

Programsko inženjerstvo

Ak. god. 2022./2023.

DogFriendly

Dokumentacija, Rev. 2

Grupa: *e404TeamNotFound*

Voditelj: *Leon Stjepan Uroić*

Datum predaje: *13.01.2023.*

Nastavnik: *Laura Majer*

Sadržaj

1 Dnevnik promjena dokumentacije	3
2 Opis projektnog zadatka	6
2.1 Cilj i opis zadatka	6
2.2 Problematika zadatka	6
2.3 Korisnički zahtjevi	7
2.4 Inovacije koje pruža	8
2.5 Slični projekti	8
3 Specifikacija programske potpore	10
3.1 Funkcionalni zahtjevi	10
3.1.1 Obrasci uporabe	12
3.1.2 Sekvencijski dijagrami	19
3.2 Ostali zahtjevi	23
4 Arhitektura i dizajn sustava	24
4.1 Baza podataka	25
4.1.1 Opis tablica	26
4.1.2 Dijagram baze podataka	29
4.2 Dijagram razreda	31
4.2.1 Dijagram razreda nakon završene aplikacije	36
4.3 Dijagram stanja	37
4.4 Dijagram aktivnosti	38
4.5 Dijagram komponenti	40
5 Implementacija i korisničko sučelje	41
5.1 Korištene tehnologije i alati	41
5.2 Ispitivanje programskog rješenja	42
5.2.1 Ispitivanje komponenti	42
5.2.2 Ispitivanje sustava	46
5.3 Dijagram razmještaja	50

5.4 Upute za puštanje u pogon	51
5.4.1 Konfiguracija baze podataka	51
5.4.2 Backend poslužitelj	52
5.4.3 Frontend poslužitelj	53
6 Zaključak i budući rad	55
Indeks slika i dijagrama	58

1. Dnevnik promjena dokumentacije

Rev.	Opis promjene/dodatka	Autori	Datum
0.1	Napravljen predložak dokumentacije.	Luka Marković	26.10.2022
0.2	Dodani dionici i funkcionalni zahtjevi.	Nela Štobelj	28.10.2022
0.3	Dodan opis projektnog zadatka.	Filip Jakovina	29.10.2022
0.4	Dodani obrasci uporabe i UML dijagrami.	Mario Hošnjak, Zoa Horvat	31.10.2022
0.5.1	Arhitektura sustava	David Winkler	3.11.2022.
0.5.2	Arhitektura baze podataka	Leon Stjepan Uročić	2.11.2022
0.6.1	Ažuriran opis	Luka Marković	3.11.2022.
0.6.2	Izmjenjen UML i UC	Mario Hošnjak, Zoa Horvat	3.11.2022.
0.6.3	Ažurirana arhitektura sustava	David Winkler	4.11.2022.
0.7	Dizajn korisničkog sučelja početne stranice i stranice za prijavu korisnika	Nela Štobelj	6.11.2022.

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

Rev.	Opis promjene/dodataka	Autori	Datum
0.8	Ažurirana dokumentacija baze podataka	Leon Stjepan Uroić	7.11.2022.
0.9	Dodani sekvencijski dijagrami i ostali zahtjevi	Mario Hošnjak	10.11.2022.
0.10	Napravljeni dijagrami razreda	Filip Jakovina	16.11.2022.
0.11	Korekcije i priprema prve verzije dokumentacije	Luka Marković	18.11.2022.
1.0	Verzija samo s bitnim dijelovima za 1. ciklus	Svi	17.11.2022.
1.1	Upute za puštanje u pogon	Leon Stjepan Uroić	05.01.2023.
1.2	Dodan dijagram stanja	Luka Marković	10.01.2023.
1.3	Dodan dijagram komponenti	Luka Marković	10.01.2023.
1.3	Dodani korištene tehnologije i alati	Luka Marković	10.01.2023.
1.4	Dodan dijagram aktivnosti	Nela Štubelj	11.01.2023.
1.5	Dodan konceptualni dijagram razreda	Filip Jakovina	12.01.2023.
1.6	Dodan dijagram razmještaja	Filip Jakovina	13.01.2023.
1.7	Dodano ispitivanje sustava	Mario Hošnjak	13.01.2023.

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

Rev.	Opis promjene/dodataka	Autori	Datum
1.8	Dodano ispitivanje komponenti	Luka Marković, Filip Jakovina	13.01.2023.
2.0	Završna verzija dokumentacije	Svi	13.01.2023.

2. Opis projektnog zadatka

2.1 Cilj i opis zadatka

Cilj ovog projekta je razviti programsku podršku za web aplikaciju "Dog Friendly" koja će omogućiti korisnicima da na interaktivnoj karti pronađu prikladne lokacije za druženje sa svojim ljubimcima i time im olakšati kretanje. U svrhu financiranja aplikacije, omogućit ćemo vlasnicima obrta koja su povezana sa psima da postave svoju reklamu na stranicu uz određenu pretplatu.

2.2 Problematika zadatka

Aplikacija Dog Friendly omogućuje svojim korisnicima kretanje po interaktivnoj karti, ručni unos tržene adrese te, uz dozvolu, lociranje vlastitog uređaja. Korisnici pristupaju aplikaciji preko web stranice gdje se mogu prijaviti kao osnovni korisnik ili vlasnik obrta. Odabirom vrste korisnika im se otvara tražena registracijska forma.

Za stvaranje računa osnovnog korisnika potrebno je:

- **ime**
- **prezime**
- **korisničko ime**
- **e-mail**
- **lozinku**
- **opis**

Registriranim korisnicima je osim osnovnih funkcija omogućeno postavljanje novih lokacija, označavanje lokacija prikladnim/neprikladnim za pse, ostavljanje recenzija i komentara na lokacije.

Za vlasnike obrta je dodatno potrebno unijeti:

- **ime obrta**
- **OIB**
- **e-mail**
- **lozinka**
- **broj telefona**
- **opis**
- **tip obrta**
- **broj kartice**
- **CVV**
- **datum isteka kartice**

Vlasnici obrta nakon registracije na karti odabiru adrese svojih obrta i postavljaju na njih markere. Prilikom odabira lokacije osnovnom korisniku se prikazuje adresa i javni dio podataka unesen prilikom registracije (ime obrta, kontakt i opis). Na dnu obrasca za registraciju se nalazi "mock" obrazac za unos kartičnih podataka (obavezno za vlasnike obrta).

2.3 Korisnički zahtjevi

Neregistrirani korisnik može se kretati po karti i vidjeti sve prikladne i neprikladne lokacije i plaćene lokacije (obrte) koje su mu dodatno istaknute. U polju za pretragu može unijeti specifičnu lokaciju koja će mu se centrirati na karti. Moguće je odabrati kategoriju čiji se markeri onda prikazuju. Klikom na marker moguće je saznati više informacija o lokaciji kao što su njezin naziv, ocjena i tip lokacije. Klikom na marker obrta dodatno je moguće vidjeti naziv, adresu, opis obrta i kontakt. Osim toga, korisnik može omogućiti lociranje vlastitog uređaja.

Korisnik može poslati zahtjev za registracijom te potom ispuniti tražene podatke.

Registracija se potvrđuje preko e-mail potvrde i korisnik dobiva potvrdu o uspješnoj registraciji. Vlasnik obrta uz potvrdu o registraciji dobiva i potvrdu o plaćanju.

Osnovnom korisniku je, osim svih funkcionalnosti neregistriranog korisnika, omogućeno označavanje lokacija kao prikladne i neprikladne. Prilikom označavanja potrebno je unijeti ime lokacije i odabir kategorije iz izbornika. Također za već postojeće lokacije moguće je potvrditi ili negirati postojeću oznaku i dodati recenziju. Korisniku je omogućena promjena korisničkog imena i lozinke.

Vlasnik obrta može pregledavati i odgovarati na recenzije svog obrta. Omogućena mu je promjena korisničkog imena i lozinke, promjena naziva, opisa i kategorije obrta te dodavanje jedne ili više adresa na kojima se obrt nalazi. Pretplatom na aplikaciji osigurava da njegove lokacije budu drugačije istaknute od običnih lokacija.

2.4 Inovacije koje pruža

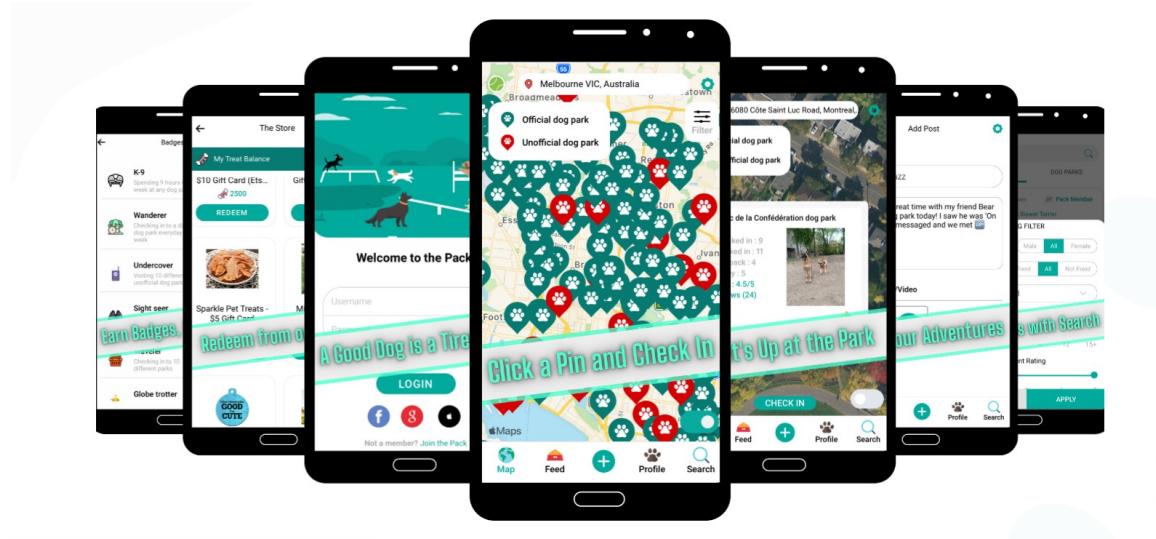
Aplikacija bi bila prva takva u Hrvatskoj koja ne samo da pruža pronalazak prikladnih lokacija za pse, nego korisnici mogu pronaći i neprikladne lokacije kako bi ih mogli izbjjeći. Osim toga, aplikacija nudi i prikaz obrta kao što su: pet shopovi, veterinarske ordinacije, frizerski saloni za pse, itd. To omogućuje našim korisnicima da na jednom mjestu pronađu sve potrebno za njihove ljubimce.

2.5 Slični projekti

Pronalazak parkova za pse i različitih potrepština je posebno izazovan za nove vlasnike i korisnike koji se po prvi put nađu u novom gradu. Zbog toga na tržištu postoji potreba za ovakvom aplikacijom, ali i slična rješenja

DogPack

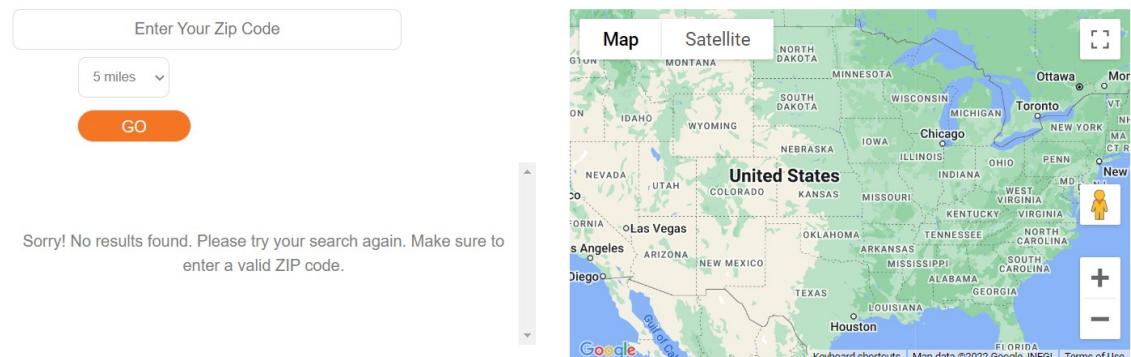
DogPack je mobilna aplikacija koja uključuje detaljnu kartu parkova za pse u brojnim zemljama, pa tako i Hrvatskoj. Nudi mogućnost lokacije traženog parka, prijave za dolazak u odabrani park i popis trenutnih posjetitelja parku. Aplikacija korisnicima omogućuje izradu profila, komunikaciju s drugim korisnicima, objavljanje slika i videozapisa i slično.



Slika 2.1: Prikaz mogućnosti koje nudi aplikacija DogPack

Dog Park Finder

Aplikacija tvrtke Nylabone omogućuje korisnicima da unesu svoj poštanski broj i radius pretrage i aplikacija prikazuje na interaktivnoj karti lokacije svih parkova za pse u zadanom radiusu. Aplikacije je ograničena na korisnike iz SAD-a.



Slika 2.2: Prikaz karte

Sniffspot

Aplikacija omogućuje svojim korisnicima da unajme sigurne i privatne pseće par-kove gdje možete sami sa svojim ljubimcem provoditi kvalitetno vrijeme. Aplika-cija je ograničena na korisnike iz SAD-a.

3. Specifikacija programske potpore

3.1 Funkcionalni zahtjevi

Dionici:

1. Vlasnik obrta (naručitelj)
2. Korisnici
3. Razvojni tim

Aktori i njihovi funkcionalni zahtjevi:

1. Vlasnik obrta (inicijator) može:
 - (a) dodati ili obrisati obrt
 - (b) dodati, izbrisati ili promijeniti
 - i. naziv i opis obrta
 - ii. kategoriju lokacije
 - (c) pregledavati i odgovarati na recenzije korisnika
 - (d) pretplatom na aplikaciju osigurati neke dodatne pogodnosti poput reklame
2. Neregistrirani korisnik (inicijator) može:
 - (a) pregledati na karti prikladno ili neprikladno označene lokacije za pse, te plaćene lokacije koje su dodatno istaknute
 - (b) pretraživati specifičnu lokaciju ili odabrati kategoriju čiji se markeri onda prikazuju
 - (c) odabirom markirane lokacije saznati više informacija o odabranom položaju (ime lokacije, ocjenu i kategorija) ili u slučaju obrta (naziv obrta, ocjenu, tip obrta, opis obrta i kontakt)
 - (d) se registrirati u sustav stvaranjem novog korisničkog računa

3. Registrirani korisnik (inicijator) može:

- (a) označavati novu lokaciju što uključuje unos imena lokacije te odabir njene kategorije
- (b) potvrditi ili negirati oznaku neke već postojeće lokacije
- (c) pregledavati i mijenjati osobne podatke
- (d) izbrisati svoj korisnički račun
- (e) pisati recenziju i dati ocjenu

4. Baza podataka (sudionik) može:

- (a) pohranjuje sve podatke o
 - i. korisnicima i njihovim ovlastima
 - ii. obrtimu i njihovoj kategoriji

3.1.1 Obrasci uporabe

Opis obrazaca uporabe

UC1: Registracija korisnika

- **Glavni sudionik:** Korisnik
- **Cilj:** Omogućiti korisniku prijavu u sustav, čime će korisnik ostvariti pravo na korištenje više funkcionalnosti aplikacije
- **Sudionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
 1. Korisnik odabire opciju za registraciju
 2. Korisnik u registracijskom sučelju popunjava formu sa svojim podacima
 3. Korisnik dobiva e-mail na unesenu adresu u kojem potvrđuje svoju registraciju
- **Opis mogućih odstupanja:**
 - Korisnik nije unio sve obavezne podatke
 - Odabir već zauzetog korisničkog imena i/ili e-maila
 - Unos podatka u nedozvoljenom formatu

UC2: Prijava

- **Glavni sudionik:** Korisnik, vlasnik obrta
- **Cilj:** Dobiti pristup korisničkom sučelju s dodatnim funkcionalnostima
- **Sudionici:** Baza podataka
- **Preduvjet:** Uspješna registracija
- **Opis osnovnog tijeka:**
 1. Unos korisničkog imena i lozinke u formu za prijavu
 2. Potvrda o ispravnosti unešenih podataka
 3. Redirekcija na korisničko sučelje aplikacije

- **Opis mogućih odstupanja:**
 - Pogrešno korisničko ime i/ili lozinka

UC3: Pregled osobnih podataka

- **Glavni sudionik:** Korisnik, vlasnik obrta
- **Cilj:** Omogućiti pregled osobnih podataka
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen
- **Opis osnovnog tijeka:**
 1. Korisnik odabire opciju za prikaz osobnih podataka
 2. Prikaz osobnih podataka korisnika

UC4: Promjena osobnih podataka korisnika

- **Glavni sudionik:** Korisnik, vlasnik obrta
- **Cilj:** Omogućiti promjenu korisničkog imena i/ili lozinke korisničkog računa
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen
- **Opis osnovnog tijeka:**
 1. Korisnik odabire opciju za promjenom osobnih podataka
 2. Korisnik unosi nove osobne podatke i spremi promjene
 3. Promjene se spremaju u bazu podataka
- **Opis mogućih odstupanja:**
 - Novo korisničko ime je zauzeto

UC5: Brisanje korisničkog računa

- **Glavni sudionik:** Korisnik
- **Cilj:** Korisnik briše svoj korisnički račun

- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen
- **Opis osnovnog tijeka:**
 1. Korisnik odabire opciju za brisanje svog korisničkog računa
 2. Odjava iz aplikacije
 3. Korisnički račun se briše iz baze podataka
 4. Redirekcija na naslovnu stranicu

UC6: Promjena podataka obrta

- **Glavni sudionik:** Vlasnik obrta
- **Cilj:** Omogućiti vlasnicima obrta promjenu podataka o obrtima
- **Sudionici:** Baza podataka
- **Preduvjet:** Vlasnik obrta je prijavljen
- **Opis osnovnog tijeka:**
 1. Vlasnik obrta odabire opciju za promjenu podataka obrta
 2. Vlasnik mijenja ime, opis, kategoriju, OIB, kontakt broj i/ili neki drugi podatak
 3. Vlasnik spremi promjene
 4. Promjene se spremaju u bazu podataka

UC7: Brisanje računa vlasnika obrta

- **Glavni sudionik:** Vlasnik obrta
- **Cilj:** Omogućiti vlasniku obrta brisanje računa
- **Sudionici:** Baza podataka
- **Preduvjet:** Vlasnik obrta je prijavljen
- **Opis osnovnog tijeka:**
 1. Vlasnik obrta odabire opciju za brisanje računa

2. Odjava iz aplikacije
3. Račun vlasnika obrta se briše iz baze podataka
4. Redirekcija na naslovnu stranicu

UC8: Pregled podataka lokacije

- **Glavni sudionik:** Korisnik, vlasnik obrta
- **Cilj:** Pregled podataka bilo koje unesene lokacije
- **Sudionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
 1. Učitava se karta sa unesnim lokacijama
 2. Klikom na pojedinu lokaciju prikazuju se njeni podaci

UC9: Potvrđivanje ili negiranje oznake postojeće lokacije

- **Glavni sudionik:** Korisnik, vlasnik obrta
- **Cilj:** Omogućiti prijavljenim korisnicima da potvrde ili negiraju oznaku postojeće lokacije
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik ili vlasnik obrta su prijavljeni
- **Opis osnovnog tijeka:**
 1. Kod prikaza informacija pojedine lokacije, korisnik odabire opciju potvrđivanja ili negiranja lokacije
 2. Informacija se sprema u bazu podataka

UC10: Označavanje novih lokacija

- **Glavni sudionik:** Korisnik, vlasnik obrta
- **Cilj:** Omogućiti unos neke lokacije koja je (ne)prikladna za pse
- **Sudionici:** Baza podataka

- **Preduvjet:** Korisnik ili vlasnik obrta su prijavljeni
- **Opis osnovnog tijeka:**
 1. Korisnik odabire opciju za označavanjem nove lokacije
 2. Korisnik unosi ime i kategoriju lokacije te je li ona prikladna za pse
 3. Podaci se spremaju u bazu podataka

UC11: Prikaz lokacija po odabranoj kategoriji

- **Glavni sudionik:** Korisnik, vlasnik obrta
- **Cilj:** Omogućiti prikaz lokacija na karti po odabranoj kategoriji
- **Sudionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
 1. Korisnik na karti odabire kategoriju po kojoj želi filtrirati lokacije
 2. Na karti se prikazuju samo lokacije sa odabranom kategorijom

UC12: Registracija vlasnika obrta

- **Glavni sudionik:** Vlasnik obrta
- **Cilj:** Omogućiti vlasniku obrta prijavu u sustav, čime će vlasnik obrta ostvariti pravo na korištenje više funkcionalnosti aplikacije
- **Sudionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
 1. Vlasnik obrta odabire opciju za registraciju
 2. Vlasnik obrta u registracijskom sučelju popunjava formu sa svojim podacima i podacima obrta
 3. Vlasnik obrta na unesenu e-mail adresu dobiva potvrdu o uspješnoj registraciji i uspješnom plaćanju
- **Opis mogućih odstupanja:**

- Vlasnik obrta nije unio sve obavezne podatke
- Odabir već zauzetog korisničkog imena i/ili e-maila
- Unos podatka u nedozvoljenom formatu

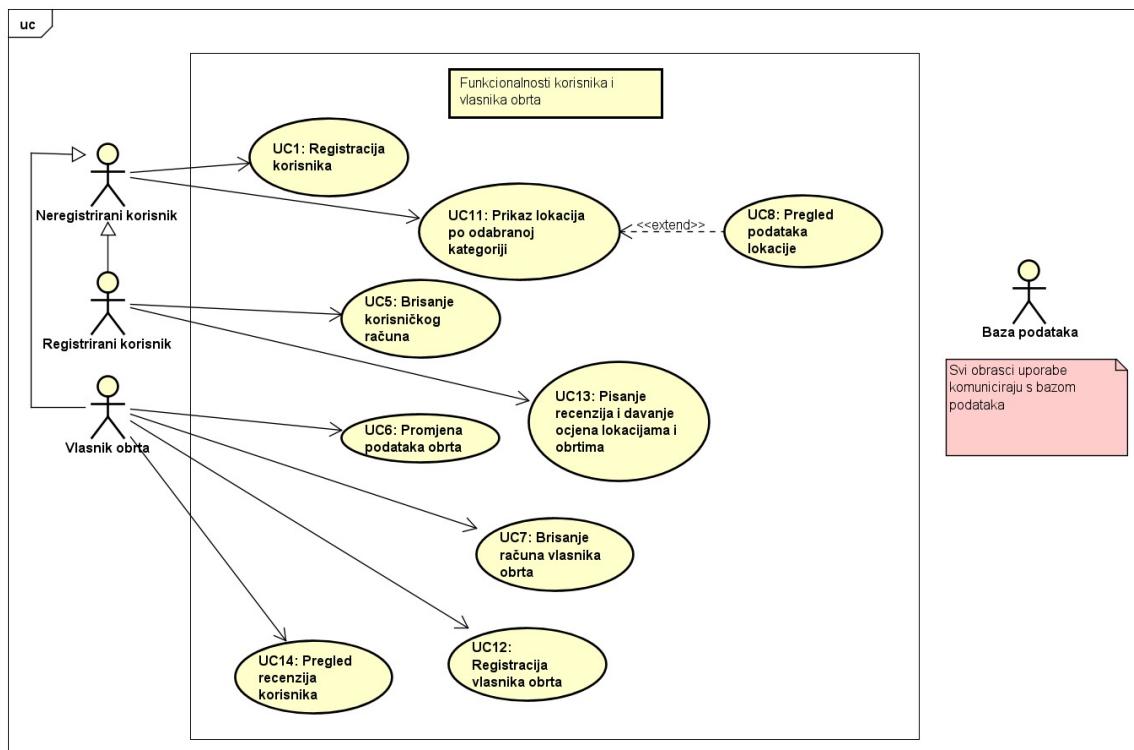
UC13: Pisanje recenzija i davanje ocjena lokacijama i obrtima

- **Glavni sudionik:** Korisnik
- **Cilj:** Napisati recenziju i ocijeniti lokaciju ili obrt
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen
- **Opis osnovnog tijeka:**
 1. Korisnik napiše recenziju i ocijeni lokaciju ili obrt
 2. Ocjena i recenzija se pohranjuju u bazu podataka i ažurira se prosječna ocjena lokacije ili obrta
- **Opis mogućih odstupanja:**
 - Korisnik ne želi napisati osvrt

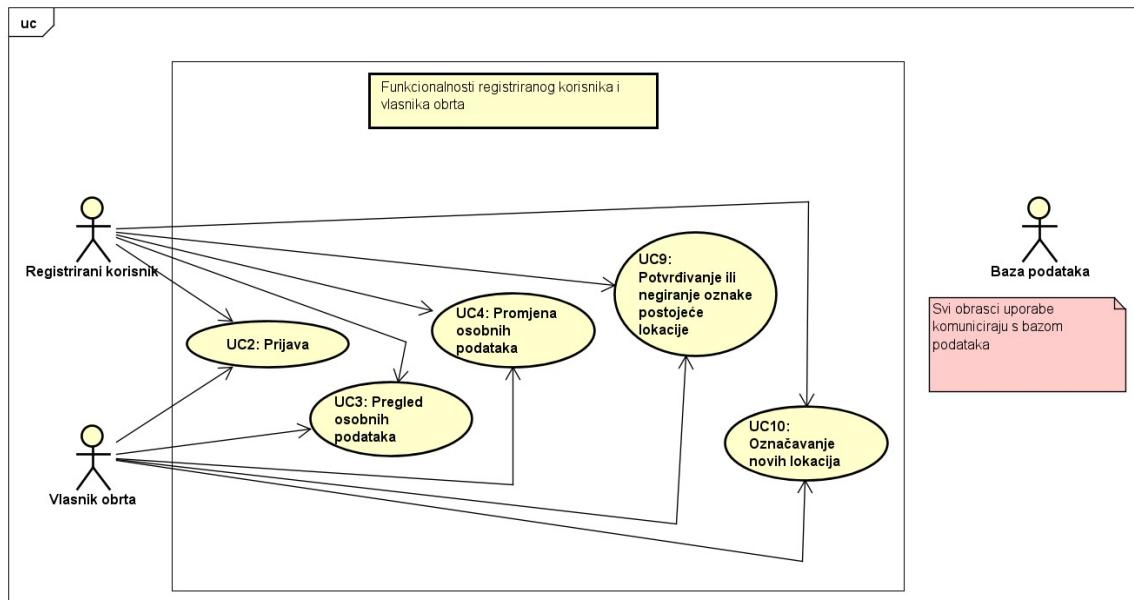
UC14: Pregled recenzija korisnika

- **Glavni sudionik:** Vlasnik obrta
- **Cilj:** Pregledati recenzije korisnika za vlasnikov obrt
- **Sudionici:** Baza podataka
- **Preduvjet:** Vlasnik obrta je prijavljen
- **Opis osnovnog tijeka:**
 1. Vlasnik odabire opciju “Pregledaj recenzije”
 2. Prikažu se sve dotadašnje recenzije i ocjene vlasnikovog obrta

Dijagrami obrazaca uporabe



Slika 3.1: Funkcionalnost korisnika i vlasnika obrta

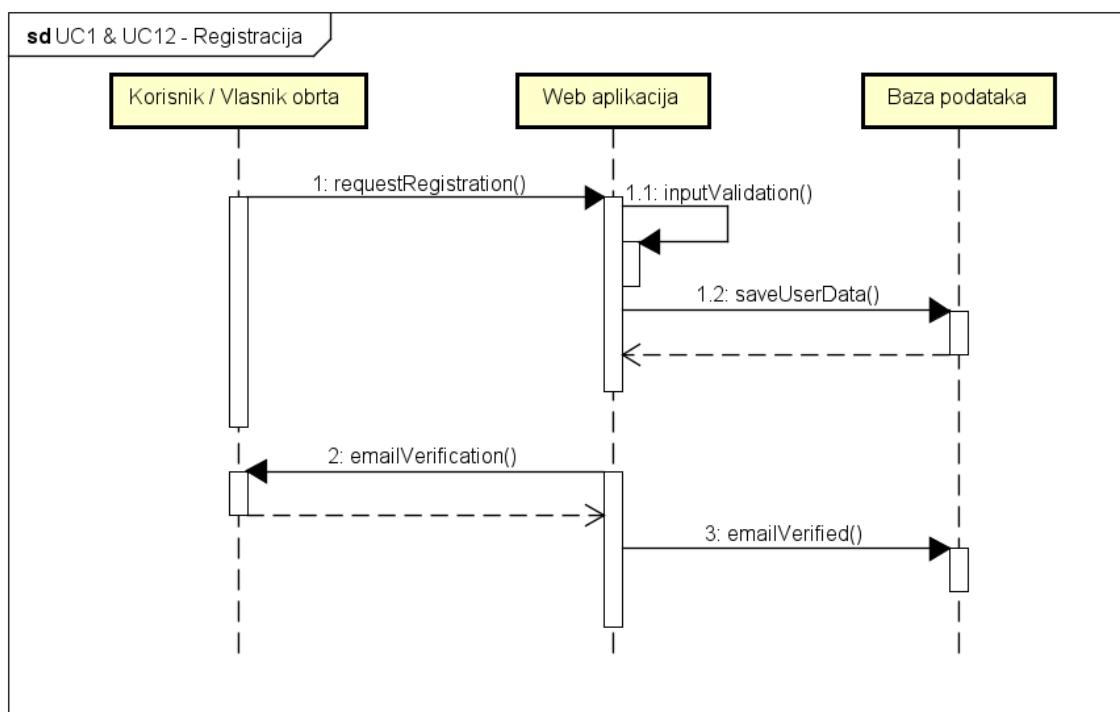


Slika 3.2: Funkcionalnost registriranog korisnika i vlasnika obrta

3.1.2 Sekvencijski dijagrami

Obrasci uporabe UC1 i UC12 - Registracija

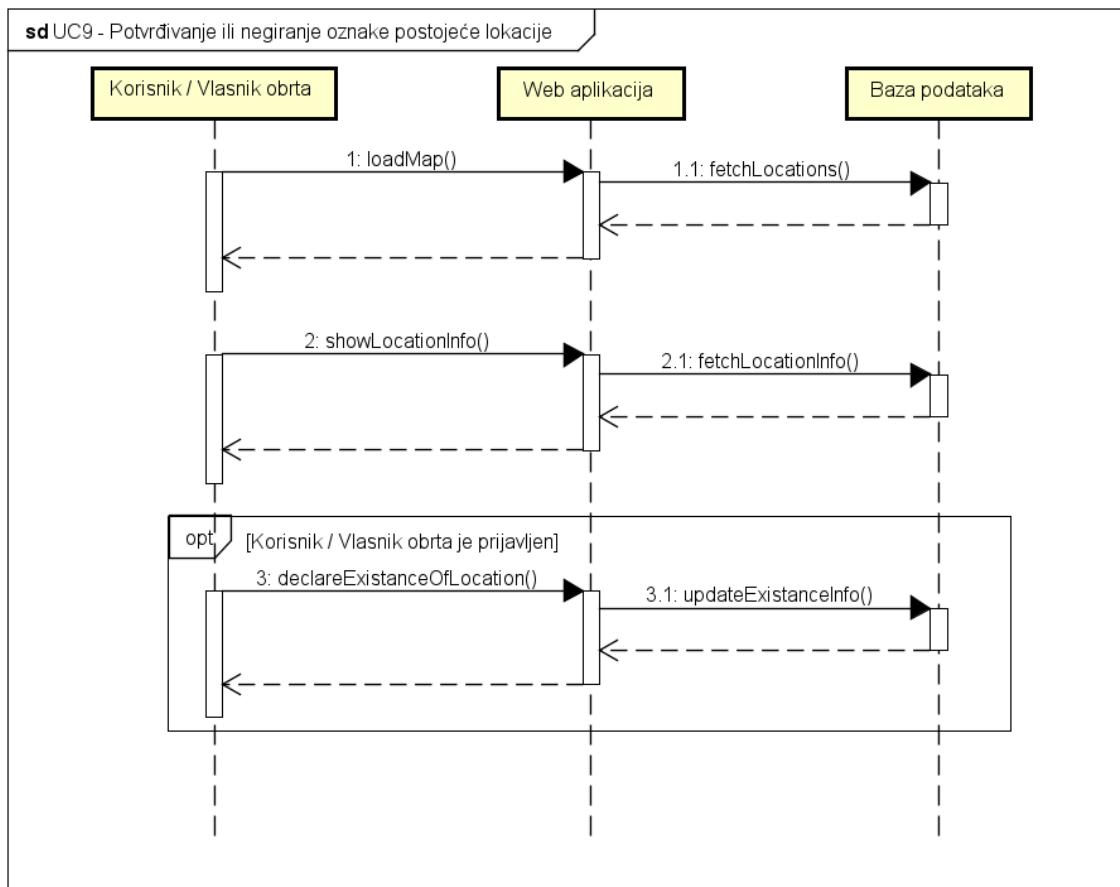
Korisnik ili vlasnik obrta šalje zahtjev za registracijom koji za osnovne korisnike sadrži e-mail adresu, korisničko ime, lozinku, ime, prezime i opis. Zahtjev za registraciju vlasnika obrta sadrži e-mail adresu, lozinku te naziv, adresu, OIB, kontakt broj, djelatnost i kratki opis obrta. Web aplikacija validira ulazne podatke te se podaci spremaju u bazu podataka. Korisniku se nakon toga na navedenu e-mail adresu šalje link za verifikaciju računa.



Slika 3.3: Sekvencijski dijagram registracije korisnika ili vlasnika obrta

Obrazac uporabe UC9 - Potvrđivanje ili negiranje oznake postojeće lokacije

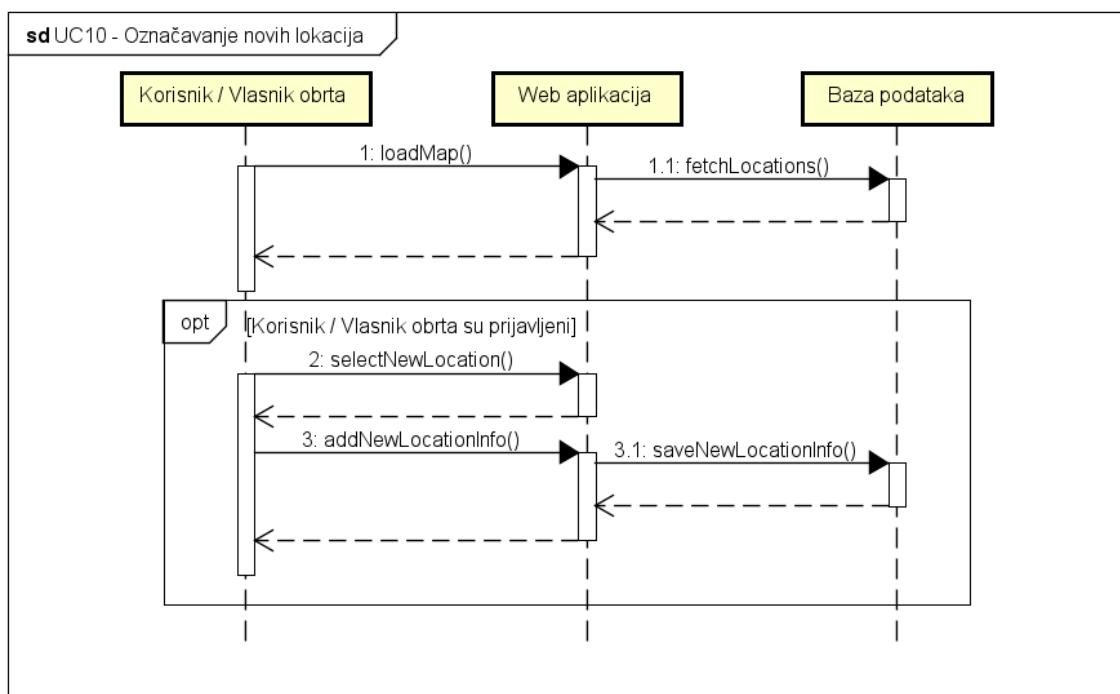
Korisnik ili vlasnik obrta šalje zahtjev za učitavanjem karte na naslovnoj stranici, podaci se povlače iz baze podataka te se lokacije iscrtavaju na karti. Klikom na neku od lokacija korisniku se prikazuju osnovni podaci odabrane lokacije ili obrta. Ukoliko je korisnik(ili vlasnik obrta) prijavljen, on može potvrditi ili negirati postojanje označene lokacije, što će se zatim spremiti u bazu podataka.



Slika 3.4: Sekvencijski dijagram potvrđivanja ili negiranja postojećih lokacija

Obrazac uporabe UC10 - Označavanje novih lokacija

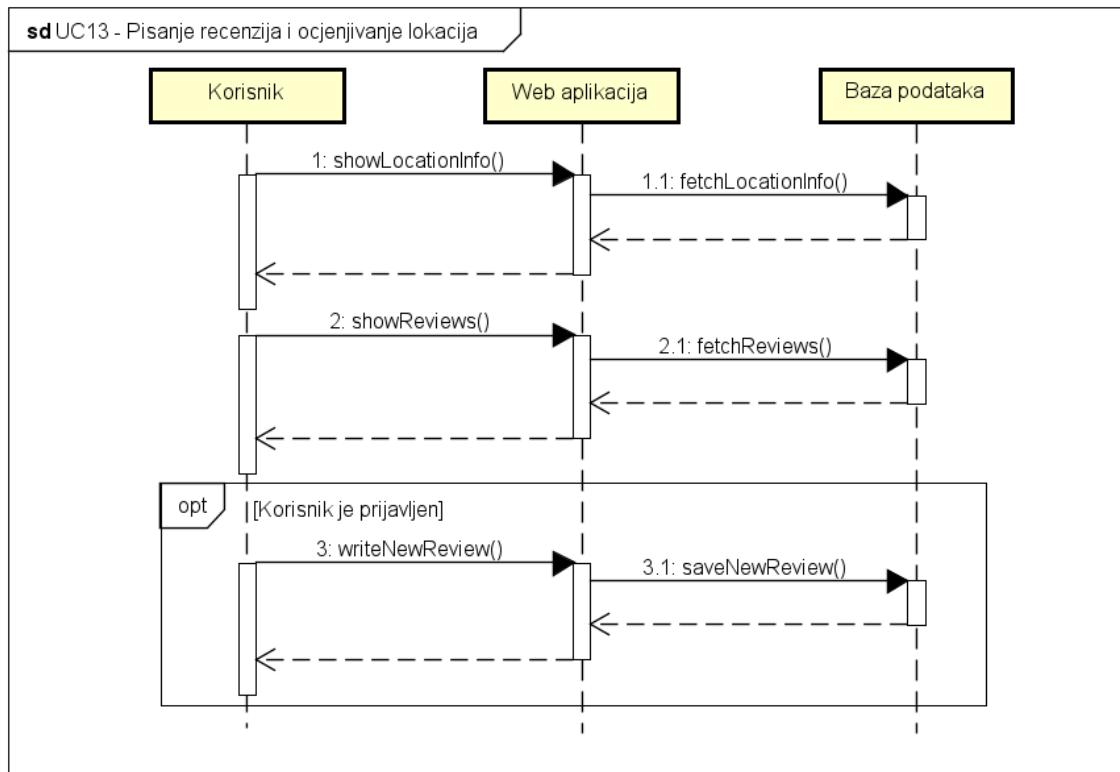
Korisnik prilikom učitavanja naslovne stranice šalje zahtjev za učitavanjem karte s lokacijama, uslijed čega web aplikacija povlači iz baze podataka potrebne podatke o lokacijama te ih iscrtava na karti. Registrirani i prijavljeni korisnik može označiti novu lokaciju na karti te za nju navesti ime lokacije, kategoriju te je li prikladna za pse. Uneseni podaci se spremaju u bazu podataka.



Slika 3.5: Sekvencijski dijagram označavanja novih lokacija

Obrazac uporabe UC13 - Pisanje recenzija i ocjenjivanje lokacija

Korisniku se klikom na marker lokacije prikazuju osnovni podaci unesene lokacije ili obrta. Nadalje, korisnik može poslati zahtjev za recenzijama odabrane lokacije koje se zatim povlače iz baze podataka. Ukoliko je korisnik prijavljen, on može napisati vlastitu recenziju i/ili ocijeniti lokaciju s jednom do pet zvjezdica. Napisana recenzija i/ili odabrana ocjena se zatim spremaju u bazu podataka.



Slika 3.6: Sekvencijski dijagram pisanja recenzija i ocjenjivanja lokacija

3.2 Ostali zahtjevi

- Dohvaćanje podataka iz baze podataka ne smije trajati dulje od nekoliko sekundi
- Lozinke i ostali povjerljivi podaci u bazi podataka moraju biti kriptirani
- Neispravno korištenje korisničkog sučelja i neispravan unos podataka ne smiju narušiti funkcionalnost aplikacije i integritet baze podataka
- Sustav treba biti implementiran kao web aplikacija koristeći objektno-orientirane jezike
- Sustav treba funkcionirati pri istovremenom korištenju više korisnika
- Korisničko sučelje treba biti jednostavno i intuitivno za korištenje
- Nadogradnje sustava ne smiju narušavati postojeće funkcionalnosti sustava
- Veza s bazom podataka mora biti kvalitetno zaštićena, brza i otporna na vanjske greške
- Pristup sustavu mora biti omogućen iz javne mreže pomoću HTTPS

4. Arhitektura i dizajn sustava

Arhitektura se može podijeliti na tri podsustava:

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

Web preglednik je aplikacija za pristup World Wide Web-u. Kada korisnik želi otvoriti web stranicu, web preglednik šalje HTTP protokol preko kojega zahtjeva sadržaj stranice. Nakon što dobije sadržaj stranice web preglednik ga prevodi i prikazuje korisniku kao jasno i razumljivo grafičko sučelje.

Web poslužitelj je računalo na kojem se izvodi aplikacija. Njegova glavna zadaća je prihvatanje zahtjeva putem HTTP protokola. Poslužitelj šalje zahtjeve web aplikaciji na obradu i korisniku vraća odgovor.

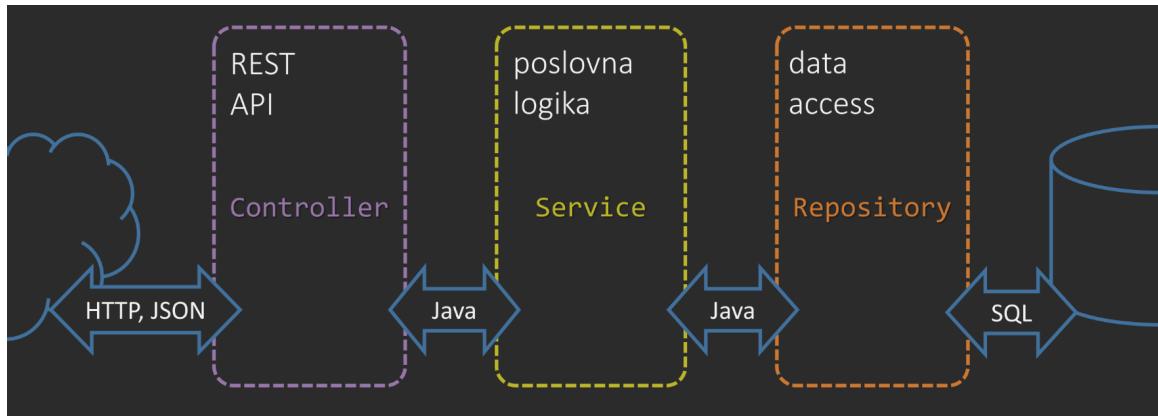
Web aplikacija je zasluzna za obradu korisničkih zahtjeva. Ima pristup bazi podataka. Odgovore šalje u obliku HTML dokumenta web poslužitelju koji ih prosljeđuje korisniku. Arhitektura aplikacije se može podijeliti na front-end i back-end. Cijeli sustav se poslužuje pomoću **digitalOcean-a**

Za izradu backend dijela naše aplikacije odabrali smo radni okvir Spring Boot. Programski jezik koji koristimo je Java.

Struktura toga dijela aplikacije biti će troslojna:

- **Controller** je sloj na kojem se nalazi naš REST API

- **Service** sloj upravlja poslovnom logikom naše aplikacije
- **Repository** sloj služi za pristup podatcima u bazi podataka



Slika 4.1: Prikaz troslojne strukture Spring Boot radnog okvira

Za izradu frontend dijela naše aplikacije odabrali smo Javascript library React. Uz React su nam potrebni i alat node.js te upravitelj paketa npm.

4.1 Baza podataka

Analizom zahtjeva utvrđeno je da aplikacija treba podržavati više korisnika koji istodobno mogu pregledavati, uređivati, stvarati i/ili brisati podatke bez da se pojavе nekonzistentnosti. Podatci koje je potrebno spremati, unaprijed su poznati i dobro definirani, a nisu prirodno strukturirani kao graf pa je relacijska baza podataka izabrana kao najbolje rješenje za pohranu podataka. Za bazu podataka smo koristili Postgresql.

4.1.1 Opis tablica

Entitet Račun sadrži informacije koje su iste za svakog korisnika i koje korisnik unosi prilikom registracije. Atributi računa su: ID računa koji se dodijeljuje automatski, email, lozinku, opis, vrsta korisnika (osnovni ili vlasnik obrta). Atribut zaključano je postavljen na false, a mijenja se na true ako korisnik više puta pogriješi lozinku. Omogućeno se postavlja na true ako je račun aktivan.

account (Račun)		
account_id	INT	identifikator
email	VARCHAR(50)	adresa e-pošte
password	VARCHAR(60)	enkriptirana lozinka
bio	VARCHAR(200)	biografija
user_role	VARCHAR(10)	razina autorizacije korisnika
locked	BOOLEAN	istina ako korisnik više puta unese pogršnu lozinku
enabled	BOOLEAN	istina ako je račun aktiviran

Nakon ispunjavanja registracijske forme korisnik na mail dobiva link kojim potvrđuje račun. Ta potvrda se spremi u tablicu Token za potvrdu računa. Tablica sadrži atribute ID tokena za potvrdu, token za potvrdu, datum kreiranja u kojima se bilježe podatci o tokenu te ID računa potreban da se poveže s računom koji se aktivira.

confirmation_token (Token za potvrdu računa)		
confirmation_token_id	INT	identifikator
confirmation_token	VARCHAR(36)	token za potvrdu računa
created_date	DATE	datum kreiranja tokena
account_id	INT	iden. računa za koji je vezan token

Tablica Korisnički račun sadrži atribute vezane uz osnovnog korisnika. Atribut ID računa posuđen je iz tablice račun uz koji se veže. Dodatni atributi su još korisničko ime, ime i prezime.

user_account (Korisnički račun)		
account_id	INT	posuđeni ključ
username	VARCHAR(30)	korisničko ime
first_name	VARCHAR(30)	ime
last_name	VARCHAR(30)	prezime

Poslovni račun je tablica za drugi tip korisnika. Također od entiteta račun posuđuje atribut ID računa, a uz njega još sadrži ime obrta, OIB, kontakt broj i ID tipa obrta.

business_account (Poslovni račun)		
account_id	INT	posuđeni ključ
business_name	VARCHAR(30)	ime obrta
oib	CHAR(11)	oib
phone_number	VARCHAR(15)	broj telefona
business_type_id	INT	ključ relacije bussinesType

Tablica lista obrta povezana je preko atributa ID tipa obrta s poslovnim računom i u nju se spremi tip obrta.

business_type (Vrsta obrta)		
business_type_id	INT	identifikator
business_type	VARCHAR(30)	vrsta obrta

Tablica Lokacija sadrži atribut ID lokacije koji nam je potreban za povezivanje ostalih entiteta vezanih uz lokaciju. Osim toga sadrži još atrbute za geografsku dužinu i širinu, adresu, ime lokacije i njezin opis, datum i vrijeme kreiranja. Atribut promovirana je postavljen na nulu osim ako se radi o promoviranoj lokaciji (obrt). Atribut "dog friendly" je boolean koji je true ako je lokacija prikladna za pse. U tablici se također spremi i ID računa korisnika koji ju je kreirao te se još postavlja ID tipa lokacije.

location (Lokacija)		
location_id	INT	identifikator
longitude	NUMERIC(8,5)	longituda
latitude	NUMERIC(8,5)	latituda
address	VARCHAR(50)	adresa
location_name	VARCHAR(30)	ime lokacije
location_description	VARCHAR(200)	opis lokacije
date_time_created	TIMESTAMP	vrijeme i datum objave lokacije
promoted	INT	nije nula ako je lokacija promovirana
dog_friendly	BOOLEAN	je li lokacija prikladna za pse
account_id	INT	identifikator korisnika koji je stvorio lokaciju
location_type_id	INT	identifikator vrste lokacije

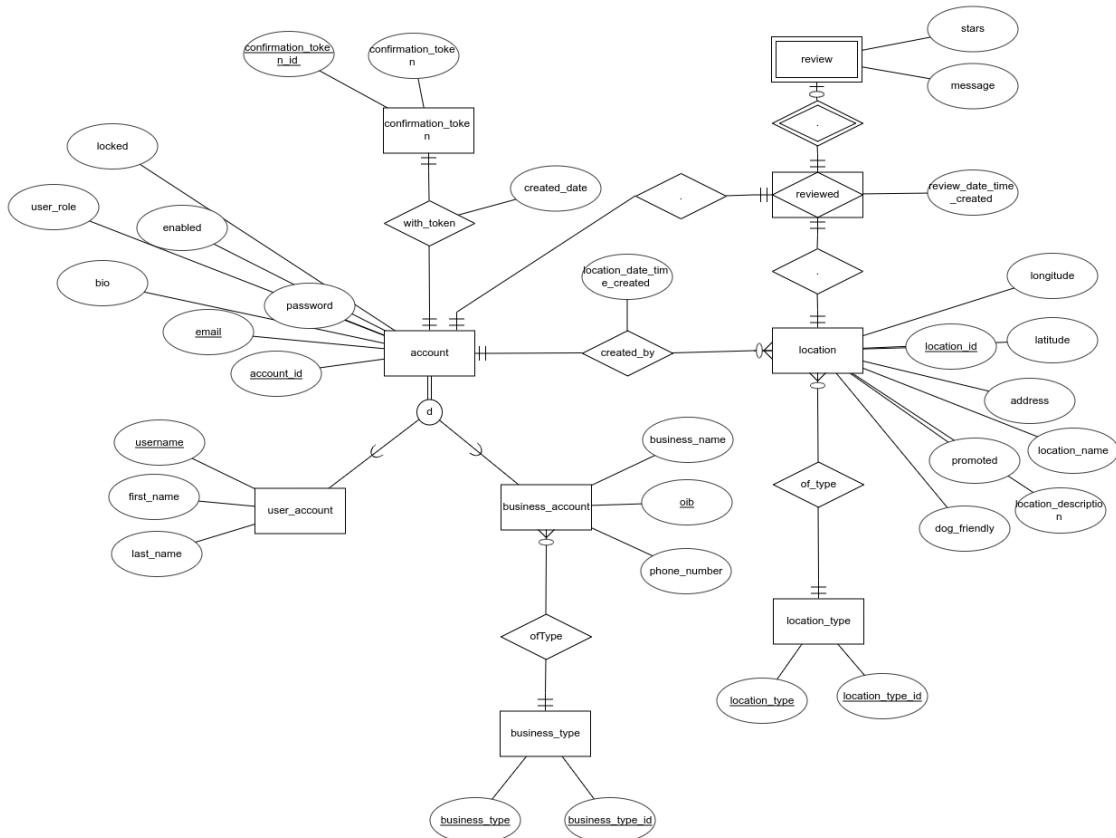
Tip lokacije je tablica koja se preko atributa ID tipa lokacije povezuje se pret-hodnom tablicom, a dodatno još ima atribut tip lokacije u koji se on upisuje.

location_type (Tip lokacije)		
location_type_id	INT	identifikator
location_type	VARCHAR(30)	vrsta lokacije

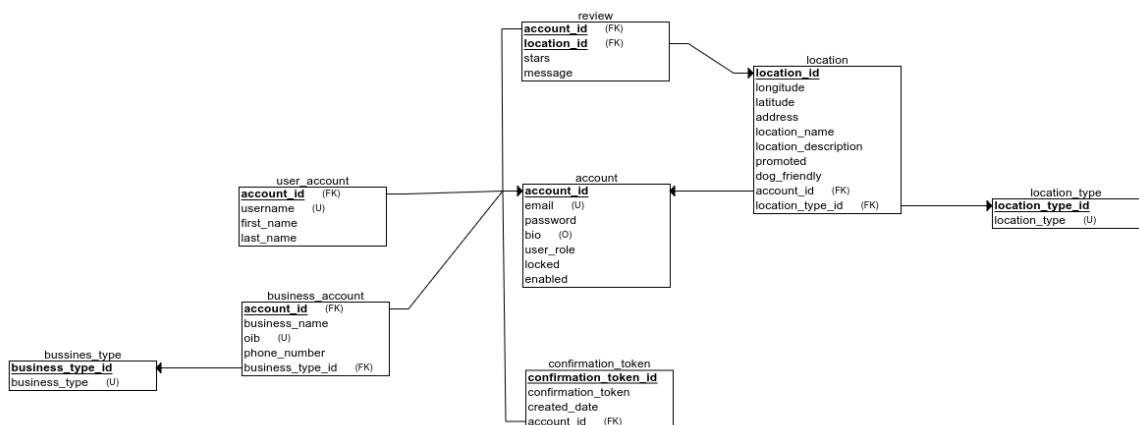
Posljednja tablica je Ocjena. U nju se spremaju atributi potrebi za prikaz ocjene određene lokacije. Preko atributa ID lokacije se povezuje s lokacijom na koju se odnosi, a ID računa zapisuje tko je postavio tu ocjenu. Ostali atributi su vrijeme i datum kreiranja, broj zvijezdica i poruka uz ocjenu.

review (Ocjena)		
location_id	INT	idn. lokacije za koju je napisana ocjena
account_id	VARCHAR(30)	idn. korinika koji je napisao ocjenu
date_time_created	TIMESTAMP	vrijeme i datum objave ocjene
stars	FLOAT	ocjena lokacije između 1 i 5
message	VARCHAR(200)	tekst ocjene

4.1.2 Dijagram baze podataka

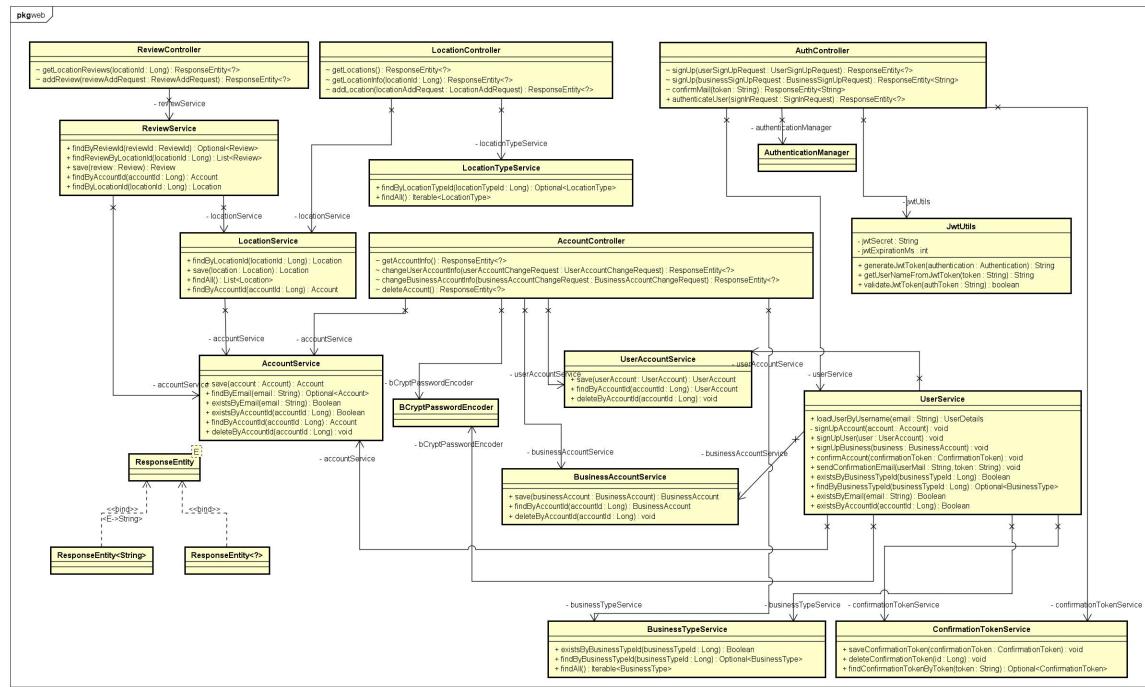


Slika 4.2: ER dijagram baze podataka

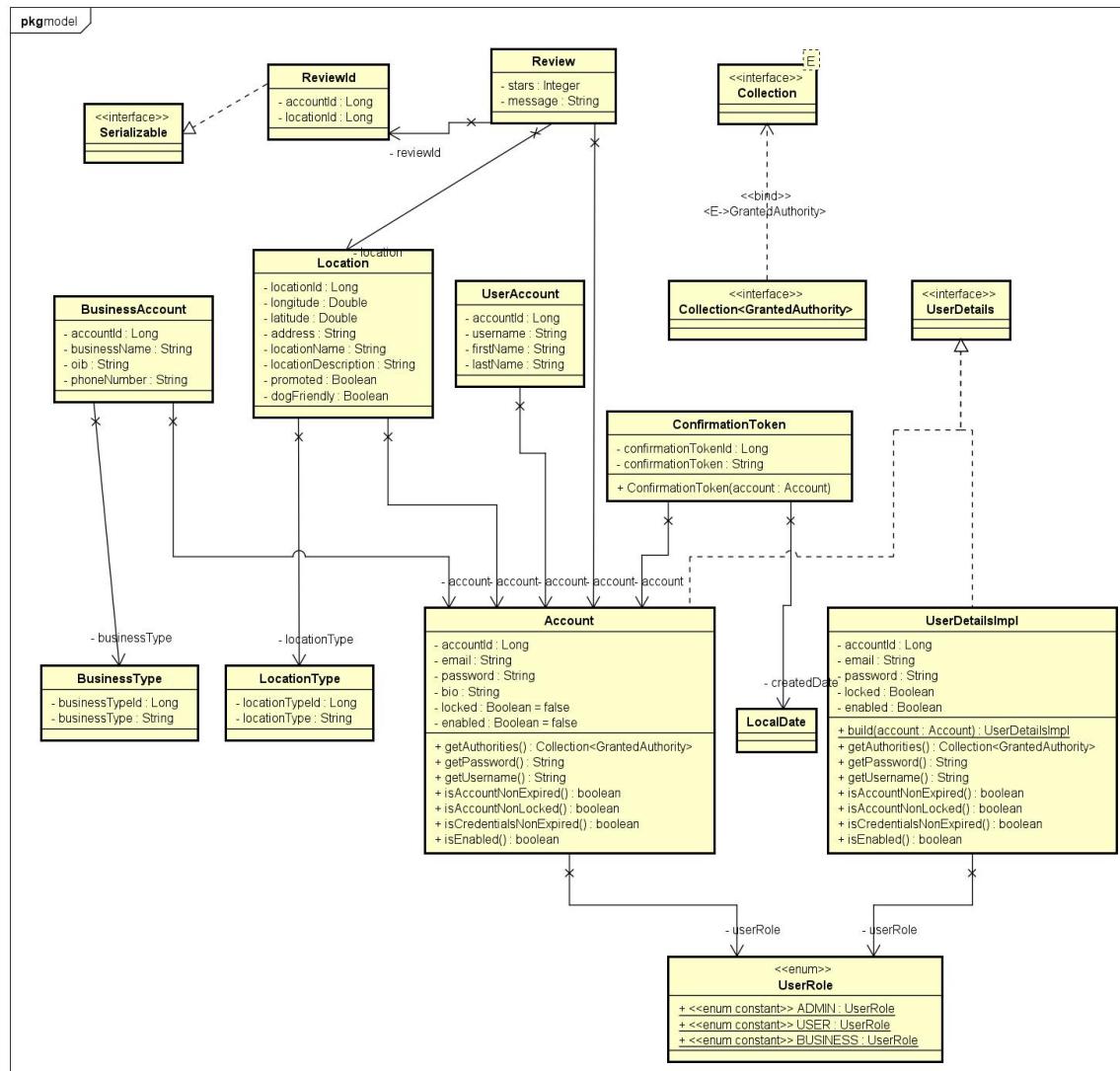


Slika 4.3: Relacijski dijagram baze podataka

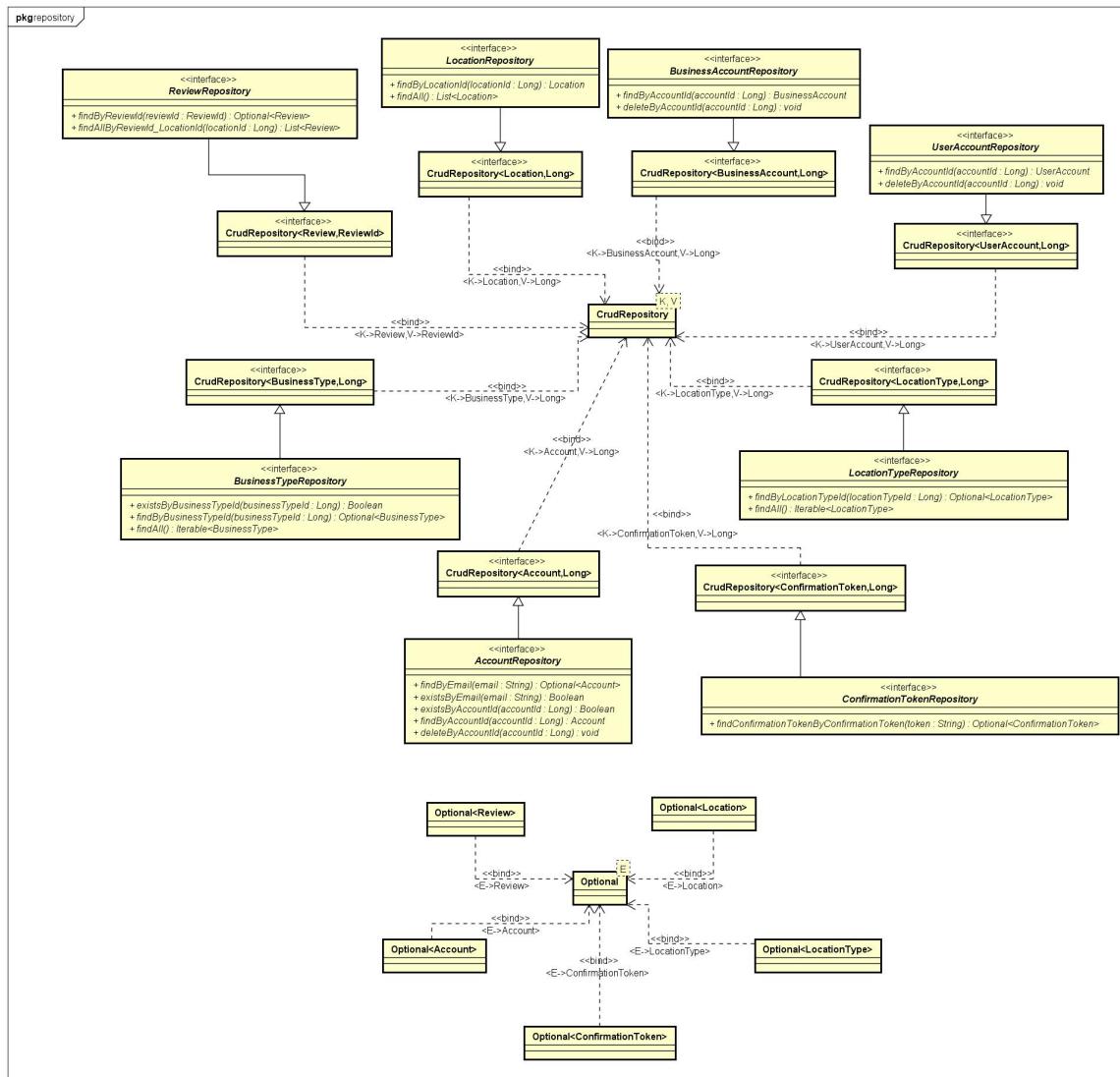
4.2 Dijagram razreda



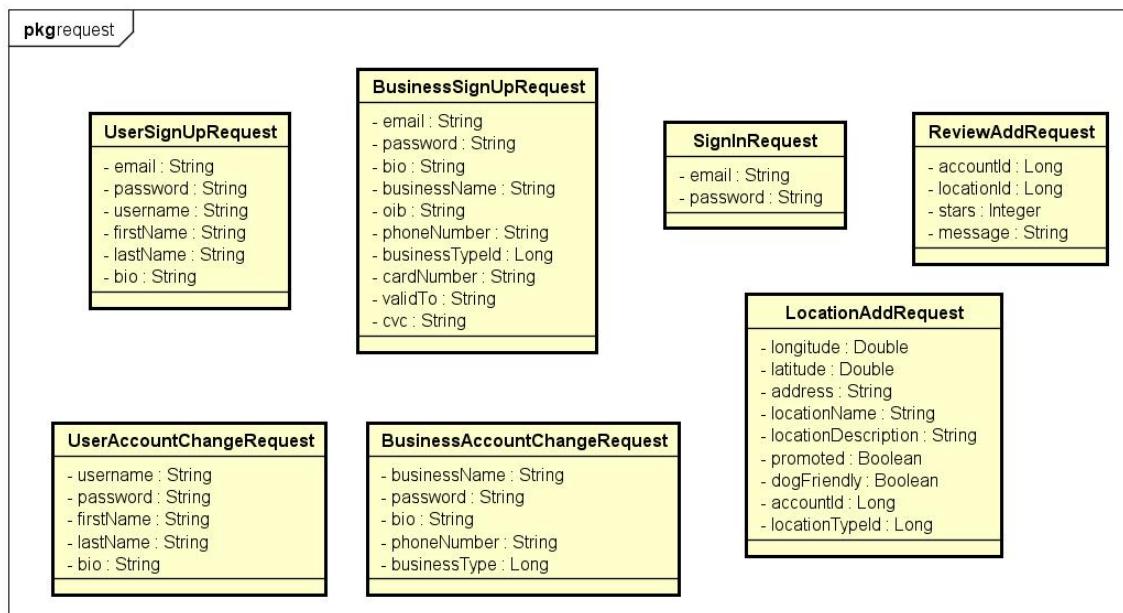
Slika 4.4: Controller class



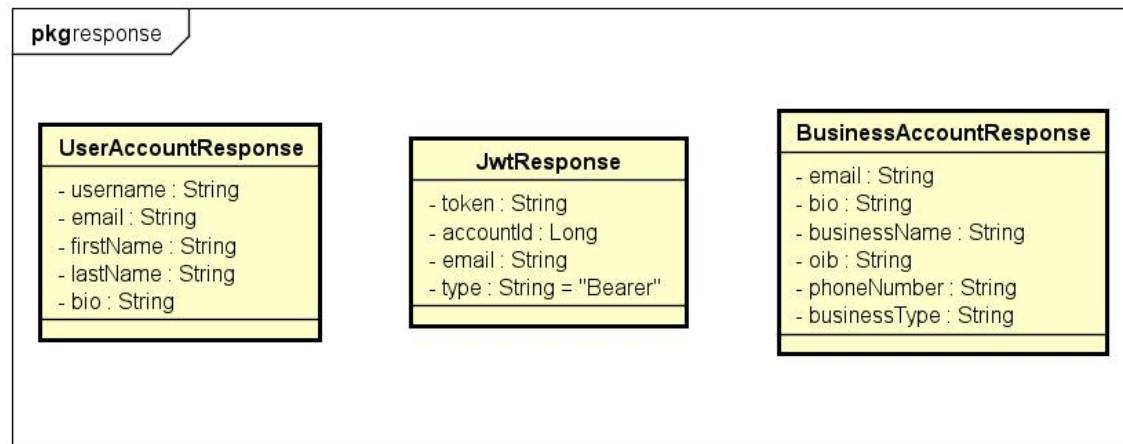
Slika 4.5: Model class



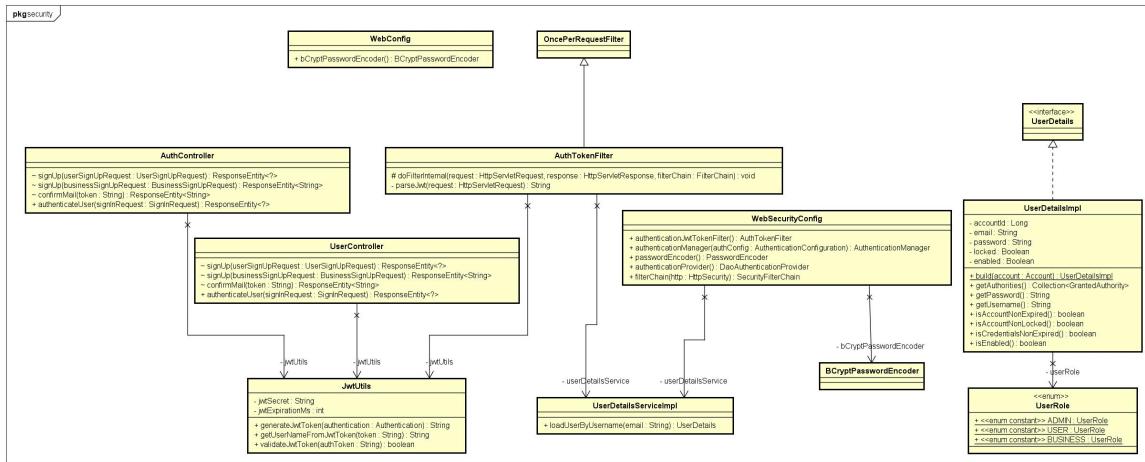
Slika 4.6: Repository class



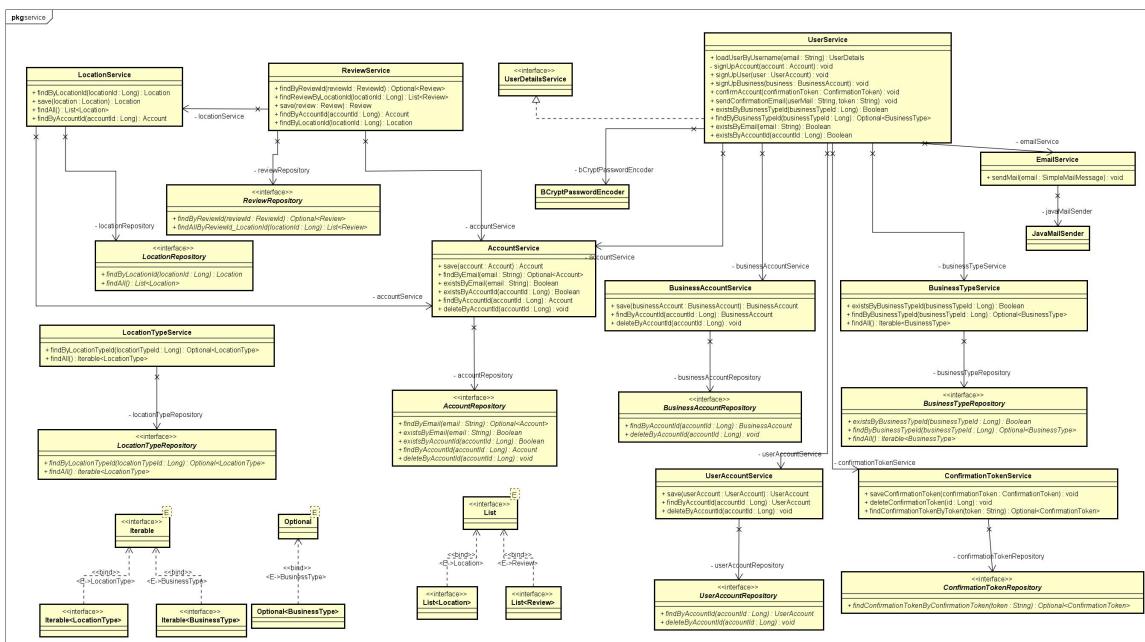
Slika 4.7: Request class



Slika 4.8: Response class

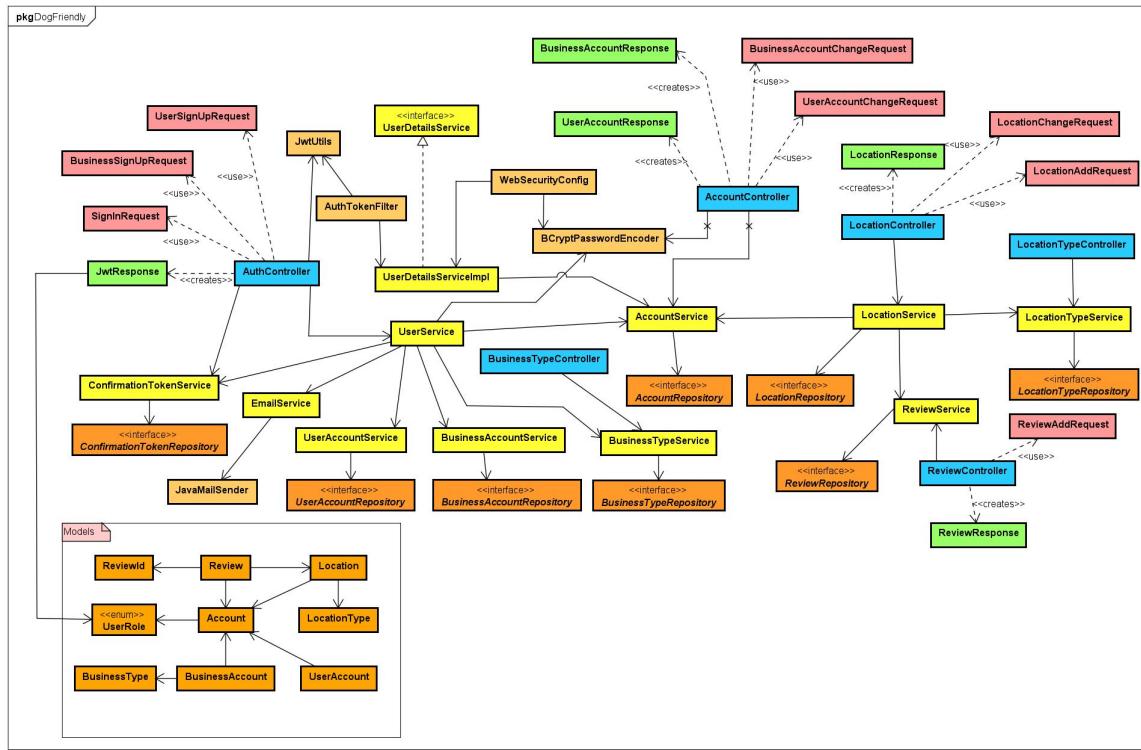


Slika 4.9: Security class



Slika 4.10: Service class

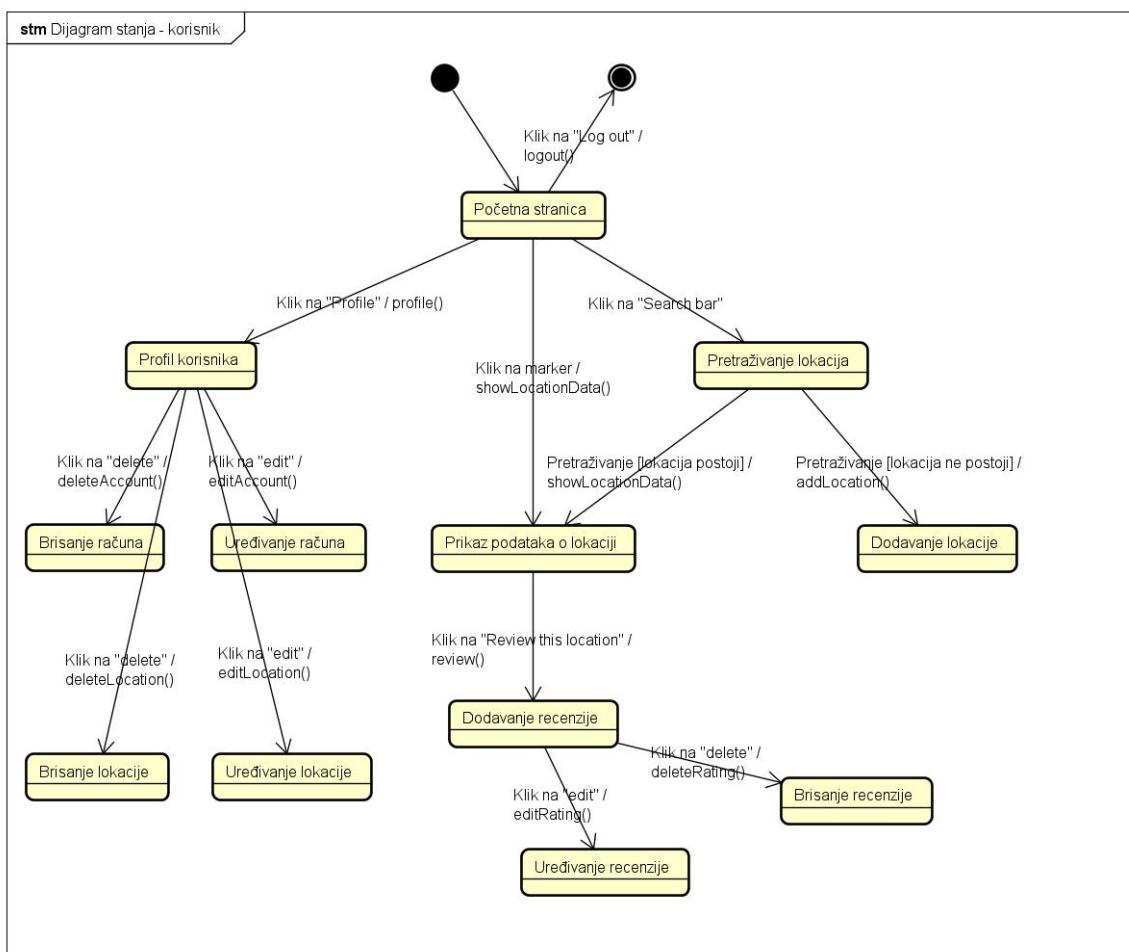
4.2.1 Dijagram razreda nakon završene aplikacije



Slika 4.11: Konceptualni dijagram razreda

4.3 Dijagram stanja

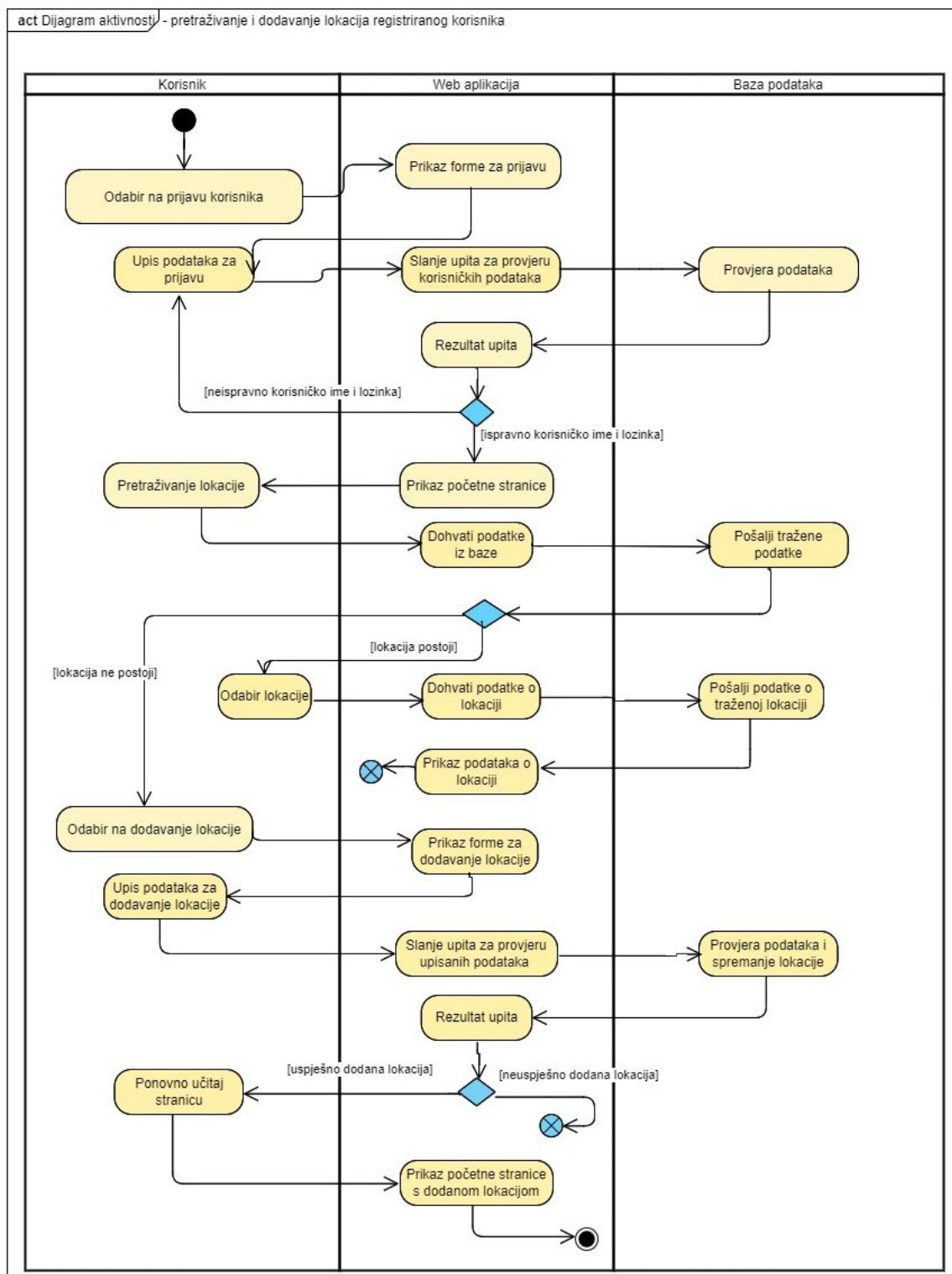
Na slici je prikazan dijagram stanja za prijavljenog korisnika. Korisniku se otvara početna stranica s kartom. U gornjem desnom kutu je gumb koji ga vodi na njegov profil. Na svom profilu korisniku je omogućeno mijenjanje podataka profila ili brisanje računa. Također, sve lokacije koje je korisnik dodaо su mu ponuđene na profilu da ih može mijenjati ili obrisati. Do podataka o lokaciji i pisanja recenzije može doći odabirom markera na karti ili pretraživanjem lokacije. Ako pretraživana lokacija ne postoji, korisniku se nudi opcija dodavanja lokacije.



Slika 4.12: Dijagram stanja

4.4 Dijagram aktivnosti

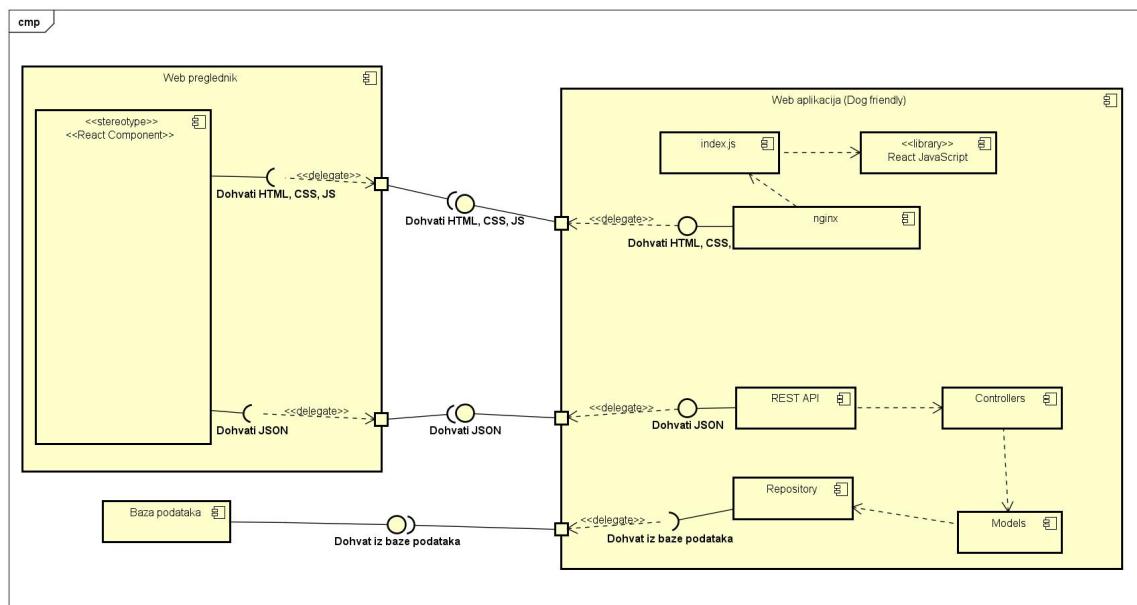
Dijagram aktivnosti primjenjuje se za modeliranje upravljačkog i podatkovnog toka. U modeliranju toka upravljanja koristi se "pull" način djelovanja tj. svaki novi korak poduzima se nakon završenog prethodnog. Cilj je jednostavno i sažeto prikazati tok kontrole između pojedinih interakcija. Na dijagramu aktivnosti prikazan je proces pretraživanja i dodavanja lokacija registriranog korisnika. Korisnik, nakon prijave u sustav, pretražuje željenu lokaciju. U slučaju da lokacija već postoji u sustavu, prikazuju mu se podatci o odabranoj lokaciji, u suprotnom, registrirani korisnik ima mogućnost dodati novu vlastitu lokaciju. Ispravnim unosom podataka i ponovnim učitavanjem stranice lokacija postaje vidljiva na karti.



Slika 4.13: Dijagram aktivnosti

4.5 Dijagram komponenti

Dijagram komponenti predstavlja specifikaciju arhitekture programske potpore i služi za vizualizaciju međuvisnosti i organizacije između implementacijskih komponenata. Sučeljem za dohvat HTML, CSS i JS datoteka se pristupa nginx komponenti koja poziva index.js. REST API komponenti se pristupa preko sučelja za dohvat JSON datoteka, a ona nam je potrebna za pristup bazi podataka.



Slika 4.14: Dijagram komponenti

5. Implementacija i korisničko sučelje

5.1 Korištene tehnologije i alati

Koristili smo open-source sustav za upravljanje i pohranjivanje izvornog koda na platformi **GitLab**. (<https://git-scm.com> , <https://gitlab.com>)

UML dijagrami izrađeni su uz pomoć **Astah Professional** aplikacije, a dijagrami baze podataka preko **ERDPlus**. ([https://astah.net/products astah-professional/](https://astah.net/products	astah-professional/) , <https://erdplus.com/>)

Za komunikaciju s asistentom korišten je **Microsoft Teams**, a za međusobnu komunikaciju **WhatsApp**. Popis i podjela zadataka napravljena je na **Google Sheets**. (<https://www.microsoft.com/hr-hr/microsoft-teams/group-chat-software/> , <https://web.whatsapp.com> , <https://www.google.com/sheets/about/>)

Za izradu dokumentacije primarno su korišteni **Overleaf** i **TeXstudio**. (<https://www.overleaf.com> , <https://www.texstudio.org>)

Za pisanje frontend dijela aplikacije korištena je JavaScript biblioteka **React** te sintaksna ekstenzija JSX. Kod smo pisali u **Visual Studio Code-u**. Za kartu je korišten **Mapbox API**. (<https://code.visualstudio.com> , <https://www.mapbox.com> , <https://www.npmjs.com> , <https://reactjs.org>)

Backend dio aplikacije pisan je u **IntelliJ IDEA** IDE-u u radnom okviru **Spring Boot** jezikom Java. (<https://www.jetbrains.com/idea/>)

Za deploy baze podataka (kao i aplikacije) korišten je **DigitalOcean**, a ista baza podataka korištena je i za testiranja. (<https://www.digitalocean.com/>)

5.2 Ispitivanje programskog rješenja

U nastavku ovog poglavlja biti će opisano ispitivanje implementiranih funkcionalnosti na razini komponenti te na razini cijelog sustava s prikazom odabralih ispitnih slučajeva.

U ispitivanju sustava korišten je radni okvir *Selenium* te programsko sučelje *Selenium WebDriver*. Metoda koja slijedi poziva se prije izvršavanja svakog testa.

```
1  @BeforeEach
2  public void init() {
3      driver = new ChromeDriver();
4      System.setProperty("webdriver.chrome.driver", "C:\\\\Program Files (
5          x86)\\\\Chrome Driver\\\\chromedriver.exe");
6
7      driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(10));
8      driver.get("http://dog-friendly.me/");
9  }
```

5.2.1 Ispitivanje komponenti

U prvom testu stvaramo objekt korisnika s već postojećim podatcima. Pozivamo funkciju koja se poziva prilikom pokušaja registracije. Test je uspješan ako test vrati vrijednost *true* koja nam govori da korisnik s tom mail adresom već postoji.

```
1  @Test
2  @Order(1)
3  public void testCreateExistingUserException() {
4      Account user = new Account(711, "lm53309@fer.hr", "Husky123",
5          "Novi du an", UserRole.BUSINESS, false, false);
6
7      accountService.existsByEmail(user.getEmail());
8
9      assertEquals(true, accountService.existsByEmail(user.getEmail()))
10 }
11 }
```

Drugi test provjerava funkciju promjene podataka lokacije. Želimo promijeniti samo jedan podatak (u ovom slučaju opis) dok ostali podatci moraju ostati isti. Test je Uspješan ako je objekt nakon promjene jednak spremljenoj lokaciji s novim podatkom.

```
1  @Test
2  @Order(2)
3  public void testLocationChange() {
4      LocationChangeRequest locationChangeRequest = new
5          LocationChangeRequest(21, null,
6              null, "Fakultet elektrotehnike i raunarstva", null,
7              null);
8      LocationResponse locationResponse = locationService.
9          getByLocationId(locationChangeRequest.getLocationId());
10     Location location = new Location(
11         locationResponse.getLocationId(),
12         locationResponse.getLongitude(),
13         locationResponse.getLatitude(),
14         locationResponse.getAddress(),
15         locationChangeRequest.getLocationName() != null ?
16             locationChangeRequest.getLocationName() : locationResponse.
17             getLocationName(),
18             locationChangeRequest.getLocationDescription() != null ?
19                 locationChangeRequest.getLocationDescription() : locationResponse.
20                 getLocationDescription(),
21                 locationChangeRequest.getPromoted() != null ?
22                     locationChangeRequest.getPromoted() : locationResponse.getPromoted(),
23                     locationChangeRequest.getDogFriendly() != null ?
24                         locationChangeRequest.getDogFriendly() : locationResponse.
25                         getDogFriendly(),
26                         accountService.findById(711),
27                         locationTypeService.findById(
28                             locationChangeRequest.getLocationTypeId() != null ?
29                                 locationChangeRequest.getLocationTypeId() : locationResponse.
30                                 getLocationTypeId()).orElse(null)
31             );
32
33     locationService.save(location);
34
35     assertEquals(location, locationService.save(location));
36 }
```

Treći test radi provjera funkcije za pretraživanje. Navodimo string lokacije koja postoji u sustavu. Test je uspješan ako je lokacija nađena funkcijom jednaka stringu.

```
1  @Test
2  @Order(3)
3  public void testSearchLocation() {
4      String description = "808";
5      List<LocationResponse> locations = locationService.search(
6          description);
7      LocationResponse location = locations.get(0);
8
9      assertEquals("808", location.getLocationName());
10 }
```

Četvrtim testom pokušavamo dohvatiti srednju vrijednost recenzije za lokaciju određenu s *locationId*. Test je uspješan ako je dohvaćena vrijednost jednaka očekivanoj.

```
1  @Test
2  @Order(4)
3  public void testReviewAverage() {
4      Long locationId = 21;
5      reviewService.getAverageForLocation(locationId);
6      double avg = (double) 13/3;
7
8      assertEquals(Double.valueOf(avg), reviewService.
9      getAverageForLocation(locationId));
10 }
```

Peti test provjerava funkciju za autentifikaciju korisnika. Korisnik se pokušava ulogirati na račun, ali s pogrešnom lozinkom. Ovaj test je uspješan ako kao odgovor dobijemo HttpStatus.BADREQUEST.

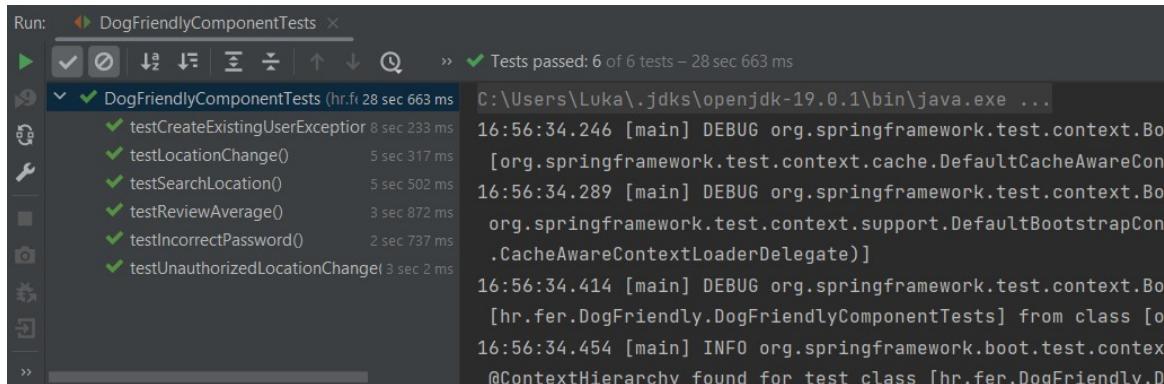
```
1  @Test
2  @Order(5)
3  public void testIncorrectPassword() {
4      String email = "1m53309@fer.hr";
5      String password = "provala";
6
7      Authentication authentication;
8      ResponseEntity responseEntity = new ResponseEntity(HttpStatus.OK
9  );
```

```
9
10     try {
11         authentication = authenticationManager.authenticate(
12             new UsernamePasswordAuthenticationToken(email,
13             password));
14         ResponseEntity = new ResponseEntity<>(HttpStatus.BAD_REQUEST
15     );
16
17         assertEquals(HttpStatus.BAD_REQUEST, ResponseEntity.
18         getStatusCode());
19 }
```

U posljednjem testu provjeravamo povezanost lokacije s korisnikom koji ju je kreirao. Korisnik pokušava promijeniti lokaciju koju on nije kreirao. Prilikom promjene prvo se provjerava ima li korisnik ovlasti za odabranu akciju. Test je uspješan ako kao odgovor dobijemo da korisnik nema dozvolu za promjenu (HttpStatus.UNAUTHORIZED).

```
1 @Test
2 @Order(6)
3 public void testUnauthorizedLocationChange() {
4     Long currentlyLoggedInAccountId = 711;
5     Long locationId = 1961;
6     LocationResponse locationResponse = locationService.
7     getByLocationId(locationId);
8     ResponseEntity responseEntity = new ResponseEntity(HttpStatus.OK
9 );
10
11     if(locationResponse.getAccountId() != currentlyLoggedInAccountId
12 )
13         responseEntity = new ResponseEntity<>(HttpStatus.
14 UNAUTHORIZED);
15
16     assertEquals(HttpStatus.UNAUTHORIZED, responseEntity.
17     getStatusCode());
18 }
```

Svi ispitani testovi su bili uspješni



Slika 5.1: Rezultati testova ispitivanja komponenti

5.2.2 Ispitivanje sustava

Prvi test ispituje uspješnu prijavu registriranog korisnika u sustav. Test je uspješan ako Selenium sustav dođe do *homepage-a*, tj. URL Web preglednika više ne sadrži */login*.

```
1  @Test
2  @Order(1)
3  void testUserLoginGoodCreds() {
4      driver.findElement(By.id("account-info")).click();
5      driver.findElement(By.id("login")).click();
6
7      WebElement element = driver.findElement(By.id("inputEmail"));
8      element.sendKeys("mario.hosnjak009@gmail.com");
9
10     element = driver.findElement(By.id("inputPassword"));
11     element.sendKeys("lozinka");
12
13     driver.findElement(By.className("button")).click();
14
15     try {
16         Thread.sleep(1000);
17     } catch (Exception e) {
18         e.printStackTrace();
19     }
20
21     String redirectedURL = driver.getCurrentUrl();
22     boolean result = !(redirectedURL.contains("login"));
23 }
```

```
24     assertTrue(result);  
25 }  
26
```

Drugi će test isprovocirati grešku u sustavu tako što će se sustav Selenium pokušati prijaviti s pogrešnim emailom i/ili lozinkom. Test će biti uspješan ako se pojavi poruka "Pogrešan e-mail ili lozinka".

```
1  @Test  
2  @Order(2)  
3  void testUserLoginBadCreds() {  
4      driver.findElement(By.id("account-info")).click();  
5      driver.findElement(By.id("login")).click();  
6  
7      WebElement element = driver.findElement(By.id("inputEmail"));  
8      element.sendKeys("krivi.email@gmail.com");  
9  
10     element = driver.findElement(By.id("inputPassword"));  
11     element.sendKeys("krivalozinka");  
12  
13     driver.findElement(By.className("button")).click();  
14  
15     try {  
16         Thread.sleep(1000);  
17     } catch (Exception e) {  
18         e.printStackTrace();  
19     }  
20  
21     String errorMsg = driver.findElement(By.cssSelector("div p")).  
getattribute("innerHTML");  
22  
23     assertTrue(errorMsg.contains("Pogrešan e-mail ili lozinka"));  
24 }
```

Treći test ispituje uspješno uređivanje podataka profila registriranog korisnika. Test se smatra uspješnim ako sustav Selenium uspije pročitati nove podatke sa stranice korisničkog profila.

```
1  @Test  
2  @Order(3)  
3  void testEditUserInfo() {  
4      testUserLoginGoodCreds();  
5      driver.findElement(By.id("account-info")).click();
```

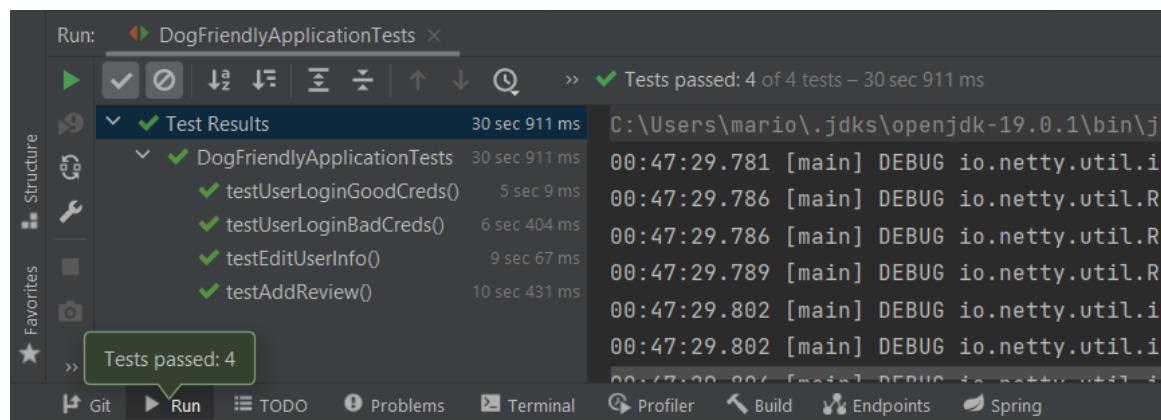
```
6     driver.findElement(By.id("profileInfo")).click();
7     driver.findElement(By.cssSelector("#editInfo > path:nth-child(2)")).click();
8
9     WebElement element = driver.findElement(By.id("editUsername"));
10    element.clear();
11    element.sendKeys("AutomatedTestUsername");
12
13    element = driver.findElement(By.id("editPassword"));
14    element.sendKeys("lozinka");
15
16    element = driver.findElement(By.id("editBio"));
17    element.clear();
18    element.sendKeys("AutomatedTestBio");
19
20    driver.findElement(By.id("saveChanges")).click();
21
22    try {
23        Thread.sleep(2000);
24    } catch (Exception e) {
25        e.printStackTrace();
26    }
27
28    String resElement1 = driver.findElement(By.name("username")).getAttribute("innerHTML");
29    String resElement2 = driver.findElement(By.name("bio")).getAttribute("innerHTML");
30
31    assertTrue(resElement1.contains("AutomatedTestUsername") &&
32    resElement2.contains("AutomatedTestBio"));
33 }
```

Četvrti test ispituje uspješno dodavanje recenzije na neku lokaciju. Test se smatra uspješnim ako sustav Selenium uspije pročitati novu recenziju.

```
1 @Test
2 @Order(4)
3 void testAddReview() {
4     testUserLoginGoodCreds();
5     driver.findElement(By.id("search")).click();
6
7     WebElement element = driver.findElement(By.id("search"));
8     element.sendKeys("Proba");
```

```
9
10    driver.findElement(By.id("0")).click();
11    driver.findElement(By.className("review-location-btn")).click();
12    driver.findElement(By.cssSelector("span:nth-child(4)")).click();
13
14    element = driver.findElement(By.cssSelector("textarea"));
15    element.sendKeys("Automated Test Review");
16
17    driver.findElement(By.id("submitReview")).click();
18
19    String reviewRes = driver.findElement(By.name("reviewText")).getAttribute("innerHTML");
20
21    assertTrue(reviewRes.contains("Automated Test Review"));
22}
23
```

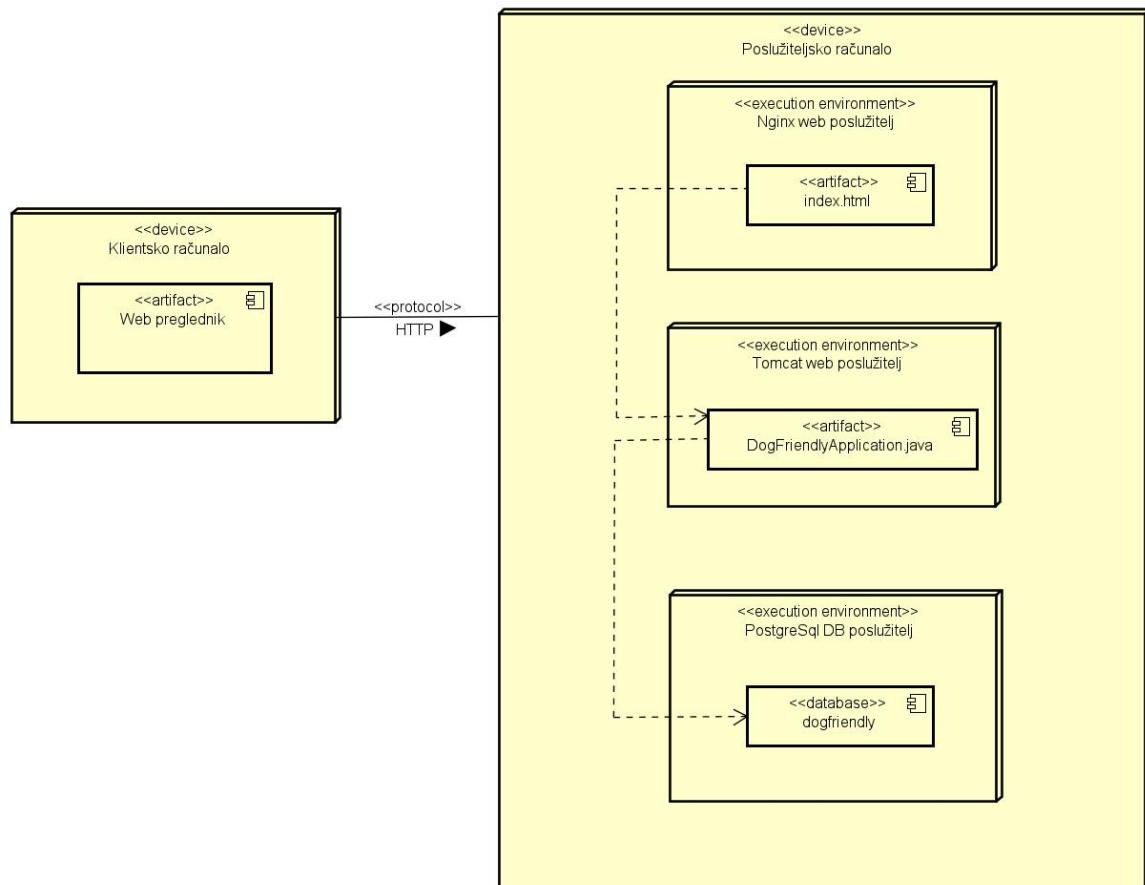
Sve ispitane funkcionalnosti su uspješno ispitane te svi testovi prolaze.



Slika 5.2: Rezultati testova ispitivanja sustava

5.3 Dijagram razmještaja

Dijagram razmještaja prikazuje odnos programskih i sklopoških dijelova te opisuje topologiju sustava. Klijent, koristeći HTTP protokol, preko klijentskog računala pristupa poslužiteljskom računalu. Poslužitelj koristi radni okvir Node.js preko kojeg se dohvata stranica, Tomcat koji sadrži aplikaciju i funkcionalnosti te PostgreSQL s bazom podataka.



Slika 5.3: Dijagram razmještaja

5.4 Upute za puštanje u pogon

Preduvjet za slijedenje sljedećih uputa je instalirana instanca Ubuntu 22.04 operacijskog sustava na poslužitelju. Koraci u ovim uputama će gotovo sigurno raditi bez modifikacija na ostalim modernim inačicama Ubuntu operacijskog sustava, a uz manje modifikacije i na ostalim Linux operacijskim sustavima.

5.4.1 Konfiguracija baze podataka

Instalacija i pokretanje postgresql servisa

```

1 sudo apt update
2 sudo apt install postgresql postgresql-contrib
3 sudo systemctl start postgresql.service
4

```

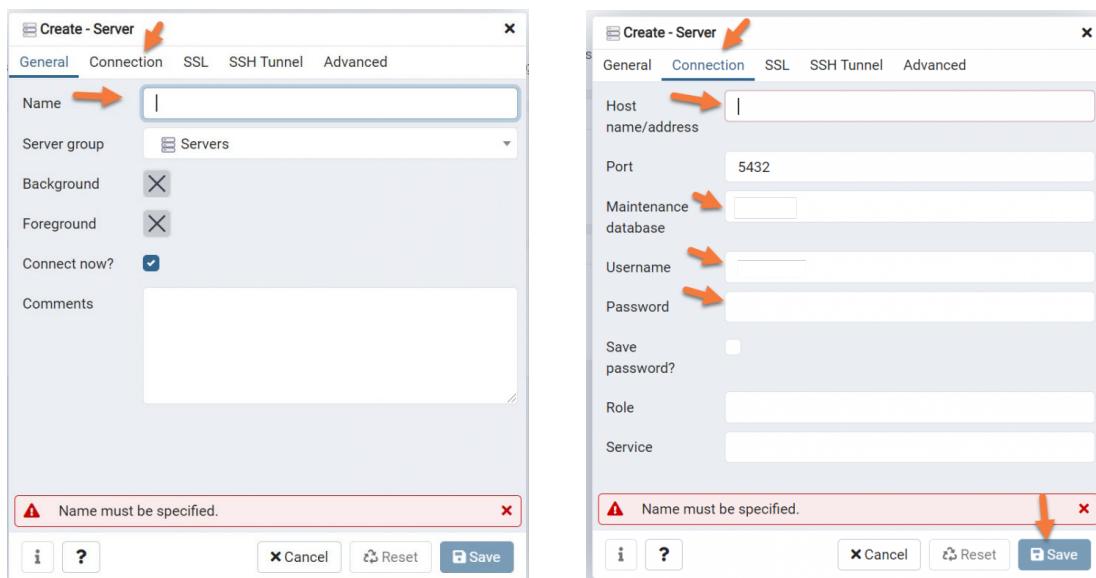
Zatim je potrebno konfigurirati vatrozid

```

1 sudo ufw allow 5432
2

```

Sada se na sustav za upravljanje bazom podataka moguće spojiti pomoću alata PgAdmin. Potrebno je kliknuti na gumb "Add New Server" i zatim popuniti polja prikazana na slici.



Slika 5.4: Spajanje na sustav za upravljanje bazom podataka

Desnim klikom na upravo povezan poslužitelj, koji se nalazi u lijevoj traci alata, dolazimo do opcije "Create Database". Bazi podataka je potrebno dodijeliti ime i

zatim je možemo pohraniti. Desnim klikom na novu bazu podataka dolazi se do opcije "Query Tool" koja otvara prozor s mogućnošću unosa nove SQL naredbe. Potrebno je kopirati sadržaj datoteke db.sql koja se nalazi unutar repozitorija te je zatim pokrenuti. Kopirane naredbe će stvoriti sve potrebne tablice, korisnike, uloge...

5.4.2 Backend poslužitelj

Prvi korak je izgradnja jar datoteke. Iz direktorija *IzvorniKod/backend* potrebno je pokrenuti naredbu

```
1 mvn package  
2
```

Ako sve prođe po planu, generiranu jar datoteku koja se nalazi u target direktoriju, potrebno je kopirati na poslužitelj. Zatim je na poslužitelju potrebno stvoriti novi servis s putanjom */etc/systemd/system/dogfriendly-backend.service* i u njega staviti sljedeći kod

```
1 [Unit]  
2 Description=Spring Boot backend for DogFriendly application  
3 After=network.target  
4 StartLimitIntervalSec=0  
5  
6 [Service]  
7 Type=simple  
8 Restart=always  
9 RestartSec=1  
10 User=root  
11 ExecStart=/usr/bin/java -jar /home/leon3428/backend/DogFriendly  
12 -0.0.1-SNAPSHOT.jar  
13  
14 [Install]  
15 WantedBy=multi-user.target
```

Novonastali sevis tada možemo pokrenuti s

```
1 sudo systemctl enable dogfriendly-backend.service  
2
```

5.4.3 Frontend poslužitelj

Poslužitelj nginx instaliramo s

```
1    sudo apt update  
2    sudo apt install nginx  
3
```

Nakon što instalacija završi, potrebno je omogućiti port

```
1    sudo ufw allow 'Nginx HTTP'  
2
```

Sljedeći korak je konfiguracija bloka za dogfriendly domenu

```
1    sudo mkdir -p /var/www/your_domain/html  
2    sudo chown -R $USER:$USER /var/www/your_domain/html  
3    sudo chmod -R 755 /var/www/your_domain  
4
```

Potrebno je kopirati sljedeću konfiguraciju u datoteku */etc/nginx/sites-available/your_domain*

```
1    server {  
2        listen 80;  
3        listen [::]:80;  
4  
5        root /var/www/dog-friendly.me/html;  
6        index index.html index.htm index.nginx-debian.html;  
7  
8        server_name dog-friendly.me www.dog-friendly.me;  
9        access_log /var/log/nginx/access.log;  
10  
11       location / {  
12           try_files $uri /index.html;  
13       }  
14       location /api {  
15           proxy_pass http://localhost:5000;  
16           proxy_http_version 1.1;  
17           proxy_set_header Upgrade $http_upgrade;  
18           proxy_set_header Connection 'upgrade';  
19           proxy_set_header Host $host;  
20           proxy_cache_bypass $http_upgrade;  
21       }  
22   }  
23  
24 }
```

Konfiguracijsku datoteku omogućimo s

```
1 sudo ln -s /etc/nginx/sites-available/your_domain /etc/nginx/
2 sites-enabled/
```

Poslužitelj bi sada trebao biti aktivran. Preostaje nam samo izgraditi React aplikaciju što možemo jednostavno napraviti s naredbom

```
1 npm run build
2
```

pozvanom iz direktorija *IzvorniKod/frontend*. Sadržaj build direktorija je sada potrebno kopirati u direktorij */var/www/your_domain/html* na poslužitelju.

6. Zaključak i budući rad

Projektni zadatak naše grupe bio je izrada aplikacije DogFriendly, web aplikacije koja bi vlasnicima pasa omogućila lakše pronalaženje željenih i potrebnih lokacija. Također, s obzirom da na aplikaciji postoje i promovirane lokacije vlasnicima obrta bi to služilo kao reklama.

Sam proces izrade projekta podijeljen je u 2 ciklusa. Prvi ciklus trajao je 8 tjedana. Započeo je s izradom početnog dijela dokumentacije, definiranjem temeljnih zadataka na osnovnoj razini, a potom detaljnijeg proučavanja. Nakon što je u dokumentaciji završena ideja i dogovorena arhitektura započeli smo s programiranjem. S obzirom da se većina tima prvi put susrela s tehnologijama koje smo odabrali koristiti, na početku je uloženo znatno vrijeme u učenje, istraživanje i grupno rješavanje određenih problema. Na kraju prvog ciklusa bio je završen potrebni dio dokumentacije s generičkim funkcionalnostima aplikacije (registracija i prijava), implementiranom kartom i osnovnim prikazom podataka korisnika.

U drugom ciklusu, u trajanju od 7 tjedana, bilo je potrebno ostvariti ostale funkcionalnosti aplikacije i dovršiti dokumentaciju. Većina backend dijela aplikacije je bila gotova tako da je fokus bio na funkcionalnostima u frontend dijelu i mapbox API-u koji smo koristili za prikaz karte i podataka na karti.

Na kraju smo uspjeli ostvariti sve funkcionalnosti koje su bile zadane u zadatku. U slučaju da se aplikacija ide učiniti javnom primarno bi se morao pronaći drugi server kako bi se sama aplikacija ubrzala i imala više radne memorije i memorije za spremanje. Također, morao bi se odabrati određeni servis koji bi regulirao spremanje kartičnih podataka obrta i samu naplatu za promociju.

Na ovom projektu smo svi stekli važna znanja i iskustvo. Naučili smo kako funkcionira timski rad na nekoj aplikaciji i kako se dijeli posao. Vjerujem da će nam stečeno zanje pomoći u dalnjim projektima i olakšati barem početak, koji nam je na ovom projektu bio vrlo velik izazov.

Popis literature

1. Programsко inžinjerstvo, FER, <https://www.fer.unizg.hr/predmet/proinz>
2. Astah Community, <http://astah.net/editions/uml-new>
3. The Unified Modeling Language, <https://www.uml-diagrams.org/>

Indeks slika i dijagrama

2.1 Prikaz mogućnosti koje nudi aplikacija DogPack	9
2.2 Prikaz karte	9
3.1 Funkcionalnost korisnika i vlasnika obrta	18
3.2 Funkcionalnost registriranog korisnika i vlasnika obrta	18
3.3 Sekvencijski dijagram registracije korisnika ili vlasnika obrta	19
3.4 Sekvencijski dijagram potvrđivanja ili negiranja postojećih lokacija	20
3.5 Sekvencijski dijagram označavanja novih lokacija	21
3.6 Sekvencijski dijagram pisanja recenzija i ocjenjivanja lokacija	22
4.1 Prikaz troslojne strukture Spring Boot radnog okvira	25
4.2 ER dijagram baze podataka	29
4.3 Relacijski dijagram baze podataka	30
4.4 Controller class	31
4.5 Model class	32
4.6 Repository class	33
4.7 Request class	34
4.8 Response class	34
4.9 Security class	35
4.10 Service class	35
4.11 Konceptualni dijagram razreda	36
4.12 Dijagram stanja	37
4.13 Dijagram aktivnosti	39
4.14 Dijagram komponenti	40
5.1 Rezultati testova ispitivanja komponenti	46
5.2 Rezultati testova ispitivanja sustava	49
5.3 Dijagram razmještaja	50
5.4 Spajanje na sustav za upravljanje bazom podataka	51
6.1 Dijagram pregleda promjena	63

Indeks tablica

1.0	3
6.0	61

Dodatak: Prikaz aktivnosti grupe

Dnevnik sastajanja

1. sastanak

- Datum: 20.10.2022.
- Prisustvovali: Svi
- Teme sastanka:
 - diskutiranje teme projekta
 - početni prijedlozi za način rada

2. sastanak

- Datum: 25.10.2022.
- Prisustvovali: Svi
- Teme sastanka:
 - dogovoren način rada i arhitektura
 - podijeljeni zadatci za razradu dokumentacije

3. sastanak

- Datum: 3.11.2022.
- Prisustvovali: Svi
- Teme sastanka:
 - dogovoren sadržaj baze podataka
 - napravljene izmjene u dokumentaciji (UC)

4. sastanak

- Datum: 16.11.2022.
- Prisustvovali: Svi
- Teme sastanka:
 - završna revizija dokumentacije
 - pregled i testiranje aplikacije
 - priprema za demonstraciju generičke funkcionalnosti

5. sastanak

- Datum: 06.12.2022.
- Prisustvovali: Svi
- Teme sastanka:
 - raspisivanje ostalih funkcionalnosti
 - podjela zadataka potrebnih za prezentiranje alfa inačice

6. sastanak

- Datum: 21.12.2022.
- Prisustvovali: Svi
- Teme sastanka:
 - prezentiranje alfa inačice
 - definiranje konačnog izgleda i funkcija

7. sastanak

- Datum: 05.01.2023.
- Prisustvovali: Svi
- Teme sastanka:
 - podjela zadataka za dokumentaciju
 - dogovor oko deploy-a i testiranja

	Leon Stjepan Uročić	Luka Marković	Filip Jakovina	Mario Hošnjak	David Winkler	Nela Štubelj	Zoa Horvat
Upravljanje projektom	2	0	0	0	0	0	0
Opis projektnog zadatka	0	2	4	0	0	0	0
Funkcionalni zahtjevi	0	0	0	0	0	3	0
Opis pojedinih obrazaca	0	0	0	3	0	0	3
Dijagram obrazaca	0	0	0	1	0	0	1
Sekvencijski dijagrami	0	0	0	3	0	0	0
Opis ostalih zahtjeva	0	0	0	1	0	0	0
Arhitektura i dizajn sustava	0	0	0	0	4	0	0
Baza podataka	5	0	0	0	0	0	0
Dijagram razreda	1	0	3	0	0	0	0
Dijagram stanja	0	2	0	0	0	0	0
Dijagram aktivnosti	0	0	0	0	0	2	0
Dijagram komponenti	0	2	0	0	0	0	0
Korištene tehnologije i alati	0	1	0	0	0	0	0
Ispitivanje programskog rješenja	0	4	2	4	0	0	0
Dijagram razmještaja	0	0	1	0	0	0	0
Upute za puštanje u pogon	3	0	0	0	0	0	0
Dnevnik sastajanja	0	1	0	0	0	0	0
Zaključak i budući rad	0	1	0	0	0	0	0
Popis literature	0	0.5	0	0	0	0	0

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

	Leon Stjepan Uroić		Luka Marković		Filip Jakovina		Mario Hošnjačak		David Winkler		Nela Štubelj		Zoa Horvat
<i>frontend</i>	20	40	3	40	30	12	35						
<i>backend</i>	40	0	50	2	0	0	0						
<i>izrada baze podataka</i>	4	0	0	0	0	0	0						

Dijagrami pregleda promjena



Slika 6.1: Dijagram pregleda promjena