

Programsko inženjerstvo

Ak. god. 2020./2021.

Humanitarni šetači pasa

Dokumentacija, Rev. 1

Grupa: *Cyfer*

Voditelj: *Jana Juroš*

Datum predaje: 14.1.2021.

Nastavnici: Doria Bukić, Hrvoje Šimić

Sadržaj

1 Dnevnik promjena dokumentacije	3
2 Opis projektnog zadatka	5
2.1 Motivacija i cilj	5
2.2 Postojeća slična rješenja	5
2.3 Opseg projektnog zadatka	7
3 Specifikacija programske potpore	9
3.1 Funkcionalni zahtjevi	9
3.1.1 Obrasci uporabe	12
3.1.2 Sekvencijski dijagrami	23
3.2 Ostali zahtjevi	30
4 Arhitektura i dizajn sustava	31
4.1 Baza podataka	35
4.1.1 Opis tablica	35
4.1.2 Dijagram baze podataka	39
4.2 Dijagram razreda	40
4.3 Dijagram stanja	42
4.4 Dijagram aktivnosti	43
4.5 Dijagram komponenti	44
5 Implementacija i korisničko sučelje	45
5.1 Korištene tehnologije i alati	45
5.2 Ispitivanje programskog rješenja	46
5.2.1 Ispitivanje komponenti	46
5.2.2 Ispitivanje sustava	61
5.3 Dijagram razmještaja	71
5.4 Upute za puštanje u pogon	72
5.4.1 Puštanje u pogon backend aplikacije	72
5.4.2 Dodavanje baze podataka na Heroku	73

5.4.3 Puštanje u pogon frontend aplikacije	73
6 Zaključak i budući rad	76
Popis literature	78
Indeks slika i dijagrama	80
Dodatak: Prikaz aktivnosti grupe	81

1. Dnevnik promjena dokumentacije

Rev.	Opis promjene/dodatka	Autori	Datum
0.1	Napravljen predložak.	Juroš	22.10.2020.
0.2	Napisan opis projektnog zadatka	Juroš	27.10.2020.
0.3	Dodani funkcionalni zahtjevi	Juroš	1.11.2020
0.3.1	Dodan prvi dio obrazaca uporabe	Juroš	3.11.2020
0.3.2	Dodan ostatak obrazaca uporabe	Bokarica, Almer, Presečki, Lukač, Sabalić, Lisica, Juroš	4.11.2020.
0.4	Dodani dijagrami obrazaca uporabe	Juroš	10.11.2020.
0.5	Dodani nefunkcionalni zahtjevi i arhitektura sustava	Juroš	10.11.2020.
0.6	Dodan sekvencijski dijagram - Promjena osobnih podataka korisnika	Lukač	10.11.2020.
0.6.1	Dodan ostatak sekvencijskih dijagrama	Juroš	11.11.2020.
0.7	Dodani dijagrami i opisi razreda	Bokarica, Lisica	11.11.2020.
0.8	Dodani dijagrami i opisi baze	Bokarica, Lisica	12.11.2020.
1.0	Korigiranje teksta i provjera dokumentacije	Juroš	13.11.2020.
1.1	Ispravak arhitekture	Juroš	14.1.2021.
1.2	Dodano ispitivanje programskog rješenja	Juroš	14.1.2021.
1.3	Dodan dijagram stanja	Lukač	14.1.2021.
1.4	Dodan dijagram razmještaja	Sabalić	14.1.2021.

Rev.	Opis promjene/dodataka	Autori	Datum
1.5	Dodan dijagram komponenti	Almer	14.1.2021.
1.6	Dodan dijagram aktivnosti	Presečki	14.1.2021.
1.7	Dodane upute za puštanje u pogon	Lisica	14.1.2021.
1.8	Dodan zaključak	Juroš	14.1.2021.
1.9	Dodani dijagrami promjena	Juroš	14.1.2021.
1.91	Dodane korištene tehnologije i alati	Juroš	14.1.2021.
2.0	Konačni tekst predloška dokumentacije	Juroš	14.1.2021.

2. Opis projektnog zadatka

2.1 Motivacija i cilj

Cilj ovog projekta je stvoriti aplikaciju koja može na jednostavan i brz način spojiti građane i udruge za životinje kako bi se organizirale šetnje nezbrinutih pasa.

Naime, broj nezbrinutih životinja raste u Hrvatskoj te je procijenjeno da ima 10 000 napuštenih životinja. Skloništa i udruge za nezbrinute životinje spašavaju ranjene i nezbrinute životinje, te žele potaknuti građane na angažiranost, pomoći pri brizi za životinje i za njihove udomljavanje. Šetnja je psima primarna potreba, a skloništa nemaju dovoljno resursa za hranu, a kamoli za plaćanje šetnji. Pretpostavljamo da bi se ljudi rado uključili u šetanje pasa kad bi postojao način, pogotovo ako planiraju usvojiti jednoga. Naša aplikacija im to odsad može i omogućiti.

2.2 Postojeća slična rješenja

Što se tiče postojećih rješenja iste tematike, ne postoji previše sličnosti s našim projektom. Postoje 3 web-stranice koje mogu spojiti šetače s vlasnicima pasa, no vlasnici plaćaju za te usluge te udruge nisu uključene. Prve dvije funkcioniraju na temelju oglasa - korisnik stavi oglas sa slikom svog kućnog ljubimca, vrijeme kad mu je potrebna usluga te koliko će platiti. Te stranice su:

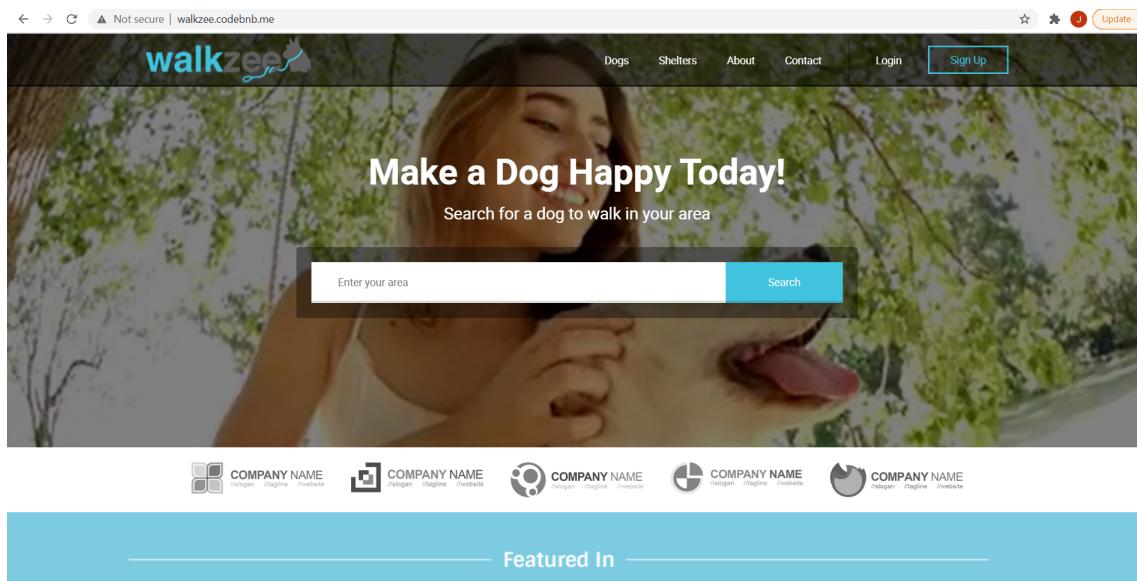
Njuškalo: – najveći online oglasnik u Hrvatskoj

Čuvalica: – Nacionalni portal za brigu o obitelji – briga za djecu, starije, ljubimce

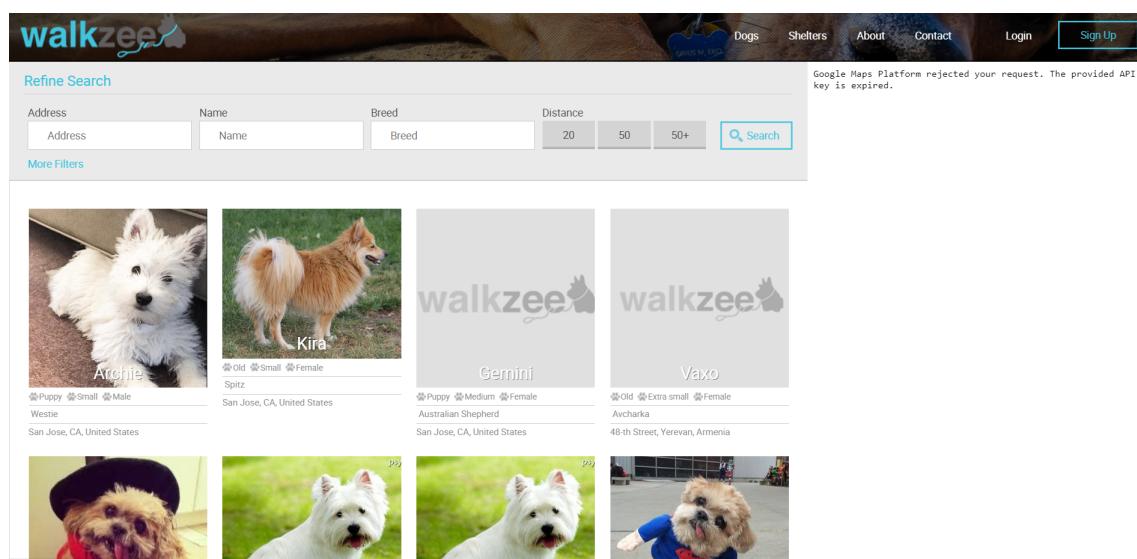
Također, postoji web-stranica obrta **PETS STEP** koji nudi uslugu čuvanja i šetnje pasa. Na stranici imaju kontakt i cjenik.

Kao što možemo vidjeti, u Hrvatskoj ne postoji usluga slična našoj, no u svijetu ih možemo pronaći. Najsličniji primjer bi bila web platforma **Walkzee** – “*1st free online platform connecting shelter dogs in need of a walk to dog lovers looking for a walking buddy!*” Walkzee su stvorili Cristina i Charlie Saunders 2015. te je ideja

identična kao ideja iza našeg projekta. Nažalost, njihova stranica nije zaživjela te nema mnogo udruga niti pasa na stranici. Naime, ima samo 10 stranica te 14 pasa. Osim toga, stranica izgleda nedovršeno pa ne znamo je li ikada uopće funkcionala. Nadamo se da njihov neuspjeh leži u lošoj egzekuciji, a ne u lošoj ideji odnosno indiferentnosti javnosti prema napuštenim životinjama.



Slika 2.1: Walkzee – naslovna stranica



Slika 2.2: Walkzee – pregled pasa

2.3 Opseg projektnog zadatka

Glavni zadatak aplikacije je povezati udruge za životinje s građanima koji imaju želju i vrijeme za šetanje pasa, te time povećati izglede udomljavanja pasa i psihološkog efekta dobrobiti socijalizacije za psa i za čovjeka.

Aplikaciju će koristiti registrirane udruge za životinje, registrirani građani i javni posjetitelji koji nisu registrirani i imaju mogućnost pristupa naslovnoj stranici aplikacije i detaljima profila udruge. Udruge i građani se mogu registrirati, pri čemu građani upisuju sljedeće podatke:

- ime
- prezime
- korisničko ime
- adresa e-pošte

a udruge još dodatno:

- naziv udruge
- OIB udruge

Javni posjetitelj može doći u aplikaciju i pregledati sve udruge na naslovnoj stranici, zatim otići na detalje profila pojedine udruge, a osim toga ima i uvid u rang listu registriranih šetača. Rang lista prikazuje poredak šetača s obzirom na broj šetnji, broj pasa, te duljinu šetnje koju su odradili u prethodnih mjesec dana. Na profilu pojedine udruge, posjetitelj može dobiti uvid u profile pasa, statistike o šetanjima svih pasa, lokaciju, te mogućnost prijave za šetanje pasa (ako se registrira). Statistika o šetnjama pruža informacije koji psi su češće bili u šetnji od ostalih, te time koji psi imaju veću potrebu za šetnjom. Ukoliko posjetitelj odluči pripomoći udruzi i priključiti se šetnji pasa, ima opciju registracije. Registracijom posjetitelj postaje registrirani građanin.

Registrirani građanin ima opciju prijave u vlastiti profil i pregleda vlastitih rasporeda šetanja, vlastitih statistika šetnji, zajedno s mogućnošću označavanja statistike šetanja kao javnih; kako bi podaci građana dospjeli na rang listu na javnoj

stranici. Svi registrirani korisnici imaju mogućnost mijenjanja podataka u svom profilu. Građanin može odabrati psa/e, odabrati željeni termin šetnje i prijaviti se za šetača. Termin šetnje se odabire u obliku datuma i vremena. Nakon uspješnog „rezerviranja“ psa za šetnju, termin za odabranog psa vidljiv je na kalendaru registriranim građanima. Građani imaju opciju skinuti raspored za odabrani dan, tjedan ili mjesec u PDF obliku.

Svaka registrirana udruga može kreirati vlastiti profil koji će se prikazivati na javnoj stranici. Stranica udruge će sadržavati neke bitne detalje vezane uz samu udrugu poput:

- ime udruge
- voditelj udruge
- lokacija udruge
- kontakt: e-mail adresa
- OIB udruge
- IBAN udruge (za moguće donacije)

Također, svaka udruga održava listu vlastitih pasa koji su raspoloživi za šetnju. Neke od bitnih informacija o pojedinom psu uključuju:

- ime psa
- vrsta psa (ako je poznata)
- slika psa
- opis psa (osobnost, izgled)
- dob psa
- raspored odnosno raspoloživost psa za određeni vremenski period (datum i vrijeme)
- za kakve šetnje je pas predodređen (skupne ili individualne šetnje).

Svaka udruga ima opciju mijenjati svoj profil. To može uključivati mijenjanje vlastitih podataka (vezanih uz samu udrugu) te uređivanje liste i profila pasa.

3. Specifikacija programske potpore

3.1 Funkcionalni zahtjevi

Dionici:

1. Voditelji udruga
2. Šetači pasa
3. Zaposlenici i volonteri u udrugama
4. Razvojni tim
5. Javni posjetitelji

Aktori i njihovi funkcionalni zahtjevi:

1. Javni posjetitelj (inicijator) može:
 - (a) pregledati listu udruga na naslovnoj stranici
 - (b) odabrati udrugu te pregledati:
 - i. detalje profila udruge:
 - A. ime udruge
 - B. voditelj udruge
 - C. kontakt: email adresa i broj mobitela
 - D. lokacija
 - E. OIB udruge
 - F. IBAN udruge - u slučaju da netko želi napraviti donaciju
 - ii. listu pasa iz te udruge koji su raspoloživi za šetnju
 - (c) odabrati profil psa iz liste pasa te pregledati detalje profila psa:
 - i. ime psa
 - ii. vrsta psa (ako je poznata)
 - iii. slika psa
 - iv. opis psa (osobnost, izgled)
 - v. dob psa

- vi. raspored odnosno raspoloživost psa za određeni vremenski period (datum i vrijeme)
 - vii. vrsta šetnje za koju je pas predodređen (skupna ili individualna)
 - (d) otvoriti statistiku svih pasa raspoloživih za šetnju i vidjeti koji pas se najmanje šteao, odnosno kojem psu je šetnja najpotrebnija
 - (e) otvoriti rang-listu svih registriranih šetača poredanu s obzirom na broj šetnji, broj pasa te duljinu šetnje koju su odradili u proteklih mjesec dana
 - (f) registrirati se u sustav kao građanin - za stvaranje korisničkog računa potrebni su mu:
 - i. ime i prezime
 - ii. e-mail adresa
 - iii. korisničko ime
 - iv. lozinka
 - (g) registrirati u sustav svoju udrugu - za stvaranje korisničkog računa potrebni su mu:
 - i. ime i prezime
 - ii. e-mail adresa
 - iii. korisničko ime
 - iv. lozinka
 - v. naziv udruge
 - vi. OIB udruge
2. Registrirani korisnik (inicijator) preuzima sve funkcionalnosti javnog posjetitelja (osim registracije) te može dodatno:
- (a) prijaviti se u sustav (s e-mailom i lozinkom)
 - (b) uz uvjet da je prijavljen u sustav:
 - i. pregledati vlastiti profil
 - ii. uređivati vlastiti profil
 - iii. obrisati vlastiti profil
3. Registrirani građanin (inicijator) preuzima sve funkcionalnosti registriranog korisnika te (uz uvjet da je prijavljen) može još dodatno:
- (a) odabrati psa te na njegovom profilu prijaviti se za šetnju

- (b) pregledati vlastiti raspored šetnji te skinuti (eng. download) raspored za odabrani dan, tjedan ili mjesec, u PDF obliku
 - (c) pregledati vlastitu statistiku šetanja
 - (d) označiti vlastite statistike šetanja kao javne kako bi podaci građana dospjeli na rang listu na javnoj stranici
4. Registrirana udruga (inicijator) preuzima sve funkcionalnosti registriranog korisnika te (uz uvjet da je prijavljena) može dodatno:
- (a) dodavati i brisati pse iz liste raspoloživih pasa te udruge
 - (b) uređivati profile pasa koji su iz te udruge
5. Baza podataka (sudionik):
- (a) pohranjuje sve podatke o korisnicima i udrugama
 - (b) pohranjuje sve podatke o šetnjama (obavljenim i rezerviranim) te o psima iz udruga

3.1.1 Obrasci uporabe

UC1 - Pregled profila udruga i pasa iz te udruge

- **Glavni sudionik:** Javni posjetitelj
- **Cilj:** Otvoriti profil pojedine udruge ili psa iz te udruge
- **Sudionici:** Baza podataka
- **Preduvjet:** Pristup aplikaciji
- **Opis osnovnog tijeka:**
 1. Korisnik sa naslovne strane odabire udrugu koju želi proučiti
 2. Iz liste pasa te udruge korisnik odabire psa koji ga zanima
 3. Korisnik može proučavati podatke, raspored i statistiku šetanja željenog psa

UC2 - Pregled liste profila svih pasa

- **Glavni sudionik:** Javni posjetitelj
- **Cilj:** Otvoriti profil psa iz bilo koje udruge
- **Sudionici:** Baza podataka
- **Preduvjet:** Pristup aplikaciji
- **Opis osnovnog tijeka:**
 1. Korisnik iz izborne trake odabire "Lista svih pasa"
 2. Iz liste pasa korisnik odabire psa koji ga zanima
 3. Korisnik može proučavati podatke, raspored i statistiku šetanja željenog psa

UC3 - Pregled statistike svih pasa

- **Glavni sudionik:** Javni posjetitelj
- **Cilj:** Otvoriti statistiku šetanja svih pasa
- **Sudionici:** Baza podataka
- **Preduvjet:** Pristup aplikaciji
- **Opis osnovnog tijeka:**
 1. Korisnik iz izborne trake odabire "Lista svih pasa"
 2. Korisnik odabire opciju "Statistika svih pasa"
 3. Korisnik može proučiti statistiku šetanja svih pasa

UC4 - Pregled rang-liste svih šetača

- **Glavni sudionik:** Javni posjetitelj
- **Cilj:** Dobiti uvid u rang-listu svih šetača
- **Sudionici:** Baza podataka
- **Preduvjet:** Pristup aplikaciji
- **Opis osnovnog tijeka:**
 1. Korisnik iz izborne trake odabire "Rang-lista šetača"
 2. Aplikacija prikazuje poredak šetača s obzirom na broj šetnji, broj pasa te duljinu šetnji koje su odradili u prethodnih mjesec dana
 3. Korisnik može proučiti rang-listu svih šetača
- **Opis mogućih odstupanja:**
 - 2.a U slučaju da nijedan šetač nije označio statistiku svoje šetnje kao javnu, rang lista će biti prazna

UC5 - Registracija korisnika (građanina ili udruge) u sustav

- **Glavni sudionik:** Javni posjetitelj
- **Cilj:** Stvoriti korisnički račun za pristup sustavu
- **Sudionici:** Baza podataka
- **Preduvjet:** Pristup aplikaciji
- **Opis osnovnog tijeka:**
 1. Javni posjetitelj odabire opciju (gumb) za registraciju
 2. Javni posjetitelj bira "Registriraj se kao građanin" ili "Registriraj se kao udruga"
 3. Javni posjetitelj unosi potrebne korisničke podatke (ime, prezime, korisničko ime, email adresa i lozinka za građanina te dodatno ime udruge i OIB udruge za udrugu)
 4. Javni posjetitelj odabire opciju "Stvori korisnički račun"
 5. Korisnik prima obavijest o uspješnoj registraciji
- **Opis mogućih odstupanja:**
 - 2.a Unos podataka u neispravnom formatu
 1. Sustav obavještava korisnika o neispravnim podatcima i vraća ga na stranicu za registraciju
 2. Korisnik mijenja potrebne podatke te završava unos ili odustaje od registracije
 - 2.b Odabrana email adresa i/ili korisničko ime su već zauzeti

1. Sustav obaveštava korisnika o zauzetom korisničkom imenu/email adresi i vraća ga na stranicu za registraciju
2. Korisnik mijenja potrebne podatke te završava unos ili odustaje od registracije

UC6 - Prijava korisnika (građanina ili udruge) u sustav

- **Glavni sudionik:** Registrirani korisnik (građanin/udruga)
- **Cilj:** Dobiti pristup odgovarajućem korisničkom sučelju
- **Sudionici:** Baza podataka
- **Preduvjet:** Registracija građanina ili udruge u sustav
- **Opis osnovnog tijeka:**
 1. Unos korisničkog imena i lozinke
 2. Potvrda o ispravnosti unesenih podataka
 3. Pristup odgovarajućim korisničkim funkcijama (ovisi prijavljuje li se građanin ili udruga)
- **Opis mogućih odstupanja:**
 - 2.a Neispravno korisničko ime/lozinka
 1. Sustav obaveštava korisnika o neuspjelom upisu i vraća ga na stranicu za prijavu

UC7 - Pregled osobnih podataka korisnika (udruge ili građanina)

- **Glavni sudionik:** Registrirani korisnik (udruga/građanin)
- **Cilj:** Pregledati osobne podatke/podatke udruge
- **Sudionici:** Baza podataka
- **Preduvjet:** Registrirani korisnik (udruga ili građanin) je prijavljen u sustav
- **Opis osnovnog tijeka:**
 1. Korisnik iz izborne trake odabire "Osobni podatci"
 2. Otvara se stranica sa korisnikovim osobnim podatcima

UC8 - Promjena osobnih podataka korisnika

- **Glavni sudionik:** Registrirani korisnik (udruga/građanin)
- **Cilj:** Promijeniti osobne podatke/podatke udruge
- **Sudionici:** Baza podataka
- **Preduvjet:** Registrirani korisnik (udruga ili građanin) je prijavljen u sustav
- **Opis osnovnog tijeka:**
 1. Korisnik iz izborne trake odabire "Osobni podatci"

2. Otvara se stranica sa korisnikovim osobnim podatcima
 3. Korisnik bira "Uredi profil"
 4. Korisnik mijenja podatke
 5. Korisnik bira opciju "Pohrani promjene"
 6. Baza se ažurira
- **Opis mogućih odstupanja:**
 - 4.a Korisnik promijeni svoje osobne podatke, ali ne odabere opciju "Pohrani promjene"
 1. Sustav upozorava korisnika da nije spremio podatke prije izlaska iz prozora
 2. Korisnik se vraća i pohranjuje promjene ili izlazi iz prozora ostavljajući podatke bez promjene
 - 4.b Korisnik promijeni svoje osobne podatke, ali u neispravan format
 1. Sustav upozorava korisnika da je podatak u neispravnom formatu i ne dopušta pohranu takvih podataka
 2. Korisnik mijenja podatak i pohranjuje promjene ili izlazi iz prozora ostavljajući podatke bez promjene
 - 6.b Korisnik promijeni svoje osobne podatke, ali je nova email adresa i/ili korisničko ime već zauzeto
 1. Sustav upozorava korisnika da je email adresa i/ili korisničko već zauzeto i da promjene ne mogu biti pohranjene
 2. Korisnik mijenja email adresu i/ili korisničko ime i pohranjuje promjene ili izlazi iz prozora ostavljajući podatke bez promjene

UC9 - Brisanje korisničkog računa (udruge ili građanina)

- **Glavni sudionik:** Registrirani korisnik (udruga/gradjanin)
- **Cilj:** Izbrisati vlastiti korisnički račun
- **Sudionici:** Baza podataka
- **Preduvjet:** Registrirani korisnik (udruga ili građanin) je prijavljen u sustav
- **Opis osnovnog tijeka:**
 1. Korisnik iz izborne trake odabire "Osobni podatci"
 2. Otvara se stranica sa korisnikovim osobnim podatcima
 3. Korisnik bira "Uredi profil"
 4. Korisnik bira "Izbriši korisniči račun"
 5. Sustav upozorava korisnika da je brisanje korisničkog računa trajno te provjerava je li siguran

6. Korisnik potvrđuje brisanje
 7. Korisnički račun se briše iz baze podataka
 8. Otvara se naslovna stranica
- **Opis mogućih odstupanja:**
 - 5.a Korisnik poništi brisanje računa
 1. Račun se ne briše te se baza ne ažurira
 2. Korisnik je ostavljen na stranici "Uredi profil"

UC10 - Dodavanje novog profila psa u listu prijavljene udruge

- **Glavni sudionik:** Registrirana udruga
- **Cilj:** Dodati profil nekog psa iz te (prijavljene) udruge
- **Sudionici:** Baza podataka
- **Preduvjet:** Registrirana udruga je prijavljena u sustav
- **Opis osnovnog tijeka:**
 1. Korisnik iz izborne trake odabire "Moji psi"
 2. Otvara se lista svih pasa iz te udruge
 3. Korisnik bira opciju "Dodaj novog psa"
 4. Otvara se stranica za upis podataka o novom psu
 5. Korisnik upiše podatke, optionalno dodaje i sliku
 6. Korisnik bira opciju "Dodaj psa"
 7. Baza se ažurira
- **Opis mogućih odstupanja:**
 - 4.a Korisnik upiše podatke o psu, ali ne odabere opciju "Dodaj psa"
 1. Sustav upozorava korisnika da nije spremio nove podatke prije izlaska iz prozora
 2. Korisnik se vraća i pohranjuje podatke ili izlazi iz prozora bez dodavanja novog psa
 - 4.b Korisnik upiše podatke o psu, ali novi podaci su u neispravanom formatu
 1. Sustav upozorava korisnika da je podatak (ili više njih) u neispravnom formatu i ne dopušta pohranu takvih podataka
 2. Korisnik mijenja podatak/e i pohranjuje ih ili izlazi iz prozora bez dodavanja novog psa

UC11 - Uređivanje profila psa neke udruge

- **Glavni sudionik:** Registrirana udruga

- **Cilj:** Urediti profil nekog psa iz te (prijavljene) udruge
- **Sudionici:** Baza podataka
- **Preduvjet:** Registrirana udruga je prijavljena u sustav
- **Opis osnovnog tijeka:**
 1. Korisnik iz izborne trake odabire "Moji psi"
 2. Otvara se lista svih pasa iz te udruge
 3. Korisnik bira psa čiji profil želi urediti
 4. Korisnik mijenja podatke
 5. Korisnik bira opciju "Pohrani promjene"
 6. Baza se ažurira
- **Opis mogućih odstupanja:**
 - 4.a Korisnik promijeni podatke o psu, ali ne odabere opciju "Pohrani promjene"
 1. Sustav upozorava korisnika da nije spremio podatke prije izlaska iz prozora
 2. Korisnik se vraća i pohranjuje promjene ili izlazi iz prozora ostavljajući podatke bez promjene
 - 4.b Korisnik promijeni podatke o psu, ali novi podatci su u neispravanom formatu
 1. Sustav upozorava korisnika da je podatak (ili više njih) u neispravnom formatu i ne dopušta pohranu takvih podataka
 2. Korisnik mijenja podatak/e i pohranjuje promjene ili izlazi iz prozora ostavljajući podatke bez promjene

UC12 - Brisanje profila psa iz liste neke udruge

- **Glavni sudionik:** Registrirana udruga
- **Cilj:** Obrisati profil nekog psa iz liste pasa te (prijavljene) udruge
- **Sudionici:** Baza podataka
- **Preduvjet:** Registrirana udruga je prijavljena u sustav
- **Opis osnovnog tijeka:**
 1. Korisnik iz izborne trake odabire "Moji psi"
 2. Otvara se lista svih pasa iz te udruge
 3. Korisnik bira psa čiji profil želi obrisati
 4. Korisnik bira opciju "Obriši profil psa"
 5. Sustav šalje upit korisniku je li siguran
 6. Korisnik potvrđuje brisanje

7. Profil tog psa se briše iz baze podataka
 8. Otvara se stranica "Moji psi"
- **Opis mogućih odstupanja:**
 - 5.a Korisnik poništi brisanje profila psa
 1. Profil psa se ne briše te se baza ne ažurira
 2. Korisnik je ostavljen na stranici "Uredi profil psa"

UC13 - Rezervacija termina šetnje

- **Glavni sudionik:** Registrirani građanin (šetač)
- **Cilj:** Šetač rezervira psa i termin u kojem će obaviti šetnju tog psa
- **Sudionici:** Baza podataka
- **Preduvjet:** Registrirani građanin (šetač) je prijavljen u sustav
- **Opis osnovnog tijeka:**
 1. Šetač dolazi na stranicu za odabir profila psa preko:
 - (a) profila neke udruge (UC1)
 - (b) liste svih pasa (UC2)
 2. Šetač odabire psa kojeg želi šetati
 3. Šetač dobiva uvid u slobodne termine odabranog psa te bira neki
 4. Sustav šalje potvrdu rezervacije termina šetaču
 5. Nakon uspješne rezervacije, termin za odabranog psa vidljiv je na kalendaru šetaču
- **Opis mogućih odstupanja:**
 - 4.a Odabrani pas nema slobodnih termina za šetnje
 1. Šetač izlazi iz profila psa i može tražiti novog psa za šetnju

UC14 - Pregled vlastitih statistika šetnji

- **Glavni sudionik:** Registrirani građanin (šetač)
- **Cilj:** Pregledati svoju statistiku šetnji u određenom razdoblju
- **Sudionici:** Baza podataka
- **Preduvjet:** Registrirani građanin (šetač) je prijavljen u sustav
- **Opis osnovnog tijeka:**
 1. Šetač bira opciju "Moj profil" na izbornoj traci odnosno odlazi na stranicu svog profila
 2. Šetač odabire opciju "Moja statistika" na stranici profila
 3. Šetač bira razdoblje u kojem želi pregledati statistiku
 4. Statistika je prikazana te ju šetač može proučiti

UC15 - Označavanje statistike šetnji kao javne

- **Glavni sudionik:** Registrirani građanin (šetač)
- **Cilj:** Učiniti svoju statistiku šetnji javno dostupnom kako bi se mogla prikazati na rang-listi šetača
- **Sudionici:** Baza podataka
- **Preduvjet:** Registrirani građanin (šetač) je prijavljen u sustav
- **Opis osnovnog tijeka:**
 1. Šetač bira opciju "Moj profil" na izbornoj traci odnosno odlazi na stranicu svog profila
 2. Šetač odabire opciju "Moja statistika" na stranici profila te se prikazuje stranica njegove statistike
 3. Šetač odabire opciju "Označi svoju statistiku javnom"
 4. Njegova statistika je prikazana na javnoj rang-listi šetača

UC16 - Pregled i skidanje (eng.download) kalendara registriranog građanina

- **Glavni sudionik:** Registrirani građanin (šetač)
- **Cilj:** Pregledati svoj kalendar odnosno raspored rezerviranih šetnji i skinuti raspored za dan, mjesec ili godinu u PDF-u
- **Sudionici:** Baza podataka
- **Preduvjet:** Registrirani građanin (šetač) je prijavljen u sustav
- **Opis osnovnog tijeka:**
 1. Šetač bira opciju "Moj profil" na izbornoj traci odnosno odlazi na stranicu svog profila
 2. Šetač odabire opciju "Moj kalendar" na stranici profila
 3. Šetač može birati želi li vidjeti raspored za dan, mjesec ili godinu
 4. Šetač može skinuti odabrani raspored u PDF-u

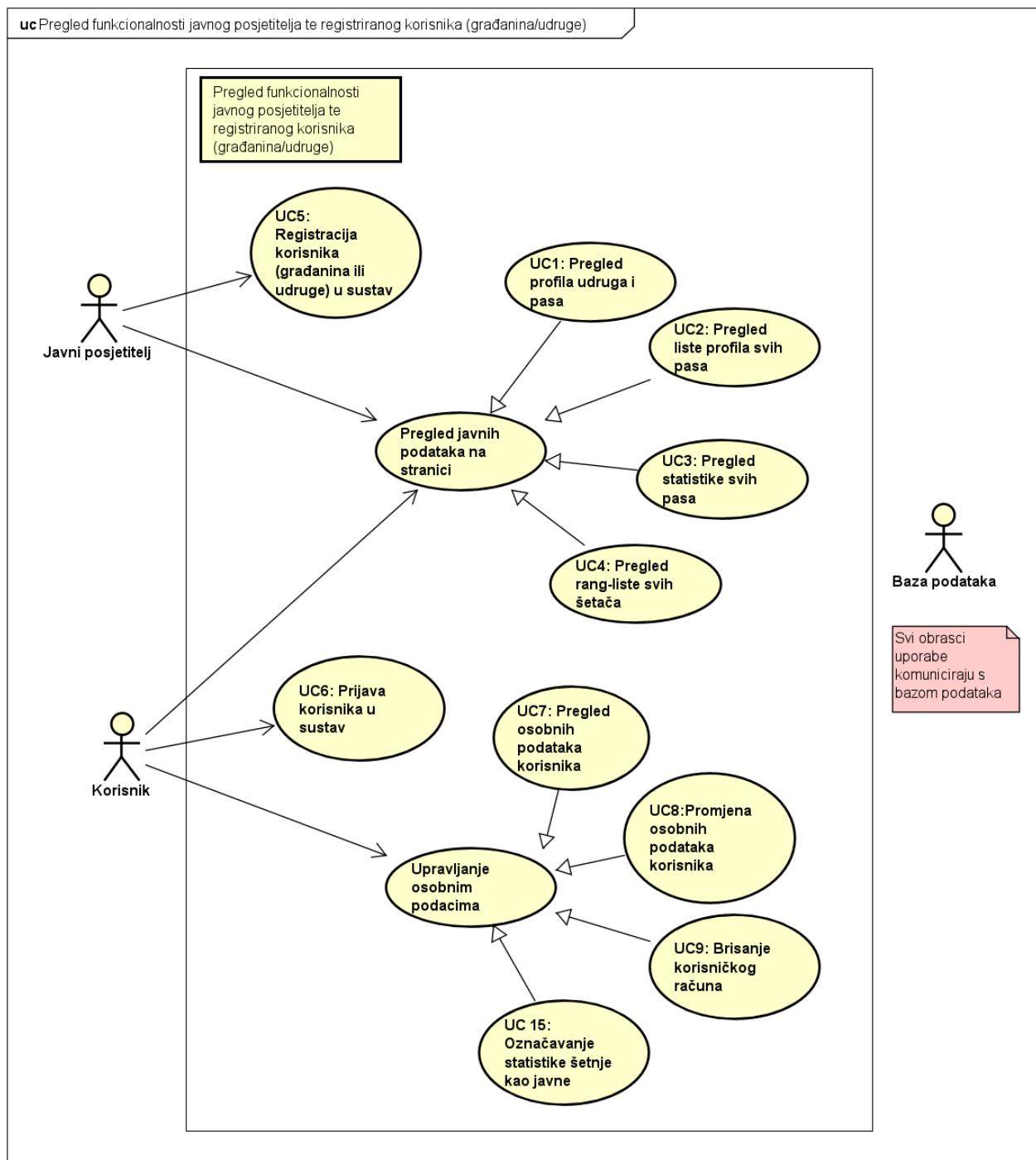
UC17 - Pregled korisnika

- **Glavni sudionik:** Administrator
- **Cilj:** Pregledati registrirane korisnike
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je registriran i dodijeljena su mu prava administratora
- **Opis osnovnog tijeka:**
 1. Administrator bira opciju pregledavanja korisnika
 2. Prikaže se lista svih ispravno registriranih korisnika s osobnim podacima

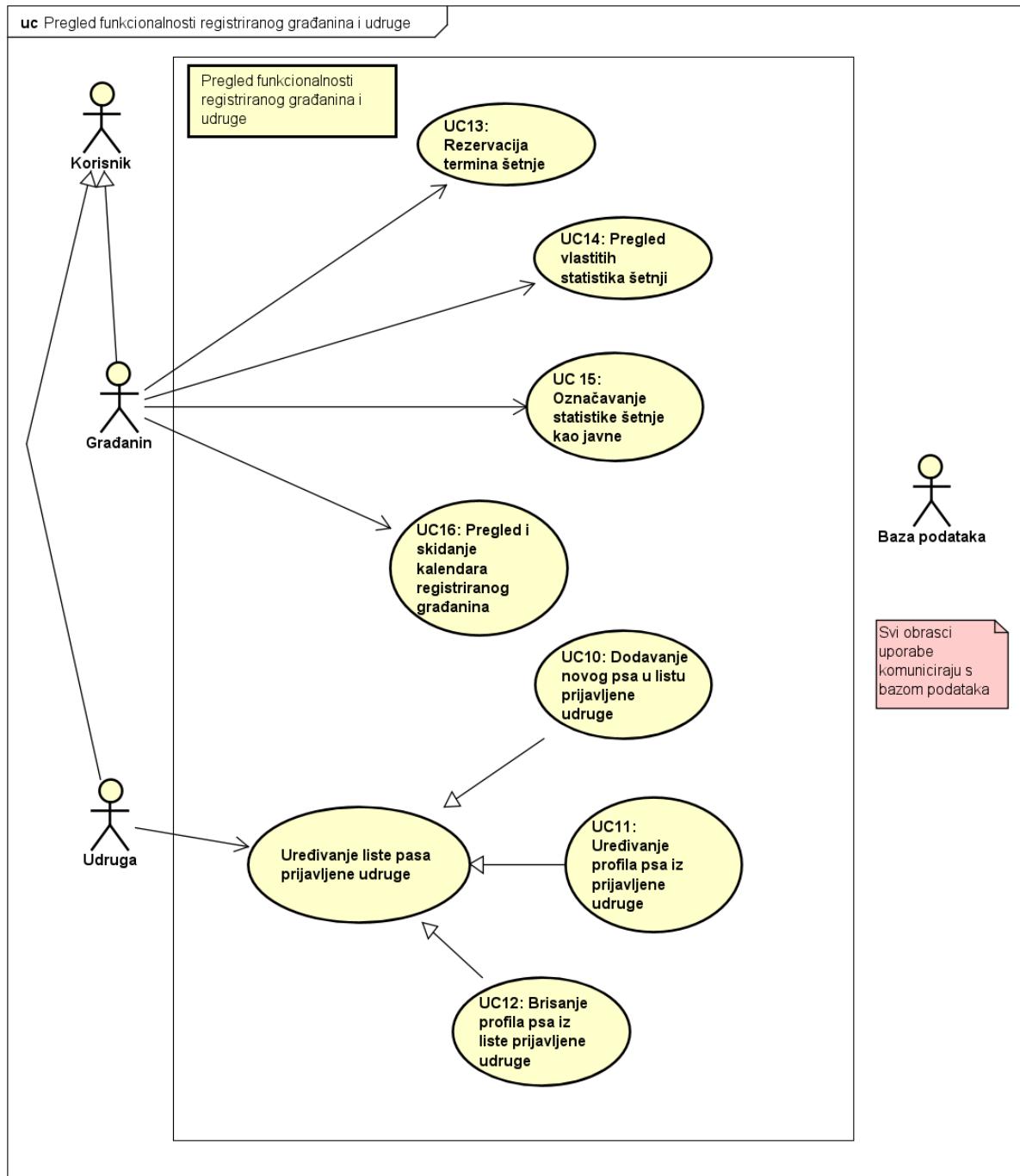
UC18 - Brisanje korisnika korisnika

- **Glavni sudionik:** Administrator
- **Cilj:** Obrisati korisnika
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je registriran i dodijeljena su mu prava administratora
- **Opis osnovnog tijeka:**
 1. Administrator bira opciju uklanjanja korisnika
 2. Administrator pronalazi željenog korisnika
 3. Administrator uklanja željenog korisnika i njegove podatke iz baze podataka

Dijagrami obrazaca uporabe



Slika 3.1: Dijagram obrasca uporabe: Prikaz funkcionalnosti javnog posjetitelja te registriranog korisnika

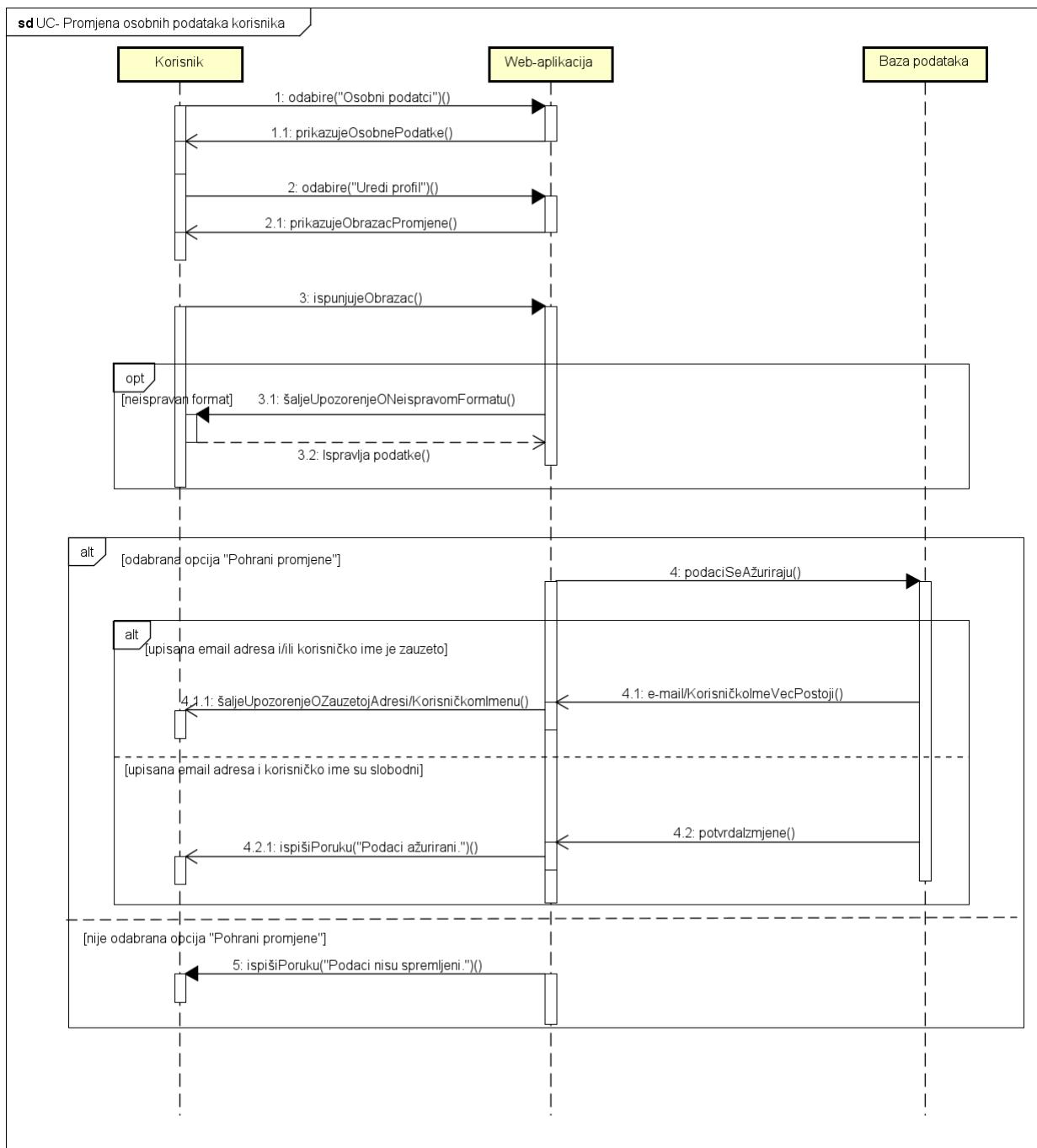


Slika 3.2: Dijagram obrasca uporabe: Prikaz funkcionalnosti registriranog građanina i udruge

3.1.2 Sekvencijski dijagrami

Obrazac uporabe UC8: Promjena osobnih podataka korisnika

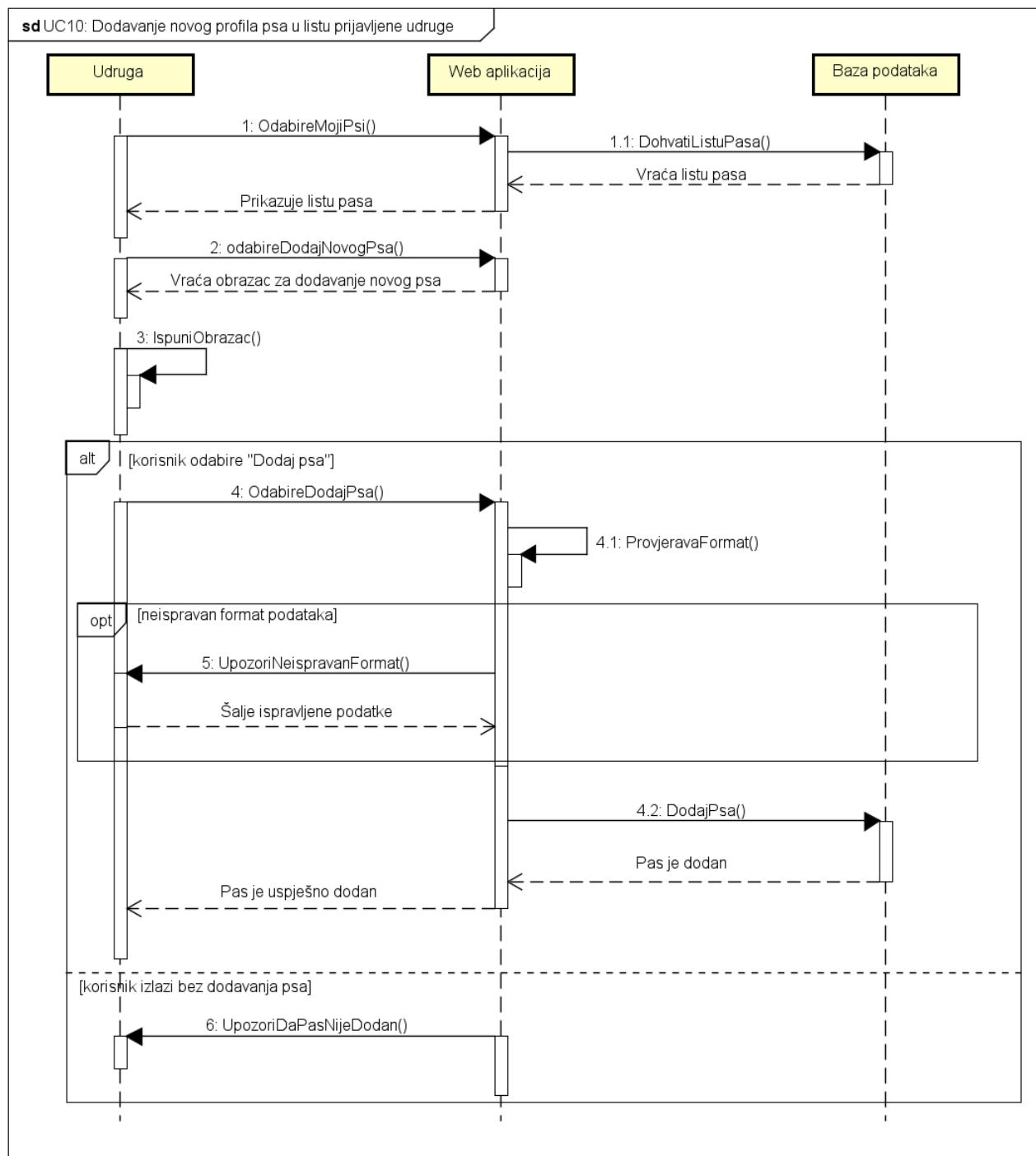
Korisnik šalje zahtjev za prikaz osobnih podataka. Poslužitelj dohvaća osobne podatke te ih prikazuje. Korisnik zatraži uređivanje osobnih podataka te mu aplikacija ponudi obrazac za promjenu podataka. Korisnik ispunjava podatke te bira "Spremi podatke". Ako su podaci u nespravnom formatu, aplikacija ga upozorava na to. Ako korisnik uredi podatke u ispravan format, podaci se šalju bazi. U slučaju da je nova email adresa već zauzeta, baza šalje upozorenje aplikaciji, a aplikacija korisniku. Ako su podaci ispravni, aplikacija šalje korisniku poruku da su podaci uspješno izmijenjeni. U slučaju da korisnik izađe iz obrasca za uređivanje osobnih podataka, aplikacija korisniku šalje upozorenje da podaci nisu spremljeni.



Slika 3.3: Sekvencijski dijagram: UC8 - Promjena osobnih podataka korisnika

Obrazac uporabe UC10: Dodavanje novog profila psa u listu prijavljene udruge

Korisnik (koji je prijavljena udruga) na izbornoj traci odabire opciju "Moji psi". Aplikacija od baze dohvaća listu pasa te ju prikazuje korisniku. Korisnik odabire opciju "Dodaj novog psa" u tom prozoru te mu aplikacija vraća prazni obrazac. Korisnik ispuni obrazac te odabire opciju "Dodaj psa". U slučaju da je neki od podataka u neispravnom obliku, aplikacija upozorava korisnika da podaci nisu spremjeni jer su u neispravnom formatu. Korisnik tada ispravlja podatke i ponovno ih šalje. Jednom kada su podaci ispravni, baza vraća potvrdu o unosu podataka te aplikacija vraća korisniku poruku da je dodan novi pas. U slučaju da korisnik izlazi prije uspješnog dodavanja novog psa, aplikacija mu šalje upozorenje da novi pas nije dodan.



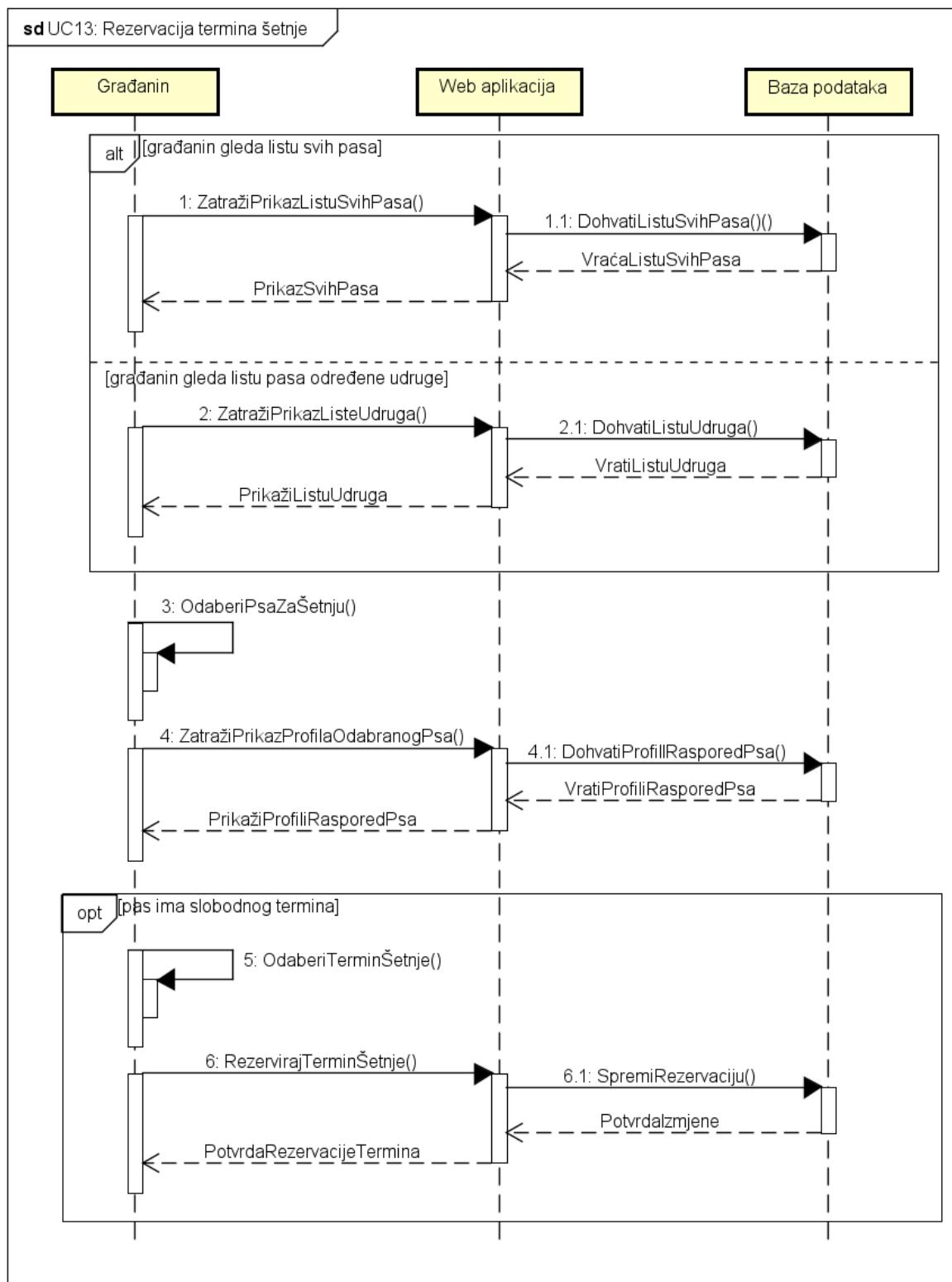
Slika 3.4: Sekvencijski dijagram: UC10 - Dodavanje novog profila psa u listu prijavljene udruge

Obrazac uporabe UC13: Rezervacija termina šetnje

Korisnik (koji je građanin) može pregledati pse na dva načina:

- preko liste svih pasa:
 - korisnik na izbornoj traci odabire "Psi"
 - aplikacija od baze dohvaća sve pse
 - korisniku je sada prikazana lista pasa iz svih udruga
- preko liste pasa određene udruge:
 - korisnik na izbornoj traci odabire "Udruge"
 - aplikacija od baze dohvaća sve udruge te prikazuje korisniku listu svih udruga korisnik odabire udrugu
 - aplikacija od baze dohvaća listu svih pasa iz odabrane udruge te ih prikazuje korisniku

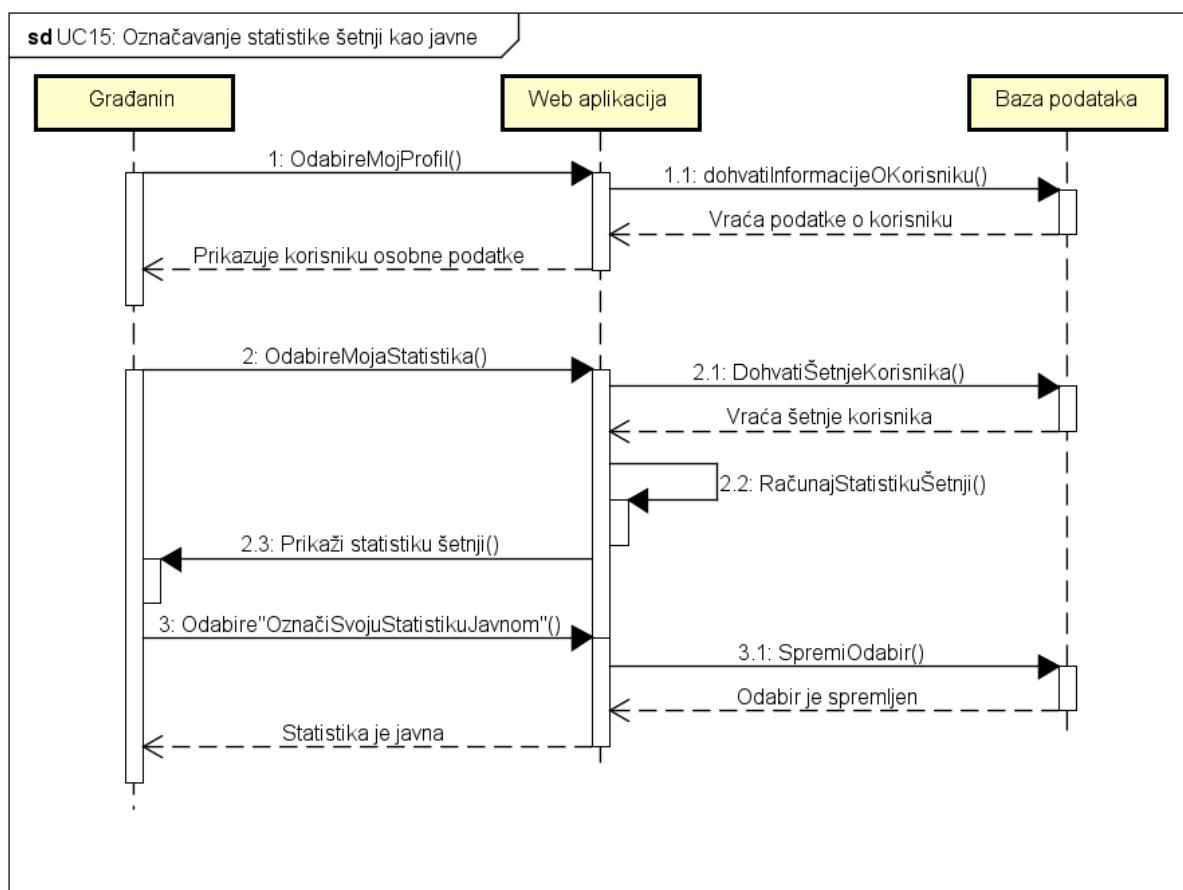
Nakon otvaranja liste pasa, korisnik bira psa kojeg bi volio šetati. Aplikacija od baze dohvaća profil (informacije) te raspored željenog psa te ih prikazuje korisniku. Korisnik bira željeni termin šetnje te aplikacija šalje rezervaciju bazi. Baza potvrđuje unos te aplikacija korisniku potvrđuje rezervaciju.



Slika 3.5: Sekvencijski dijagram: UC13 - Rezervacija termina šetnje

Obrazac uporabe UC15: Označavanje statistike šetnji kao javne

Korisnik (koji je građanin) odabire opciju "Moj profil" sa izborne trake. Web aplikacija dohvaća od baze podatke o korisniku te ih vraća. Aplikacija prikazuje korisniku njegove podatke. Korisnik odabire opciju "Moja statistika". Aplikacija od baze dohvaća podatke o obavljenim štenjama korisnika te ih prikazuje. U tom prozoru postoji opcija "Označi svoju statistiku javnom" te korisnik odabire tu opciju. Statistika postaje vidljiva u rang-listi svih šetača.



Slika 3.6: Sekvencijski dijagram: UC15 - Označavanje statistike šetnji kao javne

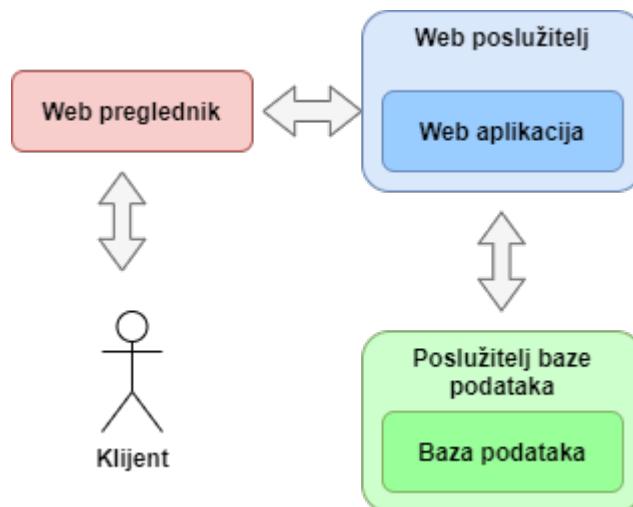
3.2 Ostali zahtjevi

- Sustav treba omogućiti rad više korisnika u stvarnom vremenu
- Korisničko sučelje i sustav moraju podržavati hrvatsku abecedu (dijakritičke znakove) pri unosu i prikazu tekstualnog sadržaja
- Izvršavanje dijela programa u kojem se pristupa bazi podataka ne smije trajati duže od nekoliko sekundi
- Sustav treba biti implementiran kao web aplikacija koristeći objektno-orientirane jezike
- Neispravno korištenje korisničkog sučelja ne smije narušiti funkcionalnost i rad sustava
- Sustav treba biti jednostavan za korištenje, korisnici se moraju znati koristiti sučeljem bez opširnih uputa
- Nadogradnja sustava ne smije narušavati postojeće funkcionalnosti sustava
- Veza s bazom podataka mora biti kvalitetno zastićena, brza i otporna na vanjske greške
- Pristup sustavu mora biti omogućen iz javne mreže pomoću HTTPS.
- S obzirom da je aplikacija na hrvatskom, datum prikazuje na odgovarajući način: DD.MM.YYYY.

4. Arhitektura i dizajn sustava

S najviše razine apstrakcije, arhitekturu možemo podijeliti na tri podsustava:

- Web poslužitelj
- Web aplikacija
- Baza podataka



Slika 4.1: Apstraktna arhitektura sustava

Web preglednik (eng. web browser) je program koji korisniku omogućuje pre-gled web-stranica i multimedijalnih sadržaja vezanih uz njih. Preglednik omogućuje komunikaciju između klijenta i poslužitelja. Dakle, korisnik će putem preglednika slati zahtjeve poslužitelju te će preglednik znati prikazati sadržaj koji poslužitelj vraća.

Web poslužitelj ima zadatak da pohranjuje, obrađuje i dostavlja klijentima web stranice. Komunikacija između web poslužitelja i web klijenta (najčešće, web pretraživača) odvija se korištenjem HTTP i drugih sličnih protokola (HTTPS, HTTP/2 i različitih nadogradnjih tih protokola). U komunikaciji putem HTTP-a, web poslužitelj

tipično sluša nadolazeće zahtjeve klijenata na portu 80. Web stranice koje web poslužitelj isporučuje klijentu su HTML dokumenti, koji uključuju tekst, slike, stilove, skripte i drugi sadržaj.

Web aplikacija obrađuje korisnikove zahtjeve te ako je potrebno, pristupa bazi podataka i dohvaca/mijenja podatke. Nakon toga preko poslužitelja vraća korisniku odgovor u obliku HTML dokumenta koji se prikazuje u web pregledniku.

Baza podataka pohranjuje podatke na sustavan način. Računalni program korišten za upravljanje i ispitivanje baze podataka nazvan je sustav upravljanja bazom podataka (SUBP).

Arhitektura i stil našeg sustava se može identificirati kao **višeslojni stil arhitekture**. Naime, naša arhitektura je uglavnom definirana činjenicom da koristimo **Spring Boot**. Obrazac koji se koristi u okviru radnog okvira Spring Boot je jedan od suvremenih primjera organizacije višeslojne arhitekture, a karakteriziraju ga dva principa:

- inverzija upravljanja (engl. inversion of control)
- ubacivanje ovisnosti (engl. dependency injection)

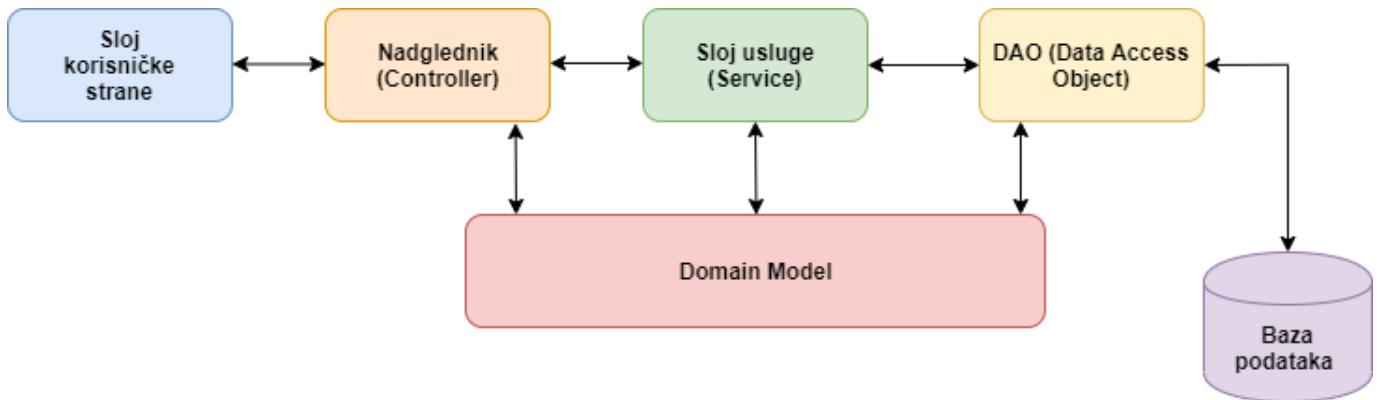
Inverzija upravljanja funkcioniра tako da korisnički napisani dijelovi (web) aplikacije dobiju upravljački tok i izvode se kada ih prozove neki generički radni okvir. To je inverzno u odnosu na tradicionalno programiranje u kojem korisnički kod poziva određene knjižnice kako bi se mogao izvršavati. U slučaju inverzije upravljanja, radni okvir za web aplikacije (u ovom slučaju, Spring Boot) poziva korisnički kod.

Ubacivanje ovisnosti je najčešći način kako u praksi funkcioniра inverzija upravljanja. Korisnički kod prima kao argumente metoda određene objekte, koji se u ovoj terminologiji zovu uslugama (engl. service), a koji su specificirani od strane radnog okvira kako bi sustav uspješno radio. Tako radni okvir, koji se u ovoj terminologiji naziva ubacivač ili injektor (engl. injector) ubacuje ovisnost o svojem određenom ugrađenom objektu u korisnički kod i definira sučelje (engl. interface) putem kojeg korisnički kod pristupa usluzi.

Naša arhitektura se tako sastoji od slojeva:

- sloj korisničke strane – implementiran u JavaScriptu, koristeći knjižnicu **React** koja omogućuje prikaz korisničkog sučelja

- sloj nadglednika (engl. controller) – povezuje korisničku stranu s poslužiteljskom stranom
- sloj usluge (engl. service) – obavlja svu poslovnu logiku i potrebne izračune
- sloj domene (engl. domain) – ima razrađeni model podataka domene
- sloj za pristup podatcima (engl. data access object, kraće: DAO) – koji omogućuje spremanje i dohvatanje podataka iz određene baze podataka te razmjenu tih podataka sa slojem domene
- sloj baze podataka – koji omogućuje stvarnu pohranu podataka u neku bazu,(u našem slučaju relacijsku bazu Postgre).



Slika 4.2: Spring Boot arhitektura

Želimo li, radi lakšeg razumijevanja, našu arhitekturu usporediti sa stilom MVC (Model-View-Controller), možemo uočiti da se neki slojevi djelomično poklapaju. Primjerice sloj modela kod MVC-a odgovarao bi sloju domene kod Spring Boota, sloj nadglednika kod MVC-a sloju nadglednika kod Spring Boota, a sloj pogleda bio bi sloj korisničke strane.

Kao što je navedeno, sloj korisničke strane (frontend) je implementiran u Java Scriptu koristeći knjižnicu React. **React** je besplatna biblioteka otvorenog koda za programski jezik JavaScript koja omogućava razvoj korisničkih sučelja SPA (eng. single-page application). SPA aplikacija je tip web-aplikacije koja u interakciji s korisnikom dinamički mijenja dijelove stranice, za razliku od tradicionalnih web-aplikacija koje učitavaju cijele nove stranice s poslužitelja. One se pokreću u browseru

i ne zahtijevaju ponovno učitavanje nakon korištenja. React omogućava sastavljanje korisničkog sučelja pomoću elemenata koji se nazivaju komponente. Njih je moguće iznova koristiti u aplikaciji proizvoljno veliki broj puta. Taj pristup pokazao se lakši za održavanje, proširivanje i višestruko korištenje.

4.1 Baza podataka

U sklopu svih zahtjeva vezanih uz razvoj aplikacije nalaze se različiti entiteti i opisi veza među njima. Nijedno se najčešće ne zadaje eksplisitno, već se od inženjera očekuje da ih prepozna, nekako zapiše i pravilno implementira u svoj sustav. Kako bi sve te entitete prikazali na jednom mjestu te kako bi mogli pohranjivati informacije o njima važno je razviti dobru bazu podataka. Baza podataka za koju smo se odlučili u ovome projektu je relacijska; njen glavni objekt je relacija. Jedna relacija predstavlja jedan entitet. Svaka relacija je prikazana u obliku tablice koja ima svoje ime i attribute koji ju opisuju. Tablice su međusobno povezane pri čemu treba pripaziti na vrste veza među njima. Baza podataka se sastoji od 6 glavnih entiteta:

- Walker
- Shelter
- Walk
- Reservation
- Dog
- Location

Značenje entiteta i opisi veza među njima su dani ispod.

4.1.1 Opis tablica

Walker

Tablica Walker predstavlja registriranog šetača/građana. Nakon što se korisnik aplikacije registrirao u sustavu, ili prijavio ukoliko već ima korisnički račun, on postaje registrirani šetač. Svaki registrirani šetač ima attribute poput imena, prezimena, adrese e-pošte i podataka za prijavu (korisničko ime i lozinka). Taj šetač ima opciju rezervacije termina i psa za šetnju. Iz tog razloga je povezan jedino s entitetom Reservation čija će uloga biti objašnjena ispod. Svaki šetač ima statistiku i raspored svojih šetnji. Ta dva entiteta nisu prepoznata kao zasebne tablice već kao potencijalni upiti nad rezervacijama koje je taj šetač napravio u sustavu. Važan

atribut svakog šetača je statVisibility koji označava hoće li se šetačeva statistika šetanja vidjeti na javno dostupnoj rang listi. Svaki šetač ima opciju postavljanja te statistike kao javne ili privatne ovisno o vlastitim preferencijama.

WALKER		
walkerId	INT	Jedinstveni identifikator šetača
firstName	VARCHAR	Ime šetača
lastName	VARCHAR	Prezime šetača
password	VARCHAR	Hash lozinka šetača za login u sustav
email	VARCHAR	Adresa e-pošte šetača
username	VARCHAR	Korisničko ime šetača
statsVisibility	BOOLEAN	(Ne)vidljivost šetačeve statistike šetnji javno

Reservation

Tablica Reservation predstavlja logičku poveznicu između tri entiteta. Naime, svaki pas kojeg je šetač odabrao za neki termin biva zapisan u zasebnu rezervaciju. Na taj način je omogućeno da se različiti psi nalaze u šetnji kod istog šetača. Svaka rezervacija sadržava 4 atributa, svoj identifikator koji nam u pogledu razumijevanja modela ne znači previše i ostala 3 koja zapravo definiraju jednu rezervaciju. Ti atributi su: walkId, walkerId i dogId. Zapisi (n-torce) u tablici rezervacija nam daju jasne informacije o tome koji šetači su šetali koje pse i u kojim terminima. Na taj način smo kreirali tablicu iz koje pomoći jednostavnih upita možemo doći do statistika šetnji, kako za pse tako i za šetače.

RESERVATION		
reservationId	INT	Jedinstveni identifikator rezervacije
walkerId	INT	Jedinstveni identifikator šetača
walkId	INT	Jedinstveni identifikator šetnje
dogId	INT	Jedinstveni identifikator psa

Walk

Tablica Walk predstavlja jedan ugovoren termin šetnje. Budući da je preko entiteta Reservation povezana s psima i šetačima, ne možemo direktno u n-torkama ove tablice vidjeti koji pas je sudjelovao u kojoj šetnji niti dobiti informaciju tko ga je šetao. Ukoliko pozajmimo identifikator šetnje i ne koristimo druge tablice možemo saznati dvije informacije, a to su: datum i vrijeme početka i trajanje same šetnje. Kako bi se nova n-torka unijela u ovu tablicu nužno je da šetač napravi rezervaciju čime se implicitno podrazumijeva da se njen termin i ugovorenog trajanja tu zapisuje.

WALK		
walkId	INT	Jedinstveni identifikator šetnje
dateTime	TIMESTAMP	Datum i vrijeme početka šetnje
duration	INT	Duljina šetnje u minutama

Dog

Tablica Dog predstavlja psa. Svaki pas ima udrugu kojoj pripada, s time da je jasno da više pasa može pripadati istoj udruzi. Osim oznake kojoj udruzi pas pripada, on ima atribut koji označavaju njegovu lokaciju, ime, sliku i kratak opis. Osim toga svaki pas ima naznačeno koji tip šetnje preferira (grupne ili individualne). Individualna šetnja pretpostavlja da će jedan registrirani šetač rezervirati samo jednog psa za neki termin te će time napraviti jednu rezervaciju. Grupna šetnja označava situaciju u kojoj šetač odabire više pasa za isti termin te time radi onoliko rezervacija koliko je pasa odabrao.

DOG		
dogId	INT	Jedinstveni identifikator psa
image	VARCHAR	URI slike psa
description	VARCHAR	Opis psa
typeOfWalk	VARCHAR	Tip šetnje za koju je pas predodređen (skupna/individualna)
name	VARCHAR	Ime psa
shelterId	INT	Jedinstveni identifikator udruge
locationId	INT	Jedinstveni identifikator lokacije

Shelter

Tablica Shelter predstavlja udrugu koja se u kontekstu aplikacije smatra drugom vrstom prijavljenog korisnika (osim već spomenutog šetača). Udruga ima svoj profil koji prikazuje slike i kratke opise pasa koji joj pripadaju. Osim informacije s kojim psima udruga raspolaže ona ima i svoje ime, OIB, lokaciju na kojoj se nalazi i naravno podatke za prijavu (korisničko ime i lozinku). Udruga raspolaže statistikom o šetnjama svojih pasa koja se također prikazuje na njenom profilu. Do te statistike se također dolazi putem pregleda svih rezervacija na kojima su njeni psi sudjelovali.

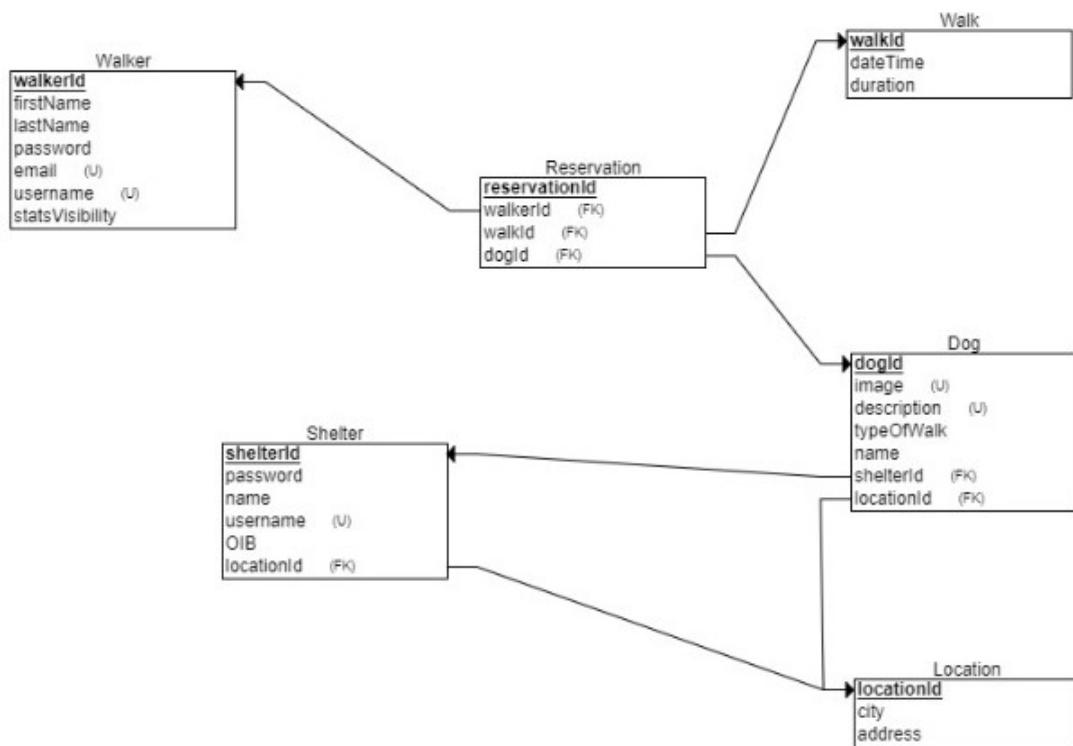
SHELTER		
shelterId	INT	Jedinstveni identifikator udruge
password	VARCHAR	Hash lozinka šetača za login u sustav
name	VARCHAR	Ime udruge
username	VARCHAR	Korisničko ime udruge
OIB	VARCHAR	OIB udruge
locationId	INT	Jedinstveni identifikator lokacije

Location

Tablica Location predstavlja geografsku lokaciju. Tu lokaciju mogu imati udruge i psi. Pretpostavili smo da se dvije udruge neće nalaziti na istoj lokaciji budući da ju definiraju grad i točna adresa. No, više pasa može živjeti na istoj lokaciji te time ograničenje između broja pasa i lokacije ne postoji. Informacije o lokaciji udruge se nalaze na profilu udruge, dok se lokacija nekog psa vidi u slučaju da ga neki šetač želi rezervirati.

LOCATION		
locationId	INT	Jedinstveni identifikator lokacije
city	VARCHAR	Grad
adress	VARCHAR	Adresa

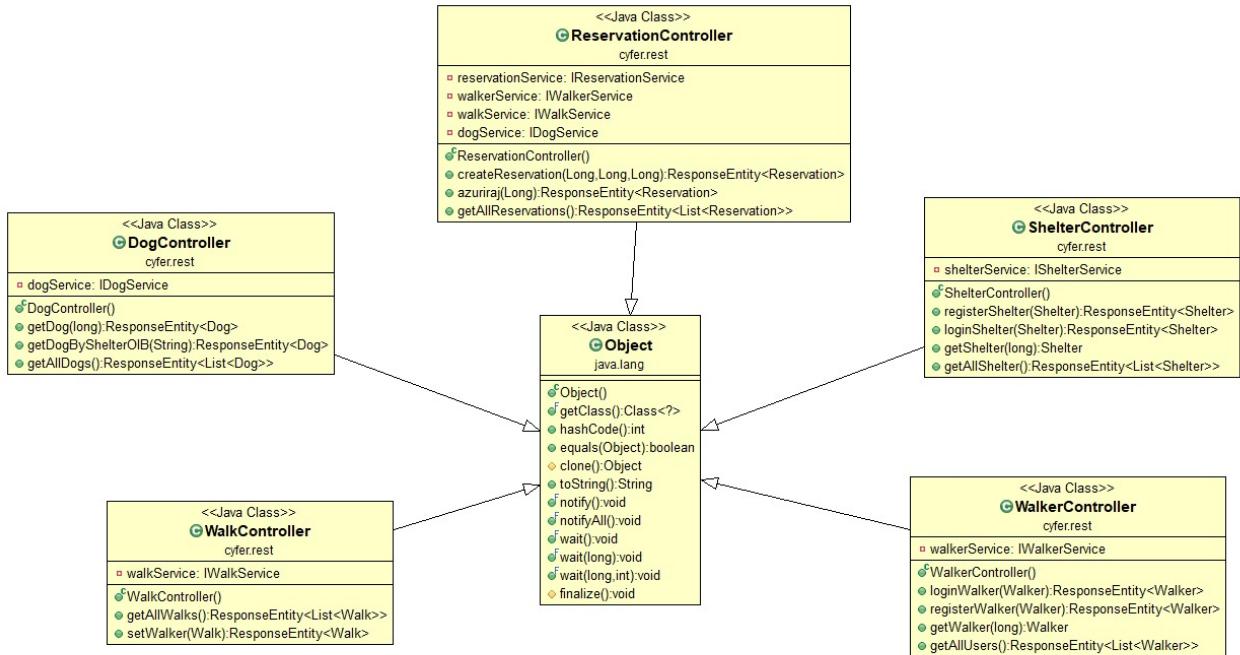
4.1.2 Dijagram baze podataka



Slika 4.3: Relacijski dijagram baze podataka

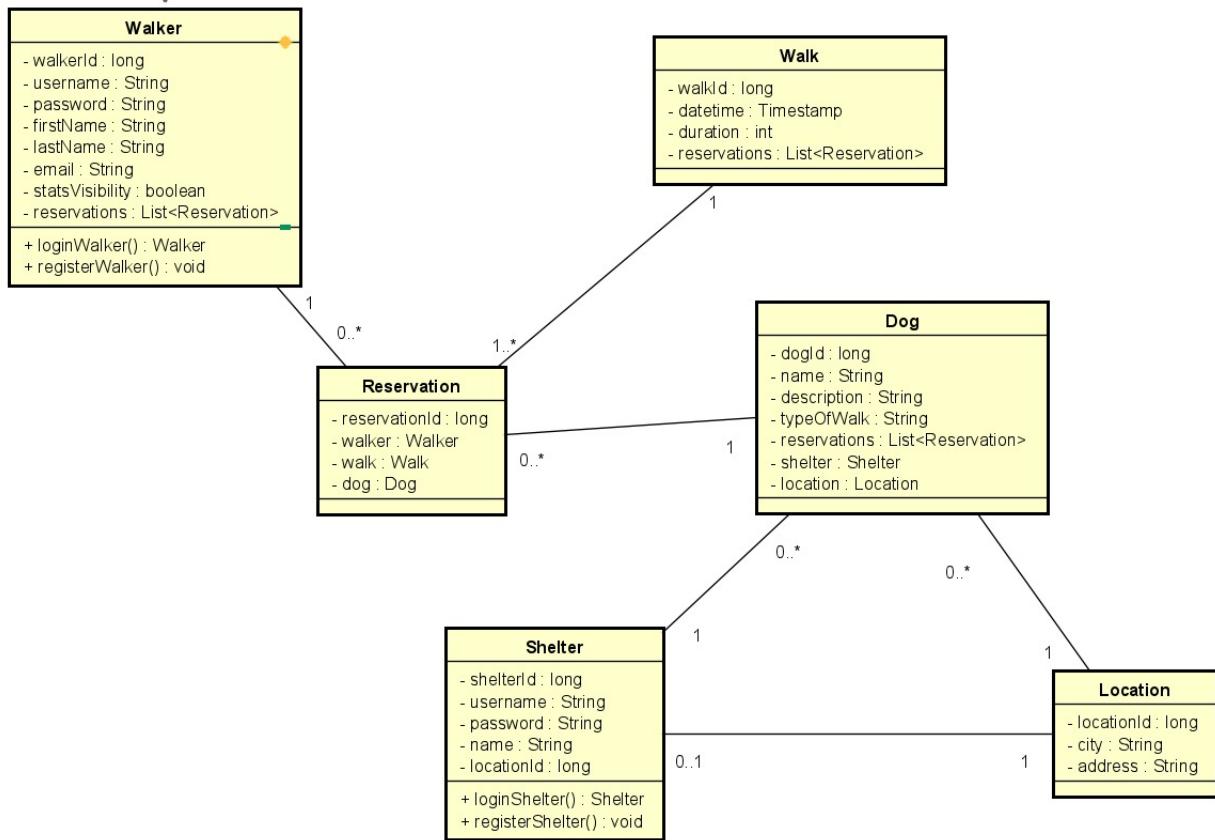
4.2 Dijagram razreda

Na slikama su prikazani dijagrami razreda koji predstavljaju backend dio MVC arhitekture ostvarene korištenjem Java Spring Boota.



Slika 4.4: Razredi controlleri

Razredi prikazani na slici 4.4 predstavljaju MVC Kontrolere čije metode mapiraju određene zahtjeve putem URI-a, te vraćaju JSON datoteke koje se potom obraduju na određeni način. Također šalju i HTML statusni kod. Kontroleri su implementirani za pet klasa tipa Model: Dog, Walk, Shelter, Walker i Reservation. Svaki razred ima konstruktor i metode koje mapiraju određene zahtjeve zadanim HTTP zahtjevom.

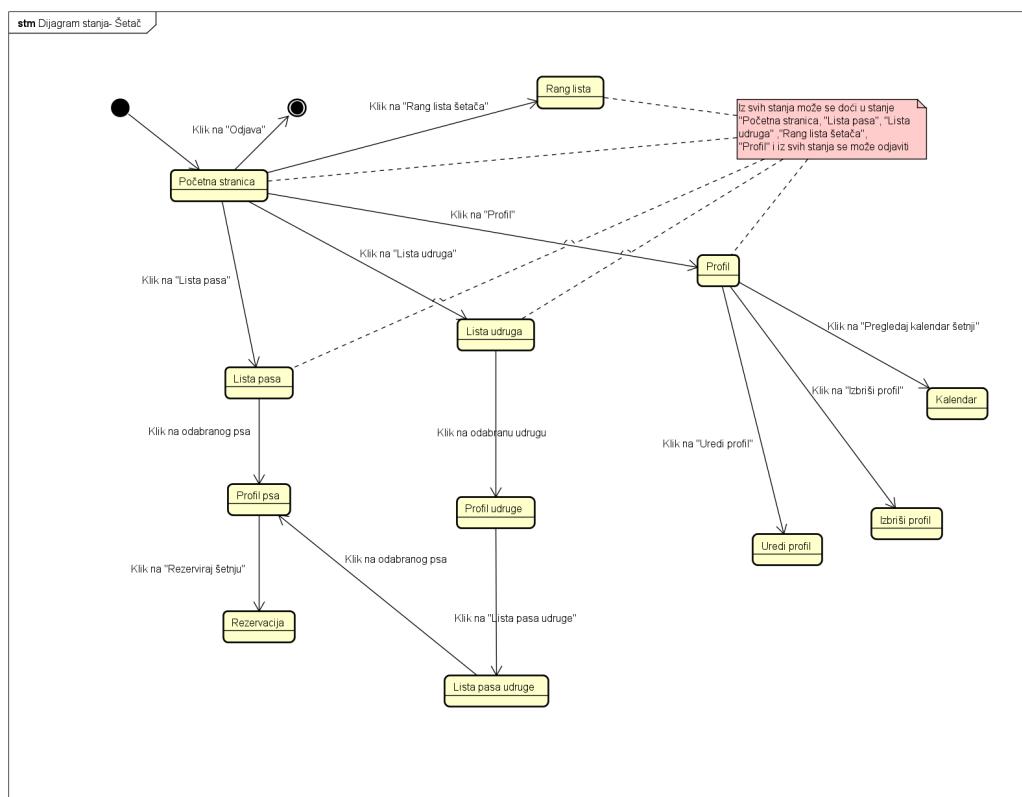


Slika 4.5: Razredi modeli

Razredi prikazani na slici 4.5 predstavljaju MVC Modele koji odgovaraju entitetima u bazi podataka. Svaki razred sadrži svoj konstruktor, privatne članske varijable koje su mu svojstvene i koje odgovaraju atributima u bazi, kao i sve gettere i setttere. Razred Dog predstavlja psa koji je pridruženoj određenoj udruzi (Shelter) i dostupan je za šetnje. Razred Walker predstavlja registriranog korisnika (šetača) koji može odabrati psa/pse za šetnju. Razred Shelter predstavlja registriranu udrugu koja ima svoj profil i pse dostupne za šetnju. Razred Walk predstavlja šetnju koja se dogodila s jednim šetačom i jednim ili više pasa. Razred Location predstavlja lokaciju (grad i adresu) na kojoj može biti smješten pas ili udruga. Kardinalnosti veza među razredima su prikazane na dijagramu.

4.3 Dijagram stanja

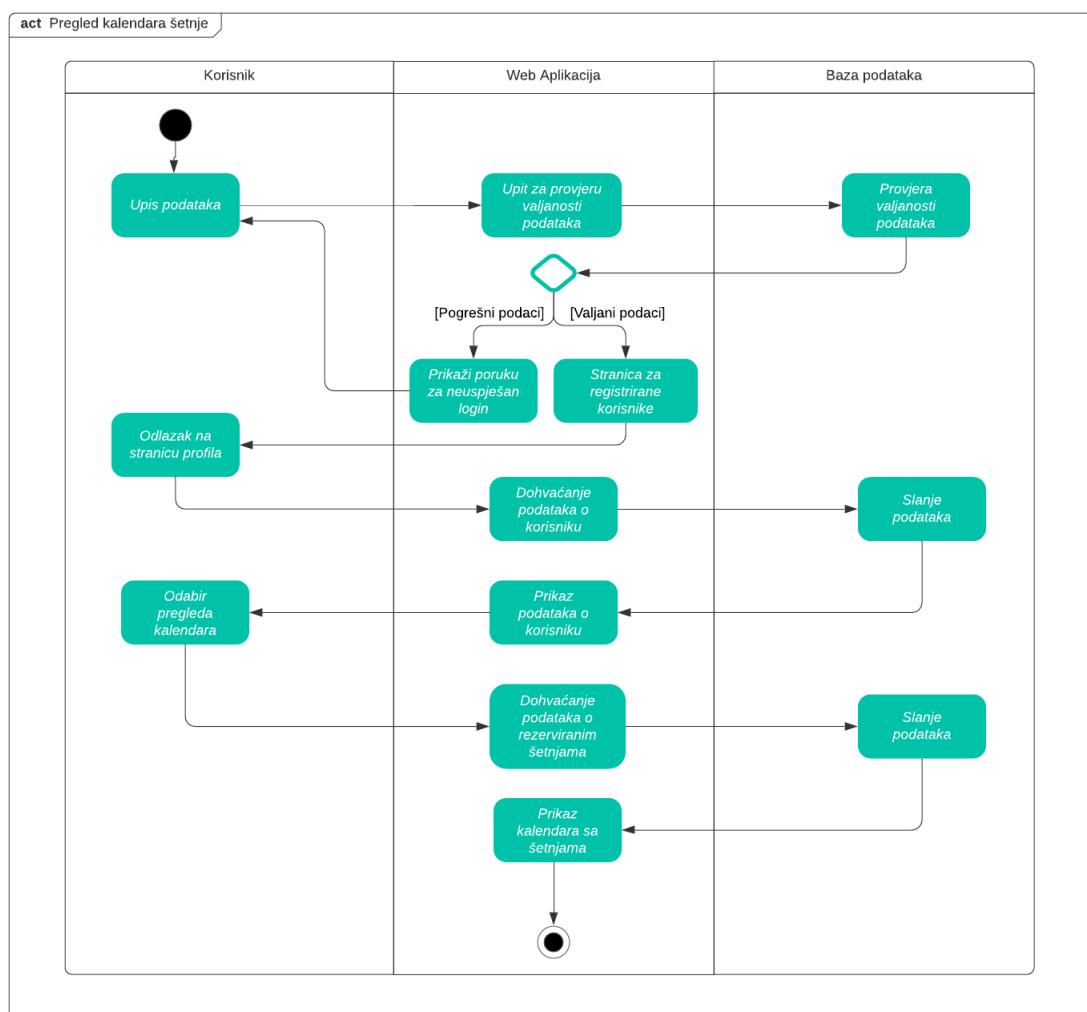
Dijagram stanja omogućava uočavanje različitih promjena unutar sustava, odnosno prikazivanje promjena koje se događaju u njegovim objektima i komponentama uslijed interakcije ili proteka vremena. Odnosno, prikazuje sva moguća stanja koja objekt klase može imati i događaje koje prouzrokuje kada se stanje mijenja. Dijagram na slici broj slike prikazuje stanja koja pripadaju registriranom korisniku. Korisniku se, nakon prijave ili registracije, prikazuje početna stranica i popis registriranih udruga. Za odabranu udrugu prikazuje se njen profil i lista pasa koji pripadaju toj udruzi. Odabirom željenog psa, prikazuje se njegov profil i mogućnost rezervacije tog psa za šetnju. Nakon rezervacije, korisniku se u prikazuje rezervacija o šetnji u njegovom kalendaru kojem može pristupiti klikom na "Profil". Na profilu korisnika prikazuju se njegovi podatci, kalendar i opcije za uređivanje ili brisanja profila. Korisnik također može pristupiti rang listi šetača u kojoj se vide statistike šetnji u posljednjih mjesec dana. Iz svih stanja uvijek se može doći do ustanje "Početna stranica", "Lista pasa", "Lista udruga", "Rang lista šetača", "Profil" i "Odjava" pritiskom na gumb u zaglavlju stranice.



Slika 4.6: Dijagram stanja - Šetač

4.4 Dijagram aktivnosti

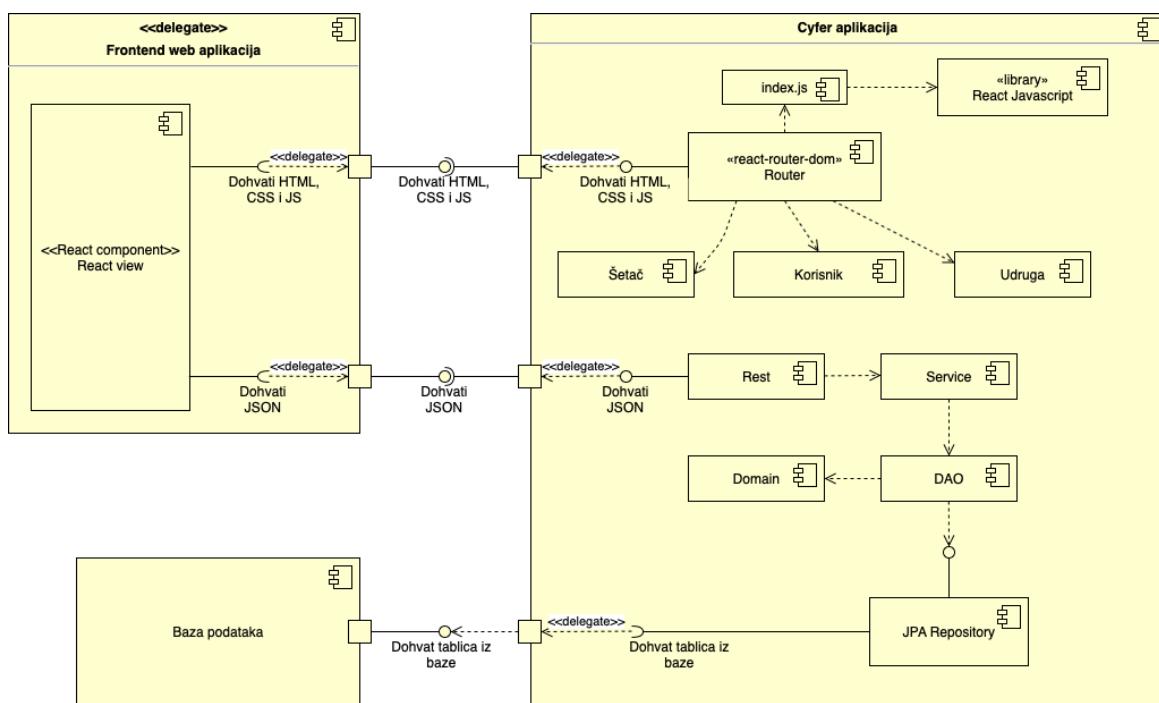
Korisnik upisuje podatke za prijavu. Poslužitelj dohvaća podatke o registriranim korisnicima iz baze podataka te provjerava postoji li korisnik u bazi. Ako ne postoji, vraća poruku o neuspješnoj prijavi. Ako postoji, preusmjerava korisnika na stranicu za registrirane korisnike. Korisnik s te stranice odabire stranicu profila. Poslužitelj dohvaća podatke o korisniku iz baze te ih prikazuje. Korisnik odabire opciju pregleda kalendara sa šetnjama. Poslužitelj iz baze dohvaća podatke o rezerviranim šetnjama korisnika i prikazuje kalendar sa svim rezerviranim šetnjama.



Slika 4.7: Dijagram aktivnosti: Pregled kalendarova šetnje

4.5 Dijagram komponenti

React-view komponenta s aplikacijom komunicira preko dva sučelja. Jednim sučeljem dohvaća potrebne HTML, CSS i JS datoteke s *frontend* dijela aplikacije. Komponenta Router je zadužena za određivanje koja će se datoteka poslužiti na sučelje. JavaScript datoteke gumbe, forme i slične gotove komponente dohvaćaju iz React biblioteke. Podacima sa *backend* dijela aplikacije, tj. REST API-u se pristupa sučeljem za dohvatanje JSON podataka. Za dohvatanje podataka iz baze podataka zadužen je JPA Repository, prema kojemu komponenta DAO ima ovisnost.



Slika 4.8: Dijagram komponenti

5. Implementacija i korisničko sučelje

5.1 Korištene tehnologije i alati

Komunikacija u timu realizirana je korištenjem aplikacije WhatsApp¹. Za izradu UML dijagrama korišten je alat Astah Professional², a kao sustav za upravljanje izvornim kodom Git³. Udaljeni repozitorij projekta je dostupan na web platformi GitLab⁴.

Kao razvojno okruženje korišten je IntelliJ IDEA⁵ - integrirano razvojno okruženje (IDE) tvrtke JetBrains⁶. Prvenstveno se koristi za razvoj računalnih programa, kao i za web-stranice, web-aplikacije, i mobilne aplikacije. Aplikacija je napisana koristeći radni okvir SpringBoot⁷ i jezik Java⁸ za izradu backenda te React⁹ i jezik JavaScript¹⁰ za izradu frontenda. React, također poznat kao React.js ili ReactJS, je biblioteka u JavaScriptu za izgradnju korisničkih sučelja. Održavana je od strane Facebooka¹¹. React se najčešće koristi kao osnova u razvoju web ili mobilnih aplikacija. Složene aplikacije u Reactu obično zahtijevaju korištenje dodatnih biblioteka za interakciju s API-jem. Spring¹² je široki skup biblioteka i alata koji olakšava razvoj aplikacija u Javi. Koristi anotacije, notacije i konvencije, koristeći oblikovne obrasce poput template-a i aspektno-orientiranog programiranja. Jezgra Springa je tzv. inversion of control container ili aplikacijski kontekst, koji se temelji na oblikovnom obrascu koji se zove dependency injection. Koristili smo H2¹³ bazu podataka na početku, a onda smo se prebacili prezistentnu Postgres¹⁴ bazu.

¹<https://www.whatsapp.com/>

²<https://astah.net/>

³<https://git-scm.com/>

⁴<https://gitlab.com/>

⁵<https://www.jetbrains.com/idea/>

⁶<https://www.jetbrains.com/company/>

⁷<https://spring.io/projects/spring-boot>

⁸<https://www.java.com/en/>

⁹<https://reactjs.org/>

¹⁰<https://www.javascript.com/>

¹¹<https://www.facebook.com/>

¹²<https://spring.io/projects/spring-framework>

¹³<https://www.h2database.com/html/main.html>

¹⁴<https://www.postgresql.org/>

5.2 Ispitivanje programskog rješenja

5.2.1 Ispitivanje komponenti

Testirali smo 8 ispitnih slučajeva unoseći ispravne i neispravne podatke te provjeravajući vraća li sustav odgovore kakve očekujemo. Od osam ispitnih primjera, četiri se odnose na šetače te četiri na udruge. Rezultati i opisi pojedinih testova su priloženi ispod. Kao što ćemo vidjeti u rezultatima, svi su ispitni prolazni jer smo pri upisu neispravnih podataka i očekivali odgovore različite od 200 OK.

1. ispitni primjer - registracija udruge sa zauzetim OIB-om

U ispitnom primjeru prvo smo stvorili i registrirali prvu udrugu sa zadanim OIB-om 22000000000. Dobili smo očekivani odgovor sa statusom 200 OK. Zatim smo pokušali registrirati drugu udrugu sa istim tim OIB-om (ali drugačijim korisničkim imenom, koji također mora biti jedinstven). Dobili smo očekivani odgovor sa statusom 406 NOT ACCEPTABLE. Nakon toga smo promijenili OIB u 22000000001 te ponovno poslali zahtjev za registracijom. Dobili smo očekivani odgovor 200 OK.

```
@Test
void shelterRegistrationOIBTaken() throws Exception {
    //stvara i registrira prvi shelter
    Location location1 = new Location();
    location1.setAddress("Ilica 37");
    location1.setCity("Zagreb");

    Shelter shelter1=new Shelter();
    shelter1.setOIB("22000000000");
    shelter1.setName("Shelter2");
    shelter1.setUsername("shelter2");
    shelter1.setPassword("12345");
    shelter1.setLocation(location1);
    shelter1.setImage("https://www.peanuts.com/sites/default/files/sn-color.jpg");

    String json1 = new ObjectMapper().writeValueAsString(shelter1);

    this.mockMvc.perform(post( urlTemplate: "/shelter/signup")
        .contentType(MediaType.APPLICATION_JSON).content(json1))
        .andExpect(status().isOk());

    //stvara i registriraj drugi shelter
    Shelter shelter2=new Shelter();
    //OIB isti kao u gornjem shelteru - NOT ACCEPTABLE
    shelter2.setOIB(shelter1.getOIB());
    shelter2.setName("Shelter3");
    shelter2.setUsername("shelter3");
    shelter2.setPassword("12345");
    shelter2.setLocation(location1);
    shelter2.setImage("https://www.peanuts.com/sites/default/files/sn-color.jpg");

    json1 = new ObjectMapper().writeValueAsString(shelter2);
    this.mockMvc.perform(post( urlTemplate: "/shelter/signup")
        .contentType(MediaType.APPLICATION_JSON).content(json1))
        .andExpect(status().isNotAcceptable());

    //popravi OIB - OK
    shelter2.setOIB("22000000001");
    json1 = new ObjectMapper().writeValueAsString(shelter2);

    this.mockMvc.perform(post( urlTemplate: "/shelter/signup")
        .contentType(MediaType.APPLICATION_JSON).content(json1)).
        andExpect(status().isOk());
}
```

2. ispitni primjer - registracija udruge sa zauzetim korisničkim imenom

U ispitnom primjeru prvo smo stvorili i registrirali prvu udrugu sa zadanim korisničkim imenom "ella". Dobili smo očekivani odgovor sa statusom 200 OK. Zatim smo pokušali registrirati drugu udrugu sa istim tim korisničkim imenom (ali drugačijim OIB-om, koji također mora biti jedinstven). Dobili smo očekivani odgovor sa statusom 409 CONFLICT. Možemo primjetiti da je to drugačiji odgovor od onog u gornjem testu kada šaljemo OIB koji nije jedinstven (409 NOT ACCEPTED). To je zato što smo pomoću tih odgovora razlikovali kakav odgovor tj. feedback treba dati korisniku pri registraciji. Nakon toga smo promijenili korisničko ime u "lily" te ponovno poslali zahtjev za registracijom. Dobili smo očekivani odgovor 200 OK.

```
@Test
void shelterRegistrationUsernameTaken() throws Exception {
    //stvara i registrira shelter
    Location location1 = new Location();
    location1.setAddress("Ilica 38");
    location1.setCity("Zagreb");

    Shelter shelter1=new Shelter();
    shelter1.setOIB("12345678989");
    shelter1.setName("Ella");
    shelter1.setUsername("ella");
    shelter1.setPassword("12345");
    shelter1.setLocation(location1);
    shelter1.setImage("https://www.peanuts.com/sites/default/files/sn-color.jpg");

    String json1 = new ObjectMapper().writeValueAsString(shelter1);
    this.mockMvc.perform(post( urlTemplate: "/shelter/signup")
        .contentType(MediaType.APPLICATION_JSON).content(json1))
        .andExpect(status().isOk());

    //stvari i registriraj drugi shelter s istim usernamemom kao prvi - CONFLICT
    Shelter shelter2 = new Shelter();
    shelter2.setOIB("12344444447");
    shelter2.setName("Ella");
    shelter2.setUsername("ella");
    shelter2.setPassword("12345");
    shelter2.setLocation(location1);
    shelter2.setImage("https://www.peanuts.com/sites/default/files/sn-color.jpg");

    json1 = new ObjectMapper().writeValueAsString(shelter2);
    this.mockMvc.perform(post( urlTemplate: "/shelter/signup")
        .contentType(MediaType.APPLICATION_JSON).content(json1))
        .andExpect(status().isConflict());

    //postavi drugaciji username - OK
    shelter2.setUsername("lily");
    json1 = new ObjectMapper().writeValueAsString(shelter2);

    this.mockMvc.perform(post( urlTemplate: "/shelter/signup")
        .contentType(MediaType.APPLICATION_JSON).content(json1))
        .andExpect(status().isOk());
```

3. ispitni primjer - mijenjanje podataka udruge sa i bez autorizacije

U ispitnom primjeru prvo smo stvorili i registrirali udrugu. Dobili smo očekivani odgovor sa statusom 200 OK. Također, u tijelu odgovora nam je vraćena ta udruga koju smo upravo registrirali uz dodatak ID-a udruge koji je stvoren na backendu. Taj vraćeni objekt smo spremili u našu varijablu udruge jer ćemo koristiti vraćeni ID u putanji za mijenjanje podataka. Zatim smo promijenili ime udruge u "Novo-Ime" te poslali zahtjev za promjenom. Dobili smo očekivani odgovor sa statusom 401 UNAUTHORIZED jer nismo dodali autorizaciju na zahtjev. Nakon toga smo pokušali ponovno sa dodanom autorizacijom te dobili očekivani odgovor 200 OK te udrugu sa promijenjenim imenom u tijelu odgovora. Na kraju smo provjerili da je u vraćenoj udruzi stvarno promijenjeno ime.

```
@Test
```

```
void updateShelterInfo() throws Exception {

    //stvori i registriraj shelter
    Location location1 = new Location();
    location1.setAddress("Ilica 35");
    location1.setCity("Zagreb");

    Shelter shelter1=new Shelter();
    shelter1.setOIB("11111110000");
    shelter1.setName("Shelter1");
    shelter1.setUsername("shelter1");
    shelter1.setPassword("12345");
    shelter1.setLocation(location1);
    shelter1.setImage("https://www.peanuts.com/sites/default/files/sn-color.jpg");

    String json1 = new ObjectMapper().writeValueAsString(shelter1);
    String izvornaLozinka = shelter1.getPassword();

    MvcResult result = this.mockMvc.perform(post( urlTemplate: "/shelter/signup")
        .contentType(MediaType.APPLICATION_JSON).content(json1)
        .andExpect(status().isOk()).andReturn();

    String content = result.getResponse().getContentAsString();
    shelter1 = new ObjectMapper().readValue(content, Shelter.class);

    //promjena imena
    shelter1.setName("NovoIme");
    json1 = new ObjectMapper().writeValueAsString(shelter1);

    //bez autentifikacije - unauthorized
    this.mockMvc.perform(post( urlTemplate: "/shelter/update/" +shelter1.getShelterId())
        .contentType(MediaType.APPLICATION_JSON).content(json1))
        .andExpect(status().isUnauthorized());

    //autentifikacija
    String auth = Base64Utils.encodeToString((shelter1.getUsername() + ":" + izvornaLozinka).getBytes());
    //dohvaćanje info
    result = this.mockMvc.perform(post( urlTemplate: "/shelter/update/" +shelter1.getShelterId())
        .contentType(MediaType.APPLICATION_JSON).content(json1)
        .header(HttpHeaders.AUTHORIZATION, ...values: "Basic " + auth))
        .andExpect(status().isOk()).andReturn();

    content = result.getResponse().getContentAsString();
    shelter1 = new ObjectMapper().readValue(content, Shelter.class);
    assertEquals(shelter1.getName(), actual: "NovoIme");
}
```

4. ispitni primjer - brisanje udruge sa i bez autorizacije

U ispitnom primjeru prvo smo stvorili i registrirali udrugu. Dobili smo očekivani odgovor sa statusom 200 OK. Također, u tijelu odgovora nam je vraćena ta udruga koju smo upravo registrirali uz dodatak ID-a udruge koji je stvoren na backendu. Taj vraćeni objekt smo spremili u našu varijablu udruge jer ćemo koristiti vraćeni ID u putanji za brisanje udruge. Zatim smo poslali zahtjev za brisanjem bez autorizacije. Dobili smo očekivani odgovor sa statusom 401 UNAUTHORIZED jer nismo dodali autorizaciju na zahtjev. Nakon toga smo pokušali ponovno sa dodanom autorizacijom te dobili očekivani odgovor 200 OK.

@Test

```
void deleteShelter() throws Exception {

    //stvori i registriraj shelter
    Location location1 = new Location();
    location1.setAddress("Ilica 39");
    location1.setCity("Zagreb");

    Shelter shelter1=new Shelter();
    shelter1.setOIB("5556667788");
    shelter1.setName("Dita");
    shelter1.setUsername("dita");
    shelter1.setPassword("12345");
    shelter1.setLocation(location1);
    shelter1.setImage("https://www.peanuts.com/sites/default/files/sn-color.jpg");

    String json1 = new ObjectMapper().writeValueAsString(shelter1);
    String izvornaLozinka = shelter1.getPassword();

    MvcResult result = this.mockMvc.perform(post( urlTemplate: "/shelter/signup")
        .contentType(MediaType.APPLICATION_JSON).content(json1))
        .andExpect(status().isOk()).andReturn();

    String content = result.getResponse().getContentAsString();
    shelter1 = new ObjectMapper().readValue(content, Shelter.class);

    //brisanje bez autentifikacije - unauthorized
    this.mockMvc.perform(post( urlTemplate: "/shelter/delete/"+shelter1.getShelterId())
        .contentType(MediaType.APPLICATION_JSON).content(json1))
        .andExpect(status().isUnauthorized());

    //autentifikacija
    String auth = Base64Utils.encodeToString((shelter1.getUsername()+":"+izvornaLozinka).getBytes());

    //brisanje sa autentifikacijom - OK
    this.mockMvc.perform(post( urlTemplate: "/shelter/delete/"+shelter1.getShelterId())
        .contentType(MediaType.APPLICATION_JSON).content(json1)
        .header(HttpHeaders.AUTHORIZATION, ...values: "Basic " + auth))
        .andExpect(status().isOk());
}

}
```

5. ispitni primjer - registracija šetača sa zauzetom email adresom

U ispitnom primjeru prvo smo stvorili dva šetača sa jednakim email adresama (ali različitim korisničkim imenima, koja također moraju biti jedinstvena). Registrirali smo prvog i dobili očekivani odgovor sa statusom 200 OK. Zatim smo pokušali registrirati drugog šetača te smo dobili očekivani odgovor sa statusom 406 NOT ACCEPTABLE jer email adresa šetača mora biti jedinstvena.

```
@Test
void registerWalkerEmailTaken() throws Exception {

    //pokušaj registriranja dva korisnika s istim usernameom - not acceptable

    Walker walker1 = new Walker();
    walker1.setUsername("korisnik1");
    walker1.setEmail("korisnik@gmail.com");
    walker1.setFirstName("Kori");
    walker1.setLastName("Snik");
    walker1.setPassword("testiranje1");

    Walker walker2 = new Walker();
    walker2.setUsername("korisnik2");
    walker2.setEmail("korisnik@gmail.com");
    walker2.setFirstName("Kori");
    walker2.setLastName("Snik");
    walker2.setPassword("testiranje2");

    String json1 = new ObjectMapper().writeValueAsString(walker1);
    String json2 = new ObjectMapper().writeValueAsString(walker2);

    this.mockMvc.perform(post( urlTemplate: "/walker/signup")
        .contentType(MediaType.APPLICATION_JSON).content(json1))
        .andExpect(status().isOk());

    this.mockMvc.perform(post( urlTemplate: "/walker/signup")
        .contentType(MediaType.APPLICATION_JSON).content(json2))
        .andExpect(status().isNotAcceptable());
}
```

6. ispitni primjer - registracija šetača sa zauzetim korisničkim imenom

U ispitnom primjeru prvo smo stvorili dva šetača sa jednakim korisničkim imenom (ali različitim email adresama, koje također moraju biti jedinstvene). Registrirali smo prvog i dobili očekivani odgovor sa statusom 200 OK. Zatim smo pokušali registrirati drugog šetača te smo dobili očekivani odgovor sa statusom 409 CONFLICT jer je korisničko ime šetača mora biti jedinstveno. Kao i u primjeru s udrugama, šaljemo različite statuse ovisno je li došlo do pogreške zbog korisničkog imena ili lozinke jer tako znamo kakvu poruku poslati korisniku.

```
@Test
void registerWalkerUsernameTaken() throws Exception {

    //pokušaj registriranja dva korisnika s istim usernameom - confllict

    Walker walker1 = new Walker();
    walker1.setUsername("korisnik3");
    walker1.setEmail("korisnik3@gmail.com");
    walker1.setFirstName("Kori");
    walker1.setLastName("Snik");
    walker1.setPassword("testiranje1");

    Walker walker2 = new Walker();
    walker2.setUsername("korisnik3");
    walker2.setEmail("korisnik33@gmail.com");
    walker2.setFirstName("Kori");
    walker2.setLastName("Snik");
    walker2.setPassword("testiranje2");

    String json1 = new ObjectMapper().writeValueAsString(walker1);
    String json2 = new ObjectMapper().writeValueAsString(walker2);

    this.mockMvc.perform(post( urlTemplate: "/walker/signup")
        .contentType(MediaType.APPLICATION_JSON).content(json1))
        .andExpect(status().isOk());

    this.mockMvc.perform(post( urlTemplate: "/walker/signup")
        .contentType(MediaType.APPLICATION_JSON).content(json2))
        .andExpect(status().isConflict());
}
```

7. ispitni primjer - mijenjanje podataka šetača sa i bez autorizacije

U ispitnom primjeru prvo smo stvorili i registrirali šetača. Dobili smo očekivani odgovor sa statusom 200 OK. Također, u tijelu odgovora nam je vraćen taj šetač kojeg smo upravo registrirali uz dodatak ID-a šetača koji je stvoren na backendu. Taj vraćeni objekt smo spremili u našu varijablu šetača jer ćemo koristiti vraćeni ID u putanji za mijenjanje podataka. Zatim smo promijenili korisničko ime šetača u "noviKorisnik" te poslali zahtjev za promjenom. Dobili smo očekivani odgovor sa statusom 401 UNAUTHORIZED jer nismo dodali autorizaciju na zahtjev. Nakon toga smo pokušali ponovno sa dodanom autorizacijom te dobili očekivani odgovor 200 OK te šetača sa promijenjenim korisničkim imenom u tijelu odgovora. Na kraju smo provjerili da je vraćenom šetaču stvarno promijenjeno ime.

```
@Test
```

```
void updateWalkerWithandWithoutAuthentication() throws Exception {
    Walker walker1 = new Walker();
    walker1.setUsername("korisnik4");
    walker1.setEmail("korisnik4@gmail.com");
    walker1.setFirstName("Kori");
    walker1.setLastName("Snik");
    walker1.setPassword("testiranje1");
    String izvorniPassword = walker1.getPassword();

    String json1 = new ObjectMapper().writeValueAsString(walker1);

    MvcResult result = this.mockMvc.perform(post( urlTemplate: "/walker/signup")
        .contentType(MediaType.APPLICATION_JSON).content(json1))
        .andExpect(status().isOk()).andReturn();

    String content = result.getResponse().getContentAsString();
    walker1 = new ObjectMapper().readValue(content, Walker.class);

    assertEquals(walker1.getUsername(), actual: "korisnik4");

    //promijeni username
    String stariUsername = walker1.getUsername();
    walker1.setUsername("noviKorisnik");
    json1 = new ObjectMapper().writeValueAsString(walker1);

    //bez autentifikacije - unauthorized
    this.mockMvc.perform(post( urlTemplate: "/walker/update/" + walker1.getWalkerId())
        .contentType(MediaType.APPLICATION_JSON).content(json1))
        .andExpect(status().isUnauthorized());

    //sa autentifikacijom - ok
    String auth = Base64Utils.encodeToString((stariUsername + ":" + izvorniPassword).getBytes());
    result = this.mockMvc.perform(post( urlTemplate: "/walker/update/" + walker1.getWalkerId())
        .header(HttpHeaders.AUTHORIZATION, ...values: "Basic " + auth)
        .contentType(MediaType.APPLICATION_JSON).content(json1))
        .andExpect(status().isOk()).andReturn();

    content = result.getResponse().getContentAsString();
    walker1 = new ObjectMapper().readValue(content, Walker.class);

    assertEquals(walker1.getUsername(), actual: "noviKorisnik");
}
```

8. ispitni primjer - mijenjanje podataka šetača sa i bez autorizacije

U ispitnom primjeru prvo smo stvorili i registrirali šetača. Dobili smo očekivani odgovor sa statusom 200 OK. Zatim smo stvorili novu šetnju te poslali zahtjev za rezervacijom šetnje (odabrali smo psa sa ID-jem 13 i stavili 13 u stazu "/reserve/13"). Dobili smo očekivani odgovor sa statusom 401 UNAUTHORIZED jer nismo dodali autorizaciju na zahtjev. Nakon toga smo pokušali ponovno sa dodanom autorizacijom te dobili očekivani odgovor 200 OK.

```
@Test
void createReservationAuthorization() throws Exception {

    //registracija korisnika
    Walker walker1 = new Walker();
    walker1.setUsername("korisnik5");
    walker1.setEmail("korisnik5@gmail.com");
    walker1.setFirstName("Kori");
    walker1.setLastName("Snik");
    walker1.setPassword("testiranje1");
    String izvorniPassword = walker1.getPassword();

    String json1 = new ObjectMapper().writeValueAsString(walker1);

    MvcResult result = this.mockMvc.perform(post( urlTemplate: "/walker/signup")
        .contentType(MediaType.APPLICATION_JSON).content(json1))
        .andExpect(status().isOk()).andReturn();

    String content = result.getResponse().getContentAsString();
    walker1 = new ObjectMapper().readValue(content, Walker.class);

    assertEquals(walker1.getUsername(), actual: "korisnik5");

    //stvaranje šetnje
    Walk walk = new Walk();
    walk.setDateTime(new Timestamp(new Date().getTime()));
    walk.setDuration(5);

    json1 = new ObjectMapper().writeValueAsString(walk);

    //bez autentifikacije - unauthorized
    this.mockMvc.perform(post( urlTemplate: "/reserve/13")
        .contentType(MediaType.APPLICATION_JSON).content(json1))
        .andExpect(status().isUnauthorized());

    //sa autentifikacijom - ok
    String auth = Base64Utils.encodeToString(
        (walker1.getUsername() + ":" + izvorniPassword).getBytes());
    result = this.mockMvc.perform(post( urlTemplate: "/reserve/13")
        .header(HttpHeaders.AUTHORIZATION, ...values: "Basic " + auth)
        .contentType(MediaType.APPLICATION_JSON).content(json1))
        .andExpect(status().isOk()).andReturn();

}
```

Rezultati izvođenja ispitnih primjera

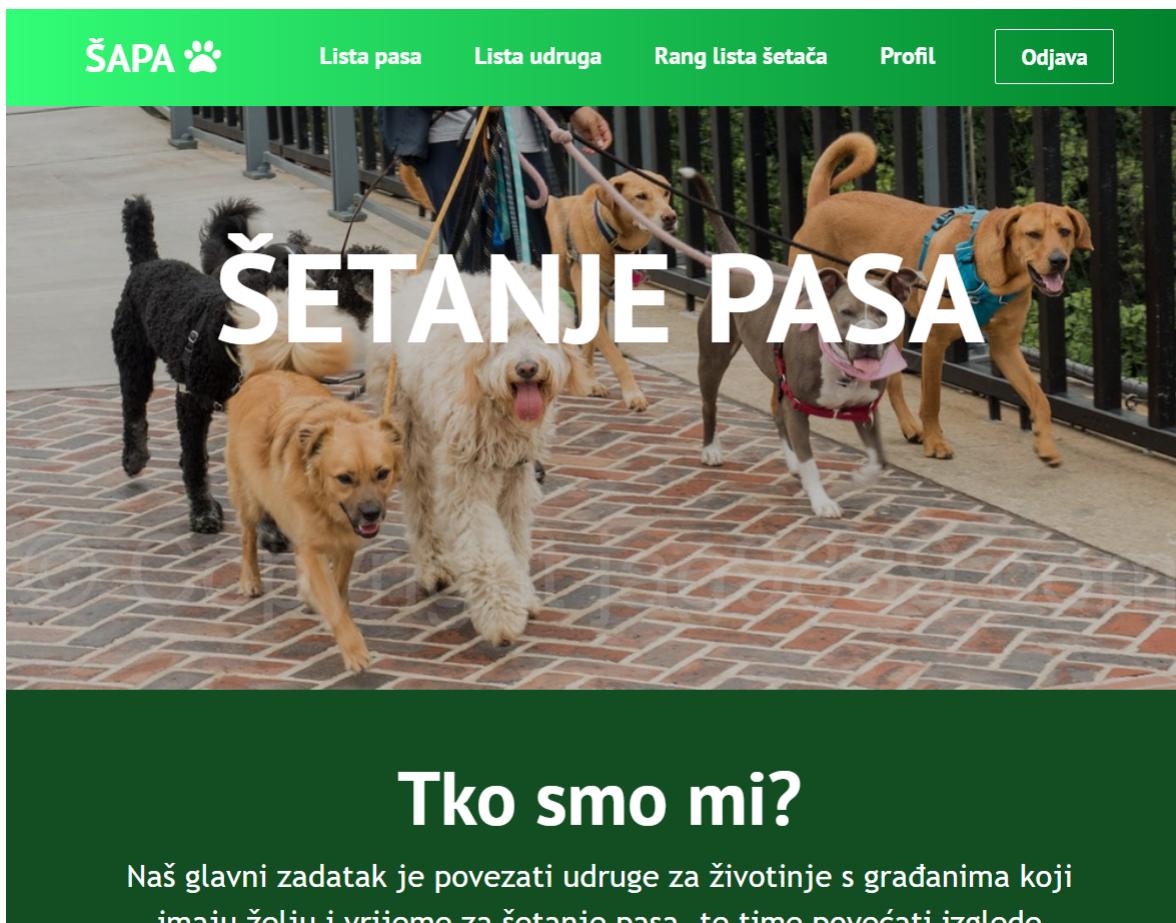
✓ rest (cyfer)	2 s 90 ms
▼ ✓ ShelterControllerTest	1 s 429 ms
✓ shelterRegistrationOIBTaken()	663 ms
✓ updateShelterInfo()	382 ms
✓ deleteShelter()	204 ms
✓ shelterRegistrationUsernameTaken()	180 ms
▼ ✓ WalkerControllerTest	661 ms
✓ updateWalkerWithandWithoutAuthentification()	276 ms
✓ createReservationAuthorization()	207 ms
✓ registerWalkerEmailTaken()	89 ms
✓ registerWalkerUsernameTaken()	89 ms

5.2.2 Ispitivanje sustava

Za ispitivanje sustava koristili smo radni okvir Selenium. Uz podršku Selenium WebDrivera, napisali smo četiri ispita u programskom jeziku Java. Kao i u gornjim primjerima, svi ispiti su uspješno prošli jer smo pri slanju krivih podataka očekivali da su korisniku prikazane informacije o tome što je krivo. Svi ispiti su vezani uz varijacije registracija šetača i udruge. Očekivali smo da sustav zna reći koji podaci su neispravni kako bi ih mogao ispraviti.

1. ispitni primjer - registracija šetača sa ispravnim podatcima

U ovom primjeru smo demonstrirali uspješnu registraciju šetača. Upisali smo sve ispravne podatke - jedinstvenu email adresu i korisničko ime te lozinke koje se podudaraju. Od sustava smo očekivali da nas onda preusmjeri na početnu stranicu, što je ispunjeno.



Slika 5.1: Ispitni primjer 1 - Početna stranica

```
@Test
void testRegisterWalkerGoodCreds() throws InterruptedException {

    System.setProperty("webdriver.chrome.driver",
        "C:\\\\Program Files (x86)\\\\Chrome Driver\\\\chromedriver.exe");
    WebDriver driver = new ChromeDriver();
    driver.manage().timeouts().implicitlyWait( time: 10, TimeUnit.SECONDS);
    //stranica za registraciju korisnika
    driver.get("localhost:3000/sign-up");

    //URL pocetne stranice
    String pocetniURL = "http://localhost:3000/";

    //Upis podataka
    WebElement element = driver.findElement(By.name("username"));
    element.sendKeys( ...keysToSend: "user123");

    element = driver.findElement(By.name("firstName"));
    element.sendKeys( ...keysToSend: "MyName");

    element = driver.findElement(By.name("lastName"));
    element.sendKeys( ...keysToSend: "MyLastName");

    element = driver.findElement(By.name("email"));
    element.sendKeys( ...keysToSend: "user123@gmail.com");

    element = driver.findElement(By.name("password"));
    element.sendKeys( ...keysToSend: "12345");

    element = driver.findElement(By.name("repeatPassword"));
    element.sendKeys( ...keysToSend: "12345");

    //submit (register)
    driver.findElement(By.id("register")).click();

    //pricekaj da se stranica preusmjeri
    Thread.sleep( millis: 500);

    String redirURL = driver.getCurrentUrl();

    //ako je registracija uspješna,
    // korisnik treba biti preusmjeren na početnu stranicu
    assertEquals(redirURL, pocetniURL);

    driver.quit();
}
```

2. ispitni primjer - registracija šetača s neispravnim podatcima

U ovom primjeru smo demonstrirali neuspješnu registraciju šetača. Upisali smo isto korisničko ime kao i u gornjem primjeru - "user123", a kako smo ispite pokretali slijedno, takav korisnik je već bio u bazi. Od sustava smo očekivali da nam javi kakva je pogreška u pitanju. Sustav je ostao na istoj stranici (nije se dogodilo preusmjeravanje kao pri uspješnoj registraciji) te nam je ispisao poruku "Neuspješna registracija - korisničko ime je zauzeto." U ispitnom primjeru smo provjerili je li doista ispisana ta poruka. Ispod je priložena slika izvođenja u browseru te kod ispitnog primjera.

The screenshot shows a registration form for a dog walker. The fields are filled as follows:

- Ime: MyName
- Prezime: MyLastName
- E-mail: user321@gmail.com
- Lozinka: (redacted)
- Ponovi lozinku: (redacted)

A red error message at the bottom left of the form area states: **Neuspješna registracija - korisničko ime je zauzeto.** (Registration failed - user name is taken).

At the bottom right of the form is a green button labeled **Registiraj se** (Register). Below the form, there is a blue link that says **Želite registrirati udrugu?** (Do you want to register a club?).

Slika 5.2: Ispitni primjer 2 - Korisničko ime zauzeto

```
@Test
void testRegistrateWalkerBadCreds() {

    System.setProperty("webdriver.chrome.driver",
        "C:\\\\Program Files (x86)\\\\Chrome Driver\\\\chromedriver.exe");
    WebDriver driver = new ChromeDriver();
    driver.manage().timeouts().implicitlyWait( time: 10, TimeUnit.SECONDS);
    //stranica za registraciju korisnika
    driver.get("localhost:3000/sign-up");

    //pokušati ćemo registrirati korisnika sa zauzetim usernamemom

    WebElement element = driver.findElement(By.name("username"));
    element.sendKeys( ...keysToSend: "user123");

    element = driver.findElement(By.name("firstName"));
    element.sendKeys( ...keysToSend: "MyName");

    element = driver.findElement(By.name("lastName"));
    element.sendKeys( ...keysToSend: "MyLastName");

    element = driver.findElement(By.name("email"));
    element.sendKeys( ...keysToSend: "user321@gmail.com");

    element = driver.findElement(By.name("password"));
    element.sendKeys( ...keysToSend: "12345");

    element = driver.findElement(By.name("repeatPassword"));
    element.sendKeys( ...keysToSend: "12345");

    //submit (registriraj)
    driver.findElement(By.id("register")).click();

    //element koji prikazuje tekst pogreške
    element = driver.findElement(By.className("error"));
    String text = element.getText();

    // pogreška - zauzeto korisničko ime
    assertEquals(text,
        actual: "Neuspješna registracija - korisničko ime je zauzeto.");
}

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

Slika 5.3: Ispitni primjer 2 - registracija šetača s neispravnim podatcima

3. ispitni primjer - registracija udruge sa ispravnim podatcima

U ovom primjeru smo demonstrirali uspješnu registraciju udruge uz ispravak lozinke. Upisali smo sve ispravne podatke osim lozinke i ponovljene lozinke, koje su bile različite. Od sustava smo očekivali da nam javi kakva je pogreška u pitanju. Sustav je ostao na istoj stranici (nije se dogodilo preusmjeravanje kao pri uspješnoj registraciji) te nam je ispisao poruku "Lozinke se ne poklapaju." Zatim smo ispravili ponovljenu lozinku i ponovno poslali zahtjev za registracijom. Sustav nas je očekivano preusmjerio na početnu stranicu. Ispod je priložena slika izvođenja u browseru (neispravne lozinke) te kod ispitnog primjera.

The screenshot shows a registration form for a dog shelter. The fields filled with correct data are:

- Ime udruge: NewShelter
- Grad udruge: Zagreb
- Adresa udruge: Ulica 123
- Ponovi lozinku: (The password entered here matches the one in the first field)

The field where the password was misspelled contains four dots ("....."). Below the password fields, a red error message reads "Lozinke se ne poklapaju." (Passwords do not match). At the bottom of the form is a large green button labeled "Registriraj se" (Register).

Slika 5.4: 3. ispitni primjer - Nepodudaranje lozinka

```
@Test
void testRegisterShelterGoodCreds() throws InterruptedException {

    System.setProperty("webdriver.chrome.driver",
        "C:\\\\Program Files (x86)\\\\Chrome Driver\\\\chromedriver.exe");
    WebDriver driver = new ChromeDriver();
    driver.manage().timeouts().implicitlyWait( time: 10, TimeUnit.SECONDS);
    //stranica za registraciju udruge
    driver.get("localhost:3000/RegUdr");

    String pocetniURL = "http://localhost:3000/";

    //registrirati ćemo novu udrugu, ali ćemo prvo uz lozinke koje se ne podudaraju,
    //a onda uz ispravne lozinke

    WebElement element = driver.findElement(By.name("username"));
    element.sendKeys( ...keysToSend: "shelter123");

    element = driver.findElement(By.name("oib"));
    element.sendKeys( ...keysToSend: "11111111123");

    element = driver.findElement(By.name("name"));
    element.sendKeys( ...keysToSend: "NewShelter");

    element = driver.findElement(By.name("city"));
    element.sendKeys( ...keysToSend: "Zagreb");

    element = driver.findElement(By.name("address"));
    element.sendKeys( ...keysToSend: "Ilica 123");

    element = driver.findElement(By.name("password"));
    element.sendKeys( ...keysToSend: "12345");
```

```
element = driver.findElement(By.name("repeatPassword"));
element.sendKeys( ...keysToSend: "123456");

//submit (registriraj)
driver.findElement(By.id("register")).click();

//element koji prikazuje tekst pogreške
element = driver.findElement(By.className("error"));
String text = element.getText();

// lozinke se ne podudaraju
assertEquals(text, actual: "Lozinke se ne podudaraju.");

//ispravimo lozinku
element = driver.findElement(By.name("repeatPassword"));
element.clear();
element.sendKeys( ...keysToSend: "12345");

//submit (register)
driver.findElement(By.id("register")).click();

//pričekaj da se stranica preusmjeri
Thread.sleep( millis: 500);

String redirURL = driver.getCurrentUrl();

//ako je registracija uspješna, korisnik treba biti preusmjeren na početnu stranicu
assertEquals(redirURL, pocetniURL);

driver.quit();
}
```

Slika 5.5: Ispitni primjer 3 - registracija udruge sa ispravnim podatcima

4. ispitni primjer - registracija udruge s neispravnim podatcima

U ovom primjeru smo demonstrirali neuspješnu registraciju udruge. Upisali smo isti OIB kao i u gornjem (trećem) primjeru - "11111111123", a kako smo ispite pokretali slijedno, takva udruga je već bila u bazi. Od sustava smo očekivali da nam javi kakva je pogreška u pitanju. Sustav je ostao na istoj stranici (nije se dogodilo preusmjeravanje kao pri uspješnoj registraciji) te nam je ispisao poruku "Neuspješna registracija - postoji već udruga sa danim OIB-om." U ispitnom primjeru smo provjerili je li doista ispisana ta poruka. Ispod je priložena slika izvođenja u browseru te kod ispitnog primjera.

Lista pasa Lista udruga Rang L

Ime udruge:
NewShelter

Grad udruge:
Zagreb

Adresa udruge:
Ilica 123

Lozinka:
.....

Ponovi lozinku:
.....

Neuspješna registracija - postoji već udruga sa danim OIB-om.

Registiraj se

Želite se registrirati kao korisnik?

Slika 5.6: Registracija udruge sa zauzetim OIB-om

```
@Test
void testRegisterShelterBadCreds() {

    System.setProperty("webdriver.chrome.driver",
        "C:\\Program Files (x86)\\Chrome Driver\\chromedriver.exe");
    WebDriver driver = new ChromeDriver();
    driver.manage().timeouts().implicitlyWait( time: 10, TimeUnit.SECONDS);
    //stranica za registraciju udruge
    driver.get("localhost:3000/RegUdr");

    //pokušati čemo registrirati udrugu sa zauzetim OIB-om

    WebElement element = driver.findElement(By.name("username"));
    element.sendKeys( ...keysToSend: "shelter1234");

    element = driver.findElement(By.name("oib"));
    element.sendKeys( ...keysToSend: "11111111123");

    element = driver.findElement(By.name("name"));
    element.sendKeys( ...keysToSend: "NewShelter");

    element = driver.findElement(By.name("city"));
    element.sendKeys( ...keysToSend: "Zagreb");

    element = driver.findElement(By.name("address"));
    element.sendKeys( ...keysToSend: "Ilica 123");

    element = driver.findElement(By.name("password"));
    element.sendKeys( ...keysToSend: "12345");

    element = driver.findElement(By.name("repeatPassword"));
    element.sendKeys( ...keysToSend: "12345");
    //submit (registriraj)
    driver.findElement(By.id("register")).click();

    //element koji prikazuje tekst pogreške
    element = driver.findElement(By.className("error"));
    String text = element.getText();

    assertEquals(text,
        actual: "Neuspješna registracija - postoji već udruga sa danim OIB-om.");
    //driver.quit();
}
```

Slika 5.7: Ispitni primjer 4 - registracija udruge s neispravnim podatcima

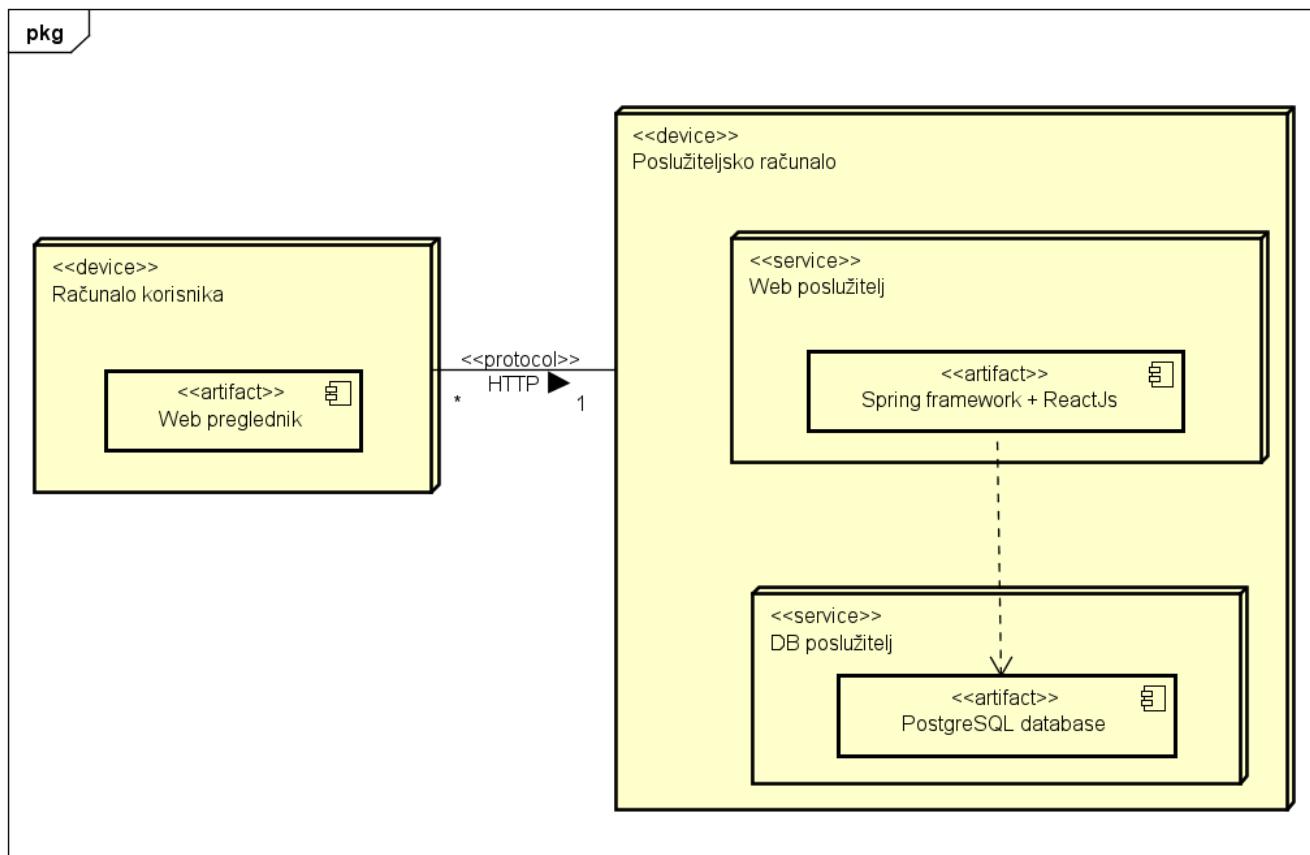
Rezultati izvođenja ispitnih primjera

Test Results		22 s 796 ms
▼	✓ JUSeTest	22 s 796 ms
▼	✓ testRegisterShelterGoodCreds()	7 s 191 ms
▼	✓ testRegisterWalkerGoodCreds()	5 s 875 ms
▼	✓ testRegisterShelterBadCreds()	4 s 936 ms
▼	✓ testRegisterWalkerBadCreds()	4 s 794 ms

Slika 5.8: Rezultati izvođenja ispitnih primjera - Selenium

5.3 Dijagram razmještaja

Topologija našeg projekta sastoji se od dva čvora, od kojih su oba uređaji. Jedan je korisničko računalo, drugi je poslužiteljsko računalo. Na poslužiteljskom računalu nalaze se svi potrebni izvorni kodovi i baza podataka kako bi se ostvario uspješan deploy web aplikacije. Programska potpora sastoji se od kombinacije Spring frameworka i ReactJs-a. Također, korisničko i poslužiteljsko računalo međusobno razmjenjuju informacije putem HTTP protokola.



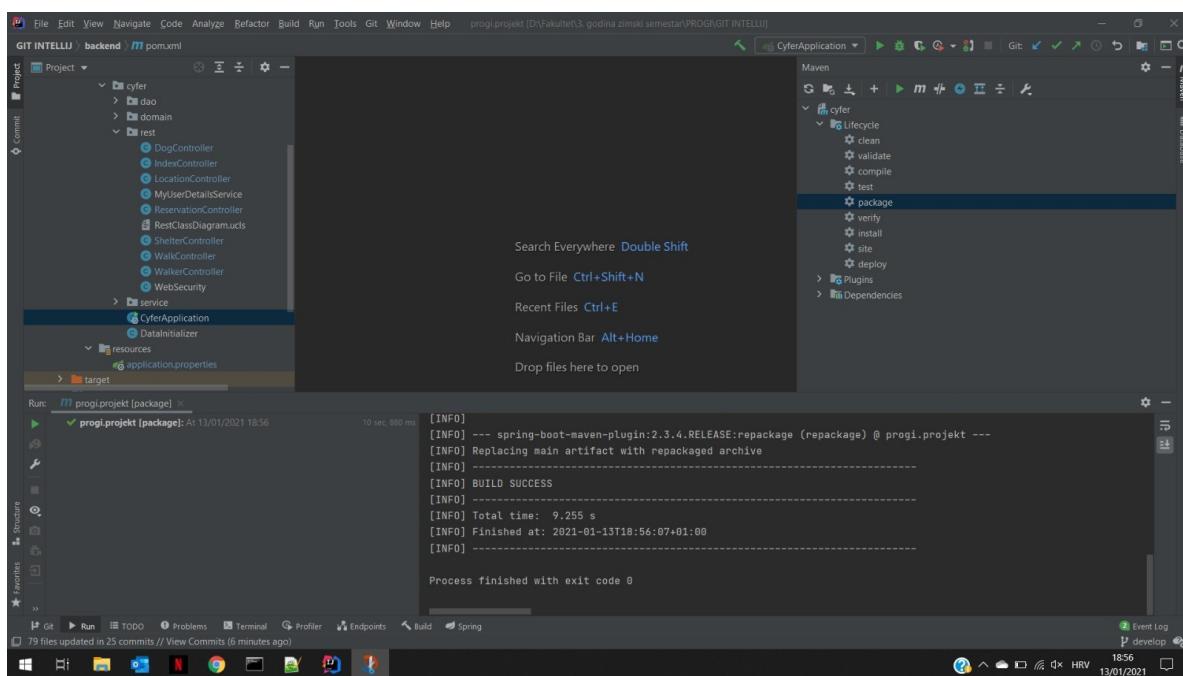
Slika 5.9: Dijagram razmještaja

5.4 Upute za puštanje u pogon

Puštanje aplikacije u pogon (engl. deploy) proces je na kraju kojega aplikacija koju smo prethodno pokretali s našeg lokalnog poslužitelja (engl. localhost) postaje javno dostupna na internetu. Kao cloud platformu za deploy odabrali smo Heroku, na koji smo kao odvojene aplikacije postavili frontend i backend. Na backend aplikaciji smo podesili PostgreSQL bazu koja se isto tako nalazi na Heroku. Na kraju smo povezali frontend i backend aplikacije putem proxyja tako da su na jednom mjestu dostupne sve funkcionalnosti. Frontend i backend smo deployali kao odvojene aplikacije jer imamo monorepozitorij na GitLabu, što bi nam otežalo proces puštanja u pogon da smo se odlučili za opciju deploya preko gita.

5.4.1 Puštanje u pogon backend aplikacije

Puštanje u pogon backend aplikacije Prvi korak deploya backend bio je postavljanje porta s kojeg se slušaju zahtjevi na vrijednost PORT (varijabla okruženja) ili 8080 ako je slobodan. Drugi korak bio je generiranje izvršnih datoteka našeg Maven projekta koje smo napravili naredbom package. Tada je unutar mape target nastala .jar datoteka našeg projekta (progi.projekt-1.0.jar) koju smo onda deployali na Heroku aplikaciju cyfer-backend koju smo prethodno napravili.



Slika 5.10: Generiranje .jar datoteke

Kako bismo pustili u pogon našu backend aplikaciju, pozicionirali smo se u naredbenom retku u mapu backend u našem projektu i izvršili sljedeću naredbu:

deploy:jar target/progi.projekt-1.0.jar -app cyfer-backend

Backend aplikacija je nakon toga bila dostupna na internetu na URI-ju ***https://cyfer-backend.herokuapp.com***. Grafičkog sučelja naravno nema, pa na sve zahtjeve koje šaljemo (samo putem URI-ja) dobivamo odgovore u obliku JSON-a.

5.4.2 Dodavanje baze podataka na Heroku

U fazi razvoja aplikacije koristili smo H2, dok smo na kraju izrade prešli na PostgreSQL bazu na localhostu. Kako bi baza bila dostupna nakon puštanja aplikacije u pogon, dodali smo ju na našu backend aplikaciju na Heroku kao add-on, te smo izmijenili application.properties kako bi bio omogućen pristup toj bazi. Da povezivanje bude moguće, morali smo stvoriti varijablu okruženja naredbom:

heroku run echo \$JDBC_DATABASE_URL

5.4.3 Puštanje u pogon frontend aplikacije

Deploy frontend aplikacije je nešto složeniji jer zahtjeva korištenje dockera i namještanje proxyja. Naime, dok smo aplikaciju pokretali s localhosta u datoteci package.json imali smo definiran proxy (odnosno posrednik koji s frontenda preusmjerava HTTP zahtjeve na backend i vraća odgovor) koji je slušao na portu 8080 (defaultni port za spring boot backend aplikacije). Pri puštanju pogon morali smo dodati nekoliko novih datoteka koje će konfigurirati usmjeravanje tih zahtjeva, kao i sam prikaz grafičkog sučelja aplikacije definiranog na frontendu. Za virtualizaciju i prikaz smo koristili docker kojega smo konfigurirali u datoteci Dockerfile, a kao poslužitelj nginx. Usmjeravanje putem proxyja smo konfigurirali novom datotekom default.conf unutar koje je zapisano da svi zahtjevi za backend trebaju imati prefiks /api, te smo taj prefiks dodali u kôd i na frontend i na backend kako bi mapiranje zahtjeva i vraćanje odgovora bilo funkcionalno. U Heroku smo stvorili aplikaciju cyfer-frontend. Pozicionirali smo se u frontend mapu našeg projekta i pokrenuli sljedeće naredbe iz naredbenog retka:

npm run build(izgradnja React aplikacije)

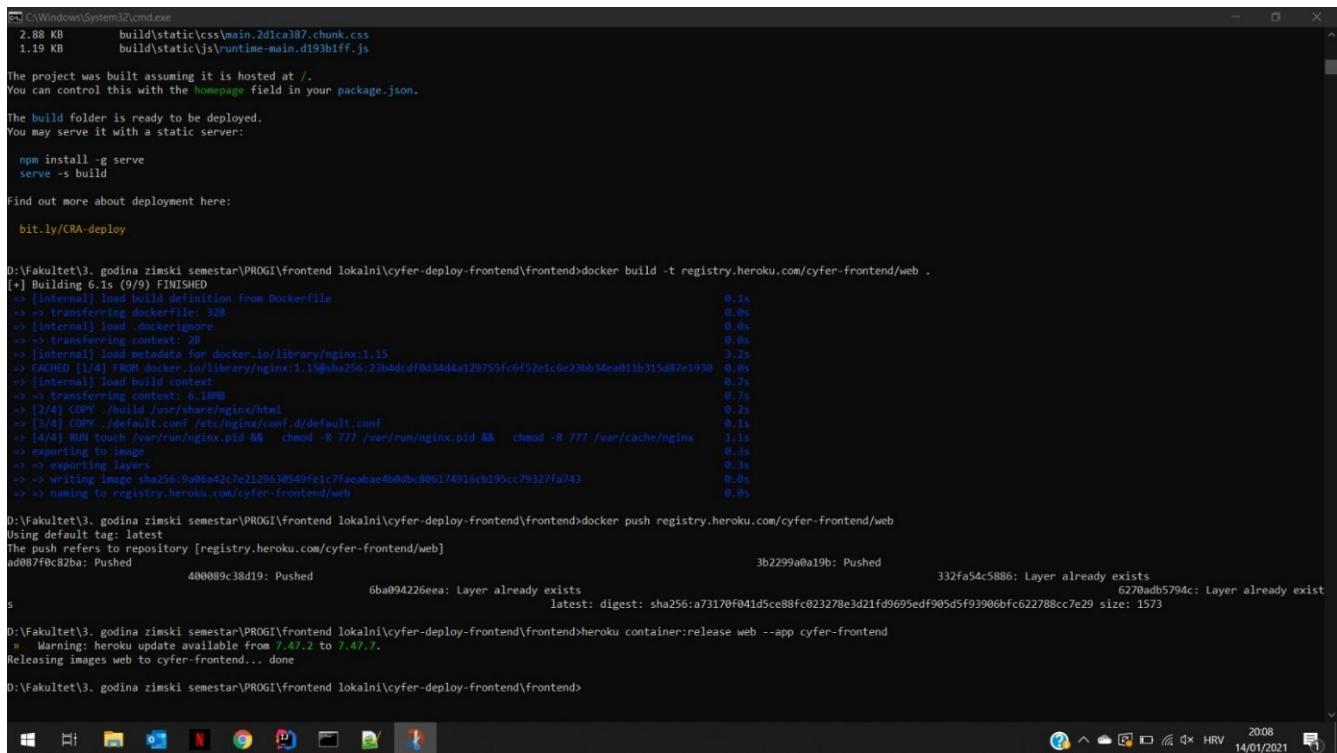
docker build -t registry.heroku.com/cyfer-frontend/web .(stvaranje slike)

docker push registry.heroku.com/cyfer-frontend/web („pushanje“ aplikacije na Heroku)

heroku container:release web –app cyfer-frontend (puštanje u pogon na dostupnom URL)

Aplikacija je nakon toga puštena u pogon i dostupna na:

<https://cyfer-frontend.herokuapp.com>.



```
C:\Windows\System32\cmd.exe
2.88 KB      build\static\css\main.2dica387.chunk.css
1.19 KB      build\static\js\runtime-main.e193b1ff.js

The project was built assuming it is hosted at /.
You can control this with the homepage field in your package.json.

The build folder is ready to be deployed.
You may serve it with a static server:

  npm install -g serve
  serve -s build

Find out more about deployment here:
  bit.ly/CRA-deploy

D:\Fakultet\3. godina zimski semestar\PROGI\frontend lokalni\cyfer-deploy-frontend>docker build -t registry.heroku.com/cyfer-frontend/web .
[*] Building 6.15 (9/9) FINISHED
=> [internal] load build definition from Dockerfile          0.1s
=> [internal] transfering Dockerfile: 32B                  0.0s
=> [internal] load metadata for docker.io/library/nginx:1.15 0.0s
=> [internal]  transfer context: 2B                         0.0s
=> [internal] load metadata for docker.io/library/nginx:1.15 3.2s
=> CACHED [1/4] FROM docker.io/library/nginx:1.15@sha256:23b4dcdf0d34d4a129755fc6f52e1c6e23bb34ea011b315d87e1930 0.0s
=> [internal] load build context                         0.7s
=> [internal] load build context                         0.7s
=> transferring context: 6.1MB                           0.2s
=> [2/4] COPY ./build /usr/share/nginx/html             0.2s
=> [3/4] COPY ./default.conf /etc/nginx/conf.d/default.conf 0.1s
=> [4/4] RUN touch /var/run/nginx.pid && chmod -R 777 /var/run/nginx.pid && chmod -R 777 /var/cache/nginx           1.1s
=> exporting to image                                    0.3s
=> --> exporting layers                                 0.3s
=> --> writing image sha256:9ba06a42c7e2129630349fe1c7facabae4b0db086174916cb195cc79327fa743 0.0s
=> --> naming to registry.heroku.com/cyfer-frontend/web 0.0s

D:\Fakultet\3. godina zimski semestar\PROGI\frontend lokalni\cyfer-deploy-frontend>docker push registry.heroku.com/cyfer-frontend/web
Using default tag: latest
The push refers to repository [registry.heroku.com/cyfer-frontend]
ad087ff0c82ba: Pushed          400009c38d19: Pushed          3b2299a0a19b: Pushed          332fa54c5886: Layer already exists
6ba094226eea: Layer already exists          latest: digest: sha256:a73170f041d5ce88fc023278e3d21fd9695edf905d5f93906bf622788cc7e29 size: 1573          6270ad5794c: Layer already exists
=> Warning: heroku update available from 7.47.2 to 7.47.7.
Releasing images web to cyfer-frontend... done

D:\Fakultet\3. godina zimski semestar\PROGI\frontend lokalni\cyfer-deploy-frontend>
```

Slika 5.11: Pokretanje naredbi za deploy aplikacije

The screenshot shows the Heroku dashboard interface. At the top, there's a navigation bar with the Heroku logo, a search bar labeled "Jump to Favorites, Apps, Pipelines, Spaces...", and user account options. Below the navigation, a "Personal" dropdown menu is open. A "New" button is also visible. A search bar at the top of the main content area is labeled "Filter apps and pipelines". Two application cards are listed:

- cyfer-backend**: Deployed to heroku-deploy, heroku-18, United States. It's a Container-based app.
- cyfer-frontend**: Deployed to Container, container, United States. It's also a Container-based app.

At the bottom of the dashboard, there are links for heroku.com, Blogs, Careers, Documentation, Support, Terms of Service, Privacy, Cookies, and a copyright notice for © 2021 Salesforce.com.

Slika 5.12: Deployani frontend i backend

6. Zaključak i budući rad

Zadatak naše grupe je bio napraviti aplikaciju za šetanje pasa iz udruga. Aplikacija treba omogućiti registraciju udruga i šetača, dodavanje pasa te rezervaciju šetnji. Projekt smo radili otprilike 16 tjedana te se dinamika u tih 16 tjedana značajno mijenjala.

Projekt je bio podijeljen u dvije faze koje su bile određene ocjenjivanjem našeg projekta. U prvoj fazi, dokumentacija je nosila značajno više bodova od same aplikacije, dok je u drugoj fazi bilo obrnuto. Prva faza je započela dobivanjem projektnog zadatka, uvodnim sastankom te zatim upoznavanjem sa pomoćnim alatima i framework-ovima pomoću kojih smo napravili aplikaciju. Također, u prvoj fazi se značajno radilo na slaganju baze podataka te je ona opisana u dokumentaciji. Ipak, kako je značajni dio prve predaje činila dokumentacija, većina fokusa je bila na njoj. Bilo je bitno definirati specifikaciju programske potpore - aktore, funkcionalne i nefunkcionalne zahtjeve. To nam je uvelike pomoglo za izgradnju same aplikacije jer nam je služilo kao plan tj. kostur aplikacije. U svakom trenutku smo znali što i kako trebamo promijeniti jer je sve bilo već smisljeno. Također, opisali smo arhitekturu i framework-ove te dodali sekvencijske dijagrame, obrasce uporabe te dijagrame razreda.

U drugoj fazi projekta fokus je bio na aplikaciji. Druga je faza također bila mnogo dinamičnija i intenzivnija faza jer smo gotovo cijelu aplikaciju sagradili zapravo u drugoj, kraćoj fazi. Nitko od članova grupe nije bio otprije upoznat s alatima tako da su svi bili primorani izdvojiti značajno vrijeme kako bi uopće mogli započeti projekt. Zatim, kako je projekt odmicao bilo je sve lakše jer smo se navikli na ponašanje spomenutih alata. Funkcionalnosti je bilo mnogo te je izgradnja nekih tekla očekivanim tokom, dok su neke trajale značajno dulje nego očekivano. Ipak, sve smo zadatke stigli ispuniti. Uz aplikaciju, postoji i značajan dio dokumentacije u drugoj fazi. Dodali smo testove programskog rješenja, opisali korištene tehnologije i alate te napisali upute za puštanje u pogon. Također dodali smo četiri dijagrama: dijagram stanja, dijagram komponenti, dijagram aktivnosti te dijagram razmještaja.

Komunikacija među članovima se održava putem WhatsAppa te povremeno na

Discordu. Na Discordu su bili virtualni sastanci dok je Whatsapp grupa služila za svakodnevno informiranje o napretku i zahtjevima projekta.

Sudjelovanje na ovom projektu je bilo vrlo intenzivno i dinamično. Bilo je vrlo naporno, ali i korisno jer smo napravili mnogo u malo vremena. Također, upoznali smo se s novim alatima koje ćemo sigurno ponovno koristiti. Još jedna dobra strana projekta je što smo se sami organizirali te smo mogli vidjeti kakve to prednosti i mane nosi.

Popis literature

1. Programsko inženjerstvo, FER ZEMRIS, <http://www.fer.hr/predmet/proinzh>
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>
7. Materijali za učenje Reacta:
<https://gitlab.com/jtomic/opp-project-teams-frontend/-/blob/master/education.md>
8. Materijali za učenje Spring Boota:
<https://gitlab.com/hrvojesimic/opp-project-teams-backend/-/blob/master/education.md>

Indeks slika i dijagrama

2.1	Walkzee – naslovna stranica	6
2.2	Walkzee – pregled pasa	6
3.1	Dijagram obrasca uporabe: Prikaz funkcionalnosti javnog posjetitelja te registriranog korisnika	21
3.2	Dijagram obrasca uporabe: Prikaz funkcionalnosti registriranog građanina i udruge	22
3.3	Sekvencijski dijagram: UC8 - Promjena osobnih podataka korisnika	24
3.4	Sekvencijski dijagram: UC10 - Dodavanje novog profila psa u listu prijavljene udruge	26
3.5	Sekvencijski dijagram: UC13 - Rezervacija termina šetnje	28
3.6	Sekvencijski dijagram: UC15 - Označavanje statistike šetnji kao javne	29
4.1	Apstraktna arhitektura sustava	31
4.2	Spring Boot arhitektura	33
4.3	Relacijski dijagram baze podataka	39
4.4	Razredi controlleri	40
4.5	Razredi modeli	41
4.6	Dijagram stanja - Šetač	42
4.7	Dijagram aktivnosti: Pregled kalendara šetnje	43
4.8	Dijagram komponenti	44
5.1	Ispitni primjer 1 - Početna stranica	61
5.2	Ispitni primjer 2 - Korisničko ime zauzeto	63
5.3	Ispitni primjer 2 - registracija šetača s neispravnim podatcima	64
5.4	3. ispitni primjer - Nepodudaranje lozinka	65
5.5	Ispitni primjer 3 - registracija udruge sa ispravnim podatcima	67
5.6	Registracija udruge sa zauzetim OIB-om	68
5.7	Ispitni primjer 4 - registracija udruge s neispravnim podatcima	69
5.8	Rezultati izvođenja ispitnih primjera - Selenium	70
5.9	Dijagram razmještaja	71

5.10 Generiranje .jar datoteke	72
5.11 Pokretanje naredbi za deploy aplikacije	74
5.12 Deployani frontend i backend	75
6.1 Dijagrami pregleda promjena	85

Dodatak: Prikaz aktivnosti grupe

Dnevnik sastajanja

1. sastanak

- Datum: 15. listopada 2020.
- Prisustvovali: L.Bokarica, A.Lukač, V. Sabalić, S. Almer, D. Lisica
- Teme sastanka:
 - specificiranje tehnologija za aplikaciju
 - specificiranje funkcionalnosti i zahtjeva aplikacije
 - entiteti baze podataka

2. sastanak

- Datum: 22. listopada 2020.
- Prisustvovali: L.Bokarica, V. Sabalić, D. Lisica, P. Presečki
- Teme sastanka:
 - pisanje backenda
 - rasprava o entitetima baze

3. sastanak

- Datum: 29. listopada 2020.
- Prisustvovali: L.Bokarica, V. Sabalić, S. Almer, D. Lisica, J. Juroš, P. Presečki, A. Lukač
- Teme sastanka:
 - pisanje backenda
 - pisanje frontend-a
 - sastanak sa asistenticom i usmjeravanje oko baze podataka
 - dogovor oko pisanja dokumentacije

4. sastanak

- Datum: 7. studenoga 2020.
- Prisustvovali: L.Bokarica, V. Sabalić, S. Almer, D. Lisica, J. Juroš, P. Presečki, A. Lukač
- Teme sastanka:

- pisanje backenda
- pisanje frontenda
- osposobljavanje login-a i signup-a korisnika
- dogovor oko pisanja dokumentacije

5. sastanak

- Datum: 12. prosinca 2020.
- Prisustvovali: L.Bokarica, V. Sabalić, S. Almer, D. Lisica, J. Juroš, P. Presečki, A. Lukač
- Teme sastanka:
 - pisanje backenda
 - pisanje frontenda
 - dodavanje novih funkcionalnosti na frontendu i backendu:
 - * dodavanje autorizacija
 - * slanje zahtjeva s frontend-a
 - * stranice profila pasa, šetača, udruga

6. sastanak

- Datum: 4. siječnja 2021.
- Prisustvovali: L.Bokarica, V. Sabalić, S. Almer, D. Lisica, J. Juroš, P. Presečki, A. Lukač
- Teme sastanka:
 - pisanje backenda
 - pisanje frontenda
 - dodavanje novih funkcionalnosti na frontendu i backendu:
 - * dodavanje pasa
 - * uređivanje pasa
 - * rezervacija šetnji
 - * prikaz statistika šetnji

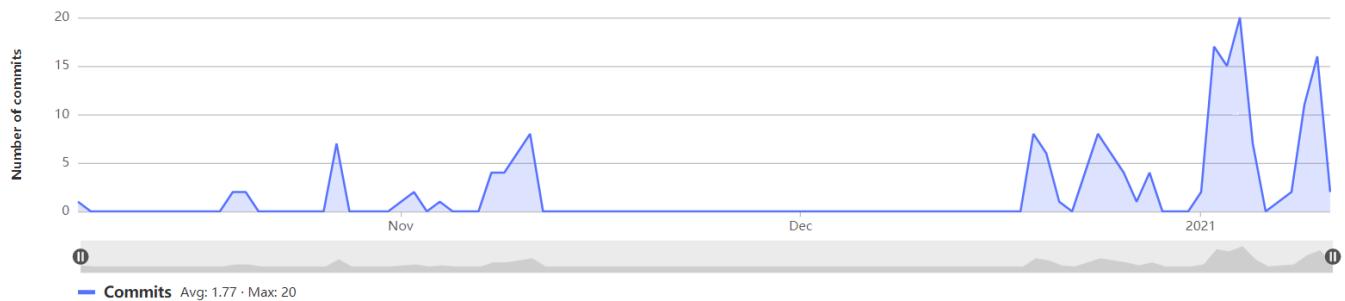
Tablica aktivnosti

	Jana Juroš	Anamarija Lukač	Petra Presečki	Luka Bokarica	David Lisica	Vito Sabalić	Sven Almer
Upravljanje projektom	6	2	0	3	2	1	1
Opis projektnog zadatka	3	3	0	2	2	0	1
Funkcionalni zahtjevi	1.5	3	0	1	1.5	0	1
Opis pojedinih obrazaca	3	4	0	0.5	2.5	0.5	0
Dijagram obrazaca	1.5	1	0	0	1	0	0
Sekvencijski dijagrami	3	1	0	0	0.5	0.5	0
Opis ostalih zahtjeva	0.5	0	0	0	1	0	0
Arhitektura i dizajn sustava	2.5	4	0	4	2	0	3
Baza podataka	0.5	7	0	8	7	0	4
Dijagram razreda	0	1	0	1	1	0	0.5
Dijagram stanja	0	0	0	0	0	0	0
Dijagram aktivnosti	0	0	0	0	0	0	0.5
Dijagram komponenti	0	0	0	0	0	0	0
Korištene tehnologije i alati	0	0	0	0	0	0	0
Ispitivanje programskog rješenja	12	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	0.5	0	0	0.5	1	0	0
Zaključak i budući rad	0.5	0	0	0	0	0	0
Popis literature	0.5	0	0	0	0	0	0
front end	30	1	14	1	1	14	0
izrada baze podataka	0	3	0	5	5	2	4
spajanje s bazom podataka	0	2	0	0.5	0	0	0
back end	4	25	0	40	30	0	15

Dijagrami pregleda promjena

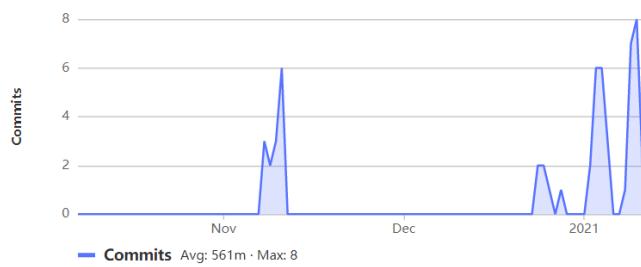
Commits to deploy

Excluding merge commits. Limited to 6,000 commits.



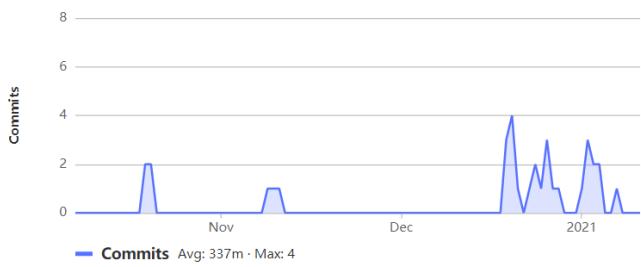
Jana Juroš

55 commits (jj51380@fer.hr)



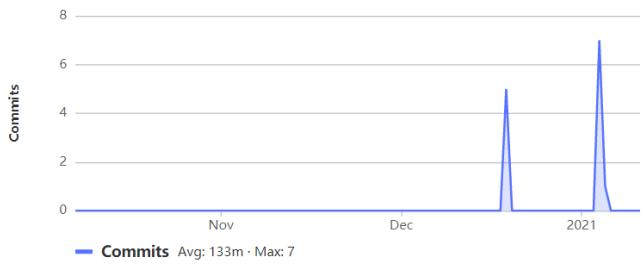
Luka Bokarica

33 commits (luka.bokarica1999@gmail.com)



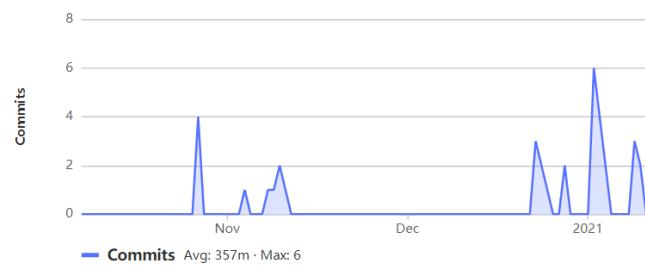
Anamarija

13 commits (al51487@fer.hr)



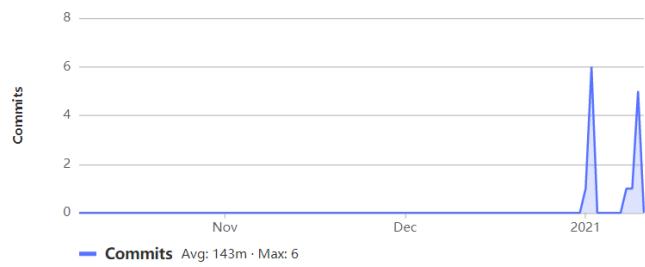
vitosabalic

35 commits (vito.sabalic@fer.hr)



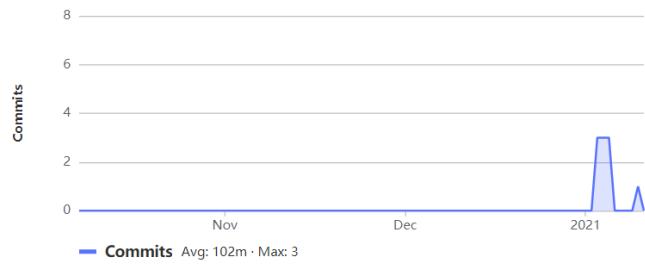
Luka Bokarica

14 commits (luka.bokarica@fer.hr)



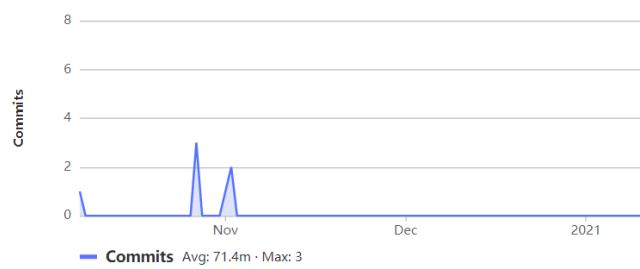
petra.presecki

10 commits (pp51719@fer.hr)

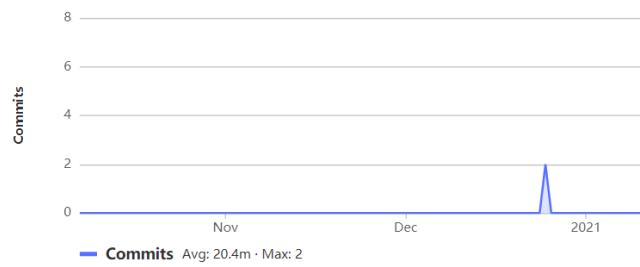


Jana Juroš

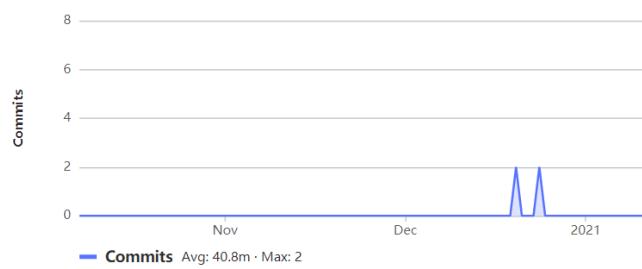
7 commits (jana.juros@fer.hr)

**svenalmer**

2 commits (sven.almer@fer.hr)

**dlišica**

4 commits (david.lisica@fer.hr)



Slika 6.1: Dijagrami pregleda promjena