Programsko inženjerstvo ak.god 2025./2026

Sveučilište u Zagrebu

Fakultet elektrotehnike i računarstva

StanPlan

Tim: TG09.2
Ime tima: Pčelice
Članovi tima: Dominik Topić, Bruno Pleše, Luka Sever, Diana Agejev, Vito Gašparac, Amalija Novalić, Dora Vukelić
Nastavnik: Vlado Sruk

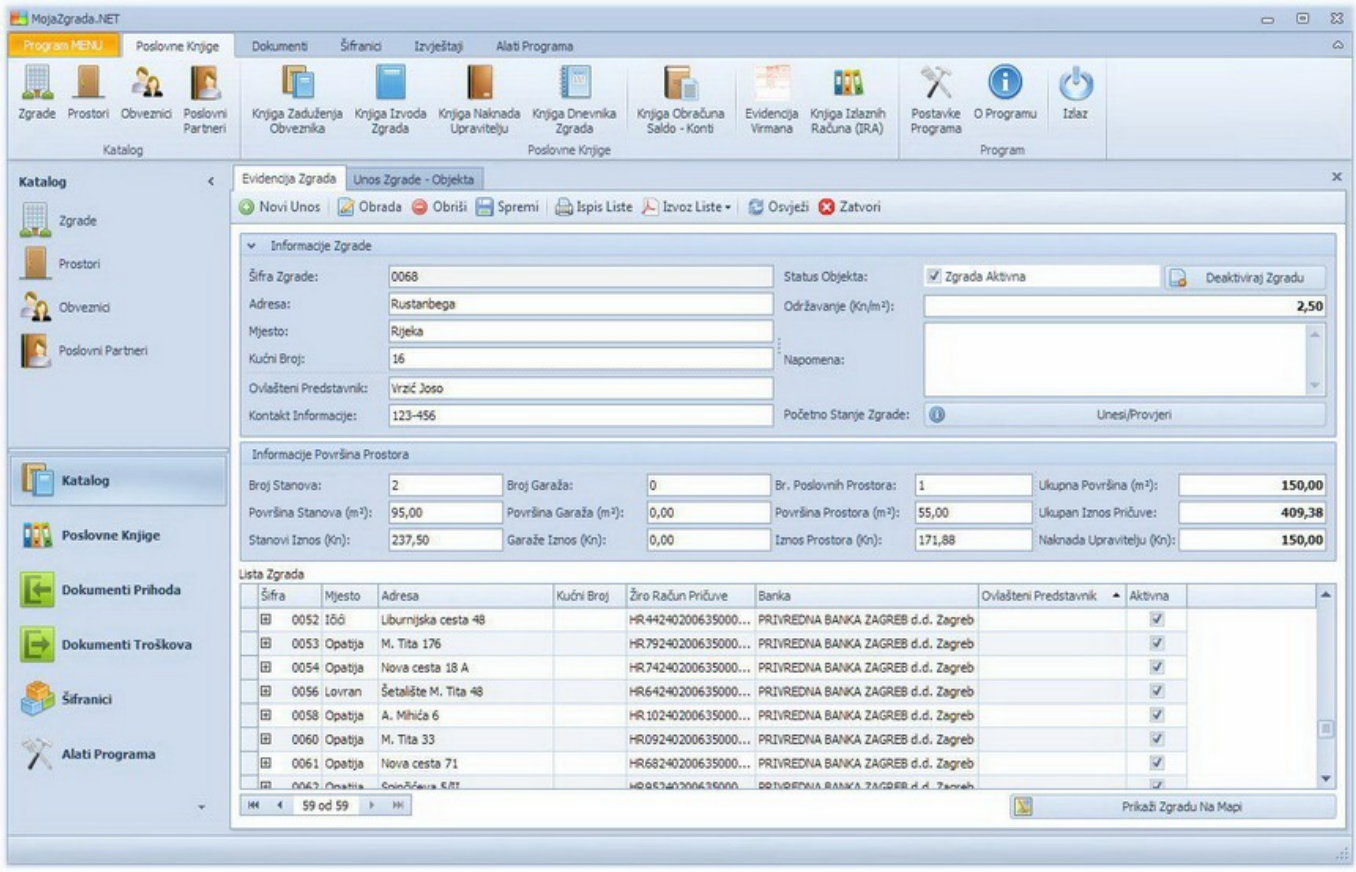
Motivacija

Komunikacija među stanarima u stambenim zgradama često je izazovna. Najčešće se odvija preko oglasne ploče u haustoru zgrade, a posljednjih desetak godina mnoge stambene zajednice komuniciraju putem grupa na aplikacijama poput WhatsAppa ili Vibera. Problem s komunikacijom preko oglasnik ploča jest da informacija ne dolazi do korisnika već bi korisnik morao doći do informacije - ako stanar za vrijeme kad je obavijest ovješena na oglasnoj ploči ne obitava u zgradi već je primjerice na godišnjem odmoru u drugom gradu, informacija će ga zaobići. Ako pretpostavimo da stanari i jesu u zgradi, to bi značilo da moraju redovito provjeravati što se nalazi na oglasnoj ploči, a to zbog svakodnevne žurbe i reklamnih sadržaja koji se tamo znaju naći, najčešće nije slučaj. Bez obzira na to radi li se o zgradama od tri, tristo ili preko tisuću stanova što je primjerice slučaj u zagrebačkoj Mamutici, imobilna informacija ne garantira to da će naći put do svih primatelja. Nedostatak komunikacije u grupama stanara na WhatsAppu ili Viberu jest izražena vertikalnost layouta chatova. Razgovori u kojem sudjeluje puno osoba i u kojem je poslano puno poruka, nepregledni su i zbog toga bitne stavke razgovora lako mogu promaknuti i izgubiti se u ostalim porukama - pogotovo ako u razgovoru ne sudjelujemo uživo već poruke čitamo naknadno. Također, razlikuju se preference i stilovi komunikacije ljudi pa se tako neki više vole dopisivati dok drugi žele zadržati komunikaciju koja se tiče zgrade što je sažetije i konkretnije moguće.

Specijalizirana aplikacija za komunikaciju stanara o aktualnim i važnim pitanjima vezanim za njihovu zgradu osigurala bi da sve obavijesti dođu do svakog stanara u najpreglednijem i sistematičnijem mogućem obliku. Osim što bi uvelike olakšao planiranje sastanaka stanara te služio kao zapisnik istih, prostor aplikacije otvoren je i za buduća proširenja. Tako možemo zamisliti da za pojedine projekte koje stanari žele sprovesti u zgradi, na primjer obnovu fasade, u aplikaciji možemo stvoriti diskusiju. Nadalje, aplikacija bi automatski mogla pohranjivati dokumentaciju vezanu za zgradu ili zakone koji bi mogli biti relevatni za aktivnosti u zgradi. Time bi se stanarima uštedjelo vrijeme kojeg u suvremenom tempu života nikad nemamo dovoljno. Cilj aplikacije bio bi olakšati i lokalizirati komunikaciju stanarima te aplikaciju učiniti njihovim idealnim posrednikom.

Postojeća rješenja

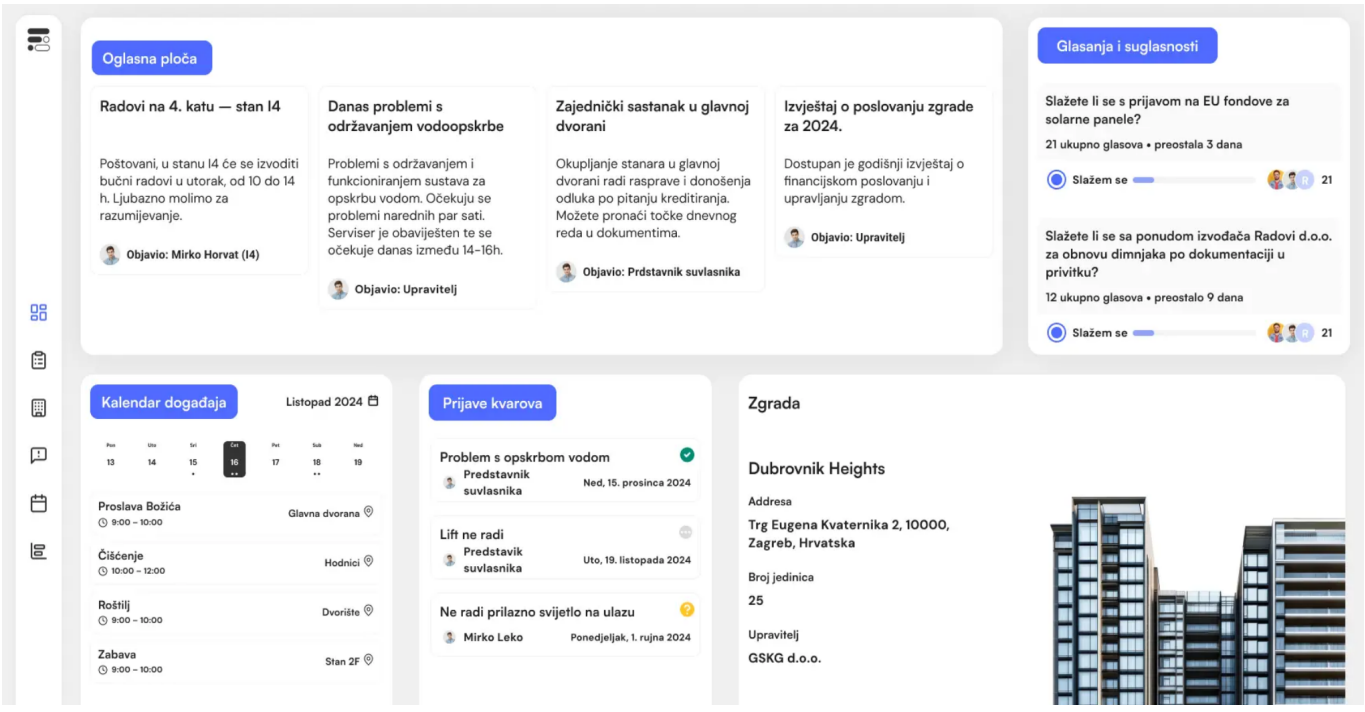
MojaZgrada.NET



Sučelje MojaZgrada.NET

Aplikacija omogučuje evidenciju troškova, objavu računa, vođenje pričuve, pregled arhive dokumenata i komunikaciju upravitelja i stanara. Razlika te i naše aplikacije je u tome što njihova ima jako kompleksan sustav te je namjenjen profesionalnim upraviteljima zgrada, ne samim stanarima. Naša aplikacija će biti jednostavna za korištenje, ne samo financija zgrade već i normalnu komunikaciju stanara.

Flatie



Sučelje Flatie

Flatie je aplikacija koja služi dijeljenju računa i praćenje stanja nekretnina. Namjenjena je prvenstveno stanodavcima i najmoprimce dok mi ciljamo na komunikaciju u stalnoj zajednici unutar zgrade.

BoardSpace

Sučelje BoardSpace

BoardSpace nudi usluge vođenja sastanaka, glasanja, podjelu zadataka i njihovu dokumentaciju. Ova aplikacija se koristi u velikim zajednicama te su njene funkcije zato neprikladne za našu željenu uporabu. Također aplikacija je samo na engleskom jeziku što bi moglo otežati korištenje nekim stanarima.

Funkcionalni zahtjevi

ID zahtjeva	Opis	Prioritet	Izvor	Kriteriji prihvaćanja
F-001	Sustav omogućuje administratoru kreiranje korisničkih računa za predstavnike suvlasnika i suvlasnike.	Visok	Naručitelj	Administrator može dodati korisnika, dodijeliti e-mail i ulogu te korisnik prima inicijalnu lozinku.
F-002	Sustav omogućuje korisnicima promjenu inicijalne lozinke nakon prve prijave.	Srednji	Razvojni tim	Korisnik može unijeti staru lozinku, novu lozinku, te potvrditi promjenu.

ID zahtjeva	Opis	Prioritet	Izvor	Kriteriji prihvaćanja
F-003	Sustav omogućuje korisnicima prijavu putem vanjskog servisa za autentifikaciju (OAuth 2.0).	Visok	Razvojni tim	Korisnik se može prijaviti putem odabranog vanjskog servisa.
F-004	Sustav omogućuje predstavniku kreiranje novog sastanka u stanju "Planiran".	Visok	Naručitelj	Predstavnik može unijeti naslov, sažetak namjere, vrijeme i mjesto sastanka.
F-005	Sustav omogućuje predstavniku dodavanje točaka dnevnog reda sastanku.	Visok	Naručitelj	Predstavnik može dodati, uređivati ili brisati točke dnevnog reda, te označiti imaju li pravni učinak.
F-006	Sustav omogućuje predstavniku objavu sastanka, čime postaje vidljiv svim suvlasnicima.	Visok	Naručitelj	Sastanak s barem jednom točkom može biti objavljen.
F-007	Sustav šalje obavijesti e-mailom suvlasnicima prilikom objave i arhiviranja sastanka.	Visok	Naručitelj	E-mail poruke se automatski šalju na adrese svih suvlasnika kad se sastanak objavi ili arhivira.
F-008	Sustav omogućuje suvlasnicima pregled objavljenih sastanaka i točaka dnevnog reda.	Visok	Naručitelj	Suvlasnik vidi listu objavljenih sastanaka i njihove detalje.
F-009	Sustav omogućuje suvlasnicima da potvrde sudjelovanje na sastanku.	Visok	Naručitelj	Suvlasnik može označiti hoće li sudjelovati na sastanku, prikazuje se broj potvrđenih sudjelovanja na prikazu sastanka.
F-010	Sustav omogućuje predstavniku prelazak stanja sastanka u "obavljen" nakon isteka vremena sastanka.	Visok	Naručitelj	Sustav bilježi vrijeme završetka, Predstavnik tada može stanje sastanka promijeniti u "obavljen".
F-011	Sustav omogućuje unos zaključaka za svaku točku dnevnog reda nakon sastanka.	Visok	Naručitelj	Predstavnik može dodati tekst zaključka i označiti status točke ("Izglasan", "Odbijen").

ID zahtjeva	Opis	Prioritet	Izvor	Kriteriji prihvaćanja
F-012	Sustav omogućuje arhiviranje sastanka nakon unosa svih potrebnih zaključaka.	Visok	Naručitelj	Sastanak prelazi u stanje "Arhiviran". Sustav prikazuje upozorenje i brani arhiviranje dok svi zaključci s pravnim učinkom nisu dodani. Arhivirani sastanak se više ne može mijenjati.
F-013	Sustav omogućuje povezivanje točke dnevnog reda s diskusijom iz aplikacije StanBlog.	Srednji	StanBlog	Predstavnik može odabrati diskusiju iz liste preuzete s API-ja StanBloga i dodati poveznicu.
F-014	Sustav omogućuje aplikaciji StanBlog kreiranje sastanka s jednom točkom dnevnog reda putem API-ja.	Srednji	StanBlog	StanBlog može pozvati API StanPlana i kreirati sastanak u stanju "Obavljen".
F-015	Administrator može unijeti adresu StanBlog servera za integraciju sa sučeljem za diskusije StanBlog aplikacije.	Srednji	Razvojni tim	Administrator može unijeti i ažurirati adresu API-ja StanBlog aplikacije u postavkama sustava.

Ostali zahtjevi

Zahtjevi pouzdanosti i performansi

ID zahtjeva	Opis	Prioritet
NF-1.1	Sustav treba raditi pouzdano i bez prekida tijekom rada korisnika.	Visok
NF-1.2	Vrijeme učitavanja osnovnih stranica (popis sastanaka, obavijesti) ne smije biti duže od 3 sekunde.	Visok

Zahtjevi sigurnosti

ID zahtjeva	Opis	Prioritet
NF-2.1	Sustav mora omogućiti siguran pristup putem korisničkog imena i lozinke (OAuth2).	Visok
NF-2.2	Komunikacija između korisnika i sustava mora biti zaštićena (HTTPS).	Visok
NF-2.3	Sustav mora čuvati korisničke podatke na siguran način.	Visok
NF-2.3.1	Sustav treba redovito izrađivati sigurnosne kopije podataka.	Visok

Zahtjevi upotrebljivosti i dostupnosti

ID zahtjeva	Opis	Prioritet
NF-3.1	Sustav treba biti jednostavan za korištenje, s preglednim i razumljivim sučeljem.	Visok
NF-3.1.1	Sučelje treba biti konzistentno i prilagođeno korisnicima bez tehničkog predznanja.	Srednji
NF-3.2	Sustav treba biti dostupan i na mobilnim uređajima.	Srednji

Zahtjevi održavanja

ID zahtjeva	Opis	Prioritet
NF-4.1	Sustav treba biti oblikovan tako da omogućuje jednostavno održavanje.	Visok
NF-4.2	Sustav treba imati dovoljnu dokumentaciju.	Visok
NF-4.2.1	Dokumentacija treba uključivati „Priručnik za rad“ s opisom pravilne upotrebe sustava.	Visok
NF-4.2.2	Dokumentacija treba sadržavati „Plan implementacije“ za pravilno postavljanje sustava.	Visok

Dionici

Dionici sustava:

1. Suvlasnik - ima osnovna korisnička prava (pregledavati i sudjelovati na sastancima)
2. Predstavnik suvlasnika - ima sve mogućnosti kao suvlasnik uz dodatne ovlasti za kreiranje i oglašavanje sastanka
3. Administrator - odgovoran za održavanje sustava i upravljanje zahtjevima korisnika te kreiranje korisnika
4. Razvojni tim - odgovoran za kod, bazu i sigurnost (OAuth2, HTTPS)
5. StanBlog - vanjski sustav za diskusije koji se integrira putem API-ja
6. Naručitelj - osoba ili tijelo koje definira poslovne ciljeve i financira razvoj sustava

Aktori i njihovi funkcionalni zahtjevi:

A-1 Suvlasnik može:

> Promijeniti inicijalnu lozinku (F-002)
> Prijava putem OAuth 2.0 (F-003)
> Primati e-mail obavijesti o objavi/arhiviranju sastanka (F-007)
> Pregled objavljenih sastanka i dnevnog reda (F-008)
> Potvrda sudjelovanja na sastanku (F-009)

A-2 Predstavnik suvlasnika može:

> Kreiranje planiranih sastanaka (F-004)
> Upravljanje točkama dnevnog reda (dodavanje, brisanje, pravni učinak) (F-005)
> Objava sastanka suvlasnicima (F-006)
> Promjena stanja sastanka u "Obavljen" (F-010)
> Unos zaključaka i ishoda glasanja po točkama (F-011)
> Arhiviranje dovršenih sastanaka (F-012)
> Povezivanje točaka s diskusijama iz StanBloga (F-013)

A-3 Administrator može:

> Kreirati korisničke račune i dodjeljivati uloge (F-001)
> Upravljati integracijama s vanjskim servisima (F-015)

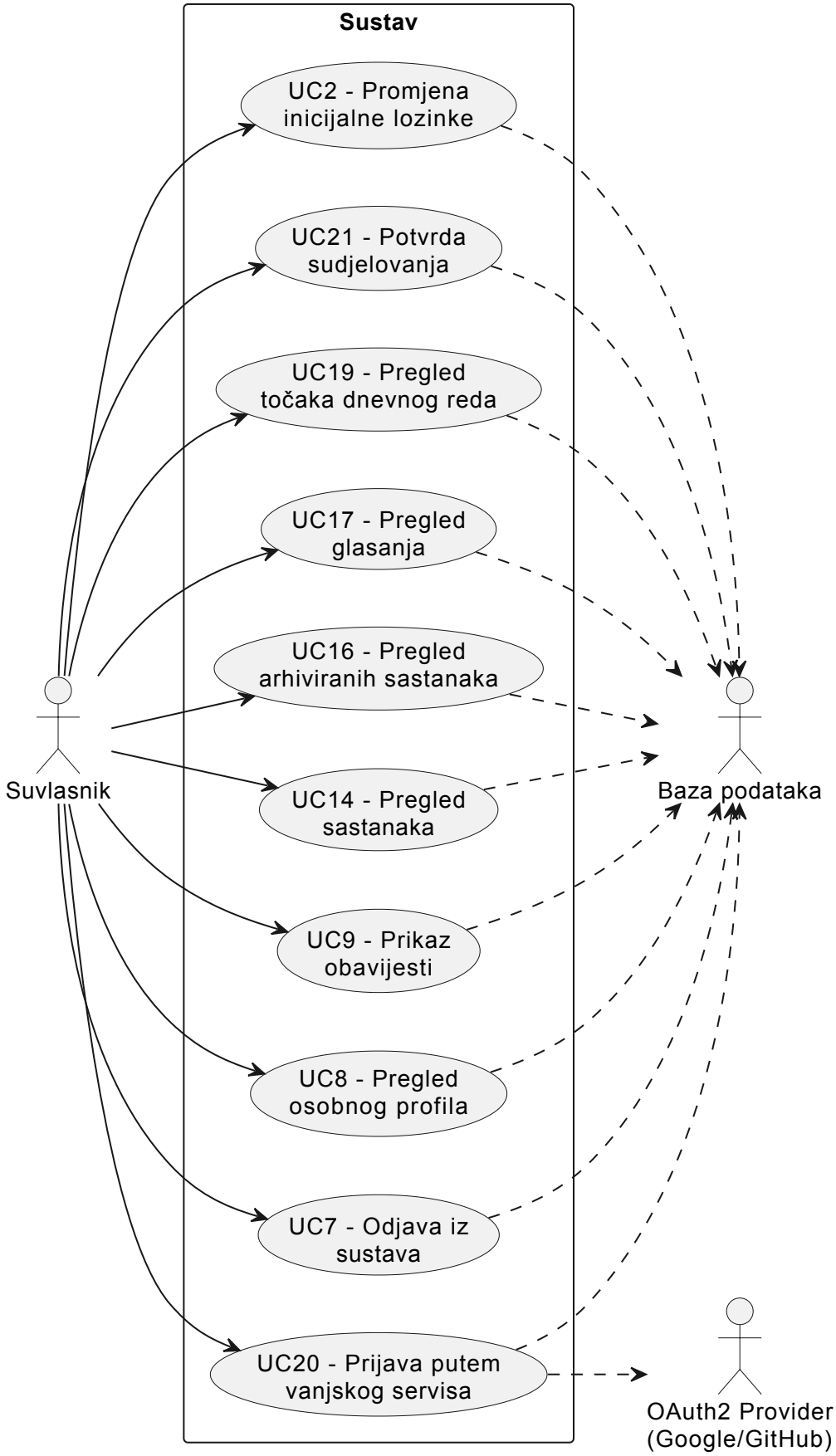
A-4 StanBlog može:

> Kreirati sastanak s barem jednom točkom dnevnog reda putem API-ja (F-014)

Obrasci uporabe

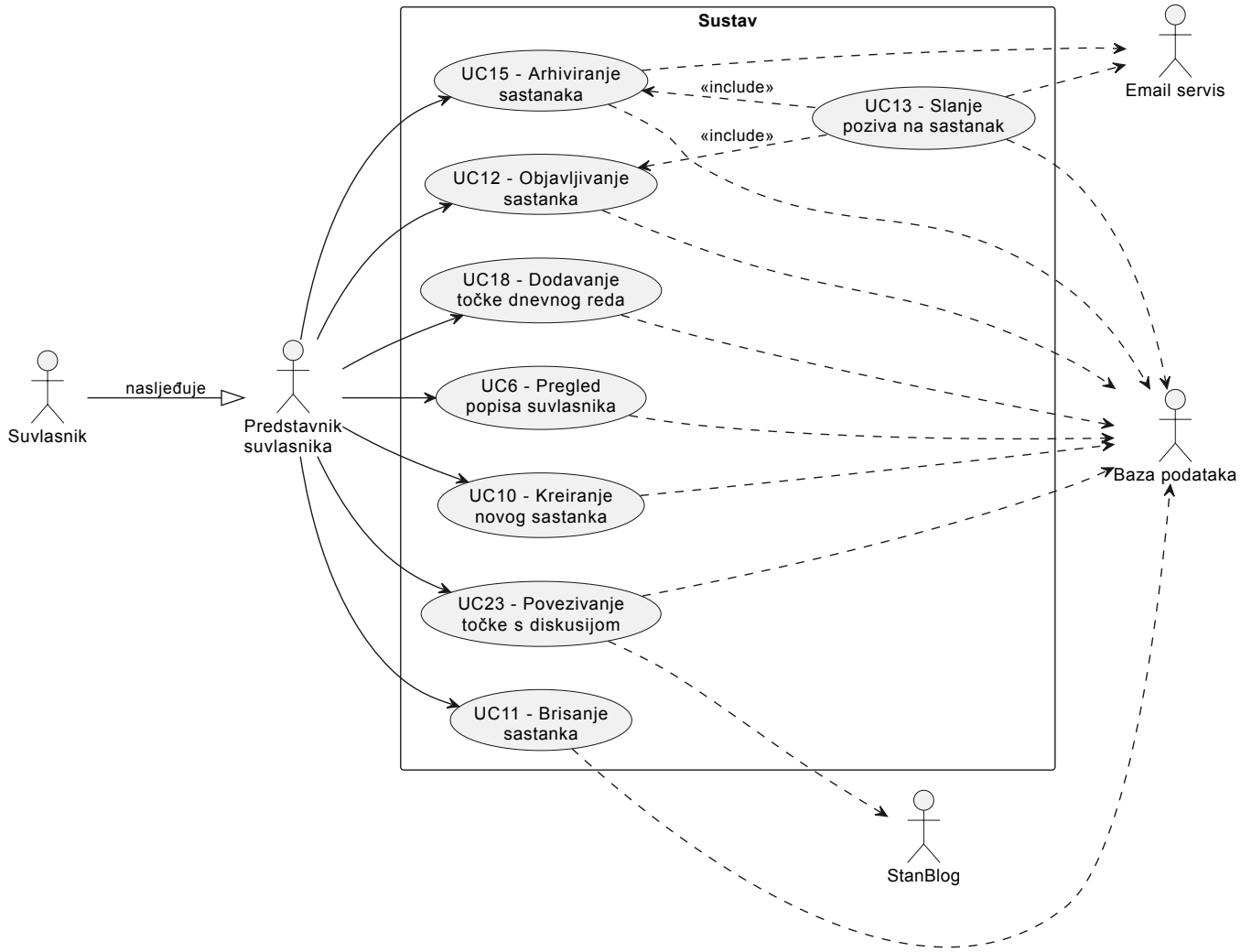
Dijagrami obrazaca uporabe

1. Funkcionalni zahtjevi aktora suvlasnik



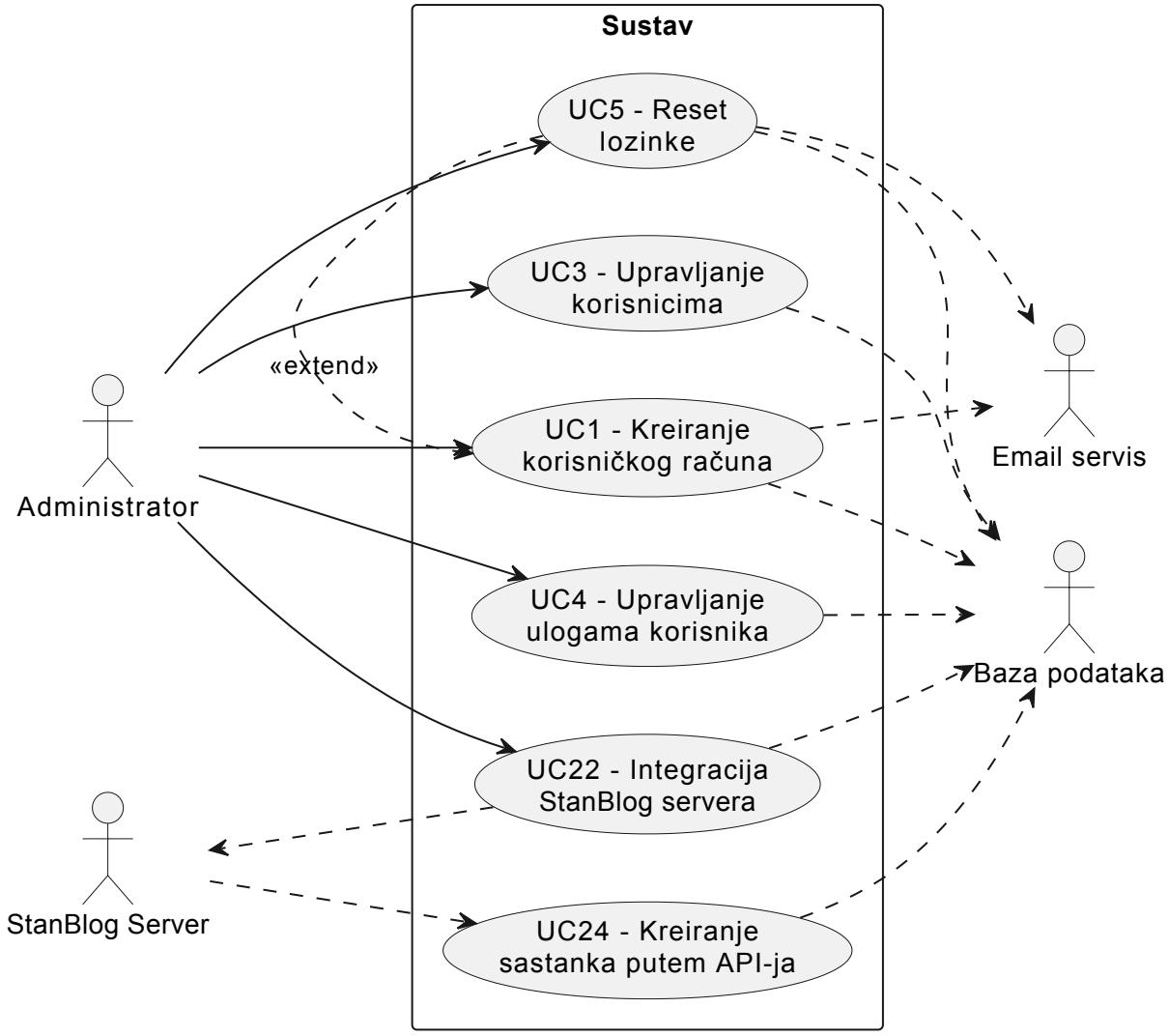
Dijagram prikazuje funkcionalne zahtjeve sustava iz perspektive uloge suvlasnika. Suvlasnik može izvršavati razne operacije, poput promjene inicijalne lozinke, odjave iz sustava, pregleda osobnog profila i arhiviranih sastanaka, praćenja obavijesti, sudjelovanja u glasanjima i sastancima te pregledavanja točaka dnevnog reda. Također, dijagram uključuje mogućnost autentikacije putem vanjskog (OAuth) servisa te, uz interakciju s bazom podataka, prikazuje ključne funkcionalnosti dostupne ovom korisniku u aplikaciji.

2. Funkcionalni zahtjevi aktora predstavnik suvlasnika



Dijagram prikazuje funkcionalnosti koje su dostupne predstavniku suvlasnika, koji ima šire ovlasti u odnosu na običnog suvlasnika. Predstavnik može upravljati sastancima (kreirati, brisati, arhivirati i objavljivati sastanke, te slati pozive i dodavati točke dnevnog reda), mijenjati lozinku, pregledavati popis suvlasnika, te povezivati točke dnevnog reda s diskusijama iz StanBlog aplikacije. Dijagram prikazuje integraciju s bazom podataka, sustavom za e-mail obavijesti, kao i s vanjskom StanBlog aplikacijom za upravljanje diskusijama. Također je naznačen odnos nasljeđivanja, gdje predstavnik suvlasnika automatski nasljeđuje sve funkcionalnosti dostupne suvlasniku.

3. Funkcionalni zahtjevi aktora administrator



Na dijagramu su prikazane funkcije kojima raspolaže administrator sustava. Administrator ima mogućnost kreiranja korisničkih računa, upravljanja korisnicima te njihovim ulogama u sustavu. Poseban slučaj predstavlja resetiranje lozinki, koji se može izvesti kao dodatna funkcionalnost unutar upravljanja korisnicima. Nadalje, administrator je odgovoran za konfiguraciju integracije s vanjskom StanBlog aplikacijom (UC22), čime se omogućava suradnja između sustava. Dijagram prikazuje kako su StanBlog događaji (npr. API zahtjevi za kreiranje novih sastanaka) obrađeni kao sistemske operacije (UC24). Kao i u prethodnim dijagramima, prikazana je i povezanost s bazom podataka koja podržava sve opisane procese, te s E-mail servisom i StanBlog vanjskom aplikacijom.

Opis obrazaca uporabe

UC1 - Kreiranje korisničkog računa

- Glavni sudionik: Administrator
- Cilj: Kreirati korisnički račun za predstavnika suvlasnika ili suvlasnika.
- Sudionici: Baza podataka
- Preduvjet: Administrator mora biti prijavljen u sustav.
- Opis osnovnog tijeka:

> 1. Administrator odabire opciju “Dodaj korisnika”. (F-001)
> 2. Upisuje ime, prezime, e-mail adresu i ulogu korisnika.
> 3. Sustav provjerava ispravnost unesenih podataka.

> 4. Sustav generira inicijalnu lozinku i pohranjuje korisnika u bazu podataka.
> 5. Sustav automatski šalje e-mail korisniku s inicijalnim podacima za prijavu. (F-002)

• Opis mogućih odstupanja:

> 3. Neispravan unos podataka.
> 3.a Sustav prikazuje poruku o pogrešci i vraća administratora na unos.
> 4. E-mail već postoji u sustavu.
> 2.b Sustav obavještava administratora i traži unos nove adrese.

UC2 - Promjena inicijalne lozinke

- Glavni sudionik: Korisnik (predstavnik ili suvlasnik)
- Cilj: Promijeniti inicijalnu lozinku nakon prve prijave.
- Sudionici: Baza podataka
- Preduvjet: Korisnik je prijavljen pomoću inicijalne lozinke.
- Opis osnovnog tijeka:

> 1. Korisnik odabire opciju “Promjena lozinke”.
> 2. Unosi staru lozinku i novu lozinku te potvrđuje unos. (F-002)
> 3. Sustav provjerava ispravnost stare lozinke.
> 4. Sustav ažurira lozinku u bazi podataka.
> 5. Prikazuje se poruka o uspješnoj promjeni lozinke.

• Opis mogućih odstupanja:

> 3. Stara lozinka nije točna
> 3.a Sustav prikazuje poruku o pogrešci i traži ponovni unos.

UC3 – Upravljanje korisnicima

- Glavni sudionik: Administrator
- Cilj: Pregledati, urediti ili obrisati postojeće korisničke račune.
- Sudionici: Baza podataka
- Preduvjet: Administrator mora biti prijavljen u sustav.
- Opis osnovnog tijeka:

> 1. Administrator otvara stranicu “Korisnici”.
> 2. Sustav dohvaća popis svih korisnika iz baze podataka.
> 3. Administrator može pretraživati korisnike po imenu, e-mailu ili ulozi.
> 4. Administrator odabire opciju “Uredi” ili “Obriši” za određenog korisnika. (F-001)
> 5. Sustav sprema promjene ili briše korisnika iz baze.

UC4 – Upravljanje ulogama korisnika

- Glavni sudionik: Administrator
- Cilj: Promijeniti ulogu korisnika (npr. suvlasnik → predstavnik).
- Sudionici: Baza podataka

- Preduvjet: Administrator mora biti prijavljen u sustav.
- Opis osnovnog tijeka:

> 1. Administrator otvara popis korisnika. (F-001)
> 2. Odabire korisnika kojem želi promijeniti ulogu.
> 3. Odabire novu ulogu iz padajućeg izbornika.
> 4. Sustav ažurira ulogu u bazi podataka.
> 5. Sustav prikazuje poruku o uspješnoj promjeni uloge.

• Opis mogućih odstupanja:

> 3. Administrator pokušava dodijeliti nepostojeću ulogu.
> 3.a Sustav prikazuje poruku “Odabrana uloga nije valjana.”

UC5 – Reset lozinke

- Glavni sudionik: Korisnik (suvlasnik, predstavnik ili administrator)
- Cilj: Omogućiti korisniku ponovno postavljanje lozinke u slučaju zaborava.
- Sudionici: Sustav za e-mail obavijesti, Baza podataka
- Preduvjet: Korisnik ima registriranu e-mail adresu u sustavu.
- Opis osnovnog tijeka:

> 1. Korisnik na stranici za prijavu odabire opciju “Zaboravljena lozinka”.
> 2. Unosi svoju e-mail adresu.
> 3. Sustav provjerava postoji li korisnik s tom adresom.
> 4. Sustav generira privremeni link za reset lozinke.
> 5. Sustav šalje e-mail s uputama za postavljanje nove lozinke. (F-002)

• Opis mogućih odstupanja:

> 3. E-mail adresa ne postoji u sustavu.
> 3.a Sustav prikazuje poruku “Korisnik s unesenom e-mail adresom ne postoji.”

UC6 – Pregled popisa suvlasnika

- Glavni sudionik: Predstavnik suvlasnika
- Cilj: Pregledati popis svih suvlasnika poveznih s njegovom zgradom.
- Sudionici: Baza podataka
- Preduvjet: Predstavnik mora biti prijavljen u sustav.
- Opis osnovnog tijeka:

> 1. Predstavnik otvara modul “Suvlasnici”.
> 2. Sustav dohvaća popis suvlasnika povezanih s njegovom zgradom.
> 3. Prikazuju se osnovni podaci: ime, prezime, stan, kontakt e-mail.
> 4. Predstavnik može vidjeti popis imena ili brojeva stanova.

• Opis mogućih odstupanja:

> 2. Nema registriranih suvlasnika za tu zgradu.
> 2.a Sustav prikazuje poruku “Nema dostupnih suvlasnika za prikaz.”

UC7 – Odjava iz sustava

- Glavni sudionik: Korisnik (bilo koja uloga)
- Cilj: Sigurno se odjaviti iz sustava.
- Sudionici: —
- Preduvjet: Korisnik mora biti prijavljen u sustav.
- Opis osnovnog tijeka:

> 1. Korisnik klikne na opciju “Odjava”.
> 2. Sustav prekida korisničku sesiju.
> 3. Sustav preusmjerava korisnika na početnu (login) stranicu.
> 4. Prikazuje se poruka “Uspješno ste se odjavili.”

• Opis mogućih odstupanja:

> 2. Došlo je do greške pri odjavi.
> 2.a Sustav prikazuje poruku “Odjava nije uspjela, pokušajte ponovno.”

UC8 – Pregled osobnog profila

- Glavni sudionik: Korisnik (bilo koja uloga)
- Cilj: Pregledati osobne podatke i status računa.
- Sudionici: Baza podataka
- Preduvjet: Korisnik mora biti prijavljen.
- Opis osnovnog tijeka:

> 1. Korisnik otvara svoj profil.
> 2. Sustav prikazuje osobne podatke (ime, prezime, e-mail, uloga).
> 3. Korisnik može pregledati povezane zgrade i aktivnosti.

• Opis mogućih odstupanja:

> 1. Profil nije pronađen.
> 1.a Sustav prikazuje poruku “Profil nije pronađen.”

UC9 – Prikaz obavijesti unutar sustava

- Glavni sudionik: Korisnik
- Cilj: Pregledati sve obavijesti koje je korisnik primio kroz sustav.
- Sudionici: Baza podataka
- Preduvjet: Korisnik mora biti prijavljen.
- Opis osnovnog tijeka:

> 1. Korisnik otvara modul “Obavijesti”.
> 2. Sustav dohvaća sve obavijesti vezane uz korisnika.

> 3. Prikazuje popis obavijesti s naslovom i datumom.
> 4. Korisnik može otvoriti detalje svake obavijesti.

• Opis mogućih odstupanja:

> 3. Nema dostupnih obavijesti.
> 3.a Sustav prikazuje poruku “Trenutno nema obavijesti.”

UC10 – Kreiranje novog sastanka

- Glavni sudionik: Predstavnik suvlasnika
- Cilj: Kreirati novi sastanak u statusu "Planiran".
- Sudionici: Baza podataka
- Preduvjet: Predstavnik mora biti prijavljen u sustav.
- Opis osnovnog tijeka:

> 1. Predstavnik otvara stranicu “Novi sastanak”. (F-004)
> 2. Unosi naslov, opis, datum, vrijeme i mjesto sastanka.
> 3. Sustav sprema podatke o sastanku u bazu u statusu "Planiran".
> 4. Sustav prikazuje poruku o uspješnom kreiranju sastanka.

• Opis mogućih odstupanja:

> 4. Nedostaje neki obavezni podatak.
> 4.a Sustav prikazuje poruku “Potrebno je ispuniti sva polja.”

UC11 – Brisanje sastanka

- Glavni sudionik: Predstavnik suvlasnika
- Cilj: Obrisati sastanak koji je još u statusu "Planiran".
- Sudionici: Baza podataka
- Preduvjet: Sastanak ne smije biti objavljen.
- Opis osnovnog tijeka:

> 1. Predstavnik otvara popis sastanaka. (F-004)
> 2. Odabire sastanak sa statusom "Planiran".
> 3. Klikne opciju “Obriši”.
> 4. Sustav traži potvrdu brisanja.
> 5. Sustav briše sastanak iz baze podataka.

• Opis mogućih odstupanja:

> 3. Sastanak je već objavljen.
> 3.a Sustav prikazuje poruku "Objavljeni sastanak nije moguće obrisati."

UC12 – Objavljivanje sastanka

- Glavni sudionik: Predstavnik suvlasnika

- Cilj: Objaviti planirani sastanak kako bi bio vidljiv svim suvlasnicima.
- Sudionici: Baza podataka, Sustav za e-mail obavijesti
- Preduvjet: Postoji barem jedna točka dnevnog reda.
- Opis osnovnog tijeka:

> 1. Predstavnik odabire sastanak sa statusom "Planiran".
> 2. Klikne opciju “Objavi sastanak”.
> 3. Sustav provjerava uvjete objave (barem jedna točka postoji).
> 4. Sustav mijenja status sastanka u "Objavljen". (F-006)
> 5. Sustav automatski šalje e-mail obavijesti svim suvlasnicima. (F-007) (UC-13)

• Opis mogućih odstupanja:

> 3. Sastanak nema nijednu točku dnevnog reda.
> 3.a Sustav prikazuje upozorenje i ne dopušta objavu.

UC13 – Slanje poziva na sastanak

- Glavni sudionik: Predstavnik suvlasnika
- Cilj: Poslati e-mail poziv svim suvlasnicima na objavljeni sastanak.
- Sudionici: Sustav za e-mail obavijesti, Baza podataka
- Preduvjet: Sastanak mora imati status "Objavljen".
- Opis osnovnog tijeka:

> 1. Predstavnik otvara detalje objavljenog sastanka.
> 2. Odabire opciju “Pošalji pozive”.
> 3. Sustav dohvaća e-mail adrese svih suvlasnika.
> 4. Sustav automatski šalje e-mail s pozivnicom. (F-007)
> 5. Sustav prikazuje poruku o uspješnom slanju poziva.

UC14 – Pregled sastanaka

- Glavni sudionik: Suvlasnik
- Cilj: Pregledati sve dostupne sastanke (planirane, objavljene i obavljene).
- Sudionici: Baza podataka
- Preduvjet: Suvlasnik mora biti prijavljen u sustav.
- Opis osnovnog tijeka:

> 1. Suvlasnik odabire opciju “Sastanci” u glavnom izborniku. (F-008)
> 2. Sustav dohvaća sve sastanke vezane uz zgradu korisnika.
> 3. Sustav prikazuje popis sastanaka s osnovnim informacijama (naslov, datum, status).
> 4. Suvlasnik može odabrati pojedini sastanak za detaljan prikaz.

• Opis mogućih odstupanja:

> 3. Ne postoji nijedan sastanak za prikaz.
> 3.a Sustav prikazuje poruku “Trenutno nema dostupnih sastanaka.”

UC15 – Arhiviranje sastanaka

- Glavni sudionik: Predstavnik suvlasnika
- Cilj: Arhivirati završene sastanke radi lakšeg pregleda i evidencije.
- Sudionici: Baza podataka
- Preduvjet: Sastanak mora imati status "Obavljen". (F-010)
- Opis osnovnog tijeka

> 1. Predstavnik otvara popis obavljenih sastanaka.
> 2. Odabire sastanak koji želi arhivirati.
> 3. Klikne opciju “Arhiviraj”.
> 4. Sustav mijenja status sastanka u "Arhiviran". (F-012)
> 5. Sustav prikazuje poruku o uspješnom arhiviranju.
> 6. Sustav automatski šalje e-mail obavijesti svim suvlasnicima. (F-007)

• Opis mogućih odstupanja:

> 4. Sastanak je već arhiviran.
> 4.a Sustav prikazuje poruku “Sastanak je već arhiviran.”

UC16 – Pregled arhiviranih sastanaka

- Glavni sudionik: Suvlasnik
- Cilj: Pregledati sve prethodne (arhivirane) sastanke.
- Sudionici: Baza podataka
- Preduvjet: Suvlasnik mora biti prijavljen u sustav.
- Opis osnovnog tijeka:

> 1. Suvlasnik otvara sekciju “Arhiva sastanaka”. (F-008)
> 2. Sustav dohvaća sve arhivirane sastanke povezane s korisnikom.
> 3. Suvlasnik odabire sastanak za prikaz detalja.
> 4. Sustav prikazuje zapisnik i rezultate glasanja.

• Opis mogućih odstupanja:

> 2. Nema arhiviranih sastanaka.
> 2.a Sustav prikazuje poruku “Trenutno nema arhiviranih sastanaka.”

UC17 – Pregled glasanja

- Glavni sudionik: Suvlasnik
- Cilj: Pregled rezultata glasanja.
- Sudionici: Baza podataka
- Preduvjet: Sastanak mora biti "Obavljen".
- Opis osnovnog tijeka:

> 1. Suvlasnik otvara obavljeni sastanak.
> 2. Sustav dohvaća podatke o rezultatima.

> 3. Prikazuje "Izglasan" ili "Odbijen". (F-011)

• Opis mogućih odstupanja:

> 2. Nema dovršenih glasanja.
> 2.a Sustav prikazuje poruku “Trenutno nema dostupnih podataka za prikaz.”

UC18 – Dodavanje točke dnevnog reda

- Glavni sudionik: Predstavnik suvlasnika
- Cilj: Dodati novu točku dnevnog reda za određeni sastanak.
- Sudionici: Baza podataka
- Preduvjet: Predstavnik mora biti prijavljen u sustav i sastanak mora postojati.
- Opis osnovnog tijeka:

> 1. Predstavnik otvara detalje planiranog sastanka.
> 2. Odabire opciju “Dodaj točku dnevnog reda”. (F-005)
> 3. Unosi naslov i opis točke.
> 4. Sustav sprema točku u bazu podataka.
> 5. Sustav prikazuje poruku o uspješnom dodavanju točke.

• Opis mogućih odstupanja:

> 4. Nedostaje naslov ili opis točke.
> 4.a Sustav prikazuje poruku “Potrebno je unijeti naslov i opis.”

UC19 – Pregled točaka dnevnog reda

- Glavni sudionik: Suvlasnik
- Cilj: Pregledati sve točke dnevnog reda određenog sastanka.
- Sudionici: Baza podataka
- Preduvjet: Sastanak mora biti objavljen.
- Opis osnovnog tijeka:

> 1. Suvlasnik otvara detalje objavljenog sastanka.
> 2. Sustav prikazuje popis točaka dnevnog reda s opisima. (F-011)
> 3. Suvlasnik može otvoriti pojedinu točku za više detalja.

• Opis mogućih odstupanja:

> 2. Sastanak nema dodanih točaka.
> 2.a Sustav prikazuje poruku “Nema točaka dnevnog reda za prikaz.”

UC20 – Prijava u sustav putem vanjskog servisa

- Glavni sudionik: Korisnik (bilo koja uloga)
- Cilj: Omogućiti korisniku prijavu putem vanjskog servisa (OAuth 2.0).
- Sudionici: Sustav za autentifikaciju treće strane (npr. Google OAuth), Baza podataka

- Preduvjet: Korisnik mora imati račun u sustavu povezan s vanjskim servisom.
- Opis osnovnog tijeka:

> 1. Korisnik odabire opciju “Prijava putem Google/Microsoft računa”. (F-003)
> 2. Sustav preusmjerava korisnika na stranicu vanjskog servisa za autentifikaciju.
> 3. Korisnik unosi vjerodajnice na servisu treće strane.
> 4. Sustav prima potvrdu o identitetu od servisa.
> 5. Sustav prijavljuje korisnika i preusmjerava ga na početnu stranicu sustava.

•Opis mogućih odstupanja:

> 2. Korisnik ne ovlasti pristup aplikaciji.
> 2.a Sustav prikazuje poruku “Prijava nije dovršena – pristup nije odobren.”

UC21 – Potvrda sudjelovanja na sastanku

- Glavni sudionik: Suvlasnik
- Cilj: Označiti hoće li suvlasnik sudjelovati na sastanku.
- Sudionici: Baza podataka
- Preduvjet: Sastanak mora biti objavljen, a suvlasnik prijavljen u sustav.
- Opis osnovnog tijeka:

> 1. Suvlasnik otvara objavljeni sastanak.
> 2. Odabire opciju “Potvrdi sudjelovanje”. (F-009)
> 3. Sustav bilježi potvrdu sudjelovanja u bazi podataka.
> 4. Sustav ažurira broj prijavljenih sudionika.
> 5. Sustav prikazuje poruku o uspješnoj potvrdi.

UC22 – Integracija StanBlog servera

- Glavni sudionik: Administrator
- Cilj: Konfigurirati adresu vanjskog StanBlog servera kako bi integracija bila moguća.
- Sudionici: Baza podataka
- Preduvjet: Administrator mora biti prijavljen u sustav.
- Opis osnovnog tijeka:

> 1. Administrator otvara stranicu „Postavke sustava“.
> 2. Administrator unosi ili ažurira URL adresu StanBlog API-ja. (F-015)
> 3. Sustav validira format unesene adrese.
> 4. Sustav sprema adresu u konfiguraciju baze podataka.
> 5. Sustav potvrđuje uspješno spremanje postavki.

UC23 – Povezivanje točke s StanBlog diskusijom

- Glavni sudionik: Predstavnik suvlasnika
- Cilj: Povezati točku dnevnog reda s relevantnom diskusijom iz StanBlog aplikacije.
- Sudionici: Baza podataka, StanBlog
- Preduvjet: Sastanak je u statusu „Planiran“, a Administrator je ispravno konfigurirao StanBlog URL.
- Opis osnovnog tijeka:

> 1. Predstavnik otvara detalje planiranog sastanka i točku dnevnog reda.
> 2. Odabire opciju „Poveži s diskusijom“. (F-013)
> 3. Sustav putem API-ja dohvaća listu aktivnih diskusija sa StanBlog servera.
> 4. Predstavnik odabire diskusiju s popisa.
> 5. Sustav sprema poveznicu (link) na diskusiju uz točku dnevnog reda.

UC24 – Kreiranje sastanka putem StanBlog API-ja

- Glavni sudionik: StanBlog (vanjski sustav)
- Cilj: StanBlog može kreirati sastanak u StanPlanu s barem jednom točkom dnevnog reda putem API poziva.
- Sudionici: Baza podataka
- Preduvjet: Administrator je konfigurirao adresu StanBlog servera (UC22), sustav je dostupan.
- Opis osnovnog tijeka:

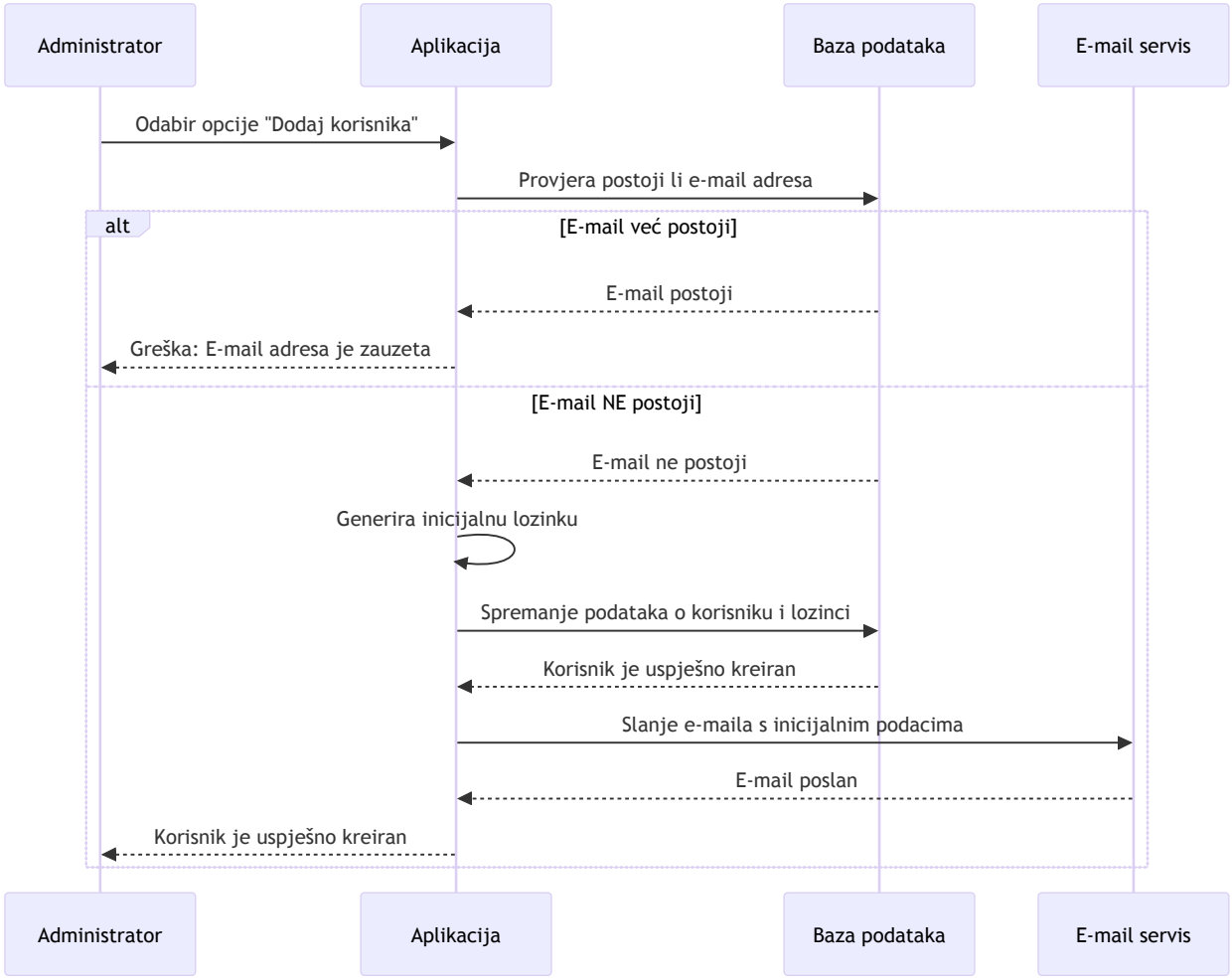
> 1. StanBlog API šalje zahtjev za kreiranje novog sastanka s minimalnim podacima (naslov, točka dnevnog reda).
> 2. Sustav validira sadržaj zahtjeva.
> 3. Sustav sprema sastanak u bazu podataka u statusu "Obavljen".
> 4. Sustav vraća potvrdu s ID-om kreirane sastanka StanBlogu.

• Opis mogućih odstupanja:

> 2. Zahtjev nema obaveznih podataka. 2.a Sustav vraća grešku i ne kreira sastanak.

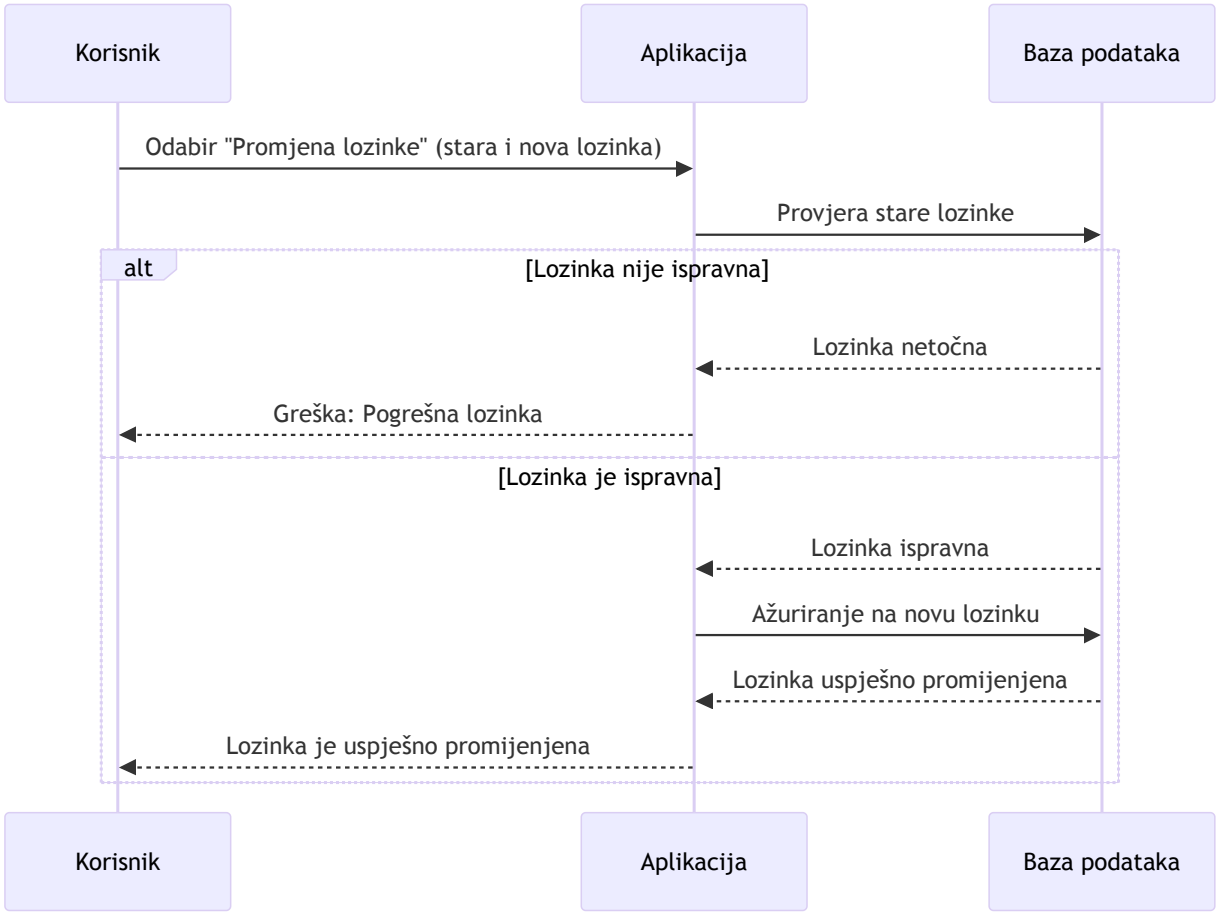
Sekvencijski dijagrami

UC1 – Kreiranje korisničkog računa



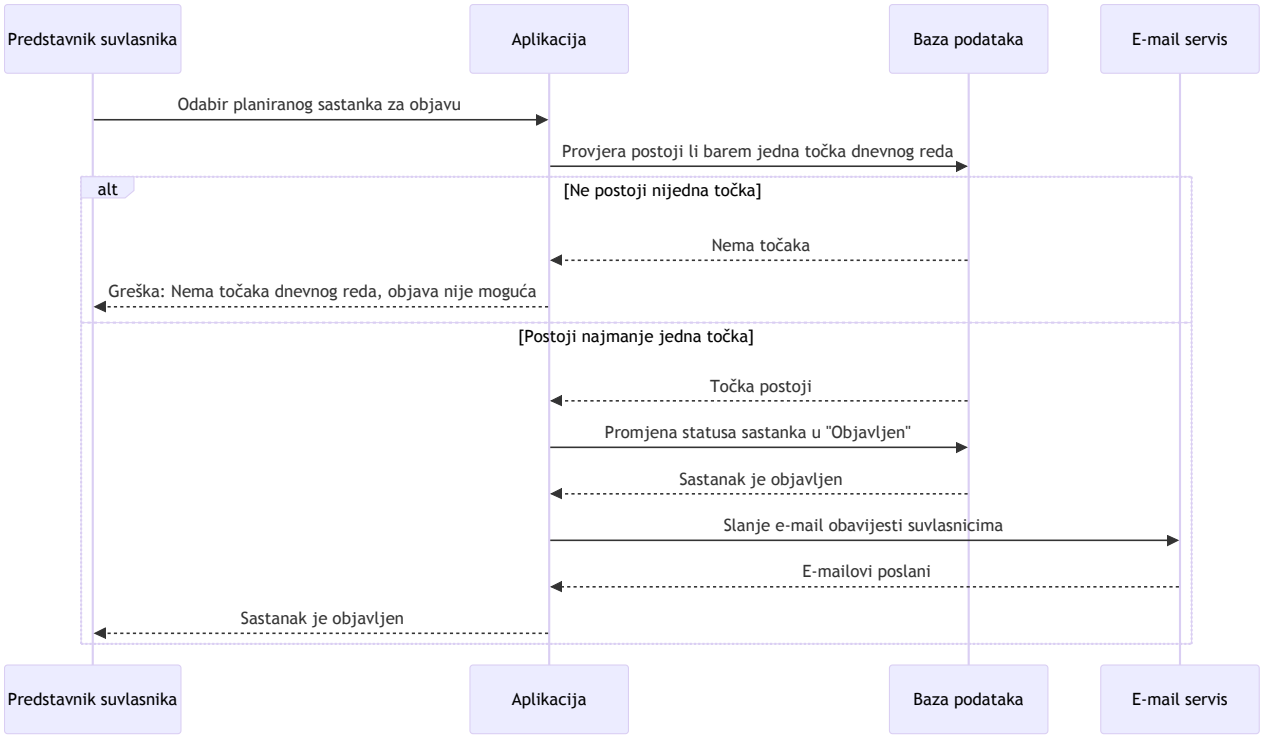
Dijagram prikazuje tijek komunikacije između administratora, sustava, i baze podataka prilikom kreiranja novog korisničkog računa za predstavnika suvlasnika ili suvlasnika. Administrator inicira proces unosom korisničkih podataka, a sustav provodi validaciju, pohranu i obavještavanje korisnika putem e-maila.

UC2 – Promjena inicijalne lozinke



Dijagram prikazuje proces promjene inicijalne lozinke koji korisnik mora izvršiti prilikom prve prijave u sustav. U procesu sudjeluju korisnik, aplikacija i baza podataka. Cilj je povećati sigurnost sustava i osigurati da svatko koristi jedinstvenu lozinku koju zna samo on.

UC12 – Objavljivanje sastanka



Dijagram prikazuje proces objavljivanja planiranog sastanka od strane predstavnika suvlasnika. Prije objave,

sustav provjerava postoji li barem jedna točka dnevnog reda. Nakon uspješne objave, sastanak postaje vidljiv svim suvlasnicima, a automatski se šalju e-mail obavijesti.

Provjera uključenosti ključnih funkcionalnosti u obrasce uporabe

ID zahtjeva	Opis	Obrasci uporabe koji ga obuhvaćaju
F-001	Kreiranje korisničkih računa	UC1, UC3, UC4
F-002	Promjena i reset lozinke	UC2, UC5
F-003	Prijava putem vanjskog servisa (OAuth 2.0)	UC20
F-004	Kreiranje sastanka	UC10, UC11
F-005	Dodavanje točke dnevnog reda	UC18
F-006	Objavljivanje sastanka	UC12
F-007	Slanje e-mail obavijesti	UC12, UC13, UC15
F-008	Pregled sastanaka	UC14, UC16
F-009	Potvrda sudjelovanja	UC21
F-010	Prelazak sastanka u “obavljen”	UC12, UC15
F-011	Unos zaključaka po točkama	UC17, UC19
F-012	Arhiviranje sastanka	UC15
F-013	Povezivanje točke s diskusijom	UC23
F-014	Kreiranje sastanka putem API-ja	UC24
F-015	Unos adrese StanBlog servera	UC22

Arhitektura sustava

Cilj ovog poglavlja je pružiti jasan, sažet i strukturiran pregled arhitekture programskog sustava StanPlan 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:** Sustav je dizajniran koristeći **klijent-poslužitelj (Client-Server)** arhitektonski stil. Backend sustava je implementiran kao **slojeviti monolit (Layered Monolith)**, koristeći Java Spring Boot radni okvir. Ovaj stil je odabran jer pruža robusnu osnovu za razvoj, jasnu podjelu odgovornosti i omogućuje brži razvoj u početnim fazama projekta.

- **Podsustavi:**
  - **Implementirani podsustavi:**
    - **Podsustav za upravljanje korisnicima i zgradama:** Uspostavlja temeljnu strukturu podataka za korisnike (`USERS`), zgrade (`BUILDING`) i gradove (`CITY`), definirajući njihove međusobne odnose. Podržava i lokalnu autentifikaciju (`Username`, `PasswordHash`) i uloge.
    - **Podsustav za autentifikaciju i autorizaciju:** Kombinira **lokalnu prijavu** (korisničko ime/lozinka) i integraciju s vanjskim **OAuth 2.0** providerima (Google, GitHub), što je konfigurirano u Spring Security sloju.
    - **Podsustav za upravljanje sastancima:** Upravlja kompletan životni ciklus sastanaka (entiteti `Meeting`, `Item`, status enumi `MeetingStatus`, `ItemStatus`), od kreiranja u statusu "Planiran" do arhiviranja. Uključuje mogućnost upravljanja točkama dnevnog reda i stanjem sastanka.
    - **Podsustav za notifikacije:** Uključen u procese objave (UC12) i arhiviranja sastanaka (UC15), te slanja poziva na sastanke (UC13), koji triggeriraju e-mail obavijesti suvlasnicima.
    - **Podsustav za integraciju s vanjskim servisima:** Omogućuje integraciju s **StanBlog aplikacijom** putem API-ja (UC22, UC23, UC24), što uključuje konfiguraciju serverskog URL-a, povezivanje točaka s diskusijama, te mogućnost kreiranja sastanaka iz eksterne aplikacije.
- **Preslikavanje na radnu platformu:** Arhitektura je predviđena za implementaciju na **cloud platformama** (npr. Heroku, AWS) ili **lokalnim poslužiteljima**. Backend (Java aplikacija) se pokreće unutar **JVM-a**, a frontend se servira kao statički sadržaj putem web poslužitelja.
- **Spremišta podataka:** Sustav koristi **relacijsku bazu podataka**. Podacima se pristupa putem Spring Data JPA, koji mapira Java entitete u tablice baze podataka definirane SQL shemom. Trenutno se, zbog jednostavne integracije i brze konfiguracije, u razvojnoj i testnoj fazi koristi **H2 baza podataka**. U kasnijim fazama razvoja, planirano je proširenje sustava na **PostgreSQL** kao glavno spremište podataka, čime će se osigurati veća pouzdanost, sigurnost i skalabilnost.
- **Mrežni protokoli:** Komunikacija između klijenta i poslužitelja odvija se putem **HTTP/S protokola**. Backend izlaže svoje funkcionalnosti putem **REST API**-ja, a podaci se razmjenjuju u **JSON** formatu.
- **Globalni upravljački tok (primjer):** Korisnik se prijavljuje putem klijentske aplikacije. Zahtjev za prijavu šalje se na backend. Spring Security obrađuje zahtjev, provjerava `Username` i `PasswordHash` u `USERS` tablici. Ako je prijava uspješna, generira se sesija. Korisnik zatim dohvaća podatke o svojoj zgradi (`GET /api/buildings/{id}`), što `Controller` putem `Repository` sloja dohvaća podatke iz `BUILDING` i `USERS` tablica i vraća ih kao JSON odgovor.
- **Slojevi implementacije:**
  - **Controller sloj:** Implementiran - obrađuje HTTP zahtjeve, validira unos i vraća JSON odgovore (`AdminController`, `CoOwnerController`, `UserController`). AdminController koristi `@PreAuthorize("hasRole('ADMIN')")` za sigurnost.
  - **Service sloj:** Djelomično implementiran - `CoOwnerService` s metodama `createCoOwner()` i `emailPresent()` za upravljanje korisnicima. Potrebno je proširiti s dodatnim servisima za sastanke, notifikacije i StanBlog integraciju.
  - **Repository sloj:** Parcijalno implementiran (`CoOwnerRepository` vidljiv); arhitektura predviđa JPA repositories za sve entitete prema standardnom Spring Data JPA pristupu.
  - **Entity sloj:** Potpuno implementiran - sadrži JPA entitete (`CoOwner`, `Building`, `Meeting`, `Item`) s detaljnim mapiranjima: `@ManyToOne`, `@ManyToMany`, `@OneToOne` relacijama,

`@EmbeddedId` za kompozitne ključeve (Item), unique constraintima na poslovnoj razini (`time_space`, `building_time`, `item_title`), te enum tipovima (`MeetingStatus`, `ItemStatus`, `RoleType`).

- **Sklopovsko-programski zahtjevi:**
  - **Backend:** Java Development Kit (JDK) 21 (kao definirano u pom.xml).
  - **Frontend:** Moderni web preglednik, Vite build tool, React/JSX komponente.
  - **Baza podataka:** H2 za razvoj, PostgreSQL za produkciju.

Obrazloženje odabira arhitekture

- **Izbor arhitekture temeljen na principima oblikovanja:**
  - **Odvajanje briga (Separation of Concerns):** Arhitektura je postavljena tako da se koristi pojednostavljeni slojeviti pristup (Controller, Repository, Entity), gdje Controller obrađuje HTTP zahtjeve, a pristup podacima ide direktno kroz Repository sloj do entiteta.
  - **Jednostavnost:** Odabir monolitne arhitekture je primjeren za opseg projekta, izbjegavajući nepotrebnu kompleksnost mikroservisa.
- **Razmatrane alternative:** Mikrouslužna arhitektura je odbačena kao prekompleksna za ovu fazu projekta.

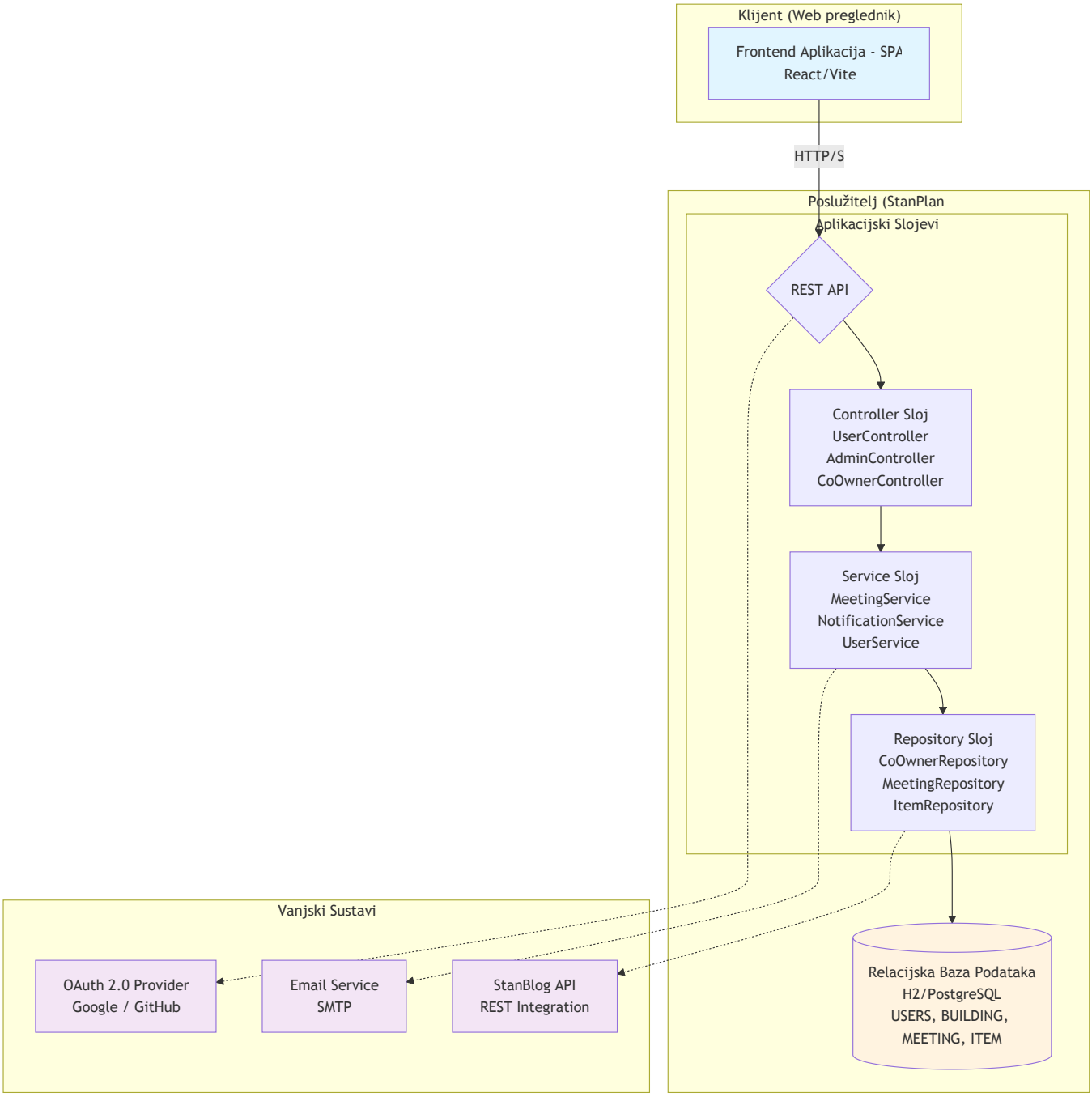
Organizacija sustava na visokoj razini

- **Klijent-poslužitelj: Klijent** je SPA aplikacija u pregledniku. **Poslužitelj** je Spring Boot aplikacija koja upravlja podacima i sigurnošću.
- **Baza podataka:** Centralno spremište podataka, čija je struktura precizno definirana SQL shemom i kojom se upravlja putem JPA entiteta.
- **Grafičko sučelje:** SPA aplikacija koja komunicira s poslužiteljem putem REST API-ja.

Organizacija aplikacije

- **Frontend i Backend slojevi:** Aplikacija je strogo podijeljena na backend i frontend, koji komuniciraju isključivo preko definiranog API-ja.
- **MVC arhitektura (u kontekstu REST API-ja):**
  - **Model:** JPA entiteti koji odgovaraju SQL shemi (`CoOwner`, `Building`, `City`, `Meeting`, `Item`), s kompozitnim ključevima (`MeetingItemId` za Item), enum tipovima (`MeetingStatus`, `ItemStatus`, `RoleType`), te JPA relacijama (`@ManyToOne`, `@ManyToMany`, `@OneToOne`).
  - **View:** JSON odgovori - Spring Boot automatski serializira JPA entitete u JSON pomoću Jackson biblioteke.
  - **Controller:** REST kontroleri (`AdminController`, `CoOwnerController`, `UserController`) koji primaju HTTP zahtjeve, pozivaju Service sloj, i vraćaju ResponseEntity s HTTP statusima i JSON payloadima.

Dijagram visoke razine



Dijagram prikazuje cjelokupnu arhitekturu sustava na visokoj razini. Kroz njega je vidljivo kako Single Page Application (SPA) frontend aplikacija, pokrenuta u web pregledniku, komunicira s poslužiteljem putem HTTP/S protokola. Poslužitelj (backend) implementiran je kao slojevita aplikacija: REST API prima zahtjeve te ih prosljeđuje kroz Controller i Repository slojeve prema relacijskoj bazi podataka. Dijagram također prikazuje povezivost s vanjskim sustavima, konkretno OAuth 2.0 providerom koji omogućuje autentifikaciju korisnika putem vanjskih servisa.

Baza podataka

Sustav koristi relacijsku bazu podataka, a struktura je precizno definirana SQL shemom. Sustav upravljač je strukturom koja uključuje tablice za korisnike, zgrade, gradove, kao i dodatne tablice za upravljanje sastancima i točkama dnevnog reda. Sve tablice su međusobno povezane kako bi osigurale integritet podataka i konzistentnost poslovne logike.

Opis tablica

Tablica: CO_OWNER

Atribut	Tip podatka	Opis varijable
co_owner_id (PK)	SERIAL	Jedinstveni identifikator korisnika
username	VARCHAR(20)	Korisničko ime (jedinstveno)
passwd	VARCHAR(100)	Hesirana lozinka za lokalnu prijavu
first_name	VARCHAR(20)	Ime korisnika
last_name	VARCHAR(20)	Prezime korisnika
email	VARCHAR(100)	Adresa elektroničke pošte (jedinstvena)
role_type	VARCHAR(20)	Uloga korisnika ('ADMIN', 'PREDSTAVNIK', 'SUVLASNIK')
building_id (FK)	INTEGER	Poveznica na zgradu kojoj korisnik pripada

Tablica: MEETING

Atribut	Tip podatka	Opis varijable
meeting_id (PK)	SERIAL	Jedinstveni identifikator sastanka
building_id (FK)	INTEGER	Poveznica na zgradu kojoj sastanak pripada
title	VARCHAR(50)	Naslov sastanka
summary	VARCHAR(500)	Sažetak namjere sastanka
meeting_start_time	TIMESTAMP	Vrijeme početka sastanka
meeting_end_time	TIMESTAMP	Vrijeme završetka sastanka
meeting_location	VARCHAR(100)	Lokacija održavanja sastanka
status	VARCHAR(20)	Status sastanka (Planiran, Objavljen, Obavljen, Arhiviran)

Unique Constraints:

- `time_space`: Kombinacija (meeting_start_time, meeting_location) - sprječava dvostruko bukiranje prostora
- `building_time`: Kombinacija (meeting_start_time, building_id) - osigurava da zgrada ima samo jedan sastanak u isto vrijeme

Tablica: ITEM (Točka dnevnog reda)

Atribut	Tip podatka	Opis varijable
meeting_id (PK, FK)	INTEGER	Dio kompozitnog ključa, poveznica na sastanak
item_number (PK)	INTEGER	Dio kompozitnog ključa, redni broj točke

Atribut	Tip podatka	Opis varijable
title	VARCHAR(50)	Naslov točke dnevnog reda
summary	VARCHAR(500)	Detaljni opis točke
status	VARCHAR(20)	Status točke (Planiran, Izglasan, Odbijen, Neriješen)
legal	INTEGER	Indikator pravnog učinka (0 = nema, 1 = ima)
conclusion	VARCHAR(500)	Zaključak nakon obrade točke

Unique Constraint:

- `item_title`: Kombinacija (meeting_id, title) - osigurava da točke imaju jedinstvena imena unutar sastanka

Tablica: PARTICIPATION (sudjelovanje u sastanku)

Atribut	Tip podatka	Opis varijable
meeting_id (PK, FK)	INTEGER	Dio kompozitnog ključa, poveznica na sastanak
co_owner_id (PK, FK)	INTEGER	Dio kompozitnog ključa, poveznica na suvlasnika

Tablica: CITY

Atribut	Tip podatka	Opis varijable
city_id (PK)	SERIAL	Jedinstveni identifikator grada
postal_code	INTEGER	Poštanski broj (jedinstven)
city_name	VARCHAR(50)	Naziv grada

Tablica: BUILDING

Atribut	Tip podatka	Opis varijable
building_id (PK)	SERIAL	Jedinstveni identifikator zgrade
city_id (FK)	INTEGER	Poveznica na grad u kojem se zgrada nalazi
address	VARCHAR(100)	Adresa zgrade
rep_id (FK)	INTEGER	ID predstavnika suvlasnika (jedinstven, veza na CO_OWNER.co_owner_id)

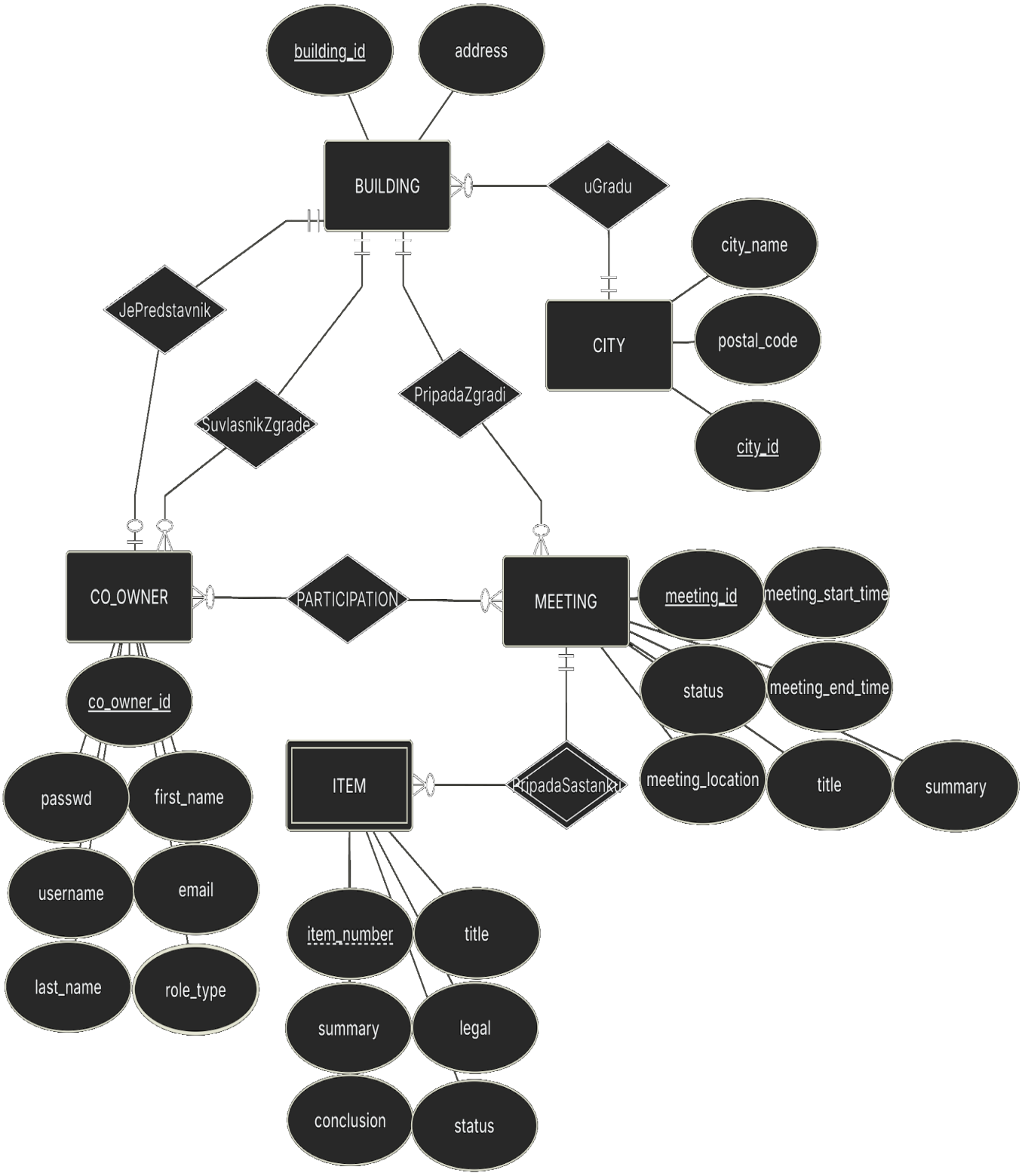
Unique Constraint:

- `city_address`: Kombinacija (city_id, address) - osigurava da se adresa ne ponavlja unutar istog grada

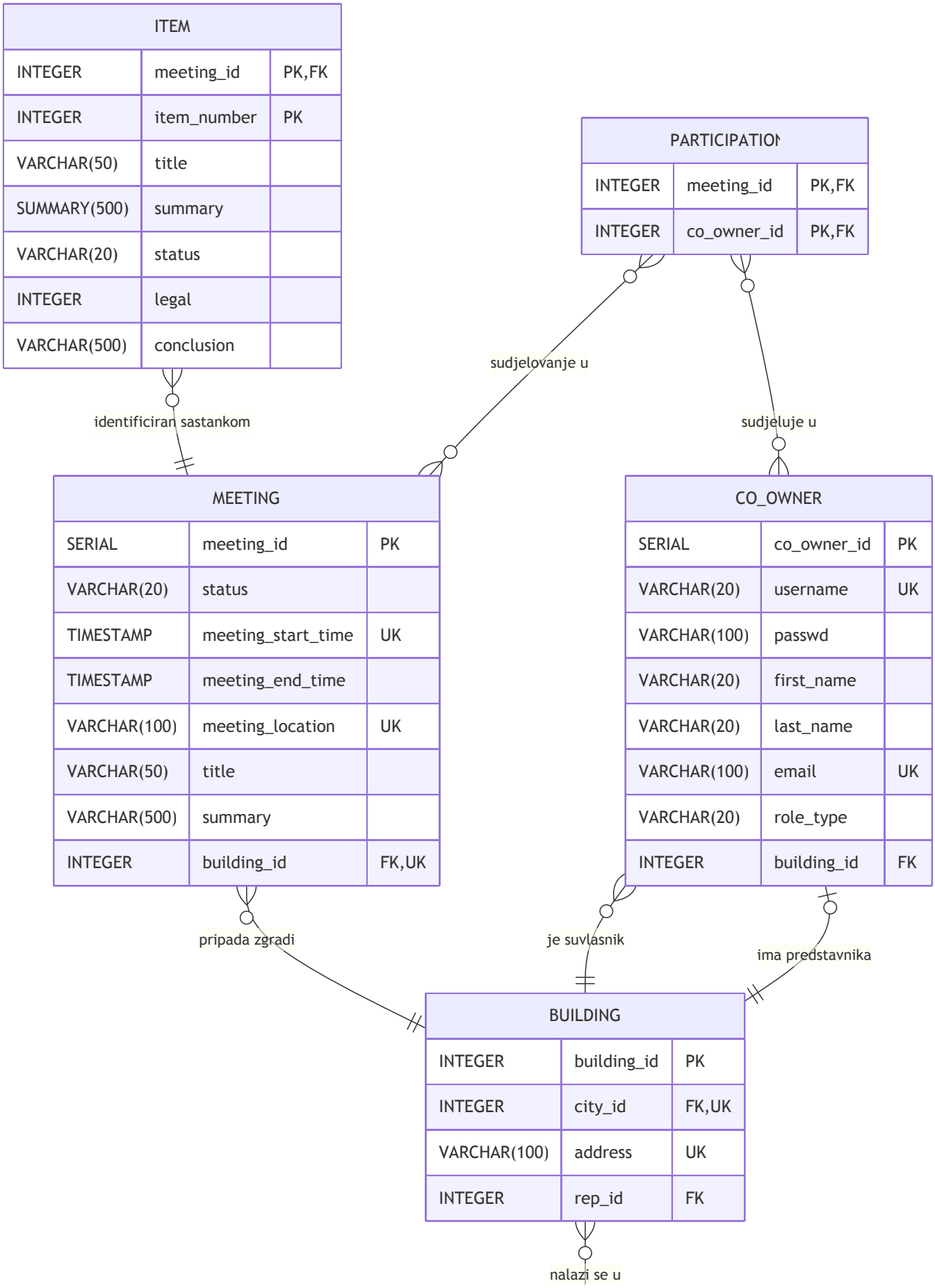
Indeksi:

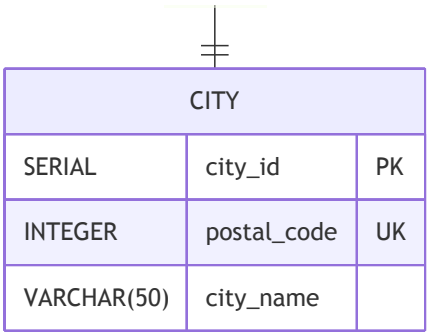
- `i_meeting_by_building`: Na MEETING(building_id) - ubrzava pretragu sastanaka po zgradi
- `i_part_by_coowner`: Na PARTICIPATION(co_owner_id) - ubrzava pretragu sudjelovanja po korisniku
- `i_coowner_by_building`: Na CO_OWNER(building_id) - ubrzava pretragu korisnika po zgradi
- `i_building_by_rep`: Na BUILDING(rep_id) - ubrzava pretragu zgrada po predstavniku

Dijagram baze podataka



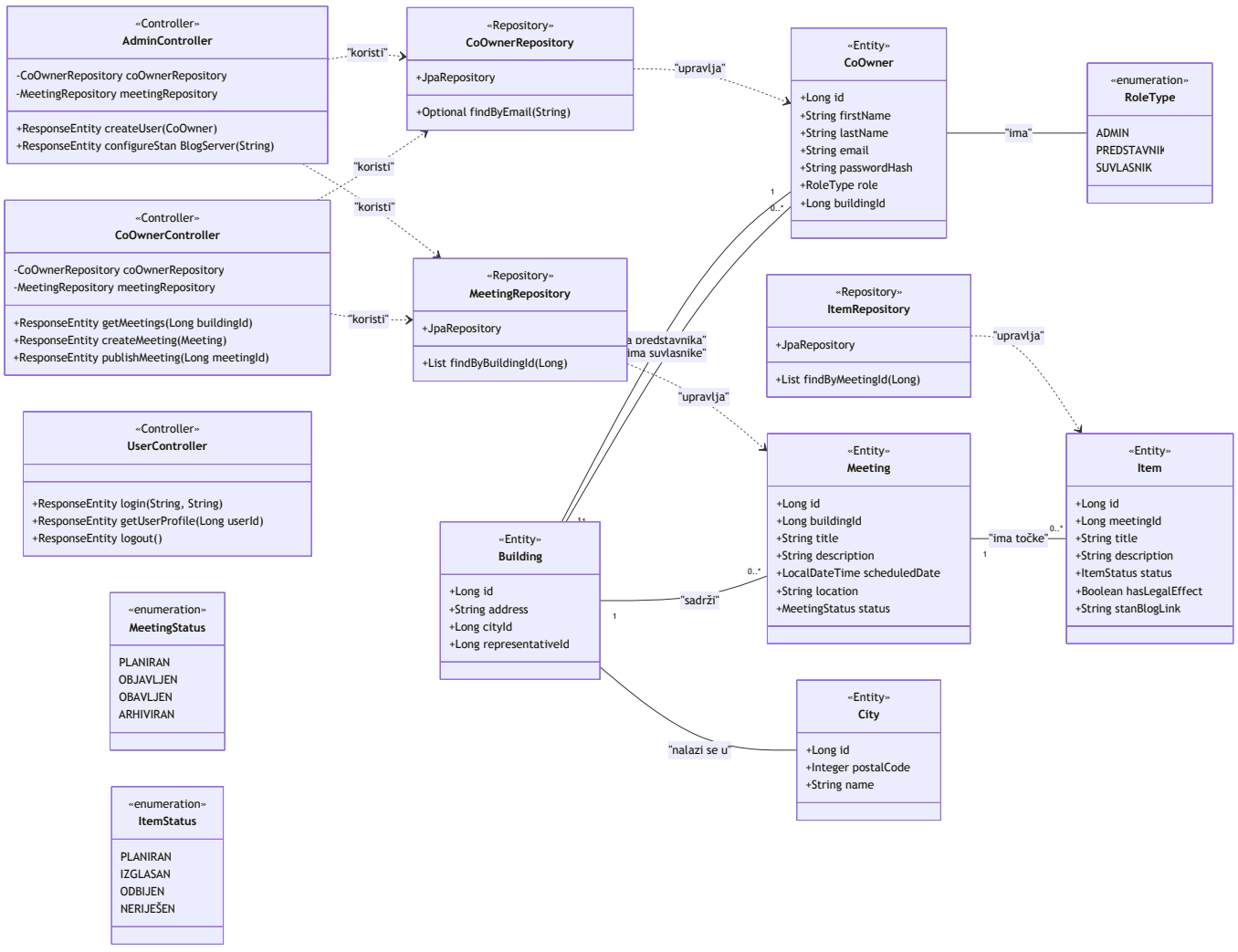
ER dijagram ilustrira logičku strukturu baze podataka. Prikazani su glavni entiteti: CO_OWNER (suvlasnici), BUILDING (zgrade), CITY (gradovi), MEETING (sastanci) i ITEMS(točke dnevnog reda), zajedno s njihovim osnovnim atributima i međusobnim odnosima. Dijagram jasno prikazuje kako suvlasnici pripadaju zgradama, a svaka zgrada nalazi se u određenom gradu. Sastanci pripadaju zgrada i točke dnevnog reda sastancima.





Dijagram prikazuje apstraktnu organizaciju tablica u bazi podataka. Svaka entitetna klasa modelirana je kao tablica s jasno definiranim podacima: nazivima atributa, tipovima podataka te oznakom primarnih (PK) i vanjskih ključeva (FK) te alternativnih ključeva (UK). Prikazane su i relacije između tablica, uključujući vezu između CO_OWNER i BUILDING (korisnik pripada zgradi), BUILDING i CITY (zgrada se nalazi u gradu), BUILDING i CO_OWNER (predstavnik preko rep_id), te MEETING i BUILDING (sastanci pripadaju zgradi), MEETING i ITEM (sastanci sadrže točke dnevnog reda), kao i many-to-many veza preko PARTICIPATION tablice (korisnici potvrđuju sudjelovanje na sastancima).

Dijagram razreda



Dijagram prikazuje backend implementaciju sustava s naglaskom na ključne komponente. Vidljivi su kontroleri (`AdminController`, `CoOwnerController`, `UserController`) koji upravljaju HTTP zahtjevima,

service sloj (`CoOwnerService`) koji implementira poslovnu logiku, repository sloj (`CoOwnerRepository`, `MeetingRepository`, `ItemRepository`) za pristup podacima, te JPA entiteti (`CoOwner`, `Building`, `Meeting`, `Item`, `City`). Prikazani su i enum tipovi za status sastanaka (`MeetingStatus`: Planiran, Objavljen, Obavljen, Arhiviran) i status točaka (`ItemStatus`: Planiran, Izglasan, Odbijen, Neriješen), te uloge korisnika (`RoleType`: ADMIN, PREDSTAVNIK, SUVLASNIK). Dijagram jasno prikazuje odnose korištenja („koristi"), upravljanja („upravlja") i agregacije („sadrži", „ima") među komponentama, uključujući many-to-many relaciju između CoOwner i Meeting preko participation tablice.

dio 1. revizije

Prilikom prve predaje projekta, potrebno je prilo ̌ziti potpuno razraden dijagram razreda vezan uz generičku funkcionalnost sustava. Ostale funkcionalnosti trebaju biti idejno razrađene u dijagramu sa sljedećim komponentama: nazivi razreda, nazivi metoda i vrste pristupa metodama (npr. javni, zaštićeni), nazivi atributa razreda, veze i odnosi između razreda.

Napomena: Prilikom druge predaje projekta dijagram razreda i opisi moraju odgovarati stvarnom stanju implementacije

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.

Zadatak:

- Dokumentirajte dva dinamička dijagrama koji prikazuju značajne ili neke složene procese u aplikaciji. Registracija i prijava korisnika nisu ključni procesi u ovom kontekstu, stoga se usmjerite na specifične, značajne procese.

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.

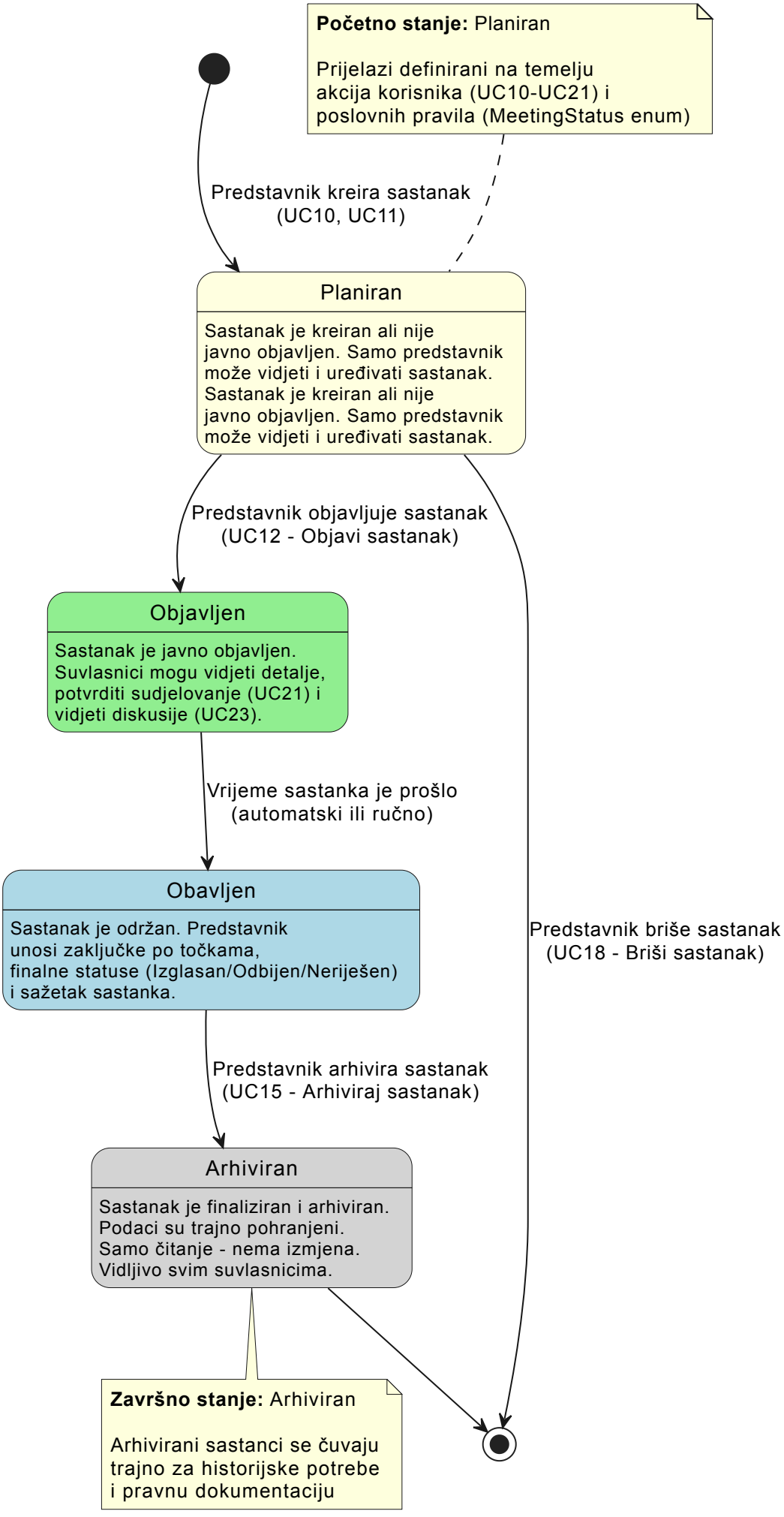
Preporuke za izradu UML dijagrama stanja:

1. Odaberite značajan objekt sustava (npr. prijava štete, obrada zahtjeva).
2. Identificirajte stanja objekta (npr. aktivan, neaktivan, obrisan).
3. Definirajte događaje i uvjete prijelaza.
4. Ako je potrebno, koristite hijerarhiju stanja za složenije probleme.

5. Dodajte bilješke za pojašnjenje važnih prijelaza i elemenata.

> Primjer: Dijagram stanja za prijavu štete može uključivati:
>
> - Stanja: *Kreirana prijava*, *Obrađena prijava*, *Zatvorena prijava*... .

Dijagram stanja: Životni ciklus sastanka


Početno stanje: Planiran
Prijelazi definirani na temelju akcija korisnika (UC10-UC21) i poslovnih pravila (MeetingStatus enum)
Predstavnik kreira sastanak (UC10, UC11)
Planiran
Sastanak je kreiran ali nije javno objavljen. Samo predstavnik može vidjeti i uređivati sastanak.
Sastanak je kreiran ali nije javno objavljen. Samo predstavnik može vidjeti i uređivati sastanak.
Predstavnik objavljuje sastanak (UC12 - Objavi sastanak)
Objavljen
Sastanak je javno objavljen. Suvlasnici mogu vidjeti detalje, potvrditi sudjelovanje (UC21) i vidjeti diskusije (UC23).
Vrijeme sastanka je prošlo (automatski ili ručno)
Obavljen
Sastanak je održan. Predstavnik unosi zaključke po točkama, finalne statuse (Izglasan/Odbijen/Neriješen) i sažetak sastanka.
Predstavnik briše sastanak (UC18 - Briši sastanak)
Predstavnik arhivira sastanak (UC15 - Arhiviraj sastanak)
Arhiviran
Sastanak je finaliziran i arhiviran. Podaci su trajno pohranjeni. Samo čitanje - nema izmjena. Vidljivo svim suvlasnicima.
Završno stanje: Arhiviran
Arhivirani sastanci se čuvaju trajno za historijske potrebe i pravnu dokumentaciju

Opis dijagrama stanja sastanka:

Dijagram prikazuje životni ciklus sastanka (`Meeting` entitet) kroz četiri glavna stanja definirana `MeetingStatus` enumeracijom:

Stanja:

1. **Planiran** - Početno stanje kada predstavnik kreira sastanak (UC10, UC11). Sastanak je vidljiv samo predstavniku koji ga može uređivati (dodavanje/brisanje točaka dnevnog reda). U ovom stanju sastanak još nije javno objavljen suvlasnicima.

2. **Objavljen** - Sastanak je javno objavljen (UC12). Svi suvlasnici zgrade mogu vidjeti detalje sastanka, potvrditi sudjelovanje (UC21) i pristupiti povezanim diskusijama iz StanBlog aplikacije (UC23). Email notifikacije su poslane svim suvlasnicima s detaljima sastanka.

3. **Obavljen** - Sastanak je održan i predstavnik unosi zaključke. U ovom stanju predstavnik popunjava zaključke za svaku točku dnevnog reda, postavlja finalne statuse točaka (`ItemStatus`: Izglasan/Odbijen/Neriješen) i piše sažetak sastanka.

4. **Arhiviran** - Završno stanje (UC15). Sastanak je finaliziran i arhiviran kao trajni zapis. Podaci su samo za čitanje - nema mogućnosti izmjene. Arhivirani sastanci služe za historijske potrebe, pravnu dokumentaciju i transparentnost prema suvlasnicima.

Prijelazi:

- **Planiran → Objavljen**: Predstavnik objavljuje sastanak (UC12) - trigerira slanje email notifikacija
- **Objavljen → Obavljen**: Automatski prijelaz nakon što vrijeme sastanka prođe, ili ručno od strane predstavnika
- **Obavljen → Arhiviran**: Predstavnik finalizira i arhivira sastanak (UC15). Ne može biti arhiviran ako nisu riješene pravne točke.

Dijagram jasno prikazuje jednosmjerne prijelaze koji osiguravaju integritet podataka i konzistentnost poslovne logike - sastanak ne može biti "de-arhiviran" ili vraćen u prethodna stanja, što osigurava trajnost dokumenata.

UML dijagrami aktivnosti

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

Zadatak:

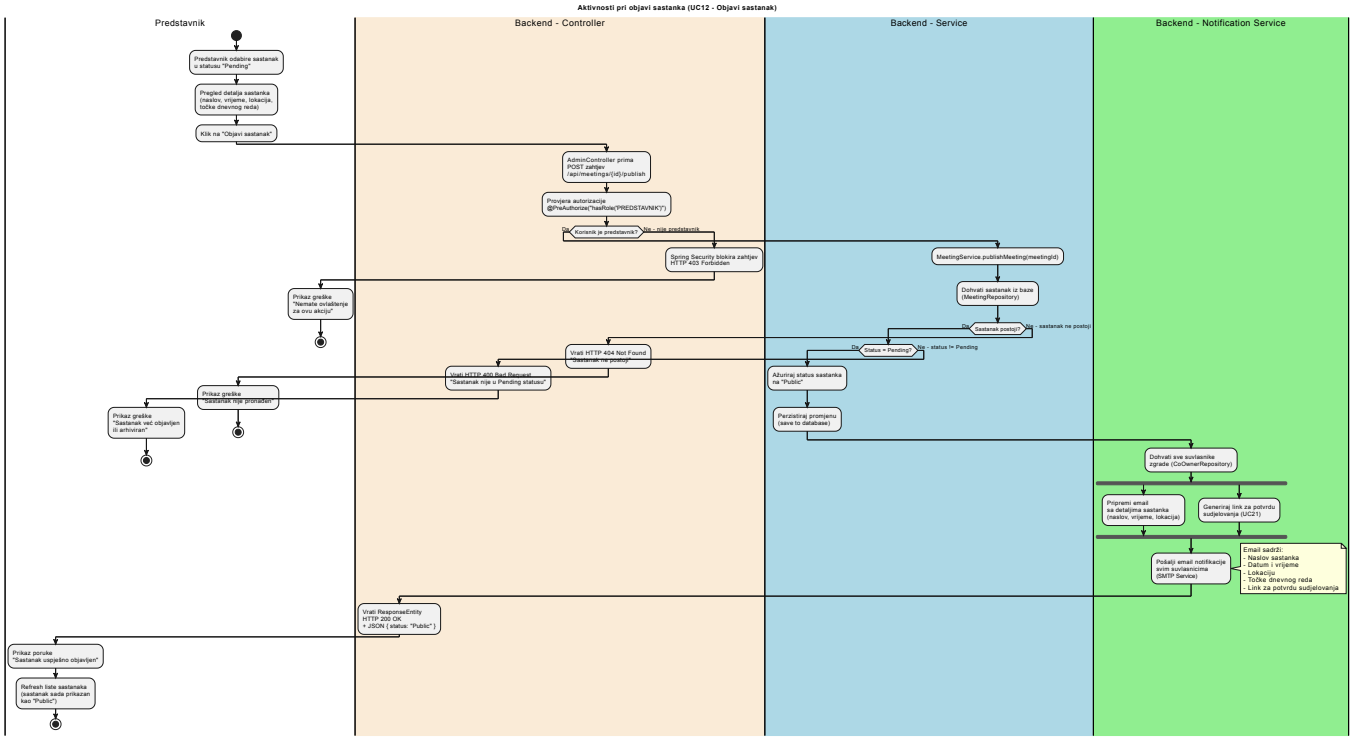
1. Identificirajte proces koji želite modelirati (npr. obrada zahtjeva).

2. Razlomite proces na aktivnosti i povežite ih za prikaza slijeda izvršavanja.

3. Upotrijebite pseudo čvorove za jasan prikaz.

> Primjer:** Na primjer, u aplikaciji za upravljanje štetama, dijagram aktivnosti može prikazivati proces obrade zahtjeva za pomoć, što uključuje aktivnosti: *Primanje zahtjeva*, *Analiza podataka*, *Odobravanje resursa*, *Slanje pomoći*.

Dijagram aktivnosti: Objava sastanka (UC12)



Opis dijagrama aktivnosti - Objava sastanka:

Dijagram prikazuje kompletan tijek aktivnosti za slučaj uporabe UC12 "Objavi sastanak", koji je kritičan proces u sustavu jer uključuje:

- Autentifikaciju i autorizaciju korisnika
- Validaciju poslovnih pravila
- Transakcijsku promjenu stanja u bazi podataka
- Notifikacije prema vanjskim servisima (SMTP email)

Glavni akteri i komponente:

1. **Predstavnik** - Korisnik koji inicira objavu sastanka
2. **Backend Controller** (`AdminController`) - Prima HTTP zahtjev, provjerava autorizaciju
3. **Backend Service** (`MeetingService`) - Implementira poslovnu logiku
4. **Backend Notification Service** - Šalje email notifikacije suvlasnicima
5. **Repository sloj** - Pristup bazi podataka (`MeetingRepository`, `CoOwnerRepository`)

Tijek aktivnosti:

1. **Inicijacija (Predstavnik):**
   - Predstavnik odabire sastanak u statusu "Pending"
   - Pregleda detalje sastanka (naslov, vrijeme, lokaciju, točke dnevnog reda)
   - Klikne na "Objavi sastanak"
2. **Autorizacija (Controller sloj):**
   - `AdminController` prima POST zahtjev na `/api/meetings/{id}/publish`
   - Spring Security provjera: `@PreAuthorize("hasRole('PREDSTAVNIK')")`

- Ako korisnik nema odgovarajuću ulogu → HTTP 403 Forbidden

3. **Validacija (Service sloj):**
   - `MeetingService.publishMeeting(meetingId)` dohvaća sastanak iz baze
   - Provjera postojanja sastanka (404 Not Found ako ne postoji)
   - Provjera statusa - mora biti "Pending" (400 Bad Request ako nije)
4. **Promjena stanja (Repository sloj):**
   - Ažuriranje statusa sastanka s "Pending" na "Public"
   - Perzistiranje promjene u bazu podataka (SQL UPDATE)
   - Transakcijska sigurnost osigurana Spring Data JPA transakcijama
5. **Notifikacije (Notification Service):**
   - Dohvat svih suvlasnika zgrade (`CoOwnerRepository.findByBuildingId()`)
   - Priprema email sadržaja s detaljima sastanka i linkom za potvrdu sudjelovanja
   - Slanje email notifikacija svim suvlasnicima putem SMTP servisa (paralelno)
6. **Odgovor i prikaz:**
   - Backend vraća HTTP 200 OK s JSON statusom { "status": "Public" }
   - Frontend prikazuje poruku "Sastanak uspješno objavljen"
   - Refresh liste sastanaka - sastanak sada prikazan kao "Public"

Alternativni tijekovi (iznimke):

- **Neautoriziran korisnik**: Spring Security blokira zahtjev s HTTP 403
- **Sastanak ne postoji**: Vraća se HTTP 404 Not Found
- **Pogrešan status**: Vraća se HTTP 400 Bad Request ako sastanak nije u "Pending" statusu

Dijagram jasno prikazuje kako backend arhitektura (Controller → Service → Repository) osigurava odvajanje briga, validaciju na više razina i robusnu obradu grešaka.

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.

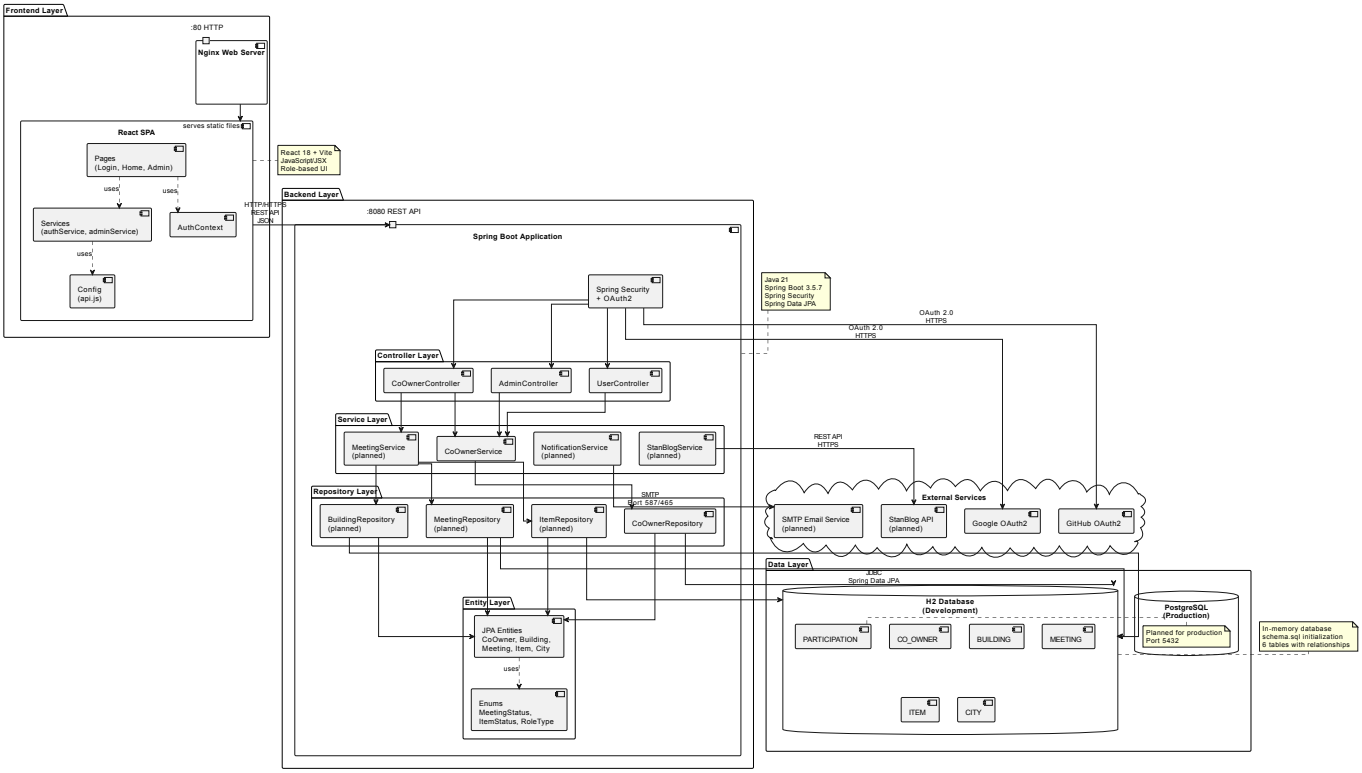
Arhitektura sustava, u kontekstu dijagrama komponenata i razmještaja, pruža uvid u strukturu i raspored ključnih dijelova aplikacije. Ovi dijagrami nisu korisni samo tijekom faza oblikovanja i implementacije, već služe i kao alati za održavanje i optimizaciju sustava u budućnosti.

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.



Opis komponenata sustava StanPlan

Frontend Layer:

- **React SPA** - Moderna single-page aplikacija koja koristi React 18 i Vite build tool. Organizirana je u AuthContext za upravljanje autentifikacijom, Pages komponente (Login, Home, Admin) za role-based view-ove, Services (authService, adminService) za komunikaciju s backendom putem HTTP klijenata, te Config modul za konfiguraciju API endpointa.
- **Nginx Web Server** - Poslužuje statičke React build fileove i omogućuje SPA routing fallback (try_files $uri /index.html) na portu 80.

Backend Layer:

- **Spring Boot Application** - Glavni backend poslužitelj (port 8080) sastavljen od više slojeva:
  - **Spring Security + OAuth2** - Komponenta za autentifikaciju i autorizaciju s integracijom Google i GitHub OAuth2 providera te role-based access control (@PreAuthorize anotacije za ADMIN, PREDSTAVNIK, SUVLASNIK uloge).

- **Controller Layer** - REST API kontroleri: AdminController (kreiranje korisnika - samo ADMIN), CoOwnerController (upravljanje sastancima i profil), UserController (autentifikacija i odjava).
- **Service Layer** - Poslovna logika: CoOwnerService (implementiran s createCoOwner() i emailPresent() metodama), planirani MeetingService, NotificationService i StanBlogService.
- **Repository Layer** - Pristup podacima: CoOwnerRepository (JpaRepository<CoOwner, Integer>), planirani MeetingRepository, ItemRepository, BuildingRepository.
- **Entity Layer** - JPA entiteti s relacijama (@ManyToOne, @ManyToMany, @OneToOne): CoOwner, Building, Meeting, Item, City, te enumeracije MeetingStatus, ItemStatus, RoleType.

Data Layer:

- **H2 Database (Development)** - In-memory relacijska baza s 6 tablica (CO_OWNER, BUILDING, CITY, MEETING, ITEM, PARTICIPATION) inicijalizirana iz schema.sql. Koristi se za razvoj i testiranje.
- **PostgreSQL (Production)** - Planirana produkcijska baza podataka (port 5432) s istom shemom kao H2, ali s perzistencijom i naprednim značajkama (automatic backups, read replicas).

External Services:

- **Google OAuth2 i GitHub OAuth2** - Vanjski provideri autentifikacije preko OAuth 2.0 protokola putem HTTPS-a.
- **SMTP Email Service** - Planirana integracija s email providerima (SendGrid/AWS SES) za slanje obavijesti o sastancima (UC12, UC13, UC15).
- **StanBlog API** - Planirana integracija s vanjskim REST API-jem za konfiguraciju URL-a (UC22), dohvat diskusija (UC23) i kreiranje sastanaka (UC24).

Komunikacijska sučelja:

- Frontend ↔ Backend: HTTP/HTTPS REST API, JSON format, JWT tokeni za autentifikaciju
- Backend ↔ Database: JDBC protokol, Spring Data JPA/Hibernate ORM
- Backend ↔ External Services: OAuth 2.0 (HTTPS), SMTP (port 587/465), REST API (HTTPS)

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.

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

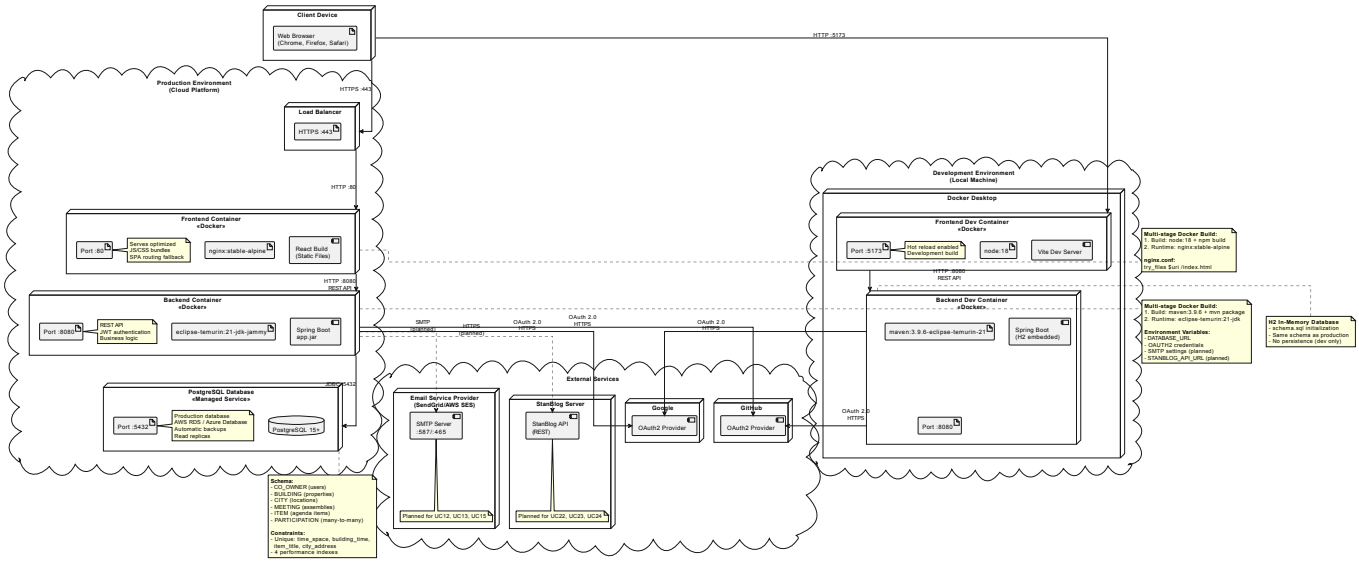
- Dijagram razmještaja instanci prikazuje način na koji su aplikacijske komponente raspoređene unutar infrastrukture, uključujući fizičke i virtualne čvorove (poslužitelje) na kojima se izvode. Ovaj dijagram detaljno opisuje kako su komponente povezane i kako međusobno komuniciraju.

> 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.

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).



Opis arhitekture razmještaja sustava StanPlan

Sustav StanPlan dizajniran je za deployment u kontejneriziranom okruženju koristeći Docker tehnologiju, što omogućuje fleksibilnost u izboru deployment platforme (lokalni serveri, cloud platforme poput AWS, Azure, Heroku). Arhitektura razlikuje razvojno i produkcijsko okruženje.

Razvojno okruženje (Development Environment):

Razvojno okruženje pokrenuto je lokalno na razvojnom računalu koristeći Docker Desktop:

- **Client Device** - Razvojno računalo s web preglednikom pristupa aplikaciji na http://localhost:5173
- **Frontend Dev Container (Docker)** - Kontejner baziran na node:18 image-u koji pokreće Vite Dev Server na portu 5173. Hot reload je omogućen za brži razvoj, a VITE_API_BASE_URL je konfiguriran na http://localhost:8080.
- **Backend Dev Container (Docker)** - Kontejner baziran na maven:3.9.6-eclipse-temurin-21 image-u koji pokreće Spring Boot aplikaciju s embedded H2 bazom (in-memory) na portu 8080. Koristi application.yml konfiguraciju za razvoj.
- **External Services** - Razvojno okruženje se povezuje na stvarne OAuth2 providere (Google i GitHub) putem HTTPS-a za testiranje autentifikacije.

Komunikacija: Client → Frontend Dev Container :5173 → Backend Dev Container :8080 (REST API) → H2 In-Memory Database

Produkcijsko okruženje (Production Environment):

Produkcijsko okruženje deploya se na cloud platformu (AWS, Azure, Heroku ili slično) s većom pouzdanošću, skalabilnošću i sigurnošću:

- **Client Device** - Krajnji korisnik pristupa aplikaciji putem https://stanplan.example.com preko bilo kojeg modernog web preglednika.
- **Load Balancer** - Cloud load balancer prima HTTPS zahtjeve na portu 443 i distribuira promet prema frontend kontejneru. Osigurava TLS/SSL terminaciju.
- **Frontend Container (Docker)** - Kontejner baziran na nginx:stable-alpine image-u koji poslužuje optimizirane React build artifakte (JS/CSS bundleove) na portu 80. Multi-stage Docker build: prvo node:18 image buildira aplikaciju s npm build, zatim nginx image poslužuje statičke fileove. Nginx je konfiguriran s SPA routing fallbackom (try_files $uri /index.html).
- **Backend Container (Docker)** - Kontejner baziran na eclipse-temurin:21-jdk-jammy image-u koji izvršava Spring Boot app.jar na portu 8080. Multi-stage Docker build: prvo maven:3.9.6 image kompajlira kod s mvn package, zatim JRE image pokreće JAR file. Environment varijable uključuju DATABASE_URL (PostgreSQL connection string), OAUTH2 credentials (Google, GitHub), SMTP postavke (planirano) i STANBLOG_API_URL (planirano).
- **PostgreSQL Database (Managed Service)** - Managed database service (npr. RenderDB for PostgreSQL) s PostgreSQL 15+ na portu 5432. Baza sadrži 6 tablica (CO_OWNER, BUILDING, CITY, MEETING, ITEM, PARTICIPATION) s unique constraintima (time_space, building_time, item_title, city_address) i 4 performance indexa. Podržava automatic backups, high availability i read replice za skalabilnost.
- **External Services:**
  - **Google OAuth2 / GitHub OAuth2** - Vanjski OAuth 2.0 provideri za autentifikaciju korisnika putem HTTPS-a. Callback URL: https://stanplan.example.com/login/oauth2/code/{provider}
  - **SMTP Email Server** - Planirana integracija s email providerima (SendGrid, AWS SES, Mailgun) na portu 587 (STARTTLS) ili 465 (SSL) za slanje obavijesti o sastancima (UC12, UC13, UC15).
  - **StanBlog API** - Planirana integracija s vanjskim REST API-jem putem HTTPS-a za konfiguraciju URL-a (UC22), dohvat diskusija (UC23) i API kreiranje sastanaka (UC24).

Mrežna arhitektura i sigurnost:

- **TLS/SSL** - Enforced na razini load balancera za svu komunikaciju između klijenta i aplikacije (HTTPS :443).
- **Private Network** - Backend ↔ Database komunikacija odvija se preko privatne VPC mreže (bez javnog pristupa).
- **Firewall Rules** - Port 80/443 otvoren na load balanceru, port 8080 i 5432 dostupni samo interno unutar privatne mreže.
- **CORS** - Konfiguriran u Spring Security za dozvoliti zahtjeve s frontend origina.

Deployment strategija:

Build proces: Frontend (npm run build → Vite generira optimizirane bundleove → kopiraju se u Nginx container), Backend (mvn clean package → Maven kompajlira Java kod → kreira Spring Boot executable JAR → kopira se u JRE container).

Preporučuje se Continuous Deployment (CI/CD) putem GitHub Actions ili GitLab CI: automatski build i deploy na commit u main branch, push Docker image-a u Container Registry (Docker Hub, AWS ECR, GitHub Container Registry), deployment na cloud platform (Heroku, AWS ECS, Azure App Service).

Skalabilnost i monitoring:

- Horizontalno skaliranje: Frontend i backend kontejneri su stateless i mogu se skalirati s više instanci iza load balancera. PostgreSQL podržava read replice za distribuiranje read operacija.
- Monitoring: Spring Boot Actuator za health checks i metrics, cloud platform monitoring (AWS CloudWatch, Azure Monitor), centralizirani logging (ELK stack).
- Disaster Recovery: Automatski database backupi (daily snapshots), multi-region deployment za kritične sustave, rollback strategija putem container versioning.

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

Ispitivanje komponenti

1. Pokretanje stranice

- Pokreće li se stranica?
- Ulazni podaci: nema ih
- Očekivani podaci: naslov stranice StanPlan
- Postupak:
    - Otvori se stranica

2. Ulogiravanje s admin korisnikom

- provjeravamo da su sustavi provjeravanja emaila funkcionalni
- ulazni podaci:
    - Email: brunoplese0@gmail.com
    - Lozinka: password123
- Očekivani izlaz: vraćeni smo na početnu stranicu prijavljeni kao admin
- Postupak:
    - Otvoriti stranicu
    - Stisnuti na gumb "Prijavi se"
    - Upisati zadane podatke za email i password
    - Prijaviti se

3. Provjerava da se ne može ući s neispravnim mailom/korisničkim imenom i lozinkom

- Onemugoćava se pristup korisnicima s neispravnim mailom/korisničkim imenom i lozinkom
- ulazni podaci.
    - Email: bruawdasdawnoplese0@gmail.com
    - Lozinka: password123
- Očekivani izlaz: Nevažeći podaci za prijavu. Molimo pokušajte ponovno.
- Postupak:
    - Otići na StanPlan stranicu
    - Stisnuti na "Prijavu"

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.

Zadaci:

1. Kreiranje korisnika

- Omogućuje se privilegiranim osobama (adminima) kreiranje korisnika
- ulazni podaci:
    - Email: luka.sever20@gmail.com
    - Ime: L
    - Prezime: S
    - Korisničko ime: lukas
    - Lozinka: password123
    - Potvrdi lozinku: password123
    - Odabra se jedna od zgrada
    - Izabere se jedan od uloga, predstavnik ili suvlasnik
- Očekivani izlaz: korisnik dobiva poruku "Korisnik lukas je uspješno kreiran!"
- Postupak:
    - Otići na glavnu stranicu
    - Prijaviti se s jednim od računa koji ima admin privilegije
    - Upisati sve podatke u za to predviđena polja
    - Stisnuti "Kreiraj korisnika"

2. Dodavanje osobe zgradi

- Admin može dodai osobu zgradi
- Ulazni podaci:
    - Odarbana zgrada: (jedna od opcija)
    - Email: brunoplese0@gmail.com
- Očekivani izlaz: Poruka korisniku "Predstavnik uspješno dodan zgradi!"
- Postupak:
    - Ulogiravanje s admin privilegijama
    - Pritiskom gumba "Admin" odlazi se na admin stranicu gdje se može predstavnik dodijeliti zgradi
    - Upišu se podaci
    - Vrati se poruka korisniku da je predstavnik ni/je dodan zgradi

3. Testiranje komunikacije sa StanBlogom

- šalje se GET zahtjev za listom diskusija po kojima se kreiraju sastanci
- Ulazni podaci:
    - Header s API ključem koji omogućuje komunikaciju
- Očekivani izlaz: Lista diskusija u zadanom formatu
- Postupak:
    - šalje se GET zahtjev sa zadanim parametrima u headeru sa web sjedišta https://progistanblog.azurewebsites.net/api/stanplan/discussions/positive
    - u odgovoru su vraćni svi parametri za diskusiju

1. Razviti minimalno **6 ispitnih slučajeva** koji obuhvaćaju:

    - **Redovne slučajeve**: testiranje uobičajenog ponašanja funkcionalnosti.
    - **Rubne uvjete**: provjera ulaznih podataka na granici valjanosti.
    - **Izazivanje pogreške (exception throwing)**: testiranje reakcije na iznimke.

- **Nepostojeće funkcionalnosti**: provjera reakcije na poziv neimplementirane funkcionalnosti.

Struktura ispitivanja:

Za svaki ispitni slučaj potrebno je:

1. Opišite funkcionalnost koju testirate (npr. dodavanje korisnika, validacija podataka).
2. Navedite ispitni slučaj:
   - Ulazne podatke.
   - Očekivane rezultate.
   - Dobivene rezultate (prolaz/pad ispitivanja).
3. Opišite postupak prvođenja ispitivanja
4. U Gitu moraju biti dostupni izvorni kodovi ispitnih slučajeva.

**Ispitivanje sustava **

IS-1: Kreiranje sastanka (REP → automatska dodjela zgrade)

- **Funkcionalnost**: REP kreira sastanak; backend mora dodijeliti sastanku zgradu prijavljenog REP-a.
- **Ulazi**: - REP korisnik: `rep1@test.com` / `password123`, uloga `REP`, pripada zgradi `b1` - API poziv: `POST /api/meetings/newMeeting` - Payload: `title`, `summary`, `meetingLocation`, `meetingStartTime`, `meetingEndTime`, `status`, `items`
- **Koraci**: - U testu se kreira zgrada `b1`. - U testu se kreira REP korisnik vezan uz `b1`. - Šalje se HTTP `POST /api/meetings/newMeeting` s Basic Auth.
- **Očekivani rezultat**: - HTTP 2xx - Vraćen `Meeting` s `meetingId != null` - `meeting.building.buildingId == b1.buildingId`
- **Dobiveni rezultat**: **PASS**

IS-2: Dodavanje prve točke dnevnog reda (rubni uvjet: prva točka)

- **Funkcionalnost**: Dodavanje prve točke dnevnog reda u sastanak koji još nema nijednu točku.
- **Ulazi**: - REP korisnik (Basic Auth) - API poziv: `POST /api/meetings/{meetingId}/items` - Payload: `title`, `summary` (namjerno bez `legal` i `status`)
- **Koraci**: - U testu se kreira sastanak bez točaka (`items` prazno). - Šalje se `POST .../items` s payloadom bez `legal/status`.
- **Očekivani rezultat**: - HTTP 2xx - `items.size == 1` - `itemNumber == 1` (rubni slučaj – prva točka) - Default vrijednosti: `legal == 0`, `status == Pending`
- **Dobiveni rezultat**: **PASS**

IS-3: Objavljivanje sastanka – mail se šalje samo korisnicima iste zgrade

- **Funkcionalnost**: Kod objave sastanka, mail obavijest smiju dobiti samo korisnici u istoj zgradi kao i sastanak.

- **Ulazi**: - Zgrade: `b1`, `b2` - Korisnici: - REP u `b1` - CO_OWNER u `b1`: `co1@test.com` - CO_OWNER u `b2`: `co2@test.com` - API poziv: `POST /api/meetings/{id}/publish`
- **Koraci**: - Kreira se sastanak vezan uz `b1`. - Poziva se `/publish` kao REP (`withBasicAuth`). - Preko Mockito (`@SpyBean EmailService`) verificira se kome je "poslan" mail.
- **Očekivani rezultat**: - HTTP 2xx - Status sastanka postaje `Public` - Mail poslan korisnicima u `b1` (REP + `co1@test.com`) - Mail **nije** poslan korisniku u `b2` (`co2@test.com`)
- **Dobiveni rezultat**: **PASS**

IS-4: Arhiviranje sastanka – greška ako postoji pravna točka bez zaključka (izazivanje pogreške)

- **Funkcionalnost**: Arhiviranje sastanka je dozvoljeno samo ako sve pravne točke imaju zaključak.
- **Ulazi**: - REP korisnik (Basic Auth) - Sastanak status `Obavljen` - Točka dnevnog reda: `legal = 1` i `conclusion = null` - API poziv: `POST /api/meetings/{id}/archive`
- **Koraci**: - Kreira se sastanak (`Obavljen`) i doda se pravna točka bez zaključka. - Pokušaj arhiviranja → očekuje se greška (4xx). - Dodaje se zaključak preko `POST /api/meetings/{mId}/items/1/conclusion`. - Ponovno arhiviranje → očekuje se uspjeh i status `Archived`.
- **Očekivani rezultat**: - Prvi `/archive` vraća 4xx (kontrolirano odbijanje) - Nakon zaključka: `/archive` vraća 2xx, `status == Archived`
- **Dobiveni rezultat**: **PASS**

Prezentacija rezultata

```
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------< com.pcelice:backend >-------------------
-----
[INFO] Building backend 0.0.1-SNAPSHOT
[INFO]   from pom.xml
[INFO] --------------------------------[ jar ]---------------------------
-----
[INFO]
[INFO] --- clean:3.4.1:clean (default-clean) @ backend ---
[INFO] Deleting /home/lukas/Documents/FER/PROGI/stamb-
plan/StanPlan/backend/target
[INFO]
[INFO] --- resources:3.3.1:resources (default-resources) @ backend ---
[INFO] Copying 1 resource from src/main/resources to target/classes
[INFO] Copying 2 resources from src/main/resources to target/classes
[INFO]
[INFO] --- compiler:3.14.1:compile (default-compile) @ backend ---
[INFO] Recompiling the module because of changed source code.
[INFO] Compiling 38 source files with javac [debug parameters release 21]
to target/classes
[INFO]
[INFO] --- resources:3.3.1:testResources (default-testResources) @ backend
---
[INFO] skip non existing resourceDirectory
/home/lukas/Documents/FER/PROGI/stamb-
```

```
plan/StanPlan/backend/src/test/resources
[INFO]
[INFO] --- compiler:3.14.1:testCompile (default-testCompile) @ backend ---
[INFO] Recompiling the module because of changed dependency.
[INFO] Compiling 3 source files with javac [debug parameters release 21]
to target/test-classes
[INFO]
[INFO] --- surefire:3.5.4:test (default-test) @ backend ---
[INFO] Using auto detected provider
org.apache.maven.surefire.junitplatform.JUnitPlatformProvider
[INFO]
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO] Running com.pcelice.backend.DiscussionIntegrationTest
21:46:50.035 [main] INFO
org.springframework.test.context.support.AnnotationConfigContextLoaderUtil
s -- Could not detect default configuration classes for test class
[com.pcelice.backend.DiscussionIntegrationTest]: DiscussionIntegrationTest
does not declare any static, non-private, non-final, nested classes
annotated with @Configuration.
21:46:50.135 [main] INFO
org.springframework.boot.test.context.SpringBootTestContextBootstrapper --
Found @SpringBootConfiguration com.pcelice.backend.BackendApplication for
test class com.pcelice.backend.DiscussionIntegrationTest


  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/

 :: Spring Boot ::                (v3.5.7)

2026-01-23T21:46:50.465+01:00  INFO 53807 --- [backend] [           main]
c.p.backend.DiscussionIntegrationTest    : Starting
DiscussionIntegrationTest using Java 25.0.1 with PID 53807 (started by
lukas in /home/lukas/Documents/FER/PROGI/stamb-plan/StanPlan/backend)
2026-01-23T21:46:50.466+01:00  INFO 53807 --- [backend] [           main]
c.p.backend.DiscussionIntegrationTest    : The following 1 profile is
active: "security"
2026-01-23T21:46:51.033+01:00  INFO 53807 --- [backend] [           main]
.s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA
repositories in DEFAULT mode.
2026-01-23T21:46:51.149+01:00  INFO 53807 --- [backend] [           main]
.s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository
scanning in 106 ms. Found 3 JPA repository interfaces.
2026-01-23T21:46:51.695+01:00  INFO 53807 --- [backend] [           main]
o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port 0
(http)
2026-01-23T21:46:51.707+01:00  INFO 53807 --- [backend] [           main]
o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2026-01-23T21:46:51.707+01:00  INFO 53807 --- [backend] [           main]
```

```
o.apache.catalina.core.StandardEngine    : Starting Servlet engine:
[Apache Tomcat/10.1.48]
2026-01-23T21:46:51.748+01:00  INFO 53807 --- [backend] [           main]
o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded
WebApplicationContext
2026-01-23T21:46:51.749+01:00  INFO 53807 --- [backend] [           main]
w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext:
initialization completed in 1268 ms
2026-01-23T21:46:51.850+01:00  INFO 53807 --- [backend] [           main]
com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Starting...
2026-01-23T21:46:52.002+01:00  INFO 53807 --- [backend] [           main]
com.zaxxer.hikari.pool.HikariPool        : HikariPool-1 - Added connection
conn0: url=jdbc:h2:mem:testdb user=SA
2026-01-23T21:46:52.004+01:00  INFO 53807 --- [backend] [           main]
com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Start completed.
2026-01-23T21:46:52.112+01:00  INFO 53807 --- [backend] [           main]
o.hibernate.jpa.internal.util.LogHelper  : HHH000204: Processing
PersistenceUnitInfo [name: default]
2026-01-23T21:46:52.146+01:00  INFO 53807 --- [backend] [           main]
org.hibernate.Version                    : HHH000412: Hibernate ORM core
version 6.6.33.Final
2026-01-23T21:46:52.172+01:00  INFO 53807 --- [backend] [           main]
o.h.c.internal.RegionFactoryInitiator    : HHH000026: Second-level cache
disabled
2026-01-23T21:46:52.364+01:00  INFO 53807 --- [backend] [           main]
o.s.o.j.p.SpringPersistenceUnitInfo      : No LoadTimeWeaver setup:
ignoring JPA class transformer
2026-01-23T21:46:52.419+01:00  INFO 53807 --- [backend] [           main]
org.hibernate.orm.connections.pooling    : HHH10001005: Database info:
        Database JDBC URL [Connecting through datasource 'HikariDataSource
(HikariPool-1)']
        Database driver: undefined/unknown
        Database version: 2.3.232
        Autocommit mode: undefined/unknown
        Isolation level: undefined/unknown
        Minimum pool size: undefined/unknown
        Maximum pool size: undefined/unknown
2026-01-23T21:46:53.120+01:00  INFO 53807 --- [backend] [           main]
o.h.e.t.j.p.i.JtaPlatformInitiator       : HHH000489: No JTA platform
available (set 'hibernate.transaction.jta.platform' to enable JTA platform
integration)
2026-01-23T21:46:53.152+01:00  INFO 53807 --- [backend] [           main]
j.LocalContainerEntityManagerFactoryBean : Initialized JPA
EntityManagerFactory for persistence unit 'default'
2026-01-23T21:46:53.449+01:00  INFO 53807 --- [backend] [           main]
r$InitializeUserDetailsManagerConfigurer : Global AuthenticationManager
configured with UserDetailsService bean with name
coOwnerUserDetailsService
WARNING: A terminally deprecated method in sun.misc.Unsafe has been called
WARNING: sun.misc.Unsafe::allocateMemory has been called by
io.netty.util.internal.PlatformDependent0$2
(file:/home/lukas/.m2/repository/io/netty/netty-
common/4.1.128.Final/netty-common-4.1.128.Final.jar)
WARNING: Please consider reporting this to the maintainers of class
```

```
io.netty.util.internal.PlatformDependent0$2
WARNING: sun.misc.Unsafe::allocateMemory will be removed in a future
release
2026-01-23T21:46:53.682+01:00  WARN 53807 --- [backend] [           main]
JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is
enabled by default. Therefore, database queries may be performed during
view rendering. Explicitly configure spring.jpa.open-in-view to disable
this warning
2026-01-23T21:46:53.703+01:00  INFO 53807 --- [backend] [           main]
o.s.b.a.w.s.WelcomePageHandlerMapping    : Adding welcome page: class path
resource [static/index.html]
2026-01-23T21:46:54.520+01:00  INFO 53807 --- [backend] [           main]
o.s.b.a.h2.H2ConsoleAutoConfiguration    : H2 console available at '/h2-
console'. Database available at 'jdbc:h2:mem:testdb'
2026-01-23T21:46:54.593+01:00  INFO 53807 --- [backend] [           main]
o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port 35347
(http) with context path '/'
2026-01-23T21:46:54.601+01:00  INFO 53807 --- [backend] [           main]
c.p.backend.DiscussionIntegrationTest    : Started
DiscussionIntegrationTest in 4.368 seconds (process running for 5.095)
Mockito is currently self-attaching to enable the inline-mock-maker. This
will no longer work in future releases of the JDK. Please add Mockito as
an agent to your build as described in Mockito's documentation:
https://javadoc.io/doc/org.mockito/mockito-
core/latest/org.mockito/org/mockito/Mockito.html#0.3
OpenJDK 64-Bit Server VM warning: Sharing is only supported for boot
loader classes because bootstrap classpath has been appended
WARNING: A Java agent has been loaded dynamically
(/home/lukas/.m2/repository/net/bytebuddy/byte-buddy-agent/1.17.8/byte-
buddy-agent-1.17.8.jar)
WARNING: If a serviceability tool is in use, please run with -
XX:+EnableDynamicAgentLoading to hide this warning
WARNING: If a serviceability tool is not in use, please run with -
Djdk.instrument.traceUsage for more information
WARNING: Dynamic loading of agents will be disallowed by default in a
future release
WARNING: A restricted method in java.lang.System has been called
WARNING: java.lang.System::loadLibrary has been called by
io.netty.util.internal.NativeLibraryUtil in an unnamed module
(file:/home/lukas/.m2/repository/io/netty/netty-
common/4.1.128.Final/netty-common-4.1.128.Final.jar)
WARNING: Use --enable-native-access=ALL-UNNAMED to avoid a warning for
callers in this module
WARNING: Restricted methods will be blocked in a future release unless
native access is enabled

2026-01-23T21:46:55.322+01:00  INFO 53807 --- [backend] [o-auto-1-exec-1]
o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring
DispatcherServlet 'dispatcherServlet'
2026-01-23T21:46:55.322+01:00  INFO 53807 --- [backend] [o-auto-1-exec-1]
o.s.web.servlet.DispatcherServlet        : Initializing Servlet
'dispatcherServlet'
2026-01-23T21:46:55.323+01:00  INFO 53807 --- [backend] [o-auto-1-exec-1]
o.s.web.servlet.DispatcherServlet        : Completed initialization in 1
```

```
ms
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed:
7.665 s -- in com.pcelice.backend.DiscussionIntegrationTest
[INFO] Running com.pcelice.backend.SeleniumTest
2026-01-23T21:46:58.283+01:00  WARN 53807 --- [backend] [           main]
o.o.selenium.devtools.CdpVersionFinder   : Unable to find an exact match
for CDP version 144, returning the closest version; found: 143; Please
update to a Selenium version that supports CDP version 144
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed:
0.969 s -- in com.pcelice.backend.SeleniumTest
[INFO] Running com.pcelice.backend.UserTest
2026-01-23T21:46:59.094+01:00  WARN 53807 --- [backend] [           main]
o.o.selenium.devtools.CdpVersionFinder   : Unable to find an exact match
for CDP version 144, returning the closest version; found: 143; Please
update to a Selenium version that supports CDP version 144
2026-01-23T21:47:02.057+01:00  WARN 53807 --- [backend] [           main]
o.o.selenium.devtools.CdpVersionFinder   : Unable to find an exact match
for CDP version 144, returning the closest version; found: 143; Please
update to a Selenium version that supports CDP version 144
2026-01-23T21:47:03.164+01:00  WARN 53807 --- [backend] [           main]
o.o.selenium.devtools.CdpVersionFinder   : Unable to find an exact match
for CDP version 144, returning the closest version; found: 143; Please
update to a Selenium version that supports CDP version 144
2026-01-23T21:47:06.029+01:00  WARN 53807 --- [backend] [           main]
o.o.selenium.devtools.CdpVersionFinder   : Unable to find an exact match
for CDP version 144, returning the closest version; found: 143; Please
update to a Selenium version that supports CDP version 144
[INFO] Tests run: 4, Failures: 0, Errors: 0, Skipped: 0, Time elapsed:
8.064 s -- in com.pcelice.backend.UserTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 6, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------
-----
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------
-----
[INFO] Total time:  19.936 s
[INFO] Finished at: 2026-01-23T21:47:07+01:00
[INFO] ------------------------------------------------------------------
-----
```

```
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------< com.pcelice:backend >-------------------
-----
[INFO] Building backend 0.0.1-SNAPSHOT
[INFO]   from pom.xml
[INFO] --------------------------------[ jar ]---------------------------
```

```
-----
Downloading from central:
https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-
surefire-plugin/3.5.4/maven-surefire-plugin-3.5.4.pom
Downloaded from central:
https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-
surefire-plugin/3.5.4/maven-surefire-plugin-3.5.4.pom (4.9 kB at 6.8 kB/s)
Downloading from central:
https://repo.maven.apache.org/maven2/org/apache/maven/surefire/surefire/3.
5.4/surefire-3.5.4.pom
Downloaded from central:
https://repo.maven.apache.org/maven2/org/apache/maven/surefire/surefire/3.
5.4/surefire-3.5.4.pom (19 kB at 282 kB/s)
Downloading from central:
https://repo.maven.apache.org/maven2/org/apache/maven/maven-
parent/44/maven-parent-44.pom
Downloaded from central:
https://repo.maven.apache.org/maven2/org/apache/maven/maven-
parent/44/maven-parent-44.pom (52 kB at 645 kB/s)
Downloading from central:
https://repo.maven.apache.org/maven2/org/apache/apache/34/apache-34.pom
Downloaded from central:
https://repo.maven.apache.org/maven2/org/apache/apache/34/apache-34.pom
(24 kB at 418 kB/s)
Downloading from central:
https://repo.maven.apache.org/maven2/org/junit/junit-bom/5.12.1/junit-bom-
5.12.1.pom
Downloaded from central:
https://repo.maven.apache.org/maven2/org/junit/junit-bom/5.12.1/junit-bom-
5.12.1.pom (5.6 kB at 107 kB/s)
Downloading from central:
https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-
surefire-plugin/3.5.4/maven-surefire-plugin-3.5.4.jar
Downloaded from central:
https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-
surefire-plugin/3.5.4/maven-surefire-plugin-3.5.4.jar (46 kB at 720 kB/s)
[INFO]
[INFO] --- resources:3.3.1:resources (default-resources) @ backend ---
[INFO] Copying 2 resources from src/main/resources to target/classes
[INFO] Copying 1 resource from src/main/resources to target/classes
[INFO]
[INFO] --- compiler:3.14.1:compile (default-compile) @ backend ---
[INFO] Nothing to compile - all classes are up to date.
[INFO]
[INFO] --- resources:3.3.1:testResources (default-testResources) @ backend
---
[INFO] skip non existing resourceDirectory
/Users/amalijanovalic/projects/StanPlan/backend/src/test/resources
[INFO]
[INFO] --- compiler:3.14.1:testCompile (default-testCompile) @ backend ---
[INFO] Recompiling the module because of changed source code.
[INFO] Compiling 4 source files with javac [debug parameters release 21]
to target/test-classes
[WARNING]
```

```
/Users/amalijanovalic/projects/StanPlan/backend/src/test/java/com/pcelice/
backend/MeetingIntegrationTest.java:[53,6]
org.springframework.boot.test.mock.mockito.SpyBean in
org.springframework.boot.test.mock.mockito has been deprecated and marked
for removal
[INFO]
[INFO] --- surefire:3.5.4:test (default-test) @ backend ---
Downloading from central:
https://repo.maven.apache.org/maven2/org/apache/maven/surefire/surefire-
api/3.5.4/surefire-api-3.5.4.pom
Downloaded from central:
https://repo.maven.apache.org/maven2/org/apache/maven/surefire/surefire-
api/3.5.4/surefire-api-3.5.4.pom (3.7 kB at 49 kB/s)
Downloading from central:
https://repo.maven.apache.org/maven2/org/apache/maven/surefire/surefire-
logger-api/3.5.4/surefire-logger-api-3.5.4.pom
Downloaded from central:
https://repo.maven.apache.org/maven2/org/apache/maven/surefire/surefire-
logger-api/3.5.4/surefire-logger-api-3.5.4.pom (3.5 kB at 62 kB/s)
Downloading from central:
https://repo.maven.apache.org/maven2/org/apache/maven/surefire/surefire-
shared-utils/3.5.4/surefire-shared-utils-3.5.4.pom
Downloaded from central:
https://repo.maven.apache.org/maven2/org/apache/maven/surefire/surefire-
shared-utils/3.5.4/surefire-shared-utils-3.5.4.pom (4.0 kB at 75 kB/s)
Downloading from central:
https://repo.maven.apache.org/maven2/org/apache/maven/surefire/surefire-
extensions-api/3.5.4/surefire-extensions-api-3.5.4.pom
Downloaded from central:
https://repo.maven.apache.org/maven2/org/apache/maven/surefire/surefire-
extensions-api/3.5.4/surefire-extensions-api-3.5.4.pom (3.6 kB at 74 kB/s)
Downloading from central:
https://repo.maven.apache.org/maven2/org/apache/maven/surefire/maven-
surefire-common/3.5.4/maven-surefire-common-3.5.4.pom
Downloaded from central:
https://repo.maven.apache.org/maven2/org/apache/maven/surefire/maven-
surefire-common/3.5.4/maven-surefire-common-3.5.4.pom (7.3 kB at 140 kB/s)
Downloading from central:
https://repo.maven.apache.org/maven2/org/apache/maven/surefire/surefire-
booter/3.5.4/surefire-booter-3.5.4.pom
Downloaded from central:
https://repo.maven.apache.org/maven2/org/apache/maven/surefire/surefire-
booter/3.5.4/surefire-booter-3.5.4.pom (5.0 kB at 84 kB/s)
Downloading from central:
https://repo.maven.apache.org/maven2/org/apache/maven/surefire/surefire-
extensions-spi/3.5.4/surefire-extensions-spi-3.5.4.pom
Downloaded from central:
https://repo.maven.apache.org/maven2/org/apache/maven/surefire/surefire-
extensions-spi/3.5.4/surefire-extensions-spi-3.5.4.pom (1.7 kB at 34 kB/s)
Downloading from central:
https://repo.maven.apache.org/maven2/org/apache/maven/surefire/surefire-
api/3.5.4/surefire-api-3.5.4.jar
Downloaded from central:
https://repo.maven.apache.org/maven2/org/apache/maven/surefire/surefire-
```

```
api/3.5.4/surefire-api-3.5.4.jar (174 kB at 2.2 MB/s)
Downloading from central:
https://repo.maven.apache.org/maven2/org/apache/maven/surefire/surefire-
logger-api/3.5.4/surefire-logger-api-3.5.4.jar
Downloading from central:
https://repo.maven.apache.org/maven2/org/apache/maven/surefire/surefire-
shared-utils/3.5.4/surefire-shared-utils-3.5.4.jar
Downloading from central:
https://repo.maven.apache.org/maven2/org/apache/maven/surefire/surefire-
extensions-api/3.5.4/surefire-extensions-api-3.5.4.jar
Downloading from central:
https://repo.maven.apache.org/maven2/org/apache/maven/surefire/surefire-
booter/3.5.4/surefire-booter-3.5.4.jar
Downloading from central:
https://repo.maven.apache.org/maven2/org/apache/maven/surefire/maven-
surefire-common/3.5.4/maven-surefire-common-3.5.4.jar
Downloaded from central:
https://repo.maven.apache.org/maven2/org/apache/maven/surefire/maven-
surefire-common/3.5.4/maven-surefire-common-3.5.4.jar (314 kB at 3.3 MB/s)
Downloading from central:
https://repo.maven.apache.org/maven2/org/apache/maven/surefire/surefire-
extensions-spi/3.5.4/surefire-extensions-spi-3.5.4.jar
Downloaded from central:
https://repo.maven.apache.org/maven2/org/apache/maven/surefire/surefire-
extensions-spi/3.5.4/surefire-extensions-spi-3.5.4.jar (8.2 kB at 55 kB/s)
Downloaded from central:
https://repo.maven.apache.org/maven2/org/apache/maven/surefire/surefire-
extensions-api/3.5.4/surefire-extensions-api-3.5.4.jar (26 kB at 99 kB/s)
Downloaded from central:
https://repo.maven.apache.org/maven2/org/apache/maven/surefire/surefire-
logger-api/3.5.4/surefire-logger-api-3.5.4.jar (14 kB at 48 kB/s)
Downloaded from central:
https://repo.maven.apache.org/maven2/org/apache/maven/surefire/surefire-
booter/3.5.4/surefire-booter-3.5.4.jar (119 kB at 406 kB/s)
Downloaded from central:
https://repo.maven.apache.org/maven2/org/apache/maven/surefire/surefire-
shared-utils/3.5.4/surefire-shared-utils-3.5.4.jar (2.9 MB at 7.8 MB/s)
[INFO] Using auto detected provider
org.apache.maven.surefire.junitplatform.JUnitPlatformProvider
Downloading from central:
https://repo.maven.apache.org/maven2/org/apache/maven/surefire/surefire-
junit-platform/3.5.4/surefire-junit-platform-3.5.4.pom
Downloaded from central:
https://repo.maven.apache.org/maven2/org/apache/maven/surefire/surefire-
junit-platform/3.5.4/surefire-junit-platform-3.5.4.pom (5.2 kB at 75 kB/s)
Downloading from central:
https://repo.maven.apache.org/maven2/org/apache/maven/surefire/surefire-
providers/3.5.4/surefire-providers-3.5.4.pom
Downloaded from central:
https://repo.maven.apache.org/maven2/org/apache/maven/surefire/surefire-
providers/3.5.4/surefire-providers-3.5.4.pom (2.5 kB at 46 kB/s)
Downloading from central:
https://repo.maven.apache.org/maven2/org/apache/maven/surefire/common-
java5/3.5.4/common-java5-3.5.4.pom
```

```
Downloaded from central:
https://repo.maven.apache.org/maven2/org/apache/maven/surefire/common-
java5/3.5.4/common-java5-3.5.4.pom (3.1 kB at 58 kB/s)
Downloading from central:
https://repo.maven.apache.org/maven2/org/junit/platform/junit-platform-
engine/1.12.1/junit-platform-engine-1.12.1.pom
Downloaded from central:
https://repo.maven.apache.org/maven2/org/junit/platform/junit-platform-
engine/1.12.1/junit-platform-engine-1.12.1.pom (3.2 kB at 59 kB/s)
Downloading from central:
https://repo.maven.apache.org/maven2/org/junit/platform/junit-platform-
commons/1.12.1/junit-platform-commons-1.12.1.pom
Downloaded from central:
https://repo.maven.apache.org/maven2/org/junit/platform/junit-platform-
commons/1.12.1/junit-platform-commons-1.12.1.pom (2.8 kB at 48 kB/s)
Downloading from central:
https://repo.maven.apache.org/maven2/org/junit/platform/junit-platform-
launcher/1.12.1/junit-platform-launcher-1.12.1.pom
Downloaded from central:
https://repo.maven.apache.org/maven2/org/junit/platform/junit-platform-
launcher/1.12.1/junit-platform-launcher-1.12.1.pom (3.0 kB at 44 kB/s)
Downloading from central:
https://repo.maven.apache.org/maven2/org/apache/maven/surefire/surefire-
junit-platform/3.5.4/surefire-junit-platform-3.5.4.jar
Downloaded from central:
https://repo.maven.apache.org/maven2/org/apache/maven/surefire/surefire-
junit-platform/3.5.4/surefire-junit-platform-3.5.4.jar (35 kB at 682 kB/s)
Downloading from central:
https://repo.maven.apache.org/maven2/org/apache/maven/surefire/common-
java5/3.5.4/common-java5-3.5.4.jar
Downloading from central:
https://repo.maven.apache.org/maven2/org/junit/platform/junit-platform-
engine/1.12.1/junit-platform-engine-1.12.1.jar
Downloading from central:
https://repo.maven.apache.org/maven2/org/junit/platform/junit-platform-
commons/1.12.1/junit-platform-commons-1.12.1.jar
Downloading from central:
https://repo.maven.apache.org/maven2/org/junit/platform/junit-platform-
launcher/1.12.1/junit-platform-launcher-1.12.1.jar
Downloaded from central:
https://repo.maven.apache.org/maven2/org/apache/maven/surefire/common-
java5/3.5.4/common-java5-3.5.4.jar (18 kB at 245 kB/s)
Downloaded from central:
https://repo.maven.apache.org/maven2/org/junit/platform/junit-platform-
engine/1.12.1/junit-platform-engine-1.12.1.jar (256 kB at 3.1 MB/s)
Downloaded from central:
https://repo.maven.apache.org/maven2/org/junit/platform/junit-platform-
commons/1.12.1/junit-platform-commons-1.12.1.jar (152 kB at 1.4 MB/s)
Downloaded from central:
https://repo.maven.apache.org/maven2/org/junit/platform/junit-platform-
launcher/1.12.1/junit-platform-launcher-1.12.1.jar (208 kB at 1.8 MB/s)
Downloading from central:
https://repo.maven.apache.org/maven2/org/junit/platform/junit-platform-
launcher/1.12.2/junit-platform-launcher-1.12.2.pom
```

```
Downloaded from central:
https://repo.maven.apache.org/maven2/org/junit/platform/junit-platform-
launcher/1.12.2/junit-platform-launcher-1.12.2.pom (3.0 kB at 59 kB/s)
Downloading from central:
https://repo.maven.apache.org/maven2/org/junit/platform/junit-platform-
launcher/1.12.2/junit-platform-launcher-1.12.2.jar
Downloaded from central:
https://repo.maven.apache.org/maven2/org/junit/platform/junit-platform-
launcher/1.12.2/junit-platform-launcher-1.12.2.jar (208 kB at 3.3 MB/s)
[INFO]
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO] Running com.pcelice.backend.MeetingIntegrationTest
22:59:44.015 [main] INFO
org.springframework.test.context.support.AnnotationConfigContextLoaderUtil
s -- Could not detect default configuration classes for test class
[com.pcelice.backend.MeetingIntegrationTest]: MeetingIntegrationTest does
not declare any static, non-private, non-final, nested classes annotated
with @Configuration.
22:59:44.203 [main] INFO
org.springframework.boot.test.context.SpringBootTestContextBootstrapper --
Found @SpringBootConfiguration com.pcelice.backend.BackendApplication for
test class com.pcelice.backend.MeetingIntegrationTest


  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/

 :: Spring Boot ::                (v3.5.7)

2026-01-23T22:59:44.963+01:00  INFO 25320 --- [backend] [           main]
c.p.backend.MeetingIntegrationTest       : Starting MeetingIntegrationTest
using Java 22 with PID 25320 (started by amalijanovalic in
/Users/amalijanovalic/projects/StanPlan/backend)
2026-01-23T22:59:44.964+01:00  INFO 25320 --- [backend] [           main]
c.p.backend.MeetingIntegrationTest       : The following 1 profile is
active: "security"
2026-01-23T22:59:46.474+01:00  INFO 25320 --- [backend] [           main]
.s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA
repositories in DEFAULT mode.
2026-01-23T22:59:46.660+01:00  INFO 25320 --- [backend] [           main]
.s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository
scanning in 174 ms. Found 3 JPA repository interfaces.
2026-01-23T22:59:47.960+01:00  INFO 25320 --- [backend] [           main]
o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port 0
(http)
2026-01-23T22:59:47.979+01:00  INFO 25320 --- [backend] [           main]
o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2026-01-23T22:59:47.979+01:00  INFO 25320 --- [backend] [           main]
o.apache.catalina.core.StandardEngine    : Starting Servlet engine:
```

```
[Apache Tomcat/10.1.48]
2026-01-23T22:59:48.062+01:00  INFO 25320 --- [backend] [           main]
o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded
WebApplicationContext
2026-01-23T22:59:48.064+01:00  INFO 25320 --- [backend] [           main]
w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext:
initialization completed in 3072 ms
2026-01-23T22:59:48.266+01:00  INFO 25320 --- [backend] [           main]
com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Starting...
2026-01-23T22:59:48.405+01:00  INFO 25320 --- [backend] [           main]
com.zaxxer.hikari.pool.HikariPool        : HikariPool-1 - Added connection
conn0: url=jdbc:h2:mem:testdb user=SA
2026-01-23T22:59:48.408+01:00  INFO 25320 --- [backend] [           main]
com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Start completed.
2026-01-23T22:59:48.588+01:00  INFO 25320 --- [backend] [           main]
o.hibernate.jpa.internal.util.LogHelper  : HHH000204: Processing
PersistenceUnitInfo [name: default]
2026-01-23T22:59:48.633+01:00  INFO 25320 --- [backend] [           main]
org.hibernate.Version                    : HHH000412: Hibernate ORM core
version 6.6.33.Final
2026-01-23T22:59:48.660+01:00  INFO 25320 --- [backend] [           main]
o.h.c.internal.RegionFactoryInitiator    : HHH000026: Second-level cache
disabled
2026-01-23T22:59:48.873+01:00  INFO 25320 --- [backend] [           main]
o.s.o.j.p.SpringPersistenceUnitInfo      : No LoadTimeWeaver setup:
ignoring JPA class transformer
2026-01-23T22:59:48.942+01:00  INFO 25320 --- [backend] [           main]
org.hibernate.orm.connections.pooling    : HHH10001005: Database info:
        Database JDBC URL [Connecting through datasource 'HikariDataSource
(HikariPool-1)']
        Database driver: undefined/unknown
        Database version: 2.3.232
        Autocommit mode: undefined/unknown
        Isolation level: undefined/unknown
        Minimum pool size: undefined/unknown
        Maximum pool size: undefined/unknown
2026-01-23T22:59:50.222+01:00  INFO 25320 --- [backend] [           main]
o.h.e.t.j.p.i.JtaPlatformInitiator       : HHH000489: No JTA platform
available (set 'hibernate.transaction.jta.platform' to enable JTA platform
integration)
2026-01-23T22:59:50.302+01:00  INFO 25320 --- [backend] [           main]
j.LocalContainerEntityManagerFactoryBean : Initialized JPA
EntityManagerFactory for persistence unit 'default'
2026-01-23T22:59:51.024+01:00  INFO 25320 --- [backend] [           main]
r$InitializeUserDetailsManagerConfigurer : Global AuthenticationManager
configured with UserDetailsService bean with name
coOwnerUserDetailsService
Mockito is currently self-attaching to enable the inline-mock-maker. This
will no longer work in future releases of the JDK. Please add Mockito as
an agent to your build as described in Mockito's documentation:
https://javadoc.io/doc/org.mockito/mockito-
core/latest/org.mockito/org/mockito/Mockito.html#0.3
WARNING: A Java agent has been loaded dynamically
(/Users/amalijanovalic/.m2/repository/net/bytebuddy/byte-buddy-
```

```
agent/1.17.8/byte-buddy-agent-1.17.8.jar)
WARNING: If a serviceability tool is in use, please run with -
XX:+EnableDynamicAgentLoading to hide this warning
WARNING: If a serviceability tool is not in use, please run with -
Djdk.instrument.traceUsage for more information
WARNING: Dynamic loading of agents will be disallowed by default in a
future release
Java HotSpot(TM) 64-Bit Server VM warning: Sharing is only supported for
boot loader classes because bootstrap classpath has been appended
2026-01-23T22:59:53.262+01:00  WARN 25320 --- [backend] [           main]
JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is
enabled by default. Therefore, database queries may be performed during
view rendering. Explicitly configure spring.jpa.open-in-view to disable
this warning
2026-01-23T22:59:53.342+01:00  INFO 25320 --- [backend] [           main]
o.s.b.a.w.s.WelcomePageHandlerMapping    : Adding welcome page: class path
resource [static/index.html]
2026-01-23T22:59:56.007+01:00  INFO 25320 --- [backend] [           main]
o.s.b.a.h2.H2ConsoleAutoConfiguration    : H2 console available at '/h2-
console'. Database available at 'jdbc:h2:mem:testdb'
2026-01-23T22:59:56.177+01:00  INFO 25320 --- [backend] [           main]
o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port 62557
(http) with context path '/'
2026-01-23T22:59:56.198+01:00  INFO 25320 --- [backend] [           main]
c.p.backend.MeetingIntegrationTest       : Started MeetingIntegrationTest
in 11.804 seconds (process running for 13.225)
2026-01-23T22:59:57.614+01:00  INFO 25320 --- [backend] [o-auto-1-exec-1]
o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring
DispatcherServlet 'dispatcherServlet'
2026-01-23T22:59:57.615+01:00  INFO 25320 --- [backend] [o-auto-1-exec-1]
o.s.web.servlet.DispatcherServlet        : Initializing Servlet
'dispatcherServlet'
2026-01-23T22:59:57.618+01:00  INFO 25320 --- [backend] [o-auto-1-exec-1]
o.s.web.servlet.DispatcherServlet        : Completed initialization in 2
ms
Sending archive emails to 1 co-owners in building 2
MOCK EMAIL TO: rep4@test.com
Subject: Novi sastanak objavljen: M
Meeting time: null
---
Sending emails to 2 co-owners in building 4
MOCK EMAIL TO: rep3@test.com
Subject: Novi sastanak objavljen: M
Meeting time: null
---
MOCK EMAIL TO: co1@test.com
Subject: Novi sastanak objavljen: M
Meeting time: null
---
[INFO] Tests run: 4, Failures: 0, Errors: 0, Skipped: 0, Time elapsed:
15.75 s -- in com.pcelice.backend.MeetingIntegrationTest
[INFO]
[INFO] Results:
[INFO]
```

```
[INFO] Tests run: 4, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
------
[INFO] Total time:  26.577 s
[INFO] Finished at: 2026-01-23T23:00:00+01:00
[INFO] ------------------------------------------------------------------------
------
```

Korištene tehnologije i alati

Ova stranica opisuje sve tehnologije, alate i konfiguracije korištene u projektu StanPlan, uz obrazloženje izbora i specifične postavke.

---

1. Programski jezici

- **Java 21** (backend, Spring Boot)
- **JavaScript (ES2020+)** (frontend, React)
- **SQL** (PostgreSQL, H2 za testiranje)

2. Radni okviri i biblioteke

- **Backend:**
  - Spring Boot 3.5.7 (REST API, sigurnost, OAuth2, JPA)
  - Spring Data JPA (povezivanje s bazom)
  - Spring Boot Starter Web, Validation, Mail
  - OAuth2 Client (Google, GitHub prijava)
  - PostgreSQL JDBC 42.7.7 (proizvodna baza)
  - H2 Database (testna baza)
  - WebJars (Bootstrap 4.3.1, jQuery 3.4.1, js-cookie 2.1.0)
  - Selenium 4.39.0 (UI testovi)
  - JUnit Jupiter (jedinični testovi)
- **Frontend:**
  - React 19.1.1 (SPA, komponentni pristup)
  - React DOM 19.1.1
  - React Router DOM 7.9.5 (navigacija)
  - Vite 7.1.7 (build alat)
  - ESLint 9.36.0 (linting, pravila)
  - Styled-components nisu korištene, stilizacija je ručna (CSS)

3. Baza podataka

- **PostgreSQL 13** (proizvodna baza, relacijski model)

    - Tablice: CO_OWNER, CITY, BUILDING, MEETING, ITEM, PARTICIPATION
    - Indeksi i vanjski ključevi za integritet i performanse
- **H2** (testna baza, automatsko pokretanje za testove)

4. Razvojni alati

- **IDE:**
    - VS Code (glavni editor)
    - IntelliJ IDEA (backend razvoj)
    - Figma (dizajn sučelja)
- **Verzioniranje:**
    - Git 2.34+
    - GitHub (repozitorij, wiki, CI/CD)

5. Alati za ispitivanje

- **Backend:**
    - JUnit Jupiter (jedinični testovi)
    - Selenium 4.39.0 (UI testovi, integracija s ChromeDriverom)
- **Frontend:**
    - Mock podaci (JSON, localStorage)
    - Ručno testiranje UI-a

6. Alati za razmještaj

- **Docker 20.10+**
    - Backend: Maven build, Eclipse Temurin JDK 21, Spring Boot jar
    - Frontend: Node 18, Vite build, Nginx za produkcijsko posluživanje
    - Konfiguracija: API_BASE_URL kao environment varijabla, portovi 8080 (backend) i 80 (frontend)

7. Cloud platforma

- Trenutno nije hostano na cloud platformi. Lokalno pokretanje putem Docker-a.

Obrazloženje izbora tehnologija:

- **Spring Boot** je odabran zbog jednostavne integracije s bazom, sigurnosti i brzog razvoja REST API-ja.
- **React** omogućuje brzu izradu interaktivnog sučelja, komponentni pristup i jednostavnu integraciju s backendom.
- **PostgreSQL** je pouzdana relacijska baza, podržava napredne SQL funkcionalnosti i lako se integrira sa Spring Bootom.
- **Docker** olakšava razmještaj i testiranje aplikacije na različitim sustavima bez ručne instalacije ovisnosti.
- **OAuth2** (Google, GitHub) pruža sigurnu autentifikaciju bez potrebe za vlastitim upravljanjem lozinkama.
- **JUnit i Selenium** omogućuju automatsko testiranje funkcionalnosti i korisničkog sučelja.

Specifične konfiguracije:

- **Backend:**
  - `application.yml`: konfiguracija OAuth2 klijenata, H2 konzole, datasource URL, session cookie sigurnost, mail postavke
  - Dockerfile: Maven build, Temurin JDK 21, jar pokretanje
- **Frontend:**
  - Vite konfiguracija, API_BASE_URL kao env varijabla
  - ESLint pravila: no-unused-vars, preporučene React hooks i refresh konfiguracije
  - Dockerfile: build s Node 18, posluživanje s Nginx
- **Baza:**
  - SQL skripte definiraju tablice, indekse, vanjske ključeve i migracije

Zaključak:

Projekt koristi moderne, dobro podržane tehnologije s jasnim razlozima izbora. Konfiguracije su dokumentirane i prilagođene za timski rad, testiranje i jednostavno proširenje.

1. Instalacija

Preduvjeti

- Node.js 18
- Java JDK 21
- Maven
- Git

Preuzimanje

```
git clone https://github.com/Pcelicee/StanPlan.git
cd StanPlan
```

Frontend:

```
cd frontend
npm install
```

Backend:

```
cd backend
./mvnw clean install -DskipTests
```

2. Postavke

Frontend varijable

- Stvorite `.env` u `frontend/`: VITE_API_BASE_URL=http://localhost:8080

Backend varijable

- `DATABASE_URL` - URL baze
- `GOOGLE_CLIENT_ID` - Google OAuth
- `GOOGLE_CLIENT_SECRET` - Google secret
- `FRONTEND_URL` - http://localhost:5173

3. Pokretanje aplikacije

Lokalno

Backend:

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

Dostupan na http://localhost:8080

Frontend:

```
cd frontend
npm run dev
```

Dostupan na http://localhost:5173

Build za produkciju

Frontend:

```
npm run build
```

Backend:

```
./mvnw clean package -DskipTests
java -jar target/backend-0.0.1-SNAPSHOT.jar
```

4. Upute za administratore

Pristup Admin Panelu

- URL: http://localhost:5173/admin
- Uvjet: Korisnik mora imati ulogu ADMIN u bazi podataka.

Redovito održavanje

- **Arhiviranje baze podataka**: Zbog ograničenja Render Free plana, bazu je potrebno kopirati (dump) svakih 30 dana pomoću `pg_dump` naredbe i uvesti u novu instancu.
- **Pregled logova**: Dijagnostika se vrši unutar Render Dashboarda pod tabom **"Logs"** za Backend i Frontend servise.
- **Ažuriranje aplikacije**:
    1. Lokalno: Povući promjene naredbom `git pull origin main` i restartati servise.
    2. Produkcija: Render automatski pokreće novi build nakon svakog `git push` zahvata na `main` granu.

Rješavanje problema

- U slučaju grešaka, provjeriti status baze i konzolu u pregledniku (F12)

5. Render deploy

Backend

1. New Web Service na Render.com
2. GitHub repozitorij
3. Root Directory: `backend`, Runtime: Docker
4. Varijable: DATABASE_URL, GOOGLE_CLIENT_ID, GOOGLE_CLIENT_SECRET, FRONTEND_URL

Frontend

1. New Static Site
2. Root Directory: `frontend`
3. Build Command: `npm install && npm run build`
4. Publish: `dist`

Baza

1. New PostgreSQL
2. Database: `stanplan`
3. Kopirajte Internal Database URL

Zaključak i budući rad

Tijekom izrade projekta StanPlan, tim je detaljno proučio zahtjeve zadatka i implementirao većinu ključnih funkcionalnosti za podršku radu predstavnika suvlasnika u stambenim zgradama. Projekt je razvijan iterativno, s naglaskom na suradnju između backend i frontend tima, integraciju s bazom podataka i povezivanje s vanjskim servisima.

Tehnički izazovi

Najveći izazovi bili su:

- **Modeliranje procesa sastanaka**: Implementacija stanja sastanka (Planiran, Objavljen, Obavljen, Arhiviran) i pravila za prelazak između stanja, uključujući validaciju broja sudionika i zaključaka.
- **Glasanje i pravni učinak zaključaka**: Razlikovanje zaključaka s pravnim učinkom i bez njega, te povezivanje s procesom glasanja i arhiviranja.
- **Integracija sa StanBlog aplikacijom**: Dohvat i prikaz diskusija iz vanjske aplikacije, povezivanje točaka dnevnog reda s diskusijama.
- **Notifikacije e-mailom**: Priprema sustava za slanje obavijesti korisnicima u ključnim trenucima (objava i arhiviranje sastanka), ali bez potpune implementacije.
- **Administracija korisnika**: Omogućavanje administratoru kreiranje i upravljanje korisnicima, uz osnovnu validaciju i upravljanje ulogama.
- **Autentikacija i sigurnost**: Korištenje OAuth2 i lokalnih računa, postavljanje sigurnosnih pravila i CORS-a.

Neki izazovi su djelomično riješeni ili su ostali za budući rad, posebno automatizacija procesa, naprednije upravljanje korisničkim pravima i potpuna integracija s vanjskim servisima.

Stečena znanja

Tijekom projekta stečena su znanja iz:

- Razvoja REST API-ja u Spring Bootu
- Upravljanja relacijskim bazama podataka i migracijama
- Razvoja React aplikacija s Viteom
- Integracije s vanjskim servisima i API-jima
- Dockerizacije i pripreme za deployment
- Upravljanja korisničkim ulogama i sigurnošću
- Timskog rada i verzioniranja

Za brži i kvalitetniji razvoj dodatno bi koristila znanja iz:

- Automatiziranog testiranja i CI/CD procesa
- Naprednog upravljanja sigurnošću i enkripcijom
- Optimizacije performansi i skalabilnosti
- Napredne integracije s vanjskim servisima

Perspektive za nastavak rada

Projekt ima potencijal za proširenje kroz:

- Potpunu implementaciju e-mail notifikacija
- Automatizaciju arhiviranja i brisanja sastanaka
- Naprednije upravljanje korisničkim pravima i ulogama
- Detaljniju administraciju i pregled aktivnosti korisnika
- Unapređenje UX-a i mobilnu prilagodbu
- Dodatne integracije (npr. e-mail, mobilna aplikacija, naprednije glasanje)

Funkcionalnosti koje nisu (u potpunosti) implementirane

Na temelju zadatka i analize koda, funkcionalnosti koje nisu u potpunosti implementirane:

- **Slanje e-mail notifikacija** (pripremljeno, ali nije dovršeno za sve slučajeve)
- **Automatsko arhiviranje i brisanje starih sastanaka**
- **Povezivanje točaka dnevnog reda s diskusijama iz StanBlog-a** (osnovno dohvaćanje postoji, ali nije omogućeno povezivanje svake točke)
- **Glasanje i statusi zaključaka ('Izglasan', 'Odbijen')** (nije implementirano glasanje po točkama)
- **Detaljna administracija korisničkih uloga i prava** (trenutno samo osnovne uloge)
- **Napredna validacija podataka na frontendu**
- **Pregled i uređivanje profila korisnika**
- **Povijest aktivnosti i logiranje promjena**
- **Višekorisnička podrška za istovremeno uređivanje**
- **Jednostavniji sustav za povrat lozinke**
- **Naprednije filtriranje i pretraga sastanaka i diskusija**
- **Mobilna prilagodba korisničkog sučelja**

Ove funkcionalnosti predstavljaju smjer za budući rad i mogu značajno unaprijediti aplikaciju, kako bi u potpunosti zadovoljila sve zahtjeve iz izvornog zadatka.

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
7. Spring boot dokumentacija, https://docs.spring.io/spring-framework/reference/

Rev.	Opis promjene/dodatka	Autori	Datum
1.0	Dodan predložak iz repozitorija Programsko-inzenjerstvo. Uređeni Home, Footer. Započet opis projektnog zadatka (motivacija, postojeća rješenja) te evidentirane aktivnosti grupe.	Dora Vukelić	18.10.2025
1.1	Uređeni funkcionalni zahtjevi	Vito Gašparac	18.10.2025
1.2	Uređena analiza zahtjeva: prošireni funkcionalni i nefunkcionalni zahtjevi	Amalija Novalić	20.10.2025
1.3	Evidentiran sastanak grupe u dokumentaciji.	Dora Vukelić	20.10.2025
1.4	Napisana i proširena sekcija 'Opis projektnog zadatka' — detaljno razrađena motivacija i postojeća rješenja te analizirane slične aplikacije.	Bruno Pleše	20.10.2025
1.5	Dodatno uređena i proširena analiza zahtjeva (detaljniji opisi i ažurirane tablice zahtjeva).	Dora Vukelić	01.11.2025

Rev.	Opis promjene/dodatka	Autori	Datum
1.6	Uređena specifikacija zahtjeva sustava: uređena i dopunjena tablica provjere uključenosti funkcionalnosti u obrasce uporabe, ažurirani opisi obrazaca uporabe, dodan prvi dijagram slučajeva uporabe.	Dora Vukelić	04.11.2025
1.7	Dodani i ažurirani svi dijagrami obrazaca uporabe i prvi sekvencijski dijagrami (UC1, UC2, UC12).	Dora Vukelić	05.11.2025
1.8	Nastavak izrade i dorade sekvencijskih dijagrama te proširenje opisa procesa za složenije slučajeve uporabe.	Dora Vukelić	08.11.2025
1.9	Daljnja analiza zahtjeva, ispravci i proširivanje opisa potreba korisnika i zahtjeva sustava.	Dora Vukelić	09.11.2025
2.0	Dorada i proširenje opisa projektnog zadatka (posebno motivacije i usporedbe postojećih rješenja).	Diana Agejev	10.11.2025
2.1	Detaljno uređena sekcija “Arhitektura i dizajn sustava”: opis arhitekture (klijent-poslužitelj, monolit, Spring Boot), implementirani i planirani podsustavi, organizacija sustava na visokoj razini, opis aplikacijskih slojeva, baze podataka i MVC pristupa, dodani i opisani dijagrami baze podataka, ER dijagram, dijagram razreda i upute za izradu UML dijagrama stanja i aktivnosti.	Diana Agejev	11.11.2025
2.2	Daljnja nadopuna i ispravci u sekcijama arhitekture i dizajna: proširenje opisa dijagrama, ažurirani sekvencijski dijagrami u “Specifikacija zahtjeva sustava”.	Diana Agejev	12.11.2025
2.3	Dodani članovi tima na Home stranicu wiki dokumentacije.	Dora Vukelić	12.11.2025
2.4	Dodan i opisan ER dijagram baze podataka (veze korisnika, zgrade, grada, ključevi, atributi).	Dominik Topić	12.11.2025
2.5	Dodani nedostajući opisi svih dijagrama (dijagrami obrazaca uporabe, dijagrami razreda, baze podataka, visoke razine) te napisan, konsolidiran dnevnik promjena dokumentacije na temelju povijesti uređivanja i dogovora s kolegama.	Diana Agejev	13.12.2025
2.6	Proširena i detaljnije opisana sekcija "Arhitektura i dizajn sustava" s novim dijagramima i opisima slojeva.	Diana Agejev	04.01.2026
2.7	Dodana nova verzija ER dijagrama baze podataka i ažurirani opisi veza među entitetima.	Dominik Topić	18.01.2026
2.8	Dopunjena sekcija "Upute za puštanje u pogon" s detaljnim koracima za dockerizaciju i deployment.	Dominik Topić	22.01.2026
2.9	Ažurirani i prošireni opisi obrazaca uporabe te dodani novi sekvencijski dijagrami (UC1, UC2, UC12).	Dora Vukelić	19.01.2026

Rev.	Opis promjene/dodatka	Autori	Datum
3.0	Dodana tablica usporedbe implementiranih i neimplementiranih funkcionalnosti u "Specifikacija zahtjeva sustava".	Diana Agejev	19.01.2026
3.1	Proširena sekcija "Ispitivanje programskog rješenja" s opisom testiranja i prijedlozima za buduće testove.	Luka Sever	22.01.2026
3.2	Ažuriran popis literature i referenci korištenih tijekom izrade projekta.	Diana Agejev	22.01.2026
3.3	Dodana i proširena sekcija "Zaključak i budući rad" s detaljnom analizom izazova, stečenih znanja i neimplementiranih funkcionalnosti.	Diana Agejev	22.01.2025

- sve promjene su vidljive u povijesti promjena Wiki stranice

> Evidencija promjena sadrži popis promjena izvršenih tijekom cijelo 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.

TO DO

#Dnevnik sastajanja ***Kontinuirano osvježavanje***

1. sastanak (online)

- Datum: 17. listopada 2025.
- Prisustvovali: D.Topić, B.Pleše, L.Sever, D.Agejev, V.Gašparac, A.Novalić, D.Vukelić
- Teme sastanka:

> - opis prve teme
>
> > - LOŠE praćenje i kašnjenje tjednih aktivnosti na projektu
> > - BOLJE dogovoren sastanak uživo za tri dana radi boljeg planiranja projekta
>
> - opis druge teme
> - potrebno dodati asistenta i demosa na projekt te postaviti i urediti wiki

2. sastanak

- Datum: 20. listopada 2025.
- Prisustvovali: D.Topić, B.Pleše, L.Sever, D.Agejev, V.Gašparac, A.Novalić, D.Vukelić
- Teme sastanka:

> - opis prve teme
>
> > - raspodjela posla

> - opis druge teme
>
>> - tehnologija i razvoj aplikacije

3. sastanak

- Datum: 18. prosinca 2025.
- Prisustvovali: D.Topić, B.Pleše, L.Sever, D.Agejev, V.Gašparac, A.Novalić, D.Vukelić
- Teme sastanka:

> - opis prve teme
>
>> - poboljšanje raspodjele posla
>
> - opis druge teme
>
>> - dogovor oko konkretne implementacije
>
> - opis treće teme
>
>> - plan pisanja dokumentacije

Plan rada

Tjedan	Aktivnost	Sudionici (akronimi)
1.	Analiza zahtjeva i planiranje	LS, DT, BP, DA, VG, AN, DV
2.	Izrada arhitekture i baze	DT, DA, DV
3.	Postavljanje repozitorija i CI/CD	DV, LS
4.	Backend inicijalna implementacija	BP, VG, DT
5.	Frontend inicijalna implementacija	LS, AN, DV
6.	Spajanje backend i baze	BP, VG, DT
7.	Razvoj korisničkog sučelja	LS, AN, DV
8.	Implementacija autentikacije	BP, VG, LS
9.	Integracija s vanjskim servisima	BP, VG, DT
10.	Testiranje i QA	DA, LS
11.	Priprema dokumentacije	DA, DV
12.	Deploy i završna provjera	DT, DA

Legenda akronima: DT - Dominik Topić, BP - Bruno Pleše, LS - Luka Sever, DA - Diana Agejev, VG - Vito Gašparac, AN - Amalija Novalić, DV - Dora Vukelić

Tablica aktivnosti

Kontinuirano osvježavanje

Napomena: Doprinose u aktivnostima treba navesti u satima po članovima grupe po aktivnosti. Potrebno je navesti koliko je sati koja osoba uložila u pojedinu komponentu, možete oblikovati tablicu ili ispisati za svaku osobu.

Zadaci	Luka Sever	Dominik Topić	Bruno Pleše	Diana Agejev	Vito Gašparac	Amalija Novalić	Dora Vukelić
Upravljanje projektom	12			8			
Opis projektnog zadatka			2	1			
Funkcionalni zahtjevi	4			3			1
Opis pojedinih obrazaca				4			4
Dijagram obrazaca				2			3
Sekvencijski dijagrami				4			4
Opis ostalih zahtjeva				1			2
Arhitektura i dizajn sustava	4	10		6			
Baza podataka		12					
Dijagram razreda				1			
Dijagram stanja				2			
Dijagram aktivnosti				2			
Dijagram komponenti				2			
Korištene tehnologije i alati	5			2			
Ispitivanje programskog rješenja	1	4			3		3
Dijagram razmještaja							
Upute za puštanje u pogon	2	10					
Sastanci	3	3	3	3	3	3	3

Zadaci	Luka Sever	Dominik Topić	Bruno Pleše	Diana Agejev	Vito Gašparac	Amalija Novalić	Dora Vukelić
Zaključak i budući rad				1			
Popis literature				1			
Izrada početne stranice	2					3	
Izrada baze podataka		6					
Spajanje s bazom podataka		4					
Izrada prezentacije							
Dizaj korisničkog sučelja							9
Izrada korisničkog sučelja	1		5	2		15	7
Izrada programske potpore	4	8	45	1	40	10	10

Dijagram pregleda promjena

Kjučni izazovi i rješenja

Tijekom razvoja projekta StanPlan suočili smo se s nekoliko ključnih izazova:

1. Kašnjenje u razvoju i raspodjela zadataka - Zbog opsežnosti zadatka i paralelnog rada na backendu i frontendu, došlo je do kašnjenja u implementaciji nekih funkcionalnosti (npr. kreiranje sastanaka i zgrade). Raspodjela posla je povremeno bila neujednačena, što je utjecalo na dinamiku tima. - Nismo ostavili dovoljno vremena za deploy prije prve predaje jer nismo znali opseg posla. - *Rješenje:* Bolje smo raspodjelili poslove i pojačali komunikaciju u grupi. Zapamtili smo da treba na vrijeme istražiti koliko vremena je potrebno za odredenu aktivnost.

2. Tehnički izazovi u integraciji i implementaciji - Integracija s vanjskim servisima (StanBlog) i implementacija e-mail notifikacija pokazali su se zahtjevnima zbog ograničenih resursa i vremena. Dio funkcionalnosti je ostao djelomično implementiran (npr. povezivanje točaka s diskusijama, potpuna automatizacija notifikacija). - *Rješenje:* Fokusirali smo se na stabilnu osnovnu funkcionalnost i ostavili napredne mogućnosti za budući rad. Dokumentirali smo ograničenja i planove za proširenje.

3. Upravljanje bazom podataka i migracijama - Prilagodba aplikacije za rad s H2 i PostgreSQL bazom zahtijevala je dodatno testiranje i prilagodbu SQL shema. Dio migracija je pisan ručno, što je povećalo mogućnost grešaka. - *Rješenje:* Uveli smo jasne procedure za testiranje migracija i redovito provjeravali konzistentnost između entiteta i baze.

4. Testiranje i pokrivenost rubnih slučajeva - Nisu svi rubni slučajevi i neimplementirane funkcionalnosti pokriveni testovima. - *Rješenje:* Testirali smo osnovne funkcionalnosti, a za napredne mogućnosti ostavili prijedloge za buduće testove.

Naučene lekcije:

- Važnost rane i jasne podjele zadataka te kontinuirane komunikacije u timu.
- Potreba za realnim planiranjem opsega posla i pravovremenim prepoznavanjem ograničenja.
- Dokumentiranje djelomično implementiranih i neimplementiranih funkcionalnosti olakšava budući razvoj i održavanje.