

Programsko inženjerstvo

Ak. god. 2022./2023.

Dog Friendly

Dokumentacija, Rev. 2

Grupa: Simplicity

Voditelj: Timoteja Piveta

Datum predaje: 13. 1. 2023.

Nastavnik: Laura Majer, mag. ing.

Sadržaj

1 Dnevnik promjena dokumentacije	3
2 Opis projektnog zadatka	5
3 Specifikacija programske potpore	9
3.1 Funkcionalni zahtjevi	9
3.1.1 Obrasci uporabe	11
3.1.2 Sekvencijski dijagrami	23
3.2 Ostali zahtjevi	26
4 Arhitektura i dizajn sustava	27
4.1 Baza podataka	28
4.1.1 Opis tablica	28
4.1.2 Dijagram baze podataka	32
4.2 Dijagram razreda	33
4.3 Dijagram stanja	39
4.4 Dijagram aktivnosti	40
4.5 Dijagram komponenti	41
5 Implementacija i korisničko sučelje	42
5.1 Korištene tehnologije i alati	42
5.2 Ispitivanje programskog rješenja	45
5.2.1 Ispitivanje komponenti	45
5.2.2 Ispitivanje sustava	53
5.3 Dijagram razmještaja	63
5.4 Upute za puštanje u pogon	64
6 Zaključak i budući rad	78
Popis literature	80
Indeks slika i dijagonama	83

Dodatak: Prikaz aktivnosti grupe

84

1. Dnevnik promjena dokumentacije

Rev.	Opis promjene/dodatka	Autori	Datum
0.1	Napravljen predložak.	Timoteja Piveta	25.10.2022.
0.2	Dokumentiranje zahtjeva (opis projektnog zadatka).	Timoteja Piveta	26.10.2022.
0.3	Osvježavanje dnevnika sastanaka.	Timoteja Piveta	28.10.2022.
0.4	Napisani aktori i njihovi funkcionalni zahtjevi.	Mateo Hitl, Doma-goj Penava	28.10.2022.
0.5	Napravljeni obrasci uporabe i dijagrami obrazaca uporabe.	Mateo Hitl, Doma-goj Penava	29.10.2022.
0.6	Napravljeni sekvensijski dijagrami i ostali zahtjevi.	Mateo Hitl, Doma-goj Penava	30.10.2022.
0.7	Izrađene tablice u bazi podataka i ER dijagram baze.	Luka Novosel, Ni-kola Bukvić	02.11.2022.
0.7.1	Revizija specifikacije programske potpore.	Mateo Hitl, Doma-goj Penava	10.11.2022.
0.8	Popunjavanje dnevnika sastajanja i tablice aktivnosti.	Timoteja Piveta	14.11.2022.
0.9	Arhitektura i dijagrami razreda.	Bruno Perković	17.11.2022.
0.10	Popunjavanje tablice aktivnosti.	Nikola Bukvić	17.11.2022.
1.0	Revizija prve verzije dokumentacije.	Ana Žanko	18.11.2022.
1.1	Revizija opisa projektnog zadatka.	Timoteja Piveta	3.1.2023.

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

Rev.	Opis promjene/dodataka	Autori	Datum
1.2	Revizija obrazaca uporabe i sekveničijskih dijagrama.	Mateo Hitl, Domaškoj Penava	7.1.2023.
1.3	Napravljeni dijagrami stanja, aktivnosti i komponenti.	Mateo Hitl, Domaškoj Penava	9.1.2023.
1.4	Napravljen dijagram razmještaja	Mateo Hitl, Domaškoj Penava	12.1.2023.
1.5	Popravljeni dijagrami razreda i tablice baze podataka	Ana Žanko	12.1.2023.
1.6	Napisane korištene tehnologije i alati	Timoteja Piveta	12.1.2023.
1.7	Novi dijagram baze podataka i strani ključevi u tablicama	Ana Žanko	12.1.2023.
1.8	Napisan zaključak i budući rad	Mateo Hitl	13.1.2023.
1.9	Ispitivanje uvod i ispitivanje komponenti	Ana Žanko	13.1.2023.
1.10	Literatura, tablica aktivnosti i dnevnik promjena	Ana Žanko	13.1.2023.
1.11	Upute za puštanje u pogon	Mateo Hitl, Timoteja Piveta	13.1.2023.
1.11	Dijagrami pregleda promjena	Timoteja Piveta	13.1.2023.

2. Opis projektnog zadatka

Pronaći lokacije koje su dostoje i prikladne za čovjekovog najboljeg prijatelja nije uvijek lako i jednostavno. Stoga smo u cilju lakšeg pronaleta lokacija pogodnih za ljubimce razvili programsku podršku za web aplikaciju Dog Friendly.

Dog Friendly aplikacija pokazuje svojim korisnicima, vlasnicima i ljubiteljima pasa, korisne lokacije kao što su veterinarske ordinacije i frizerski saloni za pse, ali i lokacije koje nisu prikladne za ljubimce kako bi ih mogli lakše zaobići.

Na početnoj stranici korisnik saznaće informacije o ideji web stranice. Odmah ispod tog opisa nalazi se gumb "Karta" koji vodi korisnika na glavnu mogućnost web stranice, a to je prikaz gore navedenih lokacija. Pristup karti imaju registrirani i neregistrirani korisnici.

Svi korisnici (neregistrirani i registrirani) na karti mogu vidjeti lokacije koje su označene kao prikladne ili neprikladne za pse. Kako bi se korisniku olakšalo i ubrzalo traženje informacija pokraj karte su tražilica (za upis adrese i imena lokacije/obrta) i prostor za filtriranje lokacija i obrta po kategoriji.

Na karti se prikazuju markeri kao pokazatelji željenih i neželjenih lokacija. Pritisom miša na lokaciju označenu markerom korisnik može sazнати:

- ime lokacije
- ocjenu
- kategoriju (park, plaža, kafić, restoran, ostalo)

Registrirane korisnike dijelimo na osnovne korisnike i vlasnike obrta. Prilikom registracije korisniku se objašnjava razlika između navedenih i daje mogućnost odabira.

Obični korisnik, osim funkcionalnosti koje se prikazuju neregistriranim korisnicima, ima mogućnost dodavanja novih lokacija i ocjenjivanja. S druge strane, vlasnik obrta ne može ocjenjivati, ali može promovirati svoj obrt.

U slučaju da osoba odabere "Korisnik" (osnovni korisnik) popunjava formu sa sljedećim upitima:

- korisničko ime
- e-mail adresa
- lozinka

Pritiskom gumba "Registracija" pristiže joj e-mail koji sadrži link za potvrdu registracije. Klikom na link korisnika se vodi na stranicu za uspješnu registraciju te on time dobiva mogućnost prijave u aplikaciju.

U slučaju da korisnik prilikom registracije izabere "Vlasnik obrta" za kreiranje računa, uz već navedene podatke korisničkog imena, e-mail adrese i lozinke, potrebni su i sljedeći podaci:

- naziv obrta
- lokacija obrta (adresa i grad)
- OIB obrta
- kontakt broj
- kratki opis obrta
- vrsta obrta (veterinarska ordinacija, salon za pse, dućan za pse, vrtić, ostalo.)
- kartični podaci (broj kartice, datum isteka valjanosti i sigurnosni kod)

Vlasniku obrta se nakon registracije šalje link za potvrdu registracije kao i korisniku, ali dodatno i e-mail s potvrdom o uspješnom plaćanju.

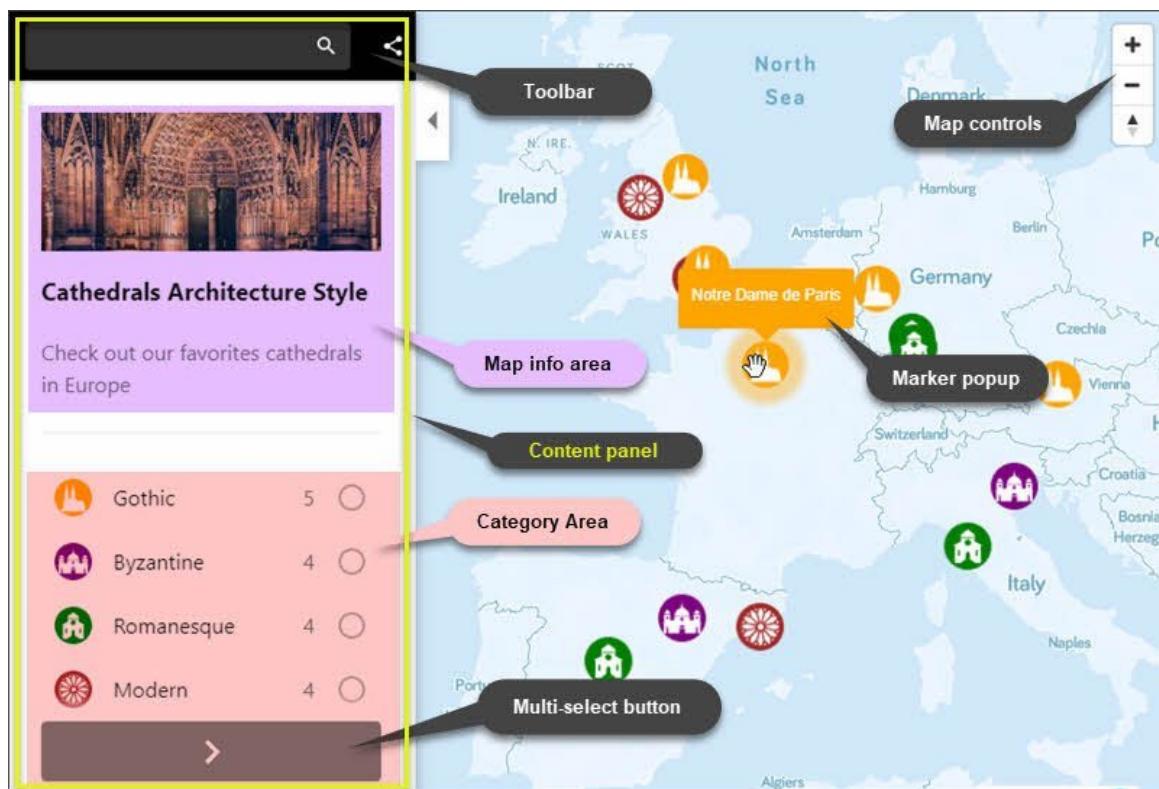
Kartični podaci vlasnika obrta potrebni su jer se aplikacija financira:

- preplatom vlasnika obrta
- plaćanjem dodatnog isticanja obrta

Vlasnik obrta plaća preplatu kako bi imao mogućnost objave svog obrta na Dog Friendly web aplikaciji i postavljanja istog obrta na kartu. Vlasnici obrta također mogu platiti da se njihov obrt dodatno istakne klijentima. U tom slučaju on će se nalaziti na posebnom popisu pokraj karte s drugim istaknutim obrtima.

Svim korisnicima omogućena je naknadna promjena podataka kao što su korisničko ime i lozinka. Nadalje, vlasnicima obrta također je omogućena promjena naziva i opisa obrta.

Prijavljeni korisnik, kao i neprijavljeni, ima pristup karti koja se centrirala ovisno o njegovoj trenutnoj lokaciji (uz prethodno dopuštenje) te obrima i lokacijama koje su ili povoljne ili nepovoljne za pse. Klikom na marker lokacije pojavljuje se prozorčić s detaljima kao što su ime lokacije, sveukupna ocjena i kategorija (plaža, kafić, restoran, itd.). Klikom na marker registriranog obrta moguće je vidjeti ime, opis, vrstu i kontakt broj obrta kao i kratki opis. Također je omogućen ručni unos adrese čime će se marker na karti pozicionirati na tu adresu te unos imena lokacije/obrta koji će se moći izabrati iz padajućeg izbornika te se zatim pozicionirati na karti. U slučaju da korisnika zanima određena vrsta obrta/lokacije korisnik će odabrat navedeno u odjelu za filtriranje te će mu se prikazati svi markeri navedene vrste kao što se vidi na slici 2.1. Pokraj karte je sekcija "Preporučeni obrti" u kojoj se klijentima preporučuje nekolicina obrta.



Slika 2.1: Primjer karte s tražilicom i prostorom za filtriranje.

Nakon što se prijavi, klijent može podijeliti svoje mišljenje o povoljnosti lokacije za pse. Pronalaskom željene lokacije (upis adrese ili imena u tražilicu) potvrđuje ili negira njezinu trenutnu oznaku, što ulazi u konačnu ocjenu te lokacije. Uz to,

korisnik može stvoriti novu lokaciju koju će ocijeniti.

Vlasnik obrta ima mogućnosti korisnika vezane za pregled karte, traženje lokacija i filtriranje, ali nema mogućnost ocjenjivanja postojećih ili stvaranja novih lokacija za ocjenjivanje. Nadalje, plaćanjem pretplate pri registraciji on objavljuje vlastiti obrt na karti kako bi postao vidljiv svim korisnicima aplikacije. Iz tog razloga nužno unosi sve podatke koje korisnici vide:

- naziv
- adresa
- grad
- kontakt broj
- kratki opis obrta
- vrsta obrta (veterinarska ordinacija, salon za pse, dućan za pse, itd.)

Kao povlasticu, vlasnik svoj obrt može dodatno promovirati plaćanjem naknade uz već postojeću pretplatu. Iстicanje se prezentira pojačavanjem markera na karti, ali i postavljanjem obrta u "Preporučeni obrti" prostoru gdje se u web aplikaciji preporučuju obrti vlasnika koji su dodatno uložili u isticanje. U slučaju da nema dodatno promoviranih obrta onda se u prostor "Preporučeni obrti" postavljaju najstariji obrti. U slučaju da obrti uopće ne postoje ispisuje se poruka "Trenutno nemamo preporučenih obrta."

Sustav podržava rad više korisnika (registriran, neregistrirani) u stvarnom vremenu.

3. Specifikacija programske potpore

3.1 Funkcionalni zahtjevi

Dionici:

1. Registrirani korisnik
 - (a) Korisnik
 - (b) Vlasnik obrta
2. Neregistrirani korisnik
3. Razvojni tim

Aktori i njihovi funkcionalni zahtjevi:

1. Neregistrirani/neprijavljeni korisnik (inicijator) može:
 - (a) na karti pregledati lokacije prikladne za pse
 - (b) odabrati marker i dobiti prikaz općih informacija za lokaciju (ime lokacije, ocjena, vrsta) ili obrta (ime obrta, adresa, OIB, kontakt-broj, kratki opis, djelatnost)
 - (c) vidjeti plaćene lokacije koje su dodatno istaknute
 - (d) unijeti specifičnu lokaciju na karti koja ga zatim centririra
 - (e) odabrati kategoriju čiji se markeri onda prikazuju
 - (f) stvoriti korisnički račun "Korisnik" za koji su mu potrebni korisničko ime, lozinka, e-mail
 - (g) stvoriti korisnički račun "Vlasnik obrta" za koji su mu potrebni korisničko ime, lozinka, e-mail te podaci o obrtu i kartici
2. Običan korisnik (inicijator) može:
 - (a) potvrditi e-mail o uspješnoj registraciji
 - (b) pregledavati i mijenjati osobne podatke
 - (c) izbrisati svoj korisnički račun
 - (d) podijeliti svoje mišljenje o povoljnosti lokacije za pse

- (e) unosom imena lokacije i odabirom njezine kategorije iz izbornika označiti lokaciju za koju želi iskazati mišljenje
- (f) stvoriti lokaciju koju će označiti prikladnom ili neprikladnom

3. Vlasnik obrta (inicijator) može:

- (a) potvrditi e-mail o uspješnoj registraciji
- (b) pregledavati i mijenjati osobne podatke
- (c) pregledavati i mijenjati podatke o svom obrtu
- (d) izbrisati svoj korisnički račun
- (e) promovirati svoj obrt uz novčanu naknadu
- (f) odgovoriti na recenzije korisnika

4. Baza podataka (sudionik):

- (a) pohranjuje sve podatke o korisnicima i njihovim ovlastima
- (b) pohranjuje sve podatke o obrtima
- (c) pohranjuje sve podatke o karticama

5. Karta (sudionik) može:

- (a) slati upit korisniku o njegovoj trenutnoj adresi
- (b) dohvatiti podatke iz baze podataka
- (c) prikazivati obrte i lokacije markerima
- (d) prikazivati informacije o lokacijama

3.1.1 Obrasci uporabe

Opis obrazaca uporabe

UC1 - Registracija

- **Glavni sudionik:** Neregistrirani korisnik
- **Cilj:** Stvoriti korisnički račun za pristup sustavu
- **Sudionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
 1. Neregistrirani korisnik u zaglavlju odabire opciju "Prijava"
 2. Odabire opciju "Registrirajte se"
 3. Odabire jednu od dvije mogućnosti registracije kao: korisnik ili vlasnik obrta
 4. Unosi tražene podatke (korisničko ime, e-mail, lozinka)
 - (a) Ako je odabrao opciju vlasnik obrta unosi dodatne podatke o obrtu i kartici
 5. Prima obavijest o uspješnoj registraciji putem e-maila
 - (a) Ako je odabrao opciju vlasnik obrta prima i e-mail s potvrdom o uspješnom plaćanju
- **Opis mogućih odstupanja:**
 - 2.a Odabir već zauzetog korisničkog imena/e-maila, unos podataka u nedozvoljenom formatu
 1. Sustav upozorava korisnika o neuspješnom unosu te ga vraća na stranicu za registraciju
 2. Korisnik mijenja potrebne podatke, završava unos ili odustaje od registracije

UC2 - Prijava u sustav

- **Glavni sudionik:** Registrirani korisnik
- **Cilj:** Dobiti pristup stranici
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je registriran
- **Opis osnovnog tijeka:**
 1. Korisnik u zaglavlju odabire opciju "Prijava"
 2. Unos korisničkog imena i lozinke

3. Verifikacija unesenih podataka
 4. Pristup korisničkim funkcijama
- **Opis mogućih odstupanja:**
 - 2.a Neispravno korisničko ime i/ili lozinka
 1. Sustav obaveštava korisnika o neuspješnoj prijavi

UC3 - Pregled osobnih podataka

- **Glavni sudionik:** Registrirani korisnik
- **Cilj:** Pregledati osobne podatke
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen
- **Opis osnovnog tijeka:**
 1. Korisnik u zaglavlju odabire opciju "Profil"
 2. Aplikacija prikazuje osobne podatke korisnika
 3. Prikazuju se ocjene koje je korisnik dodijelio lokacijama

UC4 - Promjena osobnih podataka

- **Glavni sudionik:** Registrirani korisnik
- **Cilj:** Promijeniti osobne podatke
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen
- **Opis osnovnog tijeka:**
 1. Korisnik u zaglavlju odabire opciju "Profil"
 2. Odabire opciju "Uredi profil"
 3. Mijenja osobne podatke
 4. Sprema promjene
 5. Baza podataka se ažurira
- **Opis mogućih odstupanja:**
 - 3.a Korisnik promijeni svoje osobne podatke, ali ne odabere opciju "Spremi promjene"
 1. Sustav upozorava korisnika da nije spremio podatke prije izlaska iz prozora
 - 3.b Upis već zauzetog korisničkog imena
 1. Sustav upozorava korisnika o neuspješnoj promjeni
 2. Korisnik mijenja potrebne podatke, završava unos ili odustaje od promjene podataka

UC5 - Brisanje korisničkog računa

- **Glavni sudionik:** Registrirani korisnik
- **Cilj:** Izbrisati svoj korisnički račun
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen
- **Opis osnovnog tijeka:**
 1. Korisnik u zaglavljku odabire opciju "Profil"
 2. Odabire opciju "Izbriši profil"
 3. Sustav upozorava korisnika je li siguran da želi izbrisati korisnički račun
 4. Korisnički račun se briše iz baze podataka
 - (a) Ako je korisnik ocijenio lokacije te ocjene brišu se zajedno s računom
 5. Korisnik se preusmjerava na početnu stranicu

UC6 - Dodavanje obrta

- **Glavni sudionik:** Vlasnik obrta
- **Cilj:** Dodati obrt prilikom registracije
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je u tijeku registracije
- **Opis osnovnog tijeka:**
 1. Tijekom registracije korisnik upisuje podatke o obrtu
 2. Nakon registracije baza podataka se ažurira
- **Opis mogućih odstupanja:**
 - 1.a Odabirom već zauzetog imena obrta i/ili već zauzetog OIB-a obrta, unos OIB-a obrta u nedozvoljenom formatu
 1. Sustav upozorava vlasnika o neuspješnom unosu
 2. Vlasnik mijenja potrebne podatke, završava unos ili odustaje od registracije

UC7 - Dodavanje kartice

- **Glavni sudionik:** Vlasnik obrta
- **Cilj:** Dodati karticu prilikom registracije
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je u tijeku registracije
- **Opis osnovnog tijeka:**
 1. Tijekom registracije korisnik upisuje podatke o kartici

2. Nakon registracije baza podataka se ažurira
- **Opis mogućih odstupanja:**
 - 1.a Odabirom nevažeće kartice (datum isteka kartice)
 1. Sustav upozorava vlasnika o neuspješnom unosu
 2. Vlasnik mijenja potrebne podatke, završava unos ili odustaje od registracije

UC8 - Pregled vlastitog obrta

- **Glavni sudionik:** Vlasnik obrta
- **Cilj:** Pregledati podatke o obrtu
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je vlasnik obrta
- **Opis osnovnog tijeka:**
 1. Vlasnik u zaglavlju odabire opciju "Profil"
 2. Pregledava podatke o obrtu

UC9 - Promjena podataka o obrtu

- **Glavni sudionik:** Vlasnik obrta
- **Cilj:** Promijeniti podatke o obrtu
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je vlasnik obrta
- **Opis osnovnog tijeka:**
 1. Vlasnik u zaglavlju odabire opciju "Profil"
 2. Mijenja podatke o obrtu
 3. Potvrđuje unesene podatke
 4. Baza podataka se ažurira
- **Opis mogućih odstupanja:**
 - 3.a Upis već zauzetog imena obrta
 1. Sustav upozorava vlasnika o neuspješnoj promjeni
 2. Vlasnik mijenja potrebne podatke, završava unos ili odustaje od promjene podataka

UC10 - Prikaz karte

- **Glavni sudionik:** Korisnik
- **Cilj:** Prikazati kartu
- **Sudionici:** Baza podataka, Karta

- **Preduvjet:** -
- **Opis osnovnog tijeka:**
 1. Korisnik u zaglavlju odabire opciju "Karta"
 2. Može pregledati lokacije na karti

UC11 - Prikaz trenutne lokacije

- **Glavni sudsionik:** Korisnik
- **Cilj:** Prikazati trenutnu lokaciju
- **Sudsionici:** Karta
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
 1. Korisnik u zaglavlju odabire opciju "Karta"
 2. Pojavljuje se prozor u kojem korisnik dopušta očitanje svoje lokacije
 3. Karta se centrira na korisnikovu lokaciju
- **Opis mogućih odstupanja:**
 - 2.a Korisnik nije dopustio očitavanje svoje lokacije
 1. Karta se ne centrira na korisnikovu lokaciju

UC12 - Pretraživanje lokacije/obrta po imenu

- **Glavni sudsionik:** Korisnik
- **Cilj:** Pretražiti lokacije/obrta po imenu
- **Sudsionici:** Baza podataka, Karta
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
 1. Korisnik u zaglavlju odabire opciju "Karta"
 2. Upisuje ime lokacije/obrta u polje za pretragu
 3. Na karti se centrira željena lokacija

UC13 - Filtriranje po kategorijama

- **Glavni sudsionik:** Korisnik
- **Cilj:** Filtrirati lokacije po kategorijama
- **Sudsionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
 1. Korisnik u zaglavlju odabire opciju "Karta"
 2. Odabire kategorije za filtriranje lokacija

3. Prikazuju se filtrirane lokacije na karti

UC14 - Pregled lokacije

- **Glavni sudionik:** Korisnik
- **Cilj:** Pregledati opis lokacije
- **Sudionici:** Baza podataka, Karta
- **Preduvjet:** Lokacija postoji
- **Opis osnovnog tijeka:**
 1. Korisnik u zaglavlju odabire opciju "Karta"
 2. Korisnik na karti odabire lokaciju
 3. Pojavljuje se prozor u kojem može saznati više informacija o lokaciji (ime, ocjena i kategorija)

UC15 - Pregled obrta

- **Glavni sudionik:** Korisnik
- **Cilj:** Pregledati obrt
- **Sudionici:** Baza podataka, Karta
- **Preduvjet:** Obrt postoji
- **Opis osnovnog tijeka:**
 1. Korisnik u zaglavlju odabire opciju "Karta"
 2. Korisnik na karti odabire obrt
 3. Pojavljuje se prozor u kojem može saznati više informacija o obrtu (naziv, vrsta, opis i kontakt)

UC16 - Dodjela prikladnosti lokacije

- **Glavni sudionik:** Registrirani korisnik
- **Cilj:** Dodijeliti oznaku prikladnosti lokaciji
- **Sudionici:** Baza podataka, Karta
- **Preduvjet:**
 - Lokacija postoji
 - Korisnik nije još dodijelio prikladnost za tu lokaciju
- **Opis osnovnog tijeka:**
 1. Korisnik u zaglavlju odabire opciju "Karta"
 2. Korisnik na karti odabire lokaciju
 3. Lokaciji dodjeljuje oznaku prikladnosti
 4. Sprema oznaku lokacije

5. Baza podataka se ažurira

UC17 - Promjena prikladnosti lokacije

- **Glavni sudionik:** Registrirani korisnik
- **Cilj:** Promijeniti oznaku prikladnosti lokacije
- **Sudionici:** Baza podataka, Karta
- **Preduvjet:**
 - Lokacija postoji
 - Lokacija već ima dodijeljenu oznaku prikladnosti od korisnika
- **Opis osnovnog tijeka:**
 1. Korisnik u zaglavlju odabire opciju "Profil"
 2. Lokaciji mijenja oznaku prikladnosti
 3. Baza podataka se ažurira

UC18 - Pregled prikladnosti lokacije

- **Glavni sudionik:** Korisnik
- **Cilj:** Pregledati prikladnost obrta
- **Sudionici:** Baza podataka, Karta
- **Preduvjet:** Lokacija postoji
- **Opis osnovnog tijeka:**
 1. Korisnik u zaglavlju odabire opciju "Karta"
 2. Korisnik na karti odabire lokaciju
 3. Prikazuje se prikladnost odabrane lokacije

UC19 - Prikaz preporučenih obrta

- **Glavni sudionik:** Korisnik
- **Cilj:** Prikazati preporučene obrte
- **Sudionici:** Baza podataka
- **Preduvjet:** Postoje pretplaćeni obrti
- **Opis osnovnog tijeka:**
 1. Korisnik u zaglavlju odabire opciju "Karta"
 2. Prikazuju se preporučeni obrti

UC20 - Promocija obrta

- **Glavni sudionik:** Vlasnik
- **Cilj:** Promovirati obrt

- **Sudionici:** Baza podataka
- **Preduvjet:** Obrt postoji
- **Opis osnovnog tijeka:**
 1. Vlasnik u zaglavlju odabire opciju "Profil"
 2. Prikazuju se podatci o obrtu
 3. Odabire trajanje promocije obrta
 4. Potvrđuje plaćanje za promociju obrta
 5. Plaća za promociju obrta
- **Opis mogućih odstupanja:**
 - 4.a Vlasnik nije potvrdio plaćanje
 1. Obrt se ne promovira

UC21 - Pregled istaknutih obrta na karti

- **Glavni sudionik:** Korisnik
- **Cilj:** Pregledati istaknute obrte
- **Sudionici:** Baza podataka, Karta
- **Preduvjet:**
 - Obrt postoji
 - Vlasnik je platio za dodatno isticanje
- **Opis osnovnog tijeka:**
 1. Korisnik u zaglavlju odabire opciju "Karta"
 2. Na karti se dodatno ističu markeri koji predstavljaju pretplaćene obrte
- **Opis mogućih odstupanja:**
 - 2.a Nema pretplaćenih obrta
 1. Na karti se ne prikazuju dodatno istaknuti markeri

UC22 - Dodavanje lokacije

- **Glavni sudionik:** Registrirani korisnik
- **Cilj:** Dodati lokaciju
- **Sudionici:** Baza podataka, Karta
- **Preduvjet:** Korisnik je registriran
- **Opis osnovnog tijeka:**
 1. Korisnik u zaglavlju odabire opciju "Karta"
 2. Korisnik na karti pronađe lokaciju koju želi dodati
 3. Unosi podatke o lokaciji(ime, vrsta i prikladnost)
 4. Podaci se spremaju

5. Lokacija se dodaje u bazu podataka
- **Opis mogućih odstupanja:**
 - 2.a Korisnik dodaje lokaciju koja već postoji
 1. Sustav upozorava korisnika da ta lokacija već postoji

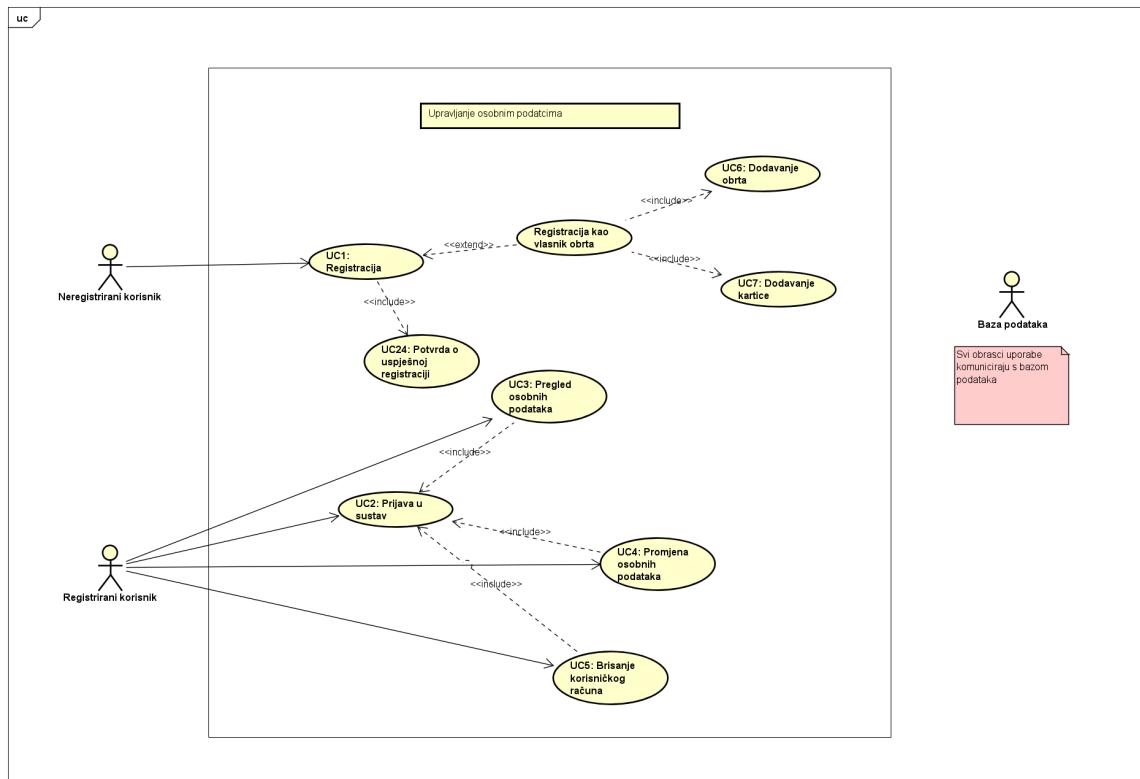
UC23 - Brisanje lokacije

- **Glavni sudionik:** Korisnik
- **Cilj:** Obrisati lokaciju
- **Sudionici:** Baza podataka, Karta
- **Preduvjet:** Lokacija postoji
- **Opis osnovnog tijeka:**
 1. Korisnik briše prikladnost lokacije
 2. Ako lokacija nema niti jednu prikladnost automatski se briše iz baze podataka

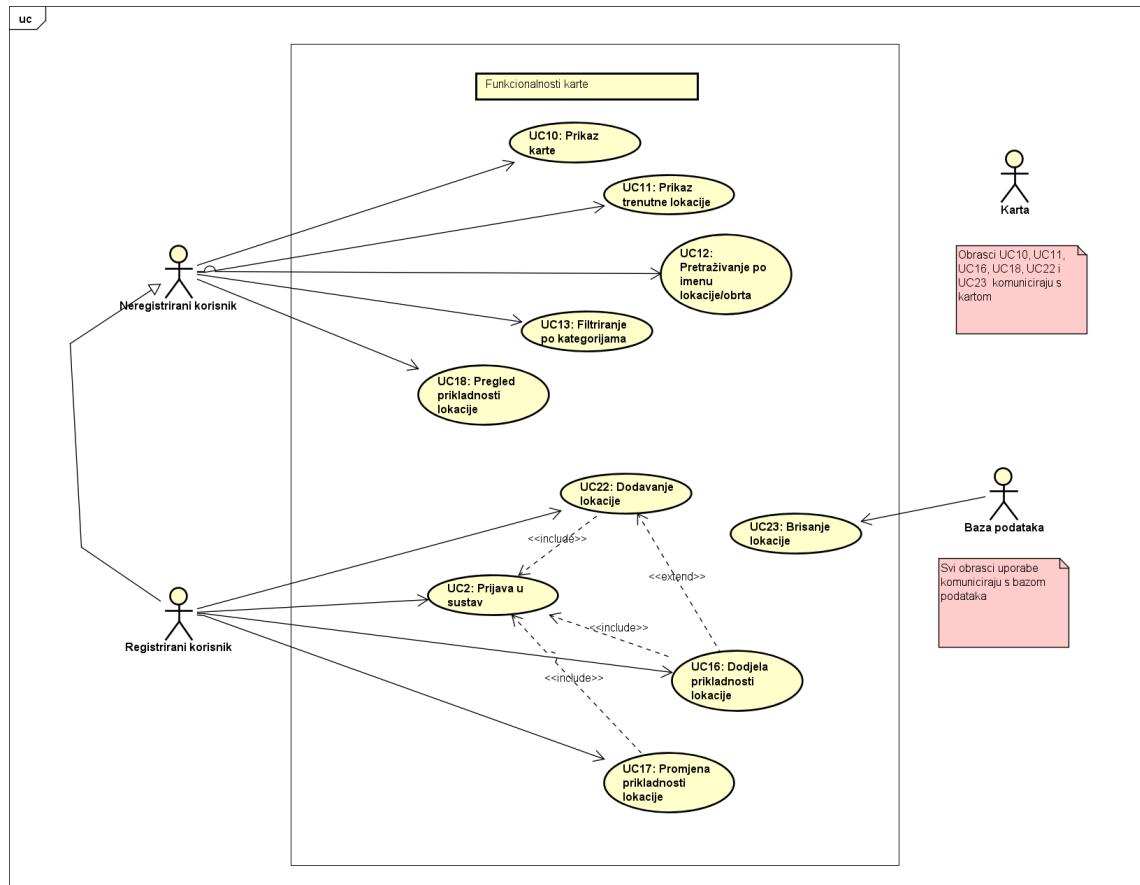
UC24 - Potvrda o uspješnoj registraciji

- **Glavni sudionik:** Korisnik
- **Cilj:** Potvrditi registraciju
- **Sudionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
 1. Nakon uspješne registracije, korisniku se šalje e-mail s linkom za potvrdu registracije računa
 2. Korisnik zatim odlazi na svoj e-mail i potvrđuje registraciju pritiskom na link
 3. Podatak o aktivaciji računa spremi se u bazu podataka
 4. Korisnika se obavještava o aktivaciji računa

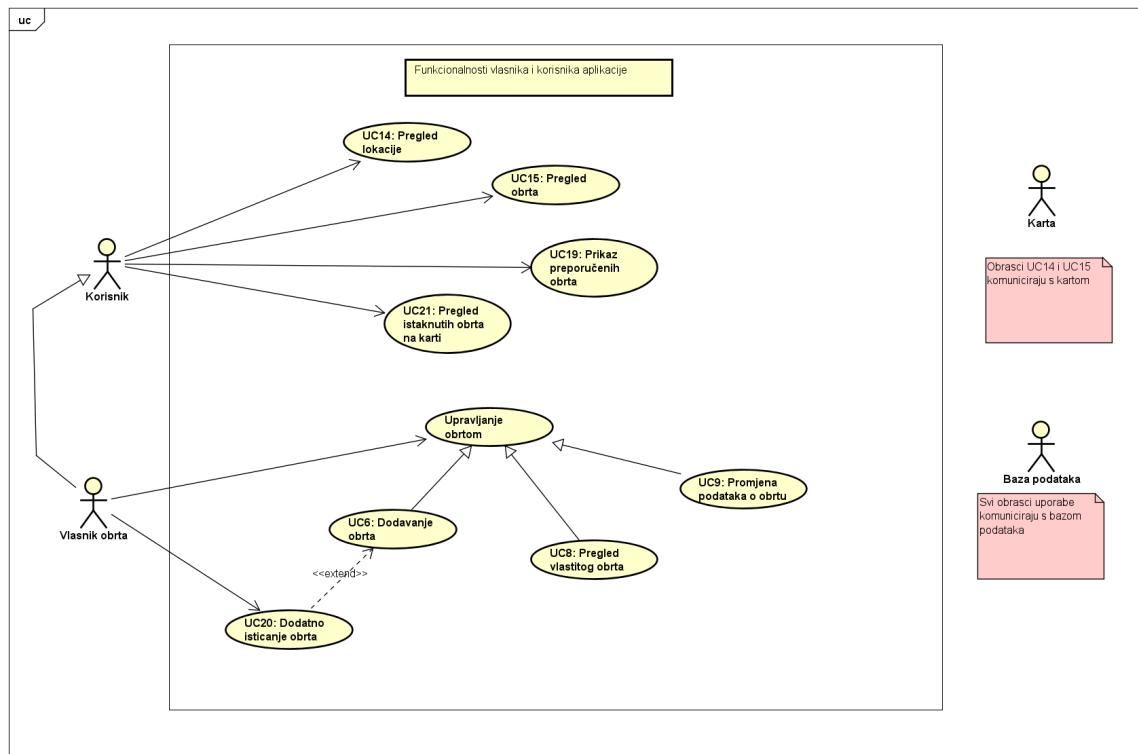
Dijagrami obrazaca uporabe



Slika 3.1: Dijagram obrasca uporabe, upravljanje osobnim podatcima



Slika 3.2: Dijagram obrasca uporabe, funkcionalnost karte

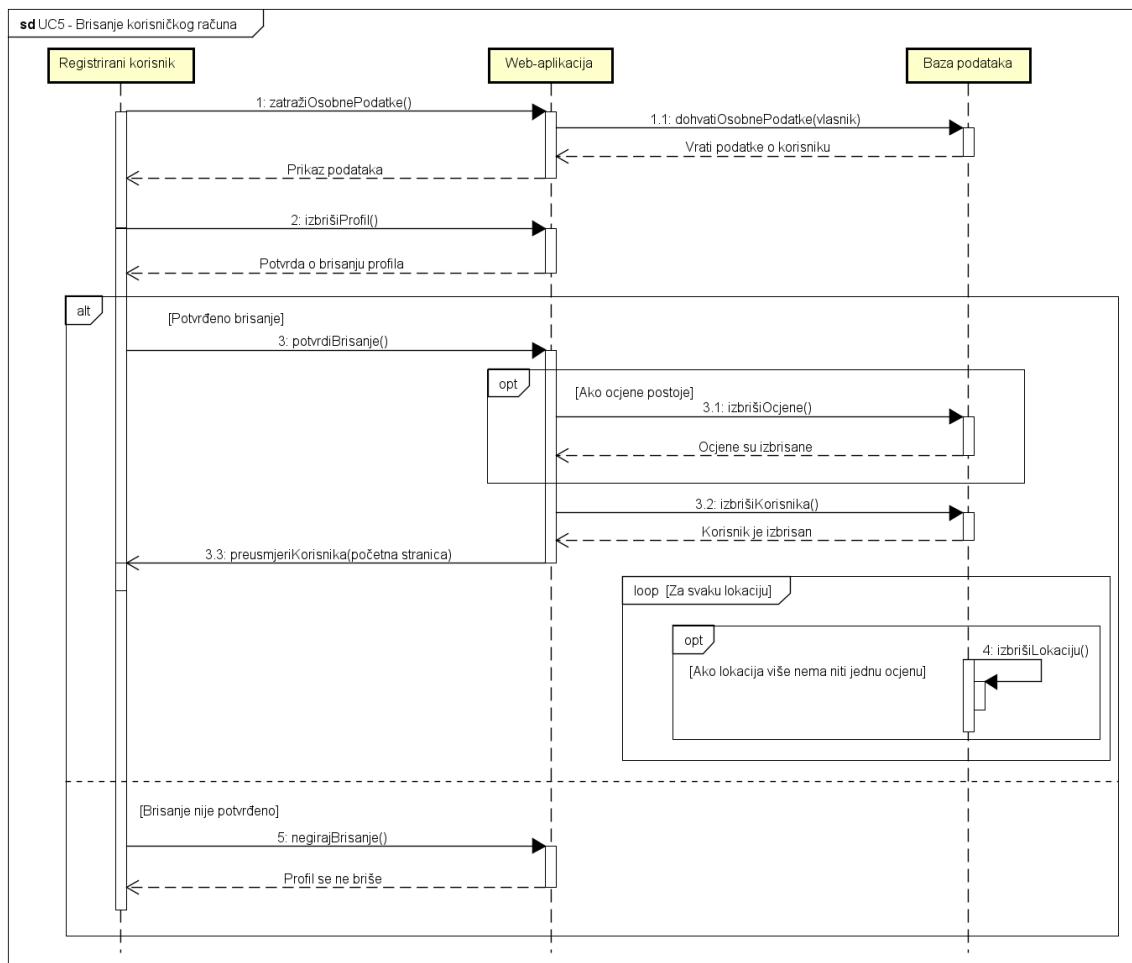


Slika 3.3: Dijagram obrasca uporabe, funkcionalnost korisnika i vlasnika obrta

3.1.2 Sekvencijski dijagrami

Obrazac uporabe UC5 – Brisanje korisničkog računa

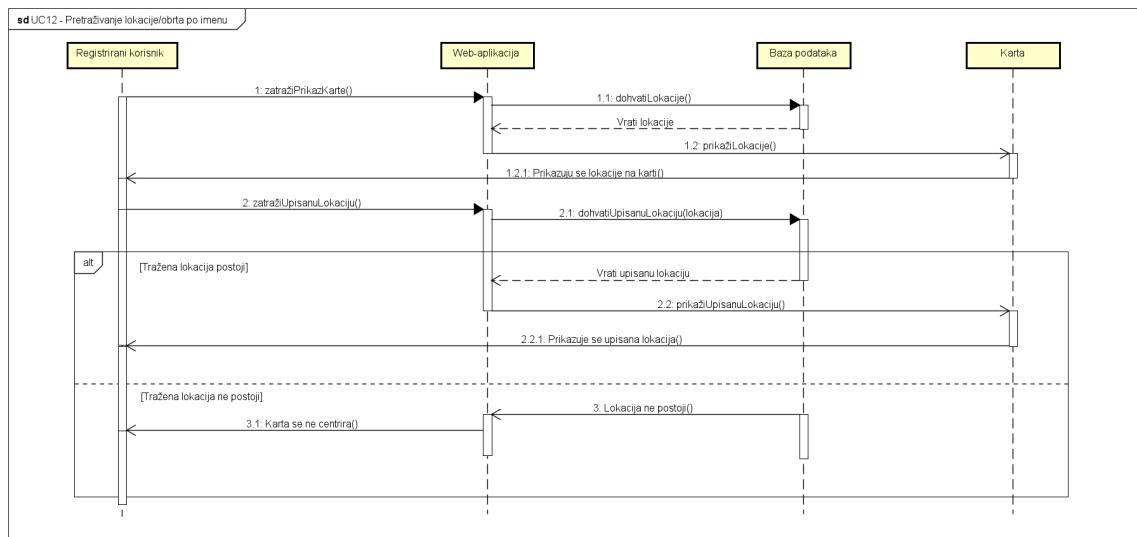
Registrirani korisnik odlazi na svoj profil gdje mu se prikazuju vlastiti podatci. Kako bi obrisao svoj profil korisnik mora poslati zahtjev za brisanjem računa i potvrditi da je siguran da ga želi obrisati. Ako je potvrdio brisanje onda se odmah brišu i sve njegove ocjene te ga aplikacija preusmjerava na početnu stranicu. Zatim se još u bazi podataka brišu sve lokacije koje više nemaju niti jednu ocjenu.



Slika 3.4: Sekvencijski dijagram za UC5

Obrazac uporabe UC12 – Pretraživanje lokacije/obrta po imenu

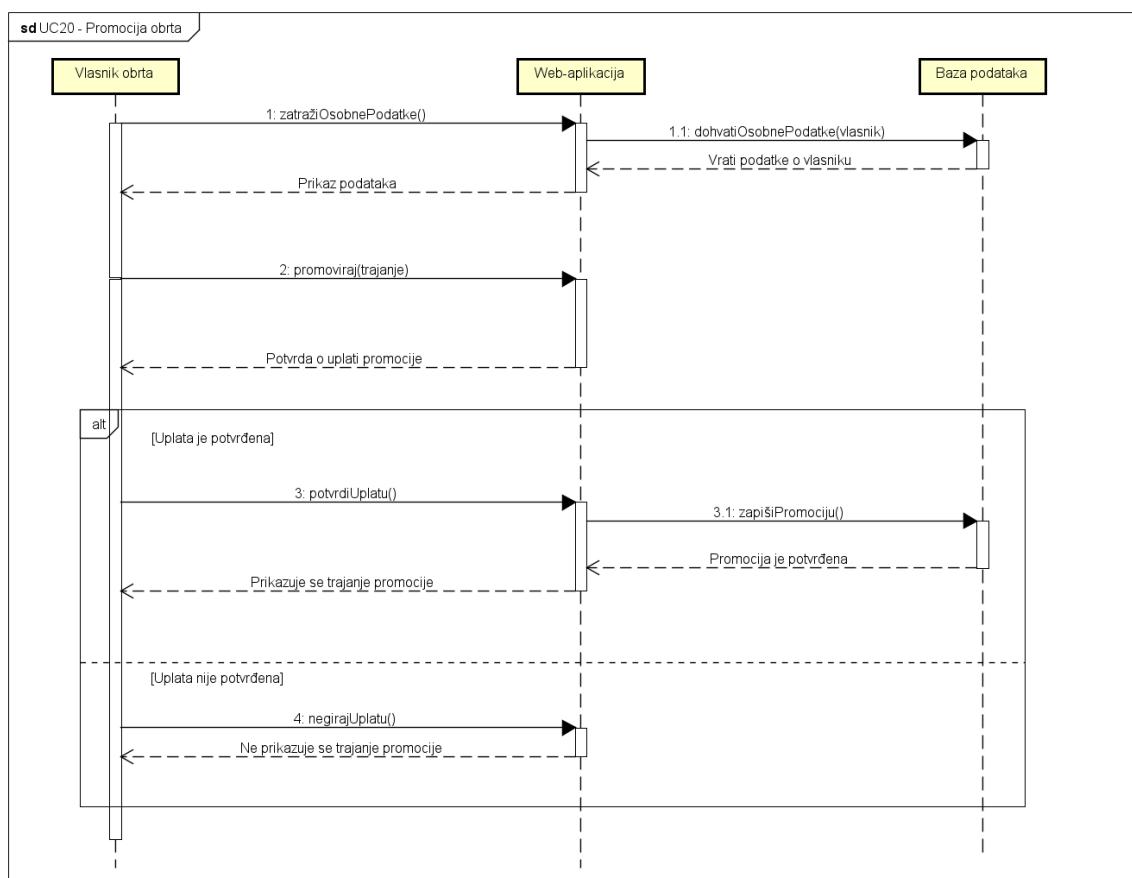
Registrirani korisnik odlazi na kartu te mu baza podataka vraća sve lokacije, a karta ih onda prikazuje. Korisnik zatim upisuje lokaciju u tražilicu te mu baza podataka vraća rezultat pretrage. Ako je lokacije postojeca onda mu se karta centriira na traženu lokaciju. Uprotivnom karta se ne centriра.



Slika 3.5: Sekvencijski dijagram za UC12

Obrazac uporabe UC20 – Promocija obrta

Vlasnik obrta odlazi na svoj profil gdje mu se prikazuju vlastiti podatci. Odabire trajanje promocije te potvrđuje je li siguran da želi platiti za promociju obrta. Ako je potvrdio promociju trajanje promocije prikazuje mu se pod podatima o obrtu, u bazi podataka sprema se datum promocije i trajanje. U suprotnom mu se prikazuje da obrt nije promoviran.



Slika 3.6: Sekvencijski dijagram za UC20

3.2 Ostali zahtjevi

- Aplikacija mora biti izvedena kao web aplikacija kojoj će korisnici pristupati uz pomoć korisničkog imena i lozinke
- Oblikovanje aplikacije mora slijediti načela objektno-orientiranog programiranja
- Aplikacija mora biti prilagođena za različite veličine ekrana
- Unos lokacija mora podržavati dijakritičke znakove hrvatske abecede
- Aplikacija mora omogućiti rad više korisnika u stvarnom vremenu
- Sustav kao valutu koristi euro
- Aplikacija treba biti jednostavna za korištenje, a sučelje pregledno i intuitivno
- Pristup bazi podataka ne bi trebao trajati duže od 5 sekundi

4. Arhitektura i dizajn sustava

Arhitektura aplikacije sastoji se od **web poslužitelja**, **web aplikacije** i **baze podataka** te predstavlja ključni dio proizvoda. Ovaj tip arhitekture omogućuje razdvajanje odgovornosti na tri glavna dijela sustava, što pomaže u poboljšanju performansi, skalabilnosti i sigurnosti aplikacije.

Web poslužitelj je dio arhitekture koji se bavi komunikacijom između korisnika i web aplikacije. On prima zahtjeve korisnika putem HTTP protokola te ih prosljeđuje web aplikaciji za obradu. Web poslužitelj također služi za hosting web aplikacije i omogućuje joj pristup bazi podataka.

Web aplikacija je komponenta koja se bavi obradom zahtjeva korisnika i generiranjem odgovora. Ona se sastoji od **kontrolera**, **servisa** i **repozitorija**. Kontroler služi za prijem zahtjeva te prosljeđivanje prema odgovarajućim servisima. Servisi su oni koji sadrže većinu funkcionalnosti aplikacije te djeluju kao veza između kontrolera i repozitorija. Repozitoriji su oni koji se bave komunikacijom s bazom podataka i dohvaćanjem potrebnih podataka za obradu zahtjeva.

Baza podataka je komponenta koja se bavi pohranjivanjem i upravljanjem podacima aplikacije. Ona omogućuje web aplikaciji pristup podacima koji su potrebni za obradu zahtjeva korisnika. Baza podataka također služi za pohranu podataka koji se generiraju tijekom rada aplikacije.

Ovaj tip arhitekture omogućuje lakše održavanje, razvoj i skalabilnost aplikacije. Razdvajanjem odgovornosti na tri glavna dijela sustava omogućava se lakše testiranje i razvoj svakog dijela posebno. Baza podataka se može lako promijeniti bez utjecaja na ostatak sustava, a web aplikacija se može razvijati i testirati neovisno od web poslužitelja.

Za izradu web aplikacije koristimo programske jezike Typescript (React library) i Java (Spring Framework). Za bazu podataka izabrali smo PostgreSQL.

4.1 Baza podataka

Kako bi spremili sve potrebne podatke koristit ćemo relacijsku bazu podataka. Osim spremanja podataka, relacijska baza podataka će biti odličan alat za izmjenu, brisanje i dohvata podataka. Objekte u bazi ćemo prikazivati u obliku relacija, građivnih jedinica baze podataka. Naša baza podataka sastojat će se od sljedećih relacija:

- User
- Owner
- Business
- Card
- Location
- UserRating

4.1.1 Opis tablica

Entitet **User** (tablica nazvana "webusers") sadržava informacije o korisniku aplikacije. Informacije koje korisnik upisuje su korisničko ime, email adresa i lozinka. Ovisno o vrsti registracije (običan korisnik ili vlasnik obrta) naš sustav korisniku dodjeljuje ulogu USER ili OWNER. Tablica sadrži i informaciju je li korisnički račun aktiviran putem linka poslanog na upisanu e-mail adresu. Entitet User je generalizacija te idući entitet Owner nasljeđuje klasu User. Ovaj entitet je u *OneToOne* relaciji s entitetom UserRating.

User		
id	BIGINT	Primarni generirani ključ korisnika

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

User		
account_activated_by_email	BOOLEAN	Odgovara na pitanje je li korisnik aktivirao svoj korisnički račun linkom poslanim na e-mail
email	VARCHAR(255)	E-mail korisnika
password	VARCHAR(255)	Lozinka računa
username	VARCHAR(255)	Korisničko ime
role	INTEGER	Vrsta korisničkog računa

Relacija **Owner** sadržavat će informacije o korisniku aplikacije koji je ujedno i vlasnik obrta. Entitet Owner je u relaciji *OneToOne* sa entitetima Business i Card. U tablici se nalaze id korisnika (naslijедeno od entiteta User) te reference na entitete Business i Card. Poveznica se obavlja preko id-a oba identiteta.

Owner		
id	BIGINT	Primarni generirani ključ vlasnika korisnika (vlasnika obrta)
user_business	BIGINT	Obrt korisnika
user_card	BIGINT	Kartica korisnika

Relacija **Business** sadržavat će informacije o obrtu koji pripada nekom vlasniku. Informacije o obrtu su naziv obrta, adresa, grad, OIB obrta, službeni kontakt broj, opis i tip obrta. Obrt se može promovirati te se ti podaci ujedno nalaze i ovoj tablici. Entitet je u *OneToOne* relaciji s entitetom Owner.

Business		
id	BIGINT	Primarni ključ obrta

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

Business		
business_oib	VARCHAR(11)	OIB obrta
business_name	VARCHAR(255)	Ime obrta
business_address	VARCHAR(255)	Adresa obrta
business_city	VARCHAR(255)	Grad obrta
business_mobile_number	VARCHAR(255)	Kontakt broj
business_description	VARCHAR(255)	Opis obrta
business_type	INTEGER	Tip obrta
promotion_duration	VARCHAR(255)	Vrsta promocije
promotion_start	DATE	Datum početka promocije

Relacija **Card** sadržavat će informacije o kartici koja pripada nekom vlasniku obrta. Informacije o kartici su broj kartice, kontrolni broj te datum isteka kartice. Enitet je u *OneToOne* relaciji s entitetom Owner.

Card		
id	BIGINT	Primarni ključ kartice
card_number	VARCHAR(255)	Broj kartice
cvv	VARCHAR(4)	Kontrolni broj (CVV)
end_date	TIMESTAMP	Datum isteka valjanosti kartice

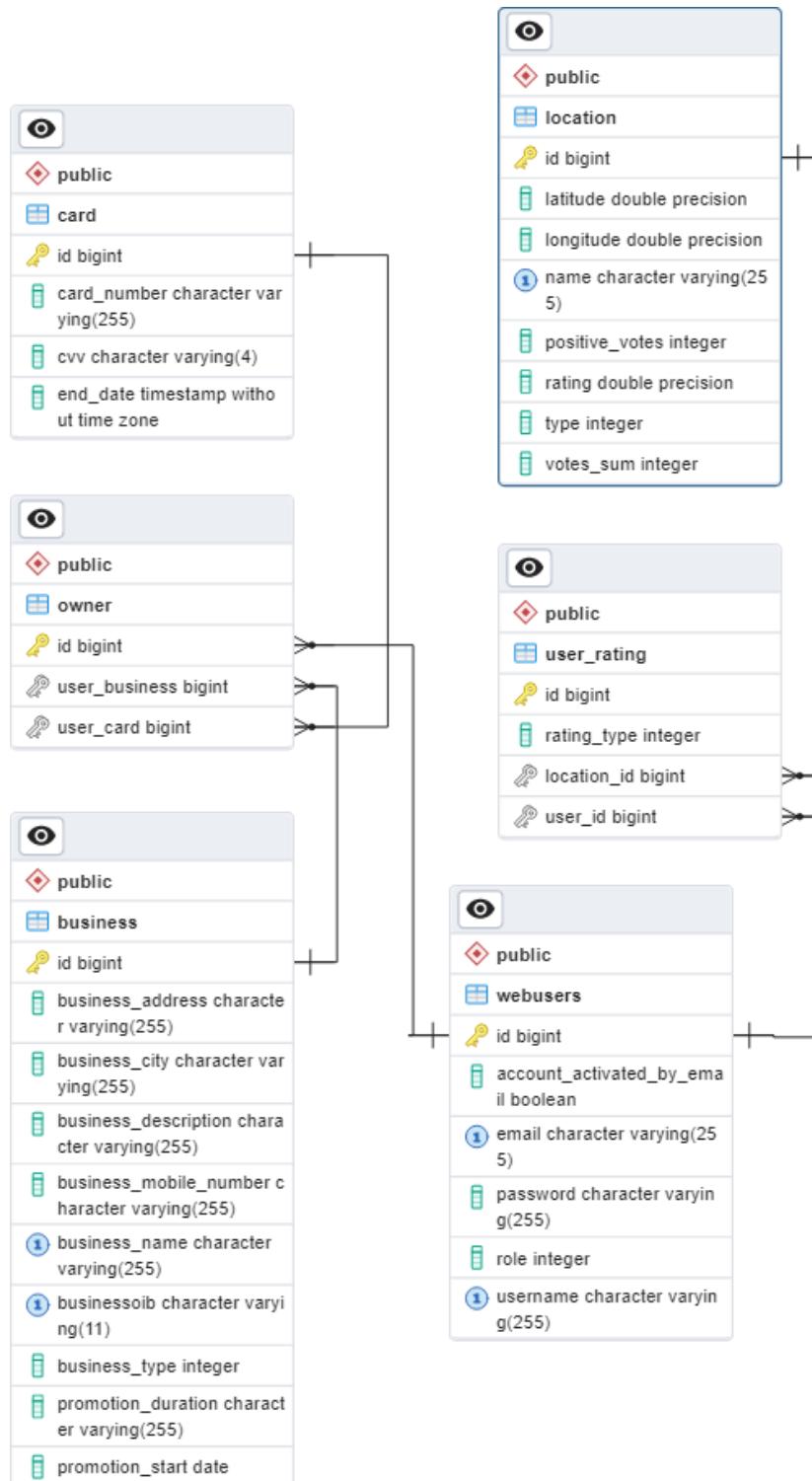
Relacija **Location** sadržavat će informacije o lokacijama na karti. Informacije o lokaciji su ime, vrsta, njena geografska širina i dužina koje se automatski stvaraju kada korisnik odabere mjesto lokacije na karti. Nadalje, spremaju se i ocjene lokacije koja se izračunava pomoću broja pozitivnih ocjena i ukupnog broja ocjena za zadanu lokaciju. Enitet je u *OneToOne* relaciji s entitetom UserRating.

Location		
id	BIGINT	Primarni ključ lokacije
latitude	DOUBLE PRECISION	Geografska širina
longitude	DOUBLE PRECISION	Geografska dužina
name	VARCHAR(255)	Ime lokacije
type	INTEGER	Vrsta lokacije
positive_votes	INTEGER	Broj pozitivnih ocjena
votes_sum	INTEGER	Ukupni broj ocjena
rating	DOUBLE PRECISION	Konačna ocjena lokacije

Relacija **UserRating** sadržavat će informacije o ocjenama koje je neki korisnik dao nekoj lokaciji. Informacije koje se zapisuju su korisnik, lokacija i vrsta ocjene koju je korisnik dodijelio. Korisnik i lokacija referencirani su pomoću njihovih primarnih ključeva. Enitet je u *OneToOne* relaciji s entitetima User i Location.

UserRating		
id	BIGINT	Primarni ključ ocjene
location_id	BIGINT	Id korisnika koji je ocijenio
user_id	BIGINT	Id lokacije koja je ocijenjena
rating_type	INTEGER	Dodijeljena ocjena

4.1.2 Dijagram baze podataka



Slika 4.1: Dijagram baze podataka

4.2 Dijagram razreda

Na slikama 4.2-4.7 nalaze se dijagrami razreda koji pripadaju web aplikaciji.



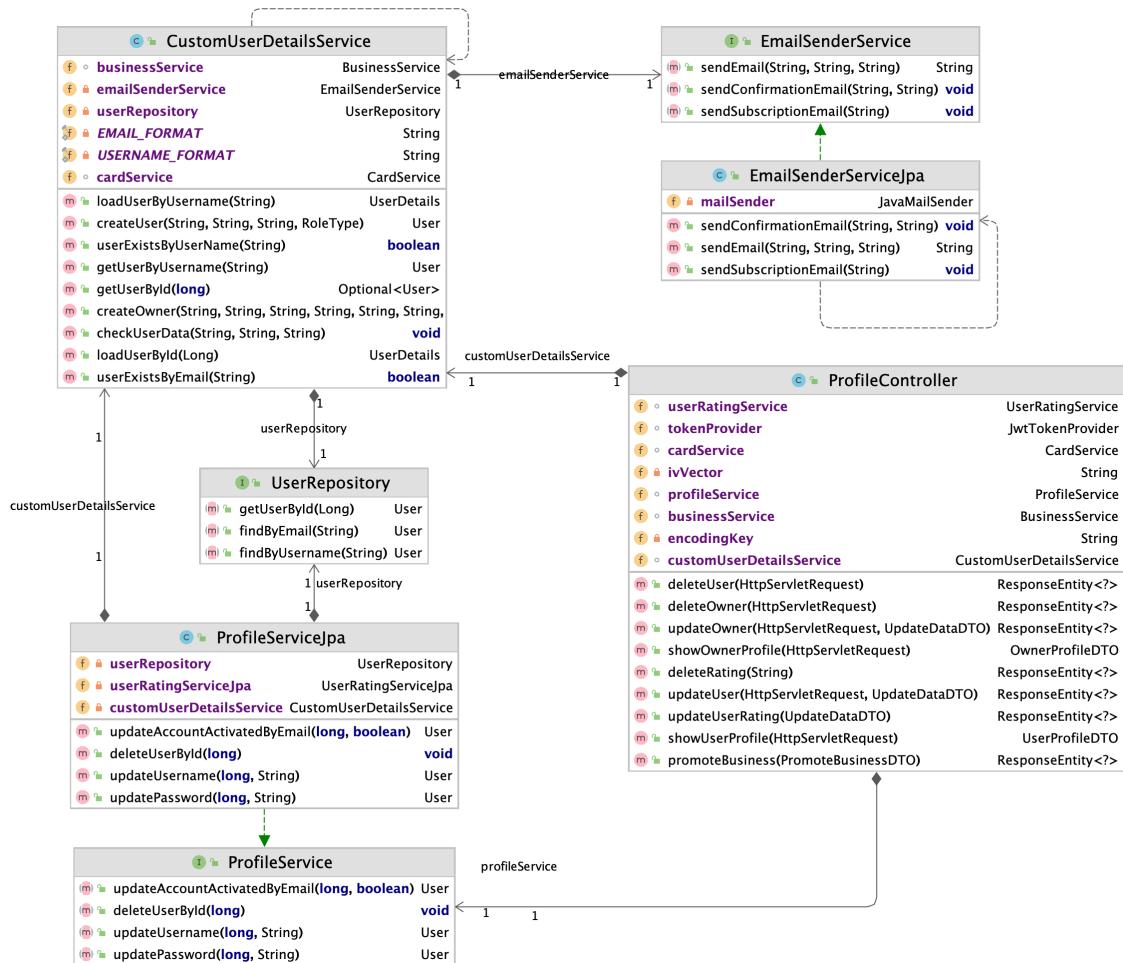
Slika 4.2: Dijagram klasa i metoda - Objekti za slanje podataka i controlleri

Prikazuje pomoćne DTO (Data Transfer Object) klase koje služe kao okvir unutar kojega se mogu slati zahtjevi i odgovori na zahtjeve. Primjerice, za zahtjeve možemo vidjeti da razred `AuthController` prima `LoginRequestDTO`. S druge strane, za odgovore možemo vidjeti da razrede `UserProfileDTO` i `OwnerProfileDTO` stvara `ProfileController`.



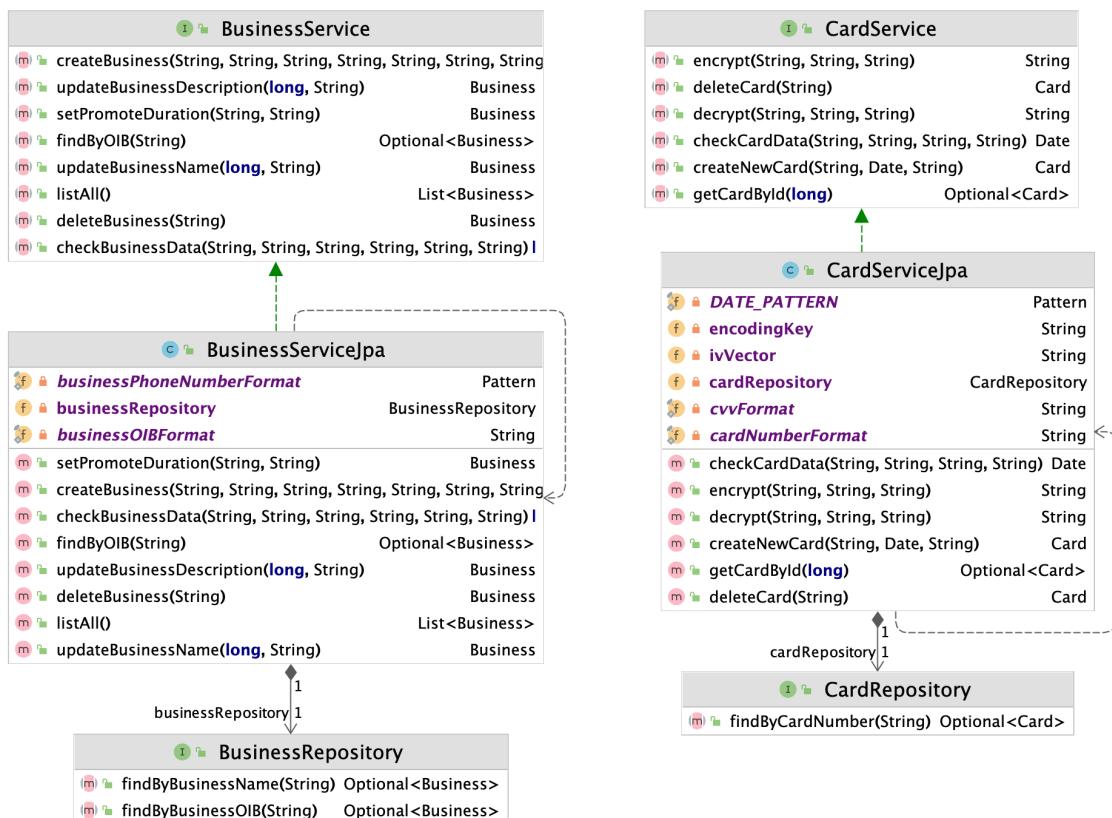
Slika 4.3: Dijagram klasa i metoda - Objekti i enumeracije

Prikazuje razrede koji reprezentiraju objekte koji se spremaju u bazu podataka te sudjeluju kao aktori u aplikaciji te enumeracije koje koristimo kao attribute prikazanih entiteta.

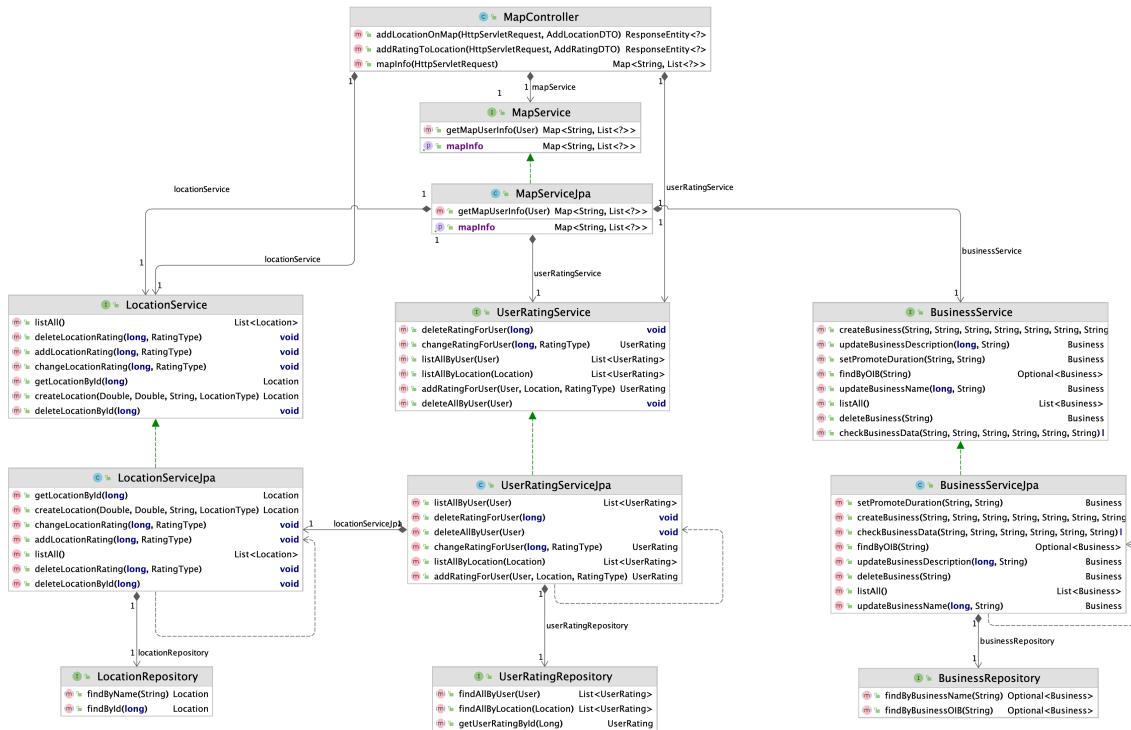


Slika 4.4: Dijagram klasa i metoda - Korisnici

Prikazuje strukturu razreda vezanih uz korisnika. `CustomUserDetailsService` je razred koji koristimo prilikom registracije i prijave korisnika. `ProfileController` preuzima zahtjeve na profilu korisnika i proslijeđuje zadatke `ProfileService`-u. Podacima se pristupa putem `UserRepository`-a. Dodatno je prikazan i `EmailSenderService` zadužen za slanje e-maila prilikom registracije korisnika.

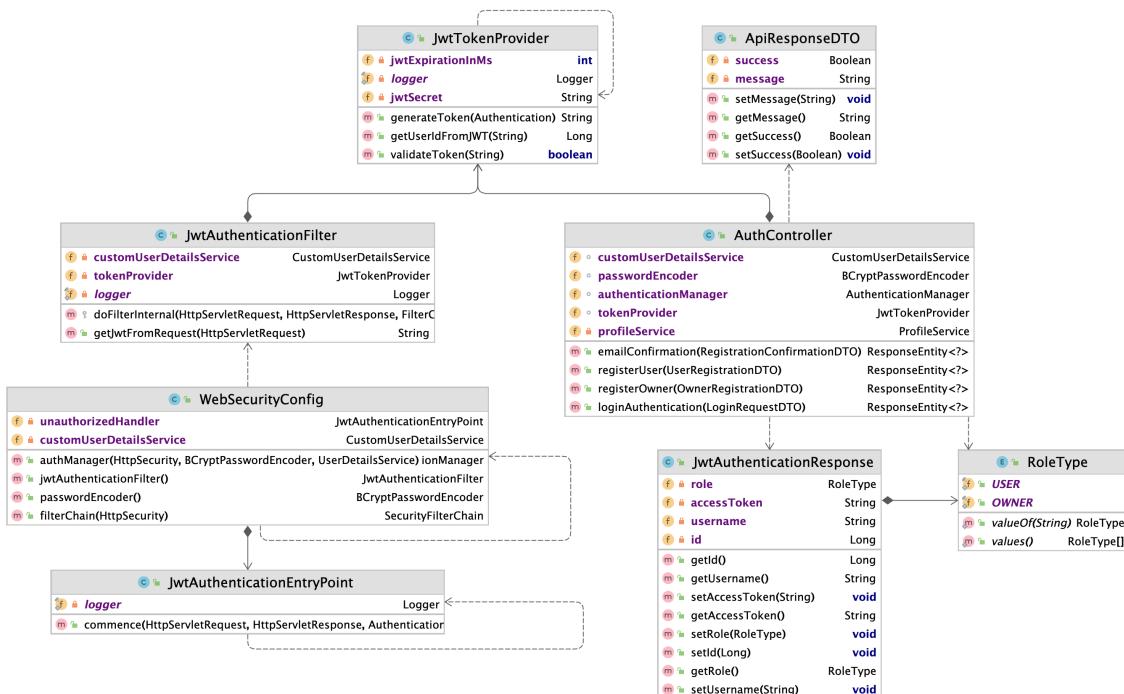


Slika 4.5: Dijagram klasa i metoda - Obrt i kartica
Prikazuje Service i Repository sloj za entitete Business i Card.



Slika 4.6: Dijagram klasa i metoda - Mapa

Prikazuje Controller - Service - Repository model mape, odnosno sve vezano uz obrte, lokacije i ocjene korisnika.

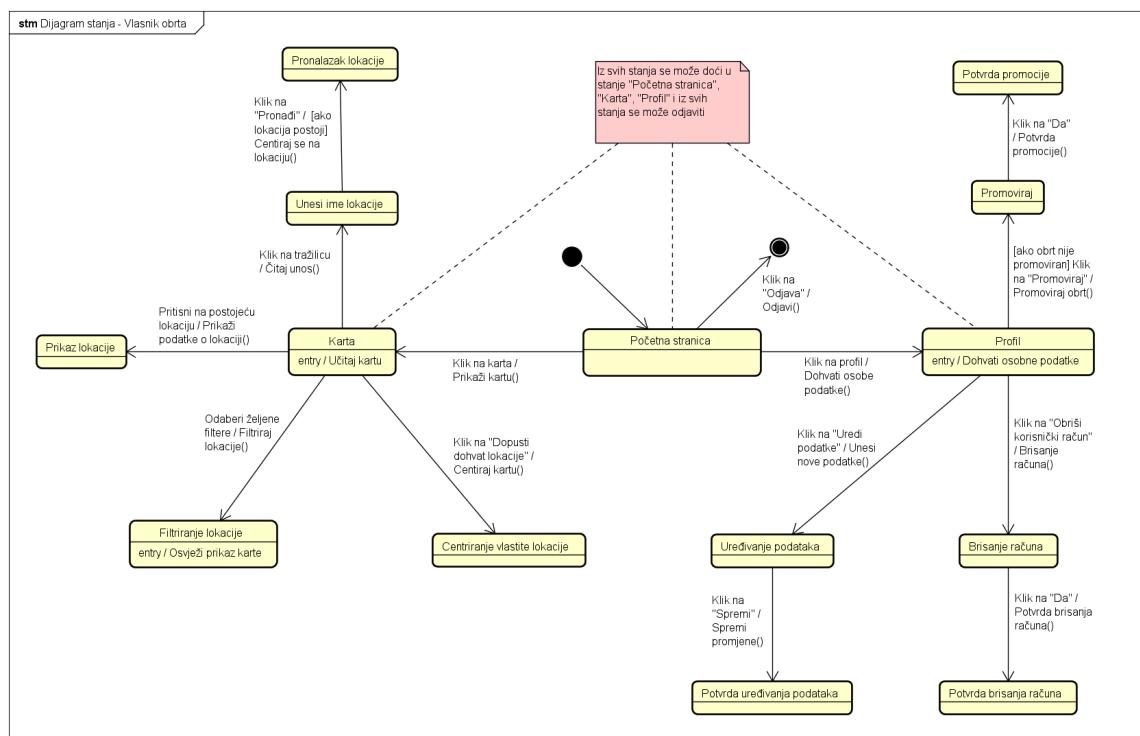


Slika 4.7: Dijagram klasa i metoda - Sigurnost

Prikazuje razrede kojima postižemo sigurnost aplikacije. Svi razredi koji započinju s Jwt zaduženi su za ispravno funkcioniranje JSON Web Token sustava. AuthController prima zahtijeve vezane za registraciju i prijavu. WebSecurityConfig je općenita konfiguracija sigurnosti, primjerice autorizacije pristupa određenim URL-ovima.

4.3 Dijagram stanja

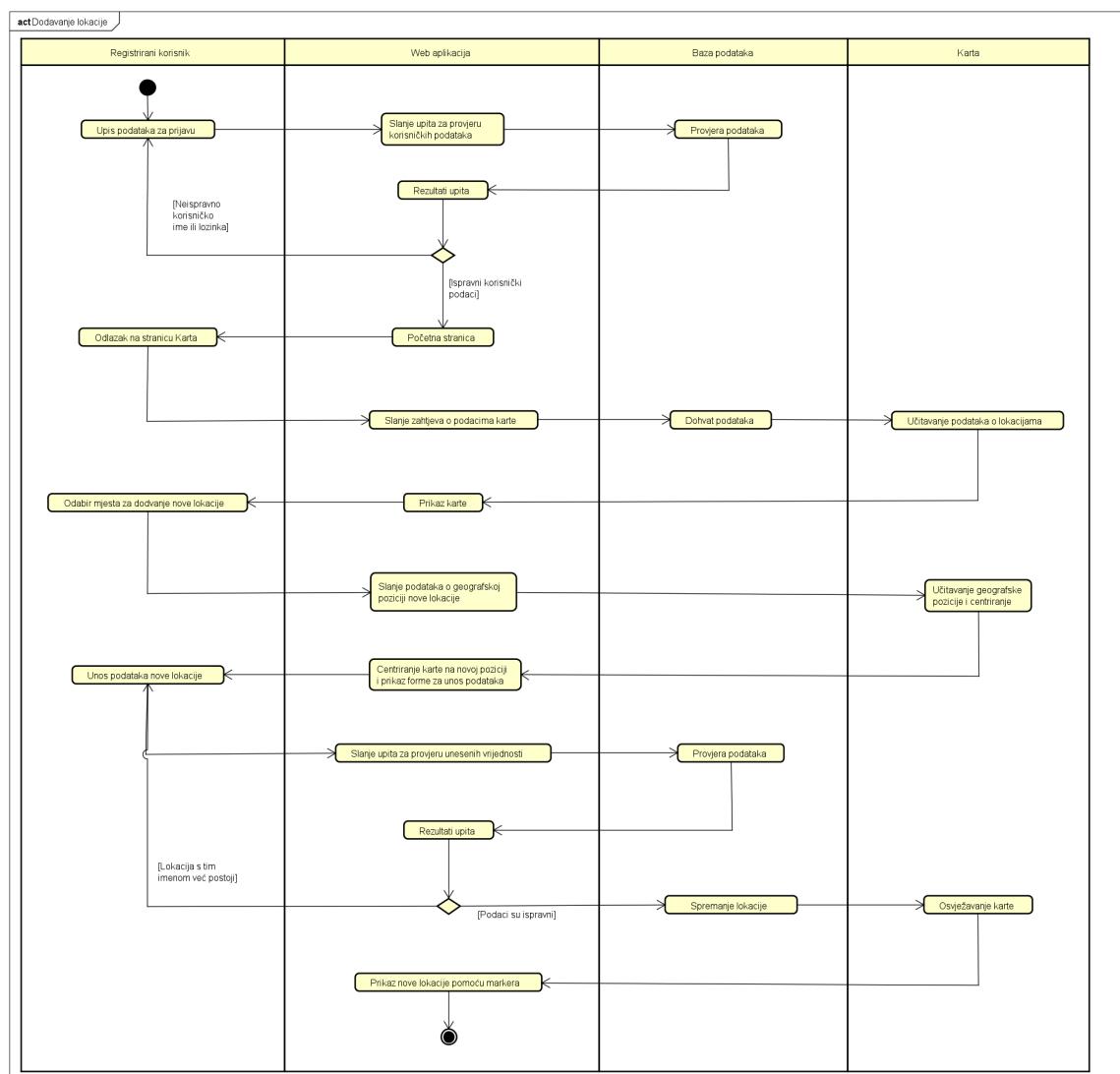
Dijagram stanja prikazuje stanja objekta te prijelaze iz jednog stanja u drugo temeljene na dogadajima. Na slici 4.8 prikazan je dijagram stanja za vlasnika obrta. Nakon prijave, vlasniku se prikazuje početna stranica, s koje može preći na stranicu karta. Na karti može dopustiti pristup vlastitoj lokaciji koja ga centrira, može filtrirati lokacije, te pronaći bilo koju važeću lokaciju upisom u tražilicu nakon koje se karta centrira na tu lokaciju. Klikom na "Profil" ima opciju promocije vlastitog obrta za koju mora potvrditi plaćanje, uređivanja podataka o profilu i obrtu, te brisanje profila.



Slika 4.8: Dijagram stanja za vlasnika obrta

4.4 Dijagram aktivnosti

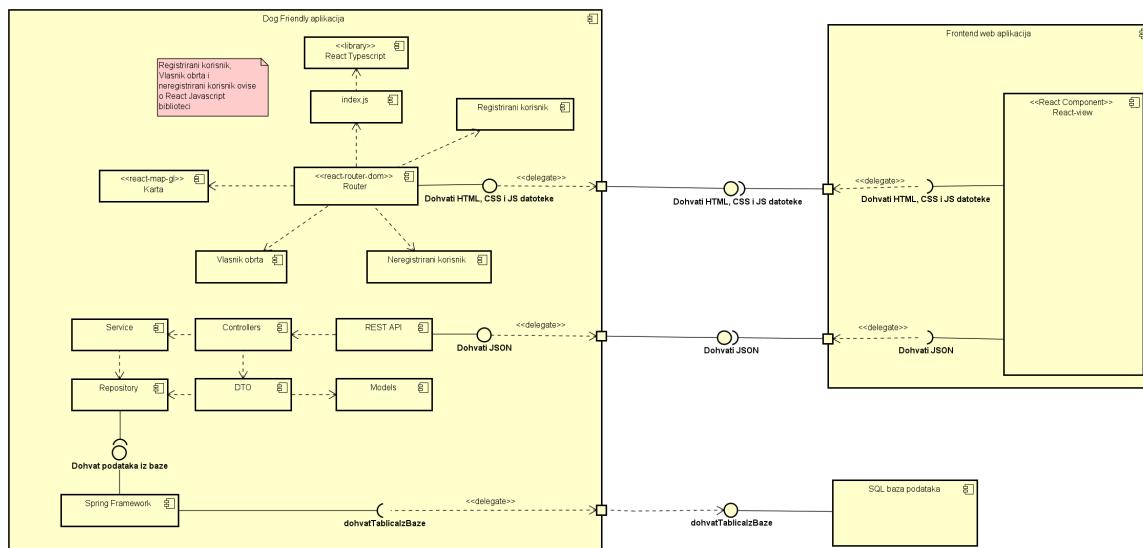
Dijagram aktivnosti služi za modeliranje ponašanja nizom akcija u kojima mogu biti definirani odgovarajući uvjeti prije i nakon izvođenja. Jedna aktivnost obuhvaća više čvorova i veza koji predstavljaju odgovarajući slijed zadataka. Na slici 4.9 prikazan je proces dodavanja lokacije. Korisnik da bi dodao novu lokaciju mora se prvo prijaviti u sustav. Nakon što se prijavio, korisnik mora otići na stranicu Karta gdje može prizvoljno kliknuti na kartu i dodati tu novu lokaciju. Zatim mu se otvara forma za upis gdje joj upisuje željene podatke. Ako su svi podaci ispravni, karta se centririra na novododanu lokaciju.



Slika 4.9: Dijagram aktivnosti za dodavanje lokacije

4.5 Dijagram komponenti

Dijagram komponenti prikazan na slici 4.10 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 pripradaju frontend dijelu aplikacije. Router je komponenta koja na upit 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 koje 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 REST API komponenti. REST API poslužuje podatke koji pripadaju backend dijelu aplikacije. Spring Framework je zadužen za dohvaćanje tablica iz baze podataka pomoću SQL upita. Podaci koji su pristigli iz baze se šalju dalje u arhitekturu Controller-Service-Repository u obliku DTO-a (Data transfer object). React-view komponenta preko dostupnih sučelja komunicira sa Dog Friendly aplikacijom te ovisno o korisnikovim akcijama osvježava prikaz i dohvaća nove podatke ili datoteke.



Slika 4.10: Dijagram komponenti za Dog Friendly aplikaciju

5. Implementacija i korisničko sučelje

5.1 Korištene tehnologije i alati

U nabranju korištenih tehnologija i alata krećemo s onima koji su realizirali komunikaciju u timu, Discord¹, te komunikaciju s asistentom i demosom, Microsoft Teams². Navedeni su omogućili dopisivanje, "glasovne razgovore", kao i mogućnost dijeljenja slika i zaslona u stvarnom vremenu. Istovremeni rad i brzo dijeljenje koda omogućio je Git³, specifično udaljeni repozitorij projekta na web platformi GitLab⁴.

Korišteno razvojno okruženje na backendu su IntelliJ⁵ i Eclipse⁶. IntelliJ IDEA je integrirano razvojno okruženje (IDE) napisano u Javi⁷ za razvoj računalnog softvera napisanog u Javi, Kotlinu, Groovyju i drugim jezicima koji se temelje na JVM-u. Razvio ga je JetBrains (ranije poznat kao IntelliJ). Eclipse je također IDE koje se koristi u računalnom programiranju. Sadrži osnovni radni prostor i proširivi plugin sustav za prilagođavanje okruženja. Primarna mu je upotreba za razvoj Java aplikacija, no može se koristit i u drugim programskim jezicima putem dodataka. Većina pripadnika grupe je koristila IntelliJ jer je Git podržan i integriran unutar IntelliJa te su ga smatrali bolje prilagođenom korisniku u odnosu na Eclipse.

Na frontendu je kao razvojno okruženje korišten Visual Studio Code⁸ također poznat kao VS Code. Napravio ga je Microsoft s Electron Frameworkom, za Windows, Linux i MacOS. Po definiciji je "uređivač izvornog koda" s mnogim značajkama od kojih je jedna ugrađeni Git. Koristili smo ga zbog jednostavnosti uporabe i prilagođenosti korisniku zbog čega je i rangiran kao jedan od najpopularnijih alata za razvojno okruženje.

¹<https://discord.com/>

²<https://www.microsoft.com/en-us/microsoft-teams/log-in>

³<https://git-scm.com/>

⁴<https://about.gitlab.com/>

⁵<https://www.jetbrains.com/idea/>

⁶<https://www.eclipse.org/ide/>

⁷<https://www.oracle.com/java/>

⁸<https://code.visualstudio.com/>

Aplikacija je pisana koristeći radni okvir Spring framework⁹ i jezik Java¹⁰ na backendu te React¹¹ i kombinaciju TypeScripta¹², HTML-a¹³ i CSS-a¹⁴ na frontendu.

Spring framework je jednostavni Java razvojni okvir otvorenog koda koji pruža programski i konfiguracijski model za razvoj Java aplikacija na visokoj razini. Cilj mu je omogućiti programerima efektivnije i brže programiranje aplikacija pojednostavljanjem Java. Spring uključuje mnoge module i ekstenzije od kojih smo u projektu koristili Spring Boot¹⁵, Spring Security¹⁶ i Spring Data¹⁷.

Java, temelj backenda, je objektno orijentirani programski jezik visoke razine koji se temelji na klasama i dizajniran je da ima što manje ovisnosti o implementacijama. Namijenjen je kako bi programeri svoj kod napisali samo jednom te ga pokrenu bilo gdje jer se kompajlirani Java kod može izvoditi na svim platformama koje podržavaju Javu bez potrebe za ponovnim kompajliranjem.

Na frontendu, najčešće korištena kombinacija za stvaranje web aplikacija/stranica je HTML (stvaranje i funkcionalnost stranice), CSS (stilska jezik koji opisuje prezentaciju dokumenta pisanog u HTML-u, u ovom slučaju) i JS¹⁸ (programska jezik koji je jedan od temeljnih tehnologija WWW-a uz HTML i CSS). U našem slučaju se koristi kombinacija HTMLa, CSSa i TypeScripta. TypeScript je besplatni programski jezik otvorenog koda koji je razvijen i održavan od strane Microsoft-a. Razlika TypeScripta i JavaScripta je u tome što je TypeScript nastao iz JavaScripta kada su programeri JS-a došli do zaključka da nije ispunio ideju iz koje je nastao (objektno orijentiran programski jezik) te je vremenom postao težak i kompleksan. Produkt toga je TypeScript, objektno orijentirani programski jezik koji se može koristiti bilo gdje, na svakom web-pregledniku, mobilu i OS-u. TSX podržava sučelja, ukazuje na greške u kodu prije kompilacije i podržava static tipove za razliku od JS-a. U globalu, JavaScript je TypeScript jer se može konvertirati samom promjenom ekstenzije, no TSX nije JS, već njegova jednostavnija objektno orijentirana verzija.

Kao i na backu, frontend se bazira na derivatu JavaScripta, specifično Reactu, JavaScript biblioteci za izradu korisničkih sučelja, koju je 2013. razvio Meta(Facebook)

⁹<https://spring.io/projects/spring-framework>

¹⁰<https://www.oracle.com/java/>

¹¹<https://reactjs.org/>

¹²<https://www.typescriptlang.org/>

¹³<https://html.com/>

¹⁴<https://en.wikipedia.org/wiki/CSS>

¹⁵<https://spring.io/projects/spring-boot>

¹⁶<https://spring.io/projects/spring-security>

¹⁷<https://spring.io/projects/spring-data>

¹⁸<https://www.javascript.com/>

te se danas koristi za mnoge web i mobilne aplikacije. Bazira se na komponentama koje veličinom mogu varirati od gumba, polja pa do obrasca i većih. Komponente mogu upravljati svojim stanjem i priopćiti to stanje podređenim komponentama.

Baza podataka se nalazi na poslužitelju u oblaku Render¹⁹. Za lokalni uvid u bazu podataka smo koristili pgAdmin²⁰, najpopularniju administrativnu i razvojnu platformu otvorenog koga s bogatim značajkama za PostgreSQL²¹, najnapredniju bazu podataka otvorenog koda na svijetu. PostgreSQL je besplatan sustav za upravljanje relacijskim bazama podataka otvorenog koda s naglaskom na proširivost i ukladjenost sa SQL-om.

Za izradu UML dijagrama smo koristili Astah²².

¹⁹<https://render.com/>

²⁰<https://www.pgadmin.org/>

²¹<https://www.postgresql.org/>

²²<https://portal.azure.com/>

5.2 Ispitivanje programskog rješenja

Testiranje komponenti provedeno je koristeći JUnit *SpringRunner.class* te anotaciju *@SpringBootTest(webEnvironment = SpringBootTest.WebEnvironment.MOCK, classes = DogFriendlyApplication.class)*. Za potrebe testiranja stvorena je nova H2 baza podataka te se podaci za nju nalaze u datotetci *application-unitTest.properties*. Ovim putem imitiramo bazu podataka aplikacije kako ne bi utjecali na njen sadržaj prilikom testiranja.

Za testiranje izabrali smo komponentu registracije vlasnika obrta. Registracijom se stvara objekt Owner (generalizacija objekta User) koji posljedično stvara objekte Business i Card. Za svaki test ispituje se ispravan unos metodom *assertDoesNotThrow()*, nekoliko primjera krivog unosa podataka metodom *assertThrows()*, a u slučajevima jedinstvenih vrijednosti ispituje se ponovni unos iste vrijednosti također metodom *assertThrows()*. Nakon svakog testa podaci se brišu iz baze podataka kako bi ona bila spremna za novo testiranje.

Testiranje sustava...

5.2.1 Ispitivanje komponenti

Za sve ispitne slučajeve koriste se navedeni podaci:

```
private static final String username = "user";
private static final String email = "user.dogfriendly@gmail.com";
private static final String password = "lozinka123";
private static final String businessName = "PET_SHOP";
private static final String businessType = "SHOP";
private static final String businessAddress = "Zagrebacka 5";
private static final String businessCity = "Zagreb";
private static final String businessOIB = "01234567890";
private static final String businessMobileNumber = "+385 99-234-56-78";
private static final String businessDescription = "Prodajemo sve za kucne
ljubimce";
private static final String cardNumber = "0000111122223333";
private static final String expiryDateMonth = "5";
private static final String expiryDateYear = "2025";
private static final String cvv3 = "123";
private static final String cvv4 = "1234";
```

1. ispitni slučaj namijenjen je provjeri korisničkog imena prilikom registracije vlasnika obrta. Prvo se kreira jedan objekt razreda "Owner" s točnim podacima. Zatim se testira unošenje krivog korisničkog imena. Prvi primjer je vrijednost null, drugi primjer je prazno polje, treći primjer je krivi format imena. Na samom kraju testira se registracija novog korisnika s korisničkim imenom koje je već zauzeto.

```
@Test
@Order(1)
public void registrationCheckUsername() {
    //CHECK ALL INFO
    assertDoesNotThrow(() -> userService.createOwner(username, email,
        password, businessName, businessType, businessAddress, businessCity,
        businessOIB, businessMobileNumber, businessDescription, cardNumber,
        expiryDateMonth, expiryDateYear, cvv3));

    assertThrows(Exception.class, () -> userService.createOwner(null, email,
        password, businessName, businessType, businessAddress, businessCity,
        businessOIB, businessMobileNumber, businessDescription, cardNumber,
        expiryDateMonth, expiryDateYear, cvv3));
    assertThrows(Exception.class, () ->
        userService.createOwner("?_kriviUsername", email, password,
        businessName, businessType, businessAddress, businessCity,
        businessOIB, businessMobileNumber, businessDescription, cardNumber,
        expiryDateMonth, expiryDateYear, cvv3));
    assertThrows(Exception.class, () ->

        //SAME USERNAME
        assertThrows(Exception.class, () -> userService.createOwner(username,
            email, password, businessName, businessType, businessAddress,
            businessCity, businessOIB, businessMobileNumber, businessDescription,
            cardNumber, expiryDateMonth, expiryDateYear, cvv3));

    User user = userService.getUserByUsername(username);
    profileService.deleteUserById(user.getId());
```

}

2. ispitni slučaj namijenjen je provjeri e-mail adrese prilikom registracije vlasnika obrta. Prvo se kreira jedan objekt razreda "Owner" s točnim podacima. Zatim se testira unošenje krive e-mail adrese. Prvi primjer je vrijednost null, drugi primjer je prazno polje, treći primjer je krivi format. Na samom kraju testira se registracija novog korisnika s e-mail adresom koja je već zauzeta.

```
@Test
@Order(2)
public void registrationCheckEmail() {
    //CHECK ALL INFO
    assertDoesNotThrow(() -> userService.createOwner(username, email,
        password, businessName, businessType, businessAddress,
        businessCity, businessOIB, businessMobileNumber,
        businessDescription, cardNumber, expiryDateMonth, expiryDateYear,
        cvv3));

    assertThrows(Exception.class, () -> userService.createOwner(username,
        null, password, businessName, businessType, businessAddress,
        businessCity, businessOIB, businessMobileNumber,
        businessDescription, cardNumber, expiryDateMonth, expiryDateYear,
        cvv3));
    assertThrows(Exception.class, () -> userService.createOwner(username,
        "", password, businessName, businessType, businessAddress,
        businessCity, businessOIB, businessMobileNumber,
        businessDescription, cardNumber, expiryDateMonth, expiryDateYear,
        cvv3));
    assertThrows(Exception.class, () -> userService.createOwner(username,
        "neispravanEmail", password, businessName, businessType,
        businessAddress, businessCity, businessOIB, businessMobileNumber,
        businessDescription, cardNumber, expiryDateMonth, expiryDateYear,
        cvv3));

    //SAME EMAIL
    assertThrows(Exception.class, () -> userService.createOwner(username,
        email, password, businessName, businessType, businessAddress,
        businessCity, businessOIB, businessMobileNumber,
```

```
businessDescription, cardNumber, expiryDateMonth, expiryDateYear,
cvv3));  
  
User user = userService.getUserByUsername(username);  
profileService.deleteUserById(user.getId());  
}
```

3. ispitni slučaj namijenjen je provjeri imena obrta prilikom registracije vlasnika obrta. Prvo se kreira jedan objekt razreda "Owner" s točnim podacima. Zatim se testira unošenje krivog imena. Prvi primjer je vrijednost null, drugi primjer je prazno polje, treći primjer je predugačko ime (više od 50 znakova). Na samom kraju testira se registracija novog korisnika, a time i novog obrta s imenom koje je već zauzeto.

```
@Test  
 @Order(3)  
public void registrationCheckBusinessName() {  
    //CHECK ALL INFO  
    assertDoesNotThrow(() -> userService.createOwner(username, email,  
        password, businessName, businessType, businessAddress,  
        businessCity, businessOIB, businessMobileNumber,  
        businessDescription, cardNumber, expiryDateMonth, expiryDateYear,  
        cvv3));  
  
    assertThrows(Exception.class, () -> userService.createOwner(username,  
        email, password, null, businessType, businessAddress, businessCity,  
        businessOIB, businessMobileNumber, businessDescription, cardNumber,  
        expiryDateMonth, expiryDateYear, cvv3));  
    assertThrows(Exception.class, () -> userService.createOwner(username,  
        email, password, "", businessType, businessAddress, businessCity,  
        businessOIB, businessMobileNumber, businessDescription, cardNumber,  
        expiryDateMonth, expiryDateYear, cvv3));  
    assertThrows(Exception.class, () -> userService.createOwner(username,  
        email, password, "Ovo je preveliko odnosno predugako ime za  
        postaviti nekom obrtu na njoj stranici", businessType,  
        businessAddress, businessCity, businessOIB, businessMobileNumber,  
        businessDescription, cardNumber, expiryDateMonth, expiryDateYear,  
        cvv3));
```

```
//SAME NAME  
assertThrows(Exception.class, () -> userService.createOwner(username,  
    email, password, null, businessType, businessAddress, businessCity,  
    businessOIB, businessMobileNumber, businessDescription, cardNumber,  
    expiryDateMonth, expiryDateYear, cvv3));  
  
User user = userService.getUserByUsername(username);  
profileService.deleteUserById(user.getId());  
}
```

4. ispitni slučaj namijenjen je provjeri OIB-a obrta prilikom registracije vlasnika obrta. Prvo se kreira jedan objekt razreda "Owner" s točnim podacima. Zatim se testira unošenje krivog OIB-a. Prvi primjer je vrijednost null, drugi primjer je prazno polje, treći primjer je prekratak OIB (manje od 11 znakova), četvrти primjer je predugačak OIB (više od 11 znakova) i peti primjer je unos znakova koji nisu znamenke. Na samom kraju testira se registracija novog korisnika, a time i novog obrta s OIB-om koji je već zauzet.

```
@Test  
 @Order(4)  
public void registrationCheckBusinessOIB() {  
    //CHECK ALL INFO  
    assertDoesNotThrow(() -> userService.createOwner(username, email,  
        password, businessName, businessType, businessAddress,  
        businessCity, businessOIB, businessMobileNumber,  
        businessDescription, cardNumber, expiryDateMonth, expiryDateYear,  
        cvv3));  
  
    assertThrows(Exception.class, () -> userService.createOwner(username,  
        email, password, businessName, businessType, businessAddress,  
        businessCity, null, businessMobileNumber, businessDescription,  
        cardNumber, expiryDateMonth, expiryDateYear, cvv3));  
    assertThrows(Exception.class, () -> userService.createOwner(username,  
        email, password, businessName, businessType, businessAddress,  
        businessCity, "", businessMobileNumber, businessDescription,  
        cardNumber, expiryDateMonth, expiryDateYear, cvv3));  
    assertThrows(Exception.class, () -> userService.createOwner(username,
```

```
email, password, businessName, businessType, businessAddress,
businessCity, "0123456789", businessMobileNumber,
businessDescription, cardNumber, expiryDateMonth, expiryDateYear,
cvv3));
assertThrows(Exception.class, () -> userService.createOwner(username,
email, password, businessName, businessType, businessAddress,
businessCity, "012345678901", businessMobileNumber,
businessDescription, cardNumber, expiryDateMonth, expiryDateYear,
cvv3));
assertThrows(Exception.class, () -> userService.createOwner(username,
email, password, businessName, businessType, businessAddress,
businessCity, "0123456789a", businessMobileNumber,
businessDescription, cardNumber, expiryDateMonth, expiryDateYear,
cvv3));

//SAME OIB
assertThrows(Exception.class, () -> userService.createOwner(username,
email, password, businessName, businessType, businessAddress,
businessCity, businessOIB, businessMobileNumber,
businessDescription, cardNumber, expiryDateMonth, expiryDateYear,
cvv3));

User user = userService.getUserByUsername(username);
profileService.deleteUserById(user.getId());
}
```

5. ispitni slučaj namijenjen je provjeri broja kartice prilikom registracije vlasnika obrta. Prvo se kreira jedan objekt razreda "Owner" s točnim podacima. Zatim se testira unošenje krivog broja kartice. Prvi primjer je vrijednost null, drugi primjer je prazno polje, treći primjer je prekratak broj (manje od 16 znakova), četvrti primjer je predugačak broj (više od 16 znakova) i peti primjer je unos znakova koji nisu isključivo znamenke iako je točna duljina unosa.

```
@Test
@Order(5)
public void registrationCheckCardNumber() {
    //CHECK ALL INFO
    assertDoesNotThrow(() -> userService.createOwner(username, email,
```

```
password, businessName, businessType, businessAddress,
businessCity, businessOIB, businessMobileNumber,
businessDescription, cardNumber, expiryDateMonth, expiryDateYear,
cvv3));  
  
assertThrows(Exception.class, () -> userService.createOwner(username,
email, password, businessName, businessType, businessAddress,
businessCity, businessOIB, businessMobileNumber,
businessDescription, null, expiryDateMonth, expiryDateYear, cvv3));
assertThrows(Exception.class, () -> userService.createOwner(username,
email, password, businessName, businessType, businessAddress,
businessCity, businessOIB, businessMobileNumber,
businessDescription, "", expiryDateMonth, expiryDateYear, cvv3));
assertThrows(Exception.class, () -> userService.createOwner(username,
email, password, businessName, businessType, businessAddress,
businessCity, businessOIB, businessMobileNumber,
businessDescription, "0000", expiryDateMonth, expiryDateYear,
cvv3));
assertThrows(Exception.class, () -> userService.createOwner(username,
email, password, businessName, businessType, businessAddress,
businessCity, businessOIB, businessMobileNumber,
businessDescription, "00001112223334", expiryDateMonth,
expiryDateYear, cvv3));
assertThrows(Exception.class, () -> userService.createOwner(username,
email, password, businessName, businessType, businessAddress,
businessCity, businessOIB, businessMobileNumber,
businessDescription, "aaaabbbbccccdd", expiryDateMonth,
expiryDateYear, cvv3));  
  
User user = userService.getUserByUsername(username);
profileService.deleteUserById(user.getId());  
}
```

6. ispitni slučaj namijenjen je provjeri kontrolnog broja kartice (CVV) prilikom registracije vlasnika obrta. Prvo se kreira jedan objekt razreda "Owner" s točnim podacima i koristi se duljina kontrolnog broja od 3 znaka. Zatim se testira unošenje krivog CVV-a. Prvi primjer je vrijednost null, drugi primjer je prazno polje, treći primjer je prekratak broj (manje od 3 znaka), četvrti primjer je predugačak broj

(više od 4 znaka) i peti primjer je unos znakova koji nisu isključivo znamenke iako je točna duljina unosa. Na samom kraju testira se registracija s kontrolnim brojem duljine 4 što je isto dozvoljena vrijednost.

```
@Test
@Order(6)
public void registrationCheckCVV() {
    //CHECK ALL INFO
    assertDoesNotThrow(() -> userService.createOwner(username, email,
        password, businessName, businessType, businessAddress,
        businessCity, businessOIB, businessMobileNumber,
        businessDescription, cardNumber, expiryDateMonth, expiryDateYear,
        cvv3));

    assertThrows(Exception.class, () -> userService.createOwner(username,
        email, password, businessName, businessType, businessAddress,
        businessCity, businessOIB, businessMobileNumber,
        businessDescription, cardNumber, expiryDateMonth, expiryDateYear,
        null));

    assertThrows(Exception.class, () -> userService.createOwner(username,
        email, password, businessName, businessType, businessAddress,
        businessCity, businessOIB, businessMobileNumber,
        businessDescription, cardNumber, expiryDateMonth, expiryDateYear,
        ""));
    assertThrows(Exception.class, () -> userService.createOwner(username,
        email, password, businessName, businessType, businessAddress,
        businessCity, businessOIB, businessMobileNumber,
        businessDescription, cardNumber, expiryDateMonth, expiryDateYear,
        "12"));

    assertThrows(Exception.class, () -> userService.createOwner(username,
        email, password, businessName, businessType, businessAddress,
        businessCity, businessOIB, businessMobileNumber,
        businessDescription, cardNumber, expiryDateMonth, expiryDateYear,
        "12345"));

    assertThrows(Exception.class, () -> userService.createOwner(username,
        email, password, businessName, businessType, businessAddress,
        businessCity, businessOIB, businessMobileNumber,
        businessDescription, cardNumber, expiryDateMonth, expiryDateYear,
        "123456"));
}
```

```
"abc"));

User user = userService.getUserByUsername(username);
profileService.deleteUserById(user.getId());

assertDoesNotThrow(() -> userService.createOwner(username, email,
    password, businessName, businessType, businessAddress,
    businessCity, businessOIB, businessMobileNumber,
    businessDescription, cardNumber, expiryDateMonth, expiryDateYear,
    cvv));

user = userService.getUserByUsername(username);
profileService.deleteUserById(user.getId());
}
```

Rezultati testiranja prikazani su na slici.

✓ ✓ RegistrationUnitTests (hr.fer.progi.simplicity)	18 sec 325 ms
✓ registrationCheckEmail	4 sec 115 ms
✓ registrationCheckUsername	2 sec 360 ms
✓ registrationCheckCVV	4 sec 854 ms
✓ registrationCheckBusinessName	2 sec 418 ms
✓ registrationCheckBusinessOIB	2 sec 359 ms
✓ registrationCheckCardNumber	2 sec 219 ms

Slika 5.1: Rezultati JUnit testova

5.2.2 Ispitivanje sustava

Ispitivanje sustava se provelo pomoću radnog okvira Selenium. Specifičnije, prva dva od sedam testova se obradilo koristeći Selenium WebDriver, te preostalih pet od sedam testova se obradilo pomoću dodatka za preglednik Selenium IDE. U svim testovima, pregledniku je odbijen pristup lokaciji zbog lakšeg testiranja.

Prvi test registrira prvog korisnika na stranicu, uz to da se testiraju moguće greške koje bi običan korisnik mogao pokušati napraviti. Prvo, test pokušava prijaviti korisnika kao da već postoji. Sustav mu ne daje da se prijavi jer nje-gov korisnički račun još ne postoji. Nakon što se potvrди da je došlo do upozo-

renja, test ide registrirati korisnika. Test prvo pokuša registrirati korisnika bez ikakvih unošenih podataka. Nakon što se potvrdi pojavljivanje upozorenja, sustav pokušava unositi podatke koje sustav ne prihvata sve dok mu sustav ne prestane bacati upozorenja za neprihvatljive podatke. Kada su unešeni podaci ispravni, sustav omogućuje testu da registrira prvog korisnika.

```
@Test
public void seleniumWebUserRegistration() {
    WebDriver driver = new ChromeDriver();
    System.setProperty("webdriver.chrome.driver", "C:\\\\Program Files
        (x86)\\\\chromedriver.exe");
    driver.manage().timeouts().implicitlyWait(20, TimeUnit.SECONDS);
    driver.get("http://localhost:3000/");

    driver.findElement(By.xpath("//a[@href='/auth/login']")).click();

    WebElement element = driver.findElement(By.id("username"));
    element.sendKeys("DF_TestUser");
    element = driver.findElement(By.id("password"));
    element.sendKeys("123");

    driver.findElement(By.xpath("//button[@type='submit']")).click();

    if(driver.findElement(By.className("error-container")).isDisplayed())
        System.out.println("Element is Visible");

    driver.findElement(By.xpath("//a[@href='/auth/register']")).click();
    driver.findElement(By.xpath("//a[@href='/auth/register/user']")).click();

    element = driver.findElement(By.id("username"));
    element.sendKeys("DF_TestUser");
    element = driver.findElement(By.id("email"));
    element.sendKeys("user");
    element = driver.findElement(By.id("password"));
    element.sendKeys("123");

    driver.findElement(By.xpath("//button[@type='submit']")).click();
```

```
if(driver.findElement(By.id("email-helper-text")).isDisplayed())
    System.out.println("Element is Visible");
if(driver.findElement(By.id("password-helper-text")).isDisplayed())
    System.out.println("Element is Visible");

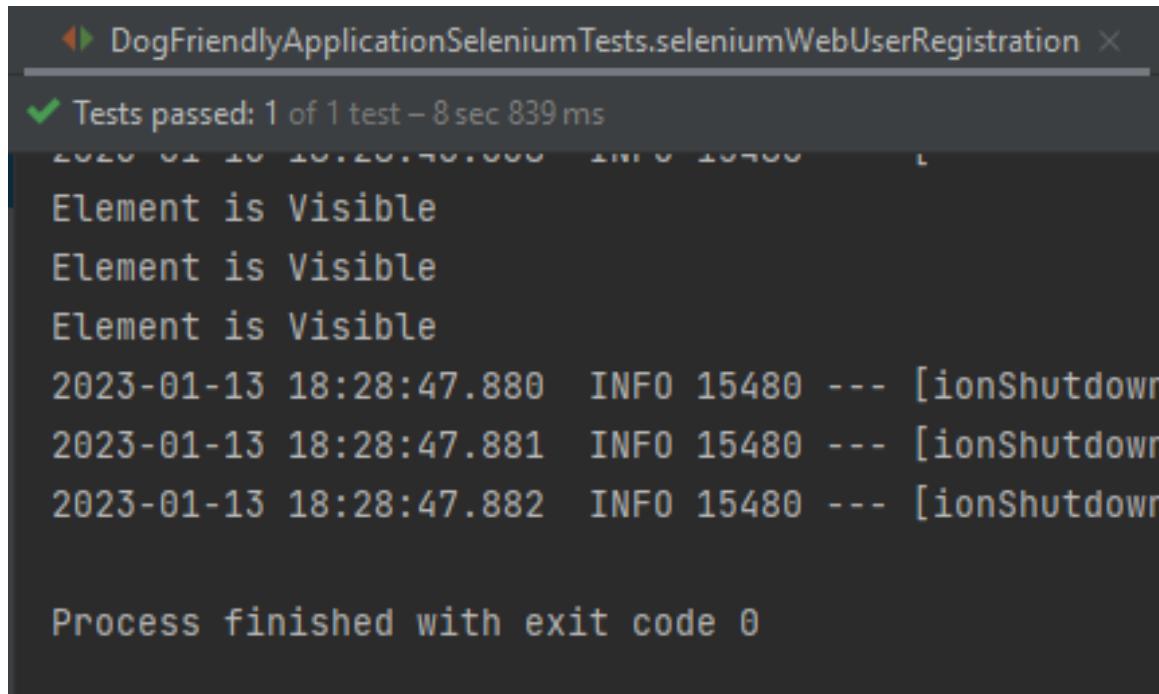
element = driver.findElement(By.id("email"));
element.sendKeys(Keys.CONTROL + "a");
element.sendKeys(Keys.DELETE);
element.sendKeys("dogfriendly.test.owner1@gmail.com");
element = driver.findElement(By.id("password"));
element.sendKeys(Keys.CONTROL + "a");
element.sendKeys(Keys.DELETE);
element.sendKeys("12345678");

driver.findElement(By.xpath("//button[@type='submit']")).click();

boolean compRes = new WebDriverWait(driver, Duration.ofSeconds(40))
    .until(ExpectedConditions.urlToBe("http://localhost:3000/auth/login"));

String redirURL = driver.getCurrentUrl();
compRes = redirURL.contains("auth/login");
if
    (!driver.findElement(By.className("registration-message")).isDisplayed())
        compRes = false;
assertEquals(compRes, true);

driver.quit();
}
```



```
▶ DogFriendlyApplicationSeleniumTests.seleniumWebUserRegistration ×
✓ Tests passed: 1 of 1 test – 8 sec 839 ms
Element is Visible
Element is Visible
Element is Visible
2023-01-13 18:28:47.880  INFO 15480 --- [ionShutdown
2023-01-13 18:28:47.881  INFO 15480 --- [ionShutdown
2023-01-13 18:28:47.882  INFO 15480 --- [ionShutdown

Process finished with exit code 0
```

Slika 5.2: Rezultat prvog Selenium testa

Drugi Test je gotovo isti kao i prvi test uz razliku da test pokušava registrirati vlasnika obrta, a ne korisnika. Uz registraciju vlasnika obrta se registrira i novi obrt kojem je vlasnik upravo registrirani vlasnik obrta.

```
@Test
public void seleniumOwnerRegistration() {
    WebDriver driver = new ChromeDriver();
    driver.manage().window().setSize(new Dimension(700, 1300));
    System.setProperty("webdriver.chrome.driver", "C:\\\\Program Files
        (x86)\\\\chromedriver.exe");
    driver.manage().timeouts().implicitlyWait(20, TimeUnit.SECONDS);
    driver.get("http://localhost:3000/");

    driver.findElement(By.className("hamburger")).click();
    driver.findElement(By.className("hamburger")).click();
    driver.findElement(By.className("hamburger")).click();
    driver.findElement(By.xpath("//div[@class='menu-dropdown']/div[3]")).click();

    driver.findElement(By.xpath("//a[@href='/auth/register']")).click();
    driver.findElement(By.xpath("//a[@href='/auth/register/owner']")).click();
```

```
// REGISTRIRAJ SE
driver.findElement(By.xpath("//button[@type='submit']")).click();

if(driver.findElement(By.id("username-helper-text")).isDisplayed())
    System.out.println("Element is Visible");
if(driver.findElement(By.id("email-helper-text")).isDisplayed())
    System.out.println("Element is Visible");
if(driver.findElement(By.id("password-helper-text")).isDisplayed())
    System.out.println("Element is Visible");
if(driver.findElement(By.id("businessName-helper-text")).isDisplayed())
    System.out.println("Element is Visible");
if(driver.findElement(By.id("businessAdress-helper-text")).isDisplayed())
    System.out.println("Element is Visible");
if(driver.findElement(By.id("businessCity-helper-text")).isDisplayed())
    System.out.println("Element is Visible");
if(driver.findElement(By.id("businessOIB-helper-text")).isDisplayed())
    System.out.println("Element is Visible");
if(driver.findElement(By.id("businessMobileNumber-helper-text")).isDisplayed())
    System.out.println("Element is Visible");
if(driver.findElement(By.id("cardNumber-helper-text")).isDisplayed())
    System.out.println("Element is Visible");

// Upisujemo podatke vlasnika obra
WebElement element = driver.findElement(By.id("username"));
element.sendKeys("DF_TestOwner");
element = driver.findElement(By.id("email"));
element.sendKeys("dogfriendly.test.owner@gmail.com");
element = driver.findElement(By.id("password"));
element.sendKeys("12345678");
element = driver.findElement(By.id("businessName"));
element.sendKeys("TestingBusiness");

driver.findElement(By.xpath("//form[@class='register-form']/div/div[5]")).click();
driver.findElement(By.xpath("//li[@data-value='VET']")).click();

element = driver.findElement(By.id("businessAdress"));
element.sendKeys("Unska ul. 3");
```

```
element = driver.findElement(By.id("businessCity"));
element.sendKeys("Zagreb");
element = driver.findElement(By.id("businessOIB"));
element.sendKeys("OIB01234567890IB"); // WARNING
element = driver.findElement(By.id("businessMobileNumber"));
element.sendKeys("Broj telefona: +012/3456-789"); // WARNING
element = driver.findElement(By.id("businessDescription"));
element.sendKeys("Generic description.");
element = driver.findElement(By.id("cardNumber"));
element.sendKeys("Card Number 123456789"); // WARNING
driver.findElement(By.id("expiryDateMonth")).click();
driver.findElement(By.xpath("//div[@role='presentation']/div[3]/ul/li[2]")).click();
driver.findElement(By.id("getExpiryDateProps")).click();
driver.findElement(By.xpath("//div[@role='presentation']/div[3]/ul/li[4]")).click();
element = driver.findElement(By.id("cvv"));
element.sendKeys("1"); // WARNING

// REGISTRIRAJ SE
driver.findElement(By.xpath("//button[@type='submit']")).click();

element = driver.findElement(By.id("businessOIB"));
element.sendKeys(Keys.CONTROL + "a");
element.sendKeys(Keys.DELETE);
element.sendKeys("01234567890");
element = driver.findElement(By.id("businessMobileNumber"));
element.sendKeys(Keys.CONTROL + "a");
element.sendKeys(Keys.DELETE);
element.sendKeys("+012/3456-789");
element = driver.findElement(By.id("cardNumber"));
element.sendKeys(Keys.CONTROL + "a");
element.sendKeys(Keys.DELETE);
element.sendKeys("1234567890123456");
element = driver.findElement(By.id("cvv"));
element.sendKeys(Keys.CONTROL + "a");
element.sendKeys(Keys.DELETE);
element.sendKeys("123");

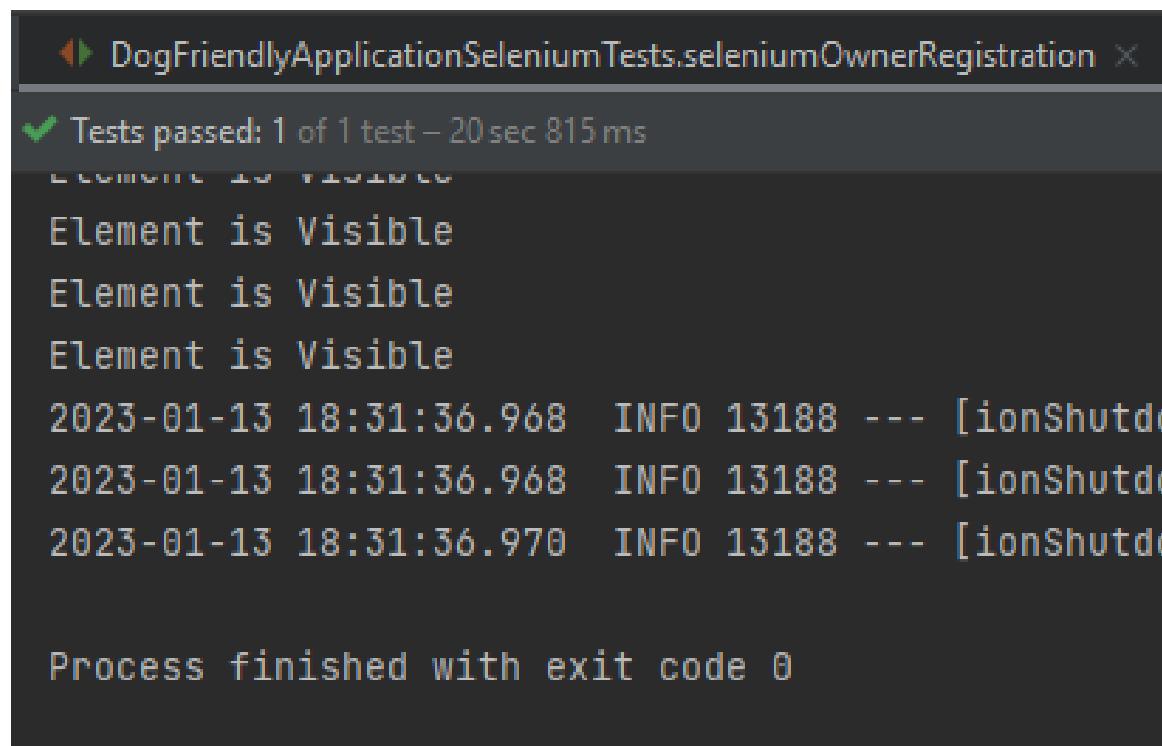
// REGISTRIRAJ SE
```

```
driver.findElement(By.xpath("//button[@type='submit']")).click();

boolean compRes = new WebDriverWait(driver,
    Duration.ofSeconds(40)).until(ExpectedConditions.urlToBe("http://localhost:3000/auth"));

String redirURL = driver.getCurrentUrl();
compRes = redirURL.contains("auth/login");
if
    (!driver.findElement(By.className("registration-message")).isDisplayed())
    compRes = false;
assertEquals(compRes, true);

driver.quit();
}
```



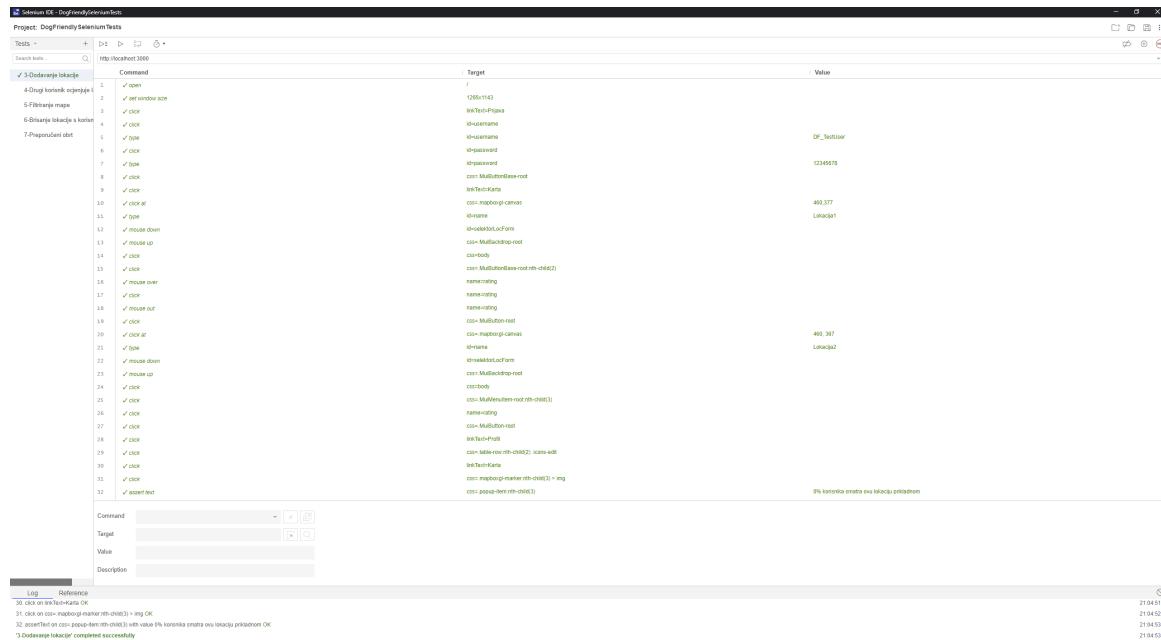
```
DogFriendlyApplicationSeleniumTests.seleniumOwnerRegistration ×
✓ Tests passed: 1 of 1 test – 20 sec 815 ms
Element is Visible
Element is Visible
Element is Visible
2023-01-13 18:31:36.968  INFO 13188 --- [ionShutdownHook]
2023-01-13 18:31:36.968  INFO 13188 --- [ionShutdownHook]
2023-01-13 18:31:36.970  INFO 13188 --- [ionShutdownHook]

Process finished with exit code 0
```

Slika 5.3: Rezultat drugog Selenium testa

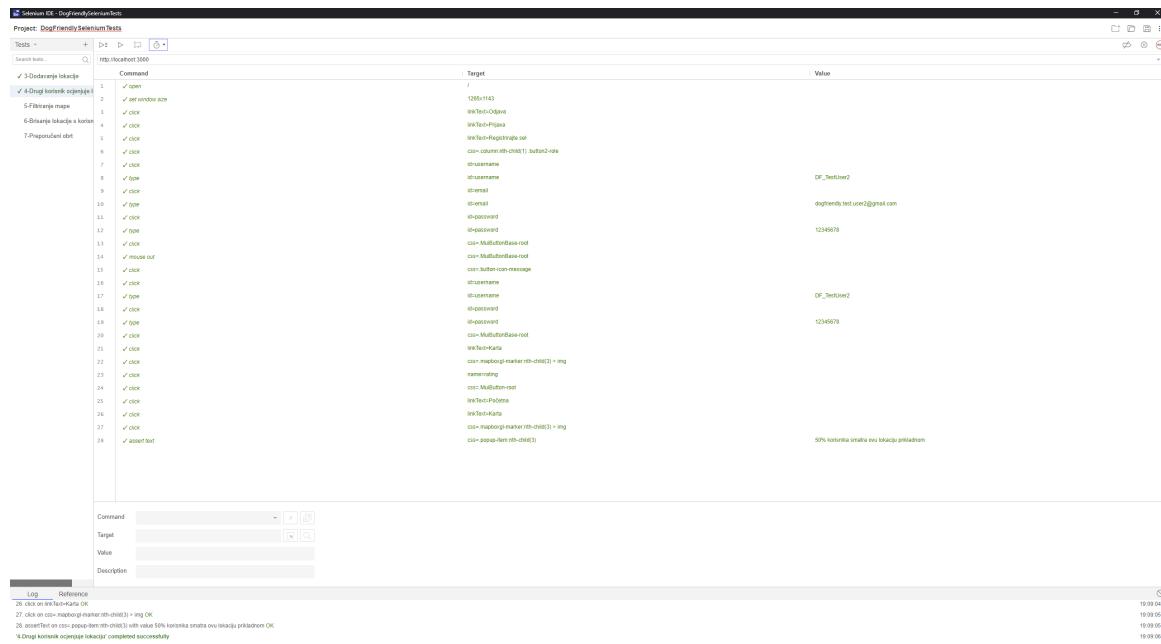
Treći test dodaje dvije lokacije na mapi kao prvi registrirani korisnik te mijenja ocjenu lokacije. Ovaj test je napravljen da se može vidjeti mogućnost mijenjanja ocjena na lokacijama. Prvo, test se prijavi kao prvi registrirani korisnik, nakon

čega dodaje dvije lokacije s pozitivnim ocjenama. Lokacija1 pod kategorijom park, te Lokacija2 pod kategorijom restoran. Nakon dodavanja lokacija, test ide promjeniti ocjenu na prvoj lokaciji. Kad se ocjena promijeni, test ide na stranicu karte i provjerava je li postotak pozitivnog mišljenja preuređen na nula posto.



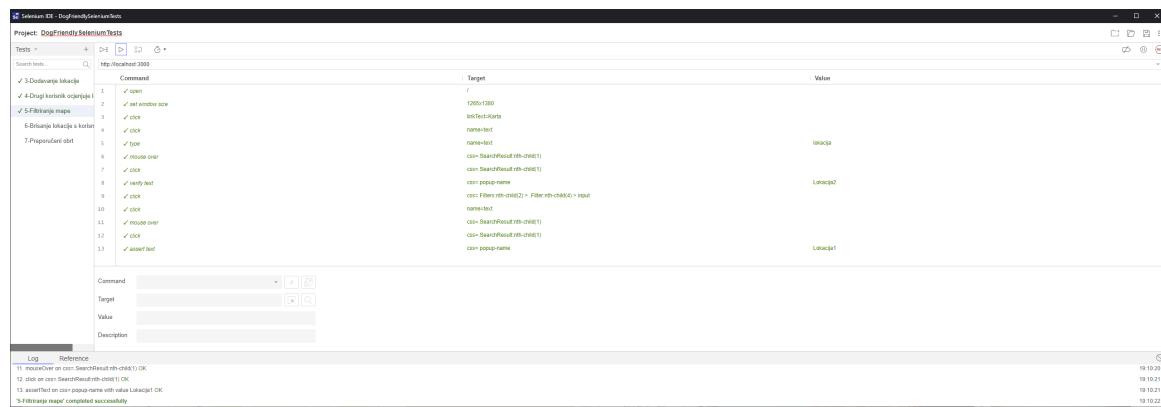
Slika 5.4: Rezultat trećeg Selenium testa

Četvrti test pokazuje da drugi korisnici mogu ocjenjivati lokaciju koji su stvorili drugi korisnici. U ovom testu, test registrira novog korisnika te kao novi korisnik, ocjenjuje prvo dodanu lokaciju, Lokacija1. Nakon ponovnog učitavanja stranice, vidi se da se postotak pozitivnog mišljenja promijenio sa nula posto na pedeset posto.



Slika 5.5: Rezultat četvrtog Selenium testa

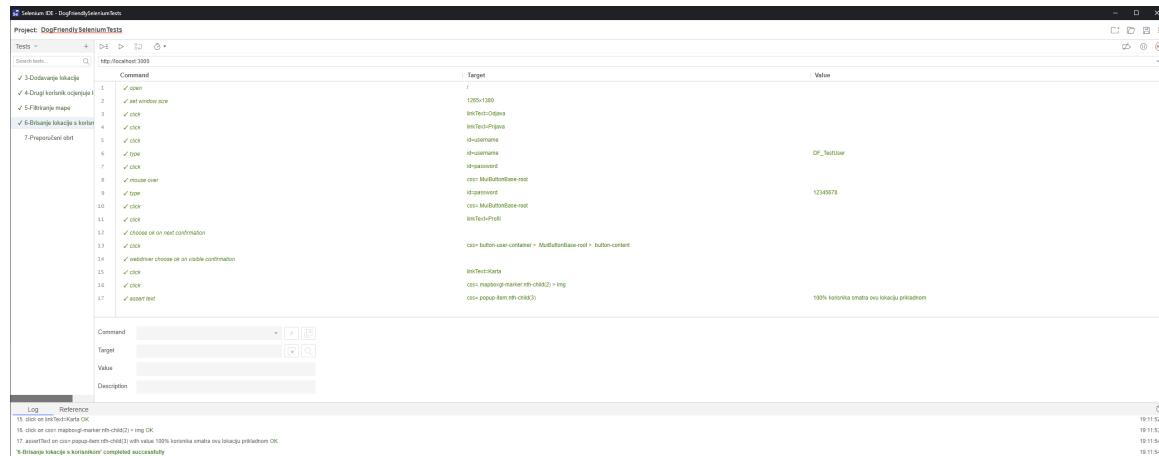
Pomoću petog testa, prikazuje se funkcionalnost tražilice i filtera na stranici mape. Kada otvorimo stranicu sa mapom i utipkamo u tražilici "lokacija", dobijemo Lokacija1 i Lokacija2 kao rezultat. Ako na filteru maknemo restorane, vidi se da sad na tražilici Lokacija2 više nije rezultat tražilice jer je ona svrstana pod kategorijom restorana.



Slika 5.6: Rezultat petog Selenium testa

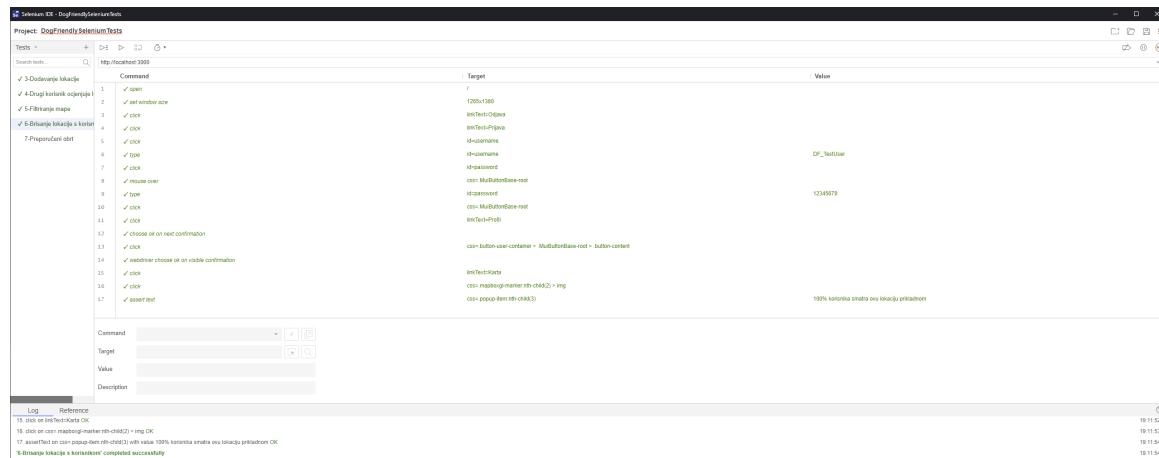
Šesti test prikazuje da će lokacija na mapi ostati zapisana unatoč tome što se korisnički račun autora lokacije izbrisao. Ostat će na mapi sve dok postoji neki korisnik koji je dao ocjenu za tu istu lokaciju. U ovom slučaju, test briše korisnički

račun prvog korisnika. Poslije test ide na mapu da vidi da se ocjena lokacije promijenila, ali nije i nestala, nakon što se izbrisao račun.



Slika 5.7: Rezultat šestog Selenium testa

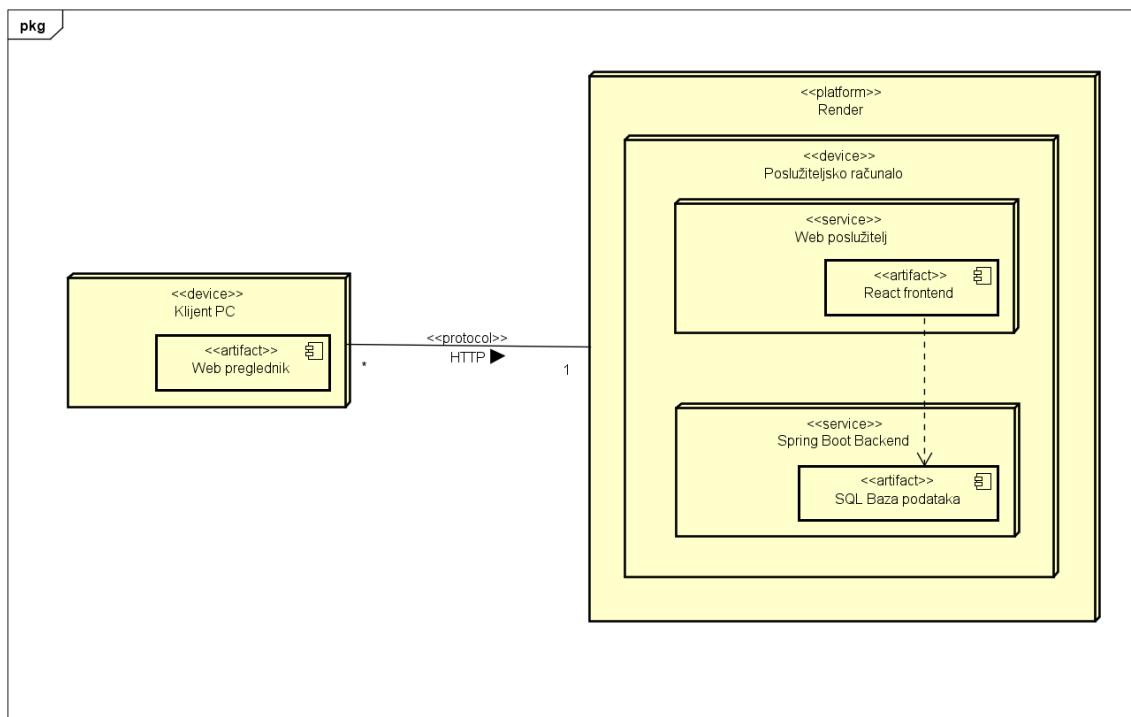
Kao račun vlasnika obrta iz drugog testa, sedmi test promovira obrt tog istog računa. Kada se obrt promovira, obrt mogu vidjeti drugi korisnici, a i sam vlasnik obrta, među preporučenim obrtimima na stranici mape.



Slika 5.8: Rezultat sedmog Selenium testa

5.3 Dijagram razmještaja

UML dijagram razmještaja je staticki UML dijagram koji opisuje topologiju sustava i usredotočen je na odnos sklopovskih i programskih dijelova. Naš dijagram je specifikacijski dijagram te prikazuje pregled implementacije artefakata bez upućivanja na specifične slučajeve artefakata ili čvorova. Klijenti koriste web preglednik kako bi pristupili web poslužitelju. Arhitektura sustava je "klijent - poslužitelj", a komunikacija između računala klijenta i poslužitelja se odvija preko HTTP veze. Klijent se putem web preglednika spaja na korisničko sučelje (front-end) preko kojeg komunicira s poslužiteljskom aplikacijom (back-end). Korisničko sučelje je implementirano u radnom okviru React i pokrenuto je na platformi Render gdje se ujedno nalazi i SQL baza podataka.



Slika 5.9: Dijagram razmještaja

5.4 Upute za puštanje u pogon

Naš projekt je za puštanje u pogon koristio Render, objedinjeni oblak za izradu i pokretanje svih aplikacija i web stranica s besplatnim TLS certifikatima, globalnim CDN-om, DDoS zaštitom, privatnim mrežama i automatskim puštanjem u pogon koda s Gita.

Konfiguracija backenda za puštanje u pogon

Konfiguracija backenda kreće od Apache Tomcata²³. Tomcat je besplatna, otvorena implementacija tehnologija Jakarta Servlet, Jakarta Expression Language i WebSocket koja omogućuje "čistu Java" HTTP web poslužiteljsku okolinu u kojoj se također može izvoditi Java kod. Dakle, Tomcat je Java web aplikacijski poslužitelj iako nije potpuni JEE aplikacijski poslužitelj. U našem projektu se koristi Spring Boot koji pruža ugrađenu verziju Apache Tomcata (7). Spring Boot automatski ugradi u pom.xml dependency potreban za Tomcat, koji je prema zadanim postavkama verzija 7. U projektu se backend pokreće kao Tomcat server te kreće slušati nakon puštanja aplikacije u pogon.

```
: Tomcat started on port(s): 10000 (http) with context path '/api'  
: Started DogFriendlyApplication in 206.29 seconds (JVM running for 223.703)
```

Slika 5.10: Pokretanje Tomcat servera

Poviše je prikazan isječak koda koji se odvija u Renderu. Prikazuje početak slušanja Tomcata na vratima 10000 s konteksnim putem "/api" koji će se pobliže objasniti u potpoglavlju Konfiguracija frontenda za puštanje u pogon. Dodatne instalacije Tomcata nisu bili potrebne te ga iz tog razloga nećemo obrađivati.

Za CI (kontinuiranu integraciju) baze podataka i pojednostavljeni proces prerade promjene baze podataka koristi se Liquidbase²⁴. U našem projektu je bilo potrebno napraviti datoteku pod nazivom "changelog_master.xml" koju Liquidbase koristi kao konfiguracijsku datoteku. U njoj se definira način praćenja i upravljanja promjenama nad bazom podataka kod aplikacije koja je puštena u pogon, nakon što su nad bazom izvršeni određeni upiti.

²³<https://tomcat.apache.org/>

²⁴<https://www.liquibase.org/>

Za puštanje backenda u pogon je također važan Dockerfile²⁵. Docker je otvorena platforma za razvoj, isporuku i pokretanje aplikacija koja omogućuje odvajanje aplikacije od infrastrukture. Izgled našeg Dockerfilea je sljedeći:

```
// Container za izgradnju (build) aplikacije
FROM openjdk:17-alpine AS builder

//Kopiranje izvornog koda u container
COPY ../../mvnw .mvn
COPY ../../mvnw .
COPY ../../pom.xml .
COPY ../../src src
RUN chmod +x mvnw

// Pokretanje builda
RUN ./mvnw clean package

//Stvaranje containera u kojem će se vrtiti aplikacija
FROM openjdk:17-alpine

//Linux distro koji se koristi je Alpine, stoga se kao package manager koristi apk
//RUN apk install <nesto>

//Kopiranje izvršnog JAR-a iz build containera u izvršni container
COPY --from=builder target/*.jar /app.jar

//Izlaganje porta
EXPOSE 8080

//Naredba kojom se pokreće aplikacija
ENTRYPOINT ["java","-jar","/app.jar"]
```

Skraćeno: u dockerfileu definiramo što ćemo iz backend datoteke staviti u container koji se izvodi kao .JAR pogodan za izvođenje na poslužitelju.

Nadalje potrebna komponenta za povezivanje s frontendom, ali isto tako i puštanje aplikacije u pogon je Spring Boot anotacija @CrossOrigin. Cross-origin dijeljenje resursa (CORS) je standardni protokol koji definira interakciju između preglednika i poslužitelja za sigurno rukovanje HTTP zahtjevima s različitim izvorima. Jednostavno rečeno, cross-origin HTTP zahtjev je zahtjev prema određenom resursu koji se nalazi na različitom resursu, točnije domeni, protokolu i vratima, od onog klijenta koji izvršava zahtjev. U našem projektu se @CrossOrigin koristi u MapController.java i ProfileController.java unutar paketa hr.fer.progi.simplicity.controllers. Za lokalni rad koristio se:

@CrossOrigin(origins="http://localhost:3000") , dok se za globalni rad, to jest u aplikaciji puštenoj u pogon koristi se @CrossOrigin(origins="https://dogfriendly-

²⁵<https://docs.docker.com/get-docker/>

frontend.onrender.com"). Na navedenoj adresi se može dosegnuti frontend ppušten u pogon, no to će biti pobliže objašnjeno u dijelu puštanja aplikacije u pogon preko Rendera.

Konfiguraciju backenda za puštanje u pogon ćemo završiti dijelom koda application.properties iz našeg projekta.

```
//Port na kojem će se vrtiti api, obavezno izložiti, ovu varijablu koristi Render
server.port=${PORT:8080}

//Korijenska putanja ("prefiks") za sve zahtjeve na backend - preporuča se postaviti ovo
zbog proxy konfiguracije, ko je npr. u controlleru navedena putanja /test, moći će joj
se pristupiti pomoću putanje /api/test
server.servlet.context-path=/api

//Lokacija Liquibase master chagelog-a
spring.liquibase.change-log=classpath:/db/changelog/changelog-master.xml

//Konfiguracija baze podataka
spring.datasource.password=${DB_PASS:bazepodataka}
spring.datasource.username=${DB_USERNAME:postgres}
spring.datasource.url=${DB_URL:jdbc:postgresql://localhost:5432/DogFriendlyDB}
spring.datasource.driverClassName=${DB_DRIVER:org.postgresql.Driver}

spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=create-drop
```

Unatoč tome što se konfiguracija baze podataka za puštanje u pogon mogla zasebno odraditi, s obzirom da ona nema svoje zasebne upute za konfiguraciju i posve je isprepletena s backendom, obrađena je kroz backend.

Konfiguracija frontenda za spajanje s backendom

Kako bi se frontend uspješno povezao s backendom, što u konačnici omogućuje točan prijenos podataka od baze sve do korisnika, na frontendu je korišten axios²⁶. Axios je HTTP klijent temeljen na obećanjima za node.js i preglednik. Izomorfni je (može se izvoditi u pregledniku i u node.js s istim baznim kodom). Na strani poslužitelja koristi izvorni node.js HTTP modul, dok na klijentu (pregledniku) koristi XMLHttpRequests. XMLHttpRequest²⁷ objekti se koriste za interakciju sa serverima. Podaci se mogu dohvatiti s URL-a bez potrebe za punim osvježavanjem stranice, što omogućuje dijelovima stranice osvježavanje bez prekida korisnikovih radnji. U našem kodu se axios poziva u obliku: `export const AxiosInstance = axios.create(baseURL: 'https://dogfriendly-webservice-fc8h.onrender.com/api/')` U prijevodu, axiosom definiramo komunikaciju s backendom, to jest odakle očekujemo podatke i gdje ih šaljemo. "baseURL" je u ovom slučaju link preko kojeg se dohvaća backend koji je pušten u pogon. Dodatak na navedeni link, "/api", bi se mogao protumačiti kao dodatan stupanj sigurnosti, to jest specifikacija komunikacije. U lokalnoj verziji web aplikacije smo za "baseURL" koristili "`http://localhost:3000/api`". AxiosInstance konstanta je nova instanca axiosa koja se koristi za prilagođenu komunikaciju.

```
AxiosInstance.interceptors.request.use(async request => {
  const token = sessionStorage.getItem('token')
  if (token && request.headers) {
    request.headers['Authorization'] = 'Bearer ' + token;
  }
  return request
})
```

Navedeni kod prikazuju specifičan primjer prilagođene komunikacije iz našeg projekta. Interceptor²⁸ je metoda Axios klase koja presretne svaki zahtjev i/ili odgovor i odradio dio koda naveden u then bloku. U primjeru poviše interceptor presretne zahtjev prije nego što je poslan te u zaglavljivo umetne token (spremljen u SessionStorageu). Token služi kao zaštita podataka korisnika tijekom komunikacije. Kako se ne bi direktno slalo korisničko ime ili identifikator kao element raspoznaće koristi se generirani token. Token omogućuje frontendu i backendu informaciju o tome postoji li trenutni korisnik i tko je on. Svi budući zahtjevi (post,

²⁶<https://axios-http.com/docs/intro>

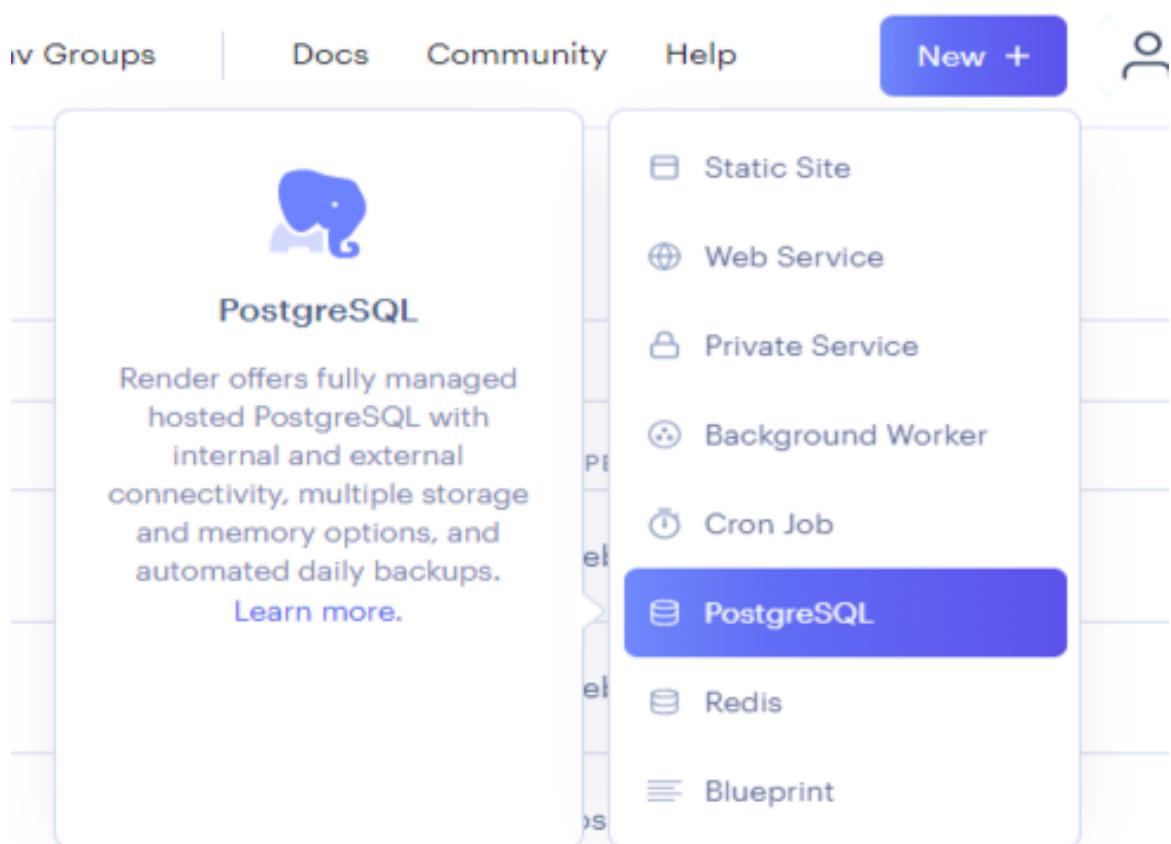
²⁷<https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest>

²⁸<https://axios-http.com/docs/interceptors>

put, get) idu preko AxiosInstancea.

Postavljanje varijable za puštanje aplikacije u pogon preko Rendera

Za puštanje web aplikacije u pogon preko Rendera je potreban korisnički račun. Jednom ulogirani korisnik ima više mogućnosti kreiranja, no mi ćemo koristiti "PostgreSQL" za kreiranje online baze podataka u oblaku i "Web Service" za puštanje backenda i frontenda u pogon (zasebno).



Slika 5.11: Kreiranje online baze podataka

Kreiranje baze u online oblaku je poprilično jednostavno. Unosi se ime baze, odabire regija u kojoj je baza aktivna, u našem slučaju Frankfurt (EU Central) i verzija PostgreSQLa.

New PostgreSQL

Name ⓘ

Database ⓘ

User

Region
The [region](#) where your Database runs. Services must be in the same region to communicate privately and you currently have services running in Frankfurt.

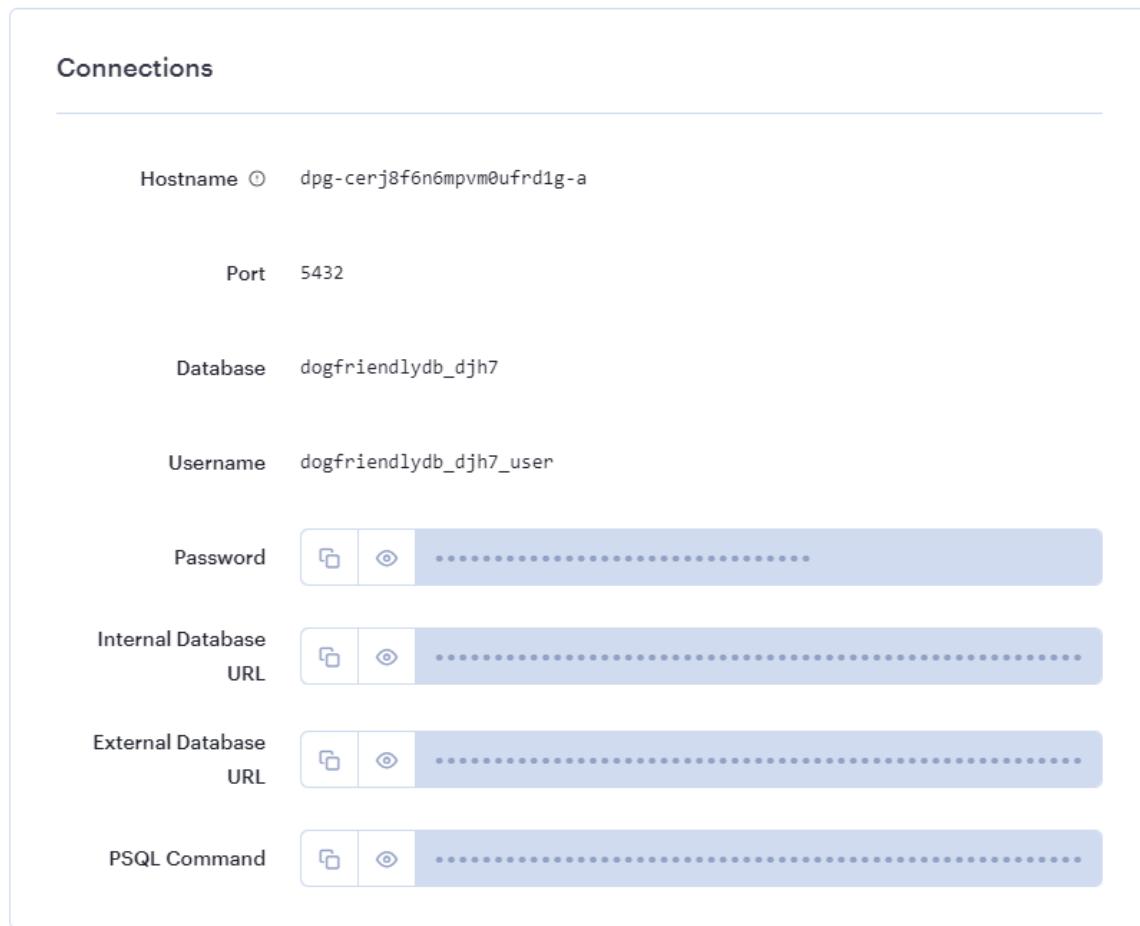
Frankfurt (EU Central)

PostgreSQL Version

Datadog API Key ⓘ

Slika 5.12: Unos parametara za izradu online baze podataka

Pod informacijama kreirane baze se nalaze osnovni podaci o kreaciji, ali i informacije o povezanosti baze.



Slika 5.13: Osnovni podaci o online bazi podataka

Na slici poviše su sve informacije potrebne za povezivanje "Web Service" bac-kenda s napravljenom bazom.

Nadalje kreiramo "Web Service". U našem slučaju, zbog jednostavnosti je naš Render račun povezan s GitLabom. Prilikom kreiranja Web Servicea daje nam se mogućnost povezivanja s jednim od korisnikovih Git repozitorija, ali i nekog jav-nog Git repozitorija. Nakon odabira repozitorija (u našem slučaju simplicity1/projektrepo) odabiremo ime našeg web servisa (dogfriendly_webservice), regiju (Frankfurt), granu s Git repozitorija (main), korijenski direktorij (IzvorniKod/backend), razvojno okruženje (Docker).

Name
A unique name for your web service.

dogfriendly_webservice

Region
The **region** where your web service runs. Services must be in the same region to communicate privately and you currently have services running in **Frankfurt**.

Frankfurt (EU Central)

Branch
The repository branch used for your web service.

main

Root Directory Optional
Defaults to repository root. When you specify a **root directory** that is different from your repository root, Render runs all your commands in the **specified directory** and ignores changes outside the directory.

Izvornikod/backend/DogFriendly

Environment
The runtime environment for your web service.

Docker

Slika 5.14: Odabir parametara kod kreiranje Web Servicea

Otvaramo napredne postavke i dodajemo varijable okruženja "DB_USERNAME", "DB_PASS" i "DB_URL" čije su vrijednosti redom "Username", "Password" i malo izmjenjeni "Internal Database URL" iz informacija baze podataka na slici 5.15. Za DB_URL se koristio format "jdbc:postgresql://hostname:port/database". Zadnji korak je pod "Dockerfile Path" unijeti put do našeg Dockerfilea u odnosu na korijen repozitorija, što je u ovom slučaju ./docker/maven/Dockerfile.

Advanced X

Use environment variables to store API keys and other configuration values and secrets. You can access them in your code like regular environment variables, for example with `os.getenv()` in Python or `process.env` in Node.

DB_USERNAME	<input type="text" value="value"/> Required	<input type="button" value="Generate"/>	<input type="button" value="Delete"/>
DB_PASS	<input type="text" value="value"/>	<input type="button" value="Generate"/>	<input type="button" value="Delete"/>
DB_URL	<input type="text" value="value"/>	<input type="button" value="Generate"/>	<input type="button" value="Delete"/>
Add Environment Variable			

You can store secret files (like `.env` or `.npmrc` files and private keys) in Render. These files can be accessed during builds and in your code just like regular files.

All secret files you create are available to read at the root of your repo (or Docker context). They are also available to load by absolute path at `/etc/secrets/<filename>`.

Add Secret File

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

<code>/healthz</code>

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

Izvornikod/backend/DogFriendly/	.
---------------------------------	---

Dockerfile Path
Path to your Dockerfile relative to the repository root. This is *not* relative to your Docker build context. For

Izvornikod/backend/DogFriendly/	<code>./docker/maven/Dockerfile</code>
---------------------------------	--

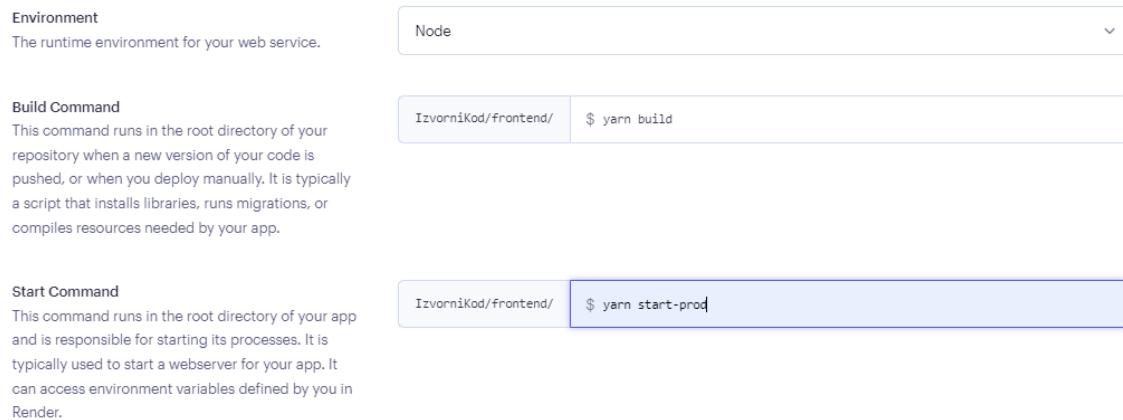
Slika 5.15: Dodavanje naprednih postavki kod kreiranja Web Servicea

Puštanje backenda u pogon započinje klikom na gumb "Create Web Service" nakon čega se gradi backend i pokreće "deploy". Ovisno o rezultatu deploja, status je ili "Deploy succeeded/Live" ili "Deploy failed". Među informacijama upravo napravljenog Web Servicea možemo pronaći link preko kojeg mu se može pristupiti što će biti potrebno za puštanje u pogon frontenda.



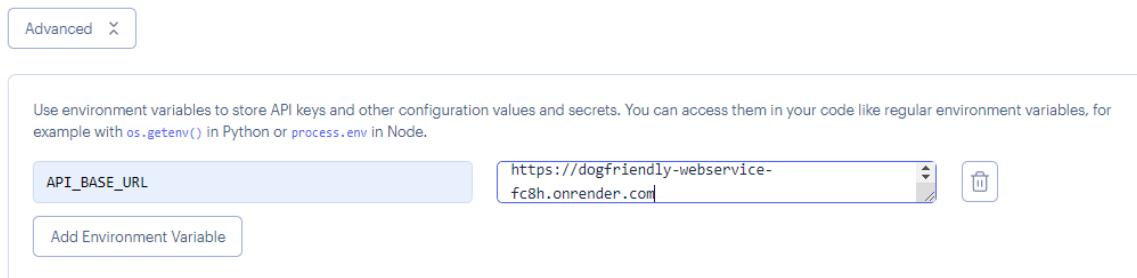
Slika 5.16: Link za backend

Puštanje frontenda u pogon je u početku isto. Kreiramo novi Web Service, odbiremo isti Git repozitorij, ime (dogfriendly_frontend), regiju, granu repozitorija i korijen direktorija (IzvorniKod/frontend). Kao okruženje izabiremo Node. Naredba za izgradnju je "yarn build", a naredbe za pokretanje "yarn start-prod".



Slika 5.17: Odabir parametara za deploy frontenda

Pod napredne postavke se dodaje varijabla okruženja "API_BASE_URL" čija je vrijednost prethodno spomenuta web adresa na kojoj se može dosegnuti backend.



Slika 5.18: Dodavanje naprednih postavki prilikom deploya frontenda

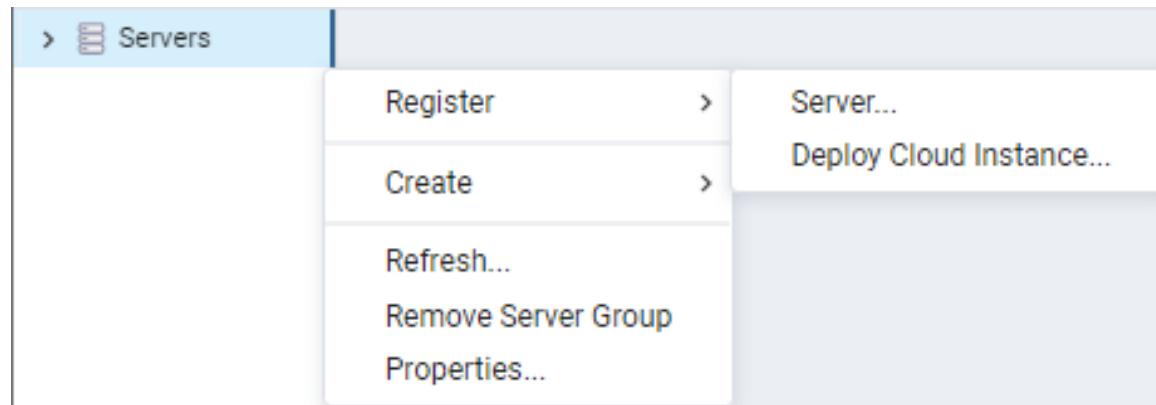
Sve što je preostalo je kliknuti "Create Web Service" gumb. Rezultati puštanja u pogon frontenda su identični backendu za uspjeh/neuspjeh. Frontend, kao i sveukupna funkcionalna web aplikacija (baza+backend+frontend) su pušteni u pogon i dostupni na internetskoj stranici čiji se link nalazi u podacima dogfriendly_frontend.



Slika 5.19: Link za frontend

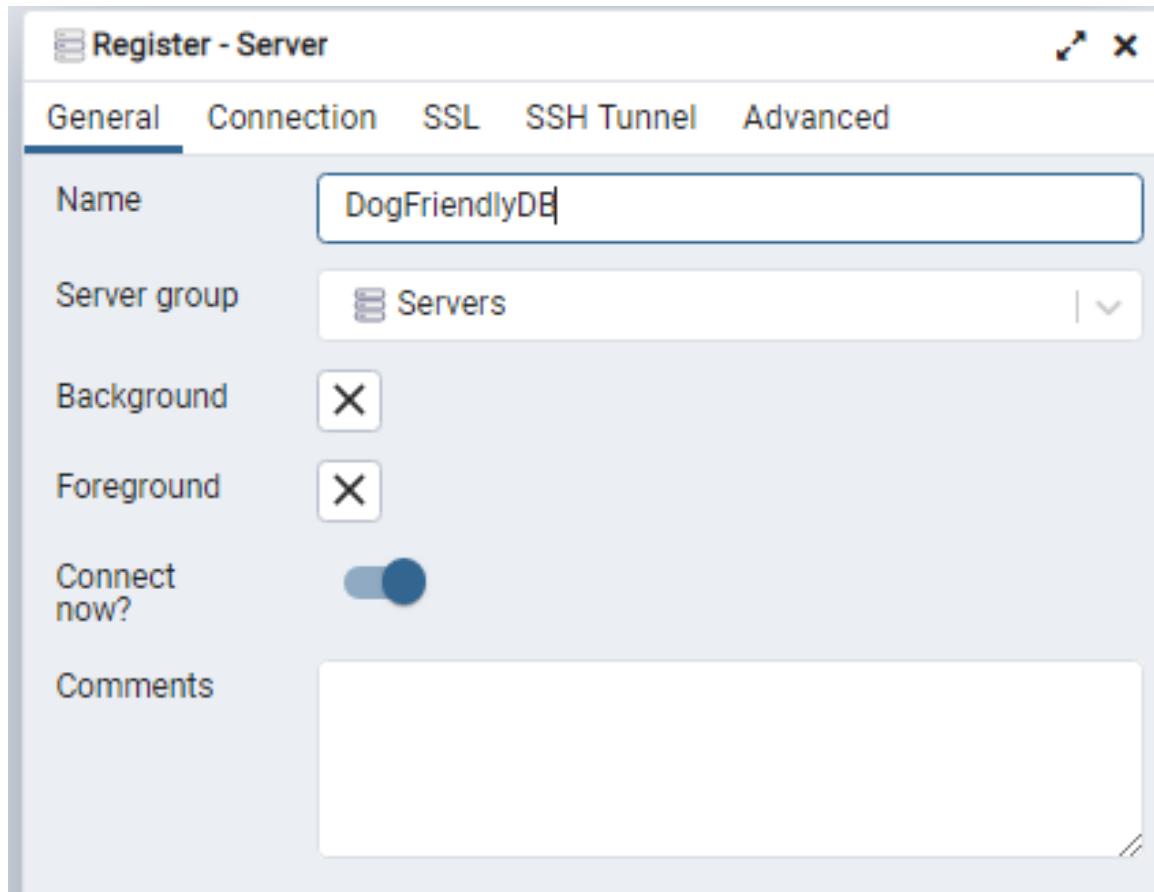
Prikaz podataka iz baze nakon puštanja u pogon

Nakon puštanja web aplikacije zanimaju nas podaci koji se nalaze u njoj. Nažalost Render nema mogućnost prikaza baze, tablica i njezinog sadržaja, već ju je potrebno povezati s nečime što može prikazati te podatke. U nastavku su kratke upute za prikaz podataka iz online oblak baze puštene u pogon preko pgAdmina. Prvi korak je provjeriti ima li korisnik najnoviju verziju pgAdmina. U slučaju da nema, potrebno ga je ažurirati. Nadalje, potrebno je ulogirati se, ako postoji ta mogućnost. Drugi korak je desni klik na Server pa Register onda opet Server.



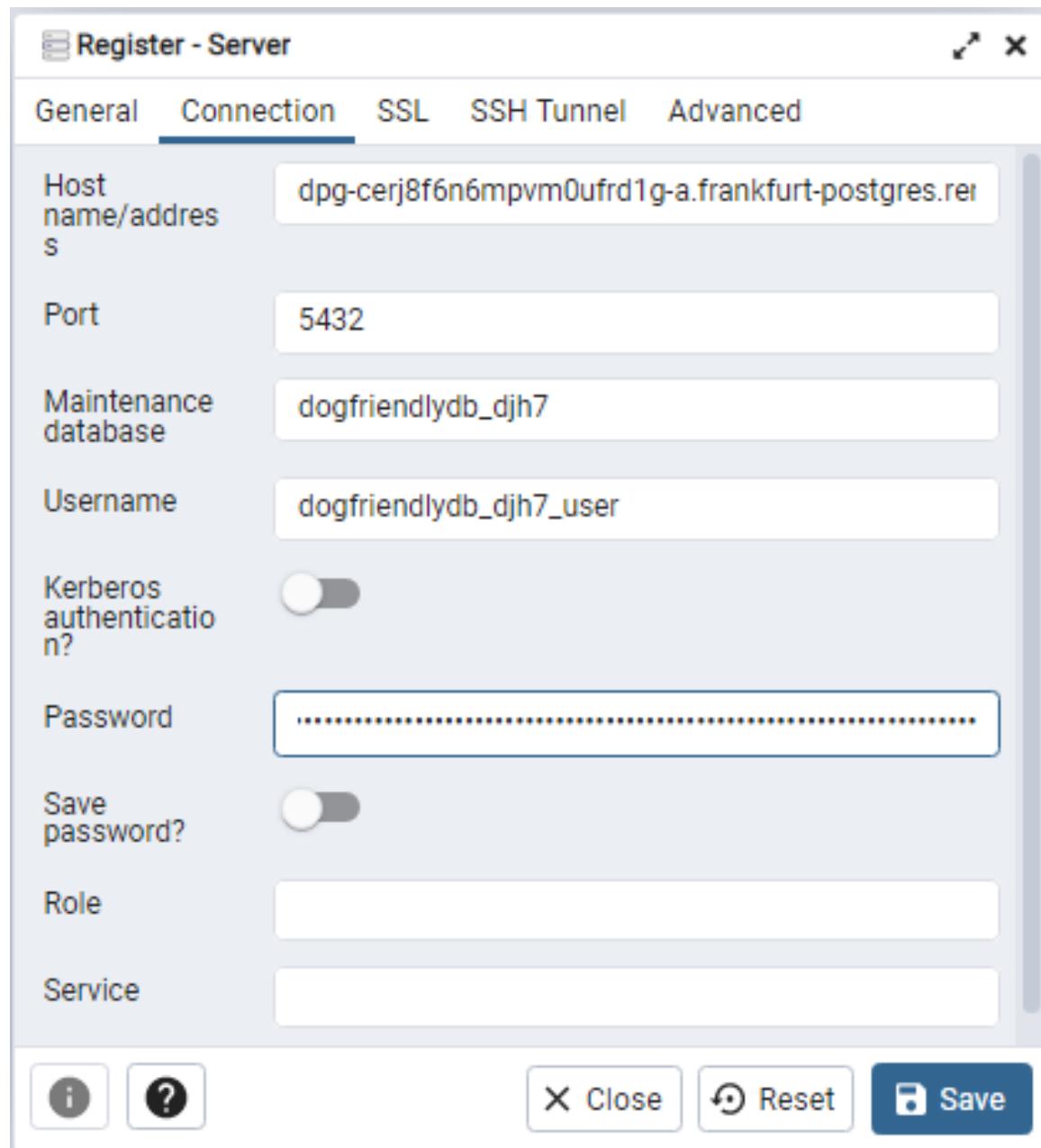
Slika 5.20: Prikaz koraka za login na bazu podataka

Treći korak: Kada se otvori novi prozor upisati podatke koje se mogu naći u informacijama baze puštene u pogon. Pod ime se upisuje ime baze.



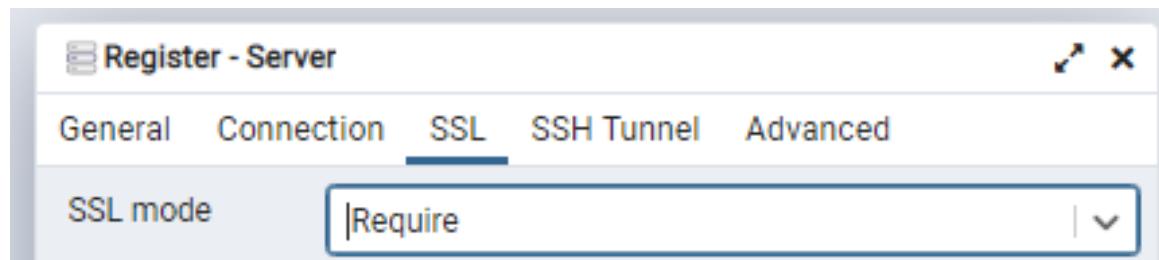
Slika 5.21: Upis imena baze podataka

U Connectionu unijeti Port, Username i Password identično podacima koji se mogući naći pod informacija baze puštene u pogon. Maintenance database je identičan Database, dok se Hostname/address vadi iz External Database URLa na sljedeći način: ako je format vanjskog URL baze postgres://aaa:bbb@ccc.frankfurt-postgres.render.com/ddd u Hostname se upisuje ccc.frankfurt-postgres.render.com.



Slika 5.22: Unos dodatnih parametara za prikaz baze

Na posljeku u SSL-u pod SSL mode postaviti "Require".



Slika 5.23: Postavljanje SSL-a na Require

Kliknuti Save i pristupiti bazi i podacima koji su spremljeni u nju.

6. Zaključak i budući rad

Tema našeg projektnog zadatka bila je izrada web aplikacije pod imenom Dog Friendly. Bit same aplikacije je bilo napraviti interaktivnu kartu kako bi ljubiteljima i vlasnicima pasa omogućili pregled prikladnih i neprikladnih lokacija na interaktivnoj karti i time olakšali kretanje i druženje. Izrada cijelog projekta je trajala 14 tjedana i bila je podijeljena u dva ciklusa.

Napredak je u prvom ciklusu bio nešto sporiji, a glavni razlog tome bilo je prvočno okupljanje tima, međusobno upoznavanje, upoznavanje sa samom temom projekta i upoznavanje s novim alatima koje smo trebali koristiti u izradi projekta. Nama je ta početna faza upoznavanja trajala relativno kratko zato što su svi članovi tima odmah shvatili važnost ovog projekta i znali su koji cilj trebamo postići. U timu smo se organizirali u dvije skupine, frontend i backend, što nam je kasnije uvelike pomoglo zato što su svi članovi znali svoju ulogu i znali su koji će ih posao kasnije čekati. Ta prva faza se fokusirala većinom na izradu projektne dokumentacije, a manje na izradu same aplikacije. Najveći tehnički izazovi koji su se javljali tada su bili korištenje Git-a (distribuirani sustav za upravljanje izvornim kodom) i korištenje LaTeX-a (programske jezik za pisanje dokumentacije), no te smo izazove savladali s vremenom kada smo se svi zajedno malo bolje upoznali s tim alatima. Najbitnija uloga uspješne izrade prvog dijela projekta bila je uloga voditeljice tima koja nas je konstantno obavještavala o našem napretku na projektu i koja nam je svima zadavala zadatke koji su nam bili u planu izrade. Dobro postavljeni temelji prve faze projekta poput kvalitetne izrade obrazaca uporabe i dobre organizacije tima su nam uvelike pomogli u izradi iduće faze projekta.

U drugom ciklusu je veći fokus bio na implementaciji same aplikacije. U ovoj fazi su svi članovi imali potpunu samostalnost nad svojim zadacima, svi članovi frontenda i backenda su bili u komunikaciji u dijelovima gdje je to bilo potrebno zato što je tako bilo najlakše ispuniti sve zahtjeve koje je aplikacija morala zadovoljiti. Tehnički izazovi koji su nam ovdje predstavljali najveće probleme su bili izrada interaktivne karte na frontendu i izrada programskih ispita na backendu pomoću kojih bi ispitali ponašanje nekog dijela sustava i pojedinih komponenti koje implementiraju neke temeljne funkcionalnosti aplikacije. Ti problemi su se

javljali zbog neiskustva članova tima u izradi tih specifičnih zadataka, no te smo probleme uspješno riješili uz pomoć asistentice koja je cijelo vrijeme nadzirala naš napredak na projektu. Uspjeh druge faze projekta bio je zbog dobre organizacije posla među članovima.

Znanja koja smo stekli na ovom projektu su mnogobrojna. Neka od praktičnih znanja su pisanje programske dokumentacije, korištenje UML modeliranja u projektu, oblikovanje arhitekture programske potpore prema objektno orijentiranoj paradigmi, analiza korisničkih zahtjeva i korištenje različitih razvojnih okruženja za izradu aplikacije. Također smo stekli mnogo iskustvenog znanja kao što su timski rad, organizacija i vođenje samog projekta, međusobna komunikacija svih članova tima, razmjenjivanje različitih mišljenja u pronalasku zajedničkog rješenja i mnoga druga. Jedino znanje koje bi bilo korisno znati prije samog početka izrade projekta bi bilo korištenje Git-a.

Ostvarili smo sve zahtijevane funkcionalnosti koje su se od nas tražile tako da možemo smatrati ovaj projekt uspješnim. Jedna od mogućih funkcionalnosti koje nismo implementirali, ali smatramo da bi mogla poboljšati izgled aplikacije bi bila mogućnost dodavanja slike obrta te bi se te slike onda prikazivale pod sekcijom preporučenih obrta. Osim svih funkcionalnosti koje smo ostvarili također smo naučili da je međusobna komunikacija u timu izuzetno važna što je i bio sami cilj projekta. Sve u svemu zadovoljni smo izradom našeg projekta i veselimo se upotrijebiti stečeno znanje i na neke druge projekte.

Popis literature

1. Programsko inženjerstvo, FER ZEMRIS, <http://www.fer.hr/predmet/proinz>
2. IntelliJ IDEA, <https://www.jetbrains.com/idea/>
3. React, <https://reactjs.org>
4. Selenium, <https://www.selenium.dev/>
5. JUnit4, <https://junit.org/junit4/>
6. Astah Community, <http://astah.net/editions/uml-new>
7. Baeldung, <https://www.baeldung.com/spring-tutorial>
8. Mapbox, <https://docs.mapbox.com/>
9. Github projekt, <https://github.com/visgl/react-map-gl>
10. W3Schools, <https://www.w3schools.com/>
11. MUI, <https://mui.com/>
12. Responsive Test Tool, <http://responsetesttool.com/>
13. Flat Icon, <https://www.flaticon.com/>
14. Png Tree, <https://pngtree.com/freepng>
15. Free Icons Png, <https://www.freeiconspng.com/images/close-icon>
16. Mappity, https://www.mappity.org/marker_icons
17. Nice Png, <http://astah.net/editions/uml-new>
18. Png Find, <https://www.pngfind.com>
19. Loading, <https://loading.io>
20. Dreams Time, <https://www.dreamstime.com>

21. IColorPalette, <https://icolorpalette.com/color/f3aba4>
22. UI Bakery, <https://uibakery.io/regex-library/phone-number-java>
23. AConvert, <https://www.aconvert.com/image/merge/>

Indeks slika i dijagrama

2.1 Primjer karte s tražilicom i prostorom za filtriranje.	7
3.1 Dijagram obrasca uporabe, upravljanje osobnim podatcima	20
3.2 Dijagram obrasca uporabe, funkcionalnost karte	21
3.3 Dijagram obrasca uporabe, funkcionalnost korisnika i vlasnika obrta	22
3.4 Sekvencijski dijagram za UC5	23
3.5 Sekvencijski dijagram za UC12	24
3.6 Sekvencijski dijagram za UC20	25
4.1 Dijagram baze podataka	32
4.2 Dijagram klase i metoda - Objekti za slanje podataka i controlleri . .	33
4.3 Dijagram klase i metoda - Objekti i enumeracije	34
4.4 Dijagram klase i metoda - Korisnici	35
4.5 Dijagram klase i metoda - Obrt i kartica	36
4.6 Dijagram klase i metoda - Mapa	37
4.7 Dijagram klasa i metoda - Sigurnost	38
4.8 Dijagram stanja za vlasnika obrta	39
4.9 Dijagram aktivnosti za dodavanje lokacije	40
4.10 Dijagram komponenti za Dog Friendly aplikaciju	41
5.1 Rezultati JUnit testova	53
5.2 Rezultat prvog Selenium testa	56
5.3 Rezultat drugog Selenium testa	59
5.4 Rezultat trećeg Selenium testa	60
5.5 Rezultat četvrтog Selenium testa	61
5.6 Rezultat petog Selenium testa	61
5.7 Rezultat šestog Selenium testa	62
5.8 Rezultat sedmog Selenium testa	62
5.9 Dijagram razmještaja	63
5.10 Pokretanje Tomcat servera	64
5.11 Kreiranje online baze podataka	68

5.12 Unos parametara za izradu online baze podataka	69
5.13 Osnovni podaci o online bazi podataka	70
5.14 Odabir parametara kod kreiranje Web Servicea	71
5.15 Dodavanje naprednih postavki kod kreiranja Web Servicea	72
5.16 Link za backend	72
5.17 Odabir parametara za deploy frontenda	73
5.18 Dodavanje naprednih postavki prilikom deploja frontenda	73
5.19 Link za frontend	74
5.20 Prikaz koraka za login na bazu podataka	74
5.21 Upis imena baze podataka	75
5.22 Unos dodatnih parametara za prikaz baze	76
5.23 Postavljanje SSL-a na Require	77
6.1 Prikaz aktivnosti na repozitoriju, grana develop, prvi dio	93
6.2 Prikaz aktivnosti na repozitoriju, grana develop, drugi dio	94
6.3 Prikaz aktivnosti na repozitoriju, grana devdoc, prvi dio	95
6.4 Prikaz aktivnosti na repozitoriju, grana devdoc, drugi dio	96

Dodatak: Prikaz aktivnosti grupe

Dnevnik sastajanja

1. sastanak

- Datum: 12. listopada 2022.
- Prisustvovali: T.Piveta, A.Žanko, L.Novosel, B.Perković, M.Hitl, N.Bukvić, D.Penava
- Teme sastanka:
 - izbor naziva tima
 - izbor između rada s asistentom ili pod vodstvom tvrtke CROZ

2. sastanak

- Datum: 15. listopada 2022.
- Prisustvovali: T.Piveta, A.Žanko, L.Novosel, B.Perković, M.Hitl, N.Bukvić, D.Penava
- Teme sastanka:
 - razgovor o predloženom zadatku (Dog Friendly)
 - konačan odabir alata i tehnologije (Spring i React)
 - podjela zadataka (proučiti rad u Gitu)

3. sastanak

- Datum: 20. listopada 2022.
- Prisustvovali: T.Piveta, A.Žanko, L.Novosel, B.Perković, M.Hitl, N.Bukvić, D.Penava
- Teme sastanka:
 - sastanak s asistentom i demonstratorom
 - asistent potvrđio dosadašnje odluke grupe o tehnologijama

4. sastanak

- Datum: 20. listopada 2022.
- Prisustvovali: T.Piveta, A.Žanko, L.Novosel, B.Perković, M.Hitl, N.Bukvić, D.Penava
- Teme sastanka:

- prijedlozi izgleda stranice
- razgovor o dalnjem toku projekta
- L.Novosel postavljen vođom backend-a, T.Piveta vođom frontend-a
- podjela zadataka (naučiti više o tehnologijama Spring i React)

5. sastanak

- Datum: 25. listopada 2022.
- Prisustvovali: T.Piveta, M.Hitl, D.Penava
- Teme sastanka:
 - dogovor o finalnom izgledu početne stranice
 - podjela zadataka (header, footer, dokumentiranje zahtjeva)
 - dogovor o strukturi grana na GitLabu

6. sastanak

- Datum: 25. listopada 2022.
- Prisustvovali: T.Piveta, A.Žanko, L.Novosel, B.Perković, N.Bukvić
- Teme sastanka:
 - dogovor o strukturi baze
 - podjela zadataka (user, owner, business, card)

7. sastanak

- Datum: 26. listopada 2022.
- Prisustvovali: T.Piveta, D.Penava, M.Hitl
- Teme sastanka:
 - odabir DogFriendly i Simplicity loga

8. sastanak

- Datum: 29. listopada 2022.
- Prisustvovali: T.Piveta, B.Perković, M.Hitl, D.Penava
- Teme sastanka:
 - pojašnjenje karte u dokumentaciji
 - razjašnjenje problema u dokumentaciji

9. sastanak

- Datum: 29. listopada 2022.
- Prisustvovali: T.Piveta, A. Žanko, L.Novosel, N.Bukvić
- Teme sastanka:
 - rasprava o načinu registracije vlasnika obrta (3-step registration ili produžetak forme za registraciju)
 - razgovor o sigurnosti web aplikacije

10. sastanak

- Datum: 31. listopada 2022.
- Prisustvovali: A.Žanko, T.Piveta, M.Hitl, D.Penava, B.Peković, L.Novosel, N.Bukvić
- Teme sastanka:
 - dogovor o stranici za registraciju (izgled i komunikacija fronta i backa)
 - podjela zadataka

11. sastanak

- Datum: 2. studenog 2022.
- Prisustvovali: A.Žanko, T.Piveta, B.Peković, L.Novosel, N.Bukvić
- Teme sastanka:
 - dogovor o stranici za registraciju (izgled i komunikacija fronta i backa)
 - podjela zadataka

12. sastanak

- Datum: 5. studenog 2022.
- Prisustvovali: L.Novosel, N.Bukvić, B.Perković
- Teme sastanka:
 - spajanje grana
 - zajedničko povezivanja i nadopuna koda

13. sastanak

- Datum: 5. studenog 2022.
- Prisustvovali: L.Novosel, N.Bukvić
- Teme sastanka:
 - dokumentacija (tablice i dijagram baza, dijagram klasa)

14. sastanak

- Datum: 5. studenog 2022.
- Prisustvovali: T.Piveta, M.Hitl, D.Penava
- Teme sastanka:
 - revizija rada
 - razgovor o budućim zadacima

15. sastanak

- Datum: 8. studenog 2022.
- Prisustvovali: A.Žanko, T.Piveta, M.Hitl, D.Penava, B.Perković, L.Novosel, N.Bukvić
- Teme sastanka:

- evaluacija dosadašnjeg rada
- podjela zadataka (stranica korisnika, karta, autentifikacija na frontu, spajanje s bazom podataka)

16. sastanak

- Datum: 10. studenog 2022.
- Prisustvovali: A.Žanko, T.Piveta, M.Hitl, D.Penava, B.Perković, L.Novosel, N.Bukvić
- Teme sastanka:
 - sastanak s asistenticom - komentiranje dokumentacije, dijagrama klasa, obrazaca upotrebe i samog koda
 - preinake baze podataka i stranice za registraciju
 - podjela zadataka (registracija, dokumentacija)

17. sastanak

- Datum: 10. studenog 2022.
- Prisustvovali: T.Piveta, L.Novosel, N.Bukvić
- Teme sastanka:
 - uspješna registracija korisnika
 - definiranje problema kod registracije vlasnika obrta

18. sastanak

- Datum: 11. studenog 2022.
- Prisustvovali: L.Novosel, M.Hitl, D.Penava, B.Perković, T.Piveta
- Teme sastanka:
 - preinake backa za front
 - uspješna registracija korisnika
 - definiranje problema kod registracije vlasnika obrta

19. sastanak

- Datum: 14. studenog 2022.
- Prisustvovali: L.Novosel, M.Hitl, D.Penava, B.Perković, T.Piveta, A.Žanko, N.Bukvić
- Teme sastanka:
 - razgovor o web aplikaciji, svim promjenama i problemima
 - podjela zadataka (dokumentacija, aplikacija)

20. sastanak

- Datum: 17. studenog 2022.
- Prisustvovali: L.Novosel, M.Hitl, D.Penava, B.Perković, T.Piveta, A.Žanko,

N.Bukvić

- Teme sastanka:

- demonstracija generičkih funkcionalnosti pred asistenticom i demonstratoricom

21. sastanak

- Datum: 18. studenog 2022.
- Prisustvovali: L.Novosel, M.Hitl, D.Penava, B.Perković, T.Piveta, A.Žanko, N.Bukvić
- Teme sastanka:
 - završni deploy aplikacije
 - popunjavanje dokumentacije za predaju

22. sastanak

- Datum: 5. prosinca 2022.
- Prisustvovali: L.Novosel, M.Hitl, D.Penava, B.Perković, T.Piveta, A.Žanko, N.Bukvić
- Teme sastanka:
 - objašnjavanje koda i komentiranje projekta

23. sastanak

- Datum: 15. prosinca 2022.
- Prisustvovali: L.Novosel, M.Hitl, D.Penava, B.Perković, T.Piveta, A.Žanko, N.Bukvić
- Teme sastanka:
 - dogovor oko markera na mapi
 - generalizacija korisnika
 - lokacija i ocjene na backendu

24. sastanak

- Datum: 20. prosinca 2022.
- Prisustvovali: L.Novosel, M.Hitl, D.Penava, B.Perković, T.Piveta, A.Žanko, N.Bukvić
- Teme sastanka:
 - podjela posla za kartu i profil

25. sastanak

- Datum: 21. prosinca 2022.
- Prisustvovali: L.Novosel, M.Hitl, D.Penava, B.Perković, T.Piveta, A.Žanko, N.Bukvić

- Teme sastanka:

- dogovor oko rada za praznike vezano uz dokumentaciju i kod

26. sastanak

- Datum: 5. siječnja 2023.

- Prisustvovali: L.Novosel, M.Hitl, D.Penava, B.Perković, T.Piveta, A.Žanko, N.Bukvić

- Teme sastanka:

- posljednji detalji na frontendu
 - plan testiranja na backendu

27. sastanak

- Datum: 13. siječnja 2023.

- Prisustvovali: L.Novosel, M.Hitl, D.Penava, B.Perković, T.Piveta, A.Žanko, N.Bukvić

- Teme sastanka:

- provjere dokumentacije i popisivanje aktivnosti
 - deployment

Tablica aktivnosti

Doprinosi članova navedeni u satima.

	Timoteja Piveta	Ana Žanko	Luka Novosel	Nikola Bukvić	Bruno Perković	Mateo Hitl	Domagoj Penava
Upravljanje projektom	40		10				
Opis projektnog zadatka	3.5						
Funkcionalni zahtjevi					1	1	
Opis pojedinih obrazaca					7	7	
Dijagram obrazaca					1.5	1.5	
Sekvencijski dijagrami					2	2	
Opis ostalih zahtjeva					1	1	
Arhitektura i dizajn sustava				2			
Baza podataka		2.5	1		1.5		
Dijagram razreda		2.5	4	4	5		
Dijagram stanja						1.5	1.5
Dijagram aktivnosti						1.5	1.5
Dijagram komponenti						1.5	1.5
Korištene tehnologije i alati	3						
Ispitivanje programskog rješenja		1.5	2	2			
Dijagram razmještaja						1.5	1.5
Upute za puštanje u pogon	6		1				
Dnevnik sastajanja	3						
Zaključak i budući rad					1.5		

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

	Timoteja Piveta	Ana Žanko	Lukta Novosel	Nikola Bukvić	Bruno Perković	Mateo Hitl	Domagoj Penava
Popis literature	0.5	0.5				0.5	
Revizija dokumentacije		7		0.5		3	3
Zaglavljе	1					7.5	
Podnožje	0.5						3
Početna stranica	4						
Stranica s kartom	39				59		
Stranica s podacima korisnika						11	14
Stranice za registraciju	23					5.5	
Stranica za prijavu	3.5					2.5	2
Stranica za uređivanje podataka						6	3
Autentifikacija	5.5						
Responzivni dizajn	8				5	9	12
Čišćenje i popravci front-end-a	4.5				11.5		
Korisnik		11	10				
Vlasnik obrta (generalizacija)		9					
Obrt		2	3	10			0.5
Kartica		2	6	6	6		
Lokacija		8					
Mapa		2					
Sigurnost		30					
Registracija		15					

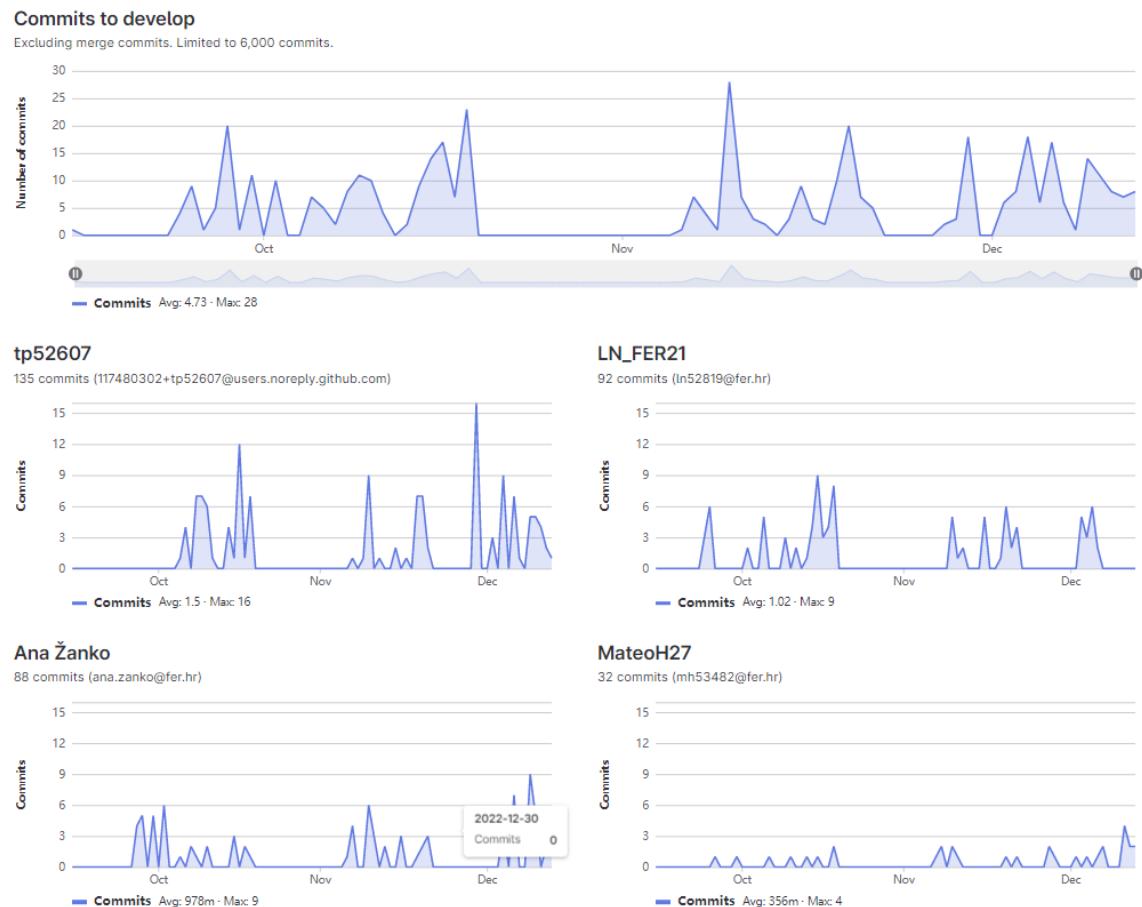
Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

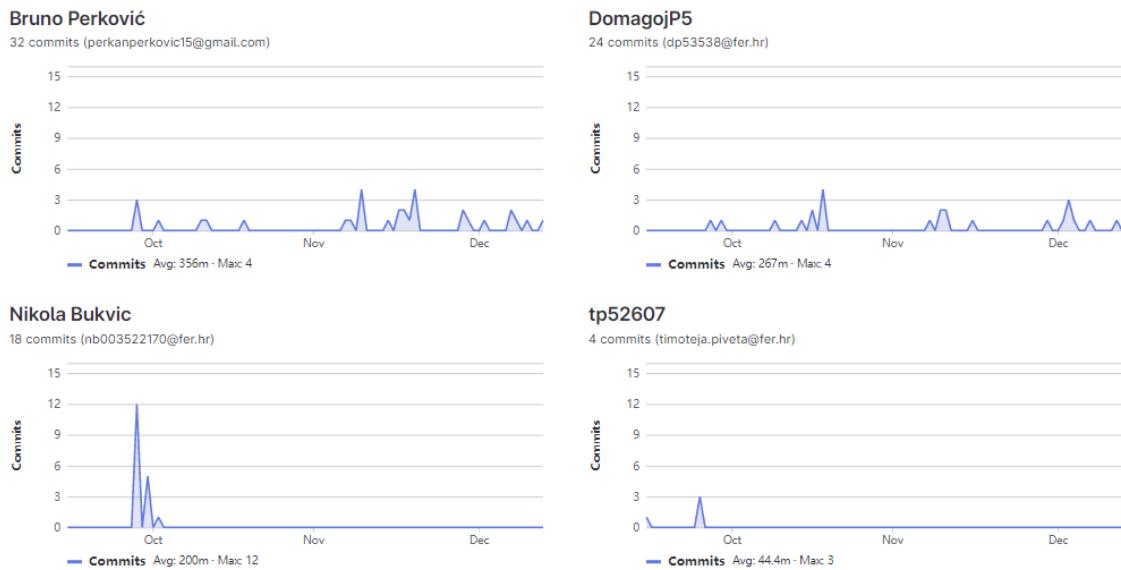
	Timoteja Piveta	5	Ana Žanko	Lukka Novosel	Nikola Bukvić	Bruno Perković	Mateo Hitl	Domagoj Penava
Prijava								
Prikaz profila			20					
E-mail za registraciju		1	7	4				
Baza podataka		1	7	6	2			
Povezivanje front-a i back-a	8	2	5					
Upravljanje greškama		1	0.5	0.5				
Postavljanje aplikacije u pogon	13	6	14	7.5				
Ispitivanje		5	8	12				
Čišćenje i popravci backend-a		11						

Dijagrami pregleda promjena

Prikaz aktivnosti na repozitoriju, grana develop

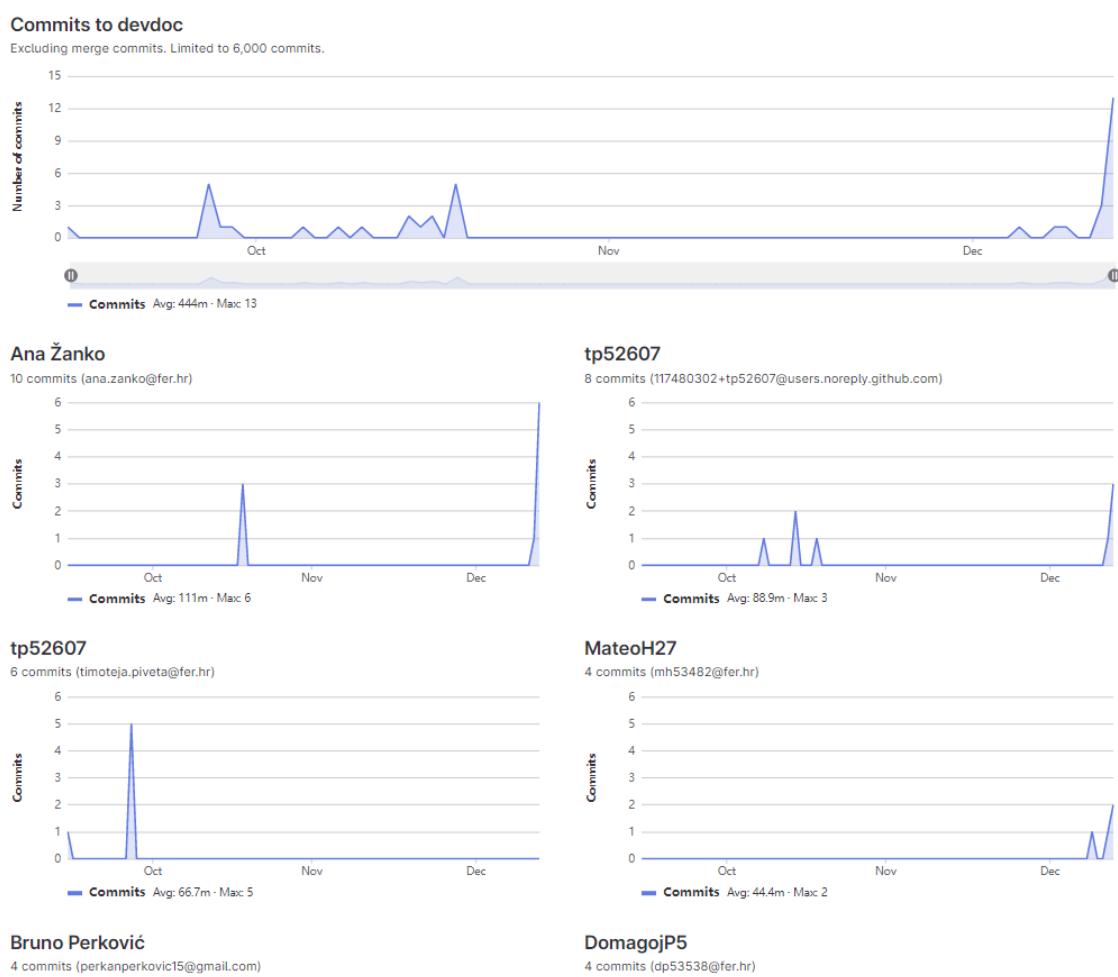


Slika 6.1: Prikaz aktivnosti na repozitoriju, grana develop, prvi dio

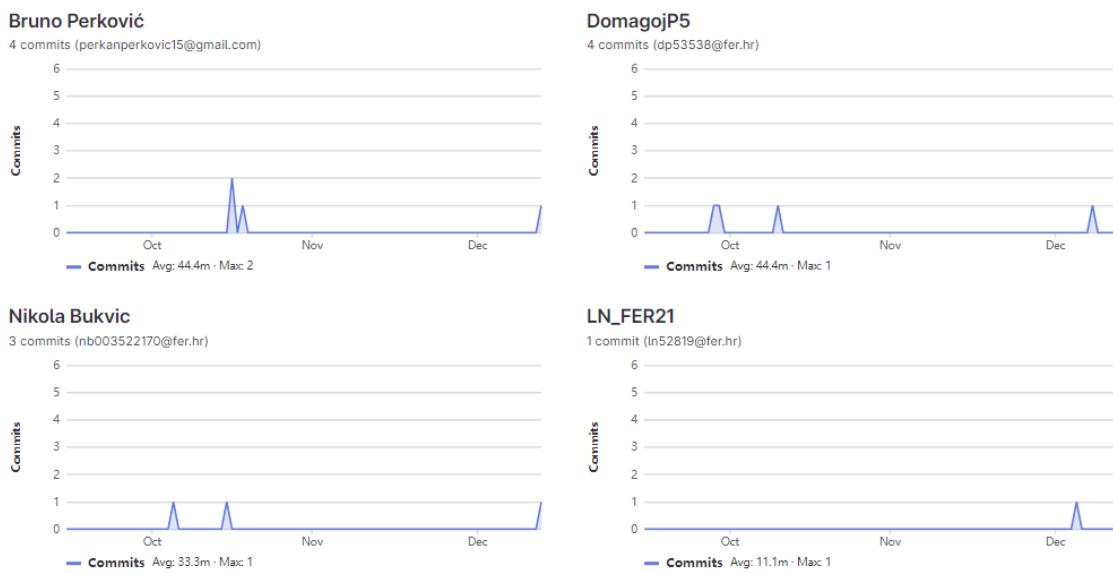


Slika 6.2: Prikaz aktivnosti na repozitoriju, grana develop, drugi dio

Prikaz aktivnosti na repozitoriju, grana devdoc



Slika 6.3: Prikaz aktivnosti na repozitoriju, grana devdoc, prvi dio



Slika 6.4: Prikaz aktivnosti na rezitoriju, grana devdoc, drugi dio