

Programsko inženjerstvo

Ak. god. 2022./2023.

Čuvari pasa

Dokumentacija, Rev. 2

Grupa: Primavara

Voditelj: Antonio Lukić

Datum predaje: 18. studenoga 2022.

Nastavnik: Antea Šetka

Sadržaj

1 Dnevnik promjena dokumentacije	3
2 Opis projektnog zadatka	6
3 Specifikacija programske potpore	10
3.1 Funkcionalni zahtjevi	10
3.1.1 Obrasci uporabe	12
3.1.2 Sekvencijski dijagrami	25
3.2 Ostali zahtjevi	29
4 Arhitektura i dizajn sustava	30
4.1 Baza podataka	31
4.1.1 Opis tablica	31
4.1.2 Dijagram baze podataka	38
4.2 Dijagram razreda	39
4.3 Dijagram stanja	41
4.4 Dijagram aktivnosti	43
4.5 Dijagram komponenti	45
5 Implementacija i korisničko sučelje	47
5.1 Korištene tehnologije i alati	47
5.2 Ispitivanje programskog rješenja	50
5.2.1 Ispitivanje komponenti	50
5.2.2 Ispitivanje sustava	54
5.3 Dijagram razmještaja	62
5.4 Upute za puštanje u pogon	63
5.4.1 Konfiguracija poslužitelja baze podataka	63
5.4.2 Konfiguracija backenda	63
5.4.3 Konfiguracija frontenda	64
6 Zaključak i budući rad	65

Popis literature	67
Indeks slika i dijagrama	68
Dodatak: Prikaz aktivnosti grupe	69

1. Dnevnik promjena dokumentacije

Rev.	Opis promjene/dodatka	Autori	Datum
0.1	Napravljen predložak	Mario Petek	30.10.2022.
0.2	Napisan opis projektnog zadatka	Ivan Kuzmić, Mario Petek	31.10.2022.
0.3	Napisani funkcijski zahtjevi	Ivan Kuzmić, Mario Petek	01.11.2022.
0.4.1	Napisan dio obrazaca uporabe	Ivan Kuzmić, Mario Petek	01.11.2022.
0.4.2	Ažuriranje obrazaca uporabe	Antonio Lukić, Mario Petek	02.11.2022.
0.4.3	Napisan ostatak obrazaca uporabe i popravljivanje grešaka	Ivan Kuzmić	05.11.2022.
0.4.4	Dodavanje potrebnih informacija obrascima uporabe	Ivan Kuzmić	11.11.2022.

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

Rev.	Opis promjene/dodatka	Autori	Datum
0.5	Dodavanje dijagrama obrazaca uporabe i sekvencijskih dijagrama	Ivan Kuzmić, Mario Petek	14.11.2022.
0.6	Dodavanje ostalih zahtjeva, arhitekture i opisa baze podataka	Ivan Kuzmić	14.11.2022.
0.7	Dodavanje dijagrama razreda	Antonio Lukić	16.11.2022.
1.0	Verzija samo s bitnim dijelovima za 1. ciklus	Ivan Kuzmić	18.11.2022.
1.1	Izmjena napomenutih dijelova iz prvog ocjenjivanja	Ivan Kuzmić, Mario Petek	03.01.2023.
1.2	Dodavanje ostatka dijagrama razreda	Antonio Lukić	04.01.2023.
1.3	Dodavanje dijagrama aktivnosti	Antonio Lukić	07.01.2023.
1.4.1	Dodavanje prve verzije dijagrama stanja	Mario Petek	09.01.2023.
1.4.2	Dodavanje konačne verzije dijagrama stanja	Mario Petek	10.01.2023.
1.5	Dodavanje korištenih tehnologija i alata	Mario Petek	10.01.2023.
1.6	Dodavanje dijagrama razmještaja	Ivan Kapusta	10.01.2023.

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

Rev.	Opis promjene/dodatka	Autori	Datum
1.7	Dodavanje zaključka	Mario Pe- tek	12.01.2023.
1.8	Puštanje aplikacije u pogon	Sven Leko, Ivan Kapusta	12.01.2023.
1.9	Dodavanje dijagrama komponenti	Mario Pe- tek	13.01.2023.
1.10	Ispitivanje sustava	Lovro Malojčić, Eugen Preglej	13.01.2023.
2.0	Konačni tekst predloška dokumentacije		

2. Opis projektnog zadatka

U današnje vrijeme veliki broj ljudi za svog ljubimca odabire psa, koji većinu vremena provodi uz svog vlasnika. No, što ako vlasnik ima neodgodivu obavezu na koju nikako ne može povesti svog psa i treba biti odsutan neko vrijeme? Kome se obratiti za pomoć, ako mu je nitko koga zna nije u mogućnosti ponuditi? Negdje u blizini se sigurno krije osoba koja bi rado preuzela privremenu brigu, no kako do nje?

Kako bi vlasnici mogli obavljati svoje obaveze bezbrižno znajući da je njihov pas zbrinut, a s druge strane ljudi koji bi rado prošetali ili nahranili psa i privremeno se igrali s njim mogli ispuniti vrijeme na koristan i zabavan način, potrebna je aplikacija preko koje bi se oni mogli povezati i pronaći rješenje za svoje brige.

Glavni cilj ovog projekta je stvaranje aplikacije "*Čuvari pasa*" koja će vlasniku pasa pomoći u pronalasku osobe koja najbolje odgovara za privremenu brigu (od nekoliko sati do nekoliko tjedana ili mjeseci) o njegovom ljubimcu.

Ukratko, ideja je da vlasnik pasa u aplikaciji predaje zahtjev kojim traži osobu za privremenu brigu o njegovom psu. S druge strane, osobe koje su voljne čuvati nekog psa na određeno vrijeme u aplikaciji predaju oglas u kojem navode uvjete čuvanja/brige. Na temelju zahtjeva vlasnika i uvjeta čuvara, aplikacija može pronaći najbolji odabir, koji su obje strane u mogućnosti prihvatiti ili odbiti, pa u slučaju odbijanja pretražujući ostale zahtjeve/oglasne pronaći drugi odabir.

Prilikom pokretanja aplikacije prikazuje se početna stranica na kojoj se nalazi nekoliko informacija o samoj aplikaciji te navigacijski izbornik putem kojeg se može pristupiti registraciji i prijavi u sustav te objavljenim zahtjevima i oglasima. Pregledavanje tih zahtjeva i oglasa omogućeno je i neregistriranim korisnicima, ali za korištenje ostalih funkcionalnosti, kao što su objava vlastitih zahtjeva/oglasa te stupanje u kontakt s objavljiivačima, potrebna je registracija ili u slučaju već postojećeg računa, prijava. Registrirati se može bilo tko na način da unese osnovne podatke koji su potrebni za korištenje aplikacije, a to su: ime, prezime, korisničko ime, e-mail adresa, lozinka i uloga (*vlasnik/čuvvar/oboje*).

S druge strane, korisnik koji je već registriran može se prijaviti u sustav unoseći korisničko ime i lozinku. Registracijom u sustav korisniku se dodjeljuje odabrana

uloga (*vlasnik/čuvar/oboje*). On u tom slučaju može pregledavati svoje osobne podatke te se odjaviti iz sustava, a u nastavku su opisane funkcionalnosti i mogućnosti za pojedinu ulogu korisnika.

Korisnik (vlasnik pasa):

U slučaju da je korisnik odabrao ulogu vlasnika pasa, on ima mogućnost kreiranja zahtjeva za čuvanje pasa u aplikaciji. Pri kreiranju zahtjeva potrebno je unijeti osnovne podatke vezane za čuvanje, a to su: potrebni period čuvanja, potrebne aktivnosti (npr. šetnja, istrčavanje, hranjenje – potrebna količina hrane ako se radi o hranjenju itd.), lokacija čuvanja te željene karakteristike potencijalnog čuvara (ima li čuvar iskustva u čuvanju te ima li vlastitog psa). Osim osnovnih podataka, vlasnik iz liste vlastitih pasa može odabrati pse koje je potrebno čuvati. Svoje pse vlasnik može dodavati na posebnoj stranici i pritom mora unijeti ime psa, datum rođenja i objaviti sliku svog psa. Sve zahtjeve koje je korisnik kreirao može pregledati na zasebnoj stranici za prikaz vlastitih stvorenih zahtjeva. Također, korisnik putem zaglavlja ima mogućnost pregleda svih oglasa koje su objavili čuvari pasa i svih zahtjeva koje su objavili drugi vlasnici pasa.

Korisnik (čuvar pasa):

U slučaju da je korisnik odabrao ulogu čuvara pasa, on ima mogućnost kreiranja oglasa u aplikaciji kojim nudi uslugu čuvanja. Pri kreiranju oglasa potrebno je unijeti osnovne podatke relevantne za oglas, a to su: pasmina koju želi čuvati, preferirana dob pasa, mogući period čuvanja (te je li period fiksna ili fleksibilan), lokacija i željeni broj pasa za čuvanje. Sve oglase koje je korisnik kreirao može pregledati na zasebnoj stranici za prikaz vlastitih stvorenih oglasa. Također, korisnik putem zaglavlja ima mogućnost pregleda svih zahtjeva koje su objavili vlasnici pasa i svih oglasa koje su objavili drugi čuvari pasa.

Uz ulogu korisnika, postoji i uloga administratora.

Administrator sustava, uz funkcionalnosti ranije opisanih korisnika, ima i neke dodatne mogućnosti kao što su: upravljanje zahtjevima/oglasima koje su kreirali vlasnik/čuvar pasa, blokiranje korisnika koji narušavaju pravila sustava te dodjeljivanje administratorskih ovlasti drugom korisniku. Osobni podaci registriranog korisnika bit će vidljivi samo administratoru i koristit će se u svrhu ostvarenja kontakta između uparenih vlasnika i čuvara pasa. Prije javne objave bilo kojeg zahtjeva/oglasa, on se šalje administratoru na uvid u posebnoj kartici koja je vidljiva samo korisnicima s administratorskim ovlastima u aplikaciji, a koji potom odlučuje hoće li taj zahtjev/oglas biti javno dostupan.

Nakon što administrator odobri stvoreni zahtjev/oglas, on postaje javno dostupan te potom korisnik ima dvije mogućnosti za pronalazak najboljeg odgovarajućeg čuvara pasa/psa za čuvanje:

- Odabir "*Najbolja ponuda*" kojim aplikacija na temelju pojedinih karakteristika (pasmina, dob psa, period čuvanja, udaljenost lokacija vlasnika i čuvara) pronalazi najpogodnijeg čuvara/psa. Ukoliko se pronađe takav zahtjev/oglas te korisnik koji je inicirao ovu opciju odluči prihvatiti pronađenu ponudu, korisniku čiji je zahtjev/oglas bio ponuđen, u aplikaciji se pojavljuje zahtjev/oglas za koji je odabrana opcija najbolje ponude na stranici za pregled pristiglih ponuda. Korisnici imaju mogućnost prihvatanja pristigle ponude i ako oba korisnika prihvate odabir tada im se na pregledu trenutnih čuvanja prikazuje i kontakt drugog korisnika te se mogu dogovoriti za detalje. Ako barem jedan korisnik odbije odabir, tada se smatra da nije poželjna daljnja suradnja.
- Korisnik ručno pregledava sve zahtjeve/oglase u aplikaciji, pronalazi onaj koji mu se najviše sviđa i odabirom javljanja na istog šalje čuvaru/vlasniku pasa taj zahtjev/oglas u pristigle ponude čime se daje do znanja da je zainteresiran za suradnju. Čuvar/vlasnik pasa taj odabir prihvaća te se potom na stranici trenutnih čuvanja prikazuje kontakt za daljnji dogovor oko detalja, ili odbija što bi značilo da jedna od strana ne želi ostvariti suradnju.

Nakon završene suradnje vlasnik pasa može ocijeniti koliko je zadovoljan uslugom čuvanja. S druge strane, čuvar pasa isto tako može ocijeniti kakav je bio pas za čuvanje. Prosječna ocjena kojom je ocjenjen čuvar pasa služi tome da bi vlasnik pasa mogao na temelju tuđih mišljenja procijeniti je li baš taj čuvar pogodan za njegovog ljubimca. Isto tako, prosječna ocjena psa služi tome da bi potencijalni čuvar mogao procijeniti je li taj pas pogodan za čuvanje na temelju tuđih iskustava.

Aplikacija treba biti izvedena kao web aplikacija prilagođena mobilnom uređaju i tabletu kojoj će registrirani korisnici pristupati uz pomoć korisničkog imena i lozinke. Također, sustav treba podržavati rad više korisnika u stvarnom vremenu.

Aplikacija ima puno prostora za nadogradnju u budućnosti. Neke od početnih ideja nadogradnje bile bi dodavanje mogućnosti pregledavanja korisničkih računa drugih korisnika (uz to bi se i nadogradile mogućnosti uređivanja samog korisničkog računa - npr. moguće objavljivanje fotografija svojih ljubimaca) i slanja zahtjeva za prijateljstvo, kako bi osobe mogle ostati povezane za potencijalnu buduću

suradnju. Također, moguće proširenje aplikacije bila bi implementacija sustava kojim se oglašavaju psi za udomljavanje i zbrinjavanje.

Neke funkcionalnosti ove aplikacije mogle bi se usporediti s funkcionalnostima aplikacije za pružanje usluga prijevoza pod nazivom "*Uber*". Aplikacija "*Uber*" ima funkcionalnost pronalaska najboljeg odabira, koja radi na način da uparuje vozače i korisnike s najpogodnijom lokacijskom udaljenosti, tako da korisniku u što kraćem roku bude pružena željena usluga. Po završetku vožnje korisnik također ima opciju ocijeniti koliko je bio zadovoljan vožnjom. Navedene funkcionalnosti su vrlo slične funkcionalnostima aplikacije kojom se bavi ovaj projekt.

3. Specifikacija programske potpore

3.1 Funkcionalni zahtjevi

Dionici:

1. Neregistrirani korisnik
2. Registrirani korisnik
 - (a) Vlasnik pasa
 - (b) Čuvar pasa
3. Administrator
4. Razvojni tim

Aktori i njihovi funkcionalni zahtjevi:

1. Neregistrirani korisnik (inicijator) može:
 - (a) Pregledavati objavljene zahtjeve i oglase te njihove pojedinosti (lokacija, period čuvanja itd.)
 - (b) Registrirati se u sustav, stvoriti novi korisnički račun za koji mu trebaju ime, prezime, korisničko ime, e-mail adresa, lozinka i uloga.
2. Registrirani korisnik (inicijator) može:
 - (a) Pregledavati osobne podatke
 - (b) Odjaviti se s korisničkog računa i prijaviti se nekim drugim računom
 - (c) Pregledavati pristigle ponude i povijest ponuda
 - (d) Pregledavati trenutna čuvanja i stupiti u kontakt s drugim korisnikom
 - (e) Imati ulogu vlasnika pasa:
 - i. Spremiti podatke o svom psu
 - ii. Pregledavati svoje pse
 - iii. Pregledavati sve odobrene oglase čuvara pasa i sve odobrene zahtjeve drugih vlasnika pasa
 - iv. Pregledavati vlastite zahtjeve

- v. Stvarati novi zahtjev za čuvanje pasa
- vi. Pronaći najbolji odabir čuvara pasa
- vii. Ocjenjivati uslugu čuvara pasa

(f) Imati ulogu čuvara pasa:

- i. Pregledavati sve odobrene zahtjeve vlasnika pasa i sve odobrene oglase drugih čuvara pasa
- ii. Pregledavati vlastite oglase
- iii. Stvarati novi oglas za čuvanje pasa
- iv. Pronaći najbolji odabir vlasnika pasa
- v. Ocjenjivati ponašanje psa

(g) Imati ulogu vlasnika i čuvara pasa:

- i. Ima sve navedene funkcionalnosti vlasnika pasa i čuvara pasa

3. Administrator (inicijator) može:

- (a) Obavljati sve funkcionalnosti registriranog korisnika
- (b) Dodijeliti administratorsku ulogu drugim korisnicima
- (c) Odobriti ili odbaciti zahtjev/oglas
- (d) Blokirati korisnike koji narušavaju pravila sustava

4. Baza podataka (sudionik) može:

- (a) Pohraniti sve podatke o registriranim korisnicima i njihovim ulogama
- (b) Pohraniti sve podatke o psima
- (c) Pohraniti sve podatke o stvorenim zahtjevima i oglasima
- (d) Pohraniti podatke o mogućima aktivnostima sa psima

3.1.1 Obrasci uporabe

Opis obrazaca uporabe

UC01 - Pregledavanje zahtjeva/oglasa

- **Glavni sudionik:** Neregistrirani korisnik
- **Cilj:** Pregledati ponuđene zahtjeve i oglase
- **Sudionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
 1. Korisnik putem zaglavlja aplikacije odabire opciju za pregled objavljenih zahtjeva/oglasa
 2. Korisniku se otvara stranica s objavljenim zahtjevima/oglasima koje potom može pregledavati

UC02 - Registracija

- **Glavni sudionik:** Neregistrirani korisnik
- **Cilj:** Stvoriti korisnički račun za korištenje svih funkcionalnosti aplikacije
- **Sudionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
 1. Korisnik putem zaglavlja aplikacije odabire opciju za registraciju
 2. Korisnik unosi korisničke podatke potrebne za registraciju i potvrđuje unos
 3. Korisnik se upisuje u bazu podataka i preusmjerava na početnu stranicu
- **Opis mogućih odstupanja:**
 - 2.a Odabir već zauzetog korisničkog imena i/ili e-mail adrese, unos korisničkih podataka u neispravnom formatu, nepodudaranje lozinki
 1. Sustav obavještava korisnika o neuspjelom unosu i vraća ga na stranicu za registraciju
 2. Korisnik mijenja potrebne podatke te završava unos ili odustaje od registracije

UC03 - Prijava

- **Glavni sudionik:** Registrirani korisnik
- **Cilj:** Dobivanje pristupa svim funkcionalnostima aplikacije

- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je pohranjen u bazu podataka
- **Opis osnovnog tijeka:**
 1. Korisnik putem zaglavlja aplikacije odabire opciju za prijavu
 2. Korisnik unosi korisničko ime i lozinku te potvrđuje upis
 3. Korisnik se preusmjerava na početnu stranicu
- **Opis mogućih odstupanja:**
 - 2.a Neispravan unos korisničkog imena i/ili lozinke
 1. Sustav obavještava korisnika o neuspjelom unosu i vraća ga na stranicu za prijavu
 2. Korisnik unosi ispravne podatke za prijavu ili odustaje od prijave

UC04 - Odjava

- **Glavni sudionik:** Registrirani korisnik
- **Cilj:** Odjaviti se iz sustava
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav
- **Opis osnovnog tijeka:**
 1. Korisnik pristupa padajućem izborniku u zaglavlju aplikacije i odabire opciju za odjavljivanje iz sustava
 2. Aplikacija korisnika odjavljuje s trenutnog korisničkog računa
 3. Korisnik se preusmjerava na početnu stranicu

UC05 - Pregledavanje osobnih podataka

- **Glavni sudionik:** Registrirani korisnik
- **Cilj:** Pregledati osobne podatke
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav
- **Opis osnovnog tijeka:**
 1. Korisnik pristupa padajućem izborniku u zaglavlju aplikacije i odabire opciju za pregled vlastitog računa
 2. Aplikacija prikazuje osobne podatke korisnika

UC06 – Pregledavanje tuđih zahtjeva

- **Glavni sudionik:** Registrirani korisnik
- **Cilj:** Pronalaženje odgovarajućeg zahtjeva

- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav
- **Opis osnovnog tijeka:**
 1. Korisnik putem zaglavlja aplikacije pristupa objavljenim zahtjevima
 2. Korisnik na stranici s objavljenim zahtjevima prolazi kroz listu tuđih zahtjeva te pregledava podatke pojedinih zahtjeva

UC07 – Pregledavanje tuđih oglasa

- **Glavni sudionik:** Registrirani korisnik
- **Cilj:** Pronalaženje odgovarajućeg oglasa
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav
- **Opis osnovnog tijeka:**
 1. Korisnik putem zaglavlja aplikacije pristupa objavljenim oglasima
 2. Korisnik na stranici s objavljenim oglasima prolazi kroz listu tuđih oglasa te pregledava podatke pojedinih oglasa

UC08 – Pregledavanje vlastitih pasa

- **Glavni sudionik:** Vlasnik pasa
- **Cilj:** Pregledavanje informacija o psima
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav, korisnik ima ulogu vlasnika pasa
- **Opis osnovnog tijeka:**
 1. Korisnik na padajućem izborniku u zaglavlju aplikacije ili na stranici s osobnim podacima odabire opciju za pregled vlastitih pasa
 2. Korisnik na stranici s prikazom vlastitih pasa pregledava informacije o pojedinim psima

UC09 – Dodavanje pasa

- **Glavni sudionik:** Vlasnik pasa
- **Cilj:** Dodavanje psa na vlastiti račun
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav, korisnik ima ulogu vlasnika pasa
- **Opis osnovnog tijeka:**
 1. Korisnik na padajućem izborniku u zaglavlju aplikacije ili na stranici s osobnim podacima odabire opciju za pregled vlastitih pasa

2. Korisnik na stranici s prikazom vlastitih pasa odabire opciju za dodavanje psa
 3. Korisnik ispunjava podatke o psu te potvrđuje unos
 4. Podaci o psu se pohranjuju u bazu podataka
- **Opis mogućih odstupanja:**
 - 3.a Korisnik unosi ime psa koji već postoji na njegovom računu ili unosi neispravan datum rođenja
 1. Sustav obavještava korisnika o neuspjelom unosu i vraća ga na stranicu za dodavanje psa
 2. Korisnik unosi nove podatke za psa ili odustaje od dodavanja

UC10 – Pregledavanje vlastitih zahtjeva

- **Glavni sudionik:** Vlasnik pasa
- **Cilj:** Pregledavanje svojih kreiranih zahtjeva
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav, korisnik ima ulogu vlasnika pasa
- **Opis osnovnog tijeka:**
 1. Korisnik pristupa padajućem izborniku u zaglavlju aplikacije i odabire opciju za prikaz vlastitih zahtjeva
 2. Korisniku se prikazuje lista zahtjeva koje potom može pregledavati

UC11 – Pregledavanje vlastitih oglasa

- **Glavni sudionik:** Čuvar pasa
- **Cilj:** Pregledavanje svojih kreiranih oglasa
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav, korisnik ima ulogu čuvara pasa
- **Opis osnovnog tijeka:**
 1. Korisnik pristupa padajućem izborniku u zaglavlju aplikacije i odabire opciju za prikaz vlastitih oglasa
 2. Korisniku se prikazuje lista oglasa koje potom može pregledavati

UC12 – Kreiranje novog zahtjeva

- **Glavni sudionik:** Vlasnik pasa
- **Cilj:** Objavljivanje novog zahtjeva za čuvanje pasa
- **Sudionici:** Baza podataka, administrator
- **Preduvjet:** Korisnik je prijavljen u sustav, korisnik ima ulogu vlasnika pasa

- **Opis osnovnog tijeka:**

1. Korisnik putem zaglavlja aplikacije pristupa padajućem izborniku na kojem odabire opciju za prikaz vlastitih zahtjeva ili na zaglavlju aplikacije pristupa stranici sa svim objavljenim zahtjevima
2. Korisnik na odgovarajućoj stranici odabire opciju za kreiranje novog zahtjeva
3. Korisnik unosi sve tražene podatke za kreiranje zahtjeva (period čuvanja, potrebne aktivnosti itd.) i potvrđuje unos
4. Zahtjev se pohranjuje u bazu podataka i šalje administratoru na uvid

- **Opis mogućih odstupanja:**

- 3.a Korisnik nije ispravno unio sve podatke

1. Korisnika se ponovno vraća na stranicu za kreiranje zahtjeva s odgovarajućom porukom o greški
2. Korisnik unosi ispravne podatke ili odustaje od kreiranja zahtjeva

UC13 – Kreiranje novog oglasa

- **Glavni sudionik:** Čuvar pasa

- **Cilj:** Objavljivanje novog oglasa za čuvanje pasa

- **Sudionici:** Baza podataka, administrator

- **Preduvjet:** Korisnik je prijavljen u sustav, korisnik ima ulogu čuvara pasa

- **Opis osnovnog tijeka:**

1. Korisnik putem zaglavlja aplikacije pristupa padajućem izborniku na kojem odabire opciju za prikaz vlastitih oglasa ili na zaglavlju aplikacije pristupa stranici sa svim objavljenim oglasima
2. Korisnik na odgovarajućoj stranici odabire opciju za kreiranje novog oglasa
3. Korisnik unosi sve tražene podatke za kreiranje oglasa (preferirana pasmina, preferirana dob, period čuvanja itd.) i potvrđuje unos
4. Oglas se pohranjuje u bazu podataka i šalje administratoru na uvid

- **Opis mogućih odstupanja:**

- 3.a Korisnik nije ispravno unio sve podatke

1. Korisnika se ponovno vraća na stranicu za kreiranje oglasa s odgovarajućom porukom o greški
2. Korisnik unosi ispravne podatke ili odustaje od kreiranja oglasa

UC14 – Pronalaženje najboljeg odabira čuvara pasa

- **Glavni sudionik:** Vlasnik pasa
- **Cilj:** Pronalaženje najprikladnijeg čuvara pasa
- **Sudionici:** Baza podataka, čuvar pasa
- **Preduvjet:** Korisnik je prijavljen u sustav, korisnik ima ulogu vlasnika pasa, korisnikov zahtjev je potvrđen od strane administratora
- **Opis osnovnog tijeka:**
 1. Korisnik pristupa padajućem izborniku u zaglavlju aplikacije i odabire opciju za prikaz vlastitih zahtjeva
 2. Korisnik na listi vlastitih zahtjeva pokraj željenog zahtjeva odabire opciju za pronalaženje najboljeg odabira
 3. Korisniku se prikazuje najbolji odabir kojeg on potom prihvaća
 4. Čuvaru pasa se na stranici za pristigle ponude pojavljuje ponuđeni zahtjev kojeg on potom prihvaća
 5. Vlasniku i čuvaru se na stranici s čuvanjima u tijeku prikazuju osobni podaci za stupanje u kontakt
- **Opis mogućih odstupanja:**
 - 3.a Vlasnik pasa odbija pronađeni odabir
 1. Daljnja suradnja između korisnika nije moguća
 - 4.a Čuvar pasa odbija ponuđeni zahtjev
 1. Daljnja suradnja između korisnika nije moguća

UC15 – Pronalaženje najboljeg odabira vlasnika pasa

- **Glavni sudionik:** Čuvar pasa
- **Cilj:** Pronalaženje najprikladnijeg psa za čuvanje
- **Sudionici:** Baza podataka, vlasnik pasa
- **Preduvjet:** Korisnik je prijavljen u sustav, korisnik ima ulogu čuvara pasa, korisnikov oglas je potvrđen od strane administratora
- **Opis osnovnog tijeka:**
 1. Korisnik pristupa padajućem izborniku u zaglavlju aplikacije i odabire opciju za prikaz vlastitih oglasa
 2. Korisnik na listi vlastitih oglasa pokraj željenog oglasa odabire opciju za pronalaženje najboljeg odabira
 3. Korisniku se prikazuje najbolji odabir kojeg on potom prihvaća
 4. Vlasniku pasa se na stranici za pristigle ponude pojavljuje ponuđeni oglas kojeg on potom prihvaća
 5. Čuvaru pasa i vlasniku pasa se na stranici s čuvanjima u tijeku prikazuju

osobni podaci za stupanje u kontakt

- **Opis mogućih odstupanja:**

- 3.a Čuvar pasa odbija pronađeni odabir

- 1. Daljnja suradnja između korisnika nije moguća

- 4.a Vlasnik pasa odbija ponuđeni oglas

- 1. Daljnja suradnja između korisnika nije moguća

UC16 – Javljanje na zahtjev

- **Glavni sudionik:** Čuvar pasa

- **Cilj:** Pronalaženje najprikladnijeg psa za čuvanje

- **Sudionici:** Baza podataka, vlasnik pasa

- **Preduvjet:** Korisnik je prijavljen u sustav, korisnik ima ulogu čuvara pasa

- **Opis osnovnog tijeka:**

- 1. Korisnik u zaglavlju aplikacije pristupa svim objavljenim zahtjevima

- 2. Korisnik prolazi kroz listu objavljenih zahtjeva i pregledava informacije o pojedinim zahtjevima

- 3. Korisnik na željenom zahtjevu odabire opciju za javljanje

- 4. Vlasniku pasa se na stranici za pristigle ponude pojavljuje zahtjev na koji se čuvar odlučio javiti

- 5. Vlasnik pasa prihvata javljanje

- 6. Čuvaru pasa i vlasniku pasa se na stranici s čuvanjima u tijeku prikazuju osobni podaci za stupanje u kontakt

- **Opis mogućih odstupanja:**

- 5.a Vlasnik pasa odbija javljanje

- 1. Daljnja suradnja između korisnika nije moguća

UC17 – Javljanje na oglas

- **Glavni sudionik:** Vlasnik pasa

- **Cilj:** Pronalaženje najprikladnijeg čuvara pasa

- **Sudionici:** Baza podataka, čuvar pasa

- **Preduvjet:** Korisnik je prijavljen u sustav, korisnik ima ulogu vlasnika pasa

- **Opis osnovnog tijeka:**

- 1. Korisnik u zaglavlju aplikacije pristupa svim objavljenim oglasima

- 2. Korisnik prolazi kroz listu objavljenih oglasa i pregledava informacije o pojedinim oglasima

- 3. Korisnik na željenom oglasu odabire opciju za javljanje

4. Čuvaru pasa se na stranici za pristigle ponude pojavljuje oglas na koji se čuvar odlučio javiti
 5. Čuvar pasa prihvata javljanje
 6. Čuvaru i vlasniku se na stranici s čuvanjima u tijeku prikazuju osobni podaci za stupanje u kontakt
- **Opis mogućih odstupanja:**
 - 5.a Čuvar pasa odbija javljanje
 1. Daljnja suradnja između korisnika nije moguća

UC18 – Pregledavanje pristiglih ponuda

- **Glavni sudionik:** Registrirani korisnik
- **Cilj:** Pregledavanje mogućih suradnji te potencijalno sklapanje dogovora između čuvara pasa i vlasnika pasa
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav, vlasnik pasa ili čuvar pasa je inicirao želju za suradnjom
- **Opis osnovnog tijeka:**
 1. Korisnik putem padajućeg izbornika u zaglavlju aplikacije pristupa stranici s pristiglim ponudama
 2. Korisnik pregledava listu pristiglih ponuda te ima opciju prihvatiti ili odbiti pojedinu ponudu

UC19 – Pregledavanje povijesti ponuda

- **Glavni sudionik:** Registrirani korisnik
- **Cilj:** Pregledavanje prošlih suradnji te moguće ocjenjivanje iskustva
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav, vlasnik pasa i čuvar pasa imali su poslovni odnos
- **Opis osnovnog tijeka:**
 1. Korisnik putem padajućeg izbornika u zaglavlju aplikacije pristupa stranici s pristiglim ponudama
 2. Korisnik pregledava listu povijesti ponuda te ima opciju ocijeniti iskustvo

UC20 – Pregledavanje čuvanja u tijeku

- **Glavni sudionik:** Registrirani korisnik

- **Cilj:** Praćenje aktivnih suradnji i dobivanje podataka za ostvarenje kontakta
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav, vlasnik pasa i čuvar pasa imaju poslovni odnos u tijeku
- **Opis osnovnog tijeka:**
 1. Korisnik putem padajućeg izbornika u zaglavlju aplikacije pristupa stranici s čuvanjima u tijeku
 2. Korisnik pregledava listu čuvanja u tijeku te saznaje potrebne informacije

UC21 - Ocjenjivanje usluge čuvanja

- **Glavni sudionik:** Vlasnik pasa
- **Cilj:** Pružiti informaciju ostalim vlasnicima pasa o kvaliteti čuvara pasa
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav, korisnik ima ulogu vlasnika pasa, vlasnik pasa i čuvar pasa imali su poslovni odnos
- **Opis osnovnog tijeka:**
 1. Korisnik preko padajućeg izbornika u zaglavlju aplikacije odabire opciju za prikaz pristiglih ponuda
 2. Korisnik u listi povijesti ponuda pronalazi željenog čuvara s kojim je poslovao
 3. Korisnik ocjenjuje iskustvo s čuvarom pasa

UC22 - Ocjenjivanje ponašanja psa

- **Glavni sudionik:** Čuvar pasa
- **Cilj:** Pružiti informaciju ostalim čuvarima pasa kakav je pas za čuvanje
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav, korisnik ima ulogu čuvara pasa, vlasnik pasa i čuvar pasa imali su poslovni odnos
- **Opis osnovnog tijeka:**
 1. Korisnik preko padajućeg izbornika u zaglavlju aplikacije odabire opciju za prikaz pristiglih ponuda
 2. Korisnik u listi povijesti ponuda pronalazi željeno čuvanje koje je odradio
 3. Korisnik ocjenjuje iskustvo sa psom

UC23 - Dodjeljivanje administratorske uloge korisniku

- **Glavni sudionik:** Administrator
- **Cilj:** Pružiti korisniku administratorsku ulogu
- **Sudionici:** Baza podataka
- **Preduvjet:** Administrator je prijavljen u sustav, korisnik nema ulogu administratora
- **Opis osnovnog tijeka:**
 1. Administrator preko padajućeg izbornika u zaglavlju aplikacije pristupa stranici za administratorsko upravljanje
 2. Administrator u listi korisnika pronalazi željenog korisnika
 3. Administrator odabranom korisniku dodjeljuje administratorsku ulogu

UC24 - Potvrđivanje stvorenog zahtjeva

- **Glavni sudionik:** Administrator
- **Cilj:** Potvrditi stvoreni zahtjev da se može objaviti
- **Sudionici:** Baza podataka
- **Preduvjet:** Administrator je prijavljen u sustav, vlasnik pasa je stvorio zahtjev
- **Opis osnovnog tijeka:**
 1. Administrator preko padajućeg izbornika u zaglavlju aplikacije pristupa stranici za administratorsko upravljanje
 2. Administrator u listi prikazanih zahtjeva i oglasa pronalazi željeni zahtjev
 3. Administrator odabirom opcije za potvrđivanje potvrđuje zahtjev
 4. Zahtjev se objavljuje
- **Opis mogućih odstupanja:**
 - 3.a Administrator smatra zahtjev nevaljanim
 1. Administrator odabirom opcije za odbijanje odbija zahtjev
 2. Zahtjev se ne objavljuje

UC25 - Potvrđivanje stvorenog oglasa

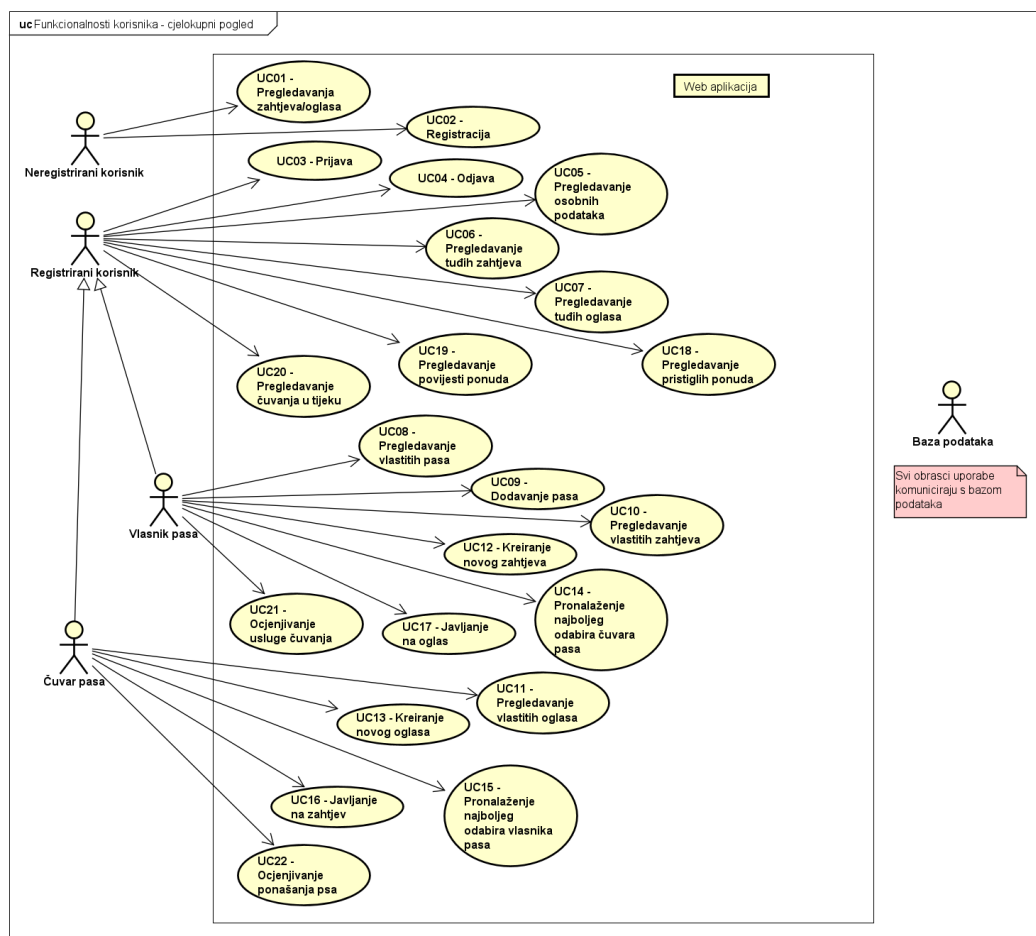
- **Glavni sudionik:** Administrator
- **Cilj:** Potvrditi stvoreni oglas da se može objaviti
- **Sudionici:** Baza podataka
- **Preduvjet:** Administrator je prijavljen u sustav, čuvar pasa je stvorio oglas
- **Opis osnovnog tijeka:**

1. Administrator preko padajućeg izbornika u zaglavlju aplikacije pristupa stranici za administratorsko upravljanje
 2. Administrator u listi prikazanih zahtjeva i oglasa pronalazi željeni oglas
 3. Administrator odabirom opcije za potvrđivanje potvrđuje oglas
 4. Oglas se objavljuje
- **Opis mogućih odstupanja:**
 - 3.a Administrator smatra oglas nevaljanim
 1. Administrator odabirom opcije za odbijanje odbija oglas
 2. Oglas se ne objavljuje

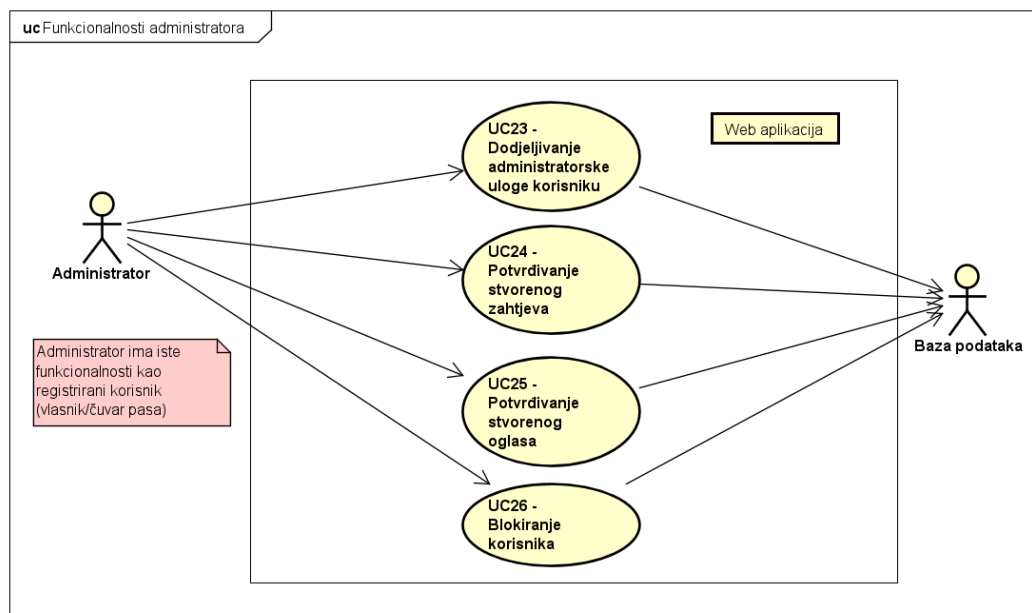
UC26 - Blokiranje korisnika

- **Glavni sudionik:** Administrator
- **Cilj:** Blokirati korisnika radi narušavanja pravila sustava
- **Sudionici:** Baza podataka
- **Preduvjet:** Administrator je prijavljen u sustav, korisnik je narušio pravila sustava
- **Opis osnovnog tijeka:**
 1. Administrator preko padajućeg izbornika u zaglavlju aplikacije pristupa stranici za administratorsko upravljanje
 2. Administrator u listi korisnika pronalazi željenog korisnika
 3. Administrator blokira odabranog korisnika

Dijagrami obrazaca uporabe



Slika 3.1: Dijagram obrasca uporabe - Funkcionalnost neregistriranog korisnika i registriranog korisnika (vlasnika pasa i čuvara pasa)

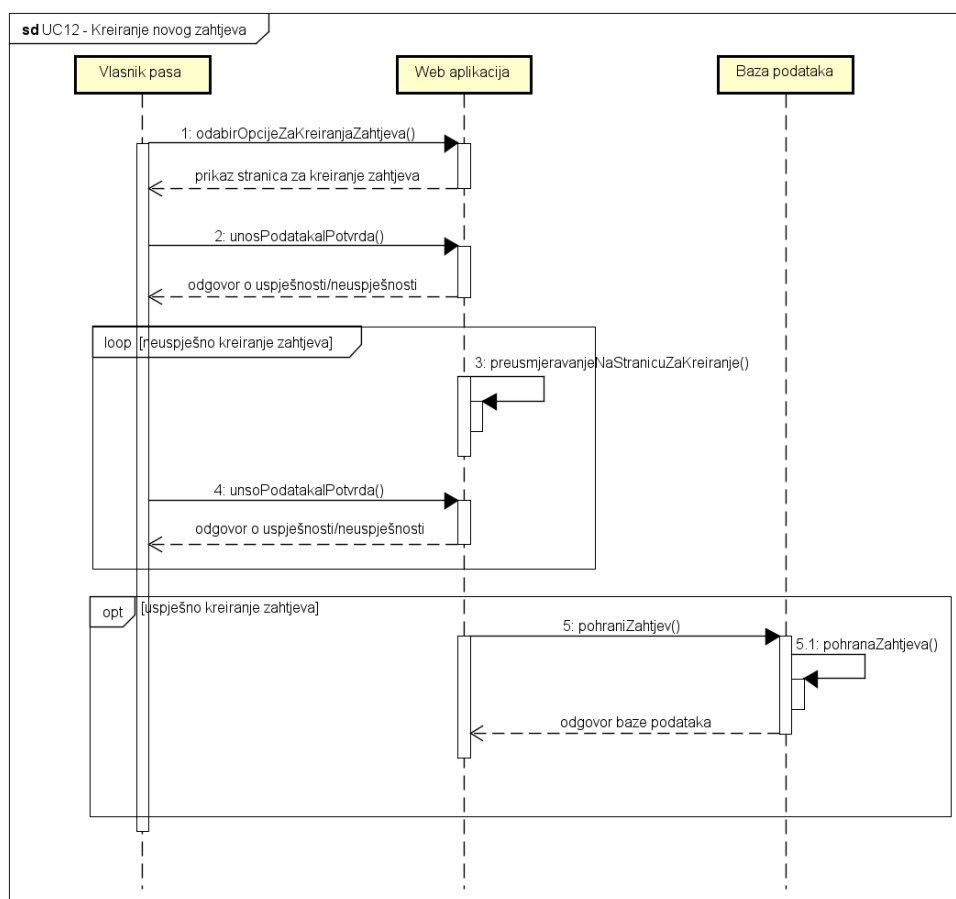


Slika 3.2: Dijagram obrasca uporabe - Funkcionalnost administratora

3.1.2 Sekvencijski dijagrami

Obrazac uporabe UC12 - Kreiranje novog zahtjeva

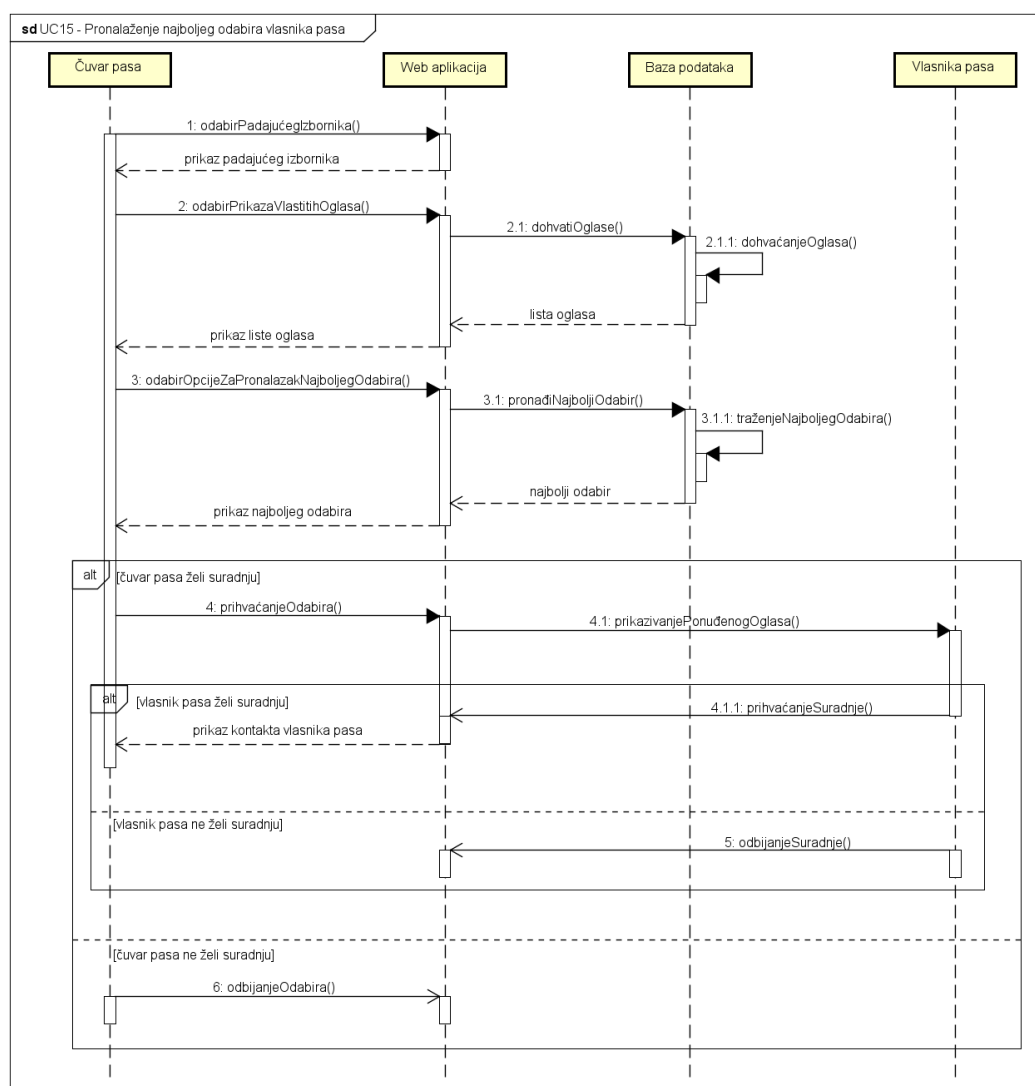
Vlasnik pasa na stranici s prikazom vlastitih zahtjeva ili na stranici sa svim objavljenim zahtjevima odabire opciju za kreiranje novog zahtjeva. Web aplikacija mu otvara stranicu za kreiranje zahtjeva. Korisnik tamo unosi tražene podatke (period čuvanja, potrebne aktivnosti itd.) za kreiranje zahtjeva. Potom korisnik potvrđuje kreiranje zahtjeva. Ako korisnik nije ispravno unio podatke, prilikom potvrde web aplikacija ga ponovno usmjerava na istu stranicu s prikladnom porukom o greški. Ako su svi podaci uneseni ispravno, web aplikacija mu daje odgovor o uspješnom kreiranju zahtjeva. Potom se zahtjev pohranjuje u bazu podataka.



Slika 3.3: Sekvencijski dijagram za UC12

Obrazac uporabe UC15 - Pronalaženje najboljeg odabira vlasnika pasa

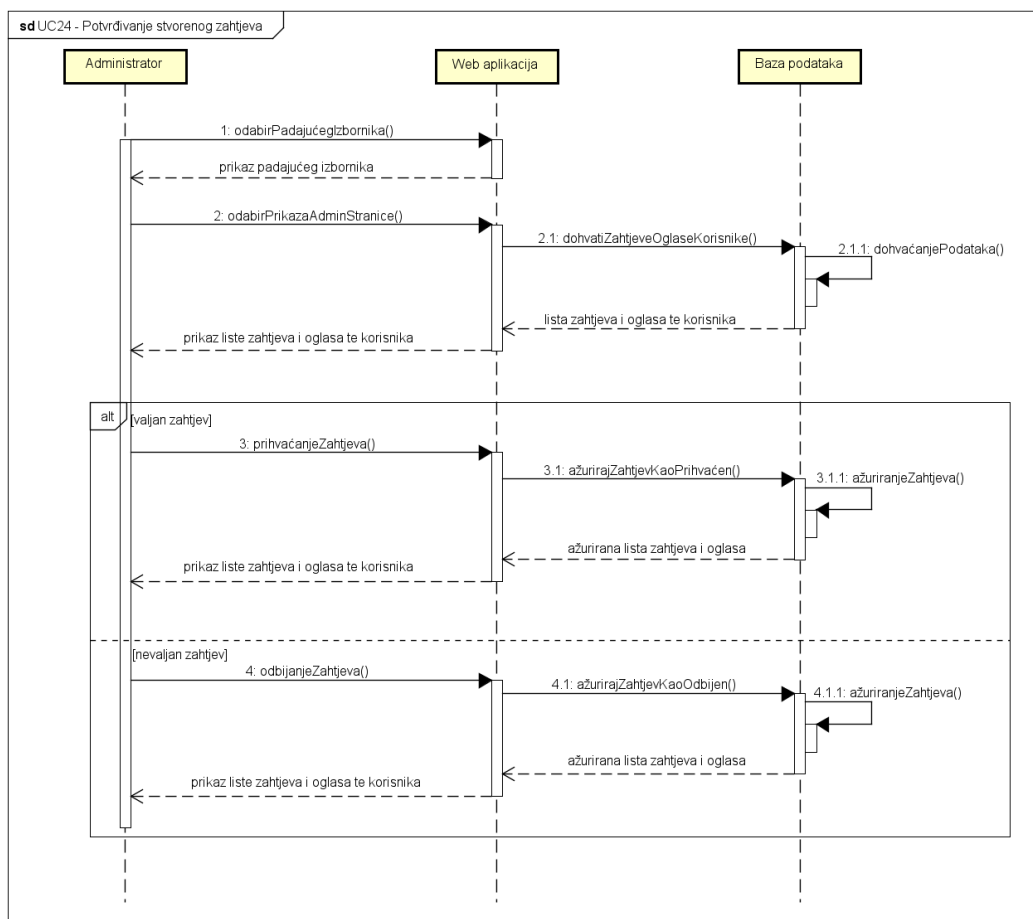
Čuvar pasa u zaglavlju aplikacije putem padajućeg izbornika odabire opciju za prikaz vlastitih oglasa. Web aplikacija mu otvara stranicu s njegovim oglasima. Korisnik tamo pokraj željenog oglasa odabire opciju za pronalaženje najboljeg odabira. Web aplikacija pronalazi najboljeg vlasnika pasa i šalje ga korisniku. Korisnik ima mogućnost prihvatiti ga, ali i odbiti. Ako korisnik odbije, nema daljnje suradnje s vlasnikom pasa. Ako ga korisnik prihvati, oglas nad kojim je zatražen najbolji odabir se šalje vlasniku pasa. Tada vlasnik pasa ima također opciju prihvatiti suradnju kao i odbiti ju.



Slika 3.4: Sekvencijski dijagram za UC15

Obrazac uporabe UC24 - Potvrđivanje stvorenog zahtjeva

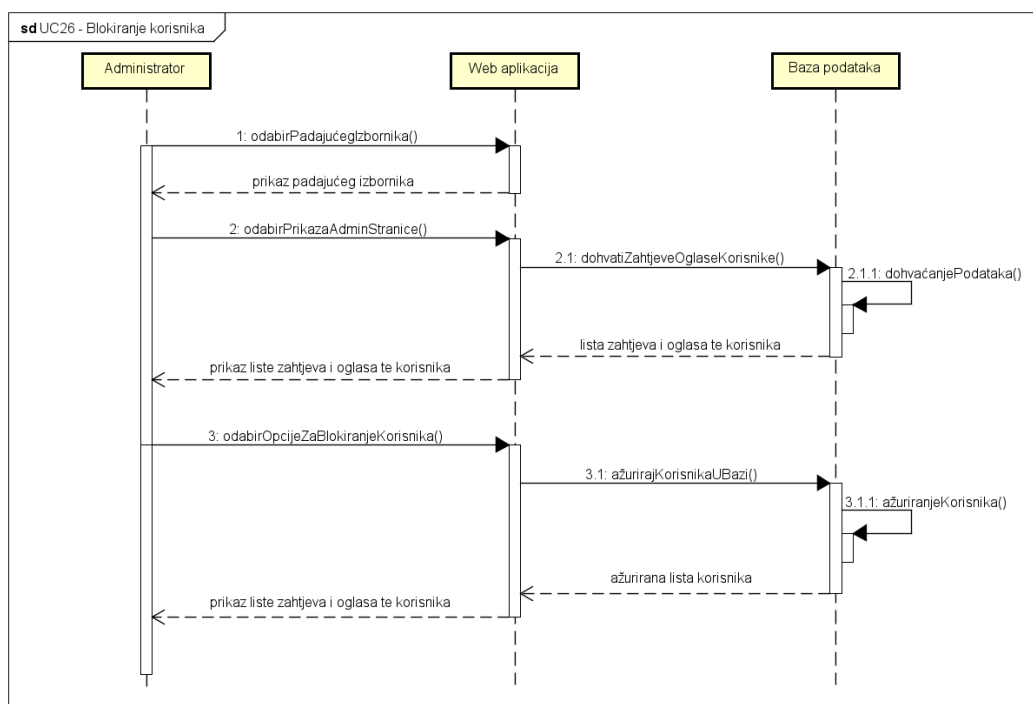
Administrator sustava putem padajućeg izbornika u zaglavlju aplikacije odabire opciju za pristupanje stranici za administratorsko upravljanje. Web aplikacija šalje upit za potrebnim elementima (svi nepotvrđeni zahtjevi i oglasi te svi korisnici) bazi podataka. Primitkom istih, web aplikacija otvara stranicu administratoru. Administrator u listi zahtjeva i oglasa pronalazi željeni zahtjev te kraj njega odabire opciju za njegovo potvrđivanje. Web aplikacija šalje upit za ažuriranjem zahtjeva bazi podataka. Baza podataka ga ažurira i šalje ažuriranu listu zahtjeva i oglasa web aplikaciji. Web aplikacija otvara ponovno stranicu administratoru, ali sada s ažuriranom listom zahtjeva i oglasa.



Slika 3.5: Sekvencijski dijagram za UC24

Obrazac uporabe UC26 - Blokiranje korisnika

Administrator sustava putem padajućeg izbornika u zaglavlju aplikacije odabire opciju za pristupanje stranici za administratorsko upravljanje. Web aplikacija šalje upit za potrebnim elementima (svi nepotvrđeni zahtjevi i oglasi te svi korisnici) bazi podataka. Primitkom istih, web aplikacija otvara stranicu administratoru. Administrator u listi korisnika pronalazi željenog korisnika te kraj njega odabire opciju za njegovo blokiranje. Web aplikacija šalje upit za ažuriranjem korisnika bazi podataka. Baza podataka postavlja atribut kojim se označava da je korisnik blokiran na istinit i šalje ažuriranu listu korisnika web aplikaciji. Web aplikacija otvara ponovno stranicu administratoru, ali sada blokirani korisnik kraj sebe nema gumb za blokiranje.



Slika 3.6: Sekvencijski dijagram za UC26

3.2 Ostali zahtjevi

- Aplikacija treba biti izvedena kao web aplikacija prilagođena mobilnom uređaju i tabletu
- Aplikaciji pristupaju registrirani korisnici uz pomoć korisničkog imena i lozinke
- Sustav treba podržavati rad više korisnika u stvarnom vremenu
- Aplikaciju treba implementirati u arhitekturi klijent-poslužitelj
- Na poslužiteljskoj strani se treba koristiti programski jezik Java i radni okvir Spring Boot
- Podaci se trebaju spremati u relacijsku bazu podataka koristeći JPA
- Funkcionalnost web aplikacije se treba izložiti kroz REST web servis
- Na klijentkoj strani treba implementirati korisničko sučelje u web pregledniku koristeći React, koje se spaja na navedene servise

4. Arhitektura i dizajn sustava

Arhitektura aplikacije "*Čuvari pasa*" može se podijeliti na web poslužitelj, web aplikaciju i bazu podataka.

Web poslužitelj je zapravo funkcionalnost web aplikacije. On komunicira s klijentom preko web preglednika. Primarna uloga mu je "nabavljanje" HTTP zahtjeva i slanje HTTP odgovora. Ovisno o zahtjevu, web poslužitelj komunicira i s bazom podataka u kojoj se nalaze potrebni podaci.

Web preglednik je softver koji služi kao sučelje između poslužitelja i klijenta i prikaz web dokumenta klijentu. Njegova primarna uloga je slanje HTTP zahtjeva i primanje HTTP odgovora. Web preglednici prevode kod koji dobivaju u HTTP odgovoru te ga zatim prikazuju.

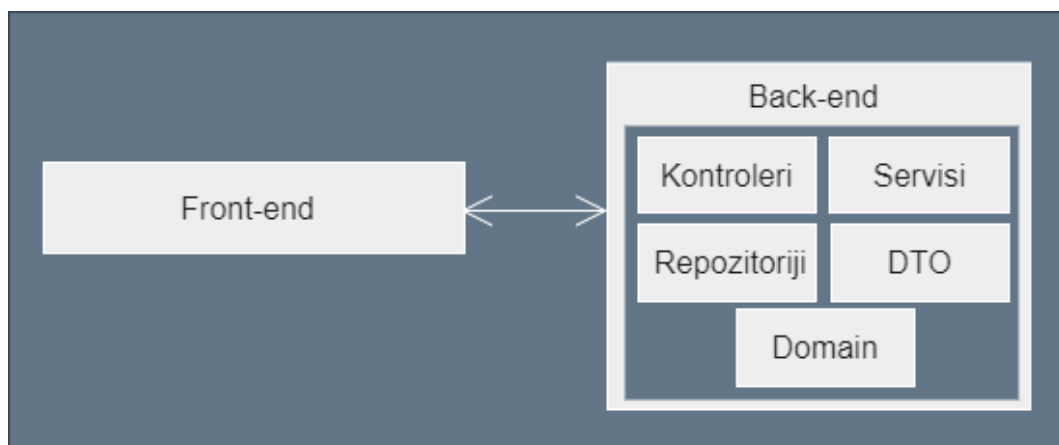
Baza podataka (podatkovni sloj) koristi se za sigurno spremanje podataka. Povezana je s web aplikacijom koja joj šalje upite za potrebnim podacima. Baza podataka je dalje opisana u poglavlju 4.1 Baza podataka.

Web aplikacija podijeljena je na back-end i front-end.

Back-end je dio sustava u kojem se zapravo obrađuju zahtjevi i vrše daljnje akcije. Njegova organizacija na kontrolere, servise i repozitorije pomaže u "razdvajanju zabrinutosti". Kontroleri upravljaju REST sučeljima prema poslovnoj logici koja je implementirana u servisima. Repozitoriji su zaslužni za spremanje i dohvaćanje nekog skupa podataka. Back-end također sadrži DTO-e (Data Transfer Object) i Domain modele. DTO-i se koriste za razmjenu podataka između procesa ili slojeva, a Domain modeli služe za razmjenu podataka s bazom podataka.

Front-end je prezentacijski dio aplikacije odnosno kako korisnik vidi aplikaciju u web pregledniku.

Front-end smo odlučili raditi u programskom jeziku JavaScript koristeći radni okvir React, a back-end u programskom jeziku Java s radnim okvirom Spring Boot. Razvojna okruženja u kojima radimo su Visual Studio Code i IntelliJ IDEA.



Slika 4.1: Dijagram arhitekture web aplikacije

4.1 Baza podataka

Sustav koristi relacijsku bazu podataka koja će biti implementirana u PostgreSQL-u. Relacijsku bazu podataka koristimo radi lakšeg oblikovanja sustava kao stvarnog svijeta, a PostgreSQL jer smo najbolje upoznati s njim. U njemu se entiteti modeliraju kao tablice koje imaju vlastito ime i skup atributa.

Baza podataka nam je potrebna zbog njezine sigurnosti podataka, ali i brzog dohvata, pohrane i izmijene podataka koje sustav koristi za daljnje akcije. Baza podataka ovog sustava koristiti će sjedeće entitete:

- role
- appuser
- breed
- dog
- request_dog
- request_guardian
- request_guardians_dog
- activity
- request_activity
- agreed_request

4.1.1 Opis tablica

role je entitet koji sadrži sve važne informacije o ulogama korisnika u aplikaciji. Sastoji se od atributa: `role_id`, `name`. Povezan je vezom *One-To-Many* s tablicom

appuser preko vlastitog atributa role_id.

role		
role_id	INT	Jedinstveni identifikator uloge korisnika
name	VARCHAR	Naziv uloge u sustavu

appuser je entitet koji sadrži sve važne informacije o korisniku aplikacije. Sastoji se od atributa: user_id, role_id, username, first_name, last_name, password, rating_sum, rating_count, email, has_dog, has_experience, blocked. Povezan je vezom *Many-To-One* s tablicom role preko atributa role_id tablice role. S tablicom dog je povezan vezom *One-To-Many* preko vlastitog atributa user_id. Isto vrijedi za tablice request_dog i request_guardian. S tablicom agreed_request ima dvije veze, a to su *One-To-Many* i *One-To-One* preko vlastitog atributa user_id.

appuser		
user_id	INT	Jedinstveni identifikator korisnika
role_id	INT	Jedinstven identifikator uloge korisnika (role.role_id)
username	VARCHAR	Korisničko ime u sustavu
first_name	VARCHAR	Ime korisnika
last_name	VARCHAR	Prezime korisnika
password	VARCHAR	Hash lozinke korisnika
rating_sum	INT	Zbroj svih ocjena
rating_count	INT	Broj svih ocjenjivanja
email	VARCHAR	Email korisnika
has_dog	BOOLEAN	Ima li korisnik psa
has_experience	BOOLEAN	Ima li korisnik iskustvo
blocked	BOOLEAN	Je li korisnik blokiran

breed je entitet koji razlikuje sve pasmine u sustavu. Sastoji se od atributa: breed_id, name. Povezan je vezom *One-To-Many* s tablicom dog preko vlastitog atributa breed_id. Isto vrijedi i za tablicu request_dog.

breed		
breed_id	INT	Jedinstveni identifikator pasmine
name	VARCHAR	Naziv pasmine

dog je entitet koji sadrži sve važne informacije o psima vlasnika. Sastoji se od atributa: dog_id, name, rating_sum, rating_count, user_id, breed_id, date_of_birth, photo. Povezan je vezom *Many-To-One* s tablicom appuser preko atributa user_id tablice appuser i istom takvom vezom s tablicom breed preko atributa breed_id tablice breed. Vezom *One-To-One* povezan je s tablicom request_guardians_dog preko vlastitog atributa dog_id.

dog		
dog_id	INT	Jedinstveni identifikator psa
name	VARCHAR	Ime psa
rating_sum	INT	Zbroj svih ocjena
rating_count	INT	Broj svih ocjenjivanja
user_id	INT	Jedinstveni identifikator vlasnika psa (appuser.user_id)
breed_id	INT	Jedinstveni identifikator pasmine (breed.breed_id)
date_of_birth	DATE	Datum rođenja psa
photo	TEXT	Slika psa

request_dog je entitet koji sadrži sve važne informacije o oglasima koje čuvari pasa objavljuju. Sastoji se od atributa: request_dog_id, dog_age, dog_time_begin, dog_time_end, is_flexible, location, number_of_dogs, is_published, is_reviewed, breed_id, user_id, location_name. Povezan je vezom *Many-To-One* s tablicom breed preko atributa breed_id tablice breed i istom takvom vezom s tablicom appuser preko atributa user_id tablice appuser. Vezom *One-To-One* je povezan s tablicom agreed_request preko vlastitog atributa request_dog_id.

request_dog		
request_dog_id	INT	Jedinstveni identifikator oglasa za čuvanje pasa
dog_age	INT	Preferirana dob pasa
dog_time_begin	TIMESTAMP	Početak čuvanja pasa
dog_time_end	TIMESTAMP	Završetak čuvanja pasa
is_flexible	BOOLEAN	Je li vrijeme čuvanja fleksibilno
location	VARCHAR	Koordinate lokacije čuvanja pasa
number_of_dogs	INT	Broj pasa
is_published	BOOLEAN	Je li oglas objavljen
is_reviewed	BOOLEAN	Je li oglas pregledan od strane administratora
breed_id	INT	Jedinstveni identifikator pasmine (breed.breed_id)
user_id	INT	Jedinstveni identifikator čuvara pasa (appuser.user_id)
location_name	VARCHAR	Naziv lokacije čuvanja pasa

request_guardian je entitet koji sadrži sve važne informacije o zahtjevima koje vlasnici pasa objavljuju. Sastoji se od atributa: request_guardian_id, location, number_of_dogs, guard_time_begin, guard_time_end, is_published, is_reviewed, user_id, has_experience, has_dog, location_name. Povezan je vezom *Many-To-One* s tablicom appuser preko atributa user_id tablice appuser. S tablicama request_guardians_dog i request_activity povezan je vezom *One-To-Many* preko vlastitog atributa request_guardian_id. Vezom *One-To-One* je povezan s tablicom agreed_request preko vlastitog atributa request_guardian_id.

request_guardian		
request_guardian_id	INT	Jedinstveni identifikator zahtjeva za čuvanje pasa
location	VARCHAR	Koordinate lokacije čuvanja pasa

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

request_guardian		
number_of_dogs	INT	Broj pasa
guard_time_begin	TIMESTAMP	Početak čuvanja pasa
guard_time_end	TIMESTAMP	Završetak čuvanja pasa
is_published	BOOLEAN	Je li zahtjev objavljen
is_reviewed	BOOLEAN	Je li zahtjev pregledan od strane administratora
user_id	INT	Jedinstveni identifikator vlasnika pasa (appuser.user_id)
has_experience	BOOLEAN	Želi li vlasnik da čuvar ima iskustvo
has_dog	BOOLEAN	Želi li vlasnik da čuvar ima vlastitog psa
location_name	VARCHAR	Naziv lokacije čuvanja pasa

request_guardians_dog je entitet koji sadrži sve važne informacije o psima koji su povezani s pojedinim zahtjevima za čuvanje koje je objavio vlasnik pasa. Potreban je kako bi se unutar jednog zahtjeva za čuvanje moglo nalaziti više pasa koje je potrebno istovremeno čuvati. Sastoji se od atributa: request_guardians_dog_id, request_guardian_id, dog_id. Povezan je vezom *One-To-One* s tablicom dog preko atributa dog_id tablice dog. S tablicom request_guardian povezan je vezom *Many-To-One* preko atributa request_guardian_id tablice request_guardian.

request_guardians_dog		
request_guardians_dog_id	INT	Jedinstveni identifikator psa povezanog sa zahtjevom za čuvanje pasa
request_guardian_id	INT	Jedinstveni identifikator zahtjeva za čuvanje pasa (request_guardian.request_guardian_id)
dog_id	INT	Jedinstveni identifikator psa (dog.dog_id)

activity je entitet koji sadrži sve važne informacije o aktivnostima koje bi pas i

čuvar trebali raditi. Sastoji se od atributa: `activity_id`, `activity_name`. Povezan je vezom *One-To-Many* s tablicom `request_activity` preko vlastitog atributa `activity_id`.

activity		
activity_id	INT	Jedinstveni identifikator aktivnosti
activity_name	VARCHAR	Naziv aktivnosti

request_activity je entitet koji sadrži sve važne informacije o aktivnostima koje su povezane s pojedinim zahtjevima za čuvanje koje su objavili vlasnici pasa. Potreban je kako bi se unutar jednog zahtjeva za čuvanje moglo nalaziti više aktivnosti koje je potrebno raditi sa psima tijekom čuvanja. Sastoji se od atributa: `request_activity_id`, `activity_id`, `feeding_quantitiy`, `request_guardian_id`. Povezan je vezom *Many-To-One* s tablicom `activity` preko atributa `activity_id` tablice `activity` i istom takvom vezom s tablicom `request_guardian` preko atributa `request_guardian_id` tablice `request_guardian`.

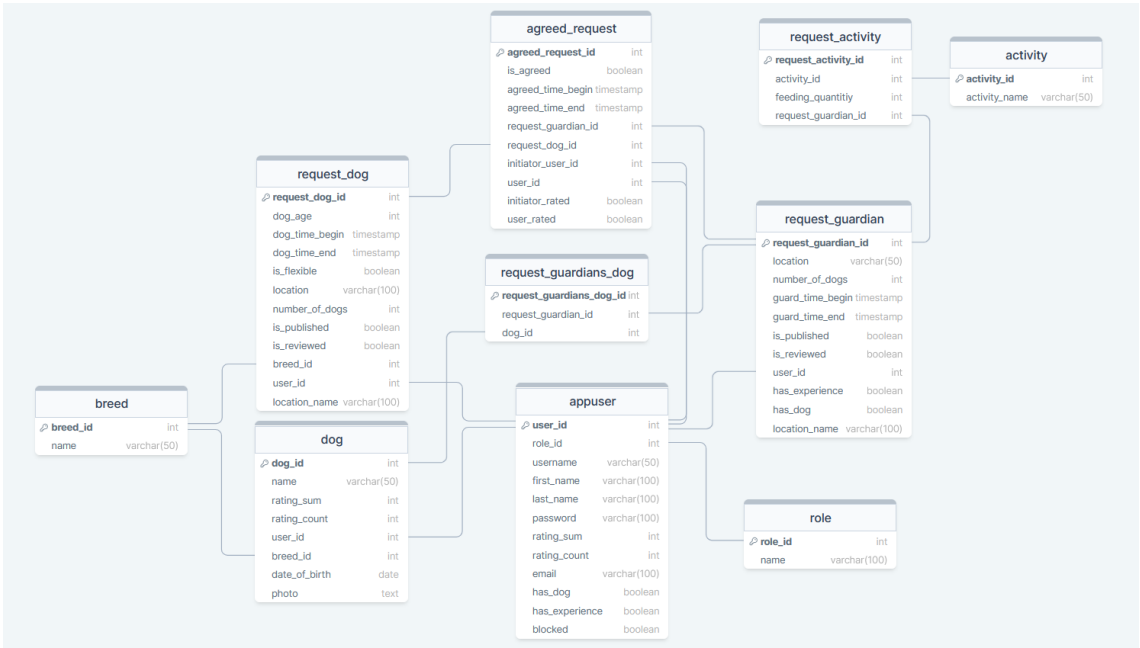
request_activity		
request_activity_id	INT	Jedinstveni identifikator aktivnosti povezane sa zahtjevom za čuvanje pasa
activity_id	INT	Jedinstveni identifikator aktivnosti (activity.activity_id)
feeding_quantity	INT	Količina hrane ukoliko je odabrana aktivnost koja uključuje hranu
request_guardian_id	INT	Jedinstveni identifikator zahtjeva za čuvanje pasa (request_guardian.request_guardian_id)

agreed_request je entitet koji sadrži sve važne informacije o dogovorenim/ nedogovorenim zahtjevima i oglasima za čuvanje od strane vlasnika i čuvara. Sastoji se od atributa: `agreed_request_id`, `is_agreed`, `agreed_time_begin`, `agreed_time_end`, `request_guardian_id`, `request_dog_id`, `initiator_user_id`, `user_id`, `initiator Rated`, `user Rated`. Povezan je vezom *One-To-One* s tablicom `request_guardian` preko atributa `request_guardian_id` tablice `request_guardian`, istom takvom vezom s tablicom `request_dog` preko atributa `request_dog_id` tablice `request_dog` te istom tak-

vom vezom s tablicom appuser preko atributa user_id tablice appuser. S tablicom appuser još je povezan vezom *Many-To-One* preko atributa initiator_user_id koji zapravo predstavlja atribut user_id tablice appuser.

agreed_request		
agreed_request_id	INT	Jedinstveni identifikator dogovorenih/ nedogovorenih zahtjeva i oglasa za čuvanje
is_agreed	BOOLEAN	Je li postignut dogovor
agreed_time_begin	TIMESTAMP	Dogovoren početak čuvanja
agreed_time_end	TIMESTAMP	Dogovoren kraj čuvanja
request_guardian_id	INT	Jedinstveni identifikator zahtjeva za čuvanje pasa (request_guardian.request_guardian_id)
request_dog_id	INT	Jedinstveni identifikator oglasa za čuvanje pasa (request_dog.request_dog_id)
initiator_user_id	INT	Jedinstveni identifikator korisnika koji je započeo interakciju (appuser.user_id)
user_id	INT	Jedinstveni identifikator korisnika s kojim je započeta interakcija (appuser.user_id)
initiatorRated	BOOLEAN	Je li korisnik koji je započeo interakciju ocijenio iskustvo
userRated	BOOLEAN	Je li korisnik s kojim je započeta interakcija ocijenio iskustvo

4.1.2 Dijagram baze podataka

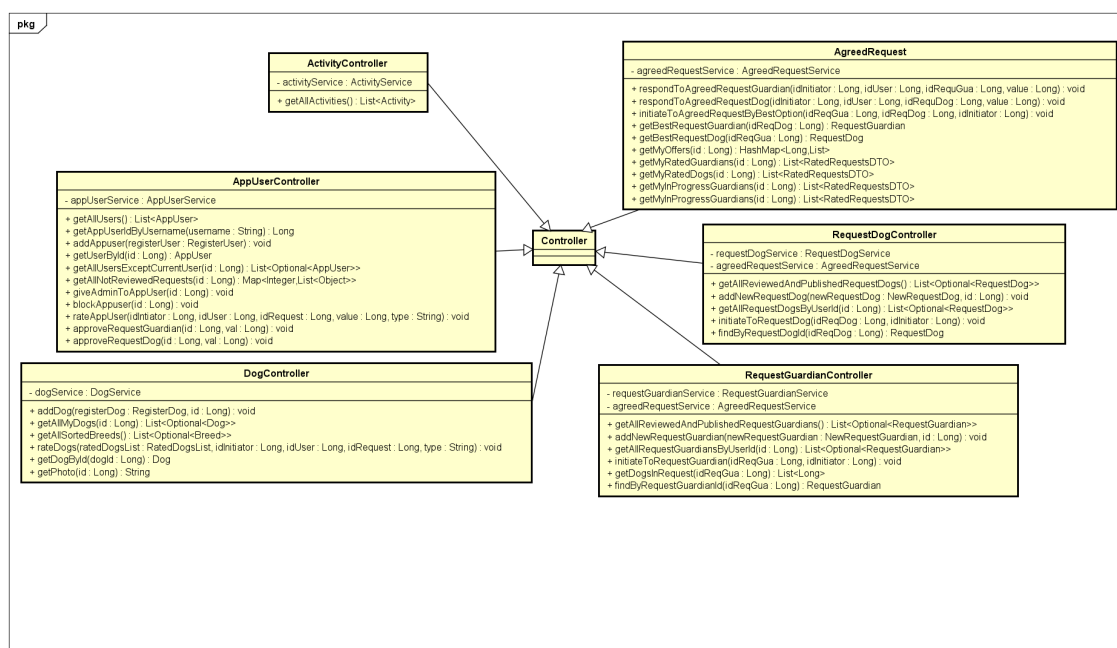


Slika 4.2: Relacijski dijagram baze podataka

4.2 Dijagram razreda

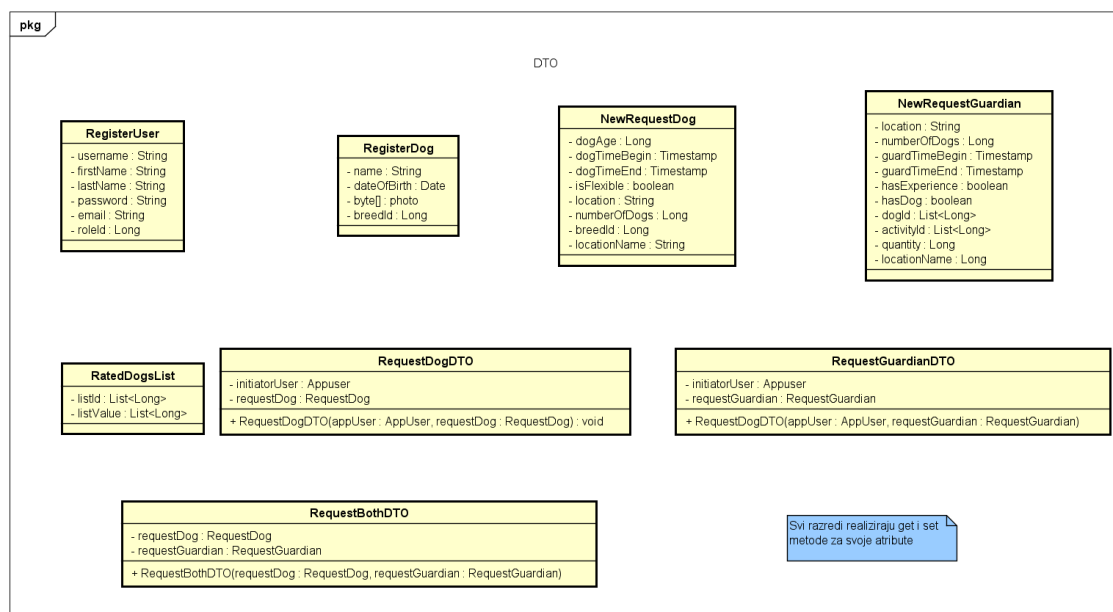
Na slikama 4.3, 4.4 i 4.5 prikazani su razredi koji pripadaju backend dijelu s arhitekturom podijeljenom na kontrolere, repozitorije i servise te uključuju Domain i DTO (Data Transfer Object) modele.

Na slici 4.3 prikazani su razredi koji nasljeđuju Controller razred. Metode implementirane u tim razredima manipuliraju s DTO-ima, a oni su dohvaćeni pomoću metoda implementiranih u Domain razredima.



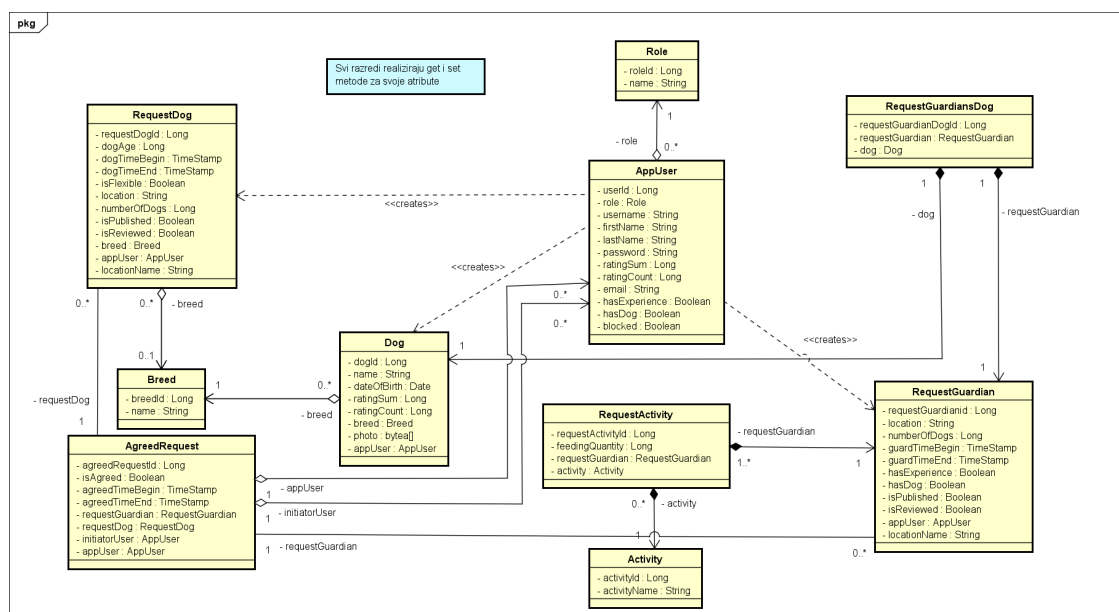
Slika 4.3: Dijagram razreda - dio Controllers

Data Transfer Objects služe za razmjenu podataka između procesa ili slojeva. RegisterUser i RegisterDog služe za stvaranje novog objekta User, odnosno Dog. NewRequestDog i NewRequestGuardian služe za stvaranje novog objekta RequestDog, odnosno RequestGuardian. RatedDogsList služi za primanje podataka o ocijenjenim psima. RequestGuardianDTO, RequestBothDTO, RatedRequestsDTO služe kao pomoćni objekti za čitanje tih objekata iz baze podataka.



Slika 4.4: Dijagram razreda - dio Data Transfer Objects

Domain razredi preslikavaju strukturu baze podataka u aplikaciji. Implementirane metode direktno komuniciraju s bazom podataka te vraćaju tražene podatke. Razred AppUser predstavlja korisnika web aplikacije koji se može registrirati unos-eći potrebne informacije. On može izabrati svoju ulogu pri registraciji (razred Role). Administrator je korisnik aplikacije koji ima sve mogućnosti razreda AppUser. Vlasnik pasa može dodati vlastitog psa u sustav (razred Dog). Vlasnik, odnosno čuvar mogu tražiti čuvara za svog pasa (razred RequestGuardian), odnosno psa za čuvanje (razred RequestDog). Vlasnik može i tražiti koju aktivnost će čuvar raditi sa psom (razredi Activity i RequestActivity). Dogovor vlasnika i čuvara predstavlja razred AgreedRequest. U RequestGuardiansDog se popisuje psi koju se koriste u zahtjevu RequestGuardian.

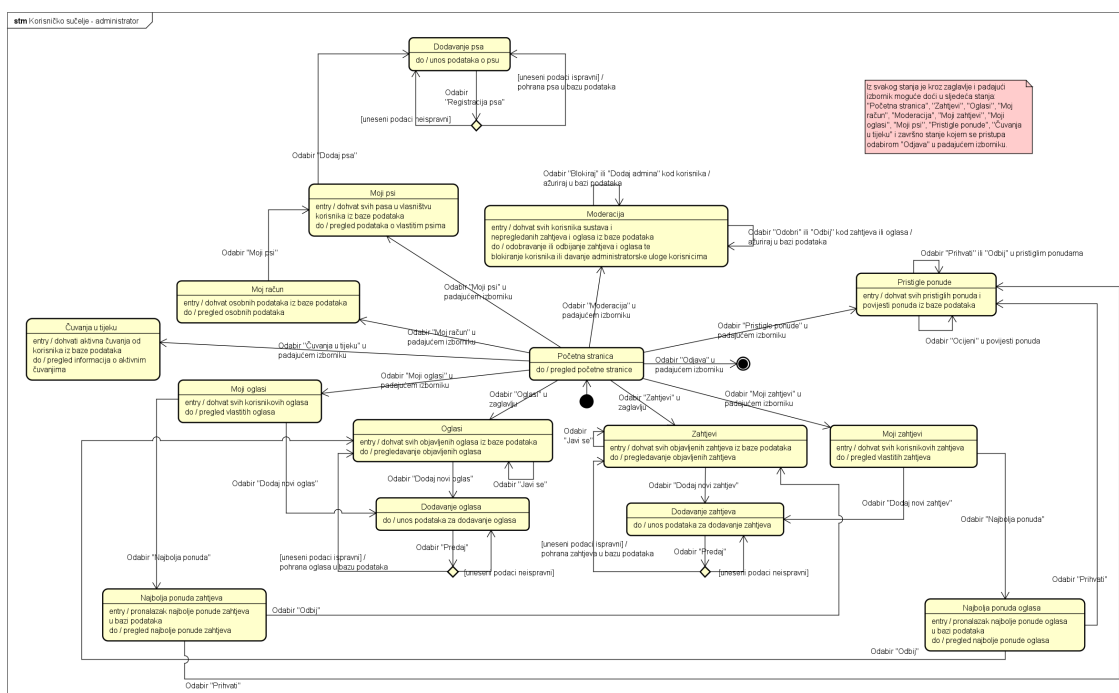


Slika 4.5: Dijagram razreda - dio Domain

4.3 Dijagram stanja

Dijagram stanja prikazuje stanja objekta te prijelaze među stanjima koji su potaknuti okidačima. Na slici 4.6 prikazan je dijagram stanja prijavljenog korisnika s ulogom administratora. Nakon prijave u sustav korisnik se nalazi na početnoj stranici na kojoj može pročitati više informacija o samom sustavu. Kroz zaglavlje aplikacije može pristupiti svim objavljenim zahtjevima i oglasima, a kroz padajući izbornik u zaglavlju može pristupiti svom računu, stranici za upravljanje korisnicima te zahtjevima i oglasima, vlastitim zahtjevima i oglasima, vlastitim psima, pristiglim ponudama, čuvanjima u tijeku te se može odjaviti iz sustava. Pošto se zaglavlje nalazi na svim stranicama aplikacije, korisnik prethodno navedenim opcijama ima pristup gdje god da se nalazi u aplikaciji. Na stranici sa svim objavljenim zahtjevima ima mogućnost dodavanja novog zahtjeva te mogućnost javljanja na pojedine zahtjeve, a na stranici sa svim objavljenim oglasima ima mogućnost dodavanja novog oglasa te mogućnost javljanja na pojedine oglase. Na stranici s vlastitim zahtjevima ima mogućnost odabira najbolje ponude oglasa za pojedini zahtjev koju potom može prihvatiti ili odbiti, a isto može i na stranici s vlastitim oglasima. Također na stranicama s vlastitim zahtjevima i oglasima ima mogućnost dodavanja novog zahtjeva, odnosno oglasa. Stranici za pregled vlastitih pasa korisnik može pristupiti kroz padajući izbornik, ali i kroz pregled vlastitog računa

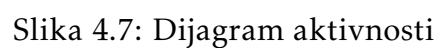
te potom tamo ima mogućnost dodavanja novog psa. Ponude za moguću suradnju, kao i protekle suradnje korisnik može vidjeti na stranici za pristigle ponude na kojoj onda može prihvaćati ili odbijati suradnju te ocjenjivati protekla iskustva. Ukoliko korisnik ima aktivnu suradnju, onda potrebnim informacijama može pristupiti putem stranice za pregled čuvanja u tijeku. Za kraj, korisnik ima pristup stranici za upravljanje zahtjevima i oglasima te korisnicima na kojoj onda može odobravati ili odbijati zahtjeve i oglase te blokirati korisnike ili im davati administratorsku ulogu.



Slika 4.6: Dijagram stanja

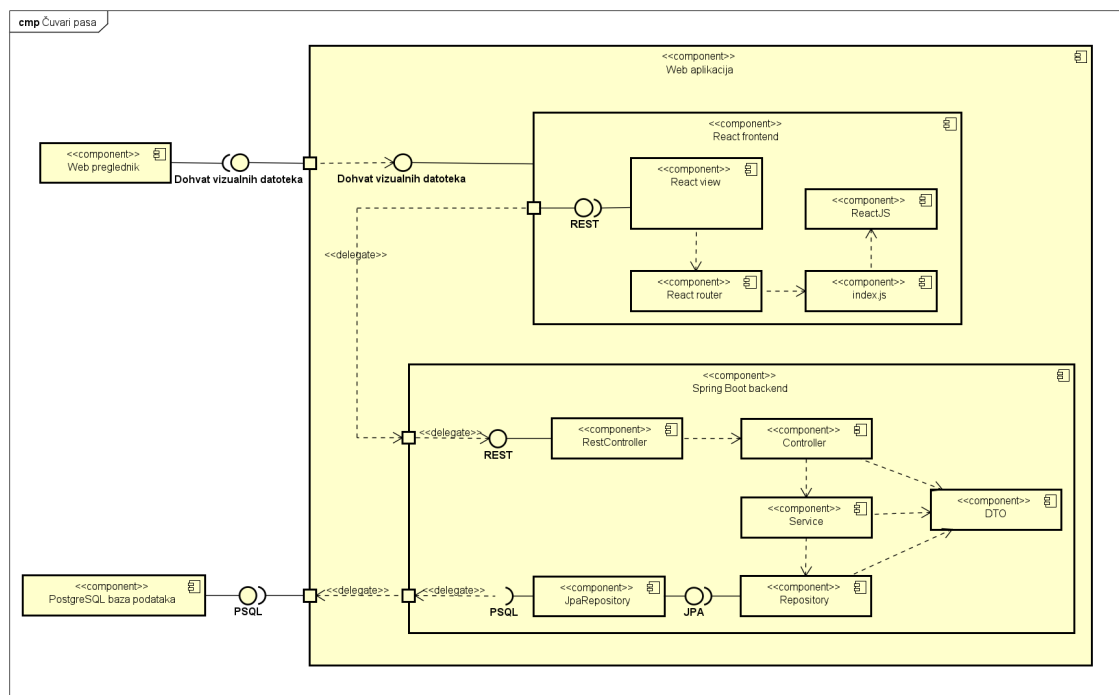
4.4 Dijagram aktivnosti

Dijagram aktivnosti primjenjuje se za opis modela toka upravljanja ili toka podataka. Ne upotrebljava se za modeliranje događajima poticanog ponašanja. U modeliranju toka upravljanja svaki novi korak poduzima se nakon završenog prethodnog, a naglasak je na jednostavnosti. Na dijagramu aktivnosti sa slike 4.7 prikazan je proces sinkronizacije vlasnika i čuvara, odobravanje upita te ocjenjivanje. Vlasnik, čuvar i administrator se prijave u sustav. Vlasnik dodaje svog psa te radi zahtjev dok čuvar radi svoj oglas. Ako je sve u redu administrator potvrđuje te upite i započinje sinkronizacija između vlasnika i čuvara. Nakon što je čuvanje završilo, vlasnik može ocijeniti čuvara dok čuvar može ocijeniti pse.



4.5 Dijagram komponenti

Dijagram komponenti je strukturni dijagram kojim se vizualizira organizacija i međuovisnost između implementacijskih komponenti te odnos programske potpore prema okolini. Dijagram komponenti na slici 4.8 sastoji se od komponente web preglednika, baze podataka, te same aplikacije koja ima svoje dvije glavne podkomponente, a to su frontend i backend. U web pregledniku se preko sučelja za dohvat vizualnih datoteka prikazuju pojedine stranice. O tome koja će se datoteka dohvatiti, odnosno stranica prikazati odlučuje React router komponenta frontenda na temelju URL-a kojem se pristupa. Frontend se još sastoji od komponenti React view, index.js i ReactJS koje su međusobno zavisne. ReactJS je sama biblioteka iz koje se dobivaju gotove komponente za prikaz. index.js služi kao početna komponenta unutar koje se nalazi organizirana hijerarhija ostalih elemenata za prikaz. React view komponenta pak komunicira s backendom preko REST API-ja i razmjenjuje podatke s backendom u JSON formatu, a ovisno o korisnikovim akcijama osvježava prikaz na stranici. Controller komponenta backenda prima zahtjeve, preovjerava ih te šalje odgovore na dobivene zahtjeve prema klijentskoj strani. Backend se još sastoji od komponente Service koja prima zahtjeve od Controllera, obrađuje ih i prosljeđuje komponenti Repository koja onda preko JPA sučelja komunicira sa PostgreSQL bazom podataka u koju se pohranjuju podaci.



Slika 4.8: Dijagram komponenti

5. Implementacija i korisničko sučelje

5.1 Korištene tehnologije i alati

Frontend aplikacije pisan je u programskom jeziku **JavaScript** uz pomoć biblioteke **React**. React je open-source JavaScript biblioteka za izgradnju komponenti korisničkog sučelja te je odgovorna samo za prezentacijski sloj aplikacije. Glavni cilj Reacta je razvoj korisničkog sučelja koje poboljšava brzinu aplikacija, pa stoga koristi virtualni DOM (Document Object Model) koji je brži od običnog DOM-a. Aplikacije pisane u Reactu su jednostranične što također pridonosi brzini. Korištenjem komponenti u Reactu, poboljšava se čitljivost i olakšava se održavanje većih aplikacija. React je razvijen od strane Facebooka te korišten i u njihovim aplikacijama poput Instagrama i WhatsAppa.

Kao razvojno okruženje za frontend aplikacije korišten je **Visual Studio Code**. Visual Studio Code je lagan, ali moćan uređivač izvornog koda razvijen od strane tvrtke Microsoft. Neke od glavnih značajki alata su podrška za ispravljanje pogrešaka, isticanje sintakse, inteligentno dovršavanje koda i mnoge druge. Dolazi s ugrađenom podrškom za JavaScript, TypeScript i Node.js, a ima i bogat ekosustav proširenja za druge jezike i okruženja kao što su C++, C#, Java, Python, PHP, Go, .NET itd.

Backend aplikacije pisan je u programskom jeziku **Java** uz pomoć radnog okvira **Spring Boot**. Spring Boot je vrlo popularan radni okvir za izgradnju samostojećih aplikacija koje koriste Spring. Spring Boot je specijalizacija radnog okvira Spring razvijen s ciljem jednostavnijeg i bržeg oblikovanja web aplikacija, pa stoga u svojoj automatskoj konfiguraciji olakšava posao programeru jer neke stvari, kao npr. servleti, rad s JSON datotekama, rad s bazama podataka itd., koje su karakteristične za većinu web aplikacija ima automatski podešeno.

Kao razvojno okruženje za backend aplikacije korišten je **IntelliJ IDEA**. IntelliJ IDEA je integrirano razvojno okruženje (IDE) razvijeno od strane tvrtke JetBrains. Samo razvojno okruženje pisano je u Javi s ciljem poboljšanog razvoja softvera u jezicima Java, Kotlin, Groovy i sl. Ovaj IDE pruža značajke kao što su dovršavanje koda analizom konteksta, navigacija u kodu pri čemu je moguće izravno skakanje

na klasu ili deklaraciju u kodu, refaktoriranje koda, otklanjanje pogrešaka i brojne druge. IntelliJ IDEA također podržava dodatke pomoću kojih se može ostvariti dodatna funkcionalnost. Dodaci se mogu preuzeti i instalirati putem njihovog web repozitorija dodataka ili putem IDE-ove ugrađene opcije instaliranja dodataka.

Baza podataka je izvedena u **PostgreSQL-u**. PostgreSQL je open-source sustav za upravljanje relacijskim bazama podataka (RDBMS) kojim se proširuje funkcionalnost SQL-a. PostgreSQL nudi transakcije s okidačima, stranim ključevima, pohranjenim procedurama, automatski ažuriranim prikazima i sl. Transakcije također imaju svojstva atomarnosti, konzistentnosti, izolacije i izdržljivosti (ACID). PostgreSQL dizajniran je da izdrži različita radna opterećenja, od pojedinačnih računala, pa sve do skladišta podataka ili web usluga s mnogo istodobnih korisnika.

Kao okruženje za upravljanje bazom podataka korišten je **pgAdmin**. pgAdmin je open-source grafički alat za administrativno upravljanje PostgreSQL bazama podataka.

Sama dokumentacija je pisana u jeziku **LaTeX**. LaTeX je jezik za pisanje strukturiranih tekstova profesionalne kvalitete. Za razliku od nekih programa za obradu teksta s grafičkim sučeljem poput Microsoft Worda, dokumenti se u LaTeXu pišu kao običan tekst s dodanom semantičkom strukturom te se time postiže usredotočenost na sadržaj, ujednačenost izgleda te brži i stabilniji rad.

Kao okruženje za pisanje dokumentaciju korišten je **TeXstudio**. TeXstudio je open-source integrirano okruženje za izradu LaTeX dokumenata. Posjeduje brojne značajke kao što su pogled na strukturu dokumenta, napredno isticanje sintakse, interaktivna provjera pravopisa, gramatike i referenci, jasan pogled na upozorenja i greške u dokumentu itd.

Za izradu UML dijagrama unutar dokumentacije korišteno je okruženje **Astah UML**. Astah UML je grafički alat koji je jednostavan za rukovanje te se koristi za izradu potrebnih UML dijagrama. Ima podršku za stvaranje brojnih dijagrama, a samo neki od njih su dijagrami obrazaca uporabe, sekvencijski dijagrami, dijagrami razreda, dijagrami stanja, dijagrami aktivnosti, dijagrami komponenti te dijagrami razmještaja za čiju je izradu alat i korišten.

Za izradu dijagrama baze podataka unutar dokumentacije korišten je online alat **DrawSQL**.

Kao sustav za upravljanje verzijama projekta korišten je **Git**. Git je open-source distribuirani sustav za upravljanje različitim verzijama datoteka. Obično se koristi za koordinaciju rada među programerima koji zajednički razvijaju neki softver.

Cilj Gita je brzina, integritet podataka i podrška za distribuirane i nelinearne tijekove rada. Svaki Git direktorij na bilo kojem računalu je spremište s potpunom poviješću i punim mogućnostima praćenja verzija.

Udaljeni repozitorij projekta dostupan je na web platformi u oblaku **GitLab**. GitLab je open-source end-to-end platforma u oblaku za razvoj softvera s ugrađenom kontrolom verzija, praćenjem problema, pregledom koda, CI/CD-om i više.

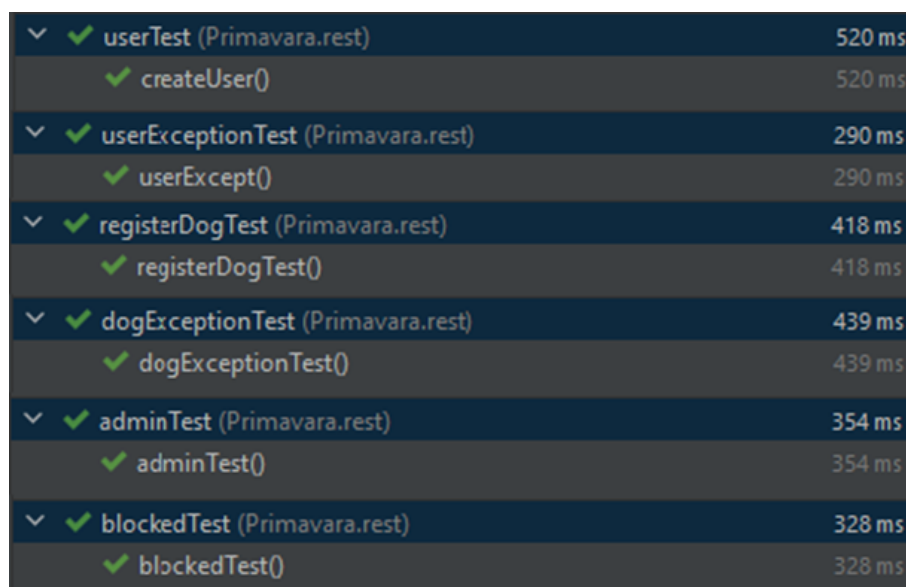
Za deploy aplikacije korišten je **Render**. Render je objedinjeni sustav u oblaku koji služi za izgradnju i pokretanje web aplikacija i aplikacija općenito. Pruža pogodnosti poput besplatnih TLS (Transport Layer Security) certifikata, globalnog CDN-a (Content Delivery Network), DDoS (Distributed Denial of Service) zaštite, privatnih mreža i automatske implementacije iz Gita.

Za komunikaciju u timu korišten je **Discord**. Discord je društvena platforma na kojoj korisnici imaju mogućnost komuniciranja glasovnim pozivima, videopozivima, tekstualnim porukama, medijima i datotekama u privatnim razgovorima ili kao dio zajednica koje se nazivaju serveri. Server je skup soba za razgovor i glasovnih kanala te mu je moguće pristupiti putem pozivnice.

5.2 Ispitivanje programskog rješenja

5.2.1 Ispitivanje komponenti

Za testiranje backenda naše aplikacije koristili smo JUnit testove. Svi su testovi bili uspješni, što se vidi na slici 5.1. U sljedećim potpoglavljima prikazani su pojedinačni testovi komponenta.



✓ userTest (Primavara.rest)	520 ms
✓ createUser()	520 ms
✓ userExceptionTest (Primavara.rest)	290 ms
✓ userExcept()	290 ms
✓ registerDogTest (Primavara.rest)	418 ms
✓ registerDogTest()	418 ms
✓ dogExceptionTest (Primavara.rest)	439 ms
✓ dogExceptionTest()	439 ms
✓ adminTest (Primavara.rest)	354 ms
✓ adminTest()	354 ms
✓ blockedTest (Primavara.rest)	328 ms
✓ blockedTest()	328 ms

Slika 5.1: Uspješni testovi komponenata

Test 1: Stvaranje korisnika

U ovom testu napravili poslali smo bazi gotovi objekt koji koristi za zapisivanje novog korisnika. Za provjeru testa dohvatili smo id iz baze i provjerili je li se taj korisnik ispravno zapisao u bazu.

```
@SpringBootTest
@TestInstance(TestInstance.Lifecycle.PER_CLASS)
public class UserTest {

    4 usages
    private String user = "asdasd17";

    1 usage
    private RegisterUser registerUser = new RegisterUser(user, firstName: "ivan", lastName: "kapusta", password: "asdasdasd", email: user + "@asd.com", roleId: 2L);

    3 usages
    @Autowired
    private AppUserService appUserService;

    @Autowired
    private RoleRepository roleRepository;

    ▲ Levo Matijic
    @Test
    void createUser(){
        appUserService.addAppUser(registerUser);
        Long id = appUserService.getIdByUsername(user);
        AppUser testUser = appUserService.getUserById(id);
        assertEquals(user, testUser.getUsername());
    }
}
```

Slika 5.2: Test 1: Stvaranje korisnika

Test 2: Stvaranje postojećeg korisnika

U ovom testu poslali smo bazi korisnika s korisničkim imenom koje već postoji u bazi. Potom smo provjerili baca li baza dobru vrstu iznimke te je li tekst te iznimke ispravan.

```
@SpringBootTest
@TestInstance(TestInstance.Lifecycle.PER_CLASS)
public class UserExceptionTest {

    3 usages
    private String user = "asd";

    1 usage
    private RegisterUser registerUser = new RegisterUser(user, firstName: "ivan", lastName: "kapusta", password: "asdasdasd", email: user + "@asd.com", roleId: 2L);

    1 usage
    @Autowired
    private AppUserService appUserService;

    @Autowired
    private RoleRepository roleRepository;

    ▲ Levo Matijic
    @Test
    void userExcept(){
        RequestDeniedException requestDeniedException = assertThrows(RequestDeniedException.class, () -> appUserService.addAppUser(registerUser));
        assertEquals("expected: 'AppUser with username ' + user + ' already exists", requestDeniedException.getMessage());
    }
}
```

Slika 5.3: Test 2: Stvaranje postojećeg korisnika

Test 3: Registracija novog psa

U ovom testu poslali smo bazi gotov objekt za registraciju novog psa. Zatim smo dohvatili sve pse korisnika kojem smo dodali psa, pa smo provjerili postoji li u toj listi pasa pas s istim imenom kao onaj kojeg smo poslali.

```
@SpringBootTest
@TestInstance(TestInstance.Lifecycle.PER_CLASS)
public class registerDogTest {

    1 usage
    Date date1= Date.valueOf("2020-01-01");
    3 usages
    String name = "flokittest";
    1 usage
    RegisterDog registerDog = new RegisterDog(name, date1, photo: "", breedid: 3L);

    2 usages
    @Autowired
    private DogService dogService;

    Lovro Malojcic
    @Test
    public void registerDogTest(){
        dogService.addDog(registerDog, id: 1L);
        List<Optional<Dog>> dogs = dogService.getAllMyDogs(id: 1L);
        String trazen = "";
        for (Optional<Dog> dog : dogs){
            if (Objects.equals(dog.get().getName(), name)){
                trazen = dog.get().getName();
                break;
            }
        }
        assertEquals(name, trazen);
    }
}
```

Slika 5.4: Test 3: Registracija novog psa

Test 4: Stvaranje psa s pogrešnim datumom rođenja

U ovom testu poslali smo bazi gotov objekt za registraciju novog psa, ali s datumom rođenja u budućnosti. Zatim smo provjerili baca li baza dobru vrstu iznimke te je li tekst te iznimke ispravan.

```
@SpringBootTest
@TestInstance(TestInstance.Lifecycle.PER_CLASS)
public class dogExceptionTest {

    1 usage
    Date date1= Date.valueOf("2024-01-01");
    1 usage
    String name = "aetrtte";
    1 usage
    RegisterDog registerDog = new RegisterDog(name, date1, photo: "", breedid: 3L);

    1 usage
    @Autowired
    private DogService dogService;

    Lovro Malojcic
    @Test
    public void dogExceptionTest(){
        RequestDeniedException exception = assertThrows(RequestDeniedException.class, () -> dogService.addDog(registerDog, id: 1L));
        assertEquals("expected: 'Date of birth must be in the past', exception.getMessage());
    }
}
```

Slika 5.5: Test 4: Stvaranje psa s pogrešnim datumom rođenja

Test 5: Davanje uloge administratora korisniku

U ovom testu poslali smo bazi id korisnika kojemu želimo dati ulogu administratora. Zatim smo korisnika s tim id-om dohvatili te provjerili odgovara li mu identifikacijski broj uloge ulozi administratora.

```
@SpringBootTest
@TestInstance(TestInstance.Lifecycle.PER_CLASS)
public class adminTest {
    2 usages
    Long id = 36L;

    2 usages
    @Autowired
    private AppUserService appUserService;

    Lovro Malojcic
    @Test
    void adminTest(){
        appUserService.giveAdminToAppUser(id);
        AppUser user = appUserService.getUserById(id);
        assertEquals( expected: 4L, user.getRole().getRoleId());
    }
}
```

Slika 5.6: Test 5: Davanje uloge administratora korisniku

Test 6: Blokiranje korisnika

U ovom testu poslali smo bazi id korisnika kojega želimo blokirati. Zatim smo korisnika s tim id-om dohvatili te provjerili vrijednost atributa blocked.

```
@SpringBootTest
@TestInstance(TestInstance.Lifecycle.PER_CLASS)
public class blockedTest {
    2 usages
    Long id = 35L;

    2 usages
    @Autowired
    private AppUserService appUserService;

    Lovro Malojcic
    @Test
    void blockedTest(){
        appUserService.blockAppUser(id);
        AppUser user = appUserService.getUserById(id);
        assertEquals( expected: true, user.getBlocked());
    }
}
```

Slika 5.7: Test 6: Blokiranje korisnika

5.2.2 Ispitivanje sustava

Ispitivanje sustava proveli smo pomoću Selenium ispita. Koristili smo Selenium WebDriver za preglednik Google Chrome.

Ispitni slučaj 1: Prijavljivanje korisnika

- **Ulaz:** Korisničko ime i lozinka korisnika
- **Očekivani izlaz:** Nakon prijave, nalaziti ćemo se na početnoj stranici, što je u našem slučaju `http://localhost:3000/`, s obzirom da smo testove obavljali lokalno
- **Koraci:**
 1. Korisnik odabere gumb 'Prijava'
 2. Korisnik unese korisničko ime
 3. Korisnik unese lozinku
 4. Korisnik odabere gumb 'Prijavi se'

```
async function loginTest(){
  let driver = await new Builder().forBrowser( name: "chrome").build()
  await driver.get("http://localhost:3000")

  await driver.findElement(By.linkText( text: "Prijava")).click()
  await sleep( milliseconds: 500)

  await driver.findElement(By.name( name: "username")).sendKeys("asd")
  await driver.findElement(By.name( name: "password")).sendKeys("asdasdasd")
  await driver.findElement(By.id( id: "login")).click()
  await sleep( milliseconds: 500)

  if (await driver.getCurrentUrl() == "http://localhost:3000/") {
    await driver.quit()
    console.log("uspjeh")
    return true;
  } else {
    await driver.quit()
    console.log("neuspjeh")
    return false;
  }
}

loginTest();
```

Slika 5.8: Ispitni slučaj 1: Prijavljivanje korisnika

Ispitni slučaj 2: Dodavanje novog administratora

- **Ulaz:** Korisničko ime i lozinka administratora
- **Očekivani izlaz:** Dobit ćemo obavijest od stranice koja nas obavještava da je korisnik uspješno dodan kao administrator
- **Koraci:**
 1. Korisnik odabere gumb 'Prijava'
 2. Korisnik unese korisničko ime
 3. Korisnik unese lozinku
 4. Korisnik odabere gumb 'Prijavi se'
 5. Korisnik, koji se nalazi na početnoj stranici, pozicionira miš iznad svojeg korisničkog imena kako bi otkrio padajući izbornik
 6. Korisnik odabire gumb 'Moderacija'
 7. Korisnik odabire gumb 'Dodaj admina'
 8. Korisnik provjerava tekst obavijesti


```
async function addAdminTest(){
  let driver = await new Builder().forBrowser( name: "chrome").build()
  await driver.get("http://localhost:3000")

  await driver.findElement(By.linkText( text: "Prijava")).click()
  await sleep( milliseconds: 500)

  await driver.findElement(By.name( name: "username")).sendKeys("asd")
  await driver.findElement(By.name( name: "password")).sendKeys("asdasdasd")
  await driver.findElement(By.id( id: "login")).click()
  await sleep( milliseconds: 500)

  const hoverable = driver.findElement(By.className( name: "dropdown-username"));
  const actions = driver.actions({async: true});
  await actions.move({origin: hoverable}).perform();

  await driver.findElement(By.linkText( text: "Moderacija")).click()
  await sleep( milliseconds: 500)

  const admins = await driver.findElements(By.className( name: "admin-button"))
  for (let admin of admins){
    if (admin.getText() != null){
      admin.click()
    }
  }

  await sleep( milliseconds: 500)

  if (await driver.switchTo().alert().getText() == "Uspješno davanje admina"){
    await driver.quit()
    console.log("uspjesno")
    return true;
  } else {
    await driver.quit()
    console.log("neuspjesno")
    return false;
  }
}
```

Slika 5.9: Ispitni slučaj 2: Dodavanje novog administratora

Ispitni slučaj 3: Registracija novog psa

- **Ulaz:** Korisničko ime, lozinka, ime psa, datum rođenja psa, identifikacijski broj vrste psa
- **Očekivani izlaz:** Korisnik će se nakon izvođenja nalaziti na stranici korisnikovih pasa, gdje će ga nakon registracije preusmjeriti samo ukoliko je registracija uspješna
- **Koraci:**
 1. Korisnik odabere gumb 'Prijava'
 2. Korisnik unese korisničko ime
 3. Korisnik unese lozinku

4. Korisnik odabere gumb 'Prijavi se'
5. Korisnik, koji se nalazi na početnoj stranici, pozicionira miš iznad svojeg korisničkog imena kako bi otkrio padajući izbornik
6. Korisnik odabire gumb 'Moji psi'
7. Korisnik odabire gumb 'Dodaj psa'
8. Korisnik unosi ime i datum rođenja psa
9. Korisnik odabire vrstu psa
10. Korisnik odabire gumb 'Registracija psa'

```
async function registerDogTest(){
  let driver = await new Builder().forBrowser( name: "chrome").build()
  await driver.get("http://localhost:3000")

  await driver.findElement(By.LinkText( text: "Prijava")).click()
  await sleep( milliseconds: 500)

  await driver.findElement(By.name( name: "username")).sendKeys("asd")
  await driver.findElement(By.name( name: "password")).sendKeys("asdasd")
  await driver.findElement(By.id( id: "login")).click()
  await sleep( milliseconds: 500)

  const hoverable = driver.findElement(By.className( name: "dropdown-username"));
  const actions = driver.actions({async: true});
  await actions.move({origin: hoverable}).perform();

  await driver.findElement(By.LinkText( text: "Moji psi")).click()
  await sleep( milliseconds: 500)

  await driver.findElement(By.LinkText( text: "Dodaj psa")).click()
  await sleep( milliseconds: 500)

  await driver.findElement(By.id( id: "name")).sendKeys("testnprizjer2")
  await driver.findElement(By.id( id: "dateOfBirth")).sendKeys("00\t1991")

  await driver.findElement(By.id( id: "breed")).click()
  await driver.findElement(By.id( id: "15")).click()

  await driver.findElement(By.className( name: "button button-primary")).click()
  await sleep( milliseconds: 500)

  if (await driver.getCurrentUrl() == "http://localhost:3000/users/dogs") {
    await driver.quit()
    console.log("uspjeh")
    return true;
  } else {
    await driver.quit()
    console.log("neuspjeh")
    return false;
  }
}
```

Slika 5.10: Ispitni slučaj 3: Registracija novog psa

Ispitni slučaj 4: Dodavanje nove ponude

- **Ulaz:**
 - Korisničko ime i lozinka
 - Datum i vrijeme početka i kraja čuvanja, adresa, željeni broj pasa i željena dob psa
- **Očekivani izlaz:** Korisnik će se nakon izvođenja nalaziti na stranici korisni-

kovih oglasa, gdje će biti preusmjeren samo ako će unos biti uspješan

- **Koraci:**

1. Korisnik odabere gumb 'Prijava'
2. Korisnik unese korisničko ime
3. Korisnik unese lozinku
4. Korisnik odabere gumb 'Prijavi se'
5. Korisnik odabire gumb 'Oglasi'
6. Korisnik odabire gumb 'Dodaj novi oglas'
7. Korisnik unosi datum i vrijeme početka i kraja čuvanja, adresu, željeni broj pasa i željenu dob psa
8. Korisnik odabire gumb 'Predaj'

```
async function newOfferTest(){
    let driver = await new Builder().forBrowser( name: "chrome").build()
    await driver.get("http://localhost:3000")

    await driver.findElement(By.LinkText( text: "Prijava")).click()
    await sleep( milliseconds: 500)

    await driver.findElement(By.name( name: "username")).sendKeys("asd")
    await driver.findElement(By.name( name: "password")).sendKeys("asdasdasd")
    await driver.findElement(By.id( id: "login")).click()
    await sleep( milliseconds: 500)

    await driver.findElement(By.LinkText( text: "Oglašaj")).click()
    await sleep( milliseconds: 500)

    await driver.findElement(By.LinkText( text: "Dodaj novi oglas")).click()
    await sleep( milliseconds: 500)

    await driver.findElement(By.id( id: "startDate")).sendKeys("06\t2023\t27a")
    await driver.findElement(By.id( id: "endDate")).sendKeys("06\t2023\t27p")

    await driver.findElement(By.id( id: "address")).sendKeys("Ulica Brune Bušića 30")
    await driver.findElement(By.id( id: "dogAge")).sendKeys("4")
    await driver.findElement(By.id( id: "numberOfDogs")).sendKeys("2")

    await driver.findElement(By.className( name: "button button-primary")).click()
    await sleep( milliseconds: 500)

    if (await driver.getCurrentUrl() == "http://localhost:3000/users/offers") {
        await driver.quit()
        console.log("uspjeh")
        return true;
    } else {
        await driver.quit()
        console.log("neuspjeh")
        return false;
    }
}
```

Slika 5.11: Ispitni slučaj 4: Dodavanje nove ponude

Ispitni slučaj 5: Neispravna prijava

- **Ulaz:** Neispravno korisničko ime i/ili lozinka
- **Očekivani izlaz:** Dobit ćemo obavijest od aplikacije s tekстом 'Unauthorized'
- **Koraci:**
 1. Korisnik odabere gumb 'Prijava'
 2. Korisnik unese korisničko ime
 3. Korisnik unese lozinku
 4. Korisnik odabere gumb 'Prijavi se'

```
async function loginErrorTest(){
  let driver = await new Builder().forBrowser( name: "chrome").build()
  await driver.get("http://localhost:3000")

  await driver.findElement(By.linkText( text: "Prijava")).click()
  await sleep( milliseconds: 500)

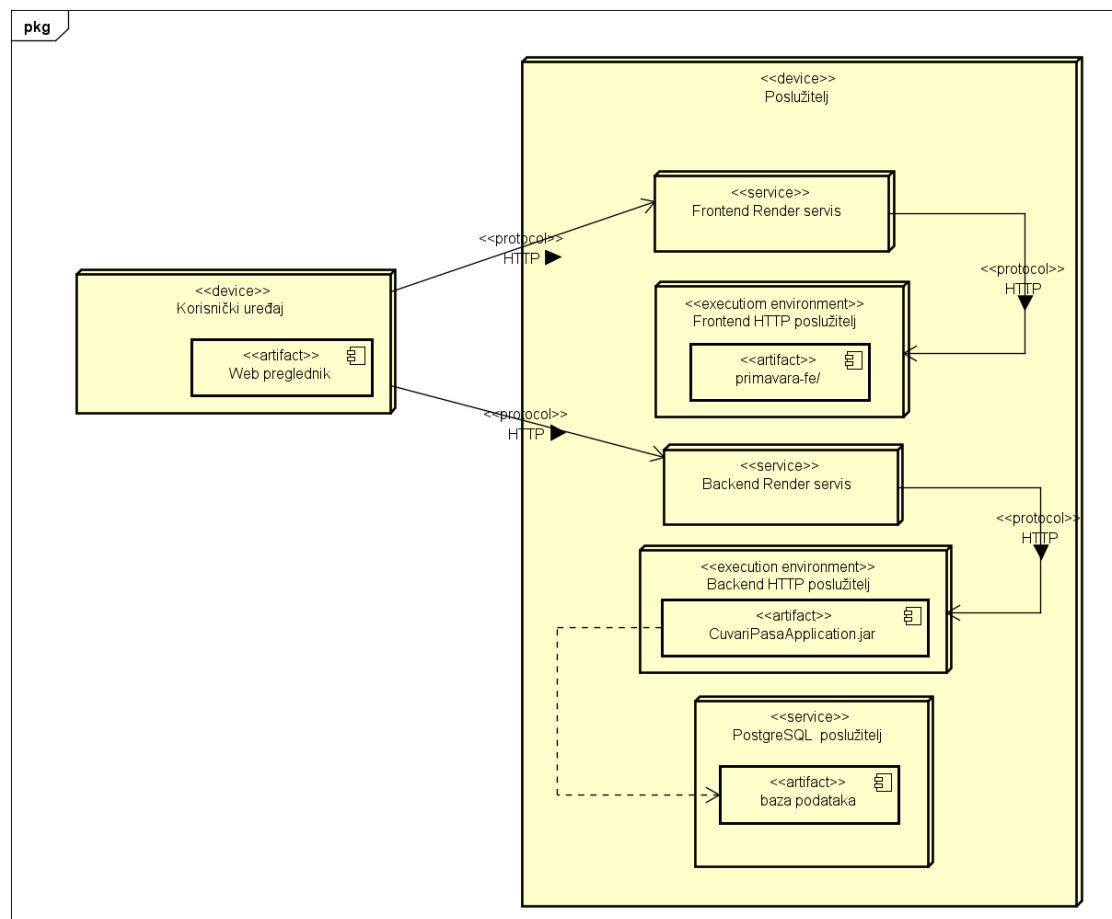
  await driver.findElement(By.name( name: "username")).sendKeys("tes")
  await driver.findElement(By.name( name: "password")).sendKeys("aaaaaaaaaaaa")
  await driver.findElement(By.id( id: "login")).click()
  await sleep( milliseconds: 500)

  if (await driver.switchTo().alert().getText() == "Unauthorized"){
    await driver.quit()
    console.log("uspjesno")
    return true;
  } else {
    await driver.quit()
    console.log("neuspjesno")
    return false;
  }
}
```

Slika 5.12: Ispitni slučaj 5: Neispravna prijava

5.3 Dijagram razmještaja

Dijagram razmještaja identificira fizičke i virtualne čvorove koji su prisutni u topologiji sustava te opisuje programsku potporu u implementaciji sustava. Sam sustav baziran je na odnosu klijent-poslužitelj + HTTP protokol za komunikaciju između njih. Cijela poslužiteljska strana ostvarena je pomoću servisa Render.

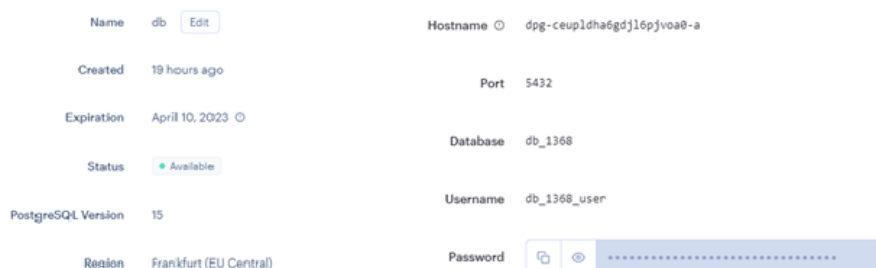


Slika 5.13: Dijagram razmještaja

5.4 Upute za puštanje u pogon

5.4.1 Konfiguracija poslužitelja baze podataka

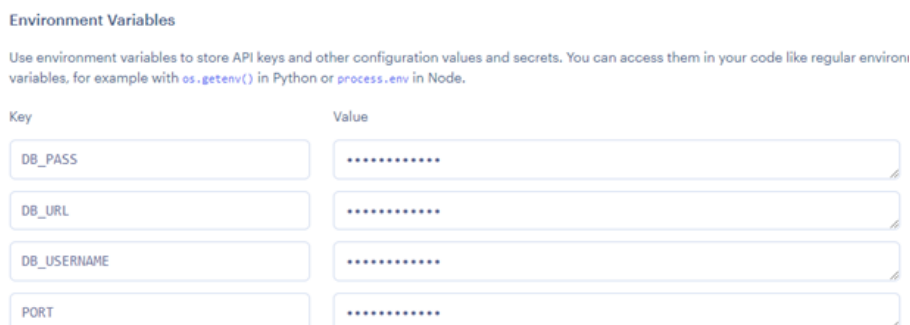
Na poslužitelju Render potrebno je konfigurirati PostgreSQL bazu podataka. Potrebno je postaviti ime baze, korisničko ime korisnika baze, regiju postaviti na Frankfurt (EU Central) i kliknuti Create Database. Pri prvom pokretanju backenda, automatski će se kreirati svi elementi baze Flyway migracijama.



Slika 5.14: Konfiguracija baze na Renderu

5.4.2 Konfiguracija backenda

Na poslužitelju Render potrebno je konfigurirati backend. Potrebno je povezati GitLab račun s Renderom. Nakon toga potrebno je kreirati novi servis i odabrati projekt Čuvari pasa. Za regiju je potrebno odabrati Frankfurt (EU Central), za granu main, za root directory primavara-be. Environment je potrebno postaviti na Docker. Dockerfile Path je ./docker/maven/Dockerfile, a Docker Build Context Directory je . Potrebno je postaviti environment varijable sa slike 5.3 i nakon toga kliknuti Create Web Service.



Slika 5.15: Konfiguracija backenda na Renderu

5.4.3 Konfiguracija frontenda

Na poslužitelju Render potrebno je konfigurirati frontend. Potrebno je povezati GitLab račun s Renderom. Nakon toga potrebno je kreirati novi servis i odabrati projekt Čuvari pasa. Za regiju je potrebno odabrati Frankfurt (EU Central), za granu main, za root directory primavara-fe. Environment je potrebno postaviti na Node. Build command je potrebno postaviti na yarn build, a start command na yarn start-prod. Potrebno je postaviti environment varijable sa slike 5.4 i nakon toga kliknuti Create Web Service.

Key	Value
API_BASE_URL

Slika 5.16: Konfiguracija frontenda na Renderu

6. Zaključak i budući rad

Zadatak našeg tima bio je razvoj web aplikacije za traženje i pružanje usluga čuvanja pasa te međusobnu interakciju korisnika prije početka i nakon završetka čuvanja. U protekla 3 mjeseca, svakog tjedna smo radili na razvoju projekta te time ostvarili glavninu funkcionalnosti koje su bile na početku planirane, a usput smo stekli brojna nova i korisna iskustva. Sami razvoj projekta bio je podijeljen u dvije faze, od kojih je prva bila više organizacijske naravi, dok je druga bila potpuno fokusirana na razvoj i implementaciju samog rješenja.

Na početku prve faze slijedilo je okupljanje tima te međusobno upoznavanje, a zatim smo dobili uvid u projektni zadatak koji je bilo potrebno provesti u djelo. Potom smo uz nekoliko timskih sastanaka putem platforme Discord razjasnili i prokomentirali samu suštinu problema te smo na vrlo visokoj razini apstrakcije izradili izgled nekih stranica aplikacije kako bi svi članovi tima dobili dojam o tome kako bi naša aplikacija otprilike funkcionirala. Nakon toga smo krenuli na podjelu poslova i organizaciju u pojedinim podtimovima. Tim je bio podijeljen na podtime zadužene za razvoj frontenda i backenda te podtim koji se bavi povezivanjem odgovarajućih frontend i backend komponenti. Detaljnije funkcionalnosti koje aplikacija mora obavljati razradili smo kroz funkcionalne zahtjeve i obrasce uporabe te pripadajuće dijagrame koji su nam uvelike pomogli i olakšali daljnji razvoj aplikacije. Pošto većina članova tima prethodno nije bila upoznata s pojedinim tehnologijama koje smo planirali koristiti za razvoj, kroz niz tehničkih radionica i samostalno istraživanje polako smo se upoznavali sa potrebnim alatima i načinom na koji ih trebamo koristiti. Na kraju prve faze ostvarili smo generičke funkcionalnosti kao što su prijava i registracija u sustav te pripadajući dizajn i izgled tih stranica.

Dolazimo do druge faze u kojoj stvari postaju malo zahtjevnije zbog nekih faktora kao što su količina funkcionalnosti koje je još bilo potrebno ostvariti te vremenski period koji nam je bio na raspolaganju za njihovo ostvarenje. U narednim tjednima rad na aplikaciji i samoj dokumentaciji bio je mnogo intenzivniji. Preostalo je dokumentirati podosta poglavlja te izraditi dijagrame koji odgovaraju radu naše aplikacije. Uz to, kao i uvijek, naišli smo i na neke probleme u razvoju koji su izi-

skivali puno vremena da se nađe odgovarajuće rješenje. Usprkos tome, na vrijeme smo bili gotovi s ostvarenjem planiranih funkcionalnosti naše aplikacije i nakon toga preostalo je dovršiti izgled same aplikacije te neke sitne preinake kako bi sve bilo ispravno i u funkciji.

Na samom kraju, zadovoljni smo onime što smo postigli i napravili. Svakako najbitnije od svega je iskustvo koje smo stekli tokom rada na ovom projektu. Timski rad te međusobna komunikacija i organizacija je ono što nam je pomoglo da dođemo do završnog proizvoda. Izrada samog projekta vjerojatno bi bila dosta brža i bezbolnija da smo prethodno bili upoznati s korištenim tehnologijama. No, zato smo kroz ovaj projekt dobili potrebna znanja i vjetar u leđa za buduće projekte.

Popis literature

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

Indeks slika i dijagrama

3.1	Dijagram obrasca uporabe - Funkcionalnost neregistriranog korisnika i registriranog korisnika (vlasnika pasa i čuvara pasa)	23
3.2	Dijagram obrasca uporabe - Funkcionalnost administratora	24
3.3	Sekvencijski dijagram za UC12	25
3.4	Sekvencijski dijagram za UC15	26
3.5	Sekvencijski dijagram za UC24	27
3.6	Sekvencijski dijagram za UC26	28
4.1	Dijagram arhitekture web aplikacije	31
4.2	Relacijski dijagram baze podataka	38
4.3	Dijagram razreda - dio Controllers	39
4.4	Dijagram razreda - dio Data Transfer Objects	40
4.5	Dijagram razreda - dio Domain	41
4.6	Dijagram stanja	42
4.7	Dijagram aktivnosti	44
4.8	Dijagram komponenti	46
5.1	Uspješni testovi komponenata	50
5.2	Test 1: Stvaranje korisnika	51
5.3	Test 2: Stvaranje postojećeg korisnika	51
5.4	Test 3: Registracija novog psa	52
5.5	Test 4: Stvaranje psa s pogrešnim datumom rođenja	52
5.6	Test 5: Davanje uloge administratora korisniku	53
5.7	Test 6: Blokiranje korisnika	54
5.8	Ispitni slučaj 1: Prijavljivanje korisnika	55
5.9	Ispitni slučaj 2: Dodavanje novog administratora	56
5.10	Ispitni slučaj 3: Registracija novog psa	58
5.11	Ispitni slučaj 4: Dodavanje nove ponude	60
5.12	Ispitni slučaj 5: Neispravna prijava	61
5.13	Dijagram razmještaja	62
5.14	Konfiguracija baze na Renderu	63

5.15 Konfiguracija backenda na Renderu	63
5.16 Konfiguracija frontenda na Renderu	64

Dodatak: Prikaz aktivnosti grupe

Dnevnik sastajanja

1. sastanak

- Datum: 20. listopada 2022.
- Prisustvovali: Mario Petek, Ivan Kuzmić, Lovro Malojčić, Antonio Lukić, Ivan Kapusta, Eugen Preglej, Sven Leko
- Teme sastanka:
 - sastanak s asistentom i demonstratorom
 - diskusija o radu na projektu
 - odabir tehnologije

2. sastanak

- Datum: 23. listopada 2022.
- Prisustvovali: Mario Petek, Ivan Kuzmić, Lovro Malojčić, Antonio Lukić, Ivan Kapusta, Eugen Preglej, Sven Leko
- Teme sastanka:
 - sastavljanje svih ekrana
 - općeniti izgled

3. sastanak

- Datum: 27. listopada 2022.
- Prisustvovali: Mario Petek, Antonio Lukić, Lovro Malojčić
- Teme sastanka:
 - sastanak s asistenticom
 - pojašnjenje zadatka
 - poboljšanje baze

4. sastanak

- Datum: 12. studenoga 2022.
- Prisustvovali: Mario Petek, Ivan Kuzmić, Antonio Lukić, Ivan Kapusta, Eugen Preglej, Sven Leko, Lovro Malojčić
- Teme sastanka:

- pregled dosadašnjega rada
- prijenos međusobnog znanja ostalim članovima
- definiranje završnog rada za prvu verziju

5. sastanak

- Datum: 17. studenoga 2022.
- Prisustvovali: Mario Petek, Ivan Kuzmić, Antonio Lukić, Ivan Kapusta, Eugen Preglej, Sven Leko, Lovro Malojčić
- Teme sastanka:
 - prezentiranje početne stranice
 - login
 - logout
 - deploy

6. sastanak

- Datum: 05. prosinca 2022.
- Prisustvovali: Mario Petek, Ivan Kuzmić, Antonio Lukić, Ivan Kapusta, Eugen Preglej, Sven Leko, Lovro Malojčić
- Teme sastanka:
 - prvo kolokviranje
 - revizija napravljene aplikacije

7. sastanak

- Datum: 22. prosinca 2022.
- Prisustvovali: Mario Petek, Ivan Kuzmić, Antonio Lukić, Ivan Kapusta, Eugen Preglej, Sven Leko, Lovro Malojčić
- Teme sastanka:
 - prezentacija alfa inačice aplikacije
 - dogovor za završnu verziju aplikacije

8. sastanak

- Datum: 05. siječnja 2023.
- Prisustvovali: Antonio Lukić, Ivan Kapusta, Sven Leko
- Teme sastanka:
 - diskusija vezana oko basicAuth
 - rješavanje problema kod učitavanja i prikazivanja slike

Tablica aktivnosti

	Antonio Lukić	Ivan Kapusta	Ivan Kuzmić	Mario Petek	Lovro Maločaj	Sven Leko	Eugen Preglej
Upravljanje projektom	15	2	2	2	2	2	2
Opis projektnog zadatka	0	0	1	1	0	0	0
Funkcionalni zahtjevi	0	0	3	3	0	0	0
Opis pojedinih obrazaca	0	0	1	1	0	0	0
Dijagram obrazaca	0	0	2	2	0	0	0
Sekvencijski dijagrami	0.5	0	2	2	0	0	0
Opis ostalih zahtjeva	0	0	0.5	0.5	0	0	0
Arhitektura i dizajn sustava	0	0	3	0	0	0	0
Baza podataka	0	0	4	0	0	0	0
Izrada baze podataka	4	0	0	0	0	0	0
Dijagram razreda	3	0	1	0	0	0	0
Dijagram stanja	0	0	0	0	0	0	0
Dijagram aktivnosti	0	0	0	0	0	0	0
Dijagram komponenti	0	0	0	0	0	0	0
Korištene tehnologije i alati	0	0	0	0	0	0	0
Ispitivanje programskog rješenja	0	0	0	0	0	0	0
Dijagram razmještaja	0	0	0	0	0	0	0
Upute za puštanje u pogon	0	0	0	0	0	0	0
Dnevnik sastajanja	2	0	0.5	0.5	0	0	0
Zaključak i budući rad	0	0	0	0	0	0	0

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

	Antonio Lukić	Ivan Kapusta	Ivan Kuzmić	Mario Petek	Lovro Malojčić	Sven Leko	Eugen Preglej
Popis literature	0	0	0.5	0.5	0	0	0
Izrada početne stranice	0	0	0	0	0	0	15
Spajanje s bazom podataka	0.5	3	0	0	0	3	0
Setup back-enda	5	2	0	2	5	2	0
Setp front-enda	0	1	0	0	8	1	8
Deploy	2	20	0	0	0	20	0

Dijagrami pregleda promjena

dio 2. revizije

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