SUSTAV ZA UPRAVLJANJE PROCESOM ODRŽAVANJA STAMBENE ZGRADE

Barbarić, Nina

VISOKO UČILIŠTE ALGEBRA

ZAVRŠNI RAD

SUSTAV ZA UPRAVLJANJE PROCESOM ODRŽAVANJA STAMBENE ZGRADE

Nina Barbarić

Zagreb, veljača 2023.

„Pod punom odgovornošću pismeno potvrđujem da je ovo moj autorski rad čiji niti jedan dio nije nastao kopiranjem ili plagiranjem tuđeg sadržaja. Prilikom izrade rada koristila sam tuđe materijale navedene u popisu literature, ali nisam kopirala niti jedan njihov dio, osim citata za koje sam navela autora i izvor, te ih jasno označila znakovima navodnika. U slučaju da se u bilo kojem trenutku dokaže suprotno, spremna sam snositi sve posljedice uključivo i poništenje javne isprave stečene dijelom i na temelju ovoga rada“.

U Zagrebu, 28.3.2023.

Predgovor

Od srca zahvaljujem svojem mentoru, profesoru Danielu Bele, koji me je uveo u svijet informatike, nadahnjujući me svojom posvećenošću struci te nesebičnim podređivanjem i svog slobodnog vremena mom obrazovanju i obrazovanju mojih kolega. Zahvaljujem što nas je ohrabrivao i motivirao da predano radimo i svladavamo tajne zanata. Posebno sam zahvalna na pomoći, savjetima i cjelokupnom angažmanu oko mog završnog rada koji su doveli do šireg i kvalitetnijeg sagledavanja obrađene teme i u socijalnom i u informatičkom smislu.

Također se zahvaljujem i ostalim profesorima na znanju koje su nam prenijeli, na prihvaćanju studenata kao hvalevrijednih budućih kolega, na kontinuiranom bodrenju da se krene dalje i dostigne više.

Zahvaljujem se i kolegama i prijateljima radi kojih je moje školovanje proteklo u toploj atmosferi punoj podrške i prijateljske bliskosti.

Na kraju zahvaljujem i svojoj obitelji, naročito mami i tati, koji su bez zadrške, diskretno iz pozadine, pratili i podupirali moje školovanje i izlazili u susret mojim željama i potrebama.

Prilikom uvezivanja rada, Umjesto ove stranice ne zaboravite umetnuti original potvrde o prihvaćanju teme završnog rada kojeg ste preuzeli u studentskoj referadi

Sažetak

Završnim radom prikazuju se koristi i načini digitalizacije procesa održavanja zgrada. Za svaku vrstu korisnika (stanar, predstavnik stanara i upravitelj zgrade) predlažu se i opisuju rješenja koja poboljšavaju i olakšavaju život u zgradi. Stanarima se omogućava proaktivno praćenje aktivnosti zgrade, sudjelovanje u donošenju odluka te pravedna podjela troškova. Predstavnicima stanara i upraviteljima zgrada se olakšava proces vođenja zgrade, komunikacija sa stanarima, ažuriranje i praćenje troškova zgrade i kretanja, tj fluktuacije stanara.

Ključne riječi: aplikacija, stanari, predstavnik stanara, upravitelj zgrade, zgrada, digitalizacija, održavanje zgrada.

Abstract

The final paper presents the benefits and methods of digitalizing the building maintenance process. For each type of user (tenant, tenant representative and building manager), solutions, which improve and facilitate life in the building, are proposed and described. The system enables tenants to monitor the activities of the, participate in decision-making, and share costs fairly. It also facilitates the process of managing the building for tenant representatives and building managers, solves the problem of communication between tenant representatives and building managers, as well as updating and monitoring costs and tenants.

Keywords: application, tennants, tenant representative, building manager, building, digitalization, building management.

Sadržaj

1. Uvod

Tema završnog rada je izrada sustava koji bi digitalizirao praćenje aktivnosti stambenih zgrada (ali i cijelog stambenog fonda), omogućio ažurnu evidenciju stanara te pravednu podjelu zajedničkih troškova zgrade. U skladu s važećom regulativom detaljno se objašnjavaju vrste korisnika aplikacije, korištena arhitektura za razvoj aplikacije, odnosno njen značaj za aplikaciju, verzioniranje koda aplikacije, sigurnosne razine i popis funkcionalnosti aplikacije s objašnjenjima.

Završni rad ima 10 poglavlja. Prva poglavlja fokusiraju se na organizaciju održavanja zgrada, sudionike procesa i načine njihova djelovanja.

Središnja poglavlja opisuju arhitekturu aplikacije, načine čuvanja koda i sigurnosne procese za kontrolu pristupa aplikaciji.

Osmo i deveto poglavlje odnose se na funkcionalne i nefunkcionalne zahtjeve i pružaju uvid u mogućnosti aplikacije.

Posljednje je poglavlje osvrt na aplikaciju, s kratkom analizom ostvarenih ideja i mogućnostima nadogradnje aplikacije.

Svrha ovog završnog rada je olakšavanje iskazivanja svakodnevnih potreba stanara, predstavnika stanara, upravitelja zgrade i ostvarivanje transparentosti svih podataka i aktivnosti pojedine stambene zgrade.

2. Problem komunikacije suvlasnika i sustanara u zgradama i njegovo rješenje

U ovom poglavlju predstavljeni su problemi trenutnog načina održavanja stambenih zgrada i života u njima te se predlaže i opisuje novi način rješavanja problema u procesu održavanja zgrada koji bi olakšao sustanarima praćenje i proaktivno sudjelovanje u tom procesu, na dobrobit zajednice zgrade.

2.1. Opis problema

Upravljanje zgradama u prošlim vremenima praktično nije postojalo. Nakon velike reforme stambene politike na kraju 20. stoljeća stanari su postali vlasnici stanova i zgrada [1]. Zakonom je omogućena prodaja stanova u društvenom vlasništvu i vlasništvu poduzeća nad kojima je izvršena pretvorba vlasništva. Nažalost, reformu imovinsko pravnih odnosa nije adekvatno popratila regulativa koja bi osvijestila suvlasnike o novonastaloj situaciji i njihovim obavezama. Zgrade su dobile svog upravitelja, tj. firmu koja radi popravke i brine o zgradi, ali komunikacija između suvlasnika i upravitelja, zamišljena putem predstavnika stanara, nije dovoljno definirana niti obavezujuća za stanare. Trenutno, održavanje zgrade svodi se na rješavanje hitnih intervencija. Obaviještenost stanara o onome što se dešava sa zgradom je nedostatna, evidencije nedostupne, sredstva koja se ulažu u održavanje su premala i stambeni fond propada. Dodatni problem je povećana mobilnost ljudi koja pridonosi ravnodušnosti stanara glede zajedničkih potreba zgrade. Naime, stanarima je stan, često, privremeno boravište te ne smatraju potrebnim brinuti ni o stanu, niti o zgradi. Zajednica stanara kao izdvojene interesne grupe praktično ne postoji.

2.2. Trenutna rješenja i njihovi nedostaci

Sadašnja rješenja, sastanci i oglasne ploče, ne mogu animirati suvlasnike i sustanare na veći angažman. Stanari pojedinih zgrada samoinicijativno se organiziraju u grupe na WhatsApp-u i neformalno komuniciraju. Međutim, formalna veza s predstavnikom stanara i upraviteljem ne postoji pa stanari nemaju uvid u aktivnosti oko održavanja zgrade, ne znaju

probleme, nemaju uvid u troškove održavanja i gotovo da nemaju utjecaj na odluke o trošenju sredstava pričuve.

2.3. Prijedlog sustava kao rješenje navedenog problema

Sustav koji bi digitalizirao praćenje aktivnosti zgrade (ali i cijelog stambenog fonda) omogućio bi ažurnu evidenciju stanara te pravednu podjelu troškova. Suvlasnici, odnosno stanari svake pojedine zgrade, u svakom bi trenutku, tj. kada kome odgovara, mogli pregledati rad svoga upravitelja, predstavnika stanara te predlagati, komentirati i sudjelovati u aktivnostima što se poduzimaju u vezi zgrade.

Digitalizacija sustava, putem aplikacije, znatno bi olakšala rješavanje nagomilanih problema. Naime, osuvremenjivanjem komunikacije privuklo bi se veći broj mlađih suvlasnika pa bi među sustanarima cirkuliralo više ideja i novih rješenja što bi u konačnici pridonijelo kvaliteti održavanja i osuvremenjivanju zgrade. Demokratizacija procesa održavanja, pretvorila bi suvlasnike u aktivne sudionike u održavanju zgrade, povezala bi ih u zajednicu koja će biti zainteresirana promicati boljitak zgrade.

3. Korisnici sustava

Korisnici aplikacije su fizičke osobe – suvlasnici, sustanari i predstavnik stanara, odnosno pravne osobe – upravitelji. Kako bi mogli koristiti aplikaciju i sve njene funkcionalnosti, korisnici moraju izraditi korisnički račun na aplikaciji. Svaki korisnik je jedinstven, a aplikacija ga raspoznaje od drugih prema njegovu imenu i prezimenu.

Različite vrste korisnika imaju različite mogućnosti korištenja aplikacije.

Stanari su korisnici aplikacije – fizičke ili pravne osobe - koji se prijavljuju kao suvlasnici ili kao osobe koje, uz dozvolu suvlasnika, u određenom razdoblju žive u pojedinoj zgradi (članovi obitelji, podstanari i sl.). Stanari mogu pregledati popis svih zgrada koje koriste aplikaciju i poslati zahtjev za pridruživanjem novoj zgradi. Pridruživanjem nekoj zgradi, što odobrava predstavnik stanara te zgrade, korisnik može uređivati svoj korisnički profil, gledati tuđe korisničke profile, pregledati opće informacije o zgradi, troškove zgrade, prijaviti kvar na zgradi, pregledati novosti, pokrenuti debatu, glasovati i pokrenuti novo glasovanje, sudjelovati u raspravama i komentirati sve što se dešava.

Predstavnik stanara je fizička osoba koju stanari biraju da ih predstavlja pred upraviteljem i svim trećim stranama [2]. Predstavnik stanara, prema zakonu, mora biti i suvlasnik u zgradi. On se može koristiti svim funkcionalnostima dostupnim stanarima i dodatno funkcionalnostima dodavanja novih korisnika aplikacije u zgradu koju predstavlja, izbacivanja korisnika aplikacije iz zgrade, ažuriranja općih informacija o zgradi, ažuriranja radova na održavanju, ažuriranja troškova i sl. Predstavnik stanara koristi aplikaciju i za komunikaciju s upraviteljem zgrade.

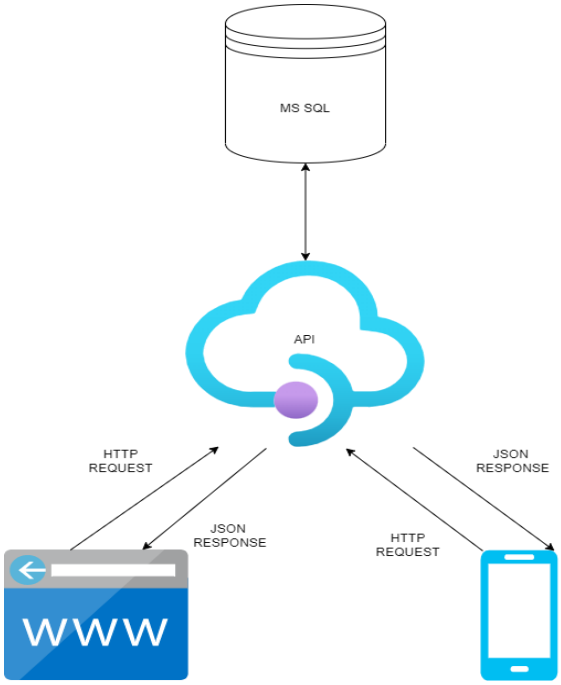
Upravitelj zgrade, kojeg prema zakonu mora imati svaka zgrada, pravna je osoba koja organizira sve popravke na zgradi, brine o redovnom održavanju i eventualnim investicijama u poboljšavanje funkcionalnosti zgrade. Redovno održavanje građevine obuhvaća provođenje skupa preventivnih mjera koje se provode prema prethodno utvrđenom planu i programu kako bi se trajno zadržala primjerena uporabljivost građevine tijekom njezina trajanja te skup preventivnih ili interventnih mjera koje obuhvaćaju zamjenu, dopunu i/ili popunu dijelova građevine i ugrađene opreme u razmacima i opsegu određenim projektom građevine, odnosno u slučaju kada dio građevine više nije uporabljiv, a ta neuporabljivost nije posljedica kakvog izvanrednog događaja [3]. Upravitelj zgrade kao korisnik aplikacije

ima mogućnost koristiti sve funkcionalnosti stanara, osim glasovanja. Dodatno, upravitelj ima mogućnost izravne komunikacije s predstavnikom stanara te, u koordinaciji s njim, vodi evidenciju o kvarovima na zgradi, ažurira status prijavljenih kvarova, stanje radova na održavanju, troškove održavanja, iznosi godišnje planove održavanja i ažurira podatke o njihovu izvršavanju te obavještava suvlasnike o zakonskim obavezama glede održavanja npr. obaveza tehničkog pregleda zgrade jednom u 10 godina što se posebno plaća.

4. Arhitektura sustava

Ovo poglavlje bavi se arhitekturom na kojoj je izgrađen cijeli sustav. Svaki sloj odabrane arhitekture detaljno je opisan te su predstavljene primijenjene tehnologije, programski jezici i pomoćni programi korišteni pri izradi pojedinog sloja.

4.1. Opis arhitekture sustava



Slika 4.1 Arhitektura sustava

Obrazac dizajna programske podrške korišten za izradu ovog sustava troslojna je arhitektura [4]. Troslojna klijent-poslužiteljska arhitektura odabrana je jer se funkcionalna procesna logika, pristup podacima a i pohranjivanje podataka i korisničko sučelje razvijaju i održavaju kao neovisni moduli na zasebnim platformama. Sastoji se od klijentskog sloja, logičkog sloja i podatkovnog sloja. Klijentski sloj je grafičko korisničko sučelje, logički sloj upravlja logikom aplikacije, a podatkovni sloj pohranjuje informacije unutar aplikacije. Slojevi su logički, ne fizički, i rade na istom fizičkom poslužitelju.

Radi neovisnosti slojeva unutar aplikacije troslojna arhitektura je jednostavnija za održavanje i po potrebi dogradnju koda i razvijanje aplikacije. Ukoliko se promijeni jedan sloj to neće utjecati na drugi. Svi dijelovi mogu se istovremeno razvijati i to od različitih

timova programera koji kodiraju na različitim jezicima. Postojeće aplikacije ili kritični dijelovi mogu se trajno ili privremeno zadržati i enkapsulirati unutar nove razine.

Prednosti korištenja troslojne arhitekture uključuju i poboljšanu horizontalnu skalabilnost, izvedbu i dostupnost.

4.1.1. Klijentski sloj

Klijentski sloj aplikacije, također zvan i prezentacijski, sloj je na kojem se podaci pohranjeni u bazi prezentiraju korisniku. Čine ga web sučelje i mobilno sučelje. Oba sučelja sastojat će se od komponenata:

- Prijava
- Registracija
- Izbornik
- Stranica korisničkog profila
- Opće informacije o zgradi
- Sučelje za kreiranje i čitanje novosti
- Sučelje za glasovanje
- Forum
- Sučelje za pregled katova
- Sučelje za pregled stana
- Sučelje za pridruživanje zgradi
- Sučelje za kreaciju zgrade
- Sučelje za prijavu i pregled kvarova.

Web aplikacija napisana je u JavaScript skriptnom jeziku pomoću ReactJS okvira (*engl. frameworka*). Mobilna aplikacija također je izgrađena u Javascript-u pomoću ReactNative okvira (*engl. frameworka*). ReactJS je odabran kao framework radi izrade komponenti te radi SPA (Single-page Application) implementacije. SPA implementacija učitava samo jedan web-dokument te nikada ne osvježava taj web-dokument. Umjesto toga, sav potreban kod dohvaća se učitavanjem prve stranice ili se odgovarajući resursi dinamički učitavaju i dodaju na stranicu prema potrebi, obično kao odgovor na radnje korisnika. To korisnicima omogućuje korištenje web stranica bez učitavanja potpuno novih stranica s poslužitelja, što dovodi do poboljšanja performansi i dinamičnijeg iskustva [5]. Korišteno razvojno okruženje je Visual Studio Code, a pomoćni alati korišteni pri izradi aplikacije su React

devtools i Redux devtools. Prezentacijski sloj komunicira s ostalim slojevima putem poziva raznih servisa (API).

U razvojnom okruženju klijentski sloj živi lokalno na portu 3000.

4.1.2. Logički sloj

Aplikacijski sloj (servisni ili logički sloj), napisan je u programskom jeziku Java, verzije 8, pomoću razvojnog okruženja, NetBeans IDE (engl. *Integrated Development Environment*) 8.2. Za razvoj praćeni su obrasci dizajna programske podrške - *Singleton* i *Factory,* detaljnije opisano u poglavlju „Implementacija rješenja po slojevima“. Ovaj sloj sadrži poslovnu logiku koja podržava temeljne funkcije aplikacije.

Pri razvoju logičkog sloja korištene su i vanjske biblioteke poput GSON-a (engl. *Google JavaScript Object Notation*) za pretvaranje objekata u JSON (engl. *JavaScript Object Notation*) oblik te JSON objekata u objekte unutar Java-e i JDBC-a (engl. *Java Database Connection*) za spajanje na bazu i pozivanje procedura. *BuildingManager* aplikacija nalazi se na *Apache Tomcat* poslužitelju.

U razvojnom okruženju logički sloj živi lokalno na portu 8080.

4.1.3. Podatkovni sloj

Podatkovni sloj čini relacijska baza podataka u koju se spremaju svi podaci unutar sustava. Baza je kreirana po 3NF (engl. *Normal Form*) standardu. Normalizacija je proces minimiziranja redundantnosti podataka unutar tablica [6]. Redundancija može uzrokovati anomalije umetanja, brisanja i ažuriranja podataka. Stoga, normalizacija funkcionira u nizu faza koje se nazivaju normalne forme. Normalne forme koriste se za uklanjanje ili smanjenje redundantnosti u tablicama baze podataka. Treća normalna forma kaže da svaka ćelija tablice treba sadržavati samo jednu vrijednost, svaki redak mora biti jedinstven, svaka tablica mora imati primarni ključ koji služi kao jedinstveni identifikator te relacije među tablicama moraju se izvršavati preko stranih ključeva. Izrađena je u *SQL Server Managment Studio*-u, koristeći T-SQL (engl. *Transact Structured Query Language*) programski jezik. Manipulacija podataka unutar baze izvedena je putem procedura kako bi se podaci zaštitili od mogućih prijetnji poput *SQL Injection*-a, opisan u poglavlju 7.3.3. Svaki entitet izveden je putem tablica jer se radi o relacijskoj bazi podataka, a svakoj tablici pripadaju odgovarajuće CRUD

(engl. *Create, Read, Update and Delete*) procedure. Baza nije usko povezana uz REST API (engl. *Representational State Transfer Application Programming Interface*) servis, što govori da su metode na servisu i entiteti unutar baze gotovo neovisni ili manje ovisni jedni o drugima (princip *Loose coupling*). U slučaju da dođe do promjene na REST API - ju nije potrebno mijenjati bazu podataka i obrnuto.

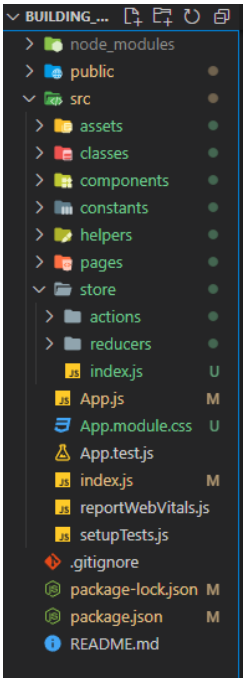
5. Implementacija rješenja po slojevima

Unutar ovog poglavlja predstavlja se implementacija pojedinog sloja arhitekture. Primijenjene programerske prakse, principi i obrasci korišteni tijekom razvoja aplikacije, struktura pojedinog sloja te dijelovi koda što pokreću aplikaciju detaljno su opisani i potkrijepljeni slikovnim prikazima.

5.1. Klijentski sloj

Klijentski sloj izrađen je na principu *separation of concerns*. Navedeno načelo koristi se u programiranju za razdvajanje aplikacije u jedinice, s minimalnim preklapanjem između funkcija pojedinačnih jedinica. Razdvajanje problema postignuto je korištenjem modularizacije, enkapsulacije i rasporeda datoteka u mape i podmape.

5.1.1. Struktura



Slika 5.1 Struktura klijentskog sloja

Kod je strukturiran u više dijelova (komponenti i sučelja). Osnovni HTML5 (engl. *HyperText Markup Language*) elementi tvore komponente, a komponente tvore sučelja. HTML5 je deskriptivni jezik koji specificira strukturu web stranice. Sastoji se od niza

elemenata koji programerima olakšavaju izradu web stranica [7]. Struktura komponenti u oba sučelja je ista. Međutim, razlikuju se u izgledu komponenti i elemenata od kojih su komponente složene.

Unutar ReactJs (*front-end* biblioteka za izgradnju korisničkih sučelja temeljena na komponentama grafičkog sučelja [8]) okvira kod je podijeljen na jednu datoteku - paket (engl. *package*) i dvije mape - javno (engl. *public*) i izvor (engl. *source*).

Datoteka paket sadrži sve vanjske biblioteke korištene u projektu poput ikona, grafova, react-redux-a, navigacije i sl. U njemu su također definirane skripte za pokretanje, testiranje i razvijanje aplikacije.

Unutar mape javno nalaze se sve mape i datoteke koje su dostupne na globalnoj razini unutar projekta, a i početna stranica aplikacije (engl. *index*) koja učitava sve komponente aplikacije.

Unutar mape izvor nalazi se sav kod aplikacije podijeljen u manje cjeline. Sastoji se od mapa – assets, components, constants, helpers, pages i store.

Mapa assets sadrži sve pomoćne objekte poput slika ili videa.

Mapa components sadržava sve komponente korištene za izradu sustava. Komponente su manji dijelovi sustava koji se pojavljuju na više mjesta unutar aplikacije, a sastavljene su od HTML5 oznaka.

Mapa constants sadržava sve konstante korištene u kodu poput iskočnih poruka, enumeracija, pristupnih točki servisa i slično.

Mapa helpers sadrži pomoćne funkcije korištene na više mjesta u kodu poput funkcije za „front-end“ validaciju tj. provjeru ispravnosti podataka na sučelju.

Mapa pages sadržava sva sučelja aplikacije. Sučelja su napravljena od skupa komponenti, a svako sučelje ima i dodatnu css datoteku preko koje može još dodatno stilizirati sučelje.

Mapa store je mapa koja služi za upravljanje i centraliziranje stanja aplikacije s pomoću Redux toolkit biblioteke. Sastoji se od akcija i reduktora. Podaci se ažuriraju slanjem akcije koja govori kako bi se podaci trebali promijeniti. Svaka akcija u ovom slučaju zove neki servis implementiran na logičkom sloju i vraća poruku o uspješnosti. Reduktori su funkcije koje primjenjuju akcije na podatke, prate njihovo izvršavanje i vraćaju novo stanje, umjesto da mijenjaju prethodno stanje. Prilikom učitavanja sučelja ili komponenti (unutar metode useEffect, koja se izvršava prije iscrtavanja sadržaja sučelja ili komponente na ekran

korisnika) te prilikom klika na pojedini gumb zovu se akcije koje dohvaćaju podatke sa servisa te se ti podaci potom iscrtavaju na sučelje, a pomoću reduktora unutar Redux devtools, pomoćne ekstenzije, vidljivi su podaci o uspješnosti provedbe akcije.

5.1.2. Komponente

Komponente pridonose lakoj skalabilnosti i čistoći koda. Podijeljene su u dvije kategorije – zajedničke komponente i komponente specifične za pojedino sučelje.

Zajedničke komponente su komponente poput gumbova, polja za unos podataka, notifikacija i formi koje se pojavljuju na više mjesta unutar aplikacije. Takve komponente izdvojene su radi lakše nadogradnje ili izmjene koda. One također pridonose kontinuiranom izgledu aplikacije. Naime, ukoliko bude trebalo mijenjati dizajn komponente, to će biti dovoljno učiniti na jednom mjestu i utjecat će na sve instance te komponente.

Komponente specifične za neko sučelje koriste se samo u jednom sučelju i ne pojavljuju se više od jednog puta u kodu, poput obrasca za unos podataka na sučelju za registraciju korisnika.

Komponentama su dodane i dodatne funkcionalnosti koje originalni HTML5 elementi nemaju.

5.1.3. Sučelja

Klijentski sloj aplikacije čini jedanaest sučelja:

- sučelje autentifikacije i autorizacije
- sučelje za pregled svih pridruženih zgrada
- izbornik
- sučelje za pregled stanova po katu
- sučelje o općim informacijama o zgradi
- forum
- sučelje za pregled troškova
- sučelje za pregled novosti
- sučelje za pregled korisničkog profila
- sučelje za prijavu kvarova
- sučelje za glasovanje

Sučelje autentifikacije i autorizacije početno je sučelje koje se otvara neregistriranim korisnicima. Ovo sučelje sačinjeno je od podsučelja unutar kojega se korisnik odlučuje za ulogu na koju se želi prijaviti ili registrirati. Koristi zajedničke komponente, gumb i polja za unos, za kreaciju obrasca. Prilikom unosa podataka zove servis za prijavu ili registraciju ovisno o odabiru korisnika.

Sučelje za pregled svih pridruženih zgrada koristi komponente kartica za prikaz pojedine zgrade i zajedničku komponentu gumb. Prije iscrtavanja komponenti sučelja poziva se servis koji vraća sve zgrade u kojima je korisnik registriran. Vraćeni podaci također ovise o ulozi koju korisnik obnaša.

Izbornik je sučelje na kojemu korisnik prolazi kroz aplikaciju i koristi sve dostupne funkcionalnosti. Sastoji se od komponenata kartica i grafičkih ikona uključenih u projekt pomoću vanjske biblioteke. Kartice koje će biti prikazane korisniku, ovisit će o njegovoj ulozi, npr. upravitelju zgrade neće biti prikazana kartica za glasovanje.

Sučelje za pregled stanova po katu većim djelom tvore osnovni HTML5 elementi, a manjim djelom zajedničke komponente. Prije iscrtavanja komponenti sučelja poziva se servis koji vraća sve stanove na tom katu te sve stanare unutar pojedine zgrade. Svaki stan može biti detaljno pregledan. Iznos pričuve pojedinog stana ne vuče se iz baze, radi dinamičnosti podataka, već se računa na sučelju po formuli: površina stana*iznos pričuve po m2. Iznos računa vode ovisi o tome ima li stan brojač ili ne. Ukoliko nema, iznos računa vode po stanu računa se po formuli: ukupna potrošnja stanova koji nemaju brojilo/brojem stanara u stanovima koji nemaju brojilo*broj stanara pojedinog stana. Ukoliko stan ima brojilo vode, račun za vodu se povlači sa stranice Holdinga, što znači da se trošak vode u tom slučaju ne računa unutar aplikacije već se samo ispisuje.

Sučelje s općim informacijama o zgradi čine dvije glavne komponente – 3d model zgrade i metapodaci zgrade. Prije iscrtavanja komponenti sučelja poziva se servis koji vraća sve podatke o pojedinoj zgradi.

Forum je sastavljen od jedne zajedničke komponente, polja za unos, i padajućeg izbornika uključenog u projekt s pomoću vanjske biblioteke. Prije iscrtavanja komponenti sučelja poziva se servis koji vraća sve debate aktivne i zatvorene a i sve komentare. Tijekom objave novog komentara poziva se servis za kreaciju komentara, a prilikom objave nove debate, zove se servis za kreiranje nove debate. Prilikom označavanja pojedinog komentara sa „sviđa mi se“ zove servis za brojanje komentara.

Sučelje za pregled troškova zgrade sastoji se od zajedničkih komponenti - gumba, obrasca te grafičkog prikaza uključenog u projekt s pomoću vanjske biblioteke. Prije iscrtavanja komponenti sučelja poziva se servis koji prikazuje sve troškove pojedine zgrade. Minimalan iznos za pričuvu zgrade određen je zakonom te iznosi 0,54% vrijednosti pojedinog dijela nekretnine godišnje. Budući da zakonom nije određena vrijednost posebnih dijelova nekretnine, uobičajeno je da se za izračun vrijednosti pojedinog dijela zgrade, kao parametar, uzima etalonska vrijednost građenja po m2 neto korisne površine koju svake godine objavljuje Ministarstvo graditeljstva i prostornog uređenja. Taj iznos može se mijenjati naviše u skladu s dogovorom stanara (najavljeni novi zakon o zgradama mogao bi izmijeniti navedeni minimalni postotak jer se pokazalo da je takav iznos nedostatan za kvalitetno održavanje). Sredstva zajedničke pričuve namjenjena su za pokriće troškova održavanja i poboljšavanja zajedničkih dijelova nekretnine te ev. za otplaćivanje zajma za pokriće tih troškova [9]. Prilikom brisanja pojedinog troška, poziva se servis za brisanje troškova unutar zgrade.

Sučelje za pregled novosti koristi zajedničke komponente – karticu, polja za unos i skočni prozor. Prije iscrtavanja komponenti sučelja poziva se servis koji vraća sve novosti pojedine zgrade. Prilikom unosa nove obavijesti poziva se servis za kreiranje nove obavijesti.

Sučelje za pregled korisničkog računa sastoji se od osnovnih HTML5 elemenata i zajedničkih komponenti poput gumba i metapodataka. Prije iscrtavanja komponenti sučelja poziva se servis što vraća sve informacije o prijavljenom korisniku. Prilikom ažuriranja korisničkog profila, poziva se servis za ažuriranje korisničkih podataka, a prilikom brisanja korisničkog profila poziva se servis za brisanje korisničkih podataka.

Sučelje za prijavu kvarova koristi zajedničke komponente poput gumba i polja za unos te ikona uključenih u projekt s pomoću vanjske biblioteke. Prije iscrtavanja komponenti sučelja, poziva se servis koji vraća sve kvarove pojedine zgrade. Prilikom prijave novog kvara, poziva se servis za kreiranje novog kvara. Pri brisanju kvara, poziva se servis za brisanje kvara. Prilikom ažuriranja kvara, poziva se servis za ažuriranje kvara.

Sučelje za glasovanje koristi zajedničke komponente poput gumba i polja za unos i odabir podataka. Prije iscrtavanja komponenti sučelja poziva se servis koji vraća sva glasovanja pojedine zgrade. Prilikom otvaranja novog glasovanja, poziva se servis za kreiranje novog glasovanja. Pri davanju glasa za neku od mogućnosti, poziva se servis za brojanje glasova.

5.2. Logički sloj

Komunikacija između klijentskog i podatkovnog sloja odvija se putem logičkog ili servisnog sloja. Unutar aplikacije implementiran je kao REST Api servis. Servis je dio putem kojeg klijentski sloj dobiva i manipulira podacima iz logičkog sloja. Logički sloj *BuildingManager* aplikacije podijeljen je u dvije cjeline – servis i repozitorij.

Servis se sastoji od kontrolera i filtera. Svaki kontroler mapiran je na jedinstveni lokator izvora (*URL*). Servis živi na serveru koji ga pokreće i osluškuje zahtjeve klijenata. Pozivom jedinstvenog lokatora izvora, servis čita zahtjev (*engl. request*) pošiljatelja. Većina zahtjeva unutar aplikacije poslana je POST metodom radi veće sigurnosti podataka te radi nepostojećeg ograničenja u pogledu duljine podataka. POST metoda jedna je od metoda slanja podataka, na internetu, od klijenstkog do logičkog sloja. Unutar aplikacije korištene su POST i GET metode [10]. GET metodom može se poslati samo ograničena količina podataka jer se podaci šalju u zaglavlju zahtjeva te su samim time vidljivi unutar URL-a. U slučaju POST metode može se poslati velika količina podataka jer se podaci šalju u tijelu zahtjeva te radi toga nisu vidljivi unutar URL-a zahtjeva. GET metoda je idempotentna. To znači da će drugi zahtjev biti zanemaren dok se ne dostavi odgovor na prvi zahtjev. POST metoda, s druge strane, nije idempotentna te može obavljati više zahtjeva istovremeno.

```
public class BugServlet extends HttpServlet {

    private SqlBug bugRepo;

    @Override

    protected void doGet(HttpServletRequest request, HttpServletResponse
response)

            throws ServletException, IOException {

    }

    @Override

    protected void doPost(HttpServletRequest request, HttpServletResponse
response)

            throws ServletException, IOException {

    }

}
```

Kôd 5.1 GET i POST funkcije

Svaki zahtjev sadržava parametar akcija (engl. *action*), koji služi kao pomoćni parametar, za određivanje procedure koju treba izvršiti i za podatke, u JSON formatu, o objektu koji se šalje.

```
String requestData = request.getReader().lines().collect(Collectors.joining());
        String action = request.getParameter("action");
        try {
            switch (action) {
                case "create":
                    Bug bug = GsonUtils.convertFromGson(requestData, Bug.class);
                    bugRepo.createBug(bug);
                    break;
                case "all":
                    int buidingId = GsonUtils.convertFromGson(requestData,
int.class);
                    List<Bug> bugs = bugRepo.getBugs(buidingId);
                    GsonUtils.convertToGson(bugs, false, response);
                    break;
                case "resolve":
                    int bugId = GsonUtils.convertFromGson(requestData, int.class);
                    bugRepo.resolveBug(bugId);
                    GsonUtils.convertToGson("Report successfully resolved!", false,
response);
                    break;
                case "delete":
                    int id = GsonUtils.convertFromGson(requestData, int.class);
                    bugRepo.deleteBug(id);
                    GsonUtils.convertToGson("Report successfully deleted!", false,
response);
                    break;
                case "progress":
                    int idBug = GsonUtils.convertFromGson(requestData, int.class);
                    bugRepo.changeProgress(idBug);
                    GsonUtils.convertToGson("Success!", false, response);
                    break; }
        } catch (Exception ex) { GsonUtils.convertToGson(ex, true, response);  }
```

Kôd 5.2 Filtriranje parametra akcija

Nakon što kontroler pronađe akciju koju treba izvršiti, počinje čitati proslijeđeni objekt. Kako se informacije između pošiljatelja i primatelja isporučuju u JSON formatu, koji Java ne zna automatski pročitati prvo ga mora parsirati. Parsiranje objekata izvodi se s pomoću vanjske biblioteke GSON.

```
public static <T> T convertFromGson(String message, Class<T> desiredClass)
throws IOException {

        Gson gson = new Gson();

        T javaObject = gson.fromJson(message, desiredClass);

        return javaObject;

  }
```

Kôd 5.3 Parsiranje objekta u klasu

Ukoliko parsiranje objekta prođe uspješno, proslijeđuje se dalje u repozitorij te se ovisno u uspjehu u klijentski sloj vraća odgovor (engl. *response*) - poruku o uspjehu ili neki drugi objekt, ovisno o akciji, pretvoren u JSON format. Po ispunjenju zahtjeva klijenta, server će prekinuti komunikaciju.

```
public static void convertToGson(Object message, boolean isError,
HttpServletResponse response) throws IOException {

        String messageJson = new Gson().toJson(message);

        PrintWriter out = response.getWriter();

        response.setContentType("application/json");

        response.setCharacterEncoding("UTF-8");

        if(isError) response.sendError(404, messageJson);

        out.print(messageJson);

        out.flush();

}
```

Kôd 5.4 Parsiranje objekta iz klase u JSON format

Cijeli repozitorij izgrađen je s pomoću *Factory* obrazca. *Factory* obrazac temeljen je na sučeljima koja omogućuju lakšu nadogradnju aplikacije tako što sve nove klase implementiraju potrebno im sučelje. Također, logika stvaranja objekta skrivena je od klijenta [11]. U repozitoriju su određene sve klase i njihovi atributi, a i pomoćne metode za dohvat i manipulaciju klasa [12]. Atributi klasa isti su kao i atributi koji opisuju entitete u bazi. Osim klasa unutar repozitorija nalaze se i sučelja, što komuniciraju direktno s bazom podataka, tj.

procedurama te klase koje implementiraju ta sučelja. Povezivanje i komunikacija s bazom odrađena je praćenjem *Singleton* obrasca za dizajn programske podrške na principu *lazy loadinga. Lazy loading* je princip odgađanja učitavanja ili inicijalizacije resursa ili objekata dok se ne pojavi potreba za njim. On oklijeva stvoriti novi objekt sve dok se, određena funkcija koja kreira ili učitava objekt, prvi put ne pozove u kodu. Taj prinicip pridonosi poboljšanju performansi te štedi resurse sustava [13]. *Singleton* je dizajn obrazac koji ograničava instanciranje klase na jedan objekt [14]. Na taj se način čuva memorija jer se instanca objekta ne kreira sa svakim pozivom baze već se koristi samo jedna instanca, koja se kreira prvim pozivom baze unutar logičkog sloja. Za spajanje s bazom, izdavanje upita i naredbi te rukovanje skupovima rezultata dobivenih iz baze podataka korištena je vanjska biblioteka JDBC.

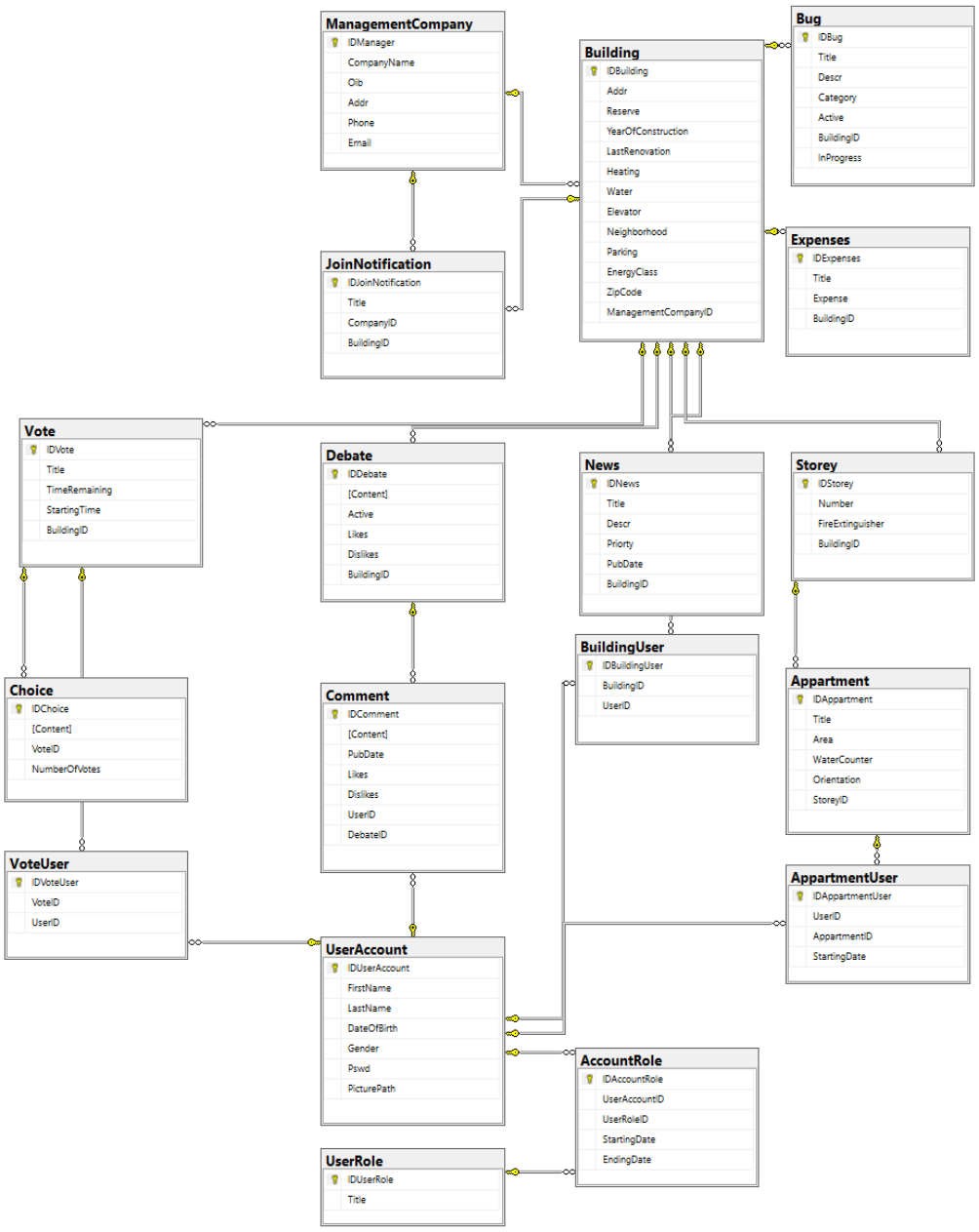
```
import com.microsoft.sqlserver.jdbc.SQLServerDataSource;
import javax.sql.DataSource;

public class DataSourceSingleton {
    private static DataSource instance;

    private DataSourceSingleton() {
    }

    public static DataSource getInstance() {
        if (instance==null) {
            instance=createInstance();
        }
        return instance;
    }

    private static DataSource createInstance() {
        SQLServerDataSource dataSource=new SQLServerDataSource();
        dataSource.setServerName(SERVER);
        dataSource.setUser(USER);
        dataSource.setDatabaseName(DB);
        dataSource.setPassword(PWD);
        return dataSource;
    }
```

Kôd 5.5 Spajanje na bazu praćenjem *Singleton* obrasca

5.3. Podatkovni sloj

Sve tablice unutar baze pohranjuju podatke o nekom entitetu i sve su povezane vezama (jedan na jedan, jedan na više i više na više) unutar relacijske baze podataka. Svaku tablicu opisuju njezini atributi, jedinstven identifikator i, ako postoji, jedan ili više stranih ključeva. Jedinstveni identifikator unutar svake tablice implementiran je kao primarni ključ koji se

automatski generira prilikom unosa te instance u tablicu tj. bazu. On je obavezan za svaku instancu unutar tablice te se instanca nikada neće kreirati bez njega [15].



Slika 5.2 Struktura tablica unutar baze podataka

Unutar **tablice korisnički račun** (engl. *UserAccount*) spremaju se korisnički računi registrirani unutar aplikacije. Tablica se sastoji od primarnog ključa i atributa (ime, prezime, godina rođenja, spol, lozinka i slika). Atributi: ime, prezime i lozinka obavezni su te se novi korisnik ne može kreirati bez njih. Ostali atributi mogu se naknadno dodavati i brisati.

Tablica unutar koje se spremaju moguće **korisničke uloge** (engl. *UserRole*) (stanar, predstavnik stanara i upravitelj zgrade) vezana je uz tablicu korisničkog računa vezom više na više putem treće tablice uloga korisničkog računa (engl. *AccountRole*). Ova veza upućuje na to da jedan stanar može imati više uloga, npr. biti ujedno stanar i predstavnik stanara te da ulogu stanara može obnašati više korisnika unutar aplikacije. Pomoćna, treća tablica također sprema i dva dodatna podatka, a to su vrijeme početka obnašanja uloge i vrijeme završetka obnašanja uloge.

Tablica komentara (engl. *Comment*) sprema sve komentare unutar aplikacije. Sastoji se od primarnog ključa, dva strana ključa (korisnik i debata) i atributa (sadržaj, datum objave i brojača „sviđa mi se“). Na atribute sadržaj i datum objave postavljeno je ograničenje NOT NULL, koje onemogućava unos novog komentara bez sadržaja i datuma objave. Datum objave automatski se generira pri unosu novog komentara, tj. korisnik ne unosi datum objave već ga sustav sam odredi i upiše. Putem dva strana ključa treća tablica povezuje vezom jedan na više tablice debata i korisnički račun, što znači da korisnik s jednog korisničkog računa može kreirati jedan ili više komentara unutar jedne ili više debata.

Tablica debata (engl. *Debate*) sprema sve debate kreirane unutar aplikacije. Sastoji se od primarnog ključa, jednog stranog ključa (zgrada) i atributa (sadržaj, informaciju da li je aktivna i brojača „sviđa mi se“). Na atribut sadržaj postavljeno je ograničenje NOT NULL, koje onemogućava kreiranje nove debate bez sadržaja. Informacija o tome je li debata aktivna pri kreiranju debate postavlja se na istinu i tu informaciju korisnik nigdje ne unosi. Putem stranog ključa povezana je vezom jedan na više s tablicom zgrada, što znači da jedna zgrada unutar sebe može imati više debata.

Tablica zgrada (engl. *Building*) sprema sve opće podatke o nekoj zgradi registriranoj unutar aplikacije. Sastoji se od primarnog ključa, atributa (adresa, poštanski broj, kvart, godina izgradnje, godina zadnje obnove, način grijanja, iznos pričuve, smije li se voda piti iz slavine, broj dizala, ima li parking, energetski razred) i stranog ključa (upravitelj). Svi atributi, osim godine zadnje obnove, obavezni su pri unosu nove zgrade u aplikaciju. Strani ključ omogućuje vezu jedan na više s tablicom upravitelj, što znači da jednoj zgradi može pripadati samo jedan upravitelj, ali jedan upravitelj može održavati više zgrada. Također, jedna zgrada može imati više stanara, a jedan stanar može imati više stanova unutar različitih zgrada. Stoga je potrebna veza više na više, što se ostvaruje dodatnom tablicom *BuildingUser*.

Tablica upravitelj (engl. *ManagementCompany*) sastoji se od primarnog ključa i atributa (ime, OIB, adresa, telefon i adresa elektroničke pošte). Svi atributi su obavezni. Novi redak ne može se umetnuti u tablicu bez svih podataka.

Tablica novosti (engl. *News*) sprema sve novosti, unutar pojedine zgrade, objavljene od strane predstavnika stanara i upravitelja zgrade. Sastoji se od primarnog ključa, atributa (naslov, opis, prioritet, datum objave) i stranog kluča (zgrada). Svi atributi su obavezni, a atribut datum objave sam se generira pri unosu nove instance u tablicu. Putem stranog ključa omogućena je veza jedan na više s tablicom zgrade, što znači da jedna zgrada može imati više novosti.

Tablica kvarova (engl. *Bug*) sprema sve kvarove pojedine zgrade što je objavljuju stanari zgrade. Sastoji se od primarnog ključa, atributa (naslov, opis, kategorija, da li je aktivan, da li se riješava) i stranog kluča (zgrada). Svi atributi su obavezni. Atribut u koji se sprema informacija o aktivnosti kvara pri unosu nove instance automatski se postavlja na istinu, a atribut inProgress postavljen je na laž. Putem stranog ključa omogućena je veza jedan na više s tablicom zgrade, što znači da jedna zgrada može imati više prijavljenih kvarova istovremeno.

Tablica troškova (engl. *Expenses*) sprema sve troškove pojedine zgrade. Sastoji se od primarnog ključa, atributa (naslov i iznosa) i stranog kluča (zgrada). Svi su atributi obavezni. Putem stranog ključa omogućena je veza jedan na više s tablicom zgrada, što znači da jedna zgrada može imati više prijavljenih troškova istovremeno.

Tablica kat (engl. *Storey*) sprema sve informacije o pojedinom katu, unutar zgrade. Sastoji se od primarnog ključa, atributa (broj i ima li aparat za gašenje vatre) i stranog kluča (zgrada). Svi atributi su obavezni. Putem stranog ključa omogućena je veza jedan na više s tablicom zgrada, što znači da jedna zgrada može imati više katova.

Tablica stan (engl. *Appartment*) sprema sve informacije o pojedinom stanu, unutar zgrade. Sastoji se od primarnog ključa, atributa (naslov, površina, orijentacija, brojač vode) i stranog ključa (kat). Svi su atributi obavezni. Putem stranog ključa omogućena je veza jedan na više s tablicom kat, što znači da jedan kat može imati više stanova. Također, unutar jednog stana može živjeti više stanara te isto tako jedan stanar može živjeti/posjedovati više stanova unutar pojedine zgrade. Stoga se tu pojavljuje veza više na više, putem dodatne tablice *AppartmentUser*, koja osim stranih ključeva sprema i datum početka boravka stanara u stanu.

Tablica glasovanja (*engl. Vote*) sprema sva glasovanja pojedine zgrade. Sastoji se od primarnog ključa, atributa (naslov, preostalo vrijeme i vrijeme početka) i stranog ključa (zgrada). Svi atributi su obavezni. Putem stranog ključa omogućena je veza jedan na više s tablicom zgrada, što znači da jedna zgrada može imati više glasovanja istovremeno.

Svako glasovanje može imati jedno ili više mogućih opcija za odabir stoga je kreirana **tablica opcija** (*engl. Choice*) u koju se spremaju opcije pojedinog glasovanja. Tablica opcija ima primarni ključ, atribute (sadržaj i broj glasova) i strani ključ (glasovanje). Svi atributi su obavezni. Atribut sadržaj korisnik sam unosi putem aplikacije, a atribut broj glasova na početku je postavljen na nulu. Kasnije se svakim glasom za određenu opciju automatski povećava za jedan.

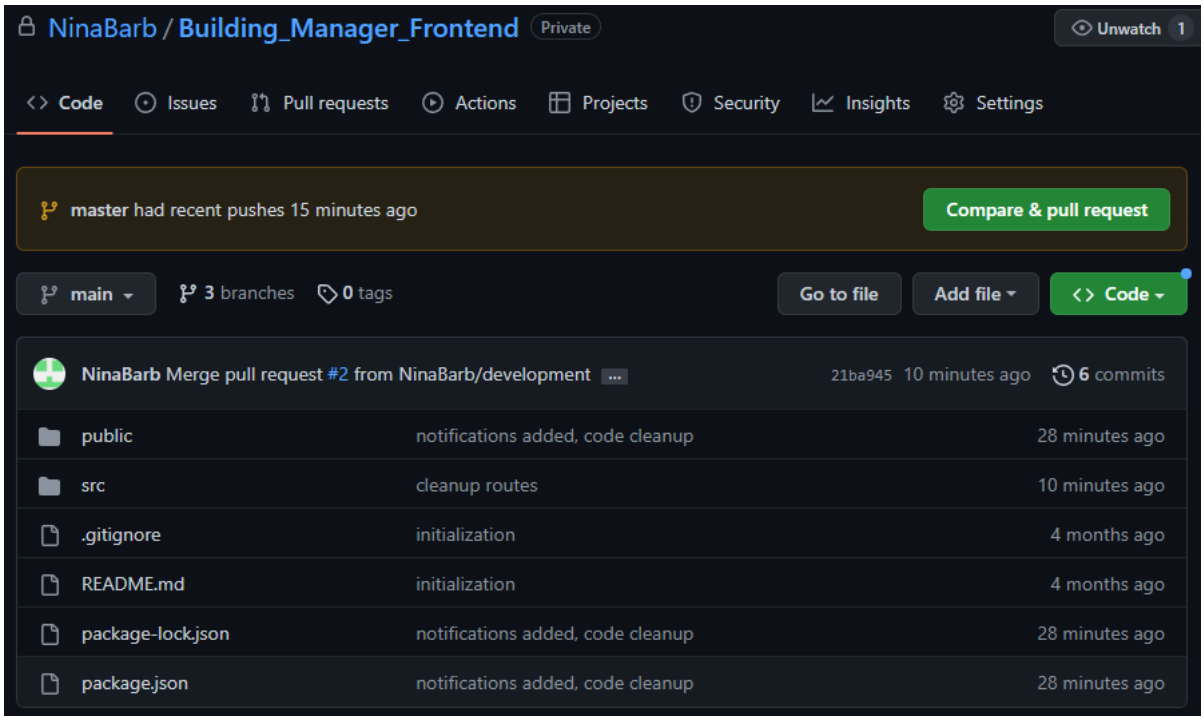
Tablica notifikacija (*engl. JoinNotification*) sastoji se od primarnog ključa, atributa (naziv) i stranih ključeva (zgrada i upravitelj). Unutar te tablice ne uzimaju se podaci uneseni izravno od korisnika, već ih se pri pozivu procedure uzima iz sesije, unutar klijentskog sloja ili se uzima postavljena konstantna vrijednost. S pomoću stranog ključa zgrada omogućena je i veza jedan na više (jednoj zgradi može pripadati više notifikacija, ali pojedina notifikacija odnosi se samo na jednu zgradu) te je putem stranog ključa upravitelju omogućena veza jedan na više (jednom upravitelju može pripadati više notifikacija, ali pojedina notifikacija odnosi se samo na jednog upravitelja).

6. Verzioniranje koda

Upravljanje izvornim kodom izvedeno je s pomoću Git alata. Git je distribuirani sustav za čuvanje i kontrolu verzija koda. On prati promjene u bilo kojem skupu datoteka te koordinira rad programera koji zajednički razvijaju izvorni kod tijekom razvoja softvera. Prednosti Git-a uključuju brzinu, integritet podataka i podršku za distribuirane, nelinearne tijekove rada (tisuće paralelnih grana koje rade na različitim sustavima) [16].

Napravljena su tri privatna repozitorija za čuvanje verzija koda. *BuildingManagerFrontend* repozitorij je na kojem se čuvaju verzije koda za web klijentski sloj. *BuildingManagerMobile* je repozitorij na kojem se čuvaju verzije koda za mobilni klijentski sloj. *BuildingManagerService* je repozitorij na kojem se čuvaju verzije koda logičkog i podatkovnog sloja aplikacije.

Prilikom svake izmjene koda, sve promjene spremljene su u poseban Git repozitorij. Trenutno se unutar sva tri repozitorija nalaze najnovija verzije koda na grani master. Svi repozitoriji sastoje se od dvije grane - *main* i *development*. *Development* grana razvojna je grana na kojoj se čuvaju izmjene u radu, one koje još nisu testirane odnosno provjerene. Jednom kad je kod na razvojnoj grani testiran i provjeren, tj. kad se smatra da neće biti dodatnih izmjena tada se sjedinjuje sa *main* granom na kojoj se čuva provjereni kod aplikacije. Prilikom sjedinjavanja razvojne grane s *main* granom razvojna grana se briše. Svakom novom izmjenom koda, prije objavljivanja koda na git granu, ponovno se iz *master* grane stvara razvojna grana na kojoj se kod objavljuje.



Slika 6.1 Verzioniranje koda

7. Sigurnosne razine

Kao članica EU, Hrvatska je 25. svibnja 2018., Zakonom osigurala provedbu Opće uredbe o zaštiti podataka [17]. Zakon je u nekim područjima nedorečen te tumačenje Zakona u području upravljanja zgradama za što je nadležna Agencija za zaštitu osobnih podataka, koja je osnovana da nadzire provođenje Zakona, odstupa od europske prakse.

Budući da je jednoobrazna primjena Zakona obavezujuća na cjelokupnom teritoriju EU, očekuje se da će najavljeni Zakon o upravljanju i održavanju zgrada precizno definirati prava i obaveze svih sudionika u održavanju zgrada i nedvosmisleno prezentirati tko i na koji način može imati uvid u osobne podatke suvlasnika [18]. Zakon bi trebao stupiti na snagu tijekom 2023.

Uzimajući u obzir opisanu situaciju aplikacija se u dijelu zaštite podataka priklonila rješenjima većine europskih zemalja. Osobni podaci neće biti javno dostupni, npr. objavljeni na web stranici zgrade ili upravitelja, ali suvlasnici mogu imati uvid u osobne podatke ostalih suvlasnika suvlasničke zajednice: ime i prezime, OIB, broj korisnika pojedinog posebnog dijela zgrade, broj članova obitelji ili broj podstanara u pojedinoj samostalnoj prostornoj cjelini zgrade. To se odnosi i na dugovanja pojedinih suvlasnika za pričuvu, režije i sl. a i na sve troškove pojedinog suvlasnika koji imaju utjecaja na ostale suvlasnike[18].

Povremeno, svaki suvlasnik može zatražiti i dodatne podatke o drugom suvlasniku npr. koliko plaća vodu, ali te podatke ne može dobivati na uvid kontinuirano više od nekoliko mjeseci za redom i može ih zatražiti najviše jednom ili dvaput godišnje.

Međuvlasničkim ugovorom mogu se precizno definirati sva ograničenja koja će onda predstavnik stanara prezentirati u općim podacima u zgradi. Zakonski minimum, kada Zakon bude donesen, bit će implementiran u aplikaciju bez mogućnosti umanjenja.

Predstavnik suvlasnika i upravitelj imaju dodatne mogućnosti korištenja osobnih podataka te ih mogu prikupljati, analizirati i na osnovi zaključaka predlagati suvlasnicima unapređenja u održavanju i upravljanju.

7.1. Autentifikacija

Autentifikacija je proces provjere identiteta nekog korisnika. Od korisnika se traže određeni podaci po kojima se može utvrditi da je korisnik upravo onaj kojim se predstavlja. Autentifikacija se provodi s pomoću lozinki, jednokratnih pinova, biometrije i sl. U nekim slučajevima sustavi zahtijevaju provjeru više od jednog čimbenika prije odobravanja pristupa. Zahtjev za provjeru autentičnosti s više faktora često se primjenjuje kako bi se povećala sigurnost izvan onoga što same lozinke mogu pružiti što u konkretnoj aplikaciji nije neophodno, ali se ta mogućnost ostavlja za eventualnu nadogradnju sustava [19].

Pri prijavi u aplikaciju, od korisnika se traži ime, prezime i lozinka. Ukoliko sustav pronađe korisnika s odgovarajućim podacima unutar baze podataka, korisnika se dalje propušta u aplikaciju i daju mu se određena prava, ovisno o tome kojoj vrsti korisnika pripada. U suprotnom, korisniku se vrati i ispisuje poruka o nastaloj pogrešci i moli ga se za provjeru unesenih podataka.

Sustav za provjeru, je li korisnik onaj kojim se predstavlja, te je li prijavljen, implementiran je unutar komponente filter. Filter se izvršava tijekom poziva kontrolera aplikacije, a prije ispunjavanja programske logike, tj. dohvata podataka pojedinog kontrolera. Unutar svakog autentifikacijskog filtera provjerava se je li korisnik onaj kojim se predstavlja i je li mu sesija važeća. Taj filter odnosi se na sve kontrolere osim kontrolera za prijavu i registraciju. Što znači da i neregistrirani korisnici mogu pristupiti tim sučeljima.

```
public class AuthenticationFilter implements Filter {
     private HttpServletRequest httpRequest;
     private HttpServletResponse httpResponse;
     private static final String[] loginRequiredURLs = {
        "/options", "/menu", "/myprofile",
        "/vote", "/building", "/tenant",
        "/expense", "/bug", "/news", "/debate", "/appartment",
        "/floor"
    };
    @Override
    public void doFilter(ServletRequest request, ServletResponse response,
            FilterChain chain)
            throws IOException, ServletException {
        httpRequest = (HttpServletRequest) request;
        httpResponse = (HttpServletResponse) response;
        HttpSession session = httpRequest.getSession(false);
        boolean isLoggedIn = (session != null &&
session.getAttribute("currentUser") != null);

        if (!isLoggedIn && isLoginRequired())
            httpResponse.addHeader("error", response.toString());
        else
            chain.doFilter(request, response);
    }
    private boolean isLoginRequired() {
        String requestURL = httpRequest.getRequestURL().toString();

        for (String loginRequiredURL : loginRequiredURLs) {
            if (requestURL.contains(loginRequiredURL))return true;
        }
         return false;
    }}
```

Kôd 7.1 Filter autentifikacije

7.2. Autorizacija

Autorizacija slijedi autentifikaciju. Autorizacija je proces određivanja prava pristupa resursima ili funkcijama. Tijekom rada, sustav koristi pravila kontrole pristupa kako bi odlučio hoće li zahtjevi za pristup korisnika biti odobreni ili odbijeni tj. hoće li korisnik moći pristupiti podacima na sučelju ili ne.

U ovoj aplikaciji. upotrijebljena je strategija autorizacije temeljena na ulogama, RBAC (engl. *Role-based access control*).

RBAC tretira autorizaciju kao dozvole povezane s ulogama, a ne izravno s korisnicima [20]. Uloga nije ništa drugo nego zbirka dopuštenja. Npr. predstavnik stanara, u ovoj aplikaciji, ima sukladno svojoj ulozi više dopuštenja od ostalih korisnika – stanara i upravitelja: mogućnost ažuriranja podataka o zgradi, objava novosti, dodavanje stanara u zgradu, promjena upravitelja i sl.

Autorizacija je implementirana s pomoću sesije. Provodi se na klijentskom sloju.

Pri prijavi spremaju se podaci o korisniku (osobni podaci te uloga koju obnaša). Pri svakom pozivu sučelja ili pojedine komponente iz sesije se dohvaćaju podaci o korisniku te se provjeravaju korisnikove uloge.

Ovisno o korisničkoj ulozi prikazuju se i odgovarajuća sučelja ili komponente.

Ukoliko uloga trenutno prijavljenog korisnika odgovara ulozi koja dopušta pregled traženih podataka, korisnika se propušta na sučelje ili mu se prikazuje tražena komponenta. Međutim, ukoliko korisnikova uloga ne odgovara ulozi potrebnoj za pregled sučelja ili komponente, korisniku se prikazuje druga komponenta vezana uz njegovu ulogu, ne propušta ga se na traženo sučelje te mu se blokira pristup traženim podacima.

```
[login.fulfilled]: (state, { payload }) => {
      state.userInfo = payload.value;
      state.updated = payload.value.account;
      state.userToken = true;
      state.success = true;
      sessionStorage.setItem('user', JSON.stringify(payload))
},
```

Kôd 7.2 Pospremanje podataka o korisniku u sesiju prilikom prijave

```
<Stats

        create={false}

        readonly={!sessionStorage.getItem("company") ? (user.role.Title ===
"PREDSTAVNIK STANARA" ?    false : true) : true}

        anybody={!sessionStorage.getItem("company") ? (user.role.Title ===
"PREDSTAVNIK STANARA" ?   true : false) : false}/>
```

Kôd 7.3 Prikaz komponenti ovisno o podacima spremljenim u sesiji

7.3. Zaštita osobnih podataka

Sigurnost podataka praksa je zaštite digitalnih informacija od neovlaštenog pristupa, slučajnog gubitka, otkrivanja i modifikacije, manipulacije ili oštećenja tijekom cijelog životnog ciklusa podataka.

Na taj način održava se povjerljivost, integritet i dostupnost podataka na aplikaciji. Povjerljivost se odnosi na čuvanje privatnosti podataka, integritet na osiguravanje pouzdanosti podataka te dostupnost za pružanje pristupa ovlaštenim subjektima.

7.3.1. Enkripcija

Svi osjetljivi podaci koji se spremaju u bazi, poput korisničkih lozinki, kriptirani su. Enkripcija podataka odvija se na klijentskom sloju prije slanja podataka i poziva servisa. Za enkripciju se koristi vanjska biblioteka bcrypt. Takvi podaci nisu čitljivi ljudskom oku unutar baze podataka.

7.3.2. Privremeno brisanje

Korisnici mogu u bilo kojem trenutku pobrisati svoje podatke i zatvoriti svoj korisnički račun. U tom slučaju u bazi se izvrši procedura za brisanje podataka temeljena na principu privremenog brisanja (engl. *soft delete)*. Korisnički podaci ostaju upisani u bazi čak i nakon brisanja korisničkog računa. Administratori sustava i dalje će vidjeti podatke, ali korisnik više neće moći pristupiti svom profilu. Tim principom omogućeno je lakše i brže poništavanje brisanja, praćenje povijesti (čuvanje izbrisanih podataka u svrhu revizije) te lakše usklađivanje tijekom oporavka od kvara.

7.3.3. Procedure

Radi opasnosti od SQL *injection*-a (sigurnosna ranjivost na webu koja napadaču omogućuje miješanje u upite koje aplikacija postavlja svojoj bazi podataka, pregled podataka koje inače ne smije i ne može dohvatiti, npr. podaci drugih korisnika ili bilo koji drugi podaci kojima sama aplikacija može pristupiti) u bazi su, za svaki entitet, implementirane CRUD procedure [21].

Prilikom unosa ili ažuriranja podataka logički sloj aplikacije poziva odgovarajuću proceduru kojoj predaje parametre po potrebi. Ti parametri moraju odgovarati tipu i veličini varijable

u čije se polje upisuju, a koje je definirano unutar upita baze tj. unutar razvojne okoline SSMS (engl. *SQL Server Management Studio*). Ukoliko procedura uoči nepravilnosti u parametrima vratit će iznimku logičkom sloju koji će ju potom uhvatiti i proslijediti klijentskom sloju.

```
create proc createBug
      @InProgress bit,
      @Title nvarchar(100),
      @Descr nvarchar(max),
      @Category nvarchar(50),
      @Active bit,
      @BuildingID int,
      @IDBug int out
as
insert into Bug(Title, Descr, Category, Active, BuildingID, InProgress)
values(@Title, @Descr, @Category, @Active, @BuildingID, 0)
set @IDBug = SCOPE_IDENTITY()
go

create proc selectBugs
      @BuildingId int
as
select*from Bug
where BuildingID = @BuildingId
go

create proc resolveBug
      @IDBug int
as
update Bug
set Active = 0
where IDBug = @IDBug
go

create proc deleteBug
      @IDBug int
as
delete from Bug
where IDBug = @IDBug
go

create proc changeProgress
      @IDBug int
as
update Bug
set InProgress = 1
where IDBug = @IDBug
go
```

Kôd 7.4 CRUD procedure na tablici Kvar

7.3.4. CORS mehanizam

Za većinu web stranica sva sredstva (slike, tekst, datoteke itd.) koja koriste nalaze se na istom poslužitelju na kojem je web stranica smještena. Budući da web stranici treba vjerovati poslužitelj koji ju pokreće, web stranici se daje dopuštenje za korištenje resursa koje drži domaćin poslužitelj. Međutim, kada sredstva koje traži stranica nisu dostupna na istom poslužitelju, kao što je to ovdje slučaj, onda se unutar HTTP zaglavlja navodi bilo koja domena osim vlastite s koje bi preglednik trebao dopustiti učitavanje resursa.

CORS mehanizam implementiran je unutar filtera naziva „CORSFilter“ koji se izvršava prije svakog poziva kontrolera. Pri izvršavanju filtera provjerava se putem koje metode (GET ili POST) je zahtjev za podacima poslan. Potom se u zaglavlje odgovora postavljaju željeni parovi varijabli – ključ i vrijednost. Pristup je dan svim uređajima putem ključa „Access-Control-Allow-Origin“ a i svim metodama putem ključa „Access-Control-Allow-Methods“.

```
public class CORSFilter implements Filter {


    public CORSFilter() {}

    public void destroy() {}


    public void doFilter(ServletRequest servletRequest,
ServletResponse servletResponse, FilterChain chain)

            throws IOException, ServletException {

        HttpServletRequest request = (HttpServletRequest)
servletRequest;

        System.out.println("CORSFilter HTTP Request: " +
request.getMethod());

        ((HttpServletResponse) servletResponse).addHeader("Access-
Control-Allow-Origin", "*");

        ((HttpServletResponse) servletResponse).addHeader("Access-
Control-Allow-Methods", "GET, OPTIONS, HEAD, PUT, POST");

        ((HttpServletResponse) servletResponse).setHeader("Access-
Control-Allow-Headers", "Content-Type, Access-Control-Allow-Headers,
Authorization, X-Requested-With");


        HttpServletResponse resp = (HttpServletResponse)
servletResponse;


        if (request.getMethod().equals("OPTIONS")) {

            resp.setStatus(HttpServletResponse.SC_ACCEPTED);

            return;

        }

        chain.doFilter(request, servletResponse);

    }


    public void init(FilterConfig fConfig) throws ServletException {

    }

}
```

Kôd 7.5 CORS filter

8. Funkcionalni zahtjevi

Funkcionalni zahtjev govori o tome 'što sustav treba raditi' i definirani su u skladu sa zahtjevima arhitekta, stanara, predstavnika i upravitelja.

Tablica 8.1 Pregled funkcionalnosti po ulogama

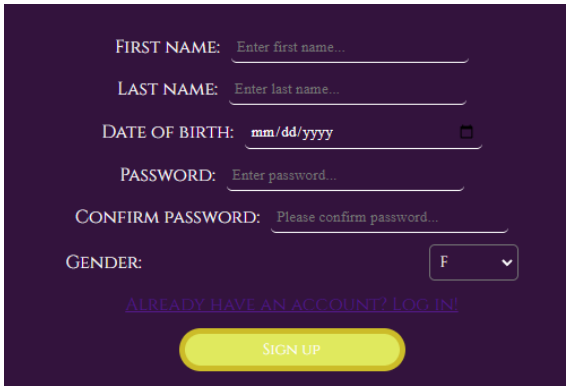
Funkcionalnost	Stanar	Predstvanik stanara	Upravitelj zgrade
Registracija	✓	✓	✓
Prijava	✓	✓	✓
Brisanje vlastitog korisničkog računa	✓	✓	✓
Uređivanje vlastitog korisničkog računa	✓	✓	✓
Pregled pridruženih zgrada	✓	✓	✓
Odabir zgrade	✓	✓	✓
Pridruživanje zgradi		✓	
Kreiranje zgrade	✓	✓	
Brisanje zgrade		✓	
Pregled zgrade	✓	✓	✓
Pregled novosti	✓	✓	✓
Prijava kvara	✓	✓	✓
Pregled kvarova	✓	✓	✓
Glasovanje	✓	✓	
Forum	✓	✓	✓

Pregled stanova i stanara	✓	✓	✓
Objava novosti		✓	✓
Riješavanje kvara		✓	✓
Brisanje kvara		✓	✓
Dodavanje stanara zgradi		✓	
Dodavanje upravitelja zgradi		✓	
Dodavanje stanara u stan		✓	
Ažuriranje zgrade		✓	
Ažuriranje troškova		✓	
Kreiranje stanova		✓	

8.1. Funkcionalnosti za stanare

Funkcionalnosti za stanare dostupne su svim korisnicima sustava. Napravljene su na temelju zahtjeva stanara zgrade i nude mogućnost korištenja aplikacije za različite svrhe.

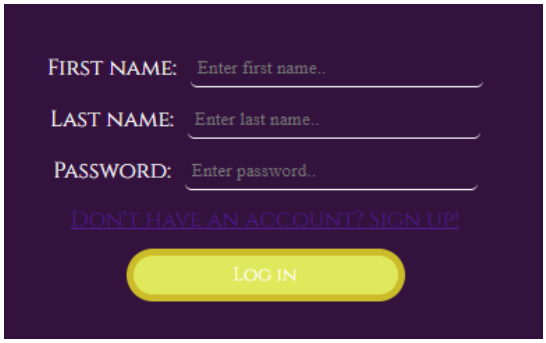
8.1.1. Registracija



Slika 8.1 Registracija

Registracija je obavezna za sve korisnike aplikacije. Pri prvom korištenju aplikacije od stanara se traži registracija u sustav, unosom obaveznih podataka - ime, prezime, OIB, godina rođenja, spol, godina početka boravljenja u zgradi, lozinka i potvrda lozinke. Ukoliko je registracija bila uspješna na ekranu će se pojaviti skočni prozor, zelene boje, s porukom o uspješnoj prijavi u sustav. U suprotnom na ekranu se pojavljuje skočni prozor s porukom o nastaloj pogrešci. Nakon uspješne registracije, korisnika se preusmjerava na izbornik aplikacije od kuda može koristiti sve funkcionalnosti namijenjene toj kategoriji korisnika.

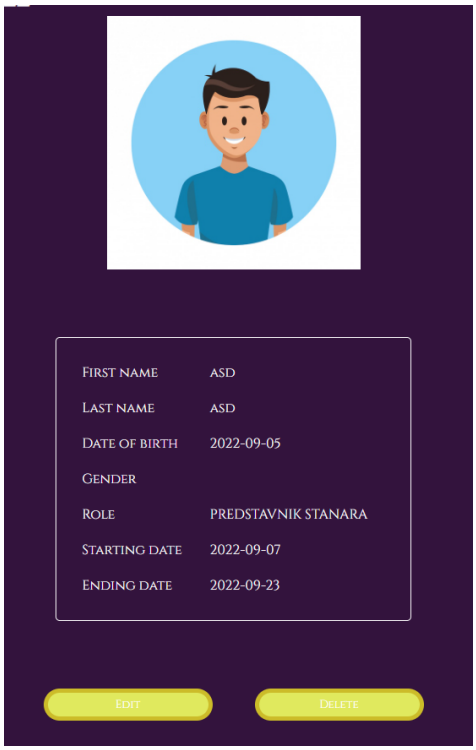
8.1.2. Prijava



Slika 8.2 Prijava

Tijekom svake naknadne prijave u sustav od korisnika se traže sljedeći podaci: adresa, ime, prezime i lozinka. Ukoliko je prijava bila uspješna na ekranu će se pojaviti skočni prozor, zelene boje, s porukom o uspješnoj prijavi u sustav. U suprotnom, na ekranu se pojavljuje skočni prozor s porukom o nastaloj pogrešci. Nakon uspješne prijave, korisnika se preusmjerava na izbornik aplikacije od kuda može koristiti sve funkcionalnosti, prema svojoj kategoriji i podacima dostupnim za pojedinu zgradu.

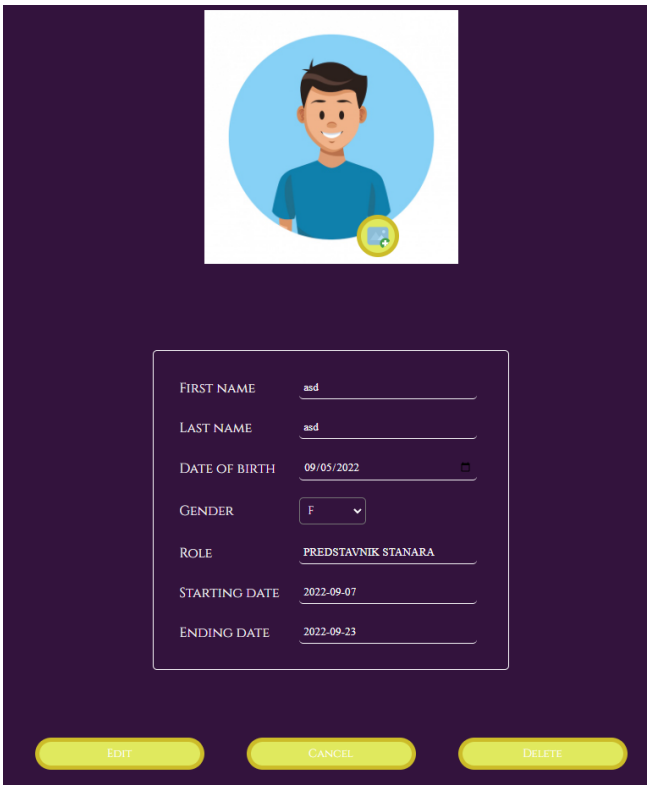
8.1.3. Brisanje vlastitog korisničkog računa



Slika 8.3 Brisanje vlastitog korisničkog računa

Odlaskom na stranicu korisničkog profila korisniku je dostupna funkcionalnost brisanja korisničkog računa. Prilikom brisanja, korisnički račun i svi povezani podaci, tj. sve objave pojedinog korisnika, ne brišu se iz baze nego ih se stavlja u stanje neaktivnosti. Nakon brisanja korisničkog računa korisnik ne može više pristupiti svome računu. Sve njegove objave ostaju na stranici, vidljive svim korisnicima.

8.1.4. Uređivanje vlastitog korisničkog računa



Slika 8.4 Uređivanje vlastitog korisničkog računa

Odlaskom na stranicu korisničkog profila korisniku je dostupno uređivanje korisničkog računa. Svi podaci vidljivi na korisničkom profilu mogu se uređivati, tj. mijenjati. Ukoliko se korisnik tijekom izmjene predomisli i želi zadržati postojeće podatke, od izmjene može odustati klikom na gumb „odustani“. U tom slučaju promjene se ne spremaju u bazu i podaci ostaju nepromijenjeni.

8.1.5. Pregled i odabir pridruženih zgrada

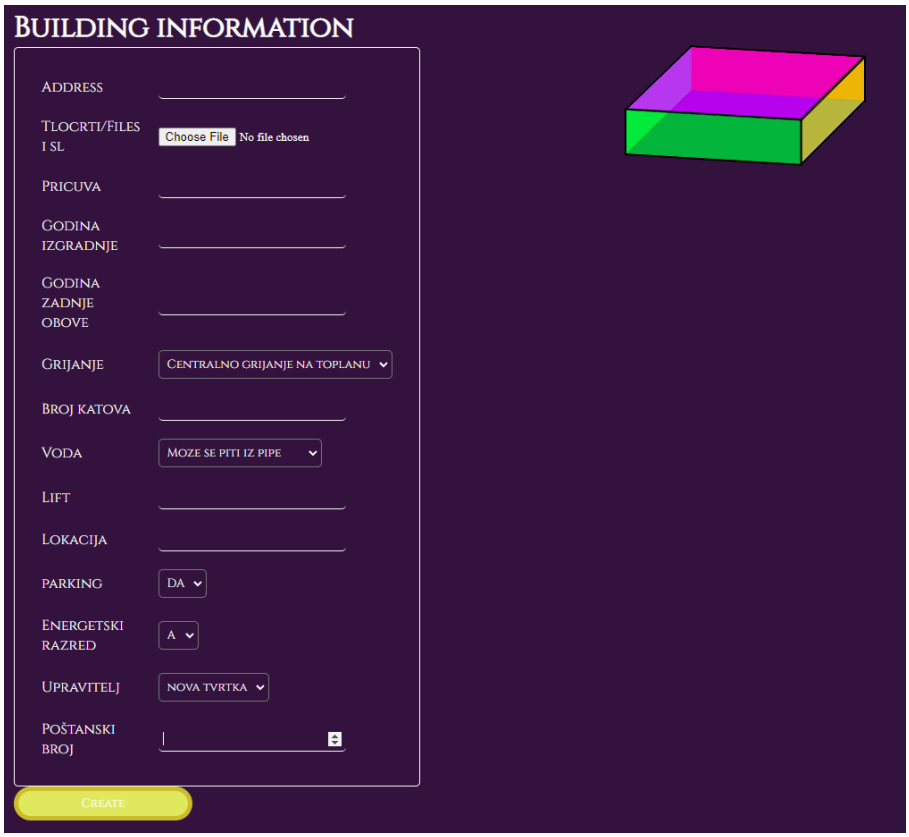
Slika 8.5 Pregled pridruženih zgrada

Ova funkcionalnost dostupna je na dva mjesta unutar aplikacije – odmah nakon prijave u sustav ili u izborniku.

Nakon uspješne prijave u sustav korisnika se preusmjerava na sučelje za pregled svih zgrada kojima je pridružen. Na sučelju je prikazana lista svih zgrada kojima je korisnik pridružen i još dvije mogućnosti. Klikom na neku od zgrada s popisa, unutar sučelja za pregled svih pridruženih zgrada, korisnik se preusmjerava na izbornik putem kojeg su mu, u skladu s njegovom ulogom, dostupni svi podaci i funkcionalnosti za odabranu zgradu.

Tijekom korištenja aplikacije korisnik se može vraćati na sučelje za pregled zgrada koliko god puta želi. To čini putem izbornika, klikom na karticu „Moje zgrade“.

8.1.6. Kreiranje zgrade



Slika 8.6 Kreiranje zgrade

Ova funkcionalnost moguća je za sve registrirane korisnike svih kategorija. Pri kreaciji nove zgrade potrebno je unijeti informacije o zgradi – adresu, poštanski broj, broj katova, broj dizala, način grijanja, nacrte, može li se voda piti iz slavine ili ne, godina izgradnje, godina zadnje obnove, slike, naziv upravitelja zgrade i podaci o predstavniku stanara ako postoji. Predstavnik stanara informacija je koju se pri kreaciji zgrade ne mora unositi. Ukoliko zgrada s upisanom adresom već postoji u aplikaciji, ona neće biti ponovno kreirana.

8.1.7. Brisanje zgrade

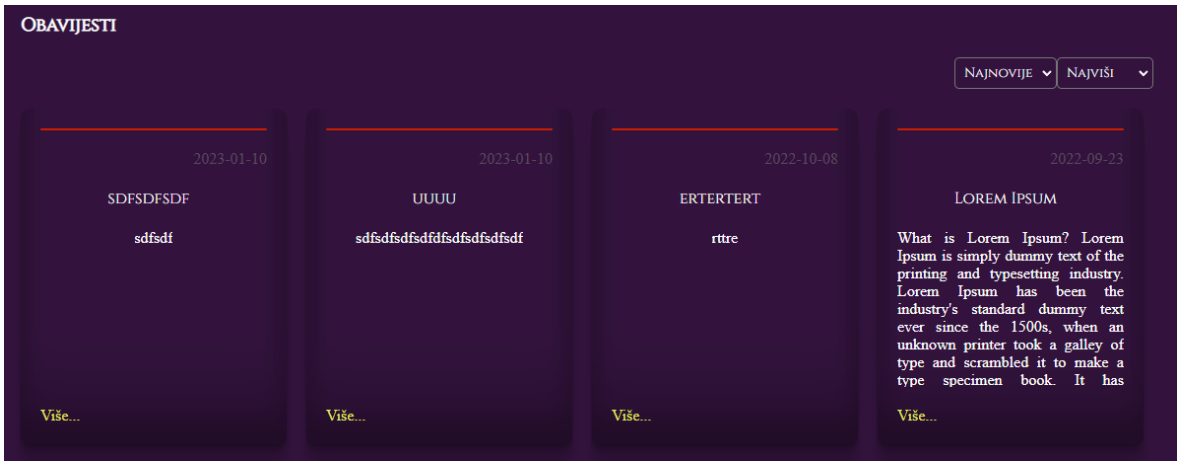
Brisanje zgrade može pokrenuti samo predstavnik stanara te zgrade. Ukoliko se predstavnik stanara odluči za brisanje zgrade iz sustava, pokreće se obavezno glasovanje. Glasovati moraju svi registrirani stanari zgrade. Ako rezultat glasovanja ne bude 100% za brisanje zgrade, zgrada neće biti obrisana iz sustava. Međutim, ako rezultat glasovanja bude 100% za brisanje zgrade, zgrada će se obrisati iz sustava, po principu *soft delete*, te više neće biti

vidljiva niti na listi pridruženih zgrada nekog korisnika niti na listi zgrada evidentiranih u aplikaciji «Slika 8.10.

8.1.8. Pregled zgrade

Opće informacije, unesene pri kreaciji, pojedine zgrade dostupne su na aplikaciji. One su vidljive svim stanarima te zgrade, predstavniku stanara i upravitelju zgrade. Osim općih informacija unutar sučelja za pregled zgrade bit će i trodimenzionalni animirani model zgrade «Slika 8.6.

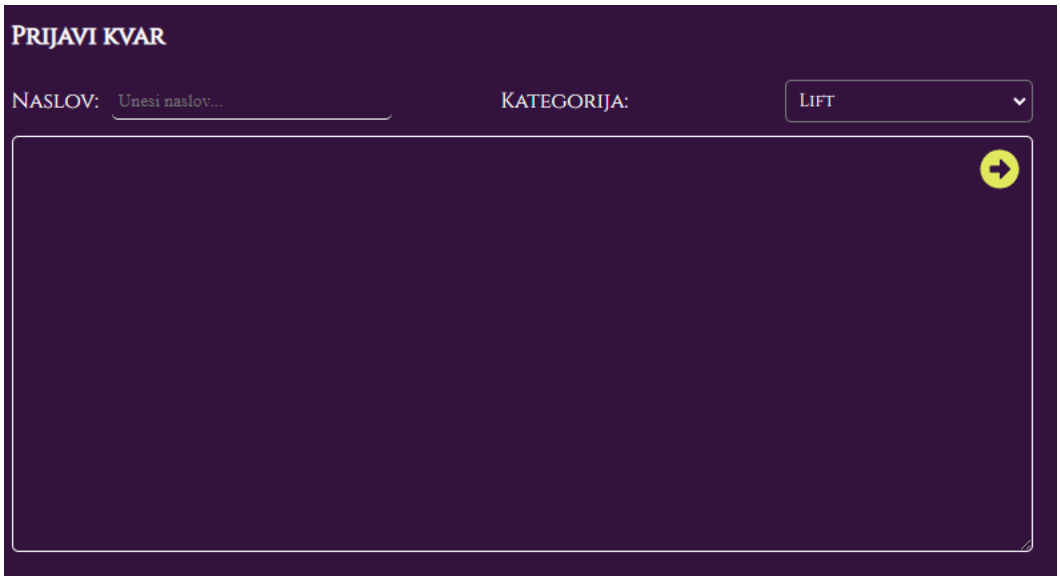
8.1.9. Pregled novosti



Slika 8.7 Pregled novosti

Novosti o pojedinoj zgradi dostupne su svim korisnicima unutar aplikacije. Mogu se filtrirati s pomoću tri vrste filtara – prioritet, primatelj i datum. Svaka novost prikazana je na posebnoj kartici. Novosti koje su namijenjene samo određenoj grupi ljudi kao primatelju bit će vidljive samo tim korisnicima i predstavniku stanara. Ostali korisnici neće vidjeti tu novost.

8.1.10. Prijava kvara



Slika 8.8 Prijava kvara

Ukoliko dođe do kvara na zgradi, korisnici ga mogu prijaviti putem obrasca za prijavu kvarova. Pri ispunjavanju obrasca za prijavu kvarova od korisnika se traži unos naslova, opis kvara te kategorija kvara. Svi podaci su obavezni. Ukoliko neki podatak nedostaje obrazac nećebiti spremljen i neće biti vidljiv na popisu kvarova. Ukoliko je prijava kvara uspješna, prijava će biti vidljiva svim kategorijama korisnika unutar zgrade.

8.1.11. Pregled kvarova

Slika 8.9 Pregled kvarova

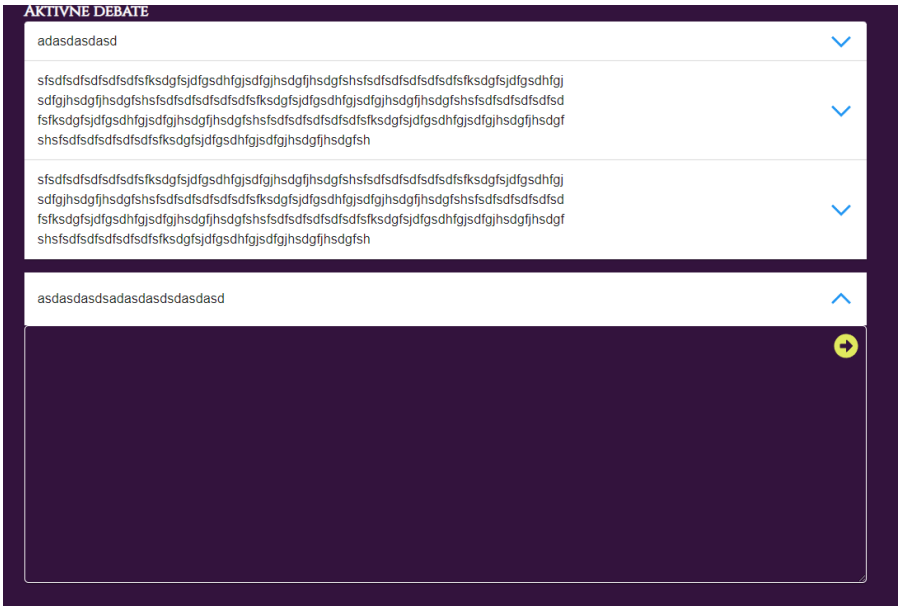
Kvarovi su podijeljeni u aktivne i riješene kvarove. Aktivni kvarovi vidljivi su svim kategorijama korisnika dok su riješeni kvarovi vidljivi samo predstavniku stanara i upravitelju zgrade.

8.1.12. Glasovanje

Slika 8.10 Glasovanje

Pokrenuti glasovanje mogu stanari i predstavnik stanara. Prilikom pokretanja glasovanja potrebno je ispuniti obrazac za kreiranje glasovanja. Unutar obrasca treba unijeti naslov, opis, opcije glasanja i vremensko ograničenje. Vremensko ograničenje može biti neograničeno ili ograničeno. Ukoliko je vremensko ograničenje postavljeno na neograničeno, glasači mogu glasati sve dok korisnik koji je kreirao glasovanje ručno ne zatvori glasovanje. U slučaju da je glasovanje vremenski ograničeno, glasači mogu glasovati samo u danom vremenskom roku. Nakon isteka roka glasovanje se automatski zatvara.

8.1.13. Forum



Slika 8.11 Forum

Na stranici za debate vidljiva je lista svih debata – aktivnih i neaktivnih. Debate mogu pokrenuti svi korisnici aplikacije zgrade. Prilikom pokretanja debate, potrebno je ispuniti obrazac za kreiranje debate. U obrazac se unosi tema i rok trajanja. Ukoliko korisnik želi

sudjelovati u pojedinoj debati to može učiniti upisom svog komentara u prostor za komentare unutar pojedine debate. Isto tako svi korisnici mogu neki komentar označiti sa „sviđa mi se“.

U raspravama mogu sudjelovati svi stanari zgrade, upravitelj i predstavnik stanara.

8.1.14. Pregled stanova i stanara

Slika 8.12 Pregled stanova i stanara

Na stranici za pregled općih informacija zgrade dostupan je i trodimenzionalni prikaz modela zgrade. Klikom na neki od katova unutar modela korisniku se prikazuje sučelje sa popisom svih stanova na odabranom katu. Klikom na neki od stanova otvara se sučelje s prikazom više informacija o stanu - orijentacija stana, površina, ima li vodomjer, popis svih stanara.

8.2. Funkcionalnosti za predstavnika stanara

Predstavniku stanara dostupne su sve funkcionalnosti dostupne stanarima te dopunske vezane za njegovu ulogu, navedene i opisane u nastavku.

8.2.1. Objava novosti

Slika 8.13 Objava novosti

Pri objavi novosti treba unijeti naslov novosti, opis novosti, primatelja i prioritet novosti. Svi podaci obavezni su za uspješnu kreaciju novosti.

8.2.2. Rješavanje kvara

Slika 8.14 Rješavanje kvara

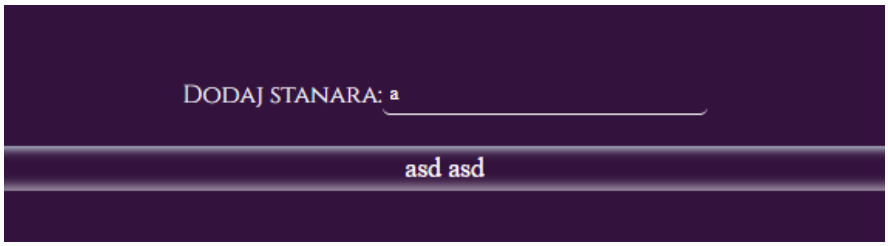
Pokrenuti rješavanje prijavljenog kvara mogu predstavnik stanara i upravitelj zgrade. Tijekom rješavanja, na kartici problema prikazana je animacija koja označava da je rješavanje problema u tijeku. Po završetku radova i otklanjanju kvara, predstavnik stanara ili upravitelj zgrade označavaju problem kao riješen. Problem se automatski uklanja s liste aktivnih kvarova i sprema pod listu riješenih kvarova.

8.2.3. Brisanje kvara

Slika 8.15 Brisanje kvara

Prijavljeni kvar mogu izbrisati predstavnik stanara ili upravitelj zgrade. Pri brisanju, korisniku koji je pokrenuo brisanje, iskače poruka o potvrdi brisanja. Ukoliko ju korisnik potvrdi, kvar se briše te prestaje biti vidljiv na popisu aktivnih kvarova. Međutim, ukoliko korisnik odbije brisanje, ili u slučaju nenamjernog pokretanja brisanja, kvar se neće obrisati i bit će i dalje vidljiv na listi kvarova.

8.2.4. Dodavanje stanara zgradi



Slika 8.16 Dodavanje stanara zgradi

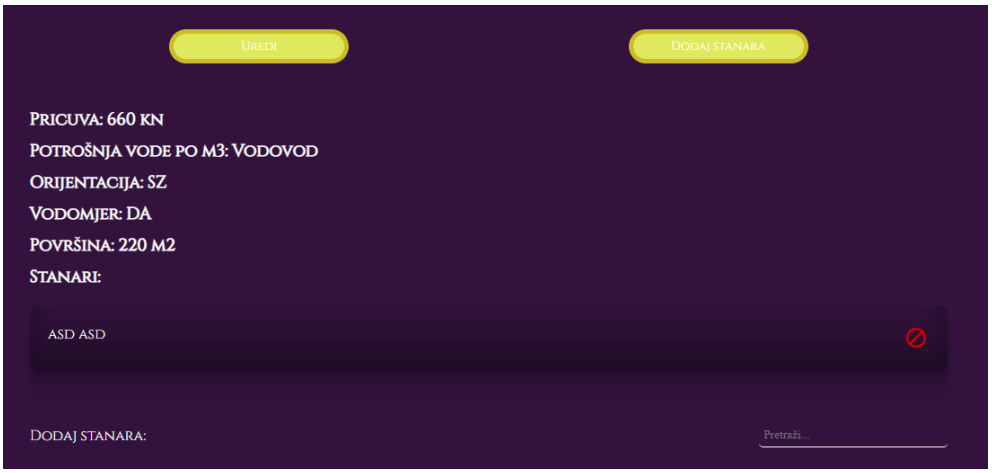
Nove stanare unutar zgrade može dodati samo predstavnik stanara. Unutar aplikacije pojedine zgrade vidljivi su svi korisnici koji još nisu dodani u zgradu. Odabirom nekog korisnika s popisa šalje mu se upit o pridruženju zgradi. Upit sadrži informacije o zgradi u koju ga se želi dodati te ime i prezime predstavnika stanara. Korisnik taj upit može odbiti ili prihvatiti. Ukoliko korisnik prihvati upit za pridruženjem zgradi, od tog trenutka njemu ta zgrada postaje vidljiva na listi pridruženih zgrada stanar može koristiti sve funkcionalnosti te zgrade. Ukoliko korisnik odbije zahtjev za pridruživanjem zgradi, neće biti dodan u aplikaciju zgrade.

8.2.5. Dodavanje upravitelja zgradi

Slika 8.17 Dodavanje upravitelja zgradi

Unutar aplikacije pojedine zgrade vidljivi su svi prijavljeni upravitelji zgrada u sustavu. Odabirom nekog upravitelja s popisa šalje mu se upit o pridruženju zgradi. Upit sadrži informacije o zgradi u koju ga se želi dodati. Upravitelj taj upit može odbiti ili prihvatiti. Ukoliko upravitelj prihvati upit za pridruženjem zgradi, njemu ta zgrada postaje vidljiva na listi pridruženih zgrada te može koristiti sve funkcionalnosti vezane uz tu zgradu. Međutim, ukoliko upravitelj odbije zahtjev za pridruživanjem, neće biti dodan u aplikaciju zgrade.

8.2.6. Dodavanje stanara u stan



Slika 8.18 Dodavanje stanara u stan

Unutar aplikacije, na sučelju za pregled stanova po katu, vidljivi su svi korisnici u zgradi. Odabirom nekog korisnika s popisa dodaje ga se u stan. Od tog trenutka korisnik je vidljiv na popisu korisnika stana. Dodavanjem stanara u stan povećavaju se troškovi tog stana.

8.2.7. Ažuriranje zgrade

Slika 8.19 Ažuriranje zgrade

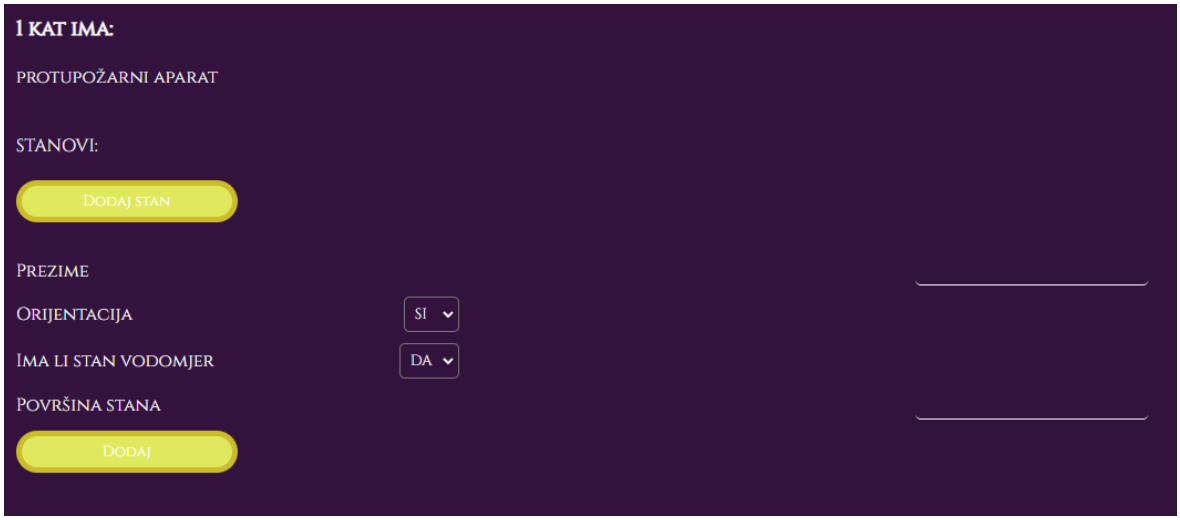
Ažurirati se mogu adresa, poštanski broj, broj katova, broj dizala, način grijanja, nacrti, podaci o kvaliteti vode, godina zadnje obnove, slike i upravitelj zgrade. Korisnik može u bilo kojem trenutku odustati od unesenih promjena pri čemu se one neće spremiti u bazu.

8.2.8. Ažuriranje troškova

Slika 8.20 Ažuriranje troškova

Pregled svih troškova pojedine zgrade jedna je od opcija dostupna unutar menija aplikacije. Vidljivi podaci su dogovoreni iznos pričuve te lista troškova koje zgrada ima. Troškovi se mogu dodavati i brisati po potrebi. Na stranici je također s pomoću grafičkog prikaza, slikovito prikazan udio pojedinog troška zgrade.

8.2.9. Kreiranje stanova



Slika 8.21 Kreiranje stanova

Na stranici za pregled općih informacija zgrade dostupan je i trodimenzionalni prikaz modela zgrade. Klikom na neki od katova unutar modela korisniku se prikazuje sučelje s popisom svih stanova na odabranom katu i gumb za kreiranje novog stana. Klikom na gumb za kreiranje novog stana otvara se forma za unos podatak o novom stanu – prezime suvlasnika, orijentacija stana, površina, ima li stan vodomjer isl.

8.3. Funkcionalnosti za upravitelja zgrade

Upravitelj zgrade ima pristup svim funkcionalnostima stanara, osim glasovanja. Uz te funkcionalnosti dostupne su mu i funkcionalnosti objave novosti i brisanje i rješavanje kvarova.

Mobilna aplikacija namijenjena je stanarima i predstavniku stanara zgrade. Mobilnu aplikaciju ne mogu koristiti upravitelji zgrada. Upravitelji zgrada smatraju se pravnim subjektima te su im sve funkcionalnosti dostupne na web aplikaciji, radi preglednosti sadržaja.

Tablica 8.2 Prikaz vrsta korisnika koji mogu koristiti aplikacije

	Web aplikacija	Mobilna aplikacija
Stanar	✓	✓
Predstavnik stanara	✓	✓
Upravitelj zgrade		

9. Nefunkcionalni zahtjevi

Nefunkcionalni zahtjevi objašnjavaju aspekte kvalitete sustava koji će se graditi (performanse, prenosivost, upotrebljivost) te govore o tome kakav sustav treba biti. Izvedeni su iz funkcionalnih zahtjeva na temelju uloga dionika.

Nefunkcionalni zahtjevi unutar ove aplikacije su:

- lozinka mora sadržavati 8 znakova, od kojih najmanje jedno malo i veliko slovo, najmanje jedan broj i posebni simbol (.,?/()&%)
- korisnička sesija traje 30 minuta nakon čega korisnik ponovno mora unijeti podatke za prijavu
- svaka zgrada može imati samo jednog predstavnika stanara
- svaka zgrada može imati samo jednog upravitelja
- polja za unos smiju sadržavati samo slova i brojeve

10. Osvrt na realizaciju sustava i moguća poboljšanja

Od trenutka kad se javila ideja o aplikaciji za zgrade, opcije koje bi aplikacija mogla ponuditi korisnicima rasle su iz dana u dan. Odlučeno je, u prvom koraku, digitalizirati sve postojeće funkcije sustava za upravljanje i održavanje zgrada, njihovo sređivanje prema tehničkim kriterijima djelatnosti, prema ekonomskim kriterijima upravljanja i prema socijalnim kriterijima. Olakšavanjem komunikacije očekuje se aktivacija potencijalnih korisnika unutar pojedine zajednice, rad na unapređenju zajedničkih prostora i humanizaciji sredine u kojoj provode velik dio vremena. Kao djelatnost koja se tek počela razvijati u društvu, održavanje zgrada je nedovoljno regulirano radi čega često dolazi do blokiranja aktivnosti neophodnih za kvalitetno upravljanje. Izmjene i dopune regulative su kontinuirane. Zbog toga je u izradi aplikacije ostalo mjesta dogradnjama, izmjenama i dopunama koje će biti nužne radi izmjena regulative kao i radi sugestija korisnika.

Tijekom izrade aplikacije odvijala se komunikacija s dvoje predstavnika stanara, više aktivista iz ove djelatnosti te je njihovo viđenje situacije usmjeravalo na konkretna rješenja u aplikaciji. Zahvaljujući stečenom znanju na fakultetu, smjer programsko inženjerstvo, pretakanje govornog jezika, želja i sugestija u informatičke jezike (programske kodove) teklo je glatko. Cilj koji je postavljen ostvaren je u cijelosti, a dodatna poboljšanja izvedena su uz pomoć mentora.

Zaključak

Tema završnog rada je izrada sustava koji bi digitalizirao praćenje aktivnosti stambenih zgrada (ali i cijelog stambenog fonda). Prioritet je na održavanju zgrada, a uz to aplikacija će omogućiti ažurnu evidenciju stanara, pravednu podjelu troškova, stvaranje male interesne zajednice zgrade koja će unaprijediti zadovoljenje svakodnevnih potrebe stanara, olakšati rad predstavnika stanara i upravitelja zgrade te ostvariti potpunu transparentnost podataka i aktivnosti pojedine stambene zgrade.

Trenutno je u Hrvatskoj oko 85% stanova u privatnom vlasništvu dok se još pred 30-tak godina taj postotak stanova odnosio na društveno vlasništvo. Promjenom regulative država je odustala od vođenja stambene politike i prepustila rješavanje tog egzistencijalnog pitanja svakom pojedincu ponaosob, odnosno privatnom sektoru, odnosno tržištu. Promjena je bila nagla i ostavila je mnoga područja pravno neregulirana. Jedna od posljedica takva zaokreta u stambenoj politici je propadanje stambenog fonda. Naime, svijest o nužnosti održavanja zgrade od strane suvlasnika pojedinih etažnih dijelova ne postoji, a nije bilo ni zakonske obaveze koja bi vlasnike prisilila da održavaju zajedničke dijelove. Tek kada su zgrade počele pokazivati fizičko propadanje i time narušavati kvalitetu stanovanja počela se probijati misao o odgovornosti suvlasnika za stanje zgrade čiji su etažni vlasnici.

Ovom aplikacijom želi se omogućiti proaktivan stav svih zainteresiranih i potaknuti postupanja u skladu s europskom praksom.

Aplikacija je namijenjena stanarima, predstavnicima stanara i upraviteljima.

Stanari su fizičke ili pravne osobe koje se prijavljuju kao suvlasnici ili kao osobe koje u određenom razdoblju žive u pojedinoj zgradi.

Predstavnici stanara su fizičke osobe koju stanari pojedine zgrade biraju da ih predstavlja pred upraviteljem i trećim stranama. U aplikaciji oni imaju pristup osnovnim funkcionalnostima.

Predstavnik stanara, prema zakonu i suvlasnik u zgradi, može se koristiti svim funkcionalnostima dostupnim stanarima i dodatno funkcionalnostima namijenjenim samo za predstavnika stanara.

Upravitelj zgrade, kojeg prema zakonu mora imati svaka zgrada, pravna je osoba koja izvodi sve popravke na zgradi, brine o redovnom održavanju i investicijama u poboljšavanje uvjeta života u zgradi. Upravitelji zgrade, kao korisnici aplikacije, imaju mogućnost koristiti sve funkcionalnosti stanara, osim glasovanja,i neke od funkcionalnosti predstavnika stanara.

Aplikacija zahtijeva unos osobnih podataka korisnika radi čega je sigurnost aplikacije prioritet. Upotrijebljeni sigurnosni protokoli su autorizacija, autentifikacija, enkripcija, procedure i sigurnosne politike te je korištenje svih funkcionalnosti aplikacije sigurno i ležerno.

Za izradu aplikacije korištena je troslojna arhitektura, klijent-poslužiteljska arhitektura u kojoj se programska logika, baza podataka i korisničko sučelje razvijaju i održavaju kao neovisni moduli na zasebnim platformama. Ovakav tip arhitekture omogućava nadogradnju svakog sloja ponaosob, neovisno o drugim slojevima. Time je omogućen razvoj aplikacije u svim smjerovima u kojima se tijekom upotrebe pokaže potreba.

Tijekom rada na aplikaciji nastojalo se sagledati postojeću situaciju u području održavanja zgrada sa što više strana. Ali, kao pionirski projekt u ovom području bilo je važno napraviti aplikaciju koja se, tijekom vremena, može jednostavno dograđivati i u koju se bez velikog napora može implementirati buduća zakonska regulativa. U tom smislu, ova aplikacija je zamišljena kao temelj na kojem se može razvijati i izgraditi cjeloviti sustav praćenja sveukupnog stambenog fonda.

Popis kratica

API	*Application Programming Interface*	sučelje za programiranje aplikacija
CORS	*Cross-Origin Resource Sharing*	mehanizam dijeljenja resursa
CRUD	*Create, Read, Update and Delete*	stvaranje, čitanje, ažuriranje i brisanje
GSON	*Google JavaScript Object Notation*	Google JavaScript objektna notacija
HTML	*HyperText Markup Language*	hipertekst prezentacijski jezik
IDE	*Integrated Development Environment*	integrirano razvojno okruženje
JDBC	*Java Database Connectivity*	povezivost Jave i baze podataka
JSON	*JavaScript Object Notation*	JavaScript objektna notacija
NF	*Normal form*	forma normalizacije
RBAC	*Role-based access control*	kontrola pristupa temeljena na ulozi
REST	*Representational State Transfer*	Prijenos reprezentativnog stanja
SQL	*Structured Query Language*	strukturirani upitni jezik
SSMS	*SQL Server Management Studio*	Studio za upravljanje poslužiteljem
T-SQL	*Transact Structured Query Language*	transakcijski strukturirani upitni
URL	*Uniform Resource Locator*	jedinstveni identifikator izvora

Popis slika

Popis tablica

Popis kôdova

Literatura

[1] NN 43/92, Zakon o prodaji stanova nad kojima postoji stanarsko pravo, https://www.zakon.hr/z/275/Zakon-o-prodaji-stanova-na-kojima-postoji-stanarsko-pravo, lipanj 1992.

[2] NN 81/2015, Zakon o vlasništvu i drugim stvarnim pravima, https://narodne-novine.nn.hr/clanci/sluzbeni/2015_07_81_1548.html, prosinac 2022.

[3] NN 122/2014, Pravilnik o održavanju građevina, https://narodne-novine.nn.hr/clanci/sluzbeni/2014_10_122_2343.html, prosinac 2022.

[4] Kristijan Čulina, Dizajn i implementacija troslojne arhitekture pri razvoju aplikacija za Android, Završni rad, Sveučilište u Zagrebu, 2022.

[5] Mdn web docs, SPA (Single-page application), https://developer.mozilla.org/en-US/docs/Glossary/SPA, siječanj 2023.

[6] Microsoft 365, Description of the database normalization basics, https://learn.microsoft.com/en-us/office/troubleshoot/access/database-normalization-description, prosinac 2023.

[7] Mdn web docs, HTML: HyperText Markup Language, https://developer.mozilla.org/en-US/docs/Web/HTML, siječanj 2023.

[8] React, React Documentation, https://reactjs.org/docs/getting-started.html, prosinac 2023.

[9] NN 64/1997, Uredba o održavanju zgrada, https://narodne-novine.nn.hr/clanci/sluzbeni/1997_06_64_1002.html, prosinac 2022.

[10] REST API Tutorial, Using HTTP Methods for RESTful Services, https://www.restapitutorial.com/lessons/httpmethods.html, prosinac 2023.

[11] Geeksforgeeks, Factory Method design pattern in Java, https://www.geeksforgeeks.org/factory-method-design-pattern-in-java/, prosinac 2023.

[12] Brilliant, Classes (OOP), https://brilliant.org/wiki/classes-oop/, prosinac 2023.

[13] GeeksforGeeks, Lazy Loading Design Pattern, https://www.geeksforgeeks.org/lazy-loading-design-pattern/, 2013.

[14] GEEKSFORGEEKS, Singleton Design pattern, https://www.geeksforgeeks.org/singleton-design-pattern/, prosinac 2023.

[15] IBM, Primary Keys, https://www.ibm.com/docs/en/iodg/11.3?topic=reference-primary-keys, 2014.

[16] GIT, Git Documentation, https://git-scm.com/, 2023.

[17] GENERAL DATA PROTECTION REGULATION, uredba 2016/679 Europskog parlamenta i Vijeća, travanj 2016.

[18] ZAKON.HR, Opća uredba o zaštiti podataka, https://www.zakon.hr/z/3112/Op%C4%87a-uredba-o-za%C5%A1titi-podataka---Uredba-%28EU%29-2016-679-, siječanj 2023.

[19] STRONGDM, The Definitive Guide to Authentication, https://www.strongdm.com/authentication, 2023.

[20] AUTH0 DOCS, Role-Base Access Control, https://auth0.com/docs/manage-users/access-control/rbac, prosinac 2022.

[21] OWASP, SQL Injection, https://owasp.org/www-community/attacks/SQL_Injection, prosinac 2023.

ALGEBRA
VISOKO
UČILIŠTE

NASLOV ZAVRŠNOG RADA

Pristupnik: Nina Barbarić, 0321013640

Mentor: struč.spec.ing.comp. Daniel Bele