Programsko inženjerstvo

Ak. god. 2023./2024.

Nestali ljubimci

Dokumentacija, Rev. 2.

Grupa: *A-Team*

Voditelj: *Patrik Marinić*

Datum predaje: *19. siječnja 2024.*

Nastavnik: *Alan Jović*

Sadržaj

1. Dnevnik promjena dokumentacije

Rev.	Opis promjene/dodatka	Autori	Datum
0.1	Napravljen predložak.	Patrik Marinić	25.10.2023.
0.2	Dodan dijagram obrasca uporabe za funkcionalnosti korisnika.	Andrija Krklec	27.10.2023.
0.3	Dodan opis projektnog zadatka.	Robert Vitaliani	28.10.2023.
0.4	Dodani opisi obrazaca uporabe. Dodani dionici i aktori. Dodani funkcionalni zahtjevi.	Luka Raić, Vedran Mesar, Luka Rogoz	29.10.2023.
0.5	Sekvencijski dijagrami.	Anđelko Prskalo	31.10.2023.
0.6	Dodani ostali zahtjevi i arhitektura sustava.	Andrija Krklec	04.11.2023.
0.7	Dodan dijagram baze podataka.	Luka Rogoz	06.11.2023.
0.8	Dodan opis tablica baze te ispravljen dijagram baze.	Luka Rogoz	8.11.2023.
0.9	Izmjena opisa projektnog zadatka i popravljen izgled dokumenta.	Patrik Marinić	12.11.2023.

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

Rev.	Opis promjene/dodatka	Autori	Datum
0.10	Dodani dijagrami razreda.	Luka Raić	16.11.2023.
0.11	Dodan opis dijagrama razreda.	Vedran Mesar	17.11.2023.
0.12	Izmijenjeni obrasci uporabe.	Andrija Krklec	17.11.2023.
0.13	Popravljene greške u dijagramima razreda.	Luka Raić	17.11.2023.
1.0	Korigiranje teksta i provjera dokumentacije.	Robert Vitaliani	17.11.2023.
1.1	Dodani dijagram stanja i dijagram aktivnosti.	Luka Rogoz	17.12.2023.
1.2	Dodani dijagram komponenti i dijagram razmještaja.	Vedran Mesar	21.12.2023.
1.3	Ažurirani dijagrami razreda. Korekcija sekvencijskih dijagrama.	Luka Raić, Anđelko Prskalo	14.1.2024.
1.4	Izmjena opisa projektnog zadatka.	Robert Vitaliani	15.1.2024.
1.5	Dodane korištene tehnologije i alati. Ažurirana tablica aktivnosti. Dodan zaključak i budući rad.	Robert Vitaliani	17.1.2024.
1.6	Dodane upute za puštanje aplikacije u pogon.	Anđelko Prskalo	18.1.2024.

2. Opis projektnog zadatka

Cilj ovog projekta je razviti programsku podršku za stvaranje web aplikacije „Nestali ljubimci“ koja će korisniku omogućiti pregledavanje, pretragu i objavljivanje oglasa o nestalim kućnim ljubimcima, kao i pružiti resurse za kontaktiranje skloništa za životinje i druge korisnike u svrhu pronalaženja izgubljenih ljubimaca, bilo da se radi o vlastitom ili tuđem ljubimcu. Ovom web aplikacijom smanjit će se pojava nepraktičnog oglašavanja nestanaka, opažanja i pronalaska s preciznim podacima kućnih ljubimaca koja se danas vrlo često može pronaći na raznim internetskim platformama poput društvenih mreža, stranica za oglašavanje, foruma itd. Web aplikacija će biti intuitivna i korisna, pružajući vlasnicima kućnih ljubimaca, skloništima za životinje i ostalim uključenim osobama pomoć u rješavanju ovih stresnih situacija. Dodatno, web aplikacija će biti optimizirana za mobilne uređaje kako bi pružila korisnicima najbolje iskustvo. Ovakav razvoj web aplikacije pružit će svim dionicima brži pristup i reakciju, što često igra ključnu ulogu u uspješnom završetku potrage za izgubljenim kućnim ljubimcem.

Prilikom pokretanja sustava prikazuju se oglašeni nestali kućni ljubimci.

Neregistrirani korisnik ima pristup funkcionalnosti pretraživanja oglašenih nestalih kućnih ljubimaca prema raznim kategorijama podataka o ljubimcu, kao i pretraživanju skloništa za životinje putem naziva skloništa. Pretraživanje nestalih kućnih ljubimaca prema kategorijama obuhvaća sljedeće informacije o ljubimcu: vrstu, ime na koje se ljubimac odaziva, boju, starost i naziv skloništa(ako je oglas objavilo sklonište). Sve navedene kategorije o nestalom ljubimcu mogu se detaljnije pregledati klikom na određeni oglas. Osim ovih podataka, oglas sadrži i kontakt podatke korisnika koji se automatski povlače iz korisničkih podataka danih pri registraciji (poput e-pošte, telefona te posebno za skloništa - naziv skloništa). Dodatno, odabirom određenog kućnog ljubimca omogućava se detaljniji uvid u informacije o njemu, kao i pregled komunikacije vezane uz potragu za ljubimcem. Neregistriranom korisniku je omogućeno prijavljivanje u sustav s postojećim računom (potrebno je upisati korisničko ime i lozinku) ili kreiranjem novog računa. Za kreiranje novog računa potrebni su sljedeći podaci:

- korisničko ime

- lozinka
- ime
- prezime
- broj telefona
- e-pošta

Registracijom u sustav, korisniku se dodjeljuju prava registriranog korisnika. Prava registriranog korisnika uključuju sva prava neregistriranog korisnika, te dodatno:

- postavljanje oglasa o nestalom kućnom ljubimcu
- uklanjanje oglasa o nestalom kućnom ljubimcu
- izmjena oglasa o nestalom kućnom ljubimcu
- sudjelovanje u komunikaciji oko potrage za ljubimcem

Ako registrirani korisnik odluči postaviti oglas obavezan je unijeti niz kategorija podataka o ljubimcu, uključujući vrstu, ime na koje se ljubimac odaziva, datum i vrijeme nestanka, lokaciju nestanka (uz korištenje vanjske usluge za geolociranje, poput OpenStreetMap-a), boju, starost, tekstualni opis i do tri slike. Registrirani korisnici imaju mogućnost komunikacije putem poruka u svrhu potrage za izgubljenim ljubimcem. Poruke mogu sadržavati tekst, slike i geolokaciju (putem vanjske usluge), uz jasno istaknute kontakt informacije korisnika. Registrirani korisnik može ukloniti oglas koje je postavio. Ako korisnik ukloni svoj oglas, taj oglas i sva njegova komunikacija nestat će iz popisa vidljivih oglasa, međutim, oglas će i dalje ostati sačuvan u bazi podataka. Registriranim korisnicima omogućeno je uređivanje oglasa s mogućnošću izmjene svih kategorija podataka, uključujući promjenu kategorije oglasa. Raspoložive kategorije oglasa uključuju:

1. ljubimac je nestao i za njim se traga
2. ljubimac je sretno pronađen
3. ljubimac nije pronađen i za njim se više aktivno ne traga
4. ljubimac je pronađen u nesretnim okolnostima

Svaka izmjena kategorije oglasa u onu koja nije da se za ljubimcem aktivno traga prebacuje oglas automatski u popis neaktivnih oglasa, koji mogu pretraživati samo registrirani korisnici.

Skloništa za životinje su specijalni tip registriranih korisnika koji, osim funkcionalnosti koji imaju ostali registrirani korisnici, imaju dodatnu mogućnost oglašavanja životinja koje su pronašli i koje se nalaze u njihovom prostoru. Takvi oglasi imaju dodatnu kategoriju – u skloništu, pa bi njihove kategorije oglasa bile:

1. ljubimac je nestao i za njim se traga
2. ljubimac je sretno pronađen
3. ljubimac nije pronađen i za njim se više aktivno ne traga
4. ljubimac je pronađen u nesretnim okolnostima
5. ljubimac je u skloništu

Ova web aplikacija podržava tri tipa korisnika te u aplikaciji nema podržane uloge administratora koji bi se trebao brinuti o administraciji podataka(oglasa, registracije i korisničkih podataka). Prema razini ovlasti koju imaju u web aplikaciji, korisnici su podijeljeni u tri kategorije, rangirane od najviše razine ovlasti do najniže razine ovlasti:

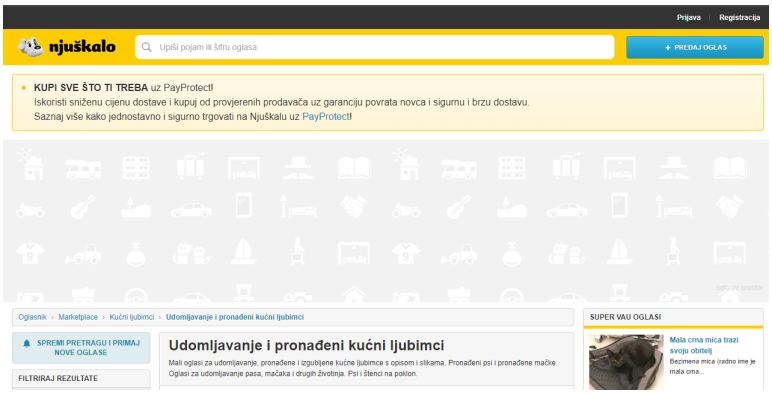
1. skloništa za životinje
2. registrirani korisnici
3. neregistrirani korisnici

Svaka registracija korisnika i njihovih korisničkih podataka bit će zabilježena u bazi podataka, s naglaskom na jedinstvenost korisničkih imena i korisničkih podataka. To znači da neće biti moguće imati dva korisnika s istim korisničkim imenom ili e-poštom. Ova jedinstvenost također se primjenjuje na skloništa za životinje. Sustav će podržavati rad više korisnika u stvarnom vremenu.

Ova web aplikacija pružit će značajnu korist svim vlasnicima kućnih ljubimaca. Svojim funkcionalnostima ne samo da će povećati šanse za pronalazak izgubljenih ljubimaca, već će povezati i sve druge ljude koji imaju jednak cilj, a to je pronalazak izgubljenih ljubimaca. Svi smo svjesni da je gubitak ljubimca izuzetno emotivno zahtjevan događaj. Ovim projektom teži se pronalasku rješenja koje će smanjiti stresne situacije i pružiti podršku osobama koje se nalaze u potrebi da pronađu svoje izgubljene ljubimce. Naglašavamo da će svaki pokušaj zloupotrebe web aplikacije biti strogo kažnjen, s potencijalnim dodatnim sankcijama ako se to smatra potrebnim.

Do sada nije postojala jasna i učinkovita web aplikacija za pronalazak nestalih ljubimaca, već samo postoje neka potencijalna i neučinkovita rješenja poput foruma, oglasnih stranica ili pak društvenih stranica. Neka od tih rješenja prikazana su u nastavku.

Slika 2.1: Facebook



Slika 2.2: Njuškalo

Web aplikacija za pronalazak nestalih ljubimaca će predstavljati puno učinkovitije rješenje u kojem će se korisnici lako i brzo snalaziti. Dodatno, važno je istaknuti da će svaki oglas za izgubljenog ljubimca, s odgovarajućom komunikacijom, biti tretiran kao samostalna i odvojena cjelina, bez međusobnih preklapanja s drugim oglasima. Tako će se ostvariti puno jednostavnija, brža i učinkovitija pretraga svih obavljenih oglasa.

Prilagodbe rješenja i nadogradnje projektnog zadatka bit će moguće u budućnosti, uz naglasak na korisničkom zadovoljstvu i njihovim mišljenjima. Korisnike se potiče da iznose svoja moguća rješenja i sugestije kako bi se unaprijedila web aplikacija te postala sve korisnija, učinkovitija i uspješnija u pomoći korisnicima u potrazi za izgubljenim ljubimcima.

3. Specifikacija programske potpore

3.1 Funkcionalni zahtjevi

Dionici:

1. Vlasnik kućnog ljubimca
2. Sklonište za životinje
3. Ostali korisinici aplikacije(klijent)
4. Razvojni tim

Aktori i njihovi funkcionalni zahtjevi:

1. Neregistrirani korisnik (inicijator) može:
   (a) pretraživati oglašene nestale kućne ljubimce i skloništa za životinje
   (b) detaljno pregledavati informacije o nestalim kućnim ljubimcima
   (c) pregledavati komunikaciju oko potrage za ljubimcem
2. Vlasnik kućnog ljubimca (inicijator) može:
   (a) postaviti oglas o nestalom kućnom ljubimcu
   (b) sudjelovati u komunikaciji oko potrage za ljubimcem
   (c) ukloniti oglas o nestalom kućnom ljubimcu
   (d) izmijeniti oglas o nestalom kućnom ljubimcu
   (e) pretraživati neaktivne oglase o nestalim kućnim ljubimcima
   (f) sve što i neregistrirani korisnik može
3. Sklonište za životinje (inicijator) može:
   (a) oglašavati životinje koje je pronašlo i koje se nalazi u njegovom prostoru
   (b) postaviti posebnu kategoriju (”u skloništu”) kod postave oglasa
   (c) sve što i vlasnik kućnog ljubimca može
4. Baza podataka (sudionik):
   (a) pohranjuje sve podatke o registriranim korisnicima, njihovim oglasima i komunikacijama oko potrage za ljubimcem

(b) pohranjuje sve podatke o skloništima za životinje, njihovim oglasima i komunikacijama oko potrage za ljubimcem

3.1.1 Obrasci uporabe

Opis obrazaca uporabe

UC1 - Pregled oglasa nestalih ljubimaca

- **Glavni sudionik:** Neregistrirani korisnik, Vlasnik kućnog ljubimca, Sklonište za životinje
- **Cilj:** Pregledavanje oglašenih nestalih kućnih ljubimaca i skloništa za životinje
- **Sudionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
  1. Nestali ljubimci su prikazani prilikom učitavanja aplikacije
  2. Korisnik odabire nestalog ljubimca
  3. Prikazuje se detaljniji pregled informacija i pregled komunikacija oko potrage

UC2 - Registracija

- **Glavni sudionik:** Neregistrirani korisnik
- **Cilj:** Stvoriti korisnički račun za pristup sustavu
- **Sudionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
  1. Korisnik odabire opciju za registraciju
  2. Korisnik unosi vlastite podatke
  3. Podaci se spremaju u bazu podataka
  4. Korisnik prima obavijest o uspješnoj registraciji
- **Opis mogućih odstupanja:**

  2.a Podaci koje je korisnik unio odgovaraju postojećem korisničkom računu
  1. Pojavljuje se poruka o neuspješnoj registraciji i zahtjev za ponovnim unosom podataka
  2. Korisnik mijenja podatke ili odustaje od registracije

  2.b Korisnik nije unio sve zahtijevane podatke
  1. Pojavljuje se poruka o neuspješnoj registraciji i zahtjev za ponovnim unosom podataka
  2. Korisnik unosi zahtijevane podatke ili odustaje od registracije

UC3 - Pretraživanje oglasa nestalih ljubimaca

- **Glavni sudionik:** Neregistrirani korisnik, Vlasnik kućnog ljubimca, Sklonište za životinje
- **Cilj:** Pretraživanje nestalih ljubimaca po svim dostupnim kategorijama podataka
- **Sudionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
  1. Korisnik odabire kategorije podataka po kojima obavlja pretraživanje
  2. Korisnik unosi podatke (filtre) za odabrane kategorije
  3. Prikazuje se popis filtriranih oglasa

UC4 - Prijava u sustav

- **Glavni sudionik:** Vlasnik kućnog ljubimca, Sklonište za životinje
- **Cilj:** Dobiti pristup korisničkom sučelju
- **Sudionici:** Baza podataka
- **Preduvjet:** Registracija
- **Opis osnovnog tijeka:**
  1. Unos korisničkog imena ili e-mail adrese i lozinke
  2. Provjera ispravnosti unesenih podataka
  3. Pristup korisničkom sučelju web aplikacije
- **Opis mogućih odstupanja:**

  2.a Korisnik je unio nepostojeće korisničko ime ili e-mail adresu
  1. Pojavljuje se poruka o neuspješnoj prijavi i zahtjev za ponovnim unosom podataka
  2. Korisnik unosi ispravne podatke ili odustaje od prijave

  2.b Korisnik je unio neispravnu lozinku za odgovarajuće korisničko ime odnosno e-mail
  1. Pojavljuje se poruka o neuspješnoj prijavi i zahtjev za ponovnim unosom podataka
  2. Korisnik unosi ispravne podatke ili odustaje od prijave

UC5 - Komuniciranje oko potrage za nestalim ljubimcem

- **Glavni sudionik:** Vlasnik kućnog ljubimca, Sklonište za životinje
- **Cilj:** Ostvariti komunikaciju s vlasnikom izgubljenog ljubimca
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen
- **Opis osnovnog tijeka:**
  1. Korisnik odabire nestalog ljubimca
  2. Prikazuje se popis komunikacije oko potrage za ljubimcem
  3. Korisnik unosi poruku koja može sadržavati tekst, sliku i geolokaciju

UC6.1 - Postavljanje oglasa nestalog ljubimca

- **Glavni sudionik:** Vlasnik kućnog ljubimca, Sklonište za životinje
- **Cilj:** Postaviti oglas o izgubljenom ljubimcu
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen
- **Opis osnovnog tijeka:**
  1. Korisnik odabire opciju za novi oglas
  2. Korisnik unosi podatke o ljubimcu za zadane kategorije
  3. Korisnik objavljuje oglas
- **Opis mogućih odstupanja:**

  2.a Korisnik je unio invalidne podatke u određenu kategoriju
  1. Pojavljuje se poruka o pogrešci i zahtjev za ponovnim unosom podataka
  2. Korisnik unosi ispravne podatke ili odustaje od objave oglasa

  2.b Korisnik nije unio potrebne podatke
  1. Pojavljuje se poruka o pogrešci i zahtjev za ponovnim unosom podataka
  2. Korisnik unosi zahtijevane podatke ili odustaje od objave oglasa

UC6.2 - Uklanjanje oglasa nestalog ljubimca

- **Glavni sudionik:** Vlasnik kućnog ljubimca, Sklonište za životinje
- **Cilj:** Ukloniti oglas o izgubljenom ljubimcu
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen i objavio je oglas
- **Opis osnovnog tijeka:**
  1. Korisnik odabire oglas
  2. Korisnik odabire opciju za brisanje oglasa
  3. Oglas i sva komunikacija vezana uz njega nestaju iz popisa vidljivih oglasa
  4. Oglas ostaje pohranjen u bazi podataka

UC6.3 - Izmjena oglasa nestalog ljubimca

- **Glavni sudionik:** Vlasnik kućnog ljubimca, Sklonište za životinje
- **Cilj:** Izmjena oglasa o izgubljenom ljubimcu
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen i objavio je oglas
- **Opis osnovnog tijeka:**
  1. Korisnik odabire opciju za izmjenu oglasa
  2. a) Korisnik mijenja podatke po kategorijama oglasa
     b) Korisnik mijenja kategoriju oglasa
  3. Korisnik sprema promjene
- **Opis mogućih odstupanja:**

  2.a Korisnik je unio invalidne podatke u određenu kategoriju
  1. Pojavljuje se poruka o pogrešci i zahtjev za ponovnim unosom
  2. Korisnik unosi ispravne podatke ili odustaje od izmjene oglasa

  2.b Korisnik je promijenio kategoriju oglasa u kategoriju koja ne odgovara aktivnoj potrazi za ljubimcem
  1. Oglas se prebacuje u popis neaktivnih oglasa

UC7 - Odjava iz sustava

- **Glavni sudionik:** Vlasnik kućnog ljubimca, Sklonište za životinje
- **Cilj:** Odjaviti svoj korisnički profil iz aplikacije.
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen
- **Opis osnovnog tijeka:**
  1. Korisnik odabire opciju za odjavu iz sustava
  2. Korisnik se preusmjerava na sučelje neregistriranog korisnika.

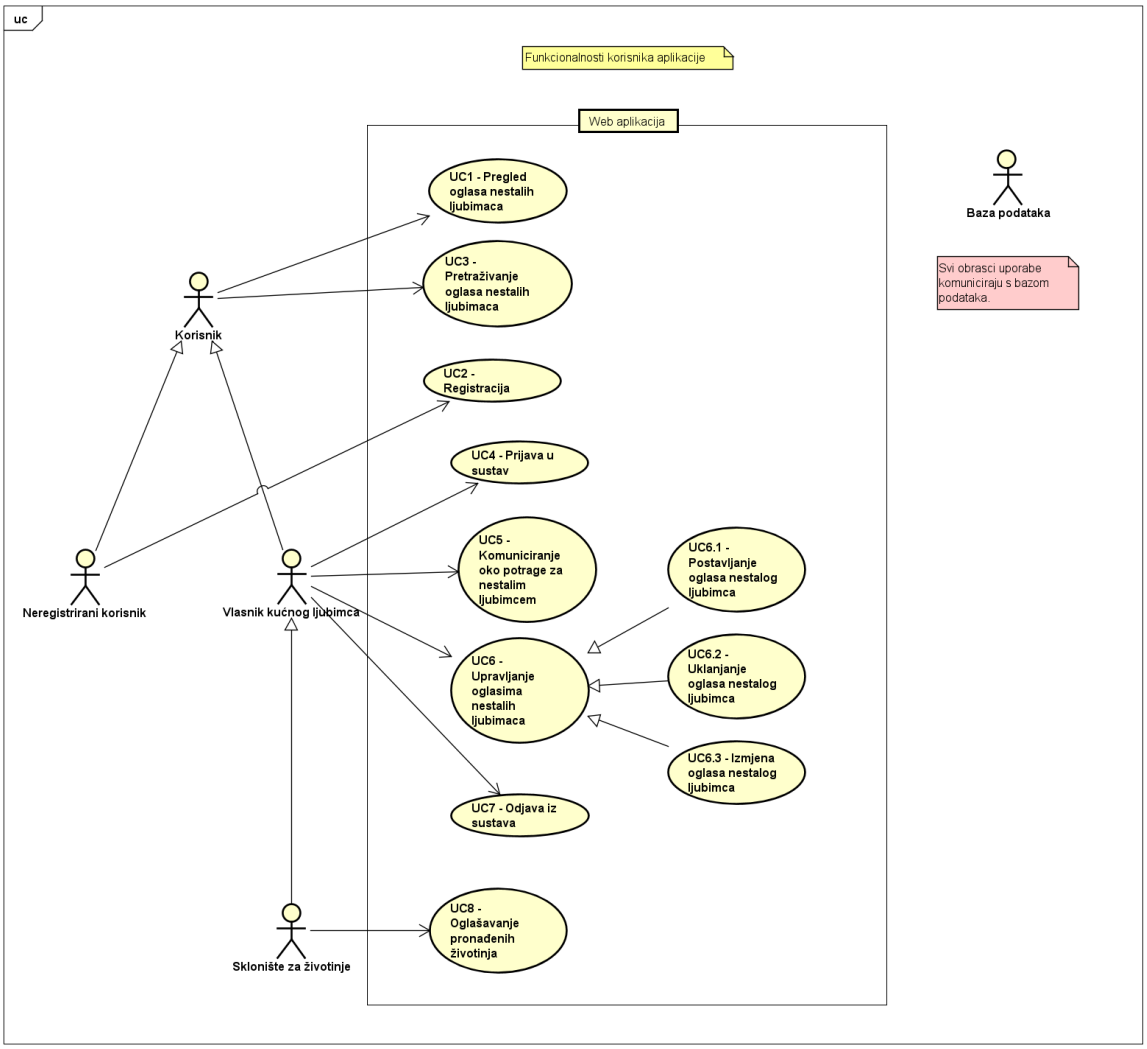
UC8 - Oglašavanje pronađenih životinja

- **Glavni sudionik:** Sklonište za životinje
- **Cilj:** Oglašavanje pronađenih životinja koje se nalaze u prostoru skloništa
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen
- **Opis osnovnog tijeka:**
  1. Korisnik odabire opciju za novi oglas
  2. Korisnik unosi podatke o ljubimcu za zadane kategorije (uključujući i dodatnu kategoriju – u skloništu)
  3. Korisnik objavljuje oglas
- **Opis mogućih odstupanja:**

  2.a Korisnik je unio invalidne podatke u određenu kategoriju
  1. Pojavljuje se poruka o pogrešci i zahtjev za ponovnim unosom podataka
  2. Korisnik unosi ispravne podatke ili odustaje od objave oglasa

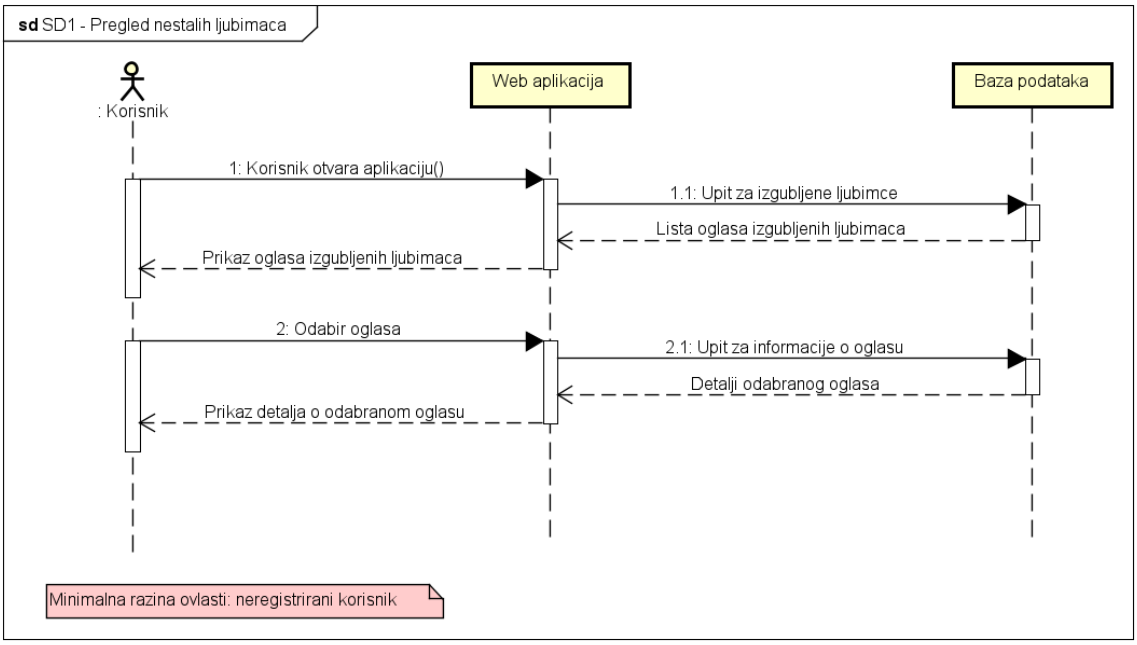
  2.b Korisnik nije unio potrebne podatke
  1. Pojavljuje se poruka o pogrešci i zahtjev za ponovnim unosom podataka
  2. Korisnik unosi zahtijevane podatke ili odustaje od objave oglasa

Dijagrami obrazaca uporabe



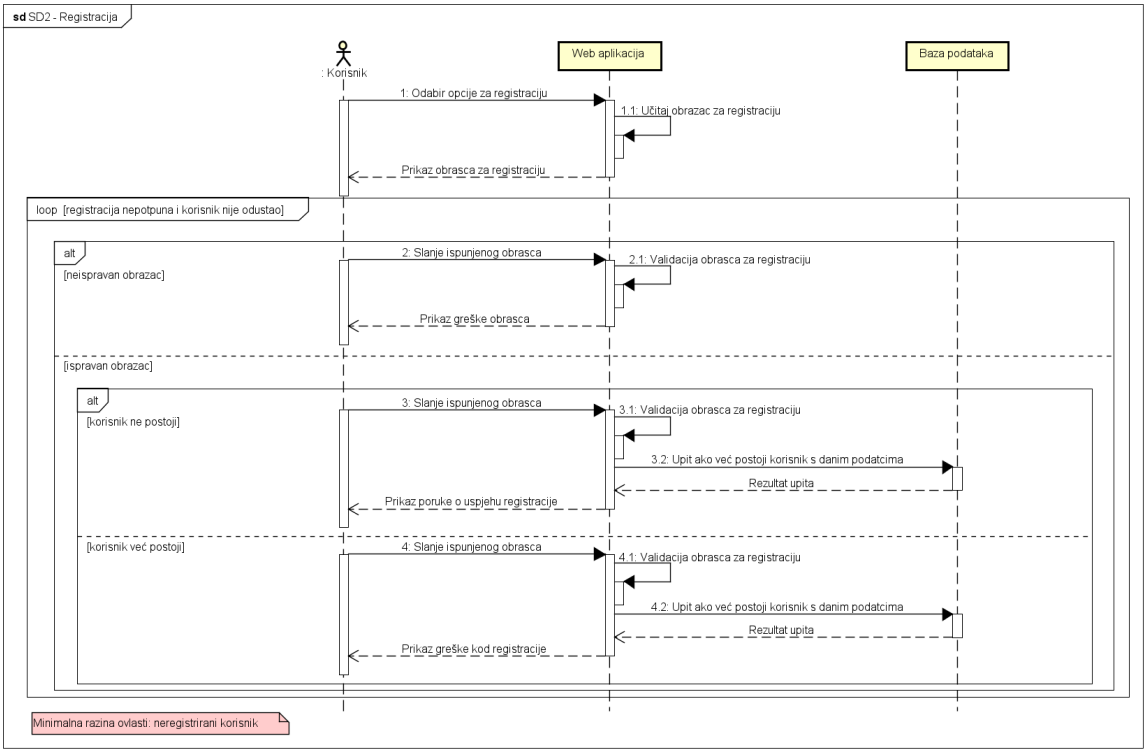
Slika 3.1: Dijagram obrasca uporabe, funkcionalnost korisnika

3.1.2 Sekvencijski dijagrami



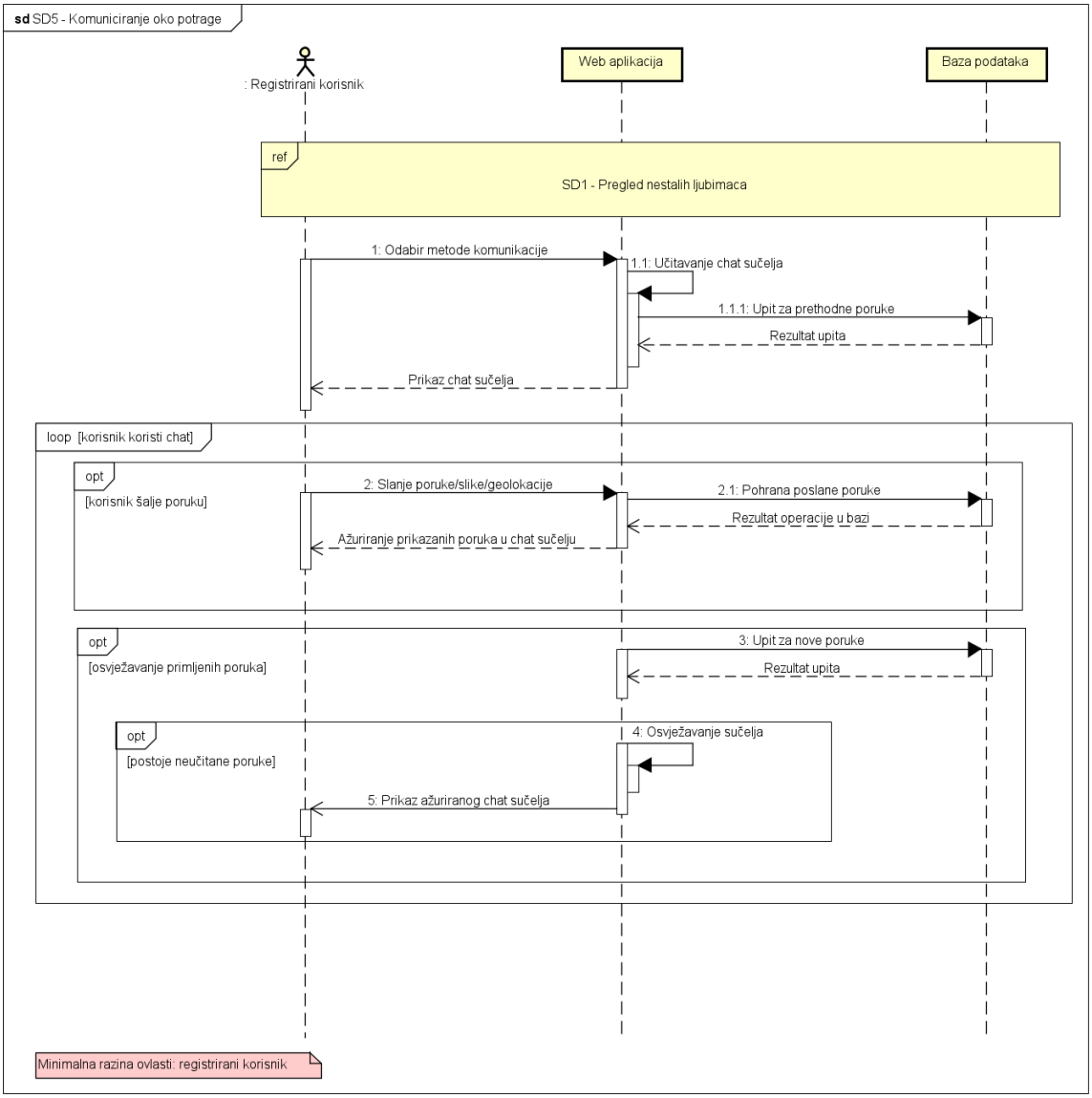
Slika 3.2: Sekvencijski dijagram - Pregled nestalih ljubimaca

Detaljnije objašnjenje: Nakon što korisnik otvori aplikaciju, aplikacija će poslati upit bazi podataka za trenutno aktivne oglase. Odgovor baze se zatim formira u obliku popisa oglasa koji se prikazuju korisniku. Nakon što korisnik odabere oglas, aplikacija šalje upit bazi podataka za detaljnim informacijama o odabranom oglasu. Baza podataka vraća detaljne informacije o odabranom oglasu. Aplikacija zatim prikazuje detaljne informacije o odabranom oglasu i moguće metode komunikacije.



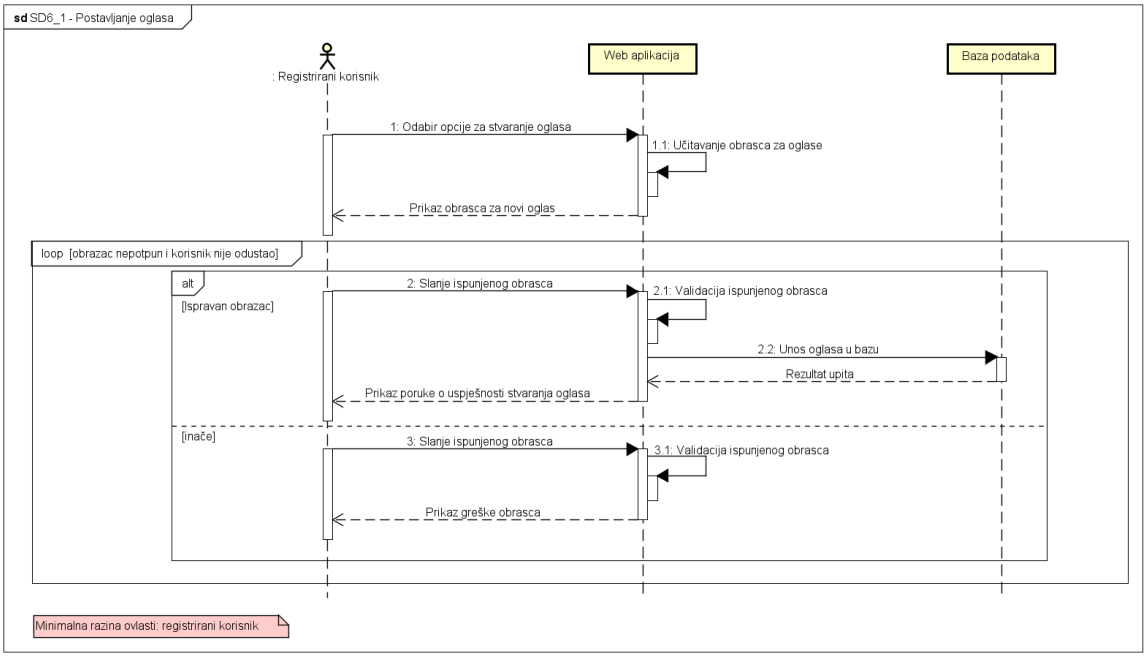
Slika 3.3: Sekvencijski dijagram - Registracija

Detaljnije objašnjenje: Nakon što korisnik odabere opciju registracije, aplikacija učitava obrazac i prikazuje ga korisniku. Korisnik zatim ispunjava i šalje ispunjen obrazac nad kojim se provodi validacija (provjera jesu li ispravno ispunjena potrebna polja), a zatim upit prema bazi podataka ako korisnik s danim podatcima već postoji. Ako korisnik s danim podatcima ne postoji, aplikacija šalje upit bazi podataka za spremanje novog korisnika. Baza podataka zatim sprema novog korisnika i vraća odgovor aplikaciji. Aplikacija zatim prikazuje poruku o uspješnoj registraciji. U suprotnom, aplikacija prikazuje odgovarajuću poruku (korisnik već postoji ili obrazac nije ispravno popunjen). Proces ispunjavanja obrasca se odvija sve dok korisnik ne odustane od registracije ili ne uspije uspješno provesti registraciju.



Slika 3.4: Sekvencijski dijagram - Komuniciranje oko potrage

Detaljnije objašnjenje: Korisnik odabere oglas i opciju komunikacije. Aplikacija učita chat sučelje i šalje upit bazi podataka za sve prethodne poruke. Baza podataka vraća prethodne poruke koje aplikacija prikazuje korisniku. Korisnik unese poruku i aplikacija je šalje bazi podataka. Baza podataka sprema poruku i vraća odgovor aplikaciji. Aplikacija prikazuje poruku korisniku. U određenim intervalima, aplikacija šalje upit bazi podataka za nove poruke. Baza podataka vraća nove poruke koje aplikacija prikazuje korisniku. Proces se nastavlja dok korisnik ne napusti chat sučelje.



Slika 3.5: Sekvencijski dijagram - Postavljanje oglasa

Detaljnije objašnjenje: Korisnik odabere opciju za novi oglas. Aplikacija učitava obrazac i prikazuje ga korisniku. Korisnik zatim ispunjava i šalje ispunjen obrazac nad kojim se provodi validacija (provjera jesu li ispravno ispunjena potrebna polja). Ako je obrazac ispravno ispunjen, aplikacija šalje upit bazi podataka za spremanje novog oglasa. Baza podataka zatim sprema novi oglas i vraća odgovor aplikaciji. Aplikacija zatim prikazuje poruku o uspješnoj objavi oglasa. U suprotnom, aplikacija prikazuje odgovarajuću poruku (obrazac nije ispravno popunjen). Proces ispunjavanja obrasca se odvija sve dok korisnik ne odustane od objave oglasa ili ne uspije uspješno objaviti oglas.

3.2 Ostali zahtjevi

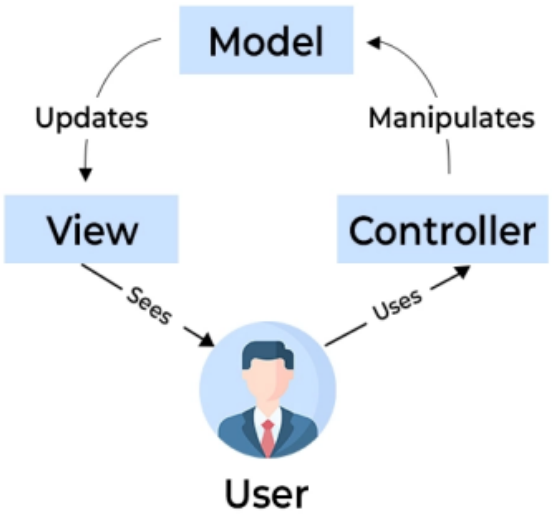
- Osnovni jezik korisničkog sučelja je hrvatski.
- Sustav treba podržati hrvatski jezik i hrvatske dijakritičke znakove - Unicode standard .
- Vrijeme odgovora na korisnički zahtjev ne smije biti duže od nekoliko sekundi, mora biti prikladno hitnosti zahtjeva.
- Sustav mora podržati rad više korisnika istovremeno.
- Sustav mora biti jednostavan i korisničko sučelje intuitivno za korištenje.
- Web aplikacija mora biti responzivna, potrebno je uzeti u obzir korisnike koji pristupaju putem mobilnih uređaja, tableta i sl.
- Komunikacija između korisnika i poslužitelja mora biti kriptirana, potrebno implementirati SSL vezu odnosno HTTPS protokol.
- Iznenadni prekid rada sustava ne smije ugroziti unesene podatke korisnika.
- Neispravno korištenje korisničkog sučelja ne smije narušiti funkcionalnost sustava.

4. Arhitektura i dizajn sustava

Arhitektura koju koristimo se zasniva na MVC (Model-View-Controller) konceptu, varijacije arhitekture zasnovane na događajima.

Cjelokupni sustav se može podijeliti na četiri glavne komponente:

- Web preglednik
- Web poslužitelj (server)
- Web aplikaciju
- Bazu podataka



Slika 4.1: MVC model

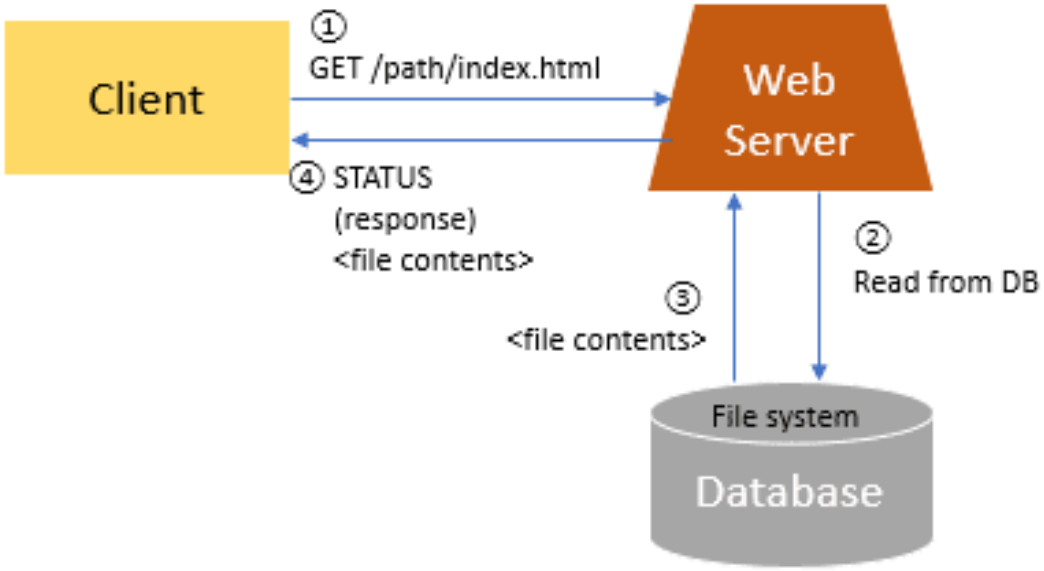
Web preglednik je program (software) koji omogućava korisnicima pregledavanje i prikazivanje web stranica na World Wide Webu (WWW). Predstavlja sučelje između korisnika i internetskog sadržaja. Pomoću njega korisnik sustava komunicira s web aplikacijom, točnije *View* i *Controller* komponentom.

Web aplikaciju pokreće **poslužitelj**. To je program čiji je osnovni zadatak odgovarati na HTTP (*Hyper Text Transfer Protocol*) zahtjeve klijenata, primati i slati određene resurse - posluživati samu aplikaciju.

U većini slučajeva klijent će zahtjevati pristup podacima kojima web poslužitelj nema pristup. U tom slučaju će *Controller* komponenta poslati zahtjev *Model* komponenti, zaslužnoj za pohranu korisničkih podataka, koju u našem slučaju predstavlja **baza podataka**.

Baza podataka organizirana je zbirka logički povezanih, pretražljivih i međusobno ovisnih podataka. Ključna je komponenta mnogih aplikacija i sustava zbog mogućnosti sigurne pohrane i brzog pretraživanja SQL upitima.

Za našu implementaciju korisničkog sučelja odabrali smo programski jezik *JavaScript* s bibliotekom *React*. Za implementaciju poslužiteljske strane odabrali smo programski jezik *Java* i *Spring* framework, točnije proširenje *Spring Boot*. Kao razvojnu okolinu odabrali smo *Visual Studio Code* i *JetBrains IntelliJ IDEA*. Za implementaciju baze podataka odabrali smo *PostgreSQL*.



Slika 4.2: Prikaz osnovnog rada sustava

4.1 Baza podataka

Za naš sustav koristit ćemo relacijsku bazu podataka, koja je strukturno pogodna za modeliranje stvarnog svijeta. Osnovna komponenta ove baze je relacija, koja se definira svojim imenom i skupom atributa. Glavna svrha ove baze podataka je omogućiti brzo i jednostavno pohranjivanje, mijenjanje i dohvaćanje podataka radi daljnje obrade. Baza podataka za ovu aplikaciju uključuje sljedeće entitete:

- Korisnik
- Registrirani
- Skloniste
- Oglas
- Ljubimac
- Lokacija
- Mjesto
- Zupanija
- Slika
- Poruka
- Kategorija

4.1.1 Opis tablica

Korisnik Entitet sadržava sve važne informacije o korisniku web-aplikacije. Atributi su sljedeći: sifKorisnik, korisnickoIme, lozinka, email, telefon, postBr. Primarni ključ ovog entiteta je atribut sifKorisnik. Entitet je povezan s entitetom Mjesto pomoću atributa postBr kroz vezu *Many-to-One*, a s entitetima Registrirani i Skloniste, kao specijalizacije, pomoću atributa korisnickoIme. Postoji i povezanost vezom *One-to-Many* s identifikacijskim slabim entitetom Poruka pomoću atributa korisnickoIme.

Korisnik		
sifKorisnik	VARCHAR	jedinstvena šifra korisnika
korisnickoIme	VARCHAR	korisničko ime
lozinka	VARCHAR	hash lozinke

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

Korisnik		
email	VARCHAR	e-mail adresa korisnika
telefon	VARCHAR	telefonski broj korisnika
postBr	INT	poštanski broj mjesta stanovanja korisnika

Registrirani Entitet je specijalizacija entiteta Korisnik i sadrži dodatne podatke o registriranom korisniku. Atributi entiteta su ime, prezime i korisnickoIme. Entitet je kao specijalizacija povezan entitetom Korisnik kroz atribut korisnickoIme.

Registrirani		
ime	VARCHAR	ime registriranog korisnika
prezime	VARCHAR	prezime registriranog korisnika
korisnickoIme	VARCHAR	jedinstveno korisničko ime

Skloniste Entitet je specijalizacija entiteta Korisnik i sadrži dodatne podatke o skloništu za životinje. Atributi entiteta su nazivSklonista i korisnickoIme. Entitet je kao specijalizacija povezan entitetom Korisnik kroz atribut korisnickoIme.

Skloniste		
nazivSklonista	VARCHAR	naziv skloništa za životinje
korisnickoIme	VARCHAR	jedinstveno korisničko ime

Ljubimac Entitet sadržava sve važne podatke o kućnome ljubimcu. Atributi entiteta su: sifLjubimac, ime, vrsta, starost, boja, opis. Primarni ključ ovog entiteta je atribut sifLjubimac. Entitet je povezan vezom *One-to-Many* kroz atribut sifLjubimac s entitetom Slika. Također, postoji veza *One-to-Many* s entitetom Oglas pomoću atributa sifOglas.

Ljubimac		
sifLjubimac	INT	jedinstvena šifra ljubimca
ime	VARCHAR	ime na koje se ljubimac odaziva

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

Ljubimac		
vrsta	VARCHAR	vrsta ljubimca
starost	INT	starost ljubimca
boja	VARCHAR	boja ljubimca
opis	VARCHAR	dodatan opis ljubimca

Oglas Entitet sadržava sve važne informacije o oglasu za kućnog ljubimca. Atributi su sljedeći: sifOglas, datumVrijemeNestanka, koordinate, sifLjubimac, sifKategorija, korisnickoIme. Primarni ključ ovog entiteta je atribut sifOglas. Entitet je u vezi *Many-to-One* s entitetom Lokacija pomoću atributa koordinate, u vezi *Many-to-One* s entitetom Ljubimac pomoću atributa sifLjubimac, u vezi *Many-to-One* s entitetom Kategorija pomoću atributa sifKategorija te u vezi *Many-to-One* s entitetom Korisnik pomoću atributa korisnickoIme.

Oglas		
sifOglas	INT	jedinstveni identifikator oglas
datumVrijemeNestanka	TIMESTAMP	datum i vrijeme nestanka ljubimca
koordinate	VARCHAR	koordinate lokacije nestanka ljubimca
sifLjubimac	INT	jedinstveni identifikator ljubimca
sifKategorija	INT	jedinstveni identifikator kategorije oglasa
korisnickoIme	VARCHAR	jedinstveno korisničko ime korisnika koji je postavio oglas

Lokacija Entitet sadržava podatke o lokaciji. Atributi entiteta su sifLokacija, geografskaSirina, geografskaDuzina, imeLokacije i postBr, gdje je primarni ključ atribut sifLokacija. Postoji veza *Many-to-One* s entitetom Mjesto kroz atribut postBr.

Lokacija		
sifLokacija	VARCHAR	jedinstvena šifra lokacija
geografskaSirina	VARCHAR	geografska širina
geografskaDuzina	VARCHAR	geografska dužina
imeLokacija	VARCHAR	ime lokacije
koordinate	VARCHAR	jedinstvene koordinate lokacije
postBr	INT	poštanski broj mjesta

Mjesto Entitet je sastavljen od podataka vezanih za mjesto pomoću atributa postBr, koji je primarni ključ, nazivMjesto i sifZup. Postoje veze *One-to-Many* pomoću atributa postBr s entitetima Lokacija i Korisnik te veza *Many-to-One* s entitetom Zupanija pomoću atributa sifZup.

Mjesto		
postBr	INT	poštanski broj mjesta
nazivMjesto	VARCHAR	naziv mjesta
sifZup	INT	jedinstveni identifikator županije

Zupanija Entitet je sastavljen od podataka vezanih za županiju pomoću atributa sifZup i nazivZup. Primarni ključ je sifZup. Postoji veza *One-to-Many* pomoću atributa sifZup s entitetom Mjesto.

Zupanija		
sifZup	INT	jedinstveni identifikator županije
nazivZup	VARCHAR	naziv županije

Poruka Entitet sadržava sve važne informacije o poruci te je oblikovan kao identifikacijski slabi entitet. Atributi su sljedeći: tekstPoruke, datumVrijemeSlanja, korisnickoIme, koordinate, vezaNaSliku i sifOglas. Entitet je povezan s entitetom Korisnik pomoću atributa korisnickoIme kroz vezu *Many-to-One*, s entitetom Lokacija kroz vezu *Many-to-One* pomoću atributa koordinate te s entitetom Oglas kroz vezu *Many-to-One* pomoću atributa sifOglas.

Poruka		
tekstPoruke	VARCHAR	tekst poruke
datumVrijemeSlanja	TIMESTAMP	datum i vrijeme slanja poruke
korisnickoIme	VARCHAR	jedinstveno korisnicko ime
koordinate	VARCHAR	jedinstvene koordinate lokacije
vezaNaSliku	VARCHAR	jedinstvena poveznica do slike
sifOglas	INT	jedinstveni identifikator oglasa

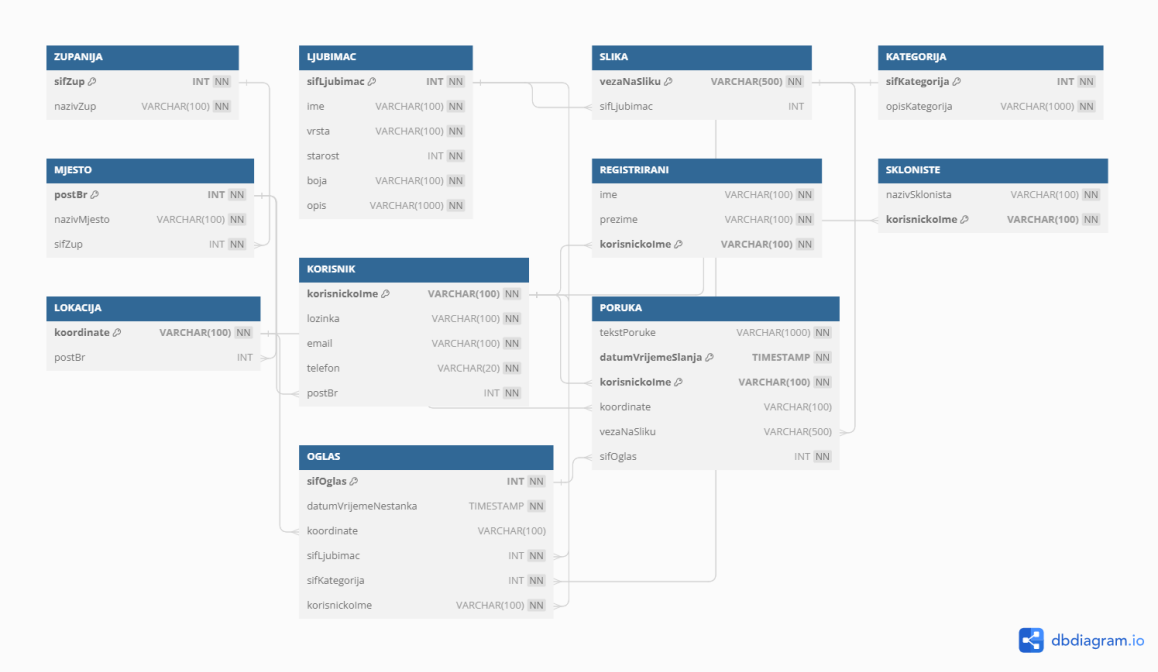
Slika Entitet sadržava podatke o slici. Atributi su vezaNaSliku, koji je primarni ključ, i sifLjubimac. S entitetom su povezani: Ljubimac s vezom *Many-to-One* pomoću atributa sifLjubimac, Poruka s vezom *One-to-Many* uz pomoć atributa vezaNaSliku.

Slika		
vezaNaSliku	VARCHAR	jedinstvena poveznica do slike
sifLjubimac	INT	jedinstveni identifikator ljubimca

Kategorija Entitet se sastoji od podataka o kategoriji: sifKategorija (primarni ključ) i opisKategorija. Postoji veza *One-to-Many* između entiteta Kategorija i Oglasa kroz atribut sifKategorija.

Kategorija		
sifKategorija	INT	jedinstveni identifikator kategorije
opisKategorija	VARCHAR	opis kategorije

4.1.2 Dijagram baze podataka

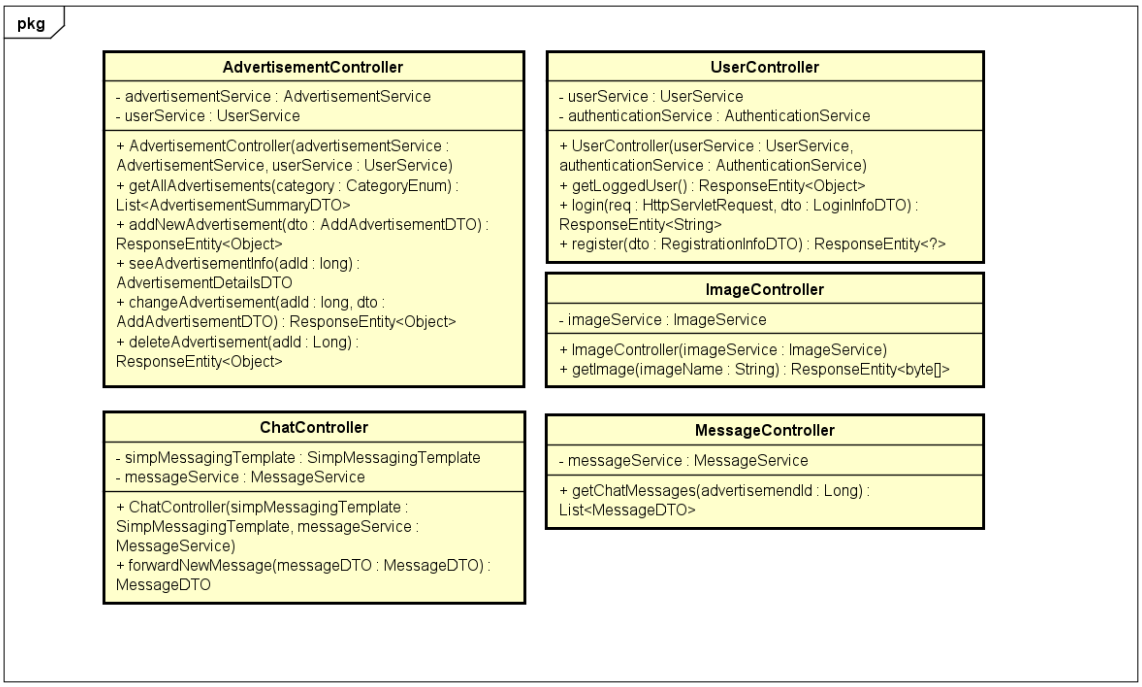


Slika 4.3: E-R dijagram baze podataka

4.2 Dijagram razreda

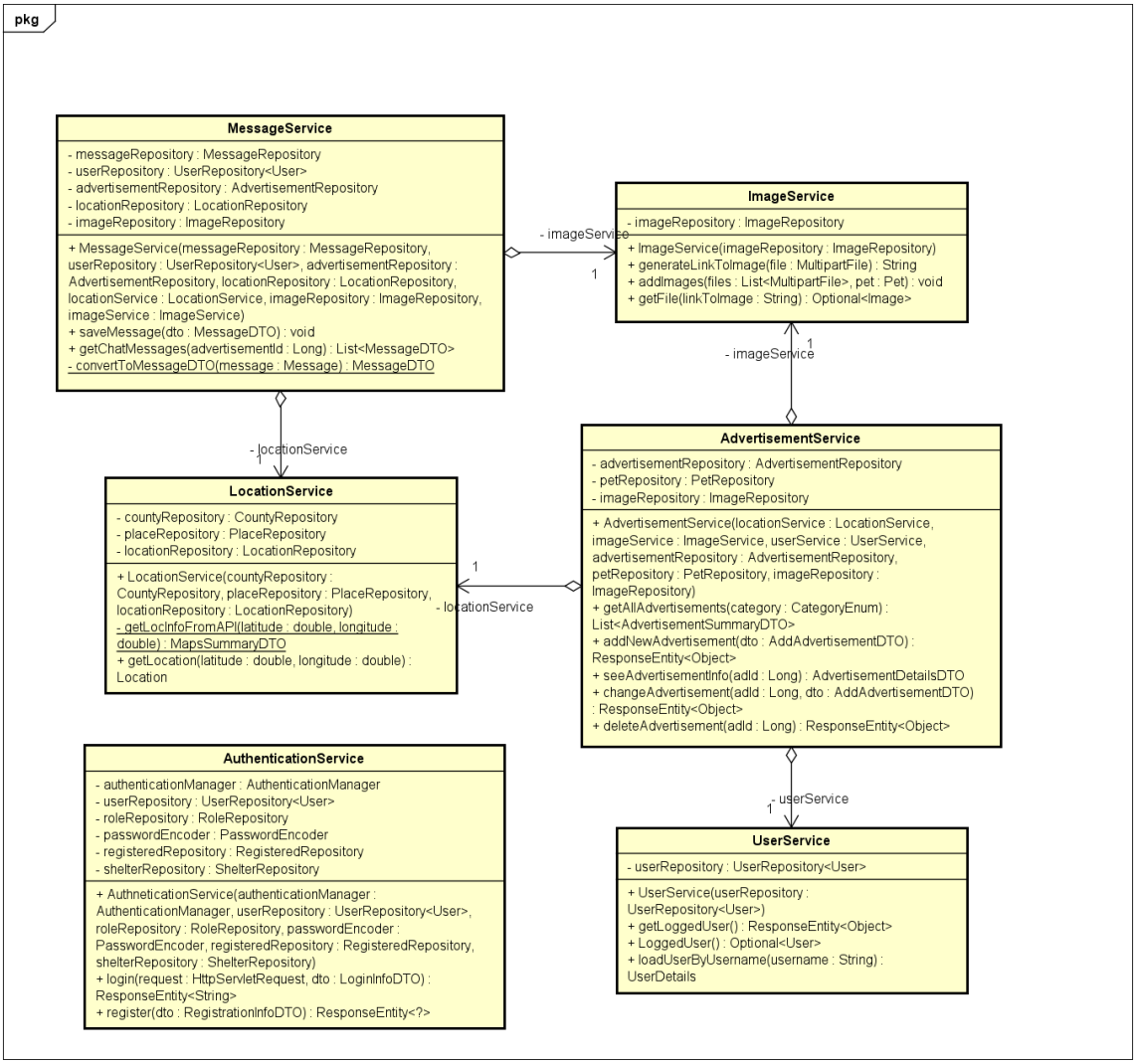
Na slikama 4.4, 4.5, 4.6 i 4.7 su prikazani razredi koji pripadaju *backend* dijelu MVC arhitekture. Na slici 4.4 je vidljiv paket *controller* koji dijelimo na razrede. Metode koje su implementirane u tim razredima koriste DTO-ove*(Data transfer object)*, a njima pristupamo putem metoda koje su implementirane u *entity* razredima. Metode unutar *controller* paketa vraćaju JSON objekte s odgovarajućim HTTP status kodovima.

Razredi su strukturirani prema pravima pristupa metodama različitih aktora kako bi se rasteretio dijagram od prevelike složenosti. Prikazane su samo ovisnosti između razreda unutar istog segmenta dijagrama. Na osnovu naziva i tipova atributa unutar razreda, može se zaključiti vrsta veza među različitim entitetima.



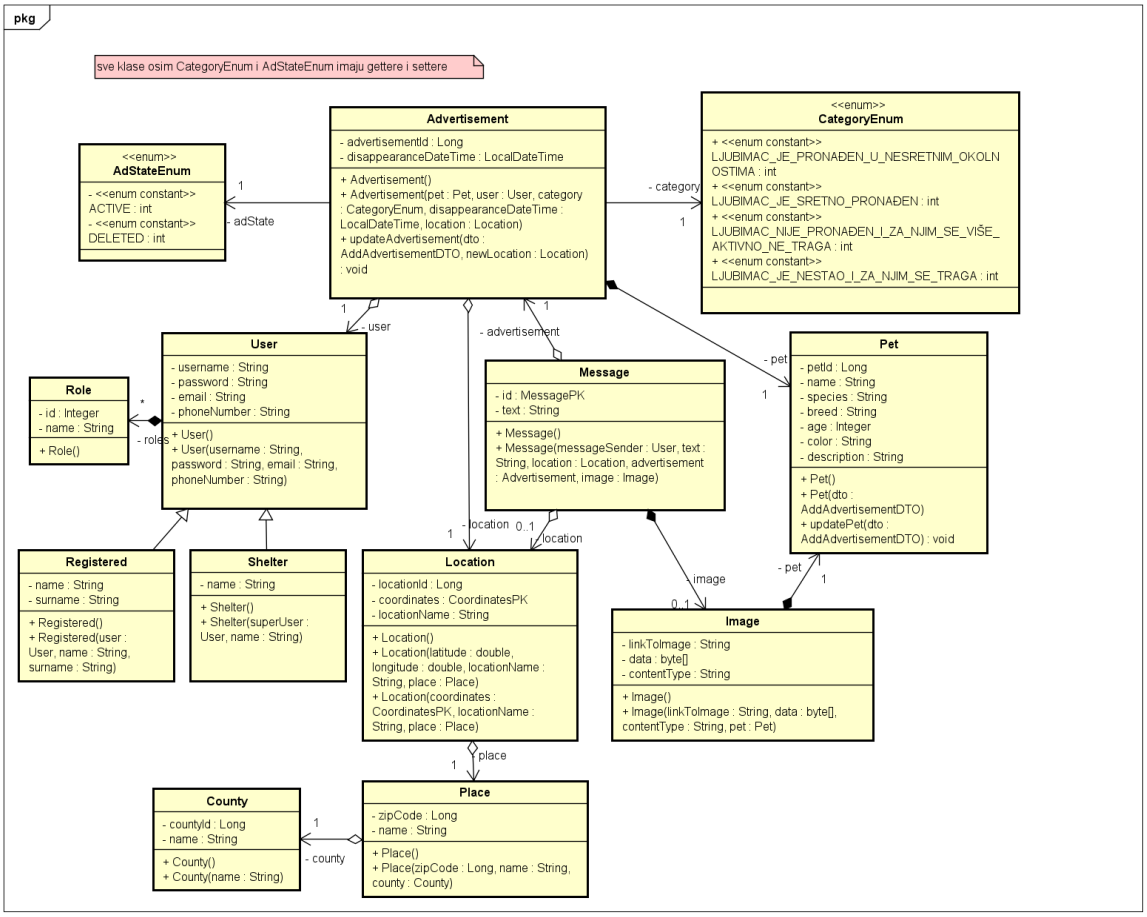
Slika 4.4: Dijagram razreda - paket ”controller”

Servisni sloj pruža poslovnu logiku i pristup podacima za različite funkcionalnosti aplikacije. Razredi paketa *services* koriste repozitorije za pristup podacima. Rezultati ovih servisa se koriste u razredima koji se nalaze u paketu *controller* kako bi se generirali odgovori na zahtjeve korisnika.

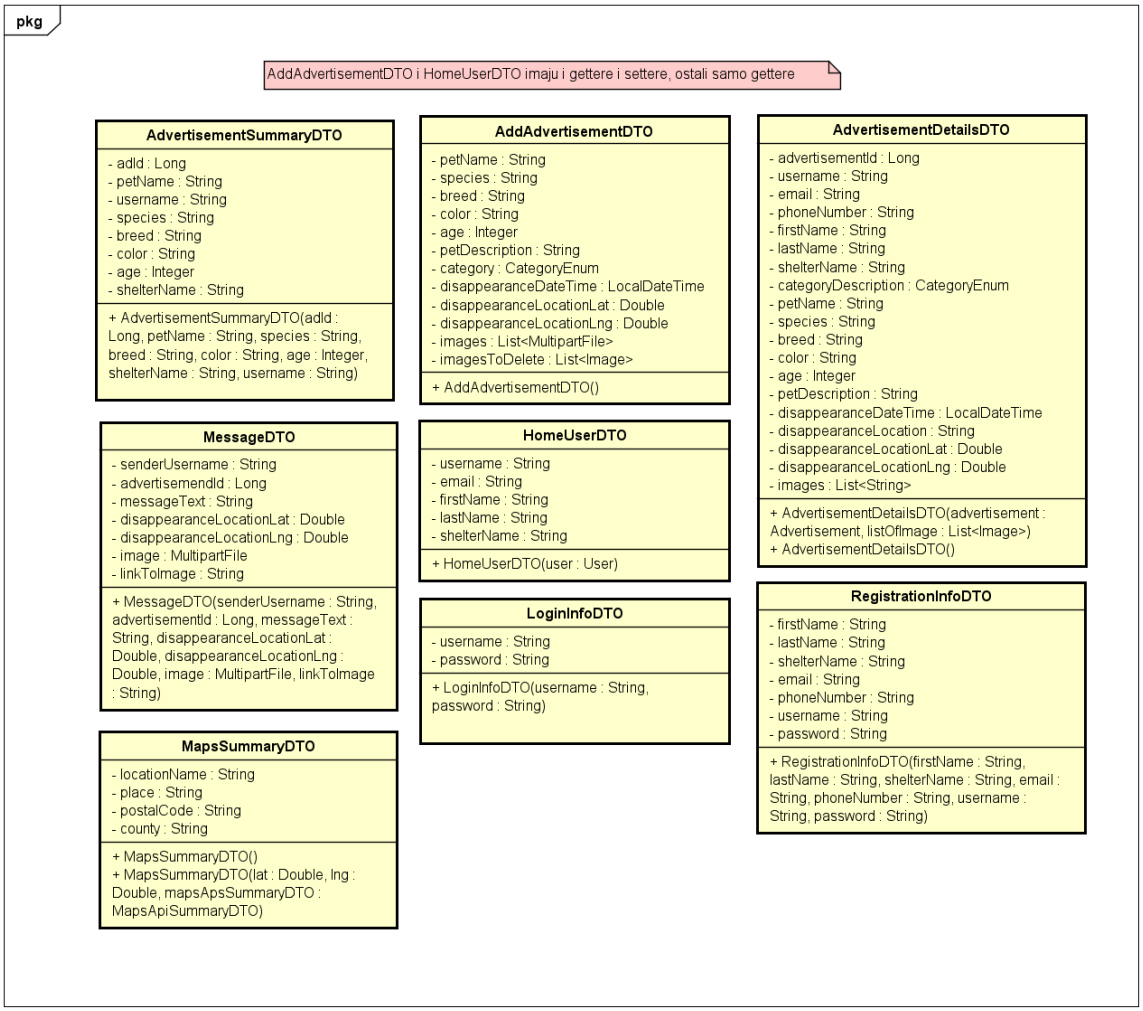


Slika 4.5: Dijagram razreda - paket ”services”

Entity razredi odražavaju strukturu baze podataka u aplikaciji. Metode unutar ovih razreda komuniciraju izravno s bazom podataka kako bi dohvatili željene podatke. Razred *User* predstavlja neregistriranog korisnika koji se može registrirati popunjavajući obrazac svojim osnovnim informacijama. Registrirane korisnike dijelimo na razred *Registered* koji predstavlja korisnika koji je registriran i može koristiti osnovne funkcionalnosti sustava te na razred *Shelter* koji predstavlja sklonište za životinje. Sklonište ima mogućnost oglašavanja pronađenih životinja.



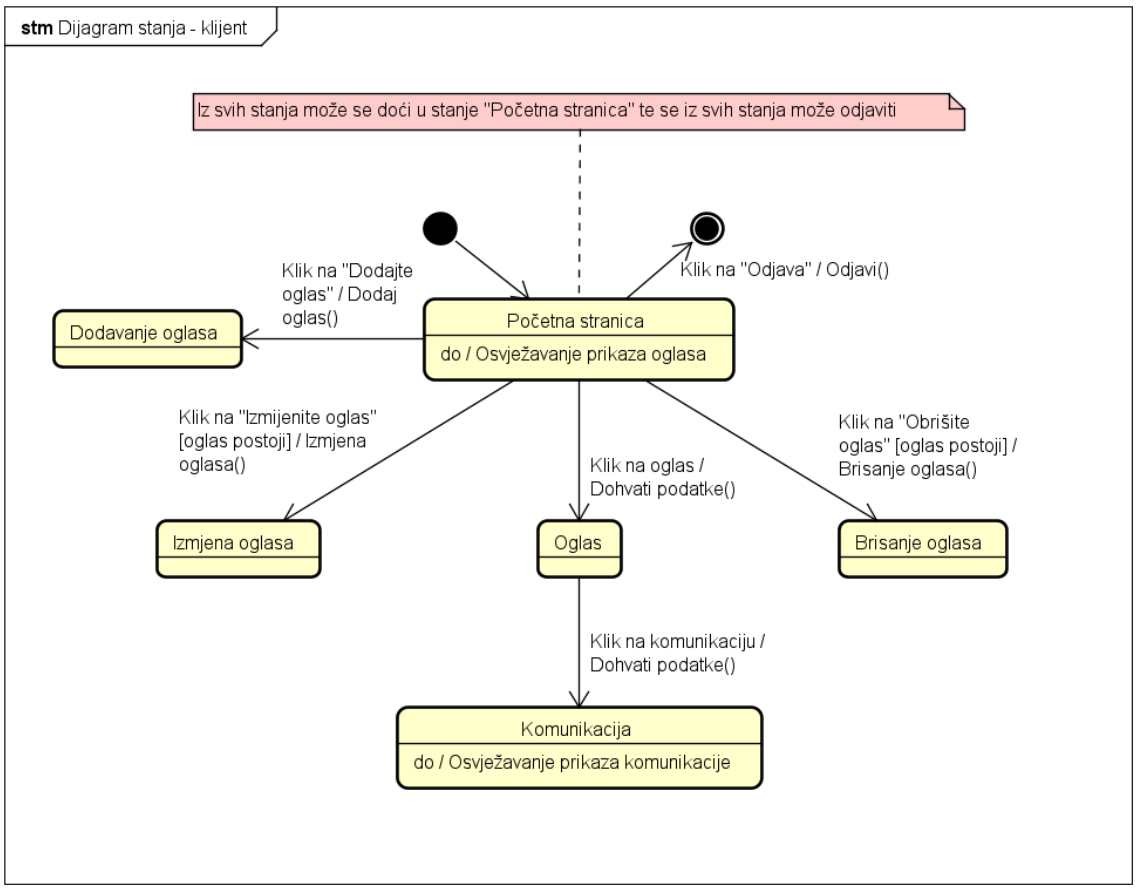
Slika 4.6: Dijagram razreda - paket ”entity”



Slika 4.7: Dijagram razreda - paket ”dto”

4.3 Dijagram stanja

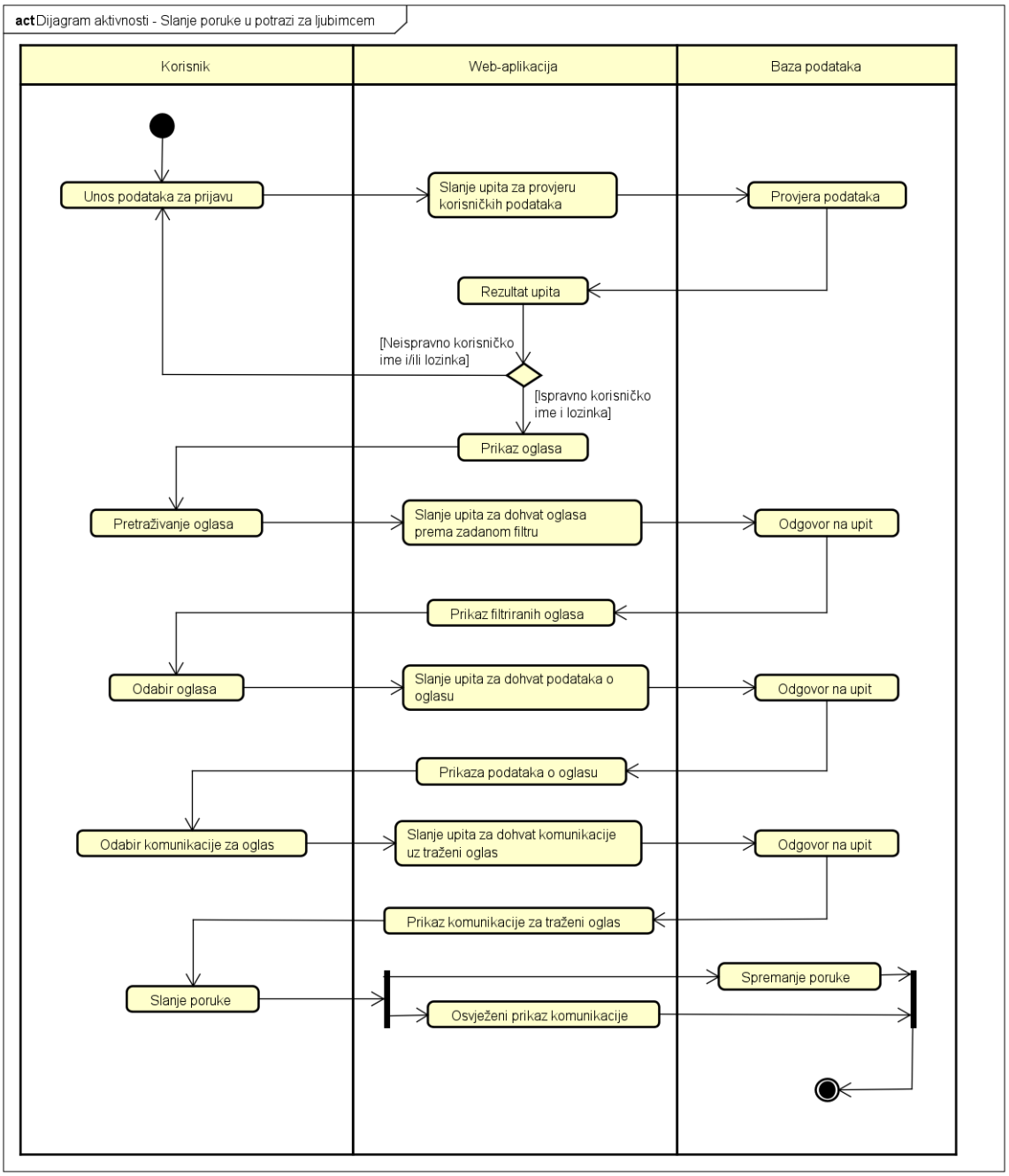
Dijagram stanja prikazuje stanja objekta te prijelaze iz jednog stanja u drugo temeljene na događajima. Na slici 4.8 prikazan je dijagram stanja za registriranog korisnika. Nakon prijave, klijentu se prikazuje početna stranica na kojoj može pregledavati oglase. Za odabrani oglas, registrirani korisnik nadalje može pregledavati i sudjelovati u komunikaciji vezanoj uz potrazi za ljubimcem iz oglasa. S početne stranice klijent također može doći do opcija dodavanja oglasa te, u slučaju da je klijent objavio oglas, izmjene i brisanja oglasa.



Slika 4.8: Dijagram stanja - klijent

4.4 Dijagram aktivnosti

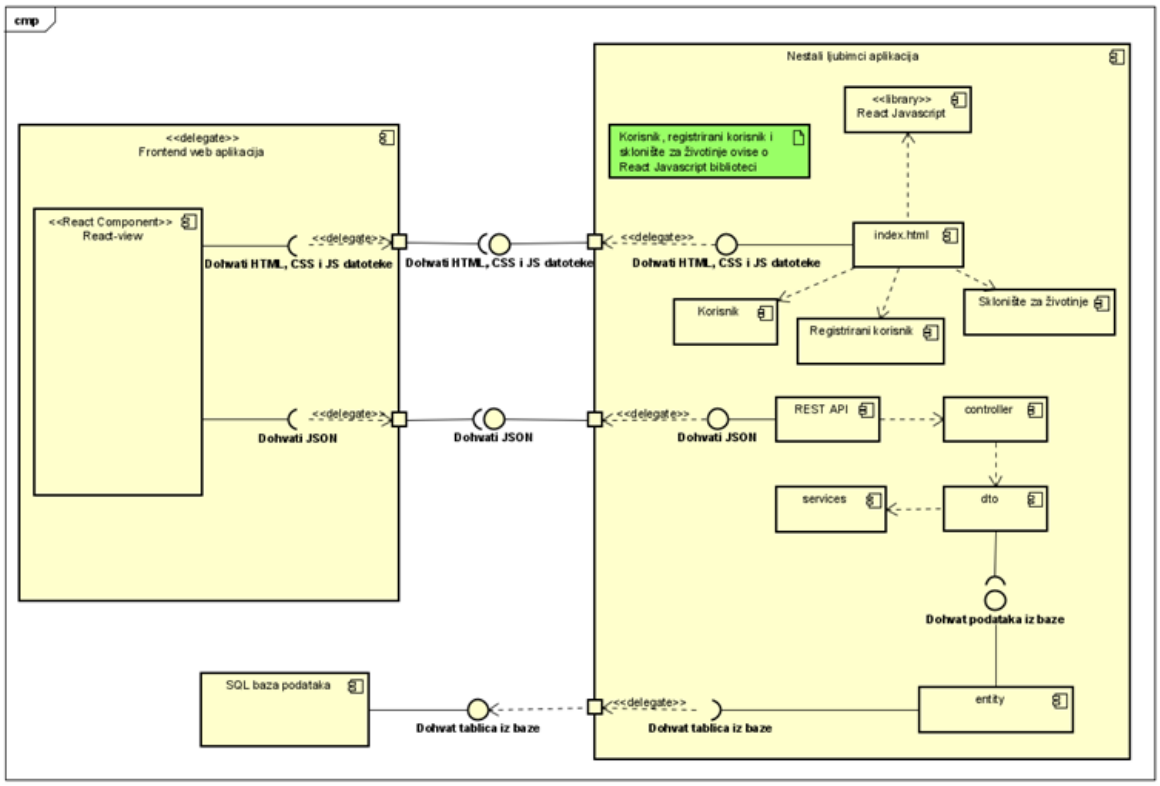
Dijagram aktivnosti prikazuje izvođenje aktivnosti kroz niz akcija koje čine upravljačke tokove i tokove objekata. Na slici 4.9 prikazan je dijagram aktivnosti za slanje jedne poruke u potrazi za ljubimcem. Korisnik se najprije prijavljuje u web-aplikaciju, a zatim pretražuje oglase prema kategoriji po odabiru. Nakon pronalaska oglasa, registrirani korisnik može odabrati opciju za prikaz komunikacije vezane uz pronalazak ljubimca iz oglasa. Konačno, korisnik može poslati poruku koja se sprema u bazu podataka.



Slika 4.9: Dijagram aktivnosti - Slanje poruke u potrazi za ljubimcem

4.5 Dijagram komponenti

Dijagram komponenti, kao što je prikazano na slici 4.10, opisuje organizaciju, međuovisnost komponenti, te interne strukture i odnose prema okolini u sustavu. Sustav ima dva različita sučelja: sučelje za dohvat HTML, CSS i JS datoteka koje poslužuje *frontend* dio aplikacije te sučelje za dohvat JSON podataka koje pristupa REST API komponenti. *Frontend* dio sastoji se od niza JavaScript datoteka koje su grupirane u logičke cjeline prema tipovima aktora koji pristupaju aplikaciji. Sve JavaScript datoteke ovise o React biblioteci iz koje dohvaćaju gotove komponente kao što su gumbi, forme i slično. REST API poslužuje podatke koji pripadaju *backend* dijelu aplikacije. Podaci se dohvaćaju putem REST API sučelja, a *entity* je zadužen za dohvaćanje tablica iz baze podataka pomoću SQL upita. *Backend* koristi MVC arhitekturu za slanje podataka u obliku dto *(Data Transfer Object)* prema Reactview komponenti. React-view komponenta komunicira sa *Nestali ljubimci* aplikacijom preko dostupnih sučelja. Ovisno o korisnikovim akcijama, osvježava prikaz, dohvaća nove podatke ili datoteke te ih prikazuje korisniku.



Slika 4.10: Dijagram komponenti

5. Implementacija i korisničko sučelje

5.1 Korištene tehnologije i alati

Komunikacija u timu realizirana je korištenjem aplikacije Discord[1]. Za izradu UML dijagrama korišten je alat Astah Professional[2], a kao sustav za upravljanje izvornim kodom Git[3]. Udaljeni repozitorij projekta je dostupan na web platformi GitHub[4].

Kao razvojno okruženje korišten je IntelliJ IDEA[5] - integrirano je razvojno okruženje (IDE) tvrtke JetBrains. Prvenstveno se koristi za razvoj računalnih programa za operacijski sustav Windows, kao i za web-stranice, web-aplikacije, web-usluge i mobilne aplikacije. IntelliJ IDEA za razvoj softvera koristi JetBrains platforme i tehnologije za olakšavanje procesa razvoja. To uključuje podršku za razvoj u raznim programskim jezicima(poput Java, Kotlin, Groovy i mnogih drugih), za Android razvoj, Spring Framework, Web development(*frontend* i *backend*) itd. Uz IntelliJ IDEA ponešto je korišten i Visual Studio Code[6], ali uglavnom kao text editor te za pisanje docker[7] deploy skripte. Docker predstavlja platformu kao uslugu(Paas) koja koristi virtualizaciju na razini OS-a za isporuku programa u spremnicima. Spremnici osiguravaju izolaciju procesa, mreže i datotečnog sustava.

Aplikacija je napisana koristeći radni okvir Spring Framework[8] i jezik Java[9] za izradu *backenda* te React[10] i jezik TypeScript[11] za izradu *frontenda*. React, također poznat kao React.js ili ReactJS, je biblioteka u JavaScriptu za izgradnju korisničkih sučelja. Održavana je od strane Facebooka. React se najčešće koristi kao osnova u razvoju web ili mobilnih aplikacija. Složene aplikacije u Reactu obično zahtijevaju

[1] https://discord.com/
[2] https://astah.net/products/astah-professional/
[3] https://git-scm.com/
[4] https://github.com/
[5] https://www.jetbrains.com/idea/
[6] https://code.visualstudio.com/
[7] https://www.docker.com/
[8] https://spring.io/
[9] https://www.java.com/en/
[10] https://react.dev/
[11] https://www.typescriptlang.org/

korištenje dodatnih biblioteka za interakciju s API-jem. Programski jezik TypeScript je nadogradnja na JavaScript, a napravio ga je Microsoft. Radni okvir Spring Framework nadograđuje mogućnosti samog operativnog sustava. Radi se o posebnoj infrastrukturi koja programerima nudi gotova rješenja i funkcionalnosti da bi ubrzala i pojednostavila razvoj aplikacija svih vrsta i oblika.

Baza podataka se nalazi na poslužitelju u oblaku Render[12].

[12] https://render.com/

5.2 Ispitivanje programskog rješenja

dio 2. revizije

U ovom poglavlju je potrebno opisati provedbu ispitivanja implementiranih funkcionalnosti na razini komponenti i na razini cijelog sustava s prikazom odabranih ispitnih slučajeva. Studenti trebaju ispitati temeljnu funkcionalnost i rubne uvjete.

5.2.1 Ispitivanje komponenti

Ispitivanje komponenti sustava smo proveli uz pomoć alata JUnit [13], okvira za jedinično testiranje za programski jezik Java koji je iznimno važan u razvoju vođenom testiranjem. Pri razvoju aplikacije korištena je inačica 5.

Izvorni kôd svih ispitnih slučajeva i rezultati izvođenja ispita prikazani su u nastavku

Kod 5.1: Test testGetAllRequestedAdvertisements

```
@Test
@DisplayName("All requested advertisements are fetched")
public void testGetAllRequestedAdvertisements() {

  Advertisement advertisement1 = mock(Advertisement.class);
  when(advertisement1.getAdState()).thenReturn(AdStateEnum.
      ACTIVE);
  when(advertisement1.getCategory()).thenReturn(CategoryEnum.
      LJUBIMAC_JE_SRETNO_PRONADEN);

  Advertisement advertisement2 = mock(Advertisement.class);
  when(advertisement2.getAdState()).thenReturn(AdStateEnum.
      ACTIVE);
  when(advertisement2.getCategory()).thenReturn(CategoryEnum.
      LJUBIMAC_JE_NESTAO_I_ZA_NJIM_SE_TRAGA);
  when(advertisement2.getPet()).thenReturn(mock(Pet.class));
  when(advertisement2.getUser()).thenReturn(mock(Registered.
      class));
  when(imageRepository.findAllByPetPetId(anyLong())).thenReturn(
      Arrays.asList(mock(Image.class)));
```

[13] https://junit.org/junit5/

```
    when(advertisementRepository.findAll()).thenReturn(Arrays.
        asList(
    advertisement1, advertisement2)
    );

    List<AdvertisementSummaryDTO> result = advertisementService.
        getAllAdvertisements(
    CategoryEnum.LJUBIMAC_JE_NESTAO_I_ZA_NJIM_SE_TRAGA
    );

    assertEquals(1, result.size());
  }
```

U Kodu 5.1 prikazuje se ispitni slučaj koji osigurava da metoda AdvertisementService.getAllAdvertisements(CategoryEnum category) ispravno dohvaća i vraća oglase koji odgovaraju zadanom kriteriju kategorije. Kreiraju se dva mock (simulirana) oglasa (advertisement1 i advertisement2) koristeći Mockito okvir za testiranje. Oba oglasa su postavljena na aktivno stanje (AdStateEnum.ACTIVE), ali pripadaju različitim kategorijama. S obzirom da samo 1 objekt pripada traženoj kategoriji (advertisement2), sa assertEquals(1, result.size()) provjeravamo jesmo li dobili samo 1 oglas.

Kod 5.2: Test testNoneAdsOfRequestedCategoryFound

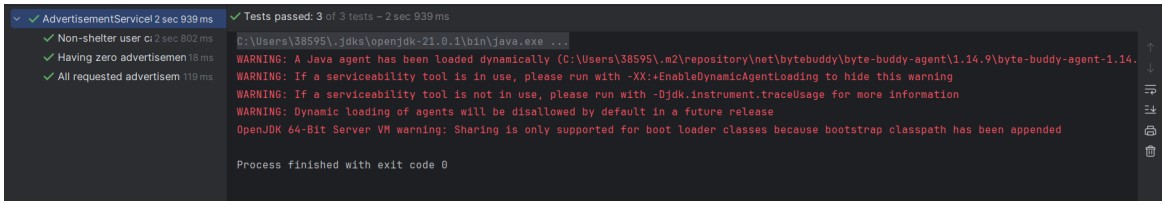
```
@Test
@DisplayName("Having zero advertisements of requested category
    is handled correctly")
public void testNoneAdsOfRequestedCategoryFound() {

  Advertisement advertisement1 = mock(Advertisement.class);
  when(advertisement1.getAdState()).thenReturn(AdStateEnum.
      DELETED);

  Advertisement advertisement2 = mock(Advertisement.class);
  when(advertisement2.getAdState()).thenReturn(AdStateEnum.
      ACTIVE);
  when(advertisement2.getCategory()).thenReturn(CategoryEnum.
      LJUBIMAC_JE_SRETNO_PRONADEN);

  when(advertisementRepository.findAll()).thenReturn(Arrays.
      asList(
  advertisement1, advertisement2)
  );

  List<AdvertisementSummaryDTO> result = advertisementService.
      getAllAdvertisements(
  CategoryEnum.LJUBIMAC_JE_PRONADEN_U_NESRETNIM_OKOLNOSTIMA
  );

  assertEquals(0, result.size());
}
```

U Kodu 5.2 prikazuje se ispitni slučaj koji, kao i u prethodnom ispitnom slučaju, ispituje funkcionalnost metode AdvertisementService.getAllAdvertisements(CategoryEnum category), ali ovaj put ispitujemo rubni slučaj: u bazi ne postoji oglas koji ispunjava korisnički zadane kriterije (ili ne pripada traženoj kategoriji ili je "obrisan"). Sa assertEquals(0, result.size()) provjeravamo je li povratna vrijednost metode ispitivane metode prazna lista.

Kod 5.3: Test testRegisteredUserCannotSetInShelterCategory

```
@Test
@DisplayName("Non-shelter user cannot set U_SKLONISTU category")
public void testRegisteredUserCannotSetInShelterCategory() {

  assertThrows(SecurityException.class, () -> {

    Long adId = 1L;
    AddAdvertisementDTO dto = mock(AddAdvertisementDTO.class);
    when(dto.getCategory()).thenReturn(CategoryEnum.U_SKLONISTU)
        ;
    when(dto.getDisappearanceLocationLat()).thenReturn(null);

    when(advertisementRepository.existsByAdvertisementId(anyLong
        ())).thenReturn(true);

    Advertisement changedAdvertisement = mock(Advertisement.
        class);
    when(changedAdvertisement.getUser()).thenReturn(mock(
        Registered.class));

    when(advertisementRepository.findByAdvertisementId(anyLong()
        )).thenReturn(Optional.of(changedAdvertisement));

    advertisementService.changeAdvertisement(adId, dto);
  });
}
```

U Kodu 5.3 nalazi se ispitni slučaj koji provjerava ispravnost rada metode AdvertisementService.changeAdvertisement(long adId, AddAdvertisementDTO dto). Testira se slučaj kada obični (registrirani) korisnik pokušava promijeniti svoj oglas tako da mu postavi kategoriju "U SKLONISTU". S obzirom da on nema dopuštenje za postavljanje te kategorije (nije sklonište), metoda changeAdvertisement() bi trebala baciti SecurityException iznimku, što i provjeravamo sa kodnim isječkom assertThrows(SecurityException.class, ...).



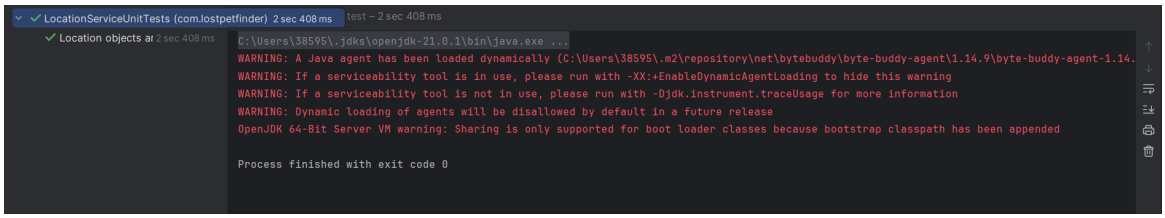
Slika 5.1: Testovi prolaze

Kod 5.4: Test testGetLocation

```
@BeforeEach
public void setup() {
  locationService = new LocationService(countyRepository,
      placeRepository, locationRepository) {
    @Override
    protected MapsSummaryDTO getLocInfoFromAPI(double latitude,
        double longitude) {
      MapsSummaryDTO dto = new MapsSummaryDTO();
      dto.setLocationName("Zagreb");
      dto.setPlace("Zagreb");
      dto.setCounty("Grad Zagreb");
      dto.setPostalCode("10000");
      return dto;
    }
  };
}

@Test
@DisplayName("Location objects are retrieved successfully")
public void testGetLocation() {
  when(countyRepository.existsByName(anyString())).thenReturn(
      true);
  when(countyRepository.findByName(anyString())).thenReturn(
      Optional.of(new County("Grad Zagreb")));
  when(placeRepository.existsByZipCode(anyLong())).thenReturn(
      true);
  when(placeRepository.findByZipCode(anyLong())).thenReturn(
      Optional.of(
  new Place(10000L, "Zagreb", new County("Grad Zagreb"))
```

```
    ));

    locationService.getLocation(45.815399, 15.966568);

    Mockito.verify(locationRepository, times(1)).save(any(Location
        .class));

  }
```

U Kodu 5.3 osigurava da metoda LocationService.getLocation(double latitude, double longitude) ispravno dohvaća informacije o lokaciji s API-ja, pravilno interagira s repozitorijima i sprema dobivene informacije o lokaciji. Test provjerava je li metoda save() pozvana nad objektom locationRepository točno jednom: ako je, to znači da je implementacija metode ispravna prema ovom scenariju, a ako nije, to ukazuje na potencijalni problem u implementaciji metode. NAPOMENA: s obzirom da povezivanje na API pri izvođenju unit testa nije preporučeno, metoda getLocInfoFromAPI(double latitude, double longitude) je konfigurirana da uvijek vraća informacije o lokaciji Zagreb, bez obzira na ulazne parametre.



Slika 5.2: Test prolazi

Kod 5.5: Test testSaveMessageWithTextOnly

```
@Test
@DisplayName("Messages with text only are successfully saved")
public void testSaveMessageWithTextOnly() {

    MessageInputDTO dto = mock(MessageInputDTO.class);
    when(dto.getSenderUsername()).thenReturn("testUser");
    when(dto.getMessageText()).thenReturn("testMessageText");
    when(dto.getAdvertisementId()).thenReturn(1L);

    when(userRepository.findByUsername(anyString())).thenReturn(
        Optional.of(mock(User.class)));
    when(advertisementRepository.findByAdvertisementId(anyLong()))
        .thenReturn(
    Optional.of(mock(Advertisement.class))
    );

    messageService.saveMessage(dto);

    Mockito.verify(messageRepository, times(1)).save(any(Message.
        class));
}
```

U Kodu 5.5 nalazi se ispitni slučaj koji provjerava funkcionalnost metode MessageService.saveMessage(MessageInputDTO dto) kada je u sklopu poruke poslan tekst, ali ne i lokacija. Kreira se simulirani objekt MessageInputDTO pomoću metode mock() koji se zatim konfigurira da vrati "testUser" kao korisničko ime pošiljatelja, "testMessageText" kao tekst poruke i 1L kao ID oglasa. Metode findByUsername() u userRepository i findByAdvertisementId() u advertisementRepository su konfigurirane da uvijek vraćaju Optional objekt User i Advertisement, bez obzira na ulazne parametre. Metoda saveMessage se zatim poziva s prethodno kreiranim MessageInputDTO objektom. Na kraju, test provjerava je li metoda save() u messageRepository pozvana točno jednom.

Kod 5.6: Test testSaveMessageWithLocationProvided

```
@Test
@DisplayName("Messages with location included are successfully
    saved")
public void testSaveMessageWithLocationProvided() {

  MessageInputDTO dto = mock(MessageInputDTO.class);
  when(dto.getSenderUsername()).thenReturn("testUser");
  when(dto.getMessageText()).thenReturn("testMessageText");
  when(dto.getAdvertisementId()).thenReturn(1L);
  when(dto.getDisappearanceLocationLat()).thenReturn(45.815399);
  when(dto.getDisappearanceLocationLng()).thenReturn(15.966568);

  when(userRepository.findByUsername(anyString())).thenReturn(
      Optional.of(mock(User.class)));
  when(advertisementRepository.findByAdvertisementId(anyLong()))
      .thenReturn(
  Optional.of(mock(Advertisement.class))
  );
  when(locationRepository.existsByCoordinates(any(CoordinatesPK.
      class))).thenReturn(false);
  when(locationService.getLocation(anyDouble(), anyDouble())).
      thenReturn(mock(Location.class));

  messageService.saveMessage(dto);

  Mockito.verify(locationRepository, times(1)).save(any(Location
      .class));
  Mockito.verify(messageRepository, times(1)).save(any(Message.
      class));
}
```

U Kodu 5.6 prikazuje se ispitni slučaj koji, kao i u prethodnom ispitnom slučaju, ispituje funkcionalnost metode MessageService.saveMessage(MessageInputDTO dto), ali u slučaju kada je lokacija zadana. Dio koda Mockito.verify(locationRepository, times(1)).save(any(Location.class)) provjerava je li pozvana metoda save() nad objektom locationRepository kako bi se utvrdilo da je zadana lokacija spremljena.

Kod 5.7: Test testIfRetrievedChatMessagesAreOrdered

```
@Test
@DisplayName("Messages are retrieved in correct order")
public void testIfRetrievedChatMessagesAreOrdered() {

    Message message1 = mock(Message.class);
    when(message1.getAltId()).thenReturn(new MessagePK(mock(User.
        class), LocalDateTime.now().minusDays(1)));
    when(message1.getAdvertisement()).thenReturn(mock(
        Advertisement.class));
    when(message1.getText()).thenReturn("Hello, World from
        testUser1!");

    Message message2 = mock(Message.class);
    when(message2.getAltId()).thenReturn(new MessagePK(mock(User.
        class), LocalDateTime.now()));
    when(message2.getAdvertisement()).thenReturn(mock(
        Advertisement.class));
    when(message2.getText()).thenReturn("Hello, World from
        testUser2!");

    when(messageRepository.findAllByAdvertisementAdvertisementId(
        anyLong())).thenReturn(
    Arrays.asList(message1, message2)
    );

    List<MessageDTO> result = messageService.getChatMessages(1L);

    assertEquals(2, result.size());
    assertEquals("Hello, World from testUser2!", result.get(0).
        getMessageText());
}
```

U Kodu 5.7 testira se vraća li metoda MessageService.getChatMessages(Long advertisementId) poruke za zahtijevani oglas u odgovarajućem poretku (prvo novije, zatim starije). Kreiraju se dva mock objekta: message1 (predstavlja stariju poruku) i message2 (predstavlja noviju poruku). Ako metoda getChatMessages() vrati 2 objekta tipa MessageDTO i ako je tekst prve poruke u vraćenoj listi "Hello,

World from testUser2!", možemo zaključiti da metoda ispravno funkcionira.

Slika 5.3: Testovi prolaze

5.2.2 Ispitivanje sustava

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

5.2.3 Ispitivanje sustava

Ispitivanje sustava proveli smo pomoću dodatka za preglednik Selenium IDE [14] te koristeći Selenium WebDriver [15] unutar JUnit testova.

Ispitivanje pomoću Selenium WebDrivera prikazana su na slikama * i *

Kod 5.8: Test testLoginGoodCreds

```
@Test
public void testLoginGoodCreds() {
  WebDriver driver = new ChromeDriver();
  System.setProperty(
  "webdriver.chrome.driver",
  "C:\\Users\\akrkl\\AppData\\Local\\Microsoft\\WindowsApps\\
     chromedriver.exe"
  );
  driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

  driver.get("http://localhost:5173/login");

  WebElement element = driver.findElement(By.id("email"));
  element.sendKeys("testadmin");

  element = driver.findElement(By.id("lozinka"));
  element.sendKeys("123");

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

[14] https://www.selenium.dev/documentation/ide/
[15] https://www.selenium.dev/documentation/webdriver/

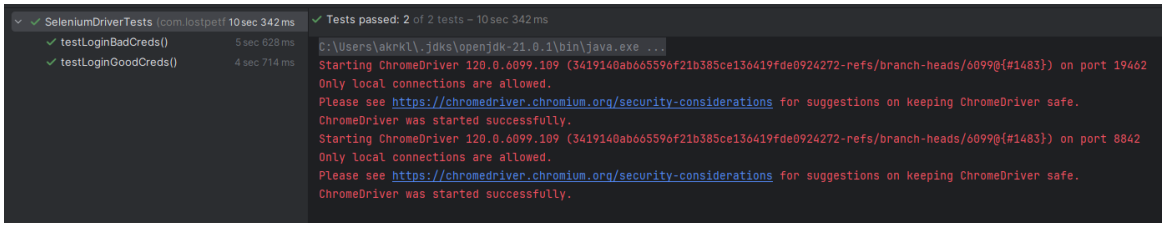
```
  WebDriverWait wait = new WebDriverWait(driver, Duration.
      ofSeconds(2));
  try {
  wait.until(
    ExpectedConditions.presenceOfElementLocated(By.id("
        searchBarByCategories"))
  );

  String redirectUrl = driver.getCurrentUrl();

  Assertions.assertEquals(
    "http://localhost:5173/", redirectUrl
  );
  }
  finally {
  driver.quit();
  }
}
```

U Kodu 5.8 testira se uspješan login korisnika. WebDriver otvara stranicu za login, pronalazi elemente za unos korisničkog imena i lozinke te unosi podatke. Nakon toga klikne na gumb za prijavu i čeka da se pojavi element za pretraživanje po kategorijama. Na kraju provjerava je li korisnik preusmjeren na početnu stranicu.

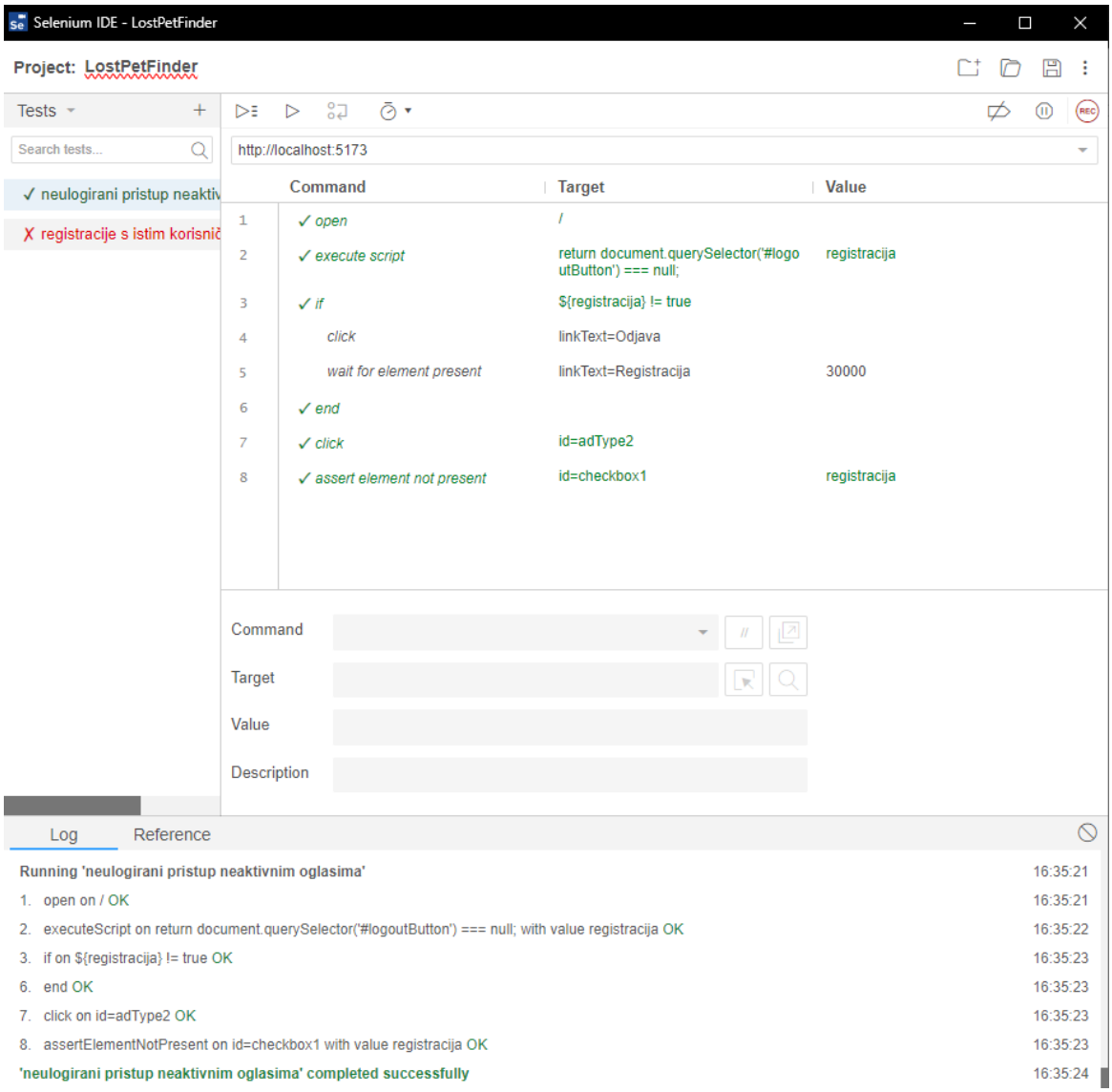
Kod 5.9: Test testLoginBadCreds

```
@Test
public void testLoginBadCreds() {
  WebDriver driver = new ChromeDriver();
  System.setProperty(
    "webdriver.chrome.driver",
    "C:\\Users\\akrkl\\AppData\\Local\\Microsoft\\WindowsApps\\
       chromedriver.exe"
  );
  driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
  driver.get("http://localhost:5173/login");

  WebElement element = driver.findElement(By.id("email"));
  element.sendKeys("testadmin");

  element = driver.findElement(By.id("lozinka"));
  element.sendKeys("wrongPassword");

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

  try {
    String redirectUrl = driver.getCurrentUrl();

    Assertions.assertNotEquals(
      "http://localhost:5173/", redirectUrl
    );
  }
  finally {
    driver.quit();
  }
}
```

U Kodu 5.9 testira se neuspješan login korisnika. WebDriver otvara stranicu za login, pronalazi elemente za unos korisničkog imena i lozinke te unosi podatke. Nakon toga klikne na gumb za prijavu i provjerava je li korisnik preusmjeren na početnu stranicu.



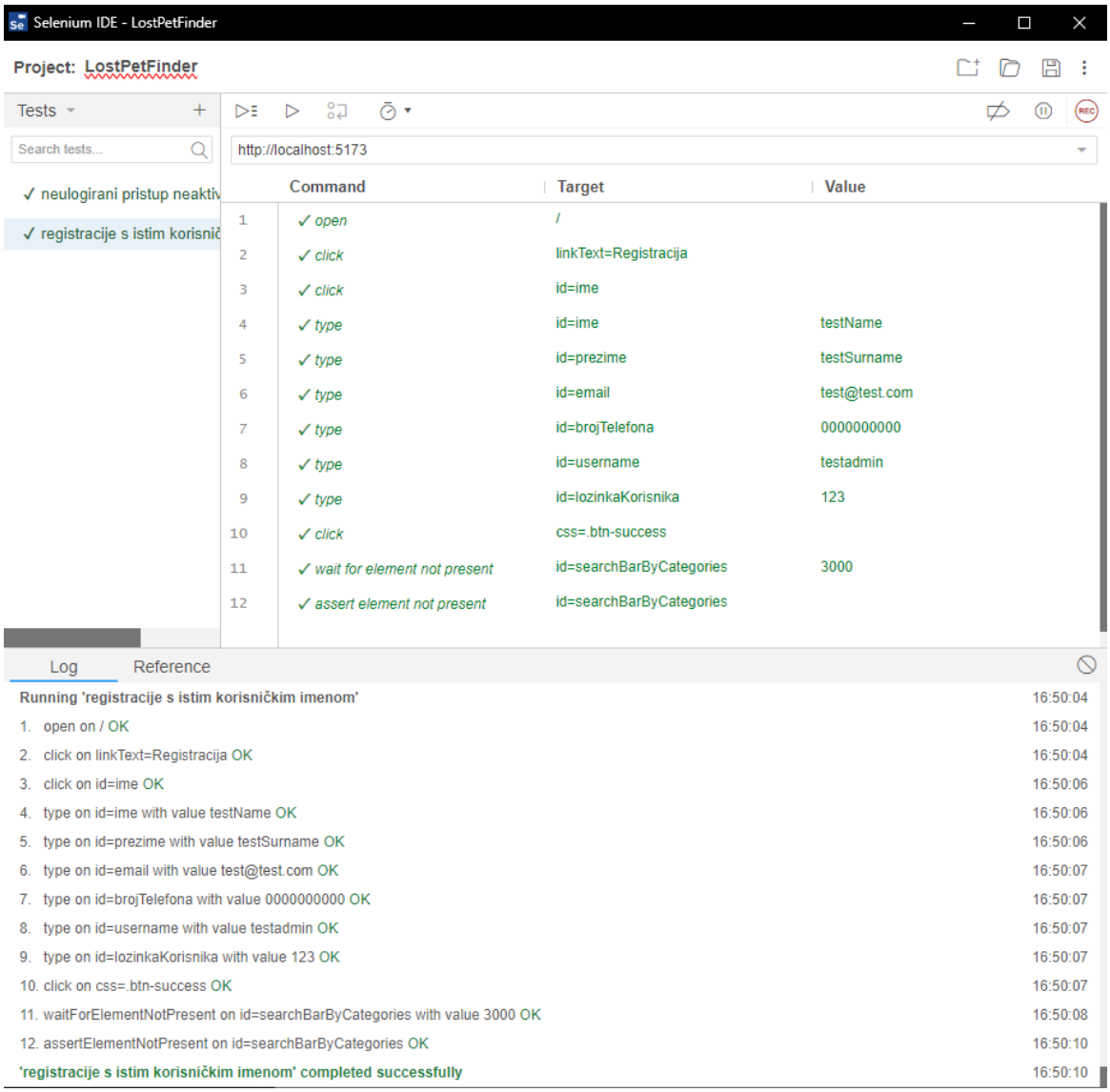
Slika 5.4: Testovi prolaze



Slika 5.5: Test: neulogirani korisnik pristupa neaktivnim oglasima

U ovom testu pomoću Selenium IDE-a provjerava se može li neulogirani koris-

nik pristupiti neaktivnim oglasima klikom na pripadajući radio button. Prvo se provjerava je li korisnik ulogiran, u tom slučaju se odjavljuje. Nakon toga se otvara stranica za prikaz oglasa, klikne se na radio button za prikaz neaktivnih oglasa i provjerava se je li se pojavio checkbox za filtriranje neaktivnih oglasa. Ako nije, test prolazi. Iz priloženog se vidi da aplikacija prolazi test.



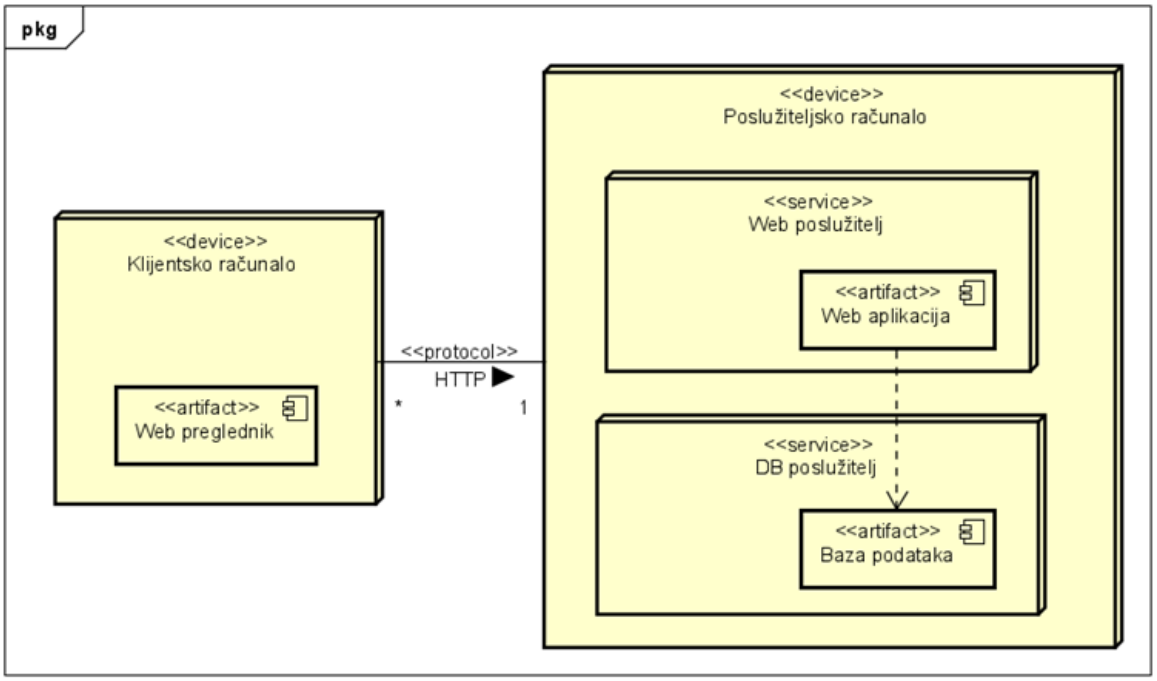
Slika 5.6: Test: registracija s već postojećim podacima korisnika

U ovom testu pomoću Selenium IDE-a provjerava se kako aplikacija reagira na pokušaj registracije korisnika s podacima već postojećeg korisnika. Nakon unosa klikne se na gumb koji šalje formu i provjerava se je li se pojavio element za pretraživanje po kategorijama, odnosno je li se dogodio redirect na početnu stranicu.

Ako nije, registracija nije uspjela i test prolazi. Iz priloženog se vidi da aplikacija prolazi test.

5.3 Dijagram razmještaja

Dijagrami razmještaja opisuju kako su komponente sustava raspoređene unutar njihovog radnog okruženja, uključujući sklopovlje i softversku podršku. Na poslužiteljskom računalu nalaze se web poslužitelj i poslužitelj baze podataka. Klijenti pristupaju web aplikaciji putem web preglednika. Arhitektura sustava temelji se na modelu "klijent-poslužitelj", gdje se komunikacija između računala korisnika (korisnik, registrirani korisnik, sklonište za životinje) i poslužitelja odvija preko HTTP veze.



Slika 5.7: Dijagram razmještaja

5.4 Upute za puštanje u pogon

dio 2. revizije

Sljedeće upute prikazuju kako pustiti aplikaciju pogon pomoću servisa Render[16].

Izvorni kod aplikacije prvo treba pripremiti u svoj repozitorij na GitHubu (izradom forka repozitorija ove aplikacije) te bi trebalo izraditi korisnički račun na Renderu na koji treba povezati svoj GitHub račun. Nakon toga treba stvoriti novi poslužitelj na Renderu te u njega postaviti izvorni kod aplikacije. Za to je potrebno napraviti sljedeće korake:

5.4.1 Izrada i postavljanje baze podataka aplikacije

Proces izrade baze podataka je sljedeći:

1. Na Render dashboardu odabrati *New – PostgreSQL*.
2. Popuniti polje *Name* (proizvoljno).
3. Odabrati *PostgreSQL Version* – 15.
4. Odabrati tip stroja (za osnovnu funkcionalnost je dovoljan *Free - 1 GB*).
5. Odabrati *Create Database*.

Nakon što je baza podataka stvorena, potrebno je postaviti konfiguraciju backenda da se može povezati na nju. To se radi na sljedeći način:

1. Na Render dashboardu odabrati bazu podataka nazvanu u prethodnom koraku.
2. U kategoriji *Info* pronaći skupinu *Connections* u kojem se trebaju nalaziti sljedeća polja: *Hostname*, *Port*, *Database*, *Username*, *Password* potrebna za idući korak.
3. Na temelju tih podataka treba urediti/dodati segment u datoteci *application.properties* koji se nalazi u direktoriju *IzvorniKod/backend/src/main/resources* prema sljedećem obrascu:

[16] https://render.com/

```
# Konfiguracija baze podataka
spring.datasource.password=${DB_PASS:⟨Password⟩}
spring.datasource.username=${DB_USERNAME:⟨Username⟩}
spring.datasource.url=${DB_URL:jdbc:postgresql://⟨Hostname⟩:⟨Port⟩/⟨Database⟩}
```

5.4.2 Puštanje u pogon backenda aplikacije

Uz preduvjet da je baza podataka kreirana i konfigurirana ispravno, proces puštanja backenda u pogon je sljedeći:

1. Na Render dashboardu odabrati *New – Web Service*.
2. Odabrati opciju *Build and deploy from GitHub*.
3. Odabrati repozitorij u kojem se nalazi izvorni kod aplikacije na povezanom GitHub računu pritiskom na tipku *Connect* pokraj njega.
4. Popuniti polje *Name* s proizvoljnim imenom za backend.
5. Odabrati odgovarajuću granu repozitorija u kojoj se nalazi izvorni kod aplikacije (obično je to `main`).
6. U polje *Root Directory* upisati `./IzvorniKod/backend`.
7. Za *Runtime* odabrati *Docker*.
8. Odabrati odgovarajući model računala u *Instance Type*
9. Na dnu stranice proširiti *Advanced* opcije.
10. U polje *Dockerfile Path* upisati `./Dockerfile`.
11. Potvrditi izradu poslužitelja pritiskom na *Create Web Service*.
12. Nakon što se poslužitelj izradi, potrebno je kopirati adresu poslužitelja koja se nalazi ispod dodijeljenog imena poslužitelja nakon što se klikne na njega na Render dashboardu (adresa će biti potrebna pri konfiguraciji frontenda).

5.4.3 Puštanje u pogon frontenda aplikacije

Nakon uspostavljanja backenda, sljedeći korak je puštanje u pogon frontenda. To se radi na sljedeći način:

1. Na Render dashboardu odabrati *New – Web Service*.
2. Odabrati opciju *Build and deploy from GitHub*.
3. Odabrati repozitorij u kojem se nalazi izvorni kod aplikacije na povezanom GitHub računu pritiskom na tipku *Connect* pokraj njega.
4. Popuniti polje *Name* s proizvoljnim imenom za frontend.
5. Odabrati odgovarajuću granu repozitorija u kojoj se nalazi izvorni kod aplikacije (obično je to `main`).
6. U polje *Root Directory* upisati `./IzvorniKod/frontend`.
7. Za *Runtime* odabrati *Node*.
8. U polje *Build Command* upisati `yarn`.
9. U polje *Start Command* upisati `yarn start-prod`.
10. Odabrati odgovarajući model računala u *Instance Type*.
11. Potrebno je dodati dvije varijable okruženja u *Environment Variables*:
    `API_BASE_URL` s vrijednosti `<Adresa poslužitelja backenda>`
    `PORT` s vrijednosti `5173`

Nakon uspješnog postavljanja svih triju komponenti, aplikacija je spremna za korištenje te se može posjetiti na adresi koja je dodijeljena frontendu.

6. Zaključak i budući rad

Zadatak naše grupe bio je razvoj web aplikacije za pronalazak nestalih ljubimaca gdje postoji mogućnost objavljivanja, izmjene i brisanja oglasa te razgovora putem chat-a s drugim sudionicima koji su uključeni u potragu za nestalim ljubimcima. Nakon 15 tjedana rada u timu i razvoja, ostvarili smo zadani cilj. Sama provedba projekta bila je kroz dvije faze.

Prva faza projekta uključivala je okupljanje tima za razvoj aplikacije, dodjelu projektnog zadatka i intenzivan rad na dokumentiranju zahtjeva. Kvaliteta provedbe prve faze uvelike je olakšala daljnji rad pri realizaciji osmišljenog sustava. Izrađeni obrasci i dijagrami(obrasci uporabe, sekvencijski dijagrami, model baze podataka, dijagram razreda) bili su od pomoći podtimovima zaduženima za razvoj *backenda* i *frontenda*. Izrada vizualnih prikaza idejnih rješenja problemskih zadataka uštedjela je mnogo vremena u drugom ciklusu kada su članovi tima nailazili na nedoumice oko implementacije rješenja.

Druga faza projekta, iako nešto kraća od prve, bila je puno intenzivnija po pitanju samostalnog rada članova. Manjak iskustva članova u izradi sličnih implementacijskih rješenja primorao je članove na samostalno učenje odabranih alata i programskih jezika kako bi ispunili dogovorene ciljeve. Osim realizacije rješenja, u drugoj fazi je bilo potrebno dokumentirati ostale UML dijagrame i izraditi popratnu dokumentaciju kako bi budući korisnici mogli lakše koristiti ili vršiti preinake na sustavu. Dobro izrađen kostur projekta uštedio nam je mnogo vremena prilikom izrade aplikacije te smo izbjegli moguće pogreške u izradi koje bi bile vremenski skupe za ispravljanje u daljnjoj fazi projekta.

Komunikacija među članovima tima bila je putem Discorda čime smo postigli informiranost svih članova grupe o napretku projekta. Moguće proširenje postojeće inačice sustava je izrada mobilne aplikacije čime bi cilj projektnog zadatka bio ostvaren u većoj mjeri.

Sudjelovanje na ovakvom projektu bio je vrijedno iskustvo svim članovima tima jer smo kroz intenzivnih nekoliko tjedana rada iskusili zajednički rad na istom projektu. Također, osjetili smo važnost dobre vremenske organiziranosti i koordiniranosti između članova tima. Zadovoljni smo postignutim bez obzira na golemi

prostor za usavršavanje izrađene aplikacije što je posljedica neiskustva na takvim i sličnim projektima.

Popis literature

1. Programsko inženjerstvo, FER ZEMRIS, http://www.fer.hr/predmet/proinz
2. I. Sommerville, "Software engineering", 8th ed, Addison Wesley, 2007.
3. T.C.Lethbridge, R.Langaniere, "Object-Oriented Software Engineering", 2nd ed. McGraw-Hill, 2005.
4. I. Marsic, Software engineering book", Department of Electrical and Computer Engineering, Rutgers University, http://www.ece.rutgers.edu/~marsic/books/SE
5. The Unified Modeling Language, https://www.uml-diagrams.org/
6. Astah Community, http://astah.net/editions/uml-new

Indeks slika i dijagrama

Dodatak: Prikaz aktivnosti grupe

Dnevnik sastajanja

1. sastanak
    - Datum: 16. listopada 2023.
    - Prisustvovali: A. Krklec, P. Marinić, V. Mesar, A. Prskalo, L. Raić, L. Rogoz, R. Vitaliani
    - Teme sastanka:
        - sastanak s asistentom i demonstratorom
        - raščišćavanje osnovnih dilema funkcionalnosti
        - analiza zadatka
        - plan i razrada obrade projektnog zadatka
        - definiranje i dodjela poslova
2. sastanak
    - Datum: 28. listopada 2023.
    - Prisustvovali: A. Krklec, P. Marinić, V. Mesar, A. Prskalo, L. Raić, L. Rogoz, R. Vitaliani
    - Teme sastanka:
        - konačan odabir alata i tehnologija
        - podjela rada
        - opis projektnog zadatka
        - funkcionalni zahtjevi, aktori i dionici
        - ostali zahtjevi
        - dijagrami obrazaca uporabe
        - opis obrazaca uporabe
3. sastanak
    - Datum: 5. studenog 2023.
    - Prisustvovali: A. Krklec, P. Marinić, V. Mesar, A. Prskalo, L. Raić, L. Rogoz, R. Vitaliani
    - Teme sastanka:
        - daljnja podjela rada

  - sekvencijski dijagrami
  - dijagram baze podataka
  - opis arhitekture i dizajn sustava
  - osmišljavanje stranice i pojedinih pregleda
  - početak rada na backendu i frontendu

4. sastanak
   - Datum: 10. studenog 2023.
   - Prisustvovali: A. Krklec, P. Marinić, V. Mesar, A. Prskalo, L. Raić, L. Rogoz, R. Vitaliani
   - Teme sastanka:
     - daljnja podjela rada
     - opis baze podataka
     - opis tablica baze podataka
     - opis arhitekture i dizajn sustava
     - rad na backendu i frontendu
5. sastanak
   - Datum: 13. studenog 2023.
   - Prisustvovali: A. Krklec, P. Marinić, V. Mesar, A. Prskalo, L. Raić, L. Rogoz, R. Vitaliani
   - Teme sastanka:
     - prva revizija dijagrama razreda
     - opis dijagrama razreda
     - sastanak s asistentom i demonstratorom - evaluacija rada
6. sastanak
   - Datum: 16. studenog 2023.
   - Prisustvovali: A. Krklec, P. Marinić, V. Mesar, A. Prskalo, L. Raić, L. Rogoz, R. Vitaliani
   - Teme sastanka:
     - privođenje 1. revizije web aplikacije i dokumentacije kraju
     - aplikacija puštena u pogon i poslane informacije o korisničkim imenima
7. sastanak
   - Datum: 17. prosinca 2023.
   - Prisustvovali: A. Krklec, P. Marinić, V. Mesar, A. Prskalo, L. Raić, L. Rogoz, R. Vitaliani

- Teme sastanka:
  - opis dijagrama stanja
  - opis dijagrama aktivnosti
  - planiranje rada na frontendu i backendu

8. sastanak
   - Datum: 18. prosinca 2023.
   - Prisustvovali: A. Krklec, P. Marinić, V. Mesar, A. Prskalo, L. Raić, L. Rogoz, R. Vitaliani
   - Teme sastanka:
     - rad na frontendu i backendu
     - planiranje daljnjih poslova
9. sastanak
   - Datum: 21. prosinca 2023.
   - Prisustvovali: A. Krklec, P. Marinić, V. Mesar, A. Prskalo, L. Raić, L. Rogoz, R. Vitaliani
   - Teme sastanka:
     - opis dijagrama komponenti
     - opis dijagrama razmještaja
     - rad na frontendu i backendu
10. sastanak
    - Datum: 14. siječnja 2024.
    - Prisustvovali: A. Krklec, P. Marinić, V. Mesar, A. Prskalo, L. Raić, L. Rogoz, R. Vitaliani
    - Teme sastanka:
      - ažurirani dijagrami razreda
      - ispravak sekvencijskih dijagrama
      - mala izmjena opisa projektnog zadatka

Tablica aktivnosti

	Patrik Marinić	Andrija Krklec	Vedran Mesar	Anđelko Prskalo	Luka Raić	Luka Rogoz	Robert Vitaliani
Upravljanje projektom	8						
Opis projektnog zadatka	3						5
Funkcionalni zahtjevi					3	5	
Opis pojedinih obrazaca		6			2		
Dijagram obrazaca					5		
Sekvencijski dijagrami				7			
Opis ostalih zahtjeva		2					
Arhitektura i dizajn sustava		3					
Baza podataka						6	
Dijagram razreda			6		3	5	
Dijagram stanja						5	
Dijagram aktivnosti			2			4	
Dijagram komponenti			5				
Korištene tehnologije i alati	1			1			3
Ispitivanje programskog rješenja	4	12	3	5	13	2	9
Dijagram razmještaja			4			2	
Upute za puštanje u pogon		6		11			3
Dnevnik sastajanja	2						4
Zaključak i budući rad							3
Popis literature	2	2	2	2	2	2	2

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

	Patrik Marinić	**Andrija Krklec**	**Vedran Mesar**	**Anđelko Prskalo**	**Luka Raić**	**Luka Rogoz**	**Robert Vitaliani**
Izrada baze podataka		5			6	8	
Spajanje s bazom podataka	1	4	1	3	4	2	1
Korisnička autentikacija i autorizacija	1	5			5	1	2
Backend		60	14	10	50	6	
Frontend	45						60
Testiranje	7	8	4	4	8	4	9
Deploy		6		16			

Dijagrami pregleda promjena

dio 2. revizije

Prenijeti dijagram pregleda promjena nad datotekama projekta. Potrebno je na kraju projekta generirane grafove s gitlaba prenijeti u ovo poglavlje dokumentacije. Dijagrami za vlastiti projekt se mogu preuzeti s gitlab.com stranice, u izborniku Repository, pritiskom na stavku Contributors.