

Oblikovanje programske potpore

Ak. god. 2019./2020.

Upravljanje kućnim otpadom

Dokumentacija, Rev. 2

Grupa: *Kolege*

Voditelj: *Stjepan Kovačić*

Datum predaje: *16. 01. 2020.*

Nastavnik: *Doc. dr. sc. Alan Jović*

Contents

1	Dnevnik promjena dokumentacije	3
2	Opis projektnog zadatka	5
3	Specifikacija programske potpore	9
3.1	Funkcionalni zahtjevi	9
3.1.1	Obrasci uporabe	11
3.1.2	Sekvencijski dijagrami	21
3.2	Ostali zahtjevi	25
4	Arhitektura i dizajn sustava	26
4.1	Stil arhitekture	26
4.1.1	Backend	29
4.1.2	Frontend	29
4.2	Baza podataka	30
4.2.1	Opis tablica	30
4.2.2	Dijagram baze podataka	36
4.3	Dijagram razreda	37
4.4	Implementacijski dijagram razreda	38
4.4.1	Sloj domene	38
4.4.2	Sloj za pristup podacima	39
4.4.3	Sloj usluge	40
4.4.4	Sloj nadglednika	41
4.4.5	DTO razredi	42
4.4.6	REST dijagram (Controller + DTO + Service)	42
4.5	Dijagram stanja	43
4.6	Dijagram aktivnosti	45
4.7	Dijagram komponenti	46
5	Implementacija i korisničko sučelje	47
5.1	Korištene tehnologije i alati	47

5.2	Ispitivanje programskog rješenja	50
5.2.1	Ispitivanje komponenti	50
5.2.2	Ispitivanje sustava	53
5.3	Dijagram razmještaja	58
5.4	Upute za puštanje u pogon	59
5.4.1	Priprema baze podataka	59
5.4.2	Pokretanje backend aplikacije	60
5.4.3	Pokretanje frontend aplikacije	60
6	Zaključak i budući rad	61
6.1	Projektni zadatak	61
6.1.1	Prva faza	61
6.1.2	Druga faza	61
6.2	Opći dojmovi	62
	Indeks slika i dijagrama	65
	Dodatak: Prikaz aktivnosti grupe	66

1. Dnevnik promjena dokumentacije

Kontinuirano osvježavanje

Rev.	Opis promjene/dodatka	Autori	Datum
0.1	Napravljen predložak.	Mandić	22.10.2019.
0.2	Dodani opisi obrazaca uporabe i dijagrami	Mandić, Horvat	01.11.2019.
0.3	Dodani sekvencijski dijagrami	Mandić, Horvat	03.11.2019.
0.4	Dodana arhitektura programske potpore	Kovačić	04.11.2019.
0.5	Dodani funkcionalni zahtjevi. Dodani ostali zahtjevi.	Cirimotić	05.11.2019.
0.6	Ažurirani dosadašnji sastanci	Kovačić	05.11.2019.
0.7	Napravljen opis baze podataka i kreiran dijagram baze podataka	Glivarec	06.11.2019.
0.8	Opisi obrazaca uporabe	Mandić Horvat	01.11.2019.
0.9	Dodan dijagram razreda za prvu predaju	Ovčariček, Kovačić	13.11.2019.
0.10	Napravljen opis projektnog zadatka	Štambuk	11.11.2019.
0.11	Dodan zaključak	Kovačić	15.11.2019.
0.12	Ažurirana literatura	Kovačić	15.11.2019.
0.13	Ažurirani obrasci uporabe	Mandić	15.11.2019.
1.0	Konačna verzija za prvi ciklus	*	15.11.2019.
1.1	Dodane korištene tehnologije i alati	Kovačić	13.01.2020.
1.2	Dodane upute za puštanje u pogon	Kovačić	13.01.2020.
1.3	Dodani JUnit testovi	Ovčariček	14.01.2020.
1.4	Dodani dijagrami razmještaja i aktivnosti	Horvat	16.01.2020.
1.5	Dodan zaključak	Kovačić	16.01.2020.
1.6	Dodani dijagram komponenti	Mandić, Štambuk	16.01.2020.

Rev.	Opis promjene/dodatka	Autori	Datum
1.7	Dodana ispitivanja sustava	Mandić, Štambuk	16.01.2020.
1.8	Dodani implementacijski dijagrami razreda	Kovačić	16.01.2020.
1.9	Dodani dijagrami stanja	Glivarec	16.01.2020.

2. Opis projektnog zadatka

Cilj ovog projekta jest stvoriti web aplikaciju kojom korisnik može lakše upravljati kućnim otpadom. Razvrstavanje i odlaganje različitih vrsta otpada postalo je zahtjevno pa bi aplikacija olakšala taj posao informiranjem koja vrsta otpada pripada kojem odlagalištu. Osim toga, aplikacija nudi mogućnost slanja zahtjeva za odvoz glomaznog otpada i slanja zahtjeva za resurse prikupljanja otpada. U slučaju da su građani nezadovoljni uslugom zbrinjavanja otpada, postoji mogućnost slanja pritužbi. Ovakva aplikacija odlično je rješenje za automatizaciju cijelog procesa zbrinjavanja otpada.

Trenutačno postoji web aplikacija Čistoće <https://www.cistoca.hr/> koja informira građane gdje bi mogli odložiti otpad i u kojim terminima te se na tu adresu mogu slati zahtjevi za odvoz glomaznog otpada.



Figure 2.1: Karta odlagališta na stranici <https://www.cistoca.hr/>

e-Zahtjev za odvoz glomaznog otpada

Čistoća > Info centar > e-Zahtjev za odvoz glomaznog otpada

Upute za popunjavanje zahtjeva pročitajte [OVDJE](#)

Serijski broj zahtjeva *

Sistemska broj objekta: *

[Kliknite za preuzimanje podataka iz sustava](#)

Sistemska broj obveznika: *

Vrsta prostora:

Figure 2.2: Slanje zahtjeva za odvoz glomaznog otpada s <https://www.cistoca.hr>

Ipak, postoji potreba za novom aplikacijom jer trenutačna nedovoljno informira o razvrstavanju otpada te se ne može izravno poslati pritužba Čistoći. Prednosti nove aplikacije su i dobra preglednost te jednostavnost njezina korištenja.

Aplikaciju smiju koristiti registrirani građani (dalje: građani), zaposlenici poduzeća koje zbrinjava otpad (dalje: zaposlenici) i administratori. Građanima treba omogućiti registraciju, pri čemu je potrebno navesti:

- ime
- prezime
- adresu stanovanja (ulica, broj i mjesto)
- adresu e-pošte
- korisničko ime
- lozinku

Zaposlenici se registriraju tako što navode:

- korisničko ime
- lozinku

Administratori se evidentiraju izravno u bazi podataka.

Glavni zadatak aplikacije je informiranje građana oko svega što je vezano uz upravljanje kućnim otpadom. Podatke za informiranje građana ažuriraju zaposlenici i administratori te će biti dostupni svim registriranim građanima. Osim toga, aplikacija treba omogućiti slanje zahtjeva građana za dodatnim resursima za prikupljanje kućnog otpada, zahtjeva za odvozom krupnog otpada i slanje pritužbi.

Informiranje građana uključuje sljedeće funkcionalnosti:

1. odabir naziva ili kategorije proizvoda koji se razvrstava, za koje se navodi tip resursa za dozvoljeno odlaganje (određeni tip vrećice, tip kante ili kontejnera, uz grafički prikaz)
2. dostupnost odlagališta otpada u blizini mjesta stanovanja, uz otvaranje Google Mapsa s odgovarajućim lokacijama i opisom tih lokacija (vrste podržanog otpada, radno vrijeme)
3. informacije o terminima odvoza kućnog otpada za određenu lokaciju
4. informacije o poduzeću koje zbrinjava otpad (uključujući kontakt-informacije)

Informacije će biti specifične za grad u kojem se zbrinjava otpad te će administrator unositi te informacije i zaposlenici će moći mijenjati neke informacije.

Prilikom slanja zahtjeva građani na aplikaciji moraju imati preglednik vrsta resursa i upis komada odabranog resursa za odlaganje, uz navedeno obrazloženje. Slanje pritužbi treba uključivati detaljni opis pritužbe. Nakon slanja pritužbe ili zahtjeva, potrebno je omogućiti ispis svih poslanih informacija u PDF-formatu i iskrcavanje PDF-a na lokalno računalo. Osim funkcionalnosti printanja zahtjeva i pritužbi, građani će moći vidjeti povijest odgovora na svoje zahtjeve i pritužbe koje su im poslali zaposlenici.

Osim građana postoje još dvije vrste korisnika, a to su:

- zaposlenik
- administrator

Zadaća zaposlenika:

- pregledavanje zahtjeva i pritužbi građana te odgovaranje na upite
- mijenjanje informacija

Zaposlenicima će biti prikazan popis zahtjeva i pritužbi poredan po vremenu kako bi na njih lakše odgovarali. Nakon što se jednom odgovori na pritužbu ili zahtjev, ta pritužba ili zahtjev više neće biti prikazana korisniku.

Zadaća administratora:

- smiju mijenjati sve informacije
- brisanje bilo kojeg korisničkog računa

Aplikacija bi se mogla unaprijediti pregledom računa za odvoz smeća te prikazom stanja duga. Pod informacije bi se mogli staviti cjenik usluga zbrinjavanja otpada i materijali za informiranje građana zbog čega je važno razvrstavati otpad. Još jedna ideja bi bila dodavanje slanja zahtjeva za čišćenje privatnih površina.

3. Specifikacija programske potpore

3.1 Funkcionalni zahtjevi

Dionici:

1. Građani
2. Zaposlenici
3. Administratori

Aktori i njihovi funkcionalni zahtjevi:

1. Neregistrirani/neprijavljeni korisnik (inicijator) može:
 - (a) se registrirati u sustav, stvoriti novi korisnički račun kao građanin ili zaposlenik
 - i. građanima je potrebno ime, prezime, adresa(ulica, kućni broj, mjesto), e-mail adresa, korisničko ime i lozinka
 - ii. zaposlenima je potrebno korisničko ime i lozinka
 - (b) vidjeti informacije o uslugama
 - i. pretraživati naziv/kategoriju proizvoda
 - ii. popis dostupnih odlagališta s informacijama o odlagalištu (vrsta otpada, radno vrijeme)
 - iii. termine odvoza otpada za određenu lokaciju
 - iv. informacije o poduzeću koje zbrinjava otpad kao i njegove kontakt podatke
2. Građanin (inicijator) može:
 - (a) vidjeti informacije o uslugama
 - i. pretraživati naziv/kategoriju proizvoda
 - ii. popis dostupnih odlagališta u blizini mjesta stanovanja s informacijama o odlagalištu (vrsta otpada, radno vrijeme)
 - iii. termine odvoza otpada za određenu lokaciju
 - iv. informacije o poduzeću koje zbrinjava otpad kao i njegove kontakt podatke
 - (b) napraviti zahtjeve prema poduzeću

- i. izbor vrste i broj komada resursa za odlaganje(vrećice, kante, kontejneri) – potrebno obrazloženje uz zahtjev
- ii. za odvozom krupnoga otpada - obrazloženje uz zahtjev
- iii. ispis zahtjeva u pdf formatu
- (c) slati pritužbu poduzeću koja sadrži naslov i opis, koju može ispisati u pdf formatu
- (d) imati uvid u odgovor zaposlenika za svaki zahtjev i pritužbu ako postoji

3. Zaposlenik (inicijator) može:

- (a) odgovarati na pritužbe i zahtjeve građanina
- (b) brisati pritužbe i zahtjeve u suprotnosti s pravilima korištenja aplikacije
- (c) izmijeniti informacije o:
 - i. prikladnim resursima za određeni proizvod
 - ii. odlagalištima
 - iii. terminima odvoza

4. Administrator (inicijator) može:

- (a) vidjeti popis svih registriranih korisnika i njihovih osobnih podataka
- (b) brisati korisnike
- (c) izmijeniti informacije o:
 - i. prikladnim resursima za određeni proizvod
 - ii. odlagalištima
 - iii. terminima odvoza

5. Baza podataka (sudionik):

- (a) pohranjuje sve:
 - i. podatke o korisnicima i zaposlenicima te njihovim ovlastima
 - ii. pritužbe i zahtjeve korisnika te odgovore na iste
 - iii. podatke o odlagalištima, terminima odvoza i resursima

3.1.1 Obrasci uporabe

Opis obrazaca uporabe

UC1 -Registracija građana

- **Glavni sudionik:** Građanin
- **Cilj:** Stvoriti korisnički račun za pristup sustavu
- **Sudionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
 1. Građanin odabire opciju za registraciju
 2. Građanin unosi ime, prezime, adresu stanovanja, adresu e-pošte, korisničko ime i lozinku
 3. Građanin prima obavijest o uspješnoj registraciji
- **Opis mogućih odstupanja:**
 - 2.a Odabir postojećeg korisničkog imena ili e-maila, unos korisničkih podataka u nevaljalom formatu ili unos nepostojećeg emaila
 1. Sustav obavještava korisnika o neuspjeloj registraciji i vraća ga na stranicu za registraciju. Građanin mijenja nevaljale podatke ili odustaje od registracije.

UC2 -Prijava u sustav

- **Glavni sudionik:** Registrirani korisnik
- **Cilj:** Dobiti pristup korisničkom sučelju
- **Sudionici:** Baza podataka
- **Preduvjet:** Registracija
- **Opis osnovnog tijeka:**
 1. Korisnik unosi korisničko ime i lozinku
 2. Potvrda o ispravnosti unesenih podataka
 3. Pristup korisničkim funkcijama
- **Opis mogućih odstupanja:**
 - 2.a Unos nepostojećeg korisničkog imena/lozinke
 1. Sustav obavještava korisnika o neuspjelom upisu i vraća ga na stranicu za prijavu

UC3 -Pretraživanje otpada

- **Glavni sudionik:** Građanin

- **Cilj:** Pretraživanje naziva ili kategorije proizvoda koji se razvrstava, za koji se navodi tip resursa za dozvoljeno odlaganje
- **Sudionici:** Baza podataka
- **Preduvjet:** Građanin je prijavljen
- **Opis osnovnog tijeka:**
 1. Građanin odabire opciju "Vrste otpada"
 2. Građanin pretražuje otpad po imenu ili kategoriji
 3. Prikaže se tip resursa za dozvoljeno odlaganje (određeni tip vrećice, tip kante ili kontejnera) i slika istoga
- **Opis mogućih odstupanja:**
 - 2.a Za pretraživani naziv ili kategoriju proizvoda koji se razvrstava ne postoji informacija u sustavu
 1. Sustav obavještava građanina o nedostupnosti informacija za zatraženi naziv ili kategoriju

UC4 -Pregled obližnjih odlagališta

- **Glavni sudionik:** Građanin
- **Cilj:** Informiranje korisnika o dostupnim odlagalištima otpada u blizini njegove adrese stanovanja
- **Sudionici:** Baza podataka
- **Preduvjet:** Građanin je prijavljen
- **Opis osnovnog tijeka:**
 1. Građanin odabire opciju "Obližnja odlagališta"
 2. Građaninu se prikazuju obližnja odlagališta otpada
 3. Građanin odabire odlagalište koje ga zanima
 4. Otvara se Google Maps i lokacija odlagališta, te sve informacije o odlagalištu (vrste podržanog otpada, radno vrijeme)

UC5 -Pregled termina odvoza otpada

- **Glavni sudionik:** Građanin
- **Cilj:** Saznati termine odvoza otpada za određenu lokaciju
- **Sudionici:** Baza podataka
- **Preduvjet:** Građanin je prijavljen
- **Opis osnovnog tijeka:**
 1. Građanin odabire opciju "Termini odvoza"
 2. Građaninu se prikazuju termini odvoza kućnog otpada za njegovu adresu stanovanja

UC6 -Informacije o poduzeću za odvoz otpada

- **Glavni sudionik:** Građanin
- **Cilj:** Pregled informacija o poduzeću koje zbrinjava otpad
- **Sudionici:** Baza podataka
- **Preduvjet:** Građanin je prijavljen
- **Opis osnovnog tijeka:**
 1. Pregled informacija o poduzeću koje zbrinjava otpad koje uključuje kontakt-informacije

UC7 -Zahtjev za dodatnim resursima za odlaganje otpada

- **Glavni sudionik:** Građanin
- **Cilj:** Slanje zahtjeva za dodatnim resursima za prikupljanje kućnog otpada
- **Sudionici:** Baza podataka
- **Preduvjet:** Građanin je prijavljen
- **Opis osnovnog tijeka:**
 1. Građanin odabire opciju "Zahtjev za dodatnim resursima"
 2. Odabire resurs za odlaganje
 3. Piše obrazloženje zašto zahtijeva dodatne resurse
 4. Prima potvrdu o uspješno poslanom zahtjevu
 5. Građanin ima mogućnost iskrcavanja zahtjeva u obliku PDF-a na lokalno računalo
- **Opis mogućih odstupanja:**
 - 2.a Slanje zahtjeva bez obrazloženja
 1. Sustav upozorava građana da mora napisati obrazloženje
 2. Građanin piše obrazloženje ili odustaje od slanja zahtjeva

UC8 -Zahtjev za odvoz krupnog otpada

- **Glavni sudionik:** Građanin
- **Cilj:** Slanje zahtjeva za odvozom krupnog otpada
- **Sudionici:** Baza podataka
- **Preduvjet:** Građanin je prijavljen
- **Opis osnovnog tijeka:**
 1. Građanin odabire opciju "Zahtjev za odvoz krupnog otpada"
 2. Šalje zahtjev za odvoz krupnog otpada sa svoje adrese stanovanja
 3. Dobiva potvrdu o zaprimljenom zahtjevu i datum odvoza

4. Građanin ima mogućnost iskrćavanja zahtjeva u obliku PDF-a na lokalno računalo

UC9 -Stvaranje PDF-a

- **Glavni sudionik:** Građanin
- **Cilj:** Stvaranje PDF-a na temelju poslanog zahtjeva ili pritužbe
- **Sudionici:** Baza podataka
- **Preduvjet:** Zahtjev za odvoz krupnog otpada ili slanje pritužbe je poslan
- **Opis osnovnog tijeka:**
 1. Sustav zaprima zahtjev za stvaranje PDF-a na temelju poslanog zahtjeva ili pritužbe
 2. Sustav stvara PDF i vraća ga naručitelju
- **Opis mogućih odstupanja:**

UC10 -Slanje pritužbe

- **Glavni sudionik:** Građanin
- **Cilj:** Slanje pritužbe vezane za uslugu odvoza otpada i upravljanja otpadom
- **Sudionici:** Baza podataka
- **Preduvjet:** Građanin je prijavljen
- **Opis osnovnog tijeka:**
 1. Građanin odabire opciju "Pošalji pritužbu"
 2. Piše pritužbu i šalje ju
 3. Dobiva potvrdu o zaprimljenoj pritužbi
 4. Građanin ima opciju iskrćati pritužbu u formatu PDF-a na lokalno računalo
- **Opis mogućih odstupanja:**
 - 2.a Slanje prazne pritužbe
 1. Sustav obavještava korisnika o neuspjelom zaprimanju prazne pritužbe
 2. Građanin piše pritužbu ili odustaje od slanja pritužbe

UC11 -Registracija zaposlenika

- **Glavni sudionik:** Zaposlenik
- **Cilj:** Stvoriti račun zaposlenika za pristup sustavu
- **Sudionici:** Baza podataka
- **Preduvjet:** Zaposlenik je evidentiran u sustavu kao radnik
- **Opis osnovnog tijeka:**
 1. Zaposlenik odabire opciju za registraciju
 2. Zaposlenik unosi ime, prezime, adresu e-pošte, korisničko ime i lozinku
 3. Zaposlenik prima obavijest o uspješnoj registraciji
- **Opis mogućih odstupanja:**
 - 2.a Odabir postojećeg korisničkog imena ili e-maila, unos podataka u nevaljalom formatu ili unos nepostojećeg emaila
 1. Sustav obavještava korisnika o neuspjeloj registraciji i vraća ga na stranicu za registraciju
 2. Zaposlenik mijenja nevaljale podatke ili odustaje od registracije

UC12 -Odgovaranje na zahtjeve za dodatnim resursima

- **Glavni sudionik:** Zaposlenik
- **Cilj:** Obrada zahtjeva građana za dodatnim resursima
- **Sudionici:** Baza podataka
- **Preduvjet:** Zaposlenik je prijavljen i postoji barem jedan neobrađen zahtjev za dodatnim resursima
- **Opis osnovnog tijeka:**
 1. Zaposlenik odabire opciju "Zahtjevi za dodatnim resursima"
 2. Dobiva pregled svih neobrađenih zahtjeva i odabire jedan
 3. Na temelju obrazloženja odlučuje prihvaća li se zahtjev ili ne
 4. Odgovara građanu i zatvara zahtjev
- **Opis mogućih odstupanja:**

UC13 -Odgovaranje na pritužbe građana

- **Glavni sudionik:** Zaposlenik
- **Cilj:** Odgovoriti na pritužbe građana
- **Sudionici:** Baza podataka
- **Preduvjet:** Zaposlenik je prijavljen i postoji barem jedna neobrađena pritužba
- **Opis osnovnog tijeka:**

1. Zaposlenik odabire opciju "Pritužbe"
2. Dobiva pregled svih neobrađenih pritužbi i odabire jednu
3. Odgovara na pritužbu i zatvara pritužbu

UC14 -Izmjena informacija o resursima za zbrinjavanje otpada

- **Glavni sudionik:** Zaposlenik
- **Cilj:** Izmijeniti / dodati informacije o resursima za zbrinjavanje otpada
- **Sudionici:** Baza podataka
- **Preduvjet:** Zaposlenik je prijavljen i postoji resurs koji želi izmijeniti
- **Opis osnovnog tijeka:**
 1. Zaposlenik odabire opciju "Uređivanje informacija o resursima za odvoz"
 2. Dobiva pregled svih resursa za zbrinjavanje otpada i odabire jedan od njih
 3. Mijenja željene podatke i zaključava promjene
 4. Ako želi dodati novi resurs za zbrinjavanje otpada odabire opciju "Dodaj"
 5. Unosi sve potrebne podatke i zaključava unos
- **Opis mogućih odstupanja:**
 - 2.a Kod stvaranja novog resursa za zbrinjavanje otpada ne unosi sve obavezne podatke
 1. Sustav obavještava zaposlenika o neuspjelom unosu i vraća ga na stranicu za dodavanje resursa
 2. Zaposlenik unosi obavezne podatke ili odustaje od dodavanja novog resursa

UC15 -Izmjena lokacija odlagališta

- **Glavni sudionik:** Zaposlenik
- **Cilj:** Izmijeniti / dodati informacije o odlagalištima otpada
- **Sudionici:** Baza podataka
- **Preduvjet:** Zaposlenik je prijavljen i postoji lokacija koju želi izmijeniti
- **Opis osnovnog tijeka:**
 1. Zaposlenik odabire opciju "Uređivanje informacija o odlagalištima otpada"
 2. Dobiva pregled svih odlagališta i odabire jedno od njih
 3. Mijenja željene podatke i zaključava promjene
 4. Ako želi dodati novo odlagalište odabire opciju "Dodaj"
 5. Unosi sve potrebne podatke i zaključava unos
- **Opis mogućih odstupanja:**
 - 2.a Kod stvaranja novog odlagališta ne unosi sve obavezne podatke

1. Sustav obavještava zaposlenika o neuspjelom dodavanju novog odlagališta i vraća ga na stranicu za dodavanje odlagališta
2. Zaposlenik unosi obavezne podatke ili odustaje od dodavanja novog odlagališta

UC16 -Izmjena termina odvoza otpada

- **Glavni sudionik:** Zaposlenik
- **Cilj:** Izmijeniti / dodati termine odvoza otpada
- **Sudionici:** Baza podataka
- **Preduvjet:** Zaposlenik je prijavljen i postoji termin koji želi izmijeniti
- **Opis osnovnog tijeka:**
 1. Zaposlenik odabire opciju "Uređivanje termina za odvoz otpada"
 2. Dobiva pregled svih termina za odvoz i odabire jedan od njih
 3. Mijenja željene podatke i zaključava promjene
 4. Ako želi dodati novi termin odabire opciju "Dodaj"
 5. Unosi sve potrebne podatke i zaključava unos
- **Opis mogućih odstupanja:**
 - 2.a Kod stvaranja novog termina ne unosi sve obavezne podatke
 1. Sustav obavještava zaposlenika o neuspjelom unosu i vraća ga na stranicu za dodavanje termina
 2. Zaposlenik unosi obavezne podatke ili odustaje od dodavanja novog termina

UC17 -Pregled svih korisnika

- **Glavni sudionik:** Administrator
- **Cilj:** Pregled svih registriranih građana i zaposlenika
- **Sudionici:** Baza podataka
- **Preduvjet:** Administrator je prijavljen
- **Opis osnovnog tijeka:**
 1. Administrator odabire opciju "Pregled svih korisnika"
 2. Dobiva pregled svih registriranih građana i zaposlenika

UC18 -Brisanje korisnika

- **Glavni sudionik:** Administrator
- **Cilj:** Obrisati registrirani građanski račun ili zaposlenika
- **Sudionici:** Baza podataka
- **Preduvjet:** Administrator je prijavljen i postoji korisnički račun koji želi obrisati
- **Opis osnovnog tijeka:**
 1. Administrator odabire opciju "Obriši korisnika"
 2. Unosi korisničko ime korisnika kojega želi obrisati
 3. Otvara se korisnički račun i otvara se potvrda za brisanjem računa
 4. Administrator briše korisnika ili odustaje od brisanja
- **Opis mogućih odstupanja:**
 - 2.a Nepostojeće korisničko ime koje je administrator upisao
 1. Sustav obavještava administratora o nepostojećem korisniku

Dijagrami obrazaca uporabe

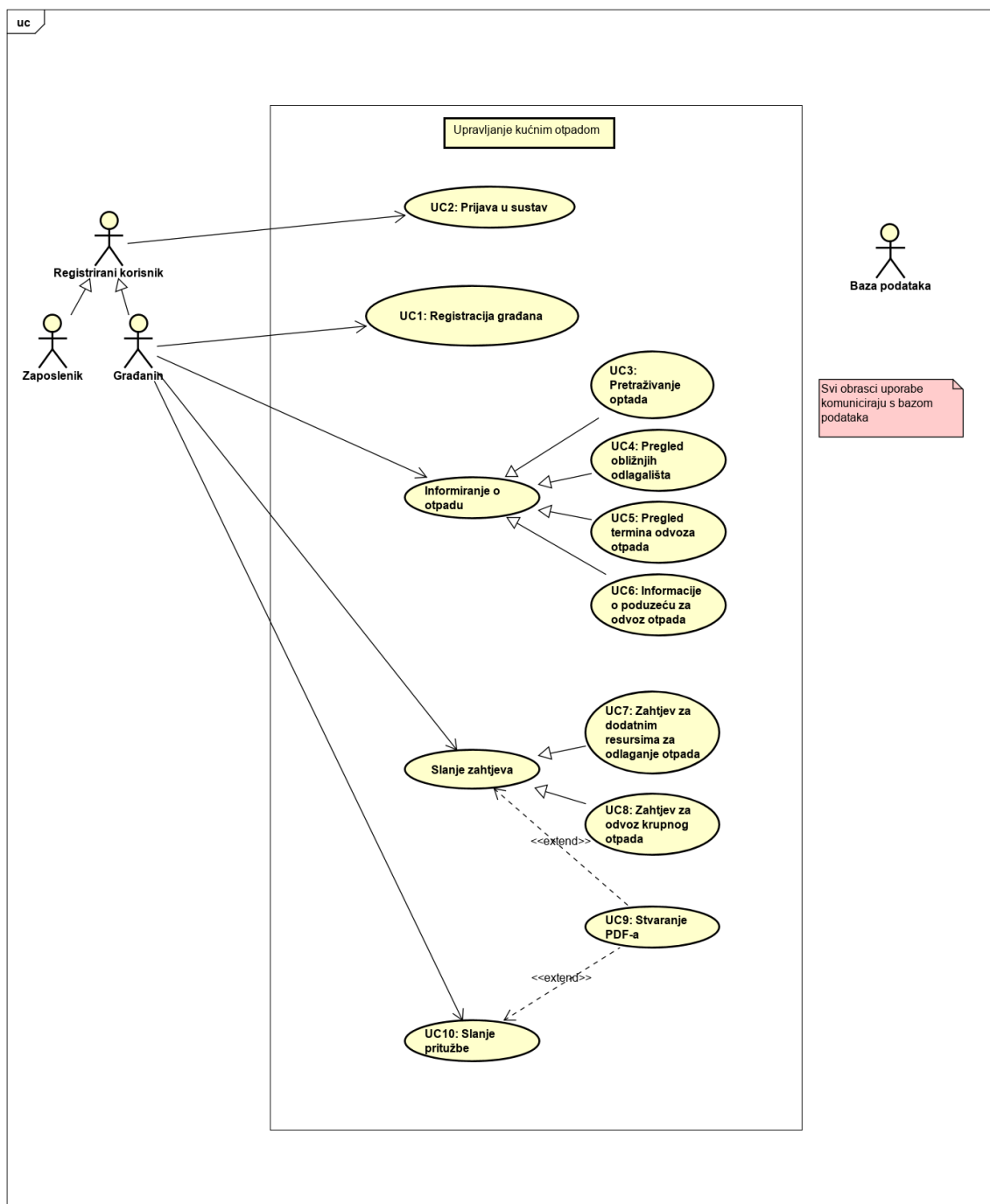


Figure 3.1: Dijagram obrasca uporabe, funkcionalnost građana i registriranog korisnika

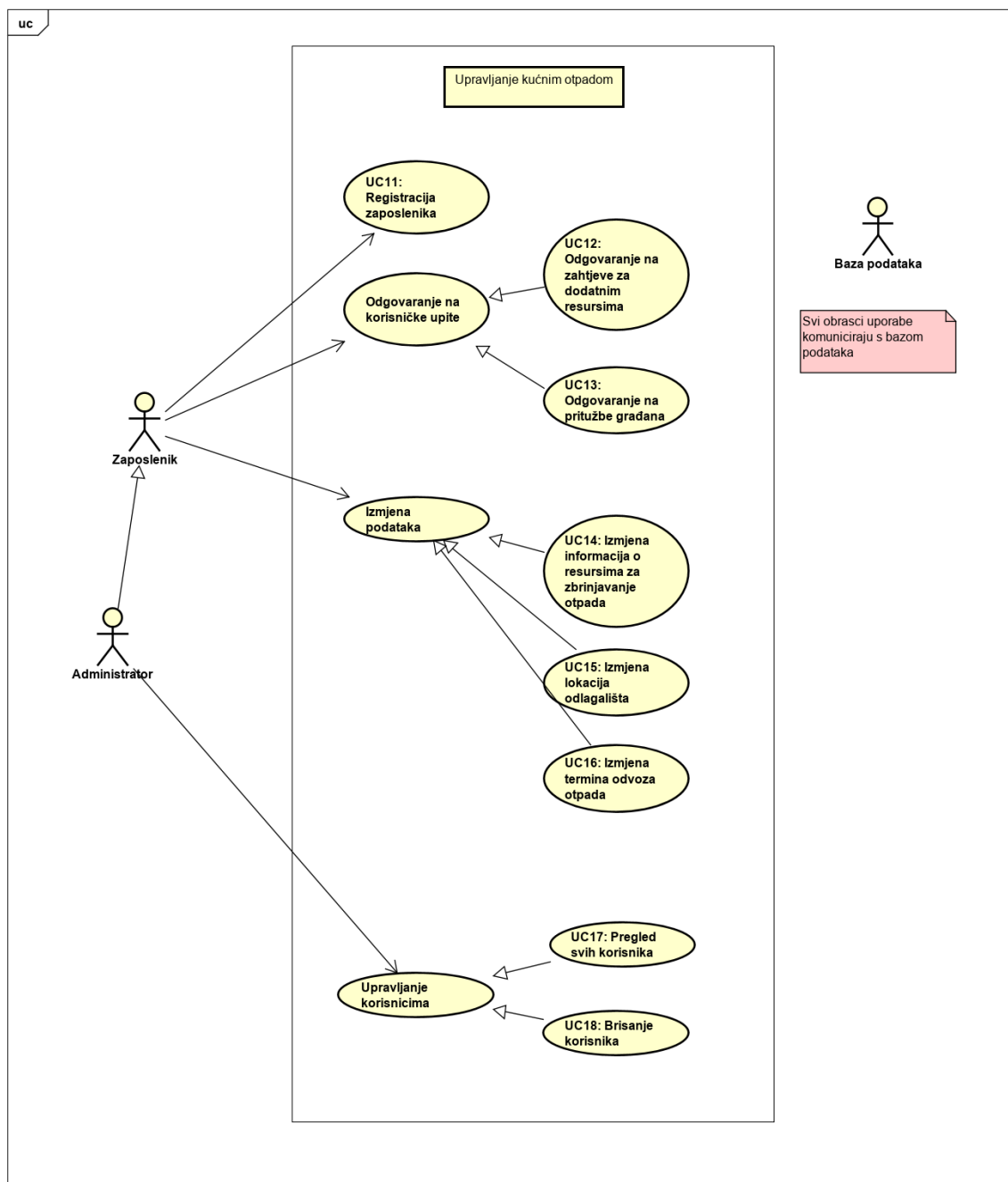


Figure 3.2: Dijagram obrasca uporabe, funkcionalnost zaposlenika i administratora

3.1.2 Sekvencijski dijagrami

Obrazac uporabe UC4 - Pregled obližnjih odlagališta

Klijent šalje zahtjev za pregled odlagališta u blizini svoje adrese stanovanja. Poslužitelj dohvaća najbliža odlagališta za otpad i prikazuje ih. Odabirom odlagališta, poslužitelj iz baze podataka dohvaća osnovne podatke o odlagalištu (vrsta otpada, radno vrijeme, kontakt...) i otvara lokaciju u Google Maps-u.

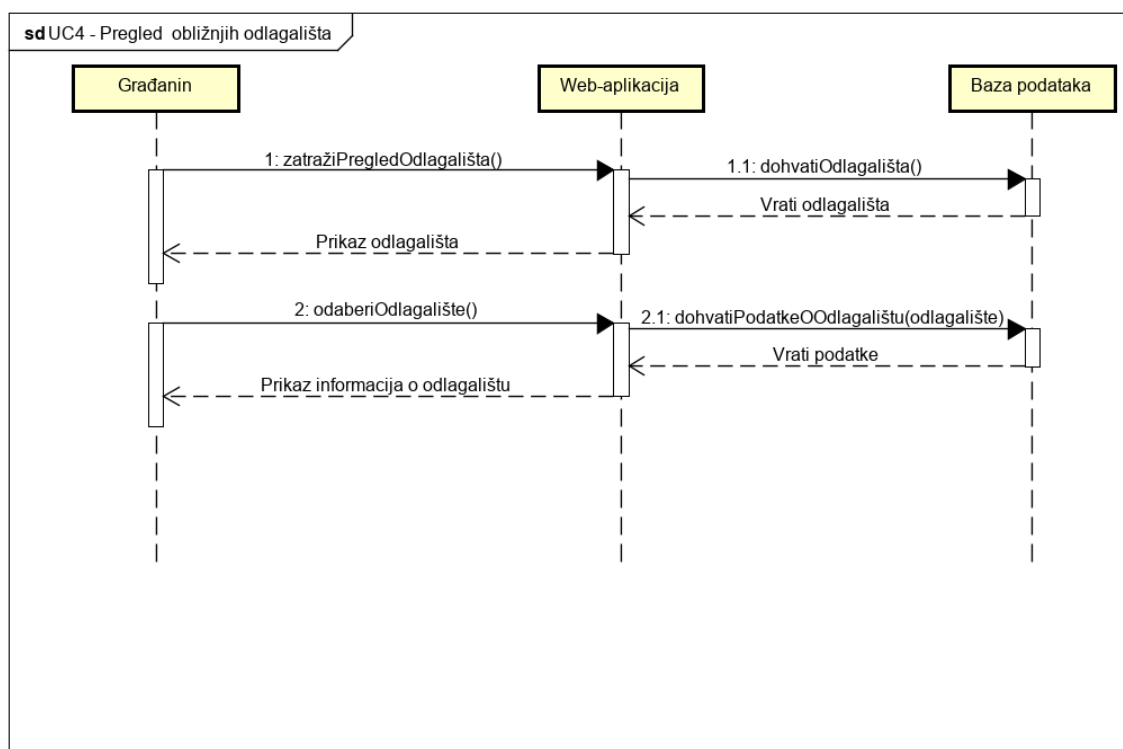


Figure 3.3: Sekvencijski dijagram za UC4

Obrazac uporabe UC18 - Brisanje korisnika

Administrator upisuje korisničko ime koje želi izbrisati. Poslužitelj dohvaća željenog korisnika i prikazuje sve korisničke podatke. Administrator tada potvrđuje svoju akciju brisanja. Ako ne postoji korisnik s upisanim korisničkim imenom poslužitelj o tome obavještava administratora.

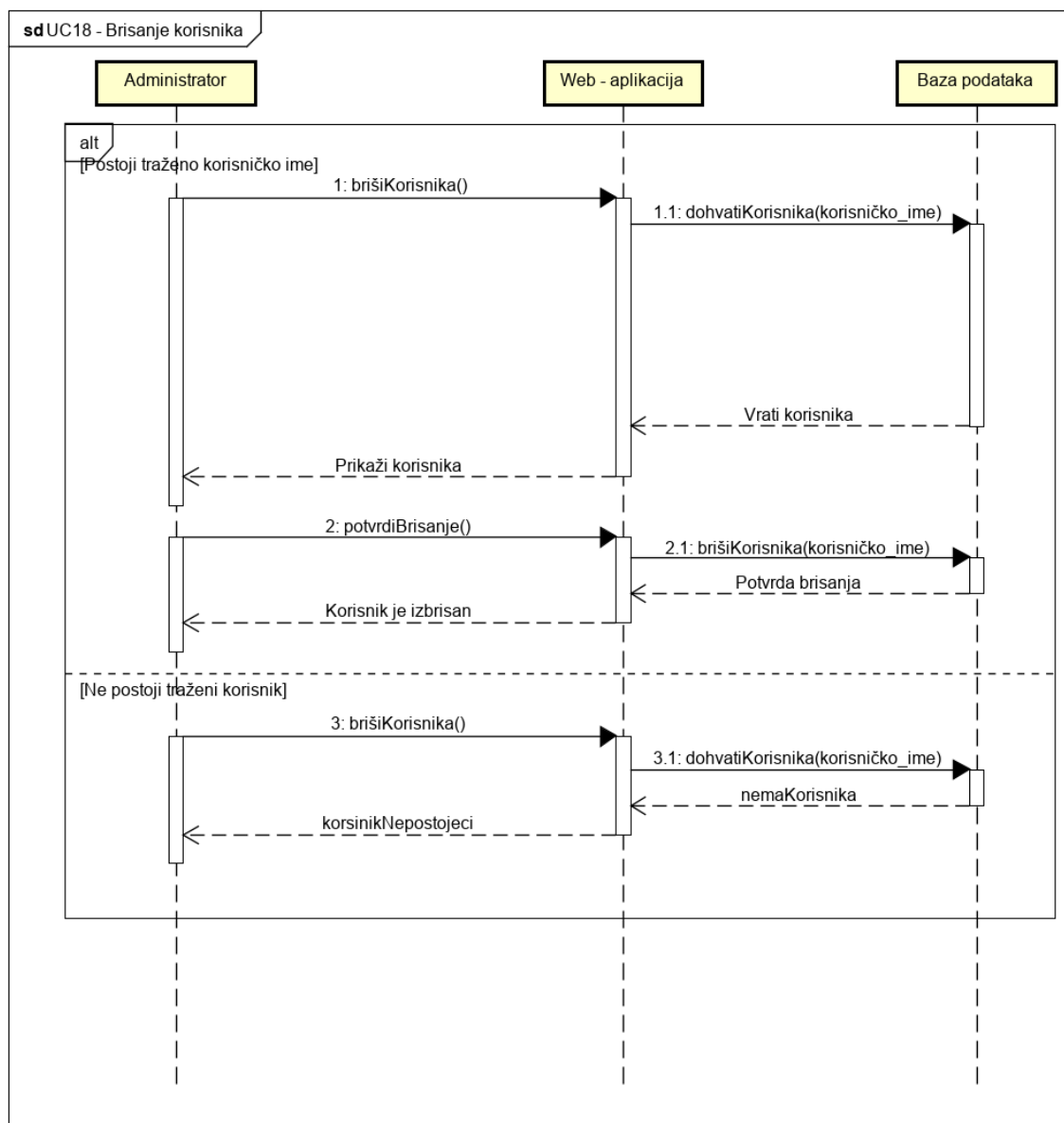


Figure 3.4: Sekvencijski dijagram za UC18

Obrazac uporabe UC13 - Odgovaranje na pritužbe građana

Zaposlenik šalje zahtjev za prikazom pritužba građana. Baza podataka vraća pritužbe te zaposlenik odabire jednu od njih na koju odgovara. Baza podataka ažurira zadnju promjenu koju je načinio zaposlenik, te građanin može vidjeti odgovor na svoju pritužbu.

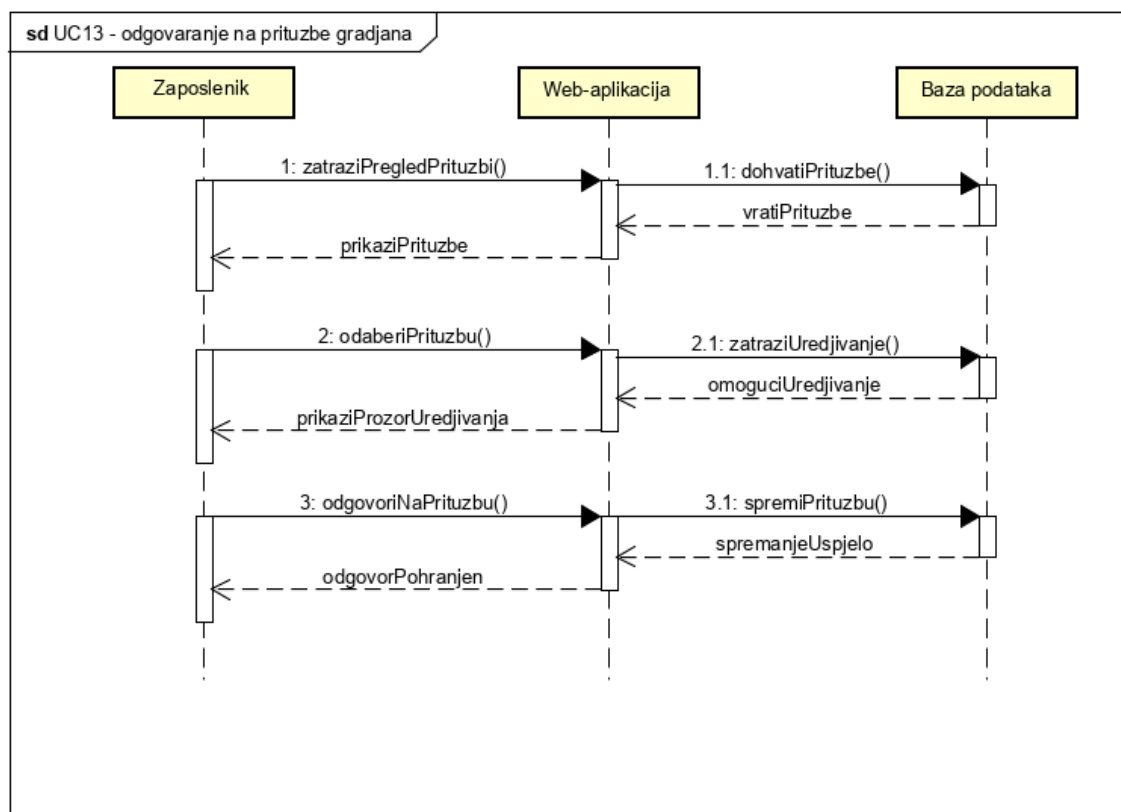


Figure 3.5: Sekvencijski dijagram za UC13

Obrazac uporabe UC16 - Izmjena termina odvoza

Zaposlenik odabire lokaciju za koju mu baza podataka daje informacije. Zatim, zaposlenik mijenja termin odvoza otpada za navedenu lokaciju što se ažurira u bazi podataka i postaje vidljivo svim korisnicima web-aplikacije.

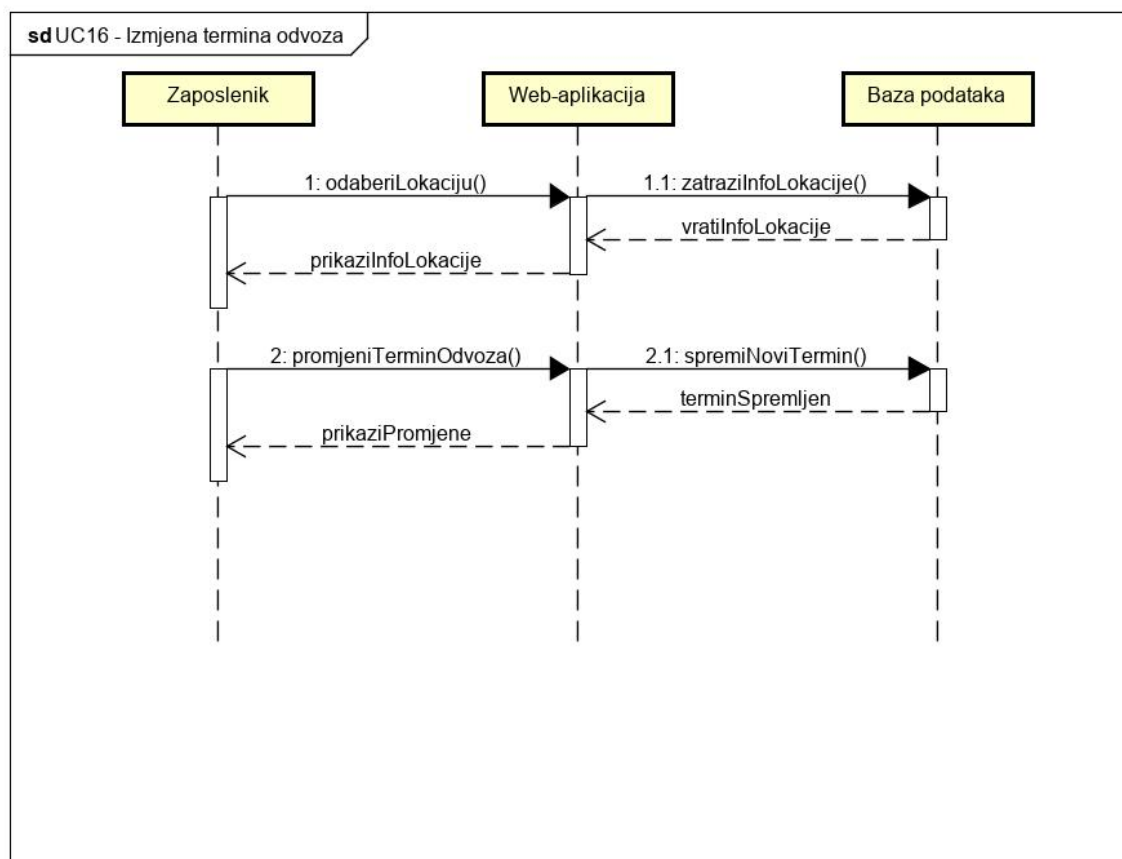


Figure 3.6: Sekvencijski dijagram za UC16

3.2 Ostali zahtjevi

- Informacije o uslugama tvrtke bit će ažurirane
- Sigurnost rada u sustavu osigurat će se unosom korisničkog imena i lozinke
- Ako dođe do pogreške na web stranici na to treba upozoriti korisnika porukom
- Svaki događaj u sustavu, čak i ako je neuspješan, treba korisniku dati povratnu poruku.
- Sustav treba omogućiti rad više korisnika u stvarnom vremenu
- Korisničko sučelje i sustav moraju podržavati hrvatsku abecedu(dijakritičke znakove) pri unosu i prikazu tekstualnoga sadržaja
- Izvršavanje dijela programa u kojem se pristupa bazi podataka ne smije trajati duže od nekoliko sekundi
- Sustav mora raditi na operativnom sustavu Windows 10
- zastoji u radu sustava ne smiju prijeći 5 sekundi dnevno
- Sustav treba biti implementiran kao web aplikacija koristeći objektno-orijentirane jezike
- Neispravno korištenje korisničkim sučelja ne smije narušiti funkcionalnost i rad sustava
- Sustav treba biti jednostavan za korištenje, korisnici se moraju znati koristiti sučelje bez opširnih uputa
- Nadogradnja sustava ne smije narušavati postojeće funkcionalnosti sustava
- Veza s bazom podataka mora biti kvalitetno zaštićena, brza i otporna na vanjske greške
- Pristup sustavu mora biti omogućen iz javne mreže pomoću HTTP.

4. Arhitektura i dizajn sustava

4.1 Stil arhitekture

Budući da je glavna ideja bila napraviti jednostraničnu aplikaciju (engl. *single page application, SPA*) i koristiti radni okvir Spring Boot, odlučili smo se za **višeslojnu arhitekturu** (engl. *multi-layer architecture*).

Višeslojna arhitektura je logička nadogradnja klijent-poslužitelj arhitekture (engl. *client-server architecture*).

Arhitektura **klijent-poslužitelj** se sastoji od dva potpuno odvojena sustava.

Klijent je računalo ili program koji dohvaća neke podatke. Te podatke mu pruža drugi program, kojeg zovemo **poslužitelj**. Poslužitelj ne pohranjuje nikakve podatke o klijentu, niti se nalazi u različitim stanjima (engl. *stateless*).

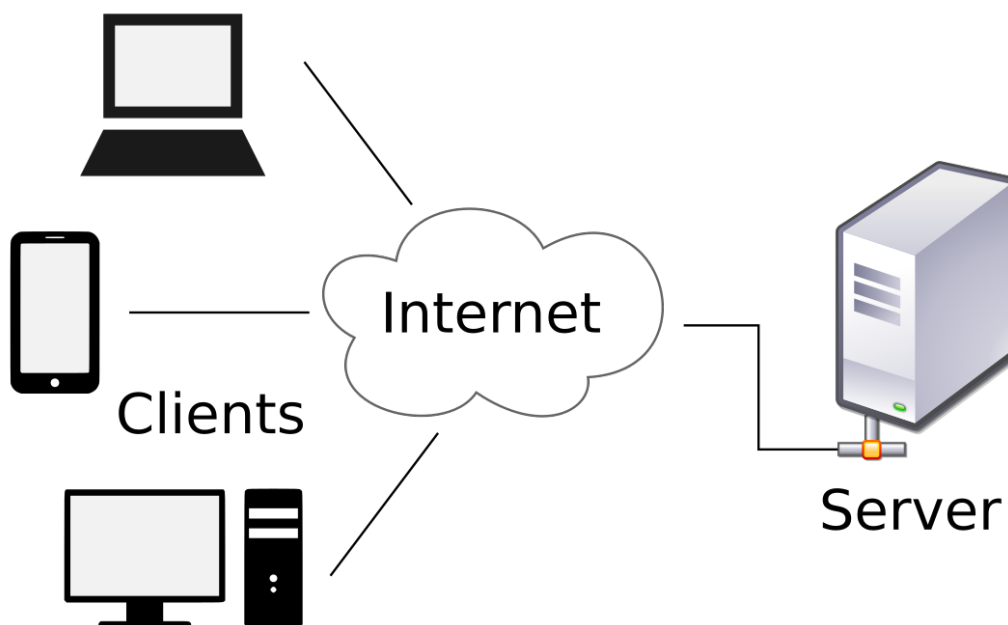


Figure 4.1: Klijent-poslužitelj arhitektura, Izvor: https://en.wikipedia.org/wiki/Client-server_model

Višeslojna arhitektura također koristi načelo razdvajanja klijentske i poslužiteljske strane, ali se programska potpora sastoji od više od dva logička sloja.

U našoj implementaciji, za koju ćemo koristiti radni okvir (engl. *framework*) **Spring Boot** i knjižnicu (engl. *library*) **React**, višeslojna arhitektura će imati 6 slojeva:

1. sloj korisničke strane

- Sloj koji je izravno vidljiv korisniku, preko njega korisnik koristi aplikaciju. U pozadini se koristi Javascript kojim se šalju zahtjevi i primaju odgovori u JSON formatu.
- Pripada klijentskoj strani.

2. sloj nadglednika (engl. *controller*)

- Prvi sloj na poslužiteljskoj strani.
- Prihvaća zahtjeve i poslovnu logiku prosljeđuje sloju usluge tj. servisima za obradu. Vraća na klijentsku stranu podatke koje sloj usluge pripremi za njega.

3. sloj usluge (engl. *service*)

- Sloj koji obavlja poslovnu logiku i u svojem radu često komunicira sa slojem za pristup podacima.

4. sloj za pristup podacima (engl. *data access object, DAO*)

- Sloj kojeg servisi(sloj usluge) koriste za komunikaciju s bazom podataka (dohvat, spremanje).

5. sloj domene (engl. *domain*)

- Model baze podataka predložen u programski kod, predstavlja podatke kojima DAO pristupa.

6. sloj baze podataka

- Sloj koji služi stvarnom spremanju u bazu podataka.
- Interna Spring Boot implementacija, nije dio implementacije našeg razvojnog tima.

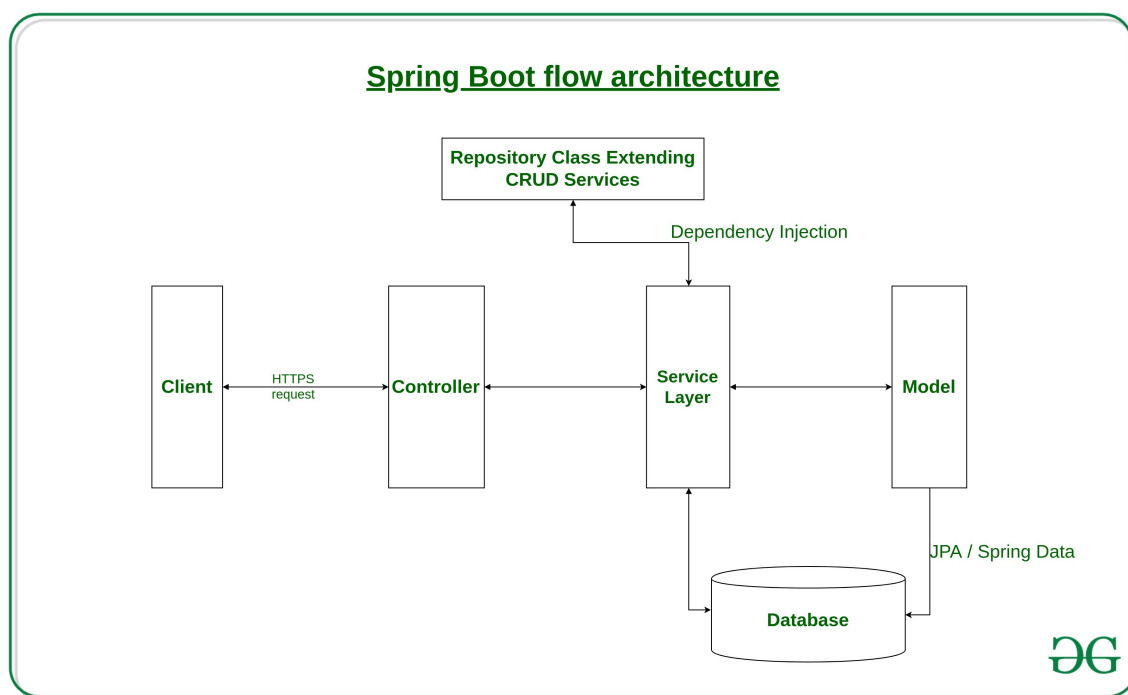


Figure 4.2: Spring Boot višeslojna arhitektura, Izvor:
<https://www.geeksforgeeks.org/introduction-to-spring-boot>

4.1.1 Backend

Za razvoj backend dijela aplikacije se koristi Java programski jezik s radnim okvirom **Spring Boot** sa svojim podsustavima (Spring Web, Spring Data, Spring Security). Spring Boot pruža odličnu podršku za razvoj višeslojne aplikacije.

More serious view

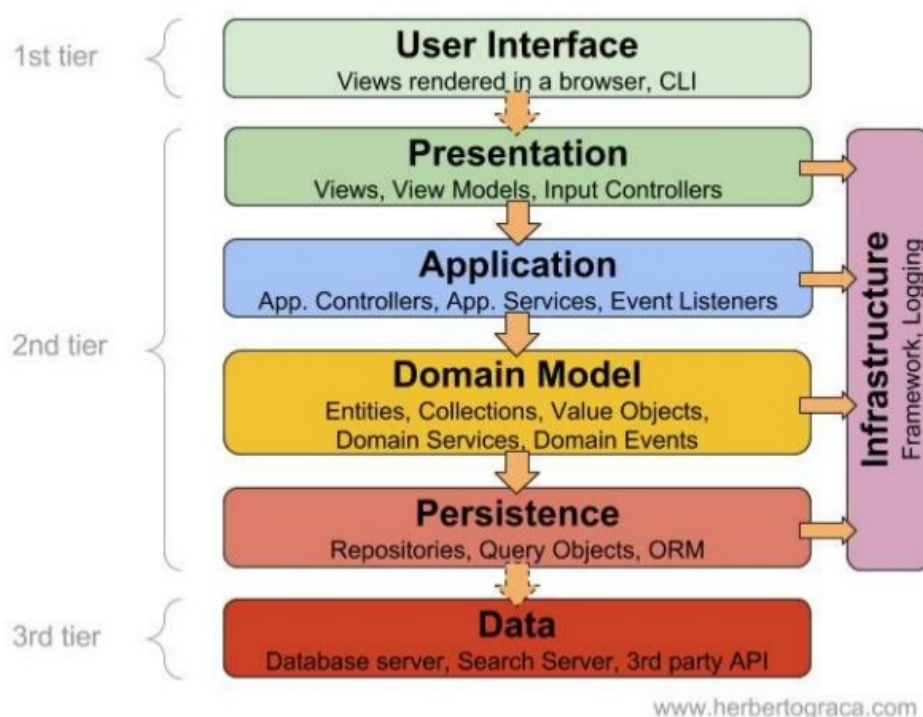


Figure 4.3: Spring Boot - 6 slojeva arhitekture, Izvor: <https://www.slideshare.net/alimenkou/hexagonal-architecture-with-spring-boot-136745841>

4.1.2 Frontend

Za razvoj frontend dijela aplikacije se koristi knjižnica **React** koja olakšava izradu komponenti za prikaz korisničkog sučelja te omogućava razvoj SPA (engl. *single page application*). Komponente će se graditi s **HTML**-om (Hypertext Markup Language) te oblikovati s **CSS**-om (Cascading Style Sheets). Osim vlastitog oblikovanja koristit će se CSS radni okvir **Bootstrap** koji ima velik izbor već uređenih komponenti.

4.2 Baza podataka

Za potrebe ove baze koristit ćemo sljedeće entitete:

- korisnik
- uloga
- obrazac_odgovor
- obrazac_korisnik
- obrazac_vrsta
- resurs
- lokacija
- odvoz_lokacija
- odvoz
- odlagalište
- otpad_vrsta
- proizvod

Odlučili smo se za relacijsku bazu podataka koja sadrži entitete i attribute zbog lakše prilagodbe stvarnom svijetu. Kod naše baze svaki entitet će kao primarni ključ sadržavati ID koji će se pri unosu novih podataka automatski povećavati.

4.2.1 Opis tablica

Korisnik Entitet sadrži atribut ID koji je primarni ključ, a uz njega sadrži i attribute korisničko_ime, lozinka, ime, prezime, lokacija_id, email adresa i uloga_id. Attribute korisničko_ime i lozinka korisnik sam određuje i tu kombinaciju imena i lozinke koristi za login na stranicu. Osim podataka koji su potrebni za login korisnik ako ima ulogu Građanin također unosi i svoje osobne podatke koji se spremaju na mjesto ostalih atributa dok kod uloge Zaposlenik ti atributi se postavljaju na NULL. Entitet korisnik vezan je s entitetima uloga, obrazac_odgovor, korisnikov_obrazac i lokacija. S entitetom uloga je vezan vezom Many-to-One gdje se uloga_id referencira na atribut ID kod entiteta uloga jer korisnik može

imati samo jednu ulogu dok se ista uloga može dodijeliti više korisnika. Preko veze s entitetom lokacija koja je One-to-One jer korisnik može imati samo jednu adresu stanovanja, a jedna adresa može biti dodijeljena samo jednom korisniku, dobivamo adresu stanovanja korisnika. Ovisno o ulozi koja je dodijeljena korisniku on može biti povezan s još 2 entiteta preko svojeg ID-a, a ti entiteti su obrazac_odgovor i obrazac_korisnik, a veza je opisana kod ta dva entiteta.

korisnik			
ID	INT	jedinstveni identifikator	primarni ključ, increment
korisnicko_ime	VARCHAR	ime koje je korisnik izabrao za login	not null, unique
lozinka	VARCHAR	hash lozinke	not null
ime	VARCHAR	ime građanina	
prezime	VARCHAR	prezime građanina	
lokacija_id	INT	ID lokacije koji određuje adresu korisnika	unique, strani ključ->lokacija.ID
email adresa	VARCHAR	email adresa građanina	unique
uloga_id	INT	ID uloge koja je dodijeljena korisniku	unique, strani ključ->uloga.ID

Uloga Entitet sadrži atribut ID koji je primarni ključ i pomoću njega se povezuje s entitetom korisnik, a uz njega ima i atribut naziv koji poprma vrijednost mogućih uloga.

uloga			
ID	INT	jedinstveni identifikator	primarni ključ, increment
naziv	VARCHAR	dodijeljena uloga	unique

Obrazac_korisnik Osim primarnog ključa entitet sadrži attribute naslov, korisnik_id, opis, vrsta_id, resurs_id, resurs_količina. Atribut naslov označava o čemu je obrazac, dok atribut opis sadržava detaljni opis što korisnik želi. Entitet je povezan s entitetom korisnik preko korisnik_id-a i to vezom Many-to-One jer jedan korisnik može napraviti više obrazaca dok jedan obrazac može pripadati samo jednom korisniku. Osim što je vezan s entitetom korisnik, entitet se također povezuje s entitetima obrazac_vrsta i resurs preko iste veze, a to je Many-to-One jer jedan resurs ili obrazac_vrsta može pripadati više različitih obrazaca_korisnik.

obrazac_korisnik			
ID	INT	jedinstveni identifikator	primarni ključ, increment
naslov	VARCHAR	naslov obrasca	not null
korisnik_id	INT	ID korisnika koji je kreirao obrazac_korisnik	not null, strani ključ->korisnik.ID
opis	TEXT	detaljan opis što korisnik želi dobiti obrascem	not null
vrsta_id	INT	određuje tip obrasca	not null, strani ključ->obrazac_vrsta.ID
resurs_id	INT	određuje tip resursa	strani ključ->resurs.ID
resurs_kolicina	INT	broj koliko korisnik želi resursa	not null ako resurs_id nije null

Obrazac_odgovor Entitet osim primarnog ključa sadrži atribute obrazac_korisnik_id, korisnik_id i sadržaj. Pomoću atributa obrazac_korisnik_id povezuje se s entitetom obrazac_korisnik vezom One-to-One tj. jedan obrazac_odgovor pripada jednom obrascu_korisnik i obrnuto. Također je povezan s entitetom korisnik sa korisnik_id vezom Many-to-One jer jedan korisnik može napraviti više obrazaca_odgovor dok jedan obrazac može biti kreiran od točno jednog korisnika. Zadnji atribut sadržaj sadrži odgovor na neki obrazac_korisnik.

obrazac_odgovor			
ID	INT	jedinstveni identifikator	primarni ključ, increment
obrazac_korisnik_id	INT	ID obrasca_korisnik na koji obrazac_odgovor odgovara	not null, strani ključ -> obrazac_korisnik.ID
korisnik_id	INT	ID korisnika koji je kreirao obrazac_odgovor	not null, strani ključ->korisnik.ID
opis	TEXT	odgovor na obrazac_korisnik	not null

Obrazac_vrsta Sadrži samo dva atributa i to ID koji je primarni ključ i naziv koji kaže o kojoj se vrsti obrazac_korisnik radi.

obrazac_vrsta			
ID	INT	jedinstveni identifikator	primarni ključ, increment
naziv	VARCHAR	vrsta obrazac_korisnik	not null, unique

Resurs Sadrži samo dva atributa i to ID koji je primarni ključ i tip koji kaže o kojem se tipu resurs radi.

resurs			
ID	INT	jedinstveni identifikator	primarni ključ, increment
tip	VARCHAR	tip resursa	not null, unique

Lokacija Entitet kao i ostali sadrži atribut ID koji je primarni ključ, a uz njega sadrži attribute grad, ulica i kućni_broj koji kada se gledaju zajedno trebaju biti unique.

lokacija			
ID	INT	jedinstveni identifikator	primarni ključ, increment
grad	VARCHAR	dio adrese	not null
ulica	VARCHAR	dio adrese	not null
kućni_broj	VARCHAR	dio adrese	not null

Odvoz_lokacija Entitet uz ID kao primarni ključ još sadrži i attribute lokacija_id i odvoz_id. Preko lokacija_id povezan je s entitetom lokacija vezom Many-to-One isto kao i s entitetom odvoz samo preko odvoz_id-a. Entitet nam služi kako bi razriješili vezu između odvoza i lokacije.

odvoz_lokacija			
ID	INT	jedinstveni identifikator	primarni ključ, increment
lokacija_id	INT	ID lokacije koju posjećuje odvoz	not null, strani ključ -> lokacija.ID
odvoz_id	INT	ID odvoza koji posjećuje lokaciju	not null, strani ključ->odvoz.ID

Odvoz Uz primarni ključ sadrži i atribut vrijeme koji nam govori kada se pojedini odvoz događa.

odvoz			
ID	INT	jedinstveni identifikator	primarni ključ, increment
vrijeme	TIMESTAMP	vrijeme kada se odvoz događa	not null, unique

Odlagalište Entitet nam pomoću atributa lokacija_id govori na kojoj se lokaciji nalazi odlagalište, a iz atributa radno_vrijeme_pocetak i radno_vrijeme_kraj saznajemo od kad do kad odlagalište radi. Entitet je povezan s entitetom lokacija preko lokacija_id vezom One-to-One jer se na pojedinoj lokaciji može nalaziti točno jedno odlagalište, a vrijedi i obrnuto jedno odlagalište može imati samo jednu lokaciju.

odlagalište			
ID	INT	jedinstveni identifikator	primarni ključ, increment
lokacija_id	INT	ID lokacije na kojoj se odlagalište nalazi	not null, unique strani ključ->lokacija.ID
radno_vrijeme_pocetak	TIME	početak radnog vremena	not null
radno_vrijeme_kraj	TIME	kraj radnog vremena	not null

Odlagalište_otpad Osim atributa ID koji služi kao primarni ključ, entitet sadrži i attribute odlagalište_id i vrsta_otpada_id. Vezan je s dva entiteta vezom Many-to-One, a ti entiteti su odlagalište i otpad_vrsta. S odlagalištem je vezan pomoću atributa odlagalište_id dok je s entitetom otpad_vrsta vezan preko atributa vrsta_otpada_id. Veza je Many-to-One jer za jedno Odlagalište_otpad možemo imati jedno odlagalište i otpad_vrstu dok odlagalište i otpad_vrsta mogu pripadati više odlagalište_otpad. Time odlagalište_otpad spaja entitete odlagalište i otpad_vrsta čija bi veza bila Many-to-Many.

odlagalište_otpad			
ID	INT	jedinstveni identifikator	primarni ključ, increment
odlagalište_id	INT	koje odlagalište spaja s vrstom otpada	not null, strani ključ->odlagalište.ID
vrsta_otpada_id	INT	koja vrsta_otpada pripada	not null, strani ključ -> otpad_vrsta.ID

Otpad_vrsta Uz primarni ključ ID entitet sadrži samo još jedan atribut, a to je naziv koji označuje o kojoj se vrsti otpada radi.

otpad_vrsta			
ID	INT	jedinstveni identifikator	primarni ključ, increment
naziv	VARCHAR	naziv otpada	not null, unique

Proizvod Uz primarni ključ ID entitet sadrži još dva atributa, a to su vrsta_id koji označuje o kojoj se vrsti otpada radi i ime koji dodjeljuje proizvodu ime. Povezan je s entitetom otpad_vrsta preko atributa vrsta_id i to vezom Many-to-One jer se jedan proizvod može biti napravljen od jedne vrste otpada dok jedna vrsta otpada može biti na više proizvoda.

proizvod			
ID	INT	jedinstveni identifikator	primarni ključ, increment
vrsta_id	INT	ID vrste otpada	strani ključ -> otpad_vrsta.ID
ime	VARCHAR	ime proizvoda	not null, unique

4.2.2 Dijagram baze podataka

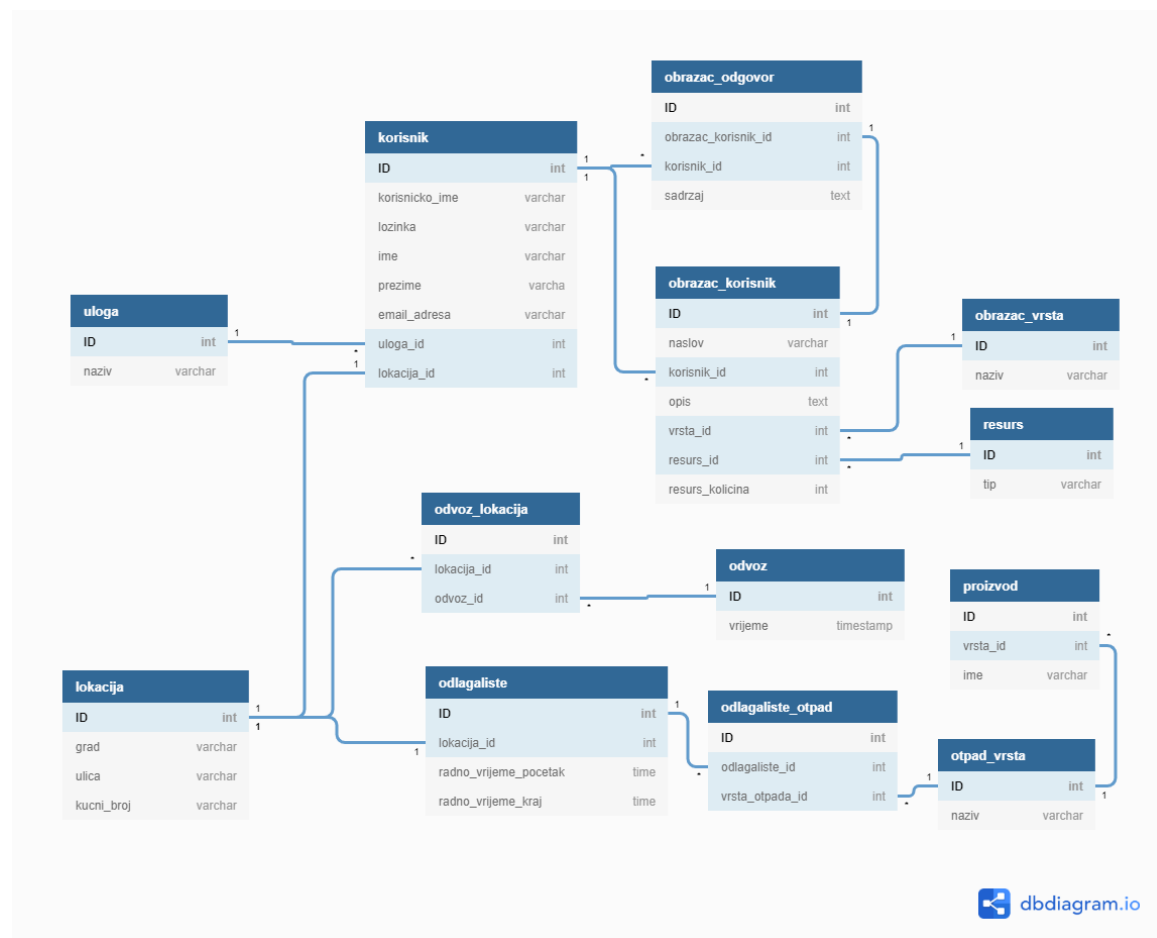


Figure 4.4: Baza podataka

4.3 Dijagram razreda

Dijagram razreda prikazuje odnose asocijacije i agregacije između razreda u našoj aplikaciji. Dijagram je modeliran

prema opisu baze tako da brojnost odgovara modelu baze. Korisnik je apstraktni razred kojeg nasljeđuju građanin, zaposlenik i administrator u implementaciji aplikacije. Za slanje obrazaca su vezani razredi ObrazacOdgovor i Korisnikov obrazac koji prema enumeraciji može biti zahtjev za resursima, zahtjev za glomaznim otpadom ili pritužba. Razred Lokacija je povezan sa građaninom (boravište) i odlagalištem. Razred Odvoz ukazuje na mogućnost odvoza na određenim lokacijama, dok odlagališta mogu sakupljati različite vrste otpadnih proizvoda.

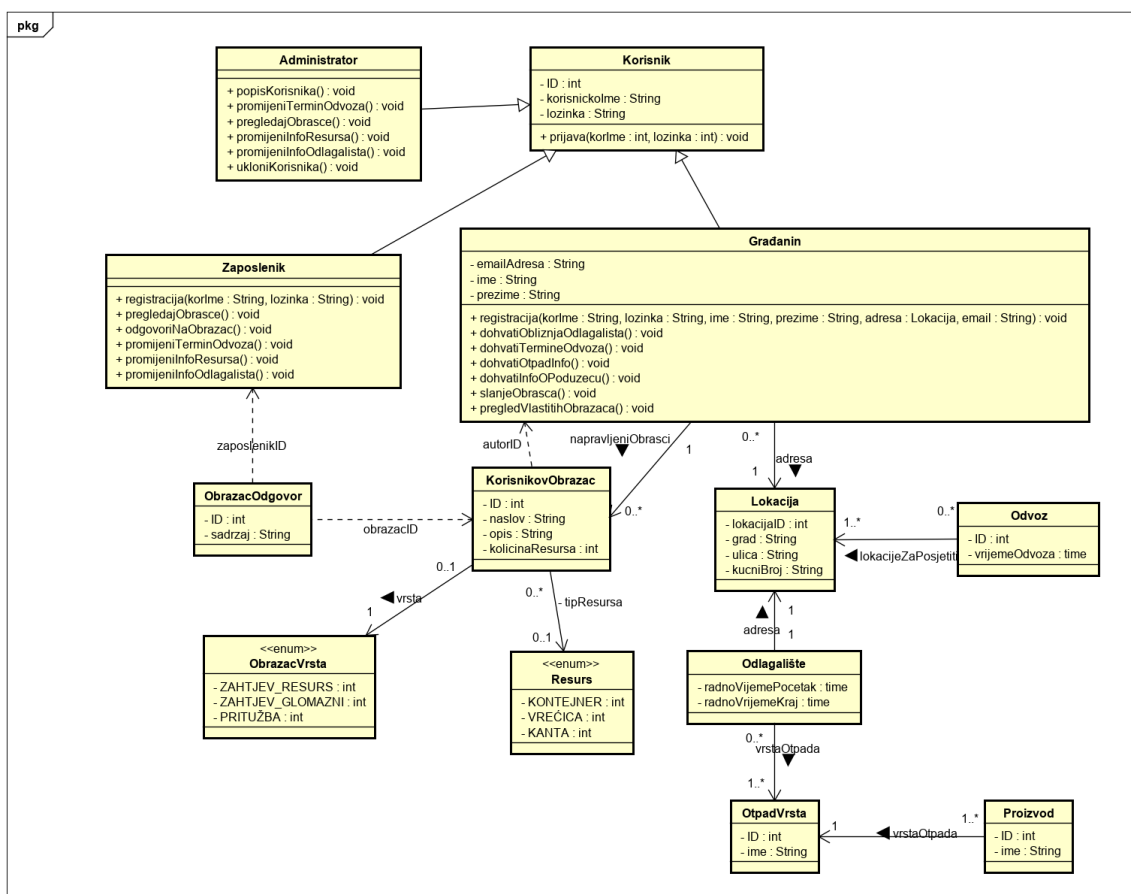


Figure 4.5: Dijagram razreda

4.4 Implementacijski dijagram razreda

Zbog kompleksnosti dijagrama, on će biti podijeljen na nekoliko njih po slojevima.

4.4.1 Sloj domene

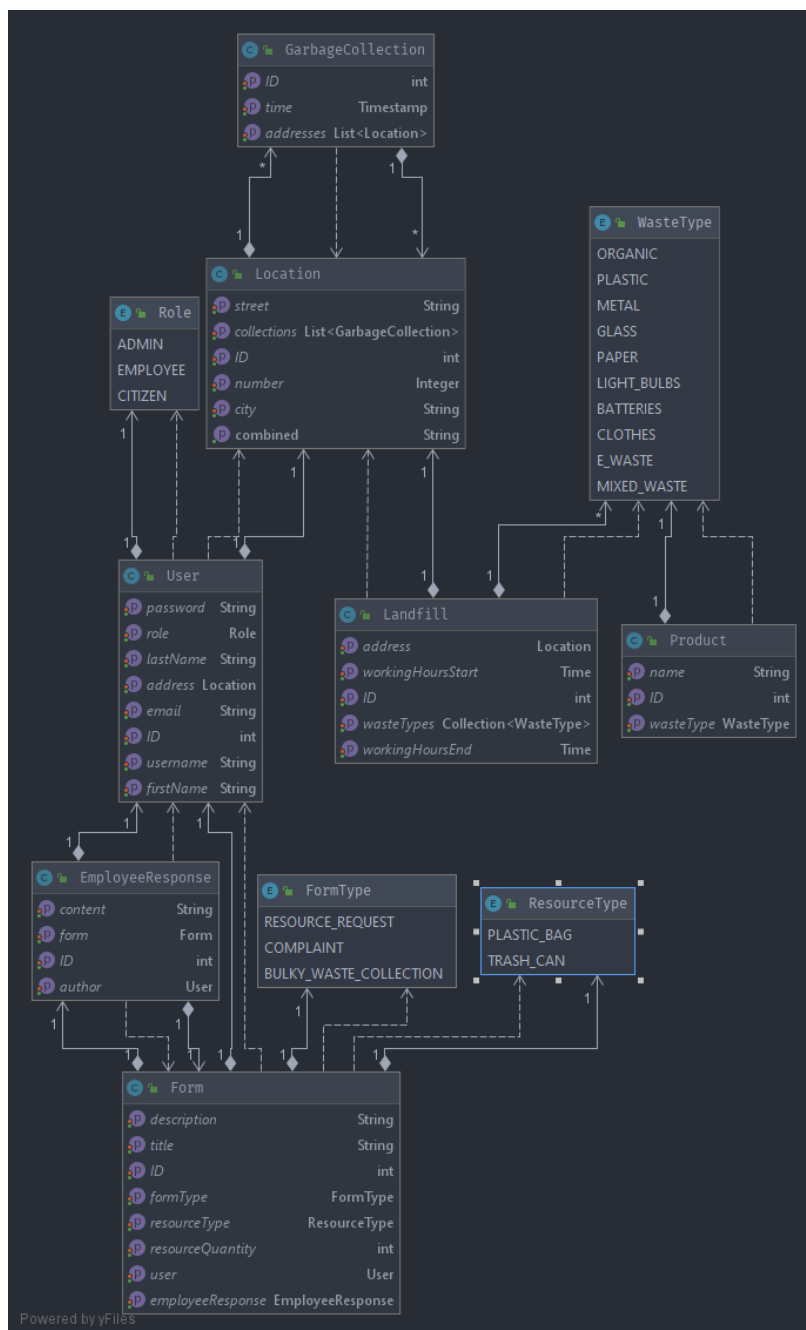


Figure 4.6: Sloj domene

4.4.2 Sloj za pristup podacima

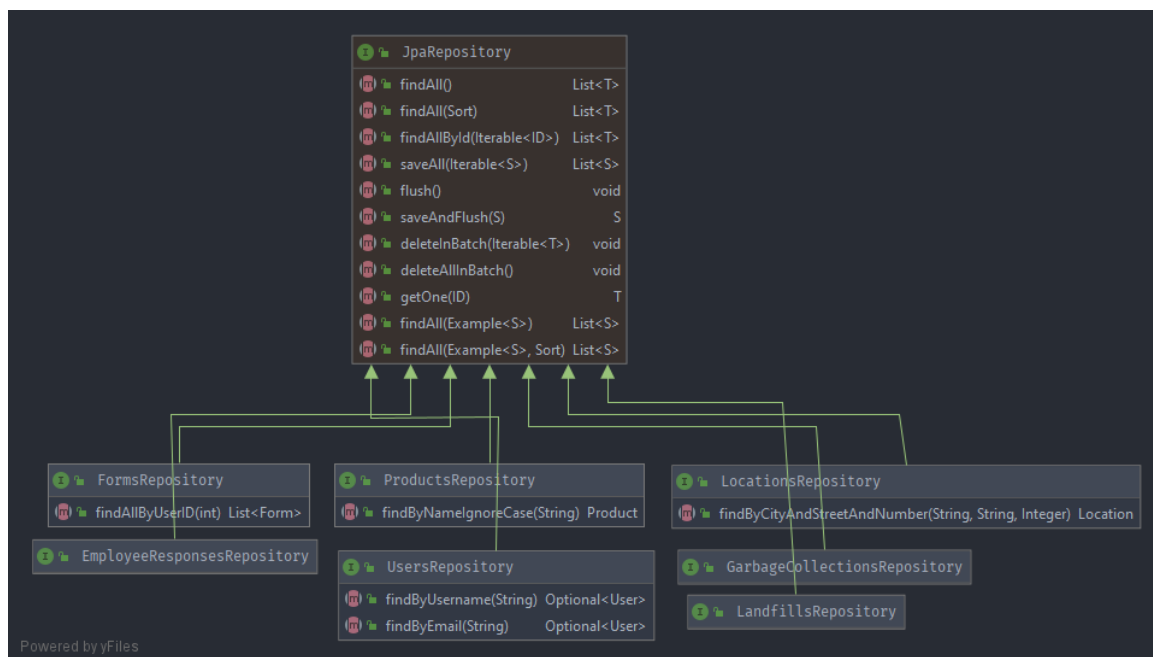


Figure 4.7: Sloj za pristup podacima

4.4.3 Sloj usluge

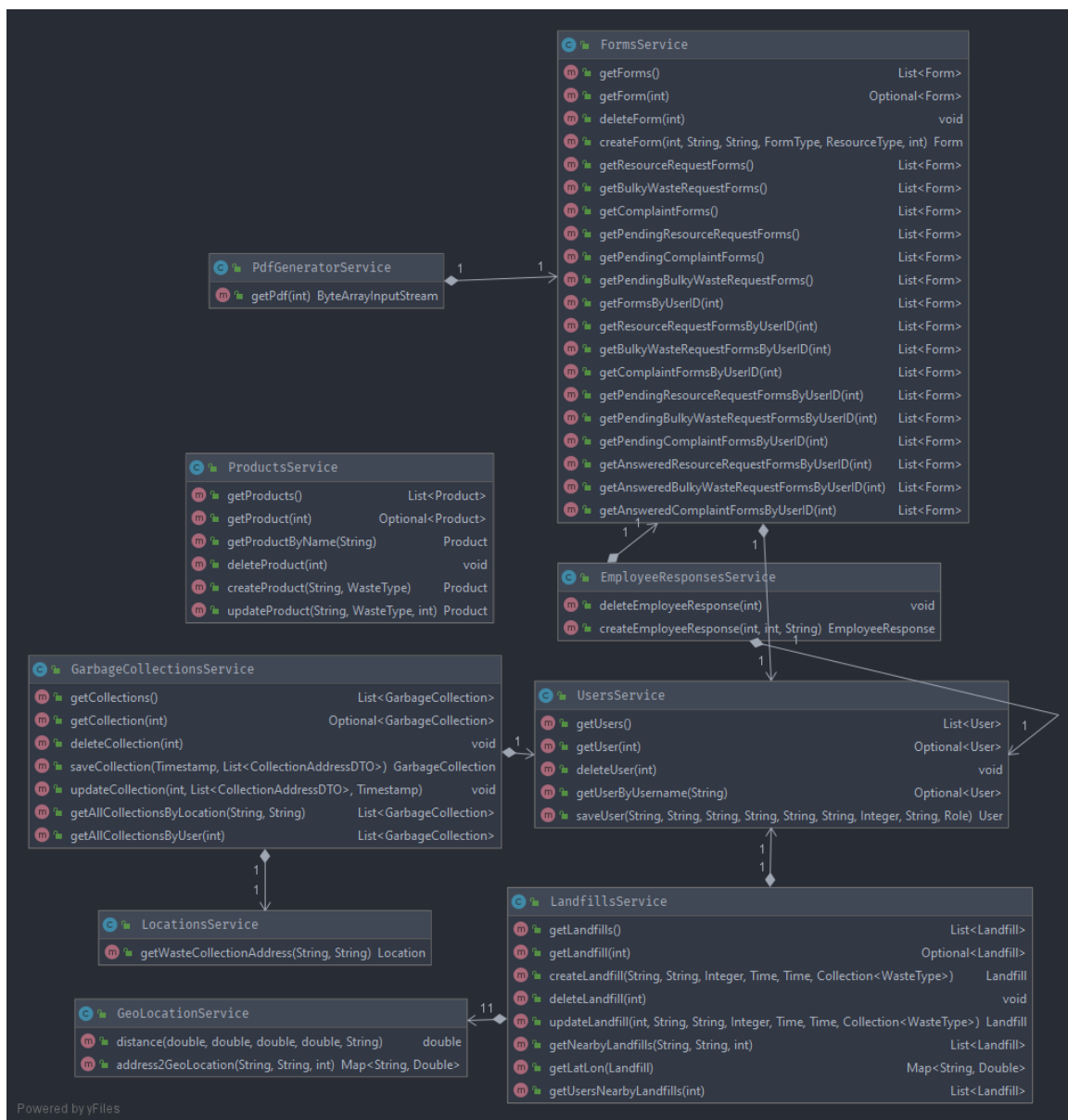


Figure 4.8: Sloj usluge

4.4.4 Sloj nadglednika

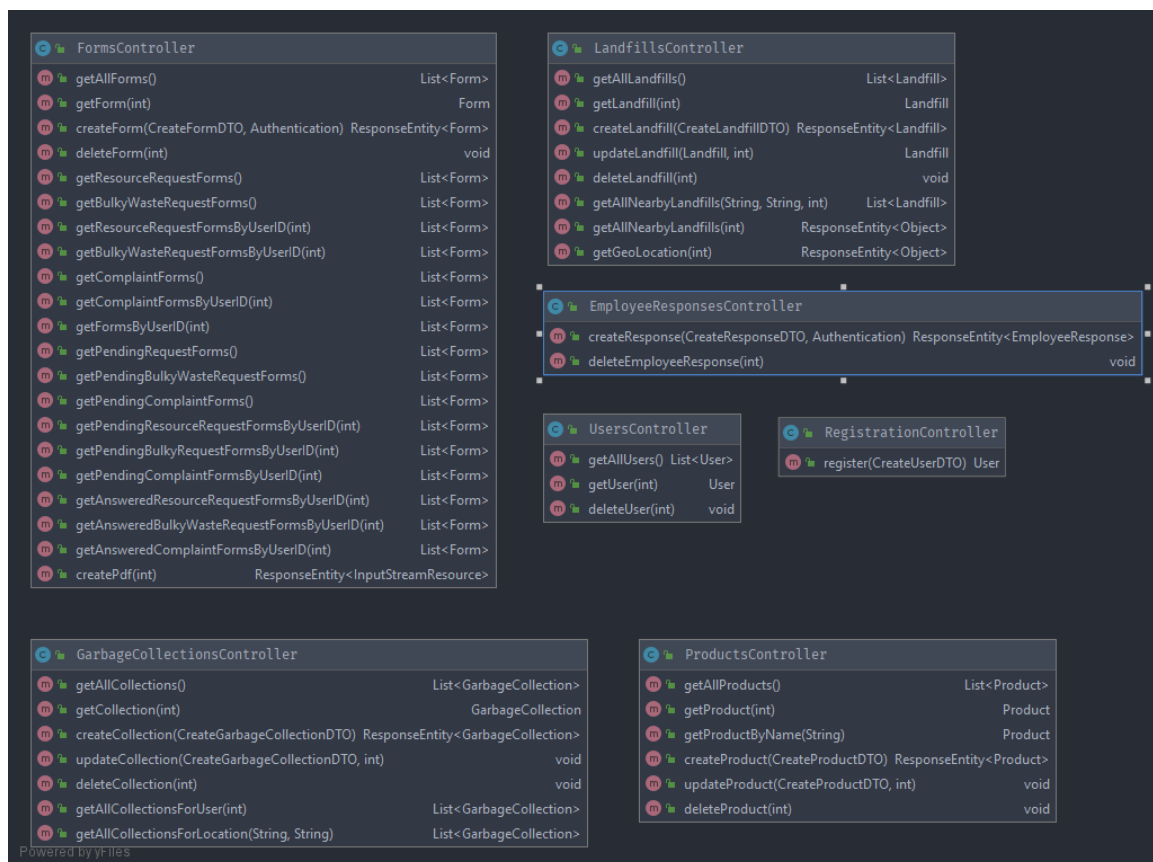


Figure 4.9: Sloj nadglednika

4.4.5 DTO razredi

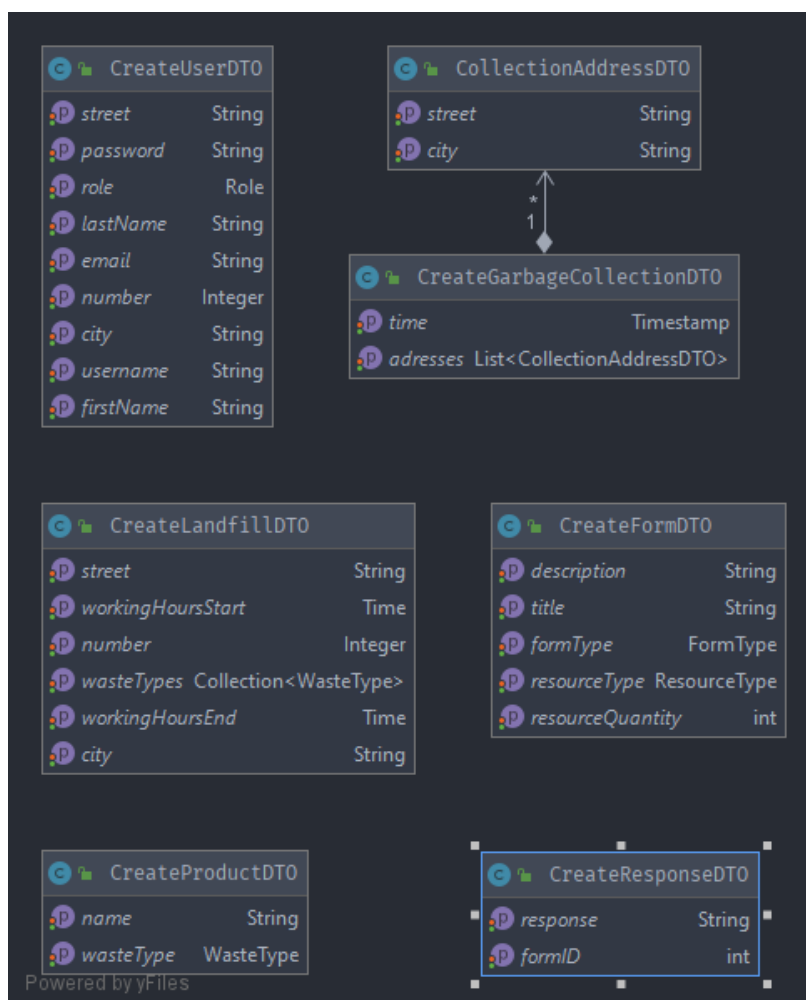


Figure 4.10: DTO razredi

4.4.6 REST dijagram (Controller + DTO + Service)

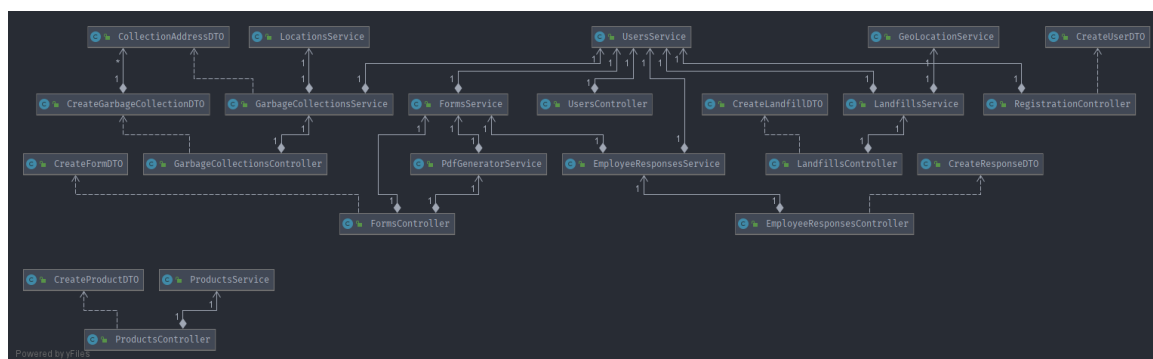


Figure 4.11: REST dijagram

novih termina i lokacija, ali ima iste mogućnosti pregledavanja tih termina i lokacija. Osim pregledavanja korisnik može kada se nalazi u stanju "Dodatni resursi" zatražiti dodatne resurse klikom na "Novi zahtjev" gdje mu se otvara novi prostor za unos detalja o zahtjevi. Iste događaje može napraviti kada se nalazi i u stanju "Glomazni otpad" i "Moje pritužbe".

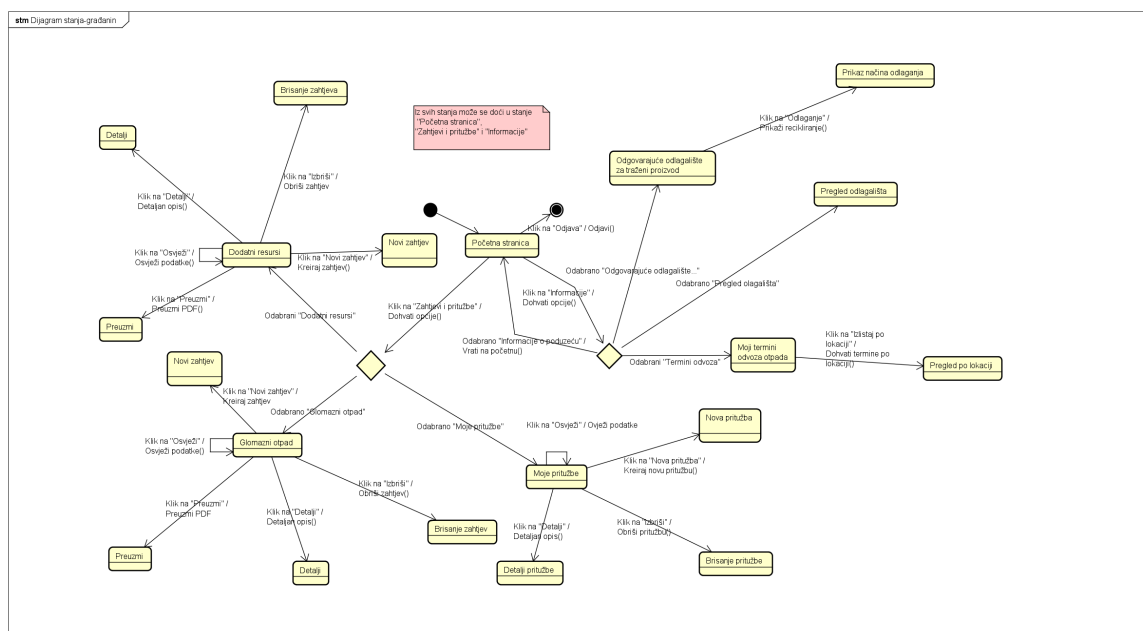


Figure 4.13: Dijagram stanja-građanin

4.6 Dijagram aktivnosti

Dijagram aktivnosti služi za opis modela toka upravljanja ili toka podataka. Inicijalno stanje označeno je crnim kružićem, dok je završno stanje prikazano zaokruženim crnim kružićem. Na dijagramu aktivnosti 4.6 prikazan je proces kreiranja zahtjeva za glomaznim otpadom. Korisnik se najprije prijavljuje u sustav, te ako su njegovi podaci točni, proslijeđuje ga se na početnu stranicu web aplikacije. Nakon toga korisnik odabire zahtjev za odvoz glomaznog otpada te može generirati vlastiti zahtjev ukoliko ne postoji nikakav neodgovoreni zahtjev. Slična implementacija funkcionira za slanje pritužbi te za slanje zahtjeva za dodatnim resursima.

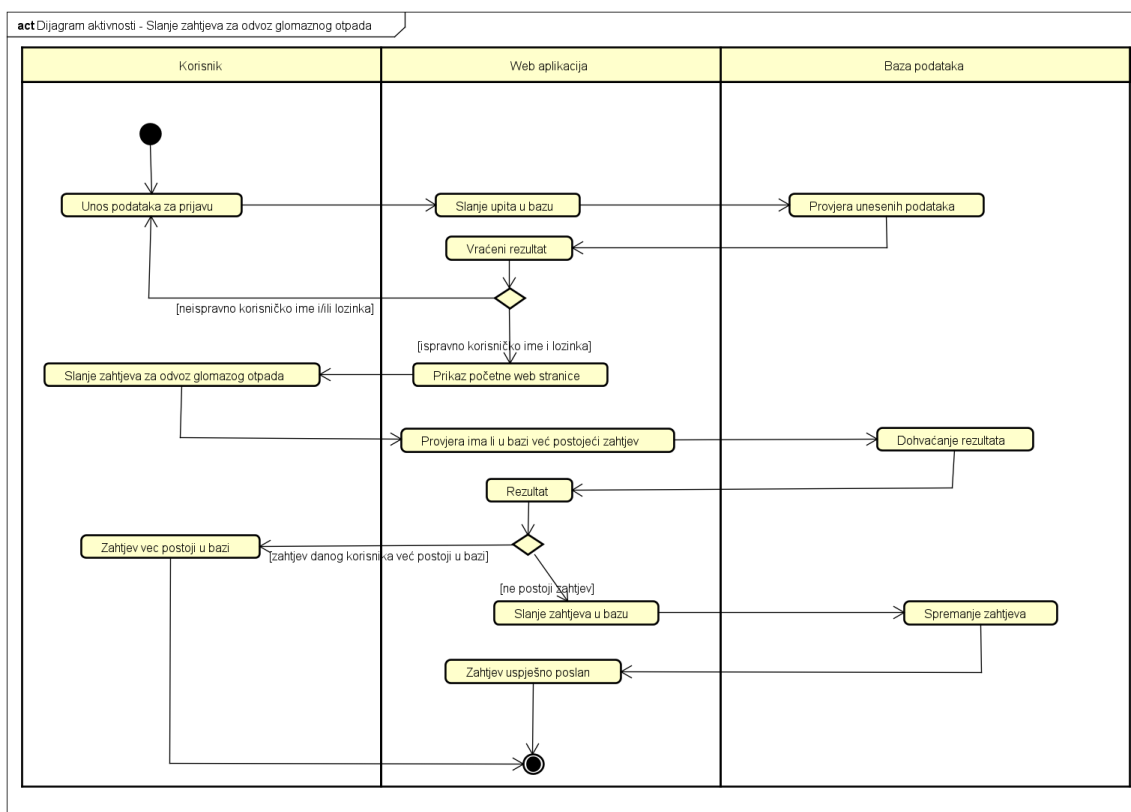


Figure 4.14: Dijagram aktivnosti

4.7 Dijagram komponenti

Dijagram komponenti koristi se za prikaz međuovisnosti komponenata, internu strukturu te odnose prema okolini. Sustav koristi dva sučelja. Sa sučeljem dohvat HTML, CSS i JS datoteka dohvaća se frontend dio aplikacije. Komponenta koja opslužuje HTML, CSS i JS datoteke je router koji zna koju datoteku opslužiti pomoću zahtjeva s url-om. Routerom se može doći do datoteka za prijavu, registraciju i početnih stranica određenih aktera. Biblioteke koje se uključuju u datoteke frontend dio aplikacije su react, bootstrap za ukrašene komponente i leaflet za karte. Sučeljem za dohvat JSON podataka pristupa se REST API komponenti. REST API prihvaća sve podatke i poziva odgovarajuće metode za prihvrat u Controllers komponenti. Jpa Repositories zadužen je za dohvaćanje tablica iz baze podataka. Pretvara podatke u DAO modele i šalje ih ostatku backend aplikacije (Service sloju). Controllers komponenta obrađuje pristigle upite iz REST API komponente i poziva odgovarajuće metode iz Service sloja. Service sloj zadužen je za usluge koje se obavljaju nad podacima iz baze podataka koji su mapirani u komponenti Models.

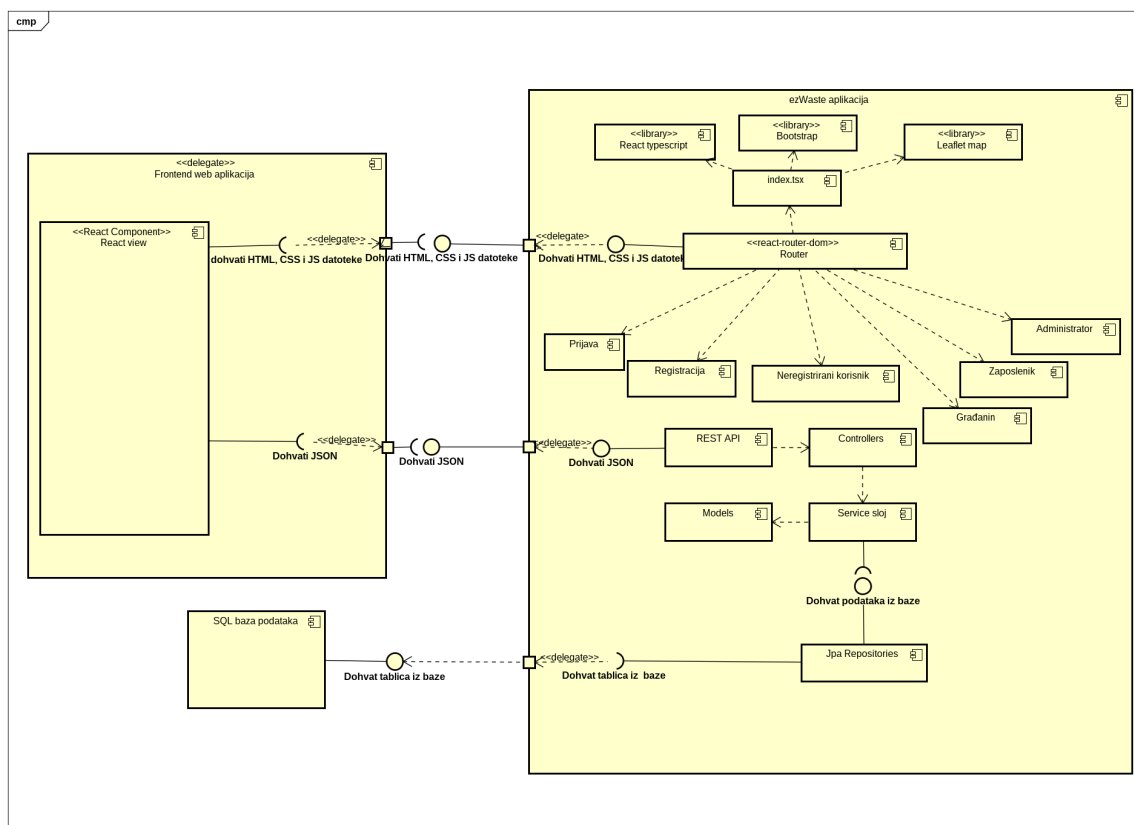


Figure 4.15: Dijagram komponenti

5. Implementacija i korisničko sučelje

5.1 Korištene tehnologije i alati

1. Java

- Objektno orijentirani programski jezik koji smo koristili za izradu naše backend aplikacije.

- <https://docs.oracle.com/en/java/>

2. Spring Boot

- Radni okvir(engl. *framework*) koji smo koristili za efikasniju izradu backend aplikacije koju smo pisali u programskom jeziku Java. Sadrži velik niz već uključenih knjižnica, kao što su Jackson, Hibernate itd.

- <https://spring.io/projects/spring-boot>

3. Hibernate ORM

- Alat koji omogućava pretvorbu tj. mapiranje redova iz tablice baze podataka u objekte Java aplikacije.

- <https://hibernate.org/orm/>

4. Jackson

- Knjižnica(engl. *library*) koja omogućava pretvorbe između objekata u Java aplikaciji i Stringova u JSON formatu.

- <https://github.com/FasterXML/jackson-docs>

5. React

- Knjižnica koju smo koristili za efikasniju izradu frontend aplikacije tj. korisničkog sučelja za čiju izradu su se koristile tri glavne web tehnologije (HTML, CSS, Javascript).

- <https://reactjs.org/>

6. HTML

- Osnovni jezik za izradu web stranica. Stvara html elemente, ali ne podržava veću stilizaciju tih elemenata.

- <https://developer.mozilla.org/en-US/docs/Web/HTML>

7. CSS

- Jezik za uređivanje html elemenata tj. stiliziranje web stranice.
- <https://developer.mozilla.org/en-US/docs/Web/CSS>

8. Javascript

- Programski jezik koji smo koristili za implementaciju funkcionalnosti na frontend aplikaciji.
- <https://developer.mozilla.org/en-US/docs/Web/JavaScript>

9. React Bootstrap

- Radni okvir koji sadrži veliku količinu već uređenih React komponenti.
- <https://react-bootstrap.github.io/>

10. Git

- Distribuirani sustav za upravljanje različitim verzijama podataka (npr. programskog koda ili teksta).
- <https://git-scm.com/>

11. Gitlab

- Web platforma na kojoj se nalazi naš udaljeni git repozitorij.
- <https://about.gitlab.com/>

12. Slack

- Platforma koju smo koristili za komunikaciju unutar tima.
- <https://slack.com/intl/en-hr/>

13. Heroku

- Platforma na kojoj se poslužuje naša web aplikacija (frontend, backend i baza podataka).
- <https://www.heroku.com/home>

14. PostgreSQL

- Sustav za upravljanje bazom podataka koji smo koristili za relacijsku bazu podataka za našu aplikaciju.
- <https://www.postgresql.org/>

15. IntelliJ IDEA

- Integrirana razvojna okolina koja služi primarno za pisanje Java aplikacija, ali ju je osim backend tima, koristio i frontend tim.
- <https://www.jetbrains.com/idea/>

16. Eclipse

- Integrirana razvojna okolina za pisanje Java aplikacija koju je koristio dio backend tima.
- <https://www.eclipse.org/>

17. Visual Studio Code

- Razvojna okolina koja podržava mnoge tehnologije, a u našem timu ju je koristio frontend tim.
- <https://code.visualstudio.com/>

18. PgAdmin4

- Alat za upravljanje PostgreSQL bazom podataka.
- <https://www.pgadmin.org/>

19. Swagger2

- Alat koji smo koristili za lakšu komunikaciju backenda i frontenda. Alat iz programskog koda dokumentira cijeli API, te podržava grafičko sučelje koje je frontend tim koristio kako bi se informirao o API-ju kojeg treba koristiti.
- <https://swagger.io/>

20. HERE REST APIs

- Besplatni API koji smo koristili za geokodiranje adresa iz naše aplikacije.
- <https://developer.here.com/develop/rest-apis>

5.2 Ispitivanje programskog rješenja

5.2.1 Ispitivanje komponenti

Pomoću JUnit 4 napravljeno je 12 testova za klasu UsersService kojima se testiraju CRUD operacije spremanja, brisanja i dohvaćanja korisnika.

```
@SpringBootTest
public class UpravljanjeKucnimOtpadomApplicationTests {

    @Autowired
    private UsersService serv;

    @Test
    public void returnsEmptyIfUserIsNotFound() {
        assertEquals(Optional.empty(), serv.getUser(-1));
    }

    @Test(expected=BadRequestException.class)
    public void cantCreateTwoUsersWithSameUsername() {
        serv.saveUser("Username", "password", "firstName",
            "lastName", "city", "street", 1, "email10",
            Role.ADMIN); //dodan
        serv.saveUser("Username", "password", "firstName",
            "lastName", "city", "street", 1, "email9", Role.ADMIN);
        fail();
    }

    @Test(expected=IllegalArgumentException.class)
    public void userNameCantBeNull() {
        serv.saveUser(null, "password", "firstName", "lastName",
            "city", "street", 1, "email8", Role.ADMIN);
        fail();
    }

    @Test(expected=IllegalArgumentException.class)
    public void userNameCantBeBlank() {
        serv.saveUser("", "password", "firstName", "lastName",
```

```
        "city", "street",1, "email6", Role.ADMIN);
fail();
}

@Test(expected=IllegalArgumentException.class)
public void passwordNameCantBeNull() {
    serv.saveUser("Username", null, "firstName", "lastName",
        "city", "street",1, "email7", Role.ADMIN);
fail();
}

@Test(expected=IllegalArgumentException.class)
public void passwordNameCantBeBlank() {
    serv.saveUser("Username", "", "firstName", "lastName",
        "city", "street",1, "email5", Role.ADMIN);
fail();
}

@Test(expected=IllegalArgumentException.class)
public void passwordLengthMustBeGreaterOrEquals8() {
    serv.saveUser("Username", "passwor", "firstName", "lastName",
        "city", "street",1, "email4", Role.ADMIN);
fail();
}

@Test(expected=IllegalArgumentException.class)
public void roleMustNotBeNull() {
    serv.saveUser("Username", "password", "firstName",
        "lastName", "city", "street",1, "email3", null);
fail();
}

@Test
public void getByUsername() {
    serv.saveUser("Usernamee", "password", "firstName",
        "lastName", "city", "street",1, "email",
        Role.ADMIN); //dodan
    User user=serv.getUserByUsername("Usernamee").get();
}
```

```
        if(!user.getUsername().equals("Usernamee")) {
            fail();
        }
    }

@Test
public void userDeleteTest() {
    serv.saveUser("Username", "password", "firstName",
        "lastName", "city", "street",1, "email1",
        Role.ADMIN);//dodam
    User toBeDeleted=serv.getUserByUsername("Username").get();
    serv.deleteUser(toBeDeleted.getID());
}

@Test(expected=BadRequestException.class)
public void createTwoUsersWithSameEmails() {
    serv.saveUser("Username", "password", "firstName",
        "lastName", "city", "street",1, "emailsame",
        Role.ADMIN);//dodan
    serv.saveUser("User", "password", "firstName", "lastName",
        "city", "street",1, "emailsame", Role.ADMIN);
    fail();
}

@Test
public void getAllUsers() {
    serv.saveUser("User", "password", "firstName", "lastName",
        "city", "street",1, "emailsame", Role.ADMIN);//dodan
    serv.saveUser("Username", "password", "firstName",
        "lastName", "city", "street",1, "email1",
        Role.ADMIN);//dodam
    serv.saveUser("Userr", "password", "firstName", "lastName",
        "city", "street",1, "email10", Role.ADMIN);//dodan
    if(serv.getUsers().size() != 3){
        System.out.println(serv.getUsers().size());
        fail();
    }
}
```

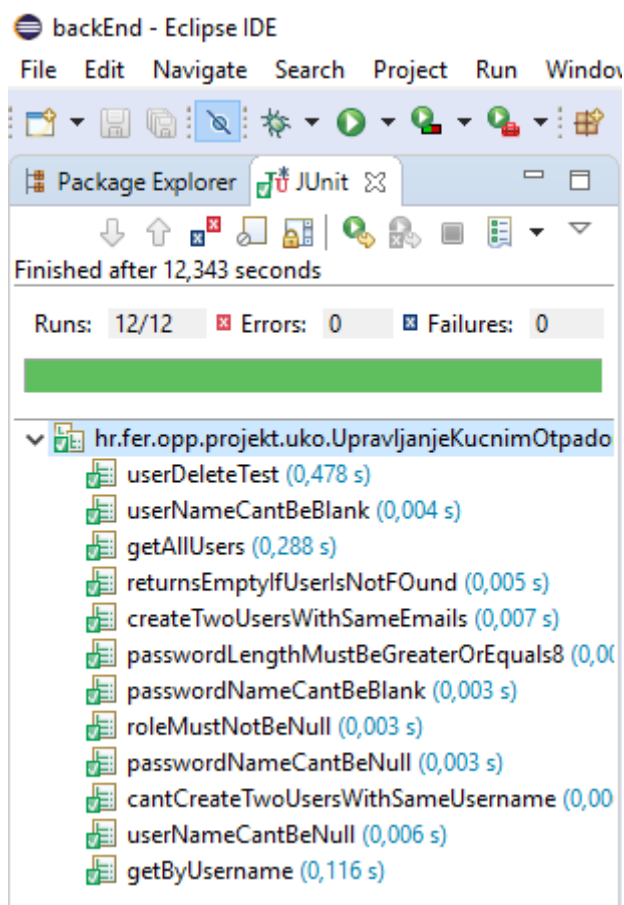
}

Figure 5.1: JUnit 4 testovi

5.2.2 Ispitivanje sustava

Ispitivanje sustava se izvodilo pomoću dodatka za preglednik Selenium IDE. U nastavku će se pokazati primjeri ispitivanja rubnih slučajeva, pozivanje funkcionalnosti koja nije potpuno implementirana te ispitivanje redovitih slučajeva.

Ispitni slučaj 1: Registracija korisnika s postojećim usernameom

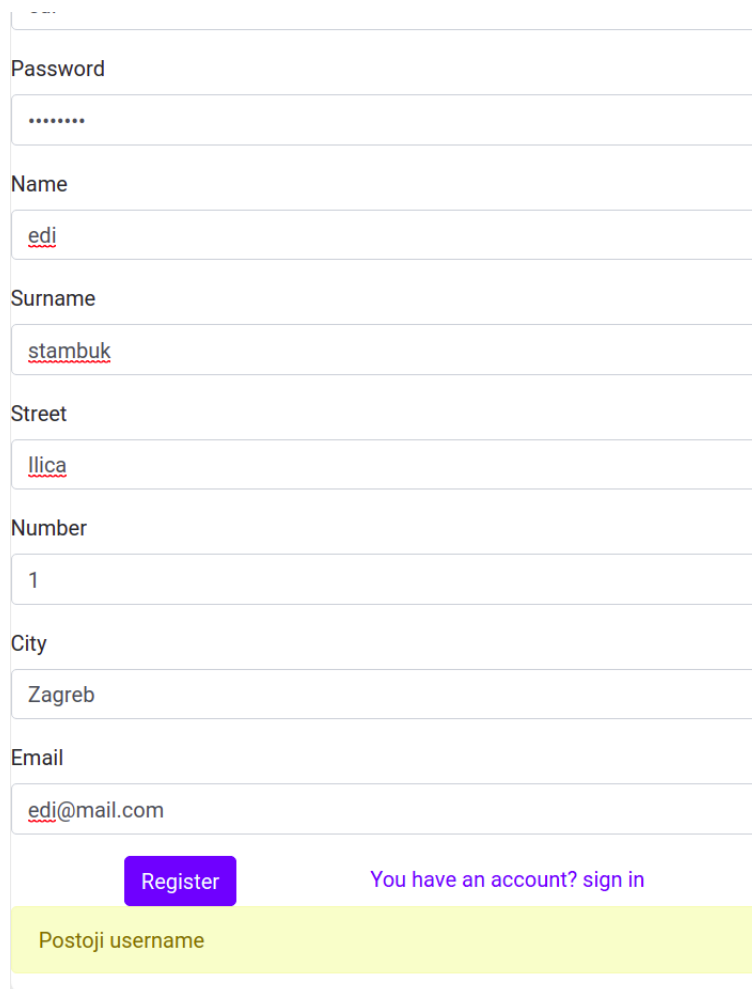
Ulaz:

1. Otvaranje stranice za registraciju u web pregledniku
2. Upisivanje podataka za korisnika s već postojećim username podatkom

Očekivani rezultat:

1. Pojavljivanje upozorenja da username već postoji
2. Registracija ne prolazi

Rezultat: Pojavilo se upozorenje. Aplikacija je prošla test



The screenshot shows a registration form with the following fields and values:

- Password: (masked)
- Name: edi
- Surname: stambuk
- Street: Ilica
- Number: 1
- City: Zagreb
- Email: edi@mail.com

Below the fields, there is a purple "Register" button and a link "You have an account? sign in". At the bottom, a yellow banner displays the warning message "Postoji username".

Figure 5.2: Upozorenje

Ispitni slučaj 2: Upisivanje slova u polje za upis vremena**Ulaz:**

1. prijavljivanje kao zaposlenik ili administrator
2. Otvaranje stranice za dostupnost odlagališta

3. Pritisak mišem na lokaciju na karti
4. Ažuriranje podataka za lokaciju
5. upisivanje slova u polje za upis vremena

Očekivani rezultat:

1. Pojavljivanje greške zbog netočnog formata datuma

Rezultat: Korisnik se nije registrirao i nije se pojavila greška zbog netočnog upisa. Aplikacija nije prošla test

pis korisnika Informacije ▾

Dodaj lokaciju ✕

Grad
Zagreb

Ulica broj
Ulica Vučak 4

☒ ORGANIC ☐ PLASTIC ☐ METAL ☐ GLASS ☐ E-WASTE
☐ PAPER ☐ LIGHT_BULBS ☐ BATTERIES ☐ CLOTHES

Početno Vrijeme
11:00bf

Završno Vrijeme
07:00

Close Save Changes

Uredi lokaciju

info o lokaciji

Zagreb Ulica Vučak 4

Na odlagalištu se odlaže
ORGANIC Radi od 20:00 do 07:00

izbrisi uredi

Figure 5.3: Upis slova u komponentu za vrijeme

Ispitni slučaj 3: Uspješno slanje zahtjeva za resursima**Ulaz:**

1. Prijavljivanje kao građanin
2. Otvaranje stranice za slanje zahtjeva za resursima

3. Odabir resursa i broj traženih resursa

4. Pritisak na gumb pošalji

Očekivani rezultat:

1. Pojavljivanje informacija o poslanom zahtjevu

2. Pojavljivanje zahtjeva na listi poslanih zahtjeva

Rezultat: Korisnik je uspješno poslao zahtjev. Aplikacija je prošla test

ZAHTJEVI ZA DODATNIM RESURSIMA

Vrsta resursa	Detalji	Status	PDF
VREČICA	Detalji	BEZ odgovora	Preuzmi
VREČICA	Detalji	BEZ odgovora	Preuzmi

[Novi Zahtjev](#)

ZAHTJEV ZA RESURSIMA

Korisnik: Edi Stambuk
Adresa: Ilica 1, Zagreb

Vrsta resursa: VREČICA
Količina resursa: 2
Za otpad

Figure 5.4: Zahtjev za resursima

Ispitni slučaj 4: Pretraga odgovarajućeg odlagališta za pojedini proizvod

Ulaz:

1. Otvaranje stranice za prikaz odgovarajućeg odlagališta za pojedini proizvod

2. Upisivanje i odabir proizvoda za odlaganje

3. Pritisak na gumb odlaganje

Očekivani rezultat:

1. Pojavljivanje odgovarajućeg tipa odlagališta

Rezultat: Korisnik je uspješno dobio odgovarajući tip odlagališta. Aplikacija je prošla test



Figure 5.5: Pronalazak odgovarajućeg tipa odlagališta

5.3 Dijagram razmještaja

Dijagrami razmještaja prikazuju računalne resurse koji su neophodni za ispravno funkcioniranje sustava i njihove međusobne odnose: stvarne uređaje (poslužitelje, radne stanice, korisnička računala, itd.), komponente programske podrške koje se na njima izvršavaju i veze između prikazanih resursa. Na računalu poslužitelja nalaze se poslužitelj baze podataka i web poslužitelj. Korisnik preko web poslužitelja pristupa sadržaju web aplikacije. HTTP veza je zaslužna za interakciju između korisnika i poslužitelja, dok je cijeli sustav građen na arhitekturi "klijent-poslužitelj", koja je jedna od najprimjenjivanijih metoda današnjice.

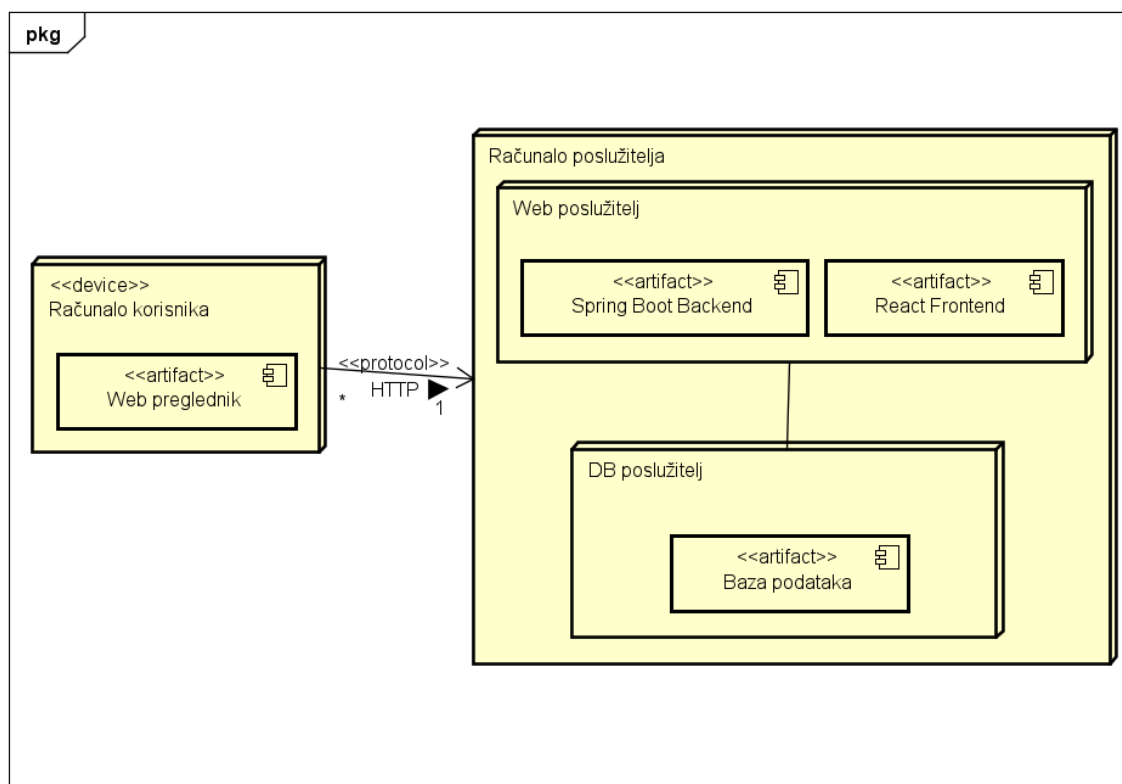


Figure 5.6: Dijagram razmještaja

5.4 Upute za puštanje u pogon

Aplikacija je javno dostupna na adresi <https://ezwaste.herokuapp.com/>

5.4.1 Priprema baze podataka

1. Na računalo instalirati neki od sustava za upravljanje bazom podataka (SUBP), u primjeru ćemo koristiti PostgreSQL. PostgreSQL SUPB se može preuzeti sa stranice <https://www.postgresql.org/>.
2. Nakon preuzimanja, pokrenuti instalaciju i pri instalacija odabrati password koji će se koristiti za spajanje na baze podataka koje se poslužuju na lokalnom poslužitelju. (username će biti 'postgres', password koji ćemo odabrati za primjer je 'admin'). Također tijekom instalacije potrebno je označiti opciju 'pgAdmin4'. To je program koji ima grafičko sučelje kroz koje korisnici mogu stvarati baze podataka i raditi upite nad njima.
3. U sučelju programa pgAdmin4, pod rubrikom Servers odabrati PostgreSQL (11/12/...) server i povezati se s username i password koji ste prethodno odabrali.
4. Desni klik na 'Databases', zatim 'Create', pa 'Database...'. U sljedećem prozoru unesite neko ime baze podataka (npr. 'ezwaste' za našu aplikaciju).
5. Desni klik na ime baze podataka koju ste napravili, zatim 'Query tool...'. Sada će vam biti otvoren prozor u kojem možete izvršavati upite nad bazom podataka. U prozor kopirajte sadržaj datoteke 'ezwaste_create_tables.sql' i pokrenite naredbe s tipkom F5.
6. Ako želite osim tablica, u bazu unijeti i neke podatke za te tablice, onda napravite isti postupak kao u koraku prije, ali za datoteku 'insert_data.sql'.
7. U ovom trenutku će vam sve tablice biti stvorene u bazi podataka i možete krenuti na povezivanje aplikacije na lokalnu bazu podataka.
8. U datoteci application.properties nalazi se sva konfiguracija koju backend aplikacija treba za povezivanje na bazu podataka.
9. Ako ste pratili sve korake, ovako treba izgledati konfiguracija koju upisujete u application.properties datoteku:
`spring.datasource.url = jdbc:postgresql://localhost:5432/ezwaste`

```
spring.datasource.username = postgres  
spring.datasource.password = admin  
spring.jpa.hibernate.ddl-auto = none
```

5.4.2 Pokretanje backend aplikacije

1. Za pokretanje backend aplikacije potrebno je otvoriti /izvorniKod/backend direktorij. U tom direktoriju se nalazi direktorij 'upravljanje-kucnim-otpadom', to je root direktorij našeg maven projekta kojeg otvorite u željenoj razvojnoj okolini (Intellij IDEA, Eclipse itd.)
2. Spring boot aplikacija se pokreće tako da pokrenete (engl. run) main metodu koja se nalazi u 'UpravljanjeKucnimOtpadomApplication' klasi.

5.4.3 Pokretanje frontend aplikacije

1. Za pokretanje frontend aplikacije potrebno je imati instaliran Node.js (<https://nodejs.org/en/>) te npm (dolazi uz Node.js).
2. Potrebno je "reći" frontend aplikaciji na kojoj se adresi nalazi API kojeg koristi. Otvoriti src/store/BaseUrl i provjeriti konstantu 'POCETNA_ADRESA', ako nije, postaviti ju na 'localhost:8080'.
3. Aplikacije se pokreće tako da se pozicionirate u root direktorij projekta. /izvorniKod/frontend/upravljanje_kucnim_otpadom te u komandnoj liniji izvršite naredbe :
-npm install
-npm start

6. Zaključak i budući rad

6.1 Projektni zadatak

Naš projektni zadatak je bio napraviti web aplikaciju za upravljanje kućnim otpadom. Cijeli razvoj je bio podijeljen u dvije faze.

6.1.1 Prva faza

Prvi zadatak nam je svima bio oformiti tim od 7 članova. Nakon osnivanja tima, bilo je potrebno ravnomjerno podijeliti posao i da se uz to, svaki član tima, bavi zadacima u kojima se najbolje snalazi i/ili koje bi želio (dodatno) naučiti.

Prvi ciklus razvoja programske potpore i dokumentacije se sastojao od dva glavna dijela:

- Razvoj generičke funkcionalnosti, koja se sastojala od:
 - Razvoja stranica koje služe za registraciju i prijavu te početnih stranica za svaku vrstu korisnika
 - Autentikacije i autorizacije koje se događaju u pozadini
- Pisanje dokumentacije koja se sastojala od:
 - Opisa projektnog zadatka
 - Specifikacije programske potpore
 - Opisa arhitekture

Tijekom prvog ciklusa izrade projekta, iskušali smo se, dakle, i u razvojnom, i u dokumentacijskom dijelu projekta.

6.1.2 Druga faza

Druga faza nije bila neovisna o prvoj fazi. Sva dokumentacija koju smo napravili u prvoj fazi nam je koristila u drugoj, to su najviše bili model baze, dijagram razreda i popis funkcionalnih zahtjeva. Dobro obavljena prva faza projekta, olakšala nam je drugu fazu.

Većinu vremena u drugoj fazi članovi tima su potrošili upravo na implementaciju.

Drugi ciklus razvoja programske potpore i dokumentacije se sastojao od tri glavna dijela:

- Implementacija svih potrebnih funkcionalnosti
 - Izrada api-ja na backendu
 - Izrada grafičkog sučelja na frontendu
- Testiranje rada web aplikacije
- Pisanje dokumentacije koja se sastojala od:
 - Dijagrama stanja, aktivnosti, komponenti i razmještaja
 - Popisa tehnologija i alata koje smo koristili
 - Uputa za puštanje aplikacije u pogon

6.2 Opći dojmovi

Svi članovi tima su zadovoljni projektom i konačnim proizvodom. Uloženo je mnogo truda i rada, ali ako razmotrimo koliko smo naučili tijekom ovom projekta, te kako izgleda konačni proizvod, onda se svi slažemo da je bilo vrijedno.

Radom na projektu naučili smo kako je raditi u timu na zajedničkom projektu, a većinu vremena komunicirati preko platformi kao što je Slack.

Tijekom procesa razvoja, kako bi se ispoštovali krajnji rokovi i zahtjevi, bilo je potrebno redovito pratiti napredak podtimova i pojedinaca u razvojnom timu, te organizirano dijeliti posao po određenim tjednima. Stizanjem takvih rokova, naučili smo koliko je bitna dobra organizacija, i kako uspjeti napraviti dobru organizaciju.

Naravno nije uvijek bilo sve sjajno tijekom projekta, i putem smo se sreli s mnogo problema, ali smo isto tako onda zajedno sudjelovali u traženju rješenja za svaki od tih problema.

Popis literature

1. Sjedi pa jedi, Oblikovanje programske potpore, FER, https://www.fer.unizg.hr/_download/repository/SjediPaJedi-primjer.pdf
2. Materijali s predavanja, Oblikovanje programske potpore, FER, <http://www.fer.hr/predmet/opp>
3. Spring Boot tutorials, <https://www.baeldung.com/spring-boot>

Indeks slika i dijagrama

2.1	Karta odlagališta na stranici https://www.cistoca.hr/	5
2.2	Slanje zahtjeva za odvoz glomaznog otpada s https://www.cistoca.hr .	6
3.1	Građani	19
3.2	Građani	20
3.3	Sekvencijski dijagram za UC4	21
3.4	Sekvencijski dijagram za UC18	22
3.5	Sekvencijski dijagram za UC13	23
3.6	Sekvencijski dijagram za UC16	24
4.1	Klijent-poslužitelj arhitektura, Izvor: https://en.wikipedia.org/wiki/Client-server_model	26
4.2	Spring Boot višeslojna arhitektura, Izvor: https://www.geeksforgeeks.org/introduction-to-spring-boot	28
4.3	Spring Boot - 6 slojeva arhitekture, Izvor: https://www.slideshare.net/alimenkou/hexagonal-architecture-with-spring-boot-136745841	29
4.4	Baza podataka	36
4.5	Dijagram razreda	37
4.6	Sloj domene	38
4.7	Sloj za pristup podacima	39
4.8	Sloj usluge	40
4.9	Sloj nadglednika	41
4.10	DTO razredi	42
4.11	REST dijagram	42
4.12	Dijagram stanja-zaposlenik	43
4.13	Dijagram stanja-građanin	44
4.14	Dijagram aktivnosti	45
4.15	Dijagram komponenti	46
5.1	JUnit 4 testovi	53
5.2	Upozorenje	54

5.3	Upis slova u komponentu za vrijeme	55
5.4	Zahtjev za resursima	56
5.5	Pronalazak odgovarajućeg tipa odlagališta	57
5.6	Dijagram razmještaja	58

Dodatak: Prikaz aktivnosti grupe

Dnevnik sastajanja

1. sastanak

- Datum: 9. listopada 2019.
- Prisustvovali: Marko Cirimotić, Robert Glivarec, Stjepan Kovačić, Luka Mandić, Marin Ovčariček, Edi Štambuk
- Teme sastanka:
 - Upoznavanje članova tima
 - Zajednički pregled zadatka

2. sastanak

- Datum: 10. listopada 2019.
- Prisustvovali: Marko Cirimotić, Robert Glivarec, Bruno Horvat, Stjepan Kovačić, Luka Mandić, Marin Ovčariček, Edi Štambuk
- Teme sastanka:
 - Odabir tehnologija u kojima ćemo raditi
 - Podjela članova na backend i frontend dio
 - Dodatni pregled i rasprava o zadatku

3. sastanak

- Datum: 29. listopada 2019.
- Prisustvovali: Robert Glivarec, Bruno Horvat, Stjepan Kovačić, Luka Mandić, Marin Ovčariček, Edi Štambuk
- Teme sastanka:
 - Rješavanje problema oko izrade dokumentacije (LaTeX, TeXstudio, TeX Live)
 - Razgovor i rješavanje pojedinih problema u razvoju generičke funkcionalnosti aplikacije

Tablica aktivnosti

	Stjepan Kovačić	Marko Cirimotić	Robert Glivarec	Bruno Horvat	Luka Mandić	Marin Ovčariček	Edi Štambuk
Upravljanje projektom	5						
Opis projektnog zadatka							3
Funkcionalni zahtjevi		3					
Opis pojedinih obrazaca							
Dijagram obrazaca				3	4		
Sekvencijski dijagrami				2	2		
Opis ostalih zahtjeva		2					
Arhitektura i dizajn sustava	3						
Baza podataka	4		7				
Dijagram razreda	3					3	
Dijagram stanja			2				
Dijagram aktivnosti				2			
Dijagram komponenti					2		2
Korištene tehnologije i alati	2						
Ispitivanje programskog rješenja							2
Dijagram razmještaja				1			
Upute za puštanje u pogon	2						
Dnevnik sastajanja							
Zaključak i budući rad	1						
Popis literature							

	Stjepan Kovčić	Marko Cirimotić	Robert Glivarec	Bruno Horvat	Luka Mandić	Marin Ovčariček	Edi Štambuk
Izrada početne stranice							1
Izrada stranice za prijavu							1
Izrada stranice za registraciju							1
Autentikacija i autorizacija	8					8	3
Registracija-backend	1					1	
Izrada informacijskih stranica		2	17				10
Stranica za popis korisnika							3
Stranica za zahtjeve i pritužbe		20		18		1	
Backend - izrada API-ja	35						
Deployment aplikacije i BP	3						
Rad na bazi podataka	3						