Programsko inženjerstvo

Ak. god. 2023./2024.

Nestali ljubimci

Dokumentacija, Rev. *2*

Grupa: *I'm a teapot*

Voditelj: *Nora Ivić*

Datum predaje: *19.01.2024.*

Nastavnik: *Alan Jović*

Sadržaj

1 Dnevnik promjena dokumentacije 3

2 Opis projektnog zadatka 6

3 Specifikacija programske potpore 11

3.1 Funkcionalni zahtjevi 11

3.1.1 Obrasci uporabe 13

3.1.2 Dijagram baze podataka 19

3.1.3 Sekvencijski dijagrami 21

3.2 Ostali zahtjevi 25

4 Arhitektura i dizajn sustava 26

4.1 Baza podataka 27

4.1.1 Opis tablica 28

4.1.2 Dijagram baze podataka 31

4.2 Dijagram razreda 33

4.3 Dijagram stanja 41

4.4 Dijagram aktivnosti 42

4.5 Dijagram komponenti 44

5 Implementacija i korisničko sučelje 46

5.1 Korištene tehnologije i alati 46

5.2 Ispitivanje programskog rješenja 47

5.2.1 Ispitivanje komponenti 47

5.2.2 Ispitivanje sustava 57

5.3 Dijagram razmještaja 62

5.4 Upute za puštanje u pogon 63

5.4.1 Lokalni deploy aplikacije 63

5.4.2 Postupak izgradnje i objave aplikacije 65

6 Zaključak i budući rad 69

Popis literature **70**

Indeks slika i dijagrama **72**

Dodatak: Prikaz aktivnosti grupe **73**

1. Dnevnik promjena dokumentacije

Rev.	Opis promjene/dodatka	Autori	Datum
0.1	Napravljen predložak.	Marta Matulić	20.10.2023.
0.2	Aktori i funkcionalni zahtjevi.	Marta Matulić, Ivan Žinić	26.10.2023.
0.3	Dodan opis zadatka. Ubačene slike sličnih rješenja.	Marta Matulić, Ivan Žinić	27.10.2023.
0.4	Dodani opisi obrazaca uporabe.	Marta Matulić, Ivan Žinić	29.10.2023.
0.4.1	Dodani novi obrasci uporabe.	Ivan Žinić	31.10.2023.
0.5	Ostali zahtjevi. Ispravak opisa zadatka.	Marta Matulić, Ivan Žinić	01.11.2023.
0.6	Dijagrami baze podataka.	Marta Matulić	04.11.2023.
0.7	Dijagrami obrazaca uporabe. Sekvencijski dijagrami.	Marta Matulić, Ivan Žinić	05.11.2023.
0.7.1	Ispravak opisa zadatka.	Marta Matulić	08.11.2023.

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

Rev.	**Opis promjene/dodatka**	**Autori**	**Datum**
0.8	Opis baze podataka. Opis tablica baze.	Ivan Žinić	10.11.2023.
0.9	Opis arhitekture sustava.	Marta Matulić	10.11.2023.
0.9.1	Dopuna arhitekture sustava.	Marta Matulić	11.11.2023.
0.9.2	Ispravak opisa i dijagrama baze. Ispravak opisa zadatka.	Marta Matulić, Ivan Žinić	15.11.2023.
0.10	Dijagrami razreda. Opis dijagrama razreda.	Marta Matulić, Ivan Žinić	17.11.2023.
1.0	Ispravak pravopisnih grešaka. Gotova prva verzija dokumentacije.	Marta Matulić, Ivan Žinić	17.11.2023.
1.1	Korištene tehnologije i alati.	Marta Matulić	08.01.2024.
1.2	Zaključak i budući rad.	Ivan Žinić	09.01.2024.
1.3	Dijagram aktivnosti.	Marta Matulić	09.01.2024.
1.4	Druga revizija dijagrama razreda.	Marta Matulić	13.01.2024.
1.5	Dijagram stanja. Dijagram razmještaja.	Ivan Žinić	14.01.2024.
1.6	Dijagram komponenti.	Marta Matulić	15.01.2024.

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

Rev.	**Opis promjene/dodatka**	**Autori**	**Datum**
1.6.1	Dupuna dijagrama stanja. Dopuna dijagrama razmještaja.	Ivan Žinić	16.01.2024.
1.6.2	Dupuna dto dijagrama razreda. Dopuna repository dijagrama razreda.	Ivan Žinić	17.01.2024.
1.7	Testiranje komponenti.	Marta Matulić	18.01.2024.
1.7.1	Dopuna view dijagrama razreda.	Marta Matulić	18.01.2024.
1.8	Implementacija i korisničko sučelje	Marta Matulić, Ivan Žinić	19.01.2024.
2.0	Ispravak pravopisnih grešaka. Konačna verzija.	Marta Matulić, Ivan Žinić	19.01.2024.

2. Opis projektnog zadatka

Velik broj kućnih ljubimaca iz raznih razloga odluta od svojih vlasnika. Jako ih je teško samostalno pronaći, pogotovo u velikim gradovima s mnogo ljudi i gustim prometom. Kako je nepraktično oglašavati njihove nestanke, opažanja i pronalaske s preciznim podacima putem uobičajenih internetskih platformi za komunikaciju koje nisu razvijene s ovom namjenom (npr. društvene mreže, stranice za oglašavanje i sl.), potrebno je razviti jednostavnu i korisnu aplikaciju koja će vlasnicima kućnih ljubimaca, skloništima za životinje, ali i drugim uključenim ljudima olakšati rješavanje ovih stresnih situacija. Tako će napokon svi podaci o nestancima, opažanjima i pronalascima biti okupljeni na jednome mjestu, umjesto kao do sada raspršeni. Osobi će pri pronalasku ili nestanku ljubimca obavještavanje ostalih brižnih vlasnika i drugih ljudi biti udaljeno samo nekoliko klikova. Naša aplikacija razvija se upravu u tu svrhu. U današnje vrijeme kad je tehnologija lako i široko dostupna, gotovo svi imaju mobilne uređaje pri ruci u svakom trenutku. Zato će aplikacija biti ciljano razvijena za mobilne uređaje, te će za sve dionike u ovom procesu podržavati što veću brzinu reagiranja, koja je često ključna za uspješan završetak potrage za kućnim ljubimcem. Nadamo se da će aplikacija pomoći ponovo ujediniti što je moguće više vlasnika s njihovim sretno pronađenim ljubimcima, te na jednom mjestu okupiti sve ljude koji su voljni pomoći u potragama.

Slična aplikacija čije su slike dane kao primjer je američka aplikacija „MissingPetFinder“ koja je prilagođena korisnicima na njihovom području. Njihova aplikacija ne nudi opciju skloništa, već je prilagođena isključivo za osobnu upotrebu. Također nudi dodatne opcije poput printanja letaka i slanja notifikacija koje za sada naša aplikacija ne nudi, iako je takva nadogradnja moguća ukoliko bi se pokazalo korisnim.

Aplikacija treba podržavati rad tri tipa korisnika:

- Neregistrirani korisnik
- Registrirani korisnik
- Sklonište

Pri otvaranju aplikacije korisniku se otvara opcija za prijavu u sustav, uz dodatne opcije registracije i ulaska kao gost.

Prvi tip je **neregistrirani korisnik**. Pri ulasku u aplikaciju kao gost prikazuje mu se popis svih trenutno aktivnih oglasa o nestalim ljubimcima i osnovnim informacijama o njima (ime ljubimca, slike i broj mobitela vlasnika). Primjer sličnog rješenja nalazi se na slici ispod (slika 2.1).

Slika 2.1: Primjer sličnog rješenja

Takav korisnik može i pretraživati oglašene nestale kućne ljubimce i skloništa za životinje. Pretraživanje je omogućeno po svim kategorijama podataka o ljubimcu koje su dostupne pri oglašavanju (osim slike i lokacije, vidi dolje), kao i po nazivu skloništa. Također, odabirom nekog od kućnih ljubimaca otvara se mogućnost detaljnijeg pregleda informacija o njemu (sve informacije unesene pri objavi oglasa) kao i pregled dosadašnje komunikacije oko potrage za ljubimcem. Ako želi sudjelovati u potrazi za ljubimcem davanjem nekih informacija, neregistrirani korisnik se treba registrirati. Pri registraciji on unosi potrebne podatke kako bi kreirao svoj korisnički račun:

- Korisničko ime
- Lozinka
- Ime
- Prezime
- E-mail adresa

- Broj mobitela

Ako je uspješno unio sve podatke, registracija je uspješno provedena i korisnik je stvorio svoj profil u aplikaciji. U suprotnome će aplikacija zahtijevati ponovni unos podataka.

Drugi tip korisnika aplikacije je **registrirani korisnik**. On je u prošlosti već stvorio svoj profil. Pri ulasku u aplikaciju on odabire opciju za prijavu te unosi svoje korisničko ime i lozinku. Uspješna prijava omogućuje mu pristup dodatnim stavkama ove aplikacije. Klikom na ikonu na *toolbaru* može pregledati svoj profil. On, kao i neregistrirani korisnik ima opciju pregleda i pretraživanja oglasa, no njemu su pri pretraživanju dodatno dostupni ne samo aktivni, nego svi postojeći oglasi bez obzira koja im je kategorija oglasa postavljena. Osim ovih funkcija, nakon prijave korisnik također može i:

1. postaviti oglas o nestalom kućnom ljubimcu
2. sudjelovati u komunikaciji oko potrage za ljubimcem
3. ukloniti oglas o nestalom kućnom ljubimcu
4. izmijeniti oglas o nestalom kućnom ljubimcu

Takvom korisniku dostupna je opcija pregleda vlastitih oglasa. Kada ju odabere može pregledati sve oglase koje je on objavio, te mu postaju dostupne opcije za uklanjanje i izmjenu oglasa.

Kada želi prijaviti nestanak ljubimca, korisnik odabire opciju postavljanja novog oglasa. Postavljanje oglasa uključuje unos sljedećih kategorija podataka:

- Vrsta
- Ime na koje se odaziva
- Datum i sat nestanka
- Lokacija nestanka (bilježi se koristeći vanjsku uslugu za geolociranje *Mapbox*)
- Boja
- Starost
- Tekstni opis
- Slika (maksimalno tri po oglasu)

Oglas, osim ovih podataka, sadrži i kontakt podatke korisnika koji se automatski povlače iz korisničkih podataka danih pri registraciji (za običnog korisnika to su e-pošta i broj telefona, a dodatno za skloništa povlači se i naziv skloništa). Aplikacija pri unosu automatski provjerava je li korisnik unio sve podatke ispravno.

Ako je unos ispravan, oglas je objavljuje i postaje dostupan svim korisnicima, a u protivnome se korisnik obavještava o grešci i od njega se traži ponovni upis podataka.

Kao i neregistrirani korisnik, nakon prijave korisnik može odabrati bilo koji oglas i klikom na njega dobiti više informacija o tom ljubimcu (slika 2.2), kao i pregled dotadašnje komunikacije o potrazi. No, za razliku od neregistriranog, sada korisnik može i sudjelovati u komunikaciji ako misli da ima bilo kakvu korisnu informaciju (slika 2.3). Komunikacija omogućuje unos tekstualne poruke (npr. obavijestiti vlasnika o pronalasku ili pojavi sličnog ljubimca, zatražiti više informacija ako nije siguran i slično), slike i slanje geolokacije (putem vanjske usluge, isto kao i kod postavljanja oglasa). Pri slanju bilo koje informacije aplikacija također automatski bilježi i kontakt podatke osobe koja informaciju šalje.

Slika 2.2: Prikaz više informacija o ljubimcu kod sličnog rješenja

Slika 2.3: Komunikacija kod sličnog rješenja

Iz bilo kojih razloga, korisnik uvijek ima mogućnost ukloniti oglas koji je sa svoga profila postavio. Uklanjanjem oglasa određeni oglas i sva njegova komunikacija nestati će iz popisa vidljivih oglasa, ali se oglas ne briše iz baze podataka. Tako on više neće biti dostupan ni njemu ni drugim korisnicima na pregled.

Registriranom korisniku nudi se i opcija izmjene oglasa koje je on postavio. Izmjena oglasa omogućuje izmjenu svih kategorija podataka o ljubimcu (navedeni gore kod postavljanja oglasa). Moguće je izmijeniti i kategoriju oglasa. Izbor kategorija oglasa uključuje:

1. Za ljubimcem se traga
2. Ljubimac je sretno pronađen

3. Ljubimac nije pronađen, ali je potraga obustavljena
4. Ljubimac je pronađen uz nesretne okolnosti

Po izvornim postavkama, pri objavi oglasa kategorija je automatski namještena da se za ljubimcem trenutno traga. Svaka izmjena kategorije oglasa u onu koja nije da se za ljubimcem aktivno traga automatski prebacuje oglas u popis neaktivnih oglasa, koji mogu pretraživati samo registrirani korisnici.

Treći tip korisnika su **skloništa za životinje**. Skloništa za životinje su specijalni tip registriranih korisnika. Pri registraciji, kada se želi stvoriti profil za sklonište, potrebno je odabrati opciju „sklonište“. Unose se iste informacije kao i pri registraciji običnog korisnika, osim što se umjesto imena i prezimena osobe bilježi naziv skloništa. Prijava funkcionira isto kao i kod prijave običnih korisnika. U aplikaciji skloništa imaju dostupne sve opcije koje se nude i registriranim korisnicima (pregled i pretraživanje, postavljanje, uklanjanje i izmjena oglasa, te komunikacija oko potrage). Međutim, zbog toga što skloništa predstavljaju privremeni dom za mnoge tek pronađene ljubimce, kako bi se vlasnicima olakšalo lociranje njihovih ljubimaca, skloništa imaju i dodatnu mogućnost oglašavanja životinja koje su pronašli i koje se nalaze u njihovom prostoru. Zato skloništa pri postavljanju oglasa imaju ponuđenu i dodatnu opciju „u skloništu“ kojima tražene vlasnike obavještavaju da je ljubimac privremeno smješten kod njih, te čeka da vlasnici dođu po njega. Također, ako pronađu ljubimca čiji oglas već postoji, kada odaberu pregled detalja tog oglasa nudi im se opcija da označe da se taj ljubimac nalazi kod njih.

Ukoliko se aplikacija pokaže korisnom, nudi i brojne mogućnosti nadogradnje. Moguće bi bilo proširenje informacija koje se bilježe o ljubimcima, formiranje grupa za razgovor među vlasnicima koje bi mogle poslužiti i kao utjeha tijekom potrage za ljubimcima, dodavanje opcije videa, nagrada za pronalazak ljubimca ako vlasnik to želi, slanje obavijesti o nestancima u blizini i još mnogo drugih opcija.

3. Specifikacija programske potpore

3.1 Funkcionalni zahtjevi

Dionici:

1. Neregistrirani korisnik
2. Registrirani korisnik
   a) Sklonište
3. Razvojni tim

Aktori i njihovi funkcionalni zahtjevi:

1. Neregistriran/neprijavljen korisnik (inicijator) može:
   (a) pregledavati aktivne oglase
   (b) pretraživati aktivne oglase
   (c) registrirati se
   (d) prijaviti se u sustav (za postojećeg korisnika)
   (e) detaljnjije pregledati oglas i komunikacije
2. Registriran/prijavljen korisnik (inicijator) može:
   (a) pregledavati sve oglase
   (b) pretraživati sve oglase
   (c) detaljnije pregledati oglas i komunikacije
   (d) komunikacirati u vezi potrage
   (e) postaviti oglas
   (f) ukloniti oglas
   (g) izmjeniti oglas
3. Sklonište (inicijator) može:
   (a) pregledavati sve oglase
   (b) pretraživati sve oglase
   (c) detaljnijije pregledati oglas i komunikacije

(d) komunicirati u vezi potrage
(e) ukloniti oglas
(f) izmijeniti oglasa
(g) postaviti oglas o ljubimcu u skloništu

4. Baza podataka (sudionik) može:
   (a) pohranjivati sve podatke o korisničkim računima
   (b) pohranjivati sve podatke o oglasima nestalih ljubimaca
   (c) pohranjivati komunikaciju o nestalim ljubimcima

3.1.1 Obrasci uporabe

Opis obrazaca uporabe

UC1 -Registracija

- **Glavni sudionik:** neregistriran korisnik
- **Cilj:** stvoriti korisnički račun za pristup aplikaciji
- **Sudionici:** baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
  1. korisnik odabire opciju za registraciju
  2. korisnik unosi potrebne korisničke podatke
  3. korisnik prima obavijest o uspješnoj registraciji
  4. unos podataka u bazu podataka
- **Opis mogućih odstupanja:**

  2.a odabir već zauzetog korisničkog imena ili emaila, unos korisničkih podataka u nedozvoljenom formatu
    1. sustav obavještava korisnika o neuspjelom upisu i vraća ga na stranicu za registraciju
    2. korisnik mijenja potrebne podatke te završava unos ili odustaje od registracije

UC2 -Prijava

- **Glavni sudionik:** registriran korisnik
- **Cilj:** prijava u postojeći korisnički račun za pristup aplikaciji
- **Sudionici:** baza podataka
- **Preduvjet:** registracija
- **Opis osnovnog tijeka:**
  1. korisnik odabire opciju za prijavu
  2. korisnik unosi korisničko ime i lozinku
  3. potvrda o ispravnosti unesenih podataka
  4. pristup korisničkim funkcijama
- **Opis mogućih odstupanja:**

  3.a neispravno korisničko ime ili lozinka
    1. sustav obavještava korisnika o neuspjelom upisu
    2. vraćanje na stranicu za prijavu

UC3 -Pregled oglasa

- **Glavni sudionik:** neregistriran korisnik, registriran korisnik
- **Cilj:** pregledati oglase
- **Sudionici:** baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
  1. oglasi su prikazani prilikom učitavanja aplikacije
  2. korisnik pregledava oglase

UC4 -Pretraživanje aktivnih oglasa

- **Glavni sudionik:** neregistriran korisnik, registriran korisnik
- **Cilj:** pronaći željeni aktivni oglas
- **Sudionici:** baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
  1. korisnik odabire opcije za pretraživanje
  2. korisnik unosi željene podatke o kategorijama
  3. korisnik odabire opciju "Pretraži"
  4. prikaz pronađenih oglasa
- **Opis mogućih odstupanja:**

  3.a nijedan oglas ne ispunjava kriterije pretrage
    1. korisnika se obavještava da nema pronađenih oglasa

UC5 -Pretraživanje neaktivnih oglasa

- **Glavni sudionik:** registriran korisnik
- **Cilj:** pronaći željeni neaktivni oglas
- **Sudionici:** baza podataka
- **Preduvjet:** prijava
- **Opis osnovnog tijeka:**
  1. korisnik odabire opcije za pretraživanje
  2. korisnik unosi željene podatke o kategorijama
  3. korisnik odabire opciju "Pretraži"
  4. prikaz pronađenih oglasa
- **Opis mogućih odstupanja:**

  3.a nijedan oglas ne ispunjava kriterije pretrage

1. korisnika se obavještava da nema pronađenih oglasa

UC6 - Postavljanje oglasa

- **Glavni sudionik:** registriran korisnik
- **Cilj:** postaviti oglas o izgubljenom kućnom ljubimcu
- **Sudionici:** baza podataka
- **Preduvjet:** prijava
- **Opis osnovnog tijeka:**
  1. korisnik odabire opciju za postavljanje novog oglasa
  2. korisnik korisnik unosi informacije o nestalom ljubimcu
  3. automatsko povlačenje korisničkih podataka
  4. korisnik odabire opciju za objavu
  5. spremanje podataka u bazu podataka
- **Opis mogućih odstupanja:**

  4.a neispravan unos jedne ili više kategorija podataka o ljubimcu
    1. korisnika se vraća na ponovni upis oglasa

UC7 - Komunikacija oko potrage

- **Glavni sudionik:** registriran korisnik
- **Cilj:** razmjena informacija o određenom izgubljenom ljubimcu
- **Sudionici:** baza podataka
- **Preduvjet:** prijava, postojeći aktivni oglas
- **Opis osnovnog tijeka:**
  1. korisnik odabire opciju za komunikaciju
  2. korisnik korisnik unosi informacije, sliku i lokaciju nestalog ljubimca
  3. korisnik odabire opciju za slanje
  4. nove informacije se prikazuju svim korisnicima
  5. spremanje u bazu podataka

UC8 - Detaljni pregled oglasa i komunikacije

- **Glavni sudionik:** neregistriran korisnik, registriran korisnik
- **Cilj:** pregled svih informacija i dosadašnje komunikacije o određenom ljubimcu
- **Sudionici:** baza podataka
- **Preduvjet:**
- **Opis osnovnog tijeka:**

1. korisnik odabire opciju „Prikaži više“
2. prikaz svih podataka i komunikacije

UC9 - Uklanjanje vlastitih oglasa

- **Glavni sudionik:** registriran korisnik
- **Cilj:** ukloniti postojeći oglas s popisa vidljivih oglasa
- **Sudionici:** baza podataka
- **Preduvjet:** prijava, pregled vlastitih oglasa, postojeći vlastiti oglas
- **Opis osnovnog tijeka:**
  1. korisnik odabire opciju za uklanjanje oglasa
  2. nestanak oglasa s liste vidljivih oglasa
  3. ažuriranje baze podataka

UC10 - Izmjena vlastitih oglasa

- **Glavni sudionik:** registriran korisnik
- **Cilj:** izmijeniti željene kategorije podataka o ljubimcu i kategoriju vlastitog oglasa
- **Sudionici:** baza podataka
- **Preduvjet:** prijava, pregled vlastitih oglasa, postojeći vlastiti oglas
- **Opis osnovnog tijeka:**
  1. korisnik odabire opciju za izmjenu oglasa
  2. korisnik izmjenjuje podatke ili kategoriju oglasa o ljubimcu
  3. korisnik odabire opciju za spremanje promjena
  4. ažuriranje baze podataka

UC11 - Oglašavanje ljubimaca u skloništu

- **Glavni sudionik:** sklonište
- **Cilj:** obavijestiti u kojem se skloništu nalazi nestali ljubimac
- **Sudionici:** baza podataka
- **Preduvjet:** prijava
- **Opis osnovnog tijeka:**
  1. odabir opcije za postavljanje novog oglasa
  2. unos informacija o nestalom ljubimcu
  3. automatsko povlačenje korisničkih podataka
  4. odabir opcije za objavu
  5. spremanje podataka u bazu podataka

- **Opis mogućih odstupanja:**
  4.a neispravan unos jedne ili više kategorija podataka o ljubimcu
    1. vraćanje na ponovni upis oglasa

UC12 - Pregled vlastitih oglasa

- **Glavni sudionik:** registriran korisnik, sklonište
- **Cilj:** pregled vlastitih oglasa koje je korisnik objavio
- **Sudionici:** baza podataka
- **Preduvjet:** prijava, postojeći vlastiti oglas
- **Opis osnovnog tijeka:**
  1. korisnik odabire opciju za prikaz vlastitih oglasa
  2. korisnik pregledava sve svoje oglase
- **Opis mogućih odstupanja:**
  1.a korisnik do sad nije objavio nijedan oglas
    1. korisnika se obavještava da nema izrađenih oglasa

UC13 - Izmjena tuđih oglasa

- **Glavni sudionik:** sklonište
- **Cilj:** označiti ljubimca s tuđeg oglasa kao smještenog u sklonište
- **Sudionici:** baza podataka, registriran korisnik
- **Preduvjet:** prijava, postojeći aktivni oglas
- **Opis osnovnog tijeka:**
  1. odabir opcije za detaljniji pregled tuđeg oglasa
  2. označavanje ljubimca kao smještenog u sklonište
  3. ažuriranje baze podataka

UC14 -Pregled korisničkog profila

- **Glavni sudionik:** registriran korisnik
- **Cilj:** pregledati podatke o svom profilu
- **Sudionici:** baza podataka
- **Preduvjet:** prijava
- **Opis osnovnog tijeka:**
  1. korisnik odabire opciju za prikaz svog profila preko ikone na toolbaru
  2. korisnik pregledava informacije o svom profilu

UC15 - Pregled liste skloništa

- **Glavni sudionik:** neregistriran korisnik, registriran korisnik
- **Cilj:** pregledati skloništa
- **Sudionici:** baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
  1. odabir opcije za prikaz liste skloništa
  2. korisnik pregledava sva prijavljena skloništa
- **Opis mogućih odstupanja:**

  1.a ne postoje prijavljena skloništa za ljubimce
    1. korisnika se obavještava da nema skloništa

Dijagrami obrazaca uporabe

3.1.2 Dijagram baze podataka

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

Slika 3.2: Dijagram obrasca uporabe, funkcionalnost skloništa

3.1.3 Sekvencijski dijagrami

Obrazac uporabe UC4 - Pretraživanje aktivnih oglasa

Pokretanjem aplikacije korisnik šalje zahtjev za prikaz aktivnih oglasa. Poslužitelj vraća trenutno aktivne oglase i prikazuje ih. Zatim korisnik odabire opciju za pretraživanje pri čemu mu aplikacija otvara polja za unos filtera. Nakon unosa, korisnik šalje željene filtere poslužitelju koji zatim komunicira s bazom. Ukoliko postoje oglasi koji zadovoljavaju uvjete pretraživanja oni se prikazuju korisniku, a u suprotnom mu se ispisuje odgovarajuća poruka.

Slika 3.3: Sekvencijski dijagram - UC4

Obrazac uporabe UC6 - Postavljanje oglasa

Pokretanjem aplikacije korisnik šalje zahtjev za prikaz aktivnih oglasa. Poslužitelj vraća trenutno aktivne oglase i prikazuje ih. Zatim korisnik odabire opciju postavljanja novog oglasa i poslužitelj mu otvara prozor za unos informacija. Korisnik šalje podatke za novi oglas, te poslužitelj provjerava ispravnost unosa. Ako je

sve ispravno uneseno, novi oglas se sprema u bazu i prikazuje korisniku. Inače poslužitelj korisnika obavještava o neispravnom unosu podataka.

Slika 3.4: Sekvencijski dijagram - UC6

Obrazac uporabe UC7 - Komunikacija oko potrage

Pokretanjem aplikacije korisnik šalje zahtjev za prikaz aktivnih oglasa. Poslužitelj vraća trenutno aktivne oglase i prikazuje ih. Korisnik odabire oglas, te mu poslužitelj iz baze podataka dohvaća sve informacije o tom oglasu i dosadašnju komunikaciju. Registrirani korisnik može sudjelovati u komunikaciji sve dok ne zatvori odabrani oglas. On unese poruku, šalje je poslužitelju koji je sprema u bazu podataka. Nakon spremanja poruka je vidljiva svim korisnicima.

Slika 3.5: Sekvencijski dijagram - UC7

Obrazac uporabe UC10 - Izmjena vlastitih oglasa

Pokretanjem aplikacije korisnik šalje zahtjev za prikaz aktivnih oglasa. Poslužitelj vraća trenutno aktivne oglase i prikazuje ih. Zatim korisnik traži prikaz njegovih vlastitih oglasa pa ih poslužitelj pokušava dohvatiti iz baze. Ukoliko postoje, prikazuju se u aplikaciji. Sada može odabrati opciju izmjene čime mu poslužitelj otvara prozor za izmjenu oglasa sa svim informacijama o odabranom oglasu iz baze. Registrirani korisnik unosi informacije i šalje ih poslužitelju koji ih zatim provjerava. Ako je sve ispravno uneseno, promjene se spremaju u bazu podataka i izmjenjeni oglas se prikazuje korisniku. Inače, ispisuje mu se poruka o neispravnom unosu. Ako korisnik nije do sada objavio nijedan oglas, ispisuje mu se odgovarajuća poruka.

Slika 3.6: Sekvencijski dijagram - UC10

3.2 Ostali zahtjevi

- Sustav treba biti implementiran kao mobilna aplikacija koristeći objektno-orijentirane jezike
- Sustav treba podržavati rad više korisnika u stvarnom vremenu
- Administracija podataka obavlja se kroz sučelje baze podataka
- Sustav i korisničko sučelje trebaju podržavati hrvatsku abacedu pri prikazu i unosu tekstualnog sadržaja
- Neispravna upotreba korisničkog sučelja ne smije narušiti funkcionalnost
- Sustav treba biti jednostavan za korištenje svim klijentima
- Veza s bazom podataka treba biti kvalitetno zaštićena, brza i otporna na vanjske greške
- Postojeće funkcionalnosti ne smiju biti narušene pri nadogradnji sustava
- Izvršavanje dijela programa u kojem se pristupa bazi podataka ne smije trajati više od nekoliko sekundi

4. Arhitektura i dizajn sustava

Arhitekturu možemo podijeliti na tri podsustava (slika 4.1):

- Mobilna aplikacija
- Poslužitelj
- Baza podataka

Slika 4.1: Arhitektura sustava

Korisničko sučelje aplikacije korisniku omogućuje pregled i unos podataka u aplikaciju. Sučelje je jednostavno i prilagođeno za jednostavnu uporabu svima. Korisnik preko sučelja šalje zahtjeve na poslužitelj.

Poslužitelj služi za komunikaciju klijenta s aplikacijom. Komunikacija se odvija preko HTTP (engl. *Hyper Text Transfer Protocol*) protokola, što je protokol u prijenosu informacija. Poslužitelj je onaj koji pokreće aplikaciju te joj prosljeđuje zahtjeve.

Mobilna aplikacija obrađuje korisničke zahtjeve, te ovisno o zahtjevu pristupa bazi podataka i preko poslužitelja vraća odgovor korisniku preko .json dokumenta kojeg zatim sučelje prikazuje na način razumljiv svakom korisniku.

Za izradu naše aplikacije odabrali smo programski jezik Python s FastAPI radnim okvirom te programski jezik Kotlin. Razvojna okruženja koja koristimo su PyCharm za Python i Android Studio za Kotlin.

Arhitektura aplikacije temeljit će se na MVC (Model-View-Controller) konceptu (slika 4.2).

Slika 4.2: MVC koncept

Koncept dijeli aplikaciju na tri sloja – model, view i controller. Svaki sloj izvršava određeni skup zadataka, a svi slojevi djeluju zajedno kako bi ostvarili traženu funkcionalnost aplikacije. Omogućen je nezavisan razvoj pojedinih dijelova čime se pojednostavljuje razvoj i testiranje.

Model je središnja komponenta ovog koncepta. Upravlja podacima i logikom aplikacije, komunicira s bazom te nema dodira sa sučeljem. Prima zahtjeve od upravljača da se ažurira.

View sadrži komponente aplikacije vidljive korisniku. Omogućuje vizualizaciju podataka spremljenih u modelu i nudi interakciju korisnikom.

Controller prima unesene podatke i prevodi ih u naredbe za model i view. Upravlja zahtjevima korisnika i prenosi ih ostalim elementima.

4.1 Baza podataka

Za potrebe našeg sustava, odabrali smo relacijsku bazu podataka zbog njezine strukture koja olakšava modeliranje stvarnog svijeta. Osnovna građevna jedinica baze je relacija, odnosno tablica koja je identificirana imenom i skupom atributa.

Glavna funkcija baze podataka je brza i jednostavna pohrana, izmjena i dohvat podataka za daljnju obradu. Baza podataka naše aplikacije uključuje sljedeće entitete:

- UserCustom *(korisnik)*
- UserAuth *(prijava za korisnika)*
- Advertisement *(oglas)*
- Pet *(ljubimac)*
- Picture *(slika)*
- Message *(poruka)*

4.1.1 Opis tablica

UserCustom Ovaj entitet sadržava sve važne informacije o korisniku aplikacije. Sadrži atribute: id (korisnički identifikacijski broj), username (korisničko ime), isShelter, firstName (ime korisnika), lastName (prezime korisnika), shelterName (ime skloništa ako je riječ o skloništu), email i phoneNumber (broj telefona). Ovaj entitet u vezi je *One-to-One* s entitetom UserAuth preko atributa username (korisničko ime korisnika), u vezi *One-to-Many* s Message (poruka) preko korisničkog identifikacijskog broja, u vezi *One-to-Many* s entitetom Advertisement preko korisničkog identifikacijskog broja (ili identifikacijskog broja skloništa).

UserCustom		
id	INT	jedinstveni brojčani identifikator korisnika
username	VARCHAR	jedinstveni identifikator korisnika
isShelter	BOOLEAN	oznaka je li korisnik sklonište
firstName	VARCHAR	ime korisnika
lastName	VARCHAR	prezime korisnika
shelterName	VARCHAR	ime skloništa
email	EMAIL *(definirano u bazi podataka)*	e-mail adresa korisnika
phoneNumber	VARCHAR	broj telefona korisnika

UserAuth Ovaj entitet sadržava sve što je potrebno kako bi se u sustav prijavio korisnik, a to je njegov username (korisničko ime) i password (pripadajuća

loznika).Ovaj entitet je u vezi *One-to-One* s entitetom UserCustom preko atributa username (korisničko ime).

UserAuth		
username	VARCHAR	jedinstveni identifikator korisnika
password	VARCHAR	hash lozinke

Advertisement Ovaj entitet sadržava sve važne informacije o oglasima koje će korisnici moći postavljati ili čitati. Sadrži atribute: id (identifikacijski broj oglasa), category (kategoriju), deleted (podatak o tome je li oglas izbrisan), dateTimeAdv (vrijeme oglašavanja), isInShelter (podatak o tome je li oglas ljubimca označen da je u skloništu), userId (ID korisnika), petId (ID ljubimca) i shelterId (ID skloništa). Ovaj entitet je u vezi *Many-to-One* s entitetom UserCustom preko atributa userId i shelterId, u vezi *One-to-Many* s entitetom Message preko identifikacijskog broja oglasa, u vezi *One-to-Many* s entitetom Picture također preko identifikacijskog broja oglasa te u vezi *One-to-One* s entitetom Pet (ljubimcem) preko identifikacijskog broja ljubimca.

Advertisement		
id	INT	jedinstveni brojčani identifikator oglasa
category	ENUM	kategorije: lost, found, abandoned, sheltered, dead
deleted	BOOLEAN	oznaka je li oglas izbrisan
dateTimeAdv	TIMESTAMP	vrijeme postavljanja oglasa
isInShelter	BOOLEAN	oznaka je li ljubimac smješten u sklonište
userId	INT	jedinstveni brojčani identifikator korisnika
petId	INT	jedinstveni brojčani identifikator ljubimca
shelterId	INT	jedinstveni brojčani identifikator skloništa

Pet Ovaj entitet sadržava sve važne informacije o nestalom ljubimcu koje će biti od pomoći pri potrazi. Sadrži atribute: id (identifikacijski broj ljubimca), species (vrsta ljubimca), name (ime), color (boja), dateTimeLost (vrijeme kad je ljubimac izgubljen), locationLost (lokacija na kojoj je izgubljen) i description (opis).

Ovaj entitet je u vezi *One-to-One* s entitetom Advertisement preko atributa identifikacijskog broja ljubimca.

Pet		
id	INT	jedinstveni brojčani identifikator ljubimca
species	ENUM	vrsta životinje
name	VARCHAR	ime ljubimca
color	VARCHAR	boja ljubimca
age	INT	starost ljubimca
dateTimeLost	TIMESTAMP	vrijeme kad je ljubimac izgubljen
locationLost	VARCHAR	opisuje lokaciju na kojoj je ljubimac izgubljen
description	TEXT	sve dodatne informacije o ljubimcu

Picture Ovaj entitet omogućava pohranu i korištenje slika koje se koriste pri potrazi za ljubimcima. Sadrži atribute: id (identifikacijski broj slike), link, advertId (identifikacijski broj oglasa) i messageId (identifikacijski broj poruke) Ovaj entitet je u vezi *Many-to-One* s entitetom Advertisement preko atributa identifikacijskog broja ljubimca te je u vezi *Many-to-One* s entitetom Message preko ID-a poruke.

Picture		
id	INT	jedinstveni brojčani identifikator slike
link	VARCHAR	poveznica prema slici
advertId	INT	jedinstveni brojčani identifikator oglasa
messageId	INT	jedinstveni brojčani identifikator poruke

Message Ovaj entitet sadržava sve važne informacije tijekom komunikacije korisnika prilikom potrage za nestalim ljubimcima. Sadrži atribute: id (identifikacijski broj poruke), text, location (lokaciju), dateTimeMess (vrijeme slanja poruke), advertId (ID oglasa) i userId (ID korisnika koji je poslao poruku). Ovaj entitet je u vezi *Many-to-One* s entitetom UserCustom preko atributa identifikacijskog broja korisnika, u vezi *Many-to-One* s entitetom Advertisement preko ID-a oglasa te je u vezi *Many-to-One* s entitetom Picture preko identifikacijskog broja poruke.

Message		
id	INT	jedinstveni brojčani identifikator poruke
text	TEXT	tekstualni dio poruke
location	VARCHAR	lokacija koju korisnik može poslati
dateTimeMess	TIMESTAMP	vrijeme slanja poruke
advertId	INT	jedinstveni brojčani identifikator oglasa
userId	INT	jedinstveni brojčani identifikator korisnika

4.1.2 Dijagram baze podataka

Slika 4.3: Relacijski dijagram baze podataka

Slika 4.4: E-R dijagram baze podataka

4.2 Dijagram razreda

Na slikama 4.8, 4.9, 4.10 i 4.11 su prikazani razredi koji pripadaju *backend* dijelu MVC arhitekture. Na slikama 4.5, 4.6 i 4.7 prikazani su razredi *frontend* dijela arhitekture, podjeljeni na 3 povezana dijagrama zbog lakšeg snalaženja. Razredi prikazani na slici 4.10 prikazuju *Repository* koji je zadužen za interakciju s bazom. Razredi na slici 4.11 nasljeđuju razred *APIRouter* i predstavljaju *controllere*. Metode implementirane u tim razredima manipuliraju s DTO *(Data transfer object)*, a oni su dohvaćeni pomoću metoda implementiranih u *Model* razredima.

Zbog lakše organizacije, razredi su podijeljeni logički po pravu pristupa metodama određenih aktora. Prikazane su isključivo ovisnosti između razreda koji pripadaju istom dijelu dijagrama. Ostale ovisnosti mogu se zaključiti iz naziva i tipova atributa u razredima.

Slika 4.5: Dijagram razreda - 1. dio View: Pets i Shelters klase

Slika 4.6: Dijagram razreda - 2. dio View: Login, Register i Map klase

Slika 4.7: Dijagram razreda - 3. dio View: Advert i Message klase

Model razredi preslikavaju strukturu baze podataka u aplikaciji. Implementirane metode direktno komuniciraju s bazom podataka te vraćaju tražene podatke. Razred UserCustom predstavlja registriranog korisnika koji ima pristup svim stavkama aplikacije. Razred UserAuth predstavlja podatke za prijavu registriranog korisnika. Razred Advertisement predstavlja objavljeni oglas. Pet je razred koji sadrži unesene podatke o nestalom ljubimcu. Razred message predstavlja poruke

vezane za komunikaciju o potrazi. Razred Picture predstavlja sliku koja se nalazi ili u oglasu ili u poruci. PetSpecies je enumeracija s mogućim vrstama ljubimca, a AdvertisementCategory enumeracija s kategorijama oglasa.

Slika 4.8: Dijagram razreda - dio Models

Slika 4.9: Dijagram razreda - dio Data transfer objects

Slika 4.10: Dijagram razreda - dio Repository

Slika 4.11: Dijagram razreda - dio Controllers

4.3 Dijagram stanja

Koristeći aplikaciju korisnik prelazi iz jednog stanja u drugo. Prijelaz iz stanja prikazuje se na dijagramu. Na slici 4.12 nalazi se dijagram stanja za klijenta aplikacije. Nakon što korisnik obavi prijavu, prikazuje mu se početna stranica na kojoj može listati oglase. S pomoću jasno definiranih tipki lako se može navigirati kroz aplikaciju odabirući nekoliko mogućnosti. Može objaviti novi oglas, pregledati svoj profil, vidjeti sve vlastite oglase ili pretražiti postojeće oglase.

Slika 4.12: Dijagram stanja

4.4 Dijagram aktivnosti

UML-dijagrami aktivnosti su ponašajni UML-dijagrami koji se upotrebljavaju za modeliranje i vizualizaciju dinamičkog ponašanja sustava. Prikazuju tijek aktivnosti, akcija i odluka procesa u sustavu. Svaki novi korak ovisi o završetku i rezultatu prethodnog. Na dijagramu 4.13 prikazan je proces kreiranja novog oglasa za nestalog ljubimca. Korisnik se prvo prijavljuje u sustav, a zatim odabire opciju za objavu novog oglasa. Unosi željene podatke o ljubimcu koji se onda provjeravaju u aplikaciji i na bazi. Ako su neispravni aplikacija traži ponovni unos, inače se spremaju u bazu. Novi oglas se zatim prikazuje korisniku.

Slika 4.13: Dijagram aktivnosti - Postavljanje oglasa

4.5 Dijagram komponenti

Na slici 4.14 prikazan je dijagram komponenti naše mobilne aplikacije. Opisuje organizaciju i međuovisnost komponenti. Instalacijom aplikacije na mobilni uređaj korisnika, na uređaju se nalaze sve datoteke koje pripadaju *frontend* dijelu aplikacije. *Frontend* dio aplikacije sastoji se od niza Kotlin datoteka koje su raspoređene u logičke cjeline s obzirom na funkcionalnosti koje ostvaruju. Sve Kotlin datoteke ovise o nizu Kotlin biblioteka. PetsAPIService je komponenta koja dohvaća podatke sa poslužitelja i određuje koji fragment mobilne aplikacije se prikazuje korisniku. Sustavu se pristupa preko dva sučelja. Preko sučelja za dohvat JSON podataka pristupa se FastApi komponenti. Ona poslužuje podatke koji pripadaju *backend* dijelu aplikacije. SQLAlchemy je zadužen za dohvaćanje tablica iz baze preko SQL upita. Pristigli podaci dalje se šalju MVC arhitekturi u obliku DTO (*Data transfer object*). Sučelje za dohvaćanje slika pristupa Media komponenti na poslužitelju u kojoj se nalaze spremljene sve fotografije aplikacije. Aplikacija preko dostupnih sučelja komunicira s poslužiteljem te ovisno o korisničkim akcijama dohvaća nove podatke ili slike.

Slika 4.14: Dijagram komponenti

5. Implementacija i korisničko sučelje

5.1 Korištene tehnologije i alati

Komunikacija unutar tima realizirana je korištenjem aplikacija WhatsApp[1] i Discord[2]. Za izradu UML dijagrama korišten je alat Astah Professional[3]. Kao sustav za upravljanje izvornim kodom upotrebljavali smo Git[4], a udaljeni repozitorij projekta je dostupan na web platformi GitHub[5].

Kao razvojna okruženja korišteni su Android Studio[6] i PyCharm[7]. Android Studio je integrirano razvojno okruženje za Googleov operativni sustav Android, izgrađeno na JetBrainsovom IntelliJ IDEA softveru i dizajnirano posebno za Android razvoj. Dostupan je za preuzimanje na Windows, macOS i Linux operativnim sustavima. PyCharm je integrirano razvojno okruženje koje se koristi za programiranje u Pythonu koji je razvila tvrtka JetBrains. Omogućuje analizu koda, integrirani tester jedinica *(engl. unit testing)*, grafički *debugger* i podržava web razvoj s Djangom. Isto kao i Android Studio, dostupan je na različitim operacijskim sustavima.

Aplikacija je napisana koristeći radni okvir FastAPI[8] i jezik Python[9] za izradu *backenda* te jezik Kotlin[10] za izradu *frontenda*. FastAPI je moderan web okvir za izgradnju RESTful API-ja u Pythonu. Popularan je među programerima zbog svoje jednostavnosti, robusnosti i brzine.

Baza podataka se nalazi na poslužitelju u Renderu[11]. To je objedinjeni oblak za izradu i pokretanje svih aplikacija i web stranica s besplatnim TLS certifikatima, globalnim CDN-om, privatnim mrežama i automatskim deploymentom iz Gita.

[1] https://www.whatsapp.com/
[2] https://discord.com/
[3] https://astah.net/
[4] https://git-scm.com/
[5] https://github.com/
[6] https://developer.android.com/studio
[7] https://www.jetbrains.com/pycharm/
[8] https://fastapi.tiangolo.com/
[9] https://www.python.org/
[10] https://kotlinlang.org/
[11] https://render.com/

5.2 Ispitivanje programskog rješenja

Ispitivanje aplikacije provelo se na strukturiran način pomoću unit testova na *backendu* i ručno na *frontendu* zato što se preporučeni Selenium testovi koriste za web, a ne i za mobilne aplikacije. Aplikaciju smo ispitali po obrascima uporabe kako bi provjerili svu osnovnu funkcionalnost aplikacije, kao i nasumičnim kretanjem po aplikaciji kako bi otkrili moguća nepredviđena ponašanja i ispravili neočekivane greške. Ispitan je cijeli sustav, no zbog jednostavnosti dokumentacije ovdje će biti prikazan samo dio testova.

5.2.1 Ispitivanje komponenti

Proveli smo ispitivanje jedinica (engl. *unit testing*) nad razredima koji implementiraju osnovne funkcionalnosti. U nastavku će biti prikazan izvorni kod i prikaz rezultata izvođenja ispita u razvojnom okruženju.

Za konfiguraciju i provođenje svih testova bile su nam potrebne sljedeće funkcije:

- pytest_configure - stvara testnu bazu na lokalnom poslužitelju prije testiranja
- pytest_unconfigure - *dropa* lokalnu tekstnu bazu nakon završetka testiranja
- pytest fixture test_session - da testovi nebi utjecali jedan na drugog, za svakog stvori izoliranu okolinu
- mock_user_token - dependency override za provjeru tokena na endpointima, token se provjerava zasebno
- user_factory - direktno stvara usera na bazi
- advert_factory - direktno stvara advert na bazi

Izvorni kod navedenih funkcija priložen je u nastavku.

```
def pytest_configure():
    create_database(
        URL.create(
            "postgresql",
            username=settings.POSTGRES_USER,
            password=settings.POSTGRES_PASSWORD,
            host=settings.POSTGRES_HOST,
            port=settings.POSTGRES_PORT,
            database=settings.TEST_DATABASE,
        )
```

```
    )

def pytest_unconfigure():
    drop_database(
        URL.create(
            "postgresql",
            username=settings.POSTGRES_USER,
            password=settings.POSTGRES_PASSWORD,
            host=settings.POSTGRES_HOST,
            port=settings.POSTGRES_PORT,
            database=settings.TEST_DATABASE,
        )
    )

@pytest.fixture(scope="function")
def test_session():
    EngineManager.set_database(settings.TEST_DATABASE)
    engine = EngineManager.get_engine()
    Base.metadata.create_all(engine)

    test_session_mold = sessionmaker(bind=engine, expire_on_commit=False
    )
    test_session = test_session_mold()
    yield test_session

    Base.metadata.drop_all(engine)
    EngineManager.unset_database()

def mock_user_token():
    return 1

def user_factory(user_id: int, session: Session):
    user = (
        UserCustom(
                id=user_id,
                username=('test_username_' + str(user_id)),
                is_shelter=False,
                email='test_email',
                phone_number='test_number'
        )
    )
    user_auth = UserAuth(username=('test_username_' + str(user_id)),
    password='1234')
```

```
    session.add(user)
    session.add(user_auth)
    session.commit()
    return user

def advert_factory(id: int, user_id: int, session: Session):
    pet = Pet(
        id=id,
        name=('test_pet_' + str(id)),
        location_lost=('test_location_' + str(id)),
    )
    advert = Advertisement(
        id=id,
        pet_id=id,
        user_id=user_id,
        category='lost'
    )
    session.add(pet)
    session.add(advert)
    session.commit()

```

U nastavku je naveden dio testova koje smo proveli.

Klasa *TestAuthentication* provodi dva testa. Prvi test ispitao je generiranje i dekodiranje korisničkog tokena. Drugi test ispitao je funkcionalnost prijave i registracije korisnika pri ulasku u aplikaciju. Komponenta je prošla testove i rezultat ispitivanja prikazan je na slici 5.1. Izvorni kod za ova dva testa je sljedeći:

```
class TestAuthentication:

    def test_auth_utils(self, test_session):
        user = user_factory(1, test_session)
        token = generate_token(user)
        assert len(token) == 181

        decoded = validate_token(token)
        assert decoded == user.id

    def test_signup_and_login(self, test_session):
        with TestClient(app) as client:
            result_signup = client.post(
                "/api/authorization/signup",
                json={
```

```
                    "username": "test_shelter",
                    "password": "password",
                    "is_shelter": True,
                    "shelter_name": "Test Shelter",
                    "email": "test@shelter.com",
                    "phone_number": "111"
                }
            )

            assert result_signup.status_code == 201
            assert result_signup.json().get("detail") == "User successfully saved"

            result_login = client.post(
                "/api/authorization/login",
                json={
                    "username": "test_shelter",
                    "password": "password"
                }
            )

            assert result_login.status_code == 200
            assert result_login.json().get("token") is not None
            assert result_login.json().get("is_shelter")
            assert result_login.json().get("current_user_username") == "test_shelter"

```

Slika 5.1: Rezultati testa 1. i testa 2.

Klasa *TestMessaging* provjerava slanje i dohvaćanje poruka o nestalom ljubimcu. Komponenta je prošla test i rezultat ispitivanja prikazan je na slici 5.2. Izvorni kod testa je sljedeći:

```
class TestMessaging:

    def test_add_and_fetch_message(self, test_session):
        app.dependency_overrides[validate_token] = mock_user_token
        user_factory(1, test_session)
        user_factory(2, test_session)
        advert_factory(1, 1, test_session)

        with TestClient(app) as client:
            result = client.post(
                "/api/messages/1/add",
                json={
                    "text": "message_1"
                }
            )
            assert result.status_code == 200

            client.post(
                "/api/messages/1/add",
```

```
                    json={
                        "text": "message_2"
                    }
                )
                assert result.status_code == 200

                result = client.get(
                    "/api/messages/1"
                )
                assert result.status_code == 200
                assert len(result.json()) == 2

```

Slika 5.2: Rezultati testa 3.

Klasa *TestAdvertActions* izvodi tri testa nad oglasima o nestalim ljubimcima. Ispitala je postavljanje, izmjenu i brisanje oglasa. Komponenta aplikacije je prošla sva tri testa i rezultati su prikazani na slici 5.3. Izvorni kod ovih testova je sljedeći:

```
class TestAdvertActions:

    def test_create(self, test_session):
        app.dependency_overrides[validate_token] = mock_user_token
        user_factory(1, test_session)
        with TestClient(app) as client:
```

```
7            result = client.post(
8                "/api/advert/create",
9                json={
10                   "pet_name": "Edgar",
11               }
12           )
13           assert result.status_code == 200
14
15       def test_edit(self, test_session):
16           app.dependency_overrides[validate_token] = mock_user_token
17           user_factory(1, test_session)
18           advert_factory(1, 1, test_session)
19           with TestClient(app) as client:
20               result = client.put(
21                   "/api/advert/1/edit",
22                   json={
23                       "pet_name": "Allan",
24                   }
25               )
26               assert result.status_code == 200
27               assert result.json().get("pet_name") == "Allan"
28
29       def test_delete(self, test_session):
30           app.dependency_overrides[validate_token] = mock_user_token
31           user_factory(1, test_session)
32           advert_factory(1, 1, test_session)
33           with TestClient(app) as client:
34               result = client.delete(
35                   "/api/advert/1/delete"
36               )
37               assert result.status_code == 200
38
```

Slika 5.3: Rezultati testa 4., testa 5. i testa 6.

Klasa *TestHome* provodi dva testa. Prvi je ispitivanje pretraživanja po imenu ljubimca i po korisničkom imenu. Komponenta je prošla ovaj test, a rezultat ispitivanja prikazan je na slici 5.4. Druga stvar koju ova klasa ispituje je pretraživanje po lokaciji. Ta funkcionalnost nije implementirana u našoj aplikaciji, pa zato komponenta pada na ovom testu. Rezultat je prikazan na slikama 5.5 i 5.6. Izvorni kod klase koja provodi ova ispitivanja prikazan je u nastavku.

```
class TestHome:

    def test_filter__by_pet_name_and_username(self, test_session):
        user_factory(1, test_session)
        user_factory(2, test_session)
        advert_factory(1, 1, test_session)
        advert_factory(2, 2, test_session)

        with TestClient(app) as client:
            result = client.get(
                "/api/advert/",
                params={"pet_name": "test_pet_1", "username": "test_username_1"}
            )

            assert result.status_code == 200
```

```
16            assert len(result.json()) == 1
17
18    def test_filter__by_location(self, test_session):
19        user_factory(1, test_session)
20        user_factory(2, test_session)
21        advert_factory(1, 1, test_session)
22        advert_factory(2, 2, test_session)
23
24        with TestClient(app) as client:
25            result = client.get(
26                "/api/advert/",
27                params={"location_lost": "test_location_1"}
28            )
29
30            assert result.status_code == 200
31            assert len(result.json()) == 1
32
```

Slika 5.4: Rezultati testa 7.

Slika 5.5: Rezultati testa 8. - 1. dio

Slika 5.6: Rezultati testa 8. - 2. dio

5.2.2 Ispitivanje sustava

Ispitivanje sustava provedeno je ručno za sve obrasce uporabe. U nastavku su prikazani rezultati ispitivanja za UC2 (s UC15), UC3 (uključujući UC8), UC6 i UC9.

Ispitni slučaj 1: Prijava korisnika u sustav

Ulaz:

1. Otvaranje stranice za prijavu.
2. Unos korisničkog imena i lozinke.
3. Pritisak na gumb za prijavu.

Očekivani rezultat:

1. Prikazuje se lista oglasa
2. Odabirom ikone korisnika u *toolbar-u* mogu se pregledati podaci o korisniku

Rezultat: Očekivanja su zadovoljena. Aplikacija je prošla test.

Slika 5.7: Prijava u sustav

Slika 5.8: Prikaz profila

Ispitni slučaj 2: Postavljanje oglasa

Ulaz:

1. Odabir opcije "Dodaj oglas".

2. Otvaranje forme za unos podataka.
3. Unos podataka o nestalom ljubimcu.
4. Pritisak na gumb za objavu oglasa.

Očekivani rezultat:

(a) Prikazuje se detaljni prikaz stvorenog oglasa.
(b) Nudi se mogućnost pisanja komentara.

Rezultat: Očekivanja su zadovoljena. Aplikacija je prošla test.

Slika 5.9: Stvaranje novog oglasa

Slika 5.10: Prikaz stvorenog oglasa

Ispitni slučaj 3: Prikaz liste i pregled oglasa
Ulaz:

1. Otvaranje aplikacije (već prijavljen korisnik ili odabir ulaska kao gost).

2. Odabire se oglas za prikaz.

Očekivani rezultat:

1. Prikazuje se lista svih oglasa.
2. Prikazuje se detaljni prikaz odabranog oglasa.

Rezultat: Očekivanja su zadovoljena. Aplikacija je prošla test.

Slika 5.11: Lista oglasa

Slika 5.12: Detaljni prikaz oglasa

Ispitni slučaj 4: Brisanje oglasa
Ulaz:

1. Pregled vlastitih oglasa.
2. Odabir opcije za brisanje.
3. Potvrda brisanja.

Očekivani rezultat:

(a) Ispis poruke o uspješnom brisanju.
(b) Prikaz preostalih vlastitih oglasa ili poruka da ih više nema.

Rezultat: Očekivanja su zadovoljena. Aplikacija je prošla test.

Slika 5.13: Odabir brisanja

Slika 5.14: Potvrda brisanja

Slika 5.15: Prikaz preostalih oglasa

5.3 Dijagram razmještaja

UML-dijagrami razmještaja prikazuju fizičku arhitekturu programskog sustava, prikazujući razmještaj programskih artefakata na sklopovskim čvorovima ili virtualnim okruženjima. Arhitektura sustava prepoznaje dvije različite funkcionalnosti - klijenta i poslužitelja. Korisnici pristupaju aplikaciji putem svojih mobilnih uređaja, dok se web poslužitelj i poslužitelj baze podataka nalaze na poslužiteljskom računalu. Komunikacija između korisnika aplikacije i poslužitelja odvija se putem HTTP veze.

Slika 5.16: Dijagram razmještaja

5.4 Upute za puštanje u pogon

5.4.1 Lokalni deploy aplikacije

Baza podataka

Prije pokretanja backend servisa potrebno je lokalno stvoriti bazu podataka. Za to je potreban PostgreSQL poslužitelj baza podataka. Preporučamo dodatno korištenje grafičkih alata kao što su pgAdmin ili DBeaver, ali dovoljan je i samo PostgreSQL driver i konfiguracija iz komandne linije. Upute pratiti na službenim stranicama alata, ovisno o OS-u računala na kojem se pokreće.

Nakon instalacije potrebno je stvoriti bazu podataka. Server očekuje da će se baza zvati „lost_pets“, ali navedeno se može „pregaziti“ u settings.py fileu u root folderu backenda, u varijabli DEFAULT_DATABASE. Ipak, preporučujemo da bazu nazovete „lost_pets“.

Ako se koristi pgAdmin, baza se može stvoriti odabirom „Servers“ → „PostgreSQL“ na lijevoj strani ekrana, a zatim desni klik na „PostgreSQL“ pa „Create“ → „Database“. Zatim se otvara prozor u kojem se unosi ime baze (5.17) i nakon toga „Save“.

Slika 5.17: Unos imena baze u pgAdminu

User i password za pristup bazi su konfigurabilni, i preporučujemo mijenjati ih, ako je potrebno, u .env fileu, u varijablama POSTGRES_USER i POSTGRES_PASSWORD (Prilagoditi svojim PostgreSQL postavkama). Kako se postgres server defaultno nalazi na portu 5432, a bazu konfiguriramo lokalno, preporučujemo ostaviti POSTGRES_HOST="localhost" te POSTGRES_PORT=5432, iako se navedene postavke mogu i mijenjati.

Ove se postavke kod deploya na remote poslužitelj također "pregaze" pomoću varijabli okruženja.

Kada je sve od navedenog postavljeno, baza je spremna za početak rada servera. Tablice stvara repeatable skripta run_migrations, a početne podatke ubacuje repeatable skripta init_populate, koje se obje pokreću automatski kod pokretanja servisa.

Pokretanje servisa

Najlakši način za deploy servisa jest buildati Docker image iz Dockerfilea, a zatim pokrenuti Docker container iz spomenutog imagea. Ovo osigurava da će server imati sve što mu je potrebno i pokretati se u izoliranom okruženju. Upute za to pratiti na službenim Docker stranicama.

Ovaj će se pristup skoro uvijek koristiti, i automatiziran je, kod deploya na remote server, a za detaljnije upute (poput postavljanja varijabli okruženja objašnjenih prije) potrebno je slijediti upute alata koji pruža usluge poslužitelja.

Ipak, postoji relativno jednostavan način za pokretanje bez buildanja Docker image-a, a to je u virtualnom okruženju, uz lokalni python interpreter. Ukoliko ste na Linuxu, dovoljno će biti izvršiti sljedeće naredbe iz root foldera backenda (folder backend-service):

```
sudo apt-get update
sudo apt-get install python3
sudo apt-get install python3-pip
sudo apt install python3-venv
python3 -m venv ./venv
source venv/bin/activate
pip3 install -r requirements.txt
uvicorn service.main:app
```

Na windows OS-u pokretanje je nešto drugačije.

Potrebno je imati instaliran python 3.10 (radi i sa 3.11) i odgovarajući pip installer.

Nakon toga pozicionirati se u root folder backenda (backend-service) u command promptu i pokrenuti slijedno naredbe:

```
1 python -m venv ./venv
2 venv\Scripts\activate
3 pip install -r requirements.txt
4 uvicorn service.main:app
```

U oba slučaja servis bi se sada trebao nalaziti na http://localhost:8000, a Swagger dokumentacija (popis endpointa i mogućnost slanja direktnog requesta) dostupna je na http://localhost:8000/docs .

5.4.2 Postupak izgradnje i objave aplikacije

Priprema za objavu

a) Odabir ikone aplikacije

b) Konfiguracija aplikacije

Konfiguracija aplikacije uključuje: isključivanje debugginga

```
android {
  ...
  buildTypes {
    release {
      isDebuggable = false
      ...
    }
    debug {
      isDebuggable = true
      ...
    }
  }
  ...
}
```

Slika 5.18: Isključivanje logiranja, pregleda android manifesta te build opcija, odabir odgovarajučeg ID-a aplikacije.

Postavljanje verzije

```
defaultConfig { this: ApplicationDefaultConfig
    applicationId = "progi.imateacup.nestaliljubimci"
    minSdk = 24
    targetSdk = 34
    versionCode = 1
    versionName = "1.0"
```

Slika 5.19: Postavljanje verzije

Potpis aplikacije i release build aplikacije

Potrebno je generirati upload key i keystore, što je moguće korištenjem alata Android Studio.

Nakon što je keystore generiran, moguće je izgraditi release verziju aplikacije u .aab ili .apk formatu. U Android Studiu odabrati `Build -> Select Build Variant` i odabrati "release". Zatim `Build -> Generate Signed Bundle(s) / Apk`, odabrati, u našem slučaju, APK.

Slika 5.20: Odabir APK

Stisnuti `Next`, odabrati `key store path` koji vodi na ranije generirani keystore i unjeti `keystore`, `key password` i `alias`.

Slika 5.21: Unos za keystore, key password i allias

Stisnuti `Next`, odabrati `build variant`, odredišni folder i kliknuti `Create`. Potpisani APK će se nalaziti u specificiranom folderu.

Slika 5.22: Odabir mape za APK i izrada

Objava aplikacije na odabranom appstore-u

Objava aplikacije znatno se razlikuje na različitim trgovinama aplikacija, a postupak objave često je rigorozan te se nerijetko plaća. Zbog takve kompleksnosti i spomenutih razlika u procesu objave, nećemo ulaziti u detalje već ćemo ukratko opisati postupak objave na Amazon Appstoreu.

Za početak, potrebno je napraviti račun na Amazon Appstoreu te se navigirati na sljedeću stranicu: `https://developer.amazon.com/docs/app-submission/submitting-apps-to-amazon-appstore.html`.

Prateći upute na priloženoj stranici, potrebno se ulogirati u Amazon Developer konzolu. Zatim kliknuti na *dashboard* te nakon toga odabrati *Add a New App* iz padajućeg izbornika i odabrati *Android*. Ponovno odabrati *Add a New App* te *Android*. Zatim je potrebno pratiti formu te ispuniti sva polja koja Amazon zahtijeva te generirati zahtjev za objavu. Nakon što se to uspješno obavi, potrebno je čekati da Amazon Appstore odobri upload aplikacije.

6. Zaključak i budući rad

Naš zadatak bio je razviti aplikaciju za pronalaženje nestalih ljubimaca s mogućnošću komunikacije oko potrage, pregledom aktivnih i neaktivnih oglasa te mogućnošću uređivanja oglasa i funkcionalnost korisnika registriranih kao sklonište. Tijekom 15 tjedana rada u timu, ostvarili smo postavljeni cilj kroz dvije ključne faze.

U prvoj fazi projekta, okupili smo tim, odabrali projektni zadatak i posvetili se intenzivnom dokumentiranju zahtjeva. Također smo i jednoglasno odabrali vođu tima koji je obavljao razgovor s asistentom i profesorom predmeta. Odlučili smo se za razvoj android aplikacije jer nam se činila kao najbolje rješenje za zadani problem. Tim smo podijelili u backend, frontend i tim za dokumentaciju. Pisanje dokumentacije i izrada raznih dijagrama pružili su smjernice podtimovima za razvoj backenda i frontenda. Naravno da smo imali nesuglasica, ali njih bi brzo riješili sastancima uživo koje smo imali često.

U drugoj fazi je svaki član tima znao na čemu treba raditi, pa je bilo više samostalnog rada nego u prvoj fazi. Uz pomoć ostalih članova tima i raznih materijala smo zajedno prelazili prepreke na koje bi naišli. Komunikacija je bila ključna jer smo u ovoj fazi mnogo ovisili o ostalim članovima tima i njihovom radu. Na kraju druge faze smo radili razne testove kako bi osigurali da aplikacija radi kako je i zamišljena.

Komunikacija među članovima odvijala se putem Whatsappa i Discorda kojeg je frontend najviše koristio za videopozive. Sudjelovanje u projektu bilo je vrijedno iskustvo za članove tima, naglašavajući važnost dobre organizacije i suradnje. Unutar tima smo imali studente s različitim prijašnjim iskustvima na sličnim projektima te su oni s više znanja uvijek bili voljni pomoći. Unatoč prostoru za usavršavanje, postignuto je značajno napredovanje u razumijevanju i implementaciji aplikacije.

Popis literature

1. Programsko inženjerstvo, FER ZEMRIS, http://www.fer.hr/predmet/proinz
2. MissingPetApp, https://missingpet.app/
3. GeeksforGeeks, MVC (Model View Controller) Architecture Pattern in Android, https://www.geeksforgeeks.org/mvc-model-view-controller-architecture-pattern-in-android-with-example/?ref=lbp
4. Mapbox, https://www.mapbox.com/

Indeks slika i dijagrama

2.1 Primjer sličnog rješenja 7
2.2 Prikaz više informacija o ljubimcu kod sličnog rješenja 9
2.3 Komunikacija kod sličnog rješenja 9

3.1 Dijagram obrasca uporabe, funkcionalnost neregistriranog i registriranog korisnika 19
3.2 Dijagram obrasca uporabe, funkcionalnost skloništa 20
3.3 Sekvencijski dijagram - UC4 21
3.4 Sekvencijski dijagram - UC6 22
3.5 Sekvencijski dijagram - UC7 23
3.6 Sekvencijski dijagram - UC10 24

4.1 Arhitektura sustava 26
4.2 MVC koncept 27
4.3 Relacijski dijagram baze podataka 31
4.4 E-R dijagram baze podataka 32
4.5 Dijagram razreda - 1. dio View: Pets i Shelters klase 34
4.6 Dijagram razreda - 2. dio View: Login, Register i Map klase 35
4.7 Dijagram razreda - 3. dio View: Advert i Message klase 36
4.8 Dijagram razreda - dio Models 37
4.9 Dijagram razreda - dio Data transfer objects 38
4.10 Dijagram razreda - dio Repository 39
4.11 Dijagram razreda - dio Controllers 40
4.12 Dijagram stanja 41
4.13 Dijagram aktivnosti - Postavljanje oglasa 43
4.14 Dijagram komponenti 45

5.1 Rezultati testa 1. i testa 2. 51
5.2 Rezultati testa 3. 52
5.3 Rezultati testa 4., testa 5. i testa 6. 54
5.4 Rezultati testa 7. 55

5.5 Rezultati testa 8. - 1. dio 56
5.6 Rezultati testa 8. - 2. dio 56
5.7 Prijava u sustav 57
5.8 Prikaz profila 57
5.9 Stvaranje novog oglasa 58
5.10 Prikaz stvorenog oglasa 58
5.11 Lista oglasa 59
5.12 Detaljni prikaz oglasa 59
5.13 Odabir brisanja 60
5.14 Potvrda brisanja 60
5.15 Prikaz preostalih oglasa 61
5.16 Dijagram razmještaja 62
5.17 Unos imena baze u pgAdminu 63
5.18 Isključivanje logiranja, pregleda android manifesta te build opcija, odabir odgovarajučeg ID-a aplikacije. 65
5.19 Postavljanje verzije 66
5.20 Odabir APK 66
5.21 Unos za keystore, key password i allias 67
5.22 Odabir mape za APK i izrada 67

6.1 Prikaz aktivnosti na repozitoriju 79

Dodatak: Prikaz aktivnosti grupe

Dnevnik sastajanja

1. sastanak
   - Datum: 16. listopada 2023.
   - Prisustvovali: N. Ivić, M. Krajinović, M. Matulić, B. Mikan, A. Radoš, M. Žagar, I. Žinić
   - Teme sastanka:
     - sastanak s asistentom i demonstratorom
     - raspodijela zadataka
     - dogovor o korištenim tehnologijama
2. sastanak
   - Datum: 19. listopada 2023.
   - Prisustvovali: B. Mikan, A. Radoš, M. Žagar
   - Teme sastanka:
     - uvod u frontend tehnologije
     - uvod u proces razvoja frontenda
3. sastanak
   - Datum: 21. listopada 2023.
   - Prisustvovali: N. Ivić, M. Krajinović
   - Teme sastanka:
     - dogovorena struktura backend dijela aplikacije
     - objašnjen fastapi i sqlalchemy
     - detaljnije dogovorena podjela posla u podtimu
4. sastanak
   - Datum: 23. listopada 2023.
   - Prisustvovali: N. Ivić, M. Krajinović, M. Matulić, B. Mikan, A. Radoš, M. Žagar, I. Žinić
   - Teme sastanka:
     - dogovaranje detalja oko implementacije

    - raščišćavanje osnovnih dilema funkcionalnosti

5. sastanak
    - Datum: 25. listopada 2023.
    - Prisustvovali: M. Matulić, I. Žinić
    - Teme sastanka:
        - detaljnija razrada opisa zadatka
        - raspisivanje funkcionalnih zahtjeva i opis obrazaca uporabe
6. sastanak
    - Datum: 28. listopada 2023.
    - Prisustvovali: B. Mikan, A. Radoš, M. Žagar
    - Teme sastanka:
        - diskusija oko dizajniranja sučelja u alatu figma
        - uvod u proces razvoja frontenda
        - odabir color palete
        - generalni apstraktni plan arhitekture aplikacije
        - branch naming konvencija
        - podijeljeni prvi jira taskovi
7. sastanak
    - Datum: 6. studenoga 2023.
    - Prisustvovali: N. Ivić, M. Krajinović, M. Matulić, B. Mikan, A. Radoš, M. Žagar, I. Žinić
    - Teme sastanka:
        - rješavanje nesuglasica
        - daljnja raspodjela posla
        - dogovor oko budućih uvjeta zadatka
        - uvod u tehnički proces puštanja aplikacije u pogon
8. sastanak
    - Datum: 13. studenoga 2023.
    - Prisustvovali: N. Ivić, M. Krajinović, M. Matulić, B. Mikan, A. Radoš, M. Žagar, I. Žinić
    - Teme sastanka:
        - organizacija puštanja aplikacije u pogon
        - testiranje aplikacije
        - provjera detalja aplikacije i dokumentacije
9. sastanak

- Datum: 4. prosinca 2023.
- Prisustvovali: N. Ivić, M. Krajinović, M. Matulić, B. Mikan, A. Radoš, I. Žinić
- Teme sastanka:
  - daljnja raspodjela posla
  - dogovori između frontenda i backenda

10. sastanak
    - Datum: 11. prosinca 2023.
    - Prisustvovali: N. Ivić, M. Krajinović, M. Matulić, B. Mikan, A. Radoš, M. Žagar
    - Teme sastanka:
      - pregled grešaka prve revizije
      - razgovor s profesorom
      - planiranje budućeg rada
11. sastanak
    - Datum: 18. prosinca 2023.
    - Prisustvovali: N. Ivić, M. Krajinović, M. Matulić, B. Mikan, A. Radoš, M. Žagar, I. Žinić
    - Teme sastanka:
      - pregled do sada obavljenog dijela
      - daljnja raspodjela posla
      - planiranje budućeg rada
12. sastanak
    - Datum: 8. siječnja 2024.
    - Prisustvovali: N. Ivić, M. Krajinović, M. Matulić, B. Mikan, M. Žagar
    - Teme sastanka:
      - pregled do sada obavljenog posla
      - dogovor oko testiranja i deploymenta
      - rješavanje pitanja u vezi dokumentacije
13. sastanak
    - Datum: 15. siječnja 2024.
    - Prisustvovali: N. Ivić, M. Krajinović, M. Matulić, B. Mikan, A. Radoš, M. Žagar, I. Žinić
    - Teme sastanka:
      - pregled do sada obavljenog posla

- rješavanje nesuglasica
- planiranje zadnjeg tjedna rada
- dogovori oko prezentacije

Tablica aktivnosti

Napomena: Doprinosi u tablici su navedeni u satima.

	Nora Ivić	**Mirta Krajinović**	**Marta Matulić**	**Bruno Mikan**	**Antonio Radoš**	**Marko Žagar**	**Ivan Žinić**
Upravljanje projektom	14						
Opis projektnog zadatka			7				3
Funkcionalni zahtjevi			6				9
Opis pojedinih obrazaca			2				6
Dijagram obrazaca							9
Sekvencijski dijagrami			6				1
Opis ostalih zahtjeva			2				
Arhitektura i dizajn sustava			6				1
Opći prioriteti i svrha sustava	5	5	9	5	5	5	9
Dizajn baze podataka	3	12	2				
Dokumentacija baze podataka		2	8				8
Dijagram razreda	1	1	16	2		2	16
Dijagram stanja			1				9
Dijagram aktivnosti			6				
Dijagram komponenti			8				1
Korištene tehnologije i alati	10	9	13	9	12	12	8
Ispitivanje programskog rješenja	15	15	12	9	10	8	2
Dijagram razmještaja							7
Upute za puštanje u pogon	5			2			6

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

	Nora Ivić	**Mirta Krajinović**	**Marta Matulić**	**Bruno Mikan**	**Antonio Radoš**	**Marko Žagar**	**Ivan Žinić**
Plan rada	14	14	14	19	19	19	14
Dnevnik sastajanja			2				5
Zaključak i budući rad							5
Popis literature			1				
Definiranje endpointa	6						
Izrada baze podataka		17					
Spajanje s bazom	3	3					
Backend	46	38					
Deployment	10	10					
Dizajn izgleda aplikacije				13	10	14	
Implementacija izgleda				58	41	53	
Povezivanje s backendom				8	6	8	

Dijagrami pregleda promjena

Slika 6.1: Prikaz aktivnosti na repozitoriju

Napomena: Mirta Krajinović je imala namješten krivi email pa joj nije brojalo commitove