

Programsko inženjerstvo

Ak. god. 2023./2024.

WildTrack

Dokumentacija, Rev. 2

Grupa: *Zoolanders*

Voditelj: *Filip Ljubotina*

Datum predaje: *19.01.2024.*

Nastavnik: *Hrvoje Nuić*

# Sadržaj

<b>1 Dnevnik promjena dokumentacije</b>	<b>3</b>
<b>2 Opis projektnog zadatka</b>	<b>6</b>
<b>3 Specifikacija programske potpore</b>	<b>13</b>
3.1 Funkcionalni zahtjevi . . . . .	13
3.1.1 Obrasci uporabe . . . . .	15
3.1.2 Sekvencijski dijagrami . . . . .	28
3.2 Ostali zahtjevi . . . . .	34
<b>4 Arhitektura i dizajn sustava</b>	<b>35</b>
4.1 Baza podataka . . . . .	36
4.1.1 Opis tablica . . . . .	36
4.1.2 Dijagram baze podataka . . . . .	42
4.2 Dijagram razreda . . . . .	43
4.3 Dijagram stanja . . . . .	46
4.4 Dijagram aktivnosti . . . . .	47
4.5 Dijagram komponenti . . . . .	49
<b>5 Implementacija i korisničko sučelje</b>	<b>51</b>
5.1 Korištene tehnologije i alati . . . . .	51
5.2 Ispitivanje programskog rješenja . . . . .	52
5.3 Ispitivanje komponenti . . . . .	52
5.3.1 Ispitivanje funkcije checkActionDto: . . . . .	52
5.3.2 Ispitivanje funkcije checkAnimalDto: . . . . .	54
5.3.3 Ispitivanje funkcije checkAvailableSearcherDto: . . . . .	56
5.3.4 Ispitivanje funkcije checkRequestDto: . . . . .	58
5.3.5 Ispitivanje funkcije checkTaskDto: . . . . .	60
5.3.6 Ispitivanje funkcije emailRegexCheck: . . . . .	62
5.3.7 Ispitivanje funkcije validateCoordinates: . . . . .	63
5.3.8 Ispitivanje funkcije putRemoveFromAction: . . . . .	65

5.4 Ispitivanje sustava . . . . .	67
5.4.1 Ispitivanje Prijave Korisnika: . . . . .	67
5.4.2 Ispitivanje Registracije: . . . . .	71
5.5 Dijagram razmještaja . . . . .	77
5.6 Upute za puštanje u pogon . . . . .	78
<b>6 Zaključak i budući rad</b>	<b>81</b>
<b>Popis literature</b>	<b>82</b>
<b>Indeks slika i dijagrama</b>	<b>84</b>
<b>Dodatak: Prikaz aktivnosti grupe</b>	<b>85</b>

# 1. Dnevnik promjena dokumentacije

*Kontinuirano osvježavanje*

Rev.	Opis promjene/dodatka	Autori	Datum
0.1	Napravljen predložak.	Filip Ljubotina	29.10.2023.
0.2	Opisi obrazaca uporabe	Marko Pavić	29.10.2023.
0.3	Napravljeni dijagrami obrazaca uporabe.	Filip Ljubotina	29.10.2023.
0.3.1	Dorada dijagrama obrazaca uporabe.	Filip Ljubotina	17.11.2023.
0.4	Opis projektnog zadatka	Lara Ćorić	01.11.2023.
0.5	Dorada dosadašnjeg dijela dokumentacije i sekvencijski dijagrami	Ana Vuksanović	03.11.2023.
0.6	Ostali zahtjevi	Katarina Klarić	04.11.2023.
0.7	Opis tablica i dijagram baze podataka	Mihael Breznički-Herceg	05.11.2023.
0.7.1	Prva dorada opisa tablica i dijagrama baze podataka	Mihael Breznički-Herceg	13.11.2023
0.7.2	Druga dorada opisa tablica i dijagrama baze podataka	Mihael Breznički-Herceg	16.11.2023
0.8	Napravljen opis arhitekture	Marko Pavić	07.11.2023.
0.8.1	Napravljeni dijagrami razreda za generičku funkcionalnost	Katarina Klarić	14.11.2023.

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

<b>Rev.</b>	<b>Opis promjene/dodataka</b>	<b>Autori</b>	<b>Datum</b>
0.8.2	Napravljeni dijagrami razreda za ostatak funkcionalnosti	Noa Milin, Filip Ljubotina	17.11.2023.
<b>1.0</b>	Priprema dokumentacije za predaju za 1. ciklus	Filip Ljubotina	17.11.2023.
1.1	Ispravak grešaka i dodavanje detaljnijeg opisa za bazu podataka	Mihael Breznički-Herceg	15.01.2024.
1.1.1	Dovršavanje opisa baze podataka	Filip Ljubotina	18.01.2024.
1.1.2	Dorada dijagrama razreda	Noa Milin	1.01.2024.
1.2	Dodani dijagram stanja i dijagram aktivnosti	Katarina Klarić	16.01.2024.
1.2.1	Ispravljeni dijagram stanja i dijagram aktivnosti	Katarina Klarić	18.01.2024.
1.3	Dodane korištene tehnologije i alati	Marko Pavić	16.01.2024.
1.4.1	Dodano ispitivanje sustava i dio ispitivanja komponenti	Filip Ljubotina	17.01.2024.
1.4.2	Dodan ostatak ispitivanja komponenti	Marko Pavić	18.01.2024.
1.5	Ispravak grešaka prve predaje u poglavljju Specifikacije programske potpore	Ana Vuksanović	18.01.2024.
1.6.1	Dodane upute za puštaje u pogon.	Filip Ljubotina	18.01.2024.
1.6.2	Nadodane upute za puštanje u pogon	Marko Pavić	18.01.2024.
1.7	Dodan Dijagram komponenti	Mihael Breznički-Herceg	18.01.2024.
1.8	Dodan dijagram razmještaja	Marko Pavić	18.01.2024.

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

<b>Rev.</b>	<b>Opis promjene/dodataka</b>	<b>Autori</b>	<b>Datum</b>
1.9	Dodan zaključak i ispravak grešaka opisa projekta	Lara Čorić	18.01.2024.
2.0	Priprema dokumentacije za predaju	Filip Ljubotina	19.01.2024.

## 2. Opis projektnog zadatka

Cilj projekta jest razviti web aplikaciju za olakšavanje koordinacije i praćenja životinja u divljini. Aplikacija omogućuje korisnicima da se prijavljuju i time sudjeluju u različitim akcijama.

Pri učitavanju web aplikacije, neregistriranom korisniku omogućena je prijava u sustav s postojećim računom (potrebno je upisati korisničko ime ili email adresu i lozinku) ili kreiranje računa. Za kreiranje novog računa potrebni su sljedeći podaci:

- ime
- prezime
- fotografija
- email adresa
- korisničko ime
- lozinka

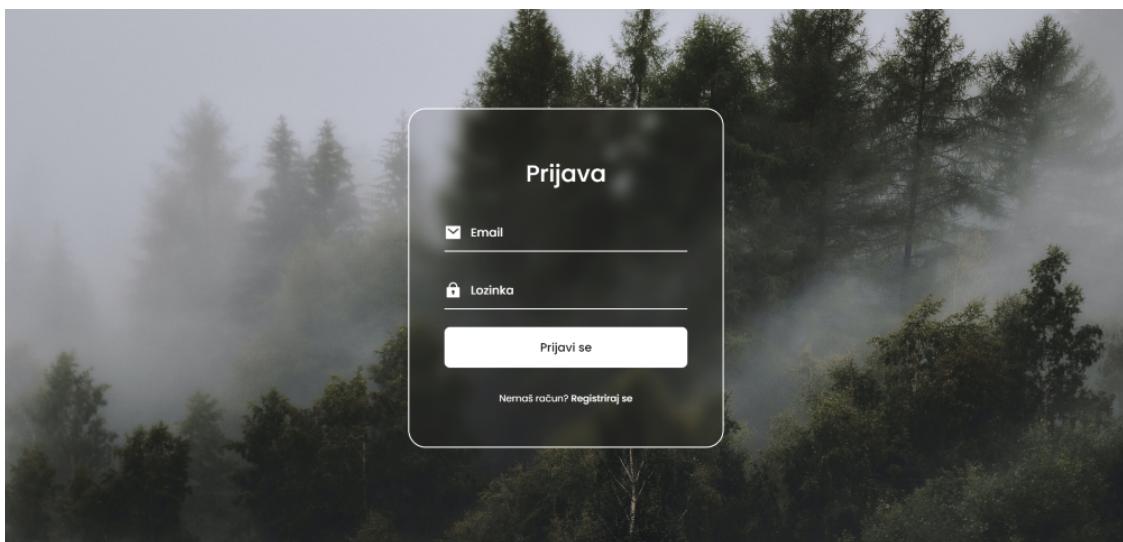


Figure 2.1: Primjer login screena

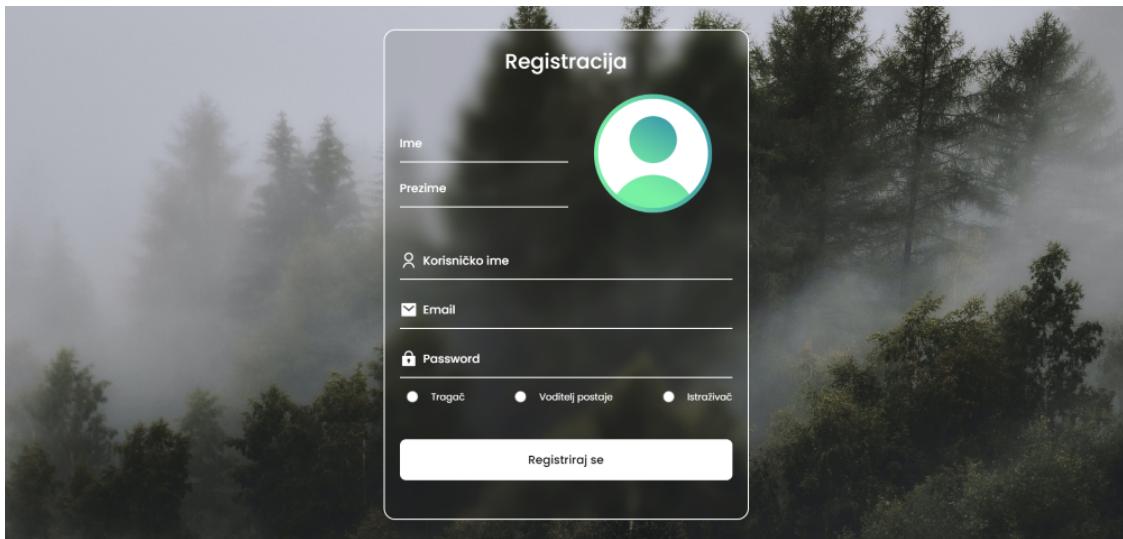


Figure 2.2: Primjer registracije

Pri registraciji korisnik također mora odabrati jednu od navedenih uloga:

- voditelj postaje
- istražitelj
- tragač

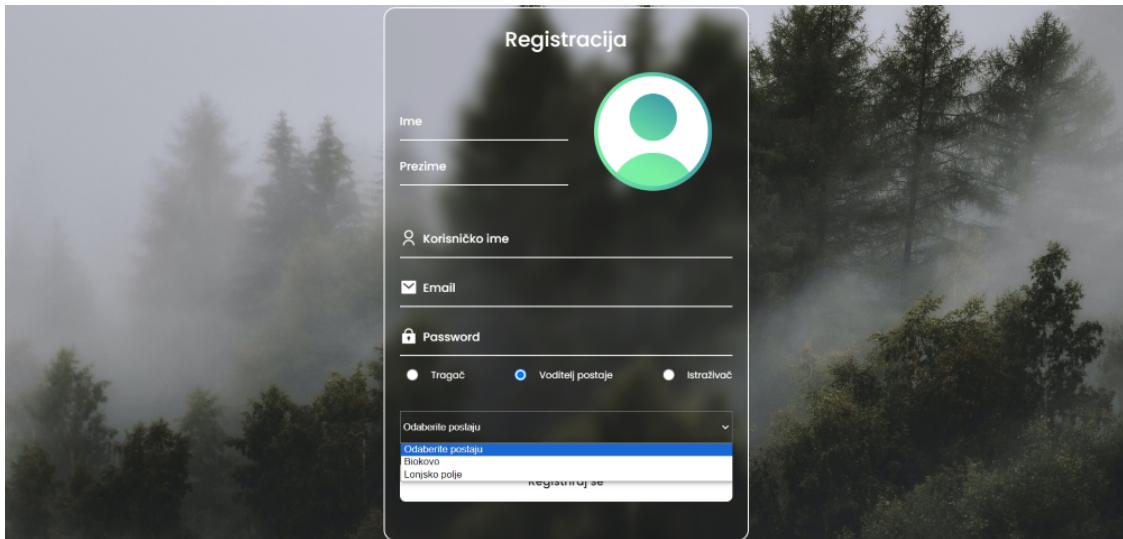


Figure 2.3: Primjer biranja uloge voditelja i unos željene postaje

U slučaju odabira uloge voditelja postaje, korisnik dodatno mora odabrati za koju postaju se želi registrirati. Registracija korisnika potvrđuje se mailom, a u slučaju registracije za ulogu *istraživača* ili *voditelja postaje* potrebna je i dodatna

potvrda od strane administratora. Registrirani korisnik može pregledati svoje osobne podatke.

**Voditelj postaje** ima pregled svih članova postaje. Omogućeno mu je dodavanje novih tragača u postaju i uklanjanje dosadašnjih. Voditelj postaje je ujedno zadužen i za definiranje na koji su način tragači osposobljeni izvoditi pretraživanje. Mogući načini izvođenja pretrage su:

- pješke
- dronom
- automobilom
- cross motorom
- brodom
- helikopterom

Svaka metoda pruža različitu vidljivost i područje pokrivanja. Na primjer, zračno pretraživanje će obuhvatiti veće područje, ali neće pružiti toliko detalja kao što bi se dobilo pješačenjem.

Još jedna bitna zadaća voditelja postaje je da alocira svoje tragače prema pristiglim zahtjevima od strane istražitelja. Voditelj šalje tragača na sudjelovanje u navedenoj akciji, a smije ga maknuti s akcije i ponovno učiniti raspoloživim isključivo ako je tragač završio sa svojim zadatkom u toj akciji.

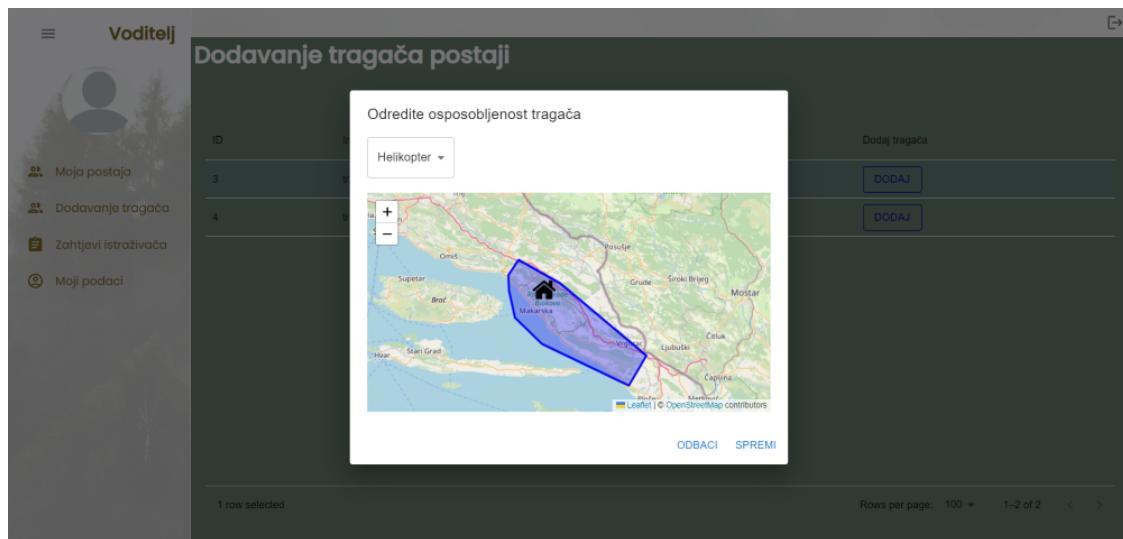


Figure 2.4: Primjer biranja osposobljenja tragača (1)

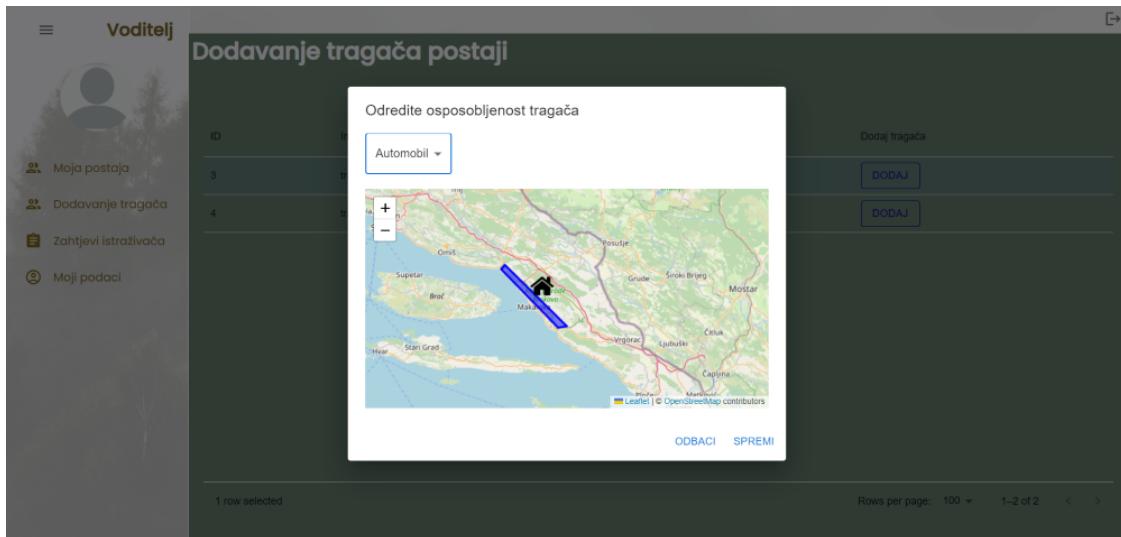


Figure 2.5: Primjer biranja osposobljenja tragača (2)

**Istraživač** je osoba koja ne pripada niti jednoj postaji, a njegova uloga je organizacija akcija pretraživanja i praćenja s detaljima o određenim vrstama, jedinkama ili staništima za proučavanje. Istraživač može stvoriti novu akciju sa svim potrebnim detaljima te poslati zahtjeve za tragačima s opisom o potrebnim kvalifikacijama voditeljima različitih postaje. On ima pregled svih akcija koje je pokrenuo, kao i tragača koji sudjeluju u tim akcijama. Istraživač preko karte tragačima pojedinačno zadaje zadatke. Zadaci mogu tražiti prolazak određenom rutom i dolazak do određene lokacije te postavljanje kamere ili uređaja za praćenje. Svaki zadatak može imati i dodatan komentar od istraživača. Istraživač je taj koji po završetku zadatka označava da je tragač s njim završio.

Glavni alat dostupan istraživaču je interaktivna karta. Njome se istraživaču prikazuju informacije o pozicijama životinja, tragača i postaja, a istraživač može izabrati da se za njenu izradu koristi neka od idućih informacija:

- povijesne pozicije svih praćenih životinja, filtrirano po vrsti ili pojedinačno po jedinkama
- trenutne pozicije praćenih životinja
- povijesne pozicije svih tragača na nekoj akciji, filtrirano po tipu prijevoza ili pojedinačno po tragaču
- trenutne pozicije tragača aktivnih na akciji

Informacije o povijesnim pozicijama se prikazuju preko toplinskih karata (engl. heatmap). Toplinske karte tragača su vizualizacija staza kojima su tragači putovali

i načina kojim su se kretali, a služe kako bi istraživač mogao analizirati obrasce kretanja životinja i omiljena staništa.

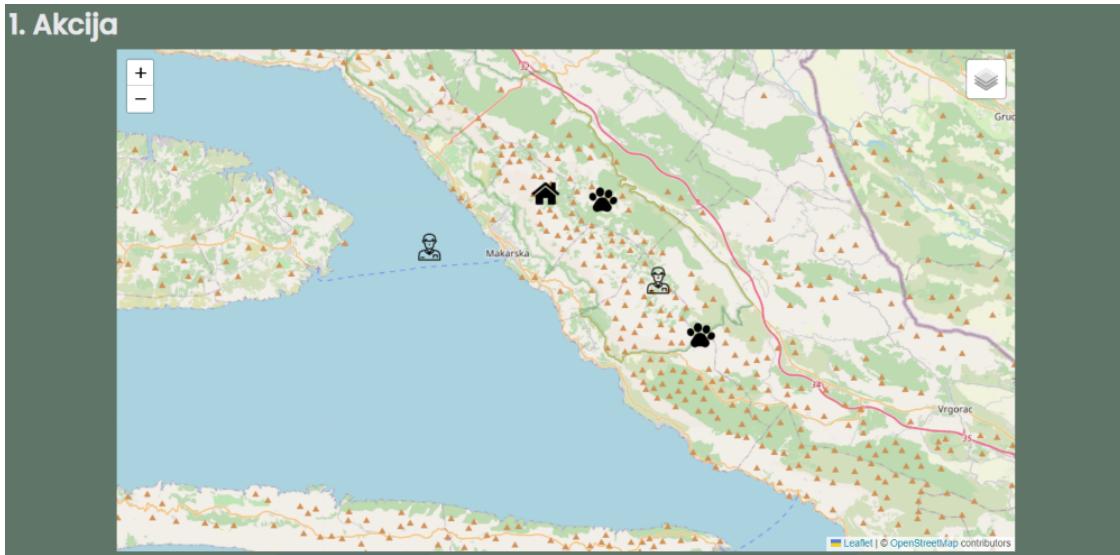


Figure 2.6: Primjer kartografskog prikaza trenutnih lokacija tragača i životinja

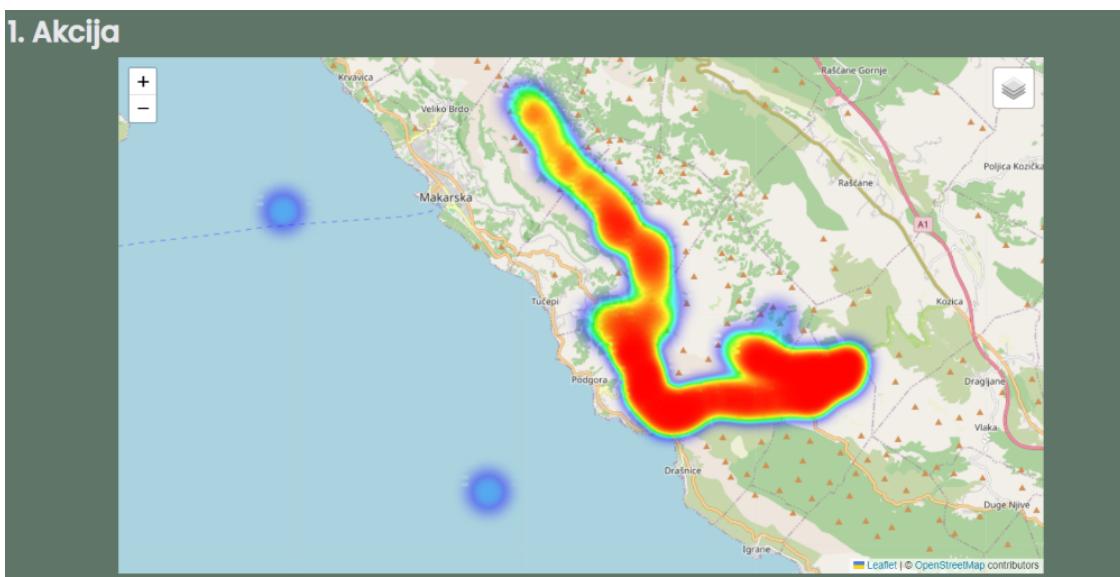


Figure 2.7: Primjer kartografskog prikaza povijesnog kretanja tragača i životinja

**Tragač** će nakon same kreacije korisničkog računa biti slobodan i ostati takav dok ga neki voditelj ne doda u svoju postaju. Time mu se otvara mogućnost sudjelovanja u akcijama. Za vrijeme neke akcije, tragaču se na karti prikazuju zadaci koje treba obaviti, trenutna pozicija ostalih tragača aktivnih na istoj akciji, te trenutna pozicija praćenih životinja. Praćene životinje na sebi imaju gps uređaj koji aplikaciji

odašilje svoju poziciju. O praćenim životinjama se zapisuju povijesni podaci gdje se nalazila, naziv vrste, slika i opis. Tragač može praćenoj životinji tijekom akcije ostaviti komentar. Također, tragač i istraživač mogu na karti ostaviti komentar za ostale sudionike u akciji.

**Administrator** sustava ima najveće ovlasti. On ima pregled svih poslanih zahtjeva za registraciju u ulozi istraživača ili voditelja postaje koje on mora potvrditi ili odbiti. Administrator ima pristup bazi s popisom svih registriranih korisnika i njihovih osobnih podataka te ih može mijenjati. On ujedno može i mijenjati ulogu dodijeljenu korisniku.

Iako naša aplikacija ima mnogo specifičnih značajki usmjerenih praćenju životinja te ostalih korisnika, možemo pronaći slične značajke i u nekim drugim aplikacijama koje se bave praćenjem i kontekstu prirode i istraživanja. Primjeri uključuju aplikacije kao što su:

- **Wildme**

Wildme je organizacija koja razvija tehnologiju prepoznavanja uzoraka koja omogućuje automatsko prepoznavanje pojedinaca u divljini putem fotografija i drugih podataka. Ima seriju online platformi unutar **Wildbook** projekta usmjerena na analizu i praćenje životinja. Time omogućuje zoologima i istraživačima praćenje lakše praćenje jedinki.

- **iNaturalist**

iNaturalist je stranica koja omogućuje korisnicima dijeljenje svojih opažanja divljih biljka i životinja. Koristi se za identifikaciju vrsta te obilježavanje raznolikosti vrsta u prirodi.

- **Movebank**

Movebank je stranica za praćenje migracija životinja. Koristi se za pohranu i analiziranje podataka o kretanju životinja označene GPS uređajima

- **Panthera**

Pantera je organizacija posvećena očuvanju divljih mačaka. Njihovi projekti uključuju praćenje životinja te suradnje s lokalnim zajednicama.

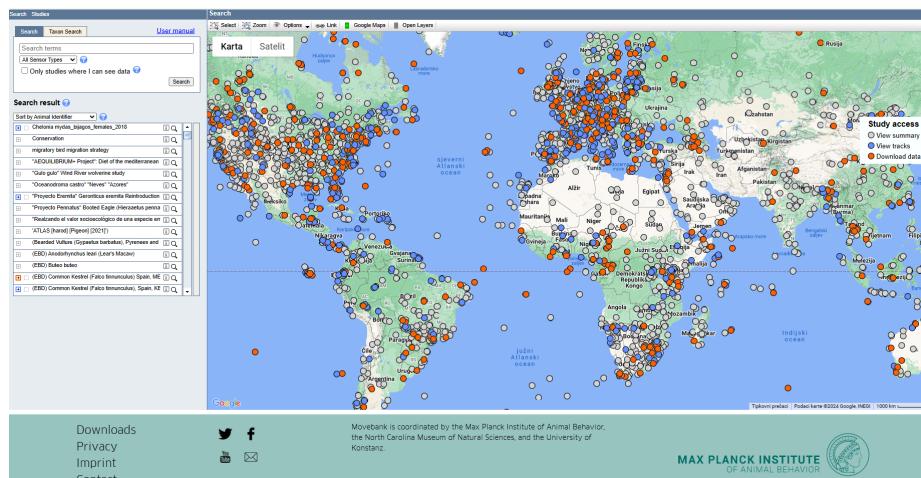


Figure 2.8: Karta za praćenje životinja - Movebank



Figure 2.9: Wildbook za Pirenejskog risa



Figure 2.10: Stranica iNaturalist

# 3. Specifikacija programske potpore

## 3.1 Funkcionalni zahtjevi

Dionici:

1. Naručitelj
2. Korisnici
  - (a) Voditelj postaje
  - (b) Istraživač
  - (c) Tragač
3. Administrator
4. Razvojni tim

Aktori i njihovi funkcionalni zahtjevi:

1. Neregistrirani korisnik (inicijator) može:
  - (a) poslati zahtjev za registraciju sa željenom ulogom za koju se prijavljuje
2. Voditelj postaje (inicijator) može:
  - (a) dobiti pregled koji su tragači dio njegove postaje
  - (b) dodati ili ukloniti tragače njegove postaje
  - (c) definirati na koji način su tragači osposobljeni izvoditi pretraživanje (pješke, dronom, automobilom, cross motorom, brodom ili helikopterom)
  - (d) odabrati konkretnе tragače koji će sudjelovati u akciji
  - (e) ukloniti tragače s pojedine akcije po završetku njihova rada
3. Istraživač (inicijator) može:
  - (a) dobiti pregled svih akcija koje je pokrenuo
  - (b) stvoriti nove akcije pretraživanja i praćenja s detaljima o određenim vrstama, jedinkama ili staništima za proučavanje
  - (c) poslati zahtjev za tragačima s opisom o potrebnim kvalifikacijama voditeljima postaja

- (d) dobiti pregled koji tragači sudjeluju u akciji
- (e) preko karte tragačima pojedinačno zadati zadatke
- (f) označiti da je neki tragač gotov sa svojim zadatkom
- (g) ostaviti komentar za svaki zadatak
- (h) bilježiti i vizualizirati staze kojima su tragači putovali i način kojim su se kretali u obliku toplinskih karata
- (i) odabrati da se za izradu interaktivnih karata koriste neka od idućih informacija: povjesne pozicije svih praćenih životinja, filtrirano po vrsti ili pojedinačno po jedinku te trenutne pozicije praćenih životinja
- (j) ostaviti komentar za ostale sudionike u akciji

4. Tragač (inicijator) može:

- (a) dobiti prikaz na karti zadataka koje treba obaviti, trenutne pozicije ostalih tragača aktivnih na istoj akciji, te trenutne pozicije praćenih životinja
- (b) ostaviti komentar za ostale sudionike u akciji
- (c) ostaviti komentar o praćenoj životinji tijekom akcije

5. Administrator (inicijator) može:

- (a) potvrditi istraživača i voditelja postaje tijekom registracije
- (b) vidjeti popis svih registriranih korisnika i njihovih osobnih podataka
- (c) mijenjati osobne podatke svih registriranih korisnika te njihove uloge

6. Baza podataka (sudionik):

- (a) pohranjuje sve podatke o korisnicima i njihovim ovlastima
- (b) pohranjuje sve podatke o akcijama, lokacijama, životnjama i sl.

### 3.1.1 Obrasci uporabe

#### UC1 - 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 potrebne korisničke podatke (korisničko ime, fotografija, lozinka, ime, prezime i email adresa) te odabire željenu ulogu za koju se prijavljuje
  3. Korisnik prima obavijest o uspješnoj/neuspješnoj registraciji
- **Opis mogućih odstupanja:**
  - 2.a Odabir već zauzetog korisničkog imena i/ili e-maila, unos korisničkog podatka u nedozvoljenom formatu ili pružanje neispravnoga e-maila
    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
  - 3.a Administrator odbija registraciju korisnika koji se želi registrirati kao istraživač ili voditelj postaje
    1. Sustav obavještava korisnika o neuspjelom upisu i vraća ga na stranicu za registraciju

#### UC2 - Prijava u sustav

- **Glavni sudionik:** Korisnik
- **Cilj:** Dobiti pristup korisničkom sučelju
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik ima registrirani korisnički račun
- **Opis osnovnog tijeka:**
  1. Unos korisničkog imena i lozinke
  2. Potvrda o ispravnosti unesenih podataka
  3. Pristup korisničkim funkcijama
- **Opis mogućih odstupanja:**
  - 2.a Neispravno korisničko ime/lozinka

1. Sustav obavještava korisnika o neuspjelom upisu i vraća ga na stranicu za registraciju

### **UC3 - Pregled osobnih podataka**

- **Glavni sudionik:** Korisnik
- **Cilj:** Pregledati osobne podatke
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen
- **Opis osnovnog tijeka:**
  1. Korisnik odabire opciju "Osobni podatci"
  2. Sustav prikazuje osobne podatke korisnika

### **UC4 - Pregled članova postaje**

- **Glavni sudionik:** Voditelj postaje
- **Cilj:** Dobiti popis tragača koji su članovi postaje
- **Sudionici:** Baza podataka
- **Preduvjet:** Voditelj postaje je prijavljen
- **Opis osnovnog tijeka:**
  1. Voditelj bira opciju "Članovi postaje"
  2. Sustav prikazuje popis tragača koji su članovi postaje

### **UC5 - Dodavanje tragača u postaju**

- **Glavni sudionik:** Voditelj postaje
- **Cilj:** Dodati slobodnog tragača među članove postaje
- **Sudionici:** Baza podataka, tragač
- **Preduvjet:** Voditelj postaje je prijavljen
- **Opis osnovnog tijeka:**
  1. Voditelj postaje bira opciju "Dodaj tragača u postaju"
  2. Sustav prikazuje popis slobodnih tragača
  3. Voditelj odabire tragače koje želi dodijeliti svojoj postaji
  4. Voditelj potvrđuje odabir
  5. Sustav ažurira bazu podataka i dodjeljuje odabrane tragače postaji koju vodi voditelj
- **Opis mogućih odstupanja:**
  - 2.a Ne postoji registrirani tragač koji već nije član neke postaje
    1. Sustav obavještava korisnika o nepostojanju slobodnih tragača

### UC6 - Uklanjanje tragača iz postaje

- **Glavni sudionik:** Voditelj postaje
- **Cilj:** Ukloniti tragača iz članova postaje
- **Sudionici:** Baza podataka, tragač
- **Preduvjet:** Voditelj postaje je prijavljen i postoje tragači koji pripadaju njegovoj postaji
- **Opis osnovnog tijeka:**
  1. Voditelj postaje bira opciju "Ukloni tragača iz postaje"
  2. Sustav prikazuje popis tragača koji su članovi postaje
  3. Voditelj odabire tragače koje želi ukloniti iz svoje postaje
  4. Voditelj potvrđuje odabir
  5. Sustav ažurira bazu podataka i odabrane tragače dodaje na listu slobodnih tragača
- **Opis mogućih odstupanja:**
  - 2.a Ne postoji tragač koji je član postaje
    1. Sustav obavještava korisnika o nepostojanju tragača koji su članovi postaje

### UC7 - Definiranje osposobljenosti tragača

- **Glavni sudionik:** Voditelj postaje
- **Cilj:** Definirati na koji način su tragači osposobljeni izvoditi pretraživanje (pješke, dronom, automobilom, cross motorom, brodom ili helikopterom)
- **Sudionici:** Baza podataka, tragač
- **Preduvjet:** Voditelj je prijavljen i postoje tragači koji pripadaju njegovoj postaji
- **Opis osnovnog tijeka:**
  1. Sustav prikazuje popis tragača koji su članovi postaje
  2. Voditelj postaje odabire tragača
  3. Voditelj odabire opciju "Definiraj osposobljenost tragača"
  4. Voditelj odabire koja osposobljenja za način pretraživanja ima tragač
  5. Voditelj potvrđuje odabir
  6. Sustav ažurira bazu podataka s odabranim načinima pretraživanja

### UC8 - Pregled zahtjeva za tragačima

- **Glavni sudionik:** Voditelj postaje
- **Cilj:** Dobiti popis pristiglih zahtjeva za alociranje tragača za sudjelovanje u akciji

- **Sudionici:** Baza podataka
- **Preduvjet:** Voditelj je prijavljen
- **Opis osnovnog tijeka:**
  1. Voditelj odabire opciju "Pristigli zahtjevi"
  2. Sustav prikazuje popis pristiglih zahtjeva za alociranje tragača za sudjelovanje u akciji
- **Opis mogućih odstupanja:**
  - 2.a Nije pristigao niti jedan zahtjev
    1. Sustav obavještava korisnika o nepostojanju pristiglih zahtjeva

### UC9 - Odabir tragača za akciju

- **Glavni sudionik:** Voditelj postaje
- **Cilj:** Odabrati koji tragači će sudjelovati u nekoj akciji
- **Sudionici:** Baza podataka, tragač
- **Preduvjet:** Voditelj je prijavljen i primio je barem jedan zahtjev za alociranjem tragača za sudjelovanje u akciji
- **Opis osnovnog tijeka:**
  1. Sustav prikazuje zahtjev za alociranjem tragača za sudjelovanje u akciji
  2. Voditelj postaje odabire opciju "Odaber tragače za akciju"
  3. Sustav prikazuje popis raspoloživih tragača
  4. Voditelj odabire tragača
  5. Voditelj potvrđuje odabir
  6. Sustav ažurira bazu podataka i dodjeljuje odabrane tragače akciji
- **Opis mogućih odstupanja:**
  - 2.a Nema raspoloživih tragača
    1. Sustav obavještava voditelj o nedostatku raspoloživih tragača

### UC10 - Uklanjanje tragača s akcije

- **Glavni sudionik:** Voditelj postaje
- **Cilj:** Odabrati koji tragači će sudjelovati u nekoj akciji
- **Sudionici:** Baza podataka, tragač
- **Preduvjet:** Voditelj je prijavljen i tragač sudjeluje u nekoj akciji
- **Opis osnovnog tijeka:**
  1. Sustav prikazuje popis tragača koji su članovi postaje
  2. Voditelj odabire tragača kojeg želi ukloniti s neke akcije
  3. Voditelj odabire opciju "Ukloni s akcije"

4. Voditelj potvrđuje odabir
  5. Sustav ažurira bazu podataka i vraća tragača na popis raspoloživih tragača
- **Opis mogućih odstupanja:**
    - 3.a Tragač nije završio sa zadatkom na akciji
      1. Sustav obaveštava voditelja da mu nije dopušteno ukloniti tragača s akcije

### UC11 - Pregled akcija

- **Glavni sudionik:** Istraživač
- **Cilj:** Dobiti popis akcija u kojima istraživač sudjeluje
- **Sudionici:** Baza podataka
- **Preduvjet:** Istraživač je prijavljen
- **Opis osnovnog tijeka:**
  1. Istraživač odabire opciju "Moje akcije"
  2. Sustav prikazuje popis akcija u kojima istraživač sudjeluje

### UC12 - Stvaranje novih akcija

- **Glavni sudionik:** Istraživač
- **Cilj:** Stvoriti novu akciju pretraživanja i praćenja s detaljima o određenim vrstama, jedinkama ili staništima za proučavanje
- **Sudionici:** Baza podataka
- **Preduvjet:** Istraživač je prijavljen
- **Opis osnovnog tijeka:**
  1. Istraživač odabire opciju "Stvori novu akciju"
  2. Sustav prikazuje obrazac za unos detalja o akciji
  3. Istraživač unosi tražene podatke
  4. Istraživač potvrđuje unos
  5. Sustav ažurira bazu podataka s novom akcijom

### UC13 - Slanje zahtjeva za tragačima

- **Glavni sudionik:** Istraživač
- **Cilj:** Poslati zahtjev za tragačima s opisom o potrebnim kvalifikacijama voditeljima postaja
- **Sudionici:** Baza podataka, voditelj postaje
- **Preduvjet:** Istraživač je prijavljen i postoji barem jedan voditelj postaje s tragačima u svojoj postaji

- **Opis osnovnog tijeka:**

1. Istraživač odabire opciju "Pošalji zahtjev za tragačima"
2. Sustav prikazuje obrazac za unos detalja o zahtjevu (kvalifikacije, broj tragača, itd.)
3. Istraživač unosi tražene podatke
4. Istraživač odabire postaju kojoj želi poslati zahtjev
5. Istraživač potvrđuje slanje zahtjeva
6. Sustav ažurira bazu podataka s poslanim zahtjevom

#### UC14 - Pregled članova akcije

- **Glavni sudionik:** Istraživač
- **Cilj:** Dobiti popis tragača koji sudjeluju u akciji
- **Sudionici:** Baza podataka
- **Preduvjet:** Istraživač je prijavljen
- **Opis osnovnog tijeka:**
  1. Sustav prikazuje popis akcija u kojima istraživač sudjeluje
  2. Istraživač odabire akciju
  3. Istraživač odabire opciju "Članovi akcije"
  4. Sustav prikazuje popis tragača koji sudjeluju u akciji

#### UC15 - Dodjela zadatka tragaču

- **Glavni sudionik:** Istraživač
- **Cilj:** Dodijeliti određen zadatak akcije nekom tragaču koji sudjeluje u akciji
- **Sudionici:** Baza podataka, tragač
- **Preduvjet:** Istraživač je prijavljen i postoji barem jedan tragač koji sudjeluje u akciji
- **Opis osnovnog tijeka:**
  1. Sustav prikazuje popis tragača koji sudjeluju u akciji
  2. Istraživač odabire tragača kojem želi dodijeliti zadatak
  3. Istraživač odabire opciju "Dodijeli zadatak tragaču"
  4. Sustav prikazuje kartu
  5. Istraživač odabire rutu i odredišnu lokaciju zadatka
  6. Istraživač odabire koji uređaj želi da bude postavljen(kamera ili uređaj za praćenje)
  7. Istraživač potvrđuje dodjelu zadatka
  8. Sustav ažurira bazu podataka s dodijeljenim zadatkom tragaču

### UC16 - Pregled zadataka

- **Glavni sudionik:** Istraživač
- **Cilj:** Dobiti popis zadataka zadanih u akciji
- **Sudionici:** Baza podataka
- **Preduvjet:** Istraživač je prijavljen
- **Opis osnovnog tijeka:**
  1. Sustav prikazuje popis akcija u kojima istraživač sudjeluje
  2. Istraživač odabire akciju
  3. Istraživač odabire opciju "Pregled zadataka"
  4. Sustav prikazuje popis zadataka koje je istraživač dodijelio

### UC17 - Ostavljanje komentara za zadatak

- **Glavni sudionik:** Istraživač
- **Cilj:** Ostaviti komentar za neki zadatak
- **Sudionici:** Baza podataka
- **Preduvjet:** Istraživač je prijavljen i dodijelio je barem jedan zadatak
- **Opis osnovnog tijeka:**
  1. Sustav prikazuje popis zadataka koje je istraživač dodijelio
  2. Istraživač odabire zadatak
  3. Istraživač odabire opciju "Ostavi komentar"
  4. Istraživač unosi željeni komentar
  5. Istraživač potvrđuje unos
  6. Sustav ažurira bazu podataka s novim komentarom za zadatak

### UC18 - Dovršavanje zadataka

- **Glavni sudionik:** Istraživač
- **Cilj:** Označiti da je neki zadatak dovršen
- **Sudionici:** Baza podataka
- **Preduvjet:** Istraživač je prijavljen i dodijelio je barem jedan zadatak
- **Opis osnovnog tijeka:**
  1. Sustav prikazuje popis zadataka koje je istraživač dodijelio
  2. Istraživač odabire zadatak
  3. Istraživač odabire opciju "Označi dovršenim"
  4. Istraživač potvrđuje unos
  5. Sustav ažurira bazu podataka s oznakom da je tragač završio zadatak

### **UC19 - Prikaz interaktivne karte**

- **Glavni sudionik:** Istraživač
- **Cilj:** Stvoriti interaktivnu kartu
- **Sudionici:** Baza podataka
- **Preduvjet:** Istraživač je prijavljen
- **Opis osnovnog tijeka:**
  1. Istraživač odabire opciju "Karta"
  2. Sustav učitava kartu s posljednjim spremlijenim promjenama

### **UC20 - Uređivanje interaktivne karte**

- **Glavni sudionik:** Istraživač
- **Cilj:** Stvoriti interaktivnu kartu
- **Sudionici:** Baza podataka
- **Preduvjet:** Istraživač je prijavljen
- **Opis osnovnog tijeka:**
  1. Istraživač odabire opciju "Uredi kartu"
  2. Sustav prikazuje opcije informacija koje mogu biti prikazane na karti
  3. Istraživač uređuje kartu prema svojim potrebama
  4. Istraživač potvrđuje promjene
  5. Sustav pohranjuje interaktivnu kartu u bazu podataka

### **UC21 - Ostavljanje komentara na karti**

- **Glavni sudionik:** Istraživač, tragač
- **Cilj:** Ostaviti komentar na karti drugim sudionicima akcije
- **Sudionici:** Baza podataka
- **Preduvjet:** Istraživač ili tragač je prijavljen i sudjeluje u akciji
- **Opis osnovnog tijeka:**
  1. Sustav prikazuje kartu
  2. Istraživač ili tragač odabire opciju "Ostavi komentar"
  3. Istraživač ili tragač unosi željeni komentar
  4. Istraživač ili tragač potvrđuje unos
  5. Sustav ažurira bazu podataka s novim komentarom za sudionike akcije

### **UC22 - Ostavljanje komentara za praćenu životinju**

- **Glavni sudionik:** Tragač

- **Cilj:** Ostaviti komentar praćenoj životinji
- **Sudionici:** Baza podataka
- **Preduvjet:** Tragač je prijavljen i sudjeluje u akciji
- **Opis osnovnog tijeka:**
  1. Tragač odabire opciju "Ostavi komentar za životinju"
  2. Sustav prikazuje polje za unos komentara
  3. Tragač unosi željeni komentar
  4. Tragač potvrđuje unos
  5. Sustav ažurira bazu podataka s novim komentarom za praćenu životinju

#### UC23 - Pregled karte

- **Glavni sudionik:** Tragač
- **Cilj:** Pristupiti karti za pregled zadataka koje treba obaviti, trenutne pozicije ostalih tragača aktivnih na istoj akciji, te trenutne pozicije praćenih životinja
- **Sudionici:** Baza podataka
- **Preduvjet:** Tragač je prijavljen i sudjeluje u akciji
- **Opis osnovnog tijeka:**
  1. Tragač odabire opciju "Karta"
  2. Sustav prikazuje kartu s označenim zadacima, pozicijama tragača i praćenih životinja
  3. Tragač može interaktivno pregledavati kartu i pratiti informacije o akciji

#### UC24 - Pregled zahtjeva za registracijom

- **Glavni sudionik:** Administrator
- **Cilj:** Dobiti popis svih pristiglih zahtjeva za registracijom u ulozi voditelja postaje ili istraživača
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je registriran i dodijeljena su mu prava administratora te postoji barem jedan zahtjev za registracijom u ulozi voditelja ili istraživača
- **Opis osnovnog tijeka:**
  1. Administrator odabire opciju "Pristigli zahtjevi"
  2. Sustav prikazuje popis pristiglih zahtjeva za registraciju u ulozi voditelja postaje ili istraživača

### UC25 - Potvrda registracije

- **Glavni sudionik:** Administrator
- **Cilj:** Potvrditi/odbiti registraciju korisnika koji se želi registrirati kao istraživač ili voditelj postaje
- **Sudionici:** Baza podataka, neregistrirani korisnik
- **Preduvjet:** Administrator je registriran
- **Opis osnovnog tijeka:**
  1. Sustav prikazuje popis pristiglih zahtjeva za registraciju u ulozi voditelja postaje ili istraživača
  2. Administrator odabire zahtjev
  3. Administrator odabire opciju "Potvrdi" ili "Odbij"
  4. Ako administrator odabere opciju potvrди, sustav ažurira bazu podataka s novim korisnikom

### UC26 - Pregled svih korisnika

- **Glavni sudionik:** Administrator
- **Cilj:** Pregledati sve registrirane korisnike i njihove osobne podatke
- **Sudionici:** Baza podataka
- **Preduvjet:** Administrator je registriran
- **Opis osnovnog tijeka:**
  1. Administrator odabire opciju "Pregled korisnika"
  2. Sustav prikazuje popis svih registriranih korisnika i njihovih osobnih podataka

### UC27 - Promjena osobnih podataka korisnika

- **Glavni sudionik:** Administrator
- **Cilj:** Promijeniti osobne podatke korisnika
- **Sudionici:** Baza podataka, korisnik
- **Preduvjet:** Administrator je registriran te postoji barem jedan registrirani korisnik
- **Opis osnovnog tijeka:**
  1. Sustav prikazuje popis svih registriranih korisnika i njihovih osobnih podataka
  2. Administrator bira korisnika
  3. Administrator odabire opciju "Uredi podatke"

4. Administrator radi promjene
  5. Administrator sprema promjene
  6. Sustav ažurira bazu podataka
- **Opis mogućih odstupanja:**
    - 5.a Administrator promijeni podatke, ali ne odabere opciju "Spremi promjenu"
      1. Sustav obavještava administratora da nije spremio podatke prije izlaska iz prozora

### UC28 - Promjena uloge korisnika

- **Glavni sudionik:** Administrator
- **Cilj:** Promijeniti osobne podatke korisnika
- **Sudionici:** Baza podataka, korisnik
- **Preduvjet:** Korisnik je registriran i dodijeljena su mu prava administratora te postoji barem jedan registrirani korisnik
- **Opis osnovnog tijeka:**
  1. Sustav prikazuje popis svih registriranih korisnika i njihovih osobnih podataka
  2. Administrator bira korisnika
  3. Administrator odabire opciju "Promijeni ulogu"
  4. Administrator odabire novu ulogu korisnika
  5. Administrator sprema promjene
  6. Sustav ažurira bazu podataka
- **Opis mogućih odstupanja:**
  - 4.a Administrator promijeni ulogu voditelja postaje
    1. Sustav obavještava administratora da mora odabrati novog voditelja postaje i prikazuje mu popis članova postaje
    2. Administrator bira tragača kojem će promijeniti ulogu u voditelja postaje
  - 4.b Administrator promijeni ulogu u voditelja postaje
    1. Sustav obavještava administratora da će dosadašnjem voditelju postaje uloga biti promjena u tragača

## Dijagrami obrazaca uporabe

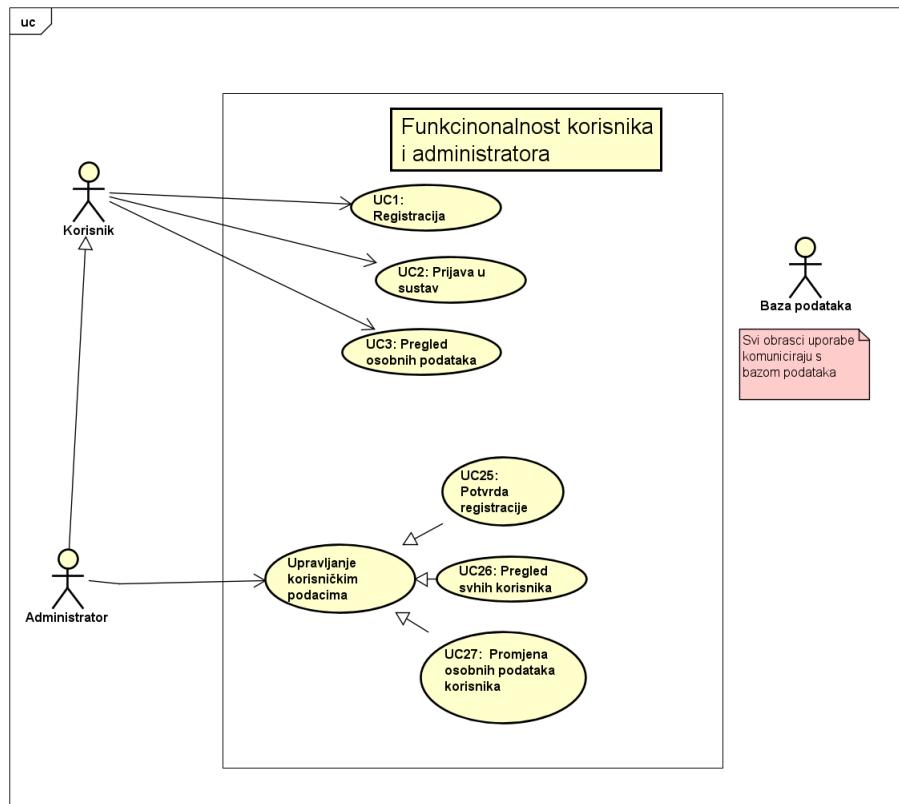


Figure 3.1: Dijagram obrasca uporabe, funkcionalnost korisnika i administratora

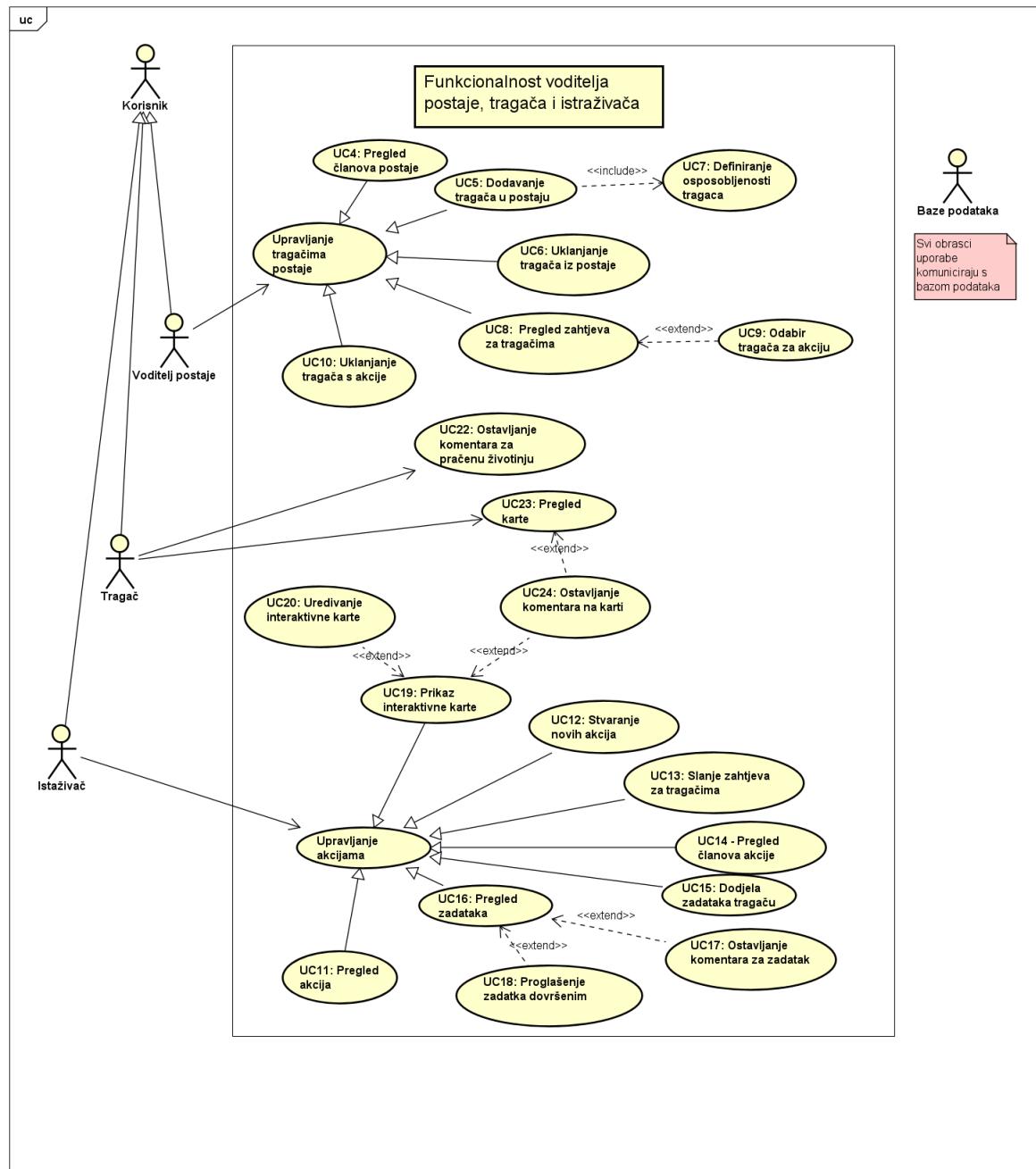


Figure 3.2: Dijagram obrasca uporabe, voditelja postaje, tragača i istraživača

### 3.1.2 Sekvencijski dijagrami

#### Obrazac uporabe UC9 - Odabir tragača za akciju

Voditelj postaje šalje zahtjev za prikazom popisa pristiglih zahtjeva za alociranjem tragača za sudjelovanje u akciji. Poslužitelj dohvaća podatke o pristiglim zahtjevima i prikazuje ih. Voditelj odabire koji zahtjev želi detaljnije pregledati. Poslužitelj dohvaća pojedinosti o zahtjevu i prikazuje ih. Voditelj zatim šalje zahtjev za odabirom tragača koji će sudjelovati u akciji. Poslužitelj dohvaća tragače koji su raspoloživi. Raspoloživi tragači su oni koji u tom trenutku ne sudjeluju u niti jednoj akciji. U slučaju da takvih tragača ne postoji, poslužitelj ispisuje poruku "Nema raspoloživih tragača". U suprotnom, poslužitelj prikaže popis raspoloživih tragača. Voditelj postaje zatim bira koje tragače želi dodati u akciju sve dok ne pošalje zahtjev za potvrdom odabira. Tada poslužitelj sprema odabir u bazu podataka.

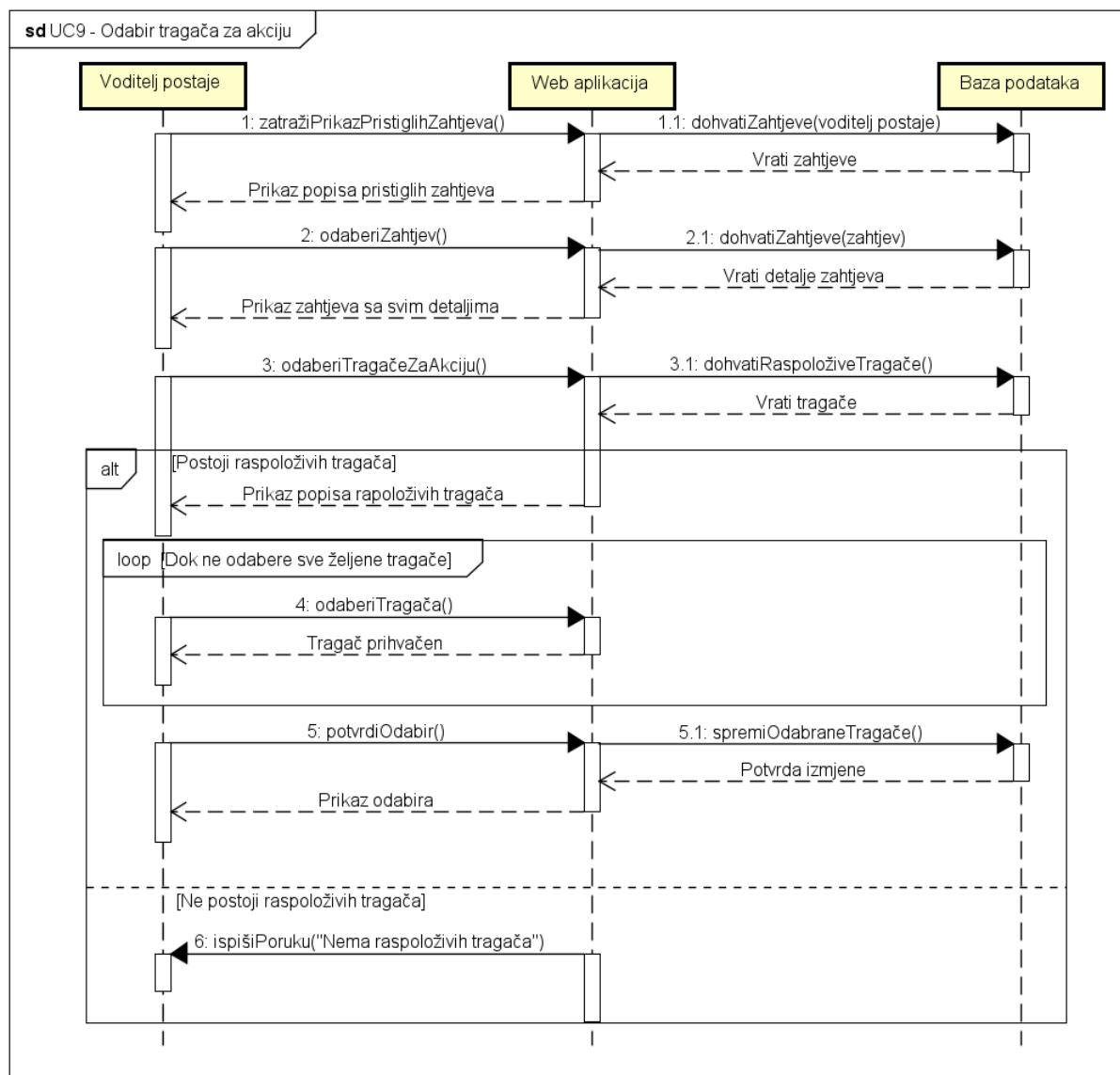


Figure 3.3: Sekvencijski dijagram za UC9

**Obrazac uporabe UC15 - Dodjela zadatka tragaču**

Istraživač šalje zahtjev za prikazom akcija u kojima sudjeluje. Poslužitelj dohvaća podatke o istraživačevim akcijama i prikazuje ih. Istraživač odabire koju akciju želi detaljni pregledati. Poslužitelj dohvaća pojedinosti o akciji i prikazuje ih. Istraživač zatim šalje zahtjev za pregled tragača koji sudjeluju u akciji. Poslužitelj dohvaća tragače koji sudjeluju u akciji i prikazuje ih. Istraživač odabire tragača i poslužitelj bilježi taj odabir. Istraživač zatim šalje zahtjev o dodjeli zadatke odabranom tragaču. Poslužitelj dohvaća spremljenu kartu korištenu za ovu akciju i prikazuje ju zajedno s dijelom namijenjenim za odabir uređaja kojeg istraživač želi da se postavi(kamera ili uređaj za praćenje). Istraživač zatim na karti označava kojom rutom želi da se tragač kreće i do koje odredišne lokacije. Poslužitelj bilježi odabranu rutu. Istraživač onda bira između ponuđenih uređaja što poslužitelj također bilježi. Na kraju, istraživač potvrđuje dodjelu zadatka odabranom tragaču. Poslužitelj spremi u bazu podataka novi zadatak s svim navedenim specifikacijama i naznačuje tragača kojem je namijenjen.

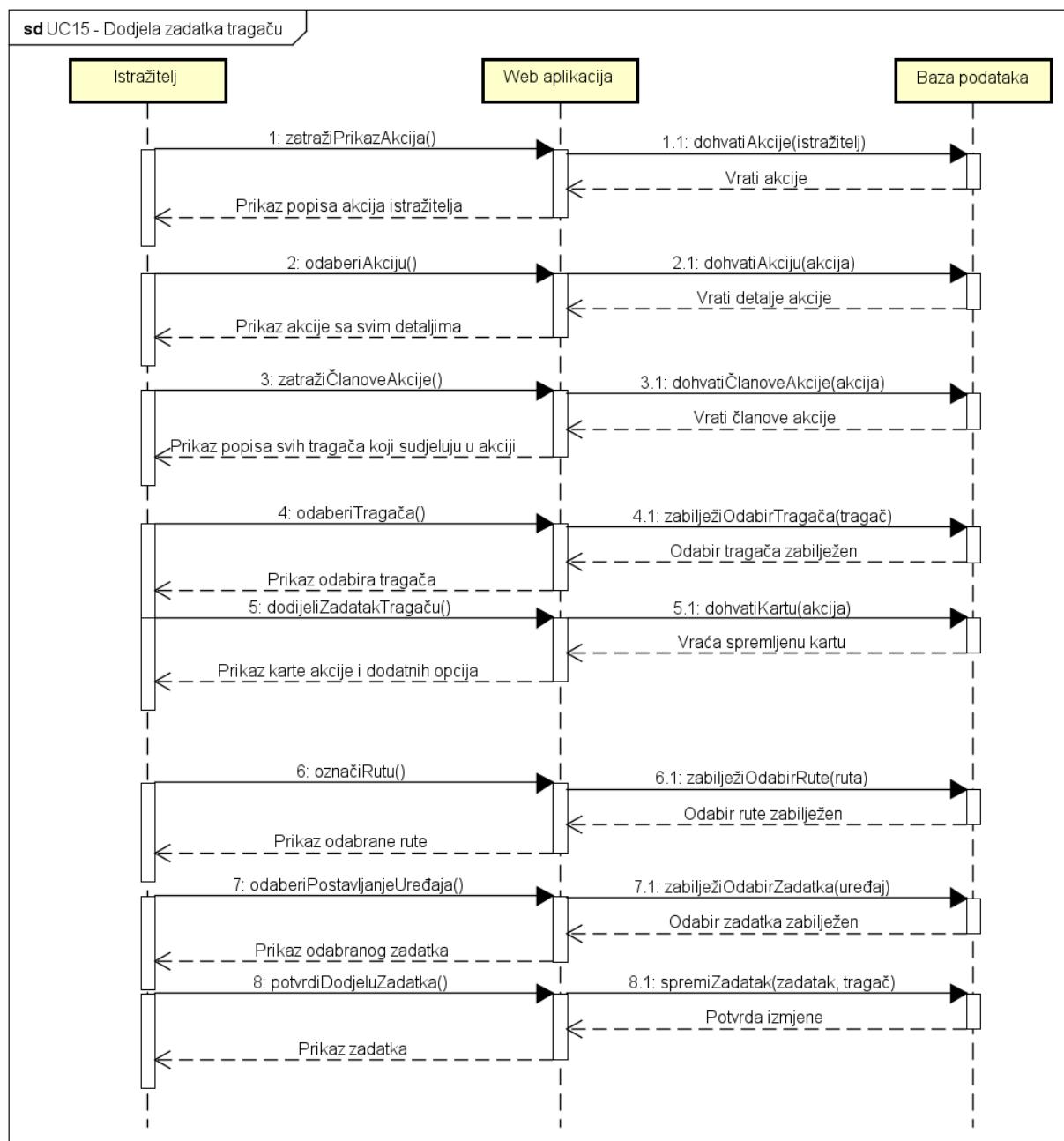


Figure 3.4: Sekvencijski dijagram za UC15

**Obrazac uporabe U20 - Uređivanje interaktivne karte**

Istraživač šalje zahtjev za prikazom interaktivne karte. Poslužitelj dohvaća podatke o posljednjom spremjenom odabiru o prikazu informacija na karti i prikazuje ih. Istraživač zatim šalje zahtjev o uređivanju karte. Poslužitelj dohvaća podatke o svim mogućim informacijama koje mogu biti prikazane na karti i prikazuje ih istraživaču. Istraživač sada šalje zahtjev s odabirom koje informacije želi da budu prikazane. Poslužitelj provjerava je li odabrana opcija prikaza povijesnih pozicija svih praćenih životinja ili povijesnih pozicija svih tragača koji sudjeluju u akciji. Ako jest, onda poslužitelj prikazuje opcije filtriranja. U slučaju prikaza povijesnih pozicija svih praćenih životinja, podaci mogu biti filtrirani po vrsti ili pojedinačno po jedinki, dok prikaz povijesnih pozicija svih tragača koji sudjeluju u akciji može biti filtriran po tipu prijevoza ili pojedinačno po tragaču. Istraživač šalje zahtjev s odabirom kako želi da informacija bude filtrirana. Poslužitelj dohvaća informaciju s primjenjenim filtriranjem iz baze podataka, a u slučaju da je pri odabiru informacija bio odabran prikaz trenutnih pozicija praćenih životinja ili tragača, onda poslužitelj dohvaća tu informaciju bez ikakvog dodatnog filtriranja. Istraživač zatim šalje potvrdu za primjenu novog uređenja. Poslužitelj zatim sprema odabir u bazu podataka te prikazuje novonastalu kartu.

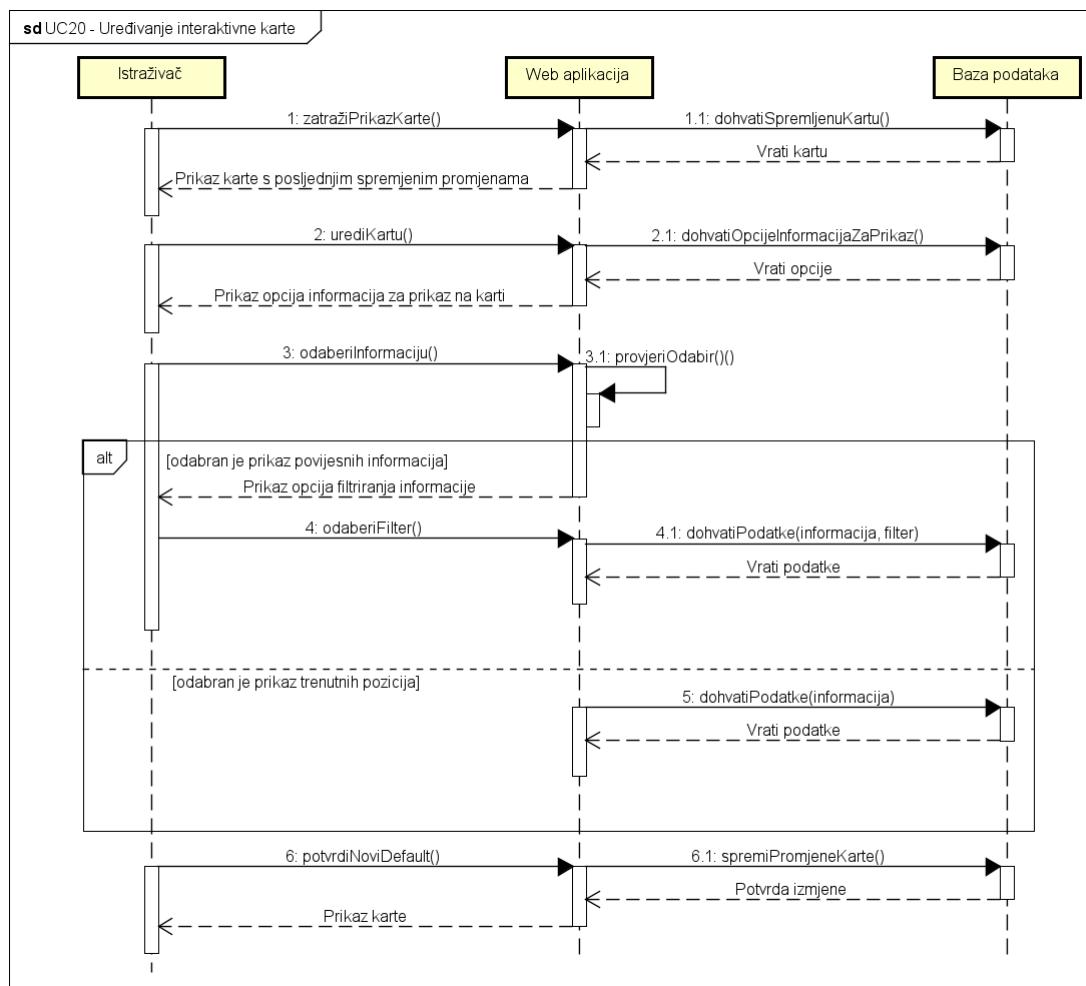


Figure 3.5: Sekvencijski dijagram za UC20

### 3.2 Ostali zahtjevi

- Web aplikaciju treba implementirati koristeći objektno-orientirane jezike.
- Korisničko sučelje pri unosu i prikazu podataka treba podržavati hrvatski jezik uključujući dijakritičke znakove.
- Potrebno je osigurati obranu sustava od pogrešnog korištenja korisničkog sučelja s dodatnim provjerama valjanosti.
- Sustav mora podržavati veći broj korisnika u stvarnom vremenu.
- Korisničko sučelje mora biti jednostavno i intuitivno za uporabu.
- Sustav je potrebno oblikovati na način koji pridonosi ponovnoj uporabi kao što je uvođenje kopči u program.
- Programska potpora treba biti pripremljena za promjene u budućnosti.
- Sustav mora odgovoriti na zahtjev korisnika u roku od nekoliko sekundi.
- Pristup bazi podataka mora biti dobro zaštićen.
- Dohvat sadržaja iz baze podataka trebao bi biti brz i učinkovit.
- Programski proizvod treba raditi na što većem broju različitih platformi što se omogućuje korištenjem programskih jezika neovisnih o platformi.
- Komunikacija između web klijenta i web poslužitelja odvija se preko HTTPS protokola.

## 4. Arhitektura i dizajn sustava

Arhitektura našeg sustava sastoji se od odvojenih backend (Spring Boot) i frontend (React) dijelova. Backend je implementiran pomoću Spring Boot radnog okvira, dok je frontend razvijen korištenjem React biblioteke.

### **Backend (Spring Boot):**

#### *Controller sloj:*

Kontroleri (Controllers) su odgovorni za obradu HTTP zahtjeva i interakciju s korisnicima preko API-ja. Svaki kontroler ima metode koje definiraju ponašanje sustava u odgovoru na različite zahtjeve (npr., dohvati podataka, ažuriranje resursa).

#### *Service sloj:*

Servisi (Services) sadrže poslovnu logiku aplikacije. Oni obrađuju zahtjeve primljene od kontrolera, vrše potrebne validacije, te komuniciraju s entitetima u bazi podataka. Servisi su odvojeni kako bi omogućili ponovnu uporabu koda i jasnu strukturu.

#### *Entity sloj:*

Entiteti predstavljaju objekte koji se pohranjuju u bazi podataka. Ovi entiteti često odražavaju strukturu podataka u aplikaciji i pomažu u mapiranju podataka iz baze.

#### *Database sloj:*

Za povezivanje s bazom podataka, koristili smo PostgreSQL. JPA (Java Persistence API) može se koristiti za mapiranje objektnih entiteta na relacijsku bazu podataka.

### **Frontend (React):**

#### *Components:*

Komponente čine osnovnu građevnu jedinicu React aplikacije. Svaka komponenta obavlja određeni dio funkcionalnosti i ima svoje stanje (state) i svojstva (props).

#### *API pozivi:*

Komponente koje zahtijevaju podatke s backend-a koriste HTTP zahtjeve prema odgovarajućim API endpointima koristeći paketa Axios.

#### *Routing:*

React Router se koristi za omogućavanje rute u aplikaciji, što omogućava navigaciju između različitih dijelova aplikacije.

### Styling:

Za stilizaciju koristimo CSS i CSS-in-JS pristup, gdje stilovi mogu biti integrirani unutar JavaScript datoteka.

Ova arhitektura omogućava jasnu odvojenost između frontend i backend dijelova, olakšava održavanje, i omogućuje timsku suradnju s obzirom na to da se razvoj obavlja na odvojenim tehnologijama i dijelovima sustava.

## 4.1 Baza podataka

Za bazu podataka koristi se relacijska baza podataka. Svaka tablica ima svoje ime i atribute. Vrste atributa koji se mogu nalaziti u tablici su primarni ključ, strani ključ ili atribut s nekom informacijom vezanom za tablicu. Baza podataka sastoji se od tablica:

- AppUser
- ConfirmationToken
- StationManager
- SearcherInTheField
- Action
- Station
- Animal
- PastRoutes
- PastLocations
- AnimalComment
- MapComment
- Task

### 4.1.1 Opis tablica

U tablici **AppUser** pohranjuju se podaci o svim korisnicima: *id, userName, image, firstName, lastName, email, password, appUserRole, locked, enabled*. Primarni ključ je *id*, i nema stranih ključeva. S atributom *id* je u odnosu One-to-One s tablicama **StationManager** i **SearcherInTheField** i u odnosu One-to-Many s tablicama **ConfirmationToken** i **Action**.

AppUser		
id	BIGINT	id korisnika
userName	VARCHAR	korisničko ime
image	BYTEA	heksadekatski zapis korisničke slike
firstName	VARCHAR	ime korisnika
lastName	VARCHAR	prezime korisnika
email	VARCHAR	email korisnika
password	VARCHAR	lozinka korisnika
appUserRole	VARCHAR	uloga korisnika
locked	BOOLEAN	je li korisnika potvrdio admin
enabled	BOOLEAN	je li korisnik potvrđen emailom

U tablici **ConfirmationToken** pohranjuju se podaci za token za potvrdu posлану кориснику: *id, token, createdAt, expiresAt, confirmedAt, user*. Svaki token je повезан с корисником којем је послан преко *user* у који се спрема id корисника. Примарни клjuč је *id*, а страни клjuč је *user*. С атрибутом *user* је у односу Many-to-One с табличом **AppUser**.

ConfirmationToken		
id	BIGINT	id tokena
token	VARCHAR	token
createdAt	TIMESTAMP	vrijeme kada je token stvoren
expiresAt	TIMESTAMP	vrijeme kada token ističe
confirmedAt	TIMESTAMP	vrijeme kada je token potvrđen
user	BIGINT	id korisnika којему је послан token

У таблици **StationManager** погранјују се додатни подаци за водитеља постaje: *stationManagerId, userId, stationId*. Примарни клjuč је *stationManagerId*, те су *userId* и *stationId* страни клјучеви. С атрибутом *userId* је у односу One-to-One с табличом **AppUser**, с атрибутом *stationManagerId* је у односу Many-to-One с табличом **Station**.

StationManager		
id	BIGINT	id korisničkog računa voditelja
stationId	BIGINT	id postaje koju vodi voditelj
userId	BIGINT	id usera

U tablici **SearcherInTheField** primarni ključ je *searcherInTheFieldId*. Strani ključevi su *stationId*, *userId* i *actionId*. S atributom *userId* je u odnosu One-to-One s tablicom **AppUser**, s atributom *stationId* je u odnosu Many-to-One s tablicom **Station**, s atributom *actionId* je u odnosu Many-to-One s tablicom **Action**.

SearcherInTheField		
searcherInTheFieldId	BIGINT	id korisničkog računa tragača
stationId	BIGINT	id postaje kojoj pripada
userId	BIGINT	id usera
qualification	VARCHAR	osposobljenosti tragača
currentPosition	JSON	trenutna pozicija tragača
actionId	BIGINT	id akcije na kojoj tragač sudjeluje

U tablici **Action** pohranjuju se podaci vezani uz određenu akciju. Primarni ključ je *actionId*, a strani ključ je *appUserId*. S atributom *appUserId* je u odnosu Many-to-One s tablicom **AppUser** i s atributom *stationId* je u odnosu One-to-One s tablicom **Station**.

Action		
actionId	BIGINT	id akcije
appUserId	BIGINT	id korisnika (istraživača) koji je započeo akciju
stationId	BIGINT	id postaje kojoj se posalo zahtjev za tragačima
actionName	TEXT	naziv akcije
actionType	TEXT	tip akcije

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

Action		
locationName	TEXT	ime lokacije na kojoj se odvija akcija (npr. Biokovo)
mapViewCriteria	JSON	odabrani kriteriji (od strane istraživača) za prikaz životinja na mapi
qualificationsJson	JSON	popis traženih kvalifikacija tragača na akciji

U tablici **Station** pohranjuju se podaci vezani uz postaju. Primarni ključ je *stationId*. Strani ključ je *stationId*. S atributom *stationId* je u odnosu Many-to-One s tablicom **Station**.

Station		
stationId	BIGINT	id postaje
stationName	VARCHAR	ime postaje
coordinatesJson	JSON	popis koordinata postaje te koordinata za prikaz područje pokrivanja za određeno osposobljenje

U tablici **Animal** o svim životinjama za određenu postaju. Primarni ključ je *animalId* nema stranih ključeva. S atributom *id* je u odnosu One-to-Many s tablicom **Location**.

Animal		
animalId	BIGINT	id životinje
name	VARCHAR	ime životinje
breed	VARCHAR	vrsta životinje
description	VARCHAR	opis životinje
image	BYTEA	slika životinje
currentPosition	JSON	trenutna pozicija životinje

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

Animal		
stationId	BIGINT	id postaje, odn. područja (npr. postaja Biokovo) kojoj životinja pripada

U tablici **PastRoutes** pohranjuju se podaci o svim prijašnjim rutama kojima je istraživač prošao na određenoj akciji. Primarni ključ je *pastRoutesId*, a strani ključevi su *actionId* i *searcherId*. S atributom *actionId* je u odnosu Many-to-One s tablicom **Action**, a s atributom *searcherId* je u odnosu Many-to-One s tablicom **SearcherInTheField**.

PastRoutes		
pastRoutesId	BIGINT	id rute
routeWaypoints	JSON	koordinate rute
actionId	BIGINT	id akcije na kojoj je istraživač prošao tom rutom
searcherId	BIGINT	id istraživača čija je ruta zabilježena

U tablici **PastLocations** pohranjuju se podaci o svim prijašnjim lokacijama životinja ili istraživača na određenoj akciji. Primarni ključ je *pastLocationId*, a strani ključevi su *actionId*, *animalId* i *searcherId*. S atributom *actionId* je u odnosu Many-to-One s tablicom **Action**, s atributom *animalId* je u odnosu Many-to-One s tablicom **Animal**, a s atributom *searcherId* je u odnosu Many-to-One s tablicom **SearcherInTheField**.

.

PastLocations		
pastLocationId	BIGINT	id lokacije
positionCoordinates	JSON	koordinate lokacije
actionId	BIGINT	id akcije tokom koje je zabilježena lokacija
searcherId	BIGINT	id istraživača čija je lokacija zabilježena
animalId	BIGINT	id životinje čija je prijašnja lokacija zabilježena

U tablici **AnimalComment** pohranjuju se svi komentari koji su ostavljeni životinji na određenoj akciji. Primarni ključ je *animalCommentId*, a strani ključevi su *actionId* i *animalId*. S atributom *actionId* je u odnosu Many-to-One s tablicom **Action**, te s atributom *animalId* je u odnosu Many-to-One s tablicom **Animal**.

AnimalComment		
animalCommentId	BIGINT	id komentara
comment	VARCHAR	komentar ostavljen životinji
userName	VARCHAR	korisničko ime korisnika koji je ostavio komentar životinji lokacije
actionId	BIGINT	id akcije tokom koje je zabilježena lokacija
animalId	BIGINT	id životinje čija je prijašnja lokacija zabilježenja

U tablici **MapComment** pohranjuju se svi komentari koji su ostavljeni na mapi na određenoj akciji. Primarni ključ je *mapCommentId*, a strani ključ je *actionId*. S atributom *actionId* je u odnosu Many-to-One s tablicom **Action**.

MapComment		
mapCommentId	BIGINT	id komentara
comment	VARCHAR	komentar ostavljen životinji
userName	VARCHAR	korisničko ime korisnika koji je ostavio komentar životinji lokacije
positionCoordinates	JSON	koordinate lokacije gdje je na mapi ostavljen komentar
actionId	BIGINT	id akcije tokom koje je zabilježena lokacija

U tablici **Task** pohranjuju se podaci zadatka koji su dodjeljeni tragaču na određenoj akciji. Primarni ključ je *taskId*, strani ključevi su *actionId* i *searcher*. S atributom *actionId* je u odnosu Many-to-One s tablicom **Action**, a s atributom *searcherId* je u odnosu Many-to-One s tablicom **SearcherInTheField**.

Task		
taskId	BIGINT	id zadatka
taskComment	VARCHAR	komentar za zadatak od strane istraživača
taskToDo	VARCHAR	opis što tragač mora napraviti
completed	BOOLEAN	je li zadatak završen ili ne
routeWaypoints	JSON	koordinate rute koju tragač treba proći na zadatku
searcherId	BIGINT	id istraživača čija je lokacija zabilježena
actionId	BIGINT	id akcije kojoj pripada zahtjev

#### 4.1.2 Dijagram baze podataka

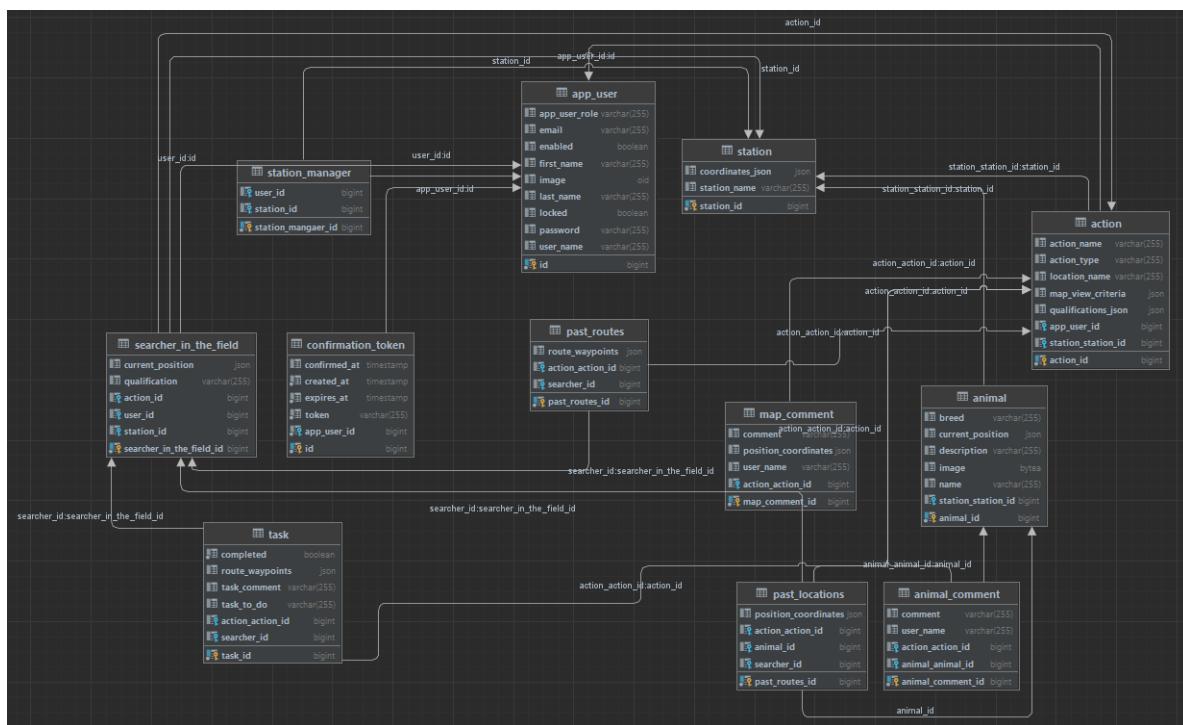


Figure 4.1: Dijagram baze podataka

## 4.2 Dijagram razreda

Dijagram razreda podijeljen je zbog bolje preglednosti na tri dijela: Controllers, Models i DTO. Prikazane su veze koje ostvaruju razredi unutar istog dijela dijagrama, a odnosi između razreda u različitim dijelovima mogu se zaključiti iz tipova atributa. Metode korištene u Controller razredima vraćaju *ResponseEntity*, koji predstavlja HTTP odgovor, ili samo kod odgovora HTTP-a. U svom radu koriste objekte za prijenos podataka (DTO) i ostvaruju komunikaciju s klijentskom stranom.

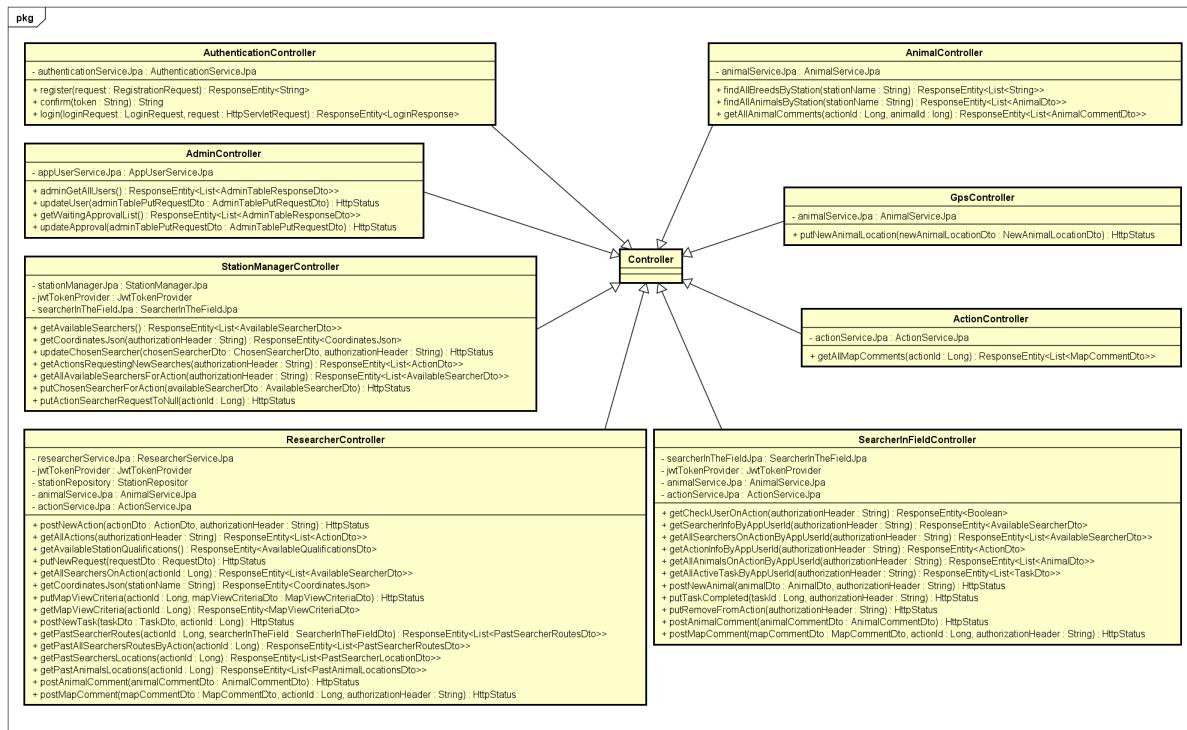


Figure 4.2: Dijagram razreda - dio Controllers

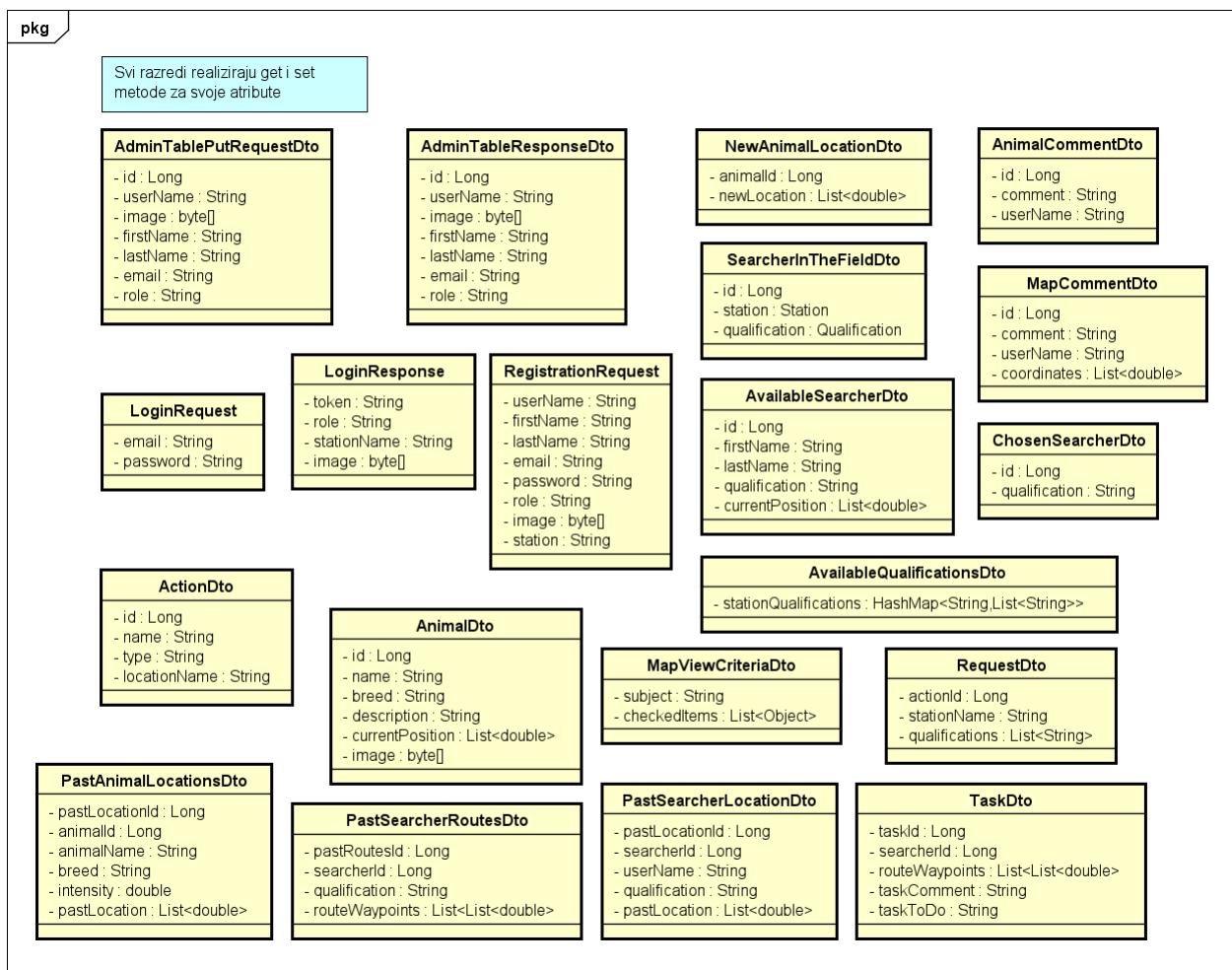


Figure 4.3: Dijagram razreda - dio DTO

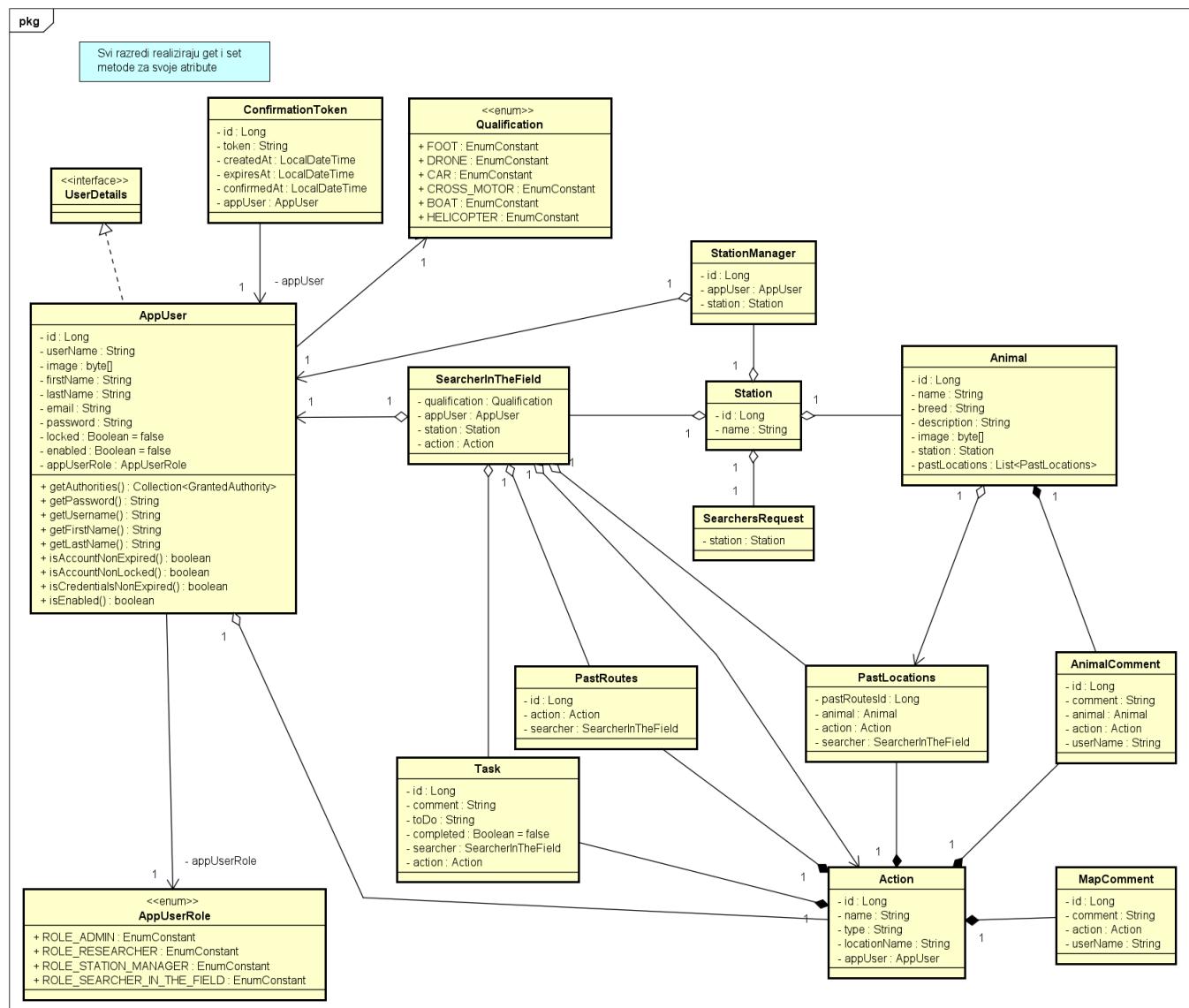


Figure 4.4: Dijagram razreda - dio Models

## 4.3 Dijagram stanja

Dijagrami stanja prikazuju kako sustav prelazi iz jednog stanja u drugo kao odgovor na događaje. Prijavljenom korisniku, u ovom slučaju istraživaču, prikazuje se početna stranica s aktivnim akcijama i nekoliko opcija: stvaranje nove akcije, prikaz interaktivne karte i slanje zahtjeva za tragačima. Nakon odabira opcije stvaranja nove akcije, korisniku se prikazuje forma za unos podataka o akciji. Spremanjem akcije korisnik ponovno vidi početnu stranicu. Odabirom slanja zahtjeva za tragačima za neku od akcija, istraživaču se prikazuje forma za unos detalja o zahtjevu. U slučaju da nema slobodnih tragača, sustav obavijesti korisnika i čeka na idući zahtjev. Odabirom opcije „Više detalja“, istraživaču se prikaže interaktivna karta koju može uređivati po želji. Ako je odabran prikaz tragača na karti, korisnik može nekome od njih dodijeliti zadatak. Zatim ima opciju unosa komentara na zadatak. Također, istraživač može ostaviti komentar na samoj karti. Korisnik može pregledati svoje osobne podatke odabirom opcije „Moji podaci“.

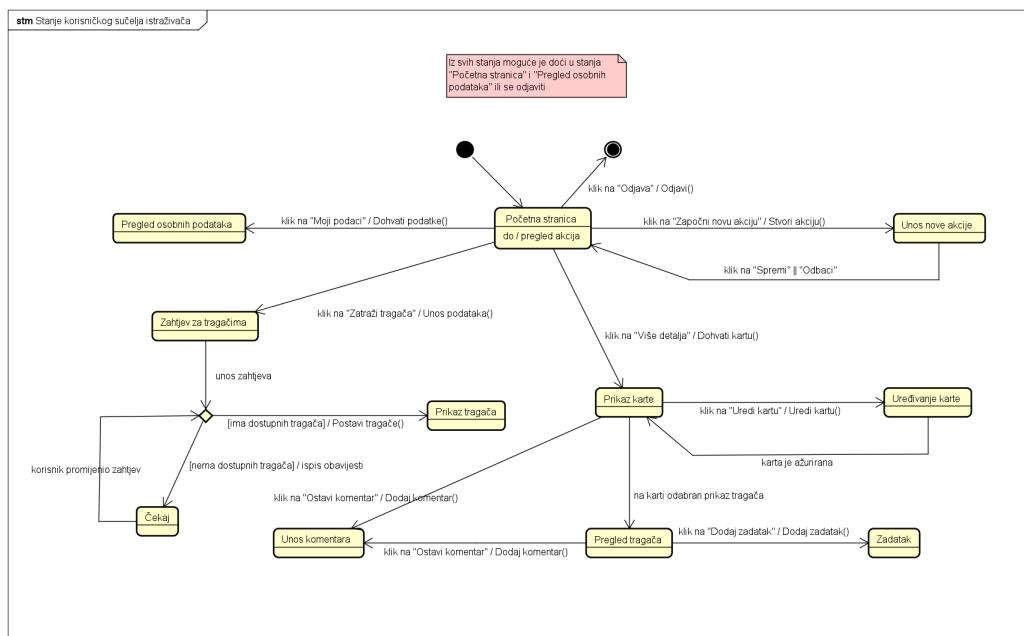


Figure 4.5: Dijagram stanja

## 4.4 Dijagram aktivnosti

Dijagrami aktivnosti upotrebljavaju se za modeliranje dinamičkog ponašanja sustava. Izvođenje aktivnosti prikazano je kroz niz akcija koje čine upravljačke tokove i tokove objekata. Prikazan je proces stvaranja nove akcije. Aktivnost započinje prijavom istraživača u sustav. Nakon uspješne prijave istraživaču se prikazuje početna stranica. Odabirom opcije za stvaranje nove akcije, aplikacija prikazuje formu za unos podataka o akciji. Istraživač unosi podatke o akciji, a sustav ih pohranjuje u bazu podataka. Zatim istraživač šalje zahtjev za tragačima, a aplikacija prikazuje formu za unos detalja o zahtjevu. Aplikacija prosljeđuje zahtjev voditelju postaje koji dodjeljuje tragače akciji (prepostavlja se da ima dostupnih tragača). Popis tragača se pohranjuje u bazu i prikazuje u aplikaciji. Na zahtjev za dodjeljivanje zadatka nekom od tragača, prikazuje se forma za unos zadatka. Zadatak se sprema u bazu podataka. Na zahtjev za pregledom zadataka dodijeljenih tragaču aplikacija prikazuje sve zadatke, a istraživač unosi komentar na odabrani zadatak. Nakon završetka unosa svih podataka o akciji aplikacija prikazuje poruku da su svi podaci spremišteni i aktivnost završava.

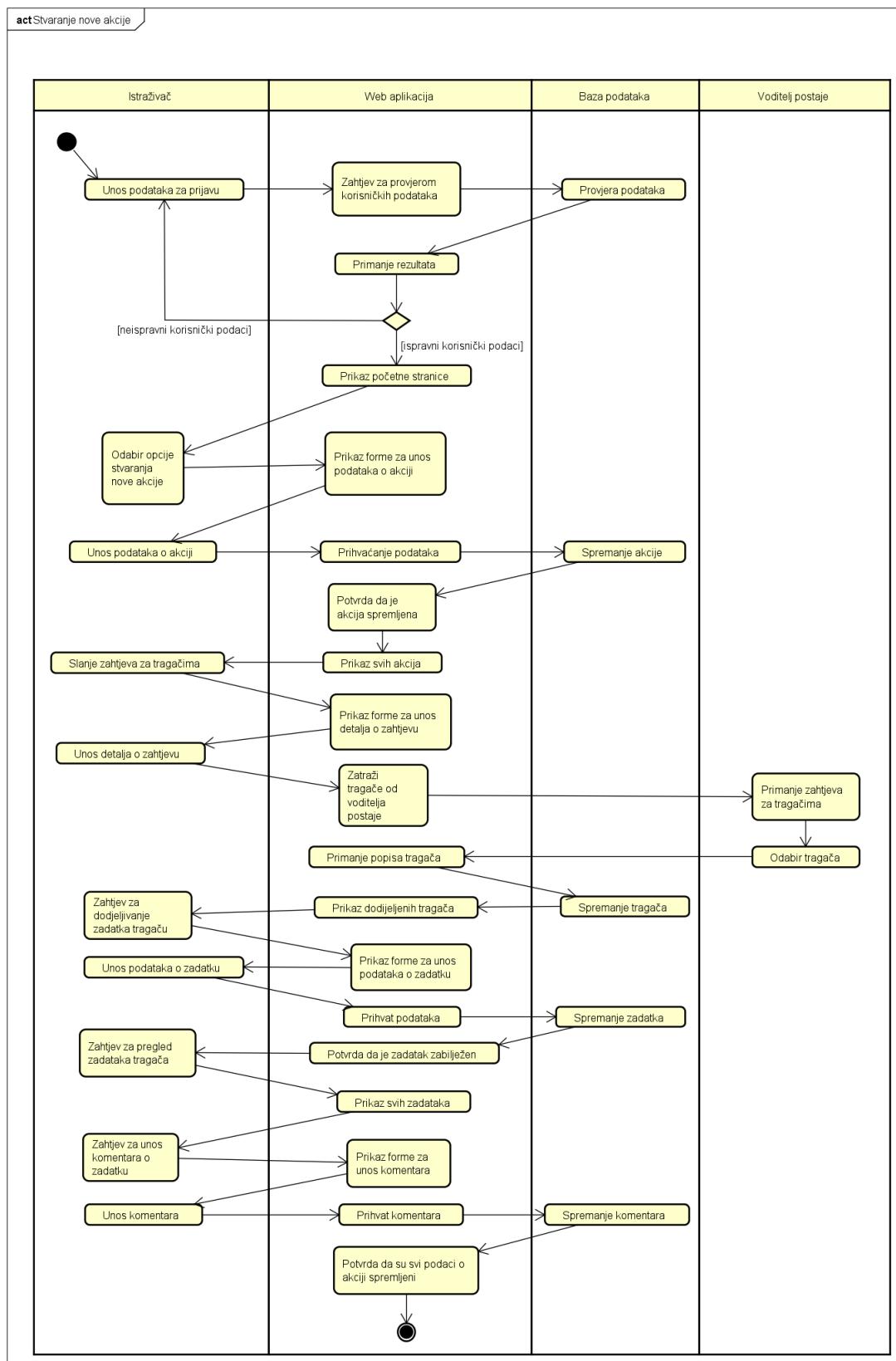


Figure 4.6: Dijagram aktivnosti

## 4.5 Dijagram komponenti

Dijagram komponenti prikazuje komponente u sustavu i njihovu organizaciju. Sustav se sastoji od tri glavne komponente: **Frontend web aplikacije**, **Backend web aplikacije** i **Baza podataka**. **Frontend web aplikacije** sastoji se od manjih komponenti: **ApiService.js**, **index.js**, **Admin**, **Assets**, **General**, **Login**, **Manager**, **Register**, **Researcher**, **Searcher**. Komponenta **ApiService.js** služi za komunikaciju s backendom pomoću HTTP metoda *get*, *post*, *put*, *delete*. Komponente **Admin**, **Assets**, **General**, **Login**, **Manager**, **Register**, **Researcher**, **Searcher** su skupovi react komponenti koji služe za prikaz za određenu svrhu. Npr. **Admin** ima komponente koje se prikazuju samo za admina, a **Manager** ima komponente koje se pokazuju samo za voditelja postaje. **Backend web aplikacije** sastoji se od komponenti **Controller**, **Service**, **Mapper**, **DTO**, **Repository**. **Controller** prima i odgovara na HTTP metode i poziva usluge u komponenti **Service**. Komponenta **Service** obavlja funkcije aplikacije, npr. registracija korisnika, dodavanje komentara, stvaranje postaje itd. Komponenta **Repository** služi za komunikaciju s **Bazom podataka**. **DTO** sadrži objekte s podacima koji se koriste u aplikaciji. **Mapper** služi za pretvorbu objekata iz **DTO** u normalne objekte, npr. pretvorbu **AnimalDto** u **Animal**.

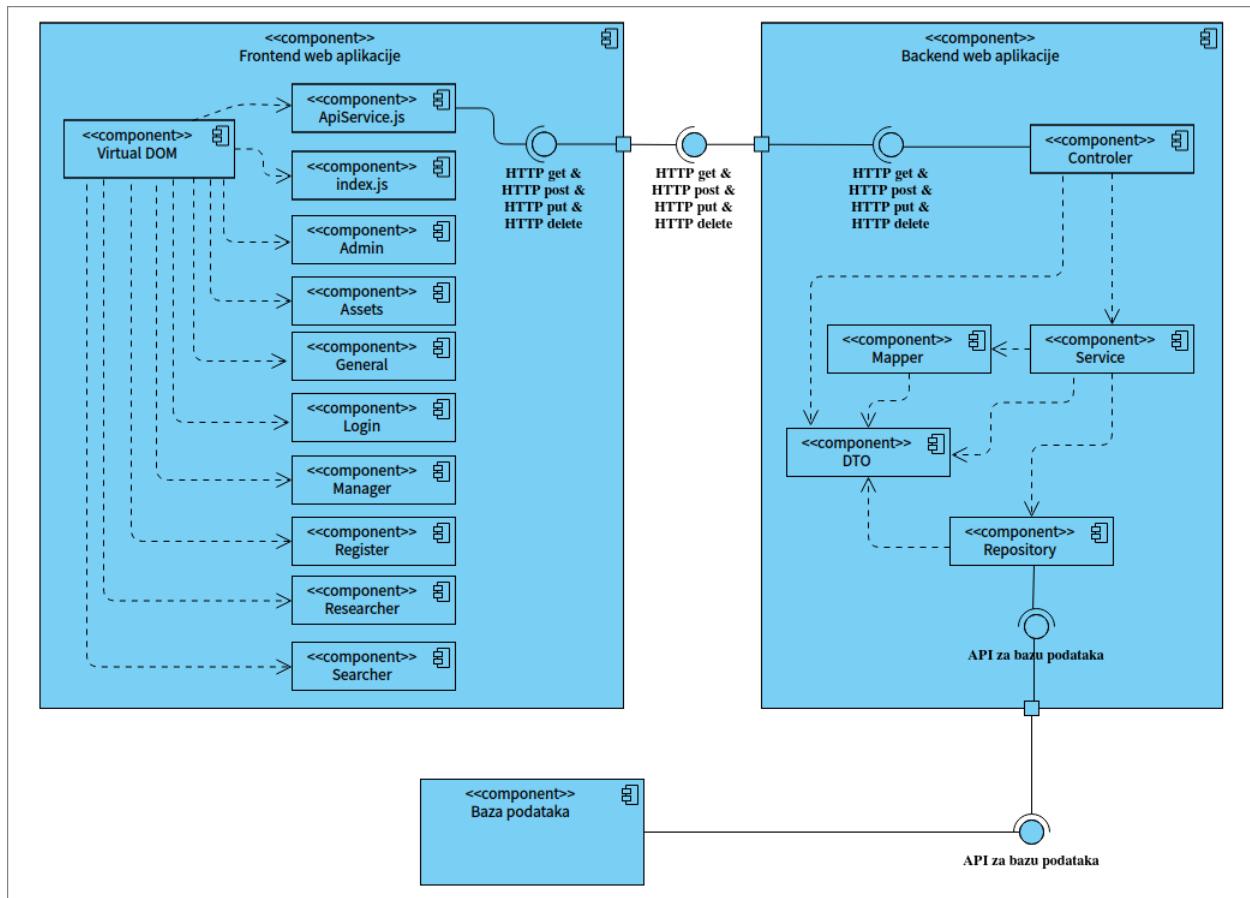


Figure 4.7: Dijagram komponenti

# 5. Implementacija i korisničko sučelje

## 5.1 Korištene tehnologije i alati

Tim je koristio WhatsApp.<sup>1</sup> i Notion.<sup>2</sup> za komunikaciju unutar tima. Za izradu UML dijagrama korišten je Astah Professional.<sup>3</sup>, dok je za upravljanje izvornim kodom odabran Git.<sup>4</sup>. Udaljeni repozitorij projekta dostupan je na web platformi GitHub.<sup>5</sup>.

Kao razvojna okruženja korišteni su Visual Studio Code.<sup>6</sup> i IntelliJ IDEA.<sup>7</sup>. Visual Studio Code je jednostavan uređivač koda s podrškom za razvojne operacije poput debugiranja, pokretanja zadataka i kontrolu verzija od tvrtke Microsoft, dok je IntelliJ popularno razvojno okruženje (IDE) razvijeno od strane tvrtke JetBrains. Namijenjeno je prvenstveno za rad s Java programskim jezikom.

Aplikacija je napisana koristeći radni okvir Spring Boot.<sup>8</sup> i jezik Java.<sup>9</sup> za izradu backenda te React.<sup>10</sup> i jezik JavaScript.<sup>11</sup> za izradu frontenda. React je besplatna i open-source JavaScript biblioteka za izgradnju korisničkih sučelja temeljenih na komponentama. Održava je Meta i zajednica pojedinačnih razvijatelja te tvrtki. Radni okvir Spring Boot je otvoreni, mikroservisni Java web okvir koji nudi Spring. Razvijen je kako bi pojednostavio proces izrade, konfiguracije i razmještanja Java aplikacija.

Baza podataka se nalazi na posluzitelju u oblaku Render.<sup>12</sup>.

---

<sup>1</sup><https://www.whatsapp.com/>

<sup>2</sup><https://www.notion.so/>

<sup>3</sup><https://astah.net/products/astah-professional/>

<sup>4</sup><https://git-scm.com/>

<sup>5</sup><https://github.com/>

<sup>6</sup><https://code.visualstudio.com/>

<sup>7</sup><https://www.jetbrains.com/idea/>

<sup>8</sup><https://spring.io/projects/spring-boot/>

<sup>9</sup><https://www.java.com/en/>

<sup>10</sup><https://react.dev/>

<sup>11</sup><https://www.javascript.com/>

<sup>12</sup><https://render.com/>

## 5.2 Ispitivanje programskog rješenja

### 5.3 Ispitivanje komponenti

Sveobuhvatan pregled našeg testiranja temeljio se na korištenju JUnit 5 za izvođenje unit testova. Većina testova odnosi se na ispitivanje ispravnosti podataka u Dto objektima (ActionDto, AnimalDto, AvailableSearcherDto, RequestDto i TaskDto). Jedan od ispitivanja služi za provjeru ispravnog formata maila, a jedan za provjeru ispravnosti koordinata. Zadnje od ispitivanja provjerava ispravnost funkcije koja služi za micanje tragača s akcije.

#### 5.3.1 Ispitivanje funkcije checkActionDto:

Provjerava se funkcionalnost metode *checkActionDto* u *researcherServiceJpa* servisu koja služi za provjeru ispravnosti podataka unutar *ActionDto* objekta.

##### **testCheckActionDtoValid:**

- U funkciju se šalje objekt *ActionDto* s ispravnim podacima.
- Očekuje se da neće doći do bacanja iznimke.

---

```
@Test
public void testCheckActionDtoValid() {
    ActionDto validActionDto = new ActionDto(1L, "ActionName",
                                             "ActionType", "LocationName");
    assertDoesNotThrow(() ->
        researcherServiceJpa.checkActionDto(validActionDto));
}
```

---

##### **testCheckActionDtoInvalidNull:**

- U funkciju se šalje objekt *ActionDto* s neispravnim podacima (*actionName* postavljen na *null*).
- Očekuje se da će doći do bacanja iznimke.

---

```
@Test
public void testCheckActionDtoInvalidNull() {
```

```
ActionDto invalidActionDto = new ActionDto(null, null,
    "ActionType", "LocationName");
assertThrows(IllegalArgumentException.class, () ->
    researcherServiceJpa.checkActionDto(invalidActionDto));
}
```

---

**testCheckActionDtoInvalidEmpty:**

- U funkciju se šalje objekt *ActionDto* s neispravnim podacima (*actionName* postavljen na "").
  - Očekuje se da će doći do bacanja iznimke.
- 

```
@Test
public void testCheckActionDtoInvalidEmpty() {
    ActionDto invalidActionDto = new ActionDto(1L, "",
        "ActionType", "LocationName");
    assertThrows(IllegalArgumentException.class, () ->
        researcherServiceJpa.checkActionDto(invalidActionDto));
}
```

---

**testCheckActionDtoInvalidWhitespace:**

- U funkciju se šalje objekt *ActionDto* s neispravnim podacima (*actionName* postavljen na " *ActionName*" ).
  - Očekuje se da će doći do bacanja iznimke.
- 

```
@Test
public void testCheckActionDtoInvalidWhitespace() {
    ActionDto invalidActionDto = new ActionDto(1L, " ActionName",
        "ActionType", "LocationName");
    assertThrows(IllegalArgumentException.class, () ->
        researcherServiceJpa.checkActionDto(invalidActionDto));
}
```

---

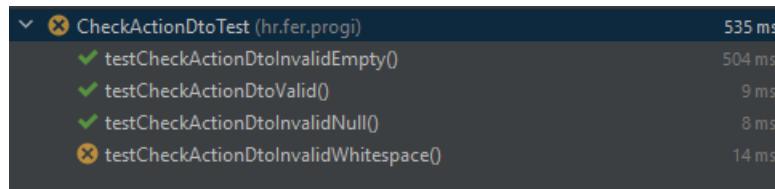


Figure 5.1: Rezultati ispitivanja

### 5.3.2 Ispitivanje funkcije `checkAnimalDto`:

Provjerava se funkcionalnost funkcije `checkAnimalDto` u `SearcherInTheFieldJpa` servisu koja služi za provjeru ispravnosti podataka unutar `AnimalDto` objekta.

#### `testCheckAnimalDtoValid`:

- U funkciju se šalje objekt `AnimalDto` s ispravnim podacima.
- Očekuje se da neće doći do bacanja iznimke.

---

```

    @Test
    public void testCheckAnimalDtoValid() {
        AnimalDto validAnimalDto = new AnimalDto();
        validAnimalDto.setAnimalName("Vuk");
        validAnimalDto.setBreed("Licki");
        validAnimalDto.setDescription("Sivi, brzi vuk");
        validAnimalDto.setCurrentPosition(Arrays.asList(1.0, 2.0));

        assertDoesNotThrow(() ->
            searcherJpa.checkAnimalDto(validAnimalDto));
    }

```

---

#### `testCheckAnimalDtoNull`:

- U funkciju se šalje objekt `AnimalDto` postavljen na `null`.
- Očekuje se da će doći do bacanja iznimke.

---

```

    @Test
    public void testCheckAnimalDtoNull() {
        AnimalDto nullAnimalDto = null;

```

```
IllegalArgumentException exception =
    assertThrows(IllegalArgumentException.class,
() -> searcherJpa.checkAnimalDto(nullAnimalDto));

assertEquals("AnimalDto is null", exception.getMessage());
}
```

---

#### **testCheckAnimalDtoMissingName:**

- U funkciju se šalje objekt *AnimalDto* s neispravnim podacima (nedostaje varijabla *AnimalName*).
- Očekuje se da će doći do bacanja iznimke.

```
@Test
public void testCheckAnimalDtoMissingName() {
    AnimalDto missingNameDto = new AnimalDto();
    missingNameDto.setBreed("Licki");
    missingNameDto.setDescription("Sivi, brzi vuk");
    missingNameDto.setCurrentPosition(Arrays.asList(1.0, 2.0));

    IllegalArgumentException exception =
        assertThrows(IllegalArgumentException.class,
() -> searcherJpa.checkAnimalDto(missingNameDto));

    assertEquals("AnimalName is missing", exception.getMessage());
}
```

---

#### **testCheckAnimalDtoMissingBreed:**

- U funkciju se šalje objekt *AnimalDto* s neispravnim podacima (nedostaje varijabla *Breed*).
- Očekuje se da će doći do bacanja iznimke.

```
@Test
public void testCheckAnimalDtoMissingBreed() {
    AnimalDto missingBreedDto = new AnimalDto();
    missingBreedDto.setAnimalName("Vuk");
    missingBreedDto.setDescription("Sivi, brzi vuk");
```

---

```

missingBreedDto.setCurrentPosition(Arrays.asList(1.0, 2.0));

IllegalStateException exception =
    assertThrows(IllegalStateException.class,
    () -> searcherJpa.checkAnimalDto(missingBreedDto));

assertEquals("Breed is missing", exception.getMessage());
}

```

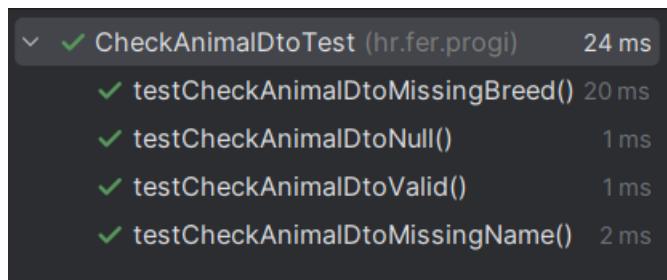


Figure 5.2: Rezultati ispitivanja

### 5.3.3 Ispitivanje funkcije `checkAvailableSearcherDto`:

Provjerava se funkcionalnost funkcije `checkAvailableSearcherDto` u `StationManagerJpa` servisu koja služi za provjeru ispravnosti podataka unutar `AvailableSearcherDto` objekta.

#### `testCheckAvailableSearcherDtoValid`:

- U funkciju se šalje objekt `AvailableSearcherDto` s ispravnim podacima.
- Očekuje se da neće doći do bacanja iznimke.

---

```

@Test
public void testCheckAvailableSearcherDtoValid() {
    AvailableSearcherDto validSearcherDto = new
        AvailableSearcherDto();
    validSearcherDto.setFirstName("John");
    validSearcherDto.setLastName("Doe");
    validSearcherDto.setQualification("Biologist");
    validSearcherDto.setCurrentPosition(Arrays.asList(1.0, 2.0));
}

```

```
    assertDoesNotThrow(() ->
        stationManagerJpa.checkAvailableSearcherDto(validSearcherDto));
}
```

---

**testCheckAvailableSearcherDtoNull:**

- U funkciju se šalje objekt *AvailableSearcherDto* postavljen na *null*.
- Očekuje se da će doći do bacanja iznimke.

```
@Test
public void testCheckAvailableSearcherDtoNull() {
    AvailableSearcherDto nullSearcherDto = null;

    IllegalArgumentException exception =
        assertThrows(IllegalArgumentException.class,
    () ->
        stationManagerJpa.checkAvailableSearcherDto(nullSearcherDto));

    assertEquals("AvailableSearcherDto is null",
        exception.getMessage());
}
```

---

**testCheckAvailableSearcherDtoMissingFirstName:**

- U funkciju se šalje objekt *AvailableSearcherDto* s neispravnim podacima (nedostaje varijabla *FirstName*).
- Očekuje se da će doći do bacanja iznimke.

```
@Test
public void testCheckAvailableSearcherDtoMissingFirstName() {
    AvailableSearcherDto missingFirstNameDto = new
        AvailableSearcherDto();
    missingFirstNameDto.setLastName("Doe");
    missingFirstNameDto.setQualification("Biologist");
    missingFirstNameDto.setCurrentPosition(Arrays.asList(1.0,
        2.0));
```

---

```

IllegalArgumentException exception =
    assertThrows(IllegalArgumentException.class,
() ->
    stationManagerJpa.checkAvailableSearcherDto(missingFirstNameDto));

assertEquals("First name is missing", exception.getMessage());
}

```

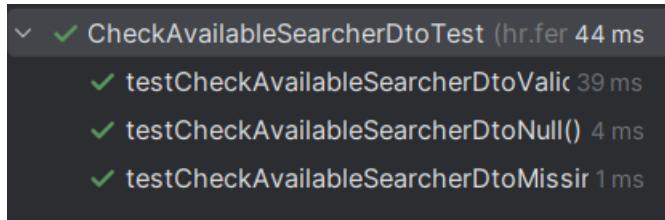


Figure 5.3: Rezultati ispitivanja

#### 5.3.4 Ispitivanje funkcije checkRequestDto:

Provjerava se funkcionalnost funkcije *checkRequestDto* u *ResearcherServiceJpa* servisu koja služi za provjeru ispravnosti podataka unutar *RequestDto* objekta.

##### **testCheckRequestDtoValid:**

- U funkciju se šalje objekt *RequestDto* s ispravnim podacima.
- Očekuje se da neće doći do bacanja iznimke.

```

@Test
public void testCheckRequestDtoValid() {
    RequestDto validRequestDto = new RequestDto();
    validRequestDto.setStationName("StationA");
    List<String> qualifications = Arrays.asList("QualificationA",
                                                "QualificationB");
    validRequestDto.setQualifications(qualifications);

    assertDoesNotThrow(() ->
        service.checkRequestDto(validRequestDto));
}

```

##### **testCheckRequestDtoNull:**

- U funkciju se šalje objekt *RequestDto* postavljen na *null*.
- Očekuje se da će doći do bacanja iznimke.

```
@Test
public void testCheckRequestDtoNull() {
    RequestDto nullRequestDto = null;

    InvalidArgumentException exception =
        assertThrows(InvalidArgumentException.class,
        () -> service.checkRequestDto(nullRequestDto));

    assertEquals("RequestDto is null", exception.getMessage());
}
```

---

#### **testCheckRequestDtoMissingStationName:**

- U funkciju se šalje objekt *RequestDto* s neispravnim podacima (nedostaje varijabla *StationName*).
- Očekuje se da će doći do bacanja iznimke.

```
@Test
public void testCheckRequestDtoMissingStationName() {
    RequestDto missingStationNameDto = new RequestDto();
    List<String> qualifications = Arrays.asList("QualificationA",
                                                "QualificationB");
    missingStationNameDto.setQualifications(qualifications);

    InvalidArgumentException exception =
        assertThrows(InvalidArgumentException.class,
        () -> service.checkRequestDto(missingStationNameDto));

    assertEquals("StationName is missing",
                exception.getMessage());
}
```

---

#### **testCheckRequestDtoMissingQualifications:**

- U funkciju se šalje objekt *RequestDto* s neispravnim podacima (nedostaje varijabla *Qualifications*).

- Očekuje se da će doći do bacanja iznimke.

```

    @Test
    public void testCheckRequestDtoMissingQualifications() {
        RequestDto missingQualificationsDto = new RequestDto();
        missingQualificationsDto.setStationName("StationA");

        InvalidArgumentException exception =
            assertThrows(InvalidArgumentException.class,
            () -> service.checkRequestDto(missingQualificationsDto));

        assertEquals("Qualifications is missing",
            exception.getMessage());
    }

```



Figure 5.4: Rezultati ispitivanja

### 5.3.5 Ispitivanje funkcije `checkTaskDto`:

Provjerava se funkcionalnost metode `checkTaskDto` u `taskServiceJpa` servisu koja služi za provjeru ispravnosti podataka unutar `TaskDto` objekta.

#### **testCheckTaskDtoValid:**

- U funkciju se šalje objekt `TaskDto` s ispravnim podacima.
- Očekuje se da neće doći do bacanja iznimke.

```

    @Test
    void testCheckTaskDtoValid() {
        TaskDto validTaskDto = new TaskDto(1L, 1L, new
            RouteWaypoints(), "TaskToDo", "TaskComment");
        assertDoesNotThrow(() ->
            taskServiceJpa.checkTaskDto(validTaskDto));
    }

```

```
}
```

---

**testCheckTaskDtoInvalidNull:**

- U funkciju se šalje objekt *TaskDto* s neispravnim podacima (*searcherId* postavljen na *null*).
- Očekuje se da će doći do bacanja iznimke.

---

```
@Test
public void testCheckActionDtoInvalidNull() {
    ActionDto invalidActionDto = new ActionDto(null, null,
        "ActionType", "LocationName");
    assertThrows(IllegalArgumentException.class, () ->
        researcherServiceJpa.checkActionDto(invalidActionDto));
}
```

---

**testCheckTaskDtoInvalidEmpty:**

- U funkciju se šalje objekt *TaskDto* s neispravnim podacima (*taskToDo* postavljen na "").
- Očekuje se da će doći do bacanja iznimke.

---

```
@Test
void testCheckTaskDtoInvalidEmpty() {
    TaskDto invalidTaskDtoEmpty = new TaskDto(null, 1L, new
        RouteWaypoints(), "TaskComment", "");
    assertThrows(IllegalArgumentException.class, () ->
        taskServiceJpa.checkTaskDto(invalidTaskDtoEmpty));
}
```

---

**testCheckTaskDtoInvalidWhitespace:**

- U funkciju se šalje objekt *TaskDto* s neispravnim podacima (*taskToDo* postavljen na " *TaskToDo*" ).
- Očekuje se da će doći do bacanja iznimke.

---

```

@Test
void testCheckTaskDtoInvalidWhitespace() {
    TaskDto invalidTaskDtoWhitespace = new TaskDto(null, 1L, new
        RouteWaypoints(), "TaskComment", "TaskToDo");
    assertThrows(IllegalArgumentException.class, () ->
        taskServiceJpa.checkTaskDto(invalidTaskDtoWhitespace));
}

```

---

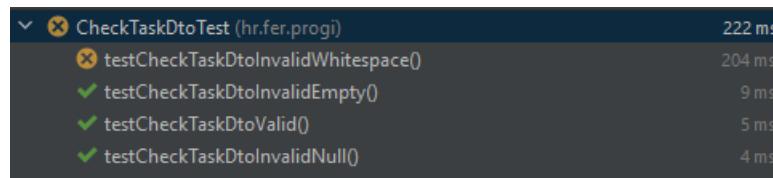


Figure 5.5: Rezultati ispitivanja

### 5.3.6 Ispitivanje funkcije emailRegexCheck:

Provjerava se funkcionalnost metode *emailRegexCheck* u *AuthenticationServiceJpa servisu* koja služi za provjeru ispravnog formata maila.

#### **emailRegexCheckValidEmailReturnsTrue:**

- U funkciju se šalje mail ispravnog formata.
- Očekuje se da funkcija vraća true.

---

```

@Test
void emailRegexCheck_ValidEmail_ReturnsTrue() {
    String validEmail = "test@example.com";
    boolean result =
        authenticationServiceJpa.emailRegexCheck(validEmail);
    assertTrue(result);
}

```

---

#### **testEmailRegexCheckInvalidEmail:**

- U funkciju se šalje mail neispravnog formata.
- Očekuje se da funkcija vraća false.

```
@Test  
public void testEmailRegexCheckInvalidEmail() {  
    String invalidEmail = "invalidemail";  
    boolean isValid =  
        authenticationServiceJpa.emailRegexCheck(invalidEmail);  
    assertFalse(isValid);  
}
```

---

**testEmailRegexCheckInvalidEmailEdgeCase:**

- U funkciju se šalje mail neispravnog formata.
  - Očekuje se da funkcija vraća false.
- 

```
@Test  
public void testEmailRegexCheckInvalidEmail_EdgeCase() {  
    String invalidEmail = "@.";  
    boolean isValid =  
        authenticationServiceJpa.emailRegexCheck(invalidEmail);  
    assertFalse(isValid);  
}
```

---

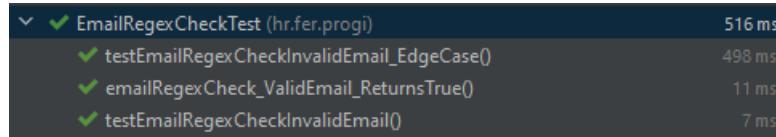


Figure 5.6: Rezultati ispitivanja

**5.3.7 Ispitivanje funkcije validateCoordinates:**

Provjerava se funkcionalnost funkcije *validateCoordinates* u *CommentServiceJpa* servisu koja služi za provjeru ispravnosti koordinata koje prima ta klasa preko objekta *MapCommentDto*.

**testValidateCoordinatesValid:**

- U funkciju se šalje objekt *MapCommentDto* s ispravnim podacima koordinata.
- Očekuje se da neće doći do bacanja iznimke.

```
@Test
public void testValidateCoordinatesValid() {
    MapCommentDto validMapCommentDto = new MapCommentDto();
    List<Double> validCoordinates = Arrays.asList(1.0, 2.0, 3.0);
    validMapCommentDto.setCoordinates(validCoordinates);

    assertDoesNotThrow(() ->
        CommentServiceJpa.validateCoordinates(validMapCommentDto));
}
```

---

**testValidateCoordinatesNullDto:**

- U funkciju se šalje objekt *MapCommentDto* postavljen na *null*.
  - Očekuje se da će doći do bacanja iznimke.
- 

```
@Test
public void testValidateCoordinatesNullDto() {
    IllegalArgumentException exception =
        assertThrows(IllegalArgumentException.class,
            () -> CommentServiceJpa.validateCoordinates(null));

    assertEquals("Coordinates cannot be null",
        exception.getMessage());
}
```

---

**testValidateCoordinatesNullCoordinate:**

- U funkciju se šalje objekt *MapCommentDto* s jednom koordinatom postavljenom na *null*.
  - Očekuje se da će doći do bacanja iznimke.
- 

```
@Test
public void testValidateCoordinatesNullCoordinate() {
    MapCommentDto nullCoordinateDto = new MapCommentDto();
    List<Double> coordinatesWithNull = Arrays.asList(1.0, null,
        3.0);
    nullCoordinateDto.setCoordinates(coordinatesWithNull);
```

---

```
IllegalArgumentException exception =
    assertThrows(IllegalArgumentException.class,
() ->
    CommentServiceJpa.validateCoordinates(nullCoordinateDto));

assertEquals("Coordinate value cannot be null",
    exception.getMessage());
}
```

---

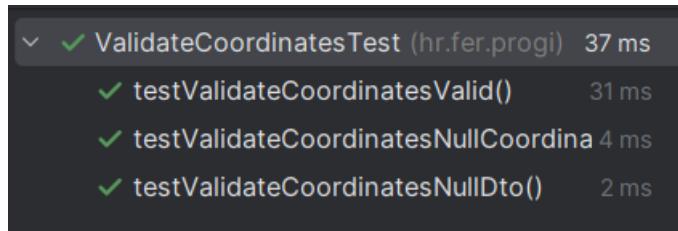


Figure 5.7: Rezultati ispitivanja

### 5.3.8 Ispitivanje funkcije `putRemoveFromAction`:

Provjerava se funkcionalnost funkcije `putRemoveFromAction` u `SearcherInTheFieldJpa` servisu koja služi za micanje tragača s akcije.

#### `testPutRemoveFromAction`:

- U funkciju se šalje `appUserId` i vraća se mock objekt `SearcherInTheField`.
- Očekuje se da neće doći do bacanja iznimke.

---

```
@Test
public void testPutRemoveFromAction() {
    Long appUserId = 123L;

    SearcherInTheField searcherMock =
        mock(SearcherInTheField.class);

    when(searcherInTheFieldJpa.findById(appUserId))
        .thenReturn(searcherMock);

    searcherInTheFieldJpa.putRemoveFromAction(appUserId);
```

```
    verify(searcherMock).setAction(null);
    verify(searcherMock).setCurrentPosition(null);

    verify(pastDataServiceJpa).searcherPositionSave(searcherMock);

    verify(searcherInTheFieldRepository).save(searcherMock);
}
```

---

**testPutRemoveFromAction\_AppUserNotFound:**

- U funkciju se šalje *appUserId* i vraća se objekt *SearcherInTheField* postavljen na *null*.
- Očekuje se da će doći do bacanja iznimke.

```
@Test
public void testPutRemoveFromAction_AppUserNotFound() {
    Long appUserId = 123L;

    when(searcherInTheFieldJpa.findByAppUserId(appUserId)).thenReturn(null);

    IllegalArgumentException exception =
        assertThrows(IllegalArgumentException.class,
        () -> searcherInTheFieldJpa.putRemoveFromAction(appUserId));

    assertEquals("SearcherInTheField is null",
        exception.getMessage());

    verifyNoMoreInteractions(searcherInTheFieldRepository,
        pastDataServiceJpa);
}
```

---

**testPutRemoveFromAction\_SaveFailure:**

- U funkciju se šalje *appUserId* i vraća se objekt *SearcherInTheField* postavljen na *null*. Provjerava se ponašanje funkcije kada spremanje u repozitorij nije uspješno tj. vraća *null*.
- Očekuje se da neće doći do bacanja iznimke.

```
@Test
public void testPutRemoveFromAction_SaveFailure() {
    Long appUserId = 123L;

    SearcherInTheField searcherMock =
        mock(SearcherInTheField.class);

    when(searcherInTheFieldJpa.findById(appUserId))
        .thenReturn(searcherMock);
    when(searcherInTheFieldRepository.save(searcherMock)).thenReturn(null);

    searcherInTheFieldJpa.putRemoveFromAction(appUserId);

    verify(pastDataServiceJpa).searcherPositionSave(searcherMock);

    verify(searcherInTheFieldRepository).save(searcherMock);
}
```

---

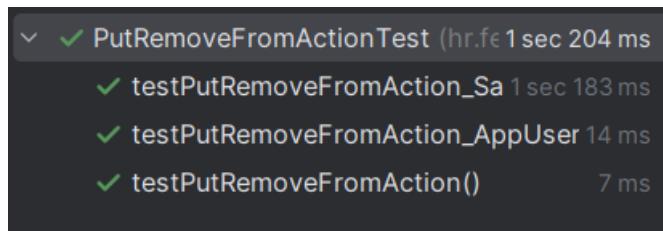


Figure 5.8: Rezultati ispitivanja

## 5.4 Ispitivanje sustava

Sustav smo detaljno testirali primjenom Selenium WebDriver dodatka unutar JUnit testova. Proveli smo dva ključna ispitivanja kako bismo osigurali funkcionalnost sustava: jedno je bilo usmjereni na proces prijave korisnika, dok je drugo obuhvatilo postupak registracije.

### 5.4.1 Ispitivanje Prijave Korisnika:

**testLoginValidCreds:**

- Unose se ispravne korisničke akreditacije (valjana e-mail adresa i lozinka).
  - Provjerava se preusmjeravanje nakon prijave.
  - Očekuje se da je korisnik uspješno preusmjeren na "dashboard".
- 

```
@Test
public void testLogin_ValidCreds(){
    ChromeOptions options = new ChromeOptions();
    options.addArguments("--remote-allow-origins=*");
    WebDriver driver = new ChromeDriver(options);
    driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
    driver.get("https://wildtrack-fe.onrender.com/login");

    WebElement element = driver.findElement(By.name("email"));
    element.sendKeys("admin@wildtrack.com");
    element = driver.findElement(By.name("password"));
    element.sendKeys("admin123");
    driver.findElement(By.cssSelector(".login-button")).click();

    try {
        Thread.sleep(5000);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
    String redirURL = driver.getCurrentUrl();
    boolean compRes = redirURL.contains("dashboard");

    assertEquals(compRes, true);

    driver.quit();
}
```

---

#### **testLoginInValidCreds:**

- Unose se nevaljane korisničke akreditacije (neispravna e-mail adresa i lozinka).
- Provjerava se preusmjeravanje nakon pokušaja prijave.
- Očekuje se da korisnik nije preusmjeren na "dashboard".

```
@Test
public void testLogin_InvalidCreds(){
    ChromeOptions options = new ChromeOptions();
    options.addArguments("--remote-allow-origins=*");
    WebDriver driver = new ChromeDriver(options);
    driver.manage().timeouts().implicitlyWait(10,
        TimeUnit.SECONDS);
    driver.get("https://wildtrack-fe.onrender.com/login");

    WebElement element = driver.findElement(By.name("email"));
    element.sendKeys("invalid");
    element = driver.findElement(By.name("password"));
    element.sendKeys("invalid");
    driver.findElement(By.cssSelector(".login-button")).click();

    String redirURL = driver.getCurrentUrl();
    boolean compRes = redirURL.contains("dashboard");

    assertEquals(compRes, false);

    driver.quit();
}
```

---

**testLoginEmptyEmail:**

- Ostavlja se polje za e-mail prazno.
  - Postavlja se lozinka na neku vrijednost.
  - Provjerava se preusmjerenje nakon pokušaja prijave.
  - Očekuje se da korisnik nije preusmjeren na "dashboard".
- 

```
@Test
public void testLogin_emptyEmail() {
    ChromeOptions options = new ChromeOptions();
    options.addArguments("--remote-allow-origins=*");
    WebDriver driver = new ChromeDriver(options);
    driver.manage().timeouts().implicitlyWait(10,
        TimeUnit.SECONDS);
```

---

```
driver.get("https://wildtrack-fe.onrender.com/login");

WebElement emailElement =
    driver.findElement(By.name("email"));
emailElement.sendKeys("");

WebElement passwordElement =
    driver.findElement(By.name("password"));
passwordElement.sendKeys("somePassword");

driver.findElement(By.cssSelector(".login-button")).click();

String redirURL = driver.getCurrentUrl();
boolean compRes = redirURL.contains("dashboard");

assertEquals(compRes, false);

driver.quit();
}
```

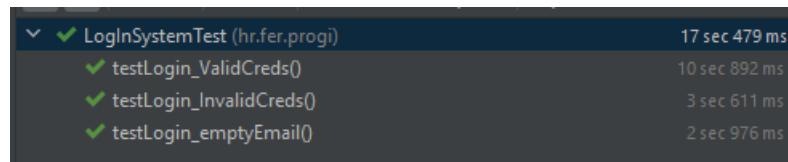


Figure 5.9: Rezultati ispitivanja

Drugo ispitivanje bilo je usmjereni na proces registracije novog korisnika. Kroz Selenium WebDriver testove, simulirali smo korisničko sučelje kako bismo provjerili ispravnost postupka registracije. Cilj je bio osigurati da sustav adekvatno obrađuje unos novih korisničkih podataka te da novi korisnik bude uspješno registriran u sustavu.

*Ključne Točke Ispitivanja:*

- Popunjavanje obrasca za registraciju s valjanim podacima.
- Klik na gumb za registraciju.
- Provjera uspješne registracije i ispravnosti prelaska na stranicu nakon registracije.

### 5.4.2 Ispitivanje Registracije:

#### testRegistrationInvalidCreds:

- Unosi se email u neispravnom formatu.
- Provjerava se ispisani tekst na stranici.
- Očekuje se je da je na stranici ispisani tekst *email not valid*.

---

```
@Test
public void testRegistration_InvalidCreds(){
    ChromeOptions options = new ChromeOptions();
    options.addArguments("--remote-allow-origins=*");
    WebDriver driver = new ChromeDriver(options);
    driver.manage().timeouts().implicitlyWait(10,
        TimeUnit.SECONDS);
    driver.get("https://wildtrack-fe.onrender.com/register");

    WebElement firstNameInput =
        driver.findElement(By.id("register-firstName"));
    firstNameInput.sendKeys("test");

    WebElement lastNameInput =
        driver.findElement(By.id("register-lastName"));
    lastNameInput.sendKeys("test");

    WebElement usernameInput =
        driver.findElement(By.id("register-username"));
    usernameInput.sendKeys("test");

    WebElement emailInput =
        driver.findElement(By.id("register-email"));
    emailInput.sendKeys("test");

    WebElement passwordInput =
        driver.findElement(By.id("register-password"));
    passwordInput.sendKeys("test");
```

```
WebElement tragacRadioButton =
    driver.findElement(By.id("register-searcher"));
tragacRadioButton.click();

WebElement fileInput =
    driver.findElement(By.id("photo-upload"));
fileInput.sendKeys("D:\\FER-predmeti\\zoolanders\\zoolanders\\be\\src
\\test\\resources\\profile.jpg");

WebElement registerButton =
    driver.findElement(By.cssSelector(".register-button"));
registerButton.click();

try {
    Thread.sleep(10000);
} catch (InterruptedException e) {
    e.printStackTrace();
}

WebElement errorElement =
    driver.findElement(By.className("register-error"));
String actualErrorText = errorElement.getText();
String expectedErrorText = "email not valid";

assertEquals(expectedErrorText, actualErrorText);

driver.quit();
}
```

---

**testRegistrationEmailAllReadyTaken:**

- Unose se korisnički podaci u ispravnom formatu.
- Provjerava se ispisan tekst na stranici.
- Očekuje se je da je na stranici ispisan tekst *email already taken*.

---

```
@Test
public void testRegistration_InvalidCreds(){
    ChromeOptions options = new ChromeOptions();
```

```
options.addArguments("--remote-allow-origins=*");
WebDriver driver = new ChromeDriver(options);
driver.manage().timeouts().implicitlyWait(10,
    TimeUnit.SECONDS);
driver.get("https://wildtrack-fe.onrender.com/register");

WebElement firstNameInput =
    driver.findElement(By.id("register-firstName"));
firstNameInput.sendKeys("test");

WebElement lastNameInput =
    driver.findElement(By.id("register-lastName"));
lastNameInput.sendKeys("test");

WebElement usernameInput =
    driver.findElement(By.id("register-username"));
usernameInput.sendKeys("test");

WebElement emailInput =
    driver.findElement(By.id("register-email"));
emailInput.sendKeys("test");

WebElement passwordInput =
    driver.findElement(By.id("register-password"));
passwordInput.sendKeys("test");

WebElement tragacRadioButton =
    driver.findElement(By.id("register-searcher"));
tragacRadioButton.click();

WebElement fileInput =
    driver.findElement(By.id("photo-upload"));
fileInput.sendKeys("D:\\FER-predmeti\\zoolanders\\zoolanders\\be\\src
\\test\\resources\\profile.jpg");

WebElement registerButton =
    driver.findElement(By.cssSelector(".register-button"));
registerButton.click();
```

```
try {
    Thread.sleep(10000);
} catch (InterruptedException e) {
    e.printStackTrace();
}
WebElement errorElement =
    driver.findElement(By.className("register-error"));
String actualErrorText = errorElement.getText();
String expectedErrorText = "email not valid";

assertEquals(expectedErrorText, actualErrorText);

driver.quit();
}
```

---

**testRegistrationValidCreds:**

- Unose se korisnički podaci u ispravnom formatu.
- Provjerava se ispisani tekst na stranici.
- Očekuje se je da je na stranici ispisani tekst *Registracija uspješna!*.

---

```
@Test
public void testRegistration_ValidCreds(){
    ChromeOptions options = new ChromeOptions();
    options.addArguments("--remote-allow-origins=*");
    WebDriver driver = new ChromeDriver(options);
    driver.manage().timeouts().implicitlyWait(10,
        TimeUnit.SECONDS);
    driver.get("https://wildtrack-fe.onrender.com/register");

    WebElement firstNameInput =
        driver.findElement(By.id("register-firstName"));
    firstNameInput.sendKeys("test");

    WebElement lastNameInput =
        driver.findElement(By.id("register-lastName"));
```

```
lastNameInput.sendKeys("test");

WebElement usernameInput =
    driver.findElement(By.id("register-username"));
usernameInput.sendKeys("test");

WebElement emailInput =
    driver.findElement(By.id("register-email"));
emailInput.sendKeys("test1@test.com");

WebElement passwordInput =
    driver.findElement(By.id("register-password"));
passwordInput.sendKeys("test");

WebElement tragacRadioButton =
    driver.findElement(By.id("register-searcher"));
tragacRadioButton.click();

WebElement fileInput =
    driver.findElement(By.id("photo-upload"));
fileInput.sendKeys("D:\\FER-predmeti\\zoolanders\\zoolanders\\be\\src
\\test\\resources\\profile.jpg");

WebElement registerButton =
    driver.findElement(By.cssSelector(".register-button"));
registerButton.click();

try {
    Thread.sleep(10000);
} catch (InterruptedException e) {
    e.printStackTrace();
}

WebElement errorElement =
    driver.findElement(By.className("register-success"));
String actualText = errorElement.getText();
String expectedText = "Registracija uspjesna!";

assertEquals(expectedText, actualText);
```

```
    driver.quit();  
}
```

---

▼ ✘ RegistrationSystemTest (hr.fer.progi)	49 sec 221 ms
✓ testRegistration_EmailAllReadyTaken()	19 sec 520 ms
✗ testRegistration_InvalidCreds()	13 sec 575 ms
✓ testRegistration_ValidCreds()	16 sec 126 ms

Figure 5.10: Rezultati ispitivanja

## 5.5 Dijagram razmještaja

Specifikacijski dijagram razmještaja prikazuje pregled implementacije artefakata bez upućivanja na specifične slučajeve artefakata ili čvorova. Korisnici pristupaju aplikaciji, koja se nalazi na računalu poslužitelja, putem web preglednika. Komunikacija između računala korisnika (tragača, istraživača, administratora) i poslužitelja ostvaruje se putem HTTP veze.

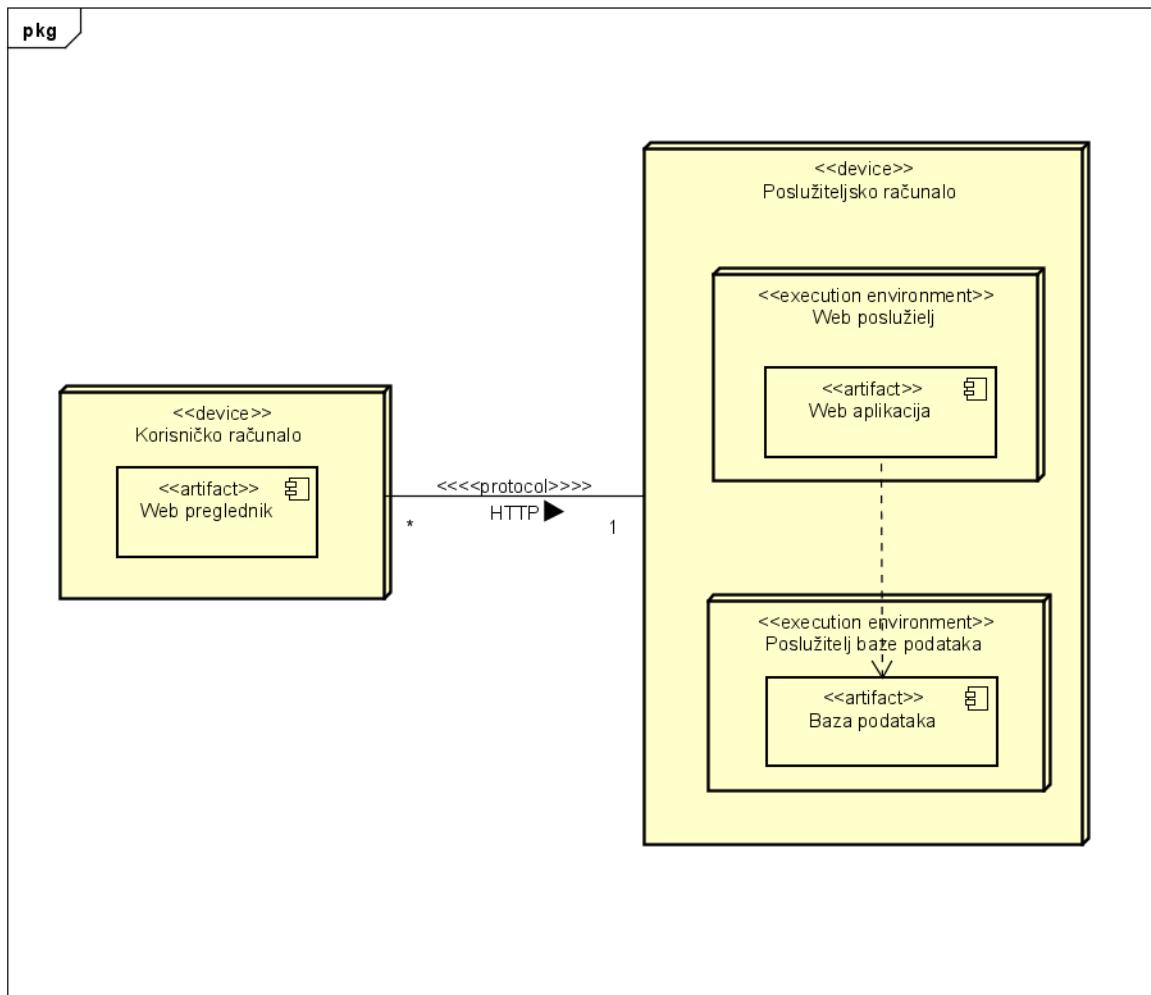


Figure 5.11: Specifikacijski dijagram razmještaja

## 5.6 Upute za puštanje u pogon

U ovom poglavlju prikazat ćemo kako se provodi pokretanje aplikacije na lokalnom računalu.

### Instalacija poslužitelja baze podataka

Potrebno je preuzeti PostgreSQL bazu podataka za operacijski sustav Windows.

Instalacija se može provesti pomoću sljedećeg linka: <https://www.postgresql.org/download/>.

Prilikom instalacije, zapamtite koji ste port odabrali za pristup bazi podataka, pošto će nam to biti potrebno kasnije (preporučeni port je 5432). Nakon uspješno provedene instalacije, PostgreSQL baza podataka bit će dostupna na vašem računalu.

### Konfiguracija baze podataka

Za daljnju konfiguraciju baze podataka i ostatak projekta koristit ćemo IntelliJ IDE. Prije svega, s GitHub repozitorija <https://github.com/filip-ljubotina/zoolanders.git> preuzet ćemo projekt. Zatim, u IntelliJ-u, u gornjem lijevom kutu odaberemo File ->Open... te se pozicioniramo u bazni direktorij projekta.

Sljedeći korak, kako je prikazano na slici, je odabrati izvor podataka (engl. Data Source).

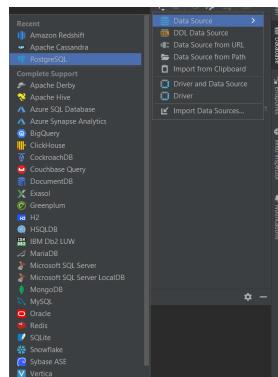


Figure 5.12: Odabir izvora podataka

Nakon odabira PostgreSQL-a kao izvora podataka, potrebno je unijeti podatke o bazi podataka. Za korisničko ime (user) koristimo "postgres", dok su port i lozinka (password) isti kao što smo ih postavili tijekom instalacije.

### Punjjenje baze podataka

Nakon konfiguracije baze podataka, potrebno je napuniti ju. Ovaj korak također obavljamo unutar IntelliJ-a tako što desnim klikom na postreg bazu odaberemo opciju "Run SQL Script", kao što je prikazano na slici. Zatim odaberemo skriptu koja se nalazi u direktoriju projekta. Nakon pokretanja, baza će biti napunjena

početnim podacima.

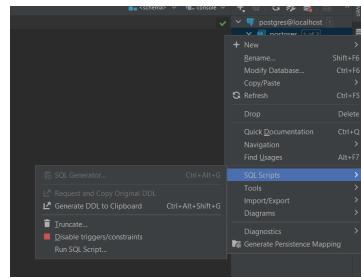


Figure 5.13: Pokretanje SQL skripte

### Konfiguracija frontend-a i backend-a

Nakon postavljanja baze podataka, slijedi konfiguracija backend-a. Potrebno je otvoriti "application.yaml" datoteku unutar "resources" paketa te postaviti korisničko ime, lozinku i URL podatkovnog izvora (datasource) tako da budu isti kao što su bili postavljeni tijekom konfiguracije datasourca.

Za konfiguraciju frontend-a, potrebno je otvoriti "fe" direktorij, zatim unutar "services" ->"ApiService" postaviti "baseURL" na "http://localhost:8080". Nakon toga, u naredbenom retku (Command Prompt-u), potrebno se pozicionirati unutar "fe" direktorija i pokrenuti naredbu "npm install".

### Pokretanje aplikacije

Za pokretanje backenda u IntelliJ-eju je potrebno pokrenuti WildtrackApplication, a za frontend u terminalu je potrebnu pokrenuti naredbu npm start.



## 6. Zaključak i budući rad

Cilj našeg projekta bio je razviti web stranicu koja omogućava lakšu koordinaciju i praćenje životinja u divljini. Aplikacija omogućuje različitim korisnicima da sudjeluju u akcijama, prate pozicije životinja te, u nekim ulogama, koordiniraju tragače u istraživačkim projektima.

Prva faza projekta bila je okupljanje tima te upoznavanje s dobivenim materijalima i zadatkom. Nakon prvog upoznavanja, počelo je dodjeljivanje uloga i rad na projektu. Kvalitetno odraćena organizacija uz pomoć koje je svaki član znao svoj posao značajno je olakšala i ubrzala rad na sadašnjem projektu. Izrađeni dijagrami i obrasci na početku rada bili su od velike pomoći pri pisanju koda i razvijanju backenda i frontenda.

Druga faza, iako kraća, bila je intenzivnija. Pri dodavanju različitih funkcionalnosti, trebali smo se samostalno upoznavati s odabranim alatima kako bismo ispunili ciljeve projekta. Uz to, dokumentirali smo ostale UML dijagrame i dovršili dokumentaciju koja će omogućiti budućim korisnicima lakše snalaženje.

Tijekom razvoja prepoznati su razni tehnički izazovi. Implementacija interaktivne karte koja prati lokaciju životinja i tragača predstavljala je kompleksan zadatak, zahtijevajući dobro poznavanje i integraciju s geolokacijskim rutama te vizualizacijom ruta. Još jedan izazov bio je sustav odobravanja registracija od strane administratora, što je zahtijevalo sigurnosne i administrativne funkcionalnosti.

Unatoč izazovima, uspješno je implementirana interaktivna karta koristeći se tehnologijama poput Reacta i Leafleta. Sustav odobravanja registracija također je uspješno implementiran, pružajući siguran pristup. Kroz projekt stečena su znanja o radu s geolokacijskim podacima, upravljanju korisnicima i implementaciji kompleksnih funkcionalnosti unutar web stranice.

Zadatak je proveden unutar planiranog vremenskog okvira, ostvarujući ključne funkcionalnosti. Tehnički izazovi bili su prepoznati i riješeni. Stečena su iskustva u radu s geolokacijskim tehnologijama i implementaciji složenih sustava.

Projekt je bio izazovan, ali i izrazito koristan za tim. Ovaj projekt je pružio priliku za razvoj vještina u području web aplikacija.

# Popis literature

## **Kontinuirano osvježavanje**

*Popisati sve reference i literaturu koja je pomogla pri ostvarivanju projekta.*

1. Programsko inženjerstvo, FER ZEMRIS, <http://www.fer.hr/predmet/proinz>
2. The Unified Modeling Language, <https://www.uml-diagrams.org/>
3. Astah Community, <http://astah.net/editions/uml-new>
4. National Geographic <https://www.nationalgeographic.com/culture/article/where-animals-go-tracking-maps>
5. USGC <https://alaska.usgs.gov/products/data/tracking/tracking.php?groupid=4>
6. designmodo <https://designmodo.com/create-css3-login-form/>
7. designmodo <https://designmodo.com/create-css3-login-form/>
8. Volunteer Match <https://www.volunteermatch.org/>
9. Movebank <https://www.movebank.org/>
10. Wildme <https://www.wildme.org/>
11. iNaturalist <https://www.inaturalist.org/>
12. Wildbook for Iberian Lynx <https://lynx.wildbook.org/>

# Indeks slika i dijagrama

2.1	Primjer login screena . . . . .	6
2.2	Primjer registracije . . . . .	7
2.3	Primjer biranja uloge voditelja i unos željene postaje . . . . .	7
2.4	Primjer biranja osposobljenja tragača (1) . . . . .	8
2.5	Primjer biranja osposobljenja tragača (2) . . . . .	9
2.6	Primjer kartografskog prikaza trenutnih lokacija tragača i životinja .	10
2.7	Primjer kartografskog prikaza povijesnog kretanja tragača i životinja	10
2.8	Karta za praćenje životinja - Movebank . . . . .	12
2.9	Wildbook za Pirenejskog risa . . . . .	12
2.10	Stranica iNaturalist . . . . .	12
3.1	Dijagram obrasca uporabe, funkcionalnost korisnika i administratora	26
3.2	Dijagram obrasca uporabe, voditelja postaje, tragača i istraživača .	27
3.3	Sekvencijski dijagram za UC9 . . . . .	29
3.4	Sekvencijski dijagram za UC15 . . . . .	31
3.5	Sekvencijski dijagram za UC20 . . . . .	33
4.1	Dijagram baze podataka . . . . .	42
4.2	Dijagram razreda - dio Controllers . . . . .	43
4.3	Dijagram razreda - dio DTO . . . . .	44
4.4	Dijagram razreda - dio Models . . . . .	45
4.5	Dijagram stanja . . . . .	46
4.6	Dijagram aktivnosti . . . . .	48
4.7	Dijagram komponenti . . . . .	50
5.1	Rezultati ispitivanja . . . . .	54
5.2	Rezultati ispitivanja . . . . .	56
5.3	Rezultati ispitivanja . . . . .	58
5.4	Rezultati ispitivanja . . . . .	60
5.5	Rezultati ispitivanja . . . . .	62
5.6	Rezultati ispitivanja . . . . .	63

5.7 Rezultati ispitivanja . . . . .	65
5.8 Rezultati ispitivanja . . . . .	67
5.9 Rezultati ispitivanja . . . . .	70
5.10 Rezultati ispitivanja . . . . .	76
5.11 Specifikacijski dijagram razmještaja . . . . .	77
5.12 Odabir izvora podataka . . . . .	78
5.13 Pokretanje SQL skripte . . . . .	79
6.1 Promjene na grani main . . . . .	88
6.2 Promjeve na grani DEV-be . . . . .	88
6.3 Promjeve na grani DEV-fe . . . . .	88
6.4 Promjeve na grani devdoc . . . . .	88

# Dodatak: Prikaz aktivnosti grupe

## Dnevnik sastajanja

*Kontinuirano osvježavanje*

### 1. sastanak

- Datum: October 19, 2023
- Prisustvovali: F.Ljubotina, M.Pavić, A.Vuksanović, L.Ćorić, K.Klarić, M.Breznički-Herceg, N.Milin
- Teme sastanka:
  - Upoznavanje tima
  - Izrada okvirnog plana rada

## Tablica aktivnosti

### *Kontinuirano osvježavanje*

*Napomena: Doprinose u aktivnostima treba navesti u satima po članovima grupe po aktivnosti.*

	Filip Ljubotina	Marko Pavić	Ana Vuksanović	Lara Čorić	Katarina Klarić	Mihael Breznički-Herceg	Noa Milin
Upravljanje projektom	8						
Opis projektnog zadatka				6			
Funkcionalni zahtjevi		2					
Opis pojedinih obrazaca		6	2				
Dijagram obrazaca	5						
Sekvencijski dijagrami			5				
Opis ostalih zahtjeva					1		
Arhitektura i dizajn sustava		1.5					
Baza podataka						7	
Dijagram razreda	2				6		6
Dijagram stanja					8		
Dijagram aktivnosti					7		
Dijagram komponenti						10	
Korištene tehnologije i alati		2					
Ispitivanje programskog rješenja	10	10					

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

	Filip Ljubotina	Marko Pavić	Ana Vuksanović	Lara Čorić	Katarina Klarić	Mihael Breznički-Herceg	Noa Milin
Dijagram razmještaja		1					
Upute za puštanje u pogon	2						
Dnevnik sastajanja	0.01						
Zaključak i budući rad				8			
Popis literature	0.1						
<i>izrada back end-a 1. dio</i>	20						
<i>izrada front end-a 1. dio</i>	5		15				
<i>deploy-anje aplikacije 1. dio</i>	5						
<i>izrada back end-a 2. dio</i>	80	40					
<i>izrada front end-a 2. dio</i>	25		80	40			
<i>deploy-anje aplikacije 2. dio</i>	2						
<i>ispravak dijagrama razreda</i>							8
<i>ispravak opisa baze podataka</i>	3					8	
<i>ispravak funkcionalnih zahtjeva</i>	0.5		3			8	

## Dijagrami pregleda promjena



Figure 6.1: Promjene na grani main

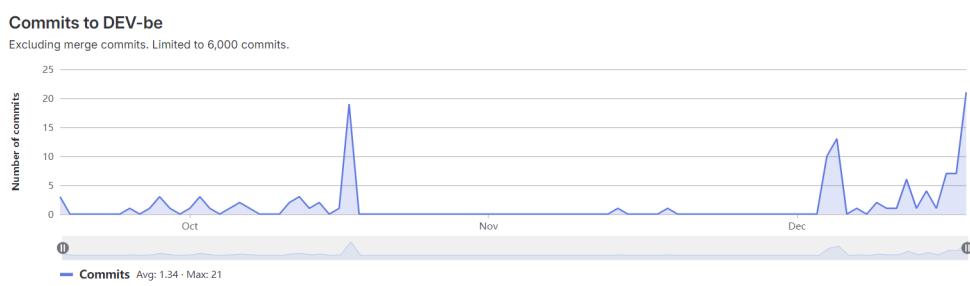


Figure 6.2: Promjeve na grani DEV-be

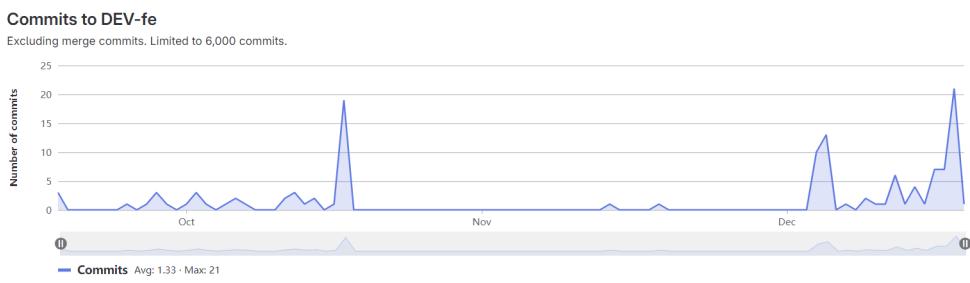


Figure 6.3: Promjeve na grani DEV-fe

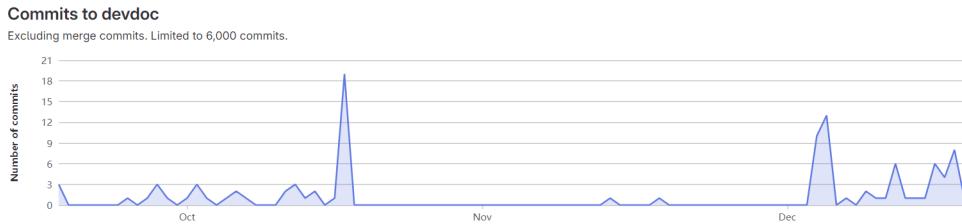


Figure 6.4: Promjeve na grani devdoc