

# Programsko inženjerstvo ak. god. 2025./2026.

---

Sveučilište u Zagrebu

Fakultet elektrotehnike i računarstva

## Susjedi

---

**Tim: TG09.3**

**Ime: Susjedi**

**URL: <https://susjedi.onrender.com>**

**Nastavnik: Vlado Sruk**

## Uvod

---

Cilj projektnog zadatka je razvoj web-aplikacije koja služi kao podrška aktivnostima predstavnika suvlasnika u stambenim zgradama. Aplikacija omogućuje učinkovitije planiranje i koordinaciju sastanaka suvlasnika i olakšava donošenje odluka unutar zgrade. Komunikacija između upravitelja zgrade i suvlasnika, odnosno njihovog predstavnika, često se odvija putem e-maila ili oglasne ploče, uz fizičke sastanke, što je problematično iz više razloga. U procesu donošenja odluka vezanih uz zajedničke prostore nedostaje transparentnosti, teško je pratiti povijest sastanaka, mnogi suvlasnici nisu redovito obaviješteni o odlukama donesenima na tim sastancima i pratećim troškovima. Česta su kašnjenja i taj se proces dodatno komplikira porastom broja suvlasnika unutar stambene zgrade. Suvlasnici su zbog navedenoga slabo angažirani jer između ostalog nemaju osjećaj kao da njihov glas i mišljenje doprinose cijelom procesu. Također, tu je i problematika neuskladenosti sa novim zakonskim propisima koji detaljnije propisuju obaveze svih dionika. Takav je način upravljanja zajedničkim prostorima zastario, a naš je cilj to promijeniti.

Neke od stvari koje će naše rješenje omogućiti su:

- povećana uključenost suvlasnika u proces donošenja odluka
- digitalno vođenje sastanaka
- pregled povijesti sastanaka i donesenih zaključaka
- automatizirane e-mail obavijesti suvlasnicima
- jednostavno upravljanje točkama dnevnog reda
- integraciju s aplikacijom StanBlog za diskusije sa drugim stanarima

## Zašto je naš projekt koristan?

---

Naša aplikacija rješava probleme svih dionika u procesu donošenja odluka unutar stambenih zgrada. Krenimo od predstavnika suvlasnika - osobe koja je poveznica upravitelja zgradom i stanara. Uz pomoć naše aplikacije, predstavnik suvlasnika moći će jednostavno održavati i organizirati sastanke, automatski obavijestiti suvlasnike o terminima sastanaka i zaključcima, imat će pregled sudjelovanja suvlasnika u sastancima i mogućnost arhiviranja istih bez ručnog pisanja evidencije. Suvlasnici dobivaju transparentan uvid u cijeli proces. Moći će jednostavno potvrditi svoje sudjelovanje na sastanku, imat će pristup donesenim zaključcima i moći će pratiti diskusije povezane sa sastanicima pomoću integracije s aplikacijom StanBlog. Sve dosad navedeno značajno olakšava komunikaciju sa upraviteljem zgrade, čime se proces donošenja odluka unutar zgrade prilagođava novoj zakonskoj regulativi i doprinosi digitalizaciji našeg društva.

## Postojeća rješenja

---

Na tržištu već postoje određena rješenja za neke od navedenih problema. Svaka od aplikacija koje ćemo navesti pokriva jedan dio problema, ali ih ne rješava sveobuhvatno.

### Upravljanje zgradama

[Upravljanje zgradama](#) zapravo je skup aplikacija kojima je svrha modernizacija dosadašnjih načina obavljanja svih poslova vezanih uz zgrade. Sastoji se od aplikacija *eSuvlasnik* i *mUpravitelj*.

*eSuvlasnik* nudi suvlasnicima pregled osnovnih podataka o zgradi, preuzimanja dokumenata, različite vrste izvještaja, sustav obavijesti i razne druge opcije. Fokus ove aplikacije više je financijski aspekt upravljanja zgradom. Nudi jednostavno korisničko sučelje i detaljan financijski uvid u poslove zgrade.

The screenshot shows the 'Informacije o zgradbi' (Building Information) section. It displays details like address (Cankarjeva 19/11, Zagreb, 10000), area (500m²), and ownership (Stan 147/0, S. 8.12.). The 'Predstavnik' (Representative) section shows a representative named Dario Eržišnik with contact info (E-mail: dario.erzisnik@stanblog.hr, Phone: 091 561 5507). A 'Zadnja obavijest' (Last Message) from 'eSuvlasnik - Anketa zadovoljstvo čistoticom garaze' is shown, dated 10.07.2022.

*mUpravitelj* je aplikacija čiji je fokus na referentu zgrade, odnosno osobi koja je zaposlenik upravitelja zgrade zadužen za tu ili više zgrade. Ova aplikacija nudi niz korisnih funkcionalnosti za referenta, od reagiranja na hitne intervencije, godišnjeg pregleda zgrade, do pregleda plaćanja pričuve od strane suvlasnika.

The mobile application interface includes sections for 'INFO O ZGRADI', 'HITNA INTERVENCIJA', 'DNEVNIK ZGRADE', 'GODIŠNJI PREGLED ZGRADE', 'POSLOVNI PARTNERI', 'PROGRAM UPRAVLJANJA', 'DUŽNICI', 'STANJE I REKAPITALACIJA', 'POMOĆNI PROGRAMI' (Barcode Scanner, Server Backup, News), and 'UPRAVLJAČKA IZVJEŠĆA' (Object Management, Analysis, Risk Management).

Navedene aplikacije nude niz korisnih funkcionalnosti, ali ne rješavaju neke od problema na koje ćemo se mi fokusirati. Aplikacija *eSuvlasnik* jako je korisna što se tiče financijskog aspekta upravljanja zgradom, ali nedostaje povezanost sa platformom za komunikaciju poput StanBlog-a. Također, nudi opciju anketiranja suvlasnika, ali ne i pregled sastanaka i potvrde sudjelovanja u njima. Ova aplikacija nije podrška aktivnostima predstavnika suvlasnika, već služi kao koristan alat suvlasnicima pomoću kojeg imaju pristup određenim pravnim dokumentima i financijskim izvješćima. Što se tiče aplikacije *mUpravitelj*, izbor funkcionalnosti koji aplikacija nudi jako je koristan ali ne odgovara na pitanja koja želimo rješiti ovim projektnim zadatkom.

## Što možemo naučiti iz postojećih rješenja?

Navedene aplikacije nude niz funkcionalnosti koje su veoma korisne, poput:

- plaćanja pričuve skeniranjem koda
- reakcije na hitne intervencije klikom gumba
- usporedbe plana i realizacije odluka vezanih uz zgradu

Naš je cilj uklopiti što više korisnih funkcionalnosti poput ovih navedenih i predstaviti ih korisniku kroz jednostavno i lijepo sučelje. Ideje koje predstavljaju postojeće aplikacije su korisne i služe kao dobar primjer dijela onoga što želimo ostvariti izradom ovog projektnog zadatka.

# Funkcionalni zahtjevi

---

ID zahtjeva	Opis	Prioritet	Izvor	Kriteriji prihvatanja
F-001	Sustav omogućuje administratoru kreiranje novih korisnika	Visok	Zahtjev dionika	Administrator može dodati korisnika, dodijeliti mu e-mail, lozinku i korisničku ulogu
F-002	Sustav omogućuje korisniku promjenu lozinke	Visok	Zahtjev dionika	Prijavljen korisnik može zatražiti promjenu lozinke upisivanjem stare i nove lozinke i spremanjem promjene
F-003	Sustav omogućuje prijavu putem vanjskih servisa (OAuth 2.0)	Visok	Postojeći sustav	Korisnik se uspješno prijavljuje putem vanjskog servisa
F-004	Sustav omogućuje predstavniku suvlasnika kreiranje novog sastanka	Visok	Zahtjev dionika	Korisnik predstavnik suvlasnika kreira sastanak u stanju PLANIRAN
F-005	Sustav omogućuje predstavniku suvlasnika dodavanje točke dnevnog reda sa/bez pravnog učinka	Visok	Zahtjev dionika	Korisnik predstavnik suvlasnika dodaje točku dnevnog reda sastanku u stanju PLANIRAN.
F-006	Sustav omogućuje predstavniku suvlasnika objavljivanje sastanaka	Visok	Zahtjev dionika	Korisnik predstavnik suvlasnika postavlja sastanak s barem jednom točkom dnevnog reda u stanje OBJAVLJEN
F-007	Sustav omogućuje automatsko slanje e-mail obavijesti i postaje vidljiv na oglasnoj ploči aplikacije prilikom prelaska sastanka u stanje OBJAVLJEN	Visok	Zahtjev dionika	Sustav svim korisnicima suvlasnicima šalje obavijest na e-mail adresu prilikom postavljanja sastanka u stanje OBJAVLJEN
F-008	Sustav omogućuje predstavniku suvlasnika dodavanje Zaključka točkama dnevnog reda	Visok	Zahtjev dionika	Korisnik predstavnik stanara dodaje Zaključak <i>Izglasan</i> ili <i>Odbijen</i> točkama dnevnog reda koje imaju pravni učinak i o kojima se glasalo ili Zaključak u obliku sažetka rasprava točkama dnevnog reda koje nemaju pravni učinak

ID zahtjeva	Opis	Prioritet	Izvor	Kriteriji prihvaćanja
F-009	Sustav omogućuje predstavniku suvlasnika prebacivanje sastanka u stanje ARHIVIRAN	Visok	Zahtjev dionika	Korisnik predstavnik suvlasnika postavlja stanje sastanka koji ima Zaključak za svaku točku dnevnog reda s pravnim učinkom u stanje ARHIVIRAN
F-010	Sustav omogućuje automatsko slanje e-mail obavijesti i pregled Zaključaka prelaskom sastanka u stanje ARHIVIRAN	Visok	Zahtjev dionika	Sustav svim korisnicima suvlasnicima šalje obavijest sa Zaključcima na e-mail adresu prilikom postavljanja sastanka u stanje ARHIVIRAN
F-011	Sustav omogućuje kreiranje sastanka s jednom točkom dnevnog reda u stanju OBAVLJEN označen kao kreiran iz specifične diskusije	Visok	Zahtjev dionika	Realizirano sučelje putem kojeg aplikacija StanBlog kreira sastanak s jednom točkom dnevnog reda označen kao kreiran iz specifične diskusije u stanju OBAVLJEN. Jednom kreirani sastanak nadalje prolazi isto što i sastanci kreirani od strane korisnika predstavnika suvlasnika
F-012	Sustav se kao klijent spaja na aplikacijsko sučelje aplikacije StanBlog	Visok	Zahtjev dionika	Prilikom kreiranja točke dnevnog reda aplikacija se kao klijent spaja na sučelje aplikacije StanBlog i preuzima listu diskusija. Odabirom diskusije postavlja se poveznica na nju u točku dnevnog reda.
F-013	Sustav omogućuje administratoru unos adrese StanBlog servera koja ostvaruje integraciju sa sučeljem za diskusije StanBlog aplikacije	Visok	Zahtjev dionika	Korisnik administrator unosi adresu StanBlog servera i uspješno se ostvaruje integracija sa njihovim sučeljem za diskusije
F-014	Sustav omogućuje korisniku suvlasniku pregled sastanaka putem oglasne ploče aplikacije	Visok	Zahtjev dionika	Korisnik suvlasnik ima pregled svih sastanaka u stanju OBJAVLJEN i ARHIVIRAN
F-015	Sustav omogućuje korisniku suvlasniku potvrdu prisutnosti na sastanku	Visok	Zahtjev dionika	Korisnik suvlasnik šalje potvrdu prisutnosti na sastanku u stanju OBJAVLJEN pritiskom na za to predviđen gumb

ID zahtjeva	Opis	Prioritet	Izvor	Kriteriji prihvaćanja
F-016	Sustav omogućuje korisniku suvlasniku brzi pregled Zaključaka	Srednji	Zahtjev dionika	Korisnik suvlasnik ima opciju pregleda svih Zaključaka točaka dnevnog reda o kojima se glasalo u zasebnom prozoru
F-017	Sustav omogućuje korisniku suvlasniku sudjelovanje u glasanju o točkama dnevnog reda koje imaju pravni učinak	Srednji	Zahtjev dionika	za vrijeme trajanja sastanka odabrati opciju glasanja za svaku točku dnevnog reda koja ima pravni učinak i sprječava ponovno glasanje za istu točku.

## Nefunkcionalni zahtjevi

ID zahtjeva	Opis	Prioritet
NF-1.1	Podrška za prikaz na desktop i mobilnim uređajima	Visok
NF-1.2	Korisničko sučelje na hrvatskom jeziku	Visok
NF-1.3	Brz odziv sustava na korisničke zahtjeve	Srednji
NF-1.4	Sustav mora moći podržavati veći broj istovremenih korisnika bez značajnog povećanja vremena odziva.	Srednji
NF-1.5	Sigurno pohranjene lozinke	Visok
NF-1.6	Zaštita pohranjenih podataka	Visok
NF-1.7	Jednostavno i intuitivno korisničko sučelje	Visok
NF-1.8	Evidentiranje korisničkih aktivnosti poput kreiranja sastanaka i zaključaka	Visok
NF-1.9	Jednostavno održavanje i nadogradnja sustava	Srednji
NF-1.10	Podrška za autentifikaciju putem vanjskog servisa (OAuth 2.0)	Srednji
NF-1.11	Sustav mora biti kompatibilan s najčešće korištenim web preglednicima (Chrome, Firefox)	Visok
NF-1.12	Sustav mora biti dostupan korisnicima 24/7, neovisno o njihovoj geografskoj lokaciji i vremenskoj zoni.	Visok
NF-1.13	Sustav treba imati dovoljnu dokumentaciju.	Visok
NF-1.14	Svaka lozinka mora imati minimalno 8 znakova i sadržavati barem jedno veliko slovo, broj i poseban znak (!@#\$%^&*).	Visok
NF-1.15	Pristup podacima i funkcionalnostima mora biti ograničen na temelju korisničkih uloga.	Visok
NF-1.16	Podaci o prijavama ne smiju biti izgubljeni ili oštećeni.	Visok

# Dionici sustava:

---

- Administrator
- Predstavnik suvlasnika
- Suvlasnik
- Naručitelj
- Razvojni tim

## Aktori i njihovi funkcionalni zahtjevi

---

### A-1 Administrator (inicijator) može:

- (F-001) kreirati korisničke račune predstavnika suvlasnika i suvlasnika
- (F-001) dodijeliti ili promijeniti uloge korisnika
- (F-002) uređivati podatke postojećih korisnika

### A-2 Predstavnik suvlasnika (inicijator) može:

- (F-004) kreirati novi sastanak s osnovnim informacijama (naslov, namjera, vrijeme, mjesto)
- (F-005) dodavati točke dnevnog reda sastanka (s ili bez pravnog učinka)
- (F-006) objaviti sastanak (suvlasnici dobivaju obavijest na adresu elektroničke pošte)
- (F-007) nakon održavanja označiti sastanak kao „Obavljen“
- (F-008) svakoj od točaka dnevnog reda predstavnik može dodati zaključak.
- (F-008) označiti zaključke pravnih točaka kao „Izglasan“ ili „Odbijen“
- (F-009) može arhivirati sastanak (time prelazi u stanje „Arhiviran“)

### A-3 Suvlasnik (sudionik) može:

- (F-014) pregledavati sve objavljene sastanke i njihove dnevne redove
- (F-014, F-016) pregledavati arhivirane sastanke i njihove zaključke
- (F-015) označiti namjeru sudjelovanja na sastanku
- (F-017) sudjelovati u glasanju za točke koje imaju pravni učinak

### A-4 Aplikacija StanBlog:

- (F-011) omogućuje aplikaciji StanPlan dohvati diskusije
- (F-012) može kreirati sastanak u aplikaciji StanPlan kao što je opisano u zahtjevu
- (F-013) osigurava uspješnu integraciju aplikacija sa svoje strane

# Obrasci uporabe

---

## Opisi obrazaca uporabe

### UC1 – Kreiranje novih korisnika

- **Glavni sudionik:** Administrator
- **Cilj:** Kreiranje novih korisničkih računa
- **Sudionici:** Administrator, Baza podataka
- **Preduvjet:** Administrator je prijavljen u aplikaciju
- **Opis osnovnog tijeka:**

1. Administrator odabire opciju za kreiranje novog korisnika
2. Administrator unosi neophodne podatke za kreiranje korisnika
3. Administrator odabire opciju kreiraj
4. Aplikacija provjerava jesu li svi podatci ispravno uneseni
5. Aplikacija pristupa bazi podataka i provjerava je li korisničko ime zauzeto
6. Administrator dobiva poruku koja ga obaviještava o valjanosti podataka

### UC2 – Prijava koristeći OAuth 2.0

- **Glavni sudionik:** Korisnik
- **Cilj:** Prijava u aplikaciju
- **Sudionici:** Korisnik, OAuth 2.0
- **Preduvjet:** Korisnikov profil postoji
- **Opis osnovnog tijeka:**

1. Korisnik odabire opciju prijave koristeći OAuth 2.0
2. Aplikacija preusmjerava korisnika na stranicu OAuth 2.0
3. Korisnik odabire način prijave
4. OAuth 2.0 autentifikacija
5. Korisnik dobiva poruku sa statusom autentifikacije
6. Ako je autentifikacija uspješna, korisnik se preusmjerava na odgovarajuću stranicu

### UC3 – Promjena lozinke

- **Glavni sudionik:** Korisnik
- **Cilj:** Promjena lozinke
- **Sudionici:** Korisnik, OAuth 2.0, Baza podataka
- **Preduvjet:** Korisnikov profil postoji
- **Opis osnovnog tijeka:**

1. Korisnik odabire opciju promjene lozinke
2. Korisnik odabire način prijave
3. Ovisno o odabranom načinu prijave, OAuth 2.0 ili Baza podataka autentificiraju korisnika
4. Ako je autentifikacija uspješna, korisnik ima mogućnost upisati novu lozinku
5. Nova lozinka se pohranjuje u bazu podataka

6. Korisnik se preusmjerava na odgovarajuću stranicu

#### UC4 – Kreiranje sastanka

- **Glavni sudionik:** Predstavnik suvlasnika
- **Cilj:** Kreiranje novog sastanka
- **Sudionici:** Predstavnik suvlasnika, StanBlog, Baza podataka
- **Preduvjet:** Predstavnik Suvlasnika je prijavljen
- **Opis osnovnog tijeka:**

1. Predstavnik suvlasnika odabire opciju kreiranja sastanka
2. Predstavnik suvlasnika odabire ili kreiranje iz diskusije ili ručno kreiranje
3. Ako odabere kreiranje iz diskusije, dohvaćaju se sve diskusije. Odabirom jedne diskusije i unosom ključnih podataka kreira se novi sastanak.
4. Ako odabere ručno kreiranje od predstavnika se traži da unese naslov, sažetak vrijeme i lokaciju sastanka
5. Sastanak se postavlja u stanje "Planiran"

#### UC5 – Dodavanje točaka dnevnog reda sastanku

- **Glavni sudionik:** Predstavnik suvlasnika
- **Cilj:** Dodavanje točaka dnevnog reda postojećem sastanku
- **Sudionici:** Predstavnik suvlasnika, StanBlog, Baza podataka
- **Preduvjet:** Predstavnik Suvlasnika je prijavljen i kreirao je sastanak
- **Opis osnovnog tijeka:**

1. Predstavnik suvlasnika odabire opciju dodavanja dnevne točke
2. Predstavnik suvlasnika odabire ili kreiranje iz diskusije ili ručno kreiranje
3. Ako odabere kreiranje iz diskusije, dohvaćaju se sve diskusije. Odabirom jedne diskusije kreira se točka dnevnog reda.
4. Ako odabere ručno kreiranje od predstavnika se traži da unese temu i klasu točke dnevnog reda
5. Točka dnevnog reda je dodana

#### UC6 – Dodavanje zaključaka točkama dnevnog reda

- **Glavni sudionik:** Predstavnik suvlasnika
- **Cilj:** Dodavanje zaključaka točkama dnevnog reda
- **Sudionici:** Predstavnik suvlasnika, Baza podataka
- **Preduvjet:** Predstavnik Suvlasnika je prijavljen, odabrao je "Obavljen" sastanak
- **Opis osnovnog tijeka:**

1. Predstavnik suvlasnika odabire "Obavljen" sastanak
2. Predstavnik suvlasnika odabire točku dnevnog reda kojoj želi dodati zaključak
3. Predstavnik suvlasnika dodaje zaključak
4. Zaključak se pohranjuje u bazu podataka

#### UC7 – Postavljanje stanja "Objavljen"

- **Glavni sudionik:** Predstavnik suvlasnika

- **Cilj:** Postavljanje stanja "Objavljen"
- **Sudionici:** Predstavnik suvlasnika, Baza podataka
- **Preduvjet:** Predstavnik Suvlasnika je prijavljen, kreirao je sastanak i dodao barem jednu točku dnevnog reda
- **Opis osnovnog tijeka:**

1. Predstavnik suvlasnika postavlja stanje odabranog sastanka na "Obavljen"
2. Novo stanje se pohranjuje u bazi podataka

#### UC8 – Postavljanje stanja "Arhiviran"

- **Glavni sudionik:** Predstavnik suvlasnika
- **Cilj:** Postavljanje stanja "Arhiviran"
- **Sudionici:** Predstavnik suvlasnika, Baza podataka
- **Preduvjet:** Predstavnik Suvlasnika je prijavljen i odabrao je sastanak sa stanjem "Obavljen"
- **Opis osnovnog tijeka:**

1. Predstavnik suvlasnika postavlja stanje odabranog sastanka na "Arhiviran"
2. Ako je dodan zaključak svim točkama s pravnim učinkom, stanje se pohranjuje u bazi podataka
3. Ako ne, ispisuje se prikladna poruka

#### UC9 – Slanje obavijesti elektroničke pošte

- **Glavni sudionik:** Predstavnik suvlasnika
- **Cilj:** Slanje obavijesti o promjeni stanja sastanka suvlasnicima
- **Sudionici:** Predstavnik suvlasnika, Suvlasnici
- **Preduvjet:** Predstavnik Suvlasnika je prijavljen
- **Opis osnovnog tijeka:**

1. Predstavnik suvlasnika postavlja stanje sastanka koji zadovoljava preuvjete na "Objavljen" ili "Arhiviran"
2. Aplikacija šalje elektroničku poštu suvlasnicima i obaviještava ih o ovoj promjeni

#### UC10 – Pregled oglasne ploče

- **Glavni sudionik:** Suvlasnik
  - **Cilj:** Pregled "Objavljenih" i "Arhiviranih" sastanaka i njihovih zaključaka
  - **Sudionici:** Suvlasnik, Baza podataka
  - **Preduvjet:** Suvlasnik je prijavljen u aplikaciju, postoje odgovarajući sastanci
  - **Opis osnovnog tijeka:**
1. Suvlasnik odabire oglasnu ploču
  2. Aplikacija dohvaća sastanke iz baze podataka
  3. Ako suvlasnik odabere "Objavljen" sastanak, dobit će poruku je li sastanak započeo
  4. Ako suvlasnik odabere "Arhiviran" sastanak, moći će pregledati sve njegove zaključke

#### UC11 – Interakcija sa sastancima

- **Glavni sudionik:** Suvlasnik
- **Cilj:** Glasovanje i potvrda prisustva na "Objavljenim" sastancima

- **Sudionici:** Suvlasnik, Baza podataka
- **Preduvjet:** Suvlasnik je prijavljen u aplikaciju, odabrao je "Objavljen" sastanak
- **Opis osnovnog tijeka:**
  1. Suvlasnik odabire "Objavljen" sastanak
  2. Ako sastanak još nije počeo, suvlasnik može potvrditi prisustvo
  3. Pohrana prisustva u bazi podataka
  4. Ako je sastanak počeo, suvlasnik može glasati za točke dnevnog reda
  5. Ako suvlasnik ponovno pokuša glasati za točku za koju je to već učinio, dobit će poruku greške

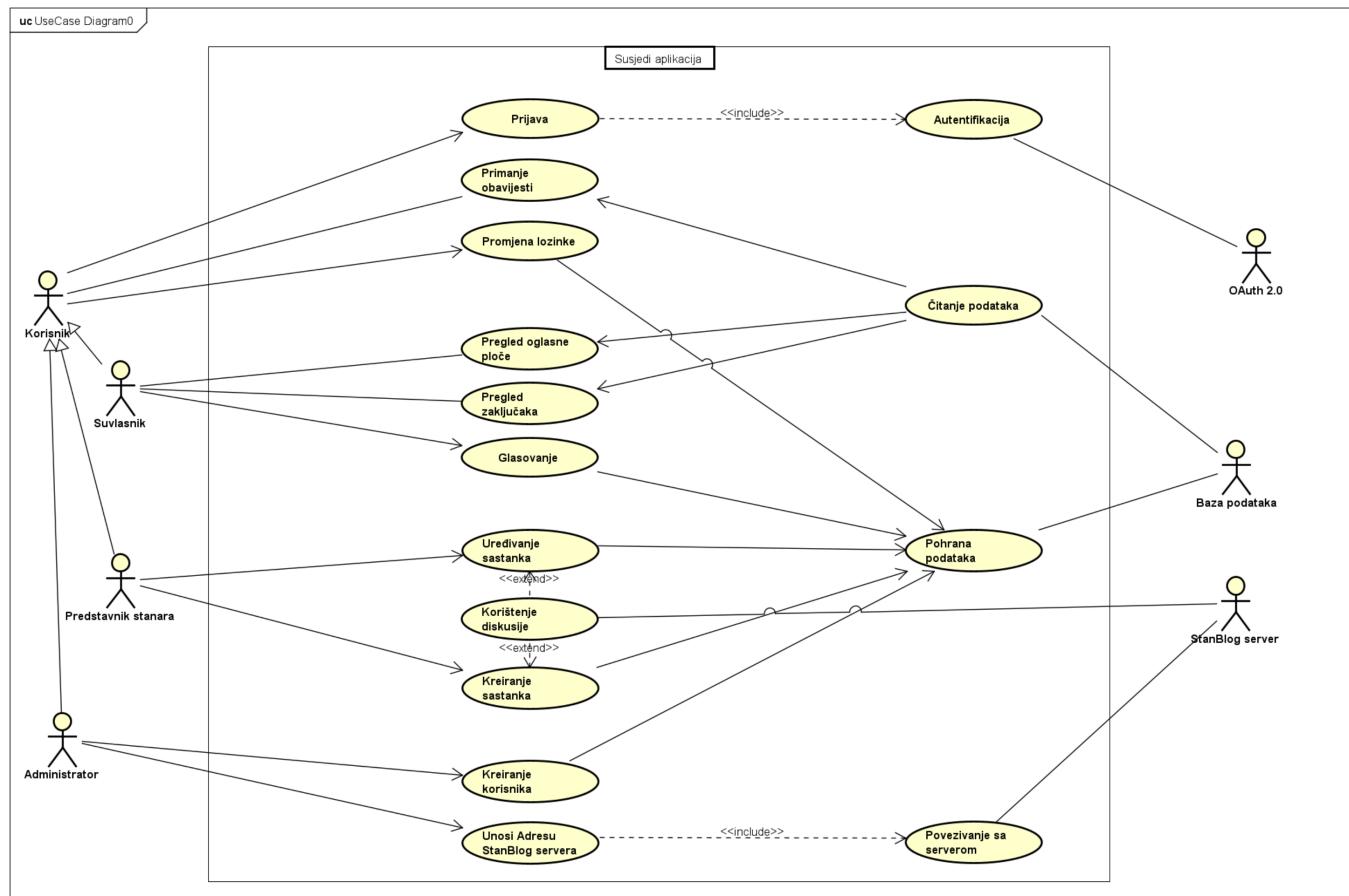
## UC12 – Unos adrese StudBlog servera

- **Glavni sudionik:** Administrator
- **Cilj:** Unos adrese StudBlog servera
- **Sudionici:** Administrator, Baza podataka
- **Preduvjet:** Administrator je prijavljen u aplikaciju
- **Opis osnovnog tijeka:**
  1. Administrator odabire opciju spajanja aplikacije sa StudBlogom
  2. Administrator unosi adresu StudBlog servera
  3. Adresa se pohranjuje u bazu podataka

## Dijagrami obrazaca uporabe

---

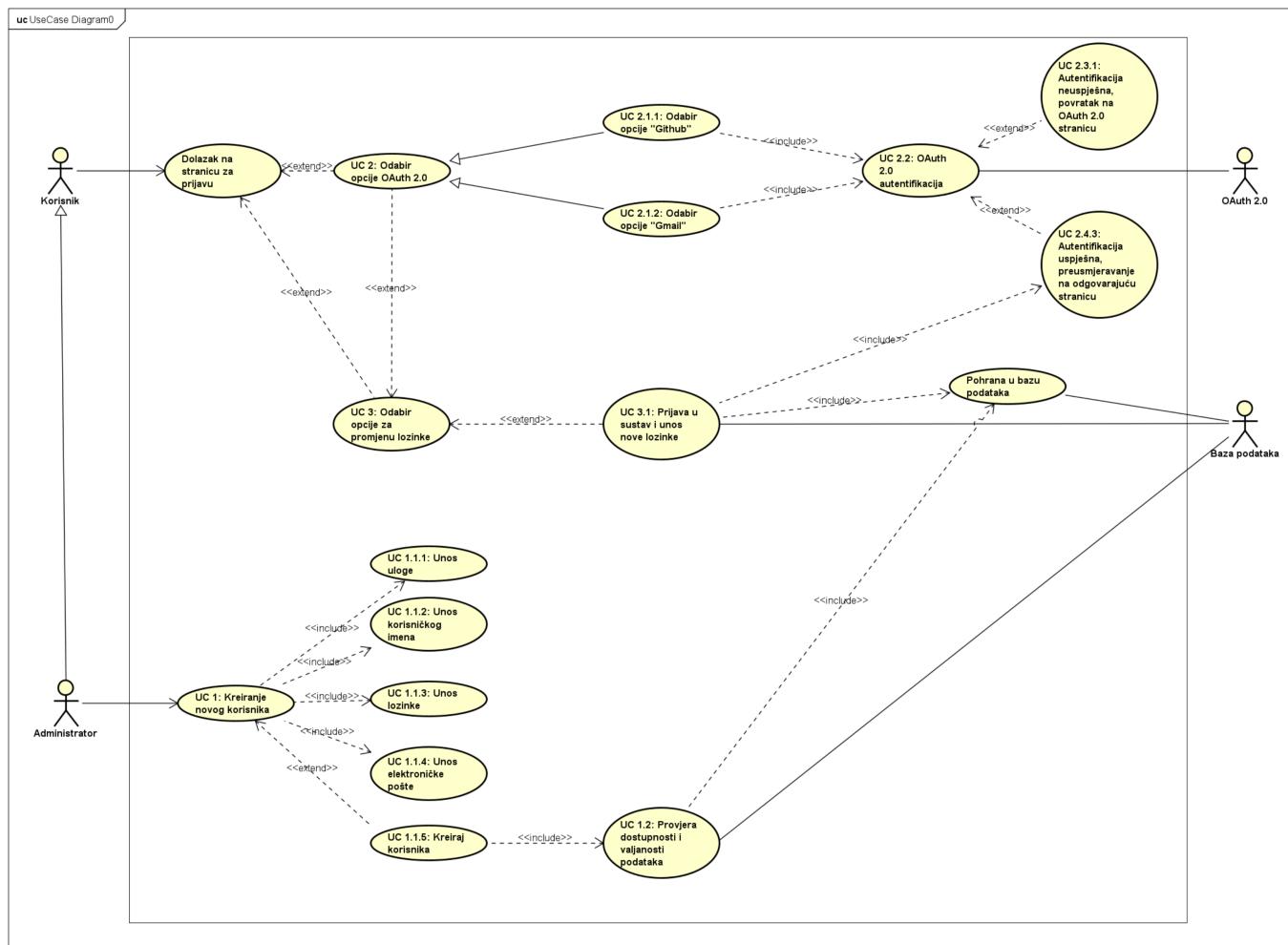
### 1. Visokorazinski dijagram obrazaca uporabe cijelog sustava



Slika 3.1: Dijagram obrazaca uporabe: cjelokupni pregled sustava

Dijagram obrazaca uporabe cijelog sustava prikazuje funkcionalnosti aplikacije Susjedi iz perspektive krajnjih korisnika. Predstavnik suvlasnika organizira sastanke na kojima suvlasnici mogu glasovati.

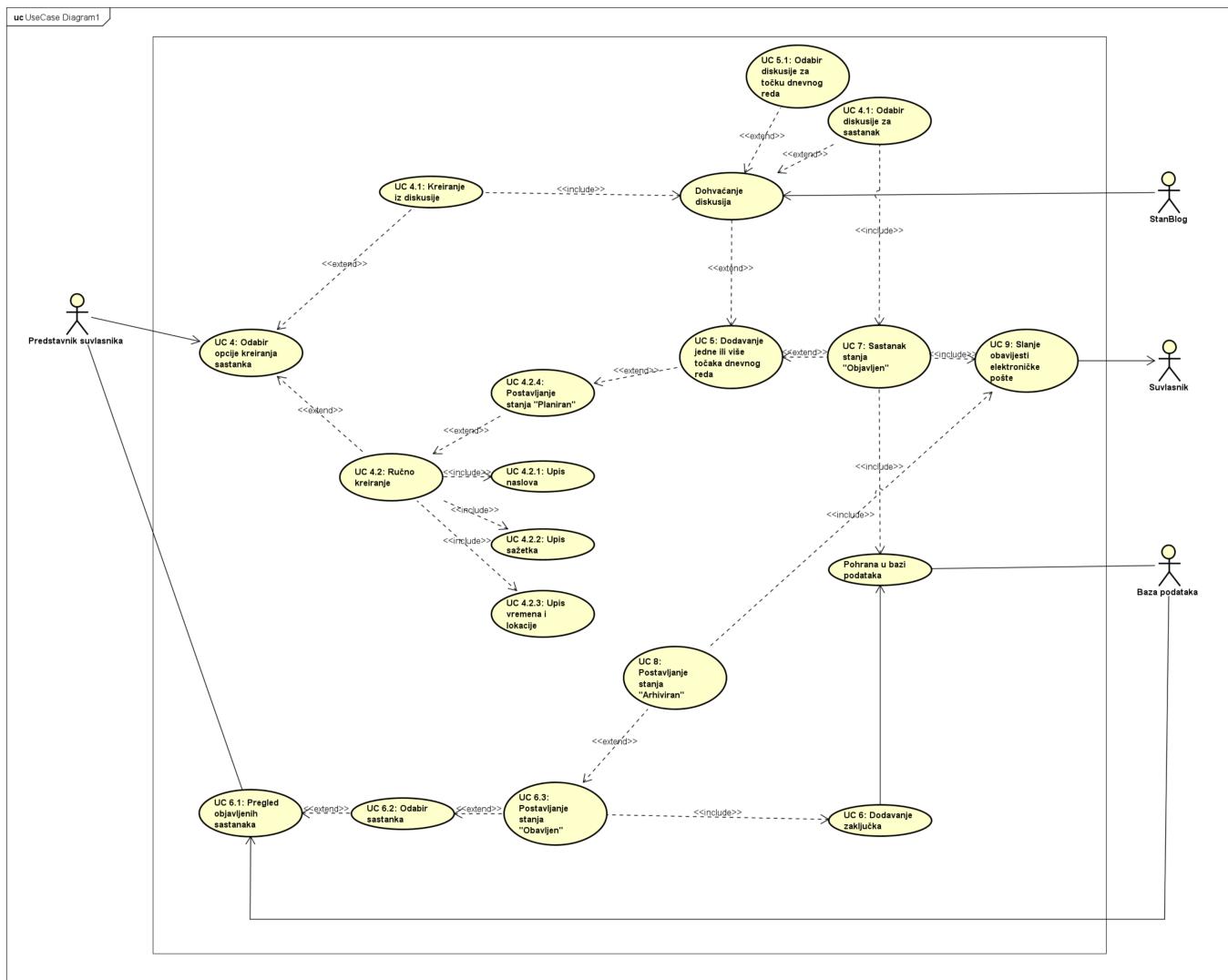
## 2. Dijagram obrazaca uporabe za ključne značajke



Slika 3.2: Dijagram obrasca uporabe: stvaranje, autentifikacija i promjena lozinke korisnika

Dijagram obrasca uporabe koji prikazuje funkcionalnosti stvaranja, autentifikacije i promjene lozinke korisnika. Samo administratori mogu stvoriti nove korisnike. Korisnici kasnije mogu promijeniti početnu lozinku koju je postavio administrator. Omogućena je i prijava putem OAuth 2.0.

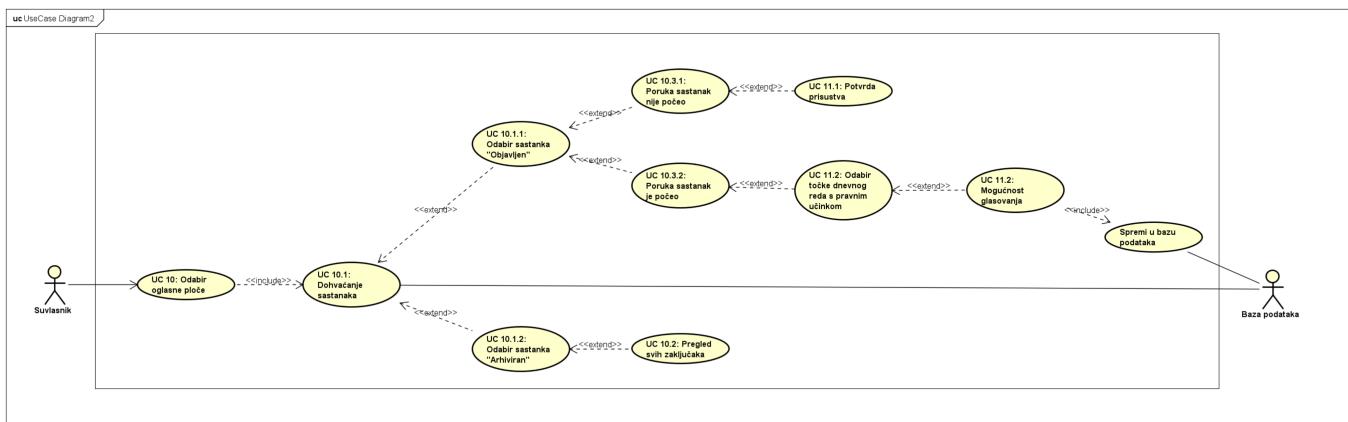
### 3. Dijagram obrazaca uporabe za korisničke uloge



Slika 3.3: Dijagram obrasca uporabe: objavljivanje novog sastanka

Fokus ovog dijagrama je predstavnik suvlasnika i njegova uloga u stvaranju i uređivanju sastanaka i praćenju točaka dnevnog reda kroz njihova stanja.

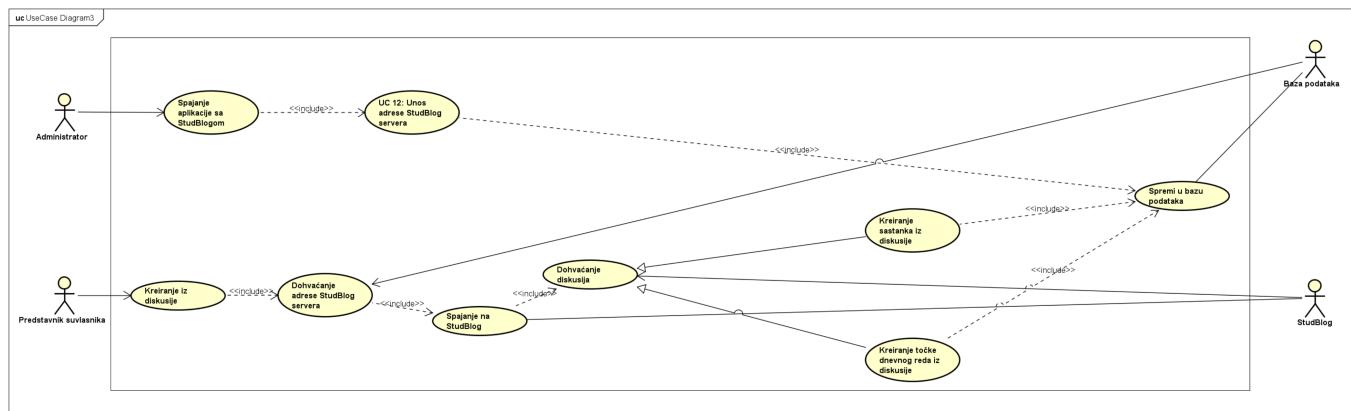
#### 4. Dijagram obrazaca uporabe za osnovne poslovne procese



Slika 3.4: Dijagram obrasca uporabe: pregled sastanaka

Dijagram obrasca uporabe koji prikazuje ulogu suvlasnika na sastancima.

#### 5. Dijagram obrazaca uporabe za kritične sustave i integracije



Slika 3.5: Dijagram obrasca uporabe: spajanje sa StudBlog serverom

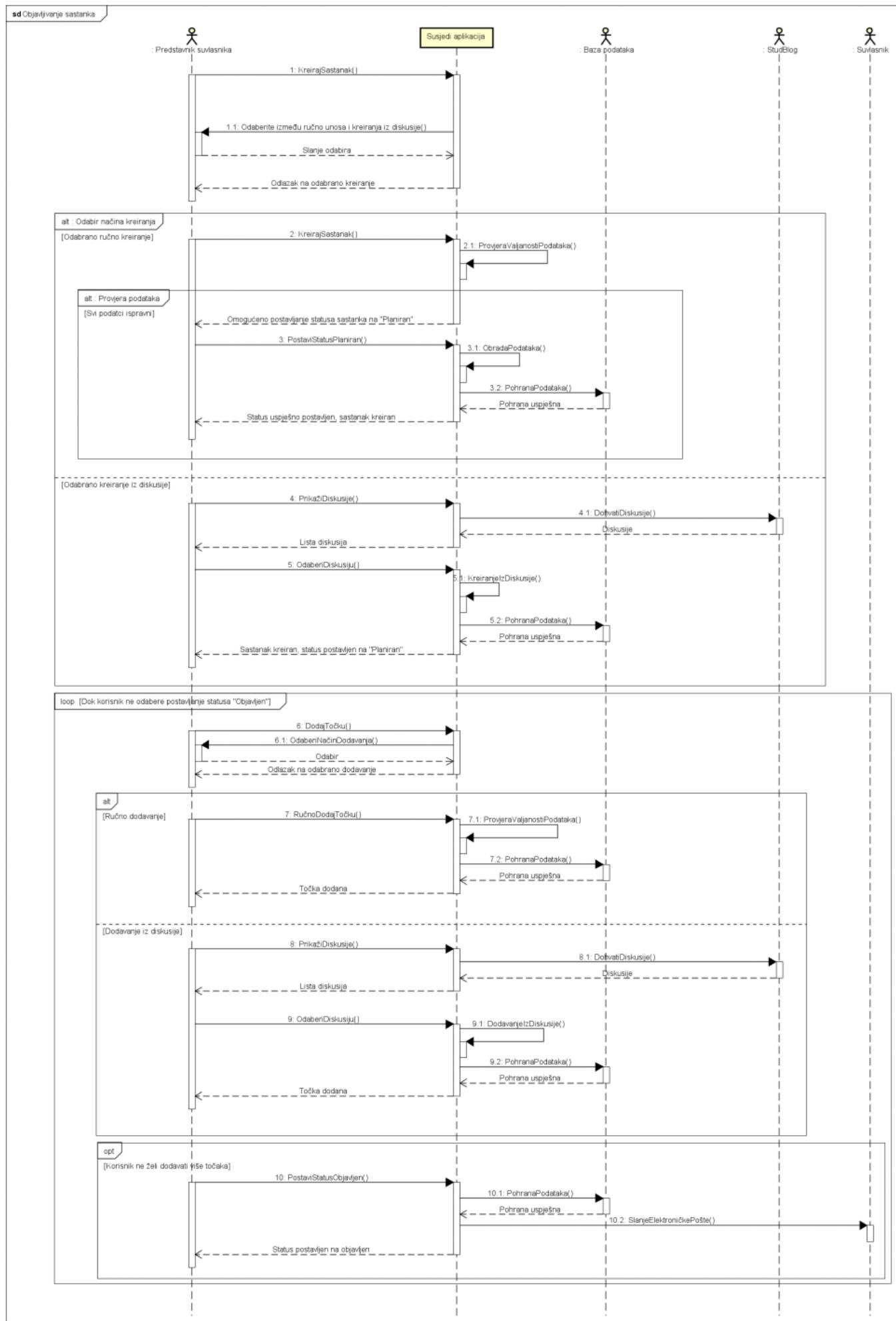
Dijagram obrasca uporabe koji prikazuje funkcionalnost dohvaćanja diskusija sa StudBlog servera koje se onda mogu koristiti za kreiranje sastanaka i točaka dnevnog reda.

## Sekvencijski dijagrami

---

### Objavljivanje sastanaka

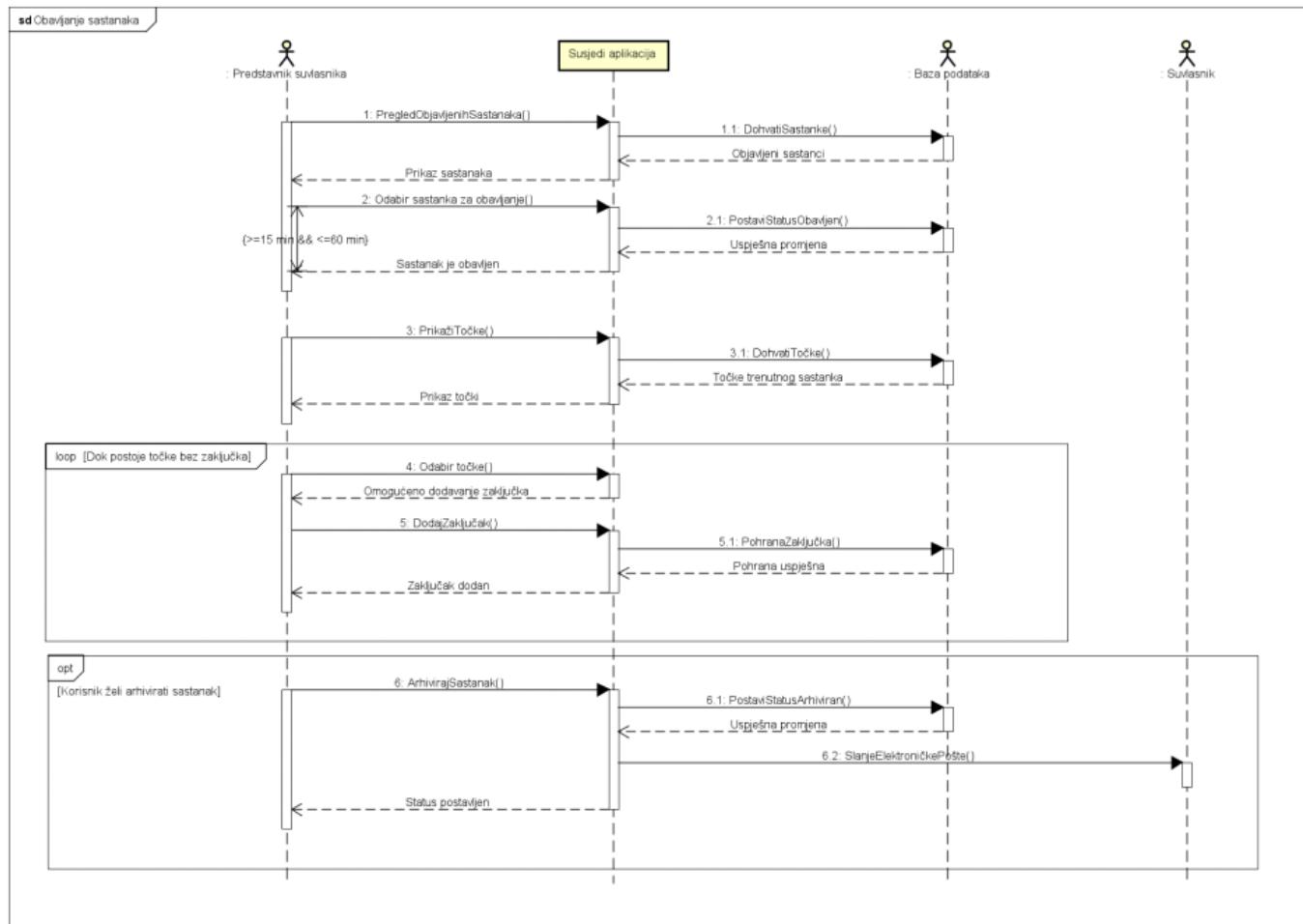
Sastanke kreira predstavnik suvlasnika. Prilikom odabira opcije kreiranja sastanka, nudi mu se opcija za ručno kreiranje i opcija za kreiranje iz StudBlog diskusije. Ako odabere prvu opciju, ručno unosi naslov, sažetak, vrijeme i lokaciju sastanka. Kada su ovi podatci ispunjeni, predstavnik suvlasnika može sastanku postaviti stanje "Planiran". Ako odabere drugu opciju, aplikacija dohvaća diskusije sa StudBlog servera i prikazuje mu ih. Odabirom jedne diskusije stvara se odgovarajući sastanak i njegovo stanje se postavlja na "Planiran". Planiranom sastanku potrebno je dodati barem jednu točku prije nego ga se može objaviti. Predstavnik suvlasnika može dodati još proizvoljan broj točaka. Točke se, slično kao i sastanci, mogu dodati i ručno i koristeći diskusije. Za točke je potrebno odabrati pravnu važnost. Kada se stanje sastanka postavi na "Objavljen", pošalje se obavijest elektroničkom poštom suvlasnicima.



Slika 3.6: Sekvencijski dijagram: Objavljivanje sastanaka

## Obavljanje sastanka

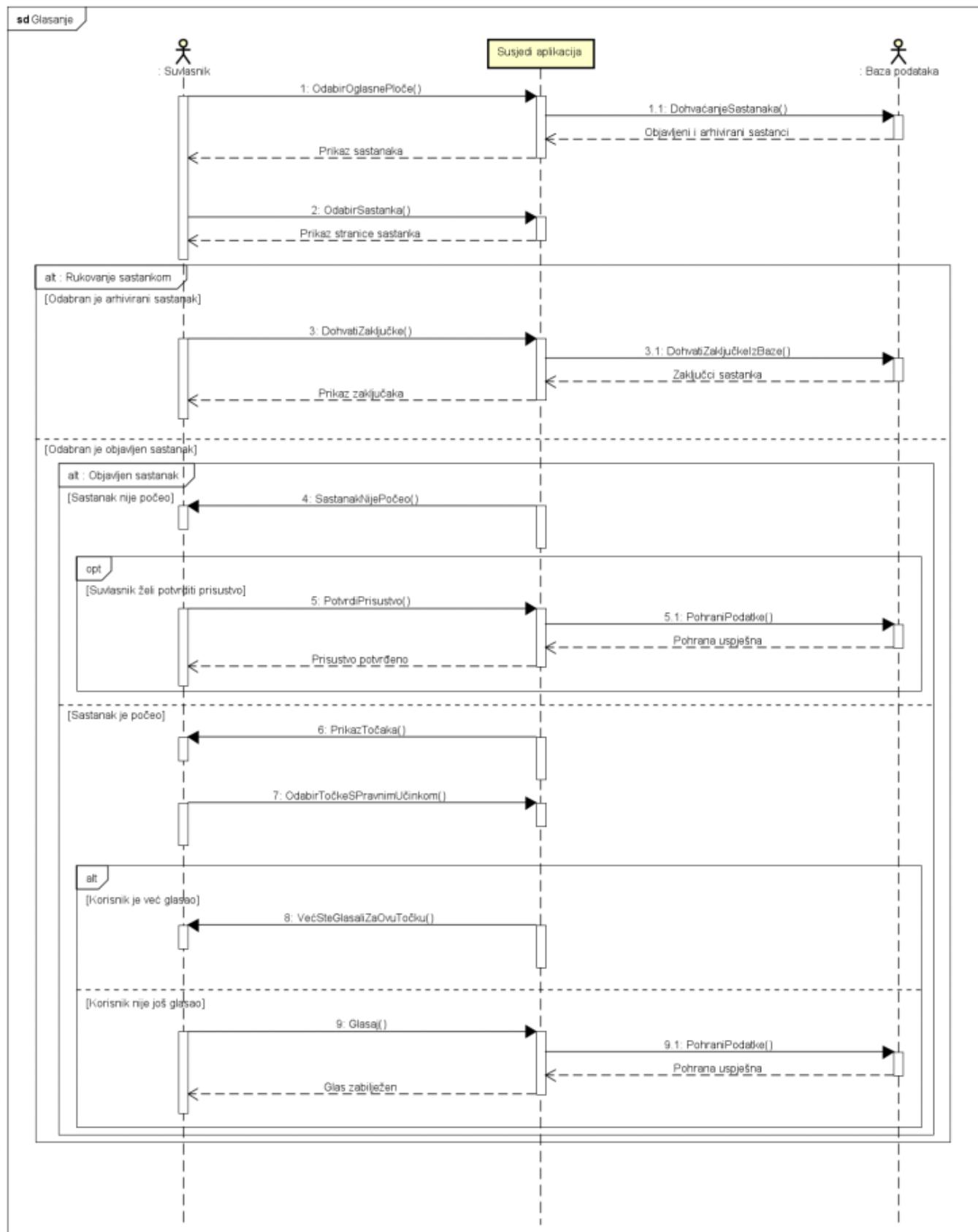
Predstavnik suvlasnika je zadužen i za obavljanje sastanaka. Predstavnik stanara može obaviti samo objavljen sastanak. Odabirom sastanka sastanak se započinje, njegovo predviđeno trajanje je od 15 do 60 minuta. Nakon sastanka, predstavnik može dodavati zaključke točkama dnevnog reda. Kada su zaključci dodani svim točkama, predstavniku stanara se nudi opcija arhiviranja sastanka. Obavijest o arhiviranju sastanka šalje se suvlasnicima putem elektroničke pošte.



Slika 3.7: Sekvenčni dijagram: Obavljanje sastanaka

## Glasanje

Suvlasnik odabirom oglasne ploče dobiva uvid u objavljene i arhivirane sastanke. Pri odabiru arhiviranog sastanka nudi se mogućnost dohvaćanja njegovih zaključaka. Ako korisnik odabere objavljeni sastanak koji još nije počeo, dobit će poruku koja ga na to upozorava. U tom slučaju može potvrditi svoje prisustvo. Ako odabere objavljeni sastanak koji je u tijeku, prikazuju mu se točke dnevnog reda. Odabirom na točku s pravnim učinkom, korisnik može glasati ili je već glasao. U slučaju da je već glasao dobiti će odgovarajuću poruku. Ako još nije glasao, sada može. Prilikom glasanja dobiva poruku da je njegov glas uspješno zabilježen.



Slika 3.8: Sekvencijski dijagram: Glasanje

### ID Obrasca uporabe    Uključenost funkcionalnih zahtjeva

UC1	F-001
UC2	F-003

<b>ID Obrasca uporabe</b>	<b>Uključenost funkcionalnih zahtjeva</b>
UC3	F-002
UC4	F-004, F-011
UC5	F-005, F-012
UC6	F-008
UC7	F-006
UC8	F-009
UC9	F-007, F-010
UC10	F-014, F-016
UC11	F-015, F-017
UC12	F-013

# Arhitektura sustava

---

Cilj ovog poglavlja je pružiti jasan, sažet i strukturiran pregled arhitekture Vašeg programskog sustava u razvoju, uz razrađivanje ključnih komponenata i njihovih međusobnih odnosa. Ova dokumentacija treba omogućiti svim sudionicima projekta potpuno razumijevanje arhitekture sustava, načina implementacije, podjele odgovornosti te kako sustav funkcionira kao cjelina. Uključivanje odgovarajućih dijagrama pomaže u kvalitetnom dokumentiranju i vizualizaciji strukture sustava i njegovih ključnih komponenti.

## Opis arhitekture

### Stil arhitekture

Arhitektura našeg sustava temelji se klijent-poslužitelj modelu s jasno razdvojenim slojevima frontenda i backenda. Taj stil arhitekture omogućuje jasnu podjelu odgovornosti između korisničkog sučelja i logike. Time se olakšava razvoj, testiranje i održavanje sustava. Frontend dio sustava razvijen je korištenjem React frameworka, koja omogućuje izradu dinamičkog single-page korisničkog sučelja. React je odabran zbog jednostavne uporabe, popularnosti i potpore, učinkovitog prikaza podataka i podrške za modularni razvoj. Backend dio razvijen je pomoću Spring Boot frameworka, koji omogućuje brzo i pouzdano postavljanje REST API servisa. Spring Boot pruža dobru integraciju s bazama podataka, podršku za sigurnosni mehanizam OAuth 2.0 i jasan okvir za organizaciju logike programa u slojevima. Klijent-poslužitelj arhitektura odabrana je jer omoguće: -neovisnost timova (frontend i backend timovi mogu raditi paralelno), -lakšu izmjenu i nadogradnju pojedinih dijelova sustava, -jasno definirano sučelje komunikacije preko REST API-ja.

### Podsustavi

Sustav je podijeljen u tri glavna podsustava:

- Podsustav za upravljanje korisnicima Odgovoran je za autorizaciju i upravljanje korisničkim računima. Administrator može kreirati, uređivati i brisati korisnike te dodjeljivati njihove uloge (administrator, predstavnik suvlasnika, suvlasnik). Ovaj podsustav implementira kontrolu pristupa na temelju korisničke uloge.
- Podsustav za upravljanje sastancima Predstavnik suvlasnika koristi ovaj podsustav za kreiranje, uređivanje i objavu sastanaka. On omogućuje dodavanje točaka dnevnog reda, označavanje sastanaka kao obavljenih te arhiviranje starih sastanaka. Također se bilježe zaključci pravnih točaka i ishodi glasanja.
- Podsustav za pregled i sudjelovanje suvlasnika Suvlasnici putem ovog dijela sustava mogu pregledavati objavljene sastanke, arhivirane odluke i izražavati namjeru sudjelovanja. Omogućeno im je i glasanje o pravnim točkama dnevnog reda. Ovi podsustavi međusobno komuniciraju putem REST API poziva.

### Preslikavanje na radnu platformu

Aplikacija će se izvoditi lokalno, što omogućuje jednostavno testiranje i demonstraciju funkcionalnosti. Za potrebe razvoja koristi se GitHub repozitorij radi verzioniranja koda i suradnje među članovima tima. Dizajn korisničkog sučelja izrađen je u alatu Figma, koji omogućuje timsku suradnju na vizualnom prototipu aplikacije i usklađivanje dizajna s React implementacijom.

### Spremišta podataka

Za pohranu podataka koristi se relacijska baza podataka PostgreSQL. Odabrana je zbog opširne dostupne dokumentacije, podrške za kompleksne SQL upite i dobre integracije sa Spring Boot frameworkom. Baza će sadržavati tablice za:

- korisnike i njihove podatke,
- sastanke i točke dnevnog reda,
- glasanja i rezultate glasanja,
- arhivirane sastanke i zaključke. Pristup bazi implementira se ručno pisanjem SQL upita, bez upotrebe ORM sustava poput Hibernate-a, čime se ostvaruje veća kontrola nad strukturom i izvedbom upita.

## Mrežni protokoli

Komunikacija između frontend i backend dijela odvija se putem HTTP/HTTPS protokola koristeći RESTful API. Backend vraća podatke u obliku JSON objekata koje frontend interpretira i prikazuje korisniku. Za potrebe autorizacije koristi se OAuth 2.0 standard, koji omogućuje sigurno upravljanje pristupnim tokenima i autorizacijom prema korisničkim ulogama.

## Globalni upravljački tok

1. Korisnik pristupa web sučelju putem React-a.
2. Frontend šalje zahtjeve prema Spring Boot REST API-ju (npr. prijava, dohvata sastanaka, glasanje).
3. Backend obrađuje zahtjev, provjerava korisničke ovlasti (putem OAuth 2.0) i izvršava poslovnu logiku.
4. Backend komunicira s PostgreSQL bazom kako bi pročitao ili pohranio podatke.
5. Rezultat obrade vraća se frontend dijelu u obliku JSON odgovora.
6. Frontend ažurira prikaz korisničkog sučelja u stvarnom vremenu.

## Sklopovsko-programski zahtjevi

Za pokretanje sustava dovoljno je osobno računalo s modernim web preglednikom i osnovnim razvojnim alatima. Tehnički zahtjevi su:

- Procesor: Intel Pentium ili ekvivalent,
- Radna memorija: preporučeno 4 GB RAM,
- Operativni sustav: Windows 10 ili 11
- Programska podrška:
- Node.js i npm (za pokretanje React frontend-a),
- Java 17+ i Spring Boot framework,
- PostgreSQL poslužitelj,
- Git za verzioniranje.

## Obrazloženje odabira arhitekture

Ključni zahtjevi projekta uključuju jasnu podjelu odgovornosti, sigurnu autorizaciju korisnika i pregledno korisničko sučelje. Zbog toga je odabran klijent-poslužitelj arhitektonski stil, koji omogućuje logičku i tehničku odvojenost korisničkog sučelja (frontend) od poslovne logike i pristupa podacima (backend). Ovaj pristup pruža timovima fleksibilnost u razvoju i testiranju, te osigurava modularnost sustava. Frontend je implementiran pomoću React frameworka, jer omogućuje izradu dinamičnih i responzivnih single-page aplikacija s visokom razinom interaktivnosti. React podržava komponentni pristup koji olakšava održavanje i ponovnu upotrebu dijelova sučelja. Backend je implementiran u Spring Boot frameworku, odabranom zbog

jasno definirane strukture projekata, podrške za REST servise i integracije sa sigurnosnim standardima poput OAuth 2.0. Korištenje PostgreSQL baze podataka omogućuje dosljednost, referencijalni integritet i pouzdanu pohranu podataka, što je ključno za aplikaciju koja obrađuje podatke o sastancima, korisnicima i glasanjima. Ova kombinacija tehnologija omogućuje:

- odvojenu implementaciju slojeva (frontend i backend),
- sigurnu autorizaciju korisnika prema njihovim ulogama,
- lagani nadogradnji i skaliranje sustava u budućnosti,
- brz razvoj i lako testiranje zahvaljujući jasnim API sučeljima.

Pri oblikovanju arhitekture primjenjeni su sljedeći principi:

- podjela odgovornosti: Svaki podsustav ima jasno definiranu odgovornost (upravljanje korisnicima, sastancima, ili glasanjima), čime se olakšava razumijevanje i održavanje koda.
- Neovisnost slojeva: Frontend i backend komuniciraju isključivo putem REST API-ja, što omogućuje neovisnost slojeva. Promjene u jednom dijelu sustava ne zahtijevaju značajne izmjene u drugom.
- Fleksibilnost: Modularna struktura React komponenti i Spring Boot servisa omogućuje jednostavno dodavanje novih funkcionalnosti (npr. ankete, obavijesti, povijest troškova) bez izmjene osnovne arhitekture.
- Sigurnost: Korištenjem OAuth 2.0 standarda omogućeno je sigurno upravljanje korisničkim pristupom, što je važno jer aplikacija obrađuje privatne podatke stanara. Primjenom ovih principa postignuta je ravnoteža između jednostavnosti implementacije i dugoročne održivosti sustava.

## Razmatrane alternative

Razmatrana je upotreba Pythona za razvoj poslužiteljske strane, no Java i Spring Boot odabrani su zbog izražene objektne orijentiranosti, bolje podrške za tipizaciju, razvijenog ekosustava biblioteka i integracije s PostgreSQL bazom. Odabirom klijent-poslužitelj arhitekture s React-om i Spring Boot-om postignut je optimalan kompromis između jednostavnosti razvoja, modularnosti i pouzdanosti.

## Zaključak

Odabrana arhitektura u potpunosti podržava potrebe projekta: omogućuje transparentnu komunikaciju, sigurnu autorizaciju, jednostavno upravljanje podacima i mogućnost budućeg proširenja.

## Organizacija sustava na visokoj razini

Sustav je organiziran prema klijent-poslužitelj arhitekturi. Korisnici pristupaju aplikaciji putem web preglednika (klijent), dok backend dio sustava (poslužitelj) obrađuje zahteve, izvršava poslovnu logiku i komunicira s bazom podataka. Komunikacija između klijenta i poslužitelja odvija se putem REST API-ja koristeći HTTP protokol. Središnje spremište podataka čini relacijska baza podataka PostgreSQL, koja pohranjuje sve informacije o korisnicima, sastancima, točkama dnevnog reda i glasanjima. Baza osigurava integritet podataka i podržava odnose između entiteta kroz relacijski model. Grafičko korisničko sučelje razvijeno je kao web aplikacija korištenjem React frameworka. Ono korisnicima omogućuje interakciju sa sustavom putem preglednika. Frontend dio komunicira s backend servisima izrađenima u Spring Bootu, čime se ostvaruje potpuna funkcionalna povezanost između korisničkog sučelja, aplikacijske logike i baze podataka.

## Organizacija aplikacije

Aplikacija se sastoji od frontend i backend dijela koji međusobno komuniciraju putem REST API-ja u JSON formatu.

## Frontend sloj

Frontend je razvijen u Reactu i čini prezentacijski sloj aplikacije. Sadrži komponente koje prikazuju podatke i omogućuju korisničke interakcije. Glavne komponente frontend sloja:

- Komponente sučelja – prikaz sastanaka, točaka dnevnog reda, glasanja i korisničkih profila.
- Servisi za API komunikaciju – upravljanje zahtjevima prema backendu (axios ili fetch servisi).
- Autentifikacija i autorizacija – integracija s OAuth 2.0 servisom.
- State management – upravljanje globalnim stanjem pomoću React Context API-ja. Frontend sloj je odgovoran isključivo za prikaz podataka i upravljanje interakcijom korisnika, bez implementacije glavne logike.

## Backend sloj

Backend je implementiran u Spring Boot frameworku i čini poslovno-logički i podatkovni sloj aplikacije. Organiziran je prema MVC principu i sastoji se od četiri osnovna sloja:

### Controller sloj

- Prima zahtjeve od frontenda i vraća odgovore u JSON formatu.
- Usmjerava zahtjeve prema odgovarajućim servisima.
- Primjeri: UserController, MeetingController, VotingController.

### Service sloj

- Sadrži poslovnu logiku (npr. pravila glasanja, validacija sastanaka, provjere korisničkih uloga).
- Predstavlja glavni logički sloj aplikacije.

### Repository sloj

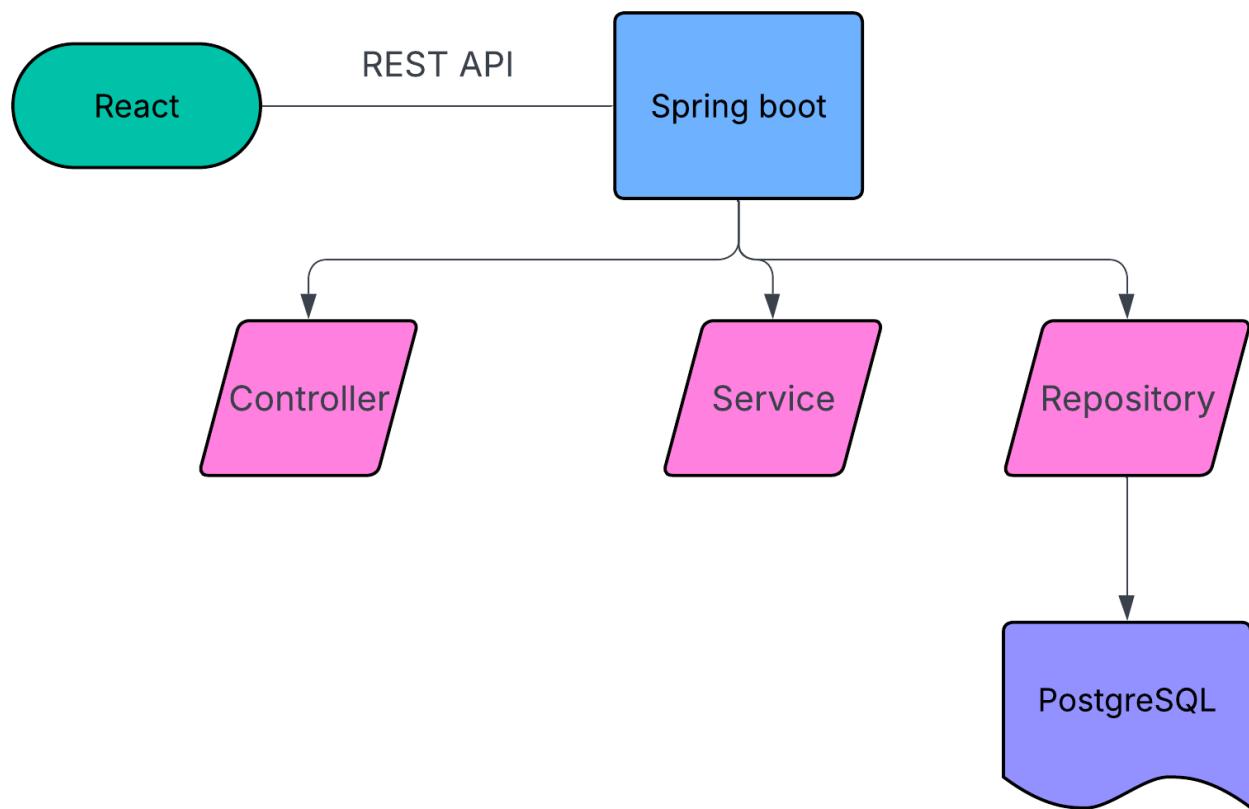
- Zadužen za komunikaciju s bazom podataka putem Spring Data JPA.
- Omogućuje dohvati, pohranu i ažuriranje entiteta poput User, Meeting, Agendaltem.

### Model sloj (Entities)

- Predstavlja podatkovne modele koji mapiraju relacije u PostgreSQL bazi podataka.
- Svaki entitet odgovara jednoj tablici u bazi.

Autentifikacija i autorizacija provode se kroz Spring Security i OAuth 2.0, dok se slanje obavijesti vrši putem zasebne komponente Email Notification Service.

## Dijagram visoke razine



## Reference

Architect modern web applications with ASP.NET core and azure - .NET, Architect modern web applications with ASP.NET Core and Azure - .NET | Microsoft Learn. Dostupno: <https://learn.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/> (7. studeni 2024.).

## Baza podataka

---

Pri izradi sustava korištena je relacijska baza podataka, točnije PostgreSQL. Prednost PostgreSQL-a je u tome što pruža stabilno i sigurno okruženje za pohranu podataka uz pratnju SQL standarda. Glavne komponente baze su relacije i veze među njima. U bazi podataka nalaze se sljedeće tablice:

- Users
- Meetings
- Agenda\_items
- Conclusions
- Meeting\_attendances
- System\_config
- Flyway\_schema\_history

## Opis tablica

### Users

Ova tablica sadržava informacije o korisničkim računima stanara. Sadržava atribute: id(primarni ključ), username, password, email, role, created\_at, oauth\_provider, oauth\_provider\_id i last\_login. Ovaj entitet ima relaciju jedan na više sa tablicama meetings i meeting\_attendances.

Atribut	Tip podatka	Opis varijable
id	BIGSERIAL	Jedinstveni identifikator(PK)
username	VARCHAR(50)	Jedinstveno ime korisnika(UNIQUE)
password	VARCHAR (100)	Šifra računa
email	VARCHAR(100)	Elektronička pošta korisnika(UNIQUE)
role	VARCHAR(15)	Uloga korisničkog računa
created_at	TIMESTAMP	Vrijeme izrade računa
oauth_provider	VARCHAR(20)	Naziv OAuth pružatelja identiteta
oauth_provider_id	VARCHAR(100)	Identifikator korisnika kod OAuth pružatelja
last_login	TIMESTAMP	Vrijeme zadnje uspješne prijave

## Meetings

Ova tablica sadržava informacije o sastancima koje kreiraju korisnici. Sadržava atribute: id(primarni ključ), title, summary, meeting\_ts, location, state, created\_by(strani ključ), created\_at i updated\_at. Ovaj entitet ima relaciju jedan na više s tablicom agenda\_items te relaciju više na više s users putem tablice meeting\_attendances.

Atribut	Tip podatka	Opis varijable
id	BIGSERIAL	Jedinstveni identifikator sastanka (PK)
title	VARCHAR(150)	Naziv sastanka
summary	TEXT	Sažetak sastanka
meeting_ts	TIMESTAMP	Vrijeme održavanja sastanka
location	VARCHAR(150)	Lokacija sastanka
state	meeting_state	Trenutno stanje sastanka (PLANIRAN / OBJAVLJEN / OBAVLJEN / ARHIVIRAN)
created_by	BIGINT	Korisnik koji je kreirao sastanak (FK; users.id)
created_at	TIMESTAMP	Vrijeme kreiranja zapisa
updated_at	TIMESTAMP	Vrijeme zadnje izmjene

## Agenda\_items

Tablica koja sadrži točke dnevnog reda pripadajuće sastanku. Svaka točka pripada jednom sastanku; može imati oznaku da ima pravni učinak ili da zahtijeva glasovanje. Sadrži atribute: id(primarni ključ), meeting\_id(strani ključ), title, description, order\_number, has\_legal\_effect, requires\_voting,

stanblog\_discussion\_url, created\_at. Relacije: više na jedan prema meetings, jedan na jedan prema conclusions (putem jedinstvenog FK).

Atribut	Tip podatka	Opis varijable
id	BIGSERIAL	Jedinstveni identifikator (PK)
meeting_id	BIGINT	ID sastanka kojem točka pripada (FK ; meetings.id)
title	VARCHAR(150)	Naslov točke dnevnog reda
description	TEXT	Opis točke / bilješke
order_number	INTEGER	Redni broj točke u dnevnom
has_legal_effect	BOOLEAN	Ima li točka pravni učinak
requires_voting	BOOLEAN	Treba li za točku provesti glasovanje
stanblog_discussion_url	VARCHAR(500)	Poveznica na diskusiju u StanBlog aplikaciji
created_at	TIMESTAMP	Vrijeme dodavanja točke

## Conclusions

Tablica zaključaka za pojedinu točku dnevnog reda. Sadrži atribute: id(primarni ključ), agenda\_item\_id(strani ključ), content, voting\_result i created\_at. Ima relaciju više na jedan prema agenda\_items.

Atribut	Tip podatka	Opis varijable
id	BIGSERIAL	Jedinstveni identifikator (PK)
agenda_item_id	BIGINT	ID točke za koju je zaključak (FK ; agenda_items.id)
content	TEXT	Tekst zaključka
voting_result	voting_result	Rezultat glasanja (IZGLASAN / ODBIJEN)
created_at	TIMESTAMP	Vrijeme kreiranja zaključka

## Meeting\_attendances

Ova tablica predstavlja potvrde dolaska korisnika na sastanke. Sadrži atribute meeting\_id(strani ključ), user\_id(strani ključ) i confirmed\_at. Sadrži relacije više na više s meetings i users.

Atribut	Tip podatka	Opis varijable
meeting_id	BIGINT	Sastanak (FK ; meetings.id)
user_id	BIGINT	Korisnik (FK ; users.id)
confirmed_at	TIMESTAMP	Vrijeme potvrde dolaska

## System\_config

Ova tablica sadržava konfiguracijske postavke sustava. Sadrži atribute id(primarni ključ), config\_key, config\_value i updated\_at.

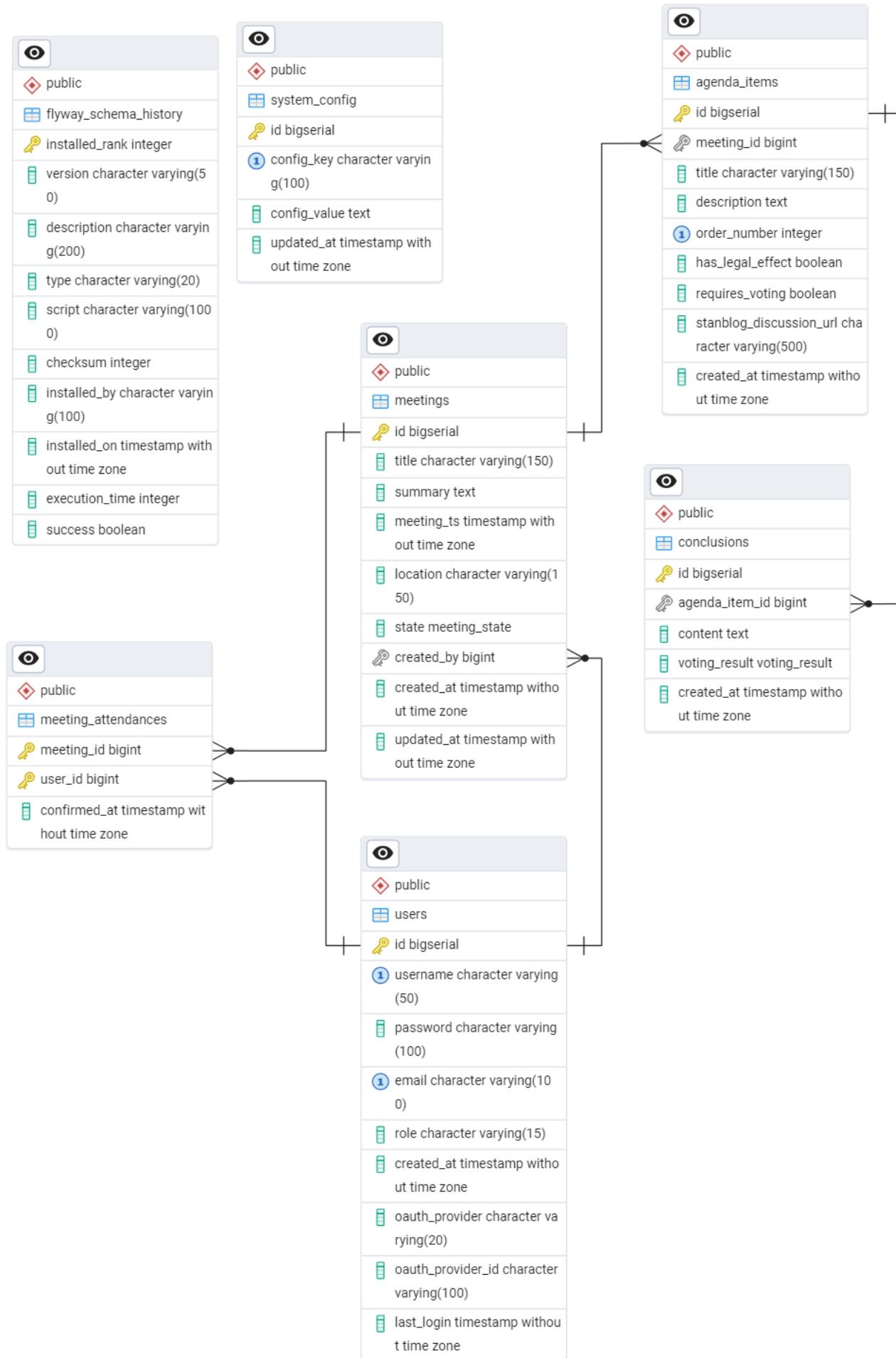
Atribut	Tip podatka	Opis varijable
id	BIGSERIAL	Jedinstveni identifikator (PK)
config_key	VARCHAR(100)	Naziv konfiguracijske postavke
config_value	TEXT	Vrijednost konfiguracijske postavke
updated_at	TIMESTAMP	Vrijeme zadnje izmjene

### Flyway\_schema\_history

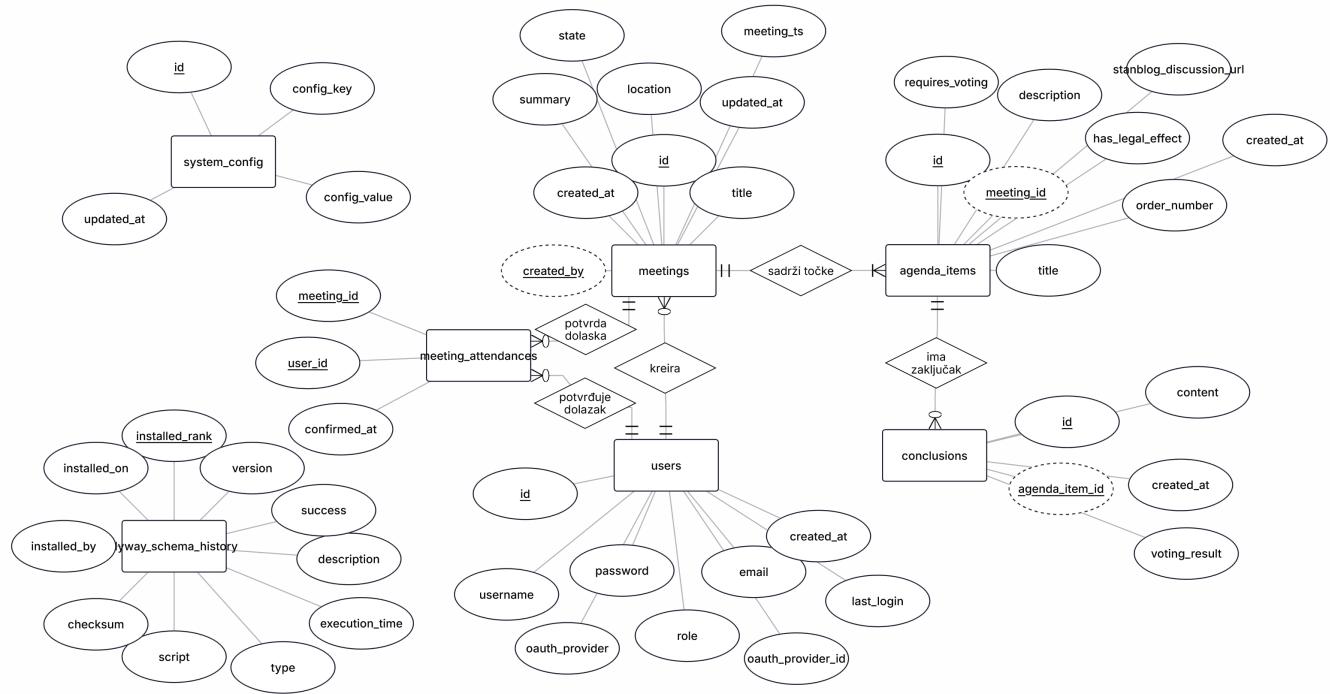
Ova tablica predstavlja povijest izvršenih migracija nad bazom podataka. Svaki zapis označava jednu migracijsku skriptu koja je izvršena putem Flyway alata. Tablica sadržava installed\_rank, version, description, type, script, checksum, installed\_by, installed\_on, execution\_time i success.

Atribut	Tip podatka	Opis varijable
installed_rank	INTEGER	Redni broj instalirane migracije (PK)
version	VARCHAR(50)	Verzija migracije
description	VARCHAR(200)	Opis migracije
type	VARCHAR(20)	Vrsta migracije (SQL/JDBC/...)
script	VARCHAR(1000)	Naziv / lokacija skripte
checksum	INTEGER	Kontrolna suma skripte
installed_by	VARCHAR(100)	Korisnik koji je pokrenuo migraciju
installed_on	TIMESTAMP	Vrijeme izvršenja migracije
execution_time	INTEGER	Vrijeme izvršavanja u ms
success	BOOLEAN	Status izvršavanja (TRUE/FALSE)

### Dijagram baze podataka

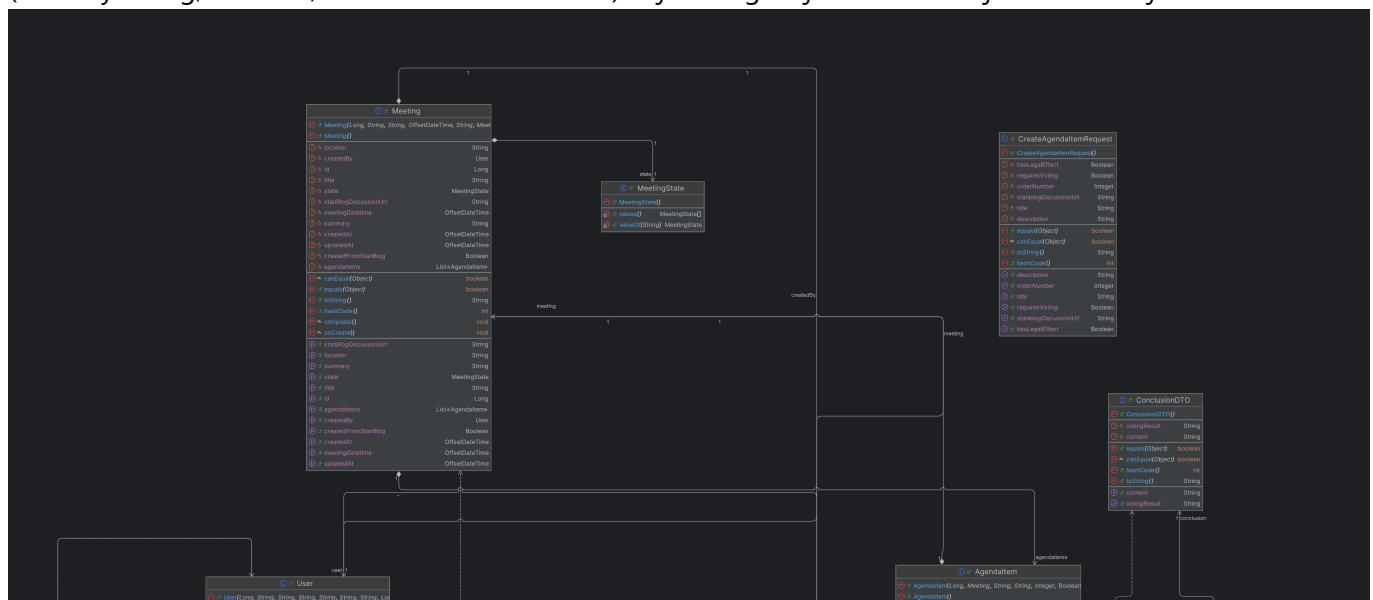


## ER model



## Dijagram razreda

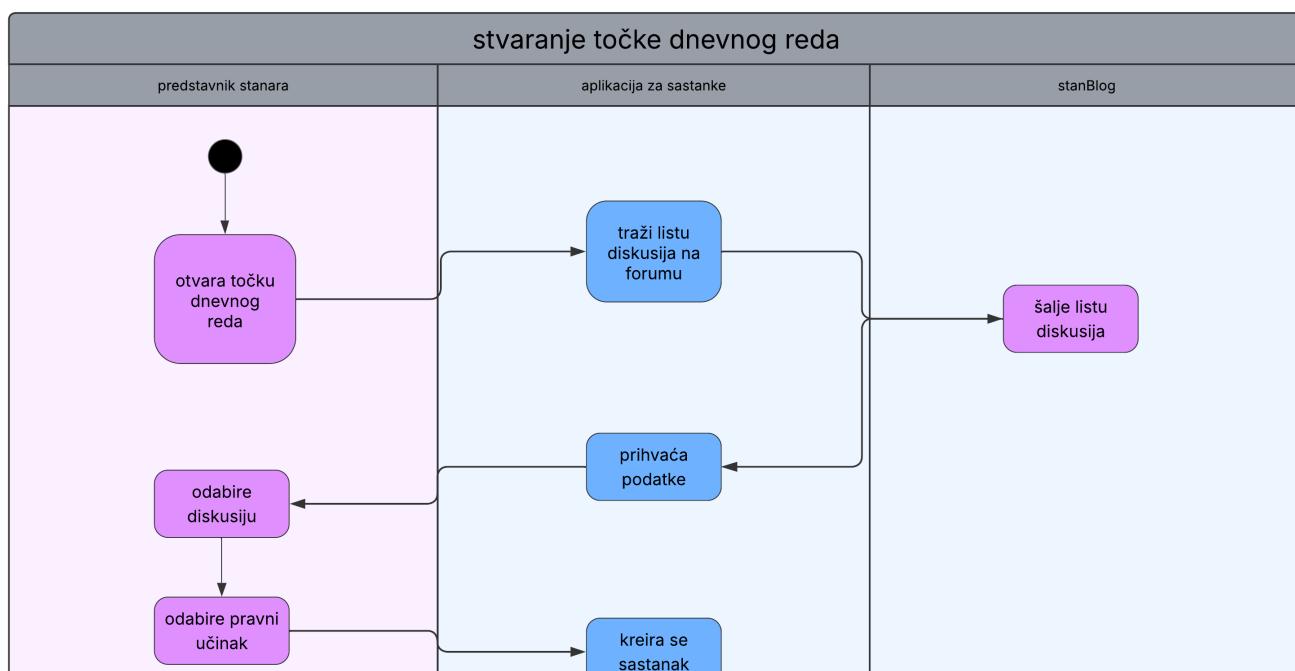
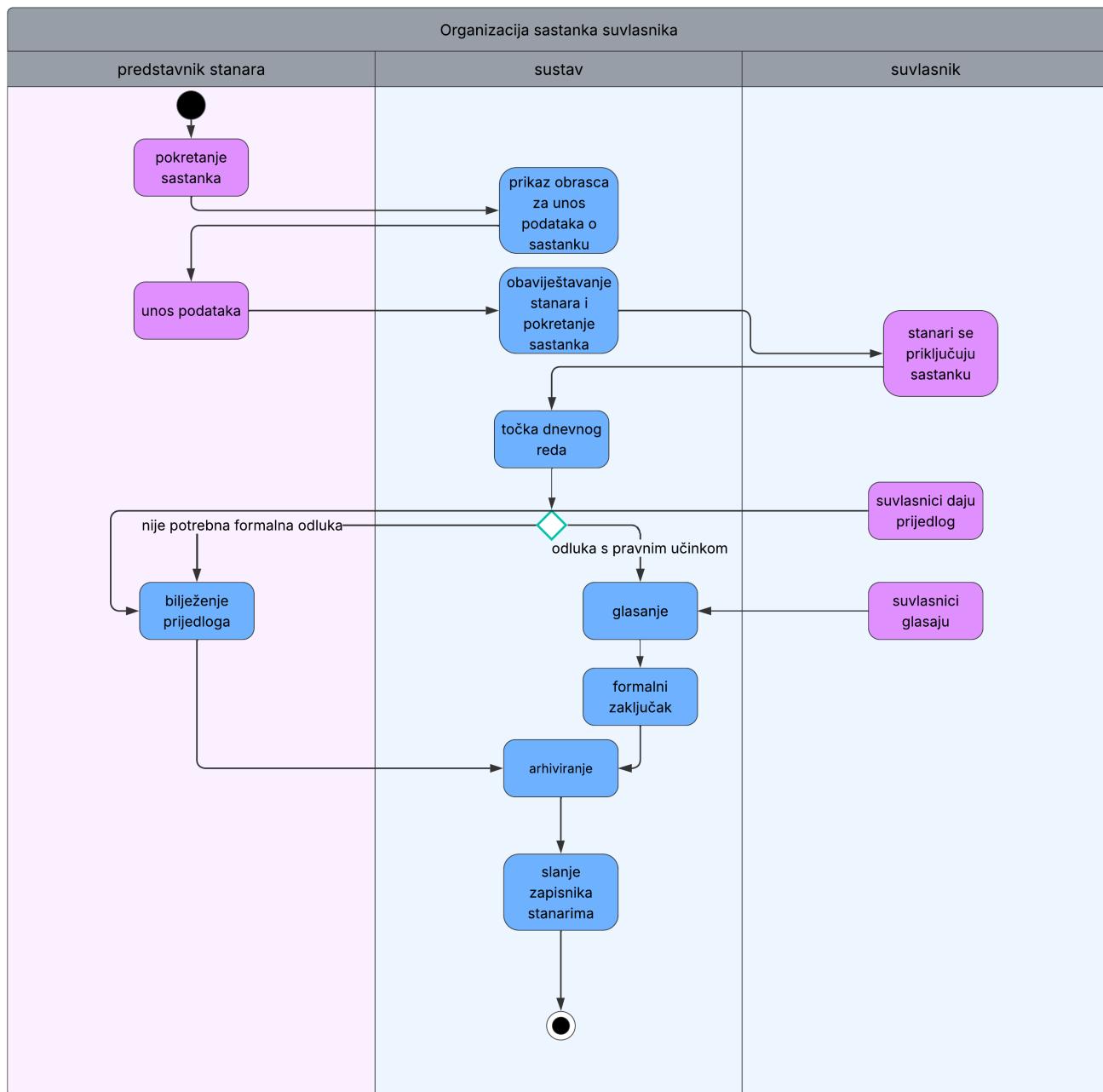
UML dijagram razreda na slici prikazuje cjelokupnu strukturu aplikacije temeljene na Spring Boot sustavu tj. backend dijelu projekta. Sustav je organiziran u više slojeva: modelni sloj obuhvaća entitete poput User, Meeting, Agendaltem i MeetingState koji predstavljaju osnovne podatkovne modele. DTO i request razredi (UserDTO, MeetingDTO, CreateMeetingRequest, RegisterRequest itd.) služe za prijenos podataka između različitih slojeva aplikacije. Servisni sloj (UserService, MeetingService, JwtService, OAuth2Service) implementira poslovnu logiku i koristi repozitorije (UserRepository, MeetingRepository) za pristup bazi podataka. Na vrhu se nalazi sloj razreda kontrolera (UserController, MeetingController, LoginController, SignUpController) koji obrađuje HTTP zahtjeve i komunicira sa servisnim slojem. Dijagram također uključuje sigurnosne komponente (SecurityConfig, JwtFilter, OAuth2SuccessHandler) koje omogućuju autentifikaciju i autorizaciju korisnika.

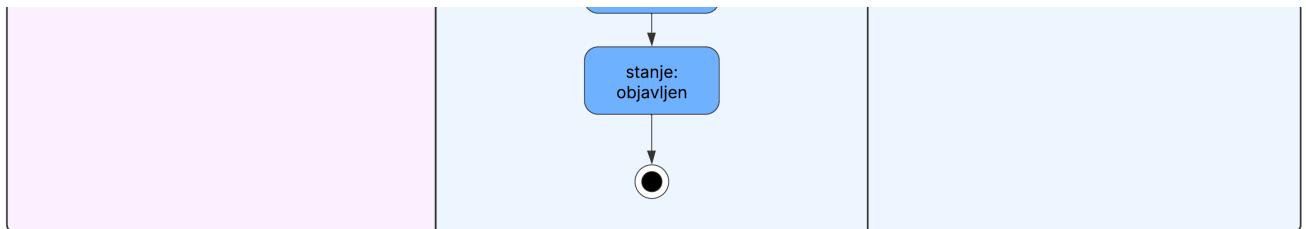




## Dinamičko ponašanje aplikacije







## UML dijagrami stanja

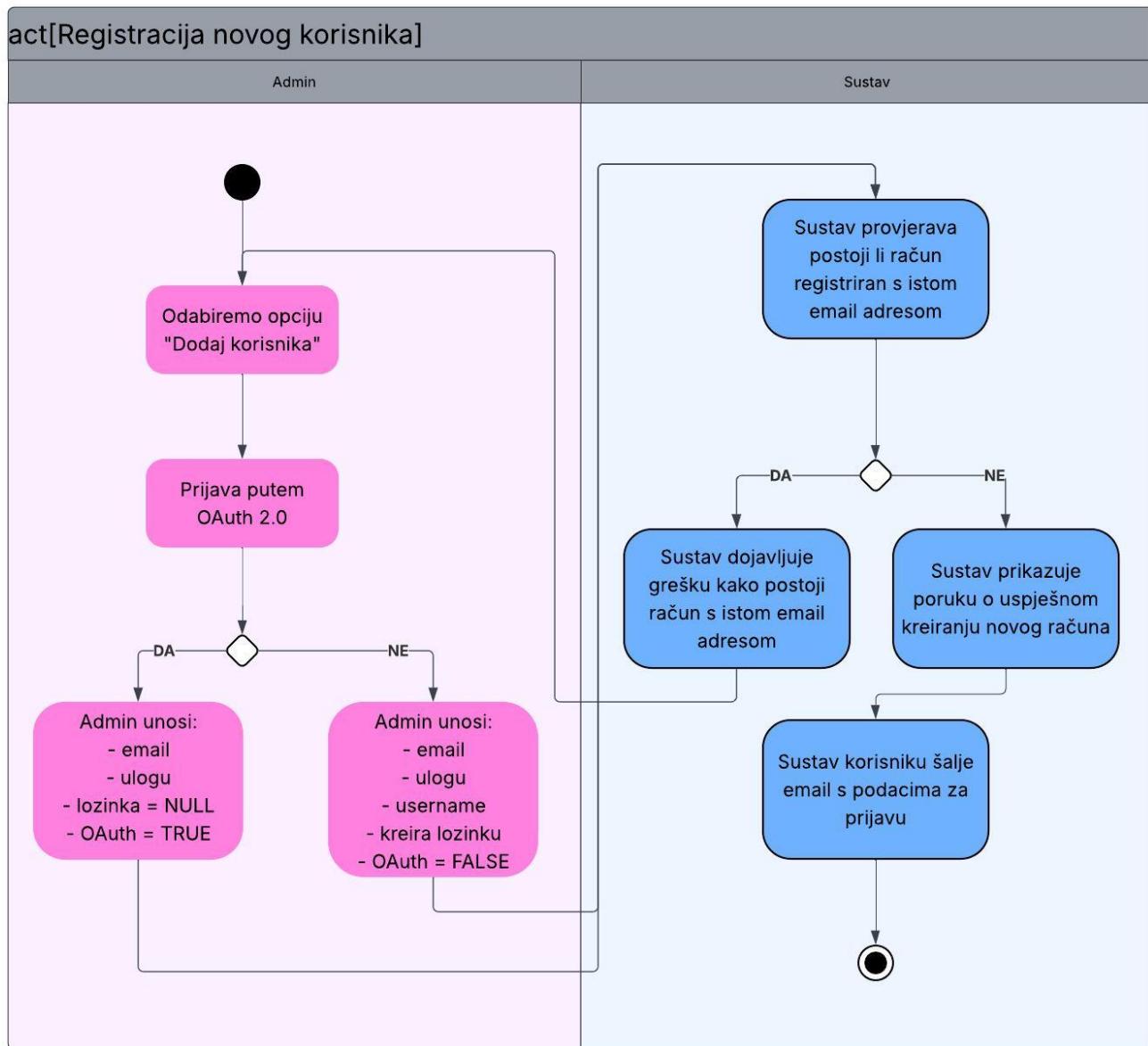
UML dijagrami stanja nužni su za razumijevanje dinamičkog ponašanja sustava. Oni jasno prikazuju promjene stanja objekata tijekom vremena ovisno o događajima i uvjetima.

**UML dijagram koji opisuje stanje sastanka**

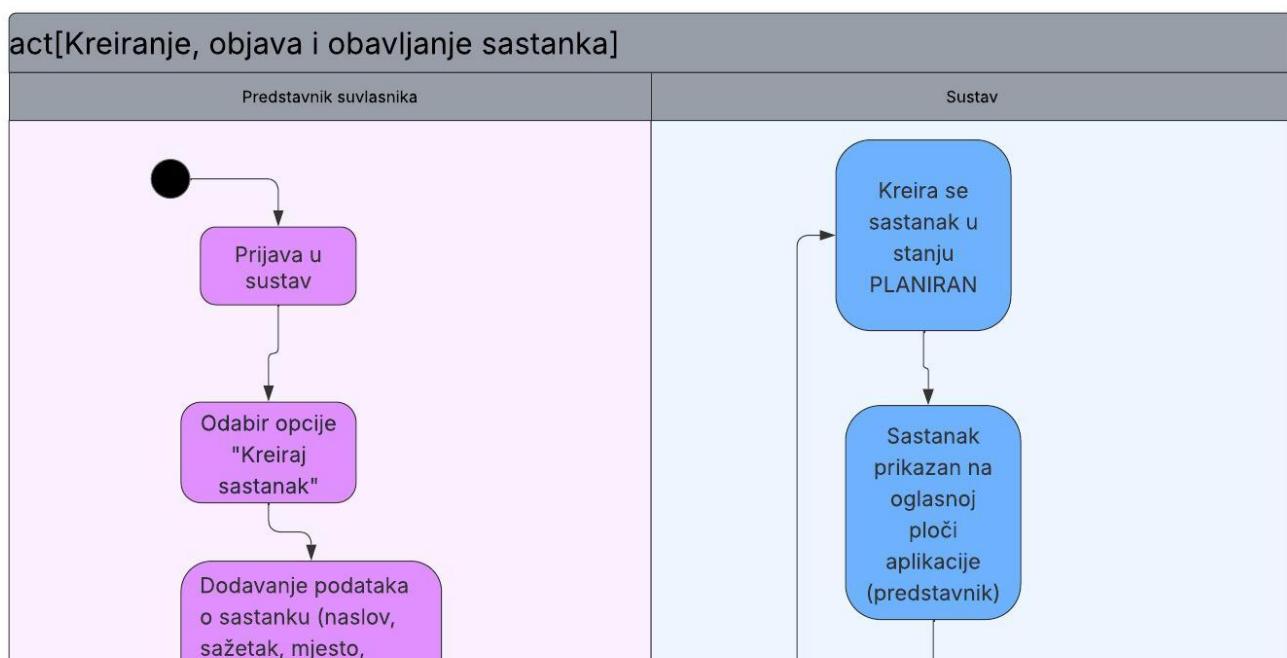


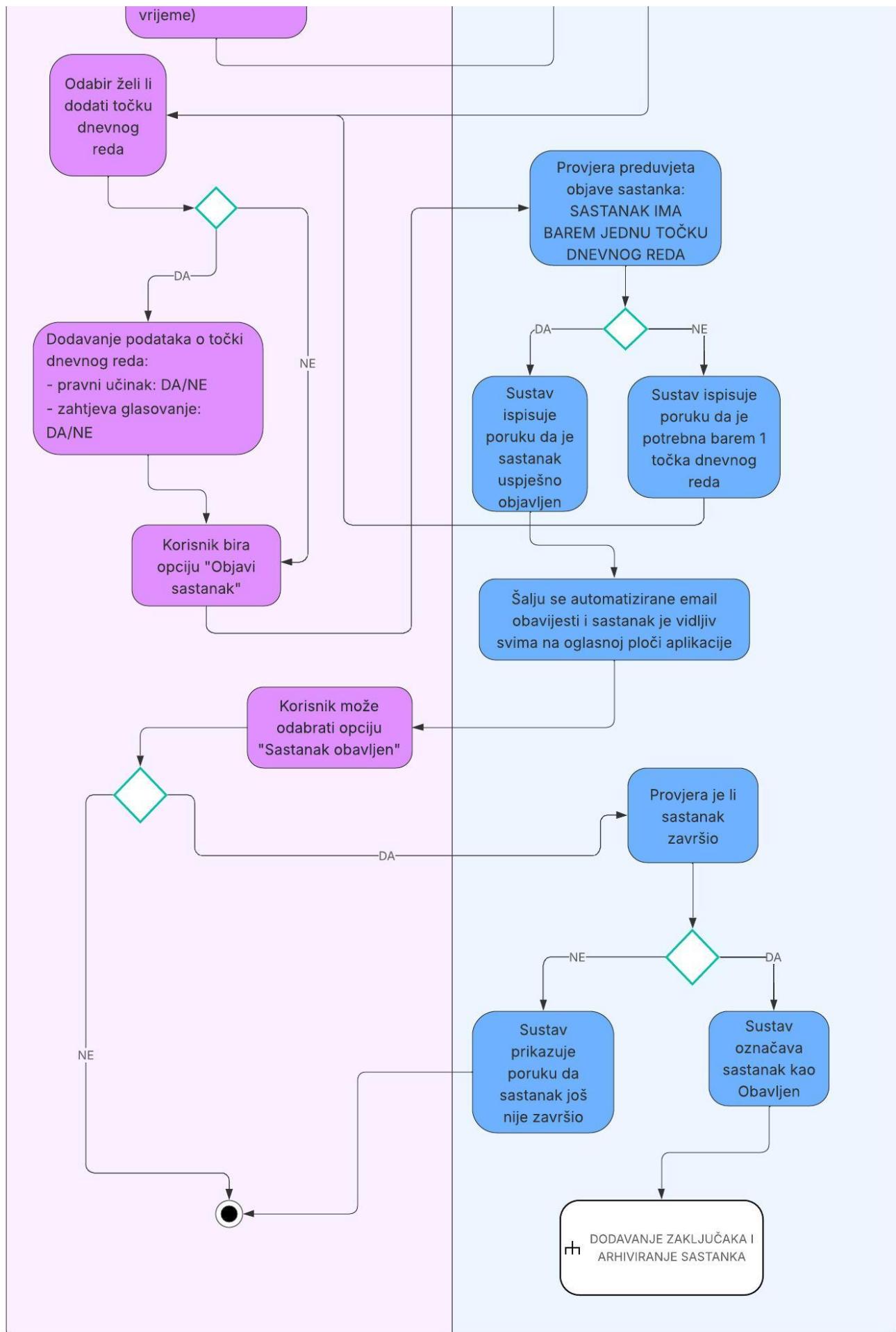
## UML dijagrami aktivnosti

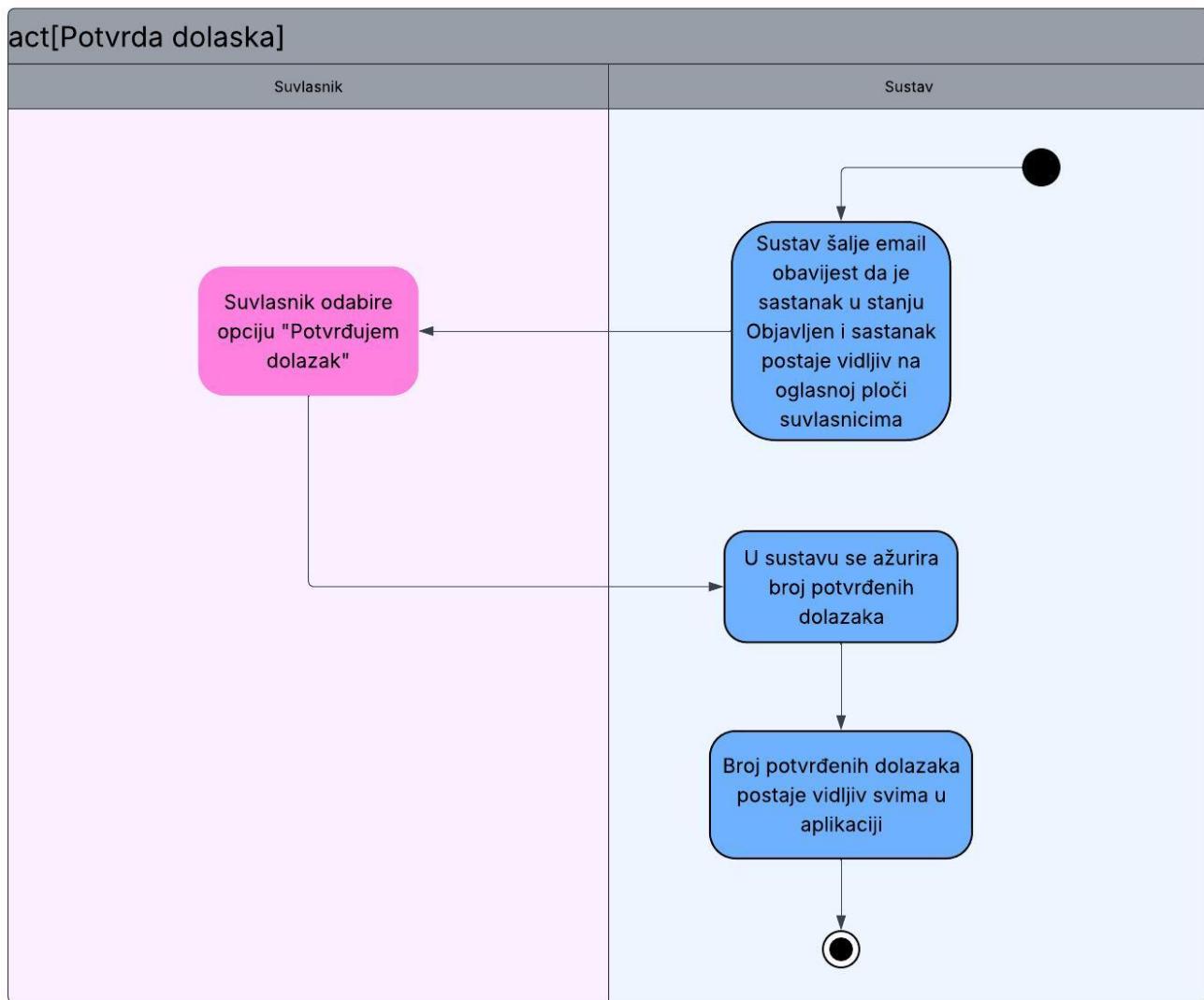
Registracija novog korisnika



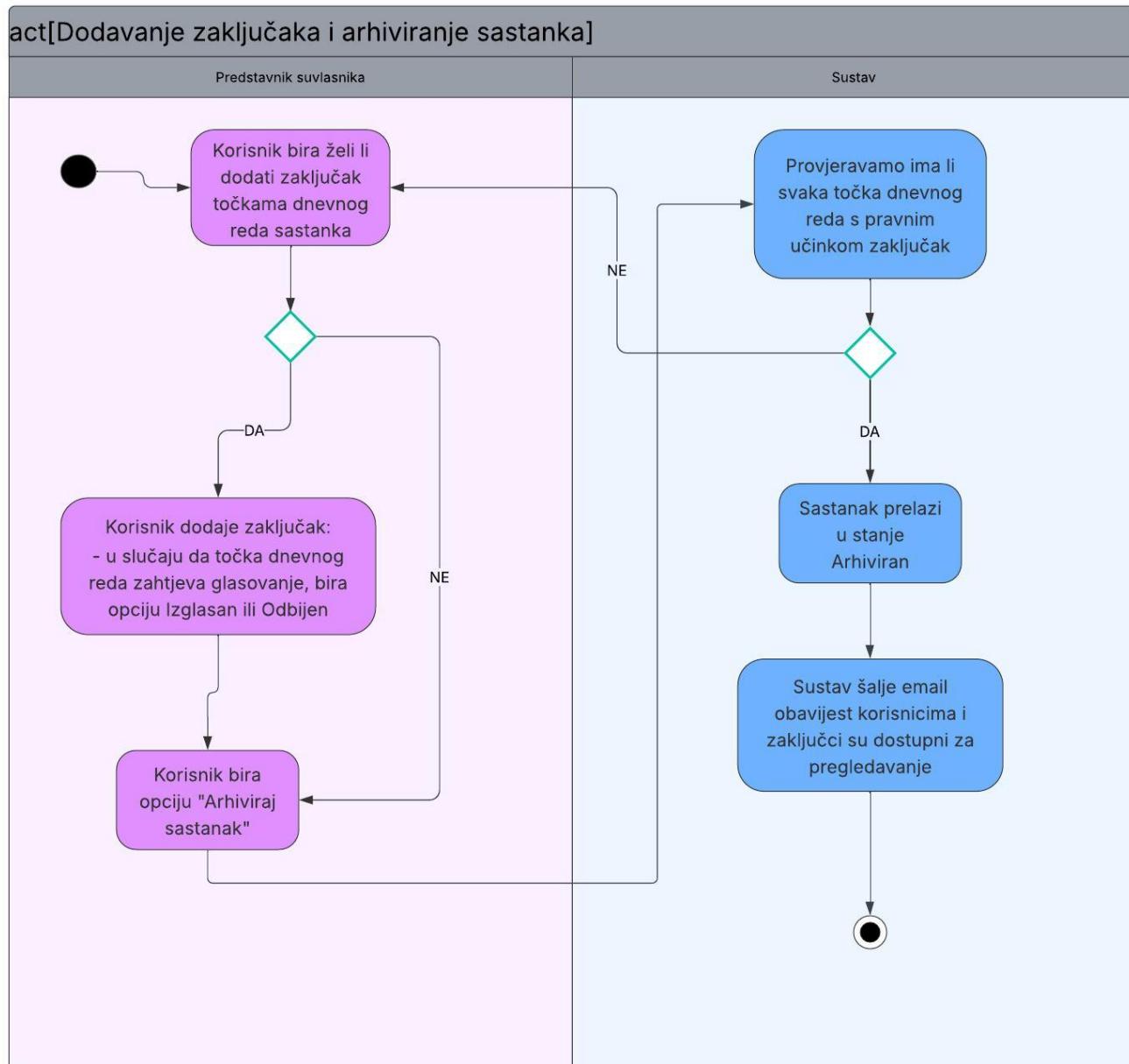
## Kreiranje, objava i obavljanje sastanka



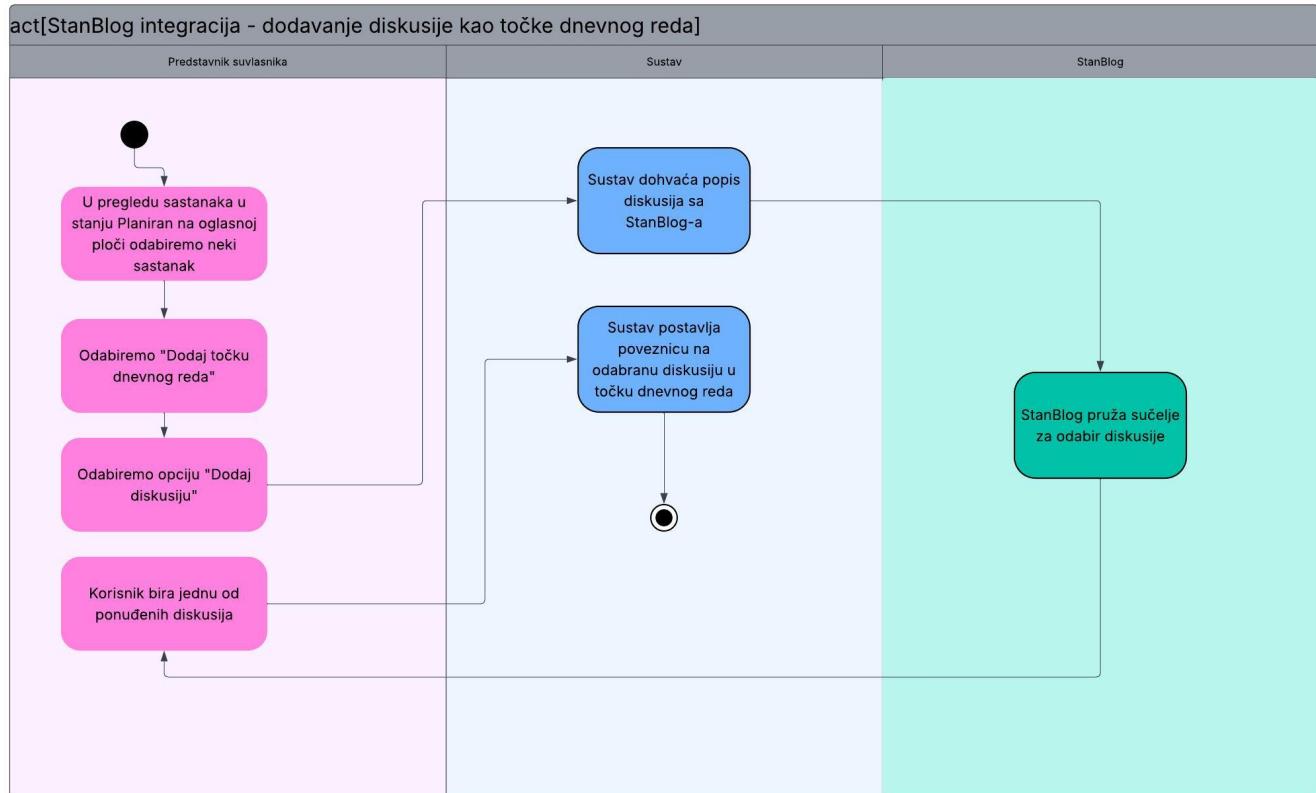




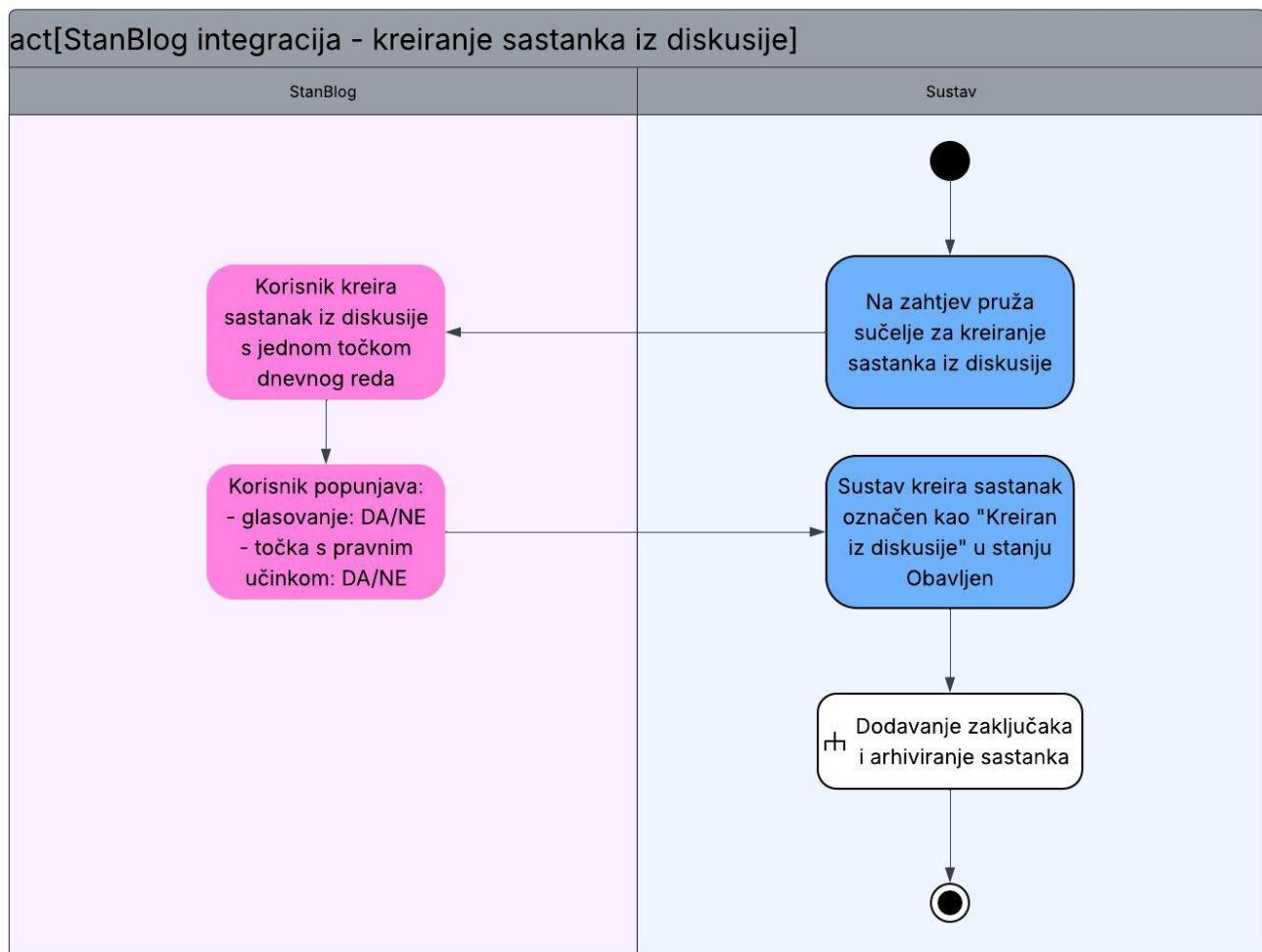
Dodavanje zaključaka i arhiviranje sastanka



StanBlog integracija - dodavanje diskusije kao točke dnevnog reda



### StanBlog integracija - kreiranje sastanka iz diskusije



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.

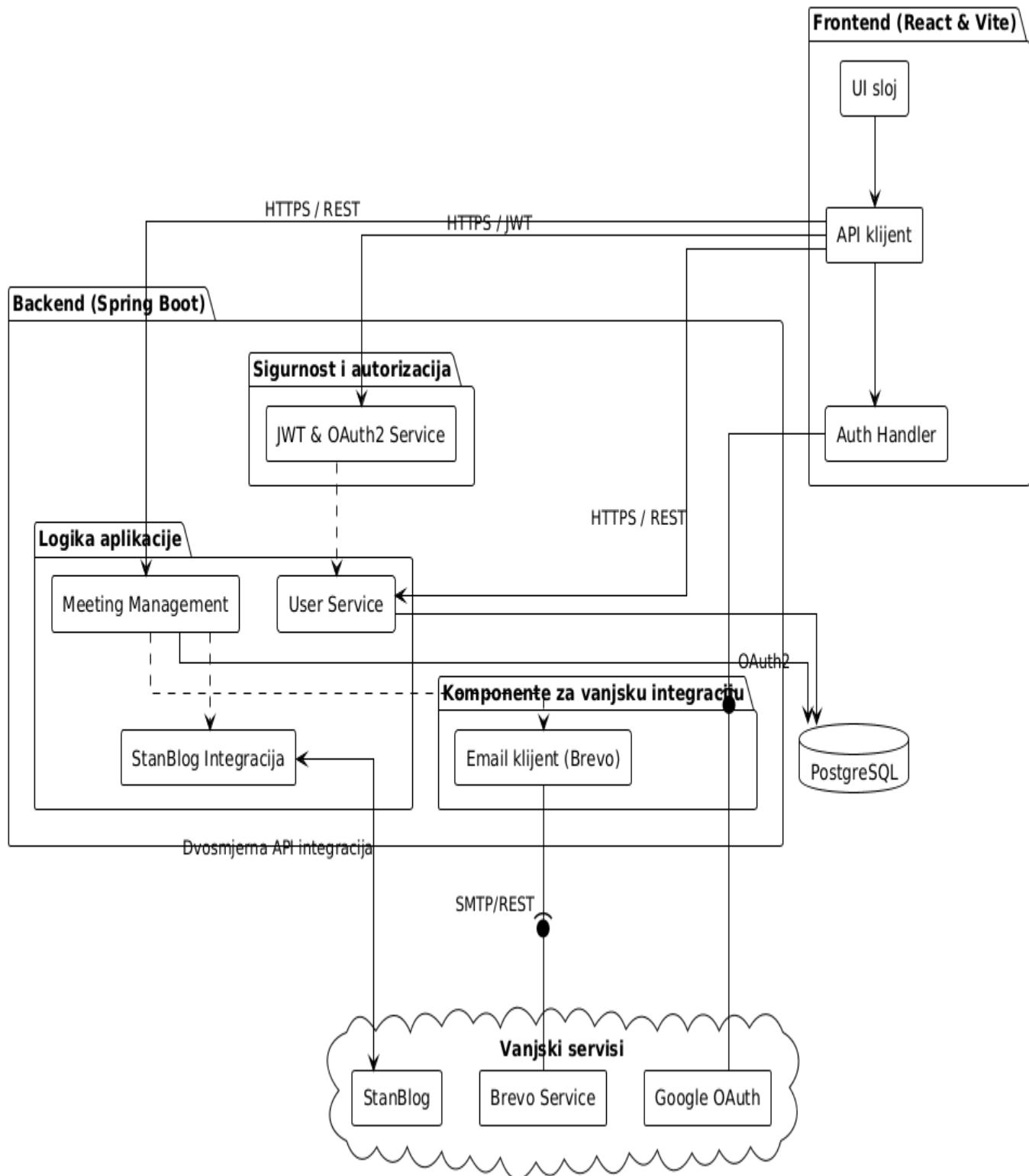
## Dijagram komponenata

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.

Dijagram komponenata prikazuje logičku organizaciju sustava StanPlan. Sustav je podijeljen na tri sloja: frontend, backend i vanjske servise. Ti su slojevi međusobno povezani standardnim komunikacijskim protokolima.

### Ključne komponente sustava:

- Frontend (React & Vite): Ovdje se nalazi UI sloj za prikaz podataka, API klijent za komunikaciju sa backendom i Auth Handler koji upravlja prijavom korisnika.
- Backend (Spring Boot):
  - Sigurnost + autorizacija: Upravlja zaštitom ruta pomoću JWT tokena i obrađuje OAuth2 odgovore.
  - Logika aplikacije: Središnji dio sustava koji upravlja fazama kroz koje prolazi sastanak (Meeting Management), korisnicima (User Service) i logikom za StanBlog integraciju.
  - Komponente za vanjsku integraciju: Upravljaju komunikacijom s vanjskim sustavima (Brevo)
- Baza podataka (PostgreSQL)
- Vanjski servisi:
  - Google OAuth: autentifikacija
  - Brevo: email obavijesti
  - StanBlog: sustav ostvaruje dvosmjernu API integraciju sa sustavom StanBlog



## Dijagram razmještaja

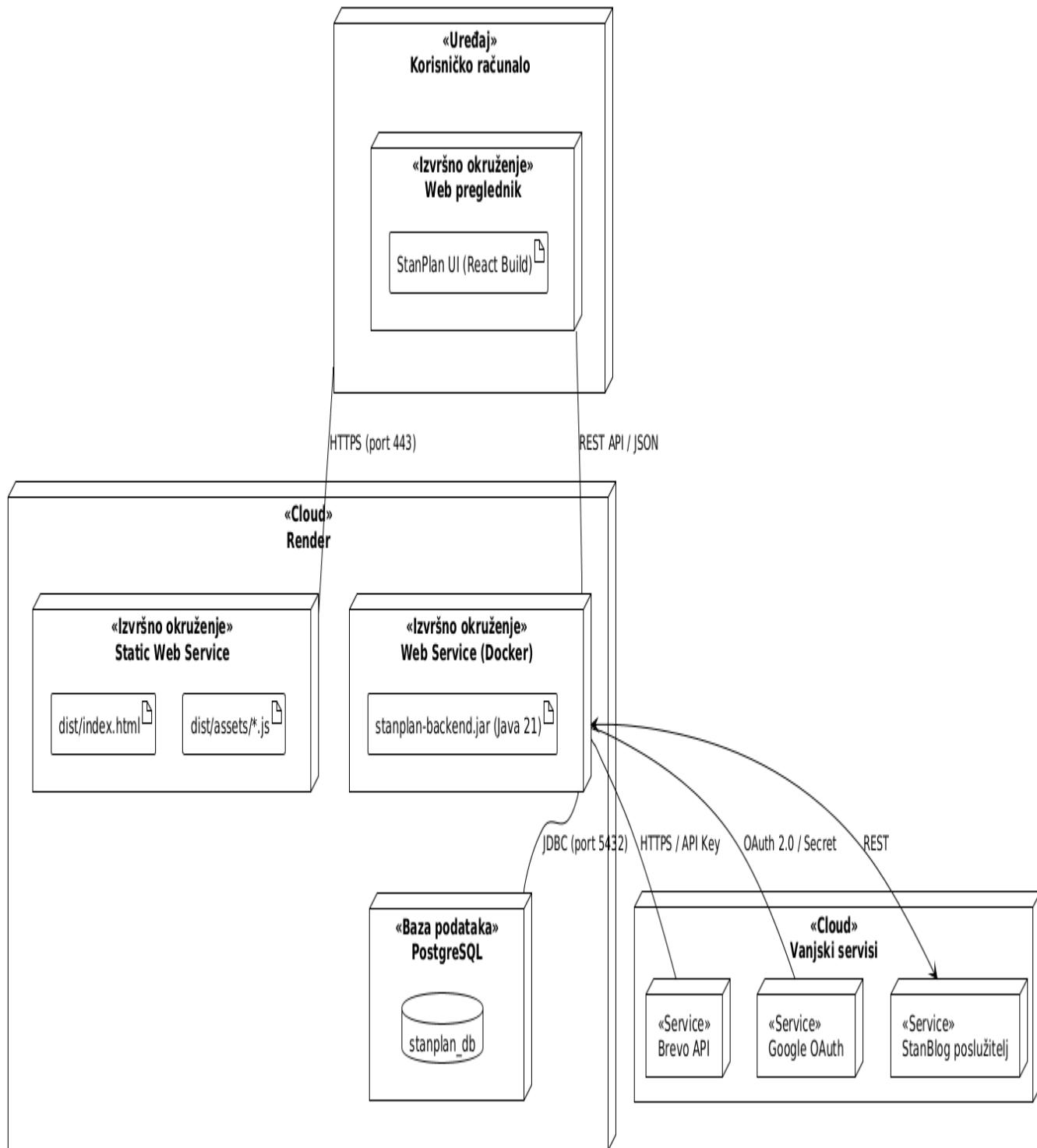
UML dijagram razmještaja prikazuje fizičku ili virtualnu raspodjelu komponenata sustava unutar infrastrukture. Cilj je prikazati kako su komponente raspoređene (npr. na poslužiteljima, u okruženjima oblaka ili na uređajima krajnjih korisnika) te način komunikacije, API-ja ili drugih komunikacijskih protokola.

## Implementacijski oblik

- **Klijentska strana (Korisničko računalo)**: Korisnici pristupaju aplikaciji putem web preglednika. Izvršno okruženje preglednika preuzima i pokreće StanPlan UI (React aplikaciju) koja putem HTTPS protokola

komunicira sa backendom.

- Render: Cijeli sustav hostan je na platformi Render. Platforma omogućuje izolaciju servisa:
  - Static Web Service: Upravlja distribucijom HTML i JS datoteka klijentu putem sigurne veze.
  - Docker: Backend dio aplikacije upakiran je u Docker kontejner unutar kojeg se izvodi Spring Boot aplikacija.
  - PostgreSQL (upravljeni): Render upravlja instancom baze podataka, a komunikacija se odvija unutar privatne mreže na portu 5432
- Integracija s vanjskim servisima:
  - Google OAuth: Koristimo za sigurnu autentifikaciju bez potrebe za pohranom lozinki u sustavu putem OAuth 2.0 protokola
  - Brevo API: Automatizirano šalje email obavijesti.
  - StanBlog: Ostvarujemo dvosmjernu komunikaciju sa sučeljem StanBlog putem REST API.



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

---

Korištene klase LoginControllerTest.java i MeetingControllerTest.java uploadane su u kod aplikacije (backend/src/test/java/hr/fer/susjedi/controller)

### 1. Uspješna prijava korisnika (redovni slučaj)

Opis funkcionalnosti

Provjera osnovne funkcionalnosti prijave korisnika s ispravnim podacima.

Ulazni podaci za postojećeg korisnika

- Email: marko@stanblog.com
- Lozinka: Marko123\$
- Korisničko ime: marko
- Uloga: suvlasnik

Očekivani rezultat

- HTTP status: 200 OK
- JSON odgovor s porukom "Prijava uspješna!", korisničkim imenom i JWT tokenom

Dobiveni rezultat

BUILD SUCCESS, primljen očekivani JSON objekt:

```
MockHttpServletResponse:  
    Status = 200  
    Error message = null  
    Headers = [Content-Type:"application/json"]  
    Content type = application/json  
    Body = {"message":"Prijava  
uspješna!","username":"marko","token":"token123"}
```

Postupak ispitivanja

U sustav je registriran korisnik (marko@stanblog.com) čiji se podaci koriste za ispitivanje.

Ispitivanje komponenata provedeno je u razvojnem okruženju korištenjem radnih okvira JUnit 5 i Mockito. Kreirana je LoginController klasa za ispitivanje. Postupak je započeo izolacijom LoginController klase od vanjskih ovisnosti poput baze podataka i sigurnosnih filtera, čime je fokus stavljen isključivo na provjeru logike same komponente.

Korištenjem alata MockMvc simulirani su HTTP POST zahtjevi s različitim testnim podacima koji su obuhvatili redovne slučajeve, rubne uvjete i scenarije izazivanja pogrešaka. Ispitivanje je zaključeno automatskom verifikacijom statusnih kodova i sadržaja JSON odgovora, čime je kroz Maven alat potvrđena stabilnost i ispravnost implementirane funkcionalnosti.

## 2. Prijava s neispravnom lozinkom (izazivanje pogreške)

### Opis funkcionalnosti

Provjera reakcije sustava na pogrešnu lozinku.

### Ulazni podaci za korisnika

- Email: marko@stanblog.com
- Lozinka: Mislav92%
- Korisničko ime: marko
- Uloga: suvlasnik

### Očekivani rezultat

HTTP status 200 OK i JSON odgovor s porukom "Pogrešna lozinka!"

### Dobiveni rezultat

BUILD SUCCESS, primljen očekivani JSON objekt:

```
MockHttpServletResponse:  
    Status = 200  
    Error message = null  
    Headers = [Content-Type:"application/json"]  
    Content type = application/json  
    Body = {"message":"Pogrešna lozinka!"}
```

### Postupak ispitivanja

Ispitivanje je provedeno dodavanjem nove testne metode unutar postojeće klase LoginControllerTest, čime je zadržana konzistentnost testnog okruženja. Postupak je obuhvatio simulaciju unosa ispravne e-mail adrese uz namjerno neispravnu lozinku kako bi se provjerila obrambena logika kontrolera.

## 3. Prijava s nepostojećim korisnikom (izazivanje pogreške)

### Opis funkcionalnosti

Provjera ponašanja sustava kada korisnik ne postoji u bazi.

### Ulazni podaci za nepostojećeg korisnika

- Email: trpimir@stanblog.com

- Lozinka: Mislav92%

### Očekivani rezultat

HTTP status 200 OK i JSON odgovor s porukom "Korisnik ne postoji!"

### Dobiveni rezultat

BUILD SUCCESS, primljen očekivani JSON objekt:

```
MockHttpServletResponse:  
    Status = 200  
    Error message = null  
    Headers = [Content-Type:"application/json"]  
    Content type = application/json  
    Body = {"message":"Korisnik ne postoji!"}
```

### Postupak ispitivanja

Ispitivanje je provedeno dodavanjem nove testne metode u klasu LoginControllerTest kojom se simulira unos nepostojećeg korisnika u bazu podataka kako bi se verificiralo ispravno rukovanje pogreškom i povratna JSON poruka.

---

## 4. Prijava s praznim poljima (rubni uvjet)

### Opis funkcionalnosti

Validacija kod prijave bez ulaznih podataka.

### Ulazni podaci za korisnika

- Email: [prazno]
- Lozinka: [prazno]

### Očekivani rezultat

HTTP status 200 OK i JSON odgovor s porukom "Korisnik ne postoji!"

### Dobiveni rezultat

BUILD SUCCESS, primljen očekivani JSON objekt:

```
MockHttpServletResponse:  
    Status = 200  
    Error message = null  
    Headers = [Content-Type:"application/json"]  
    Content type = application/json  
    Body = {"message":"Korisnik ne postoji!"}
```

## Postupak ispitivanja

Ispitivanje je provedeno dodavanjem nove testne metode u klasu LoginControllerTest kojom se simulira unos praznih polja kako bi se verificiralo ispravno rukovanje pogreškom i povratna JSON poruka.

## 5. Kreiranje sastanka s ispravnim podacima (redovni slučaj)

### Opis funkcionalnosti

Provjerava može li prijavljeni korisnik uspješno zakazati termin.

### Ulagani podaci

- Naslov: Sastanak stanara
- Opis: Dogovor o krovu
- Datum i vrijeme: 2026-06-01T18:00:00

### Očekivani rezultat

HTTP status 201 Created i povratni JSON objekt koji potvrđuje uspješno kreiranje sastanka.

### Dobiveni rezultat

```
MockHttpServletResponse:  
    Status = 201  
    Error message = null  
    Headers = [Content-Type:"application/json"]  
    Content type = application/json  
    Body = {"id":null,"title":"Sastanak  
stanara","summary":null,"meetingDatetime":null,"location":null,"state":null,"creat  
edByEmail":null,"createdAt":null,"agendaItemsCount":null,"agendaItems":null,"atten  
deeUsernames":null,"currentUserAttending":false}
```

## Postupak ispitivanja

Ispitivanje je provedeno u klasi MeetingControllerTest.java simulacijom slanja ispravno strukturiranog JSON objekta prema /api/meetings endpointu. Korištenjem Mockito okvira verificirano je da kontroler ispravno komunicira sa servisnim slojem.

## 6. Kreiranje sastanka bez naslova (rubni uvjet)

### Opis funkcionalnosti

Provjerava stabilnost sustava kada korisnik pokuša poslati prazan naslov sastanka.

### Ulagani podaci

Naslov nije unesen.

### Očekivani rezultat

HTTP status 400 Bad Request uz poruku "Naslov je obavezan"

### Dobiveni rezultat

```
MockHttpServletResponse:  
    Status = 400  
    Error message = Naslov je obavezan
```

### Postupak ispitivanja

Ispitivanje rubnog uvjeta provedeno je slanjem zahtjeva s praznim poljem za naslov i simuliranjem validacijske pogreške kako bi se verificiralo ispravno rukovanje neispravnim unosima.

## \*\*Ispitivanje sustava \*\*

---

Automatizirani testovi su izrađeni korištenjem Playwright alata.

### 1) Uspješna prijava korisnika u sustav (redovni uvjet)

#### Ulazi

- Email: marko@stanblog.com
- Lozinka: Marko123\$

#### Korišteni Playwright kod

```
test('Uspješna prijava', async ({ page }) => {  
    await page.goto('http://localhost:5173/#/');  
    await page.getByRole('textbox', { name: 'Email' }).fill('marko@stanblog.com');  
    await page.getByRole('textbox', { name: 'Lozinka:' }).fill('Marko123$');  
    await page.getByRole('button', { name: 'Prijavi se' }).click();  
  
    await expect(page.locator('h1')).toContainText('StanPlan');  
    await page.waitForTimeout(2000);  
});
```

#### Koraci ispitivanja

Playwright otvara URL stranice, pronalazi polja za unos maila i lozinke, popunjava ih danim podacima i pritišeće gumb za prijavu. Na kraju verificira da je uspješno došao na početnu stranicu aplikacije.

### Očekivani izlaz

Korisnik je preusmjeren na početnu stranicu nakon prijave.

## Dobiveni izlaz

The screenshot shows the Playwright Test runner interface. On the left, there's a tree view of test files and cases. A test named 'Uspješna prijava' is expanded, showing three sub-cases: 'Neispravna lozinka', 'Kreiran sastanak', and 'Nedostaje datum'. The 'Neispravna lozinka' case is collapsed. The 'Kreiran sastanak' case is expanded, showing steps like 'Navigate to "/"', 'Fill "marko@stanblog.com"', 'Fill "Marko123\$', 'Click', 'Expect "toContainText"', and 'Wait for timeout'. The 'Nedostaje datum' case is collapsed. The timeline at the top shows a 500ms delay followed by a 1.0s action. The test summary at the bottom left says '1/1 passed (100%)'. The test details table shows a 'Passed' status with a duration of 3.2s. The browser preview on the right shows the StanPlan application's home page with a blue header and a 'StanPlan' logo. Below the header, there are buttons for 'Prijavi se', 'Registruj (Admin)', and 'Odjavi se'. The main content area displays sections for 'Objavljeni sastanci', 'Moji sastanci (Planirani)', 'Sastanci za obradu (Obavljeni)', and 'Arhiva sastanaka'. At the bottom of the browser preview, it says 'No errors'.

## 2) Prijava korisnika s pogrešnom lozinkom (izazivanje pogreške)

### Ulazi

- Email: marko@stanblog.com
- Lozinka: Mirko123\$

### Korišteni Playwright kod

```
test('Neispravna lozinka', async ({ page }) => {
    await page.goto('http://localhost:5173/#/');
    await page.getByRole('textbox', { name: 'Email' }).fill('marko@stanblog.com');
    await page.getByRole('textbox', { name: 'Lozinka:' }).fill('Mirko123$');

    page.once('dialog', dialog => {
        console.log(`Poruka upozorenja: ${dialog.message()}`);
        dialog.dismiss().catch(() => {});
    });

    await page.getByRole('button', { name: 'Prijava se' }).click();
    await expect(page.getByRole('button', { name: 'Prijava se' })).toBeVisible();
});
```

## Koraci ispitivanja

Playwright otvara URL stranice, unosi ispravnu e-mail adresu i pogrešnu lozinku te pokušava prijavu. Na kraju verificira da je korisnik ostao na stranici za prijavu.

## Očekivani izlaz

Korisnik ostaje na login stranici nakon pokušaja prijave.

## Dobiveni izlaz

### 3) Predstavnik stanara kreira sastanak (redovni slučaj)

#### Ulazi

- Email: mirko@stanblog.com
- Lozinka: Mirko123\$
- Ime sastanka: sastanak
- Opis sastanka: sazetak
- Datum sastanka: 2026-01-01T01:01
- Lokacija sastanka: zgrada

#### Korišteni Playwright kod

```
test('Kreiran sastanak', async ({ page }) => {
  await page.goto('http://localhost:5173/');
  await page.getByRole('textbox', { name: 'Email' }).fill('mirko@stanblog.com');
```

```
await page.getByRole('textbox', { name: 'Lozinka:' }).fill('Mirko123$');
await page.getByRole('button', { name: 'Prijavi se' }).click();

await page.getByRole('button', { name: '✚ Kreiraj Sastanak' }).click();
await page.getByRole('textbox', { name: 'Naslov *' }).fill('sastanak');
await page.getByRole('textbox', { name: 'Sažetak *' }).fill('sazetak');
await page.getByRole('textbox', { name: 'Datum i vrijeme *' }).fill('2026-01-01T01:01');
await page.getByRole('textbox', { name: 'Lokacija *' }).fill('zgrada');

page.once('dialog', dialog => {
  console.log(`Dialog message: ${dialog.message()}`);
  dialog.dismiss().catch(() => {});
});

await page.getByRole('button', { name: 'Kreiraj Sastanak', exact: true }).click();
await expect(page.getText('Objavljeni sastanci')).toBeVisible();
});
```

## Koraci ispitivanja

Playwright prijavljuje korisnika, navigira do forme za kreiranje sastanka, popunjava sve potrebne podatke i potvrđuje unos.

## Očekivani izlaz

Korisnik je uspješno kreirao sastanak.

## Dobiveni izlaz

The screenshot shows the Playwright Test interface. On the left, there's a sidebar with a tree view of test files and cases. The main area has a timeline at the top showing the sequence of actions: 'Passed' (1.7s), 'Press "ArrowRight"' (15ms), 'Press "ArrowRight"' (16ms), 'Fill "2026-01-01T01:01"' (18ms), 'Click' (86ms), 'Fill "zgrada"' (15ms), 'Click' (94ms), and 'Expect "toBeVisible"' (15ms). Below the timeline, there's a screenshot of a web application window titled 'Kreiraj Novi Sastanak'. The window has fields for 'sazetak', 'Datum i vrijeme' (set to '01/01/2026 01:01 AM'), 'Lokacija' (set to 'zgrada'), and two buttons: 'Odustani' and 'Kreiraj Sastanak'. At the bottom right of the screenshot, it says 'No errors'.

#### 4) Predstavnik stanara kreira sastanak bez datuma (rubni slučaj)

##### Ulazi

- Email: mirko@stanblog.com
- Lozinka: Mirko123\$
- Ime sastanka: sastanak
- Opis sastanka: sazetak
- Datum sastanka: [prazno]
- Lokacija sastanka: zgrada

##### Korišteni Playwright kod

```
test('Nedostaje datum', async ({ page }) => {
  await page.goto('http://localhost:5173/#/');
  await page.getByRole('textbox', { name: 'Email' }).fill('mirko@stanblog.com');
  await page.getByRole('textbox', { name: 'Lozinka:' }).fill('Mirko123$');
  await page.getByRole('button', { name: 'Prijavi se' }).click();

  await page.getByRole('button', { name: '+ Kreiraj Sastanak' }).click();
  await page.getByRole('textbox', { name: 'Naslov *' }).fill('sastanak');
  await page.getByRole('textbox', { name: 'Sažetak *' }).fill('sazetak');
  await page.getByRole('textbox', { name: 'Lokacija *' }).fill('zgrada');

  await page.getByRole('button', { name: 'Kreiraj Sastanak', exact: true }).click();
```

```
await expect(page.locator('form')).toContainText('Datum i vrijeme su obavezni');
});
```

## Koraci ispitivanja

Playwright pokušava kreirati sastanak bez unosa datuma te verificira da sustav korisnika obavještava o obaveznom unosu datuma.

## Očekivani izlaz

Korisnik je obaviješten da nije unio datum sastanka.

## Dobiveni izlaz

The screenshot shows the Playwright Test runner interface. On the left, a tree view lists test results: 1/1 passed (100%) under the test file 'testovi.spec.js'. The test cases include 'Uspješna prijava' (2.9s), 'Neispravna lozinka' (468ms), 'Kreiran sastanak' (1.6s), and 'Nedostaje datum' (1.6s). The 'Nedostaje datum' test is currently selected. The main pane displays a timeline of actions: 'Passed' (1.6s), 'Before Hooks' (169ms), 'Navigate to "/"' (178ms), 'Click' (63ms), 'Fill "mirko@stanblog.com"' (19ms), 'Click' (39ms), 'Fill "Mirko123"' (11ms), 'Click' (40ms), 'Click' (213ms), and 'getByRole("button", { name: 'Kreiraj Sastanak' })'. A screenshot of the 'Kreiraj Novi Sastanak' dialog box is shown, with the date input field highlighted in red and the error message 'Datum i vrijeme su obavezni' visible. The bottom right corner of the interface shows the text 'No errors'.

# Korištene tehnologije i alati

---

Za razvoj klijentskog dijela aplikacije korišten je React (verzija 19.1), moderna JavaScript biblioteka za izgradnju korisničkih sučelja. React omogućuje napredno upravljanje stanjima i renderiranje putem komponenti, što sustav čini lakšim za održavanje. Uz React je korišten Vite (verzija 7.1) kao alat za izgradnju koji osigurava iznimno brz razvojni ciklus i optimizaciju proizvodnog koda.

Pozadinski dio sustava temelji se na Spring Boot radnom okviru (verzija 3.5.6) i jeziku Java (verzija 21 LTS), koristeći distribuciju Eclipse Temurin. Spring Boot je odabran jer omogućuje brz razvoj putem automatskih konfiguracija, dok Java 21 pruža vrhunske performanse i dugoročnu stabilnost. Za upravljanje projektom i automatizaciju izgradnje korišten je Maven (verzija 3.9.9), koji osigurava konzistentno upravljanje svim vanjskim bibliotekama.

Za pohranu i upravljanje podacima koristi se PostgreSQL (verzija 17.4), napredna objektno-relacijska baza podataka. PostgreSQL je odabran zbog svoje pouzdanosti te izvrsnih performansi pri radu s kompleksnim upitima i relacijama.

Za upravljanje izvornim kodom korišten je Git (verzija 2.45.1), dok je razvojni proces olakšan korištenjem IntelliJ IDEA (verzija 2024.1.1).

Komunikacija s korisnicima putem elektroničke pošte integrirana je pomoću Brevo API (v3) usluge, koja osigurava pouzdanu dostavu transakcijskih poruka.

Aplikacija je hostana na cloud platformi Render. Integracija s GitHub rezervorijem omogućuje automatski deployment, čime se svaka izmjena u kodu trenutno i bez prekida rada primjenjuje na proizvodnjo okruženje.

Za osiguranje kvalitete i stabilnosti pozadinskog dijela aplikacije korišten je JUnit (verzija 6.0.2), vodeći radni okvir za testiranje u Java ekosustavu. JUnit omogućuje pisanje ponovljivih jediničnih testova koji osiguravaju da svaki pojedini dio koda radi ispravno prije nego što se integrira u ostatak sustava.

Ispitivanje korisničkog sučelja provodi se pomoću alata Playwright (verzija 1.57). Playwright omogućuje automatizirano testiranje aplikacije simulirajući stvarne akcije korisnika poput prijave, navigacije i unosa podataka. Njegova sposobnost brzog izvršavanja i pouzdanog čekanja na učitavanje elemenata čini ga ključnim alatom za provjeru ispravnosti funkcionalnosti na klijentskoj strani.

# Instalacija, konfiguracija, pokretanje i administracija aplikacije

## 1. Instalacija

- **Preduvjeti:**

- Java: JDK 17 ili noviji.
- Baza podataka: PostgreSQL 15 ili noviji.
- Maven: Za izgradnju (build) aplikacije.
- Brevo račun: Za slanje transakcijskih emailova (API Key v3).

- **Preuzimanje:**

```
git clone https://github.com/filip-matijevic/susjedi.git  
cd susjedi/backend
```

### Instalacija ovisnosti:

Backend koristi Maven za upravljanje ovisnostima. Kada prvi put pokrenete projekt ili dodate nove biblioteke (poput Brevo/Sendinblue SDK-a), pokrenite sljedeću naredbu unutar /backend direktorija:

```
mvn clean install -DskipTests
```

Frontend koristi npm (Node Package Manager). Potrebno je ući u /frontend direktoriju i instalirati pakete definirane u package.json:

```
npm install
```

## 2. Postavke

Glavna konfiguracijska datoteka nalazi se na putanji: backend/src/main/resources/application.properties  
Ova datoteka sadrži reference na environment varijable koje trebate postaviti unutar IDE-a (IntelliJ/Eclipse).

.env datoteka unutar mape backend:

- BACKEND\_URL=http://localhost:8080
- DATABASE\_PASSWORD={sifra\_baze\_podataka}
- DATABASE\_URL=jdbc:postgresql://localhost:5432/stanplan
- DATABASE\_USERNAME=postgres
- FRONTEND\_URL=http://localhost:5173
- GOOGLE\_CLIENT\_ID={vas\_google\_client\_id}
- GOOGLE\_CLIENT\_SECRET={vas\_google\_client\_secret}
- JWT\_EXPIRATION=86400000
- JWT\_SECRET={vas\_jwt\_secret}
- MAIL\_PASSWORD={vas\_brevo\_api\_kljuc}

.env dadoteka unutar mape frontend:

- VITE\_API\_URL=http://localhost:8080

### Postavke baze podataka

Aplikacija koristi Flyway za automatsko upravljanje shemom baze podataka. Nije potrebno ručno pokretati SQL skripte za kreiranje tablica.

**Inicijalizacija:** Kreirajte praznu bazu podataka pod nazivom stanplan u vašem PostgreSQL sučelju (npr. pgAdmin ili psql).

**Migracije:** Prilikom svakog pokretanja aplikacije, Flyway provjerava mapu src/main/resources/db/migration i automatski izvršava sve nove .sql skripte.

### 3. Pokretanje aplikacije

Upute za pokretanje aplikacije u različitim okruženjima:

#### Razvojno okruženje:

Za razvoj je potrebno pokrenuti oba poslužitelja istovremeno:

1. Pokretanje Backenda (Spring Boot): Uđite u direktorij backend i pokrenite:

```
./mvnw spring-boot:run
```

Alternativno: Pokrenite SusjediApplication.java izravno iz IntelliJ IDEA-e uz prethodno postavljene Environment varijable.

2. Pokretanje Frontenda (Vite/React): Uđite u direktorij frontend i pokrenite:

```
npm run dev
```

#### Producčijsko okruženje:

U produkciji se aplikacija prvo mora izgraditi u izvršne datoteke, a zatim pokrenuti.

1. Prevođenje aplikacije (Build):

- Backend: Generira se .jar datoteka:

```
./mvnw clean package -DskipTests
```

- Frontend: Generiraju se statičke datoteke (HTML/JS/CSS) u mapi dist:

```
npm run build
```

2. Pokretanje poslužitelja (Start):

- Backend: Pokreće se Java virtualna mašina s generiranim JAR-om:

```
java -jar target/susjedi-0.0.1-SNAPSHOT.jar
```

- Frontend: Na Render platformi (Static Site), poslužitelj se pokreće automatski serviranjem sadržaja iz dist mape. Lokalno se produčijski build može testirati naredbom:

```
npm run preview
```

## Provjera rada:

- Backend API: <http://localhost:8080>
- Frontend sučelje: <http://localhost:5173>

## 4. Upute za administratore

Smjernice za administratore aplikacije nakon puštanja u pogon:

### Pristup administratorskom sučelju:

- Inicijalni podatci za pristup računa admina su:
- Email: admin@stanplan.com
- Lozinka: Admin123!

### Redovito održavanje i rješavanje problema

1.Pregled logova: Na platformi Render, logovi su dostupni u stvarnom vremenu pod tabom "Logs" na dashboardu usluge. U razvojnom okruženju, logovi se ispisuju u konzolu

2.Nadgledanje emailova: Administrator treba povremeno provjeravati Brevo Dashboard kako bi osigurao da kvota emailova nije potrošena i da nema velikog broja odbijenih adresa.

3.Ažuriranje aplikacije: Budući da Render koristi kontinuiranu isporuku, svako guranje koda na main granu GitHub repozitorija automatski pokreće re-build i re-deploy.

Lokalno: povlačenje novih verzija iz Git repozitorija i ponovno pokretanje aplikacije

```
git pull origin main
npm install
npm run build
npm start
```

---

## 5. Primjer za Render platformu (Cloud Deploy)

- Render je popularna cloud platforma za jednostavno smještanje aplikacija.

### Priprema repozitorija:

- Osigurajte da vaš projekt ima datoteku Dockerfile za konfiguraciju deploja.
- Primjer Dockerfile:

```
FROM maven:3.9.9-eclipse-temurin-21
WORKDIR /app
COPY backend .
RUN mvn clean package -DskipTests
EXPOSE 8080
CMD ["java", "-jar", "target/susjedi-0.0.1-SNAPSHOT.jar"]
```

### Postavljanje na Render:

### Baza podataka:

- Kliknite New + -> PostgreSQL.

- Ime: stanplan-db.
- Nakon kreiranja, kopirajte varijable baze podataka u environment varijable backend servisa.

### Backend servis:

- Kliknite New + -> Web Service.
- Runtime: Docker (ako imate Dockerfile)
- Build Command: ./mvnw clean install -DskipTests
- Start Command: java -jar target/\*.jar
- **Environment varijable:**
  - BACKEND\_URL=https://vas-link-za-backend.onrender.com
  - DATABASE\_PASSWORD={sifra-baze-podataka}
  - DATABASE\_URL=jdbc:postgresql://{vas-hostname}:5432/{ime-baze-podataka}
  - DATABASE\_USERNAME={korisnicko-ime-baze-podataka}
  - FRONTEND\_URL=https:// vas-link-za-frontend.onrender.com
  - GOOGLE\_CLIENT\_ID={vas\_google\_client\_id}
  - GOOGLE\_CLIENT\_SECRET={vas\_google\_client\_secret}
  - JWT\_EXPIRATION=86400000
  - JWT\_SECRET={vas\_jwt\_secret}
  - MAIL\_PASSWORD={vas\_brevo\_api\_kljuc}
  - JAVA\_TOOL\_OPTIONS="-Djava.net.preferIPv4Stack=true"

### Frontend servis:

- Kliknite New + -> Static Site.
- Povežite isti GitHub repozitorij.
- Build Command: npm install && npm run build
- Publish Directory: dist
- Environment Variables: VITE\_API\_URL = https://vas-link-za-backend.onrender.com

### Brevo API:

Potreban je Brevo račun za uspješno slanje email-ova. Napravite račun i napravite novi API ključ i upišite ga na MAIL\_PASSWORD.

## Opis pristupa aplikaciji na javnom poslužitelju

---

### Pristup aplikaciji

- Aplikacija StanPlan (Susjedi) hostana je na platformi Render, koja omogućuje pristup putem javnih URL adresa.
- Pristup aplikaciji putem preglednika
- Korisnici mogu pristupiti aplikaciji na sljedećoj adresi:
- URL aplikacije: <https://susjedi.onrender.com/>

### Pristup administratorskom sučelju

- Prijava: Odite na početnu stranicu
- Podaci za pristup: Unesite administratorske vjerodajnice:
- Email: admin@stanplan.com

- Lozinka: Admin123!
- Administrativne opcije: Nakon uspješne prijave, korisniku s ulogom ADMIN biti će omogućena kreacija novih računa.

### **StanBlog integracija**

Za sastanke napravljene putem StanBoga potrebno je prijaviti se na inicijalni račun s ulogom PREDSTAVNIK:

-Email: rep@stanplan.com

-Lozinka: rep123

### **Ograničenja aplikacije na javnom poslužitelju**

- Budući da se aplikacija nalazi na besplatnom paketu platforme Render i servisa Brevo, postoje određena ograničenja:
- "Hladni start": Ako aplikacija nije bila korištena neko vrijeme, Render gasi instancu. Prilikom prvog pristupa nakon pauze, može proći 30 do 50 sekundi dok se poslužitelj ponovno pokrene.
- Ograničenje emailova: Koristimo Brevo besplatni plan koji dopušta slanje maksimalno 300 emailova dnevno. Ako se taj limit prijeđe, sustav će prestati slati obavijesti do idućeg dana.
- Baza podataka: Besplatna baza podataka na Renderu automatski se briše nakon 30 dana ako se ne nadogradi na plaćeni plan.

Projekt smo realizirali u nekoliko sprintova. Ovakav pristup omogućio nam je da se u određenim razdobljima potpuno fokusiramo na specifične probleme. Tijekom rada na projektu najznačajniji izazov bio je komunikacijski. U ranim fazama suočili smo se s poteškoćama u podjeli odgovornosti i jasnom definiranju sučelja između različitih dijelova sustava, no te smo prepreke na kraju uspješno prevladali.

Stekli smo dublje razumijevanje interakcije između frontend i backend sloja, znanje o optimizaciji upita u radu s bazom podataka radi osiguravanja integriteta, te praktično iskustvo postavljanja aplikacije na poslužitelj. Upravo se deployment pokazao kao jedan od najkritičnijih koraka u prelasku iz razvojnog u produkcijsko okruženje. Iskustvo u ovim područjima bit će nam iznimno važno za brže ostvarivanje sličnih zadataka u budućnosti.

Glavni tehnički izazov bio je osigurati sinkronizaciju podataka između aplikacija StanPlan i StanBlog. Problemi s CORS restrikcijama i autentifikacijom riješeni su pravilnom konfiguracijom middleware-a na backendu i testiranjem funkcionalnosti u suradnji s drugom grupom. U konačnici, implementirane su sve zadane funkcionalnosti.

1. Programsko inženjerstvo, FER ZEMRIS, <http://www.fer.hr/predmet/proinz>
2. The Unified Modeling Language, <https://www.uml-diagrams.org/>
3. Astah Community, <http://astah.net/editions/uml-new>
4. PlantUML guide, <https://plantuml.com/>
5. Spring Boot Tutorial, <https://www.geeksforgeeks.org/advance-java/spring-boot/>
6. React Docs, <https://legacy.reactjs.org/docs/getting-started.html>
7. Web Development guide, [https://developer.mozilla.org/en-US/docs/Learn\\_web\\_development](https://developer.mozilla.org/en-US/docs/Learn_web_development)
8. Git Docs, <https://git-scm.com/docs/gittutorial>
9. Github Docs, <https://docs.github.com/en/get-started/start-your-journey/hello-world>

# Dnevnik sastajanja

---

## 1. sastanak

- Datum: 9. listopada 2025.
- Prisustvovali: P. Ćurić, A. Gabaj, M. Glavaš, A. Ivančić, F. Matijević, J. Schramadei
- Teme sastanka:
  - Upoznavanje i odabir voditelja
  - LOŠE: podjela uloga
  - BOLJE: uspostavljena komunikacija

## 2. sastanak

- Datum: 23. listopada 2025.
- Prisustvovali: P. Ćurić, A. Gabaj, M. Glavaš, A. Ivančić, F. Matijević
- Teme sastanka:
  - Podjela uloga za pisanje dokumentacije
  - LOŠE: dogovor načina komunikacije
  - BOLJE: podjela uloga za razvoj projekta
- poveznica na issue:** Oblikovanje arhitekture sustava, planiranje implementacije #7

## 3. sastanak

- Datum: 11. prosinca 2025.
- Prisustvovali: P. Ćurić, A. Gabaj, M. Glavaš, A. Ivančić, F. Matijević, J. Schramadei
- Teme sastanka:
  - Osvrt na rezultate prve predaje i razgovor o dalnjim ciljevima
  - LOŠE: propusti u dokumentaciji
  - BOLJE: dobra podloga za daljnji razvoj projekta

## 4. sastanak

- Datum: 15. siječnja 2026.
- Prisustvovali: P. Ćurić, A. Gabaj, M. Glavaš, A. Ivančić, F. Matijević, J. Schramadei
- Teme sastanka:
  - Utvrđivanje napravljenih zadataka prije posljednje predaje
  - LOŠE: podjela uloga za razvoj projekta
  - BOLJE: ostvarivanje ciljeva

## 5. sastanak

- Datum: 23. siječnja 2026.
- Prisustvovali: P. Ćurić, A. Gabaj, M. Glavaš, A. Ivančić, F. Matijević, J. Schramadei
- Teme sastanka:
  - Osvrt na završnu inačicu projekta i dogovor o posljednjim prepravcima dokumentacije. Osvrt na period od mjesec dana prije predaje.
  - LOŠE: razlike između angažmana članova tima, loša raspodjela uloga unutar tima

- BOLJE: projekt isporučen na vrijeme s dovoljno vremena za eventualne prepravke

# Plan rada

---

Tablični/Gantt/Kanban prikaz <https://github.com/users/f-matijevic/projects/1>

## Tablica aktivnosti

---

	Petar Ćurić	Andro Gabaj	Matej Glavaš	Ante Ivančić	Filip Matijević	Jakov Schramadei
Upravljanje projektom					8	4
Opis projektnog zadatka					1	
Funkcionalni zahtjevi			1		2	
Opis pojedinih obrazaca				1		
Dijagram obrazaca					4	
Sekvensijski dijagrami					8	
Opis ostalih zahtjeva	1		2			
Arhitektura i dizajn sustava		8	2			
Baza podataka			2		2	
Dijagram razreda			2			
Dinamički dijagrami		2.5				
Dijagram stanja		0.5				
Dijagram aktivnosti				2		
Dijagram komponenti					2	
Korištene tehnologije i alati			1			
Ispitivanje programskog rješenja		8				
Dijagram razmještaja					2	
Upute za puštanje u pogon			2			
Dnevnik sastajanja			0.5		0.5	
Zaključak i budući rad			0.25			
Popis literature					0.5	
Izrada aplikacije - dodatne stavke						

	Petar Ćurić	Andro Gabaj	Matej Glavaš	Ante Ivančić	Filip Matijević	Jakov Schramadei
- izrada početne stranice	6				4	
- izrada baze podataka					3	
- spajanje s bazom podataka			3		1	
- kreacija prijave i registracije	2		4			
- izrada OAuth2			4		2	
- kreacija mogućnosti dodavanja sastanaka			2		3	
- mogućnost promjene zaporke			3			
- dodavanje točaka dnevnog reda			3			
- objavljivanje sastanaka			3			
- mogućnost potvrde prisutnosti na sastanku			3			
- sastanak prelazi u stanje OBAVLJEN			3			
- dodavanje zaključaka			3			
- arhiviranje sastanaka			3			
- StanBlog integracija			5		15	
- izrada prezentacije				10		
DevOps - dodatne stavke						
- arhitektura razmještaja			5		1	4
- kontinuirana integracija i isporuka			10		5	
- postavljanje aplikacije na poslužitelj			11			

## Dijagram pregleda promjena



## Kjučni izazovi i rješenja

Kao najveći izazov izdvojili bismo organizaciju unutar tima i raspodjelu uloga i poslova. Te smo izazove pokušavali riješiti redovitom komunikacijom međusobno oko daljnih planova i organizacijom posla pomoću Project boarda. Trebalo je poraditi i na motivaciji samog tima za uspješnim završavanjem projekta budući da bi time potakli svakog člana individualno da izvršava obaveze na vrijeme. Također, cjepljanje zadataka na Project boardu na manje podzadatke nešto je na što bi u budućim projektnim radovima trebali pripaziti.

<b>Rev.</b>	<b>Opis promjene/dodataka</b>	<b>Autori</b>	<b>Datum</b>
0.1	Napravljen predložak	Filip Matijević	13.10.2025
0.2	Napisan opis projektnog zadatka	Filip Matijević	25.10.2025
0.3	Napisani funkcionalni zahtjevi	Filip Matijević	25.10.2025
0.4	Napisani ostali zahtjevi i dionici	Matej Glavaš	28.10.2025
0.5	Dodavanje aktora i njihovih funkcionalnih zahtjeva	Matej Glavaš	28.10.2025
0.6	Napisan opis arhitekture, obrazloženje odabira arhitekture i opisana organizacija sustava	Andro Gabaj	2.11.2025
0.7	dodani grafovi za dinamičko ponašanje aplikacije, dodan graf za dijagram stanja	Andro Gabaj	2.11.2025
0.8	Dodata baza podataka i dijagram baze podataka	Matej Glavaš	2.11.2025
0.9	Wiki 3. - Opisani obrasci uporabe	Ante Ivančić	3.11.2025
0.10	Wiki 3. - Dodani dijagrami obrazaca uporabe	Ante Ivančić	3.11.2025
0.11	Wiki 4. - dodani UML dijagrami aktivnosti	Filip Matijević	3.11.2025
0.11	Dodata nefunkcionalnih zahtjeva	Petar Ćurić	10.11.2025
0.12	Wiki 3. - Dodani sekvencijski dijagrami	Ante Ivančić	12.11.2025
0.13	Wiki 3. - Opisani sekvencijski dijagrami	Ante Ivančić	12.11.2025
0.14	Dodan dijagram razreda	Matej Glavaš	12.11.2025
0.15	Dodan ER model	Matej Glavaš	13.11.2025
0.16	Obavljeno i dokumentirano ispitivanje komponenti sustava	Andro Gabaj	17.1.2026
0.17	Upute za puštanje u pogon	Matej Glavaš	19.1.2026

<b>Rev.</b>	<b>Opis promjene/dodataka</b>	<b>Autori</b>	<b>Datum</b>
0.18	Obavljena i dokumentirana ispitivanja cjelokupanog sustava	Andro Gabaj	20.1.2026
0.19	Definirane tehnologije za implementaciju aplikacije	Matej Glavaš	20.1.2026
0.20	Napisano poglavlje 5. Arhitektura komponenata i razmještaja	Filip Matijević	21.1.2026
0.21	Dodan zaključak	Matej Glavaš	23.1.2026
0.22	Ažurirana literatura i prikaz aktivnosti grupe	Filip Matijević	23.1.2026
0.23	Popravljeni Use Case dijagrami po povratnim informacijama s prve predaje	Ante Ivančić	23.1.2026
0.24	Napravljena prezentacija	Ante Ivančić	23.1.2026

- ovo treba biti vidljivo u samoj dinamici promjena repozitorija

Evidencija promjena sadrži popis promjena izvršenih tijekom cijelog životnog trajanja predmeta evidencije (dokumentacije, projekta i sl.). Osnova svrha je praćenja napretka svake promjene na temelju njezina preispitivanja, odobrenja (ili odbijanja), provedbe, kao i zaključenja. Štoviše, dobar dnevnik promjena sadrži i datum promjene i njegov utjecaj na projekt u smislu rizika, vremena i troškova. Sve te promjene prenose se dionicima. Štoviše, odbačene promjene također su uključene u povijest promjena.

Zašto? Olakšano praćenje ključnim dionicima.