Programsko inženjerstvo ak. god. 2025./2026.

Sveučilište u Zagrebu

Fakultet elektrotehnike i računarstva

Susjedi

Tim: TG09.3

Ime: Susjedi

URL: https://susjedi.onrender.com

Nastavnik: Vlado Sruk

Uvod

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

Neke od stvari koje će naše rješenje omogućiti su:

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

Zašto je naš projekt koristan?

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

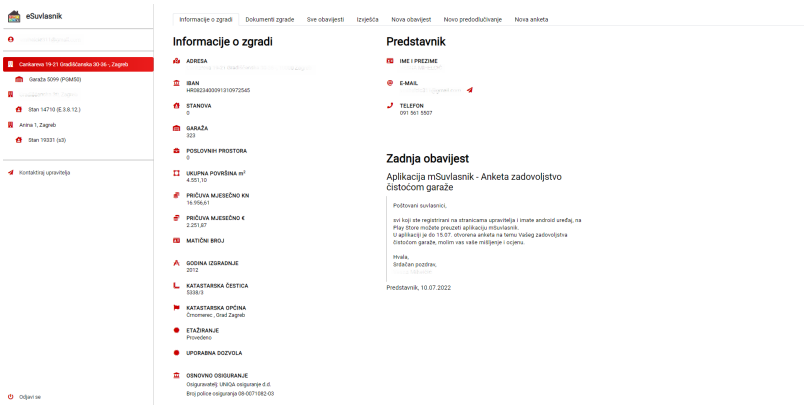
Postojeća rješenja

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

Upravljanje zgradama

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

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



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

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

Što možemo naučiti iz postojećih rješenja?

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

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

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

Funkcionalni zahtjevi

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

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

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

Nefunkcionalni zahtjevi

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

ID zahtjeva	Opis	Prioritet
NF-1.16	Pristup podacima i funkcionalnostima mora biti ograničen na temelju korisničkih uloga.	Visok
NF-1.17	Podaci o prijavama ne smiju biti izgubljeni ili oštećeni.	Visok

Dionici sustava:

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

Aktori i njihovi funkcionalni zahtjevi

A-1 Administrator (inicijator) može:

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

A-2 Predstavnik suvlasnika (inicijator) može:

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

A-3 Suvlasnik (sudionik) može:

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

Obrasci uporabe

Opisi obrazaca uporabe

UC1 – Kreiranje novih korisnika

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

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

UC2 – Prijava koristeći OAuth 2.0

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

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

UC3 – Promjena lozinke

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

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

6. Korisnik se preusmjerava na odgovarajuću stranicu

UC4 – Kreiranje sastanka

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

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

UC5 – Dodavanje točaka dnevnog reda sastanku

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

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

UC6 – Dodavanje zaključaka točkama dnevnog reda

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

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

UC7 – Postavljanje stanja "Objavljen"

- **Glavni sudionik:** Predstavnik suvlasnika

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

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

UC8 – Postavljanje stanja "Arhiviran"

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

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

UC9 – Slanje obavijesti elektroničke pošte

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

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

UC10 – Pregled oglasne ploče

- **Glavni sudionik:** Suvlasnik
- **Cilj:** Pregled "Objavljenih" i "Arhiviranih" sastanaka i njihovih zaključaka
- **Sudionici:** Suvlasnik, Baza podataka
- **Preduvjet:** Suvlasnik je prijavljen u aplikaciju, postoje odgovarajući sastanci
- **Opis osnovnog tijeka:**

1. Suvlasnik odabire oglasnu ploču
2. Aplikacija dohvaća sastanke iz baze podataka
3. Ako suvlasnik odabere "Objavljen" sastanak, dobit će poruku je li sastanak započeo
4. Ako suvlasnik odabere "Arhiviran" sastanak, moći će pregledati sve njegove zaključke

UC11 – Interakcija sa sastancima

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

- **Sudionici:** Suvlasnik, Baza podataka
- **Preduvjet:** Suvlasnik je prijavljen u aplikaciju, odabrao je "Objavljen" sastanak
- **Opis osnovnog tijeka:**

1. Suvlasnik odabire "Objavljen" sastanak
2. Ako sastanak još nije počeo, suvlasnik može potvrditi prisustvo
3. Pohrana prisustva u bazi podataka
4. Ako je sastanak počeo, suvlasnik može glasati za točke dnevnog reda
5. Ako suvlasnik ponovno pokuša glasati za točku za koju je to već učinio, dobit će poruku greške

UC12 – Unos adrese StudBlog servera

- **Glavni sudionik:** Administrator
- **Cilj:** Unos adrese StudBlog servera
- **Sudionici:** Administrator, Baza podataka
- **Preduvjet:** Administrator je prijavljen u aplikaciju
- **Opis osnovnog tijeka:**

1. Administrator odabire opciju spajanja aplikacije sa StudBlogom
2. Administrator unosi adresu StudBlog servera
3. Adresa se pohranjuje u bazu podataka

Dijagrami obrazaca uporabe

1. Visokorazinski dijagram obrazaca uporabe cijelog sustava

 pregled *Slika 3.1: Dijagram obrazaca uporabe: cjelokupni pregled sustava*

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

2. Dijagram obrazaca uporabe za ključne značajke

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

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

3. Dijagram obrazaca uporabe za korisničke uloge

 uloge *Slika 3.3: Dijagram obrasca uporabe: objavljivanje novog sastanka*

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

4. Dijagram obrazaca uporabe za osnovne poslovne procese

 procesi *Slika 3.4: Dijagram obrasca uporabe: pregled sastanaka*

Dijagram obrasca uporabe koji prikazuje ulogu suvlasnika na sastancima.

5. Dijagram obrazaca uporabe za kritične sustave i integracije

 integracije *Slika 3.5: Dijagram obrasca uporabe: spajanje sa StudBlog serverom*

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

Sekvencijski dijagrami

Objavljivanje sastanaka

Sastanke kreira predstavnik suvlasnika. Prilikom odabira opcije kreiranja sastanka, nudi mu se opcija za ručno kreiranje i opcija za kreiranje iz StudBlog diskusije. Ako odabere prvu opciju, ručno unosi naslov, sažetak, vrijeme i lokaciju sastanka. Kada su ovi podatci ispunjeni, predstavnik suvlasnika može sastanku postaviti stanje "Planiran". Ako odabere drugu opciju, aplikiacija dohvaća diskusije sa StudBlog servera i prikazuje mu ih. Odabirom jedne diskusije stvara se odgovarajući sastanak i njegovo stanje se postavlja na "Planiran". Planiranom sastanku potrebno je dodati barem jednu točku prije nego ga se može objaviti. Predstavnik suvlasnika može dodati još proizvoljan broj točaka. Točke se, slično kao i sastanci, mogu dodati i ručno i koristeći diskusije. Za točke je potrebno odabrati pravnu važnost. Kada se stanje sastanka postavi na "Objavljen", pošalje se obavijest elektroničkom poštom suvlasnicima.  objavljivanje sastanka *Slika 3.6: Sekvencijski dijagram: Objavljivanje sastanaka*

Obavljanje sastanka

Predstavnik suvlasnika je zadužen i za obavljanje sastanaka. Predstavnik stanara može obaviti samo objavljen sastanak. Odabirom sastanka sastanak se započinje, njegovo predviđeno trajanje je od 15 do 60 minuta. Nakon sastanka, predstavnik može dodavati zaključke točkama dnevnog reda. Kada su zaključci dodani svim točkama, predstavniku stanara se nudi opcija arhiviranja sastanka. Obavijest o arhiviranju sastanka šalje se suvlasnicima putem elektroničke pošte.  objavljivanje sastanka *Slika 3.7: Sekvencijski dijagram: Obavljanje sastanaka*

Glasanje

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

 glasanje *Slika 3.8: Sekvencijski dijagram: Glasanje*

ID Obrasca uporabe	Uključenost funkcionalnih zahtjeva
UC1	F-001
UC2	F-003

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

Arhitektura sustava

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

Opis arhitekture

Stil arhitekture

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

Podsustavi

Sustav je podijeljen u tri glavna podsustava:

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

Preslikavanje na radnu platformu

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

Spremišta podataka

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

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

Mrežni protokoli

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

Globalni upravljački tok

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

Sklopovsko-programski zahtjevi

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

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

Obrazloženje odabira arhitekture

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

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

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

Pri oblikovanju arhitekture primijenjeni su sljedeći principi:

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

Razmatrane alternative

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

Zaključak

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

Organizacija sustava na visokoj razini

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

Organizacija aplikacije

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

Frontend sloj

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

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

Backend sloj

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

Controller sloj

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

Service sloj

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

Repository sloj

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

Model sloj (Entities)

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

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

Dijagram visoke razine


dijagram visoke razine

Reference

Architect modern web applications with ASP.NET core and azure - .NET, Architect modern web applications with ASP.NET Core and Azure - .NET | Microsoft Learn. Dostupno: https://learn.microsoft.com/en-

us/dotnet/architecture/modern-web-apps-azure/ (7. studeni 2024.).

Baza podataka

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

- Users
- Meetings
- Agenda_items
- Conclusions
- Meeting_attendances
- System_config
- Flyway_schema_history

Opis tablica

Users

Ova tablica sadržava informacije o korisničkim računima stanara. Sadržava atribute: id(primarni ključ), username, password, email, role, created_at, oauth_provider, oauth_provider_id i last_login. Ovaj entitet ima relaciju jedan na više sa tablicama meetings i meeting attendances.

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

Meetings

Ova tablica sadržava informacije o sastancima koje kreiraju korisnici. Sadržava atribute: id(primarni ključ), title, summary, meeting_ts, location, state, created_by(strani ključ), created_at i updated:at. Ovaj entitet ima relaciju jedan na više s tablicom agenda_items te relaciju više na više s users putem tablice meeting_attendances.

Atribut	Tip podatka	Opis varijable

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

Agenda_items

Tablica koja sadrži točke dnevnog reda pripadajuće sastanku. Svaka točka pripada jednom sastanku; može imati oznaku da ima pravni učinak ili da zahtijeva glasovanje. Sadrži atribute: id(primarni ključ), meeting_id(strani ključ), title, description, order_number, has_legal_effect, requires_voting, stanblog_discussion_url, created_at. Relacije: više na jedan prema meetings, jedan na jedan prema conclusions (putem jedinstvenog FK).

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

Conclusions

Tablica zaključaka za pojedinu točku dnevnog reda. Sadrži atribute: id(primarni ključ), agenda_item_id(strani ključ), content, voting_result i created_at. Ima relaciju više na jedan prema agenda_items.

Atribut	Tip podatka	Opis varijable

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

Meeting_attendances

Ova tablica predstavlja potvrde dolaska korisnika na sastanke. Sadrži atribute meeting_id(strani ključ), user_id(strani ključ) i confirmed_at. Sadrži relacije više na više s meetings i users.

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

System_config

Ova tablica sadržava konfiguracijske postavke sustava. Sadrži atribute id(primarni ključ), config_key, config_value i updated_at.

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

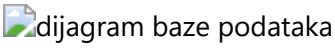
Flyway_schema_history

Ova tablica predstavlja povijest izvršenih migracija nad bazom podataka. Svaki zapis označava jednu migracijsku skriptu koja je izvršena putem Flyway alata. Tablica sadržava installed_rank, version, description, type, script, checksum, installed_by, installed_on, execution_time i success.

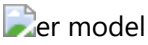
Atribut	Tip podatka	Opis varijable
installed_rank	INTEGER	Redni broj instalirane migracije (PK)
version	VARCHAR(50)	Verzija migracije
description	VARCHAR(200)	Opis migracije
type	VARCHAR(20)	Vrsta migracije (SQL/JDBC/...)

Atribut	Tip podatka	Opis varijable
script	VARCHAR(1000)	Naziv / lokacija skripte
checksum	INTEGER	Kontrolna suma skripte
installed_by	VARCHAR(100)	Korisnik koji je pokrenuo migraciju
installed_on	TIMESTAMP	Vrijeme izvršenja migracije
execution_time	INTEGER	Vrijeme izvršavanja u ms
success	BOOLEAN	Status izvršavanja (TRUE/FALSE)

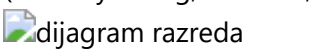
Dijagram baze podataka


dijagram baze podataka

ER model


er model

Dijagram razreda

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

dijagram razreda

Potrebno je priložiti dijagram razreda s pripadajućim opisom. Zbog preglednosti je moguće dijagram razlomiti na više njih, ali moraju biti grupirani prema sličnim razinama apstrakcije i srodnim funkcionalnostima.

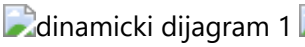
> UML dijagram razreda obavezno uključuje modelirani dijagram razreda!!

dio 1. revizije

Prilikom prve predaje projekta, potrebno je priloˇziti 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.

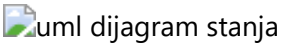
Napomena: Prilikom druge predaje projekta dijagram razreda i opisi moraju odgovarati stvarnom stanju implementacije

Dinamičko ponašanje aplikacije



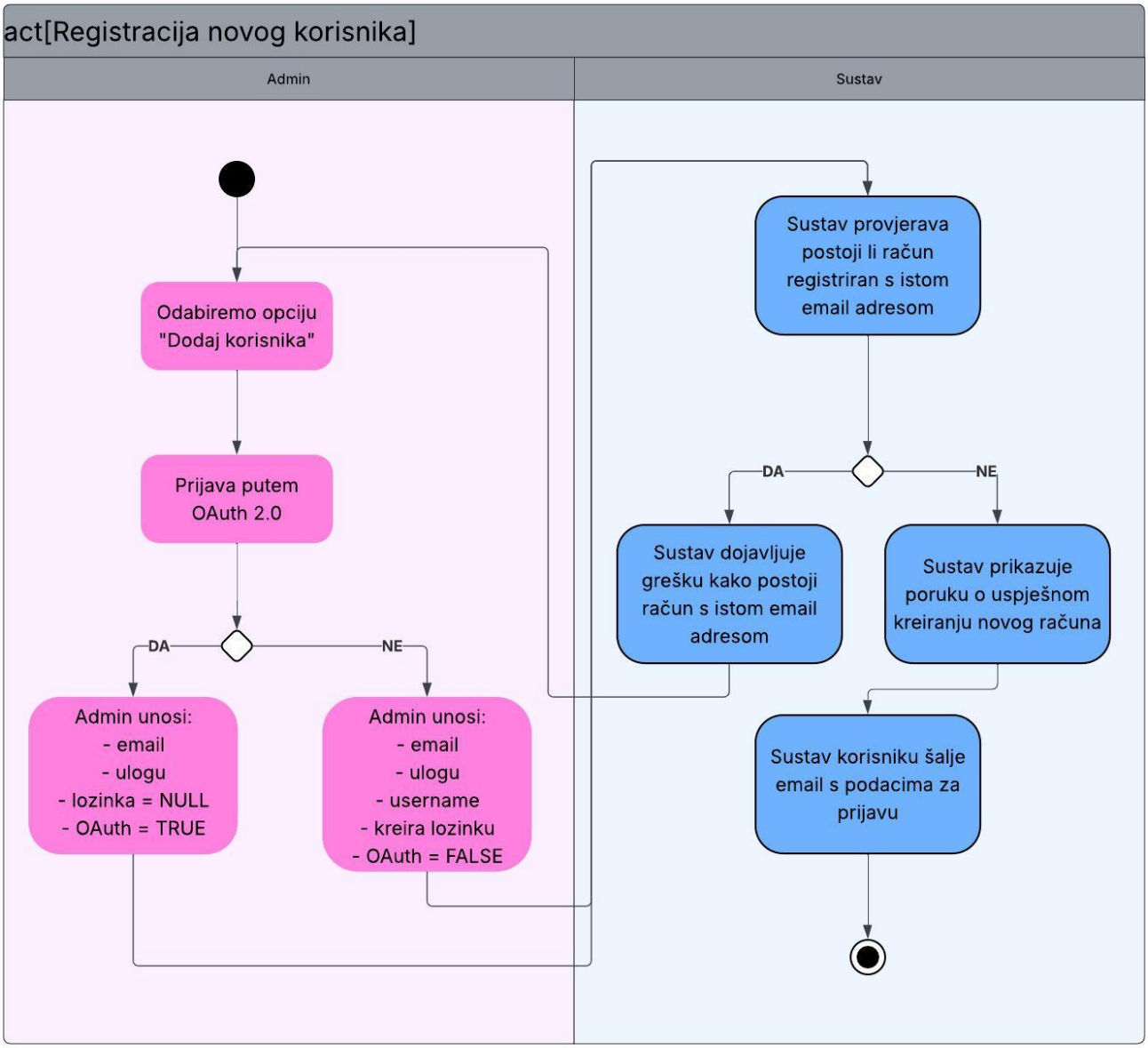
UML dijagrami stanja

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

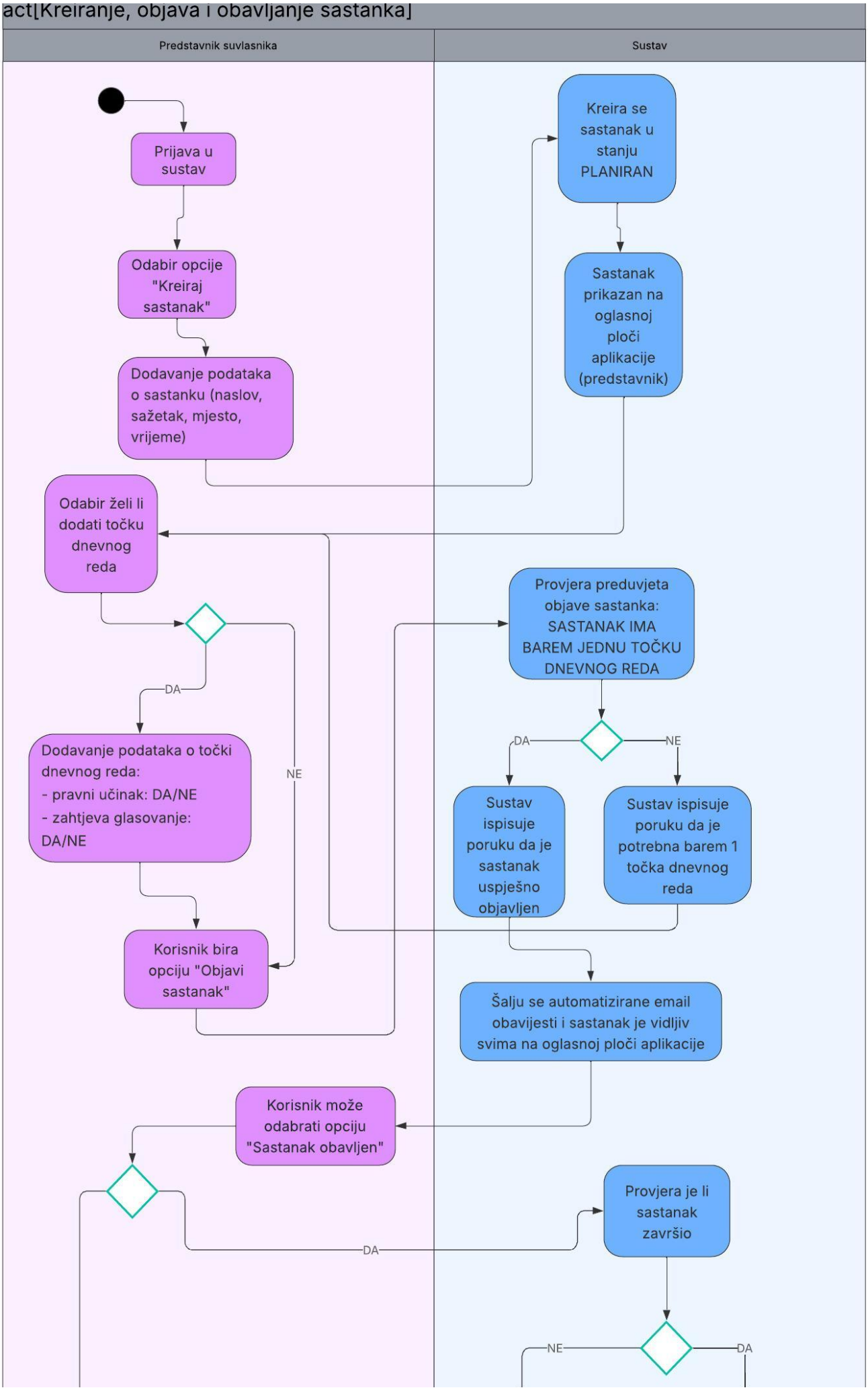


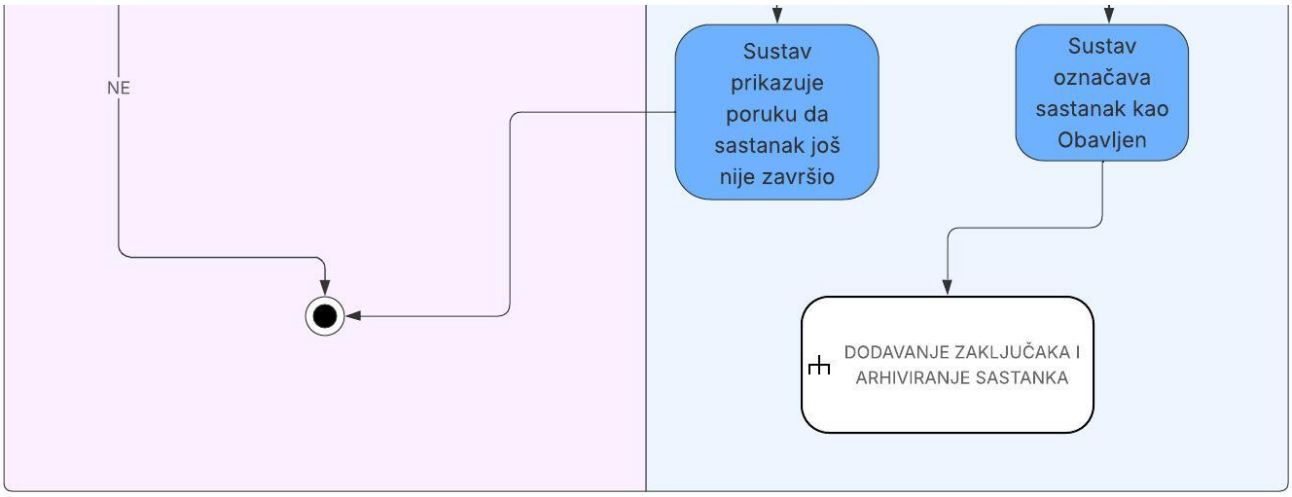
UML dijagrami aktivnosti

Registracija novog korisnika

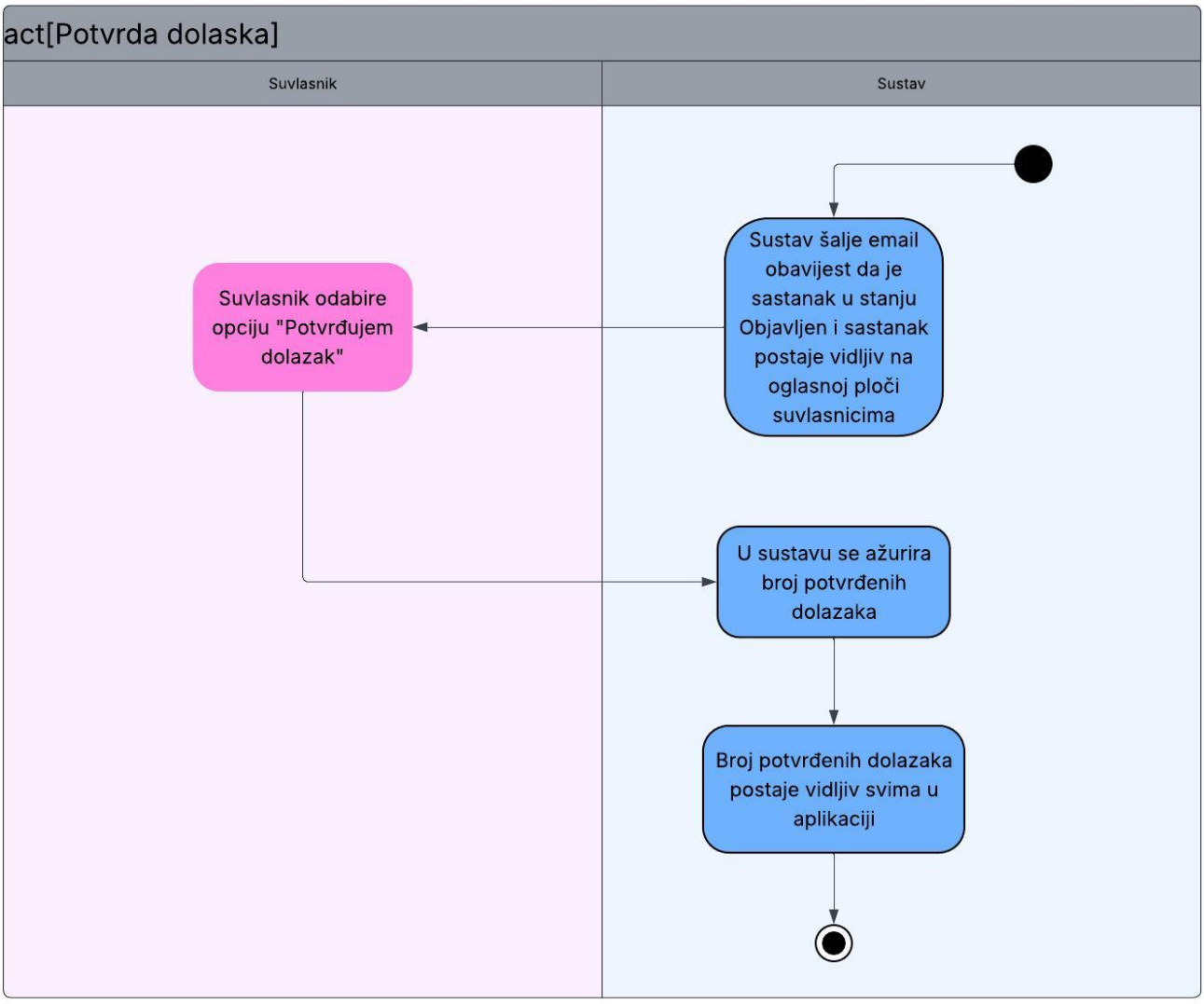


Kreiranje, objava i obavljanje sastanka

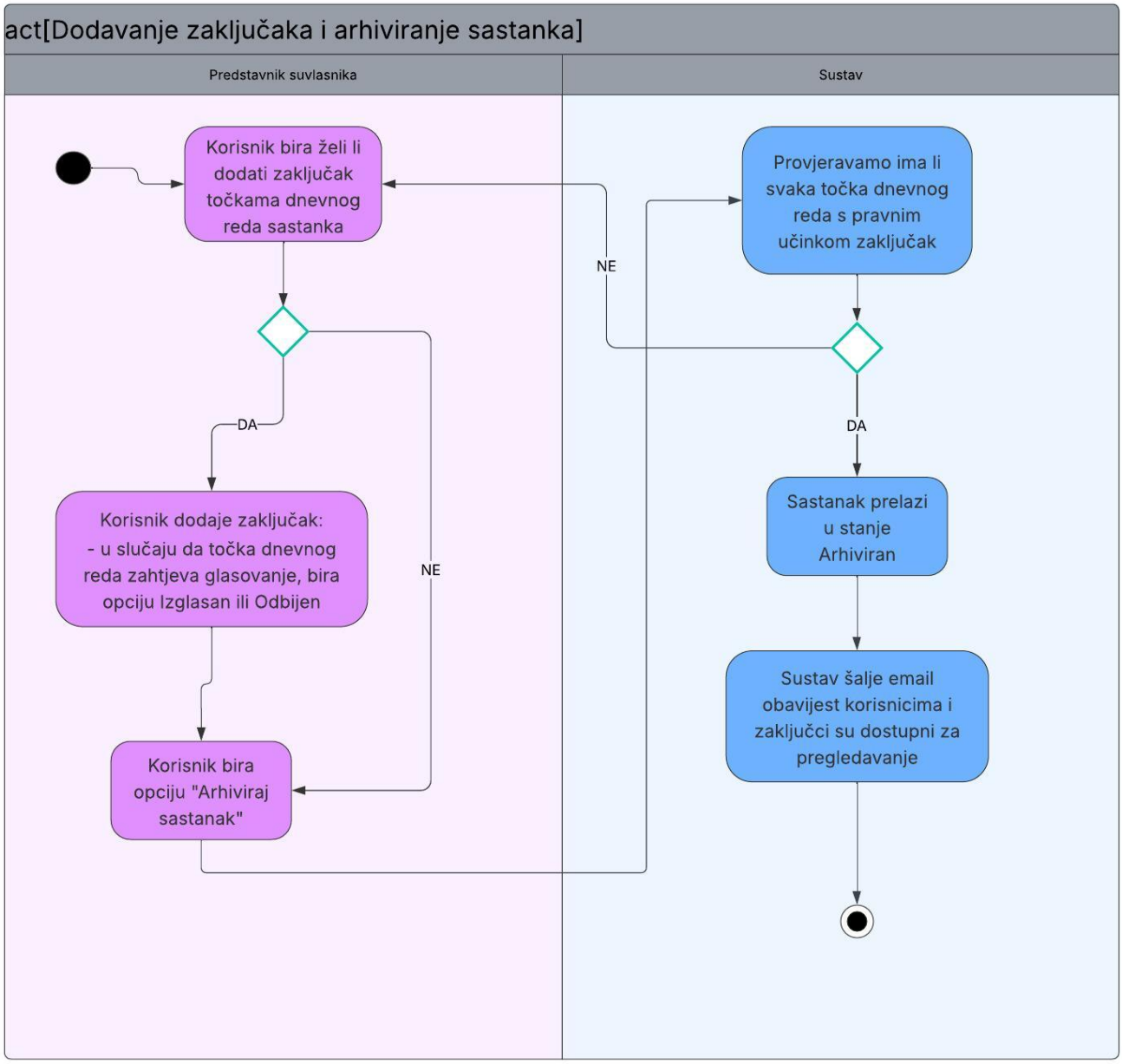

act[Kreiranje, objava i obavljanje sastanka]
Predstavnik suvlasnika
Sustav
Prijava u sustav
Odabir opcije "Kreiraj sastanak"
Dodavanje podataka o sastanku (naslov, sažetak, mjesto, vrijeme)
Kreira se sastanak u stanju PLANIRAN
Sastanak prikazan na oglasnoj ploči aplikacije (predstavnik)
Odabir želi li dodati točku dnevnog reda
DA
NE
Dodavanje podataka o točki dnevnog reda:
- pravni učinak: DA/NE
- zahtjeva glasovanje: DA/NE
Korisnik bira opciju "Objavi sastanak"
Provjera preduvjeta objave sastanka: SASTANAK IMA BAREM JEDNU TOČKU DNEVNOG REDA
DA
NE
Sustav ispisuje poruku da je sastanak uspješno objavljen
Sustav ispisuje poruku da je potrebna barem 1 točka dnevnog reda
Šalju se automatizirane email obavijesti i sastanak je vidljiv svima na oglasnoj ploči aplikacije
Korisnik može odabrati opciju "Sastanak obavljen"
DA
Provjera je li sastanak završio
NE
DA



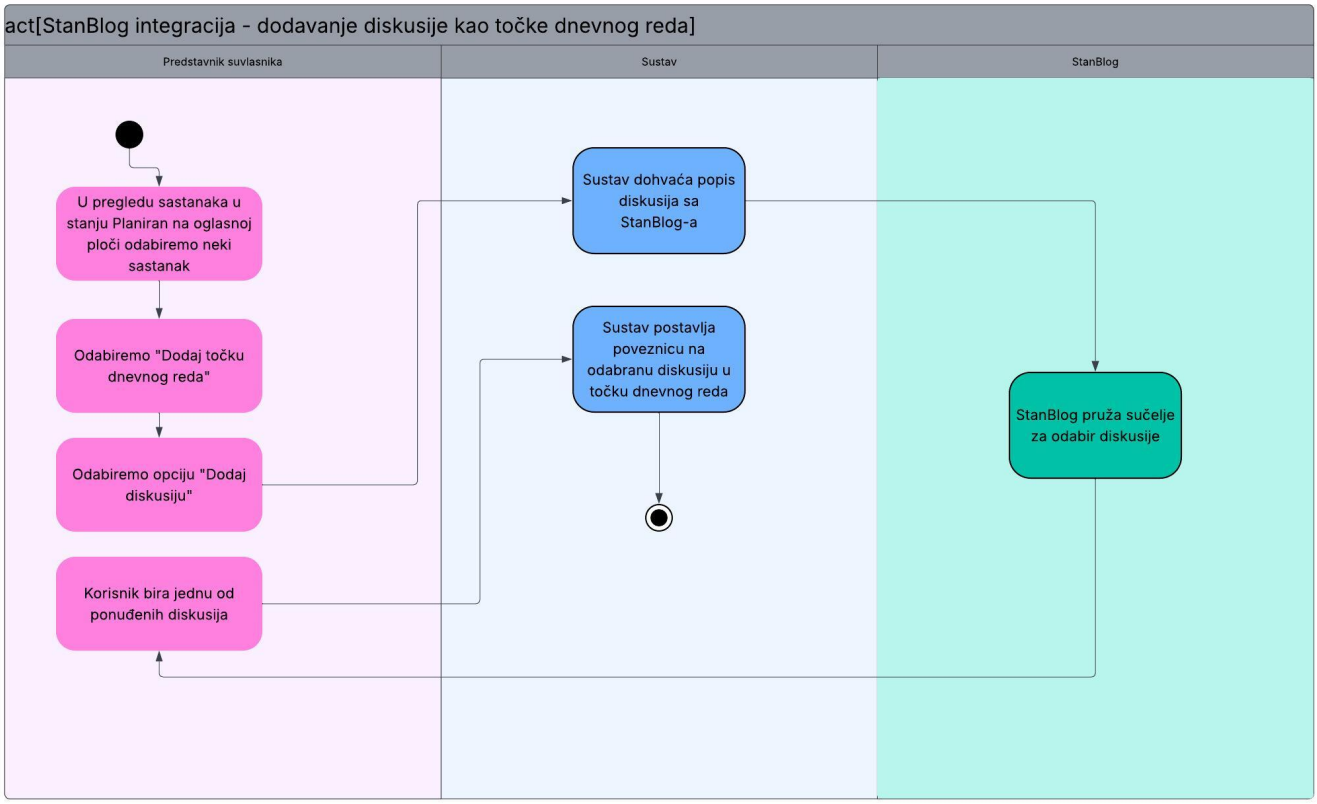
Potvrda dolaska



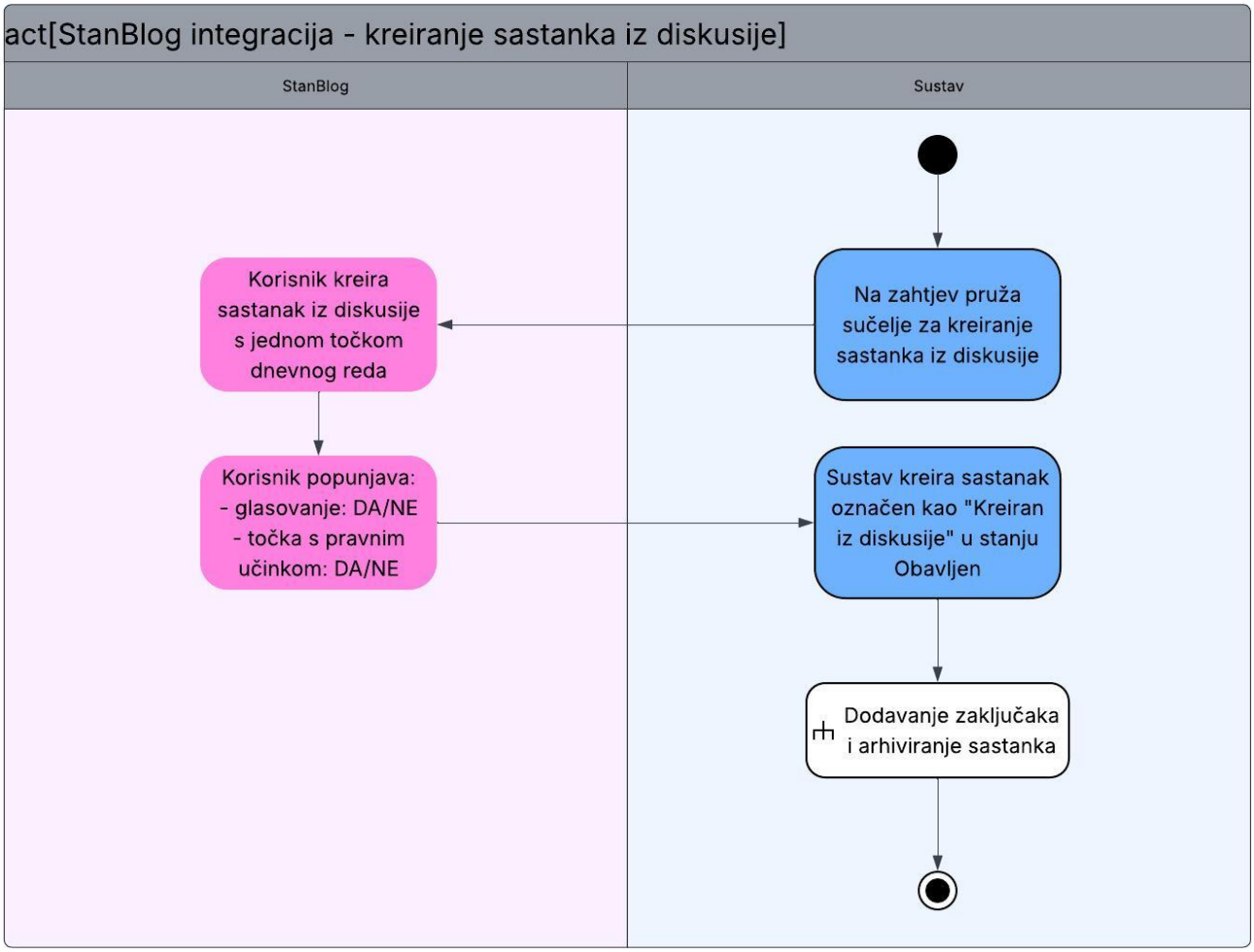
Dodavanje zaključaka i arhiviranje sastanka



StanBlog integracija - dodavanje diskusije kao točke dnevnog reda



StanBlog integracija - kreiranje sastanka iz diskusije



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

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

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

Dijagram komponenata

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

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

Dijagram razmještaja

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

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

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

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

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).

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

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. 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 **

Cilj ispitivanja sustava je testiranje ponašanja cijelog sustava u uvjetima stvarnog korištenja, uz posebnu pažnju na međusobnu povezanost svih komponenti. Ispitivanje treba obuhvatiti sve aspekte sustava i njegovu interakciju s korisnicima.

Zadaci:

Razviti minimalno **4 ispitna slučaja** koji obuhvaćaju:

- **Redovne slučajeve**: očekivano ponašanje sustava.

- **Rubne uvjete**: reakcija sustava na granične ulaze.
- **Poziv nepostojećih funkcionalnosti**: testiranje kako sustav reagira na neimplementirane ili neispravne funkcije.

Struktura ispitnih slučajeva za Selenium:

1. **Ulazi**:
    - Konkretni podaci koji se unose u sustav (npr. korisničko ime i lozinka za prijavu).
    - Simulacija korisničkih akcija (klikanje, unos teksta, navigacija).
2. **Koraci ispitivanja**:
    - Npr. Detaljan opis koraka koje Selenium izvršava:
        1. Otvoriti aplikaciju u pregledniku.
        2. Unijeti podatke u formu.
        3. Kliknuti na gumb za potvrdu.
        4. Verificirati očekivane rezultate (npr. prijava uspješna).
3. **Očekivani izlaz**:
    - Očekivani rezultat ispitivanja (npr. korisnik preusmjeren na početnu stranicu nakon prijave).
4. **Dobiveni izlaz**:
    - Priložiti logove, screenshotove ili izvještaje s generiranim rezultatima (npr. iz JUnit-a ili Selenija).

Alati za ispitivanje sustava:

- **Selenium IDE** ili prikladan obzirom na vaše razvojno okruženje:
    - Jednostavan alat za snimanje korisničkih akcija u pregledniku i automatsko ponavljanje testova.
    - Preporučuje se za osnovne ispitne slučajeve.
- **Selenium WebDriver**:
    - Omogućuje pisanje naprednih testova u različitim programskim jezicima (Java, Python, C#).
    - Preporučuje se za složenije testove, koji zahtijevaju detaljno prilagodbu i automatizaciju.

Primjeri ispitivanjasa Seleniumom:

1. **Formular za prijavu**:

> u dokumentaciji obavezna su specifičnija ispitivanja vaše aplikacije! - **Ulaz**: Korisničko ime = "user@fer.ugnz.hr", Lozinka = "password123".

```
- **Koraci**:
  1. Otvoriti aplikaciju.
  2. Unijeti korisničko ime i lozinku.
  3. Kliknuti na "Prijava".
  4. Provjeriti je li korisnik preusmjeren na početnu stranicu.
- **Očekivani izlaz**: "Prijava uspješna."
```

2. **Rubni uvjet – nevažeća lozinka**:
    - **Ulaz**: Korisničko ime = "user@example.com", Lozinka = "malimedo".
    - **Koraci**:
        1. Unijeti podatke u formu za prijavu.
        2. Kliknuti na "Prijava".
        3. Provjeriti je li prikazana poruka o grešci.
    - **Očekivani izlaz**: "Pogrešno korisničko ime ili lozinka."

Prezentacija rezultata

Za oba tipa ispitivanja (komponenti i sustava) potrebno je:

- Jasno dokumentirati ulaze, korake, očekivane i dobivene rezultate.
- Priložiti slike ekrana, logove ili izvješća generirana alatima (npr. JUnit ili Selenium).
- Detaljno opisati ponašanje sustava, posebno u rubnim uvjetima.
- Navesti broj otkrivenih grešaka!

Korištene tehnologije i alati

Cilj: Jasno i precizno opisati tehnologije korištene u projektu kako bi se olakšalo održavanje, proširenje i suradnja u timu. Uključite informacije:

1. **Programski jezici**: Navesti korištene jezike i njihove verzije (npr. JavaScript 16.13).
2. **Radni okviri i biblioteke**: Detaljno opisati alate za frontend i backend (npr. React 18, Node.js 16).
3. **Baza podataka**: Navesti vrstu baze (npr. PostgreSQL 13).
4. **Razvojni alati**: Popis korištenih IDE-ova, alata za verzioniranje (npr. VS Code, Git 2.34).
5. **Alati za ispitivanje**: Jedinični, integracijski ili UI ispitni sljučajevi (npr. Jest 27, Selenium 4.0).
6. **Alati za razmještaj**: Korišteni alati za implementaciju (npr. Docker 20.10).
7. **Cloud platforma**: Ako je aplikacija hostana, navesti platformu (npr. AWS).

Preporuke za opis:

- **Jasno i precizno**: Izbjegavati tehnički žargon i navesti točne verzije.
- **Obrazloženje izbora**: Objasniti zašto su odabrane određene tehnologije.
- **Opis konfiguracije**: Istaknuti specifične postavke alata i baza.

Primjer:
Za razvoj klijentskog dijela aplikacije korišten je **React** (verzija 18) [ref.], popularna JavaScript biblioteka za izgradnju interaktivnih korisničkih sučelja. React omogućuje stvaranje samostalnih komponenti koje se mogu ponovno koristiti i lako ažurirati, čime se poboljšava učinkovitost razvoja. Za stiliziranje su upotrijebljene **styled-components** (verzija 5.3), što omogućava integraciju stilova izravno unutar Ručat komponenti koristeći **CSS-in-JS** pristup.

Ovaj odjeljak dokumentacije treba dati detaljne smjernice za instalaciju, konfiguraciju, pokretanje i administraciju aplikacije. Cilj je olakšati postavljanje aplikacije na razvojnom, ispitnom i produkcijskom okruženju.

1. Instalacija

Ovdje treba navesti korake potrebne za instalaciju svih potrebnih komponenti:

- **Preduvjeti**: Popis potrebnog softvera i njihovih verzija (npr. Node.js 16, Docker 20.10).
- **Preuzimanje**: Upute za preuzimanje izvornog koda (npr. kloniranje Git repozitorija).

Primjer:

```
git clone https://github.com/Projekt/primjer.git

cd repo
```

Instalacija ovisnosti: Upute za instaliranje ovisnosti.

```
npm install
```

2. Postavke

Detaljne upute za konfiguraciju aplikacije:

- **Konfiguracijske datoteke**: Gdje se nalaze (npr. config.json, .env) i što treba prilagoditi.
    - Primjer .env datoteke:

```
DATABASE_URL=postgres://user:password@localhost:5432/dbname

API_KEY=your_api_key
```

- **Postavke baze podataka**: Upute za inicijalizaciju baze podataka, uključujući migracije i postavljanje inicijalnih podataka.

```
npm run db:migrate

npm run db:seed
```

3. Pokretanje aplikacije

Upute za pokretanje aplikacije u različitim okruženjima:

- **Razvojno okruženje**:

```
npm run dev
```

- **Produkcijsko okruženje**:
    - Prevođenje aplikacije:

```
npm run build
```

- Pokretanje poslužitelja:

```
npm start
```

- **Provjera rada**: Navesti npr. URL (npr. http://localhost:3000) .

4. Upute za administratore

Smjernice za administratore aplikacije nakon puštanja u pogon:

- **Pristup administratorskom sučelju**:
  - URL za admin panel (npr. /admin).
  - Početni podaci za prijavu (ako postoje).
- **Redovito održavanje**:
  - Arhiviranje baze podataka.
  - Pregled logova.
  - Ažuriranje aplikacije (primjer: povlačenje novih verzija iz Git repozitorija i ponovno pokretanje aplikacije).

```
git pull origin main

npm install

npm run build

npm start
```

- **Rješavanje problema**: Kako pristupiti logovima i dijagnosticirati greške (npr. logs/error.log ili docker logs).

---

5. Primjer za Render platformu (Cloud Deploy)

Render je popularna cloud platforma za jednostavno smješanje aplikacija.

- **Priprema repozitorija**:
  - Osigurajte da vaš projekt ima datoteku render.yaml ili Dockerfile za konfiguraciju deploya.
  - Primjer render.yaml:

```
services:

- type: web

name: my-web-app

env: node

buildCommand: npm install && npm run build
```

> startCommand: npm start
>
> plan: free

- **Postavljanje na Render**:
  - Prijavite se na Render.
  - Kreirajte novi **Web Service** i povežite ga s vašim GitHub repozitorijem.
  - Konfigurirajte postavke (npr. build i start komande).
  - Dodajte environment varijable (npr. DATABASE_URL, API_KEY).
- **Pokretanje aplikacije**:
  Render će automatski preuzeti repozitorij, instalirati ovisnosti i pokrenuti aplikaciju. Nakon deploya, aplikaciji možete pristupiti putem generiranog URL-a (npr. https://my-web-app.onrender.com).

Opis prisutpa aplikaciji na javnom poslužitelju

Pristup aplikaciji Dokumentirajte postupak i pružite jasne smjernice za korištenje aplikacije na javnom poslužitelju.

- Navedite ograničenja!
- U uputama obuhvatite kako korisnici mogu pristupiti aplikaciji putem internetskog preglednika.
- Priložite korake za pristup administratorskom sučelju ako je primjenjivo.

U ovom poglavlju potrebno je napisati osvrt na vrijeme izrade projektnog zadatka, koji su tehnički izazovi prepoznati, jesu li riješeni ili kako bi mogli biti riješeni, koja su znanja stečena pri izradi projekta, koja bi znanja bila posebno potrebna za brže i kvalitetnije ostvarenje projekta i koje bi bile perspektive za nastavak rada u projektnoj grupi. Potrebno je točno popisati funkcionalnosti koje nisu implementirane u ostvarenoj aplikaciji.

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

1. Programsko inženjerstvo, FER ZEMRIS, http://www.fer.hr/predmet/proinz
2. I. Sommerville, "Software engineering", 8th ed, Addison Wesley, 2007.
3. T.C.Lethbridge, R.Langaniere, "Object-Oriented Software Engineering", 2nd ed. McGraw-Hill, 2005.
4. I. Marsic, Software engineering book", Department of Electrical and Computer Engineering, Rutgers University, http://www.ece.rutgers.edu/˜marsic/ books/SE
5. The Unified Modeling Language, https://www.uml-diagrams.org/
6. Astah Community, http://astah.net/editions/uml-new

Dnevnik sastajanja

1. sastanak

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

2. sastanak

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

Plan rada

> Tablični/Gantt/Kanban prikaz
>
> - Prikaz vremenskog plana rada ključnih aktivnosti (tjedna granulacija)
> - Uključuje akronime angažiranih članova tima TODO: primjer

Tablica aktivnosti

	Petar Ćurić	**Andro Gabaj**	**Matej Glavaš**	**Ante Ivančić**	**Filip Matijević**	**Jakov Schramadei**
Upravljanje projektom					8	4
Opis projektnog zadatka					1	
Funkcionalni zahtjevi			1		2	
Opis pojedinih obrazaca				1		
Dijagram obrazaca				3		
Sekvencijski dijagrami				8		
Opis ostalih zahtjeva	1		2			
Arhitektura i dizajn sustava		8	2			

	Petar Ćurić	Andro Gabaj	Matej Glavaš	Ante Ivančić	Filip Matijević	Jakov Schramadei
Baza podataka			2		2	
Dijagram razreda			2			
Dinamički dijagrami		2.5				
Dijagram stanja		0.5				
Dijagram aktivnosti					2	
Dijagram komponenti						
Korištene tehnologije i alati						
Ispitivanje programskog rješenja						
Dijagram razmještaja						
Upute za puštanje u pogon						
Dnevnik sastajanja			0.5		0.5	
Zaključak i budući rad						
Popis literature						
Izrada aplikacije - dodatne stavke						
- izrada početne stranice	6					
- izrada baze podataka					3	
- spajanje s bazom podataka			3		1	
- kreacija prijave i registracije	2		4			
- izrada OAuth2			4		2	
- kreacija mogućnosti dodavanja sastanaka			2		3	
- izrada prezentacije						
DevOps - dodatne stavke						
- arhitektura razmještaja			5			4
- kontinuirana integracija i isporuka			5			5
- postavljanje aplikacije na poslužitelj			6			

Dijagram pregleda promjena

Prenijeti dijagram pregleda promjena nad datotekama projekta. Potrebno je na kraju projekta generirane grafove s githuba prenijeti u ovo poglavlje dokumentacije. Dijagrami za vlastiti projekt se mogu preuzeti s github.com stranice, u izborniku Repository, pritiskom na stavku Contributors.

Kjučni izazovi i rješenja

- Zaključno
- Opis izazova: Glavni izazovi tijekom projekta (npr. kašnjenje u razvoju, tehnički problemi).
- Rješenja: Način na koji su izazovi riješeni, kao i naučene lekcije koje su doprinijele napretku tima.

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

- ovo treba biti vidljivo u samoj dinamici promjena repozitorija

> 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.