

Programsko inženjerstvo

Ak. god. 2022./2023.

# Dog Friendly

Dokumentacija, Rev. 2

Grupa: *MontyPython*

Voditelj: *Fran Markulin*

Datum predaje: 13.1.2023.

Nastavnik: *Laura Majer*

# Sadržaj

<b>1 Dnevnik promjena dokumentacije</b>	<b>3</b>
<b>2 Opis projektnog zadatka</b>	<b>5</b>
<b>3 Specifikacija programske potpore</b>	<b>10</b>
3.1 Funkcionalni zahtjevi . . . . .	10
3.1.1 Obrasci uporabe . . . . .	11
3.1.2 Sekvencijski dijagrami . . . . .	19
3.2 Ostali zahtjevi . . . . .	22
<b>4 Arhitektura i dizajn sustava</b>	<b>23</b>
4.1 Baza podataka . . . . .	25
4.1.1 Opis tablica . . . . .	25
4.1.2 Dijagram baze podataka . . . . .	27
4.2 Dijagram razreda . . . . .	28
4.3 Dijagram stanja . . . . .	33
4.4 Dijagram aktivnosti . . . . .	33
4.5 Dijagram komponenti . . . . .	34
<b>5 Implementacija i korisničko sučelje</b>	<b>36</b>
5.1 Korištene tehnologije i alati . . . . .	36
5.2 Ispitivanje programskog rješenja . . . . .	37
5.2.1 Ispitivanje komponenti . . . . .	37
5.2.2 Ispitivanje sustava . . . . .	42
5.3 Dijagram razmještaja . . . . .	46
5.4 Upute za puštanje u pogon . . . . .	47
<b>6 Zaključak i budući rad</b>	<b>48</b>
<b>Popis literature</b>	<b>49</b>
<b>Indeks slika i dijagonama</b>	<b>51</b>



# 1. Dnevnik promjena dokumentacije

Rev.	Opis promjene/dodatka	Autori	Datum
0.1	Napravljen predložak.	Jura Starčević	5.11.2022.
0.2	Dodan opis	Jura Starčević	6.11.2022.
0.3	Dodano i djelomično uređeno poglavlje 3.1	Marko Štrk	8.11.2022.
0.4	Početak dodavanja informacija o bazi	Jura Starčević	15.11.2022.
0.5	Dodan djelomičan dijagram obrazaca uporabe	Karla Udiljak	15.11.2022.
0.6	Dodani i razrađeni Use Caseovi	Marko Štrk	16.11.2022.
0.7	Dodavanje arhitekture i opisa baze podataka	Fran Markulin	16.11.2022.
0.8	Uređivanje opisa i arhitekture	Jura Starčević	16.11.2022.
0.9	Dodavanje potpunog dijagrama obrazaca uporabe, sekvencijskih dijagrama te njihovih opisa	Karla Udiljak	16.11.2022.
0.10	Dodani Ostali zahtjevi i UML dijagram	Marko Štrk	17.11.2022.

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

<b>Rev.</b>	<b>Opis promjene/dodataka</b>	<b>Autori</b>	<b>Datum</b>
<b>1.0</b>	Finalna verzija za prvi ciklus	Jura Starčević	18.11.2022.
1.1	Dodano poglavlje 6. - Zaključak i budući rad	Marko Štrk	21.12.2022.
1.2	Dodano i uređeno poglavlje 5.2.2 i 5.4 Upute za puštanje u pogon	Marko Štrk	8.1.2023.
1.3	Uređen opis baze	Jura Starčević	12.1.2023.
1.4	Dodani opisi dijagrama	Marko Štrk	13.1.2023.
<b>2.0</b>	Finalna verzija za drugi ciklus	Jura Starčević	13.1.2022.

## 2. Opis projektnog zadatka

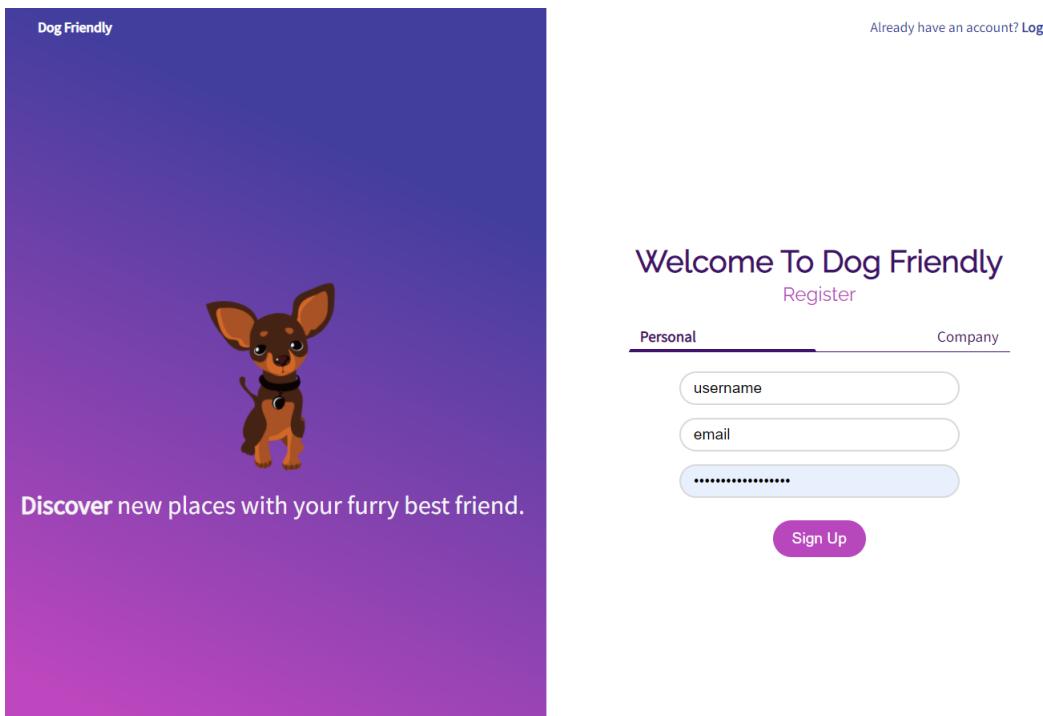
Cilj ovog projekta je razviti programsku podršku za stvaranje web aplikacije Dog Friendly koja će korisniku pomoći pronaći lokacije koje su prikladne i doštojne čovjekovog najboljeg prijatelja. Ova aplikacija će vlasnicima i ljubiteljima pasa omogućiti pregled prikladnih i neprikladnih lokacija na interaktivnoj karti i time olakšati druženje, igru i odmor. Aplikacija pruža i dodatak novih lokacija na interaktivnu kartu te za svaku novu lokaciju potrebno je dodatni nekoliko ključnih informacija poput imena lokacije, rating i kategorija( parkić, plaža, dućan, kafić, restoran, veterinarska ambulanta, frizerski salon, itd.). Ako pritisnemo na indikator lokacije prikazat će nam se opis lokacije i kontakt. Korisnike ove aplikacije dijelimo na tri vrste:

- neregistrirani korisnici
- registrirani korisnici
- vlasnici obrta

Neregistriranim korisnicima pružene su osnovne funkcionalnosti aplikacije(pregled interaktivne karte).

Registrirani korisnici su korisnici koji su prošli kroz proces registracije koji se sastoji od 2 koraka. U prvom koraku korisnik ispunjava formu s:

- korisničkim imenom
- e-mail adresom
- lozinkom



Slika 2.1: Stranica za registraciju

Nakon ispunjavanja korisnik na svoju e-mail adresu prima obavijest o registraciji i traži ga se da potvrdi svoju registraciju. Nakon potvrde proces registracije je završen. Ako korisnik zaboravi svoju lozinku ili je dobio ideju za bolje korisničko ime pruža mu se promjena oba korisnička podatka. Registriranim korisnicima pružene su sve funkcionalnosti aplikacije koje imaju i neregistrirani korisnici uz priliku dodavanja novih lokacija na kartu. Osim toga, za već postojeće lokacije korisnik može potvrditi ili negirati njenu oznaku.

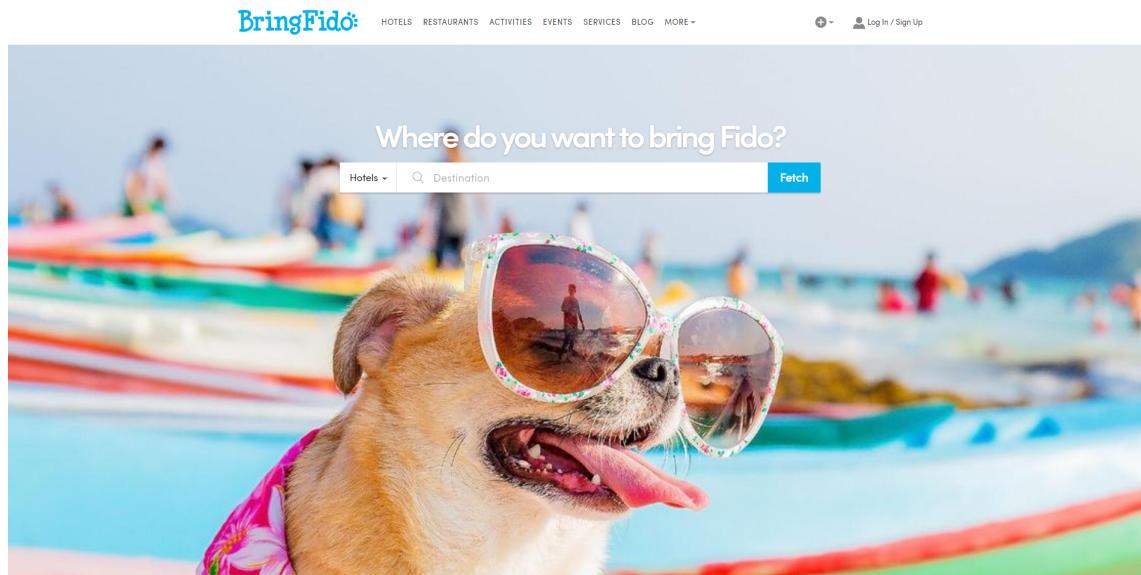
Vlasnici obrta moraju unijeti kroz dodatnu formu dodatne podatke:

- naziv
- adresu
- OIB obrta
- kontakt broj
- kratki opis
- kartični podatci

Financiranje aplikacije je zamišljeno putem reklama koje će vlasnici obrta moći izdati za određenu novčanu naknadu. Potvrdu o svakoj uplati korisnik će dobiti na svoju e-mail adresu.

Najpopularnije postojeće rješenje je BringFido u obliku web aplikacije i mobilne

aplikacije dostupne u AppStoreu i na Google Playu. Na početnoj stranici odmah su ponuđene kategorije i tražilica koja nas dalje upućuje na poželjne informacije.



Slika 2.2: Početna stranica

Nove lokacije mogu dodati samo registrirani korisnici. Za lokaciju se prvo odbire kategorija kojoj pripada a zatim se ispunjavaju osnovne informacije o lokaciji. Za razliku od Dog Friendly na kraju se forma preda na pregled radnicima BringFidoa te ako se odobri bit će objavljena na njihovoј aplikaciji što znatno komplikira i usporava proces dodavanja lokacija no povećava kvalitetu sadržaja karte.

Step 1 of 4 - The basics

**1 Business Type**

**2 Name and Location**  
Start typing to look up the business by name or address. We'll try to pull most of the contact information from Google.

Name

Address

City

Postal code

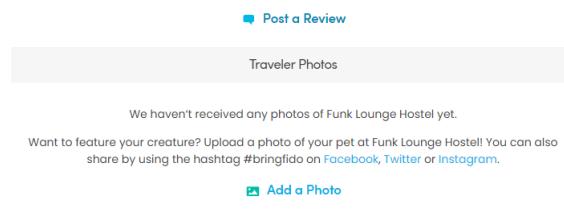
Phone number

**Save & Continue**

© 2005-2022 Kendall Media, Inc. [Privacy Policy & Advertising Disclosure](#) [Terms & Conditions](#) [Sitemap](#)

Slika 2.3: Primjer unosa novog obrta

Korisnici mogu ostaviti svoje komentare i iskustva o svakoj lokaciji, ali naravno i ocijeniti lokaciju.



Slika 2.4: Dodavanje osvrta

Još jedna značajna razlika je u sadržaju i jednostavnosti. BringFido sadrži previše nepotrebnih sadržaja koji nisu potrebni korisniku prilikom pretrage lokacija poput blogova vezanih uz kućne ljubimce.

**BringFido:**

HOTELS RESTAURANTS ACTIVITIES EVENTS SERVICES BLOG MORE ▾

Log In / Sign Up

## LATEST FROM THE BLOG



BringFido's Deal of the Week



Can I Bring My Dog to Red Lion Hotels?



New Pet-Friendly Hotels: November 2022



Brewery Dogs Across America



Advent Calendar for Dogs



Restaurants with Gourmet Dog Menus

Slika 2.5: Prikaz nepotrebnih sadržaja

# 3. Specifikacija programske potpore

## 3.1 Funkcionalni zahtjevi

Dionici:

1. Vlasnik (naručitelj)
2. Vlasnik psa
3. Vlasnik obrta
4. Razvojni tim

Aktori i njihovi funkcionalni zahtjevi:

1. Neregistrirani/neprijavljeni korisnik može:
  - (a) pregledati lokacije na karti
  - (b) odabrati lokaciju i dobiti prikaz opcih informacija (ime lokacije, adresa, telefon, OIB, kratak opis, djelatnost)
  - (c) se registrirati u sustav, stvoriti novi korisnicki račun za koji su mu potrebni korisničko ime, lozinka, ime, prezime, broj mobitela, e-mail adresa
2. Registirani korisnik može:
  - (a) pregledavati i mijenjati osobne podatke
  - (b) izbrisati svoj korisnički račun
  - (c) pisati recenzije i dati ocjene
  - (d) pregledati recenzije
3. Baza podataka(sudionik):
  - (a) pohranjuje sve podatke o korisnicima i njihovim ovlastima
  - (b) pohranjuje sve podatke o lokacijama

### 3.1.1 Obrasci uporabe

#### UC1 - Pregled lokacija na karti

- **Glavni sudionik:** Registrirani korisnik, Neregistrirani korisnik
- **Cilj:** Pregledati lokacije i osnovne informacije
- **Sudionici:** Baza podataka, Google Maps API
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
  1. Karta je prikazana prilikom učitavanja aplikacije
  2. Korisnik na karti odabire lokaciju
  3. Prikazuju se informacije o lokaciji i ponudi

#### UC2 - Registracija obrta

- **Glavni sudionik:** Neregistrirani korisnik
- **Cilj:** Stvoriti korisnički račun za pristup sustavu
- **Sudionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
  1. Korisnik odabire opciju za registraciju
  2. Korisnik odabire opciju "Company"
  3. Korisnik unosi podatke
  4. Korisnik prima e-mail za potvrdu registracije
  5. Korisnik potvrđuje registraciju
  6. Prikazuje se poruka o uspješnoj registraciji
- **Opis mogućih odstupanja:**
  - 2.a Odabir već zauzetog korisničkog imena i/ili e-maila, unos korisničkih podataka u nedozvoljenom formatu ili pružanje neispravnog e-maila, ne zadovoljavanje kompleksnosti lozinke
    1. Sustav obavještava korisnika o neuspješnom upisu i vraća ga na stranicu za registraciju
    2. Korisnik mijenja podatke te završava unos ili odustaje od registracije

#### UC3 - Registracija privatne osobe

- **Glavni sudionik:** Neregistrirani korisnik
- **Cilj:** Stvoriti korisnički račun za pristup sustavu
- **Sudionici:** Baza podataka

- **Preduvjet:** -
- **Opis osnovnog tijeka:**
  1. Korisnik odabire opciju za registraciju
  2. Korisnik odabire opciju "personal"
  3. Korisnik unosi podatke
  4. Korisnik prima e-mail za potvrdu registracije
  5. Korisnik potvrđuje registraciju
  6. Prikazuje se poruka o uspješnoj registraciji
- **Opis mogućih odstupanja:**
  - 2.a Odabir već zauzetog korisničkog imena i/ili e-maila, unos korisničkih podataka u nedozvoljenom formatu ili pružanje neispravnog e-maila, ne zadovoljavanje kompleksnosti lozinke
    1. Sustav obaveštava korisnika o neuspjelom upisu i vraća ga na stranicu za registraciju
    2. Korisnik mijenja podatke te završava unos ili odustaje od registracije

#### **UC4 - Prijava u sustav**

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

#### **U5 - Pregled osobnih podataka**

- **Glavni sudionik:** Registrirani korisnik
- **Cilj:** Pregledati osobne podatke
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen
- **Opis osnovnog tijeka:**

1. Korisnik odabire opciju "Osobni podatci"
2. Aplikacija prikazuje osobne podatke korisnika

### UC6 - Promjena osobnih podataka

- **Glavni sudionik:** Registrirani korisnik
- **Cilj:** Promijeniti osobne podatke
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen
- **Opis osnovnog tijeka:**
  1. Korisnik odabire opciju za promjenu podataka
  2. Korisnik mijenja svoje osobne podatke
  3. Korisnik spremi promjene
  4. Baza podataka se ažurira
- **Opis mogućih odstupanja:**
  - 2.a Korisnik promjeni svoje osobne podatke, ali ne odabere opciju "Spremi promjenu"
    1. Sustav obavještava korisnika da nije spremio podatke prije izlaska iz prozora
  - 2.b Odabir već zauzetog korisničkog imena i/ili e-maila, unos korisničkog podatka u nedozvoljenom formatu ili pružanje neispravnog e-maila, ne zadovoljavanje kompleksnosti lozinke, promijenjena lozinka je jednaka trenutnoj
    1. Sustav obavještava korisnika o neuspjelom upisu i vraća ga na stranicu za promjenu podataka
    2. Korisnik mijenja podatke te završava unos ili odustaje od promjene podataka

### UC7 - Brisanje korisničkog računa

- **Glavni sudionik:** Registrirani korisnik
- **Cilj:** Izbrisati svoj korisnički račun
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen
- **Opis osnovnog tijeka:**
  1. Otvara se stranica s osobnim podacima korisnika
  2. Korisnik briše račun
  3. Korisnički račun se izbriše iz baze podataka

4. Otvara se stranica za registraciju

#### **UC8 - Pregled informacija o lokaciji**

- **Glavni sudionik:** Registrirani/Neregistrirani korisnik
- **Cilj:** Pregledati detaljnije informacije o lokaciji
- **Sudionici:** Baza podataka, Google Maps API
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
  1. Korisnik odabire određenu lokaciju
  2. Prikazuju se detaljne informacije o lokaciji

#### **UC9 - Pregled informacija o obrtu**

- **Glavni sudionik:** Registrirani/Neregistrirani korisnik
- **Cilj:** Pregledati detaljnije informacije o obrtu
- **Sudionici:** Baza podataka, Google Maps API
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
  1. Korisnik odabire određeni obrt
  2. Prikazuju se detaljne informacije o tom obrtu

#### **UC10 - Filtriranje lokacije prema kategoriji**

- **Glavni sudionik:** Registrirani/Neregistrirani korisnik
- **Cilj:** Filtrirati prikazane lokacije
- **Sudionici:** Baza podataka, Google Maps API
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
  1. Karta je prikazana prilikom učitavanja aplikacije
  2. Korisnik odabire željene kategorije
  3. Prikazuju se samo lokacije koje odgovaraju unesenim kategorijama

#### **UC11 - Filtriranje lokacije prema unosu u polje za pretraživanje**

- **Glavni sudionik:** Registrirani/Neregistrirani korisnik
- **Cilj:** Filtrirati prikazane lokacije
- **Sudionici:** Baza podataka, Google Maps API
- **Preduvjet:** -
- **Opis osnovnog tijeka:**

1. Karta je prikazana prilikom učitavanja aplikacije
2. Korisnik unosi željeni tekst u polje za pretraživanje
3. Prikazuju se samo lokacije koje odgovaraju unesenom tekstu

#### UC12 - Dohvaćanje lokacije uređaja

- **Glavni sudionik:** Registrirani/Neregistrirani korisnik
- **Cilj:** Prikazati točnu lokaciju korisnika
- **Sudionici:** Google Maps API
- **Preduvjet:** Prihvaćena privola za dohvaćanje lokacije uređaja
- **Opis osnovnog tijeka:**
  1. Karta je prikazana prilikom učitavanja aplikacije
  2. Traži se privola korisnika
  3. Prikazuje se točna lokacija uređaja
- **Opis mogućih odstupanja:**
  - 2.a Korisnik ne daje privolu
    1. Ne prikazuje se korisnikova lokacija

#### UC13 - Privola za dohvaćanje lokacije uređaja

- **Glavni sudionik:** Registrirani/Neregistrirani korisnik
- **Cilj:** Dobiti od korisnika dozvolu za lociranje
- **Sudionici:** Google Maps API
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
  1. Prikazuje se prozor u kojem se od korisnika traži dopuštenje
  2. Korisnik daje dopuštenje
  3. Prikazuje se točna lokacija uređaja
- **Opis mogućih odstupanja:**
  - 2.a Korisnik ne daje privolu
    1. Ne prikazuje se korisnikova lokacija

#### UC14 - Dodavanje lokacija

- **Glavni sudionik:** Registrirani korisnik
- **Cilj:** Dodati lokaciju u sustav
- **Sudionici:** Baza podataka, Google Maps API
- **Preduvjet:** Korisnik je prijavljen
- **Opis osnovnog tijeka:**

1. Korisnik odabire opciju "Dodaj lokaciju"
  2. Korisnik unosi podatke o lokaciji
  3. Podatci o lokaciji se spremaju u bazu podataka
  4. Lokacija se prikazuje na karti
- **Opis mogućih odstupanja:**
    - 2.a Korisnik unosi podatke u nedozvoljenom formatu
      1. Sustav obavještava korisnika o pogrešci

#### UC15 - Ocjenjivanje prikladnosti lokacija

- **Glavni sudionik:** Registrirani korisnik
- **Cilj:** Ocijeniti je li lokacija prikladna za pse ili nije
- **Sudionici:** Baza podataka, Google Maps API
- **Preduvjet:** Lokacija postoji u bazi podataka
- **Opis osnovnog tijeka:**
  1. Korisnik odabire opciju "Ocijeni lokaciju"
  2. Korisnik daje svoju ocjenu
  3. Baza podataka se ažurira

#### UC16 - Reklama obrta uz naknadu

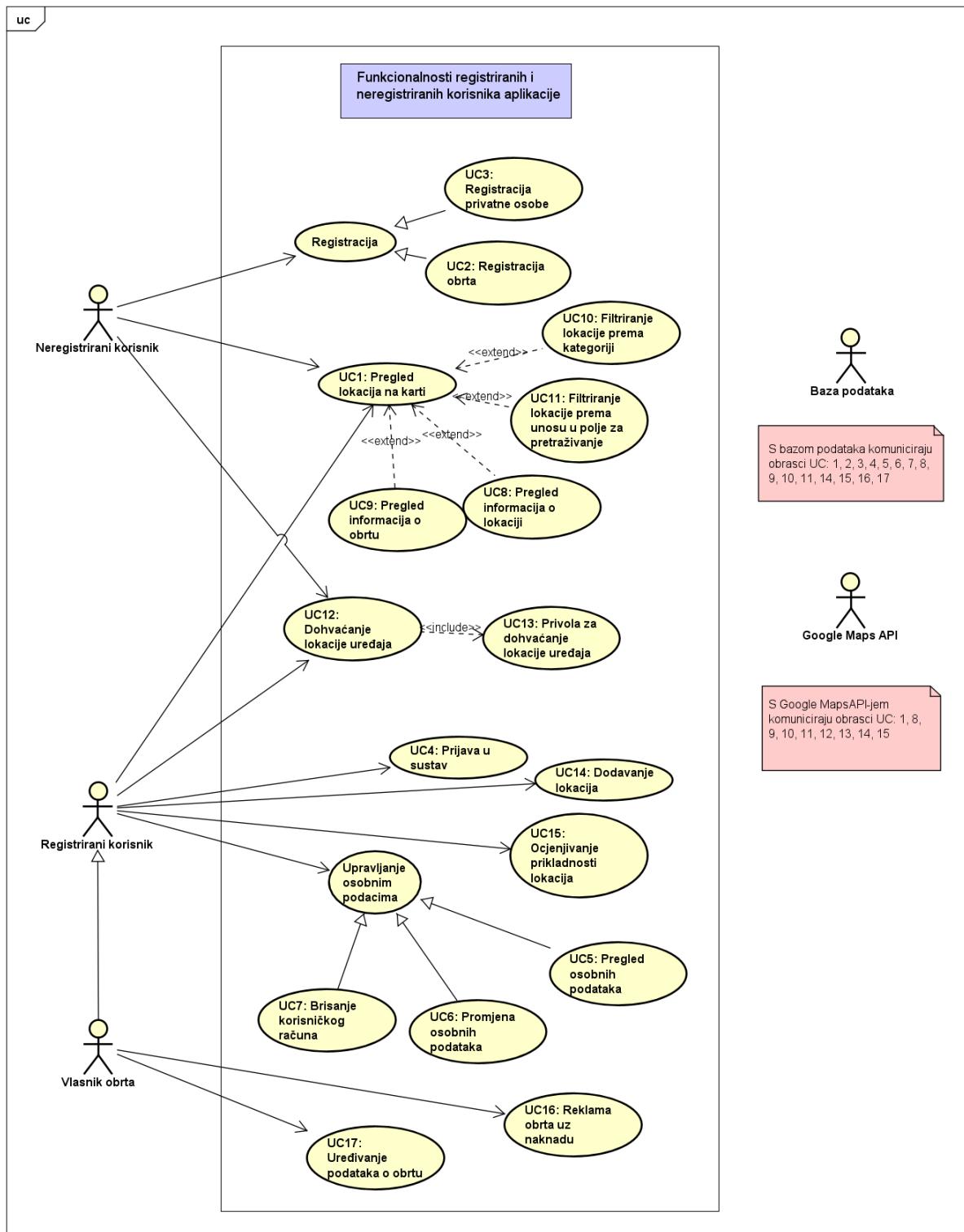
- **Glavni sudionik:** Registrirani korisnik
- **Cilj:** Istaknuti svoj objekt na karti
- **Sudionici:** Baza podataka, Google Maps API
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
  1. Korisnik odabire opciju "Istakni obrt"
  2. Korisnik unosi kartične podatke
  3. Korisnik plaća naknadu
  4. Baza podataka se ažurira
  5. Lokacija postaje istaknuta na karti
- **Opis mogućih odstupanja:**
  - 3.a Kartica je odbijena
    1. Prikazuje se poruka o grešci, korisnika se vraća na početnu stranicu

#### UC17 - Uređivanje podataka o obrtu

- **Glavni sudionik:** Registrirani korisnik
- **Cilj:** Prikazati točnu lokaciju korisnika

- **Sudionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
  1. Korisnik odabire opciju za promjenu podataka
  2. Korisnik mijenja podatke o svom obrtu
  3. Korisnik sprema promjene
  4. Baza podataka se ažurira
- **Opis mogućih odstupanja:**
  - 2.a Korisnik unosi podatke u nedozvoljenom formatu
    1. Sustav obaveštava korisnika o neuspjelom upisu

## Dijagrami obrazaca uporabe

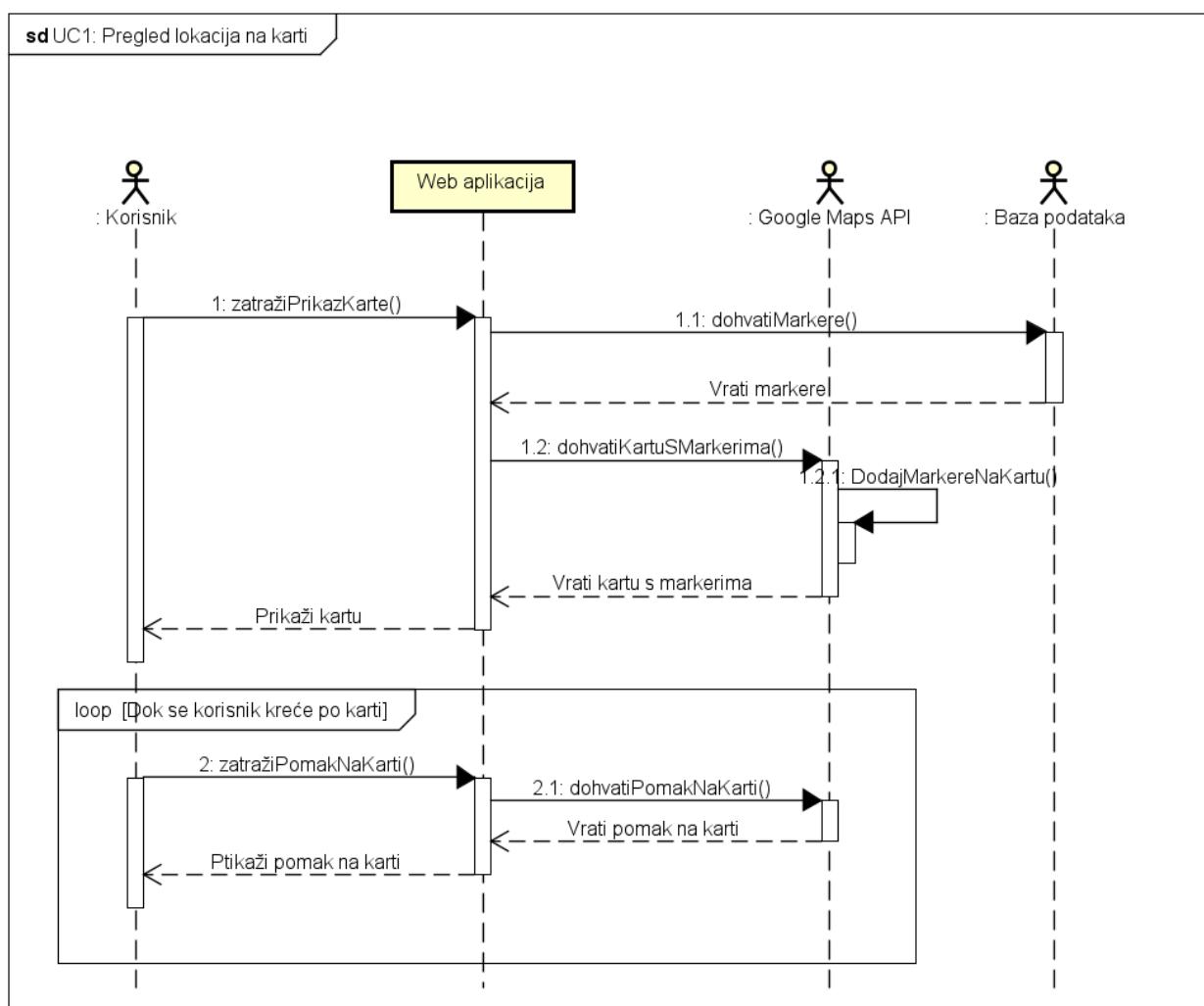


Slika 3.1: Dijagram obrazaca uporabe

### 3.1.2 Sekvencijski dijagrami

Obrazac uporabe UC1 - Pregled lokacija na karti

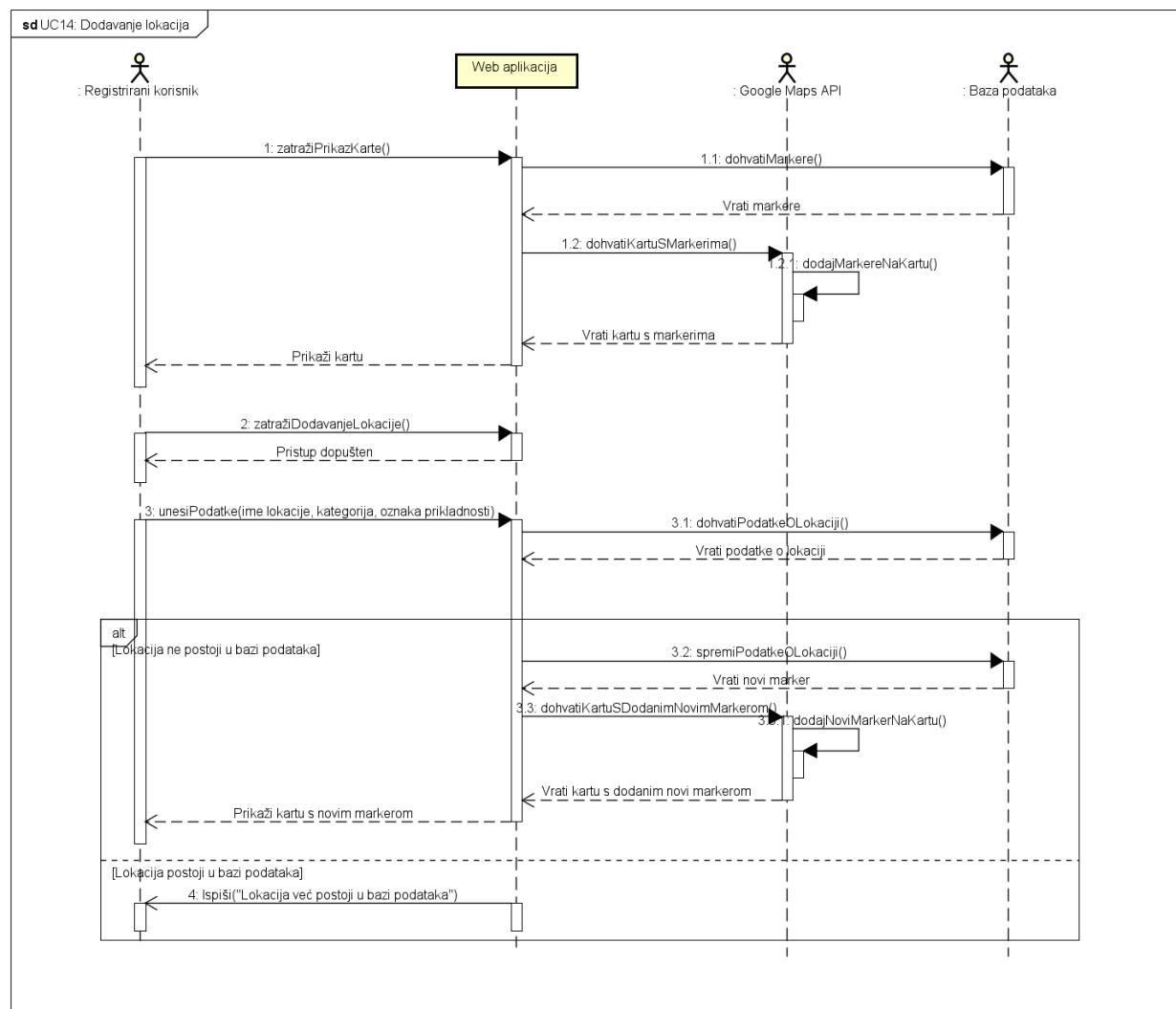
Registrirani ili neregistrirani korisnik šalje zahtjev za prikaz karte s lokacijama. Poslužitelj dohvaća markere iz baze podataka. Zatim markere preuzima Google Maps API koji ih dodaje na kartu. Poslužitelj dohvaća kartu s markerima te je prikazuje korisniku. Dok god se korisnik kreće po karti, poslužitelj od njega prima zahtjeve za svaki korak jer pomake na karti mora zatražiti od Google Maps API-ja čija je uloga omogućiti kretanje po karti. Poslužitelj zatim prikazuje pomake na karti korisniku.



Slika 3.2: Sekvencijski dijagram za UC1

Obrazac uporabe U14 - Dodavanje lokacija

Registrirani korisnik šalje zahtjev za prikaz karte s lokacijama kako bi mogao odabrati opciju dodavanja lokacije. Poslužitelj dohvaća markere iz baze podataka. Zatim markere preuzima Google Maps API koji ih dodaje na kartu. Poslužitelj dohvaća kartu s markerima te je prikazuje korisniku. Korisnik odabire opciju dodavanja lokacije, a poslužitelj mu daje pristup jer je registriran te korisnik unosi podatke o lokaciji (ime lokacije, kategoriju te ocjenu/oznaku prikladnosti). Poslužitelj za unesene podatke o lokaciji dohvaća iz baze podataka podatke o toj lokaciji. Ako oni ne postoje u bazi podataka, spremaju se u bazu podataka te poslužitelj dohvaća iz nje novi marker. Preuzima ga Google Maps API koji ga dodaje na kartu te poslužitelj dohvaća kartu s dodanim novim markerom i prikazuje je korisniku. Ako podaci o lokaciji već postoje u bazi, korisnik prima informaciju o postojanju podataka za željenu lokaciju.



Slika 3.3: Sekvencijski dijagram za UC14

### 3.2 Ostali zahtjevi

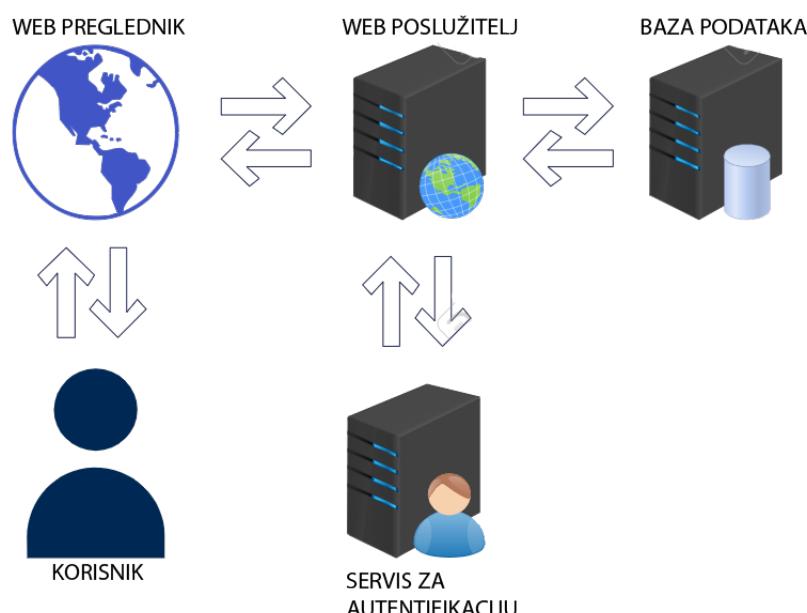
- Sustav treba omogućiti rad više korisnika u stvarnom vremenu
- Korisničko sučelje i sustav moraju podržavati hrvatsku abecedu (dijakritičke znakove) pri unosu i prikazu tekstualnog sadržaja
- Izvršavanje dijela programa u kojem se pristupa bazi podataka ne smije trajati duže od nekoliko sekundi
- Sustav treba biti implementiran kao web aplikacija koristeći objektni-orientirane jezike
- Neispravno korištenje korisničkog sučelja ne smije narušiti funkcionalnost i rad sustava
- Sustav treba biti jednostavan za korištenje, korisnici se moraju znati koristiti sučeljem bez opširnih uputa
- Nadogradnja sustava ne smije narušavati postojeće funkcionalnosti sustava
- Sustav kao valutu koristi HRK
- Veza s bazom podataka mora biti kvalitetno zaštićena, brza i otporna na vanjske greške
- Pristup sustavu mora bizi omogućen iz javne mreže pomoću HTTPS

## 4. Arhitektura i dizajn sustava

Arhitektura se može podijeliti na četiri podsustava:

- Web poslužitelj
- Web aplikacija
- Baza podataka
- Servis za autentifikaciju

*Web preglednik* je program koji korisniku omogućuje pregled web-stranica i multimedijalnih sadržaja vezanih uz njih. Svaki internetski preglednik je prevoditelj. Dakle, stranica je pisana u kodu koji preglednik nakon toga interpretira kao nešto svakome razumljivo. Korisnik putem web preglednika šalje zahtjev web poslužitelju.



Slika 4.1: Prikaz arhitekture

*Web poslužitelj* osnova je rada web aplikacije. Njegova primarna zadaća je komunikacija klijent s aplikacijom. Komunikacija se odvija preko HTTP (engl. *Hyper Text Transfer Protocol*) protokola, što je protokol u prijenosu informacija na webu. Poslužitelj je onaj koji pokreće web aplikaciju te joj prosljeđuje zahtjeve.

Korisnik korisi *web aplikaciju* za obrađivanje željenih zahtjeva. Web aplikacija obrađuje zahtjev te ovisno o zahtjevu, pristupa bazi podataka i/ili servisu za autentifikaciju nakon čega preko poslužitelja vraća korisniku odgovor u obliku HTML dokumenta vidljivog u web pregledniku.

Programski jezik kojeg smo odabrali za izradu naše web aplikacije je JavaScript za koji koristimo React biblioteku (engl. *library*) te Next.js radni okvir (engl. *framework*). Za autentifikaciju koristimo Google-ov modul za autentifikaciju iz njihovog servisa Firebase. Odabrano razvojno okruženje je Microsoft Visual Studio Code. Iako ne potpuno, arhitektura sustava temeljiti će se na MVC (Model-View-Controller) konceptu. MVC koncept nije potpuno podržan od strane Next.js-a jer je to radni okvir koji omogućava posluživanje React aplikacija sa servera, a u samom React-u MVC koncept nije potpuno podržan. No, radi lakše izrade i organizacije projekta, odlučili smo na što bliži način implementirati MVC koncept.

Karakteristika MVC koncepta je nezavisan razvoj pojedinih dijelova aplikacije što za posljedicu ima jednostavnije ispitivanje kao i jednostavno razvijanje i dodavanje novih svojstava u sustav.

MVC koncept sastoji se od:

- **Model** - Središnja komponenta sustava. Predstavlja dinamičke strukture podataka, neovisne o korisničkom sučelju. Izravno upravlja podacima, logikom i pravilima aplikacije. Također prima ulazne podatke od Controllera
- **View** - Bilo kakav prikaz podataka, poput grafa. Mogući su različiti prikazi iste informacije poput grafičkog ili tabličnog prikaza podataka.
- **Controller** - Prima ulaze i prilagođava ih za prosljeđivanje Modelu i Viewu. Upravlja korisničkim zahtjevima i zemeljem njih izvodi daljnju interakciju s ostalim elementima sustava.

## 4.1 Baza podataka

Za potrebe našeg sustava koristit ćemo Firestore, Google-ovu NoSQL cloud bazu podataka iz njihovog servisa Firebase koja svojom strukturom omogućava lage izmijene koje će pratiti skaliranje aplikacije. Gradivna jedinka baze je kolekcija (engl. *collection*) definirana svojim imenom i skupom dokumenata (engl. *documents*). Zadaća baze podataka je brza i jednostavna pohrana, izmjena i dohvata podataka za daljnju obradu. Baza podataka ove aplikacije sastoji se od sljedećih kolekcija:

- users
- companies
- locations
- PaymentInfo

### 4.1.1 Opis tablica

**Users** Ova tablica sadrži sve relevantne informacije za korisnike aplikacije. Sadrži atribute: UserId, Username, email i CompanyOwner(boolean atribut koji nam govori je li korisnik vlasnik obrta). Unutar sebe tablica sadrži kolekciju PaymentInfo u kojoj su informacije o kartičnom plaćanju. Tablica je u vezi *One-to-One* s tablicom Companies i u vezi *One-to-Many* je s tablicom Locations.

Users		
UserId	VARCHAR	Hashirani string jedinstven za svakog korisnika
Username	VARCHAR	Nadimak pod kojim će se korisnik predstavljati u aplikaciji
email	VARCHAR	E-mail adresa korisnika
CompanyOwner	BOOLEAN	True ako je korisnik vlasnik obrta, inače False
PaymentInfo	COLLECTION	Kolekcija s informacijama u plaćanju

**PaymentInfo** ova tablica(kolekcija) je sadržana unutar tablice Users. Ovakav način rada nam dopuštaju NoSql baze poput Firebasea. Unutar ove kolekcije nalaze se svi potrebni podatci o plaćanju karticom.

PaymentInfo		
VAT	INT	Identifikacijski broj
Adress	VARCHAR	Adresa vlasnika kartice
CardCVC	INT	Verifikacijski broj kartice
ExpiryDate	DATE	Datum isteka kartice
City	VARCHAR	Grad vlasnika
CompanyNamePay	VARCHAR	Ime vlasnika kartice
CompanyOIBPay	VARCHAR	OIB vlasnika kartice
Country	VARCHAR	Zemlja vlasnika
FirstName	VARCHAR	Ime vlasnika
LastName	VARCHAR	Prezime vlasnika
Region	VARCHAR	Regija vlasnika
ZipCode	INT	Poštarski broj
Geopoint	GEOPOINT	Lokacija vlasnika kartice

**Locations** Ova tablica sadrži informacije o lokacijama dodanima od strane korisnika. Sadrži atribut: Ime lokacije, id korisnika koji je dodao lokaciju, status prikladnosti i koordinate. U vezi *Many-to-One* je s tablicom Users.

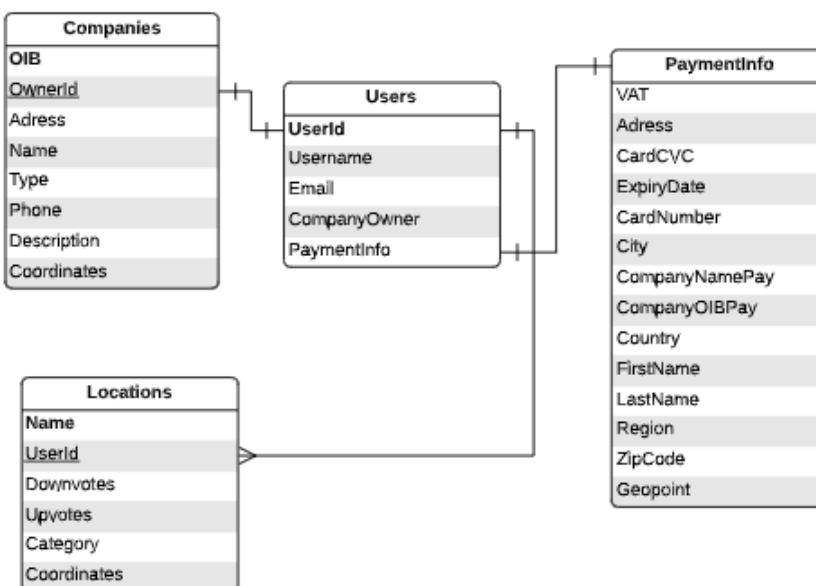
Locations		
Name	VARCHAR	Ime lokacije
UserId	VARCHAR	Hashirani string niz jedinstven za korisnika koji je dodao lokaciju
Coordinates	VARCHAR	Koordinate lokacije
Category	VARCHAR	Kategorija lokacije
Upvotes	INT	Broj pozitivnih ocjena
Downvotes	INT	Broj negativnih ocjena

**Companies** Ova tablica sadrži informacije o dodanim obrtima. Posjeduje atribut: OIB obrta, Id vlasnika obrta, adresu, naziv obrta, kojoj kategoriji pripada i kratki opis obrta. U vezi *One-to-One* je s tablicom Users.

<b>Companies</b>		
OIB	VARCHAR	Niz od 11 znamenki karakterističan za tu pravnu osobu
OwnerId	VARCHAR	Hashirani string niz jedinstven za korisnika koji je ujedino i vlasnik obrta
Adress	VARCHAR	Adresa obrta
Name	VARCHAR	Ime obrta
Type	VARCHAR	Kategorija kojoj obrt pripada
Phone	VARCHAR	Kontakt broj obrta
Description	VARCHAR	Kratki opis obrta
Coordinates	GEOPOINT	Koordinate obrta

#### 4.1.2 Dijagram baze podataka

Na dijagrame ključevi su "boldani", a strani ključevi podcrtani.



Slika 4.2: Dijagram baze podataka

## 4.2 Dijagram razreda

Na slici 4.3 prikazan je dijagram razreda cijelog projekta. Implementirane metode direktno komuniciraju s bazom podataka te vraćaju tražene podatke.

PrivatnaForm služi za unos informacija o privatnom korisniku, vrši provjeru ispravnosti unosa i ako je sve u redu kreira korisnika u firebaseu

VlasnikForm služi za unos informacija o obrtu. Sadrži podatke potrebne za neki obrt, provjerava ispravnost unesenih podataka. Sadrži podatke o kartici za plaćanje i ako je sve u redu kreira korisnika u firebaseu.

Firebase služi za konfiguraciju baze i komunikaciju s istom.

Context stvara kontekst za trenutnog korisnika

Hooks provjerava nalazi li se lozinka u "password blacklisti" i ako da, traži promjenu lozinke.

ChangePassword prikazuje stranicu za promjenu lozinke. Ako su sva polja popunjena i nema grešaka, submit gumb je omogućen. U protivnom nije. Provjerava se ispunjava li lozinka sve zahtjeve. Ako su svi zahtjevi ispunjeni, omogućuje se promjena trenutne lozinke u novu lozinku.

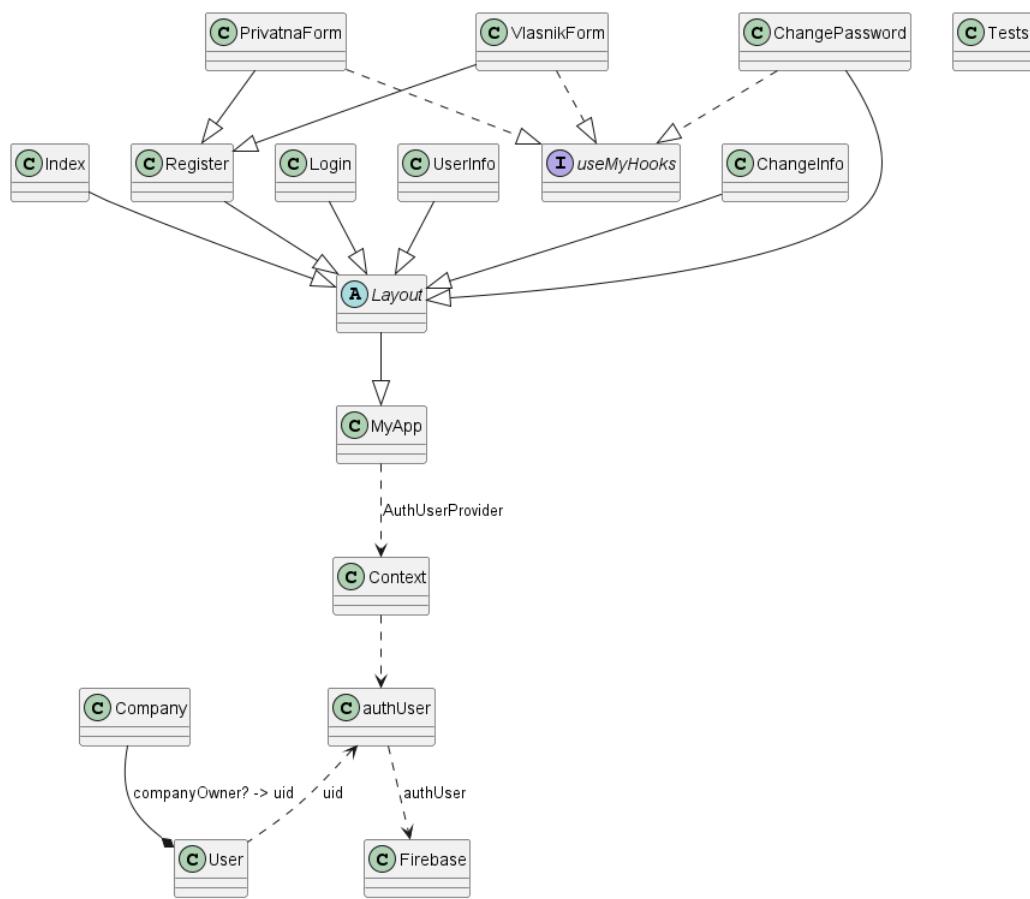
ChangeInfo prikazuje stranicu za promjenu osobnih podataka. Omogućuje promjenu osobnih podataka ako novi podatci zadovoljavaju postavljene uvjete.

Index prikazuje početnu stranicu na kojoj se prikazuje mapa, te sa koje se može prebaciti na ostale stranice.

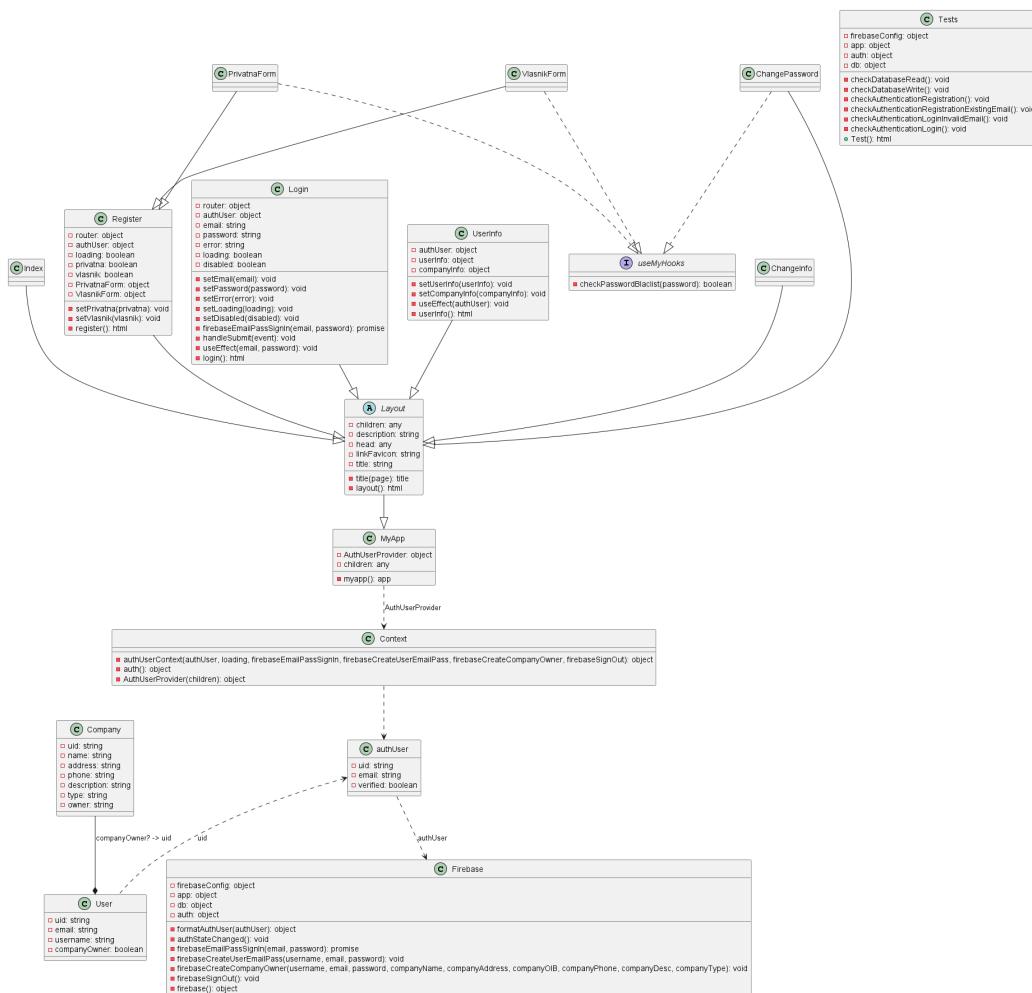
Login korisniku prikazuje stranicu za ulogiravanje u aplikaciju. Prilikom logina gleda se postoji li korisnik u bazi, ako ne javlja se greška. Ako postoji ulogirava se u aplikaciju. Ne dozvoljava se upis dok sva polja nisu ispravno upisana.

Register korisniku prikazuje stranicu za registraciju. U zavisnosti od odabranog prikazuje se ili "PrivatnaForma" ili "VlasnikForma"

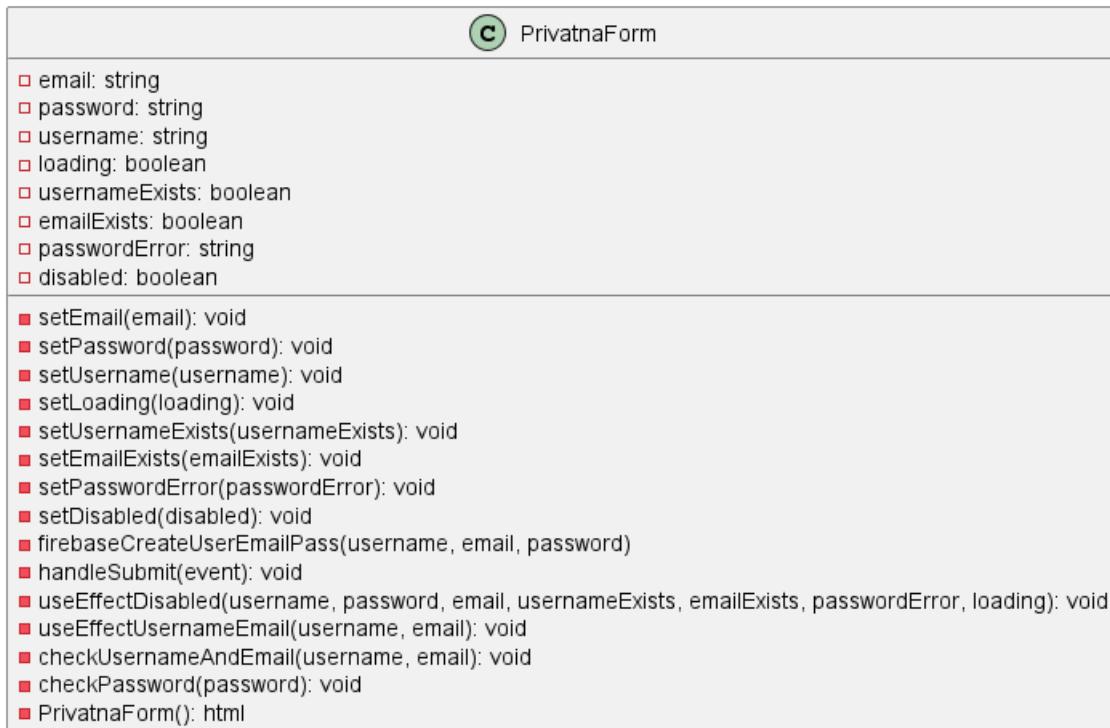
UserInfo prikazuje stranicu sa podatcima o ulogiranom korisniku. Prvotno funkcija povlači podatke o korisniku iz baze i ako je korisnik vlasnik firme, povlači i podatke o firmi. Te se isti podatci prikazuju na stranici



Slika 4.3: Prikaz ovisnosti među razredima



Slika 4.4: Dijagram razreda



Slika 4.5: Detaljniji prikaz razreda PrivatnaForm

(C) VlasnikForm

```

□ email: string
□ password: string
□ username: string
□ loading: boolean
□ companyName: string
□ companyAddress: string
□ companyGeopoint: object
□ companyOIB: string
□ companyPhone: string
□ companyDesc: string
□ companyType: string
□ usernameExists: boolean
□ emailExists: boolean
□ passwordError: string
□ companyNameError: string
□ companyOIBError: string
□ companyPhoneError: string
□ companyDescError: string
□ firstName: string
□ lastName: string
□ companyNamePay: string
□ companyOIBPay: string
□ address: string
□ geopoint: object
□ country: string
□ region: string
□ city: string
□ zipCode: string
□ VAT: string
□ cardNumber: string
□ cardExpiryDate: string
□ cardCVC: string
□ companyNamePayError: string
□ companyOIBPayError: string
□ companyTypes: array
□ disabled: boolean
□ checked: boolean
□ isLoaded: boolean
□ loadError: string
□ search: object
□ searchCard: object
□ libraries: array
□ router: object
□ userInfoSwitch: object

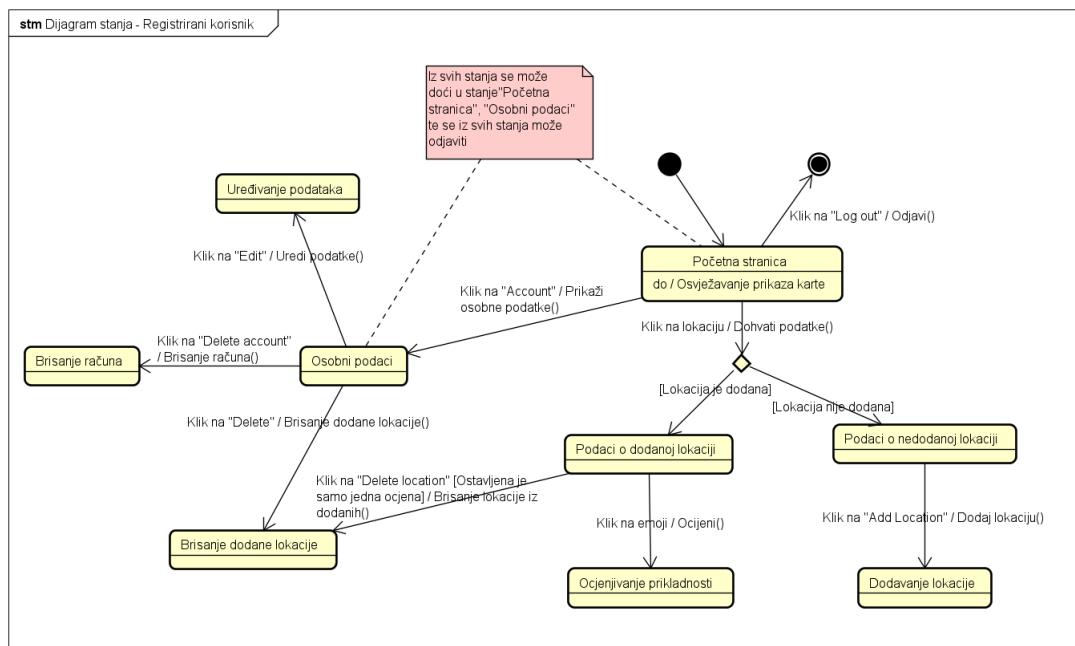
■ setEmail(email): void
■ setPassword(password): void
■ setUsername(username): void
■ setLoading/loading(): void
■ setCompanyName(companyName): void
■ setCompanyAddress(companyAddress): void
■ setCompanyGeopoint(companyGeopoint): void
■ setCompanyOIB(companyOIB): void
■ setCompanyPhone(companyPhone): void
■ setCompanyDesc(companyDesc): void
■ setCompanyType(companyType): void
■ setUsernameExists(usernameExists): void
■ setEmailExists(emailExists): void
■ setPasswordError(passwordError): void
■ setCompanyNameError(companyNameError): void
■ setCompanyOIBError(companyOIBError): void
■ setCompanyPhoneError(companyPhoneError): void
■ setCompanyDescError(companyDescError): void
■ setFirstName(firstName): void
■ setLastName(lastName): void
■ setCompanyNamePay(companyNamePay): void
■ setCompanyOIBPay(companyOIBPay): void
■ setAddress(address): void
■ setGeopoint(geopoint): void
■ setCountry(country): void
■ setRegion(region): void
■ setCity(city): void
■ setZipCode(zipCode): void
■ setVAT(VAT): void
■ setCardNumber(cardNumber): void
■ setCardExpiryDate(cardExpiryDate): void
■ setCardCVC(cardCVC): void
■ setCompanyNamePayError(companyNamePayError): void
■ setCompanyOIBPayError(companyOIBPayError): void
■ setDisabled(disabled): void
■ setChecked(checked): void
■ setSearch(search): void
■ setSearchCard(searchCard): void
■ onSearchLoad(search): void
■ onSearchLoadCard(searchCard): void
■ onPlaceChanged(): void
■ onPlaceChangedCard(): void
■ firebaseCreateCompanyOwner(username, email, password, companyName, companyAddress, companyOIB, companyPhone, companyDesc, companyType)
■ handleSubmit(event): void
■ useEffectDisabled(username, password, email, usernameExists, loading, companyName, companyAddress, companyOIB, companyPhone, companyDesc, companyType, loading): void
■ useEffectUsernameEmail(username, email): void
■ checkUserNameAndEmail(username, email): void
■ checkPassword(password): void
■ checkCompanyName(companyName): void
■ checkCompanyPhone(companyPhone): void
■ checkCompanyDesc(companyDesc): void
■ checkCompanyOIB(companyOIB): void
■ checkCompanyNamePay(companyNamePay): void
■ checkCompanyOIBPay(companyOIBPay): void
■ addUserInfo(): void
■ VlasnikForm(): html

```

Slika 4.6: Detaljniji prikaz razreda VlasnikForm

## 4.3 Dijagram stanja

Dijagram stanja prikazuje stanja objekta te prijelaze iz jednog stanja u drugo temeljenje na događajima. Na slici 4.7 prikazan je dijagram stanja za registriranog korisnika. Nakon prijave, klijentu se prikazuje početna stranica na kojoj vidi kartu te lokacije. Za odabranu lokaciju, korisnik može ostaviti ocjenu prikladnosti ako je lokacija postojeća ili dodati novu ako ne postoji u bazi. Također, korisnik može klikom na "Account" pregledati i uređivati osobne podatke te obrisati račun ili dodanu lokaciju.

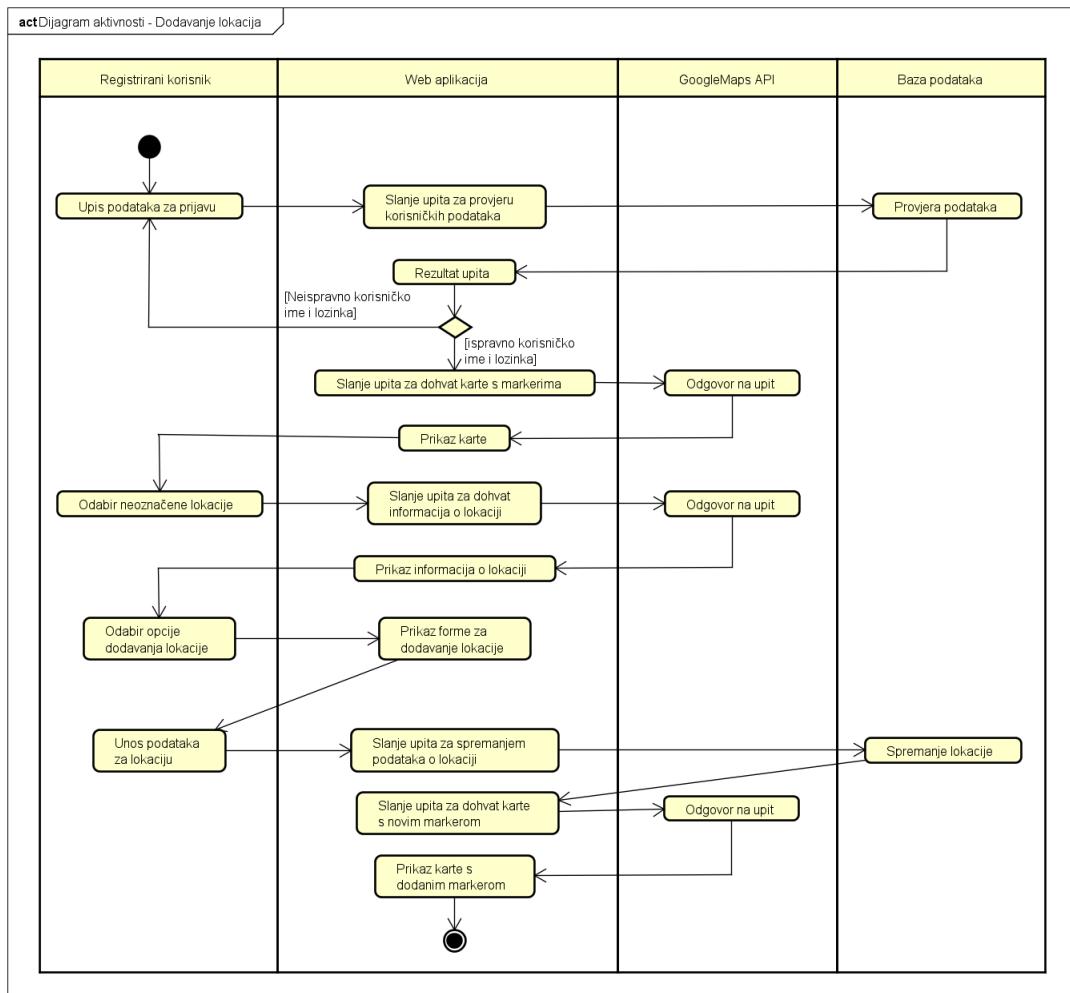


Slika 4.7: Dijagram stanja

## 4.4 Dijagram aktivnosti

Dijagram aktivnosti primjenjuje se za opis modela toka upravljanja ili toka podataka. Ne upotrebljava se za modeliranje događajima poticanog ponašanja. U modeliranju toka upravljanja svaki novi korak poduzima se nakon završenog prethodnog, a naglasak je na jednostavnosti. Na dijagramu aktivnosti 4.8 prikazan je proces dodavanja lokacije. Korisnik se prijavljuje u sustav, odabire lokaciju na karti koja već nije u bazi podataka (nema marker) te mu se prikazuju informacije o toj lokaciji. Odabire opciju dodavanja lokacije, ispunjava podatke za lokaciju i lokacija se spremi u bazu podataka te se korisniku prikazuje karta s novododanim

markerom.

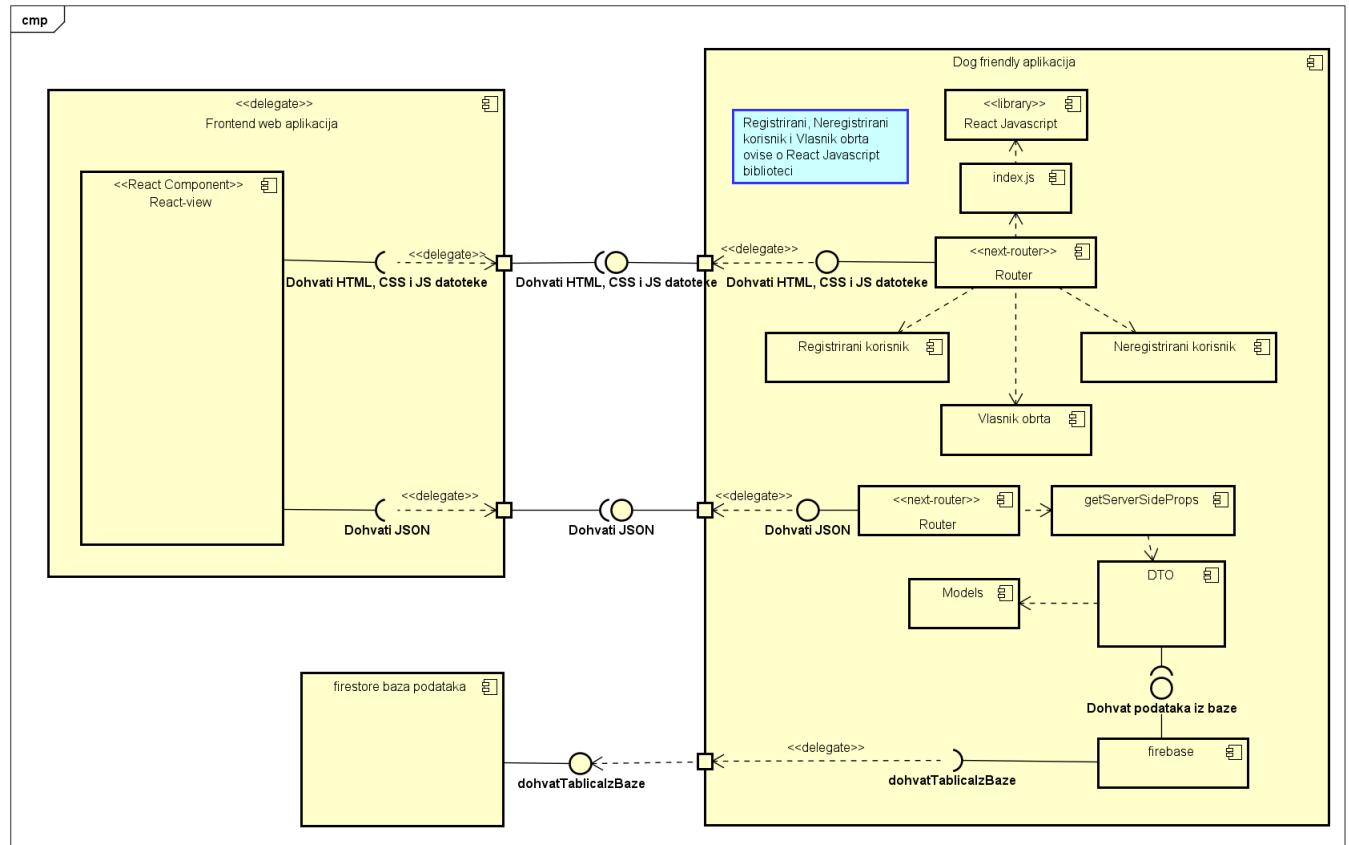


Slika 4.8: Dijagram aktivnosti

## 4.5 Dijagram komponenti

Dijagram komponenti prikazan na slici 4.9. opisuje organizaciju i međuvisnost komponenti, interne strukture i odnose prema okolini. Sustavu se pristupa preko dva različita sučelja. Preko sučelja za dohvat HTML, CSS i JS datoteka poslužuju se datoteke koje pripadaju frontende dijelu aplikacije. Router je komponenta koja na upis s url određuje koja datoteka će se poslužiti na sučelje. Frontend dio se sastoji od niza JavaScript datoteka koje su raspoređene u logičke cjeline nazvane po tipovima aktora koji im pristupaju. Sve JavaScript datoteke ovise o React biblioteci iz koje dohvaćaju gotove komponente kao što su gumbi, forme i slično. Preko sučelja za dohvat JSON podataka pristupa se next-router komponenti. next-router

poslužuje podatke koji pripadaju backend dijelu aplikacije, a podatke s poslužitelja se dohvaća pomoću komponente GetServerSideProps. Firebase je zadužen za dohvaćanje tablica iz firestore baze podataka. Podaci koji su pristigli iz baze se šalju dalje u obliku DTO (Data transfer object). Reactview komponenta preko dostupnih sučelja komunicira s Dog friendly aplikacijom te ovisno o korisnikovim akcijama osvježava prikaz i dohvaća nove podatke ili datoteke.



Slika 4.9: Dijagram komponenti

# 5. Implementacija i korisničko sučelje

## 5.1 Korištene tehnologije i alati

Članovi unutar tima su komunicirali koristeći aplikaciju Discord. Za izradu UML dijagrama koristen je alat Astah Professional , a kao sustav za upravljanje izvornim kodom Git. Udaljeni repozitorij projekta je dostupan na web platformi GitLab. Od razvojnih okruženja smo koristili Microsoft Visual Studio Code i Sublime Text 3. Svi djelovi aplikacije su ostvareni korištenjem programskog jezika JavaScript. Za izradu frontenda koristena je biblioteka ReactJs. Održavana je od strane Facebooka. React se najčešće koristi kao osnova u razvoju web ili mobilnih aplikacija. Za backend je korišten Next.js framework kreirao od strane Vercela. Next.js se koristi za "server-side rendering" i generiranje statičkih stranica. Za prikaz podataka na karti koristili smo Google Maps api. Dizajn je ostvaren koristeći web aplikaciju Figma koja služi za dizajn korisničkih sučelja. Bazu podataka smo napravili koristeći Firebase, koji služi za izradu NoSql baza i održava ga Google. Za pisanje dokumentacije korištena je web aplikacija Overleaf.

## 5.2 Ispitivanje programskog rješenja

Testiranje programskog rješenja provedeno je ručno napisanim funkcijama i selenium testovima. U nastavku slijedi programski kod funkcija i rezultati testiranja.

### 5.2.1 Ispitivanje komponenti

```
Console was cleared
Creating document reference... ✅
Document reference is valid ✅
Getting document... ✅
Successfully got document ✅
Document data: ► {companyOwner: false, email: 'jane@doe.com', username: 'janedoe'}
```

Slika 5.1: Provjera čitanja podataka iz baze

```
const checkDatabaseRead = () => {
    console.clear();
    console.log("Creating document reference... ✅");
    const docRef = doc(db, "dummyData", "WRbnbMZZwSE04N3klm82");
    if (docRef) {
        console.log("Document reference is valid ✅");
    } else {
        console.log("Document reference is invalid ❌");
    }
    console.log("Getting document... ✅");
    getDoc(docRef)
        .then((doc) => {
            if (doc.exists()) {
                console.log("Successfully got document ✅");
                console.log("Document data:", doc.data());
            } else {
                // doc.data() will be undefined in this case
                console.log("No such document! ❌");
            }
        })
        .catch((error) => {
            console.log("Error getting document ❌");
            console.log("Error getting document:", error);
        });
};
```

Slika 5.2: Provjera čitanja podataka iz baze kod

```
Console was cleared
Logging in... 🌟
Email: permanent@test.com
Password: test12341234
Successfully logged in ✅
User:
▶ UserImpl {providerId: 'firebase', proactiveRefresh: ProactiveRefresh, reloadUserInfo: {...}, reloadListener: ...}
```

Slika 5.3: Provjera prijave

```
const checkAuthenticationLogin = () => {
    console.clear();
    console.log("Logging in... 🌟");
    const email = "permanent@test.com";
    const password = "test12341234";
    console.log("Email: " + email);
    console.log("Password: " + password);

    signInWithEmailAndPassword(auth, email, password)
        .then((userCredential) => {
            // Signed in
            const user = userCredential.user;
            console.log("Successfully logged in ✅");
            console.log("User: ", user);
            // ...
        })
        .catch((error) => {
            const errorCode = error.code;
            const errorMessage = error.message;
            console.log("Error logging in ❌");
            console.log("Error code: " + errorCode);
            console.log("Error message: " + errorMessage);
        });
};
```

Slika 5.4: Provjera prijave kod

```
Console was cleared
Logging in... 🌟
Email: permanent2@test.com
Password: test12341234
✖ ▶ POST http://localhost:9099/identitytoolkit.googleapis.com/v1/accounts:signInWithPassword?key=AIZaSyDwMl6K77eBRbb6s876n50kjRc_xTp4pDY 400 (Bad Request)
Error logging in ❌
Error code: auth/user-not-found
Error message: Firebase: Error (auth/user-not-found).
```

Slika 5.5: Provjera prijave s krivim mailom

```

const checkAuthenticationLoginInvalidEmail = () => {
  console.clear();
  console.log("Logging in... ✨");
  const email = "permanent2@test.com";
  const password = "test12341234";
  console.log("Email: " + email);
  console.log("Password: " + password);

  signInWithEmailAndPassword(auth, email, password)
    .then((userCredential) => {
      // Signed in
      const user = userCredential.user;
      console.log("Successfully logged in ✓");
      console.log("User: ", user);
      // ...
    })
    .catch((error) => {
      const errorCode = error.code;
      const errorMessage = error.message;
      console.log("Error logging in ✗");
      console.log("Error code: " + errorCode);
      console.log("Error message: " + errorMessage);
    });
};

```

Slika 5.6: Provjera prijave s krivim mailom kod

```

Console was cleared
DATA BEFORE WRITE
WRbnbMZZwSEO4N3klm82 =>
▶ {companyOwner: false, email: 'jane@doe.com', username: 'janedoe'}
#####
DATA TO WRITE
▶ {username: 'test', email: 'test@testing.com', companyOwner: true}
#####
WRITING DATA
#####
DATA AFTER WRITE
9pWkIrH4qv1Blo0z09Zp =>
▶ {companyOwner: true, email: 'test@testing.com', username: 'test'}
WRbnbMZZwSEO4N3klm82 =>
▶ {companyOwner: false, email: 'jane@doe.com', username: 'janedoe'}

```

Slika 5.7: Provjera unosa podataka u bazu

```

const checkDatabaseWrite = () => {
    console.clear();
    console.log("DATA BEFORE WRITE");
    getDocs(collection(db, "dummyData"))
        .then((querySnapshot) => {
            querySnapshot.forEach((doc) => {
                console.log(`#${doc.id} => `, doc.data());
            });

            console.log("#####");
            console.log("DATA TO WRITE");
            const data = {
                username: "test",
                email: "test@testing.com",
                companyOwner: true,
            };
            console.log(data);

            console.log("#####");
            console.log("WRITING DATA");
            addDoc(collection(db, "dummyData"), data)
                .then((docRef) => {
                    // console.log("Document written with ID: ", docRef.id);
                    console.log("#####");
                    console.log("DATA AFTER WRITE");
                    getDocs(collection(db, "dummyData")).then(
                        (querySnapshot) => {
                            querySnapshot.forEach((doc) => {
                                console.log(`#${doc.id} => `, doc.data());
                            });
                        }
                    );
                })
                .catch((error) => {
                    console.error("Error adding document: ", error);
                });
        })
        .catch((error) => {
            console.log("Error getting documents: ", error);
        });
};

}

```

Slika 5.8: Provjera unosa podataka u bazu kod

Console was cleared	tests.jsx?9e4a:103
Creating user... ✅	tests.jsx?9e4a:104
Email: testing@testing.com	tests.jsx?9e4a:107
Password: test12341234	tests.jsx?9e4a:108
Successfully created user ✅	tests.jsx?9e4a:113
User:	tests.jsx?9e4a:114
▶ UserImpl {providerId: 'firebase', proactiveRefresh: ProactiveRefresh, reloadUserInfo: {...}, reloadListener: null, uid: 'fy5AGr5aNnUDB1Zbzlw8ecrMgCV', ...}	tests.jsx?9e4a:114
Deleting user... ✅	tests.jsx?9e4a:117
Successfully deleted user ✅	tests.jsx?9e4a:121

Slika 5.9: Provjera registracije

```

const checkAuthenticationRegistrationExistingEmail = () => {
    console.clear();
    console.log("Creating user... ✨");
    const email = "permanent@test.com";
    const password = "test12341234";
    console.log("Email: " + email);
    console.log("Password: " + password);
    createUserWithEmailAndPassword(auth, email, password)
        .then((userCredential) => {
            // Signed in
            const user = userCredential.user;
            console.log("Successfully created user ✅");
            console.log("User: ", user);
            // ...
            // delete user
            console.log("Deleting user... ❌");
            user.delete()
                .then(() => {
                    // User deleted.
                    console.log("Successfully deleted user ✅");
                })
                .catch((error) => {
                    // An error occurred
                    // ...
                    console.log("Error deleting user ❌");
                });
        })
        .catch((error) => {
            const errorCode = error.code;
            const errorMessage = error.message;
            console.log("Error creating user ❌");
            console.log("Error code: " + errorCode);
            console.log("Error message: " + errorMessage);
            // ...
        });
};

```

Slika 5.10: Provjera registracije kod

```

Console was cleared
Creating user... ✨
Email: permanent@test.com
Password: test12341234
✖ POST http://localhost:9099/identitytoolkit.googleapis.com/v1/accounts:signUp?key=AIzaSyDwM16K77eBRbb6s876n50kjRc_xIp4pDY 400 (Bad Request)
Error creating user ❌
Error code: auth/email-already-in-use
Error message: Firebase: Error (auth/email-already-in-use).

```

Slika 5.11: Provjera registracije sa već postojećim mailom

```
const checkAuthenticationRegistrationExistingEmail = () => {
    console.clear();
    console.log("Creating user... ✅");
    const email = "permanent@test.com";
    const password = "test12341234";
    console.log("Email: " + email);
    console.log("Password: " + password);
    createUserWithEmailAndPassword(auth, email, password)
        .then((userCredential) => {
            // Signed in
            const user = userCredential.user;
            console.log("Successfully created user ✅");
            console.log("User: ", user);
            // ...
            // delete user
            console.log("Deleting user... ✅");
            user.delete()
                .then(() => {
                    // User deleted.
                    console.log("Successfully deleted user ✅");
                })
                .catch((error) => {
                    // An error occurred
                    // ...
                    console.log("Error deleting user ✗");
                });
        })
        .catch((error) => {
            const errorCode = error.code;
            const errorMessage = error.message;
            console.log("Error creating user ✗");
            console.log("Error code: " + errorCode);
            console.log("Error message: " + errorMessage);
            // ...
        });
};|
```

Slika 5.12: Provjera registracije sa već postojećim mailom kod

### 5.2.2 Ispitivanje sustava

**Ispitni slučaj 1: Uspješna prijava korisnika** Prijavljujemo se kao već registrirani korisnik sa točnim podatcima.

Rezultat: Uspješna prijava

	Command	Target	Value
1	✓ open	/	
2	✓ set window size	1250x678	
3	✓ click	css=a:nth-child(1) > i	
4	✓ click	css=.email-container > input	
5	✓ type	css=.email-container > input	markostrk@gmail.com
6	✓ type	css=password-container > input	markostrk02
7	✓ send keys	css=password-container > input	\$(KEY_ENTER)

Slika 5.13: Selenium Uspješna prijava

2. setWindowSize on 1250x678 OK
  3. click on css=a:nth-child(1) > i OK
  4. click on css=.email-container > input OK
  5. type on css=.email-container > input with value markostrk@gmail.com OK
  6. type on css=password-container > input with value markostrk02 OK
  7. sendKeys on css=password-container > input with value \${KEY\_ENTER} OK
- 'Uspjesna prijava korisnika' completed successfully**

Slika 5.14: Selenium Uspješna prijava - log

**Ispitni slučaj 2: Neuspješna prijava korisnika** Prijavljujemo se kao već registrirani korisnik sa netočnim podatcima.

Rezultat: Neuspješna prijava - vraćanje na početnu stranicu

	Command	Target	Value
1	✓ open	/	
2	✓ set window size	1552x840	
3	✓ click	css=.Header_header__ubBbX	
4	✓ click	css=a:nth-child(1) > i	
5	✓ click	css=.email-container > input	
6	✓ type	css=.email-container > input	petarperic@gmail.com
7	✓ type	css=password-container > input	pero1234
8	✓ send keys	css=password-container > input	\$(KEY_ENTER)

Slika 5.15: Selenium Neuspješna prijava

- Running 'Prijava korisnika - neuspjesna'
1. open on / OK
  2. setWindowSize on 1552x840 OK
  3. click on css=.Header\_header\_\_ubBbX OK
  4. click on css=a:nth-child(1) > i OK
  5. click on css=.email-container > input OK
  6. type on css=.email-container > input with value petarperic@gmail.com OK
  7. type on css=password-container > input with value pero1234 OK
  8. sendKeys on css=password-container > input with value \${KEY\_ENTER} OK
- 'Prijava korisnika - neuspjesna' completed successfully**

Slika 5.16: Selenium Neuspješna prijava - log

**Ispitni slučaj 3: Uspješna registracija privatnog korisnika** Popunjavamo obrazac za registraciju sa ispravnim podatcima.

Rezultat: Uspješna registracija korisnika

	Command	Target	Value
1	✓ open	/	
2	✓ set window size	1250x678	
3	✓ mouse over	css=a:nth-child(2) > i	
4	✓ click	css=a:nth-child(2) > i	
5	✓ click	name=username	
6	✓ type	name=username	ante123
7	✓ type	name=email	ante123@gmai.com
8	✓ type	name=password	antr12345
9	✓ click	css=.PrivatnaForm_button__qhfV	

Slika 5.17: Selenium Uspješna Registracija

```
Running 'Registracija'
1. open on / OK
2. setWindowSize on 1250x678 OK
3. mouseOver on css=a:nth-child(2) > i OK
4. click on css=a:nth-child(2) > i OK
5. click on name=username OK
6. type on name=username with value ante123 OK
7. type on name=email with value ante123@gmai.com OK
8. type on name=password with value antr12345 OK
9. click on css=.PrivatnaForm_button__qhfV OK
'Registracija' completed successfully
```

Slika 5.18: Selenium Uspješna registracija - log

**Ispitni slučaj 4: Neuspješna registracija privatnog korisnika** Popunjavamo obrazac za registraciju sa neispravnim podatcima.

Rezultat: Sustav javlja pogrešku, te nam nudi ponovni upis novih podataka

	Command	Target	Value
1	✓ open	/	
2	✓ set window size	1248x682	
3	✓ click	css=a:nth-child(2) > i	
4	✓ click	name=username	
5	✓ type	name=username	petar123
6	✓ click	name=email	
7	✓ type	name=email	petar123@yahoo.com
8	✓ type	name=password	petar
9	✓ send keys	name=password	\$(KEY_ENTER)

Slika 5.19: Selenium Neuspješna Registracija

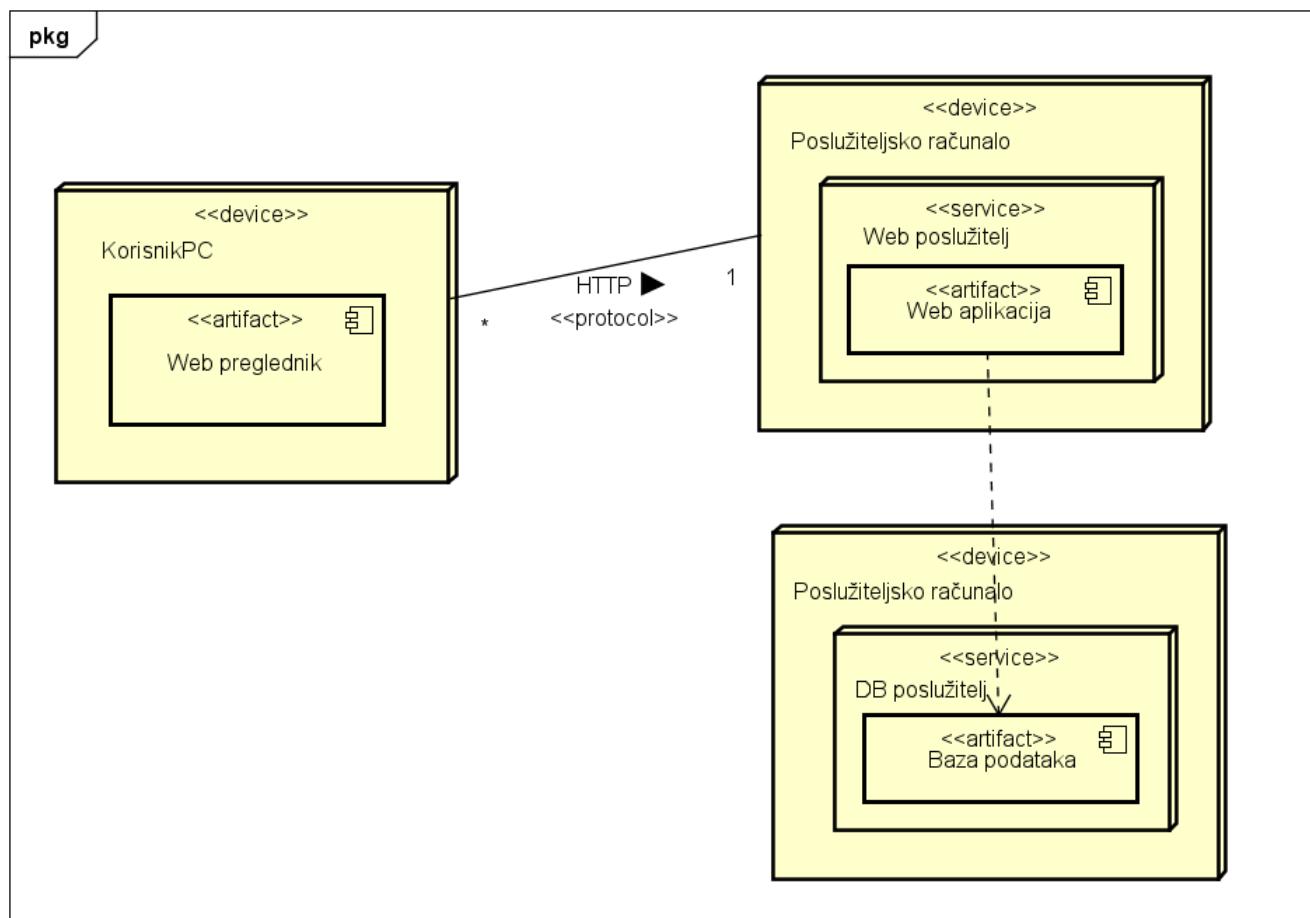
```
Running 'Neuspješna registracija'
1. open on / OK
2. setWindowSize on 1248x682 OK
3. click on css=a:nth-child(2) > i OK
4. click on name=username OK
5. type on name=username with value petar123 OK
6. click on name=email OK
7. type on name=email with value petar123@yahoo.com OK
8. type on name=password with value petar OK
9. sendKeys on name=password with value ${KEY_ENTER} OK
'Neuspješna registracija' completed successfully
```

---

Slika 5.20: Selenium Neuspješna registracija - log

### 5.3 Dijagram razmještaja

Dijagrami razmještaja opisuju topologiju sklopolja i programsku potporu koja se koristi u implementaciji sustava u njegovom radnom okruženju. Na jednom poslužiteljskom računalu nalazi se web poslužitelj, a na drugom poslužitelj baze podataka. 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 5.21: Dijagram razmještaja

## 5.4 Upute za puštanje u pogon

### Baza podataka

Za bazu podataka koristili smo Googleov Firebase u koji se prijavljujemo koristeći google račun progi.dogfriendly@gmail.com. Nakon prijave imamo pristup svim podatcima u bazi podataka. Možemo vidjeti sve tablice (collection), uređivati ih, dodavati/brisati nove podatke

### Hosting aplikacije

Za hostanje aplikacije koristili smo Vercel. To je platforma koja nam omogućuje jednostavno i besplatno povezivanje sa Gitlab računom i hostanje aplikacije. Potrebno je samo povezati gitlab račun te odabrati ime web stranice.

## 6. Zaključak i budući rad

Zadatak naše grupe bio je razvoj web aplikacije za prikaz, lociranje i dodavanje lokacija pogodnih za pse. Nakon 17 tjedana rada u timu i razvoja, ostvarili smo zadani cilj. Sama provedba projekta bila je kroz dvije faze.

Prva faza projekta uključivala je okupljanje tima za razvoj aplikacije, dodjelu projektnog zadatka i rad na dokumentiranju zahtjeva. Kvalitetan rad na prvom dijelu uvelike nam olakšava daljnji rad.

Druga faza projekta bila je puno intenzivnija po pitanju samostalnog rada članova. Osim realizacije rješenja, u drugoj fazi je bilo potrebno dokumentirati ostale UML dijagrame i izraditi popratnu dokumentaciju kako budući korisnici mogli lakše korisititi ili vršiti preinake na sustavu. Dobro izrađen kostur uštedio nam je puno vremena prilikom izrade aplikacije.

Komunikacija među članovima tima bila je putem Discorda čime smo postigli informiranost i uključenost svih članova grupe. Moguće proširenje postojeće inačice sustava je izrada mobilne aplikacije čime bi cilj projektnog zadatka bio osvaren u većoj mjeri no s web aplikacijom. Sudjelovanje na ovakovom projektu bilo je vrijedno iskustvo svim članovima tima jer smo kroz intenzivnih nekoliko tjedana rada iskusili zajednički rad na istom projektu, te naučili korisitit nove tehnologije. Također, osjetili smo važnost dobre vremenske organiziranosti i koordiniranosti između članova tima. Zadovoljni smo postignutim rezultatom bez obzira na prostor za usavrđavanje izrađene aplikacije.

# Popis literature

1. Programsko inženjerstvo, FER ZEMRIS, <http://www.fer.hr/predmet/proinzh>
2. The Unified Modeling Language, <https://www.uml-diagrams.org/>
3. Astah Community, <http://astah.net/editions/uml-new>
4. React Tutorial, <https://reactjs.org/tutorial/tutorial.html>
5. React dokumentacija, <https://reactjs.org/docs/getting-started.html>
6. Next.js Tutorial, <https://nextjs.org/learn/basics/create-nextjs-app>
7. Next.js dokumentacija, <https://nextjs.org/docs/getting-started>
8. Firebase dokumentacija, <https://firebase.google.com/docs>
9. React in 100 seconds, Fireship, <https://youtu.be/Tn6-PIqc4UM>
10. Next.js in 100 Seconds Plus Full Beginner's Tutorial, Fireship, <https://youtu.be/Tn6-PIqc4UM>
11. MUI dokumentacija, <https://mui.com/material-ui/getting-started/overview/>
12. Stackoverflow, <https://stackoverflow.com/>
13. GitHub, <https://github.com/>

# Indeks slika i dijagrama

2.1	Stranica za registraciju . . . . .	6
2.2	Početna stranica . . . . .	7
2.3	Primjer unosa novog obrta . . . . .	8
2.4	Dodavanje osvrta . . . . .	8
2.5	Prikaz nepotrebnih sadržaja . . . . .	9
3.1	Dijagram obrazaca uporabe . . . . .	18
3.2	Sekvencijski dijagram za UC1 . . . . .	19
3.3	Sekvencijski dijagram za UC14 . . . . .	21
4.1	Prikaz arhitekture . . . . .	23
4.2	Dijagram baze podataka . . . . .	27
4.3	Prikaz ovisnosti među razredima . . . . .	29
4.4	Dijagram razreda . . . . .	30
4.5	Detaljniji prikaz razreda PrivatnaForm . . . . .	31
4.6	Detaljniji prikaz razreda VlasnikForm . . . . .	32
4.7	Dijagram stanja . . . . .	33
4.8	Dijagram aktivnosti . . . . .	34
4.9	Dijagram komponenti . . . . .	35
5.1	Provjera čitanja podataka iz baze . . . . .	37
5.2	Provjera čitanja podataka iz baze kod . . . . .	37
5.3	Provjera prijave . . . . .	38
5.4	Provjera prijave kod . . . . .	38
5.5	Provjera prijave s krivim mailom . . . . .	38
5.6	Provjera prijave s krivim mailom kod . . . . .	39
5.7	Provjera unosa podataka u bazu . . . . .	39
5.8	Provjera unosa podataka u bazu kod . . . . .	40
5.9	Provjera registracije . . . . .	40
5.10	Provjera registracije kod . . . . .	41
5.11	Provjera registracije sa već postojećim mailom . . . . .	41

5.12 Provjera registracije sa već postojećim mailom kod . . . . .	42
5.13 Selenium Uspješna prijava . . . . .	43
5.14 Selenium Uspješna prijava - log . . . . .	43
5.15 Selenium Neuspješna prijava . . . . .	43
5.16 Selenium Neuspješna prijava - log . . . . .	43
5.17 Selenium Uspješna Registracija . . . . .	44
5.18 Selenium Uspješna registracija - log . . . . .	44
5.19 Selenium Neuspješna Registracija . . . . .	44
5.20 Selenium Neuspješna registracija - log . . . . .	45
5.21 Dijagram razmještaja . . . . .	46
6.1 Prikaz aktivnosti u projektu . . . . .	58

# Dodatak: Prikaz aktivnosti grupe

## Dnevnik sastajanja

### 1. sastanak

- 20.10.2022., 13:45
- Prisustvovali: Laura Majer, Petra-Dunja Grujić-Ostojić, Fran Markulin, Karla Udiljak, Branimir Medvedec, Jura Starčević, Luka Radman, Marko Štrk
- Teme sastanka:
  - sastanak s asistenticom i demonstratoricom
  - raščišćavanje osnovnih dilema funkcionalnosti
  - analiza zadatka

### 2. sastanak

- Datum: 02.11.2022., 13:00
- Prisustvovali: Fran Markulin, Karla Udiljak, Branimir Medvedec, Jura Starčević, Luka Radman, Marko Štrk
- Teme sastanka:
  - podijela dužnosti:
    - \* Fran Markulin: voditelj projekta, developer
    - \* Karla Udiljak: designer
    - \* Branimir Medvedec: developer
    - \* Jura Starčević: dokumentacija
    - \* Luka Radman: developer
    - \* Marko Štrk: dokumentacija
  - uvod u tehnologiju i arhitekturu koju bi koristili
  - rješavanje pitanja i nedoumica oko tehnologije i arhitekture
  - razrada strukture baze podataka
  - ostali dogовори

### 3. sastanak

- 05.11.2022., 10:00

- Prisustvovali: Fran Markulin, Branimir Medvedec, Luka Radman
- Teme sastanka:
  - rad na razvoju aplikacije

#### 4. sastanak

- 07.11.2022., 18:00
- Prisustvovali: Fran Markulin, Branimir Medvedec
- Teme sastanka:
  - rad na razvoju aplikacije

#### 5. sastanak

- 13.11.2022., 15:00
- Prisustvovali: Fran Markulin, Karla Udiljak, Branimir Medvedec, Jura Starčević, Luka Radman, Marko Štrk
- Teme sastanka:
  - dokumentacija - što još treba i kako dokumentirati
  - svaki član prezentirao svoj dio posla kako bi svi bili upoznati sa svime na projektu
  - dogovoreni idući koraci
  - podijela poslova
  - zadani rokovi

#### 6. sastanak

- 15.11.2022, 7:30
- Prisustvovali: Fran Markulin, Branimir Medvedec, Luka Radman
- Teme sastanka:
  - rad na razvoju aplikacije
  - priprema za prvu predaju

#### 7. sastanak

- 16.11.2022., 10:30
- Prisustvovali: Fran Markulin, Branimir Medvedec
- Teme sastanka:
  - komentiranje programskog koda
  - rad na dokumentaciji
  - priprema za izradu dijagrama razreda

#### 8. sastanak

- 17.11.2022., 13:45

- Prisustvovali: Laura Majer, Fran Markulin, Karla Udiljak, Branimir Medvedec, Jura Starčević, Luka Radman, Marko Štrk
- Teme sastanka:
  - demonstracija generičkih funkcionalnosti
  - rješavanje nedoumica

#### 9. sastanak

- 17.12.2022., 15:00
- Prisustvovali: Fran Markulin, Branimir Medvedec, Luka Radman
- Teme sastanka:
  - Programiranje

#### 10. sastanak

- 18.11.2022., 15:00
- Prisustvovali: Fran Markulin, Branimir Medvedec
- Teme sastanka:
  - Programiranje

## Tablica aktivnosti

	Fran Markulin	Branimir Medvedec	Luka Radman	Karla Uđiljak	Jura Starčević	Marko Štrk
Upravljanje projektom	48					
Opis projektnog zadatka					6	
Funkcionalni zahtjevi	1			1	1.5	
Opis pojedinih obrazaca						7
Dijagram obrazaca				4		
Sekvencijski dijagrami				6		
Opis ostalih zahtjeva						2.5
Arhitektura i dizajn sustava	3				4.5	
Baza podataka	3				3	
Dijagram razreda	6	2				2
Dijagram stanja				2.5		
Dijagram aktivnosti				2.5		
Dijagram komponenti				2.5		
Korištene tehnologije i alati					2	
Ispitivanje programskog rješenja					1	
4						
Dijagram razmještaja				1		
Upute za puštanje u pogon						1
Zaključak i budući rad						1
Popis literature	0.5				1	

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

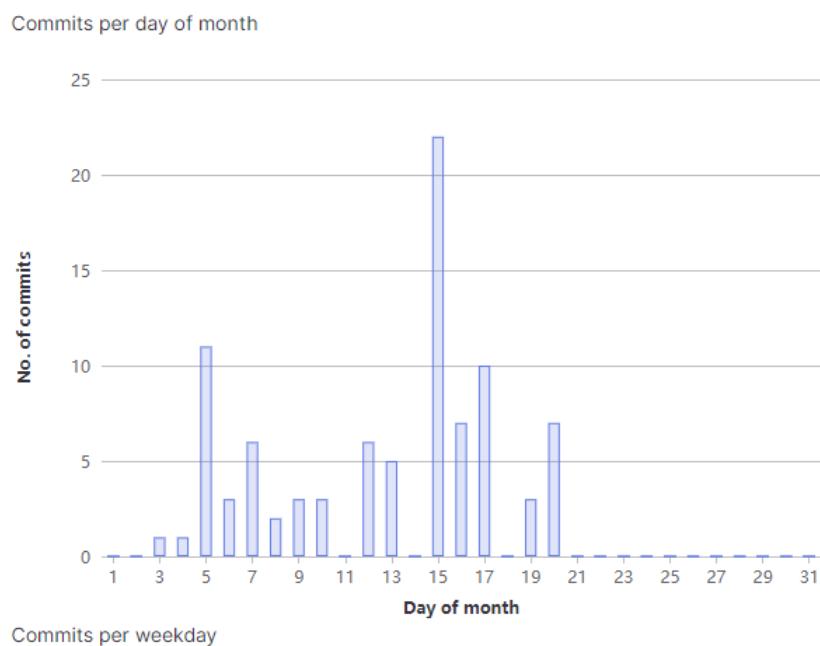
	Fran Markulin	Branimir Medvedec	Luka Radman	Karla Udiljak	Jura Starčević	Marko Štrk
Izrada login stranice	2	8				
Izrada register stranice	2	6	5			
Izrada komponente VlasnikForm	3	6	6			
Izrada komponente PrivatnaForm	3	4	3			
Izrada user info stranice	2	4	8			
Izrada komunikacije s bazom podataka	6					
Izrada konteksta	2					
Izrada komponente layout	1					
Izrada posebnih funkcija (hook.js)	1					
Izrada baze podataka	4		1			
Vođenje git repotizorija	9					
Dizajn				32		
Dnevnik sastajanja	1				1	
Moderiranje Latex dokumenta	1				35	1
Izrada Index stranice	24					
Izrada tests stranice	8					
Izrada PaymentForm		17				
Izrada Index-Home		7				

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

	Fran Markulin	Branimir Medvedec	Luka Radman	Karla Udiljak	Jura Starčević	Marko Štrk
Izrada UserInfo stranice			10			
Izrada ChangeInfo stranice			7			
Izrada ChangePassword stranice			6			

## Dijagrami pregleda promjena



Slika 6.1: Prikaz aktivnosti u projektu