Home

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

1.--Opis-projektnog-zadatka

Motivacija

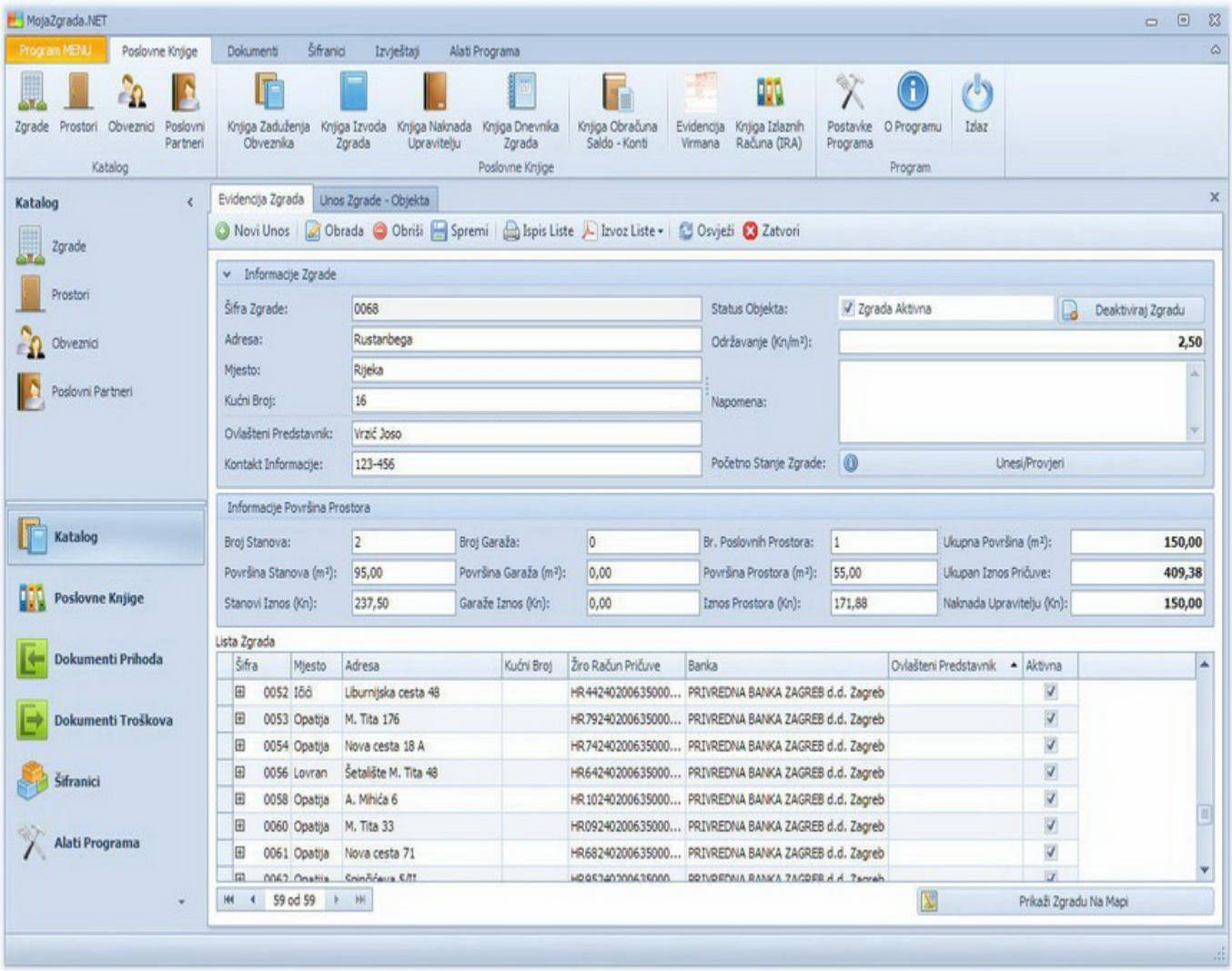
Komunikacija među stanarima u stambenim zgradama često je izazovna. Najčešće se odvija preko oglasne ploče u haustoru zgrade, a posljednjih desetak godina mnoge stambene zajednice komuniciraju putem grupa na aplikacijama poput WhatsAppa ili Vibera. Problem s komunikacijom preko oglasnik ploča jest da 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

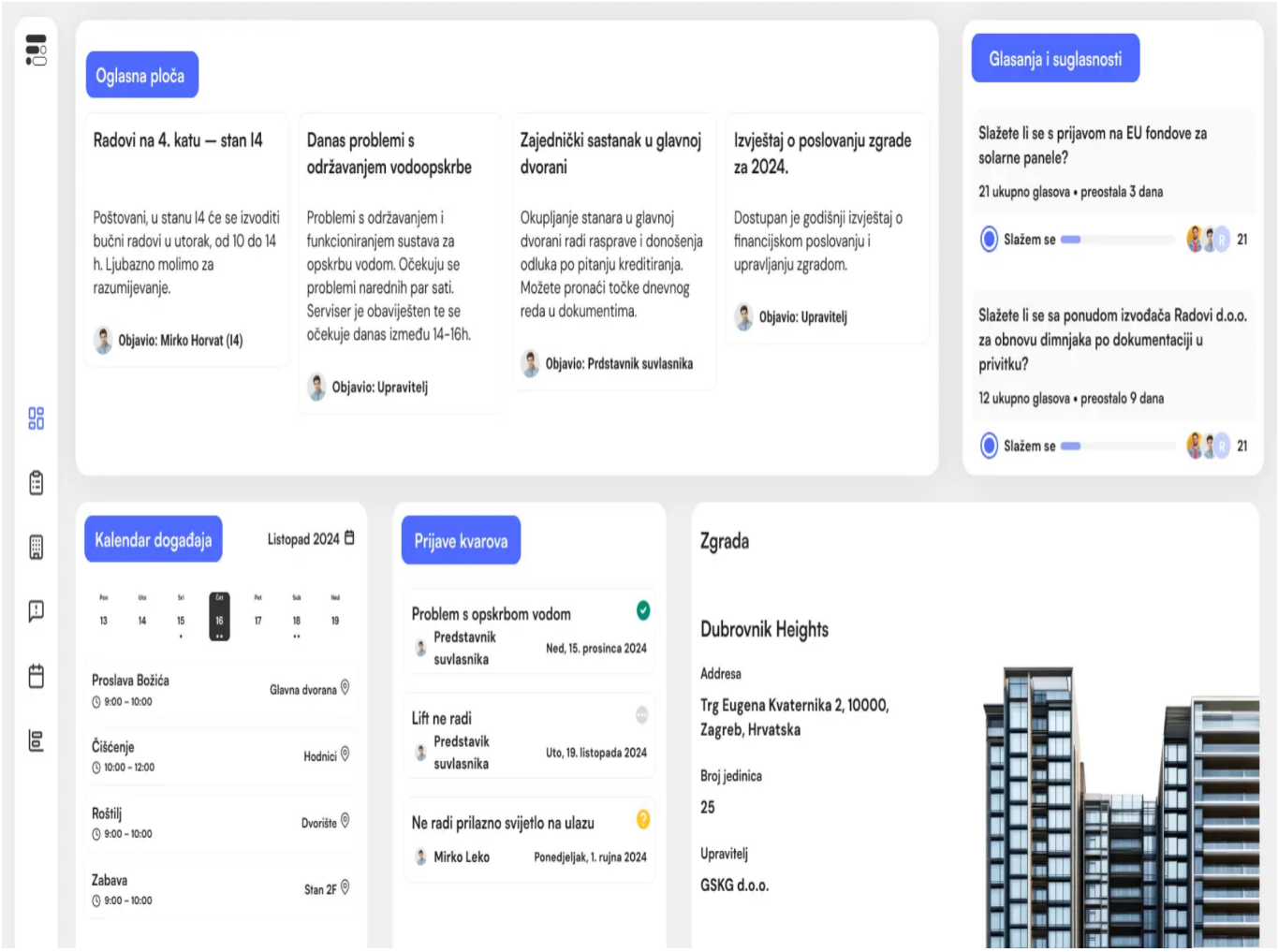
MojaZgrada.NET



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.

2.-Analiza-zahtjeva

Funkcionalni zahtjevi

ID zahtjeva	Opis	Prioritet	Izvor	Kriteriji prihvaćanja
F-001	Sustav omogućuje administratoru kreiranje korisničkih računa za predstavnike suvlasnika i suvlasnike.	Visok	Zahtjev dionika	Administrator može dodati korisnika, dodijeliti e-mail i ulogu te korisnik prima inicijalnu lozinku.

ID zahtjeva	Opis	Prioritet	Izvor	Kriteriji prihvaćanja
F-002	Sustav omogućuje korisnicima promjenu inicijalne lozinke nakon prve prijave.	Srednji	Zahtjev dionika	Korisnik može unijeti staru lozinku, novu lozinku, te potvrditi promjenu.
F-003	Sustav omogućuje korisnicima prijavu putem vanjskog servisa za autentifikaciju (OAuth 2.0).	Visok	Zahtjev dionika	Korisnik se može prijaviti putem odabranog vanjskog servisa.
F-004	Sustav omogućuje predstavniku kreiranje novog sastanka u stanju "Planiran".	Visok	Zahtjev dionika	Predstavnik može unijeti naslov, sažetak namjere, vrijeme i mjesto sastanka.
F-005	Sustav omogućuje predstavniku dodavanje točaka dnevnog reda sastanku.	Visok	Zahtjev dionika	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	Zahtjev dionika	Sastanak s barem jednom točkom može biti objavljen.
F-007	Sustav šalje obavijesti e-mailom suvlasnicima prilikom objave i arhiviranja sastanka.	Visok	Zahtjev dionika	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	Zahtjev dionika	Suvlasnik vidi listu objavljenih sastanaka i njihove detalje.
F-009	Sustav omogućuje suvlasnicima da potvrde sudjelovanje na sastanku.	Visok	Zahtjev dionika	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	Zahtjev dionika	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	Zahtjev dionika	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	Zahtjev dionika	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	Zahtjev dionika	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	Zahtjev dionika	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	Zahtjev dionika	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 te kreiranje korisnika
3. Administrator - odgovoran za održavanje sustava i upravljanje zahtjevima korisnika

Aktori i njihovi funkcionalni zahtjevi:

A-1 Suvlasnik može:

> Pregledavati objavljene sastanke (F-008)
> Potvrditi sudjelovanje na sastanku (F-009)
> Primati obavijesti o sastancima (F-007)
> Sudjelovati u glasanju i komunikaciji (F-007, F-009)

A-2 Predstavnik suvlasnika može:

> Kreirati i upravljati sastancima (F-004, F-005, F-006, F-010, F-011, F-012)
> Komunicirati sa suvlasnicima (F-007)

> Upravljati dnevnim redom i zaključcima sastanka (F-005, F-011)
> Povezati točke s diskusijama iz StanBlog aplikacije (F-013)

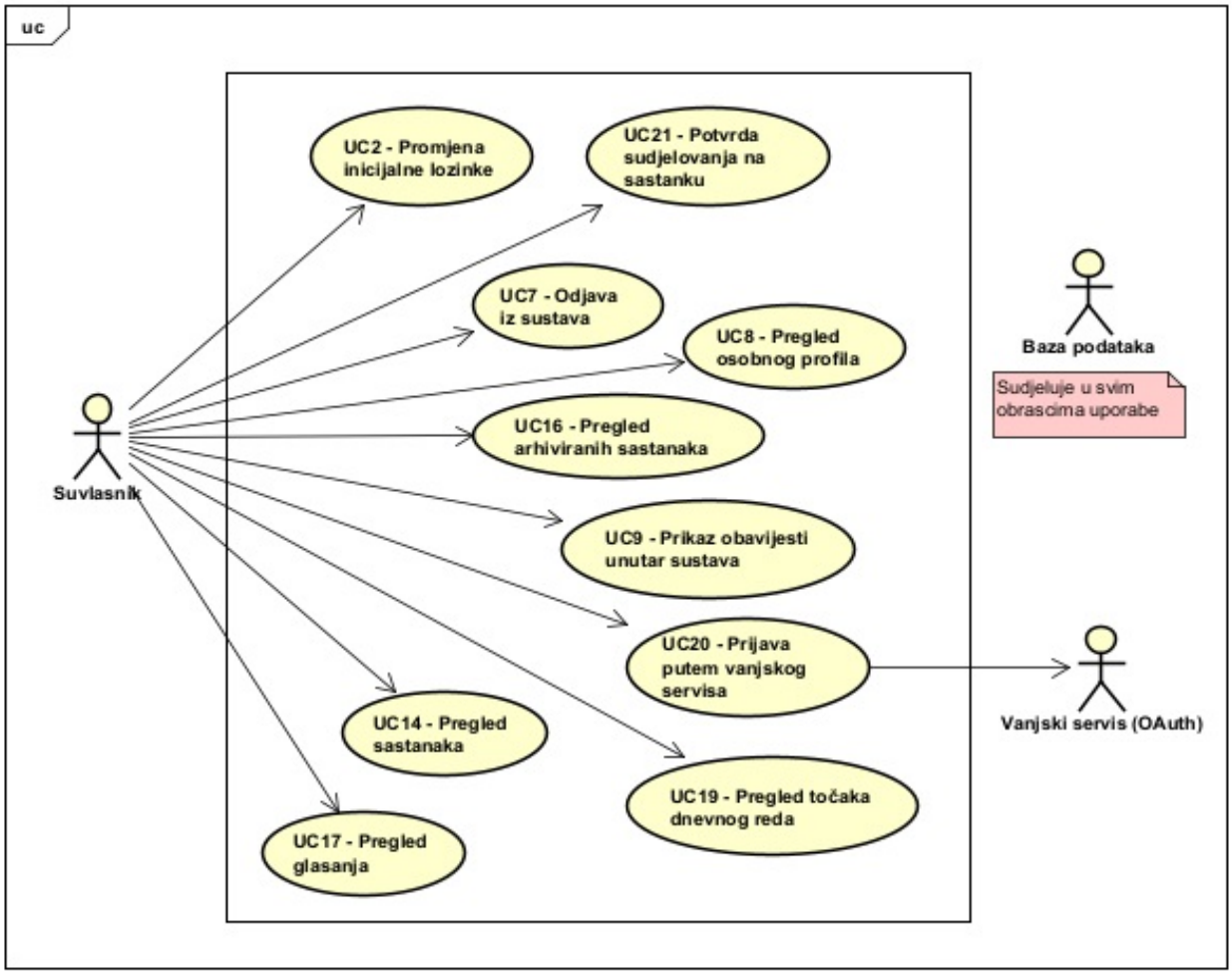
A-3 Administrator može:

> Kreirati korisničke račune i dodjeljivati uloge (F-001, F-015)
> Upravljati integracijama s vanjskim servisima (F-014, F-015)

Obrasci uporabe

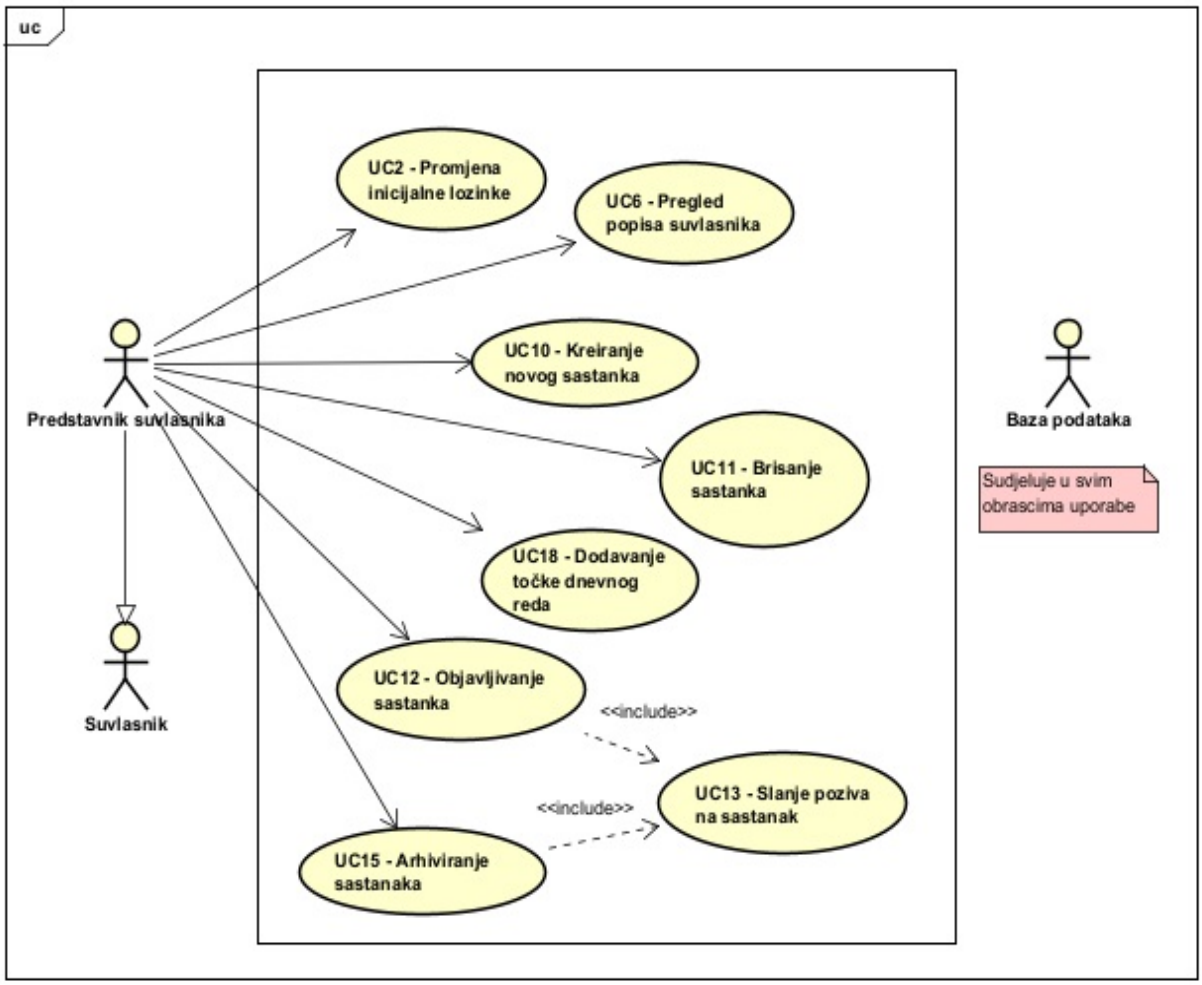
Dijagrami obrazaca uporabe

1. Funkcionalni zahtjevi aktora suvlasnik



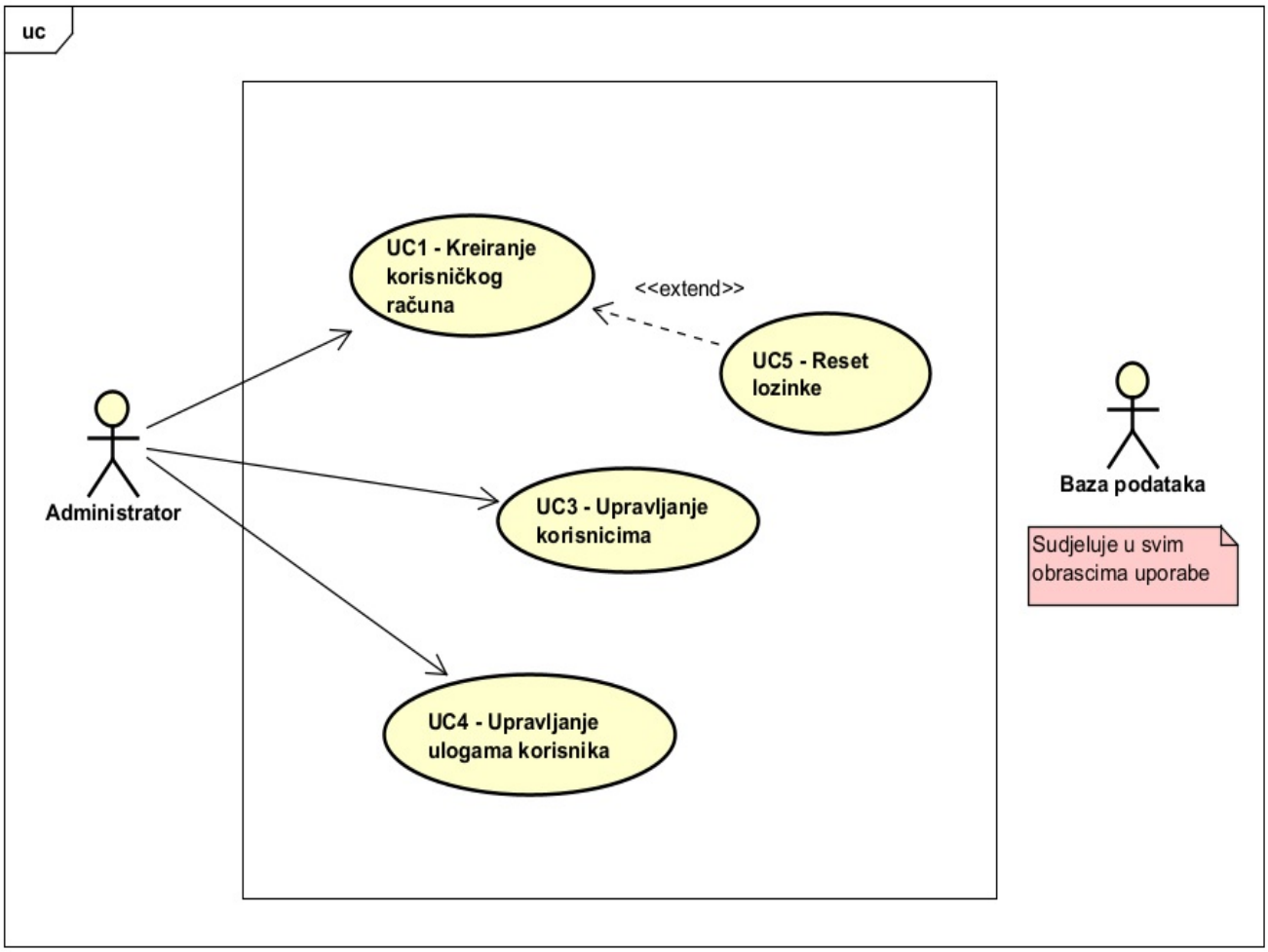
Dijagram prikazuje funkcionalne zahtjeve sustava iz perspektive uloge suvlasnika. Suvlasnik može izvršavati razne operacije, poput promjene inicijalne lozinke, odjave iz sustava, pregleda osobnog profila i arhiviranih sastanaka, praćenja obavijesti, sudjelovanja u glasanjima i sastancima te pregledavanja 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 i korištenje baze podataka. Također je naznačen odnos nasljeđivanja, gdje predstavnik suvlasnika automatski nasljeđuje i 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. Kao i u prethodnim dijagramima, prikazana je i povezanost s bazom podataka koja podržava sve opisane procese.

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 obavljen).
- 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 obavljene 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.

Sekvencijski dijagrami

UC1 – Kreiranje korisničkog računa

```
sequenceDiagram
    participant Admin as Administrator
    participant App as Aplikacija
    participant DB as Baza podataka
    participant Email as E-mail servis

    Admin->>App: Odabir opcije "Dodaj korisnika"
    App->>DB: Provjera postoji li e-mail adresa
    alt E-mail već postoji
        DB-->>App: E-mail postoji
        App-->>Admin: Greška: E-mail adresa je zauzeta
    else E-mail NE postoji
        DB-->>App: E-mail ne postoji
        App->>App: Generira inicijalnu lozinku
        App->>DB: Spremanje podataka o korisniku i lozinci
        DB-->>App: Korisnik je uspješno kreiran
        App->>Email: Slanje e-maila s inicijalnim podacima
        Email-->>App: E-mail poslan
        App-->>Admin: Korisnik je uspješno kreiran
    end
```

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

```
sequenceDiagram
    participant Korisnik
    participant Web as Aplikacija
    participant DB as Baza podataka

    Korisnik->>Web: Odabir "Promjena lozinke" (stara i nova lozinka)
    Web->>DB: Provjera stare lozinke
    alt Lozinka nije ispravna
        DB-->>Web: Lozinka netočna
        Web-->>Korisnik: Greška: Pogrešna lozinka
    else Lozinka je ispravna
        DB-->>Web: Lozinka ispravna
        Web->>DB: Ažuriranje na novu lozinku
        DB-->>Web: Lozinka uspješno promijenjena
        Web-->>Korisnik: Lozinka je uspješno promijenjena
    end
```

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

```
sequenceDiagram
    participant Predstavnik as Predstavnik suvlasnika
    participant App as Aplikacija
    participant DB as Baza podataka
    participant Email as E-mail servis

    Predstavnik->>App: Odabir planiranog sastanka za objavu
    App->>DB: Provjera postoji li barem jedna točka dnevnog reda
    alt Ne postoji nijedna točka
        DB-->>App: Nema točaka
        App-->>Predstavnik: Greška: Nema točaka dnevnog reda, objava nije
moguća
    else Postoji najmanje jedna točka
        DB-->>App: Točka postoji
        App->>DB: Promjena statusa sastanka u "Objavljen"
        DB-->>App: Sastanak je objavljen
        App->>Email: Slanje e-mail obavijesti suvlasnicima
        Email-->>App: E-mailovi poslani
```

```
        App-->>Predstavnik: Sastanak je objavljen
    end
```

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

4.-Arhitektura-i-dizajn-sustava

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, što je konfigurirano u Spring Security sloju.
  - **Planirani podsustavi:**
    - **Podsustav za upravljanje sastancima:** Bit će zadužen za kompletan životni ciklus sastanaka, od kreiranja do arhiviranja.
    - **Podsustav za notifikacije i integraciju:** Bit će zadužen za slanje e-mail obavijesti i komunikaciju sa StanBlog API-jem.
- **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.
- **Sklopovsko-programski zahtjevi:**
  - **Backend:** Java Development Kit (JDK) 17 ili noviji.
  - **Frontend:** Moderni web preglednik.
  - **Baza podataka:** Pokrenuta instanca relacijske baze podataka.

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 (`User`, `Building`, `City`).
  - **View:** JSON odgovori.
  - **Controller:** Klase koje obrađuju HTTP zahtjeve.

Dijagram visoke razine

```
graph TD
    subgraph "Klijent (Web preglednik)"
        A[Frontend Aplikacija - SPA]
    end

    subgraph "Poslužitelj (StanPlan Backend)"
        B{REST API}
        subgraph "Aplikacijski Slojevi"
            B --> C[Controller Sloj]
            C --> D[Repository Sloj - JPA]
        end
        D --> E[(Relacijska Baza Podataka)]
    end

    subgraph "Vanjski Sustavi"
        I[OAuth 2.0 Provider]
    end

    A -- "HTTP/S" --> B
    B -.-> I
```

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. Sastoji se od tri glavne tablice: `USERS`, `BUILDING` i `CITY`, koje su međusobno povezane kako bi osigurale integritet podataka.

Opis tablica

Tablica: USERS

Atribut	Tip podatka	Opis varijable
UserId (PK)	SERIAL	Jedinstveni identifikator korisnika
Username	VARCHAR(20)	Korisničko ime (jedinstveno)
PasswordHash	VARCHAR(100)	Hesirana lozinka za lokalnu prijavu
FirstName	VARCHAR(20)	Ime korisnika
LastName	VARCHAR(20)	Prezime korisnika
Email	VARCHAR(100)	Adresa elektroničke pošte (jedinstvena)
RoleType	VARCHAR(20)	Uloga korisnika (npr. 'ADMIN', 'PREDSTAVNIK')
BuildingId (FK)	INTEGER	Poveznica na zgradu kojoj korisnik pripada

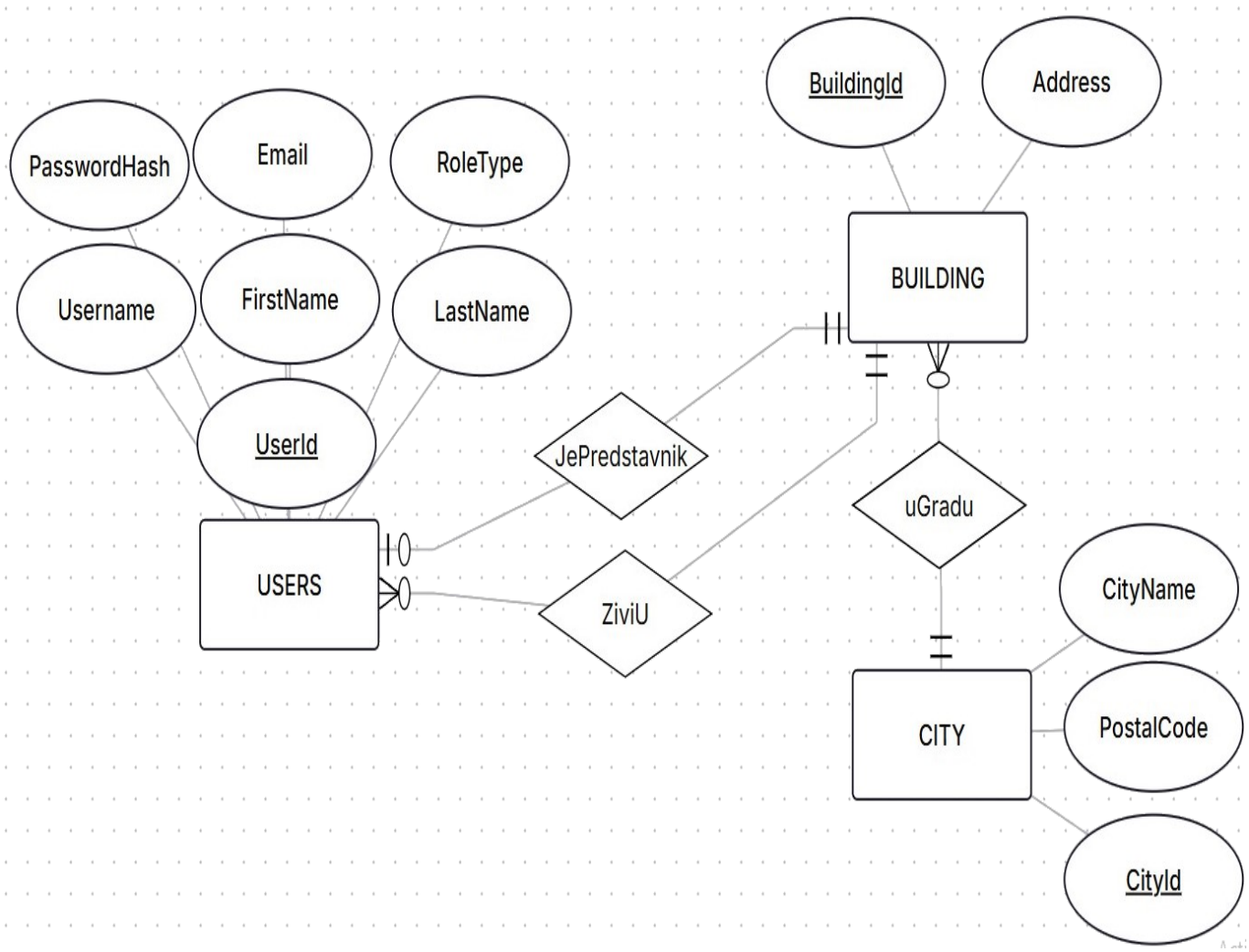
Tablica: CITY

Atribut	Tip podatka	Opis varijable
CityId (PK)	SERIAL	Jedinstveni identifikator grada
PostalCode	INTEGER	Poštanski broj (jedinstven)
CityName	VARCHAR(50)	Naziv grada

Tablica: BUILDING

Atribut	Tip podatka	Opis varijable
BuildingId (PK)	SERIAL	Jedinstveni identifikator zgrade
CityId (FK)	INTEGER	Poveznica na grad u kojem se zgrada nalazi
Address	VARCHAR(100)	Adresa zgrade
RepId (FK)	INTEGER	ID predstavnika suvlasnika (jedinstven, veza na USERS.UserId)

Dijagram baze podataka



ER dijagram ilustrira logičku strukturu baze podataka. Prikazane su glavne entitete: USERS (korisnici), BUILDING (zgrade) i CITY (gradovi), zajedno s njihovim osnovnim atributima i međusobnim odnosima. Dijagram jasno prikazuje kako korisnici pripadaju zgradama, a svaka zgrada nalazi se u određenom gradu. Ujedno su prikazani atributi kao što su korisničko ime, lozinka, e-mail, ime/prezime, adresa zgrade te ime i poštanski broj grada.

```
erDiagram
    USERS {
        INT UserId PK
        VARCHAR Username
        VARCHAR PasswordHash
        VARCHAR FirstName
        VARCHAR LastName
        VARCHAR Email
        VARCHAR RoleType
        INT BuildingId FK
    }
    BUILDING {
        INT BuildingId PK
        INT CityId FK
        VARCHAR Address
        INT RepId FK
    }
    CITY {
```

```
        INT CityId PK
        INT PostalCode
        VARCHAR CityName
    }

    BUILDING ||--|{ USERS : "pripadaju"
    CITY ||--|{ BUILDING : "nalazi se u"
    USERS |o--|| BUILDING : "je predstavnik za"
```

Dijagram prikazuje fizičku 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). Prikazane su i relacije između tablica, npr. veza između USERS i BUILDING (korisnik pripada zgradi), te između BUILDING i CITY (zgrada se nalazi u gradu), kao i veza prema predstavniku zgrade među korisnicima.

Dijagram razreda

```
classDiagram
    direction LR

    class AdminController {
        <>
        -CoOwnerRepository coOwnerRepository
        +ResponseEntity createUser(coOwner)
    }

    class CoOwnerRepository {
        <>
        +JpaRepository
        +Optional findByEmail(String)
    }

    class coOwner {
        <>
        +Long id
        +String firstName
        +String lastName
        +String email
        +Role role
    }

    class Building {
        <>
        +Long id
        +String name
    }

    class Role {
```

```
        <>
        ADMIN
        PREDSTAVNIK
        CO_OWNER
    }

    AdminController ..> CoOwnerRepository : "koristi"
    CoOwnerRepository ..> coOwner : "upravlja"
    coOwner -- Role : "ima"
    Building "1" -- "1" coOwner : "ima predstavnika"
    Building "1" -- "0..*" coOwner : "ima suvlasnike"
```

Dijagram vizualizira osnovne klase i njihove odgovornosti iz domene StanPlan sustava, s naglaskom na backend implementaciju. Prikazuje primjer AdminController klase koja koristi CoOwnerRepository za pristup podacima. CoOwnerRepository upravlja entitetima suvlasnika (coOwner), dok je Building entitet koji ima odnos prema jednom predstavniku i više suvlasnika. Također, vidljiv je i enum tip Role koji definira moguće uloge korisnika (ADMIN, PREDSTAVNIK, SUVLASNIK). Dijagram jasno prikazuje osnovne odnose korištenja („koristi“, „ima“, „upravlja“) među komponentama.

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

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*... .

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*.

5.-Arhitektura-komponenata-i-razmještaja

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).

6.-Ispitivanje-programskog-rješenja

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

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!

7.-Tehnologije-za-implementaciju-aplikacije

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.

8.--Upute-za-puštanje-u-pogon

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.

9.-Zaključak-i-budući-rad

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.

A.--Popis-literature

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/

B.-Dnevnik-promjena-dokumentacije

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
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

Rev.	Opis promjene/dodatka	Autori	Datum
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	13.11.2025

- 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.

C.-Prikaz-aktivnosti-grupe

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

Plan rada

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

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	20			8			

Zadaci	Luka Sever	Dominik Topić	Bruno Pleše	Diana Agejev	Vito Gašparac	Amalija Novalić	Dora Vukelić
Opis projektnog zadatka			2	1			
Funkcionalni zahtjevi	4						
Opis pojedinih obrazaca				4			4
Dijagram obrazaca				2			2
Sekvencijski dijagrami				4			3
Opis ostalih zahtjeva							2
Arhitektura i dizajn sustava	4	2		6			
Baza podataka		6					
Dijagram razreda				1			
Dijagram stanja							
Dijagram aktivnosti							
Dijagram komponenti							
Korištene tehnologije i alati				1			
Ispitivanje programskog rješenja	1						
Dijagram razmještaja							
Upute za puštanje u pogon		4					
Sastanci	2	2	2	2	2	2	2
Zaključak i budući rad							
Popis literature							
Izrada početne stranice	2					3	
Izrada baze podataka		3					

Zadaci	Luka Sever	Dominik Topić	Bruno Pleše	Diana Agejev	Vito Gašparac	Amalija Novalić	Dora Vukelić
Spajanje s bazom podataka							
Izrada prezentacije							
Dizaj korisničkog sučelja							7
Izrada korisničkog sučelja						5	
Izrada programske potpore			19		16	10	

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.