

# Oblikovanje programske potpore

Ak. god. 2019./2020.

## Manje smeće, više sreće

Dokumentacija, Rev. 2.0

Grupa: *Kombinacija*

Voditelj: *Sven Skender*

Datum predaje: *15. siječnja, 2020.*

Nastavnik: *Tomislav Jukić*

# Sadržaj

<b>1</b>	<b>Dnevnik promjena dokumentacije</b>	<b>3</b>
<b>2</b>	<b>Opis projektnog zadatka</b>	<b>5</b>
<b>3</b>	<b>Specifikacija programske potpore</b>	<b>8</b>
3.1	Funkcionalni zahtjevi . . . . .	8
3.1.1	Obrasci uporabe . . . . .	10
3.1.2	Sekvencijski dijagrami . . . . .	20
3.2	Ostali zahtjevi . . . . .	23
<b>4</b>	<b>Arhitektura i dizajn sustava</b>	<b>24</b>
4.1	Arhitektura sustava . . . . .	24
4.2	Arhitektura aplikacije . . . . .	25
4.3	Baza podataka . . . . .	26
4.3.1	Opis tablica . . . . .	27
4.3.2	Definicije tablica . . . . .	27
4.3.3	Dijagram baze podataka . . . . .	31
4.4	Dijagram razreda . . . . .	32
4.5	Dijagram stanja . . . . .	36
4.6	Dijagram aktivnosti . . . . .	37
4.7	Dijagram komponenti . . . . .	38
<b>5</b>	<b>Implementacija i korisničko sučelje</b>	<b>40</b>
5.1	Korištene tehnologije i alati . . . . .	40
5.2	Ispitivanje programskog rješenja . . . . .	42
5.2.1	Ispitivanje komponenti . . . . .	42
5.2.2	Ispitivanje sustava . . . . .	45
5.3	Dijagram razmještaja . . . . .	50
5.4	Upute za puštanje u pogon . . . . .	51
<b>6</b>	<b>Zaključak i budući rad</b>	<b>57</b>

<b>Popis literature</b>	<b>59</b>
<b>Indeks slika i dijagrama</b>	<b>61</b>
<b>Dodatak: Prikaz aktivnosti grupe</b>	<b>62</b>

# 1. Dnevnik promjena dokumentacije

Rev.	Opis promjene/dodatka	Autori	Datum
0.1.0	Napravljen predložak, ispunjene osnovne informacije i početak pisanja opisa projekta	Bićanić	24.10.2019.
0.2	Nastavak pisanja opisa projekta	Bićanić	27.10.2019.
0.3	Završena prva verzija opisa projekta	Bićanić	28.10.2019.
0.4	Započeto pisanje UC dijagrama	Vasilj	28.10.2019.
0.5	Završeno pisanje UC dijagrama	Vasilj	29.10.2019.
0.6	Započeto pisanje Opisa Baze Podataka	Bićanić	1.11.2019.
0.7	Završeno pisanje Opisa Baze Podataka	Bićanić	3.11.2019.
0.8	Dodan dijagram i ER model baze podataka	Bićanić	3.11.2019.
0.9	Započeto raspisivanje Use Caseova	Vasilj	29.10.2019.
0.10	Završeno raspisivanje Use Caseova	Vasilj	4.11.2019.
0.11	Raspisani ostali zahtjevi projekta	Vasilj	4.11.2019.
0.12	Dodani sekvencijski dijagrami i opisi istih	Vasilj	5.11.2019.
0.13	Napisano poglavlje o arhitekturi sustava	Bićanić	10.11.2019.
0.14	Dodan detaljniji opis arhitekture sustava i aplikacije, ispravljene greške uočene na lab. vježbi	Bićanić, Vasilj	13.11.2019.
1.0	Korigiranje teksta i provjera dokumentacije	Bićanić, Vasilj	13.11.2019.
1.1	Dodani dijagram stanja i dijagram aktivnosti	Vasilj	02.01.2020.
1.2	Dodan opis tehnologija i alata	Vasilj	03.01.2020.
1.3	Dodane upute za puštanje u pogon	Skender, Vasilj	03.01.2020.
1.4	Dodane dijagram razmještaja	Vasilj	03.01.2020.
1.5	Dodan zaključak	Vasilj	03.01.2020.
1.6	Ažurirane informacije o tehnologijama i alatima	Bićanić	06.01.2020.
1.7	Ažurirano poglavlje Dijagrami Razreda	Bićanić	06.01.2020.

Rev.	Opis promjene/dodatka	Autori	Datum
1.8	Dodani testovi sustava	Skender, Vasilj	15.01.2020.
1.9	Ažuriran dnevnik sastajanja	Vasilj	15.01.2020.
1.10	Završni pregled i prepravke dokumentacije	Vasilj	15.01.2020.
2.0	Konačna verzija		15.01.2020.

Tablica 1.1: Popis promjena dokumentacije

## 2. Opis projektnog zadatka

Svakodnevno se susrećemo sa neispravno odloženim otpadom, nerijetko upravo zato što su kontejneri predviđeni za njega puni ili čak pretrpani. Jedna od posljedica svega toga je i da recikliranje postaje znatno otežano.

Ovim projektom želimo riješiti taj problem tako što bismo razvili web aplikaciju za cjelokupan sustav odvoza otpada. Koristeći se njom, građani bi na karti mogli odabrati neki kontejner i označiti ga praznim, punim ili pretrpanim, a aplikacija bi bila integrirana sa postojećim komunalnim službama koje bi tim putem dobivale obavijesti o punim kontejnerima, što bi olakšalo planiranje rute i fokusiralo pražnjenje kontejnera na mjesta na kojima je stvarno potrebno.

Aplikacija prilikom pokretanja nudi izbornik kao i polje za unos datuma, te prikazuje kartu područja u okolini klijentovog uređaja na kojoj su označeni kontejneri. Putem izbornika se neregistrirani korisnici mogu registrirati, a neprijavljeni prijaviti. Aplikacija razlikuje tri vrste prijavljenih korisnika (poredano po razini ovlasti, od najniže prema najvišoj):

- *Građanin*
- *Komunalni radnik*
- *Administrator*

Koristeći se oznakama na karti, svi korisnici, uključujući i neregistrirane, mogu za svaki kontejner vidjeti informacije o prijavljivanju i pražnjenju kontejnera, i to za onaj dan koji je upisan u polju za unos datuma.

Građaninu je za registraciju dovoljna ispravna e-mail adresa i lozinka, pri čemu ne mogu postojati dva korisnika (bilo koje razine ovlasti) sa istom e-mail adresom. Nakon uspješne prijave i u slučaju da dan upisan u polju datuma odgovara današnjem, građaninu se pritiskom na neki kontejner na karti otvara izbornik nad tim kontejnerom putem kojega korisnik može:

- kontejner prijaviti kao pun
- kontejner prijaviti kao pretrpan

- kontejner prijaviti kao prazan, samo ako je taj kontejner već (lažno) označen kao pretrpan

Prilikom prijave kontejnera građanin uz prijavu može priložiti i fotografiju toga kontejnera kao dokaz stanja.

Ako upisani datum ne odgovara današnjem, onda je omogućen samo pregled svih prijava i pražnjenja kontejnera za taj dan. Bez obzira na datum u polju za unos datuma, korisnik može bilo koji kontejner spremi u osobni popis kontejnera kako bi imao brži pristup kontejnerima u koje češće odlaže otpad (kao što su, primjerice, oni u blizini stana).

Korisnik u ulozi komunalnog radnika ima sve ovlasti koje ima i građanin, izuzev činjenice da se on ne može registrirati sam, već tu akciju obavlja isključivo korisnik u ulozi administratora. Za komunalnog redara sustav dodatno pamti ime, prezime i Osobni Identifikacijski Broj (OIB). Svaki komunalni radnik je dodjeljen jednom području grada (ili općenito mjesta u kojem se aplikacija koristi), te za sve kontejnere toga područja ima dodatne ovlasti:

- može isplanirati rutu odvoza otpada (putem izbornika)
- može **svaki** kontejner označiti kao prazan
- može označiti kontejner kao lažno prijavljen

Ovlasti administratora sustava uz sve dosad navedene omogućuju i dodatne funkcionalnosti, a njegov korisnički račun se stvara ručno. Uloga administratora je:

- raspodjela radnika po područjima mjesta na kojem se aplikacija koristi
- premještanje radnika, ovisno o trenutnim uvjetima
- dodavanje radnika (stvaranje njegovog korisničkog računa)
- brisanje radnika i pripadajućeg korisničkog računa
- dodavanje i brisanje kontejnera

S obzirom da se u aplikaciju može prijaviti svatko, postoji izvjesna mogućnost zlouporabe u obliku lažnog označavanja kontejnera kao punog/pretrpanog. Zbog toga aplikacija interno za svakog građanina pamti reputaciju koja se gradi na temelju svih prijava nekog korisnika. Ukoliko komunalni radnik u svojem obilasku kontejnera naiđe na kontejner koji je prijavljen kao pun, onog trenutka kada ga

komunalni radnik isprazni i označi ispražnjenim, automatski se svim korisnicima koji su ga prijavili reputacija diže. Shodno tome, ukoliko naiđe na prazan kontejner koji je prijavljen kao pun, komunalni radnik prijavljuje lažnu prijavu te se svim korisnicima koji su taj kontejner prijavili reputacija smanjuje.

Prijava kontejnera od strane građana ima manju vrijednost nego prijava od strane komunalnog radnika ili administratora, uz obrazloženje da ljudi vjerojatno neće sabotirati sustav u koji su sami uključeni i u kojem rade. Vrijednost prijave građana proporcionalna je njegovoj reputaciji, ali nikad nije veća od vrijednosti prijave radnika ili administratora.

Da bi komunalne službe dobile konkretnu obavijest o punom ili pretrpanom kontejneru, potrebno je prikupiti više građanskih prijava. Kada se službi pošalje obavijest o tome da je kontejner u nekom stanju, korisnici i dalje mogu prijavljivati taj kontejner, ali se obavijesti više neće generirati. Iznimka ovom pravilu je prijavljivanje suprotnog stanja: ukoliko je kontejner lažno prijavljen kao pun ili pretrpan, i već je poslana obavijest službama, moguće je da više korisnika označi taj kontejner kao prazan te se tim putem generira protu-obavijest koja službe obavještava da je kontejner prazan.

Nakon što je neki korisnik prijavio kontejner kao pun, ne može ga više označiti kao takvog sve dok ga komunalne službe ili drugi građani ne označe praznim. Jednako tako, ako ga korisnik označi praznim, ne može to ponoviti sve dok ga dovoljno ljudi ne označi punim i generira se obavijest prema službama.



## 3. Specifikacija programske potpore

### 3.1 Funkcionalni zahtjevi

Dionici:

1. Građanin
2. Komunalni radnik
3. Administrator
4. Vlasnik (naručitelj)
5. Razvojni tim

**Aktori i njihovi funkcionalni zahtjevi:**

1. Neregistrirani/ neprijavljeni korisnik može:
  - (a) za odabrani datum na karti pregledati kontejnere
  - (b) odabrati pojedini kontejner
    - i. pregledati popis prijava i pražnjena u tom danu
    - ii. pregledati grafički prikaz prijava i pražnjenja u tom danu
  - (c) registrirati se u sustav
2. Administrator može:
  - (a) prijaviti se u sustav
  - (b) dodati novo kvartovsko poduzeće
  - (c) izbrisati postojeće kvartovsko poduzeće
  - (d) dodati novi kontejner
  - (e) izbrisati postojeći kontejner
  - (f) upravljati komunalnim radnicima
    - i. dodati novog komunalnog radnika u sustav
    - ii. pridodati novog komunalnog radnika kvartu
    - iii. premjestiti komunalnog radnika u drugi kvart
    - iv. izbrisati komunalnog radnika iz sustava
  - (g) prijaviti stanje kontejnera te priložiti sliku istoga

- i. prijaviti da je kontejner prazan
- ii. prijaviti da je kontejner pun
- iii. prijaviti da je kontejner pretrpan

3. Građanin može:

- (a) prijaviti se u sustav
- (b) prijaviti stanje kontejnera te priložiti sliku istoga
  - i. prijaviti da je kontejner prazan
  - ii. prijaviti da je kontejner pun
  - iii. prijaviti da je kontejner pretrpan

4. Komunalni radnik može:

- (a) prijaviti se u sustav
- (b) preuzeti rutu za pražnjenje kontejnera
- (c) za svaki kontejner u ruti
  - i. prijaviti da je ispražnjen
  - ii. prijaviti lažnu dojavu ukoliko kontejner naveden u ruti nije pun
- (d) prijaviti stanje kontejnera te priložiti sliku istoga
  - i. prijaviti da je kontejner prazan
  - ii. prijaviti da je kontejner pun
  - iii. prijaviti da je kontejner pretrpan

5. Baza podataka (sudionik):

- (a) pohranjuje sve podatke o korisnicima i njihovim ovlastima
- (b) pohranjuje sve podatke o kvartovima, kontejnerima i prijavama

### 3.1.1 Obrasci uporabe

#### UC01-Registracija

- **Glavni sudionik:** Neregistrirani korisnik
- **Cilj:** Stvaranje korisničkog računa za pristup sustavu
- **Sudionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
  1. Korisnik odabire registraciju u sustav
  2. Korisnik unosi ime, prezime, email i lozinku
  3. Korisnik dobiva obavijest o uspješnoj registraciji u sustav
- **Opis mogućih odstupanja:**
  - 2.a Email je prethodno već zauzet
    1. Sustav obavještava korisnika da unese mail koji ne postoji u sustavu
    2. Korisnik ponovno unosi podatke ili odustaje od registracije
  - 2.b Unos neispravnog emaila
    1. Sustav obavještava korisnika da je unio neispravan podatak
    2. Korisnik ponovno unosi podatke ili odustaje od registracije
  - 2.c Korisnik je unio manje od osam znakova za lozinku
    1. Sustav obavještava korisnika da je unio neispravan podatak
    2. Korisnik ponovno unosi podatke ili odustaje od registracije

#### UC02-Prijava u sustav

- **Glavni sudionik:** Građanin, komunalni radnik, administrator
- **Cilj:** Mogućnost pristupa korisničkom sučelju
- **Sudionici:** Baza podataka
- **Preduvjet:** Registracija
- **Opis osnovnog tijeka:**
  1. Korisnik unosi korisničko email i lozinku
  2. Sustav vraća poruku o ispravnosti unesenih podataka
  3. Sustav korisniku omogućava pristup korisničkim funkcijama
- **Opis mogućih odstupanja:**
  - 2.a Unos neispravnog emaila ili lozinke
    1. Sustav obavještava korisnika da je unio neispravan podatak i vraća korisnika u korak 1.

#### UC03-Dodaj najkorištenije kontejnere

- **Glavni sudionik:** Građanin, komunalni radnik, administrator
- **Cilj:** Brži pristup kontejnerima koji su predmet česte uporabe
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav
- **Opis osnovnog tijeka:**
  1. Korisnik odabire traženi kontejner na karti
  2. Sustav prikazuje opcije vezane za odabrani kontejner
  3. Korisnik odabire i označava opciju "omiljeni kontejner"
  4. Sustav sprema unesenu promjenu

#### UC04-Ukloni iz najkorištenijih

- **Glavni sudionik:** Građanin, komunalni radnik, administrator
- **Cilj:** Povećanje preglednosti karte i lakša uporaba iste
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav
- **Opis osnovnog tijeka:**
  1. Korisnik odabire traženu kantu na karti
  2. Sustav prikazuje opcije vezane za odabrani kontejner
  3. Korisnik odabire i označava opciju "ukloni iz omiljenih kontejnera"
  4. Sustav sprema unesenu promjenu

#### UC05-Preuzimanje rute

- **Glavni sudionik:** Komunalni radnik
- **Cilj:** Preuzimanje optimalne putanje za pražnjenje kanti
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav
- **Opis osnovnog tijeka:**
  1. Korisnik odabire preuzimanje rute
  2. Sustav prikazuje popis kontejnera za pražnjenje
  3. Korisnik pokreće rutu
- **Opis mogućih odstupanja:**
  - 1.a Nedostatan broj kontejnera spremnih za pražnjenje
    1. Sustav javlja da nema dovoljan broj kontejnera za stvoriti rutu i vraća korisnika u korak 1

#### UC06-Potvrda pražnjenja

- **Glavni sudionik:** Komunalni radnik
- **Cilj:** Potvrda pražnjenja kanti s rute
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen i pokrenuo je rutu (UC05)
- **Opis osnovnog tijeka:**
  1. Korisnik odabire kontejner iz rute
  2. Sustav prikazuje opcije vezane za odabrani kontejner
  3. Korisnik označava da je kontejner ispražnjen
  4. Sustav sprema unesenu promjenu
- **Opis mogućih odstupanja:**
  - 3.a kontejner nije pun
    1. Korisnik označava da je dojava lažna
    2. Sustav prikazuje sljedeći kontejner iz rute

#### UC07-Dodavanje komunalnog radnika

- **Glavni sudionik:** Administrator
- **Cilj:** Stvaranje korisničkog računa za novog korisnika
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav
- **Opis osnovnog tijeka:**
  1. Korisnik odabire kvart
  2. Sustav prikazuje popis radnika u odabranom kvartu
  3. Korisnik odabire stvaranje novog radnika u odabranom kvartu
  4. Sustav otvara formu za registraciju novog radnika
  5. Korisnik unosi podatke o novom radniku
  6. Sustav sprema unesenu promjenu
- **Opis mogućih odstupanja:**
  - 3.a U odabranom kvartu ima previše radnika
    1. Sustav obavještava korisnika da nije moguće unijeti novog radnika i vraća ga u korak 2
  5. a Korisnik je unio neispravne podatke o novom radniku
    1. Sustav upozorava na pogrešku pri unosu i vraća korisnika u korak 4
  5. a Korisnik je pokušao stvoriti novog radnika koji već postoji u sustavu
    1. Sustav javlja kako radnik već postoji i vraća korisnika u korak 2

#### UC08-Premještanje komunalnog radnika

- **Glavni sudionik:** Administrator
- **Cilj:** Premještanje komunalnog radnika iz jedne kvartovske podružnice u drugu
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav
- **Opis osnovnog tijeka:**
  1. Korisnik odabire kvart u kojem se trenutno nalazi traženi radnik
  2. Sustav prikazuje popis radnika u odabranom kvartu
  3. Korisnik odabire traženog radnika
  4. Sustav prikazuje podatke o odabranom radniku
  5. Korisnik mijenja kvart kojem je radnik pridjeljen
  6. Sustav sprema unesenu promjenu
- **Opis mogućih odstupanja:**
  - 5.a Prevelik broj radnika u ciljanom poduzeću
    1. Sustav obavještava korisnika o nemogućnosti izvršavanja tražene akcije i vraća korisnika u korak 4
  - 5.b Premalo radnika u početnom kvartu
    1. Sustav obavještava korisnika o nemogućnosti izvršavanja tražene akcije i vraća korisnika u korak 4

#### UC09-Prijava praznog kontejnera

- **Glavni sudionik:** Administrator, komunalni radnik, građanin
- **Cilj:** Dojaviti sustavu da kontejneru nije potrebno pražnjenje
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav
- **Opis osnovnog tijeka:**
  1. Korisnik odabire kontejner na karti
  2. Korisnik označava da je kontejner prazan
  3. Sustav sprema unesenu promjenu
- **Opis mogućih odstupanja:**
  - 2.a Korisnik ima prevelik broj lažnih dojava
    1. Sustav ne sprema unesenu promjenu

#### UC10-Prijava punog kontejnera

- **Glavni sudionik:** Administrator, komunalni radnik, građanin
- **Cilj:** Dojaviti sustavu da je kontejneru potrebno pražnjenje

- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav
- **Opis osnovnog tijeka:**
  1. Korisnik odabire kontejner na karti
  2. Korisnik označava da je kontejner pun
  3. Sustav sprema unesenu promjenu
- **Opis mogućih odstupanja:**
  - 2.a Korisnik ima prevelik broj lažnih dojava
    1. Sustav ne sprema unsenu promjenu

#### UC11-Prijava prepunog kontejnera

- **Glavni sudionik:** Administrator, komunalni radnik, građanin
- **Cilj:** Dojaviti sustavu da je kontejneru hitno potrebno pražnjenje
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav
- **Opis osnovnog tijeka:**
  1. Korisnik odabire kontejner na karti
  2. Korisnik označava da je kontejner prepun
  3. Sustav sprema unesenu promjenu
- **Opis mogućih odstupanja:**
  - 2.a Korisnik ima prevelik broj lažnih dojava
    1. Sustav ne sprema unsenu promjenu

#### UC12-Fotografiranje kontejnera

- **Glavni sudionik:** Administrator, komunalni radnik, građanin
- **Cilj:** Priložiti sliku prijavi stanja kontejnera
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav i prijavio je stanje kontejnera (UC09 - UC11)
- **Opis osnovnog tijeka:**
  1. Korisnik slika kontejner i šalje sliku
  2. Sustav sprema unesenu promjenu

#### UC13-Priegled arhive

- **Glavni sudionik:** Anonimni korisnik
- **Cilj:** Pregled dosadašnjih prijava i pražnjenja kontejnera

- **Sudionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
  1. Korisnik odabire datum
  2. Sustav prikazuje korisniku kartu sa kontejnerima za navedeni datum
  3. Korisnik odabire pojedini kontejner
  4. Sustav prikazuje popis prijava i pražnjenje tog kontejnera narednog dana
- **Opis mogućih odstupanja:**
  - 1.a Unos datuma za kojeg stanje nije poznato
    1. Sustav obavještava korisnika da nije moguće dohvatiti traženo stanje
    2. Sustav vraća korisnika u korak 1

#### UC14-Dodavanje kvarta

- **Glavni sudionik:** Administrator
- **Cilj:** Stvaranje nove kvartovske podružnice
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav
- **Opis osnovnog tijeka:**
  1. Korisnik odabire popis kvartova
  2. Korisnik odabire stvaranje novog kvarta
  3. Sustav otvara formu za stvaranje novog kvarta
  4. Korisnik ispunjava formu s podacima o kvartu
  5. Sustav sprema unesenu promjenu
- **Opis mogućih odstupanja:**
  - 2.a Kvart već postoji
    1. Sustav obavještava korisnika da je unio neispravan podatak
    2. Korisnik ponovno unosi podatke ili odustaje od stvaranja novog kvarta

#### UC15-Brisanje kvarta

- **Glavni sudionik:** Administrator
- **Cilj:** Brisanje kvartovske podružnice
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav
- **Opis osnovnog tijeka:**
  1. Korisnik odabire popis kvartova
  2. Korisnik odabire kvart za brisanje



3. Korisnik uklanjanja odabrani kvart
4. Sustav sprema unesenu promjenu
- **Opis mogućih odstupanja:**
  - 3.a Postoje djelatnici pridjeljeni odabranom kvartu
    1. Sustav dojavljuje kako nije moguće obrisati kvartovsko poduzeće dok su njemu pridjeljeni radnici i vraća korisnika u korak 2

#### UC16-Dodavanje kontejnera

- **Glavni sudionik:** Administrator
- **Cilj:** Dodavanje novog kontejnera na karti
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav
- **Opis osnovnog tijeka:**
  1. Korisnik odabire mjesto na karti
  2. Korisnik dodaje novi kontejner na odabrano mjesto
  3. Sustav sprema unesenu promjenu

#### UC17-Brisanje kontejnera

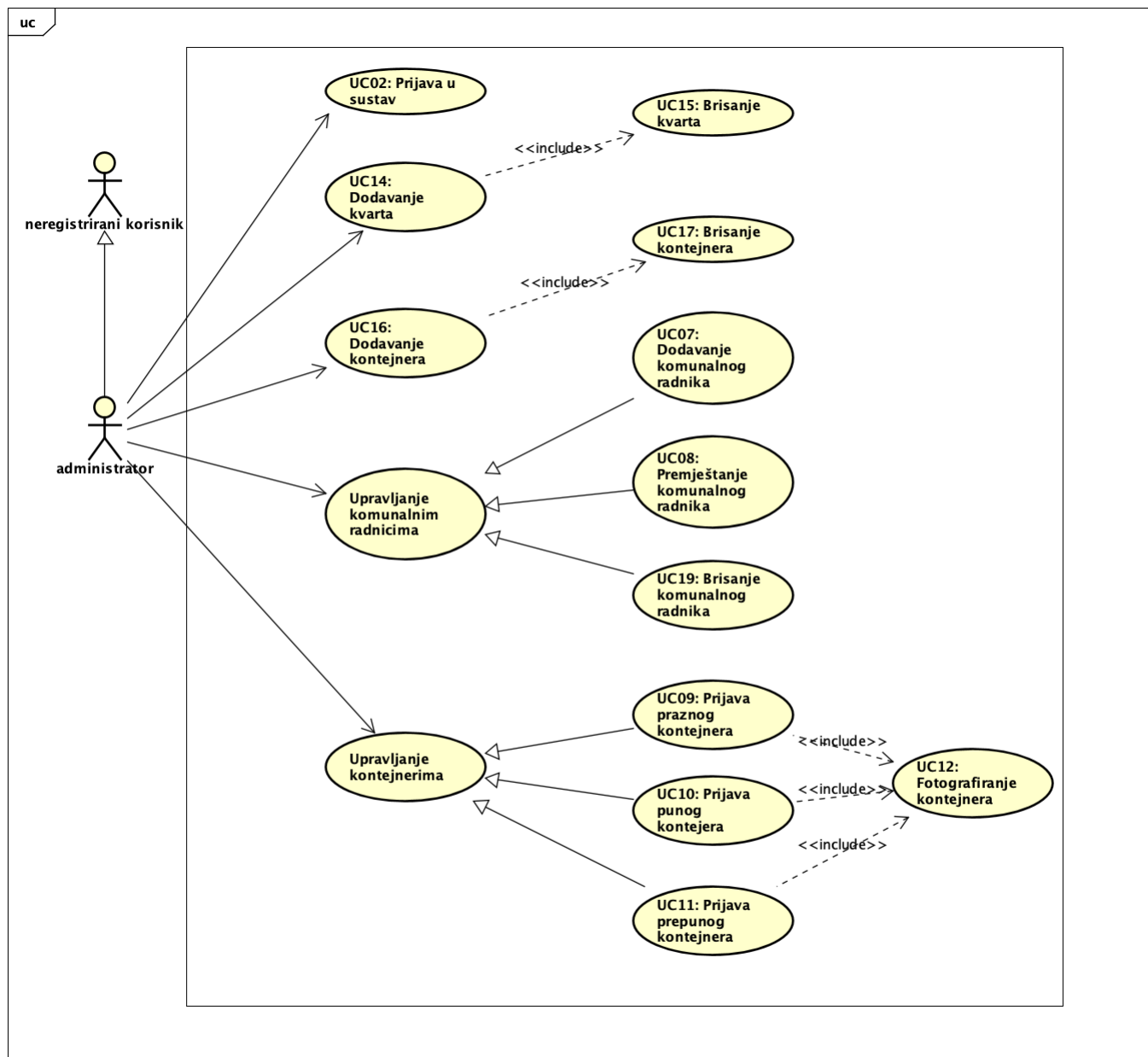
- **Glavni sudionik:** Administrator
- **Cilj:** Brisanje kontejnera s karte
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav
- **Opis osnovnog tijeka:**
  1. Korisnik odabire kontejner na karti
  2. Korisnik uklanja odabrani kontejner
  3. Sustav sprema unesenu promjenu

#### UC19-Brisanje komunalnog radnika

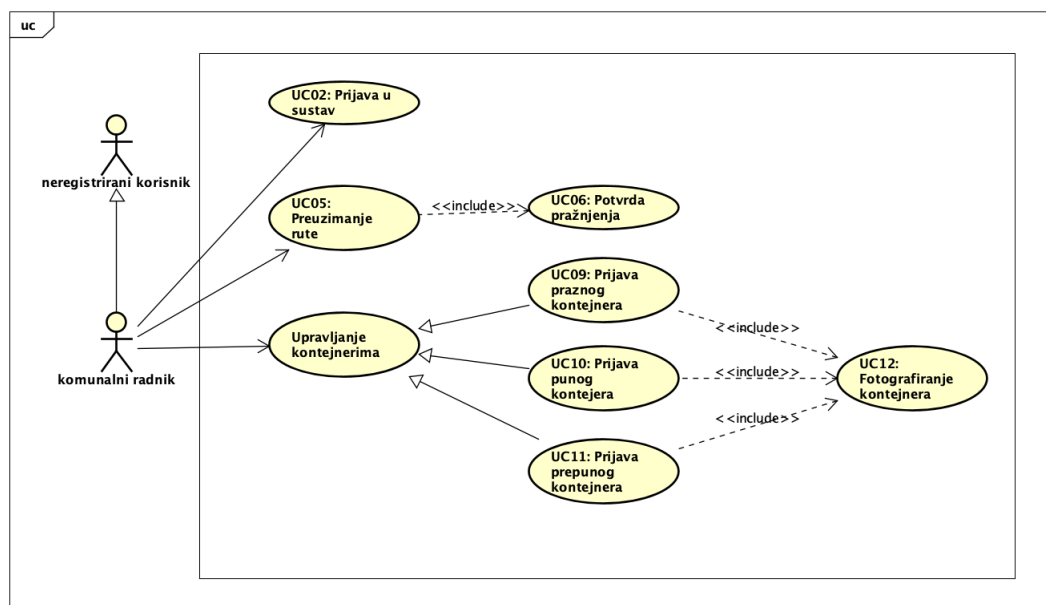
- **Glavni sudionik:** Administrator
- **Cilj:** Brisanje korisničkog računa komunalnom radniku
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav
- **Opis osnovnog tijeka:**
  1. Korisnik odabire kvart
  2. Sustava prikazuje popis radnika u odabranom kvartu
  3. Korisnik odabire komunalnog radnika

4. Sustav prikazuje podatke o odabranom radniku
5. Korisnik odabire brisanje odabranog radnika
6. Sustav sprema unesenu promjenu

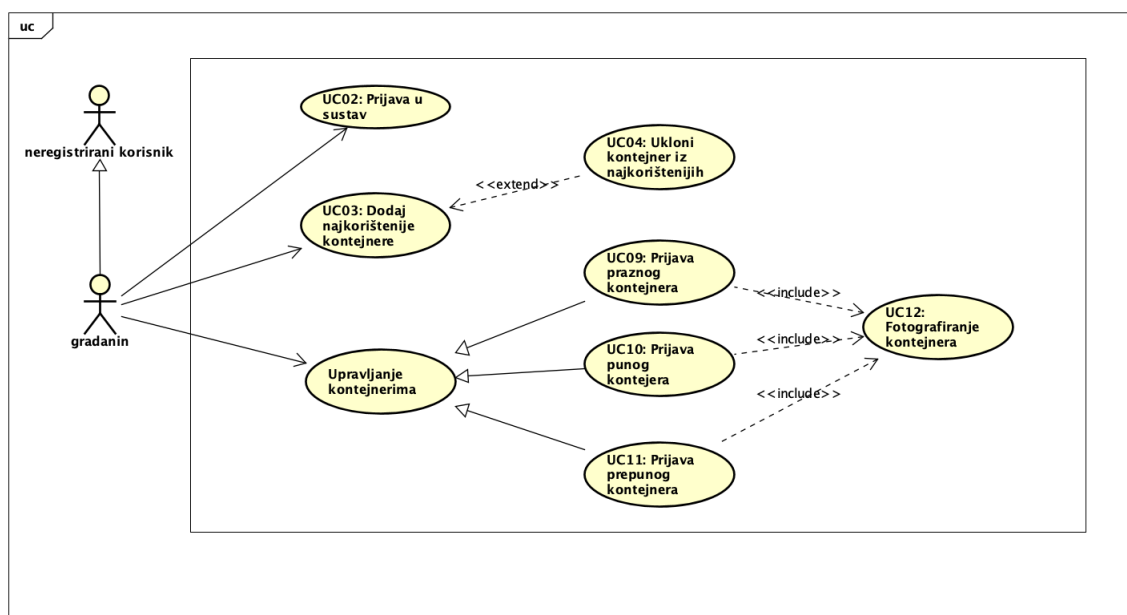
### Dijagrami obrazaca uporabe



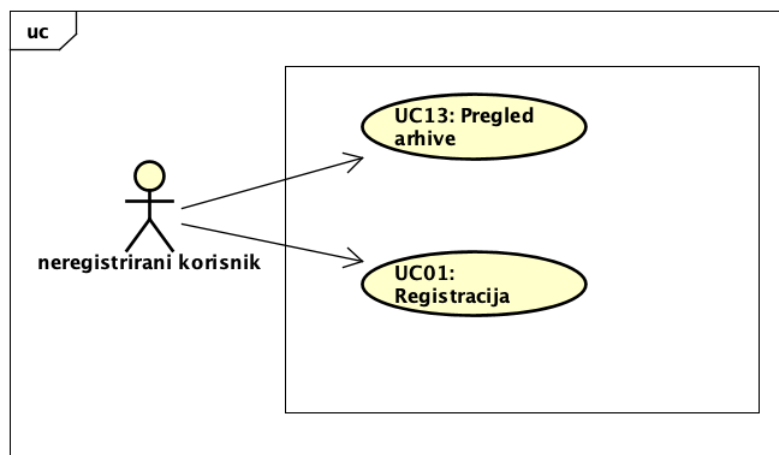
Slika 3.1: Dijagram obrasca uporabe, funkcionalnost administratora



Slika 3.2: Dijagram obrasca uporabe, funkcionalnost komunalnog radnika



Slika 3.3: Dijagram obrasca uporabe, funkcionalnost građanina

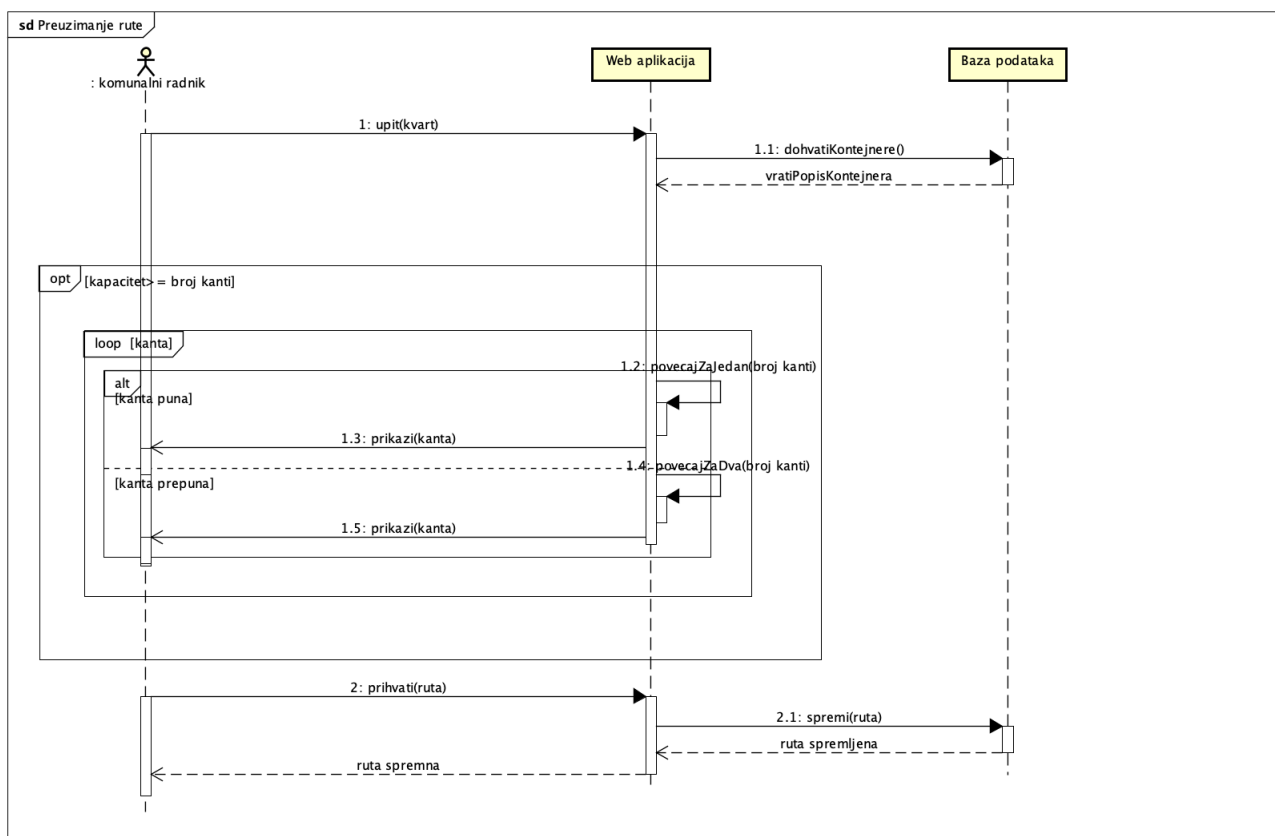


Slika 3.4: Dijagram obrasca uporabe, funkcionalnost neregistriranog korisnika

### 3.1.2 Sekvencijski dijagrami

#### Obrazac uporabe 05 - Preuzimanje rute

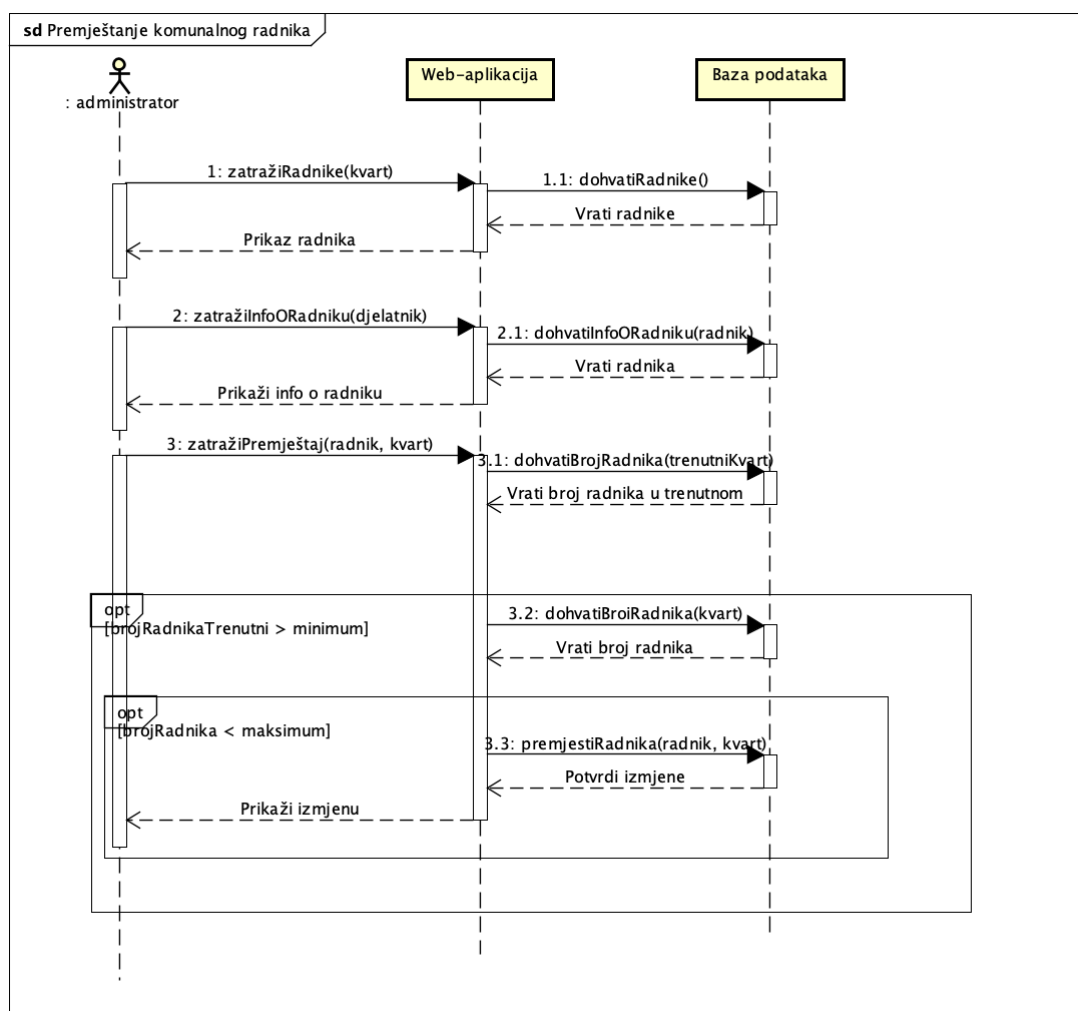
Komunalni radnik šalje zahtjev za dodjeljivanje rute kojom će prolaziti prilikom pražnjenja kontejnera u obliku popisa lokacija kontejnera koje mora isprazniti. Poslužitelj dohvaća kante u kvartu u kojem dotični komunalni radnik djeluje te radniku predaje popis kontejnera kojih ima toliko da zadovolje kapacitet kamiona. Uzmimo za primjer kamion kapaciteta 20 - dodavanjem jednog punog kontejnera na popis kapacitet se smanjuje na 19, a dodavanjem jednog prepunog kontejnera kapacitet se smanjuje na 18.



Slika 3.5: Sekvencijski dijagram za UC05

**Obrazac uporabe 08 - Premještanje komunalnog radnika**

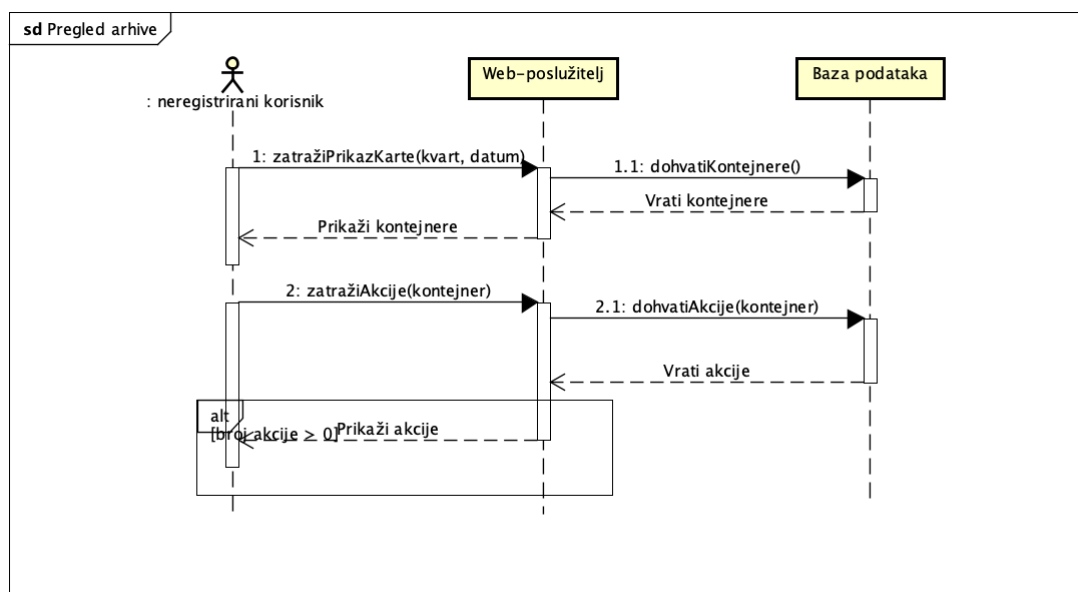
Kako bi izvršio premještanje radnika iz jednog kvartovskog poduzeća u drugo administrator mora poduzeti slijedeće; administrator bira kvart kojem je dotični radnik trenutno pridjeljen te traži u njemu istog radnika. Poslužitelj mu izlistava popis svih djelatnika u tom poduzeću te administrator bira traženog djelatnika. Poslužitelj mu izlistava informacije o tom djelatniku te mogućnosti s istim. Administrator bira premještanje te unosi željeni odredišni kvart. Ukoliko u trenutnom kvartovskom poduzeću tom akcijom neće biti premalo radnika niti u odredišnom previše poslužitelj obavlja zahtjev te dojavljuje korisniku informaciju o uspješnom izvođenju zahtjeva. U suprotnom premještanje se neće izvršiti.



Slika 3.6: Sekvencijski dijagram za UC08

**Obrazac uporabe 13 - Pregled arhive**

Kako bi neregistrirani korisnik mogao pregledavati arhivu mora prvotno unijeti kvart za kojeg želi pregledati arhivu i datum. Poslužitelj mu vraća prikaz karte sa kontejnerima koji su se u tom trenutku nalazili na tom kvartu. Korisnik može odabrati pojedinu kantu te će mu poslužitelj prikazati popis svih prijava i pražnjenja tog kontejnera toga dana.



Slika 3.7: Sekvencijski dijagram za UC13

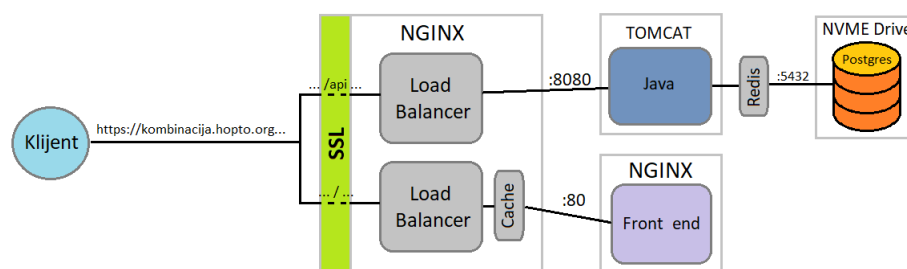
## 3.2 Ostali zahtjevi

- Sustav treba omogućiti rad više korisnika u stvarnom vremenu
- Sustav i korisničko sučelje trebaju podržavati diakritičke znakove
- Sustav treba biti implementiran kao web aplikacija koristeći objektno-orijentirane jezike
- Pristup sustavu mora biti omogućen iz javne mreže pomoću HTTPS-a
- Veza uspostavljena s bazom podataka treba biti sigurna, brza i otporna na vanjske greške
- Transakcije nad bazom podataka moraju imati ograničeno trajanje
- Nadogradnja sustava ne smije narušavati postojeće funkcionalnosti sustava
- Sustav treba biti jednostavan i praktičan za korištenje
- Pogreške pri radu korisnika u korisničkom sučelju ne smiju utjecati na ispravan rad sustava.



## 4. Arhitektura i dizajn sustava

### 4.1 Arhitektura sustava



Slika 4.1: Arhitektura sustava

Slika prikazuje arhitekturu svih sustava, odnosno način na koji su svi dijelovi aplikacije međusobno povezani te način na koji korisnik s njima interagira. U cjelokupnom sustavu prepoznaju se sljedeći entiteti i podsustavi:

- **Klijent** - putem preglednika šalje HTTPS zahtjeve aplikaciji
- **1. NGINX** - poslužitelj koji šalje i prima HTTPS zahtjeve, a sadrži dva *Load Balancer-a* i *cache* statičkih resursa (*HTML, JavaScript, CSS*)
- **2. NGINX** - poslužitelj na kojem se nalazi *front end* aplikacije
- **Tomcat** - poslužitelj na kojem je pokrenuta *Spring* aplikacija
- **Redis** - *cache* između aplikacije i baze podataka
- **NVME Drive** - poslužitelj na kojem je pokrenuta PostgreSQL baza podataka

Klijent koristeći URL aplikacije - `https://kombinacija.hopto.org` - šalje HTTPS zahtjeve aplikaciji koristeći se pritom nekim web preglednikom. Ti upiti pristižu na *reverse proxy* poslužitelj NGINX koji je osiguran SSL-om, a svojim *Load Balancer-ima* upravlja zahtjevima i raspoređuje ih na točne portove. Pri tome se zahtjevi oblika `https://kombinacija.hopto.org/api/...` proslijeđuju na port :8080 prema *Tomcat* poslužitelju, a zahtjevi oblika `https://kombinacija.hopto.org/...` na port :80 prema drugom NGINX poslužitelju. Java aplikacija pokrenuta na *Tom-*

cat poslužitelju putem *Redis cache-a* pristupa PostgreSQL bazi podataka na portu 5432.

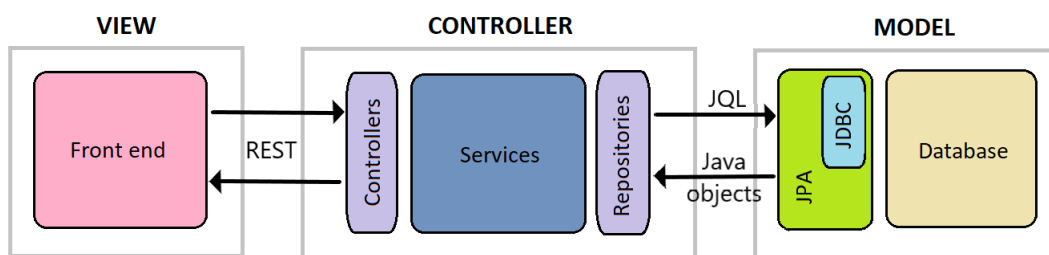
Poslužitelji *Tomcat* i *2. NGINX* su *instance*, što znači da se u slučaju preintenzivnog korištenja aplikacije može napraviti dodatni poslužitelj sa jednakim sadržajem, a *Load Balancer-i* će osigurati da se svakom daje podjednak udio zahtjeva. Zbog toga je aplikacija lako skalabilna.

## 4.2 Arhitektura aplikacije

Za izradu aplikacije na strani *back end-a* odabrali smo programski jezik Java, koristeći se pritom Spring Boot radnim okvirom koji uvelike olakšava razvoj web aplikacija. Radi lakšeg uključivanja vanjskih biblioteka te radi postizanja neovisnosti o razvojnom okruženju koristimo Maven.

Na strani *front end-a* koristimo radni okvir Bootstrap, pomoću kojeg koristimo standardne jezike za dizajniranje i ponašanje web stranica na strani klijenta - HTML, CSS i JavaScript. Radi jednostavnijeg pisanja, uz *vanilla JavaScript* koristimo i *jQuery* biblioteku, kojom je uvelike olakšano slanje asinkronih (AJAX) zahtjeva na web poslužitelj.

Arhitektura sustava se temelji na konceptu MVC (*Model-View-Controller*), kako bi se što jasnije i smislenije razdvojili pojedini dijelovi aplikacije te omogućio njihov istodoban i neovisan razvoj.



Slika 4.2: MVC prikaz aplikacije

**View** označava podsustav zadužen za *user-friendly* prikaz podataka korisniku. U slučaju naše aplikacije, razvija se potpuno neovisno od ostalih dijelova, te je pokrenut na zasebnom poslužitelju. Osim što prikazuje *model*, *view* je zadužen i za slanje zahtjeva *controller-u*.

*View* sa elementom *Front end* se u potpunosti preslikava na prethodnu sliku u poslužitelj *NGINX*, u kojem se također nalazi element *Front end*.

**Controller** označava podsustav koji upravlja zahtjevima pristiglim iz *view-a*, dohvaća stvarne podatke iz *model-a* te vrši logiku svih mogućih akcija i događaja. Svi dijelovi *controller-a* se preslikavaju u poslužitelj *Tomcat* sa prethodne slike, a njegov središnji dio predstavljaju servisi, koji se u samoj aplikaciji nalaze u posebnom paketu, a koji obavljaju cjelokupnu poslovnu logiku aplikacije. Kako bi to obavljali uspješno, moraju moći komunicirati sa ostalim podsustavima.

Komunikacija s *view-om* se izvodi putem sučelja *Controllers*, koje na slici predstavlja cijeli paket u kojem je definiran API (*Application Programming Interface*) putem kojeg se aplikaciji šalju zahtjevi ili se dohvaćaju podaci. API koji smo odabrali je REST (*REpresentational State Transfer*) API, kako bi se omogućila lakša komunikacija sa drugim sustavima, te kako bi aplikacija bila lako proširiva (primjerice, na isti API se može spojiti i mobilna aplikacija).

*Controller* sa modelom komunicira putem sučelja *Repositories* koje također predstavlja cijeli paket kojim se iz baze podataka dohvaćaju konkretne objekte i podatke. Upiti prema bazi se ostvaruju pomoću JQL (*Java Query Language*), oslanjajući se pritom većinom na već implementirane metode uključene u Spring Framework.

**Model** predstavlja sve strukture podataka, objekte i zapise vezane uz aplikaciju, koji su spremljeni u bazi podataka. Komunikacija s bazom je ostvarena putem JPA/JDBC specifikacije. Osim JPA, koji se također nalazi na poslužitelju *Tomcat*, *model* se preslikava u disk *NVME Drive* na kojem je pokrenut PostgreSQL server.

## 4.3 Baza podataka

Kao sustav upravljanja bazom podataka koristimo PostgreSQL Sustav za Upravljanje Bazom Podataka (SUBP). Na bazu se spajamo putem *Java DataBase Connectivity* (JDBC) specifikacije, koja omogućuje Java programima spajanje na SUBP i njegovo korištenje. Na JDBC specifikaciju nadograđuje se *Java Persistence API* (JPA) koji pruža usluge *Object-Relational Mapping* (ORM), čime se odabrani Java objekti automatski spremaju u bazu podataka.

### 4.3.1 Opis tablica

U bazi podataka nalazi se 9 relacija:

- **Persons** - relacija koja opisuje osobu u najširem smislu; osoba može biti korisnik aplikacije, komunalni radnik ili administrator sustava, te se u ovoj relaciji spremaju zajednički atributi tih aktora
- **Admins** - relacija u kojoj se nalaze zapisi svih administratora
- **Employees** - relacija u kojoj se nalaze zapisi svih komunalnih radnika
- **Citizens** - relacija u kojoj se nalaze zapisi svih građana - regularnih korisnika aplikacije
- **Container** - relacija u kojoj su spremljeni zapisi o kontejnerima
- **Neighborhoods** - relacija u kojoj su spremljeni zapisi o susjedstvima
- **Favorites** - relacija u kojoj su spremljeni zapisi o parovima Person-Container, sa značenjem da je osoba Person spremila kontejner Container radi brzog pristupa.
- **Pings** - relacija u kojoj su spremljeni zapisi o parovima Person-Container, sa značenjem da je osoba Person prijavila kontejner Container kao pun, pretrpan ili prazan.
- **Emptyings** - relacija u kojoj su spremljeni zapisi o pražnjenjima kontejnera oblika Employee-Container, sa značenjem da je radnik Employee ispraznio kontejner Container

Tablica Person je generička tablica za osobu, a nju nasljeđuju tablice Admin, Employee i Citizen, svaki sa svojim dodatnim atributima te stranim ključem koji pokazuje na tablicu Person gdje se nalaze zajednički atributi svojstveni svima trima entitetima. Tablice Favorites, Pings i Emptyings su tablice koje postoje kako bi se opisala N-N veza između relacija koje povezuju.

### 4.3.2 Definicije tablica

U narednim definicijama **podebljani atributi** označavaju primarne ključeve, a *kurziv* strane ključeve.

Person		
<b><i>id</i></b>	BIGINT	Identifikator osobe; svaka osoba ima svoj jedinstveni ID
name	VARCHAR	Ime osobe, ne smije biti NULL

Person		
last_name	VARCHAR	Prezime osobe, ne smije biti NULL
email	VARCHAR	E-mail osobe, koristi se pri prijavi korisnika u sustav, ne smije biti NULL i jedinstvena je vrijednost
pwd_hash	VARCHAR	Sažetak lozinke koju je osoba odabrala, ne smije biti NULL

Tablica 4.1: Tablica *Person*

Admin		
<i>id</i>	BIGINT	Jedini atribut tablice - strani ključ koji pokazuje na tablicu Person.

Tablica 4.2: Tablica *Admin*

Citizen		
<i>id</i>	BIGINT	Strani ključ koji pokazuje na tablicu Person.
reputation	INT	Atribut koji označava reputaciju korisnika; služi za procjenu vjerodostojnosti njegove prijave; ne smije biti NULL

Tablica 4.3: Tablica *Citizen*

Employee		
<i>id</i>	BIGINT	Strani ključ koji pokazuje na tablicu Person.
oib	VARCHAR(11)	OIB komunalnog radnika; ne smije biti NULL i mora biti jedinstvena vrijednost
<i>neighborhood_id</i>	BIGINT	Strani ključ koji pokazuje na tablicu Neighborhood. Određuje kojem kvartu pripada radnik.

Tablica 4.4: Tablica *Employee*

Container		
id	BIGINT	Identifikator kontejnera; svaki kontejner ima svoj jedinstveni ID
latitude	DOUBLE	Geografska širina lokacije kontejnera; ne smije biti NULL.
longitude	DOUBLE	Geografska dužina lokacije kontejnera; ne smije biti NULL.
route_status	INT	Označava je li kontejner trenutno u ruti nekog radnika; ne smije biti NULL.
pings_since_emptied	INT	Broj prijava kontejnera od njegovog zadnjeg pražnjenja; ne smije biti NULL.
neighborhood_id	BIGINT	Strani ključ koji pokazuje na tablicu Neighborhood. Određuje u kojem se kvartu nalazi kontejner.

Tablica 4.5: Tablica *Container*

Neighborhood		
id	BIGINT	Identifikator susjedstva; svako susjedstvo (kvart) ima svoj jedinstveni ID
latitude	DOUBLE	Geografska širina lokacije komunalnog središta u susjedstvu; ne smije biti NULL.
longitude	DOUBLE	Geografska dužina lokacije komunalnog središta u susjedstvu; ne smije biti NULL.
name	VARCHAR	Ime susjedstva; ne smije biti NULL.
workerCapacity	INT	Kapacitet za radnike; ne smije biti NULL.

Tablica 4.6: Tablica *Neighborhood*

Favorites		
id	BIGINT	Identifikator oznake favorita; svako označavanje ima svoj jedinstveni ID.
container_id	BIGINT	Strani ključ koji pokazuje na relaciju Container

Favorites		
<i>owner_id</i>	BIGINT	Strani ključ koji pokazuje na relaciju Person

Tablica 4.7: Tablica *Favorites*

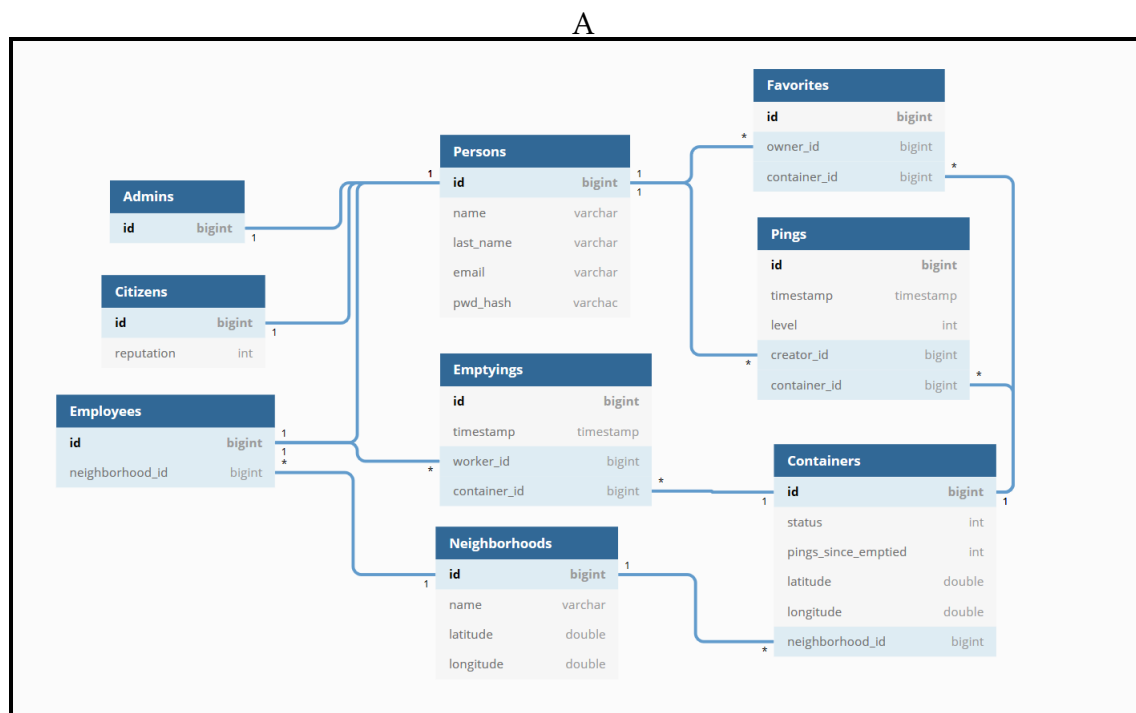
Pings		
<b>id</b>	BIGINT	Identifikator prijave; svako označavanje ima svoj jedinstveni ID.
level	INT	Level koji označava vrstu prijave: 0 je prijava praznog, 1 prijava punog i 2 prijava pretrpanog kontejnera; ne smije biti NULL
timestamp	BIGINT	Vremenska oznaka trenutka u kojem se dogodila prijava kontejnera; ne smije biti NULL
photo_path	VARCHAR	Mjesto na disku na kojem se nalazi fotografija koju je priložio korisnik
<i>container_id</i>	BIGINT	Strani ključ koji pokazuje na relaciju Container
<i>owner_id</i>	BIGINT	Strani ključ koji pokazuje na relaciju Person

Tablica 4.8: Tablica *Ping*

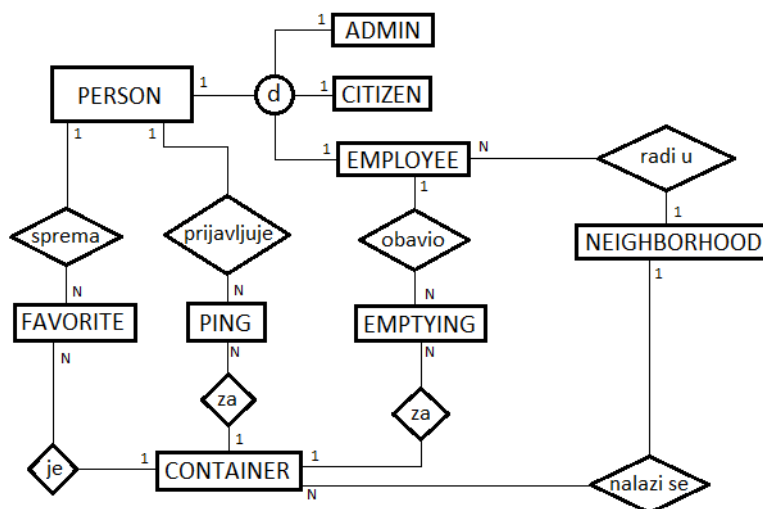
Emptyings		
<b>id</b>	BIGINT	Identifikator prijave; svako označavanje ima svoj jedinstveni ID.
timestamp	BIGINT	Vremenska oznaka trenutka u kojem se dogodila prijava kontejnera; ne smije biti NULL
<i>container_id</i>	BIGINT	Strani ključ koji pokazuje na relaciju Container
<i>worker_id</i>	BIGINT	Strani ključ koji pokazuje na relaciju Employee

Tablica 4.9: Tablica *Emptying*

### 4.3.3 Dijagram baze podataka



Slika 4.3: Dijagram baze podataka



Slika 4.4: Entitetsko-Relacijski model baze podataka



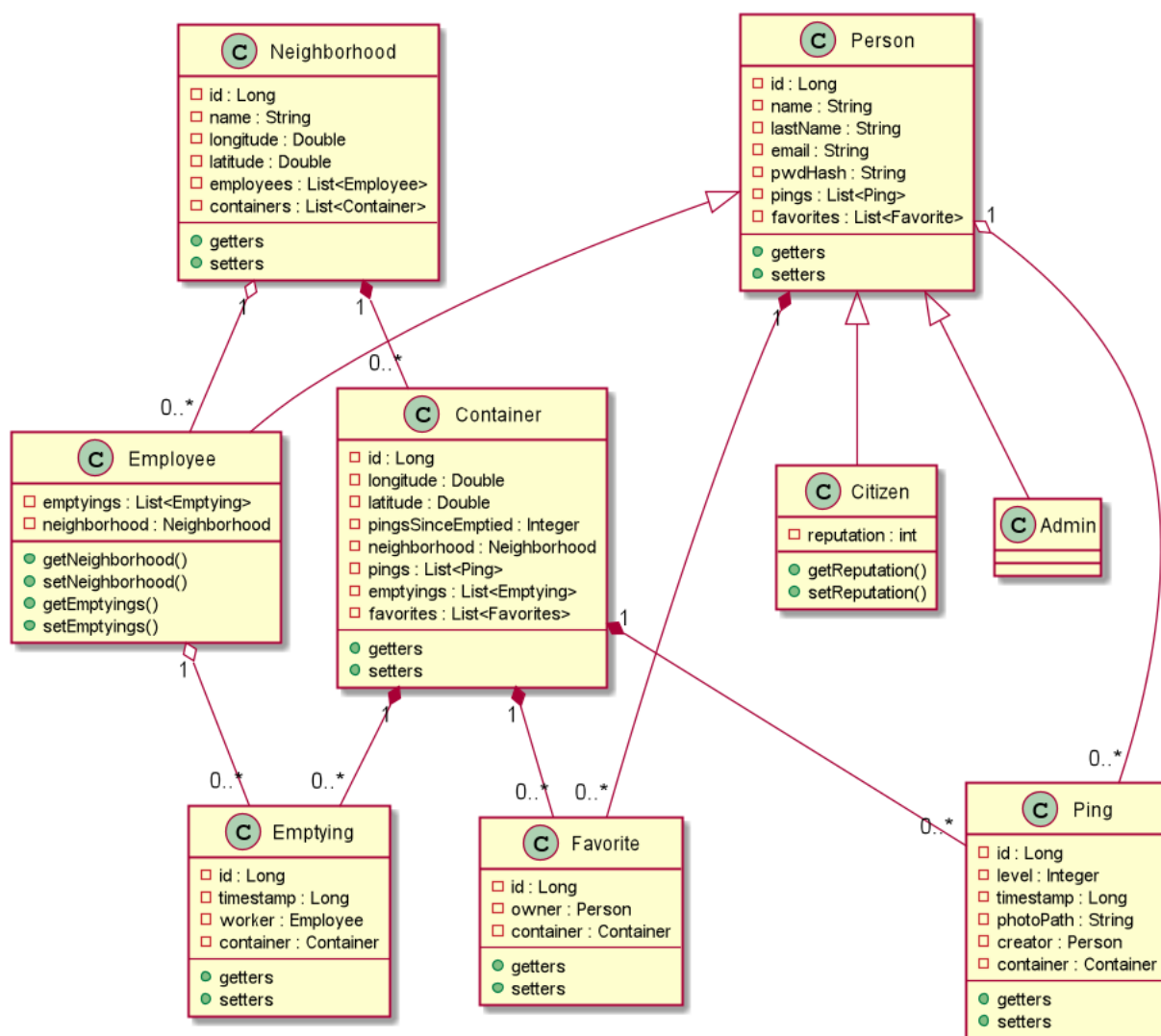
## 4.4 Dijagram razreda

Slika 4.5 prikazuje dijagram razreda koji predstavljaju model podataka - reprezentaciju stvarnog svijeta. On je vrlo sličan ER modelu i dijagramu baze podataka, ali ne i identičan: uspoređujući ih, vidljivo je kako se Java objekti i sve veze (One-to-one, One-to-many, Many-to-many) mapiraju u bazu podataka.

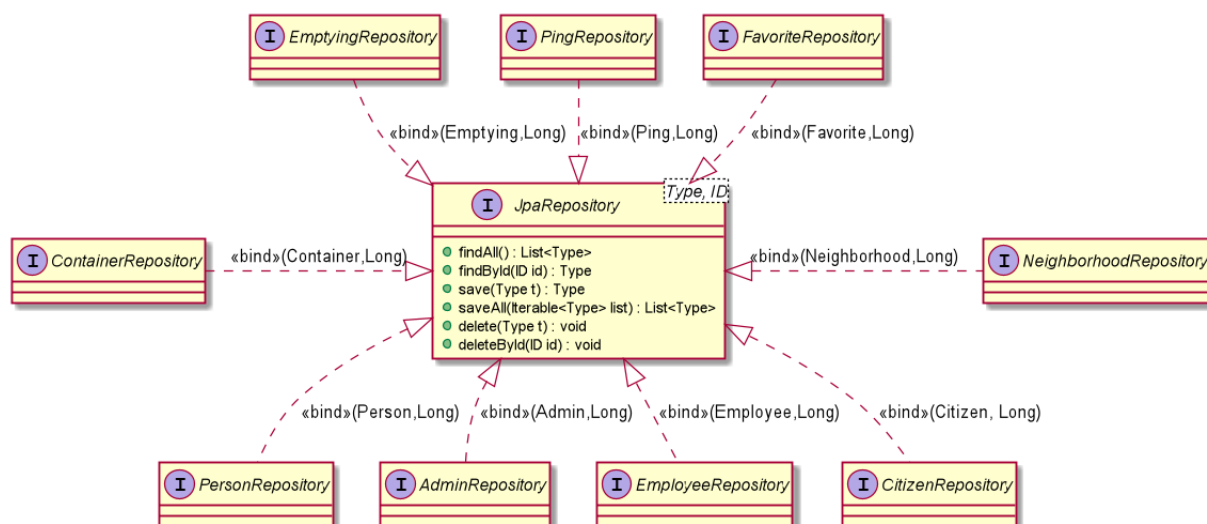
Na slici 4.6. prikazan je dijagram razreda paketa *dao*. DAO (*Data Access Object*) je paket koji predstavlja sloj između logike aplikacije i baze podataka. Pomoću DAO objekata aplikacija dohvaća objekte iz baze podataka te ih uključuje u razne akcije, mijenja, stvara nove objekte vezane uz njih i sprema ih natrag u bazu. Sve metode u svim razredima ovog paketa su već implementirane u Spring-ovom sučelju *JpaRepository*, koje sva naša sučelja nasljeđuju.

Slika 4.7 prikazuje dijagram razreda paketa *controller*, koji su zaduženi za primanje HTTP zahtjeva na način da se svaka metoda u pojedinom *controller* razredu mapira na određeni URL i određenu HTTP metodu (GET, POST, PUT...). Tako je primjerice metoda *registerUser()* u razredu *PublicController* pozvana isključivo ako korisnik pošalje POST zahtjev na URL `"/register"`, a metoda *testAuthorization()* se koristi kako bi korisnik provjerio ispravnost autorizacijskih podataka slanjem GET zahtjeva na URL `"/auth"`.

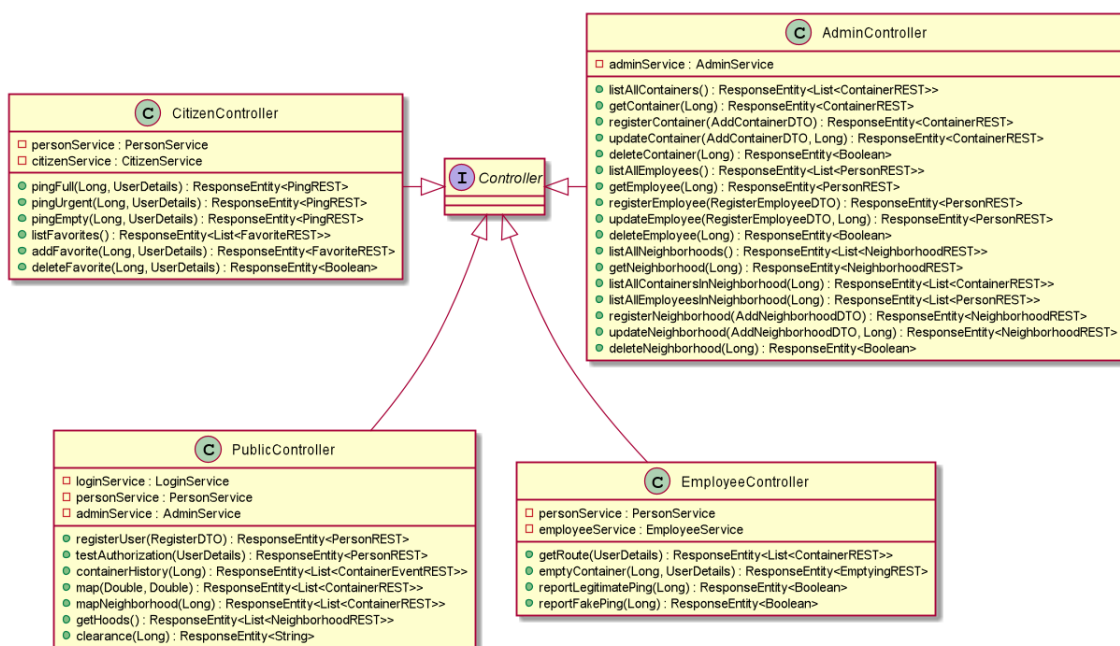
Konačno, svaki razred iz prethodnog paketa sadrži referencu na barem jedan razred iz paketa *service* sa dijagrama na slici 4.8. Razredi ovog paketa su servisi koji obavljaju poslovnu logiku aplikacije, pružajući uslugu razredima iz paketa *controller*, a koristeći razrede iz paketa *dao*. Servisi često osim na repozitorije imaju i reference na druge razrede iz istog paketa.



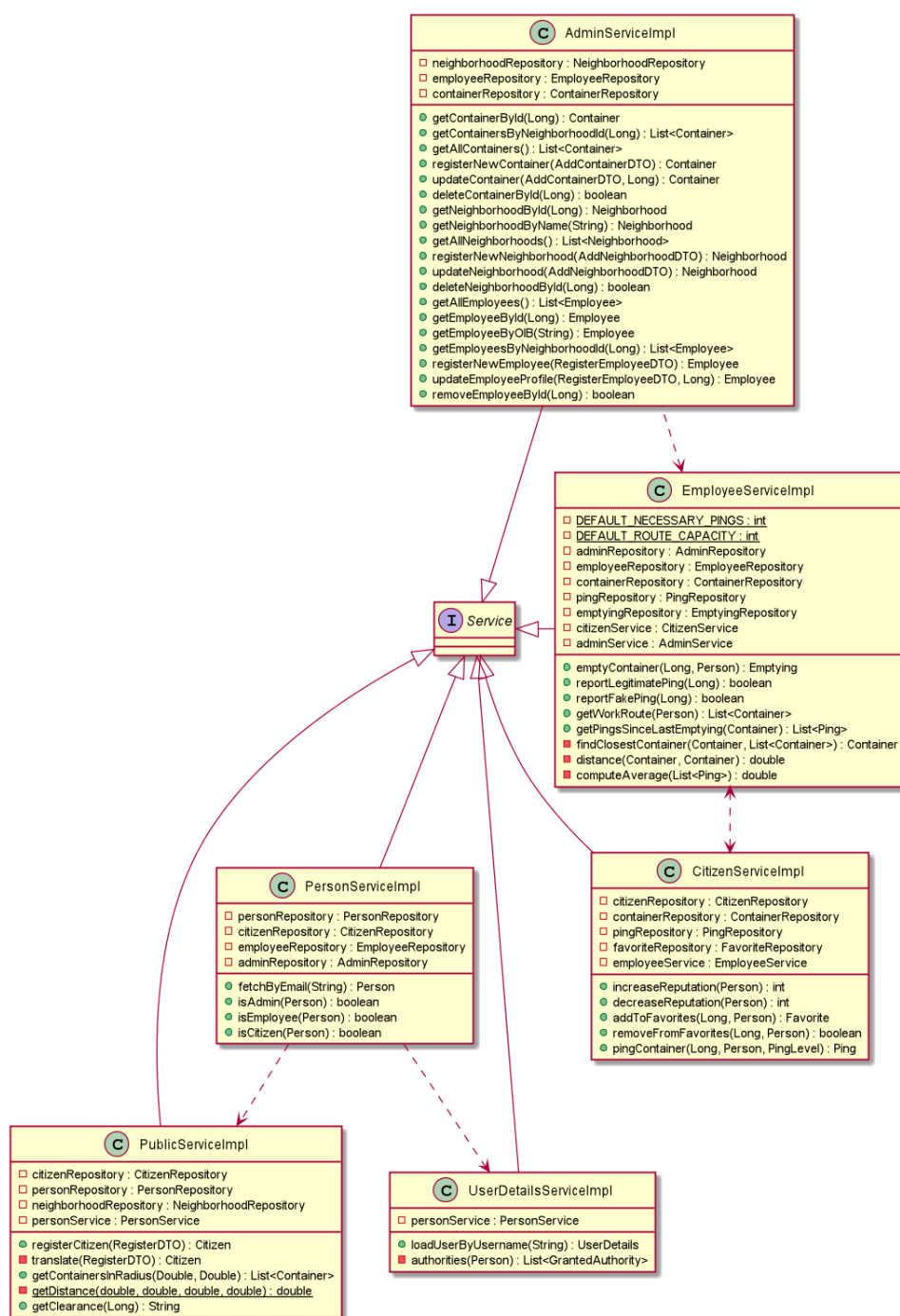
Slika 4.5: Dijagram razreda koji predstavljaju model podataka



Slika 4.6: Dijagram razreda paketa DAO



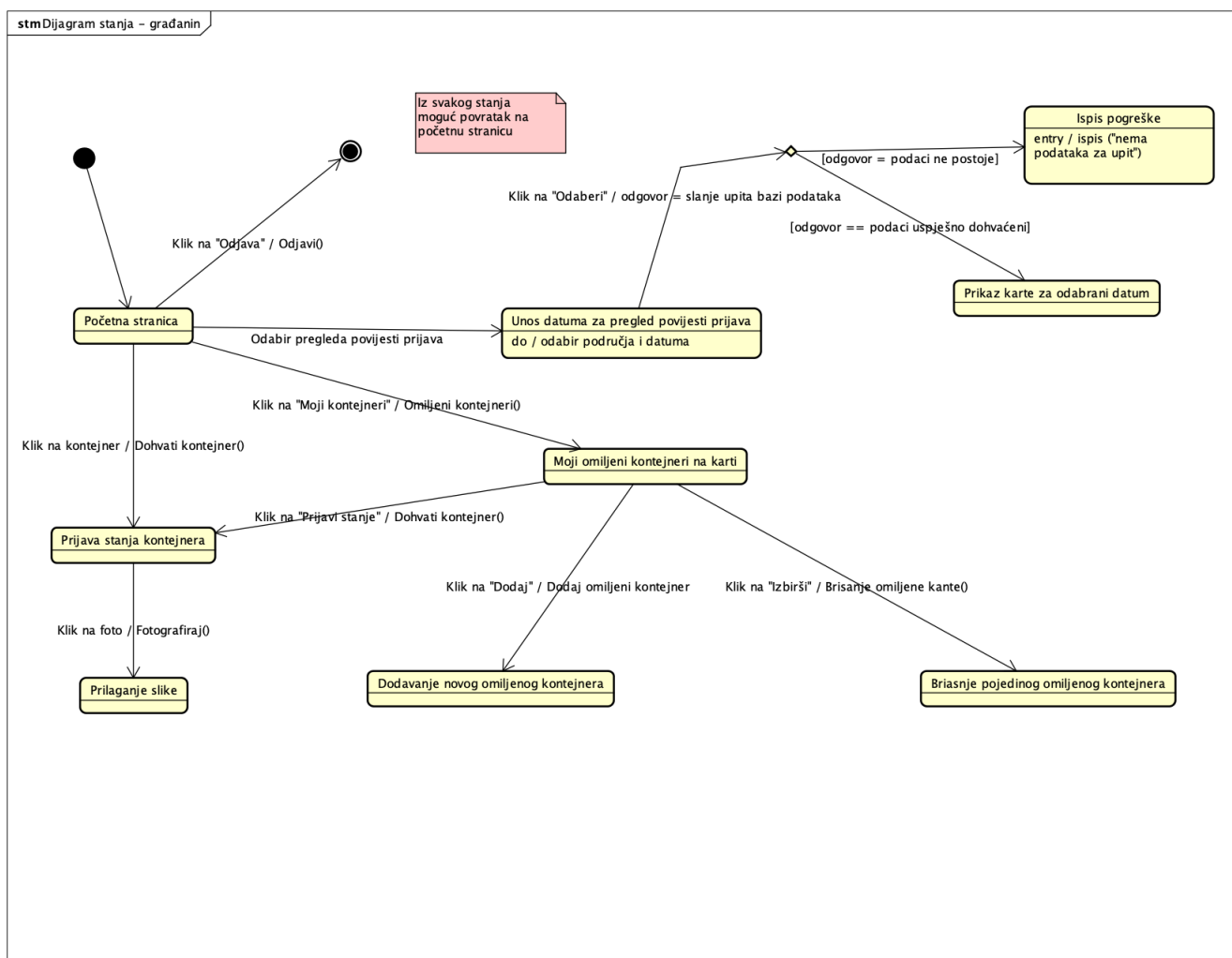
Slika 4.7: Dijagram razreda paketa controller



Slika 4.8: Dijagram razreda paketa service

## 4.5 Dijagram stanja

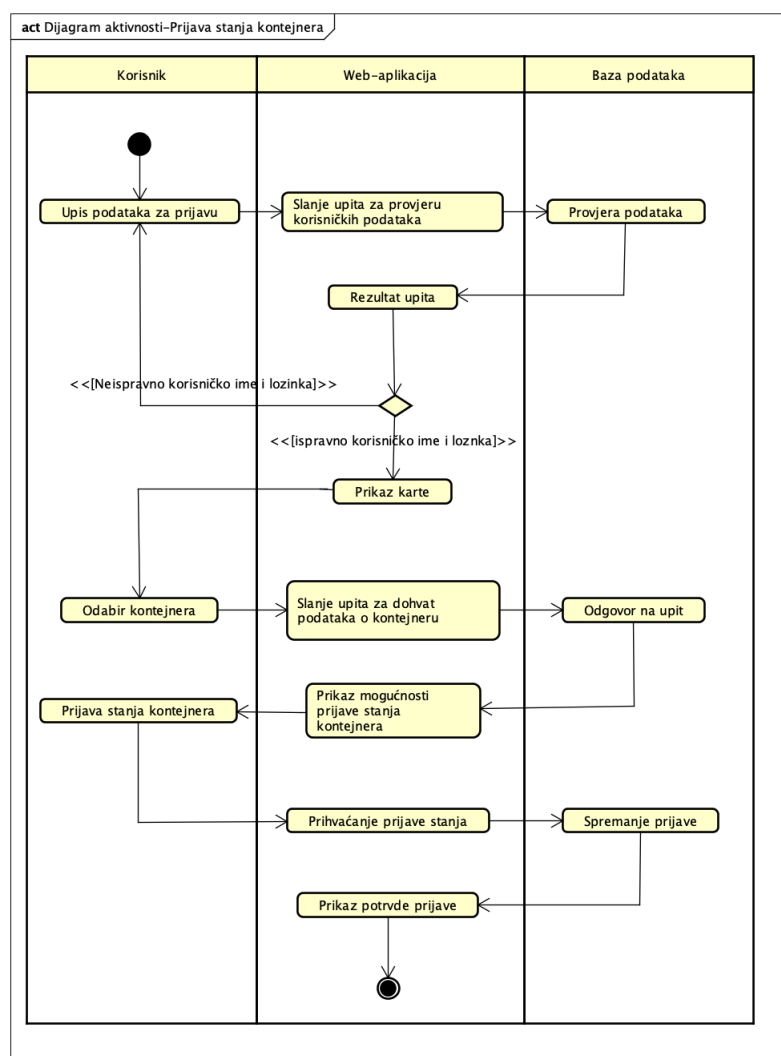
Dijagram stanja prikazuje stanja objekta te prijelaze iz jednog stanja u drugo temeljene na događajima. Na slici 4.9 prikazan je dijagram stanja za registriranog korisnika. Nakon prijave, građaninu se prikazuje početna stranica na kojoj može pregledati kartu i kontejnere. Za odabrani kontejner ima opciju prijavljivanja stanja kontejnera. Dodatno može priložiti i sliku prijavljenom stanju. Također, klikom na "Omiljeni kontejneri" prikaz naglašava prethodno označene najkorištenije kontejnere te tako olakšava prijavu stanja te iste može i obrisati ili dodati nove. Odabirom datuma i pripadnog željenog područja prikazuje se stanje karte za dotični datum na tom mjestu ukoliko traženi zapis postoji.



Slika 4.9: Dijagram stanja

## 4.6 Dijagram aktivnosti

Dijagram aktivnosti primjenjuje se za opis modela toka upravljanja ili toka podataka. Ne upotrebljava se za modeliranje događajima poticanog ponasanja. U modeliranju toka upravljanja svaki novi korak poduzima se nakon završenog prethodnog, a naglasak je na jednostavnosti. Na dijagramu aktivnosti 4.10 prikazan je proces prijave stanja kontejnera. Građanin se prijavi u sustav i odabere kontejner za kojeg želi prijaviti stanje. Zatim mu se prikazuje popis opcija za prijavu te on bira koje stanje želi prijaviti. Nakon što potvrdi odabir sustav mu vraća potvrdu o uspješnoj prijavi.



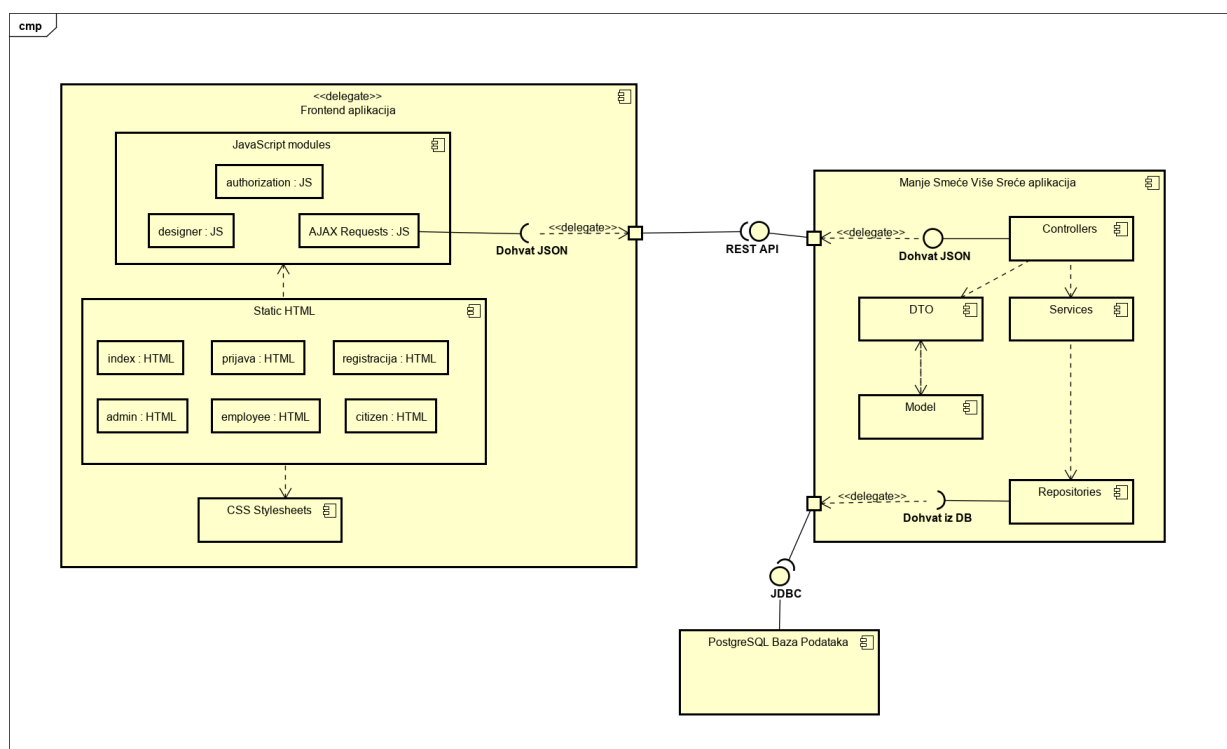
Slika 4.10: Dijagram aktivnosti

## 4.7 Dijagram komponenti

Dijagram komponenti služi za prikaz komponenti koje tvore sustav, njihove organizacije i međuovisnosti. Na slici 4.11 prikazan je dijagram komponenti za programsku potporu koju smo razvili. Aplikacija je, najgrublje gledano, podijeljena na tri strogo odvojene cjeline:

- *Frontend aplikacija* pokrenuta na jednom NGINX poslužitelju i na kojoj se nalaze statički resursi poput HTML stranica, JavaScript modula te CSS datoteka pomoću kojih su sve HTML stranice dizajnirane.
- *Manje Smeće Više Sreće aplikacija* pokrenuta na Tomcat poslužitelju u kojoj se nalazi sama implementacija aplikacije podjeljena na razine *Controller*, *Service*, *Repositories*, *Model* i *DTO*. Svaka od tih razina čini jedan neizostavni dio aplikacije.
- *PostgreSQL Baza Podataka* pokrenuta na zasebnom poslužitelju. U nju se spremaju svi podaci u obliku Java objekata pomoću JDBC/JPA specifikacije.

Statičke HTML stranice određuju formu, a njihov izgled određen je CSS *stylesheetovima* na istom poslužitelju. Za dinamičko dohvaćanje podataka s *backenda* kao i njihov prikaz ugradnjom u HTML koriste se JavaScript moduli. U modulima obuhvaćenim grupom *AJAX Requests* nalazi se po jedna metoda za svaku metodu u svim razredima *Controller*, time omogućujući jednostavno spajanje na REST API i dohvat podataka u JSON formatu.



Slika 4.11: Dijagram komponenti



## 5. Implementacija i korisničko sučelje

### 5.1 Korištene tehnologije i alati

Komunikacija u timu realizirana je korištenjem aplikacije WhatsApp<sup>1</sup> kako bi se svi mogli konzultirati pravovremeno te zbog mogućnosti grupnog poziva za lakše dogovaranje. Za izradu UML dijagrama primarno je korišten alat Astah UML<sup>2</sup>, no za određene dijagrame je radi preglednosti i oku ugodnijeg dizajna korišten *plugin* PlantUML<sup>3</sup> za IntelliJ radnu okolinu. Za upravljanje izvornim kodom i njegovim inačicama koristili smo Git<sup>4</sup>. Udaljeni repozitorij projekta u kojem se nalazi cjelokupni izvorni kod dostupan je na web platformi GitLab<sup>5</sup>.

Kao razvojno okruženje korišteni su IntelliJ IDEA<sup>6</sup> - integrirano razvojno okruženje (IDE) tvrtke JetBrains, i Eclipse<sup>7</sup> - IDE tvrtke The Eclipse Foundation. IDE pruža integraciju s drugim alatima za razvoj kao što je primjerice Maven<sup>8</sup> kojeg smo koristili za lokalno upravljanje projektom, uključivanje raznih biblioteka o kojima aplikacija ovisi, te automatiziranu izgradnju projekta i pakiranje u JAR arhivu te dakako za formatiranje koda kako bi se slagao s Javinim standardom. Oba IDE-a podržavaju sustave kontrole verzioniranja kao što je Git.

Aplikacija je napisana u jeziku Java<sup>9</sup> koristeći radni okvir Spring<sup>10</sup>, dizajniran kako bi što brže napredovali u razvoju aplikacije uz olakšanu konfiguraciju raznih komponenti. *Frontend* aplikacije je napisan koristeći HTML, CSS i JavaScript<sup>11</sup>, pri čemu smo za CSS koristili biblioteku Bootstrap<sup>12</sup> koja stranicama pruža responzivnost i kompatibilnost s mobilnim uređajima te jednostavniju izgradnju web stranice obzirom na predefinirane komponente. JavaScript smo koristili podjednako u

---

<sup>1</sup><https://www.whatsapp.com/>

<sup>2</sup><http://astah.net/editions/community>

<sup>3</sup><https://plantuml.com/>

<sup>4</sup><https://git-scm.com/>

<sup>5</sup><https://gitlab.com/>

<sup>6</sup><https://www.jetbrains.com/idea/>

<sup>7</sup><https://www.eclipse.org/>

<sup>8</sup><https://maven.apache.org/>

<sup>9</sup><https://www.java.com/en/>

<sup>10</sup><https://spring.io/>

<sup>11</sup><https://www.javascript.com/>

<sup>12</sup><https://getbootstrap.com/>

svojoj *vanilla* inačici kao i inačici proširenoj bibliotekom jQuery<sup>13</sup>. Obzirom na karakter aplikacije, odnosno nužnost uporabe iste "u pokretu" JavaScript se činio kao odlično rješenje koje omogućava racionalnije korištenje podataka koje će korisnik morati učitati prilikom svake interakcije sa sustavom.

Korisničko sučelje testirano je pomoću Seleniuma<sup>14</sup>, automatiziranog sustava za testiranje, kako bi postupak testiranja bio što konzistentniji i precizniji.

---

<sup>13</sup><https://jquery.com/>

<sup>14</sup><https://selenium.dev/>

## 5.2 Ispitivanje programskog rješenja

### 5.2.1 Ispitivanje komponenti

#### Ispitni slučaj 1: Dohvaćanje omiljenih kontejnera

Ulaz:

1. registrirani građanin

```
@Test
public void getFavoriteContainers() {
    Person p = new Person();
    p.setId(12L);

    Favorite f1 = new Favorite();
    Favorite f2 = new Favorite();

    List<Favorite> expectedFavorites = new ArrayList<>();
    expectedFavorites.add(f1);
    expectedFavorites.add(f2);

    Mockito.doReturn(expectedFavorites).when(favoriteRepository).findByOwnerId(12L);

    List<Favorite> actualFavorites = citizenService.getFavoriteContainers(p);

    assertThat(actualFavorites).isEqualTo(expectedFavorites);
}
```

Slika 5.1: Test: Dohvaćanje omiljenih kontejnera

Očekivani rezultat:

1. Dohvaćanje liste omiljenih kontejnera.

Rezultat:

Dohvaćena je lista omiljenih kontejnera. - Test je uspješno proveden.

#### Ispitni slučaj 2: Podizanje reputacije građanina

Ulaz:

1. registrirani korisnik

```
@Test
public void increaseReputationNeutral() {
    Person p = new Person();
    p.setEmail("pero.peric@fer.hr");

    Citizen c = new Citizen();
    c.setReputation(0);

    Mockito.doReturn(Optional.of(c)).when(citizenRepository).findByEmail("pero.peric@fer.hr");
    Mockito.when(citizenRepository.save(c)).thenReturn(c);

    citizenService.increaseReputation(p);

    Assert.assertEquals(expected: 10, (long)c.getReputation());
}
```

Slika 5.2: Test: Podizanje reputacije građanina

**Očekivani rezultat:**

1. Reputacija korisnika se podiže.

**Rezultat:**

Reputacija korisnika je podignuta. - Test je uspješno proveden.

**Ispitni slučaj 3: Postavljanje reputacije korisnika****Ulaz:**

1. vrijednost reputacije manja od 0

```
@Test (expected = RequestDeniedException.class)
public void increaseReputationException() {
    Person p = new Person();
    p.setEmail("pero.peric@fer.hr");

    Citizen c = new Citizen();
    c.setReputation(-156);

    citizenService.increaseReputation(p);
}
```

Slika 5.3: Test: Postavljanje reputacije korisnika

**Očekivani rezultat:**

1. Pojava iznimke, odbijen zahtjev.

**Rezultat:**

Pojava iznimke, odbijen zahtjev. - Test je uspješno proveden.

### Ispitni slucaj 4: Dohvaćanje kontejnera preko identifikacijske oznake

#### Ulaz:

1. identifikacijska oznaka kontejnera

```
@Test
public void getContainerById_Success() {
    Container c = new Container();
    c.setId(11);

    Mockito.doReturn(Optional.of(c)).when(containerRepository).findById(11);

    Container actual = adminService.getContainerById(containerId: 11);

    assertThat(actual).isEqualTo(c);
}
```

Slika 5.4: Test: Dohvaćanje kontejnera preko identifikacijske oznake

#### Očekivani rezultat:

1. Kontejner sa zadanom identifikacijskom oznakom.

#### Rezultat:

Vraćen je kontejner sa zadanom identifikacijskom oznakom. - Test je uspješno proveden.

### Ispitni slucaj 5: Dohvaćanje kontejnera iz susjedstva

#### Ulaz:

1. identifikacijska oznaka susjedstva

```
@Test
public void getContainerByNeighborhoodId_Success() {
    Neighborhood n = new Neighborhood();
    n.setId(11);

    Container c1 = new Container();
    Container c2 = new Container();

    List<Container> expectedContainers = new ArrayList<>();
    expectedContainers.add(c1);
    expectedContainers.add(c2);

    n.setContainers(expectedContainers);

    Mockito.doReturn(Optional.of(n)).when(neighborhoodRepository).findById(11);

    List<Container> actualContainers = adminService.getContainersByNeighborhoodId(11);

    assertThat(actualContainers).isEqualTo(expectedContainers);
}
```

Slika 5.5: Test: Dohvaćanje kontejnera iz susjedstva

**Očekivani rezultat:**

1. Lista kontejnera zadanog susjedstva.

**Rezultat:**

Vraćena je lista kontejnera zadanog susjedstva. - Test je uspješno proveden.

**Ispitni slučaj 6: Dohvaćanje komunalnog radnika prema OIBu****Ulaz:**

1. OIB

```
@Test
public void getEmployeeByOIB_Success(){
    Employee e = new Employee();
    e.setOIB("12345678900");

    Mockito.doReturn(Optional.of(e)).when(employeeRepository).findByOIB("12345678900");

    Employee actual = adminService.getEmployeeByOIB("12345678900");

    assertThat(actual).isEqualTo(e);
}
```

Slika 5.6: Test: Dohvaćanje komunalnog radnika prema OIBu

**Očekivani rezultat:**

1. Komunalni radnik sa zadanim OIBom.

**Rezultat:**

Vraćen je komunalni radnik sa zadanim OIBom. - Test je uspješno proveden.

### 5.2.2 Ispitivanje sustava

Svi testovi izvršeni su unutar radnog okvira Selenium. Ispitivanje se radilo po obrascima uporabe kako bi se provjerila osnovna funkcionalnost sustava. Svaki dio sustava je ispitan, no zbog jednostavnosti u dokumentaciji će biti prikazan samo dio ispitivanja. Prikazivanje ispitivanja UC1, UC2, UC7 i UC15.

**Ispitni slučaj 1: Registracija u sustav****Ulaz:**

1. ime građanina

2. prezime građanina
3. email adresa
4. lozinka
5. ponovljena lozinka

```
// register form
driver.findElement(By.id("register")).click();

// credentials
driver.findElement(By.id("name")).sendKeys(...keysToSend: "Domagoj");
driver.findElement(By.id("lastname")).sendKeys(...keysToSend: "Marinello");
driver.findElement(By.id("email")).sendKeys(...keysToSend: "domagoj.marinello@fer.hr");
driver.findElement(By.id("pass")).sendKeys(...keysToSend: "password123");
driver.findElement(By.id("re_pass")).sendKeys(...keysToSend: "password123");

// checkbox
driver.findElement(By.xpath("//label[@for='agree-term']")).click();

// submit
driver.findElement(By.id("signup")).click();

// assert
String msg = driver.findElement(By.id("welcome-msg")).getAttribute(name: "innerHTML");
assert (msg.contains("Domagoj"));
```

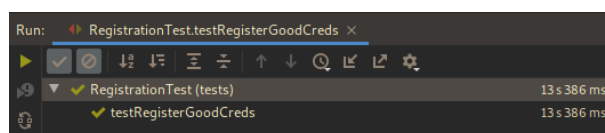
Slika 5.7: Ulazni podaci: Registracija u sustav

**Očekivani rezultat:**

1. Korisniku se prikazuje početna stranica sa kartom

**Rezultat:**

Korisniku se prikazala početna stranica sa kartom. - Test je uspješno proveden.



Slika 5.8: Rezultat: Registracija u sustav

**Ispitni slučaj 2: Prijava u sustav****Ulaz:**

1. email adresa
2. lozinka

```
// login credentials
driver.findElement(By.id("login-link")).click();

// login form
driver.findElement(By.id("email")).sendKeys(...keysToSend: "matea.vasilj@fer.hr");
driver.findElement(By.id("pass")).sendKeys(...keysToSend: "password");

// submit
driver.findElement(By.id("signin")).click();

// assert
String msg = driver.findElement(By.id("welcome-msg")).getAttribute(name: "innerHTML");
assert (msg.contains("Matea"));
```

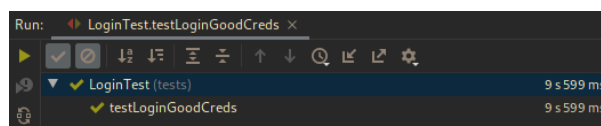
Slika 5.9: Ulazni podaci: Prijava u sustav

**Očekivani rezultat:**

1. Korisniku se prikazuje početna stranica sa kartom

**Rezultat:**

Korisniku se prikazala početna stranica sa kartom. - Test je uspješno proveden.



Slika 5.10: Rezultat: Prijava u sustav

**Ispitni slucaj 3: Brisanje susjedstva****Ulaz:**

1. email adresa admina
2. lozinka admina
3. ime kvarta koji želimo obrisati



```
// load admin page
driver.get("https://kombinacija.hopto.org/admin.html");

// select last neighborhood in list
driver.findElement(By.cssSelector("a[href*='kvartsubmenu']")).click();
driver.findElement(By.xpath("//a[@onclick='toggleVisibility('A2'); loadHoods();']")).click();
driver.findElement(By.xpath("//ul[@id='del-hood-list']/li[last()]")).click();

// delete selected
driver.findElement(By.xpath("//button[@onclick='doDeleteNeighborhood();']")).click();

Thread.sleep(1000);

// handle alert
Alert alert = driver.switchTo().alert();
String alertText = alert.getText();
alert.accept();

assert (alertText.contains("uspješno obrisano"));
```

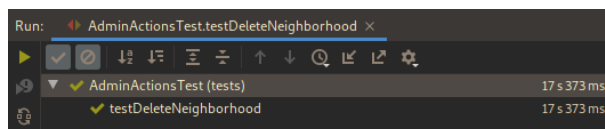
Slika 5.11: Ulazni podaci: Brisanje susjedstva

**Očekivani rezultat:**

1. Pojavljuje se obavijest da je kvart uspješno obrisano

**Rezultat:**

Pojavljuje se obavijest da je kvart uspješno obrisano. - Test je uspješno proveden.



Slika 5.12: Rezultat: Brisanje susjedstva

**Ispitni slučaj 4: Dodavanje komunalnog radnika****Ulaz:**

1. email adresa admina
2. lozinka admina
3. ime komunalnog radnika
4. prezime komunalnog radnika
5. oib komunalnog radnika
6. email adresa komunalnog radnika
7. lozinka komunalnog radnika
8. ponovljena lozinka komunalnog radnika

## 9. ime kvarta u kojeg želimo pridjeliti radnika

```
// load admin page
driver.findElement(By.id("go-admin")).click();

// open add employee form
driver.findElement(By.cssSelector("a[href*='pageSubmenu']")).click();
driver.findElement(By.xpath("//a[@onclick=\\"toggleVisibility('C1'); loadHoodsADD_E();\\"]")).click();

// fill form
driver.findElement(By.id("add-emp-new-name")).sendKeys(...keysToSend: "Leon");
driver.findElement(By.id("add-emp-new-lname")).sendKeys(...keysToSend: "Kranjcevic");
driver.findElement(By.id("add-emp-new-oib")).sendKeys(...keysToSend: "12312312355");
driver.findElement(By.id("add-emp-new-email")).sendKeys(...keysToSend: "leon.kranjcevic@gmail.com");
driver.findElement(By.id("add-emp-new-pass")).sendKeys(...keysToSend: "password");
driver.findElement(By.id("add-emp-new-pass2")).sendKeys(...keysToSend: "password");

// select neighborhood
Select dropdownNeighborhood = new Select(driver.findElement(By.id("add-emp-select")));
dropdownNeighborhood.selectByVisibleText("Maksimir");

// submit
driver.findElement(By.name("add-emp")).click();

Thread.sleep(1000);

// handle alert
Alert alert = driver.switchTo().alert();
String alertText = alert.getText();
alert.accept();

assert (alertText.contains("Komunalni radnik uspješno stvoren"));
```

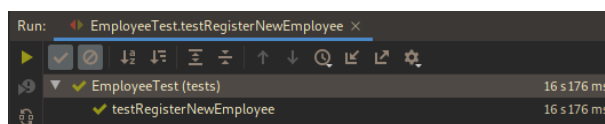
Slika 5.13: Ulazni podaci: Dodavanje komunalnog radnika

**Očekivani rezultat:**

1. Pojavljuje se obavijest da je komunalni radnik uspješno dodan u sustav

**Rezultat:**

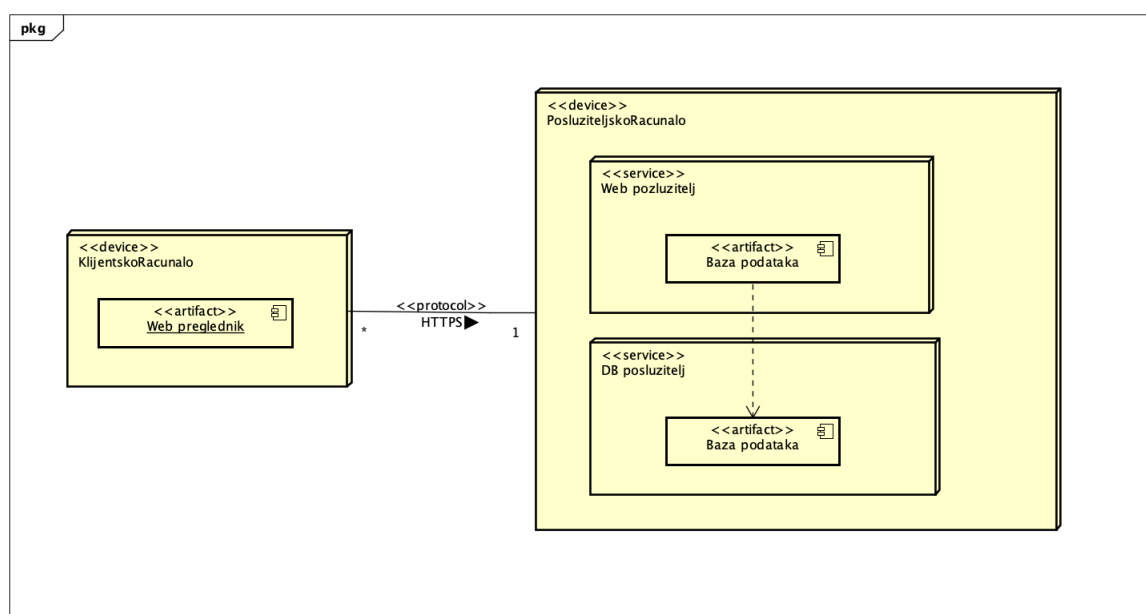
Pojavljuje se obavijest da je komunalni radnik uspješno dodan u sustav. - Test je uspješno proveden.



Slika 5.14: Rezultat: Dodavanje komunalnog radnika

## 5.3 Dijagram razmještaja

Dijagrami razmještaja opisuju topologiju sklopovlja i programsku potporu koja se koristi u implementaciji sustava u njegovom radnom okruženju. Na poslužiteljskom računalu se nalaze web poslužitelj i poslužitelj baze podataka. Klijenti koriste web preglednik kako bi pristupili web aplikaciji. Sustav je baziran na arhitekturi "klijent – poslužitelj", a komunikacija između računala korisnika (klijent, zaposlenik, vlasnik, administrator) i poslužitelja odvija se preko HTTPS veze.



Slika 5.15: Dijagram razmještaja

## 5.4 Upute za puštanje u pogon

### Mikroservisi i Docker

Sustav je razvijen idejom mikroservisne arhitekture. Svaki dio aplikacije je napravljen kao poseban servis koji može raditi na jednom ili na više različitih poslužitelja (i fizičkih i virtualnih). Ova tehnika omogućuje visoku skalabilnost, olakšan daljnji razvoj te testiranje. Jezgra aplikacije, točnije *backend* pisan u Javi, trenutno podsjeća na monolit, ali ako se u budućnosti ukaže potreba moguće ga je razlomiti na više manjih servisa.

Svaki servis je izoliran u poseban kontejner, a kontejner je virtualiziran na nivou OS-a. To je realizirano *Docker Engine-om*. Na taj način omogućeno je vrlo jednostavno puštanje u pogon, neovisno o sustavu na poslužitelju, dokle god je podržana virtualizacija.

Docker je široko dostupan na Windows i UNIX sustavima, a nalazi se u repozitorijima većine Linux distribucija. Podržan je i na Azure-u, OpenShift-u i AWS-u.

### Automatsko puštanje u pogon pomoću docker-compose datoteke

U mapama database, backend i frontend nalaze se konfiguracijske datoteke *Dockerfile*. U njima je definirano kako će svaki taj dio aplikacije biti napravljen u Docker image koji će kasnije biti pokrenut kao zaseban servis te koje parametre mu je potrebno predati za pokretanje.

Ako će se mijenjati domena aplikacije, onda je nužno u datoteci *frontend/Dockerfile* promijeniti *environment varijablu* DOMAIN\_NAME. Trenutno je stavljena na vrijednost „kombinacija.hopto.org“ što je ujedno i aktivna domena. Sve drugo nije potrebno mijenjati.

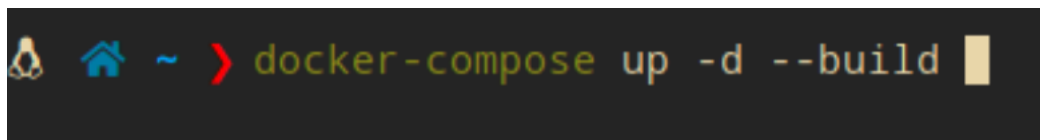
Očekuje se da SSL certifikati koji se misle koristiti već unaprijed postoje te da su spremljeni u mapu *frontend/ssl*. Također se očekuje da certifikat ima nastavak *.crt*, a ključ nastavak *.key*. Ime datoteka ni u jednom slučaju nije važno. *Dockerfile* će se pobrinuti da certifikati završe na pravom mjestu pri kreiranju *Docker image-a* i budu preimenovani kako odgovara *Docker web servisu*, odnosno *Nginx-u*. Moguće je samostalno izgenerirati certifikate, a kako bi taj korak bio sto lakši, napravljena je i skripta koja automatski generira certifikate (nalazi se u mapi *scripts/*). Skripta kreira certifikat koji ima valjanost od jedne godine, a ključeve sprema u mapu */frontend/ssl* kako bi ih kasnije *Docker* preuzeo.

Glavna datoteka je *docker-compose.yml* i nalazi se u *root mapi* projekta. Ona

spaja sve Dockerfile konfiguracijske datoteke, iz njih kreira *Docker image* te pokreće kao *Docker service*. Svakom servisu predaje potrebnu *env datoteku* s definiranim *environment varijablama* koje taj servis koristi. Brine se da konfiguracije poput korisnika i lozinke za bazu podataka završe samo kod onih servisa kojima je ta informacija potrebna. Također stvara virtualne mreže preko kojih svi servisi komuniciraju te tako dodatno izolira backend i bazu kojima nikako nije moguće pristupiti izvana. Stvara *Docker volume* za bazu podataka na koji se spremaju svi podatci. Vodi računa da su svi servisi aktivni, ako neki padne odmah ga ponovno vraća online. Omogućava skaliranje na više replika pojedinog servisa. Otvara portove 80 i 443 za web aplikaciju. Nikakvo dodatno konfiguriranje ni izmjene nisu potrebne.

Kako bi se cijeli sustav upalio treba izvršiti samo jednu komandu u terminalu:

```
$ docker-compose up -d --build
```



Slika 5.16: docker-compose up

CONTAINER ID	IMAGE
e6ad31bc3356	kombinacija_web
6eb9162b649e	kombinacija_app
9aea66dcd66e	dpage/pgadmin4
f0660203c217	kombinacija_database

Slika 5.17: docker ps

- Kreiranje baze, korisnika kao ni konfiguracija dozvola nije potrebna, sve od navedenog se obavlja automatski.
- Konfiguracija *pgAdmin* panela također nije potrebna, obavlja se automatski pri pokretanju.
- Mijenjanje varijabli u Java kodu i *properties* datotekama nije potrebno, obavlja se automatski. Kompajliranje izvornog Java koda nije potrebno, isto se obavlja automatizirano i pri svakom pokretanju provjerava ima li izmjena zbog kojih bi bilo potrebno ponovno kompajlirati kod. Izradu i kopiranje/premještanje JAR datoteke nije potrebno obavljati.

- Nikakva konfiguracija web servera (u ovom slučaju *Nginx*) nije potrebna. Sve datoteke nužne za postavke *Nginx* i posluživanje stranica se nalaze u */frontend/Nginx* mapi i automatski se kopiraju pri pokretanju na pravo mjesto. Sve postavke vezane za ime domene, SSL certifikata, URL-a za REST API i slične, se same konfiguriraju pri pokretanju aplikacije.

### Dodatna konfiguracija database servisa

*Database/Dockerfile* konfiguracijska datoteka preuzima najnoviju verziju *PostgreSQL* baze podataka. Kreira početnog korisnika „postgres“ i dodatnog korisnika „kombinacijauser“. Korisnik „kombinacijauser“ je jedini korisnik koji može raditi izmjene u bazi „kombinacijadb“. Sve navedene vrijednosti je moguće promijeniti u *env datotekama*. Nikakve druge izmjene nisu potrebne jer će svi ostali servisi uspješno primiti nove vrijednosti.

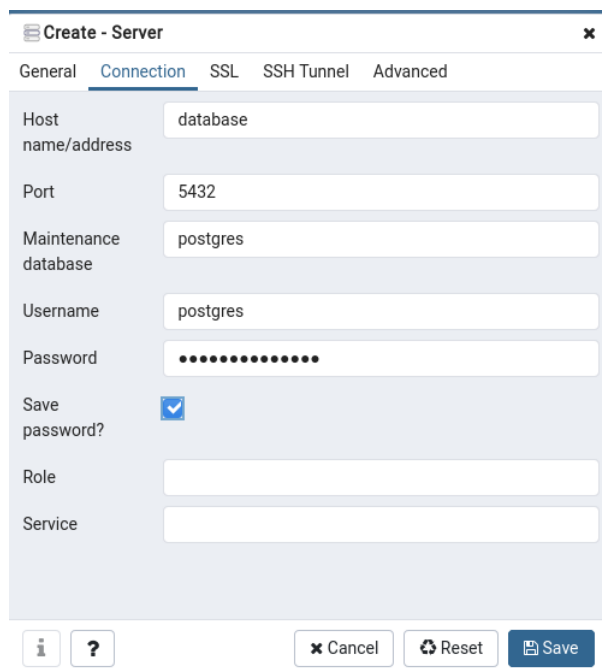
*Database servisu* nije moguće pristupiti izvana, izoliran je na unutarnjoj virtualnoj mreži, ali dijeli dodatnu mrežu s *pgAdminom*. Za slučaj da se to želi izbjeći potrebno je ukloniti tu mrežu (*pgadmin-net*) iz database servisa u *docker-compose.yml* datoteci.

```
services:
  database:
    container_name: database
    build:
      context: ./database
      dockerfile: Dockerfile
    env_file:
      - database/kombinacija.env
      - database/postgres.env
    networks:
      - backend-net
      - pgadmin-net
    restart: always
    volumes:
      - db-data:/var/lib/postgresql/data
```

Slika 5.18: database networks

Bazi je moguće pristupiti preko *pgAdmin servisa* koji je pokrenut na portu 5000. Podatci za autentifikaciju na *pgAdmin* se nalaze u *docker-compose.yml* datoteci. Podatci za pristup bazi su oni navedeni u *env datotekama*. Docker ima svoj DNS

sustav pa se kao *hostname* može koristiti ime servisa od baze podataka i interna IP adresa nije potrebna.



Slika 5.19: pgadmin connect to database

### Dodatna konfiguracija app servisa

*Backend/Dockerfile* konfiguracijska datoteka preuzima maven 3.6.2 i prevodi Java kod u JAR datoteku. Nakon toga kopira JAR paket u novi servis koji se naziva *app servis*, a *maven servis* odbacuje jer više nije potreban. Podatke za spajanje na bazu čita iz *environment varijabli* koje servisu predaje *docker-compose.yml*, a čita se iz *env datoteka* spremljenih u mapi *database*. U slučaju da su ti podatci prethodno promijenjeni, nije potrebno raditi nikakve izmjene kako u samom izvornom kodu, tako ni u samim konfiguracijskim datotekama jer će novi biti pročitani pri svakom ponovnom pokretanju.

```
# production database
spring.datasource.url=jdbc:postgresql://${DB_SERVER}/${DB_NAME}
spring.datasource.username=${DB_USER}
spring.datasource.password=${DB_PASS}
```

Slika 5.20: env app

Za dodatno konfiguriranje Spring-a na serveru potrebno je promijeniti *application-prod.properties* datoteku jer se ta prenosi u *app servis* i koristi u produkciji, a

obična `application.properties` datoteka se ignorira. Iako se Java aplikacija pokreće na portu `8080`, taj port nikada neće biti javno dostupan na serveru nego izoliran u Dockeru, a njemu će se pristupati samo unutar sustava preko web servisa koji ima definiran *proxy* specijalno za tu ulogu.

Pri prvom pokretanju svi članovi tima se dodaju u bazu kao administratori.

### Dodatna konfiguracija web servisa

*Frontend/Dockerfile* konfiguracijska datoteka preuzima najnoviju verziju *Nginx-a*, kopira mapu *frontend/public* u *Nginx* poslužiteljsku mapu kako bi HTML stranice bile javno dostupne, kopira napravljenu *Nginx konfiguraciju*, kopira SSL ključeve, namješta domenu servera koju čita iz *env variable* (ovu varijablu je potrebno promijeniti), otvara portove `80` i `443` na kojima web poslužitelj sluša te time stvara *Docker web servis* na koji se primaju zahtjevi.

*Nginx* konfiguracija se nalazi u datoteci *frontend/Nginx/kombinacija.conf*. U njoj je na početku definiran *load balancer* koji vodi na *app servis (backend)*. U *load balanceru* se trenutno nalazi samo jedan redak jer postoji samo jedna replika *app servisa*. *RR algoritam* je zamijenjen preusmjeravanjem na servis s najmanje aktivnih konekcija.

```
# backend load balancer
#
upstream kombinacijaAPI {
    least_conn;
    server app:8080    max_fails=2 fail_timeout=10s;
}
```

Slika 5.21: load-balancer

Za *Nginx* su napravljena pravila za cache kako bi se uštedio bandwidth na poslužitelju i smanjila veličina podataka koja se prenosi. Datoteke se čuvaju u web pregledniku klijenata više dana, ali se aktivno provjerava odgovaraju li onima na poslužitelju kako nijedan klijent ne bi imao zastarjelu verziju web stranice.

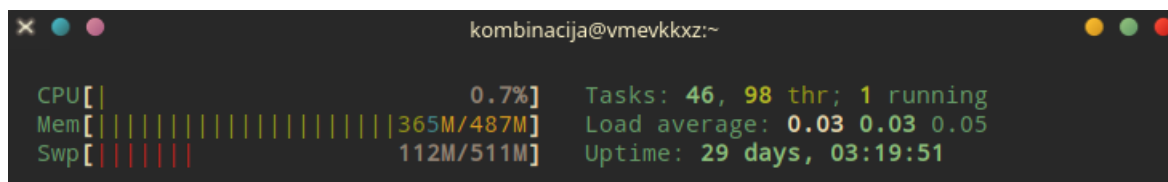
Naveden je put do SSL certifikata i pravila koja preusmjeravaju korisnike s HTTP-a na HTTPS. Putanju do certifikata je potrebno promijeniti ukoliko bi se mijenjao certifikat i lokacija istoga.

### MINIMALNI ZAHTEJEVI ZA POKRETANJE

Aplikacija je pokrenuta na VPS poslužitelju na kojem je instaliran Cent OS 8



Linux. Poslužitelj ima 500 MB RAM memorije, 500 MB SWAP memorije, 5 GB SSD memorije i jednu virtualnu jezgru Intel Xeon procesora (točan broj modela nije otkriven od strane hosting kompanije). To su ujedno i minimalni zahtjevi za pokretanje aplikacije i svih servisa o kojima ovisi. Važno je primijetiti da su to zahtjevi za pokretanje, a ne puštanje u pogon gdje se očekuje veći broj klijenata!



Slika 5.22: htop

## 6. Zaključak i budući rad

Zadatak naše grupe bio je razvoj web aplikacije za sustav odvoza otpada uz koju bi se građani mogli uključiti te dati svoj doprinos prijavljivanjem stanja kontejnera te također imaju mogućnost pregleda arhive stanja kontejnera. Aplikacija je integrirana sa postojećim komunalnim službama koje tim putem dobivaju obavijesti o punim kontejnerima, što im olakšava planiranje rute i fokusira pražnjenje kontejnera na mjesta na kojima je stvarno potrebno. Nakon 15 tjedana rada u timu i razvoja, ostvarili smo zadani cilj. Sama provedba projekta tekla je u dvije etape.

Prva etapa projekta uključivala je okupljanje tima za razvoj aplikacije, dodjelu projektnog zadatka, raspodjelu zadataka i intenzivan rad na dokumentiranju zahtjeva. Kvalitetna provedba prve faze uvelike je olakšala daljnji rad pri realizaciji osmišljenog sustava. Izrađeni obrasci i dijagrami (obraci uporabe, sekvencijski dijagrami, model baze podataka, dijagram razreda) bili su od pomoći podtimovima zaduženima za razvoj backenda i frontenda.

Druga etapa projekta, iako trajanjem neznatno kraća od prve, bila je daleko intenzivnija i zahtjevnija od prve što po pitanju samostalnog rada članova što po zahtjevnosti samih zadataka. Manjak iskustva članova u izradi sličnih implementacijskih rješenja primorao je članove na samostalno učenje odabranih alata i programskih jezika kako bi ispunili dogovorene ciljeve. Osim realizacije rješenja, u drugoj fazi je bilo potrebno dokumentirati ostale UML dijagrame i izraditi popratnu dokumentaciju kako bi budući korisnici mogli lakše koristiti ili vršiti preinake na sustavu. Dobro izrađen kostur projekta uštedio nam je mnogo vremena prilikom izrade aplikacije te smo izbjegli moguće pogreške u izradi koje bi bile vremenski skupe za ispravljanje u daljnjoj fazi projekta.

Komunikacija među članovima tima odvijala se putem Whatsapp kako bismo postigli što bolju i nadasve pravovremenu informiranost svih članova. Moguće proširenje postojeće inačice sustava svakako je izrada mobilne aplikacije čime bi cilj projektnog zadatka bio ostvaren u većoj mjeri, odnosno bio bi mnogo pristupačniji i praktičniji za korištenje no s web aplikacijom.

Sudjelovanje na ovom projektu bilo je iznimno vrijedno iskustvo zbog same prirode zadatka, odnosno odlične prilike za učenjem novih tehnologija, ali dakako

i zbog iskustva koje smo stekli intenzivnim timskim radom kroz ovih nekoliko tjedana. Projekt nas je naučio važnosti dobre vremenske raspodjele i koordiniranosti među članovima koji rade na projektu. Vidimo iznimno velik potencijal za unaprijeđivanje naše aplikacije i samog tima, ali obzirom na nedostatak vremena i iskustva veoma smo zadovoljni postignutim te svim stečenim znanjima i iskustvima.

# Popis literature

1. Oblikovanje programske potpore, FER ZEMRIS, <http://www.fer.hr/predmet/opp>
2. The Unified Modeling Language, <https://www.uml-diagrams.org/>
3. Astah Community, <http://astah.net/editions/uml-new>
4. Spring documentation, <https://spring.io/docs>
5. Baeldung, <https://www.baeldung.com/>
6. Docker, <https://www.docker.com/>
7. PostgreSQL, <https://www.postgresql.org/>
8. Nginx, <https://www.nginx.com/>
9. JavaScript documentation, <https://devdocs.io/javascript/>
10. Ajax, w3schools.com, [https://www.w3schools.com/xml/ajax\\_intro.asp](https://www.w3schools.com/xml/ajax_intro.asp)
11. Bootstrap, w3schools.com, <https://www.w3schools.com/bootstrap/>
12. Bootstrap documentation, <https://getbootstrap.com/docs/4.4/getting-started/introduction/>
13. IntelliJ IDEA documentation, <https://www.jetbrains.com/idea/documentation/>
14. Selenium documentation, <https://selenium.dev/documentation/en/>
15. Java tutorials, <https://www.javatpoint.com/>
16. RESTful Architecture, <https://dzone.com/refcardz/rest-foundations-restful?chapter=1>
17. Cross-Origin Resource Sharing, <https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS>

# Indeks slika i dijagrama

3.1	Dijagram obrasca uporabe, funkcionalnost administratora . . . . .	17
3.2	Dijagram obrasca uporabe, funkcionalnost komunalnog radnika . .	18
3.3	Dijagram obrasca uporabe, funkcionalnost građanina . . . . .	18
3.4	Dijagram obrasca uporabe, funkcionalnost neregistriranog korisnika	19
3.5	Sekvencijski dijagram za UC05 . . . . .	20
3.6	Sekvencijski dijagram za UC08 . . . . .	21
3.7	Sekvencijski dijagram za UC13 . . . . .	22
4.1	Arhitektura sustava . . . . .	24
4.2	MVC prikaz aplikacije . . . . .	25
4.3	Dijagram baze podataka . . . . .	31
4.4	Entitetsko-Relacijski model baze podataka . . . . .	31
4.5	Dijagram razreda koji predstavljaju model podataka . . . . .	33
4.6	Dijagram razreda paketa DAO . . . . .	34
4.7	Dijagram razreda paketa controller . . . . .	34
4.8	Dijagram razreda paketa service . . . . .	35
4.9	Dijagram stanja . . . . .	36
4.10	Dijagram aktivnosti . . . . .	37
4.11	Dijagram komponenti . . . . .	39
5.1	Test: Dohvaćanje omiljenih kontejnera . . . . .	42
5.2	Test: Podizanje reputacije građanina . . . . .	43
5.3	Test: Postavljanje reputacije korisnika . . . . .	43
5.4	Test: Dohvaćanje kontejnera preko identifikacijske oznake . . . . .	44
5.5	Test: Dohvaćanje kontejnera iz susjedstva . . . . .	44
5.6	Test: Dohvaćanje komunalnog radnika prema OIBu . . . . .	45
5.7	Ulazni podaci: Registracija u sustav . . . . .	46
5.8	Rezultat: Registracija u sustav . . . . .	46
5.9	Ulazni podaci: Prijava u sustav . . . . .	47
5.10	Rezultat: Prijava u sustav . . . . .	47
5.11	Ulazni podaci: Brisanje susjedstva . . . . .	48

5.12	Rezultat: Brisanje susjedstva . . . . .	48
5.13	Ulazni podaci: Dodavanje komunalnog radnika . . . . .	49
5.14	Rezultat: Dodavanje komunalnog radnika . . . . .	49
5.15	Dijagram razmještaja . . . . .	50
5.16	docker-compose up . . . . .	52
5.17	docker ps . . . . .	52
5.18	database networks . . . . .	53
5.19	pgadmin connect to database . . . . .	54
5.20	env app . . . . .	54
5.21	load-balancer . . . . .	55
5.22	htop . . . . .	56
6.1	Dijagrami pregleda promjena . . . . .	66

## Indeks tablica

1.1	Popis promjena dokumentacije . . . . .	4
4.1	Tablica <i>Person</i> . . . . .	28
4.2	Tablica <i>Admin</i> . . . . .	28
4.3	Tablica <i>Citizen</i> . . . . .	28
4.4	Tablica <i>Employee</i> . . . . .	28
4.5	Tablica <i>Container</i> . . . . .	29
4.6	Tablica <i>Neighborhood</i> . . . . .	29
4.7	Tablica <i>Favorites</i> . . . . .	30
4.8	Tablica <i>Ping</i> . . . . .	30
4.9	Tablica <i>Emptying</i> . . . . .	30

# Dodatak: Prikaz aktivnosti grupe

## Dnevnik sastajanja

### 1. sastanak

- Datum: 14. listopada 2019.
- Prisustvovali: M. Bićanić, H. Hrvoj, S. Skender, M. Tropčić, M. Turina, M. Vasilj
- Teme sastanka:
  - Diskusija o projektnom zadatku
  - Definiranje aktera u projektu
  - Definiranje osnovnog modela baze podataka
  - Rasprava o tehnologijama Spring, JPA
  - Općenita raspodjela studenata u front-end, back-end, dev-ops i dokumentacijske pod-timove

### 2. sastanak

- Datum: 21. listopada 2019.
- Prisustvovali: M. Bićanić, H. Hrvoj, S. Skender, M. Turina, M. Vasilj
- Teme sastanka:
  - Rasprava o dosad napravljenom dijelu projekta
  - Dodjela konkretnih zadataka pojedinim članovima
  - Dogovor o organizaciji same aplikacije
  - Rasprava o dokumentaciji projekta

### 3. sastanak

- Datum: 30. listopada 2019.
- Prisustvovali: M. Bićanić, S. Skender, M. Tropčić, M. Vasilj
- Teme sastanka:
  - Redefiniranje baze podataka
  - Rasprava o previđenim funkcionalnostima te dogovor oko implementacije
  - Određen i specificiran REST API aplikacije

## 4. sastanak

- Datum: 11. studenog, 2019.
- Prisustvovali: M. Bićanić, S. Skender, M. Tropčić, M. Vasilj, H. Hrvoj, M. Turina
- Teme sastanka:
  - Pregled prve verzije aplikacije i generičkih sposobnosti
  - Objašnjavanje dosadašnje implementacije (front end, back end...)

## 5. sastanak

- Datum: 10. prosinac 2019.
- Prisustvovali: M. Bićanić, H. Hrvoj, S. Skender, M. Tropčić, M. Turina, M. Vasilj
- Teme sastanka:
  - Rasprava o dosad napravljenom dijelu projekta
  - Rasprava o izgledu front-enda
  - Određeni završni zadaci na projektu

## 6. sastanak

- Datum: 3. siječnja 2020.
- Prisustvovali: M. Bićanić, H. Hrvoj, S. Skender, M. Tropčić, M. Turina, M. Vasilj
- Teme sastanka:
  - Pregled dokumentacije
  - Testiranje sustava



## Tablica aktivnosti

### Kontinuirano osvježavanje

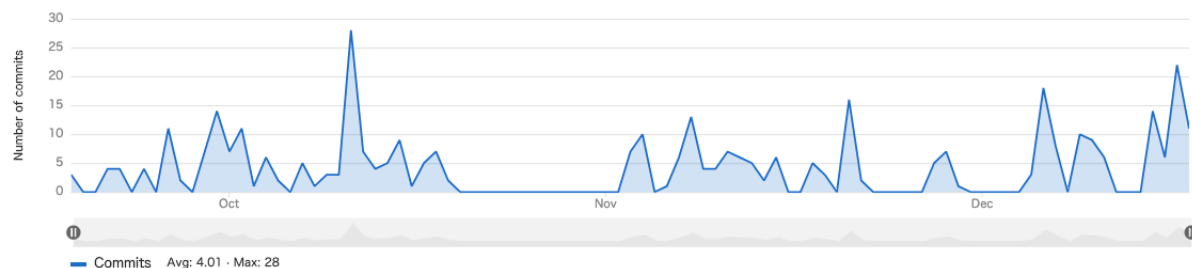
	Sven Skender	Miroslav Bićanić	Hrvoje Hrvoj	Mario Tropčić	Marko Turina	Matea Vasilj
Upravljanje projektom	15	10				10
Opis projektnog zadatka		3				
Funkcionalni zahtjevi						10
Opis pojedinih obrazaca						10
Dijagram obrazaca						5
Sekvencijski dijagrami						6
Opis ostalih zahtjeva						2
Arhitektura i dizajn sustava	2	4				2
Baza podataka	4	4				
Dijagram razreda		3				
Dijagram stanja						2
Dijagram aktivnosti						2
Dijagram komponenti		2				
Korištene tehnologije i alati	1	2				2
Ispitivanje programskog rješenja	5	1			5	4
Dijagram razmještaja						1
Upute za puštanje u pogon	5					3
Dnevnik sastajanja		1				
Zaključak i budući rad						3
Popis literature		1				
Izrada baze podataka	4	4		4		1
<i>front end</i>		40	25			
Produkcija	20					
<i>back end</i>	40	45		35	10	

	Sven Skender	Miroslav Bićanić	Hrvoje Hrvoj	Mario Tropčić	Marko Turina	Matea Vasilj
Spajanje s bazom podataka	3					

## Dijagrami pregleda promjena

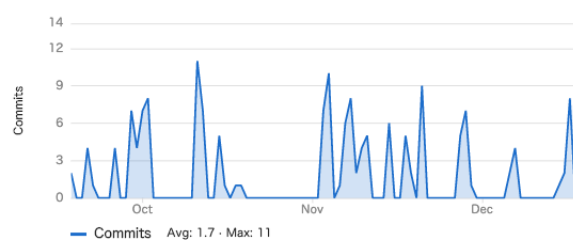
### Commits to master

Excluding merge commits. Limited to 6,000 commits.



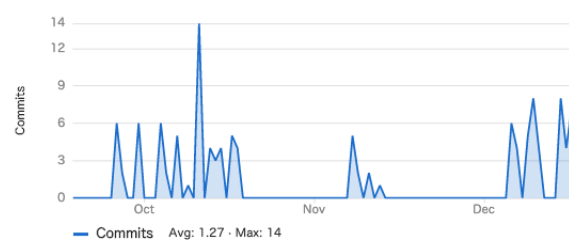
### sskender

158 commits (sven.skender@fer.hr)



### mbicanic

118 commits (miroslav.bicanic@fer.hr)



### Matea Vasilj

55 commits (vasiljmatea@gmail.com)



### mtropcic

26 commits (mario.tropcic@gmail.com)



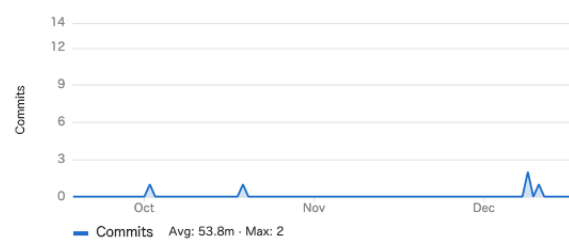
### mturina

6 commits (marko.turina@fer.hr)



### Hrvoje Hrvoj

5 commits (hrvojhrvoje@gmail.com)



Slika 6.1: Dijagrami pregleda promjena