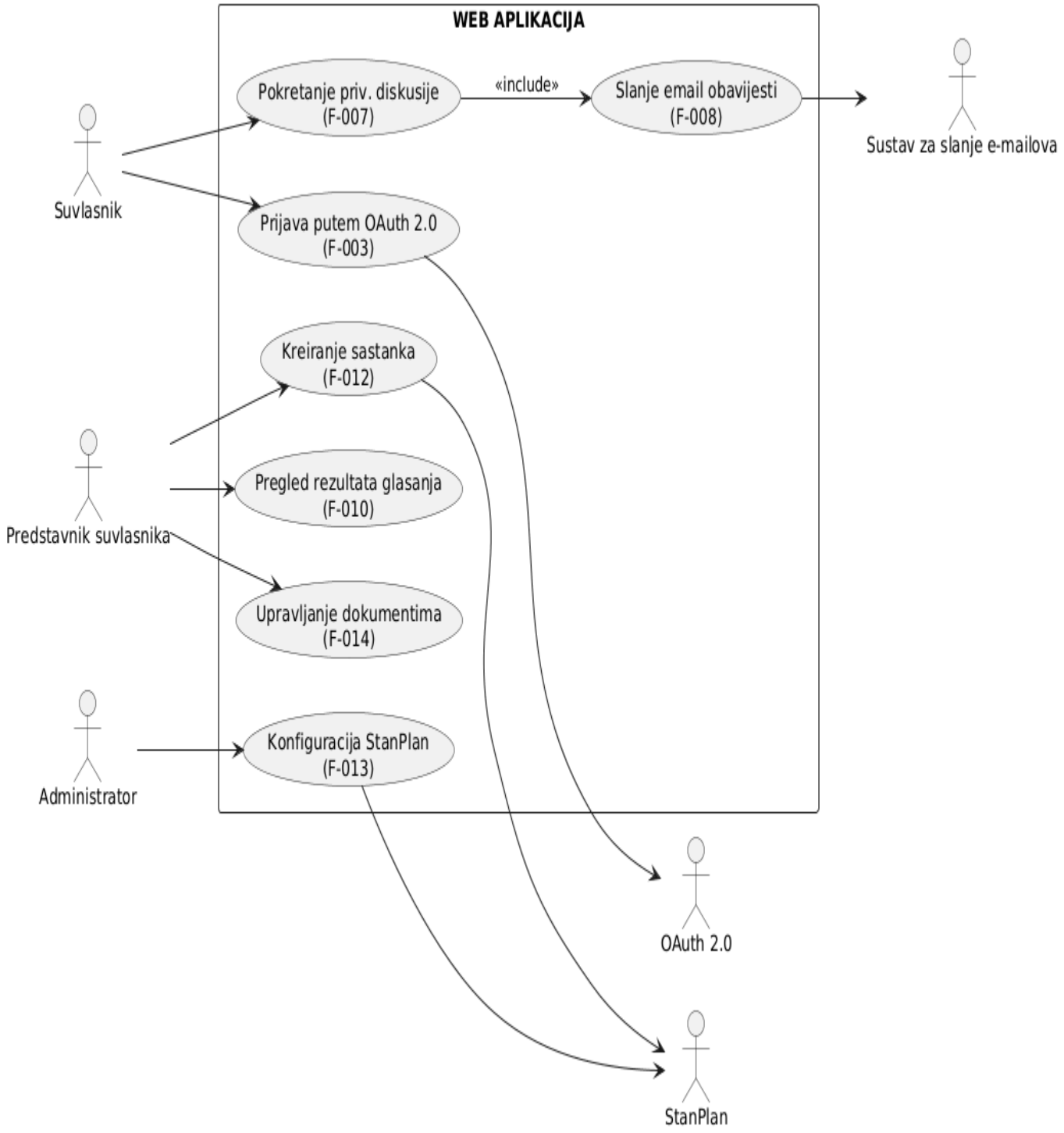
Dijagram prikazuje glavne funkcionalnosti web aplikacije i interakciju aktera s njima. Suvlasnici mogu pregledavati oglasnu ploču, pokretati i sudjelovati u diskusijama te glasati u njima. Predstavnici suvlasnika iniciraju sastanke i pregledavaju rezultate glasanja, dok administratori upravljaju korisnicima, ulogama i konfiguracijom sustava. Sustav je povezan s vanjskim servisima poput StanPlana i sustava za slanje e-mailova, a svi use case-ovi koriste bazu podataka za pohranu i dohvat informacija. Dijagram jasno prikazuje sve ključne funkcionalnosti i njihove međusobne veze.

2. dijagram obrazaca uporabe za korisničke uloge



Dijagram obrazaca uporabe za korisničke uloge – StanBlog sustav prikazuje glavne funkcionalnosti i interakciju različitih korisničkih uloga s aplikacijom. Administratori upravljaju korisnicima i konfiguracijom sustava, Predstavnici suvlasnika kreiraju sastanke i koordiniraju aktivnosti, dok Suvlasnici pregledavaju i pokreću diskusije te sudjeluju u glasovanju. Neregistrirani korisnici imaju ograničen pristup i mogu se registrirati. Dijagram jasno definira uloge, njihove funkcionalnosti i međusobne veze unutar sustava.

3. dijagram obrazaca uporabe za kritične sustave i integracije



Dijagram prikazuje glavne funkcionalnosti web aplikacije i interakciju aktera s ključnim vanjskim sustavima. Suvlasnici se prijavljuju putem OAuth 2.0 i mogu pokretati privatne diskusije, pri čemu se automatski šalju e-mail obavijesti. Predstavnici suvlasnika kreiraju sastanke, pregledavaju rezultate glasanja i upravljaju dokumentima. Administratori konfiguriraju integraciju sa StanPlan sustavom. Dijagram jasno pokazuje tko inicira akcije, kako se funkcionalnosti povezuju unutar sustava i kako aplikacija komunicira s vanjskim servisima, naglašavajući kritične integracije.

Opis obrazaca uporabe

UC01 – Registracija/Prijava

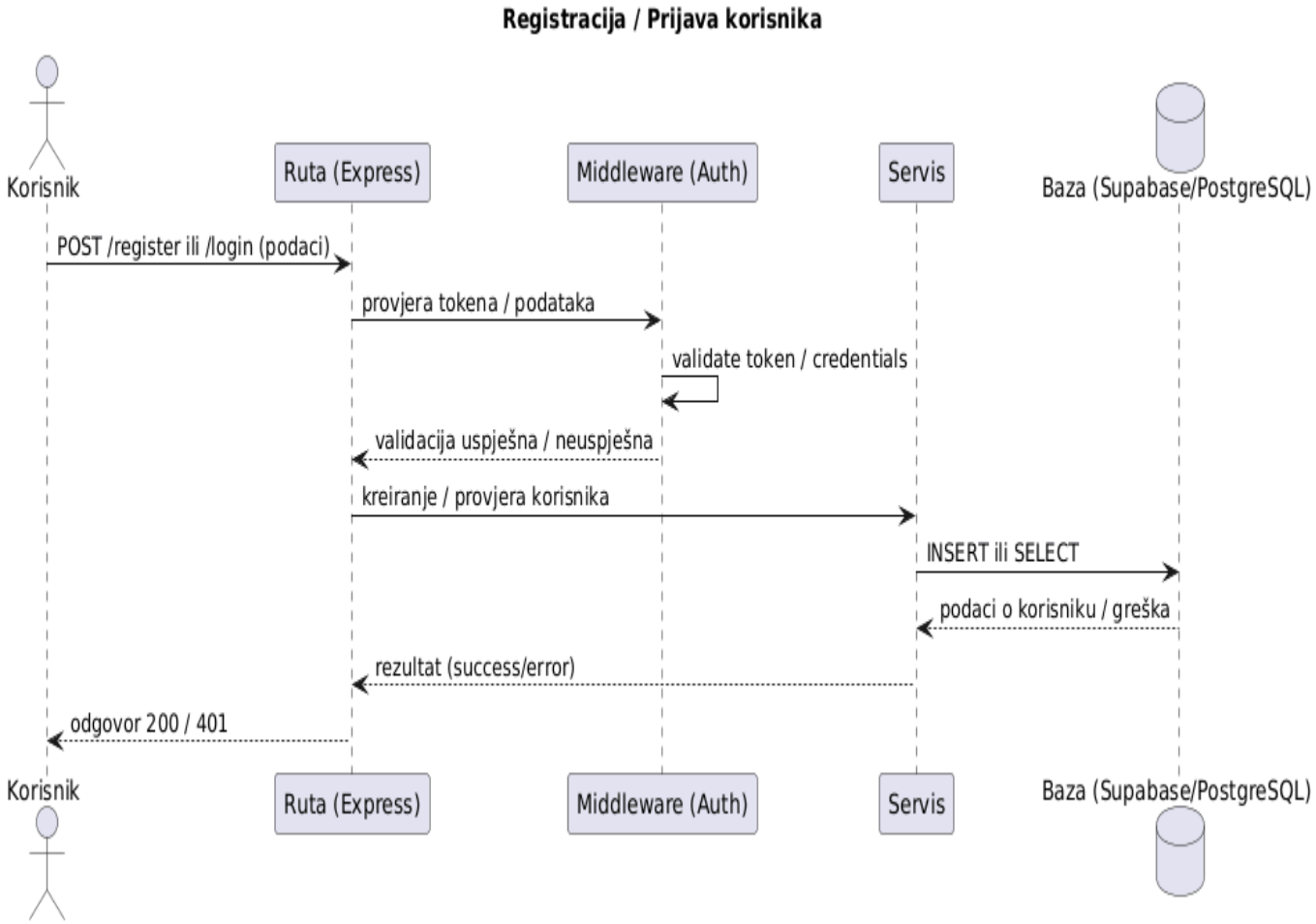
Glavni sudionik: Korisnik
Cilj: Prijaviti se u sustav
Sudionici: Korisnik, Sustav, Autentifikacijski servis
Preduvjet: Korisnik ima kreiran račun

Osnovni tijek

1. Korisnik unosi korisničko ime i lozinku.
2. Sustav provjerava vjerodajnice.
3. Sustav generira i vraća token o uspješnoj prijavi.
4. Korisnik se preusmjerava na početnu stranicu.

Odstupanja

- **1a.** Pogrešna lozinka → Poruka o grešci.
- **2a.** Korisnik ne postoji → Prikaz upozorenja.
- **3a.** OAuth servis nedostupan → Sustav nudi lokalnu prijavu.



Ovaj obrazac upotrebe opisuje procese prijave i registracije korisnika u sustav. Korisnik putem sučelja unosi vjerodajnice ili podatke potrebne za otvaranje računa, nakon čega se zahtjev šalje odgovarajućoj backend ruti. Servis provjerava postojeće podatke u bazi ili ih prosljeđuje vanjskom idetentifikacijskom servisu (OAuth 2.0). U slučaju prijave, sustav validira vjerodajnice i generira autentifikacijski token. U slučaju registracije, sustav kreira novi korisnički račun, sprema podatke u bazu i omogućuje korisniku prijavu. Nakon uspješne autentifikacije korisnik se preusmjerava na početnu stranicu. Use case pokriva i izvanredne situacije poput

pogrešne lozinke, nepostojećeg korisnika, pokušaja registracije već postojećeg računa ili nedostupnosti OAuth servisa, pri čemu sustav prikazuje odgovarajuće poruke.

UC02 – Kreiranje korisnika (Admin)

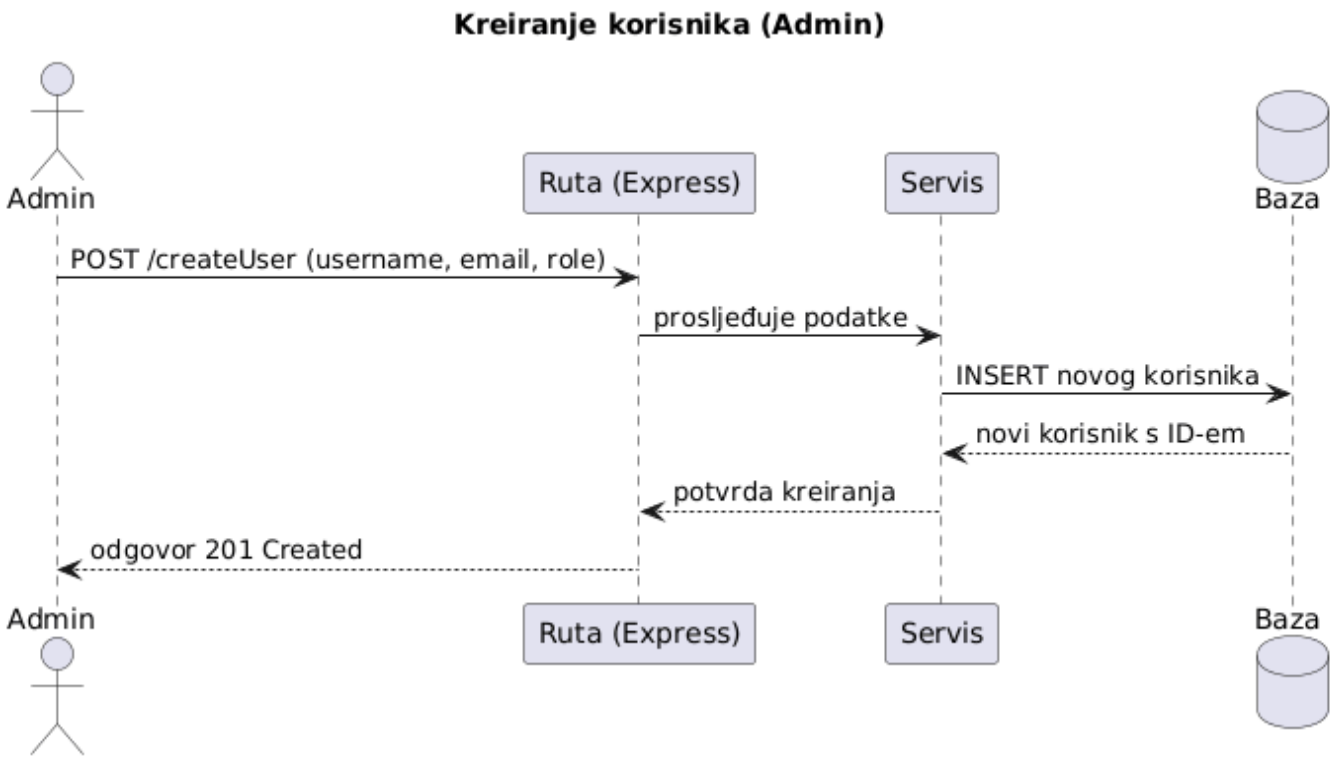
Glavni sudionik: Administrator
Cilj: Stvoriti račun suvlasnika ili predstavnika suvlasnika
Sudionici: Administrator, Sustav
Preduvjet: Administrator je prijavljen u sustav

Osnovni tijek

1. Administrator odabire opciju „Dodaj korisnika".
2. Sustav prikazuje formu za unos podataka.
3. Administrator unosi korisničko ime, email i ulogu.
4. Sustav provjerava ispravnost podataka.
5. Sustav upisuje podatke u bazu i kreira korisnika.

Odstupanja

- **1a.** Podaci nisu ispravni → Sustav javlja grešku i vraća na unos.
- **3a.** Email već postoji → Sustav javlja poruku i traži drugi email.



UC03 – Iniciranje diskusije

Glavni sudionik: Suvlasnik
Cilj: Kreirati javnu ili privatnu diskusiju

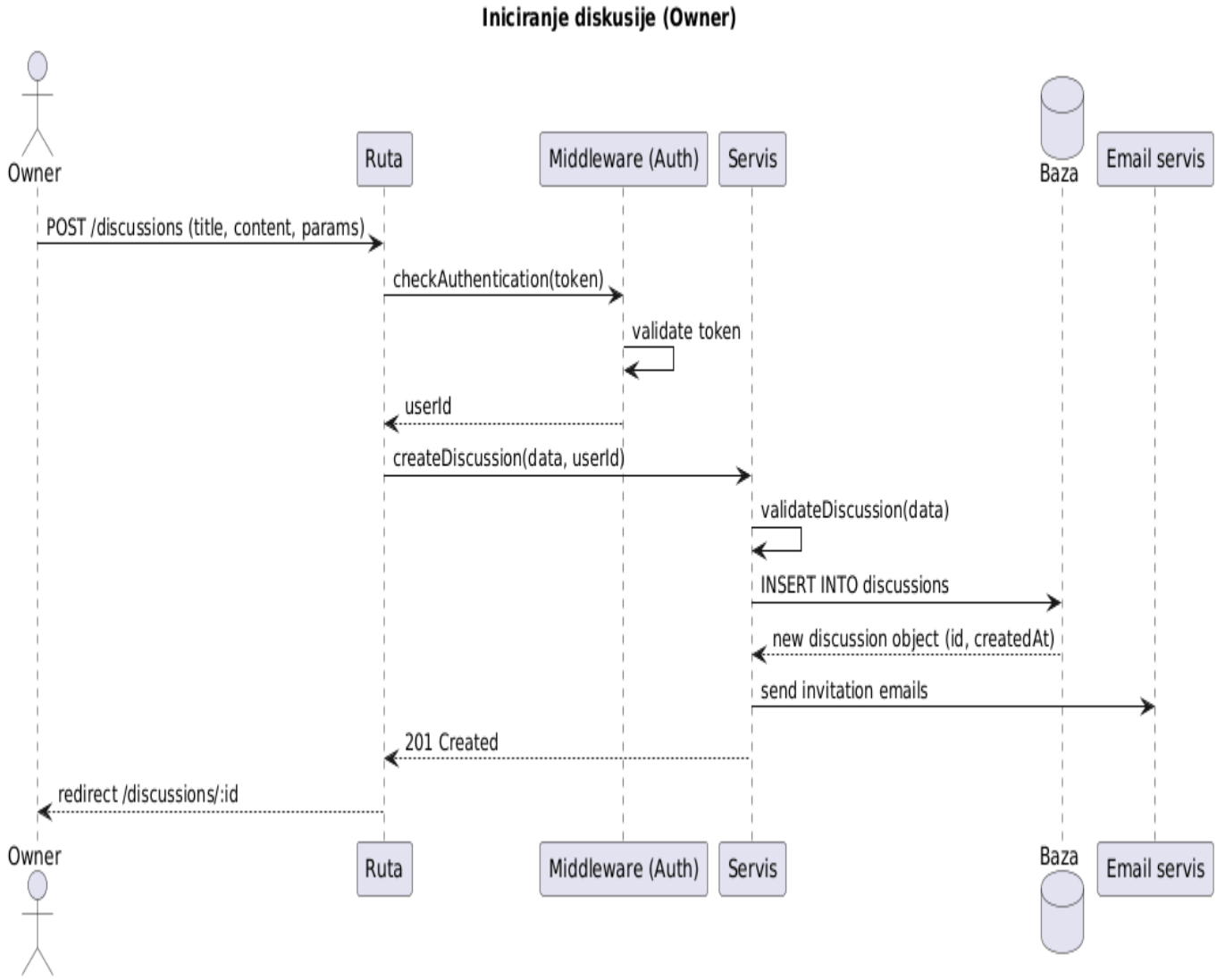
Sudionici: Suvlasnik, Sustav, Email servis
Preduvjet: Korisnik je prijavljen

Osnovni tijek

1. Suvlasnik odabire opciju „Nova diskusija”.
2. Suvlasnik unosi naslov, opis i parametre.
3. Sustav provjerava unose.
4. Sustav sprema diskusiju u bazu.
5. Ako je privatna — sustav šalje email pozvanim korisnicima.

Odstupanja

- **2a.** Nedostaje naslov → Sustav traži ispravak.
- **4a.** Baza nedostupna → Sustav prikazuje grešku.



UC04 – Upravljanje sudionicima

Glavni sudionik: Inicijator diskusije
Cilj: Dodati/ukloniti sudionike privatne diskusije

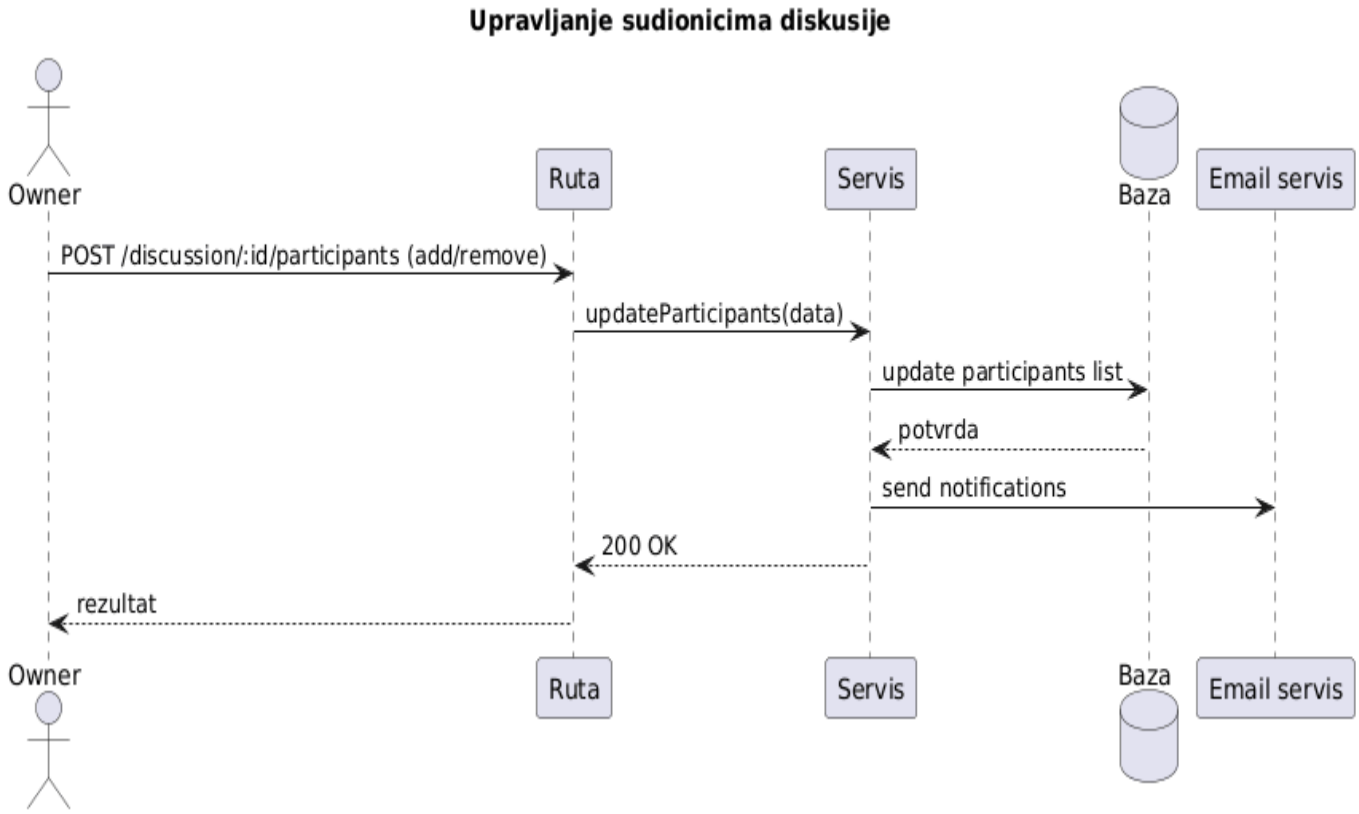
Sudionici: Inicijator, Sustav, Email servis
Preduvjet: Diskusija je privatna

Osnovni tijek

1. Inicijator otvara postavke diskusije.
2. Odabire dodavanje ili uklanjanje sudionika.
3. Sustav ažurira popis sudionika u bazi.
4. Sustav šalje email obavijest novim sudionicima.

Odstupanja

- **2a.** Korisnik već sudjeluje → Sustav javlja poruku.
- **3a.** Baza nedostupna → Sustav javlja problem.



Ovaj obrazac upotrebe opisuje proces dodavanja i uklanjanja sudionika u privatnoj diskusiji. Inicijator diskusije (owner) putem korisničkog sučelja pokreće zahtjev koji se šalje odgovarajućoj backend ruti. Backend servis provjerava valjanost zahtjeva, ažurira popis sudionika u bazi podataka i po potrebi aktivira email servis za slanje obavijesti novim sudionicima. Sustav osigurava da se korisnici ne mogu dodati dvaput te da se greške baze ili servisa pravilno prijave inicijatoru. Use case pokriva osnovni tijek upravljanja sudionicima te izvanredne situacije poput pokušaja dodavanja postojećeg sudionika ili nedostupnosti baze.

UC05 – Slanje i pregled poruka

Glavni sudionik: Korisnik
Cilj: Poslati poruku u diskusiji

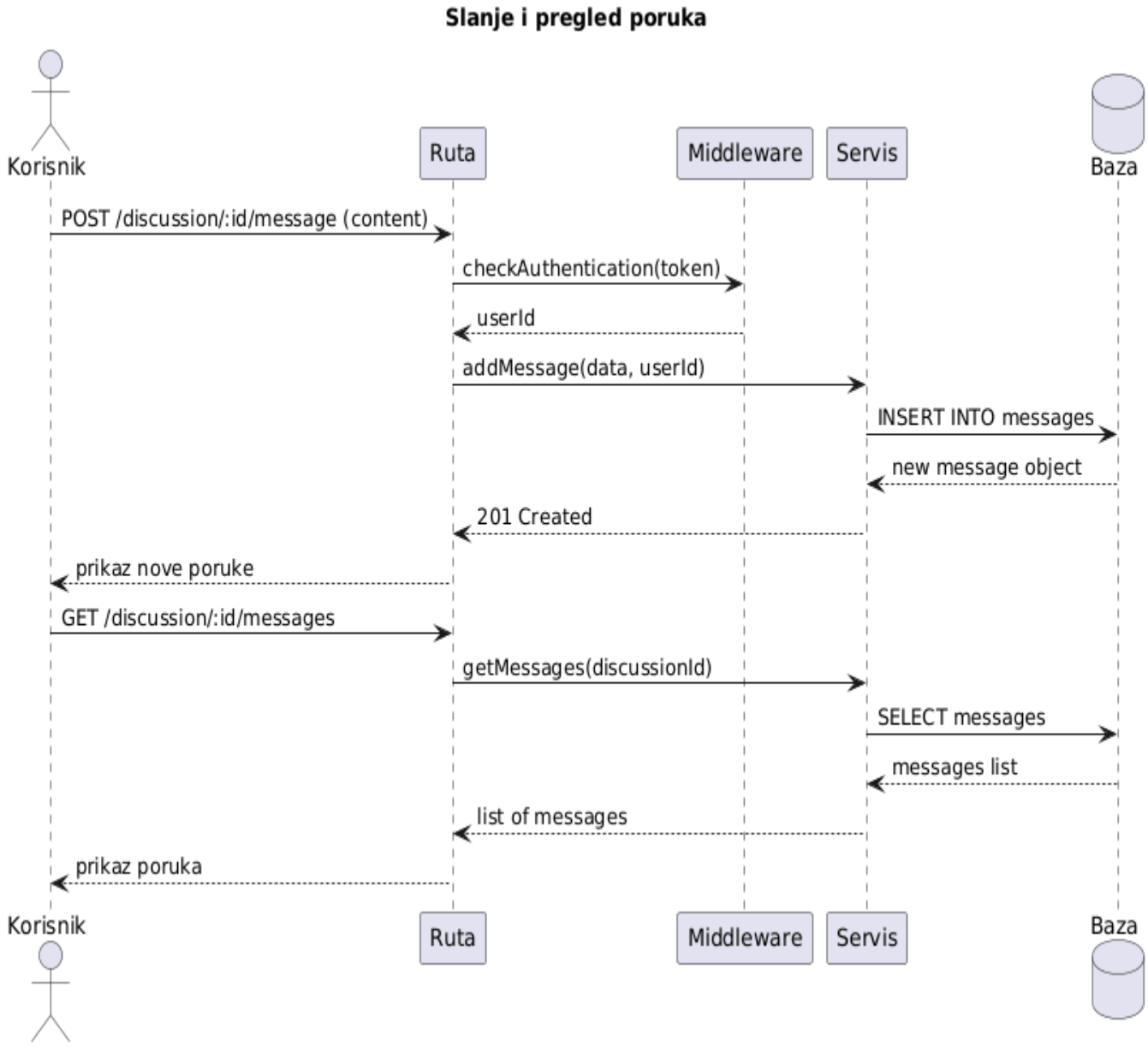
Sudionici: Korisnik, Sustav
Preduvjet: Korisnik ima pristup diskusiji

Osnovni tijek

1. Korisnik otvara diskusiju i unosi poruku.
2. Sustav provjerava je li korisnik ovlašten.
3. Sustav sprema poruku u bazu.
4. Sustav prikazuje novu poruku.

Odstupanja

- **2a.** Korisnik nije sudionik privatne diskusije → zabrana pristupa.
- **3a.** Neuspjeh zapisivanja → poruka o grešci.



UC06 – Glasovanje

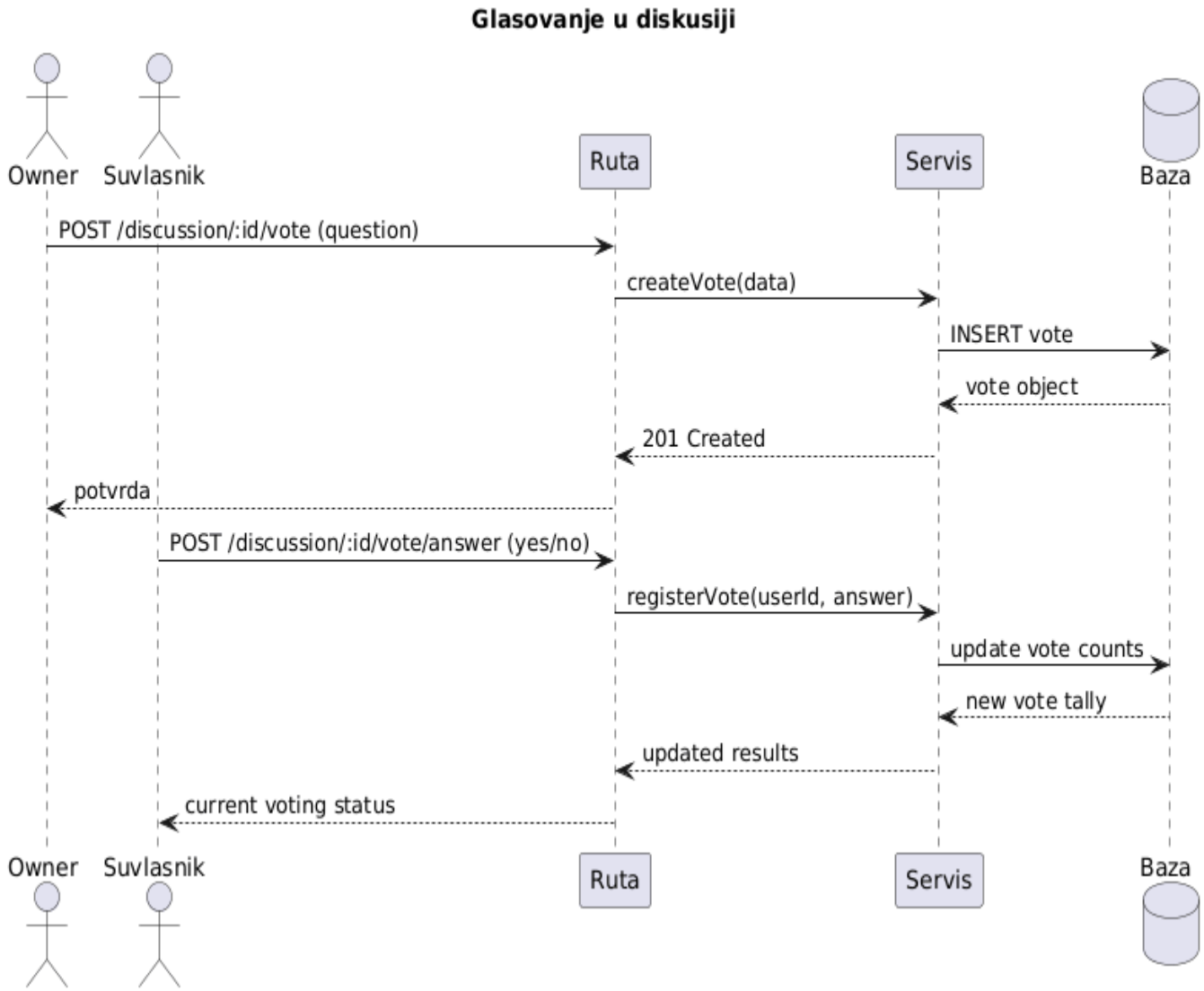
Glavni sudionik: Suvlasnik
Cilj: Glasati o prijedlogu u diskusiji
Sudionici: Suvlasnik, Sustav
Preduvjet: Glasanje je aktivno

Osnovni tijek

1. Inicijator pokreće glasanje.
2. Suvlasnik odabire svoj glas (DA/NE).
3. Sustav sprema glas.
4. Sustav ažurira broj glasova.
5. Sustav provjerava je li dosegnut prag od 25%.

Odstupanja

- **2a.** Korisnik je već glasao → sustav blokira drugi glas.
- **3a.** Neuspjeh zapisivanja → upozorenje.



UC07 – Kreiranje sastanka

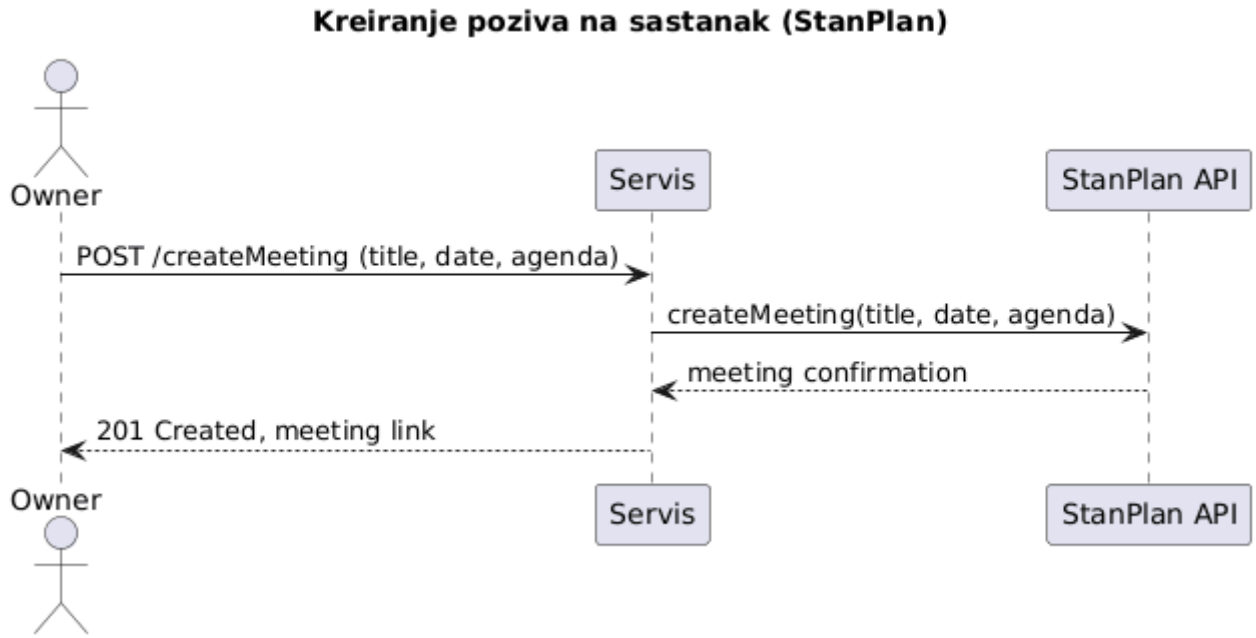
Glavni sudionik: Inicijator
Cilj: Kreirati sastanak u aplikaciji StanPlan
Sudionici: Inicijator, Sustav, StanPlan API
Preduvjet: Postotak pozitivnih glasova > 25%

Osnovni tijek

1. Korisnik odabire „Kreiraj sastanak”.
2. Sustav priprema podatke.
3. Sustav šalje zahtjev StanPlan API-ju.
4. StanPlan API vraća link sastanka.
5. Sustav prikazuje i sprema link sastanka.

Odstupanja

- **3a.** StanPlan API nedostupan → ponovni pokušaj.
- **4a.** Povratni podaci neispravni → ručni unos.



Ovaj obrazac upotrebe opisuje proces kreiranja sastanka putem integracije sa sustavom StanPlan. Kada broj pozitivnih glasova premaši zadani prag, inicijator može pokrenuti kreiranje sastanka. Korisnički zahtjev prolazi kroz backend rutu, gdje servis priprema potrebne podatke (naslov, opis, dnevni red, poveznica na diskusiju) i šalje ih StanPlan API-ju. Nakon uspješne obrade, StanPlan vraća link kreiranog sastanka, koji sustav prikazuje inicijatoru i sprema u bazu podataka. Use case pokriva i izvanredne situacije poput nedostupnosti StanPlan API-ja ili neispravnih povratnih podataka, u kojima sustav omogućuje ponovni pokušaj ili ručni unos podataka.

UC08 – Pregled diskusija

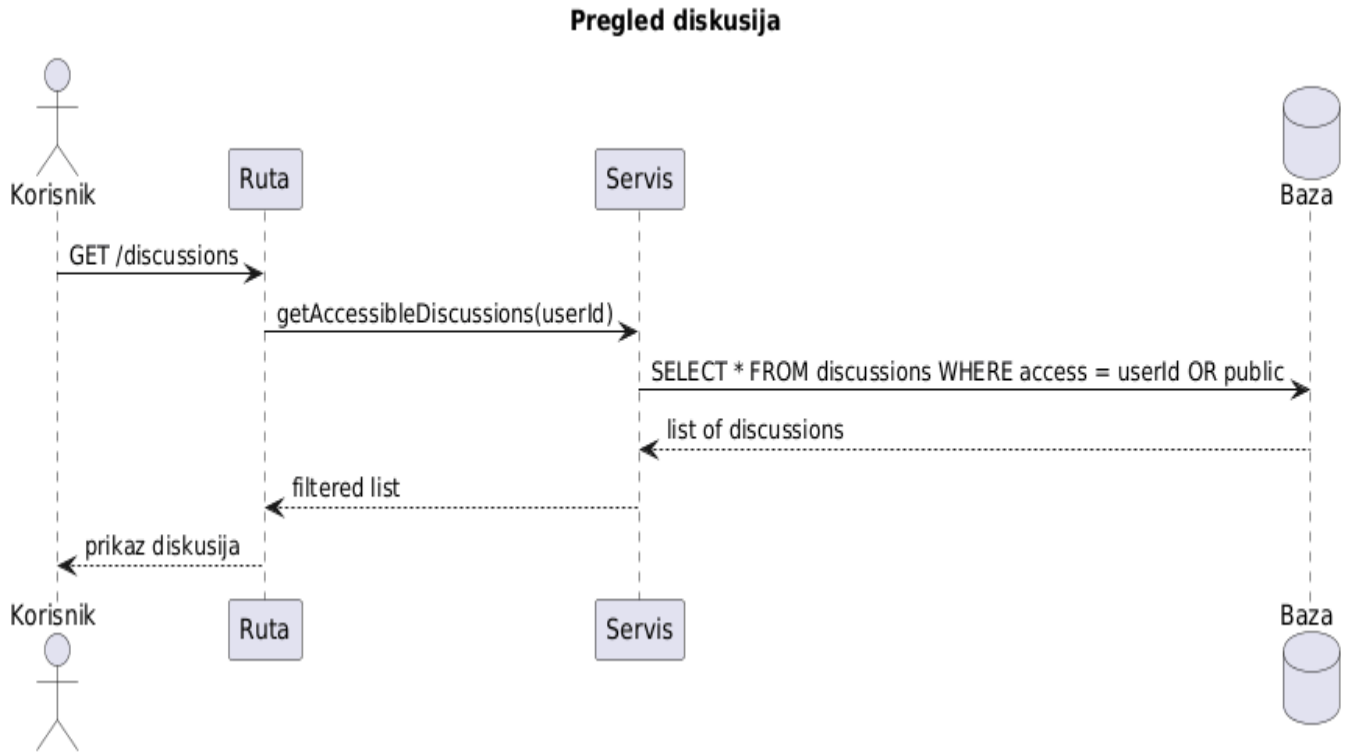
Glavni sudionik: Korisnik
Cilj: Vidjeti sve dostupne diskusije
Sudionici: Korisnik, Sustav
Preduvjet: Korisnik je prijavljen

Osnovni tijek

1. Korisnik otvara oglasnu ploču.
2. Sustav dohvaća sve diskusije.
3. Sustav filtrira privatne diskusije prema ovlaštenju.
4. Sustav prikazuje popis diskusija.

Odstupanja

- **2a.** Greška u bazi → sustav prikazuje prazan popis.



Provjera uključenosti ključnih funkcionalnosti u obrasce uporabe

UC	Funkcionalni zahtjevi
UC01 – Registracija / Prijava korisnika	F-001, F-002, F-003, F-004
UC02 – Kreiranje korisnika (Admin)	F-001, F-003
UC03 – Iniciranje diskusije	F-005, F-006, F-007, F-008
UC04 – Upravljanje sudionicima diskusije	F-006, F-007, F-008
UC05 – Slanje i pregled poruka	F-005
UC06 – Glasovanje	F-009, F-010, F-011
UC07 – Kreiranje sastanka	F-012, F-013
UC08 – Pregled diskusija	F-014, F-015

Arhitektura sustava

Opis arhitekture

•Stil arhitekture: Unutar ovog projekta korištena je klijent–poslužitelj arhitektura. Za ovaj pristup odlučili smo se zbog jednostavnosti razvoja, održivosti i potrebe za modularnošću, koja bi bila teško ostvariva u monolitskoj arhitekturi. S obzirom na relativno mali opseg i jednostavnost projekta, nije bilo potrebe za više od jednog backend servisa, pa samim time ni za primjenom mikrouslužne arhitekture.

• Podsustavi: Backend koristi Express.js, Node.js Frontend je napisan u Reactu u sklopu Vite razvojnog okruženja. Za bazu podataka koristimo Supabase te PostgreSQL relacijsku bazu podataka. Aplikacija će biti pokrenuta na Azure radnoj platformi jer adekvatno zadovoljava sve potrebe i zahtjeve naše aplikacije.

• Spremišta podataka: Sustav koristi relacijsku bazu podataka, implementiranu putem Supabase platforme koja je kompatibilna s PostgreSQL-om. Podaci se pohranjuju u jasno definirane tablice kao što su korisnici, zgrade, rasprave, poruke i glasovi, pri čemu se koriste primarni i vanjski ključevi radi osiguravanja referencijalnog integriteta. Ovakav model omogućuje strogu strukturu podataka, pouzdane veze među entitetima i učinkovito izvršavanje upita. Budući da sustav zahtijeva transakcijsku konzistentnost, kontrolu pristupa i jasne odnose između entiteta, relacijska baza se pokazala najboljim izborom u odnosu na NoSQL rješenja.

• Mrežni protokoli: Za komunikaciju između klijenta i poslužitelja koristi se HTTP protokol, pri čemu se podaci razmjenjuju putem REST API ruta definiranih u Express.js aplikaciji.
• Globalni upravljački tok: Korisnik putem preglednika šalje zahtjev (npr. za raspravu), koji backend obrađuje kroz odgovarajući podsustav, dohvaća podatke iz baze, te vraća rezultat u obliku stranice.

• Sklopovskoprogramski zahtjevi:

- Programski zahtjevi:
  - Poslužiteljski dio (backend)
    - Jezik i runtime: Node.js (verzija ≥ 18.x LTS, preporučeno 20.x)
    - Framework: Express.js 5.1.0
    - Render engine: EJS 3.1.10
    - Baza podataka: PostgreSQL (putem paketa pg i postgres)
    - Autentifikacija i sigurnost: JSON Web Tokens (jsonwebtoken) te hashiranje lozinki pomoću bcryptjs
    - Cloud integracija: Supabase (@supabase/supabase-js)
    - Konfiguracija: putem .env datoteka (paket dotenv)
    - Razvojni alati: nodemon za automatsko ponovno pokretanje poslužitelja tijekom razvoja
  - Operacijski sustav:
    - Kompatibilno s Windows 10+, macOS i Linux (npr. Ubuntu 20.04+)
    - Za produkcijsko okruženje preporučuje se Linux zbog stabilnosti i performansi
  - Dodatni alati:
    - Git za verzioniranje koda
    - Docker (opcionalno) za jednostavnije postavljanje produkcijskog okruženja
    - NPM (verzija ≥ 9.x) za upravljanje ovisnostima

- Sklopovski zahtjevi:
    - Za razvojno okruženje:
        - Procesor: minimalno 1 jezgra @ 2.0 GHze
        - Memorija: 1 GB RAM-a
        - Diskovni prostor: 500 MB slobodnog prostora
        - Internetska veza (za preuzimanje paketa i pristup Supabase servisu)
    - Za produkcijsko okruženje:
        - Procesor: 2+ jezgre @ 2.0 GHz
        - Memorija: 2–4 GB RAM-a (ovisno o broju istovremenih korisnika)
        - Diskovni prostor: 1 GB slobodnog prostora
        - Stalna mrežna povezanost (HTTP/HTTPS, port 3000 ili konfigurirani port)
        - Preporučeni deployment: Linux poslužitelj ili cloud platforma (npr. Vercel, Render, Supabase Hosting)

Obrazloženje odabira arhitekture

Glavni principi kojima smo se vodili prilikom odabira arhitekture bili su održljivosti i modularnost. Bilo nam je izuzetno bitno imati jasan, čitki kod tako da možemo povezati naše dijelove bez većih problema. Modularnost je bitna radi neovisnog razvijanja dijelova koje možemo spojiti na aplikaciju i nastaviti njen razvoj.
Odlučili smo se za ove arhitekture jer pružaju modularnost, jednostavnost i održljivost, a pružaju mnogo mogućnosti za moćno ostvarivanje web aplikacije.
Nismo razmatrili druge arhitekture jer nam ova arhitektura pruža sve što trebamo bez značajnih poteškoća, jednostavna je za korištenje i zadovoljava sve naše potrebe.

Organizacija sustava na visokoj razini

- Klijent-poslužitelj: Sustav je organiziran prema klijent-poslužitelj arhitekturi, gdje klijent (web preglednik) šalje zahtjeve putem HTTP protokola, a poslužitelj ih obrađuje pomoću Express.js aplikacije te vraća podatke ili renderirane stranice. Poslužitelj upravlja poslovnom logikom, autentikacijom i komunikacijom s bazom podataka.
- Baza podataka: U sustavu se koristi relacijska baza podataka PostgreSQL, koja služi za pohranu i dohvat strukturiranih podataka poput korisnika, rasprava, poruka, glasova i zgrada. Pristup bazi ostvaren je pomoću pg i postgres paketa, čime se omogućuje sigurna i učinkovita manipulacija podacima.
- Datotečni sustav: Supabase, PostgreSQL
- Grafičko sučelje: Korisničko sučelje temeljeno je na JSX i Javascript datotekama koje se renderiraju na poslužitelju i prikazuju korisniku putem preglednika. Sučelje ostvaruje korištenje funkcionalnosti sustava kao što su prijava, pregleda rasprava, glasanje i administracija putem REST API poziva.

Organizacija aplikacije

Opišite organizaciju aplikacije na složenijoj razini, uključujući strukturu slojeva i komponenata aplikacije:

• Frontend i Backend slojevi:

Backend sloj aplikacije organiziran je unutar direktorija /backend i temelji se na Node.js okruženju s frameworkom Express.js. Ovaj sloj predstavlja logičko središte sustava i zadužen je za obradu zahtjeva korisnika, komunikaciju s bazom podataka te generiranje odgovora koji se prosljeđuju frontend dijelu.

Struktura backend sloja:

- /auth – implementacija autentifikacije i autorizacije korisnika (JWT tokeni, bcrypt hashiranje lozinki)
- /data – sloj za pristup i upravljanje podacima iz baze
- /middleware – Express middleware funkcije za provjeru tokena, logiranje i obradu grešaka
- /routes – definicije REST API ruta koje povezuju zahtjeve korisnika s logikom aplikacije
- /services – pomoćne usluge i poslovna logika koje backend koristi za obradu podataka
- /utils – zajedničke pomoćne funkcije i alati
- server.js – glavni ulazni modul koji pokreće Express poslužitelj i inicijalizira povezivanje sa Supabaseom i bazom podataka

Odgovornosti backend sloja:

- Obrada HTTP zahtjeva i slanje odgovora klijentu *Upravljanje autentifikacijom i sesijama
- Sigurna pohrana i dohvat podataka iz baze
- Generiranje dinamičkih sadržaja (EJS) ili JSON odgovora
- Upravljanje konfiguracijom putem .env datoteka (pomoću paketa dotenv)
- Integracija s Supabase servisom kao cloud rješenjem za autentifikaciju i bazu

Frontend sloj nalazi se unutar direktorija /frontend-react i koristi React (razvojno okruženje Vite) za prikaz dinamičkih stranica generiranih na poslužitelju. Ovaj sloj je zadužen za korisničku interakciju te prikaz podataka koje backend vraća nakon obrade.

Struktura frontend sloja: Budući da je korišteno razvojno okruženje Vite, frontend folder ima puno autogeneriranih konfiguracijskij datoteka koje nisu potrebne za razumijevanje rada projekta, stoga će ovdje biti priložene jedino aktualni folderi i datoteke iz src foldera.

- /components - sadrži sve višenamjenske komponente koje se koriste na više stranica
- /pages - sadrži glavne stranice u aplikaciji
- /services - sadrži module koji komuniciraju s backend API-jem
- /utils - sadrži pomoćne funkcije koje se koriste u više dijelova aplikacije
- App.jsx - glavna React komponenta koja definira rute i osnovni layout aplikacije
- main.jsx - ulazna točka aplikacije koja slaže React aplikaciju u DOM i pokreće cijeli frontend

Odgovornosti frontend sloja:

- Prikaz korisničkog sučelja i rukovanje interakcijama korisnika
- Slanje zahtjeva backendu putem HTTP/HTTPS protokola (REST API)
- Prikaz rezultata koje backend vraća – u obliku renderirane HTML stranice ili JSON podataka

Interakcija između slojeva:

Frontend i backend međusobno komuniciraju isključivo putem HTTP/HTTPS protokola, pri čemu backend izlaže REST API rute definirane u Express.js aplikaciji. Korisnički zahtjev (npr. prijava, dohvat podataka ili objava sadržaja) prolazi kroz backend sloj, obrađuje se u odgovarajućem servisu te se vraća kao HTML prikaz ili JSON odgovor koji frontend prikazuje korisniku.

• MVC arhitektura:

Aplikacija koristi MVC (Model-View-Controller) arhitekturu, koja omogućuje jasno odvajanje poslovne logike, korisničkog sučelja i pristupa podacima, što olakšava održavanje i proširivost sustava.

Model (M)

Model sloj implementiran je unutar direktorija /data backend dijela aplikacije.
Odgovoran je za:
• Definiranje strukture podataka (npr. korisnici, rasprave, komentari)
• Interakciju s bazom podataka (PostgreSQL putem paketa pg i postgres)
• Operacije poput dohvaćanja, spremanja, ažuriranja i brisanja podataka

Model omogućuje da se poslovna logika backend sloja odvaja od načina na koji se podaci fizički pohranjuju i dohvaćaju.

View (V)

View sloj implementiran je u Reactu unutar direktorija /frontend-react/src, gdje se korisničko sučelje sastoji od React komponenti i stranica. Odgovornosti view sloja:

- Prikaz podataka dobivenih iz backend-a kroz React komponente
- Dinamičko generiranje sadržaja korištenjem React state-a i props-a
- Održavanje izgleda stranica i korisničkog iskustva aplikacije

View sloj je potpuno odvojen od poslovne logike, što omogućuje promjenu dizajna ili frontend tehnologije bez izmjene backend logike.

Controller (C)

Controller sloj organiziran je unutar direktorija /routes i djelomično kroz /services. Odgovoran je za:

- Primanje zahtjeva od frontend sloja (HTTP GET/POST/PUT/DELETE)
- Pozivanje odgovarajućih model funkcija za obradu podataka
- Pripremu podataka za view ili kao JSON odgovor klijentu
- Upravljanje poslovnom logikom i provjerom autentifikacije (pomoću middleware funkcija)

Controller povezuje frontend i backend, te osigurava da se logika prezentacije i logika podataka odvajaju od same obrade zahtjeva.

Baza podataka

Vrsta i implementacija baze

Odabrana baza podataka je **PostgreSQL**, implementirana preko **Supabase** platforme u oblaku. Za primarne ključeve koristi se tip **UUID** radi jedinstvene identifikacije unutar svih tablica. Baza je strukturirana tako da omogućuje učinkovito praćenje korisnika, zgrada, diskusija, poruka, anketa i glasova.

Tablice baze

Tablica: app_user

Atribut	Tip podatka	Opis varijable
id	UUID	Primarni ključ (PK), jedinstveni identifikator korisnika

Atribut	Tip podatka	Opis varijable
first_name	VARCHAR	Ime korisnika
last_name	VARCHAR	Prezime korisnika
email	VARCHAR	Jedinstvena email adresa korisnika, UNIQUE
phone_number	VARCHAR	Broj telefona korisnika
role	role_t	Uloga korisnika (admin, suvlasnik, predstavnik)
created_at	TIMESTAMPTZ	Datum i vrijeme stvaranja korisničkog računa
password_hash	TEXT	Hash lozinke korisnika

Tablica: building

Atribut	Tip podatka	Opis varijable
id	UUID	Primarni ključ (PK), jedinstveni identifikator zgrade
name	VARCHAR	Naziv zgrade
address	VARCHAR	Adresa zgrade
created_at	TIMESTAMPTZ	Datum i vrijeme unosa zgrade

Tablica: building_membership

Atribut	Tip podatka	Opis varijable
building_id	UUID	Primarni i strani ključ (FK), povezuje s tablicom building
user_id	UUID	Primarni i strani ključ (FK), povezuje s tablicom app_user
user_role	role_t	Uloga korisnika unutar zgrade

Tablica: discussion

Atribut	Tip podatka	Opis varijable
id	UUID	Primarni ključ (PK), jedinstveni identifikator diskusije
title	VARCHAR	Naslov diskusije, UNIQUE u kombinaciji s building_id
poll_description	VARCHAR	Opis ankete u diskusiji
building_id	UUID	Strani ključ (FK), povezuje s tablicom building
owner_id	UUID	Strani ključ (FK), povezuje s tablicom app_user, vlasnik diskusije
visibility	visibility_t	Vidljivost diskusije (javno/privatno)
status	discussion_status_t	Status diskusije (otvoreno/zatvoreno)
created_at	TIMESTAMPTZ	Datum i vrijeme kreiranja diskusije

Tablica: discussion_participant

Atribut	Tip podatka	Opis varijable
discussion_id	UUID	Primarni i strani ključ (FK), povezuje s tablicom discussion
user_id	UUID	Primarni i strani ključ (FK), povezuje s tablicom app_user
can_post	BOOLEAN	Omogućuje korisniku pisanje poruka u diskusiji
number_of_messages	INTEGER	Broj poruka koje je korisnik poslao, mora biti >= 0

Tablica: message

Atribut	Tip podatka	Opis varijable
id	UUID	Primarni ključ (PK), jedinstveni identifikator poruke
discussion_id	UUID	Strani ključ (FK), povezuje s tablicom discussion
author_id	UUID	Strani ključ (FK), povezuje s tablicom app_user, autor poruke
body	TEXT	Sadržaj poruke
created_at	TIMESTAMPTZ	Datum i vrijeme slanja poruke

Tablica: poll

Atribut	Tip podatka	Opis varijable
id	UUID	Primarni ključ (PK), jedinstveni identifikator ankete
discussion_id	UUID	Strani ključ (FK), povezuje s tablicom discussion
author_id	UUID	Strani ključ (FK), povezuje s tablicom app_user, autor ankete
question	TEXT	Pitanje ankete
created_at	TIMESTAMPTZ	Datum i vrijeme kreiranja ankete
closed	BOOLEAN	Status ankete (true ako je zatvorena)

Tablica: upload

Atribut	Tip podatka	Opis varijable
id	UUID	Primarni ključ (PK), jedinstveni identifikator datoteke
message_id	UUID	Strani ključ (FK), povezuje s tablicom message
url	TEXT	URL lokacija datoteke
filename	TEXT	Naziv datoteke
content_type	TEXT	MIME tip datoteke
size_bytes	INTEGER	Veličina datoteke u bajtovima

Atribut	Tip podatka	Opis varijable
created_at	TIMESTAMPTZ	Datum i vrijeme dodavanja datoteke

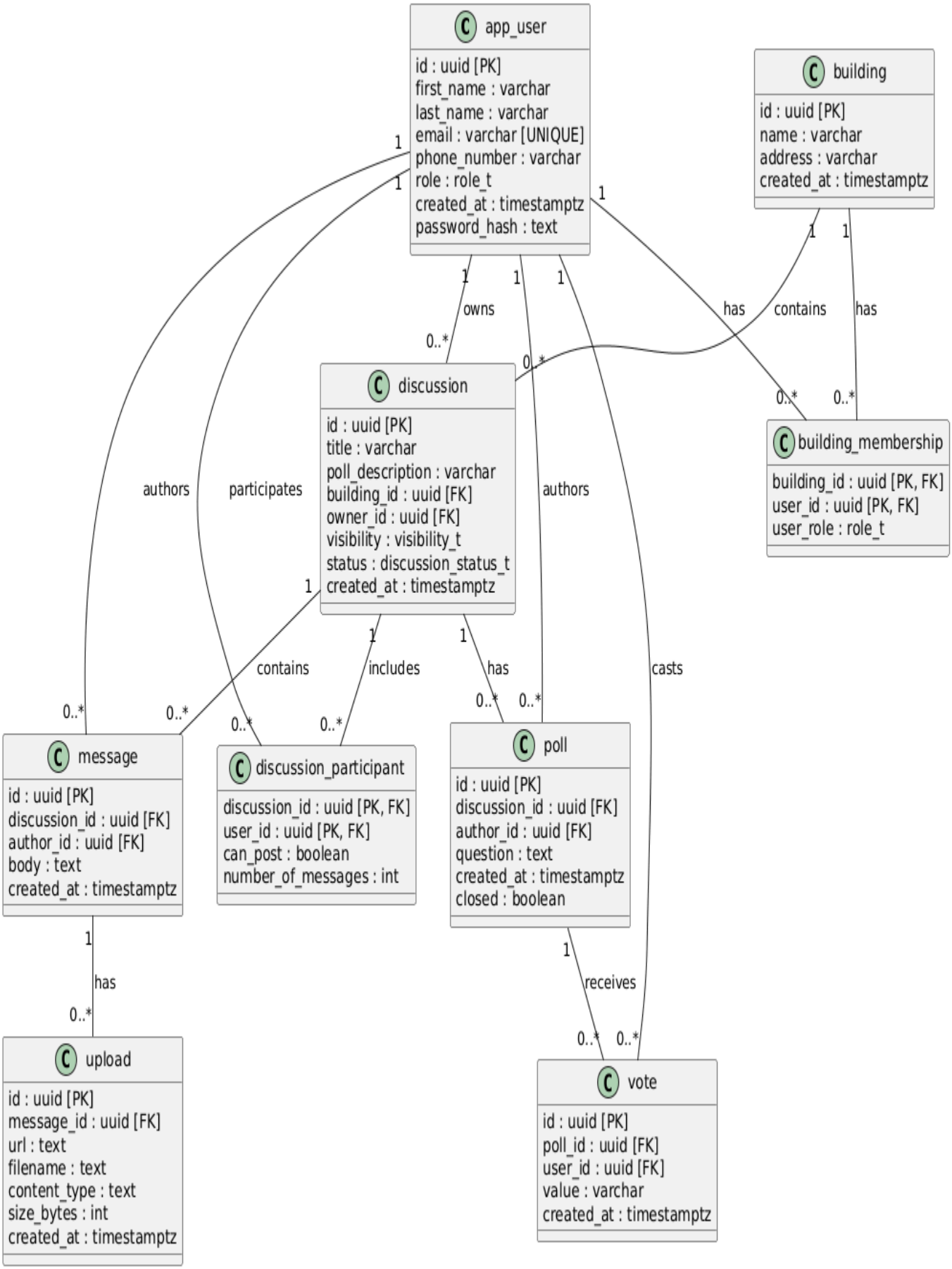
Tablica: vote

Atribut	Tip podatka	Opis varijable
id	UUID	Primarni ključ (PK), jedinstveni identifikator glasa
poll_id	UUID	Strani ključ (FK), povezuje s tablicom poll
user_id	UUID	Strani ključ (FK), povezuje s tablicom app_user
value	vote_choice_t	Vrijednost glasa (da/ne)
created_at	TIMESTAMPTZ	Datum i vrijeme glasanja

Tablica: stanplan

Atribut	Tip podatka	Opis varijable
id	UUID	Primarni ključ (PK), jedinstveni identifikator linka
link	TEXT	Poveznica na StanPlan API

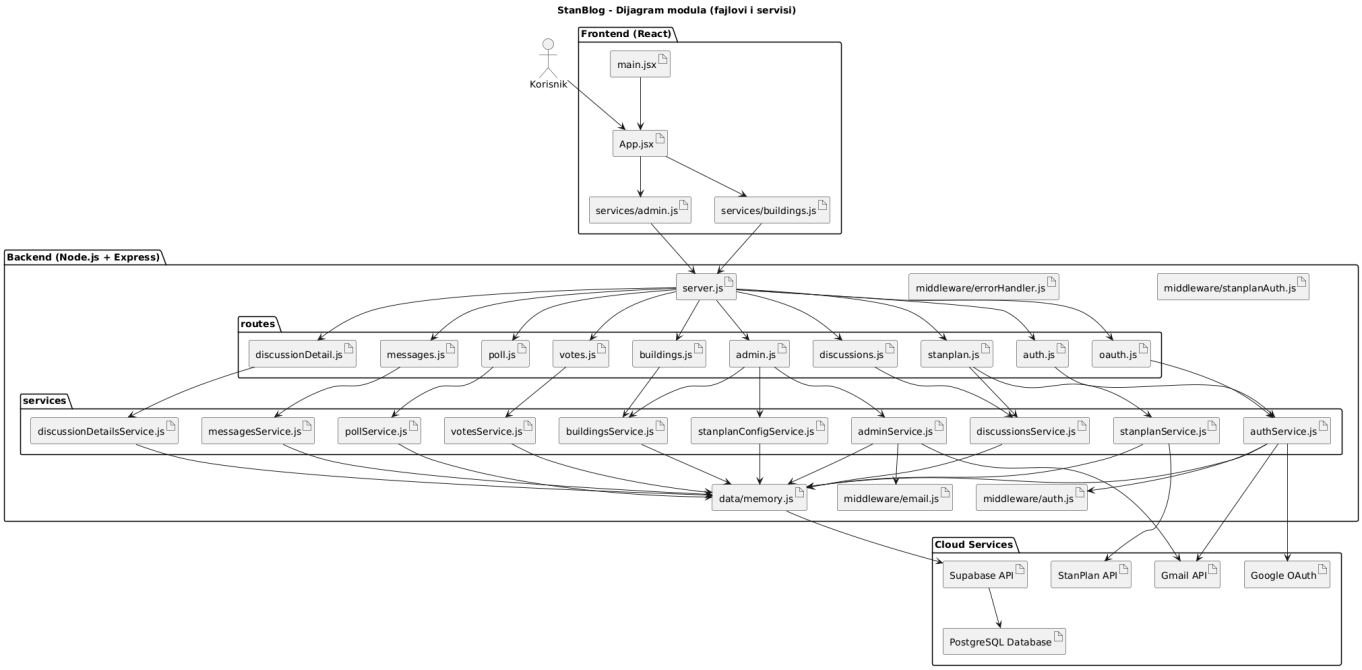
Dijagram baze podataka



Odnosi tablica i objekata u bazi podataka.

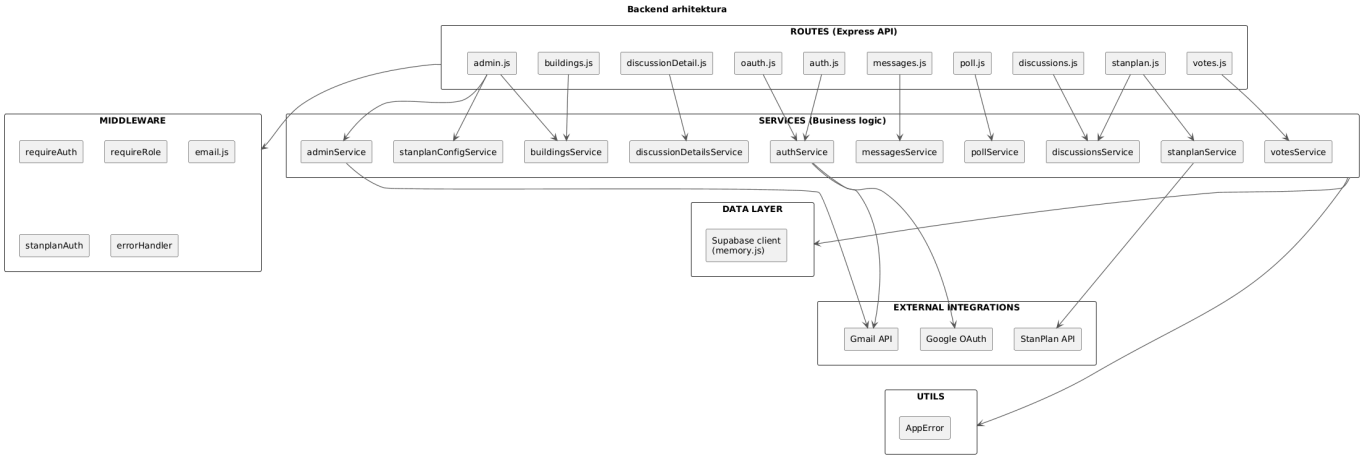
Dijagram razreda

Iako aplikacija ne koristi objektno-orijentirani pristup niti definirane klase, dijagram razreda služi kao apstraktni prikaz strukture sustava. Umjesto razreda, backend i frontend dio aplikacije organizirani su kroz JavaScript module i datoteke koje grupiraju povezane funkcije i odgovornosti. Dijagram zato prikazuje pojedine .js datoteke kao logičke jedinice, zajedno s njihovim ulogama i međusobnim odnosima. Ovaj pristup omogućuje da se funkcionalna arhitektura sustava jasno prikaže, iako se implementacija temelji na modularnim fileovima, a ne na klasama u tradicionalnom smislu.



- **Dijagram arhitekture sustava**
  Ovaj dijagram prikazuje osnovnu strukturu sustava kroz povezane module frontenda, backenda i vanjskih servisa. Frontend komunicira s backendom preko API poziva, dok backend obrađuje zahtjeve kroz rute i pripadajuće servise. Servisi koriste Supabase klijent za pristup bazi podataka te po potrebi uključuju vanjske integracije poput Google OAuth-a, Gmail API-ja i StanPlan API-ja. Dijagram daje pregled glavnih komponenti i njihovih međusobnih veza unutar sustava.

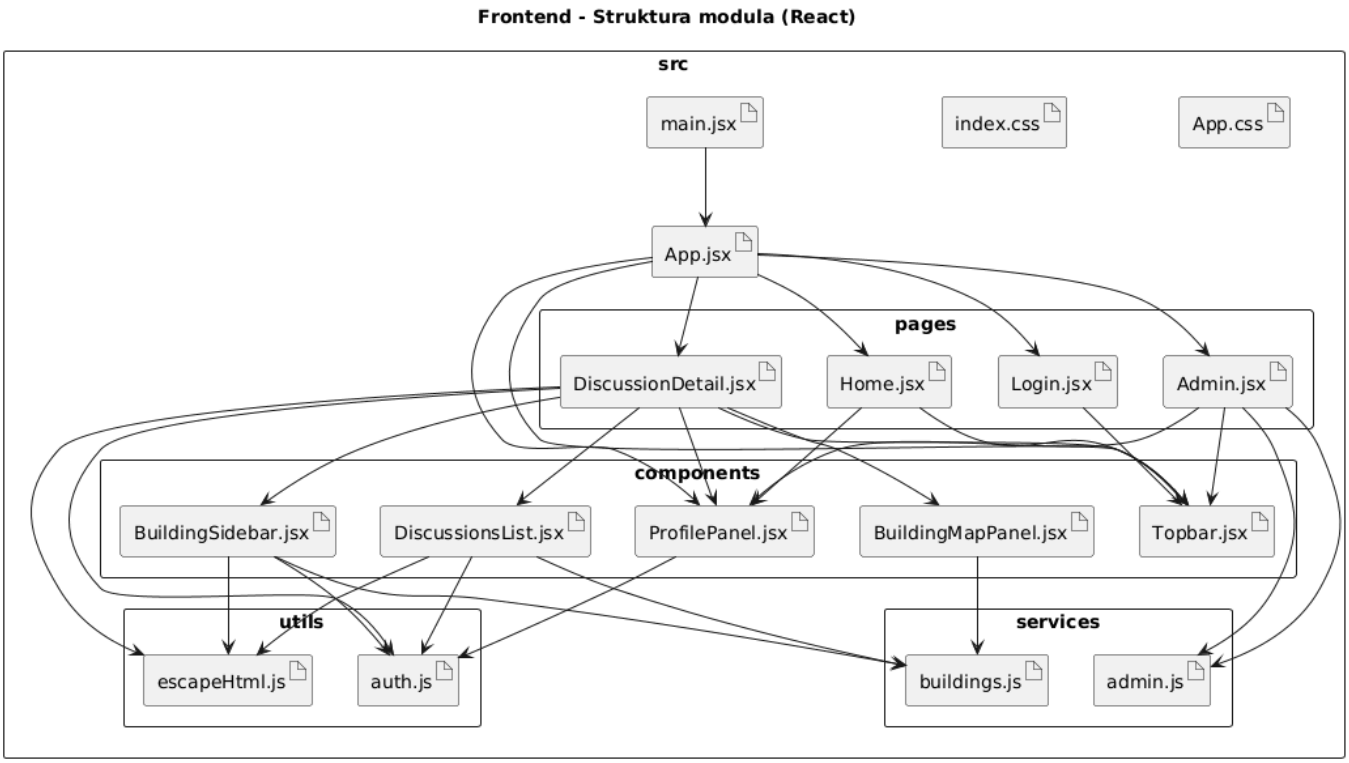
- **Dijagram backend modula (servisi i logika)**



Ovaj dijagram prikazuje backend organiziran u slojeve: rute, middleware, servise, podatkovni sloj i vanjske integracije. Rute predstavljaju ulazne točke API-ja i prosljeđuju zahtjeve prema servisima nakon prolaska kroz middleware. Servisi sadrže poslovnu logiku i komuniciraju sa Supabase bazom te vanjskim servisima. Unutar

utils foldera nalazi se djeljeni file AppError.js. Dijagram jasno prikazuje strukturu backend modula i njihove međusobne veze.

- **Dijagram frontend modula**



Dijagram prikazuje organizaciju React frontend modula raspoređenih po komponentama, stranicama, servisima i pomoćnim util funkcijama. Strelice jasno označavaju ovisnosti: stranice koriste vizualne komponente, komponente dohvaćaju podatke kroz servisne module, a utili služe za autentifikaciju i sanitizaciju sadržaja. U središtu strukture nalazi se App.jsx kao glavnu komponentu koja povezuje sve dijelove aplikacije, dok main.jsx predstavlja ulaznu točku Vite okruženja.

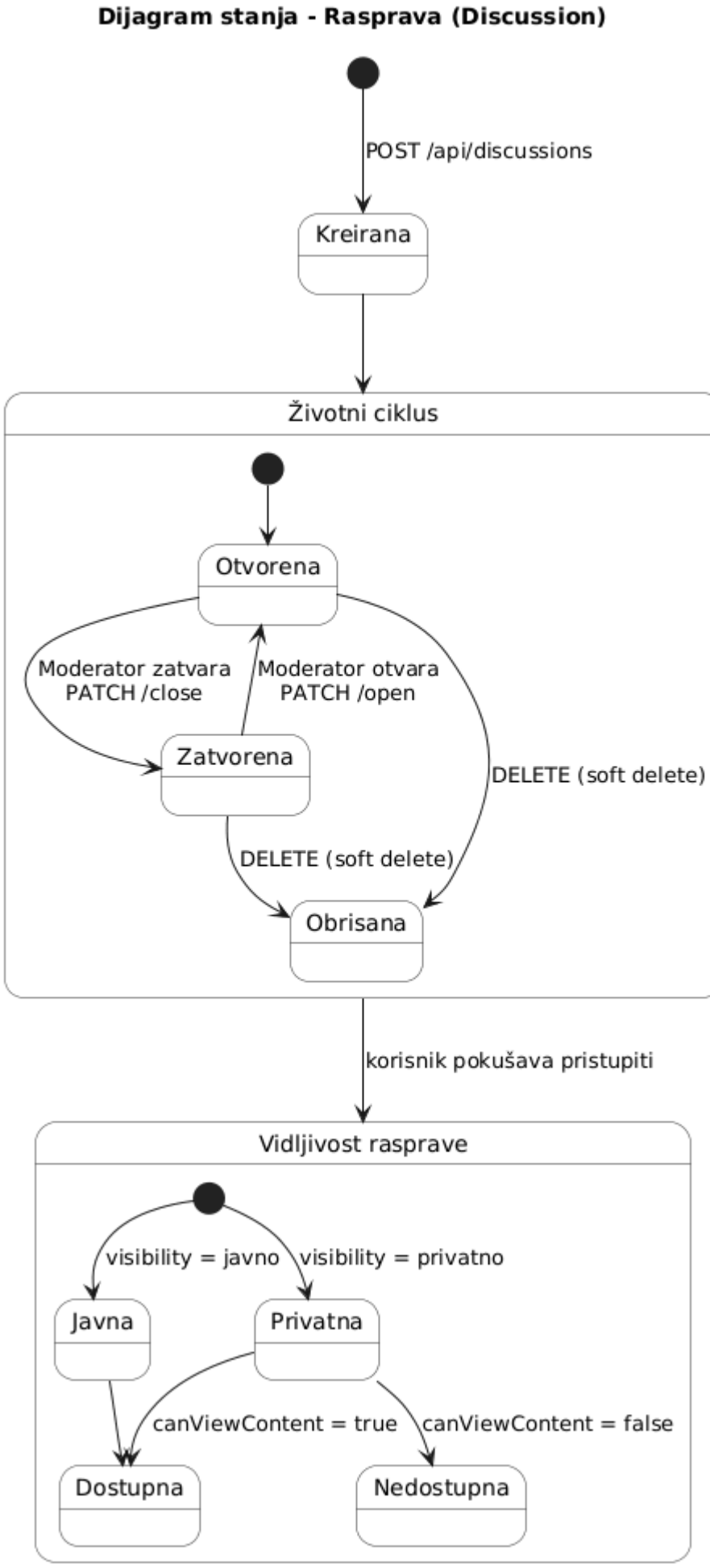
Dinamičko ponašanje aplikacije

Dinamičko ponašanje aplikacije odnosi se na način na koji objekti u sustavu evoluiraju kroz vrijeme, uključujući prijelaze između različitih stanja. To uključuje aktivnosti, događaje, odluke i interakcije unutar aplikacije. UML dijagrami stanja omogućuju vizualizaciju tih promjena i olakšavaju razumijevanje dinamike sustava.

Razumijevanje promjena stanja neophodno je za pravilno funkcioniranje aplikacije jer pruža uvid u interakcije među objektima, komponentama i korisnicima tijekom rada sustava. Korištenjem UML dijagrama stanja i aktivnosti moguće je vizualizirati prijelaze i stanja objekata, identificirati potencijalne probleme, osigurati točnu implementaciju te poboljšati komunikaciju među članovima tima.

UML dijagrami stanja

UML dijagrami stanja nužni su za razumijevanje dinamičkog ponašanja sustava. Oni jasno prikazuju promjene stanja objekata tijekom vremena ovisno o događajima i uvjetima.

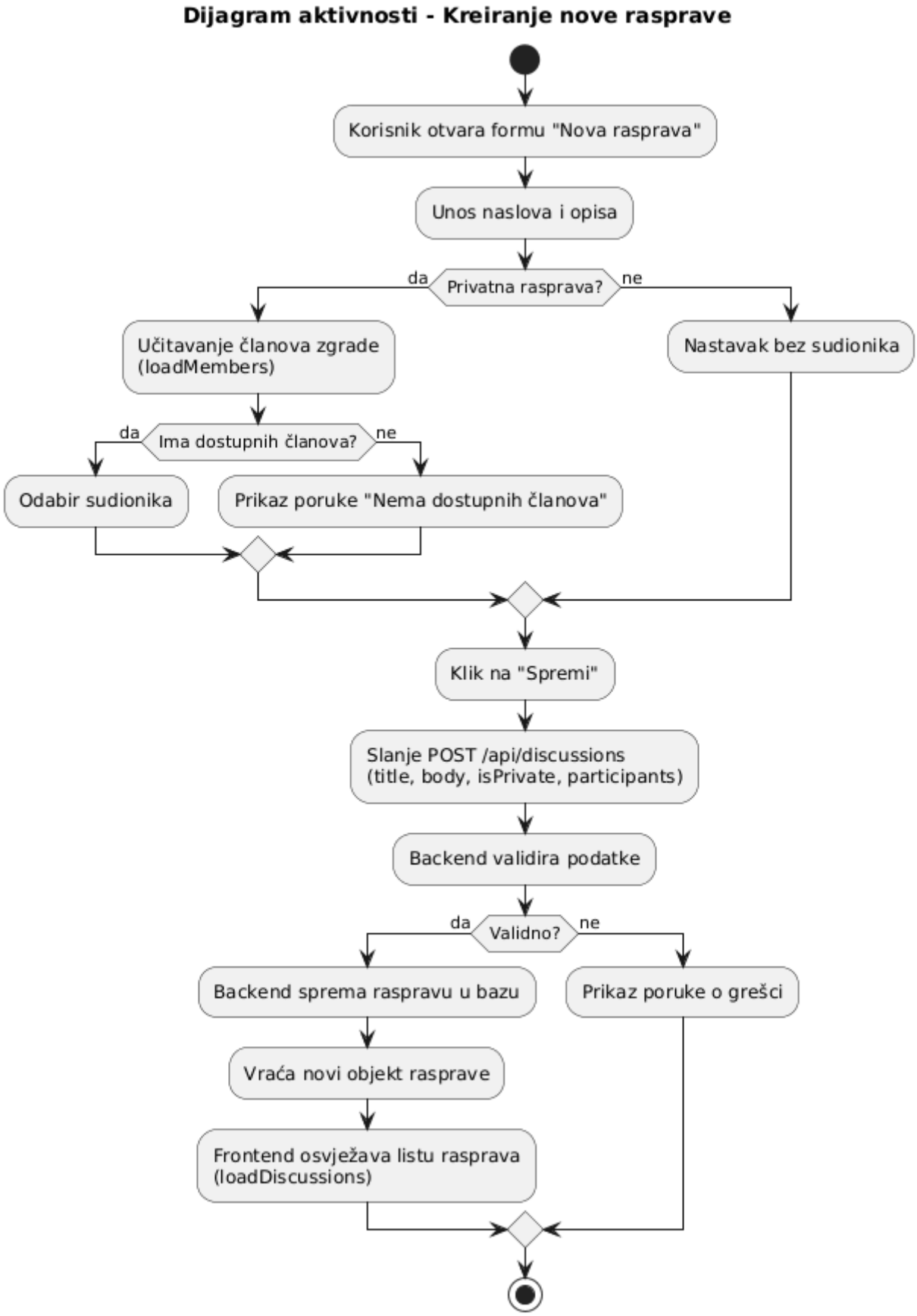

Dijagram stanja - Rasprava (Discussion)
POST /api/discussions
Kreirana
Životni ciklus
Otvorena
Moderator zatvara
PATCH /close
Moderator otvara
PATCH /open
Zatvorena
DELETE (soft delete)
DELETE (soft delete)
Obrisana
korisnik pokušava pristupiti
Vidljivost rasprave
visibility = javno
visibility = privatno
Javna
Privatna
canViewContent = true
canViewContent = false
Dostupna
Nedostupna

Dijagram stanja prikazuje dinamičko ponašanje objekta Rasprava kroz dva odvojena, ali povezana aspekta: životni ciklus i vidljivost. Nakon kreiranja, rasprava ulazi u stanje Otvorena, gdje je dostupna za pregled i dodavanje poruka. Moderator može promijeniti stanje u Zatvorena, čime se onemogućuje daljnje sudjelovanje, ali sadržaj ostaje vidljiv. U bilo kojem trenutku rasprava može biti označena kao Obrisana, što predstavlja završno stanje u kojem više nije dostupna korisnicima.

Neovisno o statusu, rasprava može biti Javna ili Privatna. Javne rasprave uvijek su dostupne svim prijavljenim korisnicima, dok privatne rasprave imaju dodatnu razinu kontrole pristupa. Ovisno o tome je li korisnik ovlašten, privatna rasprava prelazi u stanje Dostupna ili Nedostupna. Ovaj dijagram jasno prikazuje kako različiti događaji i uvjeti utječu na ponašanje rasprave tijekom njezina životnog ciklusa.

UML dijagrami aktivnosti

Dijagram aktivnosti prikazuje tijek izvršavanja određenog procesa. Osim za razumijevanje toka podataka unutar aplikacije, koristi se za analizu poslovnih procesa.



Dijagram aktivnosti prikazuje proces kreiranja nove rasprave u aplikaciji.Korisnik otvara formu, unosi osnovne podatke i odlučuje hoće li rasprava biti privatna. Ako je privatna, sustav dohvaća članove zgrade i omogućuje odabir sudionika. Nakon potvrde, frontend šalje POST zahtjev backendu, koji provodi validaciju i sprema raspravu u bazu. U slučaju uspjeha, frontend ponovno učitava listu rasprava, dok se u slučaju pogreške prikazuje odgovarajuća poruka.

Arhitektura sustava predstavlja temeljni okvir za razumijevanje i implementaciju svih njegovih funkcionalnosti. U kontekstu razvojne dokumentacije aplikacija, dijagrami komponenata i razmještaja odlučujući su za prikaz povezanosti i rasporeda različitih komponenata sustava. Ovi dijagrami omogućuju sudionicima projekta razumijevanje i vizualizaciju fizičkog i logičkog dizajna sustava, uključujući interakcije između dijelova aplikacije, što je odlučujuće za efikasnu implementaciju i dugoročnu održivost sustava.

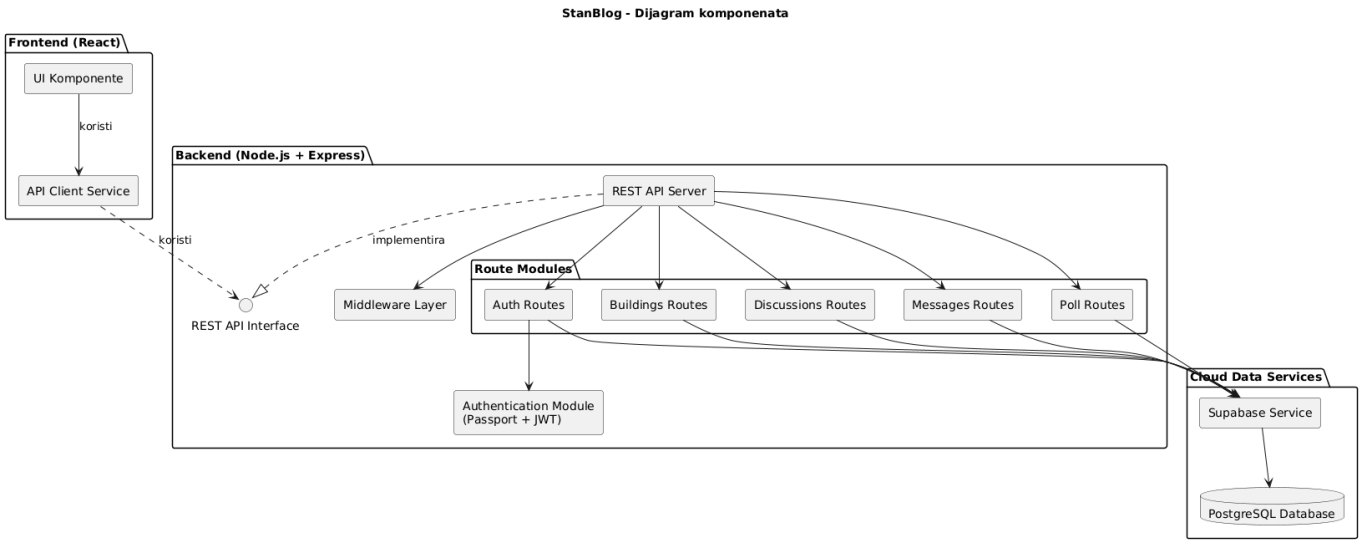
Arhitektura sustava, u kontekstu dijagrama komponenata i razmještaja, pruža uvid u strukturu i raspored ključnih dijelova aplikacije. Ovi dijagrami nisu korisni samo tijekom faza oblikovanja i implementacije, već služe i kao alati za održavanje i optimizaciju sustava u budućnosti.

Kao dio razvoja aplikacije, važno je osmisliti i dokumentirati arhitekturu sustava s naglaskom na dijagrame komponenata i razmještaja. Vaš zadatak je izraditi **dijagram komponenata** koji će jasno prikazivati ključne funkcionalne komponente aplikacije, njihovu međusobnu povezanost te sučelja za komunikaciju. Također, trebate izraditi **dijagram razmještaja** komponenata, koji treba detaljno prikazivati kako su te komponente raspoređene u infrastrukturi sustava, uključujući fizičke i virtualne resurse poput poslužitelja ili uređaja krajnjih korisnika.

Dijagram komponenata

Komponente sustava predstavljaju bitne dijelove aplikacije koji obavljaju specifične funkcije. Svaka komponenta je autonomna jedinica s vlastitim odgovornostima, ali je povezana s drugim komponentama kako bi sustav u cjelini funkcionirao. Komponente mogu biti elementi poput modula, servisa, razreda ili paketa, te komuniciraju putem jasno definiranih sučelja.

UML dijagram komponenata za vašu aplikaciju ovisi o njezinoj arhitekturi i složenosti, no općenito treba prikazivati važne funkcionalne komponente, njihovu međusobnu povezanost i sučelja za komunikaciju. Obzirom na namjenu projekta, dijagram treba biti jasan, organiziran i lako čitljiv kako bi olakšao razumijevanje strukture i suradnju unutar tima.



Dijagram komponenata prikazuje glavne funkcionalne cjeline aplikacije i njihove međusobne odnose. Klijentski dio aplikacije sastoji se od React UI komponenata i API klijentskog sloja koji komunicira s backendom putem jasno definiranog REST API sučelja. Poslužiteljski dio implementiran je u Node.js i Express okruženju te je podijeljen na modularne rute za autentifikaciju, upravljanje zgradama, raspravama, porukama i anketama, uz

zajednički sloj middlewarea i autentifikacijsku logiku. Svi backend moduli pristupaju podatkovnom sloju putem Supabase servisa, koji koristi relacijsku PostgreSQL bazu podataka.

Dijagram razmještaja

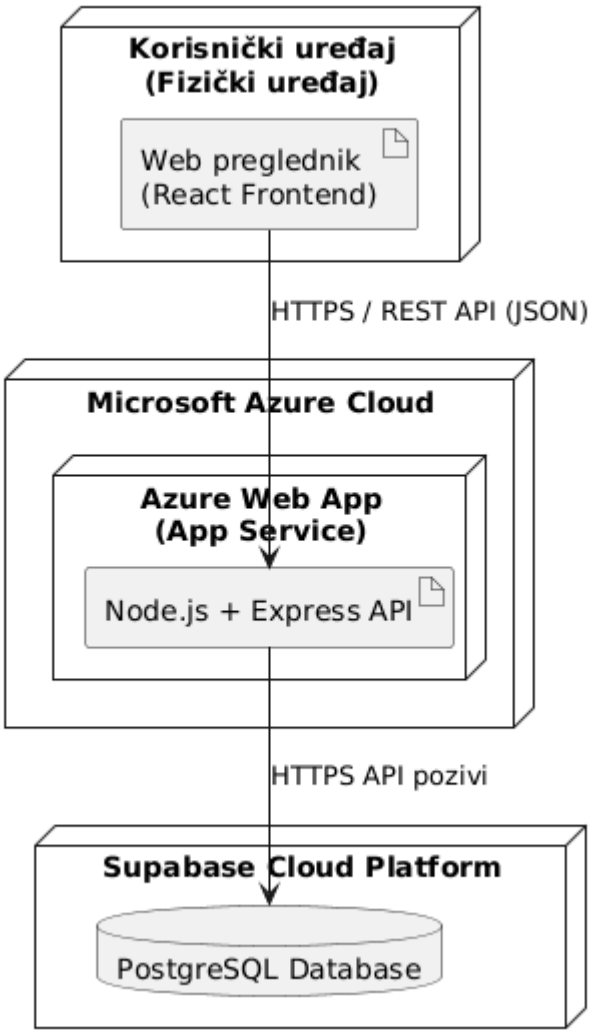
UML dijagram razmještaja prikazuje fizičku ili virtualnu raspodjelu komponenata sustava unutar infrastrukture. Cilj je prikazati kako su komponente raspoređene (npr. na poslužiteljima, u okruženjima oblaka ili na uređajima krajnjih korisnika) te način komuni čije, API-ja ili drugih komunikacijskih protokola.

U ovoj dokumentaciji preporučuje se uključiti dijagram razmještaja instanci (engl. Instance Level Deployment Diagram) ili implementacije.

- Dijagram razmještaja instanci prikazuje način na koji su aplikacijske komponente raspoređene unutar infrastrukture, uključujući fizičke i virtualne čvorove (poslužitelje) na kojima se izvode. Ovaj dijagram detaljno opisuje kako su komponente povezane i kako međusobno komuniciraju.

> U kontekstu aplikacije, npr. ona koja koristi Docker kontejner, dijagram razmještaja instanci pokazuje kako se aplikacije raspoređuju unutar Docker kontejnera na različitim poslužiteljima ili u okruženjima oblaka.

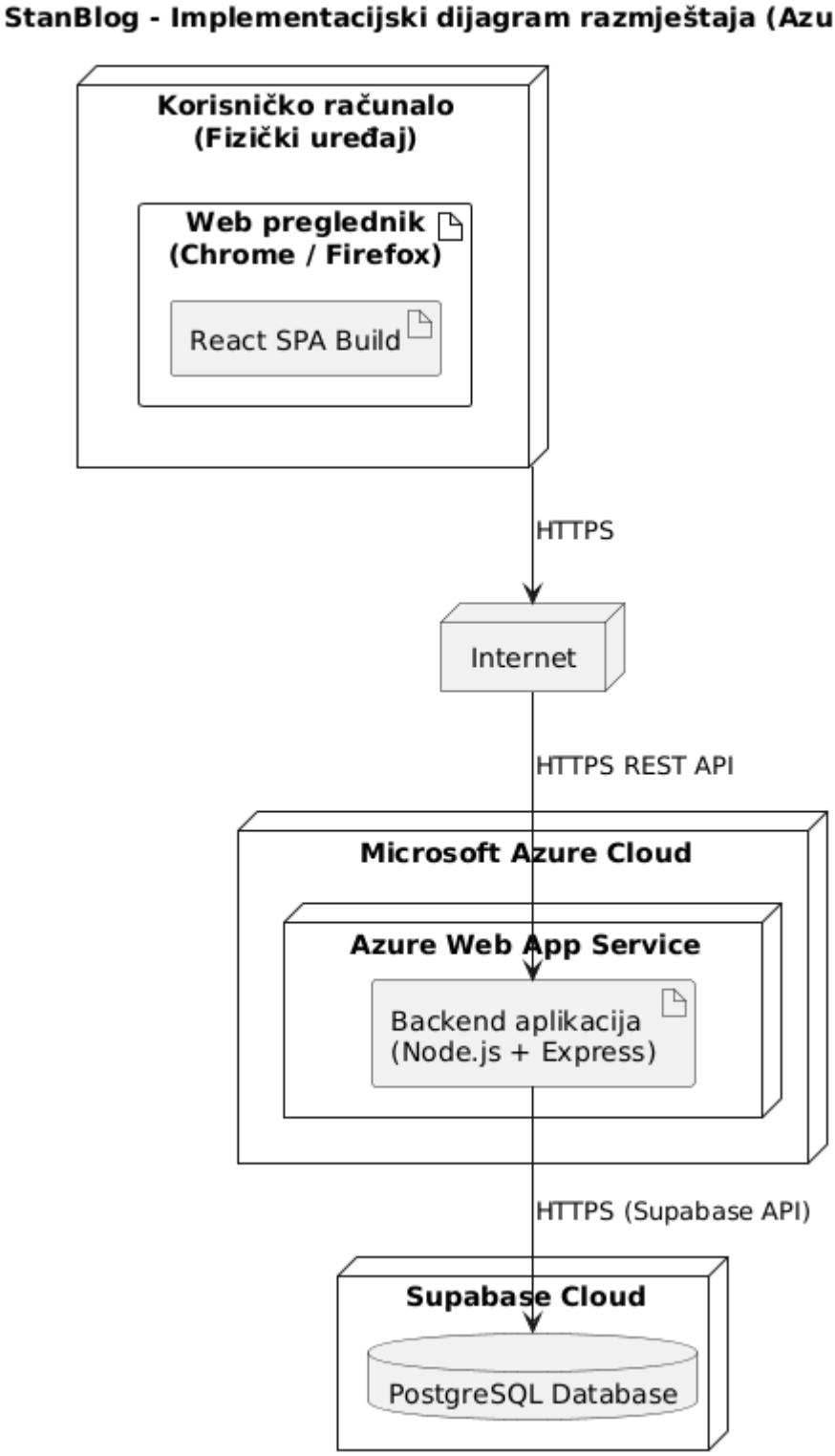
StanBlog - Dijagram razmještaja instanci (Azure)



Dijagram razmještaja instanci prikazuje kako su aplikacijske komponente raspoređene između korisničkog uređaja i cloud infrastrukture. Frontend aplikacija se izvršava u web pregledniku korisnika, dok se backend aplikacija izvodi na servisu Azure Web App unutar platforme Microsoft Azure. Komunikacija između klijenta i poslužitelja odvija se putem HTTPS REST API-ja uz razmjenu JSON podataka. Podaci se pohranjuju i dohvaćaju putem vanjskog servisa Supabase, koji koristi PostgreSQL bazu podataka. **Implementacijski oblik**

- Implementacijski oblik daje detaljan prikaz rasporeda komponenata u fizičkoj ili virtualnoj infrastrukturi. Ovdje se koriste stvarni poslužitelji, mrežne veze, uređaji korisnika, aplikacijski paketi i drugi artefakti kako bi dijagram prikazao točan fizički raspored komponenata, s naglaskom na fizičko povezivanje i tehničke resurse.

> **Primjeri:** Raspored sustava unutar fizičkog podatkovnog centra, uključujući informacije o virtualnim poslužiteljima, bazama podataka, mrežnim vezama i sklopovskim resursima; prikaz rasporeda komponenata na konkretnim poslužiteljima u oblaku (npr. AWS, Google Cloud).



Implementacijski dijagram razmještaja prikazuje konkretan fizički i virtualni raspored sustava u produkcijskom okruženju. Korisnik pristupa aplikaciji putem web preglednika na vlastitom računalu, a komunikacija se odvija preko internetske mreže prema backend aplikaciji smještenoj na servisu Azure Web App Service. Backend aplikacija komunicira sa Supabase cloud platformom, koja osigurava PostgreSQL bazu podataka i dodatne backend servise.

Ovo poglavlje treba opisati 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.

Ispitivanje komponenti

Cilj ispitivanja komponenti je provjera osnovnih funkcionalnosti implementiranih u razredima sustava. Ovdje je potrebno izolirati svaku komponentu kako bi se testirala njezina ispravnost i reakcija na različite scenarije.

Test Case 1: AUTH-TC-001 – Valjana prijava korisnika (Regular Case)

Test ID & Name

AUTH-TC-001: Valjana prijava korisnika vraća token i korisničke podatke

Test Type

Regular Case

Description

Provjerava ispravno hashiranje lozinke pomoću bcrypta i uspješnu usporedbu tijekom prijave s valjanim vjerodajnicama.

Input Data

- **Lozinka (plain text):** `password123`
- **Generirani hash:** `bcrypt.hash(password123, 10)`
- **Lozinka za usporedbu:** `password123`

Expected Output

- **bcrypt usporedba:** `true`
- **Prijava:** Dopuštena
- **Greška:** Nema

Actual Result

PASS

Procedure (AAA)

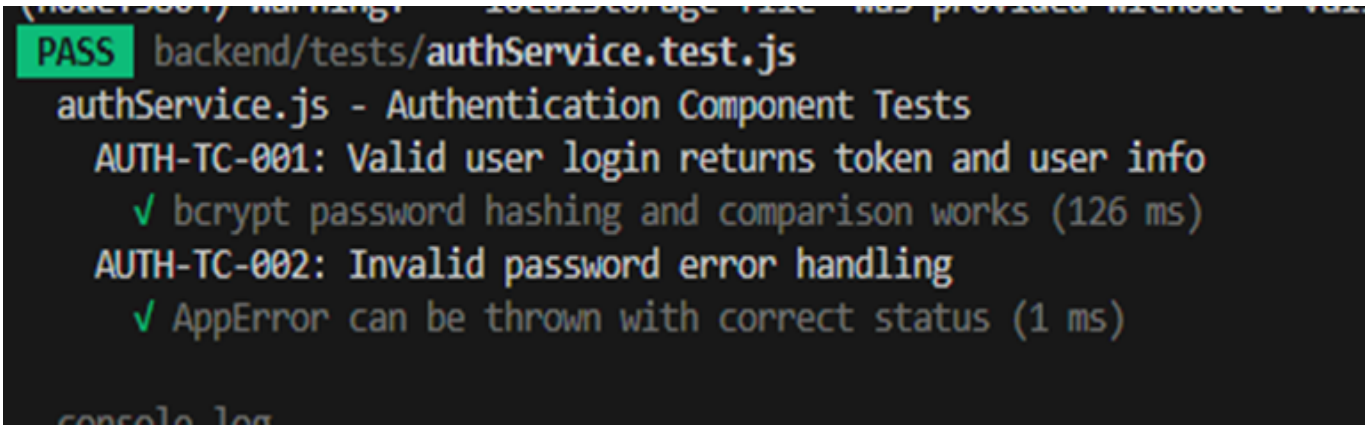
1. **Arrange:** Hashirati lozinku pomoću bcrypta
2. **Act:** Usporediti plain lozinku s hashom
3. **Assert:** Očekivati rezultat `true`

Code Location

- `backend/tests/authService.test.js` (L10–24)

- Test: `bcrypt password hashing and comparison works`

```
// AUTH-TC-001: Valid user login (Regular Case)
describe('AUTH-TC-001: Valid user login returns token and user info', () => {
  test('bcrypt password hashing and comparison works', async () => {
    const plainPassword = 'password123';
    const hashedPassword = await bcrypt.hash(plainPassword, 10);
    const isMatch = await bcrypt.compare(plainPassword, hashedPassword);
    expect(isMatch).toBe(true);
  });
});
```



Test Case 2: AUTH-TC-002 – Neispravna lozinka (Exception Case)

Test ID & Name

AUTH-TC-002: Obrada greške za neispravnu lozinku

Test Type

Exception Case

Description

Osigurava da se neispravna lozinka odbije i vrati odgovarajući AppError (401).

Input Data

- **Pogrešna lozinka:** `wrongpassword`
- **Ispravan hash:** `bcrypt.hash('password123', 10)`
- **Poruka greške:** `Neispravni podaci`
- **HTTP status:** `401`

Expected Output

- **bcrypt usporedba:** `false`
- **AppError:** Bačen (401)
- **Prijava:** Odbijena

Actual Result

PASS

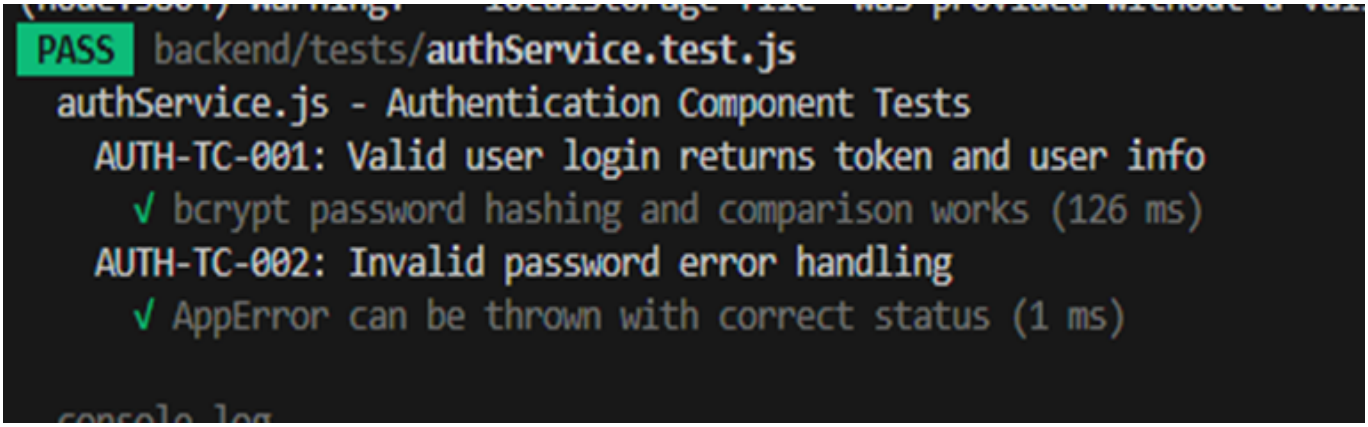
Procedure (AAA)

1. **Arrange:** Hashirati ispravnu lozinku
2. **Act:** Usporediti pogrešnu lozinku s hashom
3. **Assert:** Rezultat `false` i AppError (401)

Code Location

- `backend/tests/authService.test.js` (L42–50)
- Test: `AppError can be thrown with correct status`

```
// AUTH-TC-002: Invalid password handling (Exception Case)
describe('AUTH-TC-002: Invalid password error handling', () => {
  test('AppError can be thrown with correct status', () => {
    const error = new AppError('Neispravni podaci', 401);
    expect(error).toBeInstanceOf(AppError);
    expect(error.status).toBe(401);
    expect(error.message).toBe('Neispravni podaci');
  });
});
```



Test Case 3: DISC-TC-003 – Vlasnik ima pristup privatnoj diskusiji (Regular Case)

Test ID & Name

DISC-TC-003: Vlasnik može pristupiti vlastitoj privatnoj diskusiji

Test Type

Regular Case

Description

Provjerava da vlasnik diskusije može pristupiti diskusiji označenoj kao privatna.

Input Data

- **User ID:** `user-456`
- **Diskusija:** `{ visibility: 'privatna', owner_id: 'user-456' }`

Expected Output

- **Pristup:** `true`

Actual Result

PASS

Procedure (AAA)

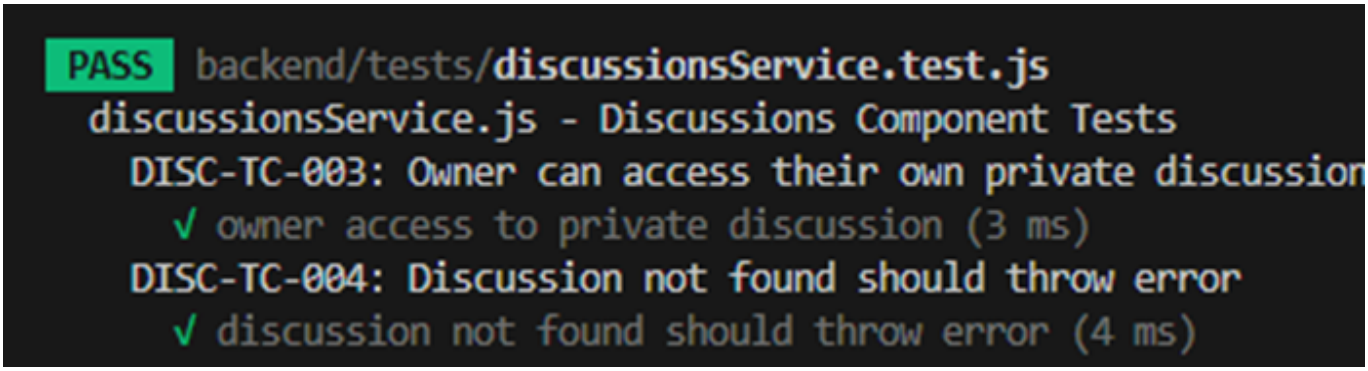
1. **Arrange:** Kreirati privatnu diskusiju s odgovarajućim vlasnikom
2. **Act:** Usporediti `owner_id === userId`
3. **Assert:** Očekivati `true`

Code Location

- `backend/tests/discussionsService.test.js` (L89–106)
- Test: `owner access to private discussion`

```
describe('discussionsService.js - Discussions Component Tests', () => {

  // DISC-TC-003: Owner access to private discussion (Regular Case)
  describe('DISC-TC-003: Owner can access their own private discussion', () => {
    test('owner access to private discussion', () => {
      const userId = 'user-456';
      const discussion = {
        id: 4,
        title: 'My Private Discussion',
        visibility: 'privatna',
        owner_id: userId
      };
      const canAccess = discussion.owner_id === userId;
      expect(canAccess).toBe(true);
    });
  });
```



Test Case 4: DISC-TC-004 – Diskusija nije pronađena (Exception Case)

Test ID & Name

DISC-TC-004: Nepostojeća diskusija baca grešku

Test Type

Exception Case

Description

Provjerava da se za nepostojeću diskusiju baca AppError sa statusom 404.

Input Data

- **Rezultat upita:** `null`
- **Poruka greške:** `Diskusija nije pronađena`
- **HTTP status:** `404`

Expected Output

- **AppError:** Bačen (404)

Actual Result

PASS

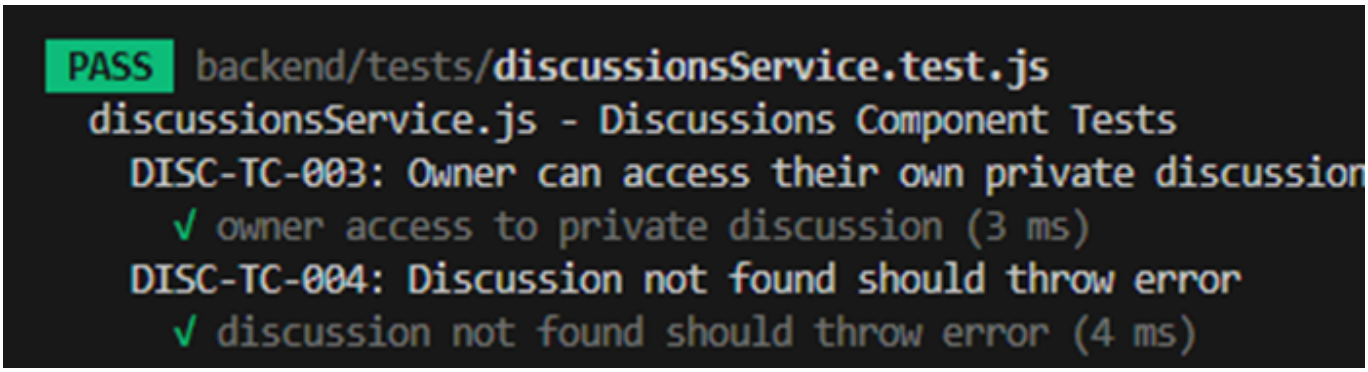
Procedure

1. **Arrange:** Simulirati `data = null`
2. **Act & Assert:** Očekivati AppError (404)

Code Location

- `backend/tests/discussionsService.test.js` (L195–206)
- Test: `discussion not found should throw error`

```
22    // DISC-TC-004: Discussion not found (Exception Case)
23    describe('DISC-TC-004: Discussion not found should throw error', () => {
24      test('discussion not found should throw error', () => {
25        const data = null;
26        expect(() => {
27          if (!data) throw new AppError('Diskusija nije pronađena', 404);
28        }).toThrow(AppError);
29      });
30    });
31  });
32
```



Test Case 5: POLL-TC-001 – Valjano kreiranje ankete (Regular Case)

Test ID & Name

POLL-TC-001: Vlasnik diskusije može kreirati valjanu anketu

Test Type

Regular Case

Description

Provjerava uspješno kreiranje ankete s ne-praznim pitanjem.

Input Data

- **Pitanje:** `Što mislite o novom izgledu?`
- **Anketa:** `{ author_id, question, closed: false }`

Expected Output

- **Validacija:** Uspješna
- **Anketa kreirana:** Da

Actual Result

PASS

Sub-tests

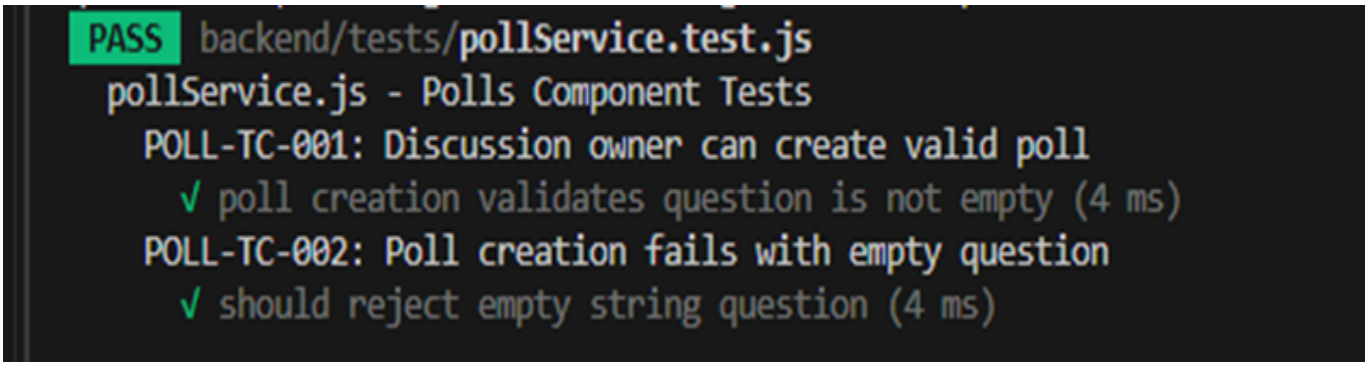
1. Validacija pitanja (`trim().length > 0`)
2. Objekt ankete sadrži `author_id`, `question`, `closed = false`

Code Location

- `backend/tests/pollService.test.js` (L12–35)

```
describe('pollService.js - Polls Component Tests', () => {

  // POLL-TC-001: Valid poll creation (Regular Case)
  describe('POLL-TC-001: Discussion owner can create valid poll', () => {
    test('poll creation validates question is not empty', () => {
      const question = 'Što mislite o novom izgledu?';
      const isValid = question && question.trim().length > 0;
      expect(isValid).toBe(true);
    });
```



Test Case 6: NON-TC-005 – Poziv nepostojeće funkcije (Negative Case)

Test ID & Name

NON-TC-005: Poziv nepostojeće funkcije baca TypeError

Test Type

Negative Case

Description

Provjerava da poziv nepostojeće funkcije rezultira TypeError greškom.

Input Data

- `discussionsService.nonExistentFunction()`

Expected Output

- **TypeError:** Bačen

Actual Result

PASS

Code Location

- backend/tests/nonexistentFunction.test.js (L1–20)

```
describe('NON-TC-005: Non-existent functionality handling', () => {
  test('calling unimplemented function throws TypeError', () => {
    expect(() => {

      discussionsService.nonExistentFunction();
    }).toThrow(TypeError);
  });
});
```

```
PASS backend/tests/nonexistentFunction.test.js
  NON-TC-005: Non-existent functionality handling
    √ calling unimplemented function throws TypeError (6 ms)
```

Test Case 7: POLL-TC-002 – Neuspješno kreiranje ankete s praznim pitanjem (Edge Case)

Test ID & Name

POLL-TC-002: Kreiranje ankete ne uspijeva s praznim pitanjem

Test Type

Edge Case

Description

Provjerava odbijanje praznih i whitespace-only pitanja uz grešku 400.

Input Data

- `''`
- `' '`

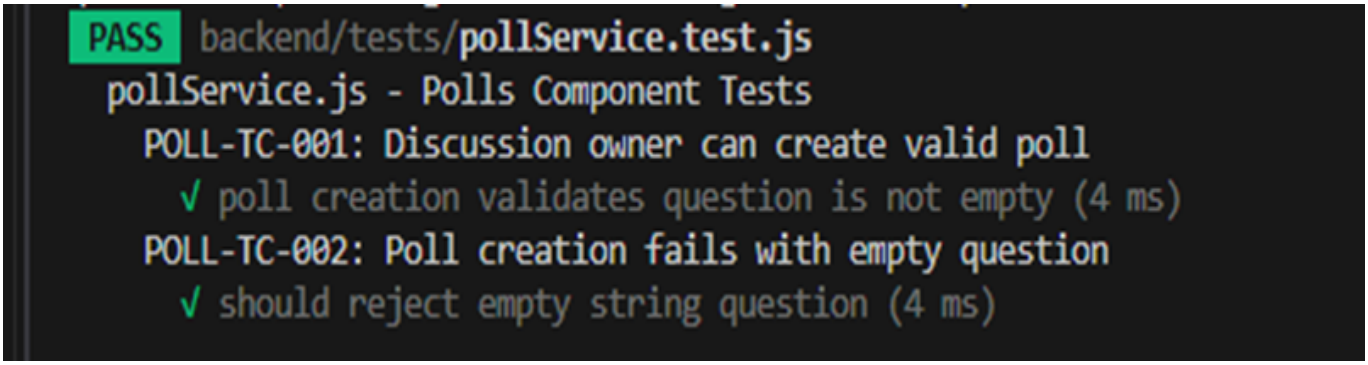
Expected Output

- **AppError:** Bačen
- **Poruka:** `Pitanje je obavezno`
- **Status:** `400`

Actual Result

```
// POLL-TC-002: Empty question validation (Edge Case)
describe('POLL-TC-002: Poll creation fails with empty question', () => {
  test('should reject empty string question', () => {
    const emptyQuestion = '';
    expect(() => {
      if (!emptyQuestion || !emptyQuestion.trim()) {
        throw new AppError('Pitanje je obavezno', 400);
      }
    }).toThrow(AppError);
  });
});
```

PASS



Validation Logic

if (!question || !question.trim()) → throw AppError(40

Code Location

- backend/tests/pollService.test.js

Summary Table

Test ID	Servis	Tip	Funkcionalnost	Status
AUTH-TC-001	authService	Regular	Provjera lozinke	PASS
AUTH-TC-002	authService	Exception	Neispravna lozinka	PASS
DISC-TC-003	discussionsService	Regular	Pristup vlasnika	PASS
DISC-TC-004	discussionsService	Exception	Nije pronađeno	PASS
POLL-TC-001	pollService	Regular	Valjana anketa	PASS
POLL-TC-002	pollService	Edge	Prazno pitanje	PASS

Ispitivanje sustava i komponenti — Dokumentacija testova

Alati

- **Selenium WebDriver (Python)**
- **Pytest + pytest-html**
- **webdriver-manager**

Artefakti:

- HTML izvještaj: `reports/report.html`
- Screenshots: `reports/screenshots/`
- Logovi: `reports/logs/pytest.log`

Broj otkrivenih grešaka: **4/15 failing tests**

Ispitivanje sustava — 4 test case-a

SYS-TC-001 — Valjana prijava korisnika

Type: Regular Case
Input Data:

- Email: `user@fer.unizg.hr`
- Lozinka: `password123`

Steps:

1. Otvoriti aplikaciju `/login`
2. Popuniti inpute
3. Kliknuti `submit`
4. Provjeriti preusmjerenje i prisutnost elementa početne stranice

Expected Output: Prijava uspješna, početna stranica prikazana
Actual Output: **PASS**

```python
def test_TC_LOGIN_001_valid_credentials(self, driver, screenshot_dir):
    login_page = LoginPage(driver)
    login_page.open()

    assert login_page.is_login_button_visible(), "Login gumb nije vidljiv"
    driver.save_screenshot(f"{screenshot_dir}/TC_LOGIN_001_start.png")

    login_page.login("user@fer.ugnz.hr", "password123")

    time.sleep(3)

    form_submitted = driver.execute_script("""
        return document.querySelector('input[type="email"]')?.value === 'user@fer.ugnz.hr' &&
               document.querySelector('input[type="password"]')?.value === 'password123';
    """)
    assert form_submitted, "Forma nije ispravno popunjena"

    driver.save_screenshot(f"{screenshot_dir}/TC_LOGIN_001_success.png")

    current_url = driver.current_url
    assert "login" not in current_url.lower(), "Korisnik nije preusmjeren sa login stranice"
```

```
backend/tests/selenium/test_cases/test_login.py::TestLogin::test_TC_LOGIN_001_valid_credentials
------------------------------------------------------------------------------ live log setup -------------------------------------------------------------------------------
INFO     WDM:logger.py:11 ====== WebDriver manager ======
INFO     WDM:logger.py:11 Get LATEST chromedriver version for google-chrome
INFO     WDM:logger.py:11 Get LATEST chromedriver version for google-chrome
INFO     WDM:logger.py:11 Driver [C:\Users\sasha\.wdm\drivers\chromedriver\win64\143.0.7499.192\chromedriver-win32/chromedriver.exe] found in cache
PASSED                                                                                                                                                          [100%]

============================================================================== 1 passed, 1 warning in 8.33s ==============================================================================
```

SYS-TC-002 — Nevažeća lozinka (Rubni uvjet)

Type: Edge Case
Input Data:

- Email: `user@fer.unizg.hr`
- Lozinka: `malimedo`

Steps:

1. Otvoriti `/login`
2. Unijeti podatke
3. Kliknuti `submit`
4. Provjeriti prikaz poruke o grešci

Expected Output: "Pogrešno korisničko ime ili lozinka."
Actual Output: **PASS** — front-end prikazuje grešku

```python
def test_TC_LOGIN_002_invalid_password(self, driver, screenshot_dir):
    login_page = LoginPage(driver)
    login_page.open()

    # Unesi krive podatke
    login_page.login("user@fer.ugnz.hr", "malimedo")

    # Čekaj da se error prikaže
    time.sleep(2)
    driver.save_screenshot(f"{screenshot_dir}/TC_LOGIN_002_error.png")

    error_message = login_page.get_error_message()

    if error_message:
        assert (
            "pogrešno" in error_message.lower()
            or "grešk" in error_message.lower()
        ), f"Neočekivana poruka: {error_message}"
    else:
        # Ako nema poruke, korisnik mora ostati na login stranici
        assert "login" in driver.current_url.lower(), \
            "Korisnik nije na login stranici"
```

```
backend/tests/selenium/test_cases/test_login.py::TestLogin::test_TC_LOGIN_002_invalid_password
---------------------------------------------------------------------------------- live log setup ----------------------------------------------------------------------------------
INFO     WDM:logger.py:11 ====== WebDriver manager ======
INFO     WDM:logger.py:11 Get LATEST chromedriver version for google-chrome
INFO     WDM:logger.py:11 Get LATEST chromedriver version for google-chrome
INFO     WDM:logger.py:11 Driver [C:\Users\sasha\.wdm\drivers\chromedriver\win64\143.0.7499.192\chromedriver-win32/chromedriver.exe] found in cache
PASSED                                                                                                                                                          [100%]

============================================================================== 1 passed, 1 warning in 16.78s ==============================================================================
PS F:\cod\progi\progiProjekt>
```

NONEX-TC-001 — Poziv nepostojeće funkcionalnosti

Type: Nonexistent functionality
Input Data:

- URL: `/nonexistent-route-automated-test-xyz`

Steps:

1. Otvoriti URL
2. Pričekati učitavanje stranice
3. Provjeriti prisutnost 404 poruke ili friendly error

4. Ako nedostaje, snimiti screenshot

Expected Output: 404 ili friendly error
Actual Output: **FAIL** — SPA vraća root (`index.html`)

```python
class TestNonexistentFunctionality:


    def test_TC_NONEXIST_001_nonexistent_route(self, driver, screenshot_dir):
        url = "http://localhost:5173/nonexistent-route-automated-test-xyz"
        driver.get(url)

        # wait for page load
        WebDriverWait(driver, 5).until(
            lambda d: d.execute_script("return document.readyState") == "complete"
        )

        page_source = driver.page_source.lower()

        keywords = [
            "404",
            "not found",
            "stranica nije pronađena",
            "page not found",
            "notfound",
        ]

        found_404 = any(k in page_source for k in keywords)

        if not found_404:
            driver.save_screenshot(f"{screenshot_dir}/TC_NONEXIST_001_failed.png")

        assert found_404, (
            "Non-existent route did not show expected 404 or friendly error page"
        )
```

```
DEBUG    selenium.webdriver.remote.remote_connection:remote_connection.py:438 Remote response: status=200 | data={"value":null} | headers=HTTPHeaderDict({'Content-Length': '14', 'Content-Type
: 'application/json; charset=utf-8', 'cache-control': 'no-cache'})
DEBUG    selenium.webdriver.remote.remote_connection:remote_connection.py:467 Finished Request
=========================== short test summary info ============================
FAILED backend/tests/selenium/test_cases/test_nonexistent_functionality.py::TestNonexistentFunctionality::test_TC_NONEXIST_001_nonexistent_route - Failed: Non-existent route did not show expec
ted 404/friendly error. Screenshot saved.
============================== 1 failed, 1 warning in 4.75s ==============================
```

DISC-TC-004 — Rubni uvjet: maksimalna duljina inputa

Type: Boundary Case
Input Data:

- Polje `title` — string od 10.000 znakova

Steps:

1. Otvoriti formu za novi unos
2. Unijeti dugački string
3. Pokušati poslati formu
4. Promatrati odgovor (client/server validacija)

Expected Output: Sustav prihvaća ograničenu duljinu ili vraća valjanu grešku; nema 500 error
Actual Output: **PASS**

```python
def test_TC_DISC_005_boundary_max_input_length(self, driver, screenshot_dir):
    try:
        login_page = LoginPage(driver)
        login_page.open()
        login_page.login("user@fer.ugnz.hr", "password123")

        time.sleep(2)

        driver.get("http://localhost:5173/discussions")
        time.sleep(2)

        try:
            new_discussion_button = driver.find_element(By.XPATH,
                "//button[contains(text(), 'Nova rasprava')] | //button[contains(text(), 'New Discussion')]")
            new_discussion_button.click()
            time.sleep(2)

            driver.save_screenshot(f"{screenshot_dir}/TC_DISC_005_form_start.png")
            title_input = driver.find_element(By.XPATH, "//input[@placeholder*='naslov' or @placeholder*='title']")
            long_title = "x" * 10000
            title_input.clear()
            title_input.send_keys(long_title)
            content_input = driver.find_element(By.XPATH, "//textarea[@placeholder*='tekst' or @placeholder*='content']")
            content_input.clear()
            content_input.send_keys("Testna rasprava sa dugim naslovom")

            driver.save_screenshot(f"{screenshot_dir}/TC_DISC_005_input_filled.png")

            try:
                submit_button = driver.find_element(By.XPATH, "//button[contains(text(), 'Objavi')] | //button[contains(text(), 'Post')]")
                submit_button.click()

                time.sleep(3)
                driver.save_screenshot(f"{screenshot_dir}/TC_DISC_005_submit.png")
                current_url = driver.current_url
                page_source = driver.page_source

                is_error_500 = "500" in page_source or "Internal Server Error" in page_source

                if is_error_500:
                    pytest.fail("Greška: HTTP 500 server error - sustav ne tolerira dugačke inpute")

                validation_error = "predugačak" in page_source.lower() or "too long" in page_source.lower()

            except Exception as e:
                driver.save_screenshot(f"{screenshot_dir}/TC_DISC_005_no_submit.png")

        except Exception as e:
            driver.save_screenshot(f"{screenshot_dir}/TC_DISC_005_no_form.png")

    except Exception as e:
        driver.save_screenshot(f"{screenshot_dir}/TC_DISC_005_error.png")
        pytest.fail(f"Test failed: {str(e)}")
```

```
backend/tests/selenium/test_cases/test_discussions.py::TestDiscussions::test_TC_DISC_005_boundary_max_input_length
------------------------------ live log setup -------------------------------
INFO     WDM:logger.py:11 ====== WebDriver manager ======
INFO     WDM:logger.py:11 Get LATEST chromedriver version for google-chrome
INFO     WDM:logger.py:11 Get LATEST chromedriver version for google-chrome
INFO     WDM:logger.py:11 Driver [C:\Users\sasha\.wdm\drivers\chromedriver\win64\143.0.7499.192\chromedriver-win32/chromedriver.exe] found in cache
PASSED                                                                   [100%]

========================= 1 passed, 1 warning in 19.05s =========================
PS F:\cod\progi\progiProjekt>
```

Korištene tehnologije i alati

Jasno i precizno opisane tehnologije korištene u projektu kako bi se olakšalo održavanje, proširenje i suradnja u timu.

Programski jezici

U projektu je korišten JavaScript (ECMAScript 2022) kao primarni programski jezik za razvoj klijentskog i poslužiteljskog dijela aplikacije. JavaScript je odabran zbog široke podrške, jednostavne integracije s web tehnologijama te mogućnosti korištenja istog jezika na frontendu i backendu, čime se smanjuje složenost održavanja sustava.

Radni okviri i biblioteke

Za razvoj korisničkog sučelja korištena je React biblioteka (verzija 18) uz alat Vite za razvojno okruženje i izgradnju produkcijske verzije aplikacije. React omogućuje izradu modularnih i ponovno iskoristivih komponenti te učinkovito upravljanje stanjem aplikacije, dok Vite omogućuje brzo pokretanje razvojnog poslužitelja i optimizirano pakiranje aplikacije.

Poslužiteljski dio aplikacije razvijen je korištenjem Node.js platforme (verzija 25.1.0) i Express frameworka(verzija 5.1.0), koji omogućuju jednostavnu implementaciju REST API-ja i modularnu organizaciju ruta. Za autentifikaciju i autorizaciju koriste se Passport (verzija 0.7.0) i passport-google-oauth20 (verzija 2.0.0), dok se za upravljanje autentifikacijskim tokenima koristi JSON Web Token – JWT (verzija 9.0.2). Sigurnost korisničkih lozinki osigurana je pomoću bcryptjs (verzija 3.0.2), a upravljanje konfiguracijskim varijablama pomoću dotenv (verzija 17.2.3). Evidencija HTTP zahtjeva omogućena je pomoću Morgan (verzija 1.10.1),a podrška za komunikaciju s različitih izvora osigurana je bibliotekom CORS (verzija 2.8.5).

Za rad s bazom podataka korištene su biblioteke pg (verzija 8.16.3) i postgres (verzija 3.4.7), dok je za dodatne backend servise integrirana platforma Supabase putem biblioteke @supabase/supabase-js (verzija 2.78.0).

Baza podataka

Korištena je relacijska baza podataka PostgreSQL, koja omogućuje pouzdanu pohranu podataka i podršku za složenije upite. PostgreSQL je odabran zbog stabilnosti, dobre skalabilnosti i široke podrške unutar Node.js ekosustava.

Razvojni alati

Za razvoj aplikacije korišteno je razvojno okruženje Visual Studio Code, a za upravljanje izvornim kodom korišten je sustav za kontrolu verzija Git uz repozitorij na platformi GitHub. Tijekom razvoja korišten je alat Nodemon (verzija 3.1.10) za automatsko ponovno pokretanje poslužitelja prilikom promjena u kodu.

Alati za ispitivanje

Za UI testiranje korišteni su Selenium 4.0 i Python 3.12 s pytest okvirom, s automatski generiranim HTML izvještajima, logovima i snimkama zaslona. Za backend unit testiranje koristi se Jest 30.2 framework s Node.js-om.

Alati za razmještaj

Za automatizirani razmještaj aplikacije koristi se GitHub Actions CI/CD sustav u kombinaciji s akcijom azure/webapps-deploy (verzija 3). Svaka promjena koja se pošalje na glavnu granu repozitorija automatski pokreće postupak izgradnje i razmještaja aplikacije. Build proces se izvršava u Linux okruženju (Ubuntu runner) s instaliranom verzijom Node.js 22.x, nakon čega se aplikacija automatski prenosi na produkcijsko okruženje.

Cloud platforma

Aplikacija je razmještena na platformi Microsoft Azure, koristeći uslugu Azure Web App (App Service). Ovakav način razmještaja omogućuje visoku dostupnost aplikacije, jednostavno skaliranje i automatizirano upravljanje infrastrukturom bez potrebe za ručnim održavanjem poslužitelja. Aplikacija se razmještava kao upravljeni servis (Platform as a Service), bez korištenja vlastitih virtualnih strojeva ili Docker kontejnera, čime se dodatno pojednostavljuje administracija i održavanje sustava.

1. Instalacija

- **Preduvjeti**: Node.js 25.1.0
- **Preuzimanje**:
  - `git clone https://github.com/stanari123/progiProjekt.git`
  - `cd progiProjekt`
- **Instalacija ovisnosti**:
  - backend
    - `npm install`
  - frontend
    - `cd front_react`
    - `npm install`

2. Postavke

- **Konfiguracijske datoteke**:
  - frontend .env
    - `VITE_API_BASE=http://localhost:3001/api`
  - backend .env
    - `SUPABASE_URL=https://your-project.supabase.co`
    - `SUPABASE_KEY=your-supabase-anon-key`
    - `GOOGLE_CLIENT_ID=your-google-client-id.apps.googleusercontent.com`
    - `GOOGLE_CLIENT_SECRET=your-google-client-secret`
    - `GOOGLE_CALLBACK_URL=http://localhost:3001/auth/google/callback`
    - `REFRESH_TOKEN=your-google-refresh-token`
    - `REDIRECT_URL=https://developers.google.com/oauthplayground`
    - `JWT_SECRET=your-secure-jwt-secret-key`
    - `STANPLAN_API_KEY=your-stanplan-api-key`
    - `STANPLAN_LINK_ROW_ID=row-id-for-stanplan`
    - `PORT=3001`
    - `WEBSITE_URL=your-website-url`
- **Postavke baze podataka**:
  - prijavite se na Supabase
  - stvorite novi projekt
  - pomoću "Table editor", stvorite sve tablice i definirajte ih kao ovdje
  - sada možete uzeti API ključeve na glavnoj stranici Supabase projekta ako kliknete na "Connect" na vrhu stranice

3. Pokretanje aplikacije

- **Razvojno okruženje**:
  - backend server

        - `npm run dev`
    - frontend server
        - `cd .\front_react\`
        - `npm run dev`

- **Produkcijsko okruženje**:
    - sve oko produkcije provodi Azure. Azure pokreće `npm run build` i pokreće backend server
- **Provjera rada**:
    - http://localhost:5173/

4. Upute za administratore

- **Pristup administratorskom sučelju**:
    - ako se prijavite računom koji ima administratorske privilegije, bit ćete preusmjereni na `/admin` stranicu gdje se mogu stvarati novi korisnici, vidjeti diskusije u zgradama i upravljati sa StanPlan API
- **Redovito održavanje**:
    - baza podataka će biti aktivna dok god ima interakcije s njom unutar tjedan dana
        - ako tjedan dana ne dolaze zahtjevi na bazu podataka, Supabase će ju deaktivirati te obavijestiti upravitelja baze koji ju može opet aktivirati
- pregled logova
    - unutar razvojnog okruženja, logovi će biti prikazani unutar terminala gdje je pokrenut backend server
    - u produkcijskom okruženju bit će prikazani na Azure pod "Log stream"
    - ažuriranje aplikacije se provodi preuzimanjem nove verzije s Github-a i ponovnim pokretanjem frontend i backend servera
        - `git pull origin main`
        - `npm install` - potrebno ako su dodane nove ovisnosti
        - `npm run dev` kao što je objašnjeno u točki "3. Pokretanje aplikacije"

> U produkcijskom okruženju, svaki commit na main branch automatski se ažurira na stranici. Azure gleda main branch i sve sam ažurira.

- **Rješavanje problema**: Servis Microsoft Azure ima "Log stream" koji prikazuje svaki GET i POST zahtjev kao i error koji se pojavi. Putem toga, lako je otkriti u čemu je greška. U razvojnom okruženju, svi zahtjevi prikazani su u terminalu, a Supabase ima svoj log stream gdje prikazuje krive zahtjeve i greške.

---

5. Postavljanje na Microsoft Azure

Aplikacija je napravljena i postavljena pomoću Microsoft Azure servisa. Microsoft Azure je jednostavna i pristupačna platforma za web-aplikacije.

- prijavite se na portal Microsoft Azure.
    - odaberite "Create a resource" te onda "Web App".
        - Basics
            - pod "Instance Details", navedite ime stranice i odaberite "Runtime stack" te postavite "Node 22 LTS"
                - kao regiju, izaberite štogod je najbliže Vama
                    - neke regije ne podržavaju sve planove pa je moguće da treba uzeti neku drugu regiju
            - "Pricing plan" izaberite koji god
                - besplatni plan je "Free F1"
        - Deployment
            - omogućite "Continuous deployment settings" kako bi se Azure automatski ažurirao kada se nešto novo postavi na main grani
            - pod "Github settings", ulogirajte se u svoj Github račun i omogućite ovlasti Azure-u
                - odaberite svoju organizaciju
                    - omogućite "Azure App Service" u organizaciji
                - odaberite repozitorij u kojem se nalazi aplikacija
                - za branch stavite "main"
        - Review + Create
            - odaberite "Create" kako bi Azure stvorio aplikaciju
    - unutar konzole za upravljanje aplikacijom preko Microsoft Azure-a
        - odaberite "Settings" i "Environment variables"
            - postavite sve varijable okruženja kao i .env datotekama u razvojnom okruženju
- Azure će stvoriti aplikaciju i URL će biti prikazan na "Overview" na Azure-u pod "Default domain"

Opis pristupa aplikaciji na javnom poslužitelju

Koristite URL https://progistanblog.azurewebsites.net za pristup aplikaciji.

Kako bi pristupili administratorskom sučelju, trebate se prijaviti s administratorskim računom nakon čega ćete biti preusmjereni na sučelje.

Ograničenje: Administrator mora stvoriti račun s Vašim e-mailom kako biste se mogli prijaviti. Kada administrator stvori Vaš račun, dobit ćete email s lozinkom. Tada ćete se moći prijaviti i putem Google-a.

Popis neimplementiranih funkcionalnosti:

-inicijatorova mogućnost na defniranje maksimalnog broja poruka za svakog suvlasnika unutar diskusije

Osvrt na projekt i zaključak

Tijekom izrade projektnog zadatka pokazalo se da je najveći izazov bio koordinirani rad unutar tima, posebno u dijelu podjele posla i praćenja međusobnih ovisnosti između zadataka. Kako je projekt napredovao, kodna baza postajala je sve složenija, što je otežavalo snalaženje i zahtijevalo sve intenzivniju komunikaciju. Ovaj problem donekle je uklonjen redovitim Teams sastancima na kojima su se dogovarali sljedeći koraci i rješavale nejasnoće, no iskustvo je pokazalo da bi detaljnije planiranje na samom početku, jasnije definirani rokovi i preciznije razrađene faze razvoja značajno ublažili kasnija usporavanja. Dodatni izazov predstavljalo je učenje novih tehnologija potrebnih za pojedine dijelove sustava, što je usporilo početne faze, ali je dugoročno doprinijelo boljem razumijevanju arhitekture i većoj samostalnosti članova tima.

Tijekom projekta stečena su znanja iz područja web tehnologija, integracije vanjskih servisa, autentifikacije, organizacije timskog rada i izrade tehničke dokumentacije. Perspektive za nastavak rada na projektu uključuju proširenje funkcionalnosti i poboljšanje korisničkog iskustva, primjerice uvođenjem naprednijih opcija glasanja, sustava obavijesti, dodatnih administracijskih mogućnosti, integracija s drugim sustavima te raznih manjih poboljšanja sučelja radi intuitivnijeg korištenja. Što se tiće budućeg rada u timu, iskustvo ovog projekta pokazalo je da bi pomnije početno planiranje ,u vidu boljeg defniranja plana izrade pojedinih ključnih dijelova projekta, zajedno sa dogovaranjem nekih rokovima izrade i sastancima za provjeru izvršenja unutar istih, uvelike olakšalo i optimiziralo izradu projekta u cjelosti.

Zaključno, iskustvo rada na projektu pokazalo je koliko su dobra organizacija, jasna komunikacija i pravovremeno planiranje ključni za uspješnu realizaciju složenijih zadataka. Unatoč izazovima, tim je uspješno savladao tehničke i organizacijske prepreke te stekao vrijedna znanja koja će biti korisna u budućim projektima. Naučene lekcije o koordinaciji, podjeli odgovornosti i važnosti ranog definiranja smjernica predstavljaju čvrstu osnovu za učinkovitiji rad u sljedećim fazama razvoja.

Rev.	Opis promjene/dodatka	Autori	Datum
0.1	Otvorena organizacija i stvoren Wiki	Jakov Svalina	14.10.2025.
0.2	Uređen README.md	Jakov Svalina	18.10.2025.
0.3	Dodan opis projektnog zadatka	Jakov Svalina	21.10.2025.
0.4	Odrađena analiza zahtjeva	Jakov Svalina	21.10.2025.
0.5	Nadopunjen dnevnik sastanaka	Jakov Svalina	14.11.2025.
0.6	Objašnjena arhitektura i dizajn	Luka Zorić	14.11.2025.
0.7	Dodani dijagrami razreda,mvc arhitektura, sklopoprogramski zahtjevi	Marko Vrkić	14.11.2025.
0.8	Napravljena analiza zahtjeva i detaljan opis projektnog zadatka	Vinko Šapina	14.11.2025.
0.8	Napravljena Specifikacija zahtjeva sustava i dopunjena Arhitektura i dizajn sustava	Oleksandr Malik	14.11.2025.
1.0	Tehnologije za implementaciju aplikacije(izuzev testiranja), zaključak i budući rad	Marko Vrkić	19.01.2026.
1.1	Ažurirani dijagrami frontend,backend i cjelokupne arhitekture susutava,nadodani dijagrami stanja i aktivnosti	Marko Vrkić	20.01.2026.
1.2	Ispravljene nepravilnosti prve revizije projekta	Marko Vrkić	22.01.2026.
1.3	Napravljena dokumentacija za ispitivanje programskog rješenja	Oleksandr Malik	22.01.2026.
1.4	Ažuriran pregled aktivnosti grupe i količine uloženog rada	Vinko Šapina	23.01.2026.
1.5	Dodane upute za puštanje u pogon	Jakov Svalina	23.1.2026.

Kontinuirano osvježavanje Popisati sve reference i literaturu koja je pomogla pri ostvarivanju projekta.

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

Dnevnik sastajanja

1. sastanak

    - datum: 18. listopada 2025.
    - prisustvovali: M.Vrkić, L.Zorić, O.Malik, V.Šapina, I.Trobradović, J.Svalina
    - teme sastanka:
        - odabir projektnog zadatka - prihvaćanje StanBlog zadatka
        - uspostavljanje WhatsApp grupe svih članova tima
        - dogovaranje uloga u razvoju projekta

2. sastanak

    - datum: 22. listopada 2025.
    - prisustvovali: M.Vrkić, L.Zorić, O.Malik, V.Šapina, I.Trobradović, J.Svalina
    - teme sastanka:
        - odabir konkretnih tehnologija koje će se koristiti za razvoj projekta - React, NodeJS, Supabase, API-jevi
        - osmišljavanje prve inačice dizajna web-stranice
        - određivanje prioritetnih funkcionalnih zahtjeva za razvoj do prve predaje projekta

3. sastanak

    - datum: 4. studenog 2025.
    - prisustvovali: M.Vrkić, L.Zorić, O.Malik, V.Šapina, I.Trobradović, J.Svalina
    - teme sastanka:
        - uspostavljanje rokova svakom članu za isporuku svoga dijela zadatka
        - modifikacija baze podataka u svrhu lakšeg backend razvoja
    - poveznica na issue:
        - #1
        - #5

4. sastanak

    - datum: 17. prosinca 2025.
    - prisustvovali: M.Vrkić, L.Zorić, O.Malik, V.Šapina, I.Trobradović, J.Svalina
    - tema sastanka: Revizija prve predaje i raspodjela daljnjih poslova

5. sastanak

    - datum: 2. siječnja 2026.
    - prisustvovali: M.Vrkić, L.Zorić, O.Malik, V.Šapina, I.Trobradović, J.Svalina
    - teme sastanka:
        - Revizija dosadašnjeg napretka
        - Modifikacija baze podataka prema idućim zahtjevima
        - Pregled i dovršavanje tranzicije na React
        - Ostvarivanje preostalih funkcionalnosti

6. sastanak

- datum: 15. siječnja 2026.
- prisustvovali: M.Vrkić, L.Zorić, O.Malik, V.Šapina, I.Trobradović, J.Svalina
- teme sastanka:
  - Implementacija StanPlan API-a i mail servisa
  - Raspodjela poslova u vezi s dokumentacijom

Plan rada

Tjedan	Ključne aktivnosti	Angažirani članovi tima
3.	raspodjela dužnosti, odabir projektnog zadatka	MV, LZ, OM, VŠ, IT, JS
4.	razrada plana razvoja web-aplikacije, dogovor oko implementiranja željenih usluga i dodatnih funkcionalnosti te odabir razvojnih tehnologija	MV, LZ, OM, VŠ, IT, JS
5.	razvoj baze podataka, razvoj prototipa projekta koristeći privremenu lokalnu bazu	VŠ, IT
6.	povezivanje aplikacije sa Supabase bazom i refactoring kako bi aplikacija radila s njom	JS, IT
7.	deployment	JS
8.	Raspodjela aktivnosti, revizija dosadašnjeg napretka	MV, LZ, OM, VŠ, IT, JS
9.	Prijelaz na React	LZ, MV
10.	Integracija Google Maps API-a i dovršavanje ostalih funkcionalnosti	JS, IT
11.	Integracija mail API-a i poboljšanje deployment sustava	JS
12.	Ostvarivanje komunikacije sa StanPlan serverom	VŠ
13.	Dokumentacija, završni radovi, ispravljanje nepravilnosti	JS, IT, VŠ, LZ, MV, OM

Tablica aktivnosti

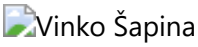
Kontinuirano osvježavanje

Zadatak	Luka Zorić	Marko Vrkić	Jakov Svalina	Isa Trobradović	Vinko Šapina	Oleksandr Malik
Upravljanje projektom			2	8	10	

Zadatak	Luka Zorić	Marko Vrkić	Jakov Svalina	Isa Trobradović	Vinko Šapina	Oleksandr Malik
Opis projektnog zadatka			1		3	
Funkcionalni zahtjevi			2		3	
Opis pojedinih obrazaca			1			11
Dijagram obrazaca			1			5
Sekvencijski dijagrami			1			7
Opis ostalih zahtjeva			1			2
Arhitektura i dizajn sustava	7	7		9		
Baza podataka					7	3
Dijagram razreda	7	6	1			
Dijagram stanja	4	4				
Dijagram komponenti						5
Korištene tehnologije i alati			4			
Ispitivanje programskog rješenja			7	12		
Dijagram razmještaja	3	5				
Upute za puštanje u pogon			3			
Puštanje u pogon			3			
Dnevnik sastajanja			1			
Izrada aplikacije	15	12	38	45	22	2
Izrada početne stranice	5	4	2	8		
Izrada baze podataka					8	3
Spajanje s bazom podataka			25	4		
Izrada programske potpore			10	18	4	
Ukupno	41	38	103	104	57	33

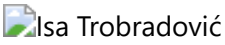
Dijagram pregleda promjena

Vinko Šapina

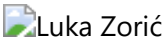


Jakov Svalina

Isa Trobradović



Luka Zorić



Marko Vrkić

Oleksandr Malik

Kjučni izazovi i rješenja

- izazovi projekta:
  - loša komunikacija i posljedični manjak koordinacije
    - rješenje: sastanci uživo i jasno podijeljeni zadatci i dobro definirani vremenski rokovi
  - korištenje nepoznatih tehnologija
    - rješenje: čitanje dokumentacije i istraživanje o uporabi vanjskih servisa poput Supabase i NodeJS
  - nedovoljno dobro definirani zahtjevi od backend tima prema timu za bazi, stoga je baza bila problematična za koristiti
    - rješenje: jasno definiranje svih atributa i parametara koje backend tim treba kako bi se mogao ostvariti prijelaz na vanjsku bazu Supabase
  - kašnjenje u razvoju
    - rješenje: sastanci na kojima se jasno definira opseg projekta koji će se implementirati u zadanom vremenskom intervalu