

Programsko inženjerstvo

Ak. god. 2023./2024.

Štete javnih površina

Dokumentacija, Rev. 2

Grupa: *CestaFix*

Voditelj: *Fran Fodor*

Datum predaje: **19. 1. 2024.**

Nastavnik: *izv. prof. dr. sc. Vlado Sruk*

Sadržaj

1. Dnevnik promjena dokumentacije

Rev.	Opis promjene/dodatka	Autori	Datum
0.1	Preuzet predložak.	Sara Podvorec	28.10.2023.
0.2	Dodane osnovne informacije. Napisan početak opisa projekta. Crvenom bojom označeni dijelovi teksta koje treba izmijeniti.	Jan Murić	28.10.2023.
0.3	Nadopuna opisa projekta.	Sara Podvorec	29.10.2023.
0.4	Nadopuna osnovnih informacija. Nadopuna dnevnika sastajanja.	Mateo Jakšić	30.10.2023.
0.5	Stiliziranje opisa projekta.	Sara Podvorec	30.10.2023.
0.6	Nadopuna dnevnika sastajanja.	Mateo Jakšić	31.10.2023.
0.7	Nadopuna opisa projektnog zadatka i funkcionalnih zahtjeva.	Sara Podvorec	31.10.2023.
0.8	Nadopuna literature i funkcionalnih zahtjeva.	Mateo Jakšić	1.11.2023.
0.9	Napisan dio obrazaca uporabe	Mateo Jakšić	1.11.2023.
0.10	Nadopuna obrazaca uporabe	Mateo Jakšić	2.11.2023.
0.11	Nadopuna obrazaca uporabe. Dodani dijagrami obrazaca uporabe.	Mateo Jakšić	6.11.2023.
0.12	Dodani sekvencijski dijagrami. Dodani ostali zahtjevi. Nadopuna dnevnika sastajanja.	Mateo Jakšić	10.11.2023.

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

Rev.	Opis promjene/dodataka	Autori	Datum
0.13	Dodane osnovne informacije o arhitekturi. Nadopuna opisa baze podataka.	Sara Podvorec	12.11.2023.
0.14	Dodan opis pojedinih entiteta baze podataka.	Sara Podvorec	13.11.2023.
0.15	Nadopuna dnevnika sastajanja. Nadopuna opisa i kreiranje ERD dijagrama baze podataka.	Mateo Jakšić	14.11.2023.
0.16	Dodatan ER dijagram baze podataka	Mateo Jakšić	15.11.2023.
0.17	Dodan opis arhitekture. Dodan vizualni prikaz arhitekture.	Sara Podvorec	16.11.2023.
0.18	Dodani dijagrami razreda i njihovi opisi. Nadopuna literature.	Mateo Jakšić	16.11.2023.
1.0	Uklanjanje uputa. Završne nadopune.	Mateo Jakšić	17.11.2023.
1.1	Napisane korištene tehnologije i alati. Nadopunjten dnevnik sastajanja. Prepravljanje osnovnih informacija.	Mateo Jakšić	10.12.2023.
1.2	Prepravljanje baze podataka.	Mateo Jakšić	1.1.2024.
1.3	Prepravljanje baze podataka i dijagrama razreda.	Mateo Jakšić	8.1.2024.
1.4	Dodan dijagram aktivnosti.	Sara Podvorec	12.1.2024.
1.5	Prepravljeni opisi baze podataka i dijagrama razreda.	Mateo Jakšić	13.1.2024.
1.6	Dodano ispitivanje sustava.	Fran Fodor	15.1.2024.
1.7	Dodan dijagram komponentni.	Mateo Jakšić	16.1.2024.
1.8	Dodan prepravljeni dijagram aktivnosti.	Sara Podvorec	16.1.2024.

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

Rev.	Opis promjene/dodataka	Autori	Datum
1.9	Dodan dijagram razmještaja.	Mateo Jakšić	16.1.2024.
1.10	Dodani opisi dijagrama komponenti i dijagrama razmještaja.	Mateo Jakšić	16.1.2024.
1.11	Dodan dijagram stanja.	Sara Podvorec	17.1.2024.
1.12	Dodane upute za puštanje u pogon.	Fran Fodor	18.1.2024.
1.13	Dodano integracijsko testiranje.	Ante Prkačin	18.1.2024.
1.14	Dodani opisi dijagrama stanja i dijagrama aktivnosti.	Sara Podvorec	18.1.2024.
1.15	Nadopunjene korištene tehnologije i alati, dijagrami razreda. Dodan zaključak i budući rad.	Mateo Jakšić	18.1.2024.
1.16	Prepravljeni dijagrami razreda.	Mateo Jakšić	19.1.2024.
1.17	Prepravljen dijagram stanja i nadopunjjen njegov opis.	Sara Podvorec	19.1.2024.

2. Opis projektnog zadatka

2.1 Opis projektnog zadatka 'Štete javnih površina'

Svakodnevno se suočavamo s brojnim problemima na javnim površinama i cestama u našim gradovima. Problemi kao što su vandalizam, oštećenje pločnika, udarne rupe na cestama, smeće i slično predstavljaju potencijalnu opasnost za građane te poprilično narušavaju kvalitetu njihovog svakodnevnog života.

Neadekvatna briga o oštećenjima i njihovoj sanaciji često ostavlja građane u vrlo nepovoljnoj situaciji kada iste žele prijaviti vlastima. Kako bi suzbili osjećaj nemoći građana i poboljšali kvalitetu života cjelokupne zajednice, potrebno je omogućiti adekvatnu prijavu oštećenja na javnim površinama.

Glavni je cilj ovog projekta razviti programsku podršku za stvaranje web aplikacije CestaFix za dojavu oštećenja i drugih problema na cestama, parkovima, javnim ustanovama i ostalim javnim mjestima kako bi se olakšala dojava, kategorizacija te u konačnici rješenja prijavljenih problema. Ideja je omogućiti građanima da na što jednostavniji način prijavljuju oštećenja i probleme na javnim površinama i cestama svojih gradova te tako pomognu lokalnim vlastima da pravodobno reagiraju na nastale probleme.

S obzirom na velik broj javih ustanova i njihovih odjeljaka koji se bave različitim problemima, teško je pratiti tko je nadležan za koju vrstu problema i nad kojim područjem. Sustav bi automatski odredio nadležno tijelo prema kategoriji prijave i drugim parametrima te proslijedio prijavu na obradu. Ključno je da proces prijave bude jednostavan i brz. Prilikom pokretanja sustava prikazuje se početna stranica na kojoj se nalazi karta s označenim lokacijama već podnesenih prijava raznovrsnih oštećenja javnih površina i navigacijski izbornik putem kojeg se pristupa registraciji/prijavi u sustav, podnošenju prijava, provjeri statusa već podnesenih prijava, često postavljanim pitanjima te informacijama o samoj aplikaciji. Svakom neregistriranom korisniku omogućeno je kreiranje novog računa prilikom čega je potrebno unijeti:

- *korisničko ime*
- *lozinku*

- *email adresu*

Ako korisnik već ima postojeći račun, prilikom prijave u sustav unosi:

- *email adresu*
- *lozinku*

Registrirani korisnik može pregledavati i mijenjati svoje osobne podatke i izbrisati svoj korisnički račun te postoji mogućnost naknadne dodjele prava službenika ili administratora stupanjem u kontakt naveden pri dnu stranice.

Također, postoji opcija anonimnog podnošenja prijave na temelju koje se status podnesene prijave prati putem jedinstvenog broja prijave bez iznošenja osobnih podataka i potrebe za registracijom odnosno prijavom.

Korisnik s pravom službenika ima mogućnost ažuriranja statusa prijava i pre-gled profila prijavitelja (ili jedinstvenog broja prijave u slučaju anonimnog prijavitelja) među kojima odabire relevantne prijave. Nakon što sustav proslijedi prijavu u nadležni gradski ured, službenik tog ureda može izabrati problem i krenuti raditi na njegovu rješenju.

Administrator sustava ima najveće ovlasti među koje pripada i pristup bazi s po-pisom registriranih korisnika i njihovim podacima te mogućnost brisanja korisnika kao i dodjeljivanje administratorskih prava i prava službenika drugim korisnicima. Kako bi olakšala proces podnošenja prijave, aplikacija omogućuje građanima da brzo i precizno dokumentiraju probleme. Svaka prijava uključuje naziv problema, kratki opis, geografske koordinate problema, jedinstveni broj prijave te optionalno fotografije kako bi se problem što bolje opisao. Sustav također olakšava identifikaciju vremenski bliskih prijava na istoj lokaciji kako bi se građani mogli povezati na postojeće prijave sličnih problema, čime se smanjuje duplicitanje prijava i ubrzava proces njihovog rješavanja. Podnesene prijave obrađuju određeni gradski uredi, a svaka promjena u statusu prijave vidljiva je prijavitelju i svim ostalim korisnicima. Gradski uredi također imaju mogućnost objediniti nepovezane prijave na istoj lokaciji, što je vidljivo prijaviteljima. Kako bi gradskim uredima olakšali razvrstavanje i obrađivanje prijava, sustav dopušta prijave za različite kategorije problema. Osnovne kategorije problema su:

- *oštećenja na cesti*
- *oštećenja na vodovodnoj infrastrukturi*
- *oštećenja na zelenim površinama*

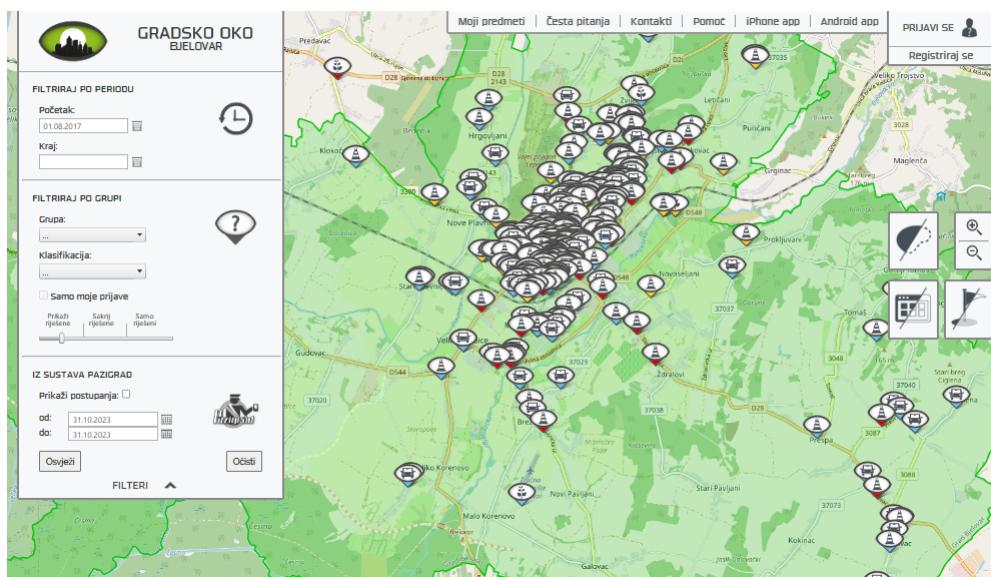
- oštećenja na elektroenergetskoj infrastrukturi

Sve podnesene prijave su javno dostupne i mogu se grupirati prema temi i lokaciji, što omogućuje građanima i vlastima da prate i analiziraju probleme u stvarnom vremenu.

Statistika dosadašnjih prijava obrađuje se u stvarnom vremenu kako bi poboljšala učinkovitost sustava i pomogla gradskim vlastima u praćenju učinkovitosti njihovih usluga. Sastoje se od broja prijava oštećenja koje čekaju obradu, broja prijava oštećenja koje su u obradi i broja prijava oštećenja koje su obrađene.

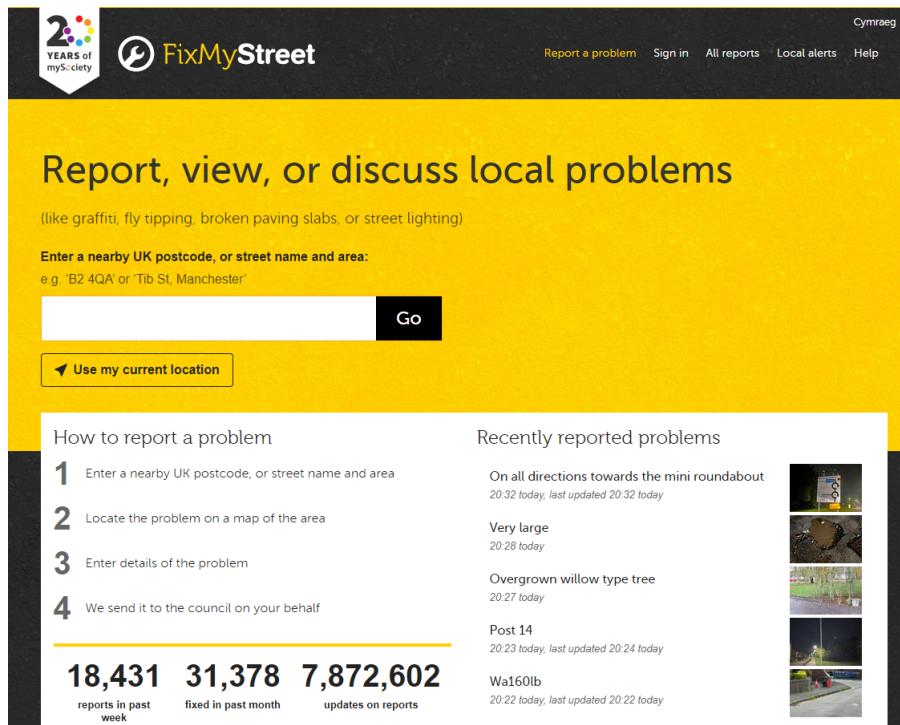
Posebna je pozornost posvećena zaštiti svih osobnih podataka te osiguranju korisničkog iskustva koje će biti jednostavno i intuitivno, kako bi se potaknulo građane na aktivno sudjelovanje u poboljšanju svojih zajednica.

Skup korisnika koji bi mogao biti zainteresiran za ovu aplikaciju je vrlo širok. To uključuje građane koji žele prijaviti probleme na javnim površinama, lokalne grade dove i njihove urede koji će obraditi prijave, službenike koji će nadzirati i rješavati prijave te administratore koji će upravljati sustavom i osigurati sigurnost podataka. Jedan od primjera sličnog sustava koji se aktivno upotrebljava u Hrvatskoj je "Gradsko Oko". Projekt je pokrenut u kolovozu 2017. godine u svrhu prijave komunalnih problema na području grada Bjelovara, a od tад se proširio na još 11 gradova i općina te na prijavu problema na moru.



Slika 2.1: Početna stranica sustava "Gradsko Oko"

Sustav "FixMyStreet" koji se koristi na području Ujedinjenog Kraljevstva također omogućava građanima da prijave probleme na javnim površinama i cestama u svojim lokalnim zajednicama kao i prikaz statističkih podataka o prijavljenim problemima, vremenskim okvirima za rješavanje i druge korisne informacije.



Slika 2.2: Početna stranica sustava "FixMyStreet"

Postojeća slična rješenja, kao što su "Gradsko Oko" i "FixMyStreet", pružaju mogućnost prijave komunalnih problema, ali se razlikuju u nekoliko ključnih aspekata. Naša aplikacija ima određene značajke koje ju čine jedinstvenom, posebno u pogledu anonimnih prijava, statistike i praćenja rada gradskih ureda te povezivanja vremenski bliskih prijava.

Ova web aplikacija ima potencijal za prilagodbu i nadogradnje u budućnosti. Moguće nadogradnje uključuju implementaciju mobilne aplikacije kako bi se olakšao proces prijave, dodatne značajke za praćenje rada gradskih ureda, uvođenje sustava za nagrađivanje korisnika koji aktivno sudjeluju u rješavanju problema te proširenje na druge regije.

3. Specifikacija programske potpore

3.1 Funkcionalni zahtjevi

Dionici:

1. Neregistrirani korisnik
2. Registrirani korisnik
 - (a) Građanin
 - (b) Službenik gradskog ureda
3. Administrator
4. Razvojni tim
5. Suradnici
 - (a) Nastavnik
 - (b) Demonstrator

Aktori i njihovi funkcionalni zahtjevi:

1. Neregistrirani korisnik (inicijator) može:
 - (a) podnosiću anonimne prijave
 - (b) pratiti status prijave jedinstvenim brojem prijave
 - (c) imati uvid u postojeće prijave
 - (d) imati uvid u statistiku statusa prijava
 - (e) registrirati korisnički račun
2. Građanin (inicijator) može:
 - (a) prijaviti se na korisnički račun
 - (b) odjaviti se s korisničkog računa
 - (c) pratiti status vlastitih prijava
 - (d) imati uvid u postojeće prijave
 - (e) imati uvid u statistiku statusa prijava
 - (f) izmijeniti podatke korisničkog računa

3. Službenik gradskog ureda (inicijator) može:

- (a) obrađivati podnesene prijave
- (b) objediniti nepovezane prijave po temi i lokaciji
- (c) promijeniti status prijave
- (d) pratiti i analizirati probleme u stvarnom vremenu
- (e) izmijeniti podatke korisničkog računa

4. Administrator (inicijator) može:

- (a) uređivati i brisati korisničke račune
- (b) pristupiti bazi podataka
- (c) kreirati i ukloniti gradski ured
- (d) dodijeliti i ukloniti ulogu službenika gradskog ureda registriranom korisniku
- (e) dodijeliti i ukloniti ulogu administratora registriranom korisniku

5. Baza podataka (sudionik) može:

- (a) pohraniti sve podatke o korisnicima i njihovim ulogama
- (b) pohraniti sve podatke o prijavljenim oštećenjima
- (c) pohraniti sve podatke o kategorijama problema i gradskim uredima za njihovo obrađivanje

3.1.1 Obrasci uporabe

Opis obrazaca uporabe

UC01 - Registracija korisničkog računa

- **Glavni sudionik:** Neregistrirani korisnik
- **Cilj:** Stvaranje korisničkog račun za pristup sustavu
- **Sudionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
 1. Korisnik odabire opciju za prijavu
 2. Korisnik odabire opciju da nema korisnički račun
 3. Korisnik unosi potrebne podatke i potvrđuje unos
 4. Korisnik se upisuje u bazu podataka i korisnik se preusmjerava na početnu stranicu
- **Opis mogućih odstupanja:**
 - 3.a Odabir već zauzete e-mail adrese, unos korisničkih podataka u neispravnom formatu, nepodudaranje lozinki
 1. Sustav obavještava korisnika o neuspjelom unosu i vraća ga na stranicu za registraciju
 2. Korisnik mijenja potrebne podatke te završava unos ili odustaje od registracije

UC02 - Prijava na korisnički račun

- **Glavni sudionik:** Registrirani korisnik
- **Cilj:** Dobivanje pristupa korisničkom sučelju s ovlastima registriranog korisnika
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je obavio registraciju i njegovi podaci se nalaze u bazi podataka
- **Opis osnovnog tijeka:**
 1. Korisnik odabire opciju za prijavu
 2. Korisnik unosi email i lozinku te potvrđuje unos
 3. Korisnik se preusmjerava na početnu stranicu
- **Opis mogućih odstupanja:**
 - 2.a Neispravni unos emaila i/ili lozinke

1. Sustav obavještava korisnika o neuspjelom unosu i vraća ga na stranicu za prijavu
2. Korisnik unosi ispravne podatke te završava unos ili odustaje od prijave

UC03 - Odjava s korisničkog računa

- **Glavni sudionik:** Registrirani korisnik
- **Cilj:** Izlazak iz korisničkog sučelja
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je obavio prijavu na korisnički račun
- **Opis osnovnog tijeka:**
 1. Korisnik postavlja pokazivač iznad svojeg imena
 2. Korisnik odabire opciju za odjavu na padajućem izborniku
 3. Korisnik se preusmjerava na početnu stranicu

UC04 - Kreiranje gradskog ureda

- **Glavni sudionik:** Administratori
- **Cilj:** Kreiranje gradskog ureda
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen i ima administrator ovlasti
- **Opis osnovnog tijeka:**
 1. Korisnik otvara padajući izbornik
 2. Korisnik odabire opciju za kreiranje gradskog ureda
 3. Korisnik unosi potrebne podatke i potvrđuje unos
 4. Gradska ured se upisuje u bazu podataka i korisnik se preusmjerava na početnu stranicu
- **Opis mogućih odstupanja:**
 - 3.a Odabir već zauzetog imena gradskog ureda
 1. Sustav obavještava korisnika o neuspjelom unosu i vraća ga na stranicu za kreiranje gradskog ureda
 2. Korisnik mijenja potrebne podatke te završava unos ili odustaje od kreiranja gradskog ureda
 - 3.b Nije odabrana nijedna kategorija oštećenja kojom će gradski ured upravljati
 1. Sustav obavještava korisnika o obveznom unosu barem jedne kategorije oštećenja kojom će ured upravljati

2. Korisnik odabire barem jednu kategoriju oštećenja te završava unos ili odustaje od kreiranja gradskog ureda

UC05 - Uklanjanje gradskog ureda

- **Glavni sudionik:** Administrator
- **Cilj:** Uklanjanje gradskog ureda
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen i ima administrator ovlasti, gradski ured postoji
- **Opis osnovnog tijeka:**
 1. Korisnik otvara padajući izbornik
 2. Korisnik odabire opciju za pregled gradskih ureda
 3. Korisnik odabire gradski ured
 4. Korisnik odabire opciju za uklanjanje gradskog ureda
 5. Gradski ured se uklanja iz baze podataka i korisnik se preusmjerava na stranicu za pregled gradskih ureda

UC06 - Dodjela uloge službenika gradskog ureda

- **Glavni sudionik:** Administrator
- **Cilj:** Dodjeljivanje uloge službenika gradskog ureda registriranom korisniku
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen i ima administratorske ovlasti, postoji odgovarajući gradski ured za registriranog korisnika
- **Opis osnovnog tijeka:**
 1. Korisnik otvara padajući izbornik
 2. Korisnik odabire opciju za pregled popisa korisničkih računa
 3. Korisnik odabire korisnički račun
 4. Korisnik odabire opciju za dodjelu uloge službenika gradskog ureda
 5. Registrirani korisnik dobiva ulogu službenika gradskog ureda i korisnik se preusmjerava na pregled korisničkog računa

UC07 - Pregled korisničkog računa

- **Glavni sudionik:** Registrirani korisnik
- **Cilj:** Pregledavanje korisničkog račun
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je obavio prijavu na korisnički račun

- **Opis osnovnog tijeka:**

1. Korisnik odabire opciju za pregled vlastitog korisničkog računa
2. Korisnik se preusmjerava na stranicu za pregled korisničkog računa

UC08 - Uređivanje podataka korisničkog računa

- **Glavni sudionik:** Registrirani korisnik

- **Cilj:** Promjena podataka na vlastitom korisničkom računu

- **Sudionici:** Baza podataka

- **Preduvjet:** Korisnik je obavio prijavu na korisnički račun

- **Opis osnovnog tijeka:**

1. Korisnik odabire opciju za pregled vlastitog korisničkog računa
2. Korisnik odabire opciju za uređivanje korisničkih podataka
3. Korisnik uređuje korisničke podatke i potvrđuje promjene
4. Korisnički podaci se ažuriraju u bazi podataka i korisnik se preusmjerava na pregled vlastitog korisničkog računa

UC09 - Brisanje korisničkog računa

- **Glavni sudionik:** Registrirani korisnik

- **Cilj:** Uklanjanje vlastitog korisničkog računa

- **Sudionici:** Baza podataka

- **Preduvjet:** Korisnik je obavio prijavu na korisnički račun

- **Opis osnovnog tijeka:**

1. Korisnik odabire opciju za pregled vlastitog korisničkog računa
2. Korisnik odabire opciju za brisanje korisničkog računa
3. Korisnik povrđuje odluku i korisnik se preusmjerava na početnu stranicu

UC10 - Kreiranje anonimne prijave oštećenja

- **Glavni sudionik:** Neregistrirani korisnik

- **Cilj:** Prijava oštećenja

- **Sudionici:** Baza podataka

- **Preduvjet:** -

- **Opis osnovnog tijeka:**

1. Korisnik odabire opciju za prijavu oštećenja
2. Korisnik unosi potrebne podatke i potvrđuje unos

3. Prijava oštećenja se upisuje u bazu podataka i korisnik dobiva jedinstveni broj prijave

- **Opis mogućih odstupanja:**

- 2.a Neuneseni obvezni podaci i/ili podaci uneseni u neispravnom formatu
 1. Sustav obavještava korisnika o neuspjelom unosu i vraća ga na stranicu za prijavu oštećenja
 2. Korisnik mijenja potrebne podatke te završava unos ili odustaje od prijave oštećenja

UC11 - Kreiranje prijave oštećenja

- **Glavni sudionik:** Registrirani korisnik

- **Cilj:** Prijava oštećenja

- **Sudionici:** Baza podataka

- **Preduvjet:** Korisnik je prijavljen

- **Opis osnovnog tijeka:**

1. Korisnik odabire opciju za prijavu oštećenja
2. Korisnik unosi potrebne podatke i potvrđuje unos
3. Prijava oštećenja se upisuje u bazu podataka i korisnik se preusmjerava na početnu stranicu

- **Opis mogućih odstupanja:**

- 2.a Neuneseni obvezni podaci i/ili podaci uneseni u neispravnom formatu
 1. Sustav obavještava korisnika o neuspjelom unosu i vraća ga na stranicu za prijavu oštećenja
 2. Korisnik mijenja potrebne podatke te završava unos ili odustaje od prijave oštećenja

UC12 - Provjera statusa anonimne prijave oštećenja

- **Glavni sudionik:** Neregistrirani korisnik

- **Cilj:** Provjera statusa anonimne prijave oštećenja

- **Sudionici:** Baza podataka

- **Preduvjet:** Podnesena prijava oštećenja

- **Opis osnovnog tijeka:**

1. Korisnik odabire opciju za provjeru statusa prijave oštećenja
2. Korisnik unosi jedinstveni broj prijave i potvrđuje unos

- **Opis mogućih odstupanja:**

- 2.a Jedinstveni broj prijave ne postoji ili je unesen u neispravnom formatu

1. Sustav obavještava korisnika o neuspjelom unosu i vraća ga na stranicu za provjeru statusa prijave oštećenja
2. Korisnik unosi ispravni jedinstveni broj prijave i završava unos ili odustaje od provjere statusa prijave oštećenja

UC13 - Provjera statusa prijave oštećenja

- **Glavni sudionik:** Registrirani korisnik
- **Cilj:** Provjera statusa vlastite prijave oštećenja
- **Sudionici:** Baza podataka
- **Preduvjet:** Podnesena prijava oštećenja, korisnik je prijavljen
- **Opis osnovnog tijeka:**
 1. Korisnik odabire opciju za pregled vlastitog korisničkog računa
 2. Korisnik odabire opciju za pregled vlastitih prijave oštećenja i provjerava status željene prijave oštećenja

UC14 - Pregled prijave oštećenja

- **Glavni sudionici:** Registrirani korisnik, Neregistrirani korisnik
- **Cilj:** Pregled prijave oštećenja
- **Sudionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
 1. Korisnik odabire opciju za pregled vlastitog korisničkog računa
 2. Korisnik odabire opciju za pregled vlastitih prijave oštećenja
 3. Korisniku odabire prijavu oštećenja i preusmjerava se na pregled prijavljenog oštećenja

UC15 - Promjena statusa prijave oštećenja

- **Glavni sudionik:** Službenik gradskog ureda
- **Cilj:** Promjena statusa prijave oštećenja
- **Sudionici:** Baza podataka
- **Preduvjet:** Podnesena prijava oštećenja, korisnik ima ulogu službenika gradskog ureda
- **Opis osnovnog tijeka:**
 1. Korisnik odabire opciju za pregled vlastitog korisničkog računa
 2. Korisnik odabire opciju za ažuriranje statusa prijave oštećenja

3. Korisnik odabire opciju za promjenu statusa prijave oštećenja i potvrđuje promjenu
4. Status prijave oštećenja se ažurira u bazi podataka i korisnik dobiva potvrdu uspješne promjene

UC16 - Provjera statistike statusa prijave oštećenja

- **Glavni sudionici:** Neregistrirani korisnik, Registrirani korisnik
- **Cilj:** Provjera statistike statusa prijave oštećenja
- **Sudionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
 1. Korisnik odabire opciju za provjeru statistike statusa prijava oštećenja
 2. Korisniku se otvara pregled statistike statusa prijava oštećenja

UC17 - Objedinjavanje nepovezanih prijava oštećenja

- **Glavni sudionik:** Službenik gradskog ureda
- **Cilj:** Objediniti nepovezane prijave oštećenja po temi i lokaciji
- **Sudionici:** Baza podataka
- **Preduvjet:** Postoje prijave oštećenja s istom temom i lokacijom, korisnik ima ulogu službenika gradskog ureda
- **Opis osnovnog tijeka:**
 1. Korisnik odabire opciju za pregled vlastitog korisničkog računa
 2. Korisnik odabire opciju za objedinjavanje prijava oštećenja
 3. Korisnik odabire dvije ili više prijava oštećenja i potvrđuje odabir
 4. Prijave oštećenja se objedinjuju u bazi podataka i korisnik se učitava novo statusa problema
- **Opis mogućih odstupanja:**
 - 3.a Korisnik je odabrao manje od dvije prijave oštećenja i pokušao napraviti objedinjavanje
 1. Sustav obavještava korisnika o neuspjelom pokušaju objedinjavanja i vraća ga na odabir prijava oštećenja
 2. Korisnik odabire više od dvije prijave oštećenja i potvrđuje odabir ili odustaje od objedinjavanja prijava oštećenja

UC18 - Dodjela kategorije za upravljanje gradskom uredu

- **Glavni sudionik:** Administrator

- **Cilj:** Dodjela kategorije oštećenja kojom će određeni gradski ured upravljati
- **Sudionici:** Baza podataka
- **Preduvjet:** Gradski ured postoji, korisnik ima ulogu administratora
- **Opis osnovnog tijeka:**
 1. Korisnik odabire padajući izbornik
 2. Korisnik odabire opciju pregleda gradskih ureda
 3. Korisnik odabire gradski ured
 4. Korisnik odabire opciju za dodavanje kategorija oštećenja
 5. Korisnik radi odabir kategorija i potvđuje odabir
 6. Ažurirani popis kategorija oštećenja za koje je zadužen gradski ured se upisuje u bazu podataka i korisnik se preusmjerava na pregled profila gradskog ureda
- **Opis mogućih odstupanja:**
 - 4.a Gradski ured već upravlja svim kategorijama oštećenja, ne postoji kategorija oštećenja kojom već ne upravlja neki gradski ured u istom mjestu
 1. Sustav obavještava korisnika da ne postoji kategorija oštećenja kojom gradski ured već ne upravlja ili da ne postoji kategorija kojom već ne upravlja neki gradski ured u istom mjestu
 2. Korisnik odustaje od dodjele kategorije za upravljanje gradskom uredu

UC19 - Uklanjanje kategorije za upravljanje iz gradskog ureda

- **Glavni sudionik:** Administrator
- **Cilj:** Uklanjanje kategorije oštećenja kojom trenutno upravlja gradski ured
- **Sudionici:** Baza podataka
- **Preduvjet:** Gradski ured postoji, korisnik ima ulogu administratora
- **Opis osnovnog tijeka:**
 1. Korisnik odabire padajući izbornik
 2. Korisnik odabire opciju pregleda gradskih ureda
 3. Korisnik odabire gradski ured
 4. Korisnik odabire opciju da uklanjanje kategorija oštećenja
 5. Korisnik radi odabir kategorija i potvđuje ga
 6. Ažurirani popis kategorija oštećenja za koje je zadužen gradski ured se upisuje u bazu podataka i korisnik se preusmjerava na pregled profila gradskog ureda
- **Opis mogućih odstupanja:**
 - 5.a Korisnik je odabrao sve kategorije oštećenja

1. Sustav obavještava korisnika da je odabrao sve kategorije oštećenja i vraća ga na stranicu za uređivanje gradskog ureda s otvorenom opcijom odabira kategorija
2. Korisnik odabire kategorije pazeći da ostavi barem jednu kategoriju i potvrđuje odabir ili odustaje od uklanjanja kategorija oštećenja kojima upravlja gradski ured

UC20 - Ponovno postavljanje zaboravljene lozinke

- **Glavni sudionik:** Registrirani korisnik
- **Cilj:** Stvaranje nove lozinke za registriranog korisnika
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je registriran i njegovi podaci se nalaze u bazi podataka
- **Opis osnovnog tijeka:**
 1. Korisnik odabire opciju za prijavu
 2. Korisnik odabire opciju za ponovno postavljanje zaboravljene lozinke
 3. Korisnik unosi e-mail adresu korisničkog računa i potvrđuje unos
 4. Korisnik dobiva personalizirani link na stranicu gdje mijenja lozinku
 5. Korisnik unosi novu lozinku i potvrđuje odabir
 6. Nova lozinka sprema se u bazu podataka i korisnik se preusmjerava na početnu stranicu
- **Opis mogućih odstupanja:**
 - 3.a Unesena e-mail adresa ne postoji ili je u neispravnom formatu
 1. Sustav obavještava korisnika da e-mail adresa ne postoji ili je u neispravnom formatu
 2. Korisnik unosi ispravnu e-mail adresu ili odustaje od ponovnog postavljanja zaboravljene lozinke
 - 4.a Uneseni identifikacijski broj ne postoji ili je u neispravnom formatu
 1. Sustav obavještava korisnika da identifikacijski broj ne postoji ili je u neispravnom formatu
 2. Korisnik unosi ispravni identifikacijski broj ili odustaje od ponovnog postavljanja zaboravljene lozinke
 - 5.a Unesena lozinka je u neispravnom formatu, lozinke se ne podudaraju
 1. Sustav obavještava korisnika da je unesena lozinka u neispravnom formatu ili da se lozinke ne podudaraju
 2. Korisnik unosi ispravno lozinku ili odustaje od ponovnog postavljanja zaboravljene lozinke

UC21 - Dodjela uloge administratora

- **Glavni sudionik:** Administrator
- **Cilj:** Dodjeljivanje uloge administratora registriranom korisniku
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen i ima administratorske ovlasti
- **Opis osnovnog tijeka:**
 1. Korisnik otvara padajući izbornik
 2. Korisnik odabire opciju za pregled popisa korisničkih računa
 3. Korisnik odabire korisnički račun
 4. Korisnik odabire opciju za dodjelu uloge administratora
 5. Registrirani korisnik dobiva ulogu administratora i korisnik se preusmjerava na pregled korisničkog računa

UC22 - Uklanjanje uloge službenika gradskog ureda

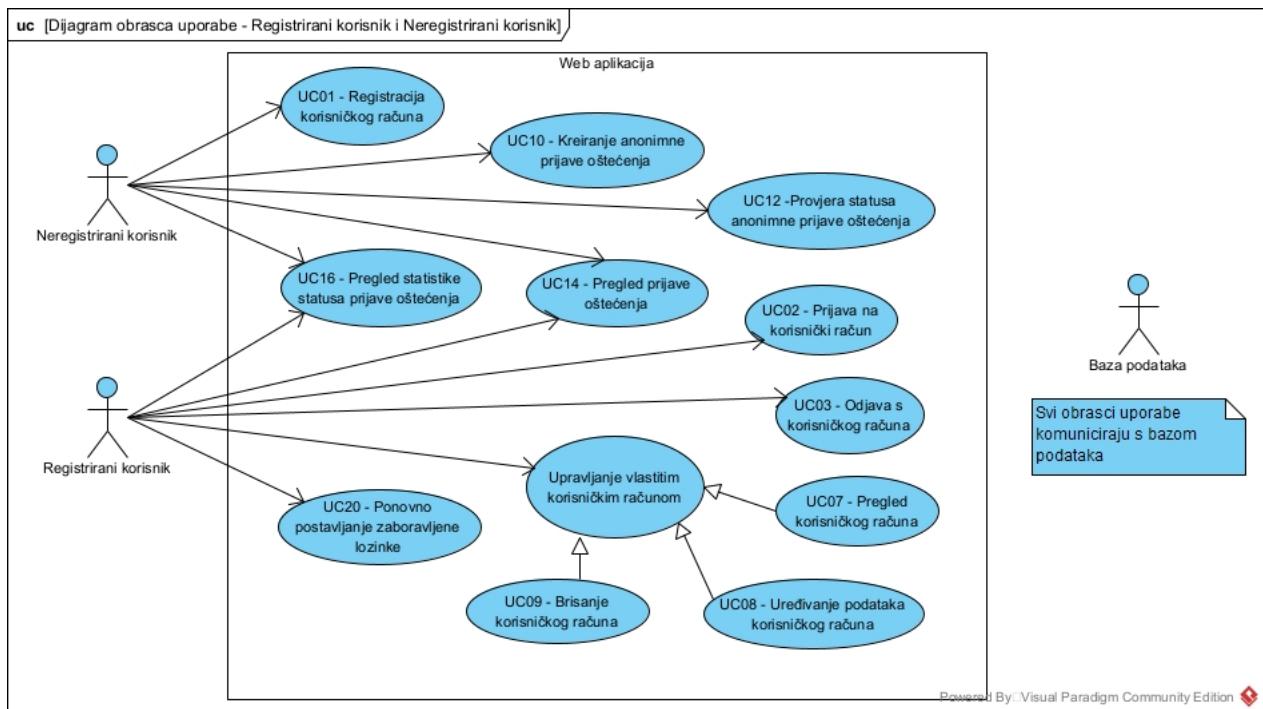
- **Glavni sudionik:** Administrator
- **Cilj:** Uklanjanje uloge službenika gradskog ureda registriranom korisniku
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen i ima administratorske ovlasti
- **Opis osnovnog tijeka:**
 1. Korisnik otvara padajući izbornik
 2. Korisnik odabire opciju za pregled popisa korisničkih računa
 3. Korisnik odabire korisnički račun
 4. Korisnik odabire opciju za uklanjanje uloge službenika gradskog ureda
 5. Registrirani korisnik ostaje bez uloge službenika gradskog ureda i korisnik se preusmjerava na pregled korisničkog računa

UC23 - Uklanjanje uloge administratora

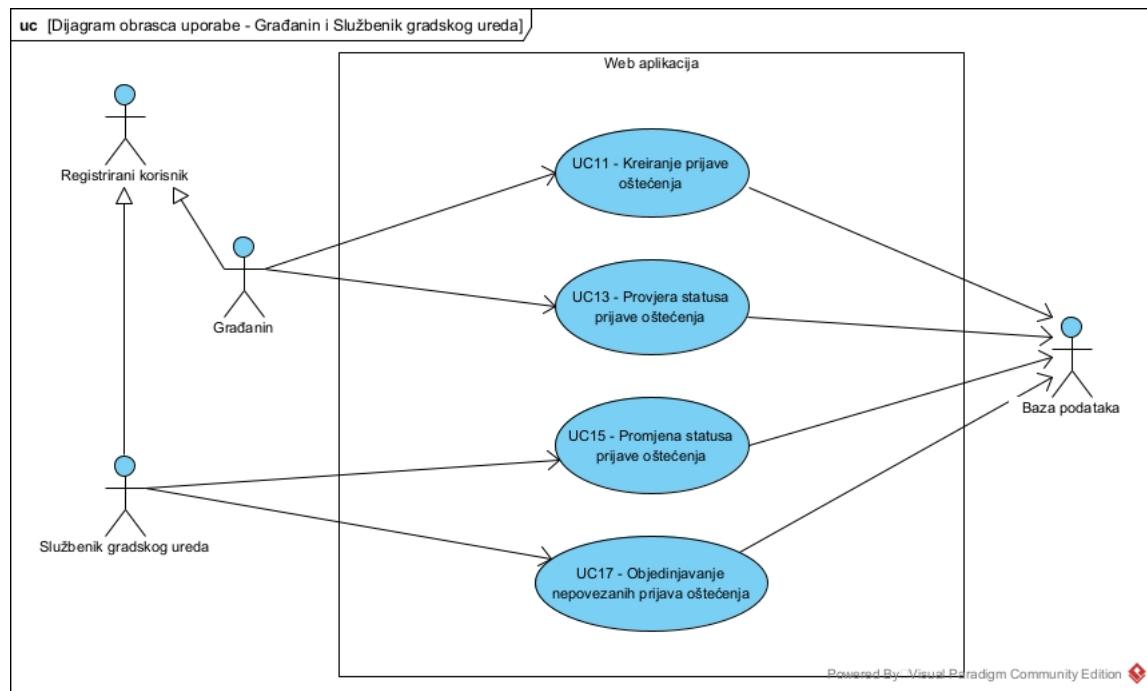
- **Glavni sudionik:** Administrator
- **Cilj:** Uklanjanje uloge administratora registriranom korisniku
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen i ima administratorske ovlasti
- **Opis osnovnog tijeka:**
 1. Korisnik otvara padajući izbornik
 2. Korisnik odabire opciju za pregled popisa korisničkih računa
 3. Korisnik odabire korisnički račun

4. Korisnik odabire opciju za uklanjanje uloge administratora
5. Registrirani korisnik ostaje bez uloge administratora i korisnik se preusmjerava na pregled korisničkog računa

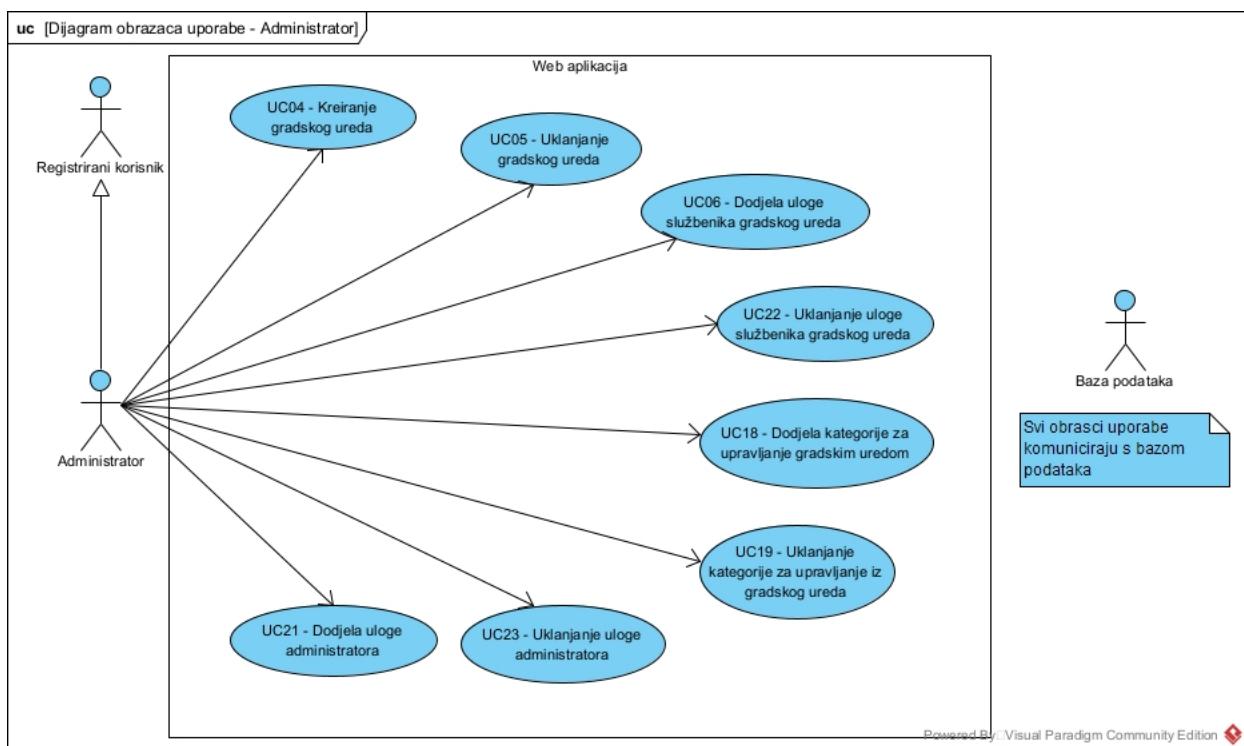
Dijagrami obrazaca uporabe



Slika 3.1: Dijagram obrasca uporabe, funkcionalnost registriranog i neregistriranog korisnika



Slika 3.2: Dijagram obrasca uporabe, funkcionalnost građanina i službenika gradskog ureda

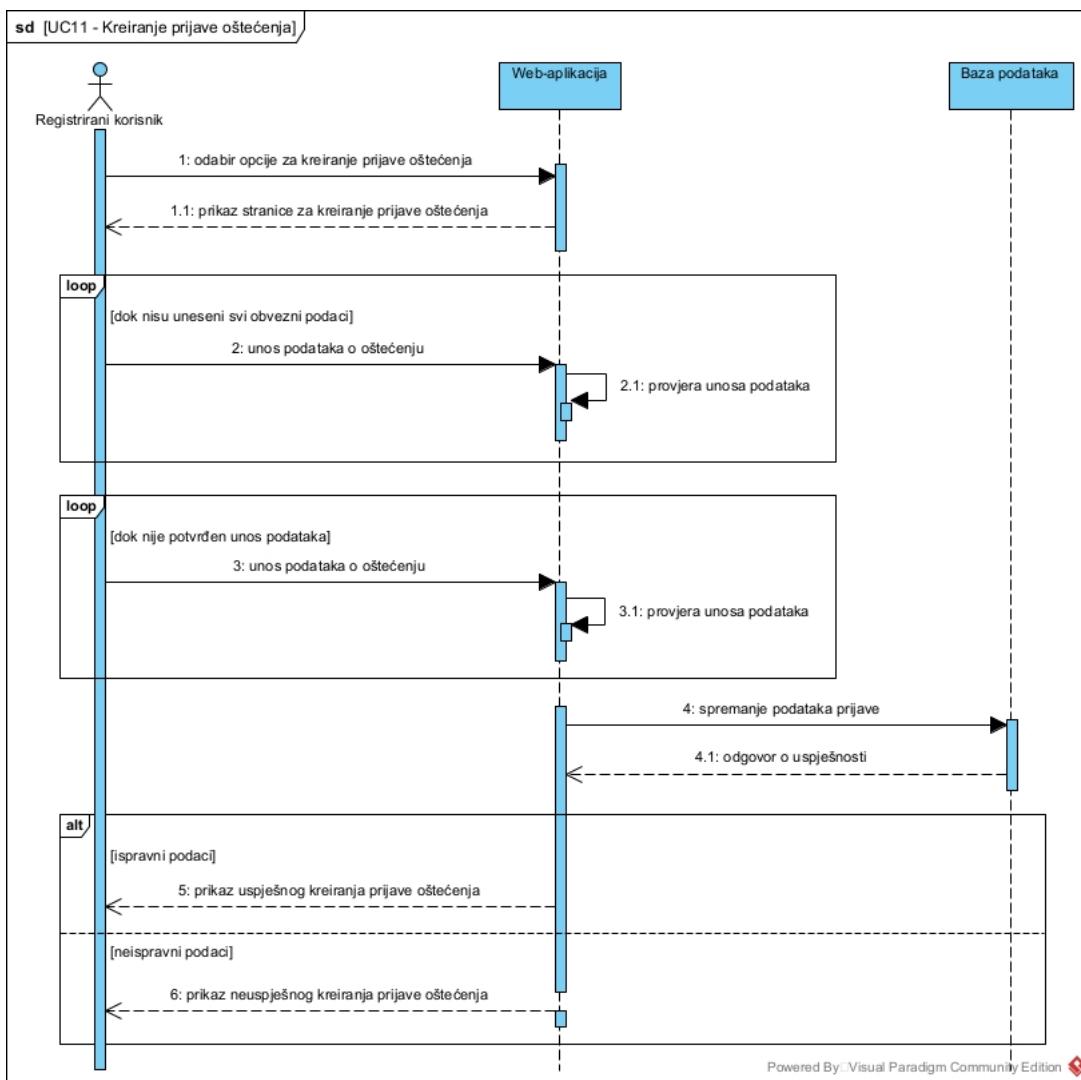


Slika 3.3: Dijagram obrasca uporabe, funkcionalnost administratora

3.1.2 Sekvencijski dijagrami

Obrazac uporabe UC11 - Kreiranje prijave oštećenja

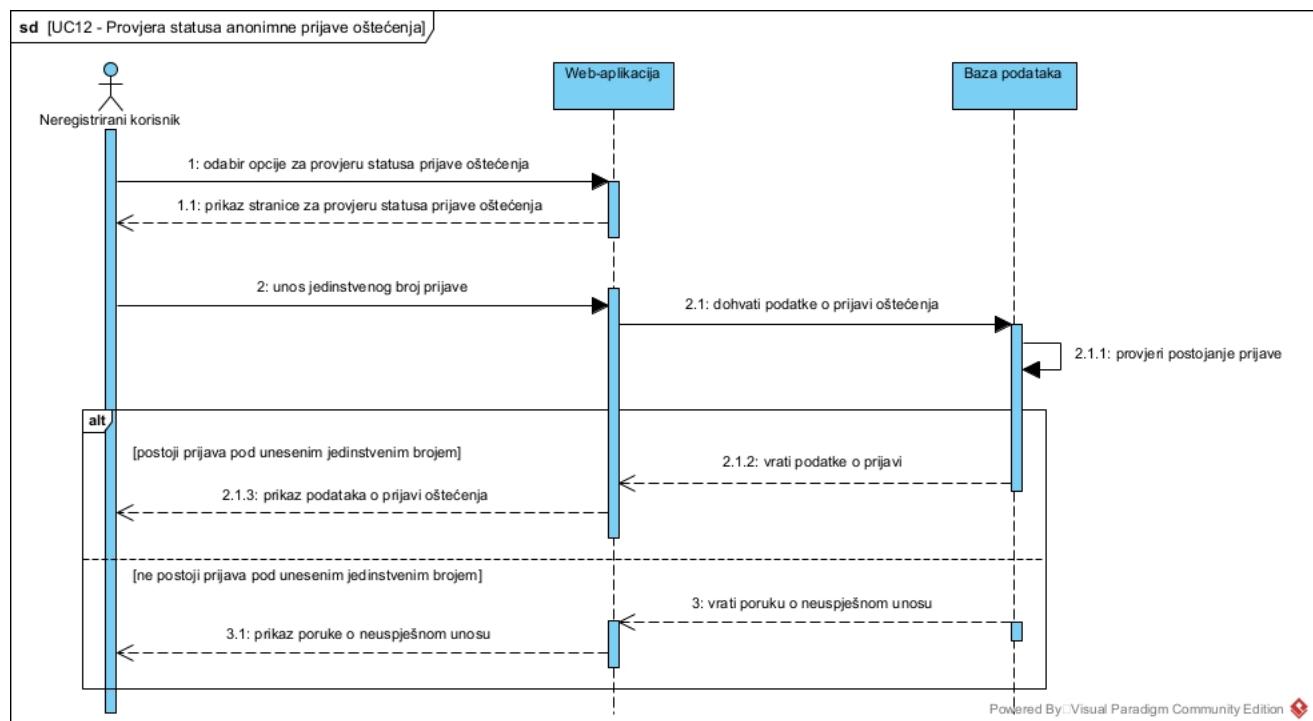
Registrirani korisnik šalje zahtjev za prikaz opcije kreiranja prijave oštećenja. Poslužitelj prikazuje stranicu za kreiranje prijave oštećenja na kojoj korisnik unosi podatke o oštećenju. Postoje obvezni i neobvezni podaci. Dok korisnik ne unese sve obvezne podatke ne prikazuje se opcija za potvrdu unosa podataka. Kada unese sve obvezne podatke prikaže se opcija za potvrđivanja unosa podataka. Korisnik odlučuje hoće li unijeti neobvezne podatke poput fotografije oštećenja. Potvrđeni podaci spremaju se u bazu podataka. Ukoliko su podaci ispravni, poslužitelj prikazuje prikaz uspješnog kreiranja prijave oštećenja, a ukoliko nisu, poslužitelj prikazuje prikaz neuspješnog kreiranje prijave oštećenja s obrazloženjem razloga.



Slika 3.4: Sekvencijski dijagram za UC11

Obrazac uporabe UC12 - Provjera statusa anonimne prijave oštećenja

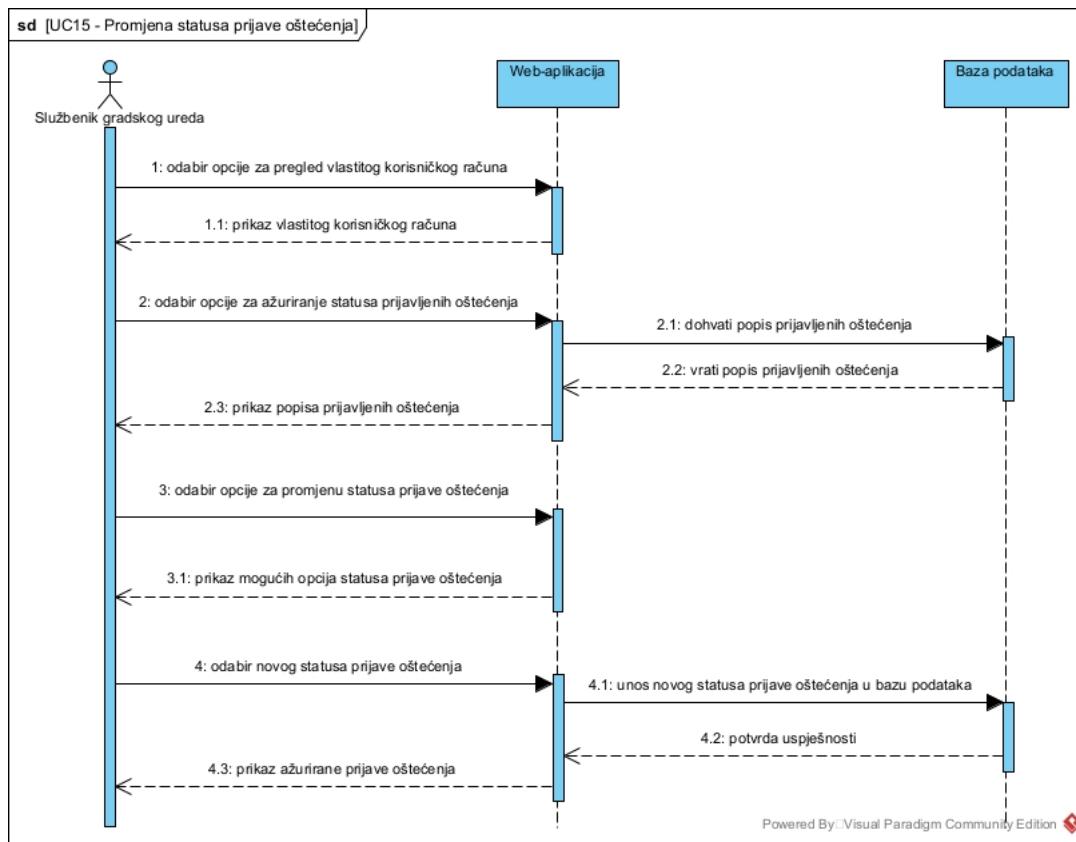
Neregistrirani korisnik šalje zahtjev za prikaz opcije za provjeru statusa anonimne prijave oštećenja. Poslužitelj prikazuje stranicu za pregled statusa anonimne prijave oštećenja. Korisnik unosi jedinstveni broj prijave. Poslužitelj provjerava postoji li prijava oštećenja s tim jedinstvenim brojem u bazi podataka. Ukoliko postoji, poslužitelj prikazuje podatke o oštećenju, a ukoliko ne postoji, poslužitelj prikazuje poruku o neuspješnom unosu jedinstvenog broja prijave.



Slika 3.5: Sekvencijski dijagram za UC12

Obrazac uporabe UC15 - Promjena statusa prijave oštećenja

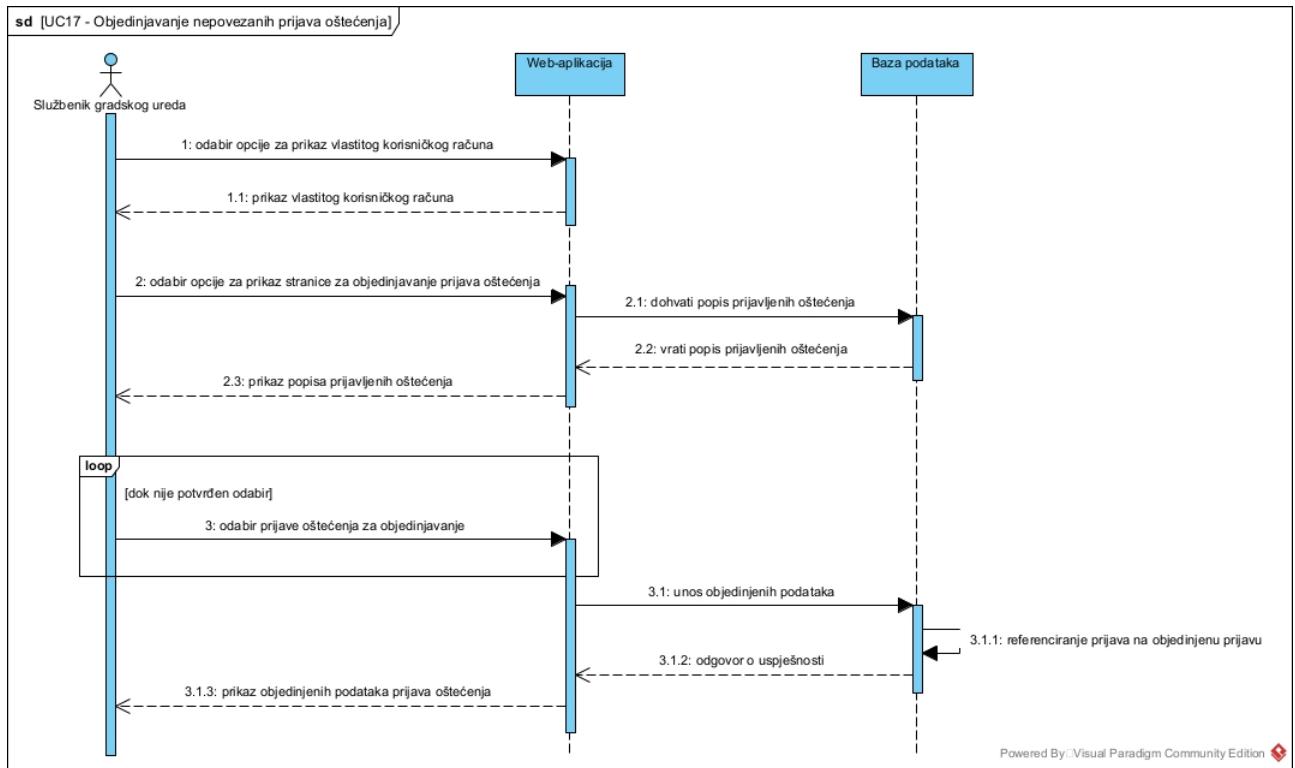
Službenik gradskog ureda šalje zahtjev za prikaz popisa prijavljenih oštećenja. Poslužitelj dohvaća popis prijavljenih oštećenja iz baze podataka te ih prikazuje službeniku. Službenik odabire prijavu oštećenja te time šalje zahtjev za prikaz podataka o toj prijavi oštećenja. Poslužitelj dohvaća podatke o prijavi oštećenja te ih prikazuje službeniku. Službenik šalje zahtjev za prikaz opcije za promjenu statusa prijave oštećenja. Poslužitelj prikazuje opciju za promjenu statusa prijave oštećenja s navedenim mogućim opcijama statusa prijave oštećenja. Službenik odabire novi status prijave oštećenja te šalje zahtjev za promjenu statusa poslužitelju. Poslužitelj ažurira status prijave oštećenja u bazi podataka. Poslužitelj prikazuje službeniku potvrdu o uspješnosti promjena statusa prijave oštećenja.



Slika 3.6: Sekvencijski dijagram za UC15

Obrazac uporabe UC17 - Objedinjavanje nepovezanih prijava oštećenja

Službenik gradskog ureda šalje zahtjev za prikaz popisa prijavljenih oštećenja. Poslužitelj dohvaća popis prijavljenih oštećenja iz baze podataka te ih prikazuje službeniku. Službenik odabire opciju za objedinjavanje prijava oštećenja. Poslužitelj prikazuje oznake za označavanje prijava oštećenja. Službenik odabire prijave oštećenja. Službenik potvrđuje odabir prijava oštećenja za objedinjavanje. Poslužitelj unosi podatke o objedinjenim prijavama oštećenja u bazu podataka. Događa se referenciranje objedinjene prijave oštećenja na postojećih prijava oštećenja. Po završetku, poslužitelj prikazuje službeniku gradskog ureda podatke objedinjene prijave oštećenja.



Slika 3.7: Sekvencijski dijagram za UC17

3.2 Ostali zahtjevi

1. Sustav treba podržavati rad više korisnika u stvarnom vremenu
2. Sustav treba biti izведен kao web aplikacija
3. Sustav treba ispravno funkcionirati na svim web preglednicima
4. Korisničko sučelje i sustav podržavaju hrvatsku abecedu (dijakritičke znakove) pri unosu i prikazu tekstualnog sadržaja
5. Sustav koristi hrvatski jezik i srednjoeuropsko standardno vrijeme, GMT+1
6. Korisničko sučelje treba biti jednostavno za korištenje bez opširnih uputa
7. Neispravno korištenje korisničkog sučelja ne smije narušiti funkcionalnost i rad sustava
8. Sustav ne smije omogućiti registraciju korisnika i promjenu lozinke dok nije unesena dovoljno jaka lozinka duljine od barem 8 znakova te barem jedno malo slovo, veliko slovo, znamenku i specijalni znak
9. Sustav sprema lozinku u sigurnom obliku različitom od običnog tekstualnog formata, koristeći bcrypt hash
10. Sustavu se pristupa s javne mreže pomoću HTTPS
11. Pristupanje bazi podataka ne smije trajati duže od nekoliko sekundi
12. Buduće nadogradnje sustava ne smiju narušiti postojeće funkcionalnosti sustava

4. Arhitektura i dizajn sustava

Pri analizi projektnih zahtjeva i detaljnog razmatranju uloga dionika te njihovih interakcija unutar aplikacije, odlučili smo strukturirati naš sustav na tri ključne razine: razinu klijenta, razinu web aplikacije i razinu baze podataka. Unutar ove podjеле, nužno je uključiti slojeve korisničkog sučelja, aplikacijske logike i pristupa podacima. Kao model arhitekture, odabrali smo višeslojnu strukturu sličnu MVC (Model-View-Controller) stilu. MVC je oblik arhitekture softvera koji organizira aplikaciju u tri komponente:

- *Model (poslovna logika i podaci)*
- *View (korisničko sučelje)*
- *Controller (upravljač, posrednik između Modela i Viewa)*

Razdvajanje logike, prezentacije i upravljanja omogućuje jednostavno održavanje i razvoj aplikacije, a promjene u jednoj komponenti ne bi trebale značajno utjecati na druge. Ova arhitektura, bazirana na klijent-poslužitelj odnosu, omogućuje jasno definiranu organizaciju slojeva, a s ciljem maksimalne ponovne uporabivosti, integrirali smo i različite programske biblioteke i radne okvire. Razvojni tim je pisao kod u razvojnom okruženju IntelliJ IDEA i Visual Studio Code, a za pokretanje, konfiguraciju i puštanje u pogon, kao i neovisnost o računalu na kojem se kod izvršava, odabrali smo platforme Docker, Render i Node.js.

Arhitektura web aplikacije "CestaFix" može se podijeliti na tri podsustava:

- *Web poslužitelj*
- *Web aplikacija*
- *Baza podataka*

Web poslužitelj je komponenta koja pruža podršku za backend sustav aplikacije i čija je ključna uloga omogućiti komunikaciju između klijenta i aplikacije. Ova

interakcija odvija se putem HTTP (Hyper Text Transfer Protocol) protokola, standardnog protokola za prijenos informacija na webu. Pokretanje web aplikacije i prosljeđivanje zahtjeva aplikaciji radi daljne obrade, inicira se upravo pomoću web poslužitelja. Za implementaciju web poslužitelja korišten je Spring Boot, Java framework, koji se ističe brzim konfiguriranjem i implementacijom web aplikacija temeljenih na Javi. Sastavljen od Model i Controller slojeva prema MVC arhitekturi, gdje se poslovna logika, kao što je upravljanje prijavama šteta, nalazi u Modelu. Web poslužitelj također omogućuje definiranje i implementaciju RESTful API-ja, što omogućuje komunikaciju između frontenda (klijenta) i backenda (poslužitelja) putem HTTP/HTTPS mrežnih protokola. Primarni zadatak Controllera unutar web poslužitelja je obrađivanje HTTP zahtjeva koji pristižu iz frontend dijela aplikacije, izvršavanje odgovarajuće funkcionalnosti te slanje odgovora. Dodatno, Spring Boot omogućuje učinkovito upravljanje stanjem aplikacije, uključujući praćenje stanja sesija za registrirane korisnike.

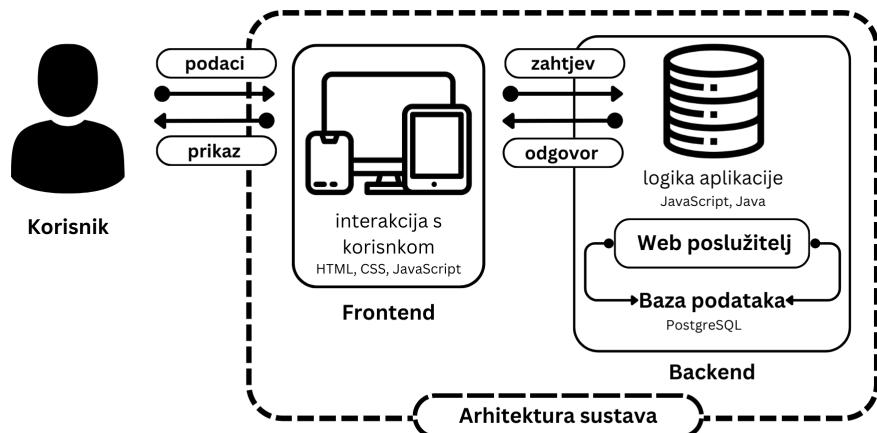
Web preglednik je softver koji djeluje kao posrednik između poslužitelja i klijenta, omogućujući korisniku prikaz web sadržaja. Osnovna funkcionalnost web preglednika ostvarena je pri slanju HTTP zahtjeva poslužitelju te prijemu i interpretaciji HTTP odgovora. Web preglednici djeluju kao prevoditelji, omogućavajući korisnicima vizualizaciju web sadržaja kroz sučelje preglednika, dekodiranjem informacija dobivenih iz HTTP odgovora.

Web aplikacija Web aplikacija je kompleksni softverski sustav koji se sastoji od frontend i backend dijelova:

- **Frontend (Klijent)** predstavlja korisničko sučelje putem kojeg korisnici ostvaruju interakciju sa sustavom. Ostvaren je korištenjem JavaScript programskog jezika, za potrebu upravljanja događajima na korisničkom sučelju, i React radnog okvira, što omogućuje kreiranje dinamičnog i intuitivnog korisničkog sučelja, čime se ostvaruje ugodno korisničko iskustvo. Frontend je strukturiran u komponente, što olakšava održavanje i ponovnu uporabu koda. Komunicira s backendom kroz HTTPS zahtjeve te pritom omogućuje prijenos podataka i ažuriranje informacija o prijavama šteta. Kroz klijentsku logiku, frontend provodi validaciju unesenih podataka kako bi osigurao ispravnost prije slanja na backend. Responsivni dizajn osigurava konzistentno korisničko iskustvo na različitim uređajima.

- **Backend** (Poslužitelj) je dio sustava unutar kojeg se obrađuju zahtjevi i izvršavaju daljnje radnje. Kako bi se postiglo "razdvajanje zabrinutosti", organiziran je na kontrolere, servise i rezervorije. Controlleri imaju ključnu ulogu u obradi ulaznih zahtjeva (HTTP zahtjeva) omogućujući organizaciju i upravljanje tijekom rukovanja zahtjevima u backend dijelu aplikacije kao i pozivanje odgovarajućih metoda u Modelu te slanje odgovora klijentskom dijelu. Servisi u backendu su učinkoviti i organizirani način obavljanja poslovne logike i specifične funkcionalnosti koje su potrebne za obradu zahtjeva, koji dolaze s frontend dijela aplikacije. Rezervorije imaju ključnu ulogu u komunikaciji s bazom podataka, odnosno omogućuju servisima da abstrahiraju detalje interakcije s bazom podataka, pružajući im jednostavan i konzistentan način komunikacije s podacima. Backend je ostvaren korištenjem Java programskog jezika i Spring Boot radnog okvira. Također, pruža RESTful API-je koji omogućuju komunikaciju između klijenta i servera te definiraju kako resursi (poput prijava šteta) mogu biti stvoreni, ažurirani i dohvaćeni. Osim toga backend sadrži i DTO-e (Data Transfer Objects) za prijenos podataka između različitih dijelova sustava odnosno slojeva aplikacije.

Baza podataka je podatkovni sloj koji se koristi za sigurnu pohranu podataka te je detaljnije opisana u sljedećem poglavlju.



Slika 4.1: Arhitektura sustava

4.1 Baza podataka

Sustav je temeljen na uporabi relacijske baze podataka implementirane u PostgreSQL-u gdje su entiteti modelirani kao tablice koje posjeduju svoje jedinstveno ime i skup atributa. Odabir relacijske baze podataka proizlazi iz potrebe za lakšim ostvarenjem naših potreba za upravljanjem podacima pri prijavljivanju oštećenja i njihovoj sanaciji odnosno kako bismo što jednostavnije modelirali sustav prema stvarnom svijetu. Ova baza podataka ključna je za sigurnost podataka i brz pristup, pohranu, umetanje, izmjenu te dohvati podataka koje sustav koristi za daljnju obradu. Baza podataka ove aplikacija sadrži sljedeće entitete:

- *Users*
- *Reports*
- *CityDept*
- *Category*
- *Problems*
- *Photo*

4.1.1 Opis tablica

Users je entitet koji sadrži sve bitne informacije o korisnicima i njihovim ulogama unutar aplikacije. Sastoji se od atributa: user_id, firstname, lastname, email, password, role i city_dept_id. Povezan je vezom *Many-to-One* s entitetom CityDept preko atributa city_dept_id i vezom *One-to-Many* s entitetom Reports preko atributa user_id.

Users		
user_id	INT	jedinstveni identifikator korisnika
firstname	VARCHAR	ime korisnika
lastname	VARCHAR	prezime korisnika
email	VARCHAR	e-mail adresa korisnika
password	VARCHAR	hash lozinke korisnika
role	VARCHAR	uloga korisnika

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

Users		
city_dept_id	INT	jedinstveni identifikator gradskog ureda (citydept.city_dept_id)

Reports je entitet koji sadrži sve bitne informacije o prijavama oštećenja. Sastoje se od atributa: report_id, user_id, title, description, address, report_time, status, problem_id, longitude, latitude i business_id. Povezan je vezom *Many-to-One* s entitetom Problems preko atributa problem_id, vezom *Many-to-One* s entitetom Users preko atributa user_id i vezom *One-to-Many* s entitetom Photo preko atributa report_id.

Reports		
report_id	INT	jedinstveni identifikator prijave
user_id	INT	jedinstveni identifikator korisnika (users.user_id)
title	VARCHAR	naziv prijave/oštećenja
description	TEXT	opis oštećenja
address	VARCHAR	adresa oštećenja
report_time	TIMESTAMP	vrijeme podnošenja prijave
status	VARCHAR	status prijave
longitude	DOUBLE	geografska dužina lokacije oštećenja
latitude	DOUBLE	geografska širina lokacije oštećenja
business_id	UUID	jedinstveni identifikator anonimne prijave
problem_id	INT	jedinstveni identifikator oštećenja (problems.problem_id)

CityDept je entitet koji sadrži sve bitne informacije o gradskim uredima. Sastoje se od atributa: city_dept_id, city_dept_name i category_id. Povezan je vezom *One-to-Many* s entitetom Users preko atributa city_dept_id i vezom *One-to-One* s entitetom Category preko atributa category_id.

CityDept		
city_dept_id	INT	jedinstveni identifikator gradskog ureda
city_dept_name	VARCHAR	naziv gradskog ureda
category_id	INT	jedinstveni identifikator kategorije oštećenja (category.category_id)

Category je entitet koji sadrži sve bitne informacije o kategoriji oštećenja. Sastoji se od atributa: category_id i category_name. Povezan je vezom *One-to-Many* s entitetom Problems preko atributa category_id i vezom *One-to-One* s entitetom CityDept preko atributa category_id.

Category		
category_id	INT	jedinstveni identifikator kategorije oštećenja
category_name	VARCHAR	naziv kategorije oštećenja

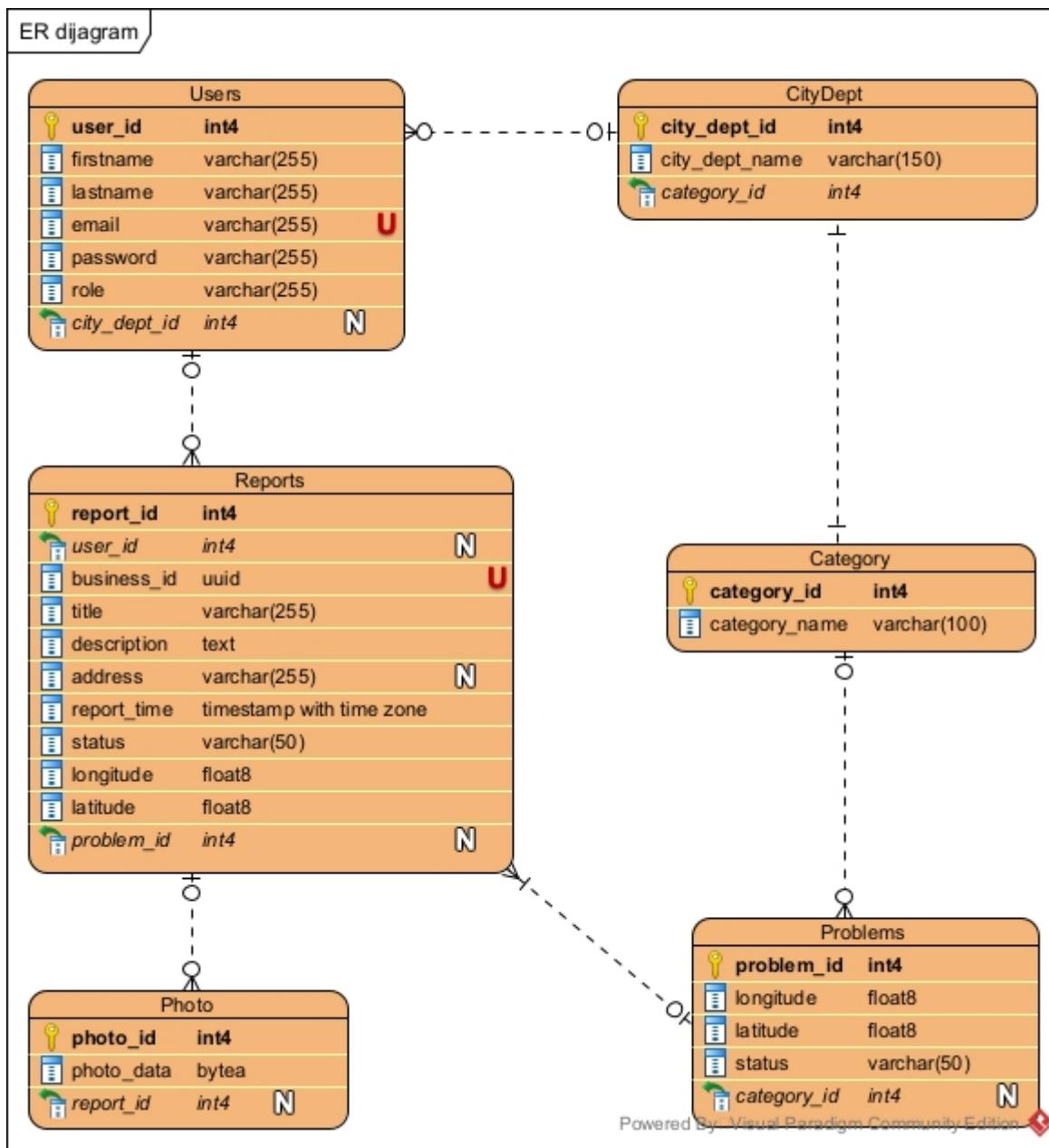
Problems je entitet koji sadrži sve bitne informacije o prijavljenom oštećenju. Sastoji se od atributa: problem_id, longitude, latitude, status i category_id. Povezan je vezom *One-to-Many* s entitetom Reports preko atributa problem_id i vezom *Many-to-One* s entitetom Category preko atributa category_id.

Problems		
problem_id	INT	jedinstveni identifikator oštećenja
longitude	DOUBLE	geografska dužina lokacije oštećenja
latitude	DOUBLE	geografska širina lokacije oštećenja
status	VARCHAR	status oštećenja
category_id	INT	jedinstveni identifikator kategorije oštećenja (category.category_id)

Photo je entitet koji sadrži sve bitne informacije vezane za sliku oštećenja. Sastoji se od atributa: photo_id, photo_data i report_id. Povezan je vezom *Many-to-One* s entitetom Reports preko atributa report_id.

Photo		
photo_id	INT	jedinstveni identifikator slike (photo.photo_id)
photo_data	BYTEA	slika oštećenja
report_id	INT	jedinstveni identifikator prijave (report.report_id)

4.1.2 Dijagram baze podataka

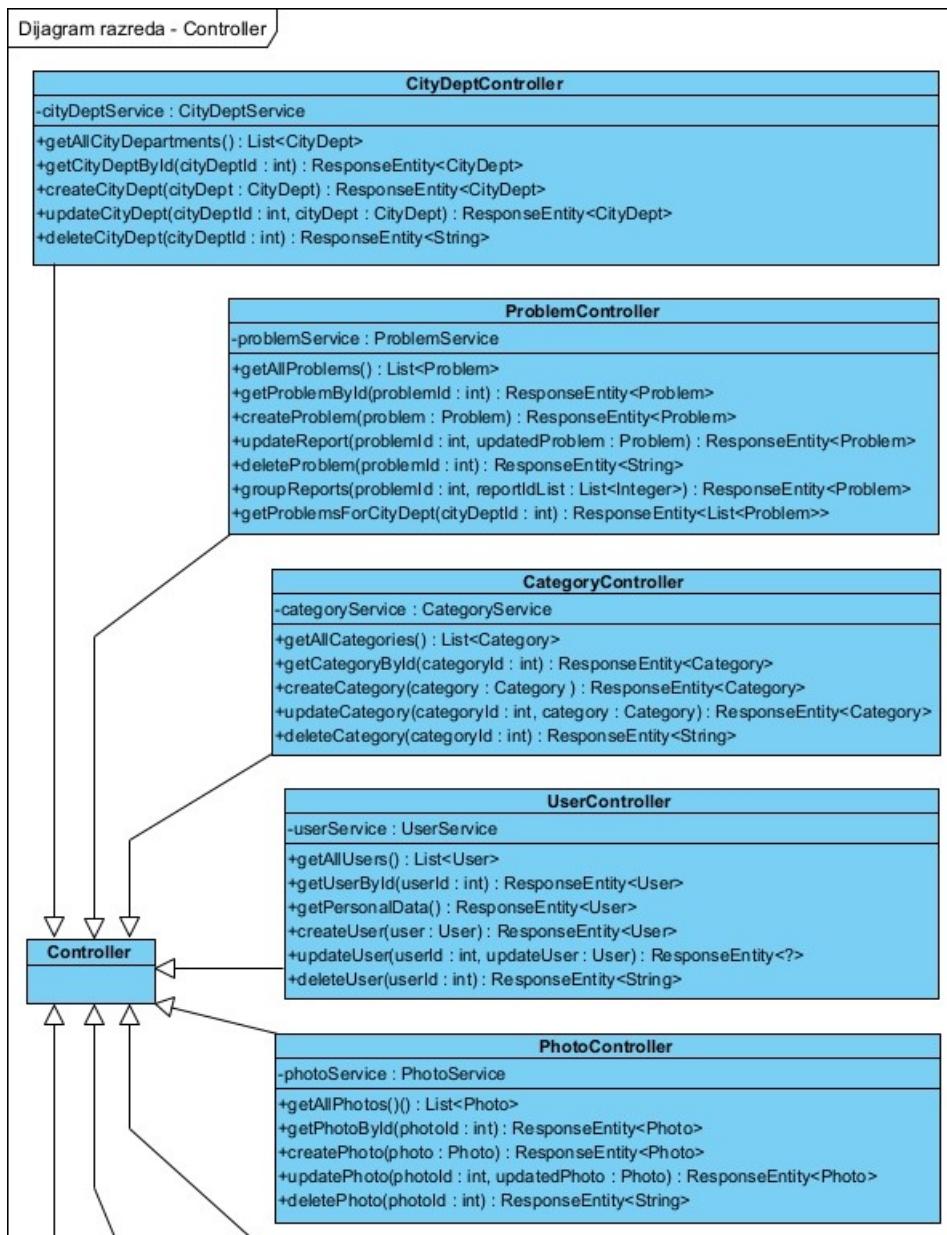


Slika 4.2: E-R dijagram baze podataka

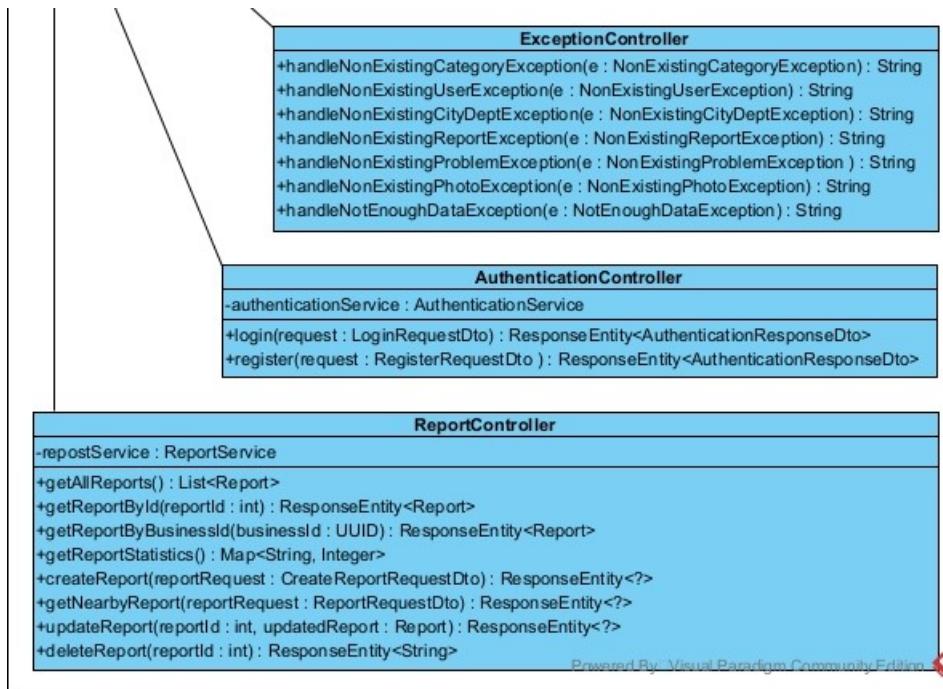
4.2 Dijagram razreda

Na slikama 4.3, 4.4, 4.5 i 4.6 prikazani su razredi koji pripadaju *backend* dijelu s arhitekturom podijeljenom na kontrolere, repozitorije i servise te uključuje DTO (Data Transfer Object) i modele.

Na slikama 4.3 i 4.4 prikazani su razredi koji nasljeđuju Controller razred. Metode implementirane u tim razredima upravljaju i rukuju DTO-ima koji su dohvaćeni pomoću metoda implementiranih u Model razredima. Omogućuju logiku interakcije i promjene. Kontroler AuthenticationController omogućuje registraciju i prijavu korisnika. Kontroler ReportController omogućuje interakcije s prijavom oštećenja, dok kontroler ProblemController omogućuje interakcije s objedinjenim problemima oštećenja s istom temom i lokacijom. Kontroler UserController omogućuje interakcije i promjene povezane s registriranim korisnicima. Kontroler CityDeptController omogućuje interakcije i promjene povezane s gradskim uređima. Kontroler PhotoController omogućuje interakciju i promjene vezane za slike oštećenja. Kontroler CategoryController omogućuje interakciju i promjene vezane za kategorije oštećenja. Kontroler ExceptionController omogućuje interakcije vezane za iznimke.

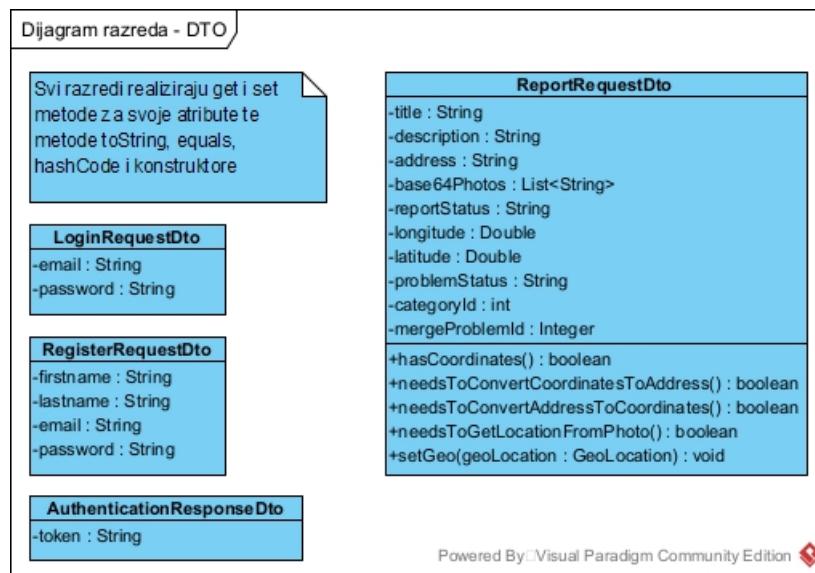


Slika 4.3: Dijagram razreda - dio Controllers - prvi dio



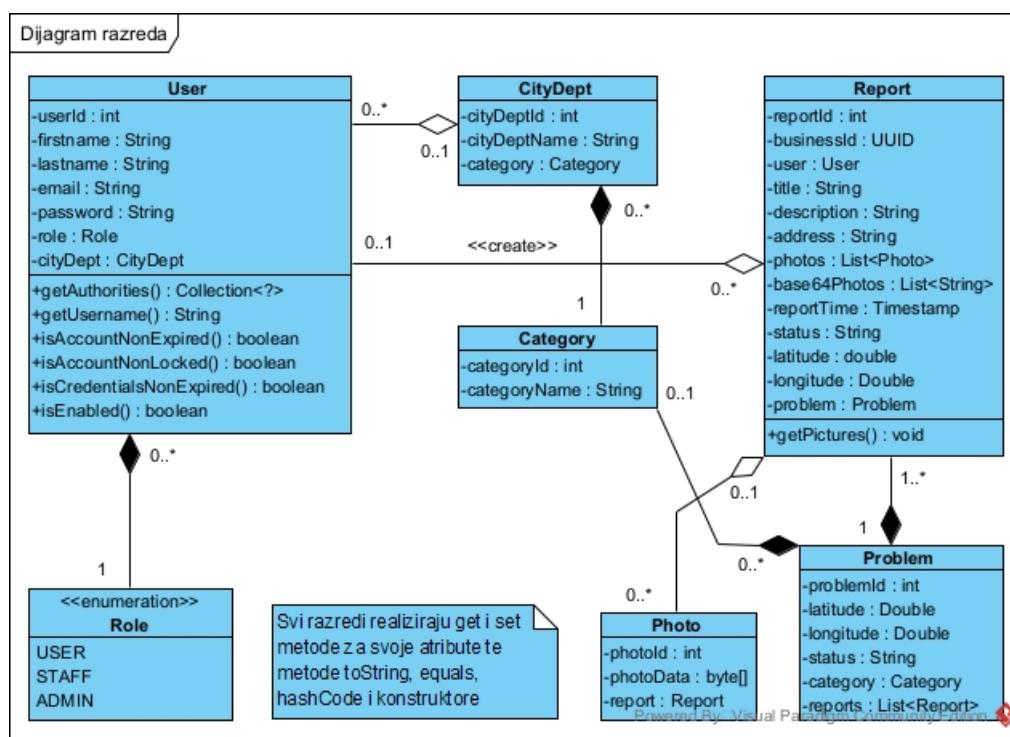
Slika 4.4: Dijagram razreda - dio Controllers - drugi dio

Na slici 4.4 prikazani su razredi koji pripadaju DTO. Data Transfer Objects (DTO) omogućavaju razmjenu podataka između procesa i slojeva. RegisterRequestDto služi za registraciju korisnika. LoginRequestDto služi za prijavu korisnika. AuthenticationResponseDto sastoji se od jwt tokena koji se vraća kada se korisnik registrira ili ulogira. ReportRequestDto služi za kreiranje podataka o prijavi oštećenja i problema.



Slika 4.5: Dijagram razreda - dio DTO

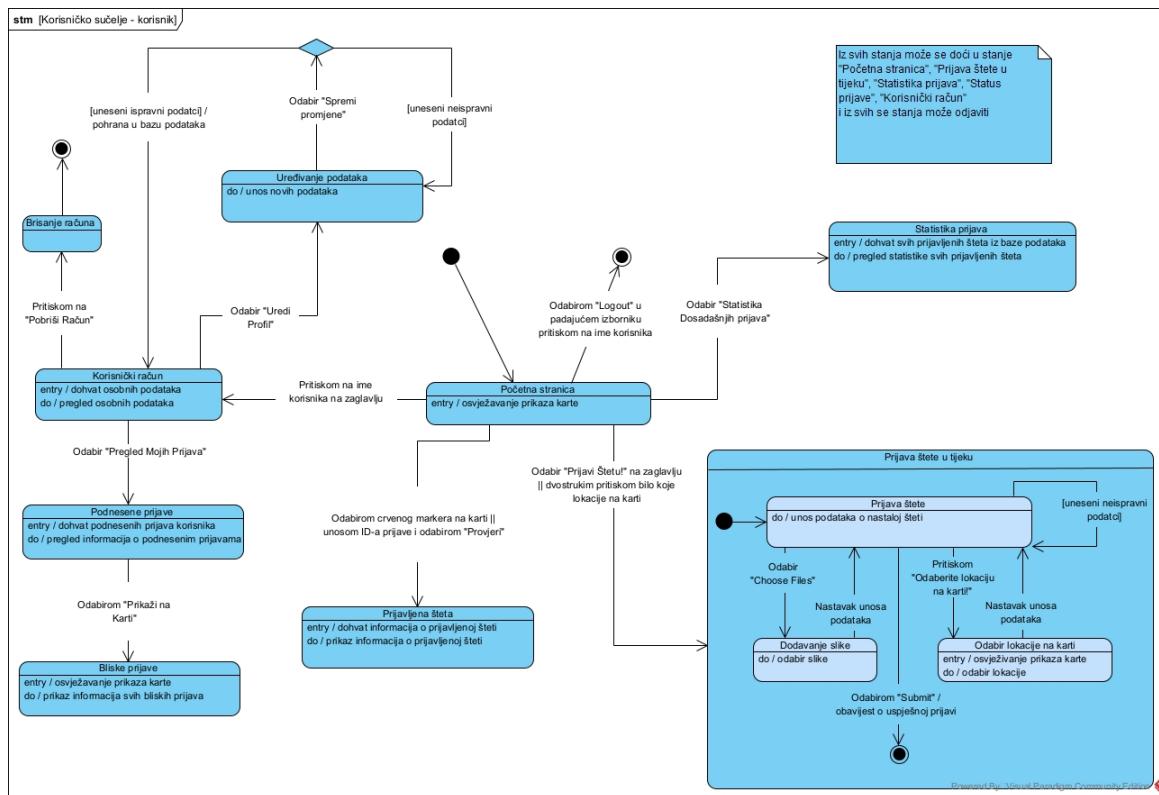
Na slici 4.5 prikazi su razredi koji pripadaju Model razredima. Oni preslikavaju strukturu baze podataka u aplikaciji. Komuniciraju s bazom podataka te vraćaju podatke iz nje. Razred User predstavlja registriranog korisnika koji je unio nužne podatke u sustav. Razred Report predstavlja prijavu oštećenja za koju je korisnik unio potrebne i opcionale podatke. Razred Problem predstavlja prijavu problema koja predstavlja jednu ili više prijava oštećenja objedinjene u prijavu problema sa zajedničkom temom i lokacijom. Razred Category predstavlja kategoriju oštećenja koju korisnik može odabrati za prijavljeno oštećenje, odnosno koju gradski ured može dobiti na upravljanje oštećenjima. Razred CityDept predstavlja gradski ured koji upravlja jednom kategorijom oštećenja koju im je administrator dodijelio. Enumeracija Role predstavlja ulogu koju registrirani korisnik ima. Razred Photo predstavlja fotografiju oštećenja koju je korisnik dodao prilikom prijave oštećenja.



Slika 4.6: Dijagram razreda - dio Model

4.3 Dijagram stanja

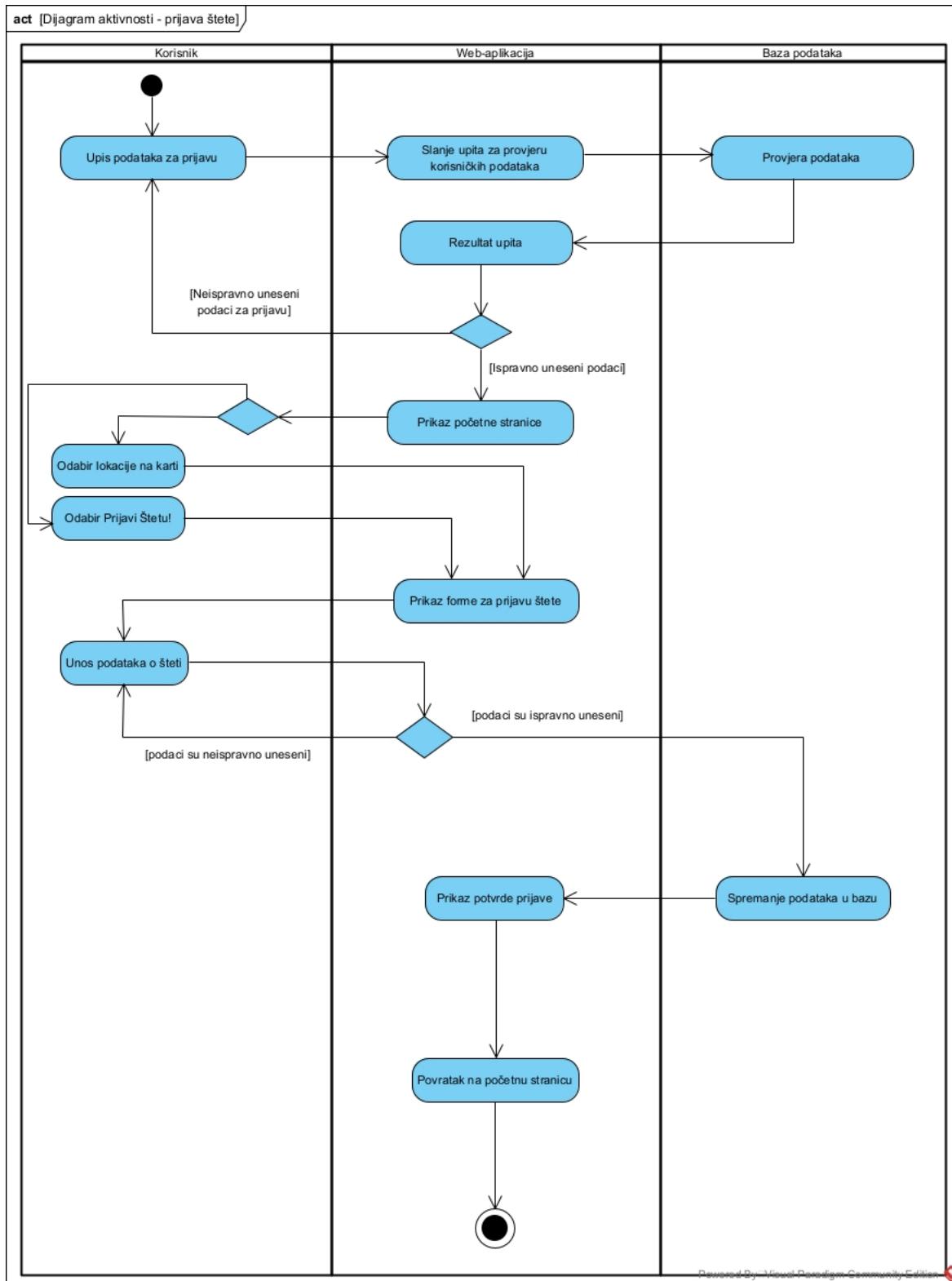
Dijagram stanja, poznat i kao state machine diagram, je ponašajni UML-dijagram koji se koristi za jasno definiranje mogućih stanja objekta, identifikaciju prijelaza između tih stanja i prikaz ponašanja sustava u različitim situacijama. Na slici 4.7 prikazan je dijagram stanja za prijavljenog korisnika. Nakon prijave, korisniku se prikazuje početna stranica s kartografskim prikazom na kojem su crvenim markerima označene već podnesene prijave određenih šteta na javnim površinama. Odabirom nekog crvenog markera, korisniku se prikazuju informacije o konkretnoj podnesenoj prijavi. Sličnu funkcionalnost korisnik ostvaruje upisom identifikacijskog broja prijave u zaglavlju čime mu se prikazuju informacije, odnosno status neke od vlastitih podnesenih prijava. Osim pregleda statusa podnesenih šteta, pomoću zaglavlja aplikacije korisnik može pristupiti svom korisničkom računu, pregledati statistiku svih do tog trenutka podnesenih prijava te pristupiti podnošenju nove prijave. Prilikom podnošenja nove prijave, osim unošenja podataka o nastaloj šteti, korisnik može odabrati sliku s vlastitog uređaja i odabrati lokaciju štete na karti. Također, pomoću padajućeg izbornika u zaglavlju, korisnik se može odjaviti iz sustava. S obzirom na to da je zaglavlje zastupljeno na svim stranicama aplikacije, korisnik svim navedenim opcijama vezanim uz zaglavlje može pristupiti neovisno o tome gdje se nalazi unutar aplikacije. Pristupom u vlastiti korisnički račun, omogućen je pregled i uređivanje osobnih podataka, pregled podnesenih prijava korisnika kao i brisanje računa. Ako korisnik prilikom pregleda informacija vlastitih podnesenih prijava, odabere jednu od njih, otvara se stranica s kartografskim prikazom koja, osim informacija o odabranoj prijavi, sadrži i prikaz svih informacija o prijavama koje su podnesene u bliskom vremenskom periodu za istu lokaciju.



Slika 4.7: Dijagram stanja

4.4 Dijagram aktivnosti

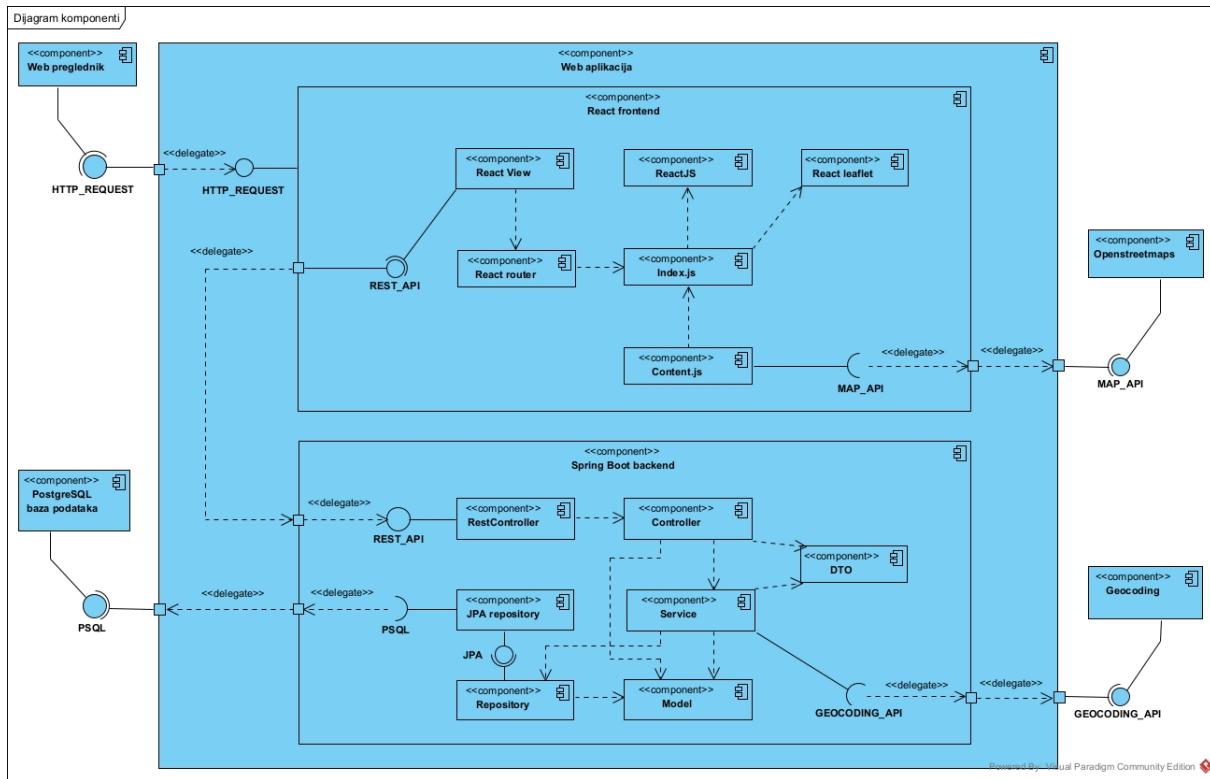
UML-dijagrami aktivnosti (activity diagrams) predstavljaju ponašajne dijagrame koji se koriste za modeliranje i grafički prikaz dinamičkog ponašanja sustava. Ovi dijagrami vizualiziraju izvođenje aktivnosti kroz niz akcija koje kontroliraju tokove i objekte, s posebnim naglaskom na slijed i uvjete toka. Dijagrami aktivnosti omogućuju jasno predstavljanje redoslijeda izvođenja aktivnosti, čime olakšavaju razumijevanje dinamike sustava. Na dijagramu aktivnosti sa slike 4.8 prikazan je proces podnošenja nove prijave štete neprijavljenog korisnika. Korisnik se prijavlji u sustav, zatim odabirom lokacije na karti ili odabirom "Prijavi Štetu!" na zagлавlju, unosi podatke o nastaloj šteti u prikazanoj formi za prijavu. Ako su podaci neispravno uneseni, korisniku se omogućuje ponovni upis podataka, a ako su podaci ispravni, spremaju se u bazu podataka te se korisniku prikazuje potvrda o uspješno podnesenoj prijavi zajedno s identifikacijskim brojem prijave, kako bi mogao provjeriti status podnesene prijave. Korisnik je pri završetku procesa pozicioniran ponovno na početnu stranicu.



Slika 4.8: Dijagram aktivnosti

4.5 Dijagram komponenti

Dijagram komponenti prikazan je na slici 4.9 te služi za vizualizaciju organizacije i međuvisnosti implementacijskih komponenata (interna struktura) te odnos programske potpore prema okolini. Sustavu se pristupa preko web preglednika pomoću sučelja HTTP_REQUEST na kojem se poslužuju datoteke koje pripadaju *frontend* dijelu aplikacije. *Frontend* dio nazvan React Frontend sastoji se od komponenti React View, React router, ReactJS, Index.js, Content.js, React leaflet te sučelja REST_API i MAP_API. React View komponenta komunicira s *backendom* preko sučelja REST_API i razmjenjuje podatke s *backendom* u JSON formatu, te ovisno o korisnikovim akcijama osvježava prikaz na stranici. Komponenta React router na temelju URL-a odlučuje koja će se stranica prikazati, odnosno datoteka dohvati. Komponenta ReactJS je centralna biblioteka za React preko koje se dobivaju gotove komponente za prikaz. Komponenta Index.js služi kao početna komponenta unutar koje se nalazi organizirana hijerarhija ostalih elemenata za prikaz. Jedan od njih je komponenta Content.js koja služi za ostvarivanje prikaza karte. Integraciju interaktivnih karata i manipulaciju geografskih podataka s React aplikacijama omogućava biblioteka React leaflet prikazana komponentom React leaflet. Sučelje MAP_API koristi se za razmjenu podataka s vanjskom komponentom Openstreetmaps. *Backend* dio nazvan Spring Boot backend sastoji se od komponenti RestController, JPA repository, Controller, Service, Model, Repository, DTO te sučelja REST_API, PSQL, JPA i GEOCODING_API. Komponenta RestController komunicira s *frontendom* preko sučelja REST_API i razmjenjuje podatke u JSON formatu. Komponenta JPA repository predstavlja dio Spring Data JPA koji omogućava jednostavan pristup podacima u bazi podataka koristeći PSQL sučelje za komuniciranje i razmjenu podataka s vanjskom komponentnom PostgreSQL baza podataka te sučelja JPA za komunikaciju s komponentnom Repository. Komponenta Controller prima zahtjeve, provjerava ih te šalje odgovore na dobivene zahtjeve prema klijentskoj strani. Komponenta Service prima zahtjeve od Controller, obrađuje ih i prosljeđuje komponentni Repository. Komponenta Model predstavlja entitete koji se pohranjuju u bazi podataka. Komponenta DTO predstavlja oblik objekta koji se koristi za prijenos podataka između dijelova aplikacije. Sučelje GEOCODING_API komunicira s vanjskom komponentom Geocoding koja služi za pretvaranje adrese u geografske koordinate i obrnuto.



Slika 4.9: Dijagram komponenti

5. Implementacija i korisničko sučelje

5.1 Korištene tehnologije i alati

Komunikacija u timu realizirana je korištenjem aplikacije **Discord**. Discord je društvena platforma na kojoj korisnici imaju mogućnost komuniciranja tekstualnim porukama, glasovnim pozivima, videopozivima, medijima i datotekama u privatnim porukama ili kao dio zajednice koju nazivaju server. Za izradu UML dijagrama korišten je alat **Visual Paradigm**. Visual Paradigm je grafički alat koji omogućava jednostavno modeliranje mnogo različitih tipova UML dijagrama, poput dijagrama obrazaca uporabe, sekvencijskih dijagrama, dijagrama razreda, dijagrama stanja, dijagrama aktivnosti, dijagrama komponenata, ERD modela baze podataka i mnogih drugih. Za upravljanje izvornim kodom korišten je **Git**. Git je open-source distribuirani sustav za upravljanje različitim verzijama datoteka. Udaljeni repozitorij projekta je dostupan na web platformi **GitHub**. GitHub pruža usluge spremanja i upravljanja kodom. Koristi se Git-om kako bi omogućio upravljanje različitim verzijama datoteka. Također, GitHub omogućava dokumentiranje programske podrške pomoću wiki-ja.

Kao razvojno okruženje korišteni su **Visual Studio Code** i **IntelliJ IDEA**. Visual Studio Code je uređivač teksta razvijen u tvrtki Microsoft. Prvenstveno se koristi za razvoj računalnih sustava na operacijskom sustavu Windows. Korišten je za razvoj programske podrške na *frontendu* i razvoj dokumentacije. IntelliJ IDEA je integrirano razvojno okruženje (IDE) razvijeno u tvrtki JetBrains. Usmjereno je na razvoj Java aplikacija, no podržava niz drugih jezika i tehnologija. Korišten je za razvoj programske podrške na *backendu*.

Aplikacija je napisana koristeći radni okvir **Spring Boot** i jezik **Java** za izradu *backenda* te jezik **JavaScript** i njegovu biblioteku **React** za izradu *frontenda*. React je biblioteka u JavaScriptu za izgradnju korisničkih sučelja. Nastala je od strane Facebooka. Glavna karakteristika Reacta je komponentna arhitektura, što znači da se korisničko sučelje sastoji od više manjih, ponovno uporabljivih komponenata. Iz-

rada složenijih aplikacija u Reactu obično zahtijeva korištenje dodatnih biblioteka za interakciju s API-jem. Radni okvir Spring Boot nudi gotova rješenja i funkcionalnosti koje ubrzavaju razvoj aplikacija. Ima automatsko upravljanje konfiguracijom i zavisnostima što olakšava i ubrzava posao programerima. Spring Boot pruža podršku za implementaciju sigurnosti u aplikacijama pomoću Spring Security modula.

Baza podataka izvedena je u **PostgreSQL**-u. PostgreSQL je open-source sustav za upravljanje relacijskim bazama podataka kojim se proširuje funkcionalnost SQL-a. Dizajniran je da izdrži različita radna opterećenja, od pojedinačnih računala, pa sve do skladišta podataka ili web usluga s mnogo istodobnih korisnika. Baza podataka se na poslužitelju u oblaku **Render**. Kao okruženje za upravljanje bazom podataka korišten je open-source grafički alat **pgAdmin**.

Dokumentacija je pisana u jeziku **LaTeX**. LaTeX je jezik za pisanje strukturiranih tekstova profesionalne kvalitete. Za razliku od nekih programske jezike za obradu teksta s grafičkim sučeljem poput Microsoft Worda, dokumenti u LaTeX-u pisani su kao obični tekst s dodanom semantičkom strukturom. Time postiže usredotočenost na sadržaj, ujednačenost izgleda te brži i stabilniji rad.

Ispitivanje sustava napravljeno je koristeći **Selenium WebDriver**. Selenium WebDriver je web radni okvir koji nam omogućuje izvođenje testova u različitim preglednicima. Koristi se za automatizaciju testiranja web aplikacija kako bi se provjerilo da li se ponašaju očekivano. Koristili smo ga za pisanje JavaScript testova uz preglednik **Google Chrome**.

Docker je platforma koja pomaže programerima da grade, dijele i pokreću aplikacije u kontejnerima. Kontejneri su izolirane jedinice koje sadrže sve što je potrebno za pokretanje aplikacije. Omogućuje brže i sigurniji razvoj aplikacija koje se mogu pokretati na bilo kojem okruženju. Koristili smo ga za lokalno pokretanje *backenda* aplikacije i baze podataka tijekom izrade projekta.

5.2 Ispitivanje programskog rješenja

5.2.1 Integracijsko testiranje

Potrebno je provesti ispitivanje jedinica (engl. *unit testing*) nad razredima koji implementiraju temeljne funkcionalnosti. Razraditi **minimalno 6 ispitnih slučajeva** u kojima će se ispitati redovni slučajevi, rubni uvjeti te izazivanje pogreške (engl. *exception throwing*). Poželjno je stvoriti i ispitni slučaj koji koristi funkcionalnosti koje nisu implementirane. Potrebno je priložiti izvorni kôd svih ispitnih slučajeva te prikaz rezultata izvođenja ispita u razvojnem okruženju (prolaz/pad ispita).

Na backendu integracijsko testiranje se sastoji od pozivanja API-a i očekivanja određenog odgovora za taj API. Na taj način se testira stvarna procedura koja se odvija kada korisnik pozove taj API. Prolazi se kroz sve slojeve backenda - od kontrolera do baze. Postoji sveukupno 6 integracijskih testova - 5 njih za modele koji reprezentiraju entitete u bazi te jedan za registraciju i prijavu korisnika. Integracijski testovi funkcioniraju na način da se prvo u dockeru podigne baza podataka, što se može vidjeti na slici ???. Nakon toga izvršava se funkcija koja je anotirana sa @BeforeEach anotacijom. Funkcija koja ima anotaciju @BeforeEach će se izvršiti prije svakog testa. Također, postoji i funkcija koja je anotirana sa @AfterEach anotacijom. Ona će se izvršiti poslije svakog testa. Sami testovi su anotirani sa anotacijom @Test. Sve navedene anotacije pripadaju org.junit biblioteci.

```
@Container
public static PostgreSQLContainer<?> postgresQLContainer = new PostgreSQLContainer<?>( dockerImageName: "postgres:latest")
    .withDatabaseName("postgres")
    .withUsername("postgres")
    .withPassword("postgres");

no usages ▲ AntePrkacin
@DynamicPropertySource
static void postgresqlProperties(DynamicPropertyRegistry registry) throws SQLException {
    registry.add( name: "spring.datasource.url", postgresQLContainer::getJdbcUrl);
    registry.add( name: "spring.datasource.username", postgresQLContainer::getUsername);
    registry.add( name: "spring.datasource.password", postgresQLContainer::getPassword);
    registry.add( name: "default.enabled", () -> false);
}
```

Slika 5.1: Dizanje baze u integracijskom testu

Nakon što se baza digne, a prije pokretanja testova, izvršava se funkcija s anotacijom @BeforeEach koja u bazu umeće podatke potrebne za testiranje. Na primjer, ako se testira metoda za brisanje prijave, u bazu će se prije tog testa umetnuti jedna prijava. Ova funkcija se izvršava prije svakog testa. Umetanje u bazu funkcionira tako da se napravi String varijabla koja je jednaka SQL naredbi. Potom se izgradi

objekt koji se želi umetnuti u bazu i u SQL naredbu se upisuju atributi tog objekta. Na kraju se SQL naredba izvršava i podatak se upisuje u bazu (Slika ??).

```
@BeforeEach
void setUpCategory() {
    try (Connection connection = dataSource.getConnection()) {
        // Category
        String sqlCategory = "INSERT INTO Category (category_id, category_name) " +
            "VALUES (?, ?)";
        Category category = Category.builder()
            .categoryId(20)
            .categoryName("cat_20")
            .build();
        PreparedStatement preparedStatementCategory = connection.prepareStatement(sqlCategory);
        preparedStatementCategory.setInt( parameterIndex: 1, category.getCategoryId());
        preparedStatementCategory.setString( parameterIndex: 2, category.getCategoryName());
        preparedStatementCategory.executeUpdate();

        // City Dept
        String sqlCityDept = "INSERT INTO Citydept (city_dept_id, city_dept_name, category_id) " +
            "VALUES (?, ?, ?)";
        CityDept cityDept = CityDept.builder()
            .cityDeptId(20)
            .cityDeptName("dept_20")
            .category(category)
            .build();
        PreparedStatement preparedStatementCityDept = connection.prepareStatement(sqlCityDept);
        preparedStatementCityDept.setInt( parameterIndex: 1, cityDept.getCityDeptId());
        preparedStatementCityDept.setString( parameterIndex: 2, cityDept.getCityDeptName());
        preparedStatementCityDept.setInt( parameterIndex: 3, cityDept.getCategory().getCategoryId());
        preparedStatementCityDept.executeUpdate();

        // ...
    }
}
```

Slika 5.2: Umetanje podataka u bazu

Testovi su funkcije s anotacijom @Test. Za testiranje se koristi mockMvc.perform() funkcija koja simulira HTTP zahtjev, definira HTTP metodu za poziv API-a, predaje JSON ako treba, definira kakav contentType treba predati, je li potreban JWT za pristup API-ju i koji HTTP status se očekuje kao odgovor tog zahtjeva (slika ??).

```
@Test
public void getAllCategoriesAndExpect200OK() throws Exception {
    mockMvc.perform(get( urlTemplate: "/public/category/getAll"))
        .andExpect(status().isOk());
}
```

Slika 5.3: Integracijski test

Po završetku svakog testa izvršava se funkcija s anotacijom @AfterEach. Ona koristi JdbcTestUtils.deleteFromTables() funkciju kako bi izbrisala sve podatke iz određene tablice u bazi. Na ovakav način se osigurava da ako se nešto umetne u bazu prilikom testa, poslije završetka testa će se i izbrisati (slika ??).

```
@AfterEach  
void tearDownCategory() {  
    JdbcTestUtils.deleteFromTables(jdbcTemplate, ...tableNames: "Users");  
    JdbcTestUtils.deleteFromTables(jdbcTemplate, ...tableNames: "Citydept");  
    JdbcTestUtils.deleteFromTables(jdbcTemplate, ...tableNames: "Category");  
}
```

Slika 5.4: Brisanje podataka iz baze nakon testa

5.2.2 Ispitivanje sustava

Ispitivanje funkcionalnosti sustava je provedeno pomoću Selenium testova pisanih i jeziku JavaScript uz preglednik Google Chrome. Ukupno je napisano 8 testova od kojih 3 provjerava kako se sustav nosi s pogrešnim unosima.

1. Ispitni slučaj: Prijava korisnika

- **Ulaz:** Email adresa i lozinka korisnika
- **Očekivani izlaz:** Nakon prijave nalazit ćemo se na početnoj stranici s korisničkim imenom prikazanim u gornjem desnom kutu.
- **Koraci:**
 1. Klik na gumb Login/Register
 2. Unos korisničkih podataka, točnije emaila i lozinke
 3. Klik na gumb Prijava

```
const { By, Key, Builder, until } = require("selenium-webdriver");

async function prijava() {
  let driver = await new Builder().forBrowser("chrome").build();
  try {
    await driver.get("https://cestafix-fe.onrender.com/");
    const buttonElement = await driver.findElement(By.css('#login'));
    await buttonElement.click();
    const usernameField = await driver.findElement(By.css('#username'));
    await usernameField.sendKeys('filip.simunovic@gmail.com');
    const passwordField = await driver.findElement(By.css('#password'));
    await passwordField.sendKeys('Simiklimi.5');
    const submitButton = await driver.findElement(By.css('#submit'));
    await submitButton.click();
    await driver.sleep(10000);
  } catch (error) {
    console.error("Test failed:", error);
  } finally {
    // Quit the driver
    console.log("Test je prosao uspjesno.");
    await driver.quit();
  }
}
```

```
}
```

```
}
```

```
prijava();
```

2. Ispitni slučaj: Registracija korisnika

- **Ulaz:** Ime, prezime, email i lozinka korisnika
- **Očekivani izlaz:** Nakon uspješne registracije nalazit ćemo se na početnoj stranici s korisničkim imenom prikazanim u gornjem desnom kutu.
- **Koraci:**
 1. Klik na gumb Login/Register
 2. Klik na link "Nemaš račun? Registriraj se!"
 3. Unos korisničkih podataka
 4. Klik na gumb Registriraj se

```
const { By , Key , Builder , until } = require("selenium-webdriver");
```

```
async function register() {  
let driver = await new Builder().forBrowser("chrome").build();  
try {  
await driver.get("https://cestafix-fe.onrender.com/");  
const buttonElement = await driver.findElement(By.css('#login'));  
await buttonElement.click();  
const registerElement = await driver.findElement(By.css('#nemaracun'));  
await registerElement.click();  
const imeField = await driver.findElement(By.css('#imeid'));  
await imeField.sendKeys('Da');  
const prezimeField = await driver.findElement(By.css('#prezimeid'));  
await prezimeField.sendKeys('Vinko');  
const emailField = await driver.findElement(By.css('#emailid'));  
await emailField.sendKeys('da.vinko@gmail.com');  
const passwordField = await driver.findElement(By.css('#passwordid'));  
await passwordField.sendKeys('Vinko.66');  
const passwordrepeatField = await driver.findElement(By.css('#repatpasswordid'));
```

```
await passwordrepeatField.sendKeys('Vinki.66');
const submitField = await driver.findElement(By.css('#signupid'));
await submitField.click();
await driver.sleep(10000);
} catch (error) {
  console.error("Test failed:", error);
} finally {
// Quit the driver
console.log("Test je prosao uspjesno.");
await driver.quit();
}

}

register();
```

3. Ispitni slučaj: Neuspjela prijava korisnika

- **Ulaz:** Email adresa i *netočna* lozinka korisnika
- **Očekivani izlaz:** Nakon unosa pogrešne lozinke korisniku se trebaju ispisati da su uneseni krivi podatci u istom prozoru.
- **Koraci:**
 1. Klik na gumb Login/Register
 2. Unos korisničnih podataka, točnije emaila i lozinke
 3. Klik na gumb Prijava

```
const { By, Key, Builder, until } = require("selenium-webdriver");

async function prijava() {
let driver = await new Builder().forBrowser("chrome").build();
try {
await driver.get("https://cestafix-fe.onrender.com/");
const buttonElement = await driver.findElement(By.css('#login'));
await buttonElement.click();
const usernameField = await driver.findElement(By.css('#username'));
await usernameField.sendKeys('filip.simunovic@gmail.com');
```

```
const passwordField = await driver.findElement(By.css('#password'));
await passwordField.sendKeys('Simiklimiii.5');
const submitButton = await driver.findElement(By.css('#submit'));
await submitButton.click();
await driver.sleep(10000);
} catch (error) {
console.error("Test failed:", error);
} finally {
// Quit the driver
console.log("Test je prosao uspjesno.");
await driver.quit();
}
}
```

prijava();

4. Ispitni slučaj: Registracija korisnika i brisanje računa

- **Ulaz:** Ime, prezime, email i lozinka korisnika
- **Očekivani izlaz:** Nakon registriranja i brisanja računa, ako se pokušamo logirati s istim podatcima dobit ćemo neuspjelu prijavu.
- **Koraci:**
 1. Klik na gumb Login/Register
 2. Klik na link "Nemaš račun? Registriraj se!"
 3. Unos korisničnih podataka
 4. Klik na gumb Registriraj se
 5. Klik na gumb koji prikazuje ime korisnika
 6. Klik na gumb Pobriši račun!!!
 7. Klik na gumb POTVRDI
 8. Klik na gumb Login/Register
 9. Unos korisničnih podataka
 10. Klik na gumb Prijava

```
const { By, Key, Builder, until } = require("selenium-webdriver");

async function register() {
```

```
let driver = await new Builder().forBrowser("chrome").build();
try {
  await driver.get("https://cestafix-fe.onrender.com/");
  const buttonElement = await driver.findElement(By.css('#login'));
  await buttonElement.click();
  const registerElement = await driver.findElement(By.css('#nemaracun'));
  await registerElement.click();
  const imeField = await driver.findElement(By.css('#imeid'));
  await imeField.sendKeys('Da');
  const prezimeField = await driver.findElement(By.css('#prezimeid'));
  await prezimeField.sendKeys('Vinki');
  const emailField = await driver.findElement(By.css('#emailid'));
  await emailField.sendKeys('da.vinkiiiiiii@gmail.com');
  const passwordField = await driver.findElement(By.css('#passwordid'));
  await passwordField.sendKeys('Vinkii.66');
  const passwordrepeatField = await driver.findElement
    (By.css('#repatpasswordid'));
  await passwordrepeatField.sendKeys('Vinkii.66');
  const submitField = await driver.findElement(By.css('#signupid'));
  await submitField.click();
  await driver.sleep(5000);
  const accountField = await driver.findElement(By.css("#account"));
  await accountField.click();
  await driver.sleep(2000);
  const deleteField = await driver.findElement(By.css("#brisiid"));
  await deleteField.click();
  await driver.sleep(2000);
  const delete2Field = await driver.findElement(By.css("#brisiaccid"));
  await delete2Field.click();
  await driver.sleep(2000);
  const loginButtonElement = await driver.findElement(By.css('#login'));
  await loginButtonElement.click();
  const username2Field = await driver.findElement(By.css('#username'));
  await username2Field.sendKeys('da.vinkiiiiiii@gmail.com');
  const password2Field = await driver.findElement(By.css('#password'));
```

```
await password2Field.sendKeys('Vinkii.66');
const submit2Button = await driver.findElement(By.css('#submit'));
await submit2Button.click();
await driver.sleep(5000);
} catch (error) {
  console.error("Test failed:", error);
} finally {
  // Quit the driver
  console.log("Test je prosao uspjesno.");
  await driver.quit();
}
}
}

register();
```

5. Ispitni slučaj: Prijava štete anonimnog korisnika

- **Ulaz:** Naziv štete, opis štete, adresa štete
- **Očekivani izlaz:** Nakon uspješne prijave izbacit će se prozor da id-jem prijave pomoću kojeg možemo pratiti istu.
- **Koraci:**
 1. Klik na gumb Prijavi Štetu!
 2. Unos podataka prijave
 3. Klik na gumb Submit

```
const { By, Key, Builder, until } = require("selenium-webdriver");

async function prijava() {
let driver = await new Builder().forBrowser("chrome").build();
try {
await driver.get("https://cestafix-fe.onrender.com/");
const buttonElement = await driver.findElement(By.css('#prijava'));
await buttonElement.click();
const nameField = await driver.findElement(By.css('#name'));
await nameField.sendKeys('Palo stup na kolnik');
const descriptionField = await driver.findElement(By.css('#description'));
await descriptionField.sendKeys('Pao STOP znak tamo di hodaju ljudi');
```

```
const addressField = await driver.findElement(By.css('#address'));
await addressField.sendKeys('Unska 4');
const submitButton = await driver.findElement(By.css('.confirmButton'));
await submitButton.click();
await driver.sleep(40000);
const confirmButton = await driver.findElement(By.css('.loginbtn'));
await confirmButton.click();
} catch (error) {
  console.error("Test failed:", error);
} finally {
  // Quit the driver
  console.log("Test je prosao uspjesno.");
  await driver.quit();
}
}
}

prijava();
```

6. Ispitni slučaj: Prijava štete prijavljenog korisnika

- **Ulaz:** Email adresa i lozinka korisnika, naziv štete, opis štete, adresa štete
- **Očekivani izlaz:** Nakon uspješne prijave izbaciti će se prozor da id-jem prijave pomoću kojeg možemo pratiti istu.
- **Koraci:**
 1. Klik na gumb Login/Register
 2. Unos korisničnih podataka
 3. Klik na gumb Prijava
 4. Klik na gumb Prijavi Štetu!
 5. Unos podataka prijave
 6. Klik na gumb Submit

```
const { By, Key, Builder, until } = require("selenium-webdriver");

async function prijava() {
let driver = await new Builder().forBrowser("chrome").build();
try {
  await driver.get("https://cestafix-fe.onrender.com/");
}
```

```
const buttonElement = await driver.findElement(By.css('#login'));
await buttonElement.click();
const usernameField = await driver.findElement(By.css('#username'));
await usernameField.sendKeys('filip.simunovic@gmail.com');
const passwordField = await driver.findElement(By.css('#password'));
await passwordField.sendKeys('Simiklimi.5');
const submitButton = await driver.findElement(By.css('#submit'));
await submitButton.click();
await driver.sleep(5000);
// prijava
const button2Element = await driver.findElement(By.css('#prijava'));
await button2Element.click();
const nameField = await driver.findElement(By.css('#name'));
await nameField.sendKeys('Palo stup na kolnik opet');
const descriptionField = await driver.findElement(By.css('#description'));
await descriptionField.sendKeys('Pao je jedan
STOP znak tamo di hodaju ljudi');
const addressField = await driver.findElement(By.css('#address'));
await addressField.sendKeys('Unska 5');
const submit2Button = await driver.findElement(By.css('.confirmButton'));
await submit2Button.click();
await driver.sleep(40000);
} catch (error) {
console.error("Test failed:", error);
} finally {
// Quit the driver
console.log("Test je prosao uspjesno.");
await driver.quit();
}
}

prijava();
```

7. Ispitni slučaj: Neuspjela registracija korisnika

- **Ulaz:** Ime, prezime, email i neispravna lozinka (ne zadovoljava uvjet da mora biti dugačka 8 znakova, imati jedno veliko, jedno malo slovo, jedan broj i

jedan specijalan znak) korisnika

- **Očekivani izlaz:** Nakon neuspješne registracije korisniku će se ispitati kako lozinka ne zadovoljava sigurnosni uvjet.

- **Koraci:**

1. Klik na gumb Login/Register
2. Klik na link "Nemaš račun? Registriraj se!"
3. Unos korisničnih podataka
4. Klik na gumb Registriraj se

```
const { By, Key, Builder, until } = require("selenium-webdriver");

// Test sa neispravnom lozinkom
async function register() {
let driver = await new Builder().forBrowser("chrome").build();
try {
await driver.get("https://cestafix-fe.onrender.com/");
const buttonElement = await driver.findElement(By.css('#login'));
await buttonElement.click();
const registerElement = await driver.findElement(By.css('#nemaracun'));
await registerElement.click();
const imeField = await driver.findElement(By.css('#imeid'));
await imeField.sendKeys('Da');
const prezimeField = await driver.findElement(By.css('#prezimeid'));
await prezimeField.sendKeys('Vinki');
const emailField = await driver.findElement(By.css('#emailid'));
await emailField.sendKeys('da.vinki@gmail.com');
const passwordField = await driver.findElement(By.css('#passwordid'));
await passwordField.sendKeys('Vinki.5');
const passwordrepeatField = await driver.findElement
(By.css('#repatpasswordid'));
await passwordrepeatField.sendKeys('Vinki.5');
const submitField = await driver.findElement(By.css('#signupid'));
await submitField.click();
await driver.sleep(10000);
} catch (error) {
```

```
console.error("Test failed:", error);
} finally {
// Quit the driver
console.log("Test je prosao uspjesno.");
await driver.quit();
}
}
register();
```

8. Ispitni slučaj: Neuspjela prijava štete logiranog korisnika

- **Ulaz:** Podatci prijave *bez* podatka o lokaciji
- **Očekivani izlaz:** Nakon što nije unesena nikakva lokacija, korisniku će se u prozoru prikazati ispis "Došlo je do greške, provjerite unos adrese prijave!"
- **Koraci:**
 1. Klik na gumb Login/Register
 2. Unos korisničnih podataka
 3. Klik na gumb Prijava
 4. Klik na gumb Prijavи Štetу!
 5. Unos podataka prijave
 6. Klik na gumb Submit

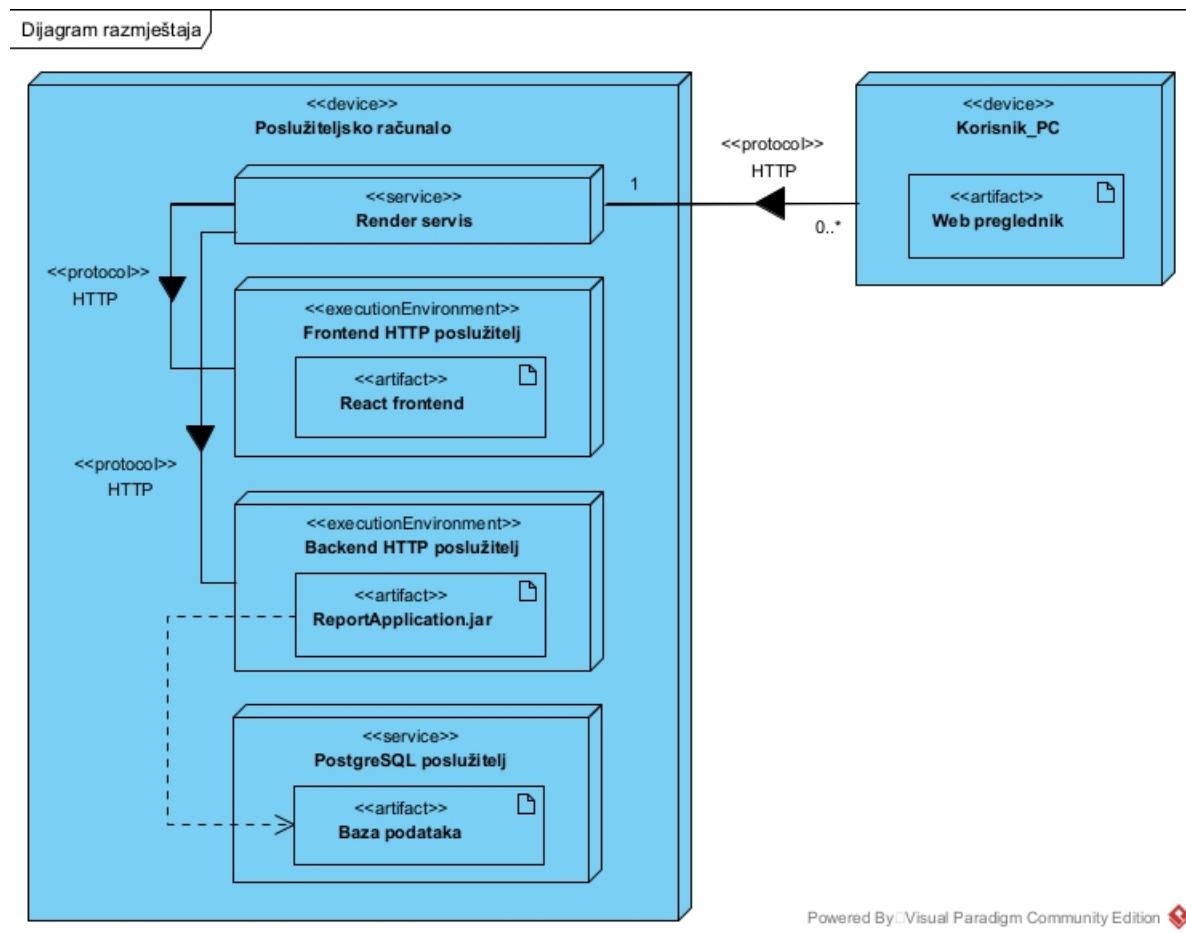
```
const { By, Key, Builder, until } = require("selenium-webdriver");

// report bez unosa lokacije - error
async function prijava() {
let driver = await new Builder().forBrowser("chrome").build();
try {
await driver.get("https://cestafix-fe.onrender.com/");
const buttonElement = await driver.findElement(By.css('#login'));
await buttonElement.click();
const usernameField = await driver.findElement(By.css('#username'));
await usernameField.sendKeys('filip.simunovic@gmail.com');
const passwordField = await driver.findElement(By.css('#password'));
await passwordField.sendKeys('Simiklimi.5');
const submitButton = await driver.findElement(By.css('#submit'));
}
```

```
await submitButton.click();
await driver.sleep(5000);
// prijava
const button2Element = await driver.findElement(By.css('#prijava'));
await button2Element.click();
const nameField = await driver.findElement(By.css('#name'));
await nameField.sendKeys('Palo stup na kolnik opet');
const descriptionField = await driver.findElement(By.css('#description'));
await descriptionField.sendKeys('Pao je jedan STOP znak tamo di hodaju ljudi');
const submit2Button = await driver.findElement(By.css('.confirmButton'));
await submit2Button.click();
await driver.sleep(40000);
} catch (error) {
console.error("Test failed:", error);
} finally {
// Quit the driver
console.log("Test je prosao uspjesno.");
await driver.quit();
}
}
}
prijava();
```

5.3 Dijagram razmještaja

Dijagram razmještaja opisuje topologiju sustava i usredotočen je na odnos sklopovskih i programskih dijelova. Prikazan je na slici 5.1. Na poslužiteljskom računalu se nalazi Render servis, Frontend HTTP poslužitelj, Backend HTTP poslužitelj te PostgreSQL poslužitelj na kojem je baza podataka. Na Frontend HTTP poslužitelju se nalazi React frontend CestaFix aplikacije. Na Backend HTTP poslužitelju se nalazi ReportApplication.jar, tj. Java Archive file u kojem se nalazi *backend* CestaFix aplikacije. Poslužiteljska strana aplikacije ostvarena je pomoću servisa Render. Klijenti koriste web preglednik kako bi pristupili web aplikaciji. Sustav je baziran na arhitekturi "klijent - poslužitelj", a komunikacija između računala korisnika i poslužitelja odvija se preko HTTP veze.



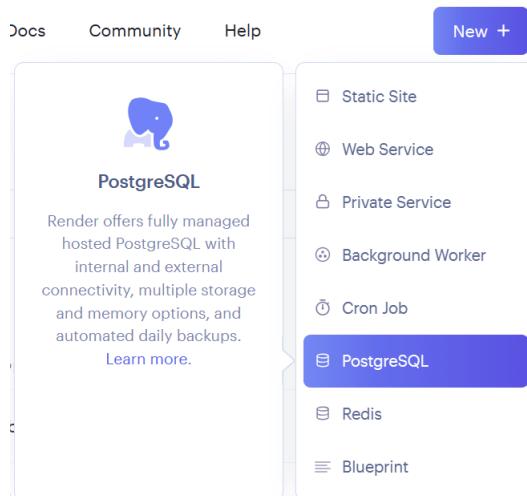
Slika 5.5: Dijagram razmještaja

5.4 Upute za puštanje u pogon

Za pogon naše aplikacije korištena je platforma Render koja pruža besplatno puštanje aplikacije u pogon sa, naravno, ograničenim mogućnostima.

5.4.1 Puštanje baze podataka u pogon

Prvo je potrebno upogoniti bazu podataka. Nakon prijavljivanja u platformu, potrebno je odabratи opciju New -> PostgreSQL (Slika ??). Zatim je na ekranu prikazan unos podataka za bazu podataka, za što smo mi ostavili sve prazno (automatski se generira) osim imena servisa, te je za regiju potrebno postaviti Frankfurt (EU Central) (Slika ??). Nedostatak besplatne verzije Rendera je to što je moguća samo jedna baza podataka koja je ograničena na trajanje od 3 mjeseca, no to je dovoljno za naše potrebe.



Slika 5.6: Kreiranje novog servisa

New PostgreSQL

[Read the docs](#)

Name	example-postgresql-name
Database	Optional The PostgreSQL `dbname`
User	Optional randomly generated unless specified
Region	Oregon (US West)
PostgreSQL Version	15
Datadog API Key	Optional The API key to use for sending metrics to Datadog. Setting this will enable Datadog monitoring.
Instance Type	For hobby projects Free \$0 / month 256 MB (RAM) 0.1 CPU 1 GB (Storage)

Slika 5.7: Unos podataka za bazu

5.4.2 Puštanje backenda u pogon

Isto tako je potrebno podesiti *backend*. Prvo je potrebno odabrati opciju Web Service u **??**. Nakon toga imamo opciju odabrati deploy preko github repozitorija, što omogućava automatsko redeployanje svaki put kada se napravi novi commit. Nakon odabira našeg repozitorija, opet imamo ekran za unos podataka **??**

Name	example-service-name
Region	Frankfurt (EU Central)
Branch	main
Root Directory	e.g., src
Runtime	Docker
Instance Type	For hobby projects Free \$0 / month 512 MB (RAM) 0.1 CPU

Slika 5.8: Unos podataka za backend

Za ime upisati željeno ime servisa, za regiju staviti Frankfurt, odabrati granu i pozicionirati se u *backend* direktorij. Za runtime staviti Docker. Nakon toga je potrebno upisat podatke za variabile okoline koje nam omogućavaju povezivanje na prethodno deployanu bazu ??

The screenshot shows the 'Environment Variables' section of a Render deployment configuration. It includes fields for DB_URL, DB_USERNAME, DB_PASS, DB_DRIVER, and PORT, all set to their respective values. Below this, there's a section for secret files and a health check path (/api/actuator/health). The Docker build context directory is set to 'IzvorniKod/Backend/' and the Dockerfile path is set to './prijava//Dockerfile'.

Variable	Value
DB_URL	a:8080/prijava_ostecenja_db
DB_USERNAME	prijava_ostecenja_db_user
DB_PASS	kT9yduov11tpa8qQsZM3PKTn2AM3t93
DB_DRIVER	org.postgresql.driver
PORT	8080

Health Check Path
If you're running a server, enter the path where your server will always return a `200 OK` response. We use it to monitor your app and for zero downtime deploys.
/api/actuator/health

Docker Build Context Directory
Docker build context directory. This is relative to your repository root. Defaults to the root.
IzvorniKod/Backend/

Dockerfile Path
Path to your Dockerfile relative to the repository root. This is *not* relative to your Docker build context. For example, `./subdir/Dockerfile`.
IzvorniKod/Backend/ ./prijava//Dockerfile

Slika 5.9: Unos varijabli okoline za backend

6. Zaključak i budući rad

Zadatak našeg tima bio je razvoj aplikacije za prijavu štete javnih površina. Aplikacija je trebala olakšati prijavu oštećenja javnih površina i cesta u gradovima i Provedba projekta bila je u dvije faze.

Prva faza projekta uključivala je formiranje tima za razvoj aplikacije, dodjelu projektnog zadatka i intenzivan rad na dokumentiranju zahtjeva. Kvalitetnom provedbom prve faze uvelike je olakšan daljnji rad na razvoju osmišljene aplikacije. Izrađeni obrasci i dijagrami (obrasci uporabe, sekvencijski dijagrami, model baze podataka, dijagrami razreda) omogućili su razvojnim timovima za *frontend* i *backend* lakši rad i organizaciju. Izrada vizualnog koncepta aplikacije uštedila je mnogo vremena u drugoj fazi razvoja, gdje je služila kao idejni temelj za razvoj. Tijekom prve faze projekta ostvarene su osnovne funkcionalnosti aplikacije.

Druga faza projekta bila je kraća, ali puno intenzivnija u razvoju aplikacije. Nedostatak iskustva članova u izradi sličnih projekata bio je izazov, no kroz projekt članovi tima imali su priliku upoznati i savladati razne alate i programske jezike kako bi uspješno realizirali projekt. Uz tehničke vještine članovi su imali priliku razvijati organizacijske vještine. Tijekom druge faze izrađeni su novi dijagrami (dijagram stanja, dijagram aktivnosti, dijagram komponenti, dijagram razmještaja) te unaprijeđeni već postojeći dijagrami (model baze podataka, dijagrami razreda). Dobro izrađen kostur projekta uštedio nam je mnogo vremena prilikom izrade aplikacije. Tijekom druge faze projekta ostvarena je većina funkcionalnosti aplikacije.

Tijekom projekta uspjeli smo realizirati većinu zamišljenih funkcionalnosti. One se odnose na funkcionalnosti za neregistriranog korisnika i registriranog korisnika, koji se dijeli na građanina i službenika gradskog ureda. Time smo ostvarili sve tražene funkcionalnosti zadane u projektnom zadatku. Iznimka od toga je obrazac uporabe UC20 koji se odnosi na ponovno postavljanje zaboravljene lozinke. Spomenuti obrazac uporabe nije zamišljen u projektnom zadatku nego kao ideja za korisnu funkcionalnost u aplikaciji, stoga zbog vremenske ograničenosti preostaje za budući rad na aplikaciji. Funkcionalnosti administratora također nam preostaju za budući rad. Konkretno je riječ o obrascima uporabe UC04, UC05, UC06, UC18, UC19, UC21, UC22 i UC23. U trenutnoj verziji aplikacije one se ostvaruju ručno u

bazi podataka.

Informiranost članova o napretku projekta ostvarili smo korištenjem aplikacije Discord i sastancima uživo. Svi članovi pokazali su zainteresiranost za projekt čime je rad bio ugodan i bez organizacijskih problema. Zadovoljni smo s realiziranim aplikacijom i zajedničkim radom na projektu.

Popis literature

1. Programsko inženjerstvo, FER ZEMRIS, http://www.fer.hr/predmet/pro_inz
2. I. Sommerville, "Software engineering", 8th ed, Addison Wesley, 2007.
3. T.C.Lethbridge, R.Langaniere, "Object-Oriented Software Engineering", 2nd ed. McGraw-Hill, 2005.
4. I. Marsic, "Software engineering book", Department of Electrical and Computer Engineering, Rutgers University, <http://www.ece.rutgers.edu/~marsic/books/SE>
5. N. Frid, A. Jović, "Modeliranje programske potpore UML-dijagramima", Fakultet elektrotehnike i računarstva Sveučilišta u Zagrebu, https://www.fer.unizg.hr/_download/repository/UML_zadaci_za_vjezbu_-_nadopuna_sveucilisnog_prirucnika%5B1%5D.pdf
6. Visual Paradigm, <https://www.visual-paradigm.com/>
7. Tehničko predavanje o backendu, <https://gitlab.com/hrvojesimic/progi-project-teams-backend/>
8. Tehničko predavanje o frontendu, <https://gitlab.com/jtomic/opp-project-teams-frontend>
9. Tehničko predavanje o deploymentu, <https://github.com/progi-devops>

Indeks slika i dijagrama

Dodatak: Prikaz aktivnosti grupe

Dnevnik sastajanja

1. sastanak

- Datum: 17. listopada 2023.
- Prisustvovali: Fran Fodor, Mateo Jakšić, Vedran Knežević, Leon Sattvik Kolenc, Jan Murić, Sara Podvorec, Ante Prkačin
- Teme sastanka:
 - formiranje tima
 - dogovor načina komunikacije
 - dogovor oko korištenih tehnologija

2. sastanak

- Datum: 19. listopada 2023.
- Prisustvovali: Fran Fodor, Mateo Jakšić, Vedran Knežević, Leon Sattvik Kolenc, Jan Murić, Sara Podvorec, Ante Prkačin
- Teme sastanka:
 - dogovor podjele poslova u početnoj fazi
 - analiza projektnog zadatka

3. sastanak

- Datum: 24. listopada 2023.
- Prisustvovali: Fran Fodor, Vedran Knežević, Ante Prkačin
- Teme sastanka:
 - sastanak s nastavnikom
 - upoznavanje s temom projekta

4. sastanak

- Datum: 27. listopada 2023.
- Prisustvovali: Fran Fodor, Mateo Jakšić, Vedran Knežević, Leon Sattvik Kolenc, Jan Murić, Sara Podvorec, Ante Prkačin
- Teme sastanka:
 - definiranje funkcijskih zahtjeva

- definiranje baze podataka

5. sastanak

- Datum: 31. listopada 2023.
- Prisustvovali: Fran Fodor, Mateo Jakšić, Sara Podvorec, Ante Prkačin
- Teme sastanka:
 - sastanak s nastavnikom
 - pojašnjenje zadatka

6. sastanak

- Datum: 6. studenoga 2023.
- Prisustvovali: Fran Fodor, Vedran Knežević, Ante Prkačin
- Teme sastanka:
 - razrada baze podataka

7. sastanak

- Datum: 7. studenoga 2023.
- Prisustvovali: Fran Fodor, Mateo Jakšić, Vedran Knežević, Leon Sattvik Kolenc
- Teme sastanka:
 - sastanak s nastavnikom
 - pojašnjenje dokumentacije

8. sastanak

- Datum: 13. studenoga 2023.
- Prisustvovali: Fran Fodor, Mateo Jakšić, Vedran Knežević, Leon Sattvik Kolenc, Jan Murić, Sara Podvorec, Ante Prkačin
- Teme sastanka:
 - analiza dosadašnjeg napretka
 - postavljanje daljnjih ciljeva

9. sastanak

- Datum: 14. studenoga 2023.
- Prisustvovali: Fran Fodor, Mateo Jakšić, Sara Podvorec
- Teme sastanka:
 - sastanak s nastavnikom
 - pregled ostvarenog

10. sastanak

- Datum: 17. studenoga 2023.

- Prisustvovali: Fran Fodor, Mateo Jakšić, Vedran Knežević, Leon Sattvik Kolenc, Jan Murić, Sara Podvorec, Ante Prkačin
- Teme sastanka:
 - analiza projekta prije predaje

11. sastanak

- Datum: 5. prosinca 2023.
- Prisustvovali: Fran Fodor, Mateo Jakšić, Vedran Knežević, Leon Sattvik Kolenc, Jan Murić, Sara Podvorec, Ante Prkačin
- Teme sastanka:
 - evaluacija prvog dijela

12. sastanak

- Datum: 11. prosinca 2023.
- Prisustvovali: Fran Fodor, Mateo Jakšić, Vedran Knežević, Leon Sattvik Kolenc, Sara Podvorec, Ante Prkačin
- Teme sastanka:
 - planiranje daljnog rada

13. sastanak

- Datum: 12. prosinca 2023.
- Prisustvovali: Fran Fodor, Ante Prkačin
- Teme sastanka:
 - sastanak s nastavnikom

14. sastanak

- Datum: 19. prosinca 2023.
- Prisustvovali: Fran Fodor, Ante Prkačin
- Teme sastanka:
 - neodržani sastanak s nastavnikom

15. sastanak

- Datum: 9. siječnja 2024.
- Prisustvovali: Fran Fodor, Leon Sattvik Kolenc, Sara Podvorec, Ante Prkačin
- Teme sastanka:
 - sastanak s nastavnikom
 - demonstracija napretka implementacije funkcionalnosti i dokumentacije

16. sastanak

- Datum: 9. siječnja 2024.
- Prisustvovali: Leon Sattvik Kolenc, Ante Prkačin
- Teme sastanka:
 - izrada funkcionalnosti aplikacije

17. sastanak

- Datum: 10. siječnja 2024.
- Prisustvovali: Fran Fodor, Vedran Knežević, Leon Sattvik Kolenc, Ante Prkačin
- Teme sastanka:
 - izrada funkcionalnosti aplikacije

18. sastanak

- Datum: 11. siječnja 2024.
- Prisustvovali: Fran Fodor, Mateo Jakšić, Vedran Knežević, Leon Sattvik Kolenc, Jan Murić, Sara Podvorec, Ante Prkačin
- Teme sastanka:
 - planiranje završne faze izrade projekta

19. sastanak

- Datum: 16. siječnja 2024.
- Prisustvovali: Fran Fodor, Sara Podvorec, Ante Prkačin
- Teme sastanka:
 - sastanak s nastavnik

Tablica aktivnosti

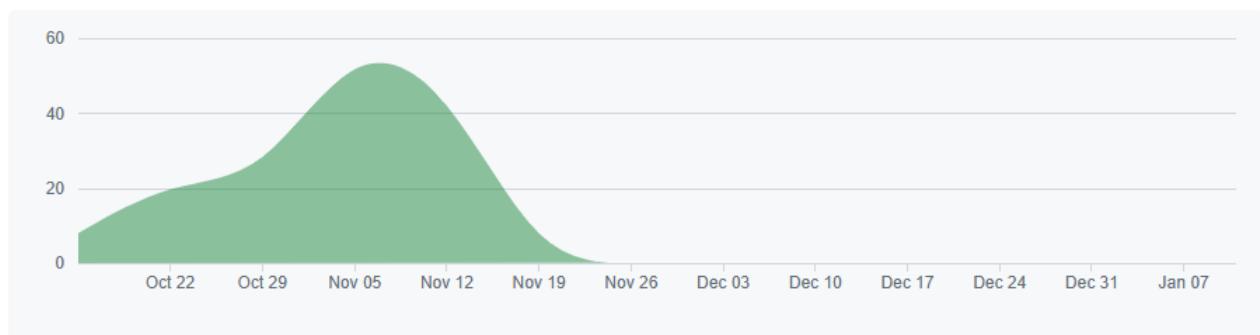
	Fran Fodor	Mateo Jakšić	Vedran Knežević	Leon Sattvik Kolenc	Jan Murić	Sara Podvorec	Ante Prkačin
Upravljanje projektom	15	13	12	11	10	12	12
Opis projektnog zadatka					1	6	
Funkcionalni zahtjevi	1	2	1	1	1	2	1
Opis pojedinih obrazaca		8					
Dijagram obrazaca		6				2	
Sekvencijski dijagrami		6				1	
Opis ostalih zahtjeva		2					
Arhitektura i dizajn sustava						5	
Baza podataka		5				4	
Dijagram razreda		8					
Dijagram stanja						8	
Dijagram aktivnosti						5	
Dijagram komponenti							
Korištene tehnologije i alati							
Ispitivanje programskog rješenja							
Dijagram razmještaja							
Upute za puštanje u pogon							
Dnevnik sastajanja		1					
Zaključak i budući rad							

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

	Fran Fodor	Mateo Jakšić	Vedran Knežević	Leon Sattvik Kolenc	Jan Murić	Sara Podvorec	Ante Prkačin
Popis literature		1					
Učenje Springa			5		6	1	12
Učenje Reacta	3			5	6		
Učenje Tailwinda	3						
Izrada dizajna stranice	2			4			
Pisanje README.md	1	1		1		1	1
Rad na arhitekturi baze	4	1	4	1	1	1	4
Implementacije arhitekture baze			3				
Prikaz mape i funkcionalnost markera	4			3			
Izrada prijave i registracije korisnika				7	3		
Prikazivanje prijava oštećenja				5			
Organizacija backenda							5
Sigurnost - Spring Security							6
Testiranje API-a					2		4
Priprema za deployment i deployment	3			4			2

Dijagrami pregleda promjena



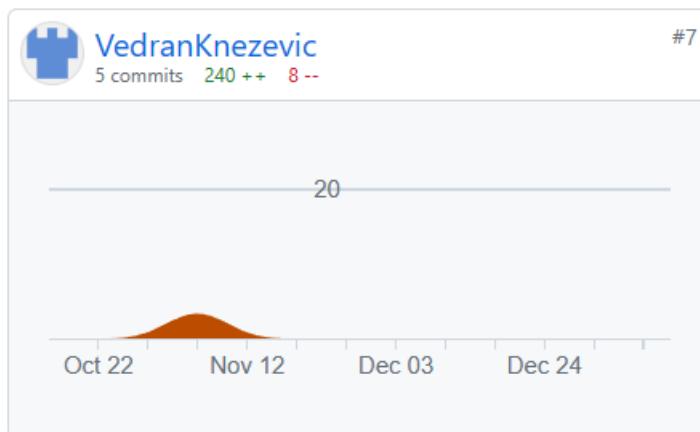
Slika 6.1: Dijagram pregleda promjena - CestaFix



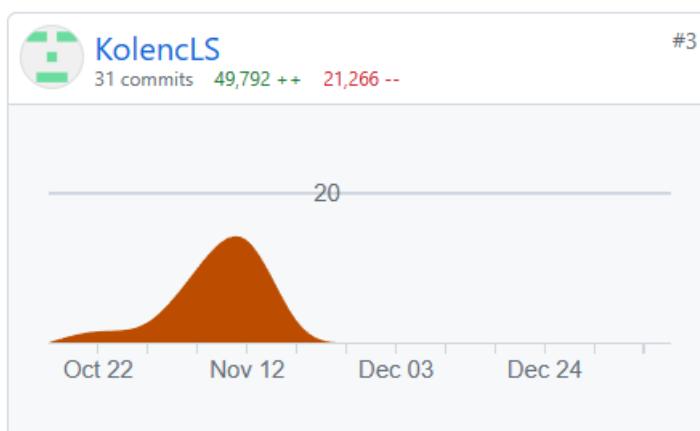
Slika 6.2: Dijagram pregleda promjena - Fran Fodor



Slika 6.3: Dijagram pregleda promjena - Mateo Jakšić



Slika 6.4: Dijagram pregleda promjena - Vedran Knežević



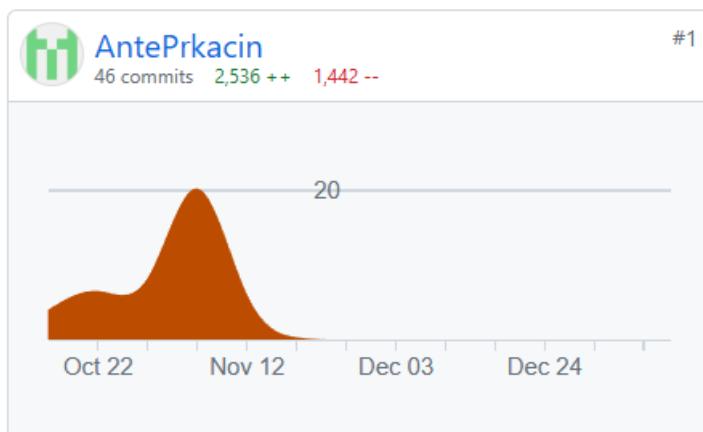
Slika 6.5: Dijagram pregleda promjena - Leon Sattvik Kolenc



Slika 6.6: Dijagram pregleda promjena - Jan Murić



Slika 6.7: Dijagram pregleda promjena - Sara Podvorec



Slika 6.8: Dijagram pregleda promjena - Ante Prkačin