

# HomeFix - Dokumentacija

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

**Sveučilište u Zagrebu**

**Fakultet elektrotehnike i računarstva**

### HomeFix

**Tim: 11.2**

**Ime tima: Bumbari**

**Nastavnik: Vlado Sruk**

### Opis projektnog zadatka

#### Uvod i cilj projektnog zadatka

Cilj ovog projektnog zadatka je razvoj informacijskog sustava koji omogućuje učinkovito upravljanje prijavama kvarova i komunikacijom između stanara, predstavnika suvlasnika i majstora u stambenim zgradama. Sustav je zamišljen kao web platforma putem koje korisnici mogu brzo i jednostavno prijaviti tehničke probleme, pratiti njihovo rješavanje te pristupiti povijesti prijava i izvještajima o održavanju. Današnji proces prijave kvarova u većini stambenih zgrada odvija se putem telefonskih poziva, e-maila ili usmenih najava. Takav način rada je neformalan te često dovodi do kašnjenja u rješavanju problema, gubitka informacija i nesporazuma. Projekt ima za cilj digitalizirati i formalizirati ovaj proces, čime bi se omogućila bolja organizacija, transparentnost i učinkovitost u upravljanju i održavanju kvarovima zgrade.

#### Problematika i opis postojećeg stanja

Održavanje stambenih zgrada predstavlja složen proces koji uključuje više sudionika – stanare, predstavnike suvlasnika, izvođače radova (majstore) i često upravitelje zgrada. U nedostatku digitaliziranog sustava, komunikacija između ovih strana odvija se neučinkovito. Stanari najčešće probleme prijavljuju usmeno ili putem telefona, što otežava praćenje zaprimljenih prijava i njihov status. Predstavnici suvlasnika moraju ručno voditi evidenciju o kvarovima, što često rezultira netočnim podacima te izazovnim upravljanjem više prijava odjednom. Majstori, s druge strane, nemaju jedinstven uvid u sve prijave, pa se često dolazi do preklapanja radova, kašnjenja ili nejasnoća oko prioriteta popravaka. U takvom sustavu nije moguće lako pratiti učinkovitost pojedinih majstora ni izraditi izvještaje o troškovima i vremenu rješavanja problema. Predloženi sustav rješava navedene izazove uvođenjem centralizirane platforme koja omogućuje digitalnu evidenciju svih prijava i aktivnosti. Na taj način svi sudionici imaju pristup ažurnim informacijama, čime se smanjuje broj nesporazuma, povećava učinkovitost i ubrzava proces donošenja odluka.

#### Potencijalna korist projekta

Razvoj ovakvog sustava donosi višestruke koristi svim uključenim stranama.

- Za **stanare**, sustav predstavlja jednostavan alat za prijavu kvarova. Korisnici mogu prijaviti problem u nekoliko koraka, dodati opis, fotografiju i lokaciju te u svakom trenutku pratiti status rješavanja. Time se povećava transparentnost i povjerenje u upravljanje zgradom, jer stanari točno znaju kada i kako je kvar riješen.
- Za **predstavnike suvlasnika**, sustav znatno smanjuje administrativno opterećenje. Umjesto ručne evidencije i usmenih komunikacija, svi podaci dostupni su kroz pregledno sučelje. Moguće je brzo filtrirati prijave prema statusu, vrsti kvara ili zgradi, što omogućuje učinkovito donošenje promjena i odluka. Sustav omogućuje i generiranje izvještaja koji mogu služiti za procjenu troškova održavanja i izvješćivanje suvlasnika. ()
- Za **majstорима** sustav omogućuje jasan uvid u dodijeljene zadatke. Mogu ažurirati status radova, dodati bilješke i fotografije popravaka te na taj način održavati transparentnu komunikaciju s predstavnicima i stanarima. Takav

pristup povećava produktivnost i smanjuje potrebu za dodatnim pozivima i objašnjenjima.

- **Administratori** sustava dobivaju mogućnost upravljanja korisnicima, postavkama i kategorizacijom prijava. Uz to, mogu pratiti opterećenje sustava i nadgledati tehničko funkcioniranje sustava.

### Slična rješenja

**Fixflo** - poznata web platforma koja omogućuje stanarima prijavu kvarova u najamnim nekretninama. Sustav automatski proslijedi prijavu odgovornim osobama i omogućuje praćenje statusa popravka.

- Prednosti - vrlo razvijen sustav obavijesti, automatsko prepoznavanje tipa problema na temelju fotografije, integracija s vanjskim servisima
- Nedostaci - orijentiran isključivo na tržište najma, nedostatak podrške za predstavnike suvlasnika i lokalno prilagođenih uloga
- Razlika - sustav HomeFix je fokusiran na stambene zajednice i suvlasnike, a ne na najmodavce i najmoprime

**Fixflo**

**Powering Property Professionals**

Book your free live tour

**PlanRadar** - međunarodna platforma koja omogućuje praćenje građevinskih i održavateljskih radova. Koristi se u građevinarstvu, nekretninama i facility managementu za evidentiranje problema, komunikaciju i praćenje zadataka.

- Prednosti - moderno i intuitivno sučelje, podrška za dodavanje bilješki i fotografija, mogućnost rada na mobilnim uređajima, integracija s alatima poput Microsoft 365 i Google Workspace
- Nedostaci - sustav je kompleksan i zahtijeva tehničku obuku, a model naplate temeljen je na mjesecnoj pretplati namijenjenoj većim tvrtkama
- Razlika - naš sustav fokusiran je na krajnje korisnike u stambenim zgradama – stanare i predstavnike suvlasnika, uz jednostavniji proces prijave i besplatno korištenje osnovnih funkcija

The screenshot shows the PlanRadar software interface. The top navigation bar includes 'Project' (Office Complex 11 > Ground Floor), 'Form' (All forms in project), 'Filter' (No filter loaded), and a search bar for 'Search tickets' (12 tickets). The left sidebar contains links for 'Dashboard', 'Tickets' (selected), 'Project reports', 'Documents', 'Projects', 'Forms and Lists', 'User Management', 'Statistics', 'Templates', and 'Settings'. The main content area displays a floor plan of a building with various rooms and hallways. Three tickets are overlaid on the plan: 'Delay in delivery' (status: Open, created: 29/09/2023), 'Worn control panel' (status: Closed, created: 28/02/2020), and 'Stuck elevator' (status: Resolved, created: 28/02/2020). Each ticket includes a thumbnail image and a location pin on the floor plan. To the right of the floor plan is a sidebar titled 'EXAMPLE PROJECT' with the following details:

Project name	Conference centre
Address	PlanRadar Lane 1
Project number	1901
Plan number	19 Test 001
Plan size	A2 (8.27m x 5.91m x 2.20m)
Scale	1:100
Plan type	Floor plan
Plan contents	Ground floor
Site owner	PlanRadar GmbH
Author	PlanRadar GmbH
Notes	

## Skup korisnika

Sustav je namijenjen krugu korisnika koji su uključeni u održavanje stambenih zgrada. Glavne skupine korisnika su:

- Stanari – krajnji korisnici koji prijavljuju kvarove, pregledavaju status svojih prijava te mogu ocijeniti izvršene usluge. Njihova glavna potreba je brz i jednostavan način komunikacije s predstavnicima zgrade.
- Predstavnici suvlasnika – osobe odgovorne za upravljanje prijavama, dodjelu zadataka majstorima i pregled izvještaja o održavanju. Njima sustav omogućuje praćenje svih aktivnosti te lakše planiranje i donošenje izvještaja.
- Majstori – tehnički izvršitelji koji zaprimaju prijave, obavljaju popravke i ažuriraju status zadataka. Njihov interes je jasan pregled radnih zadataka i mogućnost lakšeg planiranja posla.
- Administratori – korisnici s najvišim ovlastima koji upravljaju korisničkim računima, postavkama sustava i sigurnosnim protokolima. Oni osiguravaju stabilnost i pouzdanost cijelog sustava.

## Mogućnost prilagodbe rješenja

Sustav je razvijen modularno, što omogućuje jednostavnu prilagodbu različitim tipovima korisnika i scenarijima. Arhitektura aplikacije podržava dodavanje novih funkcionalnosti bez potrebe za promjenom osnovne strukture baze podataka. To omogućuje jednostavno uvođenje novih uloga korisnika, kategorija kvarova ili dodatnih modula, poput obavijesti o troškovima održavanja ili evidencije članarina. Rješenje je također moguće prilagoditi i za druge vrste objekata, kao što su poslovne zgrade, trgovački centri ili druge javne ustanove. Trebalo bi prilagoditi korisničke uloge i vrste prijava kako bi sustav odgovarao različitim kontekstima.

## Opseg projektnog zadatka

Opseg projektnog zadatka obuhvaća razvoj web aplikacije koja uključuje sve ključne funkcionalnosti potrebne za upravljanje procesom održavanja zgrada. U opseg ulazi:

- Registracija korisnika i prijava u sustav putem e-mail adrese ili OAuth 2.0 sustavom.
- Prijava kvarova uz mogućnost dodavanja opisa, fotografije i lokacije.
- Upravljanje prijavama – pregled, filtriranje, ažuriranje i dodjela majstora.
- Pregled i uređivanje profila korisnika.
- Generiranje izvještaja i statističkih podataka o prijavama.
- Upravljanje korisničkim ulogama i pristupnim ovlastima.

### Moguće nadogradnje projektnog zadatka

Kako bi se povećala funkcionalnost i korisnička vrijednost sustava, planirane su moguće nadogradnje koje se mogu implementirati u kasnijim fazama razvoja. Moguće nadogradnje uključuju:

- Mobilnu aplikaciju koja bi korisnicima omogućila brzu prijavu kvarova i push obavijesti o promjenama statusa.
- Integrirani chat sustav za direktnu komunikaciju između korisnika
- Sustav za zakazivanje termina popravaka
- Automatsko generiranje troška na temelju prijava i prosječnih cijena usluga.
- Sustav preporuka majstora temeljen na ocjenama korisnika i lokaciji.
- Integraciju s vanjskim servisima poput servisa za izdavanje računa.

## Funkcionalni zahtjevi

ID zahtjeva	Opis	Prioritet	Izvor	Kriteriji prihvaćanja
F-001	Sustav mora omogućiti registraciju triju tipova korisnika: stana, majstora i predstavnika suvlasnika.	Visok	Zahtjev dionika	Korisnik može odabrati tip profila pri registraciji i sustav spremi ulogu.
F-002	Sustav omogućuje prijavu registriranih korisnika putem e-maila i lozinke.	Visok	Dokument zahtjeva	Korisnik s valjanom e-mail adresom i lozinkom može se uspješno prijaviti.
F-003	Administrator ima mogućnost upravljanja korisničkim računima (aktivacija, deaktivacija).	Visok	Dokument zahtjeva	Administrator može promijeniti status korisnika ili dodijeliti novu ulogu.
F-004	Svaki korisnik može pregledati i uređivati vlastite profilne podatke (ime, prezime, kontakt, opis).	Visok	Zahtjev dionika	Korisnik može spremiti izmjene i one su vidljive pri sljedećoj prijavi.
F-005	Majstori imaju javni profil s osnovnim informacijama (naziv obrta ili ime i prezime, kontakt podaci, opis specijalizacije, prosječna ocjena).	Srednji	Povratne informacije korisnika	Profil majstora vidljiv je svima i prikazuje sve tražene podatke.

F-006	Predstavnici suvlasnika imaju administrativni profil s dodatnim mogućnostima (dodjeljivanje kvarova, pregled statistike, izvještaji).	Visok	Dokument zahtjeva	Predstavnik može pristupiti administrativnim alatima u svom profilu.
F-007	Stanar može prijaviti novi kvar unosom naziva, opisa, fotografije (opcionalno) kategorije kvara i lokacije unutar zgrade.	Visok	Zahtjev dionika	Korisnik može unijeti sve potrebne podatke i sustav kreira prijavu s jedinstvenim ID-om.
F-008	Svaka prijava kvara ima status ("otvoreno", u "tijeku", "rješeno") koji se ažurira tijekom procesa rješavanja.	Visok	Postojeći sustav	Nova prijava dobiva status "otvoreno", a majstor može promijeniti status u "u tijeku" ili "rješeno".
F-009	Predstavnik suvlasnika može dodijeliti prijavljeni kvar iz svoje zgrade određenom majstoru.	Visok	Dokument zahtjeva	Predstavnik može vidjeti kvarove samo iz zgrade koju predstavlja te odabrati majstora za kvar
F-010	Majstor može ažurirati status prijave i dodavati napomene vidljive stanaru i predstavniku.	Visok	Povratne informacije korisnika	Promjene statusa i napomene prikazuju se u stvarnom vremenu prijavi.
F-011	Sustav mora slati e-mail obavijest stanaru kad je kvar riješen.	Niski	Zahtjev dionika	Nakon promjene statusa stanar prima e-mail obavijest.
F-012	Stanar može ocijeniti majstora ocjenom (1-5) i opcionalno ostaviti komentar nakon što je kvar riješen.	Niski	Povratne informacije korisnika	Nakon promjene statusa u "rješeno", stanaru se omogućuje ocjenjivanje majstora.
F-013	Sustav mora majstorima omogućiti plaćanje godišnje članarine putem kreditne kartice.	Visok	Tehnički zahtjev	Plaćanje se može izvršiti putem integriranog servisa (npr. Stripe).
F-014	Administrator mora moći postaviti ili izmijeniti cijenu članarine.	Visok	Dokument zahtjeva	Promjena cijene članarine automatski se odražava u sustavu.
F-015	Sustav mora ograničiti pristup funkcionalnostima majstora ako članarina nije plaćena.	Visok	Zahtjev dionika	Majstoru s neaktivnim članstvom onemogućen je cijelovit pristup funkcionalnostima.
F-016	Predstavnik suvlasnika mora imati pregled statistike kvarova (broj prijava po kategorijama, prosječno vrijeme rješavanja, status).	Visok	Dokument zahtjeva	Statistički prikaz se može filtrirati prema kategoriji i razdoblju.
F-017	Sustav mora omogućiti generiranje mjesecnog PDF izvještaja svih kvarova putem integracije s vanjskim servisom za generiranje PDF-ova koji može preuzeti predstavnik suvlasnika.	Srednji	Tehnički zahtjev	Izvještaj sadrži sve kvarove za odabrani mjesec sa statusima, kategorijama i vremenom rješavanja.
F-018	Predstavnik suvlasnika može slati pozivnice putem e-maila stanarima za	Srednje	Zahtjev dionika	Predstavnik unosi e-mail adresu stanara, a sustav šalje

registraciju u sustav.

pozivnicu s linkom za registraciju.

## Ostali zahtjevi

### 1. Zahtjevi performansi

ID zahtjeva	Opis	Prioritet
NF-1.1	Vrijeme učitavanja početne stranice ne smije prelaziti 3 sekunde na standardnoj internetskoj vezi (10 Mbps).	Visok
NF-1.2	Vrijeme odziva sustava na korisničke akcije (klikovi, unos podataka) ne smije biti duže od 1 sekunde.	Visok
NF-1.3	Sustav mora podržavati istovremeni rad najmanje 100 aktivnih korisnika bez degradacije performansi.	Srednji
NF-1.4	Optimizacija slika i medijskih sadržaja mora biti implementirana (lazy loading, kompresija).	Srednji
NF-1.5	Aplikacija mora koristiti cache mehanizme za često korištene podatke.	Srednji

### 2. Sigurnosni zahtjevi

ID zahtjeva	Opis	Prioritet
NF-2.1	Komunikacija između klijenta i poslužitelja mora biti zaštićena HTTPS protokolom.	Visok
NF-2.2	Sustav mora koristiti sigurne metode autentifikacije (OAuth 2.0, JWT tokeni).	Visok
NF-2.3	Lozinke korisnika moraju biti pohranjene u kriptiranom obliku koristeći bcrypt ili sličan algoritam.	Visok
NF-2.4	Sustav mora implementirati zaštitu od uobičajenih sigurnosnih prijetnji (SQL injection, XSS, CSRF).	Visok
NF-2.5	Korisničke sesije moraju imati automatsko istjecanje nakon 24 sata neaktivnosti.	Srednji

### 3. Zahtjevi pouzdanosti

ID zahtjeva	Opis	Prioritet
NF-3.1	Sustav mora imati dostupnost od najmanje 99% mjesечно (downtime manji od 7.2 sata mjesечно).	Visok
NF-3.2	Sustav mora imati mehanizme za automatsko oporavak od pada servisa.	Srednji
NF-3.3	Sustav mora imati mehanizme za detekciju i obradu grešaka bez pada cijelog sustava.	Visok
NF-3.4	U slučaju greške, sustav mora omogućiti vraćanje na prethodno stabilno stanje.	Srednji

### 4. Zahtjevi skalabilnosti

ID zahtjeva	Opis	Prioritet
NF-4.1	Arhitektura sustava mora omogućiti jednostavno povećanje kapaciteta dodavanjem resursa.	Srednji
NF-4.2	Baza podataka mora biti dizajnirana tako da podržava rast broja korisnika do 10,000.	Srednji
NF-4.3	Sustav mora koristiti cloud infrastrukturu koja omogućuje elastično skaliranje (Vercel, Supabase).	Srednji

## 5. Zahtjevi održivosti

ID zahtjeva	Opis	Prioritet
NF-5.1	Sustav treba biti oblikovan tako da omogućuje jednostavno održavanje.	Visok
NF-5.2	Sustav treba imati dovoljnu dokumentaciju.	Visok
NF-5.3	Kod sustava treba biti dokumentiran prema JavaScript best practices standardima.	Visok
NF-5.4	Sustav treba biti opisan putem dokumenta oblikovanja (SRS).	Visok
NF-5.5	Sustav treba biti popraćen "Priručnikom za rad" koji opisuje pravilnu upotrebu sustava.	Visok
NF-5.6	Sustav treba imati "Plan implementacije" za pravilno postavljanje sustava.	Visok
NF-5.7	Baza podataka mora biti strukturirana tako da omogućuje jednostavno dodavanje podataka.	Visok
NF-5.8	Kod mora biti modularan i organiziran prema komponentnoj arhitekturi.	Visok

## 6. Zahtjevi prenosivosti

ID zahtjeva	Opis	Prioritet
NF-6.1	Kod mora biti neovisan o specifičnim operativnim sustavima za razvoj.	Srednji

## 7. Zahtjevi upotrebljivosti

ID zahtjeva	Opis	Prioritet
NF-7.1	Korisničko sučelje mora biti intuitivno i jednostavno za korištenje.	Visok
NF-7.2	Sustav mora podržavati hrvatski jezik kao primarni jezik sučelja.	Visok
NF-7.3	Korisničko sučelje mora biti responzivno i prilagođeno za različite veličine ekrana (desktop, tablet, mobitel).	Visok
NF-7.4	Sustav mora pružiti jasne povratne informacije korisniku o uspješnosti ili neuspješnosti akcija.	Visok
NF-7.5	Forme za unos podataka moraju imati jasnu validaciju s razumljivim porukama o greškama.	Visok

## 8. Zahtjevi dostupnosti

ID zahtjeva	Opis	Prioritet
NF-8.1	Web aplikacija mora biti dostupna 24/7 s minimalnim prekidima rada.	Visok
NF-8.2	Sustav mora imati monitoring i alerte za detekciju problema u realnom vremenu.	Srednji

## 9. Zahtjevi interoperabilnosti

ID zahtjeva	Opis	Prioritet
NF-9.1	Web aplikacija mora biti kompatibilna s najnovijim verzijama desktop preglednika (Chrome, Firefox, Safari, Edge).	Visok
NF-9.2	Web aplikacija mora biti kompatibilna s mobilnim preglednicima (Safari na iOS-u, Chrome na Android-u).	Visok
NF-9.3	Aplikacija mora podržavati preglednike koji nisu stariji od 2 glavne verzije.	Srednji
NF-9.4	Sustav mora raditi na operativnim sustavima Windows 10+, macOS 11+, i modernim Linux distribucijama.	Srednji

## 10. Zahtjevi razvoja i implementacije

ID zahtjeva	Opis	Prioritet
NF-10.1	Razvojni tim mora koristiti sustav za kontrolu verzija (Git).	Visok
NF-10.2	Kod mora proći code review proces prije integracije u glavnu granu (master).	Visok
NF-10.3	Environment varijable moraju biti odvojene od koda i ne smiju biti pohranjene u repozitoriju.	Visok

## Dionici

Dionik	Opis uloge / interesa
Stanar	Krajnji korisnik sustava koji prijavljuje kvarove i prati njihovo rješavanje.
Majstor	Vanjski suradnik koji prima prijave kvarova, obavlja popravke i plaća članarinu.
Predstavnik suvlasnika	Osoba koja upravlja prijavama kvarova, dodjeljuje ih majstorima te izrađuje izvještaje.
Administrator sustava	Osoba odgovorna za nadzor, održavanje sustava i upravljanje korisnicima.
Naručitelj sustava	Organizacija koja je naručila izradu sustava HomeFix.
Razvojni tim	Tim zadužen za izradu, testiranje i održavanje sustava HomeFix.
Vanjski servisi (Stripe, OAuth 2.0, PDF servis API)	Integrirani sustavi koji omogućuju plaćanje, autentifikaciju i generiranje izvještaja.


A-1: Stanar (inicijator) može:

- Registrirati korisnički račun i prijaviti se u sustav. (F-001, F-002)
- Prijaviti novi kvar uz opis, fotografiju i kategoriju. (F-007)
- Primiti obavijest o završenom popravku. (F-011)
- Ocijeniti majstora i opcionalno ostaviti komentar. (F-012)

A-2: Majstor (sudionik i inicijator) može:

- Registrirati se i prijaviti kao majstor. (F-001, F-002)
- Pregledati i ažurirati dodijeljene kvarove. (F-010)
- Označiti kvar kao "riješen". (F-008)
- Platiti članarinu putem Stripe servisa i aktivirati članstvo. (F-013, F-015)

A-3: Predstavnik suvlasnika (inicijator) može:

- Registrirati se i prijaviti u sustav. (F-001, F-002)
- Dodijeliti kvarove odgovarajućim majstorima. (F-009)
- Preuzeti i pregledati izvještaje o popravcima. (F-016, F-017)
- Emailom slati pozivnice stanaima zgrade za registraciju. (F-018)

A-4: Administrator (inicijator i nadzornik) može:

- Prijaviti se u sustav s administratorskim ovlastima. (F-002)
- Pregledati i upravljati korisnicima (izmjene, promjene uloga, deaktivacija). (F-003)
- Ažurirati cijenu članarine. (F-014)

A-5: Naručitelj sustava (vanjski dionik) ima interes:

- Da sustav ispravno funkcionira i smanjuje troškove održavanja zgrada.
- Da izvještaji budu dostupni predstavnicima suvlasnika. (F-016)

A-6: Razvojni tim (vanjski dionik) je odgovoran za:

- Implementaciju funkcionalnih zahtjeva (F-001 – F-018).

- Održavanje baze podataka, servera i integracija.

A-7: Vanjski servisi (sudionici) sudjeluju u:

- Autentifikaciji korisnika putem OAuth 2.0. (F-001)
  - Obradi plaćanja putem Stripe servisa. (F-013, F-015)
  - Generiranju PDF izvještaja putem PDF servisa API. (F-017)
- 

## **Obrasci uporabe**

### **Dijagrami obrazaca uporabe**

#### **1. Visokorazinski dijagram obrazaca uporabe cijelog sustava**

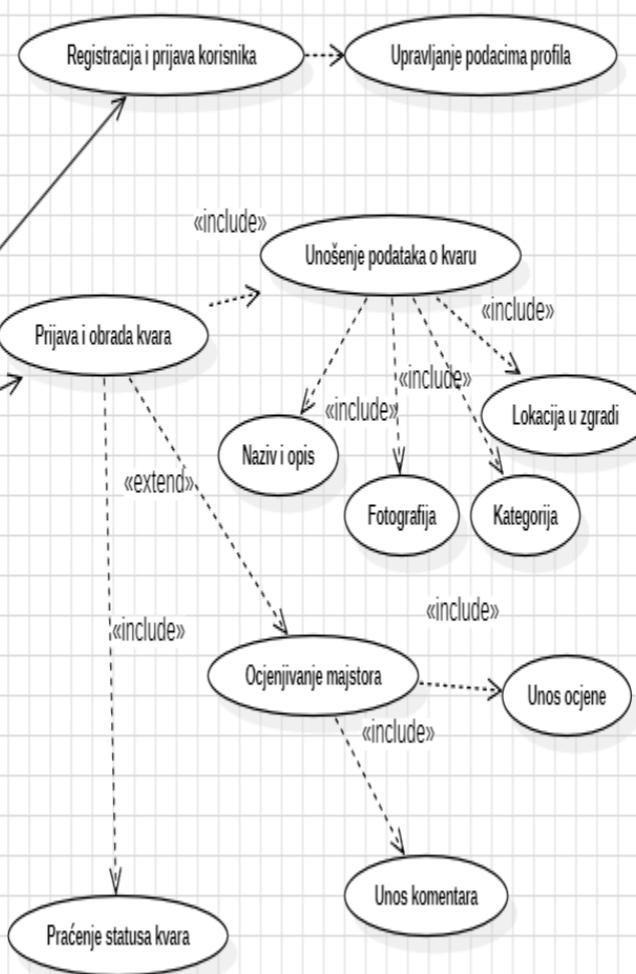


## 2. dijagram obrazaca uporabe za ključne značajke

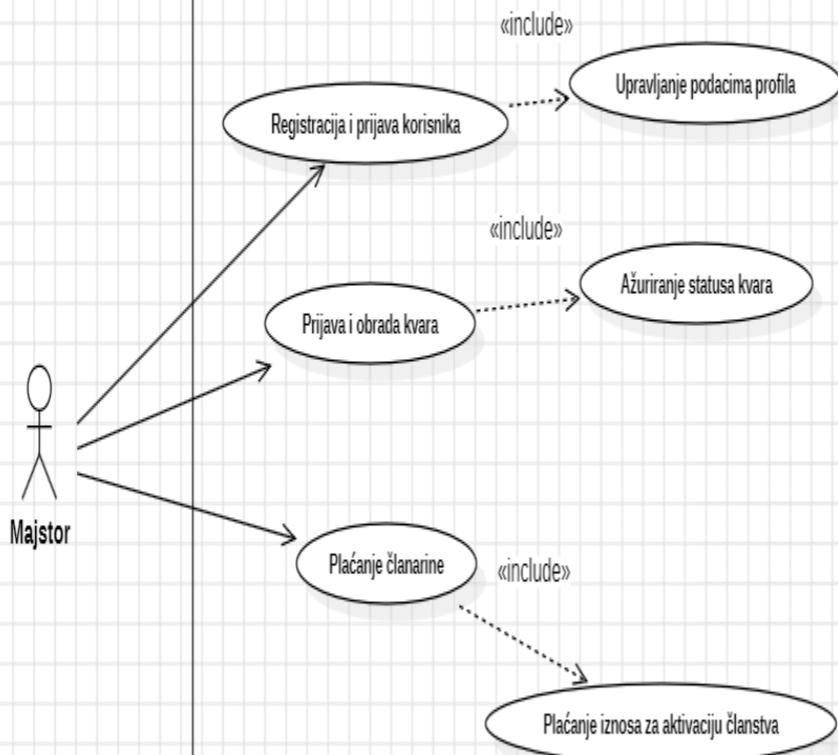
\*\*Ključne značajke:\*\* prijava i registracija korisnika, prijava i obrada kvara, plaćanje članarine, preuzimanje i generiranje izvještaja, upravljanje korisnicima

# HomeFix

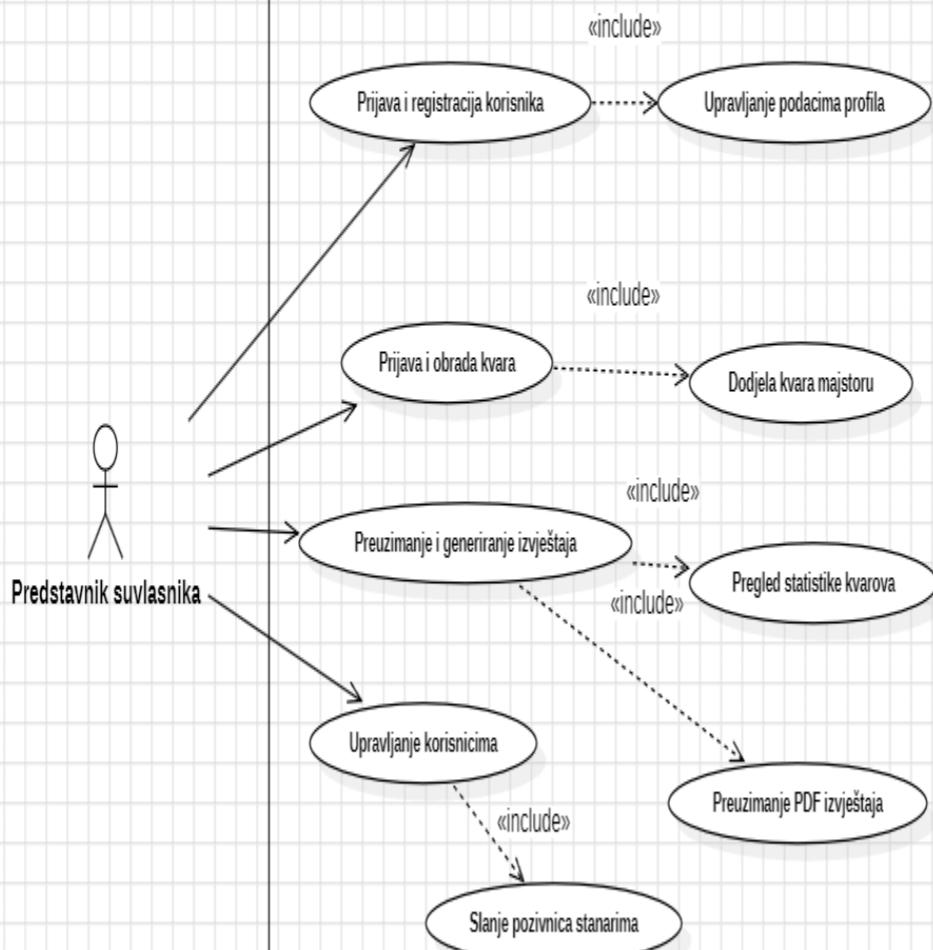
Stanar

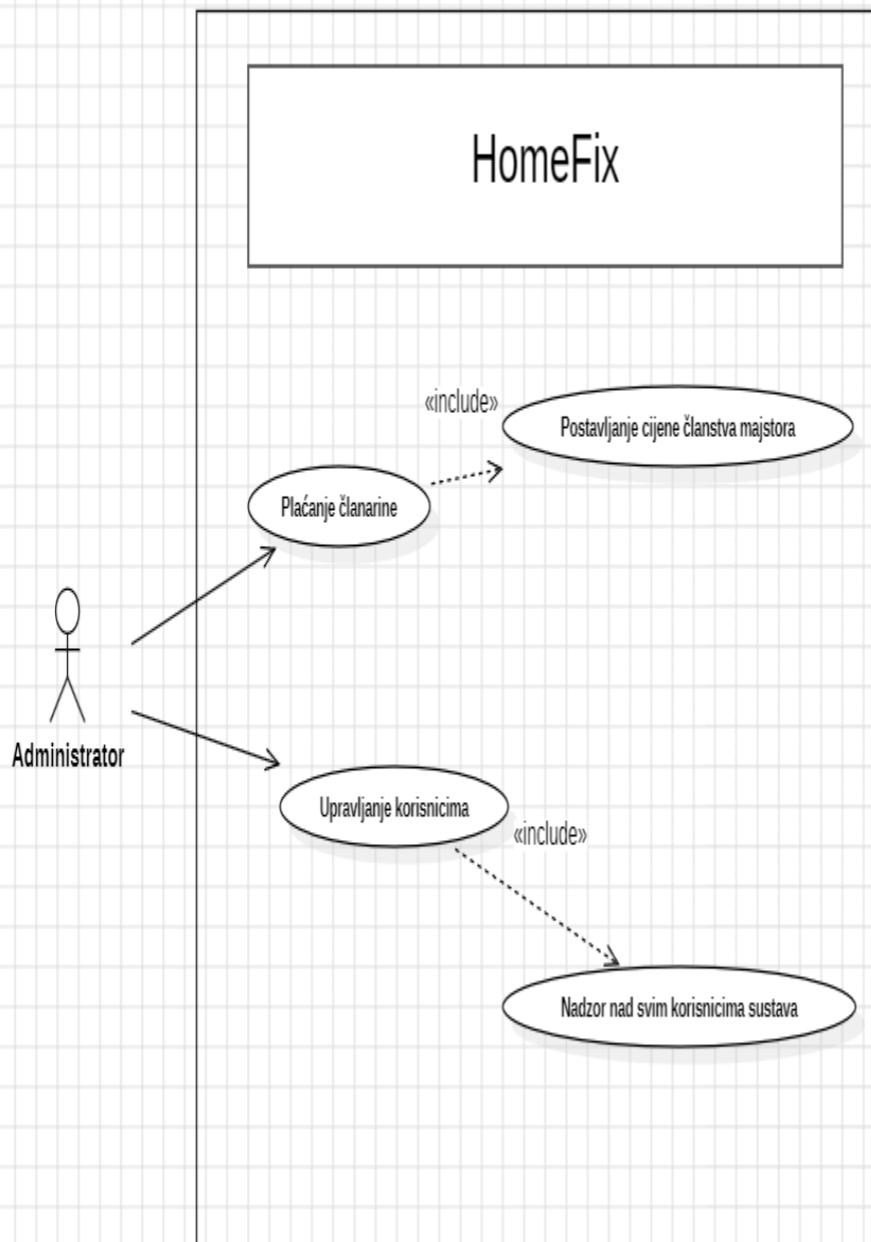


# HomeFix

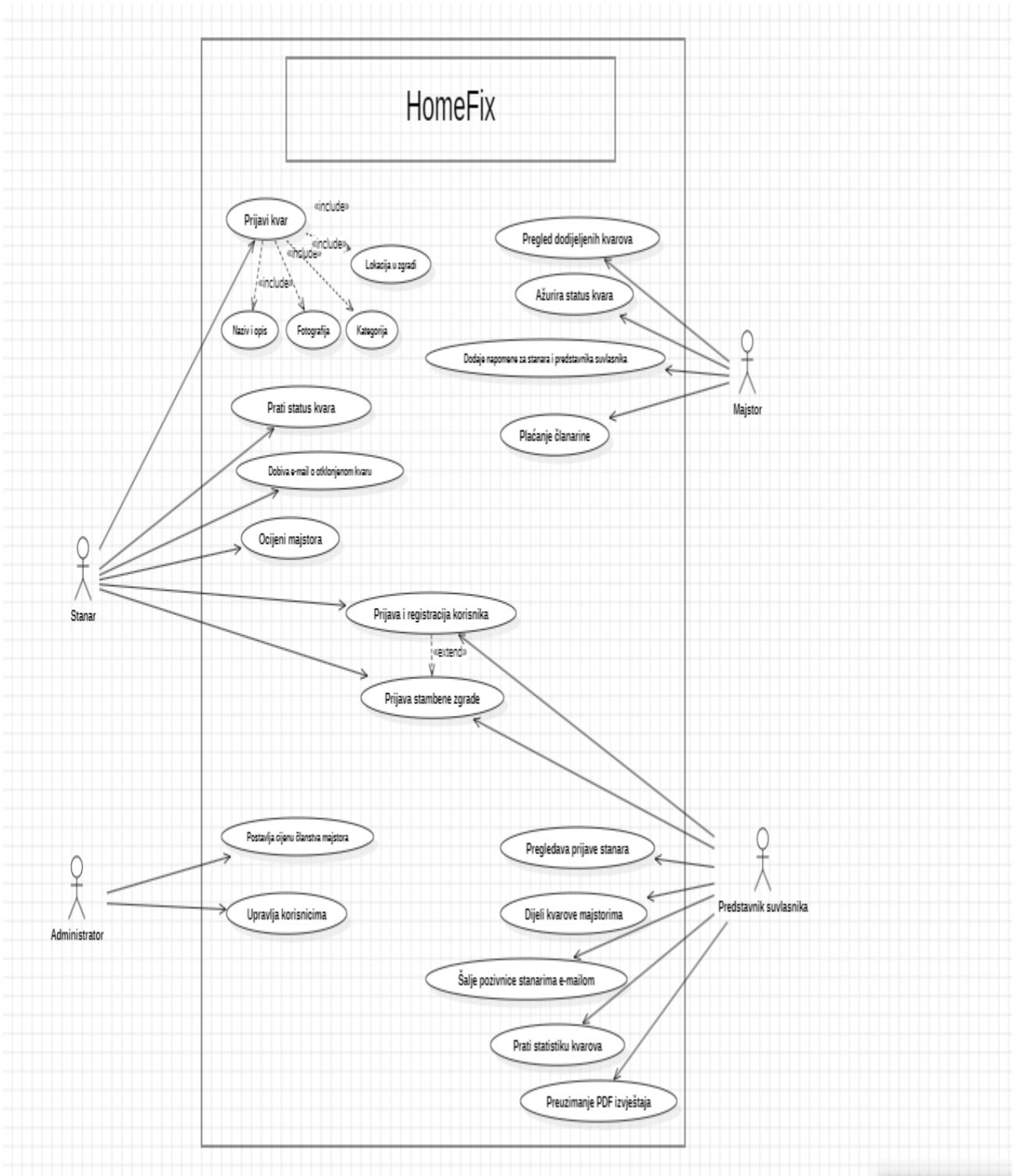


# HomeFix

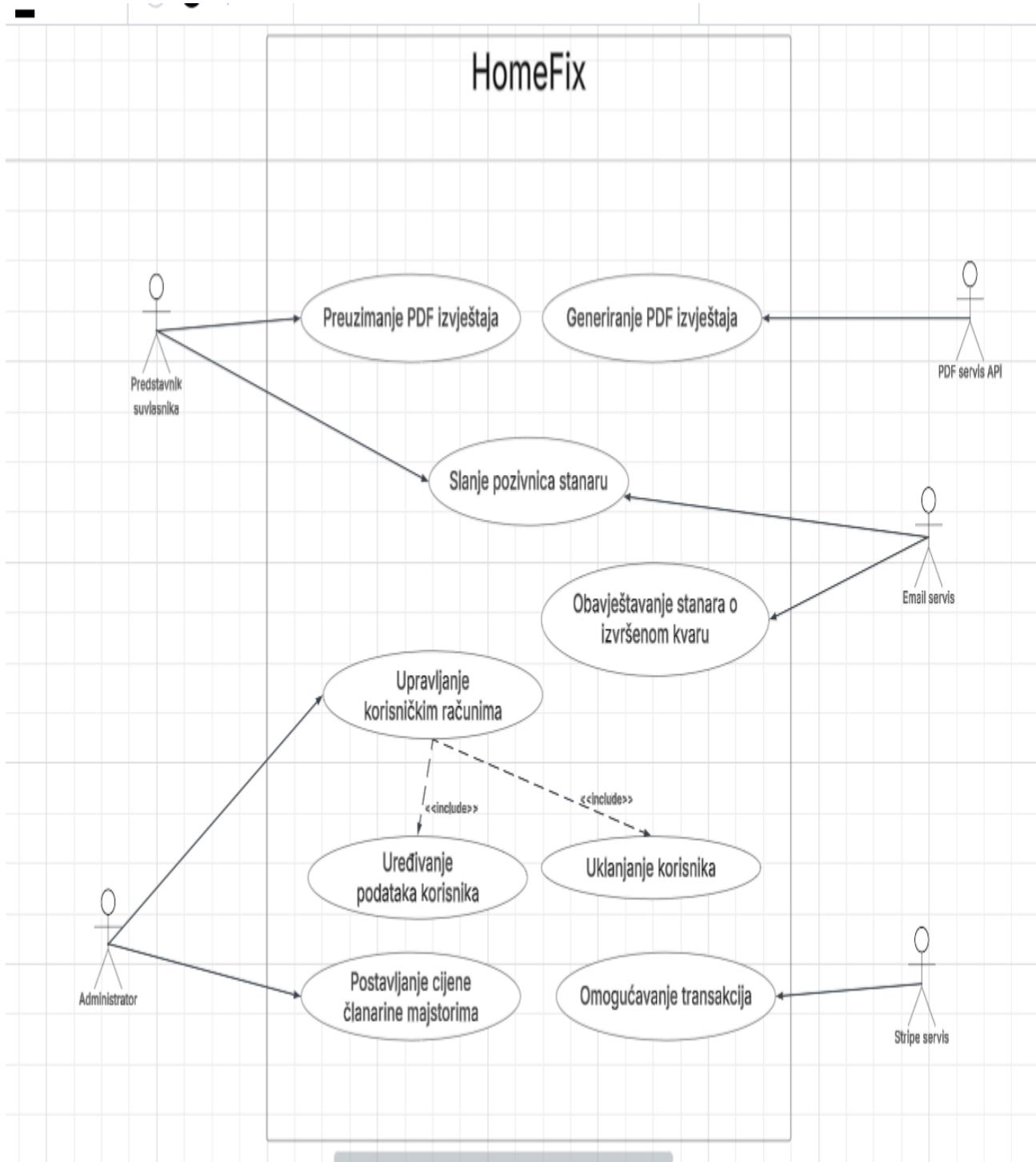




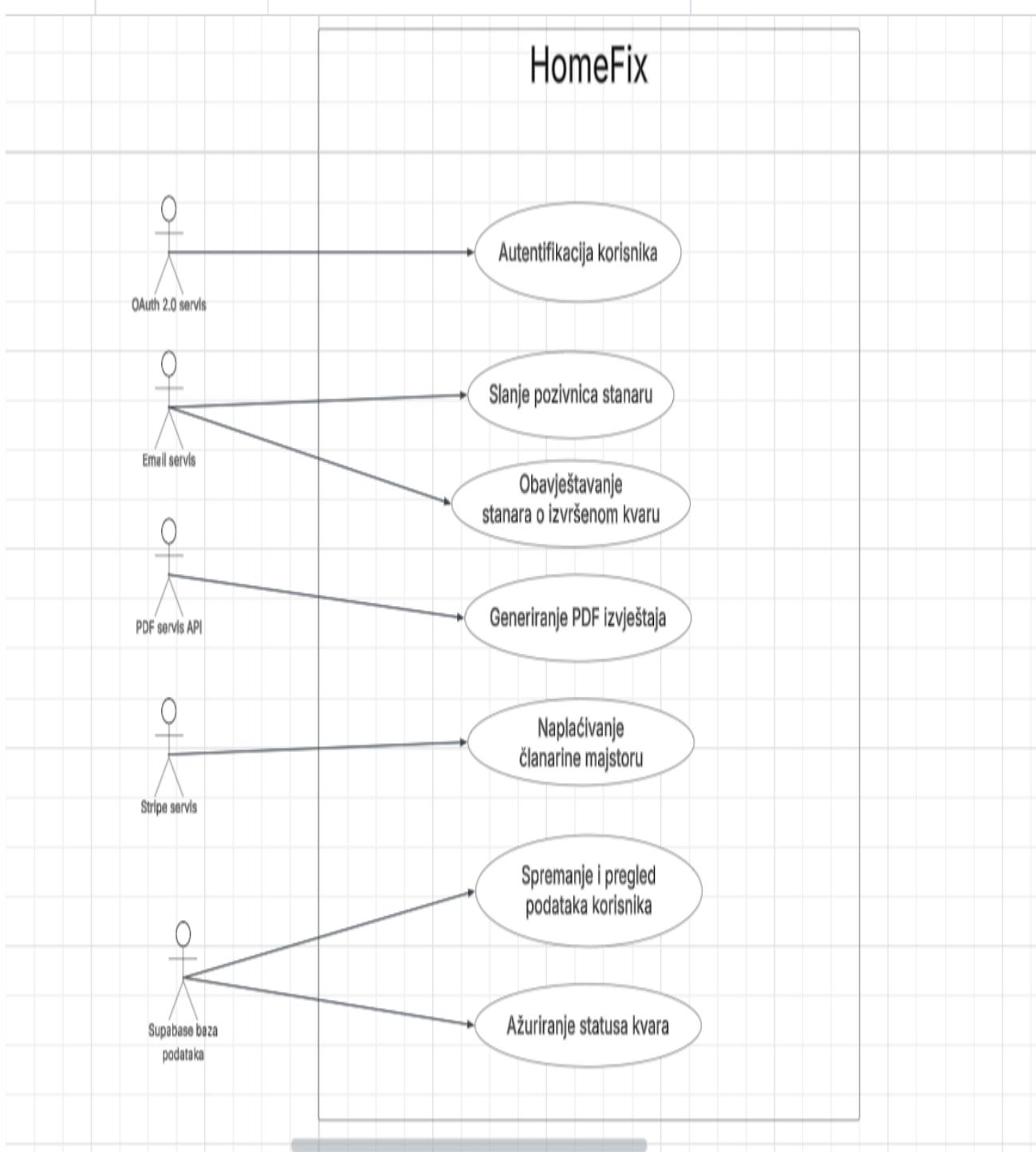
**3. dijagram obrazaca uporabe za korisničke uloge**



**4. dijagram obrazaca uporabe za osnovne poslovne procese**



**5. dijagram obrazaca uporabe za kritične sustave i integracije**



## Opis obrazaca uporabe

### UC-001 – Registracija korisnika

- Glavni sudionik: Novi korisnik (stanar, majstor, predstavnik suvlasnika)
- Cilj: Omogućiti korisniku stvaranje računa u sustavu s odgovarajućom ulogom.
- Sudionici: Korisnik, sustav, vanjski OAuth 2.0 servis
- Preduvjet: Korisnik nije registriran u sustavu.

### Opis osnovnog tijeka:

- Korisnik otvara stranicu za registraciju. (F-001)
- Korisnik unosi tražene podatke (ime, e-mail, lozinku).
- Korisnik odabire tip profila (stanar, majstor, predstavnik suvlasnika). (F-001)

- Stanar i predstavnik suvlasnika naznačuju adresu (stambenu zgradu)
- Sustav šalje podatke OAuth 2.0 servisu radi provjere autentičnosti.
- Sustav kreira korisnički račun i spremi ulogu. (F-001)
- Sustav prikazuje poruku o uspješnoj registraciji.

**Moguća odstupanja:**

- (1a) Korisnik već postoji → Sustav prikazuje poruku "Korisnik već registriran" i predlaže prijavu.
- (1b) Autentifikacija ne uspije → Sustav vraća korisnika na formu s porukom o grešci.

**UC-002 – Prijava korisnika**

- Glavni sudionik: Korisnik (stanar, majstor, predstavnik ili administrator)
- Cilj: Omogućiti korisniku pristup sustavu HomeFix s vlastitim korisničkim računom kako bi mogao koristiti funkcionalnosti sustava.
- Sudionici: sustav, baza korisnika
- Preduvjet: Korisnik mora imati registriran korisnički račun u sustavu, sustav mora biti povezan s bazom podataka korisnika.

**Opis osnovnog tijeka:**

- Korisnik otvara početnu stranicu sustava HomeFix.
- Korisnik odabire opciju "Prijava".
- Sustav prikazuje formu za unos korisničkog imena i lozinke. (F-002)
- Korisnik unosi svoje podatke i potvrđuje unos. (F-002)
- Sustav provjerava podatke u bazi podataka.
- Ako su podatci točni, sustav prepoznaje tip korisnika (stanar, majstor, predstavnik, administrator) i preusmjerava ga na odgovarajuće sučelje.
- Korisnik uspješno pristupa svom profilu i funkcijama sustava.

**Moguća odstupanja:**

- (2a) Neispravni podaci za prijavu → Sustav prikazuje poruku "Pogrešno korisničko ime ili lozinka" i omogućava ponovni pokušaj.
- (2b) Korisnik nije registriran → Sustav nudi opciju "Registriraj se" i preusmjerava korisnika na obrazac za registraciju.
- (2c) Greška u komunikaciji s bazom podataka → Sustav prikazuje poruku o tehničkoj pogrešci i predlaže ponovno pokušati kasnije.

**UC-003 – Prijava i obrada kvara**

- Glavni sudionik: Stanar
- Cilj: Omogućiti prijavu, praćenje i rješavanje kvara u zgradama.
- Sudionici: Stanar, predstavnik suvlasnika, majstor, sustav, e-mail servis
- Preduvjet: Korisnici su registrirani i prijavljeni u sustav.

**Opis osnovnog tijeka:**

- Stanar odabire opciju "Prijavi kvar". (F-007)
- Unosi naziv, opis, fotografiju i kategoriju kvara.
- Sustav kreira novu prijavu i dodjeljuje joj status "otvoreno". (F-008)
- Predstavnik pregleda prijavu i dodjeljuje je majstoru. (F-009)
- Majstor pregledava dodijeljeni kvar, mijenja status u "u tijeku" te unosi napomene. (F-010)
- Nakon popravka, majstor označava kvar kao "riješen". (F-008)
- Sustav automatski šalje e-mail obavijest stanaru o rješenju kvara. (F-011)
- Stanar može ocijeniti majstora i ostaviti komentar. (F-012)

**Moguća odstupanja:**

- (3a) Stanar ne unese obavezna polja → Sustav prikazuje poruku o grešci i ne spremi prijavu.
- (3b) Nema dostupnih majstora → Predstavnik može označiti prijavu kao "čeka majstora" dok se ne pronađe.
- (3c) Majstor slučajno označi kvar kao "riješen" prije završetka → Predstavnik može vratiti status u "u tijeku".

## UC-004 – Plaćanje članarine majstora

- Glavni sudionik: Majstor
- Cilj: Omogućiti plaćanje godišnje članarine i pristup sustavu.
- Sudionici: Majstor, sustav, Stripe servis, administrator
- Preduvjet: Majstor ima registriran račun i valjanu kreditnu karticu.

### Opis osnovnog tijeka:

- Majstor otvara stranicu za članarinu. (F-013)
- Sustav prikazuje trenutnu cijenu članarine koju je postavio administrator. (F-014)
- Majstor unosi podatke o kartici i potvrđuje plaćanje. (F-013)
- Sustav šalje zahtjev Stripe servisu.
- Nakon potvrde uplate, sustav aktivira članstvo majstora. (F-015)

### Moguća odstupanja:

- (4a) Podaci o kartici nisu valjani → sustav obavještava korisnika i traži ponovni unos.
- (4b) Stripe servis ne odgovara → sustav bilježi neuspjeh i omogućuje ponovno pokušavanje kasnije.
- (4c) Administrator promijeni cijenu tijekom plaćanja → sustav obavještava majstora i ažurira iznos prije potvrde.

## UC-005 – Generiranje i preuzimanje izvještaja

- Glavni sudionik: Predstavnik suvlasnika
- Cilj: Dobiti mjesecni pregled svih kvarova u zgradu u PDF obliku.
- Sudionici: Predstavnik, sustav, PDF generator servis
- Preduvjet: Predstavnik ima pristup svom profilu i dodijeljenim prijavama.

### Opis osnovnog tijeka:

- Predstavnik otvara izbornik "Izvještaji". (F-016)
- Odabire mjesec i godinu za izvještaj. (F-017)
- Sustav prikuplja sve podatke o kvarovima i statusima. (F-016)
- Sustav poziva vanjski PDF servis i generira izvještaj. (F-017)
- PDF dokument se prikazuje s opcijom preuzimanja. (F-017)

### Moguća odstupanja:

- (5a) Vanjski PDF servis nedostupan → sustav prikazuje poruku "Izvještaj trenutno nije moguće generirati".
- (5b) Greška pri generiranju datoteke → sustav omogućuje ponovno generiranje.

## UC-006 – Upravljanje korisnicima

- Glavni sudionik: Administrator sustava
- Cilj: Omogućiti administratoru pregled, uređivanje, deaktivaciju i brisanje korisničkih računa te nadzor nad svim korisnicima sustava.
- Sudionici: Administrator, sustav, baza korisnika
- Preduvjet: Administrator mora biti prijavljen u sustav s ovlastima upravljanja korisnicima.

### Opis osnovnog tijeka:

- Administrator otvara administratorsko sučelje sustava.
- Sustav prikazuje popis svih korisnika s osnovnim informacijama (ime, uloga, status). (F-003)
- Administrator može pretražiti korisnika po imenu, e-mail adresi ili ulozi.
- Administrator odabire korisnika i otvara detaljan prikaz profila.
- Administrator može: izmijeniti korisničke podatke, promijeniti ulogu korisnika, deaktivirati ili obrisati korisnički račun (F-003)
- Sustav bilježi sve promjene.
- Sustav prikazuje potvrdu o uspješnom ažuriranju.

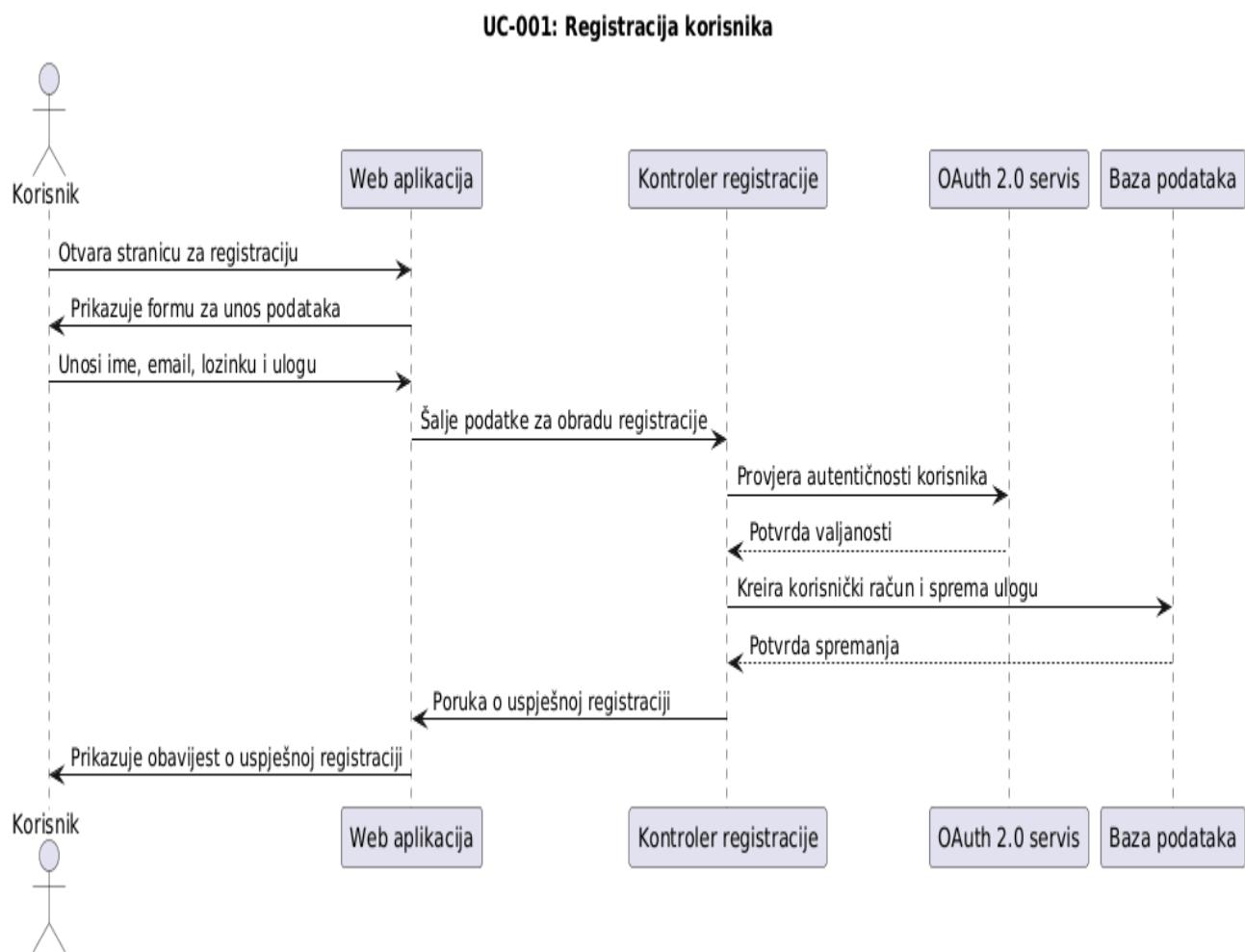
### Moguća odstupanja:

- (6a) Korisnik ne postoji u bazi → Sustav prikazuje poruku "Korisnik nije pronađen" i vraća administratora na popis korisnika.

- (6b) Administrator nema odgovarajuće ovlasti → Sustav onemogućava uređivanje i prikazuje poruku "Nemate ovlasti za ovu akciju".
- (6c) Greška pri spremanju promjena

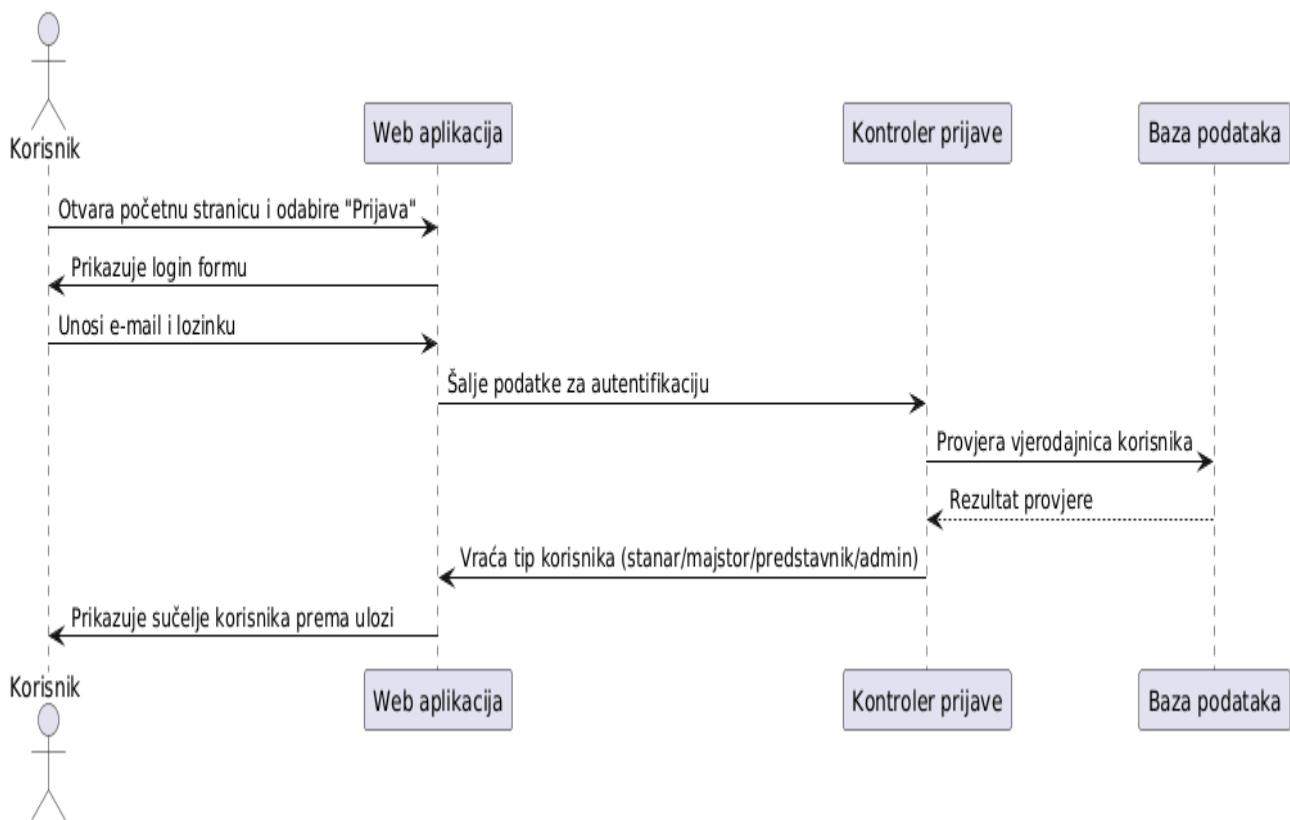
Sustav prikazuje upozorenje "Promjene nisu spremljene" i predlaže ponovno spremanje.

## Sekvencijski dijagrami



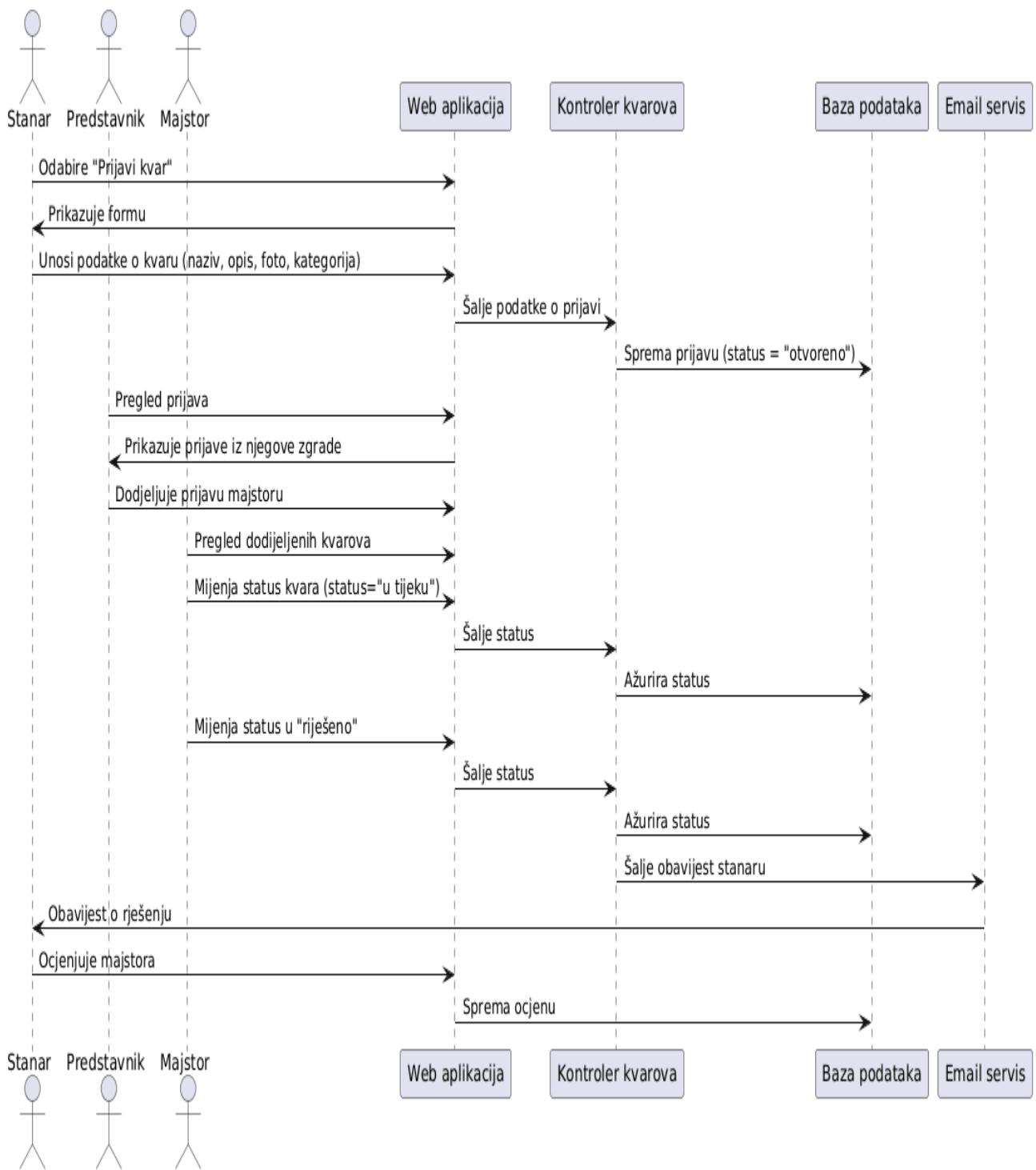
Sekvencijski dijagram prikazuje proces registracije novog korisnika u sustavu HomeFix. Korisnik otvara stranicu za registraciju i unosi osnovne podatke: ime, e-mail, lozinku i tip profila (stanar, majstor ili predstavnik suvlasnika). Sustav šalje podatke OAuth 2.0 servisu radi provjere autentičnosti korisnika. Nakon uspješne provjere, sustav kreira novi korisnički račun, sprema ulogu korisnika i prikazuje poruku o uspješnoj registraciji. Ako korisnik odabere ulogu stanara ili predstavnika, dodatno se pohranjuje i adresa zgrade.

### UC-002: Prijava korisnika



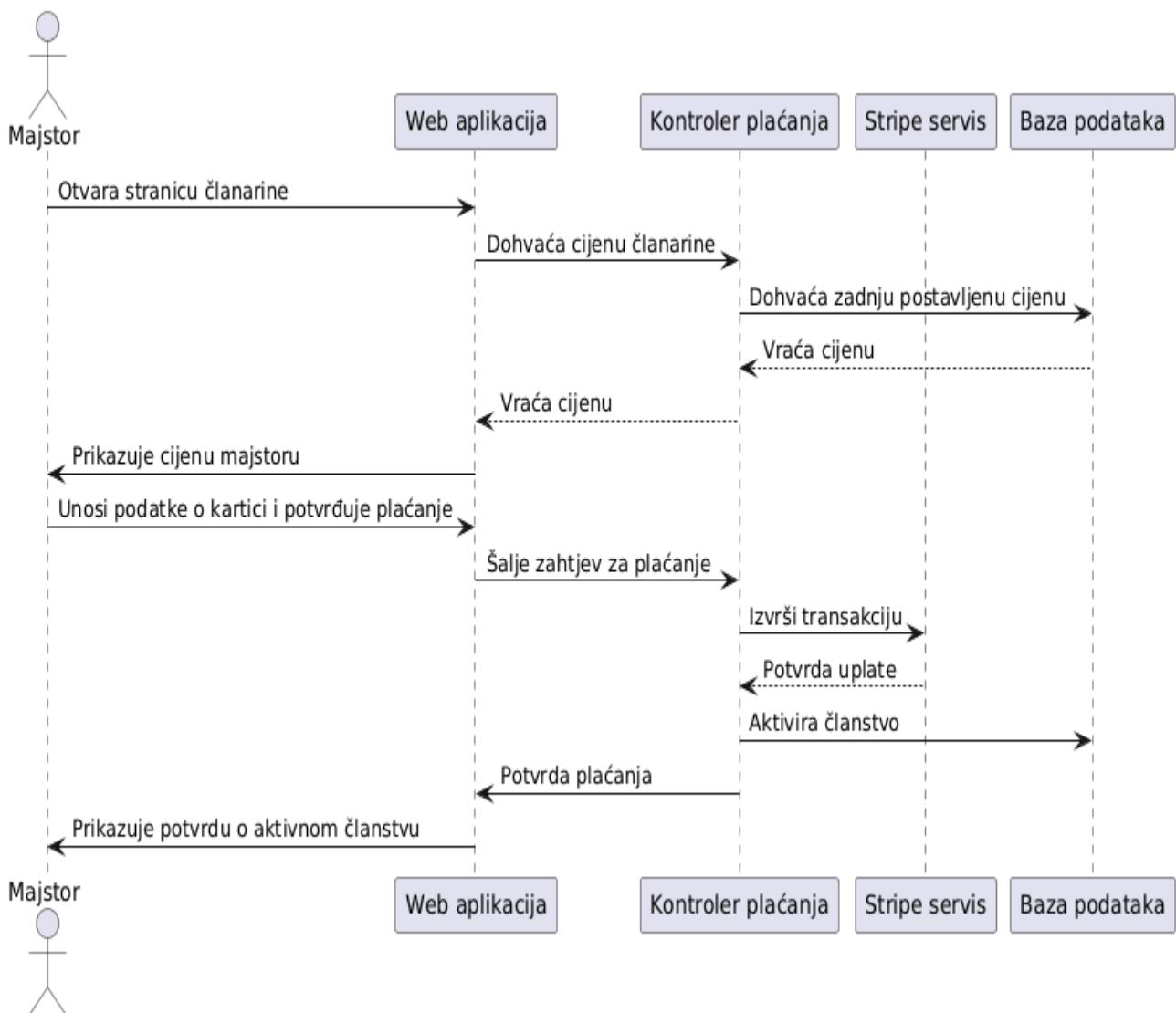
Ovaj dijagram prikazuje tijek prijave korisnika u sustav. Korisnik odabire opciju "Prijava" na početnoj stranici i unosi svoje korisničko ime i lozinku. Sustav validira unesene podatke prema zapisima u bazi. Ako su podaci točni, sustav određuje tip korisnika (stanar, majstor, predstavnik suvlasnika ili administrator) i preusmjerava ga na odgovarajuće korisničko sučelje. Ako su podaci pogrešni, sustav vraća poruku o neuspješnoj prijavi.

### UC-003: Prijava i obrada kvara



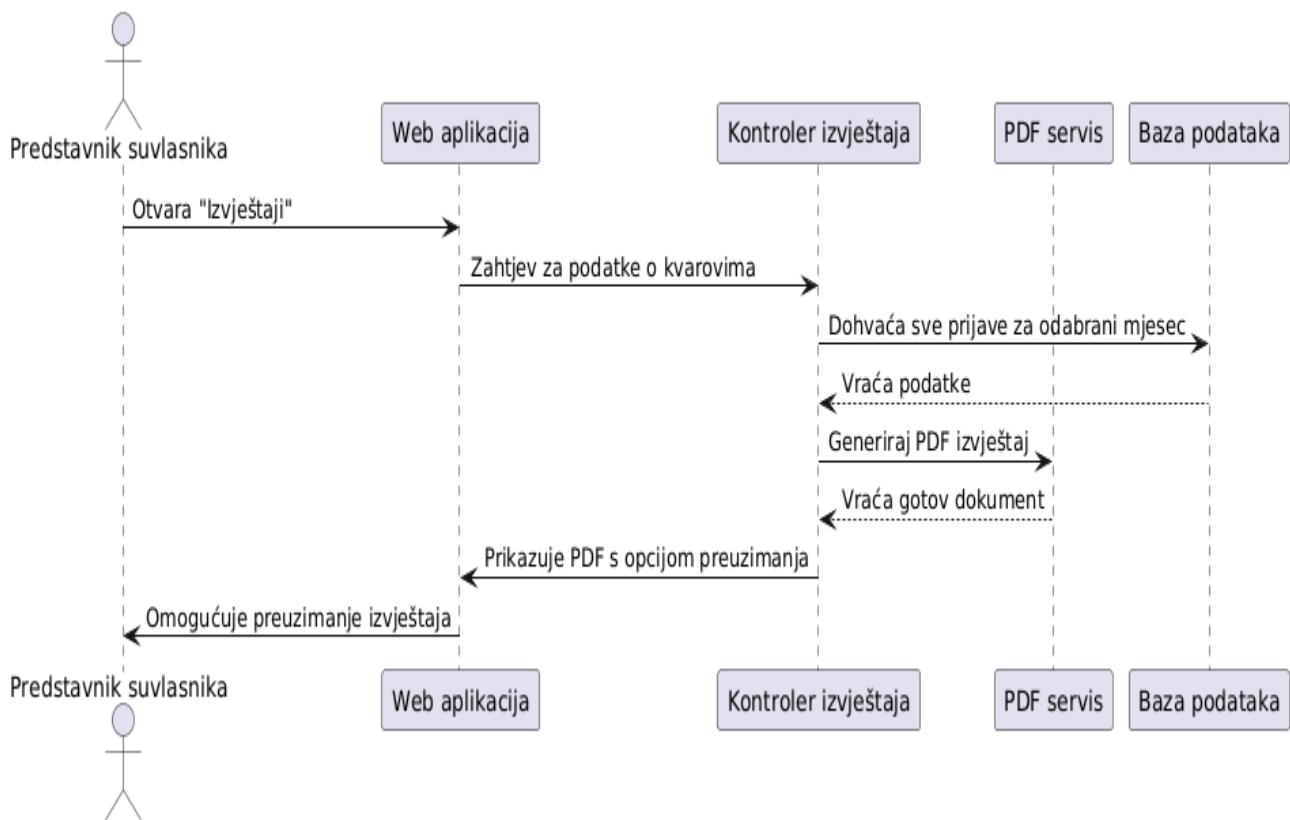
Sekvensijski dijagram prikazuje cijelokupan proces od prijave do rješenja kvara. Stanar unosi podatke o kvaru (naziv, opis, fotografiju, kategoriju) putem sučelja za prijavu kvara. Sustav kreira novi zapis s inicijalnim statusom "otvoreno". Predstavnik pregledava prijavu i dodjeljuje ju majstoru, čime se status mijenja u "dodijeljeno". Majstor preuzima zadatku, mijenja status u "u tijeku" te nakon popravka postavlja status "riješen". Sustav automatski obavještava stanara e-mailom o završetku popravka, a stanar potom može ocijeniti majstora i ostaviti komentar.

### UC-004: Plaćanje članarine majstora



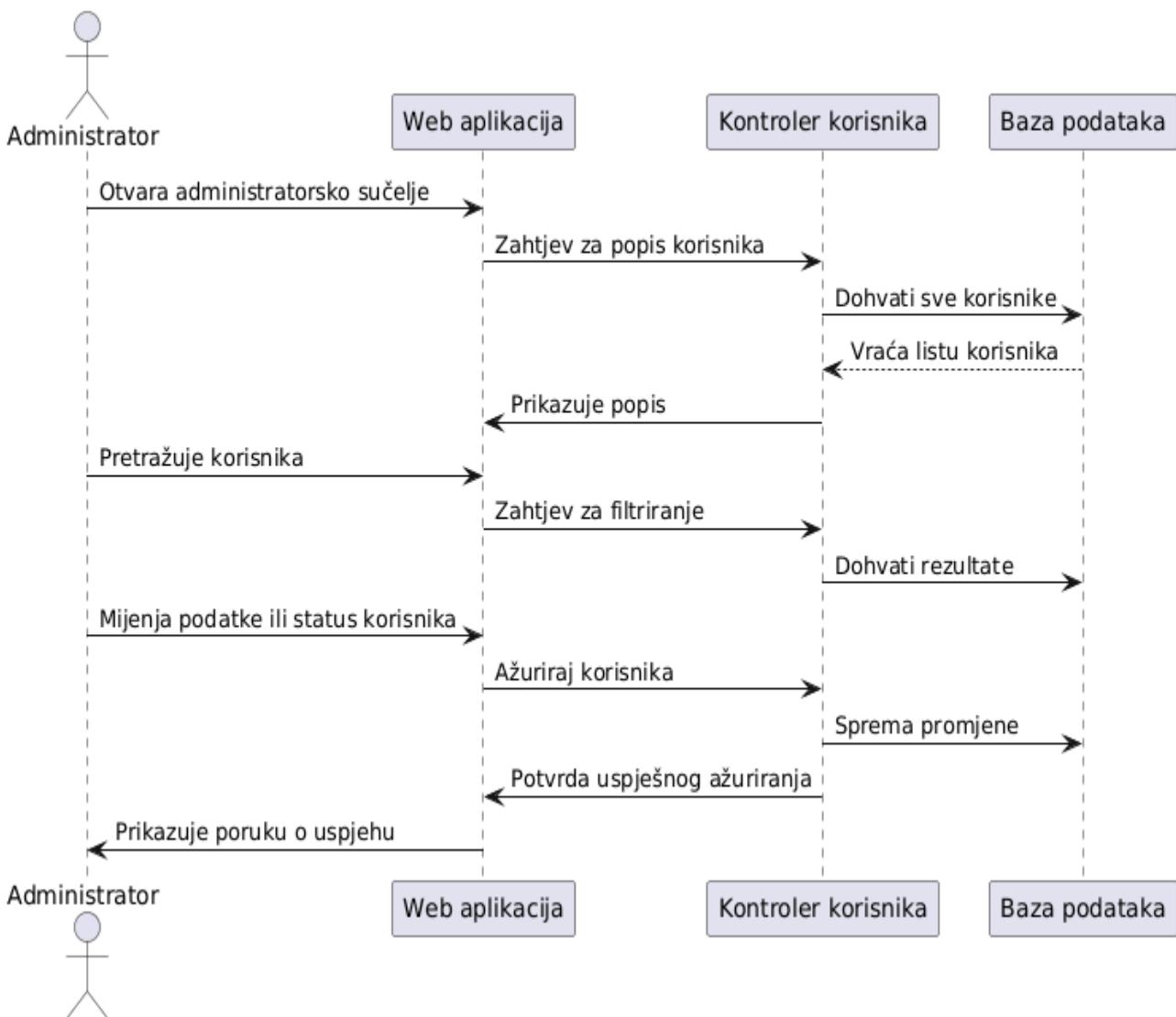
Ovaj dijagram prikazuje proces plaćanja članarine od strane majstora. Majstor otvara stranicu za plaćanje, gdje sustav dohvaća i prikazuje aktualnu cijenu članarine koju je definirao administrator. Majstor unosi podatke o kartici i potvrđuje uplatu. Sustav tada šalje zahtjev Stripe servisu za obradu plaćanja. Nakon što Stripe potvrđi transakciju, sustav ažurira status članarine majstora na "aktivna" i prikazuje poruku o uspješnom plaćanju.

### UC-005: Generiranje i preuzimanje izvještaja



Dijagram prikazuje način na koji predstavnik suvlasnika generira mjesечne izvještaje o kvarovima. Predstavnik u sučelju odabire mjesec i godinu za koje želi izvještaj. Sustav dohvaća sve relevantne zapise o kvarovima, njihovim statusima i majstorima koji su ih obrađivali. Podaci se zatim proslijeđuju vanjskom PDF servisu, koji generira dokument i vraća ga sustavu. Korisniku se prikazuje generirani PDF s mogućnošću preuzimanja.

### UC-006: Upravljanje korisnicima



Ovaj sekvencijski dijagram opisuje kako administrator upravlja korisnicima sustava. Administrator otvara administratorsko sučelje, a sustav dohvaća i prikazuje popis svih registriranih korisnika. Administrator može pretraživati korisnike, otvoriti njihov detaljni profil, mijenjati njihove podatke, uloge ili status (aktivno/deaktivirano). Sve promjene se bilježe u bazi i prikazuje se potvrda o uspješnom ažuriranju. Ovim procesom administrator osigurava pravilno funkcioniranje i nadzor nad korisničkim računima.

## Provjera uključenosti ključnih funkcionalnosti u obrascu uporabe

Oznaka obrasca uporabe	Naziv obrasca uporabe	Funkcijski zahtjevi
UC-001	Registracija korisnika	F-001
UC-002	Prijava korisnika	F-002
UC-003	Prijava i obrada kvara	F-007, F-008, F-009, F-010, F-011, F-012

UC-004	Plaćanje članarine majstora	F-013, F-014, F-015
UC-005	Generiranje i preuzimanje izvještaja	F-016, F-017, F-018
UC-006	Upravljanje korisnicima	F-003

# Arhitektura sustava

## Opis arhitekture

### Stil arhitekture:

Sustav HomeFix temelji se na klijent–poslužitelj arhitekturi sa slojevima frontenda (klijent) i backenda (poslužitelj).

Ovaj stil je odabran jer omogućava:

- Jasnu podjelu odgovornosti između korisničkog sučelja i poslovne logike,
- Jednostavnu nadogradnju pojedinih dijelova sustava bez utjecaja na druge,
- Fleksibilnost - Ovakav pristup olakšava skalabilnost, nadogradnje i potencijalnu zamjenu pojedinih komponenti sustava jer sustavi se razvijaju odvojeno

### Podsustavi:

Podsustav	Opis
Frontend (React)	Web sučelje izrađeno pomoću Next.js frameworka i React-om. React upravlja korisničkim sučeljem i interakcijama korisnika sa sustavom, pružajući dinamičan i responzivan prikaz. Komunikacija s backendom odvija se putem HTTP zahtjeva.
Backend (poslužiteljski dio)	služi kao poslužiteljska strana aplikacije, obrađuje poslovnu logiku, upravlja bazom podataka i izlaze podatke putem REST API-ja.

### Preslikavanje na radnu platformu:

Tijekom razvoja koristi se lokalni Next.js development server s Turbopack-om za brže učitavanje. Baza podataka je hostana na Supabase cloud platformi, što omogućava pristup podacima iz bilo kojeg okruženja. Za produkciju, aplikacija se može deplovati na platforme poput Vercel-a ili drugih Node.js hosting rješenja.

### Spremišta podataka:

Za pohranu podataka koristi se relacijska baza podataka zbog potrebe za konzistentnošću i jasno definiranih odnosa različitih korisnika.

Korištena tehnologija: PostgreSQL u Supabase-u

### Osnovne tablice:

- **profile** - za spremanje svih registriranih korisnika
- **representative** - za spremanje podataka predstavnika suvlasnika
- **contractor** - za spremanje podataka majstora
- **membership** - za spremanje podataka o članarini majstora
- **tenant** - za spremanje podataka stanara
- **bulding** - za spremanje zgrada
- **building\_unit** - za spremanje stambenih jedinica (stanova u zgradama)
- **invitation** - za spremanje pozivnica koje šalju predstavnici suvlasnika
- **ticket** - za spremanje prijava kvarova
- **photo** - za spremanje fotografija pripadajućim kvarovima

- **rating** - za spremanje ocjena majstora
- **monthly\_report** - za spremanje mjesecnih izvještaja kvarova

#### Mrežni protokoli:

Protokol	Opis
HTTPS	Osnovni protokol za šifriranu komunikaciju između klijenta i poslužitelja.
OAuth 2.0	Sigurnosni protokol za autentifikaciju korisnika.
Stripe	Za plaćanja i verifikaciju transakcija majstora.
PDF Generator API	Za generiranje i preuzimanje izvještaja.

#### Globalni upravljački tok:

- 1. Korisnik putem web sučelja (React komponente) inicira akciju – npr. prijavu, prijavu kvara, plaćanje.
- 2. Frontend šalje zahtjev kroz Server Action ili HTTP zahtjev prema Next.js API ruti.
- 3. Next.js backend provodi validaciju, autentifikaciju kroz Supabase Auth, i komunicira s bazom podataka putem Supabase klijenta.
- 4. Ako je potrebno, backend kontaktira vanjski servis (Stripe za plaćanja, PDF API za izvještaje).
- 5. Rezultati obrade vraćaju se frontend aplikaciji (JSON odgovor za API rute ili direktno kroz Server Action).
- 6. Frontend ažurira prikaz korisniku kroz React komponente (npr. promjena statusa kvara ili potvrda o plaćanju).

#### Sklopovsko-programski zahtjevi:

Komponenta	Minimalni zahtjevi
Server	2 CPU jezgre, 2 GB RAM, 10 GB prostora, Node.js 18+, Next.js 16+
Korisnički uređaj	Moderan web preglednik (Chrome, Firefox, Edge), podrška za JavaScript i HTTPS
Operativni sustav	Linux (Ubuntu/Debian) za server; klijent može biti bilo koji OS s preglednikom
Dodatno	Internet veza, HTTPS certifikat, Supabase račun za bazu podataka

## Obrazloženje odabira arhitekture

Odabrana arhitektura temelji se na Next.js frameworku koji kombinira React za frontend i Node.js za backend u jednoj aplikaciji. Ovaj pristup najbolje odgovara zahtjevima projekta jer omogućuje jednostavniju izradu, testiranje i održavanje sustava unutar razvojnog okruženja. Aplikacija koristi Next.js App Router arhitekturu koja omogućava server-side rendering, API rute i React Server Components, čime se postiže jasno razdvajanje između prikazne i logičke razine uz optimalne performanse.

#### Izbor arhitekture temeljen na principima oblikovanja:

Pri oblikovanju arhitekture primjenjeni su sljedeći principi:

- **Visoka kohezija** – funkcionalnosti su grupirane prema odgovornostima; logika za korisnike, kvarove i plaćanja smještena je u zasebne module i Server Actions.
- **Niska povezanost** – moduli međusobno komuniciraju samo putem jasno definiranih API ruta i Server Actions, što omogućava lakšu izmjenu pojedinih dijelova sustava bez utjecaja na ostatak.
- **Modularnost i proširivost** – arhitektura omogućava jednostavno dodavanje novih stranica, API ruta ili komponenti bez potrebe za većim promjenama u postojećem kodu.
- **Jednostavnost održavanja** – korištenjem Next.js okvira i strukturiranog App Router direktorija, sustav je lako razumljiv i prilagođen timskom radu.
- **Sigurnost** – korištenje Supabase Auth za autentifikaciju i Next.js middleware-om za zaštitu ruta omogućuje sigurnu pohranu i pristup podacima po korisniku.

#### Razmatrane alternative:

Tijekom definiranja arhitekture razmatrani su i drugi pristupi:

- Arhitektura temeljena na mikrouslugama odbačena je zbog prevelike složenosti za projekt ove veličine te povećanog opterećenja u postavljanju i održavanju više servisa.
- Tradicionalni Express.js backend s odvojenim React frontendom također je razmatran, ali je odbačen zbog dodatne kompleksnosti u održavanju dva odvojena projekta. Umjesto toga, odabran je Next.js koji kombinira React frontend i Node.js backend u jednoj aplikaciji, omogućujući server-side rendering, API rute i optimizirane performanse kroz React Server Components.
- NoSQL baza (npr. MongoDB) razmatrana je, ali je odabrana relacijska baza podataka zbog jednostavnijeg modeliranja odnosa između entiteta.

## Organizacija sustava na visokoj razini

Sustav je organiziran prema Next.js App Router arhitekturi, u kojem klijent (korisnički web preglednik) komunicira s Next.js poslužiteljem putem HTTP protokola. Poslužitelj je implementiran pomoću Next.js frameworka koji kombinira React za UI i Node.js za server-side logiku. Prikaz sadržaja generira se poslužiteljski pomoću React Server Components, koji se dinamički renderiraju s podacima prije slanja korisniku, omogućujući optimalne performanse i SEO optimizaciju.

### Klijent poslužitelj:

Klijent predstavlja korisničko sučelje koje se prikazuje u web pregledniku. Svi zahtjevi korisnika šalju se HTTP zahtjevima prema Next.js poslužitelju. Poslužitelj obrađuje zahtjev kroz Next.js App Router, dohvaća potrebne podatke iz baze putem Supabase klijenta i vraća dinamički generiranu HTML stranicu korisniku. Za interaktivne akcije koriste se Server Actions ili API rute koje vraćaju JSON odgovore. Ova komunikacija odvija se u sinkronom obliku (zahtjev-odgovor), uz mogućnost optimiziranog server-side renderinga.

### Baza podataka:

Za pohranu podataka koristi se relacijska baza podataka (PostgreSQL) koja omogućuje povezanost njezinih entiteta. Baza podataka je hostana na Supabase platformi i omogućuje dosljedno pohranjivanje, ažuriranje i dohvat podataka.

Komunikacija između aplikacije i baze odvija se pomoću Supabase JavaScript klijenta, koji omogućava siguran pristup podacima kroz autentifikacijske tokene i Row Level Security (RLS) politike.

### Datotečni sustav:

Datotečni sustav koristi se za pohranu statičkih datoteka poput CSS stilova (Tailwind CSS), JavaScript skripti i slika. Ove datoteke dostupne su putem Next.js ugrađenog sustava za statičke resurse, što omogućuje jednostavnu integraciju i brzi pristup s klijentske strane. Next.js automatski optimizira i cache-ira statičke resurse za optimalne performanse.

### Grafičko sučelje:

Korisničko sučelje je web aplikacija izrađena pomoću React komponenti i JSX sintakse. Izgled korisničkog sučelja bitno ovisi o ulozi korisnika jer svaka uloga ima posebne funkcionalnosti koje oblikuju cijeli sustav. Sučelje je responzivno i jednostavno za korištenje, s naglaskom na jasnoću i preglednost. Stilizacija se postiže pomoću Tailwind CSS frameworka, dok se za kompleksnije UI komponente koristi Radix UI biblioteka. React komponente organizirane su u Next.js App Router strukturi, gdje svaka stranica (page.js) i layout (layout.js) automatski prima podatke iz Server Components ili Server Actions, čime se osigurava dinamički prikaz informacija u stvarnom vremenu uz optimalne performanse.

## Organizacija aplikacije

Arhitektura sustava HomeFix organizirana je prema slojevitom (layered) i MVC (Model–View–Controller) obrascu. Takva arhitektura omogućuje jasno razdvajanje odgovornosti između prikaza, poslovne logike i pristupa podacima, čime se olakšava održavanje, testiranje i budući razvoj sustava.

**Frontend sloj** predstavlja korisničko sučelje sustava, koje je dostupno putem web preglednika. Izrađen je pomoću React komponenti i JSX sintakse, uz Tailwind CSS za stilizaciju. Komponente mogu biti Client Components ("use client") za

interaktivnost ili Server Components za server-side rendering.

Glavne odgovornosti frontend sloja su:

- prikaz informacija korisniku ovisno o njegovoj ulozi kroz React komponente
- omogućavanje korisničke interakcije (prijava, unos novog kvara, plaćanje članarine) putem Server Actions ili API poziva
- slanje zahtjeva prema poslužitelju putem HTTP zahtjeva (GET, POST) ili direktno kroz Server Actions
- prikaz povratnih informacija i statusa sustava (uspješne obrade, greške i slično) uz real-time ažuriranje UI-a

**Backend sloj** predstavlja srž aplikacije, implementiran pomoću Next.js App Router arhitekture.

On upravlja svim zahtjevima iz korisničkog sučelja i obavlja obradu podataka kroz API rute i Server Actions, uključujući:

- validaciju korisničkih podataka u Server Actions ili API rutama
- obradu poslovne logike (npr. kreiranje i praćenje kvarova, ažuriranje statusa prijave, obračun i potvrdu plaćanja) kroz Server Actions
- komunikaciju s bazom podataka (spremanje i dohvati informacija) putem Supabase klijenta
- generiranje dinamičkih stranica pomoću React Server Components koji se renderiraju na serveru

Arhitektura sustava HomeFix temelji se na MVC obrascu (Model-View-Controller), koji razdvaja tri osnovne komponente aplikacije:

**Model** predstavlja logiku podataka i poslovna pravila sustava. Sadrži strukture podataka i funkcije koje omogućuju pristup bazi podataka kroz Supabase klijenta.

Primjeri modela u HomeFix sustavu uključuju:

- Korisnik (podaci o korisnicima, uloge, status članarine) - pristup kroz `profile`, `representative`, `contractor`, `tenant` tablice
- Kvar (opis problema, status, prijavljeni majstor, datum) - pristup kroz `ticket` tablicu
- Plaćanje (transakcije, iznos, datum, status) - pristup kroz `membership` tablicu i Stripe integraciju
- Izvještaj (generirani statistički podaci o prijavljenim kvarovima) - pristup kroz `monthly_report` tablicu

Modeli su implementirani kroz Server Actions ( `lib/actions/` ) koje omogućuju da svi ostali dijelovi aplikacije pristupaju podacima na standardiziran i siguran način kroz Supabase klijenta.

View predstavlja prikaz podataka korisniku kroz React komponente i Next.js stranice. Ovaj sloj sadrži minimalnu logiku za prikaz, većinom se koristi za vizualizaciju podataka koje mu dostavi Server Component ili Server Action.

Primjeri View komponenti i stranica:

- `app/(dashboard)/dashboard/page.js` — prikaz osnovnih informacija korisnika i kvarova
- `components/ui/card.jsx` — UI komponenta za prikaz kartica
- `app/(auth)/login/page.js` — stranica za prijavu korisnika koja koristi `LoginForm` komponentu

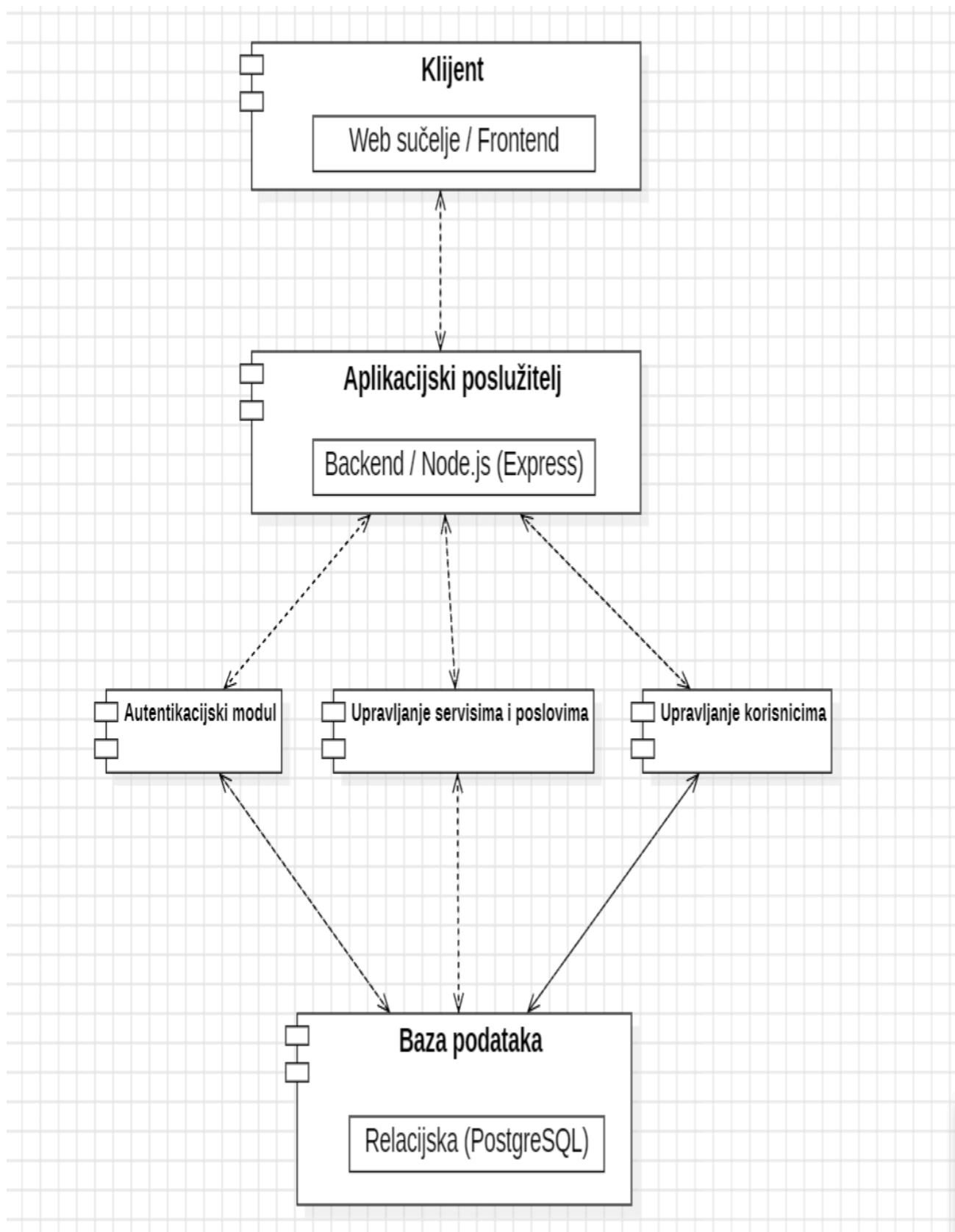
**Controller** djeluje kao posrednik između View-a i Modela. Obrađuje dolazne zahtjeve korisnika kroz Next.js API rute ( `app/(api)/*/route.js` ) i Server Actions ( `lib/actions/*.js` ), poziva odgovarajuće modele za dohvati ili ažuriranje podataka te proslijeđuje rezultate View sloju za prikaz.

Primjeri kontrolera:

- `lib/actions/auth-actions.js` — obrada prijave i registracije korisnika kroz Server Actions,
- `app/(api)/tickets/create/route.js` — API ruta za kreiranje prijava kvarova,
- `app/(checkout)/api/checkout_sessions/route.js` — API ruta za proces plaćanja članarine kroz Stripe,
- Server Actions u `lib/actions/` — generiranje izvještaja i poslovna logika.

Ova struktura omogućuje jednostavnu nadogradnju i modularnost, jer se svaki sloj može proširiti ili izmijeniti bez narušavanja ostalih dijelova sustava.

Dijagram visoke razine



Podsustav	Opis
Klijent (Frontend)	Predstavlja korisničko web sučelje. Korisnici ga koriste za prijavu, pregled i upravljanje servisnim zahtjevima. Komunicira s poslužiteljem putem API poziva.

Aplikacijski poslužitelj (Backend)	Središnji dio sustava koji sadrži poslovnu logiku, upravlja zahtjevima korisnika i komunicira s bazom podataka. Implementiran je u Next.js (React + Node.js).
Autentikacijski modul	Odgovoran za prijavu, registraciju i provjeru korisnika.
Modul za upravljanje korisnicima	Omogućuje administraciju i prikaz profila korisnika.
Modul za upravljanje servisima/poslovima	Omogućuje dodavanje, pregled i praćenje servisnih zahtjeva.
Baza podataka	Pohranjuje sve podatke o korisnicima, servisnim zahtjevima i statusima.

#### Izvadak primjera

*Sustav je dizajniran prema arhitekturi mikrousluga (engl.microservices), što znači da je funkcionalnost podijeljena u neovisne, ali međusobno povezane komponente (mikrousluge) koje zajednički obavljaju specifične zadatke unutar sustava. Svaka mikrousluga upravlja svojim vlastitim segmentom poslovne logike i podatkovnim potrebama, omogućujući lakšu skalabilnost, održavanje i razvoj.*

#### Ključne komponente

*Sustav se sastoji od nekoliko glavnih mikrousluga:*

*Usluga autentifikacije i autorizacije: Upravlja prijavom korisnika, autentifikacijom i autorizacijom te koristi JSON Web Token (JWT) za provjeru korisničkih zahtjeva.*

*Usluga upravljanja korisnicima: Upravlja kreiranjem, ažuriranjem i brisanjem korisničkih profila te čuva osnovne podatke o korisnicima.*

*Usluga za upravljanje sadržajem: Omogućuje korisnicima dodavanje, uređivanje i brisanje sadržaja unutar aplikacije.*

*Usluga notifikacija: Zadužena je za slanje obavijesti korisnicima putem e-pošte ili drugih kanala (koristi servise poput Firebase Cloud Messaging).*

*Usluga pretraživanja i filtriranja: Omogućuje pretragu i filtriranje podataka, koristeći Elasticsearch za brze pretrage velikih količina podataka.*

*Dijagram arhitekture: Dijagram prikazuje sve glavne mikrousluge u sustavu, njihovu međusobnu povezanost i komunikaciju s vanjskim servisima te bazom podataka.*

*API Gateway: Služi kao ulazna točka u sustav, prima zahtjeve od korisnika i prosljeđuje ih relevantnim mikrouslugama. Također osigurava autentifikaciju, autorizaciju i kontrolu pristupa.*

*Komunikacija među mikrouslugama: Komunikacija među mikrouslugama odvija se putem asinkronih poruka u Message Brokeru (npr. RabbitMQ ili Kafka) ili putem REST API poziva, ovisno o potrebama. Na taj način se osigurava visok stupanj izolacije i otpornosti na greške.*

*Baza podataka: Svaka mikrousluga ima vlastitu bazu podataka, omogućujući izolaciju podataka i neovisno upravljanje. Na primjer, usluga korisnika koristi relacijsku bazu podataka (npr. PostgreSQL), dok usluga pretraživanja koristi dokumentnu bazu (npr. MongoDB).*

*Vanjski servisi: Povezivanje s vanjskim servisima, kao što su pružatelji identifikacijskih usluga (OAuth) i servisi za slanje notifikacija (Twilio za SMS, Firebase za push notifikacije), omogućuje lakšu integraciju s vanjskim ekosustavom.*

#### Prednosti arhitekture mikrousluga

Ova arhitektura omogućuje:

**Skalabilnost:** Svaka mikrousluga može se skalirati neovisno, čime se optimiziraju resursi ovisno o specifičnim zahtjevima (horizontalna ili vertikalna skalabilnost).

**Otpornost na greške:** Zbog izolacije mikrousluga, pogreška u jednoj komponenti ne utječe na cijeli sustav.

**Fleksibilnost u razvoju i implementaciji:** Različiti timovi mogu raditi na različitim uslugama paralelno, koristeći optimalne tehnologije za svaki segment.

Referenca Ovaj pristup je u skladu s principima mikrouslužne arhitekture kako ih definiraju Fowler i Lewis (2014) te s industrijskim praksama navedenim u "Building Microservices" (Newman, 2015), čiji su primjeri korišteni za detaljnije definiranje uloga i komunikacije među mikrouslugama.

## Baza podataka

Relacijsku bazu podataka aplikacije modelirali smo koristeći **Supabase** (PostgreSQL). Prednosti Supabase-a su što ima jednostavnu implementaciju autentifikacije (auth.users) i odlično integrira s NextJs-om.

GLAVNE KOMPONENTE:

**Shema public** s tablicama: profile, representative, contractor, tenant, building, building\_unit, invitation, membership, ticket, photo, rating, monthly\_report

**Enumeracije:**

- role\_enum - TENANT, CONTRACTOR, REPRESENTATIVE, ADMIN (uloga korisnika)
- issue\_category – ELECTRICAL, PLUMBING, CARPENTRY, GENERAL (vrsta kvara)
- specialization - ELECTRICIAN, PLUMBER, CARPENTER, GENERAL (specijalizacija majstora)
- ticket\_status – OPEN, IN\_PROGRESS, RESOLVED (status kvara)
- payment\_status - PENDING, SUCCEEDED, FAILED (status plaćanja)
- currency – EUR, USD, GBP (valuta plaćanja)

**Autentifikacija:** profile.user\_id je strani ključ FK u auth.users(id) Supabase tablici. Ostale tablice tipično referenciraju profile.user\_id.

## Opis tablica

### profile

Atribut	Tip podatka	Opis varijable
<u>user_id</u>	uuid	jedinstveni identifikator korisnika
email	text	e-mail korisnika s kojim se prijavljuje u aplikaciju
first_name	text	ime korisnika
last_name	text	prezime korisnika
role	role_enum (enum)	uloga korisnika (predstavnik stanara, majstor, stanar ili administrator)
created_at	timestamptz	vrijeme kreiranja

### representative (predstavnik stanara)

Atribut	Tip podatka	Opis varijable

<u>user_id</u>	uuid (FK)	jedinstveni identifikator korisnika
building_id	int (FK)	jedinstveni identifikator zgrade u kojoj je korisnik predstavnik stanara
created_at	timestamptz	vrijeme kreiranja

### contractor (majstor)

Atribut	Tip podatka	Opis varijable
<u>user_id</u>	uuid (FK)	jedinstveni identifikator korisnika
phone	text	broj telefona majstora
specialization	specialization (enum)	specijalizacija majstora (električar, stolar, vodoinstalater ili opći - za razne/ostale kvarove)
company_name	text	naziv obrta ukoliko majstor pripada istom
created_at	timestamptz	vrijeme kreiranja

### membership (članstvo majstora)

Atribut	Tip podatka	Opis varijable
<u>user_id</u>	uuid (FK)	jedinstveni identifikator korisnika, odnosno majstora čija je pretplata
price	float	cijena članstva
expires_at	timestamp	datum i vrijeme kada članstvo ističe
last_paid	timestamp	datum i vrijeme kada je članstvo zadnje plaćeno
currency	currency (enum)	valuta plaćanja članstva
created_at	timestamptz	vrijeme kreiranja

### tenant (stanar)

Atribut	Tip podatka	Opis varijable
<u>user_id</u>	uuid (FK)	jedinstveni identifikator korisnika
unit_id	int (FK)	jedinstveni identifikator stambene jedinice u kojoj stanar boravi
created_at	timestamptz	vrijeme kreiranja

### building (zgrada)

Atribut	Tip podatka	Opis varijable
<u>building_id</u>	int	jedinstveni identifikator zgrade
addresss	text	adresa zgrade
postal_code	int	poštanski broj mjesta u kojem se zgrada nalazi
created_at	timestamptz	vrijeme kreiranja

### building\_unit (stambena jedinica)

Atribut	Tip podatka	Opis varijable
<u>unit_id</u>	int	jedinstveni identifikator stambene jedinice u zgradu
building_id	int (FK)	id zgrade kojoj stambena jedinica pripada
label	text	opis jedinice, npr. stan 2
floor	int	kat zgrade na kojem se jedinica nalazi
created_at	timestamptz	vrijeme kreiranja

#### **invitation (pozivnica koju šalje predstavnik svojim stanarima)**

Atribut	Tip podatka	Opis varijable
<u>invitation_id</u>	int	jedinstveni identifikator pozivnice za registraciju
from_id	uuid (FK)	id predstavnika stanara koji šalje pozivnicu
to_email	text	e-mail na koji se šalje pozivnica
created_at	timestamptz	vrijeme kreiranja

#### **ticket (kvar)**

Atribut	Tip podatka	Opis varijable
<u>ticket_id</u>	int	jedinstveni identifikator prijavljenog kvara
created_by	uuid (FK)	id stanara koji prijavljuje kvar
unit_id	int (FK)	id stambene jedinice u kojoj je kvar
issue_category	issue_category (enum)	vrsta kvara (električni, stolarski, vodoinstalacijski ili opći - razni/ostali kvarovi)
title	text	naziv kvara, upisuje stanar
description	text	opis problema koji unosi stanar prilikom prijave kvara
assigned_to	uuid (FK)	id majstora koji je dodijeljen od strane predstavnika za rješavanje tog kvara
status	ticket_status (enum)	status rješavanja kvara (otvoreno, u tijeku, riješeno)
resolved_at	timestamp	datum i vrijeme rješavanja kvara, potrebno za statistiku kvarova
comment	text	napomena o kvaru koju ostavlja majstor
created_at	timestamptz	vrijeme kreiranja

#### **photo (fotografija kvara)**

Atribut	Tip podatka	Opis varijable
<u>photo_id</u>	int	jedinstveni identifikator fotografije kvara koju postavlja stanar
ticket_id	int (FK)	id kvara kojem fotografija pripada
photo_url	text	url fotografije

created_at	timestamptz	vrijeme kreiranja
------------	-------------	-------------------

#### rating (ocjena stanara majstoru za rješavanje kvara)

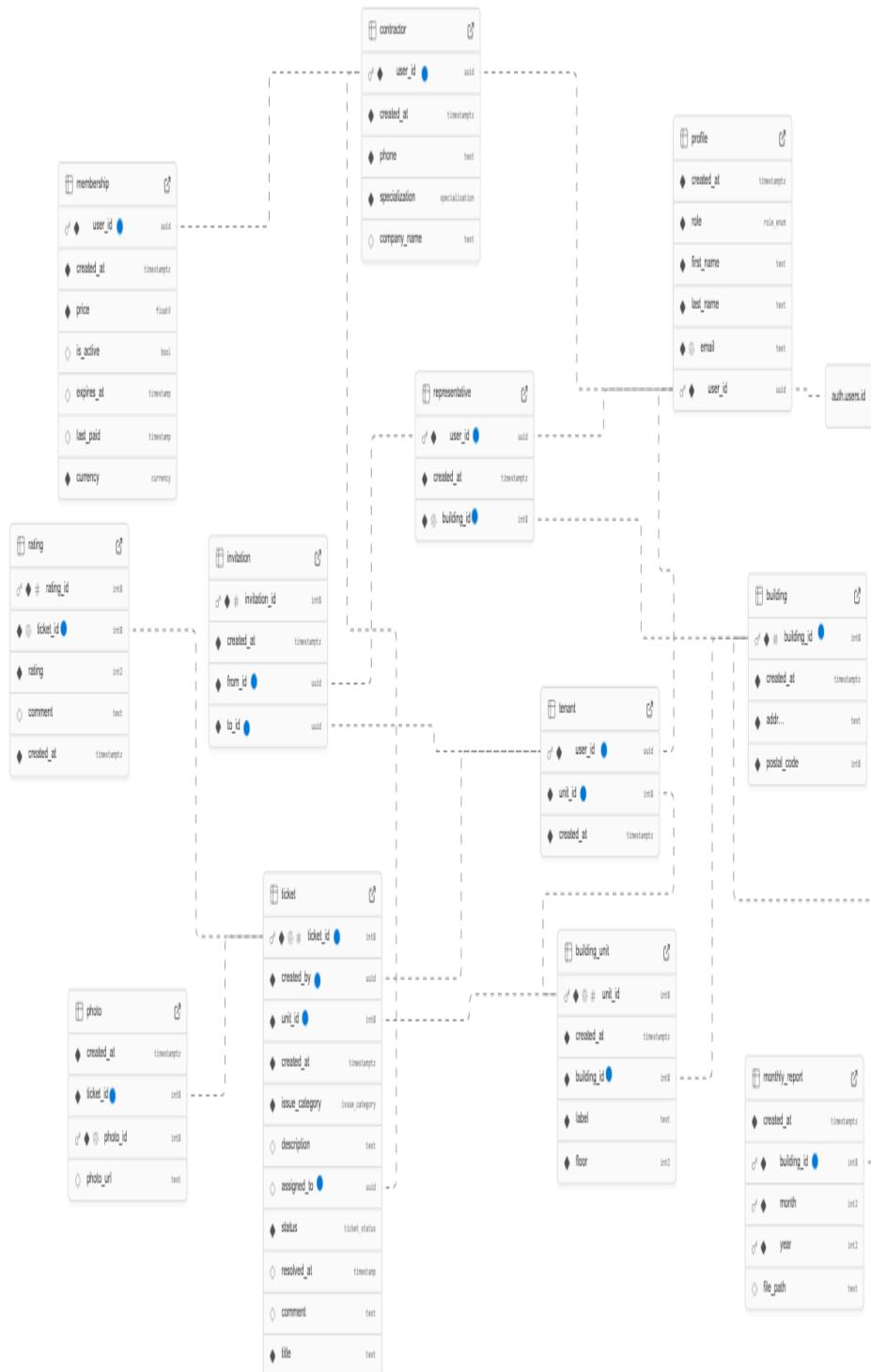
Atribut	Tip podatka	Opis varijable
<u>rating_id</u>	int	jedinstveni identifikator ocjene koju stanar ostavlja majstoru za rješavanje kvara
ticked_id	int (FK)	id kvara na koji se ocjena odnosi
rating	int	ocjena od 1 do 5
comment	text	recenzija koju piše stanar o majstoru
created_at	timestamptz	vrijeme kreiranja

#### monthly\_report (PDF izvještaji)

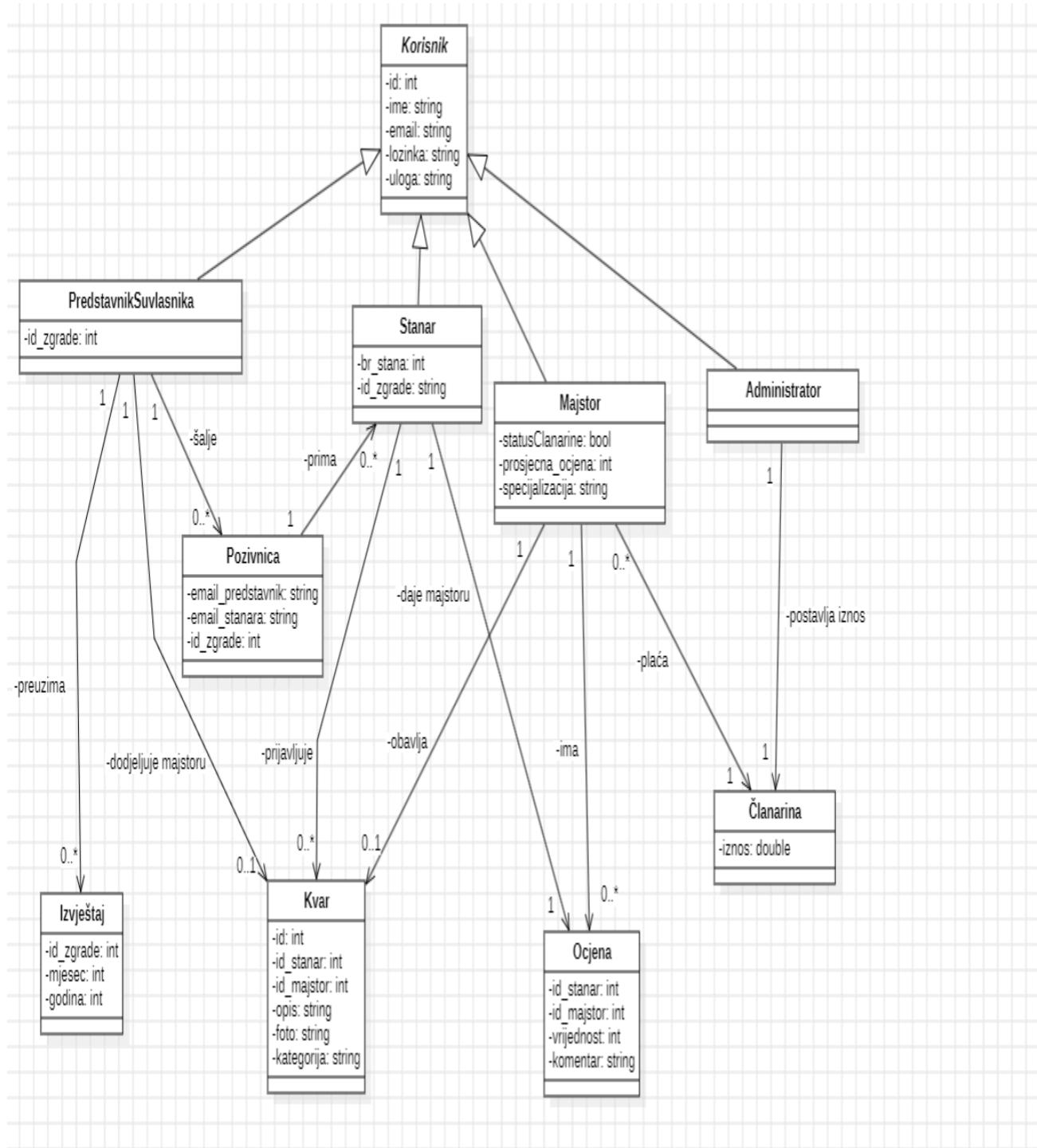
Atribut	Tip podatka	Opis varijable
<u>building_id</u>	int (FK)	id zgrade na koju se izvještaj odnosi
<u>month</u>	int	mjesec u godini na koji se izvještaj odnosi
<u>year</u>	int	godina na koju se izvještaj odnosi
file_path	text	put do mjesecnog pdf izvještaja svih kvarova

#### Dijagram baze podataka

- primarni ključevi imaju ikonicu ključa, strani ključevi imaju plave točkice



## Dijagram razreda

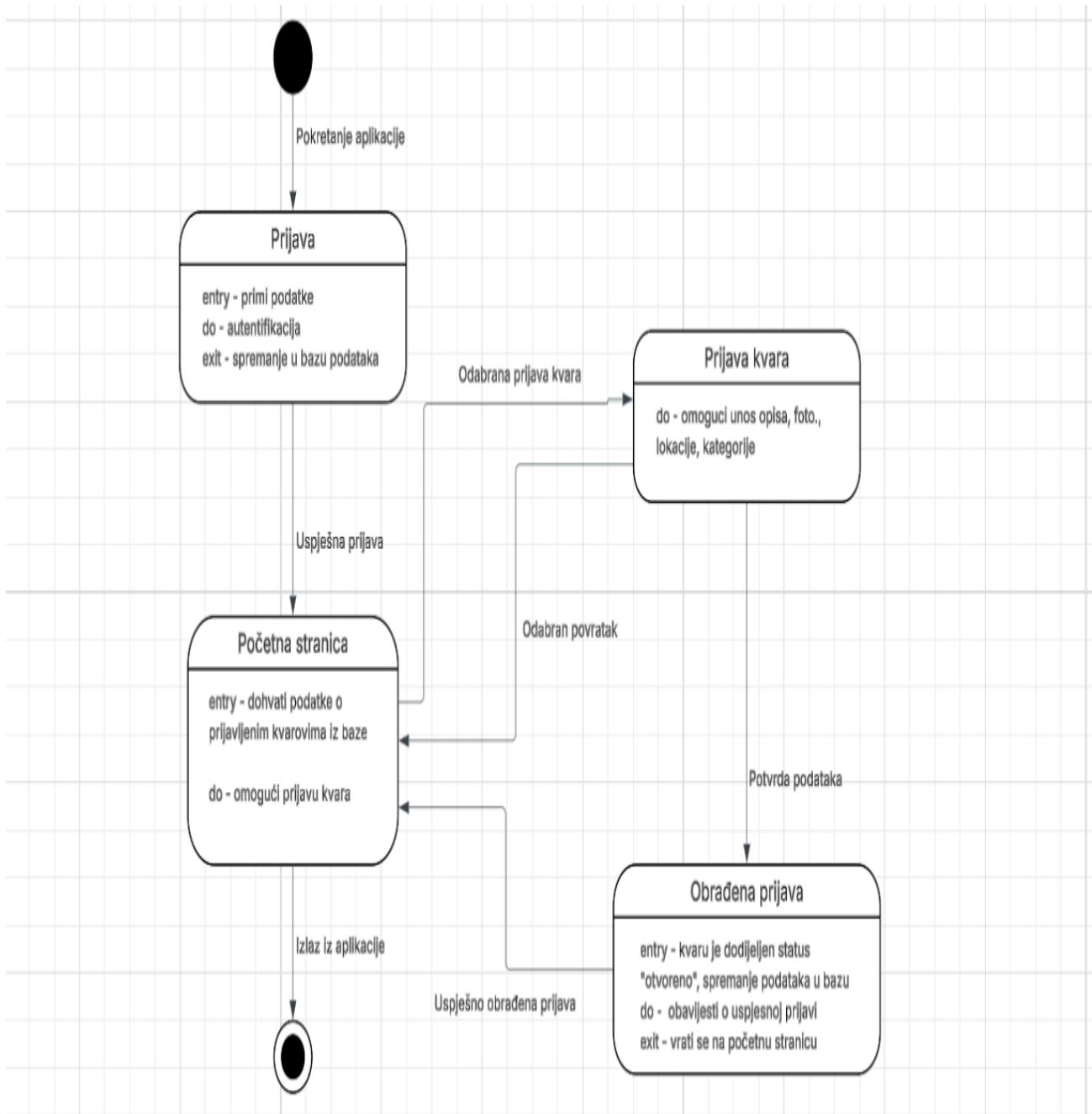


- **Korisnik:** svaki ima zasebni id, svoje ime, email, lozinku i ulogu. Klase koje ga nasljeđuju su **Stanar, Majstor, PredstavnikSuvlasnika i Administrator**. Svaki korisnik dobiva različite funkcionalnosti u sustavu s obzirom na dodijeljenu ulogu.
- **Stanar:** mora tijekom registracije naznačiti adresu zgrade i broj stana u kojem stanuje. Glavna mu je uloga da može prijaviti više kvarova (0..\*) u sustav. Nakon popravljenog kvara može ostaviti po jednu ocjenu (1) majstoru koji je obavio kvar te opcionalno ostaviti komentar. Svaki stanar može od predstavnika suvlasnika svoje zgrade primiti jednu email pozivnicu za prijavu u sustav (1).
- **Majstor:** na profilu ima vidljivu prosječnu ocjenu i svoju specijalizaciju. Za obavljanje kvarova, mora plaćati članarinu u sustav (1) po označenom iznosu.

- **PredstavnikSuvlasnika:** tijekom registracije mora naznačiti adresu zgrade (id\_zgrade) koju predstavlja. Glavna mu je funkcionalnost da svakom majstoru može dodijeliti po jedan prijavljen kvar (0..1) te može preuzeti PDF izvještaje kvarova za određene mjeseca u godini (0..). Može slati pozivnice (0..) stanarima putem emaila za registraciju u sustav.
- **Administrator:** postavlja iznos članarine majstora (1).
- **Kvar:** ima naznačenu pripadnost stanaru (id\_stanar) (1) koji ga prijavljuje te majstoru (id\_majstor) (1) koji ga obavlja. Sadrži se od opisa, fotografije i kategorije koje je postavio stanar tijekom prijave kvara. Svaki kvar dodjeljuje pojedini predstavnik suvlasnika (1) određenom majstoru (1).
- **Ocjena:** ima naznačenu pripadnost stanaru (1) koji ju postavlja te majstoru (1) kojem je dodijeljena. Stanar bira vrijednost od 1 do 5 te opcionalno ostavlja komentar.
- **Izvještaj:** ima označeni identifikator zgrade (id\_zgrade) koji se određuje po predstavniku suvlasnika koji ga želi preuzeti. Svaki se izvještaj razlikuje po mjesecu i godini koju odabire predstavnik suvlasnika. Svaki izvještaj kvarova u zgradu preuzima predstavnik suvlasnika koji predstavlja pripadajuću zgradu.
- **Članarina:** ima iznos koju postavlja administrator sustava (1). Članarini moraju platiti svi majstori koji se žele prijaviti u sustav (0..\*).
- **Pozivnica:** sadrži email adresu predstavnika suvlasnika (pošiljatelja) i email adrese stanara (primatelja). Također ima naznačen id\_zgrade od predstavnika kojim se određuje kojim će stanarima u sustavu pojedini predstavnik suvlasnika (1) poslati pozivnice (onima koji imaju isti id\_zgrade kao predstavnik). Jedna pozivnica (1) može se poslati većem broju (0..\*) stanara ako se podudara vrijednost id\_zgrade.

## Dinamičko ponašanje aplikacije

### UML dijagrami stanja

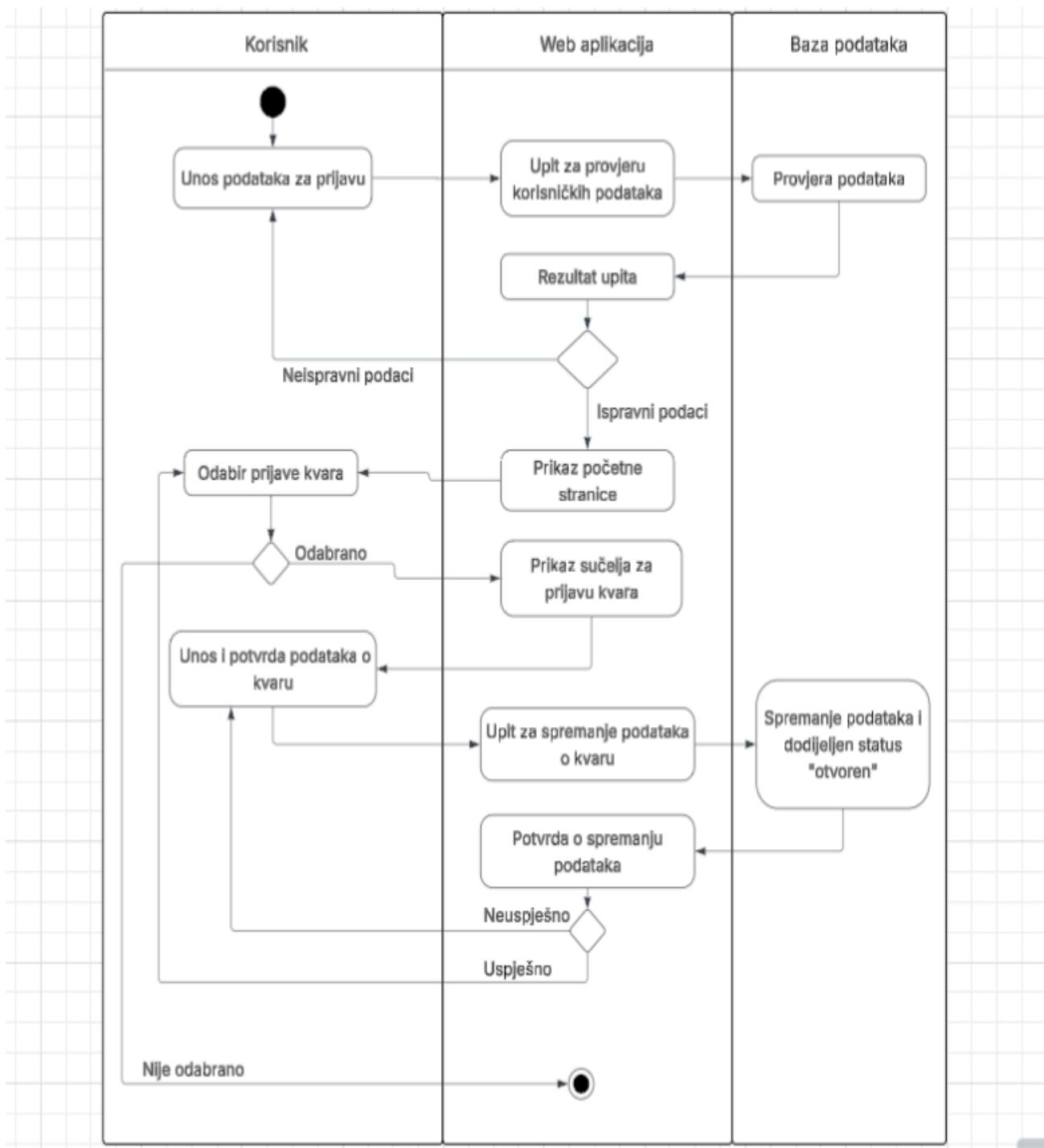


Dijagram stanja za sustav HomeFix prikazuje tok aktivnosti korisnika kod procesa prijave kvara. Korisnik prvo otvara aplikaciju te započinje interakciju u početnom stanju (prijava). Prijava uključuje unos korisničkog imena i zaporce, autentifikaciju i spremanje korisničkih podataka u bazu. Ako je prijava uspješna, korisnik prelazi na početnu stranicu.

Na početnoj stranici korisnik ima opciju prijave novog kvara ili izlaska iz aplikacije. Ako je odabrana prijava kvara, korisniku se prikazuje sučelje za prijavu kvara. Ovdje korisnik unosi opis kvara i njegovu kategoriju, svoju lokaciju u zgradi te dodatno može unijeti fotografiju kvara. U ovom sučelju korisnik se može vratiti na početnu stranicu. Kada se unesu podatci te ako ih korisnik ih potvrdi, sustav prelazi u stanje obrade prijave kvara.

U obradi prijave, sustav prijavljenom kvaru dodjeljuje status "otvoreno" i sprema unesene podatke u bazu podataka te na kraju obavještava korisnika je li njegova prijava obrađena ili ne. Nakon obavještavanja, sustav vraća korisnika na početnu stranicu.

## UML dijagrami aktivnosti



Dijagram stanja prikazuje tijek izvršavanja prijave kvara.

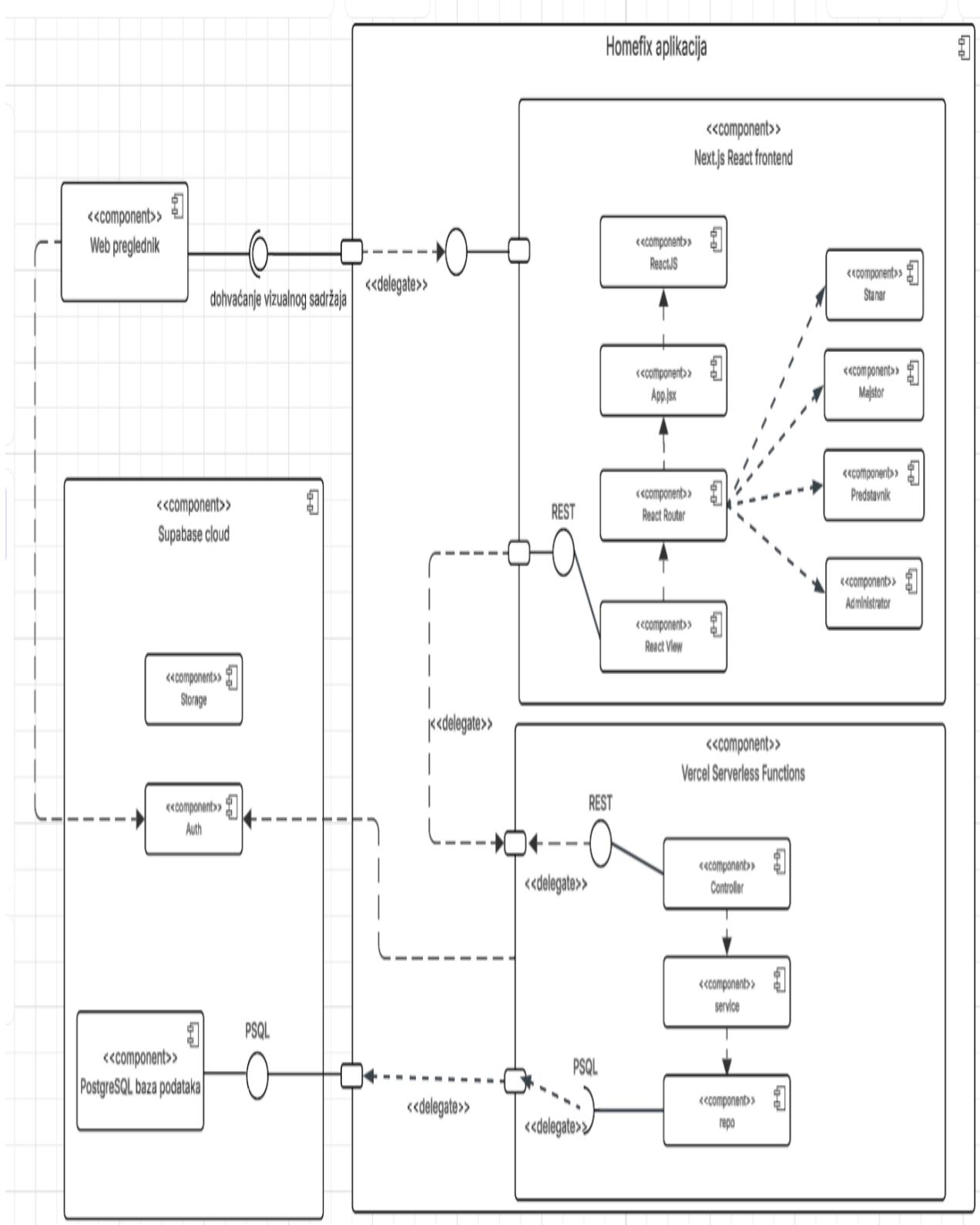
Korisnik se prvo prijavljuje unosom podataka koje preglednik provjerava pomoću baze podataka. Ako su podaci ispravni, otvara se početna stranica sustava, a u suprotnom se proces prijave ponavlja.

Na početnoj stranici korisnik može prijaviti svoj kvar. Ako nema potrebu za prijaviti kvar, korisnik može izaći iz aplikacije.

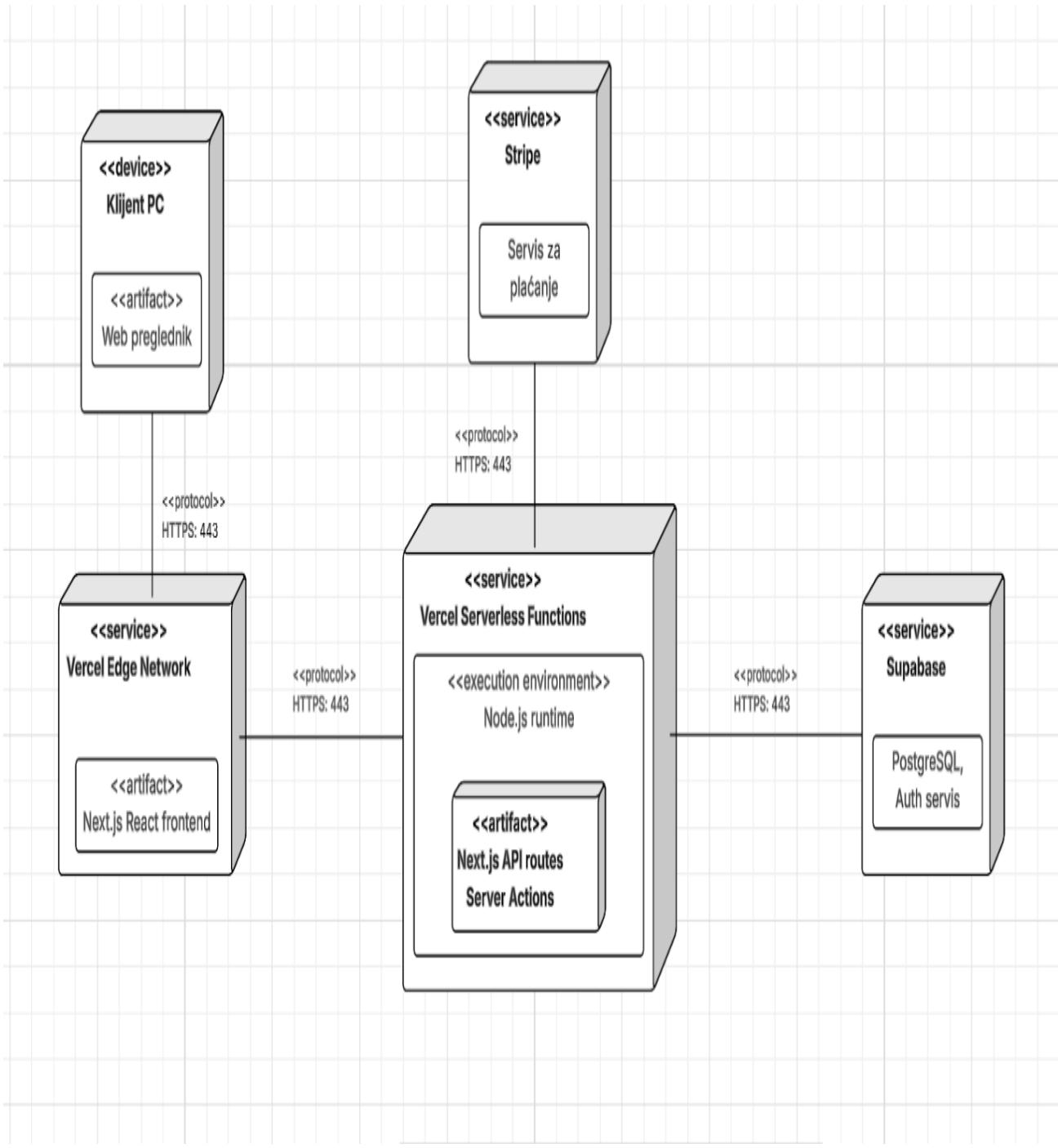
Ako je odabrana prijava kvara, korisnik unosi podatke o kvaru. Nakon unosa podataka korisnik potvrđuje njihovu ispravnost te preglednik sprema te podatke u bazi podataka. U bazi podataka svaki novi prijavljeni kvar inicijalno ima postavljen status "otvoreno".

Nakon spremanja, preglednik obavještava korisnika je li prijava odobrena ili ne. Ako je, preglednik vraća korisniku na početnu stranicu gdje može prijaviti novi kvar, a ako nije, korisnik je vraćen na unos podataka o kvaru.

## Dijagram komponenata



## Dijagram razmještaja



Dijagram prikazuje raspodjelu komponenti web aplikacije u mrežnom okruženju. Korisnik pristupa sustavu putem web preglednika na klijentskom računalu. Statički dio aplikacije (Next.js React frontend) servira se s Vercel Edge mreže, dok se poslovna logika i obrada zahtjeva izvršavaju u sklopu Vercel Serverless Functions servisa u Node.js izvršnom okruženju. Serverless backend komunicira s vanjskim servisom Stripe radi obrade plaćanja majstorskih preplata te sa Supabase servisom koji osigurava PostgreSQL bazu podataka i autentifikacijski servis. Komunikacija između svih komponenti odvija se putem HTTPS protokola, čime se osigurava sigurna komunikacija i zaštita korisničkih podataka.

## Ispitivanje komponenti

## Test 01: Validacija ulaznih podataka za kreiranje ticketa

### Funkcionalnost koju testiramo:

Metoda `createTicket({ title, description, issue_category })` iz razreda `tickets-actions.js`, koja kreira novi ticket u sustavu za upravljanje zgradama.

### Ispitni slučaj 1: Uspješno kreiranje ticketa sa validnim podacima (redovni slučaj)

#### Ulazni podaci:

- `title` : "Problema s vodom"
- `description` : "Cureća cijev u kuhinji"
- `issue_category` : "PLUMBING"
- Autentificirani korisnik s `user_id` : "user-123"
- Tenant podatak: `unit_id` : "unit-456"

#### Postupak provođenja ispitivanja:

- Pokrenite testove naredbom: `npm test tickets-actions.test.js`
- Mock autentificiranog korisnika preko Supabase `auth.getUser()`
- Mock tenant podataka iz tablice `tenant`
- Mock uspješnog inserta u tablicu `ticket`
- Pozovite funkciju `createTicket()` s validnim parametrima
- Provjerite status odgovora i vraćene podatke

#### Dobiveni rezultati:

- `error` : null
- `data` :

```
{  
  "ticket_id": "ticket-789",  
  "title": "Problema s vodom",  
  "description": "Cureća cijev u kuhinji",  
  "issue_category": "PLUMBING",  
  "status": "OPEN",  
  "created_by": "user-123",  
  "unit_id": "unit-456"  
}
```

- Funkcija `revalidatePath("/tickets")` je pozvana

### Prolaz ispitivanja

### Ispitni slučaj 2: Kreiranje ticketa sa praznim title poljem (rubni uvjet)

#### Ulazni podaci:

- `title` : "" (prazan string)
- `description` : "Opis problema"
- `issue_category` : "PLUMBING"
- Autentificirani korisnik

#### Postupak provođenja ispitivanja:

- Pokrenite testove naredbom: `npm test tickets-actions.test.js`
- Mock autentificiranog korisnika

3. Pozovite funkciju `createTicket()` s praznim `title` poljem
4. Provjerite da je vraćen odgovarajući error
5. Provjerite da Supabase insert nije pozvan

**Dobiveni rezultati:**

- `error` : "Nedostaju obavezna polja: title, description, issue\_category"
- `data` : undefined
- Supabase insert nije pozvan
- `revalidatePath` nije pozvan

 **Prolaz ispitivanja**

---

**Sažetak:**

Testirane su različite situacije za metodu `createTicket()`. Pokriveni su redovni slučaj s validnim podacima i rubni uvjet s nedostajućim obaveznim poljima. Validacija ulaznih podataka radi ispravno.

---

## Test 02: Dodjela majstora ticketu prema specijalizaciji

**Funkcionalnost koju testiramo:**

Metoda `assignContractor(ticketId, assigned_to)` iz razreda `tickets-actions.js`, koja dodjeljuje majstora ticketu uz provjeru kompatibilnosti specijalizacije.

---

### Ispitni slučaj 1: Dodjela majstora sa nekompatibilnom specijalizacijom (exception throwing)

**Ulazni podaci:**

- `ticketId` : "ticket-123"
- `assigned_to` : "contractor-456"
- Ticket s kategorijom: "PLUMBING"
- Majstor sa specijalizacijom: "ELECTRICIAN"
- Autentificirani korisnik sa ulogom: "REPRESENTATIVE"

**Postupak provođenja ispitivanja:**

1. Pokrenite testove naredbom: `npm test tickets-actions.test.js`
2. Mock autentificiranog korisnika s ulogom REPRESENTATIVE
3. Mock ticket podataka s kategorijom PLUMBING
4. Mock building podataka i representative veze
5. Mock contractor podataka sa specijalizacijom ELECTRICIAN
6. Pozovite funkciju `assignContractor()`
7. Provjerite da je vraćen error o nekompatibilnosti

**Dobiveni rezultati:**

- `error` : "Nekompatibilno: kategorija kvara (PLUMBING) ≠ specijalizacija (ELECTRICIAN)."
- `data` : undefined
- Ticket nije ažuriran u bazi podataka

 **Prolaz ispitivanja**

---

**Sažetak:**

Testirana je metoda `assignContractor()` za slučaj nekompatibilne specijalizacije majstora. Sustav ispravno odbija dodjelu majstora čija specijalizacija ne odgovara kategoriji kvara.

---

## Test 03: Ažuriranje statusa ticketa

### Funkcionalnost koju testiramo:

Metoda `updateTicketStatus(ticketId, status)` iz razreda `tickets-actions.js`, koja ažurira status ticketa s validacijom dopuštenih vrijednosti.

### Ispitni slučaj 1: Ažuriranje statusa sa nevaljanim statusom (rubni uvjet)

#### Ulagani podaci:

- `ticketId` : "ticket-123"
- `status` : "INVALID\_STATUS" (nevaljan status)
- Autentificirani korisnik

#### Postupak provođenja ispitivanja:

1. Pokrenite testove naredbom: `npm test tickets-actions.test.js`
2. Mock autentificiranog korisnika
3. Pozovite funkciju `updateTicketStatus()` sa nevaljanim statusom
4. Provjerite da je vraćen odgovarajući error
5. Provjerite da Supabase update nije pozvan

#### Dobiveni rezultati:

- `error` : "Neispravan 'status' (dozvoljeno: OPEN, IN\_PROGRESS, RESOLVED)."
- `data` : undefined
- Supabase update nije pozvan

#### Prolaz ispitivanja

### Sažetak:

Testirana je metoda `updateTicketStatus()` za slučaj nevaljanog statusa. Validacija statusa radi ispravno i sprječava postavljanje nedopuštenih vrijednosti.

## Test 04: Dohvaćanje nepostojećeg ticketa

### Funkcionalnost koju testiramo:

Metoda `getTicket(ticketId)` iz razreda `tickets-actions.js`, koja dohvaća podatke o ticketu na temelju ID-a.

### Ispitni slučaj 1: Dohvaćanje ticketa koji ne postoji (nepostojeća funkcionalnost)

#### Ulagani podaci:

- `ticketId` : "non-existent-ticket-999"
- Pretpostavka: Ticket s ovim ID-em ne postoji u bazi podataka
- Autentificirani korisnik s `user_id` : "user-123"

#### Postupak provođenja ispitivanja:

1. Pokrenite testove naredbom: `npm test tickets-actions.test.js`
2. Mock autentificiranog korisnika
3. Mock Supabase query koji vraća error za nepostojeći ticket
4. Pozovite funkciju `getTicket()` s nepostojećim ID-em
5. Provjerite da je vraćen odgovarajući error

#### Dobiveni rezultati:

- `error` : "Ticket nije pronađen."
- `data` : null

 **Prolaz ispitivanja**

---

### Sažetak:

Testirana je metoda `getTicket()` za slučaj nepostojećeg ticketa. Sustav ispravno obrađuje slučaj kada traženi resurs ne postoji u bazi podataka.

---

## Test 05: Provjera članstva majstora

### Funkcionalnost koju testiramo:

Metoda `checkMembership(userId)` iz razreda `contractor-actions.js`, koja provjerava je li članstvo majstora aktivno na temelju datuma isteka.

---

### Ispitni slučaj 1: Provjera aktivnog članstva (redovni slučaj)

#### Ulagani podaci:

- `userId` : "contractor-123"
- `expires_at` : Datum u budućnosti (1 godina od trenutnog datuma)

#### Postupak provođenja ispitivanja:

1. Pokrenite testove naredbom: `npm test contractor-actions.test.js`
2. Mock Supabase query za tablicu `membership`
3. Postavite `expires_at` na datum u budućnosti
4. Pozovite funkciju `checkMembership()`
5. Provjerite da je vraćena vrijednost `paid: true`

#### Dobiveni rezultati:

- `error` : undefined
- `paid` : true

 **Prolaz ispitivanja**

---

### Ispitni slučaj 2: Provjera isteklog članstva (rubni uvjet - granica isteka)

#### Ulagani podaci:

- `userId` : "contractor-456"
- `expires_at` : Datum u prošlosti (1 sekunda prije trenutnog vremena)

#### Postupak provođenja ispitivanja:

1. Pokrenite testove naredbom: `npm test contractor-actions.test.js`
2. Mock Supabase query za tablicu `membership`
3. Postavite `expires_at` na datum u prošlosti
4. Pozovite funkciju `checkMembership()`
5. Provjerite da je vraćena vrijednost `paid: false`

#### Dobiveni rezultati:

- `error` : undefined
- `paid` : false

 **Prolaz ispitivanja**

---

### Ispitni slučaj 3: Provjera članstva bez expires\_at datuma (rubni uvjet)

#### Ulazni podaci:

- `userId` : "contractor-789"
- `expires_at` : null

#### Postupak provođenja ispitivanja:

1. Pokrenite testove naredbom: `npm test contractor-actions.test.js`
2. Mock Supabase query za tablicu `membership`
3. Postavite `expires_at` na null
4. Pozovite funkciju `checkMembership()`
5. Provjerite da je vraćena vrijednost `paid: null`

#### Dobiveni rezultati:

- `error` : undefined
- `paid` : null

 **Prolaz ispitivanja**

---

#### Sažetak:

Testirana je metoda `checkMembership()` za različite scenarije: aktivno članstvo, isteklo članstvo i članstvo bez postavljenog datuma isteka. Logika provjere članstva radi ispravno i pokriva sve rubne uvjete.

---

## Test 07: Validacija uloga za kreiranje zgrade

#### Funkcionalnost koju testiramo:

Metoda `createBuilding(formData)` iz razreda `building-actions.js`, koja kreira novu zgradu u sustavu uz provjeru autorizacije korisnika.

---

### Ispitni slučaj 1: Kreiranje zgrade sa ulogom REPRESENTATIVE (redovni slučaj)

#### Ulazni podaci:

- `formData` :
  - `address` : "Ulica 123"
  - `postalCode` : "10000"
- Autentificirani korisnik s `user_id` : "user-123"
- Uloga korisnika: "REPRESENTATIVE"

#### Postupak provođenja ispitivanja:

1. Pokrenite testove naredbom: `npm test building-actions.test.js`
2. Mock autentificiranog korisnika
3. Mock profile podataka s ulogom REPRESENTATIVE
4. Mock uspješnog inserta u tablicu `building`
5. Mock uspješnog inserta u tablicu `representative`
6. Pozovite funkciju `createBuilding()` s validnim podacima
7. Provjerite status odgovora i vraćene podatke

#### Dobiveni rezultati:

- `error` : undefined
- `data` :

```
{  
  "building": {  
    "building_id": "building-456",  
    "address": "Ulica 123",  
    "postal_code": "10000"  
  },  
  "representative": {  
    "user_id": "user-123",  
    "building_id": "building-456"  
  }  
}
```

Prolaz ispitivanja

### Ispitni slučaj 2: Pokušaj kreiranja zgrade bez uloge REPRESENTATIVE (exception throwing)

Ulazni podaci:

- `formData` :
  - `address` : "Ulica 456"
  - `postalCode` : "20000"
- Autentificirani korisnik s `user_id` : "user-789"
- Uloga korisnika: "TENANT" (nije REPRESENTATIVE)

Postupak provođenja ispitivanja:

1. Pokrenite testove naredbom: `npm test building-actions.test.js`
2. Mock autentificiranog korisnika
3. Mock profile podataka s ulogom TENANT
4. Pozovite funkciju `createBuilding()`
5. Provjerite da je vraćen error o nedostatku ovlasti
6. Provjerite da insert zgrade nije pozvan

Dobiveni rezultati:

- `error` : "Only representatives can create buildings"
- `data` : undefined
- Supabase insert za `building` tablicu nije pozvan

Prolaz ispitivanja

### Ispitni slučaj 3: Pokušaj kreiranja zgrade bez autentifikacije (exception throwing)

Ulazni podaci:

- `formData` :
  - `address` : "Ulica 789"
  - `postalCode` : "30000"
- Korisnik nije autentificiran (`user: null`)

Postupak provođenja ispitivanja:

1. Pokrenite testove naredbom: `npm test building-actions.test.js`
2. Mock Supabase `auth.getUser()` koji vraća null
3. Pozovite funkciju `createBuilding()`
4. Provjerite da je vraćen error o nedostatku autentifikacije

Dobiveni rezultati:

- `error` : "Not authenticated"
- `data` : undefined

## Prolaz ispitivanja

### Sažetak:

Testirana je metoda `createBuilding()` za različite scenarije autorizacije: uspješno kreiranje s odgovarajućom ulogom, odbijanje korisnika bez odgovarajuće uloge, i odbijanje neautentificiranih korisnika. Sustav ispravno provjerava ovlasti prije izvršavanja operacija.

## Ukupni sažetak testiranja

**Ukupno testirano:** 7 glavnih funkcionalnosti (11 ispitnih slučajeva)

### Kategorije testova:

- **Redovni slučajevi:** 3 testa (Test 01.1, Test 05.1, Test 07.1)
- **Rubni uvjeti:** 4 testa (Test 01.2, Test 03.1, Test 05.2, Test 05.3)
- **Exception throwing:** 3 testa (Test 02.1, Test 07.2, Test 07.3)
- **Nepostojeća funkcionalnost:** 1 test (Test 04.1)

### Rezultati:

-  Svi testovi prolaze uspješno: **11/11**
- Pokretanje testova: `npm test`
- Testing framework: **Jest** s **Babel** transformacijom

### Datoteke testova:

- `__tests__/tickets-actions.test.js` - Testovi za tickete (Test 01-04)
- `__tests__/contractor-actions.test.js` - Testovi za majstore (Test 05)
- `__tests__/building-actions.test.js` - Testovi za zgrade (Test 07)

**Datum testiranja:** Siječanj 2026

**Zaključak:** Sve testirane komponente rade ispravno i pokrivaju redovne slučajeve, rubne uvjete, izazivanje iznimki i nepostojeće funkcionalnosti. Validacija ulaznih podataka i autorizacija rade prema specifikaciji.

## Ispitivanje sustava

### Pregled

Ovaj dokument opisuje E2E (end-to-end) testove za HomeFix aplikaciju. Testovi pokrivaju redovne slučajeve, rubne uvjete i nepostojeće funkcionalnosti.

**Alat za ispitivanje:** Playwright

**Broj test slučajeva:** 6 (4 glavna + 2 dodatna)

**Datum testiranja:** Siječanj 2026

## Test 01: Uspješna prijava s validnim podacima (Redovni slučaj)

### Funkcionalnost koju testiramo:

Prijava korisnika u sustav kroz web sučelje s validnim email i lozinkom.

### Ispitni slučaj 1: Uspješna prijava

#### Ulagni podaci:

- Email: `test@example.com`
- Lozinka: `password123`
- Test korisnik se automatski kreira prije testa u lokalnom Supabase-u

#### Koraci ispitivanja:

1. Otvoriti aplikaciju na `/login` stranici
2. Unijeti email adresu u polje "Email"
3. Unijeti lozinku u polje "Lozinka"
4. Kliknuti na gumb "Prijava se"
5. Provjeriti je li korisnik preusmjeren na `/dashboard` stranicu

#### Očekivani izlaz:

- Korisnik je uspješno preusmjeren na `/dashboard`
- URL sadrži `/dashboard`
- Nema poruka o grešci

#### Dobiveni rezultati:

- Test prolazi uspješno
- Korisnik je preusmjeren na `/dashboard`
- Screenshot: `test-results/screenshots/test1-successful-login.png`

#### Prolaz ispitivanja

## Test 02: Neuspješna prijava s nevažećim podacima (Rubni uvjet)

#### Funkcionalnost koju testiramo:

Reakcija sustava na pokušaj prijave s nevažećim podacima (nevažeći email ili lozinka).

#### Ispitni slučaj 1: Neuspješna prijava s nevažećim podacima

##### Ulagni podaci:

- Email: `invalid@example.com`
- Lozinka: `wrongpassword`
- Korisnik ne postoji u lokalnom Supabase-u

##### Koraci ispitivanja:

1. Otvoriti aplikaciju na `/login` stranici
2. Unijeti nevažeći email u polje "Email"
3. Unijeti nevažeću lozinku u polje "Lozinka"
4. Kliknuti na gumb "Prijava se"
5. Provjeriti je li prikazana poruka o grešci
6. Provjeriti da korisnik NIJE preusmjeren (ostaje na `/login`)

##### Očekivani izlaz:

- Poruka o grešci je prikazana (npr. "Invalid login credentials" ili "Pogrešno korisničko ime ili lozinka")
- Korisnik ostaje na `/login` stranici
- URL sadrži `/login`

##### Dobiveni rezultati:

- Test prolazi uspješno
- Poruka o grešci je prikazana

- Korisnik ostaje na /login stranici
- Screenshot: test-results/screenshots/test2-failed-login.png

**Prolaz ispitivanja**

---

## Test 03: Registracija s validnim podacima (Redovni slučaj)

### Funkcionalnost koju testiramo:

Registracija novog korisnika u sustav kroz web sučelje s validnim podacima.

---

### Ispitni slučaj 1: Uspješna registracija

#### Ulazni podaci:

- Ime: Ivan
- Prezime: Horvat
- Email: newuser@example.com
- Lozinka: password123
- Potvrda lozinke: password123
- Test korisnik se automatski briše nakon testa

#### Koraci ispitivanja:

1. Otvoriti aplikaciju na /register stranici
2. Unijeti ime u polje "Ime"
3. Unijeti prezime u polje "Prezime"
4. Unijeti email adresu u polje "Email"
5. Unijeti lozinku u polje "Lozinka"
6. Unijeti istu lozinku u polje "Potvrdi lozinku"
7. Kliknuti na gumb "Registriraj se"
8. Provjeriti je li korisnik preusmjeren na /register/choose-role stranicu

#### Očekivani izlaz:

- Korisnik je uspješno preusmjeren na /register/choose-role
- URL sadrži /register/choose-role
- Nema poruka o grešci

#### Dobiveni rezultati:

- Test prolazi uspješno
- Korisnik je preusmjeren na /register/choose-role
- Screenshot: test-results/screenshots/test3-successful-registration.png

**Prolaz ispitivanja**

---

## Ispitni slučaj 2: Registracija s nepodudarajućim lozinkama (Rubni uvjet)

### Ulazni podaci:

- Ime: Ivan
- Prezime: Horvat
- Email: test@example.com
- Lozinka: password123
- Potvrda lozinke: differentpassword (različita lozinka)

#### Koraci ispitivanja:

1. Otvoriti aplikaciju na /register stranici

2. Unijeti ime, prezime i email
3. Unijeti lozinku u polje "Lozinka"
4. Unijeti RAZLIČITU lozinku u polje "Potvrdi lozinku"
5. Kliknuti na gumb "Registriraj se"
6. Provjeriti je li prikazana poruka o grešci
7. Provjeriti da korisnik NIJE preusmjeren (ostaje na `/register`)

**Očekivani izlaz:**

- Poruka o grešci "Lozinke se ne podudaraju" je prikazana
- Korisnik ostaje na `/register` stranici
- URL sadrži `/register`

**Dobiveni rezultati:**

- Test prolazi uspješno
- Poruka "Lozinke se ne podudaraju" je prikazana
- Korisnik ostaje na `/register` stranici
- Screenshot: `test-results/screenshots/test3b-password-mismatch.png`

Prolaz ispitivanja

---

## Test 04: Pristup nepostojećoj ruti (Nepostojeća funkcionalnost)

**Funkcionalnost koju testiramo:**

Reakcija sustava na pokušaj pristupa nepostojećoj stranici ili ruti.

---

### Ispitni slučaj 1: Pristup nepostojećoj stranici

**Ulazni podaci:**

- URL: `/non-existent-page-that-does-not-exist-12345`
- HTTP metoda: GET

**Koraci ispitivanja:**

1. Navigirati na nepostojeću rutu `/non-existent-page-that-does-not-exist-12345`
2. Provjeriti HTTP status kod odgovora
3. Provjeriti da se prikazuje 404 stranica ili odgovarajuća poruka

**Očekivani izlaz:**

- HTTP status kod: `404 Not Found`
- Stranica prikazuje poruku o grešci (npr. "404", "not found", "nije pronađen")

**Dobiveni rezultati:**

- Test prolazi uspješno
- HTTP status kod je `404`
- Stranica prikazuje odgovarajuću poruku o grešci
- Screenshot: `test-results/screenshots/test4-404-page.png`

Prolaz ispitivanja

---

### Ispitni slučaj 2: Pristup nepostojećoj API ruti (Rubni uvjet)

**Ulazni podaci:**

- URL: `/api/non-existent-endpoint-xyz`
- HTTP metoda: GET

#### Koraci ispitivanja:

1. Navigirati na nepostojeću API rutu `/api/non-existent-endpoint-xyz`
2. Provjeriti HTTP status kod odgovora

#### Očekivani izlaz:

- HTTP status kod: `404 Not Found`

#### Dobiveni rezultati:

- ✓ Test prolazi uspješno
- HTTP status kod je `404`
- Screenshot: `test-results/screenshots/test4b-404-api.png`

✓ Prolaz ispitivanja

---

## Ukupni sažetak testiranja

**Ukupno testirano:** 6 ispitnih slučajeva

#### Kategorije testova:

- **Redovni slučajevi:** 2 testa (Test 01, Test 03.1)
- **Rubni uvjeti:** 3 testa (Test 02, Test 03.2, Test 04.2)
- **Nepostojeća funkcionalnost:** 1 test (Test 04.1)

#### Rezultati:

- ✓ Svi testovi prolaze uspješno: **6/6**
- Pokretanje testova: `npm run test:e2e`
- Testing framework: **Playwright**
- Browzeri testirani: Chromium, Firefox, WebKit

#### Datoteke testova:

- `e2e/tests/auth.spec.js` - Testovi za autentifikaciju (Test 01-03)
- `e2e/tests/navigation.spec.js` - Testovi za navigaciju (Test 04)

#### Generirani rezultati:

- Screenshotovi: `test-results/screenshots/`
- Video snimke: `test-results/` (samo za neuspješne testove)
- HTML izvještaj: `e2e/playwright-report/`
- JSON izvještaj: `e2e/test-results/results.json`

**Datum testiranja:** Siječanj 2026

#### Zaključak:

Svi testirani scenariji rade ispravno. Sustav ispravno reagira na:

- Uspješne i neuspješne pokušaje prijave
- Uspješnu registraciju i validaciju lozinki
- Pristup nepostojećim rutama s odgovarajućim HTTP status kodovima

**Broj otkrivenih grešaka:** 0

---

## Dodatne napomene

### Lokalni Supabase setup

Testovi koriste stvarni lokalni Supabase instance umjesto mockova. To omogućava:

- Realistično testiranje autentifikacije
- Testiranje s pravom bazom podataka
- Automatsko upravljanje test korisnicima

#### Preduvjeti:

1. Docker Desktop mora biti pokrenut
2. Lokalni Supabase mora biti pokrenut: `npm run supabase:start`
3. Supabase CLI mora biti instaliran

#### Upravljanje test korisnicima:

- Test korisnici se automatski kreiraju prije svakog testa
- Test korisnici se automatski brišu nakon svakog testa
- Global setup/teardown očisti sve test korisnike prije i nakon testova
- Test korisnici imaju emailove koji završavaju s `@example.com`

#### Screenshotovi i video

Playwright automatski generira:

- Screenshotove za svaki test (pohranjeni u `test-results/screenshots/`)
- Video snimke za neuspješne testove (pohranjeni u `test-results/`)

#### Pokretanje testova

```
# 1. Pokreni lokalni Supabase (ako nije već pokrenut)
npm run supabase:start

# 2. Pokreni E2E testove
npm run test:e2e

# Pokretanje s UI modom (interaktivno)
npm run test:e2e:ui

# Prikaz HTML izvještaja
npm run test:e2e:report
```

#### Dostupne Supabase komande:

```
npm run supabase:start  # Pokreni lokalni Supabase
npm run supabase:stop   # Zaustavi lokalni Supabase
npm run supabase:status # Provjeri status Supabase servisa
```

## Korištene tehnologije i alati

### 1. Programski jezici

- **JavaScript** - moderni ECMAScript standard

Osnovni programski jezik korišten u cijeloj aplikaciji. Omogućuje razvoj i klijentskog i serverskog dijela unutar istog ekosustava, što pojednostavljuje održavanje i dijeljenje logike.

- **TypeScript 5.9**

Nadogradnja JavaScripta s podrškom za statičko tipiziranje. Korišten radi bolje pouzdanosti koda, lakšeg refaktoriranja i ranog otkrivanja grešaka tijekom razvoja.

## 2. Radni okviri i biblioteke

- **Next.js 15**

React framework korišten za izgradnju full stack web aplikacije. Omogućuje Server Components i Server Actions, optimizirani routing i renderiranje te jednostavnu integraciju s backend logikom. Odabran je zbog performansi, skalabilnosti i modernog pristupa web aplikacija.

- **React 19**

UI biblioteka za izgradnju korisničkog sučelja temeljenog na komponentama. Omogućuje ponovnu upotrebu koda, jasnu strukturu aplikacije i učinkovito upravljanje stanjem aplikacije.

- **Tailwind CSS 4**

CSS framework koji koristi pristup temeljen na pomoćnim klasama za brzo i konzistentno stiliziranje korisničkog sučelja.

- **shadcn/ui**

Kolekcija React UI komponenti izgrađenih na osnovnim Radix UI komponentama. Odabrana zbog pristupačnosti, potpune kontrole nad stilovima i dobre integracije s Tailwind CSS-om.

- **Lucide React**

Biblioteka SVG ikona optimizirana za React. Korištena za konzistentan i lagan prikaz ikona unutar aplikacije.

## 3. Baza podataka

- **Supabase**

Supabase je platforma koja pruža backend funkcionalnosti kao uslugu, a koristi se za autentifikaciju korisnika, upravljanje bazom podataka i upravljanje pozivnicama. Odabrana zbog jednostavne integracije s Next.js aplikacijom, mogućnosti lokalnog razvoja (Supabase CLI) te integrirane baze podataka i autentifikacije.

## 4. Razvojni alati

- **Visual Studio Code**

Glavni IDE korišten za razvoj. Omogućuje velik broj dostupnih ekstenzija, Typescript podršku i dobru integraciju s Gitom.

- **Git 2.51**

Sustav za kontrolu verzija korišten za praćenje promjena koda i timsku suradnju.

- **Supabase CLI**

Alat za lokalni razvoj Supabase okruženja. Omogućuje pokretanje lokalne baze, autentifikacije i testiranje bez ovisnosti o produkcijskom okruženju.

- **Postman**

Korišten za testiranje i validaciju API endpointa tijekom razvoja.

## 5. Alati za ispitivanje

- **Jest 30**

Framework za unit testove. Testiranje se provodi izolirano nad serverskim akcijama, uz korištenje lažnih (mock) podataka za Supabase kako bi se provjerila ispravnost poslovne logike bez ovisnosti o vanjskim servisima. Dopušta brzo i pouzdano testiranje funkcionalnosti na niskoj razini.

- **Playwright 1.40**

Alat za end-to-end (E2E) testove. Alat se koristi za testiranje cijeloukupnog korisničkog tijeka kroz preglednik, simulaciju stvarnog ponašanja korisnika te provjeru autentifikacije uz korištenje lokalnog Supabase okruženja.

## 6. Alati za razmještaj i integracije

- **Stripe**

Platforma za obradu online plaćanja. Omogućuje sigurne i skalabilne financijske transakcije.

- **SendGrid**

Servis za slanje elektroničke pošte koji u projektu omogućuje sigurno i skalabilno slanje korisničkih pozivnica.

- **APDF**

Korišten je vanjski servis za generiranje PDF dokumenata koji omogućuje jednostavnu integraciju u aplikaciju. Servis je odabran zbog besplatnog plana i brze implementacije bez potrebe za složenom konfiguracijom.

## 7. Cloud platforma

- **Vercel**

Koristi se za hostanje i razmještaj web aplikacije razvijene u okviru Next.js. Platforma je odabrana zbog izvorne podrške za Next.js, automatskog razmještaja iz Git repozitorija te potpore za Server Components i Server Actions. Globalna CDN mreža omogućuje brzo učitavanje aplikacije te jednostavno upravljanje različitim okruženjima bez dodatne konfiguracije.

---

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

### 1. Instalacija

#### Preduvjeti

Prije instalacije aplikacije potrebno je imati instalirano sljedeće softvere:

- **Node.js** verzija 18 ili novija ([preuzimanje](#))
- **npm** ili **yarn** (dolazi s Node.js)
- **Git** za kloniranje repozitorija
- **Supabase račun** ([besplatno registriranje](#))
- **Stripe račun** ([besplatno registriranje](#)) - potrebno za funkcionalnost plaćanja

#### Preuzimanje izvornog koda

Klonirajte repozitorij s GitHub-a:

```
git clone https://github.com/svebaa/homefix.git
cd homefix
```

## Instalacija ovisnosti

Instalirajte sve potrebne npm pakete:

```
npm install
```

## 2. Postavke

### Konfiguracijske datoteke

HomeFix aplikacija koristi environment varijable za konfiguraciju. Kreirajte datoteku `.env.local` u root direktoriju projekta.

**Važno:** Datoteka `.env.local` je u `.gitignore` i neće biti commitana u Git repozitorij. Svaki developer mora kreirati svoj lokalni `.env.local` file.

### Primjer `.env.local` datoteke

```
# Supabase konfiguracija (obavezno)
NEXT_PUBLIC_SUPABASE_URL=your_supabase_project_url
NEXT_PUBLIC_SUPABASE_PUBLISHABLE_KEY=your_supabase_anon_key
SUPABASE_SERVICE_ROLE_KEY=your_supabase_service_role_key

# Stripe konfiguracija (obavezno za funkcionalnost plaćanja)
STRIPE_SECRET_KEY=your_stripe_secret_key
STRIPE_WEBHOOK_SECRET=your_stripe_webhook_secret

# aPDF konfiguracija (obavezno za generiranje PDF izvještaja)
APDF_API_KEY=your_apdf_api_key

# URL aplikacije (opcionalno, ali preporučeno za produkciju)
NEXT_PUBLIC_SITE_URL=https://your-app-domain.com
```

### Kako dobiti Supabase ključeve

- Prijavite se na [Supabase Dashboard](#)
- Odaberite svoj projekt (ili kreirajte novi)
- Idite na **Settings** → **API**
- Kopirajte:
  - Project URL** → `NEXT_PUBLIC_SUPABASE_URL`
  - anon/public key** → `NEXT_PUBLIC_SUPABASE_PUBLISHABLE_KEY`
  - service\_role key** → `SUPABASE_SERVICE_ROLE_KEY` (⚠ NE dijelite ovaj ključ!)

### Kako dobiti Stripe ključeve

- Prijavite se na [Stripe Dashboard](#)
- Idite na **Developers** → **API keys**
- Kopirajte:
  - Secret key** → `STRIPE_SECRET_KEY`
  - Za webhook secret, kreirajte endpoint u Stripe Dashboardu i kopirajte signing secret

## Kako dobiti aPDF API ključ

1. Prijavite se na [aPDF.io](#) ili registrirajte novi račun
2. Idite na **API Keys** ili **Settings** sekciju
3. Kopirajte **API Key** → APDF\_API\_KEY

**Napomena:** aPDF API ključ je potreban za generiranje mjesecnih PDF izvještaja kvarova. Bez ovog ključa, funkcionalnost generiranja PDF izvještaja neće raditi.

## Postavke baze podataka

HomeFix koristi Supabase kao bazu podataka i Backend-as-a-Service. Migracije se nalaze u `supabase/migrations/` direktoriju.

### Za lokalni razvoj:

- Možete koristiti lokalni Supabase instance: `npm run supabase:start`
- Provjerite status: `npm run supabase:status`

### Za produkciju:

- Migracije se primjenjuju automatski kroz Supabase Dashboard ili CLI
- U Supabase Dashboardu idite na **SQL Editor** i pokrenite migracije iz `supabase/migrations/` direktorija

## 3. Pokretanje aplikacije

### Razvojno okruženje

Pokrenite razvojni server s Turbopack podrškom:

```
npm run dev
```

Aplikacija će biti dostupna na `http://localhost:3000`.

### Producčijsko okruženje

#### Lokalno testiranje produkcijske verzije:

1. Prevođenje aplikacije:

```
npm run build
```

2. Pokretanje poslužitelja:

```
npm start
```

Aplikacija će biti dostupna na `http://localhost:3000`.

### Provjera rada

Nakon pokretanja aplikacije, provjerite:

- Aplikacija je dostupna na `http://localhost:3000`
- Možete pristupiti stranici za prijavu ( `/login` )
- Možete pristupiti stranici za registraciju ( `/register` )
- Supabase veza radi ispravno (provjerite u browser konzoli)

## 4. Upute za administratore

## Pristup administratorskom sučelju

- **URL za admin panel:** /admin
- **Pristup:** Samo korisnici s ulogom ADMIN mogu pristupiti administratorskom sučelju
- **Administratorske funkcionalnosti:**
  - Upravljanje korisnicima ( /admin/users )
  - Upravljanje članstvom ( /admin/membership )

**Napomena:** Potrebno je kreirati korisnika s ADMIN ulogom u Supabase bazi podataka u tablici profile (polje role = 'ADMIN' ).

## Redovito održavanje

### Arhiviranje baze podataka:

- Supabase automatski kreira backup-ove baze podataka
- U Supabase Dashboardu idite na **Database** → **Backups** za pristup backup-ovima

### Pregled logova:

- **Lokalno:** Logovi se prikazuju u terminalu gdje je pokrenut npm run dev
- **Vercel produkcija:** Logovi su dostupni u Vercel Dashboardu → **Deployments** → odaberi deployment → **Logs**
- **Supabase:** Logovi su dostupni u Supabase Dashboardu → **Logs** → **API Logs**

### Ažuriranje aplikacije na Vercelu:

- Vercel automatski deploys aplikaciju kada se pusha kod na povezani Git granu (npr. main ili production )
- Za ručni deploy koristite Vercel CLI:

```
vercel --prod
```

### Ažuriranje lokalno:

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

## Rješavanje problema

### Aplikacija se ne pokreće:

- Provjerite da su sve environment varijable ispravno postavljene u .env.local
- Provjerite da je Node.js verzija 18 ili novija: node --version
- Provjerite da su sve ovisnosti instalirane: npm install
- Provjerite logove za specifične greške

### Supabase veza ne radi:

- Provjerite da su NEXT\_PUBLIC\_SUPABASE\_URL i NEXT\_PUBLIC\_SUPABASE\_PUBLISHABLE\_KEY ispravno postavljeni
- Provjerite da je Supabase projekt aktivan u Supabase Dashboardu
- Provjerite network tab u browser developer tools za detaljne greške

### Stripe webhook ne radi:

- Provjerite da je STRIPE\_WEBHOOK\_SECRET ispravno postavljen
- Provjerite da je webhook endpoint konfiguriran u Stripe Dashboardu

- Provjerite da webhook endpoint pokazuje na ispravan URL (npr. `https://your-app.com/api/webhooks/stripe` )

#### PDF izvještaji se ne generiraju:

- Provjerite da je `APDF_API_KEY` ispravno postavljen
- Provjerite da je aPDF API ključ aktivan i valjan
- Provjerite da imate dovoljno kredita na aPDF računu (ako je potrebno)
- Provjerite logove za detaljne greške prilikom generiranja PDF-a

#### Admin panel ne radi:

- Provjerite da korisnik ima `ADMIN` ulogu u Supabase bazi podataka
- Provjerite da je korisnik prijavljen
- Provjerite browser konzolu za greške

## 5. Deployment na Vercel platformu

Vercel je preporučena cloud platforma za Next.js aplikacije i omogućava jednostavan deployment HomeFix aplikacije.

### Preparacija repozitorija

#### 1. Osigurajte da je kod na Git repozitoriju:

- GitHub, GitLab ili Bitbucket repozitorij

#### 2. Provjerite da `package.json` sadrži ispravne build skripte:

- `"build": "next build --turbopack"` - već postoji u projektu
- `"start": "next start"` - već postoji u projektu

### Postavljanje na Vercel

#### 1. Prijavite se na Vercel:

- Idite na [vercel.com](https://vercel.com) i prijavite se s GitHub/GitLab/Bitbucket računom

#### 2. Dodajte novi projekt:

- Kliknite na **"Add New Project"**
- Odaberite Git repozitorij s HomeFix aplikacijom
- Vercel će automatski detektirati Next.js framework

#### 3. Konfigurirajte postavke projekta:

- Framework Preset:** Next.js (automatski detektiran)
- Root Directory:** `./` (ako je projekt u root direktoriju)
- Build Command:** `npm run build` (automatski)
- Output Directory:** `.next` (automatski)
- Install Command:** `npm install` (automatski)

#### 4. Dodajte environment varijable:

- U Vercel Dashboardu idite na **Settings → Environment Variables**
- Dodajte sve potrebne varijable:

```
NEXT_PUBLIC_SUPABASE_URL
NEXT_PUBLIC_SUPABASE_PUBLISHABLE_KEY
SUPABASE_SERVICE_ROLE_KEY
STRIPE_SECRET_KEY
STRIPE_WEBHOOK_SECRET
```

```
APDF_API_KEY  
NEXT_PUBLIC_SITE_URL
```

- **Važno:** Postavite varijable za sva okruženja (Production, Preview, Development)

## 5. Konfigurirajte Stripe webhook:

- U Stripe Dashboardu kreirajte webhook endpoint
- URL: <https://your-app.vercel.app/api/webhooks/stripe>
- Odaberite događaje koje želite primati
- Kopirajte webhook signing secret u Vercel environment varijable

## Pokretanje aplikacije

Nakon što kliknete "**Deploy**", Vercel će:

1. Automatski preuzeti kod iz Git repozitorija
2. Instalirati ovisnosti ( `npm install` )
3. Buildati aplikaciju ( `npm run build` )
4. Deployati aplikaciju na Vercel servere

Aplikacija će biti dostupna na generiranom URL-u (npr. <https://homefix-xyz.vercel.app> ).

**Napomena:** Za produkcijski deployment, postavite custom domenu u Vercel Dashboardu → **Settings** → **Domains**.

## Automatski deployment

Vercel automatski deploys aplikaciju kada:

- Pushate kod na `main` granu (production deployment)
- Pushate kod na bilo koju granu (preview deployment)
- Kreirate Pull Request (preview deployment)

## Vercel CLI (alternativa)

Možete koristiti Vercel CLI za deployment:

```
# Instalacija Vercel CLI
```

```
npm i -g vercel
```

```
# Login
```

```
vercel login
```

```
# Deploy
```

```
vercel
```

```
# Production deploy
```

```
vercel --prod
```

## 6. Pristup aplikaciji na javnom poslužitelju

### Pristup aplikaciji

Nakon uspješnog deploymenta na Vercel, aplikacija je javno dostupna putem internetskog preglednika.

**Korisnici mogu pristupiti aplikaciji na:**

- **Produkcijski URL:** URL generiran od Vercela (npr. <https://homefix-xyz.vercel.app> )
- **Custom domena:** Ako je konfigurirana (npr. <https://homefix.com> )

## Funkcionalnosti dostupne korisnicima

- **Registracija:** `/register` - korisnici se mogu registrirati kao:
  - Stanar (TENANT)
  - Predstavnik zgrade (REPRESENTATIVE)
  - Majstor (CONTRACTOR)
- **Prijava:** `/login` - korisnici se mogu prijaviti s email/lozinkom ili Google računom
- **Dashboard:** `/dashboard` - glavna stranica nakon prijave
- **Upravljanje kvarovima:** `/tickets` - ovisno o ulozi korisnika
- **Profil:** `/profile` - upravljanje korisničkim profilom

## Pristup administratorskom sučelju

URL: `https://your-app-domain.com/admin`

### Zahtjevi:

- Korisnik mora biti prijavljen
- Korisnik mora imati `ADMIN` ulogu u bazi podataka

### Administratorske stranice:

- `/admin/users` - upravljanje korisnicima
- `/admin/membership` - upravljanje članstvom

**Napomena:** Početni admin korisnik mora biti kreiran direktno u Supabase bazi podataka (postaviti `role = 'ADMIN'` u tablici `profile`).

## Ograničenja

### Vercel Free Plan:

- **Bandwidth:** 100 GB/mjesec
- **Build minutes:** 6000 minuta/mjesec
- **Serverless function execution:** 100 GB-hours/mjesec
- **Edge requests:** Neograničeno
- **Custom domains:** Dozvoljeno

### Supabase Free Plan:

- **Database size:** 500 MB
- **Bandwidth:** 5 GB/mjesec
- **API requests:** 50,000/mjesec
- **Auth users:** Neograničeno
- **Storage:** 1 GB

### Stripe:

- Provjerite trenutne limite u Stripe Dashboardu za svoj plan

### Preporuke:

- Za produkcijsku upotrebu razmotrite upgrade na Vercel Pro plan
- Za veće baze podataka razmotrite Supabase Pro plan
- Redovito pratite usage metrics u Vercel i Supabase Dashboardima

## Sigurnosne napomene

- **Nikad ne commitajte `.env.local` datoteku u Git**
- **Nikad ne dijelite `SUPABASE_SERVICE_ROLE_KEY` javno**
- **Nikad ne dijelite `STRIPE_SECRET_KEY` javno**
- **Nikad ne dijelite `APDF_API_KEY` javno**

- Koristite HTTPS za sve produkcijske deploymente (Vercel automatski osigurava HTTPS)
  - Redovito ažurirajte dependencies: `npm audit` i `npm update`
  - Provjerite Supabase RLS (Row Level Security) policies za sigurnost podataka
- 

## Osvrt na izradu projektnog zadatka

Izrada projektnog zadatka odvijala se tijekom zimskog semestra u timu od šest članova. Projekt HomeFix obuhvaćao je cjelokupan razvoj web aplikacije, od početne analize zahtjeva do implementacije i testiranja, što je zahtijevalo dobru organizaciju i kontinuiranu suradnju unutar tima. Rad na projektu bio je izazovan zbog opsega funkcionalnosti i potrebe za usklađivanjem rada više članova, ali je istovremeno omogućio stjecanje vrijednog iskustva u razvoju složenijeg softverskog rješenja.

## Tehnički izazovi

Tijekom izrade projekta prepoznati su brojni tehnički izazovi, među kojima se ističe rad s bazom podataka te integracija vanjskih servisa. Posebnu složenost predstavljalo je pravilno modeliranje podataka, upravljanje odnosa između entiteta te osiguravanje konzistentnosti podataka. Izazovi su rješavani korištenjem službene dokumentacije, modularnim pristupom razvoju te međusobnom suradnjom članova tima. Dio rješenja dodatno je unaprijeđen kroz iterativni razvoj i testiranje, čime je osigurana stabilnost i ispravnost aplikacije.

## Stečena i potrebna znanja

Rad na projektu HomeFix omogućio je stjecanje praktičnih znanja iz razvoja web aplikacija i primjene suvremenih tehnologija. Kroz izradu aplikacije primijenjena su teorijska znanja u radu s bazom podataka, autentifikacijom i aplikacijskom logikom, uz razumijevanje cijelog razvojnog procesa. Istovremeno su unaprijeđene vještine timskog rada kroz suradnju i zajedničko rješavanje problema.

Za brže i kvalitetnije ostvarenje projekta posebno bi koristila naprednija znanja iz rada s bazom podataka te prethodno iskustvo s novim korištenim tehnologijama i servisima. Dodatna iskustva u planiranju većih projekata i upravljanju vremenom također bi doprinijela učinkovitijoj realizaciji.

## Nastavak rada

Za nastavak razvoja aplikacije bilo bi potrebno osmisliti i implementirati dodatne funkcionalnosti poput naprednijih statističkih pregleda, automatskih obavijesti korisnicima te mogućnosti komunikacije unutar same aplikacije između stanara, predstavnika i majstora. Također bi se mogla razviti mobilna aplikacija kako bi se korisnicima omogućio lakši i brži pristup sustavu.

## Zaključak

Sve planirane funkcionalnosti sustava uspješno su implementirane i međusobno povezane u funkcionalnu cjelinu. Ostvarena aplikacija ispunjava definirane zahtjeve te pruža učinkovito rješenje za digitalno upravljanje prijavama kvarova unutar stambenih zgrada.

## Opće reference o programskom inženjerstvu

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>

## Web razvoj i frontend tehnologije

7. Next.js Documentation, <https://nextjs.org/docs>
8. React Documentation, <https://react.dev/>
9. React Team, "React: The Complete Guide", Meta Open Source, <https://react.dev/learn>
10. Tailwind CSS Documentation, <https://tailwindcss.com/docs>
11. shadcn/ui Documentation, <https://ui.shadcn.com/>
12. Radix UI Documentation, <https://www.radix-ui.com/>
13. MDN Web Docs - JavaScript, <https://developer.mozilla.org/en-US/docs/Web/JavaScript>
14. MDN Web Docs - HTML, <https://developer.mozilla.org/en-US/docs/Web/HTML>
15. MDN Web Docs - CSS, <https://developer.mozilla.org/en-US/docs/Web/CSS>

## Backend i baza podataka

16. Supabase Documentation, <https://supabase.com/docs>
17. PostgreSQL Documentation, <https://www.postgresql.org/docs/>
18. REST API Design Best Practices, <https://restfulapi.net/>
19. J. Richardson, L. Ruby, "RESTful Web Services", O'Reilly Media, 2007.

## Autentifikacija i sigurnost

20. OWASP Top 10 Web Application Security Risks, <https://owasp.org/www-project-top-ten/>
21. Web Authentication API (WebAuthn), <https://www.w3.org/TR/webauthn/>
22. JWT.io - JSON Web Token Introduction, <https://jwt.io/introduction>

## Testiranje

23. Jest Documentation, <https://jestjs.io/docs/getting-started>
24. Playwright Documentation, <https://playwright.dev/>
25. Testing Library Documentation, <https://testing-library.com/>
26. K. C. Dodds, "Testing JavaScript", <https://testingjavascript.com/>

## UI/UX dizajn

27. Nielsen Norman Group - Usability Guidelines, <https://www.nngroup.com/articles/>
28. Material Design Guidelines, <https://m3.material.io/>
29. Web Content Accessibility Guidelines (WCAG), <https://www.w3.org/WAI/WCAG21/quickref/>

## Alati za razvoj

30. Git Documentation, <https://git-scm.com/doc>
31. GitHub Documentation, <https://docs.github.com/>
32. Node.js Documentation, <https://nodejs.org/docs/>
33. npm Documentation, <https://docs.npmjs.com/>

## Dodatne reference

34. Stripe Documentation, <https://stripe.com/docs>
35. Lucide Icons, <https://lucide.dev/>
36. date-fns Documentation, <https://date-fns.org/>
37. Web.dev - Best Practices for Web Development, <https://web.dev/>

Rev.	Opis promjene/dodataka	Autori	Datum
0.1	Napravljen predložak	Vasić	9.10.2025.
0.2	Napisan opis projektnog zadatka	Maslać	21.10.2025.

0.3	Napisani funkcionalni i nefunkcionalni zahtjevi	Maslać	21.10.2025.
0.3.1.	Male promjene opisa i analize zahtjeva	Vasić	21.10.2025.
0.3.2.	Uređenje zahtjeva i dodavanje dionika	Maslać	3.11.2025.
0.4	Specifikacija zahtjeva sustava	Maslać	4.11.2025.
0.5	Dodavanje opisa baze podataka	Perković	6.11.2025.
0.6	Organizacija i opis arhitekture i dizajna sustava	Maslać	10.11.2025.
0.6.1	Ispravak korištenih tehnologija u arhitekturi sustava	Vasić	11.11.2025.
0.6.2	Dodatno uređen opis arhitekture	Maslać	11.11.2025.
0.7	Napisan zapisnik prva tri sastanka	Klarić	12.11.2025.
0.8	Uređena specifikacija zahtjeva sustava	Maslać	13.11. i 14.11.2025.
0.9	Dodani UML dijagrami stanja i aktivnosti	Maslać	12.1.2026.
1.0	Ispitivanje programskog rješenja	Vasić	12.1.2026.
1.1	Arhitektura komponenata i razmještaja	Maslać	13.1.2025.
1.2	Tehnologije za implementaciju aplikacije	Klarić	16.1.2026.
1.3	Zaključak i budući rad	Klarić	16.1.2026.
1.4	Upute za puštanje u pogon	Vasić	19.1.2026.
1.5	Popis literature	Vasić	19.1.2026.
1.6	Ažurirani dijagrami obrazaca uporabe	Maslać	21.1.2026.
1.7	Završeno poglavlje prikaz aktivnosti grupe	Klarić i Perković	22.1.2026.
1.8	Izbrisani tekst predloška iz dokumentacije	Klarić i Perković	22.1.2026.

## Dnevnik sastajanja

### 1. sastanak

- Datum: 8. listopada 2025.
- Prisustvovali: Jan Klasić, Gabriela Perković, Svebor Vasić, Marko Maslać, Sara Klarić, Lovro Milišić
- Teme sastanka:
  - uspostavljeni WhatsApp grupa tima
  - napravljen projektni repozitorij, dodani članovi tima
  - odabir tehnologija

### 2. sastanak

- Datum: 15. listopada 2025.
- Prisustvovali: Jan Klasić, Gabriela Perković, Svebor Vasić, Marko Maslać, Sara Klarić, Lovro Milišić
- Teme sastanka:
  - odabir imena tima - Bumbari
  - postavljeni zadaci timu:
    - modeliranje baze podataka

- *oauth2 google auth*
- *kreiranje procesa registracije*
- *prvo i drugo poglavlje dokumentacije*

#### 3. sastanak

- Datum: 22. listopada 2025.
- Prisustvovali: Jan Klasić, Gabriela Perković, Svebor Vasić, Marko Maslać, Sara Klarić, Lovro Milišić
- Teme sastanka:
  - *diskusija o trenutnim nejasnoćama i pitanjima*
  - *rasprava o napravljenom i funkcijskim zahtjevima*

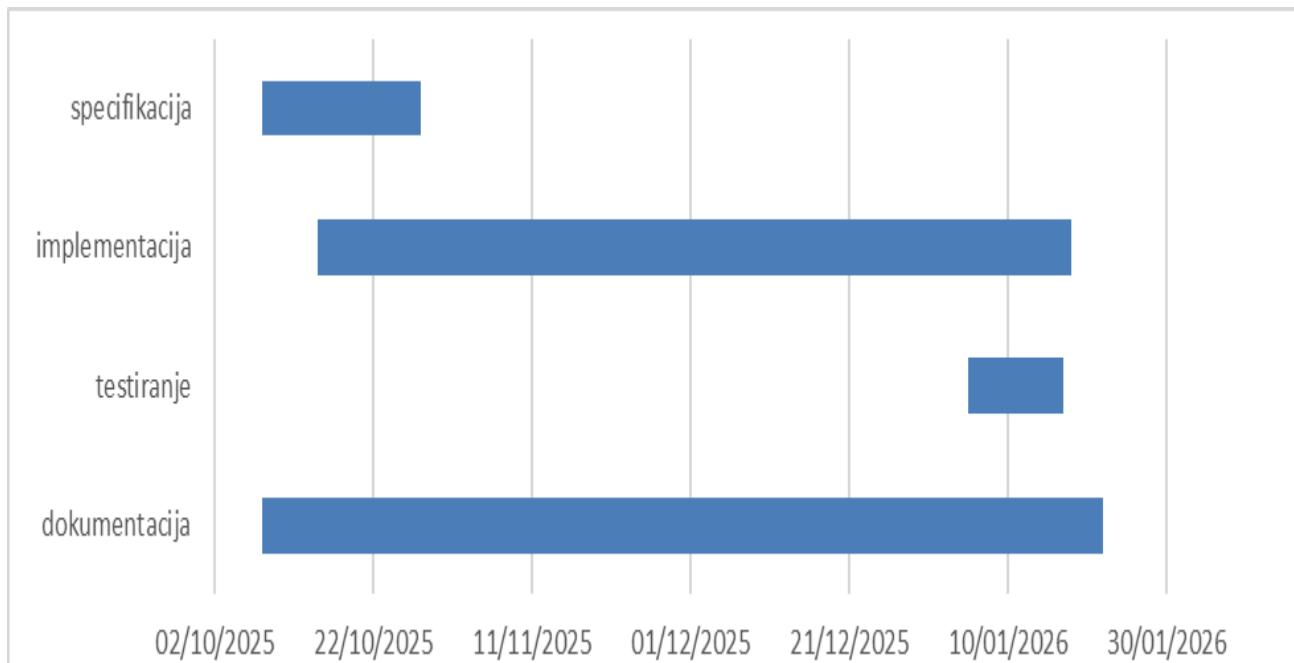
#### 4. sastanak

- Datum: 3. prosinca 2025.
- Prisustvovali: Jan Klasić, Gabriela Perković, Svebor Vasić, Marko Maslać, Sara Klarić, Lovro Milišić
- Teme sastanka:
  - *diskusija o izgledima platformi po ulogama*
  - *dodijeljeni zadaci za izradu stranica za pregled kvarova predstavnika stanara, stanara i majstora*

#### 5. sastanak

- Datum: 7. siječnja 2026.
- Prisustvovali: Jan Klasić, Gabriela Perković, Svebor Vasić, Marko Maslać, Sara Klarić, Lovro Milišić
- Teme sastanka:
  - *diskusija o završnim poboljšanjima*
  - *dodijeljeni zadaci za dotjerivanje aplikacije*

## Plan rada



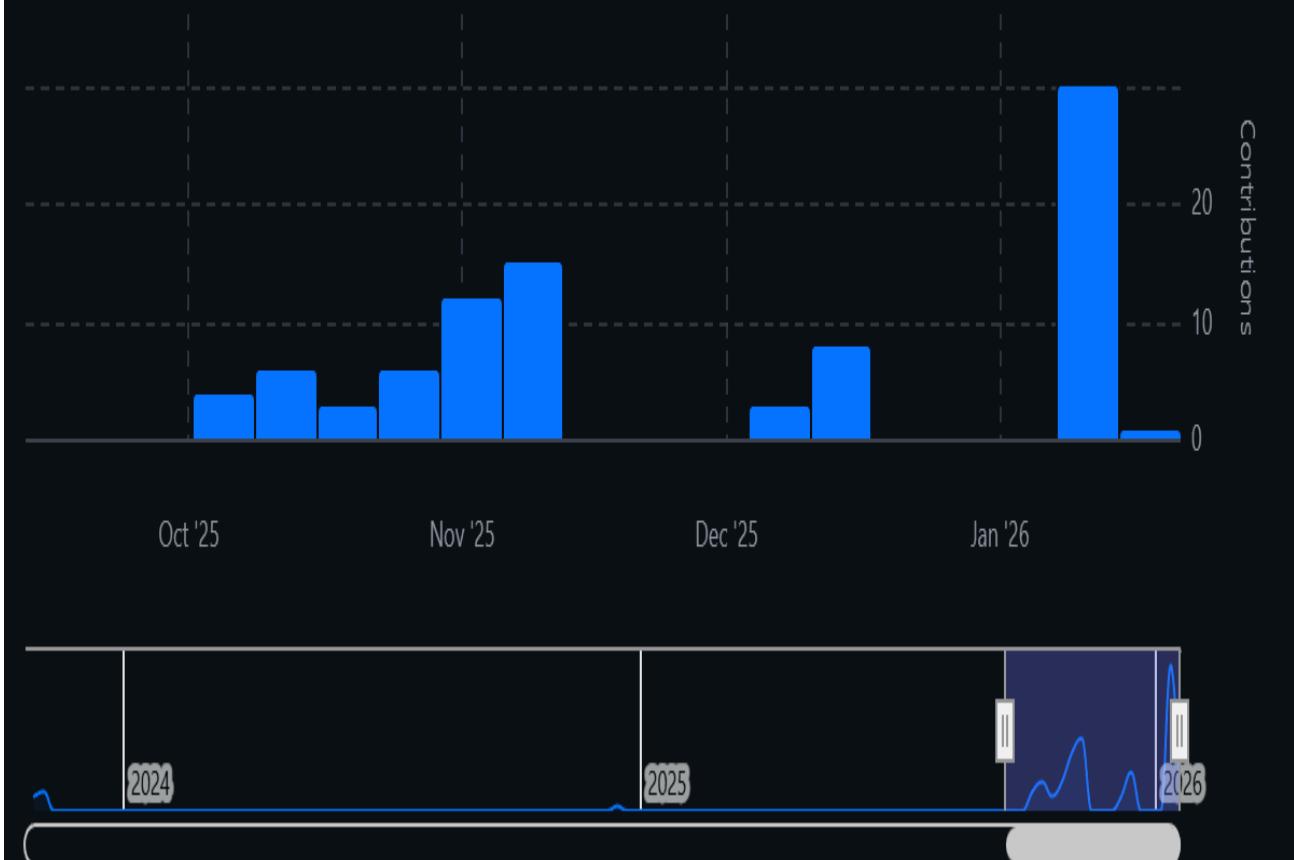
## Tablica aktivnosti

Aktivnost	Svebor Vasić	Gabriela Perković	Sara Klarić	Marko Maslać	Lovro Milišić	Jan Klasić
Upravljanje projektom	20					
Opis projektnog zadatka	1			5		
Funkcionalni zahtjevi	2	2	2	12	2	2
Opis pojedinih obrazaca				5		
Dijagram obrazaca				5		
Sekvencijski dijagrami				5		
Opis ostalih zahtjeva				5		
Arhitektura i dizajn sustava	2	2	2	5	2	2
Baza podataka	10	20	20	2	2	2
Dijagram razreda				5		
Dijagram stanja				5		
Dijagram aktivnosti				5		
Dijagram komponenti				5		
Korištene tehnologije i alati			5			
Ispitivanje programskog rješenja	10					
Dijagram razmještaja				5		
Upute za puštanje u pogon	5					
Dnevnik sastajanja		1	1			
Zaključak i budući rad			5			
Popis literature	1					
Izrada prezentacije		2	2			
Učenje novih tehnologija	5	10	10	2	20	15
Implementacija klijentske strane	20	25	20		25	20
Implementacije poslužiteljske strane	20	25	20		25	20
Postavljanje na server	10					

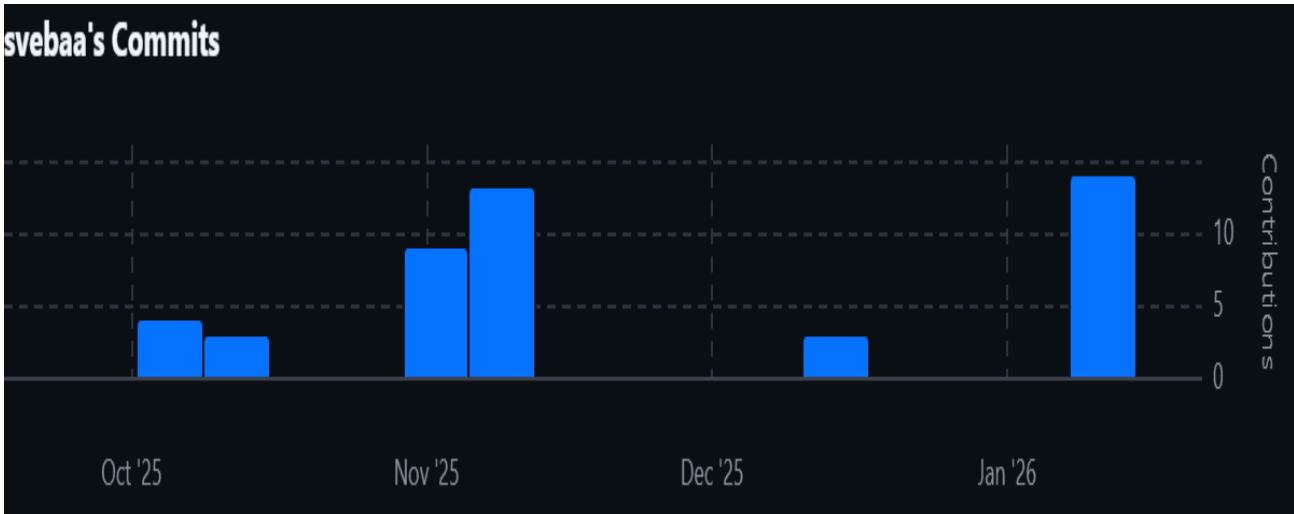
## Dijagram pregleda promjena

## Commits over time

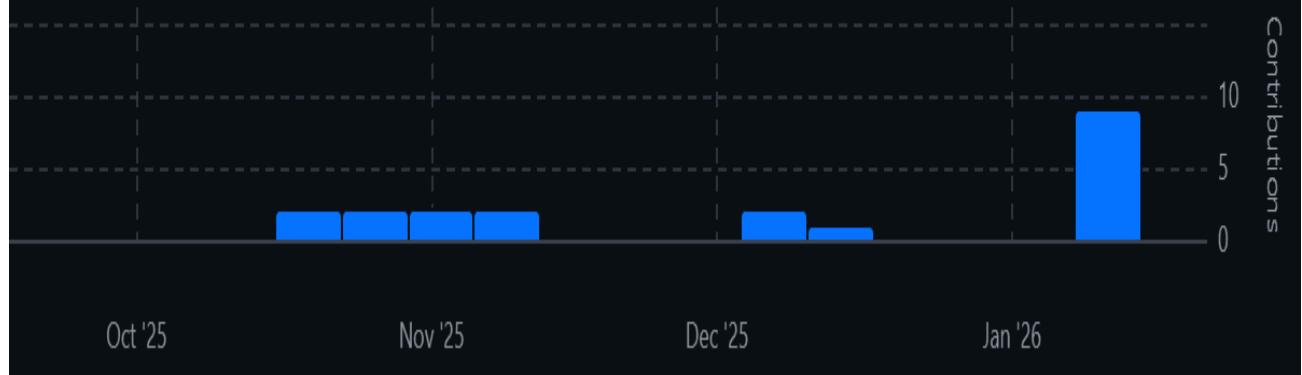
Weekly from 16. ruj 2025. to 18. sij 2026.



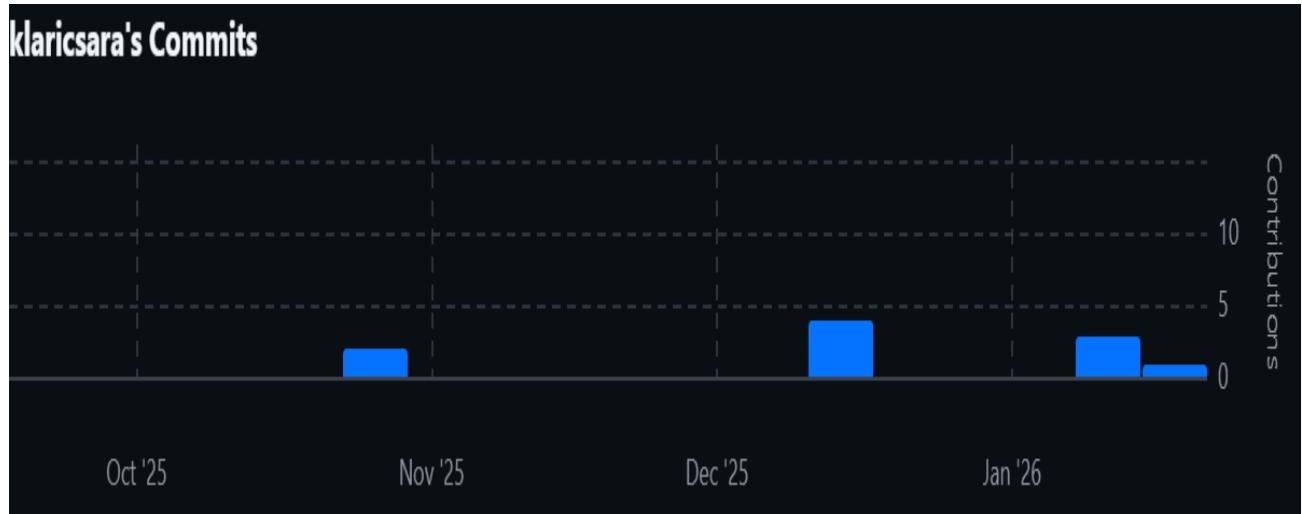
## svebaa's Commits



### gabrielaperkovic's Commits



### klaricsara's Commits



## Imilisic's Commits



## jk55976's Commits



## Kjučni izazovi i rješenja

Tijekom izrade projekta susreli smo se s različitim tehničkim izazovima, poput rada s bazom podataka i povezivanja s vanjskim servisima, kao i kašnjenja u razvoju zbog neusklađenosti u raspodjeli zadataka te tehničkih problema prilikom integracije pojedinih funkcionalnosti. Te smo izazove riješili boljom organizacijom rada, redovitom komunikacijom unutar tima i dodatnim istraživanjem potrebnih tehnologija. Kroz projekt smo stekli praktična znanja iz web razvoja, upravljanja bazama podataka i timskog rada. Iz tih situacija naučili smo važnost dobrog planiranja, fleksibilnosti i timske suradnje, što je značajno doprinijelo učinkovitijem radu i kvalitetnijem konačnom rješenju. Projekt ima potencijal za daljnju nadogradnju dodavanjem novih funkcionalnosti i razvojem mobilne aplikacije, a kao rezultat razvijeno je učinkovito rješenje za prijavu i upravljanje kvarovima.