

# Oblikovanje programske potpore

Ak. god. 2018./2019.

## *Moj Kvart*

Dokumentacija, Rev. 2

Grupa: *Javoljupci*

Voditelj: *Mate Gašparini*

Datum predaje: *17. siječnja 2019.*

Nastavnik: *Nikolina Frid*

## Sadržaj

1. Dnevnik promjena dokumentacije	3
2. Opis projektnog zadatka	5
3. Pojmovnik	7
4. Funkcionalni zahtjevi	9
4.1. Dijagrami obrazaca uporabe	15
4.2. Sekvencijski dijagrami	19
5. Ostali zahtjevi	24
6. Arhitektura i dizajn sustava	25
6.1. Svrha, opći prioriteti i skica sustava	25
6.2. Dijagram razreda s opisom	29
6.3. Dijagram objekata	31
6.4. Ostali UML dijagrami	32
7. Implementacija i korisničko sučelje	42
7.1. Dijagram razmještaja	42
7.2. Korištene tehnologije i alati	43
7.3. Isječak programskog koda vezan za temeljnu funkcionalnost sustava	44
7.4. Ispitivanje programskog rješenja	46
7.5. Upute za instalaciju	55
7.6. Korisničke upute	60
8. Zaključak i budući rad	68
9. Popis literature	69
Dodatak A: Indeks (slika, dijagrama, tablica, ispisa kôda)	70
Dodatak B: Dnevnik sastajanja	72
Dodatak C: Prikaz aktivnosti grupe	73
Dodatak D: Pregled rada i stanje ostvarenja	75

## 1. Dnevnik promjena dokumentacije

Rev.	Opis promjene/dodatka	Autor(i)	Datum
0.1	Popisani dionici, razrađeni aktori i njihovi funkcionalni zahtjevi te napravljeno prvih sedam use-caseova.	Assouline, Dukić	22.10.2018.
0.2	Napisan opis projektnog zadatka.	Dodigović	22.10.2018.
0.21	Dodani use-caseovi od osmog do dvadesetog.	Dukić, Assouline	23.10.2018.
0.22	Poboljšani obrasci uporabe.	Dukić, Assouline	23.10.2018.
0.23	Sistematizirani obrasci uporabe.	Dukić	26.10.2018.
0.3	Dodani dijagrami obrazaca uporabe.	Kršić, Kurtović	27.10.2018.
0.31	Dodani sekvencijski dijagrami.	Kršić, Kurtović	27.10.2018.
0.32	Obrisane funkcionalnosti vezane uz sustav poruka kod obrazaca uporabe.	Dukić	28.10.2018.
0.33	Izmijenjeni dijagrami obrazaca uporabe i dodan sekvencijski dijagram.	Kršić	3.11.2018.
0.34	Dodani dionici, poboljšane definicije funkcionalnih zahtjeva.	Assouline, Dukić	3.11.2018.
0.35	Kombiniranje sekvencijskih dijagramima.	Kurtović	5.11.2018.
0.4	Dodani ostali zahtjevi.	Gašparini	5.11.2018.
0.5	Dodana skica arhitekture sustava.	Dodigović	5.11.2018.
0.6	Dodana skica baze podataka.	Gašparini	15.11.2018.
0.61	Dodan opis baze podataka.	Kurtović	22.11.2018.
0.7	Dodan dijagram razreda.	Dodigović	23.11.2018.
0.71	Dodan dijagram objekata.	Kršić	24.11.2018.
0.8	Sitne izmjene baze podataka i oblikovanja.	Gašparini	24.11.2018.
0.81	Izmjena dijagrama razreda.	Gašparini	26.11.2018.
0.9	Dodani pojmovi u pojmovnik.	Dukić	27.11.2018.

0.91	Dodan indeks.	Assouline	27.11.2018.
0.92	Sitne pravopisne izmjene.	Assouline	27.11.2018.
0.93	Izmjena dijagrama objekata.	Kršić	27.11.2018.
0.94	Dodan opis sekvencijskih dijagrama.	Kršić	27.11.2018.
1.0	Konačna verzija prve revizije.	Gašparini	28.11.2018.
1.1	Dodani dijagrami stanja i razmještaja.	Kurtović	9.1.2018.
1.2	Dodani dijagrami komunikacije i aktivnosti.	Kršić	12.1.2019.
1.3	Dodani dijagrami komponenti, instalacijske upute i kratak opis važnih dijelova koda.	Assouline	16.1.2019.
1.4	Dodane korisničke upute.	Kršić	17.1.2019.
1.5	Ispravljene neke greške u dokumentaciji.	Dukić	17.1.2019.
1.6	Dodan opis korištenih tehnologija i alata.	Gašparini	17.1.2019.
1.7	Dodano ispitivanje programskog rješenja.	Dukić	17.1.2019.
1.71	Manje izmjene u korisničkim uputama.	Kršić	17.1.2019.
1.8	Proširene instalacijske upute.	Gašparini	17.1.2019.
1.81	Dodani opisi slika koji nedostaju, usklađeni formati opisa slika, ažurirana tablica slika.	Dukić	17.1.2019.
1.82	Ispuna pojmovnika.	Kršić	17.1.2019.
1.9	Dodan zaključak.	Gašparini	17.1.2019.
1.91	Dodan dnevnik sastanaka i pregled rada; sitne izmjene.	Gašparini	17.1.2019.
2.0	Konačna verzija druge revizije.	Gašparini	17.1.2019.

## 2. Opis projektnog zadatka

Cilj ovog projekta je izrada web aplikacije koja će stanovnicima unutar odabrane gradske četvrti omogućiti bolju komunikaciju, lakšu razmjenu informacija te učinkovitiju organizaciju događaja.

Iako ljudi na dnevnoj razini prolaze pokraj svojih susjeda često ih niti ne upoznaju te time propuštaju priliku za moguću suradnju, druženje ili razmjenu korisnih informacija. Osim društvenog kontakta, stanovnike četvrti vežu stvari koje se tiču svakog pojedinačno. Samo neki od primjera su izvješće sa sjednice Vijeća četvrti, uređenje kvartovske okoline, mogući problemi oko zbrinjavanje otpada ili slično. Također, web aplikacija je prikladno mjesto za početak ostvarivanja odnosa na prethodno opisanim razinama zato što se ljudi zainteresirani za ostvarivanje komunikacije ujedinjuju na jednom mjestu i time si daju mogućnost za daljnji napredak u odnosima u stvarnom životu.

Stanovnik četvrti pri otvaranju računa upisuje osnovne informacije na temelju kojih se stvara njegov korisnički račun. Informacije potrebne za registraciju su:

- Ime
- Prezime
- Lozinka
- Adresa stanovanja
- Adresa e-pošte

Na temelju tih informacija aplikacija prema adresi stanovanja automatski određuje kojoj četvrti korisnik pripada i dodjeljuje mu se uloga stanovnika četvrti. Ostali korisnici kojima je dodijeljena viša uloga se pri otvaranju računa ne registriraju nego za njih račune stvaraju postojeći administratori sustava. Registracija je za sve korisnike, neovisno o ulozi, besplatna.

Da bi aplikacija zadovoljila sve funkcionalne zahtjeve koje četvrt zahtjeva, ona je sastavljena od tri osnovne cjeline. Prva cjelina je forum čija je svrha održavanje rasprava koje su tematski vezane uz aktualna događanja i sam život stanovnika u četvrti, sljedeća cjelina su događanja koja najavljuju buduće događaje u četvrti te posljednja, vijeće četvrti gdje se u kratkim natuknicama objavljuje sadržaj sjednica Vijeća gradske četvrti uz najavu novih tema.

Aplikacija razlikuje četiri vrste korisnika:

- Stanovnik četvrti
- Vijećnik

- Moderator četvrti
- Administrator aplikacije

Stanovniku četvrti je omogućeno vođenje rasprava unutar foruma i predlaganje najave događaja u četvrti. Pri definiranju prijedloga potrebno je definirati naziv, mjesto, vrijeme trajanja te kratak opis događaja.

Vijećnik uz mogućnosti redovnog stanovnika ima ovlasti pisanja objava u cjelini "Vijeće četvrti" gdje piše kratko izvješće sa zadnje sjednice.

Moderatori četvrti također imaju mogućnosti redovnih stanovnika, ali uz to pregledavaju najavljene događaje nakon čega ih odbijaju ili potvrđuju. Događaji se odbijaju ako sadrže vrijeđanje ili diskriminaciju na bilo kojoj razini. Potvrđene događaje objavljuju u rubrici "Događaji". Na raspolaganju im je i uređivanje najava zbog mogućih jezičnih pogrešaka. Osim toga, unutar foruma im je omogućeno brisanje objava neprikladnog sadržaja.

Administratori se ne smatraju stanovnicama niti jedne četvrti. Njihov je zadatak tehničke naravi te su oni zaduženi za upravljanje korisničkim računima, definiranje četvrti i njihovih granica.

Skup korisnika koji bi mogli biti zainteresirani za razvijeni sustav su stanovnici četvrti koji bi željeli uvesti neke promjene u svoju četvrt jer im se na ovaj način pruža lakši i učinkovitiji način diskusije, oni koji bi jednostavno htjeli poboljšati društvene odnose na razini kvarta i na kraju, oni koje zanima brži i lakši protok informacija vezanih uz život u kvartu.

### 3. Pojmovnik

**Apache Maven** - Programski alat koji služi za upravljanje paketima razvijen od strane Apache Software Foundationa često korišten u Java projektima.

**Apache Tomcat** - Programski alat otvorenog koda za internetske servere razvijen od strane Apache Software Foundation-a. Predstavlja okruženje prigodno za Java Servlet-e.

**Astah** - Alat za oblikovanje programske potpore koji omogućuje laku i brzu izradu raznih dijagrama.

**BCrypt** - BCrypt je funkcija za sažimanje lozinke koju su 1999. dizajnirali Niels Provos i David Mazières. Zasnovana je na Blowfish algoritmu. Funkcija se izvršava tako što se sažimanje izvršava više puta unutar petlje.

**CSS** (Cascading Style Sheets) - Stilski programski jezik koji opisuje HTML dokument određujući prikaz njegovih elemenata.

**DTO** (Data Transfer Object) - Programski objekt čija je namjena samo sadržavanje podataka te se koristi kao način prijenosa tih podataka među procesima.

**GitLab** - Aplikacija za organizaciju, planiranje, nadzor i upravljanje izradom programske potpore u obliku zajedničkog repozitorija.

**H2** - Sustav za upravljanje bazom podataka relacijskog tipa korišten ugradnjom u Java aplikacije.

**Hibernate ORM** - Alat za mapiranje objekt-relacija (**Object/Relational Mapping**) za programski jezik Javu razvijen od strane Red Hat-a. Predstavlja radni okvir za mapiranje objektno orijentiranog modela s bazom podataka.

**HTTPS** - Internetski protokol nastao kombinacijom protokola HTTP i protokola SSL/TLS. HTTPS omogućava kriptiranu komunikaciju i sigurnu identifikaciju web poslužitelja mreže. HTTPS veze se nerijetko koriste za novčane transakcije na World Wide Webu i ostale povjerljive transakcije u korporativnim informacijskim sustavima.

**IntelliJ IDEA** - Razvojno okruženje za izradu softvera u programskom jeziku Java.

**Java Servlet** - Java komponenta koja sadrži Java razred te proširuje mogućnosti servera. Predstavlja standard za implementaciju Java razreda zaduženih za odgovaranje na zahtjeve. Često je korištena s HTTP protokolom.

**JDK** (Java Development Kit) - Okruženje za razvijanje programske potpore koji sadrži potrebne alate za izradu Java aplikacija.

**JSON** (JavaScript Object Notation) - Često korišten format podataka lako čitljiv ljudima te lagan za obradu i generaciju računalima. Služi za razmjenu podatkovnih objekata.

**JSX** - Objektno orijentirani programski jezik u obliku sintaksne ekstenzije JavaScripta. Često se koristi za izradu korisničkih sučelja uparena s React-om.

**MVC obrazac (Model-View-Controller)** - Arhitekturni obrazac često korišten u izradi programske potpore za razvijanje korisničkih sučelja koji dijeli aplikaciju na tri komponente - podatke i poslovnu logiku, prikaz modeliranih podataka i upravljanje korisničkim zahtjevima.

**PostgreSQL** - Često korišten sustav otvorenog koda za upravljanje bazom podataka objektno-relacijskog tipa s naglaskom na standardima namijenjen za opću uporabu

**React** - JavaScript biblioteka koja služi za izradu korisničkih sučelja.

**REST** (Representational State Transfer) - Široko korišten stil arhitekture za opisivanje web-aplikacija. Definira ograničenja i način komunikacije poslužitelja i klijenta s naglaskom na uporabu jednostavnijih internetskih standarda kao što je HTTP - najrašireniji internetski protokol za prijenos informacija.

**Spring** - Radni okvir otvorenog koda za Java platformu razvijen od strane Pivotal Software-a za izradu aplikacija. Prvo izdanje je napisao Rod Johnson objavljeno 2002. g.

**Spring boot** - Modul radnog okvira Spring koji omogućuje brzi razvoj aplikacija (*Rapid Application Development*) koristeći princip *konvencije nad konfiguracijom*.

**Vim** - Uređivač teksta s prigodnim alatima za uređivanje izvornog koda.



## 4. Funkcionalni zahtjevi

### Dionici:

- Stanovnik četvrti
- Vijećnik
- Razvojni programer
- Naručitelj proizvoda (FER)

### Aktori i njihovi funkcionalni zahtjevi:

Aktori mogu biti inicijatori i sudionici. Inicijatori su stanovnici četvrti, vijećnici, moderatori četvrti i administratori aplikacije. Sudionik je baza podataka.

- **Stanovnik četvrti**
  - Može se registrirati
  - Može diskutirati o temama na forumu
  - Može predlagati najave budućih događanja u susjedstvu
- **Vijećnik**
  - Ima sve ovlasti kao i stanovnik četvrti (osim registracije)
  - Ima ovlast pisanja objava u cjelini „Vijeće četvrti“
- **Moderator četvrti**
  - Ima sve ovlasti kao i stanovnik četvrti (osim registracije)
  - Može pregledavati najave događanja koje su predložili stanovnici četvrti i objaviti ih u cjelini „Događanja“
  - Može uređivati prijedloge stanovnika četvrti u cjelini „Događanja“
  - Može odbaciti prijedloge neprikladnog sadržaja iz cjeline „Događanja“
  - Može ukloniti objave neprikladnog sadržaja s foruma
- **Administrator aplikacije**
  - Otvara račune za vijećnike, moderatore i druge administratore
  - Upravlja korisničkim računima
  - Definira četvrti i područja koja one obuhvaćaju
- **Baza podataka**
  - Pohranjuje podatke o svim korisnicima
  - Pohranjuje podatke o četvrtima i ulicama od kojih se one sastoje
  - Pohranjuje objave s foruma
  - Pohranjuje objave vijećnika gradske četvrti iz cjeline „Vijeće četvrti“
  - Pohranjuje najave iz cjeline „Događanja“

<b>UC1</b>	Registracija
<b>Glavni sudionik</b>	Stanovnik četvrti
<b>Rezultat</b>	Stvaranje novog korisničkog računa za stanovnika četvrti
<b>Ostali sudionici</b>	Baza podataka
<b>Željeni scenarij</b>	<ol style="list-style-type: none"> <li>1. Korisnik upisuje svoje ime, prezime, željeno korisničko ime, lozinku i adresu stanovanja (ulica u kojoj stanuje)</li> <li>2. Koristeći unesene podatke, sustav stvara novi korisnički račun</li> <li>3. Podaci o novostvorenom korisniku dodaju se u bazu podataka</li> </ol>
<b>Mogući scenarij</b>	<ol style="list-style-type: none"> <li>1. Za unesenu e-mail adresu već postoji korisnički račun</li> <li>2. Sustav odbacuje pokušaj registracije i javlja grešku korisniku</li> </ol>
<b>UC2</b>	Otvaranje povlaštenih korisničkih računa
<b>Glavni sudionik</b>	Administrator
<b>Rezultat</b>	Administrator stvara korisnički račun za vijećnika, moderatora ili drugog administratora; ovaj postupak ima isti rezultat kao i registracija kod stanovnika četvrti
<b>Ostali sudionici</b>	Vijećnik, moderator, baza podataka
<b>Preduvjeti</b>	1. Administrator je prijavljen u sustav
<b>Željeni scenarij</b>	<ol style="list-style-type: none"> <li>1. Administrator popunjava obrazac za dodavanje novog korisničkog računa</li> <li>2. Pritiskom na gumb „Stvori korisnika” novi korisnik se stvara u bazi podataka s ulogom koju mu je dodijelio administrator</li> </ol>
<b>Mogući scenarij</b>	<ol style="list-style-type: none"> <li>1. Za unesenu e-mail adresu već postoji korisnički račun</li> <li>2. Sustav odbacuje pokušaj stvaranja novog računa i javlja grešku administratoru</li> </ol>
<b>UC3</b>	Prijava
<b>Glavni sudionik</b>	Stanovnik četvrti, vijećnik, moderator, administrator
<b>Rezultat</b>	Prijava u sustav čime se omogućuje korisniku rad u aplikaciji
<b>Ostali sudionici</b>	Baza podataka
<b>Željeni scenarij</b>	<ol style="list-style-type: none"> <li>1. Korisnik upisuje svoje korisničko ime i lozinku</li> <li>2. Sustav prepoznaje važeću prijavu i daje korisniku odgovarajuće ovlasti</li> </ol>
<b>Mogući scenarij</b>	<ol style="list-style-type: none"> <li>1. Korisnik je upisao nepostojeće korisničko ime ili neispravnu lozinku</li> <li>2. Sustav odbacuje pokušaj prijave</li> <li>3. Sustav o tome obavještava korisnika</li> </ol>
<b>UC4</b>	Odjava
<b>Glavni sudionik</b>	Stanovnik četvrti, vijećnik, moderator, administrator
<b>Rezultat</b>	Korisnik se odjavljuje iz sustava i vraća na početnu stranicu foruma
<b>Preduvjeti</b>	1. Korisnik je prijavljen u sustav
<b>Željeni scenarij</b>	<ol style="list-style-type: none"> <li>1. Korisnik pritišće gumb „Odjavi se”</li> <li>2. Sustav završava korisničku sjednicu</li> </ol>
<b>UC5</b>	Objavljivanje na forumu
<b>Glavni sudionik</b>	Stanovnik četvrti, vijećnik, moderator
<b>Rezultat</b>	Stvaranje nove objave
<b>Ostali sudionici</b>	Baza podataka

<b>Preduvjeti</b>	1. Korisnik je prijavljen u sustav
<b>Željeni scenarij</b>	1. Korisnik upisuje tekst objave i potvrđuje unos 2. Sustav prihvaća objavu i prikazuje ju u odabranoj temi na forumu
<b>UC6</b>	Brisanje vlastite objave s foruma
<b>Glavni sudionik</b>	Stanovnik četvrti, vijećnik, moderator
<b>Rezultat</b>	Objava je obrisana iz baze podataka
<b>Ostali sudionici</b>	Baza podataka
<b>Preduvjeti</b>	1. Korisnik je prijavljen u sustav
<b>Željeni scenarij</b>	1. Korisnik potvrđuje svoj odabir 2. Sustav briše objavu iz baze podataka
<b>UC7</b>	Uređivanje vlastitih objava s foruma
<b>Glavni sudionik</b>	Stanovnik četvrti, vijećnik, moderator
<b>Rezultat</b>	Promjena sadržaja objave i pohranjivanje novog sadržaja u bazu podataka
<b>Ostali sudionici</b>	Baza podataka
<b>Preduvjeti</b>	1. Korisnik je prijavljen u sustav
<b>Željeni scenarij</b>	1. Korisnik upisuje novi sadržaj i potvrđuje ga 2. U bazi podataka se izmjenjuje sadržaj originalne objave 3. Sustav prikazuje novu verziju ostalim korisnicima
<b>UC8</b>	Pregled objava na forumu
<b>Glavni sudionik</b>	Stanovnik četvrti, vijećnik, moderator
<b>Rezultat</b>	Objave se prikazuju korisniku
<b>Ostali sudionici</b>	Baza podataka
<b>Preduvjeti</b>	1. Korisnik je prijavljen u sustav
<b>Željeni scenarij</b>	1. Korisnik pritišće gumb „Forum“ 2. Korisnik bira temu koju želi pregledati 3. Objave u određenoj temi se prikazuju korisniku klikom na ime teme
<b>UC9</b>	Stvaranje teme na forumu
<b>Glavni sudionik</b>	Stanovnik četvrti, vijećnik, moderator
<b>Rezultat</b>	Tema je otvorena na forumu i moguće je pisati objave u njoj
<b>Ostali sudionici</b>	Baza podataka
<b>Preduvjeti</b>	1. Korisnik je prijavljen
<b>Željeni scenarij</b>	1. Korisnik pritišće gumb „Forum“ 2. Korisnik stvara novu temu pritiskom na gumb „Stvori temu“ i unosi ime nove teme
<b>UC10</b>	Brisanje teme s foruma
<b>Glavni sudionik</b>	Moderator
<b>Rezultat</b>	Tema je obrisana iz baze podataka zajedno sa svim objavama koje su bile u njoj
<b>Ostali sudionici</b>	Baza podataka
<b>Preduvjeti</b>	1. Moderator je prijavljen
<b>Željeni scenarij</b>	1. Moderator pritišće gumb „Forum“ 2. Moderator odabire temu i pritišće gumb „Obriši temu“

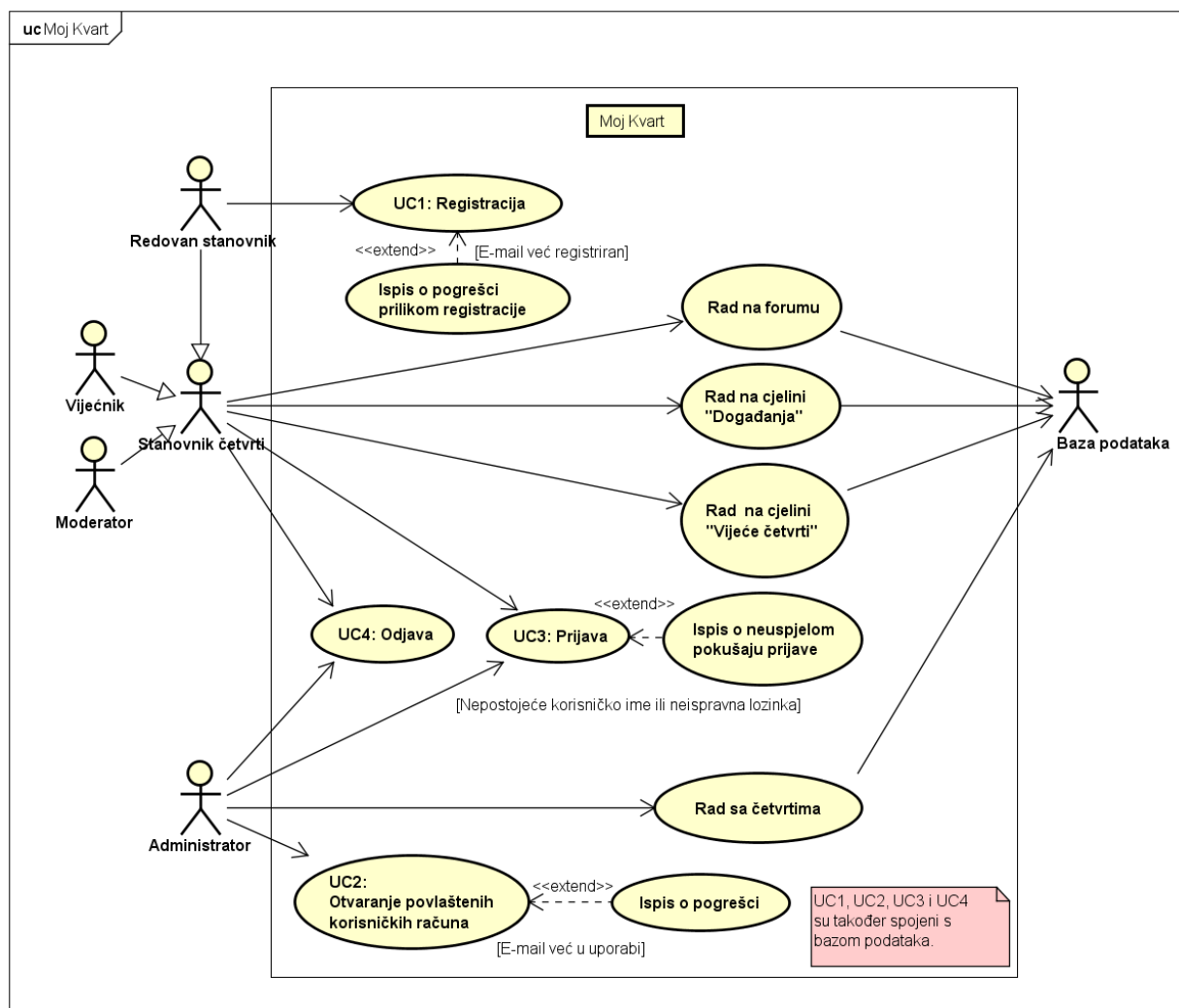
	3. Tema se uklanja iz baze podataka i iz cjeline „Forum”
<b>UC11</b>	Uklanjanje objava s foruma
<b>Glavni sudionik</b>	Moderator
<b>Rezultat</b>	Objava je obrisana iz baze podataka
<b>Ostali sudionici</b>	Baza podataka
<b>Preduvjeti</b>	1. Moderator je prijavljen
<b>Željeni scenarij</b>	1. Moderator pritišće gumb „Obriši” 2. Objava se uklanja iz baze podataka
<b>UC12</b>	Prijedlog najave događaja
<b>Glavni sudionik</b>	Stanovnik četvrti, vijećnik, moderator
<b>Rezultat</b>	Najava događaja poslana je na razmatranje moderatoru koji je zadužen za dotičnu četvrt
<b>Ostali sudionici</b>	Baza podataka
<b>Preduvjeti</b>	1. Korisnik je prijavljen u sustav
<b>Željeni scenarij</b>	1. Korisnik potvrđuje unos 2. Prijedlog se pohranjuje u bazu podataka 3. Uneseni prijedlog se prikazuje moderatoru
<b>UC13</b>	Objavljivanje u cjelini „Događanja”
<b>Glavni sudionik</b>	Moderator
<b>Rezultat</b>	Moderator dobiva prijedlog objave od stanovnika četvrti i odlučuje hoće li ga odobriti ili odbaciti
<b>Ostali sudionici</b>	Baza podataka
<b>Preduvjeti</b>	1. Moderator je prijavljen
<b>Mogući scenarij</b>	a) 1. Moderator objavljuje najavu pritiskom na gumb „Objavi” 2. Odobrena najava se pohranjuje u bazu podataka 3. Odobrena najava postaje javna i vidljiva svim korisnicima aplikacije b) 1. Moderator pregledava prijedlog pritiskom na gumb „Uredi” 2. Moderator unosi potrebne izmjene 3. Moderator pohranjuje prijedlog u bazu podataka pritiskom na gumb „Pohrani prijedlog” ili ga objavljuje odmah nakon uređivanja pritiskom na gumb „Objavi” c) 1. Moderator pritišće gumb „Obriši” 2. Prijedlog se uklanja iz baze podataka
<b>UC14</b>	Pregled odobrenih objava iz cjeline „Događanja”
<b>Glavni sudionik</b>	Stanovnik četvrti, vijećnik, moderator
<b>Rezultat</b>	Odobrene objave se prikazuju korisniku
<b>Ostali sudionici</b>	Baza podataka
<b>Preduvjeti</b>	1. Korisnik je prijavljen u sustav
<b>Željeni scenarij</b>	1. Korisnik pritišće gumb „Događanja” 2. Objave se dohvaćaju iz baze podataka i prikazuju korisniku
<b>UC15</b>	Objavljivanje u cjelini „Vijeće četvrti”

<b>Glavni sudionik</b>	Vijećnik
<b>Rezultat</b>	Objave (kratka izvješća i najave budućih tema) koje pišu vijećnici se stvaraju i moguće ih je pregledavati
<b>Ostali sudionici</b>	Baza podataka
<b>Preduvjeti</b>	1. Vijećnik je prijavljen u sustav
<b>Željeni scenarij</b>	1. Vijećnik upisuje tekst objave i potvrđuje unos 2. Sustav prihvaća objavu i prikazuje ju u cjelini „Vijeće četvrti“
<b>UC16</b>	Pregled objava vijećnika iz cjeline „Vijeće četvrti“
<b>Glavni sudionik</b>	Stanovnik četvrti, vijećnik, moderator
<b>Rezultat</b>	Kratka izvješća koja pišu vijećnici se prikazuju korisniku
<b>Ostali sudionici</b>	Baza podataka
<b>Preduvjeti</b>	1. Korisnik je prijavljen u sustav
<b>Željeni scenarij</b>	1. Korisnik pritišće gumb „Vijeće četvrti“ 2. Objave iz cjeline „Vijeće četvrti“ se dohvaćaju iz baze podataka i prikazuju korisniku
<b>UC17</b>	Dodavanje četvrti
<b>Glavni sudionik</b>	Administrator
<b>Rezultat</b>	Nova četvrt je definirana ovom akcijom
<b>Ostali sudionici</b>	Moderator, baza podataka
<b>Preduvjeti</b>	1. Administrator je prijavljen
<b>Željeni scenarij</b>	1. Administrator definira novu četvrt i unosi ulice koje se nalaze u novodefiniranoj četvrti 2. Administrator odabire moderatora za četvrt 3. Podaci se pohranjuju u bazu podataka
<b>Mogući scenarij</b>	1. Četvrt koja se dodaje postoji u bazi podataka 2. Sustav odbacuje zahtjev za stvaranje nove četvrti
<b>UC18</b>	Uređivanje četvrti
<b>Glavni sudionik</b>	Administrator
<b>Rezultat</b>	Administrator dodaje neku novu ulicu u sastavni dio određene četvrti
<b>Ostali sudionici</b>	Baza podataka
<b>Preduvjeti</b>	1. Administrator je prijavljen
<b>Željeni scenarij</b>	1. Administrator nalazi četvrt koju treba urediti preko tražilice koristeći ime četvrti 2. Administrator pritiskom na gumb „Dodaj ulicu“ dobiva pristup tekstualnom polju za upis ulice 3. Pri unosu imena ulice i pritiskom na gumb „Ok“ ulica se dodaje u sastav četvrti, a time i u bazu podataka
<b>UC19</b>	Brisanje četvrti
<b>Glavni sudionik</b>	Administrator
<b>Rezultat</b>	Četvrt je obrisana iz baze podataka
<b>Ostali sudionici</b>	Baza podataka
<b>Preduvjeti</b>	1. Administrator je prijavljen 2. Četvrt postoji u bazi podataka
<b>Željeni scenarij</b>	1. Administrator nalazi četvrt koju treba obrisati preko tražilice koristeći ime četvrti

2. Administrator pritiskom na gumb „Obriši četvrt” nepovratno briše četvrt iz baze podataka

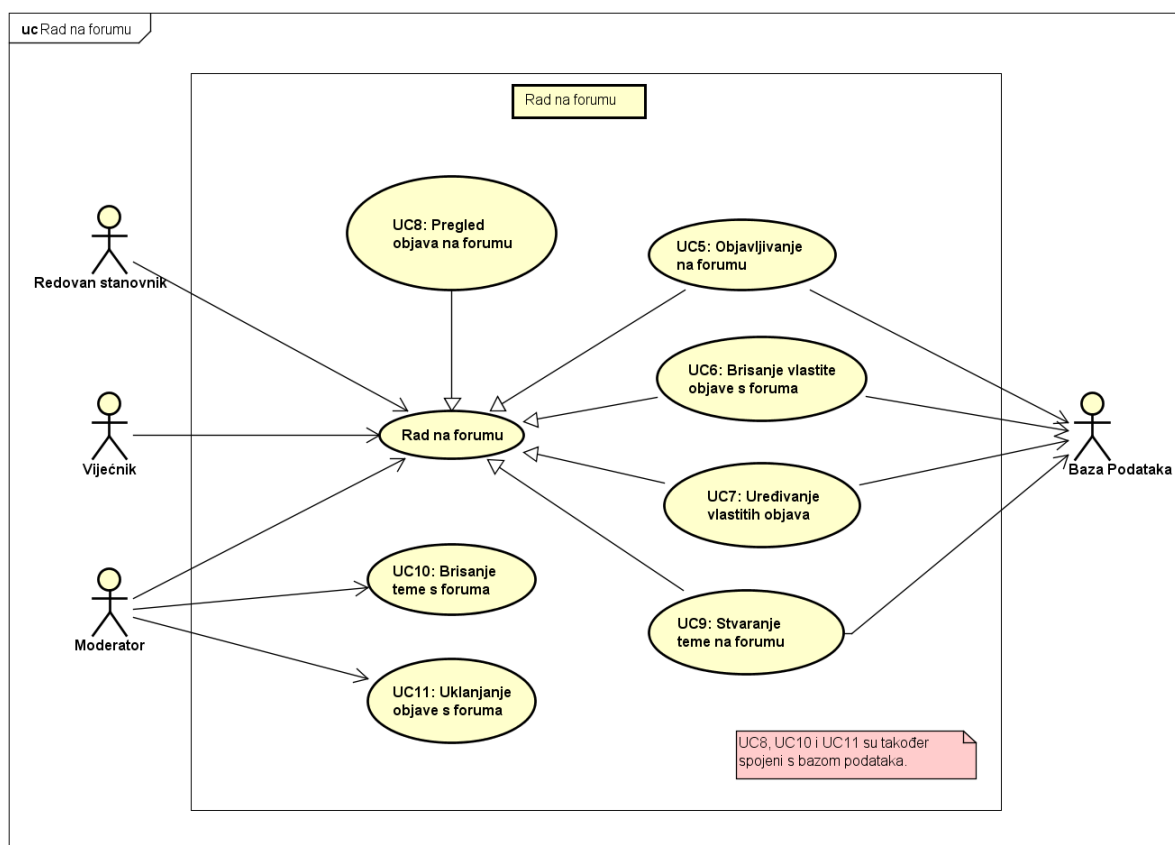
## 4.1. Dijagrami obrazaca uporabe

Vršni dijagram obrazaca uporabe:



Slika 1: Vršni dijagram obrazaca uporabe

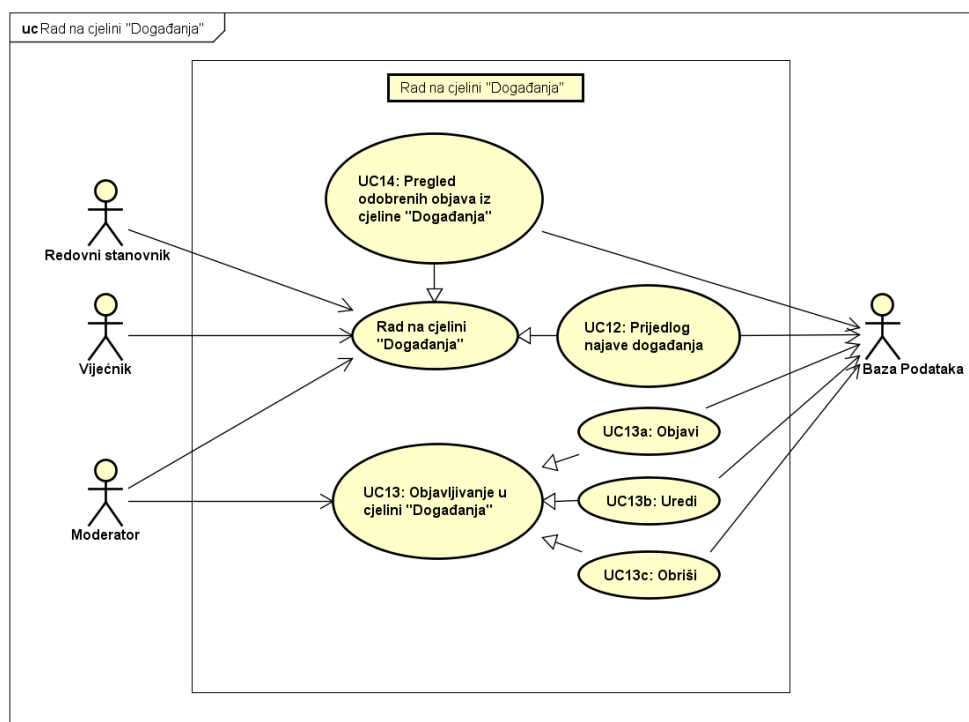
## Rad na forumu:



Slika 2: Dijagram obrazaca uporabe - Rad na forumu

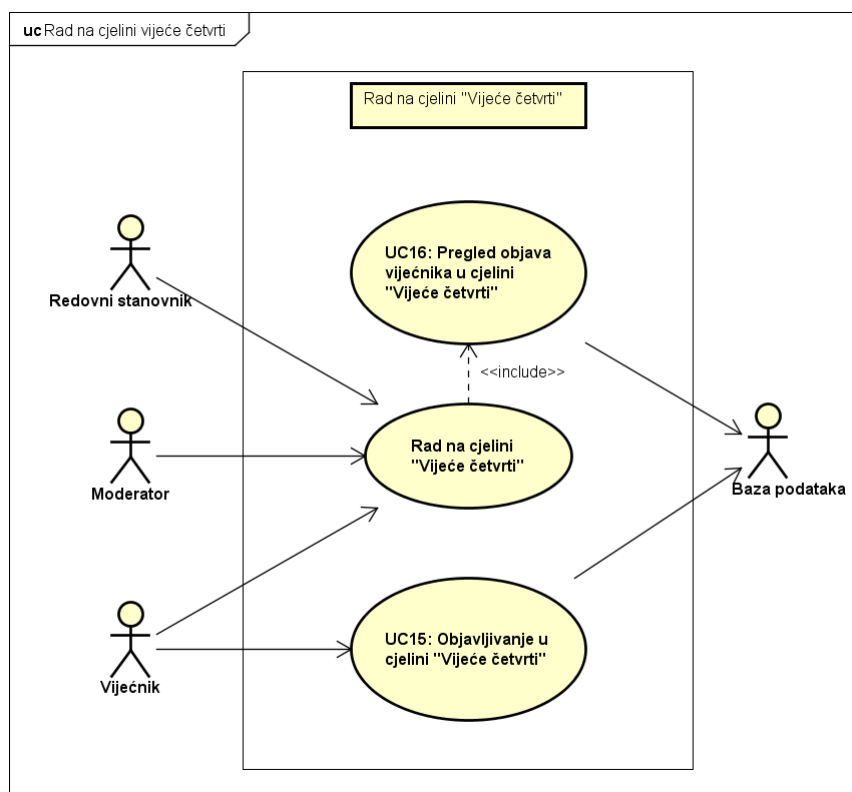


## Cjelina "Događanja":



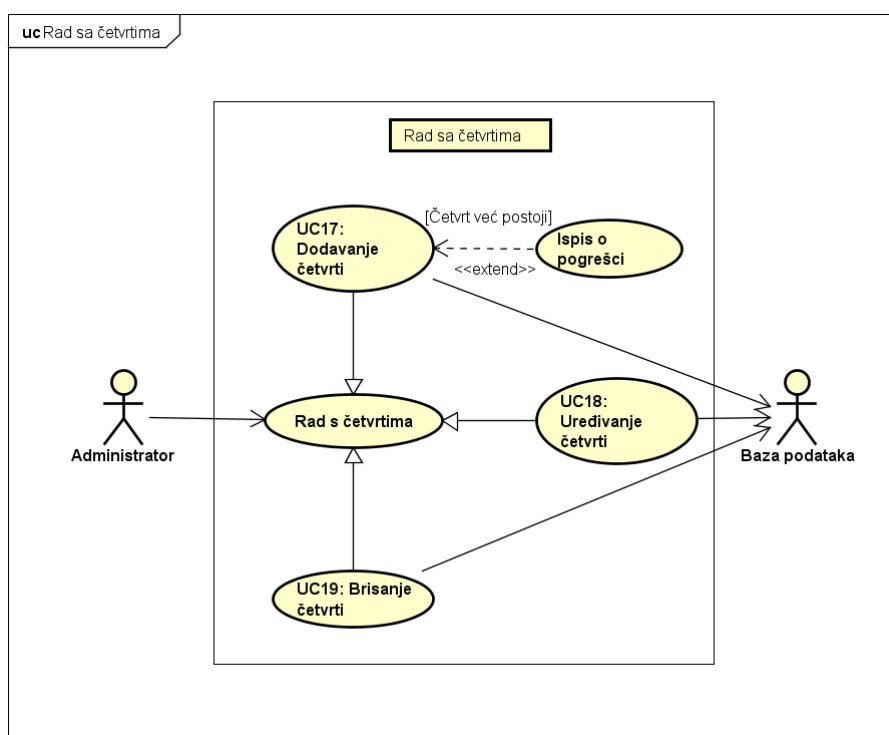
Slika 3: Dijagram obrazaca uporabe - cjelina "Događanja"

## Cjelina "Vijeće četvrti":



Slika 4: Dijagram obrazaca uporabe - cjelina "Vijeće četvrti"

## Rad sa četvrtima:

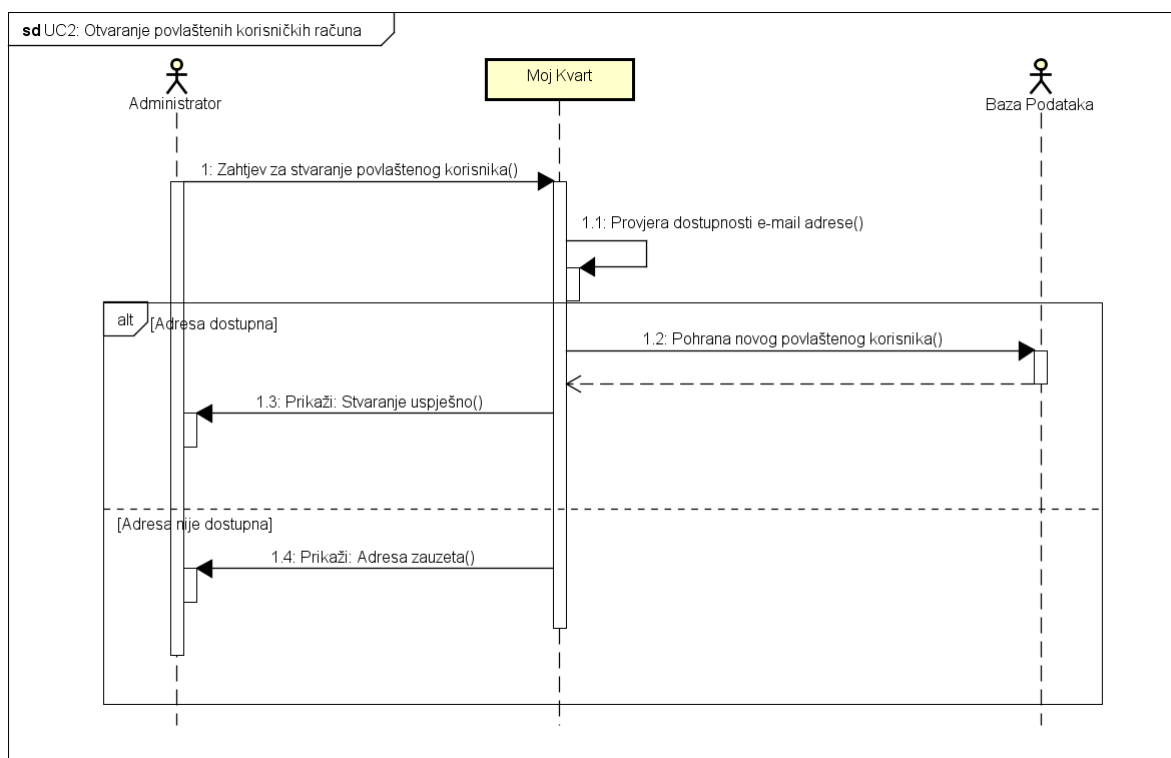


Slika 5: Dijagram obrazaca uporabe - Rad sa četvrtima

## 4.2. Sekvencijski dijagrami

### UC2: Otvaranje povlaštenih korisničkih računa

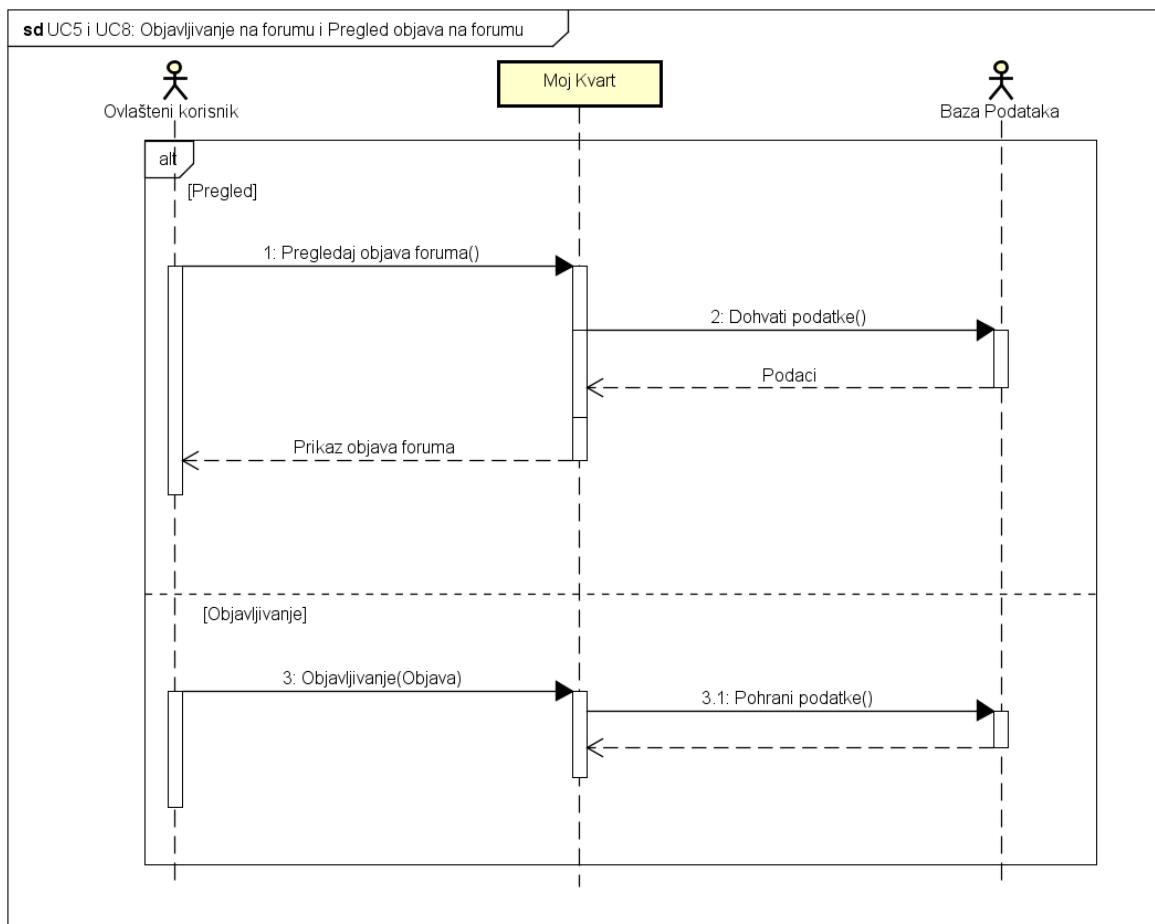
Povlašteni korisnički račun može otvoriti samo administrator. Administrator šalje zahtjev za stvaranje povlaštenog korisničkog računa s odgovarajućim podacima aplikaciji "Moj Kvar" koja zatim provjerava dostupnost e-mail adrese. U slučaju da e-mail adresa nije dostupna, aplikacija administratoru prikazuje odgovarajuću poruku i time je završena radnja. Ukoliko je e-mail adresa dostupna, aplikacija šalje bazi podataka zahtjev za pohranom te dobiva njen odgovor o uspješnosti pohrane. Pohranom u bazu podataka je račun otvoren te se administratoru ispisuje poruka o uspjehu otvaranja računa.



Slika 6: UC2: Otvaranje povlaštenih korisničkih računa

## UC5 i UC8: Objavljivanje na forumu i Pregled objava na forumu

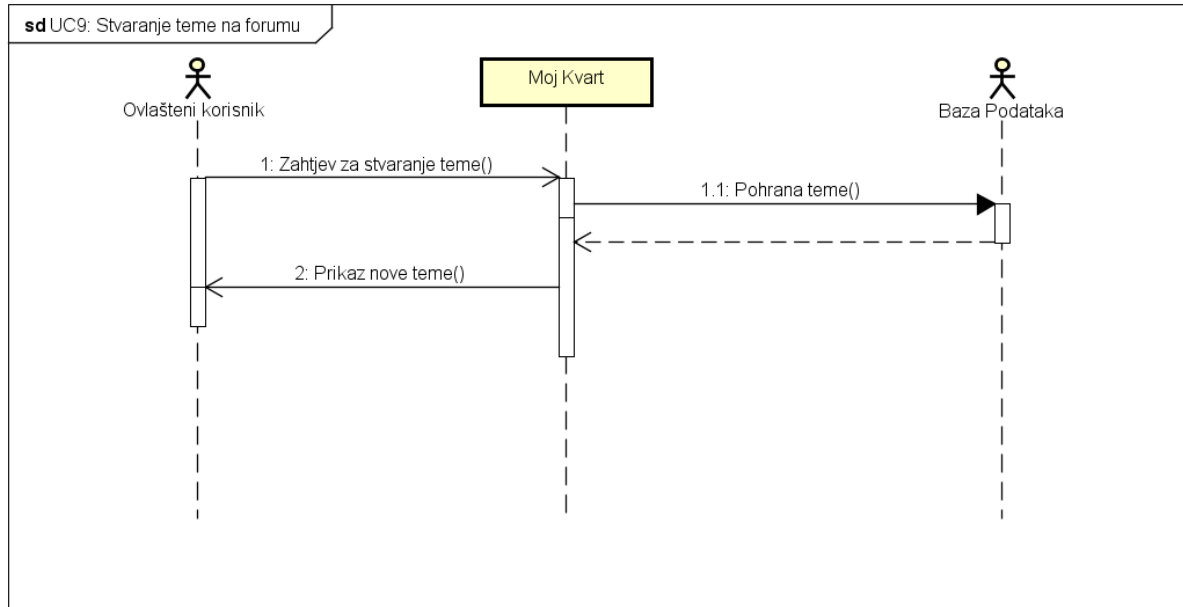
Korisnik pregledava forum posredno putem aplikacije koja dohvaća podatke iz baze podataka - odabirom teme šalje zahtjev aplikaciji koja šalje zahtjev za dohvat podataka bazi podataka. Na sličan način i objavljuje - unosi tekst objave kojeg zahtjevom šalje aplikaciji te ona zatim šalje zahtjev za pohranom podatka bazi podataka.



Slika 7: UC5 i UC8: Objavljivanje na forumu i Pregled objava na forumu

## UC9: Stvaranje teme na forumu

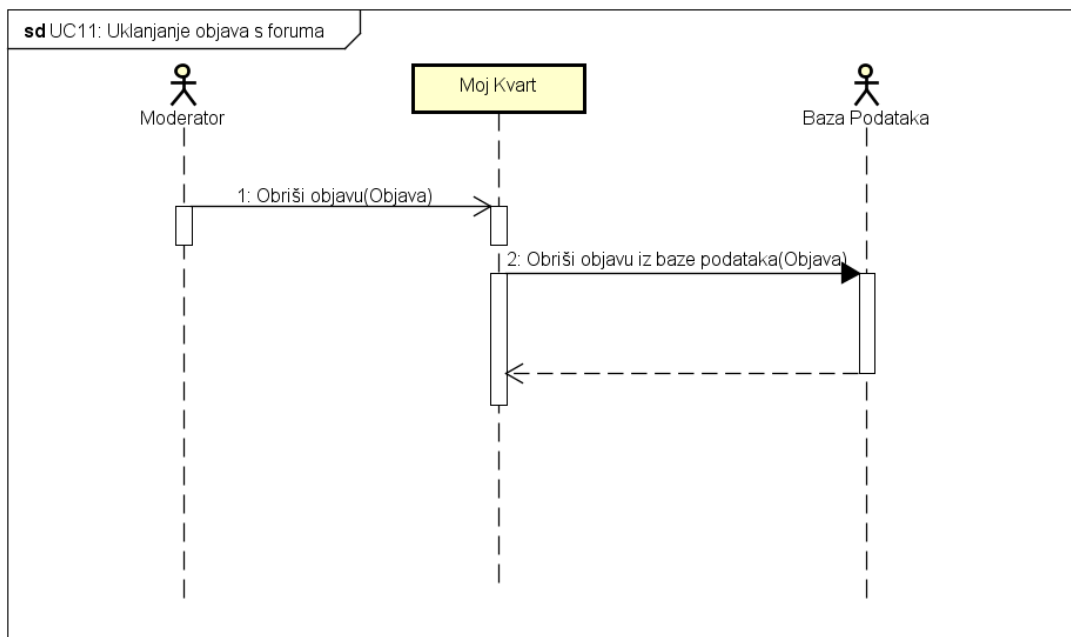
Korisnik odabire stvaranje nove teme te unosi ime nove teme - taj zahtjev se šalje aplikaciji koja pohranjuje novu temu u bazu podataka. Aplikacija nakon uspješnog stvaranja nove teme korisniku istu i prikazuje.



Slika 8: UC9: Stvaranje teme na forumu

## UC11: Uklanjanje objava s foruma

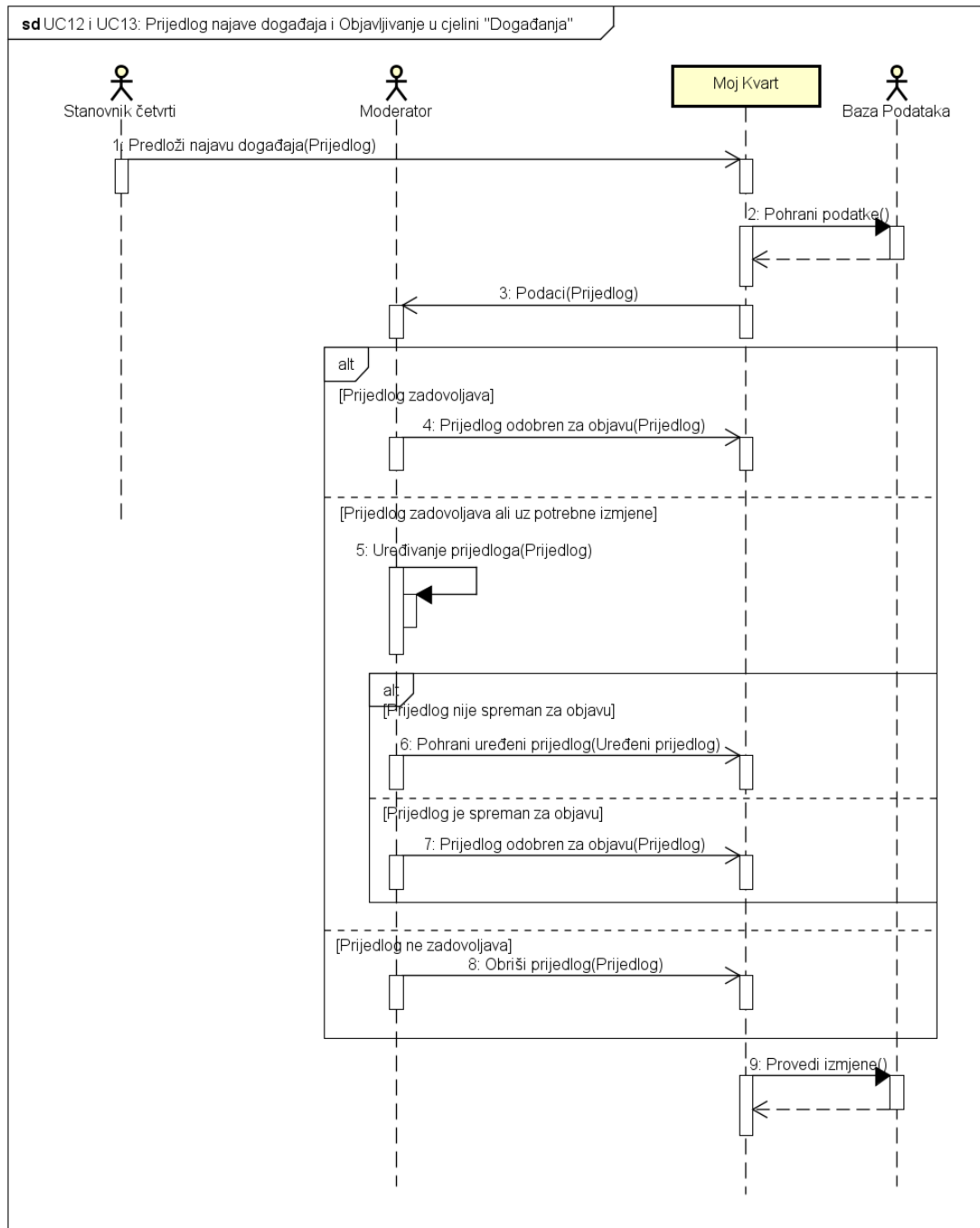
Moderator može obrisati objavu s foruma odabirom brisanja čime se šalje zahtjev aplikaciji koja zatim šalje zahtjev bazi podataka za uklanjanje odgovarajućih podataka.



Slika 9: UC11: Uklanjanje objava s foruma

## UC12 i UC13: Prijedlog najave događanja i Objavljivanje u cjelini "Događanja"

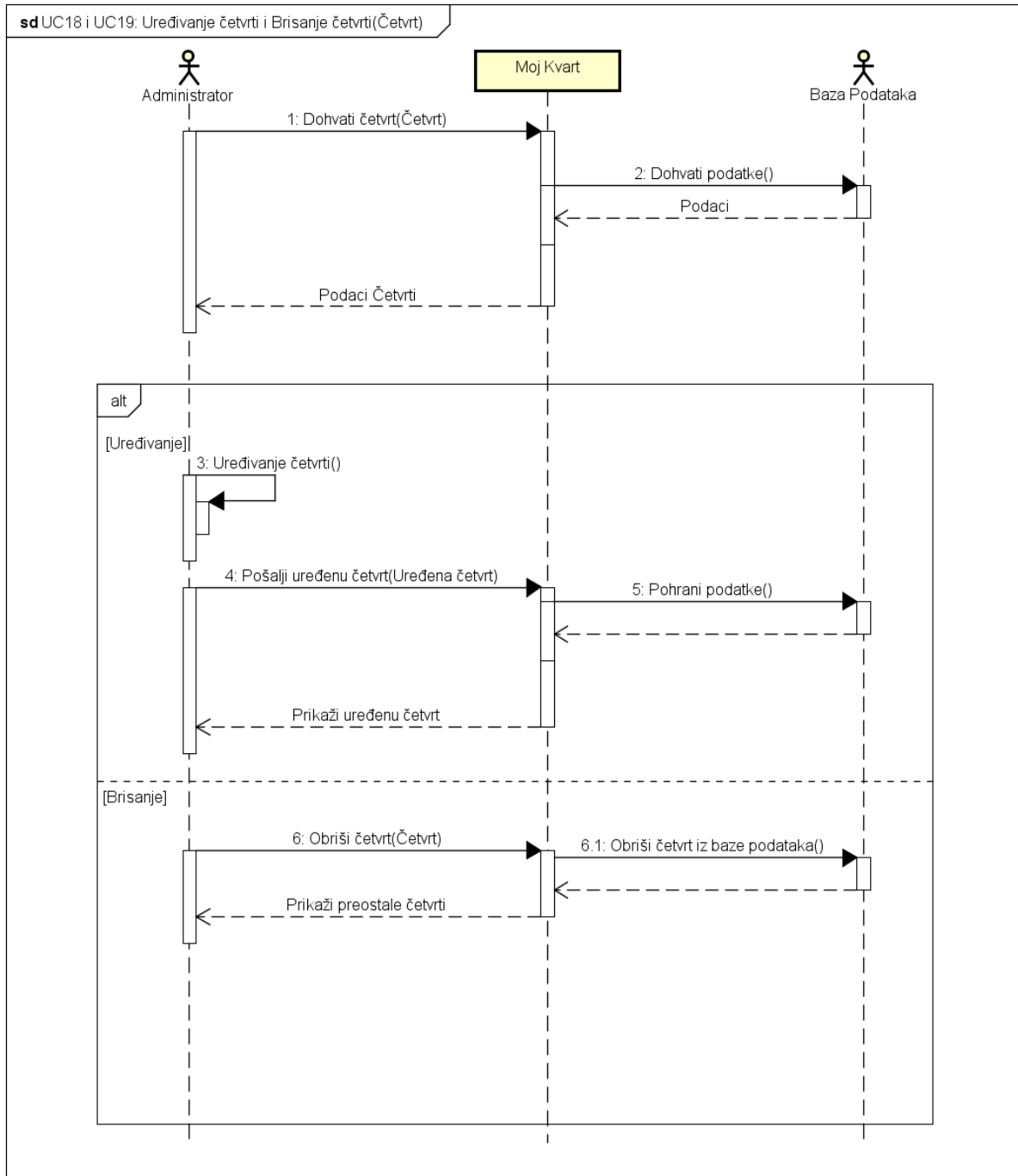
Stanovnik četvrti šalje aplikaciji zahtjev za predlaganjem najave događanja s odgovarajućim podacima. Aplikacija pohranjuje primljene podatke u bazu podataka te ispisuje prijedlog moderatoru koji odlučuje želi li prijedlog odobriti, dodatno urediti ili odbiti. U slučaju dodatnog uređivanja, uređeni prijedlog može dobiti ili pohraniti za daljnje uređivanje te se aplikaciji time šalje odgovarajući zahtjev kao i u slučaju inicijalnog prihvaćanja ili odbijanja. Aplikacija na temelju odgovora moderatora obavlja odgovarajuće izmjene u bazi podataka.



Slika 10: UC12 i UC13: Prijedlog najave događaja i Objavljivanje u cjelini "Događanja"

## UC18 i UC19: Uređivanje četvrti i brisanje četvrti

Administrator aplikaciji šalje zahtjev za dohvaćanjem željene četvrti. Aplikacija dohvaća željenu četvrt te prikazuje njene podatke administratoru. Administrator može urediti ili u potpunosti obrisati dohvaćenu četvrt. Dohvaćenoj četvrti može vršiti izmjene tako da dodaje ulice koje joj pripadaju nakon čega šalje zahtjev aplikaciji koja pohranjuje promjene u bazu podataka. Odabirom brisanja se šalje zahtjev aplikaciji koja zatim uklanja podatke te četvrti iz baze podataka.



Slika 11: UC18 i UC19: Uređivanje četvrti i Brisanje četvrti

## 5. Ostali zahtjevi

- Korisničko sučelje mora biti oblikovano u skladu sa standardnim hrvatskim jezikom
- Sustav mora podržavati hrvatske diakritičke znakove
- Sustav mora omogućiti paralelni rad više korisnika
- Vrijeme odziva u normalnim uvjetima ne smije biti veće od 10 sekundi
- Korisniku je ograničen broj znakova koje može unijeti u interakciji s pojedinim segmentima aplikacije
- Eventualne korisničke greške ne smiju utjecati na rad sustava i na stanje baze podataka
- Korisnik mora biti obaviješten o eventualnom neispravnom korištenju sustava
- Podaci se od poslužitelja do klijenta moraju prenositi protokolom HTTPS
- Korisničke lozinke ne smiju biti spremljene u bazu podataka kao jednostavan tekst, već se za njihovo šifriranje mora koristiti BCrypt funkcija za računanje sažetka
- Korisnik/gost ne smije imati uvid u dijelove sustava za koje nema ovlasti
- Web-aplikacija mora biti napisana koristeći se objektno orijentiranim jezikom
- Web-aplikacija i pripadna dokumentacija moraju biti u cijelosti dovršene do siječnja 2019. godine



## 6. Arhitektura i dizajn sustava

### 6.1. Svrha, opći prioriteti i skica sustava

Razvoj programskog produkta se sastoji od četiri generičke aktivnosti: specifikacija, razvoj i oblikovanje, validacija te evolucija. Fazi razvoja i oblikovanja kao prva, odnosno, početna aktivnost pripada izbor i oblikovanje arhitekture. Arhitektura sustava podrazumijeva osnovna pravila stvaranja i strukturiranja cijelog sustava te je ona, kada je jednom osmišljena i provedena, nešto što se ne može tek tako mijenjati pa je zbog toga prije same implementacije potrebno arhitekturi pridati mnogo pažnje i vremena.

Pravilan odabir arhitekture ima niz prednosti, od kojih su samo neke niža cijena razvoja i održavanja programskog produkta, ponovna uporaba, veća razumljivost te uočavanje i analiza ranih pogrešaka u oblikovanju. Iz navedenih razloga je vidljivo da je odabir arhitekture ključan za razvoj programskog produkta pa je u svrhu izrade web aplikacije čija će implementacija biti opisana u kasnijim poglavljima odabrana objektno orijentirana arhitektura.

Osnovni elementi koje svaka objektno orijentirana arhitektura uključuje su:

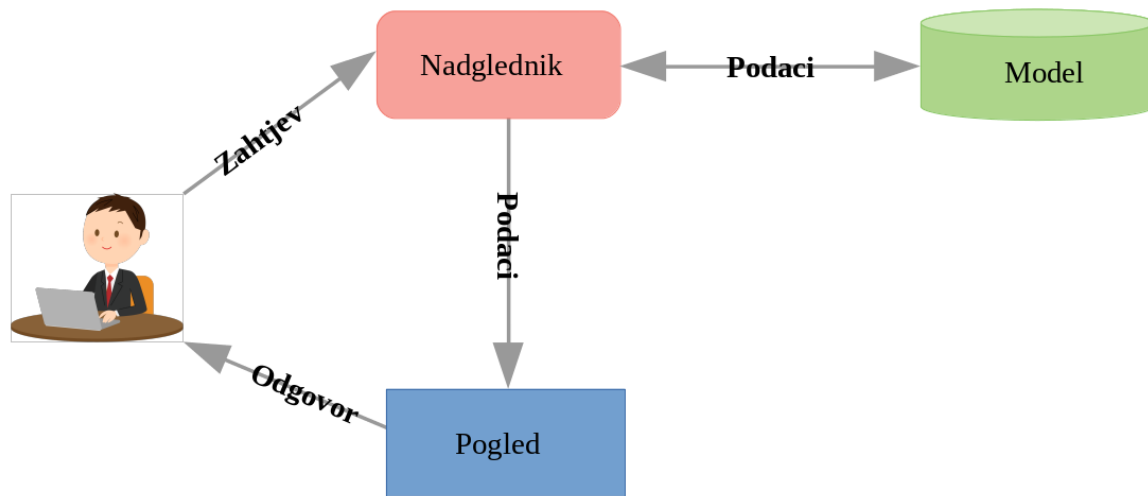
- **Objekt** (*engl. Object*) – osnovna jedinica, primjerak razreda. U objektno orijentiranoj paradigmi teži se uporabi članskih atributa. Isto tako metode pripadaju nekom konkretnom razredu.
- **Učahurivanje** (*engl. Encapsulation*) – ili enkapsulacija. Pošto ne postoje globalne varijable, pristup varijablama razreda je moguć jedino preko neke metode iz te klase. Time je onemogućeno nepredviđeno mijenjanja parametara.
- **Nasljeđivanje** (*engl. Inheritance*) – razredi mogu naslijediti svojstva svojih roditelja te ih nadopuniti svojstvima specifičnim samo za taj razred.
- **Višeobličje** (*engl. Polymorphism*) – mogućnost definiranja metoda istog imena, a različitih parametara.

Objektno orijentirana paradigma općenito je način modeliranja u kojem se svijet promatra kroz objekte. Takav način oblikovanja je već ustaljen u svijetu programiranja, modularan, razumljiv i kod je oblikovan u smislene cjeline koje nazivamo razredima.

Ovaj projekt će prilikom oblikovanja pratiti MVC ("Model-View-Controller", odnosno model – pogled – nadglednik) obrazac, a sam sustav se bazira na arhitekturi klijent – poslužitelj.

Razvoj web aplikacije bit će podijeljen u tri mehanizma:

1. **Model** – objektni model podataka. Služi skladištenju struktura podataka korisničkog sučelja.
2. **View** – prikaz grafičkog korisničkog sučelja.
3. **Controller** – logika interakcije / promjene.



Slika 12: MVC obrazac

Arhitektura sustava ovog projekta sastoji se od:

- Klijentska aplikacija
- Administratorska aplikacija
- Središnji poslužitelj
- Baza podataka

### Klijentska aplikacija

Klijentska aplikacija je namijenjena krajnjem korisniku (u vidu ovog projekta to je stanovnik četvrti). Ona nudi grafičko korisničko sučelje koje je razumljivo i intuitivno za korištenje.

### Administratorska aplikacija

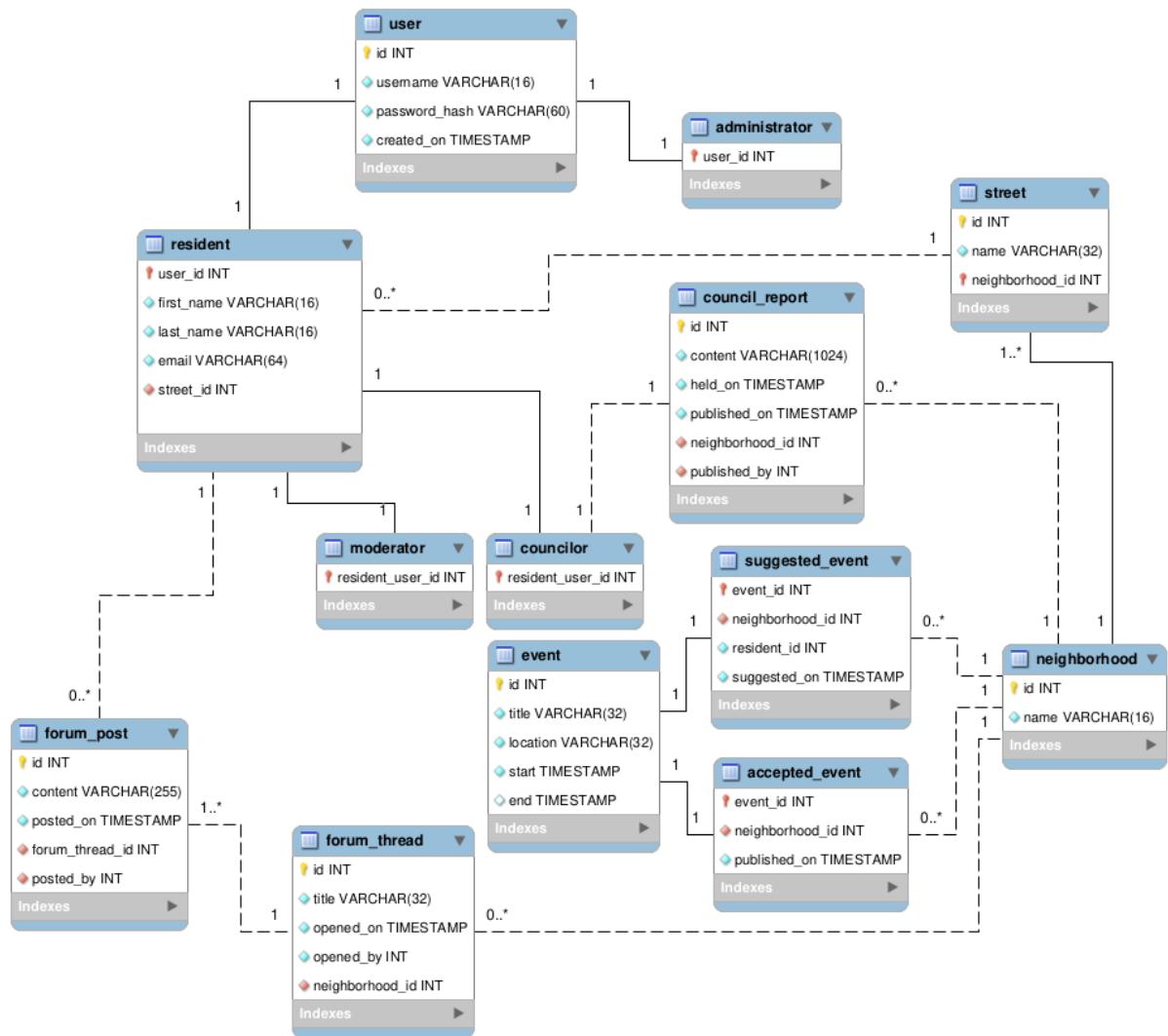
Administrator sustava upravlja računima korisnika, otvara povlaštene račune, definira četvrti i područja koja ona obuhvaćaju. Administrator je neovisan od stanovnika četvrti i ne preuzima njegove uloge te on navedenim ovlastima pristupa preko administratorskog dijela aplikacije koji je neovisan o klijentskoj aplikaciji.

### Središnji poslužitelj

Središnji poslužitelj vrši obradu zahtjeva klijentske i administratorske aplikacije. Svaki zadatak koji uključuje slanje i primanje podataka komunicira sa središnjim poslužiteljem čiji je zadatak obrada i slanje povratnih informacija klijentu. Središnji poslužitelj pri obavljanju rada komunicira s bazom podataka.

## Baza podataka

Baza podataka je važan dio sustava jer su u njoj pohranjeni svi podaci koje koristi web aplikacija. Korištena baza podataka je relacijskog tipa. Takav model organizira podatke u jednu ili više tablica (relacija) između kojih su definirane veze.



Slika 13: Skica baze podataka

Baza ovog sustava sadrži tablice s navedenim atributima:

- **Korisnik** (*user*)
  - šifra korisnika
  - korisničko ime
  - sažetak lozinke
  - datum kreiranja računa
- **Administrator** (*administrator*)
- **Stanovnik** (*resident*)
  - ime stanovnika
  - prezime stanovnika
  - email adresa stanovnika
  - šifra ulice u kojoj stanovnik živi
- **Moderator** (*moderator*)
- **Vijećnik** (*councilor*)
- **Objava** (*forum\_post*)
  - šifra objave
  - sadržaj objave
  - vrijeme objave
  - šifra autora objave
  - šifra teme kojoj objava pripada
- **Tema** (*forum\_thread*)
  - šifra teme
  - naslov teme
  - vrijeme otvaranja teme
  - autor teme
  - šifra četvrti za koju je vezana tema
- **Četvrt** (*neighborhood*)
  - šifra četvrti
  - ime četvrti
- **Događaj** (*event*)
  - šifra događaja
  - ime događaja
  - lokacija događaja
  - vrijeme početka događaja
  - vrijeme kraja događaja
    - **Predložen događaj** (*suggested\_event*)
      - šifra autora prijedloga događaja
      - vrijeme predlaganja događaja
    - **Prihvaćen događaj** (*accepted\_event*)
      - vrijeme prihvaćanja događaja
- **Ulica** (*street*)
  - šifra ulice
  - ime ulice
  - šifra četvrti u kojoj je ulica

## 6.2. Dijagram razreda s opisom

Razred **User** predstavlja registriranog korisnika. Sadrži korisničko ime, sažetak lozinke i vrijeme otvaranja računa.

**Administrator** je registrirani korisnik s posebnim ovlastima.

**Resident** je uobičajeni registrirani korisnik koji stanuje u nekoj četvrti. Sadrži puno ime, e-mail adresu i ulicu u kojoj stanuje.

**Resident** može dodatno biti **Councilor** (vijećnik četvrti) ili **Moderator**.

Razred **Street** predstavlja imenovanu ulicu, a **Neighborhood** skup ulica, odnosno imenovanu četvrt.

**Neighborhood** sadrži kolekcije instanci razreda **ForumThread** (tema na forumu), **AcceptedEvent** (prihvaćeni događaj), **SuggestedEvent** (predloženi događaj) i **CouncilReport** (izvješće vijeća četvrti).

**ForumThread** sadrži naslov, trenutak otvaranja i stanovnika (**Resident**) koji ju je otvorio te se sastoji od više postova na forumu (**ForumPost**).

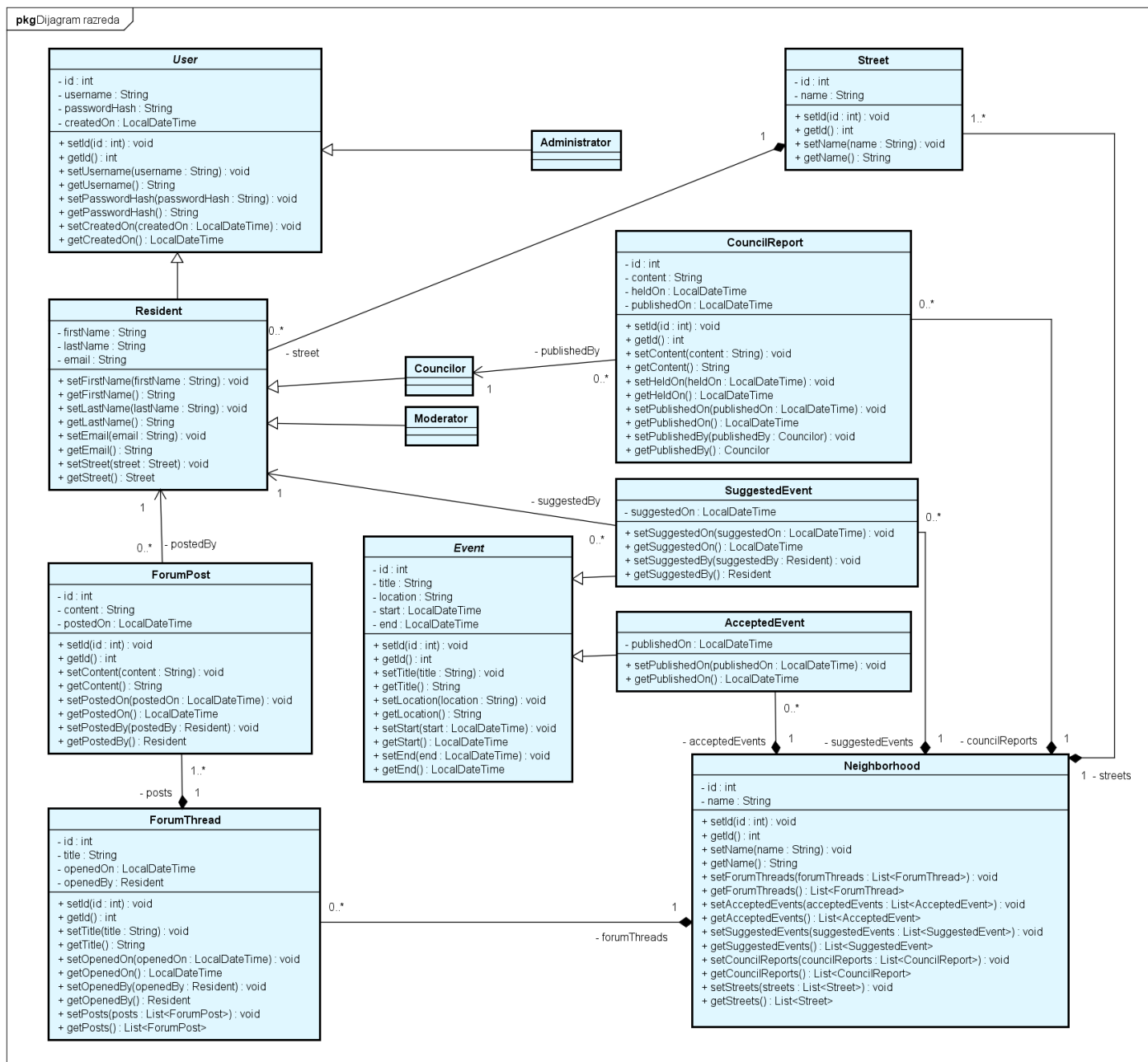
**ForumPost** sadrži tekst objave i vrijeme objave.

Razred **Event** predstavlja općenite informacije o nekom događaju, a sadrži naslov, lokaciju, trenutak početka i trenutak završetka (koji može biti i nepoznat).

**SuggestedEvent** je **Event** kojeg je neki stanovnik predložio. Osim reference na tog stanovnika (**Resident**), sadrži i trenutak kad ga je predložio.

**AcceptedEvent** je **Event** koji je prihvaćen i objavljen. Sadrži trenutak objavljivanja.

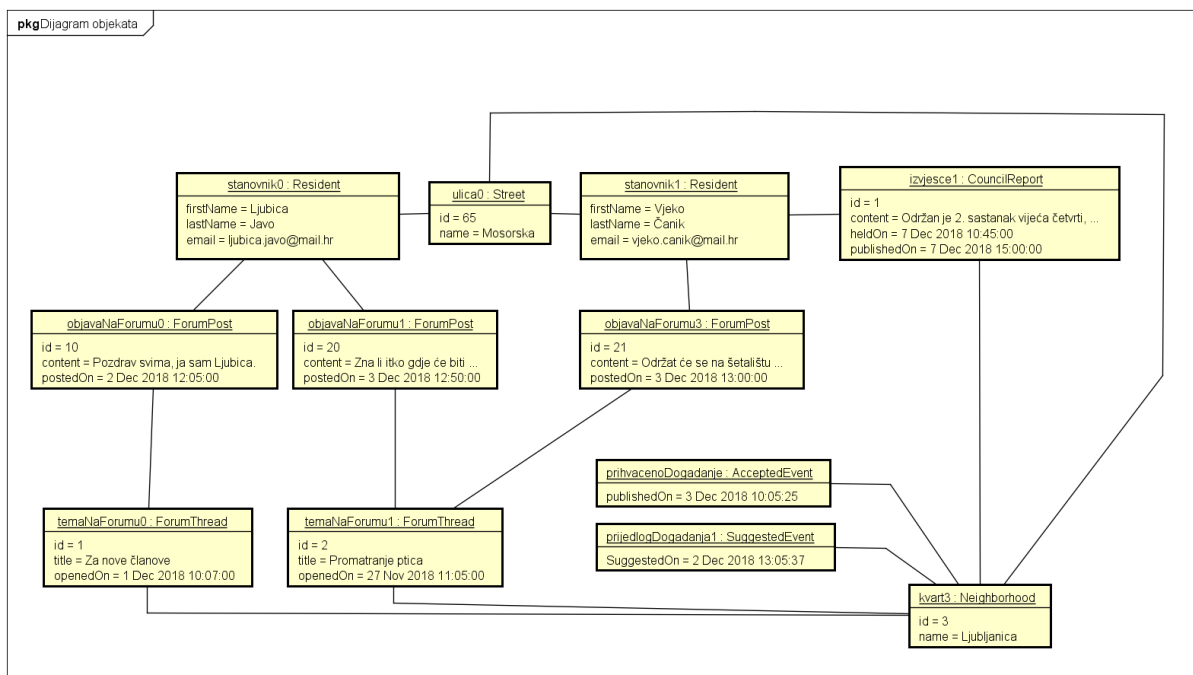
**CouncilReport** predstavlja izvješće s vijeća četvrti. Sadrži tekst izvješća, vrijeme održavanja, vrijeme objavljivanja i referencu na vijećnika (**Councilor**) koji ga je objavio.



Slika 14: Dijagram razreda

### 6.3. Dijagram objekata

Sljedeći dijagram objekata prikazuje djelomično stanje sustava nastalo registracijom i radnjama dvaju korisnika - redovnog stanovnika i vijećnika. Oba korisnika su ostavili objave na forumu. Dodatno, vijećnik je objavio izvješće vijeća četvrti, a redovni stanovnik predložio dva događanja od kojih je jedno prihvaćeno.

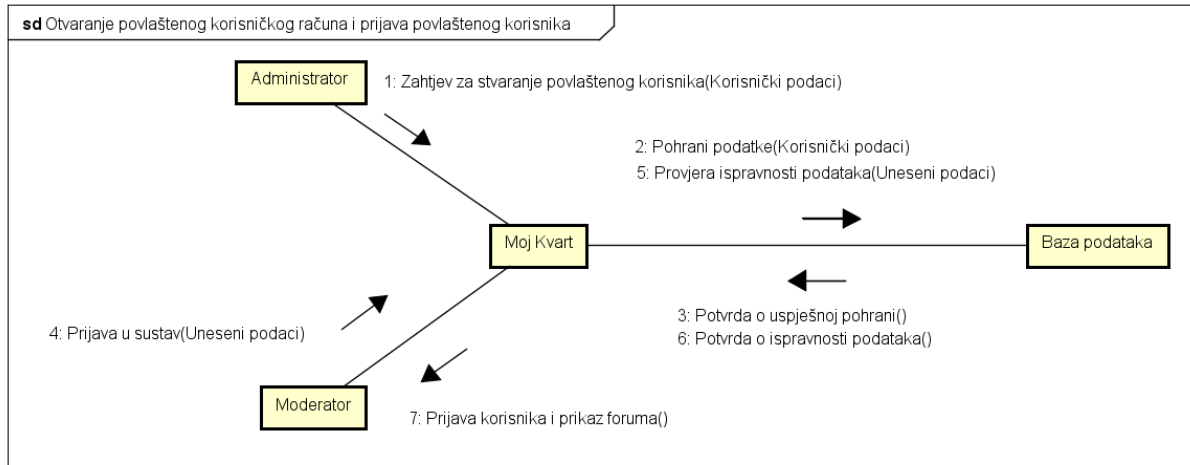


Slika 15: Dijagram objekata

## 6.4. Ostali UML dijagrami

### Dijagrami komunikacije

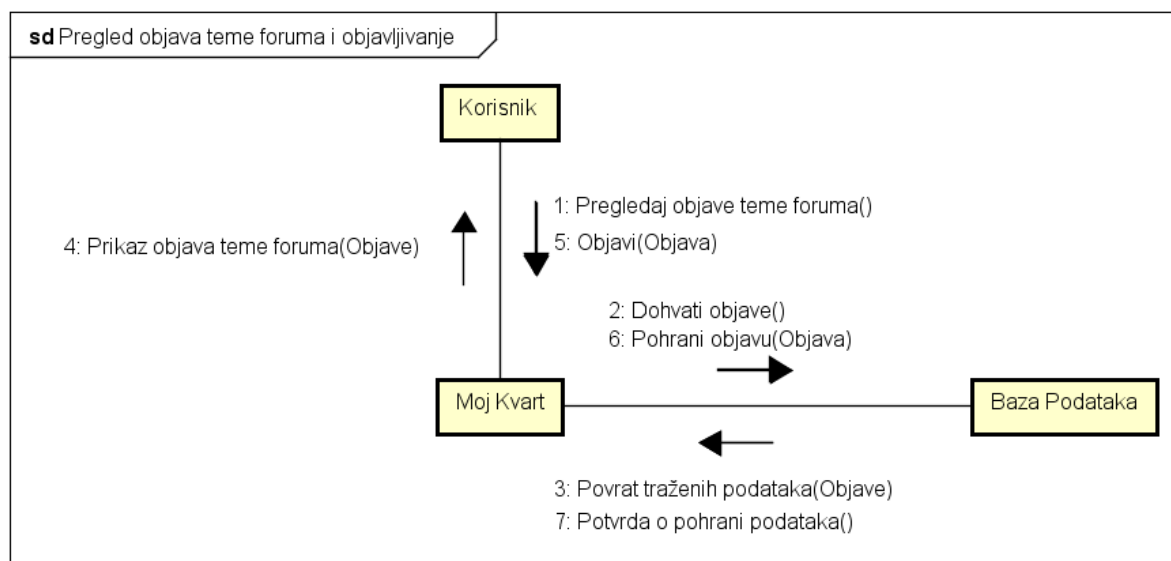
Slika 16 prikazuje dijagram komunikacije za slučaj gdje je potrebno stvoriti povlašteni korisnički račun na kojeg se potom vlasnik prijavljuje. Takav račun može otvoriti samo administrator. Administrator unosi podatke željenog korisničkog računa (ovdje s dostupnom e-mail adresom) te aplikacija pohranjuje te podatke u bazu podataka. Vlasnik računa se prijavljuje u sustav unoseći ispravne podatke čiju ispravnost aplikacija provjerava u bazi podataka te korisnika prijavljuje na sustav.



Slika 16: Dijagram komunikacije otvaranja korisničkog povlaštenog računa i prijave na isti

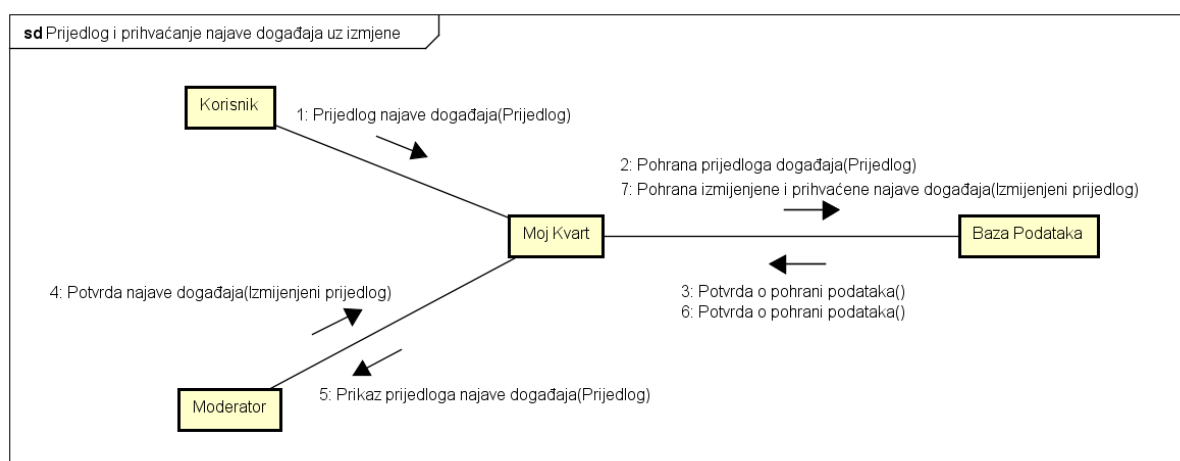


Slika 17 prikazuje dijagram komunikacije korisničkog pregledavanja foruma i objavljivanje. Korisnik otvara temu foruma kojeg aplikacija dohvaća iz baze podataka. Zatim unosi svoju objavu koju aplikacija pohranjuje u bazu podataka.



Slika 17: Dijagram komunikacije pregledavanja objava i objavljivanja na forumu

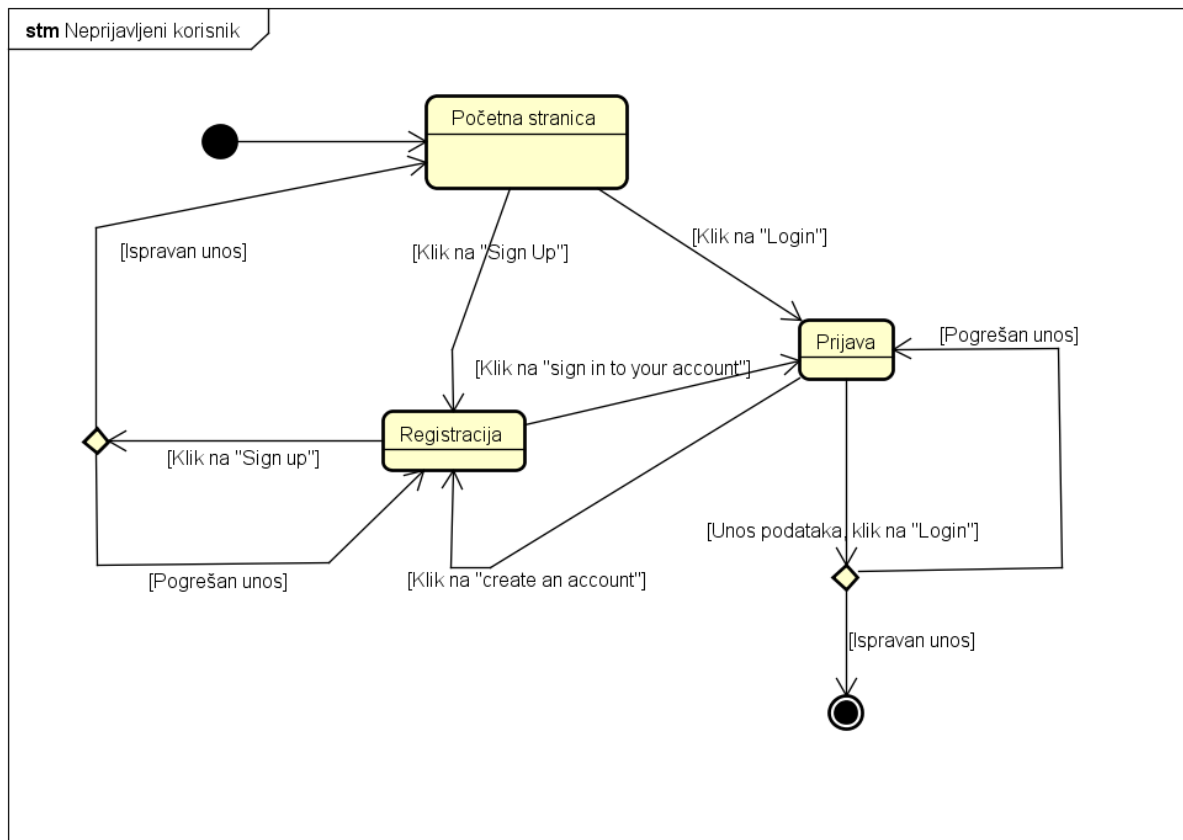
**Error! Reference source not found.** prikazuje dijagram komunikacije predlaganja najave događaja objavljenog uz izmjene. Korisnik unosi prijedlog događaja kojeg aplikacija pohranjuje u bazu podataka. Nakon pohrane, aplikacija prikazuje prijedlog moderatoru koji nakon pregledavanja uz unos željenih izmjena objavljuje prijedlog događaja - aplikacija ga unosi u bazu podataka.



Slika 18: Dijagram komunikacije predlaganja i prihvatanja događaja uz izmjene

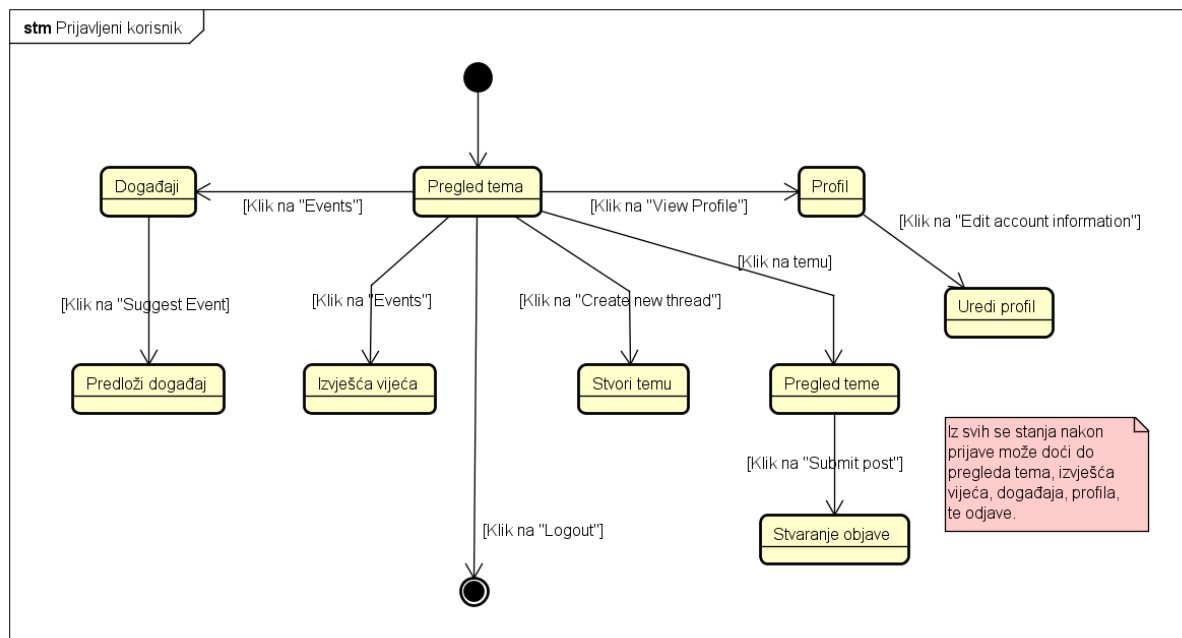
## Dijagram stanja

Neprijavljeni korisnik dolazi na početnu stranicu. Ako ima račun, može se prijaviti. Ako nema, može se registrirati isključivo kao stanovnik. Registracija i prijava se prihvataju tek kada je unos ispravan.



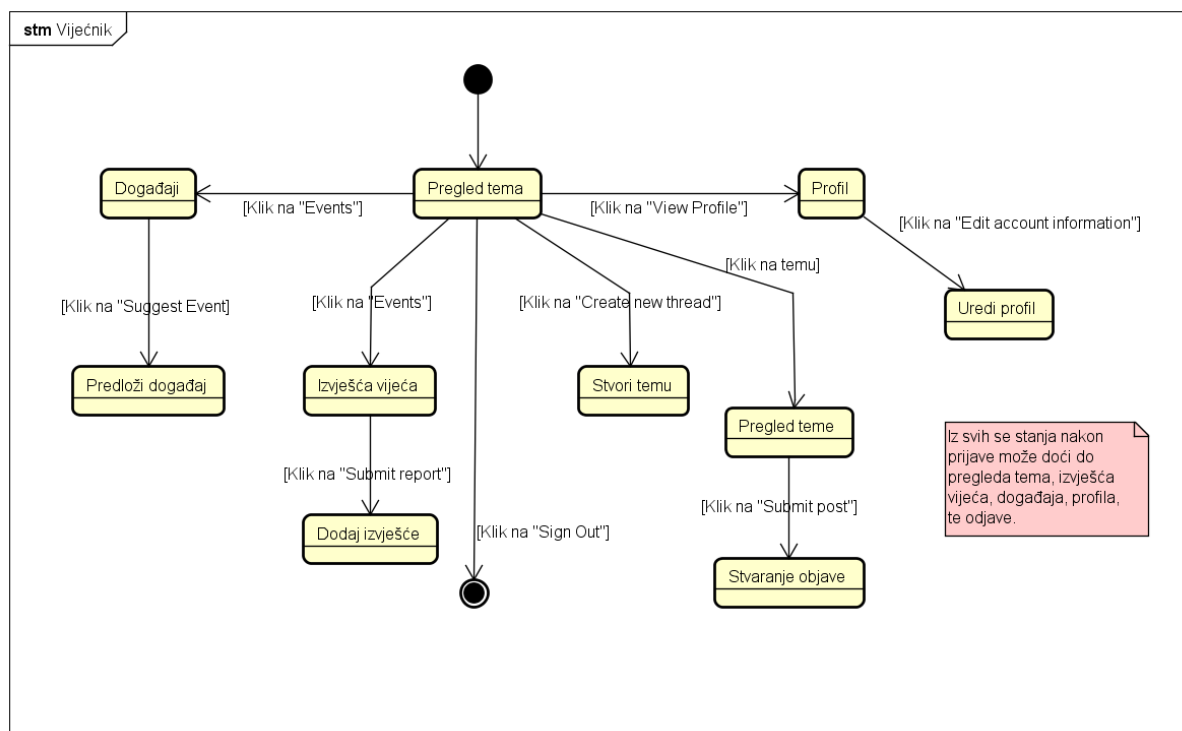
Slika 19: Dijagram stanja za neprijavljenog korisnika

Kada se korisnik prijavi, prikaže se pregled tema. Korisnik može stvarati teme na stranici „Threads“, predlagati događaje na stranici „Events“, pregledavati izvješća vijeća na stranici „Council“, pregledati svoj profil na stranici „Profile“ te se odjaviti sa „Logout“. Također može dodavati objave u temu, jednom kad ju otvori sa stranice „Threads“ ili dodavati objave u postojeću temu.



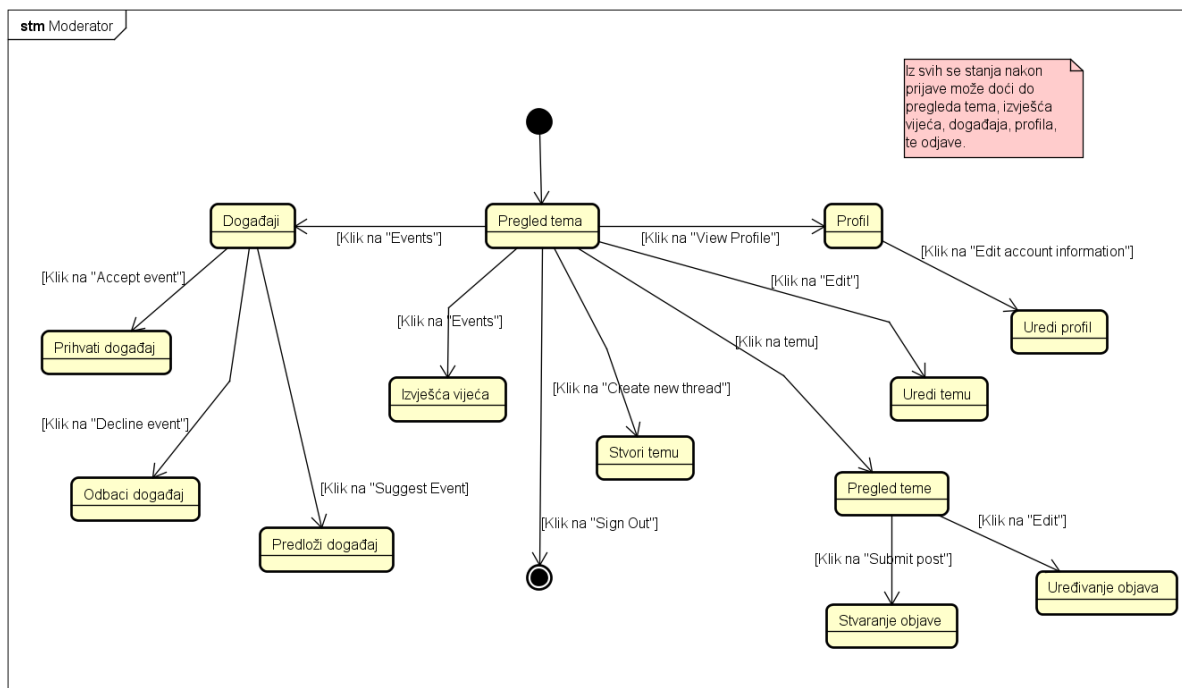
Slika 20: Dijagram stanja za prijavljenog korisnika

Vijećnik može sve što može i korisnik, i dodati izvješća vijeća na stranici „Council“.



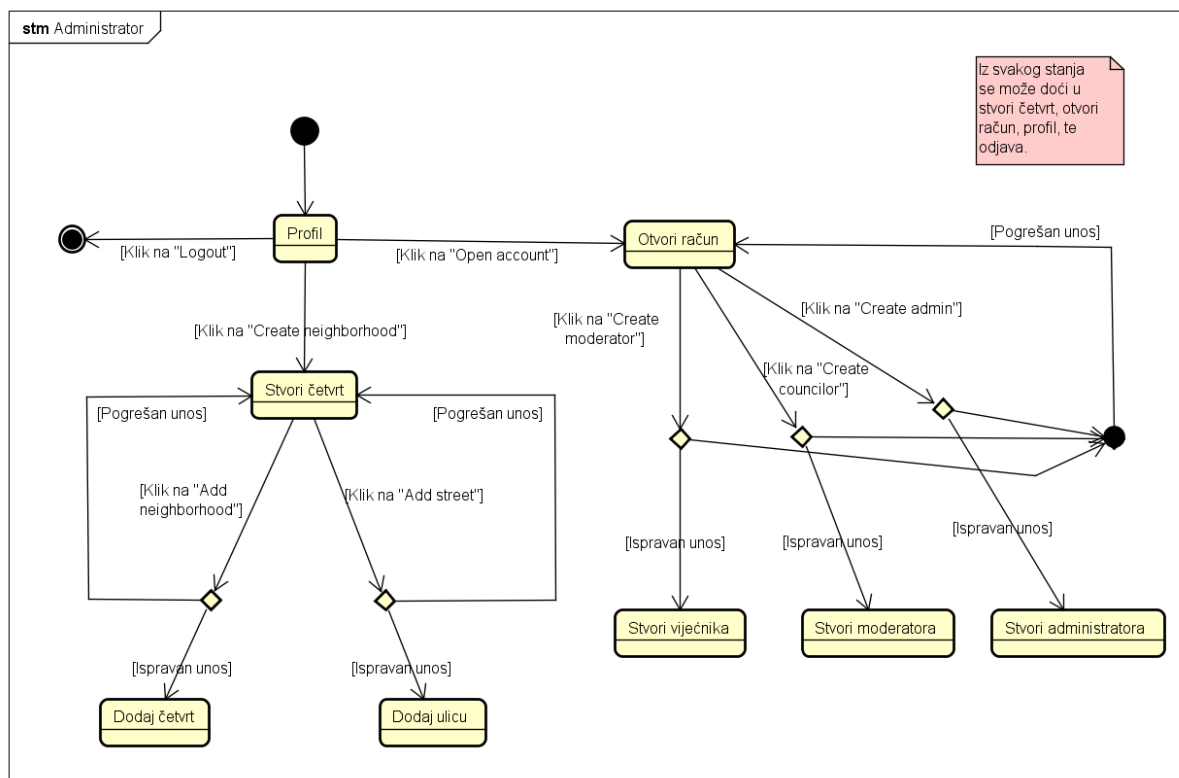
Slika 21: Dijagram stanja za vijećnika

Moderator može sve što može i korisnik, ali i uređivati teme na stranici „Threads“ te prihvaćati i odbacivati predložene događaje na stranici „Događaji“. Također može uređivati objave u temama.



Slika 22: Dijagram stanja za moderatora

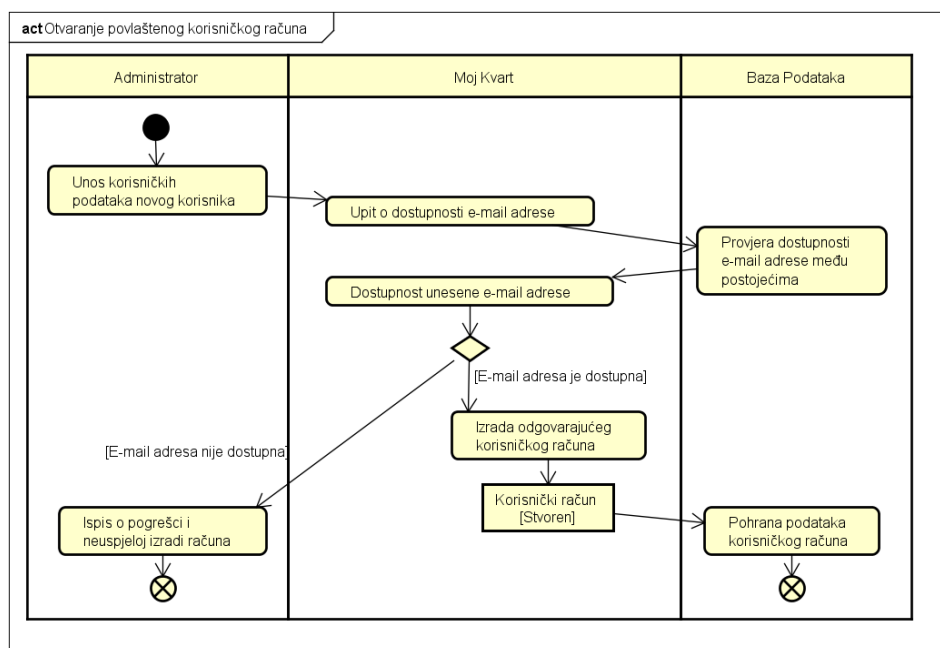
Administrator nakon prijave dolazi na stranicu svog profila. Može otvarati račune za vijećnike, moderatore i administratore. Također može stvarati četvrti i ulice.



Slika 23: Dijagram stanja za administratora

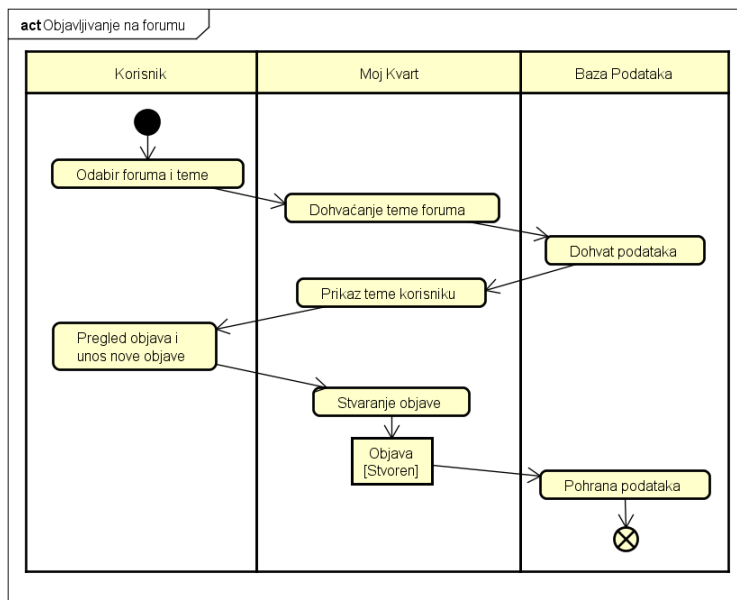
## Dijagrami aktivnosti

Slika 24 prikazuje dijagram aktivnosti za otvaranje povlaštenih korisničkih računa. Povlaštene korisničke račune može otvarati samo administrator te on upravo i u ovom slučaju započinje aktivnost. On unosi korisničke podatke za novi račun te aplikacija provjerava dostupnost unesene e-mail adrese pomoću baze podataka. Ako je e-mail adresa u uporabi - administratoru se ispisuje pogreška te novi račun nije izrađen. U drugom slučaju, ako je e-mail adresa dostupna, aplikacija izrađuje korisnički račun na temelju unesenih podataka te ga pohranjuje u bazu podataka.



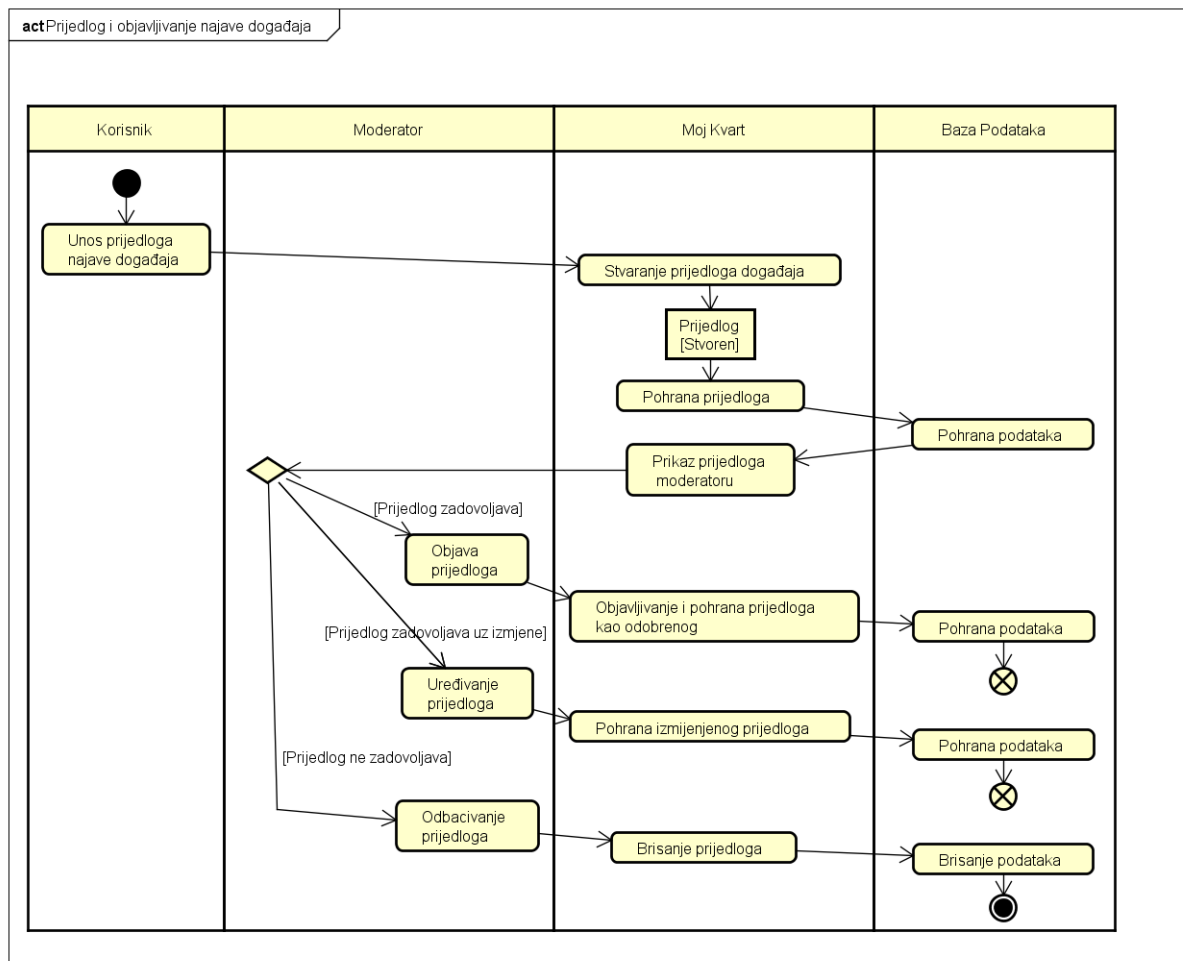
Slika 24: Dijagram aktivnosti otvaranja povlaštenog korisničkog računa

Slika 25 prikazuje dijagram aktivnosti za objavljivanje na forumu. Radnju započinje korisnik koji odabire temu na forumu. Aplikacija dohvaća temu iz baze podataka te prikazuje objave korisniku. Korisnik unosi objavu te ju aplikacija obradi i pohrani u bazu podataka.



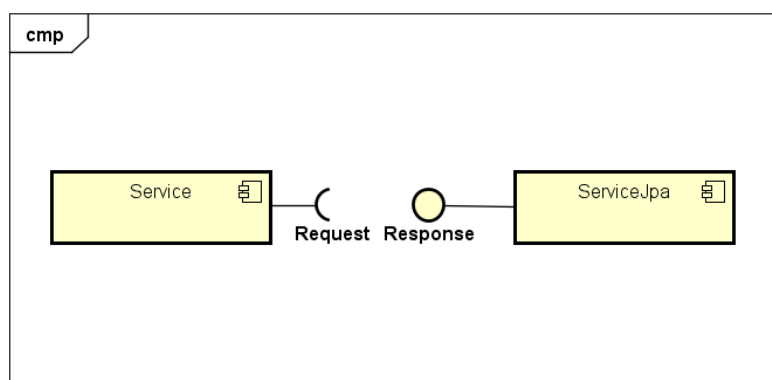
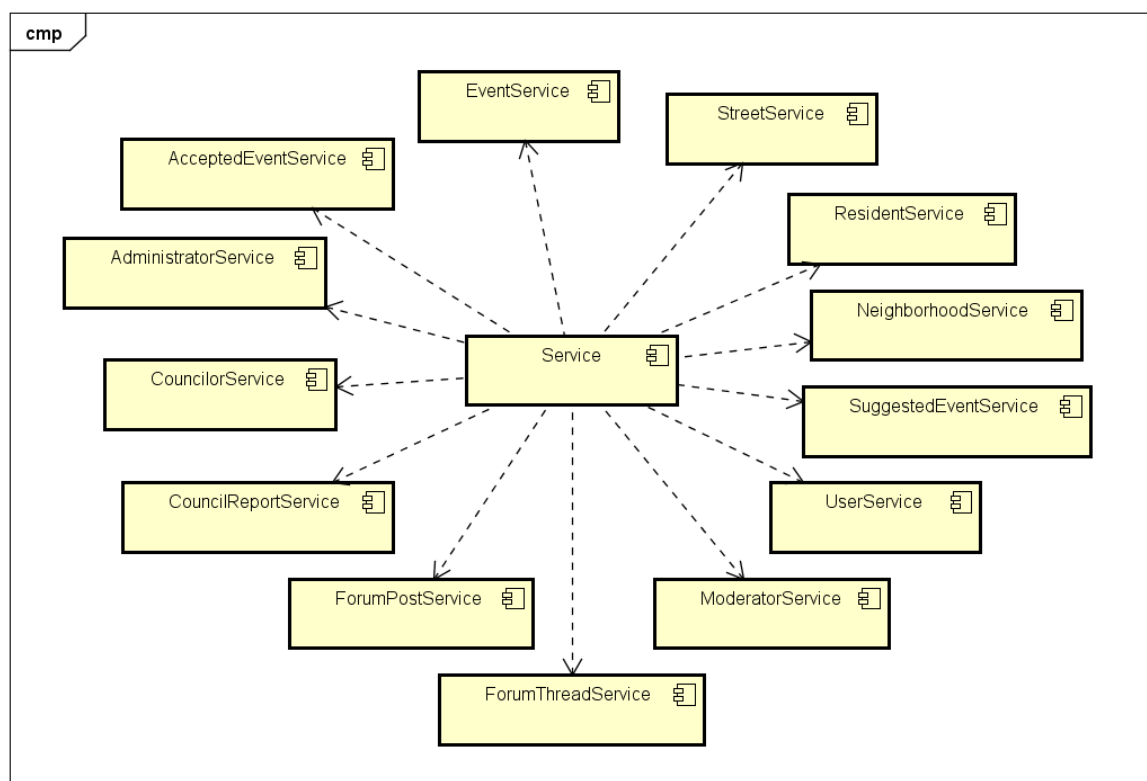
Slika 25: Dijagram aktivnosti objavljivanja na forumu

Slika 26 prikazuje dijagram aktivnosti predlaganja i objavljivanja najave događaja od strane korisnika. Proces započinje korisnikovim unosom prijedloga najave događaja. Na temelju korisnikova unosa aplikacija izrađuje prijedlog događaja te ga pohranjuje u bazu podataka. Aplikacija nakon pohrane prikazuje prijedlog događanja moderatoru na daljnje aktivnosti. Moderator, s obzirom na zadovoljstvo sadržajem prijedloga, može objaviti prijedlog kakav jest, izmijeniti ga ili u potpunosti odbaciti. Na posljetku aplikacija, na temelju moderatorove odluke, unosi odgovarajuće promjene u bazu podataka.



Slika 26: Dijagram aktivnosti predlaganja i objavljivanja najave događaja

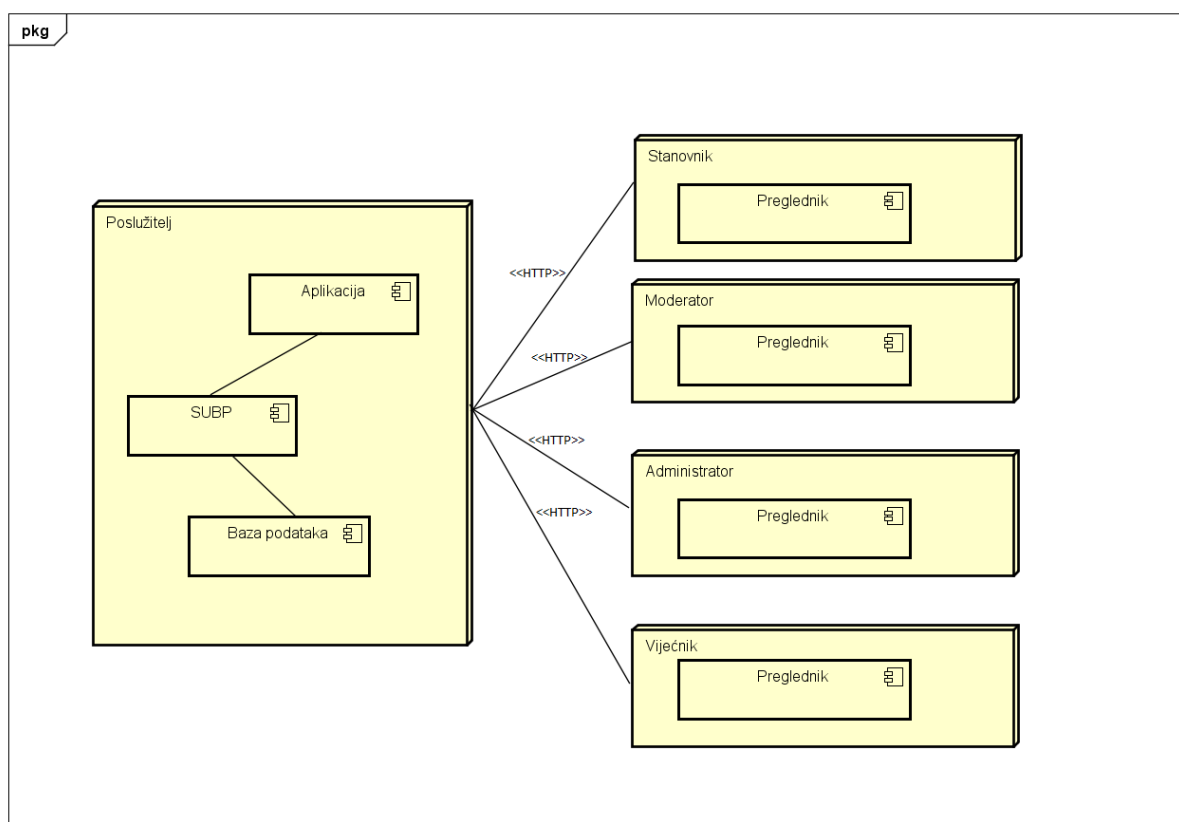


**Dijagram komponenti:***Slika 27: Dijagram komponenti 1**Slika 28: Dijagram komponenti 2*

## 7. Implementacija i korisničko sučelje

### 7.1. Dijagram razmještaja

Sustav se sastoji od poslužitelja i klijenata, koji mogu biti stanovnici, moderator, administratori i vijećnici. Klijenti komuniciraju s poslužiteljem putem HTTP-a, koristeći svoje web preglednike. Aplikacija na poslužitelju komunicira s bazom podataka preko sustava za upravljanje bazom podataka.



Slika 29: Dijagram razmještaja

## 7.2. Korištene tehnologije i alati

Backend dio web-aplikacije (*REST* poslužitelj) izrađen je uz pomoć *Spring Boot* radnog okvira u programskom jeziku *Java*.

*Spring Boot* je modul *Spring* radnog okvira koji omogućuje brzi razvoj aplikacija (*Rapid Application Development*) koristeći princip *konvencije nad konfiguracijom*.

Konkretno, korištene su različite implementacije komponenata *Controller*, *Service* i *Repository*, slijedeći na taj način koncept troslojne podjele aplikacije.

Interno, *Spring Boot* kao HTTP poslužitelj koristi *Apache Tomcat Servlet Container*.

Za mapiranje objekt-relacija (*ORM*) između *Java* aplikacije i baze podataka korišten je *Hibernate*, implementacija *Java Persistence API*-ja.

Tijekom razvoja aplikacije, kao sustav za upravljanje bazom podataka korišten je *H2* koji bazu podataka drži u radnoj memoriji, a u konačnoj inačici aplikacije, tu ulogu ima *PostgreSQL*.

Šifriranje lozinke implementirano je *BCrypt* funkcijom za računanje sažetka.

Kao alat za automatizirano upravljanje projektom korišten je *Maven*, na način da su sve potrebne ovisnosti navedene u *pom.xml* datoteci modula.

Frontend dio web-aplikacije napisan je u programskom jeziku *JavaScript*, oslanjajući se na *React* biblioteku, najčešće korištenu za razvoj jednostraničnih web-aplikacija.

Sastoji se od komponenata koje, umjesto običnog *HTML*-a, koriste tehnologiju *JSX*.

*JSX* je sintaktička nadogradnja *JavaScripta* koja olakšava opis izgleda pojedine komponente kroz oznake (*tagove*) slične *HTML*-u.

Dodatno, za vizualno oblikovanje komponenti korišten je stilski jezik *CSS*.

Za učinkovito pisanje koda, korišteno je integrirano razvojno okruženje *IntelliJ IDEA* i *Vim* uređivač teksta.

Svi UML dijagrami priloženi u dokumentaciji izrađeni su u programu *Astah UML*.

Za praćenje promjena i upravljanje izvornim kodom korišten je sustav *Git*, konkretno udaljeni repozitorij na web-poslužitelju *GitLab*.

### 7.3. Isječak programskog koda vezan za temeljnu funkcionalnost sustava

Pozadina foruma *Moj Kvart* pisana je u *Spring* radnom okviru. Spring radni okvir radi na MVC obrascu i ovo su isječci koda koji implementiraju temeljne funkcionalnosti:

Tablice u bazi podataka označene su `@Entity` anotacijama. Svi stupci su reprezentirani varijablama željenog tipa, a ograničenja se mogu specificirati parametrom u `@Column` anotaciji. Uz varijable, u razredima su napisane i pomoćne metode koje određuju na koji način se pristupa podacima u bazi podataka. *Spring* radni okvir omogućuje korisniku brz i jednostavan odabir korištene baze podataka, u ovom slučaju H2.

```
@Entity
@Inheritance(strategy = InheritanceType.JOINED)
public class User {

    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;

    @Column(nullable = false, unique = true)
    private String username;

    @Column(nullable = false)
    private String passwordHash;

    @Column(nullable = false)
    private LocalDateTime createdOn;

    public Integer getId() { return id; }

    public void setId(Integer id) { this.id = id; }

    public String getUsername() { return username; }

    public void setUsername(String username) { this.username = username; }

    public String getPasswordHash() { return passwordHash; }

    public void setPasswordHash(String passwordHash) { this.passwordHash = passwordHash; }

    public LocalDateTime getCreatedOn() { return createdOn; }

    public void setCreatedOn(LocalDateTime createdOn) { this.createdOn = createdOn; }
}
```

Slika 30: Primjer tablice u bazi podataka

Anotacija `@RestController` označava da je razred *Controller*. Controlleri upravljaju korisničkim zahtjevima i pomažu ostatku sustava obrađivati i mapirati REST zahtjeve na prave lokacije.

```
@RestController
@RequestMapping("/users/admins")
@Secured("ROLE_ADMIN")
public class AdministratorController {

    @Autowired
    private AdministratorService administratorService;

    @GetMapping("")
    public List<UserDTO> getAdministrators() {
        return administratorService.listAll().stream()
            .map(ControllerUtil::toUserDTO).collect(Collectors.toList());
    }

    @GetMapping("/{id}")
    public UserDTO getAdministrator(@PathVariable Integer id) { return toUserDTO(administratorService.fetch(id)); }

    @PostMapping("")
    public ResponseEntity<UserDTO> createAdministrator(@RequestBody UserIDTO dto) {
        UserDTO administrator = toUserDTO(administratorService.add(toAdministrator(dto)));
        return ResponseEntity.created(URI.create(
            "/users/administrators/" + administrator.getId()))
            .body(administrator);
    }

    private Administrator toAdministrator(UserIDTO dto) {
        Administrator administrator = new Administrator();
        administrator.setUsername(dto.getUsername());
        administrator.setPasswordHash(dto.getPassword());
        return administrator;
    }
}
```

Slika 31: Primjer Controllera

Sufiks „DTO“ označava da je razred „Data transfer object“ tj. da prenosi podatke između procesa. DTO-i se često koriste jer uvelike olakšavaju komunikaciju između različitih dijelova sustava (u ovom slučaju front-enda i back-enda) jer na uniforman način dostavljaju informacije između njih.

```
public class UserDTO {

    private Integer id;
    private String name;
    private String role;

    public UserDTO(Integer id, String name, String role) {
        this.id = id;
        this.name = name;
        this.role = role;
    }

    public Integer getId() { return id; }

    public String getName() { return name; }

    public String getRole() { return role; }
}
```

Slika 32: Primjer DTO-a

## 7.4. Ispitivanje programskog rješenja

Ispitivanje rada aplikacije *Moj kvart* rađeno je uporabom *React Developer Tools* alata koji se lako uključi u bilo koji poznatiji internet preglednik. *React Developer Tools* je koristan jer prikazuje sav mrežni promet koji se odvija pri komunikaciji korisničkog sučelja i baze podataka. Moguće je detaljno analizirati HTTP zahtjeve, gledati sadržaj JSON objekata, otklanjati pogreške korištenjem *debug* opcije itd. Sve ovo je moguće promatrati u pregledniku jer se aplikacija izvodi u njemu i ako nešto ne radi, preglednik ispisuje grešku te upućuje na dio koda u kojem je nastao problem itd.

Slika 33 prikazuje primjer generiranja mrežnog prometa pri uspješnoj prijavi stanovnika neke četvrti. Mogu se uočiti HTTP zahtjevi POST i GET koji su rezultirali uspjehom o čemu govori statusni kod 200.

The screenshot shows a web application interface for 'MyNeighborhood' with a navigation bar containing 'Forum', 'Council', and 'Events'. The main content area is titled 'Threads' and displays a list of threads with columns for 'Title', 'Author', and 'Date'. Two threads are visible: 'Prva tema' by 'Stanovnik1' and 'Trecu tema' by 'Stanovnik2', both dated '17/1/19'. Below the list is a 'Create new thread' form with a 'Title...' input field and a 'Submit' button. At the bottom, the Firefox Network tab is open, showing a list of network requests. The first request is a POST to 'login' with a status of 200. The second request is a GET to 'user' with a status of 200. The third request is a GET to 'threads' with a status of 200. The network tab also shows the 'Cause' and 'Type' of each request, as well as the 'Transferred' and 'Size' of the data.

Status	Method	File	Domain	Cause	Type	Transferred	Size
200	POST	login	localhost:3000	fetch	plain	898 B	0 B
200	GET	user	localhost:3000	fetch	json	471 B	46 B
200	GET	threads	localhost:3000	fetch	json	640 B	515 B

Slika 33: Primjer prikaza mrežnog prometa pri prijavi stanovnika četvrti u pregledniku Firefox

Slika 34 prikazuje primjer kako preglednik prikazuje grešku u kodu i kako ispisuje točnu liniju koda za koju misli da je uzrokovala problem. Dana greška se pojavila pri testiranju prilikom pokušaja dohvaćanja korisnika iz baze podataka prije nego je korisnik bio prijavljen. Problem je bio u tome što se pokušava parsirati JSON objekt koji je u danom trenutku poprimao NULL vrijednost. Greška je otklonjena obradom iznimke i pozivanjem dohvaćanja korisnika samo kad je korisnik prijavljen u sustav.

### Unhandled Rejection (SyntaxError): Unexpected end of JSON input

(anonymous function)  
C:/Users/dduki/IdeaProjects/Javoljupci/IzvorniKod/moj-kvart-frontend/src/components/App/App.js:28

```
25 |  
26 |   componentDidMount() {  
27 |     fetch('/user', { credentials: 'include' })  
> 28 |       .then(response => response.json())  
29 |     ^  
30 |     .then(user => {  
31 |       this.setState({loggedIn: true, role: user.role, id: user.id})  
    |     }  
    |   }  
    | }
```

View compiled

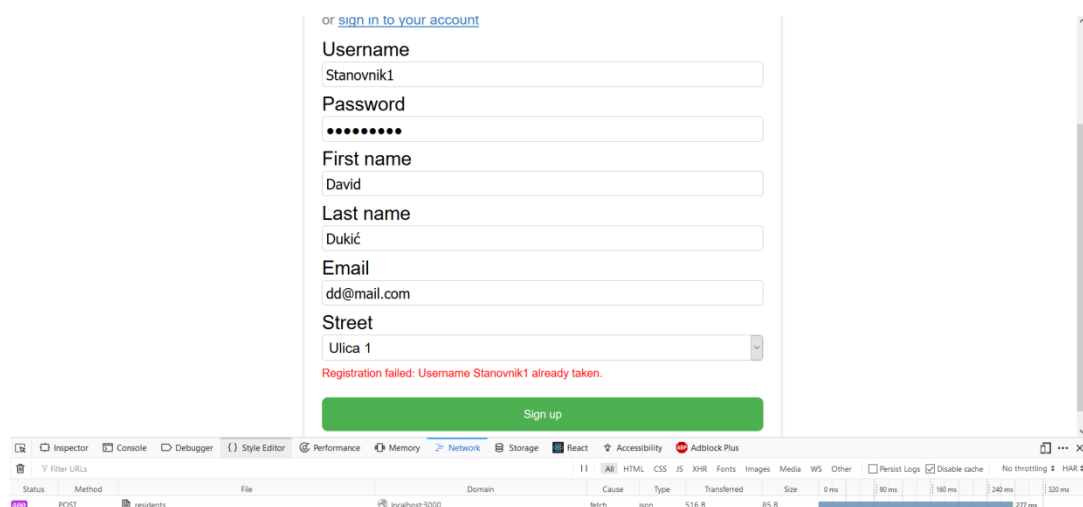
This screen is visible only in development. It will not appear if the app crashes in production.  
Open your browser's developer console to further inspect this error.

Slika 34: Ispis pogreške u pregledniku

## Ispitni scenariji

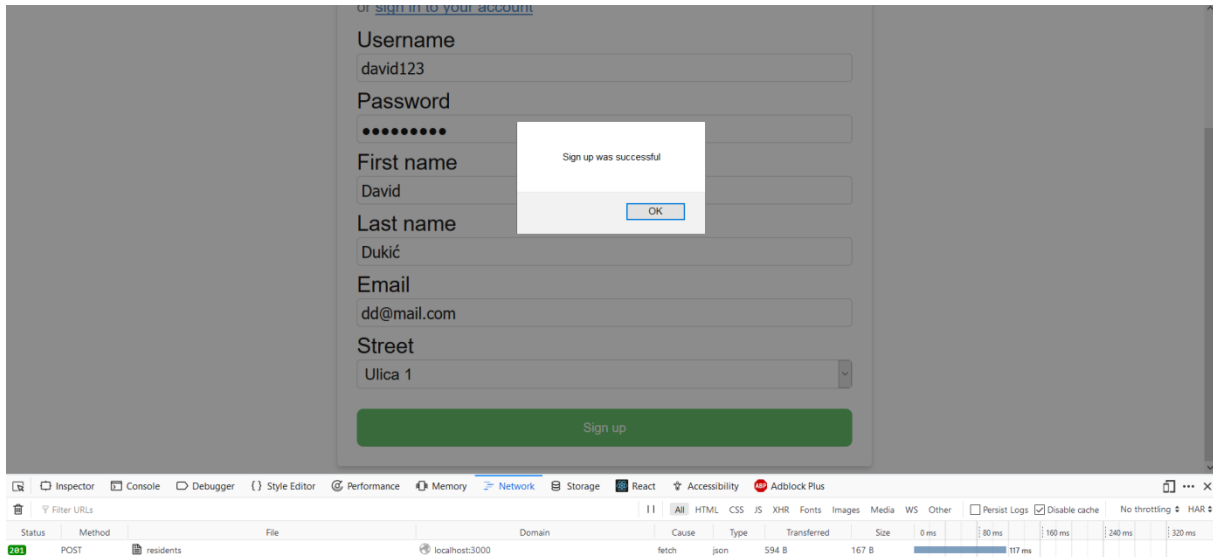
### 1. Testiranje registracije korisnika

Testiranje registracije korisnika koje je prikazano na slici 35 je izvedeno nakon povezivanja korisničkog sučelja s bazom podataka. Ispituje se je li moguće unijeti korisnika s istim korisničkim imenom kakvo već postoji u bazi podataka. Test se pokazao uspješan jer je sustav odbio izvesti ovakvu akciju. Sustav je vratio i popratnu poruku o tome kako je korisničko ime „Stanovnik1“ već zauzeto, a preglednik je pokazao da to nije moguće napraviti HTTP odgovorom 400 koji kaže da je u pitanju loš zahtjev (*bad request*).



Slika 35: Pokušaj registracije korisnika s korisničkim imenom koje je zauzeto

Ako promijenimo korisničko ime u neko koje već nije zauzeto, npr. „david123“ sustav se ponaša i dalje korektno te nakon provjere uspješno registrira novog korisnika. Povratnu informaciju korisniku daje preglednik, a programeru HTTP odgovor statusnog koda 201. Na sličan način se provjerava i kako sustav reagira na prekratku lozinku ili email koji je već u uporabi i slično.



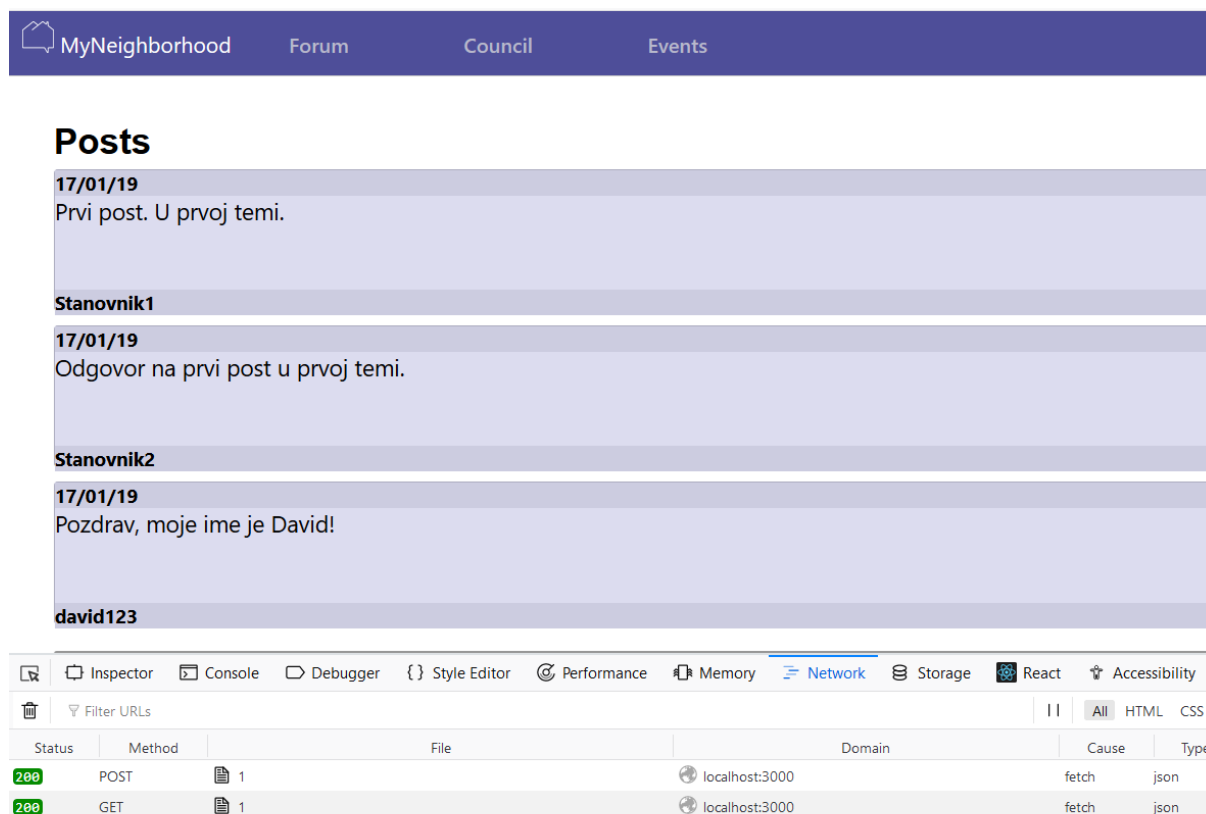
*Slika 36: Primjer uspješne registracije*

## 2. Testiranje objavljivanja na forumu

Recimo da smo se ulogirali kao redovni stanovnik i želimo napisati objavu na forum. Testiranje ove funkcionalnosti zahtijeva da se korisnik uspješno prijavi i uspješno objavi nešto. Prijava nas ne zanima u ovom testnom slučaju. Podrazumijeva se da se korisnik uspješno prijavio. Neka to bude npr. novostvoreni korisnik iz testnog scenarija 1 (korisnik „david123“).

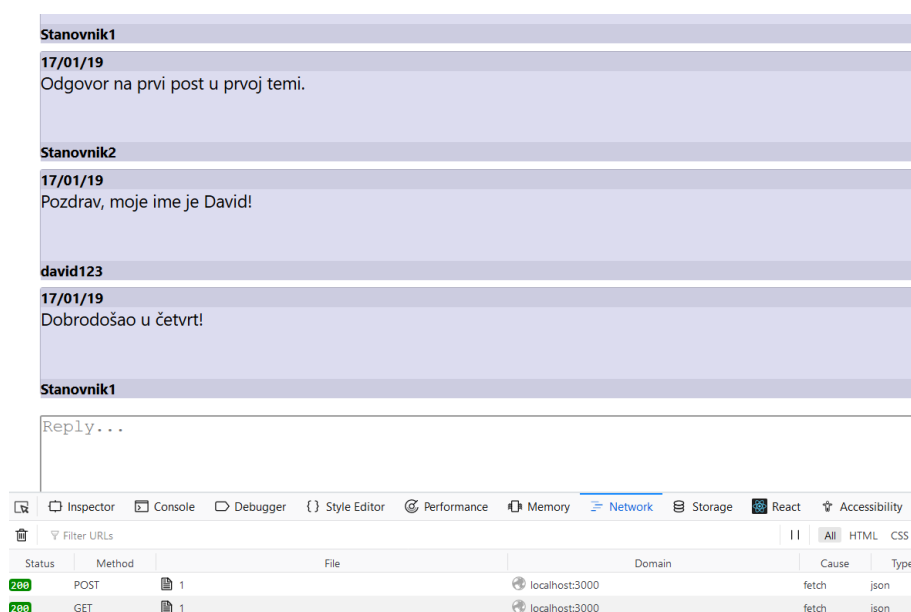
Na slici 37 je prikazan pokušaj objave na forumu koji rezultira uspjehom (vidimo da je HTTP odgovor statusnog koda 200 i da je objava dodana u forum). Da bismo se u to dodatno uvjerali možemo se odjaviti i ulogirati sa korisnikom koji živi u istoj četvrti kao korisnik „david123“, a to je npr. korisnik s korisničkim imenom „Stanovnik1“.





Slika 37: Prikaz testiranja objavljivanja na forumu

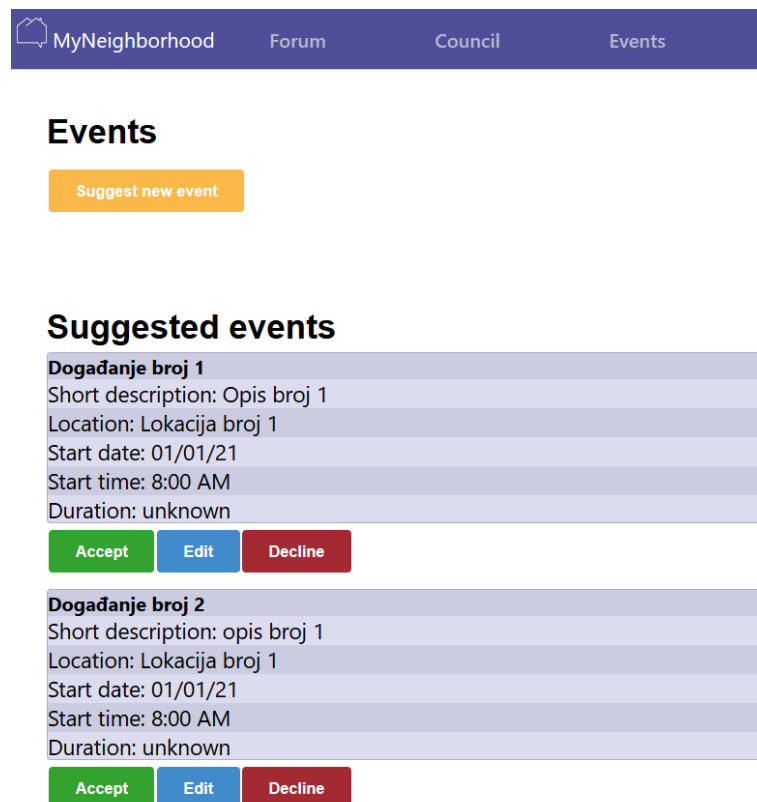
Na slici 38 prikazano je što korisnik „Stanovnik1“ vidi od objava na svome forumu i napisana je neka njegova dodatna objava. Ovime je uspješno testirana funkcionalnost objavljivanja na forumu.



Slika 38: Prikaz što "Stanovnik1" vidi na forumu nakon objave korisnika "david123"


### 3. Testiranje odobravanja i odbijanja predloženih događaja

Predložene događaje objavljuje moderator. Slika 39 prikazuje dva predložena događaja koja čekaju da ih moderator odobri. Želimo testirati tako da odobrimo događaj 1, a odbijemo događaj 2.



Slika 39: Događaji koji čekaju na odobravanje su vidljivi samo moderatoru

Slika 40 prikazuje odobreni događaj 1 i mrežni promet koji se generira kad moderator klikne na gumb accept. Odobravanje je uspješno testirano. Vidljivo je da je događaj prešao iz predloženih u prihvaćene.

 MyNeighborhood Forum Council Events











## Events


**Događanje broj 1**  
Short description: Opis broj 1  
Location: Lokacija broj 1  
Start date: 01/01/21  
Start time: 8:00 AM  
Duration: unknown

Suggest new event

## Suggested events

**Događanje broj 2**  
Short description: opis broj 1  
Location: Lokacija broj 1  
Start date: 01/01/21

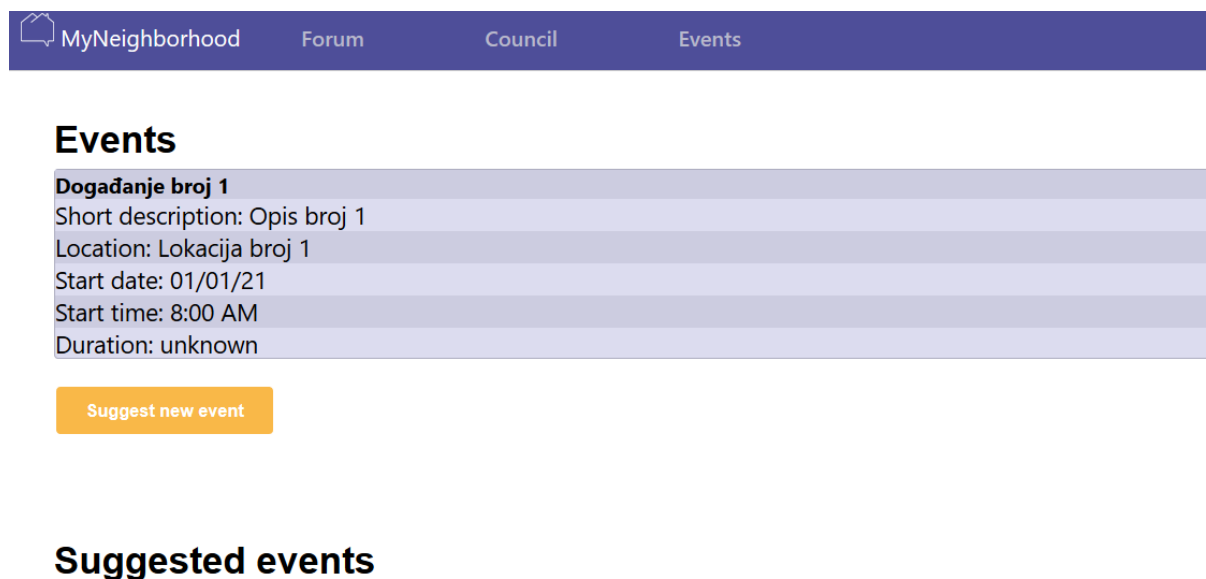
 Inspector  Console  Debugger  Style Editor  Performance  Memory  Network  Storage  React  Accessibility

 Filter URLs

Status	Method	File	Domain	Cause	Type
200	GET	accepted	localhost:3000	fetch	json
200	GET	suggested	localhost:3000	fetch	json
201	POST	1	localhost:3000	fetch	json
200	GET	accepted	localhost:3000	fetch	json
200	GET	suggested	localhost:3000	fetch	json

Slika 40: Odobrenje predloženog događaja 1

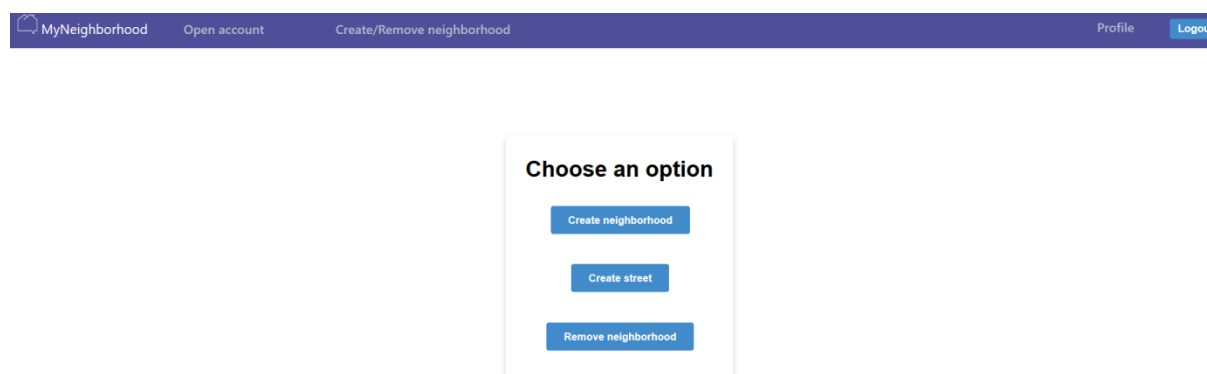
Slika 41 prikazuje izgled u aplikaciji nakon što moderator stisne gumb decline za događaj broj 2. Predloženi događaj je uklonjen iz baze podataka, a ne nalazi se više ni u predloženim ni u prihvaćenim događajima.



*Slika 41: Prikaz korisničkom sučelja nakon odbijanja predloženog događaja 2*

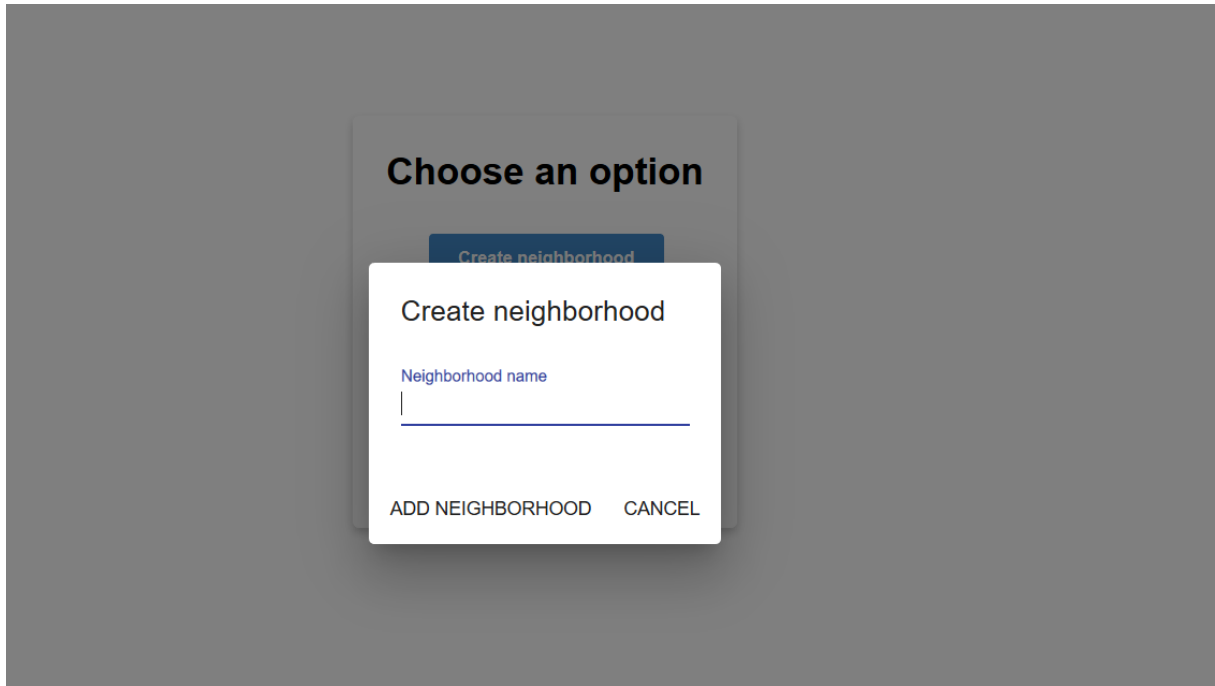
4. Testiranje funkcionalnosti dodavanja četvrti u bazu podataka koristeći administratorov dio aplikacije

Administrator može stvarati četvrti i dodavati ulice u njihov sastav. Sljedećim testnim scenarijem se želi pokazati funkcionalnost stvaranja četvrti kroz dio aplikacije namijenjen samo administratoru. Slika 42 prikazuje kako izgleda sučelje za stvaranje četvrti i ulica nakon prijave administratora. Administrator bira „Create neighborhood“.



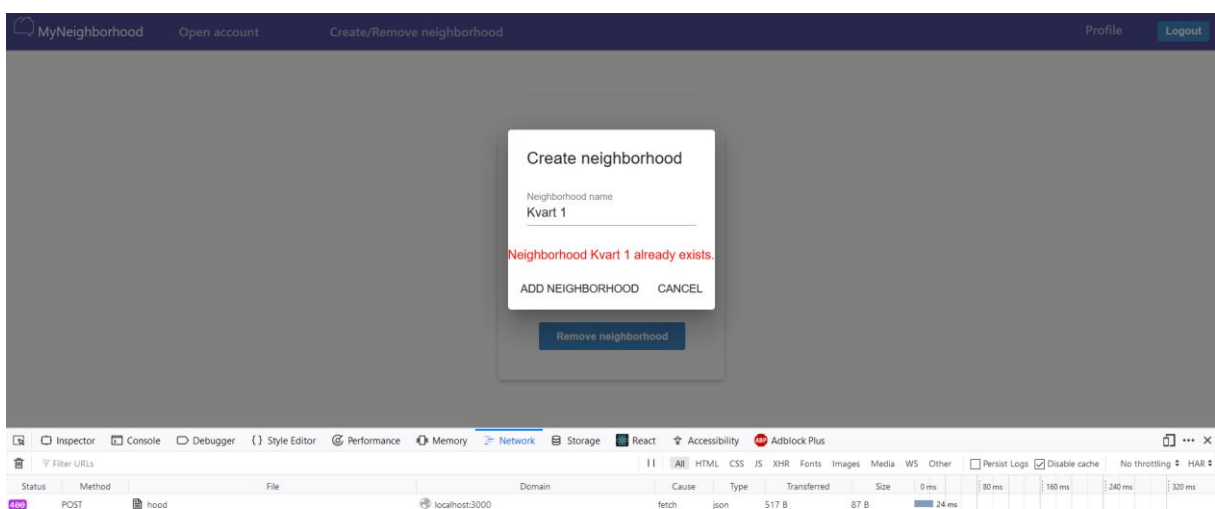
*Slika 42: Izgled korisničkog sučelja za stvaranje četvrti i ulica te brisanje četvrti*

Slika 43 prikazuje prozor koji se otvara nakon pritiska na „Create neighborhood“. U tom prozoru administrator može upisati ime četvrti.



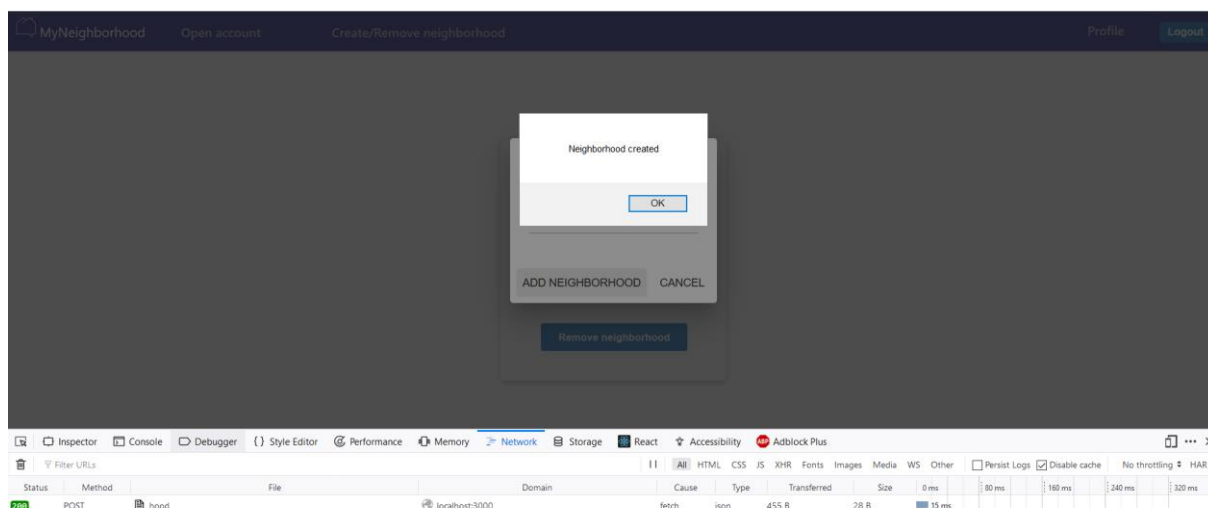
Slika 43: Prozor koji se otvara klikom na gumb "Create neighborhood"

Slika 44 prikazuje što se događa ako administrator unese ime četvrti koje već postoji u bazi podataka. Dogodila se greška što je i bilo očekivano ponašanje. Korisniku se ispisuje povratna poruka, a programer po statusnom kodu 400 vidi da nije prošlo dodavanje kvarta s postojećim imenom u bazu podataka.



Slika 44: Rezultat za unos imena koje već postoji u bazi podataka

Konačno, na slici 45 možemo vidjeti uspješno dodavanje novog kvarta s popratnom porukom korisniku i statusnim kodom 200 kao znakom programeru da je dodavanje nove četvrti u bazu podataka uspješno provedeno.



*Slika 45:Uspješno dodavanje nove četvrti u bazu podataka*

## 7.5. Upute za instalaciju

### Instalacija JDK - Windows:

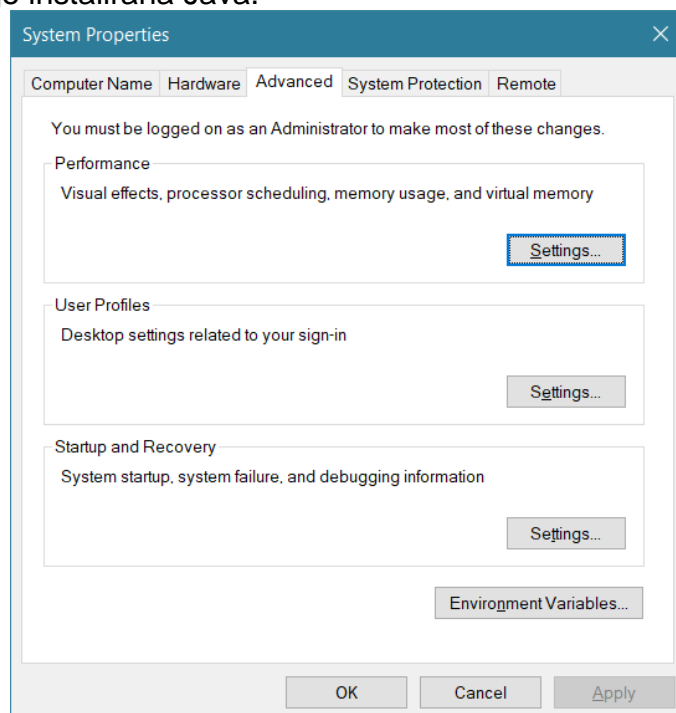
Sa web-lokacije

<https://www.oracle.com/technetwork/java/javase/downloads/index.html> potrebno je skinuti odgovarajuću verziju *Java Developers Kit*-a (ovisno o operacijskom sustavu 32 ili 64-bitnu verziju) i instalirati je.

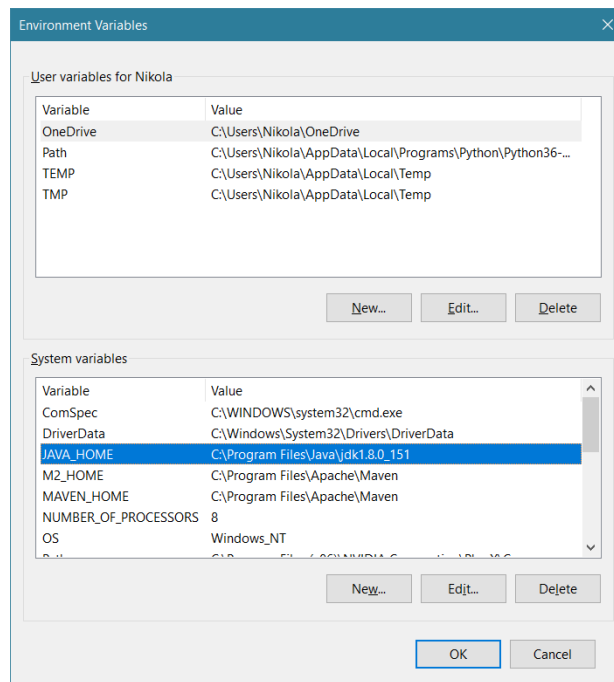


Slika 46: Prikaz Oracle web stranice s koje se skida JDK

Da bi se dovršila instalacija, potrebno je otići u izbornik *System Properties* i u podizborniku *Environment Variables* dodati varijablu `JAVA_HOME` koja pokazuje na direktorij u kojem je instalirana Java.



Slika 47: Prikaz varijabli okruženja 1



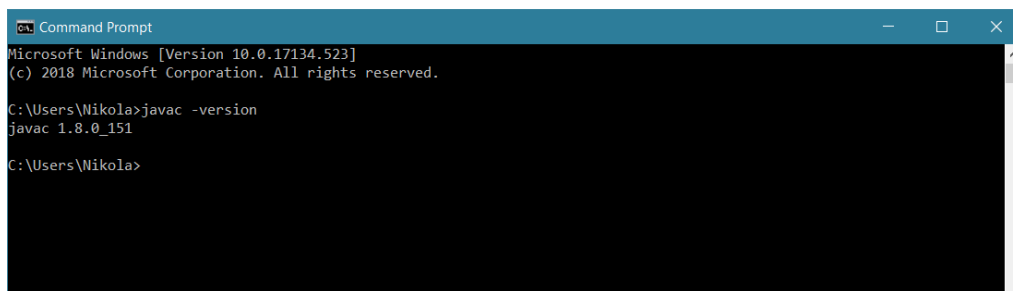
Slika 48: Prikaz varijabli okruženja 2

Također, u varijablu PATH treba dodati „%JAVA\_HOME%\bin“ i „%JAVA\_HOME%\jre\bin“.

Varijabla okruženja PATH operacijskom sustavu govori gdje da traži programe koje korisnik želi pokrenuti iz naredbenog retka

Za provjeru ispravnosti instalacije, dovoljno je otvoriti *Command prompt* (izbornik *Start -> cmd*) i upisati naredbe „java -version“ te „javac -version“.

Ispis bi trebao biti sličan ovome:



Slika 49: Ispis u command promptu



## Instalacija Maven-a:

Za instalaciju alata za upravljanje paketima *Maven*, dovoljno je skinuti instalacijski paket s web-lokacije <https://maven.apache.org/download.cgi>, raspakirati ga pomoću alata *WinZip* ili *WinRAR* i instalirati ga duplim klikom na .bin datoteku.

### Files

Maven is distributed in several formats for your convenience. Simply pick a ready-made binary distribution archive and follow the [installation instructions](#). Use a source archive if you intend to build Maven yourself.

In order to guard against corrupted downloads/installations, it is highly recommended to [verify the signature](#) of the release bundles against the public [KEYS](#) used by the Apache Maven developers.

	Link	Checksums	Signature
Binary tar.gz archive	<a href="#">apache-maven-3.6.0-bin.tar.gz</a>	<a href="#">apache-maven-3.6.0-bin.tar.gz.sha512</a>	<a href="#">apache-maven-3.6.0-bin.tar.gz.asc</a>
Binary zip archive	<a href="#">apache-maven-3.6.0-bin.zip</a>	<a href="#">apache-maven-3.6.0-bin.zip.sha512</a>	<a href="#">apache-maven-3.6.0-bin.zip.asc</a>
Source tar.gz archive	<a href="#">apache-maven-3.6.0-src.tar.gz</a>	<a href="#">apache-maven-3.6.0-src.tar.gz.sha512</a>	<a href="#">apache-maven-3.6.0-src.tar.gz.asc</a>
Source zip archive	<a href="#">apache-maven-3.6.0-src.zip</a>	<a href="#">apache-maven-3.6.0-src.zip.sha512</a>	<a href="#">apache-maven-3.6.0-src.zip.asc</a>

### *Slika 50: Prikaz web stranice za skidanje Maven paketa*

Pri instalaciji, korisno je provjeriti je li varijabla okruženja `%JAVA_HOME%` ispravno podešena.

Instalacija se može provjeriti naredbom „`mvn -v`“ u ljusci operacijskog sustava.

Jednom kad je instaliran, *Maven* će se pobrinuti za instalaciju preostalih paketa potrebnih za rad foruma *Moj Kvart*.

## Povezivanje s PostgreSQL bazom podataka:

Ukoliko na računalu nije instaliran sustav za upravljanje bazom podataka, potrebno je instalirati PostgreSQL koristeći se uputama na web-lokaciji:

<https://www.postgresql.org/docs/9.3/tutorial-install.html>.

Nakon što je PostgreSQL instaliran, dovoljno je izvesti sljedeću naredbu:

```
createdb -h localhost -p 5432 -U postgres mojkvartdb password pass123
```

te u *moj-kvart-backend* modulu dodati u datoteku

src/main/resources/application.properties sljedeće:

```
spring.datasource.url=jdbc:postgresql://localhost:5432/mojkvartdb
spring.datasource.username=postgres
spring.datasource.password=pass123
spring.jpa.hibernate.dll-auto=create
```

Ako aplikaciju deployamo na online *Heroku* poslužitelj, bazu je moguće konfigurirati tako da se izvede naredba:

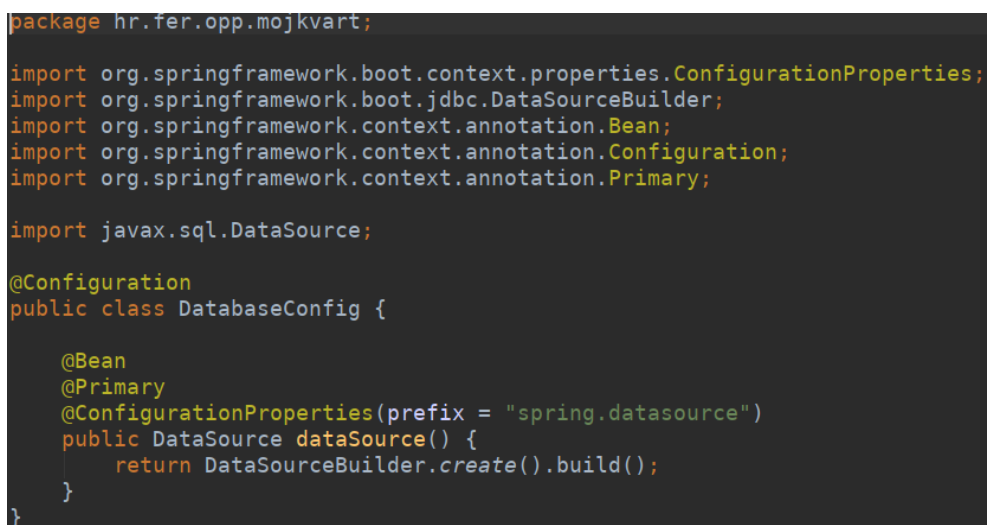
```
heroku addons:create heroku-postgresql
```

U *moj-kvart-backend* modulu u datoteku

src/main/resources/application.properties potrebno je dodati primjerice:

```
spring.datasource.driverClassName=org.postgresql.Driver
spring.datasource.maxActive=10
spring.datasource.maxIdle=5
spring.datasource.minIdle=2
spring.datasource.initialSize=5
spring.datasource.removeAbandoned=true
```

te stvoriti razred dan u nastavku.

A screenshot of a code editor showing the implementation of the DatabaseConfig class. The code is in Java and uses Spring Boot annotations. It includes imports for ConfigurationProperties, DataSourceBuilder, Bean, Configuration, Primary, and DataSource. The class is annotated with @Configuration, @Bean, and @Primary. It has a method dataSource() that returns a DataSource created by DataSourceBuilder.create().build().

```
package hr.fer.opp.mojkvart;

import org.springframework.boot.context.properties.ConfigurationProperties;
import org.springframework.boot.jdbc.DataSourceBuilder;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.Primary;

import javax.sql.DataSource;

@Configuration
public class DatabaseConfig {

    @Bean
    @Primary
    @ConfigurationProperties(prefix = "spring.datasource")
    public DataSource dataSource() {
        return DataSourceBuilder.create().build();
    }
}
```

Slika 51: Prikaz razreda DatabaseConfig

## Pokretanje backend aplikacije:

Potrebno je pozicionirati se u korijenski direktorij backend modula čiji je relativni put `IzvorniKod/moj-kvart-backend/`.

Izvođenjem naredbe:

```
mvn spring-boot:run
```

pokreće se proces prevođenja izvornog koda u `.class` datoteke i, ako je sve u redu, pokreće se poslužitelj unutar *Tomcat Containera* na portu 8080.

Alternativno, izvođenjem sljedećih naredbi moguće je zapakirati backend aplikaciju u arhivu čime se olakšava njezin prijenos i pokretanje.

```
mvn package  
java -jar target/moj-kvart-backend.1.0.jar
```

## Pokretanje frontend aplikacije:

Potrebno je pozicionirati se u korijenski direktorij frontend modula čiji je relativni put `IzvorniKod/moj-kvart-frontend/`.

Izvođenjem naredbe:

```
npm install
```

instaliraju se sve potrebne ovisnosti u direktorij `node_modules`.

Nakon instalacije, dovoljno je izvesti naredbu:

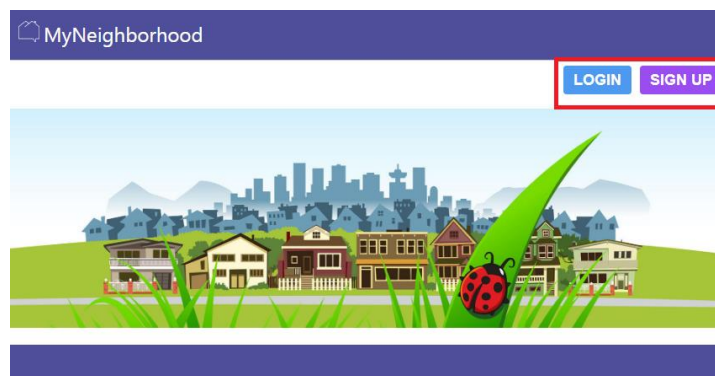
```
npm start
```

i pričekati da se frontend poslužitelj podigne na portu 3000.

## 7.6. Korisničke upute

### Registracija, prijava, profil i odjava

Slika 52 prikazuje javnu stranicu aplikacije Moj Kvart s istaknutim tipkama za prijavu ("LOGIN") i registraciju ("SIGN UP"). Daljnji pristup aplikaciji nije dostupan neregistriranim korisnicima ali se svaki zainteresirani korisnik može besplatno registrirati kao redovni stanovnik četvrti (korisničke račune vijećnika, moderatora i administratora otvaraju postojeći administratori te su odgovarajuće korisničke upute dane kasnije u ovom poglavlju).



Slika 52: Javna stranica aplikacije Moj Kvart

Kako bi se korisnik registrirao odabire tipku "SIGN UP" u gornjem desnom uglu stranice te mu se prikazuje obrazac kao na slici (Slika 53). Obrazac je potrebno ispuniti u potpunosti, redom navodeći svoje željeno korisničko ime, lozinku, ime, prezime i e-mail adresu te odabirom ulice u kojoj stanuje. Za potvrdu unosa potrebno je pritisnuti tipku "Sign up". Registracija se neće provesti te će biti ispisana odgovarajuća poruka u sljedećim slučajevima:

- korisničko ime je zauzeto
- e-mail adresa je zauzeta
- e-mail adresa je neispravna
- lozinka se ne sastoji od barem osam znakova, jednog slova i jedne brojke

Ako registracija nije uspjela potrebno je, s obzirom na dobivenu poruku, ponoviti unos s odgovarajućim izmjenama. U slučaju uspješne registracije korisnika se, nakon obavijesti, vraća na početnu stranicu.

**LOGIN** **SIGN UP** →

**Create new MyNeighborhood account**  
or [sign in to your account](#)

Username  
EddieVedder

Password  
.....

First name  
Eddie

Last name  
Vedder

Email  
ev@gmail.com

Street  
Ulica

Sign up

Slika 53: Registracija redovnog stanovnika

Za prijavu u aplikaciju, korisnik treba odabrati tipku "LOGIN" u gornjem desnom uglu stranice te mu se prikazuje obrazac za prijavu prikazan na slici (Slika 54). Obrazac za prijavu je potrebno ispuniti s vlastitim korisničkim imenom te odgovarajućom ispravnom lozinkom. Unos se potvrđuje pritiskom na tipku "Login". Ako korisničko ime i/ili lozinka nisu ispravni ispisuje se odgovarajuća poruka "Login failed". Unosom ispravnih podataka se korisnika prosljeđuje na odgovarajuću stranicu ovisno o vrsti njegovog korisničkog računa.

**LOGIN** **SIGN UP** →

**Log in to MyNeighborhood**  
or [create an account](#)

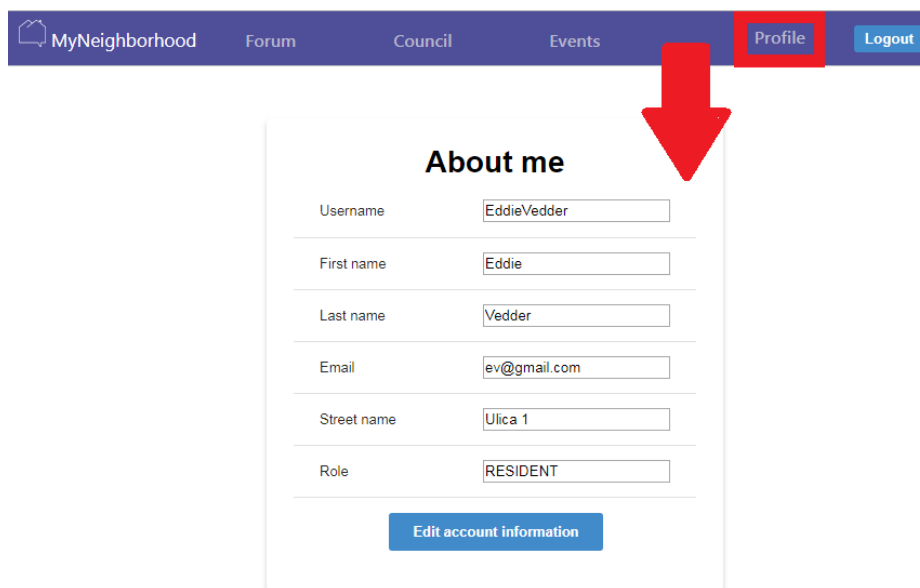
Username  
EddieVedder

Password  
.....

Login

Slika 54: Prijava u aplikaciju Moj Kvart

Svi prijavljeni korisnici mogu pregledati informacije svog profila pritiskom na tipku "Profile" u gornjem desnom kutu stranice. Prikazane su informacije korisničkog imena, imena, prezimena, e-mail adrese, ulice u kojoj stanuje te vrste korisničkog računa (redovan stanovnik, vijećnik, moderator ili administrator). Postojeće informacije se mogu izmijeniti pritiskom na tipku "Edit account information" čime se otvara obrazac za unos novih podataka. Novi unos se može potvrditi pritiskom na tipku "Change account information" ili odbaciti pritiskom na tipku "Cancel". Pri izmjeni informacija korisničkog računa vrijede ista pravila kao i prilikom registracije.



*Slika 55: Pregled informacija profila*

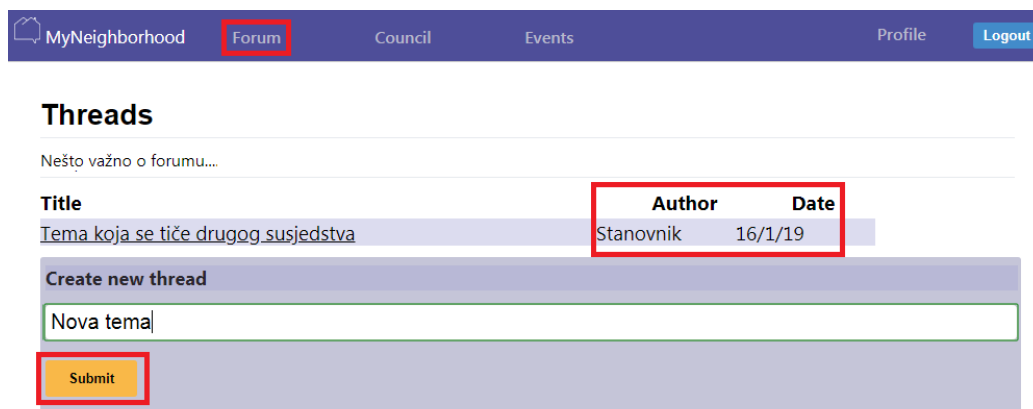
Bilo koji prijavljeni korisnik se može odjaviti u bilo kojem trenutku pritiskom na tipku "Logout" u gornjem desnom kutu stranice.

U nastavku slijede korisničke upute za postojeće korisnike, zbog većih razlika u funkcionalnostima, podijeljene prema vrstama korisnika - redovnom stanovniku, vijećniku, moderatoru i administratoru. Pored navedenih vrsta korisnika pristup aplikaciji nije dostupan neregistriranim korisnicima (osim javne stranice za prijavu i registraciju).

## Redovan stanovnik

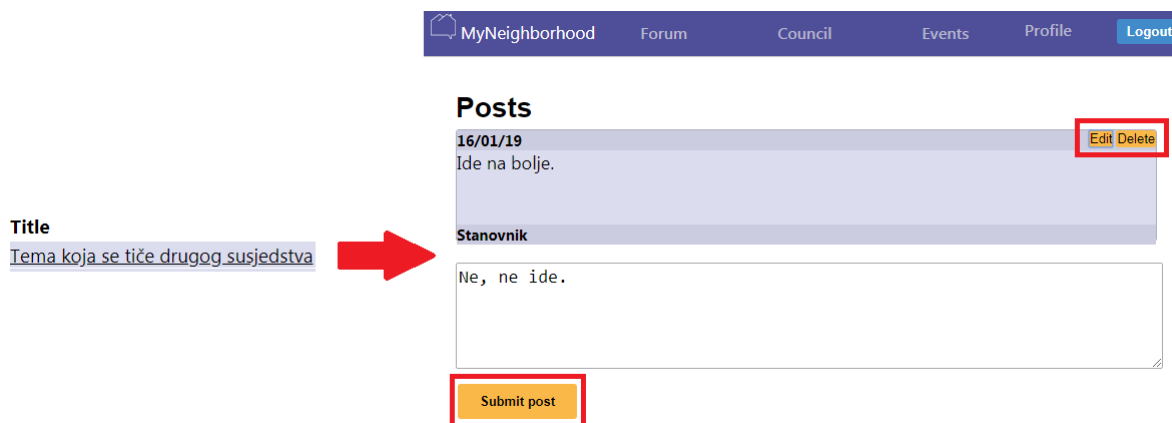
Prijavom u aplikaciju redovnom stanovniku se prikazuje stranica foruma vlastitog kvarta. Ovlasti redovnog stanovnika su sljedeće: pregledavanje i stvaranje tema i objava na forumu, pregled objava vijeća četvrti te pregled i prijedlog najava događaja. Opis izvršavanja navedenih radnji je naveden redom u nastavku.

Pregledavanje tema foruma se ostvaruje pritiskom tipke "Forum". Korisniku se prikazuju naslovi tema te njihovi autori i datumi stvaranja. Slika 56 prikazuje pregledavanje tema (s istaknutim autorom i datumom stvaranja) te otvaranje nove teme. Za otvaranje nove teme je prvo potrebno odabrati forum te zatim unijeti naslov nove teme i potvrditi unos tipkom "Submit". Nova tema će biti prikazana u popisu tema foruma.



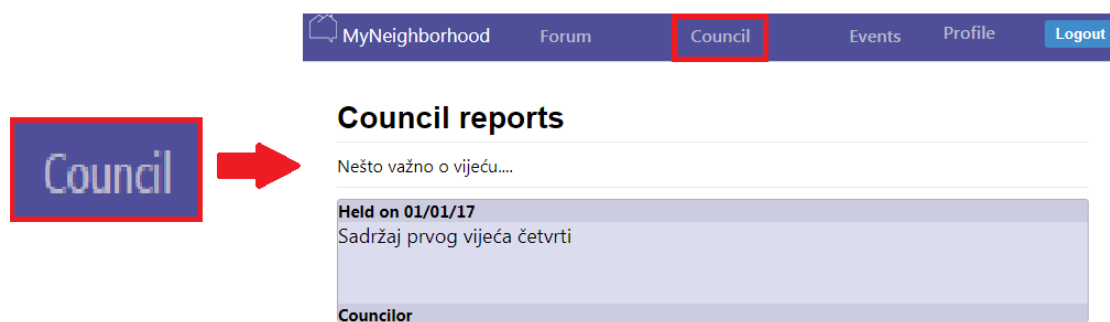
Slika 56: Pregledavanje i otvaranje tema na forumu

Pregledavanje objava foruma se ostvaruje pritiskom tipke "Forum" te zatim pritiskom na temu čije objave želimo pregledati. Objave sadrže datum objavljivanja, tekst objave te njihovog autora. Za stvaranje objave potrebno je otvoriti željenu temu te unijeti tekst nove objave. Nova objava će biti prikazana na dnu teme. Korisnik može izmijeniti i obrisati isključivo svoju objavu pritiskom na tipku "Edit" i "Delete". Slika 57 prikazuje primjer pregleda objava i objavljivanja s istaknutim tipkama za uređivanje i brisanje objave.



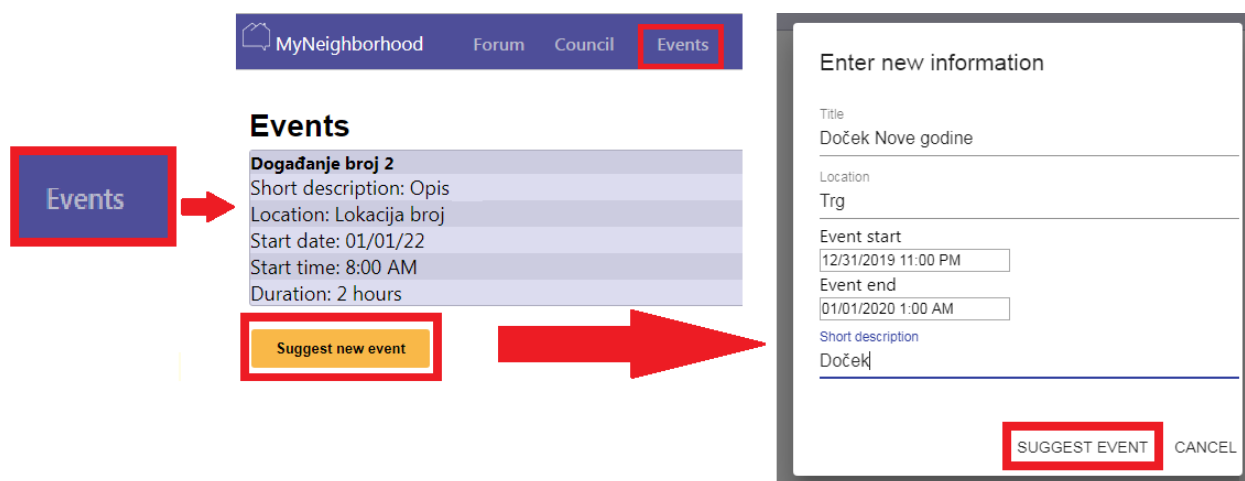
Slika 57: Pregledavanje i objavljivanje na forumu

Pregledavanje objava vijeća četvrti se ostvaruje pritiskom tipke "Council" koja korisniku prikazuje objave vijeća četvrti. Vidljiva su izvješća vijeća četvrti s podacima datuma održavanja te koji vijećnik ih je objavio. Slika 58 prikazuje primjer pregleda izvješća vijeća četvrti.



Slika 58: Pregled izvješća vijeća četvrti

Pregledavanje najave događaja se ostvaruje pritiskom na tipku "Events". Korisniku se prikazuje popis najave događaja. Svaka najava događaja sadrži svoj naziv, kratak opis, lokaciju te datum i vrijeme početka. Korisnik može i sam predložiti najavu događaja pritiskom na tipku "Suggest new event" te ispunjavanjem obrasca naslovom, lokacijom, datumom početka i završetka te kratkim opisom događanja (unos potvrđuje tipkom "Suggest event" ili odbacuje tipkom "Cancel"). Prijedlog najave događanja će biti proslijeđen moderatoru o čijoj odluci ovisi hoće li najava događaja biti objavljena. Slika 59 prikazuje pregled i prijedlog najave događaja.

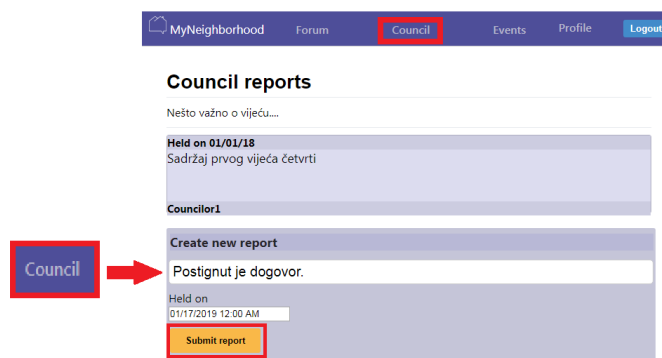


Slika 59: Pregled i prijedlog najave događaja

## Vijećnik

Vijećnik, pored svojih posebnih, ima identične ovlasti i radnje poput redovnog stanovnika te ih izvršava na identičan način. Vijećnikova posebna akcija je pisanja izvješća vijeća četvrti. Kako bi objavio takvo izvješće, vijećnik odabire tipku "Council" te unosi sadržaj izvješća i njegov datum održavanja kao što prikazuje Slika 60. Unos se potvrđuje pritiskom na tipku "Submit report".



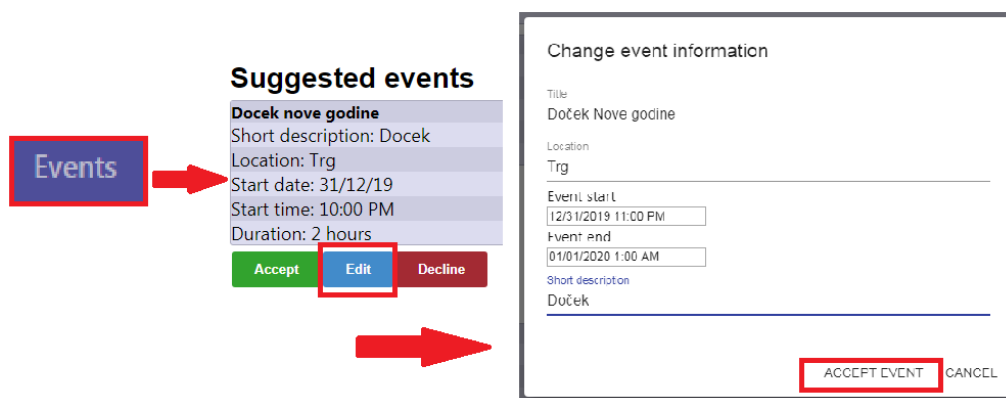


Slika 60: Objava izvješća s vijeća četvrti

## Moderator

Moderator, pored svojih posebnih, ima i identične ovlasti i radnje poput redovnog stanovnika te ih izvršava na identičan način. Moderatorove posebne akcije su rukovanje prijedlozima najava događaja te uređivanje tuđih objava i tema na forumu. Sve objave ili teme na forumu neprikladnog sadržaja moderator može ukloniti tako da uz željenu objavu pritisne tipku "Delete" ili izmijeniti pritiskom na tipku "Edit" (npr. kako bi ispravio gramatičke pogreške).

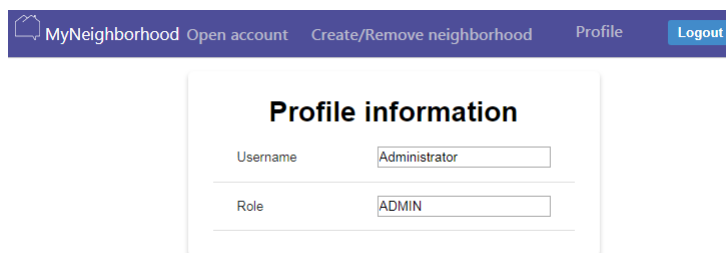
Prijedloge najave događaja moderator može pregledati pritiskom na tipku "Events" na dnu stranice. Prikazane prijedloge moderator može prihvatiti kakvi jesu pritiskom na tipku "Accept", u potpunosti odbaciti pritiskom na tipku "Decline" ili izmijeniti i prihvatiti pritiskom na tipku "Edit". Pritiskom tipke "Edit" se otvara obrazac događaja kojeg nanovo ispunjava te prihvaća pritiskom na tipku "Accept event" ili odustaje od izmjene pritiskom na tipku "Cancel". Slika 61 prikazuje primjer gdje moderator odluči urediti prijedlog najave događaja prije nego ga prihvati i objavi.



Slika 61: Primjer uređivanja i prihvaćanja najave događaja

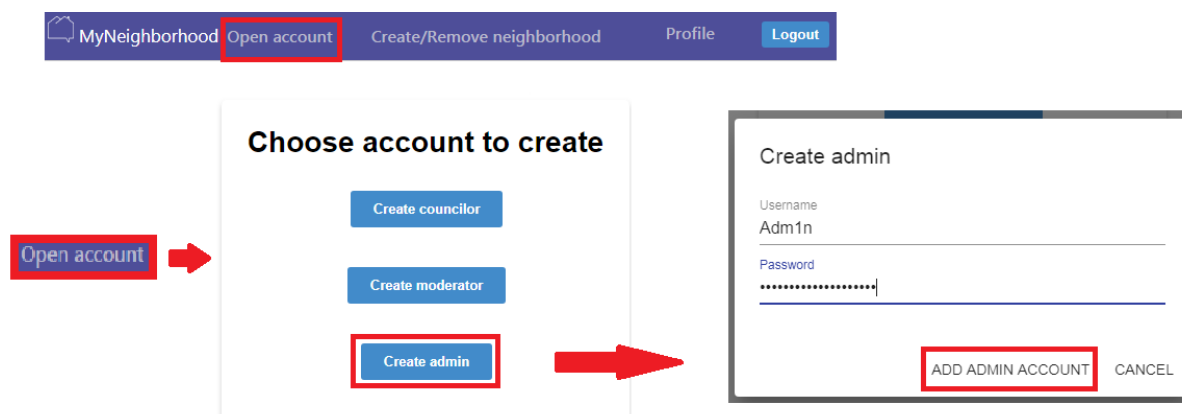
## Administrator

Administrator se kao vrsta korisničkog računa najviše razlikuje od ostalih te s njima ne dijeli funkcionalnosti. Prijavom u aplikaciju moderatoru se prikazuju informacije o njegovom korisničkom računu te ima znatno drugačije sučelje nego druge vrste korisnika (Slika 62). Administrator ima ovlasti otvaranja korisničkih računa za vijećnike, moderatore i administratore te uređivanje četvrti.



Slika 62: Sučelje za račun vrste administrator

Izrada korisničkog računa se započinje odabirom kartice "Open account" nakon koje se prikazuje odabir vrste korisničkog računa. Pritisne se željena vrsta nakon čega se prikaže obrazac za informacije koje je potrebno ispuniti. Obrazac za vijećnika i moderatora traži informacije kao i onaj pri registraciji redovnog stanovnika dok obrazac za administratoru sadrži samo korisničko ime i lozinku s obzirom na to da on nije stanovnik četvrti. Unos informacija se može odbaciti pritiskom na tipku "Cancel" ili potvrditi pritiskom na tipku "Add account". Slika 63 prikazuje primjer izrade povlaštenog korisničkog računa - u ovom slučaju administratora.



Slika 63: Otvaranje korisničkog računa za administratora

Uređivanje četvrti se vrši odabirom kartice "Create/Remove neighborhood" koja prikazuje opcije dodavanja ulice te izrade nove ili brisanje četvrti. Za dodavanje ulice potrebno je pritisnuti tipku "Create street" koja otvara obrazac u kojeg je potrebno upisati naziv ulice te odabrati kojoj četvrti će ona pripadati. Potvrda unosa se obavlja pritiskom na tipku "Add street" dok se odbacivanje obavlja pritiskom na tipku "Cancel". Dodavanje četvrti se obavlja pritiskom na tipku "Create neighborhood" te upisom imena i potvrdom unosa pritiskom tipke "Add neighborhood". Brisanje četvrti obavlja pritiskom na tipku "Remove neighborhood" te odabirom četvrti koju se želi obrisati i potvrdom odabira pritiskom na tipku "Remove neighborhood". Slika 64 prikazuje primjer dodavanja nove četvrti te zatim dodavanje nove ulicu u tu novu četvrt.

The screenshot displays the 'MyNeighborhood' web application interface. At the top, a navigation bar includes 'MyNeighborhood Open account', a highlighted 'Create/Remove neighborhood' button, a 'Profile' link, and a 'Logout' button. Below this, a 'Choose an option' dialog box presents three choices: 'Create neighborhood' (highlighted with a red box and labeled '1.'), 'Create street' (highlighted with a red box and labeled '2.'), and 'Remove neighborhood'. Red arrows indicate the flow from these options to their respective forms. The 'Create neighborhood' form (labeled '1.') contains a 'Neighborhood name' field with the text 'Kvartić' and 'ADD NEIGHBORHOOD' and 'CANCEL' buttons. The 'Create street' form (labeled '2.') features a dropdown menu for 'Neighborhood where street belongs' set to 'Kvartić', a 'Street name' field with the text 'Vartina', and 'ADD STREET' and 'CANCEL' buttons.

Slika 64: Primjer dodavanja nove četvrti i ulice

## 8. Zaključak i budući rad

Zadatak projekta bio je razvoj aplikacije koja stanovnicima gradskih četvrti omogućuje jednostavniju razmjenu informacija. Projekt je rađen u dvije faze te je nakon otprilike tri mjeseca uspješno priveden kraju.

U prvoj je fazi naglasak bio na opisivanju projektnih zahtjeva i planiranju buduće implementacije. Osim jasne specifikacije zadatka kroz dijagrame obrazaca uporabe i sekvencijskih dijagrama, izrađen je model baze te dijagrami razreda i objekata - temelj nadolazećeg rada u drugoj fazi.

Druga je faza započela razrješavanjem nedoumica oko korištenih tehnologija te formalnom podjelom posla. Konkretno, tim se podijelio u 2 ekipe. Prva je ekipa bila zadužena za razvoj backend, a druga za razvoj frontend aplikacije. Paralelno tome, kao i u prethodnoj fazi, rad je pratilo pisanje dokumentacije te je upoznat velik broj novih vrsta UML dijagrama.

Unatoč tome što se nijedan od članova tima dosad nije susreo s dobrim dijelom korištenih tehnologija, realizacija je išla začuđujuće glatko i bez prevelikih problema. U tome su nam od velike pomoći bila dodatna predavanja gostujućih predavača na predmetu te dokumentacija iz prvog dijela semestra.

Komunikacija između članova u početku se odvijala preko društvenih mreža (*Facebook Messenger*), a kasnije, kad je narasla potreba za razdvojenim kanalima i urednosti, aplikaciju *Slack*. Ipak, o najbitnijim temama raspravljalo se uživo, najčešće u prostorijama fakulteta.

Iako je rad ekipe na ovom projektu završen, postoji velik prostor za nadogradnju. To može, primjerice, biti dodavanje funkcionalnosti privatnih poruka ili čak implementacija chat klijenta. Moguća su i različita stilska unaprjeđenja, kao što su mogućnost dodavanja osobnog avatara na forumu ili multimedijskog sadržaja (slike, PDF dokumenti...) prilikom objave na forumu ili objave izvješća s vijeća četvrti.

Rad na projektu pokazao se iznimno korisnim zato što smo se, osim s novim alatima i tehnologijama, susreli s radom u timu te svim dobrim, i ponekom lošom stranom koje on donosi sa sobom. Naučili smo koliko je zapravo važna dobra organizacija i kvalitetna komunikacija između članova te smo sigurni da će nam stečeno iskustvo pomoći u budućim projektima u kojima ćemo sudjelovati.

## 9. Popis literature

- <sup>1</sup> Oblikovanje programske potpore, FER ZEMRIS, <http://www.fer.hr/predmet/opp>
- <sup>2</sup> Oblikovanje programske potpore, FER ZEMRIS,  
<http://www.zemris.fer.hr/predmeti/opp>
- <sup>3</sup> I. Sommerville, „Software engineering“, 8th ed, Addison Wesley, 2007.
- <sup>4</sup> T.C.Lethbridge, R.Langaniere, „Object-Oriented Software Engineering“, 2nd ed.  
McGraw-Hill, 2005.
- <sup>5</sup> Software engineering ,Rutgers University,  
<http://www.ece.rutgers.edu/~marsic/Teaching/SE>
- <sup>6</sup> I. Marsic, „Software engineering book“, Department of Electrical and Computer  
Engineering, Rutgers University, <http://www.ece.rutgers.edu/~marsic/books/SE>
- <sup>7</sup> Concepts: Requirements,  
[http://www.upedu.org/upedu/process/gcncpt/co\\_req.htm](http://www.upedu.org/upedu/process/gcncpt/co_req.htm)
- <sup>8</sup> UML 2 Class Diagram Guidelines,  
<http://www.agilemodeling.com/style/classDiagram.htm>
- <sup>9</sup> Domain Class Diagram Modeling Standards and Guidelines,  
<http://www.bced.gov.bc.ca/imb/downloads/classdiagramstandards.pdf>
- <sup>10</sup> Astah Community, <http://astah.net/editions/community/>

## Dodatak A: Indeks (slika, dijagrama, tablica, ispisa kôda)

### Slike:

Slika 1: Vršni dijagram obrazaca uporabe .....	15
Slika 2: Dijagram obrazaca uporabe - Rad na forumu .....	16
Slika 3: Dijagram obrazaca uporabe - cjelina "Događanja" .....	17
Slika 4: Dijagram obrazaca uporabe - cjelina "Vijeće četvrti" .....	17
Slika 5: Dijagram obrazaca uporabe - Rad sa četvrtima .....	18
Slika 6: UC2: Otvaranje povlaštenih korisničkih računa .....	19
Slika 7: UC5 i UC8: Objavljivanje na forumu i Pregled objava na forumu .....	20
Slika 8: UC9: Stvaranje teme na forumu .....	21
Slika 9: UC11: Uklanjanje objava s foruma .....	21
Slika 10: UC12 i UC13: Prijedlog najave događaja i Objavljivanje u cjelini "Događanja" .....	22
Slika 11: UC18 i UC19: Uređivanje četvrti i Brisanje četvrti .....	23
Slika 12: MVC obrazac .....	26
Slika 13: Skica baze podataka .....	27
Slika 14: Dijagram razreda .....	30
Slika 15: Dijagram objekata .....	31
Slika 16: Dijagram komunikacije otvaranja korisničkog povlaštenog računa i prijave na isti .....	32
Slika 17: Dijagram komunikacije pregledavanja objava i objavljivanja na forumu .....	33
Slika 18: Dijagram komunikacije predlaganja i prihvatanja događaja uz izmjene .....	33
Slika 19: Dijagram stanja za neprijavljenog korisnika .....	34
Slika 20: Dijagram stanja za prijavljenog korisnika .....	35
Slika 21: Dijagram stanja za vijećnika .....	35
Slika 22: Dijagram stanja za moderatora .....	36
Slika 23: Dijagram stanja za administratora .....	37
Slika 24: Dijagram aktivnosti otvaranja povlaštenog korisničkog računa .....	38
Slika 25: Dijagram aktivnosti objavljivanja na forumu .....	39
Slika 26: Dijagram aktivnosti predlaganja i objavljivanja najave događaja .....	40
Slika 27: Dijagram komponenti 1 .....	41
Slika 28: Dijagram komponenti 2 .....	41
Slika 29: Dijagram razmještaja .....	42
Slika 30: Primjer tablice u bazi podataka .....	44
Slika 31: Primjer Controllera .....	45
Slika 32: Primjer DTO-a .....	45
Slika 33: Primjer prikaza mrežnog prometa pri prijavi stanovnika četvrti u pregledniku Firefox .....	46
Slika 34: Ispis pogreške u pregledniku .....	47
Slika 35: Pokušaj registracije korisnika s korisničkim imenom koje je zauzeto .....	47
Slika 36: Primjer uspješne registracije .....	48
Slika 37: Prikaz testiranja objavljivanja na forumu .....	49
Slika 38: Prikaz što "Stanovnik1" vidi na forumu nakon objave korisnika "david123" .....	49
Slika 39: Događaji koji čekaju na odobravanje su vidljivi samo moderatoru .....	50
Slika 40: Odobrenje predloženog događaja 1 .....	51
Slika 41: Prikaz korisničkom sučelja nakon odbijanja predloženog događaja 2 .....	52
Slika 42: Izgled korisničkog sučelja za stvaranje četvrti i ulica te brisanje četvrti .....	52
Slika 43: Prozor koji se otvara klikom na gumb "Create neighborhood" .....	53

Slika 44: Rezultat za unos imena koje već postoji u bazi podataka.....	53
Slika 45:Uspješno dodavanje nove četvrti u bazu podataka .....	54
Slika 46: Prikaz Oracle web stranice s koje se skida JDK.....	55
Slika 47:Prikaz varijabli okruženja 1 .....	55
Slika 48: Prikaz varijabli okruženja 2.....	56
Slika 49: Ispis u command promptu .....	56
Slika 50: Prikaz web stranice za skidanje Maven paketa .....	57
Slika 51: Prikaz razreda DatabaseConfig.....	58
Slika 52: Javna stranica aplikacije Moj Kvart .....	60
Slika 53: Registracija redovnog stanovnika.....	61
Slika 54: Prijava u aplikaciju Moj Kvart .....	61
Slika 55: Pregled informacija profila .....	62
Slika 56: Pregledavanje i otvaranje tema na forumu .....	63
Slika 57: Pregledavanje i objavljivanje na forumu .....	63
Slika 58: Pregled izvješća vijeća četvrti .....	64
Slika 59: Pregled i prijedlog najave događaja .....	64
Slika 60: Objava izvješća s vijeća četvrti .....	65
Slika 61: Primjer uređivanja i prihvatanja najave događaja .....	65
Slika 62: Sučelje za račun vrste administrator .....	66
Slika 63: Otvaranje korisničkog računa za administratora .....	66
Slika 64: Primjer dodavanja nove četvrti i ulice .....	67

## Dodatak B: Dnevnik sastajanja

#	Datum	Prisutni	Sadržaj
01	19. listopada 2018.	<ul style="list-style-type: none"> <li>• Mate Gašparini</li> <li>• Nicolas Assouline</li> <li>• Dana Dodigović</li> <li>• David Dukić</li> <li>• Marko Kršić</li> <li>• Marino Kurtović</li> </ul>	<ul style="list-style-type: none"> <li>• Podjela poslova - opis projekta i funkcionalni zahtjevi</li> <li>• Rasprava o odabiru tehnologija</li> </ul>
02	23. listopada 2018.	<ul style="list-style-type: none"> <li>• Mate Gašparini</li> <li>• Nicolas Assouline</li> <li>• Dana Dodigović</li> <li>• David Dukić</li> <li>• Marko Kršić</li> <li>• Marino Kurtović</li> </ul>	<ul style="list-style-type: none"> <li>• Provjera točnosti funkcionalnih zahtjeva</li> <li>• Podjela poslova - dijagrami obrazaca uporabe i sekvencijski dijagrami</li> </ul>
03	30. listopada 2018.	<ul style="list-style-type: none"> <li>• Mate Gašparini</li> <li>• Marko Kršić</li> <li>• Marino Kurtović</li> </ul>	<ul style="list-style-type: none"> <li>• Provjera točnosti napravljenih dijagrama i eventualni ispravci</li> <li>• Podjela poslova - nefunkcionalni zahtjevi</li> </ul>
04	21. studenoga 2018.	<ul style="list-style-type: none"> <li>• Mate Gašparini</li> <li>• Nicolas Assouline</li> <li>• Dana Dodigović</li> <li>• David Dukić</li> <li>• Marko Kršić</li> <li>• Marino Kurtović</li> </ul>	<ul style="list-style-type: none"> <li>• Općenita rasprava o nadolazećoj implementaciji</li> <li>• Podjela poslova - pismeni opis baze, dijagram razreda</li> </ul>
05	12. prosinca 2018.	<ul style="list-style-type: none"> <li>• Mate Gašparini</li> <li>• Nicolas Assouline</li> <li>• David Dukić</li> <li>• Marko Kršić</li> <li>• Marino Kurtović</li> </ul>	<ul style="list-style-type: none"> <li>• Pokazan postupak importiranja stvorenih projekata u IDE</li> <li>• Podjela poslova - backend, frontend</li> </ul>
06	8. siječnja 2019.	<ul style="list-style-type: none"> <li>• Mate Gašparini</li> <li>• Nicolas Assouline</li> <li>• Dana Dodigović</li> <li>• David Dukić</li> <li>• Marko Kršić</li> <li>• Marino Kurtović</li> </ul>	<ul style="list-style-type: none"> <li>• Kontrola dosadašnje implementacije</li> <li>• Podjela poslova - dokumentacija revizije 2</li> </ul>

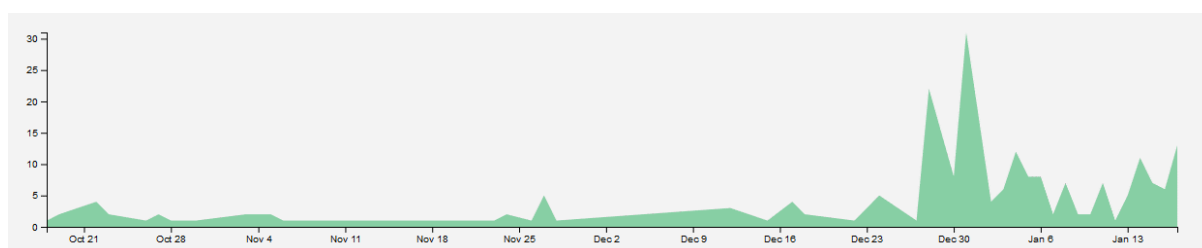


## Dodatak C: Prikaz aktivnosti grupe

Popis aktivnosti	Mate Gašparini	Nicolas Assouline	Dana Dodigović	David Dukić	Marko Kršić	Marino Kurtović
Upravljanje projektom	100%					
Opis projektnog zadatka			100%			
Rječnik pojmova				50%	50%	
Opis funkcionalnih zahtjeva		25%		25%	25%	25%
Opis ostalih zahtjeva	100%					
Arhitektura i dizajn sustava						
Svrha, opći prioriteti i skica sustava	20%		80%			
Dijagram razreda s opisom	50%		50%			
Dijagram objekata					100%	
Ostali UML dijagrami		25%			50%	25%
Implementacija i korisničko sučelje						
Dijagram razmještaja						100%
Korištene tehnologije i alati	100%					
Isječak programskog kôda		100%				
Ispitivanje programskog rješenja				100%		
Upute za instalaciju	50%	50%				
Korisničke upute					100%	
Plan rada	100%					
Pregled rada i	100%					

<b>stanje ostvarenja</b>						
<b>Zaključak i budući rad</b>	33%			33%		33%
<b>Popis literature</b>	16%	16%	16%	16%	16%	16%
<b>Dodaci</b>						
Indeks		100%				
Dnevnik sastajanja	100%					

Pregled pohrana kroz vrijeme trajanja projekta:



## Dodatak D: Pregled rada i stanje ostvarenja

Projekt je u cjelosti implementiran te su ispunjene sve inicijalno planirane funkcionalnosti. Rad se u drugoj fazi sastojao od izrade i testiranja svih mogućnosti web-aplikacije te se, kao i u prvoj fazi projekta, vodila detaljna dokumentacija.

Iako je formalno rad na ovom projektu završen, ima prostora za nadogradnju, počevši od beskrajnih mogućnosti za promjenu stila i izgleda aplikacije te prilagođavanju za širi spektar veličina zaslona. Osim toga, mogu se implementirati potpuno nove funkcionalnosti kao što su privatne poruke i/ili chat. Na forumu bi se moglo dopustiti postavljanje personaliziranih avatara za slike profila ili uvesti sustav ocjenjivanja komentara.

Dodatno, postoji mogućnost da se dopusti prijenos multimedijskog sadržaja (slike, PDF dokumenti...) prilikom objave na forumu ili objave izvješća s vijeća četvrti. Slično, nakon što je neki od aktivnih događaja u cjelini događaja završio, moglo bi se objaviti njegovo kratko izvješće, primjerice s fotografijama i videozapisima snimljenima na lokaciji prilikom tog događaja.